18-741 Final Report: DBmbench

Yan Ke (yke@cs.cmu.edu)
Justin Weisz (jweisz@cs.cmu.edu)

Dec. 8, 2006

1 Introduction

Conventional database benchmarks, such as the TPC-C and TPC-H, are extremely computationally demanding, pushing databases to their limits as they simulate the operations of real applications. However, this computational intensity demands powerful, costly server hardware, limiting the availability of these benchmarks to only those who can afford such servers. For many researchers interested in exploring new architectural techniques to improve database performance, the computational demands of the TPC-C and TPC-H benchmarks are in direct conflict with the methods of simulation often used in architecture research. Simulated environments provide the ability for researchers to control and modify aspects of computer architecture not typically accessible, such as cache organization, instruction fetch and execution units. Thus, in order to perform an evaluation of a new architecture, scaled-down benchmarks are typically used. While it is assumed that these scaled-down benchmarks exhibit the same micro-architectural characteristics as their original forms, it is unclear whether or not this is the case.

Shao *et al.* [7] have proposed a framework for scaling down the TCP-C and TCP-H benchmarks, while preserving their micro-architectural characteristics. Their primary motivation was to allow these benchmarks to be used in a simulation environment, and their study concluded that their scaled-down benchmarks captured over 95% of the processor and memory performance of Decision Support System (DSS) and Online Transaction Processing (OLTP) workloads. However, their evaluation used real hardware (Pentium III), and thus it still remains unclear whether their scaled-down benchmarks are computationally tractable in a simulation environment. Further, it is also unclear if their results are an artifact of having run on the Pentium III, or if they generalize to other processor platforms.

In our work, we extend the results of Shao *et al.*, by running their scaled down benchmarks (μTPC-C and μTPC-H) in a simulation environment. Similarly, we also run the benchmarks on the AMD Opteron architecture. The experiments will allow us to investigate the validity of the micro benchmarks in modeling the full benchmarks for these architecture.

2 Related Work

Databases are at the heart of the modern computing era. They are the backbone for every important computing activity: the web, commerce, business, science and engineering. Thus, it is important to design computer architectures that are optimized for these workloads. Many of the existing benchmarks for architecture design, such as SPEC, simulate scientific, engineering, and business application workloads. These workloads are typically compute-intensive. However, there have been studies that show that Database Management Systems (DBMS) workloads are typically memory-intensive, and thus are significantly different than the SPEC workloads. Therefore, DBMS-specific benchmarks must be used to analyze architectural tradeoffs [1]. For example, Ailamaki *et al.* show that for DBMS's, 90% of the memory stalls are due to misses in the L2D and L1I. However, their results were obtained using simple database queries, rather than the intensive TPC workloads.

A common question that arises when using micro-benchmarks is whether they are representative of large workloads in real environments. Hankins *et al.* [2] studied this issue in scaling workloads from 10 to 1000 warehouse workloads, corresponding to hundreds of transactions per second. They found that the performance scales predictably and can be fitted using simple piece-wise linear models. While it is encouraging to know that performance scales predictably, their workloads are too large for architectural simulators.

Keeton and Patterson [4] presented a technique for scaling down database workloads, designed to generate the same I/O patterns as the TPC workloads. They did this by creating databases and queries designed to generate both sequential I/O and random I/O access patterns. However, they found that their sequential I/O benchmark did not simulate a representative L2 data cache behavior (compared to traditional DSS workloads), and their random I/O benchmark did not simulate a representative instruction cache and branch misprediction behavior (compared to traditional OLTP workloads). Thus, scaling down the actual TPC workloads, rather than trying to create new workloads representative of the TPC workloads, may more accurately capture these behaviors.

