SAND2019-3201C

Tracking Network Events with Write Optimized Data Structures

Justin Raizes*, Evan West*, Thomas M Kroeger*, Brian Wight*, Cindy Phillips*, Jon Berry*,
Michael Bender†, Rob Johnson‡
*Sandia National Labs {jraizes, ewest, tmkroeg, bjwrigh, caphill, jberry}@sandia.gov
†Stoney Brook University bender@cs.stonybrook.edu
‡VMWare Labs rob@cs.stonybrook.edu

***Abstract*—The basic action of two IP addresses communicating is still a critical part of most security investigations. Typically security tools focus on logging torrents of security events. Some more advanced environments will try to send the logs to a variety of databases. Unfortunately, when faced with indexing billions of events such databases are usually unable to keep up with the rate of network traffic. As a result, security monitors typically log with little to no indexing.**

Write-optimized data structures (WODS) provides a novel approach to traditional data structures. WODS use RAM to aggregate multiple insertions into a single write and as a result are able to ingest data 10 to 100 times faster while answering queries in a timely manner. Using a write optimized B-Tree known as a B^e-tree we developed a tool (called Diventi) to index layer 3 network activity from bro connection logs as well as netflow data.

During both 2017 and 2018 Diventi was used by members of the SCiNet network security team to investigate security incidents and track activities of specific IP addresses and subnets. In 2017 we ingested bro connection logs indexing over 6 billion events at rates above 104,000 events per second, and typically answering queries in milliseconds. In 2018 we adapted our tool to index based on netflow records, tapping directly into the network fabric without the need of a security monitoring tool like bro. During Super Computing 2018, we indexed over 4.3 billion events seeing rates above 160,000 events per second. Moreover we monitored the server and query performance and were able to show that our servers CPU and IO subsystem were barely taxed, with a peak CPU usage of 47% acrosss the entire show. Our IO in busy periods was typically less than 10% of the systems capacity. For queries with 2500 or fewer results our response times were less than 100 milliseconds. In our worst case one query got almost 1 million results in about 7.6 seconds.

***Index Terms*—security, networking, indexing, write optimized, IDS**

I. Introduction

Advanced security monitoring must juggle two opposing efforts. Sensor teams focus on collection and recording data as fast as possible, while analytic teams focus on understanding and analysis, which requires access across large swaths of data. If these analytics are to provide on-line monitoring to protect systems as they operate then these systems need to perform their analytics in a timely manner. Ideally these analytics should be able to see and use a wide view of the data collected but this hinders their responsiveness. We can store one second's worth of data in one second. However, searching a year's worth of data in one second is much more challenging. If our analysts can't use our data much of its value is lost. To put this into perspective, imagine an Internet without search engines.

Our research seeks to fill the gap between sensors and analytics to find efficient ways in which sensors can still record one second of data in one second in addition to organizing the data to ensure that analytics can query one year's worth of data in one second. As the size of data scales beyond primary storage (RAM), systems are faced with one of two choices: expire data or move to slower secondary storage and fall behind. Traditionally, these challenges have been tackled by expanding the amount of primary storage available using clusters of computers with lots of RAM. However, the data eventually catches up, overwhelming the amount of primary storage available. Recent work with Write-Optimized Data Structures (WODS) has shown that it is possible to ingest torrential streams of data using larger, less expensive secondary storage, while still maintaining timely queries.

In order to attempt to perform security monitoring on IP traffic we developed a tool we call Diventi that uses a write optimize B-Tree called a B^e tree [**?**] to index network events from either netflow data or bro connection logs.

At SuperComputing 2017, Diventi ingested bro connection logs indexing IP addresses across 600 gigabits per-second (Gbps) of monitored traffic on a basic Dell server. In three days, our system indexed over 6 billion events, while maintaining point query response times of milliseconds. In contrast, the SciNet security team used a cluster of 30 nodes to monitor traffic using a commercial log management tool.

In 2018, Diventi was configured to ingest netflow data directly from the tap and aggregation infrastructure. This enabled IP monitoring without depending on more complex IDS tools like bro. During this effort we ingested over 4.3 billion events in six days with a peak rate of over 161,256 events per second. Additionally our server's CPU 3% utilized on average and even during busy periods was typically less than 10% utilized. Our IO subsystem was also barely taxed with out IO channel rarely seeing numbers greater than 10% of what we saw during benchmark testing.

