Kaleidoscope: Cloud Micro-Elasticity via VM State Coloring

Roy Bryant[*],

Alexey Tumanov[*],

Olga Irzak[*], Adin Scannell[*],

Kaustubh Joshi[†], Matti Hiltunen[†],

Andres Lagar-Cavilla[†], Eyal de Lara[*]

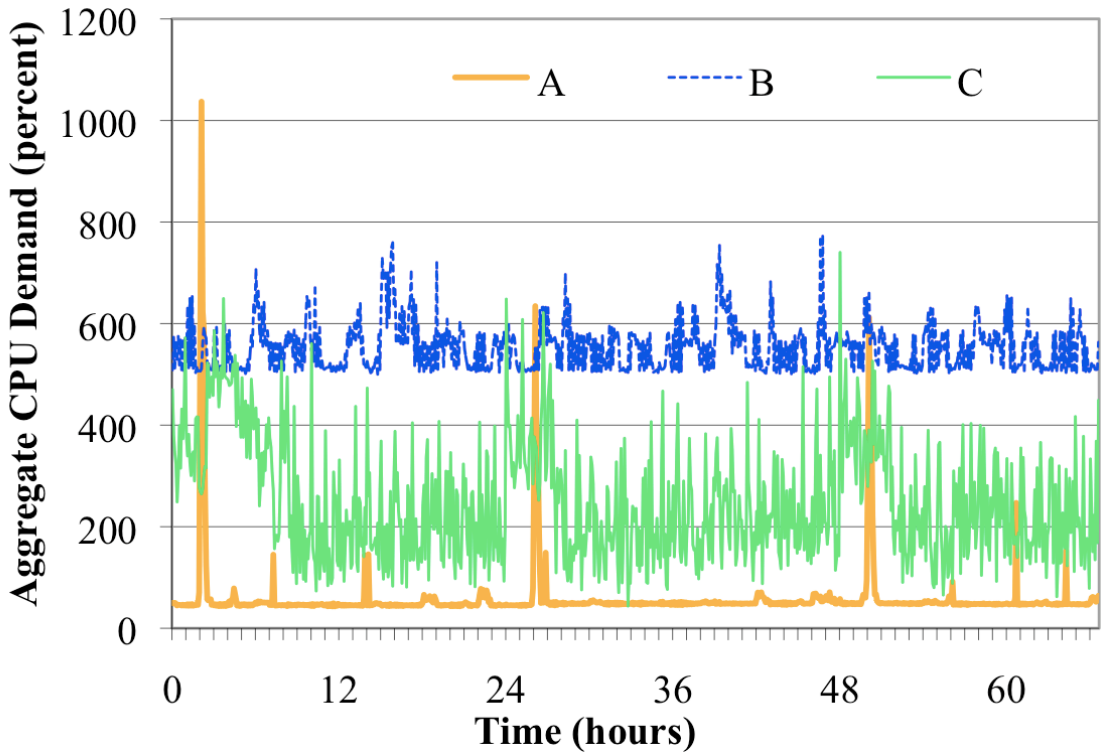
[*]University of Toronto [†]AT&T Labs Research

Outline

- Today's Clouds
- Techniques for a Better Cloud
- Evaluating Kaleidoscope
- Conclusions

