Performance Evaluation of Latency Tolerant Architectures*

S.S. Nemawarkar†, R. Govindarajan†, G.R. Gao‡, V.K. Agarwal†
†Department of Electrical Engineering
‡School of Computer Science
McGill University
Montreal, H3A 2A7, Canada
{shashank,govindr,gao,agarwal}@pike.ee.mcgill.ca

Abstract

In this paper, we analyze a single processor multi-threaded architecture using stochastic Timed Petri Net (STPN) model to study the effects of various parameters such as memory latency and thread runlength, on processor utilization. We first perform a simple analysis of the basic model with constant values for the parameters. This is followed by an extension with stochastic parameters. A detailed simulation study is conducted to validate the analysis. While earlier researchers established that an increase in the number of threads results in increased processor utilization, our results, on the other hand, indicate that average runlength and effective memory latency have stronger impact on processor utilization than the number of threads.

1 Introduction

The recent trend in computer architecture is towards designing multi-threaded architectures (refer e.g., [3, 4, 5, 7]). In a multi-threaded architecture, each processor executes a small sequence of instructions, called a *thread* or *context*. Instructions within a thread are executed using the von Neumann approach. However, the scheduling of threads is based on dataflow-like synchronization. Thus multi-threaded architectures combines the advantages of von Neumann computation model with data-driven evaluation. Further, multi-threaded architectures mask memory latency. This is accomplished by suspending the execution of the current thread and switching to another thread. Thus, in a multi-threaded architecture the processor is kept busy while the memory request is being served. This improves the processor utilization.

The performance of a multi-threaded architecture depends on a number of factors like memory latency, context switch time, number of threads, and the runlength of a thread. In this paper we evaluate the performance of a single processor multi-threaded architecture using Stochastic Timed Petri Net (STPN) model. The main contributions of this paper are: (i) identifying the impact of memory; (ii) analyzing the effect of the partitioning strategy; (iii) establishing the relationship between runlength and throughput of memory so as to achieve high processor utilization.

In the following section, we develop the analytical model. Section 3 and Section 4 deal with the analysis of the deterministic and the stochastic model respectively. In Section 5, we compare our work with other related works.

2 The Model

2.1 The Architecture

The architecture consists of a processor and a memory subsystem. The processor architecture is pipelined to achieve high throughput. The memory subsystem either has interleaved organization or has multiple ports to improve its throughput.

Terminology

- A *thread* is a sequence of instructions. A thread could be in one of the following *states*:

  Ready — when all the input dependencies of the thread are satisfied.

  Executing — when a *ready* thread is scheduled for execution on the processor.

  Suspended — when the thread is awaiting a response to a long latency operation.

  Terminated — when all the instructions in the thread are executed.

- *Runlength* R is the duration for which a thread executes on the processor before getting suspended or terminated.

*This work has been supported by MICRONET – Network Centers of Excellence

- *Memory Latency* L is the duration between the issue of a memory request and the arrival of its response.

- The number of memory ports and the number of threads in the system are denoted by n_p and n_t respectively.

- *Effective Latency* L_{eff}: Each individual thread experiences an average memory latency L. The memory system is capable of serving multiple requests concurrently, thus reducing the response time per request. The reduced response time will be referred to as the effective latency L_{eff}.
If there are n_t threads in the system and n_p memory ports and if $n_t > n_p$, then only n_p requests at most can be served at any time. Thus, the effective response time will be L/n_p. Otherwise, $L_{eff} = L/n_t$. Therefore,
$$L_{eff} = \frac{L}{min\ \{\ n_p,\ n_t\}}$$

- *Processor utilization* U is the fraction of time the processor is performing useful tasks.

- *Context Switch Time* C is the time required to save the state of a *suspended* thread and restore that of a newly scheduled thread.

