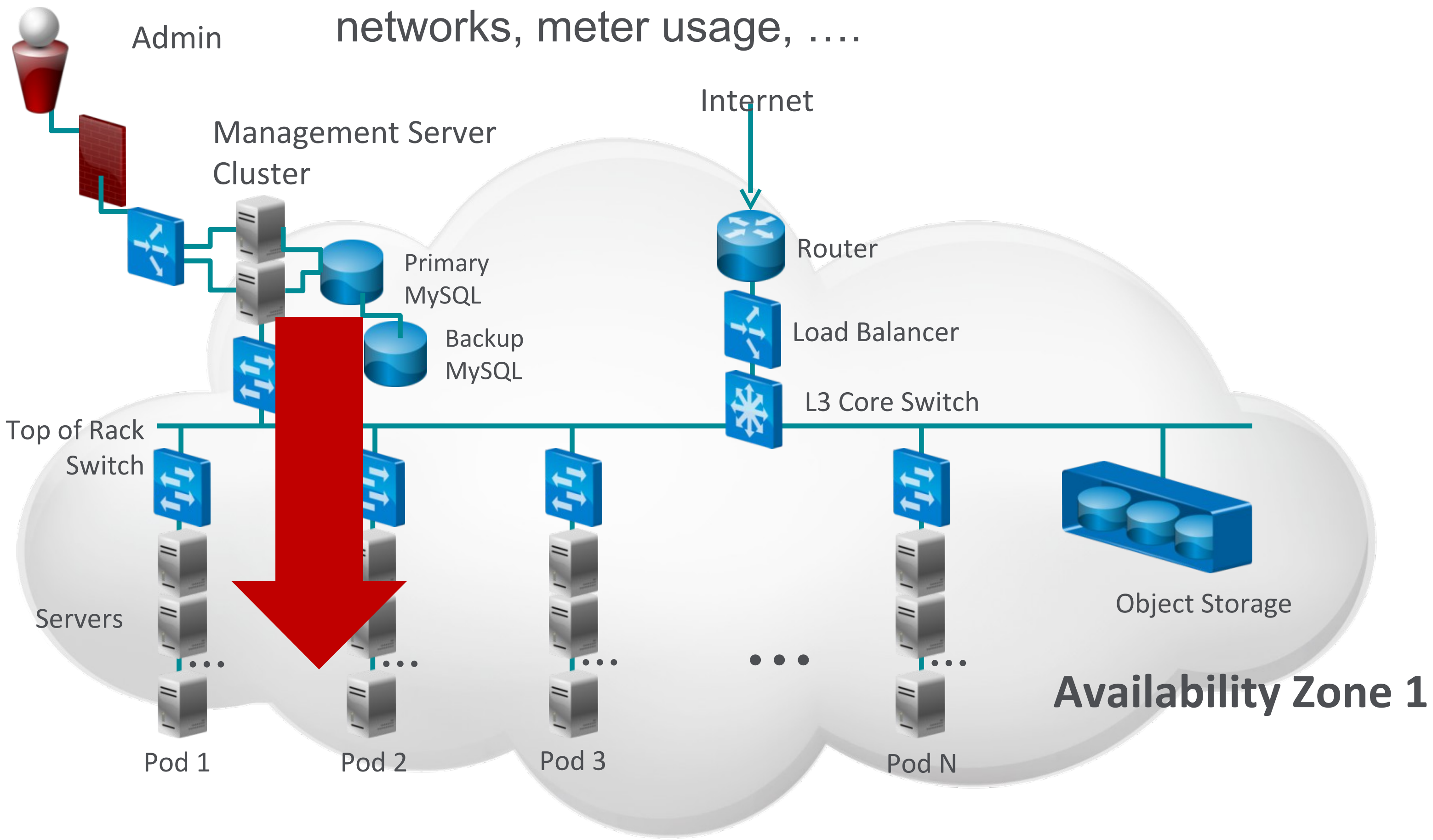
- CloudStack Scalability
- *Testing, Development, Results, and Futures*

- Anthony Xu
- Apache CloudStack contributor

Apache CloudStack: a project in incubation

- Secure, multi-tenant cloud orchestration platform
  - Turnkey platform for delivering IaaS clouds
  - Hypervisor agnostic
  - Highly scalable, secure and open
  - Complete Self-service portal
  - Open source, open standards
  - Deploys on premise

Manage hosts, create VMs, virtual disks, virtual networks, meter usage, ….

