Simulation Methodology for Decision Support Workloads

Luis A. Amigo, Valentín Puente, J. Angel Gregorio.
Computer Architecture Group, University of Cantabria.
{lamigo, vpuente, jagm}@atc.unican.es

Abstract

The impact of any new architectural proposal must be evaluated under realistic working conditions. This class of analysis requires trustworthy simulation tools and representative workloads that allow us to know the real effectiveness of the improvement. In this work, we propose a methodology that enables the use of an important family of transactional workloads, such as decision support system workloads, in a full system simulator. In contrast to numerical applications, with this type of workload it is not possible to scale down the problem size in order to reduce the computational requirements of the simulation. We will show the stationary behaviour of the workload and how it can be employed to reduce computational requirements without significant loss. Taking into account this fact, we will show how simulating only 3% of the benchmark the maximum error in the main system performance metrics is approximately 5%.

1. Introduction

The methodology to develop new architectural proposals requires tools able to evaluate their impact under realistic scenarios. Until a few years ago, most high performance systems had been almost exclusively used with scientific or engineering applications. For this reason, new architectural proposals for these kinds of systems were analysed employing benchmarks based on numerical workloads. However, in recent years, a significant percentage of the supercomputer market is being focused on transactional workloads [1] and therefore benchmarks must also be changed. In fact, an important effort is made by software and hardware vendors in order to establish a standardized framework for allowing customers to analyze their requirements before purchasing systems. One of the main organizations playing an important role in this trend is the Transaction Processing Performance Council (TPC) [2] proposing transactional standards such as the TPC-H benchmark, which we are focusing on. This benchmark emulates a Decision Support System (DSS) in cooperation with an Online Transaction Processing system (OLTP).

The main problem when using these new types of benchmarks in simulation tools, compared to numerical ones, is the dramatic increase of computational requirements. In order to know the impact of any architectural change, full system simulators, including the operating system, are essential. This is the only way that transactional loads can be employed for determining the performance variations against new architectural proposals. But this new scenario implies an important difficulty for the computer architect and it is necessary to reduce the complexity of the problem without altering its representative nature.

In numerical applications, this reduction is quite straightforward: scaling down both the problem size and the system hardware elements. On the contrary, working with transactional workloads is not so easy because the relationship between the workload size and the work carried out by the system is too complex. Therefore, reducing computational requirements this way can completely modify the workload characteristics and hence other alternatives must be explored.

Statistical approaches have also been proposed [11] in order to improve simulation speed. This kind of methodologies offers a general solution that could be useful on all kind of workloads and simulation platforms. Rather than use a general approach we would like to use the workload characteristics to improve performance.

In this work we will show how to reduce computational demand when a transactional workload is used as an effective benchmark. The stationary behaviour of the system throughout the workload execution allows us to reduce the necessary simulation time. This fact can be taken into account in order to estimate the impact of any architectural change in the system by just analyzing a reduced portion of the benchmark. In this way, we can considerably reduce the time requirements to study the phenomenon with a moderate loss of precision and this can be very important for exploring the design space.

The remainder of this paper is organized as follows: Section 2 summarizes the main differences between numerical and transactional workloads from the point of view of their study under simulated environments.

Proceedings of the 12th Euromicro Conference on Parallel, Distributed and Network-Based Processing (EUROMICRO-PDP'04)
1066-6192/04 $20.00 © 2004 **IEEE**

Section 3 introduces the workload considered in this study and its implementation. The framework employed is shown in Section 4. Section 5 details the behaviour of the benchmark within a real environment and Section 6 confirms the proposed methodology in a simulated environment. Finally, the main findings of this research are summarized in Section 7.

2. Numerical versus transactional workloads.

The execution pattern of numerical workloads is characterised by consecutive computation and communication phases. The inherent characteristics in both phases are quite different and consequently their demands on the *underlying* hardware are different. Under these conditions, in order to establish the final impact of an architectural change the whole benchmark must be executed. However, in order to reduce the computational requirements it is possible to scale down both the problem size and the architectural parameters of the simulated system [3]. This reduction is possible because the relationship between problem and system size and the computation and communication characteristics is easily found.