Internet of bursty loads

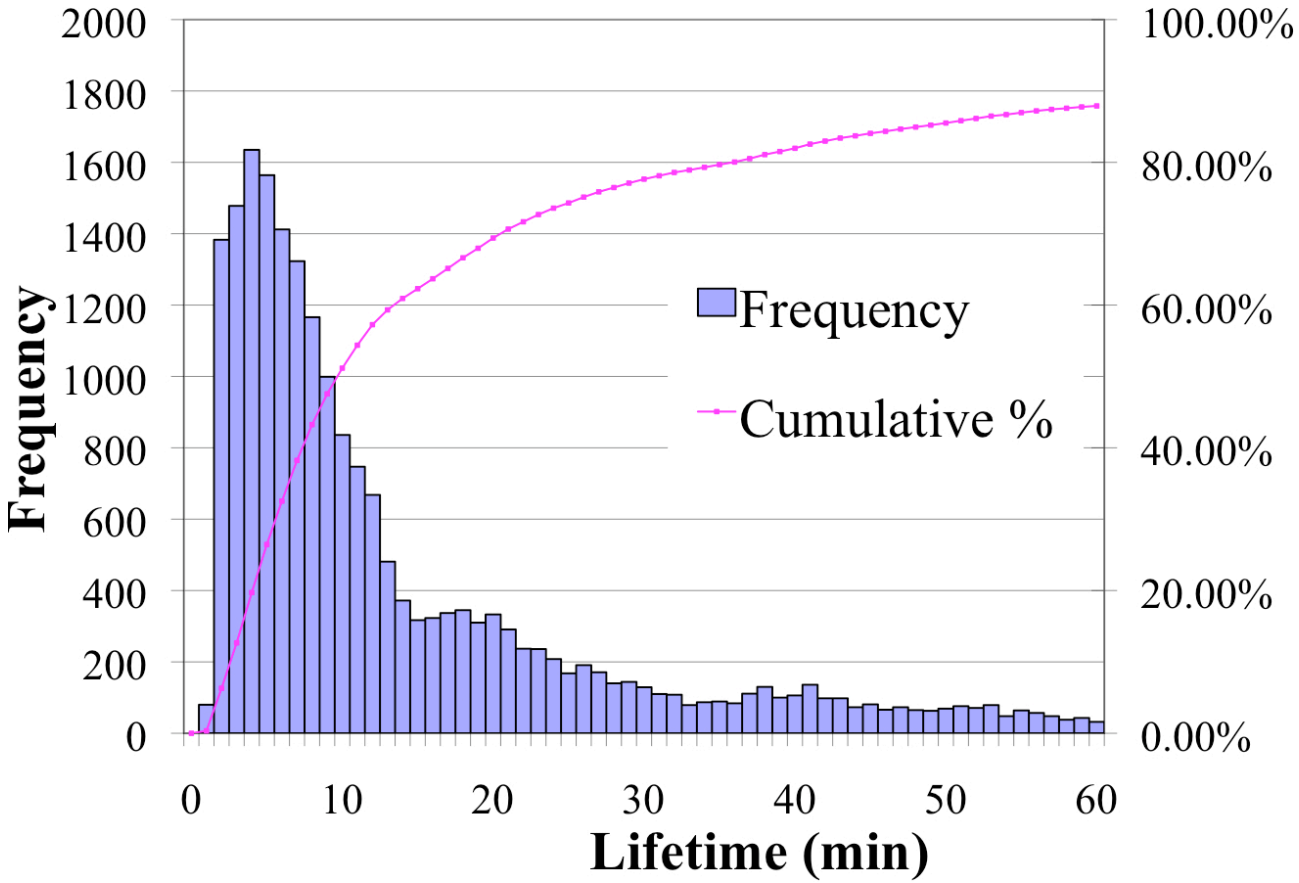
- Loads fluctuate sharply over time
- Pay-as-you-go IaaS clouds should be ideal



