Efficient On-Demand Operations in Dynamic Distributed Infrastructures

Steven Y. Ko and Indranil Gupta
Department of Computer Science
University of Illinois at Urbana-Champaign
{sko, indy}@cs.uiuc.edu

ABSTRACT

In a large-scale distributed infrastructure, users and administrators typically desire to perform *on-demand operations* that act upon the most up-to-date state of the infrastructure. These on-demand operations range from monitoring the up-to-date machine properties in the infrastructure, to making Grid scheduling decisions for different tasks based on the current status of the infrastructure. However, the scale and dynamism present in the operating environment make it challenging to support these operations efficiently.

This paper discusses several on-demand operations that we have been studying, challenges associated with them, and how to meet the challenges. Specifically, we build techniques for 1) on-demand group monitoring that allows users and administrators of an infrastructure to query and aggregate the up-to-date state of the machines (e.g., CPU utilization) in a group or multiple groups, 2) an on-demand Grid scheduling strategy that makes scheduling decisions based on the current availability of compute nodes, 3) an on-demand Grid computation strategy that chooses the best algorithm for the current input data set among multiple algorithms available. We also present our ongoing work.

1. INTRODUCTION

The success of the Internet has brought to us the daily use of large-scale distributed infrastructures. Typical web users interact with data centers daily using web-based applications. Scientific communities utilize various computational Grids to perform their experiments. Moreover, the promises of "cloud computing" initiatives by industry leaders and academics such as the Cloud Computing Testbed at University of Illinois [6], Amazon [1], Google [10], and IBM [11], and HP [12] evidence that this trend will likely continue for the years to come.

We believe that one important class of operations in these large-scale infrastructures is *on-demand operations*, necessitated either by the need from users and administrators of the infrastructures or by the characteristics of workloads running on the infrastructures. Broadly, on-demand operations are the operations that act upon the most up-to-date state of the infrastructure. For example, a data center administrator typically desires to have on-demand monitoring capability for the data center. This capability of querying the up-to-date state (e.g., CPU utilization, software versions, etc) of the entire data center on-demand allows the administrator to understand the innerworking of the data center for troubleshooting as well as to make well-informed decisions for management tasks such as capacity planning and patch management. Another example is on-demand scheduling in computational Grids, where scheduling decisions often need to reflect the most current resource availability such as the availability of compute nodes. Thus, supporting these on-demand operations efficiently is both desirable and necessary.

The challenges in supporting such operations can be categorized into two areas - scale and dynamism - and each challenge appears in a variety of flavors unique to the specific on-demand operation at hand. For example, in the case of on-demand monitoring, scale comes not only from the physical number of machines in the infrastructure, but also from the number of attributes to monitor, and in the case of on-demand scheduling, from the number of tasks to process. Dynamism comes not only from the failures of services and machines, but also from the changes in resource availability, usage, workload, etc.

In this paper, we briefly examine some of our work on design and implementation of on-demand operations. We discuss some examples of on-demand operations, different kinds of dynamism and scale associated with them, and how to address the challenges. We also discuss our ongoing work.

We believe that exploring on-demand operations reveals the many faces of dynamism and scale present in large-scale infrastructures. By doing so, we can revisit well-known aspects of dynamism and scale, and also uncover new aspects of the two characteristics that were previously overlooked. Ultimately, the collective knowledge accumulated from this exploration can form the basis of reasoning when designing new large-scale infrastructures or services.

2. OUR WORK SO FAR

In this section, we present a brief summary of on-demand operations that we have been working on.

2.1 On-Demand Group Monitoring

Several distributed aggregation systems have been proposed in the past, in order to monitor large-scale infrastructures such as data centers [26, 31, 23]. These systems provide a query interface for aggregating machine properties such as memory utilization, CPU utilization, etc. Aggregation queries play a key role as the summary view of the infrastructure often yields more insights than the individual view of each machine.