Nevertheless, the execution pattern of transactional workloads is not so clear. In this case the main phases are: connection establishment, query execution and connection termination. The first and last phases can be considered transitory with a low impact on the workload execution, but the behaviour of the central phase is complex and it is not easy to find a relationship between problem size and its demands on the architecture.

On the one hand, the amount of work of a transactional system, as well as its pattern of execution, depends on the complexity of the database employed. The same relational scheme but with different data size can cause a completely different behaviour of the system. In a DSS, queries usually include a large number of tables. Depending on the number of occurrences between these tables, the amount of work needed to carry out a query will be different. For example, in a situation where two tables must be related using a particular condition, if there is no tuple in the first table that matches this condition then a search is not necessary in the second table. Because of this, the database size must be carefully selected in order not to break the significance level of the workload.

On the other side, the database management system scheduler estimates the number of data sets to choose the most suitable query resolution method. Then, depending on the number of occurrences estimated for a particular table and its distribution, a query can be solved using sequential scanning or indexes. In other words, if the scheduler estimates that the number of occurrences that matches the query's condition is small then indexes will be used. However, if the number of estimated occurrences is high or they are physically contiguous then a sequential search will be employed.

When the load size is large enough, using indexes to resolve queries improves task performance, but it can generate locks among different processes trying to access the same data simultaneously. For this reason, if we use a reduced size of data, the scheduler always tries to use sequential methods because most occurrences would be probably contiguous. Under these conditions the behaviour can change dramatically and the significance level drops below the real conditions if we scale down database size. In consequence, it is necessary to work with a data set size large enough to assure both behaviour extremes exist.

In numerical workloads, the dependences on the operating system are practically negligible. The main influence can be motivated by the memory management subsystem. Because of this it is not usual to include the operating system effects in the analysis of architectural proposals. Consequently, tool complexity and the simulation process itself are notably simplified.

On the other hand, in transactional workloads a high dependence on the operating system appears [4]. This influence is determined by the mechanism employed to share date among the different processes and the use of these loads over the I/O subsystems. This dependency between the database management mechanisms and the host operating system makes indispensable the inclusion of the operating system effects in any computer performance analysis using transactional workloads. For this reason, a significant number of full system simulators have appeared that incorporate not only high detail hardware components but also commercial operating systems (or a slight modification of them). One of the most popular of these tools is SimOS [5], a full system simulator developed at Stanford University able to simulate MIPS, Alpha or Power PC architectures running slightly modified versions of the commercial operating systems IRIX, Digital Unix and AIX. This is the simulator we use in this work.

3. Workload description.

TPC-H [6] standard models a Decision Support System belonging to a multinational corporation. This system includes a database of customers, suppliers, orders and line items throughout 25 nations and 5 geographic regions. The DSS system is synchronized periodically with an On-Line Transaction Processing system, which receives new orders and updates existing ones. This synchronization is emulated by inserting new orders and deleting existing ones. These processes are known as refresh functions.

Over this database two tests are made whose geometric average gives us the numerical result of the benchmark.

The first test, known as the Power Test, is done by the sequential execution of the refresh functions and a stream. A stream is the name given to the sequential execution of 22 DSS queries from the same connection. The aim of this test is to obtain the best response time of the system under test.

Executing several streams concurrently with the refresh functions does the second one, known as the Throughput Test. The main objective of this test is checking the response capability of the system with heavy workloads.

Database size is proportional to a variable known as scale factor. Final size comes from the product of the scale factor and the basic size of the database. The basic size used is 1 GB of raw data. The scale factor is chosen depending on the market area of the system. Raw size increases several times when it is loaded into a relational scheme. This growth depends on the security mechanisms provided for the database, such as recovery logs, mirroring, etc.

