UNIVERSIDAD DE PANAMÁ

VICERRECTORÍA DE INVESTIGACIÓN Y POSTGRADO

MAESTRÍA EN CIENCIAS EN INGENIERIA DE SISTEMAS DE COMUNICACIONES CON ÉNFASIS EN REDES DE DATOS

Computacion en la nube para dispositivos moviles en redes inalambricas (WLAN)

Rafael Asprilla

TESIS PRESENTADA COMO UNO DE LOS REQUISITOS PARA OBTENER EL GRADO DE MAESTRO EN CIENCIAS EN INGENIERÍA DE SISTEMAS DE COMUNICACIONES CON ÉNFASIS EN REDES DE DATOS

10 de diciembre de 2013

Agradecimientos

Quisiera aprovechar esta oportunidad para agradecer a la Secretaria Nacional de Cien cia y Tecnología (SENACyT) por apoyar a la Universidad de Panamá en la creación de este programa Maestría en Ciencias de la Ingeniería de Sistemas de Comunicaciones con énfasis en Redes de Datos que me ha permitido realizar mis estudios de maestría y mi formación como científico una meta importante en mi vida profesional En especial a Jane Saldana quien siempre se preocupó porque todas nuestras interrogantes fuesen contestadas y se aseguró de que cumpliésemos con nuestros compromisos con el programa

También me gustaría agradecer a la Universidad de Panamá en especial a los profesores Mgtr Gustavo Díaz por su gestión como coordinador del programa y al Dr Iván Armuelles por su asesoría además de su gestión como Director del Centro de Investigación en Tecnologías de Información y Comunicación donde pude realizar mis desarrollos y experimentos para culminar con este trabajo de tesis

A la Universidad Carlos III de Madrid por permitirme realizar mi estadía de investigación en sus instalaciones en la comunidad de Leganés Madrid En especial al Dr Florín Isalias por su asesoría durante este tiempo gracias a su guía y conocimientos pude establecer los requerimientos para mi experimento Al grupo de investigación ARCOS por su soporte en este proyecto

Índice general

List of Figures

Índice de Tablas

Resumen

La computación se encuentra hoy en la mayoría de las actividades humanas los dispositivos móviles han revolucionado la forma en que accedemos a la información volviéndola ubicua Aun así los dispositivos móviles están limitados por su capacidad de procesamiento y por su consumo energético Aplicaciones complejas no pueden ser ejecutadas a través de dispositivos móviles ya que son lentas y consumen mucha energía En este trabajo de investigación se propone una plataforma de desarrollo distribuida que permita a los dispositivos móviles aumentar su rendimiento haciendo uso de los recursos que pueden ofrecer las nubes computacionales También se plantea un tipo de nube computacional que permita el acceso local a estos recursos una nube computacional distribuida la cual permite reducir los costos de operación de la nube y su consumo energético haciendo de la computación en la nube una solución viable y de bajo costo para dispositivos móviles con un sólo nucleo de procesamiento poca memoria y almacenamiento limitado Los resultados experimentales demostraron que existe una posibilidad de aumentar el rendimiento de un dispositivo móvil utilizando un servidor con multiples unidades de procesamiento en aplicaciones complejas con gran cantidad de operaciones a ser ejecutadas o que requieran que se ejecuten muchas tareas a la vez en tiempo real Una vez completado el análisis de los resultados obtenidos mediante la experimentación se muestran algunas conclusiones y luego se proponen nuevos proyectos de investigación y desarrollo que complementen el trabajo realizado y permitan el nacimiento de una nube computacional distribuida que soporte la computación ubicua con dispositivos móviles

Summary

Computing is nowadays in most human activities mobile devices have changed the way we access information making it ubiquitous Even now mobile devices are limited by their processing capabilities and power consumption Complex applications cannot be executed on such devices due to slow performance and rapid power consumption A distributed development platform is proposed in this research work that will allow mobile devices to increase their performance using cloud computing resources Also a new type of cloud computer is depicted one that allows local access to computing resources a distributed cloud computer which reduces cloud s operational costs and power consumption making cloud computing a viable low cost solution for single core mobile devices with limited memory and storage Experimental results demonstrated that mobiles performance can be improved by using a server with multiple processing units in complex operation intensive applications or in real time multiple tasks applications Once data analysis is completed conclusions are presented then some future research and development work that complements the current project so a new distributed cloud computer that supports ubiquitous mobile devices can be created

Introducción

La computación se encuentra en la mayoría de las actividades humanas Actual mente la mayoría de los sistemas computacionales están enfocados en brindar servicios ubicuos a las personas permitiéndoles obtener información de manera inmediata en cualquier parte Algunas aplicaciones no obstante requieren de gran capacidad de procesamiento de datos y actualizarlos en tiempo real Los dispositivos móviles actuales sin embargo no han sido diseñados para brindar prestaciones de procesamiento complejo de datos esto impide que algunas aplicaciones de tiempo real se ejecuten dentro de celulares tabletas reproductores y consolas portátiles

Los dispositivos móviles no pueden realizar cálculos complejos debido a su limi tada capacidad de procesamiento resultado de las limitaciones de la tecnología de procesadores para móviles Un procesador para móviles debe hacer un compromiso entre su consumo energético y su poder de procesamiento Hasta ahora los procesadores que se utilizan en dispositivos móviles realizan tareas simples con pocas líneas de código y que requieren de pocas iteraciones para completar un trabajo Con el avance de las telecomunicaciones la creación de nuevos Sistemas Operativos para móviles y el aumento de los sensores como cámaras giroscopios y pantallas táctiles aparecen aplicaciones que procesan gran cantidad de datos Este tipo de programas requiere de mayor cantidad de memoria generalmente las aplicaciones tienen más líneas de códi go aumentando el tiempo de ejecución Esta situación exige nuevos procesadores que sean capaces de ejecutar las nuevas aplicaciones más rápido y con mayor capacidad para almacenar datos

Un problema causado por el incremento en el consumo de dispositivos móviles es la contaminación generada por la cantidad de aparatos electrónicos obsoletos descar tados que va en aumento Una vez que una nueva generación de dispositivos móviles es implantada en el mercado por los fabricantes desplaza completamente a los dispositivos de la generación anterior Esto no sólo implica que los dispositivos no están disponibles para la venta sino que cualquier servicio de reparación o piezas de reem plazo tampoco lo estarán En estas condiciones cualquier dispositivo móvil que sufra un daño en su tarjeta base debe ser inmediatamente reemplazado por un dispositivo de la nueva generación

En otro escenario existen fabricantes que venden dispositivos móviles a bajo costo con características limitadas Estos dispositivos probablemente no posean la ultima

tecnología pero funcionan con cierto grado de limitación Estas limitaciones convier ten a estos dispositivos en soluciones temporales a las necesidades de los usuarios En este proyecto de investigación se exploran las posibilidades de extender la vida util de los dispositivos móviles de generaciones anteriores o de bajo costo cuyas li mitaciones de procesamiento y almacenamiento pueden ser solventadas por una nube computacional que brinde servicios de ejecución remota

En particular este proyecto propone una nube computacional distribuida en un área geográfica amplia Esto supone una reducción de la latencia en la comunicación aumenta la ubicuidad del servicio distribuye el consumo energético en la red eléctrica, brinda redundancia protección contra fallos y reduce los costos de mantenimiento de la nube Hasta ahora las nubes computacionales se encuentran bajo el control de una sola entidad administrativa Están centralizadas dentro de *Datacenters* que se encuentran lejos de los usuarios incluso en otros países Esto impide en gran medida la ejecución de aplicaciones en tiempo real debido a que los requerimientos para este tipo de servicios están ligados a la latencia de la red y a la disponibilidad de recursos En este trabajo de investigación se propone una nube distribuida que permita a los usuarios acceder a los recursos de forma localizada evitando los inconvenientes de la latencia y la falta de recursos además de prestar el servicio de forma ubicua a través de una red de área amplia (WAN)

El objetivo general de este trabajo de investigación es

Evaluar la capacidad de mejorar el desempeno de una aplicación móvil a través de la utilización de recursos de una nube computacional

Los objetivos específicos que se plantearon para lograrlo son

Proponer el modelo básico del concepto de nube computacional distribuida

Definir los requerimientos de *hardware* y *software* para la implementación del *cloudlet*

Crear una maqueta de pruebas que emule un dispositivo básico de la nube computacional (*cloudlet*)

- Crear una aplicación móvil distribuida que nos permita realizar pruebas y medir el rendimiento en tiempo de ejecución

Recopilar y evaluar los datos experimentales del rendimiento de la aplicación local contra la aplicación distribuida

Una vez alcanzados estos objetivos la data obtenida arrojó resultados que demuestran la posibilidad de aumentar el rendimiento de las aplicaciones para móviles con una nube computacional Aun cuando el procedimiento utilizado no fue optimi zado los resultados obtenidos demuestran que el rendimiento en tiempo de ejecución

puede reducirse considerablemente En futuros proyectos relacionados se trabajará en la optimización del código para aumentar el rendimiento y reducir el consumo energético ya que las aplicaciones utilizan intensivamente el servicio de comunicación inalámbrica principal consumidor energético dentro del móvil Algunas propuestas se presentan a lo largo de este documento que podrían ayudar a mejorar este aspecto del sistema También al finalizar se muestran otros proyectos que se se derivan de este trabajo *como* el establecimiento de un protocolo de actualización de los recursos de la nube distribuida, la creación de prototipos de *cloudlets* utilizando tarjetas de desarrollo miniatura y creación de las piezas de código genéricas que permitan a los desarrolladores móviles crear aplicaciones móviles distribuidas con piezas de código que se ejecuten en los *cloudlets*

Este documento dividido de la siguiente manera en el capítulo I se plantean los antecedentes y el sustento teórico de este proyecto y se realizan algunas propuestas a partir de ellos en el capítulo II se muestran los requerimientos de la plataforma desarrollada, en el capítulo III se muestra la metodología utilizada y las especificaciones de la plataforma experimental luego en el capítulo IV se presentan los resultados y el análisis de los datos experimentales obtenidos con la plataforma de pruebas al final se presentan las conclusiones en el capítulo V y las propuestas de investigación derivadas de este trabajo en el capítulo VI La documentación de los códigos fuentes y las tablas de datos obtenidas se muestran en la sección de anexos

Capítulo I

Estado del Arte

La computación hoy se encuentra presente en la mayoría de las actividades humanas los sistemas de comunicaciones están en su mayoría enfocados en interconectar computadores alrededor del mundo La computación ha evolucionado de los grandes computadores de tipo *mainframe* a los computadores personales luego a las redes de computadoras los clusteres y mallas computacionales hasta los computadores portátiles *handheld* y móviles Las comunicaciones en este mismo sentido han evolucionado con la tecnología actualmente se empiezan a considerar nuevos paradigmas de computación como lo es la computación ubicua y la realidad aumentada que permitirán que el ser humano comun pueda acceder a toda la información que necesita al momento Este tipo de tecnología requiere de gran poder de procesamiento sumado a una gran movilidad estos conceptos son totalmente opuestos ya que para obtener gran capacidad de procesamiento se requiere de mucha energía mientras que para obtener movilidad se requiere de utilizar eficientemente la energía Dicho de otro modo las restricciones de energía de las tecnologías móviles son impedimentos para el aumento de la capacidad de procesamiento de los computadores Los dispositivos móviles en la actualidad poseen muy buenas especificaciones pero están restringidos por la duración de la batería lo cual reduce la movilidad o están sujetos a trabajar con el

mínimo de recursos lo que impide que aplicaciones avanzadas como la realidad aumentada aplicaciones interactivas con redes de sensores o los videojuegos de realidad alternativa puedan ser ejecutados eficientemente

Hasta el momento los fabricantes de dispositivos móviles han trabajado arduamente en la creación y mejoramiento de procesadores más eficientes con muy bajos niveles de consumo energético que operan a voltajes muy bajos y frecuencias altas Recientemente se empezaron a utilizar multiples nucleos para realizar procesamiento concurrente con el fin de aumentar la complejidad de las aplicaciones disponibles para dispositivos móviles Esta tendencia sin embargo está limitada por la naturaleza del comportamiento de los elementos que la constituyen problemas de disipación de temperatura, susceptibilidad al ruido alto consumo energético y durabilidad En especial la pérdida de energía por disipación de calor es una de las principales limitantes de la tecnología móvil actualmente esto es debido a que es incómodo para el usuario que su móvil opere a temperaturas elevadas y que parte de la carga de la batería se convierta en calor limitando el tiempo de operación independiente del móvil

Existen algunas alternativas a la computación móvil de alto rendimiento que podrían eliminar la necesidad de crear dispositivos más eficientes y costosos simplemente con la utilización de entornos cooperativos donde una combinación de poder de procesamiento en un sistema fijo ayude a los móviles a realizar tareas que hasta el momento son costosas en tiempo de ejecución y consumo energético La computación de alto rendimiento nubes computacionales clusteres y mallas computacionales son sujetos de estudio para la creación de aplicaciones interconectadas que permitan la creación de aplicaciones complejas

En este capítulo se muestran algunas tecnologías que sirven como antecedentes del sistema propuesto en este trabajo de investigación Se muestra también una breve referencia de cada una de las tecnologías utilizadas dentro de la maqueta de pruebas

1 1 Antecedentes

1 1 1 Escritorios remotos y visualización remota

En un entorno de red de virtualización donde varios servidores se encargan de diferentes terminales que son accedidas a través de una red LAN o WAN por multiples usuarios se hace necesario un protocolo de control remoto de aplicaciones eficiente y ligero Existen algunos protocolos que se encargan de enviar y recibir información de los distintos periféricos comunes en todos los computadores teclado ratón y monitor Estos protocolos se utilizan actualmente para diferentes actividades como el control remoto de terminales (ej *Remote device control application protocol* ITU T T 136 ITU T (1999)) y otros además sirven para autenticación y control (ej kdm y gdm)

Tecnologías de control de terminales virtuales como *Remote Desktop Protocol*(RDP) de Microsoft y Citrix ICA utilizan un esquema de bajo nivel con compresión de datos de vídeo (Lai and Nieh 2006) Este esquema consiste en enviar la información de los píxeles en la pantalla comprimidos en un flujo de datos La compresión generalmente consiste en enviar sólo los cambios en la información de vídeo En otros sistemas se utilizan nemónicos de vídeo como comandos para reconstruir los gráficos en el lado del cliente estos requieren de un analizador de imágenes del lado del servidor para hacer la traducción Entre los sistemas que utilizan primitivas de vídeo tenemos a AT&T VNC y Oracle SUN Ray (Lai and Nieh 2006)

Algunos trabajos relacionados presentan diferentes capas de abstracción para la visualización remota tal es el caso de Baratto et al (2004) quien define por lo menos cuatro (4) niveles de abstracción para visualización remota Este trabajo pretende mantener la integridad del software que se ejecuta para evadir la necesidad de desarrollar nuevo software que se adapte al sistema La capa más alta de abstracción es la que captura los comandos de vídeo que la aplicación pasa al sistema operativo

virtualizado Este método de abstracción no se puede realizar sin cambios al software o al sistema operativo que se va a implementar

El siguiente nivel se encuentra en el *middleware* o capa de virtualización la cual es independiente del *hardware* en el servidor En esta capa se pueden capturar los comandos provenientes de la máquina virtual y enviarlos al cliente ligero Es posible que esta sea la capa de abstracción ideal para realizar la captura de los datos de vídeo pero su principal desventaja es su complejidad y la velocidad en que su código se vuelve obsoleto debido a las actualizaciones En cualquier caso el crear una capa *middleware* que trabaje directamente con un esquema de visualización remota es muy interesante más si se está dispuesto a crear nuevas aplicaciones y modificar sistemas operativos para este fin

La tercera capa de abstracción es a nivel de hardware en lo que conocemos como *framebuffer* El *framebuffer* es la memoria de datos que aparecen en pantalla o píxeles Este esquema es muy simple de utilizar pero requiere compresión de datos y genera cierto retardo ya que es la ultima capa de abstracción del vídeo en el esquema de virtualización de terminales Baratto et al (2004) plantea entonces la necesidad de una capa intermedia de abstracción Esta cuarta capa de abstracción se encuentra por encima del nivel de hardware y por debajo de la de *middleware* justo donde se encuentra la capa de abstracción de hardware (HAL) que corresponde a los con troladores de vídeo Al incluir un nuevo controlador de vídeo se pueden obtener los comandos de primitivas 2D y transmitirlos directamente a la red

La tecnología actual se basa en los dos métodos del nivel más bajo de abstracción de *hardware* el *framebuffer* y las primitivas 2D Las soluciones de escritorios remotos son generalmente propietarias e incompatibles entre sí En el ano 2009 la *Video Electronics Standards Association* (VESA) promulgó un estándar para la fabricación de clientes ligeros conocido como *Net2display* (Court 2009) en éste se describen las

bases para diseñar dispositivos de cliente ligeros que sean compatibles con todos los sistemas de virtualización de tal forma que el hardware y el software fueran realmente independientes. Este estándar define sólo la estructura del software y la arquitectura del sistema de virtualización estándar, no los algoritmos ni el hardware que debe ser utilizado para el diseño. De esta manera los fabricantes pueden realizar las mejoras a sus sistemas sin encasillar a sus usuarios en una sola tecnología permitiéndoles mejorar sus redes con el tiempo.

1.1.2 Tecnologías de virtualización de pantalla

La virtualización es la emulación de los recursos de un computador, sean estos físicos o no, dentro de otro computador más potente o un conjunto de ellos. Aunque la virtualización se conoce a partir de la década de 1960, no fue hasta la década de 1990 donde la virtualización adquiere un auge importante. Antes de eso sólo se utilizaba en recursos determinados, por ejemplo: las particiones de los discos duros, el multiprocesamiento y la memoria virtual. Con el aumento exponencial de las capacidades de los procesadores se crea software que permite emular una computadora completa dentro de un sistema operativo común (ver Figura 1.1). Estos programas permiten realizar pruebas de otros sistemas operativos para entrenamiento o simplemente para evaluación.

Figure 1.1: Arquitectura de una máquina virtual

En la actualidad los programas de virtualización pueden asignar recursos a diferentes máquinas virtuales corriendo dentro de un solo computador, esto permite la

creación de redes virtualizadas donde se prueban nuevas topologías de red y se pueden crear dispositivos virtuales para realizar pruebas Por otra parte las redes de virtualización se han vuelto muy populares en el sector privado puesto que poseen muchas ventajas que les permiten a las empresas reducir costos y aumentar la productividad Al centralizar la administración de la red se aumenta el control y se reduce la carga de trabajo del departamento de tecnologías de información (IT) permitiéndole enfocarse en tareas diferentes a la mera reparación de computadoras

El éxito de la virtualización de terminales se debe principalmente a las tecnologías de escritorio remoto Cuando se poseen multiples máquinas virtuales ejecutándose en un servidor como se muestra en la Figura 1 2 se hace necesario contar una manera sencilla y fácil de acceder a todas ellas para manejar sus entornos gráficos Esto se logra con la utilización de escritorios remotos o visualización remota con esta tecnología se es capaz de manejar varias máquinas de manera centralizada y remota La mayoría de las tecnologías de escritorio remoto utilizan protocolos de tiempo real para llevar información de las imágenes en pantalla y las acciones del usuario sobre el teclado el ratón o cualquier otro dispositivo de entrada que los usuarios utilicen para controlar sus máquinas virtuales

1 1 3 Tecnologías de Escritorio Remoto

Entre las tecnologías de visualización remota existentes haremos mención de las más actuales y comunes en ambientes comerciales Las tecnologías de escritorio remoto que utilizan diferentes niveles de abstracción de vídeo se compilan en Lai and Nieh (2006) donde se muestra una tabla de las tecnologías de visualización remota más comunes Ellos realizan una comparación de cada una de las tecnologías de escritorios remotos así como una descripción básica de las posibilidades de cada una de ellas En esta sección estaremos analizando sus ventajas y desventajas para aplicaciones con

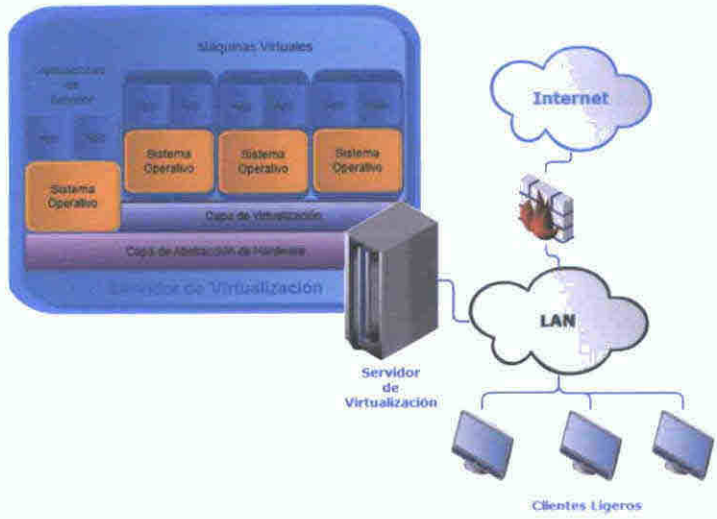


Figure 1.2: Arquitectura de un servidor de virtualización en la actualidad

dispositivos móviles, ya que la mayoría fueron diseñadas para entornos de virtualización con clientes ligeros en redes de área local utilizando cableado estructurado. El entorno de una LAN cableada no es consecuente con la movilidad de dispositivos como celulares, PDA o tabletas; se hace entonces necesario utilizar una red de cobertura más amplia y completamente inalámbrica.

En Simoens et al. (2011), Zeng and Lu (2011) e Imai et al. (2010) se presentan soluciones que utilizan VNC como una alternativa para la visualización remota en dispositivos móviles. VNC, con su protocolo RFB (Richardson, 2009), se ha adaptado a dispositivos móviles que utilizan interfaces inalámbricas y poco ancho de banda. Este protocolo utiliza primitivas 2D que permiten reconstruir la imagen del escritorio remoto en un celular o PDA; más recientemente se utiliza para controlar computadores dentro de redes de área local inalámbricas utilizando aplicaciones como *TeamViewer* para Android. Aunque es posible la visualización remota utilizando *TeamViewer*, se hace muy incómodo el control de pantallas con mayor resolución a la pantalla del dispositivo móvil. Esto se debe a que se requiere dimensionar la pantalla constantemente, para ir de un lado a otro.

Algunas investigaciones plantean crear aplicaciones con tamanos de pantalla apropiados para los dispositivos móviles El propósito de este trabajo es el mismo la diferencia se encuentra en el método que se utiliza La idea de realizar este trabajo es presentar un método diferente de visualización remota de aplicaciones que sea liviano y trabaje en tiempo real Una de las principales desventajas de VNC es que requiere que los eventos generados por el cliente ligero sean reportados al servidor antes de que cualquier acción sea ejecutada Esto implica que las respuestas se hacen en lo que demora un viaje ida y vuelta a través de la red Es posible que este tiempo sea muy elevado cuando hablamos de redes inalámbricas en áreas amplias y con gran cantidad de abonados Al respecto en Lai and Nieh (2006) se indica que los sistemas de visualización remota deben crearse para corregir los efectos de la latencia en la red más que la necesidad de ancho de banda

La necesidad de un nuevo sistema radica en solventar estos posibles inconvenientes mediante la separación de tareas por ejemplo los efectos visuales y las interfaces de entrada pueden ser manejadas por los dispositivos móviles mientras que las tareas relacionadas a las aplicaciones son reportadas al servidor para su ejecución y luego se envían las respuestas de vuelta al dispositivo móvil para la presentación de resultados El estándar *Net2Display* plantea que clientes ligeros manejen los eventos de forma local lo cual indica que los nuevos sistemas móviles deberían manejar los eventos locales de la misma manera Esto permite dar una respuesta intermedia a la petición del usuario con la finalidad de acusar el recibo de una instrucción dada

En redes de área amplia inalámbricas y con un tráfico comparable al de una LAN inalámbrica regular es posible que se requieran algunas consideraciones especiales como la latencia y los efectos de la movilidad Las tecnologías de escritorio remoto actuales no fueron disenadas para este tipo de redes Generalmente las redes de virtualización han sido para redes locales debido a su estrecha relación con la reducción

de costos y el aumento de la productividad en las empresas En redes de virtualización para dispositivos móviles se deben tomar en cuenta algunas limitaciones de los dispositivos móviles como el poder de procesamiento y la duración de la batería estas consideraciones fueron expuestas por Simoens et al (2011) En el presente trabajo se atenderán las limitaciones en redes inalámbricas móviles para aplicaciones de virtualización

1 1 4 Protocolos de Escritorios Remotos

Los protocolos de escritorios remotos hacen posible monitoreo de máquinas virtuales mediante la visualización remota Los protocolos que se presentan a continuación están ligados a las capas de abstracción propuestas en la sección 1 1 1 Generalmente están situados entre las capas de *hardware* y *middleware* En Lai and Nieh (2006) se presentan los protocolos de escritorios remotos más comunes en el mercado para el ano 2006 Ellos crearon una herramienta de medición de rendimiento de los protocolos donde sorprendentemente resultó que los protocolos de visualización remota a nivel de píxeles eran más eficientes que sus contra partes que utilizan primitivas 2D o protocolos a nivel de software para redibujar la imagen en la pantalla del cliente ligero

El estándar *Net2display* (Ocheltree et al 2009) define una estructura para los protocolos a nivel de aplicación para unificar las tareas que requieren atención en un esquema de escritorio remoto utilizando clientes ligeros Dentro de la información de los paquetes de *Net2display* se encuentran los identificadores para los canales virtuales (ver Figura 1 3) que transmiten la información de audio vídeo y de los dispositivos USB conectados al cliente ligero Estos identificadores permiten multiplexar los datos emitidos o recibidos por los diferentes canales virtuales para atender las peticiones en paralelo Al identificar a que flujo de datos pertenece cada paquete es posible manejar

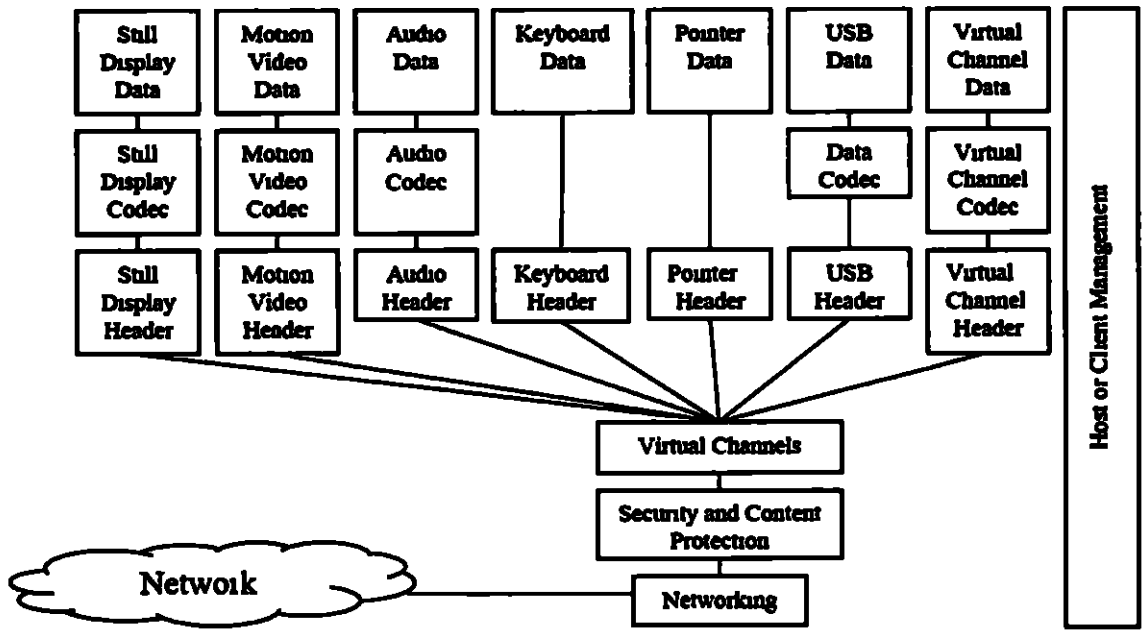


Figura 1 3 Arq itect ra sugerid por *N t2displ y* (Ocheltree et al 2009)

varias pantallas o varios dispositivos de entrada o salida Aun cuando *Net2display* no fue creado para trabajar con dispositivos móviles la arquitectura es completamente válida aunque es posible que haya que realizar algunas modificaciones

En el caso de VNC que es el sistema de visualización remota más utilizado para investigación y desarrollo en dispositivos móviles la información se transmite sólo a través de primitivas rectangulares (Richardson 2009) Este protocolo utiliza esta primitiva para enviar información sobre el *framebuffer* al cliente Al utilizar sólo rectángulos la etapa de análisis de los datos en pantalla se vuelve muy sencilla mas no así la reconstrucción Este protocolo utiliza varios tipos de compresión para enviar los datos de los píxeles en forma de primitivas 2D En cualquier caso la forma de trabajo del protocolo RFB no permite mantener un control local de los objetos esto indica la necesidad de que el servidor realice todas las operaciones en una aplicación incluso la actualización de la pantalla

Otro protocolo propietario del cual se puede obtener poca información sobre su

funcionamiento es SUN Ray de Oracle Este protocolo utiliza primitivas 2D para recrear la pantalla en sus clientes ligeros Una característica adicional de este sistema es su medición del tráfico en la red Este monitor de red permite a los servidores de virtualización saber cuanto ancho de banda hay disponible así se controla el flujo de datos a las necesidades de la red El control de flujo evita congestiones y aumenta la eficiencia porque no hay colisiones ni paquetes perdidos La seguridad del sistema es su control de acceso a las máquinas virtuales el cual se realiza con *Smart Cards* o tarjetas inteligentes las máquinas virtuales de los usuarios residen en el servidor y el software para presentación de datos es la unica aplicación que se encuentra instalada en la memoria del cliente ligero

1 1 5 Clientes pesados, ligeros y ultra-ligeros

En los sistemas de visualización remota que se han presentado hasta el momento la mayoría de los clientes poseen poca información de la ejecución de la máquina virtual que se ejecuta en el servidor Estos dispositivos se les conoce como *clientes ligeros* En los casos donde la transición a clientes ligeros no puede realizarse de inmediato se utilizan *clientes pesados* y en el caso de sistemas como SUNRay existen los *clientes ultra ligeros* o *clientes cero* El tipo de cliente en un sistema de virtualización dado está definido por las características del hardware y el tipo de software que se almacena en él

Por lo general se consideran *clientes pesados* a computadoras con especificaciones fuera de tiempo o económicas con suficientes recursos para ejecutar un sistema operativo básico y un mínimo de aplicaciones de productividad e Internet Estas son utilizadas para proyectos de virtualización donde la posibilidad de reemplazarlas de inmediato por *clientes ligeros* no es posible Generalmente estos sistemas tienen la capacidad de bajar desde el servidor un kernel para ejecutar las aplicaciones y luego

se sincronizan con la máquina virtual residente en el servidor de virtualización Como poseen suficientes recursos para manejar gráficos tampoco se requieren soluciones de escritorio remoto estos son opcionales

Un cliente ligero es un dispositivo estándar en una arquitectura de virtualización Los recursos de un cliente ligero son los mínimos suficientes para desplegar los datos en pantalla y manejar los dispositivos de entrada y salida En cuanto a software éstos poseen varios niveles de ejecución algunos ejecutan un kernel para las funciones básicas y sólo reciben actualizaciones desde el servidor otros pueden no poseer más que la capa de abstracción de hardware (HAL) y el servidor se encarga incluso de los controladores del hardware en el cliente a estos se les llama *clientes ultra ligeros* (SUN Ray) o *clientes cero* (Zero Clients)

Un dispositivo móvil actual posee un *firmware* que utiliza para inicializar y controlar su hardware este debe ejecutarse antes de pasar el control al sistema operativo que está instalado en el dispositivo móvil En algunos casos un kernel de Linux actua como *firmware* en el dispositivo móvil es este el caso de los dispositivos que utilizan sistema operativo Android de Google o Qtopia de Nokia En estos casos se puede considerar al dispositivo móvil como un *cliente ligero* ya que posee un software que controla su hardware y algunas funciones del sistema operativo

1 2 Computacion en la Nube

Los avances tecnológicos en la computación paralela y los sistemas distribuidos han permitido la creación de nubes de servidores para ejecutar software como si todos sus recursos computacionales se encontraran en el mismo computador Los clusteres de computadora permiten que una aplicación se pueda ejecutar en varias máquinas físicas a la vez y que muchos usuarios puedan acceder a una aplicación a la vez La finalidad de todos estos esfuerzos es generar un servicio masivo que esté disponible

todo el tiempo La computación en la nube es el resultado del desarrollo producto de la investigación que patrocinan grandes empresas con el fin de ofrecer servicios de virtualización de hardware y software En Li and Deng (2011) se presentan los conceptos básicos de la computación en la nube mientras en Bernstein et al (2011) se presentan algunas condiciones para que un cluster distribuido pueda ser considerado un sistema de computación en la nube

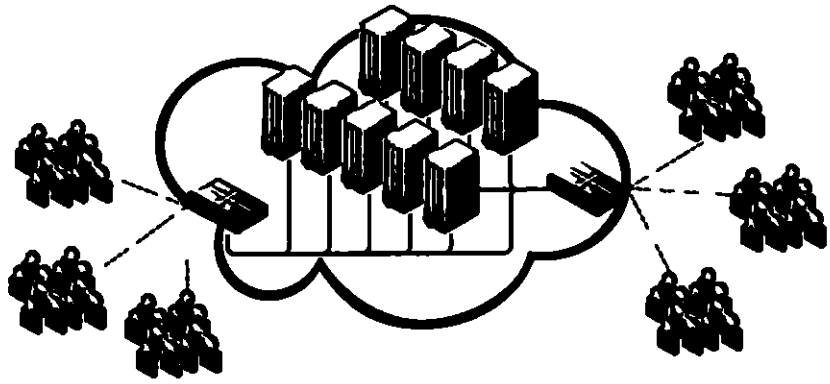


Figure 1.4 Concepto general de las nubes computacionales

El concepto general propone que una nube de computación debe ser una gran conjunto de clusteres reunidos en unos cuantos *datacenters* localizados en lugares estrategicos tal y como lo muestra la Figura 1.4 Esta topología presenta algunos problemas el primero es el cuello de botella que se genera por el tráfico de todos los abonados a la nube y la latencia de la red que aumenta con la distancia al *datacenter* Este es un gran inconveniente para servicios en tiempo real como virtualización de terminales y visualización remota aun cuando existan dos o tres *datacenters* que permitan servicios de alta disponibilidad y alto rendimiento El problema no se encuentra en la capacidad de ejecutar instrucciones de manera casi instantánea mediante computación paralela el tema es la latencia de la red de área amplia que transporta todos los datos de ida y vuelta Aun con etiquetas de QoS la latencia del sistema es la suma del tiempo que pasa cada paquete en cada equipo en la red antes de llegar al

al destino Lo cual indica que la latencia será más baja al tener menos saltos para llegar a los extremos de la conexión En este caso una red que posea pocos saltos a pesar de cubrir una gran distancia será más eficiente que una que esté cerca pero posea muchos equipos interconectados para brindar el servicio

1 2 1 Tipos de servicios de las Nubes Computacionales

La computación en la nube ofrece diferentes servicios que se prestan de diferentes maneras segun las necesidades de los usuarios Segun Li and Deng (2011) y Zhang et al (2010) las infraestructura básica de las nubes computacionales está disenada para convertir un grupo de computadores en un conjunto de recursos computacionales que pueden ser asignados de forma dinámica El primer servicio que los clusteres comerciales ofrecen es la infraestructura como un servicio (IaaS *Infraestructure as a Service*) El servicio IaaS permite la virtualización de *hardware* ejemplo de esto es el almacenamiento de red el procesamiento paralelo y las máquinas virtuales Pero esto requiere de la instalación de software y su debido mantenimiento

Un segundo nivel de servicio es la plataforma de desarrollo como un servicio (PaaS *Platform as a Service*) En este caso se le permite al usuario crear aplicaciones utilizando un conjunto de APIs que acceden a los recursos de la nube de forma segura y transparente para el usuario Este esquema de servicio permite personalizar las aplicaciones y a la vez mantiene el *software* actualizado delegando parte del mantenimiento al proveedor de servicio La forma en que la aplicación se maneja en la nube no concierne al usuario esto permite a los desarrolladores concentrarse en la aplicación y no en el funcionamiento del sistema en el cual se encuentra Más adelante en este trabajo se propone una plataforma de desarrollo como un servicio que asigna los recursos de la nube y planea la ejecución de trabajos distribuidos a unidades de procesamiento particulares

Un tercer nivel de abstracción lo ofrecen las aplicaciones como un servicio (SaaS *Software as a Service*) Son las más comunes debido a que son ofrecidas por los grandes proveedores de servicios y creadores de software como Google Microsoft y Amazon Las aplicaciones que estos proveedores de nubes computacionales ofrecen son basadas en tecnología Web generalmente son ligeras poco complejas (no ofrecen muchas funcionalidades) o en algunos casos hay que pagar para obtener mejores funciones en la nube Son muy utilizadas en dispositivos móviles pero requieren que el desarrollador las adapte a las pequenas pantallas de los móviles y generalmente su operación no es en tiempo real debido a que los paquetes de datos se envían todos en momentos definidos al móvil para ahorrar energía y recursos computacionales

1 2 2 Computación distribuida

Uno de los paradigmas de la computación concurrente es la distribución de las tareas a través de diferentes procesos que pueden o no ejecutarse en paralelo Lo ideal por supuesto es la ejecución en paralelo de multiples tareas pero esto sólo puede lograrse mediante la presencia física de multiples unidades de proceso En los primeros sistemas operativos de computadores personales (PC) se utilizaba mucho el concepto de multiprocesamiento porque se podían ejecutar varias aplicaciones a la vez esto era posible mediante la división de las aplicaciones en tareas y su ejecución multi plexada en el tiempo a gran velocidad Este concepto se ponía a prueba cuando por alguna razón las tareas no se podían completar lo suficientemente rápido La solución fue aumentar la frecuencia de operación ya que así se pueden ejecutar más instruc ciones a la vez pero esto también tuvo un límite debido al sobrecalentamiento de los procesadores

