# An Efficient Sensitivity Analysis Method for Large Cloud Simulations

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

***Abstract*—Simulations of large distributed systems, such as infrastructure clouds, usually entail a large space of parameters and responses that prove impractical to explore. To reduce the space of inputs, experimenters, guided by domain knowledge and ad hoc methods, typically select a subset of parameters and values to simulate. Similarly, experimenters typically use ad hoc methods to reduce the number of responses to analyze. Such ad hoc methods can result in experiment designs that miss significant parameter combinations and important responses, or that overweight selected parameters and responses. When this occurs, the experiment results and subsequent analyses can be misleading. In this paper, we apply an efficient sensitivity analysis method to demonstrate how relevant parameter combinations and behaviors can be identified for an infrastructure Cloud simulator that is intended to compare resource allocation algorithms. Researchers can use the techniques we demonstrate here to design experiments for large Cloud simulations, leading to improved quality in derived research results and findings.**

***Keywords- cloud computing;modeling; resource allocation; sensitivity analysis; simulation***

## I. INTRODUCTION

Paxson and Floyd [1] describe many difficult problems that impede 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 model behaviors. The same can be said for most simulations of large distributed systems, such as computational grids and clouds. In this paper, we demonstrate an efficient sensitivity analysis method that can be used to identify the most significant parameters influencing model behavior. This allows experimenters to explore a reduced set of parameter combinations by varying those parameters that contribute most to differences in model response. Our sensitivity analysis method also combines correlation analysis and clustering to identify significant model behaviors. We apply our method to Koala, an infrastructure Cloud simulator intended to investigate a range of heuristic algorithms for allocating virtual machines (VMs) to platforms. We previously applied our method to a network simulator intended to compare proposed congestion control algorithms for the Internet [2]. Application to both Internet and Cloud models shows that our method is generally applicable to experiments involving simulations of large distributed systems.

Using Koala simulations as an example, Fig. 1 locates our sensitivity analysis method within the larger context of parameter reduction techniques we adopt. Koala begins as a model with 82 parameters, which, assuming each parameter can take on $2^{32}$ values, defines a parameter space larger than atoms in the visible universe. An experimenter then groups parameters that appear to represent different aspects of a single input, reducing the parameter space by 59. For example, Koala's 21 user types can be condensed to form a single distribution of user types, reducing the parameter space by 20. Then, using domain knowledge, an experimenter identifies parameters that do not appear germane to the intended investigation, replacing 12 more parameters with fixed values. For example, adopting assumptions that all users are similarly persistent, allows us to fix average values for four user parameters. Using such model reduction techniques, an experimenter reduced the Koala parameter space to about $10^{105}$, still too large to compute feasibly.

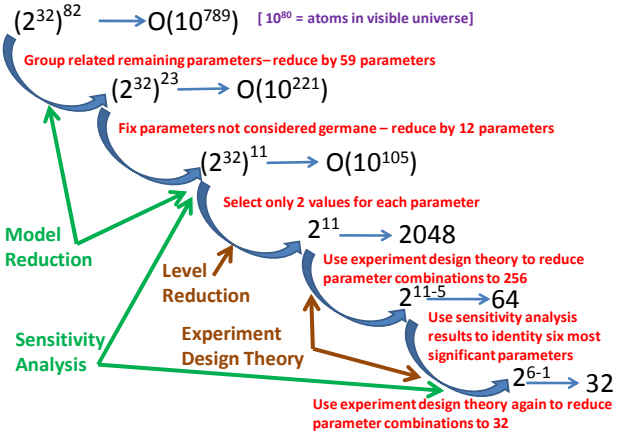



Figure 1. Sensitivity Analysis in the Context of Parameter Reduction

At this stage, our sensitivity analysis method begins. First, an experimenter selects only two values for each of the remaining 11 Koala parameters. Restricting parameters to only two values has obvious limitations: only a small number of parameter values are explored and extrapolating from the results assumes a model behaves monotonically in the range

between chosen values. On the other hand, adopting a two-level design provides some advantages [3]: (1) requires few runs per parameter, (2) facilitates interpretation of response data, (3) identifies promising directions for future experiments (and may be augmented with thorough local explorations), (4) fits naturally into a sequential strategy, which supports the scientific method and (5) forms the basis for further reduction in parameter combinations through use of fractional factorial designs (as demonstrated by the last step in Fig. 1). As we show later, an experimenter can select a handful of different two-value settings, which increases the range of robustness of conclusions associated with the simulation results. In addition, stochastic repetitions of two-value parameter combinations can be simulated efficiently.

The next step in our sensitivity analysis method applies experiment design theory [3] to select a balanced and orthogonal subset of the $2^{11}$ parameter combinations, which allows us to explore the search space in a principled fashion, as opposed to the ad hoc factor-at-a-time approaches [4] typically adopted by experimenters. Below, we explain the benefits of using orthogonal fractional factorial experiment designs. In the end, we identify a subset of ($2^{11-5}$ =) 64 parameter combinations to simulate. Below, we also demonstrate techniques that enable us to reduce 40 model responses, selected by experimenters, to as few as eight behavioral dimensions. This enables us to compress 32 redundant responses.

