2015 IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing

SMART: An Application Framework for Real Time Big Data Analysis on Heterogeneous Cloud Environments

Julio C.S. dos Anjos[1], Marcos D. Assunção[3], Jean Bez[1], Claudio Geyer[1], Edison Pignaton de Freitas[1]
Alexandre Carissimi[1], João Paulo C. L. Costa[2], Gilles Fedak[3], Felix Freitag[4], Volker Markl[5], Paul Fergus[6], Rubem Pereira[6]

[1]*Federal University of Rio Grande do Sul, Brazil*
Institute of Informatics - PPGC
{jcsanjos, geyer, jean.bez, edison.pignaton, asc}@inf.ufrgs.br

[2]*University of Brasilia, Brazil*
joaopaulo.dacosta@ene.unb.br

[3]*Inria, LIP, ENS Lyon, France*
marcos.dias.de.assuncao@ens-lyon.fr, gilles.fedak@inria.fr

[4]*Universitat Politècnica de Catalunya, Spain*
Felix@ac.upc.edu

[5]*Technische Universitat Berlin, Germany*
Volker.markl@TU-Berlin.de

[6]*Liverpool John Moores University, England*
{p.fergus, r.pereira}@ljmu.ac.uk

***Abstract*—The amount of data that human activities generate poses a challenge to current computer systems. Big data processing techniques are evolving to address this challenge, with analysis increasingly being performed using cloud-based systems. Emerging services, however, require additional enhancements in order to ensure their applicability to highly dynamic and heterogeneous environments and facilitate their use by Small & Medium-sized Enterprises (SMEs). Observing this landscape in emerging computing system development, this work presents Small & Medium-sized Enterprise Data Analytic in Real Time (SMART) for addressing some of the issues in providing compute service solutions for SMEs. SMART offers a framework for efficient development of Big Data analysis services suitable to small and medium-sized organizations, considering very heterogeneous data sources, from wireless sensor networks to data warehouses, focusing on service composability for a number of domains. This paper presents the basis of this proposal and preliminary results on exploring application deployment on hybrid infrastructure.**

I. Introduction

Human activities are increasingly supported by computing devices, which can collect and store data about their behavior almost on a continuous basis. Similarly, the environment (both built and natural) is gradually been monitored by sensing devices that sample environmental aspects and generate large volumes of data. According to IDC[1], by 2020 there will be around 40 Zettabytes (40,000,000 Petabytes) of data that will require processing of some sort. Cloud computing has increasingly been used as a platform for business applications and data processing [1].

Big-data processing is an approach with specific characteristics which distinguish it of from other applications, such as volume, variety, velocity and veracity [2]. A large volume of data (volume) coming from multiple sources can enter the cloud under different formats (variety), and can demand processing in real-time (velocity) with high levels of accuracy (veracity). The data volume handled by Big Data analysis often requires processing capabilities beyond those that current IT infrastructure can provide. Specifically related to this concern, data streaming is an important technique to support incoming data with fast input as continual streams generate large volumes of data. It is not unusual to use resources from multiple data centers and, increasingly more common, from several clouds for deploying clusters for Big Data analysis [3].

MapReduce (MR) [4] is a programming framework adopted by many companies for Big Data processing, that executes "map, merge and reduce" data transformations. It addresses applications only based in batch model, normally in homogeneous environments such as large clusters in data centers. Hadoop [5], a popular MR implementation, is considered an industrial standard to Big Data, but it does not provide services that can be composed and combined in multi-cloud or hybrid infrastructures to support different types of applications. Other transformations, such as event-driven systems are hence necessary.

[1]IDC's Digital Universe Study, sponsored by EMC, December 2012.

978-1-5090-0154-5/15 $31.00 © 2015 IEEE
DOI 10.1109/CIT/IUCC/DASC/PICOM.2015.29

This work considers a large variety of data sources, ranging from wireless sensor nodes instrumenting open and indoor environments to large corporate databases, passing by social networks and broadcast media, where there is a clear need for standardization. Observing this large domain spectrum, this work proposes a modular framework for Big Data analysis called Small & Medium-sized Enterprise Data Analytic in Real Time (SMART) that aims to simplify the deployment of Big Data services by Small & Medium-sized Enterprises (SMEs). SMART takes advantage of cloud, multi-cloud and hybrid infrastructures to provide support for SME service operation, and does not need to aggregate data in a single data center for Big Data analysis. It provides a secure and flexible cloud-based system capable of providing different types of services that can be combined to address specific needs of multiple application domains. This paper specifically builds on previous efforts made on deploying MR applications to provide SMART with an approach that enables application deployment on hybrid infrastructure.

The rest of this paper is structured as follows. Section II examines related work. In Section III, there is a description of the main characteristics of the proposed system model. Section IV describes the evaluation criteria, methodology, experiments and preliminary results. The conclusion and directions for future work are summarized in Section V.

