Rapid Adjustment and Adoption to MIaaS Clouds

Balaji Viswanathan   Akshat Verma   Bharat Krishnamurthy
Praveen Jayachandran   Kamal Bhattacharya   Rema Ananthanarayanan

IBM Research - India

ABSTRACT

Emerging Managed Infrastructure as a Service (MIaaS) clouds allow enterprises to outsource their IT infrastructure as well as their IT management needs. One of the core tenets of a MIaaS cloud is a standardized service delivery model, allowing the cloud provider to provide infrastructure management services at a lower cost. As opposed to pure IaaS clouds where arbitrary customer virtual machines can be migrated to the cloud, migration to MIaaS clouds require the customer servers to be adapted in a way such that the cloud steady state management stack can manage these virtual machines using the standardized delivery model. In this work, we address the problem of migrating customer workloads to a standardized MIaaS cloud. We present the design and implementation of *Rapid Adjustment Engine* (*RAE*). *RAE* captures the adjustment process across arbitrary customer servers with high diversity in a unified rule framework. It uses rapid image adjustment to reduce the end-to-end migration time and a flexible orchestrator framework to integrate diverse functionalities and associated tools in a single migration process. Our experimental evaluation establishes the ability of *RAE* to enable rapid, reliable and reduced cost migration to MIaaS clouds.

Categories and Subject Descriptors

K.6.4 [**MANAGEMENT OF COMPUTING AND INFORMATION SYSTEMS**]: System Management—*Centralization/decentralization*

General Terms

Management,Performance,Reliability,Design,Experimentation

Keywords

Managed infrastructure as a service, Migration to cloud

1. INTRODUCTION

Cloud computing has emerged as a disruptive technology with the potential to significantly reduce data center costs. Data center costs can broadly be classified into

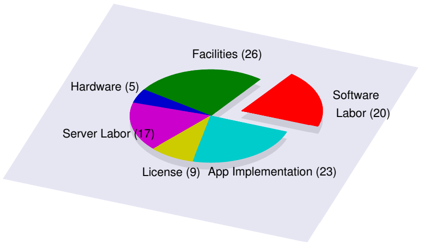


Figure 1: Data Center Cost Breakup (Source:[19])

Hardware Cost, *Server Labour Cost* (steady state), *Facilities Cost* (includes power, cooling, space, and other infrastructure costs), *Software License Cost*, *Application Implementation Cost* (one time cost), and *Software Labour Cost* (steady state software maintenance cost). In a study conducted in 2009 (Figure 1), hardware and facilities costs constituted only about 30% of total data center cost. IaaS clouds use economies of scale to reduce the cost for hardware and facilities components and provide infrastructure services at reduced costs. However, they play no role in reducing other cost components, which add up to about 70% of total data center cost. Further, with the advent of virtualization, the hardware-related costs have decreased. Hence, the relative contribution of components targeted by IaaS clouds to total data center costs is further reducing.

Component	Physical	Virtualized
Server Hardware	517500	67275
System Software	679840	154758
Storage/Network	238000	75375
Facilities	355614	46233
Provisioning	170820	26478
Administration	**1069340**	**536275**
Training	0	8723
Total	3011114	1015117

Table 1: Non-application Costs Breakup (In USD) for 100 server instances (Source: [7])

In a recent case study (Table. 1), the cost of systems administration in a non-virtualized environment was about 33% of the total server cost (excluding any application costs). However, in a virtualized environment, the cost of systems management as a fraction of total systems cost (excluding application costs) exceeds 50%. The actual server provisioning costs in a virtualized environment is less than 5% of total system management costs. The dominant costs of managing systems has led to the emergence of a new breed of cloud, namely Managed IaaS (MIaaS) clouds (e.g., IBM SCE+ [15]).

