BOSTON UNIVERSITY

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

PhD Prospectus

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

Bare-metal Marketplace at the Bottom of the Cloud

Abstract

Today's clouds offer huge benefits in terms of on-demand elasticity, economies of scale and its pay-as-you-go model. Yet many organizations continue to host their clusters outside of cloud for security, price or performance reasons. Such organizations form a large section of the economy including financial companies, medical institutions and government agencies. Clusters are typically stood up with sufficient capacity to deal with peak demand; resulting in silos of under-utilized hardware. This situation is common place not only within an individually owned data-center running multiple clusters but also across colocation facilities like the MGHPCC[2] – a 15 megawatt data-center that hosts infrastructure from five different universities.

In the thesis we ask the question, What if we could easily and securely move infrastructure across these silos to match demand? This would obviously increase utilization and reduce the cost. Moreover, as we will see, such a capability could enable an alternative model of cloud; an Open Cloud eXchange (OCX) where multiple organizations, freely cooperate and compete with each other for offering different hardware resources while customers can choose from numerous competing services instead of a single provider.

The objective of this thesis is to build a system that allows mutually non-trusting physically deployed services to efficiently share the physical servers of a data center. The approach proposed here is to build a system composed of a set of services each fulfilling a specific functionality. The scope of this work is limited to design and implementation of only the core set of functionalities critical for sharing bare-metal servers between clusters across different deployment systems and ownership.

These functionalities are: 1. *Bare-metal Allocation and Isolation Service:* to allow different users to stand up clusters from a common pool of hardware. 2. *Diskless Rapid Provisioning Service* that allows deploying the cluster fast enough to be able to respond to rapid fluctuations in demand. 3. *Security Framework* that enables cluster owners to control trade-offs between security, price, and performance. 4. *Market based incentive system* that uses an economic model of a marketplace to ensure that resources are allocated to the cluster that needs it most. we have completed (1-3) and describe our architecture and results. Its results will be discussed in brief. Thereafter open questions regarding the (4) will be discussed followed by proposed timeline for the work pending towards completion of this thesis.

Contents

1 Introduction

1.1 Background

Today's clouds offer huge benefits in terms of on-demand elasticity, economies of scale and its pay-as-you-go model. Yet, a large section of the economy including financial companies, medical institutions and government agencies continue to host their own clusters outside of the public clouds. Organizations that need total control of their hardware; have custom deployment practices; require specific security requirements and do not wish to pay for high prices of storage [5, 10] invest in their own hardware than renting it from public cloud providers.

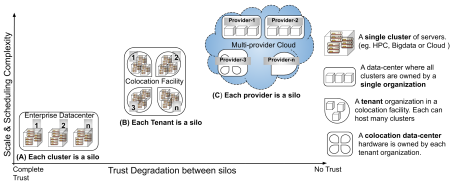

Figure 1·1: Type(A) Trust is highest as silos belong to a single owner. Complexity of scheduling is low. **Type(B)** Trust lower as each silo belongs to a different organization where each is a tenant of a colocation datacenter. Scheduling complexity is higher as it includes constraints of moving hardware within tenants' owned clusters (type-A silos) and inter-tenant clusters. **Type(C)** Silo can be a whole data-center (provider-1,2) or a subset of it. The trust is least between providers that have competing objectives. The scale of operations and scheduling complexity is highest as these includes type(A) & type(B) silos also.

Many different mechanisms are available that simplify deployment of applications and services on physical systems such as OpenStack Ironic, Canonical MaaS, Emulab, GENI, Foreman, xCat, and others [13, 6, 15, 3, 9]. Each of these tools takes control of the hardware it manages, and each provides very different higher-level abstractions. A cluster operator must thus decide between, for example, Ironic or MaaS for software deployment; and the data center operator who wants to use multiple tools is forced to statically partition the data center into silos of hardware. Moreover, it is unlikely that most data centers will be able to transition fully to using a new tool; organizations may have decades of investment in legacy tools, e.g., for deploying HPC clusters, and will be slow to adopt new tools, leading to even more hardware silos.