3 Experimental Setup

We evaluate the micro-architectural performance of the μTPC-C and μTPC-H benchmarks in two environments: on real hardware, we use the AMD Opteron platform, to contrast with Shao *et al.*'s results from the Pentium III. We also ran the full TPC benchmarks in the Simics-simulated SPARC environment to evaluate the micro benchmarks' effectiveness modeling the full benchmarks in this environment. [1]

To evaluate the Opteron platform, we we used a dual-processor, dual-core Opteron server with 4 GB of main memory. Each processor core ran at a clock rate of 1.8GHz. The Opteron is an 3-way out-of-order superscalar processor with separate 64K L1 instruction and data caches, and a unified 1MB L2 cache. To measure micro-architectural performance, we used the hardware counters provided by the processor to evaluate various aspects of process performance, such as cache miss ratios, IPC and branch prediction miss ratio. A full list of the performance counters used is detailed in Section 3.1. The Opteron server ran SUSE Linux 10.1, with kernel version 2.6.16.

The simulation environment we used was Simics, a cycle-accurate instruction-level emulator. Simics simulates a scalar SPARC processor running SunOS. It was run with separate 64K L1 instruction and data caches, and a unified 2MB L2 cache. We used the TraceUniFlex plugin, models an in-order, scalar process and gives us accurate timing information. We collect similar process performance statistics as the Opteron setup.

A breakdown of the benchmarks and architectures we evaluate is shown in Figure 3.

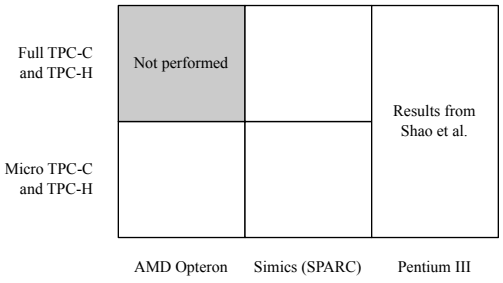


Figure 1: Experimental setup

[1] We were unable to obtain the full TPC-H benchmark for the Opteron system, and we were unsuccessful at running the full TPC-C benchmark on the Opteron.

3.1 Opteron Performance Counters

The AMD Opteron provides four hardware performance counters, and can count 79 different events. We used OProfile 0.9.1 for Linux [6] to measure the performance counters. OProfile profiles each binary, library and kernel module running on the system, and it keeps separate results for each process. Thus, we were able to ignore counters from other processes in the system, and solely focus on DB2, which performs all of it's work in a single library, `libdb2e.so`.

It is important to note that OProfile maintains a low overhead by only flushing results for each counter when a certain threshold has been reached. Each performance counter has a minimum value, representing the number of times that counter is incremented before the results are flushed to disk. For example, the ICACHE_FETCHES counter has a minimum count of 500, meaning that once this event has occurred 500 times, the ICACHE_FETCHES counter is incremented; raw counts need to be multiplied by the counter thresholds, and we perform this step in our analysis. Thus, there is a tradeoff between accuracy and performance: with small thresholds, profiling is more accurate, but takes longer, and with large thresholds, profiling is faster, but less accurate. The minimum counts we used for each performance counter we measured are shown in Table 1, and were found by running sample executions of the μTPC-C benchmark, inspecting the counters, and making sure that they counted a significant number of events, without significantly impacting performance.

To replicate the results of Shao *et al.*, we first consider their model for the breakdown of the execution time of a database query:

$$T_Q = T_C + T_M + T_B + T_R - T_{OVL} \tag{1}$$

In this equation, T_Q is the total execution time of the query; T_C is the actual computation time (in cycles); T_M is the number of cycles wasted due to misses in the cache hierarchy; T_B is the number of stalls that occur due to the branch prediction unit; T_R is the number of stalls that occur due to structural hazards such as a lack of functional units or rename registers; and T_{OVL} is the number of cycles saved by the overlap of stall time from the out-of-order execution engine.

Further, T_M, the number of cycles wasted due to misses in the cache hierarchy, is broken down in the following manner:

$$T_M = T_{L1D} + T_{L1I} + T_{L2D} + T_{L2I} + T_{DTLB} + T_{ITLB} \tag{2}$$

These represent the stalls caused by L1 cache misses (data and instruction), L2 cache misses (data and instruction), and TLB misses.

While this model was originally developed for the Pentium III processor [1], we believe it can be adapted for use with the AMD Opteron performance counters.

