ParallelPic

Erick Eduarte. Luis Felipe Rincón.

Universidad de Costa Rica

09/11/2013

Introducción

Este trabajo se basa en paralelizar la librería de procesamiento de imágenes ImageLib con dos diferentes paradigmas : OpenMP y OpenMPI, para luego poder comparar las ventajas de cada uno.

Definición

La programación en paralelo consiste en el uso simultáneo de múltiples recursos computacionales para resolver un problema en específico.

Programación en Paralelo

Definiciones

- Core: En procesadores multicore, un core es cada uno de los microprocesadores independientes del chip.
- Thread: Es la unidad de procesamiento más pequeña que puede ser planificada por un sistema operativo.

Conceptos Básicos

Definiciones

- Modelo de Memoria Distribuida: cada procesador puede tener su propia dirección de trabajo, la memoria se encuentra distribuida o separada para cada procesadory usa comunicación por paso de mensajes para realizar trabajos en conjunto.
- Modelo de Memoria Compartida: Se basa en el uso de threads, los threads de un mismo core de un procesador comparten el mismo bloque de memoria.

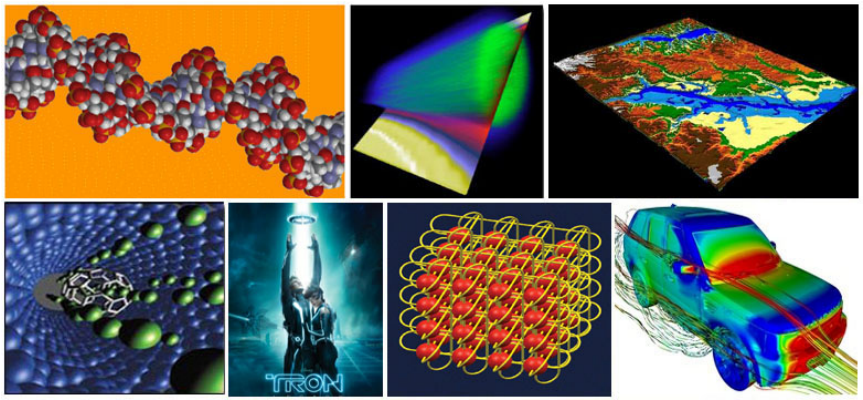
Aplicaciones

Ciencia e ingeniería

- Física
- Biociencia, biotecnología y genética
- Quimica y ciencia molecular
- Geología y sismología
- Ingeniería eléctrica
- Ingeniería mecánica

Industria y comercio

- Bases de datos
- Motores de búsqueda Web
- Gráficos y realidad virtual



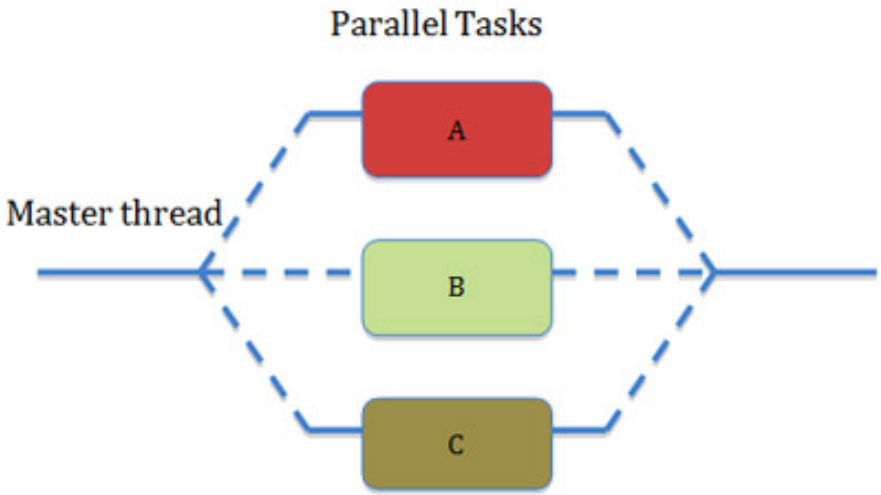
Ventajas

- Ahorro de tiempo y/o dinero
- Resulución de problemas más grandes
- Concurrencia
- Uso de recursos externos

OpenMP

Descripción

OpenMP corresponde a un paradigma de paralelización que se basa en el modelo de memoria compartida y trabaja bajo el denominado "Paralelismo Fork-Join".