Clusters are typically stood up with sufficient capacity to deal with peak demand; resulting in silos of under-utilized hardware. And when demand exceeds capacity, a cluster may suffer from degraded quality of service (QoS) or violation of service level agreements (SLAs) while enough capacity of physical servers may be available in a rack right next to it. Those servers cannot be used even if they are sitting idle simply because there is no way to move them between silos of clusters. The problem is sufficiently general whether it is a large enterprise hosting multiple clusters in its own data-center (type-A silos, Fig:1·1) or colocation facilities such as the MGHPCC[2] – a 15 megawatt data-center that hosts infrastructure from five different universities (type-B silos,Fig:1·1).

In this thesis we ask the following questions • Can we design a system such that it is possible to move hardware from one cluster to another on demand? • How can we make setting up a cluster fast enough to be able to respond to rapid fluctuations in demand? • Can we design a single system which is appropriate for a wide variety of scenarios, from multiple clusters of a single company to different tenants in the same colocation facility, to new model of cloud with multiple providers[4]? • Can we design a marketplace model, which we know is needed for a cloud that is sufficiently general to support a broad set of use cases for sharing resources?

To address this questions, the thesis proposes a general architecture that we call the

Elastic Secure Infrastructure or (ESI). It is a system composed of a set of micro-services that allows mutually non-trusting physically deployed services to efficiently share the physical servers of a data center. ESI requires that only a minimal functionality be trusted by everyone while rest can be deployed by each participating tenant themselves.

1.2 Broader Impact: An alternative to single provider public clouds

The bottom-most layer of any single provider public cloud is either a virtualization based black-box with no visibility into the allocation of resources or a bare-metal solution with limited deployment options and no choice but to trust the cloud provider completely for its security. Only a handful of organizations have the capital for setting up a public cloud. This seriously limits avenues for innovation because the research and academic community does not have access to the internal workings of an at-scale cloud. We believe that the public cloud can evolve into a marketplace in which many actors can compete and cooperate with each other, and innovation can flourish. We refer to this model as the Open Cloud eXchange model or (OCX) [4]. In addition to breaking silos ESI is a critical requirement to enable OCX as the bottom most stack of a multi-provider cloud – a single platform where stakeholders from both industry and academia together host their hardware and services to each other and to customers on pay-as-you-go basis.

The scope of this thesis is the design and implementation of the following functionalities. • *Bare-metal Allocation and Isolation Service* that uses network isolation technologies to isolate tenants' bare-metal servers • *Diskless provisioning service* that installs software on servers using network mounted storage. • *Security framework* that allows different cluster owners to attest the integrity of the physical server before choosing to add it to its cluster. • *Decentralized distribution system* that use the economic model of a marketplace to encourage different clusters owners to share their hardware with others.

We have prototype implementation of the *Bare-metal Allocation and Isolation Service*, *Diskless provisioning service* and *The Security framework*. We will discuss each one briefly in the next chapter along with results that demonstrate that it is possible to not only move hardware between silos of clusters with significantly different security requirements but it also possible to deploy it at speeds comparable to hosting a virtualized infrastructure.

1.3 Motivation: Characteristics of different clusters in a Datacenter

This sections describes workload characteristics, preferences and constraints of few type of clusters that will benefit from using ESI to share their resources.

1.3.1 High Performance Computing (HPC) & High Throughput Computing (HTC) Clusters

A typical HPC or HTC job is a non-interactive, batch job that is latency insensitive. Users submit jobs to batch systems that may take a long time to execute. These batch systems are designed to maximize the utilization of the system and care about overall computation performed over a long period of time. Such systems will be willing to release compute resources to other clusters to handle their peak demand as long as over the long term they

have gained more computational resources from sharing resources using ESI with other clusters.