II. Related Work

Previous work can be divided into the following topics: frameworks for Big Data analysis, and techniques for managing data and application deployment in hybrid infrastructure and across multiple clouds.

A. Frameworks for big data analysis

MR is a programming framework that abstracts the complexity of parallel applications by partitioning and scattering data sets across hundreds or thousands of machines, and by bringing computation and data closer together [5]. *Map* and *Reduce* phases are handled by the programmer, whereas the *Shuffle* is performed while a task is being carried out. The data is serialized and distributed across machines that compose the Distributed File System (DFS). The application executions are represented as a Directed Acyclic Graph (DAG) under a batch processing model, which can provide high-latency response when applied to stream processing where data arrives constantly to be processed [6].

Streaming systems are event-driven and their behavior differs from that of batch systems [7]. Resilient Distributed Datasets (RDDs) have been presented as an extension to the Spark cluster computing engine [8]. An RDD provides a storage abstraction that avoids replication by using lineage for fault recovery, *i.e.*, the events are grouped into micro-batches. The RDD is kept in memory as a distributed shared memory abstraction and its programming produces operations with "map, filter and join" and enables interactive data mining. Although RDDs are best suited for batch applications that apply the same operation to all elements of a data set; they are less suitable for applications that make asynchronous fine-grained updates to shared state [8].

Storm [6], [9], one of the most popular frameworks for real-time processing, offers very low latency and has mechanisms that guarantee that all events are processed, as well as an extension called Trident, which provides basic operators and state management. Trident evaluates performance characteristics for individual types of nodes in a cluster and periodically executes workload re-balancing.

MR, Spark and Storm have been conceived to handle specific application models. SMART framework, on the other hand, offers services that can be composed to support applications that handle large volumes of data coming from a large number of diverse sources. The advantage of having these services in the cloud is clear as they are intended to support applications from different domains according to their demands. Thus, the computational environment must be as flexible as possible to support the execution of applications according to their required application model.

B. Hybrid infrastructure and multi-cloud

Organizations are increasingly relying on infrastructure from multiple providers as a means to increase tolerance to failures and avoid provider lock-in. When considering multiple clouds (*i.e.* hereafter also termed as multi-cloud), application deployment becomes complex as each individual cloud may have specific configuration parameters [10], and its resource availability and utilization can change dynamically. There is therefore a need for automatic configuration of complex cloud services at different abstraction levels. Users need means for efficiently mapping the computing requirements of their services to available resources. The lack of knowledge about the underlying infrastructure can lead to inefficient allocations where either allocated resources are not fully used or the Quality of Service (QoS) of applications is compromised due to allocating insufficient resources. As optimal allocation is difficult to achieve, an approximation strategy is generally acceptable [11].

Enterprises and governments often organize their data across multiple cloud sites or availability zones in order to maintain resource proximity; create data stores with organizations that share common goals; and keep data replicas across regions for redundancy purposes. However, under certain scenarios data needs to be analyzed globally. When considering MR, one way of doing this is to aggregate data in a single data center, and another is to execute individual instances of MR jobs on each data set separately and then aggregate the results [12].

Jayalath *et al.* [12] introduced G-MR, an implementation of Hadoop MapReduce for processing geo-distributed data set across multiple data centers. It is possible, for instance, to have multiple execution paths for performing a MR job, and the performance can be quite different. Popular MR open source packages like Hadoop, however, do not support this feature and the majority of Cloud Service Providers (CSPs) normally do not provide bandwidth guarantees to massive data transfers across data centers [13].

To optimize data storage across multiple clouds, a brokering algorithm has been proposed [14]. The algorithm considers the cost to maintain one object in a cloud provider; the failure probability and QoS associated with each Service Level Agreement (SLA) with a cloud provider. An object is a target data, without particular size or defined type. The primary goal is to find the optimal chunk placement according to the user's needs and budget. An expected availability represents M objects in each data center, this number determines the expected failure of the object in each data center. The study evaluates two parameters of each cloud provider, namely the failure probability and the cost per object. Objects are replicated across multiple sites considering these metrics, but the proposed solution does not identify network overhead caused by data transfers; an important factor when data sizes approach the exabytes.