The remainder of this paper is organized as follows. In Sec. II, we describe our model and identify the parameters varied in our sensitivity analysis. In Sec. III, we describe our experiment design, aimed to identify significant model behaviors and the parameters that influence those behaviors. Later experiments will greatly benefit from these findings because we will understand what parameters to vary and what behaviors to measure when comparing a wide range of resource allocation algorithms. In Sec. IV we present our experiment results and related analysis methods. In Sec. V, we discuss our findings that Koala exhibits eight significant behaviors that are influenced mainly by six parameters. We also outline the implications of our findings for subsequent experiments intended to evaluate resource allocation heuristics. In Sec. VI, we describe related work and discuss how our methods may be applicable to other Cloud simulators and experiments. We close in Sec. VII with our conclusions and future work.

## II. MODEL

We demonstrate our approach using Koala, a discrete-event simulator inspired by the Amazon Elastic Compute Cloud (EC2) [5]. Using published information describing the EC2 application programming interface (API) [6] and available virtual machine (VM) types [7], Koala models essential features of the interface between users and EC2. Since we intended to study resource allocation algorithms, Koala needed to model only four EC2 commands: *RunInstances*, *DescribeInstances*, *Reboot Instances* and *TerminateInstances*. On the other hand, no public information was available about the internal structure and operation of EC2. Lacking such details, the internal structure of Koala is based instead on the Eucalyptus (v1.6) open-source Cloud software [8]. 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 [9], which are also simulated. In constructing Kola, we modified the design of Eucalyptus in three ways. First, we extended the Eucalyptus *RunInstances* command to allow multiple VM types within a single request, which we inferred is possible in EC2. Second, we avoided centralization of node information at the cloud controller, permitting Koala to simulate clouds up to $O(10^5)$ nodes. Third, we allowed resource allocation 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, we added an optional sub-model based on analytical equations representing VM behavior with or without tasks.

Koala is organized as five layers (see Fig. 2): (1) demand layer, (2) supply layer, (3) resource allocation 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 *x11* (see Table IV) and outputs as *y1* to *y40* (see Table V).

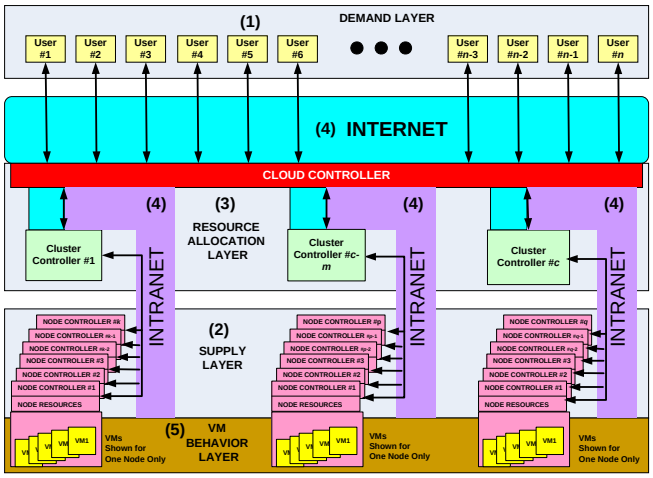



Figure 2. Schematic of Koala organization

### A. Demand Layer

The demand layer consists of a variable number (*x2*) of users who, after a 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 (*x3*). 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 parameter (*x4*). 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 will issue 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 the study of resource allocation algorithms, 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 Block Devices | | # Virtual Network Interfaces | Memory (GB) | Instruct. Arch. |
|---|---|---|---|---|---|---|---|
| | # | Speed (GHz) | # | Size (GB) of Each | | | |
| M1 small | 1 | 1.7 | 1 | 160 | 1 | 2 | 32-bit |
| M1 large | 2 | 2 | 2 | 420 | 2 | 8 | 64-bit |
| M1 xlarge | 4 | 2 | 4 | 420 | 2 | 16 | 64-bit |
| C1 medium | 2 | 2.4 | 1 | 340 | 1 | 2 | 32-bit |
| C1 xlarge | 8 | 2.4 | 4 | 420 | 2 | 8 | 64-bit |
| M2 xlarge | 8 | 3 | 1 | 840 | 2 | 32 | 64-bit |
| M4 xlarge | 8 | 3 | 2 | 850 | 2 | 64 | 64-bit |

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 |

## B. Supply Layer

The supply layer consists of a variable number (*x5*) of clusters that each manages a variable number (*x6*) 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 observation motivated us to define a fixed set of possible platform configurations for nodes. Upon creation, each node manifests, with some probability (*x7*), one of the configurations shown in Table III. Nodes retain their established configurations for the duration of a simulation run. For an instance to be allocated to a node, available resources on the node must be sufficient for the requirements specified by the instance's VM type.

TABLE III. Description of selected platform types simulated in Koala

| Platform Type | Physical Cores | | Memory (GB) | # Physical Disks by Size | | | | # Network Interfaces | Instruct. Arch. |
|---|---|---|---|---|---|---|---|---|---|
| | # | Speed (GHz) | | 250 GB | 500 GB | 750 GB | 1000 GB | | |
| C2 | 1 | 1.7 | 16 | 3 | 0 | 0 | 0 | 1 | 32-bit |
| C4 | 1 | 2 | 16 | 3 | 0 | 0 | 0 | 1 | 32-bit |
| C6 | 2 | 2.4 | 16 | 0 | 3 | 0 | 0 | 1 | 32-bit |
| C8 | 2 | 2.4 | 32 | 0 | 3 | 0 | 0 | 1 | 64-bit |
| C10 | 4 | 2.4 | 32 | 0 | 4 | 0 | 3 | 1 | 64-bit |
| C12 | 4 | 2 | 64 | 0 | 4 | 0 | 3 | 2 | 64-bit |
| C14 | 4 | 3 | 64 | 0 | 4 | 0 | 3 | 2 | 64-bit |
| C16 | 8 | 3 | 64 | 0 | 0 | 4 | 3 | 2 | 64-bit |
| C18 | 8 | 3 | 128 | 0 | 0 | 4 | 3 | 4 | 64-bit |
| C20 | 16 | 3 | 128 | 0 | 0 | 0 | 7 | 4 | 64-bit |
| C22 | 16 | 3 | 256 | 0 | 0 | 0 | 7 | 4 | 64-bit |