Emerging MIaaS clouds provide infrastructure as well as

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for pro£t or commercial advantage and that copies bear this notice and the full citation on the £rst page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior speci£c permission and/or a fee.
Middleware 2012 Industry Track, December 3-7, 2012, Montreal, Quebec, Canada.
Copyright 2012 ACM 978-1-4503-1613-2/12/12 ...$15.00.

systems management as part of the service and allow customers to focus purely on application management. MIaaS clouds also use economies of scale to provide managed services at an attractive price point by standardizing the managed virtual machines allowing automation of almost all managed services. This includes using a standard set of system management tools for each managed service. It has been observed that standardization of the operating environment can allow one engineer to support up to 80% additional server instances [11]. Hence, MIaaS clouds require all instances to be standardized allowing automated system management. One method of standardizing cloud instances for an MIaaS cloud is to re-implement customer workloads on pre-created standardized golden master images, although it can be very expensive. For example, it has been observed that application implementation may cost as much as 23% of total data center costs (Figure 1). Another set of white papers found that application installation, configuration, integration, and deployment costs are up to 10 times the hardware cost for enterprise web servers [3, 4]. Even though exact numbers vary, many enterprise applications have implementation costs up to 5 times their hardware cost [10].

The complexity of re-installing applications has led to the emergence of image-based migration where an entire enterprise application consisting of multiple server instances is migrated in a wave. The source servers are converted to virtual machine images, transferred to the cloud, and provisioned as virtual machines. This is followed by application reconfiguration to make the application aware of changes in environment parameters (e.g., IP addresses). Image-based migration is promising for pure IaaS clouds as arbitrary customer images can be onboarded on the IaaS cloud as long as the operating system of the customer server is supported by the cloud hypervisor.

Image-based migration to MIaaS clouds, on the other hand, poses significant new challenges. A MIaaS cloud requires every managed virtual instance to meet the cloud standard, allowing it to be managed using standardized tools and processes. Also each customer image may vary from the cloud standard in different aspects, thereby requiring a diverse set of adjustments to be made on each of them to ensure they meet the cloud standard and can be managed by the cloud. The process of adjustment of images to MIaaS standard is currently manual and error-prone, requiring experienced administrators who can build customized recipes for each source server instance.

Contribution: In this work, we address the problem of migrating enterprise applications to a MIaaS cloud. We present the first comprehensive look at managed infrastructure clouds and identify standardization as a significant challenge for image-based migration to MIaaS clouds. We focus on the standardization of arbitrary customer images in an automated, reliable, cost-efficient, and scalable manner. Towards this goal, we present the design and implementation of *Rapid Adjustment Engine*. *Rapid Adjustment Engine* uses a novel rule engine, a flexible orchestration framework and offline image adjustments to reduce the cost of migration. Our experimental evaluation with diverse workloads establish the effectiveness of *Rapid Adjustment Engine* to reduce the time and cost of migration.

2. BACKGROUND

2.1 Image-based Migration to Cloud

Image-based migration of enterprise workloads to a cloud is a fairly challenging problem requiring specialized services [6, 9, 2, 5]. Even though the details differ, most migration vendors broadly start with (1) discovery of source servers, their resource utilization, the software deployed and dependencies, (2) an Analysis phase to plan the migration, 3) a Transfer phase to converts the source servers into VMs, 4) a Remediation phase reconfigures the applications to deal with changes in the operating environment (IP address, host names, MAC addresses, etc) followed by 5) test and adoption in the cloud.

Migration to pure IaaS clouds is thus a challenging but well understood problem. A variety of tools exist for each phase of the IaaS migration process (e.g., [17] for Discovery, [1] for Analysis, [8] for transfer). However, migration to a MIaaS cloud requires an additional step of *Standardization* or *Adjustment*, between the transfer and application remediation steps, which includes making changes to the source instance (operating system and middleware) to ensure that the instance meets the cloud standard. This process requires understanding the cloud standard and coming up with a set of steps that will convert any arbitrary customer server to a cloud-compliant virtual machine.

