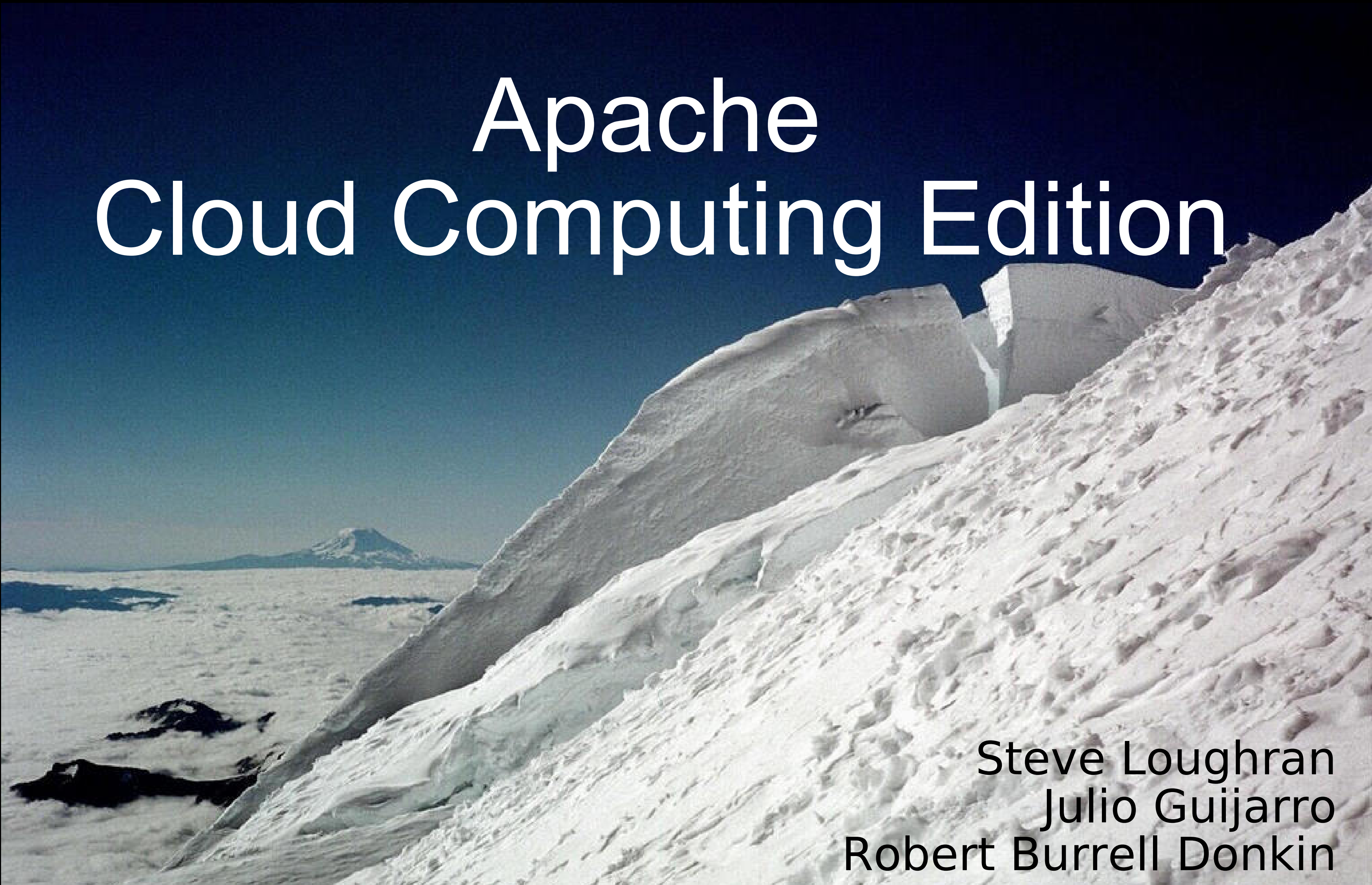


Apache Cloud Computing Edition

Steve Loughran
Julio Guijarro
Robert Burrell Donkin

Why move to *the cloud?*

Good

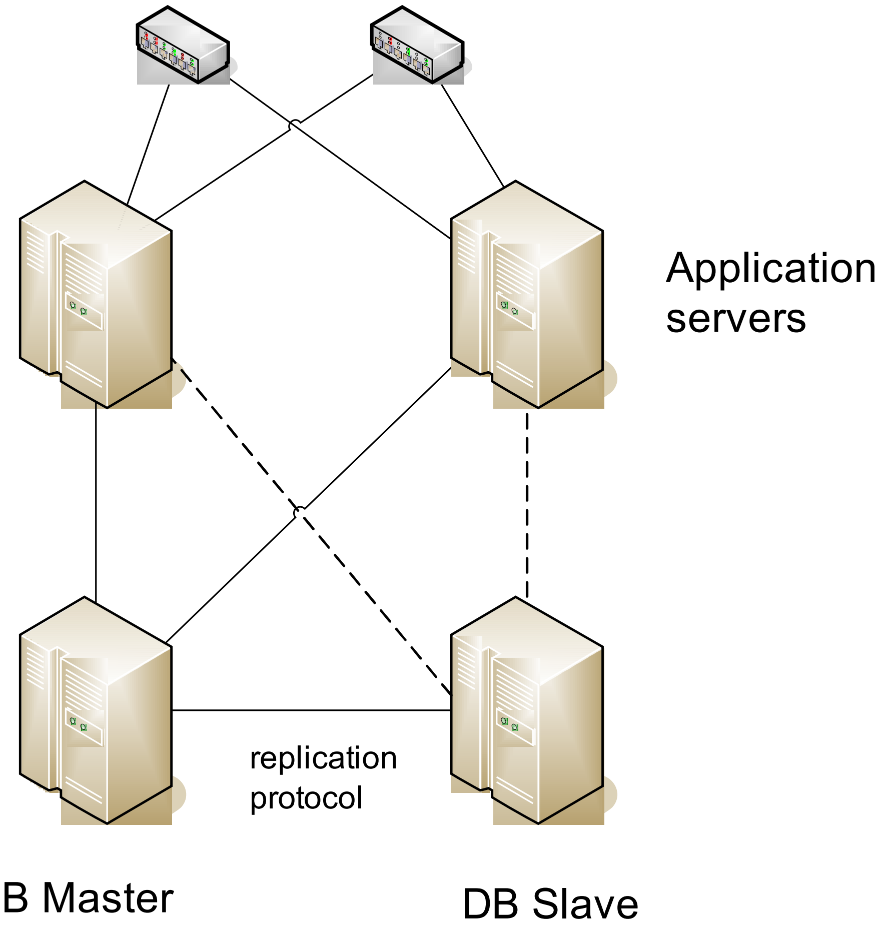
- Cost
- Outsourcing hardware problems
- Move from capital to pay-as-you-go
- To handle Petabytes of data
- For a business plan that might work

Bad: to avoid the operations team

What is a cloud application?

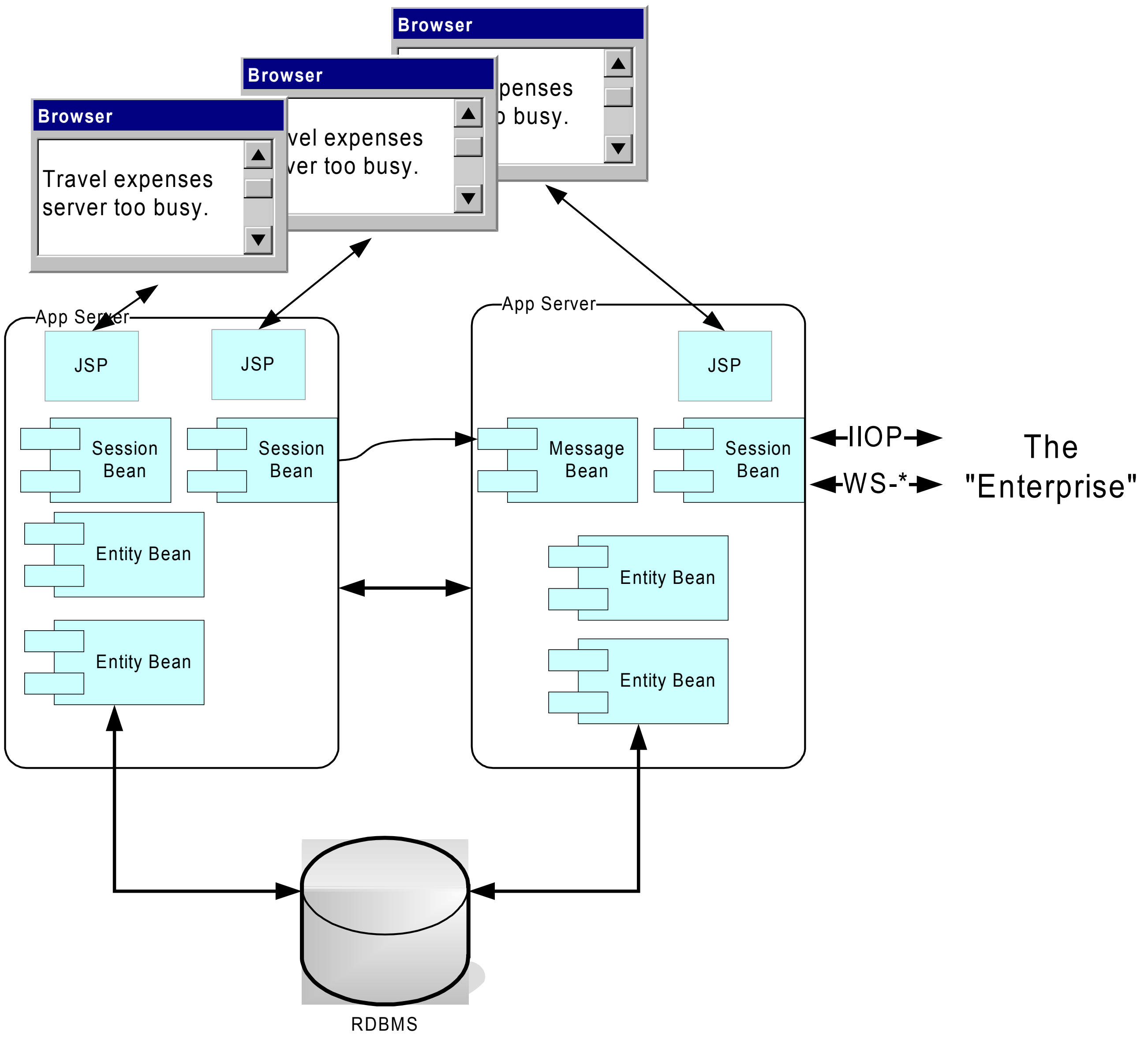
- The program that is run
- The code/data needed to run it in a datacentre
- Anything needed to configure, monitor and manage the system

This is not a cloud application



Enterprise Java on Highly Available servers

JavaEE is not a cloud application

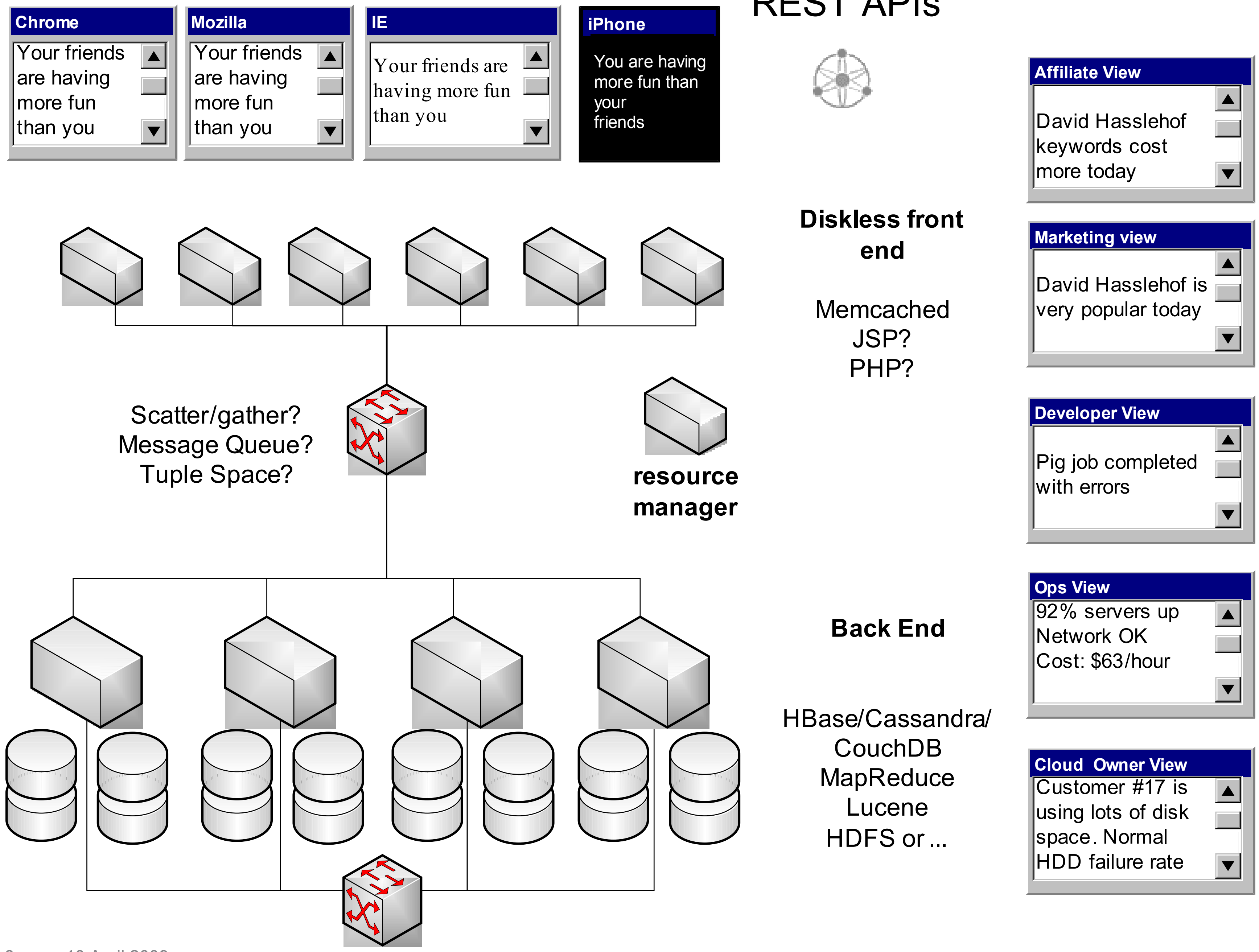


Things must change!

