Appears in the proceedings of IEEE International Conference on Computer Design (ICCD '03), October 2003

NpBench: A Benchmark Suite for Control plane and Data plane Applications for Network Processors

Byeong Kil Lee and Lizy Kurian John

Department of Electrical and Computer Engineering
The University of Texas at Austin
{blee, ljohn}@ece.utexas.edu

Abstract

Modern network interfaces demand highly intelligent traffic management in addition to the basic requirement of wire speed packet forwarding. Several vendors are releasing network processors in order to handle these demands. Network workloads can be classified into data plane and control plane workloads, however most network processors are optimized for data plane. Also, existing benchmark suites for network processors primarily contain data plane workloads, which perform packet processing for a forwarding function.

In this paper, we present a set of benchmarks, called NpBench, targeted towards control plane (e.g., traffic management, quality of service, etc.) as well as data plane workloads. The characteristics of NpBench workloads, such as instruction mix, parallelism, cache behavior and required processing capability per packet, are presented and compared with CommBench, an existing network processor benchmark suite [9]. We also discuss the architectural characteristics of the benchmarks having control plane functions, their implications to designing network processors and the significance of Instruction Level Parallelism (ILP) in network processors.

1. Introduction

As Internet and network technologies have grown and evolved exponentially, the requirements of network interfaces have become more complex and diverse. Various applications and protocols require more intelligent processing over the network. To keep up with current trends of emerging network applications, programmable microprocessors called network processors (NP) are introduced in network interfaces to handle the demands of modern network applications. Several vendors are releasing various network processors having different architectural features to meet the demands of network application workloads.

The bottleneck in communication networks is not just due to bandwidth anymore. Ability to provide flexible processing capability in order to support several emerging applications and meet their heavy processing workloads is equally important [1]. Major challenges for high bandwidth have reached tremendous advances from optical network approaches, a solution to bandwidth-centric bottleneck – currently 10Gbps (OC-192) at core router exists and 40Gbps (OC-768) is now starting to emerge. More complex protocols and various network services (e.g., Quality of Service, IPSec, IPv6, etc.) require significant processing power for highly intelligent applications; so the bottleneck of communication network has moved to the network nodes, in addition to the high bandwidth requirement. Accordingly, extracting representative benchmarks in several emerging network applications and characterizing their properties are essential for designing network processors and evaluating their performance.

While GPPs (General Purpose Processors) are flexible to rapidly developing network applications and protocols, they do not provide enough performance to process data at wire rates. For example, packet throughput of a 10Gbps link is 19.5 million packets per second, assuming a stream of minimum-sized packets of 64 bytes. Given a single processor of 1 GHz clock frequency, it can execute only 51 instructions per one packet time. Considering the required number of instructions for executing NP applications (Section 4.2.4), single processor is not enough to accomplish it. Highly parallel architectures are required to handle these workloads. Dedicated ASPs (Application Specific Processors) are designed to process packets at wire rates but are not allowed to add or change the features in order to support new environments. As shown in Figure 1, the network processor that exists at the middle points between GPPs and dedicated ASPs is a programmable processor or an instruction-set processor specialized for a particular application domain.

NP applications can be functionally categorized into two types of operations: data plane operations and control plane operations. The data plane performs packet operations such as forwarding packets. The control plane handles flow management, signaling, higher-level protocol and other control tasks [2]. Over the past few years, several vendors have been releasing NPs having a number of different architectures, but most of them are optimized for throughputs mostly in data plane. Also, existing benchmark suites for network processors primarily contain data plane workloads, which perform packet processing for a forwarding function. Although NPs have initially been targeted for data plane applications, they also play a major role in the control plane. In fact, with the increased demand for complex processing, the boundaries between data plane and control plane have become blurred [1]. The recent trend is that some control plane activities, such as TCP and SSL applications, are being considered as a commodity. Since

there are a lot of control mechanisms in TCP, it cannot be easily converted into an ASIC (Application Specific Integrated Circuit) and it has mostly been left to software solutions. From the above discussion, it is clear that control plane applications should be included in NP benchmarks.

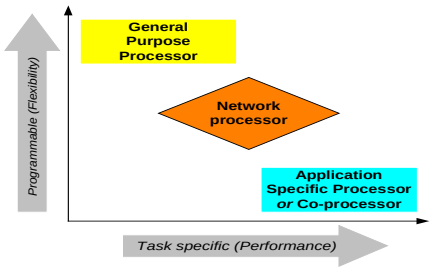


Figure 1. Network Processor

In this paper, we categorize modern NP applications into three functional groups: traffic-management and quality of service group (TQG), security and media processing group (SMG), and packet processing group (PPG). Every application of each group can be sorted to either a control plane application or a data plane application, or both. Based on these functional groups, we present a set of benchmarks, called *NpBench*, for control plane and data plane applications of modern network processors. The characteristics of NpBench workloads such as instruction mix, available parallelism, cache behavior and required number of instructions per packet are presented and compared with CommBench. We also discuss the architectural implications of the control plane benchmarks and their impact on designing NPs.