El sobrecalentamiento supone algunos problemas como un consumo elevado de energía que se pierde en forma de calor la utilización de sistemas de enfriamiento

cada vez más costosos y el corto tiempo de vida de los procesadores Gracias a las tecnologías de miniaturización de los circuitos integrados se pueden incluir en el mismo sustrato varios nucleos de procesamiento que comparten los recursos de memoria y puertos de entrada/salida permitiendo ejecutar más de una instrucción al mismo tiempo Desde este momento la computación distribuida que hasta el momento ha sido sólo para aplicaciones científicas complejas adquiere una importancia relevante para los usuarios regulares de computadores Entre las tecnologías para servidores donde la redundancia es muy importante se pueden contar algunos modelos que poseían tarjetas madres con capacidad para dos o más procesadores Estos utilizaban los mismos recursos del servidor pero no directamente como lo hacen dentro del mismo circuito integrado los procesadores actuales Esto demuestra de cualquier manera la capacidad de realizar tareas en paralelo mediante la utilización de multiples microprocesadores pero para este momento en la historia esta tecnología era costosa y no se incluía en los computadores personales

Desde el punto de vista del *software* la computación distribuida es la división de las tareas o procesos que se realizan en diferentes unidades de proceso Generalmente se utilizan multiples nodos de cómputo pertenecientes a redes de datos como es el caso de los clusteres y nubes de cómputo Los clusteres de computadoras funcionan como una entidad a pesar de poseer características que les permiten trabajar de forma independiente poseen un nodo principal donde se ejecuta un programa planificador un servicio de recepción de peticiones y un evaluador de desempeno En los nodos de cómputo se ejecuta un servicio de recepción de tareas donde se ejecutan en orden de llegada o prioridad Las aplicaciones pueden utilizar muchos segmentos de código o una aplicación que envía pedazos de código a los nodos de cómputo En este proyecto estaremos trabajando con diferentes programas que se hacen cargo de tareas específicas que de ahora en adelante llamaremos piezas de código cada programa

posee un código sencillo con un grupo de entradas y salidas estándar generalmente desde y hacia un archivo

En el planificador se reciben peticiones de operaciones mediante archivos XML que se convierten luego en archivos de instrucciones que se ejecutan en los nodos de cómputo utilizando las piezas de código Todas las aplicaciones deben partir de este mismo concepto para que se cree lo que se conoce como una base de código confiable (TCB *Trusted Code Base*) La TCB está compuesta de las piezas de código que son comunes para todas las aplicaciones realizan la misma tarea con el mismo tipo de entradas y salidas Así las aplicaciones se desarrollan de cierta forma como si se armase un rompecabezas facilitando el trabajo de desarrollo Las piezas de código residen en el nodo principal y pueden ser replicadas en los nodos de cómputo cuando son necesarias y luego descartadas cuando ya no sean necesarias esto se conoce como Código bajo Demanda (CoD *Code on Demand*) Algunas piezas de código pueden pasarse al móvil de ser necesarias pero esta práctica debe evitarse al máximo por razones de seguridad

1 2 3 La Ley de Amdahl y las Reglas de Pollack

Una de las limitaciones que se pueden encontrar al crear sistemas distribuidos y que afectan la cantidad de unidades de procesamiento que utilizamos para una aplicación dada es la Ley de Amdahl Esta ley indica que si la velocidad de procesamiento de una porción de código de un proceso f puede ser mejorada por un factor m y la otra porción no puede ser mejorada entonces la porción que no puede ser mejorada dominará el rendimiento y el mejoramiento de la porción de código que puede ser mejorada

$$A_{Amdahl} = \frac{1}{(1-f) + \frac{f}{m}} \tag{1 1}$$

En la ecuación 1 1 se muestra la relación entre la aceleración en tiempo de ejecución de un proceso con la cantidad de procesadores En el caso de que m aumente a infinito el tiempo de ejecución estará dado por la relación $A_{Amdahl} = \frac{1}{(1-f)}$ que corresponde a la sección de código que no puede ser paralelizada Esta dominará el tiempo mínimo de ejecución del proceso Esto se debe a que el procesador de un sólo nucleo utiliza una técnica de división de tiempo para realizar las operaciones de cada uno de los hilos que se generan para trabajo concurrente simulando una ejecución en paralelo En el caso de sistemas con multiples nucleos y procesadores la Ley de Amdahl debe adaptarse tal y como se muestra en Sun and Chen (2010) Ellos expresan también la posibilidad de utilizar nucleos de procesamiento menos complejos debido a que el consumo de potencia de los procesadores es directamente proporcional a su consumo energético lo cual corresponde a una de las Reglas de Pollack vista de forma inversa La disminución en el rendimiento de los nucleos de procesamiento debido a la reduc ción en la complejidad otra de las Reglas de Pollack vista también de forma inversa puede ser compensado por el aumento en el numero de nucleos

Aunque la Ley de Amdahl se impone en el tiempo de ejecución sobre la arqui tectura de multiples nucleos y los fabricantes utilizan esto como excusa para no incluir más nucleos de procesamiento en sus chips Borkar (2007) comenta que en una computadora personal los usuarios tienden a utilizar multiples aplicaciones que a su vez utilizan multiples hilos esto indica que si varias aplicaciones utilizan varios nucleos a la vez se necesitarán más nucleos para acomodar a todas las aplicaciones dentro de un procesador con multiples nucleos Aun más si extendemos este concepto a las nubes computacionales podemos tener varios escenarios de servicio a multiples usuarios y multiples aplicaciones con diferentes requerimientos de recursos Ciertamente si se atienden a multiples usuarios con la misma aplicación será fácil predecir el comportamiento de la reserva de recursos en todo momento mientras que con

multiples aplicaciones sirviendo a multiples usuarios se debe crear un planificador de tareas que sea capaz de reservar recursos de manera eficiente dentro de la nube En especial si la nube se encuentra distribuida a través de una zona geográfica extendida el reto está en crear un protocolo de asignación de recursos en tiempo real capaz de reservar en diferentes nodos de la nube En el esquema presentado por Satyanarayanan and Bahl (2009) donde la nube está conformada por un conjunto de nodos independientes la planificación de tareas requiere de una función que tome en cuenta diferentes variables como el tiempo en que llega la petición la cantidad de recursos disponibles la cantidad de recursos requeridos la movilidad del usuario (en el caso de dispositivos móviles) la distribución de los nodos la cobertura de los diferentes nodos la cantidad de usuarios en cada nodo la ley de Amdahl la complejidad del algoritmo etc

1 2 4 Planificador de tareas

El planificador de tareas (*Job Scheduler*) es la pieza clave de la computación distribuida. Esta aplicación es la encargada de separar los trabajos en tareas intermedias que serán ejecutadas en paralelo y luego asignarlas a la pila de espera de cada procesador para que sean ejecutadas lo antes posible En el caso de las nubes clusteres y mallas computacionales este servicio es de suma importancia es el encargado de ordenar las tareas que serán realizadas para el mejor aprovechamiento de los recur sos El servicio de planificador de tareas decide cuantos recursos deben ser dispuestos para ejecutar una tarea basándose en los requerimientos de la aplicación los recursos disponibles y el tiempo de entrega solicitado

En la actualidad los servicios de planificación de tareas son muy diversos cada arquitectura computacional debe utilizar un planificador adecuado a sus necesidades y características particulares El problema de disenar el mejor algoritmo planificador

de tareas es un problema de tipo NP Completo Esto implica que no puede ser resuelto por un algoritmo lineal en tiempo polinomial o que el encontrar la solución requiere de más tiempo y recursos de los disponibles para poder resolver el problema en un tiempo aceptable Algunos algoritmos resuelven el problema para arquitecturas específicas lo cual no significa que el algoritmo utilizado es el más apropiado para esa arquitectura en particular y mucho menos que sea una solución viable para otro tipo de arquitectura

En cualquier caso los planificadores de tareas permiten definir ciertos requisitos que tienen los usuarios de una nube o cluster de computadores como lo son la calidad de servicio (QoS) y la cantidad de unidades de procesamiento disponibles En las nubes comerciales la principal preocupación es como calcular los costo de las operaciones definidas por los usuarios en esquemas IaaS y PaaS Los planificadores de tareas deben tomar en consideración lo que se conoce como el Acuerdo de Nivel de Servicio (SLA *Service Level Agreement*) para la toma de decisiones en cuanto a la programación de tareas En Ahuja et al (2011) se presenta un algoritmo desarrollado para tomar en cuenta estas consideraciones en nubes computacionales comerciales basándose en el SLA Este propone un análisis previo de la estructura del trabajo a ser realizado y luego la creación de una instancia de máquina virtual con la configuración óptima para ejecutar el trabajo Una vez se encuentra esta estructura entonces se toman decisiones del costo Este tipo de planificadores están pensados para ejecutar tareas en mediano y largo plazo En el caso de aplicaciones que procesan gran cantidad de datos y tienen estructuras muy complejas es perfecto planear con antelación la manera en que vamos a ejecutar nuestra aplicación Si la aplicación va demorar unas cuantas horas pues es muy conveniente invertir algunos minutos en planificarla apropiadamente así podemos disminuir costos y aumentar el rendimiento de la nube

En Barrett et al (2011) se mencionan varios métodos de optimización de los planificadores para nubes computacionales Ellos utilizan modelos de Markov y algoritmos

genéticos para obtener mejoras en la toma de decisiones del planificador Igualmente su objetivo es mejorar el rendimiento con trabajos que se espera se ejecuten durante mucho tiempo Una pregunta obligada sería ¿Qué sucede si la aplicación a ser ejecutada debe hacerse a corto plazo? ¿En este caso podríamos aplicar este tipo de planificadores? O un nuevo tipo de planificador debe diseñarse para poder tomar decisiones de manera rápida De esta forma los trabajos podrían ser evaluados programados y ejecutados en la menor cantidad de tiempo posible Parte de este trabajo de tesis trata sobre la posibilidad de ejecutar trabajos que requieren mucho procesamiento mayormente concurrente en tiempo real

1 2 5 Computación en la nube distribuida

En Satyanarayanan and Bahl (2009) se propone un cambio considerable en la manera en que se implementan las computadoras de nube Su propuesta es crear un cluster dividido que reduzca él área de cobertura de cada *cloudlet* como el autor les llama para que parezca una red de área local Los *datacenters* pueden requerir mucho mantenimiento un *cloudlet* (ver Figura 1 5)por el contrario debe ser un aparato monolítico que incluya dentro de sí todas las características de la nube pero requiera poco o ningun mantenimiento En la Figura 1 6 se muestra un modelo conceptual de la nube distribuida a diferencia de la nube convencional mostrada en la Figura 1 4 no existe la necesidad de tener *datacenters* para albergar la nube computacional ya que los *cloudlets* se encuentran cerca de los clientes y por definición requieren de poco o ningun mantenimiento

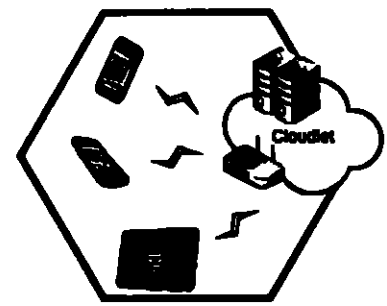
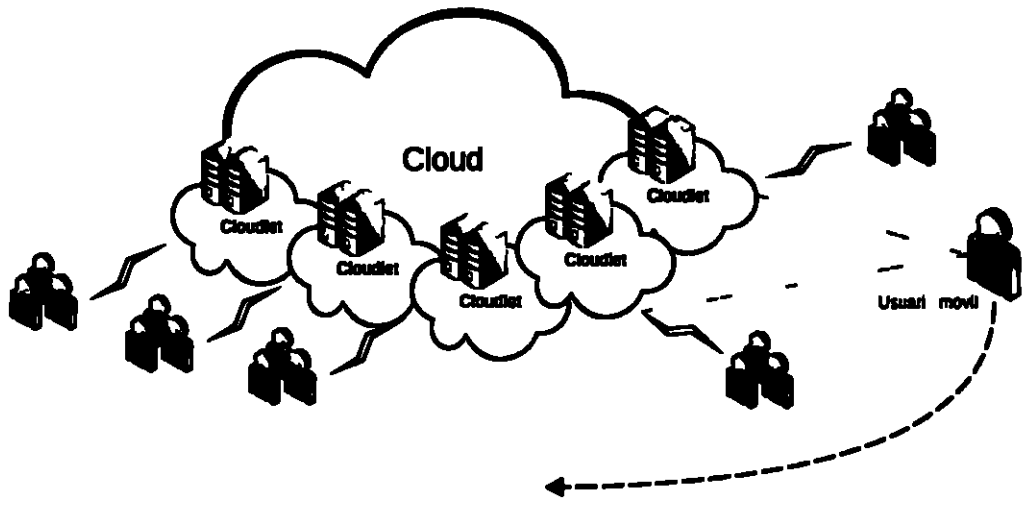


Figure 1 5 Modelo conceptual de un *cloudlet*



F gure 1 6 Modelo concept al de a be computacio al d stribu da

Este concepto encaja muy bien con el concepto de código bajo demanda (*Code on Demand* ver Lau et al (2005)) El código bajo demanda propone una plataforma multipropósito para construir aplicaciones utilizando pequenas piezas de código que se descargan para ser ejecutadas localmente ya sea en el servidor o en el móvil el *software* se crea entonces mediante la conjunción de tareas Multiples aplicaciones pueden compartir una misma tarea y varias instancias de una misma tarea pueden ser ejecutadas en un mismo *cloudlet*

Debido a que existen diferentes dispositivos es posible que existan diferentes piezas de código para una misma tarea de forma que puedan ser ejecutadas en diferentes dispositivos La inclusión de una máquina virtual permite eliminar esta necesidad unificando el código y reduciendo la complejidad de las aplicaciones y el *middleware* de la nube El propósito de este trabajo es crear una plataforma de desarrollo para aplicaciones distribuidas donde el código bajo demanda permita la división de las tareas de visualización y ejecución en la nube de manera que los dispositivos móviles con conexiones inalámbricas puedan ejecutar aplicaciones de forma remota y en tiempo real Esto sólo es posible si se reduce la latencia de la red y si las tareas se pueden ejecutar en forma paralela aquí es donde el código bajo demanda y las nubes distribuidas

utilizando *cloudlets*, entran en juego. En la Figura 1.7 se muestra gráficamente el concepto básico de las aplicaciones distribuidas dentro de la nube.

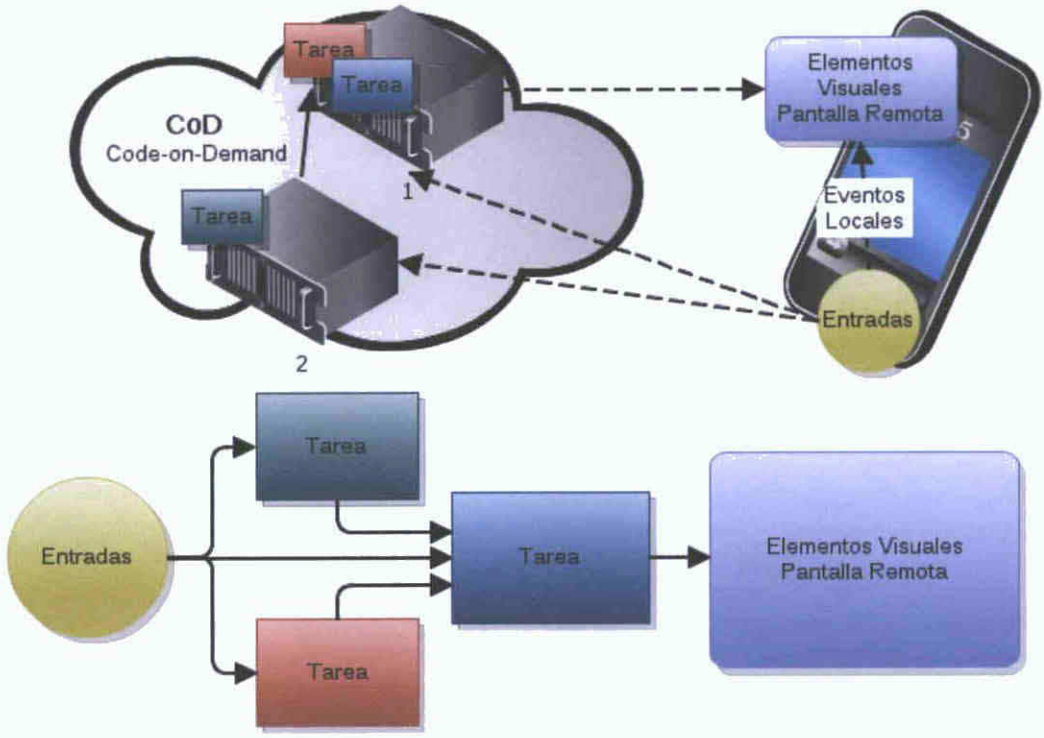


Figure 1.7: Concepto básico de aplicaciones distribuidas en la nube distribuida.

El código bajo demanda es una PaaS que puede cambiar la forma en que utilizamos los computadores actualmente. En particular, los dispositivos móviles pueden aprovechar una plataforma de desarrollo donde las piezas de código pueden reutilizarse para crear nuevas aplicaciones. Esto define dos niveles de desarrollo, el primero es a nivel de piezas de código generando programas autosuficientes que ejecutan una sola tarea. Un segundo nivel de desarrollo sería integrar todas estas piezas en una sola aplicación con diferentes tareas. Una ventaja de esta plataforma es la facilidad de ejecutar múltiples tareas en paralelo; si una aplicación requiere que múltiples tareas independientes sean realizadas estas pueden realizarse todas a la vez asignándolas en la nube a diferentes procesadores. Incluso podemos enviar las piezas de código al cliente móvil para realizar las tareas que deben ser ejecutadas allí.

El concepto de código bajo demanda presentado en Lau et al (2005) permite que algunas piezas de código con características especiales sean enviadas al dispositivo móvil para su ejecución De esta manera el cliente móvil se convierte en parte de la nube y sus recursos se unen a los recursos de la red Este concepto es importante ya que al dividir la aplicación en dos secciones es posible que los dispositivos móviles compartan algunos recursos para cumplir con las tareas que requiere la aplicación Principalmente se espera que esto ocurra en tareas de visualización ventanas y vídeos o en tareas de captura de datos como entradas de pantalla táctil o cámaras web Dado que los sistemas móviles han sido diseñados para realizar algunas funciones de procesamiento de datos es probable que algunas tareas como reconocimiento de imágenes o posicionamiento global sean enviadas al cliente móvil para su ejecución

1 3 Redes de Datos moviles

Una de las características más atractivas de los dispositivos móviles es precisamente la movilidad En radiocomunicaciones existe cualquier cantidad de esquemas de comunicación inalámbrica y se utilizan en base a su efectividad en diferentes tareas En el caso de los dispositivos móviles actuales hay dos tecnologías de comunicación inalámbrica que prevalecen la red de datos celular y la red inalámbrica IEEE 802 11 Existen otras tecnologías de comunicación inalámbrica como WiMAX o BlueTooth pero no son tan populares o no poseen los requerimientos necesarios para prestar servicios a dispositivos móviles comerciales o por lo menos los que se venden en la Ciudad de Panamá

Nos concentraremos entonces en definir las ventajas y desventajas de los sistemas de comunicación inalámbrica que encontramos en el mercado Veremos un poco de su funcionamiento y como esto podría o no encajar con los sistemas anteriormente descritos También se define en esta sección la tecnología a utilizarse en la fase expe-

rimental de esta investigación y se presentan las razones por las cuales se utiliza De cualquier forma esto no excluye la posibilidad de utilizar otro sistema de comunicación inalámbrica en cualquier otro momento

1 3 1 Redes de datos celular

Al comenzar este trabajo de investigación nos encontrabamos en la tercera generación de dispositivos móviles celulares La telefonía celular ha evolucionado como un sistema de comunicación de voz a un sistema híbrido de voz y datos Como sistema de voz los requerimientos de ancho de banda son relativamente bajos su exigencia se encuentra en su disponibilidad en el tiempo mientras que los datos demandan mucho ancho de banda y no requieren de disponibilidad temporal Este es el caso de páginas web y la descarga de archivos En la ultima década una nueva clase de servicios de datos han impuesto la necesidad de gran ancho de banda y comunicación en tiempo real Se hace entonces necesario aumentar la capacidad de los dispositivos móviles para ofrecer estos servicios de forma eficiente a pesar de las limitaciones del sistema celular

El *3rd Generation Partnership Project* (3GPP) se creó con el propósito de preparar aprobar y mantener las especificaciones y reportes técnicos de las redes móviles de tercera generación en adelante y se encarga de generar estándares para los participantes En esta se definen estándares como GPRS EDGE y UMTS que son redes de datos móviles En Panamá existen cuatro (4) proveedores de servicios de telefonía celular estos proveedores poseen infraestructuras de 2G con GPRS y EDGE de 3G con UMTS y actualmente lo que se conoce como 4G con HSPA+

Las redes GPRS (*General Packet Radio Service*) y EDGE (*Enhanced Data rates for Global Evolution*) son parte de la evolución de GSM para transmitir datos GPRS se basa en las especificaciones de R97 de GSM y tenía velocidades de conexión hasta

40Kbps aunque en la R98 y R99 se podía llegar a 171Kbps en teoría Una nueva mejora a GPRS fue EDGE esta tecnología puede alcanzar velocidades de conexión hasta 384Kbps lo cual la convierte en el estado previo de una red 3G aunque muchos consideran a EDGE una red 2 75G Los estándares actuales de la 3GPP desarrollaron *EDGE Evolution* como parte del *Release 7* para complementar la tecnología HSPA (*High Speed Packet Access*)

El *Universal Mobile Telecommunications System* UMTS es un término que cubre todas las tecnologías de radiocomunicaciones de tercera generación desarrolladas por la 3GPP Las especificaciones de acceso por radio proveen estándares para variantes de FDD (*Frequency Division Duplex*) y TDD (*Time Division Duplex*) permitiendo a las tecnologías UTRA (*UMTS Terrestrial Radio Access*) operar en un amplio ran go de bandas y coexistir con otras tecnologías de radio comunicaciones También se incluye el esquema original de W CDMA con canales de 5MHz especificados en R99 y el *Release 4* La introducción de HSPA permitió incrementar la tasa de transferen cia grandemente hasta 45Mbps incrementando las aplicaciones de conmutación por paquetes

En Panamá estos estándares están en pleno desarrollo y la máxima tasa de transferencia obtenible es de pocos Mbps pero el ancho de banda no está garantizado en todas las celdas Además los sistemas de comunicaciones de datos en redes celulares utilizan la técnica de *buffering* apilamiento de datos donde el radio se activa cada cierto tiempo para transmitir y recibir datos este método permite ahorrar en consumo de energía pero es muy ineficiente para servicios en tiempo real donde es necesario que se transmitan paquetes de forma sincronizada. Esta intermitencia de ancho de banda además de ser muy limitado presenta algunos problemas de infraestructura para algunas aplicaciones como la visualización remota, sobretodo cuando se utilizan tecnologías de escritorio remoto tradicionales Es posible que una optimiza-

ción en los protocolos de visualización remota permita la creación de aplicaciones de virtualización en Panamá La red móvil es la red de área amplia de mayor cobertura a nivel nacional mas estas limitaciones presentan algunas desventajas sobre una red experimental utilizando el estándar IEEE 802 11 mejor conocido como WiFi

1 3 2 Redes WiFi

WiFi (o *Wireles Fidelity*) es un término de referencia para todas las tecnologías establecidas en el conjunto de estándares IEEE 802 11 que cubre todas las tecnologías de redes de área local inalámbricas (WLAN) Los equipos que soportan esta tecnología pueden ser utilizados como puntos de acceso (AP) o como encaminadores inalámbri cos Existen diferentes modificaciones o adendas al estándar original los dispositivos actuales soportan IEEE 802 11a IEEE 802 11b IEEE 802 11g y IEEE 802 11n que es su modificación más actual Estas modificaciones definen diferentes anchos de banda y esquemas de modulación para dispositivos de red inalámbricos

Estos estándares definen tecnologías para las capas física y de acceso al medio del modelo OSI Las ventajas en ancho de banda son evidentes con respecto al sistema celular pero también poseen algunas limitaciones en el área de cobertura Sus radios abarcan lo que se considera un área local en cualquier caso utilizando varios radios puede extenderse la cobertura de un punto de acceso IEEE 802 11 Utilizando pro-tocolos de interconexión y adecuación de redes podemos obtener coberturas en áreas metropolitanas (MAN) e incluso en área amplia (WAN)

1 4 Computacion movil

En la actualidad los dispositivos móviles son los aparatos electrónicos más po-pulares en el mercado Su popularidad se debe a la evolución de la telefonía celular

y la computación móvil Los avances en la producción de microprocesadores y la integración de multiples funciones dentro de un solo circuito integrado permite la miniaturización de los dispositivos convirtiéndolos en dispositivos de bolsillo La reducción de consumo energético y los desarrollos en el almacenamiento de energía (baterías y celdas de combustible) permiten a estos aparatos permanecer muchas horas sin ser recargados Además el aumento de disponibilidad de redes inalámbricas y móviles así como sus capacidades en la transmisión de datos permiten el acceso ubicuo a Internet Poseen pantallas táctiles para facilidad de uso y cámaras digitales para captura de imágenes y vídeo

A pesar de todas estas nuevas características los dispositivos móviles se encuentran limitados en cuanto a capacidad de procesamiento y portabilidad Siempre se requiere que la batería del dispositivo móvil dure un poco más Y los dispositivos de entrada que nos facilitan su uso o nos permiten procesar datos ambientales utilizan algoritmos que requieren de mucho poder de procesamiento En sus primeras etapas los dispositivos móviles solo podían ejecutar una aplicación a la vez en la actualidad pueden realizar más funciones estos equipos son entonces más costosos El costo de las tecnologías con mayor integración se sigue elevando debido a las restricciones y necesidades de agregar más funciones en cada vez menos espacio Un método para aumentar la eficiencia de un dispositivo móvil y lograr que ejecute aplicaciones cada vez más complejas es utilizar el método de virtualización trasladando las aplicaciones a la nube computacional mientras dejamos el control y la visualización al dispositivo móvil

1 4 1 Sistemas Operativos para móviles

En el universo de los sistemas integrados donde los dispositivos móviles son los mayores exponentes se encuentra gran variedad de *software* para cada dispositivo en particular Los sistemas operativos integrados (*Embedded OSes*) son variaciones de

los sistemas operativos para computadoras regulares que han sido adaptados para el uso eficiente de los recursos de un sistema integrado En el caso de PDAs celulares y tabletas existen varios sistemas operativos que controlan sus funciones normal mente desarrollados por el fabricante Generalmente el código fuente del SO de un dispositivo móvil no está disponible a los usuarios esto fue hasta que recientemente se liberaron los códigos fuentes de varios sistemas operativos para móviles El movimien to de *software* libre desarrolló una versión del sistema operativo Linux para soportar la mayoría de los dispositivos dentro de un sistema integrado Versiones modificadas de Linux se han desarrollado para plataformas de *hardware* estándar y abierto por ejemplo ARM que es un procesador para dispositivos integrados muy poderoso

La capa de abstracción de *hardware* (HAL) de un sistema integrado que ha sido diseñado para Linux es estándar Todos los dispositivos en especial la tarjeta gráfica son diseñados con el mismo patrón y conectores de forma que el sistema operativo pueda reconocerlos y controlarlos La mayoría de los fabricantes de sistemas integra- dos actualmente impide que el *software* alcance el nivel donde se controla el *hardware* utilizando una máquina virtual o un software *middleware* Algunos sistemas opera- tivos como Windows CE Windows Mobile y Symbian son cerrados y por ende los fabricantes crean sus diseños de sistemas integrados para que trabajen directamente con estos sistemas operativos En cambio la compañía Google desarrolló una versión más eficiente de la máquina virtual de Java conocida como Dalvik esta es una máqui na basada en registros y su funcionamiento es similar al de una máquina real Encima de la máquina virtual se ejecuta la pila de *software* conocida como Android Otro conjunto de *software* libre para dispositivos móviles es Qt desarrollado por Nokia este realiza funciones similares a Android y es utilizado por comunidades de *software* libre como Maemo quienes desarrollan plataformas para dispositivos móviles

Es posible entonces definir las capas de abstracción en el sistema operativo in

tegrado que se deben modificar para lograr una división de tareas y la consecuente generación de un escritorio remoto Las capas de software que hemos identificado en la arquitectura de dispositivos móviles son

Capa de abstracción de hardware (HAL) lo conforman los controladores del hardware del dispositivo móvil Generalmente no está disponible a los usuarios ni desarrolladores de forma tal que el fabricante pueda garantizar que el *hardware* funcione adecuadamente en todo momento evitando danos debido a *software* defectuoso o malintencionado

Kernel es el nucleo del sistema operativo Este se encarga de la mayoría de las tareas básicas del móvil y brinda servicios a las aplicaciones instaladas

Maquina Virtual en el caso particular de los móviles de ultima generación encontramos una máquina virtual Java para soportar las aplicaciones Este *middleware* permite la portabilidad de aplicaciones entre dispositivos

- *Capa de aplicación* son el conjunto de aplicaciones que soporta el dispositivo móvil

En el caso particular de este proyecto de investigación se pretende modificar la forma en que trabaja la máquina virtual Para esto es necesario crear servicios de interac ción con la nube computacional que obtengan de ella los objetos y los datos que se requieren para ejecutar las aplicaciones Esto implica un tiempo extendido de carga de las aplicaciones pero su principal beneficio es la mejora de rendimiento que se ex perimentará una vez que la aplicación esté funcionando Este principio de operación está basado en la forma en que los usuarios perciben la operación de las aplicaciones generalmente en una computadora regular encontramos aplicaciones con un largo período de activación y luego su ejecución es relativamente rápida Esto se debe a que la aplicación carga en memoria desde el disco duro todos los recursos que necesita

para ejecutarse luego utiliza los recursos directamente reduciendo el tiempo de espera entre tareas En un esquema de virtualización tradicional las aplicaciones se ejecutan en el servidor enteramente mientras los datos de entrada y salida son el tráfico que se envía y recibe del cliente estos datos corresponden generalmente al vídeo y las entradas del usuario en teclados o ratones Un esquema de aplicación distribuida será entonces uno donde las aplicaciones intercambian paquetes de información específica de las tareas a ser ejecutadas no del vídeo ni del control ya que estos se ejecutan localmente y las tareas en la capa de abstracción inferior se ejecutan de forma remota en el servidor

Esto cumple con el mismo objetivo obtener la percepción de que la aplicación se ejecuta rápidamente una vez está preparada para ejecutarse En el caso de una LAN cableada esto es posible ya que las conexiones están garantizadas y el retardo de la red es poco aun así en este tipo de redes se pueden apreciar ciertos retrasos en la ejecución de tareas El usuario entonces percibe que algo no está funcionando y ejecuta la acción nuevamente hasta que haya un cambio Para evitar esto se hace necesario el manejo local de los eventos y las acciones del usuario La programación orientada a objetos permite la ejecución remota de objetos y el manejo de eventos de forma que el usuario pueda percibir que las órdenes que se le presentan están en ejecución y espere a que sean completadas de manera efectiva Para lograr esto se requiere de una aplicación distribuida que sincronice a través de la red las tareas de la aplicación local con la remota Al utilizar la interfaz gráfica del dispositivo móvil creada específicamente para él y al tener un servicio de ejecución de tareas en el servidor se puede obtener lo mejor de los dos mundos

1 4 2 Aplicaciones Móviles Distribuidas

Hasta ahora se han mostrado las tecnologías de escritorio remotos existentes y que se utilizan en aplicaciones comerciales Todas estas están basadas en la centralización del procesamiento en el servidor de virtualización mientras que el cliente ligero se convierte en una terminal tonta que sólo sirve como interfaz con el usuario En los dispositivos móviles esto es posible pero muy difícil de lograr debido a que existe un *firmware* que evita que lleguemos a dominar todas las funciones del dispositivo Esto puede utilizarse como una ventaja ya que se pueden incorporar parte de las funciones del dispositivo móvil dentro de un esquema de visualización remota En este caso podemos hablar de funciones como las máquinas virtuales Java que se ejecutan en la mayoría de los celulares PDAs y tabletas actualmente

Las máquinas virtuales como la del lenguaje Java permiten a los programadores acceder a las funciones del dispositivo móvil de forma segura y portátil Los objetos de Java pueden ser creados para ejecutarse dentro de dispositivos diferentes con poca o ninguna modificación del código fuente Esta portabilidad de código puede ser utilizada para crear un sistema de visualización remota orientada a objetos donde los dispositivos móviles que cumplan con estas características pueden acceder a máquinas virtuales remotas y ejecutar sus aplicaciones cargando unicamente los objetos visuales en el cliente ligero Los dispositivos móviles presentan una gran oportunidad para crear nuevo software que se adapte a nuevos sistemas de visualización remota

1 4 3 El Sistema Android

El software integrado que se escogió para ser utilizado como plataforma de prueba ha sido Android Este consiste en un kernel de Linux modificado para operar con dispositivos móviles una máquina virtual de registros llamada Dalvik y la platafor ma de desarrollo de *software* Java Android es un SDK (*Software Development Kit*)

desarrollado por Google para aplicaciones de móviles inteligentes celulares y tabletas Está basado en lenguaje Java y es completamente 'software libre La máquina virtual Dalvik posee una arquitectura basada en registros esto la hace más eficiente en el manejo de colas de instrucciones Para crear aplicaciones en Android es necesario en tender el paradigma de programación orientada a objetos y conocer el lenguaje Java Al ser software libre Android permite que muchos de los códigos sean modificados y mejorados por la comunidad de desarrolladores

En este caso se requiere modificar a Dalvik para crear una máquina virtual distribuida y se requiere crear una plataforma basada en Android donde los objetos se ejecuten con el paradigma de código bajo demanda así sólo se tendrán los objetos que se requieran en el momento y las instancias que se ejecuten serán las instancias correctas para el dispositivo utilizado Otro requerimiento es el manejo distribuido de las tareas puesto que se pretende ejecutar el lado del servidor desde una nube es importante mantener actualizadas todas las instancias donde la máquina virtual del lado del servidor se estén ejecutando

1 5 Propuesta de investigacion

Hasta ahora hemos mostrado el trabajo y los antecedentes de sistemas distribuidos y sistemas móviles que hasta el momento en que se realizó el estado del arte y se revisaron los antecedentes no tenían mucho en comun En general las aplicaciones distribuidas que se utilizan desde sistemas celulares son aplicaciones en línea (tipo Web) que en realidad son una aplicación cliente y otra servidor Las aplicaciones para utilizar escritorios remotos han demostrado poseer una gran cantidad de inconvenientes que impiden que el sistema funcione apropiadamente Esto se debe mayormente a la conexión inalámbrica intrínseca y necesaria de la tecnología móvil A pesar de esto algunas iniciativas de desarrollo se muestran en Simoens et al (2011) estas fue-

ron realizadas con dispositivos móviles que no poseían sistemas operativos como los dispositivos Android

En los nuevos teléfonos y tabletas existe un sistema operativo local que permite la utilización de aplicaciones igual que en un computador Entonces las posibilidades permiten la creación de aplicaciones distribuidas que eviten problemas de sincroni zación (ver sección 1 1) y de la movilidad En este caso la propuesta de este trabajo de investigación es la creación de una plataforma de desarrollo que permita la ejecu ción distribuida de aplicaciones con gran cantidad de operaciones Estas aplicaciones pueden ser de cualquier tipo pero mayormente nos referimos a aplicaciones con gran cantidad de cálculos matemáticos Por ejemplo aplicaciones de tratamiento digital de senales para vídeo y fotografía guías de tráfico para multiples usuarios y videojuegos Este tipo de aplicaciones podrían ejecutarse en el móvil pero estos han sido disenados para realizar otro tipo de operaciones que reducen el rendimiento de cualquier aplica- ción de este tipo En especial aplicaciones de tiempo real se ven afectadas por retardos inherentes del sistema que no pueden ser evitados Las nubes computacionales pa- recen ser la solución final a las limitaciones de los dispositivos móviles permitiendo aumentar sus capacidades en aplicaciones disenadas para trabajar utilizando lo mejor de ambos sistemas

Las nubes computacionales actuales están centralizadas esto quiere decir que son controladas por un sólo ente (generalmente una empresa) y en muchos casos sus componentes se encuentran en un sólo edificio Utilizan mucha energía y se encuentran en regiones apartadas lejos de los usuarios de móviles Una iniciativa planteada por Satyanarayanan and Bahl (2009) donde aparece el concepto de *cloudlet* dice que se pueden localizar las prestaciones de la nube en un dispositivo de bajo costo y poca necesidad de mantenimiento Esto es posible debido a las tecnologías de multiples procesadores con multiples nucleos En este sentido cualquier computador que pueda

albergar un procesador con muchos nucleos es potencialmente un *cloudlet*

El objetivo de este trabajo es crear un enlace de comunicación entre un dispositivo móvil y un computador con multiples nucleos comparar su rendimiento con el de un móvil realizando el mismo trabajo por sí solo y presentar conclusiones de la viabilidad de esta conexión Además se pretende crear una pequena plataforma de desarrollo de aplicaciones distribuidas que permita reservar los recursos del *cloudlet*de forma eficiente Se propone también crear una base confiable de piezas de código que se utilizarán para construir las aplicaciones distribuidas aumentando la seguridad de la aplicación Esto se logra ya que el móvil no tiene control sobre las piezas de código que se ejecutan en el *cloudlet* tampoco deciden en que unidad de procesamiento se ejecuta el código

