Accurate Hierarchical Traffic Measurement in Datacenters Through Differentiated Memory Allocation

Olufemi Odegbile* Shigang Chen† Yuanda Wang‡ Dimitrios Melissourgos§
Department of Computer and Information Science and Engineering
University of Florida, Gainesville, Florida, USA
Email: *oodegbile, †sgchen , ‡yuandawang, §dmelissourgos@ufl.edu

***Abstract*—A hierarchical flow model provides us the opportunity to monitor network traffic in datacenters, at once, from coarser (aggregate flows) to finer granularity (individual flows). Per-flow traffic measurement based on an idea of counter-sharing is memory-efficient by recording the sizes of all flows in a shared array of physical counters. Through this approach, flows at the same level record their sizes in uniformly sized virtual counter arrays constructed from the shared memory. However, flows at the same level have vastly different sizes. The virtual counter arrays for large flows must be big enough to uniformly distribute errors it contains. However, when large virtual counter arrays are used to record small flow sizes, the arrays accumulate more errors than the recorded flows'sizes, thereby affecting the accuracy of the estimations. The paper introduces the concept of differentiated virtual counter arrays, where the number of virtual counters allocated to each flow is based on its historical size. We derive a mathematical formula for flow-size estimation when recording flow sizes with flexible number of virtual counters and develop a heuristic algorithm to calculate how many virtual counters are assigned to each flow. Through experiments, we demonstrate that our solution is more accurate by up to 75% for aggregate flows and by up to 100% for small base flows compared with the prior art.**

I. Introduction

Modern datacenters' traffic has been growing at a rate of 25% annually and this trend will continue into the future [1]. This places ever increasing strain on how fast packets must be processed on the line cards of datacenter switches. In addition, the demand of diverse traffic measurement to support sophisticated network management only adds more to the stress [2]. Therefore, additional measurement tasks need to be designed efficiently to compete less on the limited resources such as SRAM on the data plane of a router, while at the same time delivering accurate and usable results.

The widely-used tool for traffic measurement is NetFlow [3], which measures the size of each flow such as the number of packets and other statistics. Per-flow monitoring is essential in modern network management for generating a traffic matrix, traffic engineering, clients' billing, network security, among others [4]–[9]. NetFlow is however costly in memory. There are several memory-efficient probabilistic measurement approaches based on counter-sharing architectures [5], [10]–[14]. They reduce memory requirement by sharing counters among different flows and providing approximate answers for flow statistics.

However, most prior work considers disjoint flows that do not share any packet. Therefore, each arrival packet will be recorded exactly once for the flow that it belong to. That is however not the case in datacenter traffic measurement: A datacenter is typically organized in a hierarchical structure, where racks are connected via top-of-rack switches to aggregation switches then to core routers, each rack consists of a number of servers that connect to a top-of-rack switch, and each server may host a number of VMs. In order to learn traffic distribution in a datacenter, we may want to measure traffic between any two racks, traffic between any two servers (in different racks), and traffic between any two VMs (in different racks). While the top-of-rack and aggregation switches are often off-the-shelf products, we assign the measurement tasks to the core routers which can be modularized for additional functions. We call the packets from one rank (server, VM) to another as a *rack* (*server*, *VM*) *flow*. Obviously, each rack flow contains a number of server flows, and each server flow may contain a number of VM flows. These flows can be organized in a three-level hierarchy, with rack flows at the top, server flows at the second level, and VM flows at the bottom level. It is no longer true that each packet belongs to one and only one flow. Instead, it belongs to a VM flow, a parent server flow that contains the VM flow, and a parent rack flow that contains the server flow. If we use the traditional approaches of traffic measurement, we will have to record the packet three times, one for each flow that it belongs to, which increases the processing overhead by threefold.

Chen *et al* [6] proposed hierarchical virtual counters (HVC) to process packets belonging to a flow hierarchy. Consider a packet p that belongs to a VM flow f_3, its parent server flow f_2, and the parent rack flow f_1. HVC assigns a number s_3 of virtual counters to record f_3, a number s_2 of virtual counters to record f_2, and a number n_1 of virtual counters to record f_1, where the reason the counters are called virtual is because they are shared with other flows at the same level. The key is that the virtual counters for f_3 are a subset of virtual counters for f_2, which are in turn a subset of those for f_1. By recording packet p in any counter for f_3, the packet is automatically recorded for f_2 and f_1 as well. That is, processing the packet once, it is recorded for three flows that it belongs to.

When we experiment with HVC, we discover an overlooked

978-1-7281-8275-9/20/$31.00 ©2020 IEEE
DOI 10.1109/BigCom51056.2020.00025

problem: It sets the same value for s_3 (s_2, s_1) for all flows at the same level of the hierarch. The VM flows are each recorded in s_3 virtual counters, the server flows are each recorded in s_2 counters, and the rack flows are each recorded in s_1 flows. However, flows at the same level have vastly different sizes. Assigning them with the same number of virtual counters causes significant performance problems. For example, if we set s_3 (or s_2, s_1) large, that helps large flows but makes the measurement of small flows really inaccurate due to excessive noise from counter sharing. If we set it small, that helps most small flows but causes large measurement errors for some as the noise from large flows are now concentrating in a small number of counters.

In this paper, we introduce the concept of differentiated memory allocation (DMA) in hierarchical traffic measurement, which allows different flows to have different numbers of virtual counters. We combine HVC with differentiated memory allocation to improve traffic measurement accuracy, while still ensuring that each packet is processed once for all flows that it belongs to. DMA adaptively decides on the number of virtual counters that record the packets of each flow based on the flow's historical size in previous measurement periods. Consequently, a larger flow will use a larger number of virtual counters. As the flow size evolves over time, its allocated number of counters will also change. Our contributions is summarized as follows: We first derive a mathematical formula for flow-size estimation under DMA and prove its unbiasness. We then propose a heuristic algorithm to determine (offline) how many virtual counters are allocated to each flow at each hierarchy level, subject to a user-defined accuracy requirement. Finally, through trace-based experiments, we demonstrate that traffic measurements based on DMA are significantly more accurate than the prior art.

