A Novel Cloud Based Elastic Framework for Big Data Preprocessing

Omer Dawelbeit
School of Systems Engineering,
University of Reading,
United Kingdom.
Email: o.i.o.dawelbeit@pgr.reading.ac.uk

Rachel McCrindle
School of Systems Engineering,
University of Reading,
United Kingdom.
Email: r.j.mccrindle@reading.ac.uk

***Abstract*—A number of analytical big data services based on the cloud computing paradigm such as Amazon Redshift and Google Bigquery have recently emerged. These services are based on columnar databases rather than traditional Relational Database Management Systems (RDBMS) and are able to analyse massive datasets in mere seconds. This has led many organisations to retain and analyse their massive logs, sensory or marketing datasets, which were previously discarded due to the inability to either store or analyse them. Although these big data services have addressed the issue of big data analysis, the ability to efficiently de-normalise and prepare this data to a format that can be imported into these services remains a challenge. This paper describes and implements a novel, generic and scalable cloud based elastic framework for Big Data Preprocessing (BDP). Since the approach described by this paper is entirely based on cloud computing it is also possible to measure the overall cost incurred by these preprocessing activities.**

I. Introduction

Recent growth in structured, semi structure and unstructured Web, scientific and sensory data has resulted in the emerging concept of big data. Big data is data that is too big, too fast, or too hard [1] to process using traditional tools, and although the challenge of big data has been prevalent for the last few decades, the rate of growth experienced today indicates that sooner or later data will be generated at such an enormous rate that it will be hard to store, let alone analyse. The seriousness of this is evident by the big data prominence in the peak of the Gartner Technology Hype Cycle for 2013 [2]. The main driver behind the hype surrounding big data, is the potential value associated with this data, which enables organisations and businesses to gain valuable insight, trends or business intelligence. For example, the ability to analyse online advertising clickstreams or transactions history data can highlight crucial information about consumer behaviours, hence enabling businesses to tailor their offerings accordingly.

Before the emergence of the cloud computing paradigm in 2008, dealing with and processing big data required organisations to build their own infrastructure and purchase expensive software licenses for enterprise data warehouse databases. Cloud computing, through its offerings of Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS) now gives average sized organisations the ability to store and analyse, in seconds, terabytes of big data at a fraction of the cost of purchasing software licenses and with no upfront infrastructure investment. Cloud computing provides a pool of shared computing resources that can rapidly be acquired and released with minimal effort, either manually or programmatically [3]. Due to the dynamic nature of cloud computing, it offers computing resources elasticity [4], which allows computing resources to be acquired almost instantly when needed and released instantly when not needed.

Because cloud computing resources are virtualised and run on top of physical computers that are always on, it is possible to start and shutdown these virtual computing resources at any time. In addition to this elasticity, the other key benefit of cloud computing is usage-based costing so that consumers only pay for what they use, which is highly cost efficient compared to a physical computer environment being billed for 24/7, even when the resources are not being fully utilised. Cloud computing offers many other on demand services, such as mass low latency cloud storage capable of storing terabytes of data and big data services that can interactively analyse web-scale datasets in seconds [5].

Although cloud computing has given organisations the ability to store and analyse their big data, the ability to efficiently de-normalise and prepare this data to a format that can be imported into cloud based big data services remains a challenge. In this paper we describe and implement a novel, generic and scalable cloud based elastic framework for big data preprocessing. Since our approach is entirely based on cloud computing it is also possible to measure the overall cost incurred by big data preprocessing activities.

The rest of this paper is organised as follows: In Section II a brief review of some of the key cloud computing services is provided, then in Section III we summarise the functionalities provided by big data services and propose a lightweight preprocessing framework. In Section IV our novel framework is described and in Section V experimental evaluation is conducted and finally Section VI concludes this paper and highlights possible future work.

II. Cloud Computing

Cloud computing provides a range of pay as you go services that can be used by organisations to handle their big data. Some of these service include cloud storage for storage of big data, cloud big data services for big data analysis and cloud virtual machines for big data processing. In this section a brief review of these key services is provided.

