Application Architecture for the Cloud

Steve Loughran
Julio Guijarro

HP LABS
Automated Infrastructure Lab


My other computer
is a data center

Why move to *the cloud?*

Good

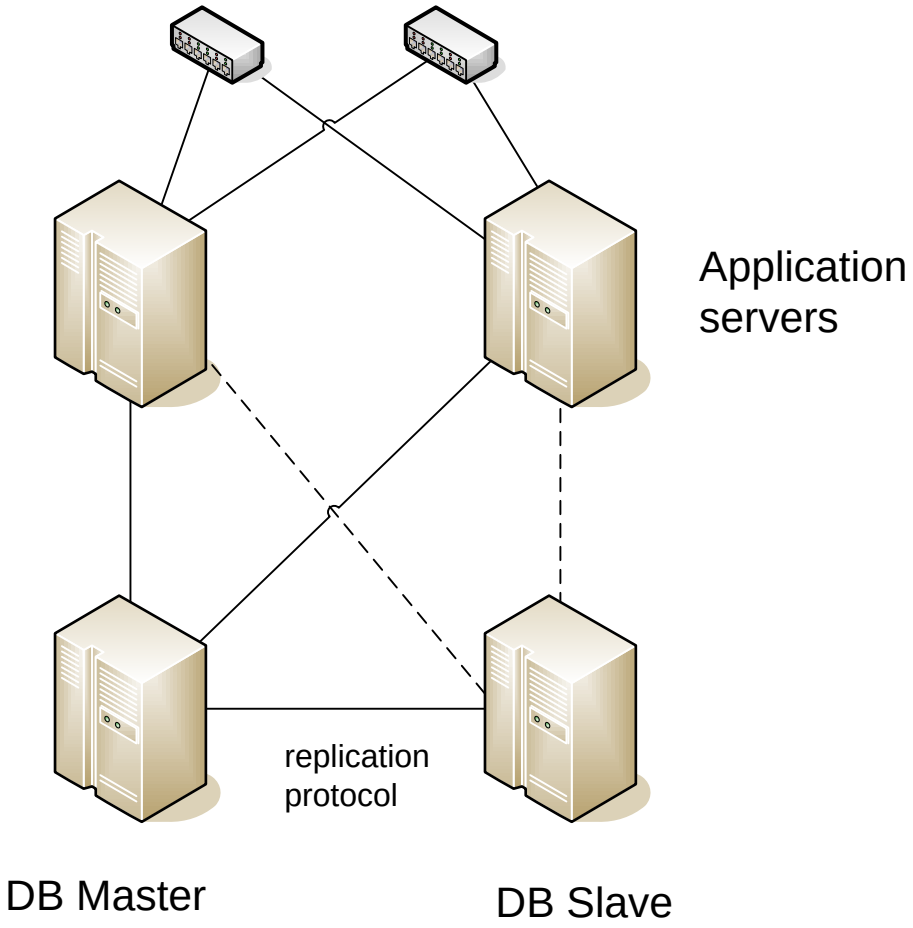
- Cost
- Outsourcing hardware problems
- Move from capital to pay-as-you-go
- To handle Petabytes of data
- For a business plan that might work

Bad: to avoid the operations team

What is a cloud application?