2.2 Standardization Challenges and Goals

Any standardized delivery model consists of

Specified Standard: A MIaaS cloud standard usually specifies a set of management tools that need to be installed on all managed virtual machines. System management tools may include monitoring, backup, patch management, license management, and domain management, among others. The standard may also include specifications of what patches need to be applied to manage servers, as well as any security compliance specifications like password length, password expiration, etc. Finally, specified standard includes any network specifications (e.g., network zones, vNICs in a zone, IP address) as well as access specifications (e.g., all managed VMs need to have ssh running).

Unspecified Standard: Unspecified standards include any specifications needed for the specified standard to be met including removal of conflicting tools and satisfying prerequisites for the installation of the cloud tools. If a cloud agent installation script makes certain assumptions about the directory structure or environment variables in the customer image, standardization needs to automatically identify those and make suitable adjustments in the customer server.

Ensuring that an arbitrary customer image meets the specified standards in a reliable and cost-efficient fashion is challenging. The problem is further exacerbated by the unspecified standards, which are not known *a priori*. Further, there is a wide disparity in the characteristics of servers across customers as well as between servers for the same customer. To take an example, some servers may use ITM for monitoring and other servers may use HP Insight before migration. The standardization tool needs to automatically assess that ITM uninstall should be performed for the former customer and HP Insight uninstall for the latter. Hence, for each server we need to identify the set of adjustments that need to be performed. Next, we present the key challenges that need to be addressed by a standardization framework.

Speed (rapid): Migration needs to be completed in a pre-specified change window and one of the main goals of migration is to increase the speed at which the overall adjustment process is accomplished.

Reduced cost: The adjustment process presents additional cost to traditional IaaS migration and minimizing the total labour cost is a key goal.

Reliability: Standardization is a complex process requiring customized steps for each source server image. Customized recipes may have bugs and operator errors may be introduced due to a variety of tools being used. Hence, ensuring end-to-end reliability is an important goal.

Extensibility: A generic framework needs to ensure that it can easily be extended to adapt to new types of source images as well as changes in the cloud standard.

3. DESIGN OVERVIEW

Rapid Adjustment Engine introduces the following architectural elements (Figure 2) to greatly reduce the cost of migration to MIaaS cloud.

Rule-driven Standardization: The variability in the adjustments needed for each customer image increases the overall cost of migration and makes the process unreliable as manual errors crop in. One of the key design choices made by *Rapid Adjustment Engine* is to define a generic rule framework that allows customized actions to be executed on specific images. A standardized set of rules ensure that all adjustments are consistent with a framework and ensure reliability of the migration process. Further, the rule framework enables additional optimizations, which we detail next.

Extensible Orchestration Framework: Our second design choice is to ensure complete automation of the adjustment process. For any given image, there are a wide variety of adjustment actions. Each type of adjustment action may require its own set of tools and need to interact with various cloud management components. *Rapid Adjustment Engine* allows a generic orchestration framework, where new orchestrators can be defined. Each orchestrator only needs to ensure that individual actions are defined using the rule framework. Orchestrators are free to use any external tools to interact with the virtual machines or cloud management components in order to perform adjustments. This generic orchestration framework allows us to have a unified automation framework that can cater to various types of adjustments. Further, new orchestrators and new rules can be added, allowing easy extensibility of the migration process.

Image Adjustments: The speed (or turnaround time) of migration is determined by the total time taken to execute various adjustment actions. To migrate a source workload to cloud, we need to create a virtual machine instance in a landing zone and perform adjustment actions. The degree of parallelism supported in the landing zone may throttle the throughput of migration. Hence, *RAE* explores application of adjustments to virtual machine images (as opposed to running virtual machine instances). Image adjustment can be parallelized across a lot of workloads and does not have many resource bottlenecks. Hence, image adjustment has the potential to greatly increase the speed of migration.

