

The Impact of Traffic Aggregation on the Memory Performance of Networking Applications

Javier Verdú[‡] Jorge García[‡] Mario Nemirovsky[§] Mateo Valero[‡]

[‡] Departament de Arquitectura de Computadors
Universitat Politècnica de Catalunya
Jordi Girona 1-3, Edifici D-6,
Campus Nord, 08034 Barcelona, Spain
{jverdu,jorge,mateo}@ac.upc.es

[§] Tidal Networks Inc.
697 River Oaks Pkwy
San Jose, CA 95134, USA
mario@tidalnetworks.net

Abstract

The trend of the networking processing is to increase the intelligence of the routers (i.e. security capacities). This means that there is an increment in the workload generated per packet and new types of applications are emerging, such as stateful programs. On the other hand, Internet traffic continues to grow vigorously. This fact involves an increment of the traffic aggregation levels and overloads the processing capacities of the routers.

In this paper we show the importance of traffic aggregation level on networking application studies. We also classify the applications according to the data management of the packet processing. Hence, we present the different impacts on the data cache performance depending on the application category. Our results show that traffic aggregation level may affect the cache performance depending on the networking application category. Stateful applications show a significant sensitivity to this impact.

1. Introduction

The trend in packet processing is to build even more intelligence into the router. Security applications are a good example: the sophistication of attack methods is progressing rapidly. One of the main weakness is the inability of the firewall to look

beyond single individual packets while inspecting network traffic (i.e. stateless firewall). However, more complex firewalls are able to detect an attack segmented along several packets of the same or different flows (i.e. stateful firewall). Therefore, we can distinguish two types of applications: stateless applications do not keep track of previous packet processing. Unlike, stateful applications keep track and update information related to the packet that is processed [8]. In fact, there is a variety of different emerging types of stateful applications, such as monitoring, billing, stateful web applications, stateful packet inspection, etc.

The development of network processors (e.g. high performance, programmable devices designed to efficiently execute communication workloads [4]) is focused on overcoming the time constraints of both network line rates and networking applications workloads. Internet traffic continues to grow vigorously close to 100% per year [12] and consequently the traffic aggregation level reflects an incremental trend. These facts involve that between two packets of the same flow, there is an increasing number of packets from an increasing number of different flows. From the stateful application point of view, there are more flow state maintained, involving a memory capacity bottleneck. Memory bottlenecks are already a challenge in the area of network processors (NPs), but in higher layer applications, and specially in state-

ful applications, the effects of memory latency become even more significant [5].

In this paper, we study the impact of the traffic aggregation on the data cache performance of networking applications. We also analyze the sensitivity of the programs depending on the packet processing of the applications. The main result we obtain is that different traffic aggregations may vary the memory behavior and consequently, the application performance. Specially, stateful programs are very susceptible to this variations.

The remainder of this paper is organized as follows. Section 2 provides related work on the evaluation of the different benchmark suites. In Section 3, we present the benchmark selection, the traffic traces obtention, and the methodology to perform the evaluation. The analysis of the traffic aggregation impact is discussed in Section 4. Finally, we conclude with conclusions in Section 5.

2. Related Work

There is a high interest and an ongoing effort in the NP community to define standard benchmarks [10]. However, benchmarking NPs is complicated due to a variety of factors [3]: there are several programming models and languages, wide variety of application domains, and emerging applications that do not yet have standard definitions.

Several benchmarks suites have been published in the NP area: CommBench [16], NetBench [9] and NpBench [7]. Wolf et al. [16] present a set of eight benchmarks called CommBench. It is focused on program kernels typical of traditional routers. The benchmarks are distributed within two groups according whether they are Header Processing Applications (HPA) or Payload Processing Applications (PPA). In this publication, the workloads are characterized and compared versus SPEC benchmarks. Memik et al. [9] present a set of nine benchmarks (although in the available suite there are ten) called NetBench. The authors categorize the benchmarks into three groups, according to the level of networking application: micro-level, IP-level and application-level. The characterization of the workload is compared versus MediaBench programs. Lastly, Lee et al. [7] propose a new set of ten benchmarks called NpBench. It is focused on both control and data plane processing. In this case, the

benchmarks are categorized according to the functionality: traffic management and quality of service group (TQG), security and media processing group (SMG), and packet processing group (PPG). The study of the workloads is compared with the CommBench workloads.