1.3.2 Clouds

Applications that are extremely interactive, are robust to performance jitters and network latency and experience high variable demand are well suited to run in a virtualization based cloud. The goal of a cloud is to maximize its profits. They want to be responsive to demand so they don't have to turn away customers, which directly affects their profits.

A cloud provider may be willing to offer its servers to other clusters if there is a surety that when the cloud experiences a surge in demand it can get sufficient servers from other clusters.

1.3.3 Systems Research Testbed

Scientific groups and researchers involved in low-level OS and systems research require access to raw (sometimes specialized) hardware, ability to setup custom automation for deployment of complex environments and support for reproducible experiments. Thus their constraint is to have access to the same or equivalent type of hardware.

When the deadline is close and hardware is limited for experiments researchers will welcome any extra capacity from other clusters as long as they get homogeneous servers and in quantity adequate to perform their experiments. If there is an assurance of getting consistent hardware of preferred configuration they may be flexible towards working at odd hours like nights and weekends.

1.3.4 Government Data Centers

The defense and federal agencies have multiple data-centers ready for response in times of national emergency. Because of the rare nature of national emergencies these data-center are grossly under-utilized while incurring constant maintenance cost. Their preference is to have access to large scale compute resources in times of emergency without having to incur continuous cost of maintaining the infrastructure. According to the RFI[1] released by IARPA their constraint is to be able to scale up rapidly in times of emergency but with security equivalent to an air-gapped private enclave.

If the tenants of the co-location data-center agree to let the federal agencies use infrastructure in times of national emergency it would require all participating organizations to provide some security framework and ability to rapidly re-purpose hardware for government use. Such an arrangement will be both time sensitive and security sensitive.

2 System Architecture

2.1 Requirements

Based on the cluster characteristics described in the previous section, we propose that ESI needs to satisfy the following requirements.

1. **Isolation & Bare-metal Multiplexing:** Each organization has developed in-house deployment methods and processes that help them to maximize usage of their resources. For a system that allows borrowing of hardware resources from one owner to another, it must provide a mechanism to provide complete control to the borrower over physical machines that includes all the capabilities they use to debug an installation when there is a failure. Also, the borrowed portion of the cluster should be isolated from the infrastructure used by other users.

2. **Fast provisioning:** Many organizations would be reluctant to offer their resources outside of their cluster if there is no surety of getting them when they need it. Especially for clusters with short term unpredictable demand and time sensitive response (see 1.3.2:clouds), the ability to have hardware ready for servicing requests rapidly is important. A mechanism that allows clusters to release under-utilized servers to other clusters or expand their cluster by borrowing servers from other organization is an important requirement if clusters with different time-sensitivity to demand are to participate and offer their resources to each other.

3. **Security:** Different organizations value security differently. A systems researcher (see: 1.3.3) may not spend as much time and effort to ensure the security of its system as would a security sensitive organization (use-case: 1.3.4). Sharing a bare-metal node between clusters with different security requirements can be a major source of concern. To make server exchange possible between group with different standards for security we need a mechanism that allows the borrowing organization to verify whether a server matches the security standard of its cluster.

4. **Incentive System:** Building a system that provides fast and secure mechanism for sharing physical servers between organizations does not guarantee that the said sharing will occur. The use-cases are an indicative list of organizations which may not trust each other or would be otherwise in competition with each other. We need a mechanism to encourage exchange of resources where no one organization dictates how the resources are allocated. Ideally, we would want a system that allocates resources where it is needed most; provides incentives to participating clusters to actively share their under-utilized resources. We adopt a marketplace like mechanism that rewards sharing of resources between organizations is an important requirement.

2.2 Design Principles and Choice of Architecture

Among distributed systems the *Microservices Architecture* is a best fit for building such a system. It is a system composed of multiple services, each of which caters to a specific function; can scale on demand without affecting other components; has an independent life-cycle of development; and has a well defined API interface for collaborating with other services. Loose coupling and interaction by message passing via well defined APIs has the advantage of composing the services in more than one way where a microservice can be either setup to work independently or can be setup to manage other microservices [11, 7].

