Cores vs. Caches

CS 838

Final Project Report

By: Matt Ramsay and Chris Feucht

1. Introduction & Motivation

As technology progresses and feature size continues to shrink, it becomes increasingly easier to incorporate additional features onto an IC. This includes larger L2, or even L3, caches as well as a recent development into having more than one microprocessor core on a chip. These chip multiprocessors (CMP) can incorporate in one IC what has traditionally be implemented as a multi-chip system, all for a potential savings in space as well as cost.

With these additional possible resources, one is now faced with a new set of design decisions. Among these are how many cores to include on a chip and how much cache should be present to support it. One must also decide how to configure the cache in order to obtain optimal performance.

In this study, we analyze the performance of a 16-processor system with a wide range of core/cache ratios per chip. We vary from the extremes of having all 16 processors on one die with only 2 MB of L2 cache to having each processor on its own chip with 16 MB of L2 cache (256 MB system-wide). In our experiments, we analyze the performance and L2 cache miss rate for several different configurations that vary the number of processors per chip, on-chip L2 size, and L2 associativity. The underlying plan was to vary the number of processors and cache size such that the die would have roughly same size between configurations. We hypothesize that there is an ideal configuration that balances number of processors with L2 size and associativity for optimal performance.

To describe this issue in further the paper is broken down as follows. Section 2 describes the experiments that were proposed for this study. Section 3 encompasses the simulator environment, and Section 4 presents the results. Section 5 discusses the limitations of this study. Section 6 contains the conclusions that are drawn from the study.

2. Proposed Experiments

As stated earlier, the goal was to study the effect of changing the number of cores on a chip while adjusting the amount of cache appropriately. The experiments were all performed on a 16-processor system, allowing the number of chips to change

as the number of cores on each chip increased. With additional cores, however, one must consider the effect on the die area. If cores were added without considering the effect on area, the IC would grow rapidly with these additional units. As a result, the overall area in this study was to be approximately the same between the different core counts. The goal is to study the effect of the tradeoff of die space taken up by adding cores while reducing the space of another key resource, the L2 cache.

To this end, one must first come up with an appropriate approximation for the relative size of an out-of-order core to that of L2 cache. For the targeted technology, the assumption was made that one processor core would be approximately equivalent to 0.5 MB of cache. In addition, an assumption would be needed with regard to the amount of initial cache for a 16-processor system, and for the purposes of this study 4 MB of on-chip L2 cache was initially intended. This seemed like it may be reasonable given that the newest generation processors are using on the order of 2MB L3, and this study would encompass designs about three years out.

With these constraints in mind, the list of intended simulations, as seen in Table 1, was developed. The cache size listed would be per chip, so a system with fewer processors per IC would contain more system L2. For example, the 16 processor-per-chip configuration would have 4MB of total L2, while the 4 processors-per-chip configuration would have 10MB in each of the four ICs, totaling 40MB of system L2. The point of interest here is that while there is now more L2 on the IC and in the system with the latter experiment, you also add coherency issues between chips. This may result in additional delay due to the off-chip communication and affect the overall performance.

Associativity would also be varied to study its effects on miss rate and system performance. Changing this parameter could help to reduce the eviction of useful data, as data blocks can now be placed in additional locations. This was considered a second-order portion of the study.

With the base experiments defined, the simulator could be setup to run the tests. The description of the simulator environment as well as the final simulator configurations are discussed as follows.

Cores Per Chip	Intended		Simulated	
	L2 Size	L2 Assoc.	L2 Size	L2 Assoc.
16	4 MB	2,4,8	2 MB	2,4,8
8	8 MB	2,4,8	4 MB	2,4,8
4	10 MB	2,4,8	8 MB	2,4,8
2	11 MB	2,4,8	12 MB	3,6,12
1	11.5 MB	2,4,8	16 MB	4,8,16

Table 1: Experiment Configuration

3. Simulator Environment

To perform the tests of the various CMP configurations, Simics would be employed. In this case we would be studying web-based applications, so this simulator would offer the capability to study the simulation of multiple transactions at a time. 16-processor checkpoints for these applications were also available, so these would not need to be derived separately.