Write Once Read Many (WORM) is a common assumption for data access for many Big Data applications, specially those that adopt the MR approach. A convenient approach for Big Data processing involving several data centers is to replicate data across different CSPs. However, the performance variability of cloud resources such as network can lead to bottlenecks [15], [16], and under such conditions the best strategy is to minimize data transfers. A study shows that there are two main approaches for modeling complex infrastructure [17], namely *analytic models* that use low-level details with workload characterization to predict performance; and *sampling methods* that do not require a priory knowledge about the underlying infrastructure. As information about network bandwidth, topology and routing strategies are not available to users of public clouds, the authors introduce a sample-based modeling technique that employs agents to monitor the environment. The agents are deployed on Virtual Machines (VMs) in each CSP where the applications are running. A *decision manager* considers how the transfer paths are established between source and destination. Transfer can be done directly from a node to a data center or use multiple paths across intermediate data centers. The data transfers are intra-site data replications due to dedicated links among data centers of a same CSP. The scientific applications interact with an API to provide data transfers over a WAN. A *monitor agent* monitors the environment and reports measurements to the decision manager. The measurements include bandwidth throughput between data centers, CPU load, I/O speed and memory status of VM nodes. The *decision manager* periodically updates the weights across the paths with these measurements.

Heterogeneous-Aware Tiered Storage (HATS) aims to improve I/O performance in Hadoop MR implementations [18]. HATS performs data placement in accordance with I/O throughput and device capacity. Each different device is a Hadoop Distributed File System (HDFS) instance in a *DataNode*. A *DataNode* with storage technologies different from the usual receives a different data size, according to its performance characteristics. The data placement concept creates policies to consider *network proximity*, *tier-awareness* and *hybrid* approaches. *Network proximity* considers retrieving replicas from nearest rack to reduce network traffic. The *tier-aware* policy ensures that a node stores a single replica even if the node has multiple HDFS instances and retrieves data from the fastest available tier. The approach for SMART can be seen as a mix of these two policies.

SALSA, a framework for configuration orchestration of services in multiple clouds [10], provides a model for application configuration and deployment of multiple types of services. The configuration information supports several levels of cloud services, such as applications, deployment relationships at multiples software stacks and the association between service units and configuration capabilities. The configuration capabilities are captured from registered services (cloud services and specifications of topology services) or user specifications. SALSA has a *service unit orchestrator* for multiple configuration services for each configuration task group. The VM creation is a process separates from other software levels. The configuration capability dependencies determine the relationships between service units. *Meta information* contains abstract nodes with generic types of service units that implement the virtual nodes. Each *service unit orchestrator* runs independently and interacts with a cloud service orchestrator. Although the framework enables heterogeneous configurations, there is not a mechanism to evaluate the performance and the user workloads to adapt load-balance in cloud computing.

Table I summarizes the main frameworks and techniques used for Big Data analysis, and compares them against the SMART framework.

III. INFRASTRUCTURE MODEL

Different cloud infrastructures have their own configuration parameters, and the availability and performance of offered resources can change dynamically due to several factors, including the degree of over-commitment that a provider employs. In this context, solutions are needed for the automatic configuration of complex cloud services. Cloud infrastructure comprising heterogeneous hardware environments may need the specification of configuration parameters at several levels such as the operating systems, service containers and network capabilities [10]. As users, who need to execute applications, may not know how to map their requirements to available resources, this lack of knowledge about the cloud provider infrastructure will lead either to overestimating or underestimating required

Table I
FRAMEWORKS AND TECHNIQUES FOR BIG DATA ANALYSIS

	Geo-Distributed Data	Consider Coast	Failure Probability	Network Overhead	I/O Throughput	Device Capacity	Replicate Objects	Minimize Transfers
G-MR	X							
Brokering Algorithm	X	X	X				X	
WORM	X			X				X
HATS	X			X	X	X	X	
SALSA	X							
SMART	X	X	X	X	X			X

capacity; both are equally bad as the former leads to waste of resources whereas the second sacrifices QoS.

Hybrid infrastructure, where there are many cloud providers with heterogeneous environments and configurations, often needs to use an orchestrator to manage the results and data input from users. The orchestrator must be decentralized [10] in order to improve data distribution in the network. The infrastructure enables the use of highly heterogeneous machines. When considering the use of a public cloud to extend the capacity of a community cloud, or desktop grid, several scenarios and data strategies are possible. The extent to which a set of data-distribution strategies is applicable to a given scenario depends on how much bandwidth is available. If one considers MR, two distinct DFS implementations may be required to handle data distribution in two scenarios, namely low-bandwidth and high-bandwidth.

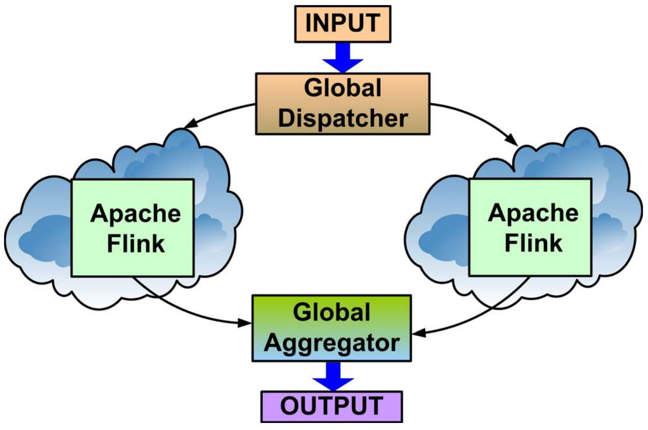

Figure 1. SMART architecture.

