Enhancing Rules For Cloud Resource Provisioning Via Learned Software Performance Models

Mark Grechanik

University of Illinois at Chicago

Qi Luo, Denys Poshyvanyk

College of William and Mary

Adam Porter

University of Maryland at College Park

Cloud Computing Is Everywhere

Global spending on public cloud services estimated to reach $110.3Bil this year

Does it affect engineering software and in what way?

What Is The Fundamental Nature of Cloud Computing?

Traditional Computing	Forecast infrastructure needs
	Purchase infrastructure hardware
	Own and maintain infrastructure
Cloud Computing	Infrastructure is rented
	Resources are dynamically (de)allocated
	Applications are deployed in virtual machines

Elasticity of Cloud Computing

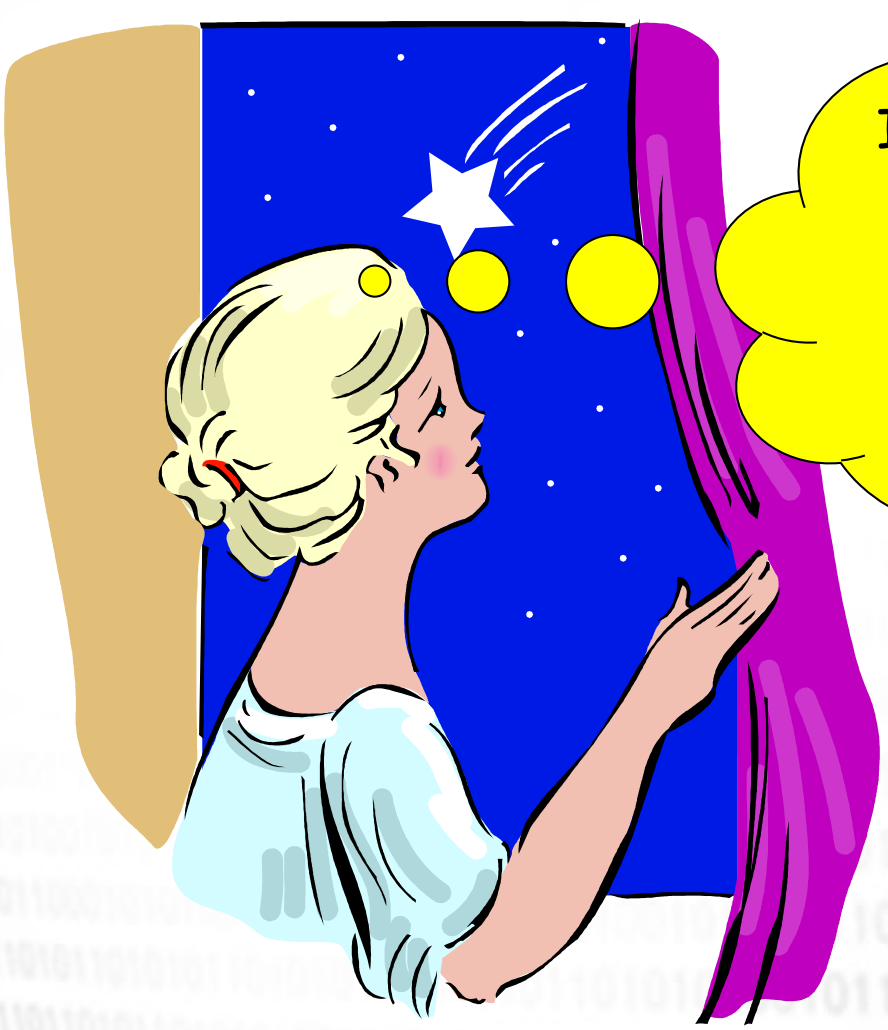
Cloud infrastructure re-allocate resources on demand and are considered **elastic**

- Scale out and up – adding resources to VMs as demand increases
- Scale in and down – releasing resources to VMs as demand decreases

Highly elastic clouds can re-provision resources rapidly and automatically in response to demand

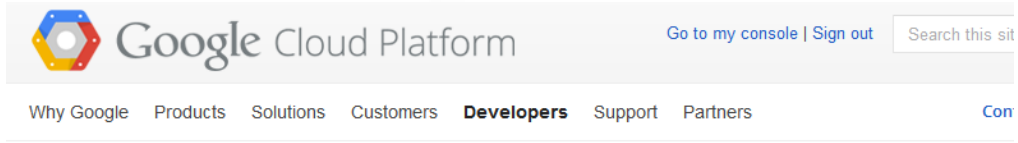
- Stakeholders pay only for using, rather than for owning and employing the hardware/software infrastructures and technical staff that support their applications

The Dream of Elastic Cloud



I wish to significantly reduce the costs by renting rather than owning elastic cloud execution infrastructure!

What Google Offers



Auto Scaling on the Google Cloud Platform

g+1 44

infrequent. The disadvantage of this approach is that many or most resources may lie idle for long periods. This approach makes the application much more expensive than it has to be, because idle but provisioned instances still cost money. Another disadvantage is that it is difficult or impossible to predict peak traffic, particularly for new applications. Mobile games are extremely variable in their uptake, for instance. Many have only a few users, while others have a high number. Resources must be provisioned appropriately for a good user experience.

The second approach is to provision for average use, which wastes fewer resources. The pitfalls of this approach are (1) it can be challenging to predict average use, especially for new applications, and (2) above-average load usually results in a bad user experience, which could mean anything from increased latency to user requests that are dropped entirely.