Una de las principales desventajas de una aplicación móvil distribuida es la transferencia de datos a través de conexiones inalámbricas En este trabajo utilizaremos el estándar IEEE 802 11 para realizar las pruebas ya que una conexión a través del sistema celular no es viable ya que nuestro servidor tendría que estar conectado a la radio base directamente o mediante una conexión de Internet que eliminaría el concepto de conexión local del servicios distribuido Esto sin contar que una conexión directa con una radio base celular acarrearía costos no disponibles al proyecto

Al finalizar este trabajo esperamos que la aplicación distribuida posea un rendimiento similar en tiempo de ejecución a una aplicación que funciona enteramente en el móvil Un rendimiento similar implica que la aplicación distribuida debe poseer un tiempo de ejecución preferiblemente igual o menor al de la aplicación móvil Aunque para algunas aplicaciones el tiempo puede ser mayor si es aun aceptable o imperceptible para el usuario

Capítulo II

Definición de Requerimientos

Para realizar una red de computación ubicua y distribuida se requiere que mul tiples nodos de procesamiento trabajen en conjunto Los clusteres de computadoras y las nubes computacionales generan capacidades para realizar procesos informáticos que se ajustan a las necesidades de los usuarios y de las aplicaciones que estos eje- cutan En el caso de los dispositivos móviles hay una gran oportunidad de hacer uso de la potencia computacional que le puedan aportar nubes computacionales ofrecidas a través de *software* como un servicio (SaaS) pero en el caso de las plataformas de desarrollo de *software* como un servicio (PaaS) esta oportunidad es aun mayor La plataforma de desarrollo Android por ejemplo utiliza código abierto con el lenguaje Java que utiliza el paradigma de programación orientada a objetos (OOP) y aun cuando se utiliza una máquina virtual para ejecutar las aplicaciones posee algunas vulnerabilidades y limitaciones de ejecución que pueden ser solventadas utilizando una PaaS en una nube computacional

PaaS es el servicio que tiene mayor potencial para crear nuevos paradigmas de programación que mejoren tanto las capacidades de los dispositivos móviles como la seguridad de las aplicaciones El paradigma de código bajo demanda (CoD) que consiste en la separación de las tareas a ser ejecutadas dentro de una aplicación en

pequenos ejecutables independientes con interfases de comunicación estándar de la lógica del programa principal permite el uso eficiente de los nodos computacionales por ejemplo en un cluster y aumenta grandemente la seguridad de las aplicaciones Si se separan todas las tareas de las aplicaciones se formarán programas simples totalmente independientes unos de otros que llamaremos piezas de código Esto hace posible separar el alcance de las entradas y salidas de las variables y objetos internos de la aplicación Esto puede evitar la creación de *software* malicioso y la intromisión en la estructura interna de las aplicaciones La creación de una TCB como se explica en la sección 1 2 2 permitirá el control de acceso a las aplicaciones limitando a los desarrolladores de *software* en el acceso a niveles críticos del sistema Al crear un sistema híbrido se evitan posibles limitaciones en el desarrollo de aplicaciones La TCB puede ser actualizada cada vez que sea necesario a petición de los desarrolladores mientras se revisa que la tarea planeada no intervenga con la seguridad del sistema

Como el objetivo es crear entonces una nube computacional con aplicaciones de tipo CoD esto tiene algunos requerimientos que serán definidos en este capítulo Empezaremos con los requerimientos básicos y luego veremos como encaja esto con el esquema de nube distribuida (*cloudlets*) El ámbito de este proyecto se limita a la comunicación entre un unico móvil con la nube distribuida y es por esto que se emulará un *cloudlet* con un servicio PaaS de CoD instalado en él Siguiendo la definición encontrada en Satyanarayanan and Bahl (2009) los *cloudlets* son unidades de cómputo que emulan una pequena nube computacional un cluster pequeno con capacidades suficientes para ejecutar aplicaciones de la nube y que requieran de poco o ningun mantenimiento Esto permitiría incluso la operación de nuestras aplicaciones cuando no haya conectividad con el resto de la nube puesto que los datos y las aplicaciones de los clientes se distribuyen a los *cloudlets* que le brindan servicio y se borran una vez este se retira de su área de cobertura

La replicación de los datos y la movilidad de los mismos es otro objetivo de esta tecnología Mientras que al utilizar un almacenamiento centralizado los datos están seguros y replicados en un sólo lugar el tiempo que se requiere para hacerlos disponibles puede llegar a ser demasiado para la experiencia del cliente En el caso de archivos muy grandes como los de vídeos es probable que la mejor forma de almacenamiento sea un almacenamiento distribuido ya que al estar distribuido en diferentes servidores puede ser accedido por diferentes canales de transmisión evitando cuellos de botellas Otra ventaja de un almacenamiento distribuido es que si el usuario se está moviendo entre diferentes zonas de cobertura puede acceder a diferentes partes del archivo en diferentes locaciones Puede predecirse en cuales zonas de cobertura estará el usuario accediendo los archivos basados en la locación de los *cloudlets*

En Satyanarayanan and Bahl (2009) se sugiere también la utilización de un archivo de perfil con la configuración de las aplicaciones y preferencias del usuario Un archivo de este tipo permite al sistema saber que piezas de código necesita del conjunto total de piezas que existen Las aplicaciones serán como un rompecabezas cada rompecabezas es diferente pero utiliza las mismas piezas que los demás Esto sugiere que cada uno de los *cloudlets* contiene todas las piezas de código que vaya a necesitar y cada actualización será distribuida a cada uno de ellos Esto evita la incompatibilidad de aplicaciones los desarrolladores podrán crear aplicaciones verdaderamente portables y además seguras Serán seguras porque nadie puede incluir código malicioso en el sistema ya que este es responsabilidad unicamente del proveedor de la plataforma y no de los desarrolladores En este caso el desarrollo de la TCB debe estar sujeto a revisión por parte del proveedor o una comunidad de desarrolladores que garanticen la integridad del código No se intenta con esto crear código cerrado o privativo muy por el contrario el código abierto ha demostrado ser la mejor opción para el desarrollo avanzado de aplicaciones pero hay que limitar la posibilidad de permitir el acceso a

individuos que hagan mal uso de estos recursos

2 1 Requerimientos del sistema

Uno de los requerimientos primordiales de la computación en la nube es la escalabilidad para servir multiples usuarios Los recursos son asignados a los usuarios segun sea requerido por las aplicaciones que estos ejecuten En el caso de la nube distribuida la cantidad de recursos en cada uno de los *cloudlets* es limitada a pesar que se pueda acceder a recursos en nodos cercanos existe la posibilidad que una unidad se haga cargo de muchos usuarios en un momento dado y sus recursos no sean suficientes Esto puede ocurrir si existe una interrupción en la comunicación con el resto de la nube o simplemente que el resto de las unidades estén igualmente saturadas Los requerimientos entonces para un sistema como este deben ser adecuados para una cantidad máxima de usuarios por zona de cobertura Es difícil definir la zona de cobertura cuando se utilizan ondas radio eléctricas para la transmisión de datos este es otro aspecto que se debe tomar en consideración

El estándar IEEE 802 11 o WiFi define diferentes tasas de transmisión basadas en la distancia a la fuente la sensibilidad de los receptores y la ganancia de las antenas que posean Puede definirse la zona de cobertura y así la cantidad máxima de usuarios mediante la definición de la tasa mínima de transmisión de datos en especial en la dirección móvil radio base Es necesario considerar que mientras los dispositivos móviles pueden posicionarse mejor para transmitir datos no poseen suficiente potencia de transmisión (ya que funcionan con baterías) para incrementar la distancia de transmisión efectiva en este caso es inevitable la pérdida de paquetes al aumentar la eficiencia energética Muy por el contrario la radio base posee una buena sensibilidad en su receptor y energía eléctrica constante proveniente del sistema eléctrico con esto puede adecuar la recepción y la transmisión de datos segun sea el caso utilizando

amplificadores

Entonces el definir los recursos que van a requerir los usuarios está directamente relacionado con la cantidad de usuarios que está definida con el área de cobertura del *cloudlet* En cualquier caso los recursos se asignan dinámicamente por ende se debe generar un numero promedio de recursos Los recursos más comunes que tenemos a disposición en la nube son unidades de procesamiento (CPUs) RAM y espacio de al macenamiento De estos tres recursos los más críticos para el buen funcionamiento son los CPUs Sin unidades de procesamiento no es posible ejecutar ninguna aplicación por tanto se hace necesario que la pila de procesadores disponibles nunca se vacíe Es posible sin embargo tener todos los CPUs planificados para ejecutar algun proceso pero no todos en el mismo instante También es necesario recordar que existen procesos de control mantenimiento y evaluación dentro del *cloudlet* que requieren de tiempo de procesamiento Generalmente en los clusteres de alto rendimiento se suele tener un procesador dedicado a estas tareas

La memoria es el segundo factor crítico dentro del *cloudlet* pero la cantidad de memoria necesaria está directamente relacionada con el proceso que se ejecuta en la unidad de procesamiento Usualmente la cantidad de memoria disponible supera por mucho la memoria necesaria para la ejecución de una tarea en particular En el sistema operativo Linux se utilizan archivos para almacenamiento de datos intermedios permitiendo liberar la memoria de estos datos y sacándolos de la pila de memoria donde podrían ser accedidos por un proceso malicioso Además los sistemas distri buidos generan multiples datos que requieren ser accedidos por diferentes programas almacenar estos datos en memoria y dejarlos para que otra aplicación los utilice puede implicar cambios en el sistema operativo (SO) La mayoría de los SO están disenados para tratar cada pieza de código compilado como una entidad con sus propios recursos al terminar la aplicación se liberan todos los recursos adquiridos durante la ejecución

El almacenamiento conectado a la red es la solución para mantener todos los nodos sincronizados en cuanto al contenido en disco soluciones de almacenamiento de red y sincronización como NFS (*Network File System*) serán parte de las especificaciones de este sistema

El almacenamiento es el ultimo de los recursos disponibles dentro de una nube computacional NFS puede ser una alternativa apropiada para lograr la sincronización de los archivos de los usuarios en diferentes locaciones Esto evita que los datos se estén copiando constantemente a diferentes locaciones una vez que el usuario deja la zona de cobertura del *cloudlet* actual los datos son montados en el siguiente *cloudlet* El almacenamiento es fácil de calcular como requerimiento del sistema simplemente hay que ver el máximo de usuarios multiplicado por el máximo de espacio que cada uno puede utilizar dentro del *cloudlet* El NFS no ofrece una característica de sincronizar sólo los datos que se están utilizando dentro de la máquina local por defecto pero es una característica que puede ser configurada o anexada en una subversión del código fuente Este máximo de usuarios es posible que no pueda utilizar el máximo de almacenamiento en cada sesión así que no es necesario tener el máximo de almacenamiento para cada usuario También es posible que los usuarios en un área de cobertura sean el máximo numero de usuarios posibles Ambas razones definen que el almacenamiento requerido para los usuarios no necesariamente es el máximo de almacenamiento por el máximo numero de usuarios pero considerando el costo del almacenamiento se puede utilizar este parámetro como un requerimiento base En cuanto el mercado requiera aumentar el almacenamiento de datos para los usuarios podemos aumentarlo hasta alcanzar el tamano crítico de almacenamiento sin tener que aumentar el tamano físico de almacenamiento

2 1 1 El planificador de procesos

Los recursos que requiere una aplicación deben ser planificados previamente por el desarrollador de software El planificador de procesos es un programa que permitirá reservar los recursos en particular las unidades de procesamiento para ejecutar las diferentes tareas que tenga la aplicación Al utilizar un paradigma de software distribuido se hace necesario crear un plan de tareas para la división del trabajo y otro para el agrupamiento de las respuestas intermedias en una sola salida coherente y ordenada Estos planificadores no sólo reservan los recursos ellos pueden programar las tareas para que se ejecuten de forma ordenada evitando el exceso de solicitudes En el caso que hayan más solicitudes que procesadores disponibles estás tendrán que esperar en una cola generalmente FIFO (*First In First Out*) aumentando el retardo en la ejecución de todas las aplicaciones También se pueden definir prioridades de ejecución que permitirá la ejecución de tareas críticas primero

En un cluster regular esta tarea la lleva a cabo el proceso que controla el cluster este asigna los recursos dependiendo de las peticiones de los clientes Una vez la cantidad de peticiones supera la cantidad de procesos dentro del cluster o la nube los procesos deben esperar en la cola para ser ejecutados lo antes posible Estos planificadores de nube generalmente tienen un proceso paralelo que entra en función cuando alguno de ellos falla (*failover*) este se puede encontrar en otra unidad dentro de la nube pero no significa que sea un proceso totalmente independiente Muy por el contrario el servidor *failover* posee los mismos datos y está esperando solamente a que el principal falle o se sature para reemplazarlo

En el caso de que existan varios clusteres cada uno posee su administrador de recursos con su planificador en este caso cada uno es independiente si algun proceso requiere recursos de los otros clusteres tendría que solicitarlos como un cliente a menos que exista un proceso regulador de los diferentes clusteres llevando el control de

todos los procesos Esto requiere de varias capas de software y mucho poder de procesamiento para llevar este control Para generar una nube computacional distribuida se requiere entonces encontrar un método de control distribuido

2 1 2 El planificador de procesos en la nube distribuida

Cuando la cantidad de tareas programadas exceda la cantidad de procesadores dentro de un *cloudlet* este debe contactar a unidades similares cercanas y reservar sus recursos disponibles Estos procesos pueden tomar un pequeno tiempo adicional dependiendo de la distancia y la tasa de transmisión que se disponga para realizar la transferencia de los datos Al poseer todos los *cloudlets* todas las piezas de código almacenadas localmente se realiza una llamada remota de la tarea y se le envían los datos correspondientes para ejecutarla Debe existir un protocolo de intercambio que mantenga actualizada la tabla de recursos disponibles y el costo de enviar los datos a los diferentes nodos dentro de la nube distribuida Estos protocolos deben funcionar de manera similar a los protocolos de encaminamiento actualizando las tablas de recursos en particular de procesadores disponibles de manera constante De la misma manera con esta información cada *cloudlet* puede crear un mapa del estatus de la nube en todo momento

Este tipo de planificación distribuida elimina la necesidad de un ente director dentro de la nube ya que cada una funciona como una entidad independiente Esto le permite a la nube un escalamiento similar al de Internet sin ningun tipo de restricciones debido a la necesidad de más poder de procesamiento para llevar la planificación de las tareas solicitadas por los clientes Una ventaja adicional es que no se tiene un director central por ende el servicio no se interrumpe completamente en toda la nube si un nodo falla Además la cobertura y la necesidad de llamar a nodos remotos será por sectores evitando que exista una concentración excesiva de tránsito hacia

un nodo central

Otro de los inconvenientes de los planificadores de procesos actuales es la multiplexación de tareas por división de tiempo heredada de la emulación de trabajo concurrente en procesadores con un sólo nucleo En la actualidad con multiples nucleos de procesamiento es posible realizar trabajo concurrente real no emulado por un procedimiento de conmutación de tareas que permita realizar varias operaciones a la vez Borkar (2007) define la posibilidad de que las empresas fabricantes creen procesadores con muchos nucleos simples y algunos complejos para realizar las operaciones que requieren las diferentes aplicaciones Actualmente los nucleos de los procesadores son de 64 bits debido a la limitación de acceso a RAM que genera una barrera de direccionamiento mientras que las operaciones realizadas por el comun de los usuarios de computadores no requiere tal precisión en las operaciones Lo que sugieren Borkar (2007) Dorn et al (2011) y Sun and Chen (2010) es que cada procesador posea su propia RAM Esto es visible en los sistemas NUMA(*Non Uniform Memory Access*) y aunque no es la opción más eficiente ciertamente es la más viable desde el punto de vista de los costos de fabricación Entonces los planificadores de procesos puedan ser disenados ahora para dedicar el tiempo total de procesamiento a cada tarea que se realiza disminuyendo el tiempo de ejecución de cada tarea Este tipo de diseno de planificadores de proceso cambia la perspectiva de ejecución de las aplicaciones ya que podemos contar cada proceso dentro de un procesador y cada procesador tendrá entonces una pila de subprocesos que deba realizar

2 1 3 Unidades de procesamiento

En la nube distribuida cada unidad de servicio si así podemos llamarle a los *cloudlets* debe poseer las mismas características de una nube computacional convencional Es posible encontrar un cluster distribuido en una región muy grande en

mallas computacionales que abarcan diferentes regiones pero aun estos se comportan como una unidad regida por un proceso de control centralizado Cada nodo es independiente de realizar la tarea que se le ha encomendado pero no tiene decisión sobre que procesos ejecutar En la nube distribuida cada unidad que brinda servicio a los clientes de la nube posee su propio ente rector que organiza las tareas y las programa para que se ejecuten lo antes posible Esta unidad puede contar con los recursos disponibles en otras unidades esto complica la situación debido a que existen varios procesos de control Uno de los requisitos para que la nube pueda brindar sus servicios a diferentes clientes con aplicaciones que requieren varios procesadores es teniendo una cantidad considerable de nucleos de procesamiento y redundancia

En el mercado de tecnología actual existen procesadores con multiples nucleos de procesamiento La cantidad de procesadores con multiples nucleos que se requiere entonces para soportar un grupo de clientes estará ligado a la cantidad máxima de usuarios que puede soportar un *cloudlet* en particular Es necesario contar un procesador dedicado para controlar el *cloudlet* y los procesos de administración y comunicación con el resto de la nube Eso nos deja con la cantidad de unidades de procesamiento que se muestra en la ec (2 1)

$$P = X_{pr}\ N - 1 \tag{2 1}$$

donde

P es el numero de unidades de procesamiento dentro del *cloudlet*,

X_{pr} es el numero de procesadores físicos dentro del *cloudlet* y

N es la cantidad de nucleos en cada procesador físico

La cantidad P requerida para un *cloudlet* estará dada por las especificaciones del fabricante debido a la relación costo beneficio Para poder determinar este numero se hace necesario definir también la cantidad de unidades de procesamiento que un sólo

usuario puede reservar dentro de la nube La Ley de Amdhal segun Borkar (2007) y Sun and Chen (2010) establece un máximo de procesadores por proceso entre ocho y dieciséis para que el aumento de unidades de proceso no afecte negativamente el tiempo de ejecución También este numero está dado por el SLA (*Service Level Agreement*) que garantiza que el usuario puede reservar una cantidad máxima de procesadores para sus aplicaciones Otra perspectiva posible es la de definir que cada aplicación puede reservar cualquier numero de procesadores segun sea la disponibilidad de los mismos siempre que cumpla con la Ley de Amdhal definiendo un mínimo de uno para cada aplicación También se puede crear una combinación de ambas perspectivas Otros factores también inciden dentro de este esquema como la prioridad de los procesos Cada proceso puede o no ejecutarse dentro del *cloudlet* local o en el *cloudlet* remoto más cercano habría que definir entonces que procesos son prioritarios y cuales no Un proceso prioritario se ejecuta dentro de un viaje redondo del cliente a la nube mientras que el resto de los procesos tendrían un tiempo adicional que es el viaje redondo entre *cloudlets* más la transmisión de los datos y resultados La sincronización y conjunción de los archivos de resultados constituye un tiempo adicional

Cada tarea que se planifica utiliza un procesador como mínimo por un tiempo limitado y es necesario que exista un procesador físico para atender a la tarea cada vez Cuando la cantidad de tareas dentro del *cloudlet* supera la cantidad de procesadores pueden ocurrir dos cosas la tarea entra en una cola de espera o la tarea es enviada a otro *cloudlet* con recursos disponibles para ser ejecutada lo antes posible Este problema es crítico en este concepto de nube distribuida puede ser que un algoritmo requiera ser ejecutado lo antes posible por operar con la percepción de procesamiento en tiempo real y sus resultados tampoco puedan esperar a ser retransmitidos varias veces En la arquitectura de nube que se presenta en este trabajo la distribución de los recursos no posee jerarquías locales esto indica que un *cloudlet* conoce donde se

encuentran todos sus vecinos en una zona determinada y puede solicitar los recursos que estén disponibles en ellos Para aplicaciones de interacción humana que no es el comun de las aplicaciones que soportan las mallas computacionales convencionales es necesario comprender que el planificador de procesos debe procesar peticiones de forma rápida

En el caso de la nube distribuida además incluimos una dificultad adicional los diferentes componentes de la nube los *cloudlets* poseen un planificador propio cada uno Se hace necesario entonces publicar a través de la red que une a la nube el estado de las unidades de proceso en todo momento de forma tal que los otros miembros de la red puedan adaptar sus necesidades a los recursos disponibles Este esquema puede ser muy similar al utilizado por los protocolos de la capa de red del modelo OSI La difusión de los recursos disponibles a los demás servicios planificadores en otros *cloudlets* permite que éstos puedan realizar peticiones que permitan a la red converger de manera rápida y sencilla

2 1 4 Memoria

En el caso de la memoria requerida por cada proceso es necesario tener en consideración el tamano de los datos de entrada y el tamano de los datos de salida En procesamiento digital de senales por ejemplo los datos de entrada generalmente suelen ser más que los de salida En otros casos como en un procesador de texto los procesos paralelos que se ejecutan podrían generar un numero muy grande de datos adyacentes a la salida Deben entonces existir algoritmos que con muy pocos datos de entrada generen muchos datos de salida El tipo de aplicaciones que se consideran en este trabajo de cualquier forma son aplicaciones para móviles Los móviles pueden generar muchos datos de entrada a las aplicaciones generalmente si se quieren procesar senales digitales de audio y vídeo y mostrarlas en pantalla En el caso de

videojuegos y aplicaciones 3D esta cantidad de datos es transmitida constantemente en el canal inalámbrico

Hay algunas consideraciones del tamano de la memoria pero considerando los disenos actuales de los computadores y la reducción de costo por KB de la RAM es difícil pensar que un *cloudlet* pueda quedarse sin memoria suficiente En cualquier caso se puede implementar una regla en el planificador de tareas que considere la utilización de memoria de forma tal que envíe procesos a otros *cloudlets* que tengan mayor cantidad de memoria disponible Debido al esquema de publicación de recur sos disponibles expuesta anteriormente es posible entrar en condiciones de carrera (*race conditions*) con los demás *cloudlets* saturando a un nodo en particular Se hace necesario entonces que exista un método de rechazo de peticiones que no afecte la ejecución de tareas de un trabajo en particular El SLA podría definir en ciertas apli caciones la posibilidad de perder algunos datos en favor de la continuidad del flujo de datos otros requieren mayor fiabilidad a un costo de mayor tiempo de ejecución Una vez un proceso es rechazado debe buscarse un nuevo procesador para que ejecute la función Otra opción unido al hecho de que los datos pueden servir a subtareas dentro de un trabajo es que se limite el tamano máximo de datos de entrada y salida para cada trabajo En este caso debe realizarse una petición que determine la cantidad de datos a ser enviados y recibidos de antemano

En el caso de que cada procesador físico posea su propia RAM dedicada sería posi ble limitar la cantidad de procesos que ese chip en particular puede ejecutar conside- rando la cantidad de memoria disponible De esta manera también podemos observar que la complejidad de los procesadores y su disipación de energía se reducirían mien tras se mantiene la capacidad de la memoria aleatoria En sistemas operativos como Linux donde es normal la creación de archivos de datos intermedios entre operaciones y siguiendo el paradigma de CoD se hace muy util que la memoria esté asignada a

cada procesador aunque no sea mucha mientras que el almacenamiento masivo es un comun del sistema Esto también encaja si el *cloudlet* posee una arquitectura de sistemas integrados que posee limitaciones en cuanto a crecimiento

2 1 5 Almacenamiento

El almacenamiento de cada usuario del *cloudlet* debe ser suficiente para ejecutar funciones intermedias de lectura y escritura de datos Debido a que el almacenamiento distribuido puede sincronizarse en todos los nodos de la red es posible que los datos sean locales en todos los *cloudlets* que atienden peticiones de otros *cloudlets* Sistemas como NFS que pueden montar y desmontar unidades de red son muy utiles para mantener sincronizados los datos sin tener que realizar un envío real de los datos hasta cuando sea realmente necesario Estos también aumentan la seguridad porque los datos de diferentes usuarios pueden estar almacenados en diferentes nodos de la red sin ningun orden particular simplemente diferenciados por un identificador

2 2 Especificaciones de una unidad *cloudlet*

Para este proyecto de investigación se diseñó una plataforma de pruebas que emu lase las capacidades de un *cloudlet* Esto se debe a que no existía un dispositivo *cloudlet* comercial al momento de empezar este proyecto de investigación es posible que todavía no exista Un *cloudlet* por definición es un dispositivo monolítico que posee *hardware* parecido al que tendría un cluster pequeno y que se comporta como una nube computacional por sí mismo La mejor forma de crear un dispositivo tal con la tecnología actual es utilizando sistemas integrados La tecnología de sistemas integrados permite la conjunción de multiples sensores actuadores y formatos de comunicación en un sólo dispositivo móvil o fijo En este caso si hubiese un sistema

cloudlet creado con esta tecnología debería poseer las siguientes especificaciones de *hardware*

Procesadores con multiples nucleos por lo menos dos con cuatro o más nucleos cada uno

- RAM dedicada para cada procesador al menos 2GB por procesador
- Comunicaciones inalámbricas como WiFi (IEEE 802 11a/b/g/n/s) EDGE UMTS HSPA+ y pronto LTE
- Almacenamiento de estado sólido o soporte para tarjetas de memoria Flash como USB SD MMC y otras
- Comunicaciones por cable UTP o coaxial lo ideal sería una conexión de fibra óptica directamente a una red FTTH

En cuanto a los requerimientos de *software* dentro del dispositivo *cloudlet* tenemos

- *Firmware* del sistema operativo preferiblemente software libre (cualquier distribución de Linux)

  Base de código confiable (TCB) con todas las piezas de código que utilizarán las aplicaciones distribuidas

- Planificador local independiente que publique el estado de los recursos a los demás nodos de la nube
- Protocolos de administración aceptación de trabajos y actualización de los datos en el *firmware*
- Sistema de archivos distribuido de red tipo NFS o aplicaciones parecidas a Dropbox o Ubuntu One

- Servicio SSH y soporte SSL y IPSec.

En la Figura 2.1 se muestra la arquitectura básica que reúne todos estos elementos dentro del esquema propuesto. Como se puede observar el *cloudlet* como parte de la nube distribuida posee todos los elementos para conformarse en una nube computacional pequeña. Este ofrece sus servicios a dispositivos móviles y a otros nodos *cloudlet* en la red. El almacenamiento distribuido en la nube permite la migración rápida de los datos de un nodo a otro permitiendo el *roaming* entre nodos *cloudlet.* Este almacenamiento además permite la distribución de las piezas de código para ejecutar las aplicaciones, así como la transmisión entre nodos de resultados intermedios. Esto implica que algunos nodos *cloudlet* no requerirán almacenamiento local, excepto para levantar sus servicios básicos; un *firmware* que puede estar integrado en su tarjeta madre.

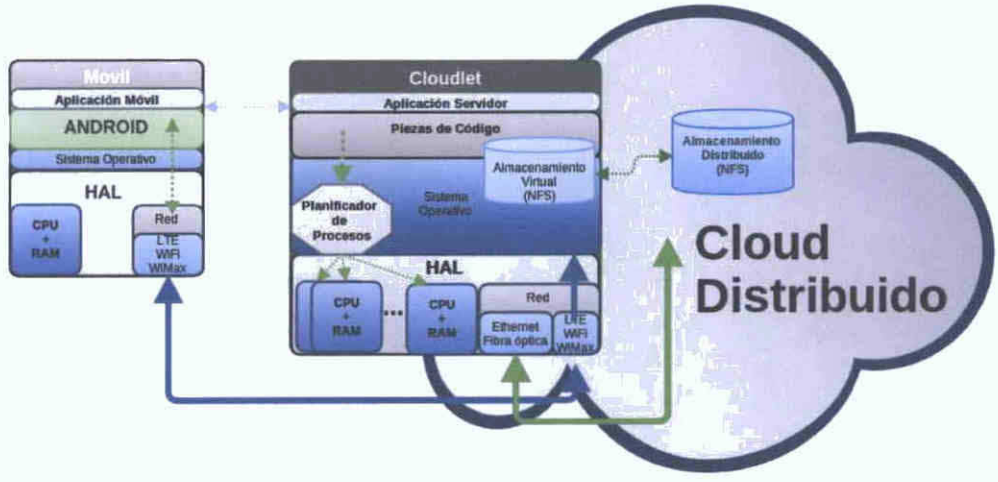


Figure 2.1: Arquitectura de la nube distribuida

Una de las especificaciones más críticas es la de comunicaciones. Esto se debe a la necesidad de reducir la latencia en la comunicación entre los móviles y la nube distribuida. Utilizar una conexión inalámbrica en un área local es muy importante, en el caso de la red celular los *cloudlets* deben residir en la radio base, o ser la

como es el caso de las pequenas celdas En los escenarios utilizando IEEE 802 11 habrá que implementar un sistema de *handoff* para mantener la conexión con el móvil esto debe realizarse en la capa de enlace de datos Debido a que los datos se transfieren a la nube y es ésta la que tiene la tarea de ejecutar la comunicación se vuelve un enlace directo sin saltos ni rutas que escoger Sería entonces posible que las otras capas del modelo OSI sean innecesarias Para el resto de la nube los protocolos de la arquitectura TCP/IP deben permanecer sin cambios

2 3 Requerimientos de la plataforma de pruebas

Una vez establecidas las especificaciones básicas del *hardware* y el *software* que conforman un *cloudlet* y sabiendo que no existe un dispositivo así debemos encon trar un equipo que reuna algunas características descritas en la sección anterior Los requerimientos más importantes son generalmente las unidades de procesamiento la memoria y la conectividad Ciertamente un servidor comercial posee algunas especi ficaciones atractivas además de redundancia para emular lo que una unidad *cloudlet* debe contener Entonces con esto en mente definimos los siguientes requerimientos para la plataforma de pruebas

1 **Especificaciones de *hardware* del servidor**

   Multiples procesadores con multiples nucleos cada uno Al menos dos pro- cesadores con cuatro nucleos cada uno

   - Al menos cuatro giga-bytes (4GB) de RAM para todo el sistema
   - Por lo menos dos conectores de red Ethernet o un conector Ethernet y una conexión inalámbrica IEEE 802 11
   - Disco duro de cien giga-bytes (100GB) o más

2 **Especificaciones de *software* del servidor**

- Sistema operativo de red para servidores de código abierto (cualquier distribución de Linux para servidores)
- Piezas de código a ser utilizadas en la aplicación de prueba
- Planificador de recursos local

  Herramientas o código de medición de tiempo de ejecución de la aplicación de prueba

  Servicios de red como DHCP y *sockets* TCP

3 **Especificaciones de *hardware* del dispositivo móvil**

  Procesador de un sólo nucleo y baja frecuencia de operación

  Al menos ciento veintiocho mega-bytes (128MB) de RAM

  Almacenamiento Flash por lo menos un giga-byte (1GB)

  Conectividad WiFi

4 **Especificaciones de *software* del dispositivo móvil**

  Sistema operativo de software libre

- Plataforma de desarrollo de software libre

Estas especificaciones para la plataforma de pruebas son fáciles de cumplir con los equipos y tecnologías actuales Para este trabajo de investigación una de las pruebas que debemos realizar para la implementación de una nube distribuida para atender aplicaciones móviles es una prueba comparativa entre el tiempo de ejecución de una aplicación corriendo en el móvil y el de la misma aplicación modificada para acceder a una nube computacional Este experimento intenta probar la aceleración que puede

alcanzar una aplicación al utilizar los servicios de una nube computacional La aplicación de pruebas debe ser tal que pueda ser mejorada separando sus subtareas en diferentes unidades de proceso

Capítulo III

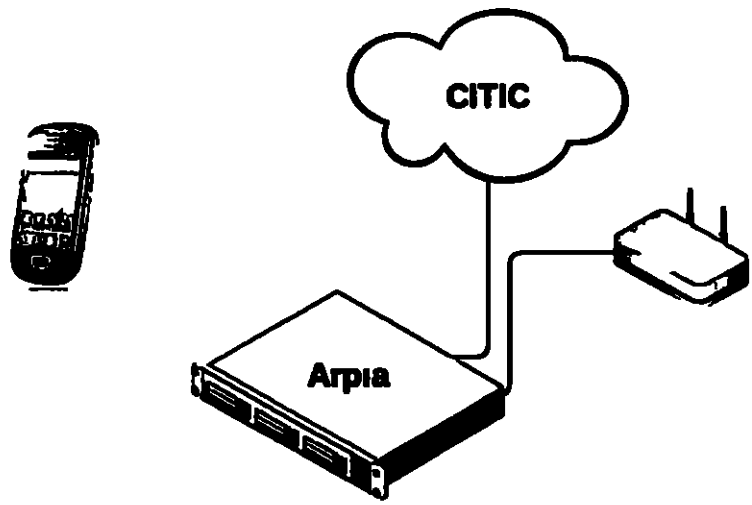
Diseño Metodológico e Implementación

La metodología utilizada en este proyecto de investigación es de tipo experimental En este capítulo se describe todo el diseno de la plataforma de pruebas que cumple con los requerimientos descritos en el capítulo anterior Está dividido en dos partes la primera describe el servidor utilizado y la segunda describe la aplicación creada tanto de control que se ejecuta dentro del móvil como la experimental que se ejecuta utilizando los servicios de la nube emulada El código fuente se encuentra en su totalidad en la dirección `https //code google com/p/tesis-oordww/` libre para descargar modificar y mejorar

3 1 El servidor Arpia

El servidor **Arpía** como se denominó a la plataforma de pruebas (ver Figura 3 1) es un servidor HP Proliant DL380 G7 con dos procesadores de seis nucleos cada uno trabajando a $2\,53GHz$ dieciséis giga-bytes (16GB) de RAM dos discos duros en configuración RAID y cuatro tarjetas de red Ethernet 10/100/1000 Le fue

distribución de Ubuntu Linux Server 12 04 LTS en versión de 64 bits Los módulos de servidor SSH los módulos de servidor de ejecución remota creado para esta investigación y las piezas de código conforman el conjunto de software instalado en **Arpía** De las cuatro tarjetas de red se utiliza una para conectarse a la red local e Internet y otra se conecta a un punto de acceso inalámbrico que permite la comunicación con el dispositivo móvil Al separar las redes se reduce el tráfico de la red local y los mensajes de difusión el unico tráfico será entonces el intercambio de datos entre el dispositivo móvil y el servidor Esta configuración está basada en el modelo de una nube computacional distribuida que utiliza *cloudlets* donde los nodos se comunican a través de conexiones de alta velocidad y conexiones físicas dedicadas y los clientes móviles a través de conexiones WiFi o redes de datos móviles



F gure 3 1 Pl taf rma d pruebas

3 1 1 Configuración de Arpía

El servidor a ser utilizado debe contener todos los elementos para emular un *cloudlet* En este caso se habilitó un servicio que recibe todas las peticiones de almacenamiento (upload) de datos ejecución de las piezas de código asignación de recursos y descarga (download) de datos En un principio se pensó crear un módulo dentro del sistema

dentro del sistema operativo Linux y recompilar una nueva versión para este proyecto Pero el tiempo de desarrollo y la cantidad de código a estudiar y modificar era demasiado en el tiempo previsto por eso se intentó crear una aplicación en lenguaje C que pudiese incluirse fácilmente en el nucleo de Linux luego Esta opción probó ser muy desafiante debido a que debía comunicarse mediante *sockets* con el lenguaje Java Al parecer los dos lenguajes tratan la información de manera diferente a pesar de ser una comunicación estándar hubieron problemas de sincronización de los datos y así no era posible medir de forma precisa el tiempo de ejecución de la aplicación de pruebas

Al final se optó por crear una clase abstracta en lenguaje Java que pudiese dar paso a todas las versiones de los objetos que serían utilizados para el experimento Luego para poder transmitir la información se utilizaron dos objetos subclases de la clase `Thread` que funcionan como servicios de transporte y sincronización de datos de forma independiente tenemos un canal de transmisión y otro de recepción de los datos

3 1 2 Servicio de ejecución remota

El servicio de ejecución remota es la base de la aplicación distribuida y una implementación sencilla del tipo de servicios que se desean implementar en los *cloudlets* Este fue creado en lenguaje Java y consta de cinco clases principales En la Figura 3 2 se muestra el diagrama de clases del servicio de ejecución remota y en el anexo C se muestra el código fuente de las clases desarrolladas

En la Figura 3 3 se muestra la clase `CommandWrap` esta sirve como envoltorio para el envío de objetos del móvil a la plataforma de pruebas y viceversa El código fuente de la clase `CommandWrap` se encuentra en el anexo C III Este tipo de técnicas pueden ser complementadas con encriptación y certificación de los datos aumentando la

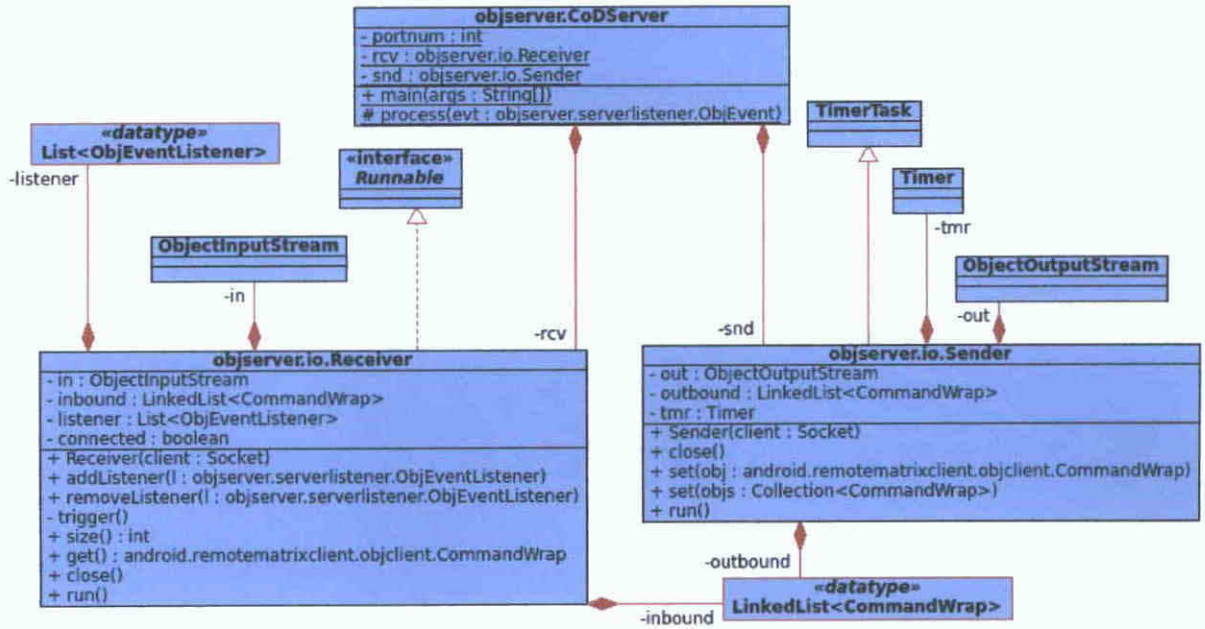


Figure 3.2: Diagrama de clases UML del servicio de ejecución remota

que es parte de lo que se intenta en este proyecto, pero el paradigma utilizado es el de utilizar archivos estándar como XML para definir el trabajo que la aplicación está solicitando (XMLRPC), haciendo fácil al desarrollador la creación de aplicaciones para usuarios finales.

