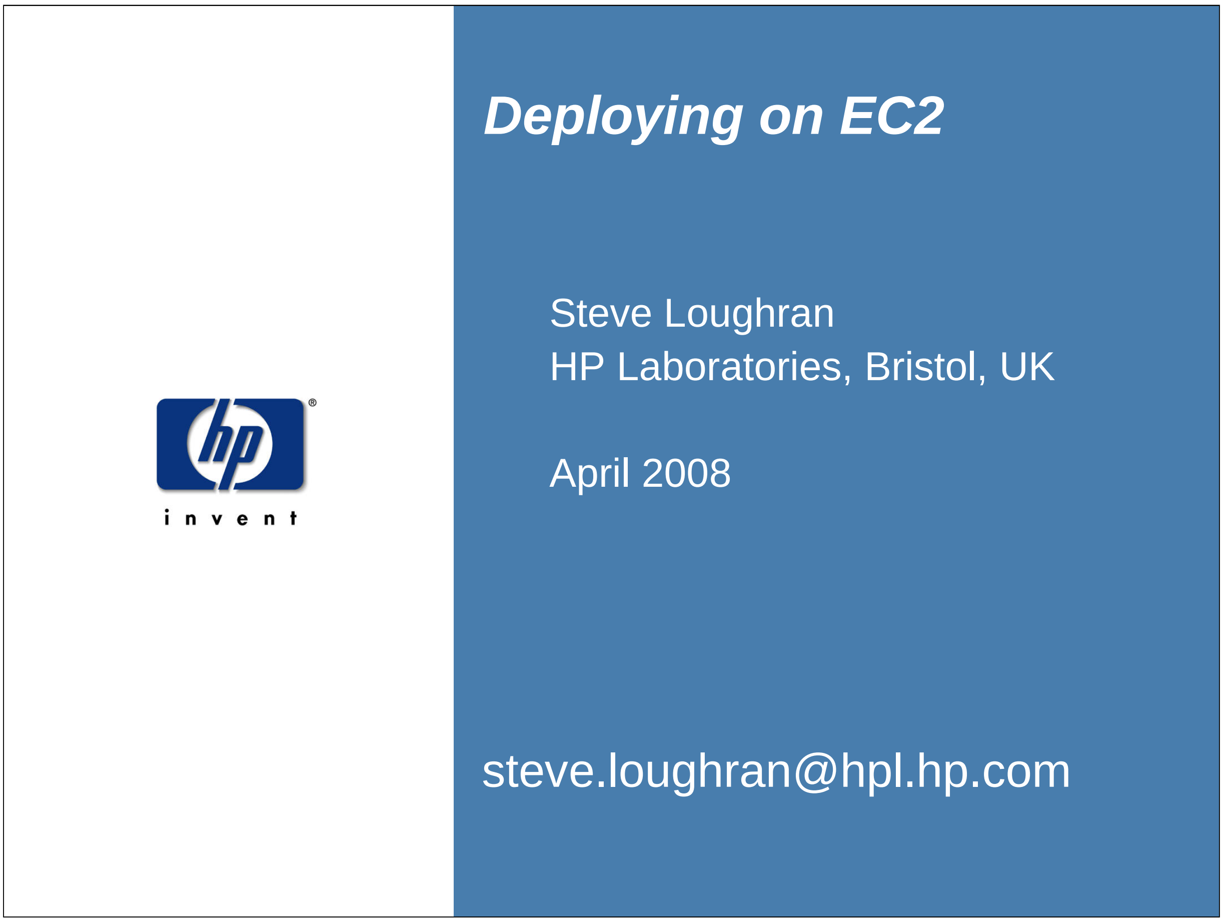


Deploying on EC2

Steve Loughran
HP Laboratories, Bristol, UK

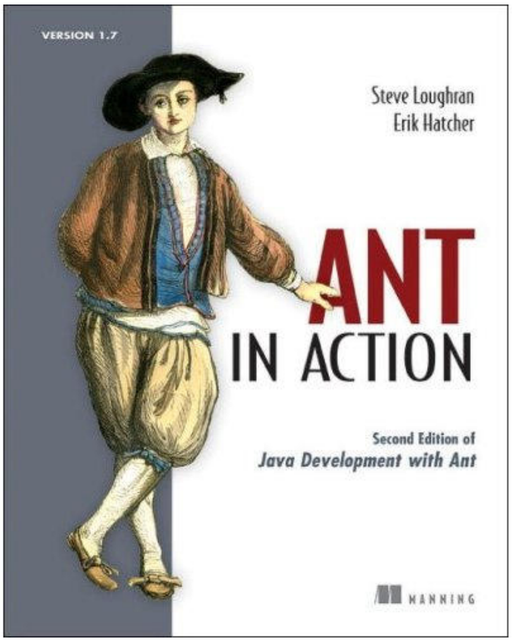
April 2008

steve.loughran@hpl.hp.com

this is a fast feather talk at apachecon 2008

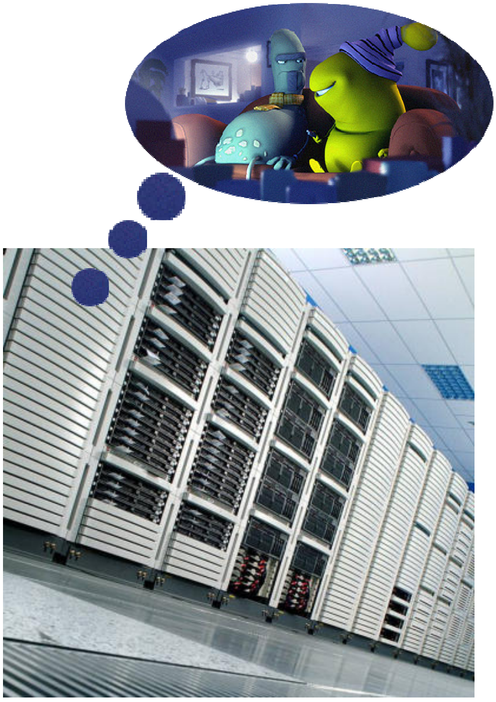
Steve Loughran

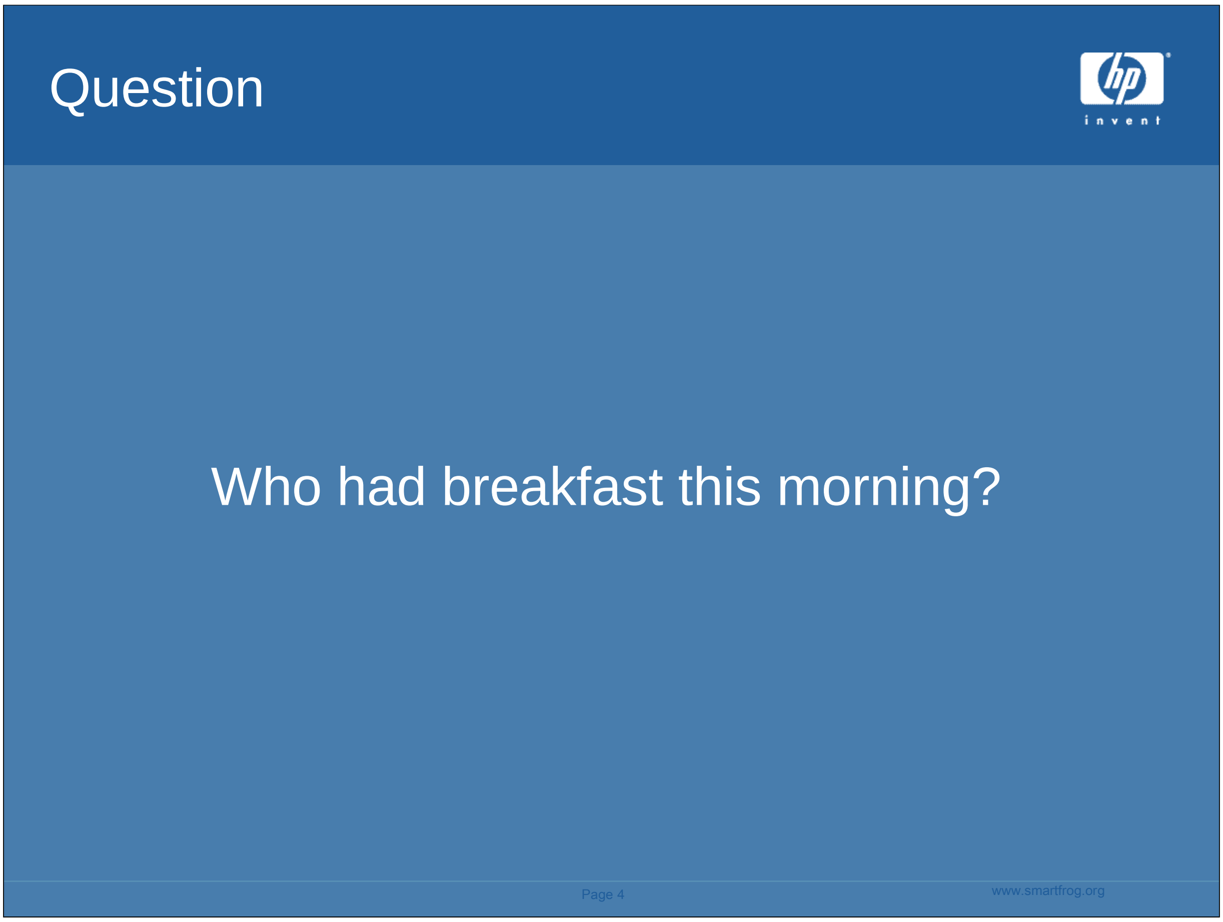
Researcher at HP Laboratories
Area of interest: Deployment
Author of *Ant in Action*