3.2 Database system

On the Opteron, we used IBM DB2 Express-C 9 as the underlying database management system. DB2 Express-C 9 is a free version of IBM's DB2 database product, which can be run on a server with up to 2 dual-core CPUs and 4GB of main memory. As this matches the specs of our AMD Opteron server, DB2 is able to take full advantage of the CPUs and memory of our experimental machine. Further, Ailamaki *et al.* [1] have shown that many commercial databases exhibit similar microarchitecture-level performance behavior, and thus we expect our results to be applicable to other commercial databases.

As Simics already had checkpoints for the TPC-C and TPC-H benchmarks, we used those checkpoints for our measurements. Those checkpoints used IBM DB2 UDB 8. We manually copied and installed the μTPC-C and μTPC-H benchmarks into the simulation environment.

On the Opteron platform, we each query that we ran represented a different parameter that we varied, for example the selectivity of the datasets or the number of warehouses making the queries. For the microbenchmarks, we ran each query for 20 seconds, and since each query took between 1 and 2 seconds, the results are averaged over 10-15 runs. The microbenchmark queries on Simcis are only run once because of

Measure	Description	Hardware counter	Counter threshold
T_C	computation time (cycles)	CPU_CLK_UNHALTED	1 million
		RETIRED_INSNS	100K
T_M	L1 and L2 stalls	ICACHE_MISSES	10 K
		ICACHE_FETCHES	100K
		DATA_CACHE_ACCESSES	10 K
		DATA_CACHE_MISSES	10 K
		DATA_CACHE_REFILLS_FROM_L2	1 K
		DATA_CACHE_REFILLS_FROM_SYSTEM	1 K
	DTLB stalls	L1_DTLB_MISSES_L2_DTLB_HITS	10 K
		L1_AND_L2_DTLB_MISSES	1 K
	ITLB stalls	L1_ITLB_MISSES_L2_ITLB_HITS	10 K
		L1_AND_L2_ITLB_MISSES	1 K
T_B	retired branches	RETIRED_BRANCHES	100 K
		RETIRED_BRANCHES_MISPREDICTED	1 K
T_R	structural hazard stalls	DISPATCH_STALLS	100 K

Table 1: AMD Opteron performance counters

time constraints. Each run takes from 6 to 24 hours, and therefore we are limited in the number of variations that we can do.

4 Results

We now give an overview of the results on all benchmarks and all platforms. In Section 5, we give a more in-depth analysis of the microbenchmark performance on an Opteron server.

4.1 AMD Opteron

In this section we present the results from the μTPC-C and μTPC-H benchmarks on the AMD Opteron server. We were not able to get the full TPC-C and TPC-H benchmarks running on the Opteron server.

4.1.1 μTPC-C

μTPC-C works by simulating transactions made by a number of clients to a number of warehouses. In this experiment, we varied the number of clients, but were only able to simulate a maximum of 10 clients before our machine ran out of memory and crashed (Shao *et al.* performed the same experiment with up to 200 clients). During this experiment, we measured the Instructions per Clock (IPC), branch misprediction ratio, and cache miss ratios for the L1I, L1D and L2 caches. These results are shown in Table 2.

There are two interesting trends to note. First, the L2 miss ratio increases with the number of clients because each client access different parts of the data, thereby increasing the memory footprint and reduces the hit rate on the L2 cache. Therefore, the IPC decreases with an increase in the number of clients.

4.1.2 μTPC-H

TPC-H contains two types of queries: "scan bound" queries which scan through an entire database, and "join bound" queries which collate results from multiple tables. In this experiment, we varied the selectivity (the amount of data returned in a query) for queries of both types. These results are shown in Table 3.

The L1D miss ratio decreases as selectivity increases because the processor doesn't have to do as many random accesses when more data from the table is accessed. In other words, more data from each cache

	No. of Clients		
	1	5	10
IPC	0.39	0.28	0.27
Branch misprediction ratio	0.077	0.075	0.075
L1 ICache Miss Ratio	0.018	0.015	0.016
L1 DCache Miss Ratio	0.0086	0.0080	0.0080
L2 Miss Ratio	0.28	0.38	0.36

Table 2: AMD Opteron μTPC-C Results

block will be used as selectivity increases. A small data cache miss ratio causes the IPC to go up with higher selectivity.

