Exploring the Effects of Hyper-Threading on Parallel Simulation

Luciano Bononi Michele Bracuto Gabriele D'Angelo Lorenzo Donatiello
Dipartimento di Scienze dell'Informazione, Università degli Studi di Bologna
Mura Anteo Zamboni 7, 40127, Bologna, Italy
{bononi, bracuto, gdangelo, donat}@cs.unibo.it

Abstract

This paper illustrates the effects of the Hyper-Threading processor technology on the runtime performance of a parallel and distributed simulation middleware. A preliminary analysis of the middleware design and execution parameters is given to identify the tuning parameters and to evaluate the scalability of parallel simulation. A real testbed scenario has been illustrated, based on the ARTÌS parallel and distributed simulation middleware. The experimental analysis has provided some interesting guidelines about the way to adapt the parallel and distributed simulation middleware to Hyper-Threading and to increase the execution speed of the simulation.

1. Introduction

The computer simulation is a widely adopted technique to obtain a performance evaluation of complex and dynamic systems and architectures. Specifically, in many circumstances, simulation models and related simulation techniques and tools have been shown to be a very effective and adequate way to support the design, tuning and performance evaluation of complex systems [1].

At an abstract level, a simulation can be seen as a process execution managing a large set of state variables (or entities): each variable update is triggered by a simulated event. The simulation can be implemented by one (single or monolithic) process, or more than one (parallel or distributed) processes. In the case of parallel or distributed simulation, a set of Physical Execution Units (PEUs) works together to run the simulation, whose implementation is split in many Logical Processes (LPs). Each LP manages the evolution of one or more model components, it can be executed over a different PEU and interacts with other LPs by message passing [2].

A lot of research has been done to speed-up Parallel and Distributed Simulations (PADS): as an example, to reduce the overhead due to the communication and synchronization requirements, to propose specifically tailored data structures for the simulation management, and by considering the load-balancing issues of distributed computations. Many of these results are valuable and effective, but the execution speed of a simulation is also related to the CPUs and networks hardware features. It is quite obvious that a better hardware support can increase the execution speed, in a cost effective way.

The Hyper-Threading (HT) is a technology initially introduced by Intel in some Pentium 4 microprocessors and quite widespread in many commercial off-the-shelf multiprocessor execution architectures. It can improve the processor performance under some workloads, allowing multiple threads to run simultaneously.

The impact and adoption guidelines of this technology on the parallel simulation field is still not studied in deep: as an example, there is no evidence of a real speed-up of parallel simulations. Furthermore, the possible side-effects of HT on the simulation performances and scalability are not well-known. The goal of our study is to verify the impact of the Hyper-Threading on the parallel simulation, by taking as a reference testbed the parallel and distributed simulation of a typical wireless scenario. In addition, we study the effects of different design principles of the generalized simulation middleware adopted for our testbed evaluation.

The paper structure is the following: section 2 illustrates the basic Hyper-Threading technology, section 3 introduces ARTÌS as the parallel and distributed simulation framework adopted in our performance analysis. The experimental tesbed is described in section 4 and section 5 reports the results of the preliminary performance evaluation. Finally, section 6 illustrates our conclusions and future work.

This work is partially supported by MIUR FIRB funds, under the project: "Performance Evaluation of Complex Systems: Techniques, Methodologies and Tools"

2. Exploiting advanced processor features: Hyper-Threading

The Hyper-Threading technology (HT) is a processor architecture implemented by Intel in the last years. HT technology makes a single physical processor appearing as two logical processors at the user's level. As a first step towards best performances, the operating system should natively support the HT technology. In general, one physical execution resource (CPU) is shared between two logical processors. To obtain this effect, with low overheads introduced, the HT technology duplicates the high level portion of the architecture state on each logical processor, while logical processors share a subset of the physical processor execution resources. Some experimental results from Intel [3, 5] have shown an improvement of CPU resources' utilization, together with higher processing throughput, for multi-threaded applications with respect to single-threaded executions. Under optimal assumptions and conditions, the performances shown an increase near to 30%.