Our research - see smartfrog.org

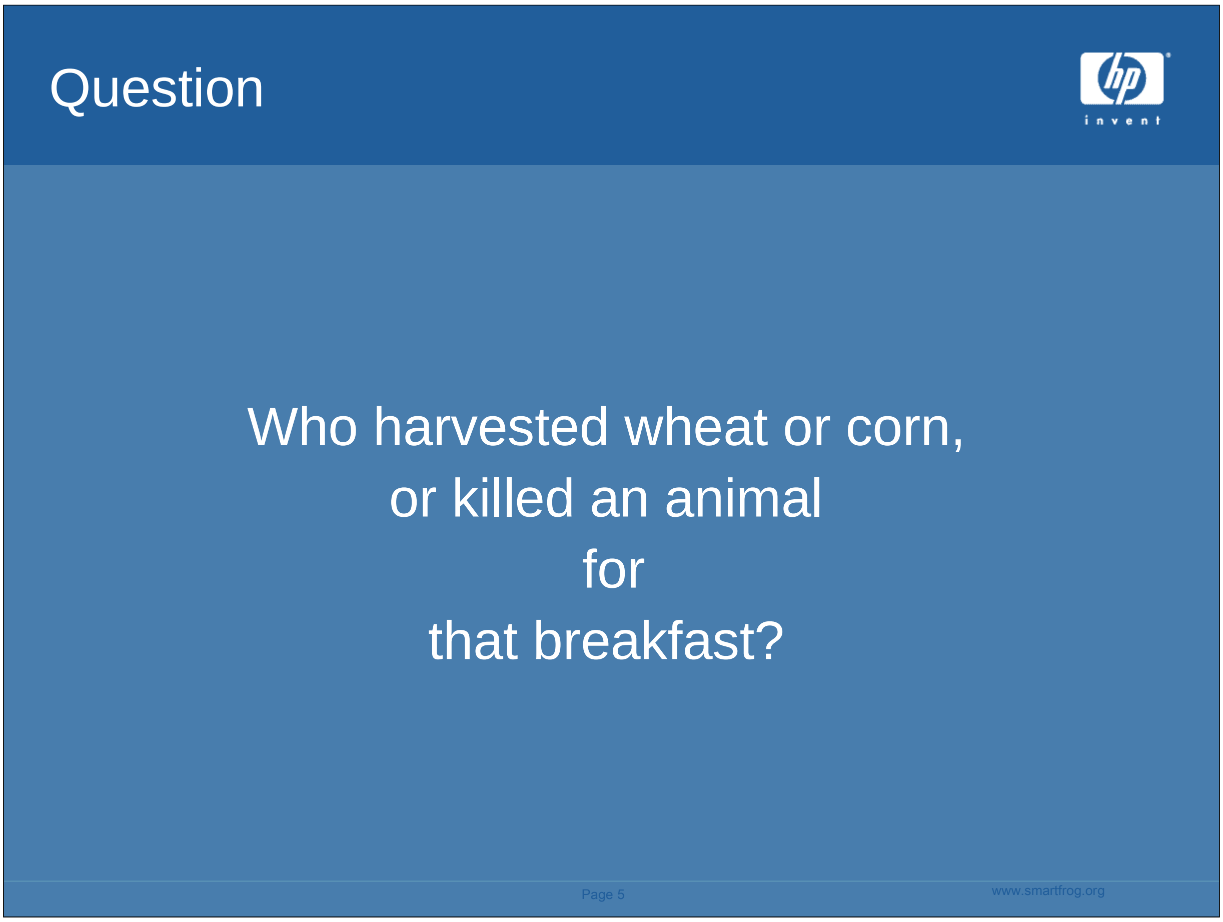
- How to host big applications across distributed resources
  - Automatically
  - Repeatably
  - Dynamically
  - Correctly
  - Securely
- How to manage them from installation to removal
- How to make dynamically allocated servers useful





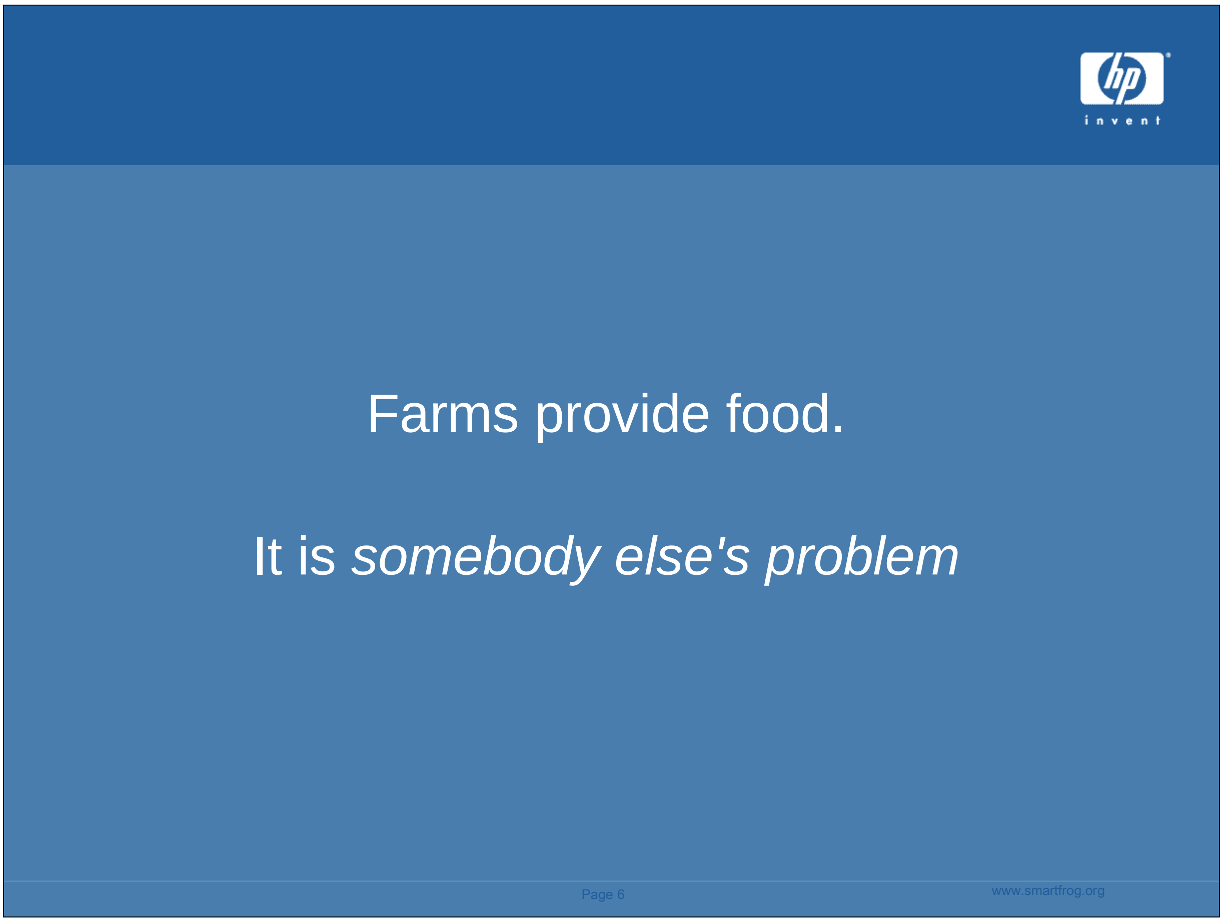
Question

Who had breakfast this morning?



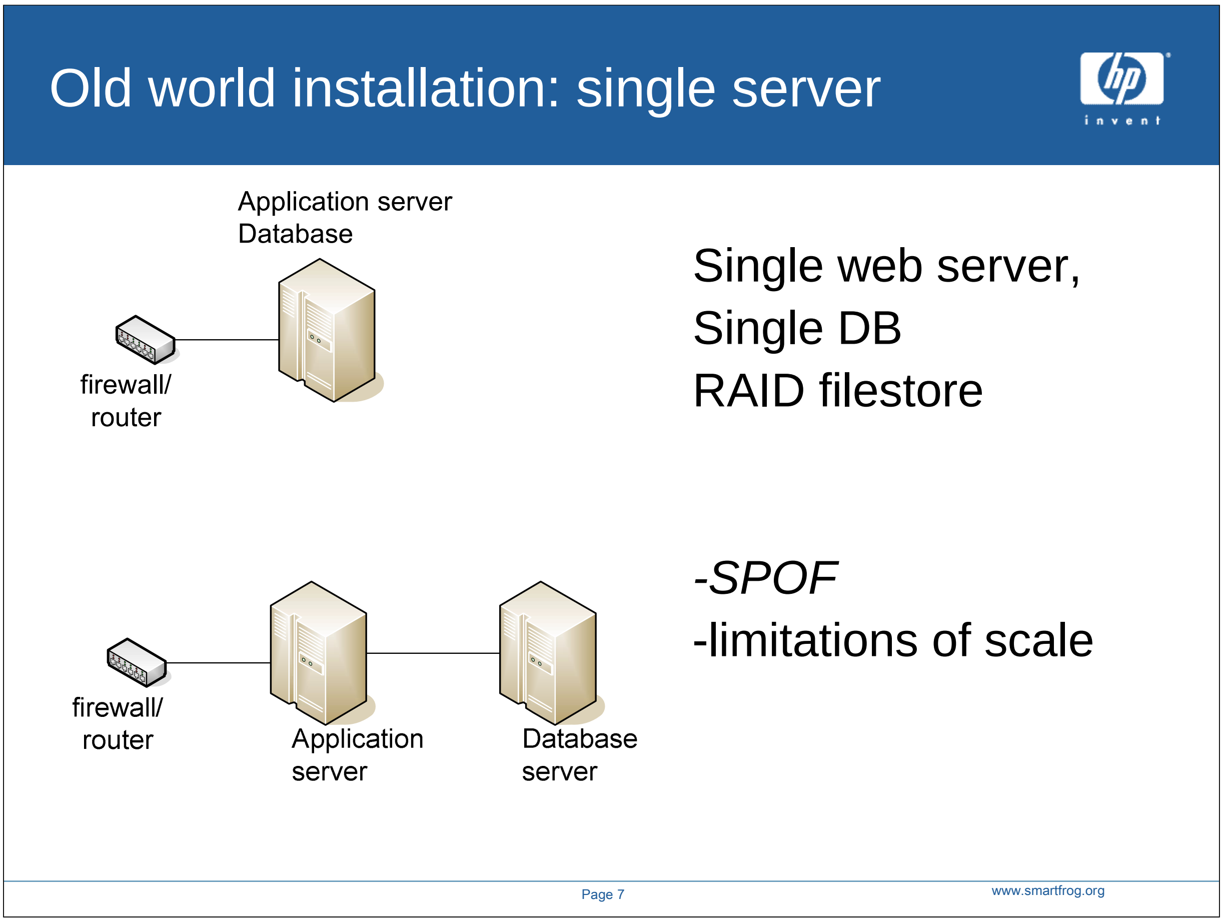
Question

Who harvested wheat or corn,
or killed an animal
for
that breakfast?



