A Reinforcement Learning Approach to Virtual Machines Auto-configuration

Jia Rao, Xiangping Bu, Cheng-Zhong Xu
Le Yi Wang and George Yin

Wayne State University
Detroit, Michigan
http://www.cic.eng.wayne.edu

Why VM Autoconfiguration?

- Server consolidation is a primary usage of virtualization to reduce TCO
- In cloud systems, virtual machines need to be configured on-demand, in real time
- VMs need to be reconfigured dynamically
  - Created from template
  - Migrated to a new host
  - Resource demands/supplies vary with

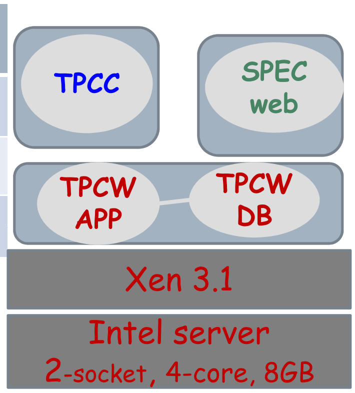
Challenges in Online Autoconfig

- A rich set of configurable parameters
  - CPU, memory, I/O bandwidth, etc
- Heterogonous applications in the same physical platform
  - Hungry for different types of resources
- Performance interference between VMs
  - Centralized virtualization layer
- Delayed effect of reconfiguration
- Scale and real-time requirements make it even harder

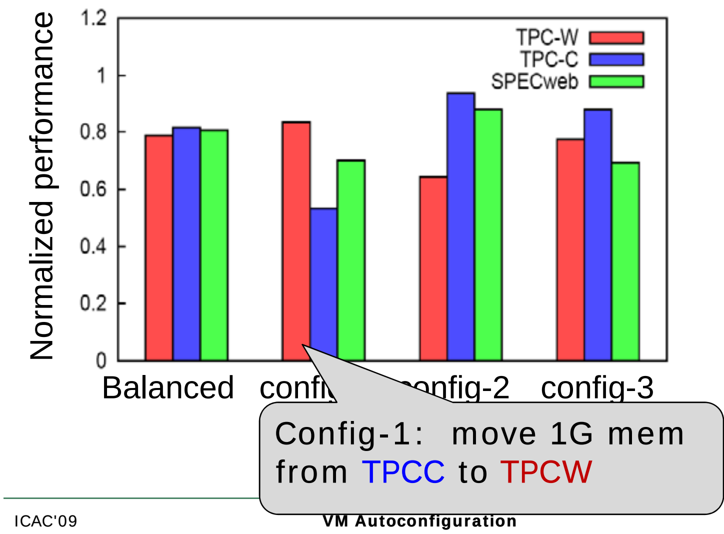
Performance Interference

- Balanced configurations

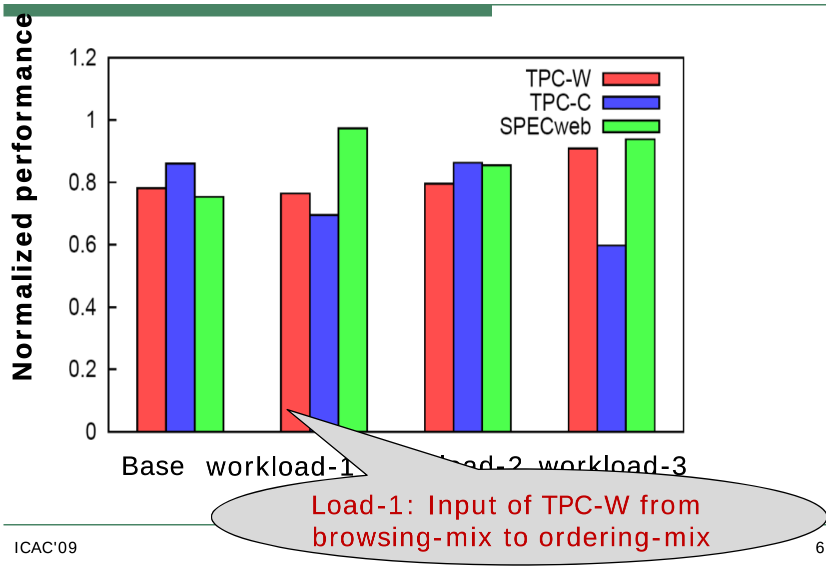
WLoad	weight	vcpu	mem
TPC-W	256	2	0.5GB
TPC-C	256	1	1.5GB
SPECweb	512	2	0.5GB