The rest of the paper is organized as follows. Section 2 provides background and related work on NPs and previously proposed benchmarks. Section 3 presents the applications in the proposed NpBench suites. In section 4, we present the application characteristics of benchmarks and discuss their implications. We conclude the paper in section 5 with an overview of functionality required for network processors of the next generation.

2. Background and Related Work

Current network bottleneck has moved from network bandwidth to the network node, where data traffic is intercepted or forwarded. Network processors can be used in various node positions over the network, such as core, edge and access routers. Core routers (10 Gbps rate) are placed in the middle of the network, so they are critical for performance and least responsive to flexibility. Edge routers are placed in between core and access devices, requiring medium data rate (2.5 Gbps) and a certain amount of flexibility. URL load balancers and firewalls are examples of edge router functions. Access routers (1 Gbps) provide network access to various devices. Most of their functions are related to aggregating and forwarding numerous traffic streams through the network [31].

The conventional applications of network interfaces mainly consist of packet processing and classification algorithms. However, modern roles of such an interface includes congestion control, network security, accounting, network address/protocol translations, load balancing and media transcoding. The processing capability of these emerging workloads must be at a level equivalent to the speed of the network. As a solution to this problem, many NP vendors use the concept of packet-level parallelism (PLP) to satisfy high-performance demands of networks. In fact, various companies use parallel architectures such as single chip multiple processor or fine-grain multithreaded processors to meet the packet-level parallelism [4].

Due to the variety of application spaces being addressed by network processors, there could be a wide range of NP architectures and implementations. For the enterprise service, several companies developed RISC-based NP with ASIC blocks for networking functions such as IXP 1200/2000 series by Intel [26], CXE-16 by Switchcore, CS2000 by Chameleon etc. For the high-end service, Motorola (C-port) [27], Lucent (FPP/RSP), EZChip (NP-1) and Vitesse/ Sitera(IQ2000) have used network-specific ASICs with the features of network classifying, QoS, etc. Some companies like Chrysalis-ITS, Alliance, NetLogic developed co-processors with the functions such as routing table, classification or cryptography [30].

Most of NP architectures employ multiple processing engines (PE), even though they each have different names such as micro engine, channel processor or task-optimized processor. Some are based on RISC cores having their PEs arranged in parallel or in a pipelined fashion. Another alternative is the VLIW based architecture [31]. Many RISC based NPs employ multithreading on their PEs to maximize the performance. To ensure fast context switching between tasks, the NP should have hardware support for multithreading. Figure 2 shows an overall architecture of typical network processor. In general, control and management functions have more complex processing requirements than data functions. GPPs have been used as control processors in commercial network products. Many NPs provide the function of control processor with an integrated core or externally via a host interface [35]. In this paper, we show that GPPs do not have enough processing capability to come up with increased demand for complex processing and higher data rates.

Benchmarks are an important issue in the design and evaluation of a processor. In NP fields, there are two benchmarks which were previously proposed: CommBench [9] and NetBench [5]. Wolf et. al. [9] present eight selected workloads called CommBench for traditional routers and active routers. CommBench has two groups of benchmarks namely Header Processing Applications (HPA) and Payload Processing Applications (PPA). Memik et. al. [5] proposed nine benchmarks called NetBench for micro-level, IP-level and application-level benchmarks. Nemirovsky [1] also discusses the guidelines for defining benchmarks and challenges of benchmark suites for network processors. He suggests that the benchmark should have two frameworks such as a task-specific benchmark focusing on a single algorithm or protocol and a rich-scenario benchmark

containing the complexity of real-life applications. EEMBC [24] and MiBench [21] have some network applications, but they only have routing and encryption applications.

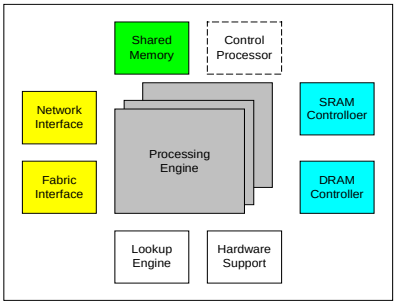


Figure 2. Overall Architecture of NP

Previously proposed benchmarks are mainly focused on data plane workloads. While the benchmarks of data plane applications have been reasonably well understood, there has been very little effort in designing control plane workloads that perform congestion control, flow management, higher-level protocols and other control tasks. Control plane workloads are just emerging and evolving in current network environments. The NpBench suite presented in this paper, is developed at the University of Texas in a project supported by Motorola. The suite includes several applications having control plane functions.

3. Overview of NpBench

This section presents the NpBench, a benchmark suite in the domain of the NP workloads. A benchmark suite should be representative of the domain of the application which it is going to be used for. We categorize network applications into three functional groups: traffic-management and quality of service group (TQG), security and media processing group (SMG), and packet processing group (PPG). This categorization is presented in Table 1. We choose ten representative applications from the functional groups for the first version of NpBench suite as shown in Table 2. The suite includes several control plane functions as they are missing from the available NP workloads.

We implement some of these selected applications to form the current release of the NpBench suite and the rest of them are referred from open source code site or other benchmarks [5][9][21]. The C code for the benchmarks is available on request [32].

