Scaling WebApps

June 4 2013 Volta Halifax
Lunch and Learn

Tim Chipman
FortechITSolutions.ca

PLEASE NOTE

These slides are provided as a courtesy to those who attended the seminar.
If you like them, please feel free to distrubute the URL to friends and colleagues, and they can download the PDF directly.
PLEASE DO NOT REDISTRUBUTE OR BUILD YOUR OWN CONTENT FROM THESE – without contacting me.
Thanks!

Context & Disclaimer :-)

- Limited Time, Big Topic. This is only an 'Overview'
- But - please ask questions, help make this as relevant for you if possible!

Busdev Slide

Fortech IT Solutions

- Halifax Based IT Consulting Services
- Server Management, Virtualization, expertise with open-source platforms
- 5 years and still going, (15+ yrs experience)
- Need help? I love interesting projects.
- Special deal, extra value, VMs for rent :-)

(902) 442-6633

Tim.Chipman@FortechITSolutions.ca

http://FortechITSolutions.ca

Talking about What ?

- Background / Intro / Context
- Old School
- Slightly less Old School: LAMP
- Horizontal scaling
- Tuning & Tweaking
- "AAS" platforms
- Considerations: Design, Targets, Planning
- Best Pick ?
- Some good tools to know of.
- (Questions, comments, Discussion -

Credit where due

Almost all images in the presentation are shamelessly snaffled from google images searches

virtually nothing here was created by me

mwah ha hahahah!

(maniacal laugh, maniacal laugh)

Context

You have a service

Clients want the service

Clients will use the service

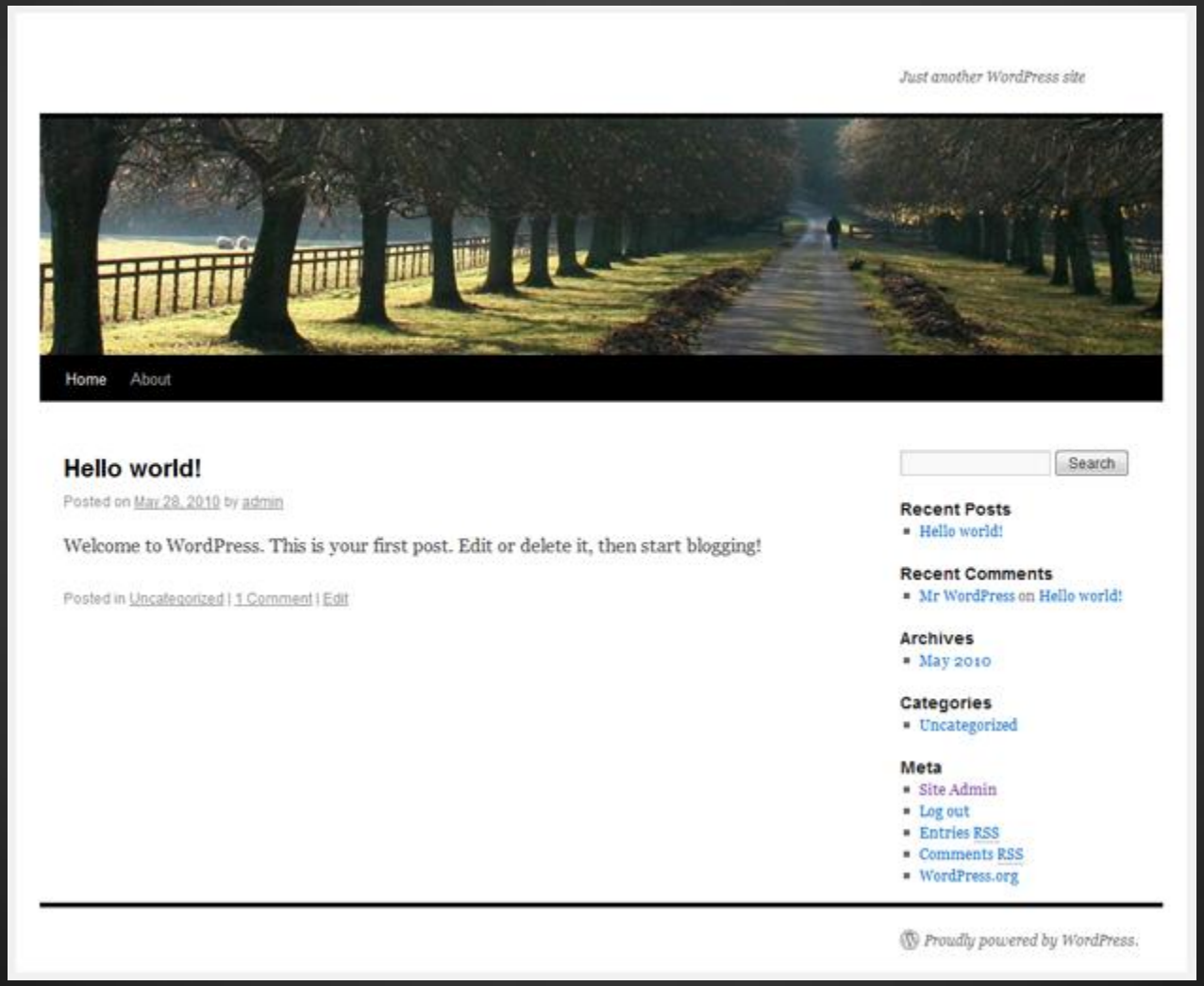
Some clients will pay for the service

You have a business model ?

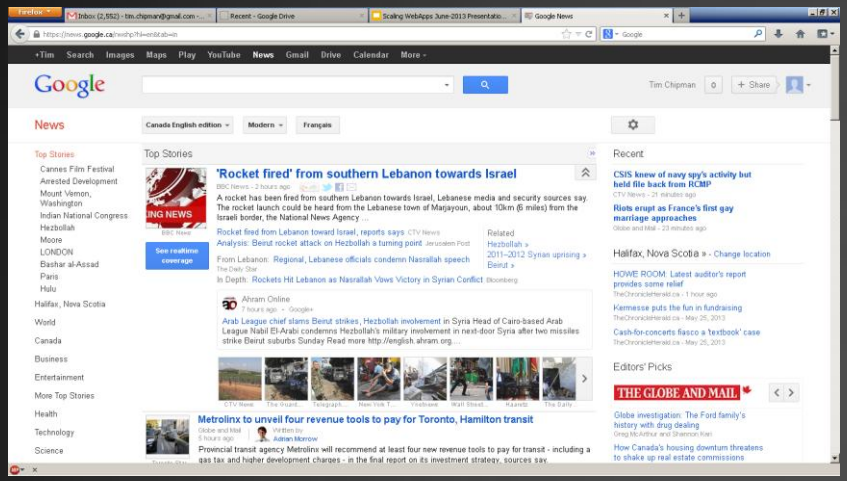
You have a plan ?