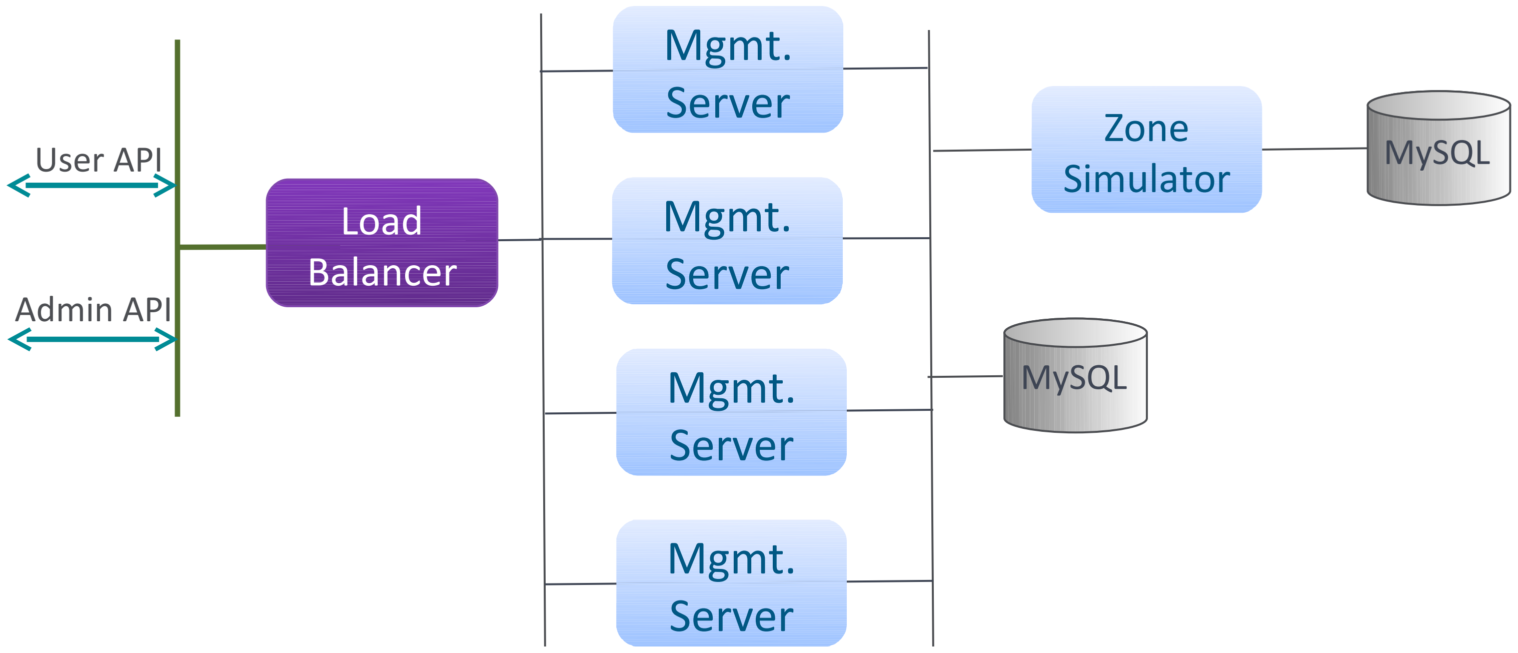


Thinking about cloud orchestration at scale

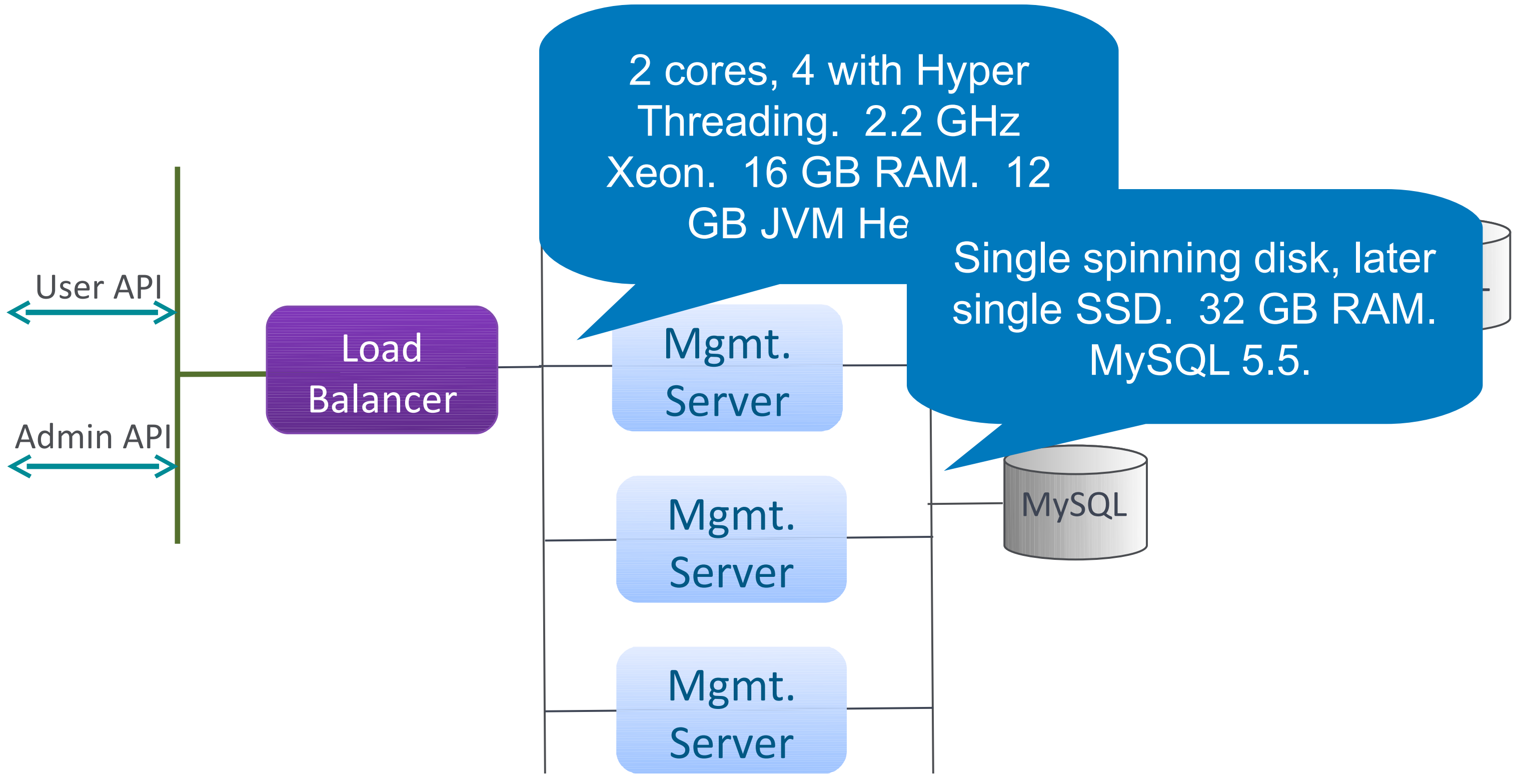
- Host management
- Capacity management
- What host to use to deploy a new VM
- Failure handling
- Security group propagation
- Set a goal

