2011 Third IEEE International Conference on Coud Computing Technology and Science

Comparing VM-Placement Algorithms for On-Demand Clouds

K. Mills, J. Filliben and C. Dabrowski
Information Technology Laboratory
NIST
Gaithersburg, MD USA
{kmills, jfilliben, cdabrowski}@nist.gov

***Abstract*—Much recent research has been devoted to investigating algorithms for allocating virtual machines (VMs) to physical machines (PMs) in infrastructure clouds. Many such algorithms address distinct problems, such as initial placement, consolidation, or tradeoffs between honoring service-level agreements and constraining provider operating costs. Even where similar problems are addressed, each individual research team evaluates proposed algorithms under distinct conditions, using various techniques, often targeted to a small collection of VMs and PMs. In this paper, we describe an objective method that can be used to compare VM-placement algorithms in large clouds, covering tens of thousands of PMs and hundreds of thousands of VMs. We demonstrate our method by comparing 18 algorithms for initial VM placement in on-demand infrastructure clouds. We compare algorithms inspired by open-source code for infrastructure clouds, and by the online bin-packing literature.**

Keywords- cloud computing; resource allocation; simulation

I. Introduction

Paxson and Floyd [1] describe many difficult problems impeding simulation of large data communication networks, which typically require hundreds of parameters that can each take on millions of values and that can also record hundreds of response variables, which might represent aspects of fewer significant underlying behaviors. The same can be said for most simulations of large distributed systems, such as on-demand infrastructure clouds.

We have developed an objective method to compare resource-allocation algorithms in simulations of large distributed systems. Our method involves several steps: (1) developing a reduced-parameter model for a large distributed system of interest, (2) conducting a sensitivity analysis to determine the most significant model behaviors and the parameters that most influence those behaviors, (3) applying two-level orthogonal fractional factorial experiment design [2] to construct a set of parameter combinations under which resource-allocation algorithms should be compared and (4) using multidimensional data analysis techniques to find patterns revealing significant similarities and differences among the algorithms being compared. In previous work [3-4], we applied our method to compare proposed congestion-control algorithms for the Internet. Also in previous work [5], we demonstrated the first two steps in our method, when applied to on-demand infrastructure clouds. We constructed a reduced-parameter model (explained below in Sec. III) and we conducted a sensitivity analysis that revealed eight behavioral dimensions and six influential parameters.

In this paper, we demonstrate steps three and four in our method, using the results from our sensitivity analysis to construct 32 parameter combinations under which we compare the macroscopic behavior of 18 possible algorithms for initially placing virtual machines (VMs) on physical machines (PMs). Our comparative conditions encompasses cases with up to $O(10^4)$ PMs and $O(10^5)$ VMs. While there are many possible algorithms to investigate (as explained below in Sec. II), we elected to focus on algorithms inspired by a combination of the Eucalyptus open-source code [6] and the online bin-packing literature [7-8]. Eucalyptus inspired us to evaluate two-level algorithms that first choose a cluster for VMs in a related request and then choose nodes within the selected cluster. The literature for online bin-packing inspired us to adopt algorithms based on well-known heuristics that can provide good (not optimal) results without infeasible computation.

Our paper makes three main contributions. First, we demonstrate an objective method for comparing possible VM-placement algorithms through simulation of large, on-demand infrastructure clouds. While we restrict our comparison to 18 selected algorithms, the approach we use should be applicable to compare any set of competing algorithms. Second, we generate some insights regarding two-level VM-placement algorithms, showing that choice of cluster has larger influence, than choice of nodes, on macroscopic behavior in an infrastructure cloud. We also provide observations about specific pairs of algorithms, where each pair combines a criterion for choosing a cluster with a heuristic for choosing PMs within a cluster. We also discuss some tradeoffs among algorithms. Third, we provide evidence showing that, on average, different algorithms for initial VM placement in on-demand infrastructure clouds yield only small quantitative differences in many of the 42 responses we measured (as explained below in Sec. IV). On the other hand, we show that selection of the algorithm for choosing a cluster can lead to very large difference in provider revenue, when aggregated over time.

The remainder of this paper is organized as follows. In Sec. II we describe the general area of VM-placement research in infrastructure clouds, setting our study within this larger context. In Sec. III we describe our model and identify both fixed and varied parameters used in our study. We give values for fixed parameters, but postpone defining values for variable parameters until Sec. IV, where we describe our experiment design. In Sec. V we present our results and related analysis methods. In Sec. VI we discuss our findings. We close in Sec. VII with conclusions and future work.

978-0-7695-4622-3 2011

DOI 10.1109/CloudCom.2011.22

II. RELATED WORK

The literature identifies that VM-placement decisions can be made under any of at least three different regimes [9]: (1) reservations [10-11], (2) on-demand access [11] and (3) spot markets [10-12]. In one reservation regime [11], a user pays a fee per instance per VM type for a period (e.g., one year) during which the specified VMs may be acquired at a discount from published usage charges. In on-demand access regimes, a user simply requests a specified number of one or more VM types needed immediately, and pays for VM usage according to a fixed schedule of fees. In spot markets, a provider's prices fluctuate over time and a user specifies the usage rates they are willing to pay for requested VMs. When the provider price falls to or below the user's willingness to pay, then the user's requested VMs are launched. Should the provider price subsequently rise above the user's willingness to pay, then the user's VMs are terminated, and can only be restarted when the price falls to the level the user is willing to pay. In the grand scheme of resource-allocation decision making, one can envision PMs migrating back and forth among three pools, each assigned to one of the three regimes, as demand for VMs varies. Consideration of how best to allocate PMs to each pool would seem a ripe area for research [9]. We restrict our study to consider only on-demand access.

In on-demand clouds, there are potentially two types of VM-placement decisions to be made: (1) initial placement [13-23] and (2) migration (and/or resizing) of VMs over time [24-30], as PM availability changes, as consolidation is needed to conserve power and in response to the degree to which service-level agreements (SLAs) are being achieved. Most previous research on initial VM placement considered only PMs within a single cloud, but in one case [22] placement decisions considered which of several clouds to choose. In the existing literature, initial placement and VM migration are usually considered as separate topics, though in some cases similar algorithms may be adopted. Future research might consider interaction between initial placement and migration decisions, especially under situations where tradeoffs are needed among power conservation, SLAs, revenue maximization and reliability. We restrict our study to consider only initial VM placement.