Snort [1], an Intrusion Detection System which is included in the NetBench suite, is the unique application that presents statefulness features. The rest of the benchmarks are stateless packet processing. Moreover, depending on Snort's configuration, the statefulness of the processing may vary.

Nevertheless, as far as we know, no papers have already been published about the influence of the traffic aggregation on the performance of the application and, in particular, its impact on the memory performance.

3. Environment and Methodology

In this section, the simulation environment is presented. We explain which are the reasons that give rise to the benchmark selection. Subsequently we describe the mechanism used to obtain the network traffic traces and the evaluation methodology used to perform the analyses.

3.1. Benchmarks Selection

We can distinguish three types of applications depending on the data management within the packet processing: *no-state* applications are those programs that do not need to search any kind of data related to the packet or connection to be able to perform the packet processing. For example, CRC only needs the IP packet header. There is another category called *stateless* applications. Stateless means that there is no record of previous packet processing and each packet processing has to be handled based entirely on its own information. For instance, IP lookup is a good example of this type of applications. The packet is forwarded with no record of information about previous packet processing. However, there is a difference between stateless and no-state programs. The stateless applications perform a search of some information related to the packet, while CRC does not generate any search to perform the crc checksum calculation. Finally, the third category is the *stateful* applications, that keep track of the state of

Type	App.	Bench. Suite
No-State	AES	NpBench
	MD5	NpBench
Stateless	NAT	NetBench
	Route	NetBench
Stateful	Snort	NetBench

Table 1. Selected Benchmarks

packet processing, usually by setting fields of state related to the flows or connections. For example, TCP termination requires to maintain the state of the TCP flows. The main difference between stateful and stateless programs is that stateful applications may update a variety of fields within the state. Instead stateless programs only requires the value and do not update any information.

Table 1 shows the selected benchmarks. We also can observe the benchmark categories according to the above classification. We select two applications of every type, except stateful category, because Snort is the unique benchmark within this category. Moreover, Snort has been configured with the most stateful configuration as far as we know. We choose the remaining benchmarks selecting those that present a cache behavior similar to the average of every application type [9] [16] [7].

3.2. Traffic Traces

In order to perform a strict comparison among applications from different benchmark suites we cannot use the default traffic traces included in the suites. Obtaining representative network traffic traces always has been an obstacle to overcome. There are several public sites (e.g. NLANR [11], CAIDA [2]) where there are publicly available traffic traces from a wide open range of routers (e.g. MRA, etc.). However, for confidentiality reasons the IP packet addresses of these traces are sanitized [13]. The sanitization of addresses involves the loss of spatial locality of the Internet IP address distribution [6] and it could affect the results of some networking application studies. However Verdú et al. [15] show that the loss of spatial IP address distribution due to sanitization, does not have significant influence on the evaluation of Snort with a stateful configuration. In our studies

the traffic sanitization may present a reduced variation in the results of the stateless benchmarks. However, currently there is no a better way to perform this analysis.

The Snort stateful processing that we use keeps track of the TCP connection state. Hence, the traces must hold packets in the two flow directions. Otherwise, the connection will not follow the TCP protocol and the flow state could not be updated (the packet processing would always generate an alert). Thus, this reduces the number of public traces useful for our studies. We select traces from an OC12c (622 Mbit/s) PoS link connecting the Merit premises in East Lansing to Internet2/Abilene (i.e. MRA traces within the NLANR site). Additionally, as the stateful benchmark is aimed to TCP packets, the remaining protocols (e.g. UDP, ICMP, etc.) do not generate any workload. Therefore, we filter the traffic traces and finally we obtain traffic traces with only TCP packets. Actually, the only consequence of this filtering is the reduction of traffic bandwidth within the trace.

Lastly, for analyzing the traffic aggregation impact, we evaluate the application performance dealing with a variety of traffic traces from links with different bandwidths. As there is no a variety of traffic bandwidths publicly available with bidirectional traffic, we synthetically generate traffic traces simulating different bandwidths. From four original traffic traces of the same bandwidth link, we sanitize them using four sanitizing mechanisms that assure the independency of IP addresses between traces. We set the timestamp of the new trace from the packet timestamp of one of the traces. Then we mix the traces taking the packets sorted with the microsecond timestamp. Finally we obtain a new sanitized traffic trace that represents roughly four times wider bandwidth than the original link.