II. FLOW MODEL AND PROBLEM STATEMENT

With the example of rack/server/VM flows in the introduction, we give a generalized, formal model for datacenter flow hierarchy as follows: Consider a flow hierarchy of l levels, where flows at the same level share no packet. A *parent* flow f_j at the jth level contains a number of n *subflows* $f_{j+1}^1, f_{j+1}^2, \cdots, f_{j+1}^n$ at $(j+1)$th level, where $1 \leq j < l$ and $n \in \mathbb{Z}^+$. By definition, $f_j = \bigcup_{i=1}^{n} f_{j+1}^i$. We will refer to flows at the lth level as *base* flows, while flows at other levels are called *aggregate* flows, each consisting multiple subflows at a lower level. Any packet that belongs to a base flow f_l also belongs to all its parent flows at the $(l-1)$th level (f_{l-1}) to the first level (f_1).

This paper studies how to improve the accuracy in estimating the sizes of all flows in the above hierarchical flow model through differentiated memory allocation (DMA), which may assign different numbers of virtual counters to flows at any given level. The flow size is defined by the number of packets in a flow. We require that a packet incurs only one counter update regardless of the number of flows that the packet belongs to. DMA allows us to better utilize the memory allocated at a core router for hierarchical flow measurement, resulting in better traffic estimation accuracy.

TABLE I

Notations	Definition
C	an array of counters
m	the size of C
l	the number of levels in an hierarchy
g_j	an aggregate flow at jth level, $1 \leq j < l$
g_l	a base flow at lth level
C_{g_j}	virtual counter array of a flow g_j
s_{g_j}	the length of C_{g_j}
$\hat{s}_{g_j}$	estimated length of C_{g_j}
n	total size of all flow in a measurement period
n_{g_j}	actual size of a flow g_j
$\hat{n}_{g_j}$	estimated size of a flow g_j
$n_{g_j}^C$	the number of packets recorded in C_{g_j}
$H_M(\cdot)$	a master hash function
$\oplus$	the XOR operator
R_{g_j}	a set of random numbers of size s_{g_j} for a flow g_j
X_{k,g_j}	a random variable representing the number of packets recorded by a virtual counter at an index k in C_{g_j}
Z_α	the corresponding percentile for a confidence level α in a standard normal distribution
ϕ	an *accuracy threshold*, specifying a user-defined confidence interval

III. HIERARCHICAL TRAFFIC MEASUREMENT WITH DIFFERENTIATED MEMORY ALLOCATION

In this section, we first describe the construction of differentiated virtual counter arrays, which we will refer to as differentiated memory allocator (DMA). Afterwards, we describe how the per-flow differentiated virtual counter array is updated online during packet forwarding. Finally, how the size of each flow at all levels are estimated at the end of a measurement interval is discussed.

A. Construction of Virtual Counter Arrays

Virtual counter arrays are pseudo-randomly constructed from a physical array C of m counters, each of which are reset to zero at the beginning of each measurement period. A counter in C is denoted as $C[i]$, $i \in [0, m)$.

Suppose that $G = \{g_i\}_{1 \leq i \leq l}$ is a sequence of hierarchical flows, where g_l is a base flow and its parents at the $(l-1)$th to the first level are $g_{l-1}, \cdots, g_1$, respectively. Let C_{g_i} represents the virtual counter array used to record packets belonging to g_j at jth level, where $1 \leq j \leq l$. Since the flows in G are dependent, we also construct a dependent virtual counter array for each flow. To this end, each flow selects a number of counters from its parent flow to construct its own virtual counter array. The first level flow g_1, which has no parent flow, constructs its virtual counter array by randomly selecting a number of counters from C.

In HVC [6], the virtual counter arrays of all flows on the same level are of the same size. In order to improve hierarchical measurement accuracy, we allocate differently

sized virtual memory to each flow on the same level. Let the differentiated length of C_{g_j} be s_{g_j}. We will discuss in detail how to calculate s_{g_j} in Section IV. In what follows, we describe how the virtual counters in C_{g_j} are pseudo-randomly selected from $C_{g_{j-1}}$, which is the virtual counter array of g_{j-1}, where g_{j-1} is the parent flow of g_j.

$$C_{g_j}[k] = C_{g_{j-1}}[H_k(g_j)],\ 0 \le k < s_{g_j}, \tag{1}$$

where $H_k(\cdot)$, $0 \le k < s_{g_j}$, are independent hash functions. Instead of selecting counters with s_{g_j} different hash functions, we can use a master hash function $H_M(\cdot)$ in the following way:

$$C_{g_j}[k] \coloneqq C_{g_{j-1}}[H_M(g_j \oplus R_{g_j}[k])],\ 0 \le j < s_{g_j}, \tag{2}$$

where R_{g_j} is a set of s_{g_j} random numbers and $\oplus$ is the XOR operator. For the first level flow g_1, we have

$$C_{g_1}[k] \coloneqq C[H_M(g_1 \oplus R_{g_1}[k])],\ 0 \le k < s_{g_1}, \tag{3}$$

B. Online Data Encoding

Consider a switch processing a data stream consisting of multiple flows. For an arrival packet, the switch first extract a pre-defined label (from the packet's header) to identify hierarchical flows $g_1, g_2, \cdots, g_l$ to which the packet belongs, where g_1 is the parent to g_2, and g_2 is the parent to g_3, and so on. For example, suppose that a hierarchical flow model has 4 levels and destination IPv6 address is the label used to identify the base and aggregate flows. The fourth level flow is identified by 48-bit prefix; the third level flow is identified by 32-bit prefix; the second level flow is identified by 24-bit prefix; the first level flow is identified by 16-bit prefix..