4. RAPID ADJUSTMENT ENGINE

4.1 Rule Framework

One of the key challenges in adjustment of server instances to the cloud environment is the reliability of the adjustment process. Source servers in a customer environment

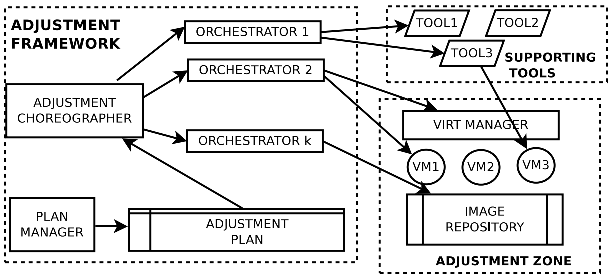


Figure 2: Rapid Adjustment Framework. Adjustments are applied as a sequence of $IF < COND > THEN < ACTION >$ rules executed by orchestrators designed to handle specific categories of standardization. Each action may be executed either as an instance adjustment or as an image adjustment.

differ greatly thereby requiring different adjustments. Currently, migration engineers use their judgment to define a customized set of adjustment steps for each server instance. Automation of steps into scripts can help reduce the labour cost of migration but would still require the enginer to choose and execute only the appropriate scripts for a given server instance. This impacts the reliability of the process. Executing inappropriate scripts could result in errors, which are labour-intensive to fix. To address the reliability problem, we have incorporated a rule framework in the Rapid Adjustment Engine.

RAE allows **Adjustment Actions** to be defined in the tool. The set of all adjustment actions is a superset of all actions that may need to be executed on any source workload being migrated. Each action has an associated script, which performs the action. The scripts associated with the individual actions need to be as self contained as possible and should take care of any looping requirements of the action. Additionally, the scipts need to conform to specified standards regarding return of specified return codes and logging to enable the orchestrator to proceed as appropriate. The **Master Adjustment Plan** consists of a sequence of rules, where each rule is of the form IF <CONDITION> THEN <ACTION>. Both the condition and action parts of a rule are enabled using adjustment scripts registered in the tool. An **Instance Adjustment Plan** is generated for each instance using the master adjustment plan and the peculiarities of the server being migrated. An instance adjustment plan is generated by executing the rules, and lists only those actions whose condition is met.

The Master Adjustment Plan essentially consists of all the guidance needed during the adjustment process. It is created once for each data center that is being migrated to the cloud. The plan is created by experts and defines the semantics for migration. It also captures any ordering requirements of the individual adjustment actions along with their conditions. Once a master adjustment plan is created, the Rapid Adjustment Engine ensures the reliability of the adjustment process as all actions performed by the engine are in conformance with the rules defined by the Master Adjustment Plan.

We have implemented the rule engine as a Java layer on DB2. The set of adjustment actions are encoded in a table, which assigns a unique action ID to each action. It also captures the path of the script and the orchestrator associated with the action. The Java layer provides interfaces to add, remove, or edit actions. The adjustment plan is stored as a separate table with columns mapping to the action table. One of the columns captures the condition and another column captures the action associated with the rule. The rule engine executes all the rules in order. For a given rule, it invokes the relevant orchestrator to execute the condition. The orchestrator executes the associated script and returns the return code to the rule engine. Based on the return code, the rule engine either includes the action script in the instance adjustment plan of the instance or excludes it.

4.2 Rapid Image Adjustments

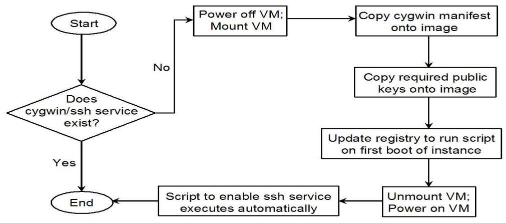

Start
Does cygwin/ssh service exist?
No
Yes
Power off VM; Mount VM
Copy cygwin manifest onto image
Copy required public keys onto image
Update registry to run script on first boot of instance
Unmount VM; Power on VM
Script to enable ssh service executes automatically
End