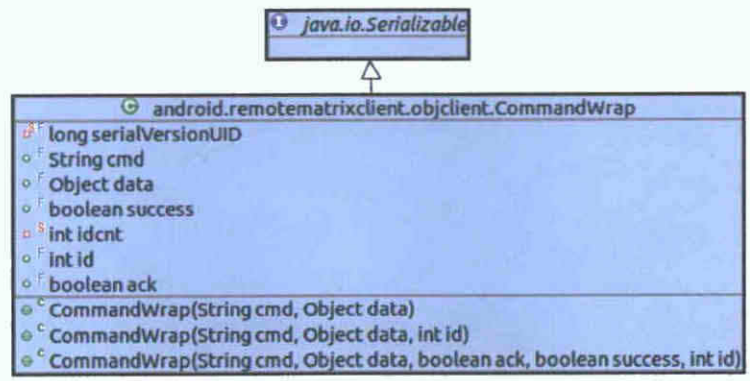


Figure 3.3: Clase `CommandWrap`

Para poder utilizar eficientemente el tiempo del procesador se creó la clase `ObjEvent` que se muestra en la Figura 3.4, su objetivo es notificar a todos los objetos registrados

que ocurrió un evento, similar a como funciona una petición de interrupción (IRQ). Este posee cuatro métodos que permiten a los objetos externos obtener información sobre los datos recibidos y permiten descargar los datos. Para poder recibir la notificación de eventos de tipo `ObjEvent` las clases deben implementar la interfase `ObjEventListener`, para definir el comportamiento del objeto frente a un evento.

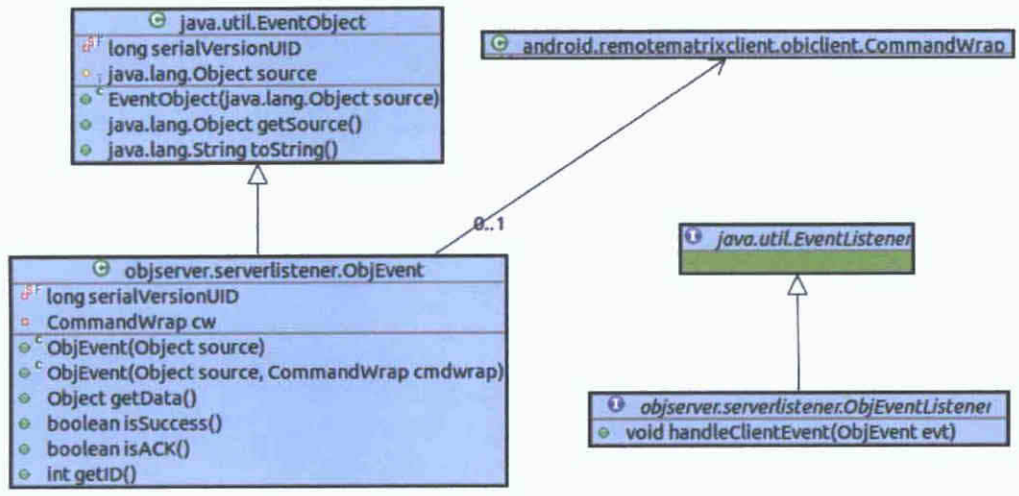


Figure 3.4: Clase `ObjEvent` e interfase `ObjEventListener`

3.1.3 Las piezas de código

El paradigma de CoD utilizado para crear la plataforma de pruebas trabaja sobre el concepto de que existe una aplicación a partir de múltiples aplicaciones sencillas, conocidas como piezas de código, que permiten distribuir las tareas de la aplicación en múltiples procesadores y aumentan la seguridad de la aplicación; al separar los contextos de los datos de entrada y salida, impidiendo la distorsión de los datos. Se hace necesario mencionar entonces las piezas de código utilizadas para la aplicación de multiplicación de matrices que creamos. Estas tenían como entradas archivos o secciones de archivos y su salidas eran archivos que son respuestas parciales o totales de la aplicación. Para este proyecto se crearon tres piezas de código: `mult`, `join`

y `split_stripes`, cuyo código fuente se encuentra en el Anexo A. Las piezas de código fueron creadas en lenguaje C, puesto que así es más sencillo ejecutarlas como aplicaciones del sistema operativo. A diferencia de usar el lenguaje Java que corre cada instancia de la aplicación sobre una máquina virtual, utilizando más recursos del sistema. También es posible convertir este código en un módulos del Kernel de Linux, de ser necesario para mejorar el rendimiento.

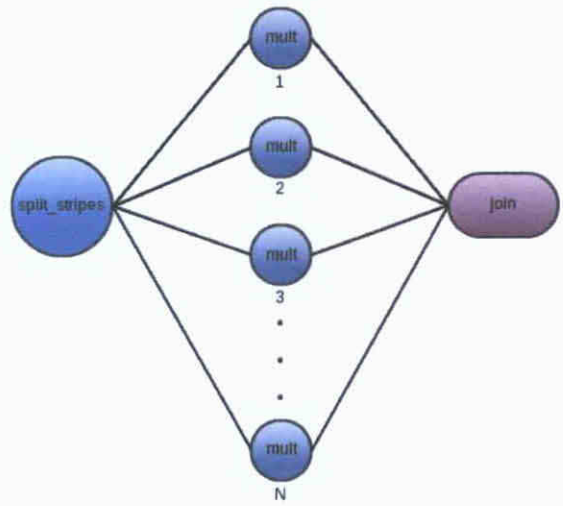


Figure 3.5: Estructura de la aplicación remota

La estructura de la aplicación remota se muestra en la Figura 3.5, es una estructura básica que divide el trabajo entre la cantidad de unidades de procesamiento disponibles. Al dividir las operaciones en grupos mejora el tiempo de ejecución puesto que las operaciones se dividen de forma equitativa y la sobrecarga debido a la división y reunión de las tareas se reduce al máximo. La pieza `mult` acepta parte de una matriz, dividida por la pieza `split_stripes`, y crea un archivo intermedio con parte de la respuesta; luego la pieza `join` reúne estas respuestas intermedias en un archivo matriz, que puede leer el objeto `FileMatrix` dentro del servidor.

3 2 Configuracion del movil de prueba

La plataforma de pruebas descrita para este proyecto incluye una aplicación móvil que realice operaciones de matrices más específicamente se escogió la operación de multiplicación de matrices por ser altamente paralelizable y fácil de implementar En un procesador sencillo esta operación se realiza mediante una iteración que resuelve las operaciones una tras otra ésto aumenta el tiempo de ejecución de forma exponencial al aumentar el tamano de las matrices Para reducir el tiempo de ejecución se requiere gran poder computacional procesadores más rápidos o multiples procesadores Las operaciones con matrices están en muchas de las aplicaciones que procesan datos Aplicaciones de realidad aumentada videojuegos y reconocimiento de patrones requieren de operaciones con matrices para su funcionamiento

El móvil utilizado es un Samsung Galaxy Mini (S5570) Este móvil tiene las siguientes especificaciones

- Bandas de operación GSM 850 / 900 / 1800 / 1900 (HSDPA 900 / 2100)

  Procesador ARM 9 @ 600MHz (1 nucleo)

- RAM 384MB

  Flash interno 160MB MicroSD 4GB

  Sistema Operativo Android 2 3 4

  WiFi IEEE 802 11b/g

Estas especificaciones concuerdan con el tipo de dispositivos móviles que propician este trabajo de investigación Éstos poseen limitados recursos como un sólo nucleo por ende todas las funciones dentro del sistema deben competir por utilizar el procesador este asigna prioridades a los procesos donde las aplicaciones son las de menor

prioridad Su consumo energético es relativamente bajo comparado con el de dispositivos con varios nucleos y su costo es mucho menor A continuación se detalla la aplicación móvil que se utilizó para realizar los experimentos con la plataforma de pruebas

3 2 1 La aplicación RemoteMatrixCalculator

La aplicación móvil RemoteMatrixCalculator fue creada en Android y tiene como finalidad recopilar la información de las operaciones con matrices que se ejecutan de forma local y remota En la Figura 3 6 se muestra el diagrama UML de clases de la aplicación móvil Ésta contiene la actividad principal de la aplicación ésta utiliza una barra de progreso para mostrar el avance de la ejecución de las tareas programadas y un cuadro de texto que muestra los resultados de las operaciones completadas Además dentro de la clase principal se crearon tres clases privadas `MainTask` `LocalTask` y `RemoteTask` La clase `MainTask` extiende de la clase `AsyncTask` que permite la interacción entre una actividad de Android y un hilo donde se ejecutan tareas que de otra forma bloquearían el normal desempeno de la aplicación

Las clases `LocalTask` y `RemoteTask` implementan la interfase `Task` que es utilizada para obtener información de las tareas realizadas por las clases de este tipo El propósito de estas dos clases es realizar una multiplicación de matrices una local y la otra remota como lo indican sus nombres y medir el tiempo que le toma ejecutar estas tareas registrándolas para que la clase `MainTask` pueda almacenar estos datos en un archivo de bitácora La interfase `Task` extiende de la interfase `Runnable` convirtiendo a todas las subclases de `Task` en candidatos para ser ejecutados en hilos Una vez la clase `MainTask` acaba con su ejecución la aplicación se cierra automáticamente

Durante la creación de la aplicación nos topamos con diferentes limitaciones que no habían sido consideradas en un principio como la cantidad de RAM que se necesita

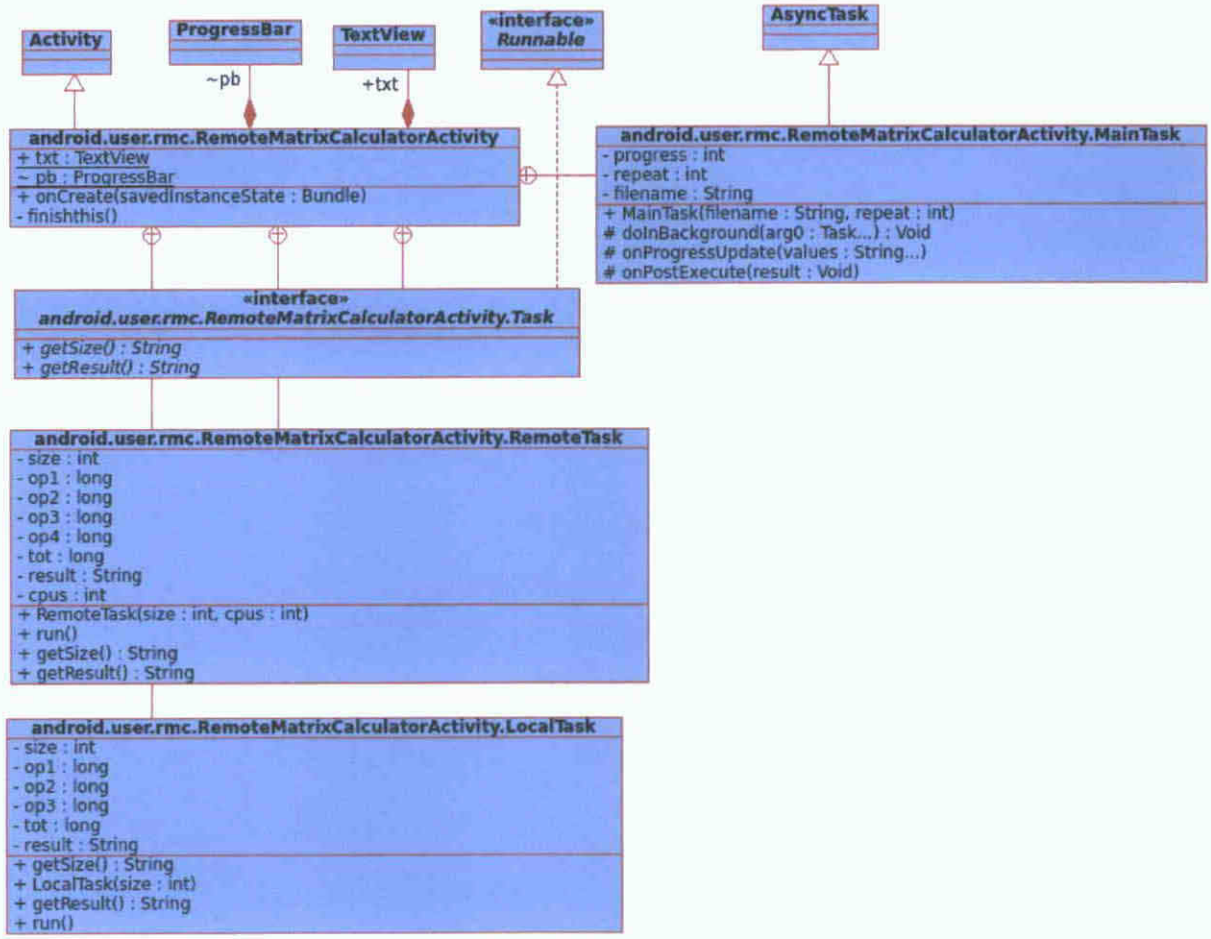


Figure 3.6: Diagrama UML de clases de la aplicación móvil.

para la aplicación o el hecho de que el recolector de basura (GC) de Java no eliminaría los registros de memoria no utilizada. Para esto se tuvo que mejorar el código y adecuarlo a las limitaciones de espacio que se tienen en el móvil. Se realizó una corrección del código fuente para evitar el goteo de memoria (*memory leak*); al parecer, los objetos que trabajan con *sockets* mantienen referencias a los datos enviados por este medio, incluso cuando los objetos habían sido descartados por la aplicación. Una vez descubierto este hecho se procedió a incluir líneas que eliminasen la información de los datos enviados y recibidos por el servicio de envío de objetos.

3 2 2 El cliente del servicio de ejecución remota

Junto con la aplicación RemoteMatrixCalculator se creó un un cliente para el servicio de ejecución remota La clase `ObjClient` se muestra en la Figura 3 7 Esta utiliza los servicios de las clases `Receiver` y `Sender` para enviar y recibir objetos entre el móvil y el servidor de pruebas El código fuente de estas clases se encuentran en el anexo C-I y C II

El cliente del servicio de ejecución remota se creó como una clase final impidiendo que multiples instancias de la clase `ObjClient` sean creadas dentro del móvil Al tener sólo una instancia para todas las aplicaciones se controla la transferencia de objetos aumentando la seguridad y evitando que código malicioso acceda furtivamente al servidor de ejecución remota También este servicio permite a los objetos enviar datos sin necesidad de crear nuevas conexiones a la red Esto evita que se reserven recursos cada vez que se envían datos En un sistema donde siempre se está conectado un servicio de este tipo es necesario Las capas más bajas de la arquitectura TCP/IP en este caso deben entonces encargarse de mantener la conexión incluso cuando se cambia de celda y de *cloudlet* Esta tarea deberá ser descrita en versiones futuras de este sistema

3 3 Las clases de matrices

Para realizar una comparación de resultados se creo una clase abstracta en Java para Android la clase `AbstractMatrix` (ver Figura 3 8) De la clase abstracta `AbstractMatrix` se derivan tres clases que fueron utilizadas para realizar operaciones con matrices la clase `Matrix` que realiza operaciones con matrices utilizando los recursos del móvil y las clases `RemoteMatrix` y `FileMatrix` que actuan en conjunto entre el móvil y el servidor de ejecución remota Los datos de la instancia de

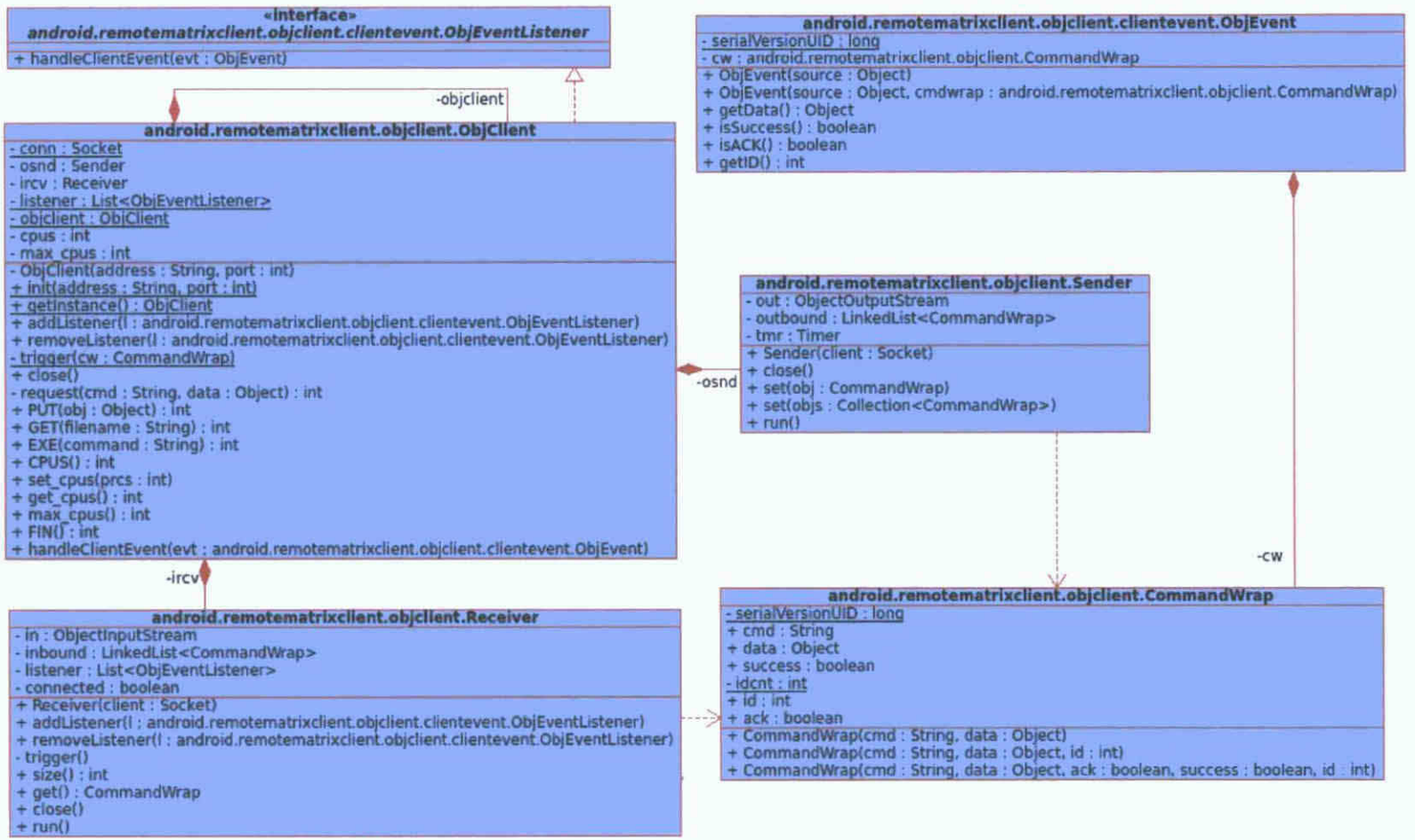


Figure 3.7: Diagrama UML de clases del cliente del servicio de ejecución remota.

`RemoteMatrix` se transmite, a través de la conexión inalámbrica, al servidor de ejecución remota y este crea una instancia de `FileMatrix` y esta instancia crea un archivo de matriz (.mat), el cual puede ser utilizado para realizar operaciones distribuidas. Una vez que una operación se ha completado se crea un archivo de matriz, una instancia de `FileMatrix` puede entonces enviar los datos a una instancia de `RemoteMatrix` para mostrar el resultado de una operación. Los códigos fuente de todas estas clases se encuentran en el anexo B.

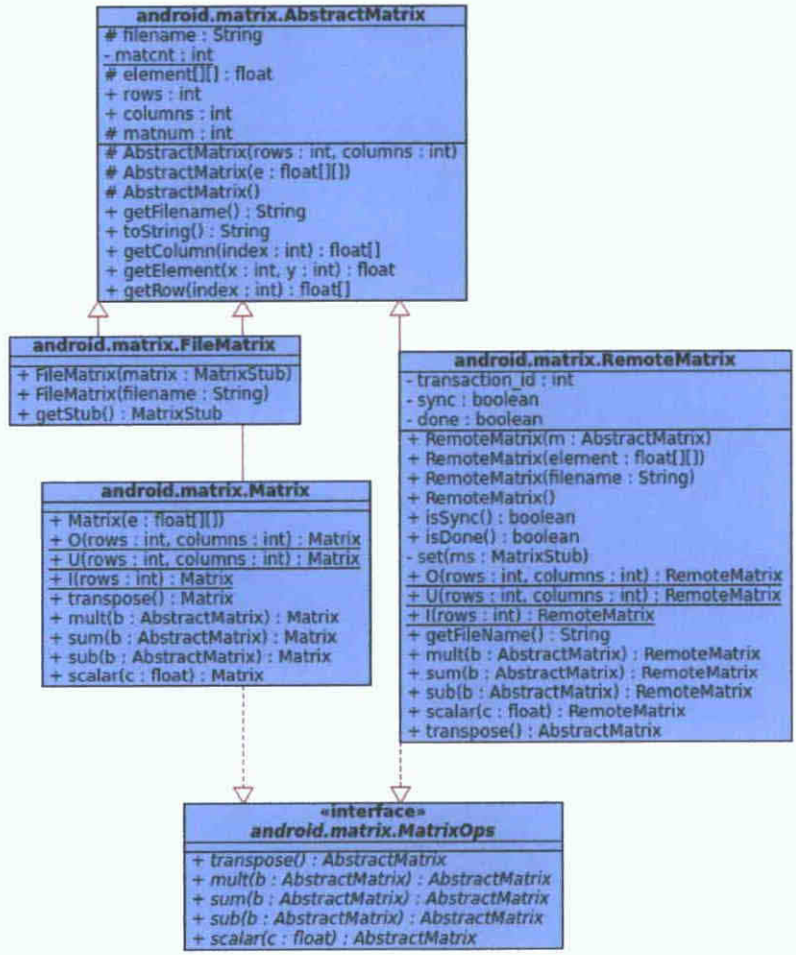


Figure 3.8: Diagrama UML de clases de matrices

Las operaciones con matrices, entonces, no están definidas en todas sus subclases. Las subclases que requieren de operaciones con matrices deben implementar la inter-

fase `MatrixOps` cuyo código se encuentra en el anexo B-II `FileMatrix` por ejemplo no requiere implementar la interfase `MatrixOps` debido a que el servicio de ejecución remota realiza estas operaciones por ella No se utilizaron hilos en la implementación remota debido a que en Java los hilos se asignan por división de tiempo del procesador lo cual no concuerda con el esquema de asignación a multiples procesadores

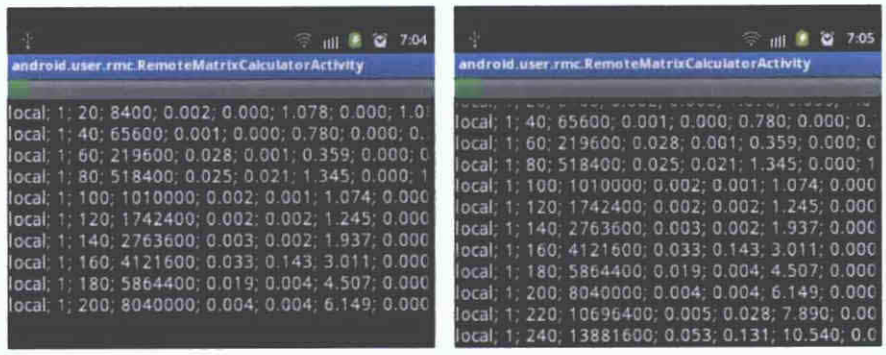
3 4 Diseño experimental

Una vez completado el diseno de la plataforma de pruebas se procedió con el experimento El experimento tiene como objetivo el reunir información acerca del rendimiento de una aplicación móvil que resuelve una gran cantidad de operaciones Como la aplicación se diseñó para realizar ambas pruebas la prueba local y las pruebas con diferentes numeros de procesadores remotos pues se configuró una conexión con el servidor de pruebas y se inició una corrida de la aplicación donde se realizaban diferentes operaciones con matrices de diferentes tamanos las matrices empezaban en tamano 20×20 e iban aumentando de veinte en veinte hasta un tamano máximo 800×800 La cantidad de procesadores se escogieron en numeros impares empezando por tres aumentando en dos hasta once

Además descubrimos que todo evento en el móvil afecta el tiempo de ejecución tanto local como remota así que se repitió cada operación cinco veces para lograr un promedio en el tiempo de ejecución y un error porcentual debido a interrupciones en la ejecución de cualquier proceso en el móvil Es posible que en el servidor de ejecución remota ocurra lo mismo pero este tiempo es prácticamente despreciable debido a que existen otros procesadores para ejecutar todas las tareas del servidor Y ya que los procedimientos se asignan arbitrariamente a los nucleos de procesamiento existen pocas posibilidades que un procesador esté ocupado cuando un proceso le sea asignado Aun si lo estuviese se ejecutaría como un hilo en la división de tiempo del

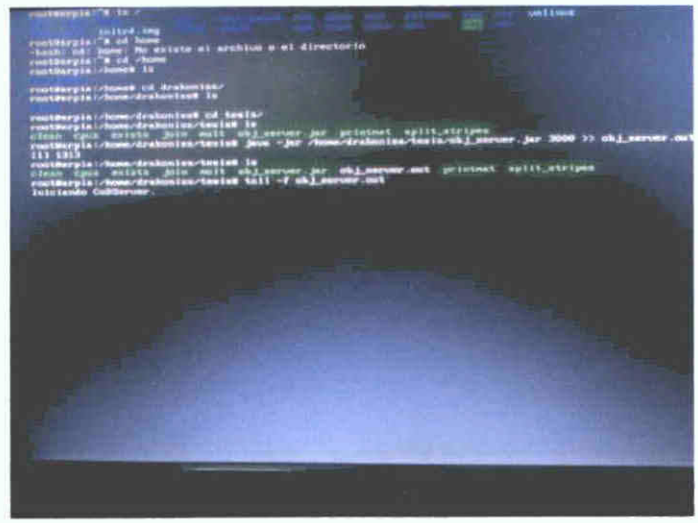
procesador y estaría en cola sólo por unos cuantos microsegundos

La ejecución de la aplicación tomó cerca de las siete horas En la Figura III 9(a) y III 9(b) se muestran algunas capturas de pantalla de la aplicación mientras que en la aplicación móvil y en la Figura III 9(c) III 9(c) y III 9(c) se muestran la aplicacion en la plataforma de desarrollo Durante este tiempo las unicas interrupciones fueron para revisar como iban los resultados en la pantalla táctil y otros servicios que se ejecutan tras bastidores en el móvil El móvil estaba conectado por WiFi al servidor que a propósito no permitía el paso de paquetes de Internet por lo cual ninguna aplicación que utiliza la web podía funcionar durante el experimento En un entorno donde otras aplicaciones coexisten con el servicio de ejecución remota se puede pensar que se encuentren más inconvenientes para la ejecución de aplicaciones distribuidas eso si las aplicaciones que acceden a servicios web no se ejecutan de forma distribuida también A continuación se presentan los resultados de este experimento y algunas conclusiones obtenidas durante todo el proceso de diseno e implementación de la plataforma de pruebas así como de los datos obtenidos Luego se presentan algunas interrogantes que en trabajos futuros podrían dar respuesta a aspectos teóricos y experimentales excluidos del presente trabajo



(a) (b)

(c) (d)



(e)

Figure 3.9: Capturas de pantalla de la aplicación en móvil.

Capítulo IV

Análisis de Resultados

Una vez completado el experimento se procedió al análisis de los datos En este capítulo se muestran los gráficos de los resultados del experimento mientras que en el anexo D se muestran tablas con los datos obtenidos luego se presenta un análisis de los resultados Durante la implementación de la plataforma de pruebas se definieron los tipos de medidas del rendimiento de un programa en ejecución Una de las medidas más importantes y la que está directamente relacionada a la percepción del usuario es el tiempo de ejecución La optimización del tiempo de ejecución en una aplicación especialmente en una de tiempo real es crucial para su éxito Otra de las caracterizaciones que se realizaron fue la tasa de intercambio de datos en el canal inalámbrico Se definieron otras medidas importantes que arrojan datos estadísticos interesantes como la tasa de operaciones de punto flotante por segundo (FLOPS) y índice de aceleración del proceso remoto contra el proceso local Estas dos ultimas con la intención de obtener información relevante matemáticamente comprobable que sirva para modelar el sistema luego

4 1 Tiempo de Ejecucion

La Figura 4 1 muestra los gráficos individuales de la mediciones del tiempo de ejecución y su respectiva desviación estándar En el Anexo A se encuentran las tablas con los valores mínimos promedios y máximos obtenidos durante el experimento Las variaciones aumentan con la reducción de los procesadores disponibles es posible que esto se deba a la asignación no priorizada de los subprocesos En la aplicación remota el servidor debe asignar los nucleos pero en vez de utilizar un planificador heurístico simplemente estamos asignando los trabajos en orden sin prioridades La razón para esto es que no se supone que ninguna otra aplicación esté corriendo en el servidor pero los diferentes servicios básicos del *kernel* no fueron cancelados para evitar errores que impidiesen la continuidad del experimento desestimando que su ejecución fuese a crear grandes cambios

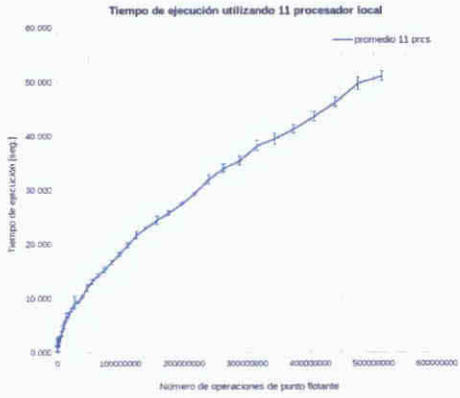
En el gráfico de la Figura 4 2 se muestra el tiempo de ejecución de la aplicación en el móvil y se compara con la aplicación remota utilizando el servidor de ejecución con diferente numero de procesadores La variable independiente es la cantidad de operaciones de punto flotante necesarias para realizar una multiplicación de matrices cuadradas La cantidad de operaciones de punto flotante en una multiplicación de matrices está dada por el procedimiento de multiplicación de matrices el cual es la sumatoria del producto de una fila por una columna como sigue

Sean $A_{k\times k}$ y $B_{k\times k}$ dos matrices cuadradas entonces

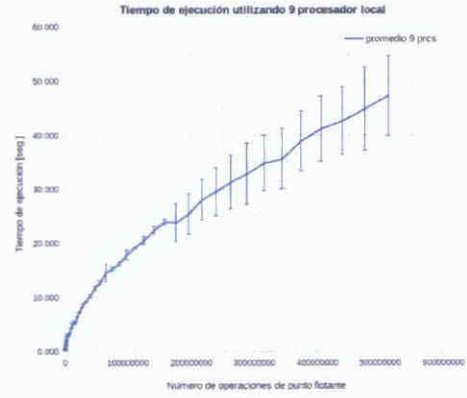
$$C = A \times B = (c_{i,j})_{k\times k}$$

donde cada elemento $c_{i,j}$ está dado por

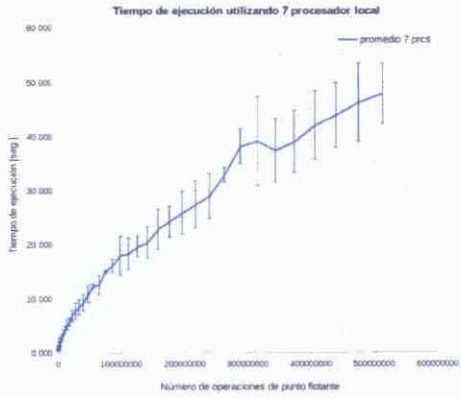
$$c_{i,j} = \sum_{=1}^{k} a_{i} \times b_{,j}$$



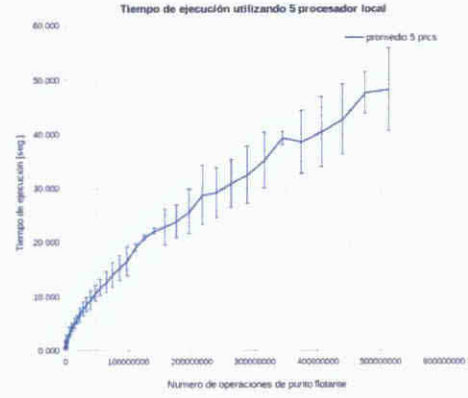
(a) 11 procesadores remotos



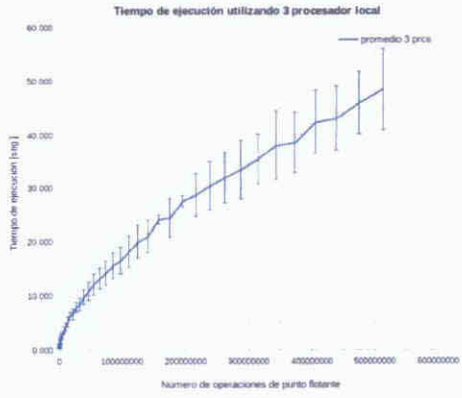
(b) 9 procesadores remotos



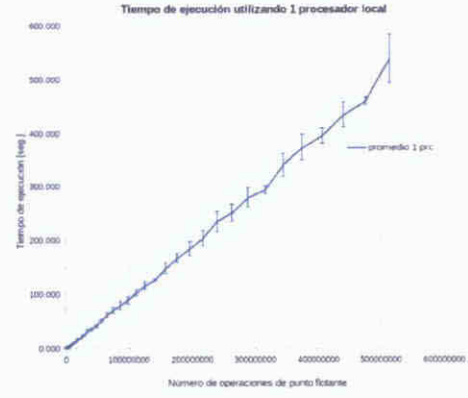
(c) 7 procesadores remotos



(d) 5 procesadores remotos



(e) 3 procesadores remotos



(f) 1 procesador local

Figure 4.1: Tiempo de ejecución y desviación estándar de las pruebas realizadas.

entonces por cada elemento requiere $k+1$ operaciones de punto flotante Al final el numero total de operaciones será

$$\begin{aligned} Ops &= k^2(k+1) \\ Ops &= k^3 + k^2 \end{aligned} \tag{4 1}$$

de esta forma el numero de operaciones de punto flotante crece de manera polinomial potencial Esto facilita la medición del tiempo de ejecución debido al crecimiento rápido en el numero de operaciones a realizar descubriendo las verdaderas capacidades del dispositivo móvil

La gráfica en la Figura 4 2 podría dividirse en dos zonas una donde el móvil resulta ser más efectivo que el servicio remoto pero el servicio remoto no lo hace tan mal y otra donde el servicio remoto es más efectivo que el móvil Una razón para que los móviles sean tan efectivos es debido a que sus aplicaciones utilizan poco al procesador una aplicación con procesamiento intensivo puede crear una percepción de ineficiencia y mala codificación La percepción entonces juega un papel determinante en cuántas personas están dispuestas a instalarla en su móvil y utilizarla Muchas aplicaciones actuales recurren entonces a la utilización de servicios en la nube para realizar cálculos intensivos el inconveniente de los sistemas de nube actuales es la latencia de la red

4 1 1 Distribución del tiempo de ejecución

En una red celular el móvil transmite todos los datos que tiene en espera con una latencia aproximada de tres segundos en cambio en una red WiFi la transmisión es casi inmediata Aun con esta ventaja existen problemas intrínsecos del canal de comunicaciones debido a ruido e interferencia co-canal que afectan la comunicación e impiden una transferencia constante de los datos Durante el experimento se utilizó un punto de acceso WiFi IEEE 802 11b