Building on previous work performed on MR for hybrid environments [19], [3], Figure 1 illustrates the solution proposed here to model a hybrid system which depicts a *Global Dispatcher* and *Global Aggregator* to be used on the infrastructure for services that use multiple data abstractions. The *Global Dispatcher* located outside the cloud has middleware functions for handling task assignment, and management of user-provided data. It is a centralized data storage system that manages policies for splitting data and distributing it in accordance with the needs of each system. The working principle is similar to a publish/subscribe service in which the system obtains data and publishes computing results [3]. The *Global Aggregator* obtains data output from both systems and merges them in order to obtain the final data set.

Apache Flink, formerly known as Stratosphere [20], is the base infrastructure of the SMART framework as depicted in Figure 2. Its flexible pipeline enables several map-reduce and extended functions like Map, MapPartition, Reduce, Aggregate, Join and Iterative. It can be used in order to allow this cloud extension. The setting will be transparent to users because a middleware in a top level abstracts the complexity away from the users.

The different layers of the stack build on top of each other and raise the abstraction level of the program representations they accept:

- The API layer implements multiple APIs that create operator DAGs for their programs. Each API needs to provide utilities (serializers, comparators) that describe the interaction between its data types and the runtime. All programming APIs are translated to an intermediate program representation that is compiled and optimized via a cost-based optimizer.
- The Flink Common API and Optimizer layer takes programs in the form of operator DAGs. The operators are specific (*e.g.*, Map, Join, Filter, Reduce, FlatMap, MapPartition, ReduceGroup, Aggregate, Union, Cross, etc) and the data is in non-uniform type. The concrete types and their interaction with the runtime are specified by the higher layers.
- The Flink Runtime layer receives a program in the form of a JobGraph. A JobGraph is a generic parallel data flow with arbitrary tasks that consume and produce data streams. The runtime is designed to perform very well both in settings with abundant memory and where memory is scarce.

Flink explores the power of massively parallel computing for advanced analysis and leverages a novel, database-inspired approach to analyze, aggregate, and query very large collections of either textual or (semi-)structured data on a virtualised, massively parallel cluster architecture. It combines the strengths of MapReduce/Hadoop with powerful programming abstractions in Java and Scala and a

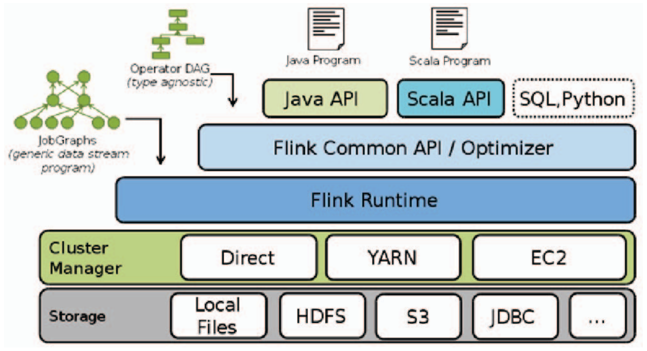

Figure 2. Apache Flink component stack.

high-performance runtime. In addition to basic dataflow concepts, common in relational databases or systems like Hadoop and Spark, Flink has native support for iterations, incremental iterations, and programs consisting of large DAGs of operations. It is possible to highlight the following features:

- Flink uses a richer set of primitives than MapReduce, including primitives that allow the easy specification, automatic optimization, and efficient execution of joins. This makes the system a more attractive compilation platform for data warehousing, information extraction, information integration, and many other applications. The programmer does not need to worry about writing parallel code or hand-picking a join order.
- Flink includes native support rather than outside loop in Mahout[2] on top of Hadoop for iterative programs that make repeated passes over a data set updating a model until they converge to a solution. Flink contains explicit "iterate" operators including bulk iteration and delta iteration that enable very efficient loops over data sets, *e.g.*, for machine learning and graph applications. These operators enable the specification, optimization, and execution of graph analytic and statistical applications inside the data processing engine.
- Different from Spark, Flink uses an execution engine that includes external memory query processing algorithms and natively supports arbitrarily long programs shaped as DAGs. Flink offers both pipeline (inter-operator) and data (intra-operator) parallelism. Flink's runtime is designed as a pipelined data processing engine rather than a batch processing engine, thus it supports both batch and streaming processing. Operators do not wait for their predecessors to finish in order to start processing data. This results in a very efficient handling of large data sets.

SMART approach take advantage of cloud, multi-cloud and hybrid infrastructures to provide support for SME service operation. The heterogeneous resources, in this scale, impose challenges to the data management and synchronizations, task distributions, result aggregations and failure tolerance mechanisms. The strategy to avoid the input data aggregation in a single data center for Big Data analysis promotes less data movement and reduces bandwidth needs. The new architecture improves SME competitiveness, because it allows them to choose the best resources with lowest prices.

