EME: An automated, elastic and efficient prototype for provisioning Hadoop clusters on-demand

Feras M. Awaysheh, Tomás F. Pena, and José C. Cabaleiro
Centro Singular de Investigación en Tecnoloxías da Información (CiTIUS), Universidade de Santiago de Compostela, Spain
{*feras.awaysheh, tf.pena, jc.cabaleiro*}*@usc.es*

Keywords: MapReduce, Hadoop, Big Data, Cloud computing, Prototyping, Elastic Computing, Quality of Service.

Abstract: Aiming at enhancing the MapReduce-based applications Quality of Service (QoS), many frameworks suggest a scale-out approach, statically adding new nodes to the cluster. Such frameworks are still expensive to acquire and does not consider the optimal usage of available resources in a dynamic manner. This paper introduces a prototype to address with this issue, by extending MapReduce resource manager with dynamic provisioning and low-cost resources capacity uplift on-demand. We propose an Enhanced Mapreduce Environment (EME), to support heterogeneous environments by extending Apache Hadoop to an opportunistically containerized environment, which enhances system throughput by adding underused resources to a local or cloud based cluster. The main architectural elements of this framework are presented, as well as the requirements, challenges, and opportunities of a first prototype.

1 INTRODUCTION

In recent years, under the explosive increase of global data, there has been an increasing emphasis on Big Data, business analytics, and "smart" living and work environments. Despite the success of large-scale commodity clusters like Apache Hadoop (White, 2012), this continuous data stacks require computational power far beyond the capability of the cluster workstations (Herodotou et al., 2011) in the high season intervals (i.e. when massive amount of jobs are submitted to the cluster). Naïve solutions may propose to add new computing nodes and scaling-out the tasks to distributed systems, like in hybrid cloud computing. Hence, two major concerns need to be addressed. First, it could be necessary additional investments on infrastructure (increasing the Total resources Cost of Ownership (TCO)), or paying to rent cloud instances. Second, the cost of data movement to and from the cloud over the Internet, which can be expensive and time-consuming.

As a comprehensive Big Data analytics platform, Hadoop has become the de-facto technology for storing and processing data-intensive applications (DIA) (Hashem et al., 2015) and large-scale data analytics. These processes mainly use parallel data tasks operating, based in the MapReduce (MR) paradigm (Dean and Ghemawat, 2008). Thus, many frameworks aim at proposing scalable methods to support existent MR-based applications by enhancing the throughput using different techniques, while keeping in mind the QoS and the cost-benefit ratio. For instance, scaling Hadoop capacity horizontally and vertically (scale-up/out techniques), as in (Chen et al., 2014; Nghiem and Figueira, 2016) or adopting special hardware accelerators, as in (Honjo and Oikawa, 2013) and (Chen et al., 2012).

Opportunistic Computing e.g., Enterprise Desktop Grids (EDG) could provide free compute cycles by harnessing idle CPUs in what is known as cycle scavenging technique (Anderson, 2004), enabling the use of all available resources within the enterprise. Using this underused resources to extend the Hadoop cluster with low-cost, efficient and elastic desktop machines will improve throughput, making the most of the IT infrastructure. In exchange, such use would indeed increase the system complexity, so a powerful scheduler and resource manager would be required. Though, we argue that using cutting edge technologies like Linux containers can tackle that issue.

In this paper, we present the EME project, short for Enhanced Mapreduce Environment. EME extends Hadoop with an adaptive hybrid (dedicated and non-dedicated) heterogeneous task allocation and provisioning on-demand. Its architecture combines High Throughput Computing (HTC) and Docker containers with large-scale dedicated clusters, leveraging all available resources. This project aims for a new

hybrid environment that can elastically tune the resources participation, optimizing the cluster throughput without the need to add new and costly machines or rent external cloud services.

2 BACKGROUND

There have been some proposals going towards managing, processing and maintaining massive datasets in what is called data-intensive processing with MapReduce Applications (MRA). This section presents a brief background on MR programming model and the Volunteer Computing systems alongside with related work and our main motivation to start this project.