## C. Resource Allocation Layer

Koala patterns resource allocation after Eucalyptus procedures, which involve two decisions: (1) on which cluster should the requested VMs be allocated and (2) on which nodes within the cluster should VMs be allocated. Allocating all VMs in a single request to the same cluster makes good sense because inter-VM communications would be local to a single cluster. We included a few resource allocation algorithms in our sensitivity analysis simply to obtain an early indication of their potential influence on cloud behavior. We chose to simulate the resource allocation algorithms implemented by Eucalyptus.

At the cluster level, Eucalyptus allocates VMs to nodes using one of two algorithms: (1) *first-fit* or (2) *next-fit*. First-fit simply searches the nodes by identifier from first to last until a node is found that can accommodate a given VM type. Next-fit remembers which node last received a VM and begins its search from the next node identifier. If the selected node cannot accommodate the VM, then the node controller *reallocates* the VM to the next node on the list. This process continues until the VM is created or until all nodes have been exhausted. If no nodes can create the VM, then the cloud controller receives a NERA fault.

At the cloud level, Eucalyptus can accommodate a choice of algorithms to select a cluster to which to assign all VMs in a request, but only one algorithm is implemented. The implemented algorithm, called *least-full-first*, polls the clusters to find out which can accommodate the VMs requested and then orders the list from the least to most full (we ordered ties by increasing cluster identifier). Then the cloud controller selects the first cluster from the list and asks that the VMs be created. If the VMs are created successfully, then the cloud controller returns the positive result to the appropriate user; otherwise, the cloud controller reallocates the VMs to the next cluster on the list. 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 order to have a second cloud-level resource allocation algorithm to use in our sensitivity analysis, we implemented an alternate ordering of clusters based on *percent allocated*, i.e., the cloud orders clusters by decreasing proportion of the requested VMs that can be allocated (we still order ties by increasing cluster identifier). For a given simulation, one parameter (*x8*) specifies a cloud-level allocation algorithm to use and another parameter (*x9*) specifies a cluster-level algorithm.

### 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, which spans a distance consistent with the globe). (While unrealistic, random geographic layout makes a reasonable starting point.) Before a simulation commences, cloud and cluster controllers are randomly placed on some number (*x10*) 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 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, which covers a reasonable range) of packets. Individual packets are subjected to transmission delay (1 Gigabits per second rate here, which is reasonable for intra-site communications and is somewhat optimistic for inter-site communications) 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, which is reasonable within a site). Individual packets are also subjected to a loss rate ($10^{-12}$ here for intra-site packets, which are rarely lost in practice). To simulate Internet congestion, the loss rate for inter-site packets varies uniformly within a range (*x11*). Lost packets are retransmitted, but only for a maximum number (3 here, which seems to be a reasonable threshold) of attempts, after which the related WS message is declared undeliverable.

## III. Experiment Design

We designed a series of sensitivity analysis experiments to identify the most significant parameters driving the behavior of Koala. Our experiments covered (see Table IV) the ten parameters (*x2* to *x11*) discussed above and one additional parameter (*x1*) specifying the number of simulated hours. This additional parameter allowed us to investigate the influence of beginning simulations in an empty state. We assessed the effect of these 11 parameters on 40 responses (see Table V), which were selected to obtain a wide view of system dynamics, and which were thought not to be redundant. As explained below in Sec. IV, our analysis methods enabled us to distinguish a lower dimensional response space embedded within these 40 responses.

We adopted a 2-level experiment design for our sensitivity analysis. This means we selected two values for each of the 11 parameters. Because we could not afford to run a full factorial design (i.e., $2^{11} = 2048$ simulations), we chose to adopt a $2_{IV}^{11-5}$ orthogonal fractional factorial (OFF) experiment design [3]. An orthogonal fractional factorial design entails running a subset ($n$) of the 2048 experiments, while maintaining a balanced ($n/2$ runs at each level) and orthogonal ($n/4$ runs at each possible pair of levels) distribution of parameter configurations. Design resolution defines the degree of confounding that may occur from running a subset of experiments; higher resolution means less confounding. Resolution IV designs eliminate confounding among single parameters or between single parameters and 2-parameter interactions.