The HT processors are quite widespread and their diffusion is rapidly increasing both in the server and desktop market segments. Intel's roadmap considers HT as a partial step towards the multi-core technology [6]. The current Intel multi-core processors are still based on HT, but the following generations should extend this approach to support a more generalized form of simultaneous multithreading.

To the best of our knowledge, the influence of HT technology on PADS architectures and frameworks has not been investigated in detail. It is necessary to verify if a HT-enabled operating system is able to transparently and efficiently adapt the parallel simulation scheduling to the HT architecture, or if it is necessary some tuning of the execution parameters and a special support in the simulation middleware. On the application side, it is quite common for parallel and distributed simulation frameworks to allocate a single LP for each available processor. This is based on the assumption that one single LP will be the only running process and would not cause context switches and other relevant overheads. On the other hand, each time the LP is blocked due to communication and synchronization delays, the CPU time would be wasted. The effects of HT technology could significantly change the assumptions related to current implementation choices.

Under the software architecture viewpoint, most of the modern PADS middlewares are based on multi-threaded implementation, and basically should take advantage of the HT support. It would be interesting to evaluate if the increase in the number of logical processes could be exploited for PADS optimization: e.g. to concurrently run more LPs than the number of physical processors, under HT processor architectures.

3. The ARTÌS middleware

The Advanced RTI System (ARTÌS)

[7] is a middleware for PADS supporting massively populated models. The design of the middleware is inspired by the IEEE 1516 standard [4] but new features have been introduced to improve scalability and simulation speed. The PADS execution speed is highly affected by the communication performance: the approach followed by ARTÌS is adaptive and exploits the LPs physical allocation. Specifically, a couple of LPs running on the same PEU will communicate using the low latency shared memory. In the case of LPs connected by LAN, WAN or Internet, they will communicate using the R-UDP/IP or TCP/IP protocols. The protocol choice is adaptive and based on the network performances. Other important features of the middleware are the multi-threading support, the implementation of data structures specifically tailored for the event-list management, the support for the simulation cloning and concurrent replication of independent runs [1].

4. The experimental testbed

To give answers to the above questions about HT technology under the PADS assumptions, instead of relying on synthetic CPU benchmarks, we evaluated the performance of the ARTÌS simulation framework on a real experimental testbed.

First, we defined a scalable model of a complex and dynamic system, whose definition contains many of the worst model assumptions that has been identified as stressing conditions under the PADS framework optimization and simulation execution performances viewpoints: a wireless mobile ad hoc network model.

The model is composed by a high number of Simulated Mobile Hosts (SMHs), each one following a Random Mobility model (RMM) with a maximum speed of 10 m/s. This mobility model is far from being real, but it is characterized by the completely unpredictable and uncorrelated mobility pattern of SMHs. The system area is modeled as a torus-shaped bi-dimensional grid-topology, 10.000x10.000 space units. The torus area, indeed unrealistic, allows to simulate a closed system, populated by a constant number of SMHs. This assumption is commonly used by modelers to prevent non-uniform SMHs concentration in any sub-area. The simulated space is flat and open, without obstacles. The modeled communication pattern between SMHs is a constant flow of ping messages (i.e. constant bit rate), transmitted by every SMH in broadcast to all neighbors within a wireless communication range of 250 space-units.

5. The experimental results

All the experiments and the analysis results shown in this paper are based on the parallel simulation of the wireless ad hoc model (Section 4), under the control, optimization and management of the ARTÌS runtime (Section 3). We performed multiple runs for each experiment, and the confidence intervals obtained with a 95% confidence level (not shown in the figures) are lower than 5% the average value of the performance index. The experiments collected in this section have been executed over a PEU equipped by Dual Xeon Pentium IV 2800 Mhz, 3 GB RAM, Debian GNU/Linux OS Kernel version 2.6.10. The experiments have been divided in two groups: the first group is based on Hyper-Threading support enabled for the PEU (HT-ON), and the second one is based on Hyper-Threading support disabled directly by BIOS settings (HT-OFF).