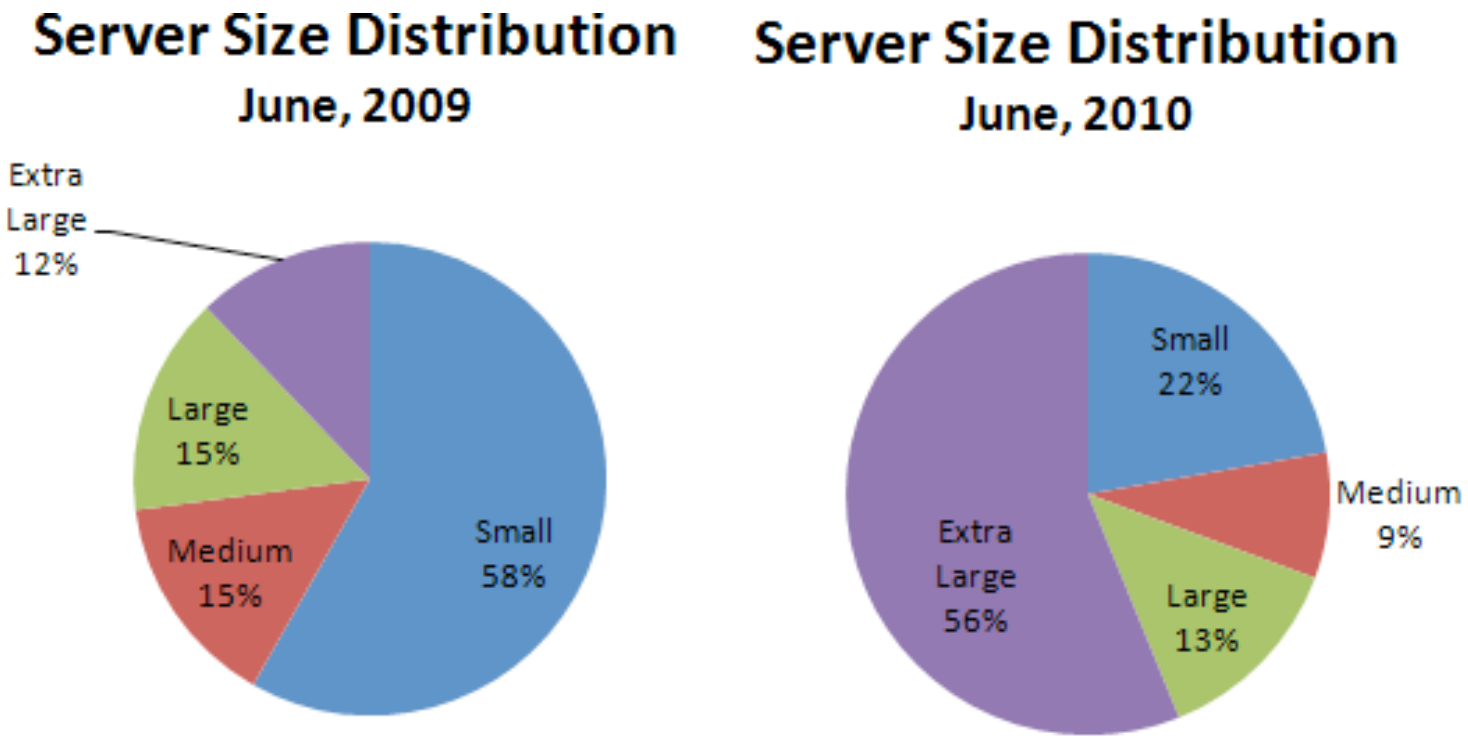
Sample loads from AT&T hosting service

Agile elasticity, tightly coupled

- AT&T's hosting data in a simulated agile cloud
  - Median lifetime ~ 10 minutes, 85% < 1 hour



But clouds becoming less elastic



- 2x increase servers living > month (EC2)
- 3x more servers, but 14x more Extra Large

Why? Too late. Too cold.

- VMs are a powerful abstraction, but…
  - VMs big, heavy, slow to boot
  - Cold workers have poor QoS

Users hoard their warm caches

Outline

- Today's Clouds
- Techniques for a Better Cloud
- Evaluating Kaleidoscope
- Implications and Conclusions

A Micro-Elastic Cloud

- New, pre-warmed workers in seconds
- Fractional VMs for less waste
- Want a better QoS from within a smaller footprint

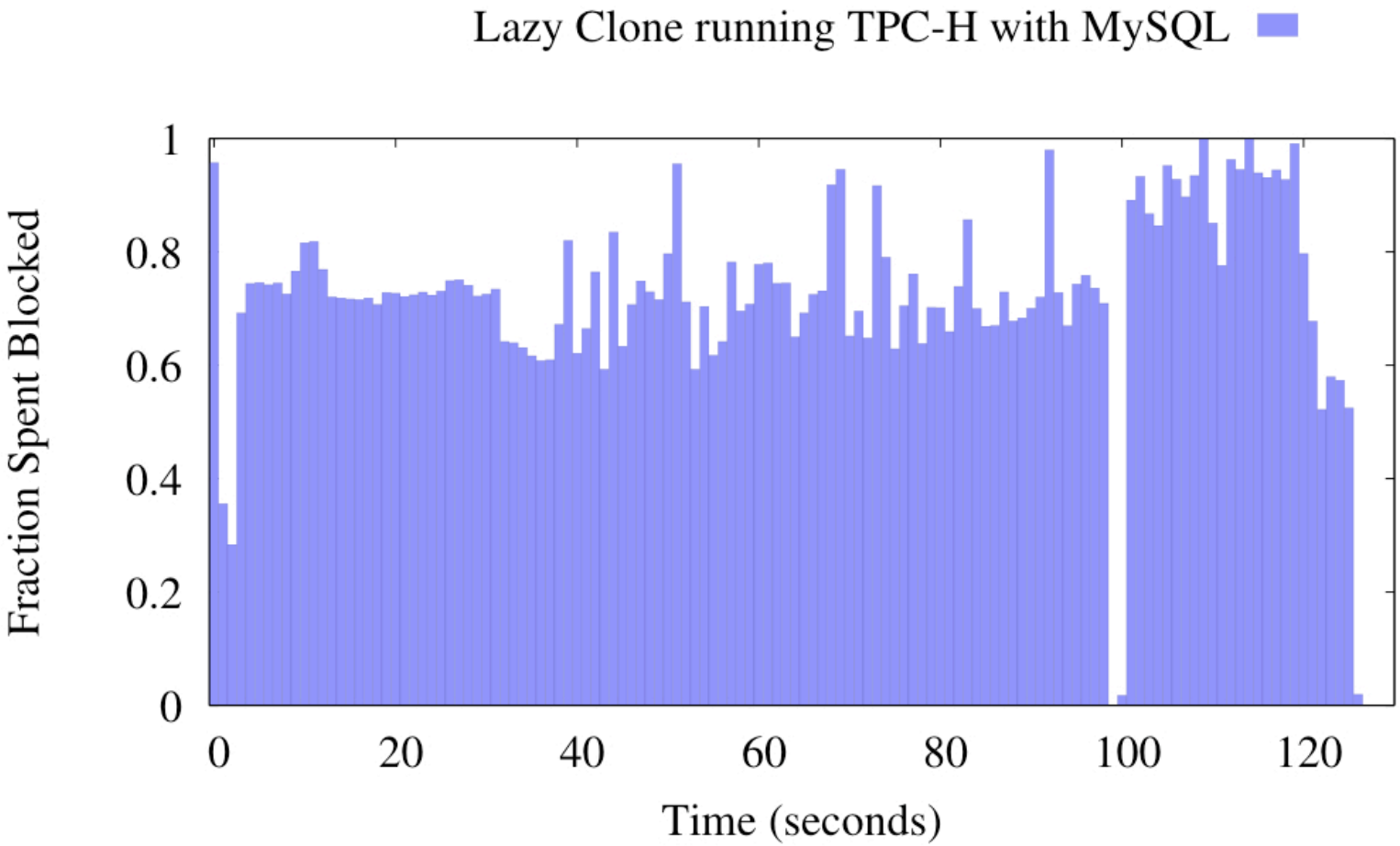
- Need 3 key techniques
  - VM cloning
  - VM state coloring
  - VM page allocation on demand