TABLE IV. Koala input parameters selected for sensitivity analysis

| Category | ID | Parameter Name |
|---|---|---|
| Duration | *x1* | Simulation duration in hours |
| Demand Layer | *x2* | Number of users |
| | *x3* | Probability of user's type |
| | *x4* | Average (and shape of) user holding time |
| Supply Layer | *x5* | Number of clusters |
| | *x6* | Number of nodes per cluster |
| | *x7* | Probability of platform configuration type |
| Resource Control Layer | *x8* | Algorithm for selecting cluster |
| | *x9* | Algorithm for selecting node |
| Internet/ Intranet Layer | *x10* | Number of sites for cloud components |
| | *x11* | Probability range of packet losses |

Table V. Koala responses selected for sensitivity analysis

| Category | ID | Response Name (Definition) |
|---|---|---|
| User-Level Responses | *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) |
| Cloud-Level Responses | *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 Count Load (Avg. Virtual NICs Allocated / Physical NICs in Cloud) |
| Cluster-Level Responses | *y16* | vCore Util. Var. (Avg. Variance in vCore Utilization across Clusters) |
| | *y17* | Memory Util. Var. (Avg. Variance in Memory Utilization across Clusters) |
| | *y18* | Disk Space Util. Var. (Avg. Variance in Disk Space Utilization across Clusters) |
| | *y19* | pCore Load Var. (Avg. Variance in pCore Load across Clusters) |
| | *y20* | Disk Count Var. (Avg. Variance in Disk Count Load across Clusters) |
| | *y21* | NIC Count Var. (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* | SD-NERA (Stand. Dev. in Avg. NERA Rate across Clusters) |
| | *y27* | SD-Full-Grant (Stand. Dev. in Avg. Full-Grant Rate across Clusters) |
| | *y28* | SD-Allocation-Rate (Stand. Dev. in Allocation Rate across Clusters) |
| VM-Level Responses | *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) |
| Message-Level Responses | *y37* | WS Message Rate (Avg. # WS Messages Send Per Simulated Hour) |
| | *y38* | Intra-Site Messages (# WS Messages Sent with Sites / # WS Messages Sent) |
| | *y39* | Inter-Site Loss Rate (Avg. Fraction of Inter-Site WS Messages Undelivered) |
| | *y40* | Intra-Site Loss Rate (Avg. Fraction of Intra-Site WS Messages Undelivered) |

The $2_{IV}^{11-5}$ design required us to run only 64 simulations ($n = 2^{11-5}$). Since a 2-level design considers only two values for each parameter, we chose to increase the robustness of our conclusions by creating two designs, which we designate SA1-small and SA2-small (see Table VI). Where possible, SA2-small increases the distance between the parameter

values used in SA1-small. Since SA1-small and SA2-small simulate relatively small Cloud configurations (250 to 12,000 nodes), we were able to run six repetitions of each of the 64 parameter combinations for each of the two designs, giving a total of (2 x 64 x 6 =) 768 simulations. Running repetitions with different random number seeds increased our confidence in the results.

We also constructed two designs, which we designate SA1-large and SA2-large, to expand the size of the simulated Cloud (2500 to 60,000 nodes) and to multiply tenfold the simulated user population. We took this step to investigate whether any conclusions changed with increasing Cloud size. Due to the time required, we were able to simulate only 64 parameter combinations each for SA1-large and SA2-large (raising the total number of simulations to 896). Table VI defines the parameter values chosen for all four experiment designs.

Table VI. Parameter settings for sensitivity analysis experiments SA1-small, SA1-large, SA2-small and SA2-large

| | SA1-small and SA1-large | | SA2-small and SA2-large | |
|---|---|---|---|---|
| **Parameter** | **Plus Level** | **Minus Level** | **Plus Level** | **Minus Level** |
| *x1* | 1200 hours | 600 hours | 1600 hours | 200 hours |
| *x2* | 500 (SA1-small)<br>5000 (SA1-large) | 250 (SA1-small)<br>2500 (SA1-large) | 750 (SA2-small)<br>7500 (SA2-large) | 125 (SA2-small)<br>1250 (SA2-large) |
| *x3* | PU1 = 0.2<br>PU2 = 0.2<br>PU3 = 0.1<br>PU4 = 0.1<br>WS1 = 0.15<br>WS2 = 0.07<br>WS3 = 0.03<br>PS1 = 0.1<br>PS2 = 0.01<br>MS1 = 0.1<br>MS3 =0.01<br>DS1 = 0.10<br>DS2 = 0.01 | PU1 = 1/6<br>PU2 = 1/6,<br>WS1 = 1/6<br>MS1 = 1/6<br>PS1 = 1/6<br>DS1 = 1/6 | PU1 = 0.4<br>PU2 = 0.4<br>PU3 = 0.1<br>PU4 = 0.05<br>PU5 = 0.025<br>PU6 = 0.025 | WS1 = 0.25<br>WS2 = 0.15<br>WS3 = 0.1<br>PS1 = 0.35<br>PS2 = 0.04<br>PS3 = 0.01<br>DS1 = 0.08<br>DS2 = 0.015<br>DS3 = 0.005 |
| *x4* | 8 hours ($\alpha$ = 1.2) | 4 hours ($\alpha$ = 1.2) | 12 hours ($\alpha$ = 1.2) | 2 hours ($\alpha$ = 1.2) |
| *x5* | 20 (SA1-small)<br>40 (SA1-large) | 10 (SA1-small)<br>20 (SA1-large) | 30 (SA2-small)<br>40 (SA2-large) | 5 (SA2-small)<br>10 (SA2-large) |
| *x6* | 200 (SA1-small)<br>1000 (SA1-large) | 100 (SA1-small)<br>500 (SA1-large) | 400 (SA2-small)<br>1500 (SA2-large) | 50 (SA2-small)<br>250 (SA2-large) |
| *x7* | C22 = 1.0 | C8 = 0.25<br>C14 = 0.25<br>C18 = 0.25<br>C22 = 0.25 | C14 = 0.2<br>C16 = 0.2<br>C18 = 0.2<br>C20 = 0.2<br>C22 = 0.2 | C2 = 0.1<br>C4 = 0.1<br>C6 = 0.1<br>C8 = 0.1<br>C10 = 0.1<br>C12 = 0.1<br>C16 = 0.1<br>C22 = 0.3 |
| *x8* | Percent Allocated | Least-Full First | Percent Allocated | Least-Full First |
| *x9* | Next-Fit | First-Fit | Next-Fit | First-Fit |
| *x10* | 4 | 1 | 8 | 1 |
| *x11* | $10^{-3}$ to $10^{-8}$ | $10^{-4}$ to $10^{-9}$ | $10^{-2}$ to $10^{-7}$ | $10^{-5}$ to $10^{-10}$ |