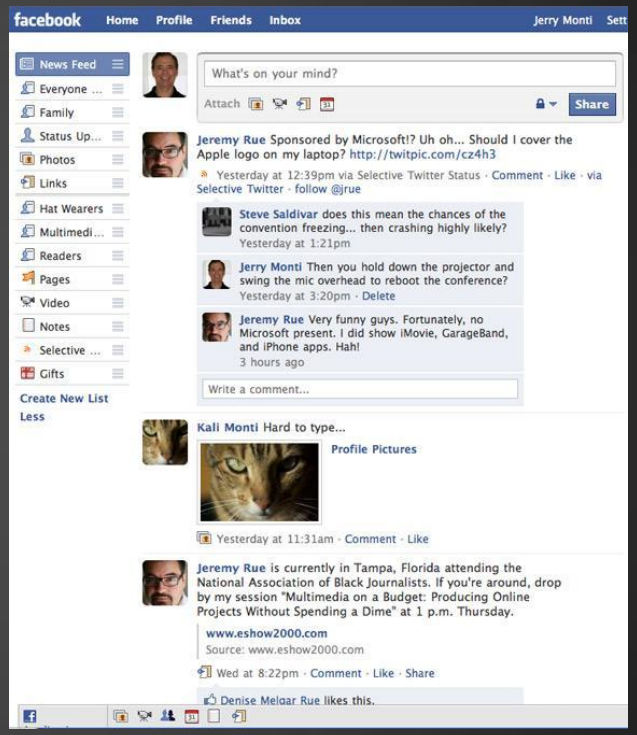
Old world: static web pages

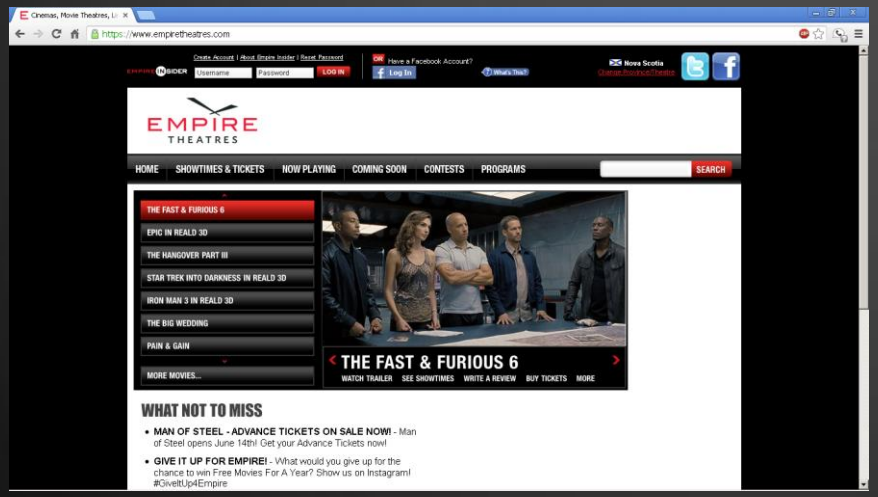
New World: Dynamic, Personalized



Better Examples of New World...







Back to what we're talking about...

You have a service

Clients want the service

Clients will use the service

Some clients will pay for the service

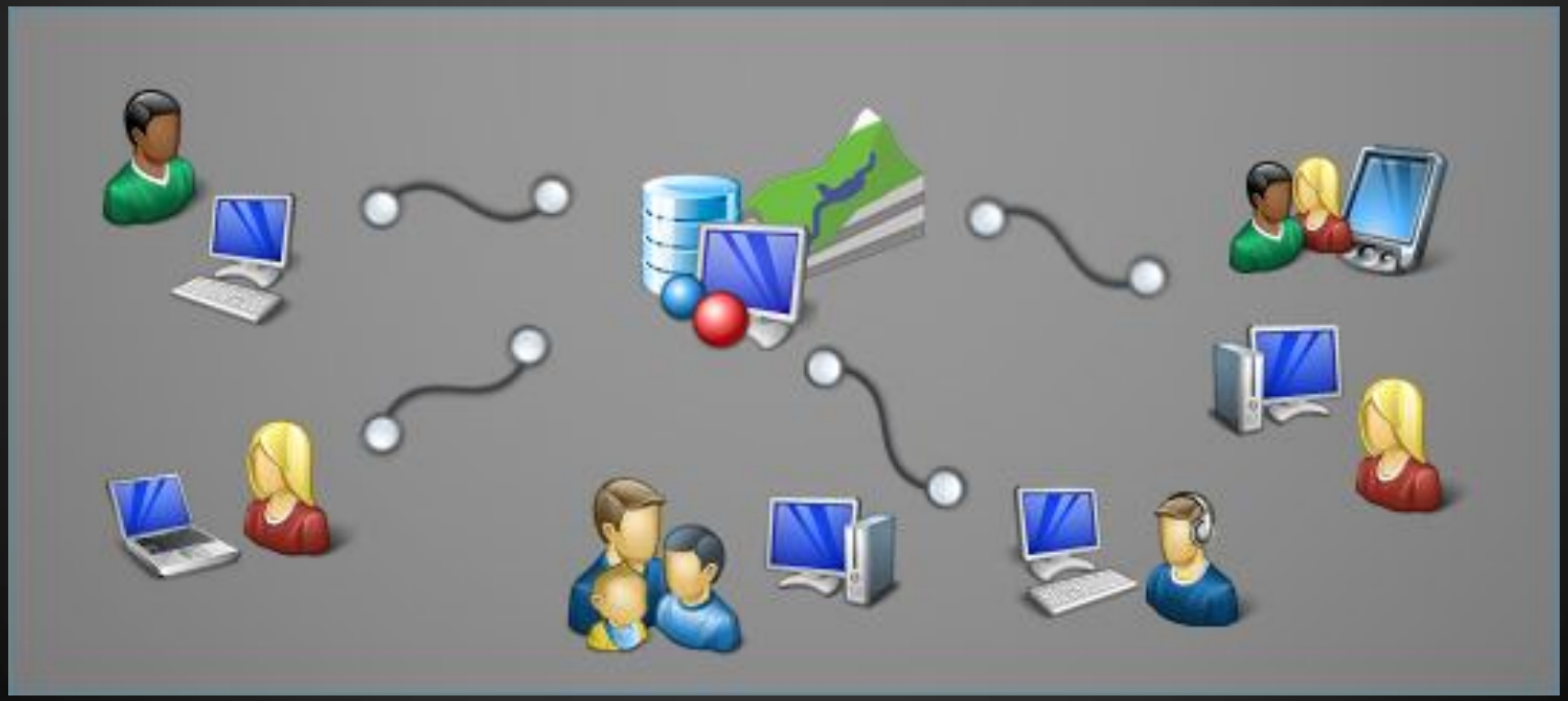
You have a business model ?

You have a plan ?