The focus of this thesis is limited to implementing the isolation service, the provisioning service, the security framework and the Incentive System.

1. *The Isolation Service* is a new fundamental layer that allows different physical provisioning systems to share the data center while allowing resources to move back and forth between them. We take an Exokernel-like [19] approach where the lowest layer only isolates/multiplexes the physical resources. It partitions physical hardware and connectivity, while enabling direct access to those resources to the physical provisioning systems that use them.

2. *The Provisioning Service* serves user images that contain the operating system (OS) and applications from remote-mounted boot drives. It relies on a fast and reliable distributed a storage system (CEPH [21], [22] in our implementation) for hosting images of provisioned bare-metal instances.

3. *The Security Framework* presents a new architecture for bare-metal clusters that enables tenants to control tradeoffs between security, price and performance. Security-sensitive tenants can minimize their trust in the provider and achieve similar levels of security and control that they can obtain in their private data center. At the same time, this framework does not impose overhead on tenants that are security insensitive nor compromises the flexibility or operational efficiency of the provider. It includes the attestation service and a Linux based firmware.

4. *The Incentive System* proposed here follows a market based economic model where • every tenants earns revenue for offering resources, • change in overall demand and supply is reflected by dynamic changes in the price of the resource, • auctions decide the best placement of resources among competing demands.

Architecture of a Elastic Secure Infratructure (ESI)

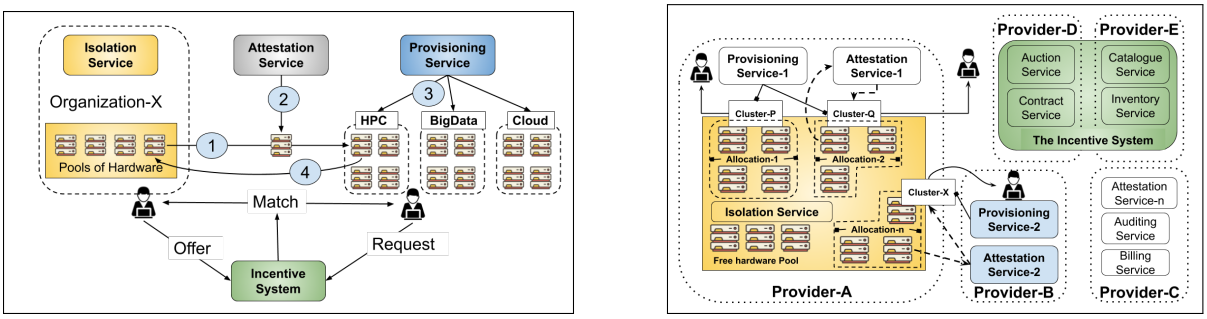

Figure 2·1: (Left) ESI Architecture Overview, (Right) Multiprovider Bare-metal market place: *The Isolation Service* is the only service of the provider (*provider-A*) that tenants have to trust. *Cluster-P* is setup using *Isolation service + Provisioning service-1*. *Cluster-Q* is setup using *Isolation service + Provisioning service-1 + Attestation service-1*. *Cluster-X* is setup using the *Isolation service + Provisioning Service-2 + Attestation Service-2*. Users can also use third party services offered by providers like *Provider-C*. The *incentive system* shows tentative list of services, that may be hosted by different providers, here (*Provider-D,E*)

Figure (2·1-*left*) shows the architecture overview of ESI. Organization-X offers its hardware via the *Incentive System* while customers request hardware from the *Incentive System*. When a match is found, 1. *Isolation Service* allocates the server. 2. *Attestation Service* checks the integrity of the server. 3. *Provisioning Service* sets it up with system and application software 4. *Isolation Service* releases the server back to the hardware pool.