Finally we monitored query response time both on the server and empirically at the security team's CLI. All queries with fewer that 2500 results saw latencies of less that 85 milliseconds. In the worst cases our largest queries got almost

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

Fig. 1: B^e tree stores data points at each node. When a node fills up the cached entries are flushed to the lower nodes.

1 million results in less that 7.6 seconds.

II. Background

Inserting data into a traditional B-tree typically requires $O(\log_B N)$ writes to secondary storage per each insert. While many of these can be cached in RAM, as N grows to years worth of data the number of writes on each insert trend towards this bound. As a result these traditional data structures don't represent an efficient way to balance primary and secondary storage for tracking torrents of security monitoring data. Because of this security monitoring systems have typically shied away from indexing or used RAM based data structures, limiting the scope of data tracked.

In contrast, a write-optimized B-tree such as the B^e-tree uses buffers at each level in the tree to aggregate multiple inserts into each write. By buffering writes at internal nodes we improve write latency as much as 10 to 100 times while losing only a small constant factor in query performance [**?**]. Each node, of size B, is divided into two sections: pivots and cache. A tunable parameter, ϵ, takes values between 0 and 1, to determines the ratio between the two components. Pivots and child pointers take up B^ϵ space and the cache takes up the remaining space $(B - B^\epsilon)$. As elements are inserted, they are added to the buffer of the root node. When a buffer fills up, its contents are flushed to lower nodes, where the insertions are again buffered. Figure 1 shows how a typical B^e-tree works.

This behavior results in an insertion performance of $O(\frac{\log_B N}{B^{1-\epsilon}})$, as compared to a standard B-tree's $O(\log_B N)$. For perspective, parameters of $B = 1024$ and $\epsilon = \frac{1}{\ln 1024} \approx .14$ provide a speedup of $\approx 54x$ insertion rate over a standard B-tree.

In trade, the B^ϵ tree pays $O(\frac{\log_B N}{\epsilon})$ for queries instead of a standard B-tree's $O(\log_B N)$. Continuing with the example of $\epsilon = \frac{1}{\ln 1024} \approx .14$, a query takes only $\approx 6.93x$ longer for a B^ϵ tree. In our experiences last year at SuperComputing, queries typically took under 500 ms. From the practical perspective of a security monitoring system, this trade off is frequently the difference between being able to index events in near real-time or having no index at all. In this light the slower queries that use an existing index are far better than the many systems we have seen which run grep in parallel and take minutes to answer the same question [**?**].

Alternative solutions involving distributing workload over clusters of computers using tools such as MongoDb, Hadoop, or ElasticSearch have also been evaluated. On a cluster of 5 machines, researchers found that none of the tools were able to exceed 200 transactions per second after $100,000,000$ records (average record size 2866 bytes) had been ingested in a write-heavy workload [**?**]. The speed and scale of security events needing to be tracked in large scale networks like SCiNet necessitate the use of write optimized algorithms that enable out of core handling high speed data streams.

III. Methods and Engineering

During the Network Research Exhibition, we received net-flow data directly from the security teams tap and aggregation infrastructure and directly from the rscope IDS tool that created bro connection logs. Diventi then indexed this data in near real-time, tracking the rates at which events arrived. The hardware used is a basic Dell server, specifically a Dell 730 server equipped with 128 GB of RAM, one 16 core processor, and 5 SSDs aggregated into an 8 TB data store. Diventi was then used by members of the SCiNet security team during investigations to answer queries about specific IP addresses and the interactions they had on our network. Additionally for consistency we developed a standard test set of queries that were preformed as a benchmark test.

A. Data Schema

Our security monitoring events typically contain a notice about two IP addresses communicating to include timestamp, ports, protocols, and some sense for the number of packets and bytes in each direction. Our B^ϵ-tree stores these events in a key-value pairing. Since each event represents two IP addresses (IP A and IP B) communicating, two separate key-value pairs are inserted per event, to enable quick lookup of either IP.

We begin each key with an IP and timestamp to enable efficient searching for a given IP, possibly over a given time frame or possibly a given subnet of IPs. For example this index can quickly find activity from 1.2.3.4 for the last month, or all activity from sub-net 1.2.3.X. On the other hand searching for all activity from 1.2.3.X from last month would require doing the indexed search of 1.2.3.x and then manually filtering the entries that are in that time frame.

One common limitation of a write optimized data structure is the inability to detect collisions on inserts. This is because detecting if that key already exists would require a full traversal down the tree to verify that it didn't exist at any level. This would require significantly more IOs per insertion and would reduce the ingestion rate. We work around this limitation by filling out our key with the rest of the unique details from this event. This ensures that each unique security event is kept in a unique key and there are no collisions. Table I shows a summary of the data we have in our key.