- 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

The agile will survive

REST APIs



Chrome
Your friends are having more fun than you

Mozilla
Your friends are having more fun than you

IE
Your friends are having more fun than you

iPhone
You are having more fun than your friends

Affiliate View
David Hasslehof keywords cost more today

Marketing view
David Hasslehof is very popular today

Developer View
Pig job completed with errors

Ops View
92% servers up
Network OK
Cost: $63/hour

Cloud Owner View
Customer #17 is using lots of disk space. Normal HDD failure rate

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 billing or test APIs
- 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?
- Billing and monitoring?
- Testing?
- 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

CouchDB
relax

SimpleJPA

Hive

Amazon SimpleDB™ BETA

?

H·BASE

YAHOO!®

PNUTS/Sherpa

Cassandra

HYPERTABLE

Everything needs a REST API

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

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

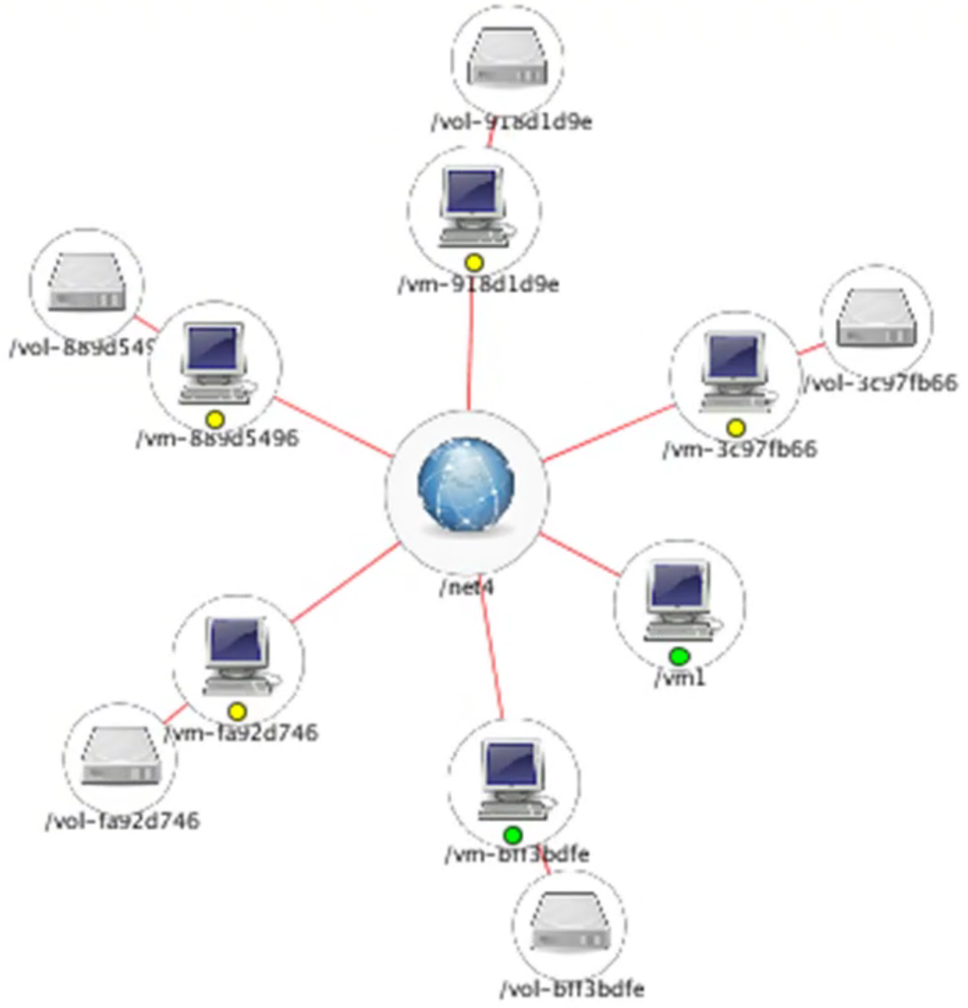
- How to build and test in this world?
- How to step through a program running on a remote datacentre?
- How to control testing costs?

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? We *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 filesystems, message queues, name-value databases

Apache: let's build our own cloud platform

Let's build Apache's Cloud Stack

VM Image Management

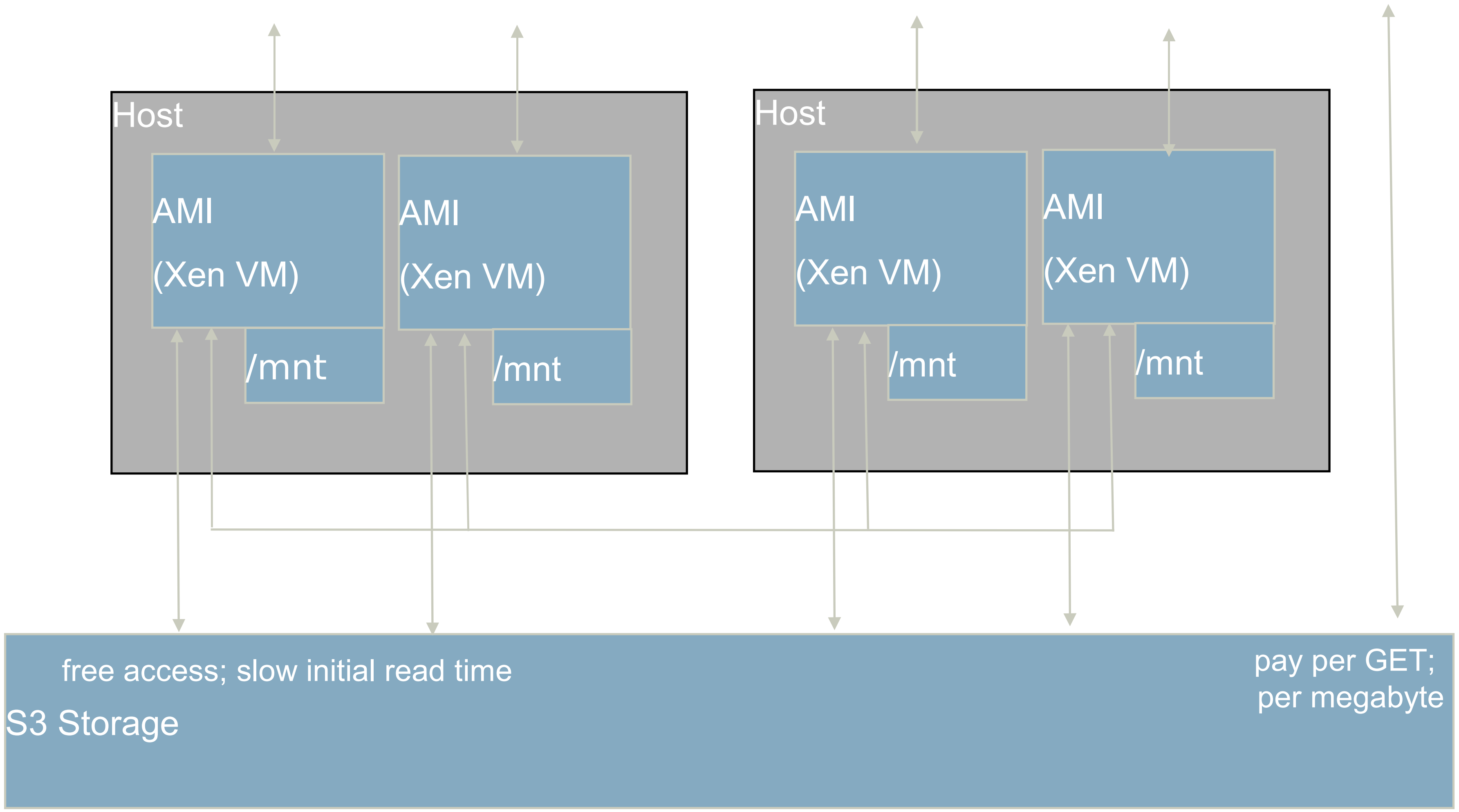
- AMI Image sprawl: 10%/month
- Old images are a security risk
- The whole PXE+Kickstart process is built for physical machines.

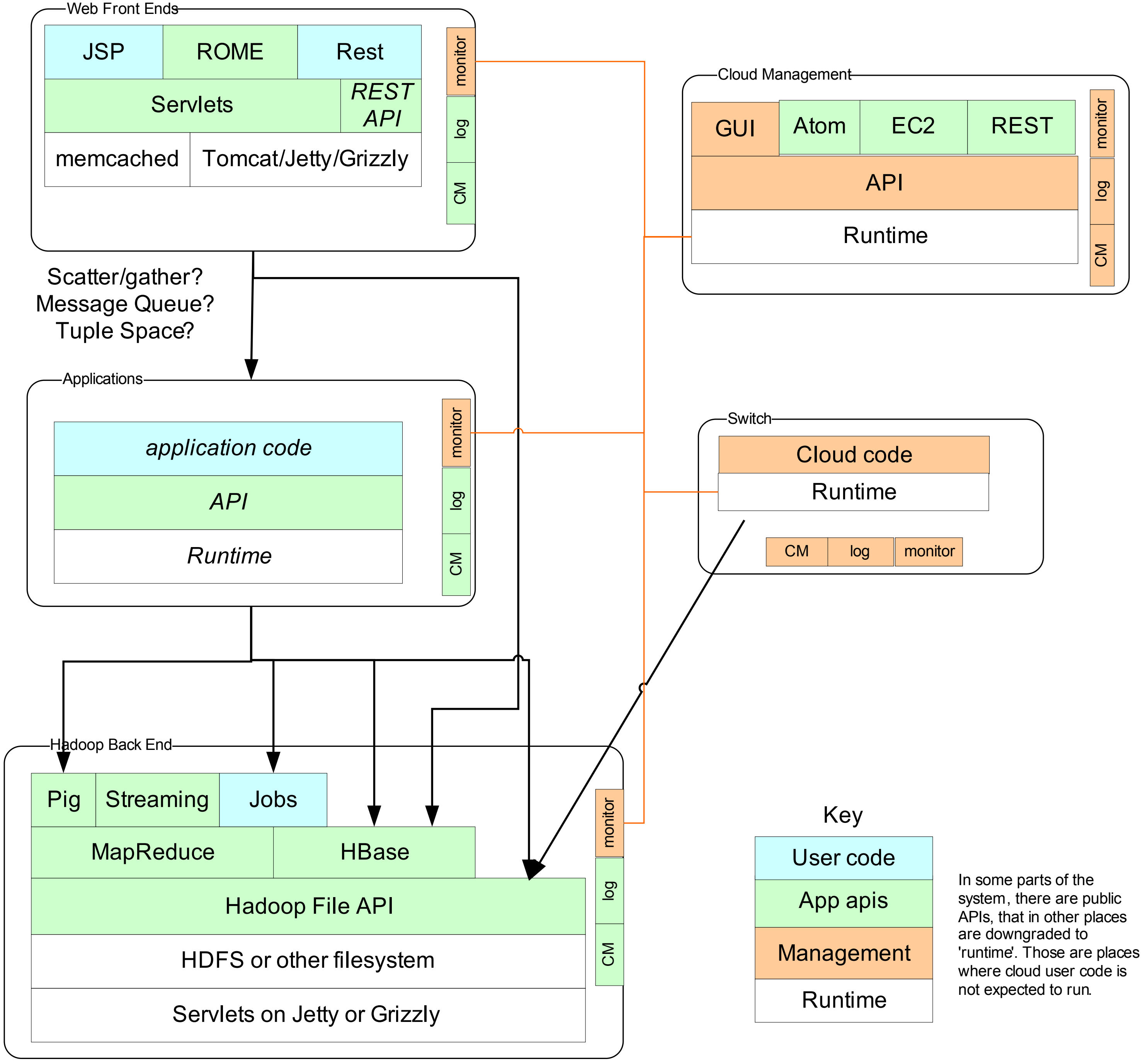
This is not an Apache problem, but we'll need to work with the OS vendors & others to integrate

HDFS improvements

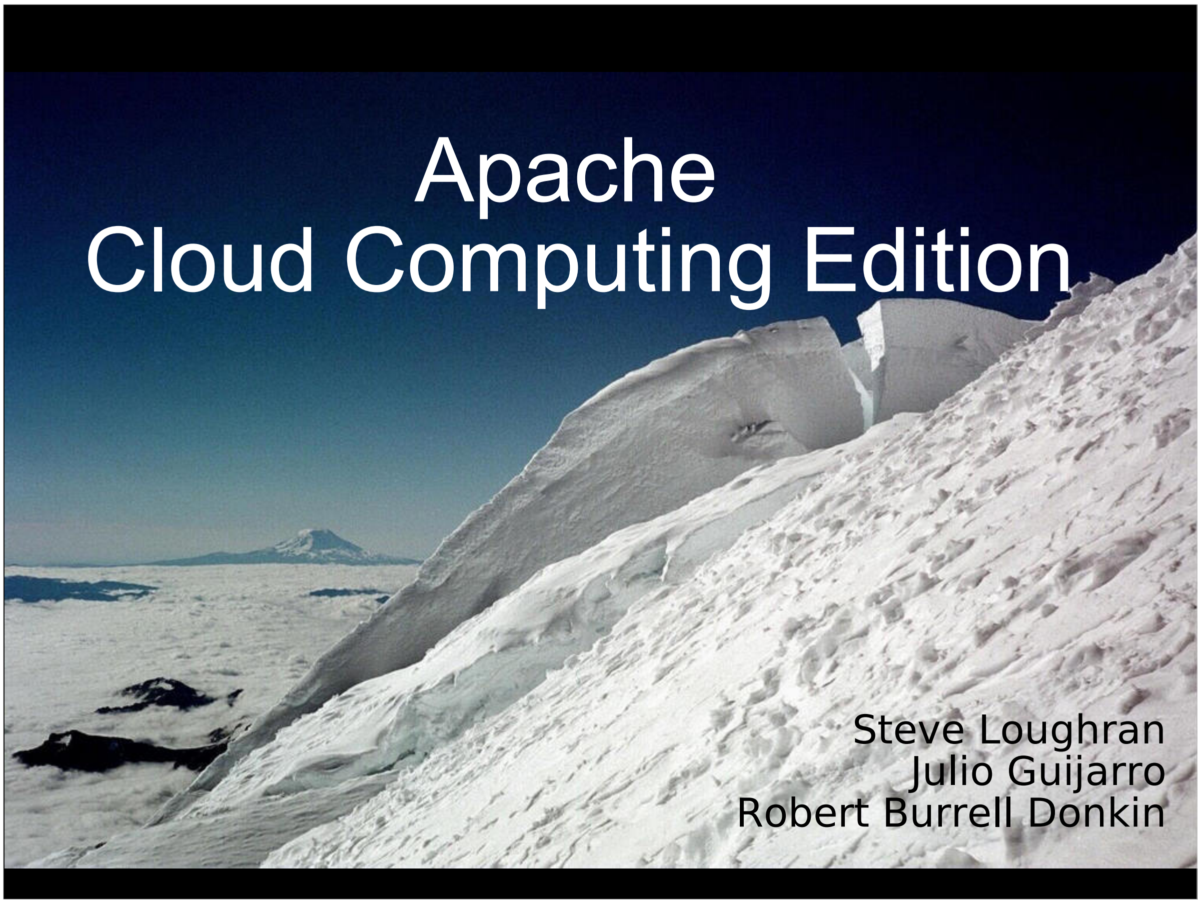
- Scale, availability, small files: hierarchical namenodes?
- Could it be a general purpose media store?
- For web sites?

