Twinkle: A Fast Resource Provisioning Mechanism for Internet Services

Professor Zhen Xiao
Dept. of Computer Science
Peking University
xiaozhen@pku.edu.cn

Joint work with Jun Zhu and Zhefu Jiang

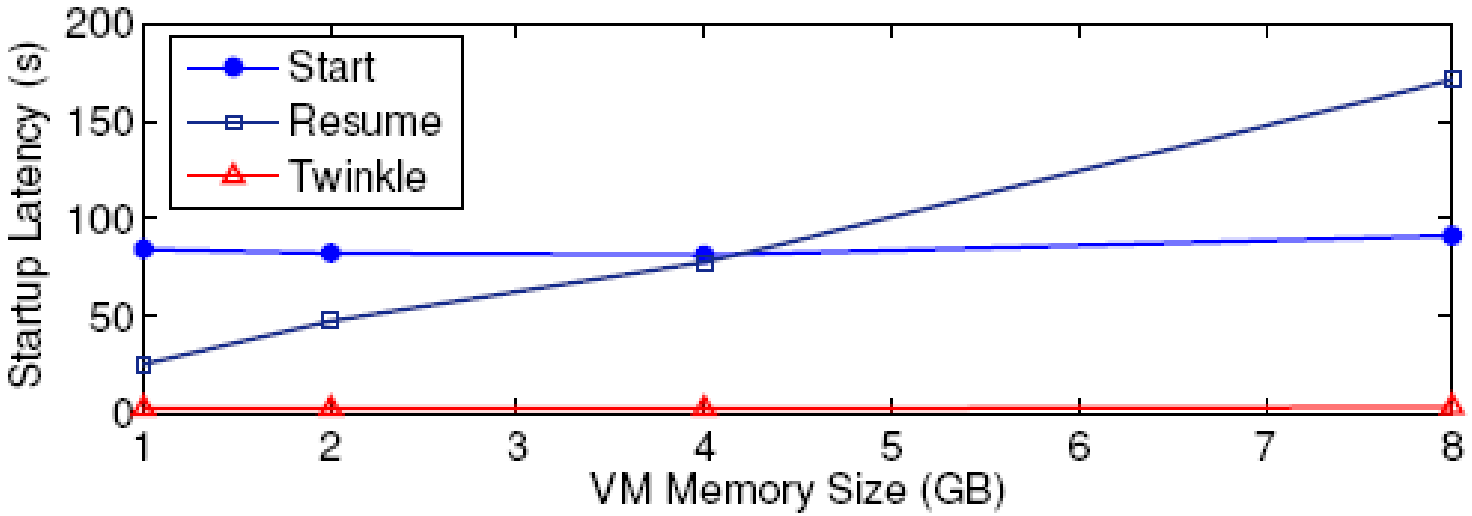
Motivation

- Auto scaling is a key property of cloud computing
  - Flash crowd, fail over, etc.
- Two decisions need to be made
  - when and where to start a VM for an application?
  - how can the VMs and the applications inside be brought up as quickly as possible?
- Unfortunately, start up latency can be really long!

Why Slow?

- VM Start
  - Virtual machine allocation, operating system startup and application initialization.
  - Complicated application: WebLogic, JBoss, etc.
- VM Resume
  - Virtual machine allocation and loading virtual machine state.
  - VM state is large: Amazon EC2 allows high throughput VM instances with tens of Gigabytes of memory.

Startup Latency



Startup latency of RUBiS service within JBoss which runs on Dell PowerEdge blade servers with Intel 5620 CPU, 24 GB RAM and10K RPM SAS disks.

