BOSTON UNIVERSITY

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

PhD Prospectus – Response to Conditions

BARE-METAL MARKETPLACE AT THE BOTTOM OF THE CLOUD

By

Sahil Tikale

B. Eng., L.D.College of Engineering, Gujarat University, India, 2003

M.S., Nanyang Technological University, Singapore, 2010

Advisor: Prof., Orran Krieger

Introduction

My research focus is building a system for sharing bare metal servers between mutually non-trusting physically deployed clusters in a data center. To this end, I have been associated with MOC since the beginning of my PhD program. During this course I have made key research contributions in the areas of isolation service, provisioning system, security model and incentive model.

1.1 The Isolation Service

At the very beginning of the PhD program I started working on a research project that involved building a new isolation layer for data-centers.

1.1.1 Hardware Isolation Layer (HIL)

HIL is a fundamental new layer, an isolation service, for data centers. It decouples the allocation and isolation of resources from the methods of deploying systems and applications software on physical servers. This is important since it enables hardware to be moved back and forth between different clusters each invested in its own methods and tools for provisioning servers. Previous systems that provide similar isolation to HIL only support their own integrated provisitiong system (eg., CloudLab, Ironic, Emulab etc) [12, 23, 28]. In contrast HIL enables a wide variety of provisioning systems to be used (MaaS, xCat, Foreman, etc) with minimal changes [23, 9, 29, 28, 6, 15].

1.1.2 Research Novelty

1. We demonstrated that it is possible to build a system that enables multiple provisioning systems to add and remove hardware from its clusters without changing the ways in which they deploy systems and application software.

2. We demonstrate that such a system incurs negligible overhead on the performance of the provisioning systems.

3. The isolation layer is a service that all tenants need to trust; we demonstrate with HIL that the implementation can be small; reducing the trusted computing base.

4. We developed a driver model that enables a variety of switches, out-of-band management modules(OBM), and authentication; enabling the code loaded for a particular deployment to be restricted to only the hardware required; reducing the trusted code required for a particular installation.

5. We demonstrated that an isolation system can have an interface with a small number of calls (with a restricted set of options) while supporting a variety of systems and use cases; suggesting a limited attack surface.

6. We demonstrated that such a tool enables a wide set of use cases such as operating systems (OS) researchers and system developers that require deterministic environments and can't use standard provisioning tools.

1.1.3 My contributions to HIL

I lead the efforts towards design and development of this project.

- I designed and implemented the driver model for both network and OBM drivers. It makes HIL extendable to support any new switches and OBM modules. It also helped in reducing the core code-base required by any installation.($\sim$ 3000 LoC). Contributes towards (3, 4 & 5) of §1.1.2

- I designed and implemented the OBM driver for IPMI tool, switch drivers to extend support to Juniper and Openvswitch. Contributes towards (4 & 5) of §1.1.2

- I developed the proof of concept demonstrating that provisioning systems like Ironic and MaaS can work with HIL. Integrated HIL with QUADS, an internal allocation system used at the scalability lab of Red Hat. With that work I demonstrated that HIL can support even a custom system used internally at RedHat. Prior to the integration with HIL, the QUADS system was tightly coupled to only Juniper switches. With the integration they can deploy the system in a data-center with different types of switches. Contributes towards (1, 2 & 3) of §1.1.2

- I designed API calls to provide users with a standard interface which is switch agnostic for network related actions and OBM agnostic for interacting with the servers. Contributes towards (5 & 6) of §1.1.2

- I designed and implemented the client library, that made it easy for HIL to be integrated with other systems. Contributes towards (5 & 6) of §1.1.2

1.2 The Provisioning System

Once we had the initial implementation of HIL ready, we could rapidly move bare-metal nodes between different clusters. However, the actual use of those nodes was very slow because the time to provision the OS and applications on those nodes took tens of minutes with the currently available provisioning tools. At that time I was working on making Ironic and Canonical MaaS work on top of HIL. While doing the testing, I would keep wondering what is the point of having the ability to move the servers faster, with fewer API calls, when it takes tens of minutes to get installed. While I was working in the industry, I used to manage SGI ICE-Clusters and had managed SAN solutions and I remembered how fast a 64 node cluster booted. This was because all the system images were booted from network mounted drives. But then the ICE cluster used infiniband networks and SAN solutions relied on dedicated optical fibre channels.

