Seminario Informativo del LICPaD*
"Paralelismo"

Germán Bianchini

Laboratorio de Investigación en Cómputo Paralelo/Distribuido - DISI
Universidad Tecnológica Nacional - Facultad Regional Mendoza,
CP (M5502AJE (Mendoza) Argentina
gbianchini@frm.utn.edu.ar

Resumen Paralelismo, o procesamiento paralelo, es una forma eficaz de explotar los sucesos concurrentes en el proceso de computación. De esta forma varios cálculos pueden realizarse simultáneamente. Basándonos en el principio de dividir grandes problemas para obtener varios problemas pequeños, éstos pueden posteriormente solucionarse en paralelo. El paralelismo se ha ido empleando cada vez más durante los últimos años, fundamentalmente para el Cómputo de Alto Rendimiento. En este documento trataremos brevemente los conceptos fundamentales asociados al paralelismo, así como otras nociones vinculadas a la idea del procesamiento paralelo en general.

1. Introducción

El modelo de computadora que más éxito ha tenido y que más ha aportado al desarrollo de la informática es el *modelo secuencial*, derivado de la concepción que tuvo John von Neuman (1903-1957). Según este modelo, una computadora secuencial lleva a cabo operaciones de procesamiento de una en una sucesivamente [4], operando sobre un esquema compuesto por: la memoria, la unidad aritmético-lógica, la unidad de control del programa y los equipos de entrada salida.

Obviamente, esta aproximación omite el trabajo simultáneo de las distintas unidades funcionales y de los circuitos que integran la computadora, pero ofrece una visión de alto nivel que permite explotar al máximo el conjunto de elementos, considerado como un todo. Básicamente, en este modelo se lee una instrucción de memoria, se decodifica, se obtienen los operandos de memoria, se ejecuta la operación indicada en la instrucción, se lleva el resultado a memoria, se procede a la búsqueda de la siguiente instrucción, etc. A este procesamiento se lo conoce como *computación secuencial*. Sin duda, es una buena forma de organizar el procesamiento, y su generalización y adaptación a las computadoras actuales ha dado lugar a un sinfín de lenguajes. También, en numerosos paradigmas de

* Este documento pertenece a la serie de seminarios organizados por el LICPaD como actividad informativa de los miembros del mismo, y por ende es propiedad del autor y del Laboratorio. (Agosto de 2011)

programación, se considera que lo que se ofrece desde un nivel inferior es una arquitectura secuencial como se especifica en dicho modelo.

Otra aproximación es considerar un modelo que nos permita mantener un cierto grado de simultaneidad. En esta aproximación consideraremos que la computadora puede ejecutar varias instrucciones de forma simultánea, definiendo así un modelo de computadoras paralelas.

Denominaremos *Computación Paralela* a la computación que se lleva a cabo en este tipo de computadoras. Obviamente, la organización que se puede y debe hacer para procesar los datos de esta manera es distinta a la aproximación secuencial. Al permitirse la ejecución de distintas instrucciones de forma simultánea, se debe tratar de aprovechar al máximo esta situación para obtener el mejor rendimiento posible. No obstante, esto nos obliga a rediseñar los algoritmos y las estructuras de datos de forma de permitir la simultaneidad. También requiere nuevos paradigmas de programación que consideren tanto los aspectos conceptuales como las particularidades físicas asociadas a este tipo de computadoras. En resumen, se nos plantea así otro tipo de programación que llamaremos *Programación Paralela*. En esta clase de programación nos vemos en la necesidad de dividir un determinado conjunto de operaciones en partes independientes que se puedan ejecutar simultáneamente.

Así entonces, llamaremos *Paralelismo* a la posibilidad de dividir un problema en partes que puedan resolverse independientemente.

2. Necesidad de la Computación Paralela

Las computadoras secuenciales actuales ofrecen elevada potencia computacional, la cual ha venido incrementándose con el tiempo debido al desarrollo de la tecnología, lo que ha permitido mayor velocidad en los circuitos básicos de los procesadores y un alto grado de integración y miniaturización de los componentes electrónicos [1].

La pregunta siguiente sería si la velocidad de tales circuitos crecerá de forma ilimitada y si el proceso de integración y miniaturización continuará hasta el infinito. La respuesta es **no**. Existen límites físicos que acotan el crecimiento de las prestaciones que se pueden obtener mediante las mejoras tecnológicas.

En su momento (abril de 1965) Gordon Moore (co-fundador de Intel) realizó una proyección que indicaba que aproximadamente cada 18 meses se duplicaría el número de transistores en un circuito integrado. Algunos años después modificó su propia ley al afirmar que el ritmo bajaría, y que la capacidad de integración se duplicaría aproximadamente cada 24 meses. Esta progresión de crecimiento exponencial, duplicar la capacidad de los circuitos integrados cada dos años, es lo que se considera la *Ley de Moore* (Fig. 1). Sin embargo, el propio Moore puso fecha de caducidad a su ley hacia el año 2017 aproximadamente [5].