A. Cloud Storage

To store terabytes and petabytes of data, organisations can either invest in their own storage solutions utilising physical disk based storage, which presents a number of challenges such as replication, backup and maintenance, or use a cloud based storage for storing and archiving their big data. Cloud storage offers a viable alternative to distributed disk based storage because there are no theoretical constraints on the size of the data that can be stored coupled with resumable fast file download/upload and web services based on the Representational State Transfer (REST) API to manage the stored data. The ideal use case is when this data is accessed by other services deployed in the same cloud service provider's network that contains the cloud storage, hence benefiting from fast network access. For example the data stored in the cloud storage can be directly imported to the cloud provider's big data services, such import will use the provider's fast network and can take only seconds to import gigabytes of data. Popular cloud storage services include Amazon Simple Storage Service (S3)[1] and Google Cloud Storage[2].

B. Cloud Big Data Services

Another cloud based feature that enables the interactive analysis of big data, is big data services, these services are based on columnar database systems [6] which, unlike traditional Relational Database Management Systems (RDBMS), store data in columns rather than rows. One of the key advantages of using a columnar data layout is the ability to run queries over tables with trillions of data in seconds [5]. The fast read capability is possible due to the fact that values belonging to the same column are stored contiguously, tightly packed and compressed [6], but this comes at the price of a slow or almost impossible update to existing columnar data. Popular cloud based big data services include Amazon Redshift[3] and Google Bigquery[4], both provide SQL like query languages and REST APIs for the execution of queries and the management of data. Google Bigquery provides an append only table structure for storing and analysing data, with the ability to store output data from queries into new tables. Typical preprocessing activities required for Google Bigquery include data formatting into either JavaScript Object Notation (JSON) or Comma Separated Values (CSV) formats, the data also needs to be de-normalised into a single table for fast access [7].

C. Cloud Virtual Machines

Cloud computing uses Virtual Machines (VMs) which, unlike physical computers, can be started and terminated programmatically through REST APIs when not needed in order to avoid incurring unnecessary processing cost. Cloud VMs can also be given a description or metadata which, they can query on an external server in order to make decisions about their processing such as the task to execute for example. It is also possible to take snapshots of the disk drives of running VMs, which can be used to start new VMs that contain the same software and disk state as the original VMs from which the snapshots were taken. This feature can be used to create many copies of the same VM for task processing.

Popular cloud based compute services such as Amazon Elastic Compute Cloud (EC2)[5] and Google Compute Engine[6], provide REST APIs for the lifecycle management of cloud VMs. Work on providing large scale elastic environments for scientific research shows promising results in regards to being able to acquire large number of cloud VMs in a very short time period, such as the ability to dynamically acquire over 400 cloud VMs within 15 minutes [8] .

III. Big Data

Big data is a term used to describe data that is too big, too fast, or too hard [1] to process using traditional tools. Gartner research provides the following definition for big data [9]:

> *Big data is high-volume, high-velocity and high-variety information assets that demand cost-effective, innovative forms of information processing for enhanced insight and decision making.*

The main driver behind the processing and analysis of big data is the potential value associated with it, such as insights and trends that can be used to enhance decision making and strategy formulation. The primary aspects of big data are widely characterised in terms of three dimensions (3Vs), these dimensions are summarised as follows:

1) Volume: big data is usually huge in size, which in today's terms is usually in the regions of Petabytes, Exabytes and Zettabytes.

2) Variety: due the diverse sources of big data, this data can be structured, semi-structured or unstructured.

3) Velocity: the rate of which big data is generated can range from batch to real time streaming.

A. A Lightweight Framework for Big Data Preprocessing

Recent research on big data workflows has attempted to provide a break down of the main functionalities that need to be handled by big data services. For example [10] lists these functionalities as:

- *Data collection*, which involves the gathering, filtering, and cleansing of data.
- *Data curation*, which involves the normalisation and selections of data models and structures.
- *Data integration and aggregation*, which involves the enforcement of data coherence.
- *Data storage* based on the adopted data model.

[1] http://aws.amazon.com/s3/

[2] https://cloud.google.com/products/cloud-storage/

[3] http://aws.amazon.com/redshift/

[4] https://cloud.google.com/products/bigquery/

[5] http://aws.amazon.com/ec2/

[6] https://cloud.google.com/products/compute-engine/

- *Data analysis and interpretation*, which involves data analysis and visualisation techniques.