The value holds many of the other relevant components of a network event. These include the duration of interaction, amount of data transferred, and any relevant tcp flags. Typically, security analysts care about the scale of data and packets transferred (e.g. 2 GB versus 2 bytes). Tools such as

Byte Range	Length	Field
0–3	4	IP A
4–11	8	Timestamp
12–14	2	Port A
15–18	4	IP B
19–21	2	Port B
22–22	1	Misc flags

TABLE I: Data fields for key

Bro record the exact values, with each of the 4 fields using 4 bytes. Since the complete data is kept in the original logs, our goal is simply to help the security analyst quickly assess whether a connection is of note. With this in mind, we record the magnitude of data transferred instead of its full value, using only a single byte per field, rather than 4. The reduction in value size reduces the amount of writing done per event, thus improving ingestion speed.

Finally we note that while our focus has been on indexing IP address, our code design has focused on clear abstractions between events, keys and values. This has made it easy to adapt our tool to ingesting both bro connection logs and netflow data. We believe this design will also enable easy adaption to indexing other security events such as URLs and e-mail addresses.

B. Context and Impact

In recent years, much effort has been focused on analysis of big data and network inspection. This work typically has the goal of creating a security system which, in an automated fashion: actively monitors the network, recognizes threats, and takes action to stop those threats. However, considerably less effort has been focused on providing that data in the context of network events in actionable times to these analytics. Without timely query responses and comprehensive data, these systems must either limit to scope of the data they consider or reduce their approach to a post event alert instead of active responses.

Berry and Porter [**?**] showed that state-of-the-art hashing and expiration methods can quickly degrade in performance as the data set needed to correctly find patterns of interest exceeds the size of available memory. They used the (non-write-optimized) reference implementation for the Firehose benchmark. When the working space could exactly hold the set of active keys, the reference implementation reported $2/3$ of the reportable keys (prematurely expiring the rest). When the working space could hold half the active keys, about $1/3$ of the reportable keys were reported. When only $1/4$ the active keys fit in memory, essentially nothing was reported. This motivates the need for tools that can efficiently use both RAM and secondary storage to increase storage capacity while still providing timely query responses.

Diventi fills this need by indexing data quickly for storage across both primary (RAM) and secondary (SSD) storage while still allowing for timely query responses. Secondary storage is both less expensive and available in larger quantities than primary storage, reducing costs significantly. The increased speed and storage capacity make more efficient use of the resources available and increase the scope of data analytics can consider. It is tools like Diventi that will help bridge the gap between sensors that can record torrents of data in one second and analytics that wish to consider years of that data in one second.

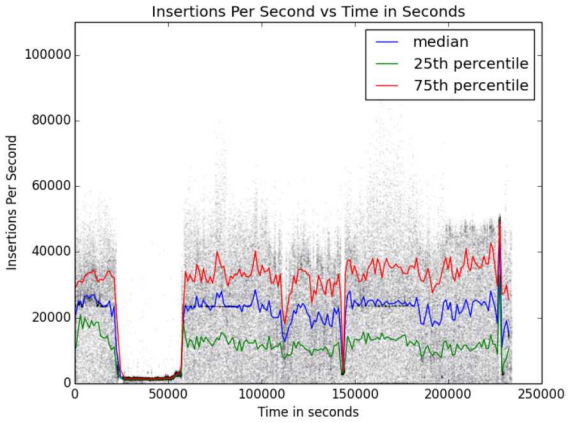

Fig. 2: Insertions per second as seen by Diventi during Super Computing 2017.

IV. Results from use at SciNet

Diventi has been used to index IP address at both SC 2017 and 2018. Here we present a summary of our experiences indexing a high speed stream of IP addresses and answering queries in support of security monitoring.

At supercomputing 2018, over the course of six days, Diventi indexed 4.3 billion events from NetFlow records. Throughout the event, the server was not stressed, typically seeing less that 10% utilization of CPU or IO capacity, and handling up to 161,256 events per second with not issues whatsoever. Query response times also provided prompt results for security operators. Taking less that 85 milliseconds for queries with less that 2500 events and in the worst case taking 7.6 seconds for a query that got almost 1 million results. Overall, the security team found Diventi to be a useful tool in protecting SciNet. But we would argue the a far great value for such a system will be when automated analytic tools are able to make broader use of the full picture that is provided by tools like Diventi.