Any large-scale application needs to run on multiple servers or VM instances to handle user traffic. The question is how many

Solution overview

The orchestration framework involves a tool implemented as an App Engine application. This tool periodically queries all running Compute Engine instances for their status. Depending on the application, the status may be CPU load and memory usage or the number of pending tasks. Based on user-specified criteria, the orchestrator spins up new instances if load is high and initiates the shutdown of instances when load is low.

even a relatively simple auto-scaling process can lead to improved resource utilization without compromising the user experience.

What Amazon AWS Offers

Auto Scaling enables you to closely follow the demand curve for your applications, reducing the need to provision Amazon EC2 capacity in advance. For example, you can set a condition to add new Amazon EC2 instances in increments of 3 instances to the Auto Scaling Group when the average CPU utilization of your Amazon EC2 fleet goes above 70 percent; and similarly, you can set a condition to remove Amazon EC2 instances in the same increments when CPU Utilization falls below 10 percent. Often, you may want more time to allow your fleet to stabilize before Auto Scaling adds or removes more Amazon EC2 instances. You can configure a cool-down period for your Auto Scaling Group, which tells Auto Scaling to wait for some time after taking an action before it evaluates the conditions again. Auto Scaling enables you to run your Amazon EC2 fleet at optimal utilization.

http://aws.amazon.com/autoscaling/

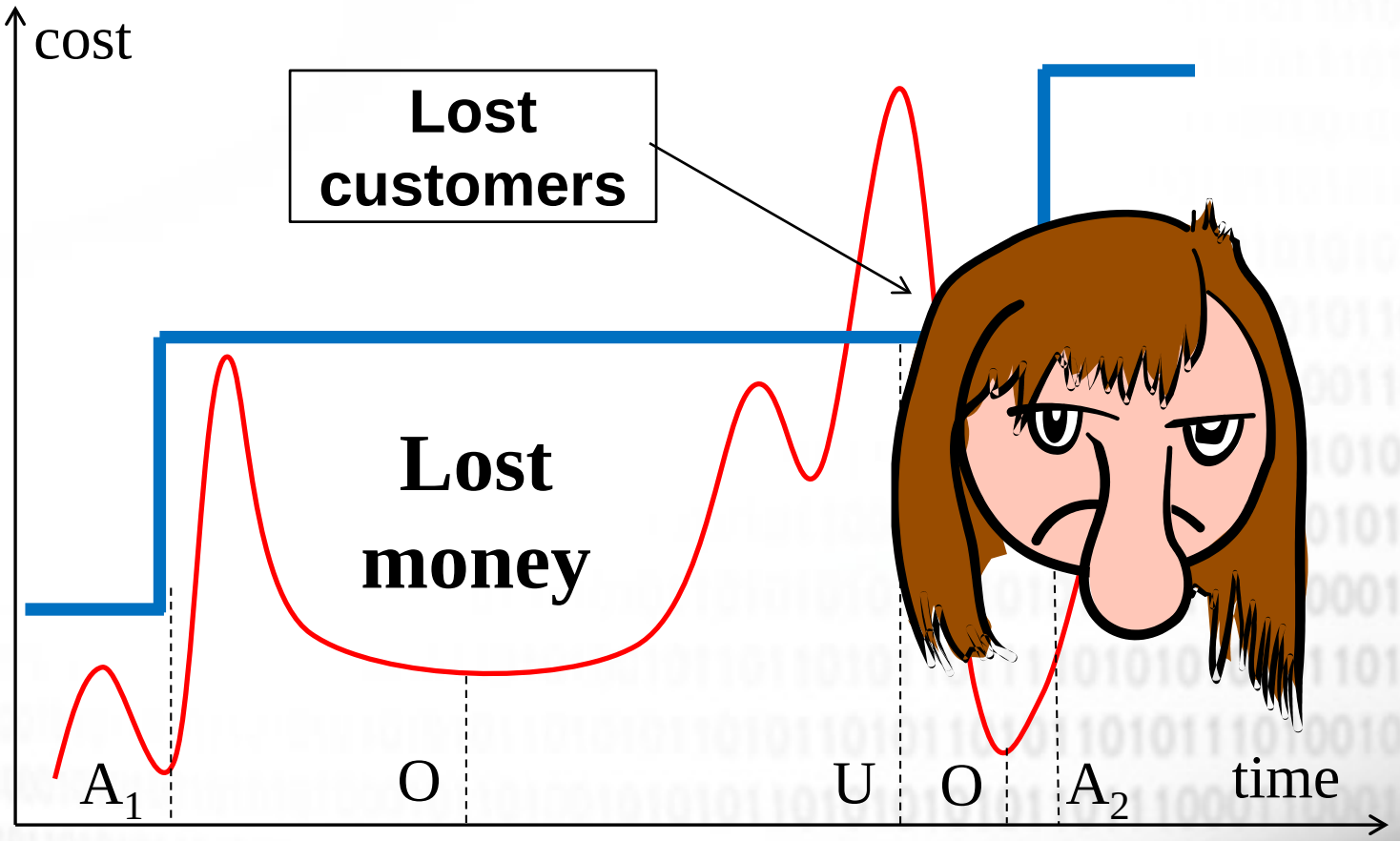
Google And Other Cloud Providers Offer Heuristics