	Selectivity			
	1	20	50	100
IPC	0.72	0.78	0.87	0.88
Branch misprediction ratio	0.072	0.063	0.064	0.061
L1 ICache Miss Ratio	0.0031	0.0039	0.0035	0.0021
L1 DCache Miss Ratio	0.0077	0.0066	0.0054	0.0050
L2 Miss Ratio	0.15	0.14	0.11	0.14

Table 3: AMD Opteron μTPC-H Results

4.2 Simics

We performed the same experiments using the μTPC-C and μTPC-H benchmarks as we did on the Opteron servers. Instead of an x86 architecture, we are running the benchmarks on a Sparc architecture.

4.2.1 μTPC-C

The results from the μTPC-C benchmark on Simics are shown in Table 4. This is the same experiment as was performed on the AMD Opteron. The L2 miss ratio is smaller than that of the Opteron, suggesting that the SPARC cache hierarchy might be better tuned to this type of application. As a result, the IPC is larger than the on the Opteron. However, it is somewhat strange that the IPC increases with the number of warehouses, even though all of the other indicators (branch miss ratio and the cache miss ratio) suggest that IPC should go down.

	No. of Clients		
	1	5	10
IPC	0.46	0.54	0.59
Branch misprediction ratio	0.014	0.028	0.051
L1 ICache Miss Ratio	0.0017	0.0027	0.0036
L1 DCache Miss Ratio	0.017	0.017	0.019
L2 Miss Ratio	0.26	0.29	0.29

Table 4: Simics μTPC-C Results

4.2.2 μTPC-H

The results from the μTPC-H benchmark on Simics are shown in Table 5. Again, this is the same experiment as was performed on the AMD Opteron. Unlike the μTPC-C, the L2 miss ratio is much larger than on the Opteron. Consequently, the the IPC is also lower than that of the Opteron. We do not see as dramatic of increase in the IPC with larger selectivity.

	Selectivity			
	1	20	50	100
IPC	0.66	0.65	0.64	0.70
Branch misprediction ratio	0.025	0.030	0.039	0.041
L1 ICache Miss Ratio	0.0027	0.0029	0.0030	0.0021
L1 DCache Miss Ratio	0.020	0.026	0.028	0.018
L2 Miss Ratio	0.27	0.29	0.29	0.24

Table 5: Simics μTPC-H Results

4.2.3 Full TPC-X Benchmarks

We now compare the the full benchmarks to the micro benchmarks in the Simics environment, shown in Table 6. This lets us verify whether the micro benchmarks are an accurate representation of the full benchmarks. Due to time constraints, only one run of the full benchmarks were done. For the TPC-C benchmarks, it appears that the micro benchmarks do not match the results of the full benchmark. For example, the the branch misprediction ratio is different by almost an order of magnitude. This is similarly true for the L2 miss ratio. The TPC-H benchmarks fare a little better. The L1I cache and the L2 cache miss ratios were within 20% of each other. However, the IPC, branch miss prediction ratio, and the l1D cache miss ratio were off by more than 50%. Therefore, the micro benchmarks are not an accurate representation of the full benchmarks in this Simics environment.

	TPC-C	TPC-H
IPC	0.59	0.45
Branch misprediction ratio	0.10	0.042
L1 ICache Miss Ratio	0.0037	0.0044
L1 DCache Miss Ratio	0.064	0.057
L2 Miss Ratio	0.077	0.29

Table 6: Simics Full TPC-X Results

5 Analysis

We now give a more in-depth analysis of the microbenchmark performance results on the Opteron server. We report the breakdown of time spent on various parts of the CPU, and also a breakdown of stalls in the cache hierarchy. The breakdown of the total execution time is given by Equation 1. On the Opteron platform, we can directly measure T_Q and T_R using the performance counters. The branch miss penalty is 11 cycles [5], which is reasonable for a processor with a 12-stage pipeline. By multiplying this with the number of mispredicted branches, we can compute T_B, the time spent on branch mis-rediction. T_{OVL}, the number of cycles saved by overlap of stalls by the out-of-order engine, is unknown. We assume it to be small because of the database workload, which is highly dependent. The remaining term is then T_M, the memory stall delays.