IV. Evaluation Methodology and Results

This section describes the environment setup and results of a primary evaluation to demonstrate the scalability of the proposed framework. The experiments comprise empirical evaluation performed on the Grid'5000 environment and discrete-event simulation. Grid'5000 is an experimental testbed carried out under the Inria ALADDIN development plan with support from CNRS, RENATER and several universities in France. The simulation is performed with the BIGhybrid simulator, introduced in previous work [3], with a focus on cloud and hybrid systems[3].

The first experiment employs two homogeneous clusters from Grid'5000 and evaluates the profile execution and scalability of the proposal. The first cluster, located in Sophia, has 64 hosts, each host with 2 Intel Xeon E5520 processors of 2.27 GHz, with 4 cores, 24 GB of RAM, 119 GB of local disk and 1 Gbps network. The cluster performance is equivalent to 55.45 GFlops. The second cluster in Nancy has 128 hosts each with 1 processor Intel Xeon X3440 of 2.53 GHz, with 4 cores, 16 GB of RAM, 298 GB of local disk and 1 Gbps network. Each experiment was executed 30 times and results consider average times, and confidence interval of 95% with a t-student distribution.

The application is a MR execution, batch type. The input data contains a log with *n* lines, where each line has a host name related to an execution time. The Map function creates a *key/value* pair when the *value* is higher than 300 seconds. This *key* is the host name and the *value* is the execution time. The Reduce function receives all *key/value* pairs and calculates the average of execution time for each host, after that it creates a new *key/value* pair, where the *key* is the host name and *value* is the average execution of host. This execution is similar to wordcount, a popular micro-benchmark widely used by the MR community [21]. With this experiment the goal is to identify if the framework scales linearly as the workload grows.

Figure 3 shows an execution with 64 hosts, where red, blue and green colors represent *Map*, *Reduce* and *Shuffle* phases respectively. In the y-axis, the execution time is presented in seconds and the x-axis is the workload in GBs. The job presents linear execution time as the workload grows. The maximum standard deviation is 4.52 for a workload of 36 GBs. Map and Reduce functions are very fast with low workloads. Most time is spent with Map functions as

[2]http://mahout.apache.org/

[3]https://github.com/Julio-Anjos/Bighybrid

the workload increases. This behavior is realistic because it resembles wordcount where the map phase counts a word number of incidences and the reduce phase only sums up its incidences.

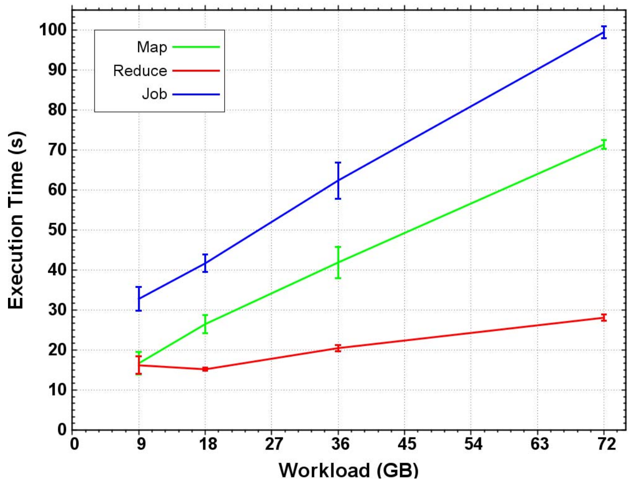

Figure 3. Profile execution time of 64 hosts.

An experiment with 128 hosts is shown in Figure 4, where red, blue and green colors represent *Map*, *Reduce* and *Shuffle* phases respectively. In the y-axis, the execution time is measured in seconds and the x-axis represents the workload in GBs. The job presents the same linear performance than in the experiment with 64 hosts, as the workload grows. This experiment is interesting because when the number of hosts is higher than what the workload requires, there is a tendency to have the same execution time of smaller settings (workloads of 9 GB and 18 GB). The standard deviations are 4.6 s and 2.8 s respectively. However, with workload of 72 GB the execution has a variability due to bandwidth competition that generates data contention with massive data transfers. For this case, the standard deviation is 9.5 s.

Figure 3 and 4 demonstrate that there is a similar behavior to different workloads and host numbers in a batch execution for data-intensive computing. The next experiment was executed in a cluster in Rennes with 25 hosts each host with 2 Intel Xeon E5-2630v3 processors of 2.4 GHz, with 8 cores, 128 GB of RAM, 600 GB of local disk and 1 Gbps network. This experiment has the goal to evaluate the performance between SMART vs. Spark, in streaming environment. The applications are wordcount and PI estimation calculation the following Monte Carlo methodology.

