

Microsoft® Research

Faculty Summit 2010

Microsoft® Research

Faculty Summit 2010

Azure for Science Research: From Desktop to the Cloud

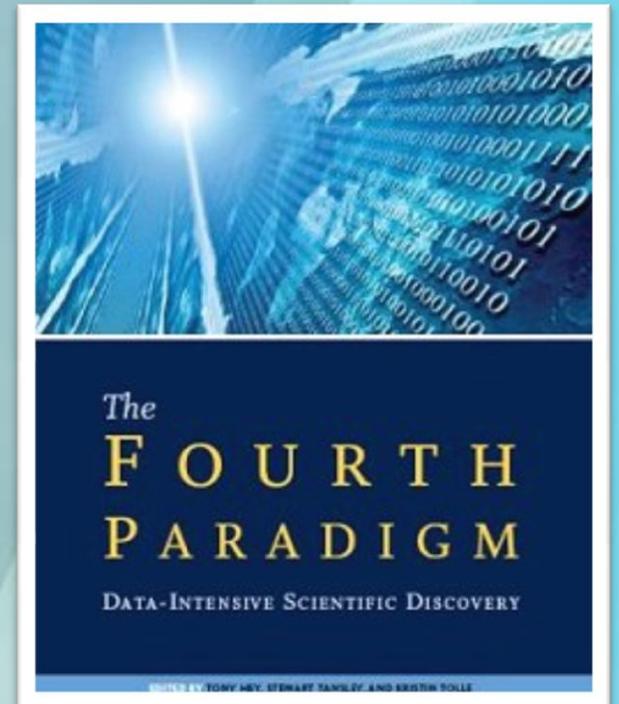
Roger Barga
Senior Architect

Catharine van Ingen
Partner Architect

Microsoft Corporation

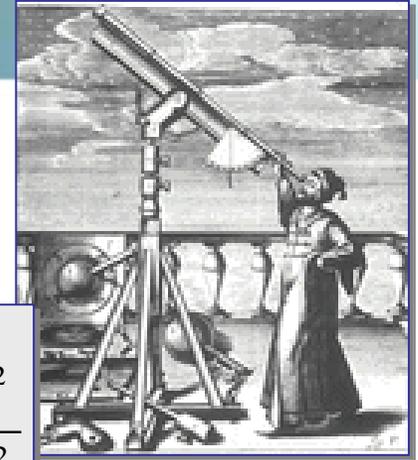
The Data Flood: Science and the 4th Paradigm

Small keys open big doors
Turkish Proverb

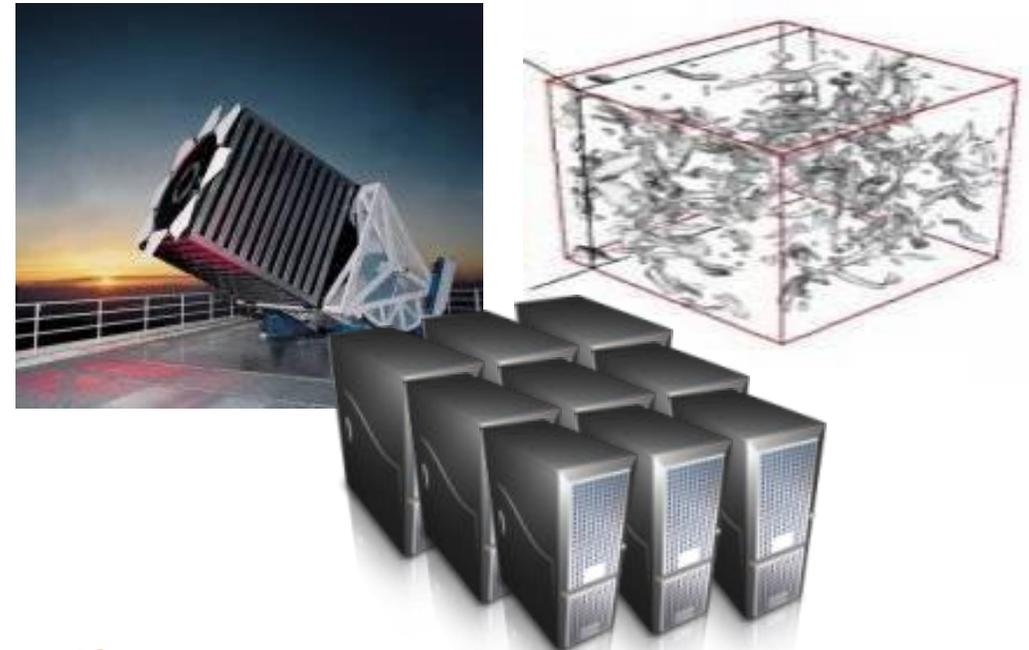


Emergence of a Fourth Paradigm

- Thousand years ago – **Experimental Science**
 - Description of natural phenomena
- Last few hundred years – **Theoretical Science**
 - Newton's Laws, Maxwell's Equations...
- Last few decades – **Computational Science**
 - Simulation of complex phenomena
- Today – **Data-Intensive Science**
 - Scientists overwhelmed with data sets from many different sources
 - Data captured by instruments
 - Data generated by simulations
 - Data generated by sensor networks
 - eScience is the set of tools and technologies to support data federation and collaboration
 - For analysis and data mining
 - For data visualization and exploration
 - For scholarly communication and dissemination



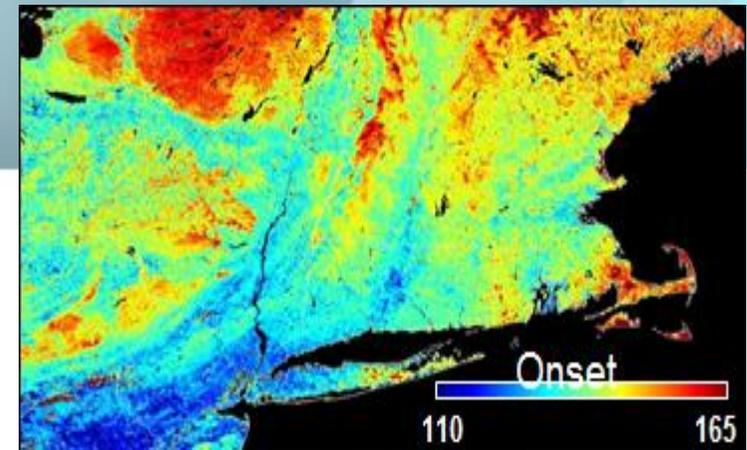
$$\left(\frac{\dot{a}}{a}\right)^2 = \frac{4\pi G \rho}{3} - K \frac{c^2}{a^2}$$



Jim Gray 2007

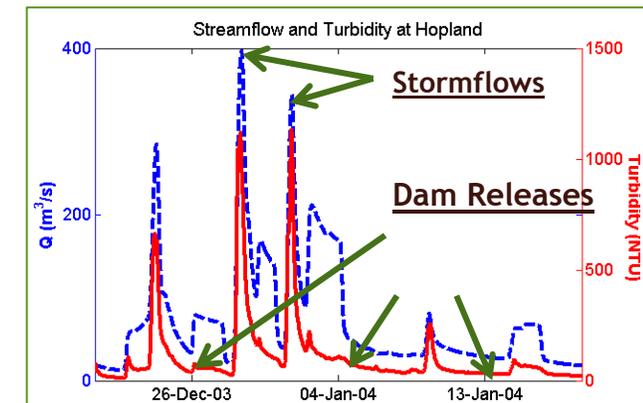
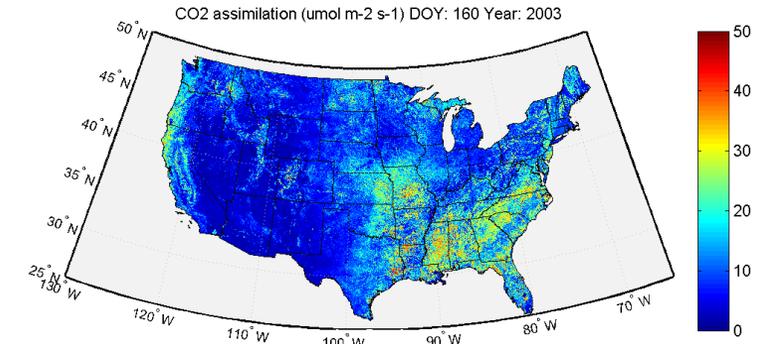
Example: The Ecological Data Flood

- We're living in a perfect storm of remote sensing, cheap ground-based sensors, internet data access, and commodity computing
- Yet deriving and extracting the variables needed for science remains problematic
 - Specialized knowledge for algorithms, internal file formats, data cleaning, etc, etc
 - Finding the right needle across the distributed heterogeneous and very rapidly growing haystacks



Rasters, Sensors, and Ancillary Data

- Rasters: time series raster data (PBs to TBs)
 - Over some period of time at some time frequency at some spatial granularity over some spatial area
 - Can be “cut out” to create virtual sensors
 - Often requires specialized skills and use esoteric formats
 - Similar, but dirtier, than model output
- Sensors: time series point data (TBs to GBs)
 - Over some period of time at some time frequency at some spatial location.
 - (Re)calibrations, gaps and errors are a way of life because batteries die and sensors fail
 - Gap filling algorithms key as regular time series enable more analyzes
- Ancillary Data: everything else (KBs to maybe GBs)
 - “constants” (latitude), intermittent measurements (water samples), events (algal blooms), descriptions (“shaded”)
 - Hard won and often requires science judgment
 - Analysis usage patterns vary widely



Why Make this Distinction?

