Themis: An I/O-Efficient MapReduce

George Porter and Mike Conley

Status Report, October 2014

1 Personnel

In addition to the PI listed above, this research is carried out in collaboration with former UCSD Professor (now at Google) Amin Vahdat. The Ph.D. student on this project is Mike Conley.

2 Research

Data-intensive computing is an increasingly important component for solving problems across a wide range of fields. Due to very large data sizes, these problems are often I/O-bound, and so minimizing the number of I/O operations is critical to improving their performance. Previous work within our group on TritonSort [9] has examined I/O-optimal computing aimed at the *sort* challenge problem. In the Themis project, we seek to generalize beyond simple sort to a address a wide variety of I/O-bound compute problems in a wide range of deployment scenarios.

2.1 "2-IO" Efficient Computing

The operational specification of the MapReduce programming model [6] requires that data presented to each reduction operation is sorted. A result by Aggarwal and Vitter [1] shows that every MapReduce system must perform at least two I/O operations per record whenever the amount of data exceeds the amount of memory in the cluster. We refer to a system that meets this lower-bound as having the "2-IO" property. Any data processing system that does not have this property is doing more I/O than it needs to. Existing MapReduce systems exceed the 2-IO limit to provide fine-grained fault tolerance and to handle variably-sized records. In the absence of failures, TritonSort exhibited the 2-IO property, and our goal with Themis is to also achieve this property. We are actively pursing the development of new features to Themis that will enable users to perform data processing tasks on data set sizes of an order of magnitude larger than existing systems, at a significant fraction of the capacity of the underlying hardware. We have reached this goal for circa 2009-era hardware clusters, and are now prototyping next-generation clusters based on faster network link speeds and fast, non-volatile storage devices. A key motivation for this research is understanding how best to design data-intensive computing systems targeted for cloud computing environments, especially those focused on the "cloud."

2.2 Recent progress

During the Spring 2014 review, we reported on two complementary efforts. The first was modifying Themis to support high-throughput non-volatile storage, rather than disk drive-based storage. The second effort involved porting Themis to run in a cloud computing environment. We are happy to report considerable progress on both of these efforts.

2.2.1 Supporting non-volatile storage

For this reason, we have updated Themis to support high-speed flash storage and higher-speed network links.

Themis was originally designed to support hard disk drive-based (HDD) storage, where drives incur large seek penalties every time the disk head is moved from one partition file to another. This seek penalty is particularly important in the map and shuffle phase, where map output records are arriving in an effectively random order. To minimize this seek penalty, data was kept in a large, application-level data structure called the *write chainer* [8]. The write chainer maintains an in-memory buffer per partition file, which can grow or shrink dynamically based on the skew in the underlying data. Whenever the disk is free, Themis writes the longest buffer to disk, which represents the largest IO operation possible at that time and maximizes HDD performance.

Amazon Web Services (AWS) and other cloud providers now offer a large number of instance types featuring flash-based storage. Flash drives do not impose the large seek penalties of disk drives, and so maximizing write sizes is far less important when Themis is deployed on SSD-based cloud infrastructure. We have modified Themis to support both SSD and HDD storage devices by making the write chainer mechanism an optional part of the MapReduce pipeline. A result of this optimization is a significant reduction in the memory footprint of Themis, freeing memory for other processing.

To further increase Themis's storage IO throughput, we changed from synchronous to asynchronous IO. The data path of a flash drive is generally more complex than a disk drive, due in part to a flash translation layer which maps OS-visible blocks to underlying physical flash blocks. Further, a single flash disk typically consists of multiple underlying flash memory chips. To maximize the bandwidth across those chips, Themis issues multiple concurrent asynchronous IO requests.