I continued to ponder if there was a way to reduce the time it takes to install a server. I read the work of Omote et.al.,[22] and was inspired by their approach. However, after brainstorming the idea with my advisor I realized that assumptions of Omote et.al., were flawed and their proposed solution was not simple to implement in practice. During the same discussion my advisor suggested that it is worth revisiting the assumptions made in the work of Omote et.al., by testing the cost associated with reading the data from a network mounted drive that uses distributed storage as a backend. I agreed, conducted initial tests and found promising results. Thereafter, I built the initial proof-of-concept that used HIL to allocate a group of bare-metal nodes and used pre-built images hosted from CEPH to setup a bare-metal cluster. This proof-of-concept led to the next project - building a Bare Metal Imaging system (BMI). BMI development was then led by another PhD student in our team, while I went back to refining HIL and developing another proof-of-concept at the scalability lab of Red Hat.

1.2.1 Bare Metal Imaging (BMI)

BMI is a system for provisioning bare metal servers using network mounted volumes out of a distributed storage system. It relies purely on commodity hardware and open source software. Previous systems were either specialized for HPC or workstation or relied on specialized hardware ([19, 26, 16, 25, 14, 11]). BMI enables services to be stood up rapidly on bare metal clusters and servers to be rapidly moved between different services.

1.2.2 Research Novelty

1. We demonstrate that it is possible to use stateless provisioning for general purpose set-up of bare-metal clusters using commodity systems and services. Previously, similar solutions were available only for specialized use-cases such as HPC systems [25, 14, 11], workstations [19, 26, 16] or required specialized hardware [5] or relied on non-commodity systems like storage area networks.

2. We demonstrate how features available in reliable distributed storage systems such as clones and snap-shotting can be used by a system to reduce the provisioning time of servers.

3. We provide a general mechanism that when integrated with HIL provides rapid re-provisioning capabilities. Releasing a server from a cluster is as easy as turning it off and within minutes it is possible to have the server servicing workload on another cluster.

4. We demonstrated that performance of such a system could be very good:

   (a) It is possible to provision and deploy bare-metal systems with overheads similar to deploying virtual machines

   (b) Clusters deployed using BMI can have performance similar to those provisioned on local disks.

5. We demonstrate that previous work [22] that incurred enormous complexity to enable fast provisioning using local storage, based on the hypothesis that network mounted drives could not deliver sufficient performance, was misguided.

1.2.3 My Contributions

- I designed and performed experiments to understand the effects of caching on the provisioning time of a remote mounted server. The initial tests included trying different combinations of turning caches on and off at RADOS block device or RBD (CEPH); at the iSCSI server (RBD client) and mounting disks read-only vs read-write, fileIO vs blockIO etc. Results from these tests became the basis of deciding the model that was adopted in building the BMI system. Contributes to (1 & 2 of §1.2.2)
- I developed the first proof of concept demonstrating that a cluster can be rapidly deployed by allocating nodes from HIL and booting them from remotely mounted drives. Contributes to (2, 3 & 4 of §1.2.2).

These led to the development of the BMI system which was led by other students and I collaborated with them.

1.3 The Security Model

HIL gave us a small trusted computing base (TCB) and BMI gave us a stateless way to provision nodes. During a discussion with the US Air Force, we realized we had to address the security aspect if we wanted our solutions (HIL and BMI) to be useful to a large class of clusters. Especially if they were to be used by security sensitive clusters hosted by federal agencies. I reviewed the literature to learn how commercial clouds were supporting the needs of security sensitive clients. I found that they would carve out a dedicated region which acted essentially as a private cloud. These were statically allocated clusters, customized to client needs and over provisioned. Hence, commercial clouds' security model was to create an inelastic protected silo within the cloud. However, we wanted a system that had desired levels of security without trading off elasticity.

This was my first opportunity at conducting security focused systems research. I soon realized that threat models are essential to security systems research. Security was not my forte and I knew very little about building threat models. However, I did not let this shortcoming limit my ability to address the security-elasticity trade-off problem and took the initiative to also learn threat modeling. After a few not-so-useful iteration of threat models, I happened to get the chance to talk to a program manager from IARPA. He had come to give a talk about the security requirements of clusters and data-centers hosted by the government. I asked him why the government is interested in an elastic solution for carving secure enclaves out of non-secure hardware in a public cloud. He shared that they were exploring public cloud alternatives to take advantage of on-demand elasticity in times of emergency and keep up-to-date with new tools and services the clouds have to offer. This would help them to consolidate their grossly underutilized data-centers. However, this meant having a model that allows them to scale up on demand during times of emergency without having to compromise with their security practices.