Performance Interference (cont')

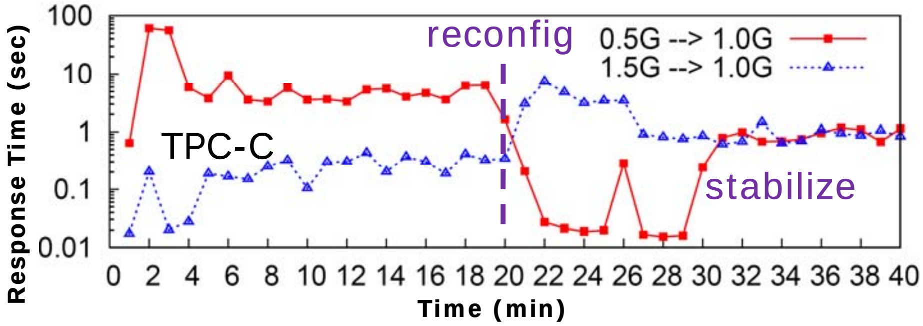


Perf Interference (cont')



Delayed Effect of Reconfiguration

- Reconfig takes effect after certain delays
- TPCC benchmark on two VMs of same conf, except mem



Problem Statement

- For VMs to be run on the same physical platform, automate the resource (re)configuration process:
  - Optimize system-wide perf (utility func) under individual SLA constraints.
  - SLA w. r.t. throughput or response time
- Multi-resources
  - CPU time (weighted scheduling in Xen), #virtual CPUs, Memory
- Work-conserving in resource allocation

Related Work

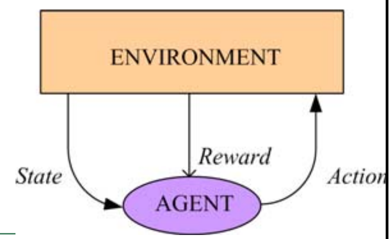
- [Wildstrom08] Regression based value est for memory
  - Single memory resource, supervised learning, not adaptive
- [Soror08] Greedy search based config enumeration for database workloads
  - Single CPU resource, assumes independent calibration of different resources
- [Padala07, Padala09] Control theory based alloc of resources (CPU, i/o bw)
  - Assumes no interference between VMs due to the use of non-work-conserving mode
  - Delayed effect of memory config is not considered

Our Contribution

- A reinforcement learning approach for online auto-configuration of multiple resources (includes memory)
- Consideration of VM interferences in work-conserving mode
- Consideration of delayed effect in resource allocation
- Prototyped in a VCONF framework and tested in real world applications

Reinforcement Learning

- Learning by interaction with env
  - State: configuration of VMs (cpu, mem, time, etc)
  - Action: reconfiguration (increase/decrease/nop of resrc)
  - Immediate reward: w.r.t. response time or throughput
- Learning Objective
  - For a given state, find an action policy that would maximize long-run return



Reinforcement Learning (cont')

- An optimal policy π^* is to select the action a in each state s that maximizes cumulative reward r

$$Q^{\pi^*}(s_t, a_t)=r_0+\gamma r_1+\gamma^2 r_2+\ldots \quad (0\leq\gamma<1)$$

- An RL solution is to obtain good estimations of $Q(s_t,a_t)$ based on interactions: (s_t, a_t, r_{t+1})
- $Q(s_t,a_t)$ of each state-action pair is updated each time an interaction finishes:

$$Q(s_t, a_t)=Q(s_t, a_t)+\alpha[r_{t+1}+\gamma^*Q(s_{t+1}, a_{t+1})-Q(s_t, a_t)]$$