3.1. Benchmark implementation.

In order to create a benchmark following the TPC-H standard we have chosen PostgreSQL [7] as RDBMS. PostgreSQL is a DBMS originally developed at the University of Berkeley. The main target of this project was the development of a free DBMS providing the same capabilities as a commercial one.

PostgreSQL uses a client-server scheme [8], that is, for each new client connection a new server process is created in order to answer the client's requests. Each server has its own memory area in which intermediate data is processed. As well as this dedicated area, all server processes are connected to a shared memory area whose integrity is guaranteed by semaphores. This shared area works as a cache, storing data used previously by any server. All data modifications are done in this shared area. After a modification is done, it has to be transferred to disk in order to guarantee data durability. Once durability is guaranteed, all servers may access modified data.

PostgreSQL does not allow query-level parallelism, that is, a query cannot be divided among processors. Due to this lack of parallelism, and in order to generate the biggest possible workload, during the Throughput Test we will run a stream for each processor in the system.

We will use a scale factor of 0.1 which gives us a raw database size of 100 MB. Raw size becomes approximately 500 MB when it is loaded into a relational scheme, generating indexes in order to speed up queries and providing recovery and roll-back mechanisms.

In previous works [9] we proved that this database size preserves the complexity of the standard and allows all data to fit in main memory. Fitting all data in main memory isolates the execution from the performance of disk devices. Even though this scale factor does not agree with TPC-H specifications since valid scale factors are integer values, we will use it because we want to focus our analysis on the architecture of the system. Using a scale factor not fitting data in main memory would imply that there would be a bottleneck in the I/O system. In order to avoid this problem and be able to find other bottlenecks we could use perfect disk models. Using these models would remove the impact of refresh functions and integrity mechanisms, resulting in a different execution scheme. Taking all of this into account we will use a complex disk model and a scale factor of 0.1 in order to avoid I/O bottlenecks and preserve the execution scheme.

4. Simulated environment.

4.1. SimOS Simulator.

.

SimOS provides three execution models based on MIPS architecture [10]:

- **Embra**. - This is the fastest execution model available in SimOS. Using a direct execution system, that does not simulate CPU or caches, allows users to prepare workloads and to interact with the operating system in the simulated environment. This model is able to run applications with an execution time just ten times higher than real systems. There is a modification of *embra* that provides a simple cache model.
- **Mipsy**. - This model provides a pipelined CPU like R3000/R4000 processors. In addition it provides two levels of cache memory and is able to simulate UMA and NUMA architectures ranging from 1 to 32 processors. Execution time is hundreds of times higher than the real system.
- **MXS**. - This model provides a superscalar processor similar to the MIPS R10000. It is able to simulate the same architectures that Mipsy does. At present, this model is incomplete and it only admits the R3000 instruction set. Simulation time is thousands of times higher than the real system.

Slowdowns are for one processor, as SimOS is a sequential application, simulating more than one processor increases execution times.

The simulated system includes a complete Memory Management Unit with exception handling in such detail that translation of virtual addresses into physical ones is done in the same way as the real system does.

SimOS also provides both complex and fixed latency hard disk models. The complex disk model includes a SCSI drive controller that uses DMA and interrupts in order to transfer data from/to main memory.

The simulation environment runs the IRIX operating system version 6.4 which allows the execution of actual applications but that cannot be upgraded to later versions. The old OS version with the limited instruction set lessens (but does not remove) the capacity to run current applications.

4.2. Computational systems used in this work.

In this work we have used two computational systems:

- A SGI PowerChallenge system with UMA architecture and eight 200-MHz MIPS R10000 processors, 1 GB of RAM memory, 2 MB secondary unified cache, 32 KB instruction cache and 32 KB data cache. This will be the system under test, so all real and simulated measurement will be done on it.
- A SGI Origin 3200 system with NUMAflex architecture and 2 nodes with four 400-MHz MIPS R12000 processors and 2 GB of RAM memory each node. 8 MB secondary unified cache, 32 KB instruction cache and 32 KB data cache. This system will be use as host of the simulation environment.