This was insightful, because I realized that such tenants have specific security requirements and also have necessary expertise to implement it themselves. All that they need from a cloud provider is the ability to get as much resources as possible in times of emergency. This discussion provided useful insights that helped me improve my threat model which led to the next project Bolted.

1.3.1 Bolted

Bolted is an architecture for allocating and providing physical nodes. It allows tenants of a cloud to control the trade-offs between security, price and performance of their clusters. It's designed to enable security sensitive tenants to minimize their trust in the provider and achieve similar levels of security and controls that they can obtain in their own private data centers. At the same time, Bolted neither imposes overhead on tenants that are security insensitive nor compromises the flexibility or operational efficiency of the provider. The prototype uses HIL, BMI, a remote attestation service [2], and (optionally) specialized firmware [18] to enable elasticity similar to virtualized clouds.

1.3.2 Research Novelty

1. We demonstrated that an architecture for bare metal provisioning can be developed where tenants can control the trade-offs between security, performance and price.

2. This is the first system that allows the security sensitive tenants to not need to fully trust the provider.

3. We demonstrated that, coupling HIL, BMI in the Bolted architecture enables provisioning time, even with encrypted disks and networks, that is substantially faster than stateful provisioning system [13]. We show that it is possible to measure all components needed to boot a server securely and yet have an elastic bare-metal cluster that can be set up faster than traditional provisioning systems [23, 13]

4. We demonstrated how tenants that don't want to trust the provider can make use of the key management system of the remote attestation service that supports automatic configuration of Linux Unified Key Setup (LUKS) [4] for disk encryption and IPsec for network encryption using keys bootstrapped during attestation. It also integrates with the Linux Integrity Measurement Architecture (IMA) [3] that allows tenants to continuously attest that a server was not compromised after boot.

5. We showed the cost, on our systems, of network and disk encryption, demonstrating that - while some workloads are insensitive (e.g. $\sim 18\%$ overhead), other applications incur over 200% overhead. This shows that it is very important that the tenants have the control to decide which workload is suitable to run depending on their trust on the provider.

6. We demonstrated that by delegating security of sensitive tenants to themselves, Bolted frees the provider from the complexity of having to directly support these demanding customers and avoids impact to customers that are less security sensitive. This is in contrast with architectures where providers tend to support one-size-fits-all solutions,

for operational efficiency. They apply uniform solutions (e.g. network encryption) to all customers; meeting the specialized requirements of highly security sensitive customers may impose unacceptable costs for others.

My Contributions

Jointly lead the effort towards and design and development of this project.

- I developed the threat model which guided the architecture of the system such as use of the *air-lock* – an isolated network into which to boot a node and test its integrity before adding it to the client's enclave. Contributes to (1 to 6 of §1.3.2)
- Together with my advisor and a postdoc developed the initial architecture and implementation that used HIL and BMI which led to the final implementation after subsequent iterations. Contributes to (1 to 6 of §1.3.2)
- I conducted the literature review for the work to establish the key contributions and wrote substantial parts of the publication. Contributes to (1 to 6 of §1.3.2)

The Incentive Model

As we were completing the bolted project, my advisor asked me what I would like to do as a final project for my thesis. I remembered at the beginning of the PhD program getting excited about the Open Cloud eXchange (OCX) model [7]. I proposed if I can work towards building an open market-place for sharing bare-metal servers. I had no knowledge about economics at the point but the idea of using economic principles to design a system that would allow exchange of resources at the bottom most layer of the cloud was exciting.

I taught myself basics of microeconomics, mechanism design, and the stock market. At first, I was trying to understand what economic model best describes what I was trying to do. Is it similar to stock market, commodities market, or will it look like an Uber and Airbnb model? I opened an online trading account to understand the design and operation of trading platforms. I also read widely about different types of auctions and delved deeper into VCG mechanism and 2nd price auctions.

I designed multiple versions of the architecture testing, revisiting my assumptions as my understanding of the requirements of the marketplace evolved. Initially I was not sure if having a marketplace was the right approach. With HIL, BMI & Bolted it was possible to efficiently move servers between clusters with different deployment and security practices. However, having a system that enables sharing of resources will not automatically lead to an increase in sharing of resources between clusters that are owned by different organizations. My advisor and I conversed with several of the administors of different HPC, cloud and analytic clusters at the MGHPCC [21], Chameleon cloud [20] and Cloudlab [12] to understand what barriers they had to adopt our approach and also what would incent them to use such a system. We found that they would not use the system if they lost control over their system. They were willing to share hardware only if they could fully control when and how to share their resources with others. This meant that we couldn't design a global scheduler that would control everything.