One could consider initial VM placement in on-demand clouds at two levels: (1) cluster and (2) node (i.e., PM). When VMs communicate, placing them on the same cluster makes good sense because communication among the VMs will be local to a cluster switch. Most existing research [13-23] considers PMs as an unstructured pool, where restricting VMs to a shared cluster would be accomplished by designating a Boolean attribute, one of potentially many attributes over which some optimization algorithm or bin-packing heuristic would be executed. In our study, guided by the open-source code in Eucalyptus (v1.6) [6], we adopt explicit use of two distinct decisions levels: (1) choosing a cluster for all VMs in a given request and then (2) choosing specific PMs within the selected cluster. Taking this course is the same as assuming that all VMs within a single request will communicate. VMs that need not communicate would then be included in separate requests.

In most VM placement algorithms, PMs are partitioned into two sets: (1) those that meet some criteria and (2) those that do not. Subsequently, the set of PMs that meet the criteria are ordered, and VM placement attempts are made starting with the first PM on the list, and continuing until all VMs have been placed or until the set of qualified PMs is exhausted. Various criteria have been used to order qualified PMs. For example, many researchers [13, 16, 18, 23, 27] adopt ordering heuristics based on the literature associated with online bin packing [7-8]. Other schemes extend those heuristics by adding specific attributes (e.g., CPU usage, network and disk controller usage, and memory usage), summarized into a weighted value used to order PMs or to assign categories (e.g., star ratings [15]) that can be used to order PMs. In some schemes, attributes used to order PMs are determined by individual VM users [23, 30, 31], while in other schemes attributes are determined by the provider [13, 14, 19, 27], or user and provider attributes are combined [21, 22, 23, 26]. To limit our study, we elected to use heuristics based on those found in online bin-packing literature. The method we use to compare placement heuristics should be applicable to any specific set of VM-placement algorithms that one wishes to compare.

III. MODEL

We based our study on Koala, a discrete-event simulator inspired by the Amazon Elastic Compute Cloud (EC2)[1] [32] and by the Eucalyptus open-source software [6]. Using published information describing the EC2 application programming interface (API) [33] and available virtual machine (VM) types [34], Koala models essential features of the interface between users and EC2. Intended to study algorithms for initial VM placement, Koala models only four EC2 commands: *RunInstances*, *DescribeInstances*, *Reboot Instances* and *TerminateInstances*. The internal structure of Koala is based on the Eucalyptus (v1.6) open-source cloud software. Specifically, Koala models three Eucalyptus components: *cloud controller*, *cluster controller* and *node controller*. As in Eucalyptus, Koala's simulated cloud, cluster and node controllers communicate using Web Services [35], which Koala also simulates.

Koala modifies the design of Eucalyptus in three ways. First, Koala extends the Eucalyptus *RunInstances* command to allow multiple VM types within a single request, which appears possible in EC2. Second, Koala avoids centralization of node information at the cloud controller, permitting simulation of clouds up to $O(10^5)$ nodes. Third, Koala allows competing *RunInstances* to proceed partially in parallel (serializing only the commitment phase), which prevents long queuing delays during periods of intense user requests. In lieu of simulating details of a hypervisor and guest VMs, Koala includes an optional sub-model based on analytical equations representing VM behavior with or without tasks.

[1] Any mention of commercial products is for information only; it does not imply recommendation or endorsement by NIST.

Koala is organized as five layers (see Fig. 1): (1) demand layer, (2) supply layer, (3) VM placement layer, (4) Internet/Intranet layer and (5) VM behavior layer. We describe each layer in turn, omitting the VM behavior layer, which is not used in the experiments discussed here. We denote experiment input parameters using designators *x1* to *x6* (see Table V) and outputs as *y1* to *y42* (see Table VI).

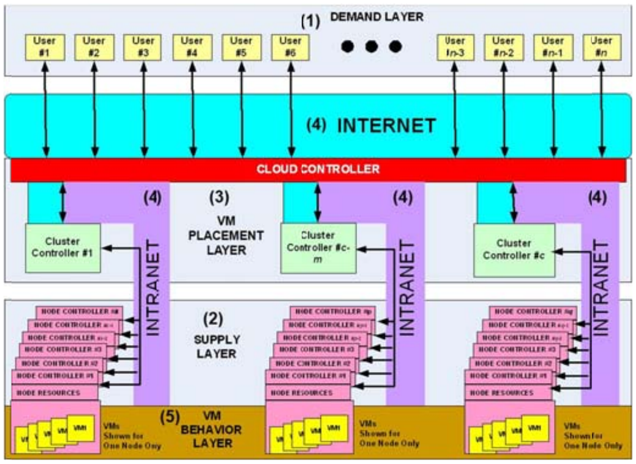

Figure 1. Schematic of Koala organization

A. *Demand Layer*

The demand layer consists of a variable number (*x1*) of users who, after random startup delay, each perform cyclically over a simulation run. During each cycle a user requests a minimum and maximum number of instances of one or more of the VM types shown in Table I. The VM types and quantities a user selects depend upon the user's type (see Table II), which is selected on each cycle with some probability (*x2*). After selecting a type, a user randomly chooses a minimum (uniform 1 to a max-min) and maximum (uniform max-min to a max-max) number of instances to request for each associated VM type. The user then issues a corresponding *RunInstances* request to the cloud controller, which may respond with an allocation of instances between the minimum and maximum for each requested VM type or with a NERA (not enough resources available) fault. A *full grant* denotes that a user was allocated the maximum requested instances of each VM type. A *partial grant* denotes that allocated VMs were below the maximum requested. If given VM instances, the user selects a holding time, Pareto distributed with variables specified by a parameter (*x3*). During the holding period, the user will first issue *DescribeInstances* requests to determine when all instances are running, and will subsequently randomly reboot, terminate and describe running instances. At the end of the holding period, the user issues a *TerminateInstances* request to stop any running instances. After terminating all instances, the user will wait an exponentially distributed time (mean 30 minutes) and then start a new cycle.

Since we believed differences in user persistence were not germane directly to our study, we assigned fixed means for each stochastic distribution controlling related behaviors. If a user receives a NERA instead of being allocated instances, then the user waits an exponentially distributed time (mean 15 minutes) before retrying the request. A user will retry a failed request over a random period (mean 4 hours) before resting for a random period (mean 16 hours). If a user request cannot be honored within a random number of rest periods (mean 4), then the user abandons the request and starts a new cycle.

