Traffic Locality Characteristics in a Parallel Forwarding System

W. Shi, M. H. MacGregor and P. Gburzynski

Department of Computing Science, University of Alberta, Edmonton, AB, T6G 2E8, Canada
{wshi, macg, pawel}@cs.ualberta.ca

SUMMARY

Due to the widening gap between the performance of microprocessors and that of memory, using caches in a system to take advantage of locality in its workload has become a standard approach to improve overall system performance. At the same time, many performance problems finally reduce to cache performance issues. Locality in system workload is the fact that makes caching possible. In this paper, we first use the reuse distance model to characterize temporal locality in Internet traffic. We develop a model that closely matches the empirical data. We then extend the work to investigate temporal locality in the workload of multi-processor forwarding systems by comparing locality under different packet scheduling schemes. Our simulations show that for systems with hash-based schedulers, caching can be an effective way to improve forwarding performance. Based on flow-level traffic characteristics, we further discuss the relationship between load-balancing and hash-scheduling, which yields insights into system design. Copyright © 2000 John Wiley & Sons, Ltd.

KEY WORDS: Internet traffic; traffic locality; traffic scheduling; hash-based scheduling

1. INTRODUCTION

Simulation is an important tool for understanding the Internet due to its size, complexity, heterogeneity and fast-evolving nature. Workload characterization is critical in simulation study in that capturing the salient features in the workload enables us to experiment with design alternatives with accuracy. Identifying the characteristics that affect system performance is the first step toward successful simulation and the basis for making sound design decisions.

In addition, workload models are useful in generating artificial workloads. This is desirable for many reasons. For one thing, for a system in the design phase, a real workload may not be available. Second, due to various limitations, recording Internet traffic is a difficult task. The recorded traffic is called a trace, and is a fixed snapshot of the traffic passing through a network device during the measurement period. Synthetic workload generation is desirable even if traces are available because synthetic traces can be created with a controlled range of parameterized characteristics, such as locality.

In this paper, we first develop a workload model that captures locality in Internet traffic. Specifically, we are interested in characterizing the temporal locality in address traces. These
are simply the sequences of destination addresses of IP packets seen passing the measurement point.

Temporal locality of network address traces refers to the phenomenon that if an address is referenced, it is likely to be referenced again in the near future. The reason for the existence of this temporal locality lies in the fact that packets with the same destination tend to be transmitted closely in time, usually as the result of transmission of data that are segmented into a sequence of packets [1]. Significant temporal locality makes caching appealing in the design of forwarding systems, especially as the performance gap between microprocessors and main memory continues to widen. For example, IP routing table lookup in Internet routers can benefit from even simple “destination address, output port” caching [2].

1.1. RELATED WORK

The LRUSM (Least Recently Used Stack Model) [3] was studied by researchers in the context of program page reference behavior. Imagine a stack as a one-dimensional array containing all possible addresses, each of them a single array element. When an address is referenced, the array (stack) index of the address is output as the stack distance. All the addresses above this value are moved down by 1 position and the address just referenced is put at position 0 of the array, that is, at the top of the stack. This is equivalent to using the LRU model to update the stack, hence the name “LRU Stack.” In the LRUSM, each entry in the address trace produces a stack distance. Corresponding to an address trace, there is a sequence of stack distances, which is called the “distance string”. As an example, a distance string can look like “38, 1, 0, 143, 1, 162, 1, 0, 40, 97, 1, 150, 63, 311, 80, 312, 1, 3, 0, 313, 127”. The “0”s in the string indicate that the address at the top of the stack, which was just referenced, is referenced again; “1”s mean that the address referenced just prior to the last reference (now at position 1 of the stack) is referenced again, and so on.

The LRUSM has been used in contexts other than program memory references. For example, it is used in [4] and [5] to characterize temporal locality and to generate representative URL references. Although the details and contexts vary, our approach in modeling temporal locality in IP destination address traces is similar to these works.

Ref. [6] compares the effectiveness of different hashing schemes for network address lookup. It is mentioned that the high-order information bits of the addresses can be used as a load balancing function to distribute the work load among different routing engines (RE’s). When a packet arrives, its destination address is hashed to yield the forwarding engine to which the packet should be dispatched. Here the destination IP address is the key and the number of processors is the hash table size. Each entry of the hash table contains a sub-table which is the memory hierarchy which contains the routing table. In this paper we study the workload characteristics under different scheduling schemes in the context of parallel network processor systems.