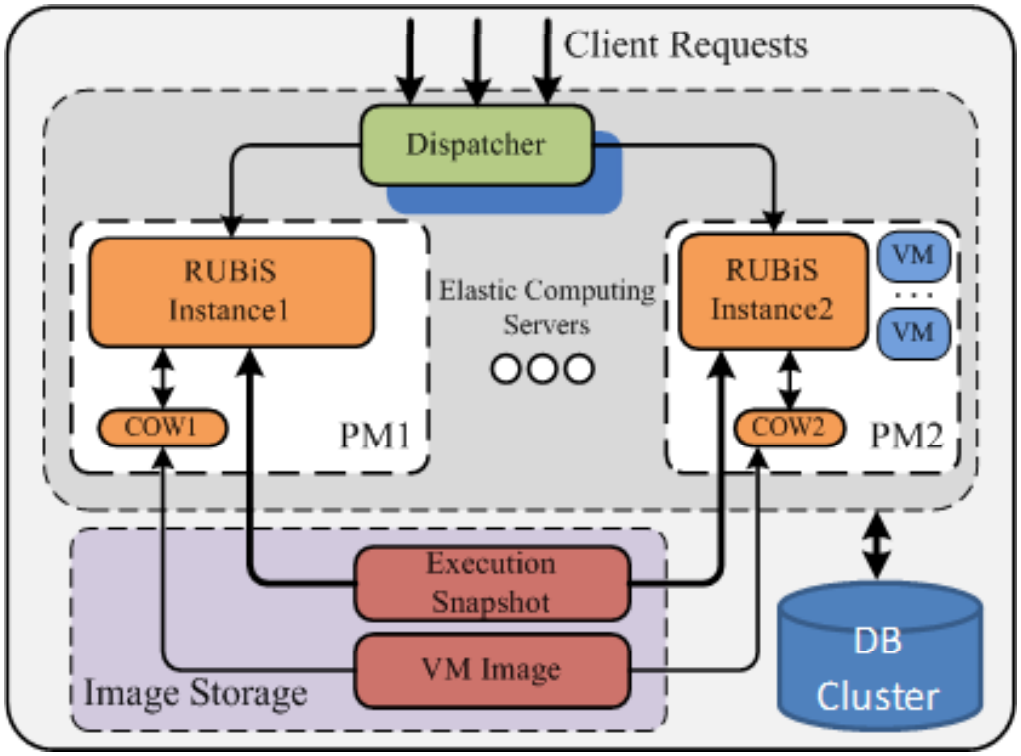
Our Approach

- VM instances are started from an initialized VM snapshot.
- Working set estimation, demand prediction and free page avoidance accelerate VM startup.

Outline

- **Architecture**
- Three optimizations
- File system snapshot
- Experiments
- Related work
- Conclusion

Architecture



A cloud data center employing fast restart mechanism.

- Image Storage
  - Execution Snapshot
  - VM Image
- Elastic Computing Servers
  - Stateless VMs
- Dispatcher
  - Policy Decision
  - Request Dispatching
- Storage
  - Database, S3, etc.

Outline

- Architecture
- **Three optimizations**
- File system snapshot
- Experiments
- Related work
- Conclusion

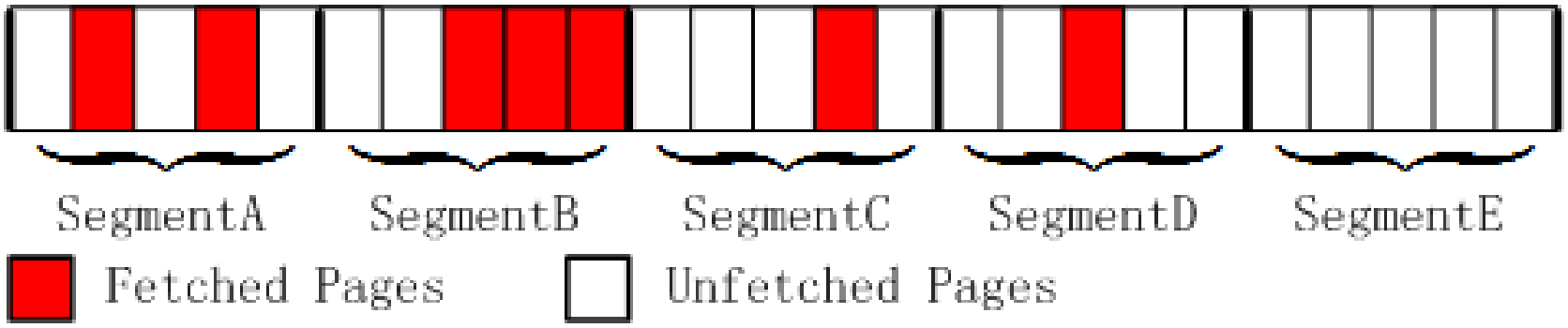
Overview

- Step 1: VMM allocates a new VM container.
- Step 2: Working set of the guest OS is loaded.
- Step 3: The guest OS starts running.
- Step 4: The other state of the VM is lazily fetched.

Working Set Estimation

- We load working set of the operating system before starting the VM.
- Post-checkpoint tracking mechanism is employed to estimate working set.
- How to track the working set?
  - After checkpoint, we mark all the nested page table entries as “non-present”.
  - The pages inducing page faults are recognized as working set.
  - We adopt an iterative and heuristic method to decide tracking window.

Demand Prediction



- Demand prediction fetches the pages that are most possibly needed in parallel with the VM execution.
- Our approach follows the principle of memory access locality.
- The address space of the newly started VM is divided into segments, each of which contains N consecutive pages.
- If the number of demand-page-traps exceeds P pages in one segment, we fetch the remaining pages in advance.

Free Page Avoidance

- For a typical Internet service, the majority of memory consumption results from dealing with client requests.
- The memory requirement of the service itself is small.

