USING REINFORCEMENT LEARNING FOR AUTONOMIC RESOURCE ALLOCATION IN CLOUDS: TOWARDS A FULLY AUTOMATED WORKFLOW

Towards a Fully Automated Workflow, Xavier Dutreilh, Sergey Kirgizov, Olga Melekhova, Jacques Malenfant, Nicolas Rivierre and Isis Truck

K.M.D.M Karunarathna

University Of Cambridge - 24th Nov 2015

Problem

- Dynamic and appropriate resource dimensioning

Solution

- **Current**

  Use ad hoc manually determined policies, such as threshold-based ones

- **Research**

  Research is being done to apply automatic decision-making approaches, such as reinforcement learning.

What they did…

- Careful initialization of the learning functions in order to have a good policy from the start
- Convergence speedups for model-based reinforcement learning which inserts complete policy evaluation steps at regular intervals into the learning phases

Cloud Delivery Models

- **IaaS** designates the provision of IT and network resources, such as processing, storage and bandwidth as well as management software.
- **PaaS** designates the deployment of applications created using particular programming languages and tools supported by a provider onto his own cloud infrastructure
- **SaaS** designates the use of applications running on a cloud infrastructure

Cloud Usage

- Data processing applications (from development tools like continuous integration suites to business tools as video transcoders/[report,convertion])
- Transaction-processing software (including social networks and e-commerce websites)
- Event-processing systems (as fraud detection tools in the financial market).

What cloud should have…

- Cope with large fluctuating loads
- Capacity planning
- Auto-scaling for unplanned events

Auto Scaling

- A pool of available resources that can be pulled or released on-demand and a control loop to monitor the system and decide in real time whether it needs to grow or shrink
- **PaaS** -

  Google App Engine and Heroku but applications developed specifically for these platforms are tied to them
- **IaaS** -

  appears more flexible since users are given free access to virtualized hardware, relying on providers like Amazon and Rackspace or open-source projects like OpenNebula and OpenStack to instantiate VM

Resource allocation and Policies

- Threshold-based policies, where upper and lower bounds on the performance trigger adaptations, and where some amount of resources are allocated or deallocated. (typically one VM at a time).

They Suggest,

- Sequential decision policies based on Markovian decision processes (MDP) models and computed using, for example, reinforcement learning

Resource allocation as an MDP

- Decision agent Repeatedly observes the current state s of the controlled system.
- Takes a decision '***a***' among the ones allowed in that state
- Then observes a transition to a new state ***s'***
- And reward ***r*** that will drive s' decisions.

MDP

The MDP that models our approach to the VM allocation problem is defined as $\mathcal{M} = \langle S, A, T, R, \beta \rangle$ where:

- $S = \{(w, u, p) \mid 0 \leq w \leq W_{max} \wedge 0 \leq u \leq U_{max} \wedge 0 \leq p \leq P_{max}\}$ is the state of the MDP where:
  - $w \in \mathbb{N}$ is the workload in number of requests per second, bounded by $W_{max} = 40$;
  - $u \in \mathbb{N}$ is the current number of homogeneous VMs allocated to the application, bounded by $U_{max} = 10$;
  - $p \in \mathbb{R}$ is the performance expressed as the average response time to requests in seconds, bounded by a value P_{max} chosen from experimental observations.
- $A = \{a \in \mathbb{Z} \mid A_{min} \leq a \leq A_{max}\}$ is the action set which consists in adding, maintaining or reducing the number of homogeneous VMs allocated to the application. The actions have been bounded between $A_{min} = -1$ and $A_{max} = 10$ in our experimental setup;
- $T : S \times A \times S \rightarrow [0, 1]$ is the probability distribution $P(s'|s, a)$ of a transition to new state s' given that the system is in state s and action a is chosen;
- $R : S \times A \rightarrow \mathbb{R}$ is the cost function expressing the expected reward when the system is in state s and action a is taken. When stochastic, it can be expressed as

  $R : S \times A \times \mathbb{R} \rightarrow [0, 1]$, the probability distribution $P(r|s, a)$ of observing a reward r when the system is in state s and action a is taken;
- $\beta, 0 < \beta < 1$ is a discount factor used to finitely evaluate the overall expected reward for an infinite sequence of decisions. The value $\beta = 0.45$ has been used throughout our experiments.

Q Learning Over DP

- *T* and *R* can be determined prior to the execution of the controlled system, using traditional dynamic programming (DP) algorithms, such as value iteration in an optimal way .
- The advantage of traditional DP algorithms is that policies are computed offline.

BUT,

- The decision-making at runtime then simply amounts to applying the pre-computed policy $\pi *$ to the sequence of observed states to provide the corresponding actions.
- *T* and *R* are often very difficult to estimate. This can require lengthy experimentation and measurement processes upon the actual controlled system and it must be redone each time a modification to the system may change the probability distributions of its transitions or rewards.

