Augmenting MapReduce with Active Volunteer Resources

R. Benjamin Clay, Zhiming Shen, Xiaosong Ma, Xiaohui Gu
Department of Computer Science
North Carolina State University
{rbclay,zshen5}@ncsu.edu, {ma,gu}@csc.ncsu.edu

ABSTRACT

The migration of interactive workloads, such as desktop applications, into clouds presents significant opportunities for efficiency improvements. The bursty and interactive nature of such workloads makes it challenging to aggressively consolidate them on multi-tenant systems. In such scenarios, utilizing *residual* or wasted CPU cycles is particularly appealing, which helps amortize the cost of physical hardware as well as improve energy efficiency.

In this work, we propose to harvest residual CPU cycles in interactive clouds by aggregating distributed, opportunistic compute capacity with batch MapReduce jobs. To accomplish this, we leverage a hybrid cluster design consisting of a small set of *dedicated* batch nodes supplemented by a larger pool of *volunteers*, which executes and gives priority to foreground interactive workloads. Further, we outline the design of an autonomous management framework, which manipulates a hybrid cluster to achieve runtime and energy performance goals under dynamic residual resource availability. We implement our cluster design on top of Apache Hadoop, and highlight our proposed techniques to cope with the diverse performance-limiting factors in this challenging environment.

1. INTRODUCTION

Interactive cloud offerings are expanding, including virtual computing laboratories, Google Docs, and online collaboration tools. These new platforms provide users with great convenience in accessing popular applications and tools, with little software/hardware/licensing overhead. Meanwhile, such systems also yield the potential for significant *residual*, or unused, resources, due to over-provisioning and the bursty, unpredictable nature of interactive workloads. For example, techniques such as virtual machine (VM) packing are unlikely to be performed aggressively with interactive cloud workloads, due to users' response time requirements. The dynamic behavior of interactive user sessions also render mechanisms such as VM migration, which are expensive and inflexible on short timescales, too cumbersome for such contexts. Regardless of the level of workload consolidation available, there will likely exist significant amounts of residual resources from interactive workloads due to the conservative allocation scheme they require. By harnessing such resources, cloud service users and providers will benefit from more cost-effective hardware usage, as well as considerable energy savings, as the *incremental* energy consumption of running additional applications using residual CPU tends to be low [1].

To capture and use residual resources from interactive cloud systems, one needs to address several challenges: the foreground workloads may be highly bursty and demand high responsiveness; users employing identical environments or application may have drastically different behavior; and, user sessions may span minutes to hours, with VMs initiated and terminated very dynamically. To construct a cost-effective, resilient, and non-intrusive computing service to collect the ever-changing leftover cycles from interactive cloud sessions is not trivial.

We propose "hybrid cloud computing" to approach this problem. Our proposed execution model is "hybrid" in two dimensions. First, two types of workloads, interactive and batch, co-execute on physical cloud nodes. Rather than consolidating workloads in a symmetric way as in most VM packing work, our proposed model adopts an asymmetric consolidation, where the foreground, interactive workloads are latency-sensitive, bursty, and overall resource-light, while the background, batch workloads are throughput-oriented, relatively consistent, and resource-intensive. By co-locating these two modes of computation, the cost to cloud customers of both workloads can potentially be lowered.

Second, to enable the background distributed computing framework to tolerate the dynamic and highly volatile nature of residual resources, in terms of both resource amount and individual VM's availability, we propose to enable distributed processing on a hybrid system, with *dedicated* nodes providing data storage and service, as well as a certain level of baseline operation and performance, and *volunteer* nodes aggressively harvesting residual resources from interactive cloud tasks.

Such a hybrid cloud computing infrastructure can solve most of the challenges mentioned earlier in utilizing dynamic residual cloud resources. However, for a user with a resource-intensive batch job to process in the cloud, it is challenging to determine the actual configuration of her "recycled" cloud. Both the foreground and background workloads running on cloud systems demonstrate high diversity, and it is impossible for a batch job owner to acquire knowledge on interactive workloads running on the same physical cloud. In this work, we address a specific problem: *given a fixed amount of dedicated resources and a background batch workload, how should we decide the number of volunteer nodes to enlist, which supplement the dedicated nodes in executing this workload?* We outline our design of an autonomous optimizing framework that dynamically scales the volunteer portion of a hybrid cloud considering the foreground residual resource level, the scalability of the background job, and energy and/or node pricing factors.

