The Cloud Tutorial

Dan Reed, Roger Barga, Dennis Gannon
Microsoft Research
eXtreme Computing Group

Rich Wolski
Eucalyptus.com

Microsoft

Tutorial Outline

- Part 1. Introduction.
  - Basic concepts.
  - Data center and cloud architectures.
- Part 2. Building Infrastructure as a Service.
  - The Amazon EC2 and Eucalyptus model.
- Part 3. Programming Platforms and Applications.
  - The Azure platform.
    - Programming and data architecture.
  - Data analysis with MapReduce and more.
  - Application Examples.
- Part 4. More Programming Models & Services.
  - Google App Engine.
  - Cloudera, SalesForce and more
  - HPC and the Cloud

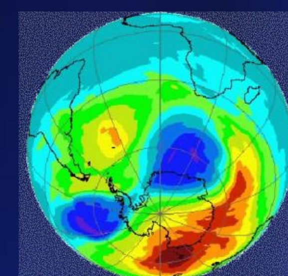
Part 1. Outline

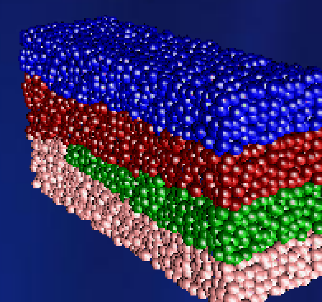
- Science in 2020
  - Our research challenges and impact of changing economics
  - A new architecture for scientific discovery
- Defining the Cloud
  - A scalable, persistent outsourced infrastructure
  - An framework for massive data analysis
  - An amplifier of our desktop experience
- The Origins
  - Modern data center architecture
- The Cloud Software Models
  - Infrastructure as a Service
  - Platform as a Service
  - Software as a Service

Science 2020

"In the last two decades advances in computing technology, from processing speed to network capacity and the Internet, have revolutionized the way scientists work.

From sequencing genomes to monitoring the Earth's climate, many recent scientific advances would not have been possible without a parallel increase in computing power - and with revolutionary technologies such as the quantum computer edging towards reality, *what will the relationship between computing and science bring us over the next 15 years?"*





http://research.microsoft.com/towards2020science

Microsoft

Sapir–Whorf: Context and Research

- Sapir–Whorf Hypothesis (SWH)
  - Language influences the habitual thought of its speakers
- Scientific computing analog
  - Available systems shape research agendas
- Consider some past examples
  - Cray-1 and vector computing
  - VAX 11/780 and UNIX
  - Workstations and Ethernet
  - PCs and web
  - Inexpensive clusters and Grids

- Today's examples
  - multicore, sensors, clouds and services …
- What lessons can we draw?

Microsoft

Our Decadal Research Changes

- Commodity clusters
  - Proliferation of inexpensive hardware
    - “Attack of the Killer Micros”
  - Race for MachoFLOPS
  - Low level programming challenges
- Rise of data
  - Scientific instruments and surveys
  - Storage, management and provenance
  - Data fusion and analysis
- Distributed services
  - Multidisciplinary collaborations
  - Interoperability and scalability
  - Multi-organizational social engineering

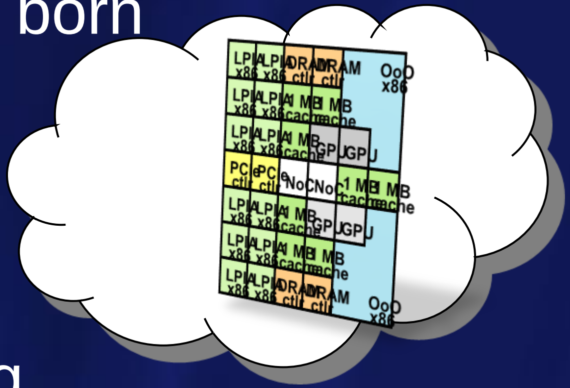
Today's Truisms (2009)

- Bulk computing is almost free
  - … but applications and power are not
- Inexpensive sensors are ubiquitous
  - … but data fusion remains difficult
- Moving lots of data is {still} hard
  - … because we're missing trans-terabit/second networks
- People are really expensive!
  - … and robust software remains extremely labor intensive
- Application challenges are increasingly complex
  - … and social engineering is not our forte
- Our political/technical approaches must change
  - … or we risk solving irrelevant problems

Microsoft

The Pull of Economics …

- Moore’s “Law” favored consumer commodities
  - Economics drove enormous improvements
  - Specialized processors and mainframes faltered
  - The commodity software industry was born



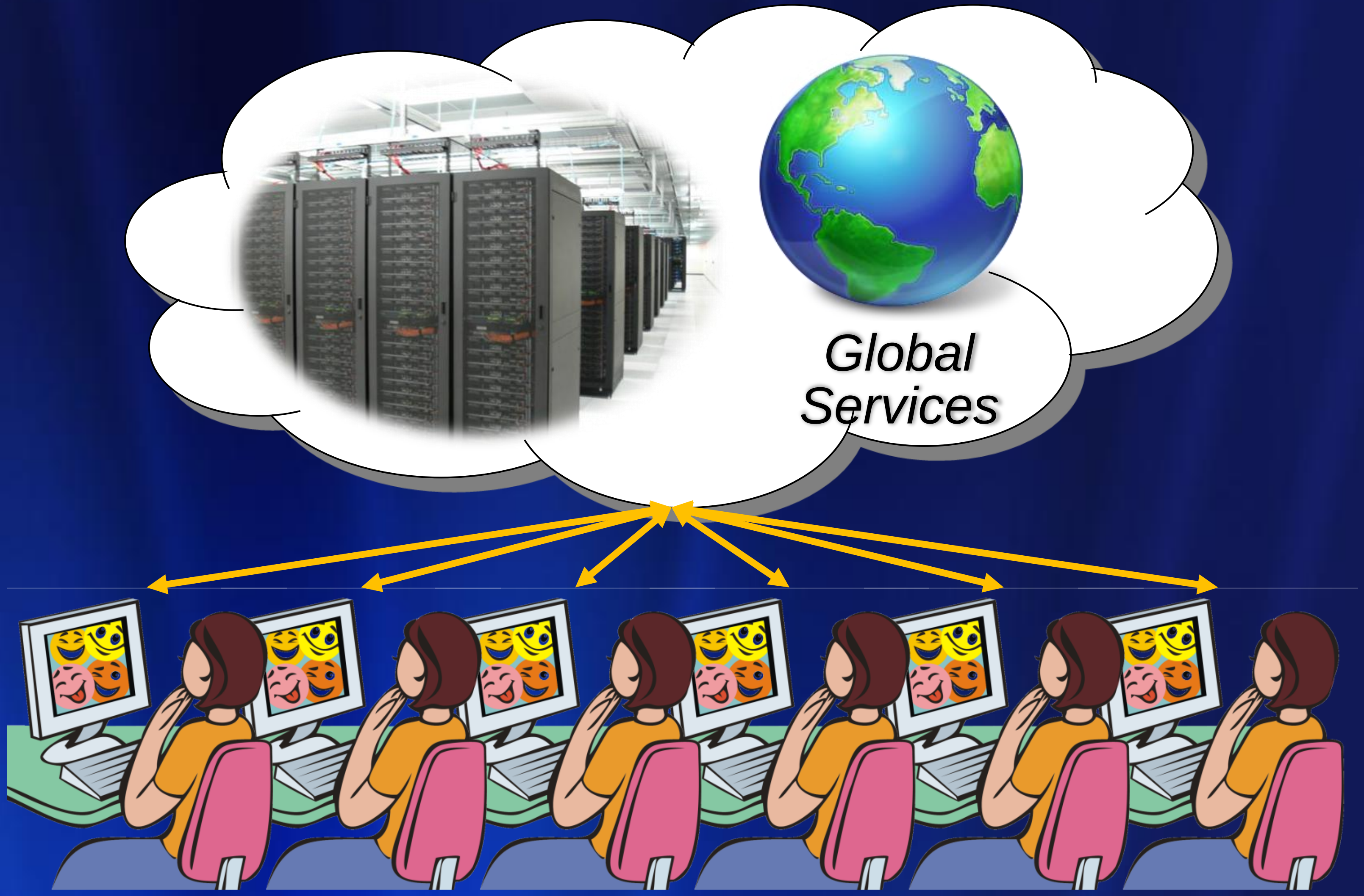
- Today’s economics
  - Manycore processors/accelerators
  - Software as a service/cloud computing
  - Multidisciplinary data analysis and fusion
- They is driving change in technical computing
  - Just as did “killer micros” and inexpensive clusters

Cloud Economics

- When applications are hosted
  - Even sequential ones are embarrassingly parallel
  - Few dependencies among users
- Moore's benefits accrue to platform owner
  - 2x processors →
    - ½ servers (+ ½ power, space, cooling …)
    - Or 2X service at the same cost
- Tradeoffs not entirely one-sided due to
  - Latency, bandwidth, privacy, off-line considerations
  - Capital investment, security, programming problems

Microsoft

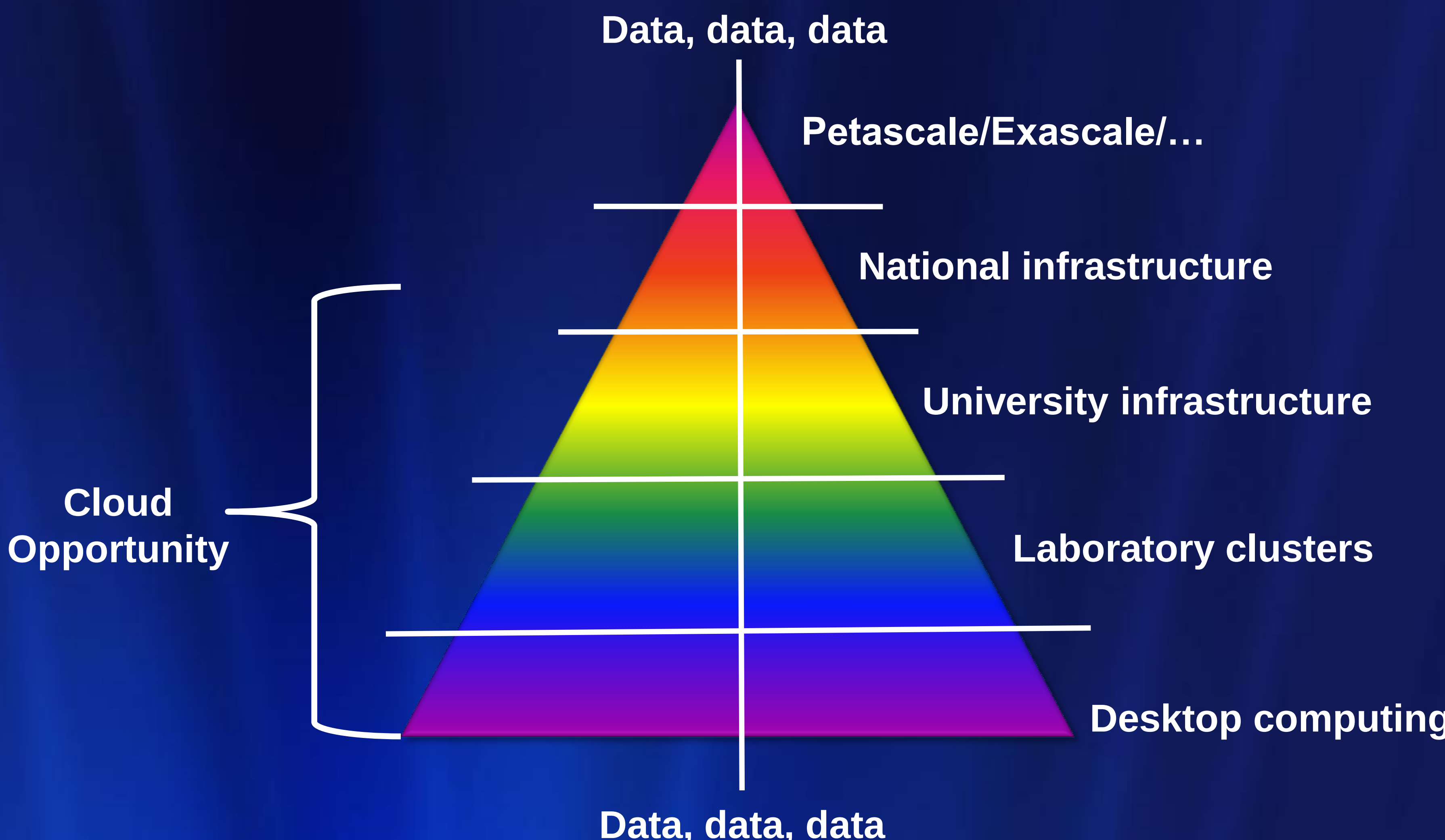
New Software Architecture



Insights: Not Just FLOPS Or Bytes

Software + Data + Services = Insights

The Computing Research Pyramid



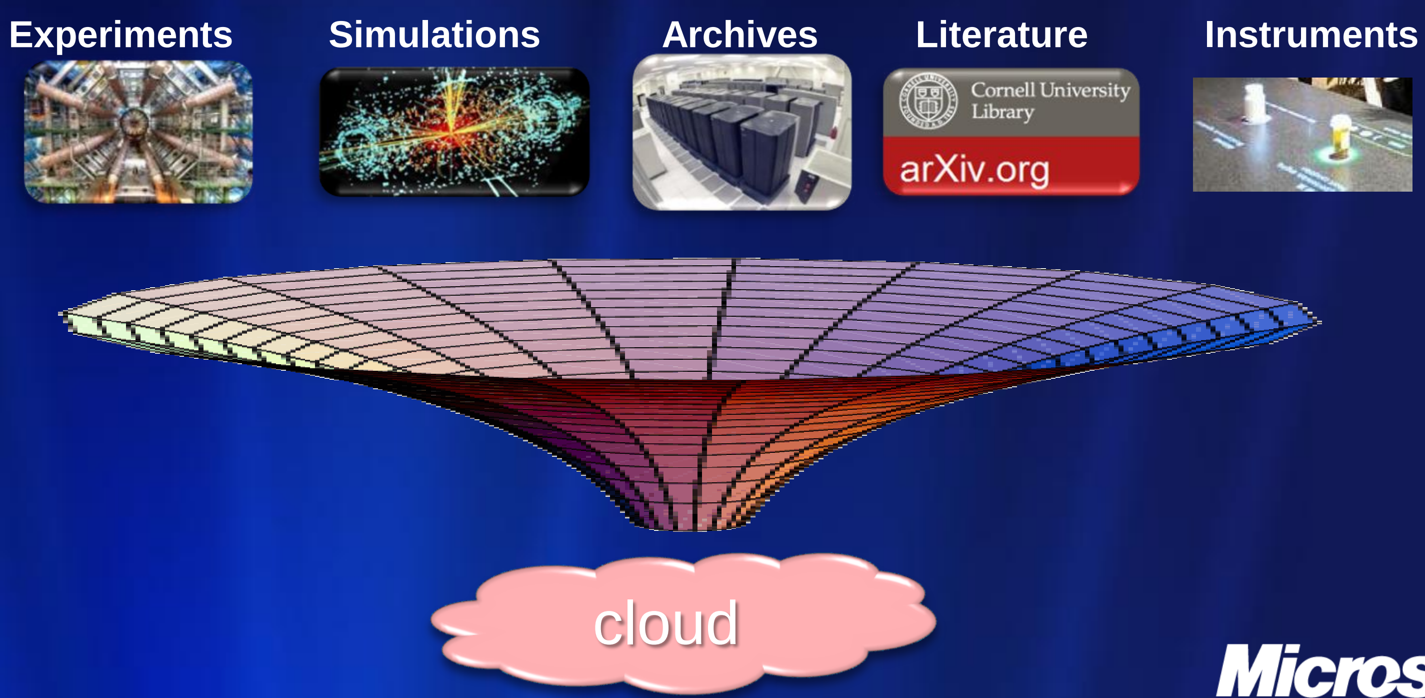
Defining the Cloud

- A model of computation and data storage based on “pay as you go” access to unlimited remote data center capabilities.
- A cloud infrastructure provides a framework to manage scalable, reliable, on-demand access to applications.
- Examples:
  - Search, email, social networks
  - File storage (Live Mesh, Mobile Me, Flicker, …)
- A way for a start-up to build a scalable web presence without purchasing hardware.

Microsoft

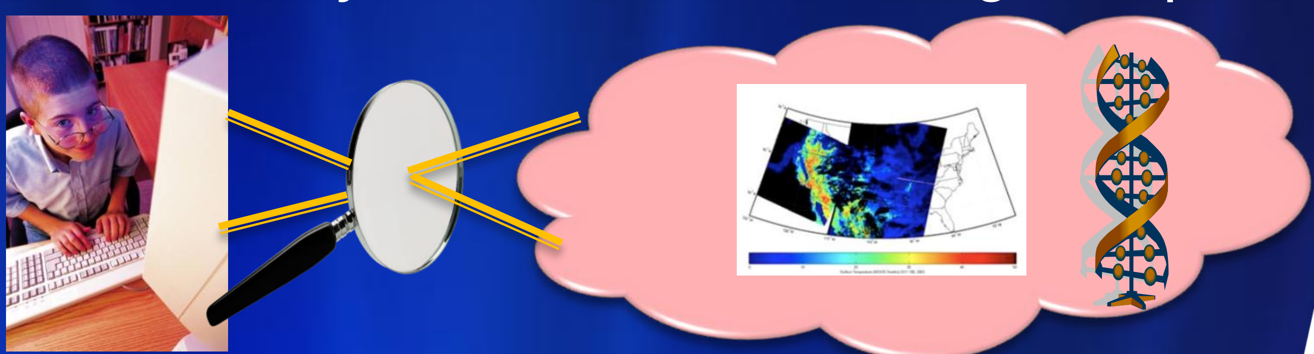
The Cloud as a Data Analysis Platform

- Deriving knowledge from vast data streams and online archives
  - Tools for massively parallel data reduction
  - Making the deep web searchable



The Cloud as an extension of your desktop and other client devices

- Today
  - Cloud storage for your data files synchronized across all your machines (mobile me, live mesh, flicker, etc.)
  - Your collaboration space (Sakai, SharePoint)
  - Cloud-enabled apps (Google Apps, Office Live)
- Tomorrow (or even sooner)
  - The lens that magnifies the power of desktop
  - Operate on a table with a billion rows in excel
  - Matlab analysis of a thousand images in parallel



Microsoft

The Clients+Cloud Platform

- At one time the "client" was a PC + browser.
- Now the cloud is an *integration point* for
  - The Phone
  - The laptop/tablet
  - The TV/Surface/Media wall
- And the future
  - The instrumented room
  - Aware and active surfaces
  - Voice and gesture recognition
  - Knowledge of where we are
  - Knowledge of our health

The Multi-Client Session

- Consider an application you open on one device.
  - You want to open a second device
  - And a third
- The state should be consistent across all the devices
- Replicate as much as possible on each device and in the cloud
- Update messages can maintain consistency.

Microsoft

The History of the Cloud

- In the beginning …
  - There was search, email, messaging, web hosting
- The challenge: How do you
  - Support email for 375 million users?
  - Store and index 6.75 trillion photos?
  - Support 10 billion web search queries/month?
  - Build an index for the entire web? And do it over and over again…
- And
  - deliver deliver a quality response in 0.15 seconds to millions of simultaneous users?
  - never go down.
- Solution: build big data centers

The Physical Architecture of Clouds

The contemporary data center

Microsoft

Clouds are built on Data Centers

- Range in size from "edge" facilities to megascale.
- Economies of scale
  - Approximate costs for a small size center (1000 servers) and a larger, 100K server center.