Following the OCX model, a multi-provider elastic secure infrastructure shall be composed of microservices owned by different tenants as shown in Figure (2·1-*right*). *The Isolation Service* is the only service of the provider (*provider-A*) that tenants have to trust. Tenants can choose to setup their clusters depending on their level of trust on the provider. For example, users that fully trust the provider (*Cluster-P*, used internally by *Provider-A*) may simplify their deployment process by not choosing to use the security framework or choose to depend on the provider for its security (*Cluster-Q*). Tenants that do not wish to trust the provider can either host their own services (*cluster-X* used by *provider-B*) or use third party services (*attestation service-n* from *Provider-C*). Similarly, the *Incentive System* is proposed to be a combination of services that may be hosted by different providers (*Provider-D,E*).

This project has involved large number of different people including students, researchers, and developers. I have co-led the design, development and prototype implementation of the isolation service and the security framework while worked as a part of the team that designed and implemented the provisioning service. I am leading the efforts towards building the incentive system.

3 Work Completed

3.1 Hardware Isolation Layer (HIL)

Our implementation of the *Bare-metal Allocation and Isolation Service* is called the "Hardware Isolation Layer" or HIL. It decouples the functionality of resource allocation of bare-metal servers from the functionality of installing systems and application software on the servers. It takes the Exokernel-like[8] approach where the lowest layer only isolates/multiplexes the resources and richer functionality is provided by systems that run on top of it. With HIL, we partition physical hardware and connectivity, while enabling direct access to those resources to the physical provisioning systems that use them. HIL uses a driver based model where a standard API maps to different out of band (OBM) modules used for power-cycling physical servers or calls related to network isolation are handled by switch specific drivers.

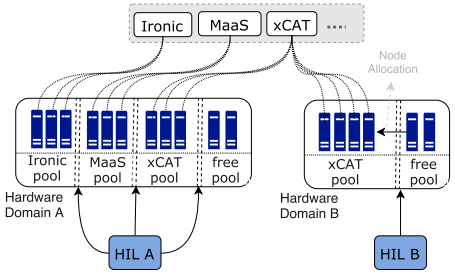

Figure 3·1: HIL provides strong network-based isolation between flexibly-allocated pools of hardware resources, enabling normally incompatible provisioning engines (e.g. Ironic, MaaS, xCAT) to manage nodes in a data center.

This approach makes HIL adaptable to any new OBM module or network control switch. It also allows existing provisioning systems, including Ironic, MaaS, and Foreman, to be adapted with little or no change. Figure 3·1 provides a diagrammatic overview of how HIL can enable provisioning systems to build clusters using servers from different pools of hardware. The fundamental operations HIL provides are *1)* allocation of physical nodes, *2)* allocation of networks, and *3)* connecting these nodes and networks. In normal use a user would interact with HIL to allocate nodes into a pool, create a management network between the nodes, and then connect this network to a provisioning tool such as Ironic or

MaaS. As demand grows, the user can allocate additional nodes from the free pool; when demand shrinks, they may be released for other use. Developed with less than 3,000 lines of code in the core functionality, this is a minimalist and the only service that has to be trusted by the organizations borrowing bare-metal nodes from owning clusters.

3.2 Bare-metal Provisioning Service (BMI)

Our implementation of *Provisioning Service* is called Bare-metal Provisioning Service or BMI. It makes use of remote-mounted boot-drives to host user images containing the operating system and applications. It exploits advancements in disaggregated storage and networking technologies to offer performance comparable to local-disk-based systems. The rapid provisioning and snapshotting capabilities offer elasticity and support for fast transition among different frameworks for datacenter administrators.

3.3 Bolted - The Security Framework

To enable a mult-provider bare-metal exchange between providers with varying security practices Bolted is an implementation of the *security framework* that enables tenants to control tradeoffs between security, price, and performance. Security-sensitive tenants can minimize their trust in the provider and achieve similar levels of security and control 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.

