UNIVERSIDAD CATÓLICA DEL MAULE

Facultad de Ciencias de la Ingeniería

Escuela de Ingeniería Civil Informática

Profesor Guía

Nombre_profesor_Guía

TÍTULO DE LA TESIS

NOMBRE(S) ALUMNO(S) TESISTA(S)

Tesis para optar al

Título Profesional de Ingeniero Civil Informático

Talca, MES 20XX

UNIVERSIDAD CATÓLICA DEL MAULE

FACULTAD DE xCIENCIAS DE LA INGENIERÍA

ESCUELA DE INGENIERÍA CIVIL INFORMÁTICA

TESIS PARA OPTAR AL

TÍTULO PROFESIONAL DE INGENIERO CIVIL INFORMÁTICO

"TÍTULO DE LA TESIS"

NOMBRE(S) ALUMNO(S) TESISTA(S)

COMISIÓN EXAMINADORA	**FIRMA**
PROFESOR GUÍA NOMBRE_PROFESOR_GUÍA	_____
PROFESOR COMISIÓN NOMBRE_PROFESOR_COMISIÓN	_____
PROFESOR COMISIÓN NOMBRE_PROFESOR_COMISIÓN	_____
NOTA FINAL EXAMEN DE TÍTULO	_____

TALCA, MES DE 20XX

Sumario

resumen trabajo.... (máximo 3 páginas)

A continuación algunos ejemplos de figuras, referencia, cita, tabla y algoritmo:

Una referencia a la Figura 1 y a las subfiguras 2(a) y 2(b). Una cita a libro de Pthreads [NBF96] y OpenMP ([CJvdP07]).

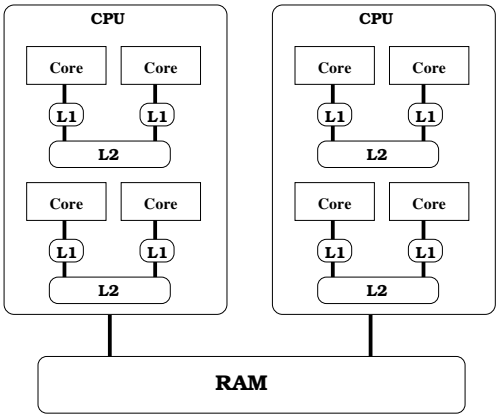

Figura 1: Plataforma multi-core.

```
int tid = ID_Thread
...
int ref1 = myArray[tid] * 1;
__syncthreads();
myArray[tid + 1] = 2;
__syncthreads();
int ref2 = myArray[tid] * 1;
result[tid] = ref1 * ref2;
...
```

(a) Caption Subfigura 1

```
int tid = ID_Thread
...
if (tid < warpSize) {
int ref1 = myArray[tid] * 1;
myArray[tid + 1] = 2;
int ref2 = myArray[tid] * 1;
result[tid] = ref1 * ref2;
}
...
```

(b) Caption Subfigura 2

Figura 2: Ejemplos para ilustrar la sincronización de los threads de un *warp*.

Algoritmo 1 *EGNAT*: búsqueda por rango r para la consulta q.

busquedarango(Nodo *P*, Consulta *q*, Rango *r*)

1: {Sea R el conjunto resultado}
2: $R \leftarrow \emptyset$
3: $d \leftarrow dist(p_0, q)$
4: **if** $d <= r$ **then**
5: se reporta p_0
6: **end if**
7: $range(p_0, q) \leftarrow [d - r, d + r]$
8: **for all** $x \in P$ **do**
9: **if** $range(p_0, q) \cap range(p_0, D_{p_x}) \neq \emptyset$ **then**
10: se agrega x a R
11: **if** $dist(x, q) <= r$ **then**
12: se reporta x
13: **end if**
14: **end if**
15: **end for**
16: **for all** $p_i \in R$ **do**
17: busquedarango(D_{p_i},q,r)
18: **end for**

Tabla 1: Características Generales

Processor	2xIntel Quad-Xeon (2.66 GHz)
L1 Cache	8x32KB + 8x32KB (inst.+data) 8-way associative, 64byte per line
L2 Unifed Cache	4x4MB (4MB shared per 2 procs) 16-way associative, 64 byte per line
Memory	16GBytes (4x4GB) 667MHz DIMM memory 1333 MHz system bus
Operating System	GNU Debian System Linux kernel 2.6.22-SMP for 64 bits

Índice general

Índice de figuras

Índice de tablas

Índice de Algoritmos

Capítulo 1

Introducción

texto...

1.1. Objetivos

El objetivo general de esta tesis es ....

Los objetivos específicos son los siguientes:

- objetivo-especifico-1
- objetivo-especifico-2
- objetivo-especifico-N

Capítulo 2

Estado del Arte

El paralelimos en computación es una técnica de programación la cual permite que varias intrucciones se puedan ejecutar similtaneamente. Se base en el principio de la división de grandes problemas en varios problemas mas pequeños, que son resueltos de forma concurrente.

2.1. Tipos de paralelismo

Existen diversos tipos de niveles de paralelismo que se describiran a continuación:

2.1.1. Nivel de bit

En la decada de 1970 hasta alrededor de 1986, la aceleración en la arquitectura de computadores lo logro aumentar el tamaño de una palabra, reduciendo asi el numero de instrucciones que un procesador debia realizar, de este modo se logro por ejemplo reducir el proceso de instrucciones en un procesador de 8 bits al sumar dos números de 16 bits.