Amazon EC2


Host
AMI
(Xen VM)
/mnt
AMI
(Xen VM)
/mnt
Host
AMI
(Xen VM)
/mnt
AMI
(Xen VM)
/mnt
free access; slow initial read time
S3 Storage
pay per GET;
per megabyte



In some parts of the system, there are public APIs, that in other places are downgraded to 'runtime'. Those are places where cloud user code is not expected to run.



Apache Cloud Computing Edition

Steve Loughran
Julio Guijarro
Robert Burrell Donkin

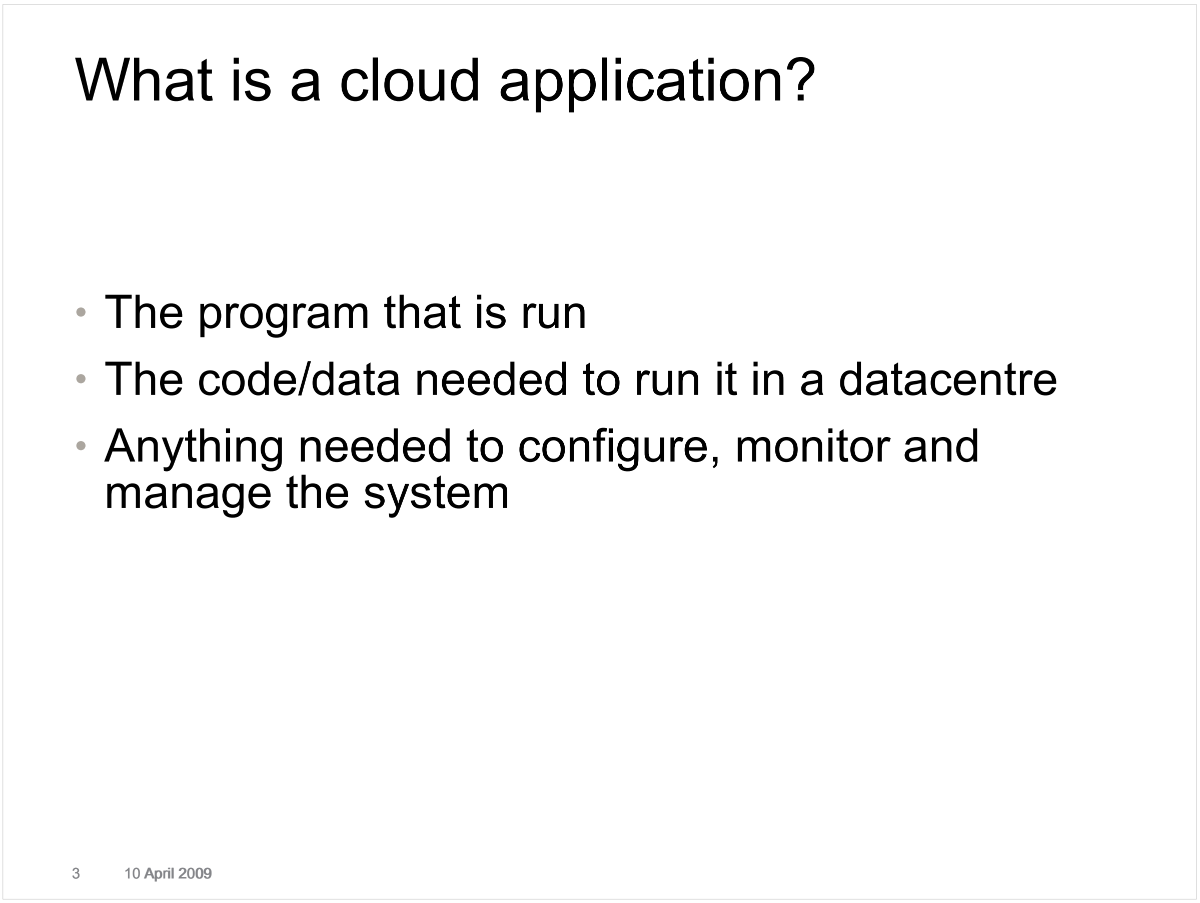
This is a talk on application architecture for the cloud.

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

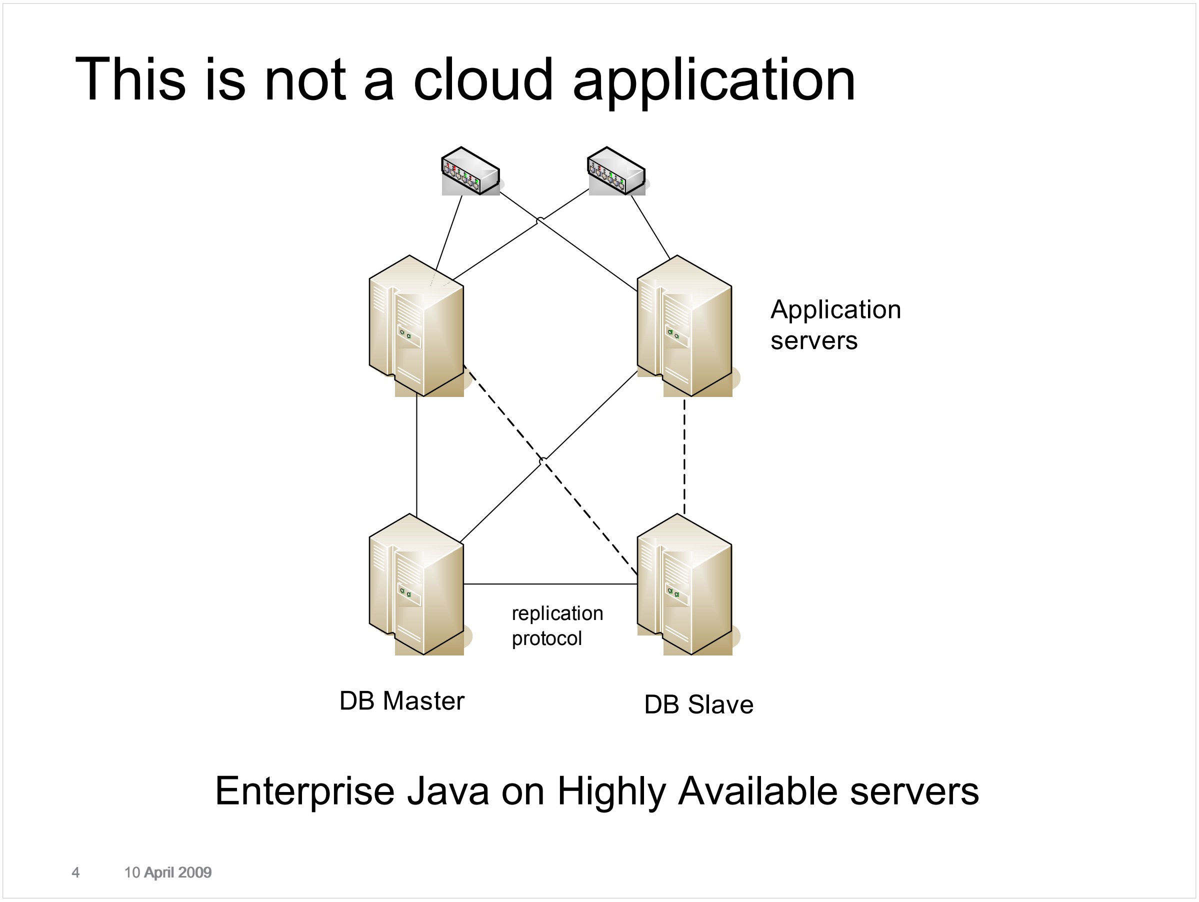
Bad: to avoid the operations team



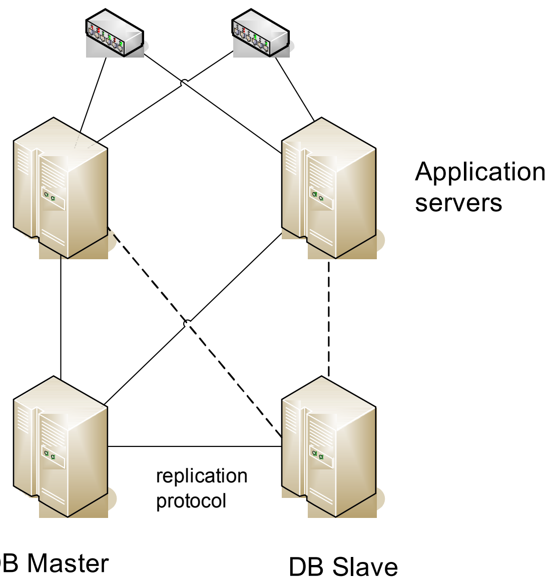
What is a cloud application?

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

We've often tended to split the "application" from the installation, because that installation usually included buying hardware, connecting it together, etc. Not any more. Now the servers and storage are just procedural or declarative instructions in a different file in the repository. Which means you need to look at everything together.



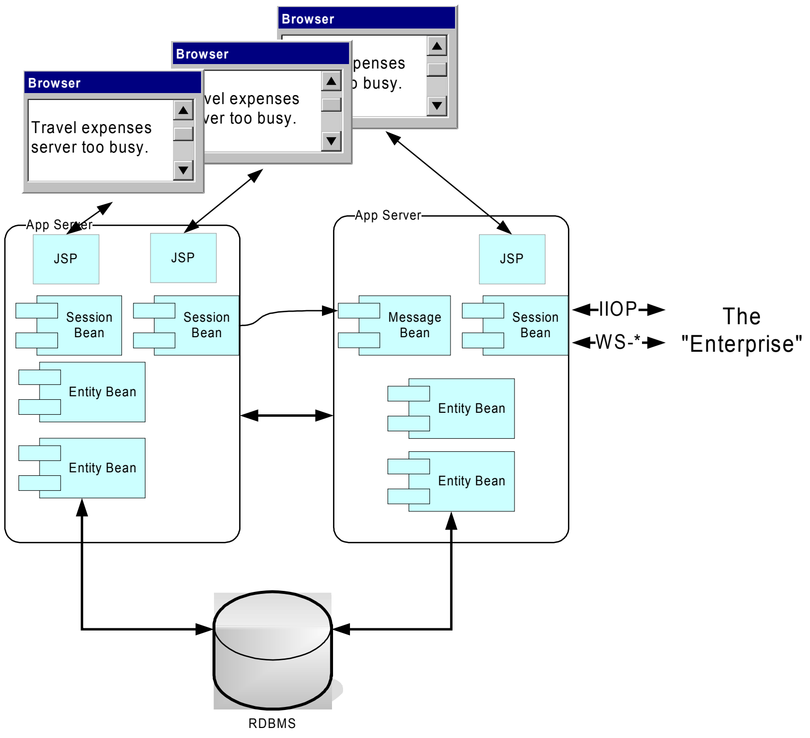
This is not a cloud application



Enterprise Java on Highly Available servers

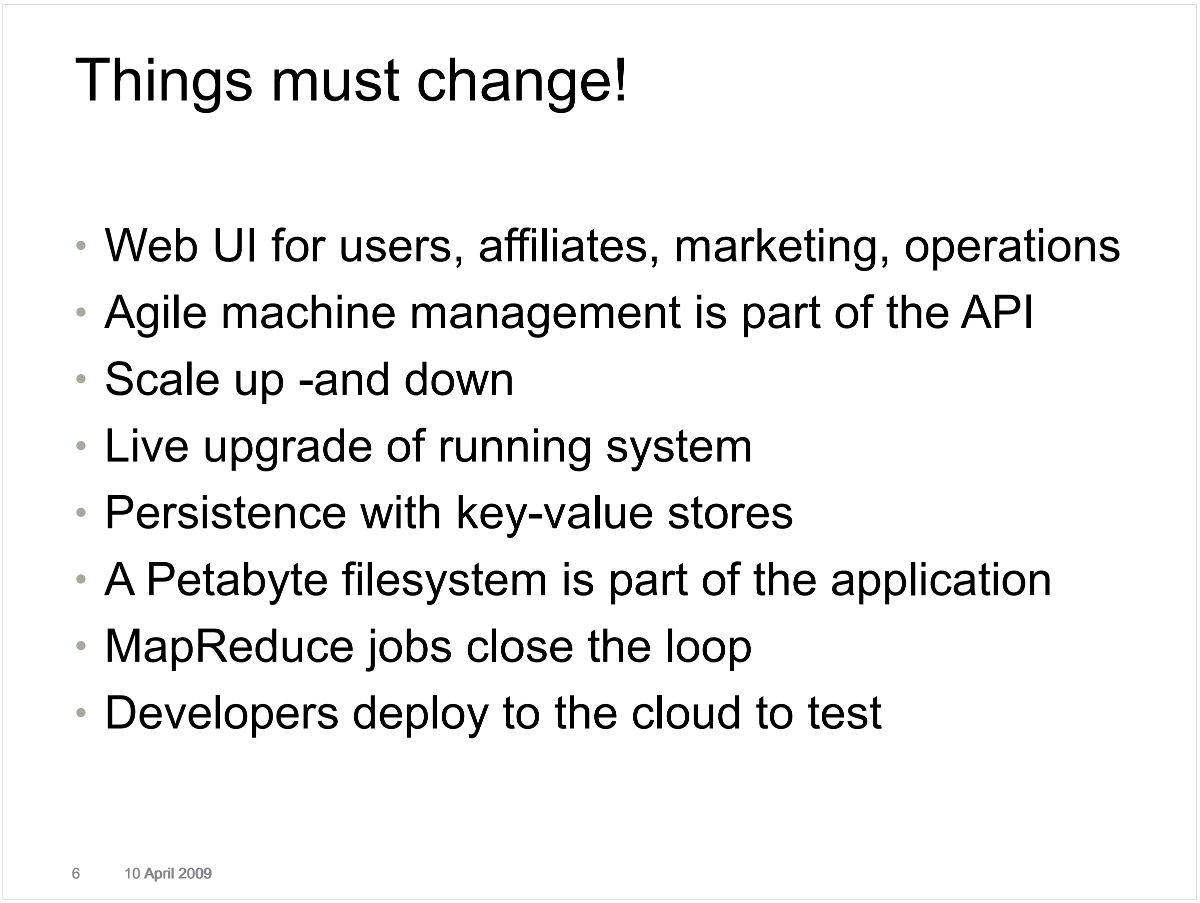
You can still run it on the cloud if you host it in a VM infrastructure that make the "cloud" look like physical machines. This is what enterprises do as it reduces hardware costs of maintaining machines for single apps, especially ones that don't get used very often. But you still have software maintenance and OS update costs, the latter is now a function of the #of OS images you have, not the #of physical machines

JavaEE is not a cloud application



Not picking on any particular server here; just the small-cluster architecture which Java EE excelled at.