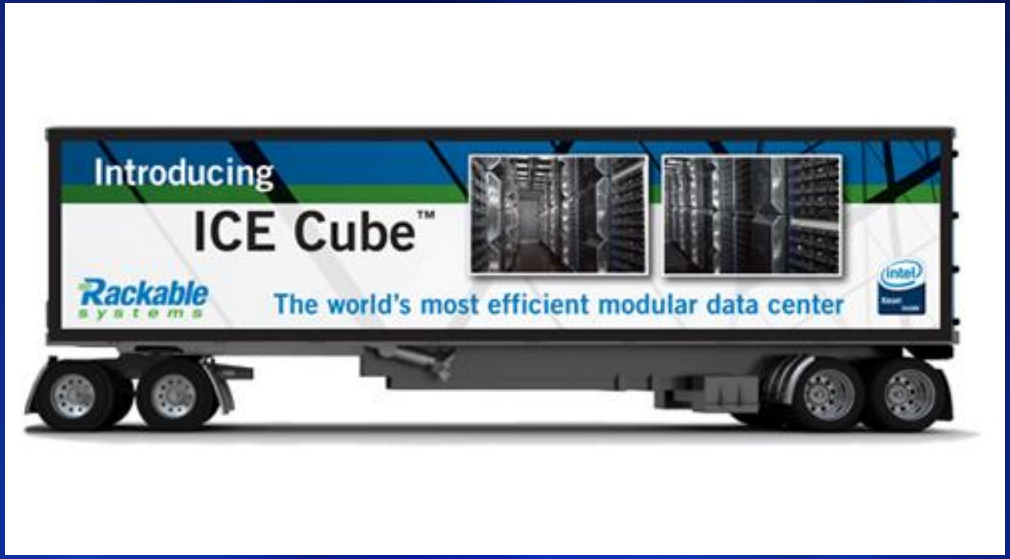
Technology	Cost in small-sized Data Center	Cost in Large Data Center	Ratio
Network	$95 per Mbps/ month	$13 per Mbps/ month	7.1
Storage	$2.20 per GB/ month	$0.40 per GB/ month	5.7
Administration	~140 servers/ Administrator	>1000 Servers/ Administrator	7.1

Each data center is
11.5 times
the size of a football field

Microsoft

Advances in DC deployment

- Conquering complexity.
  - Building racks of servers & complex cooling systems all separately is not efficient.
  - Package and deploy into bigger units:



Generation 4 data center video

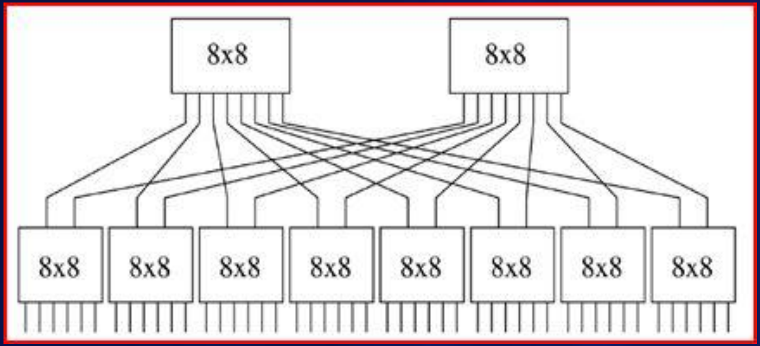
Microsoft

Containers: Separating Concers

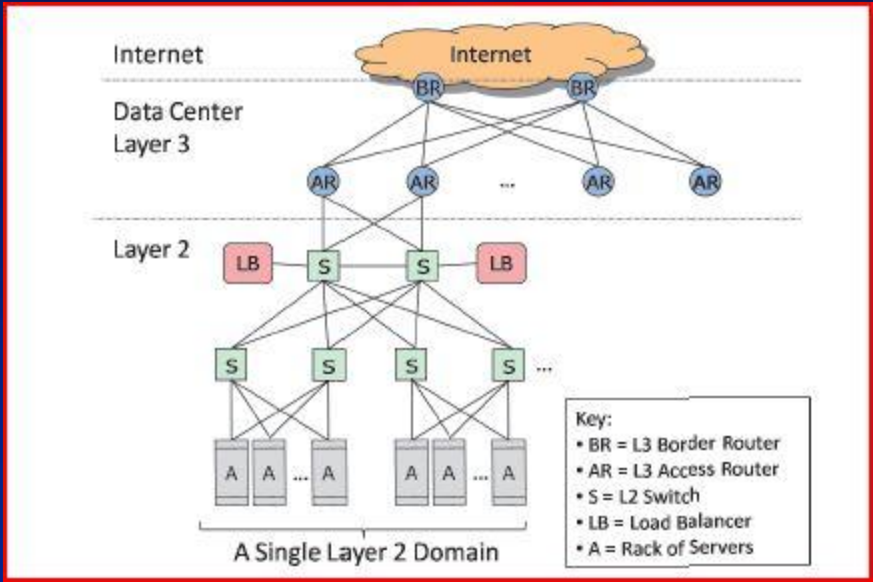
Data Center vs Supercomputers

- Scale
  - Blue Waters = 40K 8-core “servers”
  - Road Runner = 13K cell + 6K AMD servers
  - MS Chicago Data Center = 50 containers = 100K 8-core servers.
- Network Architecture
  - Supercomputers: CLOS “Fat Tree” infiniband
    - Low latency – high bandwidth
    - protocols
  - Data Center: IP based
    - Optimized for Internet Access
- Data Storage
  - Supers: separate data farm
    - GPFS or other parallel file system
  - DCs: use disk on node + memcache

Fat tree network

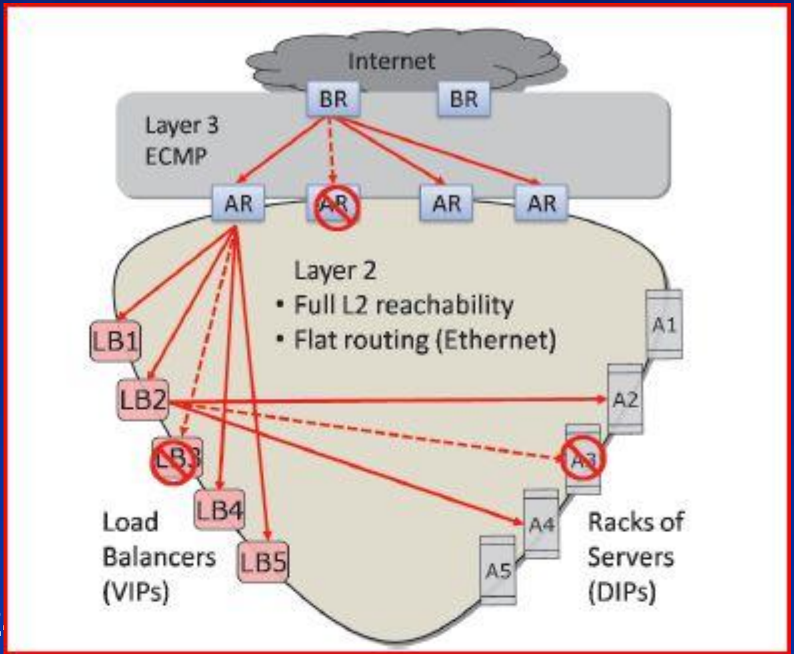


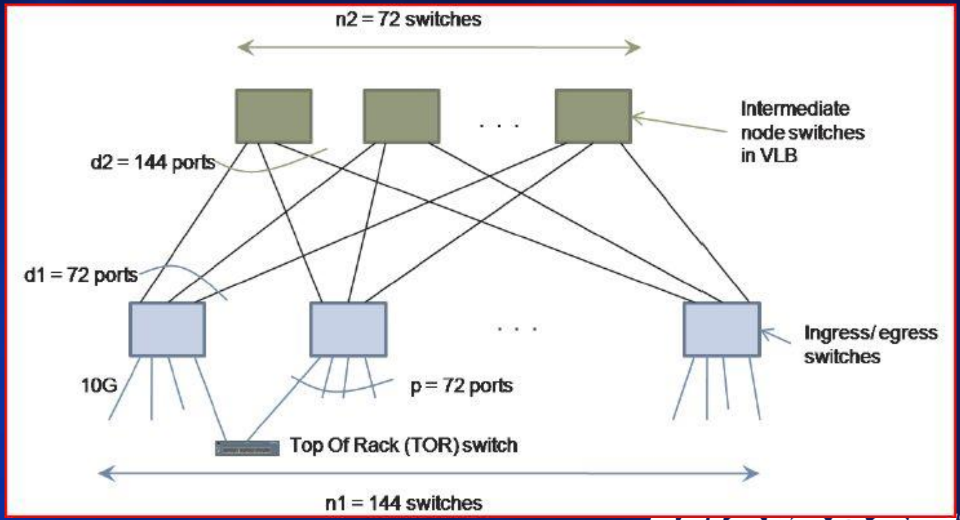
Standard Data Center Network



Next Gen Data Center Networks

- Monsoon
  - Work by Albert Greenberg, Parantap Lahiri, David A. Maltz, Parveen Patel, Sudipta Sengupta.
  - Designed to scale to 100K+ data centers.
  - Flat server address space instead of dozens of VLANS.
  - Valiant Load Balancing.
  - Allows a mix of apps and dynamic scaling.
  - Strong fault tolerance characteristics.





The Challenge of Data Centers & Apps

- The impact on the environment
  - In 2006 data centers used 61 *Tera*watt-hours of power
    - 1.5 to 3% of US electrical energy consumption today
  - Great advances are underway in power reduction
- With 100K+ servers and apps that must run 24x7 constant failure must be an axiom of hardware and software design.
  - Huge implication for the application design model.
  - How can hardware be designed to degrade gracefully?
- Two dimensions of parallelism
  - Scaling apps from 1 to 1,000,000 simultaneous users
  - Some apps require massive parallelism to satisfy a single request in less than a second.

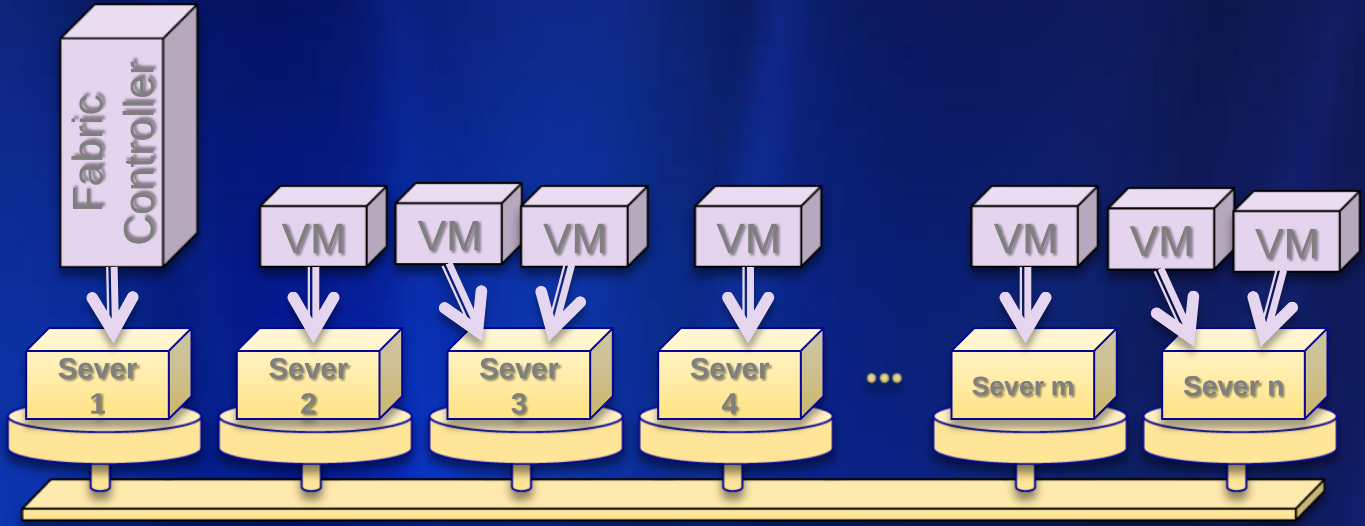
Cloud Software Models

Cloud Software Concepts

- The data center systems have a scale that makes failure a constant reality.
  - all data is replicated at least three times.
- Many applications are stateless.
  - Example: If a web search fails, user or system retries.
- Applications with state.
  - Divide computation into repeatable stateless transactions on saved state.
  - Each transaction must complete successfully before the state is modified. If a step fails, repeat it.
- Parallelism should always be dynamic
  - Elastic resource allocation to meet SLAs

Three Levels of Cloud Arcitecture

- Infrastructure as a Service (IaaS)
  - Provide App builders a way to configure a Virtual Machine and deploy one or more instances on the data center
  - Each VM has access to local and shared data storage
  - The VM has an IP Address visible to the world
  - A Fabric controller manages VM instances
    - Failure and restart, dynamic scale out and scale back.

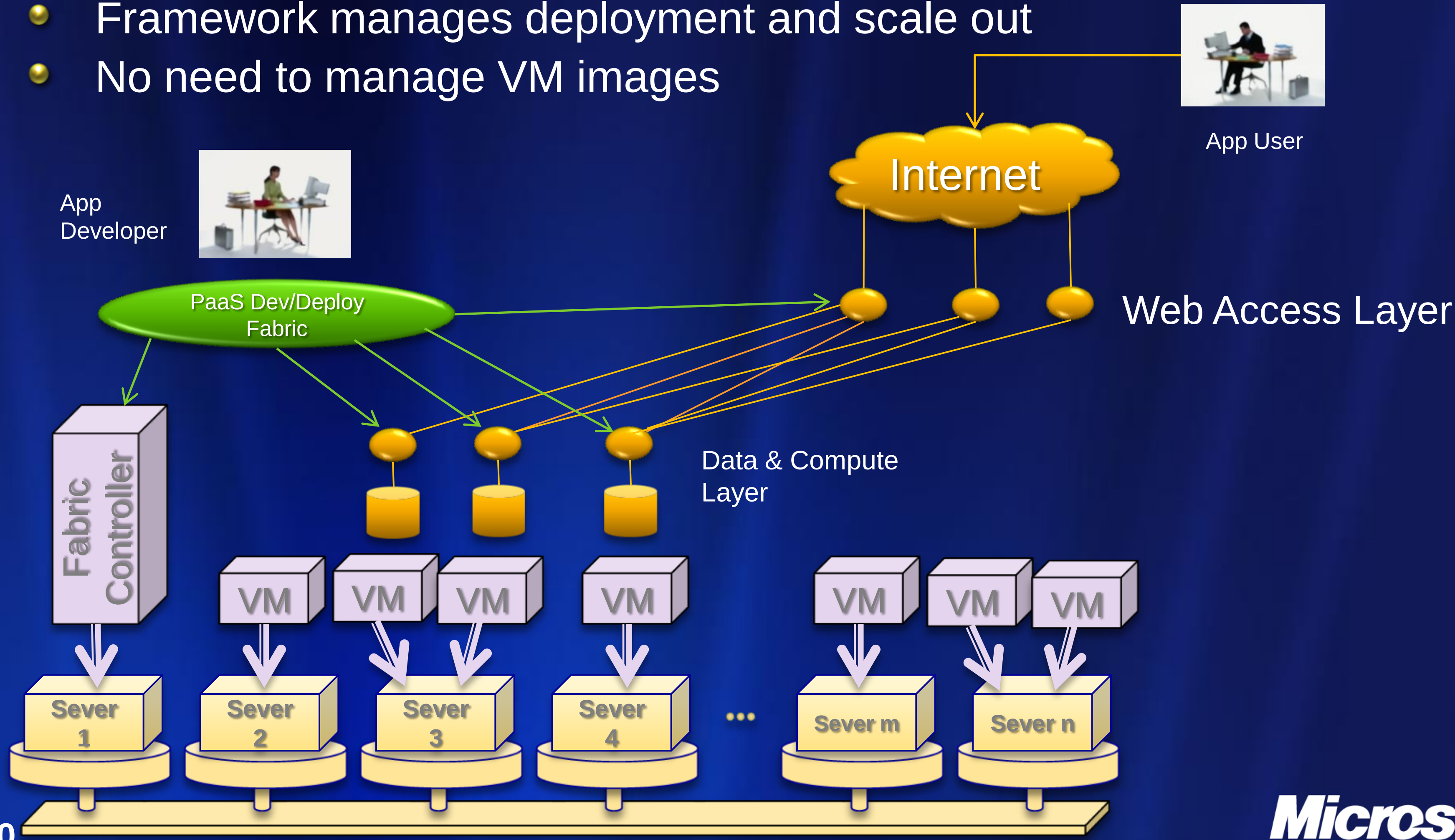


IaaS examples we will look at

- Eucalyptus.com
  - A software framework to support Amazon EC2 compatible services on private or public clusters
- Amazon EC2 + S3
  - The most widely known IaaS platform.
- Other IaaS platforms not described here
  - Flexiscale – UK based data centers
  - Rackspace – international data center hosting
  - GoGrid - cloud hosting division of ServePath
  - SliceHost –
  - Nimbus – Open Source EC2 from Argonne National Labs.

Platform as a Service

- An application development, deployment and management fabric.
- User programs web service front end and computational & Data Services
- Framework manages deployment and scale out
- No need to manage VM images



Microsoft

Sample PaaS platforms

- Microsoft Azure
  - Later in Tutorial
- Google App Engine
  - Later in Tutorial
- Others not covered in depth here
  - RightScale – cloud management via “cloud ready server templates”. Uses multiple IaaS providers.
  - SalesForce – Force: a cloud toolkit for CRM
  - Rollbase – customize prebuilt apps such as CRM
  - Bungee Connect – mashup cloud apps for CRM, etc.
  - Cloudera - Hadoop platform provider

Software as a Service

- Online delivery of applications
- Via Browser
  - Microsoft Office Live Workspace
  - Google Docs, etc.
  - File synchronization in the cloud – Live Mesh, Mobile Me
  - Social Networks, Photo sharing, Facebook, wikipedia etc.
- Via Rich Apps
  - Science tools with cloud back-ends
    - Matlab, Mathematica
  - Mapping
    - MS Virtual Earth, Google Earth
  - Much more to come.

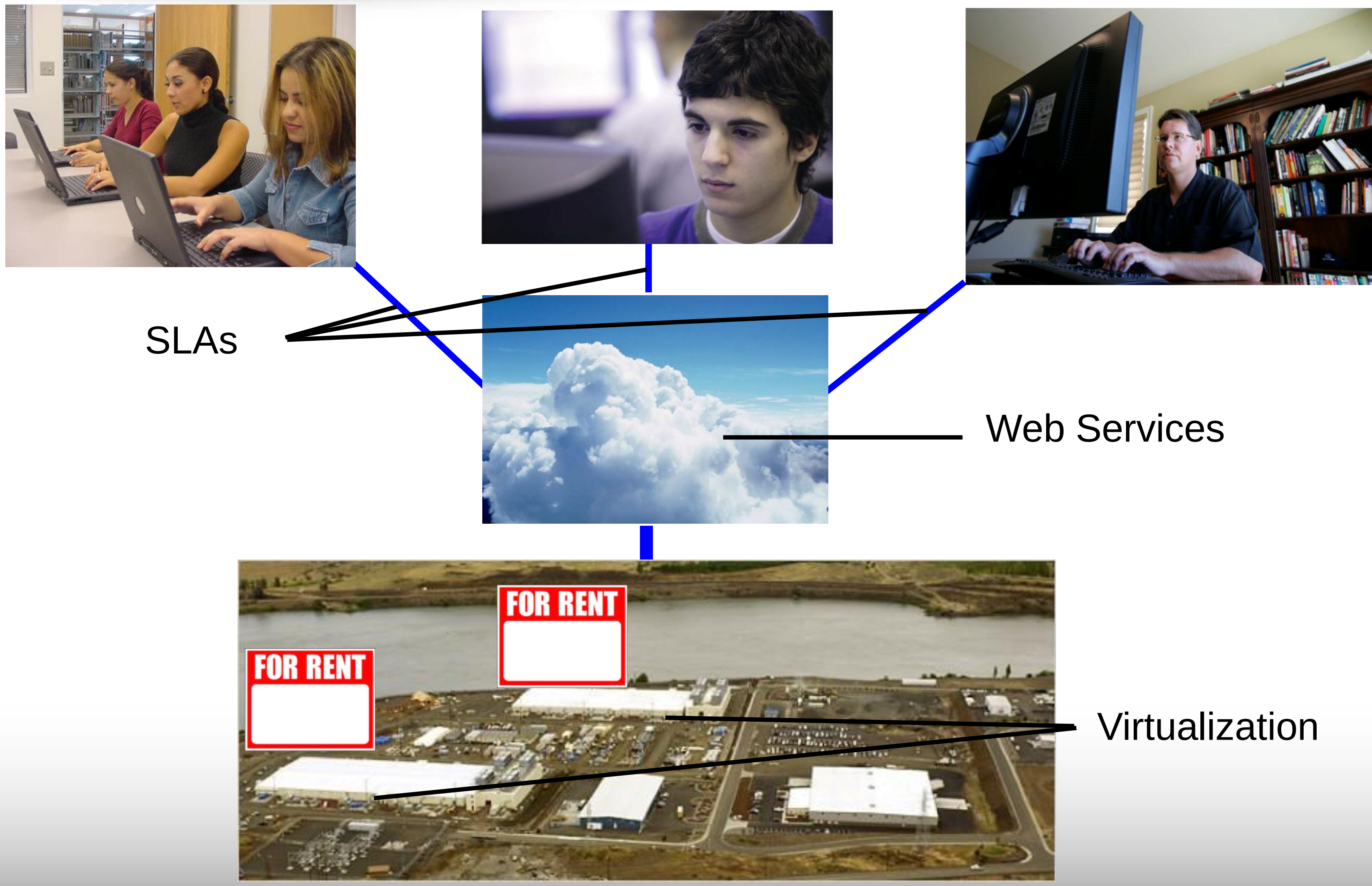
Others

- IaaS
  - Flexiscale – UK based data centers
  - Rackspace – international data center hosting
  - GoGrid - cloud hosting division of ServePath
  - SliceHost
- PaaS
  - RightScale – cloud management via “cloud ready server templates”. Uses multiple IaaS providers.
  - SalesForce – Force: a cloud toolkit for CRM
  - Rollbase – customize prebuilt apps such as CRM
  - Bungee Connect – mashup cloud apps for CRM, etc.
  - Cloudera - Hadoop platform provider.

