NetBench: A Benchmarking Suite for Network Processors

Gokhan Memik **William H. Mangione-Smith** **Wendong Hu**
Department of Electrical Engineering
University of California, Los Angeles
Los Angeles, CA 90095
{memik, billms, wendong}@ee.ucla.edu

Abstract— In this study we introduce NetBench, a benchmarking suite for network processors. NetBench contains a total of 9 applications that are representative of commercial applications for network processors. These applications are from all levels of packet processing; Small, low-level code fragments as well as large application level programs are included in the suite.

Using SimpleScalar simulator we study the NetBench programs in detail and characterize the network processor workloads. We also compare key characteristics such as instructions per cycle, instruction distribution, branch prediction accuracy, and cache behavior with the programs from MediaBench. Although the aimed architectures are similar for MediaBench and NetBench suites, we show that these workloads have significantly different characteristics. Hence a separate benchmarking suite for network processors is a necessity. Finally, we present performance measurements from Intel IXP1200 Network Processor to show how NetBench can be utilized.

1. INTRODUCTION

Emerging applications in the networking field demand increasingly higher network bandwidths. In addition, new applications and protocols not only require the network to deliver packets. Instead, they have requirements such as quality of service guarantees, secure transmission of data and intelligent/dynamic routing and switching among others. These applications require significant amount of processing which should be satisfied by the processor. Coupled with the higher network link speeds this set of features puts a heavy demand on the network processing elements.

Traditionally, embedded processors in networks are either custom-designed ASIC chips or variations of general-purpose processors. Both schemes have their advantages and shortcomings. ASIC chips have better performance, but they have higher manufacturing costs and lack the flexibility of programmable processors. If there is a change in the protocol or application, it is hard to reflect the change onto the design. General-purpose processors, on the other hand, are not optimized for networking applications and hence do not provide satisfactory performance for most of the applications. Network processors eliminate the drawbacks of general-purpose processors and ASIC designs by combining the flexibility of general-purpose programmable processors and performance of ASIC chips.

Soon after their introduction [10], network processor market became one of the fastest growing segments of the microprocessor industry. Only in the last year, more than 40 new vendors have announced their network processor architectures. Although, these processors aim at the same application domains, they vary widely in their architectural designs. Hence, there is a tremendous need to evaluate the performances of these different designs.

A designer of a product should know the type of applications, based on marketing requirements, for which the processor is optimized. Similarly, customers benefit from benchmarks by selecting the product that gives the best performance for the applications they consider important (when benchmarks are aligned with commercial workloads). In this paper, we create a benchmarking suite by defining a set of applications that are common for network processors. This benchmarking suite can be used to evaluate performance of different network processor designs. We also investigating several characteristics of these networking applications to understand their nature and compare some characteristics of these applications with the applications from MediaBench [9]. Finally, we demonstrate how NetBench can be utilized by providing a performance measurement of Intel IXP1200 Network Processor [6].

This paper is organized as follows. In the next section, we provide the necessary background and discuss the related work. In Section 3, we present the applications in NetBench. Applications in NetBench are compared with the MediaBench applications in Section 4. In Section 5, we present experimental results for Intel IXP1200 simulations. Section 6 concludes the paper.

2. RELATED WORK

Network processors are a class of programmable IC's based on SOC (system-on-a-chip) technology that implement communication-specific functions more efficiently than general-purpose processors. Crowley, et. al. [4] evaluate different design mechanisms for network processor. They measure the performance of a VLIW-based, a SMT-based, a fine-grain multithreaded multiprocessor, and a single-chip multiprocessor.

Benchmarks play a major role in any product design process. SPEC [19] benchmarks have been well accepted and used by several processor manufacturers and researchers to measure the effectiveness of their design. Other fields have popular benchmarking suites designed for the specific application domain: TCP [20] for database systems, SPLASH [22] for parallel machine architectures. The need for a benchmarking suite in the network processor area has been pointed out by several researchers. Nemirovsky [11] discusses the requirements and challenges of a benchmarking suite for network processors. He defines a set of metrics to be used with any benchmarking suite and draws the guidelines for defining a benchmark.