Background - client server model

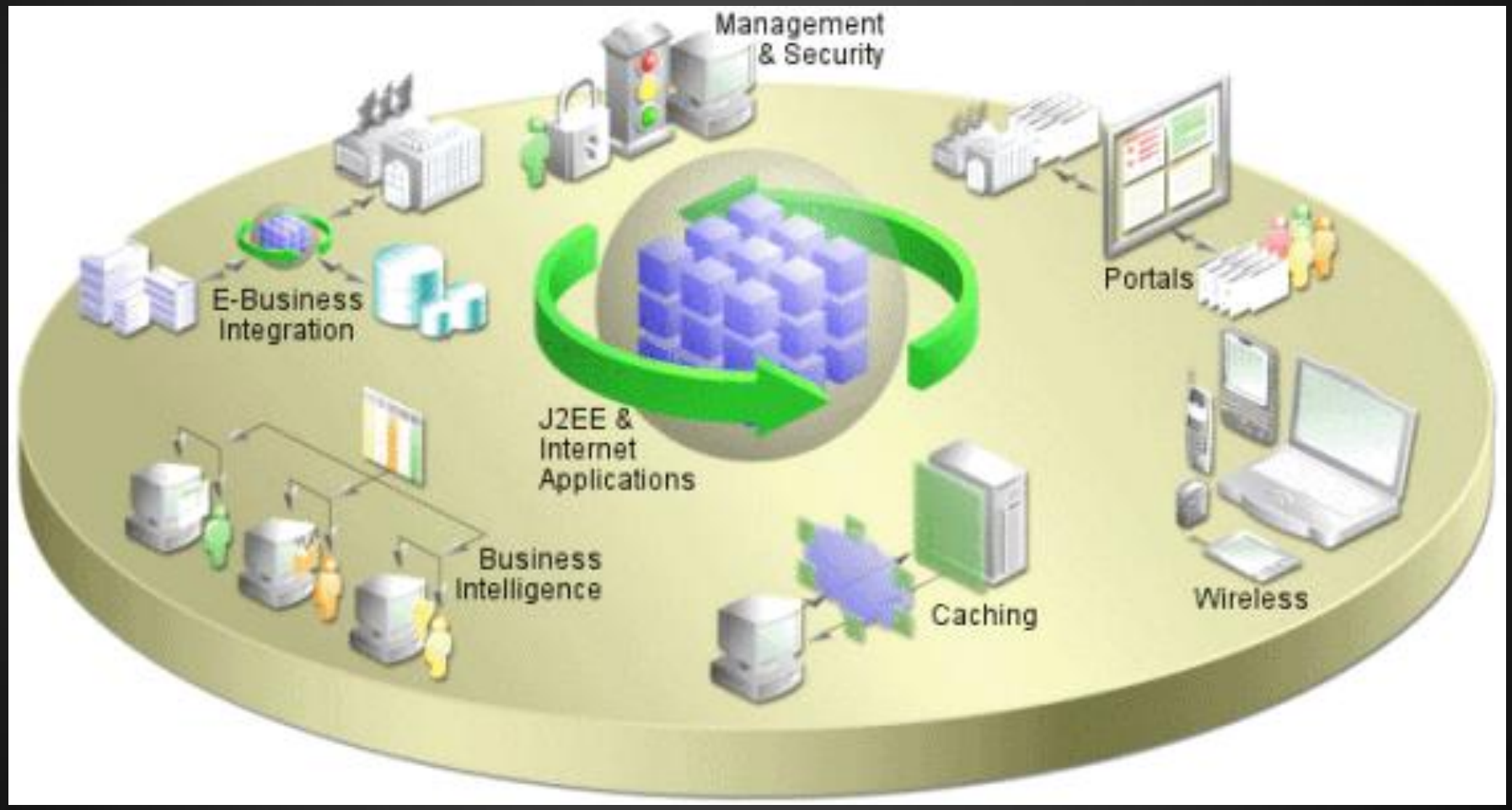
accessibility

availability



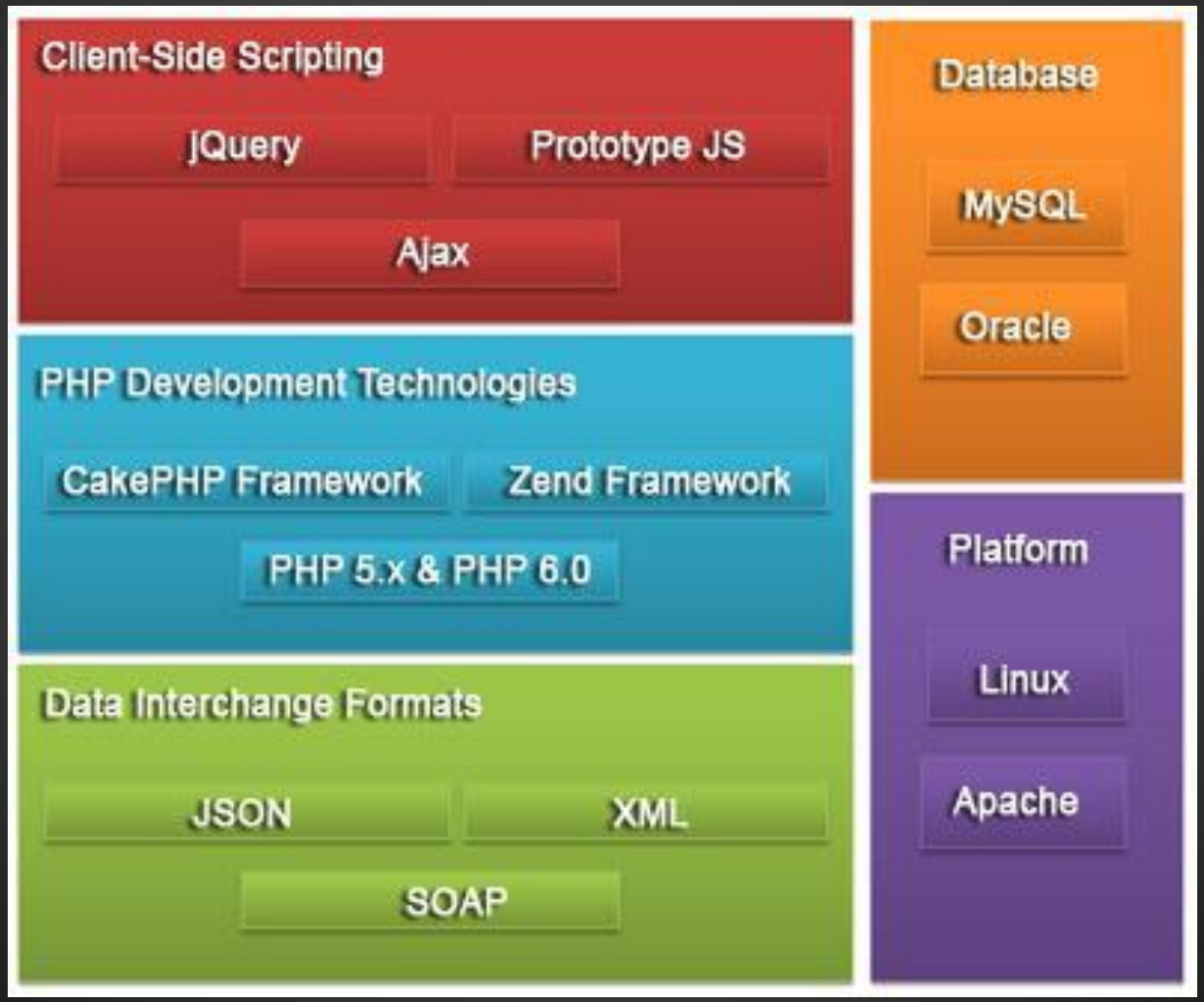
Old School

Old School


Management
& Security
E-Business
Integration
Portals
J2EE &
Internet
Applications
Business
Intelligence
Caching
Wireless