Next, the switch randomly select a counter from the virtual counter array of g_l and increase it by 1. That is,

$$C_{g_l}[k^*] \coloneqq C_{g_l}[k^*] + 1,\ \text{for some } k^* \in [0, s_{g_l}). \tag{4}$$

From (2), we can rewrite the left hand side of (4) as

$$C_{g_l}[k^*] = C_{g_{l-1}}[H_M(g_l \oplus R_{g_l}[k^*])]. \tag{5}$$

Because the $C_{g_{l-1}}$ is constructed from $C_{g_{l-2}}$, the virtual counter in (5) is equivalent to the following virtual counter in $C_{g_{l-2}}$:

$$C_{g_l}[k^*] = C_{g_{l-2}}[H_M(g_{l-1} \oplus R_{g_{l-1}}[H_M(g_l \oplus R_{g_l}[k^*])])]. \tag{6}$$

We continue in this reasoning to reach the following corresponding physical counter:

$$\begin{aligned} &C_{g_l}[k^*] \\ &\quad = C[H_M(g_1 \oplus R_{g_1}[H_M(g_2 \oplus R_{g_2}[\cdots \\ &\qquad \cdots H_M(g_l \oplus R_{g_l}[k^*])])])] \end{aligned} \tag{7}$$

Note that none of the virtual counter arrays is actually constructed during online encoding. The only operation executed by the switch after receiving a packet is to increase the physical counter in (7) by 1. That is,

$$\begin{aligned} &C[H_M(g_1 \oplus R_{g_1}[H_M(g_2 \oplus R_{g_2}[\cdots \\ &\qquad \cdots H_M(g_l \oplus R_{g_l}[k^*])])])] \mathrel{+}= 1. \end{aligned} \tag{8}$$

C. Offline Flow Size Estimation

The flow size estimation is done offline at the end of a measurement period, when the values of the counters in C are copied to a hard disk or uploaded to a centralized server or controller. Let n be the total size of all flows. Because each arrival packet only updates one physical counter in C, $n = \sum_{k=0}^{m-1} C[k]$. Suppose that the actual size of a flow g_j at an arbitrary level j, $1 < j \le l$, is n_{g_j}. To obtain an estimated value of n_{g_j}, denoted as $\hat{n}_{g_j}$, we first reconstruct the respective virtual counter arrays of g_j and g_{j-1} as follows:

$$\begin{aligned} C_{g_j}[k] = C[H_M(g_1 \oplus R_{g_1}[H_M(g_2 \oplus R_{g_2}[\cdots \\ \cdots H_M(g_j \oplus R_{g_j}[k])])])] \\ 0 \le k < s_{g_j}. \end{aligned}$$

and

$$\begin{aligned} C_{g_{j-1}}[k] = C[H_M(g_1 \oplus R_{g_1}[H_M(g_2 \oplus R_{g_2}[\cdots \\ \cdots H_M(g_{j-1} \oplus R_{g_{j-1}}[k])])])] \\ 0 \le k < s_{g_{j-1}}. \end{aligned}$$

The derivation of $\hat{n}_{g_j}$ follows similar mathematical approach proposed in HVC [6], where virtual counter arrays of uniform length are used to encode flow-size information of flows at the same level in an hierarchy. In contrast, we propose a more accurate estimator (compared to HVC) by using virtual counter array of variable length to store the number of packets belonging to each flow.

Let $n^C_{g_j}$ and $n^C_{g_{j-1}}$ be the number of packets recorded in C_{g_j} and $C_{g_{j-1}}$, respectively. Clearly, $n^C_{g_j} = \sum_{k=0}^{s_{g_j}-1} C_{g_j}[k]$ and $n^C_{g_{j-1}} = \sum_{k=0}^{s_{g_{j-1}}-1} C_{g_{j-1}}[k]$. Let X_{k^*,g_j} be a random variable representing the number of packets recorded by a virtual counter at an arbitrary location k^* in C_{g_j}. X_{k^*,g_j} can be written as a sum of two random variables as follows:

$$X_{k^*,g_j} = Y_{k^*,g_j} + e_{k^*,g_j}, \tag{9}$$

where Y_{k^*,g_j} is the number of packets contributed by g_j to $C_{g_j}[k^*]$ and e_{k^*,g_j} is the number of packets contributed by other flows to $C_{g_j}[k^*]$. From the point of view of g_j, e_{k^*,g_j} is the amount of noise contained in $C_{g_j}[k^*]$. The maximum value of Y_{k*,g_j} is n_{g_j}. The probability that a packet belonging to g_j is recorded in $C_{g_j}[k^*]$ is $\frac{1}{s_{g_j}}$. Since each packet independently updates a virtual counter, Y_{k^*,g_j} follows a binomial distribution:

$$Y_{k^*,g_j} \sim Bino\left(n_{g_j}, \frac{1}{s_{g_j}}\right) \tag{10}$$

Next, we find the approximate distribution of e_{k^*,g_j}. Note that all the packets, which belong to other flows, that can potentially update $C_{g_j}[k^*]$ are already stored in $C_{g_{j-1}}$. Therefore,

the maximum value of e_{k^*,g_j} is $n^C_{g_{j-1}} - n_{g_j}$. The probability that a packet belonging to other children flows of g_{j-1} is recorded in $C_{g_j}[k^*]$ is $\frac{1}{s_{g_{j-1}}}$, because the packet chooses from $s_{g_{k-1}}$ virtual counters. Also, because of independent counter updates, e_{k^*,g_j} follows a binomial distribution:

$$e_{k^*,g_j} \sim Bino\left(n^C_{g_{j-1}} - n_{g_j}, \frac{1}{s_{g_{j-1}}}\right) \tag{11}$$

Since $n^C_{g_{j-1}} \gg n_{g_j}$,

$$e_{k^*,g_j} \sim Bino\left(n^C_{g_{j-1}}, \frac{1}{s_{g_{j-1}}}\right) \tag{12}$$

By taking the expected value of (9), we have that

$$\begin{aligned} E(X_{k^*,g_j}) &= E(Y_{k^*,g_j} + e_{k^*,g_j}) \\ &= E(Y_{k^*,g_j}) + E(e_{k^*,g_j}) \\ &= \frac{n_{g_j}}{s_{g_j}} + \frac{n^C_{g_{j-1}}}{s_{g_{j-1}}} \end{aligned} \tag{13}$$

We can rewrite (13) as

$$n_{g_j} = s_{g_j} E(X_{k^*,g_j}) - \frac{s_{g_j}}{s_{g_{j-1}}} n^C_{g_{j-1}}. \tag{14}$$

By replacing $E(X_{k^*,g_j})$ with the instance value $\frac{1}{s_{g_j}} \sum_{k=0}^{s_{g_j}-1} C_{g_j}[k]$, which is the average value of virtual counters C_{g_j}, our size estimation is

$$\hat{n}_{g_j} = \sum_{k=0}^{s_{g_j}-1} C_{g_j}[k] - \frac{s_{g_j}}{s_{g_{j-1}}} n^C_{g_{j-1}}, \tag{15}$$