3.1. Traffic-management and QoS Group (TQG)

TQG has a set of applications related to routing, scheduling, queuing, switching, signaling and quality of services. These applications contain both control plane processing and data plane processing. The first two benchmarks, WFQ and RED are the solutions of congestion control algorithms. In general, congestion occurs at a router when incoming packets arrive at a rate faster than the rate the router can switch them to an outgoing link. The two representative algorithms for congestion control are the scheduling algorithm and the queue management algorithm [10]. The scheduling algorithm determines which packet to be sent next and is used primarily to manage the allocation of bandwidth among flows (e.g., weighted fair queuing). According to the IETF (Internet Engineering Task Force) recommendation [10], the default mechanism for managing queue lengths to meet these goals in FIFO queues is the RED algorithm. SSLD is a content-based switching algorithm and MPLS is a technology used for quick forwarding of packets across backbones.

Table 1. Functional Grouping of Network Processor Workloads

Group	Applications	Data Plane	Control Plane
TQG (Traffic-management and Quality-of-Service Group)	Routing	X	X
	Scheduling	X	X
	Content-based switching	X	X
	Weighted fair queuing	X	X
	Traffic shaping	X	X
	Load balancing	X	X
	VLAN		X
	MPLS	X	X
	RSVP	X	X
	DiffServ	X	X
	IntServ	X	X
SMG (Security and Media Processing Group)	Block cipher algorithm	X	
	Message digest algorithm	X	
	Firewall application	X	X
	IPSec	X	X
	Virtual private network	X	X
	Public encryption	X	
	Usage-based accounting [23]	X	X
	H.323	X	
	Media transcoding	X	X
	Duplicate data suppression	X	
PPG (Packet Processing Group)	IP-packet fragmentation	X	
	Packet encapsulation	X	
	Packet marking/editing	X	
	Packet classification	X	
	Checksum calculation	X	

***WFQ (Weighted Fair Queuing)*:** WFQ [6-8][18] is a queue-scheduling algorithm to serve packets in order of their finish-times considering the weight on connections. As shown in Figure 3, various lengths of packets from incoming traffic are classified into different queues, which can be used for differential service. And they are scheduled by a specific mechanism that determines packets to be sent from the queues. WFQ uses each packet's estimated finish-time to decide packets to be sent.

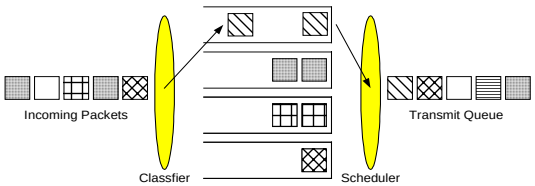


Figure 3. WFQ (Weighted Fair Queuing)

Table2. Descriptions of the NpBench Suite

Group	Application	Description
TQG	WFQ	*Weighted Fair Queuing* is a queue scheduling algorithm
	RED	*Random Early Detection* is an active queue management algorithm which drops arriving packets probabilistically
	SSLD	*Secure Sockets Layer Dispatcher* is an example of content-based switching mechanism
	MPLS	*Multi Protocol Layer Switching* is a forwarding technology using short labels
SMG	MTC	*Media Transcoding* is the process that a media object in one representation is converted into another representation for wide spectrum of client types
	AES	*Advanced Encryption Standard* (RijnDael) is a block cipher that encrypts and decrypts 128, 192 and 256 bits blocks
	MD5	*Message Digestion* algorithm takes as input a message of arbitrary length and produces as output a 128-bit fingerprint or message digest of the input
	DH	*Diffie-Hellman* key exchange allows two parties who have not met to exchange keys securely on an unsecure communication path
PPG	FRAG	*FRAG* is a packet fragmentation application
	CRC	*Cyclic Redundancy Check* is used in Ethernet and ATM Adaptation Layer 5 (AAL-5) checksum calculation

***RED (Random Early Detection)*:** RED [10][11][18] is an active queue management algorithm for routers. In contrast to the traditional queue management algorithm, which drops packets only when the buffer is full, the RED algorithm drops arriving packets probabilistically before coming into the queue. The decision of whether or not to drop an incoming packet is based on the estimation of the average queue size.

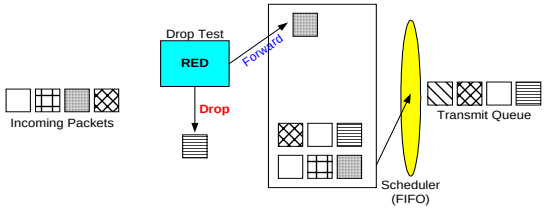


Figure 4. RED (Random Early Detection)

***SSLD (SSL Dispatcher)*:** SSLD [12][13] is one example of content-based switching mechanism in the server and client cluster environments. SSL typically runs over TCP (Transmission Control Protocol), which is used for secure processing of e-commerce applications. Once TCP connection is established, SSLD maintains the session ID information during authentication process, sharing the SSL information among the nodes in cluster. When reconnecting to the same server, a client can reuse the session state established during a previous SSL handshake which makes the workloads computationally heavy.