Ref. [7] investigated load-balancing for hash-based traffic-splitting schemes. A table-based hash scheme is shown to perform as well as the CRC-based one, and is also an effective approach to achieve load-balancing.

Ref. [8] proposes the request-distributing scheme HRW (Highest Random Weight) in the context of WWW object retrieval in a proxy server cluster environment. It is shown that name-based mapping is an effective way of achieving good hit rate in proxy server caches. Given the huge difference in response time in the cases of a cache miss and hit, it is justified
that the scheme balances load on the proxy servers in terms of objects rather than requests. Using simple hashing functions to distribute packets, our work shows similar situations exist in routing.

Among the commercial network processor products, the IBM PowerNP [9] can be used with a “load balancer”. The load balancing algorithm discussed in [10] is a table-based hashing approach. To distribute the workload onto parallel network processors (NP’s), the load balancer groups traffic flows into fine-grained bundles using information on the input queue length of the NP’s. It also needs per packet timing information and per-bundle statistics to do the scheduling. The load balancer guarantees that the packets of a particular IP flow are passed to only one network processor. Ref. [11] proposes an adaptive load-balancing mechanism based on the processor mapping described in [8]. The adaptive loop is necessary because even if the flows are distributed onto the processors evenly, imbalance can still occur because packet distributions in the flows may be different.

1.2. Reuse Distance Model

We use reuse distance which is equivalent to the stack distance in [3] as the measure of recency of IP destination addresses references. Basically, the reuse distance of an address is the number of unique addresses referenced after its previous appearance.

Let “r1, r2, ..., rn, ...” represent the corresponding reuse distance sequence of an address sequence “a1, a2, ..., an, ...”. The reuse distance sequence of an address sequence can be generated using the LRU stack [3]. Imagine a stack that contains all possible IP destination addresses, one address per entry. The index of the stack increases from top (0) to bottom. Each time an address is referenced, the index of the stack entry that contains the address is output as the reuse distance and the address is removed and pushed onto the top of the stack.

Let Ai be the set that contains the unique addresses in the address sequence a1, a2, ..., ai and |Ai| the size of Ai. Then |Ai| is the largest reuse distance seen in the trace up to and including ai. Therefore, we have

\[ r_{i+1} = \begin{cases} \text{idx}(a_{i+1}) & \text{if } a_{i+1} \in A_i \\ |A_i| + 1 & \text{if } a_{i+1} \notin A_i \end{cases} \]  

where idx(ai) yields the index of ai in the LRU stack.

2. LOCALITY CHARACTERIZATION FOR IP DESTINATION ADDRESS TRACES

By investigating the reuse pattern of IP destination addresses in real world traces, we try to develop a model that fits the empirical data well. We have used the following two traces in our initial experiments:

UofA is a 1 million-entry IP packet trace recorded using “tcpdump” at University of Alberta.

LDestIP is a trace of approximately 31.5 million entries from the NLANR (National Laboratory for Applied Network Research) Passive Measurement and Analysis project [12].

The real traces are initially used to validate parametric models of locality in streams of IP addresses. Once we have these parametric models, we can generate synthetic traces to test the performance of, for example, caching schemes exposed to traffic of varying levels of locality. In
the paper, we use the correspondence between the miss rates of the real and synthetic traces in the IP cache simulations as a further validation that synthetic traces generated using our traffic model really capture the behavior of the real traces.

2.1. Reuse Distance Pattern

The patterns of reuse distances calculated using Eq.1 are shown in Fig. 1 for the first ten thousand IP destination addresses of the two traces. The “virtual clock” in each figure is the address counter and is incremented by one each time an address is processed.

Two common features shared by the traces stand out:

- The reuse distances of the first appearances of the addresses form an upper border line.
- Under the border line, smaller reuse distances have higher density than larger ones.

The first common feature is the footprint curve [13]. The upper border lines in the two figures are the footprints of the two traces. These characterize the “number of unique addresses observed so far” as the length of the trace increases. This phenomenon is due to the definition of reuse distance in Eq. 1 and is the result of “cold start” in the measurement.