The authors provide a classification of the big data services offered by vendors in respect to these functionalities. This classification shows that the majority of the MapReduce services based on the popular Hadoop Java framework[7] do support most of the aforementioned functionalities. In contrast, services such as Google Bigquery provides only data analysis and interpretation through Bigquery SQL. Despite this, Bigquery when compared to Hadoop, provides a convenient and easy to use mechanism that does not require the setup of any infrastructure to analyse large volumes of structured and semi-structured data. Bigquery through its streaming API also addresses one of the main challenges related to the velocity of real time data currently facing MapReduce [11].

Our approach provides a lightweight framework for the preprocessing of big data before importing it into analytical services such as Google Bigquery. Our preprocessing complements analytical services by implementing some of the data collection and curation functionalities for textual based, structured and semi-structured big data. Examples of such structured and semi-structured big data include server log files, online clickstreams and sensor readings.

B. Preliminaries

In our approach we divide tasks between a number of perfectly (embarrassingly) parallel processes that are executed by cloud VMs. By using perfectly parallel processing there is no overhead of communications between the various VMs processing the tasks. These processing VMs are only started when needed and terminated when not needed by a coordinator VM in order to minimise cost. Before we continue, we provide definitions of the key concepts of our big data preprocessing framework:

Definition Coordinators $\boldsymbol{T}$ are long lived VMs that are left on continuously more like physical computers, these VMs are required to stay alive for as long as the system is running. Coordinators accept as part of their arguments descriptions of other VMs which they need to construct and run.

Definition Processors $\boldsymbol{R}$ are short lived or ephemeral VMs that will be started by a coordinator to perform a particular task and then be terminated to avoid incurring unnecessary cost.

Definition Input $\boldsymbol{W}$ is the input provided to VMs through their metadata. This is a cloud computing provided feature such that VMs can be given metadata at startup or at any time when they are running, VMs can query this metadata proactively or get notified when it changes.

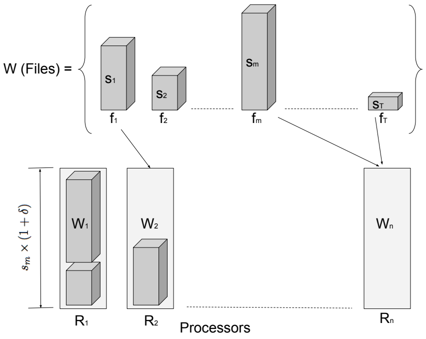


Fig. 1. Bin packing partitioning to divide files between the processors.

Definition Disk image $\boldsymbol{I}$ is a cloud disk image (snapshot) of the OS and the relevant software that should run on the VM. We take advantage of the ability to capture snapshots of running VMs to create a VM disk image $\boldsymbol{I}$ of a processor VM. This is achieved by first creating a VM containing a program that reads the VM metadata and computes with it, then a template is created from this VM and saved as a disk image.

Definition VM description $\langle \boldsymbol{VM} \rangle$ is the description of a VM such that $\langle \boldsymbol{VM} \rangle = \{\boldsymbol{S}, \boldsymbol{I}\}$ where $\boldsymbol{S}$ is the VM specifications such as machine type (for example CPU and memory), network and disk configurations, etc.. and $\boldsymbol{I}$ is a cloud disk image. The coordinator can use this description to create processor VMs that run the same program, but behave differently based on the metadata input $\boldsymbol{W}$ supplied to them.

C. Workload Distribution

The input data in our approach is largely based on textual based, structured and semi-structured files that are compressed to reduce the storage space and time required to transfer them to and from cloud based storage. When the coordinator receives an input W consisting of a number of compressed file, this input is divided further into $W_i \subseteq W$ such that the processing can be fairly spread between a number of processors. For this purpose we use the Bin Packing algorithm [12] to divide the input between a number of processors as shown in Fig. 1.

