An Experiment on Bare-Metal BigData Provisioning

Ata Turk
Boston University

Ravi S. Gudimetla
Northeastern University

Emine Ugur Kaynar
Boston University

Jason Hennessey
Boston University

Sahil Tikale
Boston University

Peter Desnoyers
Northeastern University

Orran Krieger
Boston University

Abstract

Many BigData customers use on-demand platforms in the cloud, where they can get a dedicated virtual cluster in a couple of minutes and pay only for the time they use. Increasingly, there is a demand for bare-metal bigdata solutions for applications that cannot tolerate the unpredictability and performance degradation of virtualized systems. Existing bare-metal solutions can introduce delays of 10s of minutes to provision a cluster by installing operating systems and applications on the local disks of servers. This has motivated recent research developing sophisticated mechanisms to optimize this installation. These approaches assume that using network mounted boot disks incur unacceptable run-time overhead. Our analysis suggest that while this assumption is true for application data, it is incorrect for operating systems and applications, and network mounting the boot disk and applications result in negligible run-time impact while leading to faster provisioning time.

1 Introduction

Today, virtualized IaaS based BigData analytics solutions such as those provided by Amazon EMR [1] and IBM BigInsights [2] are boasting significant shares of the BigData analytics market [3]. Virtualization, at least in the way it is enabled in today's clouds, can introduce significant overhead, unpredictability, and security concerns, which is not tolarable for certain applications [4, 5, 6]. To address the needs of applications that are sensitive to these overheads, cloud vendors like IBM [7], Rackspace [8], and Internap [9] have started to serve bare-metal IaaS cloud solutions, with much of the focus being on supporting on-demand bare-metal BigData platforms.

All these Bare-metal cloud solutions install the tenant's operating system and application into the server's local disks, incurring long delays for the user of the platform. Projects such as Ironic [10], MaaS [11], Emulab [12] have developed sophisticated mechanisms to make this process efficient. A recent ASPLOS paper by Omote et al. [13] goes a step further to reduce these delays, lazily copying the image to local disk while running the application virtualized, and then de-virtualizing when copying is complete.

Is all this effort really necessary? In fact Omote et al [13] observe that network booting was actually faster than their approach, but asserted it would incur a "continual overhead", directing every disk I/O over the network. However, it is not clear if they considered the natural approach of having a network-mounted boot drive (with OS files and applications) and using the local drive for just application data.

To evaluate this option we create a simple prototype where client machines (24-core 10GbE servers, RHEL 7.1) access their kernel and init ramdisk via standard network boot mechanisms (PXE), mount their root file system with pre-installed applications (Hadoop benchmarks) from an iSCSI volume located on a remote server, and use local disk for ephemeral storage (i.e. /swap, /tmp, and Hadoop data). With this approach, which involved a few lines of config file changes, we found that the run time overhead of having a network mounted boot drive is in fact negligible. After a short startup phase there are very few subsequent reads from the boot disk (around 3KB/s over 10 hours) suggesting that file caching is very effective for the boot drive. Boot disk writes, mostly to application log files, average 14KB/s.

These results strongly suggest that the enormous effort by on-demand bare-metal platforms to reduce the delay and overhead of installing tenants operating systems on local disks may be misguided. The much simpler approach, of separating boot and data disks and handling them differently, appears to offer improved provisioning time with little or no runtime degradation. Moreover, a system based on this approach, can allow the boot drives to be stored in a centralized repository, bringing to bare

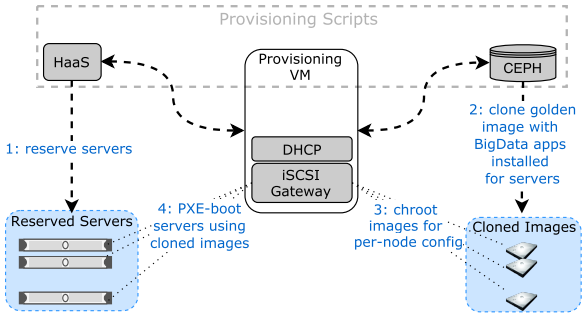

Figure 1: Architecture of our network mounted BigData provisioning environment and cluster provisioning flow.