The second common feature observed in Figs. 1 is the result of temporal locality. Small reuse distances are dominant, indicating that most references are made to addresses that appeared shortly before. Once the distribution of reuse distance is characterized for the empirical data, we will be able to say how likely it is that an address just referenced will reappear in a given number of steps in the future.

2.2. Reuse Distance Model

We choose the CCDF (Complementary Cumulative Distribution Function),

\[ S(x) = Pr[X > x] = 1 - F(x) \]  

of reuse distances to characterize temporal locality.

One reason for fitting the CCDF instead of the CDF (Cumulative Distribution Function) is that the probability of the occurrence of larger reuse distances can be more accurately captured [3]. This is based on the assumption that addresses with larger reuse distances are more important in evaluating the performance of the system in that the larger the reuse distance, the more likely the address causes a cache miss. Furthermore, fitting the CCDF makes performance measurement straightforward. That is, given the size of a fully associative cache, the miss ratio of the synthetic trace can be predicted directly from the CCDF curve. No simulation is needed.

Fig. 2 shows the reuse distance CCDF’s for the LDestIP and UofA traces. Each curve represents a CCDF calculated from a sub-trace of a certain length. All sub-traces are longer than 10,000 entries.

Evidently, the CCDF’s follow a general pattern. The curves overlap at the beginning; before the point where the left-most curve starts to diverge, the curves are nearly identical. This consistency across different lengths of traces implies statistical equilibrium. Over 60 percent of reuse distances fall in this range. Afterward, curves for different lengths diverge. We call these parts of the curves the tails. These tails also have a pattern. Generally, they start by segments of roughly straight lines, though there are bumps in the middle of these segments,
then drop off as nearly vertical straight lines. The diverging tails give the look of branches from a common trunk.

Measurements show that the final segments, i.e., the almost vertical parts, of the tails of the CCDF’s in Fig. 2 are linear. This linearity is due solely to the introduction of new addresses. The addresses causing the tails are not “reused” but appear for the first time in the traces.

It is apparent that using a longer trace to calculate the CCDF removes the linear tail of the CCDF for a shorter trace, although this introduces another longer linear tail. The reason is that a longer trace makes the reuse distances caused by the first appearances in a shorter trace more frequent than just once.

The phenomenon that linear branches stem from a common base in all the CCDF curves reflects instability due to the limited length of the traces. In other words, if we had traces long enough, there would be no linear tails in the CCDF’s and they would converge to one common shape. By “long enough”, we mean that all possible addresses appear enough times to eliminate the inaccuracy due to lack of samples in the resulted model. One indication that a trace is long enough would be that one cannot find another trace that is longer and produces a different CCDF tail. From available measurement data, which is not big enough, we have to observe and analyse patterns and predict trends, which is one of the basic ideas of modeling.

Given the above observations, our approach to fitting the CCDF of a trace is to use the longest trace available, and remove the linear tail first if there is one.

It was shown in [4] that the distribution of reuse distances for URL’s in Web access traces can be fitted by the Lognormal distribution. However, we have not been able to fit the CCDF’s shown in Fig. 2 using a single distribution. Instead, we use the sum of two distributions:

$$C(x) = pW(x) + (1 - p)P(x), \quad 0 < p < 1,$$

(3)

where $W(x)$ is the CCDF of the Weibull distribution

$$W(x) = e^{-(x/d)^c}, \quad c, d > 0,$$

and $P(x)$ is the CCDF of the Pareto distribution

$$P(x) = (1 + bx)^{-a}, \quad a, b > 0.$$

Fig. 3 shows the fitting of the W+P (shorthand for “Weibull + Pareto”) function to the real data, where the linear segments have first been removed.

Experiments show that the W+P fit predicts the decay of the tail well. Identifying the decay trend of the tail is important because it makes the model more useful. In our case, it helps to generate longer and more realistic synthetic traces. This is the reason that the combination of Weibull and Pareto is chosen. We have tried to fit the CCDF data with other combinations, for example, two Weibull distributions (changing the $P(x)$ in (3) to a Weibull CCDF). The result actually shows better visual fit than the W+P scheme. However, further experiments show that for longer traces, CCDF’s for synthetic traces based on the two-Weibull fitting have fewer unique addresses and they have tails that decay more quickly than the real traces. We believe that the tails in the finite-length traces decay more quickly than predicted by the W+P fit precisely (Fig. 3) because the former are finite in length, so that large reuse distances are necessarily under-represented.

