Architectural Considerations
for
Network Processor Design

EE 382C Embedded Software Systems

Prof. Evans

Department of Electrical and Computer Engineering

The University of Texas at Austin

David N. Armstrong

8 May 2002

ABSTRACT

Network processors are processors tailored towards the computer network space and generally perform packet-processing operations. Network processors address the need for performance in computer network design while maintaining the flexibility to adapt to future network protocols. This project examines the performance and flexibility of network processor architectures. The architectures examined are chip-multiprocessor architectures—where parallelism in the workload is exploited through one or more on-chip processing cores. Network processor architectures are modeled using discrete event simulation. An abstract workload which bears similarity to current network processor workloads such as IP forwarding, Network Address Translation and Encryption is used. Simulations are run which compare the running time of a workload vs. network processor architectures with different numbers of on-chip processing cores. This project demonstrates the benefit of parallel architectures on the network processor workload. Parallel architectures with 16 processing cores can achieve up to 90% reduction in execution time over single processor architectures. Simulations are run which compare the running time of different workloads with a varying number of dependencies between packets. Processor utilization is also measured. The limits of performance gain from different network processor architectures and network processor workloads are examined.

I. INTRODUCTION

A network processor (NP) is a computer processor tailored towards use in the computer network space. The NP solutions studied in this project reside in the data plane of computer networks and perform operations such as packet forwarding, network address translation, and encryption. Because NPs are programmable, they offer the flexibility to execute a variety of different complex algorithms, possibly simultaneously, and they achieve performance comparable to their ASIC counterparts.

This project is primarily concerned with the trade off between flexibility and performance in NP design. The objective of this project is to evaluate the performance of different NP architectures as a function of variations in the workload. A discrete event simulator, which analytically models different NP architectures, was built. Although, the workload run on this simulator resembles workloads of current NPs, the workload is abstracted to reflect the idea that NPs should not be evaluated using a limited set of algorithms. The abstracted workload allows direct control over various properties of the workload including the number of packet dependencies, arrival rate and packet processing overhead.

This paper is organized as follows. Section II describes the context of this research with a description of modern NPs and algorithms that run on NPs. Section III addresses the project implementation and the simulator that was used. Section IV describes the experiments and presents the results and analysis. Section V describes future work and concludes.

II. NETWORK PROCESSOR BACKGROUND

A. Motivation

In current computer networks, the transmission links between network nodes are generally not the performance bottleneck in the system. Instead, the bottleneck tends to lie at the

nodes of the computer network, specifically, the processing that goes on at the network nodes, such as packet forwarding, packet classification, network address translation or encryption [9].

In the past, a response to this performance bottleneck has been a trend towards using dedicated hardware, or ASICs, to perform the processing at network nodes, instead of software running on general purpose processors. However, the ASIC based solution carries with it a number of disadvantages including a longer time-to-market and a higher development cost. The disadvantages of ASIC based solutions and the desire to perform a more complex set of operations at the network nodes have motivated the development of NPs. NPs can run software that performs the same operations as the ASIC and general purpose processor solutions. NP architectures are optimized for a packet processing workload, which can yield comparable performance to previous solutions [1, 2, 4, 5]. An NP solution offers the additional advantage that it can be reprogrammed after it is deployed in a network in order to accommodate upgrades, protocol changes or bug fixes. In the best case, this leads to a longer time-in-market for NP solutions over their ASIC counterparts [4].

B. Modern Network Processors

Some examples of modern NPs include Agere Systems' NP10 and TM10, IBM's Rainier, Intel's IXP1200, and Motorola's C-5e. The architectural characteristics of these NPs include multiple, fast I/O ports, fast local buffer memory, and mechanisms to exploit parallelism in the workload, such as multithreading [2, 4, 5, 10]. Bux *et al.* note that "today's high-end NPs employ multiple (e.g., 16) multithreaded processor cores clustered into one processor complex" [1]. The Intel IXP1200 uses six micro engines on the same chip [5]. The Agere TM10 uses a similar scheme with a single chip as a "traffic manager" and multiple programmable engines in addition [10].