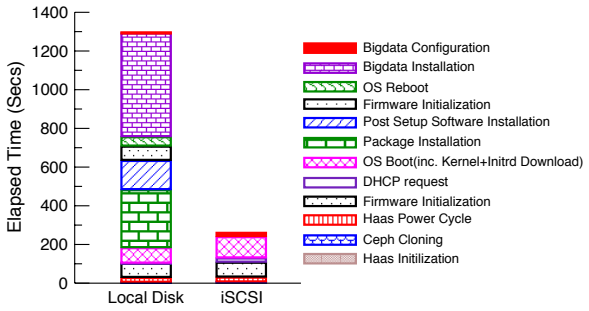

Figure 2: Provisioning time comparison of local disk installation and network (iSCSI) mounting.

metal environments many of the same capabilities available on virtualized platforms today. We are starting to develop a new Bare Metal Imaging (BMI) service based on this approach.

In the remainder of the paper, Section 2 describes the prototype we built to evaluating our approach to bare-metal BigData cluster provisioning. Section 3 presents the evaluation results we obtained. Related works are discussed in Section 4, and in Section 5 we conclude with a discussion of our findings.

2 The Prototype

Figure 1 shows the simple prototype we developed to evaluate our approach. HaaS [14] is a service we previously developed to allow users to allocate and provision physical nodes out of a shared pool. A single VM was used in the prototype for PXE services (DHCP, TFTP) and as an iSCSI server, with images (exposed to nodes as iSCSI targets) stored in a shared Ceph file system. Ceph provides us with a distributed storage system that supports efficient cloning of files. Most of the functionality in the prototype was implemented as bash scripts that interact with Ceph (to clone images), the provisioning VM, and the HaaS service.

We chose iSCSI, rather than NFS, as the protocol for mounting the drives because of the simplicity of installation—rather than crafting a shareable file system, we were able to connect a server to a blank iSCSI volume, perform a standard operating system installation, and then copy the resulting image file. Moreover, with the right hardware support, it should be possible to boot an iSCSI mounted drive with no changes to the operating system being booted. In addition, iSCSI does not incur the overhead of NFS to validate that potentially shared files have not been modified.

As shown in Figure 1, provisioning a cluster has four main steps:

1. Node Reservation: Provisioning scripts interact with HaaS to allocate physical servers.
2. Image Preparation: A golden image is cloned to create an image for each node.
3. Per-node Configuration: Each image is modified (using loopback mount) to perform per-node configuration such as SSH keys, cluster IP addresses (/etc/hosts), and specifying application-specific functionality. Each image is then exposed as an iSCSI volume by the iSCSI server.
4. PXE Boot: On boot the node requests configuration information via DHCP, downloading its kernel, initial ramdisk, and a configuration file giving the iSCSI address for that node's remote boot disk.

This prototype is designed to provide us with basic performance information, and has obvious limitations from both a functionality and performance perspective. A real implementation would have all the functionality implemented as bash scripts provided by an API-accessible service. The single provisioning VM will obviously be a performance bottleneck in the long term, as e.g. multiple iSCSI servers will be needed to scale to large numbers of nodes. Despite these issues, the current implementation provides a proof of concept, as a more carefully-constructed system would provide even better performance.

3 Evaluation

We tested the prototype on a HaaS-managed 48-node cluster; each server was equipped with two Intel Xeon E5-2630L CPUs, 128 GB memory, 300 GB 10K SAS HDDs (two nodes had 1 TB 7.2K SATA HDDs), and two Intel 82599ES 10 Gbit NICs. Storage was provided by a four-node Fujitsu CD10000 Ceph storage appliance, with 4 10 Gbit external NICs and internal 40 Gbit InfiniBand interconnect.