Recently, the distributions of many important parameters in workload models for communication networks have been found to be “heavy-tailed” or “long-tailed”, meaning that
the tails of the CCDF’s of the distributions decay slower than exponentially. Empirical data are frequently better fitted by such distributions like the Pareto, Weibull, or Lognormal. According to the definition given by [14], both the Pareto distribution and the Weibull distribution with shape parameter $c$ less than 1 are long-tailed. On the other hand, [15] showed that the Lognormal distribution is not long-tailed.

After eliminating the linear segments, we have been able to fit the CCDF’s with mixtures of Weibull and Pareto CCDF’s. The shape parameters for the Weibull’s are 1.14 for the LDestIP trace and 1.22 for the UofA trace. Neither satisfies the requirement defined in [14] for being long-tailed. The Pareto segments, however, are long tailed and the shape parameters for the UofA and LDestIP traces are 0.653 and 0.554.

It is generally not easy to tell whether a parameter is long-tail distributed or not by merely obtaining visually good fittings to some specific samples. For example, [16] shows that fitting different samples of certain measures yields different best-fit tail distributions. In our case, the available traces are not long enough to observe a stable shape of the CCDF, which makes it harder to predict the exact tail behavior.

2.3. Model Validation

We validate the model described by Eq. 3 by fitting it to the reuse distance CCDF’s of more traces. So far, we have experimented with 85 traces from two more sets from NLANR [12]. These sets are

Abilene-I Two hours contiguous bidirectional packet header traces over two OC48c Packet-over-SONET links collected at the Indianapolis router node (IPLS). We used all the 48 traces from this set,

Auckland-IV A 45 days continuous trace collected at the University of Auckland Internet access link by the by the Waikato Applied Network Dynamics (WAND) research group between February and April 2001. We retrieved and fitted the first 37 out of 94 traces from this set.

The traces are divided into groups by their directions. The Abilene-I set contains traces measured at two router ports. So there are 4 groups: CLEV-0, CLEV-1, KSCY-0, KSCY-1, each containing 12 traces. The Auckland-IV set contains traces from two directions at one port, thus two groups. They are AuckIV-0 (19 traces) and AuckIV-1 (18 traces). The reuse distance CCDF’s of these traces are shown in Figs. 4 and 5. The CCDF’s for the KSCY traces are similar to those for the CLEV traces and are not shown.

During experiments, we have found that temporal locality characteristics of traces of packets traveling in the same direction, i.e., arriving at or departing from a measurement port, are very similar. For example, using the fitting results of one trace as initial values, we have been able to automate the fitting procedure to fit all the other CCDF’s of traces in the same group. This is especially true for the traces in the Abilene-I set where one set of parameters was used as initial values to obtain the fitting of all the CCDF’s, even those of traces from different groups. This is true too for the AuckIV-1 group and most of the traces (17 out of 19) in the Auck-0 group. The closeness of the CCDF curves in the figures indicates that temporal locality features of the backbone traces are relatively consistent where those of the lower-bandwidth links have more variation.

As shown in Figs. 6 and 7, overall, our model has been successful in describing the temporal locality characteristics of a wide range of traces gathered at different levels from campus networks to Internet backbones.
We have also found that the parameters in the model generally differ for different groups of traces. We are especially interested in the parameter “p” in Eq. 3, which represents the percentage that the Weibull contributes to the mixed-CCDF model. The values of p are in the ranges of [0.62,0.90] and [0.37,0.65] for the Auck-0 and Auck-1 traces, respectively. They are in the range of [0.14,0.16] and [0.16,0.19] for the CLEV-0 and CLEV-1 traces. It seems that p tends to be larger for a campus level network but smaller for backbone networks. The p values for the UoF and LDestIP traces collected at campus-level networks and those for the KSCY-0 and KSCY-1 traces gathered at backbone networks support this hypothesis. Figs. 8 and 9 show the effects of p by decomposing the fitted CCDF’s for some traces into two components of the model, pW(x) and (1 – p)P(x).