Each experiment was executed 90 times and the result is average time, the confidence interval is 95% with a t-student distribution. In Figure 5 the red and green colors represent SMART and Spark executions respectively. In the y-axis, the execution time is measured in seconds and the x-axis, the workload is measured in GBs. In Figure 6 the red and green colors represent SMART and Spark executions respectively.

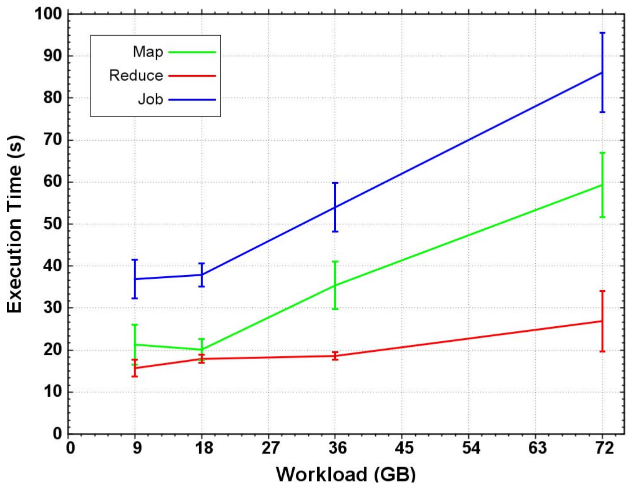

Figure 4. Profile execution time of 128 hosts.

In the y-axis, the execution time is measured in seconds and the x-axis, the interaction numbers for PI estimation is measured in units. The software versions of Spark and Flink used by Spark and SMART are 1.4 and 0.9 respectively.

The performance between SMART and Spark is better with small workloads while with large workloads the performance is similar as shown in Figure 5. The application is data-intensive, hence bandwidth contention is lower under small data transfers and grows as data transfers increase, thus minimizing the performance gain. Spark uses a resilient distributed data set approach that saves data intermediate first into memory to persist after in disk when finishing all operations.

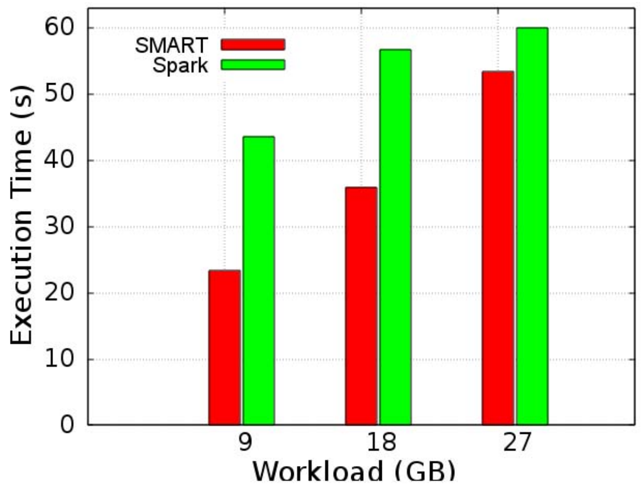

Figure 5. SMART vs. Spark streaming execution @ wordcount

Figure 6 presents the SMART implementation with a super-linear speedup. This speedup occurs because the implementation is CPU-intensive, and the SMART takes advantage with a more simplified programming model than Spark. This model is related with interact operators that enable very efficient loops over data sets. Therefore, a workload increase does not have important impacts on the system performance.

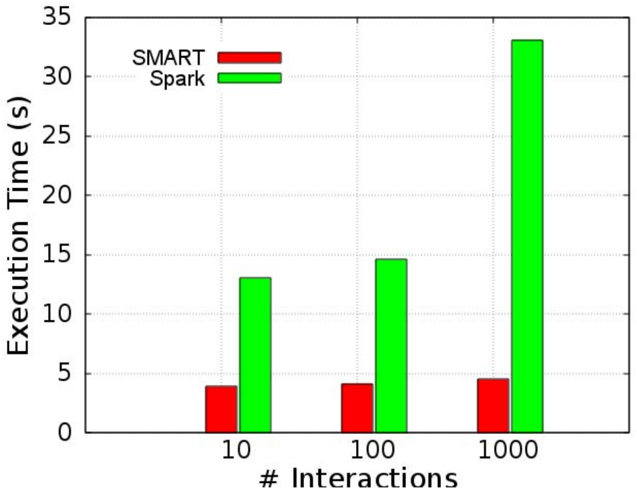

Figure 6. SMART vs. Spark streaming execution @ PI estimation

The next experiment reproduces the characterization of applications devised by Chen [22] from traces from two production environments of Yahoo and Facebook. The Yahoo traces were obtained from a 2,000 host cluster and contain 30,000 jobs spanning a period of over 3 weeks. The evaluation considers an application to data aggregation with fast jobs. The applications are data-intensive with MapReduce in batch model. This experiment is a large-scale simulation that enables to evaluate the proposal by simulating of algorithms and environment used by SMART, in a hybrid-cloud version of interest. The workload has 568 GB of input and 9,088 tasks, and each job has an execution time of 322.64 seconds from *Map* and 703.32 seconds from job. The number of mappers is 2,000 and of reducers is 1,000. This experiment has the goal to identify if the execution time of a theoretical SMART model is near to a real-world performance in large scale.