- Provenance and trust widely varies
 - Data acquisition, early processing, and reporting ranges from a large government agency to individual scientists.
 - Smaller data often passed around in email; big data downloads can take days (if at all)
- Data sharing concerns and patterns vary
 - Open access followed by (non-repeatable and tedious) pre-processing
 - True science ready data set but concerns about misuse, misunderstanding particularly for hard won data.
- Computational tools differ.
 - Not everyone can get an account at a supercomputer center
 - Very large computations require engineering (error handling)
 - Space and time aren't always simple dimensions

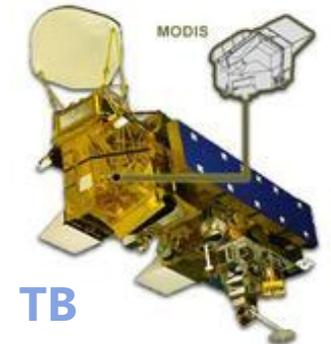
Complex shared detector

Simple instrument (if any)

Science happens when PBs, TBs, GBs, and KBs can be mashed up simply

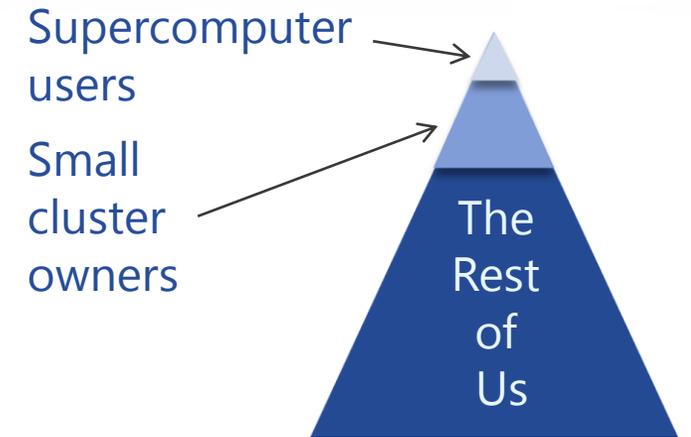
Complex and Heavy process by experts

Ad hoc observations and models

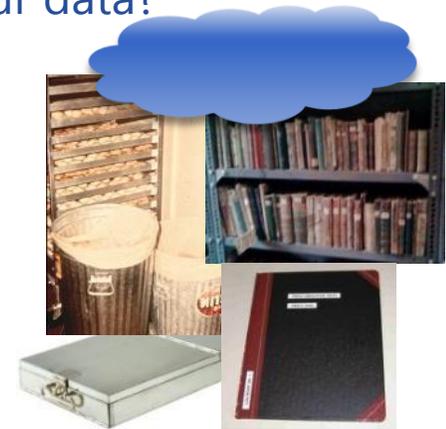


Bridging the Gap with the Cloud

- **Barriers to Science:**
 - Resource: compute, storage, networking, visualization capability
 - Complexity: specific cross-domain knowledge
 - Tedium: repetitive data gathering or preprocessing tasks
- **With Cloud Computing, we can:**
 - marshal needed storage and compute resources on demand without caring or knowing how that happens
 - access living curated datasets without having to find, educate, and reward a private data curator
 - run key common algorithms as Software as a Service without having to know the coding details or installing software
 - grow a given collaboration or share data and algorithms across science collaborations elastically



Where do you want your data?



Democratizing science analysis by fostering sharing and reuse

Azure and Cloud Computing

Ideas rose in clouds; I felt them collide until pairs interlocked, so to speak, making a stable combination.

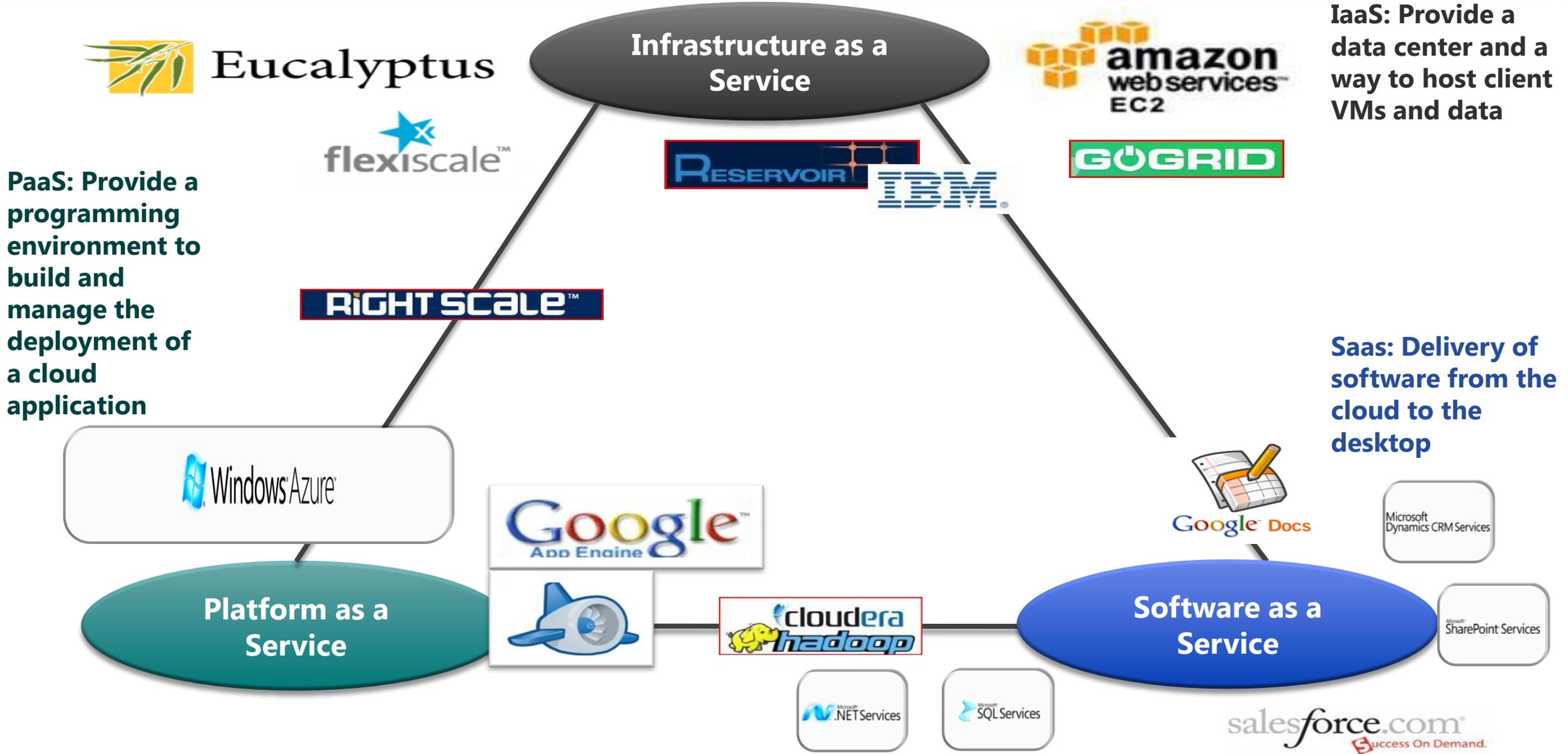
Henri Poincare

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
- A cloud is the “invisible” backend to many of our mobile applications
- Historical roots in today’s Internet apps
 - Search, email, social networks
 - File storage (Live Mesh, Mobile Me, Flickr, ...)



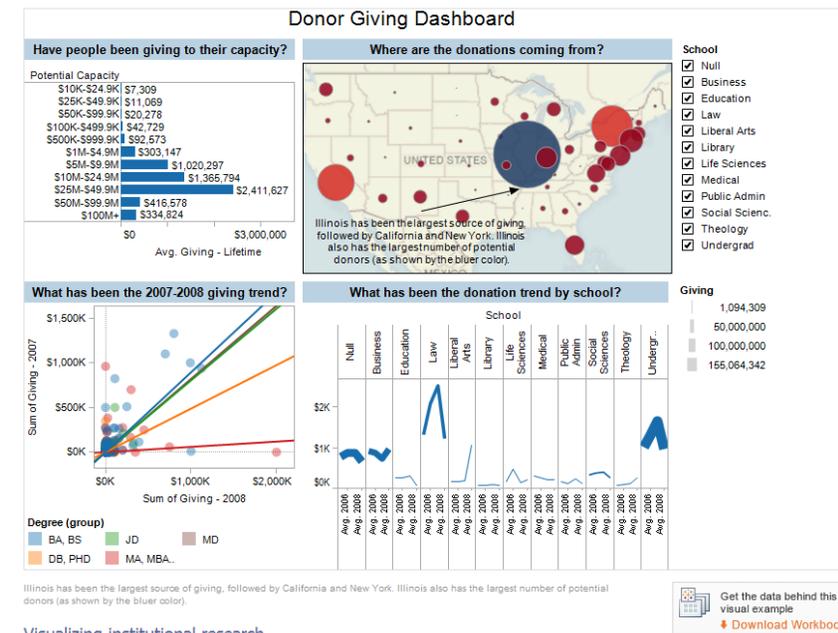
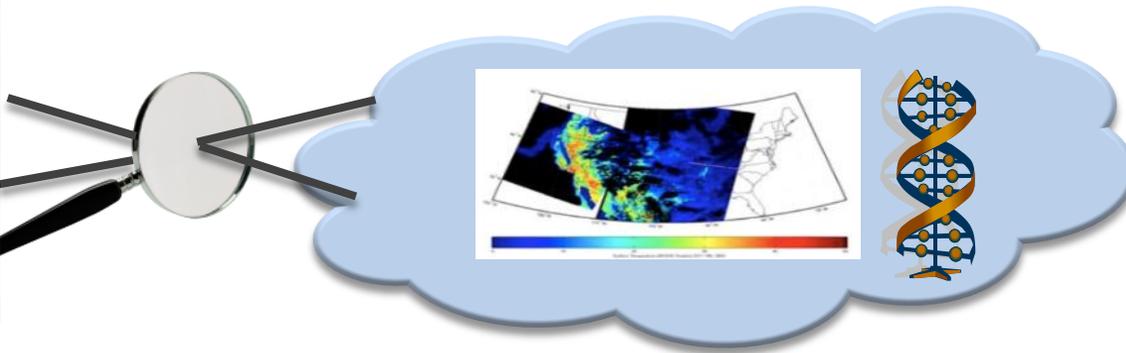
The Cloud Landscape