There has also been some effort in characterizing the network processor applications. Wolf and Franklin [21] simulate four packet header processing applications along with four payload processing applications.

3. NETBENCH PROGRAMS

In this section, we present the applications in NetBench. Any benchmarking suite should be a representative of the applications in the domain the benchmark is designed for. This was the most important criterion in our selection of the applications.

Network Processor applications contain a large variety of tasks such as traditional routing and switching tasks to much more complicated applications containing intelligent routing and switching de-

cisions. Therefore, any benchmarking suite attempting to represent the applications on Network Processors should consider all levels of a networking application. We have categorized these levels into three: Low or Micro level routines contain operations nearest to the link or operations that are part of more complex tasks; Routing level applications are similar to traditional IP level routing and similar tasks; Application level programs, which have to parse the packet header and sometimes a portion of the payload and make intelligent decisions about the destination of the packet. We list the applications in NetBench according to the category they belong:

3.1 Micro-Level Programs

In our benchmarking suite, we have 2 micro-level programs:
CRC: The CRC-32 checksum calculates a checksum based on a cyclic redundancy check as described in ISO 3309 [7]. CRC-32 is used in Ethernet and ATM Adaptation Layer 5 (AAL-5) checksum calculation. The code is available in the public-domain [3].
TL: TL is the table lookup routine common to all routing processes. We have used radix-tree routing table which was used in several UNIX systems. The code segment is from FreeBSD operating system [5].

3.2 IP-Level programs

These programs make a decision depending on the source or destination IP of the packet.
Route: Route implements IPv4 routing according to RFC 1812 [1]. Route implements the table lookup along with internet checksum (for the header). It makes the necessary changes in the header (for example, the Time-To-Live value), fragments the packet if necessary and forwards it. The code is from the FreeBSD operating system [5].
DRR: Deficit-round robin (DRR) scheduling [18] is a scheduling method implemented in several switches today. In DRR, all the connections through the router have separate queues. Using these queues, the router tries to accomplish a fair scheduling by allowing same amount of data to be passed from each queue. The implementation is based on [18].
NAT: Network Address Translation (NAT) is a common method for IP address simplification and conservation. It operates on a router, usually connecting two networks, and translates the private (not globally unique) addresses in the internal network into legal addresses before packets are forwarded onto the other network. Hence, for any departing packet, the source IP on the packet should be changed. The program accomplishing this task is using several routines from FreeBSD operating system [5].
IPCHAINS: IPCHAINS is a firewall application that checks the IP source of each of the incoming packet and decides either to pass the packet through the firewall (accept), to deny the packet (deny), to modify it (masq), or to reject the packet and send information to the sender (reject). The implementation is from Rustcorp Inc. [17].

3.3 Application-Level Programs

These programs are the most time consuming applications in NetBench due to their processing requirements.
URL: URL implements the URL-based switching, which is a commonly used context-switching mechanism. In URL-based switching, all the incoming packets to a switch are parsed and switched according to the URL requested by it. This increases the utility of specialized servers in a server farm. The implementation is based on the description from PMC-Sierra [14].
DH: Diffie-Hellman (DH) is a common public key encryption-decryption mechanism. It is the security protocol employed in several Virtual Private Networks (VPN's). The implementation is from RSA Data Security, Inc [16].
MD5: Message Digest algorithm (MD5) creates a cryptographically secure signature for each outgoing packet, which is checked at the destination [15]. If the received packet does not match the signature, then the receiver will detect it and discard the packet. The implementation is also from RSA Data Security, Inc [16].

4. PROGRAM CHARACTERISTICS

In this section, we compare several characteristics of NetBench applications with MediaBench [9] applications. MediaBench is designed for multimedia and communication systems, which are in many ways similar to network processors. We have selected MediaBench to compare against NetBench, due to this similarity of the aimed processor architectures. Although processor architectures are similar, we show that the applications for these architectures are significantly different, thus validating the need for a separate benchmarking suite for network processors.