Farms provide food.

It is *somebody else's problem*



Old world installation: single server

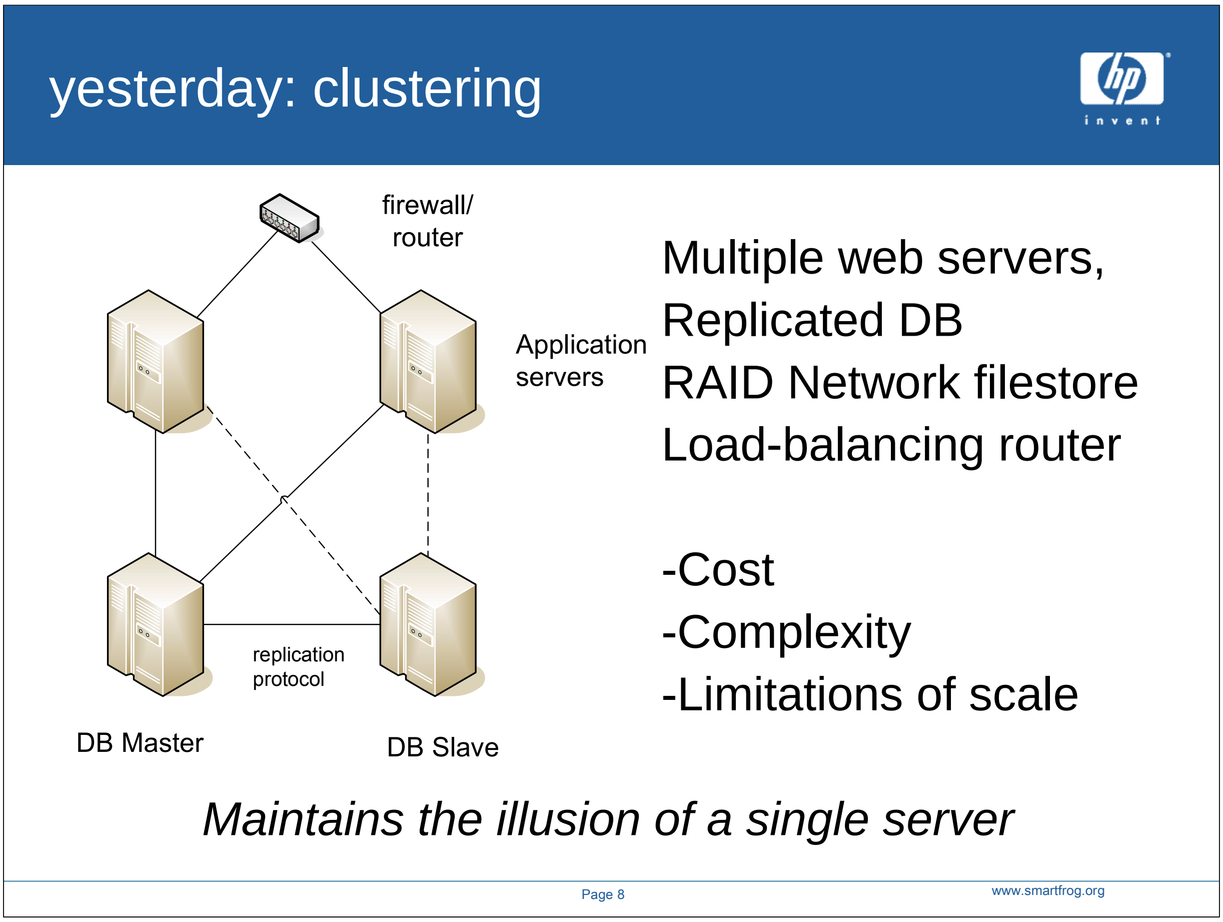
Single web server,
Single DB
RAID filestore

-SPOF
-limitations of scale

Page 7

www.smartfrog.org

- This is the base architecture for the early web apps, and a common design for small systems
- One app server, one database. Developers have them on the same box, production has them on separate boxes. It's a single point of failure



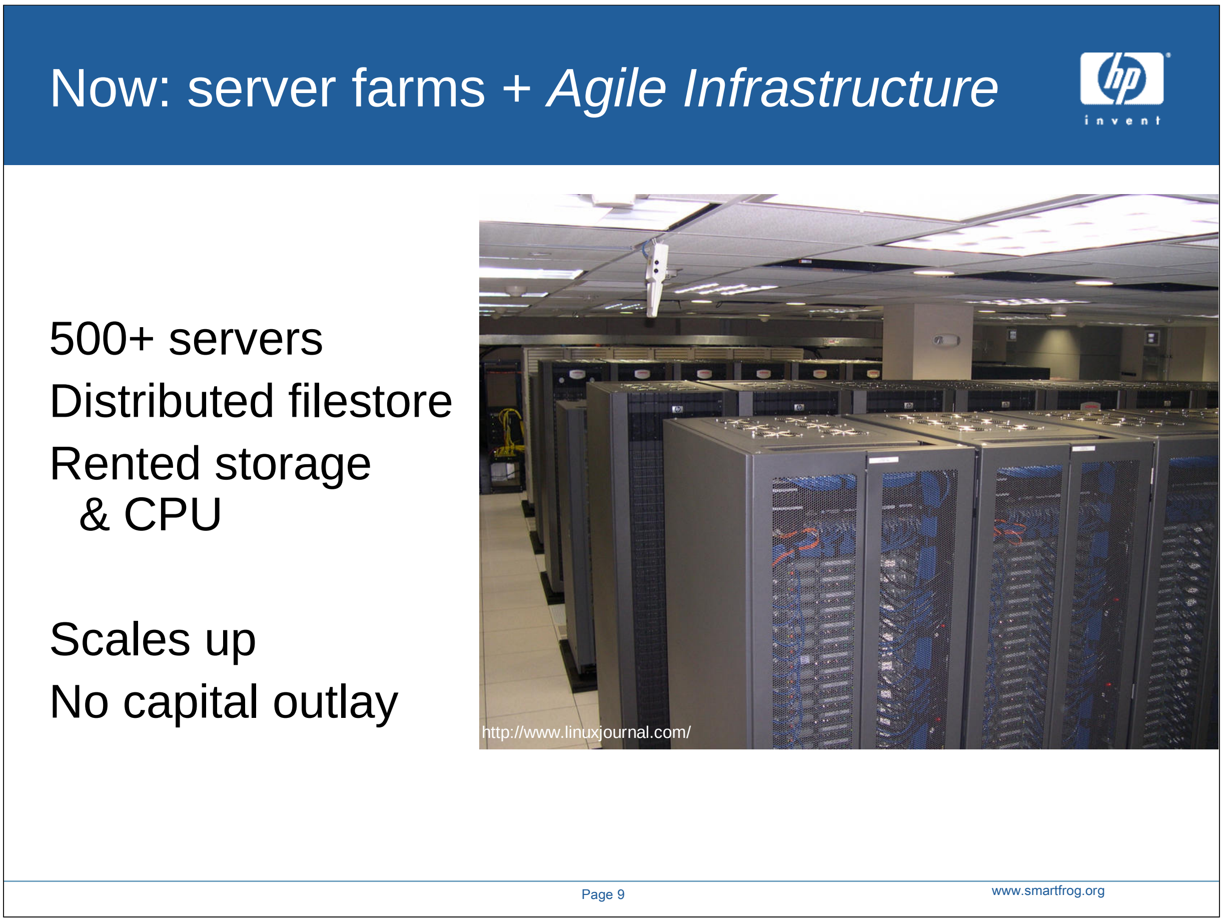
yesterday: clustering

Multiple web servers,
Replicated DB
RAID Network filestore
Load-balancing router

-Cost
-Complexity
-Limitations of scale

Maintains the illusion of a single server

- You can eliminate the SPOF by adding multiple machines; this can also address scale
- This is called *clustering*
- Here is an example app server. The router directs traffic to different app servers (only to ones that are there; not overloaded -maybe the last box that processed the same caller)
- The app servers talk to the database
- The database is redundant: failure of one computer or even one disk array can be handle. The master processes all requests, but forwards them to the slave, which stays consistent. If the master goes down, the slave takes over (tricky!)
- This architecture maintains the illusion that you have one single machine