Figure 2 iSCSI bar shows the time taken to start up from scratch a bare-metal Hadoop image using our prototype. As we can see from the figure almost half the

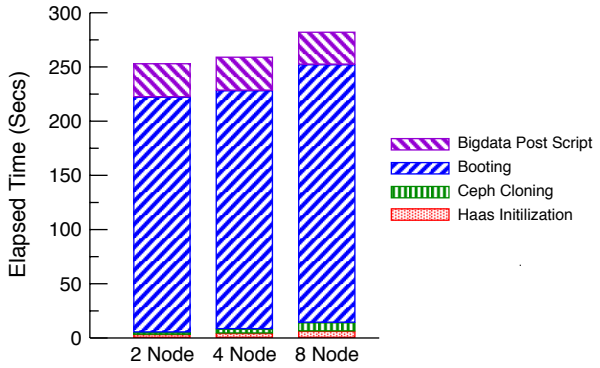

Figure 3: Scalability analysis for network (iSCSI) mounted BigData cluster provisioning.

time is spent in firmware initialization, and the overall boot time is very rapid (260 seconds) and comparable to network boot results presented in prior work [13]. As a comparison point, the Local Disk bar shows the time for a full install of a Hadoop environment using standard tools (RedHat Foreman for OS installation, Apache BigTop for Hadoop installation, ...), as is typically done in managed system environments. The remainder of this section compares the runtime overheads of these two installation mechanisms.

We have made no effort to make the prototype scalable. The provisioning scripts are sequential, cloning each image in turn, and then starting the nodes booting. Moreover, there is only a single provisioning VM in the prototype. Figure 3 demonstrates that even this very simple design is sufficient to provision a modest number of nodes in parallel with relatively modest degradation as we increase the number of concurrently provisioned nodes from two to eight.

The main goal of the prototype was to understand what is the run time impact of a network mounted boot drive for a Big Data platform. Figures 4 and 5 show the per node cumulative read and write iSCSI traffic during initial provisioning and then over five consecutive runs of random data generation followed by sorting, using the Hadoop Sort example, covering a duration of 7 to 17 hours for 128 GB and 256 GB of data respectively. These experiments were performed on two nodes allocated out of the HaaS cluster with local data stored in the one terabyte drive.

While we do not have a comparison to provisioning systems that copy (rather than install) an image to the local disk, one interesting data point is how much data would be transferred in the two cases. For the iSCSI case, Figures 4 shows that only around 250MBytes of the Boot disk are read over 10 hours. In contrast, out of the 8GB boot image image, 2.9GB were actually used. In other words, any image distribution service would need

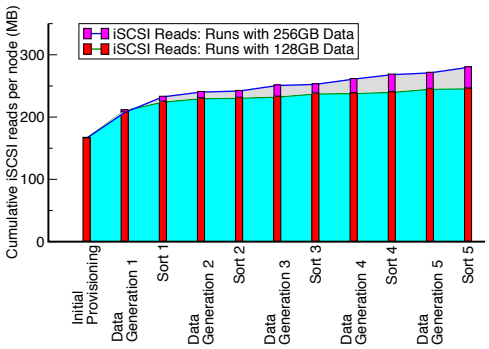

Figure 4: Per-node cumulative iSCSI read volume (MB).

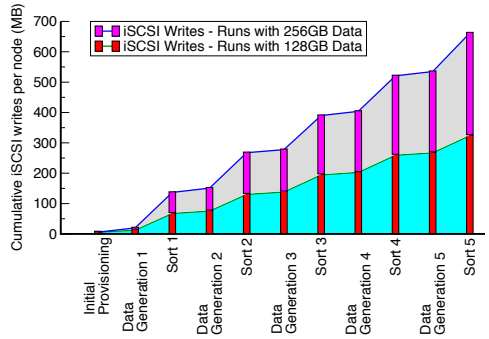

Figure 5: Cumulative writes (MB) made by servers on the iSCSI gateway.

to transfer 2.9GB over the network to each node. Worse yet, it would then need to write this data to the local disk, at typical speeds of 100 MB/s or less for single-disk systems, or $\frac{1}{10}$ of network speed.