Background: VM cloning

- Prior work: SnowFlock (Lagar-Cavilla '09)
- Multiple clones, identical copies of running parent
  - Inherit the parent's warm caches
- Fast instantiation through lazy state propagation
  - Clones start empty
  - Fault on each missing page, fetch from parent
  - Perform well for scientific applications

But…

Lazy Propagation Blocks VCPUs



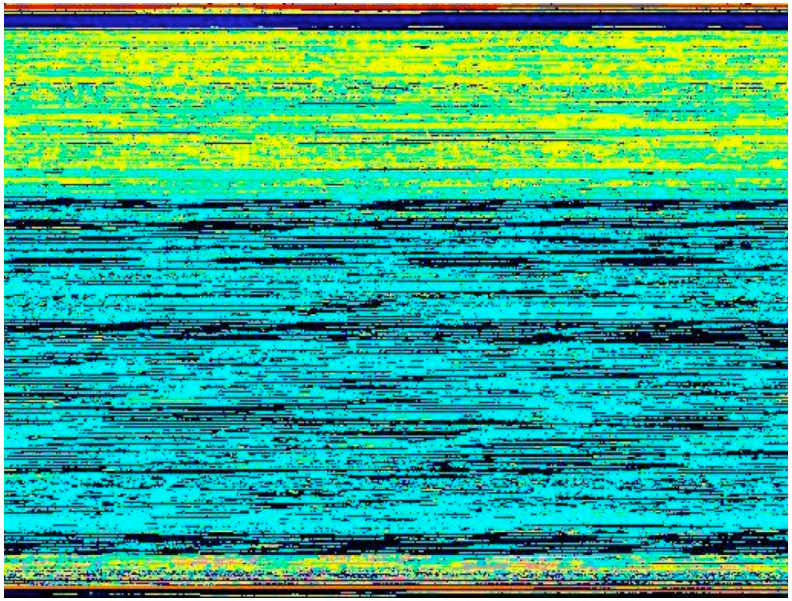
The Propagation Trade-off

- Lazy propagation
  - Fast start, but degraded QoS
- Eager propagation
  - Slow start, but good QoS

- Middle ground (fast start, good QoS)
  - Prefetch needed pages early
  - How do we find related pages?

Key Observation

- Semantically related regions are interwoven
- Blindly fetching by locality gets unneeded pages



- VMM suffers from “semantic gap” problem
- Need to bridge this gap to deliver micro-elasticity

VM State Coloring

- Our approach:
  - Use variety of hints to glean page's probable role
  - "Color" related regions so we can tune their handling
- Page table properties (x86)
  - Gives four basic colors, code vs data, kernel vs user
  - Code vs data by the NX bit
  - User vs kernel by root page table pages
- Peeking inside the pages
  - Traverse the radix tree to color the cache of each file
  - Differentiate unused pages from kernel data

Tuning prefetching by color

- Pre-fetch by locality within same color
- Tune the prefetch window size by region
- Example: Prefetching cached files
  - The clone faults on a page in file system cache
  - If a data file, read ahead 8 pages
  - If executable, use 4 page pivot window