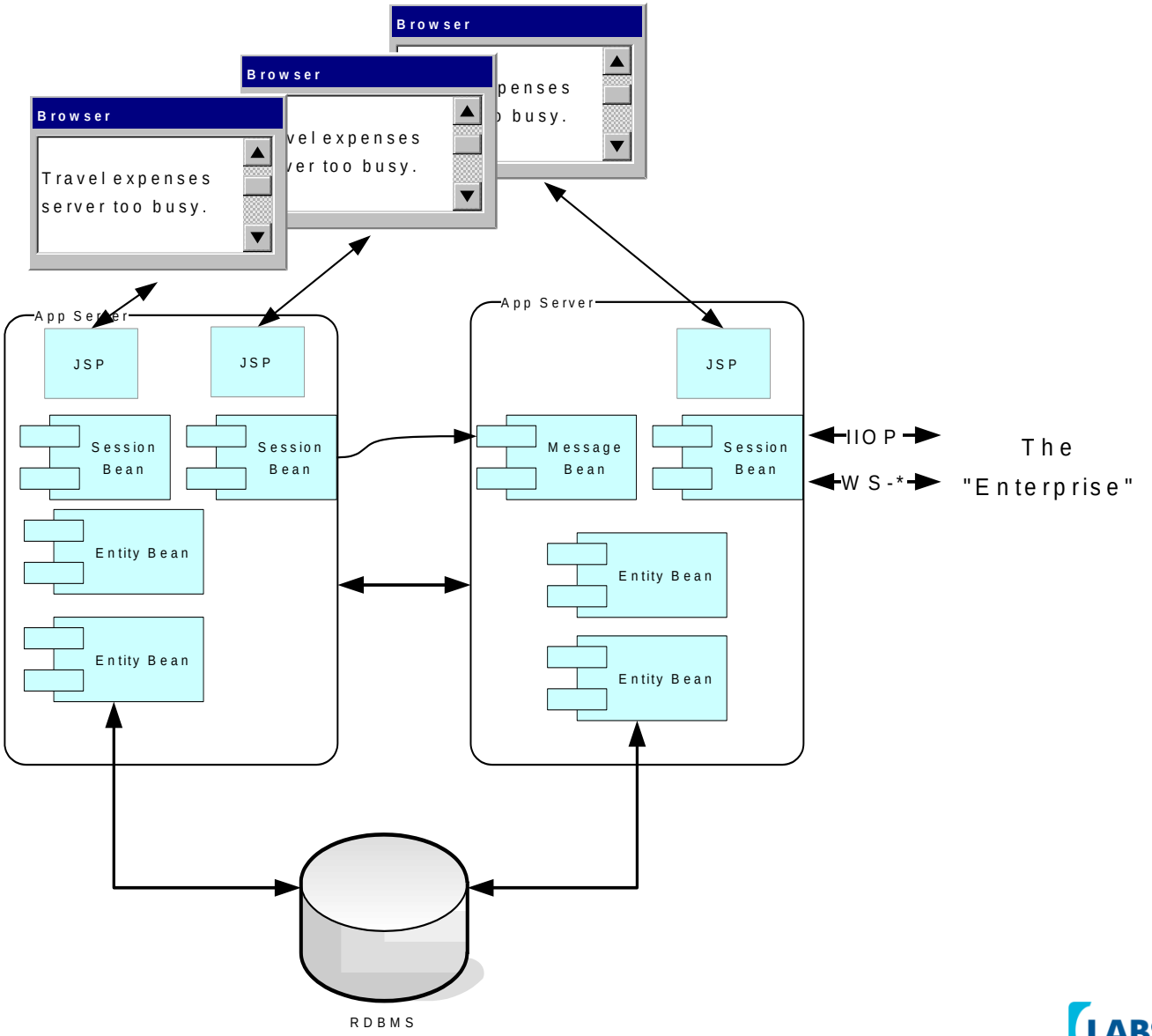
- The program that is run
- The code/data needed to run it in a datacentre
- Anything needed to configure, monitor and manage the system

This is not a cloud application



Enterprise Java on Highly Available servers

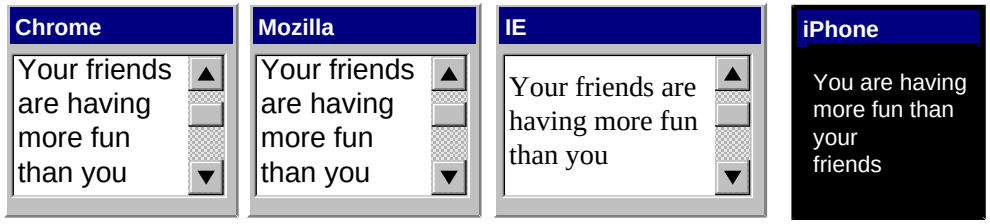
This is not a cloud application: Java EE