TABLE I. Description of VM types simulated in Koala

VM Type	Virtual Cores #	Virtual Cores Speed (GHz)	Virtual Block Devices #	Virtual Block Devices Size (GB) of Each	# Virtual Network Interfaces	Memory (GB)	Instruct. Arch.	Price in $/Hour
M1 small	1	1.7	1	160	1	2	32-bit	0.12
M1 large	2	2	2	420	2	8	64-bit	0.34
M1 xlarge	4	2	4	420	2	16	64-bit	0.96
C1 medium	2	2.4	1	340	1	2	32-bit	0.17
C1 xlarge	8	2.4	4	420	2	8	64-bit	0.68
M2 xlarge	8	3	1	840	2	32	64-bit	1.00
M4 xlarge	8	3	2	850	2	64	64-bit	2.00

TABLE II. Description of selected simulated user types: processing users (PU), distributed modeling and simulation (MS) users, peer-to-peer (PS) users, Web service (WS) users, and data search (DS) users

User Type	VM Type(s)	Max-Min VMs	Max-Max VMs	User Type	VM Type(s)	Max-Min VMs	Max-Max VMs
PU1	M1 small	10	100	PS1	C1 medium	3	10
				PS2		10	50
PU3		100	500	PS3		50	100
PU5		500	1000	WS1	M1 large M2 xlarge C1 xlarge	1	3
PU2	M1 large	10	100	WS2	M1 large M2 xlarge C1 xlarge	3	9
PU4		100	500	WS3	M1 large M2 xlarge C1 xlarge	9	12
PU6		500	1000	DS1	M4 xlarge	10	100
MS1	M1 xlarge	10	100	DS2		100	500
MS3	M1 xlarge	100	500	DS3		500	1000

TABLE III. Description of selected platform types simulated in Koala

Platform Type	Physical Cores #	Physical Cores Speed (GHz)	Memory (GB)	# Physical Disks by Size 250 GB	500 GB	750 GB	1000 GB	# Network Interfaces	Instruct. Arch.
C8	2	2.4	32	0	3	0	0	1	64-bit
C14	4	3	64	0	4	0	3	2	64-bit
C18	8	3	128	0	0	4	3	4	64-bit
C22	16	3	256	0	0	0	7	4	64-bit

B. *Supply Layer*

The supply layer consists of a variable number (*x4*) of clusters that each manages a variable number (*x5*) of nodes. When visiting an Amazon EC2 data center, we noticed the supply of nodes was composed of a limited number of platform configurations. This motivated us to define a fixed set of possible platform configurations for nodes. Upon creation, each node manifests, with some probability (*x6*), one of the configurations shown in Table III. Nodes retain their established configurations for the duration of a simulation run. For a VM to be allocated to a node, available resources on the node must be sufficient for the requirements specified by the VM's type.

C. VM Placement Layer

Koala patterns VM placement after Eucalyptus procedures, which involve two decisions: (1) on which cluster should requested VMs be placed and (2) on which nodes within the cluster should VMs be placed. In this study, we compare three alternative criteria used by the cloud controller to choose a cluster and six alternative heuristics used by cluster controllers to choose nodes. Combining these criteria and heuristics creates the (3 x 6 =) 18 VM-placement algorithms we compare.

TABLE IV. Alternative Criteria for Choosing Cluster and Alternative Heuristics for Choosing Nodes

Criteria for Choosing a Cluster		Heuristics for Choosing Nodes	
Identifier	Criterion Name	Identifier	Heuristic Name
LLF	Least-Full First	FF	First Fit
		LF	Least-Full First
PAL	Percent Allocated	MF	Most-Full First
		NF	Next Fit
RAN	Random	RA	Random
		TP	Tag & Pack

In Eucalyptus, the cloud controller polls cluster controllers to find out which clusters can accommodate the VMs requested and then orders the qualified clusters using some criterion (Table IV – left side). The *Least-Full First* (LFF) criterion orders the set of qualified clusters from the least to most full, while the *Percent Allocated* (PAL) criterion orders the set by decreasing proportion of requested VMs that can be allocated. (Under both these criteria, ties were ordered by increasing cluster identifier.) The *Random* (RAN) criterion orders randomly the set of qualified clusters.

After ordering the qualified clusters, the cloud controller selects the first cluster in the set and asks that VMs be created. If VMs are created successfully, then the cloud controller returns the positive result to the user; otherwise, the cloud controller *reallocates* the VMs to the next cluster in the set. This process continues until VMs are created or until all clusters have been exhausted. If no clusters can create the VMs, then the user receives a NERA fault.

In Eucalyptus, the cluster controller allocates VMs to nodes using one of two heuristics: (1) *First-Fit* (FF) or (2) *Next-Fit* (NF). Koala simulates FF and NF, as well as four more heuristics (Table IV – right side) inspired by online bin-packing literature. FF searches the cluster's set of nodes, ordered by identifier from first to last, until a node is found that can accommodate a given VM type. NF remembers which node last received a VM and begins its search from the next node identifier. *Least-Full First* (LF) orders the set of qualified nodes (i.e., nodes on which the VM type will fit) from least to most full. *Most-Full First* (MF) orders the set of qualified nodes from most to least full. *Random* (RA) orders randomly the set of qualified nodes. *Tag & Pack* (TP) marks nodes by VM type, so that only VMs of a designated type can be placed on marked nodes. To place a VM, TP generates a set of qualified nodes marked with the appropriate VM type, and then adds to the end of that set any free (i.e., unmarked) qualified nodes. The VM is then placed on the first node in the set. If the selected node is unmarked, TP marks the node with the appropriate VM type. Whenever the last VM on a node is terminated, the node's marking is removed.

For any of the heuristics, if a selected node cannot accommodate a VM, then the node controller *reallocates* the VM to the next node in the set. This process continues until the VM is created or until all nodes have been exhausted. If the minimum requested number of VMs cannot be created, then the cloud controller receives a NERA fault.

D. Internet/Intranet Layer

Koala assigns the cloud controller, cluster controllers and users to *sites* (1000 here) randomly located at *x,y* coordinates on a grid (8000x8000 miles here) spanning a distance consistent with the globe. Before a simulation commences, cloud and cluster controllers are randomly placed on some number (1 here) of sites. Node controllers are placed on the same site as the related cluster controller. At the beginning of each user cycle, a user is assigned randomly to one of the sites (999 here) not occupied by cloud components. This arrangement divides message communications into two categories: (1) inter-site (Internet) and (2) intra-site (Intranet). Koala components communicate through simulated Web Services (WS) messages, which each comprise a uniformly distributed number (1 to 10 here) of packets. Individual packets are subjected to transmission delay (1 Gigabits per second here) and propagation delay. For inter-site messages, propagation delay depends on distance and simulated router hops, while propagation delay within sites varies randomly (mean 250 nanoseconds here). Individual packets are also subjected to a loss rate (10^{-12} here for intra-site packets). To simulate Internet congestion, the loss rate for inter-site packets varies uniformly within a range (10^{-3} to 10^{-8} here). Lost packets are retransmitted, but only for a maximum number (3 here) of attempts, after which the related WS message is declared undeliverable.