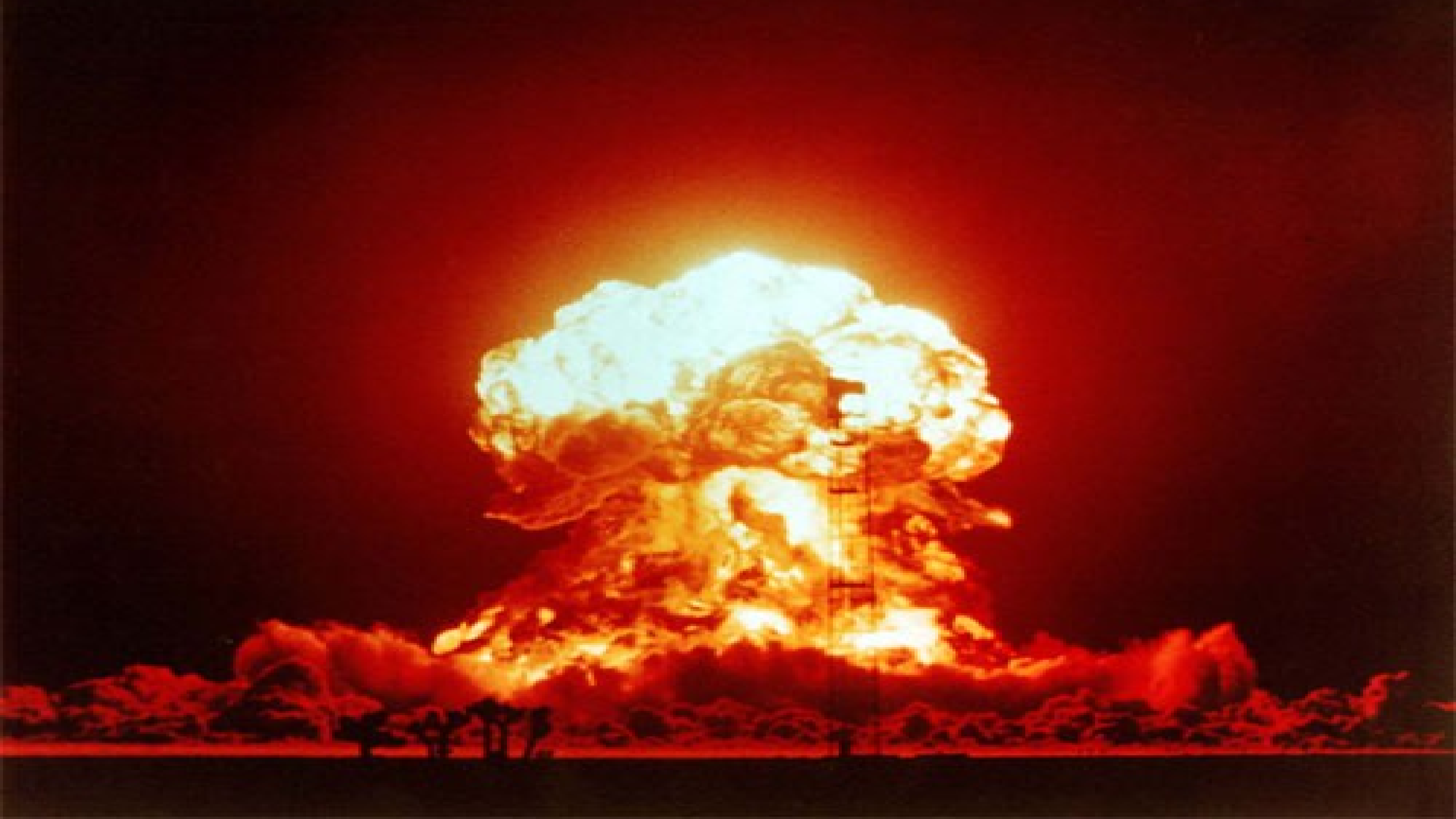
We can’t afford this as our QA lab

Simulator enables scale testing



Environment


User API
Admin API
Load Balancer
Mgmt. Server
Mgmt. Server
Mgmt. Server
MySQL
2 cores, 4 with Hyper Threading. 2.2 GHz Xeon. 16 GB RAM. 12 GB JVM He
Single spinning disk, later single SSD. 32 GB RAM. MySQL 5.5.



Allocator performance is awful with 1000 hosts

- Two minutes to decide which host to use for a new VM!
  - Computing capacity for every pod repeatedly
- Fixed that, but still 12 seconds to decide
- Use host tags, down to 2 seconds
- Major changes required to improve further
  - In 2.2.0, store capacity info in DB, skip pod altogether
- Harness the power of SQL select and all is well

Polling doesn’t scale

TRUE? FALSE?

Sometimes, it is good enough

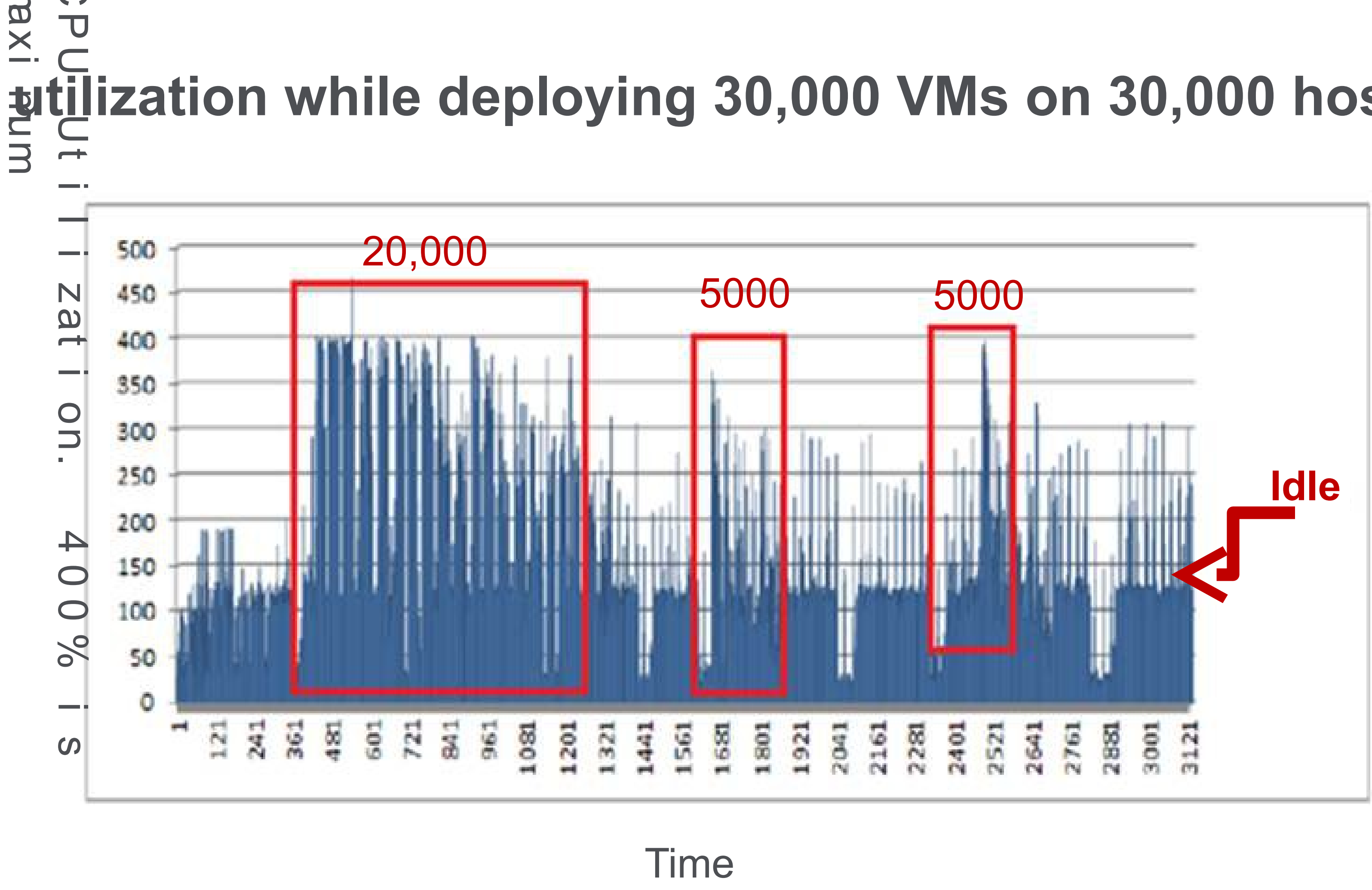
Host management

- Check host state via TCP connection
- Check every minute
    - 30,000 checks per minute, 500 per second
    - But they take 10 seconds, so 5000 in parallel
    - Not using async I/O so 5000 threads required…
    - Single JVM can support 5000+ threads so this is concerning but may not be the limiting factor