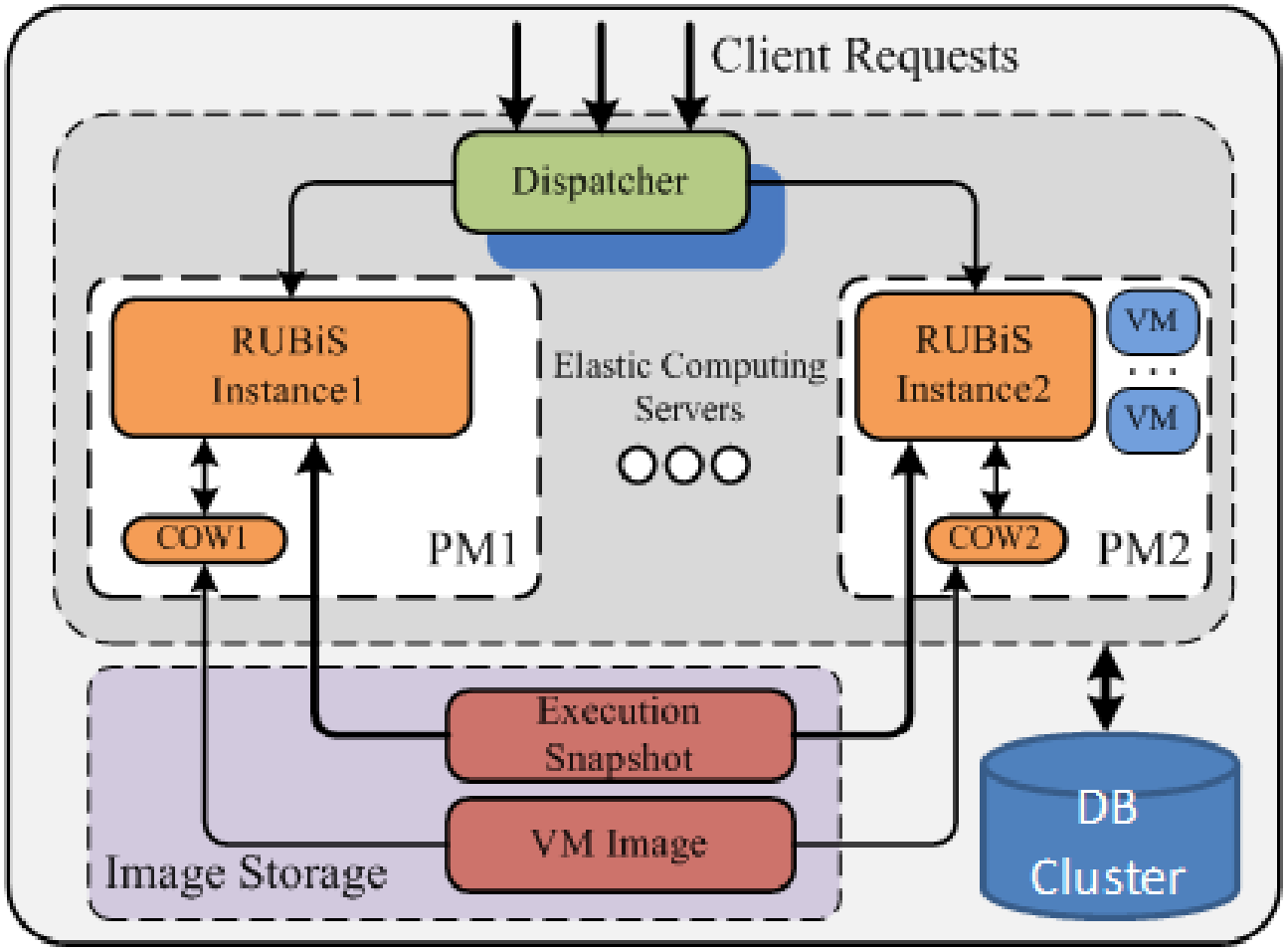
Outline

- Architecture
- Three optimizations
- **File system snapshot**
- Experiments
- Related work
- Conclusion

File System Snapshot

- Each Internet service holds one prepared VM snapshot: an execution snapshot and a root file system image.
- Modifications to the root file system are saved on the local storage using copy-on-write.
- The local state can be discarded when the VM instances shut down later.