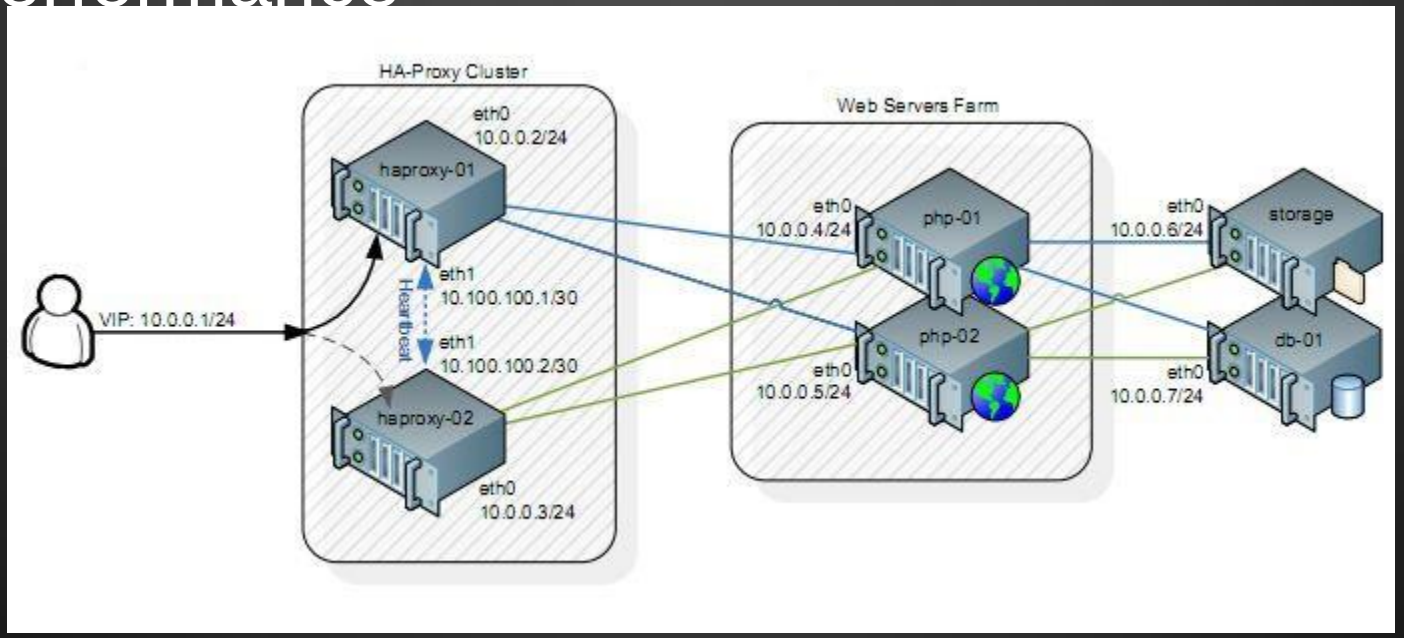
"Sort of" Less Old-School: LAMP


php
MySQL
Perl
python
LINUX
APACHE
LAMP
Linux
Apache
MySQL
PHP,Perl,Python


Client-Side Scripting
jQuery
Prototype JS
Ajax
PHP Development Technologies
CakePHP Framework
Zend Framework
PHP 5.x & PHP 6.0
Data Interchange Formats
JSON
XML
SOAP
Database
MySQL
Oracle
Platform
Linux
Apache

Conventional Approach

- monolithic servers, designated roles
- specific data flow, points of failure
- horizontal scale out for redundancy, performance



Conventional Approach

Complexity of management

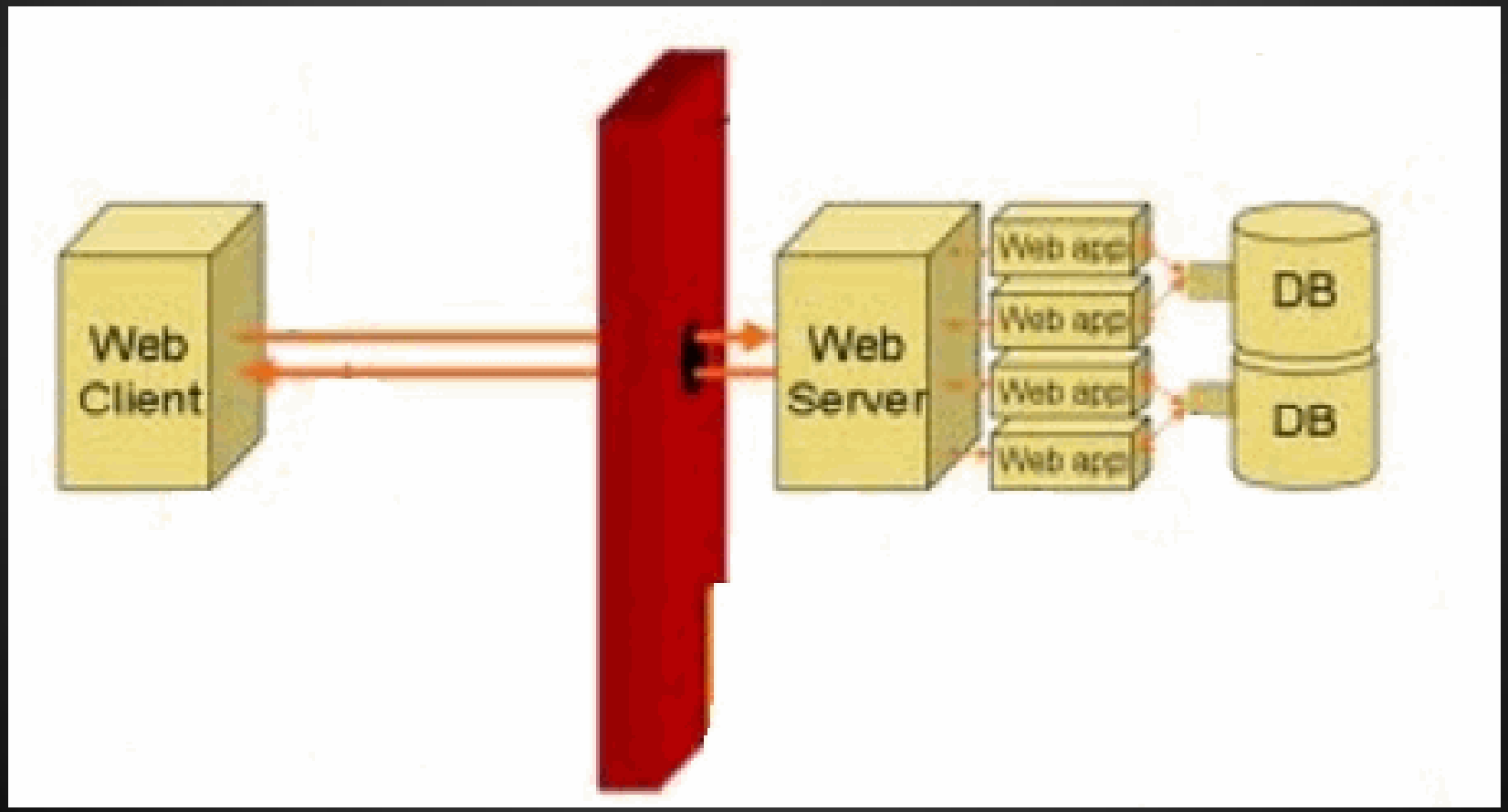
- "Physical stuff"; OS
- Components (DB, Web, App) plus extras (App security model, network security, etc)