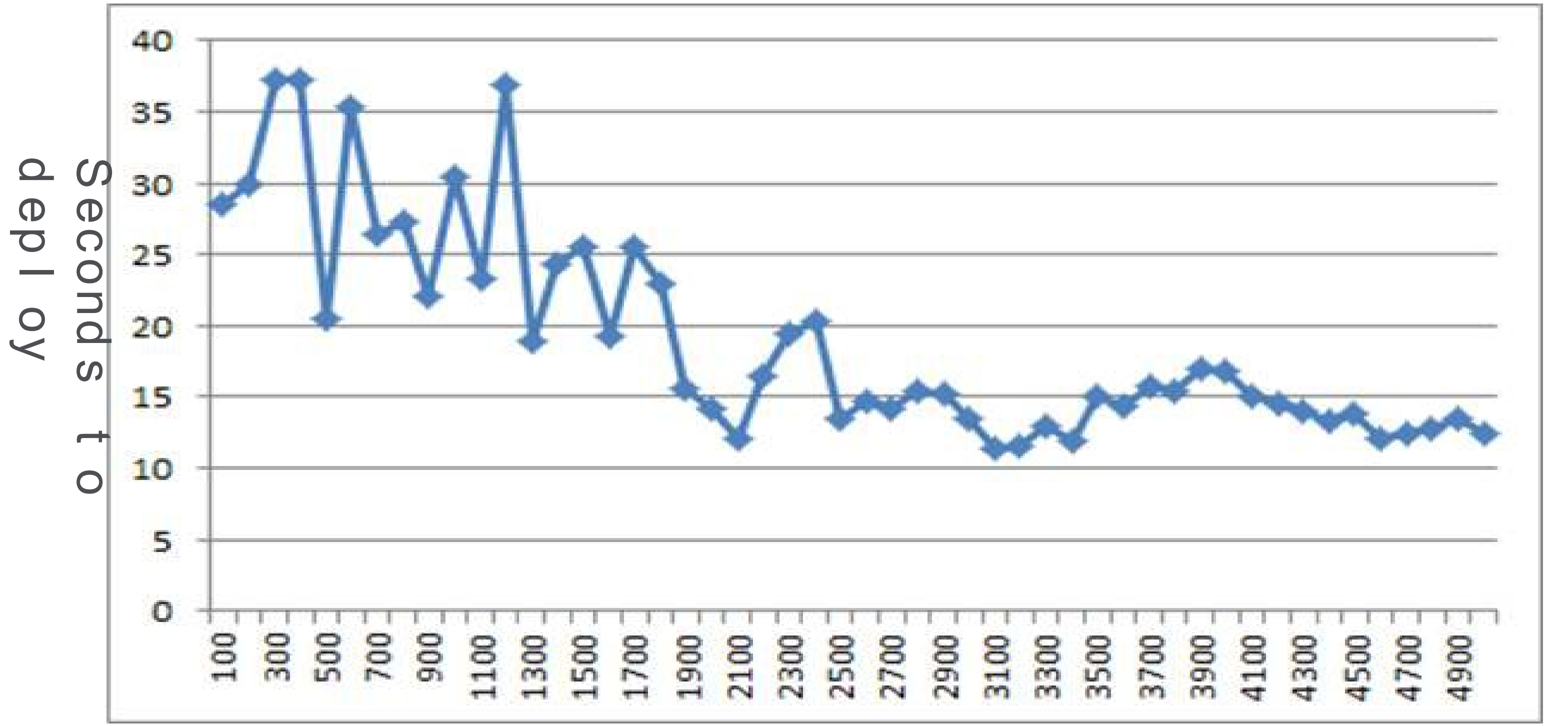
Host management

- What is the maximum feasible JVM heap size?
    - Some people use heaps with hundreds of GB
    - Commercial tools can help, but cost
    - We decided to stay below 20 GB (GC concerns)
- How much CPU is required for background processing?

CPU utilization while deploying 30,000 VMs on 30,000 hosts



Deploy time from 25,000 to 30,000 VMs



VM number: 25,000 plus X

Problem: agent load balancing

- Management servers start/stop/fail/crash
- How do newly started Management Servers get agents / work?
- When a Management Server exits, how do others pick up its load?
- When new hosts are added how is the load distributed?