Figure 1.a shows the traffic aggregation depending on the saturation of the link and the bandwidth link. This relation also depends on the traffic properties. That is, average packet size, type of traffic, etc. In order to understand the theoretical impact of the traffic aggregation on a stateful application we can observe the memory capacity requirements of Figure 1.b. This graph is based on the amount of state per flow that Snort maintains: roughly 200 Bytes. We can see that an OC-48 link at 50% of

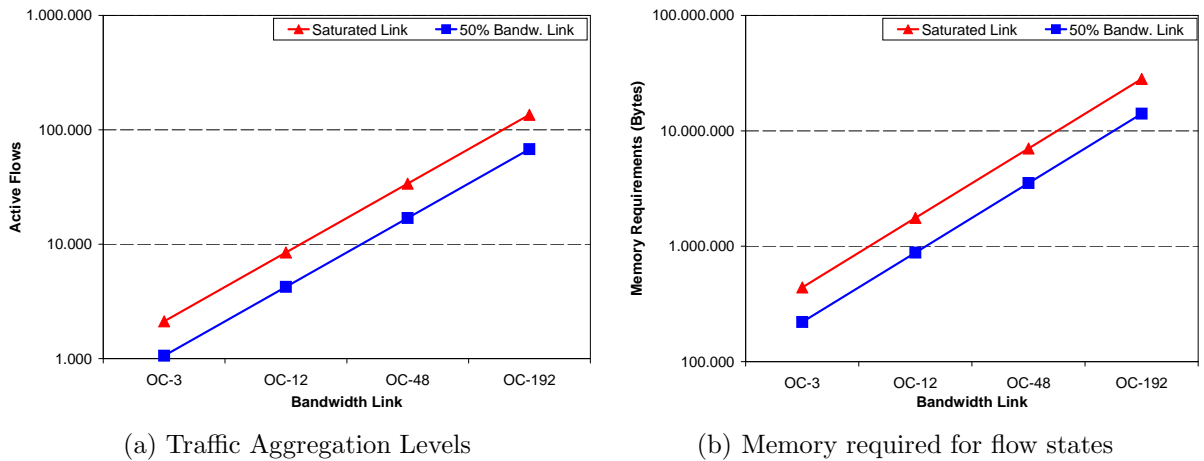


Figure 1. Information related to the bandwidth link

saturation presents roughly 20K active flows. This fact involves that Snort requires almost 4MB towards to flow states.

3.3. Evaluation Methodology

In order to perform the analysis presented in this paper we use a variety of tools. ATOM [14] is used for instrumenting the binary code and generating a trace of dynamic memory accesses. Subsequently, we employ a cache simulator in order to obtain cache statistics per packet and of the overall traffic.

For comparing statistics among the variety of benchmarks, we run every benchmark using the selected traffic traces and processing the same number of packets. Before starting to take statistics, the applications are warmed up with a large enough number of packets, reaching thus the stable behavior of the program. Our studies indicate that a reduced number of packets is usually sufficient. However we extend the warming period up to 10K packets. Additionally, a reduced number of processed packets are enough to obtain real statistics. Therefore the length of the traffic trace is fixed in 50K packets.

4. Traffic Aggregation Impact

In this section we discuss the effects of the traffic aggregation on the memory performance. We

also analyze the reasons of the data miss rate increment.

Figure 2.a shows the miss rate varying L1 data cache size. We can observe three different memory behaviors. Firstly, no-state applications are unaffected by the traffic aggregation. The packet processing is computational intensive without searching data related to the flow IP addresses of the processed packet. Moreover, the miss rate is reduced and a reduced cache obtains very similar performance than larger cache sizes. Nat and Route applications show different behaviors. In this case, packet processing is focused on generating a result from the search of a data related to the packet. For example, Route searches the IP destination address of the packet in order to do the IP forwarding. When the traffic aggregation level grows, the variety of searching is wider and the data table (e.g. lookup table) is larger. However, the information of every item is still small. Due to this, the traffic aggregation involves an increment of data miss rate, although this effect is not very significant. Finally, Snort shows considerably higher miss rates, emphasized when the traffic aggregation level grows. The data management differs in the other selected benchmarks. The packet processing requires the state of the flow in order to update a variety of fields. This stateful benchmark shows a reduction of miss rate with larger cache sizes, although the differences according to the traffic aggregation are maintained arisen.

