**ECE4074 Advanced Computer Architecture**

**Assignment 3 Multi-processor**

**Name: Man Chun Chu ID 23715030**

**Introduction:**

In this assignment, we are going to investigate how multi-processing effect the speed of a 50 x 50 matrix multiplication.

**Assumption:**

In this simulation, we are going to test 2 processor configuration, 8kB and 32kB cache size, both are 4 ways and 4 words per block. The caches of different processors are not shared and there are no L2 caches, only DRAM are share by different processors. The DRAM specification is same as assignment 2, 72 CPU clocks for CAS, 24 for RAS.

**Method:**

In this simulation, we will test 3 different implementation methods. Each processor will calculate part of the 50x50 matrix. For one by one method, processor 0 will calculate C11, then processor 1 will calculate C12, then process 2 will take care of the next element until run out of processor, processor 0 will calculate the next one until all matrix element have been assign to a processor.

The row method and the column method are very similar; the only different is the processors for row method calculate the whole row, for the column method processors would calculate the whole column. This arrangement is aim to reuse the data in cache for matrix A or matrix B as much as it can.

The need to be access were pre calculate by function called address\_cal for each processor. The matrix\_multi just need to follow the order for each processor and calculate the time for each calculation. In the matrix\_multi function, each time a processor access an address, the time delay will be the time for calculation and address accessing. That time delay for each processor will be store to an array called time\_resume. When the time\_resume match the global clock, the processor would access the next address and calculate the delay between next accesses. Multiple processors can process at the same global clock. After all processor finish processing, the number of global clock would be the clock it takes to calculating the 50x50 matrix. There would be 252500 address needed to be access in order to complete the matrix multiplication.

**Expected result:**

In this simulation, we are calculating a 50x50 matrix. The calculation assign to different processor by location of matrix C. The task are completely independent to each other because the result only dependent on matrix A and B not the result matrix C. There is no dependent part in the calculation. According to Amdahl’s Law, the maximum speedup can be up to 1/(0+1/(number of processors)). That means in our case, 50 microprocessors can result a speedup of 50 times comparing to a single core.

However, the speed of processing would be limited by the speed of memory. Since all processors access to same DRAM and it can only service 1 core only at the same time. If multiple processors need to access DRAM, they will need to queue before DRAM is free to service. We expect that would be our major bottleneck for speedup.

**Simulation result:**

Figure 1 Number of processor VS clock cycle with different cache size and implementation method

By figure 1, we can tell in our simulation, the processing speed increase up to certain number of processor when the right method is chosen. After that, the speed of processing starts to decrease because of the number of DRAM access increase. The processing speed is not necessary increase with the number of processor. Also implementation method affect the processing speed a lot. The speed of processing increase when the size of cache increases. For one by one method, the speed of processing become very fast when the number of processors is a fraction of matrix size. That is because tasks are even disturbed to processors. The data in cache is reused in more than one calculation. It is proved when the implementation method changes to column or row, the curve become much smooth than one by one method.

Figure 2 Number of processors VS number of DRAM access with different cache size and implementation method

In figure 2, we notice that the shape of number of DRAM access graph is very simular as the processing speed graph, especially when the cache size is small. That means the speed of processing is depend on the number of DRAM access. That proves the DRAM access delay is the major bottleneck for enhancing the performance.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | 8kb column time | 8kb one by one time | 8kb row time | 32kb column time | 32kb one by one time | 32kb row time |
| Best performance: | 10 | 10 | 2 | 2 | 2 | 3 |
| Min DRAM access | 10 | 10 | 1 | 1 | 1 | 1 |

Figure 3 Summary of number of core gives the best performance and number of core gives minimum DRAM access

Figure 3 is a summary of which number of cores gives the best performance and the number of cores has minimum DRAM access. It shows the number that has minimum number for DRAM access has the best performance. In the row increment case, the best performance number of processor is 2 instead of 1 because the number of DRAM access is very close to 1 core, the advantages for multiple processing starts to appear. For larger cache size, the best performance number of cores is 2 or 3 but the minimum DRAM access is 1. That means increase the cache size help to reduces the number for DRAM access and the performance enhancement by multi-processing overcome negative effect cause by DRAM accessing.

**Potential limits of multiprocessing system and ways of improvement:**

In this simulation, we notice that there are other potential limits of multiprocessing system other than Amdahl’s law. Even in a completely independent case like a matrix multiplication. The biggest limit we can see is the delay of accessing memory. In addition, the way of sharing tasks for different processors affect the speed a lot as well.

Figure 5 Number of processor VS clock cycle with different cache size and implement method with 5 times faster DRAM

Figure 5 show the result of same method and code we used before, but the CAS and RAS set to 5 times smaller. The speed of multi-processing is faster than a single core. That indicated when the bottleneck of DRAM is minimized, the advantage of multi-processing appear.

Although row method or column method can be used to maximize the spatial locality and minimize the number of memory reading. However ether number of reading matrix A or matrix B can be minimize at the same time. That is because processors need to access same row in a matrix and all element in the other. The speedup is apply to ether matrix A or B.

Introducing a L2 cache that can be access by all processor can help to reduce the delay of accessing DRAM. In this simulation, even all processor need to access the same memory location at the same time, the DRAM can only service one core. If there are 50 processors need to access the same address, same data will need to send through the bus 50 times. The delay for the data read in increased by 50 times.

Another improvement can be made was instead of implementing row one by one for each core, task can be share in chunk. For example, for 3 processors, the first processor can calculate the first 834 element in matrix C, then the second core can calculate the second 834 element, the rest is calculated by the last core. That increases the spatial locality of cache and reduces the number of DRAM access.

We can also improve the timing of DRAM access by changing the sequence of the code. That improve the speed because different processor will intended not try to access the DRAM at the same time and reduce the queue time of memory access.

**Conclusion:**

In conclusion, multi processing helps to improve the speed of processing. However the processing speed for multi-processing not only depends on the number of cores and the dependency of codes. It is also limited by the speed of data access. The way of sharing task between different processor also affects the speed a lot.