2.1.2. Nivel de instrucción

Los avances en esta área se realizarón alrededor de 1980 y 1990, donde esto constaba con el reordenamiento y combinación de instrucciones en grupos que luego son ejecutadas en paralelo sin cambiar el resultado del programa. Actualmente los procesadores cuentan con "pipeline"de instrucciones de varias etapas, Cada etapa en el pipeline corresponde a una acción diferente que

el procesador realiza en la instrucción correspondiente a la etapa; un procesador con un pipeline de N etapas puede tener hasta n instrucciones diferentes en diferentes etapas de finalización

2.1.3. Datos

El paralelismo de datos es el paralelismo inherente en programas con ciclos, de este modo corresponde a la subdivición de los datos de entrada a un programa, de modo que a cada procesador le corresponda un subconjunto de los datos divididos.
Este paradigma es mejor orientado para operaciones con matrices o vectores, debido que se realiza la misma operación sobra cada uno de los elementos.

2.1.4. Tareas

El paralelismo de tareas es la característica de un programa paralelo en la que cálculos completamente diferentes se pueden realizar en cualquier conjunto igual o diferente de datos. Esto contrasta con el paralelismo de datos, donde se realiza el mismo cálculo en distintos o mismos grupos de datos. El paralelismo de tareas por lo general no escala con el tamaño de un problema.

Actualmente la computación paralela ha aumentado su interes debido a las restricciones fisicas, en base a la arquitectura computacional que se han masificados los computadores con procesadores múltinúcleos. Según el hardware que posee el equipo se puede clasificar el nivel de paralelismo que admite, como por ejemplo los computadores multinucleos y multiprocesos permiten realizar varios procesamientos en una misma maquina, mientras que a su vez los clusters, los MPP y los grids emplean varios computadores para trabajar en la misma tarea.
En los ultimos años el uso de coprocesadores ha sido un área de basto interes en temas de investigación en el campo de la computación paralela, con el fin de acelerar procesos secuenciales, los cuales pueden ser acelerados con los procesadores propios del equipo o bien con coprocesadores integrados.

2.2. Procesadores y Coprocesadores

2.2.1. Procesadores

Un multiprocesador es un conjunto de dos o más procesadores los cuales se encuentran a menudo integrados en un solo circuito. Hoy en día los multiprocesadores mas conocidos son dual-core, quad-core, octa-core con dos, cuatro y ocho núcleos respectivamente. Estos multiprocesadores permiten un paralelismo a nivel de hilos (threads) conocido como TLP (thread-level parallelism), aca no se consideran los microprocesadores en paquetes separados.
Es posible que se realice un procesamiento simultaneo utilizando dos o más procesadores en un pc, o bien realizando la conección de dos o más pc, este ultimo suele ser mas restrintivo debido a las caracteristicas de cada uno de los pcs conectados, que en algunos casos se realiza la superposición entre ellos.

2.2.2. Coprocesadores

Un coprocesador es un microprocesador de un ordenador utilizado como suplemento de las funciones del procesador principal (la CPU). Las operaciones ejecutadas por uno de estos coprocesadores pueden ser operaciones de aritmética, procesamiento gráfico, de señales, procesado de texto o criptografía, etcétera.
Las lineas más famosas de coprocesadores en el ámbito de la investigación son INTEL con su tecnología MIC y NVIDIA GPU con su tecnología CUDA

2.2.3. Intel

Intel a desarrollado una tecnología propía, orientada a computo intensivo en computación paralela conocido como MIC (Many integrated core) asociados a su familia Xeon, similar a la Nvidia Tesla que forman una arquitectura de alto rendimiento.
Se puede considerar que cada núcleo del co-procesador Xeon Phi es una unidad de ejecución multi-hilo completamente funcional. Poseen un rendimiento de hasta 1,2 teraFlops (operaciones de coma flotante por segundo) por coprocesador para algunas de las aplicaciones más exigentes de hoy día.

2.2.4. NVIDIA

NVIDIA se ha dedicado a la elaboración de GPUs (Unidades de procesamiento gráfico) los cuales son procesadores de varios núcleos que ofrecen un alto rendimiento, tambien dedicados al procesamiento de graficos o bien operaciones de punto flotante.
NVIDIA ha desarrollado una tecnología de acuerdo a su hardware conocida como CUDA, que cuenta con una plataforma de computación paralela y un modelo de programación. Esta tecnología permite aumentos considerables en el rendimiento computacional al aprovechar la potencia de la GPU.

2.3. KNN (k-nearest neighbors)

Conocido por sus siglas en ingles como KNN (K-nearest neighbors) k vecinos más cercanos, es un método de clasificación de objetos, siendo la clafisicación de objetos un área muy importante en variados campos de investigación y de aplicaciones, como lo son el reconocimiento de patrones, inteligencia artificial, psicología cognitiva, analísis de videos y medicina.

Capítulo 3

Marco Teórico

bla bla....

Capítulo 4

Desarrollo

4.1. Introducción

texto...

Capítulo 5

Experimentos

texto...

Capítulo 6

Conclusiones

texto...

6.1. Trabajos Futuros

- uno...
- dos...
- N...

6.2. Contribuciones de la Tesis

- uno...
- dos...
- N...

Bibliografía

[CJvdP07] Barbara Chapman, Gabriele Jost, and Ruud van der Pas. *Using OpenMP: Portable Shared Memory Parallel Programming*. The MIT Press, 2007.

[NBF96] Bradford Nichols, Dick Buttlar, and Jacqueline Proulx Farrell. *Pthreads Programming: A POSIX Standard for Better Multiprocessing*. O'Reilly, 1996.