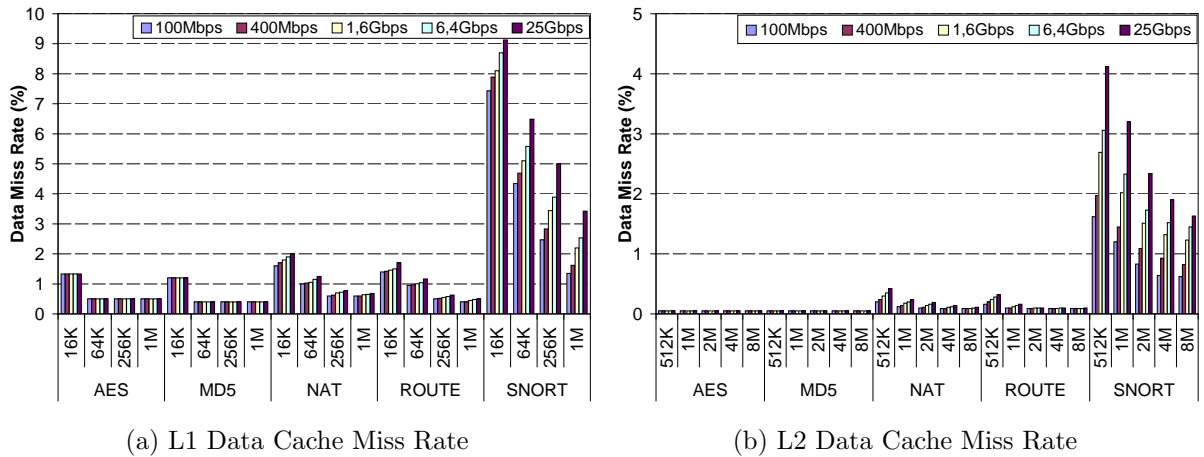


Figure 2. Traffic aggregation impact on data cache

Although larger L1 data caches reduce the L1 data miss rate, the L2 miss rates (i.e. number of L2 misses / number of memory accesses) impact takes place regardless of L1 cache size. Reduced L1 caches increase the amount of accesses to L2 cache. However, as there is a considerable quantity of variables shared between independent packets, the L2 data miss rate is reduced. Instead, using larger L1 data caches involves a reduction of traffic between L1 and L2 caches. In this case the majority of accesses to L2 are toward to flow state data. Due to capacity constraints, likely L2 cache is unable to maintain the requested flow state data and the miss rate increases. Summarizing, comparing the L2 miss rates using 16KB or 1MB L1 data cache, shows no significant variations. Due to this, the results of Figure 2.b are independent of L1 data cache size.

From the analysis of L2 miss rates (varying L2 data cache size), we can conclude that the three behavior distinctions are maintained. The applications that manage no state related to the packet flow are completely insensitive to the traffic aggregation level. The stateless programs show a reduced impact on the miss rate. In order to maintain the same L2 miss rate as obtained with a traffic of 100Mbps, when we run under 25Gbps bandwidth, we need to duplicate the L2 cache size. Snort exhibits the most significant impact from traffic aggregation. There are important differences according to the traffic bandwidth that are maintained

even though L2 is larger.

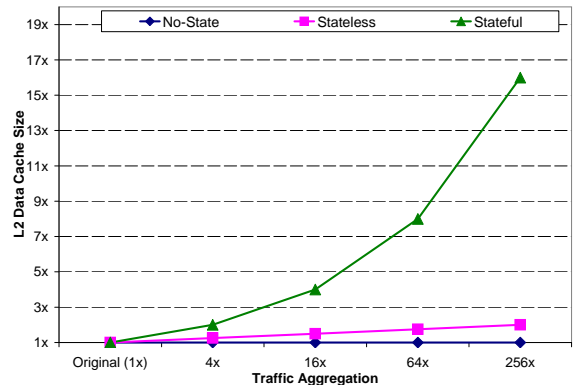


Figure 3. Data Cache Behavior

The Figure 3 shows the relation between the traffic aggregation and the L2 cache size required to maintain the miss rate. The x-axis shows the traffic aggregation level. The Original(1x) is relative to any level. The effects are the same for 100 Mbps or any other bandwidth link. The y-axis shows the increment of 512KB L2 cache size. We can observe that no flow state applications are completely independent of traffic aggregation. Instead, stateless and stateful applications show different relations. Stateless applications, such as IP forwarding, present a linear increment. Unlike stateful applications that show a more than lin-

ear increment. However the stateless and stateful trends depend on the amount of information or state related to the packet of flow respectively.

