Problem 1 part 2

Even Flørenæs

Fall 2016

TDT4200 Parallel Computing
Department of Computer Science

Contents

1 Theory

1.1 Caching

1.1.1 Fully associative cache

Fully associative cache is a system in which a new line can be placed at any location on the cache.

1.1.2 Direct mapped cache

A system in which each cache line has a unique location in the cache to which it will be assigned

1.1.3 N-way associative cache

Intermediate schemes in which each cache line can be placed in one of different locations in cache.

1.1.4 Indirect control

Programmers have only indirect control over the cache. It is important to remember how the cache may access data and in which order.

For example if a programmer is working on a two dimensional matrix looping through every element. The most intuitve method looping through the elements may for a programmer be access every column in row 0 and then row 1 and so on. But in the cache the memory will be stored as one large one dimensional vector and the performance of the looping program will depend on the programmers implementation related to the cache construction

1.1.5 Snooping cache coherence

In a snooping system, all caches on the bus monitor the bus to determine if they have a copy of the block of data that is requested on the bus. Every cache has a copy of the sharing status of every block of physical memory it has.

1.1.6 Directory based

A protocol which uses a data structure called directory. The directory stores the status of each cache line. Typically, this data structure is distributed: each core/memory pair might be responsible for storing the part of the structure that specifies the status of the cache lines in its local memory. When a variable is updated, the directory is consulted, and the cache controllers of the cores that have that variable's cache line in their caches are invalidated.

1.2 Gustafson's law

Gustafson presented an evaluation of Amdahl's law. Amdahl's present the problem of limited speedup by adding more and more processors as the execution time of the serial part of the program will dominate the total execution time.

$$S = \frac{s + p}{s + \frac{p}{N}} \tag{1}$$

Gustafson solves this issue by assuming that when we add more processors, we can also add more work to it. Leaving the serial part of the program dominate less for increased number of processors.

$$S = \frac{s + pN}{s + p} \tag{2}$$