1. MapReduce model.

   Ever since its introduction, MR framework has grabbed much attention in its ability to cope with parallel computing applications using a parallel data approach to process large volumes of data (terabytes or petabytes in size), which are typically referred to as Big Data. Apache Hadoop, the MR open source implementation, was naturally designed to be easily scalable, but it was not designed to be auto-scale or elastic. Many studies have proposed a resizable compute capacity in both local clusters (Ananthanarayanan et al., 2012) and the cloud (Dahiphale et al., 2014).

2. Volunteer and Opportunistic Computing. Volunteer Computing (VC) is a form of network-based distributed computing, which allows public participants to donate their idle workstations and help to run computationally expensive projects (Durrani and Shamsi, 2014). Opportunistic Computing (OC) is a computer technique that make use not only of the resources available in the local machine or cluster, but can also opportunistically scale-out on other resources of the environment, including those on underused desktop computers, in a reliable and secure way (Conti et al., 2010). Two major characters differ these paradigms. First, the resource ownership, i.e., VC take advantage of volunteer resources donated by participants while OC make use of untapped internal resources. Second, geographical topology, where OC follows an intranet (LAN) topology, while VC mainly follows Internet protocols (WAN). Correspondingly, HTCondor (Thain et al., 2005) is an open-source cluster resource manager aimed at Distributed High Throughput Computing (HTC) on collections of owned resources (EDG).

2.1 Related work

Many related studies in the literature classify the performance of MR data intensive computing into dedicated and non-dedicated resources. For instance, MRA++ (Anjos et al., 2015) is a heterogeneous-dedicated methodology example that proposed a solution by grouping the machines according to their computational capabilities, calculating the execution time, and distributing Map task according to these capabilities. This proposal holds some drawbacks. First, it is not applicable for opportunistic environments, and volatility of data is ignored. Second, it does not take into account data placement (locality). Further examples include homogeneous-non-dedicated methodology (Ji et al., 2013) and heterogeneous-non-dedicated methodology (Jin et al., 2012). Apache Myriad (Apache Software, 2017) is a case for multi-tenant shared services clusters. Myriad is an open source framework for scaling the YARN cluster into Mesos, which allows to expand or shrink the capacity of the cluster managed by YARN in response to events, as per configured rules and policies. Though, it differs from our proposal in that it works only on dedicated resources, like on a typical Data Center.

Correspondingly, some research using hybrid resource architecture has been carried out, where volunteer computing systems are supplemented by a set of dedicated nodes to reduce the cost. Such research includes MOON (Lin et al., 2010), which discussed the problem of how to allocate a small set of dedicated nodes to volunteer computing systems to offer reliable MR services on hybrid resource architectures, adopting the LATE algorithm (Zaharia et al., 2008). However, MOON focused on a single job and worked on homogeneous environments only. Also, their solution needs a third part to manage the resource availability. By comparison, our study will follow hybrid resource architecture by adding some dedicated nodes to non-dedicated cluster in not only heterogeneous computing environment but heterogeneous operating systems as well. This will be achieved by using Docker based containers, which will allow us to build and distribute applications on heterogeneous nodes, thereby clearly differentiating our work from others.

2.2 Motivation

Nowadays, Hadoop resource elasticity and on-demand provisioning have not yet fully exploited by the present state-of-the-art MR execution architecture. We note a lack of studies that examine, from a resource perspective, the effect of resizable compute capacity (i.e., elastic resource provisioning)