Assuming that the input W is a set of N files to process $(f_1, \cdots, f_i, \cdots, f_T)$ and the size of each file in bytes is $(s_1, \cdots, s_i, \cdots, s_T)$ with the largest file having a size of s_m such that $(0 < s_i \leq s_m)$ we divide these files into n processors where the capacity of each processor should not exceed $s_m \times (1 + \delta)$, we initially set δ as 10% of the size of the largest file. With smaller datasets where the largest file size s_m does not exceed a constant $\mathbf{K}$ bytes, we use the following processor capacity instead $\mathbf{K} \times (1 + \delta)$. Although most processors are likely to have approximately the same size of files to process there is a slight overhead when handling many smaller files compared to one large file, as the earlier will need to download and process multiple files.

IV. Big Data Preprocessing (BDP) Framework

In this section a formal definition and description of our elastic cloud based **B**ig **D**ata **P**reprocessing (BDP) framework is provided.

[7] http://hadoop.apache.org/

Definition BDP Framework is a 7-tuple, $(T, R, W, \Gamma, \Delta, \langle R \rangle, f)$, where T, R, W, Γ, Δ are all finite sets and:

1) T is the set of coordinators with one active coordinator and the rest as backup,
2) R is the set of processors,
3) W is the set of input items which in our case are the names and sizes of the files to preprocess,
4) Γ is the set of the contents of the actual files themselves,
5) Δ is the set of cloud services of mass cloud storage and analytical big data service,
6) $\langle R \rangle$ is the description of the processor VM R which includes a disk image I with an embedded program ρ and VM specifications S, and
7) $f : W \rightarrow R$, is the workload allocation function.

A. Framework Description

BDP is a cloud based generic framework that makes use of virtual machines and cloud services of mass cloud storage and analytical big data service to provide an elastic architecture for distributed work allocation and execution. Initially a program ρ that can read the metadata of the VM on which it is running is created, this program is then implemented on a template VM and saved as VM disk image I. As an alternative, instead of embedding into the disk image, the program ρ can be downloaded from a remote location to the VM when it starts using a startup script. Based on the metadata the program can:

- decide which task subroutine to execute, for our initial framework one subroutine is developed for preprocessing a set of files then loading their data into the big data services,
- read the work items W_i assigned to this processor such as file names such that $W_i \subseteq W$, and
- decide which big data table and cloud storage folder to consult and poll for the content of the input files Γ.

When the coordinator T receives a request for work to be performed it will create a set of input tasks W, then decides based on the complexity of the work and the request constraints (such as cost and time) on the number of processors to start. The coordinator then, using the disk image which has program ρ embedded, creates and starts the required number of processors R providing as metadata the set of work items W_i and the name of cloud storage folder or big data table where the input can be read. The coordinator also sets the name of the subroutine that the program should execute in the metadata and uses the workload allocation function f for dividing the work items between the processing nodes using a bin packing algorithm as shown in Algorithm 1.

Once a processor VM boots up the program ρ will be run by default, it will read the metadata then carry out the steps explained previously. When the processor finishes processing the assigned work items it updates its metadata to indicate that it is ready for further work as summarised in Algorithm 2. The coordinator can then decide to terminate the processing node or assigns it more work items.

Algorithm 1 Coordinator VM Preprocessing Data

```
function PREPROCESSDATA($W$, $\langle R \rangle$)    ▷ W is a set of input files
    $W^* \leftarrow$ BINPACKINGPARTITION($W$)
    $vmIds \leftarrow \{\}$
    for all $W_i \in W^*$ do
        $vmId \leftarrow startNewProcessor(\langle R \rangle, W_i)$
        $vmIds.add(vmId)$
    end for
    $O \leftarrow \{\}$    ▷ O is a set of output files
    for all $vmId \in vmIds$ do
        $O_i \leftarrow waitForProcessorToFinish(vmId)$
        $O \leftarrow O \cup O_i$
    end for
    return $O$
end function
```

Algorithm 2 Processor VM Preprocessing Data

```
function PREPROCESSDATA($W_i$)    ▷ $W_i$ is a set of input files
    $O_i \leftarrow \{\}$    ▷ O is a set of output files
    for all $w \in W_i$ do
        $O_w \leftarrow processFile(w)$
        $O_i \leftarrow O_i \cup O_w$
    end for
    updateMetadata()
    return $O_i$
end function
```

The coordinator also keeps track of the overall cost involved in fulfilling requests as cloud resources are billed based on usage such as the execution time or the size of the data. Fig. 2 summarises the initial model of BDP which is generic and can be implemented on any cloud platform that provides virtual machines, mass file storage and analytical big data services.