5.1 Memory Access

The memory access breakdown is given by Equation 2. All of the terms can be measured using the performance counters, which gives us counts of the number of hits and misses. We also need to measure the miss penalty time, which we do using utility provided by Hennessy and Patterson [3]. The utility creates a large array and measurements the time it takes to traverse the array with different strides. This lets us deduce the access time for the memory hierarchy. The measurements are illustrated in Figure 5.1. From the chart, we see that the L1 access time is 1.5ns, the L2 access time is 10ns, and the memory access time is 160ns.

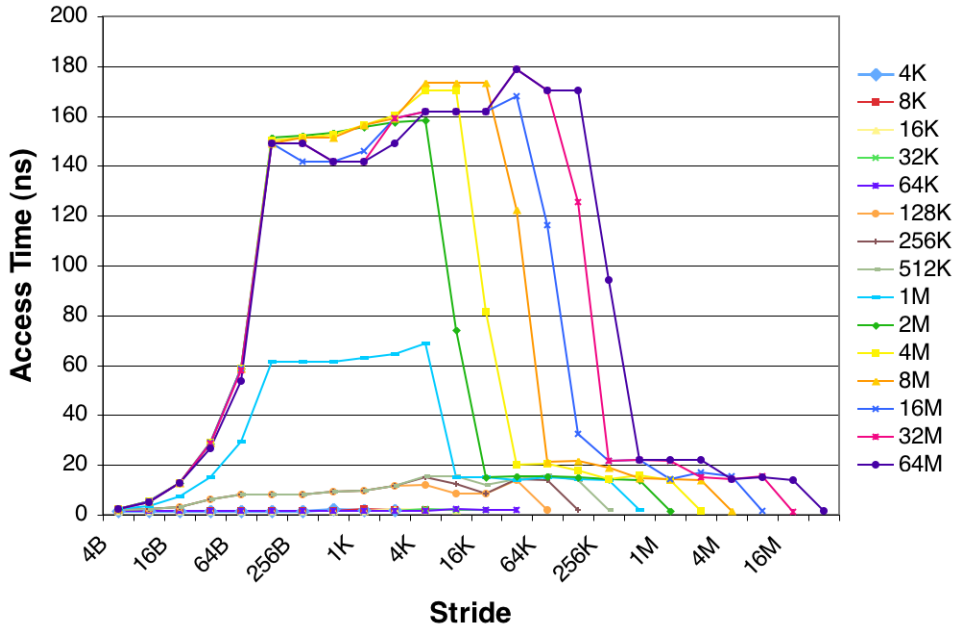


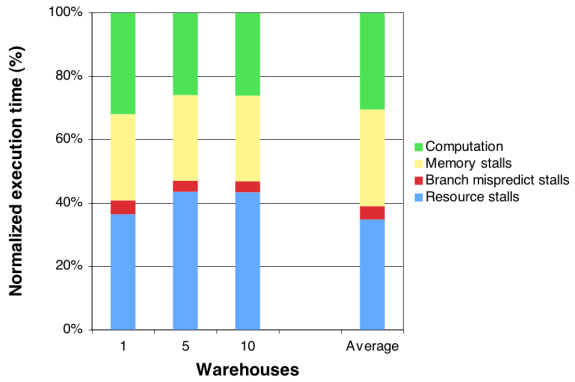
Figure 2: Access time for the Opteron memory hierarchy. From this chart, we can deduce the access times for each level of the hierarchy.

5.2 μTPC-C

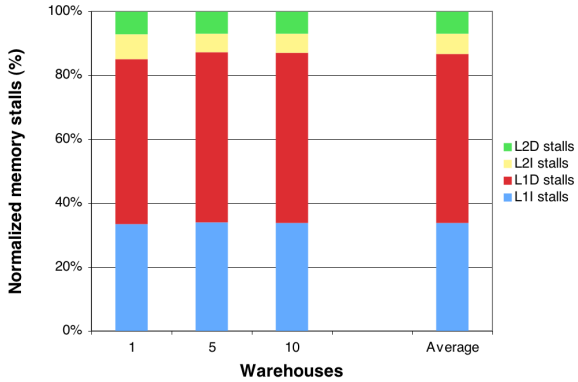
Figure 5.2 shows the results on the μTPC-C benchmarks on the Opteron server. Similar to results by Shao *et al.*, the normalized execution time only varies slightly when we vary the number of clients. However, the breakdown of the execution time in terms of computation, memory stalls, branch mispredict stalls, and resources stalls, are different than that of the Intel Pentium III architecture. In Shao *et al.*'s result, most of the time is spent on memory stalls. However, our results show that the time is evenly divided between computation, memory stalls, and resource stalls. This could be attributed to the fact that the Opteron has a faster memory subsystem. Similarly, most of the memory stalls on the Opteron are L1 stalls, as opposed to L2 stalls on the PIII, suggesting again that the memory system is faster on the Opteron.