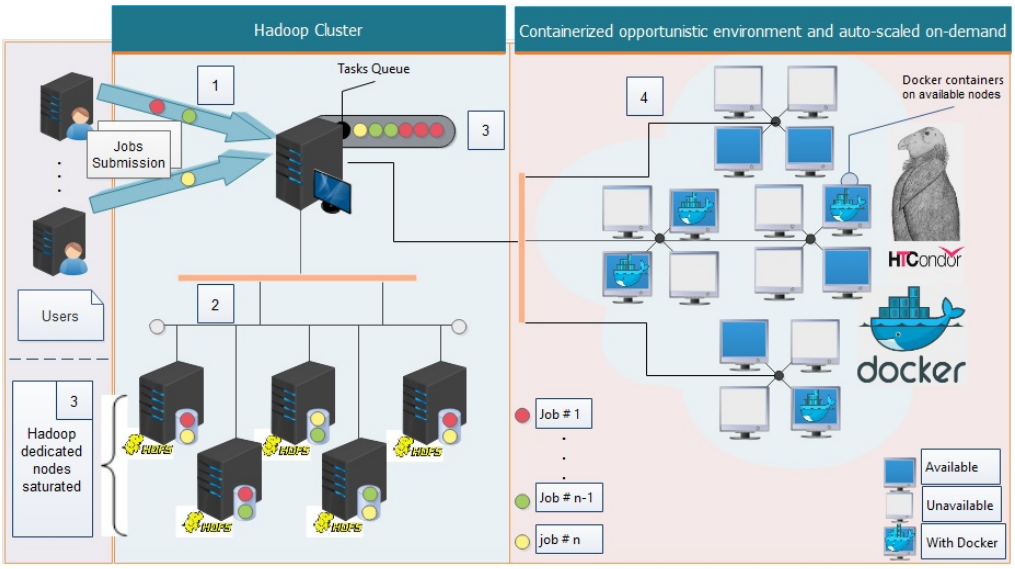

Figure 1: EME Proposed Framework Architecture

on MR applications performance. However, in this project we are committed to design, build and evaluate a hybrid distributed computing (dedicated and non-dedicated resources) architecture. We will use heterogeneous clusters, with differences in both computing power and availability (opportunistic computing), to extend Hadoop's MR task scheduling in a variety of environments and operating systems, using containerized based clusters on-demand.

The key challenges we are addressing in our work can be summarized as the following:

- Defining the Hadoop cluster threshold where resources are stretched to the limit (big data workloads start queuing).
- Efficiently execute MR operations on non-dedicated resources (isolated in containers).
- Opportunistically use resources that may be lightly loaded for long periods of time within an enterprise
- Elastically provisioning Hadoops Yarn on-demand with low-cost nodes.
- Enabling the two environments work harmoniously for the benefit of business and academia
- Making this framework automated (auto-scale), fast and extensible to the prior Hadoop Yarn architecture.

3 EME: FRAMEWORK ARCHITECTURE

Prototyping a distributed application like MR is considered a hard and time-consuming task. In this section, we will introduce EMEs design, goals and architecture overview.

3.1 Design overview

The proposed framework presented in Figure 1 consists of three major components. First are the users, EME's framework targeting a composite (multi-users/multi-jobs) architecture. Second, a dedicated environment, i.e., a typical Hadoop cluster that follows master/slave technique, where Yarn is responsible for scheduling, monitoring and re-executing failure tasks on slave machines. Third, a non-dedicated environment, which could be controlled using VC middleware like BOINC (Kurochkin and Saevskiy, 2016). However, our approach adopts an elastically and horizontally scaling (scaling-out) of resources in a controlled LAN to opportunistically execute MR tasks on collections of distributed, underused and owned resources on-demand. So, to manage and allocate the non-dedicated nodes, HTCondor (to create an EDG within LAN, i.e., an HTCondor pool) can be used in EME. Thus, the proposed integrated architecture will reduce the system complexity and avoid system degeneration due to overloaded Hadoop master. Anyway, a lightweight scheduler has to be implemented in HTCondor, which only consider container deployment inside the opportunistic pool.

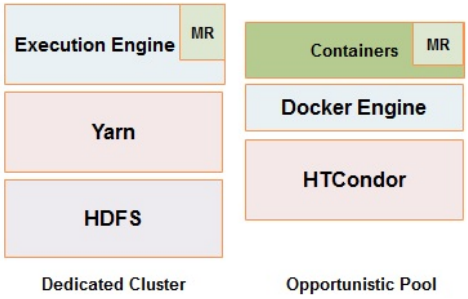

Figure 2: EME architecture layers and abstractions

Figure 2 presents a layered architecture and abstractions of both environments. For the opportunistic environment, an operating system-level virtualization environment (Docker containers) will be implemented, with HTCondor as its execution engine. This will provide not only an efficient use of the opportunistic resources, but also an isolated fault tolerance environment with high scalability potentials.

With this additional nodes, the throughput will be significantly improved. So, with a minimal cost of deployment, EME will improve scalability and optimize idle resources utilization, which would imply, as another direct impact, a greater return on infrastructure investments.