Research Clients for A Cloud Research Platform



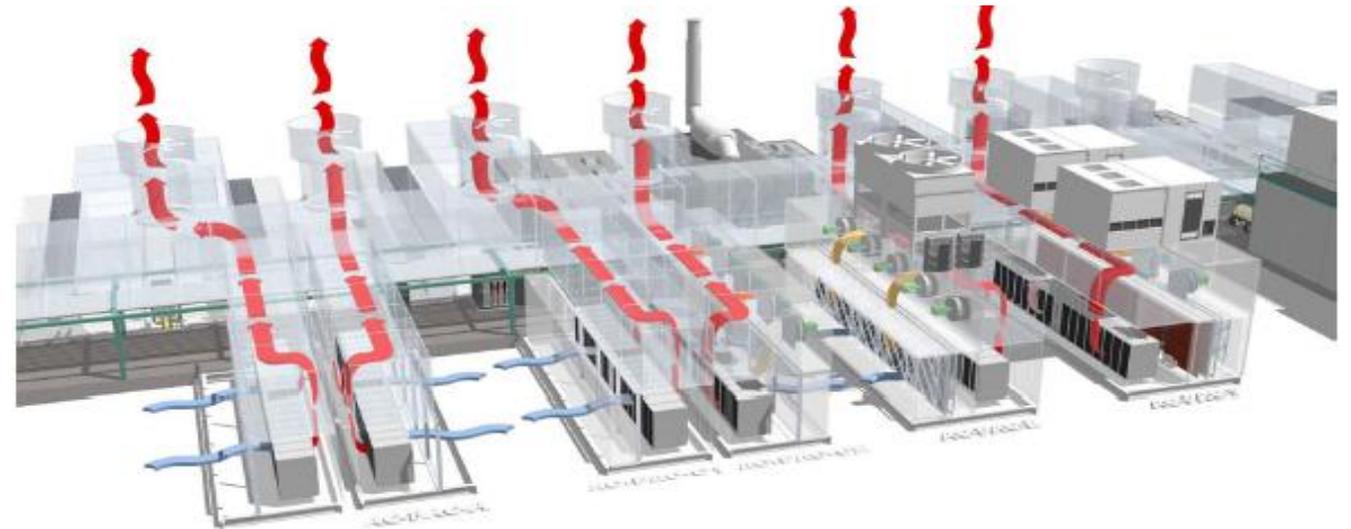
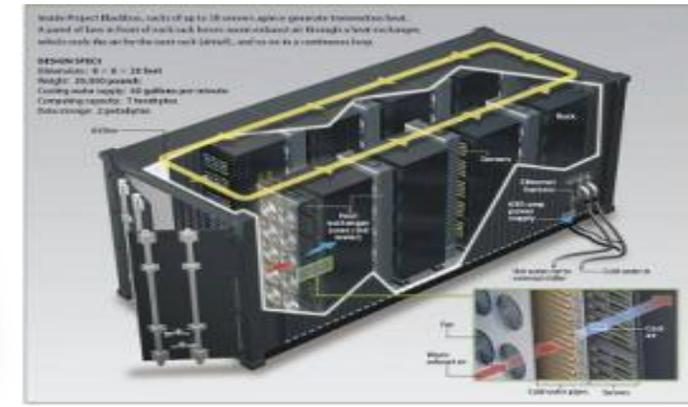
- Seamless interaction is crucial
 - Cloud is the lens that magnifies the power of desktop
 - Persist and share data from client in the cloud
 - Analyze data initially captured in client tools, such as Excel
 - Analysis as a service (SQL, Map-Reduce, R/MatLab).
 - Data visualization generated in the cloud, display on client
 - Provenance, collaboration, other core services...



The Microsoft Cloud

Data Center Infrastructure

- Purpose-built data centers to host containers at large scale
 - Cost \$500 million, 100,000 square foot facility (10 football fields)
- 40 foot shipping containers can house as many as 2,500 servers
 - Density of 10 times amount of compute in equivalent space in traditional data center
- Deliver an average PUE of 1.22
 - Power Usage Effectiveness benchmark from The Green Grid™ consortium on energy efficiency



The Microsoft Cloud

Server Container Deployment



The Microsoft Cloud Server Container Deployment



The Microsoft Cloud

Server Container Deployment



The Microsoft Cloud

Server Container Deployment



The Microsoft Cloud

Server Container Deployment



The Microsoft Cloud

Server Container Deployment



The Microsoft Cloud

Server Container Deployment



Containers: Separating Concerns



The Microsoft Cloud

~100 Globally Distributed Data Centers



Quincy, WA



Chicago, IL



San Antonio, TX



Dublin, Ireland



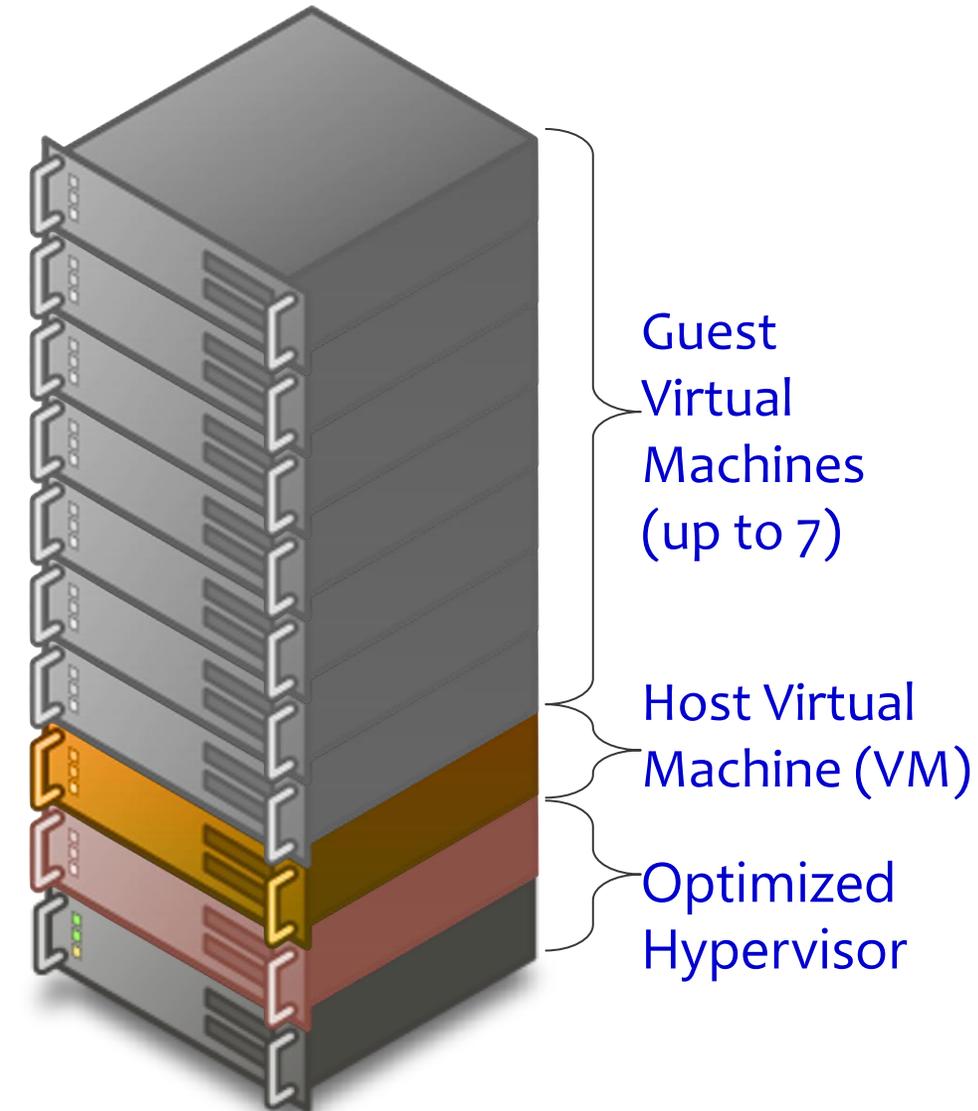
Generation 4 DCs



Windows Azure



Azure Configuration

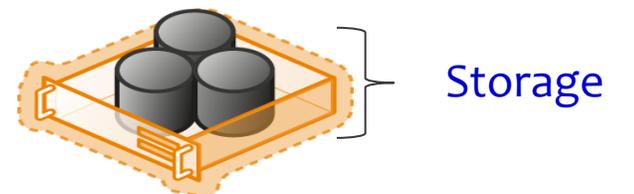
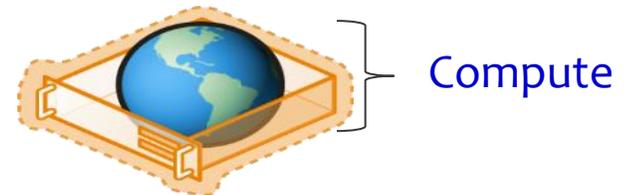


Each Guest VM has

- 1 to 8 CPU cores
- 1.6 GHz x64
- Memory: 1.7-14.2 GB
- Network: 100+ Mbps
- Local: 500GB – 2 TB