In Figure 4, both curves flatten after repeated runs, demonstrating that (even with the 256GB case where total data handled—at minimum five runs times 128GB per machine, times a replication factor of two—is substantially larger than system memory) that the file cache is effective at caching the boot drive. After initial boot and application startup, the sustained read bandwidth incurred is around 3KBytes per second; effectively negligible.

Figure 5 shows the writes to the network mounted storage; in contrast to the read case, log writes continue throughout the experiment, at an average rate of approximately 14 KB/s. On further examination these writes target paths such as `/var/log`, `/hadoop/log`, and `/var/run`.[1] Most of these writes are log file updates made by Hadoop; although they could be directed to local storage, we did not do so due to their utility for debugging and negligible rate.

The above figures examined the read/write overhead for relatively large data sets for just two nodes, and took more than 17 hours to run. To examine the performance difference between the two configurations, we also timed

[1]We should note that, in our deployments, `/tmp` and `/swap` are configured to reside on the local disk of servers.

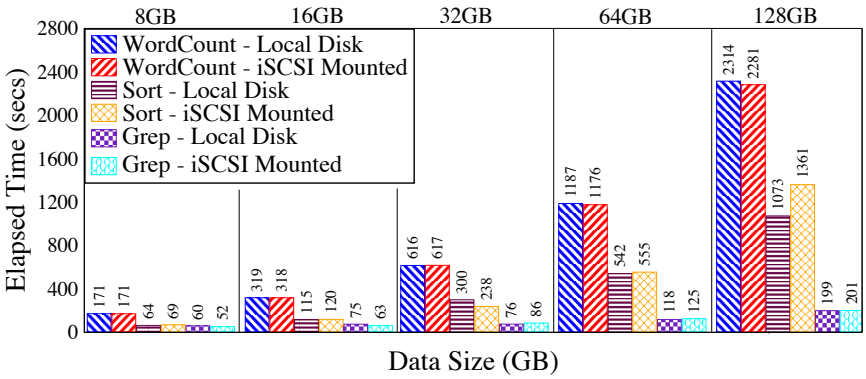

Figure 6: Performance comparison of Hadoop WordCount, Sort, and Grep applications on local disk based and iSCSI mounted systems.

a series of experiments on 8 node clusters as we varied the data set from 8GB to 128GB. In Figure 6 we compare the runtime of standard Hadoop benchmarks (Sort, Grep, WordCount) running on local disk-installed and network mounted clusters. Reported numbers are average of five runs; we observed that deviations among runs on the same configuration are negligible. As seen from the figure, the difference in runtime performances are negligible, with the exception of Sort experiments for 32GB data and 128GB data; we hypothesize that this may be caused by the non-deterministic behavior of random sorting benchmarks.

4 Related work

Network booting of computers came into widespread use almost 30 years ago [15], with remote access to both initial boot files (e.g. kernel) and file system enabling the creation of clusters of diskless workstations. More recently, remote storage has become popular in the high-performance computing field [16]. Some of the largest installations (e.g. those from Cray [17]) use this technique to allow smaller and more reliable diskless compute nodes, while others (Gordon [18]) add high-speed local storage (e.g. SSD) for ephemeral data.