Reducing the footprint

- Allocate on demand
  - Kaleidoscope allocates pages only on use
    - Each wasted prefetch increases footprint
  - Accurate (color-guided) prefetching is key
    - Blind gets every page, so on demand alloc pointless
- Avoid storing duplicate pages
  - No brute force finding de-dup opportunities
  - Data pages unlikely to yield savings

Ensuring correctness

- Cloning vs booting: slight differences
  - Caches are warm, so not reset to a common state
  - Any in-flight requests should die on clone
- Many server applications work as is
  - New IP for clones, old inbound sockets fall away
  - SPECweb, MySQL queries, httperf, etc work fine
- Hooks for intervention (if necessary)
  - Reconfiguration script, SIGCLONE signal

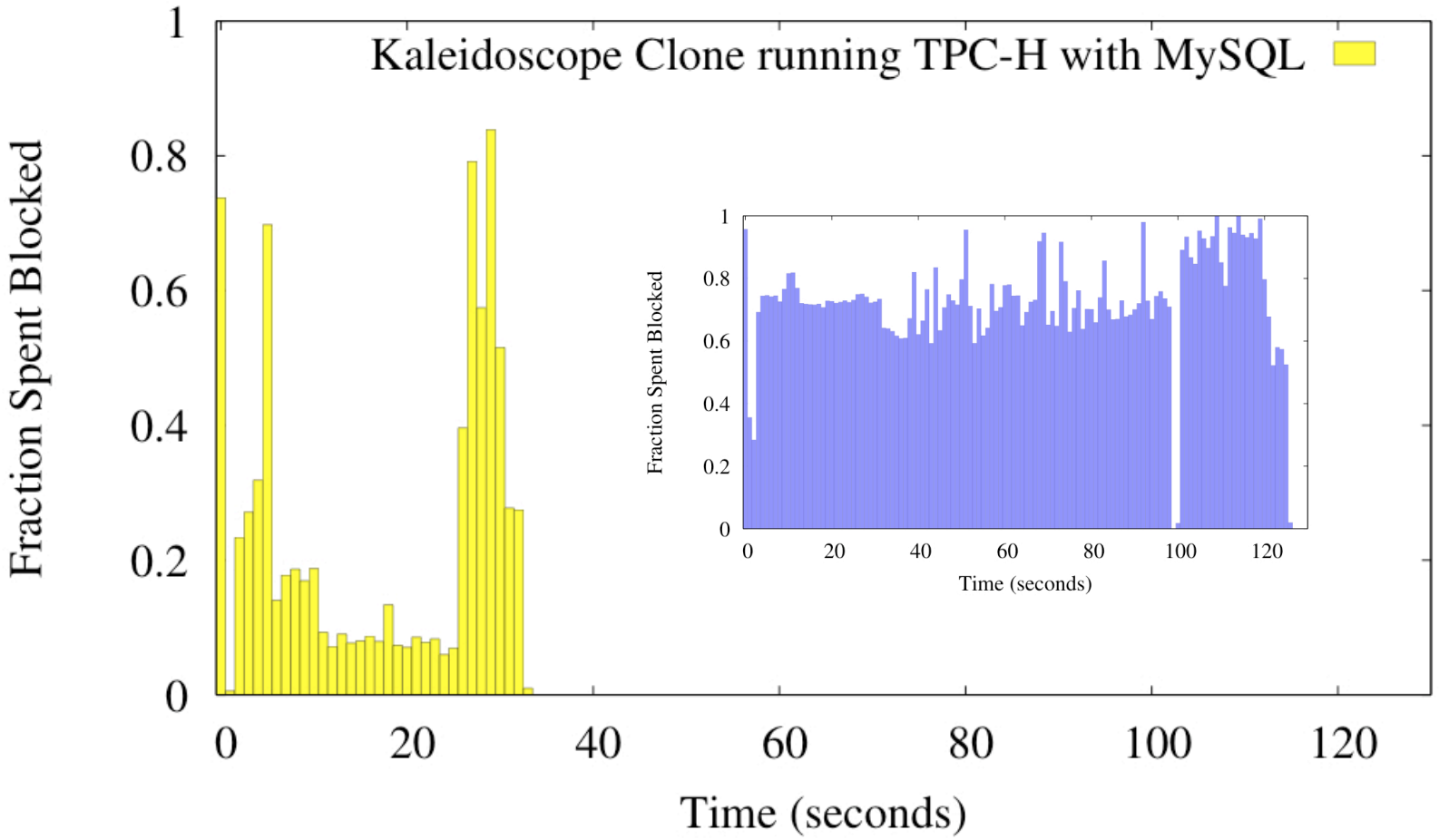
Outline

- Today's Clouds
- Techniques for a Better Cloud
- Evaluating Kaleidoscope
- Conclusions