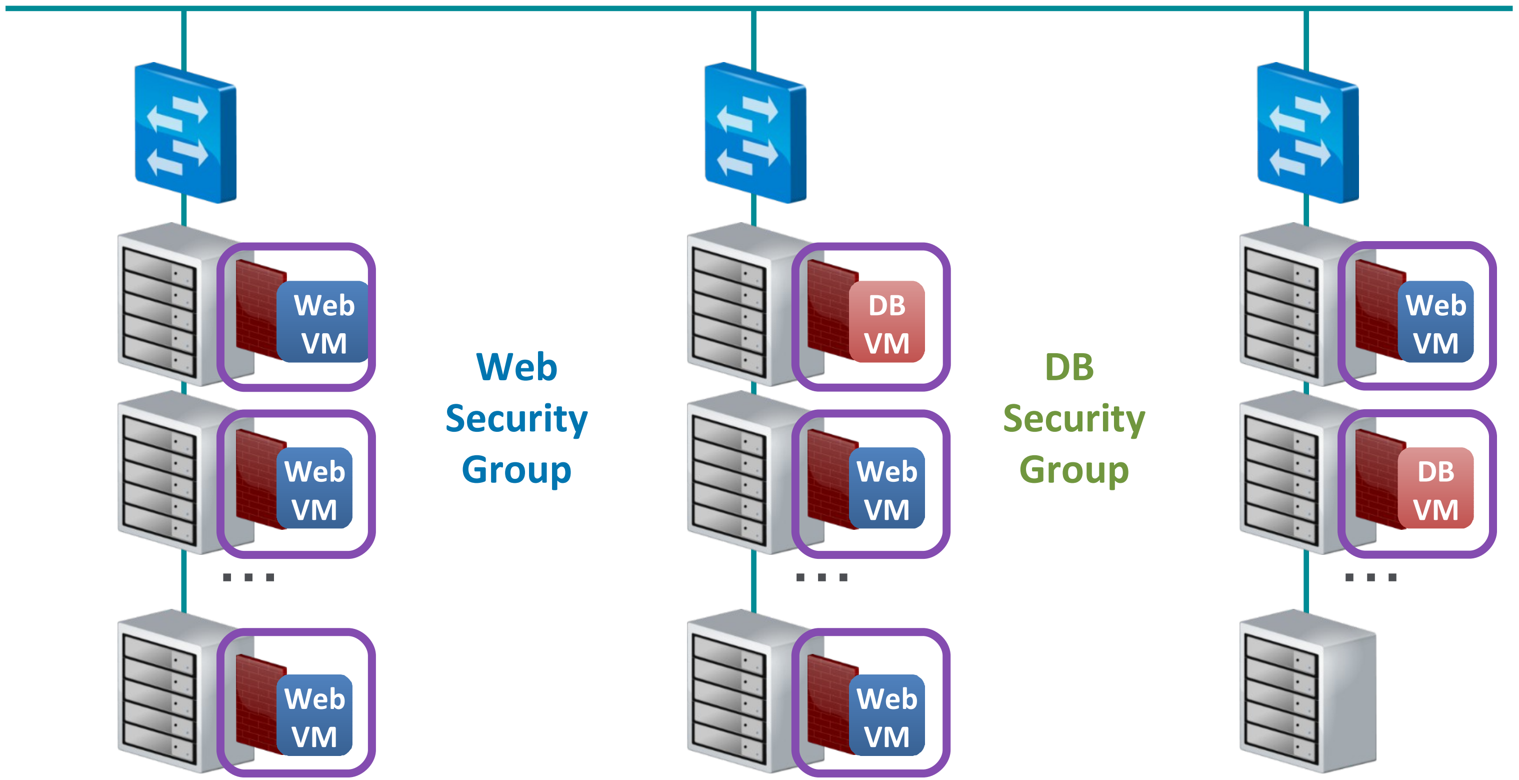
Common use case timings at scale

- 30,000 hosts and 4 Management Servers
- 4 Management Servers running, 1 fails: 10 minutes to redistribute 7500 agents
- 3 Management Servers running, add a fourth: 40 minutes to redistribute load evenly

IMPORTANT

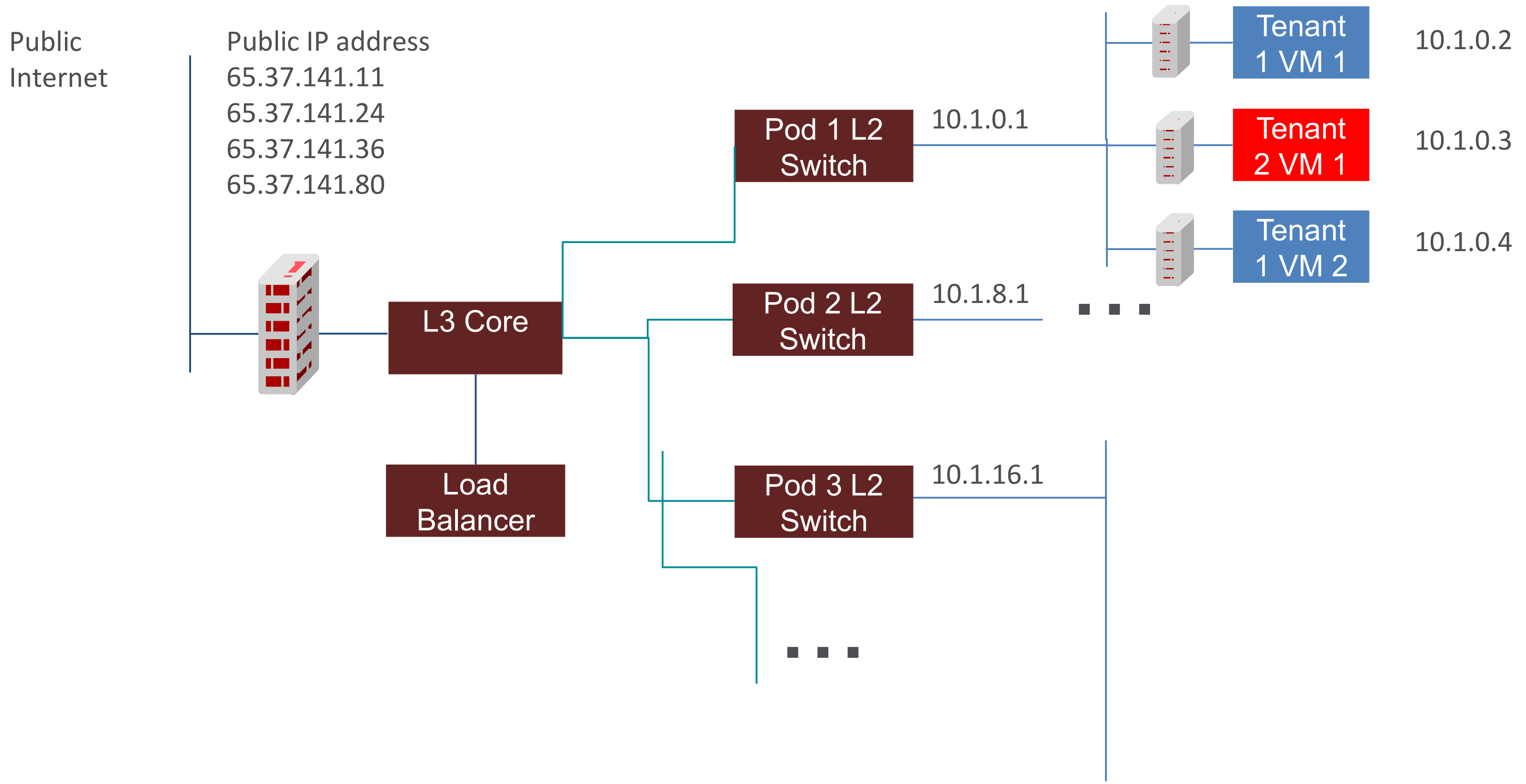
- 0 Management Servers running, start all 4 simultaneously: 16 minutes to connect to all 30,000 hosts

