Evaluating Cost of Cloud Execution in Data Repository

Zhiwu Xie, Yinlin Chen, Julie Speer, and Tyler Walters
University Libraries
Virginia Polytechnic Institute and State University
Blacksburg, VA, USA
{zhiwuxie, ylchen, jspeer, tyler.walters}@vt.edu

ABSTRACT
In this paper, we describe a set of controlled experiments to execute typical repository functions such as ingestion, fixity checking, and heavy data processing using cloud computing resources. We focus on the bridge pattern repository services where content is explicitly stored away from where it is processed. Using a large sensor dataset and AWS resources we measured the processing speed and unit cost of each scenario. The initial results reveal three distinct cost patterns: 1) spend more to buy proportionally or less than proportionally faster services; 2) more money can hardly buy better performance; and 3) spend less, but faster. Further investigations into these performance and cost patterns will help repositories to form a more effective operation strategy.

CCS Concepts
• **Information systems → Digital libraries and archives** • **Networks → Cloud computing** • *Applied computing → Digital libraries and archives*

Keywords
Institutional repository; Big data; Cloud computing; Cost analysis.

1. INTRODUCTION
With the volume steadily growing, digital libraries and archives are looking to leverage the shared cyberinfrastructure, especially the cloud, to support their operations. This is not only due to the various scalability constraints originated from the repository software and architecture [14], but also the difficulties to timely match the escalating needs with sizeable one-time investment to build up IT capacities.

The cloud has been used in digital libraries for storage, despite the argument that it is less economical for long-term use [11], as shown in the case of running LOCKSS style preservation network [12]. Many repositories use tools like DuraCloud [5] to backup and replicate extra copies of their content to off-site locations. Some also use the cloud storage as temporary overflow to preserve the primary copy exceeding the local capacities.

Conference'10, Month 1–2, 2010, City, State, Country.

DOI: http://dx.doi.org/10.1145/12345.67890

Storage, however, is only one of the many possible usages of the cloud. A use and reuse driven approach for data management [16] [17] requires much higher processing capacities a typical data repository provides today, making the on-demand, elastic cloud computing much more attractive than building local, single use facilities. Even for use cases where storage is the primary concern, the computing nodes drawn from the cloud are frequently used for moving content, ingestion, and fixity checking. For example, two of the five first nodes of the Digital Preservation Network (DPN) [6], the Chronopolis node [10] and the TDL node, both use DuraCloud not as a storage backend but a data movement mediation and frontend to staging for preservation [1]. Academic Preservation Trust (APTrust) [15], also a DPN first node, stores the content in the cloud as well as uses the cloud computing nodes to move, validate, and ingest content and frequently check the fixity afterwards.

The nature and characteristics of such cloud executions are not well understood. As a result, it is even more difficult to budget for the associated cost. The credit card backed, pay-as-you-go cloud payment approach is convenient, but can hardly form the basis of a sound business plan and strategy. The purpose of this paper is therefore to conduct controlled experiments and gather evidences, based on which a better understanding can be gained on the cost models associated with cloud executions of digital repository tasks.

A prior research indicated linear scalability [16] for some embarrassingly parallelizable jobs executed in the cloud, implying that the unit processing cost, measured in dollar per GB of data processed, may not increase significantly if more compute nodes are used to speedup processing. It is unclear, however, how far the linear scalability can sustain for more nodes, different use cases, and against different compute nodes. Will more expensive, supposedly faster compute node scales equally well and worth the extra cost? These are the questions we attempt to answer in this paper.

We limit the scope of this research to only the costs associated with the cloud execution. The storage and bandwidth costs are not considered in this research. We deliberately put the compute nodes and the storage in the same cloud availability zone so that maximum bandwidth can be sustained and data movements do not incur extra cost. The purpose is to isolate the cost factors and focus on base use cases that can be later compounded into more complicated scenarios.

2. THE BRIDGE PATTERN
We also limit the scope of the cloud execution to a so-called "bridge pattern", where data are stored in distinct locations away from where the processing is to be executed, with a network

bridge connecting the storage and compute. This section explains why this pattern is significant and how it maps to real world use cases.

In a single node digital repository, the data is usually stored at the local disk and processed on the same node where it is stored. When the size of the data grows, it is usually necessary to store the data at a separate storage facility connected with the repository node via fast network connections. The storage node will then be network mounted to the repository node as if the storage is local. Assuming the network is sufficiently fast, we may still pretend the network storage is local.