Shortly into the initial analysis of the simulator setup, it was discovered that the intended list of experiments wouldn't be possible. The simulator required the cache to have been a factor of two at some level, and the intended size of 11.5MB for the 1 processor-per-chip wouldn't fit that criteria. As a result, all of the cache sizes were adjusted to accommodate this restriction. This is reflected in the 'Simulated' column of Table 1. The maximum on-chip L2 cache size in this case was set to 16MB for one processor-per-chip, and then it was adjusted down from there to a minimum of 2MB per chip for the 16 processor case. It would no longer be possible to keep the same level of associativity between the configurations, so at 12MB of L2, the base associativity would begin to increase. Most likely caches of the larger sizes would begin to increase in associativity regardless, and this seems to fit this trend.

With the configurations selected, the benchmarks to run were then chosen. Of the available benchmarks, Apache, OLTP, and Zeus would be executed. Due to various limitations of the Condor server and time considerations, the number of performed transactions was limited to 8, 2, and 8, respectively. Despite the fact that the caches would be warmed up for these runs, it is obvious that this would not give much insight into the performance of the various experiments. The number of transactions is clearly too small, but perhaps they can indicate some trends that may hold true for

longer simulations. A discussion of the simulation results, given the above simulator environment, is as follows.

4. Results

In this section, we present and analyze the results taken from our experiments discussed earlier. Figures 1-3 graph both the performance (as measured in CPI) and L2 cache miss rates (measured in misses per thousand instructions) for Apache, OLTP, and Zeus. In each graph, the number of processors-per-chip is decreased while the L2 cache on each chip is increased to fill in the extra chip real estate.