Heuristics for Scaling

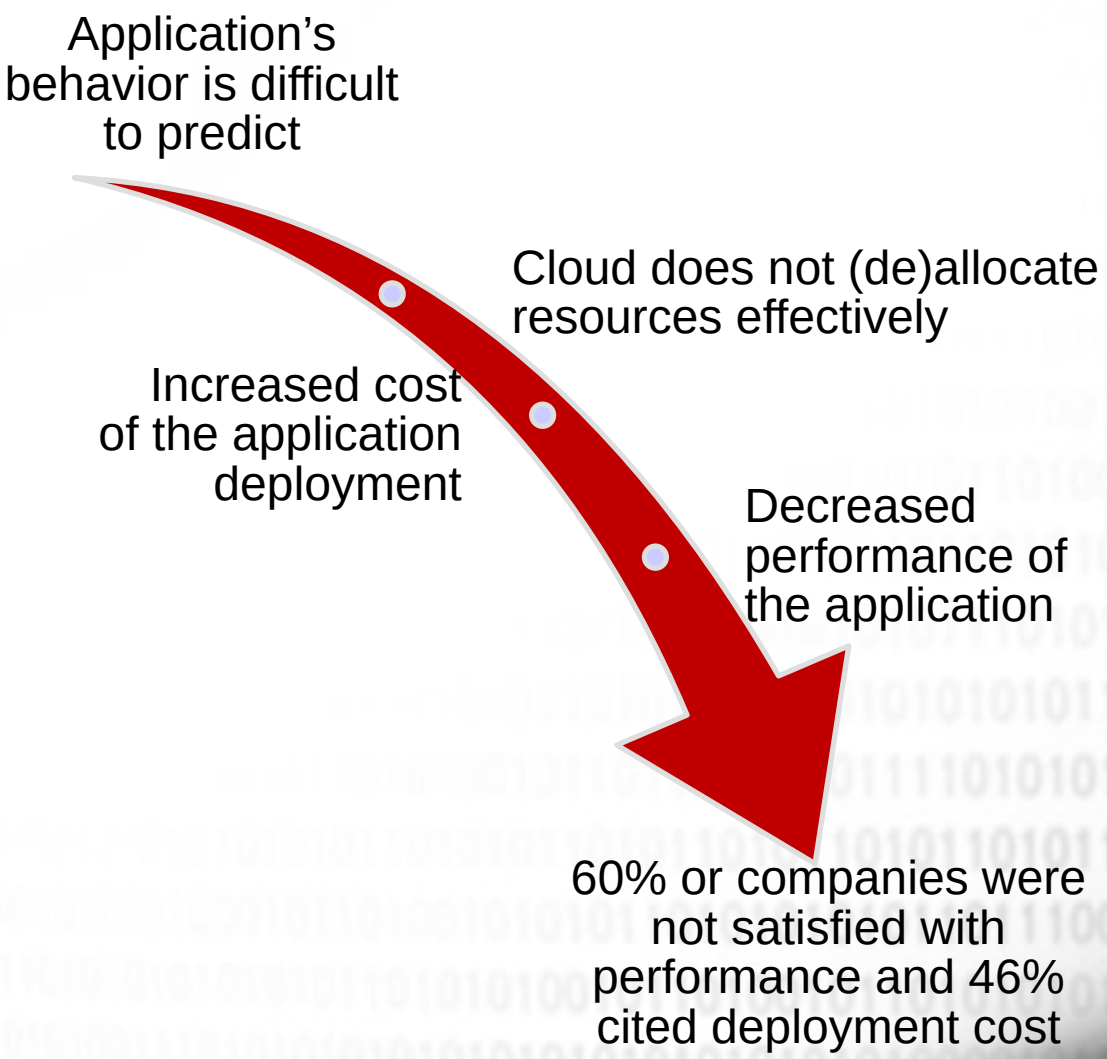
Which heuristics you should use for scaling depend on the specifics of the orchestrated application. We discuss a few cases below. For this discussion, we assume that VMs lease tasks in a predictable fashion, with a fixed number of tasks at fixed intervals. If this is not the case, orchestration is harder to accomplish and is likely based on simpler heuristics, such as CPU utilization only. The effectiveness of this strategy depends on the use case.

1. *Applications with very predictable and homogenous tasks*
   Some applications perform similar tasks repeatedly, such as a specific image-processing task. In such cases, each task or set of tasks leased by a Compute Engine VM is predictable in terms of expected completion time, CPU and