4.1 Simulation Environment

In order to compare NetBench and MediaBench applications, we have performed several simulations on SimpleScalar simulator [2]. We simulate a 4-way superscalar processor with 64 KB of direct-mapped level 1 (L1) data and instruction caches and a 1 MB unified level 2 (L2) cache much like in the Alpha 21264 [8] processor. The L1 and L2 cache latencies are set to 2 and 10 cycles, respectively. The simulated processor uses a bimodal branch predictor [23] with 2048 table entries. We have simulated 9 programs from MediaBench to perform the comparison. The remaining applications in MediaBench are office development programs and hence are left out.

NetBench applications take a IP header trace as input. We have used the traces from Columbia University available in the public domain [13]. In the experiments, the first 10000 packets are read by the application. All the applications use this trace except the dh program, which generates and communicates 20 Diffie-Hellman key pairs and hence does not need any packet trace. The routing table sizes for drr, ipchains, nat, route and tl is set to 128. The input variables along with the application code can be found at the NetBench web site [12].

4.2 Experimental Results

In this section we compare the instruction level parallelism(ILP), branch prediction accuracy, instruction distribution and cache behavior of NetBench applications with MediaBench applications. These are the key architectural characteristics of an application and hence are used to differentiate between different application sets.

4.2.1 Instruction Level Parallelism

The first characteristic we explore is the instruction level parallelism (ILP) measured in instructions per cycle (IPC). It is well known that the networking applications have a high data-level parallelism. However, dependency between the instructions that process same data is not known. We first study this characteristic. Table 1 gives the results for the instruction level parallelism. It gives the instructions per cycle values along with the total number of instructions and cycles executed. The average IPC value of NetBench applications is 14.5% higher than the average of MediaBench applications. A statistical study shows that the NetBench applications have a higher IPC value using a 90% confidence interval.

4.2.2 Branch Prediction Accuracy

The branch predictor simulated was explained in Section 4.1. Table 1 summarizes the results. In average, the predictor has a 4.78% better address prediction accuracy and 4.18% better direction prediction rate for NetBench applications. The lower prediction rate

Table 1: Instructions per cycle (IPC) and branch prediction values for the NetBench and MediaBench applications. IPC measures instruction level parallelism (ILP). IPC value is high when the dependency of the instructions within a program is low. Avg. is the arithmetic mean.

NetBench Programs						MediaBench Programs					
Program	#of inst. [M]	#of cycles [M]	IPC	address pred. rate[%]	direction pred. rate[%]	Program	#of inst. [M]	#of cycles [M]	IPC	address pred. rate[%]	direction pred. rate[%]
crc	239	121	1.97	99.1	99.1	*adpcm*	6.6	5.8	1.13	73.1	73.1
dh	2434	1432	1.69	87.8	87.9	*epic*	6.8	4.9	1.38	93.1	93.1
drr	61	41	1.48	97.9	97.9	*g721*	1076	610	1.76	90.6	91.0
ipchains	74	44	1.65	93.9	95.0	*ghostsc.*	1294	935	1.38	95.2	95.6
md5	204	104	1.96	96.8	97.0	*gsm*	73	40	1.80	98.3	98.4
nat	21	14	1.48	90.5	91.1	*jpeg*	3.5	2	1.75	93.4	94.2
route	18	12	1.51	92.1	92.2	*mesa*	68	51	1.24	94.1	99.8
tl	12	9.3	1.38	91.2	91.3	*mpeg*	1133	861	1.34	76.7	76.9
url	171	94	1.81	96.0	96.0	*pegwit*	12.7	9.7	1.31	87.8	87.8
Avg.	359	207	1.66	93.9	94.2	*Avg.*	408	280	1.45	89.1	89.9

Table 2: Percentage of instructions executed from each instruction category for NetBench and MediaBench applications. The abbreviations are: LD/ST for load and store instructions, Jump for unconditional jump instructions, Branch for conditional branch operations, Add for addition instructions, Sub for subtraction operations, Log. for bitwise logical operations, and Arit. is for arithmetic shift operations like shift left logical.