We argue that such a monitoring system should be 1) group-based, as users and administrators typically desire to monitor groups of machines - these groups are specified implicitly and could be either static (e.g., web servers, databases, etc) or dynamic (e.g., machines with CPU utilization $> 50\%$), and 2) on-demand, as the users should be able to query the most up-to-date data within a group. Although previous systems have developed techniques to support aggregation queries, they target "global" aggregation; each machine in the whole infrastructure receives and answers every aggregation query, wasting bandwidth if a query actually targets only a group of machines. There is currently no distributed technique for aggregating data from a group (or multiple groups) in a bandwidth-efficient manner without contacting all machines in the infrastructure.

This on-demand group aggregation adds a new dimension to the challenges of scale with respect to the number of machines and the number of attributes to monitor - the number of groups. This new dimension has been recognized in the context of multicast [3]. However, since aggregation employs different techniques such as in-network processing, we need a solution that is tailored towards the harmony with aggregation techniques.

We have built a system called Moara that builds per-group aggregation trees to address these challenges [19]. Moara builds a group tree that spans over the members of the group, so that each query does not have to flood the whole infrastructure as some global aggregation schemes do. Aggregation is also done over the group tree, resulting in a faster query response time than global aggregation schemes. In addition, multiple groups can share one tree to ensure scalability in terms of the number of groups.

The challenge of dynamism comes from two sources, group churn and workload. To be more specific, the size and composition of each group can change over time due to joining and leaving of machines as well as changes in the attribute-value space. In addition, a user may need to aggregate data from different groups at different times, so we can expect that the query rate and target might vary significantly over time. Thus, a group aggregation system has to be adaptive to the group churn and changes in the user workload.

This dynamism in group and workload poses an interesting trade-off in terms of bandwidth usage. On the one hand, we can save per-query bandwidth cost by utilizing group trees. However, group churn requires us to maintain each group tree, which costs bandwidth. Thus, if the query rate is high enough for a group tree, it might be better to maintain the group tree to save per-query bandwidth. However, if the group churn rate is high, it might be better to flood each query and not maintain the group tree, because the maintenance cost might be too high.

Considering this trade-off, Moara includes a distributed maintenance protocol adaptive to the current group query rate and group churn rate. Our experiments have shown that Moara reduces bandwidth usage compared to both the scheme that floods the whole infrastructure and the scheme that maintains group trees all the time.

Moara, as any distributed system subjected to dynamism in the environment, suffers from the CAP dilemma [4], which states that it is difficult to provide both strong consistency guarantees and high availability in failure-prone distributed settings. Moara treads this dilemma by preferring to provide high availability and scalability, while providing eventual consistency guarantees on aggregation results. This philosophy is in line with that of existing aggregation systems such as Astrolabe [26] and SDIMS [31]. Moara could also allow the use of metrics proposed by Jain et al. [15, 16] in order to track the imprecision of the query results.

2.2 On-Demand Scheduling

The current trend in both academia and industry shows that the workload is being shifted from computation-oriented tasks to data-intensive tasks in computational Grids. Many scientific communities such as Astronomy, Earth Science, and Biophysics already write data-intensive applications that access and generate multi-TB data [25]. Also, NSF has announced CluE initiative [7] to explore data-intensive computing with HP, Intel, and Yahoo's Cloud Computing Testbed as well as Google-IBM clusters.

We argue that any scheduling algorithm for data-intensive tasks has to be on-demand, i.e., it has to make scheduling decisions based on the most up-to-date state such as the availability of compute nodes (or "workers") and the characteristics of the current input data, due to the dynamism present in the environment. We have developed scheduling algorithms that address the dynamism of resources [18] and the dynamism of workload [17]. We elaborate each further below.

First, we have developed a series of on-demand worker-centric scheduling algorithms for data-intensive tasks [18], which considers the availability of workers as the primary criterion. Our algorithms make a scheduling decision for a task to a worker only when the worker can start executing the task immediately. We term this worker-centric scheduling as opposed to task-centric scheduling (e.g., [29, 5, 25]), where a scheduler assigns a task to a worker without considering whether or not the worker can start executing the task immediately after the task assignment.