5.3 μTPC-H

Figure 5.3 shows the μTPC-H results on the Opteron server. The results are more similar to the Intel PIII results by Shao *et al.*, however, there are still some differences. We see similar trends in that there is an increase in computation and branch misprediction ratios when going from the scan to the join operation. The branch misprediction increase is expected in that the join operation is more data dependent. However, the memory stalls account for a small fraction of the execution time, which further supports the fact that



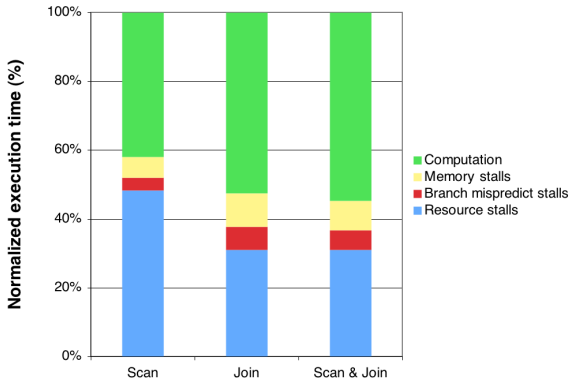
(a) Normalized total execution time



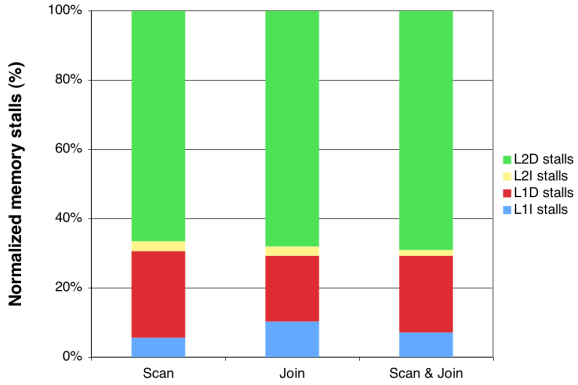
(b) Normalized memory stalls

Figure 3: Opteron μTPC-C results grouped by number of warehouses (clients).

the Opteron has a better memory subsystem. Unlike Shao *et al.*'s results, we did not see a difference in breakdown of the memory access ratios between the scan and join operation. Finally, Figures 5.3 and 5.3 shows a further breakdown of results by the join and scan operations and also by the selectivity.



(a) Normalized total execution time

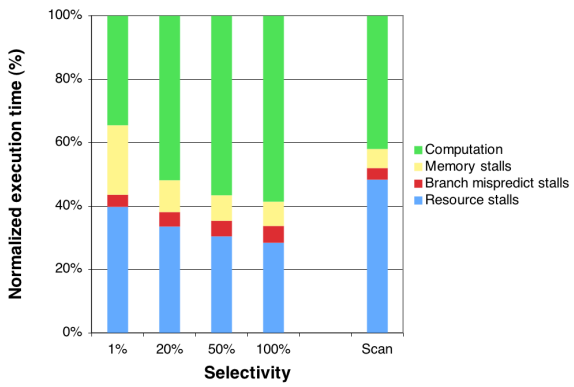


(b) Normalized memory stalls

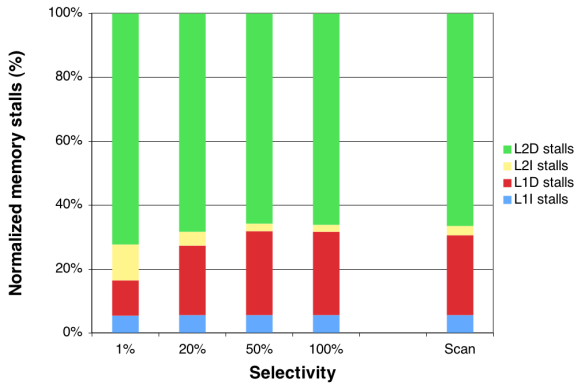
Figure 4: Opteron μTPC-H results grouped by operation.