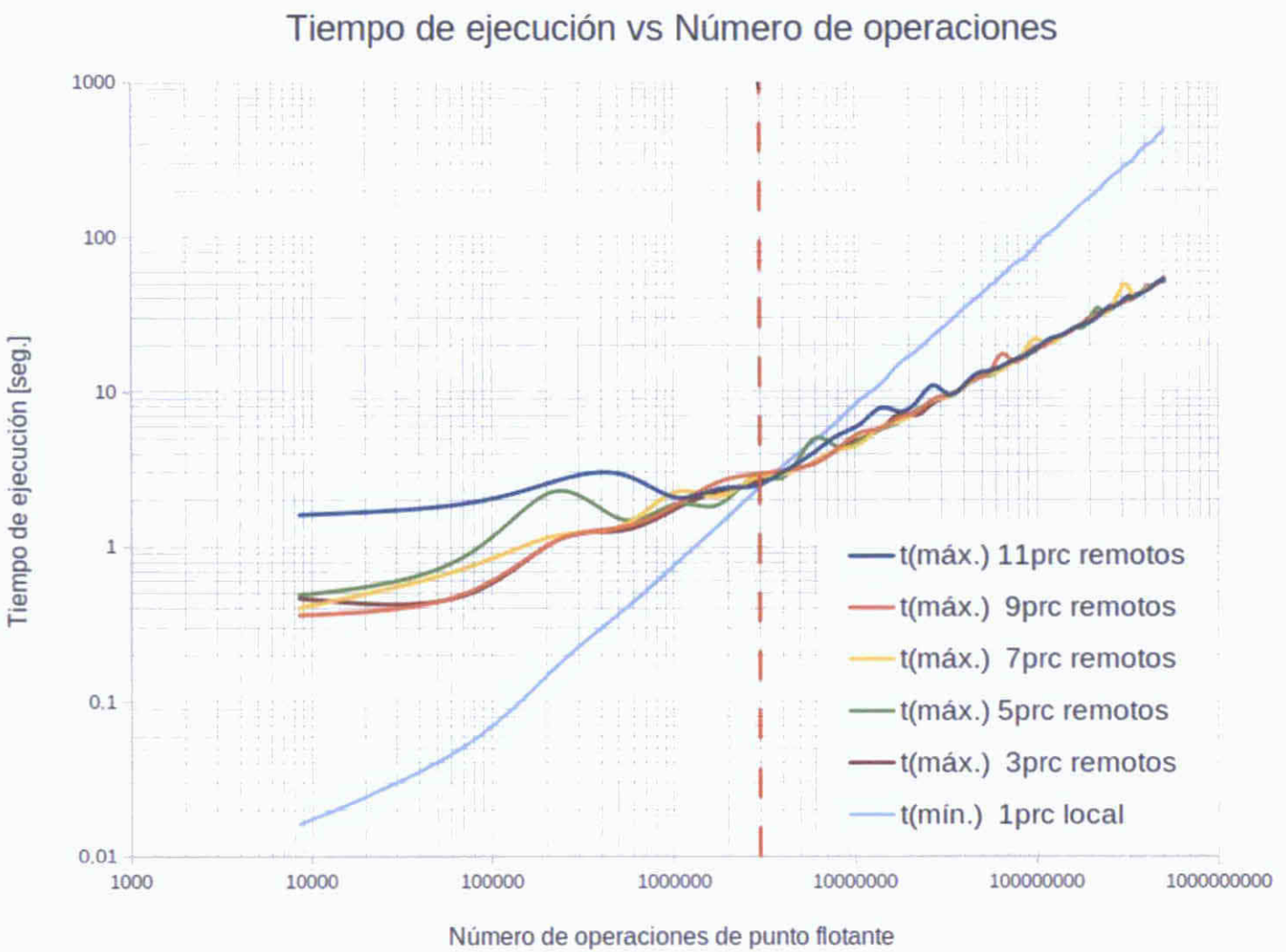


Figure 4.2: Tiempo de ejecución vs. Número de operaciones

g para la comunicación con el móvil El canal de comunicaciones que se usó en las pruebas posee una tasa de transferencia de entre 9Mbps y 12Mbps estas mediciones se realizaron utilizando la aplicación iperf Y aun cuando esta es la tasa de transmisión medida es posible que la interferencia del canal causada por otros puntos de acceso reduzca grandemente la tasa de transmisión

Los datos recopilados durante el experimento muestran que gran parte del tiempo de ejecución se debe a la sincronización de los datos con el servidor Hasta un máximo de 99 3% del tiempo total de ejecución se debe a la sincronización de datos hacia y desde el servidor Mientras que en la aplicación local es la ejecución la que requiere entre el 97% y el 99% del tiempo de ejecución Tal y como muestra la Tabla D 8 del Anexo D

Los datos para el proceso remoto muestran que el tiempo utilizado para realizar los cálculos y la sincronización de datos es mayor al 100% esto se debe a la multiplexación que realizan los *sockets*TCP La Figura 4 3 muestra un diagrama de tiempo de la ejecución de las subtareas de la aplicación remota Aquí se muestra como la matriz A se crea y se transmite e inmediatamente la matriz B es creada y transmitida al servidor remoto El tiempo de diferencia entre ambas transmisiones genera un retardo en la transmisión de la matriz B

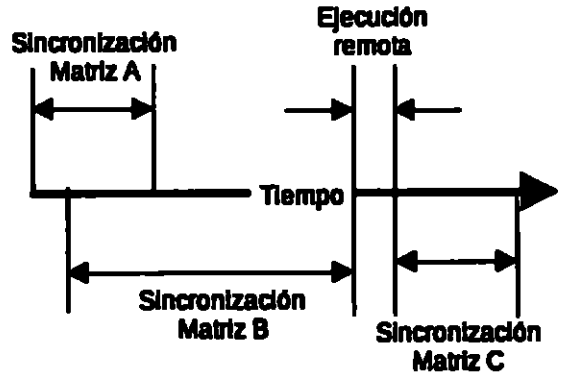


Figura 4 3 Diagrama de la distribución del tiempo de ejecución en el servidor remoto

4 2 Rendimiento de la aplicacion

Durante un experimento con sistemas distribuidos es normal que se mida el rendimiento del computador Esto es la cantidad de operaciones que se pueden realizar por segundo En este caso particular se está midiendo el numero de operaciones de punto flotante por segundo que se requieren para realizar las tareas asignadas tanto en el móvil como en el servidor de ejecución remota En el caso del HP Proliant con dos procesadores Intel Xeon E5649 tenemos que la cantidad máxima de operaciones de punto flotante que se pueden obtener están basadas en la siguiente ecuación

$$FLOPS = \#cores \times\ prc\ \ freq\ \times\ \frac{1}{\frac{\#cycles}{flop}} \tag{4 2}$$

donde

$\#cores$ es el numero total de nucleos de procesamiento en el nodo servidor

$prc\ \ freq$ es la frecuencia de los procesadores

$\frac{\#cycles}{flop}$ es el numero de ciclos de máquina que requeridos para ejecutar una operación de punto flotante

En la mayoría de los procesadores Intel actuales las operaciones de punto flotante de doble precisión se ejecutan en cuatro ciclos En procesadores ARM como el que se encuentra en el móvil la cantidad de instrucciones para operaciones de punto flotante es mayor Pero si asumimos este numero es correcto para el procesador Intel Xeon E5649 estimaremos entonces que el mínimo de operaciones de punto flotante que se pueden realizar en un segundo serán(segun las especificaciones en la sección 3 1)

$$FLOPS = 12 \times\ 2\,53 \times 10^9 \times\ 1/4$$

$$FLOPS = 7\,59 \times 10^9$$

En la Figura 4 4 se muestra la cantidad de FLOPS que se utilizaron durante la

ejecución del experimento En este gráfico se nota que el móvil posee un rendimiento más o menos constante durante la ejecución de las tareas Al contrario de las ejecuciones utilizando el servicio remoto donde el numero de FLOPS aumenta al aumentar la cantidad de operaciones esto se debe principalmente a que las tareas programadas no superan el numero de FLOPS máximo del servidor En este caso la cantidad de operaciones realizadas por numero de tiempo aumentan en base al tiempo de ejecución Algo peculiar en este gráfico es que al aumentar la cantidad de nucleos de procesamiento llega el momento donde el proceso de división de tareas/reunión de resultados afecta negativamente la eficiencia de la aplicación Aun cuando el procesamiento se realiza muy rápido el numero de procesos creados y su asignación toman más tiempo que cuando se utilizan menos procesadores

Si realizamos una observación más detallada del gráfico en la Figura 4 4 notaremos que el gráfico de nueve procesadores lleva el mismo comportamiento que el gráfico de once procesadores pero en un punto este cambia su comportamiento al de los otros gráficos de menor cantidad de procesadores El gráfico de siete procesadores tiene un comportamiento parecido pero el cambio se realiza mucho antes Una razón para esto es debido a que la cantidad de tareas y operaciones que debe realizar cada procesador al aumentar la cantidad de tareas por procesador la eficiencia de todo el sistema aumenta. Esto sugiere que existe una relación entre la cantidad de datos a procesar y la cantidad óptima de procesadores para realizar el trabajo de forma eficiente

4 3 Indice de aceleración del servicio remoto

La Figura 4 5 muestra una comparación entre el servicio local y el servicio remoto en ella se muestra la disminución del tiempo de ejecución al aumentar la cantidad de procesadores lo que se puede considerar un índice de aceleración en la ejecución de la

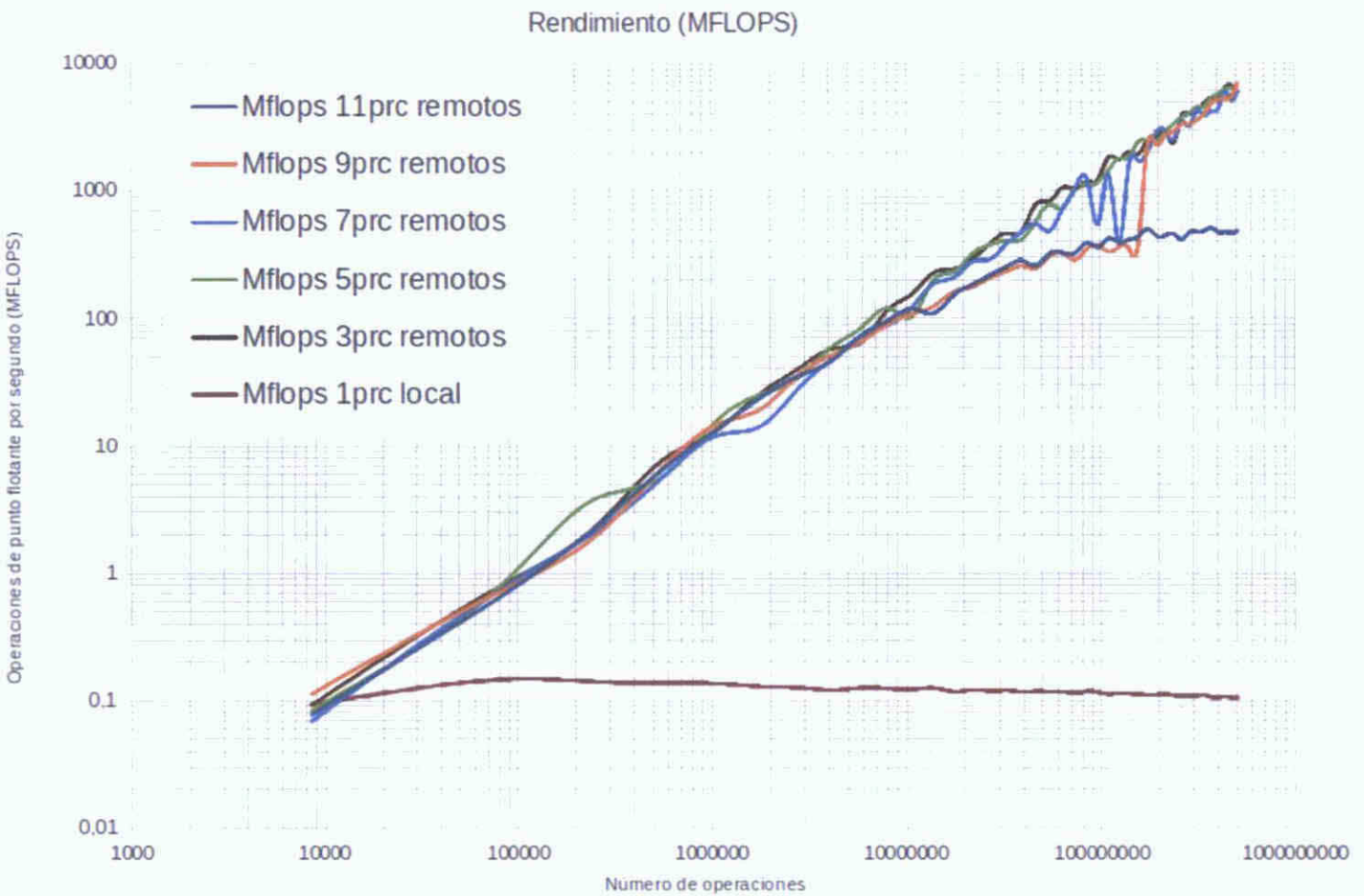
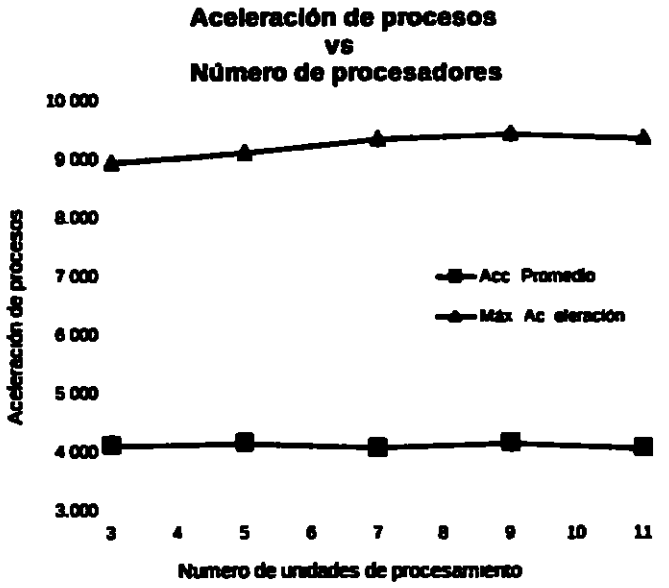


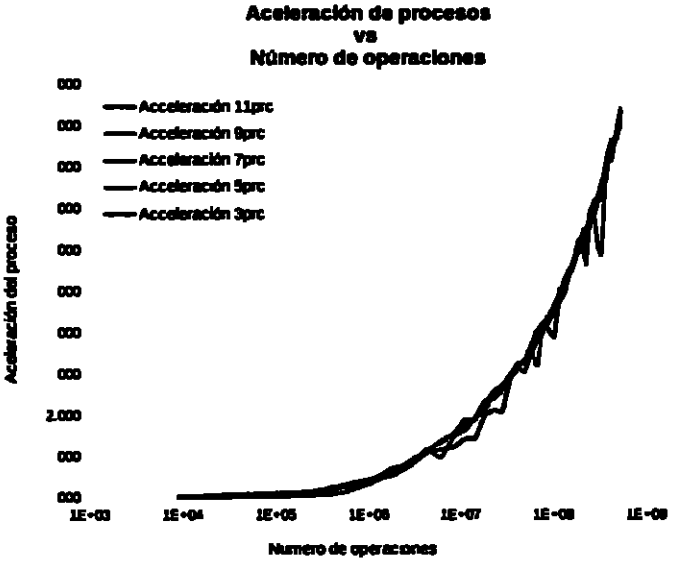
Figure 4.4: Número de operaciones de punto flotante por segundo (MFLOPS).

aplicación En la Figura IV 5(a) se muestra el índice de aceleración del procesamiento promedio contra la cantidad de procesadores Aquí se muestra que el índice de aceleración es prácticamente el mismo para las diferentes configuraciones de procesadores Segun la Ley de Amdahl esto se debe a que hemos llegado a la máxima aceleración posible ya que la porción de código que no puede ser paralelizada es mucho mayor que la que sí puede ser paralelizada Para mejorar el rendimiento en este caso se debe optimizar la sección del código que no puede ejecutarse en paralelo como la transmisión de datos mediante compresión la división los datos de entrada y la concatenación de los datos de respuesta mediante optimización de código En cambio la Figura IV 5(b) muestra que la aceleración aumenta a medida que la cantidad de operaciones de punto flotante aumenta Se puede interpretar esto como una confirmación de la Ley de Amdahl también nos dice que si seguimos aumentando la cantidad de operaciones eventualmente aumentará la aceleración del proceso hasta saturar la capacidad de los procesadores En este gráfico no alcanzamos tal saturación de los procesadores ya que para la aplicación local no era posible obtener datos de entrada Debido a que esta es una comparación directa del tiempo promedio de ejecución local contra el tiempo promedio de ejecución remota si no contamos con alguno de los datos la comparación no puede llevarse a cabo

En el caso particular de la aplicación remota, puede ser posible reducir el tiempo no paralelizable de la aplicación mediante la optimización del código Hasta ahora se han utilizado capas de *software* que no se habían contemplado en el modelo original como la máquina virtual de Java y los sockets orientados a objetos Ambos adicionan código que en teoría aseguran la transmisión de datos sin que el usuario tenga que preocuparse por el funcionamiento Es posible que una versión más ligera del modelo OSI deba ser implementada en un sistema de nube distribuida Ciertamente las direcciones físicas de la capa de enlace de datos son necesarios mas no así las direcciones de la capa de



(a) Aceleración vs número de procesadores



(b) Aceleración vs número de operaciones

F gure 4 5 Aceleración obten d ut l zando el serv do emoto

red Sería entonces posible que una aplicación crease una conexión libre de errores sin necesidad de una capa de encaminamiento puesto que todos los nodos son locales y están a un salto de distancia

Es entonces el *cloudlet* el encargado de realizar las comunicaciones con la Internet y otros miembros de la nube En este caso una unidad de este tipo debe implementar

la capa de red en su totalidad adicionalmente de una capa de mantenimiento de los recursos disponibles en la nube con tablas parecidas a las tablas de encaminamiento utilizadas en la capa de red Esto permitirá una utilización más eficiente de los recursos y la consecuente reducción del tiempo necesario para tareas no paralelizables

En el siguiente capítulo se amplían las conclusiones obtenidas a partir de este análisis de los resultados experimentales obtenidos Luego se definirán posibles líneas de investigación que se derivan de este proyecto y que complementan el trabajo presentado en este documento

Capítulo V

Conclusiones

Luego de la experimentación y el análisis de los datos obtenidos podemos concluir que la capacidad de procesamiento de un móvil puede ser incrementada mediante la utilización de una nube computacional que realice el procesamiento de los datos y devuelva sólo los resultados al móvil Esto cumple con el objetivo general de este proyecto de investigación Para esto definimos una nube computacional distribuida, en los capítulos I y II que permite reducir la latencia de transmisión y recepción de mensajes además distribuye el costo energético de la nube a los nodos que voluntariamente se agreguen a la red y distribuye los recursos computacionales de forma tal que se crean áreas de cobertura que garanticen la disponibilidad de recursos Se definió el modelo básico y los requerimientos de *software* y *hardware* cumpliendo con los dos primeros objetivos específicos En el capítulo III se presentan los desarrollos obtenidos de la maqueta de pruebas una emulación de un *cloudlet* para cumplir con nuestro tercer objetivo específico En el capítulo IV se muestran los resultados y el análisis realizado que demuestran claramente que es posible ejecutar aplicaciones con gran cantidad de operaciones de manera más eficiente en una nube computacional También se definieron algunos de los requerimientos para lograr este objetivo ya que este nuevo paradigma de desarrollo requiere que las aplicaciones se creen de un modo

distinto al actualmente utilizado

Otra conclusión que se deriva de los datos relacionados al rendimiento del sistema distribuido es que un móvil requerirá entre dos y cuatro unidades de procesamiento para resolver trabajos con pocos datos de entrada el aumentar la cantidad de procesadores más allá de esto puede afectar negativamente la eficiencia del sistema Esto se debe al tiempo adicional para dividir los datos y asignarlos a diferentes unidades de procesamiento que se generan al utilizar más procesadores En los casos donde los datos empiecen a crecer debe obtenerse una ecuación que relacione el crecimiento de los datos en el algoritmo con la cantidad de unidades de procesamiento necesarias para resolver la tarea

Aun cuando el procesamiento distribuido aumenta el rendimiento de un móvil en algunas aplicaciones en otras tareas simples el móvil es relativamente más eficiente Las aplicaciones móviles sencillas no poseen gran cantidad de cálculos y sus operaciones de lectura/escritura se realizan con una pequena cantidad de datos En este caso los servicios del móvil son suficientes para resolver estas tareas sin ayuda externa. En sistemas con más de un nucleo se utiliza esta configuración en su funcionamiento para las tareas básicas de todo móvil que puede realizar un nucleo simple de procesamiento mientras que las aplicaciones complejas como videojuegos utilizan los nucleos avanzados para operar de forma más eficiente El inconveniente con esta configuración es el consumo energético el cual aumenta cuando las aplicaciones poseen operaciones intensivas

Acerca del canal de comunicaciones podemos concluir que es la mayor limitante para la aceleración de un proceso de manera distribuida La transmisión serial de los datos por muy veloz que sea inserta un tiempo de retraso del inicio de las operaciones Esto aun cuando la partición de los datos sea suficiente para empezar el procesamiento siempre habrá que esperar a que todos los datos arriben al servidor para completar

la ejecución luego habrá que esperar que la respuesta sea enviada de vuelta al móvil para que sea mostrada al usuario En este caso se pueden implementar medidas que reduzcan el tiempo de transmisión de los datos o que los datos se transmitan a medida que se crean en el móvil esto puede reducir el tiempo de sincronización de forma tal que la ejecución remota pueda resolverse en menor tiempo Servicios de almacenamiento distribuido como NFS podrían aumentar la eficiencia de ejecución aunque aumenta el consumo de energía, ya que estos generan información redundante en el móvil y en el servidor Esto elimina la necesidad de sincronizar los datos cuando van a ser utilizados

Además se pudo constatar que algunas capas del modelo de referencia OSI no son necesarias en la conexión directa entre el móvil y el *cloudlet* La utilización de la capa de red y transporte aumenta la cantidad de datos innecesaria para la transmisión de datos Es posible establecer un protocolo más ligero de datos utilizando protocolos parecidos a PPP (Protocolo Punto a Punto) pero que permitan la transmisión confiable de los datos y de forma más segura (encriptación) Incluso los móviles que utilicen este tipo de estructura distribuida podrían requerir una modificación del sistema operativo para soportar las funciones distribuidas aumentando la eficiencia del canal de comunicaciones

Considerando lo anterior podemos afirmar que un computador con multiples nu cleos de procesamiento puede dar servicios a varios dispositivos móviles Una buena práctica es mantener uno o varios procesadores dedicados para las operaciones del sistema operativo y las peticiones de los clientes debido a que esto aumenta la velocidad de respuesta del *cloudlet* y mejora el rendimiento general del sistema. Si las funciones básicas del sistema operativo son restringidas a un solo procesador se pueden mantener los demás procesadores libres para atender a las aplicaciones de los clientes móviles reduciendo la complejidad de la planificación de procesos

Para finalizar este capítulo quiero hacer constar que este trabajo de investigación ha sido presentado en XXVI Congreso Científico Nacional bajo el título **Plataforma de Computación Distribuida mediante Nube Computacional para la Asistencia en la Ejecución de Aplicaciones en Dispositivos Móviles conectados a Redes de Área Amplia Inalámbricas** y en el SIECOM 2013 en la Facultad de Informática Electrónica y Comunicación con el título de este trabajo de tesis

Capítulo VI

Trabajos futuros

Durante este proyecto de investigación se probó que es posible crear un sistema distribuido que aumente el rendimiento de aplicaciones móviles Una de las interrogantes formuladas durante este proyecto fue la capacidad de los *cloudlets* de dar servicio a multiples usuarios Esta capacidad estará ligada a la cantidad de nucleos de procesamiento que estén disponibles y a la necesidad de cada usuario Encontrar esta relación para un *cloudlet* como unidad básica de la nube computacional distribuida es un a segura línea de investigación que surge a partir de este proyecto

Luego si establecemos que cada *cloudlet* es una entidad autónoma pero también es parte de una nube computacional más grande entonces tendríamos que encontrar la relación de la asignación de recursos utilizando a los otros miembros de la nube En este caso habría que definir como converge la red de *cloudlets* en cuanto a la actualización de los recursos disponibles en especial los nucleos de procesamiento

Estudiar el consumo energético debido a la utilización de un esquema de apli caciones móviles distribuidas y de ser posible generar una modificación al sistema operativo móvil para crear un servicio que controle el consumo de energía basado en la distancia al nodo *cloudlet*

La creación de un programador de tareas local que funcione en conjunto con un

protocolo de actualización de recursos disponibles dentro de la nube distribuida, es otra posible línea de investigación y desarrollo que surge a partir de este proyecto Una vez establecida la relación de asignación de los recursos en una nube computacional distribuida quedará crear un programador de tareas que cumpla con esta relación

Uno de los temas que se derivan de este trabajo es la modificación del sistema operativo Linux que da soporte a la plataforma Android Esta modificación debe ser basada en la creación de un protocolo de comunicaciones punto a punto entre el móvil y el *cloudlet* Una línea de investigación que desarrolle un protocolo punto a punto para multiples usuarios liviano y confiable es una meta futura que requerirá una red de nube computacional distribuida

Además se hace necesario un protocolo liviano de actualización de los recursos computacionales disponibles que permita a la red conocer los recursos disponibles en cada nodo puesto que no existe un servicio rector que los asigne de forma centralizada. Al ser cada nodo independiente del resto este asigna los recursos que posee a los usuarios locales que posee y publica los recursos que le quedan disponibles Un nodo *cloudlet* puede convertirse en cliente de otro nodo *cloudlet* de ser necesario para ejecutar trabajos de los usuarios si el numero de peticiones excede su capacidad de respuesta en un tiempo aceptable Mediante la inclusión de un pequeno retardo se pueden utilizar los recursos disponibles en un nodo con pocos usuarios locales

Un protocolo para el paso de una celda a otra controlada por un *cloudlet* diferente es otro aspecto de la nube distribuida que permitiría una operación ubícua de las aplicaciones Mejorando la movilidad y evitando la interrupción del servicio una vez el móvil se encuentre fuera del área de cobertura

Realizar pruebas con multiples nodos *cloudlet* y probar los desarrollos antes mencionados sería una etapa final de desarrollo de esta tecnología que podría extender la vida de los dispositivos móviles actuales por mucho tiempo reduciendo su costo de

producción y aumentando su rendimiento y disponibilidad al publico general

Bibliografía

Ahuja R De A and Gabrani G (2011) SLA Based Scheduler for Cloud for Storage & Computational Services *2011 International Conference on Computational Science and Its Applications* pages 258–262

Baratto R. A Nieh J and Kim L (2004) THINC A Remote Display Architecture for Thin Client Computing *Technical Report*

Barrett E Howley E and Duggan J (2011) A Learning Architecture for Scheduling Workflow Applications in the Cloud *2011 IEEE Ninth European Conference on Web Services* pages 83–90

Bernstein D Vij D and Diamond S (2011) An Intercloud Cloud Computing Economy Technology Governance and Market Blueprints

Borkar S (2007) Thousand core chips a technology perspective *Proceedings of the 44th annual Design Automation* pages 746–749

Court H (2009) VESA Net2Display Remoting Standard *Electronics*

Dorn J Hott R and McDaniel M (2011) Exploring Performance and Power Scaling in Multi Core Processors pages 1–12

Imai T Horio K Ohno T and Matsui K (2010) An Adaptive Desktop Trans-

fer Protocol for Mobile Thin Client *IEEE Globecom 2010 Workshop on Mobile Computing and Emerging Communication Networks* pages 1136–1140

ITU T (1999) Remote device control application protocol

Lai A M and Nieh J (2006) On the performance of wide-area thin-client compu ting *ACM Transactions on Computer Systems* 24(2) 175–209

Lau F Belaramani N and Kwan V (2005) Code-on demand and code adaptation for mobile computing pages 2–21

Li C and Deng Z (2011) Value of Cloud Computing by the View of Information Resources In *2011 International Conference on Network Computing and Information Security* pages 108–112 Ieee

Ocheltree K Millman S Hobbs D Mcdonnell M House A Zone S F Nieh J and Baratto R (2009) 14 2 Net2Display ™ A Proposed VESA Standard for Remoting Displays and I / O Devices over Networks *Configurations*

Richardson T (2009) The RFB Protocol *Network*

Satyanarayanan M and Bahl P (2009) The case for vm based cloudlets in mobile computing *Pervasive Computing*

Simoens P De Turck F Dhoedt B and Demeester P (2011) Remote display solutions for mobile cloud computing *Computer* pages 1–6

Sun X H and Chen Y (2010) Reevaluating Amdahl s law in the multicore era *Journal of Parallel and Distributed Computing* 70(2) 183–188

Zeng W and Lu Y (2011) Multimedia at Work Virtualized Screen A Third Element for Cloud Mobile Convergence pages 4–11

Zhang Q Cheng L and Boutaba R. (2010) Cloud computing state-of the-art and research challenges *Journal of Internet Services and Applications* 1(1) 7–18

Anexos

A Piezas de codigo

I mult c

```
//    lt
//
// C r   ght 201   R f  l A p  ll  ~     p  ll @c t c p    g~
//
// Tl s p  g        f        ftw                    d  t  b  t    t and/      d fy
//  t    d    tl   t  m    f tl  CNU G        l P bl    L      s       p bl  l  l by
// th  F     S ft        F    d t          th       e        2  f th  l
// ( t y        pt    ) a v l t
//
// Tl    p  g   n      d st  b ted      th  l  p  tl      t    ll b      f l
// b t WITHOUT ANY WARRANTY  w th               th  l  ,l  d        ty  f
// MERCHANTABILITY      FITNESS FOR A PARTICULAR PURPOSE    S e tl
// CNU C    e  l P bl    Ll        fo   m     d t  l
//
// Y     sl o ld h                 d  a    py  f tl  GNU G       l F bl      L c
//   l   g wit!  tl s j   g  m    f      t   w  t   t  tl   l        S ft
// F     l tl      l           l F a kl    St     t   F ftl Fl       B  t
// MA 02110-1301  USA

#i cl de <stdio h>
#i cl de <stdlib h>
#include <tim  h>
#i cl de <string h>

int mai (i t a gc  ch r   arg )
{
  FILE  op1   op2   r
  i t s1[2]  s2[2]      j  k
  time_t c   _time
   har cmd[512]  tmpf []  =  preXXXXXX

  if(a gc < 6)
  {
    pri tf( U  g   %s m t   _  k m t   _k j p  t_f l    t  t    d \ L  t    d    w ll b
    [e d-1] \       arg [0])
      t    0
  }

  op1 = f p n(a g [1]     b )
   p2 = f p   (  g [2]     b )

  if((i = mkstemp(tmpfn)) == -1)
      t     -5

  r s = fdope  (i    wb )

  if(op1 == NULL)
  {
    time(&curr_t me)
    sprintf( md     h  \ %s  E r      ad  g  %s\ >> od e      l g     t tok(ctime(&
    c rr_time)  \  )  arg [1])
    system(cmd)
     et     -1
  }

  if(op2 == NULL)
  {
    time(&curr_t m  )
    sprintf(cmd      h  \ %s  Err r     d  g  %s\ >>c dser     l g    st tok(ctime(&
    c    _time)  \  )  arg [2])
```

```
    y t m(cmd)
      t     -2
  }

  if( es == NULL)
    return -3


  fread( 1    i    f(i t)  2  op1)
  fread( 2   s    of(i t)  2  op2)
  fg t (op1)
  fgetc(op2)

  if(s1[1]  =  2[0])
  {
    time(&curr_tim )
    sprintf(cmd      h  \ %   Wro g     s  E        p  t  %s\ >>  d       l g    st tok
    (ctime(&curr_time)  \  )  arg [3])
     yst m(cmd)
     et     -4
  }

  float m1[s1[0]][s1[1]]  m2[ 2[0]][s2[1]]
  for(i = 0  i <  1[0]   ++)
  {
    f  ad(m1[1]  sl eof(float)  s1[1]  op1)
    fget ( p1)
  }
  f r(1 = 0  i <  2[0]  i++)
  {
    f ead(m2[1]  sizeof(float)  s2[1]   p2)
    fg tc(op2)
  }
  fclos (op1)
  fclose(op2)

  flo t  [s2[1]]
  fo (i = ato1(a gv[4])  1 < ato ( rg [5])   ++)
  {
    f  (j = 0  j <  2[1]  j++)
    {
       [j] = 0
      fo (k = 0  k < s1[1]  k++)
         [j] += m1[i][k]   m2[k][j]


    }
    fwrit (       of(float)   2[1]   es)
    fp tc( \      es)
  }
  fclose( es)
   e ame(tmpfn  arg [3])
  tim (&c   _tim )
   p i tf(cmd     ho \ %   S c     p  t  %\ >>  d       l g    st tok(ctime(&
    c   time)  \  )   rgv[3])
  system(cmd)
   et r  0
}
```

II join c

```
// j  n
//
// C p  gl t 2012 R f  l A p  ll  <     p  ll @ t  p   g>
//
// Tl   p  g       f    s ft      y        d  t  b  t    t   d/   n d f
//  t   l   h  t   s of th  GNU C   e  l P bl   L          p  bl  h d  b
// tl  F    S ft     F   d t      th       o     f th  l
// ( t       pt   )     l t
//
// Th   g       l st  l  t d    tl   l p  tl  t  t    ll b     f  l
// b t WITHOUT \NY WARRANTY    th  t      th    pl  d  w     t   f
// MERCHANT\BILITY      FITNESS FOR A PARTICULAR PURPOSE   S   tl
// CNU G     l P bl   L       f         d t  ls
//
// Y    h  ld  ha          d a c p  of  tl  CNU C     l I  bl   I    s
//  l  g w tl  th   p  g      f   t   w  t   t  th  F    S ftw
// F   l t     I      l  I  a  kl   St   t   F ftl  Fl     B  t
// MA 0 110-1301  USA

#incl de <stdio h>
#i  cl d  <tim  h>
#i  l d  < tring h>
#in l d  < tdlib h>

i t mai (i t argc  cha    a g )
{
  cha  l g[500]  tmp[] =  jo t p\XX\XX
  time_t curr_tim
  int  si e[2]  i

  if(a gc < 5)
  {
    printf( U  g   %   t_m t_f l    w    ls p o  _f l  [   ]\    arg [0])
     et    0
  }
  if((l = mkstemp(tmp)) == -1)
     t rn  -5

  FILE  prc   o t = fdop  (i   wb )
  if(o t == NULL)
  {
    time(&curr_time)
     p i tf(l g     h  \ 7   N     p  d         l    l    d    l d  % \  >>
    cod       l g
    strtok(ctime(&curr_time)  \  )  arg [1])
    sy t m(l g)
    r t rn  -1
  }

   i   [0] =  toi(a g [2])
  si e[1] = atoi(a g [3])
  l = fw it (si e  si e f(i t) 2 o t)

  if  (i  != 2)
  {
    time(&curr_time)
     printf(l g     h  \ 7   E  o  d  l ct  a de % \  >>  od er    l g
    st tok(ctime(&cur _time)  \  )  arg [l])
    system(log)
    f l s (o t)
      t     -4
  }
```

```
fp tc( \    o t)
t m (&    _t m )
sp i tf(log     h  \ %s  I ye d  %l   gm  t   \  >>   d       l g
strtok(ctime(&  rr_time)  \ )  argc - 4)
 yst m(log)

i = 4
d {
  tim (&  rr_time)
   p i tf(log     h  \ 9'   Ab   d     l  o %s\  >>   d       l g
   trt k(ctim (&c   _time)  \  )   rg [i])
   yst m(l g)

  sp intf(log   t  t -f %s   a g [i])
  while(system(l g))

  pr  = fop n(a g [i]    b )

   f(prc == NULL)
  {
    t m (&     _tim )
    spr ntf(log      h  \ %s  No se p  de l     l    h    % \   >   d       l g
    st tok(ct m (&c rr_t m )  \  )    g [i])
    system(log)
     et     -5
  }

  fseek(pr   0  SEEK_END)
  l  g s = ftell(p c)
  if(s == -1L)
  {
    time(&curr_time)
    sprintf(log     l  \ 9'   N     p  d  det  m n    l tam    d  % \  >>
    ds       log
    strt k( t m (&c   _time)  \  )   rg [i])
     y tem(l g)
    f l    (p   )
    fclo e(o t)
      t     -2
  }
  rew nd(pr )
  char b ffer[ ]
  size_t r  w
    = f ead(b ffer    i  of( h  )     p c)
  if(   = s)
  {
    t m (&     _t m )
    sprintf(log     ch  \ %s  E  o  d  l  t    d  % \  ,,  od     r l g
     t t k( t m (&    _tim )  \  )    g [i])
     y t m(l g)
    f l  e(p c)
    fclo e(  t)
     et rn  -3
  }
  w = fwr te(b ff r    i  of(cha )  s  out)
  if(w  = s)
  {
    time(&curr_time)
    spr ntf(log      h  \ %s  E ro  de   cr t r     "s \  >> c d  r    l g
     t tok(ctime(&c   _time)  \n )   rg [i])
    system(log)
    fclose(prc)
    f lo   (o t)
     et  n  -4
  }
  fclo   (pr )
```

```
    i++
  }wh i (i < a g )

  fclose(out)
     ame(tmp  arg [1])
  time(&curr_time)
   p i tf(log     h  \ %s  A ch    %     d          o  \  ~  c d       l g
  strtok( time(&c   _time)  \  )    g [1])
  system(log)
  return 0
}
```