These conversations also led to the realization that the different clusters had very different objectives. For example, the Atlas HTC cluster [1] would be willing to give-up resources for periods of time if they could get more resources later; in other words they cared about the aggregate resources over a period of a month. MOC, on the other hand, ran most of the time at relatively low utilization, but would at time have very high utilization because of demand from researchers and production clusters. During my internship at the scalability lab of Red Hat, I found that they internally ran a system that allows teams to reserve bare-metal resources, and teams can build their own cloud and other clusters to conduct product and performance testing. They also experienced high demands near the product release cycles. Talking to people from Chameleon cloud and the CloudLab I learnt that they support many researchers that need bare-metal servers and would need access to same or similar hardware for the continuity of their experiments.

From these discussions we learnt that:

1. Organizations want to retain control over their own hardware i.e., a global mechanism that forces them to share would be unacceptable.

2. Different clusters have very different objectives that they are trying to maximize. For example utilization for HPC, revenue for Cloud and access to same or similar hardware for testbeds.

1.3.3 FLOCX

Based on this information, I propose FLOCX, an incentive based system for sharing bare-metal servers that follows a market-based economic model. The system will have decentralized control where a marketplace service will match resources to users while each provider will remain in control of its resources. The sharing mechanism will involve direct interaction between the provider and the user and will adhere to the conditions obtained using the marketplace for the duration of the agreement.

The system will:

1. Allow providers to have complete control on how they wish to share their resources.

2. Offer incentives to encourage clusters to share resources with others.

3. Allow administrators (or agents acting on their behalf) to optimize for their own objectives; therefore, there is no need to design a global system that understands the objectives of all possible clusters.

4. Match resources to requirements using fair and transparent mechanisms for resolving demand conflicts.

1.3.4 Related Work

FLOCX will be the first system exploring a marketplace model for exchanging physical servers between mutually non-trusting physically deployed clusters. Previous systems with the objective of sharing physical resources, did so by imposing containers (e.g., Borg, Mesos,

Apollo) [27, 17, 8]. None of them use marketplace mechanisms to resolve demand conflicts, instead they use priorities, quotas and centralized control.

Projects that provide physical servers (e.g., Cloud Lab, Chameleon, PlanetLab) impose a global scheduler (typically first come first serve) with leases for experiments [24, 20, 12]. Grid5000 is similar in having multiple providers, but again does not offer bare metal servers, provides best effort services and no marketplace, instead participating organizations trust the platform to offer and manage resources on their behalf [10].

1.3.5 Expected Contributions

I propose to build a prototype proof of concept that will demonstrate the feasibility of the FLOCX model and a trace driven simulation to enable scale and sensitivity analysis. Using traces of real systems I intend to :

1. Show how the presence of the marketplace affects the efficiency of sharing bare-metal nodes in a data-center.

2. Study the effects of provisioning time latency on the efficiency of the marketplace. I will compare the effects of using a stateful provisioning system eg. Foreman vs a stateless provisioning system eg. BMI.

3. Study the benefits a marketplace provides for achieving different cluster objectives (e.g., utilization, revenue, work per month) over a static partitioning of resources.

4. Optionally, I will compare the efficiency of the marketplace to a centralized scheduler that a fellow doctoral student is working on.

5. Compare the efficiency of a marketplace to an oracle.

Bibliography

[1] ATLAS.

[2] python-keylime: Bootstrapping and Maintaining Trust in the Cloud. `https://github.com/mit-ll/python-keylime`.

[3] Integrity measurement architecture (ima). `https://sourceforge.net/p/linux-ima/wiki/Home/`, May 2018.

[4] Linux unified key setup (luks). `https://gitlab.com/cryptsetup/cryptsetup`, May 2018.

[5] Appavoo, J., Uhlig, V., and Waterland, A. Project kittyhawk: Building a global-scale computer: Blue gene/p as a generic computing platform. *SIGOPS Oper. Syst. Rev. 42*, 1, 77–84.

[6] Berman, M., Chase, J. S., Landweber, L., et al. Geni: A federated testbed for innovative network experiments. *Computer Networks 61*, 0 (2014), 5 – 23. Special issue on Future Internet Testbeds.

[7] Bestavros, A., and Krieger, O. Toward an open cloud marketplace: Vision and first steps. *IEEE Internet Computing 18*, 1 (2014), 72–77.

[8] Boutin, E., Ekanayake, J., Lin, W., Shi, B., Zhou, J., Qian, Z., Wu, M., and Zhou, L. Apollo: Scalable and coordinated scheduling for cloud-scale computing. In *11th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 14)* (2014), pp. 285–300.