Experiments

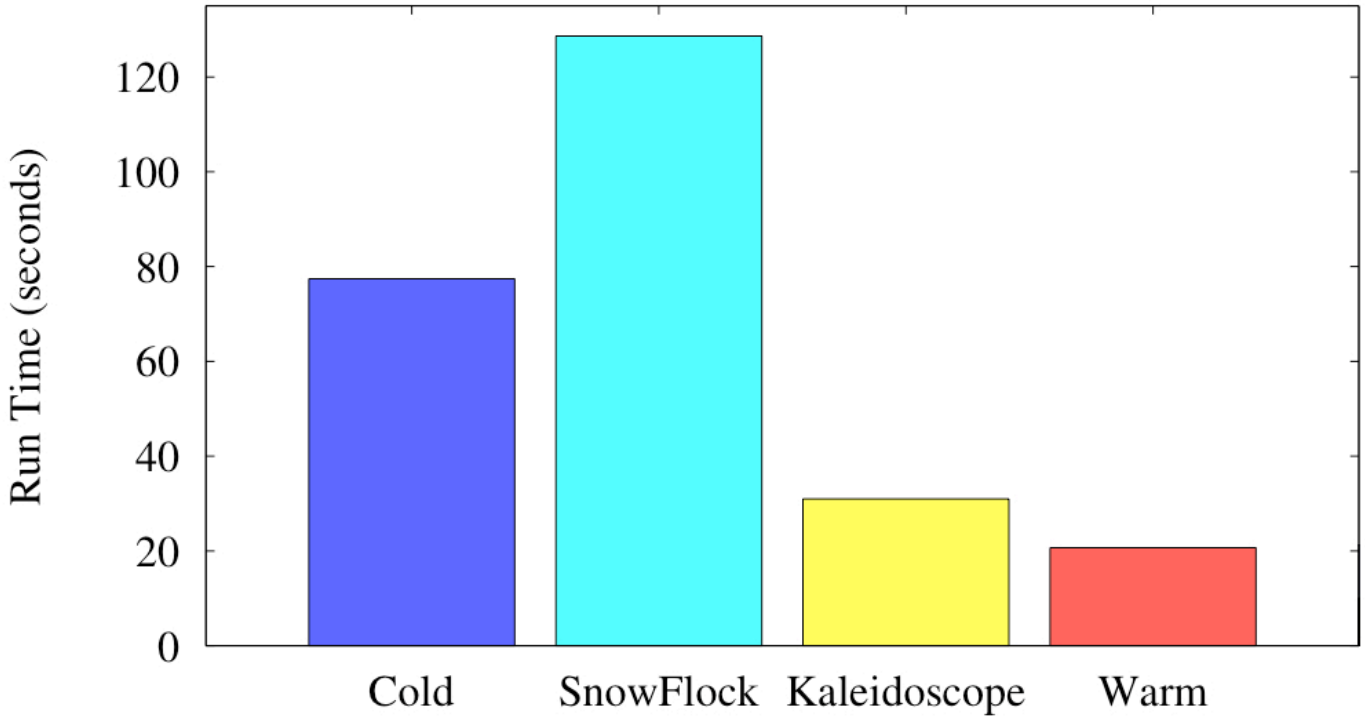
- Measured new worker behavior
  - Time to “warm up”
  - Performance during warm up
- Web Server (Apache in 768MiB VM)
  - Httperf, static and dynamic pages
  - SPECweb 2005 (banking, support and ecommerce)
- OLAP (MySQL in 2 GiB VM)
  - TPC-H read-only database queries