When $k = 1$, the flow size of the base flow g_1 is

$$\hat{n}_{g_1} = \sum_{k=0}^{s_{g_1}-1} C_{g_1}[k] - \frac{s_{g_1}}{m} n. \tag{16}$$

IV. Accuracy of DMA

In this section, we deal with how to calculate the length of virtual counter array of each flow in an hierarchy. As in the previous section, consider a sequence of hierarchical flows $g_1, g_2, \cdots, g_l$, where g_1 is a parent flow to g_2, which is in turn a parent flow to g_3, and so on. Given a flow g_j at a level $j > 1$, our idea is to calculate s_{g_j}, the length of a virtual counter array of g_j, that ensures the actual size of g_j is within a user-defined confidence interval of our estimation in (15). For example, a user may specified that the actual flow size is within 10% of the estimated size. In order to derive the confidence interval of $\hat{n}_{g_j}$, we first derive its mean and variance in what follows:

A. Mean and Variance of $\hat{n}_{g_j}$

Mean: We can calculate the mean of n_{g_j} by taking the expected value of (15) as follows:

$$E(\hat{n}_{g_j}) = E\left(\sum_{k=0}^{s_{g_j}-1} C_{g_j}[k]\right) - \frac{s_{g_j}}{s_{g_{j-1}}} E(n^C_{g_{j-1}}) \tag{17}$$

From (9),

$$\begin{aligned} \sum_{k=0}^{s_{g_j}-1} C_{g_j}[k] &= \sum_{k=0}^{s_{g_j}-1} X_{k,g_j} \\ &= \sum_{k=0}^{s_{g_j}-1} Y_{k,g_j} + \sum_{k=0}^{s_{g_j}-1} e_{k,g_j} \\ &= n_{g_j} + \sum_{k=0}^{s_{g_j}-1} e_{k,g_j} \end{aligned}$$

where e_{k,g_j} is amount of the number of packets contributed by other flows to virtual counter $C_{g_j}[k]$. This implies that

$$\begin{aligned} E\left(\sum_{k=0}^{s_{g_j}-1} C_{g_j}[k]\right) &= E(n_{g_j}) + E\left(\sum_{k=0}^{s_{g_j}-1} e_{k,g_j}\right) \\ &= n_{g_j} + \sum_{k=0}^{s_{g_j}-1} E(e_{k,g_j}) \\ &= n_{g_j} + \frac{s_{g_j}}{s_{g_{j-1}}} E(n^C_{g_{j-1}}). \end{aligned} \tag{18}$$

By substituting (18) into (17), we have that

$$E(\hat{n}_{g_j}) = n_{g_j} \tag{19}$$

This also implies that our estimator in (15) is unbiased.

Variance: We start by deriving the variance of X_{k,g_j}, where k is an index in C_{g_j}. From (9), we have that

$$Var(X_{k,g_j}) = Var(Y_{k,g_j}) + Var(e_{k,g_j}), \tag{20}$$

because Y_{k,g_j} and e_{k,g_j} are independent. Since Y_{k,g_j} follows a binomial distribution, it variance is

$$Var(Y_{k,g_j}) = \frac{n_{g_j}}{s_{g_j}} \left(1 - \frac{1}{s_{g_j}}\right) \tag{21}$$

By substituting (21) and $Var(e_{k,g_j})$ obtained in Appendix A into (20), we have that

$$Var(X_{k,g_j}) = \frac{n}{m}\left(1 - \frac{1}{m}\right) + \sum_{i=2}^{j+1} \frac{n_{g_{i-1}}}{s_{g_{i-1}}} \left(1 - \frac{1}{s_{g_{i-1}}}\right) \tag{22}$$

We can rewrite $\hat{n}_{g_j}$ in (15) as

$$\hat{n}_{g_j} = \left(1 - \frac{s_{g_j}}{s_{g_{j-1}}}\right) \sum_{k=0}^{s_{g_j}-1} C_{g_j}[k] - \frac{s_{g_j}}{s_{g_{j-1}}} \sum_{k=s_{g_j}}^{s_{g_{j-1}}-1} C_{g_{j-1}}[k] \tag{23}$$

Hence, the variance of $\hat{n}_{g_j}$ is

$$\begin{aligned} Var(\hat{n}_{g_j}) = s_{g_j} &\left(1 - \frac{s_{g_j}}{s_{g_{j-1}}}\right)^2 Var(X_{k_1,g_j}) \\ &+ (s_{g_{j-1}} - s_{g_j}) \left(\frac{s_{g_j}}{s_{g_{j-1}}}\right)^2 Var(X_{k_2,g_{j-1}}), \end{aligned} \tag{24}$$

for some $k_1 \in [0, s_{g_j})$ and $k_2 \in [0, s_{g_{j-1}})$. The variance X_{k_1,g_j} and $X_{k_2,g_{j-1}}$ can be obtained from (22). When $j = 1$, the variance of $\hat{n}_{g_1}$ is

$$Var(\hat{n}_{g_1}) = s_{g_1} Var(X_{k,g_1}), \tag{25}$$

for some $k \in [0, s_{g_1})$.

B. Confidence Interval of n_{g_j}

Distribution of Y_{k,g_j}: Consider a virtual counter at a index k in C_{g_j}. For sufficiently large $n, n_{g_1}, \cdots, n_{g_j}$, and from (10), Y_{k,g_j} can be approximated as the following normal distribution:

$$Y_{k,g_j} \sim N\left(\frac{n_{g_j}}{s_{g_j}}, \frac{n_{g_j}}{s_{g_j}}\left(1 - \frac{1}{s_{g_j}}\right)\right).$$

Distribution of X_{k,g_j}: Similarly, as shown in Appendix B, e_{k,g_j} can also be approximated as a normal distribution. This implies that X_{k,g_j} is a sum of two independent random variables. Therefore, X_{k,g_j} can also be approximated as a normal distribution.