3.2 Fault tolerance

The basic abstraction of Hadoop architecture is directed to cluster and supercomputing paradigms. In these environments, faults are less harming as most nodes are reliable and will operate flawlessly for an extended period. Though, the failure of a node is still a concern. Some techniques like task replication, speculative execution of tasks, and heartbeats may limit the threat on most scenarios. However, the enterprise grid (non-dedicated) environment may seem more vulnerable. Accordingly, an advanced technique that supports fault-tolerance, while minimizing its impact on performance is a must.

In our work, we will concentrate on the enhancement of fault-tolerance on the opportunistic pool. Keeping in mind that no single point of failure is predicted at that environment, we propose two approaches for solve this problem. On the one hand, tuning the heartbeat may enable the NodeManager in the opportunistic resources to register dynamically on the dedicated ResourceManager (see Figure 3). Additionally, the nodes history (availability) will be added as an additional policy to determine the global resources availability.

On the other hand, the results obtained in the non-dedicated environment will be checked using the majority voting technique (Moca et al., 2011). This will add two main advantages to this scenario. At first, majority voting will require a minimum of three containers task replication to be implemented, which directly impact the fault tolerance by enhancing the task failure recovery. Second, leaving the result submission (the work done flag) until we collect two out of three results for each work from different workers, will provide high reliable results.

4 DISCUSSION

Docker provides a systematic way to automate the faster deployment of variety applications and libraries inside portable containers (isolated environment in operating systems kernel). Also, Container as a Service (CaaS) is considered as a particular case of IaaS without the hypervisor layer for better performance. Hence, EME proposes idle nodes as a viable way to deploy an opportunistic environment using disposable containers (deleted after executing the task) for provisioning the Hadoop cluster.

On the other hand, HTCondor will provide a consistent environment that continuously schedules new containers when it is possible and quickly gives resources back to the user when required. In our proposed architecture, Docker will provide a runtime service that ensures three aspects. First, isolation between the MR-tasks and other resources operations with minimal configuration on the HTCondor pool. Second, secure containers deployment with lightweight virtualization that runs on near bare-metal performance. Finally, an extensible interface that copes with Hadoop Yarn, where no rewrite for the MR applications is required.

As Figure 1 illustrate, the framework can be divided into four main phases that can be classified as follows:

1. Job submission: The job/jobs are submitted to the Hadoop master by the users via a job submission manager. The job manager at the master will place these jobs into a global job pool (task queue), where tasks wait to be batched.
2. Task allocation: Yarn manages task allocation, status monitoring, and reallocation failure tasks at the dedicated environment. Each dedicated slave shall run a NodeManager and a DataNode daemon. Yarn will start an ApplicationMaster per job, which asks the master (ResourceManager) for resources. The ResourceManager will contact the NodeManagers, and will provide the required resources in the form of containers. The