***MPLS (Multi Protocol Label Switching)*:** MPLS [14-17][33] is a forwarding technology, which eliminates the lookup of bulky IP headers and uses short labels for forwarding at the edge of the MPLS domain as shown in Figure 5. In this version of NpBench we concentrate on two control plane aspects of MPLS: Label Distribution and Label Generation. Two functions are extracted from MPLS, namely an upstream routing function (for an ingress edge router or a core router) and a downstream routing function (for a core router or an egress router).

3.2. Security and Media processing Group (SMG)

As the e-commerce industry has grown, security and accounting applications such as firewalls, admission control, encryption applications and usage based accounting, have become very common. With higher bandwidth, the demand for high quality multimedia service has also increased. Data stream manipulation, media transcoding, H.323 and several encoding applications [19] can be important issues of NP, associated with QoS. For security benchmarks, three components of IPSec [20] – Authentication Header (AH), Encapsulating Security Payload (ESP) and key management – are included in the SMG subgroup of NpBench.

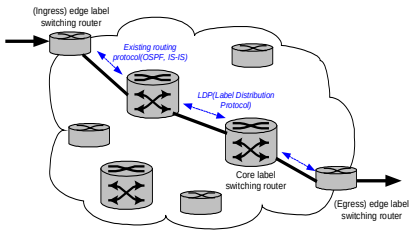


Figure 5. MPLS (Multi Protocol Label Switching)

MTC (Media TransCoding): Media Transcoding [28][29] is a process in which a data object in one representation is converted into another representation. In order to accommodate the wide spectrum of client capabilities, the media data is modified along the dimensions, fidelity, and resolution.

***AES (Advanced Encryption Standard)*:** Advanced Encryption Standard (RijnDael) [21] is a block cipher that encrypts and decrypts 128, 192 and 256 bit blocks, which is a U.S. government standard for encryption and digital signature. It is used for implementation of AH in IPSec.

***MD5 (Message Digestion)*:** MD5 algorithm [5] takes a message of arbitrary length as an input and produces a 128-bit "fingerprint" or "message digest" as an output. MD5 is a method to verify data integrity and is more reliable than checksum method. It is used to perform ESP in IPSec.

***DH (Diffie-Hellman)*:** Diffie-Hellman [5] key exchange allows two parties who have not met, to exchange keys securely on an unsecured communication path. Typically DH is used to exchange a randomly generated conventional encryption key, the rest of the exchange is then encrypted with the conventional cipher. It is used in key management in IPSec.

3.3. Packet Processing Group (PPG)

Packet processing group includes IP packet fragmentation, packet marking, editing and classification. Most applications are data plane processing.

***FRAG (Packet Fragmentation)*:** FRAG [9] is a packet fragmentation application. IP packets are split into multiple fragments for which some header fields have to be adjusted and a header checksum computed.

***CRC (Cyclic Redundancy Check)*:** 32-bit Cyclic Redundancy Check [21] is used in Ethernet and ATM Adaptation Layer 5 (AAL-5) checksum calculation.

4. Implementation and Characterization of NpBench

4.1. Implementation and Experimental Methodology

We develop NpBench control-plane functions at the application level using C language and refer most of the data-plane functions from open source code or other benchmarks [5][9][21]. The MPLS benchmark is based on effort by Datta [33]. We used the SUN Shade binary instrumentation tool [22] to obtain the dynamic traces while executing NpBench applications. We also use cachesim5, a cache analyzer tool of SUN Shade [22], to perform cache simulation and Tetra [8] to get available parallelism with constraints.

We utilize randomly generated input packets for the characterization of the benchmark. For TQG, WFQ uses packet size and queue information as an input. RED uses incoming packet size and average queue size to decide whether the packet is to be dropped or put in the FIFO queue. The clientHello message and serverHello message of the SSL protocol [13] are used with randomly generated session ID information for the SSLD experiments. The values of FEC (Forwarding Equivalence Class) identification numbers are used as input for the MPLS functions. The RED implementation allows an option of congestion environment, which is controlled by transmission rate of the queue. The SSLD inputs can be different session IDs with different reusability factors. In SMG, MTC can be separated into two components which are policy modules to get adaptive transcoding policies and transformation modules to perform real transcoding. The policy decision module can be executed independently with an execution option. AES can use any files as an input data and MD5 can make a fingerprint of any files or strings for the input. DH generates and exchanges any given number of Diffie-Hellman key pairs. FRAG and CRC employ randomly generated IP header as an input data. Under the above simulated network environments, the characteristic of NpBench is investigated.

4.2. Benchmark Characteristics

In this section, we present the experimental results on instruction distribution, cache behavior and parallelism of the NpBench. These metrics are essential information for understanding dynamic characteristics of the application and for designing the processor architecture. We also explore the required number of instructions to process one packet data, assuming a minimum-sized packet of 64 bytes.