Distribution of $\hat{n}_{g_j}$: From (23), $\hat{n}_{g_j}$ is a linear combination of $X_{k,g_{j-1}} = C_{g_{j-1}}[k]$, $0 < k \leq s_{g_{j-1}}$. Hence, $\hat{n}_{g_j}$ can be approximated as the following normal distribution:

$$\hat{n}_{g_j} \sim N(E(\hat{n}_{g_j}), Var(\hat{n}_{g_j})) = N(n_{g_j}, Var(\hat{n}_{g_j})),$$

where $Var(\hat{n}_{g_j})$ is in (24). The confidence interval of $\hat{n}_{g_j}$ is

$$\hat{n}_{g_j} \ \pm \ Z_\alpha \sqrt{Var(\hat{n}_{g_j})}, \tag{26}$$

where Z_α is the corresponding percentile for a confidence level α in a standard normal distribution.

C. Calculation of $s_{g_1}, s_{g_2}, \cdots, s_{g_j}$

In this section, we discuss how to determine the differentiated lengths of virtual counter arrays of hierarchical flows. Let $\hat{s}_{g_1}, \hat{s}_{g_2}, \cdots, \hat{s}_{g_l}$ be the estimated lengths of virtual counter arrays of heirarchical flows $g_1, g_2, \cdots, g_l$. The estimation of $\hat{s}_{g_j}$, where $1 \leq j \leq l$, is done offline, before the start of a measurement period. For example, in a datacenter, a hypervisor or a centralized controller performs the calculation.

We select s_{g_j} *such that* n_{g_j}*, the actual flow size of* g_j*, is within a user-defined confidence interval of its estimated size* $\hat{n}_{g_j}$*, where* $1 \leq j \leq l$. To this end, we define the following function of s_{g_j} from (26):

$$\Phi_k(s_{g_1}, \cdots, s_{g_j}) = \frac{Z_\alpha \sqrt{Var(\hat{n}_{g_j})}}{n_{g_j}}. \tag{27}$$

Suppose that ϕ is an *accuracy threshold*, specifying a user-defined confidence interval. For example, suppose that we want to ensure that n_{g_j} $\forall j \in [1, l]$, is within 10% of $\hat{n}_{g_j}$. In this case, ϕ is 10%. The problem of finding a s_{g_j} subject to the accuracy condition above is the following inequality:

$$\Phi_k(s_{g_1}, \cdots, s_{g_j}) \ \leq \ \phi, \tag{28}$$

Algorithm 1 $s_{g_1}, s_{g_2}, \cdots, s_{g_j}$ estimation

Input: $n_{g_1}, n_{g_2}, \cdots, n_{g_j}$, n, m and ϕ
Output: estimated $\hat{s}_{g_1}, \hat{s}_{g_2}, \cdots, \hat{s}_{g_j}$

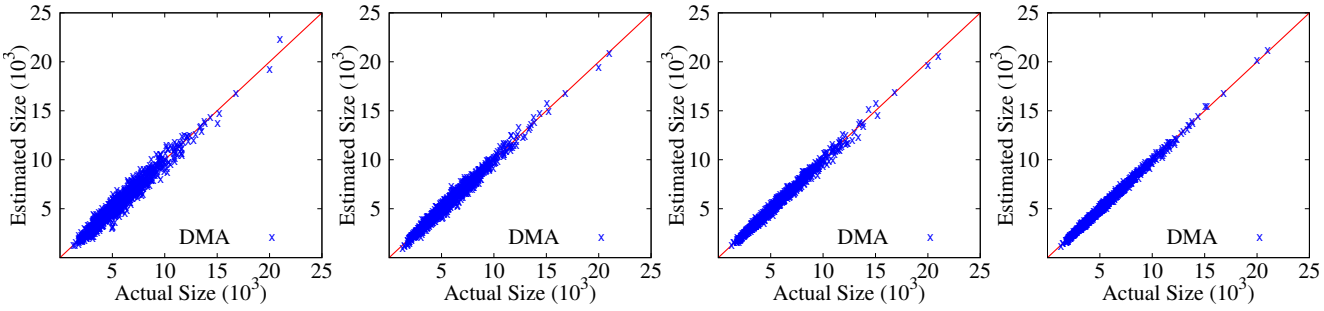
```
for i = 1 ··· j do
    switch (i)
    case 1:
        Obtain ŝ_{g_1} by solving:
        Φ(s_{g_1}) ≤ φ
    case 2:
        Obtain ŝ_{g_2} by solving:
        Φ(ŝ_{g_1}, s_{g_2}) ≤ φ
    case 3:
        Obtain ŝ_{g_2} by solving:
        Φ(ŝ_{g_1}, ŝ_{g_2}, s_{g_3}) ≤ φ
    ···  ···  ···
    ···  ···  ···
    case j:
        Obtain ŝ_{g_2} by solving:
        Φ(ŝ_{g_1}, ŝ_{g_2}, ··· , ŝ_{g_{j-1}}, s_{g_j}) ≤ φ
    end switch
end for
```

From (27), Φ is dependent on actual sizes $n_{g_1}, \cdots, n_{g_j}$, each of which can be calculated from its historical or estimated values from the previous measurement periods. Because Φ also depends on the lengths of virtual counter arrays of the parent flows of g_j, that is $s_{g_1}, \cdots, s_{g_{j-1}}$, we propose a recursive solution in Algorithim 1 to solving (28). We first estimate $\hat{s}_{g_1}$, which is a solution of (28) when $j = 1$. Then we sustitute $\hat{s}_{g_1}$ in place of s_{g_1} to estimate $\hat{s}_{g_2}$ from (28) when $j = 2$. We continue this process until $j = l$. At this point, we use previously estimated $\hat{s}_{g_1}, \cdots, \hat{s}_{g_{j-1}}$ to estimate $\hat{s}_{g_l}$ from (28).

Once the $\hat{s}_{g_1}, \hat{s}_{g_2} \cdots, \hat{s}_{g_j}$ are estimated through Algorithim 1, the controller forwards them to top-of-rack switches whose subnet addresses overlap with the identifier/descriptor of the base flow g_l. When a top-of-rack switch receives a packet p belonging to g_l, the switch embeds $\hat{s}_{g_1}, \hat{s}_{g_2} \cdots, \hat{s}_{g_j}$, in the unused header fields (such as TOS byte and fragmentation offset if the network is configured to avoid fragmentation) of p before forwarding it to the next hop. Afterwards, the embedded lengths of virtual arrays will be retrieved to encode p by a core switch responsible to monitoring network traffic.

