AcmeAir: Lessons learned from a large-scale cloud deployment

Paul Vytas, Gary Zeng

November 19, 2013

Table of Contents

Introduction

A cloud infrastructure holds a lot of appeal for testing:

- Quick access to a very large number of servers on short notice. Especially convenient for short-duration large-scale tests.
- Self-service, don't have to make requests to sys admin.
- Avoids lengthy capital planning cycle
- More efficient financially.. Only pay for what you use.

But there are some limitations as well:

- Shared environment - Less predictable performance (CPU, network, disk); can have unexpected outages.
- Less flexibility - standardized templates for I/O, CPU, memory, disk, OS
- New technology, components and tools are changing frequently.
- Need to plan for and regularly verify usage to stay within your budget.

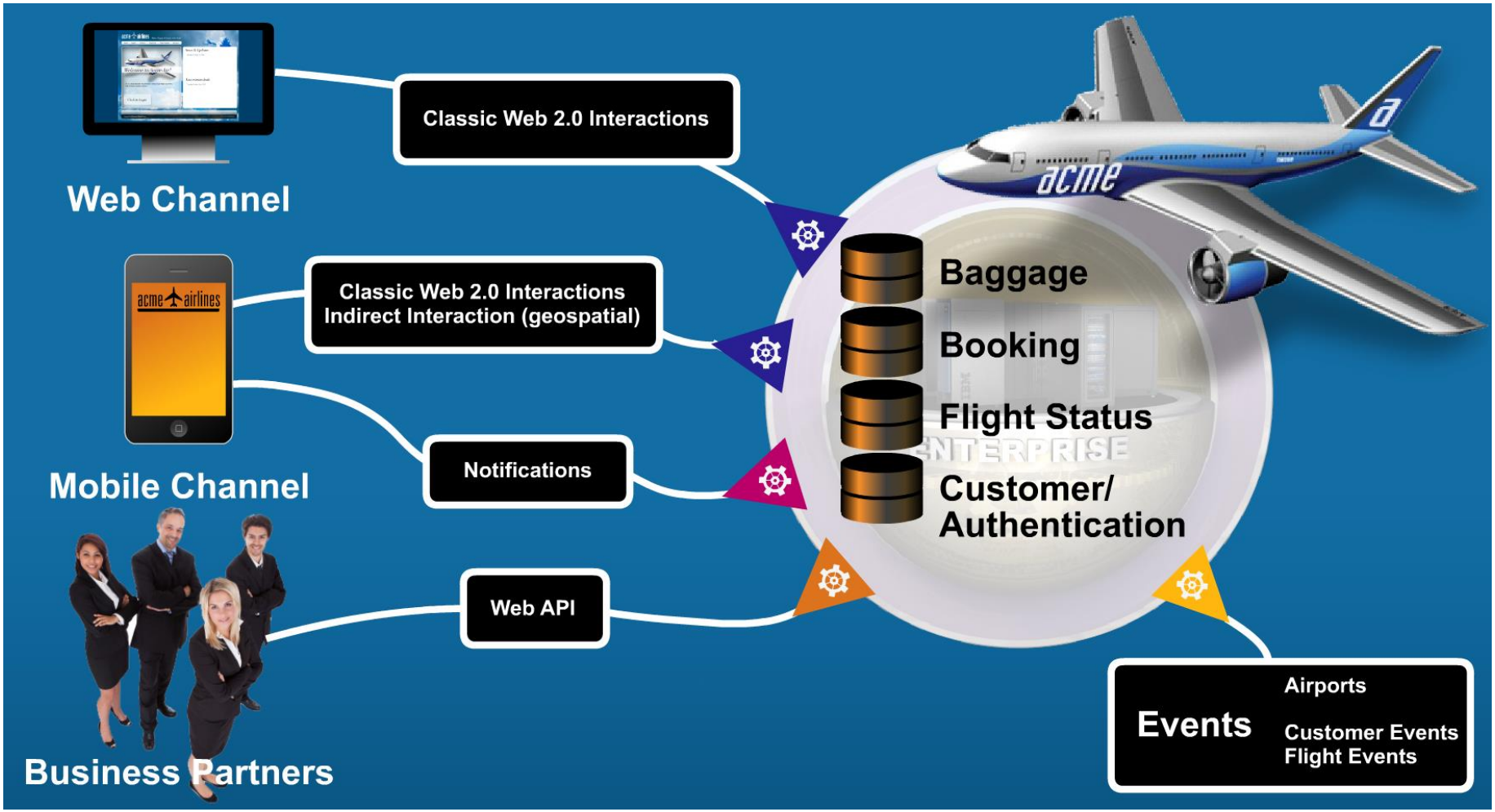
Acme Air application and benchmark

Acme Air Overview

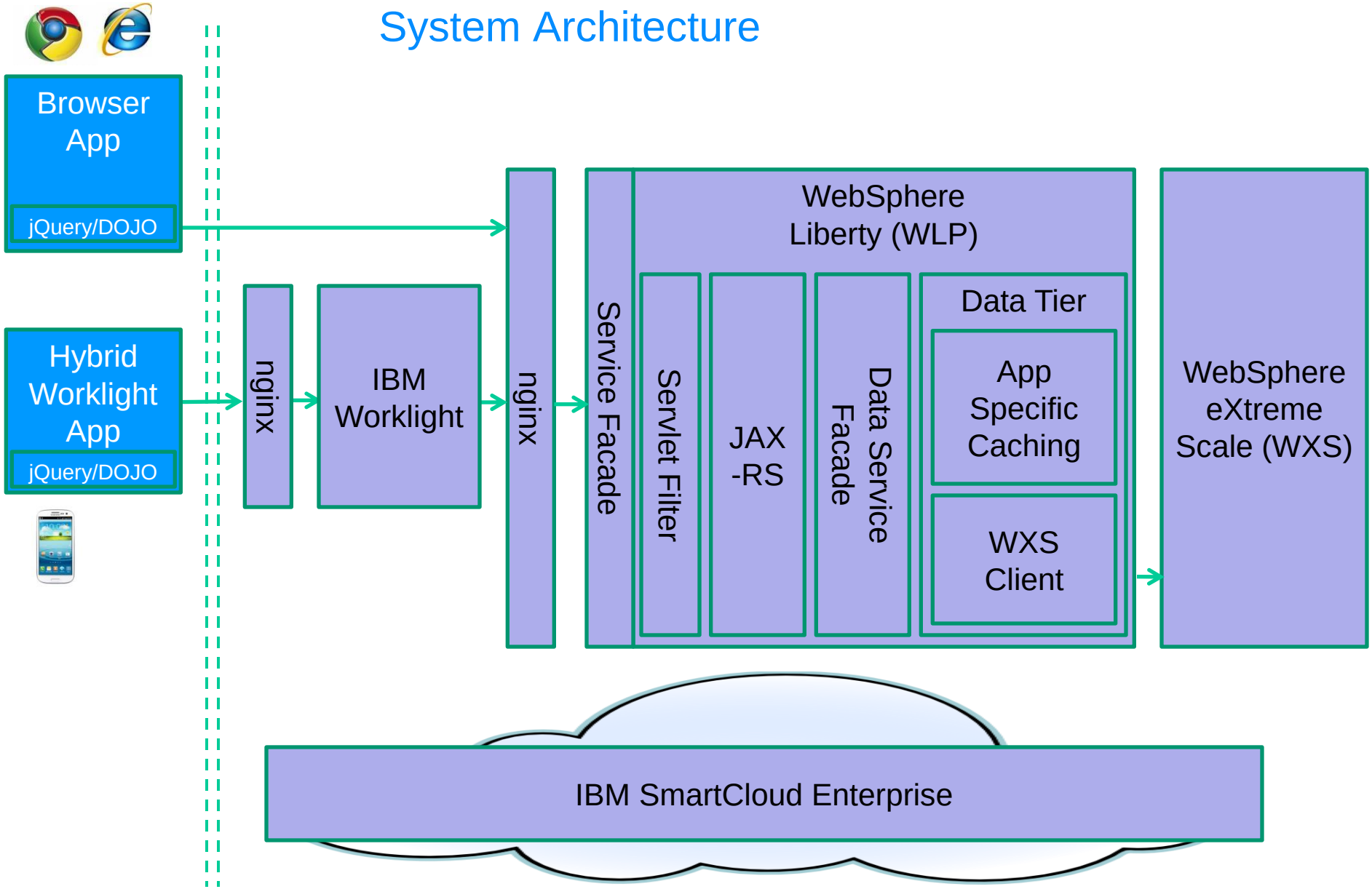
This application shows an implementation of a fictitious airline called "Acme Air". The application was built with the some key business requirements:

- the ability to scale to billions of web API calls per day,
- the need to develop and deploy the application in public clouds (as opposed to dedicated pre-allocated infrastructure), and
- the need to support multiple channels for user interaction (with mobile enablement first and browser/Web 2.0 second).