VCPU Blocking Revisited

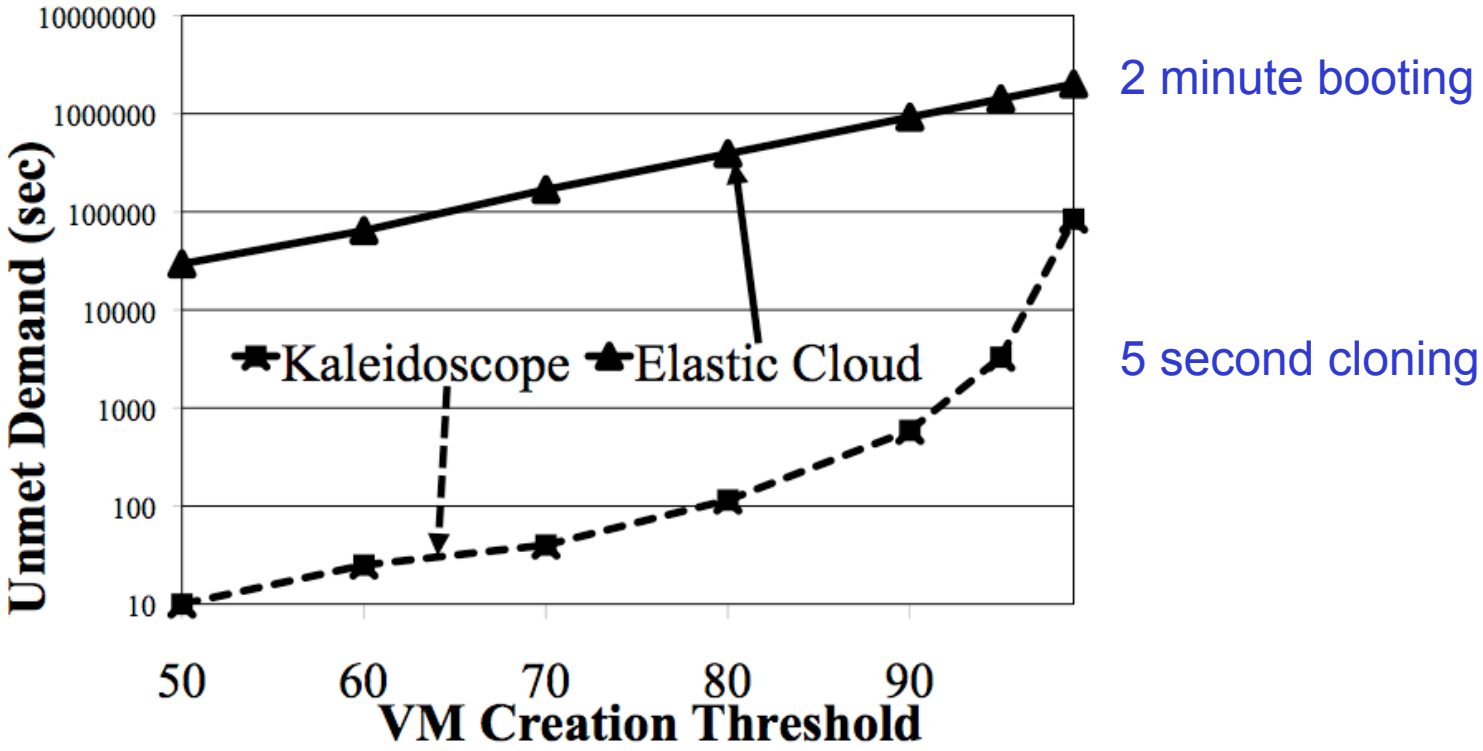


Kaleidoscope's OLAP Performance

- Vs Cold, 2.5x faster during (shorter) warmup
- Vs Warm, 67% throughput during 30 s warmup