Figure 3: Image adjustment for standardized access

One of our key design choices is to apply adjustments offline on the virtual image whenever possible, rather than applying it online on the running instance. In order to apply an adjustment offline, we first create a manifest consisting of the set of file system changes (including registry for windows instances) that the adjustment brings about. This manifest could be created manually by an expert or obtained by applying the adjustment online on a pilot instance and recording all changes to the file system. On subsequent client instances, the adjustment is carried out on the dormant instance by mounting the virtual image disk, applying all the changes contained in the manifest on to the mounted file system, and then booting up the instance again when required. All the above steps are automated and do not require manual intervention.

An example offline adjustment is access enablement for migrated Windows images (Figure 3). The condition scripts checks for a batch of VMs if they have cygwin and SSH service enabled. For those VMs that do not have cygwin,the orchestrator powers the VM off and mounts the image file. It copies a manifest containing cygwin installation and copies public SSH keys of trusted hosts in the image. Appropriate registry values are updated on the image to start the sshd daemon on startup and the VM is booted up. Once the VMs are back up, access is enabled on the customer images.

4.3 Extensible Orchestration Framework

The rule framework allows an instance adjustment plan to be created for each virtual machine instance being migrated to the cloud. However, different actions in the plan involve different components in order to perform the required adjustment. For example, instance power on/off control requires virtual machine management access, image adjustment requires tools to mount instance disk and manipulate them. Patching requires access to a centralized patching server and other actions require access to the instance over network. Install actions may require access to a shared software repository. Performing manual tasks across the various components is cumbersome and error prone. In order to handle the wide variety of supporting tools, *RAE* implements a generic orchestration framework. The adjustment actions are classified into different adjustment types, and the *Rapid Adjustment Engine* choreographer invokes an orchestrator for each adjustment type. The orchestrator internally encodes the various management components it may use to perform the required adjustment. The orchestration framework still requires all orchestrators to define the rules using the rule framework described earlier. We have implemented 5 orchestrators to perform different types of adjustments, which we describe next. Table 2 provides examples of adjustment actions captured as rules. In the interest of space, we omit presenting the entire master adjustment plan in our implementation.

Rule	Type
If <Cygwin not installed> then <Install Cygwin>	Access Enablement
If <SSH keys not present> then <Insert SSH keys>	Access Enablement
If <TSCM agent installed> then <Uninstall TSCM agent>	Uninstall
If <TEM client not installed> then <Install TEM client>	Install
If <Not patched to required level> then <Apply relevant patches>	Patching
If <Min password length not 8> then <Set min password length = 8 >	Compliance
If <Password history count not set> then <Set password history count>	Compliance

Table 2: Sample Adjustment Rules

4.3.1 *Access Enablement Orchestrator*

A managed cloud uses a standard network configuration and access mechanism to manage the supported VMs. *Access Enablement Orchestrator* ensures that customer VMs also support the standard access mechanism and network configuration. The adjustment actions include setting IP address, gateways and masks. Another rule installs a SSH service using cygwin, if not present already. Further, authorization keys are copied over to the VM to enable password less access from management components.

4.3.2 *Software Uninstall*

A key 'unspecified standard' that customer VMs need to meet is to not have any software installed that conflicts with a management software, which is part of 'specified standard' of the cloud. The *Uninstall* orchestrator performs uninstall of all such conflicting software. The exact software packages to be uninstalled is obtained by leveraging the adjustment plan and the software discovery data. The condition part of a rule in the plan uses discovery data to check if a conflicting software is present. The action part of the rule performs the actual installation. Uninstaller packages and binaries are downloaded to the instance by mounting a global repository as a network drive.

4.3.3 *Software Install*

The software install stage installs supported systems management software and agents to support cloud onboarding and steady state management of the instance. In addition to installing new software, this orchestrator includes any actions which change configuration files and/or settings. Further, if any installed software needs to be registered with a central server (e.g., ITM agent registered with ITM server), this is also performed by this orchestrator.