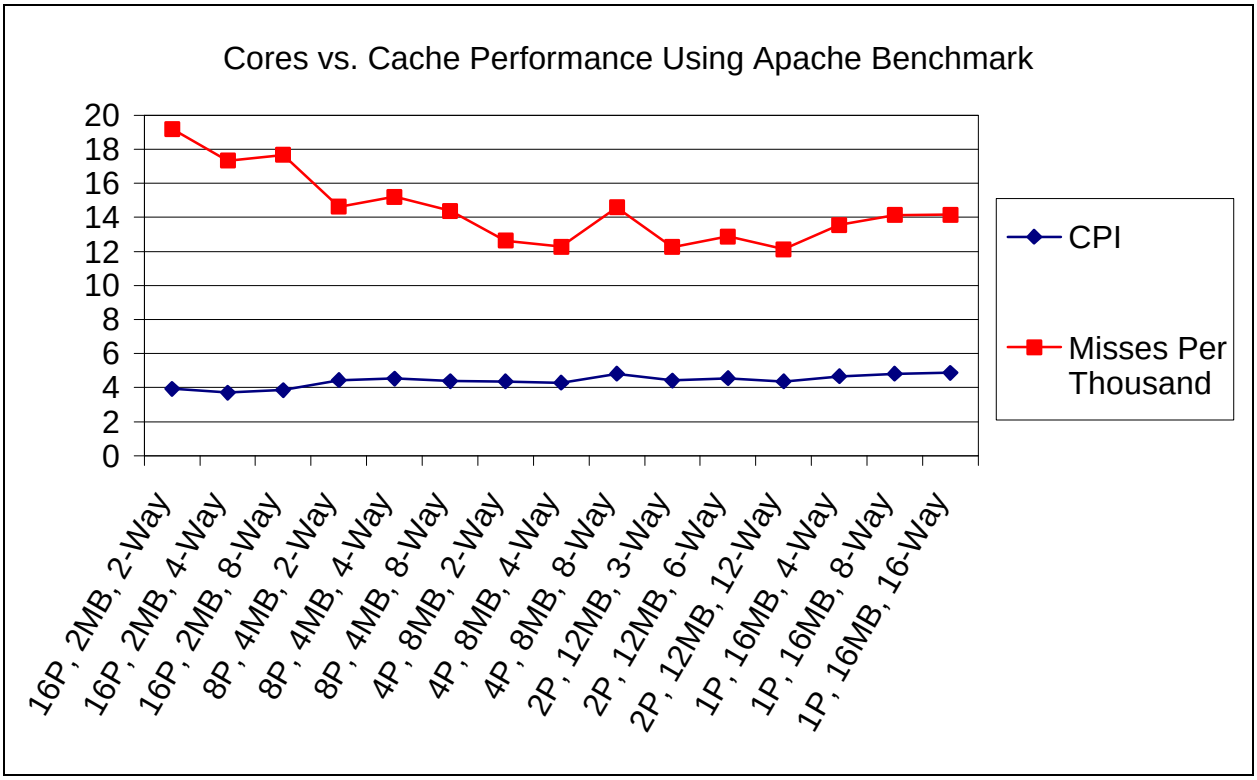

Figure 1: Apache Results

Figure 1 shows that overall performance for Apache is relatively stable across all system configurations, and in fact improves as the number of processors-per-chip is increased from 1 to 16 (right to left). The range of CPI is from 4.86 for 1 processor-per-chip (16 MB, 16-way L2) down to 3.71 for 16 processors-per-chip (2 MB, 4-way L2), an improvement of 24%. We feel that this result is both interesting and unexpected. It was surprising to learn that a system is capable of decent performance when 16 processors are sharing only 2 MB of L2 cache. Initially, we expected a system configured in this

manner to be unbalanced and would have guessed that its performance would have suffered as a result.

From Figures 2 and 3, it can be seen that similar trends hold for both OLTP and Zeus. OLTP performance ranges from an IPC of 2.71 for 1 processor-per-chip (16 MB, 4-way L2) down to 2.07 for 16 processors-per-chip (2 MB, 4-way L2), an improvement of 24%. Zeus performance ranges from an IPC of 3.99 for 8 processor-per-chip (4 MB, 2-way L2) down to 2.45 for 16 processors-per-chip (2 MB, 2-way L2), an improvement of 39%.