Worker-centric scheduling is important because it avoids two inherent problems of task-centric scheduling. First, task-centric scheduling might overload some workers with tasks, because the scheduler is agnostic to the number of tasks already assigned to each worker. Second, conditions at a worker during scheduling time of a task may be different from the conditions at the worker during execution of the task, because each task usually waits in the worker's task queue for a while. Worker-centric scheduling avoids these two problems, since each worker executes the newly-assigned task immediately. Our experiments show that the proposed worker-centric algorithms can achieve utilization of workers more than 90%, and outperform the state-of-the-art task-centric algorithm roughly up to 20% in terms of task completion time.

Second, in order to address the dynamism in the input workload, we have developed a system called HoneyAdapt, a peer-to-peer system for workers to choose the best algorithm on-demand as the input data changes [17]. This is possible since often times there are multiple algorithms that perform differently depending on the input data. For example, there are many sorting algorithms (e.g., quick sort, bubble sort, etc) that perform differently depending on how the given input data is organized. Our technique is based on the foraging behavior of honeybees that choose the best quality honey among multiple sources adaptively. Our experiments with sorting algorithms show that our technique can roughly achieve up to 40% reduction in sorting time.

3. RELATED WORK

On-Demand Monitoring:. MON [20] is among the first systems that support on-demand operations for large-scale infrastructures. It supports propagation of one-shot management commands. In PlanetLab, there are other management tools that support on-demand operations, such as vxargs and PSSH [24]. Most commercial monitoring and management systems such as HP OpenView, IBM Tivoli, and CA Unicenter are also centralized in nature. Thus, these tools do not address scalability and expressive queries simultaneously.

Several academic efforts propose distributed systems for aggregating data in large systems. Astrolabe [26] provides a generic aggregation abstraction, but uses a single static tree and hence has limited scalability with the number of metrics. SDIMS [31] constructs multiple trees for scalability with the number of metrics, but assumes a single group of the entire system. PIER [13] supports recursive SQL-style queries, but does not leverage in-network aggregation. Huebsch *et al.* [14] present a way to optimize global aggregation queries, while Moara optimizes multiple group-based aggregation trees. Seaweed [23] focuses on dealing with data unavailability, which is different from Moara's focus. MON [20] supports one-shot queries and constructs query trees on-demand, but does not support expressive queries. Finally, Ganglia [21] uses a single hierarchical tree, but collects all data without in-network aggregation.

On-Demand Scheduling:. Spatial Clustering [22] creates a task workflow based on the spatial relationship of files in the input data set. It improves data reuse and diminishes file transfers by clustering together tasks with high input-set overlap. Tasks in a cluster are then assigned to workers that sit on a same site. The great reduction in file transfers causes an important decrease in execution time. Two drawbacks to this approach are that (1) it cannot handle new jobs arriving asynchronously and (2) it is application specific.

Storage Affinity [29] also addresses file reuse for data-intensive applications. The algorithm computes a data affinity value for each task, for each site, according to the input set of each task and the data currently stored at a site's networked storage. In this way it finds the task+site pair with largest data affinity (common bytes), which is chosen as the next schedule. To address inefficient CPU assignments, they propose replicating tasks, also based on the storage affinity. The algorithm shows improved makespan and good data reuse, specially when compared to the XSufferage [5] scheduling heuristic.

Decoupling data scheduling from task scheduling was proposed by Ranganathan *et al.* [25]. The work evaluates four simple task scheduling mechanisms and three simple data scheduling mechanisms. Best results are obtained when a task is scheduled to a site that has a good part of its input data already in place, combined with proactive replication of a popular input data-set to a random/least-loaded site. Note that execution time improves with the data replication schemes due to the data set popularity distribution, which was set to be geometric –not all grid applications will have the same characteristic.

A few Grid scheduling algorithms can be considered as on-demand scheduling since they make scheduling decisions based on the current status of the Grid. Examples include a pull-based scheduler proposed by Viswanathan *et al.* [30] and theoretical work by Rosenberg *et al.* [27, 28].