Infrastructure as a Service: Seeing the (Amazon) Forest Through the (Eucalyptus) Trees

Rich Wolski
Eucalyptus Systems Inc.
www.eucalyptus.com

What is a cloud?



Public IaaS

Eucalyptus

- **Large scale infrastructure available on a rental basis**
  - Operating System virtualization (e.g. Xen, KVM) provides CPU isolation
  - "Roll-your-own" network provisioning provides network isolation
  - Locally specific storage abstractions
- **Fully customer self-service**
  - Customer-facing Service Level Agreements (SLAs) are advertized
  - Requests are accepted and resources granted via web services
  - Customers access resources remotely via the Internet
- **Accountability is e-commerce based**
  - Web-based transaction
  - "Pay-as-you-go" and flat-rate subscription
  - Customer service, refunds, etc.

Public, Private, and Premise

- **Public Cloud**
  - Large scale infrastructure available on a rental basis
  - Virtualized compute, network and storage
  - Underlying infrastructure is shared but tenants are isolated
  - Interface is transactional
  - Accounting is e-commerce based
- **Private Cloud**
  - Dedicated resources either as a rental or on-premise
- **On-premise Cloud**
  - Like public clouds but
    - Isolation must be controllable
    - Accounting is organizational

Amazon AWS

Eucalyptus

- **Compute**
  - Elastic Compute Cloud (EC2)
  - Virtual Machines for rent
- **Storage**
  - Simple Storage Service (S3) and Elastic Block Store (EBS)
  - Different levels of scalability

Cloud Platform

- **SimpleDB**
  - Attribute-value pair database
- **Simple Queue Service (SQS)**
  - Persistent message queues
- **Elastic MapReduce**
  - Hadoop
- **CloudFront**
  - Content distribution network

Cloud Services

EC2

- **Create and terminate virtual machines**
  - Create == provision and not boot
  - Terminate == destroy and not halt
- **Image**
  - initial root file system
- **Instance**
  - Image + kernel + ramdisk + ephemeral disk + private IP + public IP
- **Create an image: upload a root file system**
- **Run an instance: launch a VM with a specific**
  - Image that has been uploaded (into S3)
  - Kernel and ramdisk that Amazon provides
  - Ephemeral disk that gets created and attached

S3

- **Bucket store: buckets and objects**
  - Bucket: container for objects
  - Object: unit of storage/retrieval
  - Buckets are Created and Destroyed
  - Object are either Put or Get
- **Object storage is transactional**
  - Last write prevails
- **Eventually consistent**
  - Object writes will eventually be propagated
- **Buckets are access controlled**

EBS

- **Persistent Storage volumes that can be attached by VMs**
  - Raw block devices (must be formatted by owner/user)
  - Persist across VM creation and termination
  - Cannot be shared by multiple VMs simultaneously
  - Not accessible across “availability zones” (virtual data centers)
- **Persistent virtual local disk**

QoS and SLAs

Eucalyptus

- **Availability Zone: virtual data center**
  - Local area network performance within an availability zone
  - Wide area network performance between availability zones
  - Probability of simultaneous failure of multiple availability zones is very small
- **VM Type: minimum QoS for each VM**
  - EC2 Compute Unit: 1.0 to 1.2 GHz Xeon circa 2007
  - Small: 1 ECU, 1.7GB memory, 160GB ephemeral disk, 32 bit
  - Large: 4 ECU, 7.5GB memory, 850GB ephemeral disk, 64 bit
  - XL: 8 ECU, 15GB memory, 1690GB ephemeral disk, 64 bit

What does it look like?

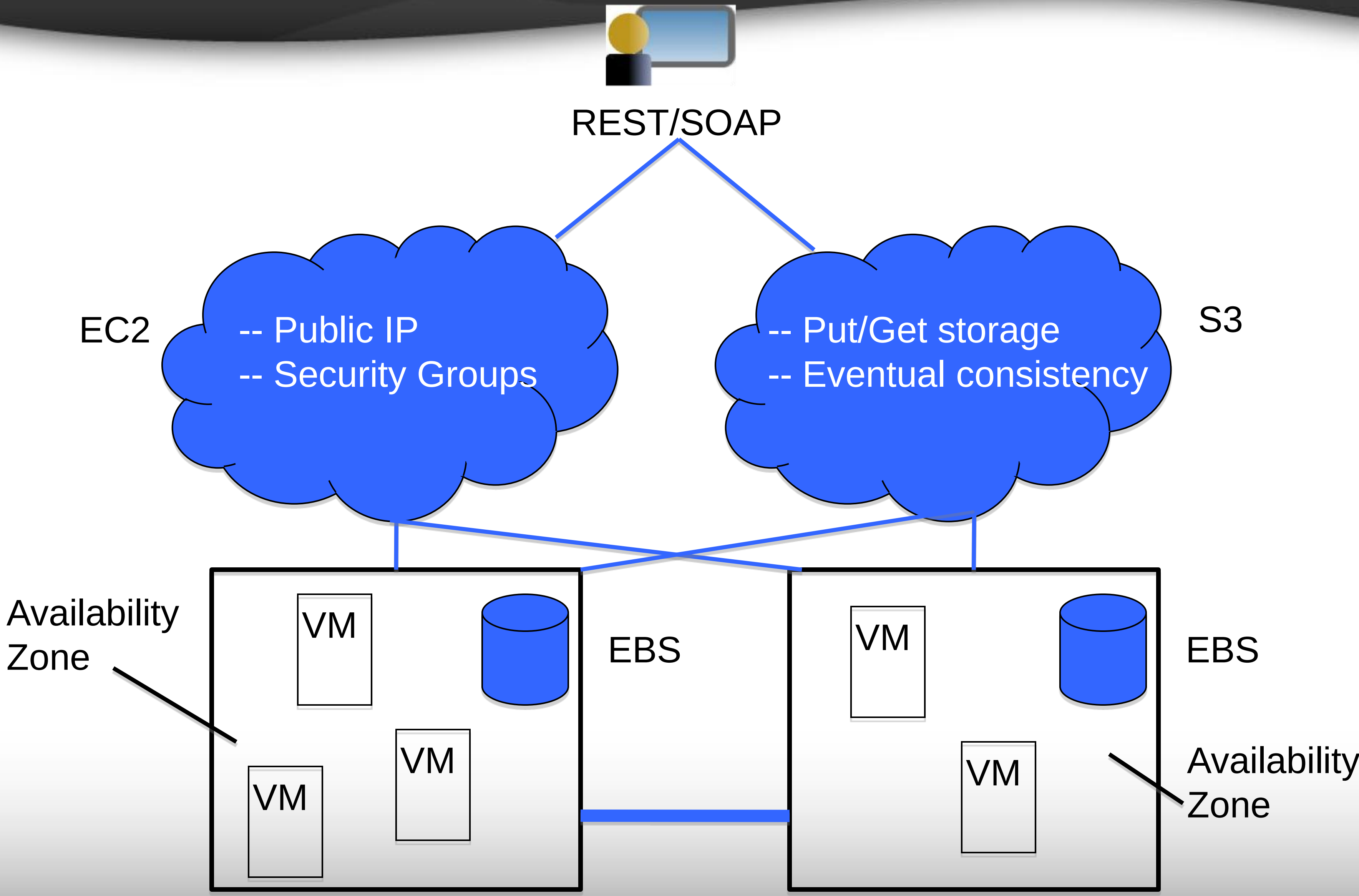
Eucalyptus

- **See the availability zones**
  - ec2-describe-availability-zones
- **Find an image**
  - ec2-describe-images -a
- **Create a key**
  - ec2-add-keypair mykey > mykey.private
- **Run an instance**
  - ec2-run-instances emi-E750108E -n 2 -k mykey
- **Create a volume**
  - ec2-create-volume --size 20 --availability-zone euca-1
- **Attach a volume**
  - ec2-attach-volume –i i-345E0661 –d /dev/sdc vol-2BD7043F

Charging

- **EC2 charging**
  - On-demand: per hour occupancy charge
  - VM type determines the rate
  - Per GB in and Out (not from AWS in same region)
- **S3 charging**
  - Per TB-month occupancy
  - Per GB in and Out (not from AWS in same region)
  - Per request
- **EBS charging**
  - Per GB-month of occupancy
  - Per million I/O requests
  - Per “snapshot” to S3

The Big Picture



Amazon and Eucalyptus

- **Public clouds are great but**
  - All data they process must “live” in the cloud
  - They are opaque
    - Compute, network, storage interaction is obscured
    - Data management is obscured
  - Accountability is e-commerce based
    - Is a refund really the best response to data loss or outage?
- **On-premise cloud**
  - Scale, self-service, and tenancy characteristics of public clouds
  - Transparency, data control, and accounting of on-premise IT
- **Eucalyptus: an open-source, on-premise cloud computing platform**

What's in a name?

Eucalyptus

- **Elastic Utility Computing Architecture Linking Your Programs To Useful Systems**
- **Web services based implementation of elastic/utility/cloud computing infrastructure**
  - Linux image hosting ala Amazon
- ***How do we know if it is a cloud?***
  - Try and emulate an existing cloud: Amazon AWS
- **Functions as a software overlay**
  - Existing installation should not be violated (too much)
- **Focus on portability, installation, and maintenance**
  - *"System Administrators are people too."*
- **Built entirely from open-source web-service (and related) technologies**

Open-source Cloud Infrastructure

Eucalyptus

- ***<u>Idea</u>:*** **Develop an open-source, freely available cloud platform for commodity hardware and software environments**
  - Stimulate interest and build community knowledge
  - Quickly identify useful innovations
  - Act to dampen the “hype”
- **Linux or Anti-Linux?**
  - Linux: open-source platform supporting all cloud applications changes the software stack in the data center
  - Anti-Linux: transparency of the platform makes it clear that clouds do not belong in the data center

Requirements for Open-source Cloud

Eucalyptus

- **Simple**
  - Must be transparent and easy to understand
- **Scalable**
  - Interesting effects are observed at scale (e.g. not an SDK)
- **Extensible**
  - Must promote experimentation
- **Non-invasive**
  - Must not violate local control policies
- **System Portable**
  - Must not mandate a system software stack change
- **Configurable**
  - Must be able to run in the maximal number of settings
- **Easy**
  - To distribute, install, secure, and maintain
- **Free**

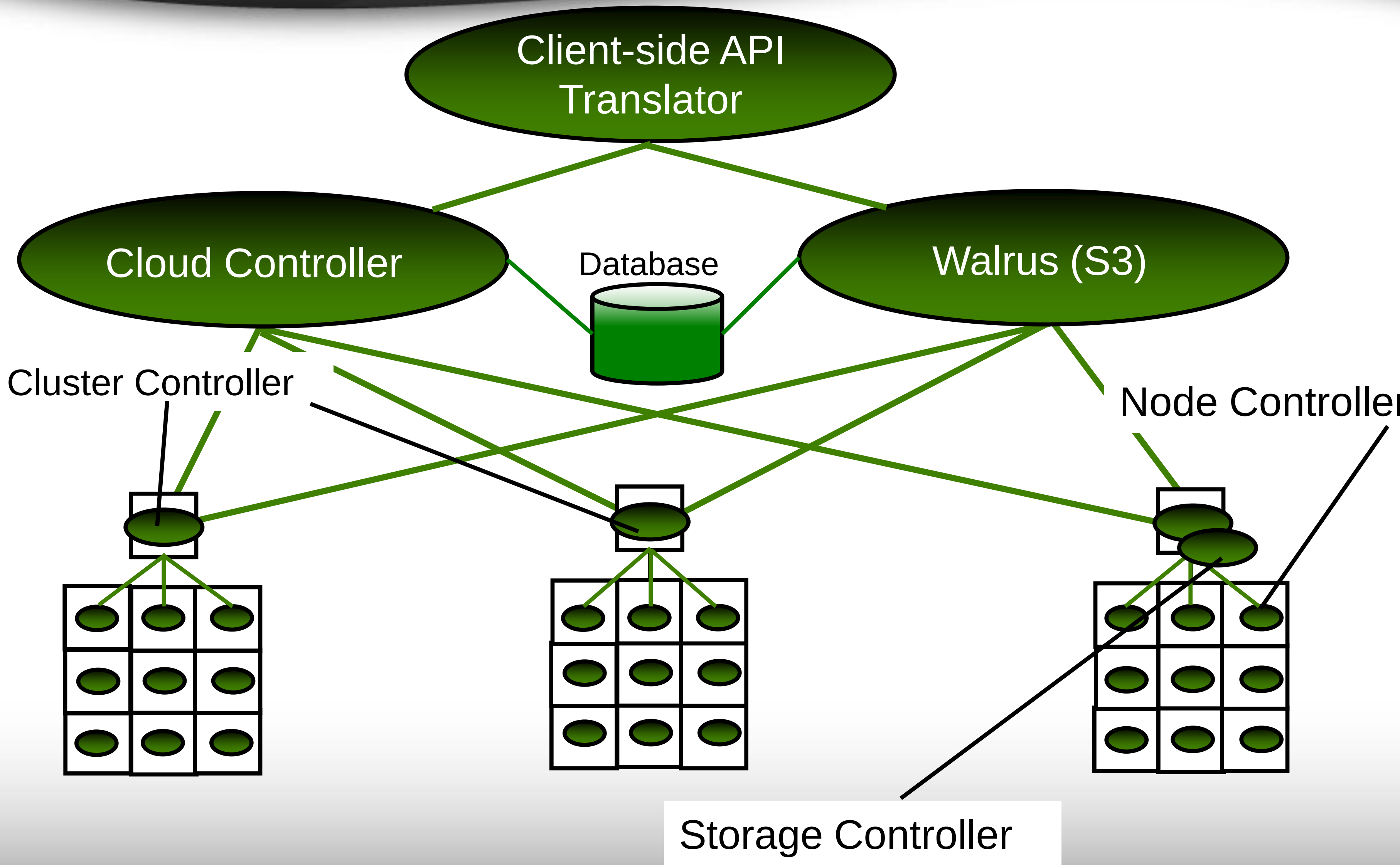
Open-source Eucalyptus

- **Is…**
  - Fostering greater understanding and uptake of cloud computing
  - Providing an experimentation vehicle prior to buying commercial cloud services
  - Homogenizing the local IT environment with Public Clouds (e.g. used as a hybrid cloud)
  - The cloud computing platform for the open source community
- **Is not…**
  - Designed as a replacement technology for AWS or any other Public Cloud service
- **AWS can't be downloaded as a Linux package**

Open-source Cloud Anatomy

- **Extensibility**
  - Simple architecture and open internal APIs
- **Client-side interface**
  - Amazon’s AWS interface and functionality (familiar and testable)
- **Networking**
  - Virtual private network per cloud
  - Must function as an overlay => cannot supplant local networking
- **Security**
  - Must be compatible with local security policies
- **Packaging, installation, maintenance**
  - system administration staff is an important constituency for uptake

Architecture



Notes from the Open-source Cloud

Eucalyptus

- **Private clouds and hybrid clouds**
  - Most users want private clouds to export the same APIs as the public clouds
- **In the Enterprise, the storage model is key**
  - Scalable “blob” storage doesn’t quite fit the notion of “data file.”
- **Cloud Federation is a policy mediation problem**
  - No good way to translate SLAs in a cloud allocation chain
  - “Cloud Bursting” will only work if SLAs are congruent
- **Customer SLAs allow applications to consider cost as first-class principle**
  - Buy the computational, network, and storage capabilities that are required

Cloud Mythologies

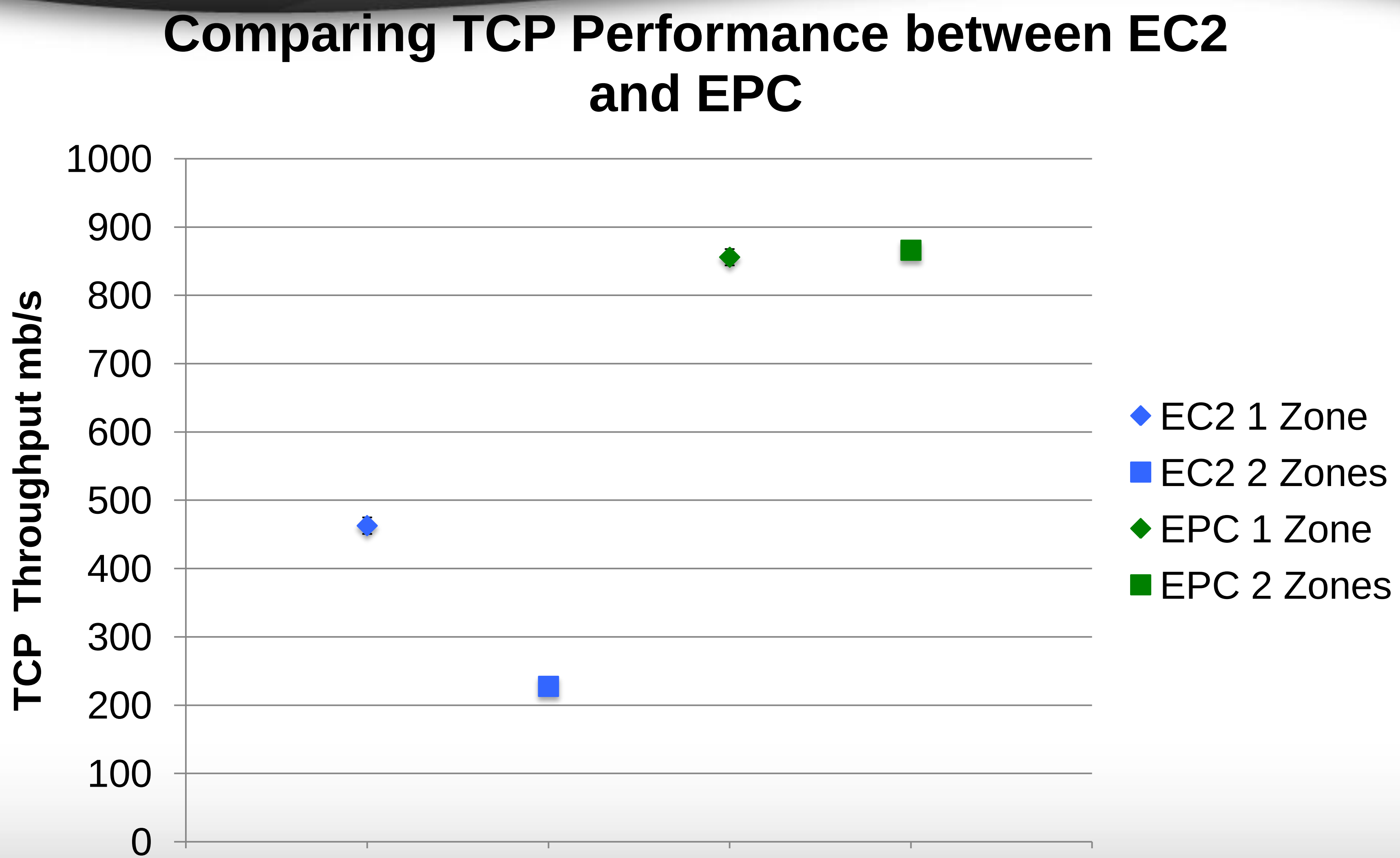
Eucalyptus

- **Cloud computing infrastructure is just a web service interface to operating system virtualization.**
  – "I'm running Xen in my data center – I'm running a private cloud."
- **Clouds and Grids are equivalent**
  – "In the mid 1990s, the term grid was coined to describe technologies that would allow consumers to obtain computing power on demand."
- **Cloud computing imposes a significant performance penalty over "bare metal" provisioning.**
  – "I won't be able to run a private cloud because my users will not tolerate the performance hit."