Architecture

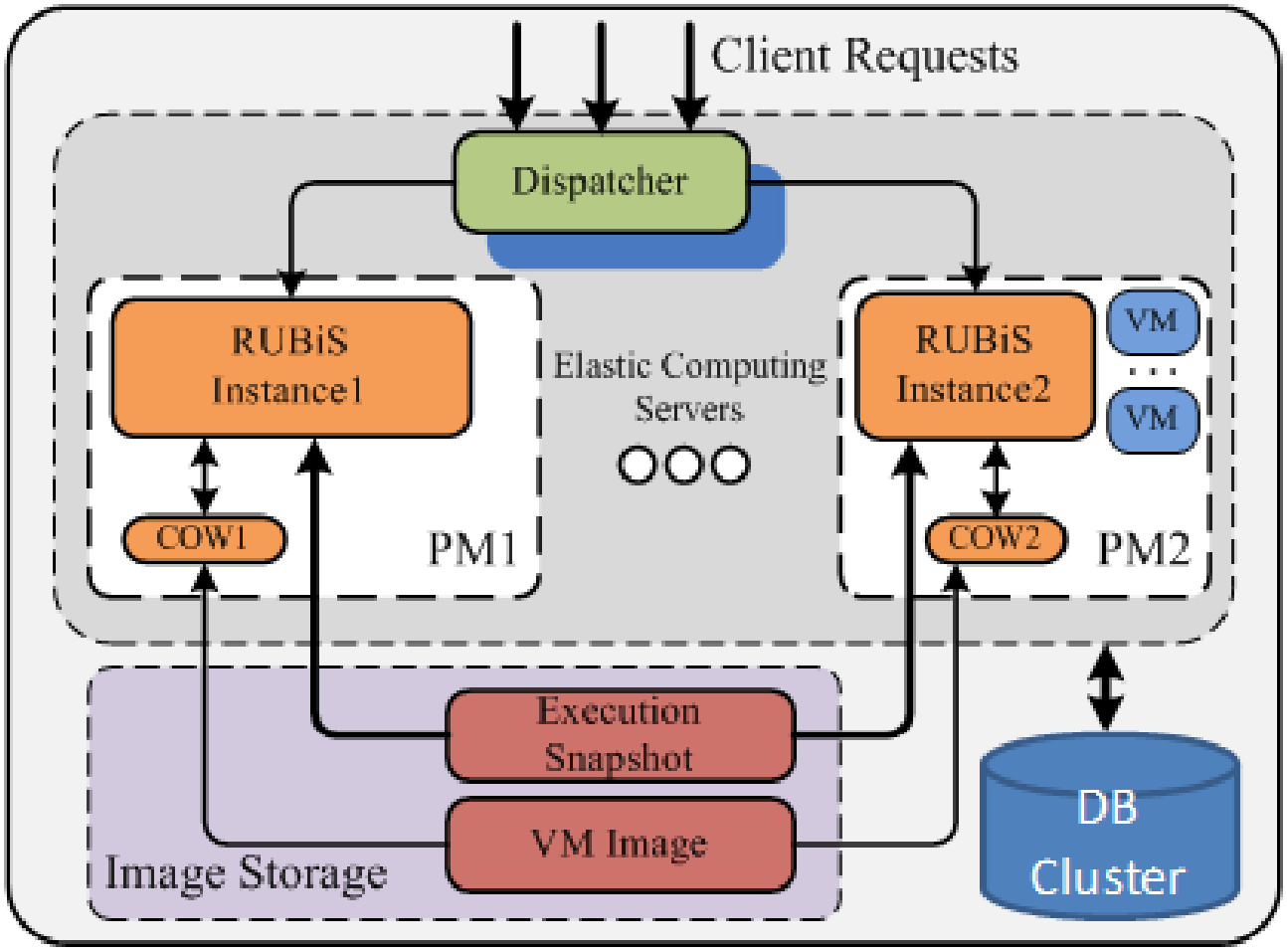


Example of a cloud data center employing fast restart mechanism.

Outline

- Architecture
- Three optimizations
- File system snapshot
- **Experiments**
- Related work
- Conclusion

Architecture



Example of a cloud data center employing fast restart mechanism.

Experiments

- Performance Evaluation (SPEC CPU2006)
  - Startup latency: the time to fetch the working set
  - Invalid execution ratio that is the percentage of CPU time when the VM is inactive
  - Remote page fault: mostly determines the performance of the newly started VM
- Application Evaluation (RUBiS and TPC-W)
  - Flash crowds with single and multiple applications
  - Failure over

Performance Evaluation

- We run the six programs from SPEC CPU2006 in separate VMs for two minutes, and then take a snapshot of each VM.
- We continue the execution of the VM from the snapshot with three approaches:
  - Xen's native resume as baseline
  - Demand paging without optimization
  - Twinkle startup