2. You have lots of beans to hide complexity of databases and web services from your developer, as XML and SQL are considered too complicated for them.
3. The database handles replication, RAID-5 delivers data security; the app server keeps the beans consistent
4. WS-* and IIOP delivers good integration with other applications in the infrastructure that are somewhat controlled. It is possible to get all the architects into a phone conference and for them to agree on which WS-* specs to use, what the single-signon protocol will be (often WinNT auth, maybe via LDAP), etc.
5. In house -your customers can't walk away. Externally, you may have scale problems at times of popularity.



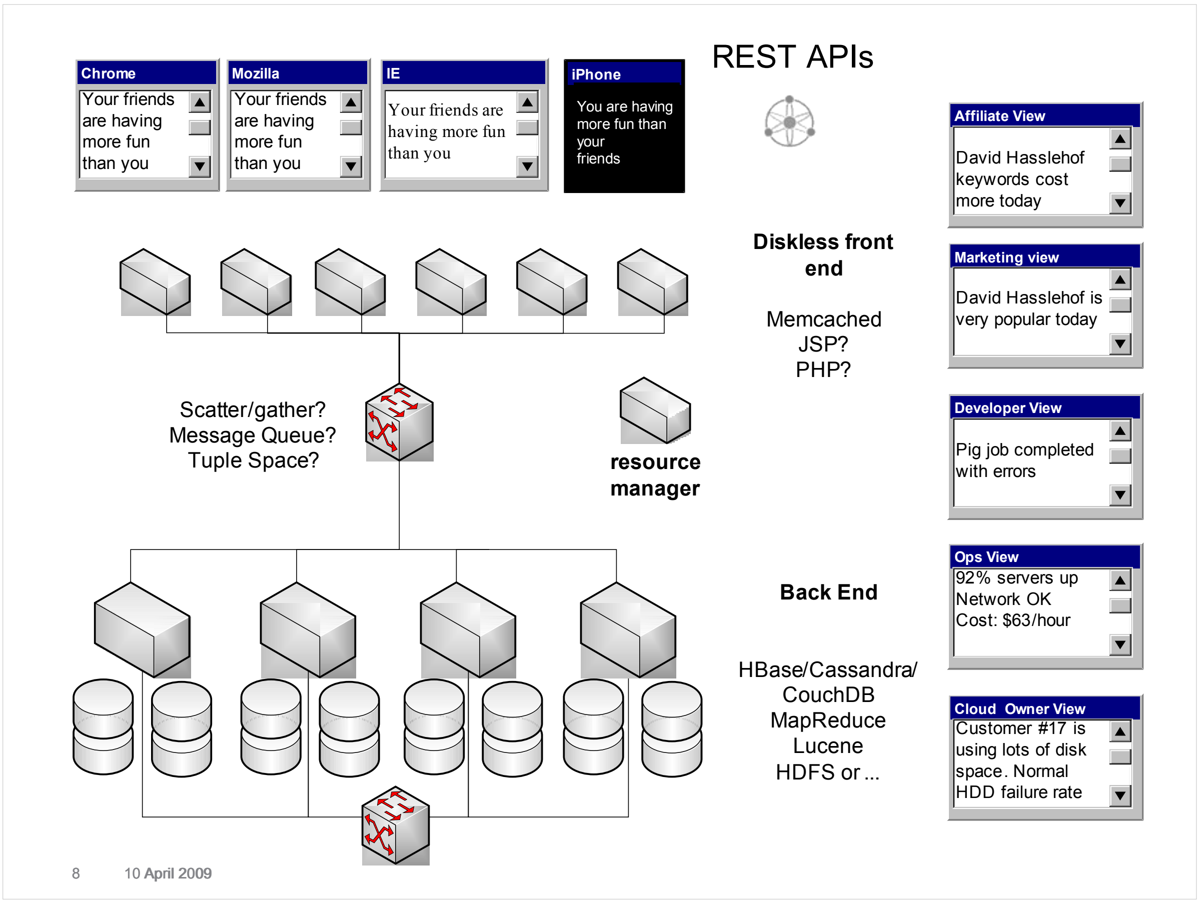
Things must change!

- 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

6 10 April 2009

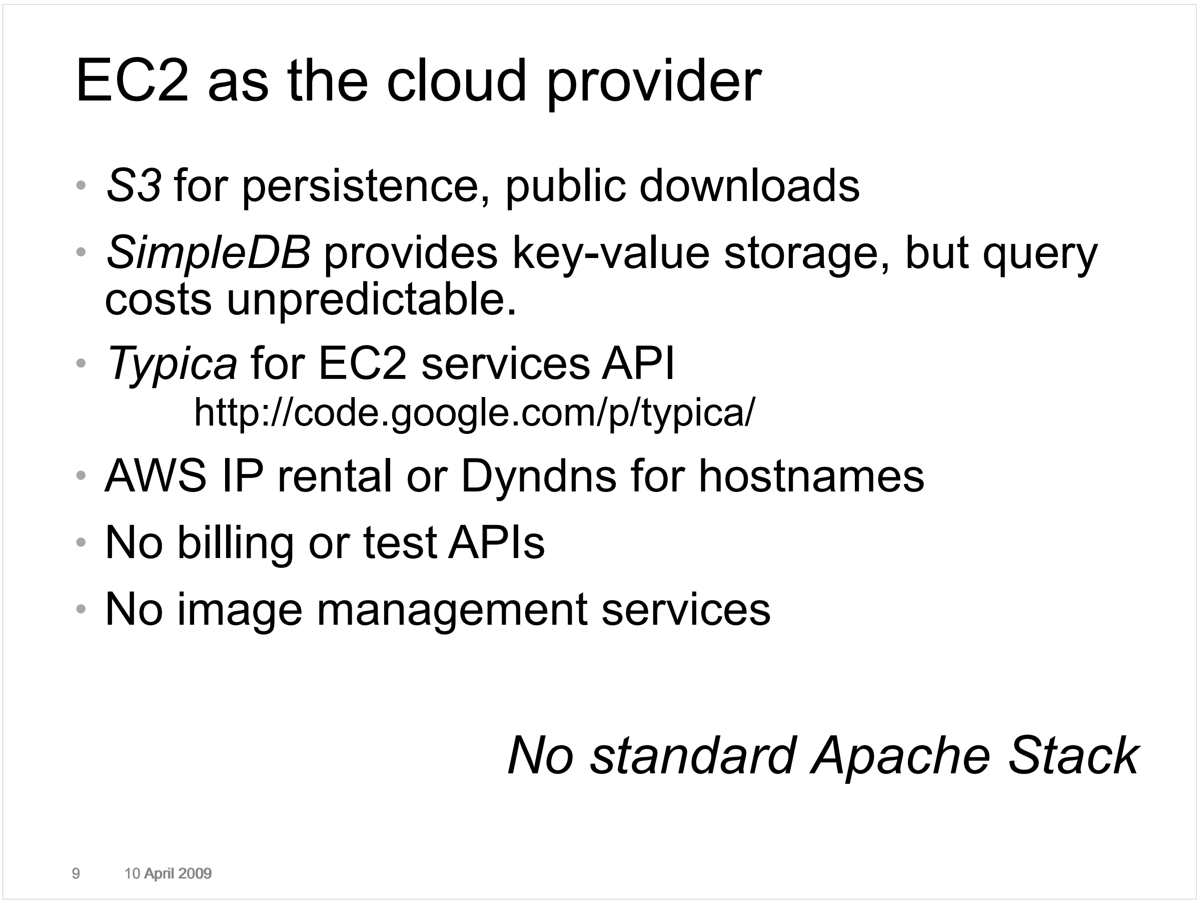
Everything needs to be agile

The agile will survive


Chrome
Your friends are having more fun than you
Mozilla
Your friends are having more fun than you
IE
Your friends are having more fun than you
iPhone
You are having more fun than your friends
REST APIs
Diskless front end
Memcached
JSP?
PHP?
Scatter/gather?
Message Queue?
Tuple Space?
resource manager
Back End
HBase/Cassandra/
CouchDB
MapReduce
Lucene
HDFS or ...
Affiliate View
David Hasslehof keywords cost more today
Marketing view
David Hasslehof is very popular today
Developer View
Pig job completed with errors
Ops View
92% servers up
Network OK
Cost: $63/hour
Cloud Owner View
Customer #17 is using lots of disk space. Normal HDD failure rate
8
10 April 2009

This is a cloud application.

- Scale: many users -or down to nearly none.
- Collaboration: if there are live feeds of all friends/colleagues, it is hard to shard
- Business model may include affiliate applications -web based
- Management, developers, ops behind the scenes
- Pay as you go infrastructure



EC2 as the cloud provider

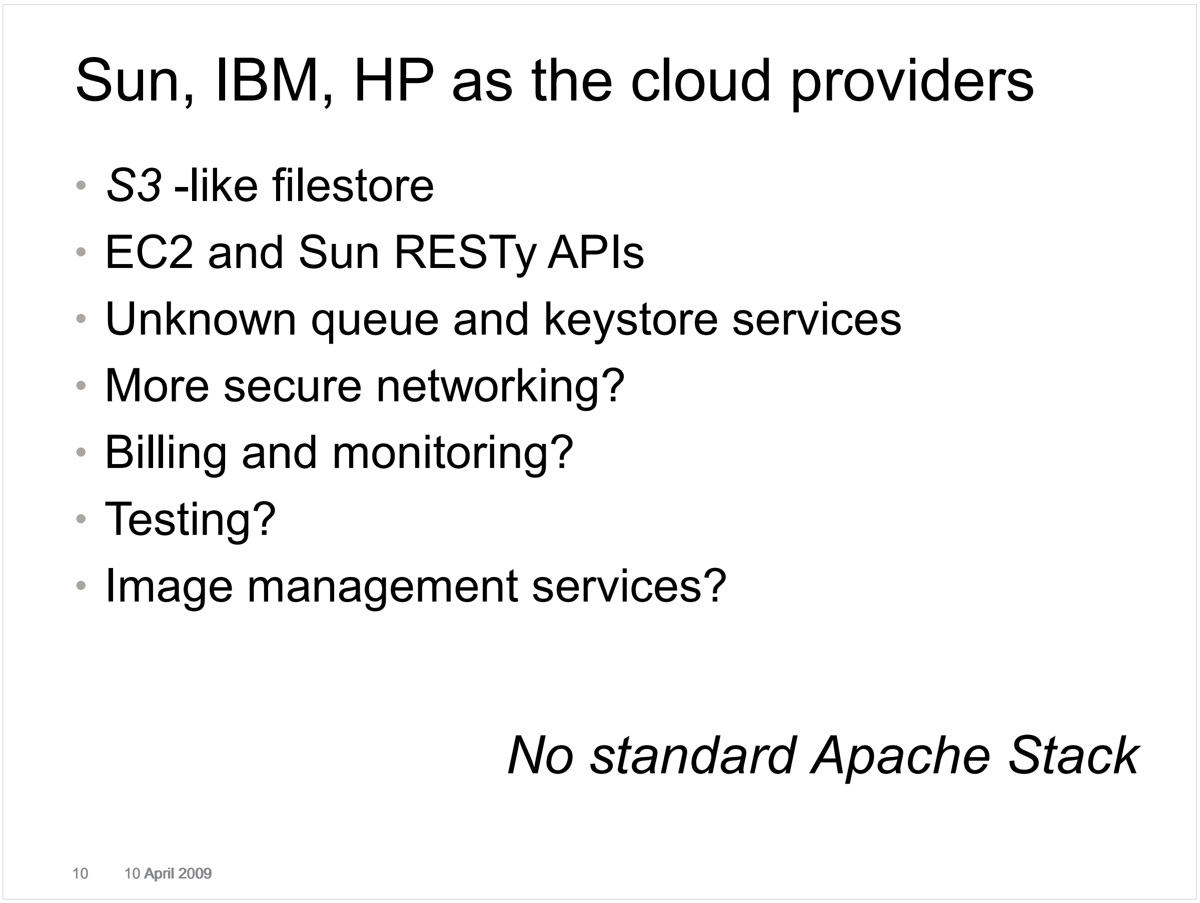
- *S3* for persistence, public downloads
- *SimpleDB* provides key-value storage, but query costs unpredictable.
- *Typica* for EC2 services API
  http://code.google.com/p/typica/
- AWS IP rental or Dyndns for hostnames
- No billing or test APIs
- No image management services

No standard Apache Stack

9 10 April 2009

Image management: provide a list of RPMs or other requirements of a machine, send a message with this list to a provider, get back the machine address to log in to. People like rightscale and similar are providing this for a fee, but AWS could do this just as easily. They are not doing it *yet*, but if your startup's business model depends on AWS not doing it, then your business model is the same as those people who provided add-ons for Windows 3.1. *Provided*. The only ones left now provide extra security for the OS, because MS now do nearly everything else themselves.