6 Conclusion

There is a need for small, yet accurate benchmarks for computer architecture research. Shao *et al.* proposed a microbenchmark that is supposed to model the full TPC benchmarks, which they validated on the Intel Pentium III platform. We extend their work to validate the benchmark on other platforms, including AMD's Opteron and a simulated Sparc platform. Our results show that first, there are significant differences between the platforms. For example, while the benchmarks are heavily stalled on memory access on the Pentium III architecture, memory access pose less of a problem for the Opteron architecture. Second, we conclude that the microbenchmarks do not reflect the full benchmarks on the Sparc architecture, simulated on Simics.

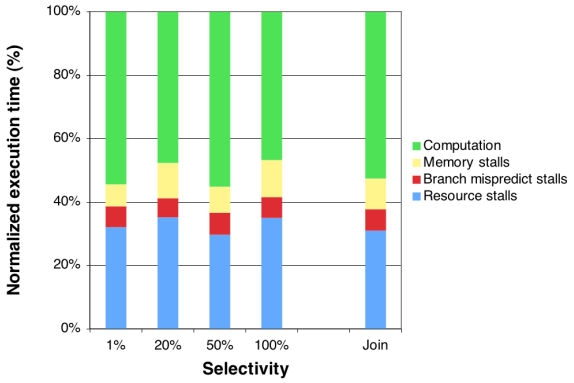


(a) Normalized total execution time

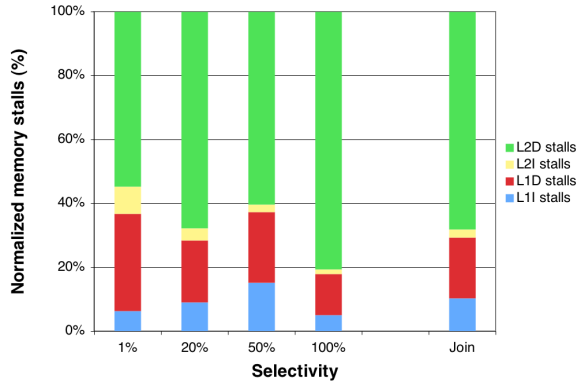


(b) Normalized memory stalls

Figure 5: Opteron μTPC-H Scan operation results grouped by selectivity.



(a) Normalized total execution time



(b) Normalized memory stalls

Figure 6: Opteron μTPC-H Join operation results grouped by selectivity.

References

[1] A. Ailamaki, D. J. DeWitt, M. D. Hill, and D. A. Wood. DBMSs on a modern processor: Where does time go? In *Proceedings of International Conference on Very Large Databases*, 1999.

[2] R. Hankins, T. Diep, M. Annavaram, B. Hirano, H. Eri, H. Nueckel, and J. Shen. Scaling and characterizing database workloads: Bridging the gap between research and practice. In *Proceedings of the International Symposium on Microarchitecture*, 2003.

[3] J. L. Hennessy and D. A. Patterson. *Computer Architecture, A Quantitative Approach*. Morgan Kaufmann, 2006.

[4] K. Keeton and D. Patterson. Towards a simplified database workload for computer architecture evaluations. *Workload Characterization for Computer System Design*, 2000.

[5] C. N. Keltcher, K. J. McGrath, A. Ahmed, and P. Conway. The AMD opteron processor for multiprocessor servers. *IEEE Micro*, 2003.

[6] Oprofile - a system profiler for linux. http://oprofile.sourceforge.net/.

[7] M. Shao, A. Ailamaki, and B. Falsafi. DBmbench: Fast and accurate database workload representation on modern microarchitecture. Technical report, Carnegie Mellon University, 2003.