The other parameters that differ significantly across trace sets are the scale parameters, i.e., d for the Weibull and b for the Pareto CCDF, b for the AuckIV set is in the range of [1.21,253] and for the IPLS set is within [0.059,0.154], d is within [6.23,47.0] for the AuckIV set but [1067,1325] for the IPLS set. Both scale parameters differ by more than one order of magnitude, respectively. On the other hand, the shape parameters, i.e., c for the Weibull and a for the Pareto, are relatively constant for different trace sets.

Thus the set of values of p, b, d, or even only one of them, can be used to pinpoint a trace’s origin. Moreover, we can generate synthetic traces using different parameter values for the evaluation of network devices at different levels.

3. IMPACT OF SCHEDULING SCHEMES ON LOCALITY

Armed with the workload model developed in section 2, we investigate locality in the workload in parallel forwarding systems. Fig. 10 shows such a system with four processors. We call these processors routing engines because one of their major tasks is to do the IP address lookup in the routing table to find the output port for the incoming packet. The caches in the figure are simple route caches that contain entries of (destination IP address, output port) pairs.

The scheduler’s job is to find the next available routing engine and deliver the next packet in the queue to it. We are interested in the impact that different packet scheduling schemes have on temporal locality in such systems.

In the following discussions, we compare the temporal locality in two traces using their re-use distance CCDF’s. Trace i is said to have better temporal locality than trace j if \(CCDF_i(x) < CCDF_j(x)\) for all \(x > 0\). Two scheduling schemes are considered:

**Round-Robin (RR)** Packets are scheduled to RE’s in a round-robin fashion. This happens when the per-packet processing time is fixed and the same for all the RE’s. The scheduler simply delivers the next packet to the first available RE. However, with caches, the packet dispatching may not be strictly round-robin since per-packet processing time is no longer fixed and the same. We use round-robin scheduling mainly for comparison purposes.

**Hashing-based** The destination address is used as the message and a hash function is applied to it to yield the index of the RE. For the particular hash function, we examined the simple Fletcher checksum [17] bits used in the Internet protocols and the 16-bit CRC (Cyclic Redundant Checksum), as both have been shown to be good hash functions for both
address lookup and load balancing.

3.1. General Impact of Scheduling Schemes on Locality

Fig. 11 shows the CCDF’s for a 4-RE system driven by the UoF and LDestIP traces. Fig. 12 shows the results for the LDestIP trace in a 16-RE system. The notations used in these figures are:

- $CCDF_{agg}(x)$ is the CCDF of the unscheduled, aggregate traffic.
- $CCDF_{rr1}(x)$ is the CCDF of the traffic processed at the first RE, under Round-Robin.
- $CCDF_{cksm1}(x)$ is the CCDF of the traffic processed at the first RE, under checksum hashing.
- $CCDF_{crc1}(x)$ is the CCDF of the traffic processed at the first RE, under 16-bit CRC hashing.

The overall patterns of the CCDF’s at other RE’s under a given scheduling scheme are very similar to that given for the first RE. Since the results are similar for the checksum and CRC hashing functions we will only discuss the results for the checksum function.

Fig. 11 shows the impact that the two scheduling schemes have on the locality of scheduled traffic. The CCDF’s can also be seen as the curves of miss ratio versus cache size with simple destination route caches [2]. The workload at each RE under hashing has much better temporal locality and thus better cache performance than for both RR and the original aggregate traffic. For example, for the UoF trace, with 50 cache lines, the miss ratio for cksm1 is 0.178755, less than one third of that for rr1, 0.568349. With larger caches or more processors, the difference between the two disciplines is larger.

It is evident that the RR-scheduled traffic has less temporal locality and hashing-scheduled traffic has more temporal locality than the original unscheduled traffic. Hashing divides the input address set and each RE sees less addresses than when there is no hashing. Some addresses between two successive instances of an address are mapped onto other RE’s. This increases temporal locality in that when the first instance appears, the second will be referenced sooner in the future than when there is no hashing. RR tends to disperse packet trains. The successive instances of an address are likely to be mapped onto different FE’s. From the perspective of an RE, the number of distinct addresses between two instances of an address increases. Thus, temporal locality is decreased compared with that in the unscheduled traffic.

The above observations on hashing are abstracted and proved in [8] as the “Partitioning Non-Harmful” theorem which says that the expected hit rate in a partitioned mapping (e.g., hashing) is greater than or equal to that in a non-partitioned mapping (e.g., round-robin).