Typica is a good Java front end API to this service

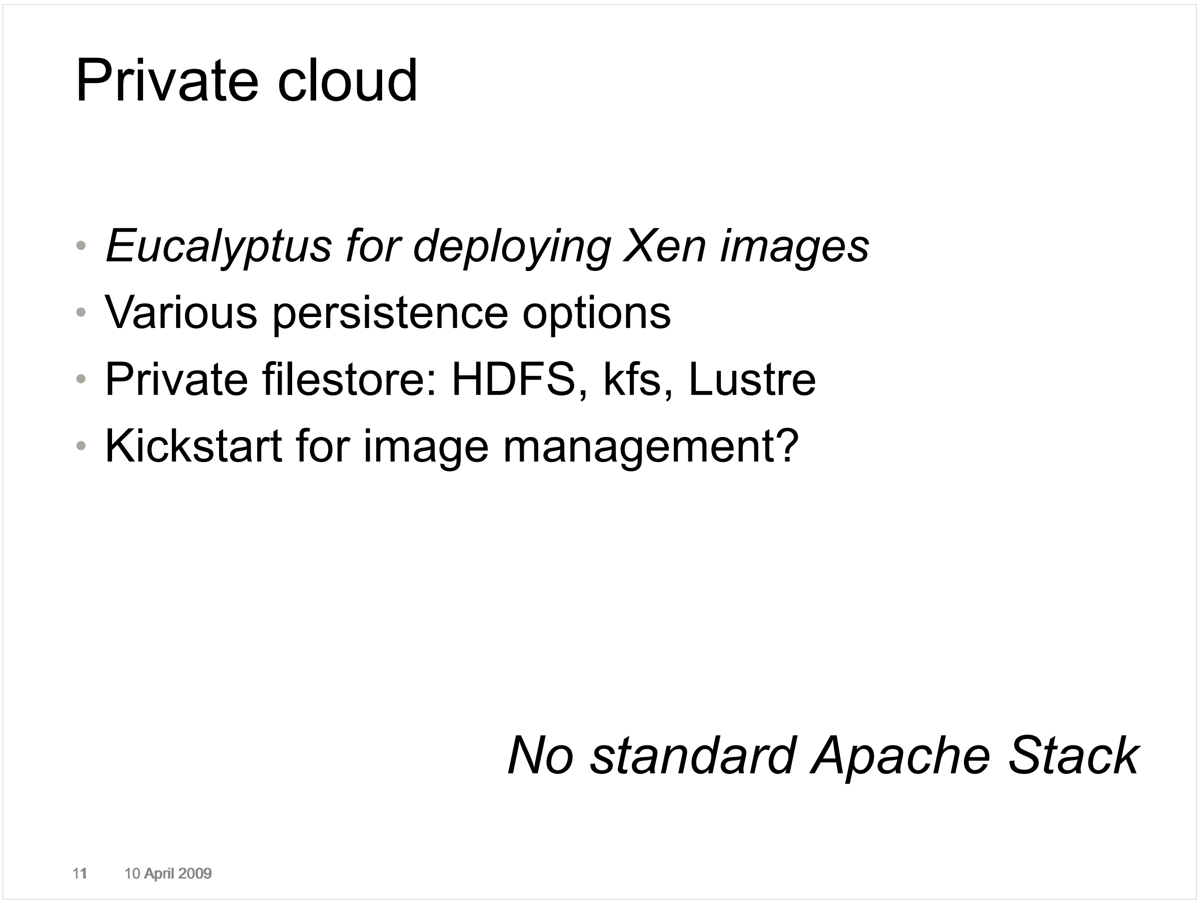


Sun, IBM, HP as the cloud providers

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

No standard Apache Stack

Who else can do this? Vendors with capital can afford to roll out infrastructure. Different business model from Amazon (who invest in datacentres for the xmas peak), they are all driven by a need to sell hardware into a world where hardware goes into datacentres.



Private cloud

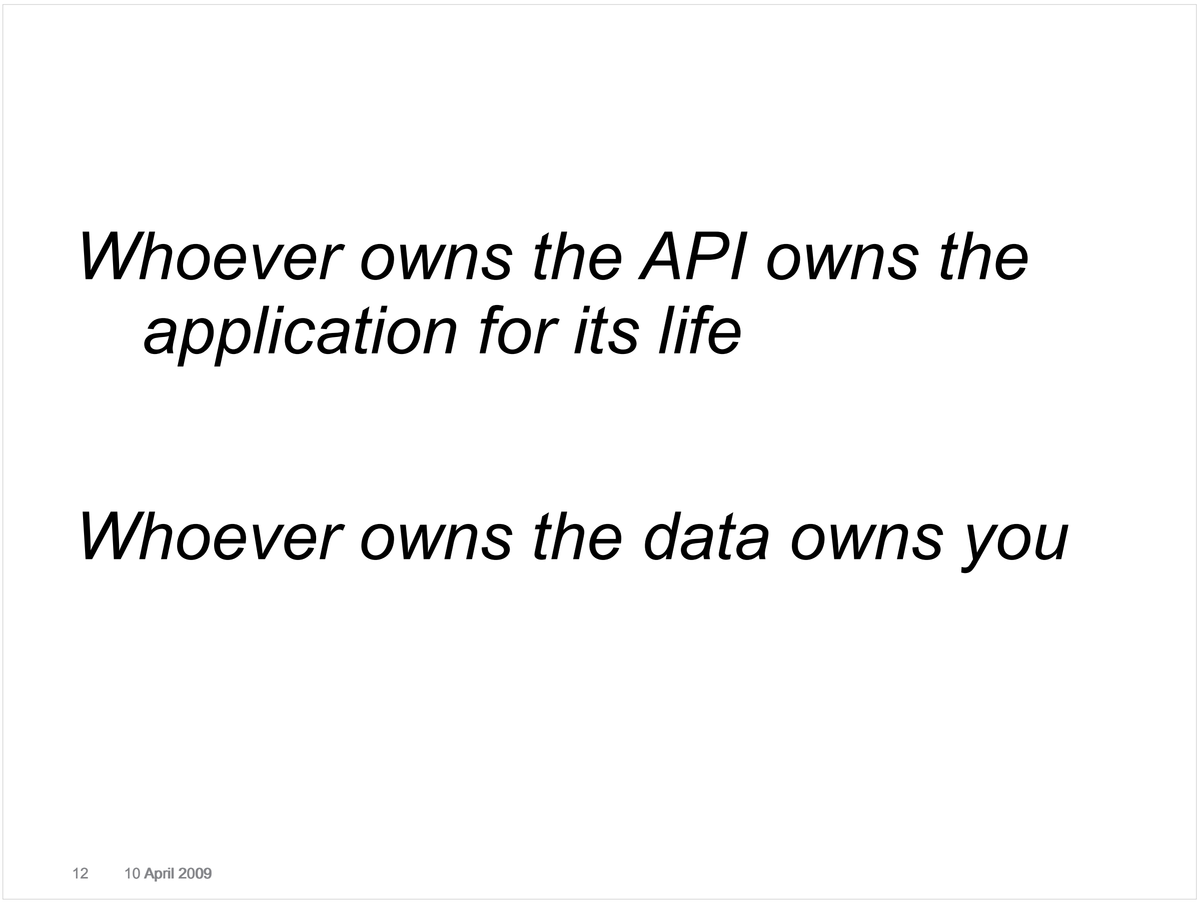
- *Eucalyptus for deploying Xen images*
- Various persistence options
- Private filestore: HDFS, kfs, Lustre
- Kickstart for image management?

No standard Apache Stack

11 10 April 2009

Private clouds are an interesting idea. You can do it today with VMWare, but it uses a different machine API, and is fairly biased towards humans and GUIS.

Eucalyptus is the OSS tool for managing a few thousand servers. "Small" datacentres -but enough for many organisations. The API gives you dynamic machines, but you are left with all the other details.

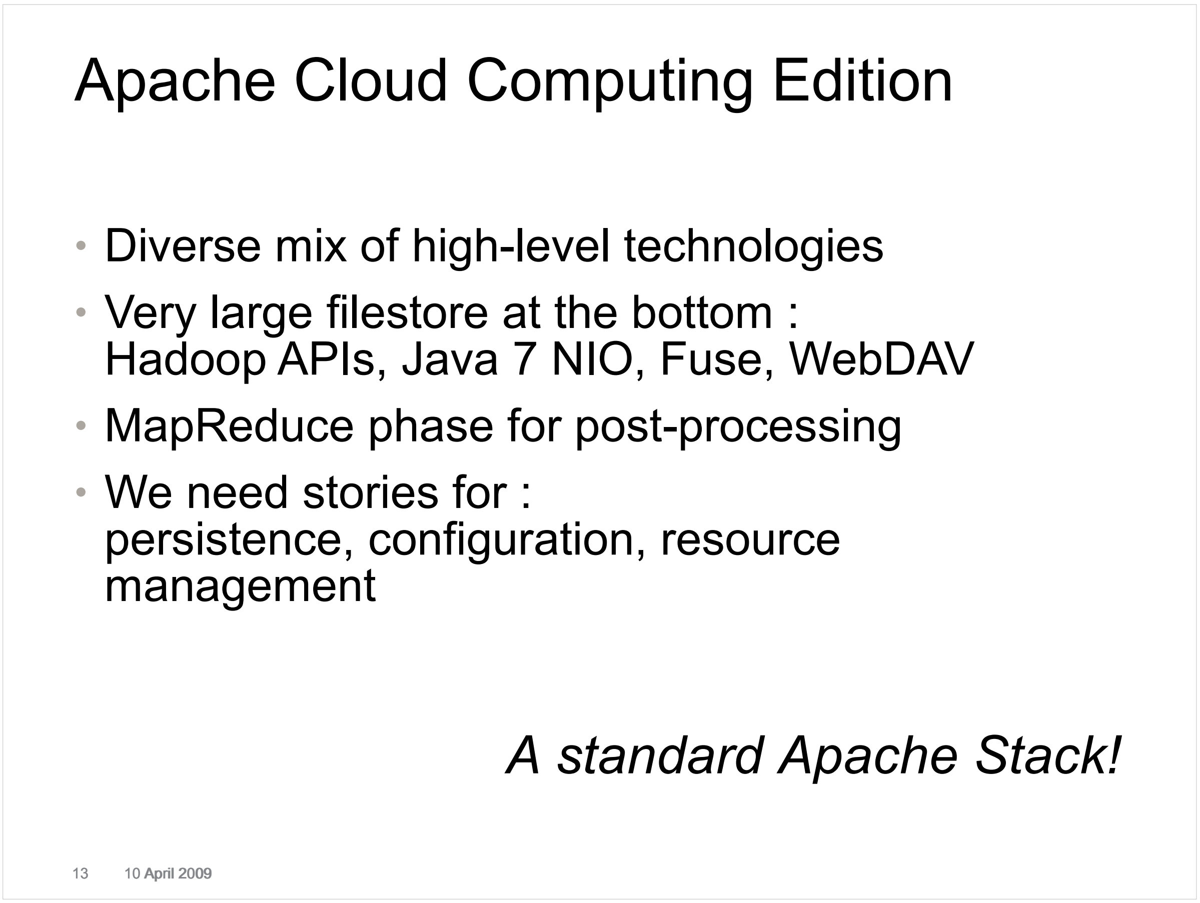


Whoever owns the API owns the application for its life

Whoever owns the data owns you

12 10 April 2009

Two observations based on the previous years of the PC business. Some people may think of MS and Oracle, but in fact IBM probably invented both of these first, those two companies just executed it better.



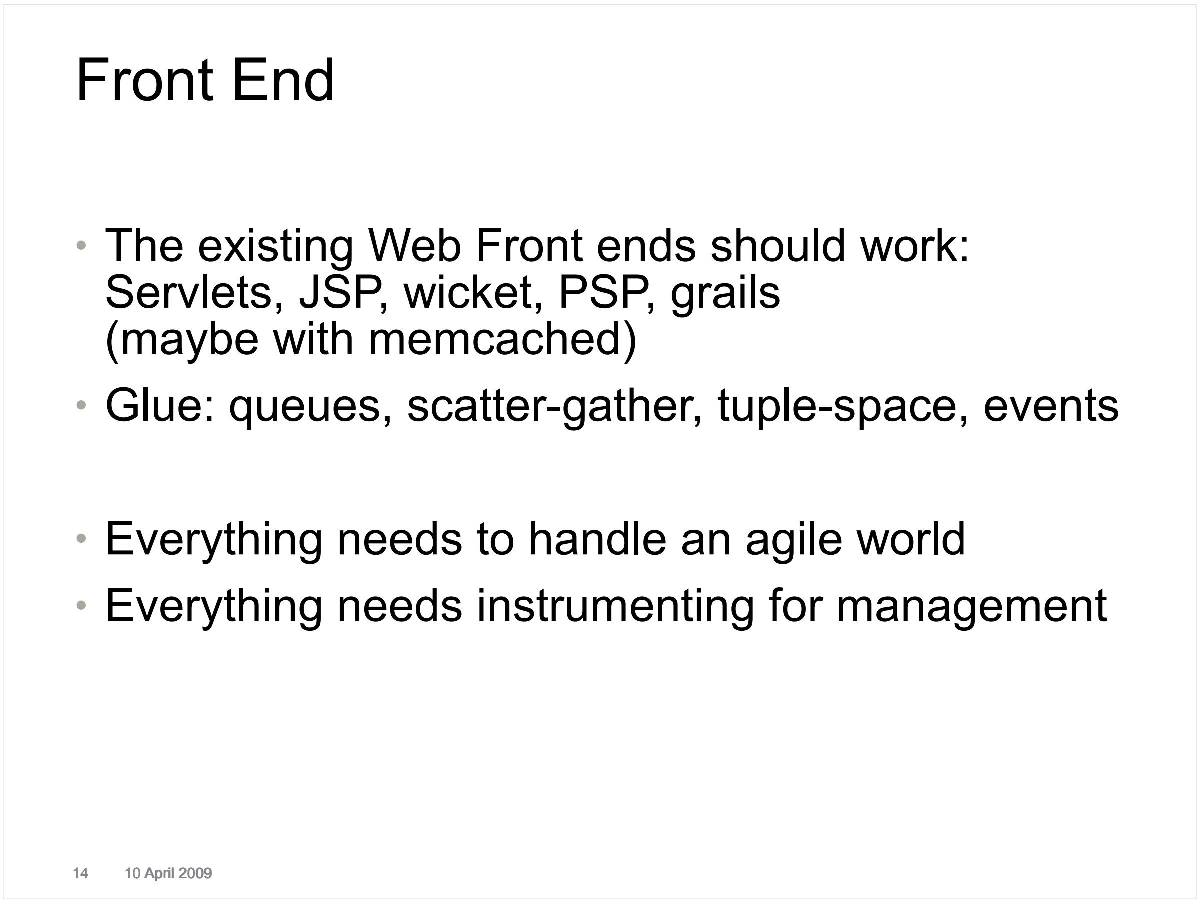
Apache Cloud Computing Edition

- 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!

