Mobile + HA + Cloud

Eugene Ciurana

pr3d4t0r - irc.freenode.net
##java, ##security, #awk, #python, #bitcoin
irc.oftc.net: #tor, #tor-dev, #tails

qcon2014@cime.net

About Eugene...

- 15+ years building HA, mission-critical systems
- State-of-the-art engineering for some of the biggest and brightest worldwide
- Open source evangelist and author
- Not a web guy...
- Adviser to several VC funds in the US, Asia, and Europe
- Now providing business and technology development advise to mobile and enhanced reality companies worldwide

Summly

cime

Very Important

Please Ask Questions!

(don't be shy...)

Mobile HA and Cloud

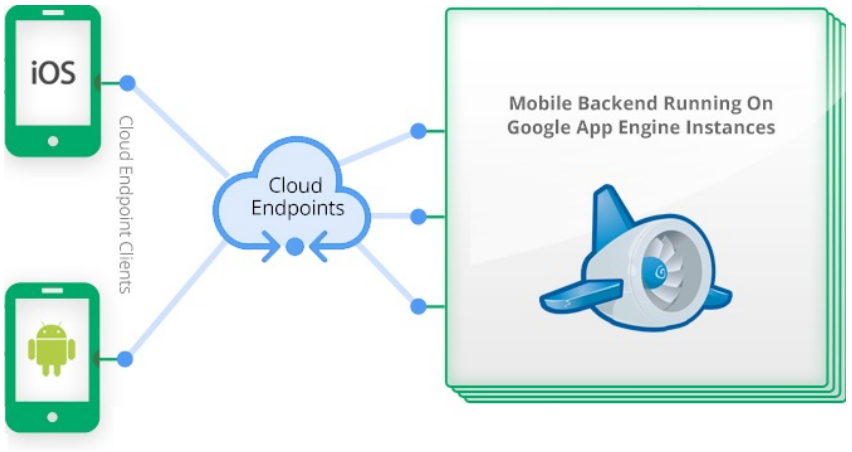
- Bootstrapping a mobile startup almost always includes a cloud component
- Cloud services and servers (Saas and PaaS)
- Main reason? Battery life!
  - Processing and net I/O == battery drain

Mobile HA and Cloud

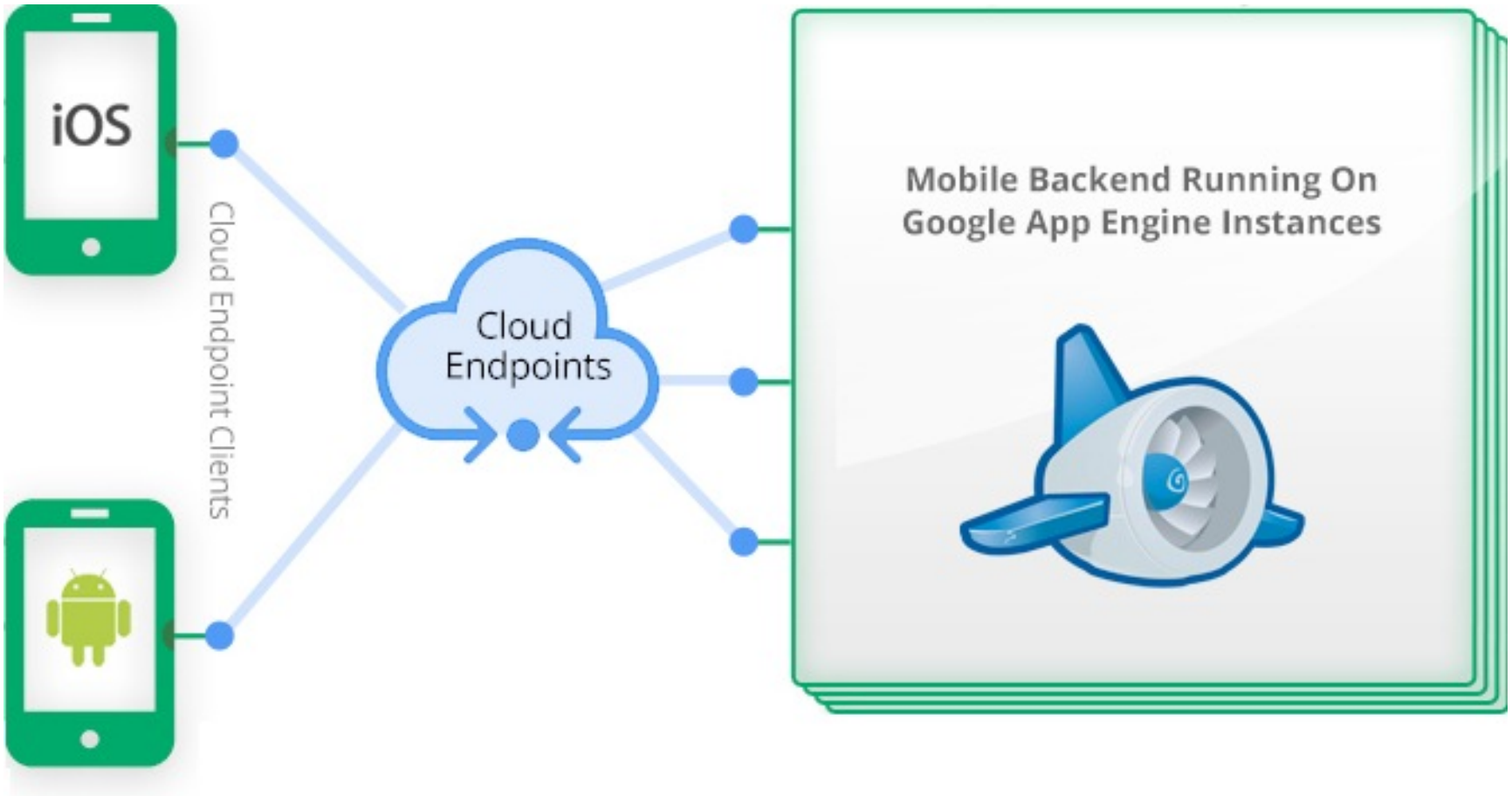
- Cloud services are **Always On**
  - Until they aren't
- App should always appear to be live for the user
- All services must be self-healing