4. ONGOING WORK AND CONCLUSION

We are currently working on a new on-demand operation and continuing to improve our current systems.

- **On-Demand Log Processing**: On-demand log processing refers to the ability of processing the most up-to-date log entries that are distributed. Although there are many applications that generate and process logs [2, 9, 8, 32]), the approaches are typically centralized. Every log is collected from the machine that generates the log, and then processed by a centralized algorithm. This centralized approach limit the capability of on-demand processing of logs because of the time involved in collection. We are working on a distributed approach that tries to minimize the overall processing time by intelligently deciding the timing and the number of log entries to ship.
- **State Management for Moara**: Since Moara maintains many states related to groups, it is important to have a mechanism to garbage-collect these states. We are investigating several policies for this purpose. Some possibilities include: 1) to garbage-collect each state after a timeout expires, 2) to keep only the last k groups queried, and 3) to garbage-collect the least frequently queried group.
- **Incorporating Grid Scheduling Algorithms into Cloud Computing Testbed (CCT)**: We are also working to incorporate our HoneyAdapt and scheduling strategies into CCT.

In conclusion, we believe that on-demand operations reveal various aspects of dynamism and scale of large-scale infrastructures such as data centers. Thus, supporting on-demand operations are necessary not only from the perspective of users and administrators, but also from the perspective of designers and architects of the infrastructure and services running over them.

5. REFERENCES

[1] Amazon Web Services. `http://www.amazon.com/gp/browse.html?node=3435361`.

[2] P. Bahl, R. Chandra, A. Greenberg, S. Kandula, D. A. Maltz, and M. Zhang. Towards Highly Reliable Enterprise Network Services via Inference of Multi-level Dependencies. In *Proceedings of ACM SIGCOMM*, 2007.

[3] M. Balakrishnan, K. Birman, A. Phanishayee, and S. Pleisch. Ricochet: Lateral Error Correction for Time-Critical Multicast. In *Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2007.

[4] E. Brewer. Towards Robust Distributed Systems (Invited Talk). In *Proceedings of the 9th ACM Symposium on Principles of Distributed Computing*, July 2000.

[5] H. Casanova, D. Zagorodnov, F. Berman, and A. Legrand. Heuristics for Scheduling Parameter Sweep Applications in Grid Environments. In *Proceedings of the 9th Heterogeneous Computing Workshop*, 2000.

[6] HP, Intel, Yahoo in cloud tie-up. `http://news.bbc.co.uk/2/hi/technology/7531352.stm`.

[7] NSF Cluster Exploratory (CluE) program. `http://www.nsf.gov/cise/clue/index.jsp`.

[8] R. Fonseca, G. Porter, R. H. Katz, S. Shenker, and I. Stoica. X-Trace: A Pervasive Network Tracing Framework. In *Proceedings of the 4th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2007.

[9] D. Geels, G. Altekar, P. Maniatis, T. Roscoe, and I. Stoica. Friday: Global Comprehension for Distributed Replay. In *Proceedings of the 4th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2007.

[10] Google App Engine. `http://code.google.com/appengine/`.

[11] Google and IBM Join in "Cloud Computing" Research. `http://www.nytimes.com/2007/10/08/technology/08cloud.html`.

[12] HP Labs Dynamic Cloud Services. `http://www.hpl.hp.com/research/cloud.html`.

[13] R. Huebsch, B. Chun, J. M. Hellerstein, B. T. Loo, P. Maniatis, T. Roscoe, S. Shenker, I. Stoica, and A. R. Yumerefendi. The Architecture of PIER: an Internet-Scale Query Processor. In *Proceedings of the Conference on Innovative Data Systems Research (CIDR)*, 2005.

[14] R. Huebsch, M. Garofalakis, J. M. Hellerstein, and I. Stoica. Sharing Aggregate Computation for Distributed Queries. In *Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data (SIGMOD)*, 2006.

[15] N. Jain, D. Kit, P. Mahajan, P. Yalagandula, M. Dahlin, and Y. Zhang. PRISM: Precision-Integrated Scalable Monitoring (extended). Technical Report TR-06-22, UT Austin Department of Computer Sciences, March 2006.