13 10 April 2009

Here's a different idea. How about Apache becoming the Apache for the cloud, with our own stack, an evolution of what we have today in terms of Apache HTTPD, the Apache Java ecosystem, and what we are doing in other parts of the community?



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

14 10 April 2009

The good news: the front end still works, mostly. Where things get into trouble is if they cache changing hostnames, or contain other assumptions about where data lives on other machines, hard-coded JDBC paths, etc.

Better Glue

4. Queues. AWS provides something built in; the competitors need them too. Again, standard APIs are nice here.
5. Scatter-Gather. Doug Cutting can explain what this is. It probably work bests on networks with multicast, which means not-on-EC2.
6. Tuple-spaces. People who remember JINI and Java Spaces may remember these, but they are in fact quite useful. Any machine can assert a fact into the T-Space, other machines can look for them, act on them. A nice way to loosely couple machines. We use this for some of our resource management, though again multicast and assumptions about linear clocks can create fun in a virtual world.

Persistence

CouchDB relax

SimpleJPA

Hive

Amazon SimpleDB™ BETA

?

H·BASE

YAHOO!

PNUTS/Sherpa

Cassandra

HYPERTABLE

15 10 April 2009

This is a real troublespot. Because one of the goals of classic O/R mapping was 1:1 mapping of entries in a db to Java objects. You can't get that if you scale out the front to 200 machines, you have to deal with *eventual consistence*, and have more of a model of read-only views versus things you can write back to.

Options

-JBDC-like API to Hadoop: Hive

-Things built atop Hadoop DFS: HBase, Hypertable, Cassandra

-Other cloud-scale apache code: CouchDB

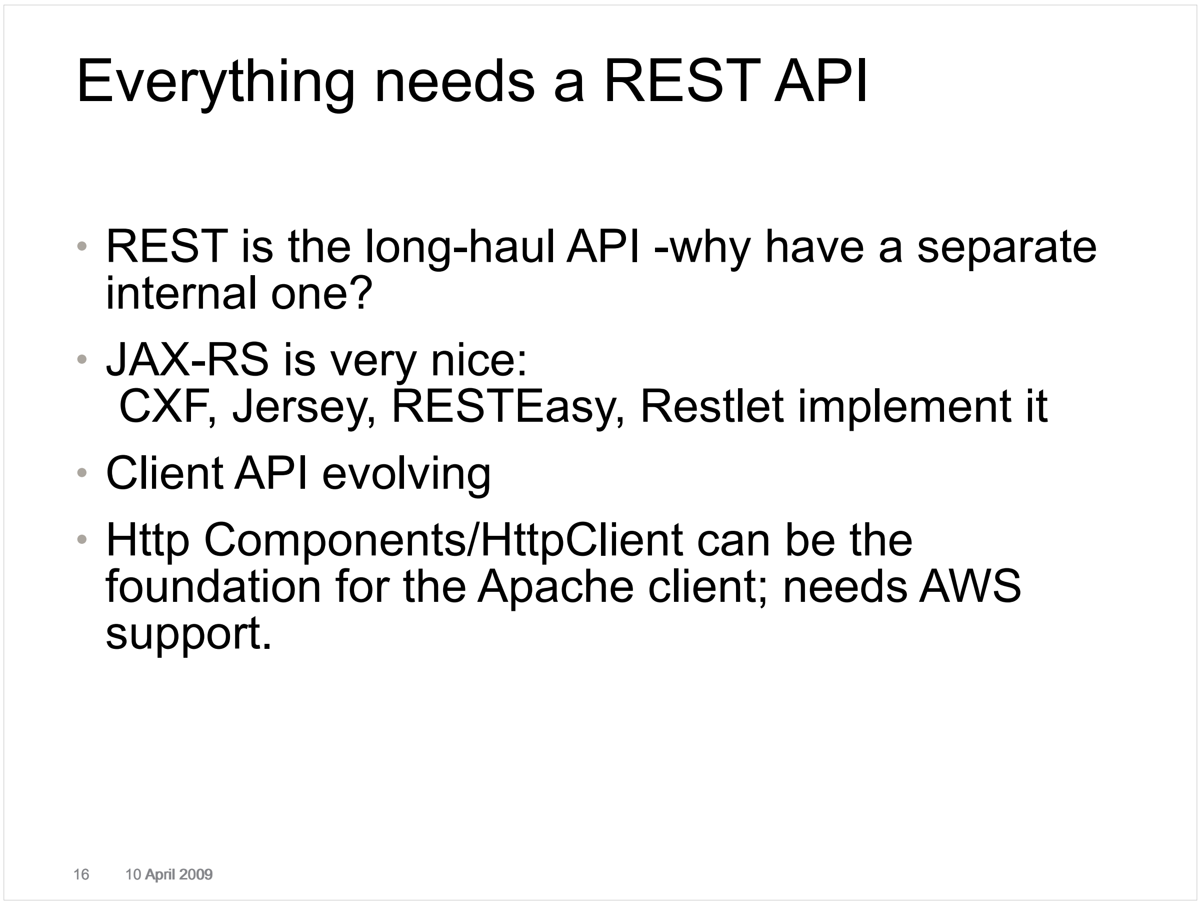
-layers to hide cloud-specific databases, such as SimpleDB and SimpleJPA

9.do we want keystore databases to be retrofitted to look like RDMS systems, or should we do something cleaner, with less locking and more eventual consistency?

10.What should the back end be?

11.What makes a good API for Java, other languages

Steve: integration with Hadoop makes database work inside MR jobs easier, but we also need fast read-access (=fast DB or slow DB+memcached), and sometimes you really do need transactions.

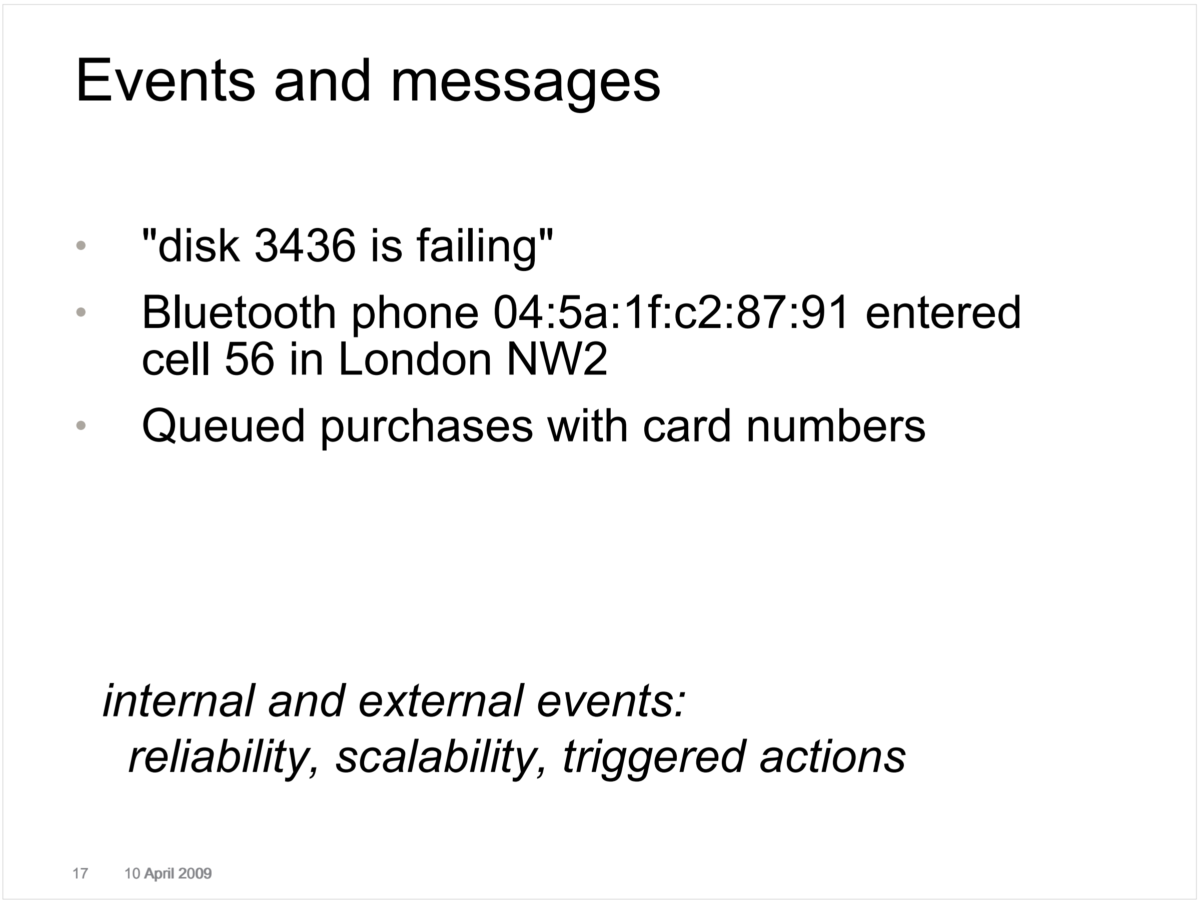


Everything needs a REST API

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

What is nice about JAX-RS is that it is an API that is nice to use (unlike, say, JAX-WS), and there are multiple implementations. I can and should be the foundation for all RESTy service endpoints in Apache unless you are tightly coupled to existing applications

Restlet is worth a play with, even though Apache folk won't like the license. It has a very clean transport-neutral model, one in which the client API is a mirror of the server one, as opposed to the java.net and Servlet APIs, that are completely different.



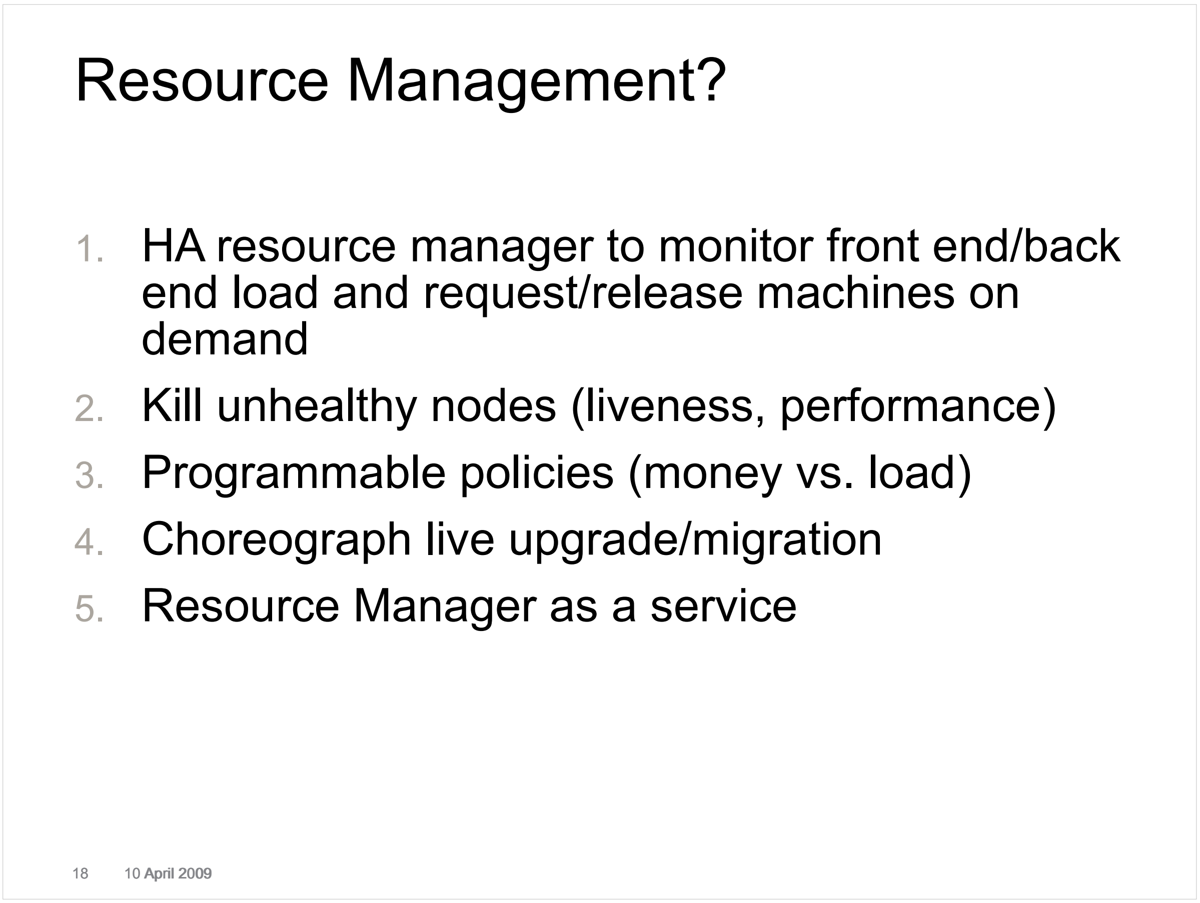
Events and messages

- "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

Eventing and messaging. Are they the same or different?