RL for Autoconfiguration

- **State space**: (mem_1, $weight_1$, $vcpu_1$,…,mem_n, $weight_n$, $vcpu_n$)
- **Action set**: *Inc*, *Dec*, and *Nop on* each resource
- **Rewards**: summarized perf over hosting applications
  - Score each VM based on normalized perf

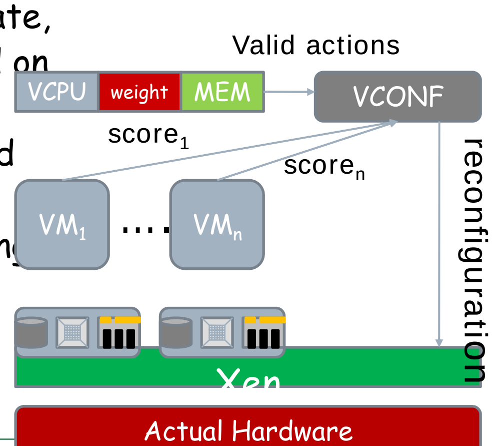
$$reward = \begin{cases} \sqrt[n]{\prod_{i=1}^{n} w_i * score_i} & if \;\; for\ all\ score_i > 0; \\ -1 & otherwise \end{cases}$$

$$score = \frac{thrpt}{ref\_thrpt} - penalty$$

$$penalty = \begin{cases} 0, if \;\; resp \le SLA; \\ \frac{resp}{SLA}, if \;\; resp > SLA. \end{cases}$$

VCONF Architecture

- VCONF resides in dom0
- Observes current state, makes decision based on Q(s, a) table
- Monitors VM perf and calculate rewards
- Updates corresponding entry in Q(s, a) table



Adaptability and Scalability

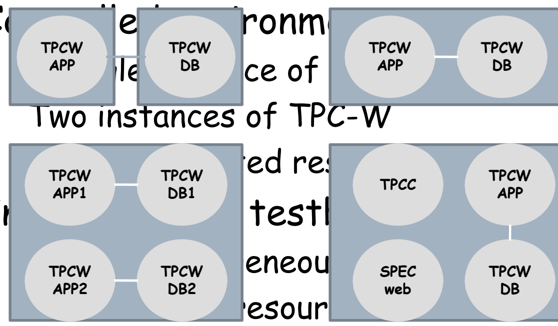
- Trivial implementation would lead to poor adaptability and scalability
- Adaptability
  - Revise existing policy when environment changes
  - Poor adaptability due to slow start
- Scalability
  - The size of the Q(s,a) table grows exponentially with the state variables

Model-based RL and Function Approx

- Build env models from collected traces
  - $(s_t, a_t) \rightarrow r_t$
  - Batch update Q(s, a) using simulated interactions from the models
  - Continuously update the models with new traces
  - Model-based RL is more data efficient
  - Model reuse when similar resource demands detected
- Replace look-up table based Q with neural network based function approximation

Experimental Results

- Settings
  - SPECweb, TPC-C, TPC-W as applications
  - Xen vm ver3.1 on 2-socket quad-core CPU
- C e ronm
  - le ce of
  - Two instances of TPC-W
  - ed re ed
- I test
  - eneou
  - esour ered