We demonstrate our hybrid cluster design and management framework using MapReduce as the batch execution model by modifying Hadoop, a widely used MapReduce implementation. MapReduce is an ideal computing paradigm for harnessing idle cycles, thanks to its popularity and built-in data management and fault tolerance. However, we believe our approach is broadly applicable to most embarrassingly-parallel batch frameworks.

2. APPROACH

Allocating a MapReduce cluster using residual resources in an interactive cloud is challenging. The dynamic nature of interactive users means that residual resource availability changes constantly. Further, *ephemeral* nodes that host interactive users are typically available for short periods, ranging from minutes to hours.

Storage of persistent data is an important aspect of the background environment which must be carefully tailored to accommodate ephemeral node transience. Physical machines are typically powered off after serving requests, and may not re-appear in the near future. Even with data replication enabled, a large batch of terminations could cause data loss. Additionally, the short lifespan of ephemeral nodes makes the I/O cost of using a new volunteer potentially large because data must be loaded or unloaded when the volunteer joins or leaves.

In light of these conditions, we enabled a hybrid MapReduce cluster design, similar to prior work [2], composed of a fixed set of dedicated nodes and a variable number of volunteer VMs. Volunteers rely on residual CPU cycles on ephemeral nodes, enforced using a work-conserving hypervisor scheduler. Because persistent data storage and transfer is such a challenging issue, we chose to host Hadoop's HDFS only on the dedicated nodes, which ensures volunteers are agile and do not risk data loss, but requires careful consideration of I/O demands.

Given a fixed group of dedicated cloud nodes with per-node storage and **single batch MapReduce workload**, we attempt to allocate and maintain a volunteer pool of ideal size with respect to the performance goal, assuming **no prior knowledge** of this workload. First, such knowledge is generally not available to the cloud system management framework. Second, we found MapReduce workloads' runtime behavior often heavily depends on input data. Finally, dynamic cloud conditions require constant adjustment to achieve ideal performance.

To accomplish such runtime self-configuration tasks, we have developed an autonomous management framework designed to elastically scale a volunteer pool to achieve a given performance goal. Our framework continuously monitors both the MapReduce cluster and the ephemeral pool, and periodically re-estimates the ideal volunteer cluster composition.

In our proof-of-concept design and implementation, we mainly investigate three factors that have the most potential to limit the background MapReduce job's scalability and cost-effectiveness: dynamic foreground workloads, I/O bottlenecks and memory restrictions. In order to predict foreground CPU demand, we employ an approach described in prior work [3]. To account for the effects of I/O bottlenecks, we choose to observe the I/O landscape and build a model in an online fashion. Finally, we use a similar online approach to adjust the number of Map slots used under memory-limited volunteer conditions.

3. ACKNOWLEDGEMENTS

This work has been supported in part by NSF grants 0546301 (CAREER), 0915861, 0937908, and 0958311, in addition to Google Research Awards, IBM Research Awards, a Graduate Merit Award from the College of Engineering at NC State University, as well as a joint faculty appointment between Oak Ridge National Laboratory and NC State University. We appreciate the helpful input from the anonymous reviewers and especially thank our shepherd, Jeff Mogul, for his valuable help. Any opinions expressed in this paper are those of the authors and do not necessarily reflect the views of the NSF or U.S. Government.

4. REFERENCES

[1] J. Li, A. Deshpande, J. Srinivasan, and X. Ma. Energy and performance impact of aggressive volunteer computing with multi-core computers. MASCOTS'09, 2009.

[2] H. Lin, X. Ma, J. Archuleta, W.-c. Feng, M. Gardner, and Z. Zhang. Moon: Mapreduce on opportunistic environments. HPDC '10, 2010.

[3] Z. Shen, S. Subbiah, X. Gu, and J. Wilkes. Cloudscale: Elastic resource scaling for multi-tenant cloud systems. SOCC '11, 2011.