4.3.4 *Patch Orchestrator*

Making client instances comply with cloud standards includes applying OS, application, and security patches to bring the instance to the latest patch level. We use Tivoli Endpoint Manager (TEM) for patch management. A reference instance for each OS that is compliant with cloud standards is scanned to decide which patches need to be applied

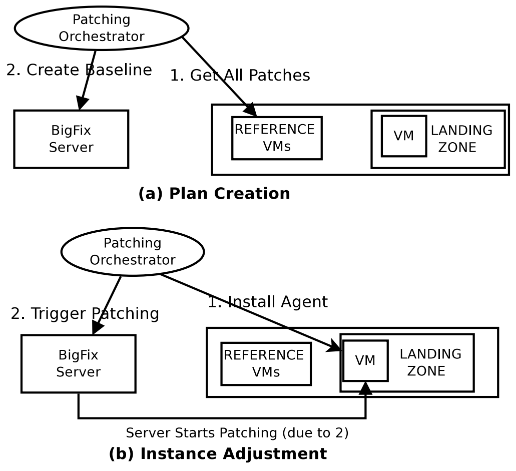


Figure 4: During plan creation, baselines are created using cloud reference VMs. During migration, instances are patched as per the baseline.

(Figure 4(a)). These patches are included in a baseline on the TEM server, which is a group of patch actions that can be deployed using a single command. A baseline is created for each operating system type. A Perl script uses TEM API to apply the patch actions in the baseline onto client instances.(Figure 4(b)).

4.3.5 Compliance Orchestrator

For each client instance, the compliance orchestrator checks for each of the controls or requirements as defined in the applicable security standard of the cloud, and if the check fails, applies the necessary fix to ensure that the instance is compliant. These checks and fixes are implemented through batch files and shell scripts, which take security-related parameters (e.g., minimum length or maximum age for a password).

5. EXPERIMENTAL EVALUATION

In order to experimentally evaluate the effectiveness of *Rapid Adjustment Engine*, we performed a large number of experiments. Our experimental setup is a small scale representation of a popular MIaaS cloud. We migrated 30 virtual machine images, sampled randomly from multiple customers, on a VMWare managed cluster in the cloud. The instances were hosted on 3 ESX hosts, each with 32 cores at 2.26 GHz and 36 GB RAM. The images were stored on a SAN consisting of 6 disks, each with 2 TB storage. *RAE* was executed on a VM with 4 virtual CPUs and 16 GB memory.

5.1 Turn-around Time Reduction

We first evaluate the reduction in turn-around time with the use of offline image adjustments (Figure 5(a)). The two adjustments we performed offline on the image were Cygwin installation and inserting keys for password-less SSH. For these adjustments, we compare the time taken by RAE with a process that performs the adjustments on a running instance. The online adjustment process takes significantly longer to complete and the time taken increases linearly with the number of instances that need to be adjusted. RAE is able to adjust larger number of images with only a marginal increase in time establishing the effectiveness of rapid image adjustments.

5.2 Labour Cost Reduction

In this experiment, we measure the labour time needed to perform the end-to-end adjustment for both *RAE* and a migration engineer performing the adjustments manually, and report the results in Figure 5(b). We observe that RAE is able to significantly reduce the labour time involved by automating the orchestration of the adjustments and executing them in parallel. We assumed that at most 8 instances would be adjusted in parallel using RAE (this limitation is purely an artifact of how many a migration engineer using *RAE* can manage in parallel). In contrast, the labour time involved for manual adjustment was longer and increased linearly with the number of instances that needed to be adjusted.