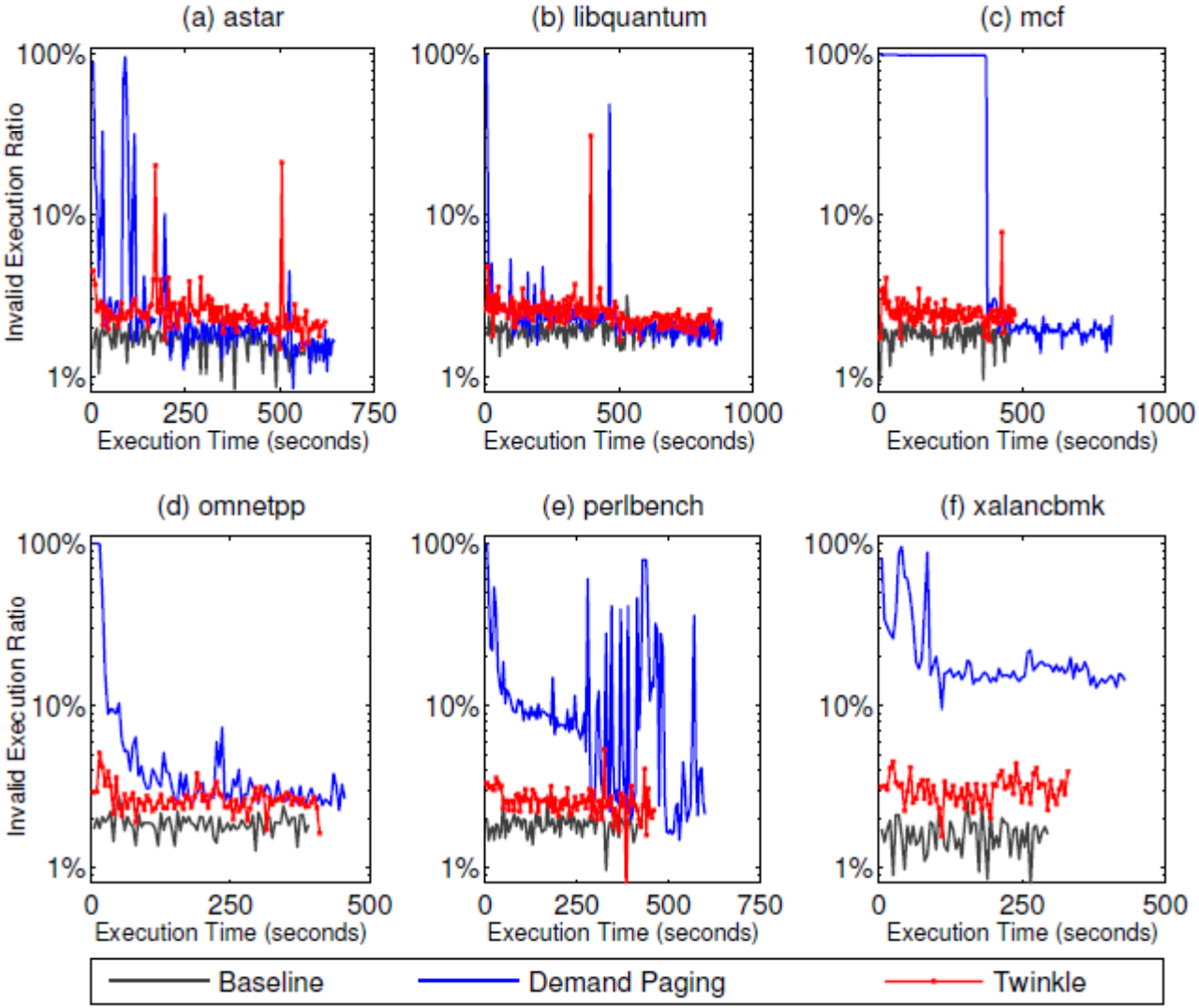
Startup Latency of SPEC CPU2006

Benchmarks	Startup Latency (Seconds)
astar	1.49 (1.8%)
libquantum	0.35 (0.4%)
mcf	11.89 (14.4%)
omnetpp	1.31 (1.6%)
perlbench	2.25 (2.7%)
xalancbmk	3.24 (3.9%)