The local machine abstraction, however, is much less convincing if the data processing is also to be outsourced to on-demand cloud instances. Even if the network mounted storage is sufficiently fast as if it was local, e.g., the Elastic Block Storage (EBS) mounted to Elastic Compute Cloud (EC2) in Amazon Web Service (AWS), if the processing is to be executed at other nodes, the data must still be explicitly copied over to the execution nodes and the latency may not be negligible. Figure 1 schematically illustrates this pattern.

Figure 1. The Bridge Pattern

The bridge pattern is a more suitable abstraction for the DPN nodes described above, where the data are explicitly copied around over the network for processing. It also abstracts many other data management scenarios not necessarily related to cloud computing. For example, HUBZero [9] allows users to run simulations in remote Grid infrastructure using data and algorithms stored in the local repository. Since most existing data repositories lack sufficient processing power to directly reuse research data from within, the processing arising from the reuse will have to occur somewhere else, again adhering to the bridge pattern.

The bridge pattern does not cover all known library related data management use cases. Some systems [4][7], especially those based on Hadoop, do not explicitly distinguish storage and processing nodes. Instead all nodes are used for both storage and processing, and intelligently replicate, balance, and co-optimize across the interconnections between them. Separating the storage, bandwidth, and compute costs in these scenarios is more complicated, therefore is left for future work.

3. EXPERIMENT DESIGN

We designed three controlled experiments to execute typical repository tasks in AWS. To run these experiments we first installed a Fedora 4 based data repository using a m4.xlarge EC2 instance. This repository instance has a large EBS storage attached to it, such that all data deposited to the repository may be considered stored locally. The cost of this instance is not counted towards the execution costs. The data used for experiments are vibration signals collected from 214 accelerometers mounted all across the Goodwin Hall at Virginia Tech [3][13]. The data are written into zlib-compressed, chunked HDF5 files, one minute per file. We used 3 full days of data, total about 223GB. We stored the data at a temporary holding area in a Simple Storage Service (S3) bucket. We then allocate n EC2 instances, either of type t2.medium or larger, more expensive m4.large, to perform the processing. All the S3, EBS storages and EC2 nodes are provisioned from the AWS US East Region, such that data movements among them are fast and free of charge.

The first experiment mimics the typical repository ingestion process, where files are copied from S3 to n EC2 staging instances, $n = 1, 2, \ldots 9$. Each staging instance runs FITS [8] against the copied data, extracts technical metadata, then creates Fedora objects for each file and the associated metadata in the data repository.

The second experiment mimics the typical fixity checking process. Files ingested in the first experiment are read back in parallel into n EC2 processing instances, $n = 1, 2, \ldots 9$, along with associated SHA1 digests. Each processing instance then calculates the SHA1 digest on each file, and compares it with the original digest. The results indicating if the files have been changed are then written back to the data repository as provenance metadata associated with the original file.

The third experiment mimics the typical data reuse process. Files ingested in the first experiment are read back in parallel into n EC2 processing instances, $n = 1, 2, \ldots 9$. Each processing instance then performs some heavy calculations, in our case fast fourier transformations and matrix manipulations. We then simply discard the results, to simulate extremely light repository write back behavior.

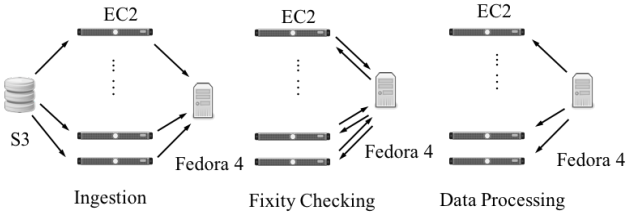

Figure 2. Data Flows in Experiments

Figure 2 illustrates the data flows of these experiments. We timed each experiments, then calculated the speed of execution as well as the unit cost in each experiment.

4. RESULTS AND ANALYSIS

4.1 Speedup

Figure 3 shows the speedup results of the three experiments. For the ingestion experiment, a linear speedup was consistently observed when using faster m4.large instances. This is easy to explain for vastly parallelizable workload like this. Because each task is fully independent from the others in terms of resources needed, doubling the resources cuts the time in half. Situations are markedly more baffling when using smaller, cheaper virtual instances. A superlinear speedup was on display when $n < 5$, then drifted to the linear or slightly sublinear region with larger n. Typically, superlinearity may be achieved when multiple resources can be interleaved. Refer to Figure 2, the ingestion process requires chaining three different types of resources: 1) the temporary storage at S3; 2) the processing node using EC2 instances; and 3) the repository node using EC2. Their interleaving is indeed a plausible cause, with the superlinearity

slowly disappearing due to the interleaving benefits been sufficiently exploited. However, it is not clear why slower processing nodes can in turn achieve faster ingestion rates, as clearly illustrated when $n > 6$. It also begs explanation why superlinearity was not observed at all when using m4.large instances. If, for some reasons, the ingestion process penalizes faster processing, then linear speedup would not have been possible either. The odd speed discrepancies observed here surely needs further investigation.