Cloud Speed

- **Extensive performance study using HPC applications and benchmarks**
- **Two questions:**
  - *What is the performance impact of virtualization?*
  - *What is the performance impact of cloud infrastructure?*
- **Tested Xen, Eucalyptus, and AWS (small SLA)**
- **Many answers:**
  - Random access disk is slower with Xen
  - CPU bound can be *faster* with Xen -> depends on configuration
  - Kernel version is far more important
  - Eucalyptus imposes no statistically detectable overhead
  - AWS small appears to throttle network bandwidth and (maybe) disk bandwidth -> *$0.10 / CPU hour*

Performance Comparison



Open-source Distribution

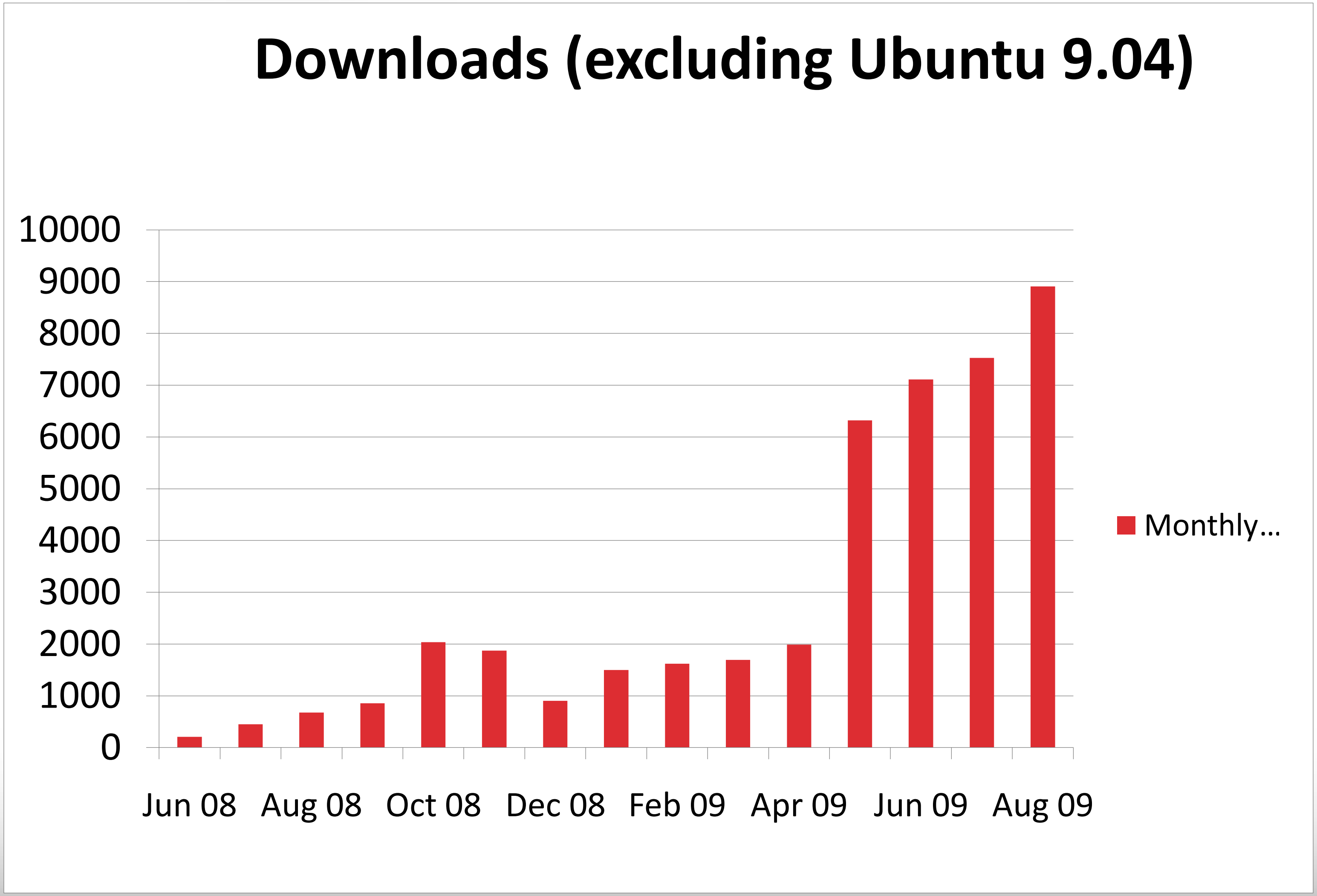
Eucalyptus

Via Linux: *Ubuntu* and Eucalyptus

- Jaunty Jackalope "Powered by Eucalyptus"
  - April 23, 2009
  - Complete build-from-source
- Karmic Koala
  - October 23, 2009
  - Full-featured Eucalyptus
- Fundamental technology
  - "Ubuntu Enterprise Cloud" ecosystem surrounding Eucalyptus
- 10,000,000 potential downloads
- *Debian* "squeeze"
  - Source release packaging under way
- Packaged for *CentOS*, *OpenSUSE*, *Debian*, and *Ubuntu* as "binary" release as well

> "Make Eucalyptus the open source reference implementation for cloud computing."
>
> Simon Wardley (head of cloud strategy), Canonical

50K Downloads (so far)



No Eucalyptus in Antarctica (yet)

Open-source Roadmap

- **5/28/08 – Release 1.0 shipped**
- **8/28/08 – EC2 API and initial installation model in V1.3**
  - Completes overlay version
- **12/16/08 – Security groups, Elastic IPs, AMI, S3 in V1.4**
- **4/19/09 – EBS, Metadata service in V1.5.1**
- **4/23/09 - Ubuntu release**
- **4/27/09 – www.eucalyptus.com**
- **7/17/09 – Bug fix release in V1.5.2**
  - First open-source release from ESI
- **10/23/09 – Karmic Koala release**
  - 10^7 downloads from “main” archive
- **11/1/09 – Final feature release as V1.6**
  - Completes AWS specification as of 1/1/2009
- **1/1/10 – release V1.7**

Eucalyptus is a Team Sport

Eucalyptus

- **Thanks to our original research sponsors…**

UCSB

VGrADS

NSF

- **…and to our new commercial friends**

BENCHMARK CAPITAL

BV CAPITAL SILICON VALLEY EUROPE

www.eucalyptus.com
805-845-8000
rich@eucalyptus.com

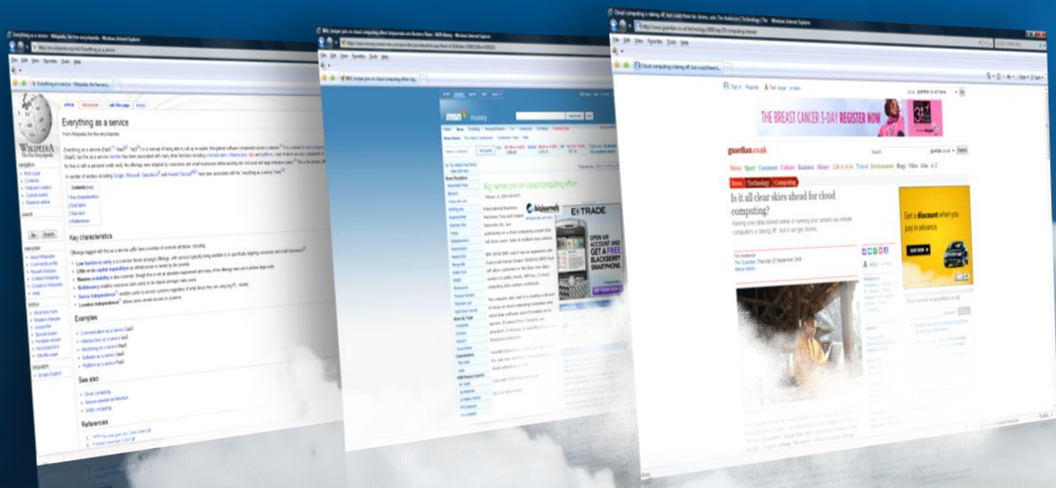
Platform as a Service

Windows Azure
Dryad & DryadLINQ

Roger Barga
Architect, Cloud Computing Futures Group
Microsoft Research (MSR)

SC Portland, OR 2009
Computing For A Changing World.
November 14-20, 2009
Oregon Convention Center
Portland, Oregon

PaaS – What is a "cloud platform"?



"…data as a service…"

"cloud computing journal reports that…"

"…software as a service…"

"…everything as a service..."

Platforms succeed when the platform helps others succeed

Platform Extension to Cloud is a Continuum

Less Constrained **Constraints in the App Model** *More Constrained*

Amazon AWS
VMs Look Like Hardware
No Limit on App Model
User Must Implement Scalability and Failover

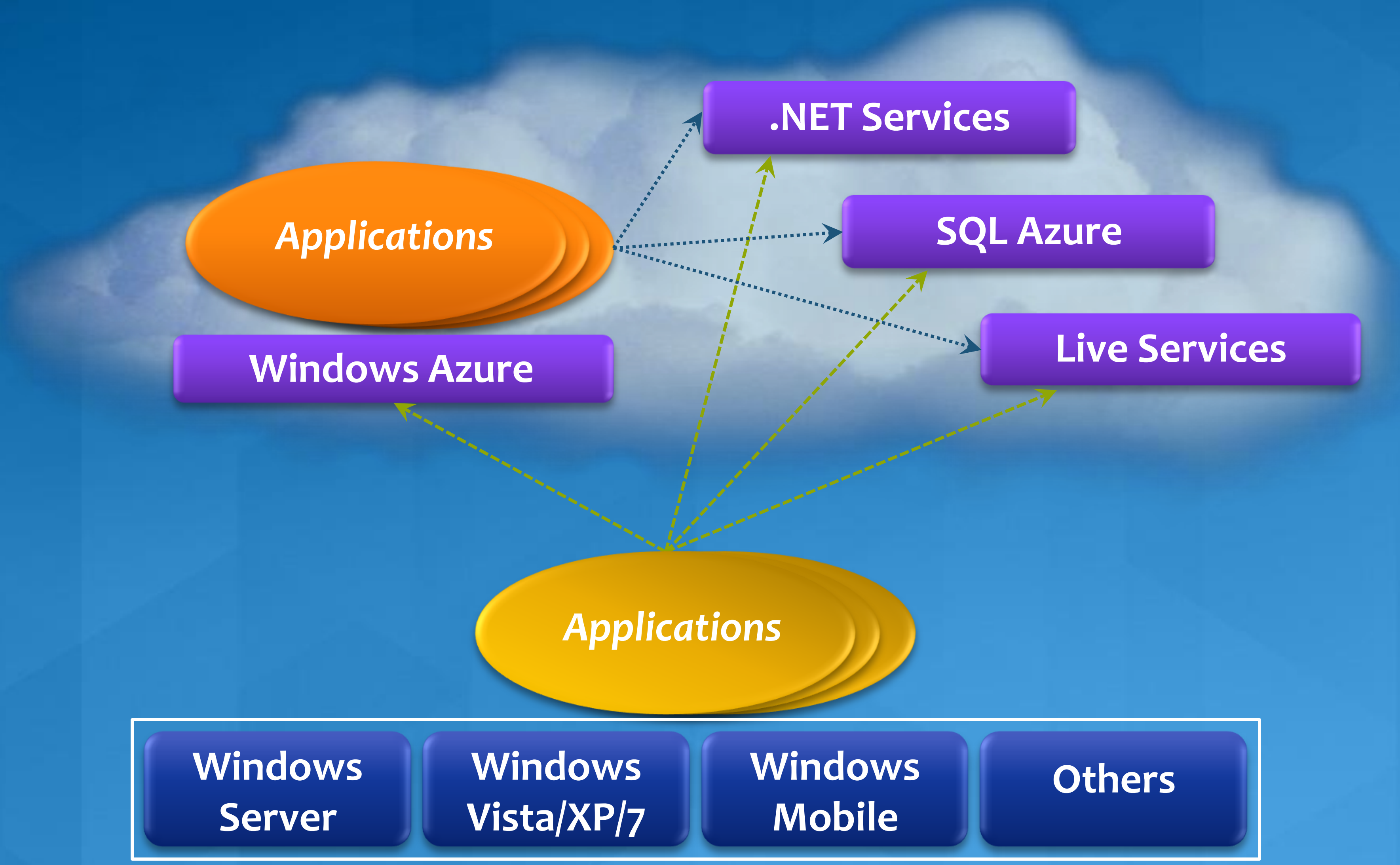
Microsoft Azure
.NET CLR/Windows Only
Choice of Language
Some Auto Failover/ Scale (but needs declarative application properties)

Google App Engine
Traditional Web Apps
Auto Scaling and Provisioning

Force.Com
SalesForce Biz Apps
Auto Scaling and Provisioning

Less Automation **Automated Management Services** *More Automation*

The Windows Azure Platform



Windows Azure Basics

The goal of Windows Azure is to provide a platform that is *scalable* and *available*

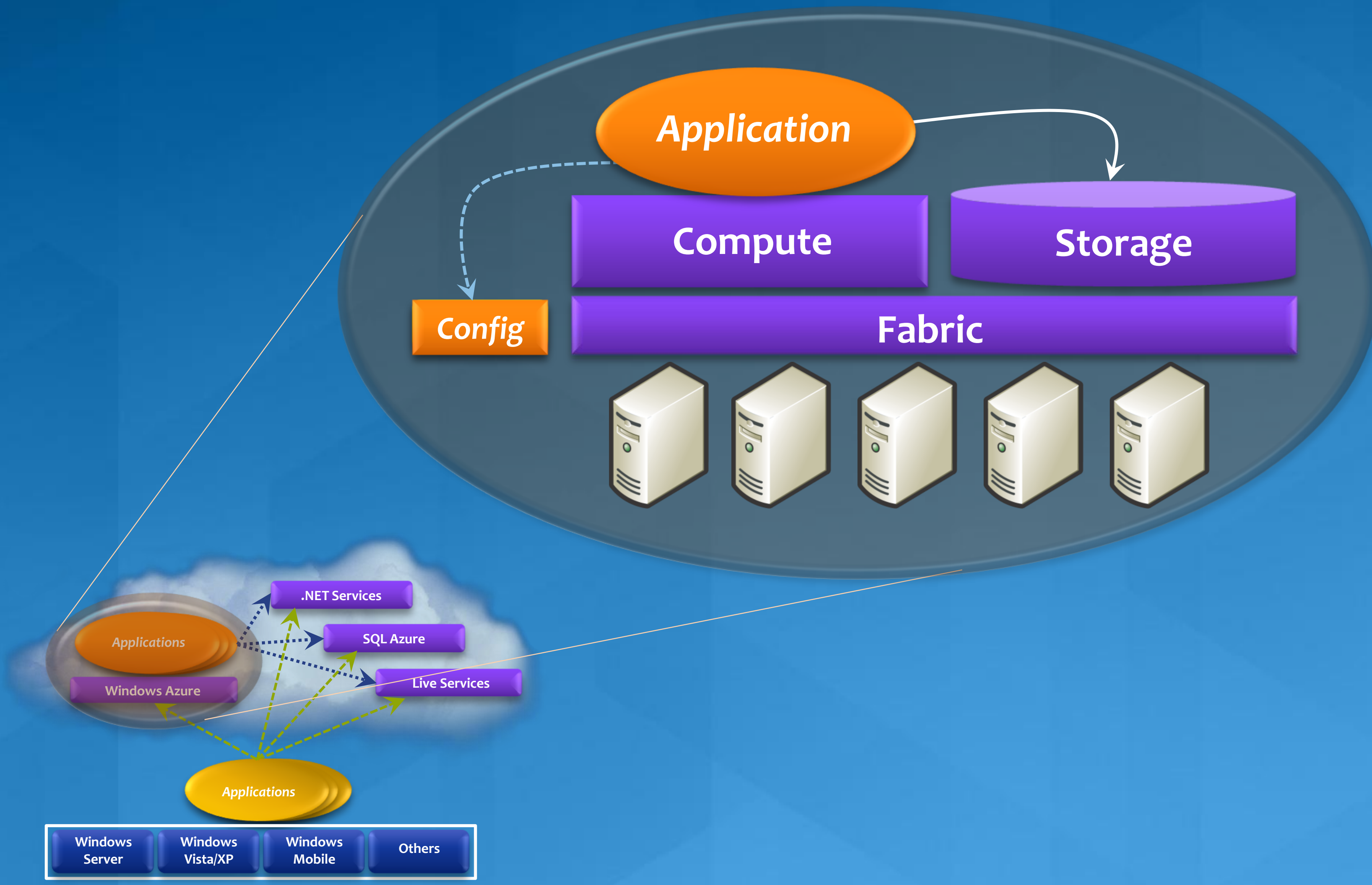
- Services are always running, rolling upgrades/downgrades
- Failure of any node is expected, state has to be replicated
- Services can grow very large, requires careful state management at scale
- Handle dynamic configuration changes due to load or failure

Windows Azure can run various kinds of Windows applications:

- .NET applications
- Unmanaged code
- PHP
- . . .