Cost-Utilization Curve



The Problem

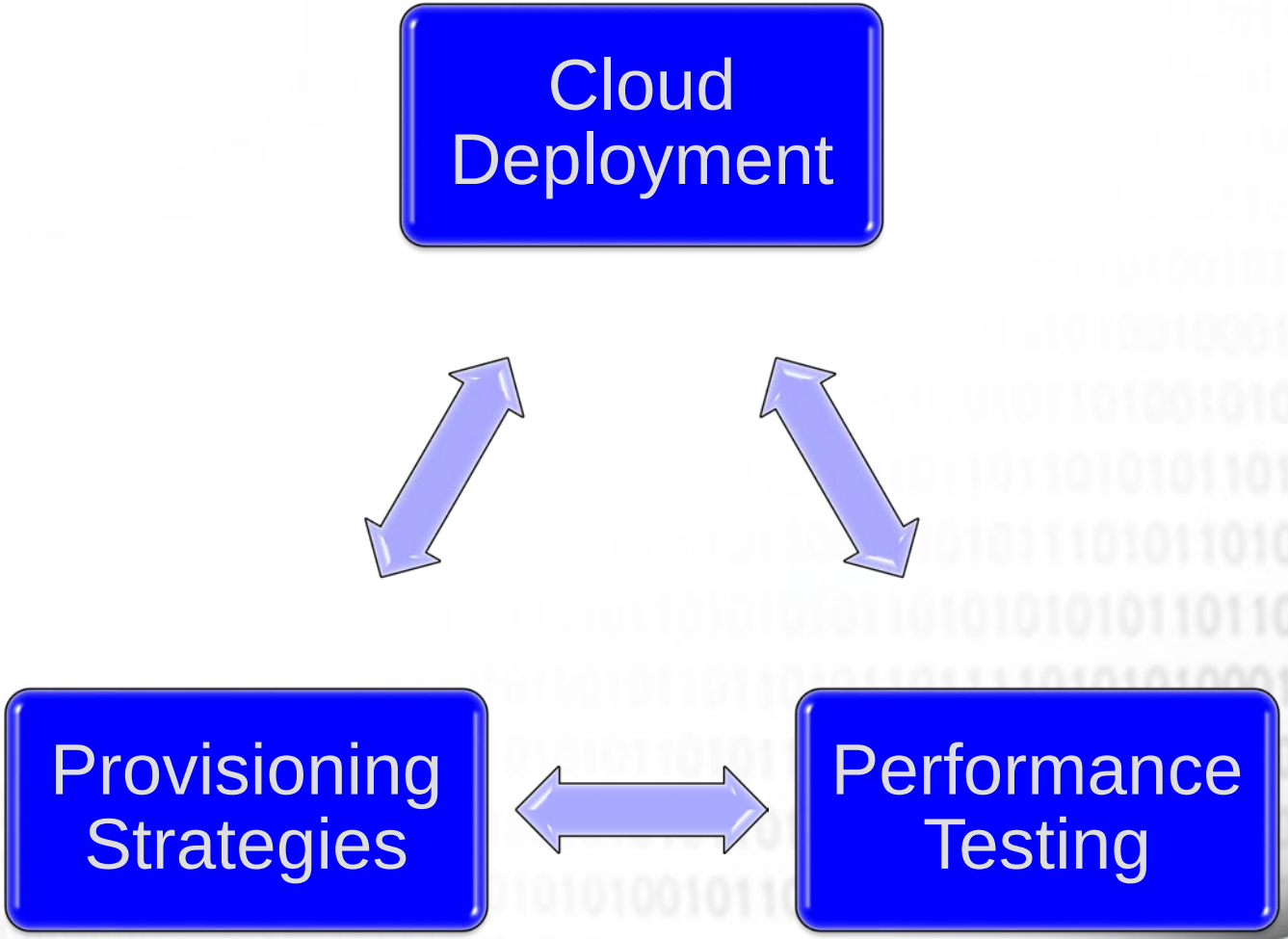

Application's behavior is difficult to predict
Cloud does not (de)allocate resources effectively
Increased cost of the application deployment
Decreased performance of the application
60% or companies were not satisfied with performance and 46% cited deployment cost

Balancing Cost And Performance

Resources cost money, per usage

Precise cloud elasticity is an answer to the problem of providing sufficient resources to software applications while minimizing the cost of deployment

Connect Cloud Deployment With Performance Testing



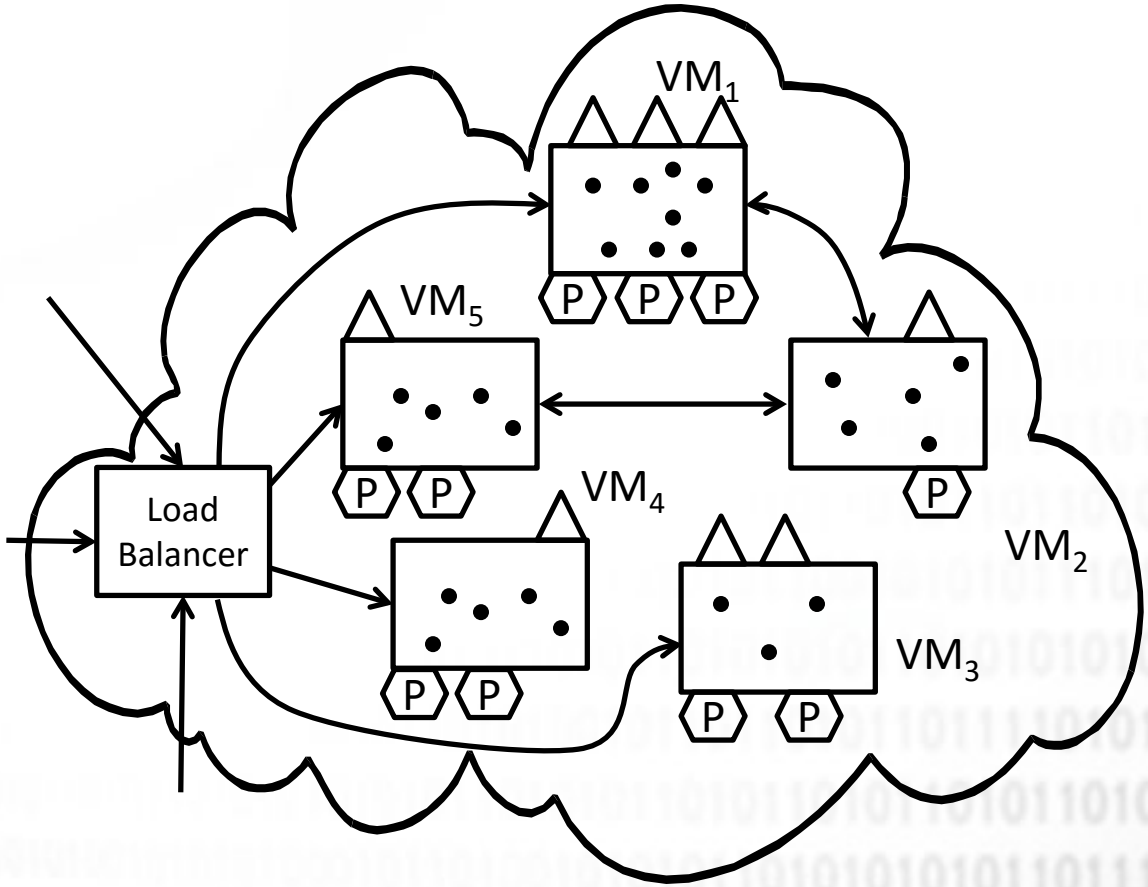
Key Insights

Key software and data artifacts are generated during different stages of the software engineering lifecycle could be, but are not currently being, used by the cloud infrastructures to improve predictive provisioning of resources.