5.3 Increased Reliability due to Complexity Reduction

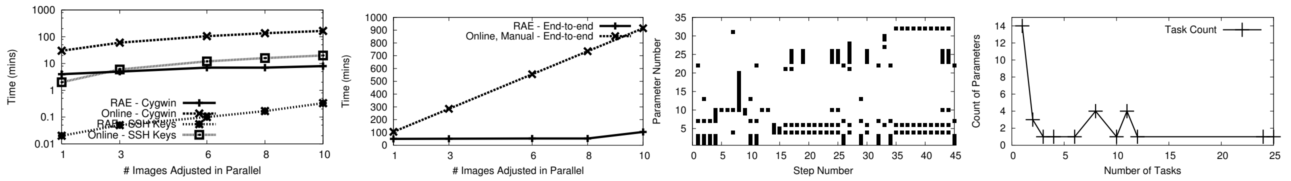
Parameter	Manual	Proposed
Num Tasks	45	5
Context Switches	12	1
Param Count	32	3
Param Usage Count	136	3
Memory Size	18	3
Memory Depth	27.8 mins	0 mins
Memory Latency	16.9	0

Table 3: Complexity Reduction using the rule-driven approach

Traditional techniques to measure reliability require conducting a statistically significant number of experiments with all techniques under comparison for the same input. This would take an inordinately large amount of time. On the other hand, reliability numbers obtained by experimenting with 30 images is not sufficient. Hence, we indirectly measure the reliability of the process by measuring the complexity of the manual standardization process against *Rapid Adjustment Engine*. We use the model of configuration complexity proposed in [12], which has been used by other studies as well [20]. We restrict ourselves to only the parameters applicable to the migration problem.

Table 3 captures the aggregate service complexity of the standardization process using manual scripts and proposed *RAE*. The number of tasks performed manually are 45 whereas *RAE* requires a migration engineer to only trigger 5 different orchestrators. Context switches captures the number of times an end user needs to switch between different subsystems. The manual process requires switching between as many as 12 systems for standardizing a single image. The automated process only requires dealing with *RAE*. The number of parameters that need to be manually configured exceed 30 and they need to be used a total of 136 times. Further, of these 18 parameters are used multiple times and need to be remembered by the migration engineer (captured as Memory Size parameter). Further, the average length of time that these parameters need to be remembered is 27.8 minutes using the manual process. Finally, memory latency captures the number of other parameters that need to be used between two usages of a given parameter, and this value is zero for *RAE*. We observe that *RAE* reduces the service complexity by an order of magnitude across the entire spectrum of complexity parameters.

Figure 5(c) captures the usage of configuration parameters across the 45 tasks (actions) performed by a migration engineer. We wanted to check if the usage of parameter is clustered in time. Such a scenario may allow a migration engineer to deal with only a subset of parameters at a time. We observed that there are no obvious patterns of the parameters employed between different tasks. There are some



(a) Impact of Image Adjustments (b) Impact of Orchestration on Total Time (c) Usage of Parameters Across the Tasks (d) Histogram of Parameter Usage

Figure 5: Experimental evaluation

tasks that exhibit both temporal locality (tasks happen close to each other) as well as spatial locality (tasks employ the same parameters). However, there are less than 5 such tasks out of a total of 45 tasks. This relatively random usage of parameters across time adds significant complexity to the manual process leading to lack of reliability. On the other hand, our automated process requires the migration engineer to only use 3 parameters, which are the URL and the access credentials for *RAE*.

We also investigate the frequency of parameter usage in the MIaaS cloud migration process. We observe a high degree of variance in the histogram of parameters (Figure 5(d)). Some parameters are used by as many as 25 tasks, whereas many parameters are used by a single task. Hence, there are some common parameters (e.g., the hostname of the server being migrated) used by multiple tasks. On the other hand, there are many parameters that are task-specific (e.g., Bigfix server used for patching task). The large number of task-specific parameters are difficult to remember and increase the complexity of the manual process.