3.2. Impact of Scheduling Schemes on Per-Processor Locality

Besides the improvement in temporal locality with hash-scheduling, we observed that the hash-dispatched workload in terms of packets is not the same for all processors in a parallel forwarding system. In this section, we explain the problem qualitatively. For hashing schemes, we use CRC as an example.

As shown in Table I, under either CRC-hashing or Round-Robin, each RE sees a similar number of flows, although overall, the number of flows seen by an RE under hashing is...
TRAFFIC LOCALITY CHARACTERISTICS IN A PARALLEL FORWARDING SYSTEM

<table>
<thead>
<tr>
<th>RE</th>
<th>No. of Flows</th>
<th>No. of Packets</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1473</td>
<td>385801</td>
</tr>
<tr>
<td>2</td>
<td>1436</td>
<td>174544</td>
</tr>
<tr>
<td>3</td>
<td>1525</td>
<td>213375</td>
</tr>
<tr>
<td>4</td>
<td>1427</td>
<td>226273</td>
</tr>
</tbody>
</table>

Hashing

<table>
<thead>
<tr>
<th>RE</th>
<th>No. of Flows</th>
<th>No. of Packets</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4282</td>
<td>249998</td>
</tr>
<tr>
<td>2</td>
<td>4268</td>
<td>249999</td>
</tr>
<tr>
<td>3</td>
<td>4303</td>
<td>249998</td>
</tr>
<tr>
<td>4</td>
<td>4258</td>
<td>249998</td>
</tr>
</tbody>
</table>

Round-Robin

Table I. No. of Flows vs No. of Packets Seen at Each RE (UofA Trace, 4 RE's)

significantly smaller than that under RR. The total number of flows in the UofA trace is 5861. Under RR, the numbers of packets seen at the RE's are the same. However, under hashing, the number of packets seen at RE1 is more than twice that seen at RE2. In other words, the load under RR is perfectly balanced but skewed under hashing. The problem is that although hashing divides the number of flows almost evenly among RE’s, due to the difference in flow rates, the numbers of packets processed by different RE’s can differ from each other.

Ref. [18] classes Internet traffic flows into alpha and beta traffic, where alpha traffic “is caused by large file transmissions over high-bandwidth links and is extremely bursty” and beta traffic is caused by file transmission over low-bandwidth links. When an alpha flow exists in the workload, under hashing, all packets of that flow will be dispatched to one particular RE. There are relatively few alpha flows compared to the number of beta flows, but when one alpha flow is scheduled to an RE, this RE has many more packets to process than another RE processing only short-lived and low-volume beta flows.

In a system where no cache is used, the observed skewness in workload distribution for the processors creates a load imbalance which can significantly reduce the system utilization and overall performance. However, as will be discussed in the rest of this section, per-processor locality measurements show that better temporal locality exists in the workload for the most loaded processor.

Fig. 13 shows the CCDF curves for the workload for each processor in a 4-RE system. They are plotted on a log-log scale to emphasize the differences between the curves under either scheduling scheme, CRC or Round-Robin. The Round-Robin curves (RR0-3) do not show noticeable difference from each other. However, the CRC curves differ from each other, with CRC0 seemingly in its own class. The other three curves under CRC, i.e., CRC1-3, are much closer to each other, but still distinguishable, for example, CRC3 is constantly lower than the other two.

Generally speaking, under hash-scheduling, it seems that the workload for heavier loaded RE’s has better temporal locality than that for relatively lightly loaded ones. This is consistent
with the results shown in Table I and Fig. 13. The implication is that under hash-scheduling, caching is not only effective in improving overall forwarding performance, but is also helpful in mitigating load-imbalance as a result of hashing. In other words, with a cache taking advantage of locality differences in the workload, the more heavily loaded an RE, the more efficient it becomes.

4. SIMULATIONS

Based on the discussion in the previous section, we can expect that in a parallel forwarding system with a hash-based scheduler, caching would be an effective way to improve system performance. Moreover, difference in temporal locality in per-processor workload indicates that caching could also be helpful in mitigating load imbalance. In this section, we describe the simulations we conducted to verify these ideas. Simulations are simplified by the assumption that only routing table lookup operations are performed by the RE’s. It is further assumed that the system has an infinite buffer that stores the incoming packets. Finally, the cache replacement algorithm is assumed to be LRU.