Applications running in the cloud do not provide important feedback to stakeholders about any performance problems their applications may have and that might ultimately require more resources and, thus, increase operational costs.

There is no two-way flow of information between software development and cloud deployment of these applications.

Model of Cloud Computing



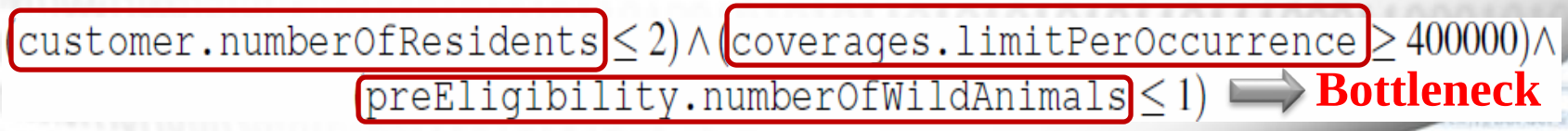
Finding Rules That Describe Performance Problems

A goal is to find situations when applications unexpectedly exhibit worsened characteristics for certain combinations of input values.

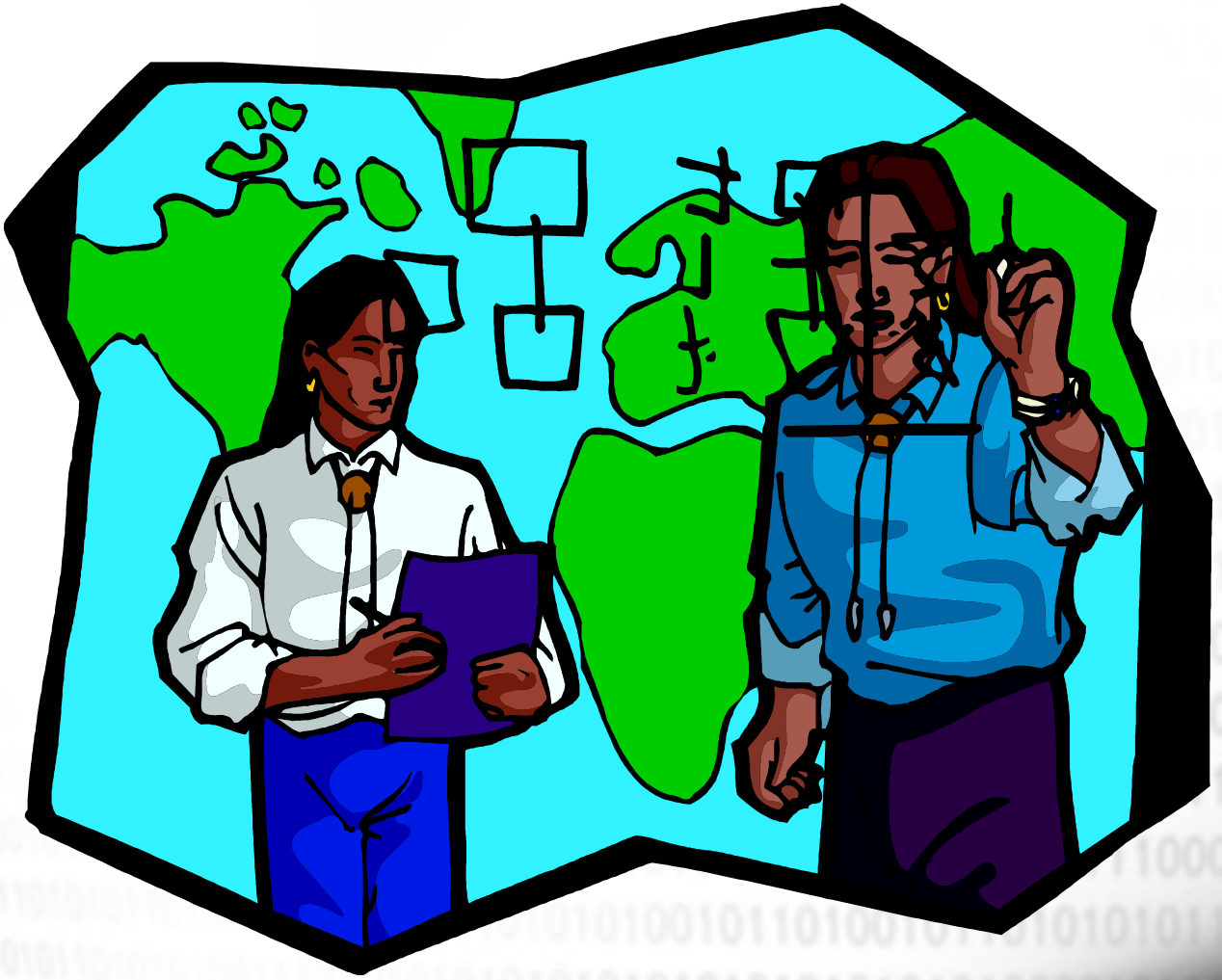
Performance Rules

Descriptive rules for selecting test input data play a significant role in software testing, where these rules approximate the functionality of the application.

- Example rule: some customers will pose a high insurance risk if these customers have one or more prior insurance fraud convictions and deadbolt locks are not installed on their premises.



