Keeping your Cloud Footprint in Check

Coburn Watson
Let us know what you think

Click ‘engage’ to rate session.
Rate 12 sessions to get the supercool GOTO reward

Join the conversation #gotoldn
@coburnw

- Cloud Performance and Reliability @ Netflix
  - Reduce TTD and TTR
  - Build innovative performance analysis tooling
  - Optimize usage of AWS Cloud
  - Steer global user traffic and support failover
  - Inject Chaos into production environment
  - Drive operational best practice adoption
NETFLIX

- 67M+ Subscribers
- > 50 countries
- > 3 billion hours of video streamed monthly
- Huge cloud footprint
- Homegrown CDN
- Strong Originals slate

Join the conversation #gotoldn
• Strong focus on open source efforts
• [https://netflix.github.io/](https://netflix.github.io/)
Our Priorities
Innova<on
Reliability
Efficiency
Innovation
Cost of Innovation and Reliability
Maximize Innovation

- Capacity On-Demand
- Commit-to-Cloud in minutes
- Single Production Account (~ 350 µservices)
- Burst into on-demand, cover with reservation purchases
Cost of Reliability

- Red-Black push model
- Over-provision for redundancy in AWS Region
- Global redundancy through failover
- Purchase “Heavy” AWS EC2 reservations to secure capacity
Efficiency
Efficiency Goals

• Have them and track them!
Monitoring Costs

- ICE: Open Source AWS Cost Monitoring Utility
- Internal Cost Reporting pushed to first-level managers
Maximize Sharing

- Single Production Account
- Fewer/Larger Pools
- Maximize Shared Capacity

> 75% in only 8 EC2 Instance Types
Encourage Borrowing

- All accounts are linked at a billing level
- Large troughs of unused capacity exist (Autoscaling)
- Interruptible workloads for internal “Spot”
Optimization

• Direct Consultation for “Big Fish”
• Tooling for Everyone

1. Develop
2. Deploy
   • Scale
3. Optimize (if needed)

1. Develop
2. Canary
3. Optimize (if needed)
   • Deploy

New Services or Features

Ongoing Service Development

Join the conversation #gotoldn
Improving Stack Observability

• Too big for commercial tools
• Patch key middleware where necessary

Transaction Tracing with Resource Demand
Mixed-Mode JVM CPU Flame Graph
Monitor Capacity Shortfalls

- Constrain On-Demand charges
- Identify/alert on significant capacity provisioning events
Data Points

• Internal Borrowing
  • Encoding consumed 135k cross-account EC2 Instance hours June 2015 (> ~ $200k/monthly savings)

• Data Platform (Hadoop, etc.) saves > $1MM/year
Summary

• Target your Innovation:Efficiency ratio

• Push cost context to the team level

• Embrace the elasticity of the Cloud
Please remember to rate this session. Thank you!
Thanks !