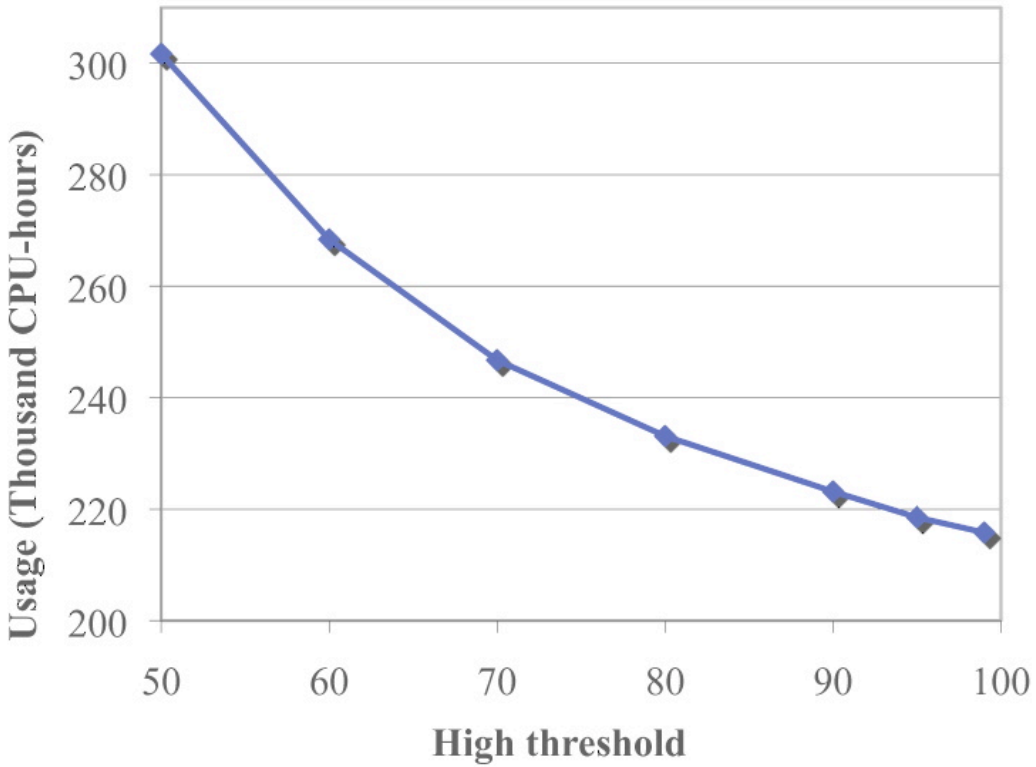


Impact of boot time on QoS



- Slow instantiation: need more slack capacity
- QoS drives creation threshold, which drives savings

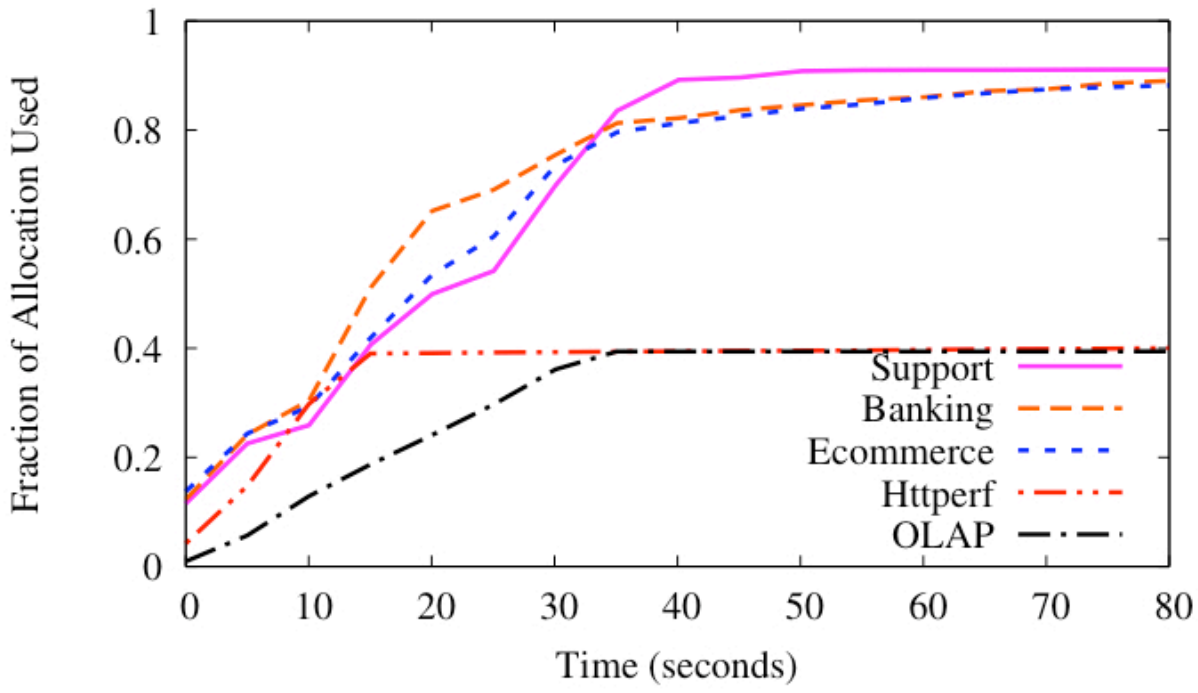
Reduced Server Usage



- Kaleidoscope delivers more with less
  - 98% fewer overloads, 26% less usage

Kaleidoscope's Fractional Workers

- VM's grow only as necessary
- Efficient de-duplication (no brute force)
  - Targets likeliest regions: 10% effort for 53% results



Smaller Datacenters

- Fast instantiation, less slack
  - So fewer, busier VMs
- Short-lived, smaller workers
- Savings depends on QoS
  - 98% fewer overloads, 50% smaller datacenter

Conclusions

- VM state coloring
  - Inexpensively bridges semantic gap
  - Enables effective state prefetching
  - Efficiently identifies de-duplication candidates
- Micro-elastic cloud (vs today's commercial clouds)
  - Faster instantiation, so less slack capacity needed
  - Pre-warmed workers perform better, sooner
  - Many small, transient workers reduce infrastructure needs

Questions?