IV. Experiment Design

We compared 18 VM-placement algorithms by exposing each of them to the same 32 combinations of six parameters, where each parameter could be set to one of two values, as shown in Table V. The selected parameters were identified in an earlier sensitivity analysis (SA) [5] as the six most significant drivers of Koala behavior. Also guided by the SA, we selected two values for each parameter. We constructed 32 parameter combinations by adopting a two-level, 2^{6-1} orthogonal fractional factorial experiment design [2] that selects ½ of the 2^6 possible combinations, while ensuring balanced coverage. We then ran 32 simulations for each of the 18 VM-placement algorithms.

We compared the VM-placement algorithms with respect to 42 response variables, as shown in Table VI, grouped into six categories representing: (1) user experience, (2) cloud-wide resource utilization and load, (3) variance in cluster utilization and load, (4) number and types of VMs, (5) WS-message load and (6) revenue. The SA found that Koala has only eight significant behavioral dimensions, which can be represented using a subset of eight of the 42 responses, so we also compared the 18 algorithms along those dimensions (we identify a response variable to represent each dimension): (1) cloud-wide demand/supply ratio (*y3*), (2) cloud-wide resource usage (*y15*), (3) variance in cluster load (*y21*), (4) mix of VM types (*y31*), (5) number of VMs (*y29*), (6) user arrival rate (*y4*), (7) reallocation rate (*y7*) and (8) variance in cluster choice (*y28*).

TABLE V. Two Selected Values for each Selected Koala Parameter

Layer	Parameter	Parameter Name	Plus (1) Level	Minus (-1) Level
Demand Layer	*x1*	Number of users	2500	250
	x2	Probability of a user's type	PU1 = 0.20 PU2 = 0.20 PU3 = 0.10 PU4 = 0.10 MS1 = 0.10 MS3 = 0.01 PS1 = 0.10 PS2 = 0.01 WS1 = 0.15 WS2 = 0.07 WS3 = 0.03 DS1 = 0.10 DS2 = 0.01	PU1 = 1/6 PU2 = 1/6 MS1 = 1/6 PS1 = 1/6 WS1 = 1/6 DS1 = 1/6
	x3	Average (& shape) of user's holding time	8 hours (a = 1.2)	4 hours (a = 1.2)
Supply Layer	*x4*	Number of clusters	20	10
	x5	Number of nodes per cluster	1000	100
	x6	Probability of a node's platform configuration type	C22 = 1.0	C8 = 0.25 C14 = 0.25 C18 = 0.25 C22 = 0.25

Table VI. Koala Response Variables Selected for Comparison

Category	ID	Response Name	Definition
User	*y1*	User Request Rate	(Requests by All Users / # User Cycles)
	y2	NERA Rate	(NERAs / Requests by All Users)
	y3	Full Grant Rate	(Full Grants / (Full Grants + Partial Grants))
	y4	User Arrival Rate	(# User Cycles / Simulated Hours)
	y5	User Give-up Rate	(# Users that Gave Up / # User Cycles)
	y6	Grant Latency	Weighted Avg. Delay in Granting VMs to Users that Got VMs
	y40	User Success Rate	((Full Grants + Partial Grants)/# User Cycles)
	y41	Avg. Fraction VMs Obtained	(Allocated VMs/Requested VMs)
	y42	Avg. *RunInstance* Response Time	Weighted avg. for successful allocations
Cloud	*y7*	Reallocation Rate	(# Times Alternate Cluster Chosen / Requests Granted)
	y8	Full Grant Proportion	(Avg. Fraction Clusters Offering Full Grants)
	y9	NERA Proportion	(Avg. Fraction Clusters Reporting NERA)
	y10	vCore Utilization	(Avg. Fraction of Virtual Cores Used in Cloud)
	y11	Memory Utilization	(Avg. Fraction of Memory in Use in Cloud)
	y12	Disk Space Utilization	(Avg. Fraction of Disk Space in Use in Cloud)
	y13	pCore Load	(Avg. Virtual Cores Allocated / Physical Cores in Cloud)
	y14	Disk Count Load	(Avg. Virtual Disks Allocated / Physical Disks in Cloud)
	y15	NIC Court Load	(Avg. Virtual NICs Allocated / Physical NICs in Cloud)
Cluster	*y16*	vCore Utilization Variance	Avg. Variance in vCore Utilization across Clusters
	y17	Memory Utilization Variance	Avg. Variance in Memory Utilization across Clusters
	y18	Disk Space Utilization Variance	Avg. Variance in Disk Space Utilization across Clusters
	y19	pCore Load Variance	Avg. Variance in pCore Load across Clusters
	y20	Disk Count Variance	Avg. Variance in Disk Count Load across Clusters
	y21	NIC Court Variance	Avg. Variance in NIC Count Load across Clusters
	y22	Node Reallocation Rate	(# Times Alternate Node Chosen / VMs Allocated)
	y23	Cluster NERA Rate	(# NERAs / # Responses Avg. across Clusters)
	y24	Cluster Full-Grant Rate	(# Full Grants / # Responses Avg. across Clusters)
	y25	Allocation Rate	(Times Cluster chosen / Cluster offered Avg. across Clusters)
	y26	Standard Deviation-NERA	Stand. Dev. in Avg. NERA Rate across Clusters
	y27	Standard Deviation-Full-Grant	Stand. Dev. in Avg. Full-Grant Rate across Clusters
	y28	Standard Deviation-Allocation Rate	Stand. Dev. in Allocation Rate across Clusters
VMs	*y29*	Current Instances	Avg. # VM Instances Extant in Cloud
	y30	M1small Instances	Fraction of Current Instances that are M1 small VMs
	y31	M1large Instances	Fraction of Current Instances that are M1 large VMs
	y32	M1xlarge Instances	Fraction of Current Instances that are M1 xlarge VMs
	y33	C1medium Instances	Fraction of Current Instances that are C1 medium VMs
	y34	C1xlarge Instances	Fraction of Current Instances that are C1 xlarge VMs
	y35	M2xlarge Instances	Fraction of Current Instances that are M2 xlarge VMs
	y36	M4xlarge Instances	Fraction of Current Instances that are M4 xlarge VMs
Internet/ Intranet	*y37*	WS Message Rate	Avg. # WS Messages Send Per Simulated Hour
	y38	Intra-Site Messages	(# WS Messages Sent with Sites / # WS Messages Sent)
Revenue	*y39*	Aggregate Revenue in $/Hour	Calculated from *y29* through *y36* & VM prices