Mobile HA and Cloud

- HA in mobile != HA for desk or web apps
- Assume the device is a cache
- The service provider is The Law



Mobile HA and Cloud



Which Cloud Provider?

- SaaS - interfacing with ready-made services; Salesforce.com
- PaaS - Google App Engine, CloudHub
- IaaS - Amazon Web Services, MS Azure

Which Cloud Provider?

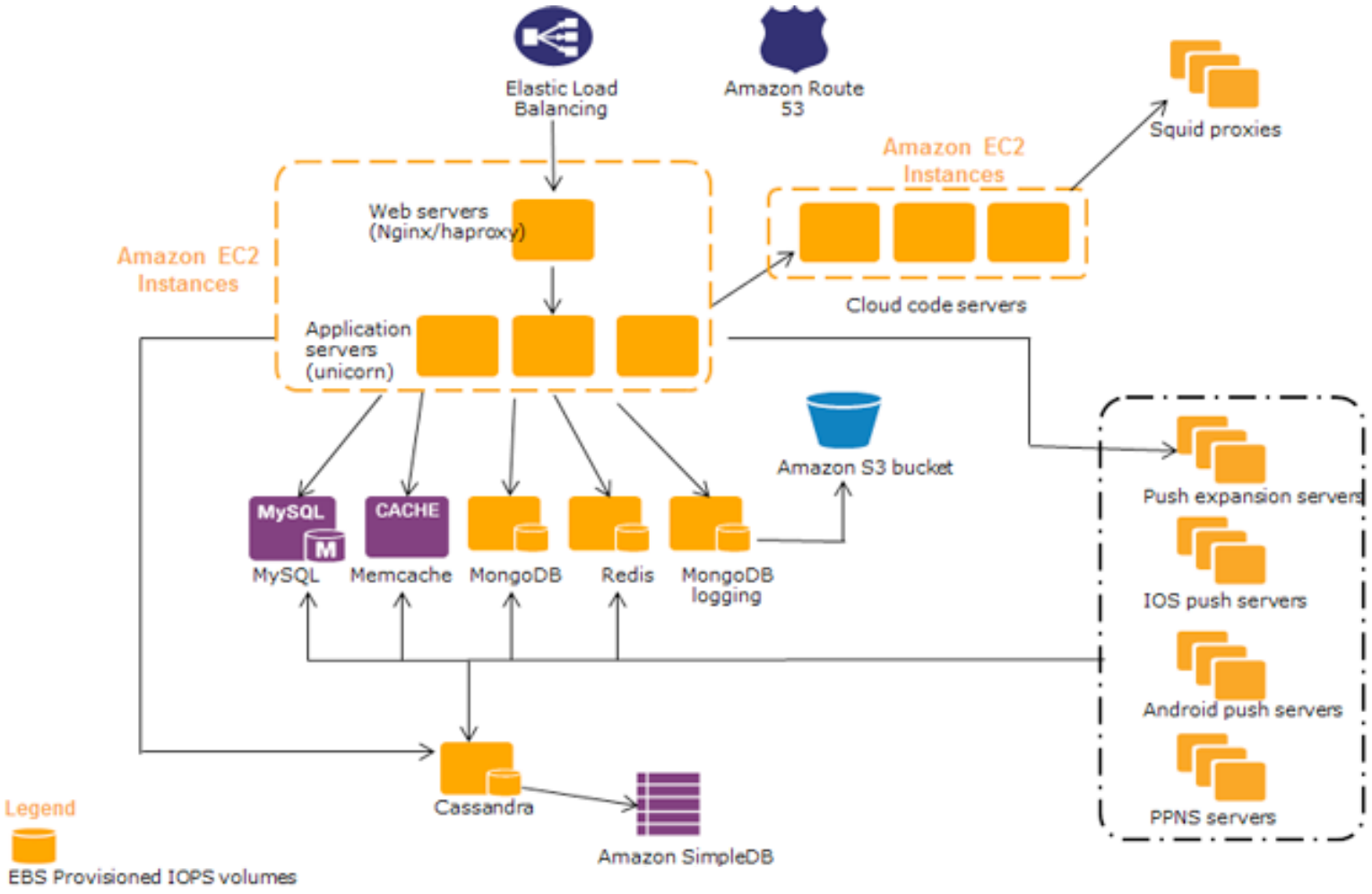
- Your architecture will be a mix of mobile, web app, services, and database
- Decisions: run your own data center, IaaS, or Paas?
- No brainer answer: AWS EC2
  - Keep an eye on that bill!