## IV. RESULTS

Each repetition of each experiment generated a multivariate dataset consisting of 64 rows (one per parameter combination) by 40 columns (one per response). We applied two types of analysis to each dataset: (1) correlation and clustering analysis (CCA) to identify salient response dimensions and (2) main-effects analysis to determine which parameters had statistically significant influence on relevant responses. CCA [10] allowed us to remove redundancy from the 40 responses, creating a lower dimensional response space and identifying the main behaviors intrinsic to Koala. CCA begins with computation of correlation coefficients ($r$) between each pair of responses. Analysis of a histogram of the resulting $r$ values enabled us to identify a threshold ($|r| >$ 0.65 here) above which to retain correlation pairs. We then clustered the retained pairs into mutually correlated groups that represent the main response dimensions.

We conducted CCA separately for the dataset from each experiment. We report the results in Table VII, which reveals eight response dimensions common to all experiments. (Note that some responses cluster into multiple groups.) In four cases, which appear as split cells in Table VII, responses that clustered together in some experiments did not cluster together in others. A response uncorrelated with any others appears as a singleton group (as shown in 12 cases in Table VII). In general, CCA found similar clustering among all experiments.

TABLE VII. Response dimensions identified with correlation analysis and clustering in each experiment (response selected to represent dimension is highlighted in red and shown in enlarged font)

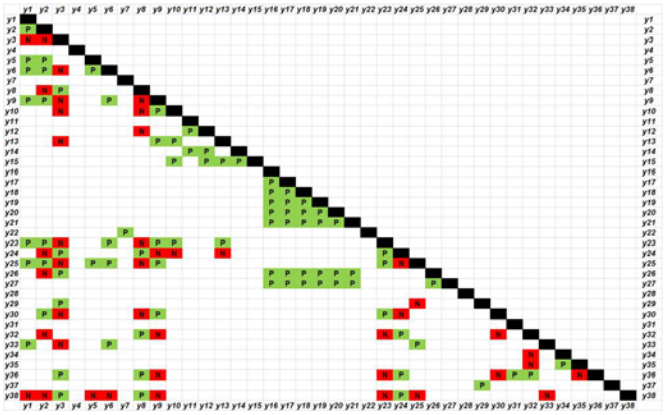
| Response Dimension | SA1-small (9 dimensions) | SA1-large (8 dimensions) | SA2-small (10 dimensions) | SA2-large (9 dimensions) |
|---|---|---|---|---|
| Cloud-wide Demand/Supply Ratio | *y1, y2,* ***y3****, y5, y6, y8, y9, y10, y13, y23, y24, y25, y29, y30, y32, y34, y36, y38* | *y1, y2,* ***y3****, y5, y6, y7, y8, y9, y10, y13, y23, y34, y25, y29, y30, y32, y33, y34, y36, y38* | *y1,* ***y2****, y3, y5, y6, y8, y9, y10, y11, y13, y14, y15, y23, y24, y25, y38* | *y1, y2, y3, y5, y6, y8, y9,* ***y23****, y24, y25, y38* |
| Cloud-wide Resource Usage | *y10, y11, y12, y13, y14,* ***y15*** | *y10, y11, y12, y13, y14,* ***y15*** | ***y10****, y11, y12, y13, y14, y15* | ***y10****, y11, y12, y13, y14, y15* |
| Variance in Cluster Load | *y16, y17, y18, y19,y20, y21,* ***y26****, y27* | *y16, y17, y18, y19,y20, y21,* ***y26****, y27* | *y16, y18, y19, y20, y21, y26,* ***y27***<br>***y17*** **(Mem. Util)** | *y16, y17, y18,* ***y19****,y20, y21, y26, y27* |
| Mix of VM Types | *y34,* ***y35*** **(WS)**<br>***y31*** **(MS)** | ***y31*** **(MS)** | *y12, y14, y15, y30, y31, y33, y34, y35,* ***y36*** | *y14, y15, y30,* ***y31****, y33, y34, y35*<br>*y15,* ***y36*** **(DS)** |
| Number of VMs | *y29,* ***y37*** | ***y37*** | *y29,* ***y37*** | ***y29*** |
| User Arrival Rate | ***y4*** | ***y4*** | ***y4*** | ***y4****, y37* |
| Reallocation Rate | ***y7****, y22* | *y7,* ***y22*** | ***y7*** **(cluster)**<br>***y22*** **(node)** | *y7,* ***y22*** |
| Variance in Choice of Cluster | ***y28*** | ***y28*** | ***y28*** | ***y28*** |

As a next step, we identified one response (highlighted in red, enlarged font in Table VII) to represent each dimension. We selected the response in a cluster that exhibited highest average correlation with other responses and that did not belong to any other clusters. (Note that this was not possible for the cluster "Cloud-wide resource usage" in SA2-small, so we selected *y10*, which exhibited highest average correlation.)