Both architectures can be used within the simulation environment so, once stationarity is verified, both architectures can be simulated.

5. Simulation methodology.

5.1. Considerations about simulation environment.

The simulation speed of an eight-processor system, using the Mipsy model, is about 1.8×10^5 instructions per second. Knowing that the execution of the entire benchmark needs about 1.5×10^{12} instructions, about a hundred of days would be necessary to finish the simulation using this model. If we use the MXS model, simulation would take more than three years to finish.

This time limitation forces us to develop another simulation methodology that allows us to study possible architectural improvements in a reasonable amount of time.

As has been said before, application size scaling brings about different behaviour. This different behaviour is due to the different methods used to solve queries, index usage, and especially due to the different number of occurrences between tables in the database. With all of this we are compelled to run the same workload that has proven effective in real systems.

Once we established that database workloads could not be scaled as numerical workloads are, we explored the possibility of doing a temporary scaling. To achieve this scaling we use the checkpoint and restart capability provided by SimOS. This capability allows swapping between processor models within the same simulation process.

In order to make a checkpoint, it is necessary to set a stop point somewhere in the code. Nevertheless, SimOS does not allow this checkpoint if a branch instruction is being evaluated in any of the system processors.

Temporary scaling must be done in such a way that it assures that the working point in the simulation environment is the same as in real system. Thus, it would be possible to study the effect of enhancements in the system.

As has been said, PostgreSQL does not allow intra-query parallelism so, as our working area is aimed at multiprocessor architectures, we will focus the simulation on the Throughput Test. This focus allows the Power Test's simulation to be done with a less detailed model; this gives us a total simulation time of 60 days using the Mipsy model.

The execution pattern of the Throughput Test shows a transitory zone at the beginning, due to connexion establishment, and another at the end, due to connexion termination and result storage. All the remaining execution shows a stationary behaviour pattern.

5.2. Testing for statonarity.

Next we will show the behaviour of the workload during its execution. Obviously, this measurement could not be made in the simulation environment because we need to know the behaviour of the whole execution, so it must be done in the real system. We will analyze different pieces of the execution trying to find the deviation of the principal parameters of the system with respect to the average of the execution. We will focus on instruction count per cycle and L1 data cache hit ratio.

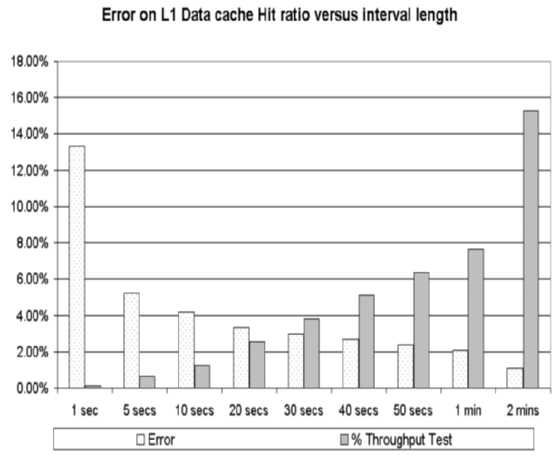

Figure 1. Error made by approximating L1 data hit ratio with the average of a time interval.

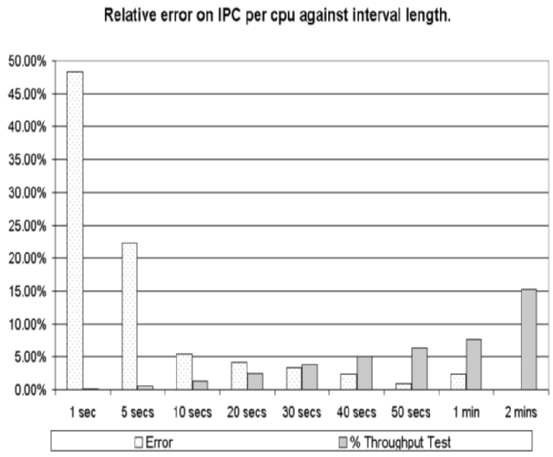

Figure 2. Error made by approximating instruction per cycle per processor.