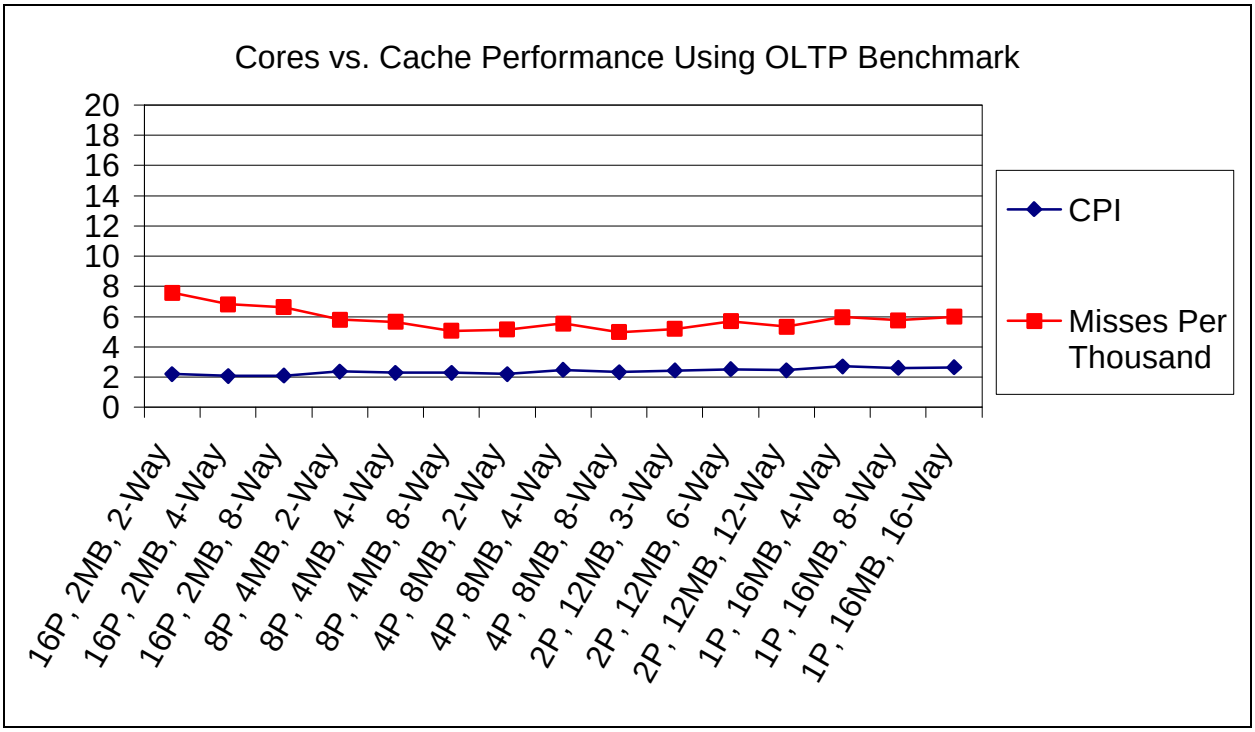

Figure 2: OLTP Results

Across all three benchmarks, the highest performing configuration had all 16 processors on one chip. As stated earlier, we found this surprising considering that all 16 processors had to share only 2 MB of L2 cache. We explain this by the fact that there are no expensive cache-to-cache transfers between L2's on different chips. When all 16 processors are on the same chip, all coherence messages are handled within the chip, thus greatly reducing latency. An additional argument for placing all processors on one die is that an off-chip L3 cache could be added to the system with no coherence complications. The addition could not be made without considerable complexity if the 16 processors were distributed across multiple chips. With an L3 cache in place, the

performance of this configuration would improve even more given that the latency of an L3 hit could probably be made less than that of a L2 cache-to-cache transfer between chips.

The other important metric that was graphed in these figures is the L2 misses per thousand. It was very interesting that while the misses were high in the 16-processor case, the performance of this case is better than the others as discussed previously. The reason for this is somewhat of a mystery at this point as the exact nature of the accounting was unable to be determined. If the miss rate doesn't account for cache-to-cache transfers, the performance characteristics displayed most likely make sense. In this case the higher miss rate for the 16-processor case would be due to capacity misses, and the miss rate for configurations with more than one chip is underestimated. Even if the miss rate did account for the cache-to-cache transfers, capacity misses could still account for the upturn in the misses as the number of processors on-chip increases. Regardless, further study of this phenomenon in conjunction with performance is needed.