4.1. Metrics

We first consider system throughput $T$ as the metric which is measured in terms of number of packets forwarded during some unit time period:

$$ T = \frac{N}{T_d(p_N) - T_e(p_i)} $$

where

- $N$ is the number of packets,
- $T_d()$ returns the time when a packet is dequeued,
- $T_e()$ returns the time when a packet is scheduled for lookup,
- $p_1, p_2, ..., p_N$ are the packets in their arrival order.

The cost of a route lookup for a destination IP address depends on the cache state. When the route is in the cache, it takes $T_h$ to finish the lookup. When the route is not in the cache, the time is $T_m$ which includes $T_h$ plus the cache miss penalty. All the variables measuring time will be expressed in terms of $T_m$ and $T_h$. $T_m / T_h$ is usually much larger than 1. For example, for the BBN MGR ([19]), it is at least 5. As the speed gap between off-chip memory and CPU widens, this ratio will become much larger. For example, in [20], it takes the $\mu$Engine 30 cycles to transfer a word both to and from memory. Even with hardware assistance, it takes 30 cycles to finish an IP lookup.

To measure the mitigating effect of cache on load imbalance, we use $B_i$ to represent the total busy time of the $i$th RE. The Coefficient of Variation (CV) of busy time is a measure of the effectiveness of load balancing:

$$ CV = \frac{\sigma_B}{\mu_B} $$

where $\sigma_B$ is the standard deviation and $\mu_B$ is the mean.

CV is chosen for its independence from the units of data. It is a measure of the combined effect of the traffic being skewed toward a few flows and the system’s ability to balance the
load. With the simulation input fixed, CV measures the latter. In the ideal case where each RE has exactly the same load, $\sigma_B$ and $CV$ would both become zero.

### 4.2. Results

<table>
<thead>
<tr>
<th>Cache Size (Entries)</th>
<th>Performance($P_{kts}/T_h$)</th>
<th>Cache Size (Entries)</th>
<th>CV</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>UofA</td>
<td>LDestIP</td>
<td>UofA</td>
</tr>
<tr>
<td>0</td>
<td>0.431999</td>
<td>0.500651</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0.606034</td>
<td>0.708440</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>0.727566</td>
<td>0.773254</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>0.867115</td>
<td>0.843081</td>
<td>4</td>
</tr>
<tr>
<td>8</td>
<td>0.956509</td>
<td>0.921658</td>
<td>8</td>
</tr>
<tr>
<td>16</td>
<td>1.040634</td>
<td>1.038129</td>
<td>16</td>
</tr>
<tr>
<td>32</td>
<td>1.227534</td>
<td>1.229992</td>
<td>32</td>
</tr>
<tr>
<td>64</td>
<td>1.481491</td>
<td>1.602327</td>
<td>64</td>
</tr>
<tr>
<td>128</td>
<td>1.927787</td>
<td>2.328812</td>
<td>128</td>
</tr>
<tr>
<td>256</td>
<td>2.415757</td>
<td>2.727164</td>
<td>256</td>
</tr>
<tr>
<td>512</td>
<td>2.515509</td>
<td>3.131331</td>
<td>512</td>
</tr>
<tr>
<td>1024</td>
<td>2.540820</td>
<td>3.262716</td>
<td>1024</td>
</tr>
<tr>
<td>2048</td>
<td>2.543437</td>
<td>3.359013</td>
<td>2048</td>
</tr>
</tbody>
</table>

(a) Forwarding Performance (4RE)  
(b) Load Balancing (8RE)

Table II. Simulation Results($T_m = 6 T_h$)

Table IIa shows the simulation results for a 4-RE system for the two traces. The use of a cache results in increasing throughput because the lookups proceed through the cache, costing $T_h$ per lookup, rather than through software lookup, costing (the much larger) $T_m$ per lookup. For both traces, a small amount of cache (4 entries in the UofA trace and 32 in the LDestIP trace) doubles the throughput. The differences between the results for the two traces are due to the peculiarities of each trace, for example, the composition of the trace at the flow level.