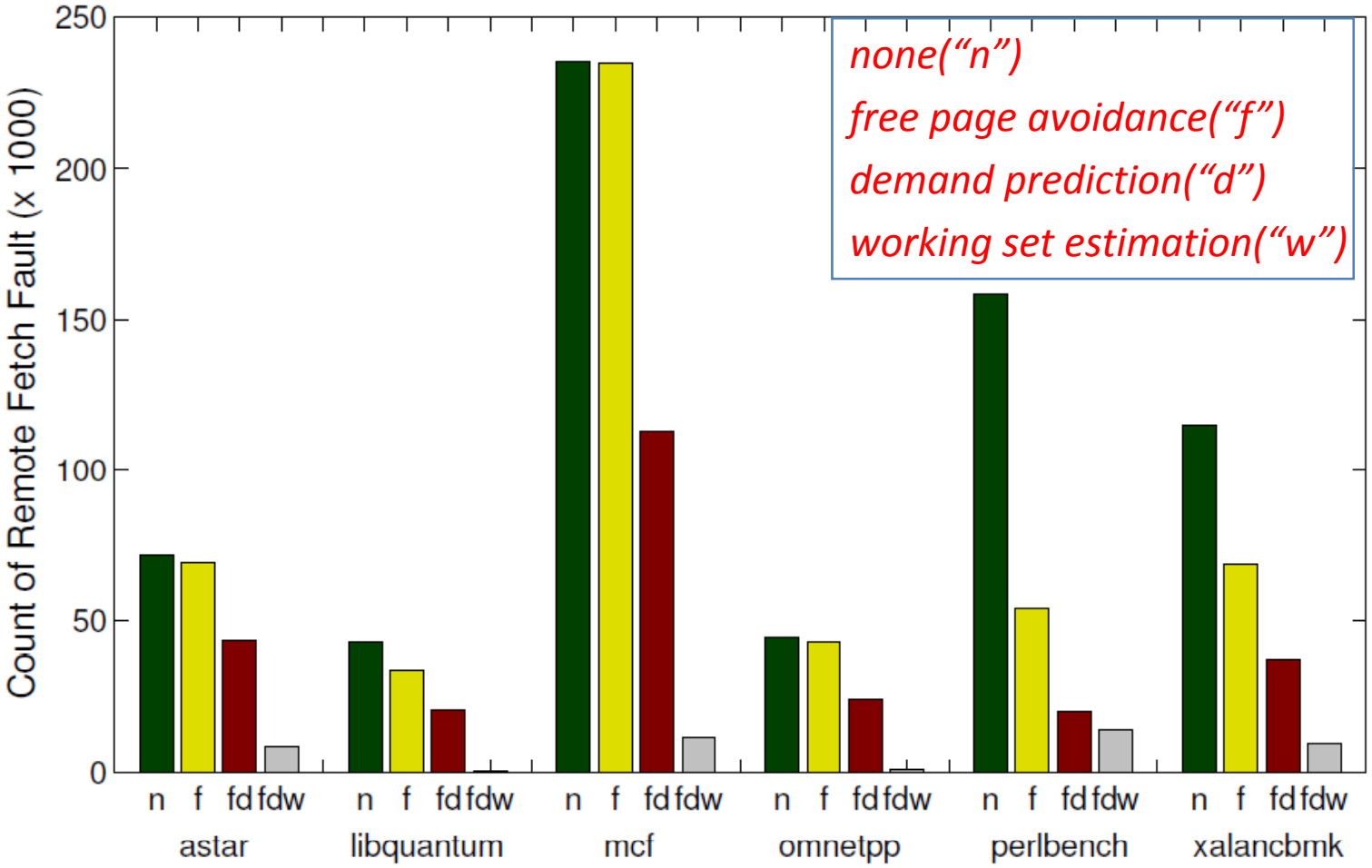
Execution Time of SPEC CPU2006

Benchmarks	Baseline	Demand Paging	Twinkle
astar	600.95	639.56(6.4%)	608.31(1.2%)
libquantum	830.57	863.28(3.9%)	841.57(1.3%)
mcf	455.84	818.93(79.7%)	470.15(3.1%)
omnetpp	383.99	459.74(19.7%)	399.75.04(4.1%)
perlbench	411.30	575.04(39.8%)	460.40(11.2%)
xalancbmk	283.27	425.34(50.2%)	325.71(11.5%)