Understanding security groups

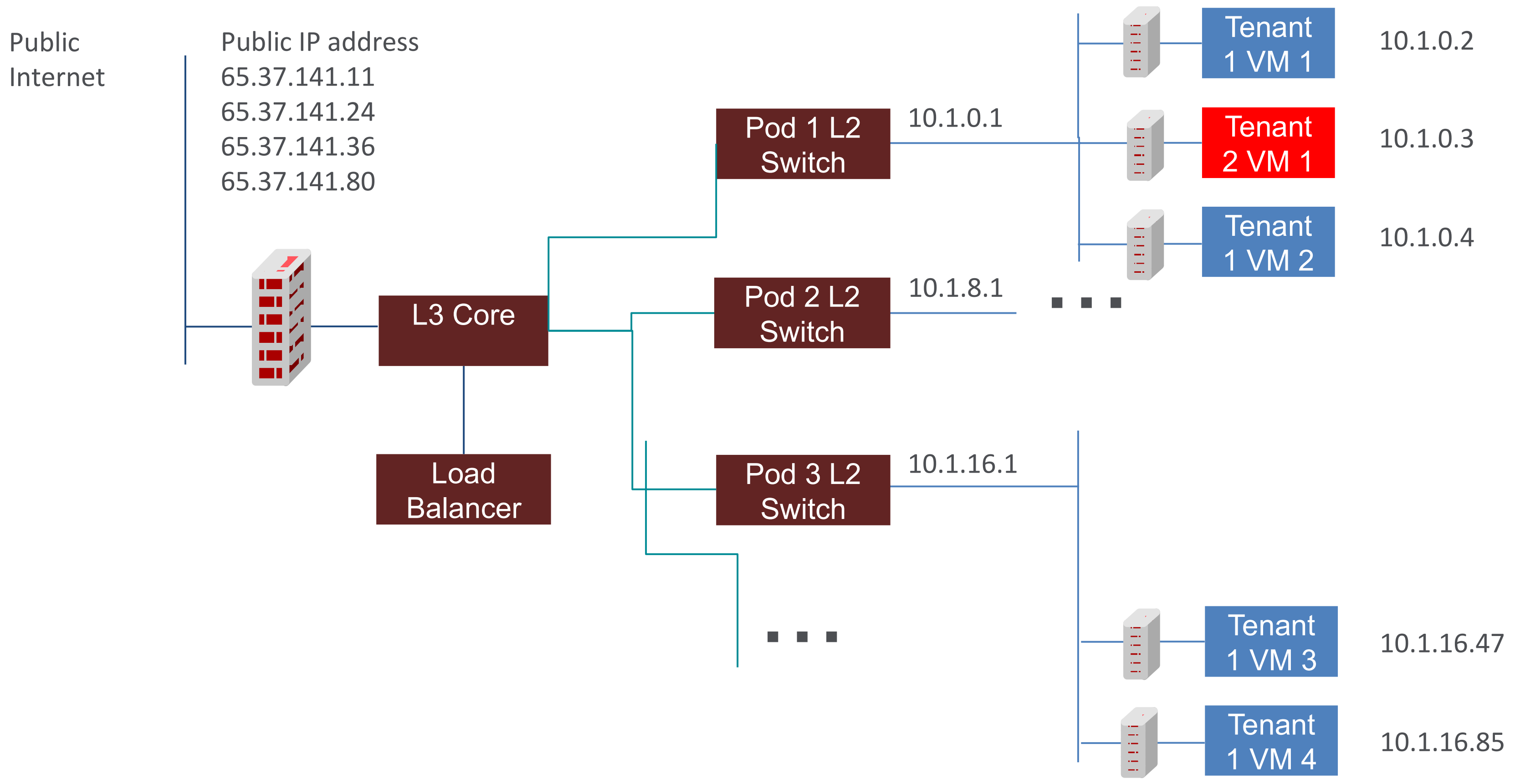


Ingress Rule: Allow VMs in Web Security Group access to VMs in DB Security Group on Port 3306