Configured with

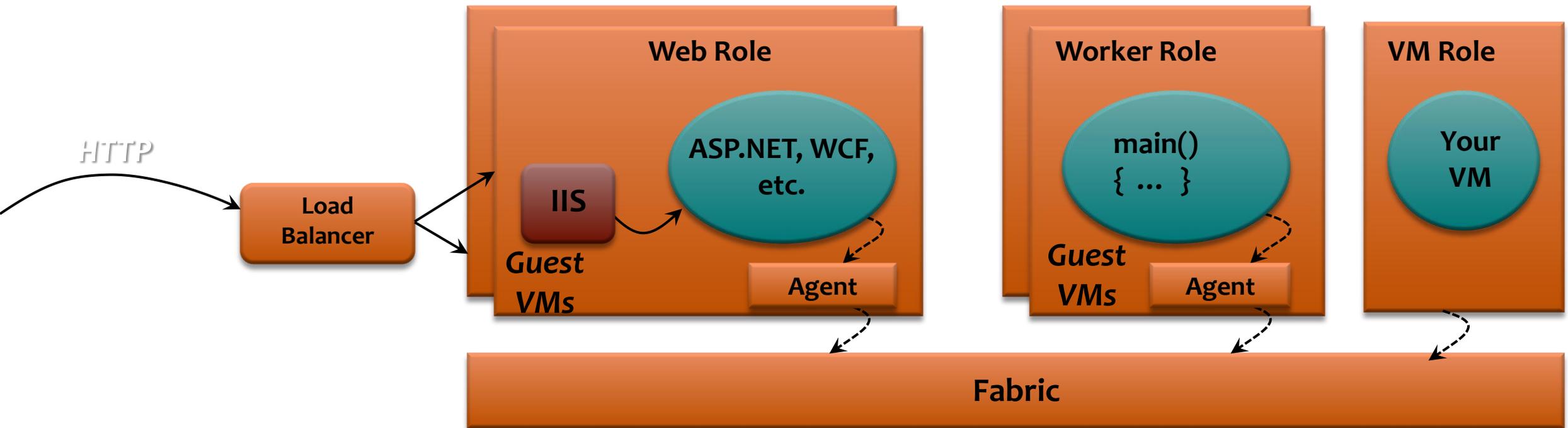
- .NET framework
- IIS 7.0
- 64-bit Windows Server 2008 Enterprise
- Azure platform



Windows Azure Compute Service



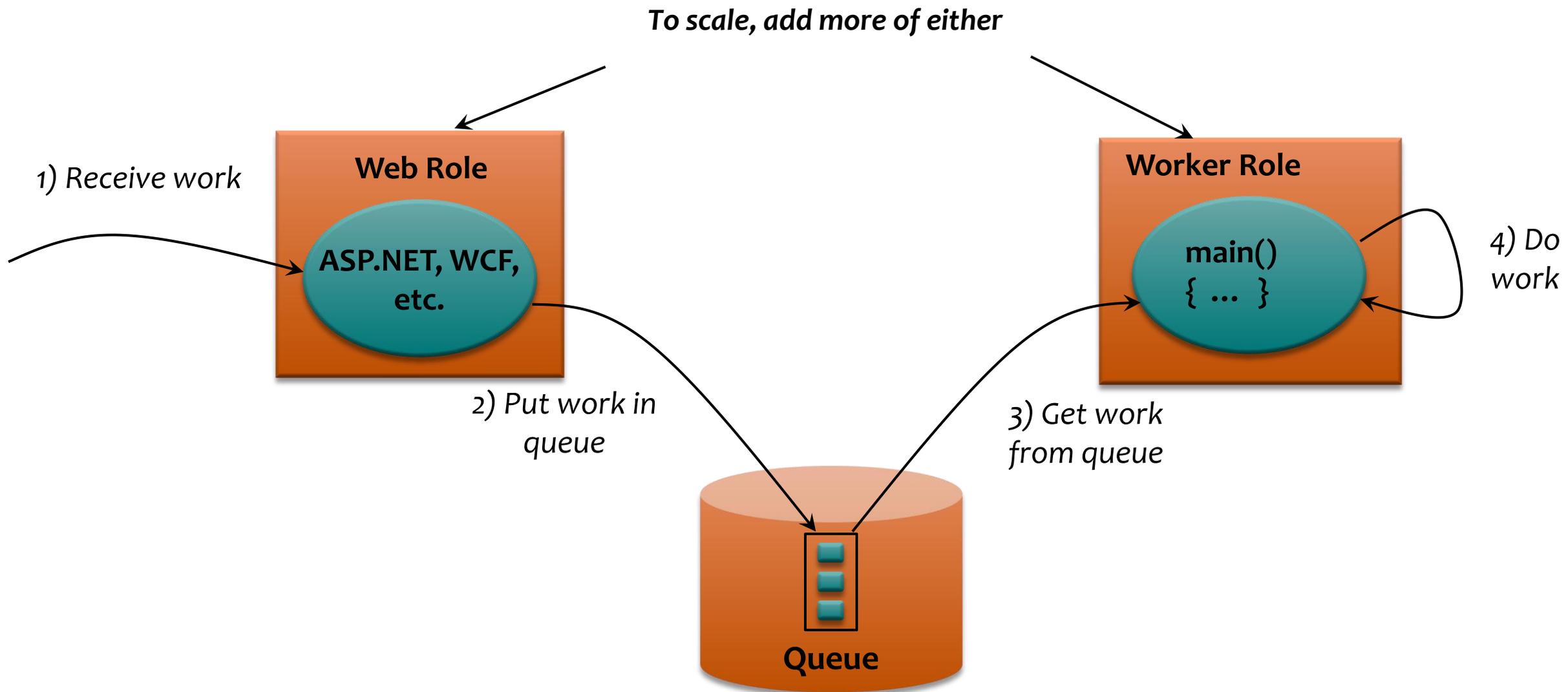
Compute



- Web Role provides client access web presence
- Worker Role does all heavy lifting
- Each can scale independently

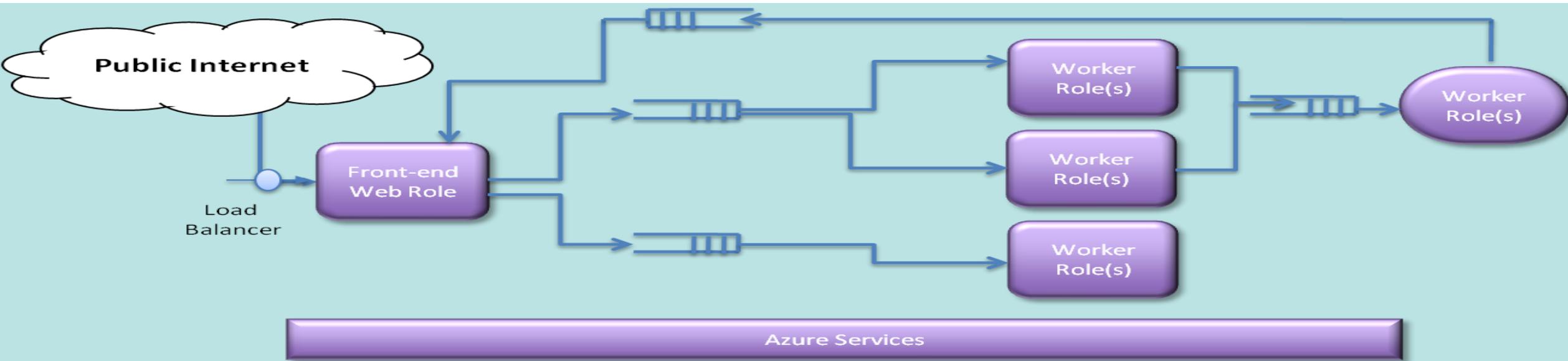
Suggested Application Model

Using queues for reliable messaging

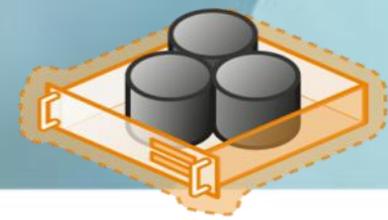


Scalable, Fault Tolerant Applications

- Queues are the application glue for loosely coupled applications
 - **Link** application components, enabling each to scale independently
 - **Resource allocation**, different priority queues and backend servers
 - **Mask faults** in worker roles through reliable messaging and retries
- Use Inter-role communication for performance
 - TCP communication between role instances



Storage in Windows Azure



Storage

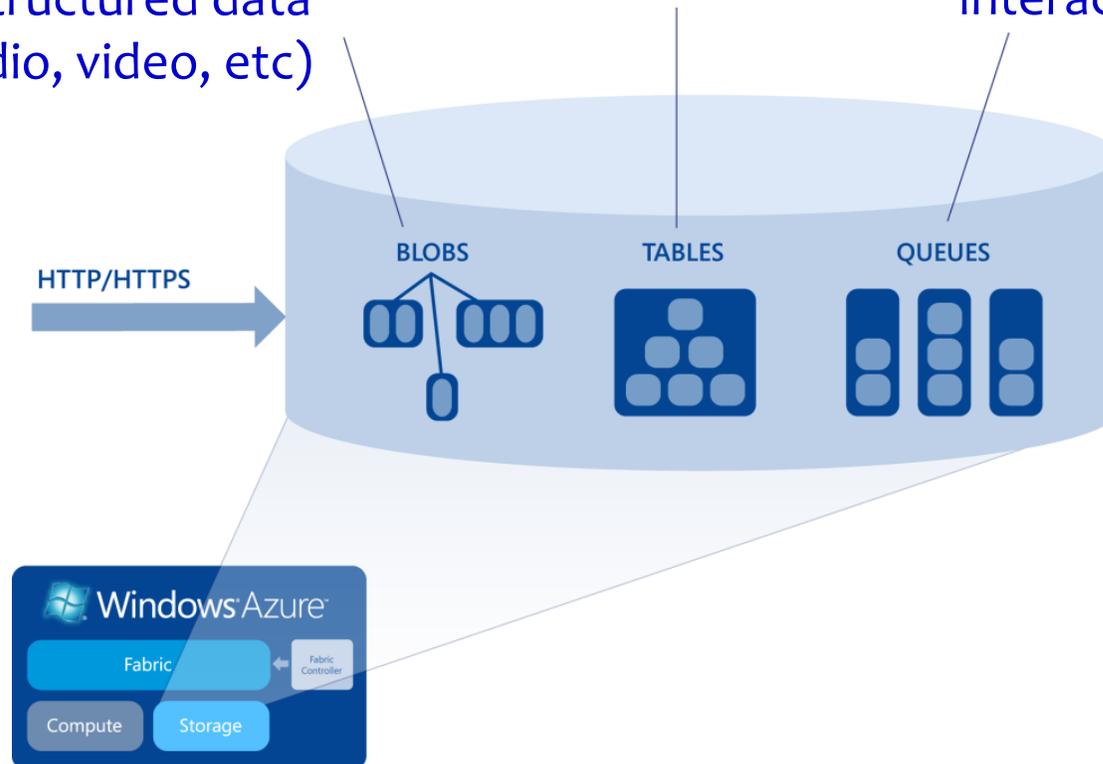
Azure applications can use native storage or SQL Azure

Application state is kept in storage services, so worker roles can replicate as needed

Tables: simply structured data, accessed using ADO.NET Data Services

Queues: serially accessed messages or requests, allowing web-roles and worker-roles to interact

Blobs: large, unstructured data (audio, video, etc)



Windows Azure Platform Basics

What the 'Value Add' ?