To obtain this data series we will use the information provided by the hardware counters included in MIPS processors. This measurement will be done using a dynamic loadable module, which induces a deviation of less than 3%.

In order to test for stationarity, we will study the Throughput Test. It must be said that the safest place to introduce a checkpoint is the point where the Power Test finishes and the Throughput Test begins. Beginning detailed model execution at this point includes in the measurement the transitory effect of connection establishment and cold start misses. Thus, in Figure 1, error made by approximating L1 data hit ratio with the average of a time interval is shown. The percentage of execution time that this interval represents with respect to the execution of the whole Throughput Test is also shown.

As can be seen, data cache hit ratio shows a stationary behaviour. With 10 seconds of execution time, the error committed is less than 5%. With 2 minutes the further error made is less than 1%. Using this result we can say that the behaviour of memory hierarchy can be characterized by simulating 10 seconds with an error of less than 5%.

In Figure 2, error committed by approximating instruction per cycle per processor with the average of a time interval is shown. The percentage of execution time that this interval represents with respect to the execution of the whole Throughput Test is also shown.

As can be seen, for short intervals, there is a substantial error, but from 20 seconds and further, the error made by approximating total average by interval average is always below 5%.

Instruction per cycle count is an important indication of the behaviour of entire system. Thus any variation in the hit rate of caches, branch predictor or TLB is shown in it. So the stationary behaviour shown by IPC is a reflection of system stationarity. This stationary behaviour allows us, by simulating a piece of the whole application, to obtain the relative impact of any architectonic improvement with this workload.

If the error is desired to be lower than 5%, a simulation interval of at least 20 seconds must be used. This interval represents only 3% of the original time required.

6. Simulation results.

In this chapter we will verify that this stationary behaviour is reflected by the simulation environment. We will simulate a PowerChallenge system, identical to that used, employing a Mipsy model. We will simulate during 10 days trying to get enough information about the simulation environment's behaviour. With the data series obtained, we can observe the evolution of IPC and L1 data hit ratio.

Figure 3 shows instantaneous IPC during simulation, moreover, total average and interval average are shown.

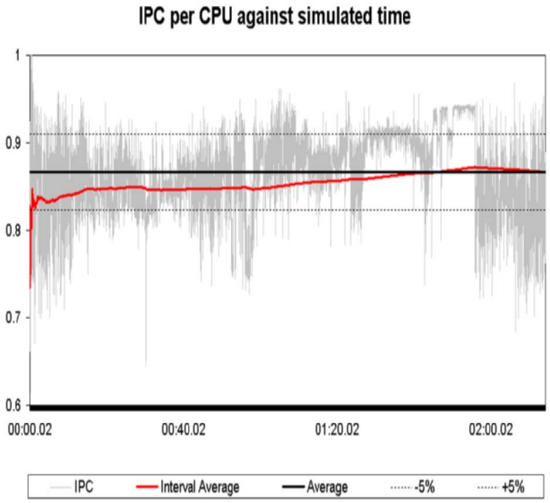

Figure 3. IPC value, average and interval average versus simulated times.

Figure 4 shows the same numbers as before but for L1 data cache hit ratio. As can be seen, as we increase interval size, the error made is constantly reducing. It has to be said that these data have been taken from a piece of the whole execution therefore the value given for total average may not be accurate. According with the real system's results, this inaccuracy is less than 5% in IPC and less than 1% in data cache hit ratio. On the other hand, data could not be compared with the real system since we are simulating a pipelined processor and real systems have superscalar ones.

Those figures agree with the estimations obtained during execution in real systems. Thus, they demonstrate the suitability of the simulation's methodology.

