Universidad Nacional de Colombia

Facultad de Ingeniería

Computación Paralela y Distribuida

Profesor: Oscar Agudelo R.

Control de lectura artículo Scheduling Multithreaded Computations by Work Stealing de R. D. Blumofe y C. E. Leiserson.

Nombre del estudiante: Gabriel Andres Anzola Tachak

1.

a. Describa de forma breve la diferencia entre work sharing y work stealing de acuerdo con lo expuesto en los dos primeros párrafos de la página 2 del artículo.

En el work sharing, cada vez que un procesador genera nuevos hilos, el planificador intenta migrar algunos de ellos a otros procesadores para distribuir el trabajo a procesadores subutilizados. En el work stealing, los procesadores subutilizados toman la iniciativa e intentan "robar" hilos de otros procesadores. La migración de hilos ocurre con menor frecuencia en el work stealing que en el work sharing, ya que cuando todos los procesadores tienen trabajo, no se migran hilos en el work stealing, pero siempre se migran hilos en el work sharing.

b. ¿Encuentra alguna diferencia en esta descripción y la que se hace en https://en.wikipedia.org/wiki/Work_stealing?

La descripción en Wikipedia añade que cada procesador tiene su propia cola de tareas y que los procesadores inactivos intentan robar trabajo de otros procesadores de forma aleatoria, algo que no se menciona en el artículo original. Además, Wikipedia menciona ejemplos de implementación en lenguajes como Cilk, Java y .NET, lo cual no está presente en el artículo de Blumofe y Leiserson.

---

2. Describa la organización del artículo. (último párrafo, sección 1)

El resto del artículo está organizado como sigue: En la Sección 2 se describe el modelo de cálculo multihilo utilizando un enfoque basado en grafos. En la Sección 3 se da un algoritmo simple que utiliza una cola centralizada. Este algoritmo de "hojas ocupadas" forma la base de nuestro algoritmo aleatorio de robo de trabajo, que presentamos en la Sección 4. En la Sección 5, introducimos el modelo de acceso atómico que utilizamos para analizar el tiempo de ejecución y los costos de comunicación del algoritmo de robo de trabajo. Para concluir, en la Sección 7, se discuten las aplicaciones de las ideas teóricas en este artículo al lenguaje de programación Cilk y a su sistema de ejecución en tiempo real.

---

3. De acuerdo con el modelo computacional multihilo descrito en el artículo defina

a. Activation frame
b. Tamaño de la pila (stack depth) de un hilo
c. Tamaño de la pila (stack depth) de una computación

- **Activation frame**: Es un bloque de memoria que se asigna a un hilo para almacenar los valores sobre los que opera el hilo.
- **Tamaño de la pila de un hilo**: Es la suma de los tamaños de los activation frames de todos los ancestros del hilo, incluido él mismo.
- **Tamaño de la pila de una computación**: Es el tamaño máximo de la pila de cualquier hilo en la computación.

---

4. Describa el algoritmo de Busy-Leaves y sus tres reglas (páginas 7 y 8 junto con la figura 1 y la figura 2 del artículo)

El algoritmo Busy-Leaves mantiene todas las hojas vivas ocupadas, es decir, garantiza que siempre haya al menos un hilo listo para ejecutarse en cada paso de la computación. Las tres reglas del algoritmo son:

1. **Spawns**: Si el hilo Ga genera un hijo Gb, el procesador coloca Ga en la cola de hilos y comienza a trabajar en Gb.
2. **Stalls**: Si el hilo Ga se bloquea, el procesador lo coloca en la cola de hilos y se queda sin trabajo.

3. **Dies**: Si el hilo Ga muere, el procesador revisa si su hilo padre Gb tiene hijos vivos. Si no tiene hijos vivos, el procesador toma a Gb de la cola y comienza a trabajar en él.

---

5. Escriba en español el enunciado del teorema 3 presentado en el artículo.

Para cualquier número P de procesadores y cualquier computación multihilo estricta con trabajo T_1, longitud del camino crítico T_∞ y profundidad de pila S_1, el Algoritmo de Hojas Ocupadas (*Busy-Leaves Algorithm*) calcula un plan de ejecución con P procesadores, cuyo tiempo de ejecución satisface $T(\sigma) \leq \frac{T_1}{P} + T_\infty$ y cuyo espacio satisface $S(\sigma) \leq S_1 P$.

---

6. Describa el algoritmo de Work Stealing aleatorio (página 10 del artículo).

En el algoritmo de Work Stealing aleatorio, cada procesador mantiene una cola de tareas listas para ejecutar (deque). Los procesadores trabajan en la tarea más nueva de su cola y roban la tarea más vieja de la cola de otro procesador cuando no tienen trabajo. Si un procesador termina su trabajo y su cola está vacía, intenta robar una tarea de la parte superior de la cola de otro procesador elegido aleatoriamente. Si tiene éxito, trabaja en la tarea robada; si no, sigue intentando robar hasta que tenga trabajo.