VM Invalid Execution Ratio



Individual Contribution

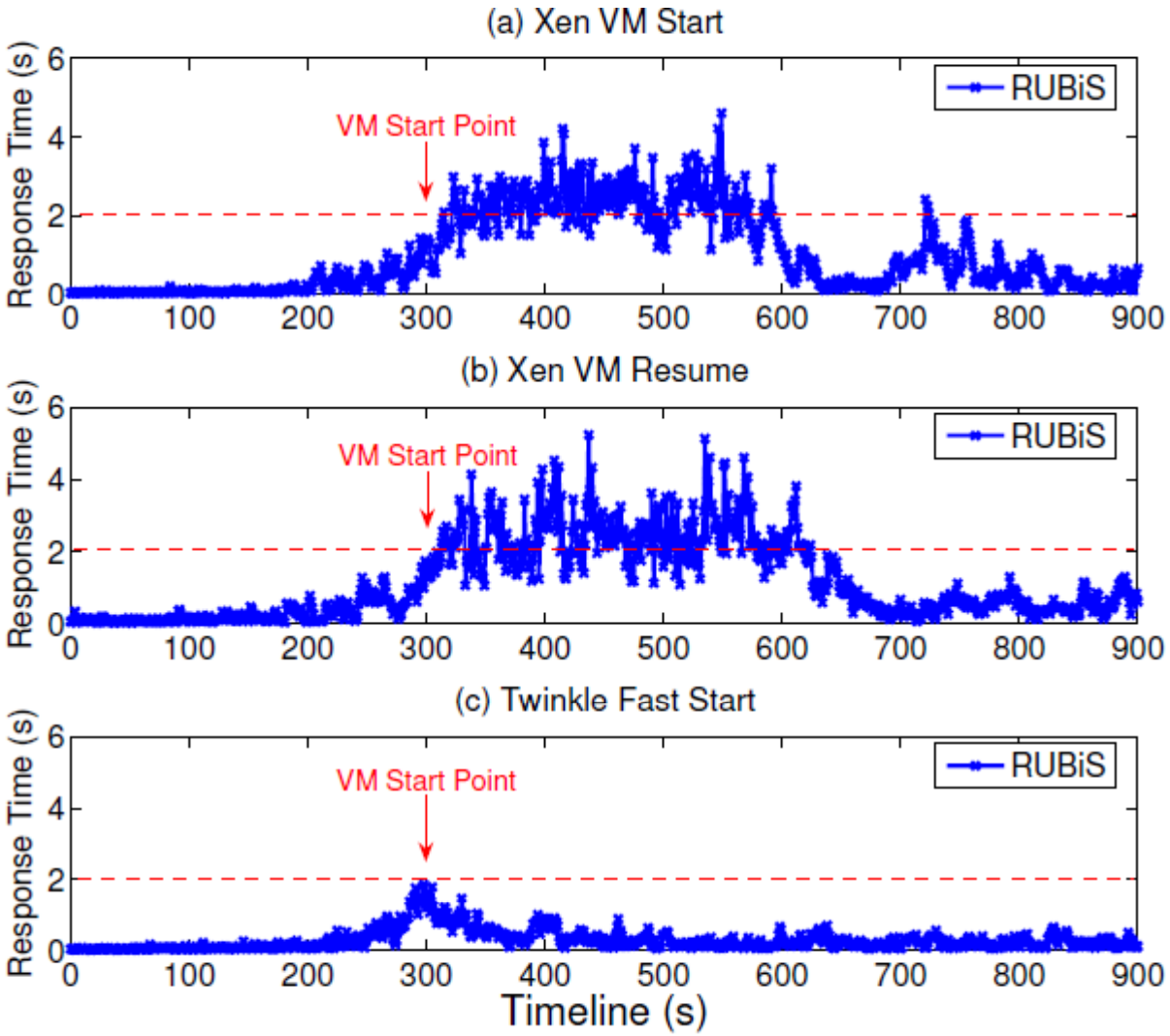


The count of remote page faults under different combinations of techniques.

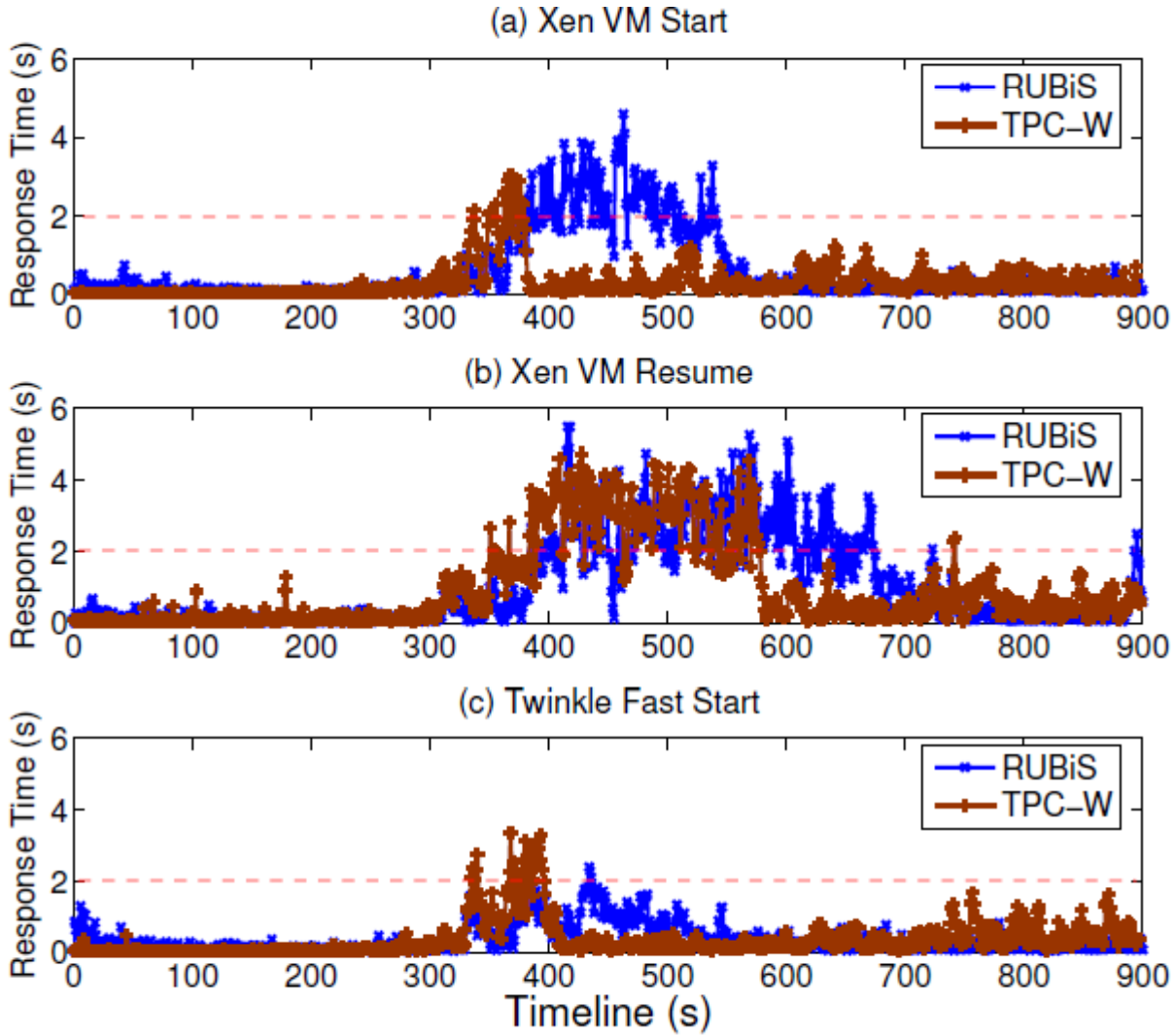
Application Evaluation

- Policy: Provision more VMs for the service when response time exceeds 2000 msec.
- Three approaches to start a new VM:
  - Xen VM Start: Start a VM from scratch
  - Xen VM Resume: Resume a VM with Xen's resume functionality
  - Twinkle Fast Start: Our approach