[9] Canonical. Metal as a Service. `https://maas.ubuntu.com/`.

[10] Cappello, F., Caron, E., Dayde, M., Desprez, F., Jegou, Y., Primet, P., Jeannot, E., Lanteri, S., Leduc, J., Melab, N., Mornet, G., Namyst, R., Quetier, B., and Richard, O. Grid'5000: a large scale and highly reconfigurable grid experimental testbed. In *The 6th IEEE/ACM International Workshop on Grid Computing, 2005.* (Nov 2005), pp. 8 pp.–.

[11] Daly, D., Choi, J. H., Moreira, J. E., and Waterland, A. Base operating system provisioning and bringup for a commercial supercomputer. In *Parallel and Distributed Processing Symposium, (IPDPS)* (2007), IEEE, pp. 1–7.

[12] Duplyakin, D., Ricci, R., Maricq, A., Wong, G., Duerig, J., Eide, E., Stoller, L., Hibler, M., Johnson, D., Webb, K., Akella, A., Wang, K., Ricart, G., Landweber, L., Elliott, C., Zink, M., Cecchet, E., Kar, S.,

and Mishra, P. The design and operation of cloudlab. In *2019 USENIX Annual Technical Conference (USENIX ATC 19)* (Renton, WA, July 2019), USENIX Association, pp. 1–14.

[13] Foreman. Foreman. `https://www.theforeman.org/`, 2019.

[14] Guler, B., Hussain, M., Leng, T., and Mashayekhi, V. The advantages of diskless hpc clusters using nas. *Technical Report Dell Power Solutions* (2002).

[15] Hat, R. Introduction to Foreman OpenStack Manager. `https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux_OpenStack_Platform/4/html/Installation_and_Configuration_Guide/chap-Foreman_Overview_and_Installation.html`. available from `access.redhat.com/documentation`, Oct 2015.

[16] Haun, C., Prouse, C., Sokol, J., and Resch, P. Providing a reliable operating system for clients of a net-booted environment, June 15 2004. US Patent 6,751,658.

[17] Hindman, B., Konwinski, A., Zaharia, M., Ghodsi, A., Joseph, A. D., Katz, R. H., Shenker, S., and Stoica, I. Mesos: A platform for fine-grained resource sharing in the data center.

[18] Hudson, T. Linuxboot. `https://github.com/osresearch/linuxboot`.

[19] Klimenko, Y. Technique for reliable network booting of an operating system to a client computer, Oct. 26 1999. US Patent 5,974,547.

[20] Mambretti, J., Chen, J., and Yeh, F. Next generation clouds, the chameleon cloud testbed, and software defined networking (sdn). In *2015 International Conference on Cloud Computing Research and Innovation (ICCCRI)* (Oct 2015), pp. 73–79.

[21] Massachusetts Green High-Performance Computing Center.

[22] Omote, Y., Shinagawa, T., and Kato, K. Improving agility and elasticity in bare-metal clouds. In *Proceedings of the Twentieth International Conference on Architectural Support for Programming Languages and Operating Systemsi, ASPLOS* (Turkey, 2015), ACM, pp. 145–159.

[23] Openstack. Ironic. `https://docs.openstack.org/ironic/latest/`, 2018.

[24] Peterson, L., and Roscoe, T. The design principles of planetlab. *SIGOPS Oper. Syst. Rev. 40*, 1 (Jan. 2006), 1116.

[25] Salah, K., Al-Shaikh, R., and Sindi, M. Towards green computing using diskless high performance clusters. In *Network and Service Management (CNSM), 2011 7th International Conference on* (2011), IEEE, pp. 1–4.

[26] Sposato, D. Method and apparatus for remotely booting a client computer from a network by emulating remote boot chips, Oct. 8 2002. US Patent 6,463,530.

[27] Verma, A., Pedrosa, L., Korupolu, M., Oppenheimer, D., Tune, E., and Wilkes, J. Large-scale cluster management at google with borg. In *Proceedings of the Tenth European Conference on Computer Systems* (2015), pp. 1–17.

[28] White, B., Lepreau, J., Stoller, L., Ricci, R., et al. An Integrated Experimental Environment for Distributed Systems and Networks. *SIGOPS Oper. Syst. Rev. 36*, SI (Dec. 2002), 255–270.

[29] xCAT. Extreme Cloud Administration Toolkit, 2019. `https://xcat-docs.readthedocs.io` Last accessed: 2019-11-15.