A Word About AWS

- Don't treat EC2 as a substitute to a data center or dedicated colocated servers
- Leverage spot and reserved instances
  - Otherwise your costs will balloon like mad!

AWS Gives You Everything

- At a price - be judicious



Pricing Horror Story

- Successful app - no capacity planning
- Daily bill? $70,000 USD / day

- Used all AWS stack services
- Lots of servers, all **regular** instances

Avoid Pricing Surprises

- Leverage spot and reserved instances
  - Spot = cheap if available when needed
  - Reserved = prepaid, much lower $
- Use Linux/open source wherever possible
- Understand the implications of using standard database, caching, etc. vs. using AWS's Elastic Cache, Dynamo, RDS, etc.
- AWS best? ELB, SSL termination

Typical Application Architecture

- iOS or Android
- App server
- Message broker
- Database
- Caching

Bad

iOS is the cool!

RoR, PHP, CherryPy - hip

Whassat??

MySQL, mongoDB, Dynamo, RDS

Later....

Success!

- You built a popular app
- You think/know you can scale because it's all "on the cloud"
- Nope! You'll have to rework a lot of stuff -- better plan ahead

Scalable Application Architecture

- iOS or Android
- App server
- Message broker
- Database
- Caching

iOS - better monetization

Mule Integration, Spring - robust

ActiveMQ, RabbitMQ

Neo4J, MySQL cluster,
mongoDB - NO Dynamo

Memcached, Redis

Managing Your Cloud

- Find the meanest, leanest, toughest, smartest macho hombre DevOps guy you can hire
- Chef, Puppet, Bcfg2
- Leverage Route 53
- Don't forget monitoring
  - Zabbix > Nagios > AWS monitoring
  - New Relic > AWS monitoring

Plan deployment via configuration - avoid AMI-based deployments! Hard and expensive to manage

App Interface

- Your mobile app talks to the servers via an API
- Your servers talk to one another over the same API
- Build around services, no tight coupling!