The ARTÌS implementation adopted in the first part of this analysis is multi-threaded, and takes advantage of the multi-threading support to manage the execution of LPs: each LP is composed by at least 3 threads (main, shared memory and network management).

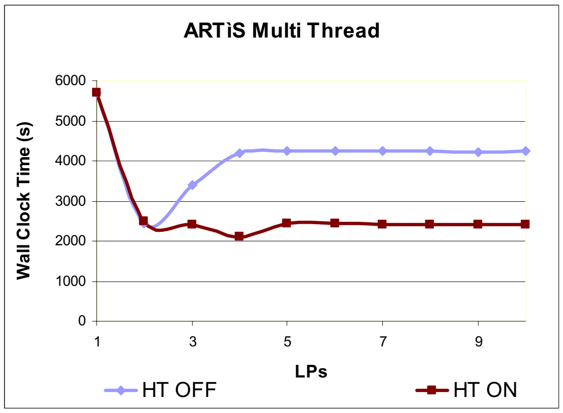

Figure 1. PEU=1, SMH=6000, ARTÌS Multi-thread implementation, Wall Clock Time with Hyper-Threading ON and OFF

Figure 1 shows the wall-clock time (WCT) required to complete one simulation run, taken as a reference. The reference run is defined as the evolution of 1000 time-steps of simulated time for the wireless ad hoc model with 6000 SMHs. The X coordinate (LPs) shows the number of concurrent LPs implementing the set of model entities for the reference scenario. When LP=1, the simulation is strictly sequential and monolithic, that is, only one processor executes the single LP that manages the execution of all the model entities of the simulated model. In the LP=2 scenario, 2 LPs manage the set of model entities, and each LP is allocated on a different physical processor of the execution architecture. When LP=[3..10] the ARTÌS framework introduces a load-sharing of model entities over LPs. In addition, ARTÌS supports communication layer adaptation, resulting in low latency shared-memory communication between LPs.

Thanks to load-sharing capability of ARTÌS, the time required to complete the simulation run (Wall Clock Time, WCT) for LP=2 is significantly lower than the one obtained with one LP (LP=1) both with HT-ON and OFF. With HT-OFF, for LP=[2..4] the WCT sharply increases, but in the following it remains constant while increasing the number of LP up to 10. This behavior is quite common in parallel and distributed simulation: increasing the number of LPs (implemented as processes and threads) induces a higher system concurrency and a consequent increase in the related overhead.

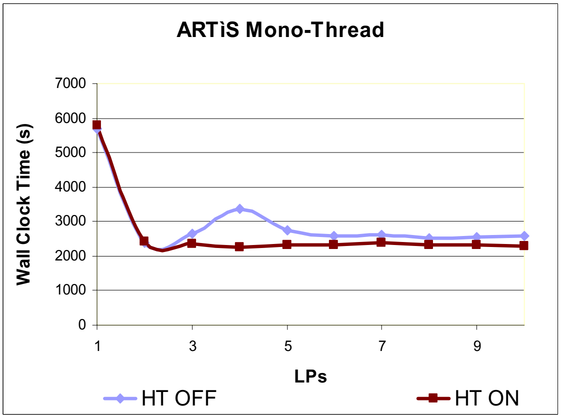

Figure 2. PEU=1, SMH=6000, ARTÌS Mono-thread implementation, Wall Clock Time with Hyper-Threading ON and OFF

With HT-ON, the best performance (lowest WCT) is obtained for LP=4. Increasing the number of LPs in the simulation the WCT is slightly worse but almost constant and always lower than simulations with HT-OFF. It is worth noting that, in this case the best performance is obtained for LP=4, that is the number of logical processors in the execution architecture, rather than LP=2 that is the number of available physical processors. Consequently, for HT-enabled processors, the policy that relates the number of LP allocated for each available physical processor should be revised.

In Figure 2, the HT support is evaluated with respect to a mono-threaded version of the ARTÌS runtime architecture. This means that the ARTÌS runtime is based on single-thread, which is responsible to manage all the model entities' executions and to manage the communications. This test is interesting to evaluate the behavior and performances of HT architectures when executing mono-threaded software.