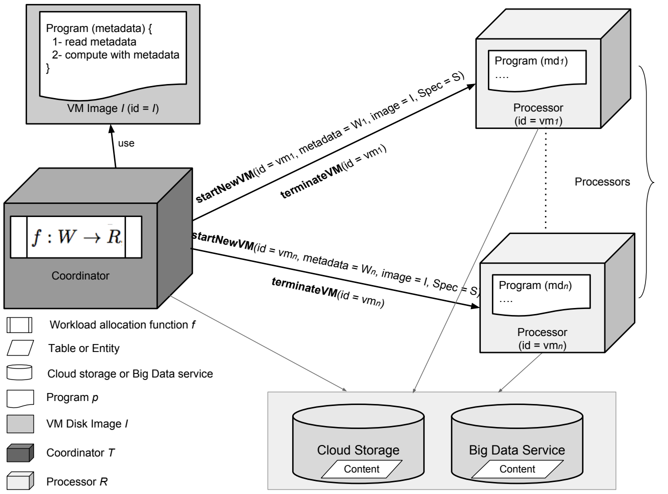


Fig. 2. Initial BDP Framework.

V. Experimental Evaluation

In this section experimental evaluation of the BDP framework is provided. To conduct the experiments we have developed a prototype based on the Java programming language and provided implementation for the Google Cloud Platform APIs[8]. The experiments were conducted on up to 10 *n1-standard-2*[9] VMs each with 2 virtual CPUs (equivalent to 2 hyperthreads on a 2.6 GHz Intel Xeon processor), 10 GB disk and 7.5 GB of main memory. The following major components were used:

- Google Compute Engine for creating and managing processor VMs,
- Google Cloud Storage for storing compressed big data files and as a central location for processor VMs to upload their processed data and

To evaluate our approach we have used the **DBpedia**[10] real world dataset, which is a collection structured data extracted from Wikipedia in Resource Description Framework (RDF) [13] format. We have used the English version of the 3.9 DBpedia dataset, which contains around 300 Million RDF statements in 64 Gzip-compressed files with a collective size of 5.3 GB compressed and 50.19 GB uncompressed. The two largest compressed files in the dataset have sizes of 916 MB and 769 MB respectively and the smallest file in the dataset has a size of 359 Bytes. In our experiments we implemented a task subroutine to cleans, normalise and convert the files from RDF to CSV format using our BDP framework and reported on the overall runtime and cost required to process the files versus the number of processor VMs. The overall runtime includes the time to download the files from Google Cloud Storage to the processor VM, the time to preprocess the data and the time to upload the files back to Google Cloud Storage ready for import into Bigquery.

TABLE I
Runtime and files sizes processed by 10 processors

Node Id	Num. of files	Gziped Size in MB	Actual size in GB	Runtime (min)	Processing time %
1	1	916	2.6	5.33	79%
2	1	769	9.5	9.33	90%
3	6	539	2.2	3.75	84%
4	5	539	8.2	6.75	91%
5	10	539	5.2	5.5	87%
6	8	539	5.0	5.33	86%
7	12	537	6.8	6.7	91%
8	10	537	6.5	6	90%
9	6	290	3.6	3.5	90%
10	5	97	0.9	1	84%

A. Results and Discussion

1) Number of processors: The number of processors were increased from 1 to 10, the runtime results are reported in Fig. 3a. As seen the total runtime decreases with increasing the number of processors. When the number of processors reaches 6 the runtime hardly decreases with increasing the number of processors, this is due to the fact the two large files in the dataset can not be divided between the processors and have to be assigned to one processor. Table I shows that the time required to process the largest uncompressed file remains at 9.33 minutes regardless of the number of processors used, which means, the overall runtime will be constraint by the time required to process the largest files. These files can be split further to enable equal workload allocation.