In other fields, initial network booting (i.e. PXE) is widely used to initiate OS installation to local disk, but network boot with remote storage access is rarely used. Instead, a rich set of open source and commercial products have been developed for automated provisioning of bare-metal systems. Chandrasekar and Gibson [19] provide a comparative analysis of commonly used provisioning systems, namely Emulab [12], OpenStack Ironic [10], Crowbar [20], Razor [21], and Cobbler [22] and evaluate in detail Emulab and OpenStack Ironic. All of these bare-metal provisioning frameworks copy a disk image to the nodes and use additional configuration management systems to set up the desired applications on provisioned systems. Canonical's Metal-as-a-Service (MAAS) [11] provides a similar solution to host a cloud on the hardware owned by a customer.

Although there are many commercial offerings for BigData as a Service, such as Amazon Elastic MapReduce [1], Google Cloud DataProc [23], and others, most of these are based on virtual machine deployment. Internap [9] and Rackspace [8] offer Hadoop on bare-metal solutions using the OpenStack Ironic [10] provisioning solution, typically coupled with BigData application platforms such as Hortonworks Data Platform [24], Cloudera Enterprise [25], or MapR Converged Data Platform [26].

In addition to open source products and commercial solutions, there exists a flurry of academic studies that investigate problems related to the focus of this study. Ekanayake and Fox. [27] study the overhead incurred by Hadoop based applications run on bare metal vs virtual nodes in a Cloud infrastructure. Their studies corroborate the performance gains achieved by bare-metal deployment of BigData solutions. Omote et al. [13] investigate mechanisms for fast provisioning of operating systems and reducing boot time on bare-metal systems in clouds. They argue that long startup times of bare metal servers act as a significant inhibitor in providing agility and elasticity in bare-metal clouds, and propose BMCast, an OS deployment system with a special purpose de-virtualizable Virtual Machine Manager that supports OS-transparent quick startup of bare-metal instances. This approach, which we think is very novel and useful for many use cases, takes longer to provision then network mounting, and lacks the potential benefits of image management a network mounted approach can offer.

5 Conclusion

Bare-metal on-demand BigData platforms are becoming increasingly important with a number of commercial and research offerings. Enormous effort has gone on in these systems to reduce the delay to install software into the

local disks. While previous work acknowledged that network booting is faster than a local installation, they rejected this approach because of the assumption that it would incur an ongoing unacceptable overhead. We hypothesized that if we separate boot and data disks, using local storage for data, this overhead would be substantially reduced. We demonstrate with a simple prototype that this very simple strategy preserves all the advantages of network booting while incurring negligible runtime overhead.

Acknowledgments

We would like to thank Dan Shatzberg for his early suggestions in supporting a network mounted imaging service, and the MOC team in general for support and understanding while performing the experiments. We also thank Cisco and Fujitsu for their generous donations of server hardware and Ceph storage, respectively.

This research was supported in part by the MassTech Collaborative Research Matching Grant Program, NSF awards 1347525 and 1414119 and several commercial partners of the Massachusetts Open Cloud who may be found at `http://www.massopencloud.org`.

References

[1] Amazon, "Amazon elastic mapreduce (amazon emr)," https://aws.amazon.com/elasticmapreduce/, 2015.

[2] IBM, "Ibm biginsights for apache hadoop," www.ibm.com/software/products/en/ibm-biginsights-for-apache-hadoop, 2015.

[3] J. Kelly, "Hadoop-nosql software and services market forecast, 2014-2017," http://wikibon.com/hadoop-nosql-software-and-services-market-forecast-2013-2017, 2014.

[4] ZDNet, "Facebook: Virtualisation does not scale," http://www.zdnet.com/article/facebook-virtualisation-does-not-scale/, 2011.

[5] A. Iosup, S. Ostermann, N. Yigitbasi, R. Prodan, T. Fahringer, and D. Epema, "Performance analysis of cloud computing services for many-tasks scientific computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 22, no. 6, pp. 931–945, Jun. 2011.

[6] T. Ristenpart, E. Tromer, H. Shacham, and S. Savage, "Hey, you, get off of my cloud: Exploring information leakage in third-party compute clouds," in *Proceedings of the 16th ACM Conference on Computer and Communications Security*, ser. CCS '09. New York, NY, USA: ACM, 2009, pp. 199–212.