C. The Network Processor Workload

NPs are intended to run a variety of different algorithms. It is difficult therefore to identify a single algorithm that all current NPs run. However, it is possible to identify common characteristics of the workload, such as packet processing for example, which yields *packet-level* parallelism [7]. For the experiments conducted in this project, a workload is used which is an abstraction of real algorithms that are run on NPs. The algorithms that form the basis of this abstracted workload include data plane algorithms such as IP packet forwarding, packet classification, encryption and authentication, statistics gathering, network address translation and congestion management. Network address translation and encryption are particularly representative and are described below. The reader is referred to the references for further details regarding these and other algorithms that might be run on NPs.

Network address translation is a mechanism proposed as a temporary solution to the Internet's IP address depletion problem [3]. Using network address translation at a stub in the network allows a single "external" IP address to represent multiple "internal" hosts. An NP connects the internal nodes with the external Internet and maintains tables that keep track of open connections between external addresses and internal nodes. The NP manipulates packet addresses such that packets are directed to their proper destinations despite only the single external IP address. The address depletion problem is alleviated since for all of the internal hosts, only the single external IP address must be unique on the Internet.

The relevant characteristic of network address translation is that it maintains network state which can introduce packet dependencies. Packet dependencies reduce the amount of packet-level parallelism by forcing the serialization of packets through the NP. Take for instance the case where the state of a connection between an internal node and an external IP address

changes, perhaps ends. The state tables must be updated before any other packets belonging to the same source or destination can be processed.

Data encryption algorithms, such as DES, RC4 or AES, use a "key" to rearrange the bits of message data in such a manner that the message is readable only by a recipient also in possession of the "key." Often the value of the key for one packet is a function of the value of the key used on the previous packet belonging to the same source and destination pair. This aspect of encryption algorithms is similar to network address translation in that it introduces dependencies between packets. Another important characteristic of encryption algorithms is that the latency to perform the encryption varies with the length of the packet. Encrypting a single block of data using RC4, for example, involves manipulating bit positions and the XOR function; it can be modeled with SDF.

III. The Model

This project uses a discrete event simulator to model different architectures for NPs and their workloads. The simulator is written in C++ and models parallel architectures in the chip-multiprocessor sense, where multiple copies of a single processor are repeated on one chip [8]. This type of architecture is motivated by modern NP designs such as the Intel IXP1200 and Agere NP10 and TM10. The simulator models an abstract packet workload for two reasons. First, the results obtained by using this workload are not tied to any particular algorithm, but rather algorithms that display certain characteristics, such as inter-packet dependencies. Second, an abstract workload allows direct control over the properties of the workload such as the packet arrival rate, the average processing time per packet, and the number of packet flows. A packet flow is a source and destination pair where packets are dependent on their predecessor and must be processed in serial within a packet flow. The simulator processes events on the granularity of

a simulated machine cycle. The processing cores are modeled as an array of objects, which can either process a packet or remain idle. Packets are modeled as objects with an associated processing latency and packet flow identification. Packets are scheduled onto an available core when a core is idle and the packet is not waiting on any previous packets to finish. The scheduler is not speculative and conservatively waits to schedule a packet until it knows that all packet dependencies are resolved.

IV. Results and Analysis

The objective of this project is to study the flexibility of NP architectures by observing their performance as their workload is perturbed. The following experiments simulate NP architectures running workloads with a varying degree of packet dependencies. Execution time and processor utilization for the experiments are presented in Figures 1 through 4.

Figure 1 shows execution time vs. the number of packet flows for different NP architectures. A single packet flow indicates a highly dependent workload. As the number of packet flows increases, packets are more likely to be independent of one another and therefore the workload exhibits more parallelism. The different curves on the graph represent different NP architectures where the number of processing cores is varied from 1 to 16.

Figure 1 shows that the effectiveness of additional processing cores is related to the number of dependencies, or packet flows, in the workload. For example, on a workload with a single flow, where almost all the packets are dependent upon one another, the difference in running time between a single core and a 16-core NP is approximately 10%. Whereas, with a highly independent workload, one with 512 different packet flows, the difference in running time between a single core and a 16-core NP is approximately 90%.