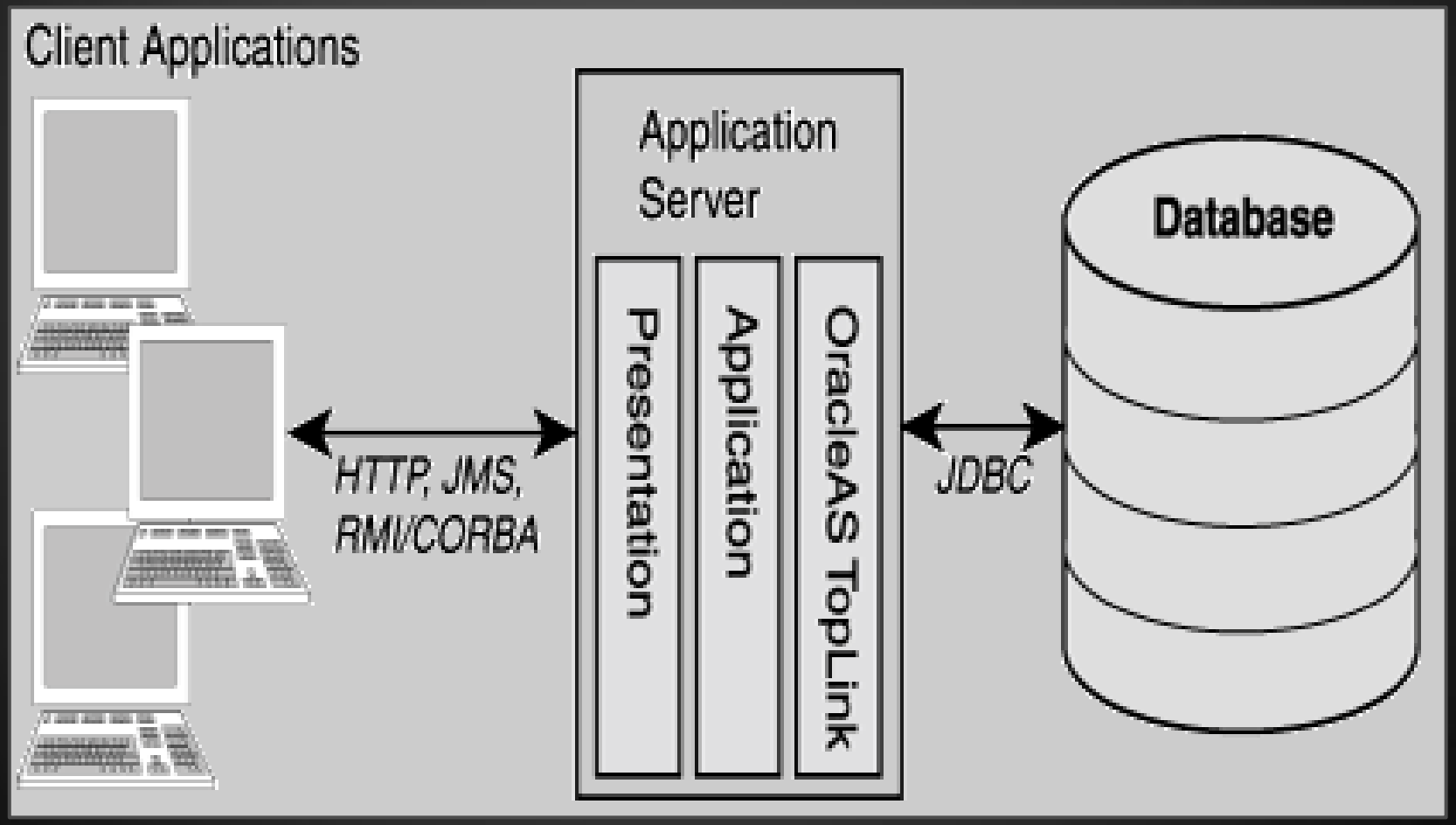
Conventional Approach

What does this all look like?



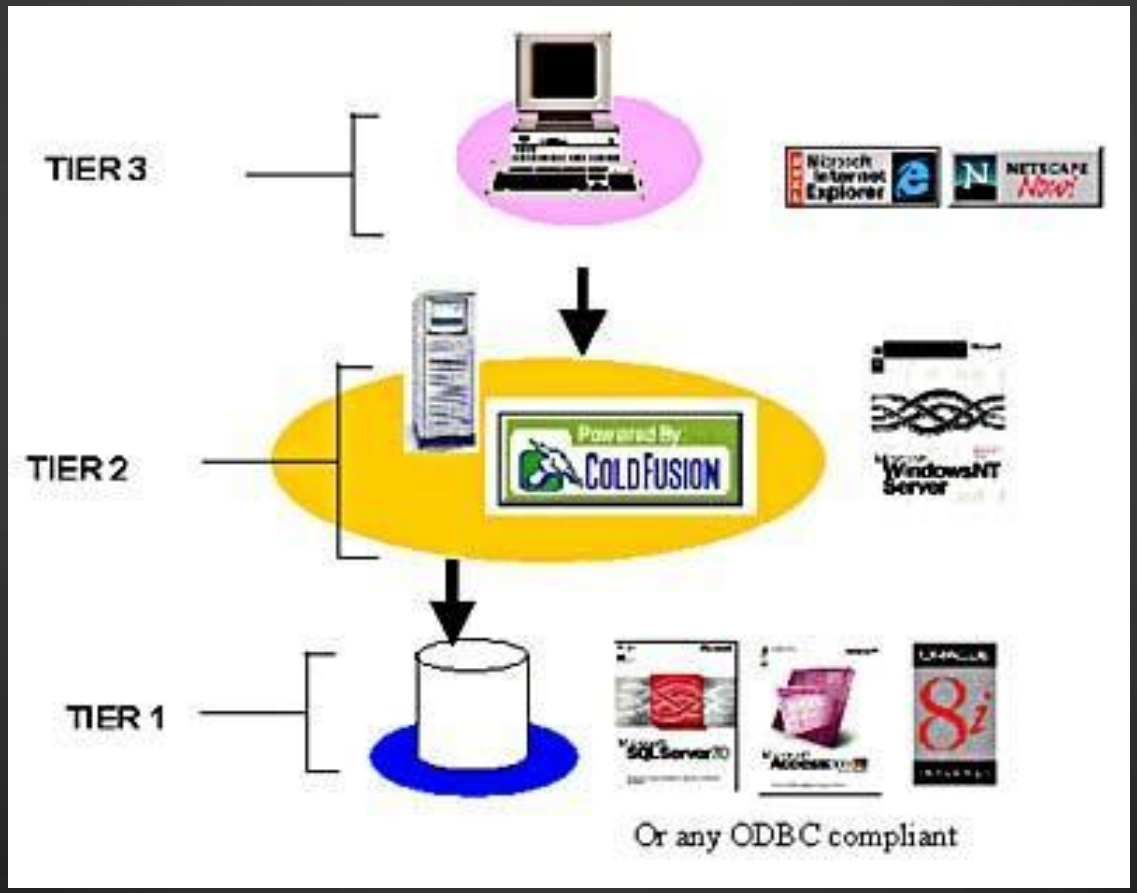
Conventional Approach

Or maybe this:



Conventional Approach

Even more amusing:



Conventional Approach

Long and short of it?

- Specific vendor hardware (Sun, IBM, etc)
- Specific OS (Solaris, AIX, etc)
- Specific DB (Oracle, DB2, etc)
- Specific App Server (WebLogic, Websphere)
- Each slice in the stack takes $$$

Conventional Approach - LAMP

Long and short of it?

- hardware (Commodity x86)
- Linux OS (Debian, CentOS, Ubuntu, etc)
- Open DB (MySQL, Postgres, etc)
- Open 'App Server Layer' (Tomcat/Java or Php, Python, Ruby, etc)
- Each slice in the stack takes less $$$

Open Source pieces, commodity hardware, but same structures.