What to do with them: some you can queue, others should trigger immediate actions, others can be dealt with later, just add more data to the filestore for mining. That phone, for example: is it a triggered action, or is it something important.



Resource Management?

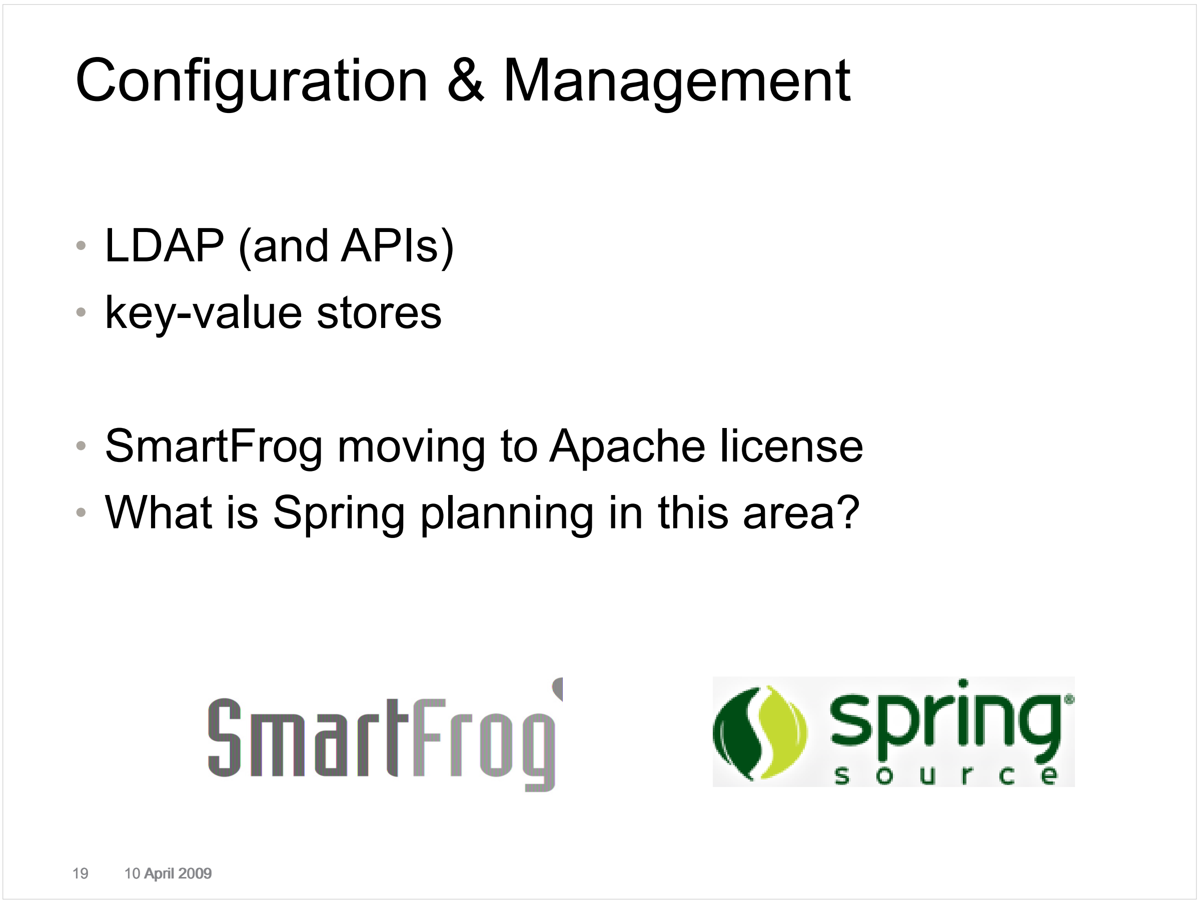
1. HA resource manager to monitor front end/back end load and request/release machines on demand
2. Kill unhealthy nodes (liveness, performance)
3. Programmable policies (money vs. load)
4. Choreograph live upgrade/migration
5. Resource Manager as a service

Resource Management is the problem of allocating enough machines for demand and your budget, but not too many. You need to create machines when load is high (within constraints), free them when low, but taking into account rules about minimum per-hour cost of machines; time to instantiate.

Also need to be able to shut down webapps in a way that they can handle. If they are crash-only this is easy, but if not, then the RM needs to shut them down gracefully.

Health also needs monitoring. In a VM-world, killing and reinstantiating is powerful -it avoids problems with overloaded racks, it avoids problems with one specific machine in the rack. But: IPAddresses can change, restart costs higher. Best to have the RM manage this gracefully with a restart followed by a full VM-termination.

The RM does not need to be something every single user of a cloud needs to host for them selves



Configuration & Management

- LDAP (and APIs)
- key-value stores

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

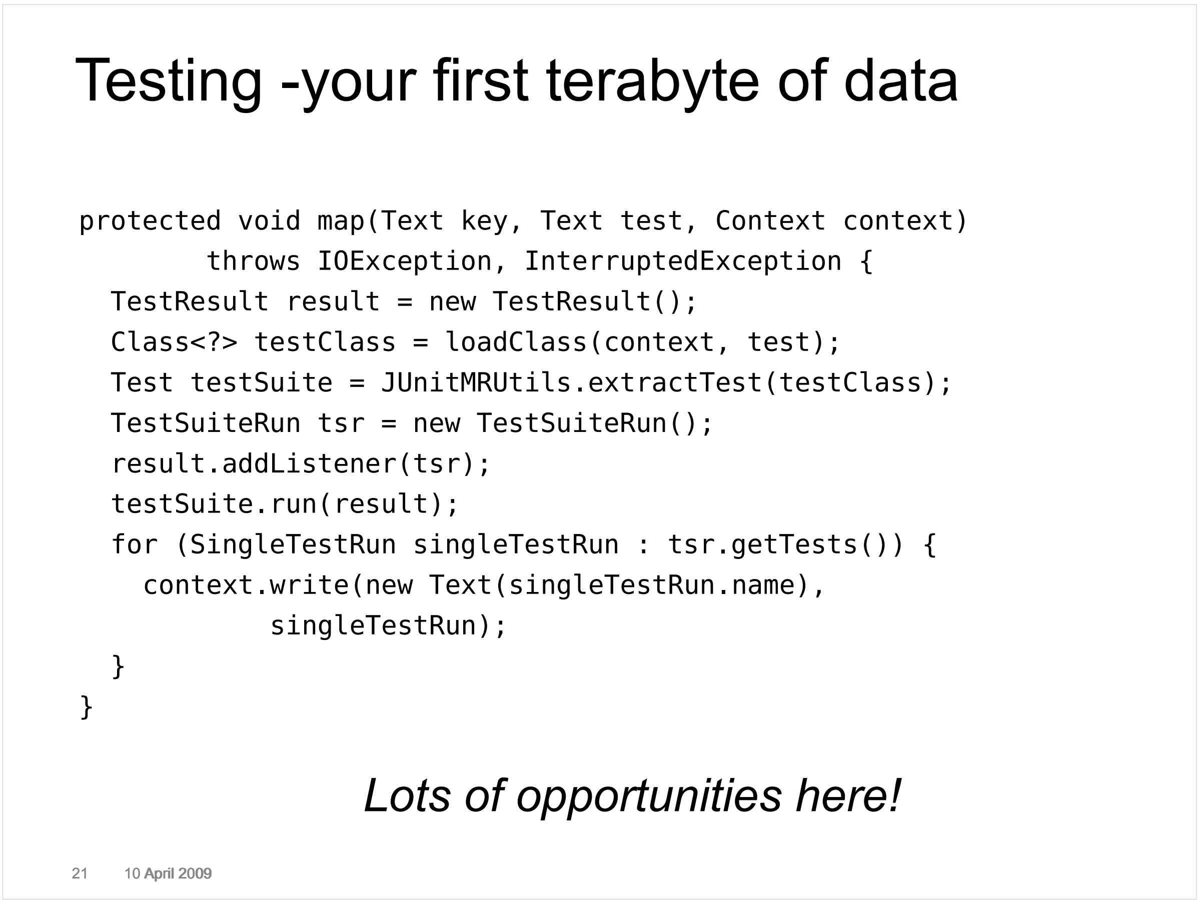
You need to worry a lot more about configuration here, because everything is agile. No more hard coded hostnames in Java pages, JDBC URLs with server hostnames in them in your jJava source

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?

I don’t want to start an Ant vs Maven debate here, as I'm stepping away from that. I will say that Eclipse is becoming the standard API, which is good in one way -one thing to target UI plugins for- bad in others. Good for uniformity, bad if you find it painful to use.

Testing is very different here. You can create machines on demand, but then your tests run up bills. A good practise here is for the developers to start their cluster when they come in in the morning; for it to be shut down when they go idle. And to use a separate credit card from production, so if they go over budget, your service doesn’t go off-line.



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!

21 10 April 2009

Testing is not merely "different", it is an opportunity to use data mining within your own process. Now you can keep all the old test run info, compare performance over time, use CM tools to explore more of the configuration space (which cluster/app options give best value for money), integrate this with the test runs.

We need:

-test runners to feed data into Hadoop filesystems

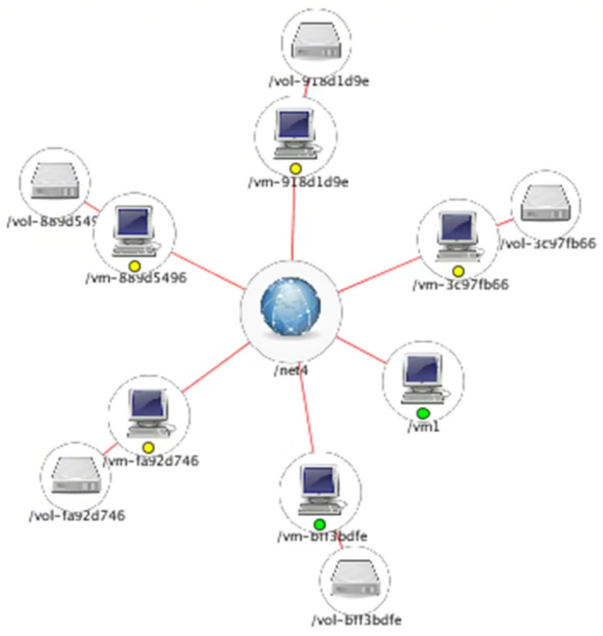
-analysis algorithms

-presentation of results.

Last.fm have been doing lots of this stuff; we've been playing with different Hadoop configs, with automated configuration generators on the plans.

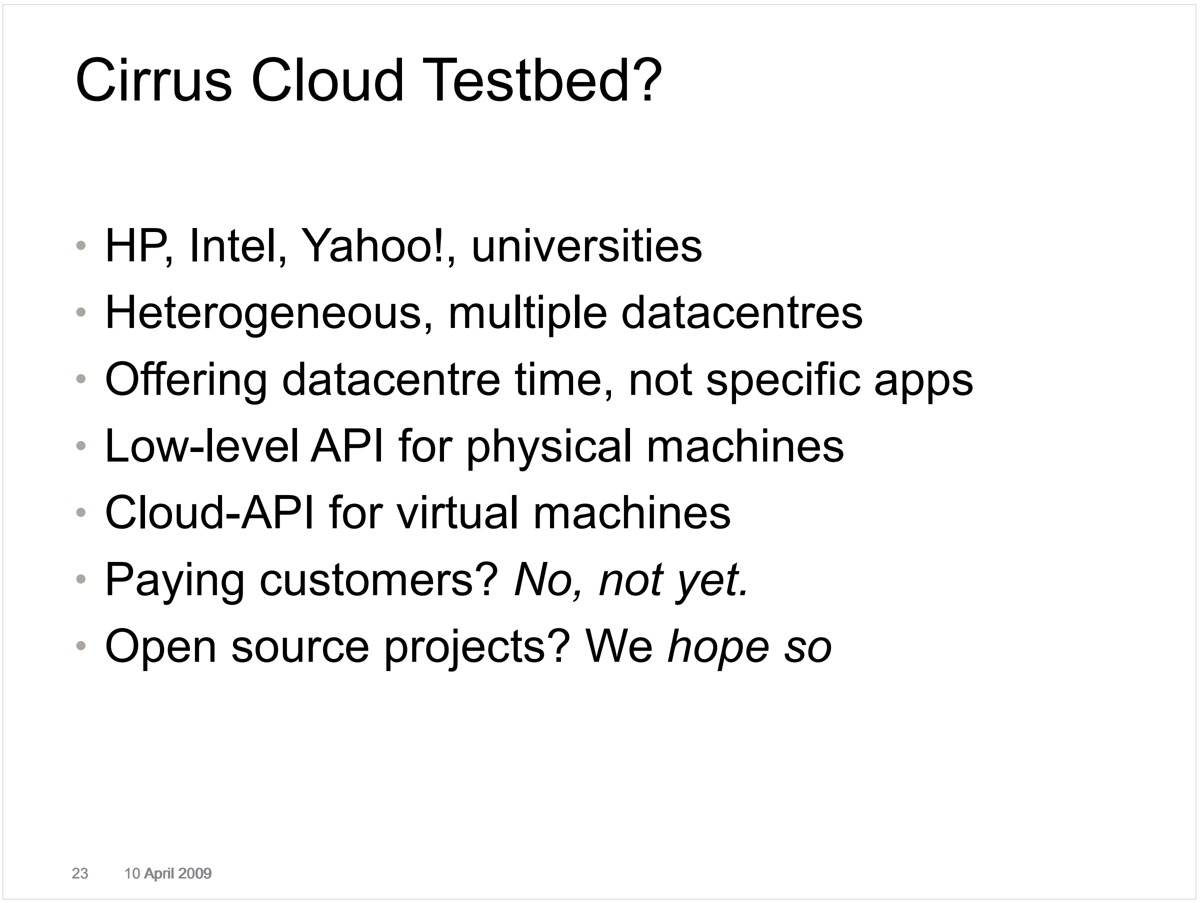
Here is something else of mine, something that runs a list of tests. takes a text file listing all test suites/classes to execute, each is a separate job. The results are pushed back as a new entry for every test in the suite.

Testing -the infrastructure can help



Pseudo-RNG driven cluster configuration

This is how HP tests some of its infrastructure-on-demand services: with a pseduo-RNG configuration generator generating more of the configuration space than humans could do themselves.