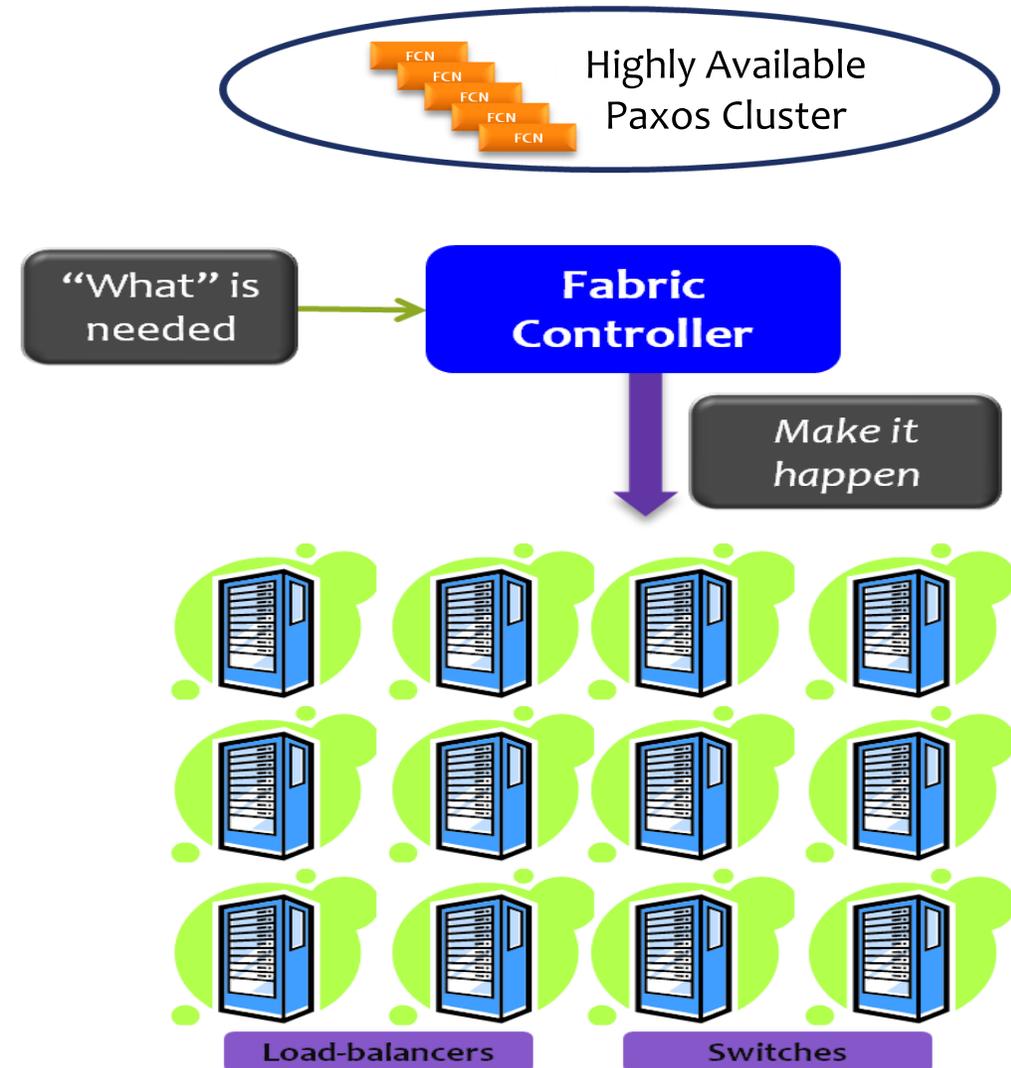
A runtime platform that is *scalable* and *available*

- Services are always running, rolling upgrades/downgrades
- Failure of any node is expected
- Failure of application code is expected, automatic recovery
- Services can grow to be large, provide state management that scales automatically
- Handle dynamic configuration changes due to load or failure
- Manage data center hardware: from CPU cores, nodes, rack, to network infrastructure and load balancers.

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

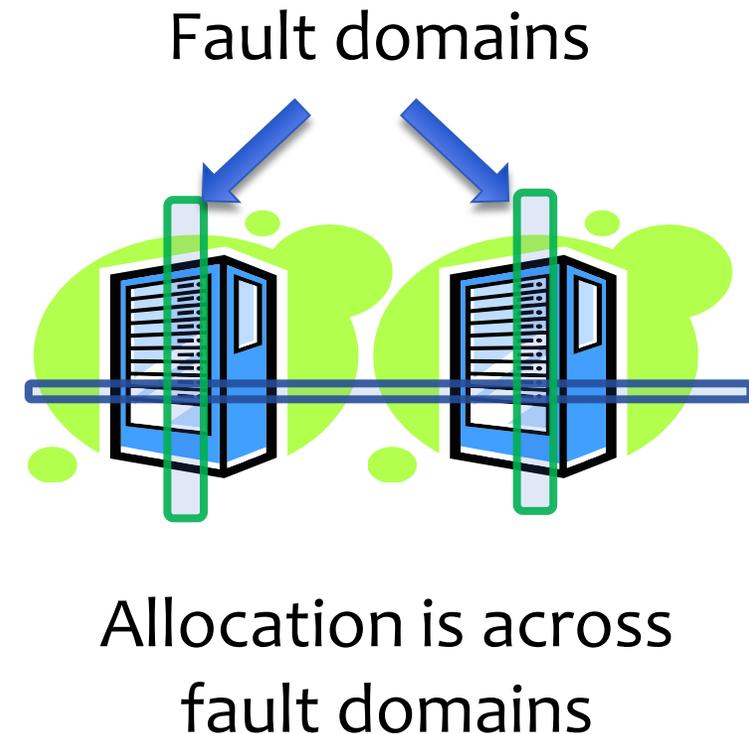
Unit of a failure

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

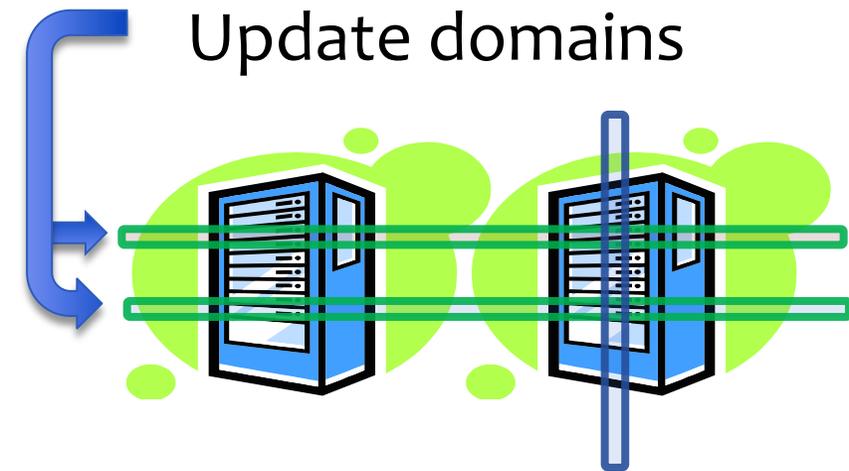


Windows Azure Compute Fabric

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



Allocation is across 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

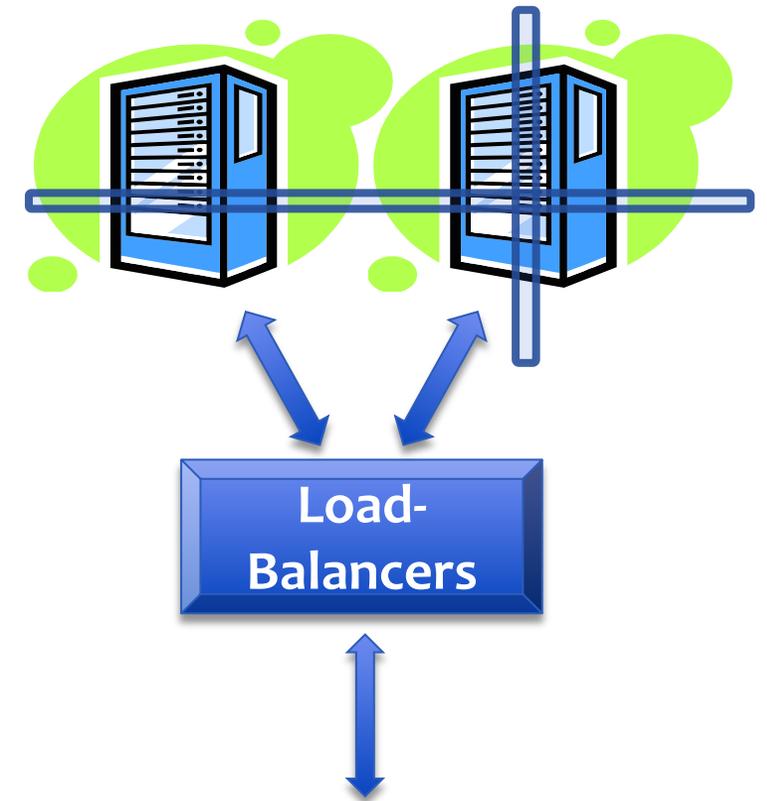
Step 4: Start Roles

Step 5: Configure load-balancers

Step 6: Maintain desired number of roles

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

Allocation across fault
and update domains



AzureBLAST : Biological Sequence Comparison in the Cloud

*In 15 years we'll have all the sequence, a list of the genes everyone has in common and those that differ among people. We know only something like a tenth of 1 percent of the sequence at the moment.
Walter Gilbert*