For the fixity checking experiments, close to linear speedup was observed at lower n for both faster and slower processing nodes, then hit a roof. This indicated a bottleneck was reached, possibly at the repository read/write speed, at around 400GB/hour. Faster machines seem to reach this bottleneck faster.

In the heavy data processing case, the bottleneck observed at the previous case is far from being reached, with highest processing speed less than 1/10 previously observed. This allows the expected linear speedup pattern to sustain all experiments, and faster machines yield a slightly faster speedup.

4.2 Cost

We calculate the unit cost by dividing the hourly rate of the aggregated processing instances by the processing speed. This is slightly different from the actual cost, since the Amazon charge rounds up the last partial hour into a full hour.

The speedup characteristics of the three different workload result in three drastically different cost patterns. The heavy data processing use case illustrates the expected pattern associated with linearity, where using more processing nodes can process data faster, but at the same or slightly higher unit cost. This pattern is further supplemented by the fixity checking use case, where the predicted cost pattern takes a sharp turn up when some kind of bottleneck is reached. Beyond this point, investing in more resources becomes wasteful. The rather surprising but cheerful cost pattern is illustrated by the ingestion example, where throwing in more resources to some extent can save money and get the work done faster at the same time. When the data volume grows higher, searching for this optimal combination will be of particular interest to repositories.

In all three sets of experiments, using cheaper, slower instances tends to be more cost effective than using the faster ones. In some cases, more expensive instances may be required in order to achieve higher processing speed. More money, however, cannot buy arbitrarily high speed.

5. SUMMARY

This paper discovers a few interesting performance and cost patterns encountered in leveraging cloud computing for repository operations. Although the use cases under investigation are representative, we caution too literal a reading into the results. The speed and unit cost numbers are indicative, but are also specific to our cases. On the other hand, the trends and patterns illustrated in these cases are more useful in cost and service planning.

Future work will be conducted to find the root cause of the superlinearity discrepancy and the scalability bottleneck.

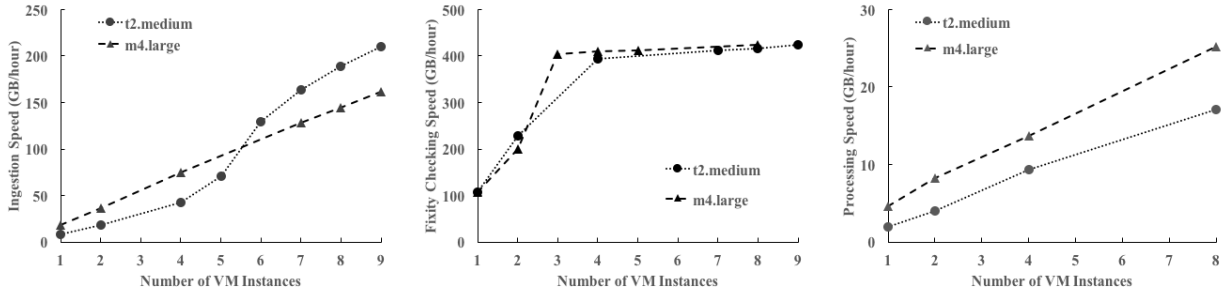

Figure 3. Speedup Using Multiple EC2 Instances.

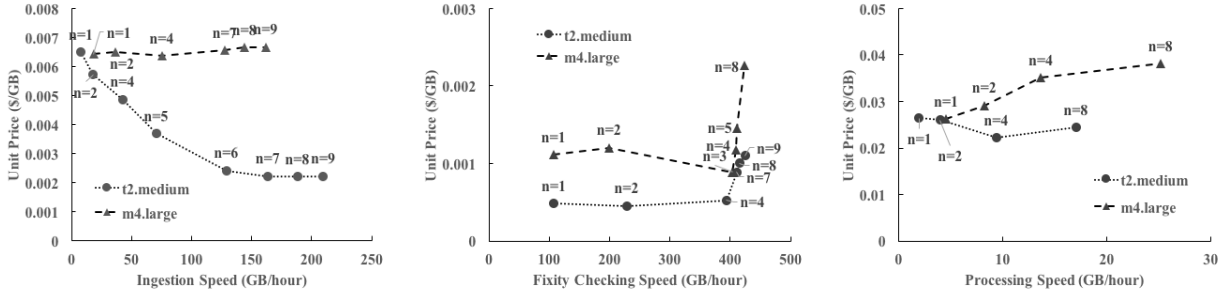

Figure 4. Unit Cost Using Multiple EC2 Instances.