---

7. Escriba en español el Lema 4

En la ejecución de cualquier computación multihilo completamente estricta por el Algoritmo de *Work-Stealing*, considera cualquier procesador p y cualquier instante de tiempo en el que p esté trabajando en un hilo. Sea G_0 el hilo en el que p está trabajando, sea k el número de hilos en la cola de preparación (*ready deque*) de p, y sean $G_1, G_2, \ldots, G_k$ los hilos en la cola de preparación de p ordenados de abajo hacia arriba, de modo que G_1 es el hilo más inferior y G_k es el hilo más superior. Si tenemos $k > 0$, entonces los hilos en la cola de preparación de p satisfacen las siguientes propiedades:

1. Para $i = 1, 2, \ldots, k$, el hilo G_i es el padre de G_{i-1}.

2. Si tenemos $k > 1$, entonces para $i = 1, 2, \ldots, k-1$, el hilo G_i no ha sido trabajado desde que generó a G_{i-1}.

---

8. Escriba en español el enunciado del teorema 5 presentado en el artículo y explique con sus palabras la razón por la cual exige que la computación sea “fully strict” y ¿qué significa el espacio S1P?

Teorema 5. Para cualquier computación multihilo completamente estricta con una profundidad de pila S_1, el Algoritmo de *Work-Stealing* ejecutado en un computador con P procesadores utiliza, como máximo, $S_1 P$ de espacio.

El teorema 5 establece que para cualquier computación completamente estricta con profundidad de pila S_1, el algoritmo de Work Stealing ejecutado en un computador con P procesadores utiliza, como máximo, $S_1 P$ espacio. Esto requiere que la computación sea completamente estricta para garantizar que los hilos no dependan de otros que no sean sus ancestros directos, lo que permite un control más eficiente del espacio utilizado. El espacio $S_1 P$ significa que el espacio máximo utilizado es lineal en relación con el número de procesadores P, multiplicado por el espacio mínimo S_1 necesario para ejecutar la computación en un solo procesador.

---

9. Traducir a español los dos primeros párrafos de la sección 5 del artículo “Atomic accesses and the recycling game”.

Esta sección presenta el modelo de "acceso atómico" que utilizamos para analizar la contención durante la ejecución de una computación multihilo por el Algoritmo de *Work-Stealing*. Introducimos un juego combinatorio de "pelotas y cestas", que utilizamos para acotar la cantidad total de retraso causado por accesos aleatorios y asincrónicos en este modelo. Usaremos los resultados de esta sección en la Sección 6, donde analizamos el Algoritmo de *Work-Stealing*.

El modelo de acceso atómico es el modelo de máquina que usamos para analizar el Algoritmo de *Work-Stealing*. Suponemos que la máquina es un computador paralelo asincrónico con P procesadores, y su memoria puede ser distribuida o compartida. Nuestro análisis asume que los accesos concurrentes a la misma

estructura de datos son encolados de manera secuencial por un adversario, como en el modelo de paso de mensajes atómicos de Liu et al. [1993]. Esta suposición es más estricta que la del modelo de Karp y Zhang [1993]. Ellos suponen que si se realizan solicitudes de robo concurrentes a una cola doble (deque), en un paso de tiempo, una solicitud es satisfecha y todas las demás son denegadas. En el modelo de acceso atómico, también asumimos que una solicitud es satisfecha, pero las demás son encoladas por un adversario, en lugar de ser denegadas. Además, del conjunto de solicitudes en espera para una deque dada, el adversario puede elegir cuál es atendida y cuáles continúan esperando. La única restricción para el adversario es que si hay al menos una solicitud para una deque, entonces el adversario no puede elegir que ninguna sea atendida.

---

10. Traducir a español el segundo y tercer párrafo de la sección 6 del artículo “Analysis of the work-stealing algorithm”.

A diferencia del Algoritmo de Hojas Ocupadas (*Busy-Leaves Algorithm*), la "pila de tareas listas" en el Algoritmo de *Work-Stealing* está distribuida, por lo que no hay contención en una estructura de datos centralizada. Sin embargo, aún es posible que surja contención cuando varios ladrones intentan robar trabajo del mismo procesador víctima simultáneamente. En este caso, como indicamos en la sección anterior, hacemos la suposición conservadora de que un adversario encola de manera secuencial las solicitudes de robo de trabajo. Además, asumimos que un procesador tarda una unidad de tiempo en responder a una solicitud de robo de trabajo. Esta suposición puede relajarse sin afectar materialmente los resultados, de modo que una respuesta de robo de trabajo tome cualquier cantidad constante de tiempo.

Para analizar el tiempo de ejecución del Algoritmo de *Work-Stealing* al ejecutar una computación multihilo completamente estricta con trabajo T_1 y longitud del camino crítico T_∞ en un computador con P procesadores, usamos un argumento contable. En cada paso del algoritmo, recolectamos P dólares, uno de cada procesador. En cada paso, cada procesador coloca su dólar en uno de tres cubos según sus acciones en ese paso. Si el procesador ejecuta una instrucción en ese paso, coloca su dólar en el cubo de *WORK*. Si el procesador inicia un intento de robo en ese paso, coloca su dólar en

el cubo de *STEAL*. Y, si el procesador simplemente espera una solicitud de robo en cola en ese paso, coloca su dólar en el cubo de *WAIT*. Derivaremos el límite de tiempo de ejecución acotando el número de dólares en cada cubo.