The benefit of adding more processing cores is significantly greater when the number of dependencies between packets is reduced. In fact, the limit to how much each additional processing core can improve performance is shown by the plateau of each curve in Figure 1. Each curve plateaus at the point where the full amount of parallelism that that architecture is capable of exploiting is saturated. Notice that the point where each curve plateaus is at a lower and lower execution time as the number of processing cores increases. This is because additional cores allow that NP architecture to exploit more parallelism. Effectively this plot shows the point at which a designer must add another processor core to the NP in order to achieve additional performance.

Figure 2 shows the performance of an NP system as a function of the number of processing cores in the NP. Whereas Figure 1 showed the maximum amount of parallelism that can be exploited by different network processor architectures, Figure 2 shows the maximum amount of parallelism exhibited by the different workloads. That is, Figure 2 shows the point at which adding one more core to the processor, for a given workload, is no longer effective. Again, this occurs at the point where each curve plateaus. For any workload, a portion of the workload can be executed in parallel and a portion of the workload must be executed serially. Another processor is no longer effective whenever the current number of processors is sufficient to handle the entire portion of the workload that can be processed in parallel. A designer would use Figures 1 and 2 to determine the appropriate number of processors to use in order to achieve the maximum performance, given an anticipated workload.

Figures 3 and 4 show the processor utilization of an NP system vs. the number of packet flows and the number of processor cores. Processor utilization is the percentage of cycles that any processing core is doing useful work. The processor utilization is an indication of how