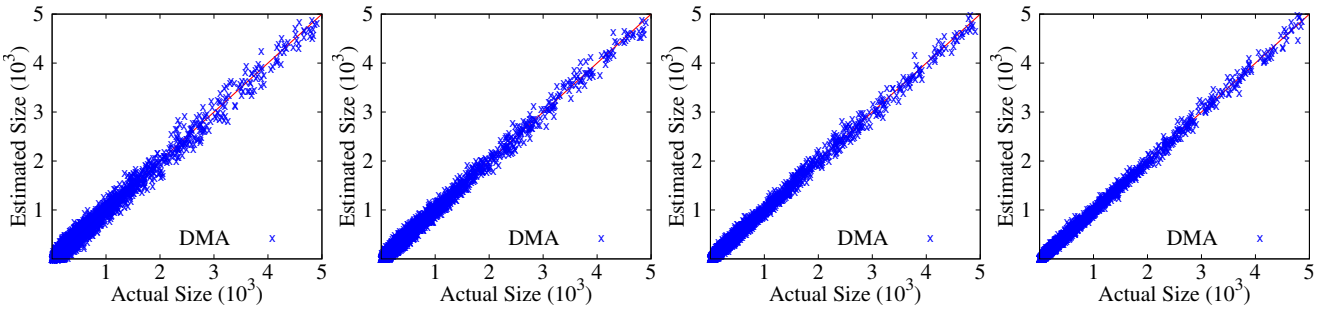
V. EXPERIMENTS

Through extensive simulations, we evaluate the accuracy of our hierarchical flow-size estimator (DMA), which is designed based on differentiated virtual memory allocation at each level of an hierarchy. We also compare our estimator with the state-of-the-art hierarchical virtual counter (HVC) [6], which uses virtual counters of the same length to estimate the size of flows at the same level of an hierarchy.



(a) $m = 0.5$MB (b) $m = 1.0$MB (c) $m = 2.0$MB (d) $m = 4.0$MB

Fig. 1: Estimated spreads of aggregate flows under DMA with $m = 0.5$MB, 1MB, and 2MB memory, respectively. Accuracy of DMA improves as we allocate more memory.



(a) $m = 0.5$MB (b) $m = 1.0$MB (c) $m = 2.0$MB (d) $m = 4.0$MB

Fig. 2: Estimated spreads of aggregate flows under DMA with $m = 0.5$MB, 1MB, and 2MB memory, respectively. Accuracy of DMA improves as we allocate more memory.

A. Settings

We simulate a two-level hierarchical flow model in a datacenter, where a number of severs communicate with one another. Each sever also hosts a number of virtual machines (VMs). An aggregate flow at the first level represents the packets between a pair of servers, while a base flow is the packets between two VMs, each hosted by different servers. There are 1000 aggregate flows, each of which contains a random number of base flows from a range $[500, 2000]$. In total, there are approximately 600000 base flows. We model the sizes of base flows following a power law from a range $[1, 5000]$. As a result, the size of an aggregate flow is from a range of $[1000, 25000]$.

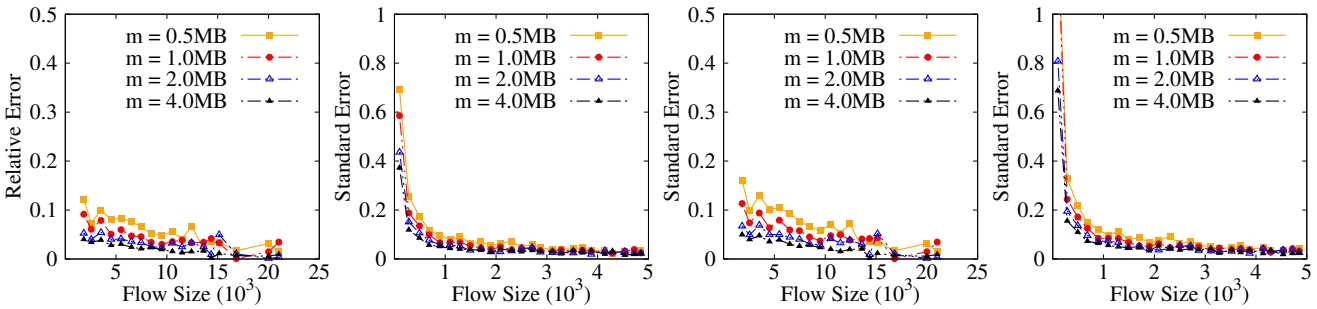
We solve (28) to calculate the number of virtual counters assigned to each aggregate flows and large base flows, where the accuracy threshold ϕ is selected as 10%. The large base flows contain 500 or more packets. We divide base flows containing less than 500 packets into the following 5 categories. We use 2, 10, 25, 50, and 100 virtual counters to encode the sizes of base flows containing less than 10 packets, less than 50 packets, less than 100 packets, less than 250 packets and less than 500 packets, respectively. In our implementation of HVC, we use 10000 and 200 virtual counters to encode the number of packets contained in each aggregate and base flows, respectively.

The performance metrics used to evaluate the accuracy of DMA and HVC are average relative error, average absolute error and average standard error. The estimator with smaller performace metrics is more accurate than the other.