4.2.1. Instruction Mix

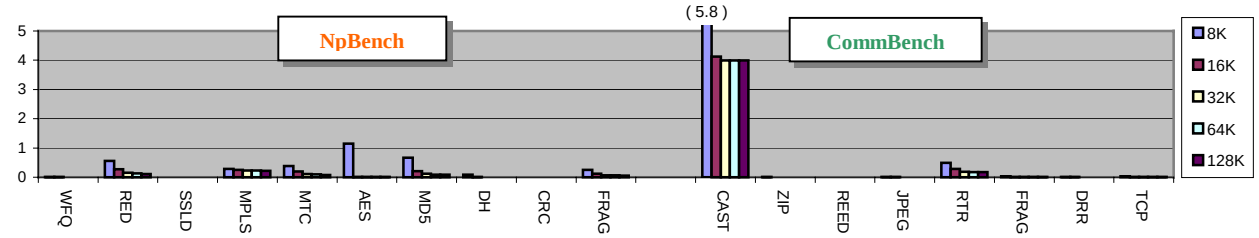
In these experiments, we investigate the number of instructions in the NpBench applications. Table 3 shows the dynamic instruction mix during execution. From this workload distribution, we can observe that computational operations occupy a significant share of the total instruction mix (53% on the average). Branch operations (branch, jump and call) are heavily used in the applications having control plane functions (23.7%) such as WFQ, SSLD and MPLS, for finding fair conditions of each packet, looking up session reuse conditions of each session request and investigating same forwarding equivalence class respectively. Data plane functions have relatively small percentage of branch operations (11.1%).

Since the data plane application is to handle more packet data and coefficients for performing the algorithm within payload processing, we observe that the data plane application uses more load and store operations (31.2% on average) than the applications having control plane functions (23.5%). In the case of SSLD, as the reusability factor used in SSLD increases, we see that the required computation workloads for new session request could be avoided and the required number of instructions could be reduced.

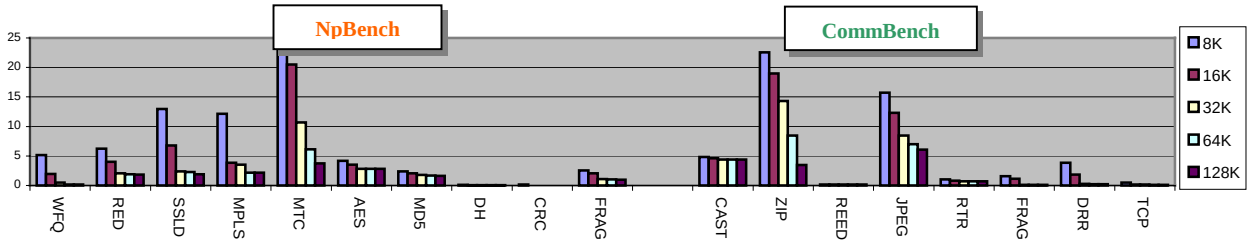
Compared to CommBench, the NpBench has similar percentage of ALU operations out of the total instructions. However, branch operations are heavily used in NpBench control plane applications (23.7%), followed by CommBench-HPA (20.6%), CommBench-PPA (15.3%) and NpBench data plane applications (11.1%).

Table 3. Instruction Mix

NpBench								CommBench							
App.	int/float	shift	logic	branch	load	store	etc	App.	int/float	shift	logic	branch	load	store	etc
WFQ	20.6	16.9	0.0	29.2	16.2	7.9	9.3	CAST	25.4	17.0	20.4	8.9	20.4	7.4	0.5
RED	39.7	7.2	0.0	15.3	23.6	10.9	3.0	ZIP	34.0	8.0	12.4	20.2	19.4	5.6	0.4
SSLD	57.0	0.0	0.0	28.3	14.4	0.2	0.0	REED	40.2	11.7	7.1	21.4	14.7	4.9	0.0
MPLS	35.8	11.4	8.6	22.0	16.1	4.9	1.3	JPEG	43.8	16.1	2.7	10.8	16.5	9.7	0.3
MTC	33.9	2.6	10.5	11.2	21.7	18.1	1.9	*PPA*	*35.8*	*13.2*	*10.7*	*15.3*	*17.8*	*6.9*	*0.3*
AES	10.5	18.4	26.9	7.0	29.4	7.7	0.1	RTR	20.6	0.7	11.0	23.4	41.3	2.7	0.2
MD5	45.1	13.0	20.5	7.5	7.1	6.7	0.2	FRAG	41.5	3.8	15.0	20.4	12.8	6.5	0.0
DH	24.0	12.0	10.9	9.7	27.0	11.3	5.1	DRR	31.7	1.0	0.2	18.3	41.8	6.9	0.1
CRC	25.0	10.0	15.0	10.0	25.0	15.0	0.0	TCP	37.2	5.2	12.5	20.5	16.4	7.1	1.3
FRAG	40.0	3.8	15.1	21.4	12.2	6.1	0.7	*HPA*	*32.8*	*2.6*	*9.7*	*20.6*	*28.1*	*5.8*	*0.4*
Avg.	*33.2*	*9.5*	*10.8*	*16.2*	*19.3*	*8.9*	*2.2*	*Avg.*	*34.3*	*7.9*	*10.2*	*18.0*	*22.9*	*6.3*	*0.4*