By considering whether correlations are positive or negative, CCA can also be used to determine if Koala yields reasonably correct behaviors. Table VIII gives a half-matrix showing whether significant ($|r| > 0.65$) response correlations were positive (green **P**) or negative (red **N**). Here we use correlations from experiment SA1-small, which gave the largest number (126) of correlated pairs. As should be expected, clusters with positive correlations appear: for cloud-wide resource usage (as indicated by the utilization of virtual cores, *y10*, memory, *y11*, and disk space, *y12*, and by the load on physical cores, *y13*, physical disks, *y14*, and network interfaces, *y15*); for variance among clusters in these utilizations (*y16-y18*) and loads (*y19-y21*); for the

number of VMs (*y29* and *y37*); and for the reallocation rate, reflecting cases where the cloud controller's choice of cluster cannot accept a set of VMs (*y7*) and where a cluster controller's choice of node cannot accept a VM (*y22*).

TABLE VIII. Correlation half-matrix for 38 responses taken from experiment SA1-small – diagonal (black cells) $r = 1.0$; cells above diagonal omitted (mirror of cells below diagonal); colored cells ($|r| > 0.65$) indicate either positive (**P** – green) or negative (**N** – red) correlation



The largest cluster, representing cloud-wide demand/supply ratio shows both positive and negative correlations. For example, full-grant rate (*y3*) is inversely correlated with user request rate (*y1*), i.e., fewer requests mean more can be fully granted; with NERA rate (*y2*), i.e., more NERA faults imply lower resource availability, leading to fewer full grants; with grant latency (*y6*), i.e., longer waiting for VMs coincides with lower resource availability, leading to fewer full grants; and so on. Upon inspection these inverse correlations appear sensible. Similarly, inverse correlations appear among responses (*y30*-*y36*) representing the mix of VM types. Of note, the proportion of M1small instances (*y30*) is inversely correlated with the proportion of M1xlarge (*y32*) and M4xlarge (*y36*) instances. And *y32* and *y36* are positively correlated. This makes sense because M1xlarge and M4xlarge VMs (recalls Table I) require more virtual cores, memory and disk space than M1small instances. More resources taken up by large VMs, means that fewer small VMs that can be accommodated.

Next we applied main-effects analysis (MEA) [11] separately to each selected response (from Table VII) in each experiment dataset, including the repetitions of SA1-small and SA2-small. MEA iterates over each selected response, which will be represented by 64 data points, one for each parameter combination. For each selected response, MEA iterates over each of the 11 parameters, dividing the 64 data points into two groups of 32: results obtained with the parameter at the PLUS level and at the MINUS level. For each parameter, we applied a *t*-test [12] to determine whether the averages of the PLUS and MINUS level data points were significantly different, and if so at which confidence level: $p < 0.05$ or $p < 0.01$. For each parameter (*x1* to *x11*), we computed the percent of responses influenced (Ψ), weighting $p < 0.05$ at ½ and $p < 0.01$ at 1, as shown with the following equation.

$$\Psi = (|\{y \mid p < 0.01\}| + ½\,|\{y \mid p < 0.05\}|\,/\,|\{y\}| \qquad (1)$$

Table IX displays the resulting Ψ for each parameter in each of the four experiments. The bottom row gives a weighted (by relative proportion of experiment repetitions) average Ψ for each parameter.

TABLE IX. Percent responses influenced (Ψ) by each parameter in each experiment and weighted average Ψ across all experiments (<green> = major influence; {yellow} = modest influence; (orange) = minor influence; [gray] = no influence)

| | | Input Parameter | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Experiment | Weight | *x1* | *x2* | *x3* | *x4* | *x5* | *x6* | *x7* | *x8* | *x9* | *x10* | *x11* |
| SA1 small | 6/14 | [1] | <57> | {22} | (11) | <44> | {29} | {30} | (12) | [0] | [1] | [0] |
| SA1 large | 1/14 | [0] | <69> | (13) | {25} | <44> | <56> | {31} | {25} | [0] | (13) | [0] |
| SA2 small | 6/14 | [2] | <73> | {38} | (10) | <45> | <62> | (10) | (17) | [1] | [0] | [0] |
| SA2 large | 1/14 | [0] | <56> | <50> | (11) | {39} | <56> | [6] | (11) | [0] | [0] | [0] |
| Avg. Ψ | Est. | [1] | <65> | {30} | (12) | <44> | <47> | {20} | (15) | [0] | [1] | [0] |

The data used to establish the significance of model parameters may also provide insight into model correctness by considering the relative effects on responses when parameter settings move from the MINUS level to the PLUS level. To measure the relative effect for a specific response and selected parameter we subtracted the mean response when the parameter was at the MINUS level from the mean response when the parameter was at the PLUS level and then divided that result by the mean response for both levels, which yields the percentage change (Δ) in response due to changing the parameter level. A positive change means that increasing the parameter level increased the response, while a negative change means increasing the parameter level decreased the response. Table X gives the relative effect, averaged over simulation repetitions, that each parameter has on eight responses, selected to represent each of the eight dimensions shown in Table VII.