III splıt_strıpes c

```
//  ı l t_  t  p
//
// C ı\ gl t  012 R f  l Asp ıll  <    ı  ll @  t ı  p  \ g>
//
// Th s p  g  m    f      ftw  e            d  t ıb t   t   d/   m d fy
//  t   d   th  t      f th  GNU G    l P bl   l l   s   s ı  bl  l d bv
// th  F    S ftw    Γ   d t  n    th      s   2  f tl   l
// ( t  o    pt   )     l t
//
// Th s ı  g       d  t  b t d    tl  h ı  th t  t   ll b    f l
// b t WITHOUT ANY WARRANTY  w th  t      tl e   ı l  l w    t   f
// MERCHANTABILITY     FITNESS FOR A PARTICULAR PURPOSE   S   tl
// CNU G n   l P bl   L c  s  f        d t  ls
//
// Y    h  ld l a          d     p   f th  GNU C     l P bl   l
//  l  g w th  th   p  g  n   f   t   w  t   t   th  F    S ftw
// F   d t    l     51 F   kl   St   t   F fth  l l     Bo t
// MA 02110-1301  US\

#includ  < tdio h>
#ı cl de <stdlib h>
#ı clude <string h>
#i cl de <time h>
#  l d  < y /stat h>

int g tn m(cha   str)
{
  cha   es[3]
  char key[] =  0123456 89
   h     p h
  pch = strpbrk (str   k y)
  strncpy(res   pch   3)
    t     atoi(r  )
}

  t m in( t   g   cha     rg )
{
  if(a gc < 4)
  {
    printf( Us ge  %s     lt   at   l m t   2 p oc sso  \t     rg [0])
      t     -1
  }

  FILE     = f pe  (a g  [2]      )
  if(a == NULL)
    return -2

    t a_filas   a_col mnas
  fr  d(&a_filas   s ze f(  t)  1  a)
```

```
  fr ad(& _col m as   s z   f(i t)   1   a)
  fclose(a)

  FILE   b = fope  (a g  [3]       )
  if(b == NULL)
    ret      -3

  i t  b_filas   b_c l mnas
  fr ad(&b_filas    s    of(   t)   1 b)
  f ead(&b_col m as         f(i t)   1   b)
  fclo   (b)

  if(a_columnas   =  b_filas)
     et      -4

  cha   tmpfn[10] =   req\X\XXXX

  FILE   tmp = fopen(mktemp(tmpfn)    w- )
  i t i   endsection       ti     = (a_filas/atoi(arg  [4]))+1
   ha   p   f  [256]   lasti ne[2048]

  sp i  tf(lasti n      f     = t   k   t      0   /j      %  %l  %l     a g  [1]     _f la
    b_col m as)

  f r(i = 0     <   toi(   g  [4])   i++)
  {
    e dsectio   = ( +1)      t   n
    if(   dsectio   > a_filas)
      endsectio   = a_columnas
     pri tf(p  f       p    %03d%03d%03d t p    g t   m(a g  [2])   g t   m(   g  [3])   l)
    fprintf(tmp     t   ks t -    /l   /m lt  %  %%  %l  %l &\      i+1   arg  [2]   arg  [3]
    p cf     (i    se tion)      ds ctio )
    st cat(lasti n     p   f )
  }

  fp i tf(tmp     %   \      la tli e)
  fprintf(tmp        t  $f     \   )
  fclose(tmp)
  chmod(tmpfn   S_IRUSR | S_IWUSR | S_IXUSR)

  spri tf(p cf      /%      tmpfn)
    t      (system(prcfn) >> 8)

}
```

B Matrices

I AbstractMatrix.java

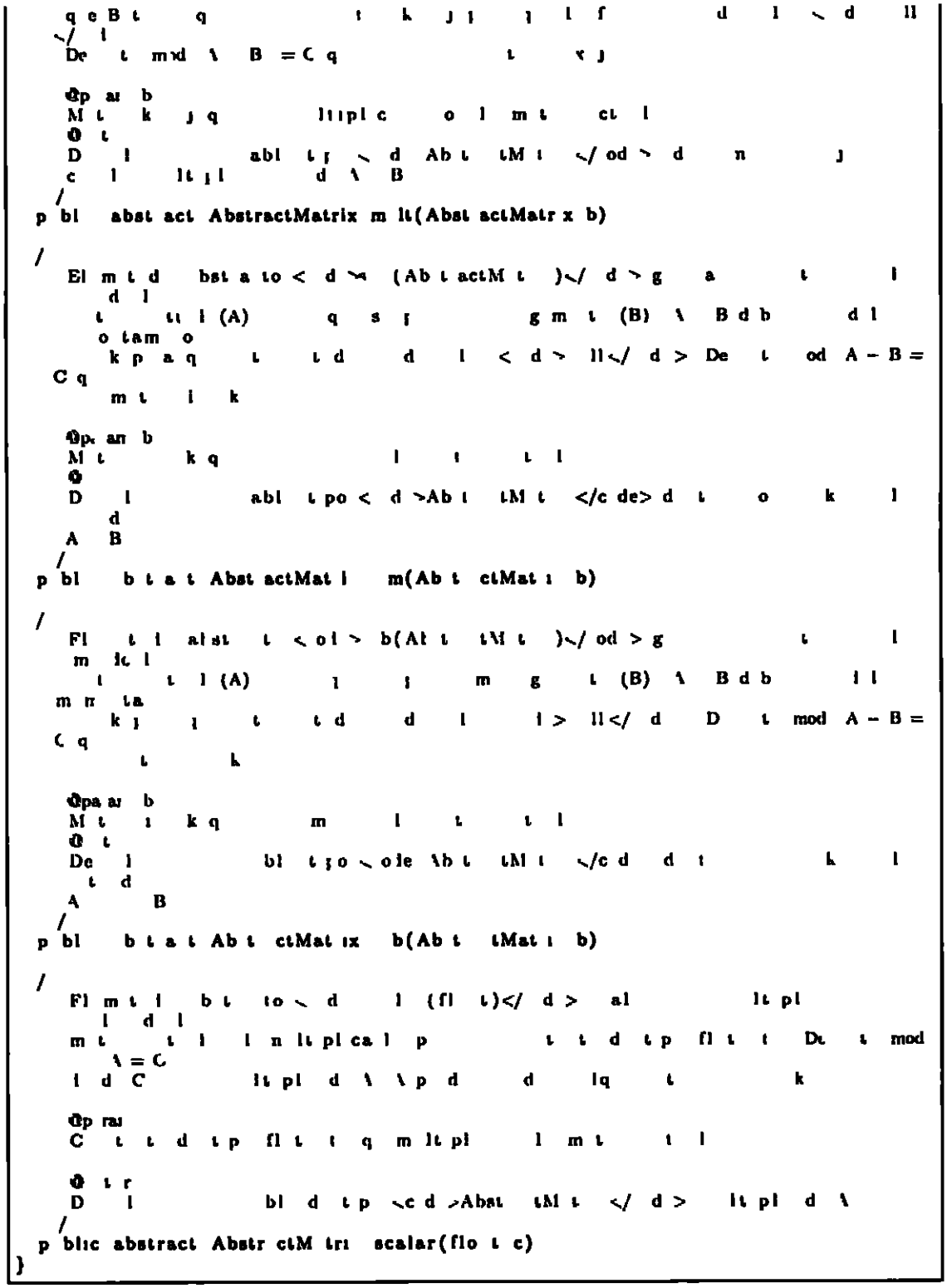
```
package a d.old.matr x

/
   L   l      b t   ct   Ab t   tM t        l   ¡l   t ll   ¡      l       t  c
    t l    l        la   pl           b t     t bl     l    p       t o   b s       q    t d
    l     d      t       d be      t

   @  th    R f   l A p   ll

  /
p bll    b t    t   lass Abst actMatr x {

  /
     At  b to q    co t e     l      b   ¡     l f   t   d l     l     d      t     p
      l obj t      t   l
   /
  protected Str ng fil n  m

  /
     At  b t   ¡ e    e ta l       t d l d      t    c     l   l    l     R m t M t
         t   t        l    pl             t   l
   /
  p i ate static   nt  m t nt = 0

  /
     C   t      l      l    s     m         d   l     l     t s d   l   m t
   /
  p otected  float  element[][]
  /
     C   t          l t          d   l     f l          l      s d   la     t
   /
  p bl     nt  row      l mns

  /
     R p      t             d     l     r   t      d l     t    d    l   l     n t
   /
  prot cted  l  t  mat   m

  /
     C              l   l     l     <   d >Ab t    M t    ~/   l ~

     @pa an     w
     N          l   f l     l   l    m t
     @p ram     l
     ~ ne o de     l         d   l    nat
   /
  p ot   t d  Ab t a tMat    (   t   ow    i t    l mn )
  {
    th s   l m nt =    w  float[  ows][c l mn ]
    this   ows = row
    th s  col  mn   = col  m s
    matnum = matc t
    filen  m   = St ing  fo mat( m t%03d  m t     matn m)
    matcnt++
  }

  /
     C    t    t    j   la   l      ~   l >Abst    M t    ~/   d >

     @p    n
```

```
   E          gl  q       t     l    l m  t   d  l      t
  /
p otect d  Ab tr  tM t i (float[][] e)
{
  this el me t = e
  this  ow  = e le gth
  thi  col m s = e[0] le gth
  matnum = m t nt
  fil  ame = String form t(   t%03d   at   matnum)
  matc t++
}

/
   C    t    t    l  l    la   <  d >Ab t acM t   </  d >
  /
pr t  t d  Ab t actMatri () {
  matnum = m t nt
  filen m   = St i g  fo mat( mat %03d    t   mat  m)
  mat nt++
}


/
   @ t    El     b   d     h
  /
p blic final String g tF l    m () {
    t     filename
}

/  (    -J    d   )
   @s   ja    l  g Obj  t#t St   g()
  /
p bl c St i g toString()
{
  StringBuilder tmp =   w St i gB ilder(   )
  fo (i t i = 0  i < row    ++)
  {
    fo (i t j = 0  j < column   j++)
    {
      tmp  pp nd( l me t[i][j])
      tmp append(    )
    }
    tmp  ppend( \   )
  }
  r t r  tmp toSt i g()
}

/
   El      d  g tC l    (  t) d    l          l      d  l     t

   @p rai    de
   N        d    l         l  t d
   @ t
   D   l           gl  flo t[] q        t        l    l   d  l     l         l c t da
  /
p blic float[] getCol mn( nt  nd x)
{
  float column[] = new float[element[0] length]

  f  (i t i = 0    < element[0] l  gth  i++)
    col m  [i] = element[i][ ndex]

    t    col m
}

/
```

```
    Fl    t do g tEl n  t p    lt       d      l     l m  t   d  l      t    de f
     d   l  l

    @par m
    P  s c      l        t l d l  l     t
    @parām y
    P  s            t   l d l  l m  t
    @ t
    D    l        fl  t        l   l   dcl el     t   l n  t[ ][y]
   /
  p bli  flo t g tEleme t(l t     l t y)
  {
    t     el m  t[ ][y]
  }

  /
    El  m t do getR w(  t) d    l        f l  d  la    t

    @p  am   de
    N m    d  f l     l    d
    @ t
    D    l            gl  fl  t[] q       t      l   l   d  l  f l    l  t d
   /
  p bl c flo t[] g tR w(i t l d   )
  {
    et   n eleme t[i de ]
  }
}
```

II MatrixOps java

```
package a d oid m t i


/
   L    t  f    <  d >M t  Oj </  d    ad          l      b las   d <  d >
    Ab t   tM t   </  d        l
     p  d d d      l        p               l   m t          t  d        ll    F t
      t  f      g  g   la
    p            d  S      R st    M  lt pl    c      T    spo          Fsc l      t  d
      t

   @a tho  R f  l Asp  ll

  /
p blic i t  fac  MatrixOps {

  /
    El    t d  <  l >t    p   ()</  l   g        l     t    t     p   t   d l  l j t
     t  l

    @ t
    D    l e una   riabl  d  tipo <c de>Abst   tM t   </ od > q     s  l   t  np  st
   d l   bj t     t  l
   /
  public abstract AbstractMatrix transpose()

  /
    El    t do <  d   n lt( Ab t   tM t   )</  d > g n          n t         l
      lt pl c c     de l
    matr     t  l (A)        q e  e p s   or o   g me t  (B)  A  s        at        k
   n e t  s
```



III Matrix java

```
package a dro d m tri

impo t ja a io IOExc ptio

/
  L  l  \  i  M t x</ od   s :   l l   l < od >Ab t cM tr </ d >
   impl i  t  l   t f
  < d >M t Op </ d  q  gr g  m t d  d  p      b       i
    m   lt pl
     l       t  t a spo    y  lt j l     l  t    1 b  h
   f       l
  d  t     p  f c   m l  t  d      t  d        t
   d  t d d

  @ th  R f el A p ll

 /
p blic class M tri   t ds Abstr tM t i impl m  t Mat i Op {

  /
    Co  t  t  d  l  l   M t

    @para
    A  gl  fl  t q  c  t    l  l  l t d  l  l  t  d la m t
   /
  p blic Mat i (float[][]  )
  {
   super(e)
  }

  /
    Fl  t d   t t  O(  t  n ) p  t       at    t l   c t  d
    g  l  0 s  E t       t  t   l y
    p  m t  l       p    l  m  pa     t  l l t
   d t   d  p  ot s op      como s m    t

    @p ar  w
    N  i  l f l s d l  m t  O
    @param  l i
    N m   d  c l  as d  l  n t  O
    @ et
    R t       bl tip M t x   l  at  O(  s c l   )
    @thr w  IOE  p
   /
  p blic stat  Mat i O(i t row   t col m s)
  {
   flo t element[][] = new float[rows][col mns]
   fo (i t l = 0  l < rows  l++)
     f  (i t j = 0  j <  l m s  j++)
       leme t[i][j] = (flo t) 0
   Matri   = new M trix( l me t)
     t
  }

  /
    El  tod   i tic U(i t  i t) p r  t       n t  c  t d   c  t  d
    g  l  l s  Est    n   t i  tr  l \
    pern i  l  er   j  o  l     a ja a  n   i  l l t   o
   d t   d  o p   t  s  p  c o      l
      lt j l    d    de  m     t

    @p ar  ows
    N me  d  f las d  l   t  U
```

```
    @p   m    l  n
    N         d    l      s d  l            U
    @  t
    R to  a              bl   t p   M t i        l       t     U(          l       )
    @t    w  IOE    pt
   /
p blic   t tic  Matrix U(i t   ows   i t col mns)
{
  fl   t element[][] =  ew float[rows][column ]
  fo (i t i = 0  i <  ows   i++)
    f r(i t j = 0  j < col mns   j++)
      elem   t[i][j] = (float) 1
  Mat ix x =    w Matrix( l m   t)
   et
}

/
   bl    t d   I(   t) g                 t     id   t d d     d  d     ls      l            m t
  t      l q           t l          d f      t    ope
           t

    @pa am    w
    N         d  f l         l         d  l        t
    @  t
    R t                bl    l  t po M t         todo   l      l m    t               j l       l
  d  g    l  p     p l
   @throws IOE c pt
   /
p bl    t t c  M tr x  I(i t   ows)
{
  float el m   t[][] = new float[ ows][r w ]
  fo (i t i = 0  i < rows   i++)
    fo (i t j = 0  j <  ows   j++)
      if(i == j)
        elem nt[i][j] = (flo t) 1
      else
        element[i][j] = (flo t) 0
  Matrix x =  ew M trix( leme t)
   et     x
}

/  (    -J    l  )
    @       d   d     t     M t i O s#t      p    ()
   /
@O erride
p bli   M t ix t a   p    ()
{
  flo t   [][] = new fl   t[this columns][this rows]
  i t i   j

  fo (j = 0  j < th s columns   j++)
    fo (i = 0  i < thi   ows   i++)
       [j][i] =  l m   t[i][j]

  r t     ew Matrix(e)
}

/  (    -J    d   )
   @s        l   d     t     M t    O s#   lt(    l    d  nat     Abst  ctM t    )
   /
@O err de
public Matrix m lt(Abst actMatrix b)
{

  float  esult[][] =  ew float[this rows][b col m s]
  i t i   j
```

```
  if(this col m s  = b  ows)
    et  n n ll
   lse
  {
    fo (  = 0  i < this row    ++)
      for(j = 0  j < b columns  j++)
      {
        result[i][j] = 0
        for(int k = 0  k < th   col m s  k++)
          esult[ ][j] += this el m  t[i][k]  b g tRow(k)[j]
      }

    et     ew M t i ( es lt)
  }
}

/  (   -J   d  )
   @      d  d   t   M t  Ops#s m(  l  j   t    \b t  tM t  )
  /
@O  rride
p blic Matrix  m(Abst actMatr x b)
{
  flo t  es lt[][] =  w float[th    ws][this col mn ]
   nt i  j

   f((b  ol m s  = th   ol m s) || (b r w  != this row ))
    return   ll
   lse
  {
    f  (i = 0  i < thi  ows  i++)
      f  (j = 0  j < th   col mns  j++)
        result[i][j] = th s  lem  t[i][j] + b g tRow(i)[j]


    et    n w M t i (  s lt)
  }

}

/  (   -J   d c)
   @      d o d   t   M t  Ops#s b(  d  d   t   Ab t  tM t  )
  /
@O errid
p blic Matr x  b(Abst actM tr  b)
{
  flo t  es lt[][] =  w flo t[th   ows][th    l m s]
  i t i  j

  if((b c l m   = th    l m s) || (b  ows  = thi  ows))
    return n ll
   l
  {
    fo (i = 0   < th s  ows   ++)
      fo (j = 0  j < th s c l m s  j++)
        es lt[ ][j] = this el m  t[i][j] - b g tRow(i)[j]
    ret      w Mat i (result)
  }

}

/  ( o -J  ad  )
   @ ee and  l m i   M t  Ops# c l  (fl  t)
  /
@O e  ide
p blic M tr   cala (flo t  )
```

```
  {
    i t i  j
    float x[][] = ew flo t[rows][ l m ]

    fo (ı = 0  ı < row  i++)
      fo (j = 0  j < columns  j++)
        x[i][j] = c  element[ı][j]

      t  ( ew Matrı (x))
  }
}
```

IV RemoteMatrix java

```
package  d oid matrix

import andro d rem t mat i clie t objclient ObjCli t
import android r m t mat i client objclient li nt  t ObjE ent
impo t a d oid emotem trixcli nt bjclie t clie te ent ObjE tListe e
import android til Log

/
  L < d > R  t M i  </ d >       b l   d < d >Ab t  tM t  </ l >
  d   b l          d     nat        j a   l    j
    d  d  j c c    ı t         t q   b  d              t e j
   l l    t   s d  t  l          d t   t    l               d
  t  f    d  bj t    t l       l     lt  o p       p  l  l
  d to  l   ı  q  l  j  t            t   s  l      l  ı
   t  o

  @ th  R l l A j ll
 /

p bli class R moteMat i extend Ab t actM t i impl m t M t i Op {


 /
   Id  t f  l r d  l  t a sacci   m t   t  l  C nb a c d  e q
   g     a    t
  /
 pri ate int tr n actio _ıd

 /
   D t t  d  l             l s  d     t  S  l  p       d
    ı  o  a   d  l  l t s f  t tosa v e  t    p   l
      d   ot   t   < od >s t = ı  </ d    c  lq   ot
   < l    = f l  </ d >
  /
 p ı ate bool a  yn

 /
   D f    l  t        t  f     pl t d     > l   p  d
   rt l z     t a  sa cı   l    d b d  l s po bl   d    d
         q e pod    lt          me te  l   t  do le l  matr
  /
 p ı ate boolean done

 /
   C  st  to  de la  lase <c d >R   t Mat ı </ od >  C  st         at
     p  t  de   l j    t  t j o d  mat   b l  d
   < ode>Ab t  M t  </cod   Objt n  l   l   b s       l a  gl
   d  l   fl t  t s  l ı ıb  d   l   l  c p  d  t  de l
        t   lg  l         t  l   t  d  bj t  d   t
```

```
      a d              d   t f   d   d   bj t     t   p      t
      t

   @p   m
   M t    d   t p      d >\b t   tM t   </  l   q       l            t
   t p  < od >R m t M t   </cod >
  /
p blic Remot M trix(Ab tr  tMat i  m)
{
  thi (m eleme t)
}

/
   C   t   t   l  l   l    < d >R   t M t   </ d >     pt  l      l
   p  d f   dos d  l    l     t   d  l   n t     L     l        t       p  to
   fl t   t   p       ;t   t d  t p      m t

   Fl      t   t   ta  b    g            b  d     l    p   d f  t  p    la
   m t      t  l q      t          f          m       l         d
     t  c        d   l  t   l  n    to

   @param  l     t
   \ l     d  l    l     t s d  l     t
   @th  ws IOE   pt
   E ta    c pc    s  g    a    t\ t   lg   p  bl    co  l s fl  j   d
     cl
  /
p bli   R moteMat i (float[][] element)
{
  s pe (element)
  syn  = false
  done = f lse
  Matri Stub ms =   w M tri St b(this element   th   fil n m )
  t a sactio _id = ObjClient g tInst   c () PUT(ms)

  ObjCli nt g tI   t   c () addListener(new ObjE   ntLi t   e (){

    @O erride
    p blic   oid handleClientE ent(ObjE    t    t) {
      if(   t g tID() == t a saction_ d){
        sync = e t  l S   c s ()
        Log d( R    t M t       Sy ch o   at o     llba k     m and  es lt     + sync)
        do e = t   e
         f(sync)
          ObjClient getInstance() rem   Listen r(this)
      }
    }

  })
}

/
   Co  t   t   d  la  l    < d  R n  t Mat   </  l      t       t   t
   g            t      d l   bj t     ; a t    d l     l     c   o    mb
        t        l  p       t    < od >f l    n  </  d  >  q            t a
    l n     d       l       d       ot   U          e       la  t
        t  t    l       d        t  y  s    t  t   d  s   g    l    l          l
    rch     no      t                 n     je              ;    t   d   f  llo   d
    t     >d           a      bj t   d   t p  <   l >M t   St  b  /  d         l
   d to  d   l   ;  t    q          l    t

   @p  an  f l
   A  h     l   m t    s q    d b       l   l    p r          l   bj t
   <c d >R   ot M tr  </  d >
   @th ow   IOEx  pt
   F t        p         g       s       t    lg    p  bl          l    fl  j    d
```

```
   I
 /
p bli  R moteMatrix(Str ng file ame)
{
  s pe ()
  sync = fal
  do e = fal
  this filename = fil  ame
  tran a tio _id = ObjCl e t getIn t  ce() GET(fil n me)


  ObjClie t g tIn t  c () addL t  e ( ew ObjE  ntListener(){

    @O  rride
    p blic  oid handl Cl  tE ent(ObjE  t evt) {

      if(e t g tID() == tran  t o _id){
        sy c = e t i S cc ss()

         f(sy c){
          MatrixSt b ms = (MatrixSt b) e t getD ta()
          set(ms)
          ObjClient g tI  tance() r m   Liste e (th s)
        }

        done = t
      }
    }
  })
}

/
   C   t  to dumm d  l   l   \ d  R m t M t  </ d > Cr         t
          p   q  p  d     m d f  da p     d  l   n t d s d  l   l
  <c d  R   t M t  </ d >
 /
p bli  RemoteMatr x() {
  s p  ()
}

/
   D   l   l  t t  d  l  t         d   n
   @ et
   D   l     t   l  pe     d              f    t  a  F l  d
    t  n od
 /
p bl c b  l a  i Sy c() {
  return  ync
}

/
   D   l   l  t t  d   l       d  l  t          t  l
   @ t
   D  el     t   l  p       e l  r l l t d  l al  d  t  nod
 /
public b  l   isDone(){
  et r  d n
}

/
   El m t d  p i  l <  d > et(M t  St b)</cod >  l    a l s d t
      l d s del    d    ot p   g g  l         t

   @p ra  ns
   D t  d  la  t z      d f  d
 /
```

```
p   ate    d   t(M tr xSt b ms){

  th    leme t = ms  l me t
  th s r w  =  lement l  gth
  this columns = elem nt[0] le gth
}

/
   El  m t d  ~  d  O(  t    t)</  d > g          n t   d            l
   d          < d ~  s</  d ~ y ~c d   l m  </  d ~

   @p  am   w
   N  e o d  f l   d  la   t   O
   @p  am   l m
   N m    de   l  n    d  la m t   O
   @  t
   R t           t   d        O[  w ][  l    ]
   @tl  w  lOE   pt
   E t      p        g            t  alg   p  bl        l   fl  j   l
     cl
  /
p bli   tat c  Rem t M t t  O( nt r ws  t t col m  )
{
    t
  float el m  t[][] = new float[ ow ][ ol m s]
  fo (t = 0  i <  ows  t++)
    fo ( nt j = 0  j < col mns  j++)
      el m  t[t][j] = (float) 0
  RemoteM t      = n w  R mot Matr x( l me t)
    t
}

/
   El   t d  ~  d  U(  t    t)~/  l  g           t   d    s    l
   d          < d   w ~/  d   y ~c l ~ l m  ~/  d ~

   @p ra    w
   N      d  f l s d  l  m t   U
   @l  ram c l
   N n    d   l      d  l  n t   U
   @  t
   R t         m t   l       U[     ][  l    ]
   @th  w  lOE   pt
   F t      p        g   a       st   lg   p  bl        l s fl  j   d
     l
  /
p bl c stat   R mot Matr x U(i t  ow     t c l mn )
{
  i t
  fl   t   l m nt[][] =  ew fl   t[ ow ][  l m  ]
  fo (t = 0  t < rows   ++)
    fo (i t j = 0  j <   l m s  j++)
      el m  t[i][j] = (fl   t) 1
  RemoteMatrix   =  ew Rem t Matrt ( l m  t)
   et rn x
}

/
   Fl   tod  ~  d ~ st t    l(  t)~/  l ~ g          t    d  t d d
      d  d   L    t    d  t d d   t      p  st  d      s     pt
   d  g   l p    p l

   @param   ws
   N m    d  f l   y   l n   d  l  mat t  l
   @  t
   R t           t    d  t d d l[  w ][  w ]
```

```
   @tl   w   IOE   pt
   E t         p          g                t    lg     p  bl m        l    fl  j    l
      h
  /
public   t  t  c  RemoteM  tri   I(ı  t  row  )
{
   nt  i
  float  elem  nt[][]  =   ew  fl    t[ ows][r  w  ]
  f   (i = 0   i < rows   i++)
    for(int  j = 0   j < row    j++)
       f(i == j)
         l me  t[i][j] = (float)  1
       l
         elem  nt[i][j] = (fl    t)  0
  RemoteMatr x  x =   ew  Remot  M  t ı  (element)
    t      x
}

/
   El  m t  d       d   g  tF l  N  m  ()\/  d  / d       l       l     mb     d        h
   p  d  t          d          l          t  d   l          t              d  l   bj  t

   @  t
   R t          l      l     d  l    t   b  t    f  l    s
  /
p  blic  String  g  tFileName()
{
   et       filen  m
}


/  (     -J      loc)
   @        d    d  m  t     M  t   Ops#m  lt(   d      l       t     Abst    tM  t     )
  /
@O  errid
p  blic  Remot  M  t  i    m  lt(Abstr   tMat  ı   b)
{
  R  mot  Mat  ı    c
  sync = fal
  do  e = fals
  if(b  l  sta  ceof  R  m  t  Mat  ı  )
    c = (Remot  M  tr   )  b
   l
    c = new  R  moteMatrix(b)

  whil  (  thi    l Sy    ()  &&      l Sy    ())

  St  l  g  o  tfn = St  ı  g  form  t(     lt%03d%03d  m  t     this  matnum    c  mat  um)
  St  ı  g  cmd = String  fo mat(  %  %  "   %l     o  tfn    this  file    m        file  am
  ObjCl  e  t  getInstan    ()  get_cpus())
  tr    saction_id = ObjCli    t  getIn  ta     ()  EXE(cmd)

  ObjClie  t  getInst  n    ()  addListen    (  ew  ObjE  entLi  t    e  (){

    @O  e ride
    p  blic   oid  handleClie  tE  ent(ObjE    t     t)  {
      if(e  t  getID() == tra  saction_  d){
        sync = e  t   sS  ccess()

        if(sy  c)
          ObjClient  g  tInstance()  r  mo  eListener(this)

        done = t
      }
    }
```

```
    })

    while('do e)

      t       w R moteMat i (outfn)

  }

  /  (    -J  ad  )
     @       i  d    t    M t   O s#s  (  d  d    t    At t   tM t   )
    /
  @O err d
  p blic RemoteMatrix   m(Abst actMatri  b)
  {
     et       ll
  }

  /  (    -J   do )
     @s     d o d m t    M t   Ops#s b(  d  d    t    \b t  ctM t   )
    /
  @O erride
  p blic Remot Matrix s b(Abst actMatr x b)
  {
     et  n null
  }

  /  (    -Ja  d  )
     @    a d   d  n t    M t   Ops#   l  (fl  t)
    /
  @O    ide
  p bli  R mot Mat i  scalar(flo t  ){
    ret r     ll
  }

  /  (    -J   d  )
     @s     d   i m t    M t   Op<#t    p   ()
    /
  @O erride
  p blic Abst actMatrix tran po  () {
    // TODO \ to-g     t d  eth i  t l
     et       ll
  }

}
```

V FileMatrix java

```
p  k g    d  id m t i

impo t ja a io DataInputStream
 mp rt ja a i  DataO tp tSt eam
import ja a i  F l
impo t ja a io FileInputStream
imp rt j  a io FileNotFoundExcepti n
imp rt j  a io FileOutputStre m
impo t ja a io IOException

/
   l   l       i  F leM tr  </  d >   t   d  d  la  l      b t
   <c de>\bst a tM tr x</  l   y p s   d     : t   t      U   qu  cr
   t  t        p  ti  l     obj t  t p  <  l >M t  \t b</  d    la    l
    l  e      lto :  p    s bclas   d  \cod \Abst  cM tr x\/  d \
```

```
  @  th   R f  l A j  ll

 /
p bl c class FileMatrix  xt  ds Abst actMat ix{

  /
    C   t   t   d  la  l    ~  l  l l M t   </ od    C          t      dc
    l  l             l l     t q   j  m t    l    ope    o    d    t   s
       t s

    @p  am     t
    E   lt     d  los dat       l     t
    @tl ow  IOE   pt
    T            p        h y    p  t l          l   l
   /
  publ   F l Matrix(Mat l St b matrix) th   w  IOE ception{

     pe (mat i  element)

    this file ame = matrix f len m
    F l  matfile =  ew File( /h m /d  k   s /t s s/ +th   fil n m )
    DataO tp tStr  m o t =  ew DataOutp tStr am(  w FileO tputStre m(m tfile))
      t w iteI t(Integer re er  Byt  ( ows))
      t w iteI t(Integer re er  Byt s(col m s))
    out writ Byt ( \  )
    fo (l t l = 0  l <  ow   l++)
    {
      fo (i t j = 0  j < c l mn   j++)
      {
        o t writ I t(I tege   e erseByt  (Flo t floatToRawIntB t ( l m  t[i][j])))
      }
      o t w iteByte( \  )
    }
    o t close()
  }

  /
    C   t   t   d  l   la   ~  d >F l M t   </  d >  Cr          t    i  d
    l  l      p  t  d     a  h

    @p  an  f l
    N mbre d     h  o    t q      t     l    d t
    @thro s IOE   pt  i
    T            p        h      pr bl          l   ch  o
   /
  p bli  Fil M t i (St   g fil   m ) th   w  IOEx  pti  {

    s pe ()

    th s filenam  = f l nam
    File f le =  ew File( /home/dr k     /t    / +this file ame)
     f( f l     i ts()){
      th  w   w Fil NotFo  dE cept on()
    }else{
      DataI p tSt eam data = new DataInp tSt eam( ew FileInp tStr am(file))
      th   r w  = I teg    e e seBytes(data r  dint())
      this column  = Intege   e erseBytes(d t   r  dInt())
      data  kip(1)
       l ment =  ew float[th s row ][th s col m s]
      fo (int i = 0  l < row   i++){
        for( nt j = 0  j < col mns  j++)
          thi   eleme t[i][j] = Float i tBitsToFloat(Integer r   rseBytes(data readInt
  ()))
        dat   k p(1)
      }
      data close()
```

```
    }
  }

  /
     C          lt     d  m t      t l   d         t      d
     <  l >M t  St b</  d >

     @ t
     R t      l      lt     d  l   d t   d  l  m t
   /
  p bll  Mat l St b getStub(){
     et  n new Mat l St b(this element  th   f l  ame)
  }
}
```

C Servicio de ejecucion remota

I Receiver java

```
package obj r    io

 mpo t ja a io EOFExc ptio
 mp rt ja a io IOExc ption
 mp  t ja a io ObjectInp tStr am
import j  a io Optio alD t Ex  pti
impo t j  a io St   mCo ruptedEx  ptio
impo t ja a net Socket
impo t ja a util Arr yLi t
impo t ja a util It r to
 mpo t ja a  til Link dList
import ja     til List

imp rt a d oid  emotem tr x li  t objcl ent CommandWrap

import obj  r    se  erlistener ObjE e t
impo t objser  r        liste er ObjE   tListe er

/
   L   l    <  d >R        </  l      pl     t                   d       i       d
    bjet           l    d      ie>R         / ode     j l m       l      te f
   <  d >R      bl </  d    q   p      t   j    t    l        t              h lo    T    b
   p s          l  t      l   d   d  l       n     l   q    lleg        g                t   d
   t p   <  d  ObjE    tL  t      / d > p      p d            t  l     l     obj t     q
      h       g  t  d   d  l            t      c     l

   @ th    R f  l  \ p  ll
 /
p bli    l  s Recei e  implem   ts R   n ble{

  /
     Fl j   d    bj t s d     t  l
   /
  p l at  ObjectInputStr  m in

  /
     I i ta e l    d   d         dos      b d s  l   ta   l    n   to
   /
  p l   t  Li k dLi t<CommandWrap> i b    d

  /
     L  t  d     bj t    t p      l >Obj    tl  t      /  l      g  t ad
       d >R             </  l >  ( ad         q      o                t       le  n  t  f
     tod s l      n   l      d    st   l  t
   /
  p l at  Li t<ObjE   tListe er> li t

  /
     L t  j   j  d d d f          l                q               t    l j to  t  l           t
       l  a
   /
  p l ate  boole n  connected = true

  /
     (    t  ct       co   de l    l      l   R         /  d 
     @param   l    t
     F    l   bj t    l  t j    <  d  >Sock t /  d    j             l     l
   /
  p blic Recei er(Sock t  clie t){
```

```
  t y {
    In =   w ObjectI putStr  m(clie t g tInp tSt eam())
  } catch (Str  mCo   ptedEx  ptio   e) {
    e p intSta kT ac  ()
    System  exit(-1)
  } catch (IOExcept on   ) {
      pri tStackTrac  ()
    Syst m   xit(-1)
  }
  inbound =    w Li kedL  t<CommandWrap>()
  liste e   = new A  ayList<ObjE    tL ste e  >()
}

/
  Ag g        bj t  d  t p  <  d >ObjF   tI  t    </c l >    l  l  t  d
   t j to   t        b        t f           d  t p  <  d  O| jL    t</  d >

  @p  ar  l
  Ol j t   d   t p   <  d >ObjE    tL  t     </c d  >
 /
p blic  yn h o l ed   o d  addL  t     (ObjEventLi t    e  l){
  liste e   add(l)
}

/
  R          l  bj t   d   t j    <  d >ObjE    tl  t     </c d > d   l   l   t   d
     t f ca          d    e     t     d   t po <cod   ObjF     t  /  d >

  @pa am  l
  Objet   d   t p   <code>ObjE     tL ste    </c d  >
 /
p blic synchr ni  d   oid   emo eLi t     (ObjE entLi t n    l){
  liste  er  remo e(l)
}

/
  El  m t  d    <   d >t  gg    ()</   l        t f c       t  d     l       bj t
    g  t   d          l   l      <c d >l  t       </  d > d   q     h              d
      t
 /
pri   t   sy ch onl ed    o d  t   gge  (){
  ObjE e t e t =    w ObjE e t(th   )
  Iterator<ObjEventListener> i = listener   terator()
  wh le(i  hasN xt()){
    ((ObjE     tList      )     xt()) ha dl Cli   tE     t(   t)
  }
}

/
  D      l      l               de    bj t     q                    t           l   l   t    l      t  l
    t   l     t

  @   t
  N    ei    d     bj t s   l        d s     l   l st    d       t  d
 /
p blic int size(){
  return inbo nd  si e()
}

/
  De     l          obj to de t p    <code CommandWrap</c dc     v l      l    i  a  dc l
  l st      l  ad   d   obj  t      l        t  d

  @ et   n
  El    m ad      l     d  t     q     ll        m     t     p          i  j l
```

```
 /
p blic CommandWrap get(){
  t    l bo  d r m   ()
}

/
  (, b    l   t d  d  ,  d          t d,/  d ,      t        l  l               d
          d  l
 /
publi   old close(){
  co nect d = false
}

/  (   -Ja  d  )
  @s  j    l  g R    bl #   ()
 /
@O    lde
p blic  old run() {
  while(co  ected){
    Obj  t obj =   ll

    t y {
      bj = in readObject()
    } catch (EOFEx  ptio  e){

    } c t h (OptionalD t E ception  ) {
      e printStackT ace()
      break
    } cat h (ClassNotFoundEx  ptio   ) {
        p l tStackTr   ()
      br ak
    } c tch (IOException e) {
      e pr  tStackTrac ()
      b eak
    }

    if(obj    t    eof CommandWrap){
      l bo nd add((CommandWrap)  bj)
      trigge ()
    }
  }
  t y {
    in close()
  } c tch (IOE ception e) {
    e printStackTrace()
    Sy t m    it(-1)
  }
 }

}
```