V. RESULTS

The simulation runs generated a multivariate dataset containing 576 rows and 42 columns (one per response). Each row was tagged with the cluster-choice criterion, node-selection heuristic and parameter combination that led to the responses. We applied one-way analysis of variance (ANOVA) to test for differences among groups of responses. For each response, for example, we computed an F-test statistic to measure the ratio of variability within data points grouped by cluster-choice criterion to variability between the groups:

$$F = \frac{df_2}{df_1} \cdot \frac{\sum_{i=1}^{3}\sum_{j=1}^{6}\sum_{k=1}^{32}(x_{ijk} - \bar{\bar{x}})^2}{\sum_{i=1}^{3}\sum_{j=1}^{6}\sum_{k=1}^{32}(x_{ijk} - \overline{x_k})^2} \quad (1)$$

where i is the cluster-choice criterion, j is the node-selection heuristic, k is the parameter combination, and df_n represents appropriate F degrees of freedom. We then computed the corresponding cumulative distribution function (cdf) value for that F-test statistic to reflect the likelihood that the groups were different.

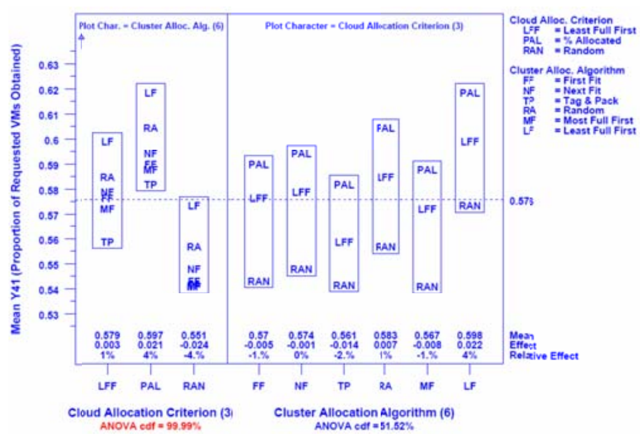

Figure 2. Plot of ANOVA Results for Response Variable *y41*- Average Fraction of VMs Obtained (red denotes significant difference among groups)

In the left hand part of Fig. 2, we show the results of an ANOVA for the average fraction of VMs obtained (*y41*). The dashed horizontal line denotes the grand mean of all data points, which is 0.576. The figure includes three blocks (labeled on the x axis), one for each cluster-choice criterion. Each block contains six labeled points, each representing the mean value of the 32 data points when the labeled node-selection heuristic was paired with the designated cluster-choice criterion. Below each block are three numbers: (1) the local mean value of the 192 data points for a given cluster-choice criterion, (2) the difference between that local mean value and the grand mean, and (3) the percentage difference. Below the x axis we report the F cdf value, highlighting in red any value ≥ 95%.

In the example shown, we observed an F-test statistic value that would – if the three groups were equal – occur only (100 – 99.99 =) .01% of the time. This leads us to conclude that cluster-choice criterion yields statistically significant differences in the average fraction of VMs obtained. In the right hand part of Fig. 2, we reverse the ANOVA analysis to test for differences among six groups of data points when the node-selection heuristic is varied. In this case, the node-selection heuristic did not lead to significant differences.

Table VII gives two summaries of ANOVA results for each of the 42 responses, one response per row. The first summary (col. 4) reports differences caused by cluster-choice criterion and the second summary (col. 5) reports differences caused by node-selection heuristic. Significant differences are highlighted in red.

Table VIII reports the mean value for each response (42 rows) given each cluster-choice criterion (cols. 3-5). Each mean averages 192 observations (32 parameter combinations by six node-selection heuristics). The means are highlighted in red for responses where ANOVA found a significant difference caused by cluster-choice criterion. Table IX reports the mean value for responses given each node-selection heuristic. Each mean averages 96 observations (32 parameter combinations by three cluster-choice criteria).

TABLE VII. Summary of 84 ANOVA Tests: Each Row Represents One of 42 Responses; Column 4 Reports Differences Attributable to Cluster-Choice Criterion and Column 5 Reports Differences Attributable to Node-Selection Heuristic – cells highlighted in red identify significant differences

Category	ID	Response Name	ANOVA Cdf Cloud Crit (3)	ANOVA Cdf Cluster Alg (6)
User	*y1*	User Request Rate	99.96	62.19
	y2	NERA Rate	100	22.33
	y3	Full Grant Rate	100	2.75
	y4	User Arrival Rate	99.87	77.15
	y5	User Give-up Rate	94.63	98.6
	y6	Grant Latency	98.01	96.11
	y40	User Success Rate	95.86	98.02
	y41	Avg. Fraction VMs Obtained	99.99	51.52
	y42	Avg. *RunInstance* Response Time	37.35	97.49
Cloud	*y7*	Reallocation Rate	99.99	9.5
	y8	Full Grant Proportion	100	0.02
	y9	NERA Proportion	100	0.4
	y10	vCore Utilization	67.85	99.81
	y11	Memory Utilization	98.97	91.47
	y12	Disk Space Utilization	97.29	96.27
	y13	pCore Load	67.85	99.81
	y14	Disk Count Load	96.76	97.56
	y15	NIC Count Load	99.78	79.49
Cluster	*y16*	vCore Utilization Variance	100	1.28
	y17	Memory Utilization Variance	100	0.09
	y18	Disk Space Utilization Variance	100	0.14
	y19	pCore Load Variance	100	1.28
	y20	Disk Count Variance	100	0.42
	y21	NIC Count Variance	100	1.02
	y22	Node Reallocation Rate	100	6.09
	y23	Cluster NERA Rate	100	0.19
	y24	Cluster Full-Grant Rate	100	0.06
	y25	Allocation Rate	99.88	77.64
	y26	Standard Deviation-NERA	63.92	61.08
	y27	Standard Deviation-Full-Grant	99.73	30.95
	y28	Standard Deviation-Allocation Rate	100	0.02
VMs	*y29*	Current Instances	99.98	50.54
	y30	M1small Instances	99.99	35.85
	y31	M1large Instances	60.58	99.02
	y32	M1xlarge Instances	99.83	77.1
	y33	C1medium Instances	99.97	27.57
	y34	C1xlarge Instances	82.1	99.89
	y35	M2xlarge Instances	74.62	99.97
	y36	M4xlarge Instances	99.95	66.03
Internet/ Intranet	*y37*	WS Message Rate	99.7	83.74
	y38	Intra-Site Messages	89	99.05
Revenue	*y39*	Aggregate Revenue in $/Hour	99.99	44.51