In Figure 7, red, blue and green colors represent *Map*, *Reduce* and *Shuffle* phases respectively. In the y-axis, the concurrent tasks are measured as units and the x-axis, the execution time is measured in seconds. Each host executes two tasks *Map* and *Reduce* concurrently. The experiment shows a *Job* time ≈ 680 s and a *Map* time ≈ 300. As the task numbers are limited to 2 tasks per host, the maximum concurrent *Map* tasks is 4,000 tasks and 2,000 concurrent *Reduce* tasks. These results indicate a good approximation from model.

V. Conclusions

In order to face the emerging challenges in cloud-based Big Data processing, this work presented a framework

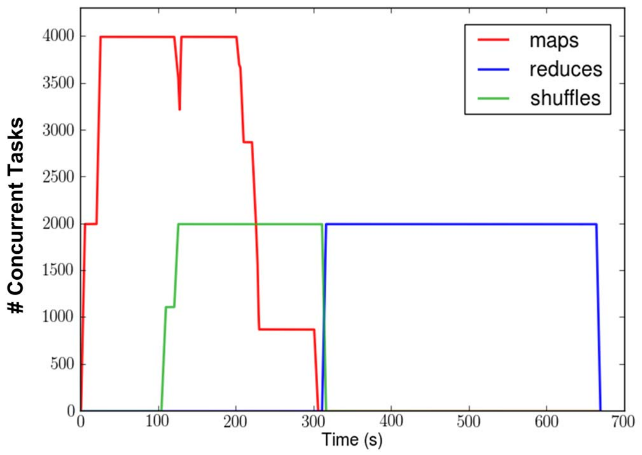

Figure 7. MapReduce execution of 2,000 host scenario from Yahoo traces.

consisting of composable data-analysis services that can be combined to address needs of specific applications. Focusing on applications for small and medium-sized organizations, the framework offers a flexible and lightweight approach that allows these organizations to take advantage of Big Data analysis in the cloud without incurring in the maintenance of heavy cloud infrastructures. Another important aspect to be highlighted is that of handling heterogeneous data sources, which makes the proposal applicable to a great number of companies and organizations running business in very different domains.

Preliminary results show good scalability of the SMART proposal, and the profile execution does not change with workload or host number. In streaming systems, the performance is workload sensitive which indicates a need for more detailed evaluation. The SMART implementation achieves better performance than Spark for CPU-intensive applications, and a workload increase does not have important impacts on the system performance. In large scale, the SMART simulation has a similar performance for large workloads in data-intensive applications.

Future work in handling data heterogeneity aiming at data standardization is a next step in the framework development. The exploration of diverse hybrid cloud infrastructures is another challenge to be addressed, as well as security and data privacy issues concerning the data analysis services performing operations on data from the heterogeneous data sources. Nevertheless, more evaluation will be needed considering data heterogeneity in future work.

Acknowledgments

This work was partly supported by CAPES - Foundation for Coordinating the Improvement of Higher Education Personnel - (Process BEX 14966/13-1). The experiments discussed in this paper were conducted with the aid of the Grid'5000 experimental testbed, under the Inria ALADDIN development plan with support from CNRS, RENATER and a number of universities (see https://www.grid5000.fr).

REFERENCES

[1] P. Mell and T. Grance, "The NIST Definition of Cloud Computing," Tech. Rep., Sep. 2011. [Online]. Available: http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf

[2] M. Stonebraker *et al.*, "Intel "Big Data" Science and Technology Center Vision and Execution Plan," *SIGMOD Rec.*, vol. 42, no. 1, pp. 44–49, May 2013. [Online]. Available: http://doi.acm.org/10.1145/2481528.2481537

[3] J. C. S. Anjos *et al.*, "BIGhybrid – A Toolkit for Simulating MapReduce in Hybrid Infrastructures," in *Computer Architecture and High Performance Computing Workshop (SBAC-PADW), 2014 International Symposium on*, Oct 2014, pp. 132–137.

[4] J. Dean and S. Ghemawat, "MapReduce - A Flexible Data Processing Tool," *Communications of the ACM*, vol. 53, no. 1, pp. 72–77, 2010.

[5] T. White, *Hadoop - The Definitive Guide*, 3rd ed. O'Reilly Media, Inc., 2012, vol. 1.