We categorize the threats that the tenant faces into the following phases: *Prior to occupancy:* Malicious (or buggy) firmware can threaten the integrity of a server, as well as that of other servers it is able to contact. A tenant servers firmware may be infected prior to the tenant using it, either by the previous tenant(e.g., by exploiting firmware bugs) or by the cloud provider in-sider (e.g., by unauthorized firmware modification). *During occupancy:* Although many side-channel attacks are avoided by disallowing concurrent tenants on the same server, if the servers network traffic is not sufficiently isolated, the provider or other concurrent tenants of the cloud may be able to launch attacks against it or eavesdrop on its communication with other servers in the enclave. Moreover,if network attached storage is used (as in our implementation) all communication that is not sufficiently secured between server and storage may be vulnerable. Finally, there is a threat to the tenant from denial of service attacks. *After occupancy:* Once the tenant releases a server, the confidentiality of a tenant may be compromised by any of its state (e.g, storage or memory) being visible to subsequent software running on the server

The security framework proposes attestation service (measuring all firmware and software and ensuring that it matches known good values) that can be implemented by the tenant rather than just validation(ensuring that software/firmware is signed by a trusted party).This is critical for firmware which may contain bugs [24, 35,40, 41, 70, 77] that can disrupt tenant security. Attestation provides a time-of-use proof that the provider has kept the firmware up to date. More generally, the whole process of incorporating a server into an enclave can be attested to the tenant. Further, we propose using deterministically built firmware, so that the tenant can not only inspect it for correct implementation but

then easily check that this is the firmware that is actually executing on the machine assigned to the tenant. For tenant-deployed functionality, small firmware with open source implementations are valuable to enable user-specific customization.

3.4 Results

Here we present some results demonstrating the effectiveness of each service in moving resources between mutually non-trusting bare-metal clusters with different security requirements.

Hardware Isolation Layer

Figure 3·2 shows the performance of synchronous HIL API operations as we scale the number of concurrent clients from 1 to 16, while making requests in a tight loop.

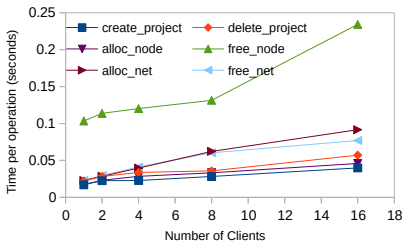

Figure 3·2: Scalability of HIL synchronous operations

As expected, operations that primarily make use of the DB, like allocating or deallocating a project, node or network, complete in less than a tenth of second even with 16 concurrent clients. Freeing a node takes about 5x the time of the other allocation-related operations and degrades more rapidly with increased concurrency because of a call out to ipmitool to ensure that any consoles connected to the node are released before it is deallocated.

Bolted - The Security Framework

To understand the elasticity Bolted supports, we first examine its performance for provisioning servers under different assumptions of security. Figure 3·3 compares the time to

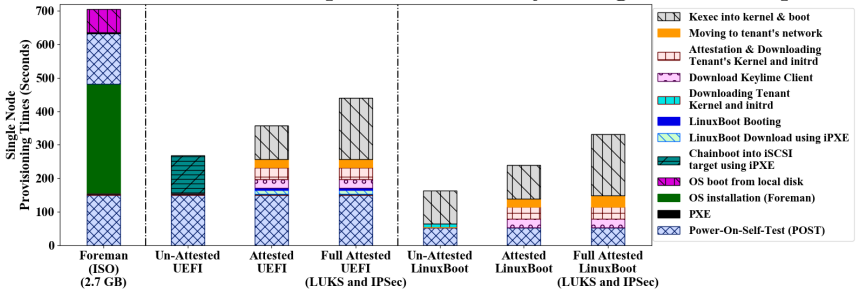

Figure 3·3: Provisioning time of one server.

provision a server with Foreman [**?**] to Bolted with both UEFI and LinuxBoot firmware under 3 scenarios: *no attestation* which would be used by clients that are insensitive to security, *attestation* where the tenant trusts the provider, but uses (provider deployed) attestation to ensure that previous tenants have not compromised the server, and *Full attestation*, where a security-sensitive tenant that does not trust the provider uses LUKS to encrypt the disk and IPsec to encrypt the path between the client and iSCSI server.