2) Cost of processing the data: Fig. 3b and 3c show the total cost in relation to the number of processors used and the overall runtime. The total cost is the per minute charge for each of the processors and the data transfer cost for transferring the data from and to the processors and cloud storage. As seen the cost gradually increases with increasing the number of processors, but increasing the cost and in turns the number of processors beyond 6 does not improve the runtime, this is due to the issue highlight previously with the largest files in the dataset.

3) Uncompressed data size: Table I shows that using the compressed file size for bin packing partitioning can be misleading, this is evident from the largest file of 9.5 GB which, when compressed becomes the second largest with a size of 769 MB. Since Fig. 3d shows that the runtime is proportional with the uncompressed file size, our bin packing partitioning can be improved by considering the uncompressed file size instead of the compressed size. Table I also shows that the average percentage of the overall runtime required to process the data is 87% with only 13% of the overall runtime for data transfer, this further strengthens the fact that data transfer within the cloud provider's network is not the bottleneck for big data processing.

Our initial experimentation with standard and Solid State Disks (SSDs) showed that the preprocessing of the Gzip-compressed files to be CPU intensive rather than disk intensive.

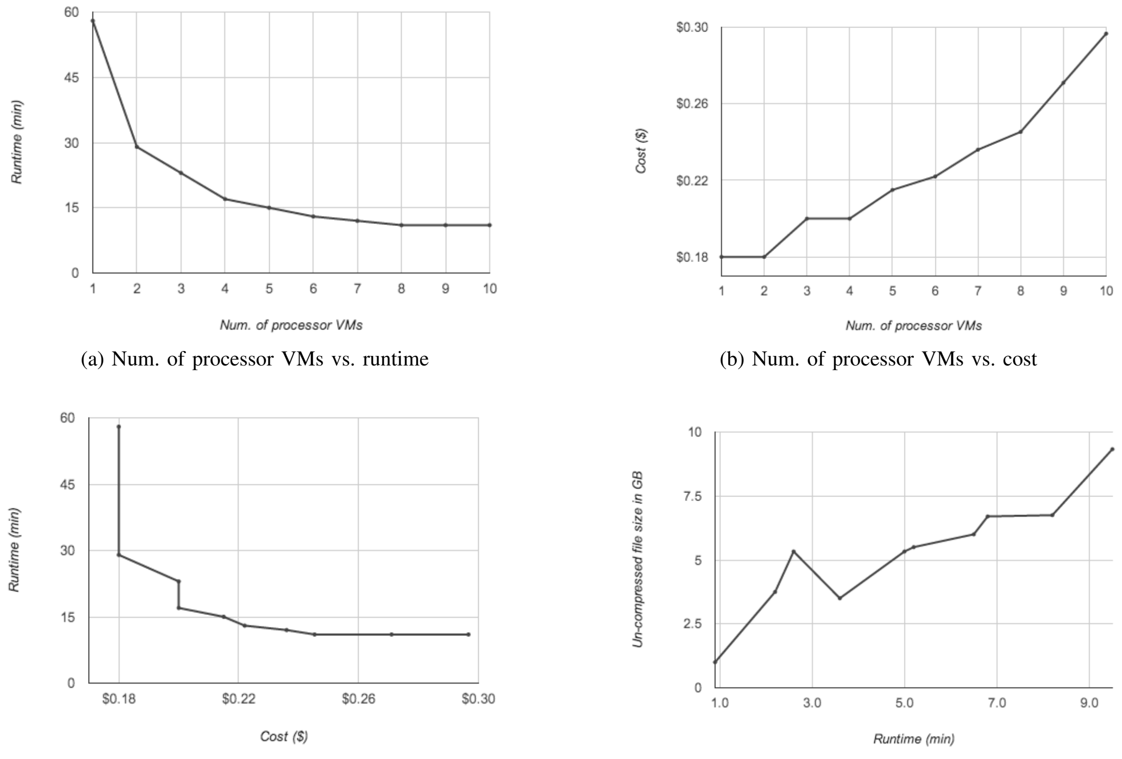
VI. Conclusion and Future Work

The continuous growth of big data has sparked a wide range of solutions for handling such growth and providing affordable means of analysing this data, which presents two challenges, firstly the ability to preprocess, format and clean this data and secondly the ability to efficiently analyse this data to extract value from it. The emergence of analytical cloud based big data services have addressed the second challenge, but the first challenge still remains. In this paper we have addressed the big data preprocessing challenge by presenting a novel and generic cloud based framework for big data preprocessing (BDP). We have utilised cloud elasticity and used a bin packing partitioning approach to distribute the workload equally between a number of processor VMs. We have evaluated our approach using a prototype and have shown that it is capable of loading and preprocessing 50 GB of real world large dataset in around 11 minutes using 8 processor VMs, we were also able to exclusively measure the cost incurred by the these preprocessing activities. Our experiments

[8] https://cloud.google.com/

[9] https://developers.google.com/compute/docs/machine-types

[10] http://wiki.dbpedia.org/Datasets



(a) Num. of processor VMs vs. runtime

(b) Num. of processor VMs vs. cost

(c) Cost vs. runtime

(d) Runtime vs. data size in GB uncompressed

Fig. 3. Experimental results