Provisioning Strategies

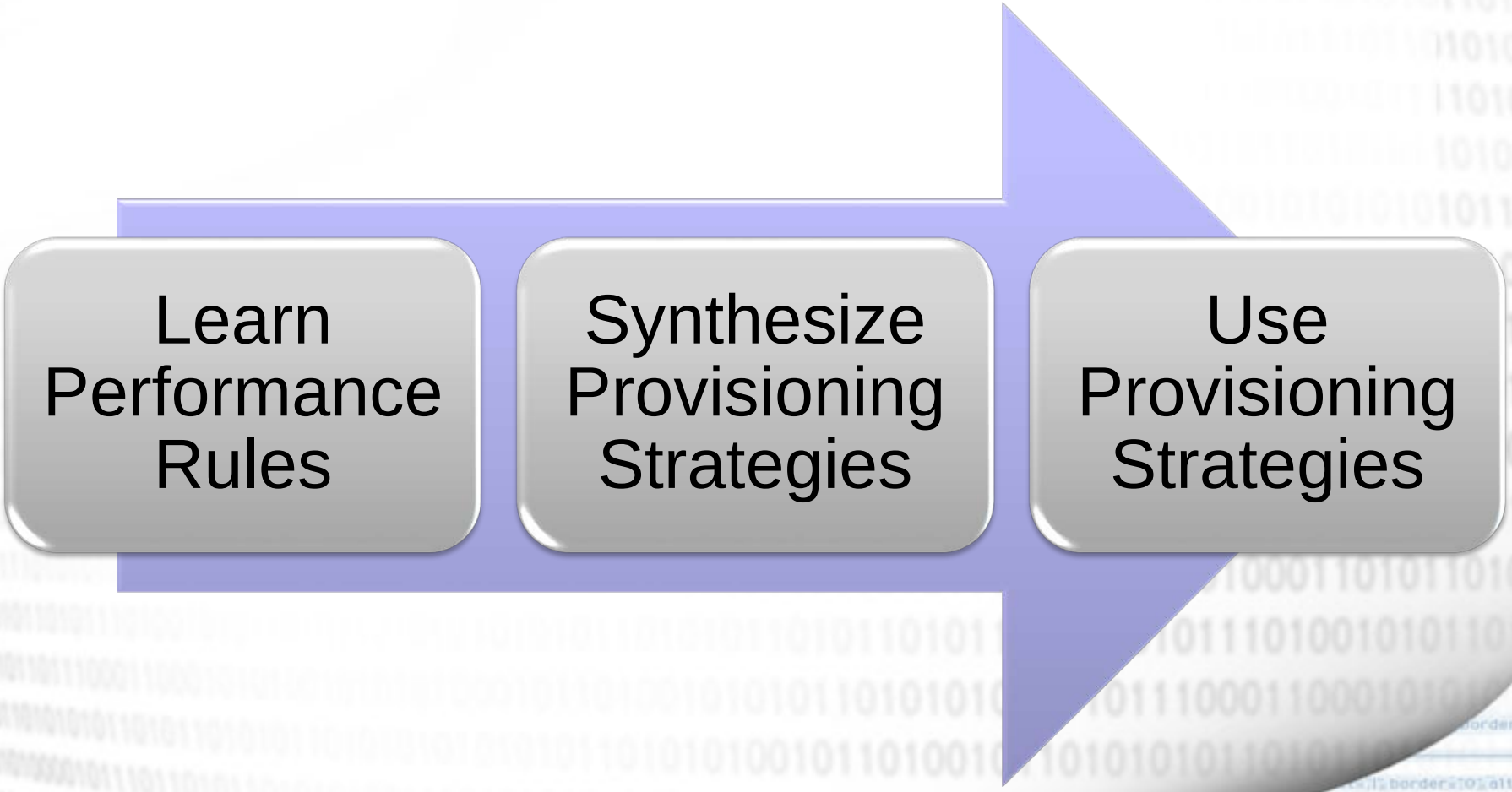


Provisioning Strategies

Definition *A provisioning strategy is a relation $P \Rightarrow R$, where P is a performance rule and R is a resource provisioning scheme in the form $R_p \diamond V_{R_p} \bullet R_q \diamond V_{R_q} \bullet \ldots \bullet R_k \diamond V_{R_k}$, where $\diamond \in \{\uparrow, \downarrow\}$ is a resource (de)allocation operators and $\bullet$ stands for logical connectors* `and` *and* `or` *and V_{I_m} is the specific amount of the resource, R_m. The operator $\uparrow$ stands for allocating a resource to a VM and the operator $\downarrow$ stands for deallocating a resource from a VM.*