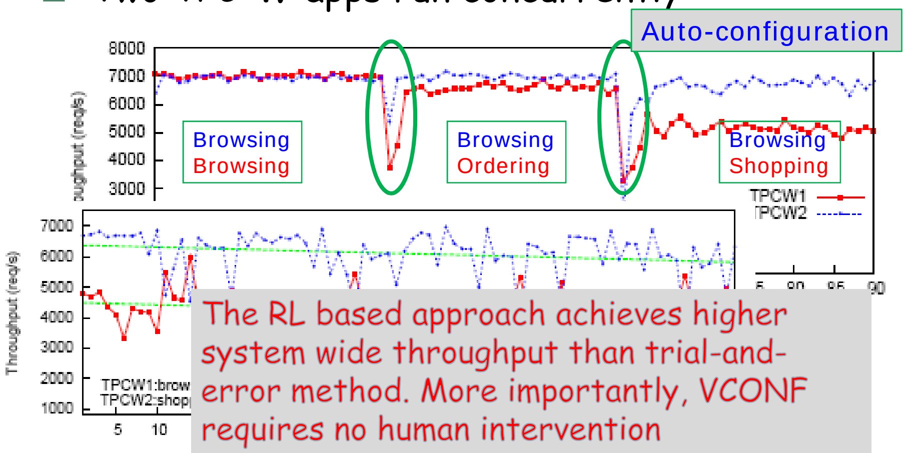
Single Application Auto-configuration

- Optimizing application throughput

RL based approach leads to optimal configurations, but without adaptive policy, RL shows poor adaptability even in small scale problems

System-Wide Perf Optimization

- Two TPC-W apps run concurrently



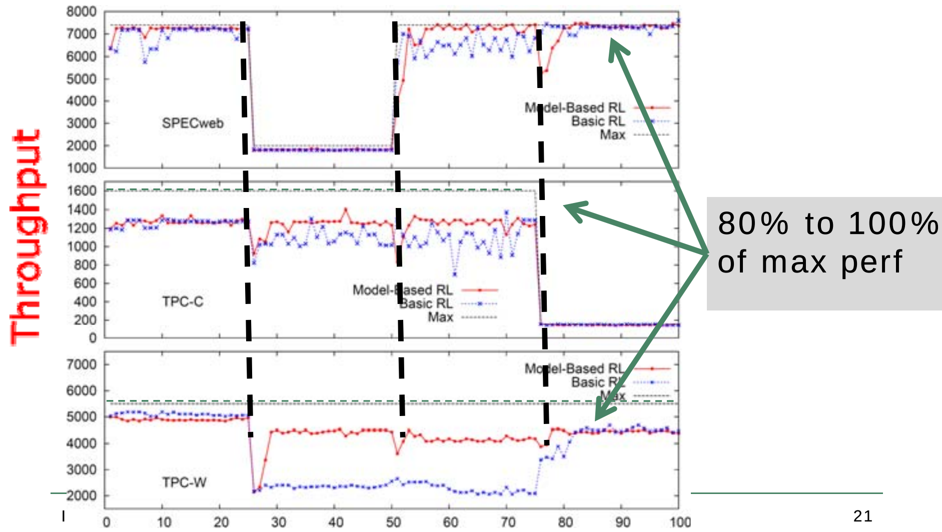
The RL based approach achieves higher system wide throughput than trial-and-error method. More importantly, VCONF requires no human intervention

VM Auto-configuration in Clouds

- Heterogeneous VMs
- Large problem size
  - More VMs, more resources considered

	TPC-W	TPC-C	SPECweb
Workload-1	600 browsing	50 warehouses, 10 terminals	800 banking
Workload-2	600 ordering	50 warehouses, 10 terminals	800 banking
Workload-3	600 browsing	50 warehouses, 1 terminals	800 banking
Workload-4	600 browsing	50 warehouses, 10 terminals	200 banking

Coordinated Auto-Reconfiguration



Conclusion

- VCONF shows the applicability of RL algorithms in VM auto-configuration
  - RL-based agent is able to obtain optimal (near-optimal) policies in small scale problems
  - In clouds with a large problem size, model-based RL shows better adaptability and scalability
- Future work
  - Consider more resources, such as I/O, shared cache
  - Integrate migration as an additional dimension in the RL framework
- Auto-configuration of appliances in clouds [see ICDCS'09]

Thank you !

Questions?

Cloud and Internet Computing Laboratory
Wayne State University, Michigan
http:/cic.eng.wayne.edu

Cheng-Zhong Xu, czxu@wayne.edu