Now: server farms + *Agile Infrastructure*

500+ servers

Distributed filestore

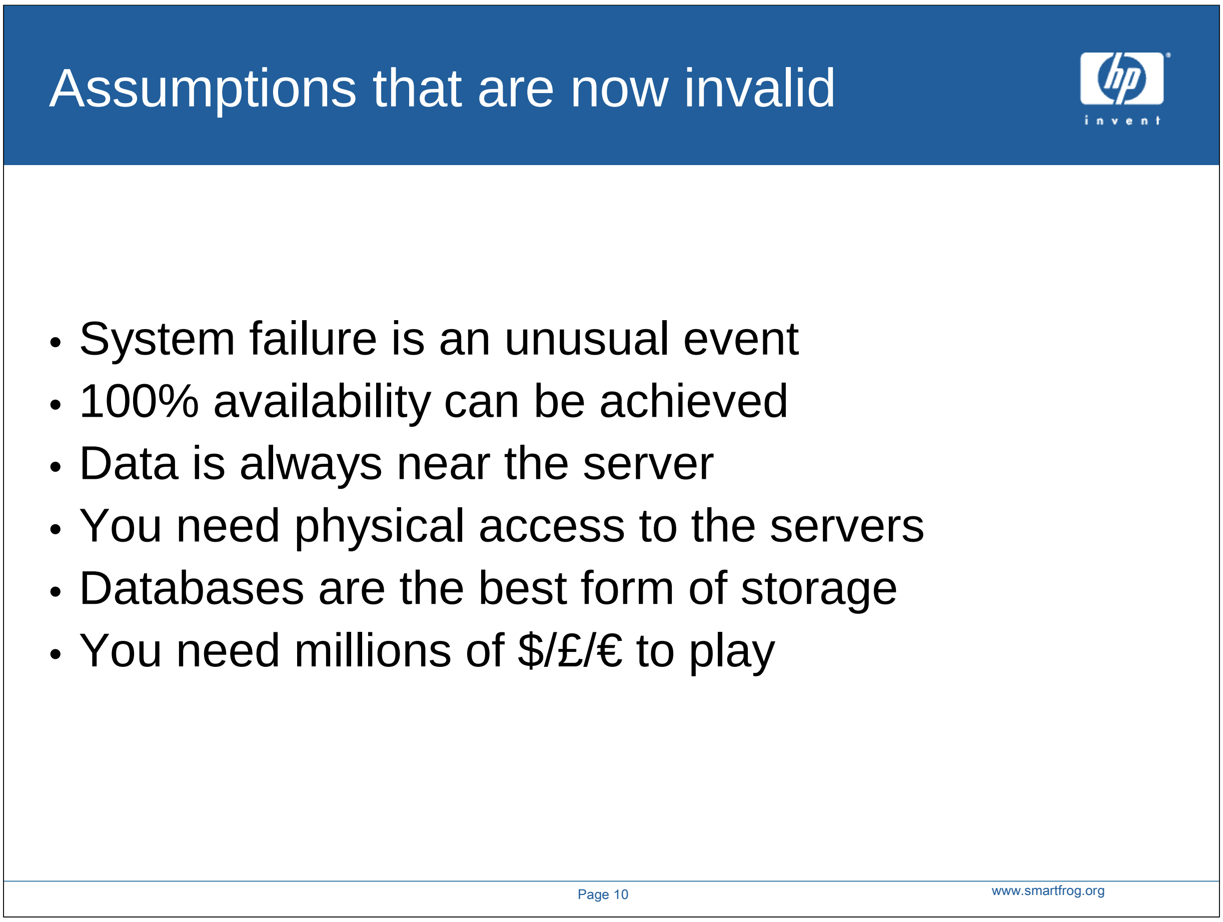
Rented storage
& CPU

Scales up

No capital outlay

http://www.linuxjournal.com/

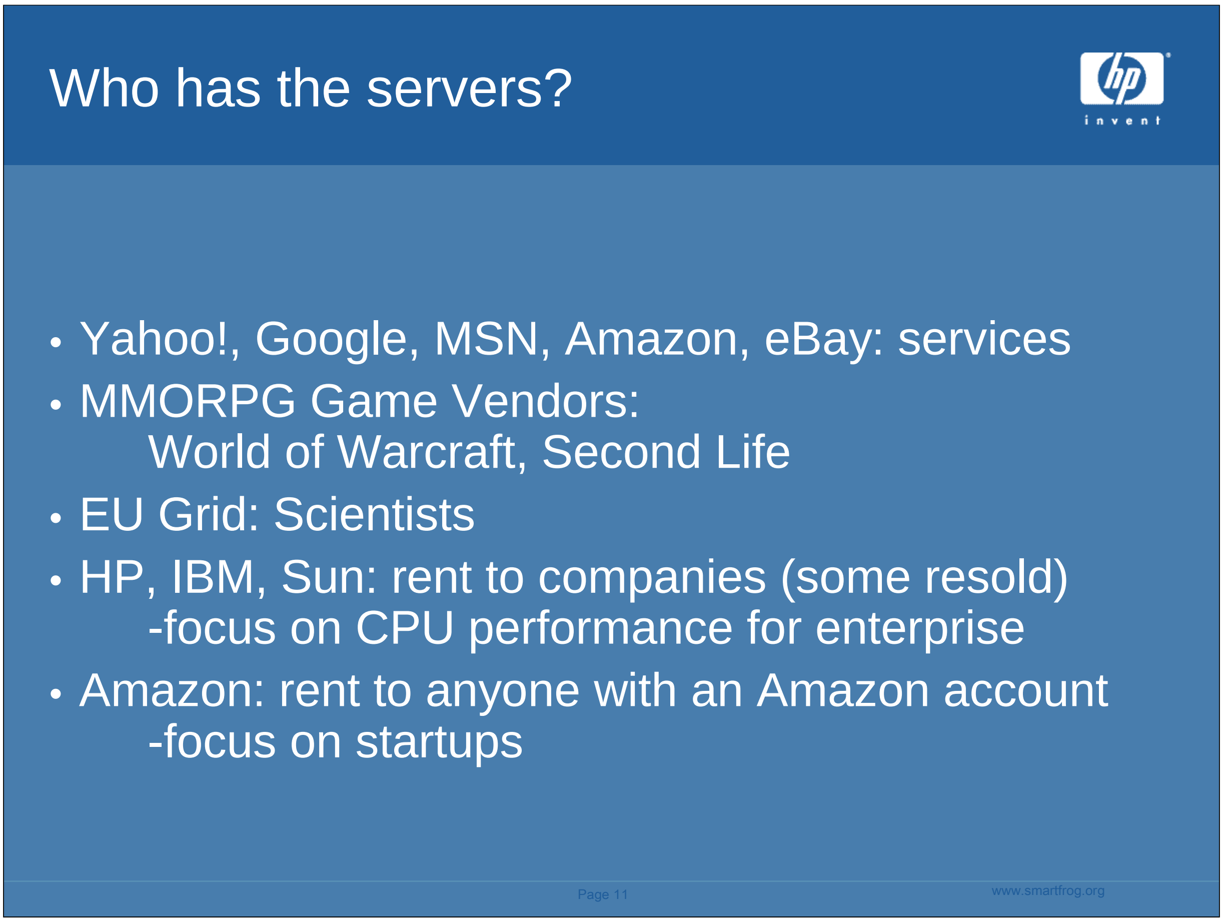
- This where things are going. No more dedicated systems; you have racks that can be repurposed based on demand. Servers are now *agile*.



Assumptions that are now invalid

- System failure is an unusual event
- 100% availability can be achieved
- Data is always near the server
- You need physical access to the servers
- Databases are the best form of storage
- You need millions of $/£/€ to play

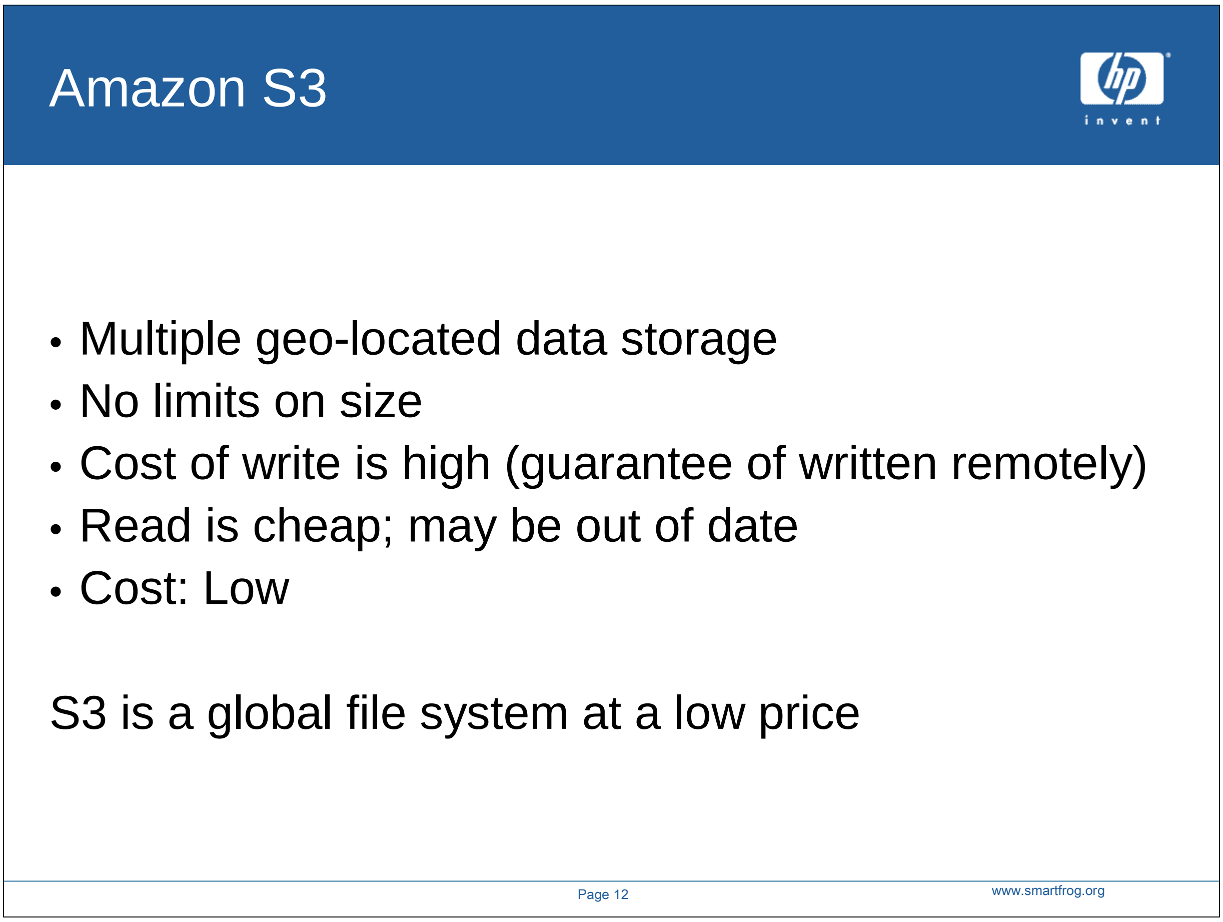
- With this many boxes, 100% uptime is unrealistic. Assume 5% disk failures in the first 8 weeks, assume one machine is always coming up. Instability is a fact of life.



Who has the servers?