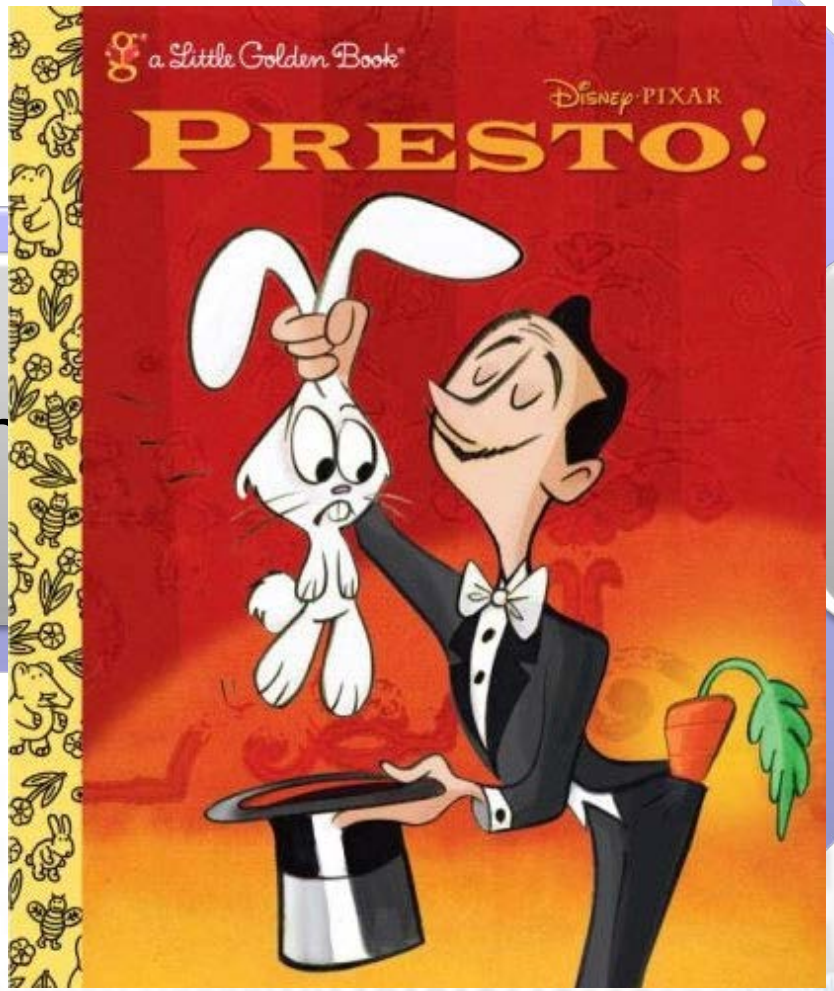
$$(A, \mathbb{R}) \Rightarrow \mathbb{P} \uparrow 2 \wedge \mathbb{M} \uparrow 3.1 \wedge \mathbb{I} \uparrow 2$$

Putting Rules And Strategies Together For Cloud Computing



Provisioning Resources with performancE Software Test automatiOn (PRESTO)

Learn Performar Rules



Use Provisioning Strategies

Performance Rules Are Behavioral Models

Behavioral model is a collection of workload profiles, constraints, performance counters, and various relations among components of the application.

Performance rules are examples of applications behavioral models.

Our goal is to obtain and use these models to synthesize provisioning strategies

Performance Testing Is Laborious and Difficult And

It Is Not Designed For The Cloud

A Computing Primitive in PRESTO - FOREPOST

Feedback-ORiEnted PerfOrmance Software Testing (FOREPOST) finds performance problems automatically by learning and using rules that describe classes of input data that lead to intensive computations.

- FOREPOST is an adaptive, feedback-directed learning testing system that learns rules from execution traces and uses these learned rules to select test input data automatically to find more performance problems in applications when compared to exploratory random performance testing.