II Sender java

```
package obj  r e  lo

import ja a io IOException
impo t ja a io ObjectOutputStr am
 mpo t ja a net Socket
import j  a  et UnknownHostExcept o
imp  t ja a util Collection
impo t ja     til Li kedList
 mpo t ja     til Timer
 mp  t ja a ut l Time Task
```

```
mp t a d oid m tematrix l t objcli t CommandWrap

/
  L l d S l </ d g l d bj t F t
  r l d t m; t lm t 10 s
  t m t do t d l bj t q h l d l l d l d
  d tb d </ d l p dol d l l m t

  E t lase xt l de l l e < d >Ti Task</ d q j t
  t o h l t l d t n; d q ad t
    t d d d l g d F l ct l l t p l r j ta
   d 10 ns to s pa t q m s j p p m h t p
     l

  @a th R f l \sp ll

 /
p bli l ss Sender t d TimerTask {

  /
    Fl j d l j t d l d
   /
  p t ObjectO tp tSt eam o t

  /
    L st l d d l bj t t p < d Co andWra / l d l d
   /
  pri ate LinkedList<CommandWrap> o tbound

  /
    T mp l p l d d t
   /
  pr at Time tmr

  /
    C t t d l l < d -> d </ d >
    @p n l e t
    S k t UDP/TCP d la l
   /
  p bl c Send r(Socket cli nt){
    t y {
      o t = ew Obj tO tp tStre m(clie t getO tp tStre m())
    } catch (UnknownHostException e) {
      // TODO A t -g t d t h ll k
        p intSt kT ac ()
    } catch (IOException e) {
      // TODO A t -g t d t h bl k
        p i tStackTra ()
    }
    tmr = ew T m r()
    tm schedul (this 10 10)
    o tb d = ew Li kedList<CommandWrap>()
  }

  /
    Ca ic l l te p z do d c r l e c el
    s d l obj t
   /
  p blic id close(){
    tmr c ncel()
    try {
      o t close()
    } catch (IOException e) {
      // TODO A to-g t d c t l bl ck
```

```
        p i tStackT  c ()
      }
  }

  /
    Ag  g      obj to  l  t p  < od  <Co nmandWrap</  d >   l  l  t     l   l  d
    bj t   d    l d   E t  p         g  g   l  m nand   l f   l d  l  l  t

    @p  am  bj
    Obj t  d   t p   <  d <Co  man dWrap</  l >              d
   /
  p bli    id    t(CommandWrap obj){
    outbound add( bj)
  }

  /
    P  m t   g  g             bj t   d  t p   <  d <Co nmandWrap</  d >   l
    l  t  d   bj t     l     n       Est  p         g  g   l    j  t      l
      m    d     l f   l  l  l  l st

    @pa a    bj
    C  j  t  d   bj t   d  t p   <  d <Co m andWrap</  l >              l
   /
  p blic  oid s t(C ll  ti   <CommandWrap> objs){
    outbound addAll( bj )
  }

  / (   -l   l c)
    @    j    l  g R    bl #    ()
   /
  @O e  ide
  p blic  oid     () {
    while(o tbo  d si e() > 0){
      t y {
        o t writ Obj ct(o tbo  d  emo e())
        out reset()
      } catch (IOException e) {
        System e   p l tln( I/O E     whil     d  g d t  )
      }
    }
  }
}
```

III CommandWrap java

```
package a droid remotematrixcl nt obj li  t

 mp  t ja   io S  i li abl


/
   L  l    <  d <Co  andWrap</ od >           l     n  lt     p  a l s   i dos
   y a g    t   e   do   l s        d  jec ci    en t   E t   l s  p
          l l  t f  do  q   se    e   t       d          st   c   d  l   l
   d  f       t   t c  pa   ga   t     i     d    d   t f   c     e              d
   c  c i    c    l  er  d r

   @  th   R fa l  \ j  lla

  /
publ c cl s  CommandWrap  mpleme ts Ser  l able{
```

```
/

 /
p i  t  static fi al lo g se ialVe sionUID = 2710121279941355322L

/
   C d        l      d    s      l ad   P     l      d     t al las
    ad     de      d
   - ACI
   - PUT
   - CET
   - EXE
   - PRC
   - FIN
  /
p bli  fi   l St i g cmd

/
   D t         d      t        d l l    fl j s d    bj t
  /
p blic fi al Object data

/
   E    l       d  q   od  cnd == ACK </  d >  d f     s   l    p   a
      t      f       t     L  p  p  d  l    l > d</  d > pe    t   l   t
   q              f     ia   l      sp    t      l          d s
  /
p bli  fi al boolea   s  c ss

/
   C  t o de ide t f   d        p rn t              t d    s     cos   p  a cad
   t
  /
pri ate   t tic int id nt = 0

/
   N n     d    l   tif          l  l      t          t  l
  /
p blic final i t id

/
   S           p    t   od >cmd ==  ACK </c d >    t         l      k</  d
   d f        l   t a          f       i
  /
p bl   f n l bo l       k

/
   Co  tr   t r d  la   l        l Co  mandWrap</ od >

   @    n  n l
   Co and            j    t d      el ot    e t       p  d                d
       b
   t mb
   @p  a   d t
   D t                l
  /
p blic CommandWrap(String cmd  Object data){
  this(cmd  d ta  f l    fals   idc t++)
}

/
   C n t  et    d  l   cl s   ode CommandW  p /  l >

   @p ra  cı d
   Comando a s   cj   t  l  cn   l   t     t  m    p  d  s      a     d
       ho
```

```
    t  b
    @p  a  l t
    D t              a l
    @par m  l
    l l  t f  d  d l  t
   /
 p blic CommandWrap(St i g cmd  Obj t dat  l t  d){
   this(cmd  data  fals   false  d)
 }

 /
    C   t  t  d  l  l   <  d >CommandWrap</  d

    @p  am  md
    C   d         j  tad      l  t    t     p  d            d
       b
    t  b
    @p  a  l t
    D t   s       d
    @ ar  n  k
    D f       l     aj  s        d   e  l
    @p  an  s
    D fi    l   t t   d  l  t
    @p  a   d
    ld  t f   l   d  l  t a
   /
 p blic CommandWrap(String cmd  Object d t  boolean  k  boolean s   s  i t ld){
   th   cmd = cmd
   this data = data
   this ack =  k
   this success =    cess
   thi  ld = ld
 }

}
```

IV MatrixStub java

```
/

  /
package android m t

 mp  t ja a lo Serializ ble

/
   L  l s  d  M t  St b</  d      l  l  l        t      l   g
      t      d  l     b l    d <  l  Abst  tM t  </  d   T t  l
      olt         l   p   t   f   l   f        d  l     t    l  l
   la  pl  a        l

   @  th   R fael A pr ll
  /

public cla s Mat ixStub  mpl me ts Serialisabl  {

  /

   /
  pri at  static fi al l ng se ialVer i nUID = -6794951589754725268L
```

```
/
   F   l    b   d l    h    q      t    l    t
  /
p bl   fi al St i g fil n m


/
   Es el c  t   d   l  l    l    t   d  l    t
  /
p blic fi al flo t[][]  l me t

/
   C   t   t          l  l  < ode> M t  St b\/ ode   k  t  q   l   d t
   p  d         d f  ad   p    t os  l    t   d l ; g      g    t    d  l
     t g  d d d  l   d t       d

   Q arum  l n  t
   El     t   d  l  m t
   Qpa am f l
   N  bre d l    h
  /

p bli  Mat i Stub(fl  t[][] element  St i g file am ){
  this filename = fil n m
  thi   l me t = elem nt
 }
}
```

D Datos experimentales

I Tiempo de ejecución

Aquí se muestran los datos promedios mínimos y máximos obtenidos durante el experimento Generalmente se muestran los datos para las diferentes configuraciones que se utilizaron varios procesadores remotos y un procesador local

Tabla D 1 Comparación d l ti mpo de ejec ción local y remotos

	Tiempo de ejec ció					
# de ps	Máx 11prc remotos [seg]	Máx 9prc rem tos [seg]	Máx 7prc remotos [seg]	Máx 5prc remotos [seg]	Máx 3prc remotos [seg]	Mí 1prc local [seg]
8400	1 615	0.361	0 402	0 488	0 466	0 016
65600	1 884	0 482	0 706	0 828	0 473	0 049
219600	2 632	1 074	1 172	2 278	1 071	0 162
518400	2 933	1 335	1 362	1 512	1 275	0 380
1010000	2 097	1 836	2 239	1 874	1 738	0 761
1742400	2 356	2 587	2 117	1 857	2 261	1 354
2763600	2 459	2 907	2 729	2 779	2 590	2 231
4121600	3 080	3 090	2 948	2 837	3 084	3 420
5864400	4 013	3 407	3 503	4 918	3 407	4 735
8040000	5 230	4 293	4 186	4 390	4 220	6 504
10696400	6 166	5 411	4 617	4 899	4 961	8 865
13881600	7 830	5 771	5 910	5 695	5 731	11 203
17643600	7 428	6 826	6 417	6 398	7 123	15 158
22030400	8 607	7.384	7 451	7 645	7 058	18 344
27090000	10 969	8 877	8 765	8 678	8 402	22 956
32870400	9 561	9 463	9 244	9 445	9 296	27 607
39419600	10 736	10 669	10 423	10 382	10 614	33 900
46785600	12 972	12 259	12 074	12 216	11 901	39 787
55016400	13 576	13 177	12 946	12 627	13 507	47 245
64160000	14 469	17 326	13 651	13 893	14 450	55 393
74264400	15 810	15 751	15 306	15 399	15 751	64 952
85377600	17 012	16 644	17 766	16 345	16 953	72 466
97547600	18 517	19 127	21 830	18 352	18 108	84 778
110822400	20 725	19 470	20 593	20 124	20 177	98 850
125250000	22 294	21 810	20 768	21 607	21 946	109 698
140878400	23 194	23 184	22 653	22 947	22 771	124 863
157755600	25 199	24 428	24 954	25 191	24 991	141 576
175929600	26 368	26 831	26 136	25 482	26 722	159 095
195448400	27 724	27 900	28 115	28 155	29 335	176 129
216360000	29 586	30 681	30 792	34 375	31 576	193 665
238712400	33 284	32 352	31 467	32 140	33 065	218 832
262553600	34 860	34 729	34 788	33 677	35 959	242 988
287931600	36 522	36 120	43 413	36 567	36 672	264 248
314894400	39 747	37 950	49 130	38 284	40 019	288 520
343490000	40 456	39 051	42 005	40 910	41 819	312 244
373766400	42 293	42 207	42 568	42 771	42 916	353 175
405771600	44 748	44 239	46 197	44 290	47 089	383 482
439553600	47 688	47 034	47 428	47 873	47 231	408 873
475160400	51 252	49 811	50 218	50 593	50 906	452 906
512640000	51 873	51 433	51 916	53 223	54 269	486.355

Tabla D 2 Tiempo d ejec ión usando 11 procesadores remotos

	Tiempo d	ecución mínimo (seg)				Ti mpo de ejecución promedi (seg)					Tiempo de ejecució máximo (seg)				
# d ps	p1	p2	mul	res p	total	p1	p2	mul	res p	total	p1	p2	mul	res p	total
8400	0 069	0 07	0 109	0 001	0 328	0 345	0 267	0 05	0 125	0 708	0 789	0 396	0 476	0 235	1 615
65600	0 132	0 215	0 123	0 118	0 483	0 145	0 397	0 177	0 256	0 834	0 206	0 723	0 385	0 777	1 884
219600	0 249	0 361	0 114	0 175	0 958	0 784	1 081	0 137	0 668	1 869	785	1 928	0 194	2 150	2 632
518400	0 277	0 6 1	0 087	0 270	1 241	0 547	1 047	0 219	0 460	1 728	0 990	1 987	0 315	0 857	2 933
1010000	0 384	0 681	0 076	0 338	1 146	0 532	0 972	0 184	0 373	1 836	0 786	1 607	0 335	0 399	2 087
1742400	0 670	0 977	0 076	0 490	1 746	0 981	1 299	0 140	0 723	2 171	1 149	1 732	0 322	1 126	2 856
2765800	0 637	1 111	0 079	0 569	1 990	0 810	1 305	0 084	0 864	2 262	1 068	1 791	0 091	1 228	2 459
4121800	0 899	1 446	0 080	1 322	2 886	1 028	1 838	0 092	1 346	2 981	1 196	1 613	0 099	1 373	3 080
5864400	1 281	1 896	0 080	0 854	2 914	1 496	2 316	0 191	0 921	3 432	1 705	2 795	0 386	0 987	4 013
8040000	1 333	2 236	0 084	1 052	3 946	1 914	2 912	0 093	1 448	4 509	2 436	3 414	0 100	1 711	5 230
10686400	1 711	2 737	0 080	1 287	4 256	2 132	3 341	0 101	1 553	5 042	3 538	4 702	0 115	1 906	6 106
13881600	1 956	3 316	0 117	2 113	5 928	2 373	3 981	0 192	2 351	6 841	2 897	4 982	0 369	2 645	7 830
17643600	2 241	3 876	0 117	1 722	6 203	2 655	4 475	0 178	2 160	6 870	3 011	5 504	0 364	2 676	7 425
22030400	2 710	4 413	0 120	2 585	7 708	3 030	4 995	0 182	2 728	7 962	3 119	5 819	0 383	2 892	8 607
27080000	2 945	5 007	0 125	2 821	8 026	3 689	5 919	0 169	3 074	9 237	4 169	6 779	0 272	3 830	10 969
32870400	3 469	5 617	0 131	3 046	9 081	3 912	6 882	0 156	3 182	9 289	4 338	6 087	0 191	3 254	9 561
39419600	3 898	6 353	0 140	3 390	10 121	4 037	6 467	0 155	3 681	10 371	4 286	6 535	0 171	3 972	10 736
46785600	4 348	7 149	0 181	3 835	11 479	4 929	7 844	0 197	4 184	12 006	6 164	7 976	0 206	5 080	12 972
55016400	4 981	7 929	0 171	4 137	12 456	5 181	8 482	0 191	4 384	13 129	5 484	9 106	0 222	4 812	13 576
64160000	5 669	8 997	0 197	4 426	13 744	6 104	9 450	0 217	4 48	14 259	6 360	9 615	0 231	4 837	14 486
74264400	5 970	9 681	0 229	4 761	14 835	6 198	9 827	0 244	5 118	5 293	6 561	9 929	0 287	5 583	15 810
85377600	6 498	0 584	0 219	5 210	16 133	6 803	10 743	0 284	5 525	16 678	6 763	11 209	0 307	5 806	17 012
97547600	7 017	11 509	0 2 1	5 6 4	17 633	7 356	1 907	0 288	5 795	18 135	7 782	12 337	0 312	6 128	18 517
110822400	7 914	12 775	0 261	6 021	19 374	8 101	13 277	0 290	6 138	19 853	8 636	4 233	0 301	6 388	20 725
125250000	8 375	3 6 7	0 319	6 456	20 686	8 615	14 648	0 326	6 540	2 716	9 445	15 177	0 331	6 630	22 294
140878400	9 789	15 431	0 340	6 980	23 030	9 861	15 520	0 346	7 050	23 116	9 919	15 587	0 351	7 095	23 194
157755600	9 293	15 405	0 359	7 374	23 447	9 910	16 485	0 370	7 478	24 482	10 548	7 34	0 390	7 870	25 190
176928400	10 74	6 783	0 33	7 943	25 383	10 522	17 033	0 382	8 202	25 801	1 063	7 681	0 410	8 736	26 368
198448400	9 990	17 846	0 441	8 383	26 938	11 116	18 058	0 464	8 622	27 393	2 21	18 411	0 490	9 143	27 724
218360000	11 454	18 984	0 487	8 9 5	28 858	11 884	19 533	0 493	9 069	29 394	12 181	19 828	0 500	9 121	29 586
238712400	12 280	20 496	0 518	9 715	31 406	12 817	20 856	0 560	10 381	31 974	12 890	21 231	0 630	11 380	33 284
262553600	13 347	2 913	0 629	0 128	33 011	13 663	22 616	0 668	10 496	34 034	14 79	23 982	0 698	11 124	34 860
287931600	3 826	22 763	0 609	10 760	34 497	14 184	23 602	0 652	11 032	35 486	4 400	24 654	0 704	11 401	36 522
3 4864400	4 3	24 567	0 683	11 422	37 408	14 928	25 059	0 71	12 242	38 193	5 603	25 648	0 759	13 214	39 747
343490000	14 507	24 763	0 709	11 850	38 339	15 425	26 202	0 746	12 267	39 803	16 030	27 360	0 771	12 642	40 486
373766400	15 514	25 926	0 732	12 599	40 201	16 661	27 529	0 785	12 792	41 330	17 149	28 593	0 791	13 343	42 263
4087 1600	16 041	28 67	0 860	13 297	42 835	17 245	28 791	0 930	13 689	43 647	18 186	29 435	1 141	13 966	44 748
439583600	17 061	30 073	0 932	14 005	45 439	17 705	30 442	0 958	14 585	46 260	8 66	30 978	0 971	15 348	47 688
475160400	18 815	32 176	1 021	14 763	48 316	19 808	33 856	1 047	14 806	49 710	20 669	35 159	1 076	14 922	51 282
512640000	18 309	32 749	1 087	15 488	49 596	19 436	33 913	1 081	15 739	51 063	20 499	34 385	1 109	16 049	51 879

Tabla D 3 Tiempo de ejec ció usando 9 procesadores remotos

	Tiempo de ejecución mínim (sec)					Tiempo de jecución promedi (sec)					Tiempo d ejecución máxim (sec)				
# de ps	p	p2	mul	res p	tal	p1	p2	mul	res p	total	p1	p2	mul	res op	total
8400	0 047	0 098	0 078	0 076	0 271	0 092	0 123	0 108	0 082	0 323	0 135	0 173	0 125	0 088	0 361
65600	0 10	0 164	0 107	0 111	0 397	0 130	0 190	0 121	0 119	0 430	0 165	0 213	0 127	0 146	0 482
219600	0 195	0 283	0 132	0 177	0 632	0 355	0 437	0 141	0 193	0 775	0 701	0 747	0 159	0 232	1 074
518400	0 362	0 473	0 087	0 252	0 841	0 461	0 624	0 123	0 320	1 069	0 720	0 866	0 184	0 490	1 335
1010000	0 472	0 668	0 071	0 352	1 176	0 494	0 708	0 141	0 727	1 579	0 833	0 772	0 320	0 977	1 838
1742400	0 679	0 821	0 092	0 454	1 858	0 820	1 046	0 167	0 692	1 930	1 087	1 293	0 342	1 354	2 587
2763600	0 754	1 141	0 079	0 597	1 891	0 823	1 206	0 090	1 069	2 388	0 925	1 311	0 104	1 484	2 907
4121600	0 908	1 500	0 081	0 702	2 819	0 970	1 807	0 144	1 006	2 967	1 006	2 221	0 308	1 349	3 090
5864400	1 161	1 850	0 089	0 881	2 837	1 443	2 159	0 099	0 883	3 164	1 600	2 359	0 108	0 921	3 407
8040000	1 823	2 142	0 090	0 969	3 466	1 635	2 437	0 099	1 233	3 827	1 931	2 958	0 115	1 726	4 293
10696400	1 657	2 592	0 099	1 236	4 103	1 841	2 885	0 159	1 712	4 828	2 208	3 840	0 329	2 179	5 411
13881600	1 807	2 986	0 113	1 484	4 737	2 049	3 176	0 124	1 968	5 342	2 174	3 350	0 150	2 113	5 771
17643600	2 392	3 612	0 110	1 657	5 468	2 680	3 780	0 103	1 890	5 941	3 438	3 981	0 462	2 381	6 826
22030400	2 373	4 308	0 126	2 040	7 153	2 719	4 497	0 153	2 536	7 254	2 921	4 883	0 179	2 701	7 384
27090000	2 976	4 843	0 131	2 779	8 065	3 404	5 326	0 150	2 873	8 474	3 770	5 708	0 169	2 961	8 877
32870400	3 546	5 666	0 142	2 886	9 057	3 695	5 840	0 215	3 099	9 258	3 948	6 018	0 441	3 213	9 463
39419600	3 970	6 335	0 155	3 360	10 020	4 101	6 444	0 168	3 529	10 250	4 291	6 877	0 188	3 974	10 669
46785600	4 027	7 167	0 190	3 732	11 367	4 407	7 521	0 218	3 887	11 719	4 741	8 131	0 240	4 225	12 259
55016400	4 662	7 749	0 182	4 072	12 302	4 938	8 045	0 212	4 348	12 713	5 545	8 814	0 233	4 748	13 177
64160000	5 79	8 441	0 200	4 453	13 527	6 248	9 465	0 287	4 606	14 541	9 149	12 179	0 511	5 021	17 326
74264400	5 720	9 521	0 261	4 719	14 748	5 896	9 799	0 29	06	5 305	6 232	10 034	0 339	5 498	15 751
85377600	6 91	10 214	0 244	5 204	15 800	6 340	10 369	0 268	5 493	16 2 6	6 599	10 576	0 300	5 926	16 644
97547600	6 290	10 980	0 263	5 617	17 010	7 055	11 645	0 336	5 750	17 915	8 898	12 980	0 368	6 274	19 127
1 0822400	7 264	12 240	0 332	6 081	18 983	7 546	12 401	0 339	6 352	19 233	7 733	12 601	0 342	6 685	19 470
125250000	7 442	12 824	0 348	6 461	19 838	8 066	13 510	0 383	6.554	20 612	8 440	14 644	0 416	6 683	21 810
40878400	7 713	13 443	0 398	6 881	21 548	8 123	14 770	0 427	7 13	22 475	8 661	15 673	0 516	7 589	23 184
5775 600	9 219	15 247	0 420	7 349	23 426	9 381	15 761	0 431	7 597	23 978	9 688	16 222	0 442	8 024	24 428
1 5929600	9 370	16 039	0 080	0 619	17 931	9 976	16 823	0 165	6 683	23 844	10 445	17 881	0 470	8 766	26 831
195448400	9 6 6	16 796	0 086	0 164	18 859	10 677	17 82	0 301	7 150	25 450	11 424	18 411	0 634	9 217	27 900
216360000	11 005	19 277	0 080	0 740	21 525	11 583	20 105	0 274	7.522	28 077	12 043	20 904	0 882	9 607	30 681
2387 2400	2 470	20 856	0 083	0 158	21 633	12 776	21 132	0 301	7 898	29 545	3 155	21 752	0 623	10 016	32 352
262553600	2 903	21 422	0 079	0 794	22 538	13 415	22 274	0 227	8 612	31 328	14 498	22 901	0 783	10 962	34 729
287931600	13 010	22 050	0 087	0 154	22 894	13 977	23 278	0 497	8 913	32 880	14 818	24 394	0 789	11 483	36 120
314894400	13 511	24 402	0 089	0 890	25 733	14 481	24 911	0 244	9 453	34 848	14 940	25 618	0 770	12 159	37 950
343490000	13 995	24 118	0 087	0 246	25 809	14 306	24 963	0 825	9 935	35 687	4 570	25 785	1 414	12 765	39 081
373766400	16 017	27 162	0 078	1 018	29 172	16 492	27 858	0 264	10 546	38 946	7 021	28 677	0 945	13 320	42 207
405771600	16 761	29 110	0 079	0 139	30 513	17 470	29 617	0 466	10 812	41 205	18 486	29 944	1 085	13 794	44 239
439553600	16 470	29 275	0 084	1 131	31 599	17 079	30 005	0 3 5	12 136	42 677	17 547	31 870	1 170	15 011	47 034
475 60400	17 232	30 276	0 088	0 500	31 378	18 038	31 589	0 683	2 370	4 878	18 607	33 493	1 705	15 871	49 811
512640000	19 653	4	74	268	34 160	1 3	33 7 7	330	2 945	47 300	20 036	35 229	1 307	16 301	51 433

Tabla D 4 Tiempo de ejec ció usando 7 procesadores remotos

	Ti mpo d		ecución mínim	(seg)		Tiemp d		ecución promedi	(seg)		Tiempo d	ejecución máxim	(seg)		
# d ps	p1	p2	mul	res p	tal	p1	p2	mul	res p	tal	p1	p2	mul	res p	total
8400	0 057	0 060	0 22	0 082	0 298	0 082	0 103	0 134	0 060	0 332	0 134	0 182	0 140	0 099	0 402
65600	0 060	0 158	0 09	0 121	0 418	0 110	0 244	0 120	0 79	0 845	0 127	0 440	0 138	0 389	0 706
319600	0 193	0 3 6	0 118	0 163	0 674	0 409	0 830	0 186	0 168	0 907	0 684	0 791	0 376	0 207	1 172
515400	0 306	0 449	0 101	0 255	0 860	0 422	0 610	0 66	0 203	1 083	0 701	0 950	0 331	0 370	1 362
1010000	0 401	0 656	0 086	0 340	1 381	0 778	0 936	0 190	0 535	1 686	0 842	1 128	0 392	1 003	2 239
1742400	0 928	0 908	0 124	0 116	1 166	0 808	1 070	0 138	0 441	1 886	0 980	1 244	0 170	0 747	2 117
2763600	0 740	1 131	0 66	0 564	2 105	0 951	1 309	0 193	0 836	2 349	1 525	1 775	0 350	1 227	2 729
4121600	1 043	1 386	0 087	0 167	1 842	1 262	1 825	0 115	0 620	2 866	1 828	2 135	0 148	0 743	2 948
5864400	1 000	1 818	0 081	0 175	2 223	1 351	2 026	0 166	0 824	3 029	1 590	2 250	0 414	1 267	3 503
8040000	1 340	2 127	0 081	0 166	2 636	1 439	2 350	0 151	1 046	3 580	1 780	2 599	0 320	1 642	4 186
10656400	1 842	2 601	0 089	0 176	3 098	1 883	2 848	0 105	1 285	4 274	2 100	3 189	0 156	1 853	4 617
13881600	1 980	3 222	0 075	0 211	3 643	2 379	3 278	0 151	1 647	5 157	2 850	3 334	0 346	2 160	5 910
17643600	2 285	3 687	0 085	0 452	4 822	2 585	3 835	0 161	1 662	5 735	2 837	3 980	0 355	2 307	6 417
22030400	2 601	4 287	0 080	0 343	4 848	2 747	4 305	0 143	2 177	6 822	3 081	4 830	0 316	2 789	7 451
27090000	2 779	4 714	0 094	0 165	5 089	3 375	5 187	0 133	2 319	7 752	3 809	5 647	0 187	2 864	8 765
32870400	3 085	5 437	0 086	0 309	5 973	3 385	5 586	0 162	2 593	8 434	3 671	5 903	0 343	3 208	9 244
39419600	3 648	6 006	0 083	0 186	6 648	3 685	6 212	0 185	2 859	9 372	3 870	6 382	0 288	3 856	10 423
46783600	4 198	7 085	0 085	0 327	8 100	4 560	7 427	0 126	3 208	10 856	5 028	7 879	0 288	4 440	12 074
55018400	4 472	7 649	0 113	4 048	12 113	4 718	7 585	0 221	4 144	12 369	4 966	8 834	0 261	4 334	12 948
64160000	4 880	8 359	0 089	0 423	9 606	5 475	8 595	0 20	3 707	12 635	6 082	8 929	0 380	4 830	13 661
74264400	5 678	9 401	0 072	4 720	14 404	5 879	9 688	0 227	4 984	15 030	6 230	10 046	0 360	5 839	15 308
85377600	6 470	9 884	0 0 0	0 762	14 458	6 107	10 762	0 177	5 075	16 153	6 883	13 264	0 201	7 884	17 766
97547600	6 325	10 924	0 178	0 280	12 174	7 169	12 266	0 625	4 908	7 961	8 916	14 187	1 620	6 283	21 850
110822400	6 820	11 785	0 083	0 466	13 423	7 403	12 764	0 256	5 159	18 331	8 117	13 487	0 601	6 889	20 593
125250000	7 328	13 036	0 321	0 297	16 289	7 585	13 6 0	0 378	5 408	19 659	8 816	14 850	1 019	6 589	20 786
140878400	7 880	13 824	0 094	0 856	16 328	8 105	14 248	0 243	5 766	20 452	8 386	14 850	0 703	7 87	22 653
157785600	9 083	15 747	0 092	0 211	16 178	9 300	16 218	0 304	6 203	22 900	9 785	17 206	0 690	8 202	24 954
176029600	9 290	15 923	0 085	0 684	19 011	10 082	17 160	0 196	6 782	24 256	10 888	18 079	0 884	8 656	26 136
195448400	10 494	17 868	0 089	0 176	18 763	11 139	18 404	0 314	6 9 1	25 875	11 999	18 700	0 680	9 089	28 118
216360000	10 711	18 349	0 073	0 785	19 844	11 325	19 397	0 214	7 631	27 416	11 950	20 681	0 683	9 736	30 792
238712400	11 740	19 629	0 086	0 186	21 504	12 076	20 484	0 492	7 606	28 044	12 295	20 984	1 083	9 689	31 487
262583600	12 268	2 088	0 0 5	10 98	31 648	12 707	21 868	0 424	10 362	32 868	13 365	22 649	0 924	10 985	34 788
287931600	13 9 9	23 922	0 090	10 630	35 115	14 808	26 229	0 684	11 016	38 165	15 770	30 206	1 199	11 760	43 413
314864400	14 305	24 456	0 075	0 985	27 084	16 015	28 931	0 280	9 723	39 113	20 186	36 776	0 904	12 257	49 130
343490000	13 957	24 952	0 089	0 443	27 550	15 300	28 687	0 482	9 979	37 400	16 849	28 608	1 149	12 796	42 005
373766400	15 875	27 082	0 089	1 055	28 983	16 488	27 806	0 287	10 854	39 014	16 914	28 503	1 048	13 496	42 868
408771600	17 117	29 052	0 093	0 93	30 983	17 899	30 228	0 556	11 033	42 086	19 056	31 331	1 243	14 238	46 197
439653600	16 728	28 829	0 073	1 444	33 468	18 088	31 240	0 883	11 8 1	43 924	19 236	32 863	1 341	14 981	47 428
475180400	18 470	32 190	0 093	0 196	33 478	19 166	32 924	0 634	12 361	46 249	20 167	33 854	1 442	16 244	50 218
512640000	18 257	32 683	0 085	1 309	36 142	19 496	33 854	0 722	12 997	47 883	20 875	35 081	1 835	16 431	51 916

Tabla D 5 Tiempo de ejec c ón usando 5 procesadores remotos

	Tiemp d ejecución mínim (sec)					Ti mp d ejecución promedi (sec)					Tiempo d ejecución máxime (sec)				
# d ps	p1	p2	mul	res p	tal	p1	p2	mul	res p	tal	p1	p2	mul	res p	total
8400	0 046	0 085	0 101	0 080	0 304	0 101	0 151	0 128	0 099	0 378	0 153	0 248	0 156	0 152	0 485
65600	0 103	0 182	0 109	0 102	0 411	0 183	0 251	0 146	0 180	0 838	0 360	0 3 5	0 281	0 343	0 828
219600	0 191	0 284	0 084	0 165	0 620	0 328	0 433	0 107	0 827	1 072	0 643	0 722	0 133	1 680	2 278
518400	0 317	0 488	0 097	0 232	0 8 0	0 500	0 671	0 161	0 266	1 103	0 787	1 131	0 354	0 336	1 512
1010000	0 441	0 661	0 089	0 353	1 093	0 661	0 854	0 112	0 450	1 402	1 023	1 060	0 171	0 642	1 874
1742400	0 507	0 887	0 070	0 119	1 459	0 775	1 130	0 095	0 396	1 623	1 131	1 377	0 114	0 497	1 887
2763600	0 802	1 106	0 082	0 871	1 980	0 954	1 284	0 154	0 969	2 431	1 192	1 881	0 423	1 233	2 770
4121600	0 997	1 533	0 070	0 676	2 378	1 328	1 806	0 088	0 785	2 662	1 806	2 039	0 081	0 946	2 837
5864400	1 206	1 884	0 070	0 250	2 935	1 523	2 096	0 210	1 107	3 428	1 759	2 346	0 439	2 896	4 918
8040000	1 370	2 231	0 087	0 183	2 850	1 885	2 608	0 141	1 086	3 763	2 036	3 167	0 373	1 785	4 390
10686400	1 641	2 649	0 104	0 181	3 083	1 982	2 886	0 128	1 326	4 366	2 165	3 202	0 173	1 921	4 899
13881600	1 855	3 096	0 071	0 209	3 513	2 023	3 250	0 105	1 604	5 029	2 159	3 306	0 123	2 187	5 695
17643600	2 231	3 692	0 077	0 196	4 436	2 677	3 974	0 119	1 592	5 778	3 351	4 243	0 149	2 328	6 388
22030400	2 523	4 088	0 069	0 263	4 808	2 813	4 301	0 178	2 079	6 641	3 130	4 701	0 411	2 732	7 645
27060000	3 187	4 968	0 072	0 179	5 481	3 358	5 166	0 111	2 320	7 707	3 645	5 320	0 164	2 948	8 678
32870400	3 445	5 539	0 081	0 290	6 117	3 774	5 771	0 104	2 569	8 583	4 082	6 088	0 179	3 188	9 445
39419600	3 718	6 081	0 066	0 166	6 439	3 855	6 266	0 203	2 826	9 402	3 997	6 430	0 456	3 606	10 382
46785600	4 035	6 686	0 081	0 339	8 318	4 829	7 072	0 176	3 285	10 650	5 254	7 721	0 827	4 896	12 216
55016400	4 391	7 229	0 071	0 183	9 103	4 840	7 985	0 143	3 477	11 715	5 258	8 802	0 236	4 750	12 827
64160000	4 855	8 326	0 088	0 352	9 363	5 118	8 612	0 139	3 792	12 673	5 518	9 39	0 374	5 180	13 895
74264400	5 370	9 041	0 070	0 160	10 041	5 834	9 727	0 171	4 080	14 080	5 999	9 918	0 307	5 474	15 390
85377600	6 146	10 418	0 076	0 482	084	6 546	10 738	0 124	4 280	5 280	6 787	10 800	0 294	5 279	16 345
97547600	6 885	11 089	0 084	0 19	11 772	7 001	11 874	0 189	4 703	16 588	7 87	2 215	0 332	6 294	16 352
110822400	7 018	11 738	0 077	6 042	18 333	7 760	12 43	0 231	6 249	19 072	8 381	12 771	0 344	6 817	20 124
125250000	8 306	13 840	0 0 0	6 79	20 638	8 384	13 849	0 315	6 541	20 584	8 477	14 876	0 383	6 637	21 607
140878400	8 871	14 206	0 078	6 945	21 418	8 806	14 681	0 200	7 046	22 128	9 139	15 272	0 440	7 251	22 947
157755600	9 130	15 677	0 066	0 189	16 908	9 833	15 277	0 226	6 196	22 873	10 004	17 226	0 445	8 047	25 191
175929600	9 848	16 779	0 072	0 564	8 439	10 090	16 971	0 152	6 833	23 892	10 496	17 472	0 825	8 116	25 482
195448400	10 169	17 884	0 08	0 209	18 370	10 668	18 113	0 438	6 978	25 710	11 097	18 469	0 968	9 072	28 185
216360000	11 301	18 948	0 076	0 6 7	19 892	12 218	20 550	0 176	7 785	28 739	4 310	22 985	0 848	11 086	34 375
238712400	12 848	20 486	0 073	0 171	20 933	2 940	20 981	0 285	7 785	29 225	13 447	21 628	0 665	9 747	32 140
262583600	12 410	20 886	0 070	0 707	23 192	13 548	22 039	0 268	8 443	30 949	13 802	22 487	1 003	10 780	33 677
287931600	12 374	21 486	0 072	0 206	23 412	13 847	23 240	0 380	8 637	32 526	14 789	24 814	0 680	10 856	36 587
314804400	15 116	24 820	0 070	0 808	28 952	15 394	25 411	0 206	9 297	35 210	15 645	26 385	0 725	11 456	38 284
343260000	14 393	25 081	0 076	12 151	37 490	15 229	26 120	0 548	12 476	39 348	15 946	27 999	0 892	12 720	40 910
373766400	15 289	26 679	0 070	1 100	28 412	16 185	27 499	0 254	10 601	38 567	17 137	29 002	0 859	13 337	42 771
405771600	15 865	27 378	0 072	0 208	29 083	16 845	28 856	0 490	10 928	40 462	18 103	29 606	1 047	14 099	44 290
439363600	16 880	29 902	0 070	0 968	31 462	17 628	30 502	0 260	11 744	42 792	18 820	3 824	0 982	14 734	47 873
475160400	17 956	31 127	0 078	0 235	41 166	21 233	34 438	0 549	12 431	47 642	27 849	40 680	1 261	16 393	50 683
512840000	20 072	33 845	0 080	048	35 041	21 084	35 068	0 311	12 573	48 290	23 030	37 423	1 132	15 811	53 225