App Interface

- Data exchange? JSON
  - JASON-LA or other specilizations OKi
  - Don't be too granular
- Treat data as resources
  - RESTful
  - Just because you use HTTP it doesn't mean it's RESTful

App Interface

- All APIs must be stateless

raml.org

  - The mobile app or the server keep state, but no session management
  - Round robin load balancing
- Cache, cache, cache, and cache
  - Even if your DB supports all colors of “smart caching” - it won’t scale

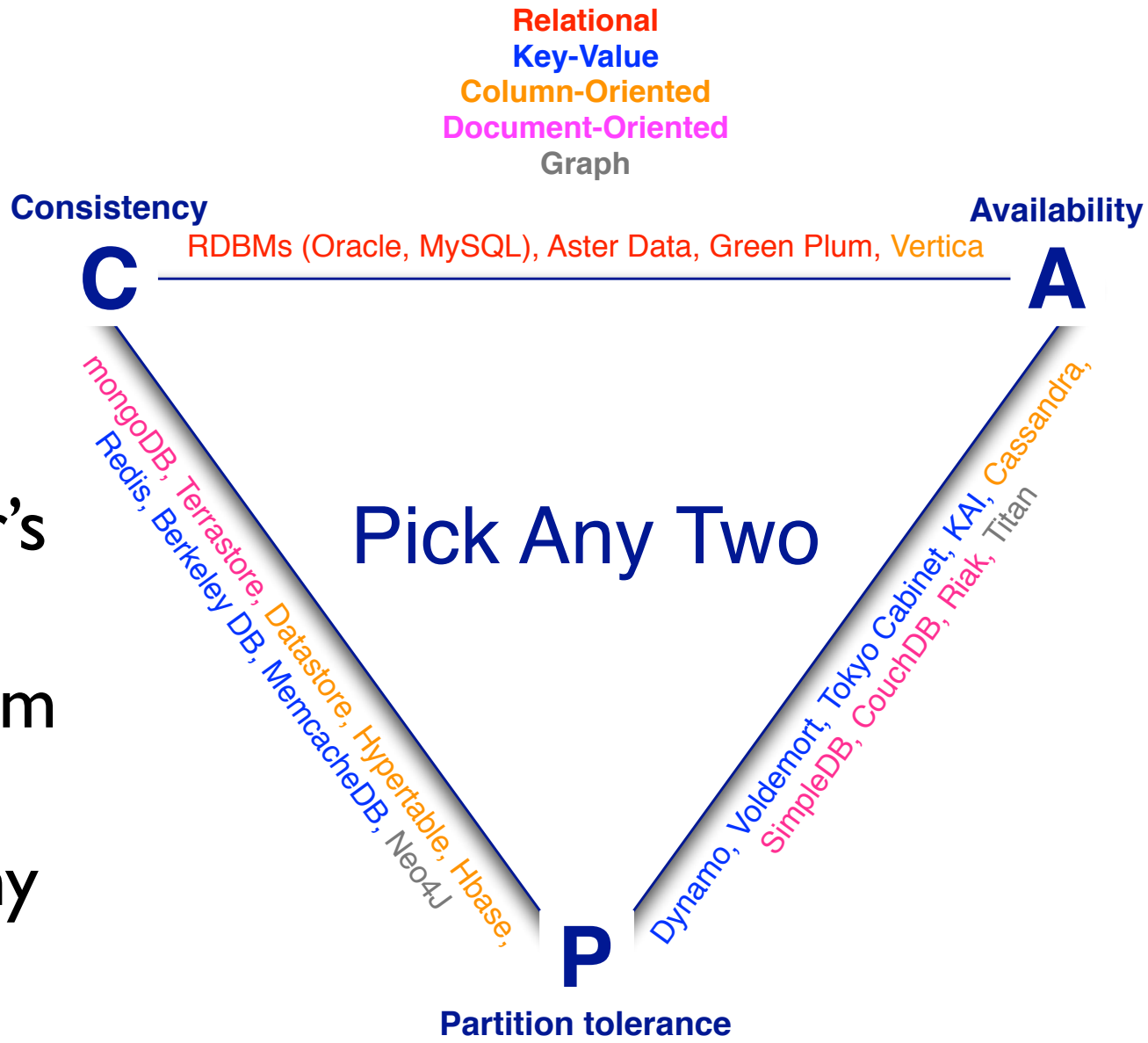
Database

- Define your data model well in advance and plan for massive growth
- All your operations must be designed and implemented for eventual consistency
- Think of full replication
- Use a DAO of some sort - don't talk to it directly

Database

Brewer's CAP Theorem

Pick any two!