2.2.2 Migrating to the Cloud

The task of designing, building, and maintaining a dedicated compute cluster to support a given application represents an enormous source of cost and on-going expense. As a result, many choose instead to deploy their applications on cloud-computing platforms. Cloud providers such as Amazon Web Services (AWS) [3], Google Cloud Platform [7] and Microsoft Azure [4] offer nearly instantaneous access to configurable compute and storage resources that can grow and shrink in response to application demands and are ideal for supporting large-scale data processing tasks.

Yet achieving cost-effective processing in shared cloud environments is challenging for two primary reasons. The first challenge is the lack of control over what types of virtual machine (VM) instances the provider offers. When it comes to building out cloud infrastructure, providers are constrained in what they offer to end users: they typically must purchase commodity components, choose resources that meet the current (and anticipated) needs of a diverse set of tenants, and choose resources that meet the budgets of their customers. The end result of this internal, multi-level optimization process is a set of offerings that tend toward the least common denominator of consumer demand. End users must then configure the different types of VMs offered by the cloud provider into a performant and cost-efficient cluster for a target application. This task is non trivial—as of this writing Amazon offers 37 different types of VMs, differing in the number of virtual CPU cores, the amount of memory, the type and number of local storage devices, the availability of GPU processors, and the available bandwidth to other VMs in the cluster.

The second challenge one encounters is the gap between the cloud abstraction and the reality of what the provider actually offers. The abstraction offered by cloud providers is of an unlimited number of VMs accessing unlimited amounts of storage, all connected by a virtual private network [2]. Yet the reality is quite different: certain VM types are in short supply, and the performance of VMs, their storage devices, and inter-node network throughput and latency is unpredictable and variable.

As a result, choosing the right set of resources to optimize processing for a particular application is a challenge and requires knowing the *scaling* behavior of the virtual cluster infrastructure and its expected performance variability. The network is particularly challenging because it has fundamental implications for implementing balanced and efficient distributed computing. As the size of the cluster increases, overall cluster utilization and efficiency can drop, requiring more VMs to meet performance targets and driving up overall cost [5].

We have deployed Themis to the Amazon Web Services (AWS) Cloud. We found that provisioning the most cost-effective cluster within AWS requires significant profiling of the scaling behavior of each of the AWS VM offerings. In other words, one cannot directly predict overall performance based on descriptions of each of the VM types. Perhaps contrary to historical practice, smaller clusters of more specialized and higher-cost VMs outperform larger clusters of general purpose compute infrastructure, despite the fact that the general purpose VMs are more cost-effective individually. Today, the network and remote storage behavior of virtual clusters is sufficiently variable and limited that users are better off paying for more predictability among a smaller set of more performant servers. This is true from a cost perspective but also from a feasibility perspective: large, e.g. $O(100)$, clusters of newer configurations are difficult to obtain in practice. Today, scale-up seems to be better than scale-out for high-performance data processing.

2.3 Evaluation Methodology

After deploying Themis/TritonSort to the Amazon AWS Cloud, we were able to signficantly improve overall system efficiency and performance. The benchmark workload that we evaluate against is that of the annual Jim Grey Sorting Competition (`http://sortbenchmark.org`). This competition is divided into a number of categories that stress different aspects of efficient data processing. These include:

Category	**Measures:**
GraySort	time required to sort 100 TB of data.
CloudSort	dollar cost to sort 100 TB in the cloud.
MinuteSort	amount of data sorted in 60 seconds.
JouleSort	energy required to sort 10 GB, 100 GB, 1 TB, or 100 TB.