TABLE X. Relative effect (Δ) that each parameter has on eight responses, selected to represent each of the eight dimensions shown in Table VII. (<green> = $\Delta > 50$; {yellow} = $\Delta \geq 30$ & $\Delta < 50$; (orange) = $\Delta > 10$ & $\Delta < 30$; [gray] = $\Delta < 10$)

| | | Input Parameter | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Dimension | Selected Response | *x1* | *x2* | *x3* | *x4* | *x5* | *x6* | *x7* | *x8* | *x9* | *x10* | *x11* |
| Cloud-wide Demand/ Supply Ratio | *y3* | [1] | {-38} | (-21) | [-5] | {37} | {40} | (25) | [-2] | [1] | [5] | [-1] |
| Cloud-wide Resource Usage | *y15* | [1] | (23) | <53> | [1] | (-22) | (-18) | (19) | [-1] | [1] | [-1] | [1] |
| Variance in Cluster Load | *y26* | [0] | <-101> | (28) | [-5] | <96> | <59> | <66> | {42} | [0] | <50> | [0] |
| Mix of VM Types | *y31* | [-1] | [-9] | {43} | [-3] | [7] | [9] | [-1] | [8] | [2] | [-4] | [0] |
| Number of VMs | *y37* | [-5] | {48} | (11) | (-23) | <79> | <53> | [-5] | [8] | [-1] | [4] | [-2] |
| User Arrival Rate | *y4* | (-17) | <87> | [2] | <-80> | (29) | {31} | (15) | [-4] | [-1] | *x1* | [-5] |
| Reallocation Rate | *y7* | [0] | [0] | [0] | [0] | [0] | [0] | [0] | [0] | [0] | [0] | [0] |
| Variance in Cluster Choice | *y28* | [6] | (-12) | {-42} | [7] | {-35} | {32} | (18) | <97> | [2] | [4] | [6] |

Table X indicates that the full-grant rate (*y3*) decreases with an increase in the number of users (*x2*) and increases with an increase in the number of clusters (*x5*) and nodes (*x6*). This makes sense intuitively. One should expect an inverse of these relationships for responses (e.g., *y1*, *y2* and *y6*) that correlate negatively with *y3*. On the other hand, cloud-wide resource usage (e.g., *y15*) increases with an

increase in the number of users (*x2*) and decreases with an increase in the number of clusters (*x5*) and nodes (*x6*). This also makes sense intuitively. Yet another sensible relationship appears for the number of VMs (*y37*), where increasing the number of users (*x2*), clusters (*x5*) and nodes (*x6*) all lead to increase in the number of VMs. The user arrival rate (*y4*) increases with an increase in the number of users (*x2*) and decreases with an increase in the time (*x4*) that users hold VMs. These relationships make sense. The number of M1large instances (*y31*) is influenced significantly only by changing the user type probability (*x3*), where changing the parameter level alters the diversity of VM types requested. Cluster reallocations (*y7*) occur so infrequently as to have no measureable relative effect for any parameter. These relationships serve to increase confidence in model correctness.

Response *y26*, representing standard deviation in the rate at which clusters report NERA, provides a substantial amount of information. First, increasing the number of users (*x2*) decreases standard deviation; while increasing the number of nodes (*x5*) and clusters (*x6*), and the capacity (*x7*) of nodes, increases standard deviation. This means that higher system loads increase correlation in clusters reporting NERA, while lower system loads diminish the correlation. Second, selecting clusters based on the percent of VMs that can be allocated (*x8*) increases the variance in load among clusters, which is reflected in increasing *y26*. Third, distributing cloud elements among geographically disparate sites (*x10*) also increases the variance in load among clusters, probably due to differences in response times to queries from the cloud controller. These relationships reveal subtleties that could have been easily overlooked in less rigorous experiment designs and analyses.

## V. Discussion

Koala behavior can be represented by about eight dimensions (i.e., eight of the 40 responses we collected). The largest grouping of responses reflects the cloud-wide ratio between supply and demand. Smaller response groupings represent cloud-wide resource usage, variance in load among clusters in a Cloud, and the mixture of VM types within a Cloud. The number of VMs, regardless of type, can be discerned separately, as can the user arrival rate. Two minor behavioral dimensions reflect the rate of reallocation decisions and variance in choice of eligible clusters on which to run a group of VMs.

Koala behavior is influenced by about seven of the 11 parameters we evaluated. The main influences encompass the number of users (*x2*) and the number of clusters (*x5*) and nodes per cluster (*x6*). More modest influence was exhibited by the distribution of user (*x3*) and platform (*x7*) types. Minor influence arose from the duration (*x4*) for which users retained acquired VMs and the algorithm (*x8*) used to select a cluster on which to place requested VMs. For the values used here, little influence on Koala behavior was found for the remaining four parameters: the duration of the simulation (*x1*); algorithm (*x9*) for choosing a node within a cluster on which to place a VM; number of sites (*x10*) over which clusters were deployed; and range (*x11*) of packet loss rates.

Our findings indicate that for subsequent experiments, comparing resource allocation algorithms, we need vary only six parameters (*x2*, *x3*, *x4*, *x5*, *x6* and *x7*). (Parameter *x8* will be covered by the use of varying cluster-level resource allocation algorithms.) Further, since the two cluster-level allocation algorithms (*x8*) we used here showed some influence on Koala behavior, we might find that additional algorithms will further influence behavior. While the node-level allocation algorithms (*x9*) used here did not influence Koala behavior, we might find that adding a variety of such algorithms, especially in combination with cluster-level algorithms, could influence behavior. Taken together, our findings suggest that if we use a $2_{VI}^{6-1}$ (Resolution VI) OFF design to compare, with respect to only 8 response variables, combinations of cluster-level and node-level resource allocation algorithms across 32 out of a possible 64 parameter combinations, then we should discern any significant behavioral differences among the algorithms. Since simulation duration (*x1*) had insignificant influence on Koala behavior for the range of simulations we envision, we can average responses across an entire experiment run without biasing results due to startup transients. Including the startup from an empty system will also allow us to discern any differences in the way that various algorithms fill the nodes in a Cloud.