- Yahoo!, Google, MSN, Amazon, eBay: services
- MMORPG Game Vendors:
  World of Warcraft, Second Life
- EU Grid: Scientists
- HP, IBM, Sun: rent to companies (some resold)
  -focus on CPU performance for enterprise
- Amazon: rent to anyone with an Amazon account
  -focus on startups

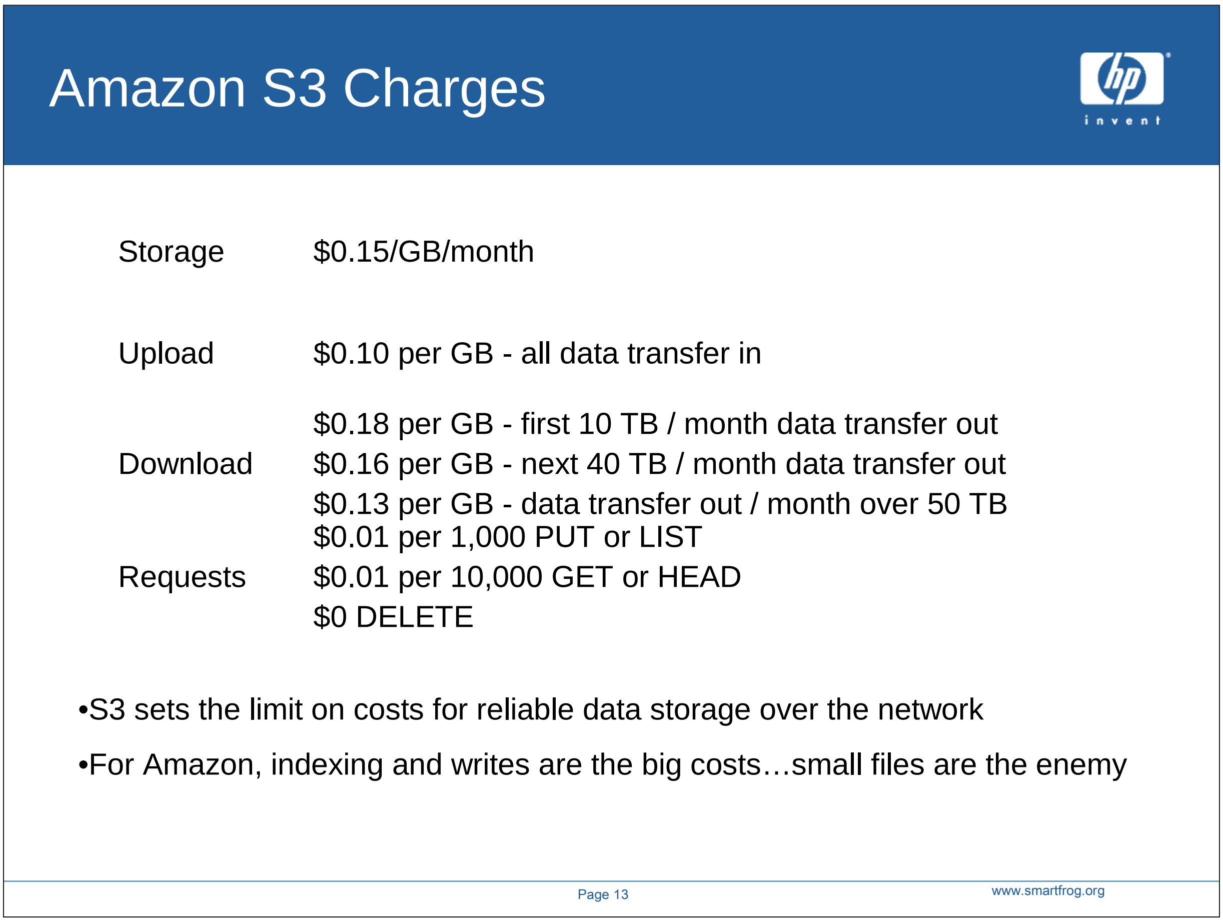
- The big server side names all have their private farms.
- So do the game companies, which is something they rarely talk about. Notice how badly Xbox lived worked over christmas though.
- The scientists get to run their jobs on the shared EU grid
- The big server vendors will sell/resell CPU+Storage to people who are interested. Their business model is fairly high-touch, and the focus is on CPU-intensive (or emergency extra load) for enterprise customers.
- Amazon is being different -they are targeting the startups, and will rent CPU time to anyone with a valid credit card.



Amazon S3

- Multiple geo-located data storage
- No limits on size
- Cost of write is high (guarantee of written remotely)
- Read is cheap; may be out of date
- Cost: Low

S3 is a global file system at a low price

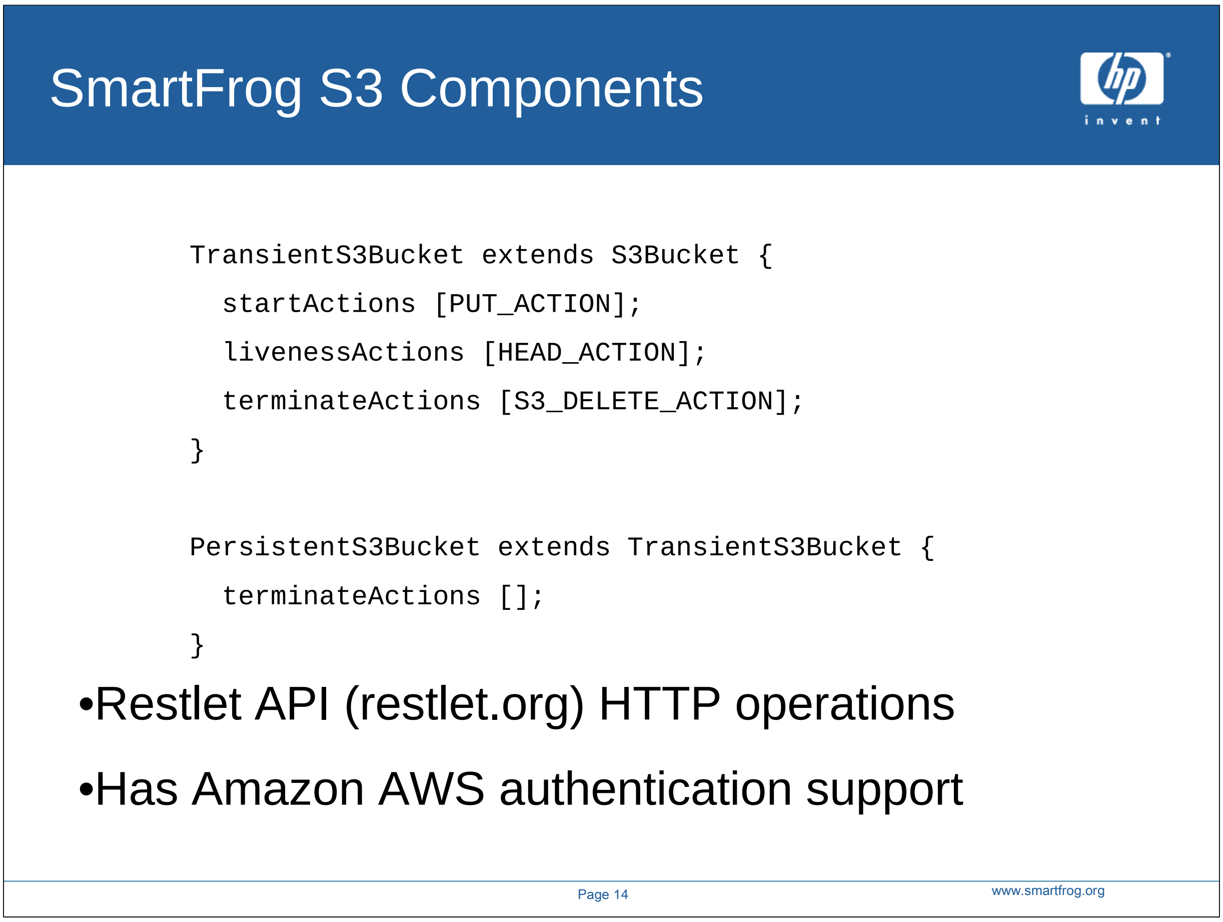


Amazon S3 Charges

Storage	$0.15/GB/month
Upload	$0.10 per GB - all data transfer in
Download	$0.18 per GB - first 10 TB / month data transfer out
	$0.16 per GB - next 40 TB / month data transfer out
	$0.13 per GB - data transfer out / month over 50 TB
Requests	$0.01 per 1,000 PUT or LIST
	$0.01 per 10,000 GET or HEAD
	$0 DELETE

- S3 sets the limit on costs for reliable data storage over the network
- For Amazon, indexing and writes are the big costs…small files are the enemy

- Google's GFS is optimised for big, 100GB+ files. In S3, the average filesize is <100KB
- These values here set the bar on what bulk storage with a good SLA should cost.



SmartFrog S3 Components

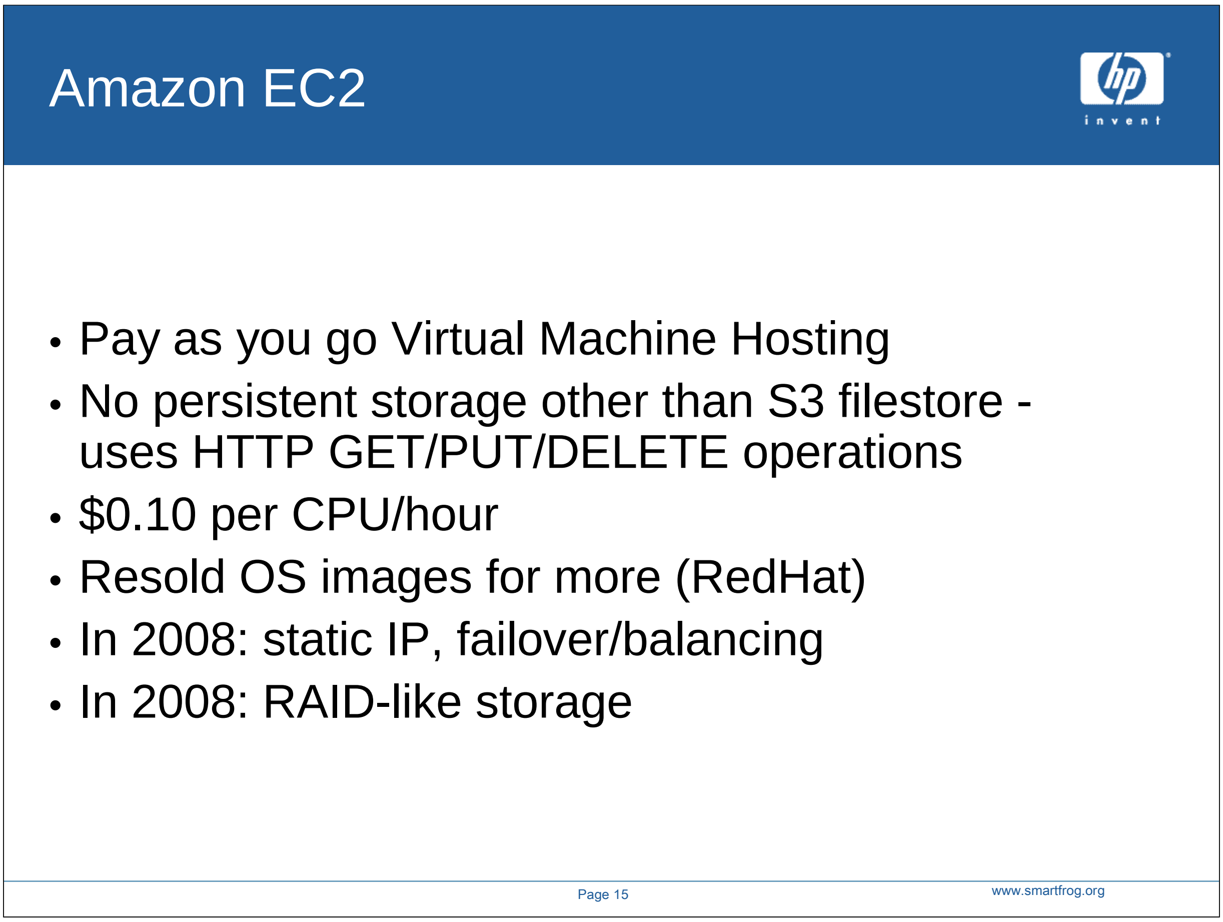
```
TransientS3Bucket extends S3Bucket {
  startActions [PUT_ACTION];
  livenessActions [HEAD_ACTION];
  terminateActions [S3_DELETE_ACTION];
}

PersistentS3Bucket extends TransientS3Bucket {
  terminateActions [];
}
```