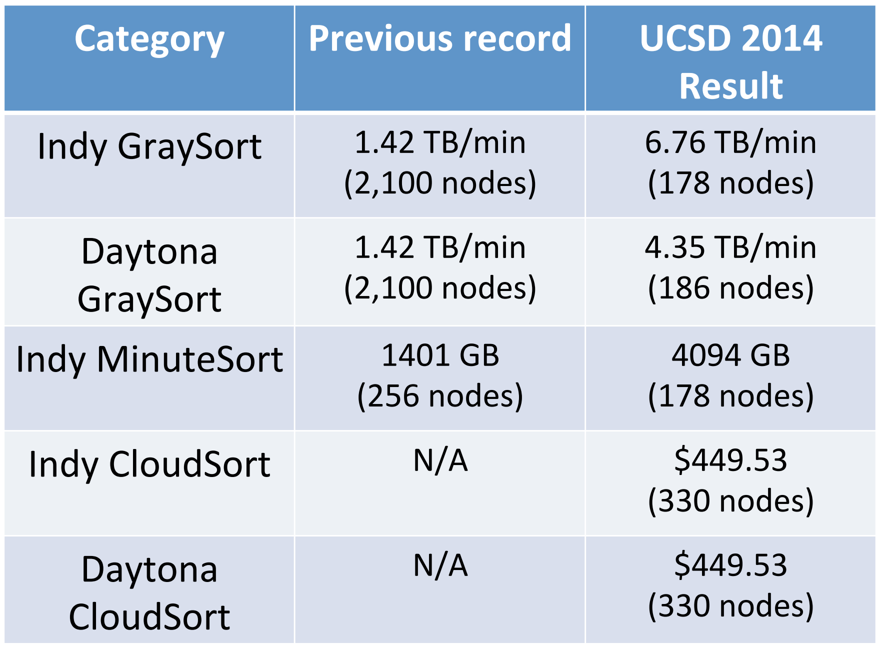
Each of these categories is further divided into two subtypes: “Indy” and “Daytona.” Entrants in the Indy variant can assume that the data consists of fixed-sized records, with 10-byte keys and 90-byte values, and further that the values of the keys are uniformly distributed across the possible keyspace. In contrast, the Daytona variant of each of the sort categories must be general purpose, supporting variable-sized records and records drawn from a skewed key distribution. CloudSort further stipulates that input and output data must be stored on persistent, replicated storage.

2.4 Results

The results of this effort are shown in Figure 1.

2.5 Project status

This CNS review marks the conclusion of this project.



Category	Previous record	UCSD 2014 Result
Indy GraySort	1.42 TB/min (2,100 nodes)	6.76 TB/min (178 nodes)
Daytona GraySort	1.42 TB/min (2,100 nodes)	4.35 TB/min (186 nodes)
Indy MinuteSort	1401 GB (256 nodes)	4094 GB (178 nodes)
Indy CloudSort	N/A	$449.53 (330 nodes)
Daytona CloudSort	N/A	$449.53 (330 nodes)

Figure 1: Performance of TritonSort/Themis on Amazon Cloud Infrastructure

References

[1] A. Aggarwal and J. Vitter. The input/output complexity of sorting and related problems. *CACM*, 31(9), Sept. 1988.

[2] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia. A view of cloud computing. *Commun. ACM*, 53(4):50–58, Apr. 2010.

[3] Amazon Web Services. `http://aws.amazon.com/`.

[4] Microsoft Azure. `http://azure.microsoft.com/`.

[5] J. Dean and L. A. Barroso. The tail at scale. *Commun. ACM*, 56(2):74–80, Feb. 2013.

[6] J. Dean and S. Ghemawat. MapReduce: Simplified data processing on large clusters. In *Proceedings of the 6th Conference on Symposium on Opearting Systems Design & Implementation - Volume 6*, OSDI'04, pages 10–10, Berkeley, CA, USA, 2004. USENIX Association.

[7] Google Cloud Platform. `http://cloud.google.com/`.

[8] A. Rasmussen, M. Conley, R. Kapoor, V. T. Lam, G. Porter, and A. Vahdat. Themis: An I/O efficient MapReduce. In *ACM SoCC*, 2012.

[9] A. Rasmussen, G. Porter, M. Conley, H. V. Madhyastha, R. N. Mysore, A. Pucher, and A. Vahdat. TritonSort: A balanced large-scale sorting system. In *NSDI*, 2011.