VI. DISCUSSION

The ANOVA tests on the experiment results demonstrate clearly that cluster-choice criterion exhibits a much stronger effect on overall cloud behavior than does node-selection heuristic. Cluster-choice criterion caused significant differences in 79% (33/42) of the responses, and also led to significant differences in 100% (8/8) of responses chosen to represent the eight behavioral dimensions of Koala. The node-selection heuristic significantly influenced only 29% (12/42) of the responses, and led to significant differences in only 12.5% (1/8) of Koala's behavioral dimensions.

Examining results in Table VIII shows that the percent-allocated (PAL) cluster-choice criterion leads to higher average loads (*y13*-*y15*) and utilizations (*y10*-*y12*) in the cloud. This occurs because admitted users are placed on clusters that can accommodate the highest fraction of requested instances (*y41*), which also leads to a higher number of running instances (*y29*), allowing the cloud to generate more revenue per hour (*y39*). While PAL generates only $384/hour more revenue than least-full first (LLF), when aggregated over a year this difference means that PAL generates about $3.4M more than LLF.

TABLE VIII. Mean for Each Response under Each of Three Cluster-Choice Criteria – cells highlighted in red per ANOVA from Table VII

Category	ID	LLF	PAL	RAN
User	*y1*	7.461	8.386	7.696
	y2	0.444	0.506	0.450
	y3	0.624	0.574	0.514
	y4	37324	35878	37170
	y5	0.066	0.074	0.067
	y6	9044	10488	9526
	y40	0.925	0.915	0.923
	y41	0.579	0.597	0.551
	y42	0.278	0.277	0.278
Cloud	*y7*	0.000052	0.000084	0.000057
	y8	0.438	0.332	0.389
	y9	0.481	0.587	0.537
	y10	0.774	0.791	0.783
	y11	0.188	0.197	0.199
	y12	0.413	0.428	0.418
	y13	0.774	0.791	0.783
	y14	0.964	0.997	0.948
	y15	1.591	1.645	1.554
Cluster	*y16*	0.0017	0.019	0.0071
	y17	0.0009	0.0034	0.0015
	y18	0.0022	0.0086	0.0038
	y19	0.0017	0.019	0.0071
	y20	0.018	0.052	0.024
	y21	0.045	0.127	0.052
	y22	0.00015	0.00015	0.00008
	y23	0.507	0.606	0.562
	y24	0.421	0.323	0.375
	y25	0.19	0.232	0.232
	y26	0.01	0.01	0.011
	y27	0.008	0.011	0.015
	y28	0.034	0.058	0.02
VMs	*y29*	21808	22139	20365
	y30	0.355	0.354	0.333
	y31	0.308	0.311	0.307
	y32	0.138	0.142	0.151
	y33	0.057	0.053	0.052
	y34	0.025	0.022	0.025
	y35	0.026	0.023	0.026
	y36	0.091	0.096	0.106
Internet/ Intranet	*y37*	60867	62677	60841
	y38	0.977	0.977	0.977
Revenue	*y39*	11322	11706	11624

On the other hand, these factors have an overall negative effect on the general population of users, who receive more NERA responses (*y2*) and so have to retry more requests (*y1*) before their VMs can be placed, which leads to greater than 20 minutes more waiting time (*y6*). The increase in requests also leads to an increase in the rate of WS messages (*y37*). PAL, then, serves fewer users (*y4*) but gives each served user a larger proportion of their requested VMs (*y41*). Dedicating more resources to fewer users also leads to significant increase in variance in resource load (*y19*-*y21*) and utilization (*y16*-*y18*) among clusters in the cloud.

Within the cloud, PAL also leads to increased conflicts among parallel allocation requests, which results in more reallocations of clusters (*y7*) and a lower proportion of full grants (*y8*). While this may seem counterintuitive, accommodating a larger proportion of each user's requested VMs means that fewer users can obtain all the VMs requested, as compared with the LFF cluster-choice criterion.

TABLE IX. Mean for Each Response under Each of Six Node-Selection Heuristics – cells highlighted in red per ANOVA from Table VII

Category	ID	FF	LF	MF	NF	TP	RA
User	y1	7.643	8.450	7.692	7.710	7.871	7.718
	y2	0.460	0.493	0.458	0.462	0.455	0.470
	y3	0.566	0.593	0.563	0.57	0.555	0.577
	y4	37138	35624	37188	36938	37051	36807
	y5	0.065	0.080	0.065	0.067	0.067	0.069
	y6	10130	8636	10439	9643	10420	8848
	y40	0.925	0.908	0.925	0.923	0.922	0.921
	y41	0.57	0.598	0.567	0.574	0.561	0.583
	y42	0.278	0.276	0.278	0.279	0.277	0.278
Cloud	y7	0.000063	0.000064	0.000068	0.000073	0.000055	0.000063
	y8	0.387	0.387	0.378	0.389	0.385	0.39
	y9	0.529	0.55	0.536	0.528	0.536	0.532
	y10	0.789	0.761	0.812	0.786	0.764	0.78
	y11	0.198	0.188	0.204	0.196	0.191	0.193
	y12	0.419	0.428	0.424	0.421	0.402	0.424
	y13	0.789	0.761	0.812	0.786	0.764	0.78
	y14	0.958	1.013	0.958	0.97	0.928	0.99
	y15	1.58	1.639	1.597	1.592	1.542	1.631
Cluster	y16	0.0085	0.008	0.0127	0.0097	0.008	0.008
	y17	0.0019	0.0020	0.0022	0.0019	0.0019	0.0017
	y18	0.0045	0.0054	0.0053	0.0050	0.0046	0.0045
	y19	0.0085	0.0089	0.0127	0.0097	0.0080	0.0080
	y20	0.029	0.036	0.032	0.032	0.029	0.029
	y21	0.067	0.088	0.080	0.074	0.065	0.073
	y22	0.00013	0.00012	0.00013	0.00014	0.00011	0.00012
	y23	0.555	0.569	0.562	0.552	0.558	0.553
	y24	0.373	0.375	0.364	0.376	0.373	0.378
	y25	0.228	0.192	0.237	0.216	0.232	0.201
	y26	0.011	0.009	0.013	0.010	0.010	0.009
	y27	0.012	0.010	0.015	0.011	0.012	0.010
	y28	0.037	0.040	0.037	0.037	0.035	0.038
VMs	y29	21237	22244	21020	21409	20824	21888
	y30	0.344	0.356	0.342	0.348	0.341	0.352
	y31	0.306	0.315	0.304	0.305	0.311	0.312
	y32	0.144	0.149	0.145	0.147	0.135	0.142
	y33	0.054	0.053	0.053	0.053	0.056	0.054
	y34	0.025	0.018	0.026	0.024	0.027	0.022
	y35	0.027	0.019	0.028	0.026	0.029	0.023
	y36	0.100	0.090	0.103	0.097	0.101	0.095
Internet/ Intranet	y37	61018	63016	61223	61156	60571	61785
	y38	0.977	0.977	0.977	0.977	0.976	0.977
Revenue	y39	11603	11529	11683	11587	11362	11541