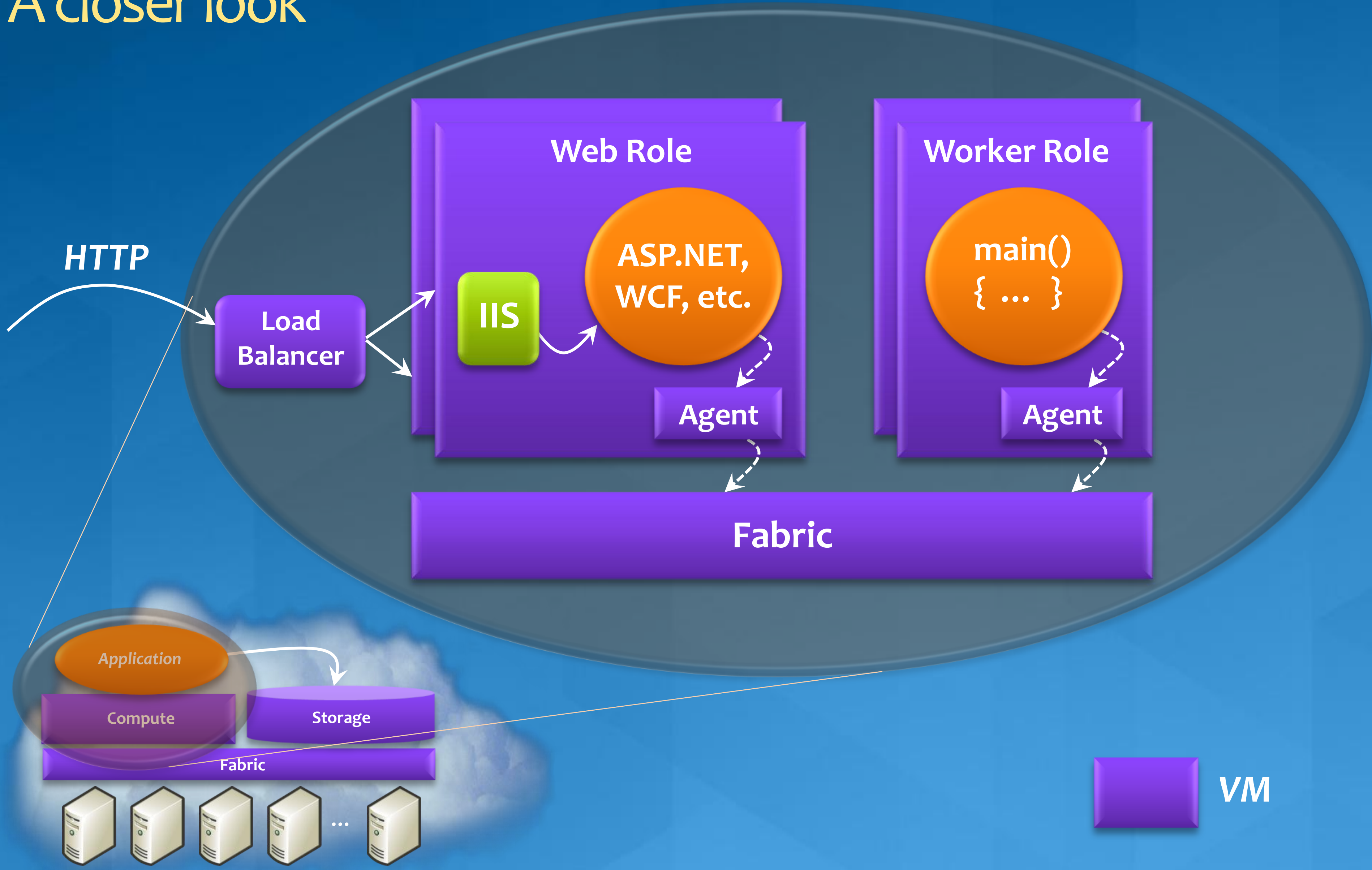
Windows Azure

An illustration



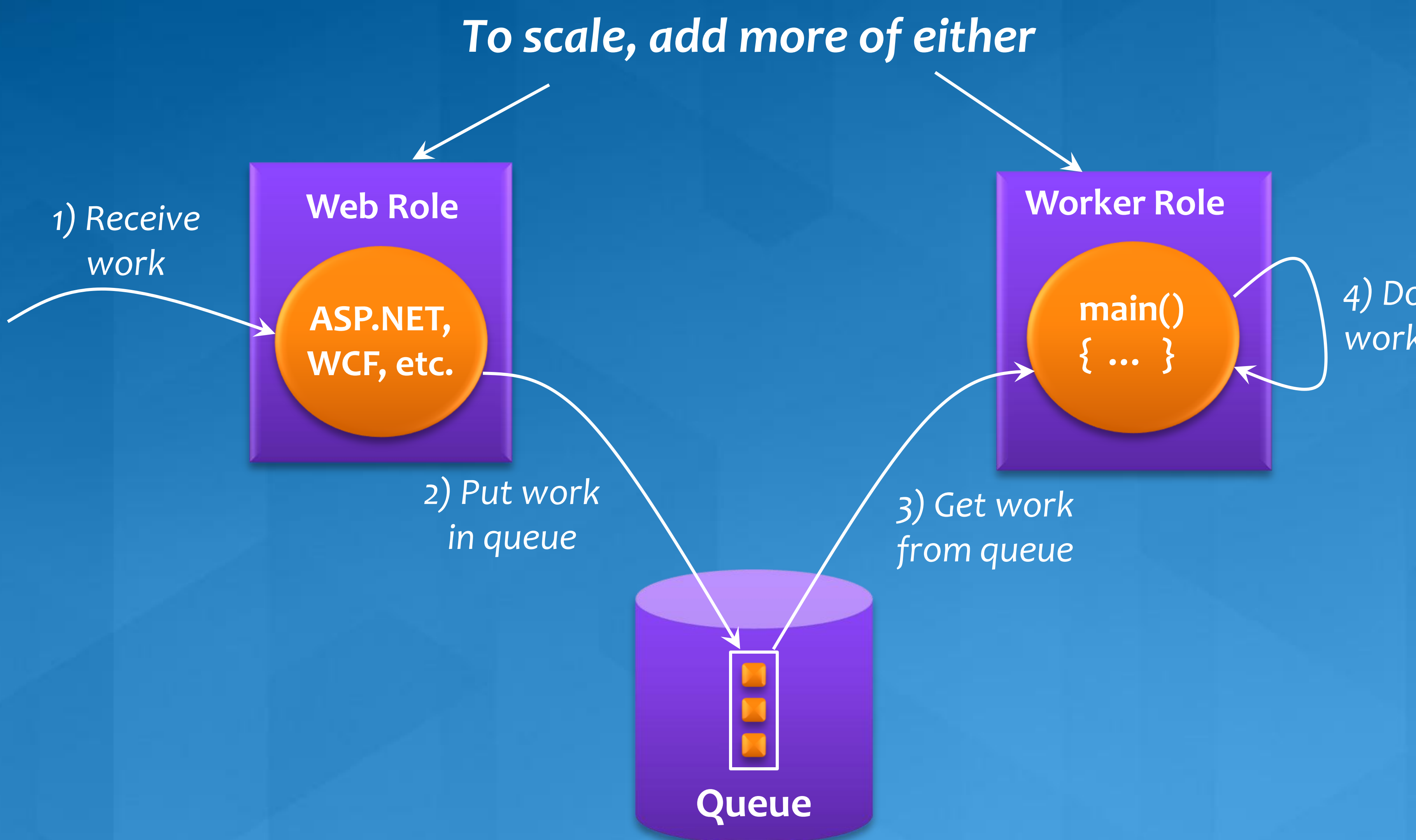
Windows Azure Compute Service

A closer look



The Suggested Application Model

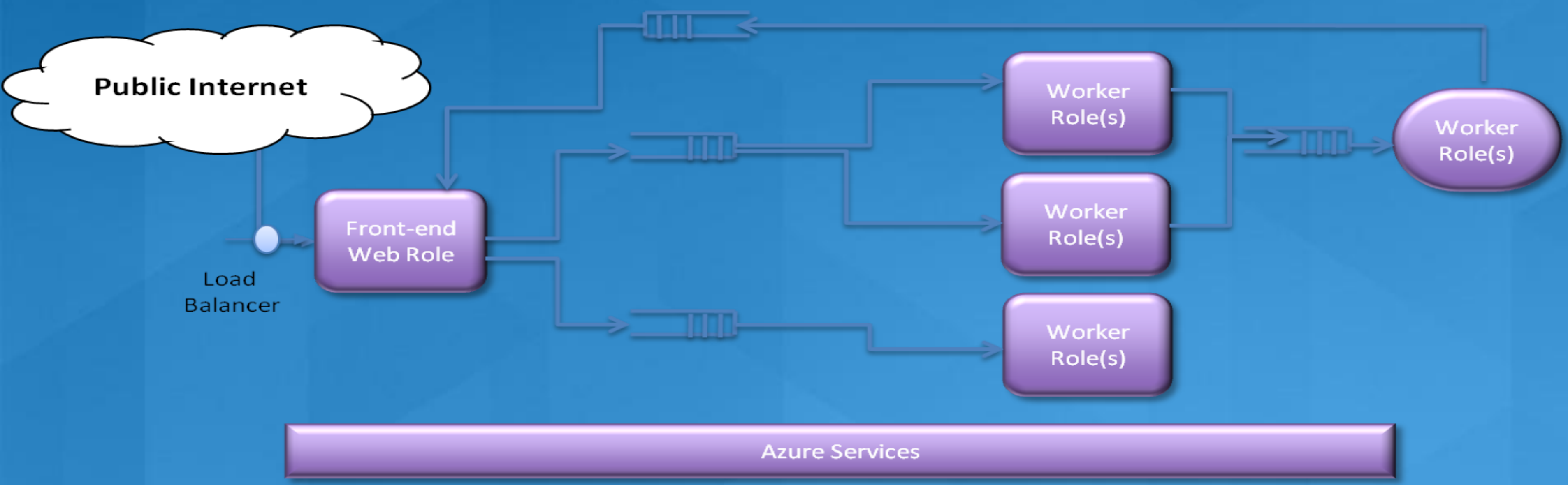
Using queues



Scalable, Fault Tolerant Applications on Azure

Queues are the application glue

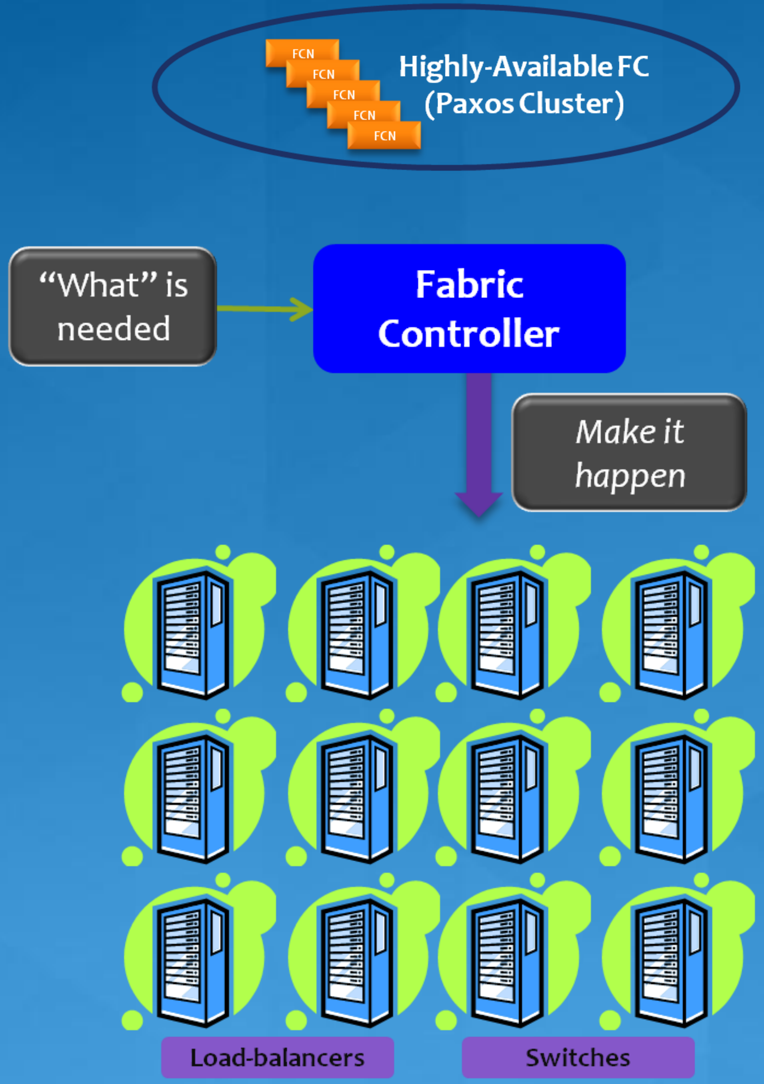
- Queues decouple different parts of application, making it easier to scale app parts independently;
- Flexible resource allocation, different priority queues and separation of backend servers to process different queues.
- Queues mask faults in worker roles.



Windows Azure Compute Fabric

Fabric Controller

- Owns all data center hardware
- Uses inventory to host services
- Deploys applications to free resources
- Maintains the health of those applications
- Maintains health of hardware
- Manages the service life cycle starting from bare metal



Windows Azure Compute Fabric

Fault Domains

Purpose: Avoid single points of failures

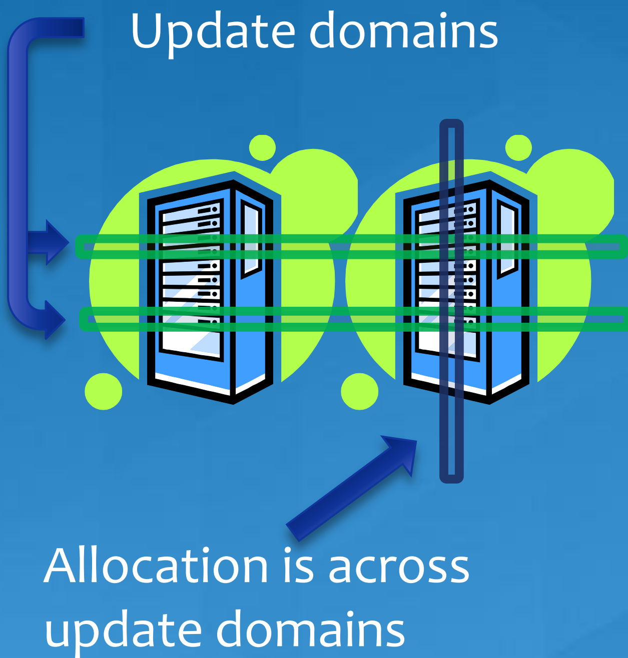
- Unit of a failure
  - Examples: Compute node, a rack of machines
- System considers fault domains when allocating service roles
- Service owner assigns number required by each role
  - Example: 10 front-ends, across 2 fault domains

Windows Azure Compute Fabric

Update Domains

Purpose: ensure the service stays up while undergoing an update

- Unit of software/configuration update
  - Example: set of nodes to update
- Used when rolling forward or backward
- Developer assigns number required by each role
  - Example: 10 front-ends, across 5 update domains



Windows Azure Compute Fabric

Push-button Deployment

Step 1: Allocate nodes

- Across fault domains
- Across update domains

Step 2: Place OS and role images on nodes

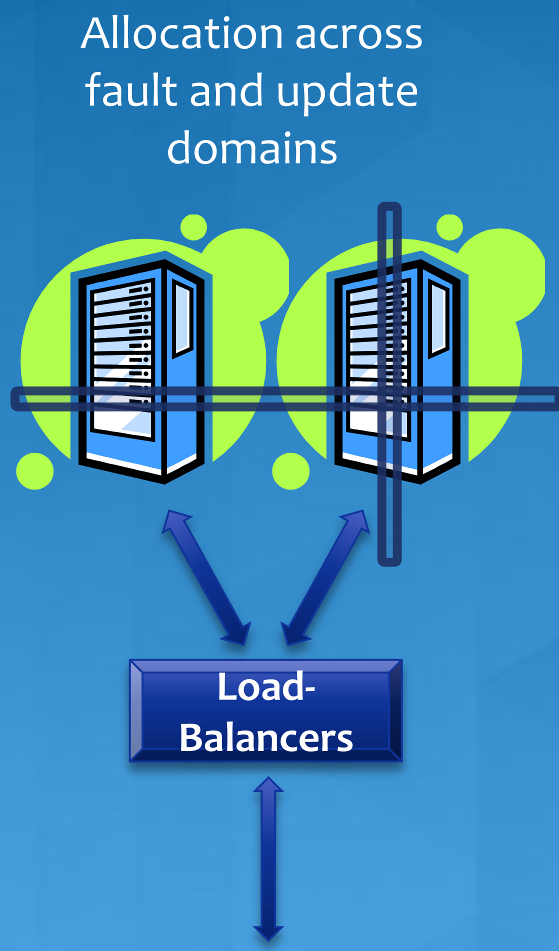
Step 3: Configure settings

Step 4: Start Roles

Step 5: Configure load-balancers

Step 6: Maintain desired number of roles

- Failed roles automatically restarted
- Node failure results in new nodes automatically allocated



Windows Azure Compute Fabric

The FC Keeps Your Service Running

Windows Azure FC monitors the health of roles

- FC detects if a role dies
- A role can indicate it is unhealthy
  - *Current state of the node is updated appropriately*
  - *State machine kicks in again to drive us back into goals state*

Windows Azure FC monitors the health of host

- If the node goes offline, FC will try to recover it

If a failed node can't be recovered, FC migrates role instances to a new node

- A suitable replacement location is found
- Existing role instances are notified of config change

Windows Azure Compute Fabric

Behind the Scenes Work

Windows Azure provisions and monitors hardware

- Compute nodes, TOR/L2 switches, LBs, access routers, and node OOB control elements

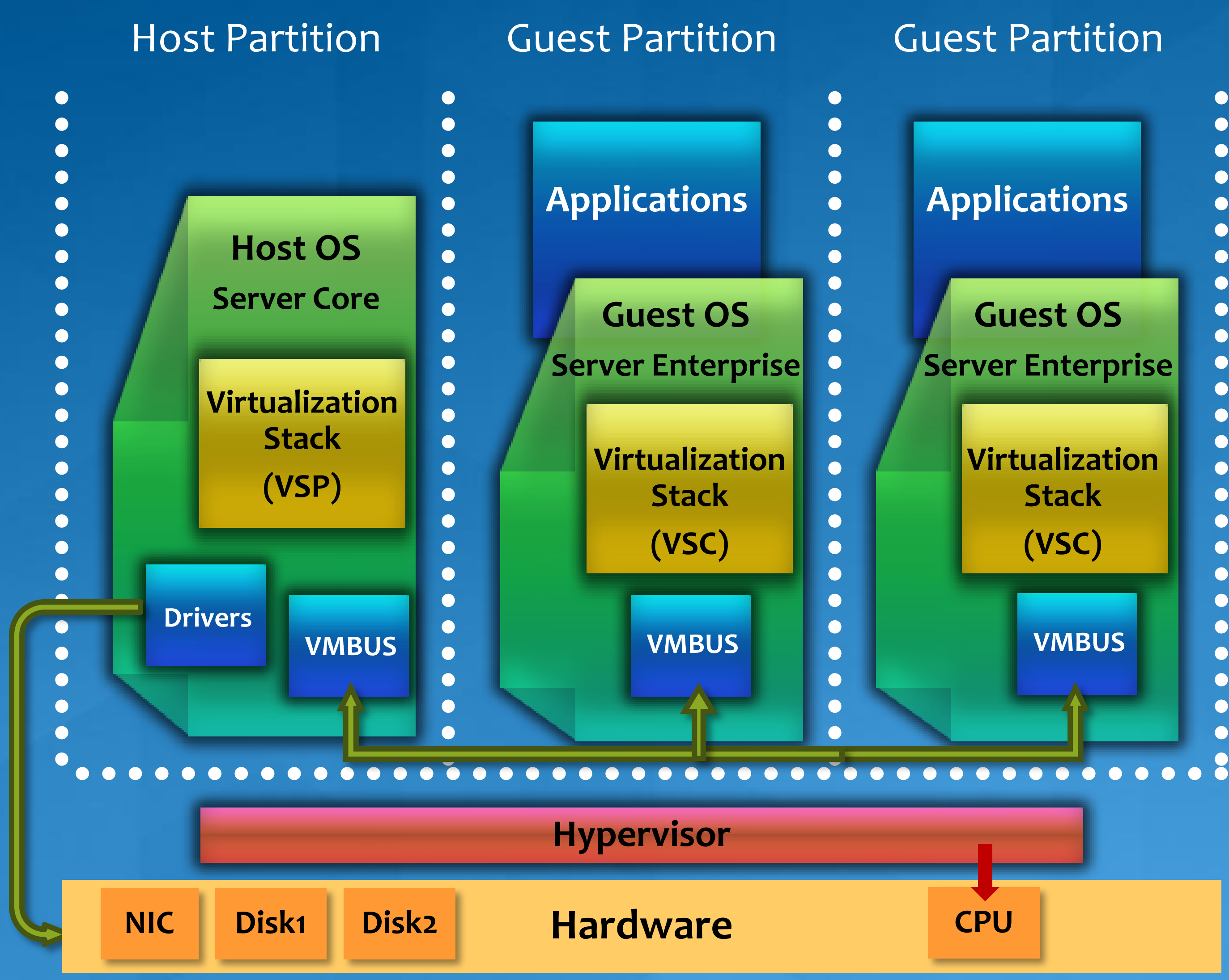
Hardware life cycle management

- Burn-in tests, diagnostics, and repair
- Failed hardware taken out of pool
  - Application of automatic diagnostics
  - Physical replacement of failed hardware