NCBI BLAST

BLAST (Basic Local Alignment Search Tool)

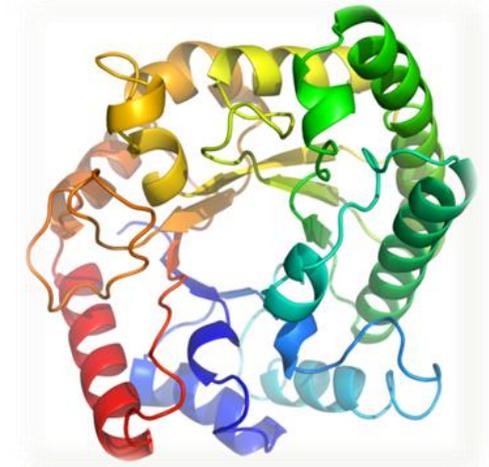
- The most important software in bioinformatics
- Identify similarity between bio-sequences

Computationally intensive

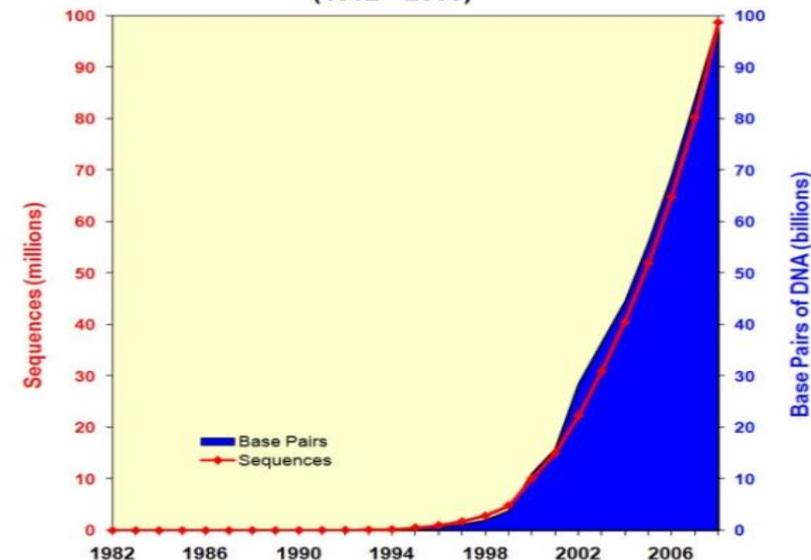
- Large number of pairwise alignment operations
- A normal BLAST running could take 700 ~ 1000 CPU hours
- Size of sequence databases growing exponentially
 - *GenBank doubled in size in about 15 months.*

For most biologists, two choices to run large jobs

- Build a local cluster
- Submit jobs to NCBI or EBI
 - Long job queuing time



Growth of GenBank
(1982 - 2008)



Opportunities for Cloud Computing

It is easy to parallelize BLAST

- Segment the input
 - Segment processing (querying) is pleasingly parallel
- Segment the database (e.g., mpiBLAST)
 - Needs special result reduction processing

Large volume data

- A normal Blast database can be as large as 10GB
 - 100 nodes means the peak storage bandwidth could reach to 1TB
- The output of BLAST is usually 10-100x larger than the input

AzureBLAST

- Parallel BLAST engine on Azure
- Query-segmentation data-parallel pattern
 - split the input sequences
 - query partitions in parallel
 - merge results together when done
- Follows the general suggested application model
 - Web Role + Queue + Worker
- With three special considerations
 - Batch job management
 - Task parallelism on an elastic Cloud
 - Large data-set management

AzureBLAST Task-Flow

A simple Split/Join pattern

Leverage multi-core of one instance

- argument “**-a**” of NCBI-BLAST
- 1,2,4,8 for small, middle, large, and extra large instance size

Task granularity

- Large partition → load imbalance
- Small partition → unnecessary overheads
 - NCBI-BLAST overhead
 - Data transferring overhead.

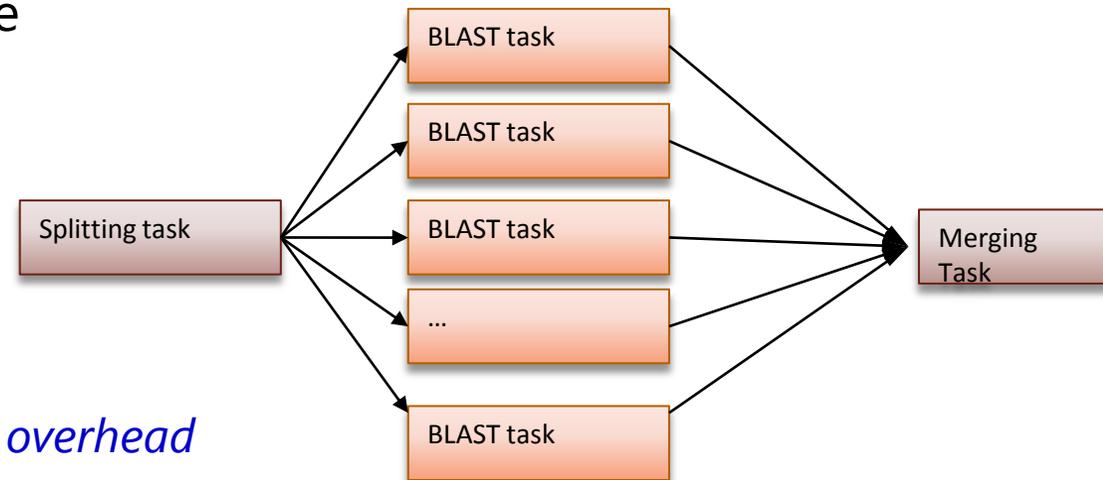
Best Practice: *test runs to profiling and set size to mitigate the overhead*

Value of **visibilityTimeout** for each BLAST task,

- Essentially an estimate of the task run time.
- too small → repeated computation;
- too large → unnecessary long period of waiting time in case of the instance failure.

Best Practice:

- *Estimate the value based on the number of pair-bases in the partition and test-runs*
- *Watch out for the 2-hour maximum limitation*



Micro-Benchmarks Inform Design

Task size vs. Performance

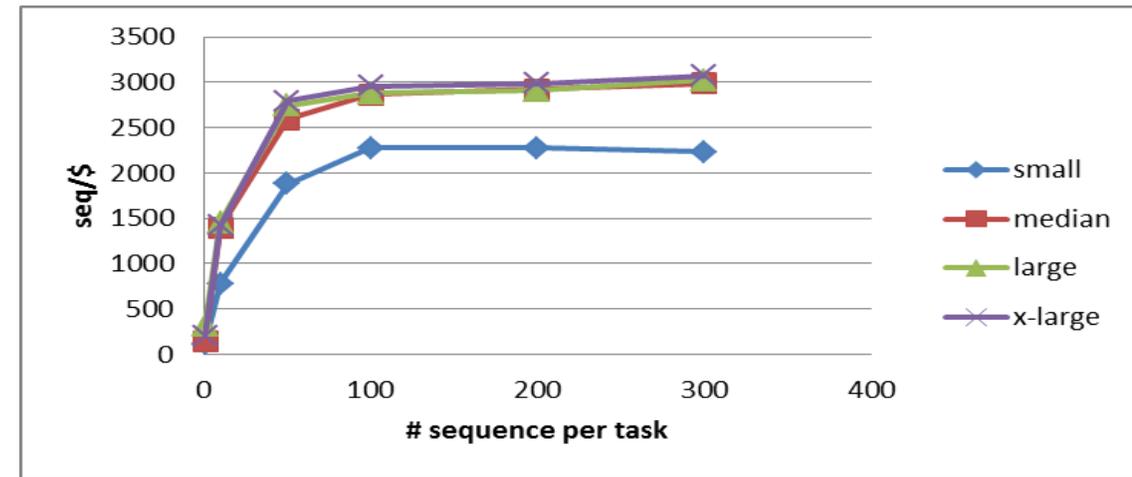
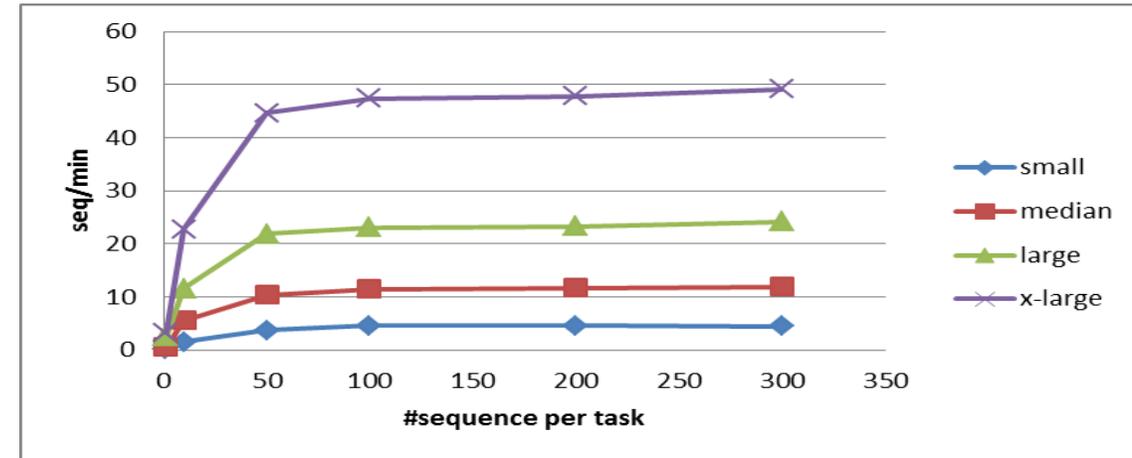
- Benefit of the warm cache effect
- 100 sequences per partition is the best choice