Acme Air Functionality

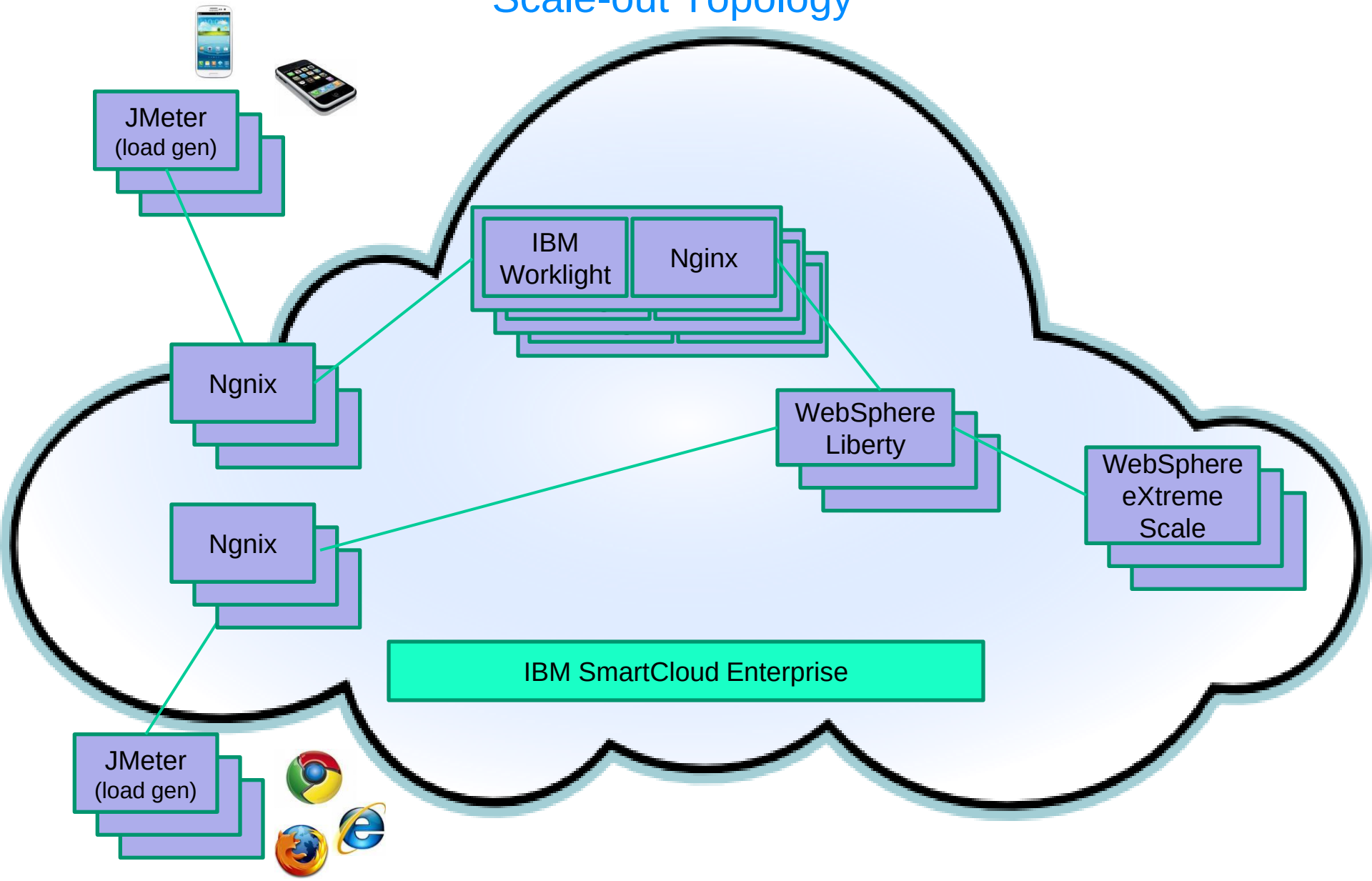


System Architecture



Large Scale Testing of Acme Air

Scale-out Topology



Performance and Scalability Results

API Billionaires Club

twitter	13 billion API calls / day *(May 2011)*
Google	5 billion API calls / day *(April 2010)*
facebook	5 billion API calls / day *(October 2009)*
NETFLIX	1.4 billion API calls / day *(May 2012)*
ACCU WEATHER	1.1 billion API calls / day *(April 2011)*
KLOUT	1 billion API calls / day *(May 2012)*
ebay	1 billion API calls / day *(Q1 2012)*
Sabre	1 billion API calls / day *(January 2012)*

Source: Programmableweb.com 2012

acme airlines

(the benchmark)