OpenMP

Sintaxis

Para iniciar cualquier instrucción de OpenMP se utiliza "#pragma omp" seguido de los distintos parámetros disponibles.
#pragma omp parallel: Utilizado para iniciar un bloque de ejecución en paralelo el cual crea un grupo de threads.
Directiva for: Esta directiva indica que las iteraciones del ciclo siguiente se van a ejecutar por un equipo de threads en paralelo.

Cláusulas

A la directiva for se le pueden añadir distintas cláusulas:

- Schedule: Describe como serán divididas las iteraciones en el equipo de threads. Puede tener varios parámetros: Dynamic, Static, Auto, Runtime
- Ordered: Especifica que las iteraciones se realizan en el mismo orden que si fueran ejecutadas por un programa secuencial.
- Critical: Esta directiva indica que sólo un thread puede realizar la acción siguiente a la vez, mientras los otros esperan.
- Barrier: Esta directiva estabece un punto de espera hasta que todos los threads del grupo alcancen ese punto.

Atributos de las cláusulas

- Private: Declara las variables privadas para cada thread, esto quiere decir que se guarda una copia del mismo objeto para cada thread, para que los threads no interfieran entre sí, un ejemplo de esto son los índices de las iteraciones.
- Shared: Esta cláusula declara las variables públicas para todos los threads del equipo, y todos pueden accederlas para leer o escribir sobre ellas.

Rutinas

- omp_set_num threads: ajusta la cantidad de threads a utilizar en la siguiente región en paralelo.
- omp_get_num_thread: Retorna la cantidad de threads que están en el equipo de la sección en paralelo en donde se llama esta función.
- omp_get_max_threads: Retorna el máximo número de threads disponibles para utilizar.

OpenMPI

Descripción

OpenMPI está basado en el uso de memoria distribuida, un ejemplo de esto es un cluster de computadoras, en donde todos los procesadores pueden trabajar en paralelo y comunicarse por una red local, lo que hace que el acceso a memoria local sea mas rápida por lo que muchas veces se usan conexiones por infiniband.

OpenMPI

Sintaxis

Todas las instrucciones y tipos en un ambiente en paralelo comienzan con MPI_.
Comunicadores: MPI utiliza objetos llamados comunicadores para englobar grupos de procesos que se pueden comunicar entre sí.
Rank: Cada proceso tiene un entero asociado a él, que lo identifica en su respectivo comunicador.

Rutinas

- MPI_Init: Siempre para iniciar un ambiente en paralelo se pone esta función
- MPI_Comm_size: Retorna la catidad de procesos en el comunicador.
- MPI_Comm_rank: Retorna el número del proceso en el comunicador.
- MPI_Finalize: Siempre se debe poner esta función para finalizar un ambiente en paralelo.

Rutinas

- MPI_Barrier: Establece un punto de espera para todos los procesos del comunicador.
- MPI_Bcast: Envía un mensaje desde el proceso principal a todos los demás procesos del comunicador.
- MPI_Scatter: Distribuye distintos mensajes con partes de información a todos los procesos del comunicador.
- MPI_Gatter: Une todas las partes de la información que se distribuyó con el Scatter.
- MPI_Reduce: Realiza una operación a los resultados locales de cada proceso, y guarda el resultado en un sólo proceso. Algunas de las operaciones son: suma, resta, máximo...

Métricas De Análisis de Algoritmos en Paralelo

Aceleración

$$S_P = \frac{T_s}{T_P}$$

Aceleración Relativa

$$RelSp(I, P) = \frac{T(I,1)}{T(I,P)}$$

Ley de Amdahl

$$S_P = \frac{T_S}{T_P} \leq \frac{1}{f_s + \frac{f_p}{P}}$$

Eficiencia

$$E_P = \frac{S_P}{P} = \frac{T_S}{PT_P}$$

Escalabilidad

En sistemas escalables la eficiencia es constante independiente de la cantidad de trabajo y la cantidad de operaciones para resolverlo, y el número de procesadores o el hardware.

OpenMP vs OpenMPI

OpenMP

- Memoria compartida
- Sincronización
- Pragmas
- Grano grueso

OpenMPI

- Memoria distribuida
- Procesos independientes
- Sintaxis compleja
- Grano fino