B. Estimation Accuracy

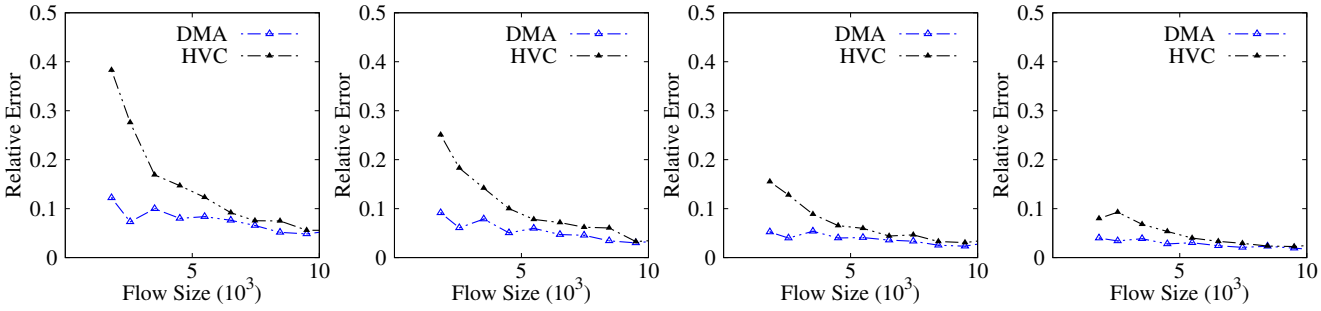
The first evaluations of the accuracy of our proposed hierarchical estimator (DMA) are shown Fig. 1-2, where plots (a), (b), (c) and (d) in each of the two figures are when a switch allocates 0.5MB, 1MB, 2MB and 4MB memory, respectively, for traffic monitoring. Each point in each plot represents a flow. The x-axis represents the actual sizes of either aggregate flows (Fig. 1) or base flows (Fig. 2). The y-axis represents the estimated sizes of aggregate flows (Fig. 1) and base flows (Fig. 2). The equality line $y = x$ in each plot is for reference, such that the closer a point is to the equality line, the more accurate the corresponding flow-size estimation is. Our results displayed in plots (a), (b), (c) and (d) in the figures show that the points are clustered around the equality line, which demonstrate the accuracy of our estimator. If we compare the plots within each figure, the points become more clustered as we allocate more physical memory.

The second set of experiments in Fig. 3 further evaluate the accuracy of DMA, when the physical memory allocation



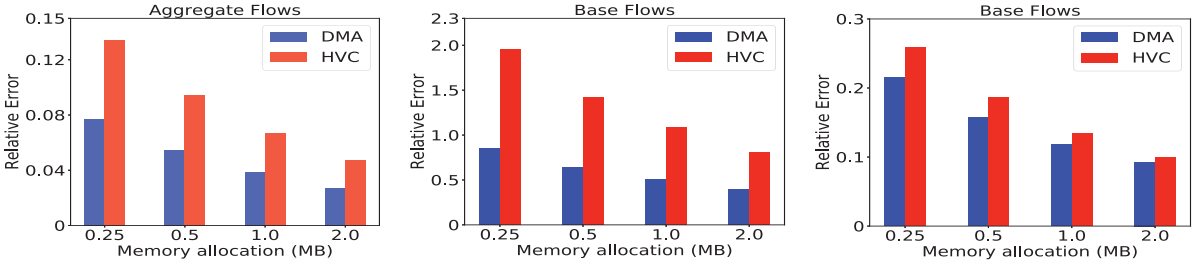
(a) Relative error of aggregate flows under DMA. (b) Relative error of base flows under DMA. (c) Relative error of aggregate flows flows under DMA. (d) Relative error of base flows under DMA

Fig. 3: Accuracy of DMA



(a) $m = 0.5$MB (b) $m = 1.0$MB (c) $m = 2.0$MB (d) $m = 4.0$MB

Fig. 4: Accuracy of flow-size measument of aggregatge (first level) flows under DMA vs HVC, when memory allocations are (a) 0.5MB, (b) 1.0MB, (c) 2.0MB, and (d) 4.0MB. DMA is more accurate than HVC by up to 75%.



(a) Average relative error of aggregate flows. (b) Relative error of base flows, which contain less than 250 packets). (c) Relative error of base flows, which contain 250 or more packets).

Fig. 5: Comparison of relative error of DMA with HVC.

is 0.5MB, 1MB, 2MB, and 4MB. In the Fig. 3(a) and (c), we show the relative and standard errors of estimated sizes of aggregate flows, respectively. Similarly, we show the relative and standard error of estimated sizes of base flows in Fig. 3(a) and (c), respectively. We can see from these figures that relative and standard deviation approach zero as the flow size increases. This implies that DMA is highly accurate for large flows, whose identification is essential for important network management tasks like traffic engineering and heavy hitter detection. In addition, as we allocate more memory for 0.5MB to 4MB, the two errors reduce and approach zero faster.

C. Comparison with prior art

Fig. 4 (a) - (d) compare the accuracy of size measurement of aggregate flows uder DMA to HVC, when the physical memory allocations are 0.5MB, 1MB, 2MB and 4MB, respectively. DMA improves the accuracy of HVC by up to 75%. Fig. 4 shows the improvement of our work for small aggregate, whose sizes are not more than 10000. Fig. 5(a) compares the average relative error of estimated sizes of all aggregate flows obtained from DMA to HVC under different memory allocations. Clearly, DMA is significantly more accurate than HVC by improving the accuracy of size measurement by up

to 50%.

In order to compare the accuracy of DMA with HVC concerning base flows, we split base flows into two categories: the flows in the first category contains fewer than 250 packets and the flows in the second category contains 250 or more packets. In Fig. 5, plots (b) and (c) show the average relative error of estimated sizes for the flows in first and second categories, respectively. As in the case of aggregate flows, DMA performs significantly (for the first category in Figure 5(b)) or slightly (for the second category in Fig. 5(c)) better than HVC. Although the average relative error of estimated sizes, which are obtained from DMA, for small base flows is higher (compared to that of aggregate flows in Fig. 5), the corresponding average absolute error of the estimations shown in Table II still remains relatively small.

TABLE II: The average absolute error of flow-size estimations that are obtained from DMA for base flows.

Memory (MB)	0.25	0.5	1.0	2.0
Flows with $n_{g_2} \leq 250$	71.12	53.63	41.03	32.06
All flows	93.15	69.38	52.85	41.80

VI. Conclusion

This paper proposes an accurate and memory-efficient hierarchical flow-size estimator through a counter sharing architecture and differentiated virtual memory allocation. Our main idea is to determine the number of virtual counters used to store the number of packets belonging to a flow based on the actual or estimated sizes, which are obtained during the past measurement periods, of the flow. We derive a mathematical formula for our estimator and develop an heuristic algorithm to select the number of virtual counters to each flow. Our simulations on a two-levels hierarchical flow model show that the proposed estimator is at least 40% more accurate for aggregate flows and 50% more accurate for small base flows than the prior art.

Acknowledgment

This work was supported by National Science Foundation of US under grant CNS-1719222.

References

[1] Cisco global cloud index: Forecast and methodology, 2016 white paper. [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/service-provider/global-cloud-index-gci/white-paper-c11-738085.html

[2] J. L. G. Dorado, F. Mata, J. Ramos, P. M. S. del Ro, V. Moreno, and J. Aracil, "High performance network traffic processing systems using commodity hardware," in *Data Traffic Monitoring and Analysi*, ser. Lecture Notes in Computer Science, E. Biersack, C. Callegari, and M. Matijasevic, Eds. Berlin: Springer Verlag, 2013, vol. 7754.