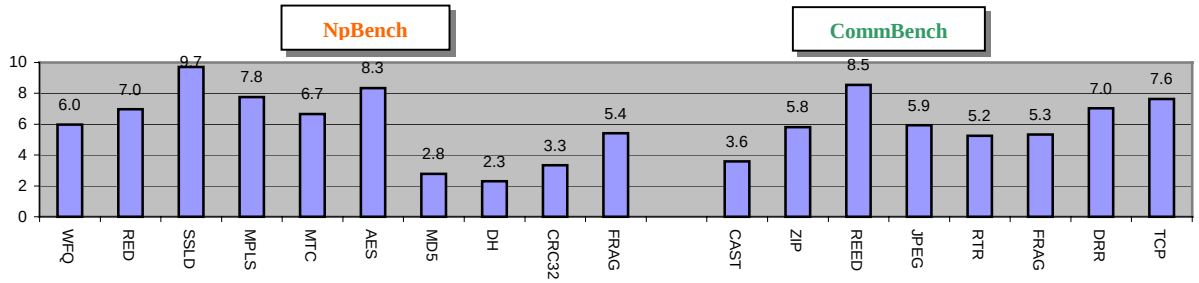


(a) Instruction Cache Miss Rate (%) with Varying Cache Size

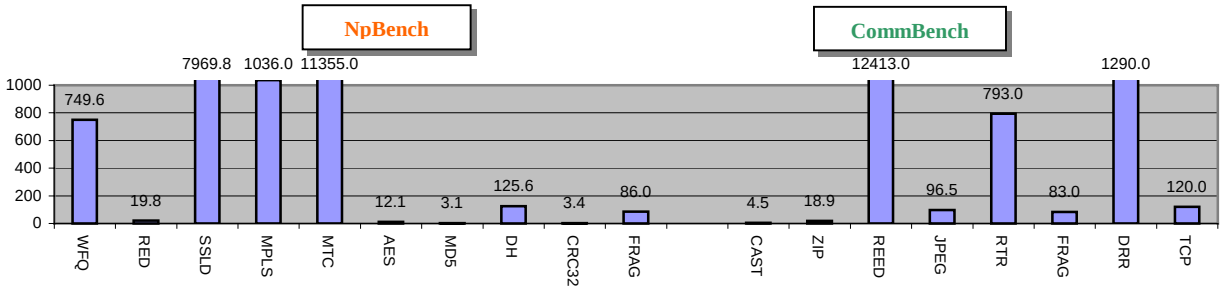


(b) Data Cache Miss Rate (%) with Varying Cache Size

Figure 6. The Cache Performance



(a) Available Parallelism with Limited Function Units of 10



(b) Available Parallelism with Unlimited Function Units

Figure 7. Available Parallelism

4.2.2. Cache Behavior

It is important to understand cache behavior of the NpBench applications. Cache performance for 2-way set associative cache was evaluated with varying cache sizes. A line size of 32 bytes was commonly used for all cache configurations. Figure 6 shows the cache miss rates for NpBench applications. If control plane processor is implemented with an integrated core within the NP, cache sizes tend to be small because of area constraints. Most of NPs have 4K to 32K of cache sizes [31]. Most of the NpBench applications perform same operations with various inputs, explaining the excellent instruction cache hit ratios. However, data cache performance of these applications, especially for control plane, is poor for small cache sizes. The average miss ratios converge to 0.056% for I-cache and 1.531% for D-cache with increasing cache sizes. Instruction cache sizes larger than 16KB marginally increase cache performance and same observations are made with data cache sizes larger than 32KB. This implies 16KB and 32KB could be optimal I-cache and D-cache size for NpBench application. As shown in Figure 6, the applications having control plane function show more sensitivity on varying cache sizes. We also find that CommBench and NpBench

show similar trends in cache miss rates, for example, poor data cache behavior.

In general, each application can be implemented with one PE having its L1 cache within the PE itself, and L2 cache of the network processor can be shared by several PEs. For reduction of L2 memory access latency, a few mechanisms are proposed [34].

4.2.3. Available ILP

We explore the instruction level parallelism of NpBench applications as a function of the inherent data dependencies and data flow constraints with limited/unlimited number of functional units. If 10 functional units are allowed, available parallelism ranges from 2 to 9 as shown in Figure 7 (a). NpBench control plane applications have more ILP (7.41 on an average) than NpBench data plane applications (4.43), and the available parallelism of CommBench (6.14) is in the middle. When the number of function units are infinity, the difference of available ILP between control plane and data plane applications are larger (NpBench control plane:4,226, NpBench data plane: 46, CommBench:1,852) as shown in Figure 7 (b). From these results, we can see that control plane operations have more opportunity to exploit aggressive ILP than data plane. Security applications except for AES exhibit lower parallelism due to the need to perform encryption tasks. Since AES is a block cipher algorithm, it shows relatively higher ILP than other security applications. Even though the applications having control plane function have large amount of branch operations, they have more execution parallelism, which means there exists a room to improve performance of control plane processor with more parallel implementation. While most of NPs are implemented with several PEs to exploit packet level parallelism (PLP) for data plane operations, they can exploit more instruction level parallelism (ILP) within the PEs or control plane processors.