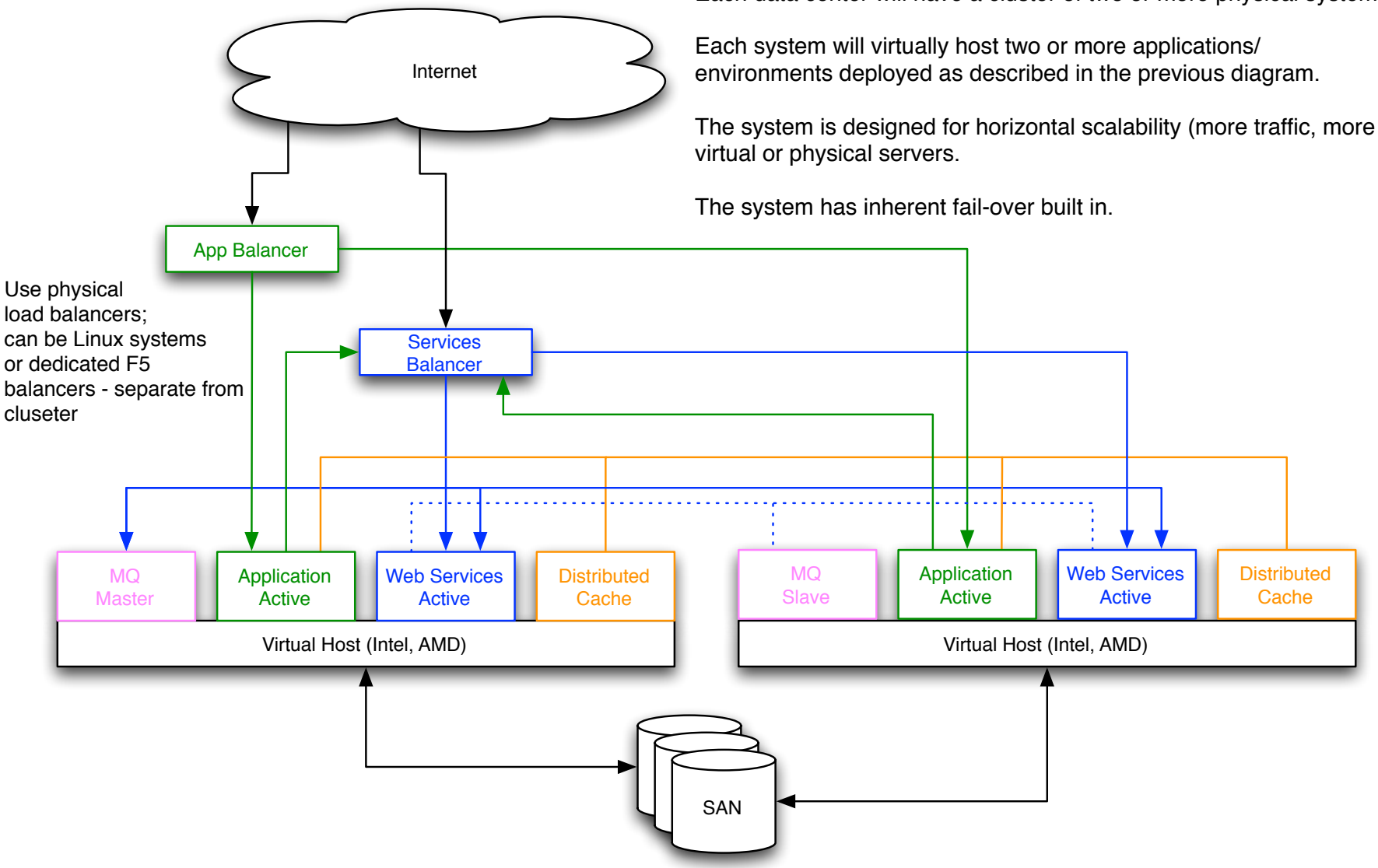


Caching

- Nobody's ever been fired for using Memcached
- Redis if the app needs access to collections, counters, and other complex data structures
- Roll your own servers - more management, but finer-grained control

Architecture - Future?

App and service requests
may come from the open Internet



Each data center will have a cluster of two or more physical systems.

Each system will virtually host two or more applications/ environments deployed as described in the previous diagram.

The system is designed for horizontal scalability (more traffic, more virtual or physical servers.

The system has inherent fail-over built in.

Almost Done

Are there any questions?

Mobile + HA + Cloud

Thanks for Coming!

Download this presentation from:
http://ciurana.eu/qcon2014/PEK/mobileHA

Eugene Ciurana

pr3d4t0r - irc.freenode.net
##java, ##security, #awk, #python, #bitcoin
irc.oftc.net: #tor, #tor-dev, #tails

qcon2014@cime.net