2.2 The Analytical Model

The multithreaded architecture can be modeled using STPN as shown in Fig. 1. Initially there are a number of ready threads n_t in the ready pool, represented by the place $p4$. The processor fetches a thread from the pool and executes the instructions in the thread. This transition may take a time equal to the runlength R of the thread. When the processor encounters a long latency operation, it saves the states of the current thread. This is represented by the transition $t2$ in Fig. 1. The processor issues a request to the memory system (in order to satisfy the long latency operation) and is ready to execute another thread from the ready pool. This is modeled by the loop involving transitions $t0$, $t1$, and $t2$, and the places $p0$, $p1$, and $p2$.

The memory system services the request issued by the processor. The memory system consists of a fixed number of ports n_p modeled by a fixed number of initial tokens in the place $p6$. The service time of the memory L, is the transition time for $t3$. Once the request has been serviced by the memory, it joins the pool of ready threads so that it could be executed by the processor subsequently. In Fig. 1, $p1$, $p2$, and $p5$ are dummy places and do not signify anything in the architecture. A transition represented as a rectangular box indicates a fixed delay time for the transition. A single line represents a transition with zero delay.

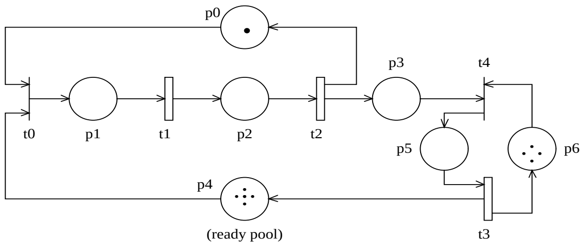

Figure 1: Stochastic Petri Net Model

3 Deterministic Model

For the STPN model of the multi-threaded architecture described above, the processor utilization can be calculated using the runlength, the memory latency, and the context switch time. For the purpose of this analysis, we consider these values to be constant. We relax these assumptions in Section 4. A token in the transitions $T1$ and $T3$ corresponds to a thread in the states, *executing* and *suspended*, respectively. The time for a token to go through a complete cycle of *executing* $\rightarrow$ *suspended* $\rightarrow$ *executing* is $R + L + C$.

At the start of execution on the system under evaluation there would be n_t threads in the ready pool, each having a runlength R. To evaluate the performance of the system, we must consider the system when it is in the *steady state*. In the steady state, the effectiveness of the architecture in masking memory latency is determined by the relation between L and $(n_t - 1) \times (R + C)$. There are four distinct cases and the processor utilization in each of these cases is:[1]

Case 1 : $n_t = 1$.
$$U = \frac{R}{R + L}$$

Case 2: $n_t \leq n_p$ and $R \times (n_t - 1) < L$.
$$U = \frac{R \times n_t}{L + R + C}$$

Case 3: $n_t \geq n_p$ and $R \times n_p < L$.
$$U = \frac{R \times n_p}{L + R + C}$$

From (2) and (3), we observe that there is no performance gain by increasing the number of threads beyond the limit $(n_t = n_p)$.

Case 4: $[R \times min\{n_t - 1, n_p\}] \geq L$ and $n_t > 1$.
$$U = \frac{R}{R + C}$$

[1]For lack of space, the details of how the equations are derived are omitted in this paper.

We observe that cases (2), and (3) arise when $R \approx L_{eff}$. Beyond this point, high processor utilization is achieved, as depicted in (4). The processor utilization increases linearly with an increase, either in R or in n_t. This happens until the maximum memory capacity is used, i.e. $n_t = n_p$.

3.1 Discussions

Earlier works on performance analysis of multi-threaded architectures [6, 1, 2] have overlooked the effect of memory. These results suggest that the use of large number of threads would mask the long memory latency. This increase with respect to the number of threads is possible only if all requests sent to the memory are served with a constant latency and there is no queuing delay at the memory. Thus a large number of ports is implicitly assumed in this model. In reality such a memory system is difficult to implement. Thus, achieving high processor utilization using large number of threads is not realistic.