•Restlet API (restlet.org) HTTP operations

•Has Amazon AWS authentication support

- How do we do this? Restlet.



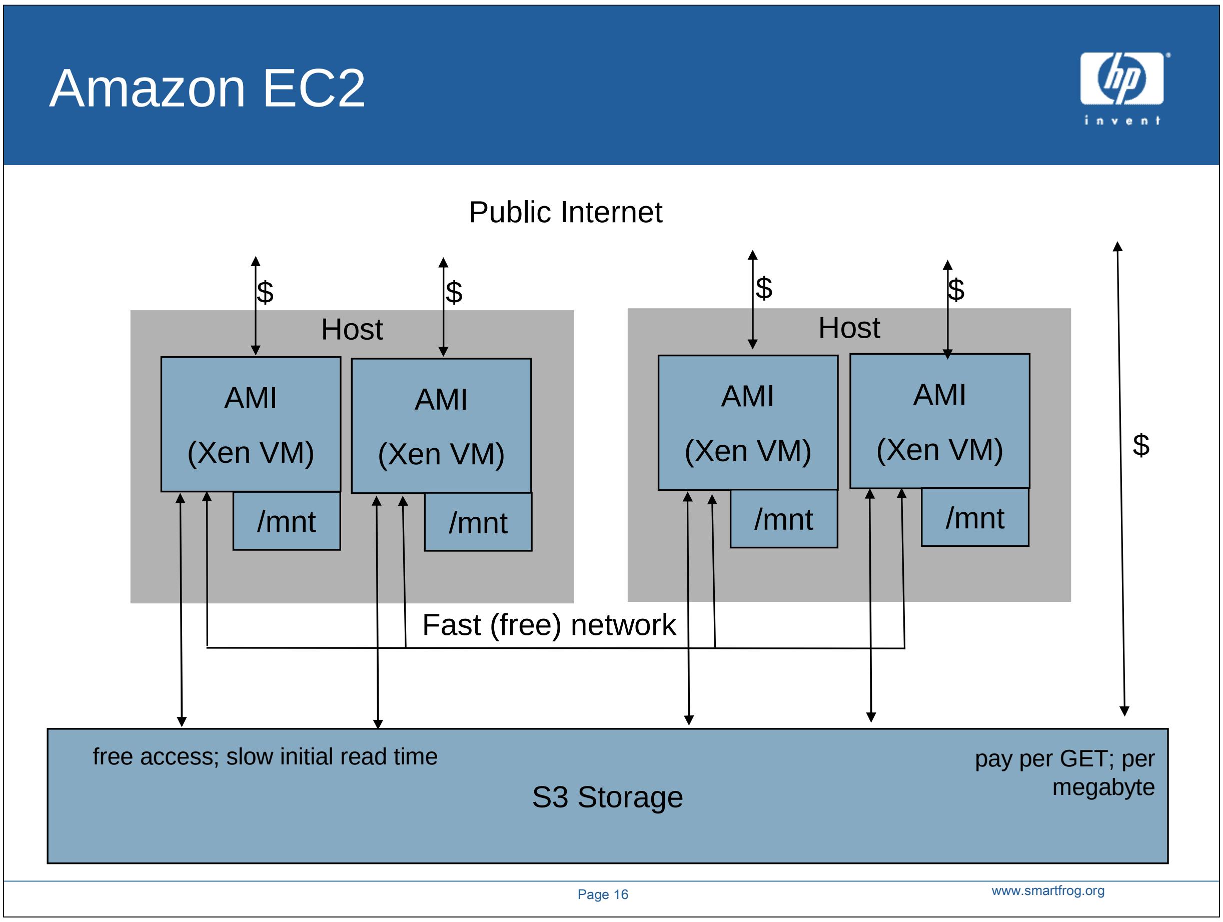
Amazon EC2

- Pay as you go Virtual Machine Hosting
- No persistent storage other than S3 filestore - uses HTTP GET/PUT/DELETE operations
- $0.10 per CPU/hour
- Resold OS images for more (RedHat)
- In 2008: static IP, failover/balancing
- In 2008: RAID-like storage

Page 15 www.smartfrog.org

- Anyone can become a reseller of CPU time

Amazon EC2


Public Internet

Host

AMI

(Xen VM)

/mnt

$

Fast (free) network

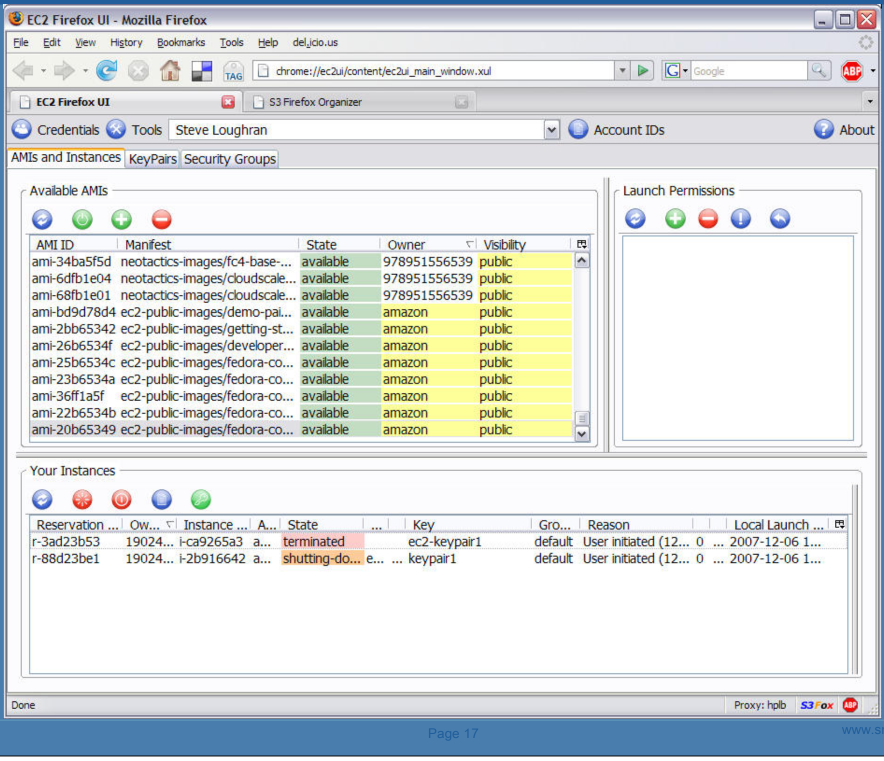
free access; slow initial read time

S3 Storage

pay per GET; per megabyte

- This is what the EC2 farm appears like
- Machines running 32-bit/64-bit linux, hosting Xen machine images
- Paid access off site; free communications between machines and with the S3 filestore
- No multicast IP, all firewall settings controlled outside the machine images
- The only persistence is the S3 filestore, which is designed to be a long-haul datastore. It is not a network mounted drive, and you must not assume so
- S3 is used for storing machine image, for serving up data directly to users without going via EC2 servers (every asset has a URL)