Capacity planning

- On-going node and network utilization measurements
- Proven process for bringing new hardware capacity online

Azure Virtual Computing Environment



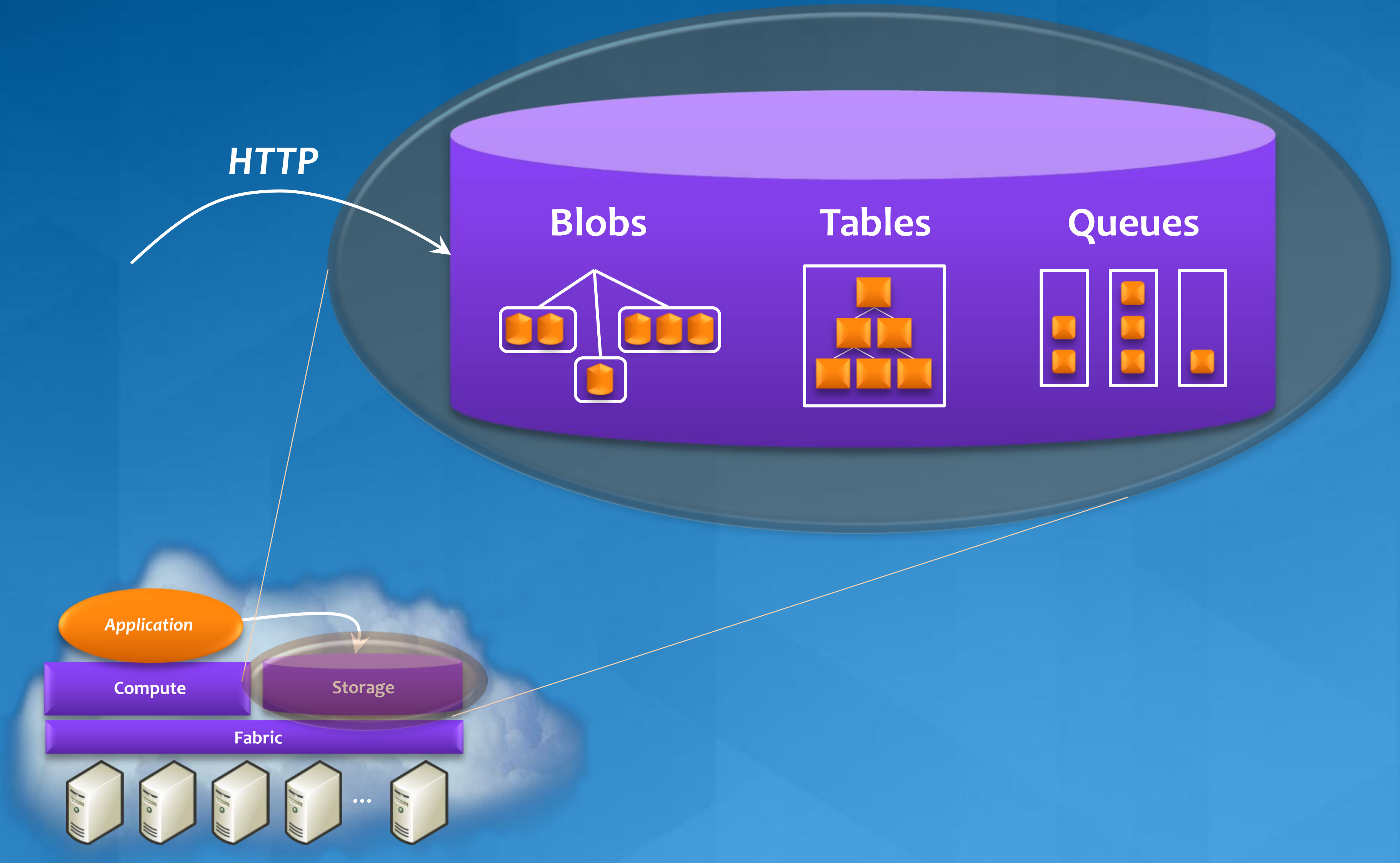
Virtual Computing Environment

Points of interest

- VMs provided by a cloud-optimized hypervisor
- For developers:
  - Applications see a 64-bit Windows Server 2008 interface
    - A few things require accessing the Windows Azure Agent, e.g., logging
  - A desktop replica of Windows Azure is provided for development
    - Called the *Development Fabric*

Windows Azure Storage Service

A closer look



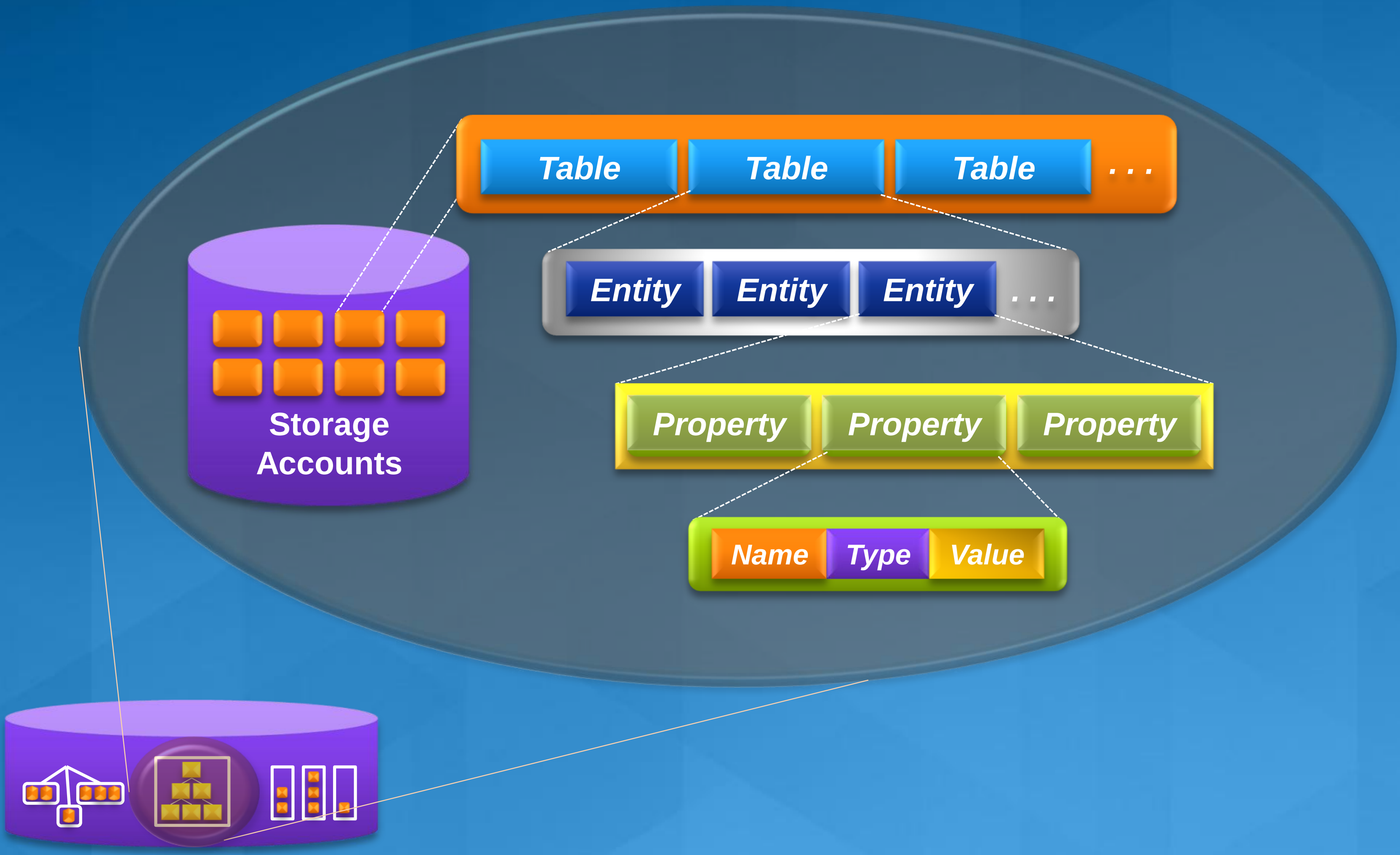
Windows Azure Storage

Points of interest

- Storage types
  - Blobs: a simple hierarchy of binary data
  - Tables: entity-based storage
    - Not relational tables
  - Queues: allow message-based communication
- Access
  - Data is exposed via .NET and RESTful interfaces
  - Data can be accessed by:
    - Windows Azure apps
    - Other on-premises or cloud apps

Windows Azure Storage

A closer look at tables

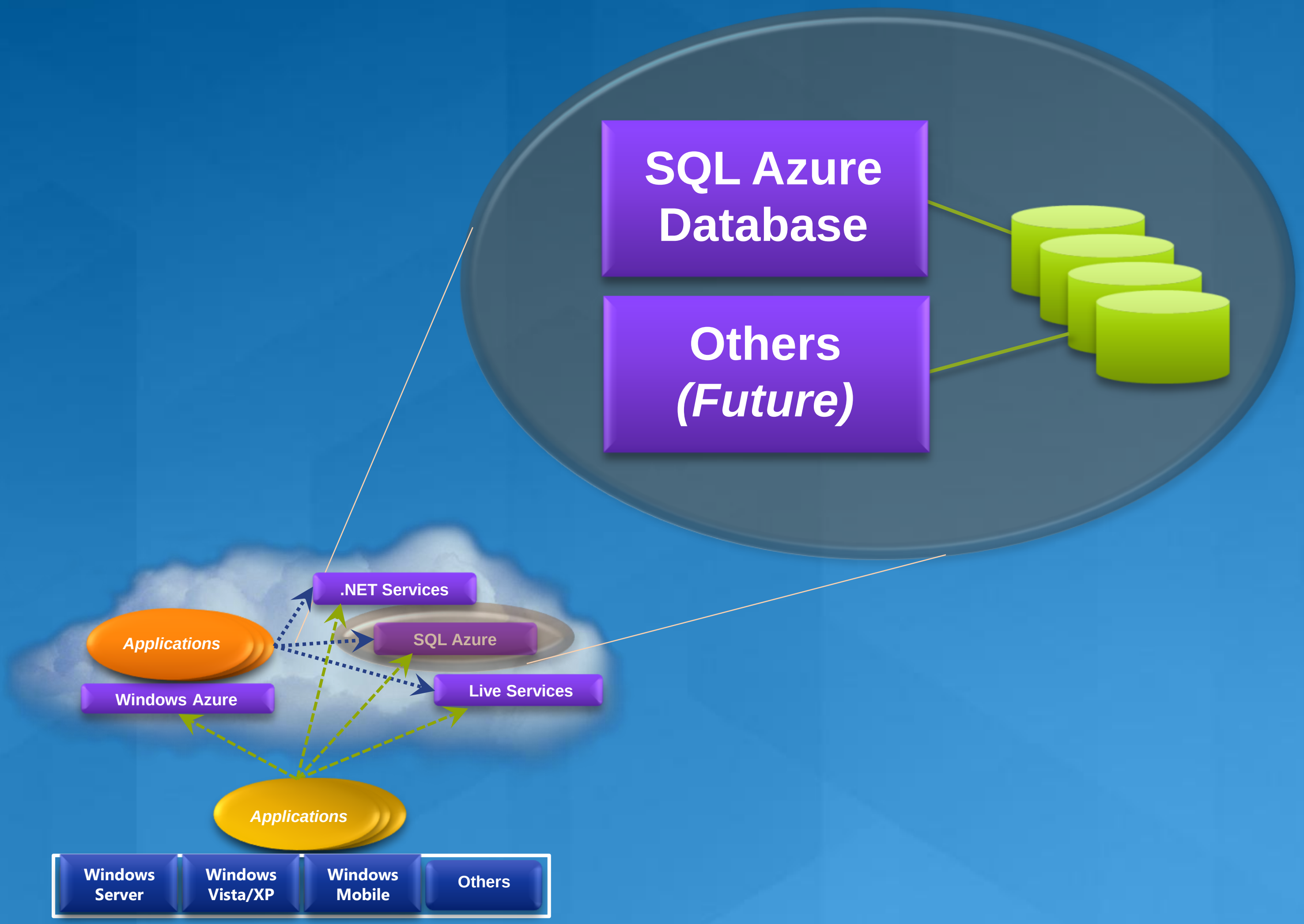


Windows Azure Storage

Tables: Strengths

- Massive scalability
  - By effectively allowing *scale-out* data
- Perspective:
  - Applied to the right problem, Windows Azure Tables are a beautiful thing
    - But they're not the optimal solution for all data storage scenarios
  - Amazon, Google, and others provide similar storage mechanisms
    - It appears to be the state of the art for scale-out data

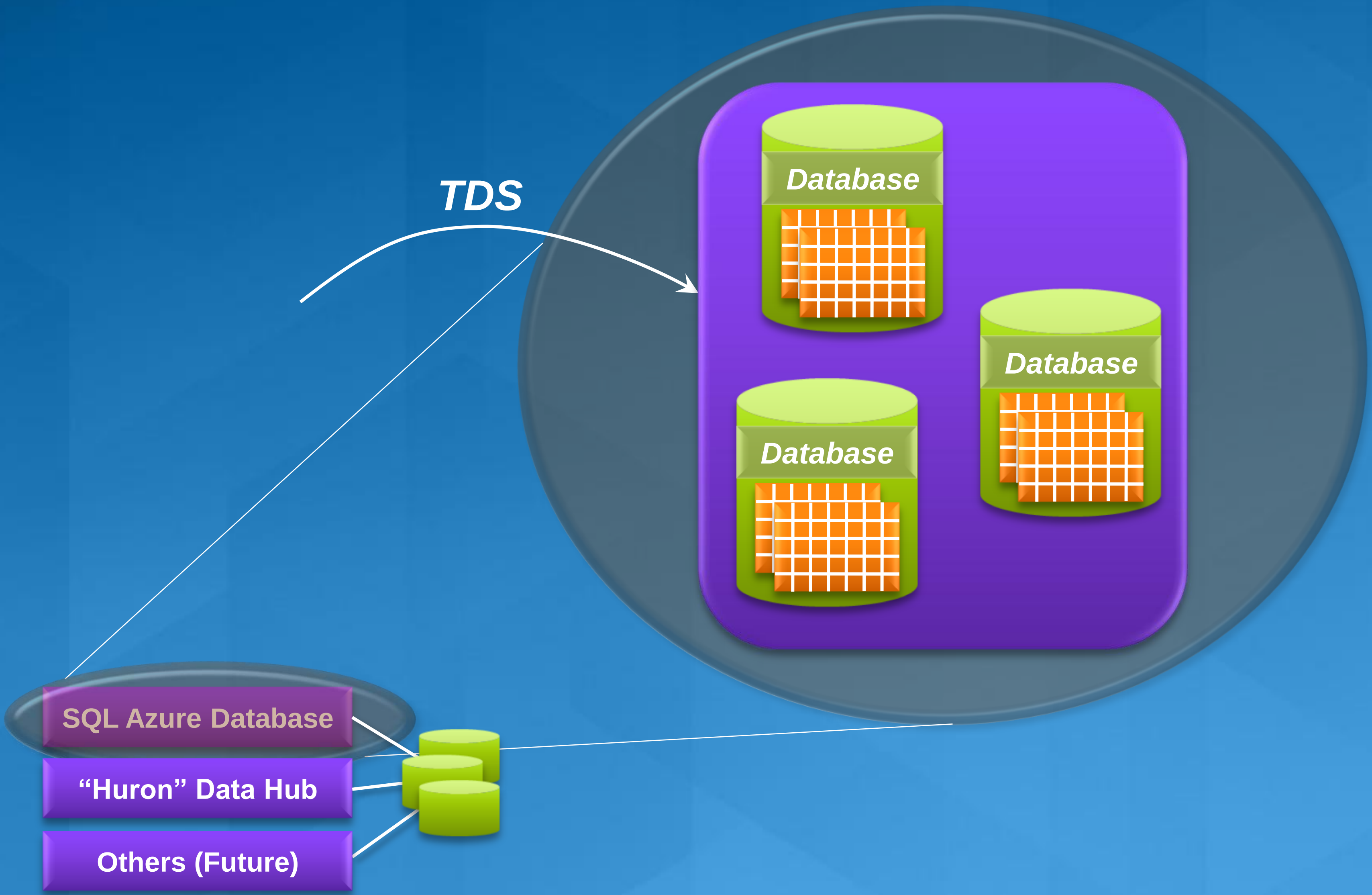
SQL Azure



SQL Azure Database
Others (Future)
.NET Services
Applications
SQL Azure
Live Services
Windows Azure
Applications
Windows Server
Windows Vista/XP
Windows Mobile
Others

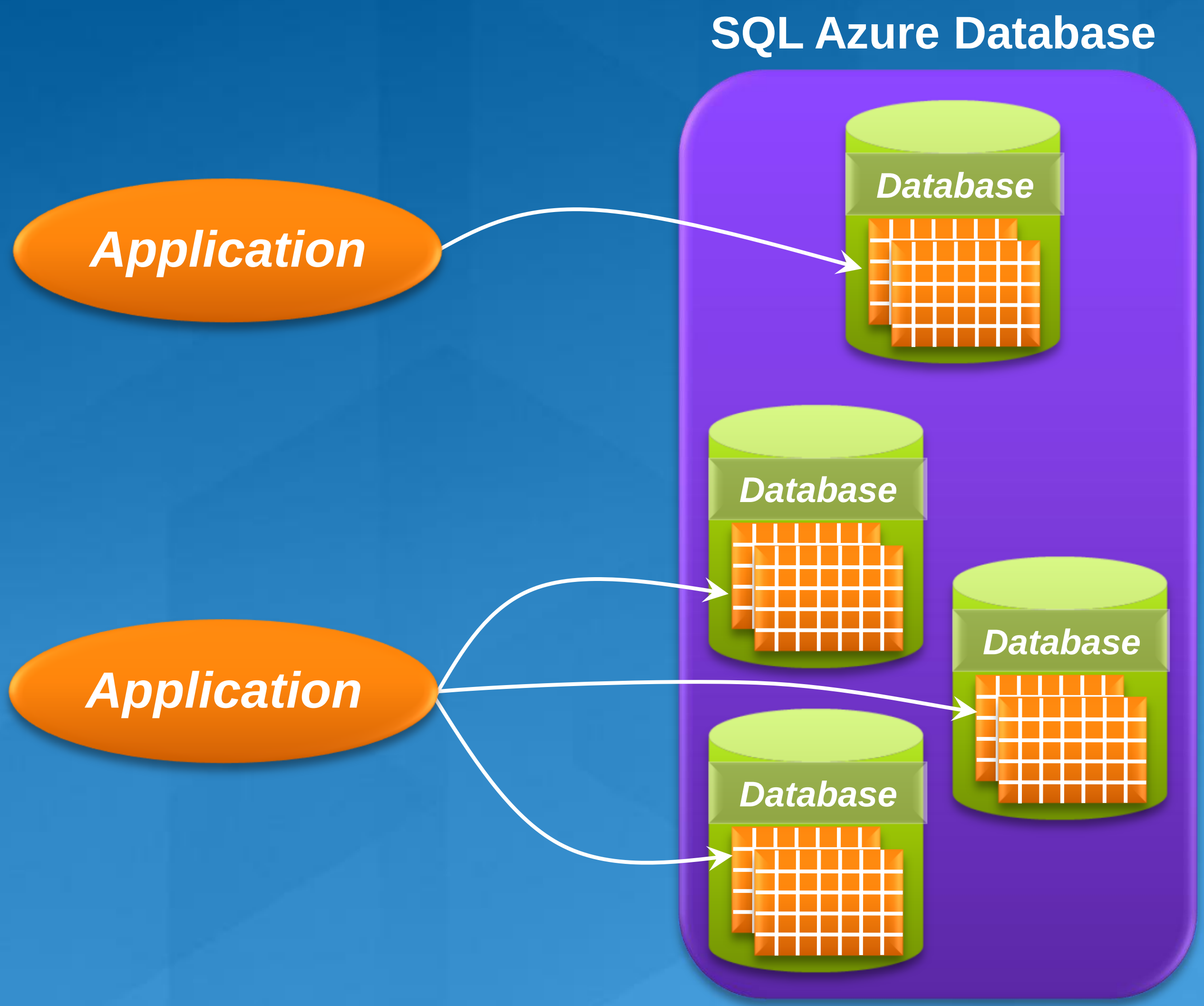
SQL Azure Database

An illustration



SQL Azure Database

Using one or multiple databases



Windows Azure Storage

Points of Interest

- Dynamic replication and scanning for bit rot
  - Automatically maintains data at a healthy number replicas
- Efficient Failover
  - Serve data immediately from another server on a failure
- Automatic Load Balancing of Hot Data
  - Monitor the usage patterns of partitions and servers
  - Automatically load balance partitions across servers
- Caching
  - Hot data pages are cached and served directly from memory at the Partition Layer
  - Hot Blobs are cached at our Front Ends to help scale out access to them

Windows Azure

Key takeaways

Cloud services have specific design considerations

- Always on, distributed state, large scale, fault tolerance
- Scalable infrastructure demands a scalable architecture
  - Stateless roles and durable queues

Windows Azure frees service developers from many platform issues

Windows Azure manages both services and servers

Distributed Data-Parallel Computing

A radical approach to programming at scale

- Nodes talk to each other as little as possible (shared nothing)
- Programmer is not allowed to communicate between nodes
- Data is spread throughout machines in advance, computation happens where it's stored.
- Master program divvies up tasks based on location of data, detects failures and restarts, load balances, etc…

Microsoft's Dryad

- Running on >> 10^4 machines, analyzing > 10Pb data daily
- Runs on clusters > 3000 machines
- Handles jobs with > 10^5 processes each
- Used by >> 100 developers
- Rich platform for data analysis

LINQ

Microsoft's Language **IN**tegrated **Q**uery

Available in Visual Studio 2008

A set of operators to manipulate datasets in .NET

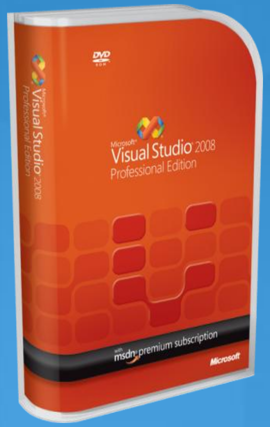
- Support traditional relational operators
  - Select, Join, GroupBy, Aggregate, etc.