4 The Stochastic Model

In Section 3, we analyzed the performance of the multi-threaded architecture model with fixed values for the input parameters, namely *runlength, memory latency*, and *number of threads of program*. Here we relax this restriction and study the effects of the variation in each of these parameters on processor utilization. We assume that the runlength, R, has an exponential distribution with a mean value of R_{avg}. We analyze the resulting STPN model using simulation. Other parameters are also varied in the simulation. All the experiments have been performed for 100,000 time units. We will notice that the results of the simulation compare favorably with those of Section 3.

4.1 Simulation Results

The results of our simulation study are shown in Fig.2-4. The following interesting observations can be made from these results.

(1) The utilization of the processor increases with an increase in the average runlength of a thread as shown in Fig. 2.

(2) The utilization saturates beyond certain value for the runlength. This is true irrespective of the variations in number of threads or number of memory ports.

(3) It can be observed from Fig. 2 that there exists a duality between the number of memory ports, and the number of threads. This is in fact predicted by the cases (2) and (3) of the Section 3.

(4) Processor utilization is relatively insensitive to initial increase in memory latency value. However, beyond certain average latency the utilization decreases

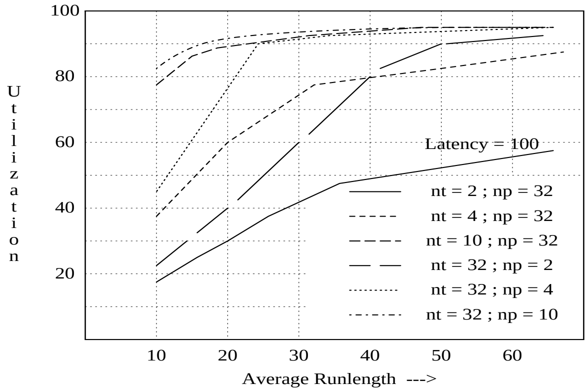

Figure 2: Processor Utilization vs. Runlength

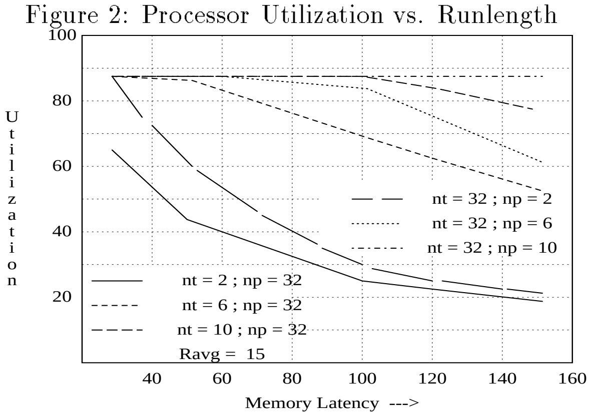

Figure 3: Processor Utilization vs. Memory Latency

with an increase in memory latency. More precisely, whenever $L_{eff} > R_{avg}$, the processor utilization becomes sensitive to an increase in memory latency.

(5) Fig. 4 depicts the effect of runlength on utilization for various L_{eff} values. We observe that the processor utilization saturates whenever R_{avg} overtakes L_{eff}.

5 Discussions and Related Work

5.1 Discussions

The performance study is useful in providing certain guidelines to partioning the programs into multiple threads and to make effective use of the resources in the multi-threaded architecture. We list some of these observations below.

(1) In order to make good use of the memory system, the number of threads (n_t) in the program should be greater than the number of memory ports (n_p). Otherwise, the memory ports will be under-utilized.

(2) To achieve high processor utilization, the average runlength R_{avg} of a thread should be greater than effective memory latency (L_{eff}).