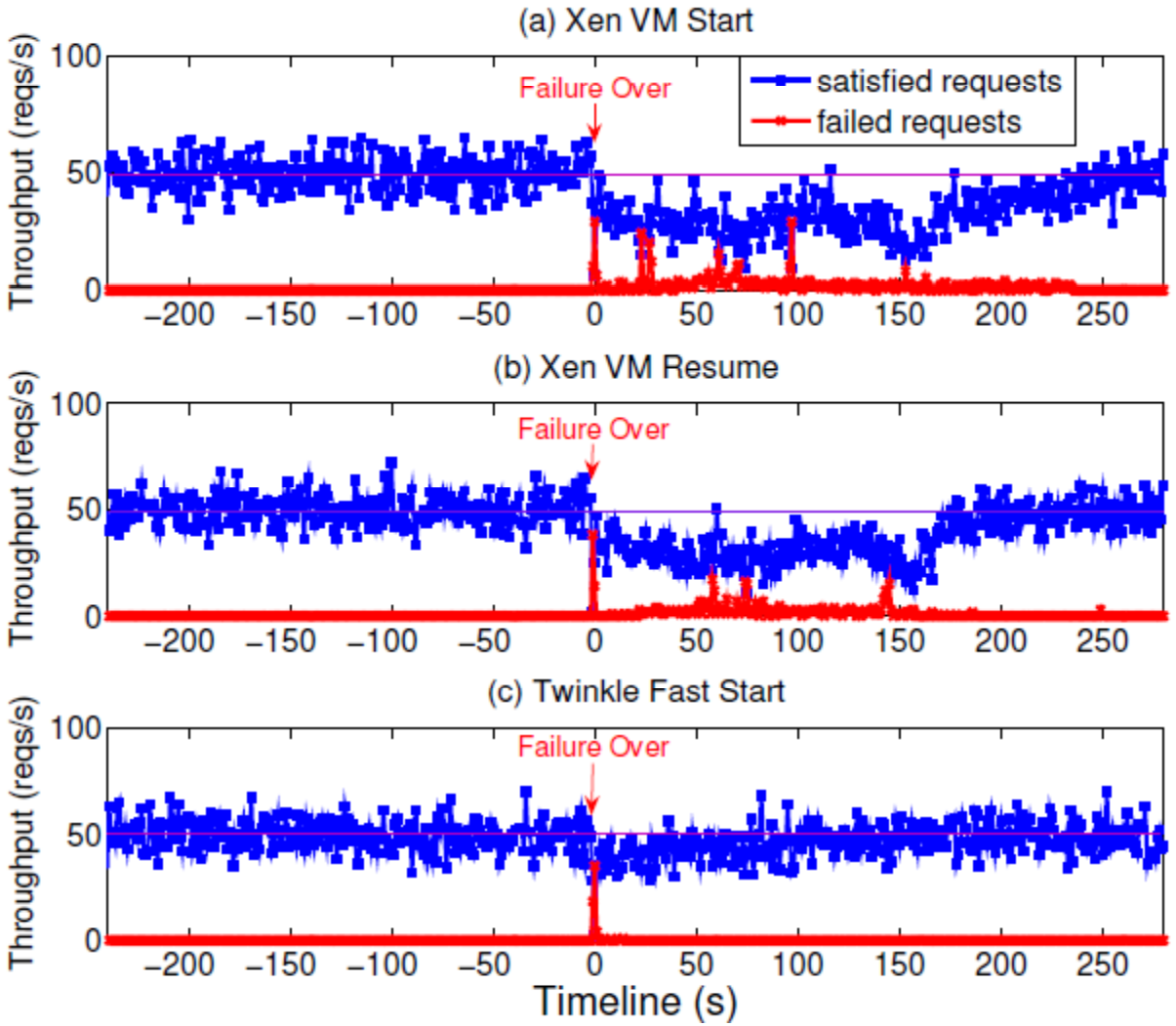
Response Time of RUBiS service during a flash crowd



Response Time of RUBiS and TPC-W during a flash crowd



Throughput of RUBiS in case of a failure over



Outline

- Architecture
- Three optimizations
- File system snapshot
- Experiments
- **Related work**
- Conclusion

Related Work

- Auto Scaling
  - **Rightscale**, **Scalr** and **EC2** focus on policy decision of when and where to start/stop virtual machines.
  - Twinkle provides a fast resource provisioning mechanism which executes these decisions more efficiently once they are made.

Related Work

- Virtual Machine Replication
  - **Collective** aims at a low-bandwidth network and begins execution after all the memory has been loaded.
  - **Potemkin** spawns virtual machines with memory copy-on-write techniques and does not fetch memory pages via a network environment.
  - **SnowFlock** enables cloning virtual machines on-the-fly for computation intensive applications and needs guest os intrusion.

Outline

- Architecture
- Three optimizations
- File system snapshot
- Experiments
- Related work
- **Conclusion**

Conclusion

- Auto scaling is essential to cloud computing service
- We presented Twinkie, a fast resource provisioning system without noticeable performance overhead

Thank You!