Instance size vs. Performance

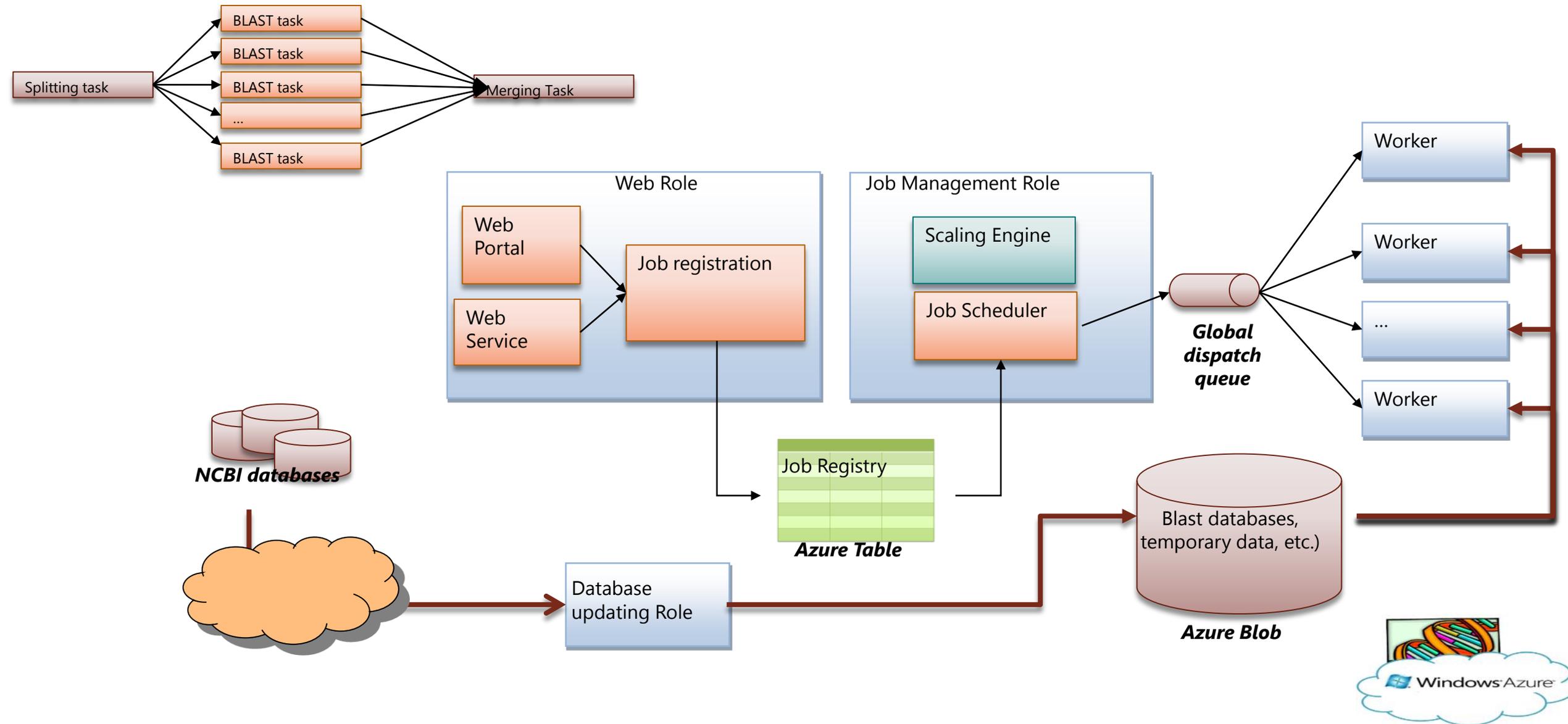
- **Super-linear** speedup with larger size worker instances
- Primarily due to the memory capability.

Task Size/Instance Size vs. Cost

- Extra-large instance generated the best and the most economical throughput
- Fully utilize the resource



AzureBLAST



AzureBLAST Job Portal

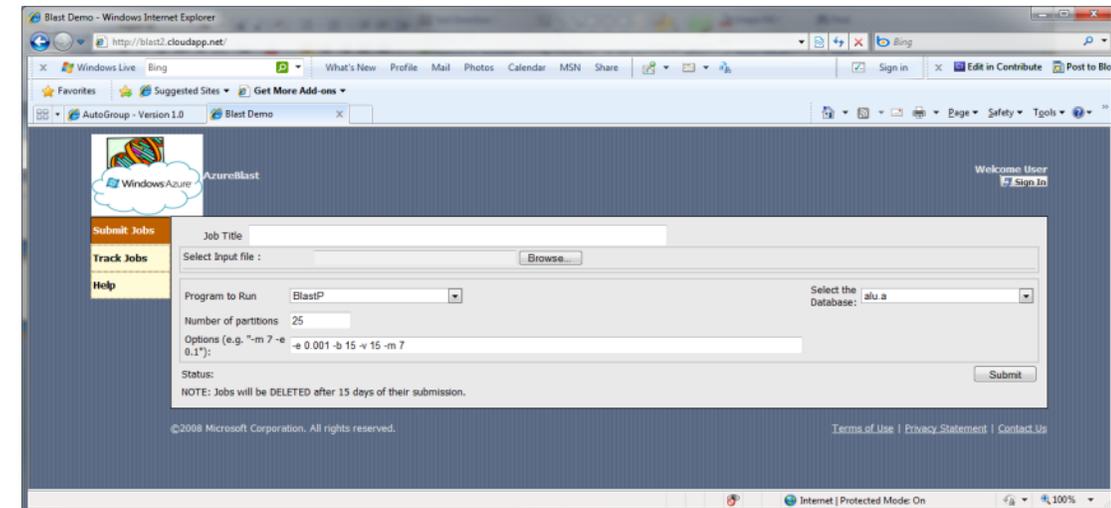
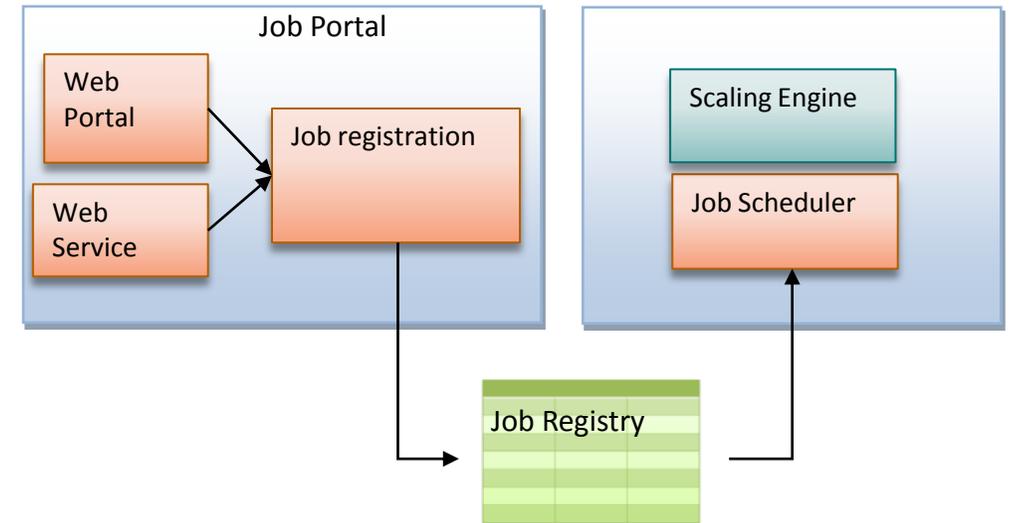
ASP.NET program hosted by a web role instance

- Submit jobs
- Track job's status and logs

Authentication/Authorization based on Live ID

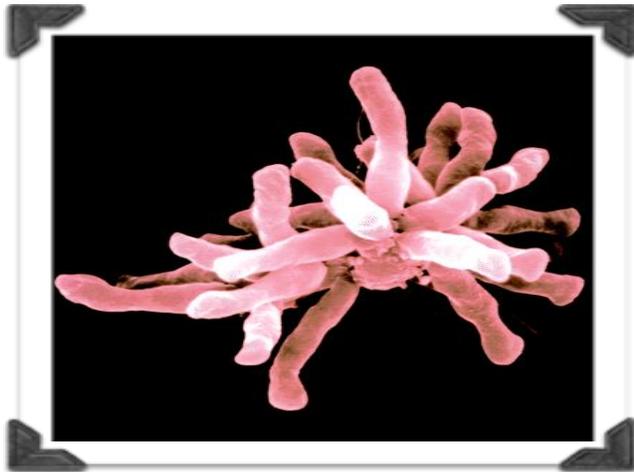
The accepted job is stored into the job registry table

- Fault tolerance, avoid in-memory states

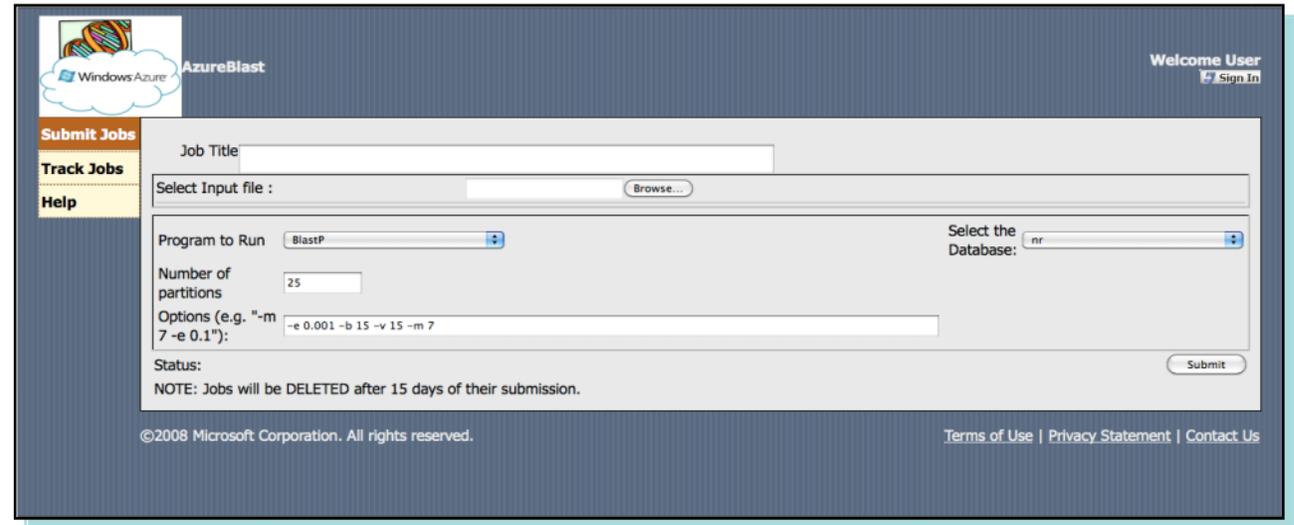


R. palustris as a platform for H₂ production

Eric Shadt, SAGE



Sam Phattarasukol Harwood Lab, UW



Blasted ~5,000 proteins (700K sequences)

- Against all NCBI non-redundant proteins: completed in 30 min
- Against ~5,000 proteins from another strain: completed in less than 30 sec

AzureBLAST significantly saved computing time...