NetBench Programs								MediaBench Programs							
Prog.	LD/ST	Jump	Branch	Add	Sub	Log.	Arit.	Prog.	LD/ST	Jump	Branch	Add	Sub	Log.	Arit.
crc	25.8	2.4	5.2	44.1	0.1	6.8	12.3	*adpcm*	7.3	2.0	22.7	25.5	5.5	9.0	27.9
dh	32.4	1.9	2.8	51.8	1.1	2.0	6.4	*epic*	19.5	0.8	15.2	44.4	1.1	3.1	9.5
drr	38.9	4.2	8.4	45.9	0.2	0.2	1.9	*g721*	31.4	4.4	6.6	47.6	0.7	1.3	7.3
ipcha.	26.5	7.5	9.7	41.5	0.3	5.8	7.9	*ghost.*	18.4	5.0	12.2	41.1	2.3	8.0	10.2
md5	13.6	0.5	5.8	41.4	0.1	19.4	15.7	*gsm*	10.1	4.8	11.4	33.9	2.1	3.7	30.9
nat	30.1	4.8	6.6	49.0	0.6	0.9	7.0	*jpeg*	23.8	0.5	3.4	54.5	2.1	3.3	12.0
route	30.3	4.7	6.4	49.1	0.6	0.9	7.3	*mesa*	26.1	4.3	7.5	38.4	0.1	5.2	2.9
tl	28.9	4.0	5.9	48.4	0.9	1.5	10.0	*mpeg*	22.4	0.9	12.3	43.4	11.4	0.4	6.1
url	22.5	3.3	13.9	41.6	0.7	12.7	3.3	*pegwit*	19.5	1.9	10.4	45.7	0.2	10.2	12.0
Avg.	27.7	3.7	7.2	45.9	0.5	5.6	8.0	*Avg.*	19.8	2.7	11.3	41.6	2.8	4.9	13.2

coupled with having more frequent branches as we will study in the next subsection causes the lower IPC values for the MediaBench applications observed in Subsection *4.2.1*.

4.2.3 Instruction Distribution

In these simulations, we count different number of instruction types in the NetBench and MediaBench applications. The results are summarized in Table 2. The table gives the number of instructions executed from each of the major instruction categories. The two benchmarking suites differ in almost all instruction categories, but we concentrate on the load/store and conditional branch operations, because they are more important than other instructions in determining the nature of an application and its performance. In average, NetBench applications have a higher load/store frequency. This shows the data-intensive nature of these applications. The MediaBench applications, on the other hand, have a higher conditional branch instruction percentage. A statistical analysis shows that with a 95% confidence interval NetBench applications have higher load/store instruction frequency, whereas the MediaBench applications have a higher conditional branch instruction frequency with a 90% confidence interval.

4.2.4 Cache Behavior

The last characteristic we are interested in is the cache behavior. The architectural values for the cache sizes were explained in Section 4.1. Table 3 gives the first and second level cache accesses and miss ratios. The last row gives the average access numbers and miss ratios. The table shows the significant difference in the miss ratios for NetBench and Mediabench applications.

We have also studied the cache behavior of NetBench applications with different level 1 cache sizes. The results are not presented due to lack of space. Our experiments reveal that instruction cache miss ratios are more effected by the cache size. For NetBench applications, the processor with a 4 KB L1 cache size have an average miss ratio is 2.8% for instruction cache, and an average data cache miss ratio of 2.6%. For 128 KB cache size the miss ratios are 0.6% and approximately zero for data and instruction caches, respectively.