Horizontal Scaling

With open source stack and commodity hardware, scale-out is not constrained by $license costs; more by incremental 'platform expenses' and 'architecture:management'

c-(lamp) --> ccccc-(lb)-(lap)(lap)(lap)(lap)(lm)

or

c-(lamp) --> c^x-(lb)-(lap)^y(lm-r)(lm-r)(lm-w)

Horizontal scaling

Step 1: crank resources; "cpu/ram is cheap"

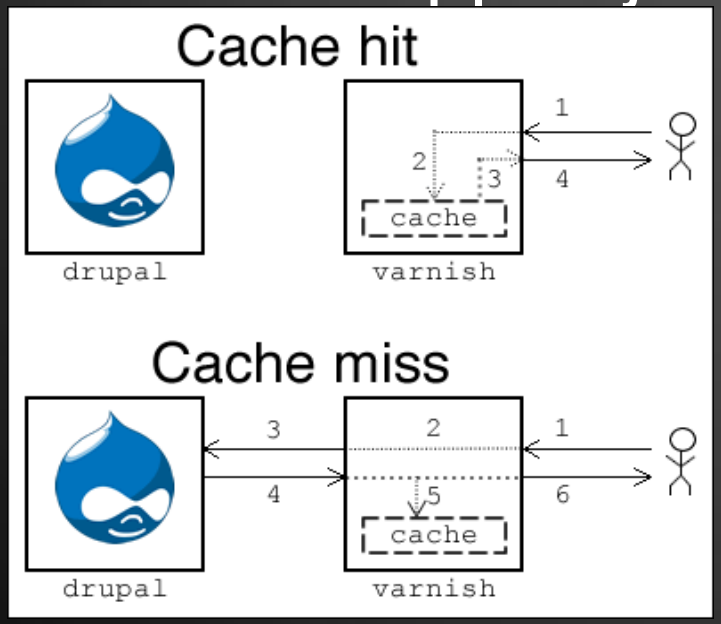
Step 2: Not enough? Divide and conquer:

- DB read vs DB write
- static web vs dynamic web content
- Delivery static from different host &/use CDN
- load balancer(s) to fan out (client facing typically)

Tweaking & Tuning

Cache

- web: nginx, varnish - proxy or subdivision / static / pseudo-dynamic
- rendered app objects (php-apc)



Tweaking & Tuning

- db queries (memcached)
- App: Internal cache, logic, optimization

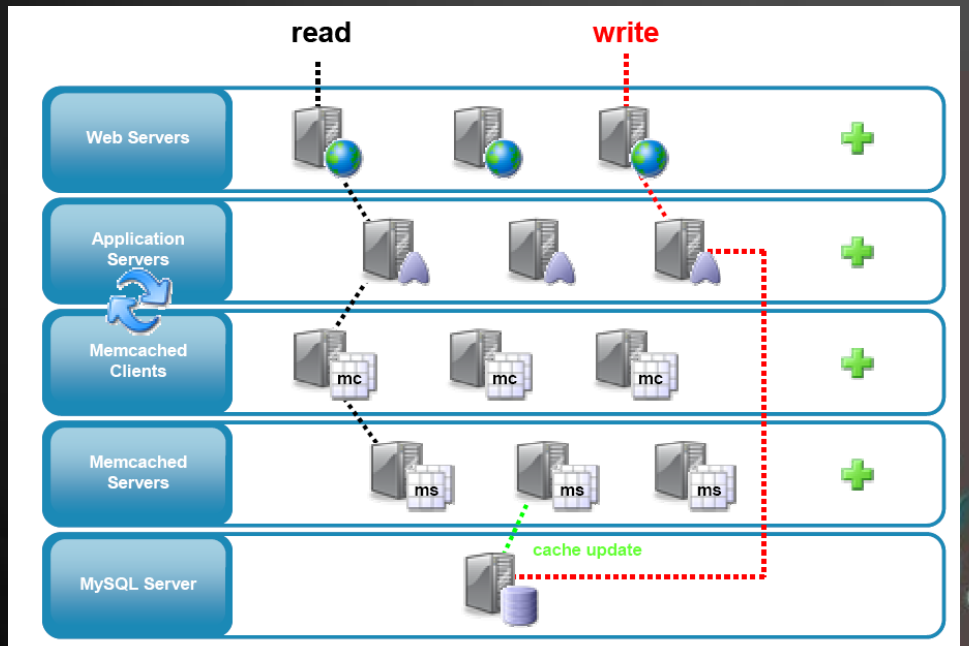
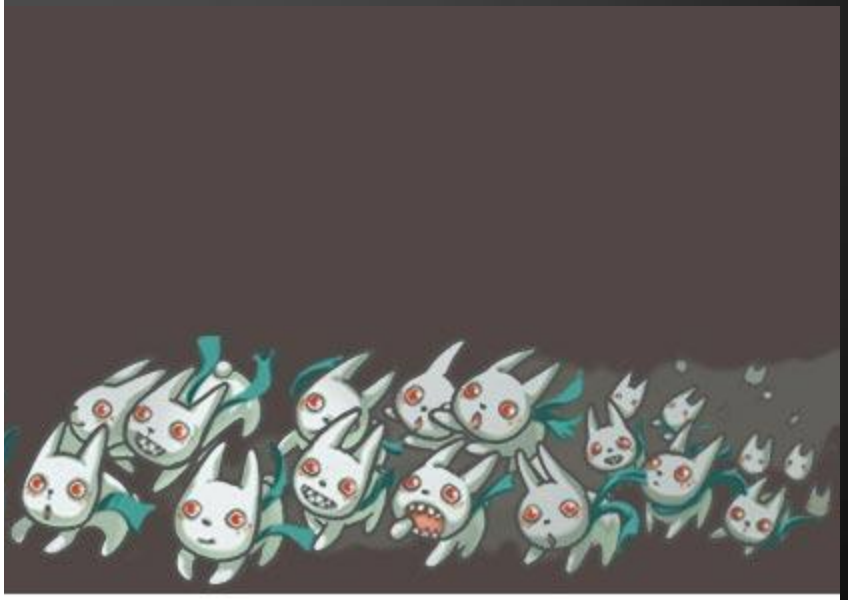


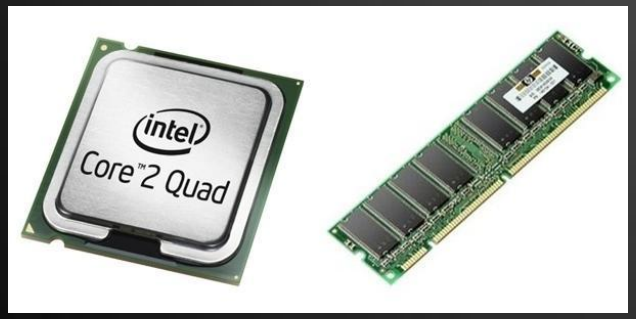
Figure 2: Multiple Memcached Servers and a Stand-Alone MySQL Server

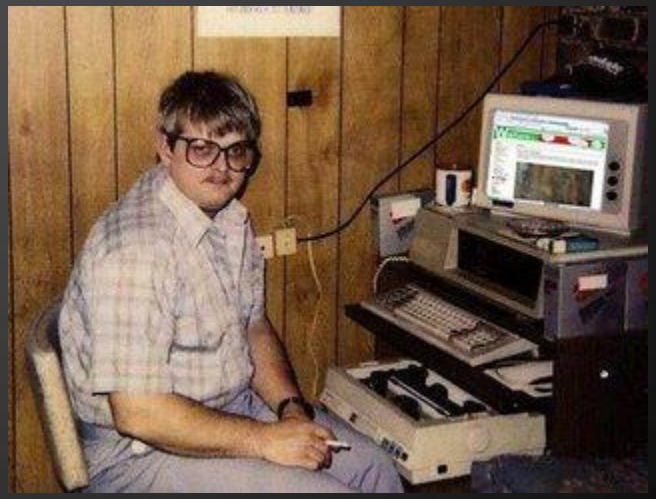


Tweaking & Tuning

Simple physical

- lower latency storage for DB, high-demand IO subsystems (Ram, SSD, SASRaid10)
- higher bandwidth / lower latency (IB vs Ether? 10g vs 1g vs 100M vs ..)





LAMP / Horizontal Scaled / Tweaked

Bottom line:

lots can be done. But it can take a lot of time:effort. Is this really what drives your business?