V. Insertion and Query Performance

At Super Computing 2017, Diventi indexed bro-conn logs as 6.7 billion events with a maximum rate of 104,060 inserts per second. At supercomputing 2018, Diventi received Netflow data directly from the network infrastructure. Over the course of the event, Diventi ingested 4.3 billion events at a maximum rate of 161,256 events per second. Figures 2 and 3 show the insertions per second as seen by Diventi during SC17 and SC18 sampled every 5 seconds. The red and green lines represent the 25th and 75th percentile over every 5 minute period. The blue line represents the median for the same period.

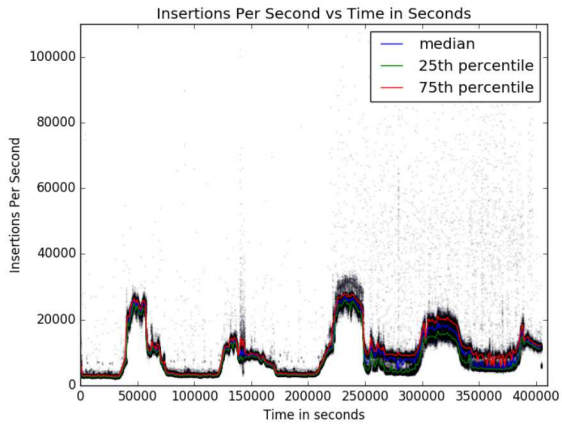

Fig. 3: Insertions per second as seen by Diventi during Super Computing 2018.

These two graphs show similar cycles of activity and inactivity. One key difference we see is that while 2018 has a higher peek rate, the scatter of points seems to be a bit wider spread in 2017. We speculate that the netflow data sent directly from our network hardware had little processing to do and reflected exactly what the network was seeing at that moment. While in 2017 our feed came from bro IDS sensors that had their own buffers and a good bit of processing that needed to be done before a connection log line was generated. As a result this additional processing added a smoother spacing to the rates at which we saw events.

A. *Query Response*

To monitor the user experience with Diventi we instrumented query response times on the server and periodically performed a standard set of queries while using the unix utility `time` to measure the total latency. The table below shows the results for our standard query set run in the last hours of SCiNet being live. In addition we list operational queries that were conducted in that same time-frame. These operational queries lack metrics for user side latency because the users were not using the `time` utility. What we can see from this data is that for any modestly sized result, say 2583 or less our response times are less that 85 milliseconds. Even more so for rather large queries our response times were still measured in single seconds. In short these response times provide a strong foundation for near real-time response for individual secure personnel. Moreover their the ability to support automated analytics in these timeframes can enable in-line analytics that could make use of a far greater situational awareness than typically see today.

VI. Benchmark Testing of our Server

Despite the rigorous demands of monitoring the entire SciNet 2018 network infrastructure, Diventi did not stress the single box it was running on at any point. In 2018, we typically saw less than 10% resource utilization even at points of active ingestion. Our CPU usage never exceeded 47% and on average was 3%. Our IO utilization at typical peaks was less than 10% of throughput we saw during performance tests of our server. TMK: **Evan to build graph of IO usage and CPU usage from vmstat**

TABLE II: Query result counts and latencies seen by the user cli and as seen on the Diventi server, for our standard query set and operational queries run in the last few hours.

Query	# Results	User (s)	Server (s)
x.x.106.8	0	0.078	0.017
x.x.106.1/24	0	0.058	0.0001
x.52.66	2	0.084	0.0001
x.x.52.1/24	2	0.063	0.0001
x.x.21.59	8	0.064	0.024
x.x.21.1/24	1250	0.074	0.015
x.x.128.132	209141	1.726	1.64
x.x.50.49	382703	2.76	2.65
x.x.122.73	721	-	0.020513
x.x.52.19	2583	-	0.040452
x.x.204.40	1216	-	0.013698
x.x.225.81	22710	-	0.216706
x.x.142.100	927344	-	6.165492
x.x.245.157	963450	-	7.554234

VII. Conclusions

Diventi's performance at supercomputing 2017 and 2018 demonstrates the viability of near real time indexing of network events and opens the door to running analytics on live network traffic data. While there are lots of efforts to collect security monitoring data quickly and lots of efforts to provide near real-time analytics that can respond to attacks, there is a big need to build the bridge between these two efforts to help ensure our analytics can make use of the largest situational awareness that our data and algorithms can provide. Diventi and our experiences during SC 2017 and 2018 have gone a long way to helping build such a bridge.

Acknowledgments Sandia National Laboratories is a multi-mission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC., a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA-0003525.