[6] M. Rychly *et al.*, "Scheduling Decisions in Stream Processing on Heterogeneous Clusters," in *Complex, Intelligent and Software Intensive Systems (CISIS), 2014 Eighth International Conference on*, July 2014, pp. 614–619.

[7] M. Zaharia *et al.*, "Discretized streams: an efficient and fault-tolerant model for stream processing on large clusters," in *Proceedings of the 4th USENIX conference on Hot Topics in Cloud Computing*, ser. HotCloud'12. Berkeley, CA, USA: USENIX Association, 2012, pp. 10–10. [Online]. Available: http://dl.acm.org/citation.cfm?id=2342763.2342773

[8] ——, "Resilient Distributed Datasets: A Fault-tolerant Abstraction for In-memory Cluster Computing," in *Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation*, ser. NSDI'12. Berkeley, CA, USA: USENIX Association, 2012, pp. 2–14. [Online]. Available: http://dl.acm.org/citation.cfm?id=2228298.2228301

[9] A. Toshniwal *et al.*, "Storm@Twitter," in *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '14. New York, NY, USA: ACM, 2014, pp. 147–156. [Online]. Available: http://doi.acm.org/10.1145/2588555.2595641

[10] D.-H. Le *et al.*, "SALSA: A Framework for Dynamic Configuration of Cloud Services," in *Cloud Computing Technology and Science (CloudCom), 2014 IEEE 6th International Conference on*, Dec 2014, pp. 146–153.

[11] L. Mashayekhy *et al.*, "A PTAS Mechanism for Provisioning and Allocation of Heterogeneous Cloud Resources," *Parallel and Distributed Systems, IEEE Transactions on*, vol. PP, no. 99, pp. 1–14, 2014.

[12] C. Jayalath *et al.*, "From the Cloud to the Atmosphere: Running MapReduce across Data Centers," *Computers, IEEE Transactions on*, vol. 63, no. 1, pp. 74–87, Jan 2014.

[13] Z. Zheng *et al.*, "STAR: Strategy-Proof Double Auctions for Multi-Cloud, Multi-Tenant Bandwidth Reservation," *Computers, IEEE Transactions on*, vol. PP, no. 99, pp. 1–14, 2014.

[14] Y. Mansouri *et al.*, "Brokering Algorithms for Optimizing the Availability and Cost of Cloud Storage Services," in *Cloud Computing Technology and Science (CloudCom), 2013 IEEE 5th International Conference on*, vol. 1, Dec 2013, pp. 581–589.

[15] A. Iosup *et al.*, "On the Performance Variability of Production Cloud Services," in *Cluster, Cloud and Grid Computing (CCGrid), 2011 11th IEEE/ACM International Symposium on*, May 2011, pp. 104–113.

[16] N. Grozev and R. Buyya, "Performance Modelling and Simulation of Three-Tier Applications in Cloud and Multi-Cloud Environments," *The Computer Journal*, vol. 58, no. 1, pp. 1–22, 2015. [Online]. Available: http://comjnl.oxfordjournals.org/content/58/1/1.abstract

[17] R. Tudoran *et al.*, "Bridging Data in the Clouds: An Environment-Aware System for Geographically Distributed Data Transfers," in *Cluster, Cloud and Grid Computing (CCGrid), 2014 14th IEEE/ACM International Symposium on*, May 2014, pp. 92–101.

[18] K. Krish *et al.*, "HATS: A Heterogeneity-Aware Tiered Storage for Hadoop," in *Cluster, Cloud and Grid Computing (CCGrid), 2014 14th IEEE/ACM International Symposium on*, May 2014, pp. 502–511.

[19] S. Delamare *et al.*, "SpeQuloS: a QoS service for BoT applications using best effort distributed computing infrastructures," in *Proceedings of the 21th international symposium on High-Performance Parallel and Distributed Computing*, ser. HPDC '12. New York, NY, USA: ACM, 2012, pp. 173–186. [Online]. Available: http://doi.acm.org/10.1145/2287076.2287106

[20] A. Alexandrov *et al.*, "The Stratosphere platform for big data analytics," *VLBD Journal*, vol. 23, no. 6, pp. 939–964, 2014. [Online]. Available: http://dx.doi.org/10.1007/s00778-014-0357-y;http://dblp.uni-trier.de/rec/bib/journals/vldb/AlexandrovBEFHHKLLMNPRSSHTW14

[21] S. Huang *et al.*, "The HiBench benchmark suite: Characterization of the MapReduce-based data analysis," in *Data Engineering Workshops (ICDEW), 2010 IEEE 26th International Conference on*, March 2010, pp. 41–51.

[22] Y. Chen *et al.*, "The Case for Evaluating MapReduce Performance Using Workload Suites," in *IEEE 19th Int. Symposium on Modeling, Analysis Simulation of Computer and Telecommunication Systems*, ser. (MASCOTS). IEEE Computer Society, July 2011, pp. 390–399.