(3) As R_{avg} is a measure of granularity, fine-grain parallelism can be exploited by making R_{avg} small. How-

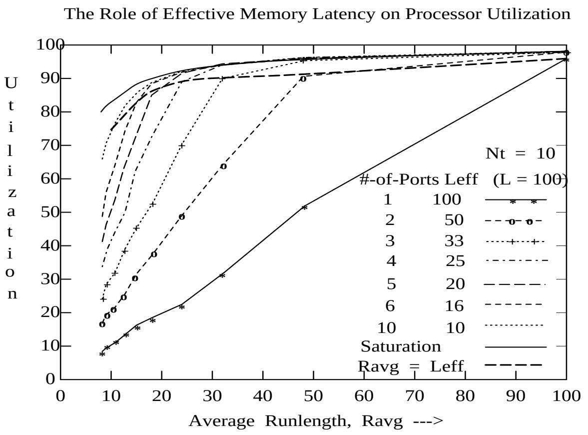

Figure 4: The role of L_{eff} on Processor Utilization

ever for small R_{avg} values, C may dominate the processor utilization.
(4) Low values of effective memory latency give better processor utilization. This can be achieved by a low L and/or large n_p.

5.2 Related Work

There have been a number of research works reported on the performance evaluation of multi-threaded architecture using analytical methods [6, 2, 1, 8]. These results indicate (i) increasing the number of threads, (ii) increasing the size of thread cache, (iii) increasing the network bandwidth, as possible ways of improving the utilization of a multi-threaded architecture. The results of [8] suggest that two to four threads per processor is sufficient to achieve good utilization.

While other researchers have neglected the effect of memory system, we show that it has a significant impact on processor utilization. Further, we demonstrate that the throughput of the memory system puts a limitation on the maximum achievable execution rate. This is irrespective of the availability of a large number of threads to mask the long latency. Also, we show that high utilization can be achieved by increasing the runlength of threads.

6 Conclusions

In this paper we evaluate the performance of multi-threaded architectures using analytical methods. A stochastic timed Petri net model of multi-threaded architectures has been developed and analysed using simulation techniques. From the results and the analysis, we draw the following conclusions:
(1) the memory system has a significant impact on the processor utilization.
(2) the *effective memory latency* captures the notion of utilization of memory.
(3) maintaining a average thread runlength greater than the minimum response time of the memory yields high processor utilization.
(4) increasing the number of threads does not increase the processor utilization, if all the memory ports are busy.
Finally, we believe that these results can serve as guidelines for a thread-partitioning scheme to extract threads with specific characteristics to achieve high processor utilization.

References

[1] A. Agarwal. Performance tradeoffs in multi-threaded processors. MIT VLSI Memo 89-566, Lab. for Computer Science, MIT, September 1989.

[2] L. Alkalaj and R.V. Bopanna. Performance of multi-threaded execution in a shared-memory multiprocessor. In *Proc. of 3rd Ann. IEEE Symp. on Parallel and Distributed Processing*, pages 330–333, Dallas, USA, December 1991. IEEE.

[3] R. Alverson and et al. The Tera computer system. In *Proc. of the 1990 Int. Conf. on Supercomputing*, pages 1–6, Amsterdam, Netherlands, June 1990. ACM.

[4] R.A. Iannucci. Towards a dataflow/von neumann hybrid architecture. In *Proc. of 15th Ann. Int. Symp. on Computer Architecture*, pages 131–140. ACM, June 1988.

[5] G.M. Papadopoulos and D.E. Culler. Monsoon: An explicit token-store architecture. In *Proc. of the 17th Ann. Int. Symp. of Computer Architecture*, pages 82–91, 1990.

[6] R.H. Saavedra-Barrera, D.E. Culler, and Thorsten von Eicken. Analysis of multithreaded architectures for parallel computing. In *Proc. of 2nd Ann. ACM Symp. on Parallel Algorithms and Architectures*, Crete, Greece, July 1990. ACM.

[7] B. Smith. The architecture of HEP. In J.S. Kowalik, editor, *Parallel MIMD Computation: HEP Supercomputer and its Application*, pages 41–55. The MIT Press, 1985.

[8] W.D. Weber and A. Gupta. Exploring the benefits of multiple contexts in a multiprocessor architecture: Preliminary results. In *Proc. of the 16th Ann. Int. Symp. on Computer Architecture*, pages 273–280. ACM, 1989.