Compute a Function That Represents the Application

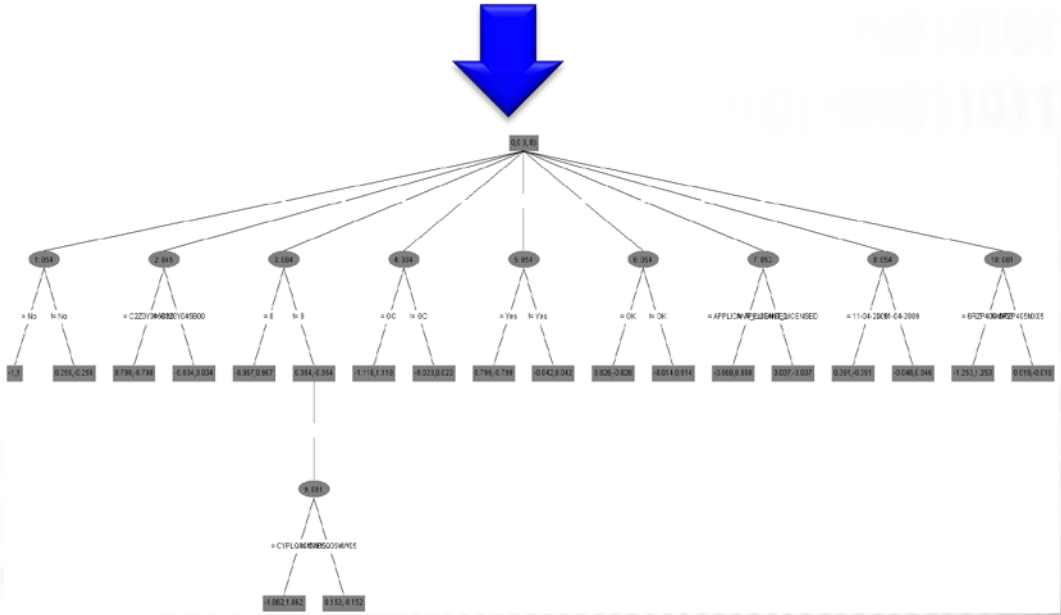
FOREPOST

CLOUD

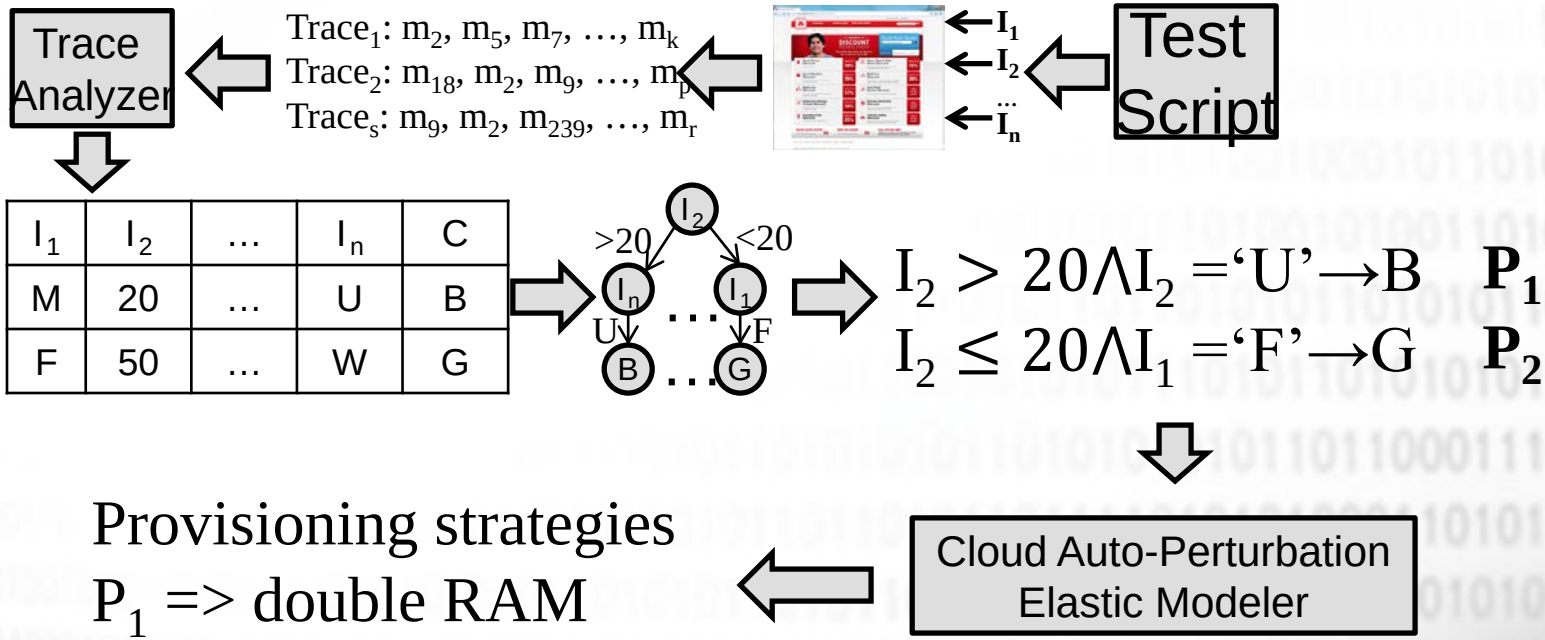
f: Input -> Performance

Constructing Data For a Learner and Learning Rules

Input_1_Name	Input_2_Name	…	Input_k_Name	Class
Value_p	Value_q	…	Value_m	Good
Value_p	Value_q	…	Value_m	Bad
…	…	…	…	…
Value_p	Value_q	…	Value_m	Good



PRESTO Architecture

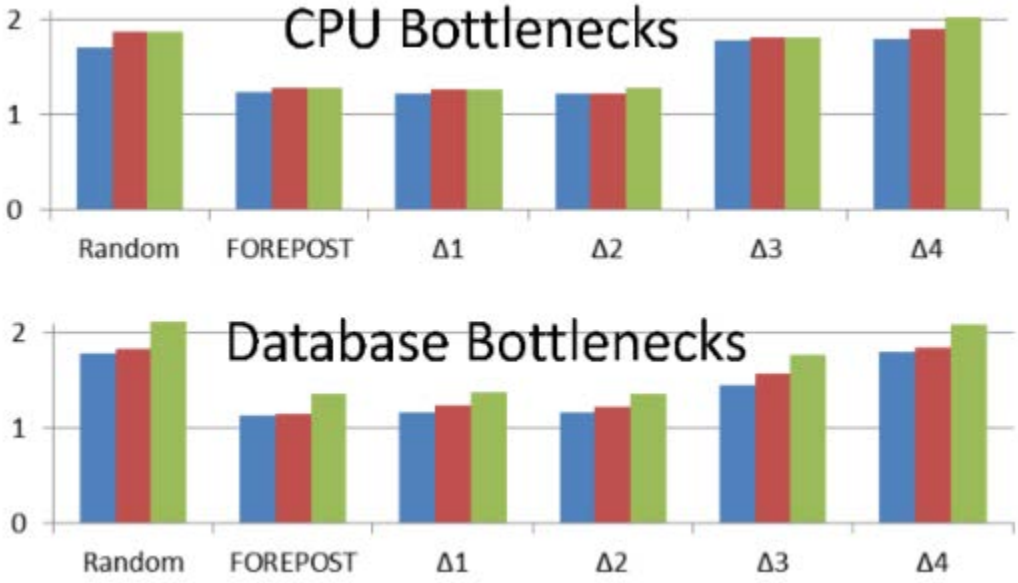


Research Questions

How effective is PRESTO in maintaining the throughput of the applications in the cloud?

How fast and efficient is PRESTO in learning provisioning strategies?

Experimenting With CPU Bottlenecks Using CloudStack



Δ_1: one VM with 1.0 GHz CPU and 1.5 GB RAM;
Δ_2: one VM with 1.5 GHz CPU and 1.0 GB RAM;
Δ_3: one VM, two 1.0 GHz core CPUs, 1.0 GB RAM;
Δ_4: two VMs, one 1.0 GHz CPU, 1.0 GB RAM each.

Conclusions

This proposed research program is novel, as to the best of our knowledge, there exists no work that utilizes automatic feedback-directed adaptive test scripts to find improve cloud elasticity.

The results of this work show that the proposed approach can effectively locate performance problems and guide stakeholders to improve the quality of software that is deployed in the cloud.

Email: drmark@uic.edu