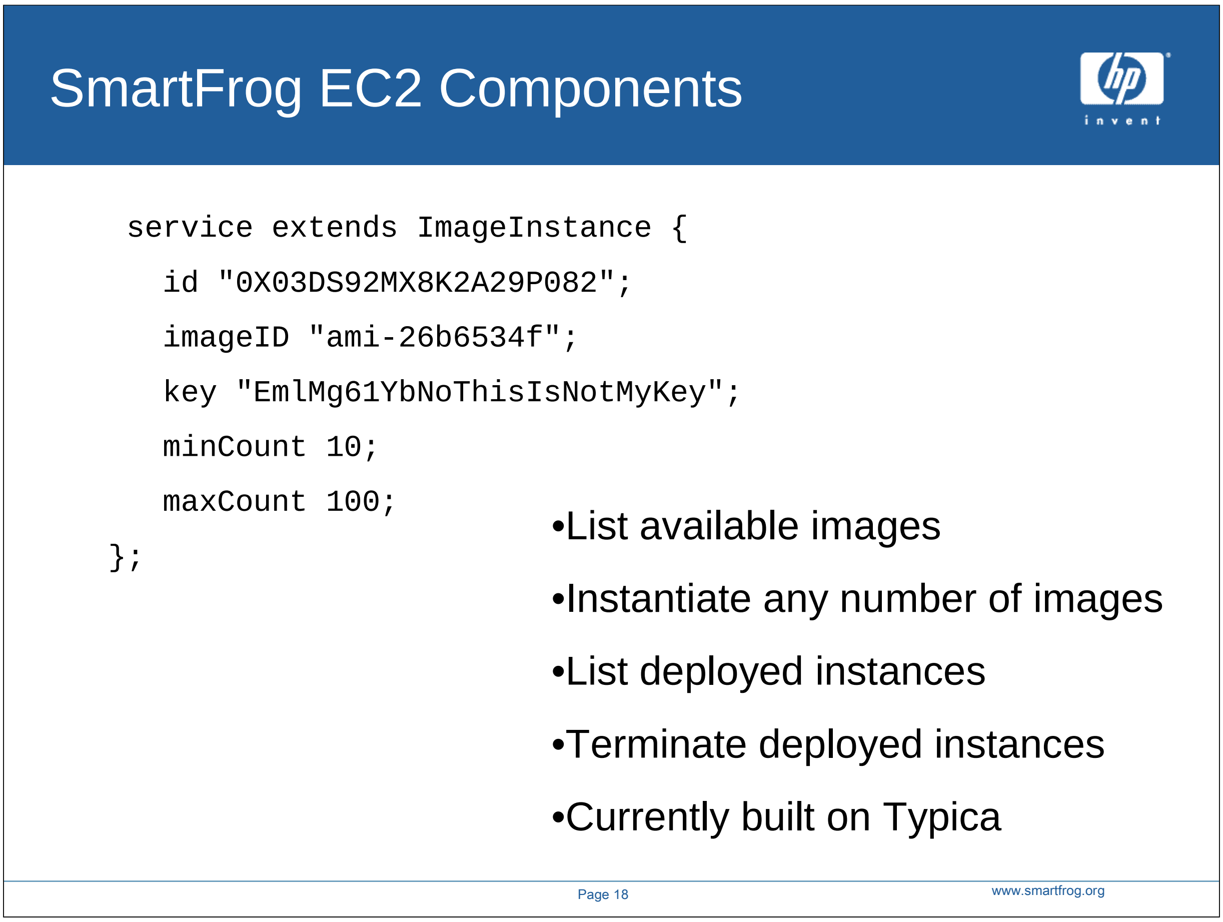
Demo



AMI ID	Manifest	State	Owner	Visibility
ami-34ba5f5d	neotactics-images/fc4-base-...	available	978951556539	public
ami-6dfb1e04	neotactics-images/cloudscale...	available	978951556539	public
ami-68fb1e01	neotactics-images/cloudscale...	available	978951556539	public
ami-bd9d78d4	ec2-public-images/demo-pai...	available	amazon	public
ami-2bb65342	ec2-public-images/getting-st...	available	amazon	public
ami-26b6534f	ec2-public-images/developer...	available	amazon	public
ami-25b6534c	ec2-public-images/fedora-co...	available	amazon	public
ami-23b6534a	ec2-public-images/fedora-co...	available	amazon	public
ami-36ff1a5f	ec2-public-images/fedora-co...	available	amazon	public
ami-22b6534b	ec2-public-images/fedora-co...	available	amazon	public
ami-20b65349	ec2-public-images/fedora-co...	available	amazon	public

Reservation ...	Ow...	Instance ...	A...	State	...		Key	Gro...	Reason			Local Launch ...
r-3ad23b53	19024...	i-ca9265a3	a...	terminated			ec2-keypair1	default	User initiated (12...	0	...	2007-12-06 1...
r-88d23be1	19024...	i-2b916642	a...	shutting-do...	e...	...	keypair1	default	User initiated (12...	0	...	2007-12-06 1...

- This is the current management GUI. They use SOAP or an HTTP query interface to talk to the management code…new management tools are planned in future.

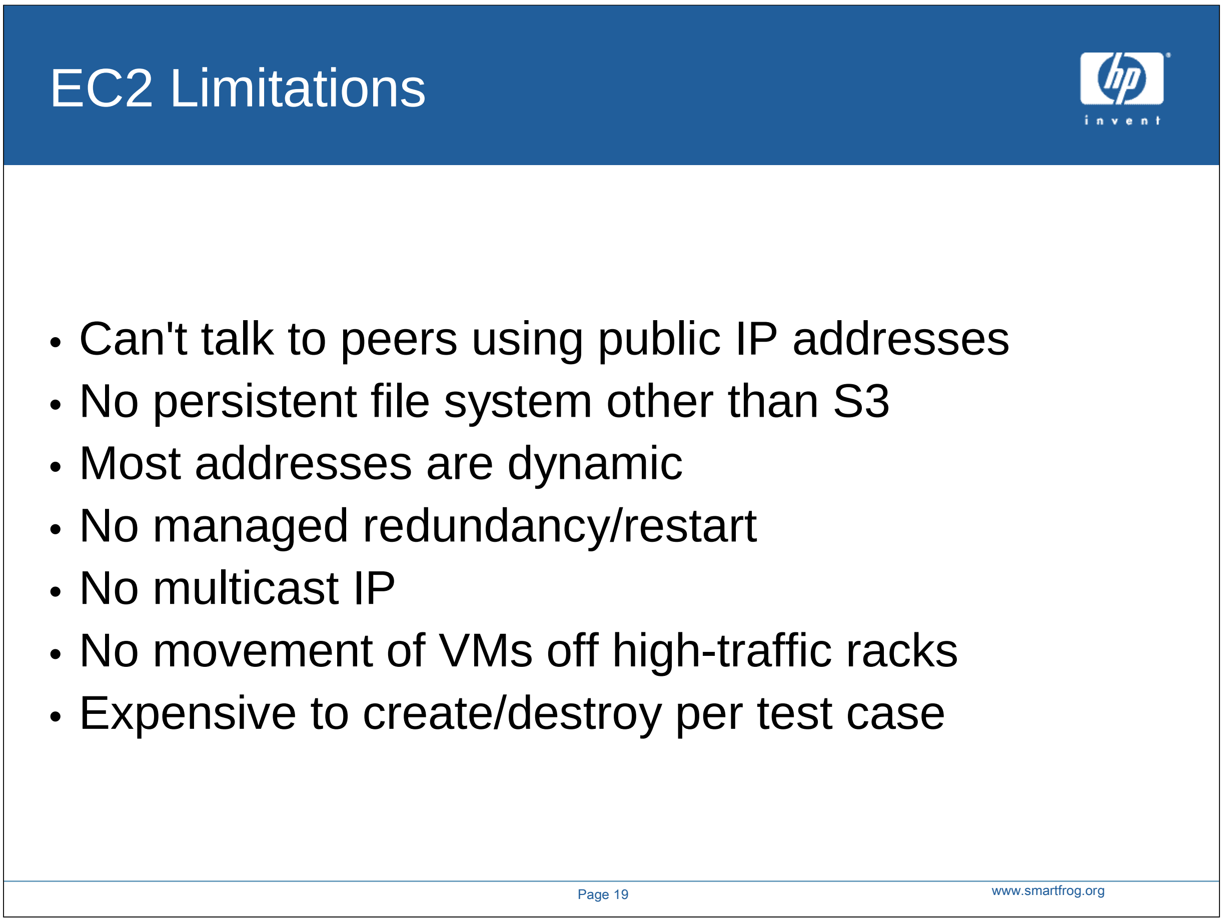


SmartFrog EC2 Components

```
 service extends ImageInstance {
   id "0X03DS92MX8K2A29P082";
   imageID "ami-26b6534f";
   key "EmlMg61YbNoThisIsNotMyKey";
   minCount 10;
   maxCount 100;
};
```

- List available images
- Instantiate any number of images
- List deployed instances
- Terminate deployed instances
- Currently built on Typica

- Typica is a library on google open source; uses JAXB underneath. JAXB is a key troublespot for me.

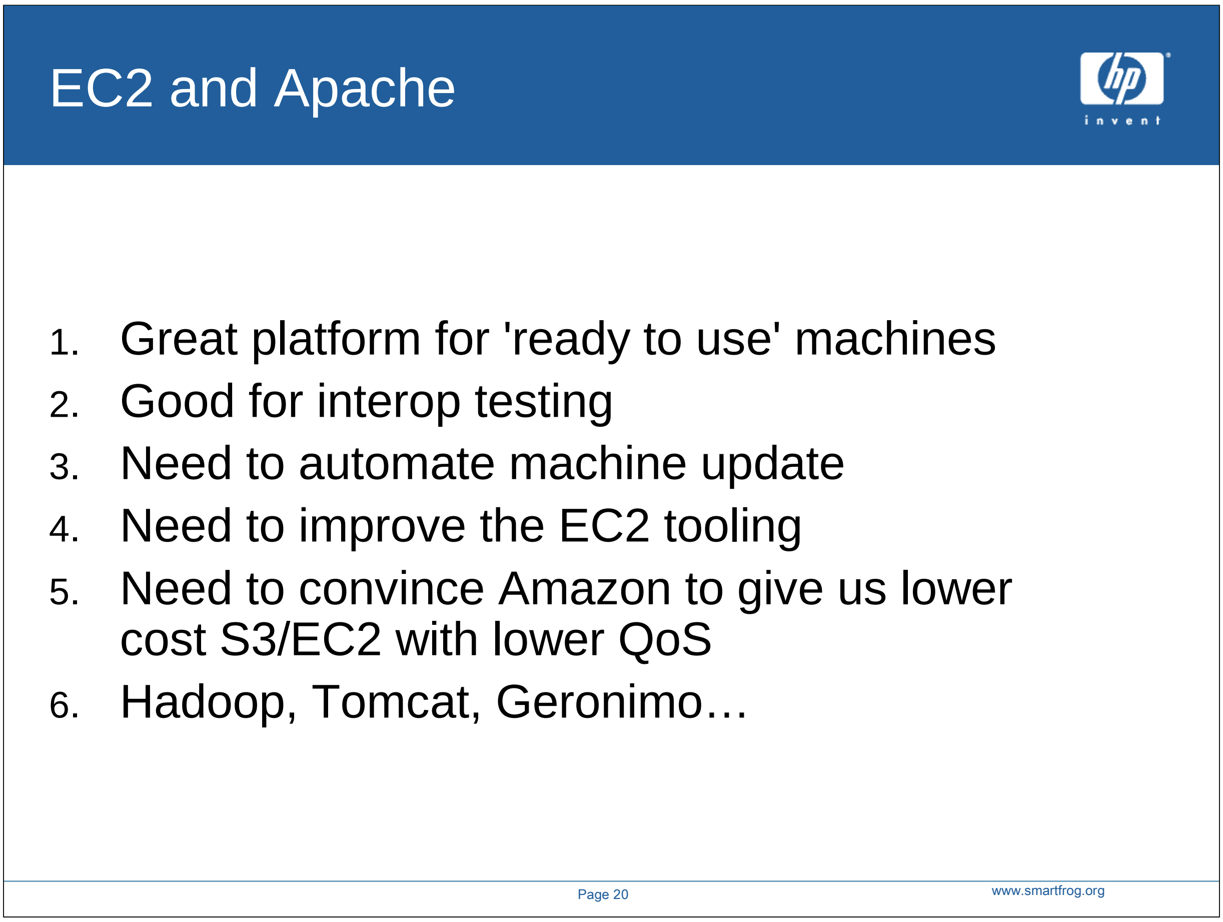


EC2 Limitations

- Can't talk to peers using public IP addresses
- No persistent file system other than S3
- Most addresses are dynamic
- No managed redundancy/restart
- No multicast IP
- No movement of VMs off high-traffic racks
- Expensive to create/destroy per test case

Here are some of the limitations of EC2. It's clear this stuff is restricted w.r.t real servers.