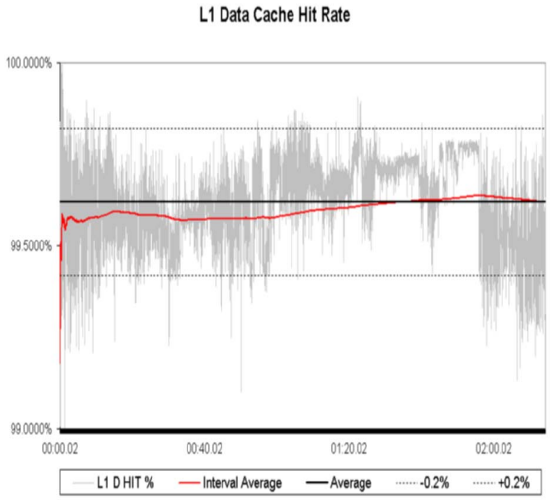

Figure 4. L1 data cache hit ratio, average and interval average versus simulated time.

7. Conclusions

The increasing use of transactional workloads in high performance computing compels us to include them in our work plan in order to check the effectiveness of new architectural proposals.

The unaffordable computation time required to execute complex transactional workloads in complete system simulators obliges us to take a less restrictive approach.

In this work we have shown the stationary nature of a TPC-H benchmark done on a database created using PostgreSQL. Using this stationary nature we are able to test relative performance of new proposals simulating a twenty-second interval. The wall clock time of this simulation, using 400 MHz MIPS R12000 processors, is less than 36 hours. Using a low detailed model allows us to get the workload ready in less than three hours for an eight-processor system. Since we are using checkpoint and restart, this workload only has to be prepared once for each system.

Taking all of this into account, we may evaluate a new proposal in only two days, for an eight-processor system, knowing that the error found would be less than five percent.

8. Acknowledgements

This work has been supported by the *"Comisión Interministerial de Ciencia y Tecnología"* (CICYT), project TIC-2001-591-C02-01.

The authors would like to express their appreciation to William Hachfeld of SGI for his help in the development of the libraries that have enabled the sampling of the hardware counters of the R10000/R12000 processors.

9. References

[1] See http://www.top500.org for further details.

[2] See http://www.tpc.org for further details.

[3] David E. Culler, Jaswinder Pal Singh. "Parallel Computer Architecture". 1999 Morgan Kaufmann Publishers, Inc.

[4] L. A. Barroso, K. Gharachorloo, and E. D. Bugnion. "Memory System Characterization of Commercial Workloads". In Proceedings of the 25th International Symposium on Computer Architecture, June 1998.

[5] M. Rosenblun, E. Bugnion, S. Devine, and S. A. Herrod. "Using the SimOS machine simulator to study complex computer systems". ACM Transactions on Modelling and Computer Simulation, Vol. 7, No. 1, January 1997, Pages 78–103.

[6] "TPC Benchmark H (Decision Support) Standard Specification Rev. 1.3.0". 1993 – 2002 Transaction Processing Performance Council.

[7] See http://www.postgresql.org for further details.
[8] "PostgreSQL Developer Version of the Docs. Internal Architecture" 1996-2003 by The PostgreSQL Global Development Group. Available on-line.
[9] L. Amigo, V. Puente, J.A. Gregorio, "*Demanda computacional del benchmark TPC-H"*. Proceedings of the *XIII Jornadas de paralelismo*, Lleida. 2002. (in Spanish)
[10] Steve Herrod, Mendel Rosenblum, Edouard Bugnion, Scott Devine, Robert Bosch, John Chapin, Kinshuk Govil, Dan Teodosiu, Emmett Witchel, and Ben Verghese, "The SimOS simulation environment". ©1996-1998 Stanford University. Available on-line at http://simos.standford.edu/userguide.
[11] Roland E. Wunderlich, Thomas F. Wenisch, Babak Falsafi, James C. Hoe. "SMARTS: Accelerating Microarchitecture Simulation via Rigorous Statistical Sampling". Proceedings of the 30th Annual International Symposium on Computer Architecture (ISCA). San Diego, California June 2003.