also allow us to extrapolate the runtime, number of required processors and cost to larger sizes of similar datasets.

As future work, we plan to implement the improvements we have suggested in subsection V-A. Another aspect that requires further work is dealing with and recovering from coordinator or processors failures. We also plan to extend our evaluation to a variety of larger datasets amounting to terabytes of data and add the ability to perform bespoke analytical queries once these datasets are loaded in the big data services.

Acknowledgment

We would like to thank Google Inc. for providing us with credits to run experiments on the Google Cloud Platform.

References

[1] S. Madden, "From Databases to Big Data," *IEEE Internet Computing*, vol. 16, no. 3, pp. 4–6, May 2012. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6188576

[2] G. Inc., "Gartner's 2013 hype cycle for emerging technologies maps out evolving relationship between humans and machines," August 2013. [Online]. Available: http://www.gartner.com/newsroom/id/2575515

[3] P. Mell and T. Grance, "The NIST Definition of Cloud Computing, Recommendations of the National Institute of Standards and Technolog," *National Institute of Standards and Technology*, 2011. [Online]. Available: http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf

[4] J. Varia, "Architecting for the cloud: Best practices," *Amazon Web Services*, no. May, pp. 1–21, 2010. [Online]. Available: https://jineshvaria.s3.amazonaws.com/public/cloudbestpractices-jvaria.pdf

[5] S. Melnik, A. Gubarev, and J. Long, "Dremel: Interactive analysis of web-scale datasets," *Proceedings of the VLDB Endowment*, vol. 3, no. 1-2, pp. 330–339, 2010.

[6] D. Abadi, P. Boncz, and S. Harizopoulos, "Column-oriented database systems," *Proceedings of the VLDB Endowment*, vol. 2, no. 2, pp. 1664–1665, 2009.

[7] A. Jacobs, "The Pathologies of Big Data," *Queue*, vol. 7, no. 6, p. 10, Jul. 2009. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1563821.1563874

[8] P. Marshall, H. Tufo, and K. Keahey, "Architecting a Large-scale Elastic Environment-Recontextualization and Adaptive Cloud Services for Scientific Computing." *ICSOFT*, 2012.

[9] M. Beyer and D. Laney, "The importance of 'big data': A definition," June 2012. [Online]. Available: https://www.gartner.com/doc/2057415/importance-big-data-definition

[10] B. Martino, R. Aversa, and G. Cretella, "Big data (lost) in the cloud," *Int. J. Big Data Intelligence*, vol. 1, 2014.

[11] K. Grolinger, M. Hayes, and W. Higashino, "Challenges for MapReduce in Big Data," *IEEE 10th 2014 World Congress on Services*, 2014.

[12] E. G. J. Coffman, M. Garey, and D. S. Johnson, "Approximation Algorithms for Bin Packing: A Survey," *Approximation Algorithms*, pp. 1–53, 1996.

[13] P. Hayes, "RDF Semantics," *W3C Recommendation*, vol. 10, pp. 1–45, 2004. [Online]. Available: http://www.w3.org/TR/2004/REC-rdf-mt-20040210/