As with most large distributed systems that we have modeled, changing combinations of fundamental parameters (such as *x2*, *x3*, *x4*, *x5*, *x6* and *x7* in this study) stimulates large differences in global system behavior. In studying Internet congestion control algorithms [2], for example, we found that changes in fundamental network parameters (such as speed, propagation delay, buffer sizes, topology and user demand) varied global network behavior much more than adopting alternate congestion control algorithms. As predicted by our sensitivity analysis, we expect that changing values of fundamental parameters in our Cloud model will drive global behavior much more than adopting alternate resource allocation algorithms. In comparing algorithms, the question becomes: How do the algorithms differ when exposed to the wide range of global behaviors that are possible within the system in which the algorithms must operate? Now that we have a well understood system model, we are prepared for rigorous comparison of resource allocation algorithms – a subject for future work.

## VI. Related Work

Our work on methods to improve simulation modeling of large systems was inspired by Paxson and Floyd [1], who describe many difficult problems that impede simulation of the Internet, and recommend two main coping strategies: search for invariants and carefully explore the parameter space. While providing sound advice on “what” to do, Paxson and Floyd did not show “how” to accomplish such search and exploration. This issue remains open over a decade since Paxson and Floyd wrote. Recently, we developed methods that can be used to search for invariants and carefully explore the parameter space of large simulation models. Previously, we demonstrated our methods in the problem domain of Internet simulation [2], which provided

the original challenge identified by Paxson and Floyd. We believe our methods can be applied generally to simulations of large distributed systems, as we demonstrated in this paper with respect to an infrastructure Cloud simulator. Other Cloud simulators stand to benefit from adopting our methods.

Andrzejak and colleagues [13] propose a model that allows users facing constraints on cost, performance and reliability to bid optimally for Cloud services in spot markets, where VMs may be allocated and withdrawn based on changing market prices. Their paper describes a typical factor-at-a-time experiment where 11 parameters are set to fixed values and the effects of those settings on system behavior are determined. Then individual Monte Carlo experiments (10,000 simulations each) are run to determine the effects of varying task length on up to six response variables for two different workloads. By adopting the methods we demonstrate, Andrzejak and colleagues could determine, with many fewer simulations, which parameters are most influential on system behavior. Subsequently they could design more informative experiments.

Fujiwara and colleagues [14] propose an auction model allowing users and multiple Cloud providers to organize a dual market that includes both spot and reservation components. Their paper reports two sets of simulation experiments. The first set, which includes seven fixed and two stochastic parameters and two responses, amounts to a factor-at-a-time experiment to verify correct operation of a scheduling algorithm. The second set, which includes seven fixed parameters, and three uniformly distributed random parameters, uses two responses to measure scalability as the number of time slots (6 values) and users (4 values) varies. By using our methods, Fujiwara and colleagues could determine which system behaviors are most significant and which parameter combinations would prove most revealing. The current reported results have a limited range of validity.

Buyya and colleagues [15] define CloudSim, a model that application developers may use to predict performance attributes for systems deployed on infrastructure Clouds. By applying our methods, Buyya and colleagues could characterize the macroscopic behavior of their model so that application developers intending to use CloudSim will understand precisely what parameter combinations to simulate and what responses to examine. Their current paper provides only information on the model's runtime and memory usage when varying three parameters.

## VII. Conclusions and Future Work

We described a sensitivity analysis method that can be used to identify parameters that drive macroscopic behavior in simulations of large distributed systems. Our method also identifies significant behavioral dimensions in such models. Previously, we demonstrated our methods in simulations of communication networks. Here we demonstrated our methods on Koala, an infrastructure Cloud simulator. We found that Koala behavior can be measured with as few as eight of the 40 responses we collected. The sensitivity analysis also found that only six parameters significantly influenced Koala behavior.

We plan to use these findings to design subsequent experiments comparing the behavioral influence of various combinations of cluster-level and node-level resource allocation algorithms. Findings from these later experiments should help guide designers of resource allocation algorithms for on-demand infrastructure clouds.

We also argued that our methods can improve the quality of experiments that simulate large distributed systems, such as communication networks and computation clouds. Our methods provide a framework for experimenters to extract that most salient information from available computational resources. The robustness and scope of most large simulation experiments can be improved by adopting our methods.

## 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] 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.

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

[4] D. Frey, F. Engelhardt and E. Greitzer. "A role for 'one-factor-at-atime' experimentation in parameter design," Research in Engineering Design, **14**:**2**, 65-74, 2003.

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

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

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

[8] 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.

[9] 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.

[10] K. Mills and J. Filliben, "Comparison of Two Dimension-Reduction Methods for Network Simulation Models", NIST Publication #906588, presented at the Winter Simulation Conference, Dec. 5-8, 2010.

[11] K. Mills and J. Filliben, "An Efficient Sensitivity Analysis Method for Network Simulation Models", NIST Publication #904961, presented at the Winter Simulation Conference, Dec. 5-8, 2010.

[12] J. L. Phillips, How to Think about Statistics, 6th ed., Freeman, 2000, 202 pages.

[13] 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.

[14] 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.

[15] R. Buyya, R. Ranjan, and R. N. Calheiros, "Modeling and Simulation of Scalable Cloud Computing Environments and the CloudSim Toolkit: Challenges and Opportunities", Proceedings of the 7th High Performance Computing and Simulation Conference, IEEE, June 21-24, 2009, pp. 1-11.