When compared with PAL, LLF serves more users (*y4*) and dedicates fewer resources to each user, which results in more full grants (*y3*), and also distributes load more evenly (*y16*-*y21*) among clusters. RAN, on the other hand, leads to fewer full grants (*y3*) because the cluster with the most available space is not always chosen. RAN also leads to lower node reallocation rate (*y22*) because there is a smaller chance that overlapping allocation requests will be assigned to the same cluster.

Examining results in Table IX reveals a few effects attributable to node-selection heuristic. LF and TP lead to lower cloud-wide virtual core utilization (*y10*) because these heuristics more often choose empty nodes on which to place VMs. By choosing empty nodes more often, LF tends to squeeze out some larger VM types (*y34* and *y35*) associated with Web service users. This factor also leads LF to yield lower user success rate (*y40*) and higher give-up rate (*y5*). By tagging nodes for particular types of VMs, TP avoids squeezing out larger VM types. TP also yields lower disk space utilization (*y12*) and disk count load (*y14*). Finally, LF and RA lead to lower grant latencies (*y6*), reflecting the fact that these heuristics allow users who successfully acquire VMs to do so with an average of about one fewer retry than the other heuristics. This occurs primarily when combined with the LLF and RAN cluster-choice criteria. When combined with PAL, LF and RA lead to grant latencies closer to the grand average.

Reviewing individual ANOVA plots (such as Fig. 2) for all 42 responses allowed us to identify particularly noteworthy combinations of cluster-choice criterion and node-selection heuristic. For example, the PAL-LF combination led to the highest user request rate (*y1*), NERA rate (*y2*), give-up rate (*y5*) and fraction of VMs obtained (*y41*), while also yielding the lowest user success rate (*y40*). At the same time, PAL-LF led to highest disk space utilization (*y12*) and disk (*y14*) and network-interface controller (*y15*) loads. Combining PAL with MF yielded highest variance among clusters for virtual core (*y16*), memory (*y17*) and disk space (*y18*) utilizations.

The LLF-LF combination led to lowest grant latency (*y6*) and virtual core (*y10*) and memory (*y11*) utilizations. Combining LLF and TP yielded lowest disk space utilization (*y12*) and the least revenue per hour (*y39*).

The RAN-TP combination gave lowest disk count load (*y14*). RAN-MF yielded the lowest faction of M1small VMs (*y30*), while RAN-LF combined to give the highest fraction of M1xlarge VMs (*y32*).

VII. Conclusions and Future Work

We have developed an objective method to compare resource-allocation algorithms in simulations of large distributed systems. Previously, we applied our method to compare proposed congestion-control algorithms for the Internet. In this paper, we demonstrated steps three and four in our method, using results from an earlier sensitivity analysis to construct parameter combinations under which we compared macroscopic behavior of algorithms for initially placing VMs on nodes in on-demand infrastructure clouds. While we restricted our comparison to 18 selected algorithms, the approach we use should be applicable to compare any set of competing algorithms.

We generated insights regarding two-level VM-placement algorithms, showing that choice of cluster has larger influence, than selection of nodes, on macroscopic behavior in an infrastructure cloud. We identified some tradeoffs among cluster-choice criteria. We provided evidence showing that, on average, different algorithms for initial VM placement in on-demand infrastructure clouds yield small quantitative differences in many measured responses. On the other hand, we showed that selection of the criterion for choosing a cluster can lead to very large difference in provider revenue, when aggregated over time.

We envision future work along three directions. First, we intend to compare these 18 VM-placement algorithms under situations where various failures inhibit cloud operation. Second, we will extend our comparison to include additional VM-placement algorithms, which have become the subject of much recent research. Finally, we will also consider applying our method to compare various proposed algorithms for moving VMs among cloud assets, whether to reduce cost, enhance user performance, or both.

Figure & Table Enlargements

Enlargements of all figures and tables in this paper may be downloaded from http://www.nist.gov/itl/antd/upload/LargerTablesPaper36.pdf

References

[1] V. Paxson and S. Floyd. "Why we don't know how to simulate the Internet," Proceedings of the 1997 Winter Simulation Conference, ed. S. Andradottir, K. J. Healy, D. H. Withers, and B. L. Nelson, pp. 1037-1044.

[2] G. E. Box, J. S. Hunter, and W. G. Hunter, Statistics for Experimenters, 2nd ed., Wiley, 2005, 639 pages.

[3] K. Mills, J. Filliben, D. Cho and E. Schwartz, "K. Mills, J. Filliben, D-Y. Cho and E. Schwartz, "Predicting Macroscopic Dynamics in Large Distributed Systems", Proceedings of ASME PVP 2011, Baltimore, MD, July 17-22, 2011.

[4] K. Mills, J. Filliben, D. Cho, E. Schwartz and D. Genin, Study of Proposed Internet Congestion Control Algorithms, NIST Special Publication 500-282, May 2010, 534 pages.

[5] K. Mills, J. Filliben and C. Dabrowski, "An Efficient Sensitivity Analysis Method for Large Cloud Simulations", Proceedings of the 4th International Cloud Computing Conference, IEEE, Washington, D.C., July 5-9, 2011.