There are a number of important high-level results from this figure. First for tenants that trust the provider, Bolted using LinuxBoot burned in the ROM is able to provision a server

in under 3 minutes in the unattested case and under 4 minutes in the attested case; numbers that are very competitive with virtualized clouds. Second, attestation adds only a modest cost to provisioning a server and is likely a reasonable step for all systems. Third, even for tenants that do not trust the provider, (i.e. LUKS & IPsec) on servers with UEFI, Bolted at ∼7 minutes is still 1.6x faster than Foreman provisioning; note that Foreman implements no security procedures and is likely faster than existing cloud provisioning systems that use techniques like re-flashing firmware to protect tenants from firmware attacks.

4 Pending work

This section discusses the open questions needed to be addressed for the completion of the thesis.

4.1 Incentive System

Building a system that provides fast and secure mechanism for sharing physical servers between organizations does not guarantee that the said sharing will occur. A system is needed that increases sharing of resources between clusters in a manner that offers resources to jobs that need it the most.

A centralized system that holds all knowledge about resource availability and job demands may be efficient but difficult to scale. The complexity of managing demands across clusters with different optimization objectives would compound as their number increases. Also, a centralized system may be adequate for sharing resources across clusters owned by a single organization (like silo type A), it is a non-starter in case of a co-location facility or a multi-provider cloud. No tenant or provider organization will be willing to expose their internal demand-utilization structure to a third party, even more so when they are in competition to each other.

We need a system that,

• Allows organizations to have complete control on how they wish to share their resources. • Provides incentives for proactively sharing their resources with others. • Matches resources to the most needful job using fair and transparent mechanisms to resolve the relative importance of jobs. • Offers a revenue mechanism for the providers to realize the OCX model.

We propose an incentive based sharing system that follows a market-based economic model. It uses incentives to encourage rather than force individual participants to offer more goods and services. Every resource is priced using a standard currency. Price of a resource (say a bare-metal server) indicates its overall demand across all clusters at a given point of time. A price rise of a resource indicates that either its availability has reduced or more jobs are demanding that specific resource. It serves as an incentive to organizations to offer more of those resources. Similarly, price drop is an indication of resource abundance or reduction in demand. Organizations may use the price drop as an incentive to fulfill more jobs. Dynamic change in prices allows different organizations to offer their resources and express their demands without revealing any more information than is needed.

The desirable properties that any market should possess in order to be efficient and sustainable is widely studied in economics. Following are the properties [1] that an OCX of

[1] Myerson-Satterthwaite impossibility theorem states that any market can only optimize for 3 out of the 4 proper-

bare-metal servers should have in order to satisfy the constraints enumerated above:

(a) *Individual Rationality:* Also known as *voluntary participation*. The incentive system should be designed such that every organization finds more value in sharing their resources outside the cluster rather than letting them sit unused in a clusters.

(b) *Incentive compatibility:* Also known as *truthfulness*. It should have mechanism that encourages organizations to express, what they believe, is the true value of a given resource or job irrespective of how others value it.

(c) *Pareto Optimal:* Also known as *allocation efficiency* which means that the resource (bare-metal server) is always allocated to the cluster that most values it.

Being decentralized it would be difficult to detect any foul play. Hence safeguards need to be in-built against behaviors that may cause the *market failures*. Two such properties are *adverse selection* and *moral hazard* . The former occurs when transactions occur between two parties, one of which has access to information about the resource that other may not have while later occurs when misleading information is provided by one of the participants involved in the transaction. This effects, if not checked early on, can cause the system to halt (market crash).

Sahil: working on examples of each, in terms of bare-metal node exchange

4.2 Questions I need to answer for the completion of this PhD

1. When there are multiple providers and multiple consumers what is the best way to match the demand with supply such that the computational resources are available to the job that needs it most. How to determine relative importance of jobs when the resources available are less than the cumulative demand. Since it is a decentralized model how is the allocation done which is transparent and fair.

   Sahil: example: do I do fair-share, priority, auctions, reservations, price mechanism etc.