Volviendo a las limitaciones físicas, tenemos como límite la velocidad de la luz. Recordemos que las señales eléctricas son flujos de electrones que recorren y atraviesan puertas lógicas que son los microcircuitos electrónicos. Por lo tanto, cualquier señal tiene como límite los 300000 Km/seg (30 cm/ns), con lo cual

estamos prácticamente en el mismo orden de tiempos de ciclos de reloj que operan en unos pocos nanosegundos.

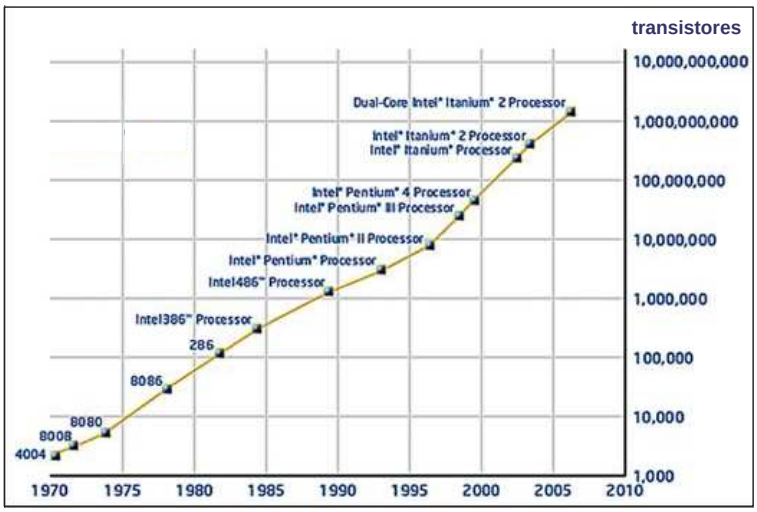

Figura 1. Ley de Moore

Por otra parte, la segunda pregunta que podemos formularnos es ¿por qué necesitaríamos mayor potencia computacional? ¿No nos bastaría con la potencia que nos ofrece un solo procesador? Otra vez, y a riesgo de parecer pesimistas, la respuesta es **no**. Actualmente existen problemas cuya solución conocemos de forma matemática, pero cuando tratamos de obtenerla se genera un número de cálculos elevado. Existen problemas cuya resolución implica un número de operaciones dado por una función que crece rápidamente con el tamaño del problema. Este tipo de problemas, denominados *problemas intratables*, se caracterizan porque dada una computadora cualquiera que sea capaz de realizar un número F de operaciones por segundo, siempre existe un problema n_0, razonablemente pequeño, a partir del cual el tiempo es excesivamente elevado para límites humanos razonables.

> *Ej. Consideremos el caso en que la función que representa el número de operaciones en función del tamaño del problema tomara la forma* $W = 10^n$*, y supongamos que se dispone de una computadora capaz de ejecutar* 10^{10} *operaciones por segundo. Si* $n = 100$*, el número de años necesario para ejecutar las operaciones necesarias sería* 10^{82}*, lo cual es inconcebible.*

En la actualidad, en muchos campos de la ingeniería y de la ciencia, existen problemas tratables pero con características que los hacen de difícil solución con las computadoras secuenciales. Algunos son *problemas de grandes dimensiones*

y otros *problemas de tiempo real*, que requieren respuestas en tiempos acotados. Estas clases de problemas son susceptibles de beneficiarse del cómputo paralelo. Un caso de particular atención son los *problemas de gran desafío* (**Grand Challenge Problems**)[2]. Estos son problemas que por su importancia social trascienden el ámbito científico y computacional, pero cuya resolución está ligada a la disponibilidad de herramientas computacionales de altas prestaciones. Algunos ejemplos son el estudio del genoma humano, la predicción climatológica a nivel mundial, el modelado de la biosfera, el modelado y la predicción de fenómenos sísmicos, oceanografía, la evolución del universo, etc. Muchos de estos problemas caen dentro del grupo de problemas tratables pero con alguna o varias limitaciones a raíz de los problemas mencionados, lo que los hace adecuados para ser planteados a través del uso de computadoras paralelas [6].

3. Aspectos de la programación paralela

El uso de la computación paralela implica el uso de técnicas de programación adecuadas al modelo computacional asociado a dichos sistemas de hardware, lo cual influye en los algoritmos y en las estructuras de datos.

Dado un problema, es necesario establecer primeramente cuáles son las partes independientes en las que el mismo se puede dividir. Luego es necesario especificar en qué recursos computacionales van a ejecutarse las tareas independientes. Finalmente, se debe garantizar la disponibilidad de los datos en las estructuras de datos a usar para aquellos procesadores que van a ejecutar procesos que accederán a tales datos. Además, debe existir una gestión completa de los procesos que garantice la actualización de los datos, que controle el orden de ejecución de los procesos que no son independientes, que reasigne procesos o redistribuya datos, etc. Todo este tipo de técnicas, que deben tener en cuenta el modelo computacional sobre el que se está trabajando, se engloban bajo el concepto de *programación paralela*.

La idea general y básica del cómputo paralelo es que n procesadores o nodos deberían proveer una velocidad computacional n veces mayor a la provista por un nodo simple, es decir, el problema debería ser resuelto en un período $1/n$ del tiempo insumido por un uniprocesador [3].