<table>
<thead>
<tr>
<th>Trace</th>
<th>UofA</th>
<th>LDestIP</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of Pkts</td>
<td>999,993</td>
<td>31,518,464</td>
</tr>
<tr>
<td>No. of Flows</td>
<td>5,861</td>
<td>130,163</td>
</tr>
<tr>
<td>No. of Pkts in 5 Flows</td>
<td>158,707 (15.9%)</td>
<td>118,334 (3.7%)</td>
</tr>
<tr>
<td>Largest Flows in 5 Flows</td>
<td>24,245 (2.4%)</td>
<td>581,495 (1.8%)</td>
</tr>
<tr>
<td>Flows</td>
<td>17,482 (1.7%)</td>
<td>235,363 (0.7%)</td>
</tr>
<tr>
<td>Flows</td>
<td>15,146 (1.5%)</td>
<td>212,150 (0.7%)</td>
</tr>
</tbody>
</table>

Table III. Dominating Flows in the Traces
Before showing the results for the effect of caching on \( CV \), we should note that with the hashing scheme fixed, the composition of traces in terms of flow rates affects the value of \( CV \). Generally, the more skewed the flow rate distribution, i.e., the more dominant a few flows are in the trace, the larger the value of \( CV \). To give an appreciation of the flow rate composition of the two traces, Table III lists the largest 5 flows in each.

Table IIb shows the simulation results for an 8-RE system. Generally, caches of all sizes help to reduce the \( CV \). However, it is apparent that certain cache sizes are optimal. For the UofA trace, the optimal size is 16 entries and for the LDestIP trace, it is 8. As cache size increases, caching tends to be less beneficial in terms of helping balancing the load. In the extreme case that there are only compulsory misses, the \( CV \) approaches a fixed value. This is the case with the UofA trace (see also Table I for the number of flows for each RE.)

5. CONCLUSIONS

By characterizing empirical reuse distance data, we have been able to establish a workload model that quantitatively describes temporal locality in destination IP address traces. The reuse distance model is then applied to characterizing and comparing temporal locality in parallel forwarding systems with different packet dispatching schemes. The immediate conclusion from this work is that compared with round robin scheduling, hashing can significantly improve temporal locality in the workload of individual forwarding engines. The natural induction is that caching would be effective in such situations. This is demonstrated by simulation results.

Unlike round-robin, hashing does not balance the workload of the processors. If the result of hashing on destination addresses is random enough, the numbers of flows scheduled onto each processor would be similar. But due to the fact that the distribution of the flow rates in the aggregate workload is skewed, each processor is assigned a different load. Measurement shows that heavier workload usually has better temporal locality. This leads to another benefit from caching under a hash-scheduler, i.e., that it helps to balance the workload. Our simulation results show that although caching alone is not likely to reduce \( CV \) to zero, its presence automatically alleviates the load imbalance problem.

ACKNOWLEDGMENT

We would like to thank Joerg Micheel of NLANR for answering many questions on the traces used in our experiments.

REFERENCES

12. NLANR (National Laboratory for Applied Network Research) Measurement and Operations Analysis Team (MOAT), http://moat.nlanr.net,

Copyright © 2000 John Wiley & Sons, Ltd.

Int. J. Commun. Syst. 2000; 00:1–16

Prepared using dc auth cs
Figure 1. Reuse Distance Over Time (UofA:left; LDestIP)

Figure 2. Reuse Distance CCDF’s of the UofA(left) and LDestIP Traces

Figure 3. Fitting CCDF’s with Weibull + Pareto (UofA(left) and LDestIP)
Figure 4. CCDF's for the traces in the CLEV-0(left) and CLEV-1 groups

Figure 5. CCDF's for the traces in the AuckIV-0(left) and AuckIV-1 groups

Figure 6. Some CCDF Fittings(CLEV)
Figure 7. Some CCDF Fittings (AuckIV)

Figure 8. Effects of $p$ Fitted CCDF's for Two AuckIV Traces and Their Components

Figure 9. Effects of $p$ Fitted CCDF's for Two IMLS-CLEV Traces and Their Components
Routing Engines:

- RE 1: cache
- RE 2: cache
- RE 3: cache
- RE 4: cache

Figure 10. A Multi-processor Forwarding System

Figure 11. CCDF's in a 4-RE System with the UofA(above) and the LDestIP Traces

Figure 12. CCDF's in a 16-RE System with the LDestIP Trace
Figure 13. Impact of Scheduling on Per Processor Locality (UofA Trace with 4 RE's)