Teoria-Tema-2-Arquitectura-de-co...

BlackTyson

Arquitectura de Computadores

2º Grado en Ingeniería Informática

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación
Universidad de Granada

Tema 2 Arquitectura de computadores.
Wuolah: **@BlackTyson**

Lección 4

1. **Problemas que plantea la programación paralela**. Punto de partida.
    a. Tenemos nuevos problemas que deben ser abordados por la herramienta de programación, por el programador o por el SO:
        - **División en tareas.**
        - **Agrupación de tareas o carga de trabajo en procesos.**
        - **Asignación a procesadores.**
        - **Sincronización y comunicación.**
2. **Herramientas para obtener código paralelo**
    a. Tipos:
        i. Las **más abstractas**: **API funciones**
        ii. Las intermedias: lenguajes paralelos y api funciones + directivas
        iii. Las **menos abstractas: compiladores paralelos.**
    b. Herramientas para obtener programas paralelos:
        i. Permiten de forma implícita o explícita:
            - Localizar paralelismo
            - Asignar las tareas a procesos o threads.
            - Crear y terminar procesos
            - Comunicar y sincronizar procesos.
        ii. El programador la herramienta o el SO se encarga de **asignar procesos a unidades de procesamiento (mapping)**
    c. Comunicaciones colectivas:
        i. Uno a todos:
            - **Difusión** (broadcast): se comunica el conjunto de información a todos los procesos.
            - **Dispersión**(scatter): se comunica una porción del conjunto a cada uno de los procesos, al ser sumada debe de ser el conjunto.
        ii. **Todos a uno**:
            - **Reducción**
            - **Acumulación.**
        iii. Comunicación múltiple uno-a-uno:
            - **Rotación:**
            - **Baraje-2**
        iv. Comunicación todos-a-todos:
            - **Todos difunden(all-broadcast)**
            - **Todos dispersan(all-scatter)**
        v. Servicios compuestos:
            - **Todos combinan**
            - **Recorrido prefijo paralelo**
            - **Recorrido sufijo paralelo.**
3. **Estilos/paradigmas de programación paralela**
    a. Estilos de programación:
        i. Paso de mensajes -> multicomputadores.
        ii. Variables compartidas -> multiprocesadores.
        iii. Paralelismo de datos -> procesadores matriciales

4. **Estructuras típicas de códigos paralelos:**
    a. **Master slave**: el master reparte trabajo a los slave los cuales recolectan resultados.
    b. **Cliente servidor**: **cliente pide información** al servidor y este arroja la respuesta.
    c. Descomposición de dominio.
    d. Estructura segmentada
    e. Divide y vencerás.

Lección 5:

1. **Proceso de paralelización:**
    a. **Descomponer en tareas independientes**: Se analiza la **dependencia entre funciones e iteraciones de bucles**
    b. **Asignar tareas a procesos y/o threads y mapeo a procesadores:**
        i. La **granularidad** asignada a los procesos **depende del número de cores y del tiempo de comunicación** frente a tiempo de cálculo
        ii. Debe haber un **equilibrio de carga** con el objetivo de que unos procesos no hagan esperar a otros. Depende de la arquitectura (mejor homogénea y uniforme) y de la aplicación.
        iii. Tipos de asignación:
            1. **Estática**: está **determinada** qué tarea va a realizar cada core. Asignada explícitamente por el programador e implícitamente por la herramienta de programación.
            2. **Dinámica**: se realiza en **tiempo de ejecución** con el objetivo de que distintas ejecuciones pueden asignar distintas tareas a un core. Es asignada explícitamente por el programador e implícitamente por la herramienta.
        iv. **El SO operativo se encarga del mapeo**, aunque puede hacerlo el entorno en tiempo de ejecución con la influencia del programador.
    c. **Redactar código paralelo**
    d. **Evaluar prestaciones**

Lección 6: evaluación de prestaciones

1. **Ganancia en prestaciones y escalabilidad:**
    a. Evaluación de prestaciones:
        i. Medidas usuales: tiempo respuesta y productividad
        ii. Escalabilidad
        iii. Eficiencia: relación prestaciones/máximas| rendimiento = prestaciones/recursos
    b. Ganancia en prestaciones. escalabilidad (prestaciones(p)/prestaciones(1) ó T_s/T_p). Se puede dar **sobrecarga** debido a la **comunicación y la creación de threads, de cálculos no presentes o de falta de equilibrado.**

2. **Ley de amdahl**
    a. La ganancia de prestaciones está **limitada** por la fracción de código no paralelizable.

$$S(p) = \frac{T_S}{T_P(p)} \leq \frac{T_S}{f \cdot T_S + \frac{(1-f)\cdot T_S}{p}} = \frac{p}{1+f(p-1)} \to \frac{1}{f} (p \to \infty)$$

    b. Para aumentar la ganancia se puede aumentar el tamaño del problema
3. **Ganancia escalable (Ley de gustafson)**

$$S(p) = \frac{T_S(n = kp)}{T_P} = \frac{fT_P + p(1-f)T_P}{T_P}$$

$$S(p) = p(1-f) + f$$

PREGUNTAS POSIBLES SOBRE LAS DIAPOSITIVAS(TEORÍA, NO EJERCICIOS):

Lección 4:

1. Encargado de abordar los problemas respecto a la programación secuencial: programador, herramienta o SO:

Lección 5:

1. Qué herramientas de programación son las más abstractas: API funciones.
2. Que permiten las herramientas de obtención de programas paralelos: localizar paralelismos, asignar, crear, terminar, comunicar y sincronizar procesos.
3. Cuáles son los tipos de comunicación todos-todos: todos dinfunden y todos dispersan
4. Y múltiples uno a uno: permutación-rotación y permutación-baraje

Lección 5:

1. De qué depende la granularización : del número de cores de procesamiento y del tiempo de comunicación frente al tiempo de cálculo.
2. Cual es el objetivo del equilibrado de la carga: que los procesos no hagan esperar a otros procesos.
3. Qué tipos de asignaciónes hay: estática y dinámica.

Lección 6:

1. Motivos de la sobrecarga en la ganancia de prestaciones: Se puede dar sobrecarga debido a la comunicación y la creación de threads, de cálculos no presentes o de falta de equilibrado.
2. Cómo podemos aumentar la ganancia de prestaciones: aumentando el tamaño del problema