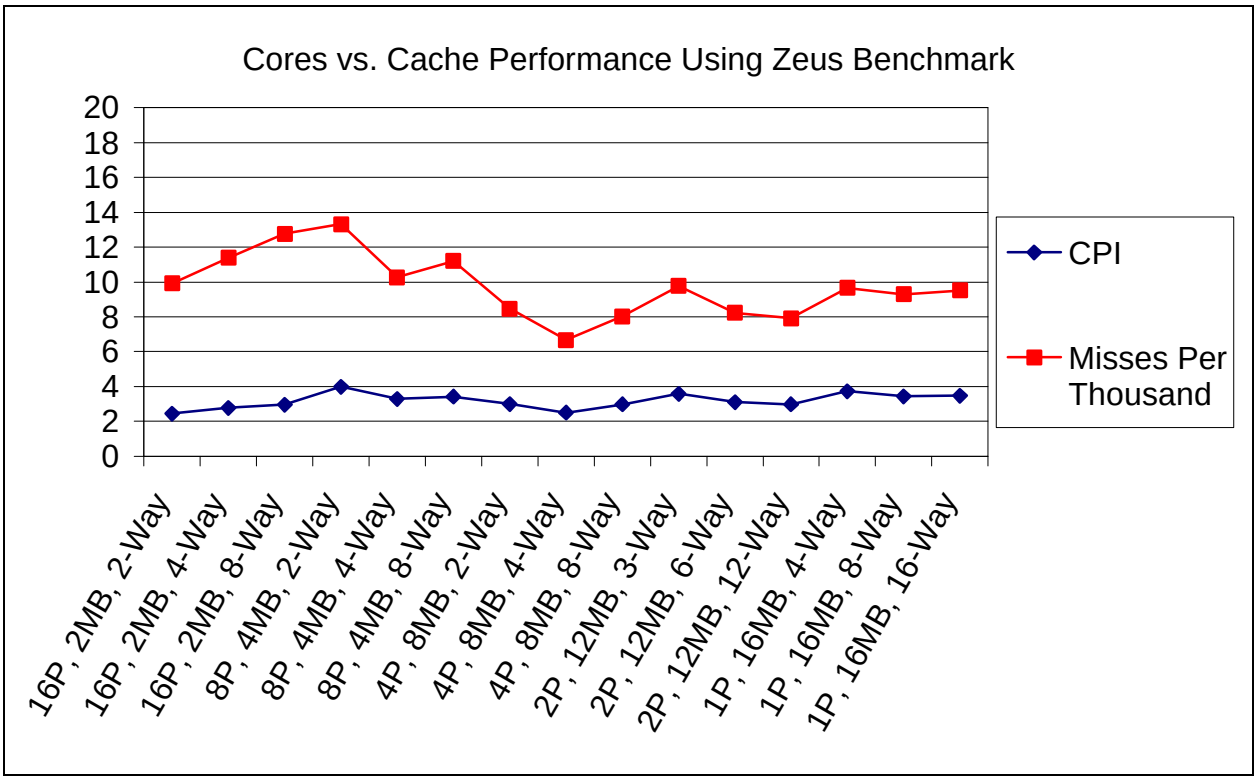

Figure 3: Zeus Results

We feel that these conclusions should be read with an asterisk. The fact that each of our experiments was conducted with a truncated simulation leads to conclusions that cannot be made with much confidence. We hope to perform these same experiments in the future with reasonable simulation lengths with the hope that the same trends hold. We then feel that the conclusions that we have drawn would be very interesting, surprising, and insightful.

5. Project Limitations

The overwhelming limitation to the depth and quality of this work was time. We were unable to become familiar enough with the simulation environment in the 5-week time frame to do some of our intended experiments. We had originally intended to investigate write-back vs. write-through caches as well as the addition of an L3 cache. These experiments were omitted, as we did not have time to change the cache coherence protocol.

We also had access to only 16-processor benchmark checkpoints. We would also have liked to study systems with a different number of total processors to see if the same trends held. It would have been interesting to see how the chips with fewer than 16 processors would have performed had they been the only chip in the system.

Our final and most frustrating limitation resulted from our ignorance with regard to the simulator's interactions with Condor. We were unaware that when Simics jobs are submitted to Condor that only the script is submitted and not the executable. Initially we submitted our 45 simulations to Condor for runs of reasonable length. Within these jobs were 15 different compilations of the Simics code to change the cache size. All of our jobs remained idle in the Condor queue for a period of time and then began to execute. When each job started, it grabbed the copy of the Simics executable currently available, not the one with which it was submitted. In the end, we ended up with 15 copies of 3 simulations. We therefore had to use the data from shorter simulations, as we did not have time to re-run the longer, more valid, simulations.

6. Conclusions

Due to the lack of validity of our simulations, few concrete conclusions can be drawn from this work. However, we did draw some conclusions based on the data that we have. Our data shows that the best configuration for a 16-processor system is to place all 16 cores on one die. In our simulations, we found that this configuration out-performed systems with all other numbers of processors-per-chip, even though it had only 2 MB of L2 cache to share among all 16 processors. We feel that this configuration could be enhanced even further with the addition of an off-chip L3 cache. This could be added with very little complexity since all coherence and consistency issues would be resolved on-chip. Obviously these conclusions need to be verified with longer simulations and further study, which we would like to do in the future.