2. Every group/cluster/workload has their own optimization model. What incentives to provide such that everyone is motivated to share their hardware in the marketplace as soon as it is available. Is there a way to measure it. As in, how do I know that they are more willing to share their nodes with others now then they were with respect to some (still need to figure out that) reference.

   Sahil: example: with a centralized scheduler, there is no incentive to do internal migration of jobs for a supplying cluster. In a marketplace mechanism a supplying cluster may reshape its own workload so as to make its machines available in future. (Reservation mechanism)

ties. Fourth being *budget balance* which means that the total amount of money exchanged among the stake-holders is constant. It is a zero-sum game.

3. What are the possibilities where such a market would crash and is there a way to have built in mechanisms to discourage such behavior ?

   Sahil: example: For provider and a customer to get into a future reservation, they will have to park some fraction money with the clearing house for the contract to be formed.

4.2.1 Evaluation Metrics for the Marketplace:

1. **HIL + Marketplace:** Since each group/cluster/workload has its own objective, how much less hardware is required when they trade hardware with each other as compared to when they were statically partitioned.

2. **HIL + Marketplace:** How much "extra-demand" were they able to meet which would not have been possible when they were statically partitioned.

3. **HIL + BMI + Marketplace:** Compared to a diskful approach (foreman) what is the effect of using BMI on the demand a marketplace is able to meet (for each workload).

4. **HIL + BMI + Security + Marketplace:** Implications of ability to share a node between security sensitive cluster with not-so-sensitive cluster on the price of the physical servers. With the ability to share physical servers between security-sensitive clusters vs normal (common security practices) clusters, what is the benefit that secure clusters have over statically partitioned and/or slow provisioning clusters.

4.3 Proposed Timeline

- Allocation and Isolation Service: Done
- Provisioning Service: Done
- Security Framework: Done
- Decentralized Distribution System:
  - Design: End of January 2020
  - Prototype: End of February 2020
  - Evaluation: End of March 2020
- Thesis writing – First Draft: April 2020
- Thesis defense: May 2020

Bibliography

[1] Creating a Classified Processing Enclave in the Public Cloud |IARPA, 2017. https://www.iarpa.gov/index.php/working-with-iarpa/requests-for-information/creating-a-classified-processing-enclave-in-the-public-cloud.

[2] The Massachussets Green High Performance Computing Center, 2019. https://www.mghpcc.org/.

[3] 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.

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

[5] Butler, B. Which is cheaper: Public or private clouds?, Oct. 2016. https://www.networkworld.com/article/3131942/which-is-cheaper-public-or-private-clouds.html.

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

[7] Dragoni, N., Giallorenzo, S., Lafuente, A. L., Mazzara, M., Montesi, F., Mustafin, R., and Safina, L. *Microservices: Yesterday, Today, and Tomorrow*. Springer International Publishing, Cham, 2017, pp. 195–216.

[8] Engler, D. R., Kaashoek, M. F., and Otoole, J. Exokernel: an operating system architecture for application-level resource management. In *ACM SIGOPS Operating Systems Review* (New York, NY, USA, 1995), SOSP '95, pp. 251–266.

[9] Foreman. Foreman provisioning and configuration system. http://theforeman.org.

[10] Kirkwood, G., and Suarez, A. Cloud Wars! Public vs Private Cloud Economics, 2017. https://www.openstack.org/summit/boston-2017/summit-schedule/events/17910/cloud-wars-public-vs-private-cloud-economics.

[11] Newman, S. *Building Microservices*, 1st ed. O'Reilly Media, Inc., 2015.

[12] Tomonori, F., and Christie, M. tgt: Framework for storage target drivers. In *Linux Symposium* (2006).

[13] Van der Veen, D., et al. Openstack ironic wiki. https://wiki.openstack.org/wiki/Ironic.

[14] Weil, S. A. *Ceph: reliable, scalable, and high-performance distributed storage*, vol. 68. 2007.

[15] 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.