[3] Netflow configuration guide. [Online]. Available: https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/netflow/configuration/15-mt/nf-15-mt-book.html

[4] M. Moshref, M. Yu, and R. Govindan, "Resource/accuracy tradeoffs in software-defined measurement," in *Proceedings of HotSDN*. IEEE, 2015.

[5] Y. Zhou, Y. Zhou, M. Chen, Q. Xiao, and S. Chen, "Highly compact virtual counters for per-flow traffic measurement through register sharing," in *Proceedings of Global Communications Conference (GLOBECOM)*. IEEE, 2016.

[6] S. Chen, Y. Zhou, and S. Chen, "Efficient hierarchical traffic measurement in software-defined datacenter networks," in *10th International Conference on Cloud Computing*. IEEE, 2017.

[7] Y. Yu, C. Qian, and X. Li, "Distributed and collaborative traffic monitoring in software defined networks," in *Proceedings of HotSDN*. IEEE, 2014.

[8] B. Kurt, E. Zeydan, U. Yabas, I. A. Karatepe, G. K. Kurt, and A. T. Cemgil, "A network monitoring system for high speed network traffic," in *Proceedings of Sensing, Communication, and Networking (SECON)*. IEEE, 2016.

[9] Y. Zhang, "An adaptive flow counting method for anomaly detection in sdn," in *Proceedings of the ninth ACM conference on Emerging networking experiments and technologies*. ACM, 2013, pp. 25–30.

[10] G. Cormode and S. Muthukrishnan, "An improved data stream summary: the count-min sketch and its applications," *Journal of Algorithms*, vol. 55, no. 1, pp. 58–75, Apr. 2003.

[11] Y. Lu, A. Montanari, B. Prabhakar, S. Dharmapurikar, and A. Kabbani, "Counter braids: a novel counter architecture for per-flow measurement," in *ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, vol. 36, no. 1. ACM, 2008, pp. 121–132.

[12] T. Li, S. Chen, and Y. Ling, "Fast and compact per-flow traffic measurement through randomized counter sharing," in *Proceedings of INFOCOM*. IEEE, 2011.

[13] M. Chen, S. Chen, and Z. Cai, "Counter tree: A scalable counter architecture for per-flow traffic measurement," *IEEE/ACM Transactions on Networking*, vol. 5, no. 2, pp. 1249–1262, Apr. 2016.

[14] Y. Zhou, S. Chen, Z. Mo, and Q. Xiao, "Point-to-point traffic volume measurement through variable-length bit array masking in vehicular cyber-physical systems," in *International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2015.

Appendix A
Variance of X_k

As we discussed in Section III-C, each virtual counter in C_{g_j} accumulates noise in terms of packets that belong to other flows. Suppose that k is an arbitrary index in C_{g_j}. Let P_{g_1} be a collection of packets, which belong to all first level flows apart from g_1. Let P_{g_i} be the collection of packets that belong to all ith level flows, except g_i, whose parent flow is g_{i-1}, where $1 < i \leq j$. Let $n^P_{g_i}$ be the size of P_{g_i}, where $1 \leq i \leq j$. Clearly, $n^P_{g_1} = n - n_{g_1}$ and $n^P_{g_i} = n_{g_{i-1}} - n_{g_i}$. We can write the e_{k,g_j} in terms of the accumulated noise as

$$e_{k,g_j} = \sum_{i=1}^{j} \epsilon_{k,g_i}, \tag{29}$$

where ϵ_{k,g_1} is a random variable representing the number of packets from P_{g_1} recorded in $C_{g_j}[k]$ and ϵ_{k,g_i} is a random variable representing the number of packets from P_{g_i} recorded in $C_{g_j}[k]$, where $1 < i \leq j$. The probability of recording a packet from P_{g_1} in $C_{g_j}[k]$ is $\frac{1}{m}$. Similarly, the probability recording a packet from P_{g_i} in $C_{g_j}[k]$ is $\frac{1}{s_{g_{i-1}}}$, where $1 < i \leq j$. Because each packet independently increases a counter by 1,

$$\epsilon_{k,g_1} \sim Bino\left(n - n_{g_1}, \frac{1}{m}\right) \approx Bino\left(n, \frac{1}{m}\right),$$

since $n \gg n_{g_1}$ and

$$\epsilon_{k,g_i} \sim Bino\left(n_{g_{i-1}} - n_{g_i}, \frac{1}{s_{g_{i-1}}}\right) \approx Bino\left(n_{g_{i-1}}, \frac{1}{s_{g_{i-1}}}\right)$$

since $n_{g_{i-1}} \gg n_{g_i}$, for $1 < i \leq j$. Because each ϵ_{k,g_i} in (29) is independent, the variance of e_{k,g_j} is

$$\begin{aligned} Var(e_{k,g_j}) &= \sum_{i=1}^{j} Var(\epsilon_{k,g_i}) \\ &= \frac{n}{m}\left(1 - \frac{1}{m}\right) + \sum_{i=2}^{j} \frac{n_{g_{i-1}}}{s_{g_{i-1}}}\left(1 - \frac{1}{s_{g_{i-1}}}\right) \end{aligned} \tag{30}$$

APPENDIX B
DISTRIBUTION OF e_{k,g_j}

For sufficiently large $n, n_{g_1}, \cdots$, and n_{g_k}, ϵ_{k,g_1} can be approximated as a the following normal distribution:

$$\epsilon_{k,g_1} \sim N\left(\frac{n}{m}, \frac{n}{m}\left(1 - \frac{1}{m}\right)\right),$$

and ϵ_{k,g_i} can also be approximated as the following normal distribution:

$$\epsilon_{k,g_i} \sim N\left(\frac{n_{g_{i-1}}}{s_{g_{i-1}}}, \frac{n_{g_{i-1}}}{s_{g_{i-1}}}\left(1 - \frac{1}{s_{g_{i-1}}}\right)\right).$$

This means that e_{k,g_j} is a linear combination of independent normal random variables. As a result, e_{k,g_j} can also be approximated as a normal distribution.