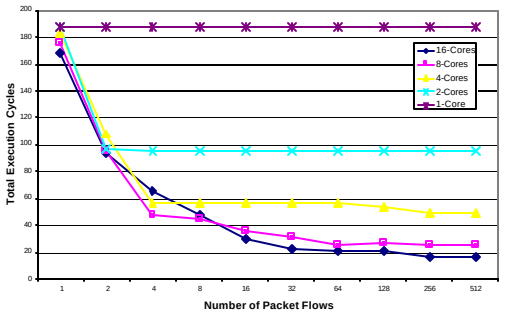

Figure 1. Execution Time vs. Packet Flows

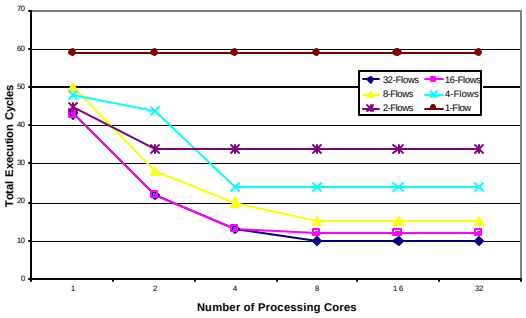

Figure 2: Execution Time vs. Number of Cores

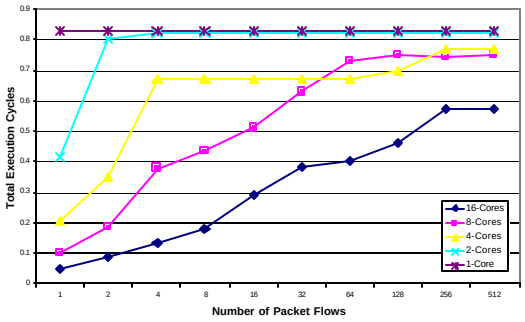

Figure 3. Processor Utilization vs. Packet Flows

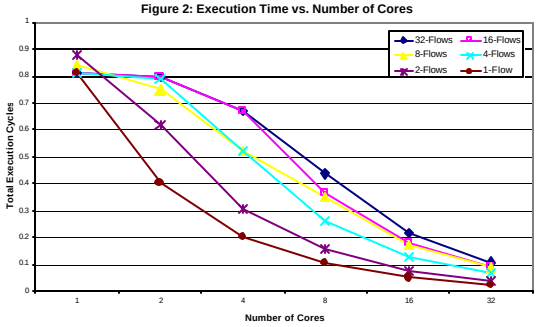

Figure 4. Processor Utilization vs. Number of Cores

efficient the NP is—a high utilization indicates that most processors are doing useful work; a low utilization indicates that most processors are idle. Figure 3 shows that as the workload becomes more independent, additional processors are well utilized; and as the workload becomes less independent, additional processors are not well utilized. This is the expected result since a completely dependent workload would mean all but one of the processing cores could be used at all. Figure 4 shows the utilization of the processing cores as a function of the number of cores. As we would expect, the utilization drops off as the number of cores increases, even for a highly independent workload. The more dependent the workload, the faster the utilization drops off—again as expected, since a dependent workload has less parallelism to exploit. A designer would use plots 3 and 4 to evaluate, in a cost/benefit sense, the effectiveness of additional cores on an anticipated workload in order to minimize product cost and development time.

V. Conclusion and Future Work

This project examined the effectiveness of parallel architectures for NP design. NP performance was measured over workloads with a varying number of inter-packet dependencies. Parallel processor core utilization was also measured. NP performance can benefit from a design that exploits parallelism, however limitations to the amount of performance improvement and processor effectiveness are shown. A limitation of this study is the amount of detail modeled in the processing elements. Both inter-processor communication and memory accesses are folded into the packet processing time and packet dependencies. Useful future studies would include modeling the overhead of both inter-processor communication and memory accesses separately.

REFERENCES

[1] W. Bux, W. E. Denzel, T. Engbersen, A. Herkersdorf and R. P. Luijten, "Technologies and Building Blocks for Fast Packet Forwarding," *IEEE Communications Magazine,* vol. 39, no. 3, Jan. 2001, pp. 70-77.

[2] P. Crowley, M. E. Fiuczynski, J. L. Baer and B. N. Bershad, "Characterizing Processor Architectures for Programmable Network Interfaces," *Proc. ACM Int. Conf. on Supercomputing,* Santa Fe, NM, May, 2000, pp. 54-65.

[3] K. Egevang and P. Francis, "The IP network address translator (NAT)," RFC 1631, Internet Engineering Task Force, May 1994. ftp://ftp.ietf.org/rfc/rfc1631.txt.

[4] L. Gwennap, "10G NPUs: Ready to Rumble," *Proc. Vitesse Network Processors Conference,* San Jose, CA, Oct. 2001, pp. 13 – 19.

[5] T. R. Halfhill, "Intel Network Processor Targets Routers," Microprocessor Report, vol. 13, no. 12, Sept. 13, 1999, pp. 66-68.

[6] S. W. Melvin and Y. N. Patt, "A Virtual Sequentially Mechanism for High Performance Network Processors," *Unpublished Manuscript,* 2001.

[7] E. M. Nahum, D. J. Yates, J. F. Kurose and D. Towsley, "Performance Issues in Parallelized Network Protocols," *Proc. of the USENIX Sym. on Operating System Design and Implementation,* Monterey, CA, Nov. 1994, pp. 125-137.

[8] B. A. Nayhfeh, L. Hammond and K. Olukotun, "Evaluation of Design Alternatives for a Multiprocessor Microprocessor," *Proc. IEEE/ACM Int. Sym. on Computer Architecture,* Philadelphia, PA, May 1996, pp. 67-77.

[9] C. Partridge, P. P. Carvey, E. Burgess, I. Castineyra, T. Clarke, L. Graham, M. Hathaway, P. Herman, A. King, S. Kohalmi, T. Ma, J. Mcallen, T. Mendez, W. C. Milliken, R. Pettyjohn, J. Rokosz, J. Seeger, M. Sollins, S. Storch, B. Tober, G. D. Troxel, D. Waitzman, and S. Winterble, "A 50-Gb/s IP Router," *IEEE/ACM Trans. on Networking,* vol. 6, no. 3, June 1998, pp. 237-248.

[10] D. Sonnier, "Empowering Intelligent Optical Networks," *Proc. Vitesse Network Processors Conference,* San Jose, CA, Oct. 2001, pp. 38-59.

[11] D. Tullsen, S. Eggers, H. Levy, "Simultaneous Multithreading: Maximizing On-Chip Parallelism," *Proc. IEEE/ACM Sym. on Computer Architecture,* Santa Margherita Ligure, Italy, June 1995, pp. 392-403.