5. Conclusions

In this paper, we compare the data cache behavior running different networking applications with a variety of traffic traces with different traffic aggregation levels. We categorize the benchmarks according to the management of information related to the processed packet: no-state, stateless, and stateful applications.

We show the importance of doing research in networking applications using traffic traces with different aggregation levels. The traffic aggregation level may affect the cache behavior and, consequently, the application performance. Depending on the application category, the impact is different. The no-state applications are unsensitive to the traffic aggregation. Unlike, stateless programs are susceptible to increment the data cache miss rate. However, stateful applications are the most sensitive to the traffic aggregation level.

In the stateful applications, as they need to maintain a state related to the flow, as traffic aggregation increases, the memory capacity requirements significantly grows. Additionally, when there are more active flows, the temporal locality of flow state is reduced and then, the data cache miss rate significantly increases. Therefore, the main conclusion of this paper is that the impact of the traffic aggregation depends on the application itself and its data structures distribution.

Acknowledgements

This work was supported by the Ministry of Science and Technology of Spain under contract TIC-2001-0995-C02-01, and grant BES-2002-2660 (J. Verdú), CEPBA, and the HiPEAC European Network of Excellence. The authors would like to thank Ayose Falcón for his comments and review inputs of this work.

References

- [1] J. Beale, J. C. Foster, J. Posluns, and B. Caswell. *Snort 2.0 Intrusion Detection*. Syngress Publishing Inc., 2003.
- [2] Cooperative association for internet data analysis. www.caida.org.
- [3] P. Chandra, F. Hady, R. Yavatkar, T. Bock, M. Cabot, and P. Mathew. Benchmarking network processors. In *Proc. NP1, Held in conjunction with HPCA-8*, Cambridge, MA, USA, Feb. 2002.
- [4] P. Crowley, M. A. Franklin, H. Hadimioglu, and P. Z. Onufryk. *Network Processor Design: Issues and Practices*, vol. 1, chapter Network Processors: An Introduction to Design Issues. Morgan Kaufmann Publishers, USA, 2002.
- [5] P. Crowley, M. A. Franklin, H. Hadimioglu, and P. Z. Onufryk. *Network Processor Design: Issues and Practices*, vol. 2, chapter Network Processors: Themes and Challenges. Morgan Kaufmann Publishers, USA, 2003.
- [6] E. Kohler, J. Li, V. Paxson, and S. Shenker. Observed structure of addresses in IP traffic. In *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurement workshop*, PA, USA, 2002.
- [7] B. K. Lee and L. K. John. Npbench: A benchmark suite for control plane and data plane applications for network processors. In *Proc. of ICCD*, San Jose, CA, USA, Oct. 2003.
- [8] S. Melvin, M. Nemirovsky, E. Musoll, J. Huynh, R. Milito, H. Urdaneta, and K. Saraf. A massively multithreaded packet processor. In *Proc. of NP2, Held in conjunction with HPCA-9*, CA, USA, 2003.
- [9] G. Memik, W. H. Mangione-Smith, and W. Hu. Netbench: A benchmarking suite for network processors. In *Proc. of ICCAD*, CA, USA, 2001.
- [10] A. Nemirovsky. Towards characterizing network processors: Needs and challenges. November 2000. Xstream Logic Inc., white paper.
- [11] National lab of applied network research. <http://pma.nlanr.net/Traces>.
- [12] A. M. Odlyzko. Internet traffic growth: Sources and implications. In *Optical Transmission Systems and Equipment for WDM Networking II*, B. B. Dingel, W. Weiershausen, A. K. Dutta, and K.-I. Sato, eds., *Proc. SPIE*, vol. 5247, Sep. 2003.
- [13] R. Pang and V. Paxson. A high-level programming environment for packet trace anonymization and transformation. In *Proceedings of the ACM SIGCOMM Conference*, Karlsruhe, Germany, 2003.
- [14] A. Srivastava and A. Eustace. ATOM - A system for building customized program analysis tools. In *Proc. ACM SIGPLAN Conf. on Programming Language Design and Implementation*, June 1994.
- [15] J. Verdú, J. García, M. Nemirovsky, and M. Valero. Analysis of traffic traces for stateful applications. In *Proc. of NP3, Held in conjunction with HPCA-10*, Madrid, Spain, February 2004.
- [16] T. Wolf and Mark A. Franklin. Commbench - a telecommunications benchmark for network processors. In *Proc. of ISPASS*, TX, USA, 2000.