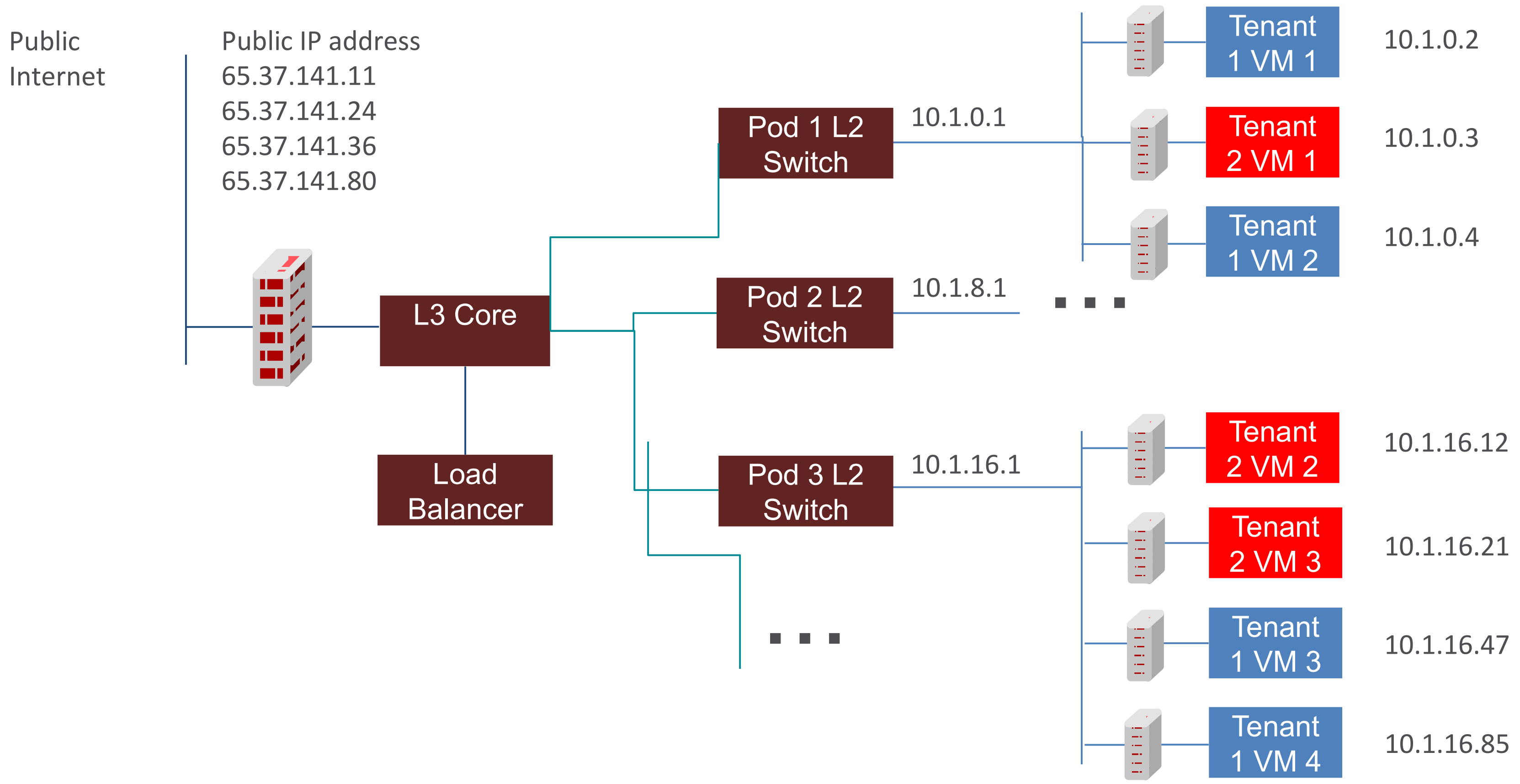
L3 isolation with distributed firewalls



L3 isolation with distributed firewalls

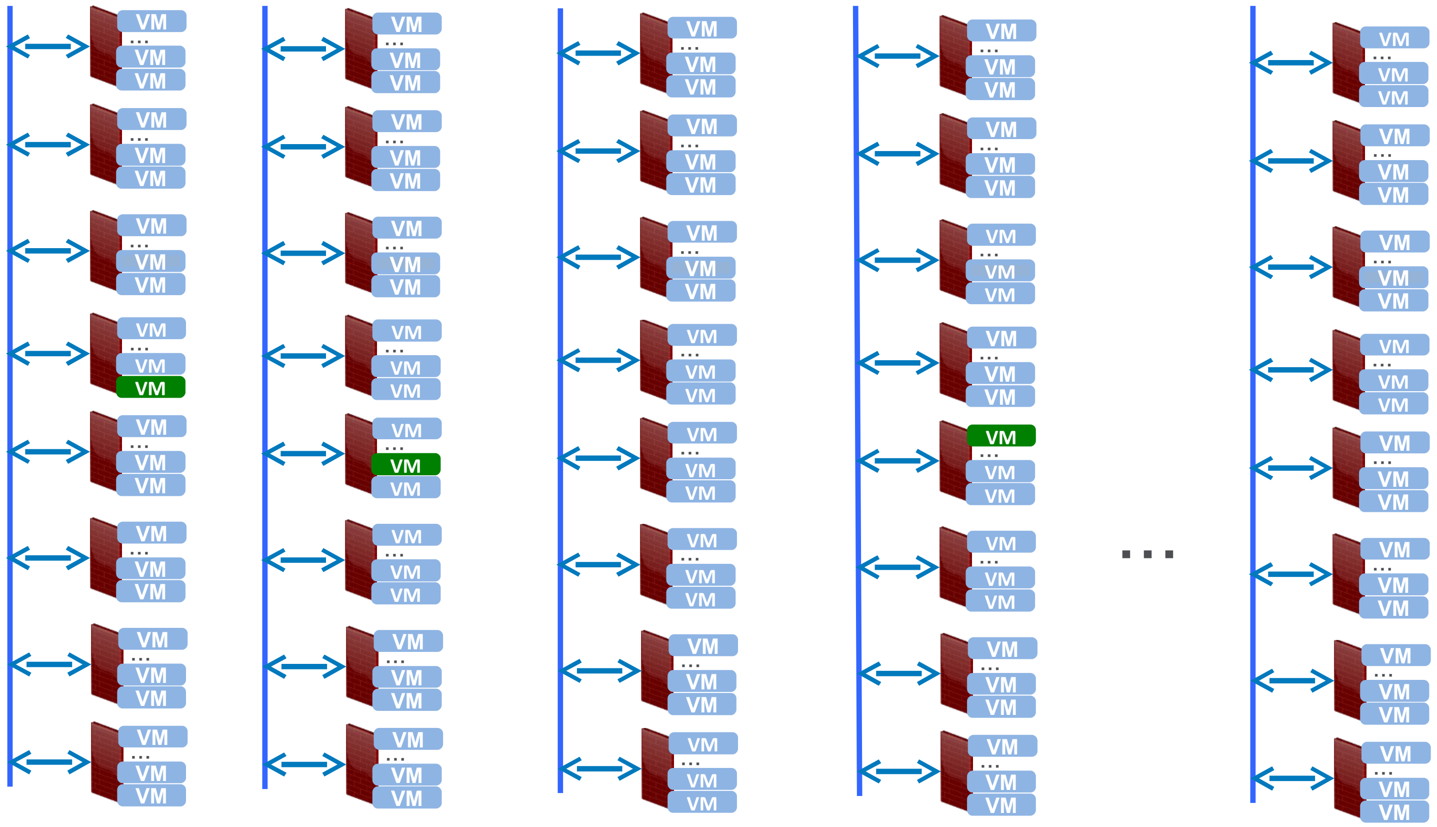


L3 isolation with distributed firewalls



1 Firewall per Virtual Machine

One million firewalls?



Orchestrating hundreds of thousands of firewalls

Well-known software scaling techniques

- Message queues
- Consistency tradeoffs
- Idempotent configuration & retries

CloudStack uses

- Special purpose queues
- Optimized for large security groups
- Eventual consistency for rule updates

Problem: firewall rules explosion in dom0

Allow Security Group {Web} on TCP port 3060

```
-A FORWARD -m tcp –p tcp –dport 3060 –src 10.1.16.31 – j ACCEPT
-A FORWARD -m tcp –p tcp –dport 3060 –src 10.1.45.112 – j ACCEPT
-A FORWARD -m tcp –p tcp –dport 3060 –src 10.1.189.5 – j ACCEPT
                        ...
-A FORWARD -m tcp –p tcp –dport 3060 –src 10.21.9.77 – j ACCEPT
```