Avg/Min/Max Throughput
49,572/48,559/50,472 req/sec

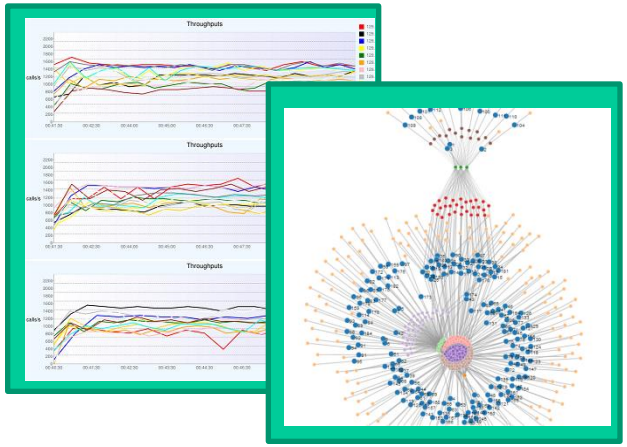
API calls per day: 4.3 Billion

Benchmark Start/End: 013/04/24 04:33:30/04:43:17

Type of Server	Number of Servers
WebSphere Liberty	51
WebSphere eXtreme Scale	47
IBM Worklight	28
Load Balancers	10

Can you stand up to the mobile channel?
Can you scale without bounds?
•Performance? Operations?

Learn how to with Acme Air
•Architecture, sample code, performance results, ops practices, and more



Best Practices and Lessons Learned

Automate some, Automation more, AUTOMATE EVERYTHING!

- Anything done by hand is a bug and won't web scale. Logging on each server and running a script is OK when you only have a few servers, but completely impractical for 100's of VMs.
- Anything that isn't under a version control system (or on instance disk image) is a bug and will create chaos

Scale out in steps

- This strategy did work as expected. We initially started with a minimum configuration (1 load driver, 1 app server, 1 datastore) then once it was working moved to 2-2-2, 4-4-4, 16-16-16, etc.
- After each step we adjusted the configuration to make sure we could load up the app servers as evenly to 95% CPU utilization and looked for bottlenecks
- During 1-1-1 did many things manually, but at 4-4-4 realized we could not do everything by hand. started adding more automation. Added load balancer after 4-4-4
- Cannot forsee what automation needs to be done up-front.. We incrementally added extra automation as we scaled out.

Get to know your cloud provider

- Understand your cloud provider's APIs
- Avoid surprises! Learn about maintenance windows, provisioning limitations, runtime performance, variability of performance
- may need to arrange agreement with provider for larger topologies
- may want get your VM's co-located if bandwidth is an issue.
- pitfalls of Public IP's for testing:
  - Scanners and attacks on your VMs can load them down and mess up testing results,
  - Likely will require additional effort setting up firewalls, etc.
  - Many cloud providers can arrange for a private subnet in cloud but this costs.

Assume that a VM can fail at any time

- With a large number of active VM's, some failures are inevitable. This can be deployments that fail, VMs were not fully alive or die during a test run or an application that hangs.
- Failed VMs can be fixed by killing and re-creating but failures can ruin a test run. Suggest a sanity-test prior to starting a big test run.
- Monitoring is critical, but difficult when there are 100's of VMs.. Tools are still evolving and we needed to create some custom automation.
- need to be able to detect imbalances, image not working as it should, VM mis-behaving
- need to track state of VMs.. some will take more time to come up.
- need to automate log collection and diagnostic/troubleshooting steps.. with 100's of VMs we cannot do this by hand.
- ..

Have a strategy for doing updates

- Inevitably during the testing and scale-out process changes will be needed for configuration files and code.
- Discovered that re-generating VM images then restarting 100's of VMs can take a long time. For small changes it was much faster to roll out application or configuration changes as needed to VMs. Of course this needs to be automated.

Know your costs and keep track of usage

- pay-as-you-go is convenient, and single VM's are cheap (as low as 6 cents/hr per VM) but costs can add up fast. Make sure that your provider plan matches your goals and your budget.
- Provision for start of test, then deprovision at end of test.
- Shut down instances when not being used, reprovision stalled VMs.
- Use private IP's when pushing load (public IP's can cost you I/O charges)

Summary of Lessons Learned

- Automate some, Automation some more, AUTOMATE EVERYTHING!
- Scale out in steps
- Get to know your cloud provider
- Assume that a VM can fail at any time
- Have a strategy for doing updates
- Know your costs and keep track of usage

Acknowledgments & References

Many thanks to…

The IBM team that created AcmeAir and its initial cloud deployment:

- Andrew Spyker - Performance Architect and Strategist, Emerging Technology Institute
- Doug Tollefson - WebSphere Application Server Performance
- Gary Zeng - Software developer, WebSphere eXtreme Scale
- Jeffrey Garrat - Senior Software Engineer - WebSphere Technology Institute
- Yang Lei - Senior Software Engineer, Emerging Technology Institute

References

- Github for getting a copy of AcmeAir:
  https://github.com/acmeair/acmeair
- Github Wiki describing how to install and run the AcmeAir application:
  https://github.com/acmeair/acmeair/wiki
- Andrew Spyker's blog on cloud issues and AcmeAir activities:
  http://ispyker.blogspot.ca/

Moss Uchida and Roland Gee are showing demos of AcmeAir in the CASCON Technology Showcase in booth L08