Sin embargo, ésta es una situación ideal que en la práctica no siempre se cumple. Debido a que existen ciertos límites de rendimiento teóricos y prácticos. Así como anteriormente comentamos las limitaciones físicas de la computación secuencial, las mismas existen para la computación paralela (por ej. el límite que impone la velocidad de la luz también está presente en las puertas lógicas de las computadoras paralelas). Asimismo, existe el límite impuesto por la acotación de grado, ya que es imposible establecer un número ilimitado de enlaces de comunicación entre los elementos de proceso de una computadora paralela.

No obstante, los límites impuestos por los sistemas paralelos son superiores a los límites de los uniprocesadores, y por ello, constituyen una plataforma ideal para aplicaciones que requieren altas prestaciones.

Algunos factores que limitan a las aplicaciones paralelas son los siguientes:

- Períodos en los que algunos procesadores permanecen ociosos
- Cálculos redundantes en cada nodo (por ejemplo recálculo de constantes)
- Comunicación entre procesos
- Regiones del programa no paralelizables, como inicialización y finalización de la aplicación.

Acerca de las regiones no paralelizables, podemos ver cómo impactan en el rendimiento general de una aplicación. Sean f la porción de tiempo requerida para procesar la región secuencial del programa y $T(1)$ el tiempo de ejecución secuencial de la aplicación, es decir en un solo procesador, el tiempo de ejecución con n nodos en paralelo será:

$$f * T(1) + (1 - f)\frac{T(1)}{n}$$

Más intuitivamente, en la Fig. 2 TS denota el tiempo insumido por la región secuencial, TP denota el tiempo insumido por la región paralela y por tanto el tiempo de ejecución con n nodos será $TS + TP$.

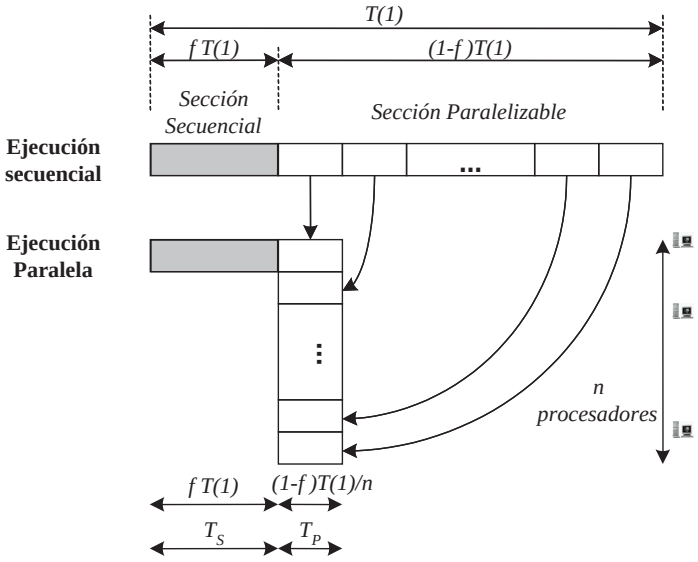

Figura 2. Porción paralela y secuencial de una aplicación

Con facilidad puede comprenderse por qué, a medida que se hace mayor la parte inherentemente secuencial de nuestra aplicación, se reduce la posibilidad de una paralelización ideal de nuestro código.

4. Conclusiones

En este breve resumen se ha realizado una introducción general acerca de algunos aspectos básicos del paralelismo, partiendo desde las necesidades de su utilización, pasando por sus características y aspectos generales, y viendo algunos usos particulares en los cuales actualmente se aplica, como así también las restricciones y dificultades de su utilización. Para tener una idea más acabada de todos los conceptos asociados al paralelismo, es menester un estudio más detallado y profundo del mismo, lo cual no es el objetivo de este documento. En el presente trabajo, asociado a las actividades del *Laboratorio de Investigación en Cómputo Paralelo/Distribuido*, hemos omitido una gran cantidad de temas vinculados como lo son las arquitecturas paralelas, las características de diseño, los modelos de software, las herramientas para la programación paralela, etc. Sin embargo, esperamos que el mismo sea de utilidad para aquellos que recién comienzan a indagar en esta rama de la informática, la cual se presenta como una puerta que nos conduce a infinitas posibilidades en el mundo de la ciencia y la industria.

Referencias

1. Francisco Almeida, Domingo Giménez, José Miguel Mantas, Antonio M. Vidal, "Introducción a la programación paralela". Paraninfo. 2008.
2. Answers.com: Information from Grand Challenge Problem. http://www.answer.com (Accedido en Agosto de 2011)
3. Grama A., Gupta A., Karypis G., Kumar V., "Introduction to Parallel Computing". Pearson Addison Wesley. Second Edition. 2003.
4. Andrew S. Tanenbaum. "Organización de Computadoras. Un enfoque estructurado. Tercera Edición". Chap. 1. Prentice Hall. 1992.
5. "Excerpts from A Conversation with Gordon Moore: Moore's Law" Video Transcript. Intel. 2005.
6. Richard S. Morrison "Cluster Computing. Architectures, Operating Systems, Parallel Processing & Programming Languages" 2002.