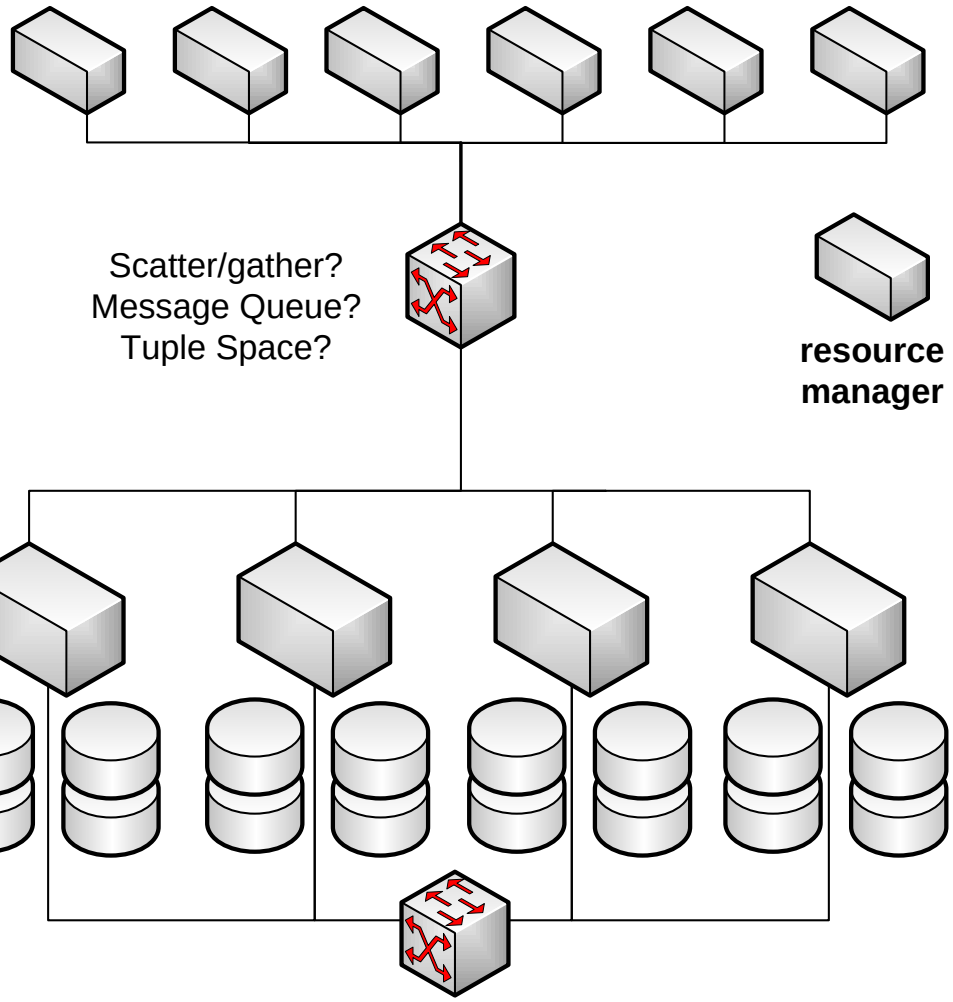


Things must change!

- Web UI for users, affiliates, marketing, operations
- Agile machine management is part of the API
- Scale up -and down
- Live upgrade of running system
- Persistence with key-value stores
- A Petabyte filesystem is part of the application
- MapReduce jobs close the loop
- Developers deploy to the cloud to test

REST APIs





Diskless front end

Memcached
JSP?
PHP?

Back End

HBase/Cassandra/
CouchDB
MapReduce
Lucene
HDFS or ...

EC2 as the cloud provider

- *S3* for persistence, public downloads
- *SimpleDB* provides key-value storage, but query costs unpredictable.
- *Typica* for EC2 services API
  http://code.google.com/p/typica/
- AWS IP rental or Dyndns for hostnames
- No image management services

No standard Apache Stack

Sun, IBM, HP as the cloud providers

- *S3* -like filestore
- EC2 and Sun RESTy APIs
- Unknown queue and keystore services
- More secure networking?
- Image management services?

No standard Apache Stack

Private cloud

- *Eucalyptus for deploying Xen images*
- Various persistence options
- Private filestore: HDFS, kfs, Lustre
- Kickstart for image management?

No standard Apache Stack

Whoever owns the API owns the application for its life

Whoever owns the data owns you

Apache Cloud Computing Edition

- Diverse mix of high-level technologies
- Very large filestore at the bottom :
  Hadoop APIs, Java 7 NIO, Fuse, WebDAV
- MapReduce phase for post-processing
- We need stories for :
  persistence, configuration, resource management

A standard Apache Stack!

Front End

- The existing Web Front ends should work:
  Servlets, JSP, wicket, PSP, grails
  (maybe with memcached)
- Glue: queues, scatter-gather, tuple-space, events

- Everything needs to handle an agile world
- Everything needs instrumenting for management

Persistence

SimpleJPA

?

PNUTS/Sherpa

Cassandra

LABS hp

Everything needs a REST API

- REST is the long-haul API -why have a separate internal one?
- Jersey JAX-RPC is very nice
- Client API evolving
- Http Components/HttpClient can be the foundation for the Apache client; needs AWS support.
- Also: Restlet

Events and messages

- "disk 3436 is failing"
- Bluetooth phone 04:5a:1f:c2:87:91 entered cell 56 in London NW2
- Queued purchases with card numbers

internal and external events:
reliability, scalability, triggered actions

Resource Management?

1. HA resource manager to monitor front end/back end load and request/release machines on demand
2. Kill unhealthy nodes (liveness, performance)
3. Programmable policies (money vs. load)
4. Choreograph live upgrade/migration
5. Resource Manager as a service

Configuration & Management

- LDAP (and APIs)
- key-value stores

- SmartFrog moving to Apache license
- What is Spring planning in this area?

Development

- How to build and test in this world?
- How to step through a program running on a remote datacentre?
- How to control testing costs?