4.3 Discussion

In the previous subsection, we studied four important characteristics of NetBench and MediaBench applications. In all these categories, NetBench applications had significantly different values than the MediaBench applications. One important property of NetBench applications is its data-intensive nature. As seen in the load/store instruction ratios, the NetBench applications make high number of memory accesses. The MediaBench applications, on the other hand, has more frequent branch instructions resulting in a lower instruction level parallelism.

5. INTEL IXP1200 PERFORMANCE MEASUREMENTS

In this section we give an example of how to utilize NetBench by presenting experimental results with the Intel IXP1200 network processor [6]. We have used the Intel IXP simulator to perform these simulations.

Intel IXP1200 processor is one of the most commonly used network processors. It combines the StrongARM microprocessor with six 32-bit RISC data engines having hardware multithread support that provide a total of 1 giga-operations per second with 200 MHz clock speed [6].

5.1 Simulation Results

In order to simulate the IXP1200, we have converted codes from

Table 3: Number of cache accesses and miss ratios of the NetBench and MediaBench applications. Ratios are given in percentage, il1 stands for first level instruction cache, dl1 stands for first level data cache and L2 is the unified level 2 cache.

NetBench Programs						MediaBench Programs					
Prog.	il1 acc.[M]	il1 miss ratio[%]	dl1 acc.[M]	dl1 miss ratio[%]	L2 miss ratio[%]	Prog.	il1 acc.[M]	il1 miss ratio[%]	dl1 acc.[M]	dl1 miss ratio[%]	L2 miss ratio[%]
crc	242	0.0	72	0.1	9.5	*adpcm*	11	0.0	0.5	0.1	53.7
dh	2745	0.0	1046	0.0	1.4	*epic*	8	0.1	1.7	5.9	7.9
drr	64	0.0	30	1.0	7.4	*g721*	1202	0.0	44	0.1	2.8
ipcha.	85	0.3	20	0.4	2.5	*ghost.*	1448	0.1	290	0.2	6.7
md5	209	0.0	31	0.2	20.9	*gsm*	75	0.0	8	0.1	8.6
nat	23	0.0	8	1.1	14.3	*jpeg*	4	1.1	1.1	0.2	30.8
route	21	0.0	7	0.8	16.9	*mesa*	75	1.7	24	0.7	3.1
tl	15	0.1	5	1.5	12.6	*mpeg*	1839	0.0	405	1.2	18.3
url	196	0.0	46	2.3	2.1	*pegwit*	16	0.1	3.1	7.4	1.4
Avg.	400	0.05	140	0.8	9.7	*Avg.*	519	0.4	86	1.8	14.8

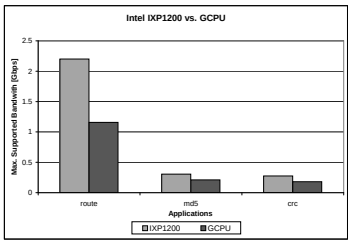

Figure 1: Intel IXP1200 Simulation results.

NetBench applications to Intel IXP Micro-code. We compare the performance of IXP1200 with a general-purpose processor (GCPU), similar to Intel Pentium, having a 1 GHz clock speed, using three applications from NetBench.

To gather information about the general-purpose processor, we again used SimpleScalar simulator [2]. Figure 1 summarizes the results. The figure gives the maximum amount of traffic the processor can handle. This value is calculated by finding the total number of bytes manipulated in the program and dividing this value to the simulated time required to execute the program. Figure 1 illustrates the power of the IXP processor. Although the simulated IXP processor had a clock speed of 200 MHz, it outperformed the GCPU in all programs; by 51% for crc, by 44% for md5, and by 80% for route.

The results also show how NetBench can be utilized. It shows that the IXP is more suitable for route than it is for MD5, because the relative performance improvement over GCPU is much higher with the route application. Also, it gives the maximum amount of traffic the IXP1200 can handle for a given application. Customers can decide whether this supported bandwidth meets their requirements.