Performance suffers for large security groups

Problem: firewall rules explosion in dom0

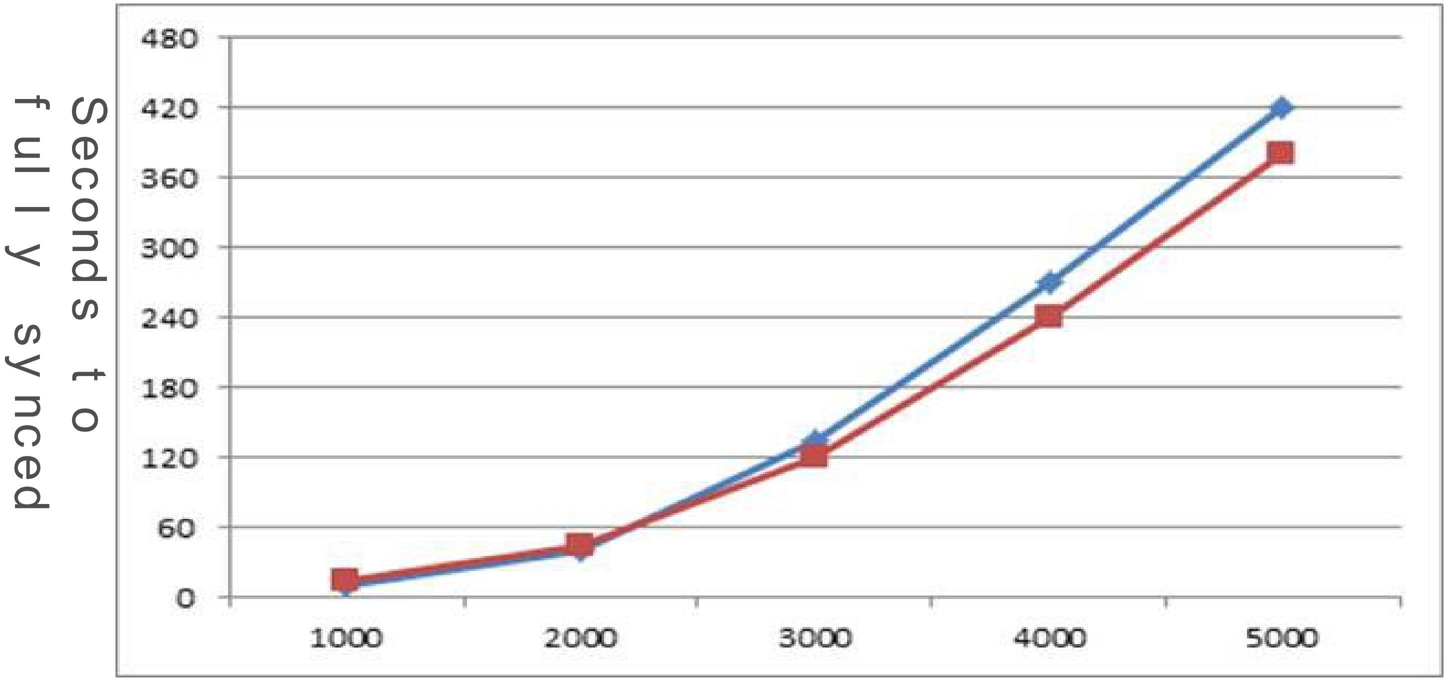
Fix with ipsets:

```
ipset –N web_sg  iptreemap
ipset –A web_sg 10.1.16.31
ipset –A web_sg 10.1.16.112
ipset –A web_sg 10.1.189.5

ipset –A web_sg 10.21.9.77
            …
-A FORWARD –p tcp –m tcp –dport 3060 –m  set –match-set web_sg src  -j ACCEPT
```

See also http://daemonkeeper.net/781/mass-blocking-ip-addresses-with-ipset/

Security group propagation time

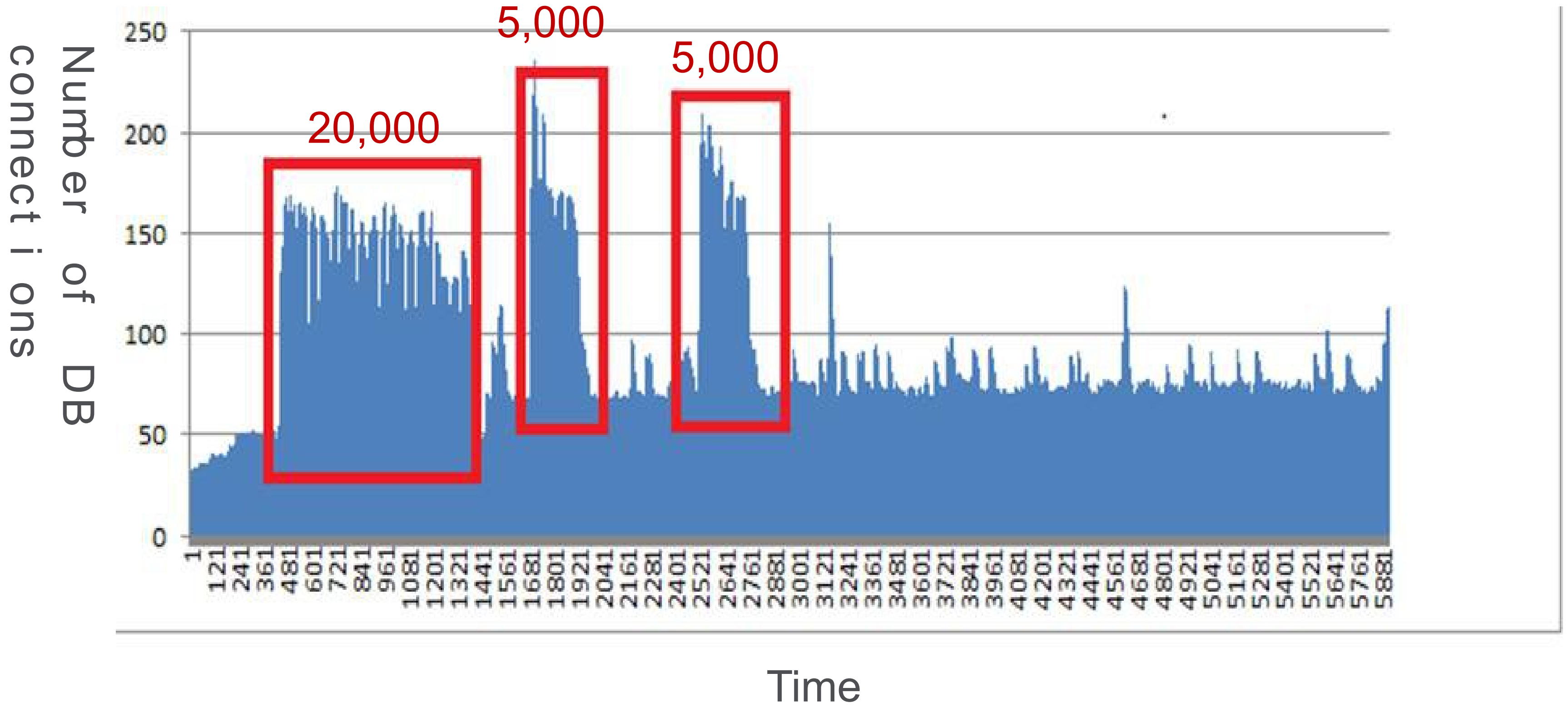


Number of VMs in security group

Problem: database connection management

- Scale testing resulted in several “too many open connections” errors from MySQL
- Common problem: holding open connections while doing long-running operations
- Took some code clean up and refactoring
- No longer an issue
    - 10,000 connections are OK
    - CloudStack is far below that

DB connections per MS while deploying 30,000 VMs



Other considerations (beyond control plane)

- Network design and devices
- Object store scalability
- Per-host and cluster scalability
- Storage
- Understand your workload

Future work

- Improve simulator accuracy
- Publish results of advanced network (VLAN) testing
- Verify assumption of VM density not impacting scale

More information and joining the project

Project web site:
http://incubator.apache.org/projects/cloudstack.html

Mailing lists:

cloudstack-dev-subscribe@incubator.apache.org

cloudstack-users-subscribe@incubator.apache.org

Scalability study:

[illegible]

- Q&A