[7] Softlayer, "Big data solutions," http://www.softlayer.com/big-data, 2015.

[8] Rackspace, "Rackspace cloud big data onmetal," http://go.rackspace.com/baremetalbigdata/, 2015.

[9] Internap, "Bare-metal agileserver," http://www.internap.com/bare-metal/, 2015.

[10] Openstack, "Ironic," http://docs.openstack.org/developer/ironic/deploy/user-guide.html, 2015.

[11] Canonical, "Metal as a service (maas)," http://maas.ubuntu.com/docs/, 2015.

[12] D. Anderson, M. Hibler, L. Stoller, T. Stack, and J. Lepreau, "Automatic Online Validation of Network Configuration in the Emulab Network Testbed," in *IEEE International Conference on Autonomic Computing, 2006. ICAC '06*, Jun. 2006, pp. 134–142.

[13] Y. Omote, T. Shinagawa, and K. Kato, "Improving Agility and Elasticity in Bare-metal Clouds," in *Proceedings of the Twentieth International Conference on Architectural Support for Programming Languages and Operating Systems*, ser. ASPLOS '15. New York, NY, USA: ACM, 2015, pp. 145–159.

[14] J. Hennessey, C. Hill, I. Denhardt, V. Venugopal, G. Silvis, O. Krieger, and P. Desnoyers, "Hardware as a service - enabling dynamic, user-level bare metal provisioning of pools of data center resources." in *2014 IEEE High Performance Extreme Computing Conference*, Waltham, MA, USA, Sep. 2014. [Online]. Available: https://open.bu.edu/handle/2144/11221

[15] R. Gusella, "The Analysis of Diskless Workstation Traffic on an Ethernet," Tech. Rep., Nov. 1987.

[16] C. Engelmann, H. Ong, and S. Scott, "Evaluating the Shared Root File System Approach for Diskless High-Performance Computing Systems," in *Proceedings of the 10th LCI International Conference on High-Performance Clustered Computing (LCI-09)*, 2009.

[17] R. Alverson, D. Roweth, and L. Kaplan, "The Gemini System Interconnect," in *2010 IEEE 18th Annual Symposium on High Performance Interconnects (HOTI)*, Aug. 2010, pp. 83–87.

[18] S. M. Strande, P. Cicotti, R. S. Sinkovits, W. S. Young, R. Wagner, M. Tatineni, E. Hocks, A. Snavely, and M. Norman, "Gordon: design, performance, and experiences deploying and supporting a data intensive supercomputer," in *Proceedings of the 1st Conference of the Extreme Science and Engineering Discovery Environment: Bridging from the eXtreme to the campus and beyond*. Chicago, Illinois, USA: ACM, 2012, pp. 1–8.

[19] A. Chandrasekar and G. Gibson, "A comparative study of baremetal provisioning frameworks," Parallel Data Laboratory, Carnegie Mellon University, Tech. Rep. CMU-PDL-14-109, 2014.

[20] OpenCrowbar, "The crowbar project," https://opencrowbar.github.io, 2015.

[21] Puppetlabs, "Provisioning with razor," https://docs.puppetlabs.com/pe/latest/razor_intro.html, 2015.

[22] Cobbler, "Cobbler," https://cobbler.github.io, 2015.

[23] Google, "Google cloud dataproc," https://cloud.google.com/dataproc/overview, 2015.

[24] Hortonworks, "Hortonworks data platform," http://hortonworks.com/hdp/, 2016.

[25] Cloudera, "Cloudera enterprise," http://www.cloudera.com/products.html, 2016.

[26] MapR, "Mapr converged data platform," https://www.mapr.com/products/mapr-converged-data-platform, 2016.

[27] J. Ekanayake and G. Fox, "High performance parallel computing with clouds and cloud technologies," in *Cloud Computing*. Springer, 2010, pp. 20–38.