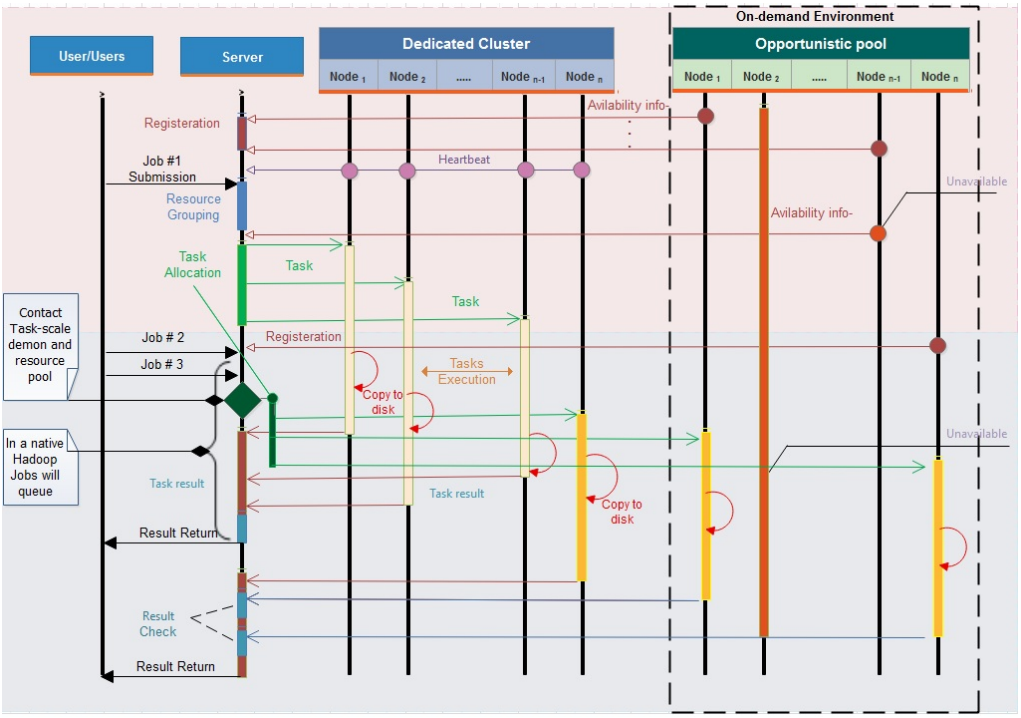

Figure 3: Framework sequence diagram

ApplicationMaster will run the jobs in the allocated containers till the cluster reaches a predefined threshold where resources are fully utilized, and no more tasks can be processes.

3. Cluster saturation: Under this circumstance, tasks start queuing at the job pool. The Yarn decision maker will use the global ResourceManager and will ask for an offer from the available (registered) resources at the opportunistic environment (i.e., in a pull approach).
4. Opportunistic resources Provisioning: At this point, the ResourceManager asks a special daemon (ContainerLauncher) to start containers (Docker) at the available opportunistic resources. These containers will run a NodeManager daemon. The new containers will refresh the ResourceManager offering these resources for the ApplicationMaster, which only exist in the dedicated environment.

Hence, the container launcher will create several Docker containers on the idle nodes for executing the independent task chunks in parallel. The ApplicationMaster will run these jobs in the allocated opportunistic containers, and the results will be sent back to the dedicated cluster via a result manager, which is in charge of result checking. Figure 3 shows a sequence diagram that explains the task execution phases in a sequence action timeline.

CONCLUSIONS AND FUTURE WORK

Over the past decade, Big Data analytics and data-intensive applications industry witness many enhancements regard large-scale processing. Apache Hadoop, with its ecosystem, is the dominant platform and moving toward becoming a comprehensive OS-like platform. However, current MapReduce implementations and resource managers are unable to employ lightly loaded resources within their controlled network, which is not yet fully exploited in the MapReduce state-of-the-art architectures.

To that end, EME project aims at improving data-intensive computing performance in both local and cloud-based clusters. EME offers a hybrid resource architecture (dedicated and non-dedicated) that supports heterogeneous capabilities by extending Hadoops cluster beyond its dedicated nodes, with elastic and low-cost resources on-demand. In particular, employing an opportunistically container-based cluster with auto-scale capabilities. EME will allow to automatically adapting the cluster size to the resources demand. This hybrid architecture aims to minimize the capital expenditure on large private infrastructures and to reduce operational costs. Additionally, it improves the MapReduce infrastructure throughput, performance, and QoS, helping to meet

job deadlines.

This paper describes the EME project blueprint, as well as the ongoing efforts behind it. Our work is being developed with Hadoop Yarn as the supporting resource manager, but we believe that the ideas presented in this research can easily be adapted to other resource management frameworks, for instance, Apache Mesos and Docker Swarm. As it is usual in the literature, we will start prototyping EME in a virtualized cluster and, when proving its usefulness, test it in a bare-metal environment. Additionally, it is expected to deploy an enterprise desktop grid, and to develop an opportunistically, elastic resource allocation scheduler to be integrated within its architecture.

ACKNOWLEDGEMENTS

This work has been partially funded by the Spanish Ministry of Science and Innovation with the Project No. TIN2016-76373-P and for the Xunta de Galicia with the Project No. GRC2014/008, the Consellería de Cultura, Educación e OU (accreditation 2016-2019, ED431G/08), the European Regional Development Fund (ERDF), the European Network HIPEAC, and the Spanish Network CAPAP-H.

REFERENCES

Ananthanarayanan, G., Douglas, C., Ramakrishnan, R., Rao, S., and Stoica, I. (2012). True elasticity in multi-tenant data-intensive compute clusters. In *Proc. 3rd ACM Symposium on Cloud Computing*, page 24.

Anderson, D. P. (2004). Boinc: A system for public-resource computing and storage. In *Grid Computing, 2004. Proceedings. Fifth IEEE/ACM International Workshop on*, pages 4–10. IEEE.