[16] N. Jain, D. Kit, P. Mahajan, P. Yalagandula, M. Dahlin, and Y. Zhang. STAR: Self Tuning Aggregation for Scalable Monitoring. In *Proceedings of the 33rd International Conference on Very Large Databases (VLDB)*, June 2007.

[17] S. Y. Ko, I. Gupta, and Y. Jo. Novel Mathematics-Inspired Algorithms for Self-Adaptive Peer-to-Peer Computing. *ACM Transactions on Autonomous and Adaptive Systems (TAAS), to appear*, 2008.

[18] S. Y. Ko, R. Morales, and I. Gupta. New Worker-Centric Scheduling Strategies for Data-Intensive Grid Applications. In *Proceedings of the ACM/IFIP/USENIX 8th International Middleware Conference (Middleware)*, 2007.

[19] S. Y. Ko, P. Yalaganula, I. Gupta, V. Talwar, D. Milojicic, and S. Iyer. Moara: Flexible and Scalable Group-Based Querying System. In *Proceedings of the 9th ACM/IFIP/USENIX Middleware, to appear*, 2008.

[20] J. Liang, S. Y. Ko, I. Gupta, and K. Nahrstedt. MON: On-demand Overlays for Distributed System Management. In *Proceedings of the 2nd USENIX Workshop on Real, Large Distributed Systems (WORLDS)*, 2005.

[21] M. L. Massie, B. N. Chun, and D. E. Culler. The Ganglia Distributed Monitoring System: Design, Implementation and Experience. *Parallel Computing*, 30(7), July 2004.

[22] L. Meyer, J. Annis, M. Mattoso, M. Wilde, and I. Foster. Planning Spatial Workflows to Optimize Grid Performance. Technical report, GriPhyN 2005-10, 2005.

[23] D. Narayanan, A. Donnelly, R. Mortier, and A. Rowstron. Delay Aware Querying with Seaweed. In *Proceedings of the 32nd Very Large Data Bases Conference (VLDB)*, 2006.

[24] PlanetLab: Contributed Software. `https://wiki.planet-lab.org/twiki/bin/view/Planetlab/ContributedSoftware`.

[25] K. Ranganathan and I. Foster. Decoupling Computation and Data Scheduling in Distributed Data-Intensive Applications. In *Proceedings of the 11th International Symposium on High Performance Distributed Computing (HPDC-11)*, 2002.

[26] R. V. Renesse, K. P. Birman, and W. Vogels. Astrolabe: A Robust and Scalable Technology for Distributed System Monitoring, Management, and Data Mining. *ACM Transactions on Computer Systems*, 21(2):164–206, May 2003.

[27] A. L. Rosenberg. On Scheduling Mesh-Structured Computations for Internet-Based Computing. *IEEE Transactions on Computers*, 53(9), September 2004.

[28] A. L. Rosenberg and M. Yurkewych. Guidelines for Scheduling Some Common Computation-Dags for Internet-Based Computing. *IEEE Transactions on Computers*, 54(4), April 2005.

[29] E. Santos-Neto, W. Cirne, F. V. Brasileiro, and A. Lima. Exploiting Replication and Data Reuse to Efficiently Schedule Data-Intensive Applications on

Grids. In *Proceedings of the 10th International Workshop on Job Scheduling Strategies for Parallel Processing (JSSPP)*, 2004.

[30] S. Viswanathan, B. Veeravalli, D. Yu, and T. G. Robertazzi. Design and Analysis of a Dynamic Scheduling Strategy with Resource Estimation for Large-Scale Grid Systems. In *Proceedings of the 5th IEEE/ACM International Workshop on Grid Computing (Grid 2004)*, 2004.

[31] P. Yalagandula and M. Dahlin. A Scalable Distributed Information Management System. In *Proceedings of ACM SIGCOMM*, 2004.

[32] Y. Zhang and V. Paxson. Detecting Stepping Stones. In *Proceedings of the 9th USENIX Security Symposium*, 2000.