4.2.4. Required number of instructions per packet

Some control plane and data plane workloads, from NpBench (e.g., WFQ, RED) and CommBench (e.g., DRR, FRAG), are used to get the required number of instructions per packet. These experiments employ one million packets of data as each input. The required processing capability of control plane is estimated at 180 to 32,000 instructions per packet, while data plane is from 330 to 440 (10,170 for control plane and 380 for data plane on average). From the graph in Figure 8, we see that control plane functions need larger processing capabilities, since their algorithms have higher complexity to meet sophisticated network services. For example, WFQ has to estimate each packet's finish-time and then classify the incoming packet into different queues, in order to maintain fairness and support QoS. This makes the algorithm more complex and the number of instructions larger. In contrast to that, FRAG performs relatively simple algorithm to split packets into multiple fragments, requiring less processing capabilities.

As shown in Table 4, larger packet throughput is demanded for higher line rate. Assuming a stream of minimum-sized packets of 64 bytes and one clock frequency for executing one instruction, packet throughput of a 10 Gbps link is 19.5 million packets per second which means one packet is arrived every 51.2 nanosecond. Given a single processor of 1 GHz clock frequency, it can execute only 51 instructions per one packet time. Since a single processor is not enough to cope up with wire speed and handle the workload of those applications, current trend of NPs is to use single chip multi-processors. Not only more parallelism, but also changes in instruction set architecture (ISA) with sophisticated programmability, should be considered to increase the number of instructions per cycle.

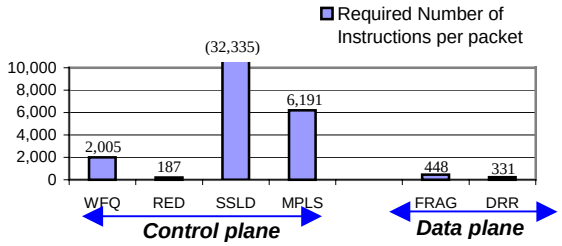


Figure 8. Required number of instructions per packet

Table 4. Processing Capability of Single Processor according to Line Rates

Line rate	Throughput (packets/s)	One packet time	Processor clock frequency	Allowable # of instructions per one packet time
1Gbps	1.95 M	512 ns	500 MHz	*256*
			1 GHz	*512*
10 Gbps	19.5 M	51.2 ns	500 MHz	*25*
			1 GHz	*51*
40 Gbps	78.12 M	12.8 ns	500 MHz	*6*
			1 GHz	*12*

4.3. Architectural Implications

Since NPs would be used in routers over the network, an important issue to be considered in designing NPs is the processing capability without slowdown of required wire speed. Based on the characterization of NpBench, several issues are relevant while designing the network processor to accomplish demanding performance and throughput.

To reduce the number of instructions per cycle, frequently used instruction pairs can be considered as new instruction sets to accomplish higher throughput that can come up with the required number of instruction per packet. In this case, hardware implementation without loss of overall performance can be an important challenging issue.

When control plane processor is implemented with an integrated core within the NP, cache sizes tend to be small. From the observation of cache behavior, data cache performance need to be improved at smaller cache sizes. When several PEs are integrated on a single chip, the problems including the shared memory problem also should be solved for network processors.

Several current network processors use packet level parallelism for data plane operations. However, if large numbers of PEs are used, the processing time for each individual packet would be longer and utilization ratio could

be deteriorated [25]. Based on the analysis of available parallelism with limited/unlimited function units, NP architectures can exploit more instruction level parallelism (ILP) within the PEs or control plane processors.

5. Conclusion

As the network environment is rapidly changing, network interfaces demand highly intelligent traffic management in addition to the basic requirement of wire speed packet forwarding. Extracting representative applications and characterizing network workloads is essential for designing network processors and for evaluating their performance. Several vendors are releasing various network processors in order to handle these demands, but they are primarily oriented for data plane functions. Also, existing benchmark suites for the network processor primarily contain data plane workloads, which perform packet processing for forwarding operations.

In this paper, we present a set of benchmarks, called NpBench, targeted towards control plane workloads as well as data plane workloads. The characteristics of NpBench workloads such as instruction mix, cache behavior, available parallelism and required processing capability per packet are presented and compared with CommBench. We also discuss the architectural implications of control plane workloads and the significance of additional parallelism to perform NP applications at wire speed.

Acknowledgements

This research is supported by Motorola Corporation. The authors are also supported in part by the National Science Foundation under grant number 0113105 and by Tivoli, Intel and IBM Corporations. We would like to thank Ramyanshu Datta for MPLS implementation efforts and Chen-Chau Chu, Kate Stewart and David Murrell (Motorola) for valuable comments.

References

[1] A. Nemirovsky, “Towards Characterizing Network Processors: Needs and Challenges,” Xstream logic, white paper.

[2] J. Williams, “Architectures for Network Processing,” IEEE International Symposium on VLSI Technology, Systems, and Applications, 2001.

[3] S. Keshav and R. Sharma, “Issues and Trends in Router Design,” IEEE Communications Magazines, May 1998.

