"Una Plataforma de Procesamiento en Paralelo Basado en MPI"

Jafet Gonzalez Castillo y Esteban Alonzo Rivero

29 de noviembre de 2010

1. Descripción del problema

Existe una problematica que se ha presentado desde los inicios de la computación, al principio las computadoras solo tenían la capacidad de atender un proceso a la vez y no era hasta terminar con la ejecución de dicho proceso que podían comenzar a ejecutar otro.

Para solucionar este problema se desarrollaron coordinadores y calendarizadores de procesos que utilizaban un enfoque de concurrencia, podemos imaginar esto como un semáforo en una avenida muy transitada y con muchas intersecciones, en este caso el semáforo nos ayuda a coordinar el paso de los carros (coordinador) y a decidir cada cuando le toca pasar a cada una de las diferentes intersecciones (calendarizador). Estos semáforos son necesarios si queremos que muchos carros (procesos) puedan utilizar la misma avenida ya que gestiona el uso de ésta.

En palabras más simples la concurrencia es la propiedad que tienen los sistemas computacionales de ejecutar más de un proceso.

En sistemas con un solo procesador los procesos concurrentes se turnan el procesador para ser ejecutados, es decir, cada proceso que se está ejecutando en el sistema tiene que competir por el uso del procesador, esta competencia por supuesto es dirigida por el coordinador y calendarizador de procesos que tienen todos los sistemas capaces de manejar procesos concurrentes. Gracias a esto es posible ejecutar más de un proceso al mismo tiempo, sin embargo, a pesar de que el sistema es capaz de coordinar y calendarizar la ejecución de más de un proceso a la vez, el procesador no puede; el procesador solo es capaz de ejecutar un proceso a la vez y a esto se le llama paralelismo simulado.Este problema se soluciona al tener más de un procesador aunque esto trae consigo otros problemas como el acceso a memoria y a los dispositivos periféricos.

Regresando al ejemplo de los semáforos podemos imaginar a la memoria y los dispositivos periféricos como la calle por la que transitan los carros. Si 2 o más carros llegaran a transitar por el mismo lugar al mismo tiempo se produciría una colisión, este mismo problema se tiene con la memoria y los dispositivos periféricos, esto se debe a que cada uno de los procesadores tiene los mismos derechos y permisos para acceder a la memoria y si 2 o más procesadores quieren

escribir sobre el mismo segmento de memoria al mismo tiempo se puede producir un error de integridad en los datos.

2. Justificación

Para solucionar el problema proponemos una serie de librerías que faciliten al estudiantado y al personal Académico el uso, comprensión e implementación de algoritmos sobre una plataforma de programación en paralelo. Agilizar el proceso de desarrollo de algoritmos en paralelo al encapsular y facilitar el uso de calendarización y coordinación del uso del procesador y el acceso a memoria que está presente en los sistemas operativos, por medio de un conjunto de bibliotecas encadenables para el lenguaje de programacion C. Si bien se sacrificara rendimiento a la hora de ejecución, se reducirá considerablemente el tiempo de desarrollo y este último es hoy día el recurso más costoso.

3. Marco teórico

Existen ya varias propuestas en el mercado que hacen del paralelismo algo transparente para el programador, a continuación presentamos unos ejemplos:

3.1. CUDA(Compute Unified Device Architecture)

Es una serie de bibliotecas de software propietario ,es decir, el código fuente no está disponible para el público en general, que permiten a los programadores usar una variación del lenguaje de programación C, el cambio en la sintaxis es tan sutil que es de rápido aprendizaje, reduciendo así la complejidad de desarrollar aplicaciones. Esta tecnología soluciona los problemas de coordinación y calendarizacion de procesos concurrentes. Los problemas que presenta la programacion con CUDA son que el programa final solo puede correr en las ultimas generaciones de hardware de la marca nVidia y solo funciona en sistemas operativos windows

3.2. ATI STREAM

Es una serie de bibliotecas en lenguaje C que facilitan la programacion en paralelo al abstraer al programador de la programacion de la coordinacion de procesos concurrentes. Sin embargo esta tecnología es dependiente del Hardware propietario y solo se ejecuta sobre GPUs marca AMD, no es capaz de coordinar procesos a través de la Red local y solo es compatible con sistemas operativos windows. Algunas de las ventajas de esta tecnología es que es open source, el paralelismo lo ejecuta a nivel de datos y la ganancia de desempeño es mayor que nuestra propuesta debido al hardware especializado.