Q Learning

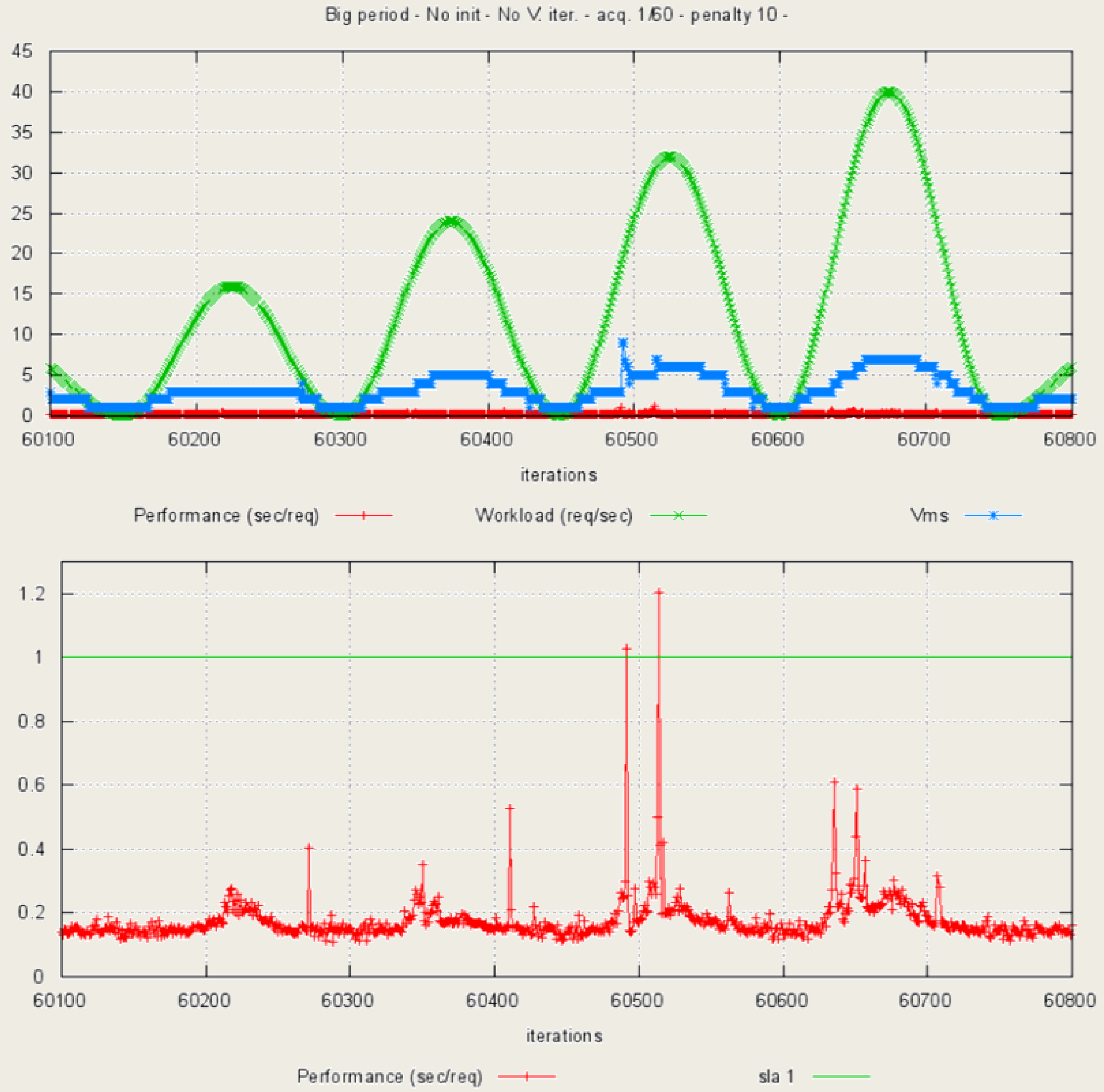
- This can update its estimation of the Q-function for state s and action a with:

$$Q[s,a] := (1-\alpha)Q[s,a] + \alpha\left(r + \beta \max_{a} Q[s',a']\right) \qquad (3)$$

where α is the rate of learning, balancing the weight of what has already been learned with the weight of the new observation. Throughout our experiments, we have used the value $\alpha = 0.8$. The basic Q-learning algorithm is then [2]:

$(\forall s \in S)(\forall a \in A(s)),$ initialize $Q(s,a)$
$s :=$ the initial observed state

Big Period


Big period - No init - No V. iter. - acq. 1/60 - penalty 10 -
45
40
35
30
25
20
15
10
5
0
60100
60200
60300
60400
60500
60600
60700
60800
iterations
Performance (sec/req)
Workload (req/sec)
Vms
1.2
1
0.8
0.6
0.4
0.2
0
60100
60200
60300
60400
60500
60600
60700
60800
iterations
Performance (sec/req)
sla 1

Small Period

Virtual reinforcement learning workflow

Besides the basic learning algorithm, the VirtRL workflow introduces three new activities as:

- Initialization of the Q-function;
- Convergence speedup phases at regular intervals of observations;
- Performance model change detection.

Future work

- Applying Reinforcement learning in the context of a larger scale workflow, where clouds could gain information from applications to applications in order to make the techniques much more successful.
- Experiment on higher level descriptions of applications and their need for adaptation in order to select from past applications learned policies from which the learning can be initialized more accurately.

However…

- Reinforcement learning is a promising approach towards an autonomic solution to the problem of dynamically adapting the amount of resources allocated to applications in cloud environments

Thank you…