AAS Model

"As A Service"

- abstracts components of service for delivery
- "simplification" of model

Trade offs?

- constrained tools / specific APIs
- harder to build than with 'conventional toolset'
- pricing, lock in

Rule one of I.T. Club

...Rule 1 of IT club is you don't talk about prices....

"Amazon has broken that rule in a rather spectacular manner, and now there's hell to pay"

Source: http://www.theregister.co.uk/2013/05/30/amazon_cloud_killing_trad_it/

re:

"Morgan Stanley analysts write in a report, Amazon Web Services: Making Waves in the IT Pond, that was released on Wednesday. Brocade, NetApp, QLogic, EMC and VMware are said to face the greatest "challenges" from the growth of AWS"

"Workloads are flying into AWS for several reasons, and Morgan Stanley believes the most compelling ones are:

No upfront investment

Pay for Only What You Use *

Price Transparency

Faster Time to Market

Near-infinite Scalability and Global Reach

Leveraging Scale – as AWS Grows Pricing Keeps Coming Down"

AAS

Amazon EC2

Rackspace Openstack

Google Compute

Google App Engine

Heroku

MongoHQ

Microsoft Azure

Continuum - Comparison

Pure Hardware-Dedicated Server -runs Service

- Server hardware you own, your data centre
- Server hardware you rent, 3rd party colo
- VM Servers you rent, 3rd party colo
- Google Compute, Amazon, Rackspace dynamic provisioned API managed VMs
- Google AppsEngine, Heroku, MongoHQ

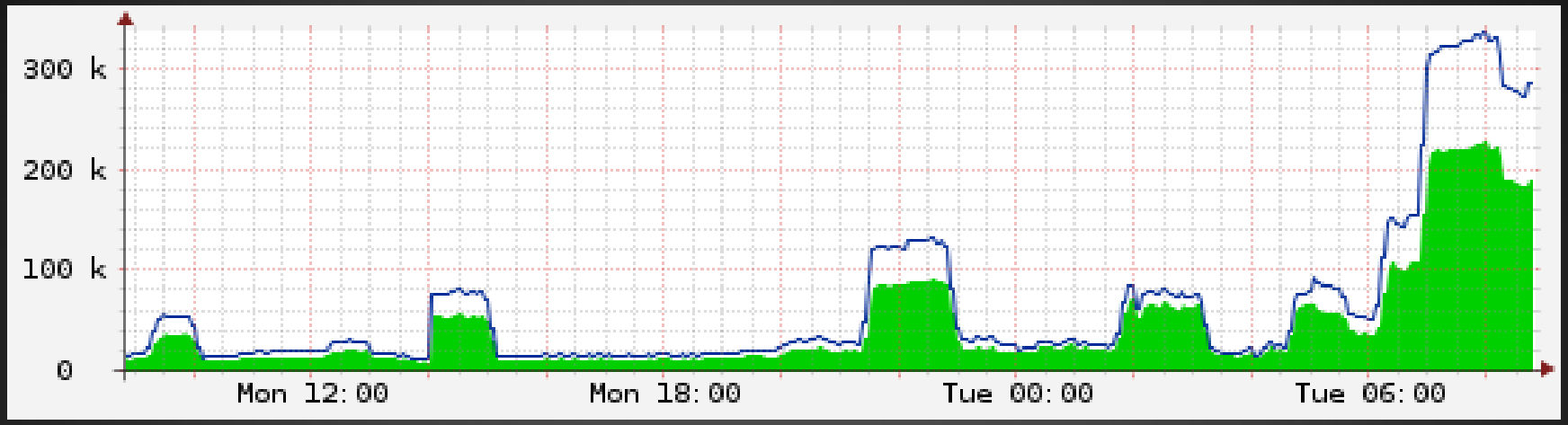
Pure Service - No Hardware, No Server

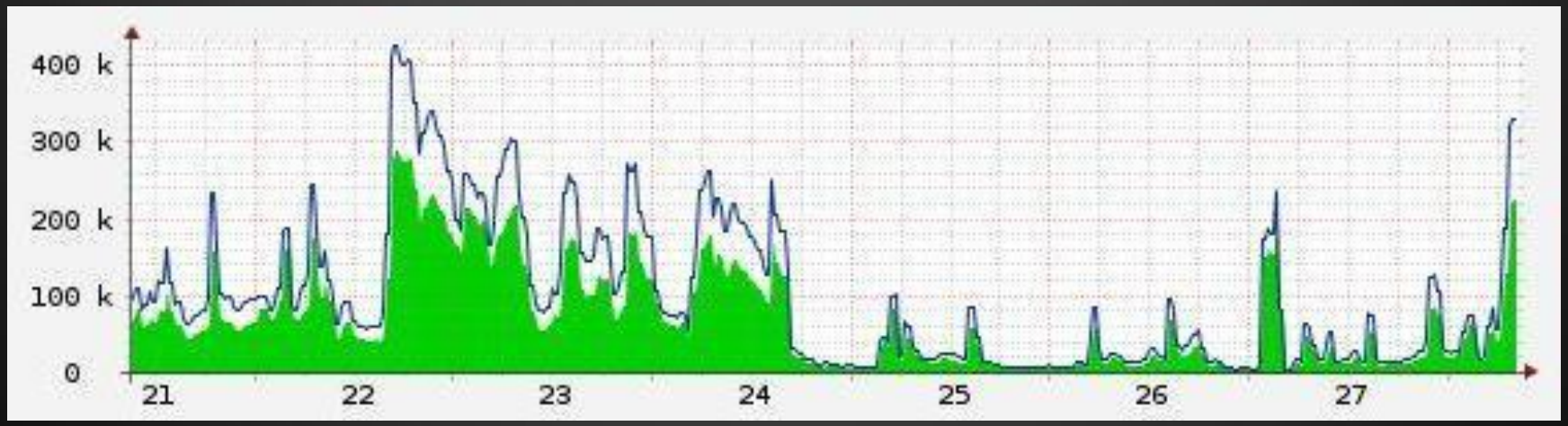
"AAS" (kind of)

Amazon EC2, Rackspace OpenStack, Google Compute:

- 3 vendors, but all same basic model.
- API managed VM instances
- Best fit for exceptionally 'peaky' workloads
- Price structure - great value if you can dynamically manage VM instances

Day, Week - Example PeakyLoad

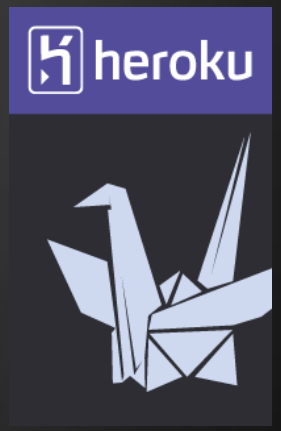




AAS

Google App Engine

- custom versions of python, java (php, go)
- custom db (NoSQL, CloudAKAMySQL, etc)
- entirely different model from Amazon. Build app and logic is core.