Figure 2 shows that the best performance is obtained for LP=2 with HT-OFF and LP=4 with HT-ON, but in both cases the WCT results are very close. Turning ON the HT has the effect to stabilize the performance, while increasing the number of LPs involved in the parallel simulation. With HT-OFF, a significant simulation slowdown is shown for LP=4, and this trend is also confirmed by LP=3 and LP=5 (with a reduced intensity). The reason for this behavior requires further investigation, but it has been confirmed by many specific analysis: our feeling is that it is at least partially due to the version of the operating system scheduler used for this performance evaluation.

To summarize, in ARTÌS the HT-ON always gives better performance with respect to HT-OFF. Furthermore, the HT changes the optimal number of LPs with respect to the underlying PEU architecture, under the simulation speedup viewpoint. On the other hand, the HT performance is also influenced by the characteristics of the runtime software architecture (i.e. mono-thread/multi-thread). With a mono-thread implementation and HT-ON, the number of running LPs has a small impact on the simulation performances. Conversely, with a multi-thread implementation the obtained guideline is to match the number of LPs to the number of logical processors in the PEU. Comparing the performances of mono-thread (Figure 1) and multi-thread (Figure 2) implementations, it is worth noting that, with HT-ON, the WCT is in both cases very close to each other. On the other hand, the lack of Hyper-Threading (HT-OFF) deeply affects the performances of the multi-thread implementation. In this case, the best WCT is obtained for LP=2 (min=2450s for test run) and it almost doubles increasing the number of LPs (4250s for LP>=4). This behavior is related to the processor scalability (with respect to the number of running threads) that is reduced due to HT-OFF. In the multi-thread implementation each LP is composed by a set of threads. This can led to an increased CPU throughput if the processor is able to run multiple threads simultaneously (HT-ON). Conversely, it increases the overhead (mainly due to unnecessary context switches) if the processor is unable to manage simultaneous threads (HT-OFF or processor without HT).

6. Conclusions and future work

In this work we have shown preliminary analysis of the influence of Hyper-Threading (HT) technology on PADS architectures and frameworks. We have shown that HT could significantly change the assumptions related to some of the current implementation choices. As an example, the common policy to allocate a single LP for each available physical processor should be reconsidered with HT-enabled processors. Furthermore, we have shown that the effects of HT on the performances are also influenced by the characteristics of the runtime software architecture (i.e. mono-thread/multi-thread implementation). In the testbed evaluation the peak performance of the parallel simulation has been slightly increased by the HT. On the other hand, the HT increases the system efficiency and scalability with respect to the number of running LPs.

Future works include the analysis of other simulation scenarios, the detailed analysis of the HT-enabled operating system scheduler and the investigation of the new multi-core HT-enabled CPUs.

7. References

[1] L. Bononi, M. Bracuto, G. D'Angelo, L. Donatiello. “Scalable and Efficient Parallel and Distributed Simulation of Complex, Dynamic and Mobile Systems”. *Proc. of the IEEE FIRB-Perf Workshop on Techniques Methodologies and Tools for Performance Evaluation of Complex Systems* (Perf 2005), 2005.

[2] Fujimoto, R.M., “Parallel and Distributed Simulation Systems”, John Wiley & Sons, 2000

[3] Hyper-Threading Technology on the Intel Xeon Processor Family for Servers. http://www.intel.com/business/bss/products/hyperthreading/server/ht_server.pdf.

[4] IEEE Std 1516-2000: IEEE standard for modeling and simulation (M&S) high level architecture (HLA)

[5] D. Marr, F. Binns, D. Hill, G. Hinton, D. Koufaty, J. Miller, and M. Upton. Hyper-threading technology architecture and microarchitecture: A hypertext history. *Intel Technology Journal*, 2002.

[6] Multiple processor cores. Multitudes of processing capability. http://www.intel.com/software/multicore/.

[7] PADS: Parallel and Distributed Simulation group, Department of Computer Science, University of Bologna, Italy. http://pads.cs.unibo.it, 2006.