Twitter's Production Scale

Mesos and Aurora Operations

Joe Smith
Tech Lead, Aurora/Mesos SRE
Gameplan
— **Background**
  — Using a central system to manage containers

— **Annoyances**
  — For Users and Operators
  — Running 10s of thousands of hosts

— **Outages**
  — Worst Case Scenarios
  — What to do when they happen
Tales from the Front Lines

— 10s of machines to 10s of Thousands
— Migrated hundreds of services from bare-metal
— Build and deploy pipeline is on YouTube
"Team, please adopt containers immediately."
"why?"
"Because they are hot"
"But, why are they hot"
"Because they are hot, damn it"

1 https://twitter.com/cloud_opinion/status/623568543771045888
Why to Use Containers

— Ease of managing a service
— Abstraction from base infrastructure
— One team to manage low-level provisioning, maintenance, and repair
Ease of Managing a Service

— Mesos is an excellent resource manager
— Aurora is an excellent Service Scheduler
  — Provides a state machine for service lifecycle
— Users build services on Opinionated Infrastructure
— SREs can focus on their service's reliability, not kernel upgrades
Abstraction from Base Infrastructure

— Abstract away user-space from operator-space
— Decouple Operating System upgrades from Application upgrades
— Different JVMs, different deploy cadence
  — Prevent your infrastructure teams from becoming TPMs
One Team to Provision, Maintain, and Repair

— Not every team needs to write deploy scripts
— No longer wait for hardware to show up
— Simple and standard checks for hardware problems
Annoyances
For Users

— Service Configuration
— Lots of knobs to tweak
— Aurora Client v1 to v2 migration
— SSL Certificates in Python
— Keep client's certs up to date
— Resource Isolation
— Clarifying throttling, OOM
For Operators

— Building Python eggs with native dependencies
— Deploying Mesos and Aurora
  — Aurora provides excellent deploy automation
— How does Aurora get deployed?
— AURORA-1075: An instance on each host

© Twitter, Inc 2015
Puppet

— Obviously, it scales
— Mutable Infrastructure
— Does not have ordering guarantees
  — Difficult to coordinate reboot-required deploys
Something is missing
Analogy - The Network

— The Network (switches, routers, hosts)
  — Well-supported
  — Mission Critical - everyone relies on it
— IPMI Network
  — out-of-band
  — Select few users
— When "The Network" is down, better hope it works
Analogy - The Network

— Aurora/Mesos are "The Network"
— What is our "IPMI" or "backdoor" network?
The Future

— Mesos SRE is building out a system using Ansible + ZooKeeper for coordination
— Remove as much *mutability* as possible
  — Still grant break-the-glass operability
— Filesystem Isolation empowers this
— Will test feasibility and, if successful, will Open Source
Outages
EASY DEPLOY THIS WEEK

JUST A SMALL SINGLE-PATCH DIFF
Third Worst Case Scenario
Rack- or host-level outage

— For the most part, a non-issue
— If it hits your pager, you're using dedicated hosts or perhaps..
Cluster-wide config changes

— Pushed out slave configuration all out once
  — (Always slow-roll out changes)
— Slaves restarted to pick up changes, didn't come back, and were marked LOST
— Limit the slave removal rate
— Stop the masters
— Hit the BigRedButton™
Second Worst Case Scenario
— Aurora Schedulers
— Mesos Masters
— ZooKeeper Ensemble
Problems

— No deploys
— No tasks get rescheduled
— No cron jobs fire
— No task reconciliation
How can this happen?
Aurora

— Accidental deploy where no known scheduler could read the log
  — Again, invest in improved build/deploy pipeline
— Took much care changing quorum size
— Timeouts when writing to the Replicated Log
  — I/O Contention (log rotate)
Mesos

— Writes timing out to the replicated log
— ZooKeeper is a big one here
ZooKeeper

— Tune your ZK client settings correctly!
  — Set appropriate session timeout
— Never co-locate Service Discovery, Leader Election, other use case ensembles
— Emphasizes the importance of isolation for shared services
  — Good fences make good neighbors
IF YOU'RE AURORA ONCALL
WHEN ZK ONCALL GETS PAGED
YOU'RE GONNA HAVE A BAD TIME
Swapping ZooKeeper Ensembles - Aurora

— Well planned
— Thousands of lines of Python
— 95%+ Unit Test Coverage
— Hours of test cluster integration testing
Can you wing it?
Swapping ZooKeeper Ensembles -Mesos

— Ensemble for leader election was hammered by misbehaving clients
— Shut-down the masters
— Live-pivoted via a puppet change to the slaves
— Brought them back up
First Worst Case Scenario
Reschedule all the things
(only once)
Reschedule all the things

— Master was paused (SysRq-T) for 17 minutes
— Mesos sent simultaneous "all slaves are LOST" message
— Aurora ~immediately marked all tasks LOST
— Aurora began rescheduling
— DDOS'd by status updates
— GC Executor launching slowed us down
Helpfully Unhelpful

— The GC Executor did its job, and reconciled the difference
— All tasks were killed
Pulling out of the nosedive

— Increased the `task_timeout`
— Lowered Aurora's scheduling rate
— Decreased interval of launching GC Executors
  — Now using Mesos' Task Reconciliation
— Could have slowly added slaves back as well
Improved

— Aurora's scheduling rate is *dramatically* improved
— More safeguards in both layers
  — Rate-limiting slave removal
  — Scheduler driver validates messages from elected master
— Task Reconciliation (no more out-of-band GC)
The

Worst Case Scenario
(Hasn't Happened Yet)
Lose the entire cluster

— Scheduler starts up *empty*
— Restore from Backup
  — If not.. users *must* resubmit all jobs
Prioritizing Reliability

— The Aurora and Mesos communities are *highly* receptive and supportive of the Operations Perspective

— These services are built for large, critical *production* infrastructure

— Entire enterprises rest on their ability to keep services up and running
Contact

— #aurora on freenode
— #mesos on freenode
— @Yasumoto
— @TwitterSRE