Heroku

- similar - abstracted / high level mgmt
- focus on processes, workers
- 'less different from normal' python:java:ruby

AAS

To some extent:

- answers the question that service providers have:
- "How can we get more revenue from our clients (you), with less service, less infrastructure, and lower costs?"
- i.e., not inherently a no-brainer 'win win' (for you)
- there is a cost for the 'simplicity and abstraction'

		RackSpace	Amazon Web Services		
$0.015/hr.	$10.80/mo.	256MB RAM / 10GB Disk			
$0.030/hr.	$21.60/mo.	512MB RAM / 20GB Disk	613MB RAM / 2 ECU (throttled) / EBS Storage Only / m1.micro	$0.020/hr.	$14.40/mo.
$0.060/hr.	$43.20/mo.	1GB RAM / 40GB Disk	1.7GB RAM / 1 ECU / 160GB Disk / m1.small	$0.085/hr.	$61.20/mo.
$0.120/hr.	$86.40/mo.	2GB RAM / 80GB Disk			
$0.240/hr.	$172.80/mo.	4GB RAM / 160GB Disk			
$0.480/hr.	$345.60/mo.	8GB RAM / 320GB Disk	7.5GB RAM / 4 ECU / 850GB Disk / m1.large	$0.340/hr.	$244.80/mo.
$0.960/hr.	$691.20/mo.	15.8GB RAM / 620GB Disk	15GB RAM / 8 ECU / 1690GB Disk / m1.xlarge	$0.680/hr.	$489.60/mo.
			17.1GB RAM / 6.5 ECU / 420GB Disk / m2.xlarge	$0.500/hr.	$360.00/mo.
			34.2GB RAM / 13 ECU / 850GB Disk / m2.2xlarge	$1.000/hr.	$720.00/mo.
			68.4GB RAM / 26 ECU / 1690GB Disk / m2.4xlarge	$2.000/hr.	$1440.00/mo.
			1.7GB RAM / 5 ECU / 350GB Disk / c1.medium	$0.170/hr.	$122.40/mo.
			7GB RAM / 20 ECU / 1690GB Disk / c1.xlarge	$0.680/hr.	$489.60/mo.
			23GB RAM / 33.5 ECU / 1690GB Disk / cc1.4xlarge	$1.600/hr.	$1152.00/mo.
			22GB RAM / 33.5 ECU / 2 NVIDIA Tesla GPUS / 1690GB Disk / cg1.4xlarge	$2.100/hr.	$1512.00/mo.

Rackspace vs Amazon - illustrative pricing ref info

(network traffic *excluded* - costs extra..)

Considerations

Design

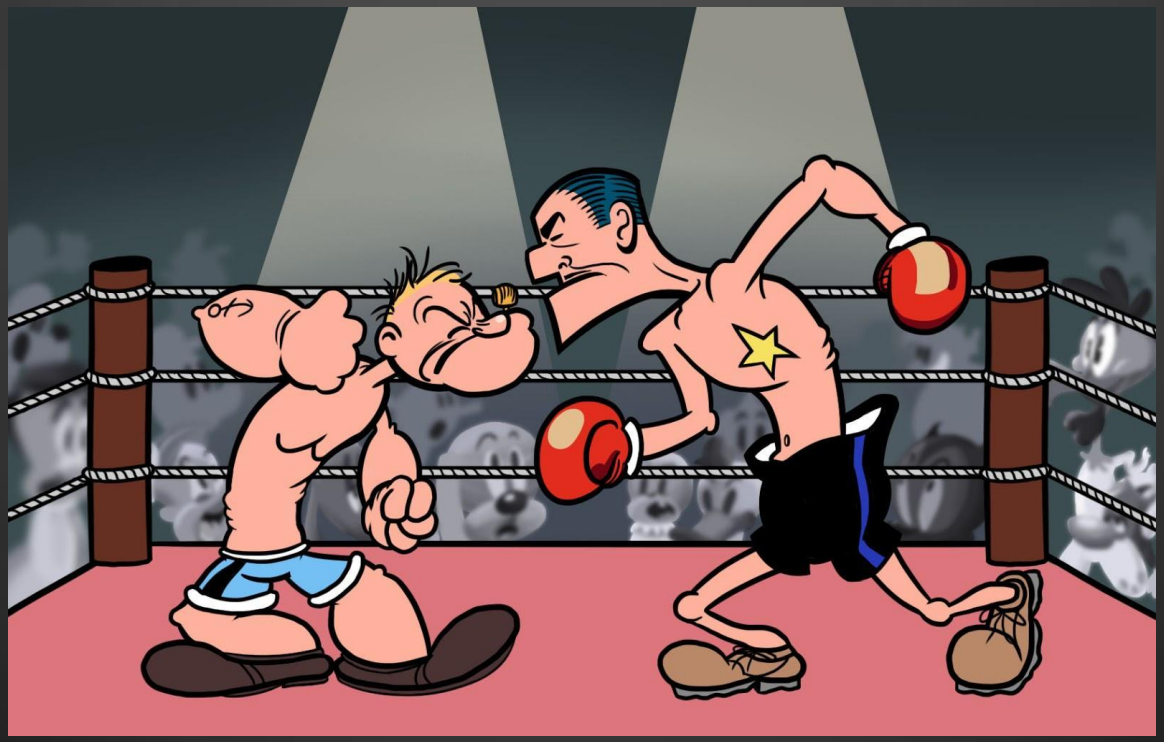
Targets

Planning

- plan early, stay focused
- proper sizing and focus is critical
- expect to build it all and throw it all away
- (likely; quite possibly a few times)

Best Pick?

Classic Architecture (Traditional / VMs) or
Dynamic Architecture (AAS / Abstracted) ?



Classic "Static" Architecture

- Workload is predictable, well characterized
  - (and potentially (HIGH:DEMANDING) i.e. drinks lots of cpu,ram,disk 'all the time')
- no _/hockeystick_/ curve
- lower costs easily possible, so long as 'fiddley management' is minimal

Dynamic "AAS" Architecture:

- Less characterized, or characterized to be less predictable
- suitable for self-manage api scaling?
- deploy:scale on demand (ie, don't just get bigger - get smaller - on demand)
- otherwise you pay the earth, miss all the benefits
  - (AAS morphs to "A Stupid Solution" or ASS)

Best Pick?

Catch 22?

- Can't know best pick until you know your needs
- Can't know your needs until you build something
- Building something takes non-trivial-effort
- ARRGH

Good tools FYI

Proxmox VE

- open source VM platform
- Linux containers, KVM full abstraction

Why Virtualize? -> Better to ask, Why not ?

(Topic for another day)

Monitor, monitor, monitor.

"It's 5 am, do you know what your services are doing?"

Cacti, Munin, Nagios, Zenoss, Spiceworks, whatever.

Thanks!

Your time and attention is appreciated!

Questions?

- Any questions?
  - Really, any questions ?

Proxmox Virtual Environment

PROXMOX Proxmox Virtual Environment
Version 2.0-18/16283a5a

You are logged in as 'root@pam' Logout Create VM Create CT

Server View

Datacenter

- Datacenter
  - proxmox-7-60
  - proxmox-7-61
  - proxmox-7-62

Search | Summary | Options | Storage | Backup | Users | Groups | Permissions | Roles | Authentication

Nodes

Name	ID	Online	Estranged	Server Address	Services
proxmox-7-61	1	Yes	No	192.168.7.61	PVECluster, RGManager
proxmox-7-62	2	Yes	No	192.168.7.62	PVECluster, RGManager
proxmox-7-60	3	Yes	No	192.168.7.60	PVECluster, RGManager

HA Service Status

Name	Owner	State	Restarts	Last transition	Last owner
pvevm:110	proxmox-7-62	started	0	Thu Dec 22 2011 18:43:34 GMT+010...	proxmox-7-60
pvevm:111	proxmox-7-62	started	0	Thu Dec 22 2011 15:33:34 GMT+010...	proxmox-7-61

Quorate: Yes

Tasks | Cluster log

Start Time	End Time	Node	User name	Description	Status
Dec 22 19:01:51	Dec 22 19:01:53	proxmox-7-60	root@pam	CT 104 - Start	OK