- For some of the back office stuff, the lack of multicast stops a lot of the clustering protocols from working. We could change Anubis to do some other bootstrap mechanism and then switch to UDP to multicast, though it causes total network traffic to go up O(N*N) instead of O(N) where N=#nodes.
- Not being able to use public ipaddrs introduces new complications - you cant use nslookup of dyndns-hosted machine names to determine your peers. You can use Jabber against, say, google talk though, at least as a bootstrap. Hey, you could even use UDDI!
- You need to keep an eye on the system from the outside.



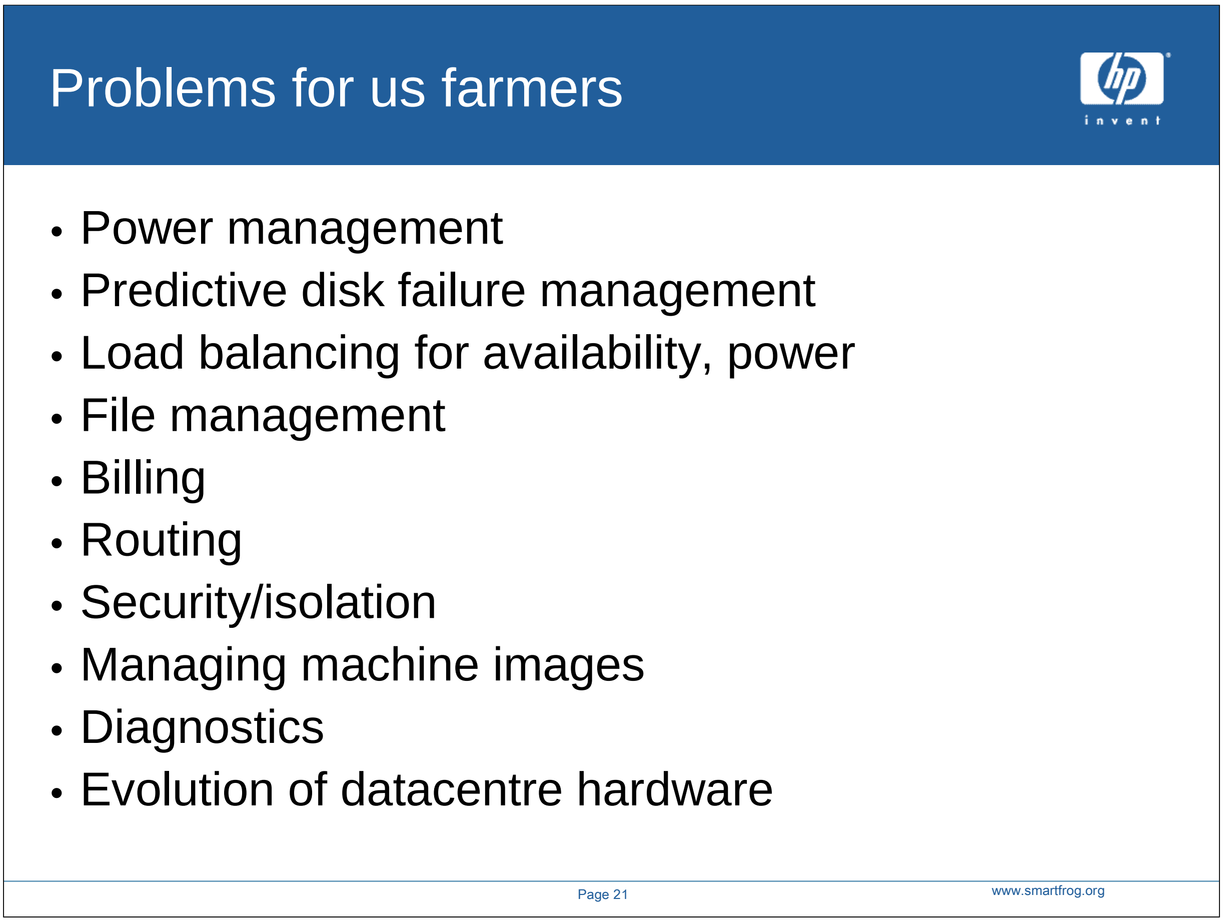
EC2 and Apache

1. Great platform for 'ready to use' machines
2. Good for interop testing
3. Need to automate machine update
4. Need to improve the EC2 tooling
5. Need to convince Amazon to give us lower cost S3/EC2 with lower QoS
6. Hadoop, Tomcat, Geronimo…

Page 20

www.smartfrog.org

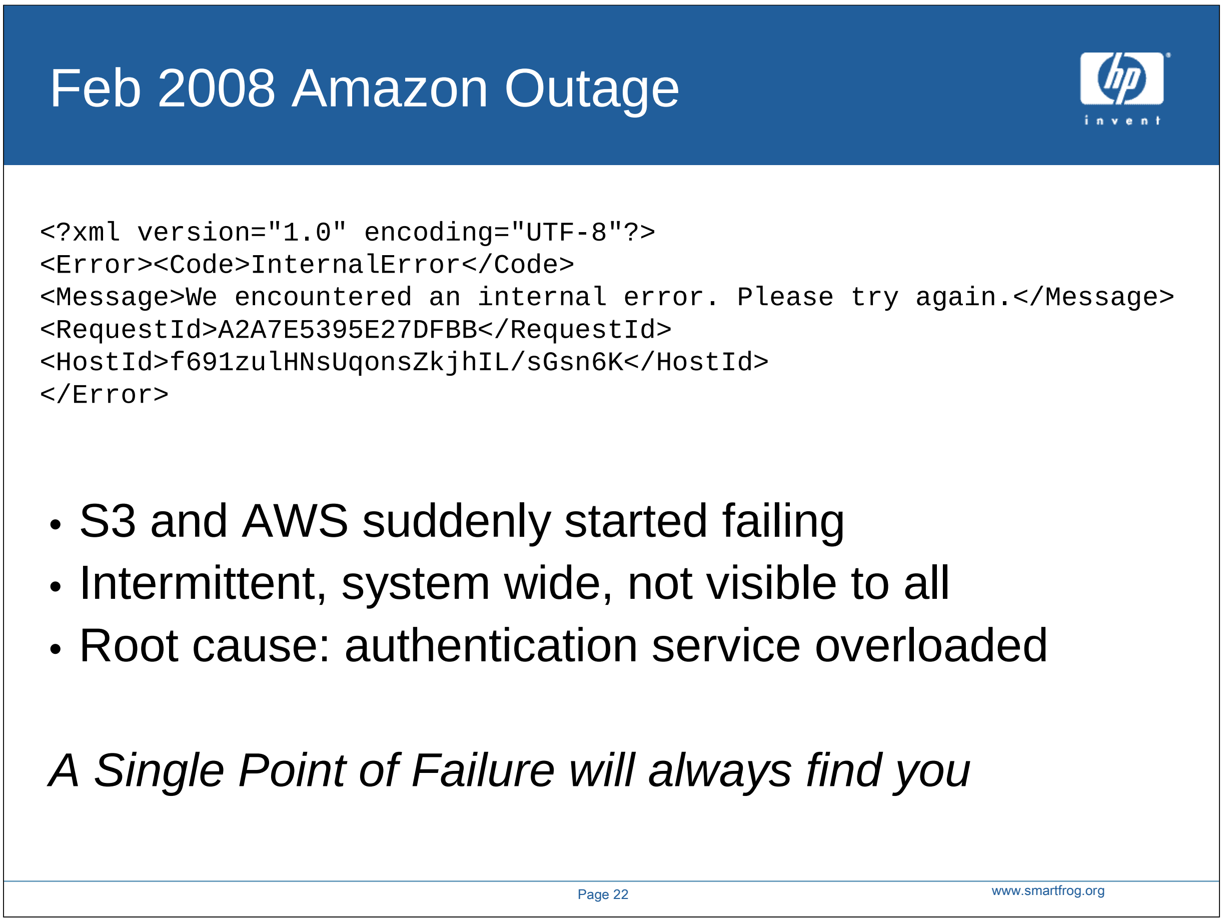
Here are some off-the-shelf thoughts.



Problems for us farmers

- Power management
- Predictive disk failure management
- Load balancing for availability, power
- File management
- Billing
- Routing
- Security/isolation
- Managing machine images
- Diagnostics
- Evolution of datacentre hardware

- These are our problems!



Feb 2008 Amazon Outage

```
<?xml version="1.0" encoding="UTF-8"?>
<Error><Code>InternalError</Code>
<Message>We encountered an internal error. Please try again.</Message>
<RequestId>A2A7E5395E27DFBB</RequestId>
<HostId>f691zulHNsUqonsZkjhIL/sGsn6K</HostId>
</Error>
```

- S3 and AWS suddenly started failing
- Intermittent, system wide, not visible to all
- Root cause: authentication service overloaded

A Single Point of Failure will always find you

Page 22 www.smartfrog.org

- side topic; what happened in Feb 2008 when S3 and EC2 went down. They were still there, but the authorisation servers overloaded (Amazon AWS authentication is home rolled and designed to work over HTTP, and to let you hand off a request to someone else for reuse.
- It clealry took a while to track down the problem, and while that happened the backlog of people trying to use the services piled up; retrying clients created even more overloaded.
- Lots of panic in the forum, 'fix this now!'. Problem is, until AWS had identified the problem, they couldn’t fix it, and shouting doesn’t help
- Moral: you always have a single point of failure, a SPOF. How do you find it? You wait. It finds you.