[4] P. Crowley, M. E. Fiuczynski, J-L Baer and B. Bershad, “Workloads for Programmable Network Interfaces,” In Workload Characterization for Computer System Design, Kluwer Academic Publishers, L. John and A. Maynard, ISBN 0-7923-7777-x.

[5] G. Memik, W. Mangione-smith and W. Hu, “NetBench: A Benchmarking Suite for Network Processors,” ICCAD 2001.

[6] A. Demers, S. Keshav, and S. Shenker, “Analysis and Simulation of a Fair Queuing Algorithm,” In ACM SIGCOMM, 1989.

[7] D. Stiliadis and A. Varma, “Efficient Fair-queuing Algorithms for Packet-switched Networks,” IEEE/ACM transactions on Networking, Apr. 1998.

[8] T. M. Austin and G. S. Sohi, “TETRA: Evaluation of Serial Program Performance on Fine-grain Parallel Processors,” University of Wisconsin Technical Report #1162, Jul. 1993.

[9] T. Wolf and M. Franklin, “CommBench - A Telecommunications Benchmark for Network Processors,” International Symposium on Performance Analysis of Systems and Software, Apr. 2000.

[10] B. Branden, et al., “Recommendations on Queue Management and Congestion Avoidance in the Internet,” IETF Internet Draft 1997.

[11] S. Floyd and V. Jacobson, “Random Early Detection Gateways for Congestion Avoidance,” IEEE Transactions on Networking, Vol.1, No. 4, Aug. 1993.

[12] G. Apostolopoulos, D. Aubespin, V. Peris, P. Pradhan and D. Saha, “Design, Implementation and Performance of a Content-Based Switch,” INFOCOM'00, Mar. 2000.

[13] SSL Protocol version 3.0, http://wp.netscape.com/eng/ssl3/ssl-toc.html.

[14] Multi Protocol Label Switching Architecture (RFC 3031) http://www.ietf.org/rfc/rfc3031.txt.

[15] LDP Specification (RFC 3036), http://www.ietf.org/rfc/rfc3036.txt.

[16] G. Armitage, “MPLS: The Magic Behind the Myths,” IEEE Communications Magazine, Jan. 2000.

[17] X. Xiao and L. M. Ni, “Internet QoS: A Big Picture,” IEEE Network, Mar./Apr. 1999.

[18] W. Bux, W. E. Denzel, T. Engbersen, A. Herkersdorf, and R. P. Luijten, “Technologies and Building Blocks for Fast Packet Forwarding,” IEEE Communication Magazine, Jan. 2001.

[19] B. Lee and L. John, “Implications of Programmable General Purpose Processors for Compression / Encryption Applications,” IEEE 13th International Conference on Application-specific Systems, Architectures and Processors, Jul. 2002.

[20] Security Architecture for the Internet Protocol (RFC 2401) http://www.ietf.org/rfc/rfc2401.txt.

[21] M. R. Guthaus, J. S. Ringenberg, D. Ernst, T. M. Austin, T. Mudge and R. B. Brown, “MiBench: A Free, Commercially Representative Embedded Benchmark Suite,” In proceedings of the 4th annual IEEE International Workshop on Workload Characterization, Dec. 2001.

[22] R. F. Cmelik and D. Keppel, “Shade: A Fast Instruction-set Simulator for Execution Profiling,” SUN Microsystems Inc., Technical Report SMLI TR-93-12, 1993.

[23] W. Fang, “Building An Accounting Infrastructure for the Internet,” Princeton University Computer Science Technical Report, TR-599-99, Mar. 1999.

[24] Embedded Microprocessor Benchmarking Consortium, http://www.eembc.org.

[25] H. Liu, "A Trace Driven Study of Packet Level Parallelism", Proc. International Conference on Communications (ICC), 2002.

[26] Intel IXP1200 Network Processor, http://www.intel.com/design/network/products/npfamily/ixp1200.htm.

[27] C-Port Network Processors, http://e-www.motorola.com.

[28] J. R. Smith, R. Mohan, C. Li, “Content-based Transcoding of Images in the Internet,” IEEE Conference on Image Processing (ICIP-98), Oct. 1998.

[29] R. Han and P. Bhagwat, et al, "Dynamic Adaptation in An Image Transcoding Proxy for Mobile Web Browsing," IEEE Personal Communications Magazine, Dec. 1998.

[30] The Electronic Design, Jul. 2000, http://www.elecdesign.com.

[31] N. Shah, “Understanding Network Processors,” Master's thesis, University of California, Berkeley, Sep. 2001.

[32] NpBench Website, Laboratory for Computer Architecture (LCA), University of Texas at Austin, http://www.ece.utexas.edu/projects/ece/lca/npbench.

[33] R. Datta, “Development and Characterization of MPLS Workloads,” UT at Austin, EE382M Project Report, 2002.

[34] G. Memik and W. H. Mangione-Smith, “Improving Power Efficiency of Multi-Core Network Processors Through Data Filtering,” In Proc. of International Conference on Compilers, Architecture and Synthesis for Embedded Systems, Oct. 2002.

[35] A. Heppel, “An Introduction to Network Processors,” Roke Manor Research Ltd., White paper, Jan. 2003.