6. ACKNOWLEDGMENTS

This research is partially supported by Amazon AWS Research Grants.

7. REFERENCES

[1] Branan, B. and Minor, D. 2015. Chronopolis and DuraCloud: Doing integration right. *2015 Preservation and Archiving Special Interest Group (PASIG) Conference* (San Diego, CA, USA, 2015).

[2] Branan, B., Schaefer, S. and Steans, R. 2015. Integrating DuraCloud with DPN at Chronopolis and the Texas Digital Library. *Open Repositories 2015* (Indianapolis, IN, USA, 2015)

[3] Hamilton, J.M., Joyce, B.S., Kasarda, M.E. and Tarazaga, P.A. 2014. Characterization of Human Motion Through Floor Vibration. *Dynamics of Civil Structures, Volume 4*. F.N. Catbas, ed. Springer International Publishing. 163–170.

[4] Jackson, A. 2014. Large-Scale Web Archive Discovery and Analytics Using Apache Solr. *2014 International Internet Preservation Consortium (IIPC) General Assembly* (Paris, France, 2014)

[5] Kimpton, M. and Morris, C. 2014. Managing and Archiving Research Data: Local Repository and Cloud-based Practices. *Research Data Management: Practical Strategies for Information Professionals*. J.M. Ray, ed. Purdue University Press. 223–238.

[6] Korner, B. 2013. DPN: An Overview from a Technical Perspective. 2013 *Preservation and Archiving Special Interest Group (PASIG) Conference* (Washington, DC, USA, 2013)

[7] Lin, J., Gholami, M. and Rao, J. 2014. Infrastructure for Supporting Exploration and Discovery in Web Archives. *Proceedings of the 23rd International Conference on World Wide Web* (Republic and Canton of Geneva, Switzerland, 2014), 851–856.

[8] McEwen, S. and Stern, R. 2009. FITS - The File Information Tool Set. *Open Repositories 2009* (Atlanta, GA, USA, 2009)

[9] McLennan, M. and Kennell, R. 2010. HUBzero: A Platform for Dissemination and Collaboration in Computational Science and Engineering. *Computing in Science & Engineering*. 12, 2 (Mar. 2010), 48–53.

[10] Minor, D., Schottaender, B.E. and Kozbial, A. 2014. Chronopolis Repository Services. *Ray, J eds. Research Data Management: Practical Strategies for Information Professionals*. Purdue University Press. 239–252.

[11] Rosenthal, D.S., Rosenthal, D.C., Miller, E.L., Adams, I.F., Storer, M.W. and Zadok, E. 2012. The economics of long-term digital storage. *Memory of the World in the Digital Age, Vancouver, BC*. (2012).

[12] Rosenthal, D.S. and Vargas, D.L. 2013. Distributed digital preservation in the cloud. *International Journal of Digital Curation*. 8, 1 (2013), 107–119.

[13] Schloemann, J., Malladi, V.V.N.S., Woolard, A.G., Hamilton, J.M., Buehrer, R.M. and Tarazaga, P.A. 2015. Vibration Event Localization in an Instrumented Building. *Experimental Techniques, Rotating Machinery, and Acoustics, Volume 8*. J.D. Clerck, ed. Springer International Publishing. 265–271.

[14] Shin, E. 2012. Built to Scale? *Open Repository 2012* (Edinburgh, UK, 2012).

[15] Turnbull, S. and Soroka, A. 2013. Academic Preservation Trust: Consortial Approach to Preservation and Services. *Open Repository 2013* (Prince Edward Island, Canada, 2013).

[16] Xie, Z., Chen, Y., Jiang, T., Speer, J., Walters, T., Tarazaga, P.A. and Kasarda, M. 2015. On-Demand Big Data Analysis in Digital Repositories: A Lightweight Approach. *Digital Libraries: Providing Quality Information*. R.B. Allen, J. Hunter, and M.L. Zeng, eds. Springer International Publishing. 274–277.

[17] Xie, Z., Chen, Y., Speer, J., Walters, T., Tarazaga, P.A. and Kasarda, M. 2015. Towards Use And Reuse Driven Big Data Management. *Proceedings of the 15th ACM/IEEE-CS Joint Conference on Digital Libraries* (New York, NY, USA, 2015), 65–74.