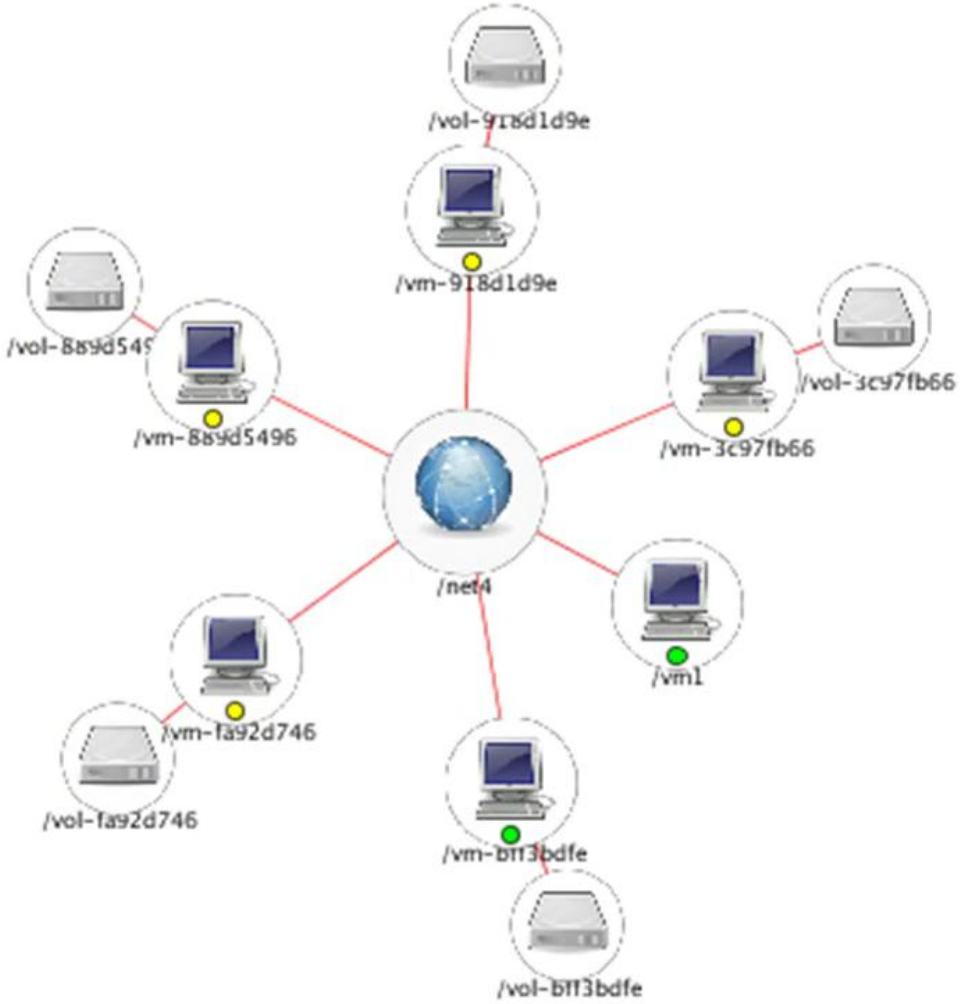
Eclipse won the Java IDE battle - It now needs to compete with Visual Studio + Azure

Testing -your first terabyte of data

```
protected void map(Text key, Text test, Context context)
        throws IOException, InterruptedException {
  TestResult result = new TestResult();
  Class<?> testClass = loadClass(context, test);
  Test testSuite = JUnitMRUtils.extractTest(testClass);
  TestSuiteRun tsr = new TestSuiteRun();
  result.addListener(tsr);
  testSuite.run(result);
  for (SingleTestRun singleTestRun : tsr.getTests()) {
    context.write(new Text(singleTestRun.name),
            singleTestRun);
  }
}
```

Lots of opportunities here!

Testing -the infrastructure can help



Pseudo-RNG driven cluster configuration

Cirrus Cloud Testbed?

- HP, Intel, Yahoo!, universities
- Heterogeneous, multiple datacentres
- Offering datacentre time, not specific apps
- Low-level API for physical machines
- Cloud-API for virtual machines
- Paying customers? *No, not yet.*
- Open source projects? *I hope so*

What next?

Apache has the core of a Cloud Computing stack

How do we take this and:

- integrate the various pieces?
- extend them where appropriate?
- provide an alternative to AppEngine and Azureus?

Call to action

- Stop writing EJB apps
- Start collecting as much data as you can and feeding that MapReduce mining-phase.
- Design for: distributed not-quite-Posix filesystem, message queues, name-value databases

Apache: let's build our own cloud platform


LABS hp

VM Image Management

- AMI Image sprawl: 10%/month
- Old images are a security risk
- The whole PXE+Kickstart process is built for physical machines.

This is not an Apache problem, but we'll need to work with the OS vendors & others to integrate

HDFS improvements

- Scale, availability, small files: hierarchical namenodes?
- Could it be a general purpose media store?
- For web sites?