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? We *hope so*

23 10 April 2009

What's up with the HP-Intel-Yahoo! Cirrus testbed?

Cirrus is more than just Hadoop and friends in a datacentre -it is low-level physical machine access letting you install physical OS images for periods of time, and layers on top. Hadoop is one key application, but not the only one.

We're hoping that we can get OSS time in here too -because it helps ensure our code runs across multiple machines

HP: 1000 cores, 256+ boxes, some disk heavy, some RAM-heavy

What next?

Apache has the core of a Cloud Computing stack

How do we take this and:

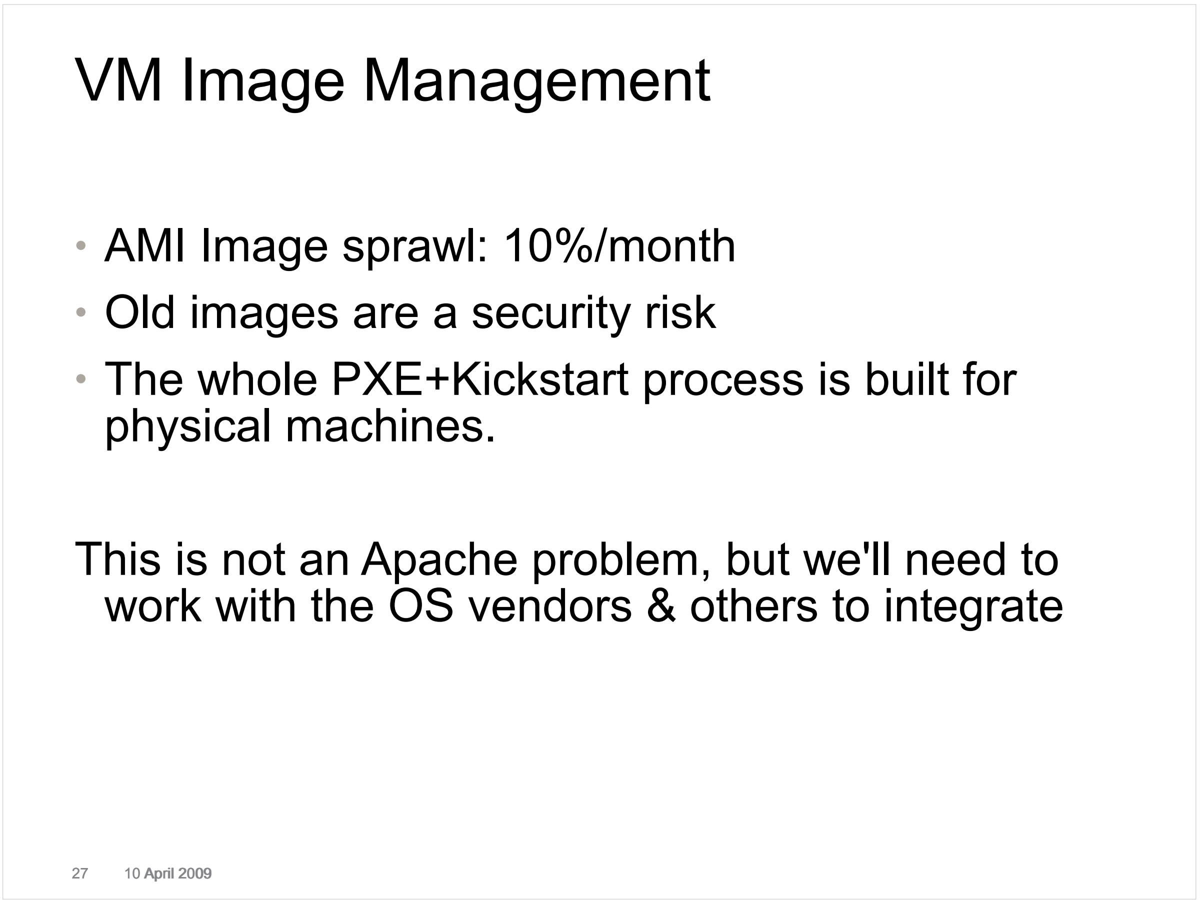
- integrate the various pieces?
- extend them where appropriate?
- provide an alternative to AppEngine and Azureus?

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 filesystems, message queues, name-value databases

Apache: let's build our own cloud platform

Let's build Apache's Cloud Stack



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

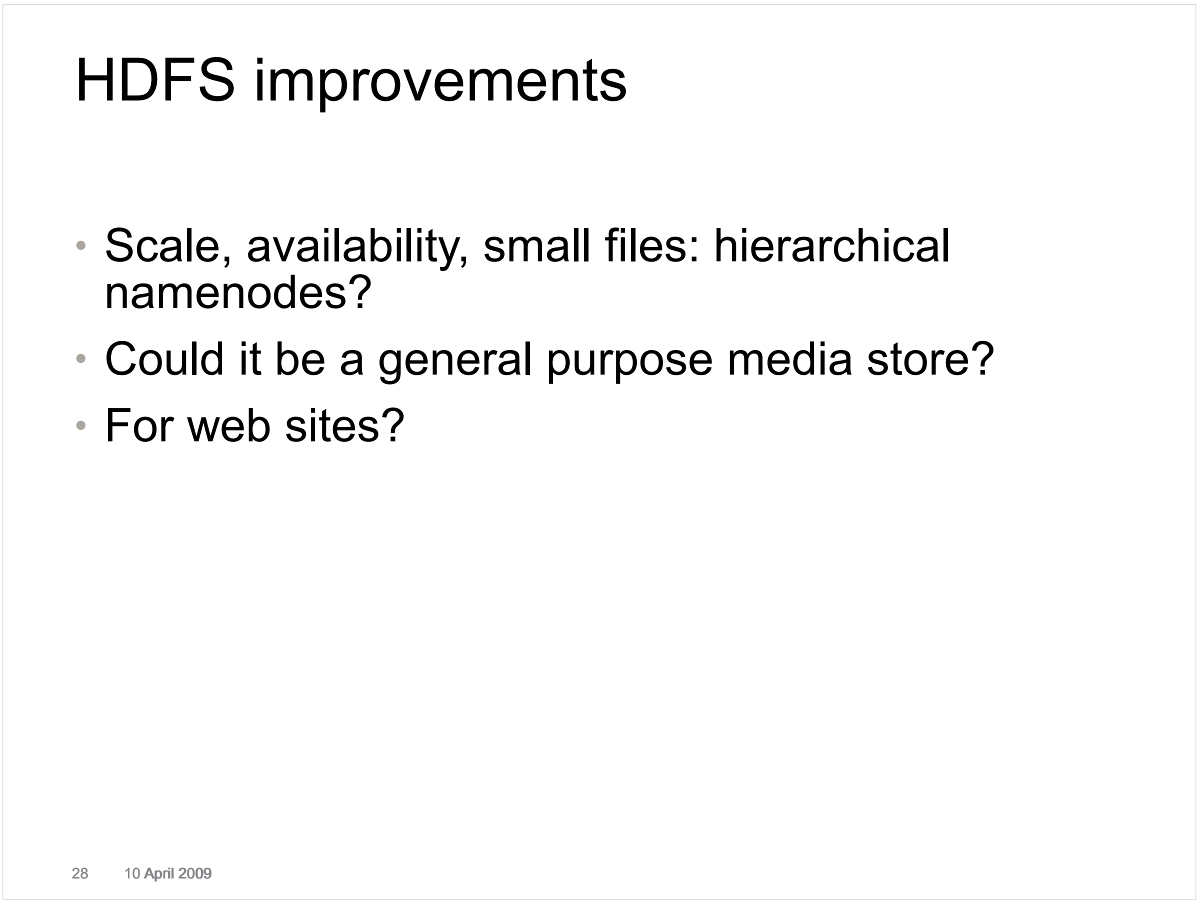
27 10 April 2009

Public EC2 images are growing at 10% month -nice earner for Amazon.

Brings on lots of maintenance costs to users, plus security risks.

For short-lived images, better to have tools to create OS images from a list of deb/RPM files and any other state changes -you needed to automate that system config anyway, didn't you?

This is not the Apache focus point -but we need to work with people doing this as their business plan is keeping our cloud services up and running.



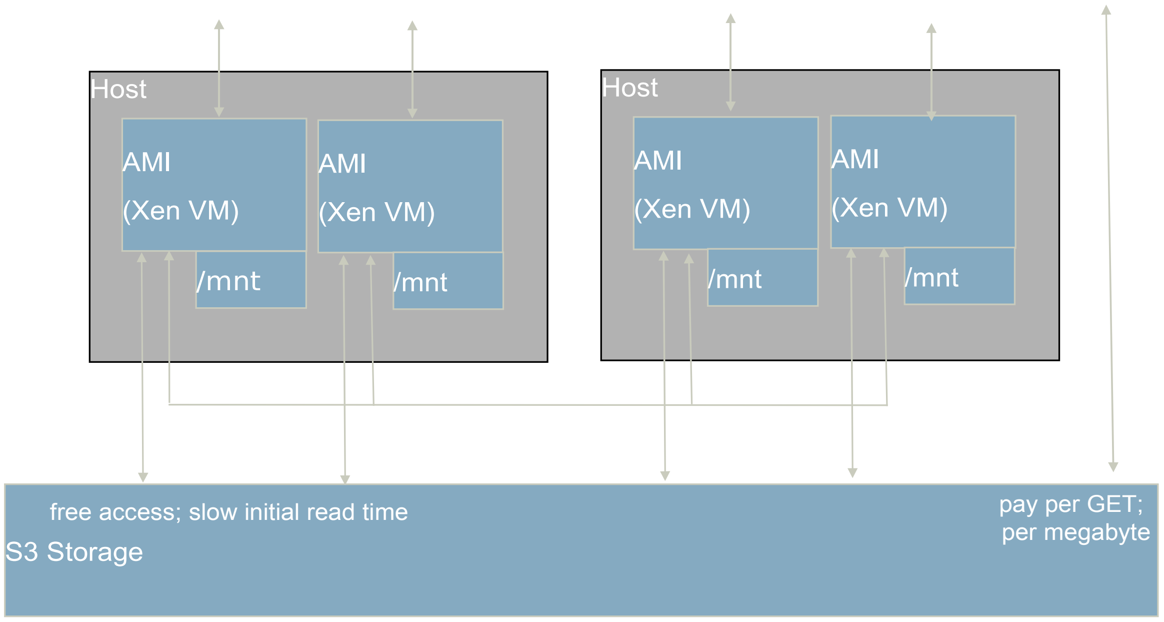
HDFS improvements

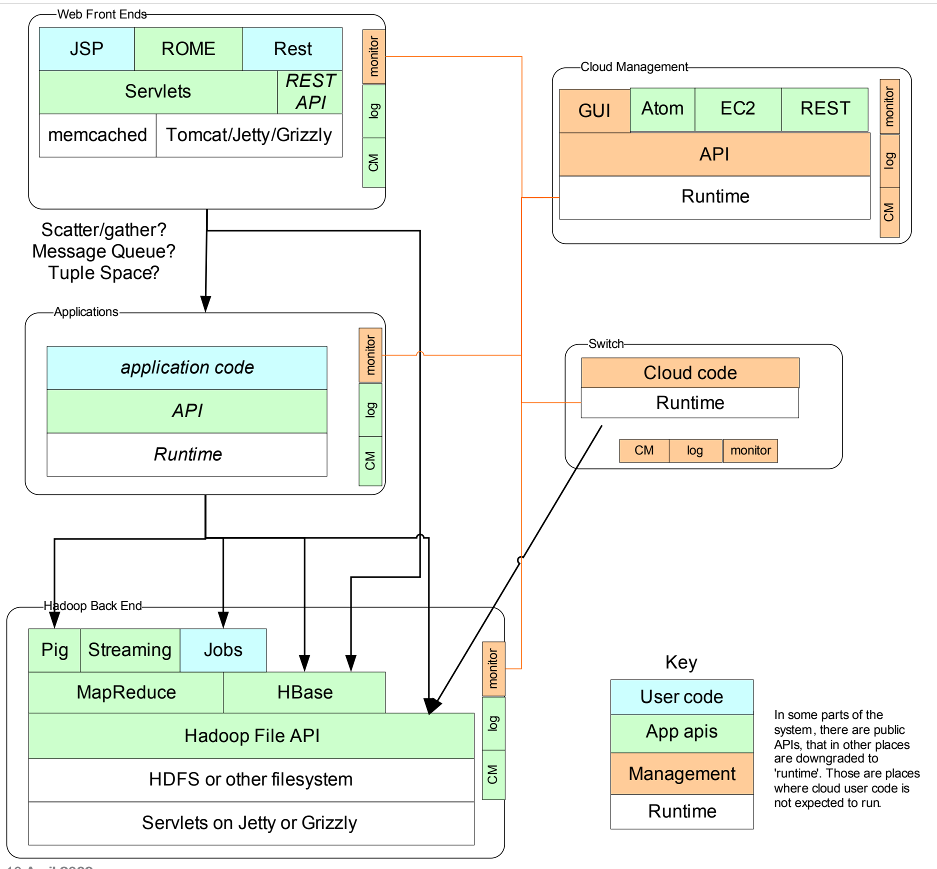
- Scale, availability, small files: hierarchical namenodes?
- Could it be a general purpose media store?
- For web sites?

28 10 April 2009

Real security heres hard. What is being done there is adequate for a network where you trust everyone.

Amazon EC2


Host
AMI
(Xen VM)
/mnt
AMI
(Xen VM)
/mnt
Host
AMI
(Xen VM)
/mnt
AMI
(Xen VM)
/mnt
free access; slow initial read time
S3 Storage
pay per GET; per megabyte



Web Front Ends
JSP
ROME
Rest
Servlets
REST API
memcached
Tomcat/Jetty/Grizzly
monitor
log
CM
Scatter/gather?
Message Queue?
Tuple Space?
Applications
application code
API
Runtime
Cloud Management
GUI
Atom
EC2
REST
API
Runtime
Switch
Cloud code
Runtime
CM
log
monitor
Hadoop Back End
Pig
Streaming
Jobs
MapReduce
HBase
Hadoop File API
HDFS or other filesystem
Servlets on Jetty or Grizzly
Key
User code
App apis
Management
Runtime
In some parts of the system, there are public APIs, that in other places are downgraded to 'runtime'. Those are places where cloud user code is not expected to run.