[6] D. Nurmi, et al., "The Eucalyptus Open-Source Cloud-Computing System", Proceedings of the 9th IEEE/ACM International Symposium on Cluster Computing and the Grid, May 18-21, 2009, pp. 124-131.

[7] S. S. Seiden, R. V. Stee, and L. Epstein, "New Bounds for Variable-Sized Online Bin Packing", SIAM Journal on Computing, Vol. 32, No. 2, 2003, pp. 455-469.

[8] E. Coffman, M. Garey, and D. Johnson. Approximation Algorithms for Bin Packing: A Survey. PWS Publishers, Boston, 1997.

[9] S. Shang, Y. Wu, J. Jiang and W. Zheng, "An Intelligent Capacity Planning Model for Cloud Market", Journal of Internet Services and Information Security, **1:1**, 37-45.

[10] I. Fujiwara, K. Aida, and I. Ono, "Applying Double-sided Combinational Auctions to Resource Allocation in Cloud Computing", Proceedings of the 10th Annual International Symposium on Applications and the Internet, IEEE, July 19-23, 2010, pp. 7-14.

[11] Amazon Elastic Compute Cloud (Amazon EC2) http://aws.amazon.com/ec2/, 2010.

[12] A. Andrzejak, D. Kondo, and S. Yi, "Decision Model for Cloud Computing under SLA Constraints," INRIA Technical Report-004/4849, Version 1, April 21, 2010.

[13] M. Cardosa, A. Singh, H. Pucha and A. Chandra, "Exploiting Spatio-Temporal Tradeoffs for Energy Efficient MapReduce in the Cloud", Dept. of Comp. Sci. and Eng., University of Minnesota, TR-10-008. April 7, 2010, 18 pp.

[14] X. Meng, V. Pappas and L. Zhang, "Improving the Scalability of Data Center Networks with Traffic-aware Virtual Machine Placement", Proceedings of IEEE 2010 INFOCOM, Mar. 14-19, 2010.

[15] B. Chandrasekaran, R. Purush, B. Douglas and D. Schmidt, "Virtualization Management Using Microsoft System Center and Dell OpenManage", Dell Power Solutions, Aug. 2007, 40-44.

[16] J. Xu and J. Fortes, "Multi-objective Virtual Machine Placement in Virtualized Data Center Environments", Proceedings of the 2010 IEEE/ACM Conference on Green Computing and Communications, 179-188.

[17] M. Mishra and A. Sahoo, "On Theory of VM Placement: Anomalies in Existing Methodologies and Their Mitigation Using a Novel Vector Based Approach", Proceedings of the 4th International Cloud Computing Conference, IEEE, Washington, D.C., July 5-9, 2011.

[18] U. Bellur, C. Rao and M. Kumar, "Optimal Placement Algorithms for Virtual Machines", Proceedings of CoRR. 2010.

[19] J. Fontan, "Session 6: Advance Usage of OpenNebula", OpenNebual Technology Days, July 20-21, 2010.

[20] M. Sindelar, R. Sitaraman and P. Shenoy, "Sharing-Aware Algorithms for Virtual Machine Colocation", Proceedings of the 23rd Annual ACM Symposium on Parallelism in Algorithms and Architectures, June 4-6, 2011.

[21] H. Van and F. Tran, "Autonomic resource management for service host platforms" Proceedings of Workshop on Software Engineering Challenges in Cloud Computing, Vancouver, Canada, April 2010.

[22] C. Mark, D. Niyato and T. Chen-Khong, "Evolutionary Optimal Virtual Machine Placement and Demand Forecaster for Cloud Computing", Proceedings of the 2011 IEE International Conference on Advanced Information Networking and Applications, 348-355.

[23] F. Machida, M. Kawato and Y. Maeno, "Redundant Virtual Machine Placement for Fault-tolerant Consolidated Server Clusters", Proceedings of the 12th IEEE/IFIP Network Operations and Management Symposium, Osaka, Japan, 2010, 32-39.

[24] N. Bobroff, A. Kochut and K. Beaty, "Dynamic Placement of Virtual Machines for Managing SLA Violations", Proceedings of the 10th IFIP/IEEE Symposium on Integrated Network Management, 2007,119-128.

[25] M. Chen, H. Zhang, Y.Y. Su, X. Wang, G. Jiang and K. Yoshihra, "Effective VM Sizing in Virtualized Data Centers", Proceedings of the 12th IFIP/IEEE Symposium on Integrated Network Management, 2011, Dublin, Ireland.

[26] S. Das, M. Kagan and D. Crupnicoff, "Faster and Efficient VM Migrations for Improving SLA and ROI in Cloud Infrastructures", DC CAVES 2010.

[27] A. Verma, P. Ahuja and A. Neogi, "pMapper: Power and Migration Cost Aware Application Placement in Virtualized Systems" Proceedings of the 9th ACM/IFIP/USENIX International Conference on Middleware, 243-264.

[28] C. Isci, J. Hanson, I. Whalley, M. Steinder and J. Kephart, "Runtime Demand Estimation for Effective Dynamic Resource Management", Proceedings of the 12th IEEE/IFIP Network Operations and Management Symposium, Osaka, Japan, 2010, 381-388.

[29] S. Lee, R. Panigrahy, V. Prabhakaran, V. Ramasubrahmanian, K. Talwar, L. Uyeda and U. Wieder, "Validating Heuristics for Virtual Machine Consolidation", Microsoft Research, MSR-TR-2011-9, January 2011, 14 pp.

[30] B. Malet and P. Pietzuch, "Resource Allocation across Multiple Cloud Data Centres", Proceedings of the 8th International Workshop on Middleware for Grids, Clouds and e-Science, ACM, 2010.

[31] C. Hyser, B. McKee, R. Gardner and B. Watson, "Autonomic Virtual Machine Placement in the Data Center", HP Laboratories, HPL-2007-189, 11 pp.

[32] Amazon Elastic Compute Cloud (Amazon EC2) http://aws.amazon.com/ec2/, 2010.

[33] Amazon Elastic Compute Cloud API Reference API Version 2009-08-15.

[34] Amazon EC2 Instance Types http://aws.amazon.com/ec2/instance-types/, 2010.

[35] F. Curbera, et al. "Unraveling the Web services web: an introduction to SOAP, WSDL, and UDDI", Internet Computing, IEEE, March/April, 2002, pp. 86-93.