6. CONCLUSION

In this paper, we introduced a benchmarking suite for network processors. In spite of the the increase in demand and supply for network processors, there still does not exist a common framework for evaluating them. Many designers still use benchmarks designed for other purposes, such as MediaBench and SPEC2000. We have shown that the applications for network processors are significantly different than the applications for media processors, hence a specific benchmarking suite is a necessity. We have also presented a performance study of a popular network processor.

7. REFERENCES

[1] Baker, F. *Requirements for IP version 4 routers.* Request for Comment: 1812, June 1995.

[2] Burger, D. and Austin, T. *The SimpleScalar Tool Set, Version 2.0*. Technical Report CS-TR-97-1342, University of Wisconsin, June 1997.

[3] Cell Relay Retreat. *CRC-32 Calculation, Test Cases and HEC Tutorial.* http://cell.onecall.net/cell-relay/publica tions/software/

[4] Crowley, P., Fiuczynski, M. E., Baer, J. L, Bershad, B. N. Characterizing Processor Architectures for Programmable Network Interfaces. In *Proc. of International Symposium on Supercomputing*, Santa Fe, NM, 2000.

[5] The FreeBSD Project. *FreeBSD Operating System.* http://www.freebsd.org

[6] Halfhill, T. R. *Intel Network Processor Targets Routers.*. Microprocessor Report, Sep. 13, 1999, vol. 13-12.

[7] International Organization for Standardization. *ISO Information Processing Systems - Data Communication High-Level Data Link Control Procedure - Frame Structure*. IS 3309, October 1984, 3rd Edition.

[8] Kessler, R. *The Alpha 21264 Microprocessor*. In IEEE Micro, 19(2), Mar/Apr 1999.

[9] Lee, C., Potkonjak, M., Mangione-Smith, W. MediaBench: A Tool for Evaluating and Synthesizing Multimedia and Communications Systems. In *Proc. of International Symposium on Microarchitecture, IEEE Micro-30*, 1997.

[10] MMC Networks, Inc. *Leading the Network Processor Revolution.* http://www.mmcnet.com/Solutions

[11] Nemirovsky, A. *Towards Characterizing Network Processors: Needs and Challenges.* XStream Logic Inc., November 2000.

[12] NetBench Web Site. http://istanbul.icsl.ucla.edu/NetBench

[13] The NLANR project. *NLANR Network Traffic Packet Header Traces.* http://moat.nlanr.net/Traces

[14] PMC-Sierra Inc. *URL-based Switching.* http://www.pmcsierra.com, PMC-2002232, Feb 2001.

[15] Rivest, R. *The MD5 Message-Digest Algorithm.* Request for Comment: 1321, April 1992.

[16] RSA Data Security, Inc. *RSA Security Downloads.* http://www.rsasecurity.com/download

[17] Russell, P. *IPCHAINS version 1.3.10.* http://netfilter.filewatcher.org/ipchains

[18] Shreedhar, M., Varghese, G. Efficient Fair Queuing using Deficit Round Robin. In *Proc. of SIGCOMM'95*, Cambridge, MA, 1995, Aug/Sep 1995.

[19] Standard Performance Evaluation Council. *Spec CPU2000: Performance Evaluation in the New Millenium, Version 1.1.* December 27, 2000.

[20] Transaction Processing Council. *TPC Benchmarks.* http://www.tpc.org.

[21] Wolf, T., Franklin, M., CommBench - A Telecommunication Benchmark for Network Processors. In *Proceedings of IEEE International Symposium on Performance Analysis of Systems and Software*, Austin, TX, April 2000.

[22] Woo, S. et al. The SPLASH-2 Programs: Characterization and Methodological Considerations. In *Proc. of International Symposium on Computer Architecture*, June 1995, pg. 24-36.

[23] Yeh, T. H., Patt, Y. Alternative implementations of two-level adaptive branch prediction. In *Proc. of International Symposium on Computer Architecture*, pp.124-134, Queensland, Australia, May 1992.