Data model

- Data elements are strongly typed .NET objects
- Much more expressive than SQL tables

Extremely extensible

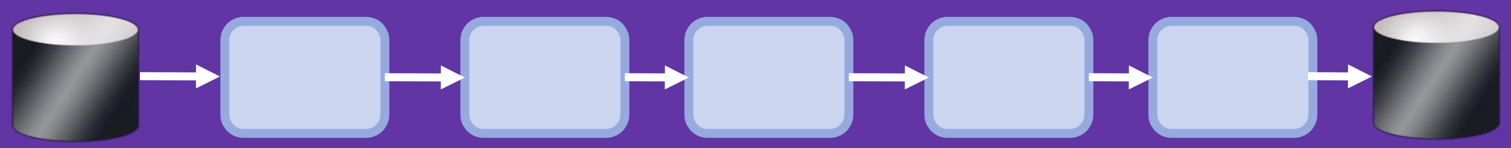
- Add new custom operators
- Add new execution providers



Dryad Generalizes Unix Pipes

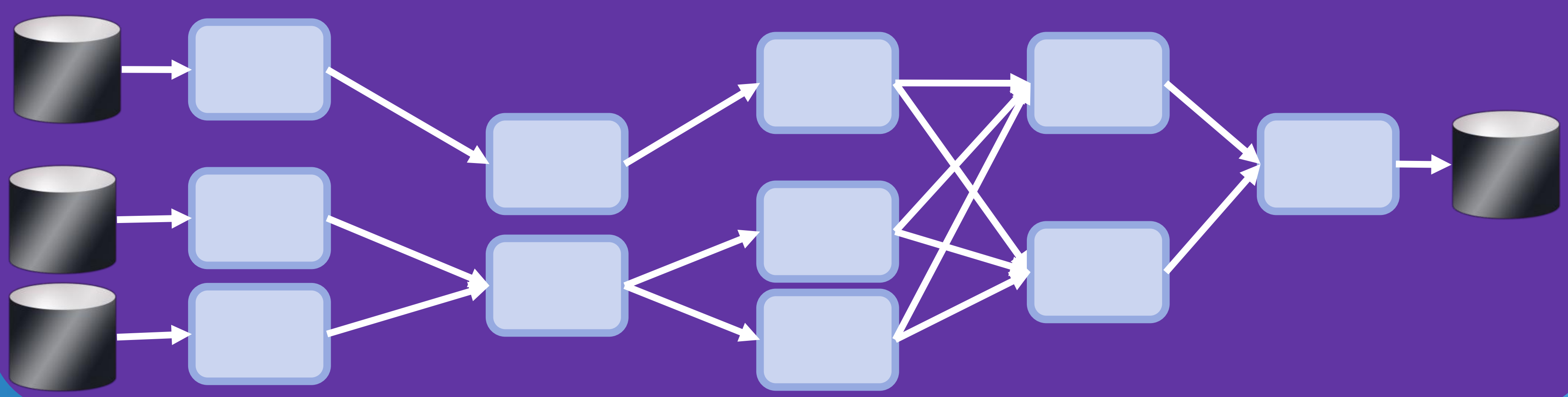
Unix Pipes: 1-D

grep | sed | sort | awk | perl

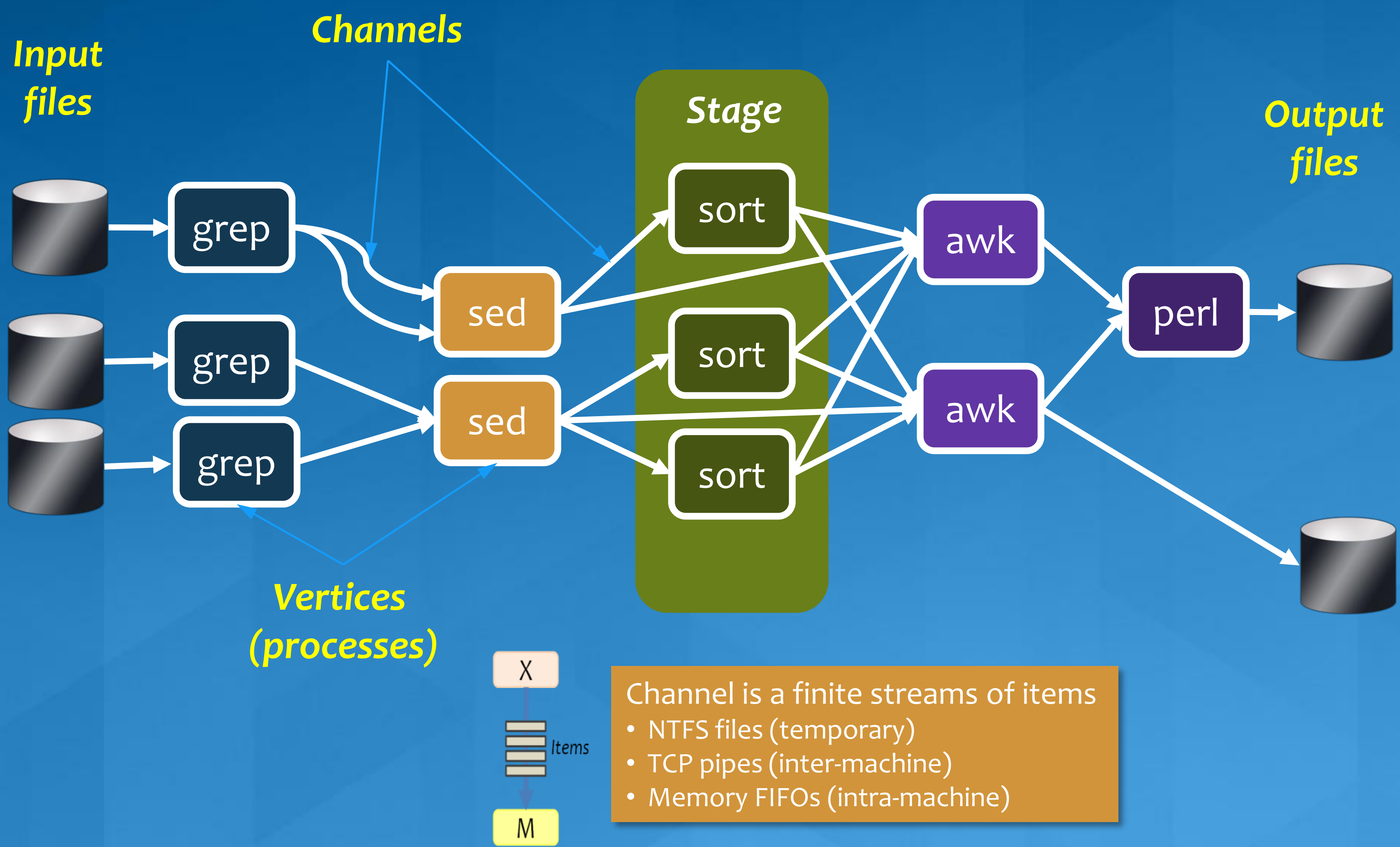


Dryad: 2-D, multi-machine, virtualized

$grep^{1000}$ | sed^{500} | $sort^{1000}$ | awk^{500} | $perl^{50}$

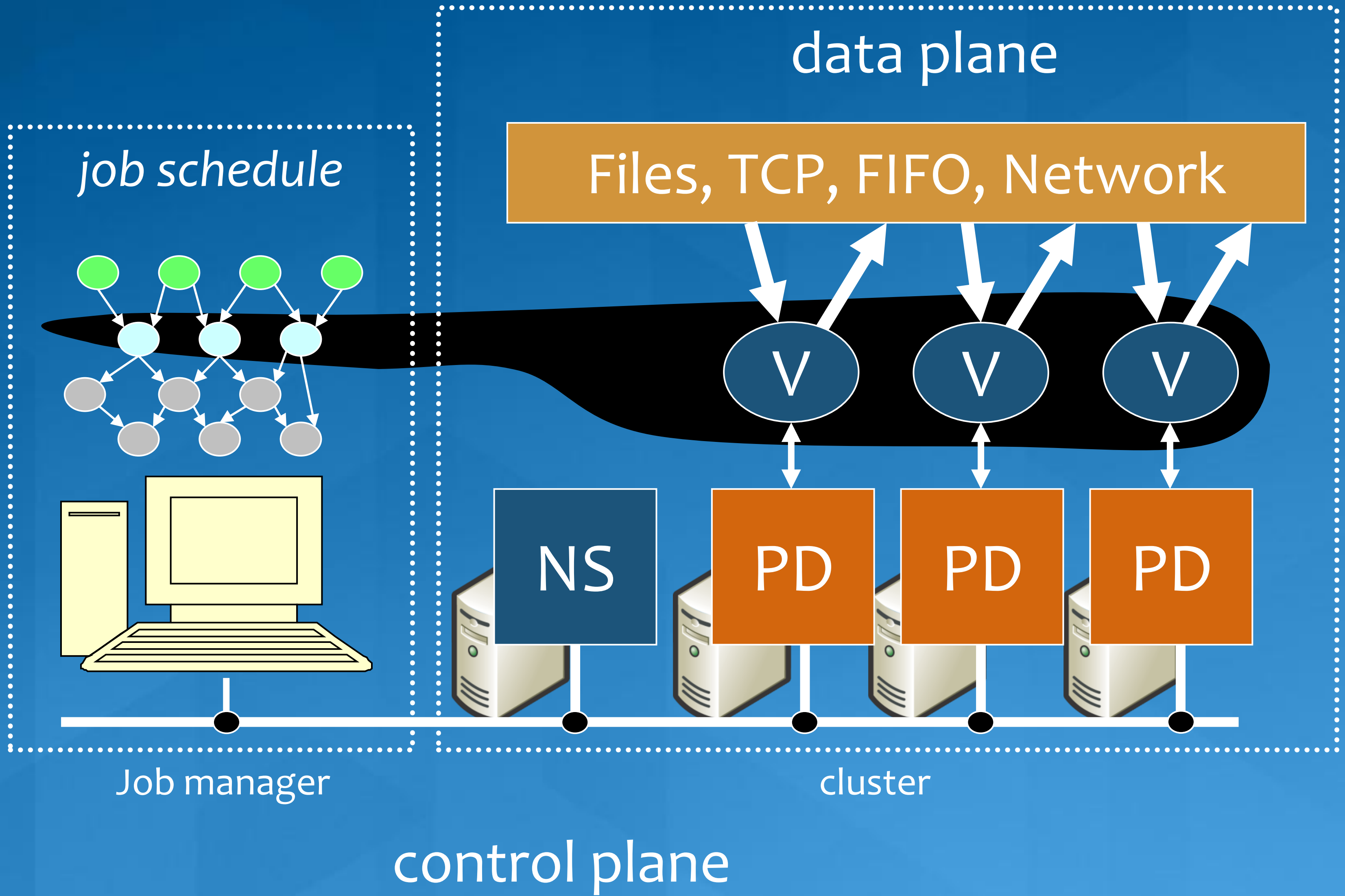


Dryad Job Structure

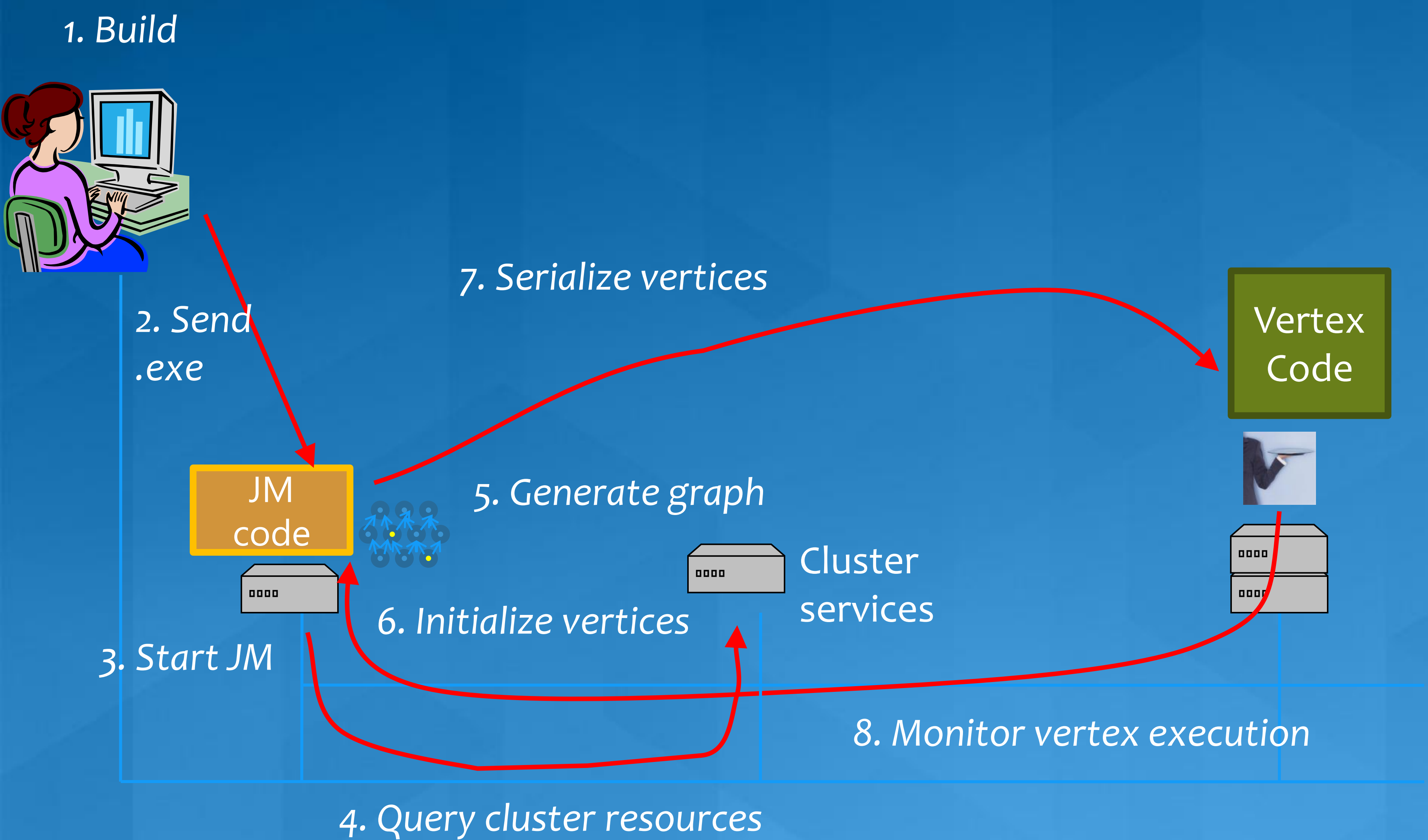


Channel is a finite streams of items
- NTFS files (temporary)
- TCP pipes (inter-machine)
- Memory FIFOs (intra-machine)