Anjos, J. C., Carrera, I., Kolberg, W., Tibola, A. L., Arantes, L. B., and Geyer, C. R. (2015). MRA++: Scheduling and data placement on MapReduce for heterogeneous environments. *Future Generation Computer Systems*, 42:22–35.

Apache Software (2017). Apache Myriad. `http://myriad.apache.org/`. Accessed: 2017-01-30.

Chen, K., Powers, J., Guo, S., and Tian, F. (2014). Cresp: Towards optimal resource provisioning for MapReduce computing in public clouds. *IEEE Trans. on Par. and Distributed Systems*, 25(6):1403–1412.

Chen, L., Huo, X., and Agrawal, G. (2012). Accelerating MapReduce on a coupled CPU-GPU architecture. In *Proc. of the Int. Conf. on High Performance Computing, Networking, Storage and Analysis*, page 25. IEEE Computer Society Press.

Conti, M., Giordano, S., May, M., and Passarella, A. (2010). From opportunistic networks to opportunistic computing. *IEEE Communications Magazine*, 48(9).

Dahiphale, D., Karve, R., Vasilakos, A. V., Liu, H., Yu, Z., Chhajer, A., Wang, J., and Wang, C. (2014). An advanced MapReduce: cloud MapReduce, enhancements and applications. *IEEE Transactions on Network and Service Management*, 11(1):101–115.

Dean, J. and Ghemawat, S. (2008). MapReduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113.

Durrani, M. N. and Shamsi, J. A. (2014). Volunteer computing: requirements, challenges, and solutions. *Journal of Network and Computer Applications*, 39:369–380.

Hashem, I. A. T., Yaqoob, I., Anuar, N. B., Mokhtar, S., Gani, A., and Khan, S. U. (2015). The rise of Big Data on cloud computing: Review and open research issues. *Information Systems*, 47:98–115.

Herodotou, H., Dong, F., and Babu, S. (2011). No one (cluster) size fits all: automatic cluster sizing for data-intensive analytics. In *Proceedings of the 2nd ACM Symposium on Cloud Computing*, page 18. ACM.

Honjo, T. and Oikawa, K. (2013). Hardware acceleration of Hadoop MapReduce. In *Big Data, 2013 IEEE International Conference on*, pages 118–124. IEEE.

Ji, Y., Tong, L., He, T., Tan, J., Lee, K.-w., and Zhang, L. (2013). Improving multi-job MapReduce scheduling in an opportunistic environment. In *Cloud Computing (CLOUD), 2013 IEEE Sixth International Conference on*, pages 9–16. IEEE.

Jin, H., Yang, X., Sun, X.-H., and Raicu, I. (2012). Adapt: Availability-aware MapReduce data placement for non-dedicated distributed computing. In *Distributed Computing Systems (ICDCS), 2012 IEEE 32nd International Conference on*, pages 516–525. IEEE.

Kurochkin, I. and Saevskiy, A. (2016). Boinc forks, issues and directions of development. *Procedia Computer Science*, 101:369–378.

Lin, H., Ma, X., Archuleta, J., Feng, W.-c., Gardner, M., and Zhang, Z. (2010). Moon: MapReduce on opportunistic environments. In *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing*, pages 95–106. ACM.

Moca, M., Silaghi, G. C., and Fedak, G. (2011). Distributed results checking for MapReduce in volunteer computing. In *Parallel and distributed processing workshops and Phd Forum (IPDPSW), 2011 IEEE international symposium on*, pages 1847–1854. IEEE.

Nghiem, P. P. and Figueira, S. M. (2016). Towards efficient resource provisioning in MapReduce. *Journal of Parallel and Distributed Computing*, 95:29–41.

Thain, D., Tannenbaum, T., and Livny, M. (2005). Distributed computing in practice: the Condor experience. *Concurrency and computation: practice and experience*, 17(2-4):323–356.

White, T. (2012). *Hadoop: The definitive guide*. " O'Reilly Media, Inc.".

Zaharia, M., Konwinski, A., Joseph, A. D., Katz, R. H., and Stoica, I. (2008). Improving MapReduce performance in heterogeneous environments. In *Osdi*, volume 8, page 7.