6. RELATED WORK AND CONCLUSION

Research work in this space of cloud migration has focused on cost benefits and risk assessment ([16]), live migration of servers, both within the same data center or across data centers ([13],[18]) and complexity of the application and services migration to the cloud [14]. Various products and tools such as Nimbula Director [6], Racemi Cloudpath [9], Cloudops [2], and NI2's cloud migration management [5] support enterprises in the migration of their applications and infrastructure to the cloud. In most of these instances, white papers and product literature describe the process at a high-level, although details regarding cost estimates and planning the actual migration execution are still highly proprietary, and/or closely held intellectual assets.

In this work, we present the first comprehensive view of image-based migration to MIaaS cloud. We establish standardization of customer servers as a key challenge, which differentiates migration to MIaaS clouds from migration to pure IaaS clouds. We identify some of the key issues plaguing migration to MIaaS clouds and preventing adoption of the cloud model by enterprises. We present the design and implementation of *Rapid Adjustment Engine* and show that it significantly reduces the duration, labour cost and complexity of the migration process. *Rapid Adjustment Engine* provides a formal framework to the ad-hoc manual migration process and ensures end-to-end reliability.

7. REFERENCES

[1] CiRBA Data Center Intelligence. http://www.cirba.com/product/CiRBA-Product-Overview.htm.

[2] CloudOps - cloud migration services. `http://www.cloudops.com/cloud -migration-services/`.

[3] Crimson Consulting Group White Paper: Cost of Ownership Analysis. http://www.oracle.com/us/products/middleware/crimson-weblogic-websphere-tco-402458.pdf.

[4] Crimson Consulting Group White Paper: Cost of Ownership Analysis. http://www.oracle.com/us/products/middleware/application-server/weblogic-vs-jboss-460235.pdf.

[5] NI2 cloud migration management. `http://www.ni2.com`.

[6] Nimbula Director. `http://nimbula.com`.

[7] Parallels Virtuozzo Containers:Total Cost of Ownership Analysis. http://www.parallels.com/r/pdf/wp/pvc/TCO_Analysis_WP.pdf.

[8] Platespin Migrate. `http://www.novell.com/pro ducts/migrate/`.

[9] Racemi CloudPath. `http://www.racemi.com`.

[10] White Paper: Comparing the Total Cost of Ownership of Business Intelligence Solutions. http://www.birst.com/why-birst/resources/whitepapers/comparing-total-cost-ownership-business-intelligence-solutions, 2010.

[11] IDC Linux Standardization White Paper. http://www.redhat.com/f/pdf/IDC_Standardize _RHEL_1118_Exec_summary.pdf, 2011.

[12] A. Brown, A. Keller, and J. Hellerstein. A Model of Configuration Complexity and its Application to a Change Management System. In *IM*, 2005.

[13] C. Clark, K. Fraser, S. Hand, J. G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield. Live migration of virtual machines. In *NSDI*, 2005.

[14] M. Hajjat et al. Cloudward Bound: Planning for Beneficial Migration of Enterprise Applications to the Cloud. In *SIGCOMM*, 2010.

[15] IBM, Inc. IBM SmartCloud Enterprise+. `http://www-935.ibm.com/services/us/en/managed-cloud-hosting/`.

[16] A. Khajah-Hosseini et al. Decision support tools for cloud migration in the enterprise. In *IEEE CLOUD*, 2011.

[17] K. Magoutis, M. Devarakonda, N. Joukov, and N. G. Vogl. Galapagos: Model-driven discovery of end-to-end application storage relationships in distributed systems. In *IBM Journal of Research and Development*, 2008.

[18] K. Nagin et al. Inter-cloud mobility of virtual machines. In *SYSTOR*, 2011.

[19] R. O'Hara, P. Ross, and R. Wright. Driving cost out of it. In *Microsoft System Center. https://www.microsoftsalesuniversity.com/infrastructure/resources/content/Presentation_Driving Cost Out of IT.pptx*, 2009. [Last Accessed Mar. 15, 2012].

[20] W. Zheng, R. Bianchini, and T. Nguyen. Automatic Configuration of Internet Services. In *Eurosys*, 2007.