Dryad System Architecture

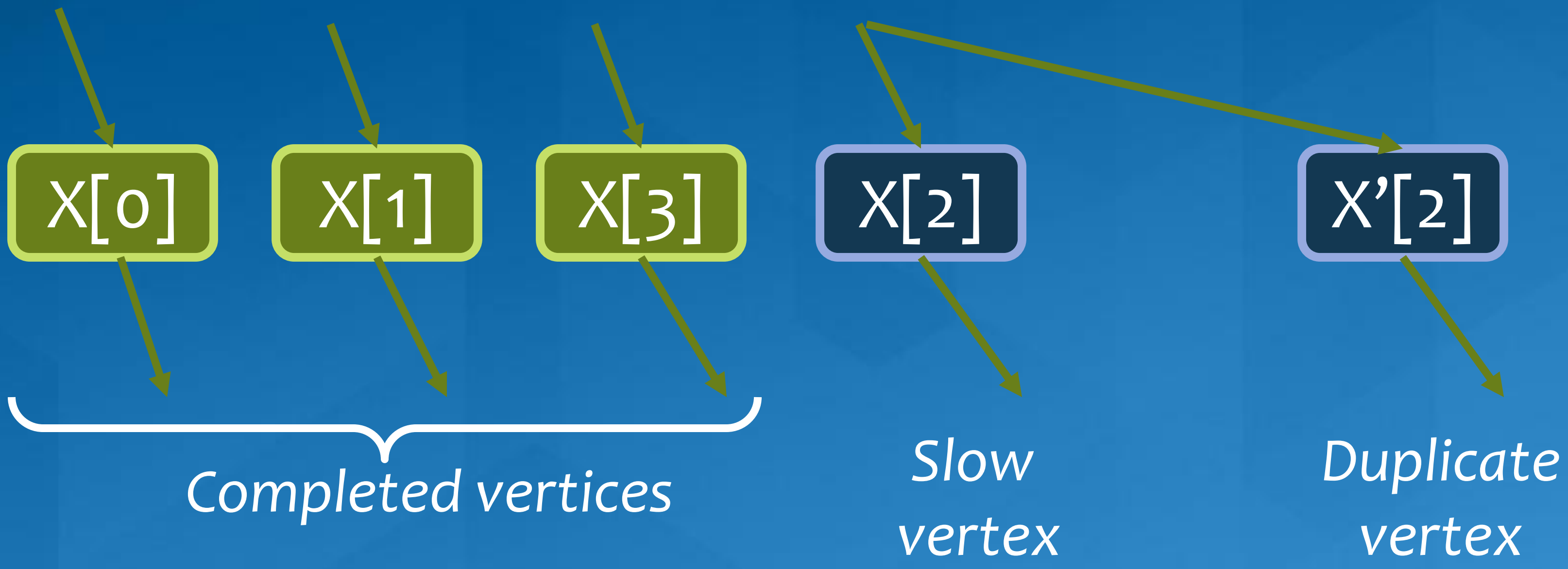


Dryad Job Staging



Fault Tolerance

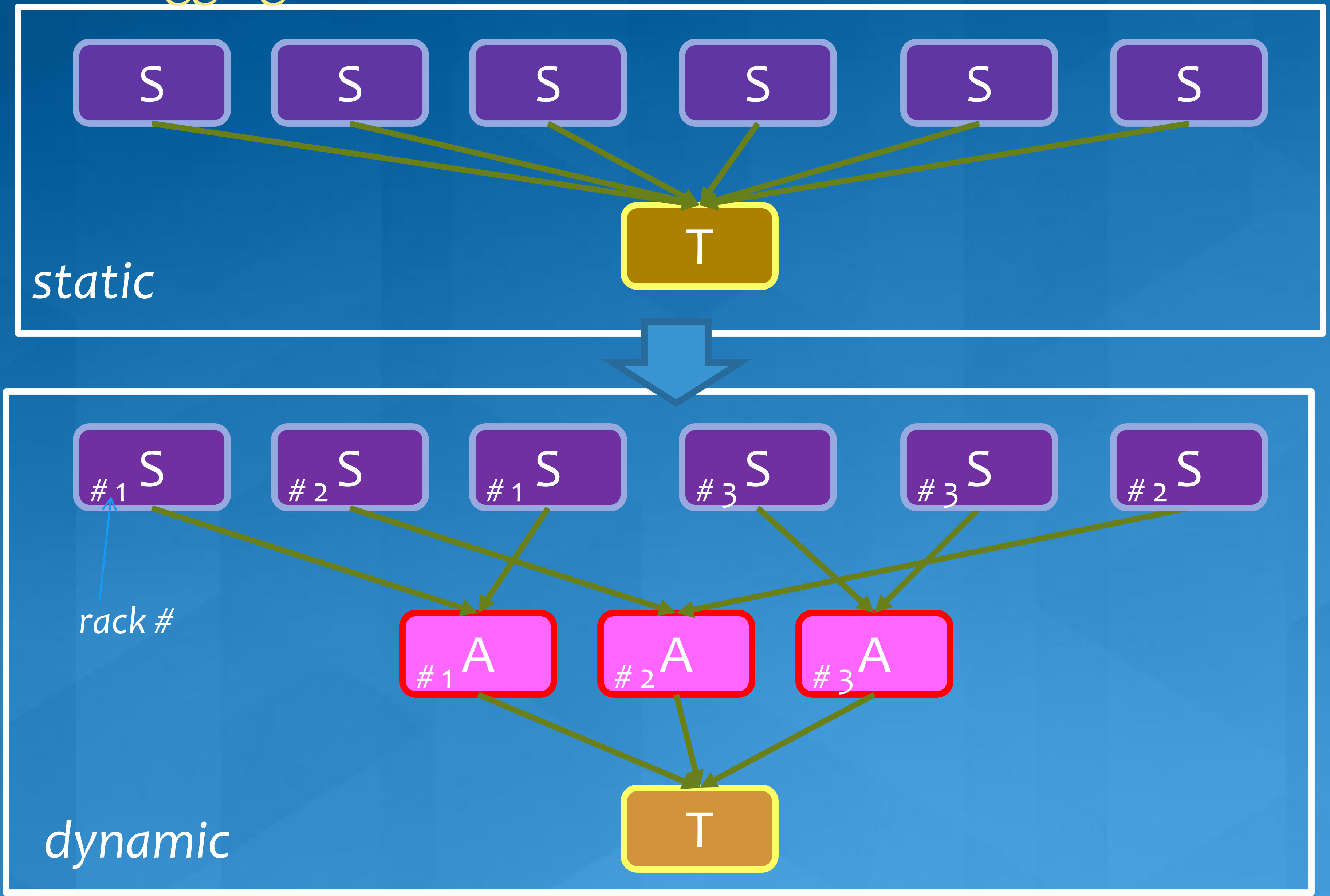
Dynamic Graph Rewriting



Duplication Policy = f(running times, data volumes)

Dryad

Dynamic Aggregation



Dryad Scheduler is a State Machine

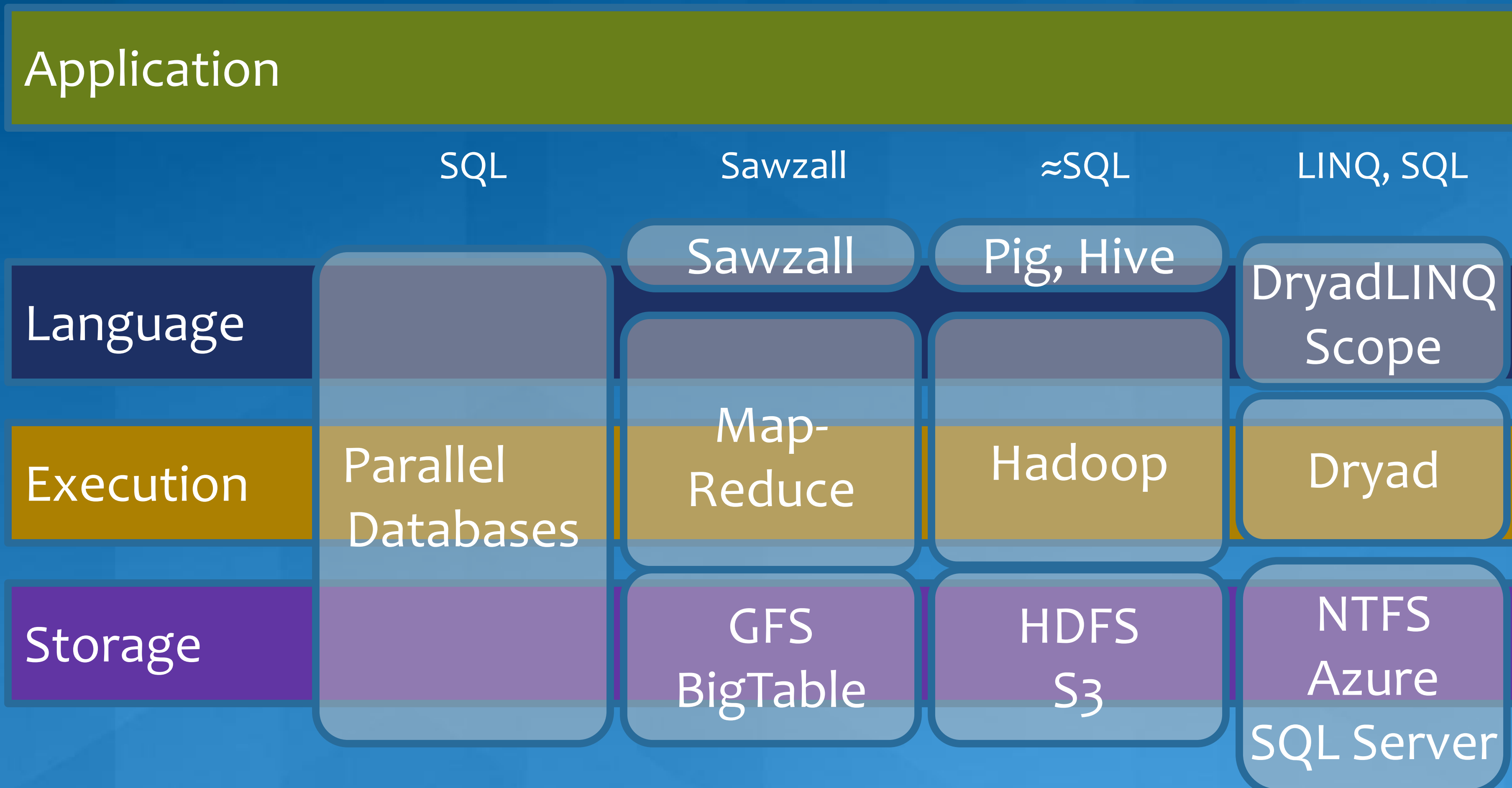
Static optimizer builds execution graph

- Vertex can run anywhere once all its inputs are ready.

Dynamic optimizer mutates running graph

- Distributes code, routes data;
- Schedules processes on machines near data;
- Adjusts available compute resources at each stage;
- Automatically recovers computation, adjusts for overload
  - If A fails, run it again;
  - If A's inputs are gone, run upstream vertices again (recursively);
  - If A is slow, run a copy elsewhere and use output from one that finishes first.
- Masks failures in cluster and network;

Dryad in Context



Windows Azure References

- Windows Azure
  - Home page
    - www.microsoft.com/azure
  - Developer SDKs (.NET, Java, Ruby and PHP)
    - www.microsoft.com/azure/sdk.mspx
    - phpazure.codeplex.com/
  - Training kit
    - www.microsoft.com/azure/trainingkit.mspx
  - Blogs and discussion groups
    - www.microsoft.com/azure/blog.mspx
- Microsoft Open Government Data Initiative (OGDI)
  - ogdisdk.cloudapp.net/

Tutorial Outline

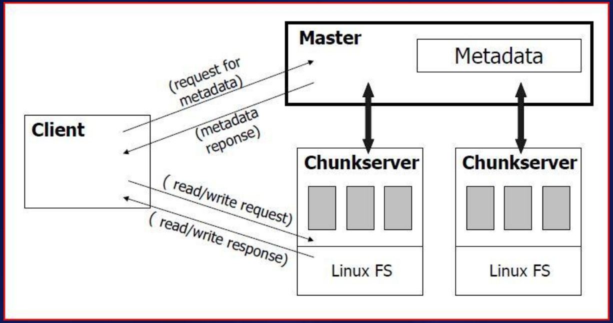
- Part 4. More Programming Models & Services.
  - Google App Engine.
  - The Zend/MS/IBM Simple Cloud APIs
  - HPC and the Cloud

Google App Engine

- App Engine is designed to make it possible to build scalable web applications without building the complex infrastructure required.
- The programmers challenge:
  - You know how to build a web app built on a single server with a database backend. It can serve 10 users concurrently.   Now, scale it to 100,000 concurrent users.
- App engine philosophy
  - Provide users standard front end tools: Python & Java for the user web-front end.
  - Given them a model for building stateless services based on a perfectly scalable replacement for the DB.

Google App Engine Foundation

- Google has built a massive infrastructure designed for their web search and indexing projects
- Built on the Google File System
  - Object partitioned into chunks
    - Managed by a “chunk server”
    - Chunks are replicated on multiple chunkservers.
  - Designed to optimize for lots of reads, few rights, highly concurrent and very reliable.
  - Strongly consistent and optimistic concurrency control



- On top of GFS is BigTable
  - Table storage similar to Azure Tables.
  - GQL is SQL without Joins

```
SELECT * FROM Story WHERE
   title = 'App Engine Launch'
   AND author = :current_user
   AND rating >= 10
  ORDER BY
    rating, created DESC
```


Microsoft

App Engine Features

- From Google's Website:
  - dynamic web serving, with full support for common web technologies
  - persistent storage with queries, sorting and transactions
  - automatic scaling and load balancing
  - APIs for authenticating users and sending email using Google Accounts
  - a fully featured local development environment that simulates Google App Engine on your computer
  - task queues for performing work outside of the scope of a web request
  - scheduled tasks for triggering events at specified times and regular intervals

Limitations of App Engine

- App Engine is not designed for large scale data analysis
  - Google has a separate MapReduce capability for data analysis.   This is not currently accessible from AE.
- App components are intended to be stateless (state should be in the datastore/BigTable) and execute quickly. This insures scalability.
- Currently there is no way to upload trusted binary executables. Everything runs in a sandbox.

The Simple Cloud APIs

- Not a standards effort.
  - The Simple Cloud API is an open source project that makes it easier for developers to use cloud application services by abstracting insignificant API differences.
- API provides interfaces for File Storage, Document Storage, and Simple Queue services.
- More to come in the future.

The Storage Goals

- Coverage
  - File Storage, such as Rackspace Cloud Files, Windows Azure Blob Storage, Amazon S3, and Nirvanix
  - Document Storage, such as Amazon SimpleDB and Windows Azure Table Storage
  - Simple Queues, such as Windows Azure Table Storage and Amazon SQS
- Designed to be very simple.
  - But allows you to also access vendor specific features.
- API is PHP
  - Covers much of the web development space!

File Storage

```
interface Zend_Cloud_StorageService {
   public function fetchItem($path, $options = null);
   public function storeItem($data, $destinationPath,
                                            $options = null);
   public function deleteItem($path, $options = null);
   public function copyItem($sourcePath, $destinationPath,
                                           $options = null);
   public function moveItem($sourcePath, $destinationPath,
                                            $options = null);
   public function fetchMetadata($path, $options = null);
   public function deleteMetadata($path);
}
```


Microsoft

Document (Table) Storage

- Based on concept of collections of documents
  - Maps to tables of rows in Azure

```
interface Zend_Cloud_DocumentService {
  public function createCollection($name, $options = null);
  public function deleteCollection($name, $options = null);
  public function listCollections($options = null);
  public function listDocuments($options = null);
  public function insertDocument($document, $options = null);
  public function updateDocument($document, $options = null);
  public function deleteDocument($document, $options = null);
  public function query($query, $options = null);
}
```

Queues

- Queues in clouds provide reliable, scalable persistent messaging.

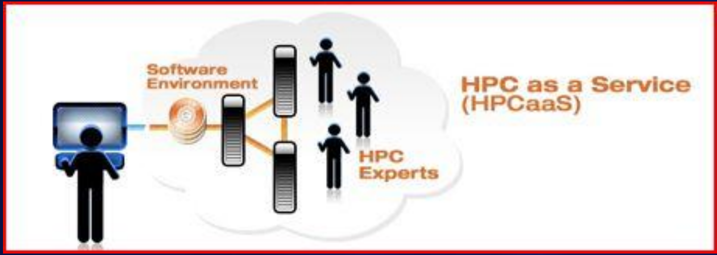
```
interface Zend_Cloud_QueueService {
   public function createQueue($name, $options = null);
   public function deleteQueue($name, $options = null);
   public function listQueues($options = null);
   public function fetchQueueMetadata($name, $options = null);
   public function storeQueueMetadata($metadata, $name, $options = null);
   public function sendMessage($message, $queueName, $options = null);
   public function recieveMessages($queueName, $max = 1, $options = null);
   public function deleteMessage($id, $queueName, $options = null);
   public function peekMessage($id, $queueName, $options = null);
}
```

Simple Cloud API

- With the basic storage API it is possible to write simple single tier PHP web apps that can be ported from one provider to another.
- Next steps
  - Can security and authentication be generalized?
  - Can this be extended to multi-tier apps?
    - Not clear as many basic model concepts differ

HPC and the Cloud

- Not a totally new idea
  - HPC as a Service™ - A new offering from Penguin Computing
  - Running a virtualized environment on the head node of a cluster. Apps run on bare cluster hardware

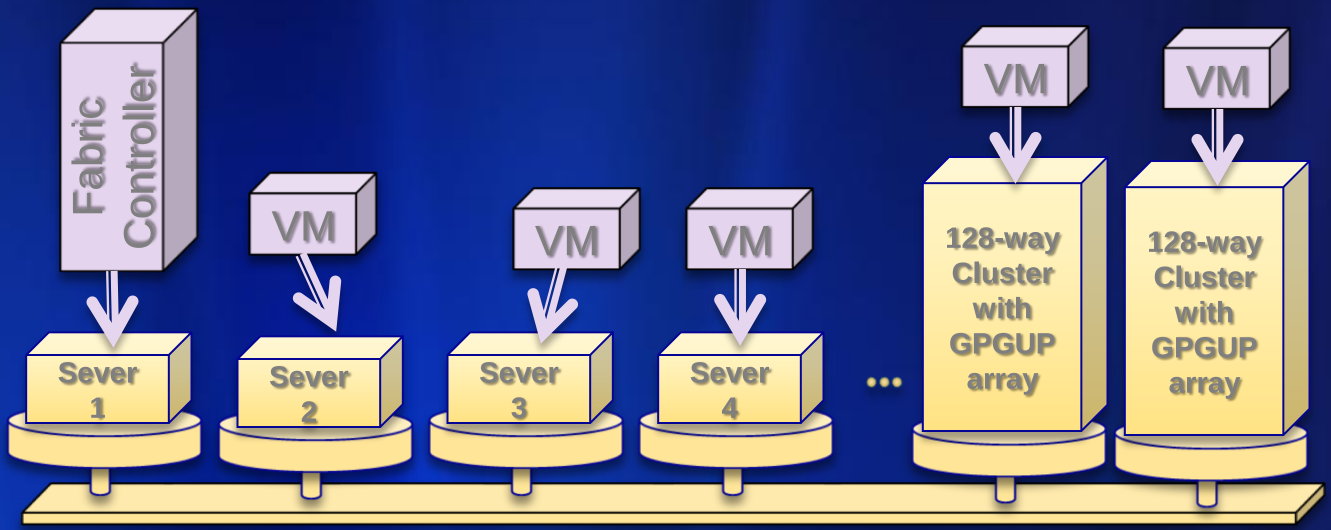


- Cloud virtualization can introduce node-to-node communication latency
  - But it has been shown it is possible to reduce this.
- Some cloud VMs can span nodes with multiple cores.
- Possible to introduce GPGPUs as well.

Microsoft

A Possible HPC Cloud

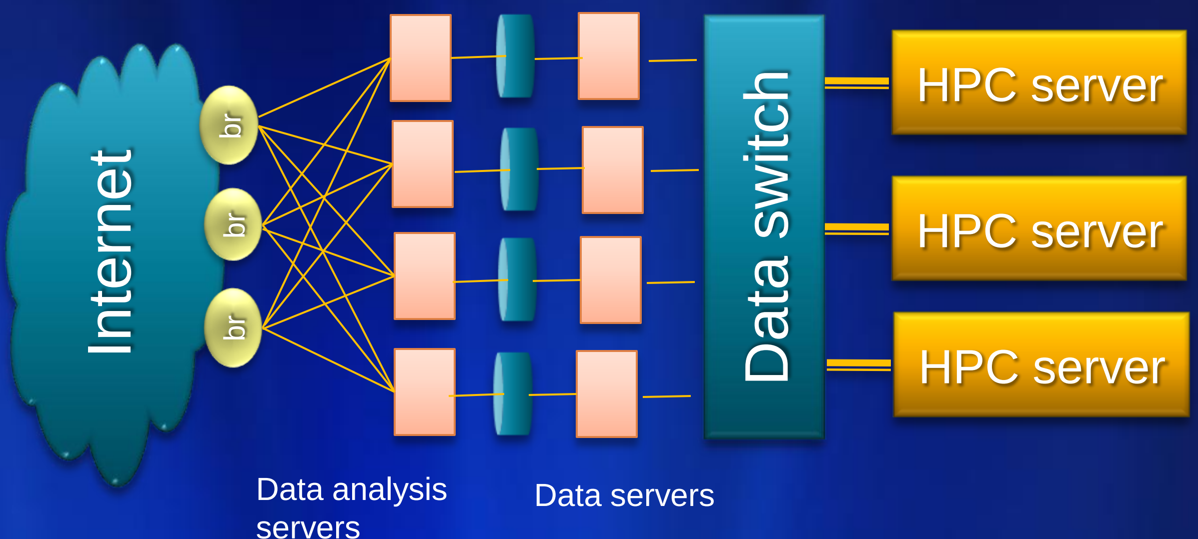
- Many HPC apps are ensembles of modestly parallel jobs.
- Introduce a heterogeneous data center model with
  - Simple servers for gateway activity and multiple back end, tightly coupled clusters for computationally intensive tasks.
- There are many challenges to make this work.



Microsoft

HPC Cloud challenges

- The data models for cloud and HPC are very different.
  - In the cloud: keep data distributed, replicated and local
  - HPC computations swap data from remote storage and computation is the expensive part.
- Can we design an interconnect that bridges both worlds?



The Cloud Tutorial

Dan Reed, Roger Barga, Dennis Gannon
Microsoft Research
eXtreme Computing Group

Rich Wolski
Eucalyptus.com

Microsoft