3.3. AXUM

Creado por Microsoft, Axum es un conjunto de librerías que se integran al lenguaje de programacion C, esta orientado a objetos y basado en la CLR (Common Languague Rutine), basado en el modelo matemático Actor, no entraremos en detalles pero básicamente es un modelo matemático que soluciona problemas de concurrencia. Está pensado para el desarrollo de aplicaciones con problemas de concurrencia pero contiene suficientes clases de propósito general de tal manera que permite no tener que recurrir a otro lenguaje de programación para el desarrollo de software especializado.

En axum cada proceso es un agente (Actor) , dichos agentes están ligeramente acoplados (esto quiere decir que el número de dependencias entre agentes es mínima) , dichos agentes no comparten recursos de memoria pues usan un modelo de paso de mensajes propietario para la comunicación y coordinación entre procesos y las solicitudes de recursos entre estos llamados canales. Axum es actualmente un prototipo que se pretende esté integrado a la plataforma de visual studio en los siguientes años.

3.4. MatLab : Parallel Computing Toolbox

Te permite solucionar problemas computacionales y de uso tenso de datos usando procesadores multinucleo , GPUs y clusters de computadores. Por medio de estructuras de alto nivel, ciclos for paralelos, tipos especiales de arreglos y algoritmos numéricos paralelizados, y cabe mencionar que te permite paralelizar aplicaciones hechas en MATLAB sin necesidad de CUDA o programacion MPI.

4. Descripción de la solución

A pesar de que los sistemas actuales proporcionan las herramientas para que el programador pueda hacer uso de procesos concurrentes y memoria compartida, estas herramientas son bastante complejas de utilizar y en algunos casos necesitan hardware especifico, lo que propicia programas poco confiables, de difícil mantenimiento y/o con altos tiempos de desarrollo,sin mencionar que se necesita gente capacitada para desarrollar este tipo de algoritmos computacionales.

El objetivo de este protocolo es proporcionar una solución al problema de la complejidad y lentitud en el desarrollo de programas que usen las ventajas de los procesos concurrentes simultáneos (paralelismo real) y memoria compartida. Nuestro proyecto esta basado en un sistema que ataca específicamente el problema de la concurrencia (se podría ver como un paralelismo local),usando semáforos y colas de procesos para coordinar el acceso a variables en memoria compartida; por medio de dicho sistema podemos coordinar el acceso a variables en un ambiente distribuido con una memoria centralizada, ejecutando funciones por medio de una interfaz de paso de mensajes llamada mpi. De esta manera proponemos una serie de librerías capaces de encapsular el uso de mpi y un sis-

tema de control de concurrencia para facilitar la programacion de aplicaciones en un sistema distribuido.

5. Resultados y conclusiones

5.1. conclusiones

Hasta el momento estamos utilizando un enfoque de robustes vs performance , sacrificamos performance pero ganamos facilidad de mantenimiento,facilidad del manejo de errores , cuando decimos que perdemos performance nos referimos a que toda la memoria compartida está centralizada en un nodo maestro haciendo así un cuello de botella, por otro lado permitimos que el parelelismo sea en algunos casos especiales implícito.

5.2. Cosas a mejorar

El programa corre únicamente en sistemas operativos linux de 32bits debido a problemas de compatibilidad de manejo de memoria, uso de aritmética de apuntadores y otras instrucciones que no pudimos migrar a 64bits, también es necesario cambiar de c a c++ para orientarlo todo en objetos y tener un programa mas legible y fácilmente modificable.

Referencias

[1] Reyes, Flavio (1992), Diseño y Desarrollo de una Máquina Paralela de Visión", México, CINVESTAV.

[2] MPI: A Message-Passing Interface Standard Version 2.2 , Septiembre 4, 2009 , http:// www.mpi-forum.org/docs/mpi-2.2/mpi22-report.pdf

[3] developing on CUDA, http://developer.nvidia.com/object/gpucomputing.html

[4] http://www.amd.com/US/PRODUCTS/TECHNOLOGIES/STREAM-TECHNOLOGY/Pages/stream- technology.aspx

[5] Axum Language Spec , Language Overview , Niklas Gustafsson , Microsoft , Jun 16 , 2009

[6] http://www.mathworks.com/products/datasheets/pdf/parallel-computing-toolbox.pdf