Multithreaded Algorithms

November 27, 2013

1 Introduction

In the commodity hardware of today, it is possible to find multiple cores per processor. An extension of these types of hardware is the Symmetric Multi-Processor (SMP) systems, which are quite popular now days in many commodity systems. Thus, it makes sense to extend our serial single threaded algorithms to be able to handle multiple threads. In addition, the centralized nature of the SMP's memories can simplify a lot the design of the multithreaded algorithms.

2 Dynamic Multithreading Programming

A common way of programming multiprocessor systems is by the use of threads, in our specific case the dynamic multithreading model. This model allows programmers to specify parallelism in applications without worrying about handling all the resources:

- Schedules
- Memory
- Etc

3 Greedy Scheduler Theorems and Corollaries

Theorem 1. *Theorem 27.1*

On an ideal parallel computer with P processors, a greedy scheduler executes a multithreaded computation with work T_1 and span T_∞ in time

$$T_P \leq \frac{T_1}{P} + T_\infty. \tag{1}$$

Proof. First, the equation (Eq. 1) correspond to two parts

- The complete steps corresponding to $\frac{T_1}{P}$.
- The incomplete steps corresponding to T_∞ i.e. the span because of the necessary serialization.

In addition we have that $T_P =$# of Complete Steps+ # of Incomplete Steps.

Thus, we have:

Case complete steps

Imagine the number of complete steps is larger than $\lfloor \frac{T_1}{P} \rfloor$. Then, the total work done by the complete steps is

$$\begin{aligned} P\left(\left\lfloor \frac{T_1}{P} \right\rfloor + 1\right) &= P\left\lfloor \frac{T_1}{P} \right\rfloor + P \\ &= T_1 - (T_1 \mod P) + P \\ &> T_1 \end{aligned}$$

This is because (Eq. 3.8) and (Inequality 3.9). This is not possible because the complete steps are making more work the total work required, contradiction. Thus , # of Complete Steps$\leq \frac{T_1}{P}$.

Case incomplete steps

Let G the DAG representing the entire computation, and assume that each strand takes unit time. Let G' be the subgraph of G that has yet to be executed before an incomplete step, and let G'' be the subgraph remaining to be executed after the incomplete step. The strands that can be run in the incomplete step are the ones that have an in-degree from other strand equal to zero. This are the ones to be executed by the greedy scheduler. Then the length of the longest critical path in G'' must be 1 less that the length on the longest path in G'. Hence, the number of incomplete steps is at most T_∞. Thus

$$T_P = \text{\# of Complete Steps} + \text{\# of Incomplete Steps} \leq \frac{T_1}{P} + T_\infty$$

□

Corollary 2. *Corollary 27.2*

The running time T_P of any multithreaded computation scheduled by a greedy scheduler on an ideal parallel computer with P processors is within a factor of 2 of optimal.

Proof. Let T_P^* be the running time produced by an optimal scheduler on a machine with P processors, and let T_1 and T_∞ be the work and span of the computation, respectively. We have, because the work and the span laws, $\max\left\{\frac{T_1}{P}, T_\infty\right\} \leq T_P^*$. Thus

$$\begin{aligned} T_P &\leq \frac{T_1}{P} + T_\infty \\ &\leq 2 \times \max\left\{\frac{T_1}{P}, T_\infty\right\} \\ &\leq 2 \times T_P^* \end{aligned}$$

□

Corollary 3. *Corollary 27.3*

Let T_P be the running time of a multithreaded computation produced by a greedy scheduler on an ideal parallel computer with P processors, and let T_1 and T_∞ be the work and span of the computation, respectively. Then, if $P \ll \frac{T_1}{T_\infty}$, we have $T_P \approx \frac{T_1}{P}$, or equivalently, a speedup of approximately P .

Proof. If we suppose that $P \ll \frac{T_1}{T_\infty}$, then we also have $T_\infty \ll \frac{T_1}{P}$, and hence we have that $T_P \leq \frac{T_1}{P} + T_\infty \approx \frac{T_1}{P}$. In addition, the work law says that $\frac{T_1}{P} \leq T_P$, we conclude that $T_P \approx \frac{T_1}{P}$ or $P \approx \frac{T_1}{T_P}$ □