Tabla D 6 Tiempo d jecuc ó usando 3 procesadores remotos

	Ti mp de	ecuci	mínim (seg)			Ti mp d	ecució promedi (seg)				Tiemp d	ecució máximo (seg)			
# d ps	p1	p2	mul	res p	tal	p1	p2	mul	res p	total	p1	p2	mul	res p	total
8400	0 044	0 099	0 082	0 071	0 300	0 095	0 132	0 112	0 080	0 359	0 156	0 181	0 132	0 085	0 466
65600	0 106	0 187	0 100	0 110	0 383	0 117	0 207	0 108	0 119	0 452	0 136	0 249	0 115	0 137	0 473
219600	0 166	0 280	0 1 3	0 166	0 562	0 290	0 450	0 124	0 90	0 767	0 823	0 789	0 144	0 222	1 071
518400	0 250	0 416	0 074	0 233	0 747	0 497	0 612	0 148	0 256	1 026	0 750	0 883	0 350	0 285	1 276
1010000	0 360	0 613	0 080	0 347	1 091	0 491	0 660	0 092	0 726	1 485	0 641	0 724	0 119	0 980	1 735
1742400	0 500	0 846	0 070	0 153	1 239	0 528	1 132	0 099	0 393	1 654	1 300	1 630	0 167	0 484	2 261
2763600	0 830	1 188	0 070	0 571	1 879	1 081	1 348	0 082	0 840	2 323	1 607	1 900	0 093	1 238	2 850
4121600	0 959	1 432	0 072	0 140	2 133	1 244	1 865	0 111	0 745	2 735	1 604	2 269	0 159	1 451	3 084
5864400	1 089	1 824	0 001	0 168	2 683	1 626	2 212	0 217	0 750	3 194	1 856	2 419	0 691	0 939	3 407
8040000	1 845	2 389	0 070	0 176	2 681	1 835	2 574	0 096	1 022	3 726	2 103	2 686	0 113	1 697	4 220
10686400	1 835	2 842	0 070	0 181	3 204	1 990	2 956	0 101	76	4 298	2 216	3 194	0 132	1 917	4 981
13881600	1 950	2 945	0 082	0 231	3 533	2 131	3 262	0 085	1 717	5 198	2 240	3 408	0 134	2 133	5 731
17643600	2 575	3 904	0 075	1 652	5 988	2 766	4 238	0 128	2 022	6.804	3 210	4 594	0 173	2 609	7 123
22035400	2 440	4 135	0 076	0 304	4 068	2 874	4 322	0 125	2 015	6.831	2 765	4 962	0 184	2 683	7 056
27090000	3 188	5 043	0 071	0 188	6 147	3 484	5 294	0 127	2 322	7 852	3 916	5 760	0 200	2 905	8 402
32870400	3 186	5 225	0 071	0 348	6 357	3 580	5 603	0 108	2 603	8 421	3 596	6 018	0 220	3 283	9 296
39419600	3 854	6 383	0 083	0 178	7 292	4 204	6 663	0 160	2 801	9 765	4 588	6 911	0 294	3 501	10 614
46788800	4 391	7 116	0 080	0.330	7 764	4 715	7 279	0 219	3 254	10 871	5 569	7 393	0 757	4 420	11 901
55016400	4 787	8 128	0 087	0 181	8 682	5 018	8 375	0 170	3 486	12 147	5 109	8 756	0 331	4 804	13 507
64160000	5 377	8 811	0 081	0 19	9 653	5 610	9 044	0 124	3 940	13 210	5 800	9 499	0 329	5 076	14 450
74264400	5 929	9 789	0 072	0 188	10 293	6 308	9 975	0 198	3 899	4 229	6 980	10 430	0 393	4 848	15 75
85377600	6 602	0 7 9	0 0 3	0 407	11 343	6 863	10 951	0 211	4 3 2	15 591	7 220	11 134	0 412	5 356	16 053
97547600	6 688	11 350	0 08	0 196	12 123	7 082	11 900	0 233	4 871	16.514	7 349	11 811	0 487	5 682	16 108
110822400	7 857	2 365	0 001	0 446	13 042	7 859	12 567	0 148	5 337	18 220	8 196	12 932	0 482	6 832	20 177
125250000	8 443	13 795	0 072	0 185	14 685	8 645	14 166	0 266	5 423	20 013	9 8 2	14 466	0 821	7 257	21 946
140878400	9 165	14 784	0 07	0 817	15 642	9 421	14 850	0 176	5 834	21 044	10 277	14 914	0 563	7 647	22 771
157 81600	8 872	14 520	0 083	7 386	22 936	9 688	15 722	0 462	7 783	24 156	10 012	16 425	0 905	8 197	24 901
175929600	10 250	16 783	0 075	0 558	18 022	10 530	17 391	0 192	6 685	24 471	10 816	18 234	0 662	8 656	26 722
195448400	10 587	17 536	0 078	0 182	26 821	12 235	20 043	0.350	6 996	27 578	16 910	26 580	0 733	9 202	29 335
216360000	1 626	19 175	0 072	0 680	21 815	12 033	20 804	0 229	7 487	28 712	13 488	22 283	0 817	9 404	31 576
238712400	12 916	21 108	0 102	0 199	22 325	13 0 3	21 828	0 459	7 896	30 390	15 185	23 075	1 125	10 207	33 066
262553600	13 606	22 1 2	0 0 0	0 728	23 843	14 182	22 979	0 527	8.5 0	31 989	14 858	24 267	1 245	10 657	35 959
287931600	14 290	23 098	0 072	0 198	23 652	14 600	24 087	0 459	8 786	33 526	15 337	25 325	1 050	11 082	38 672
314894400	14 840	24 419	0 076	0 819	27 497	15 868	25 599	0 279	9 319	35 440	15 960	27 191	1 072	11 846	40 019
343490000	14 760	26 071	0 072	0 178	26 543	15 936	27 272	0 528	9 031	37 963	16 865	28 576	1 222	13 115	41 819
373766400	15 572	26 634	0 071	0 845	28 782	16 240	27 484	0 306	10 549	38 580	17 856	28 203	1 213	13 350	42 0 6
405771600	17 634	29 008	0 080	0 178	32 043	18 249	30 476	0 6 8	10 958	42.341	18 843	31 839	1 421	14 573	47 089
439553600	8 039	30 136	0 089	0 947	32 643	18 376	30 632	0 444	11 700	43 049	18 633	31 296	1 914	14 746	47 231
475160400	18 346	30 932	0 072	0 188	35 963	19 083	32 956	0 719	11 976	45 946	19 757	35 383	1 707	15 109	50 909
5 2640000	18 736	32 741	0 081	1 049	35 203	19 687	33 970	0 976	13 249	48.516	20 432	34 983	4 843	17 398	54 269

Tabla D 7 Tiempo de ejecución usando 1 procesador local

	Tiempo de ejecución mínimo (seg)					Tiempo de ejecución promedio (seg)					Tiempo de ejecución máximo (seg)				
# de pe	p1	p2	mul	res p	tal	p1	p2	mul	res p	total	p1	p2	mul	res op	tal
8400	0 000	0 000	0 009	0 000	0 016	0 007	0 002	0 018	0 000	0 027	0 026	0 007	0 036	0 000	0 062
65600	0 001	0 000	0 046	0 000	0 049	0 006	0 002	0 057	0 000	0 066	0 021	0 010	0 066	0 000	0 097
219600	0 001	0 000	0 155	0 000	0 162	0 003	0 001	0 174	0 000	0 178	0 010	0 001	0 219	0 000	0 221
518400	0 001	0 001	0 375	0 000	0 380	0 001	0 00	0 431	0 000	0 433	0 001	0 001	0 464	0 000	0 466
1010000	0 002	0 001	0 741	0 000	0 761	0 005	0 010	0 775	0 000	0 790	0 012	0 039	0 809	0 000	0 848
1742400	0 002	0 002	1 350	0 000	1 354	0 011	0 016	1 378	0 000	1 405	0 045	0 049	1 432	0 000	1 445
2765800	0 002	0 003	2 194	0 000	2 251	0 002	0 015	2 367	0 000	2 384	0 002	0 035	2 469	0 000	2 504
4121600	0 003	0 003	3 414	0 000	3 420	0 023	0 027	3 694	0 000	3 744	0 071	0 085	3 996	0 000	4 085
5864400	0 003	0 004	4 871	0 000	4 735	0 014	0 014	4 916	0 000	4 944	0 033	0 050	5 085	0 000	5 092
8040000	0 004	0 004	6 490	0 000	6 504	0 016	0 004	6 545	0 000	6 579	0 019	0 005	6 727	0 000	6 748
10696400	0 005	0 026	8 763	0 000	8 865	0 027	0 046	9 103	0 000	9 176	0 049	0 064	10 064	0 000	10 148
13881600	0 008	0 006	11 108	0 000	11 203	0 039	0 053	12 117	0 000	12 210	0 129	0 087	13 115	0 000	13 210
17643600	0 007	0 006	15 015	0 000	15 156	0 056	0 046	16 588	0 000	16 681	0 124	0 085	18 991	0 000	19 080
22030400	0 090	0 007	18 162	0 000	18 344	0 101	0 031	18 781	0 000	18 913	0 129	0 088	19 142	0 000	19 281
27090000	0 010	0 025	22 840	0 000	22 986	0 067	0 075	23 881	0 000	24 024	0 111	0 103	26 445	0 000	26 616
32870400	0 022	0 081	27 420	0 000	27 607	0 061	0 087	31 342	0 000	31 489	0 106	0 101	36 234	0 000	36 338
39419600	0 021	0 082	33 792	0 000	33 900	0 076	0 098	35 050	0 000	35 224	0 157	0 140	38 803	0 000	38 905
46788800	0 018	0 082	39 573	0 000	39 787	0 089	0 088	40 883	0 000	41 028	0 132	0 101	44 137	0 000	44 313
55016400	0 020	0 084	46 985	0 000	47 245	0 060	0 115	49 382	0 000	49 838	0 145	0 161	85 240	0 000	85 486
64 60000	0 091	0 014	55 107	0 000	55 393	0 115	0 073	60 786	0 000	60 977	0 199	0 098	65.584	0 000	65 898
74364400	0 083	0 016	64 778	0 000	64 962	0 080	0 100	69 598	0 000	69 787	0 104	0 156	77 054	0 000	77 275
85377600	0 023	0 086	72 212	0 000	72 466	0 091	0 3	76 819	0 000	76 040	0 159	0 163	89 026	0 000	89 208
97547600	0 165	0 088	84 511	0 000	84 778	0 179	0 096	88 243	0 000	88 518	0 193	0 109	99.589	0 000	100 156
1 0822400	0 032	0 088	98 589	0 000	98 850	0 138	0 123	102 161	0 000	102 422	0 208	0 182	113.312	0 000	1 3 512
2 250000	0 104	0 022	109 503	0 000	109 808	0 44	0 109	115 831	0 000	115 784	0 176	0 165	128 289	0 000	128 626
1408784400	0 147	0 093	124 810	0 000	24 863	0 176	0 127	126 565	0 000	126 668	0 191	0 163	128 427	0 000	128 734
1577884800	0 243	0 095	141 256	0 000	141 576	0 370	0 105	147 892	0 000	148 270	0 305	0 117	165.507	0 000	165 883
175929600	0 037	0 096	158 732	0 000	159 066	0 132	0 143	166 483	0 000	166 758	0 268	0 185	181 087	0 000	181 299
1954424400	0 104	0 168	175 767	0 000	176 129	0 35	0 178	183 989	0 000	184 305	0 182	0 18	206 34	0 000	206 432
2163600000	0 117	0 166	193 378	0 000	63 665	0 149	0 170	203 557	0 000	203 878	0 180	0 171	227 677	0 000	228 020
2387124000	0 049	0 114	215 538	0 000	2 8 852	0 171	0 162	234 462	0 000	234 795	0 263	0 176	257 131	0 000	257 466
2625134400	0 120	0 103	242 765	0 000	242 985	0 147	0 106	251 070	0 000	251 323	0 240	0 117	279 576	0 000	279 802
2878331600	0 118	0 176	263 866	0 000	264 248	0 76	0 79	279 227	0 000	279 582	0 220	0 186	298.551	0 000	298 916
3146894400	0 122	0 179	288 219	0 000	288 520	0 202	0 181	293 565	0 000	293 948	0 334	0 187	303 164	0 000	303 470
3434900000	0 130	0 179	311 633	0 000	312 244	0 210	0 185	340 583	0 000	340 945	0 300	0 200	366 180	0 000	366 591
3737664400	0 134	0 181	352 803	0 000	353 176	0 224	0 184	372 631	0 000	373 339	0 338	0 187	403 401	0 000	403 724
4057716000	0 138	0 185	383 124	0 000	383 482	0 212	0 86	395 638	0 000	396 035	0 340	0 187	414 000	0 000	414 391
4395583600	0 236	0 186	408 356	0 000	408 873	0 3 0	0 187	434 288	0 000	434 786	0 341	0 189	4 1 379	0 000	471 909
47 160400	0 147	0 187	452 447	0 000	452 906	0 272	0 189	460 201	0 000	460 662	0 348	0 190	471 883	0 000	472 488
512640000	0 204	0 191	485 046	0 000	486 358	0 296	0 205	538 843	0 000	539 343	0 460	0 258	577 479	0 000	87 966

II Distribución del tiempo de ejecución

Tabla D 8 Distrib ció del tiempo de jec ción

11 pro esadores remotos					
# d ops	%matriz 1	%mat iz 2	%mult	%resultados	matri 1 – matri 2
8400	48 78 %	37 73 %	27 50 %	17 64 %	11 05 %
65600	17 40 %	47 53 %	21 28 %	30 70 %	30 13 %
219600	40 34 %	56 74 %	7 35 %	35 76 %	15 40 %
518400	31 68 %	60 59 %	12 67 %	26 62 %	28 91 %
1010000	34 63 %	63 29 %	11 98 %	24 20 %	28 66 %
1742400	45 19 %	59 85 %	6 44 %	33 31 %	14 67 %
2763600	35 79 %	57 80 %	3 72 %	38 19 %	22 00 %
4121600	34 38 %	51 41 %	3 08 %	45 00 %	17 04 %
5864400	43 65 %	67 48 %	5 56 %	26 84 %	23 83 %
8040000	42 45 %	64 58 %	2 06 %	32 10 %	22 12 %
10696400	42 28 %	66 25 %	2 01 %	30 78 %	23 97 %
13881600	36 27 %	60 39 %	2 94 %	35 95 %	24 12 %
17643600	38 69 %	65 18 %	2 60 %	31 44 %	26 49 %
22030400	36 80 %	62 74 %	2 29 %	34 27 %	25 94 %
27090000	39 94 %	64 08 %	1 83 %	33 28 %	24 15 %
32870400	42 11 %	63 32 %	1 68 %	33 93 %	21 21 %
39419600	39 12 %	62 35 %	1 50 %	35 20 %	23 24 %
46785600	41 05 %	62 84 %	1 64 %	34 85 %	21 78 %
55016400	39 46 %	64 37 %	1 46 %	33 39 %	24 91 %
64160000	42 81 %	66 28 %	1 52 %	31 42 %	23 47 %
74264400	40 53 %	64 26 %	1 60 %	33 45 %	23 73 %
85377600	39 60 %	64 42 %	1 52 %	33 14 %	24 83 %
97547600	40 45 %	65 66 %	1 59 %	31 96 %	25 21 %
110822400	40 81 %	66 88 %	1 46 %	30 91 %	26 07 %
125250000	39 67 %	67 45 %	1 50 %	30 12 %	27 78 %
140878400	42 66 %	67 14 %	1 50 %	30 50 %	24 48 %
157755600	40 48 %	67 21 %	1 51 %	30 53 %	26 73 %
175929600	40 78 %	66 02 %	1 48 %	31 79 %	25 24 %
195448400	40 59 %	66 03 %	1 70 %	31 47 %	25 45 %
216360000	40 51 %	66 59 %	1 68 %	30 91 %	26 07 %
238712400	39 15 %	65 23 %	1 75 %	32 47 %	26 08 %
262553600	40 15 %	68 45 %	1 96 %	30 85 %	26 30 %
287931600	39 88 %	68 51 %	1 84 %	31 09 %	26 63 %
314894400	39 09 %	65 61 %	1 86 %	32 05 %	26 53 %
343490000	39 05 %	66 56 %	1 89 %	31 05 %	27 51 %
373766400	40 31 %	68 61 %	1 85 %	30 95 %	26 30 %
405771600	39 51 %	65 96 %	2 13 %	31 36 %	26 45 %
439553600	38 27 %	65 81 %	2 07 %	31 54 %	27 53 %
475160400	39 85 %	67 50 %	2 11 %	29 78 %	27 66 %
512640000	38 06 %	66 41 %	2 12 %	30 82 %	28 35 %
9 procesadores remotos					
# de op	%mat iz 1	%matriz 2	%mult	%resultado	mat 1 – matri 2
8400	28 38 %	38 10 %	33 52 %	25 34 %	9 73 %
65600	30 11 %	44 19 %	28 02 %	27 60 %	14 08 %
219600	45 80 %	56 47 %	18 23 %	24 94 %	10 66 %
518400	43 18 %	58 36 %	11 49 %	29 96 %	15 18 %
1010000	31 30 %	44 85 %	8 91 %	46 00 %	13.55 %
1742400	42 49 %	54 18 %	8 66 %	35 87 %	11 69 %
2763600	34 45 %	50 49 %	3 79 %	44 76 %	16 04 %
4121600	32 69 %	60 90 %	4 85 %	33 89 %	28 21 %
5864400	45 60 %	68 23 %	3 12 %	27 90 %	22 63 %
8040000	42 72 %	63 66 %	2 59 %	32 22 %	20 94 %
10696400	38 15 %	59 79 %	3 29 %	35 48 %	21 64 %
13881600	38 38 %	59 46 %	2 33 %	36 85 %	21 10 %
17643600	44 61 %	63 63 %	3 25 %	31 81 %	19 02 %
22030400	37 48 %	61 98 %	2 11 %	34 96 %	24 50 %
27090000	40 17 %	62 85 %	1 77 %	33 91 %	22 68 %

32870400	39 92%	63 08%	2 52%	33 47%	23 16%
39419600	40 01%	62 87%	1 64%	34 43%	22 86%
46785600	37 60%	64 18%	1 86%	33 17%	26 57%
55016400	38 85%	63 28%	1 67%	34 20%	24 44%
64160000	42 97%	65 10%	1 98%	32 29%	22 13%
74264400	38 52%	64 03%	1 90%	33 36%	25 50%
85377600	39 00%	63 78%	1 64%	33 79%	24 78%
97547600	39 38%	65 00%	1 88%	32 30%	25 62%
110822400	39 23%	64 48%	1 76%	33 03%	25 24%
125250000	39 13%	65 54%	1 86%	31 80%	26 41%
140878400	36 14%	65 72%	1 90%	31 65%	29 58%
157755600	39 00%	65 73%	1 80%	31 68%	26 73%
175929600	41 84%	70 55%	0 69%	28 03%	28 72%
195448400	41 95%	70 02%	1 18%	28 10%	28 07%
216360000	41 26%	71 61%	0 98%	26 79%	30 35%
238712400	43 24%	71 52%	1 02%	26 69%	28 28%
262553600	42 82%	71 10%	0 72%	27 49%	28 28%
287931600	42 51%	70 80%	1 51%	27 11%	28 29%
314894400	41 47%	71 48%	0 70%	27 13%	30 02%
343490000	40 12%	70 01%	1 47%	27 86%	29 89%
373766400	42 34%	71 53%	0 68%	27 08%	29 18%
405771600	42 40%	71 88%	1 13%	26 24%	29 48%
439553600	40 02%	70 31%	0 74%	28 44%	30 29%
475160400	40 19%	70 39%	1 45%	27 56%	30 20%
512640000	40 86%	71 26%	0 70%	27.37%	30 40%
7 procesadores remotos					
# de ops	%mat ls 1	%mat ls 2	%mult	% resultado	|matri 1 – matri 2|
8400	24 80%	31 13%	40 28%	26 97%	6 32%
65600	20 21%	44 66%	22 08%	32 89%	24 46%
219600	45 13%	58 47%	19 80%	20 68%	13 34%
518400	38 97%	56 36%	15 29%	27 09%	17 40%
1010000	46 12%	55 49%	11 24%	31 74%	9 37%
1742400	48 78%	64 60%	8 36%	26 65%	15 82%
2763600	40 51%	55 72%	8 22%	35 58%	15 21%
4121600	48 61%	70 30%	4 44%	23 89%	21 68%
5864400	44 62%	66 88%	5 48%	27 22%	22 27%
8040000	40 74%	65 63%	4 21%	29 21%	24 89%
10696400	44 07%	66 64%	2 47%	30 08%	22 57%
13881600	44 19%	63 52%	2 94%	31 94%	19 33%
17643600	44 78%	66 86%	2 80%	28 97%	22 08%
22030400	40 26%	64 43%	2 09%	31 90%	24 17%
27090000	43 53%	66 92%	1 72%	29 92%	23 38%
32870400	39 90%	65 99%	1 92%	30 74%	26 10%
39419600	39 32%	66 29%	1 74%	30 83%	26 97%
46785600	42 01%	68 42%	1 16%	29 52%	26 41%
55016400	38 12%	63 58%	1 79%	33 51%	25 46%
64160000	43 34%	67 95%	1 59%	29 34%	24 61%
74264400	39 12%	64 46%	1 51%	33 16%	25 34%
85377600	37 81%	66 63%	1 09%	31 42%	28 82%
97547600	39 85%	68 18%	3 47%	27 61%	28 33%
110822400	40 90%	69 66%	1 40%	28 16%	28 77%
125250000	39 96%	68 73%	2 94%	27 51%	28 77%
140878400	39 69%	69 77%	1 19%	28 23%	30 08%
157755600	41 00%	70 82%	1 33%	27 10%	29 82%
175929600	41 56%	70 74%	0 82%	27 83%	29 18%
195448400	43 05%	71 13%	1 21%	26 94%	28 08%
216360000	41 31%	70 75%	0 78%	27 83%	29 44%
238712400	41 73%	70 70%	1 70%	26 59%	28 97%
262553600	38 64%	66 49%	1 29%	31 51%	27 86%
287931600	38.80%	68 72%	1 79%	28 86%	29 92%
314894400	40 95%	73 97%	0 66%	24 86%	33 02%
343490000	40 91%	71 36%	1 29%	26 68%	30 45%

373766400	42 26 %	71 27 %	0 73 %	27 26 %	29 01 %
405771600	42 55 %	71 86 %	1 32 %	26 27 %	29 31 %
439553600	41 20 %	71 12 %	1 33 %	27 03 %	29 92 %
475160400	41 44 %	71 19 %	1.37 %	26 77 %	29 75 %
512640000	40 72 %	70 70 %	1 51 %	27 14 %	29 99 %
5 procesadores remotos					
# de ops	%matriz 1	%mat iz 2	%mult	%resultados	matri 1 – matri 2
8400	26 69 %	39 96 %	33 72 %	26 27 %	13 27 %
65600	33 97 %	43 00 %	27 09 %	29 73 %	9 03 %
219600	30 33 %	40 42 %	9 96 %	49 13 %	10 09 %
518400	45 30 %	60 83 %	14 61 %	24 11 %	15 53 %
1010000	47 16 %	60 94 %	8 02 %	30 64 %	13 78 %
1742400	47 94 %	69 62 %	5 88 %	24 38 %	21 68 %
2763600	39 39 %	53 05 %	6 36 %	40 03 %	13 66 %
4121600	49 89 %	67 93 %	3 18 %	28 29 %	18 04 %
5864400	44 46 %	61 21 %	6 12 %	32 34 %	16 75 %
8040000	44 11 %	66 83 %	3 76 %	28 19 %	22 72 %
10696400	42 64 %	66 09 %	2 94 %	30 37 %	23 45 %
13881600	40 22 %	64 62 %	2 08 %	31 90 %	24 39 %
17643600	46 36 %	68 81 %	2 06 %	27 57 %	22 45 %
22030400	42 35 %	64 77 %	2 69 %	31 30 %	22 42 %
27090000	43 57 %	67 02 %	1 44 %	30 10 %	23 45 %
32870400	44 12 %	67 48 %	1 22 %	30 04 %	23 36 %
39419600	41 00 %	66 66 %	2 16 %	30 05 %	25 66 %
46785600	42.53 %	66 40 %	1 66 %	30.85 %	23 88 %
55016400	41 32 %	67 99 %	1 22 %	29 68 %	26 67 %
64160000	40 36 %	67 96 %	1 09 %	29 92 %	27 60 %
74264400	41 46 %	69 14 %	1 22 %	28 79 %	27 67 %
85377600	42 89 %	70 37 %	0 82 %	28 05 %	27 48 %
97547600	42 20 %	69 77 %	1 14 %	28 35 %	27 57 %
110822400	40 69 %	65 21 %	1 21 %	32 77 %	24 52 %
125250000	40 15 %	66 41 %	1 51 %	31.32 %	26 26 %
140878400	40 07 %	66 35 %	0 94 %	31.84 %	26 28 %
157755600	42 13 %	71 16 %	0 99 %	27 10 %	29 04 %
175929600	42 23 %	71 03 %	0 76 %	27 34 %	28 80 %
195448400	41 49 %	70 45 %	1 71 %	27 14 %	28 96 %
216360000	42 50 %	71 51 %	0 61 %	27 02 %	29 00 %
238712400	44 28 %	71 72 %	0 97 %	26 53 %	27 45 %
262553600	43 13 %	71 21 %	0 86 %	27 28 %	28 08 %
287931600	42 57 %	71 45 %	1 17 %	26 55 %	28 88 %
314894400	43 69 %	72 17 %	0 59 %	26 40 %	28 48 %
343490000	38 70 %	66 38 %	1 39 %	31 71 %	27 68 %
373766400	41 88 %	71 30 %	0 66 %	27 49 %	29 42 %
405771600	41 63 %	71 32 %	1 14 %	27 01 %	29 69 %
439553600	41 19 %	71 28 %	0 61 %	27 44 %	30 08 %
475160400	44 57 %	72 26 %	1 15 %	26 09 %	27 70 %
512640000	43 66 %	72 60 %	0 64 %	26 04 %	28 94 %
3 pro esadores remotos					
# d op	%mat iz 1	%mat iz 2	% mult	%resultados	mat 1 – matri 2
8400	26 55 %	36 75 %	31 34 %	22 25 %	10 21 %
65600	27 06 %	47 91 %	24 37 %	27 53 %	20 85 %
219600	37 76 %	58 64 %	16 11 %	24 82 %	20 89 %
518400	48 44 %	59 61 %	14 46 %	24 94 %	11 17 %
1010000	32 97 %	44 37 %	6 21 %	48 78 %	11 40 %
1742400	50 04 %	68 44 %	6 00 %	23 78 %	18 39 %
2763600	45 68 %	59 77 %	3 54 %	36 15 %	14 09 %
4121600	45 49 %	68 18 %	4 05 %	27 22 %	22 70 %
5864400	50 91 %	69 26 %	6 79 %	23 47 %	18 35 %
8040000	49 20 %	69 10 %	2 57 %	27 43 %	19 90 %
10696400	46 31 %	68 86 %	2 35 %	27 38 %	22 54 %
13881600	40 99 %	62 76 %	1 64 %	33 04 %	21 77 %
17643600	42 52 %	65 16 %	1 96 %	31 09 %	22 64 %

22030400	39 42 %	66 18 %	1 91 %	30 85 %	26 76 %
27090000	44 37 %	67 43 %	1 62 %	29 57 %	23 06 %
32870400	42 16 %	66 54 %	1 28 %	30 91 %	24 38 %
39419600	43 06 %	68 23 %	1 64 %	28 68 %	25 18 %
46785600	43 37 %	66 96 %	2 01 %	29 93 %	23 58 %
55016400	41 28 %	68 93 %	1 40 %	28 71 %	27 64 %
64160000	42 47 %	68 47 %	0 94 %	29 83 %	26 00 %
74264400	44 31 %	70 10 %	1 39 %	27 40 %	25 79 %
85377600	44 21 %	70 24 %	1 35 %	27 66 %	26 03 %
97547600	42 88 %	70 18 %	1 41 %	27 68 %	27 30 %
110822400	43 13 %	69 14 %	0 81 %	29 29 %	26 01 %
125250000	43 20 %	70 78 %	1 33 %	27 10 %	27 58 %
140878400	44 77 %	70 56 %	0 84 %	27 72 %	25 80 %
157755600	39 98 %	65 08 %	1 91 %	32 26 %	25 10 %
175929600	43 07 %	71 07 %	0 78 %	27 32 %	28 00 %
195448400	44 36 %	72 68 %	1 23 %	25 37 %	28 31 %
216360000	41 91 %	72 46 %	0 80 %	26 08 %	30 55 %
238712400	42 82 %	71 83 %	1 51 %	25 98 %	29 01 %
262553600	44.34 %	71 83 %	1 02 %	26 60 %	27 50 %
287931600	43 73 %	71 76 %	1 37 %	26 21 %	28 03 %
314894400	43 93 %	72 23 %	0 79 %	26 29 %	28 30 %
343490000	41 95 %	71 78 %	1 39 %	26 14 %	29 84 %
373766400	42 09 %	71 24 %	0 79 %	27 34 %	29 14 %
405771600	43 10 %	71 98 %	1 45 %	25 88 %	28 88 %
439553600	42 67 %	71 16 %	1 03 %	27 18 %	28 48 %
475160400	41 54 %	71 69 %	1 56 %	26 05 %	30 16 %
512640000	40 58 %	70 02 %	2 01 %	27 31 %	29 44 %

1 procesado local

# d ops	%matriz 1	%mat iz 2	% mult	%resultado	|matri 1 − matri 2|
8400	25 55 %	7 30 %	67 15 %	0 00 %	18 25 %
63600	9 39 %	3 64 %	86 97 %	0 00 %	5 76 %
219600	1 58 %	0 45 %	97 97 %	0 00 %	1 13 %
518400	0 23 %	0 23 %	99 54 %	0 00 %	0 00 %
1010000	0 63 %	1 29 %	98 08 %	0 00 %	0 66 %
1742400	0 75 %	1 15 %	98 09 %	0 00 %	0 40 %
2763600	0 08 %	0 62 %	99 30 %	0 00 %	0 54 %
4121600	0 60 %	0 73 %	98 66 %	0 00 %	0 13 %
5864400	0 28 %	0 28 %	99 43 %	0 00 %	0 00 %
8040000	0 15 %	0 07 %	99 78 %	0 00 %	0 09 %
10696400	0 29 %	0 50 %	99 21 %	0 00 %	0 21 %
13881600	0 32 %	0 44 %	99 24 %	0 00 %	0 11 %
17643600	0 39 %	0 27 %	99 34 %	0 00 %	0 12 %
22030400	0 53 %	0 16 %	99 30 %	0 00 %	0 37 %
27090000	0 28 %	0 31 %	99 41 %	0 00 %	0 03 %
32870400	0 19 %	0 28 %	99 53 %	0 00 %	0 08 %
39419600	0 21 %	0 28 %	99 51 %	0 00 %	0 06 %
46785600	0 22 %	0 21 %	99 57 %	0 00 %	0 01 %
55016400	0 12 %	0 23 %	99 65 %	0 00 %	0 11 %
64160000	0 19 %	0 12 %	99 69 %	0 00 %	0 07 %
74264400	0 13 %	0 14 %	99 73 %	0 00 %	0 01 %
85377600	0 11 %	0 17 %	99 72 %	0 00 %	0 05 %
97547600	0 20 %	0 11 %	99 69 %	0 00 %	0 09 %
110822400	0 13 %	0 12 %	99 75 %	0 00 %	0 01 %
125250000	0 12 %	0 09 %	99 78 %	0 00 %	0 03 %
140878400	0 14 %	0 10 %	99 76 %	0 00 %	0 04 %
157755600	0 18 %	0 07 %	99 74 %	0 00 %	0 11 %
175929600	0 08 %	0 09 %	99 84 %	0 00 %	0 01 %
195448400	0 07 %	0 10 %	99 83 %	0 00 %	0 02 %
216360000	0 07 %	0 08 %	99 84 %	0 00 %	0 01 %
238712400	0 07 %	0 07 %	99 86 %	0 00 %	0 00 %
262553600	0 06 %	0 04 %	99 90 %	0 00 %	0 02 %
287931600	0 06 %	0 06 %	99 87 %	0 00 %	0 00 %

314894400	0 07 %	0 06 %	99.87 %	0 00 %	0 01 %
343490000	0 06 %	0 05 %	99 88 %	0 00 %	0 01 %
373766400	0 06 %	0 05 %	99 89 %	0 00 %	0 01 %
405771600	0 05 %	0 05 %	99 90 %	0 00 %	0 01 %
439553600	0 07 %	0 04 %	99.89 %	0 00 %	0 03 %
475160400	0 06 %	0 04 %	99 90 %	0 00 %	0 02 %
512640000	0 05 %	0 04 %	99 91 %	0 00 %	0 02 %

III Operaciones de punto flotante por segundo

Tabla D 9 MFLOPS consumidos d rante el experimento

# d ops	Mflops 11prc	Mfl ps 9prc	Mflops 7prc	Mflops 5prc	Mflops 3prc	Mflops 1prc
8400	0 077	0 112	0 069	0 083	0 091	0 093
65600	0 533	0 613	0 602	0 602	0 656	0 143
219600	1 926	1 664	1 861	3 431	1 943	0 142
518400	5 959	5 959	5 133	5 344	7 005	0 137
1010000	12 785	14 225	11 744	14 638	12 625	0 136
1742400	23 232	18 939	14 052	24 891	24 891	0 129
2763600	34 982	34 982	27 636	33 702	39 480	0 126
4121600	45 796	50 884	47 375	58 880	57 244	0 121
5864400	73 305	65 892	72 400	83 777	64 444	0 126
8040000	95 714	89 333	88.352	120 000	114 857	0 124
10696400	118.849	108 044	120 184	102 850	152 806	0 122
13881600	109 304	122 846	185 088	195 515	223 897	0 125
17643600	150 800	160 396	207.572	229 138	241 693	0 118
22030400	183 587	174 844	275.380	319 281	289 874	0 121
27090000	216 720	206 794	288 191	376 250	381 549	0 119
32870400	250 919	231 482	382 214	405 807	462 963	0 120
39419600	281 569	254 320	474 935	414 943	474 935	0 117
46785600	258 484	246 240	550 419	577 600	779 760	0 118
55016400	321 733	302 288	486.871	774 879	821 140	0 117
64160000	325 685	320 800	720.899	729 091	1051 803	0 116
74264400	324 299	284 538	1031 450	1060 920	1031 450	0 115
85377600	389 852	349 908	1219 680	1094 585	1169 556	0 118
97547600	359 954	370 903	548 020	1161 281	1204 291	0 115
110822400	424 607	333 802	1335 210	1439 252	1816 761	0 112
125250000	392 633	359 914	390 187	1789 286	1739 583	0 114
140878400	414 348	353 966	1498 706	1806 133	1984 203	0 113
157755600	439 431	375 609	1714 735	2390 238	1900 670	0 112
175929600	501 224	2199 120	1999 200	2443 467	2409 995	0 111
195448400	443 194	2272 656	2832.586	2412 943	2505 749	0 111
216360000	444 271	2704 500	2963.836	2846 842	3005 000	0 112
238712400	460 835	2876 053	2486 588	3315 450	2340 318	0 109
262553600	417 414	3323 463	3500 715	3750 766	3750 766	0 108
287931600	472 794	3309 559	3199 240	3999 050	3999 050	0 109
314894400	474 954	3538 139	4313 622	4498 491	4143 347	0 109
343490000	484 471	3948 161	3859 438	4519 605	4770 694	0 110
373766400	510 610	4791 877	4199 622	5339 520	5264 315	0 106
405771600	471 827	5136 349	4363 135	5635 717	5072 145	0 106
439553600	471 624	5232 781	6021 282	6279 337	6370 342	0 108
475160400	465 387	5399 550	5109 252	6335 472	6599 450	0 105
512640000	484 995	6927 568	6031 059	6408 000	6328 889	0 105

IV Aceleración obtenida con procesadores remotos

Tabla D 10 Índice d aceleración de procesamie to co tra el núm ro de operaciones

# de ops	Aceleración 11prc	Aceleración 9prc	Aceleración 7prc	Acel ración 5prc	Aceleració 3prc
8400	0 010	0 044	0 040	0 033	0 034
65600	0 026	0 102	0 069	0 059	0 104
219600	0 062	0 151	0 138	0 071	0 151
518400	0 130	0 285	0 279	0 251	0 298
1010000	0 363	0 414	0 340	0 406	0 438
1742400	0 575	0 523	0 640	0 729	0 599
2763600	0 907	0 767	0 818	0 803	0 861
4121600	1 110	1 107	1 160	1 205	1 109
5864400	1 180	1 390	1 352	0 963	1 390
8040000	1 244	1 515	1 554	1 482	1 541
10696400	1 438	1 638	1 920	1 810	1 787
13881600	1 431	1 941	1 896	1 967	1 955
17643600	2 041	2 221	2 362	2 369	2 128
22030400	2 131	2 484	2 462	2 399	2 599
27090000	2 093	2 586	2 619	2 645	2 732
32870400	2 887	2 917	2 986	2 923	2 970
39419600	3 158	3 177	3 252	3 265	3 194
46785600	3 067	3 246	3 295	3 257	3 343
55016400	3 480	3 585	3 649	3 742	3 498
64160000	3 828	3 197	4 058	3 987	3.833
74264400	4 108	4 124	4 244	4 218	4 124
85377600	4 260	4 354	4 079	4 434	4 275
97547600	4 578	4 432	3 884	4 620	4 682
110822400	4 770	5 077	4 800	4 912	4 899
125250000	4 921	5 030	5 282	5 077	4 999
140878400	5 383	5 386	5 512	5 441	5 483
157755600	5 618	5 796	5 673	5 620	5 665
175929600	6 034	5 930	6 087	6 243	5 954
195448400	6 353	6 313	6 265	6 256	6 004
216360000	6.546	6 312	6 289	5 634	6 133
238712400	6 575	6 764	6 954	6 809	6 618
262553600	6 970	6 997	6 985	7 215	6 757
287931600	7 235	7 316	6 087	7 226	7 206
314894400	7 259	7 603	5 873	7 536	7 210
343490000	7 718	7 996	7 433	7 632	7 467
373766400	8 351	8 368	8 297	8 257	8 229
405771600	8 570	8 668	8 301	8 658	8 144
439553600	8 574	8 693	8 621	8 541	8 657
475160400	8 837	9 092	9 019	8 952	8 897
512640000	9 376	9 456	9 368	9 138	8 962

Tabla D 11 Índice d acel ració ut lizand procesamiento rem to vs Número d procesadores

# de prcs	Aceleración mínima	Aceleración promedio	Aceleración máxima
3	0 034	4 123	8 962
5	0 033	4 170	9 138
7	0 040	4 099	9 368
9	0 044	4 175	9 456
11	0 010	4 080	9 376