All-Against-All Experiment

Discovering Homologs

- Discover the interrelationships of known protein sequences

“All against All” query

- The database is also the input query
- The protein database is large (4.2 GB size)
 - *Totally 9,865,668 sequences to be queried*
- Theoretically, 100 billion sequence comparisons!

Performance estimation

- Based on the sampling-running on one extra-large Azure instance
- Would require 3,216,731 minutes (6.1 years) on one desktop

One of biggest BLAST jobs as far as we know

- This scale of experiments usually are infeasible to most scientists

Our Approach

- Allocated a total of ~4000 instances
 - 475 extra-large VMs (8 cores per VM), four datacenters, US (2), Western and North Europe
- 8 deployments of AzureBLAST
 - Each deployment has its own co-located storage service
- Divide 10 million sequences into multiple segments
 - Each will be submitted to one deployment as one job for execution
 - Each segment consists of smaller partitions
- When load imbalances, redistribute the load *manually*



Understanding Azure by analyzing logs

A normal log record should be

3/31/2010 6:14	RD00155D3611B0	Executing the task 251523...
3/31/2010 6:25	RD00155D3611B0	Execution of task 251523 is done, it took 10.9mins
3/31/2010 6:25	RD00155D3611B0	Executing the task 251553...
3/31/2010 6:44	RD00155D3611B0	Execution of task 251553 is done, it took 19.3mins
3/31/2010 6:44	RD00155D3611B0	Executing the task 251600...
3/31/2010 7:02	RD00155D3611B0	Execution of task 251600 is done, it took 17.27 mins

Otherwise, something is wrong (e.g., task failed to complete)

3/31/2010 8:22	RD00155D3611B0	Executing the task 251774...
3/31/2010 9:50	RD00155D3611B0	Executing the task 251895...
3/31/2010 11:12	RD00155D3611B0	Execution of task 251895 is done, it took 82 mins

Surviving System Upgrades

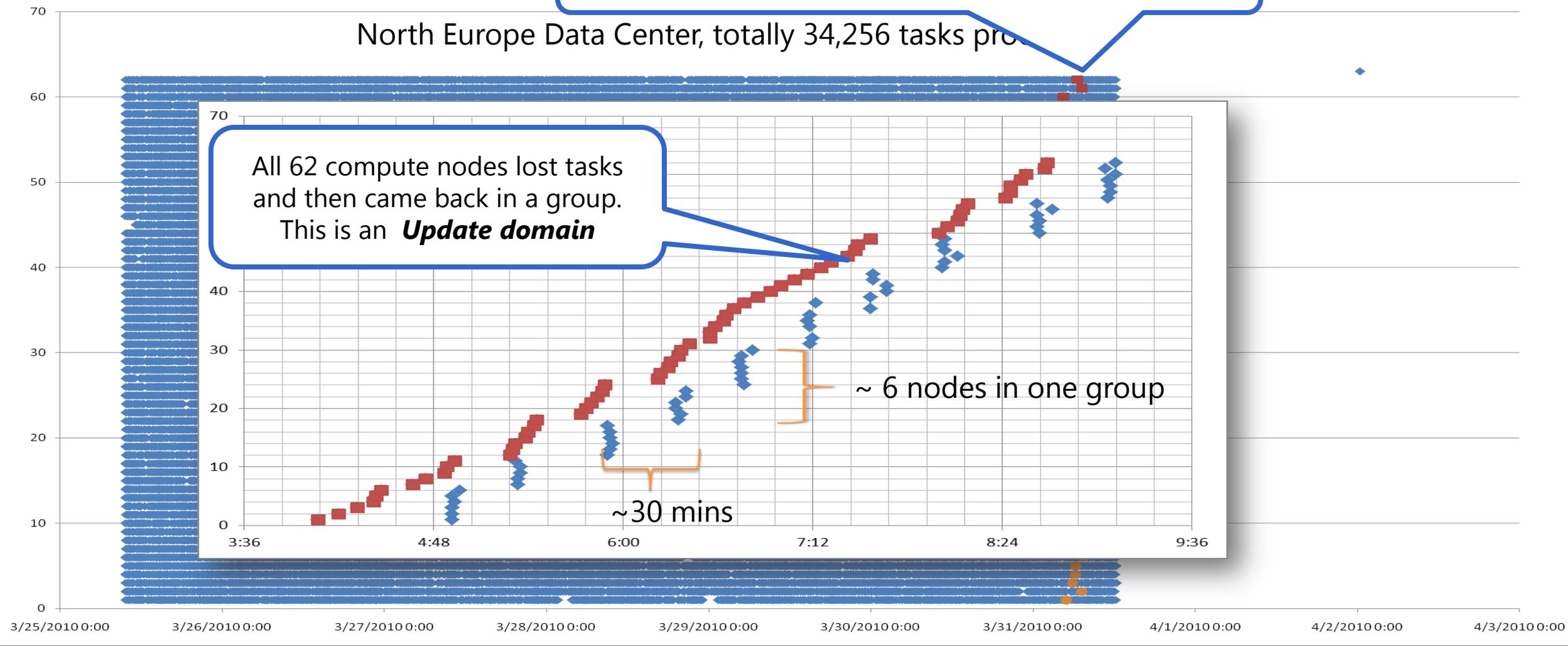
3/31/2010 8:22 1 RD00155D3611B0 Executing the Part 251774...
3/31/2010 9:50 1 RD00155D3611B0 Executing the Part 251895...
3/31/2010 11:12 1 RD00155D3611B0 Execution of Part 251895 is done, it takes 82.3911189933333 mins

North Europe Data Center, totally 34,256 tasks pro

All 62 compute nodes lost tasks and then came back in a group.
This is an **Update domain**

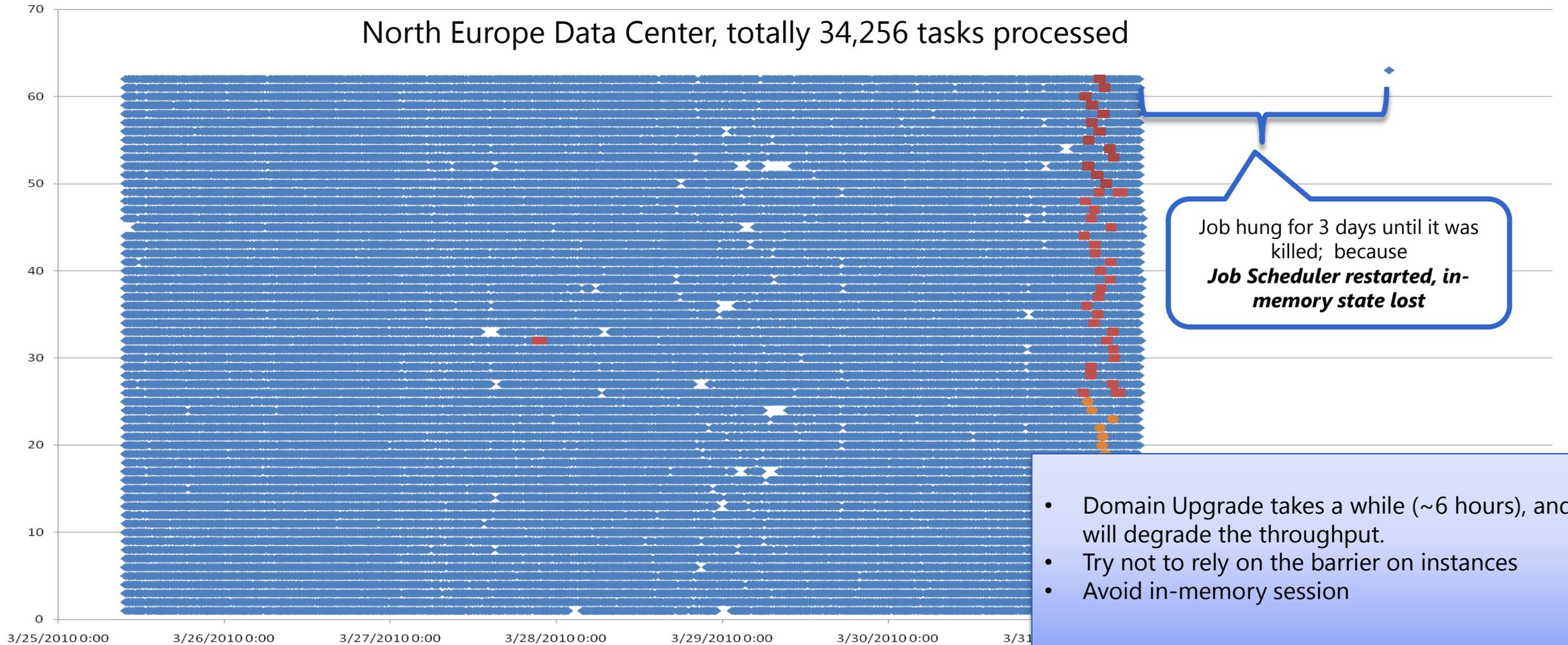
~ 6 nodes in one group

~30 mins



Surviving System Upgrades

North Europe Data Center, totally 34,256 tasks processed

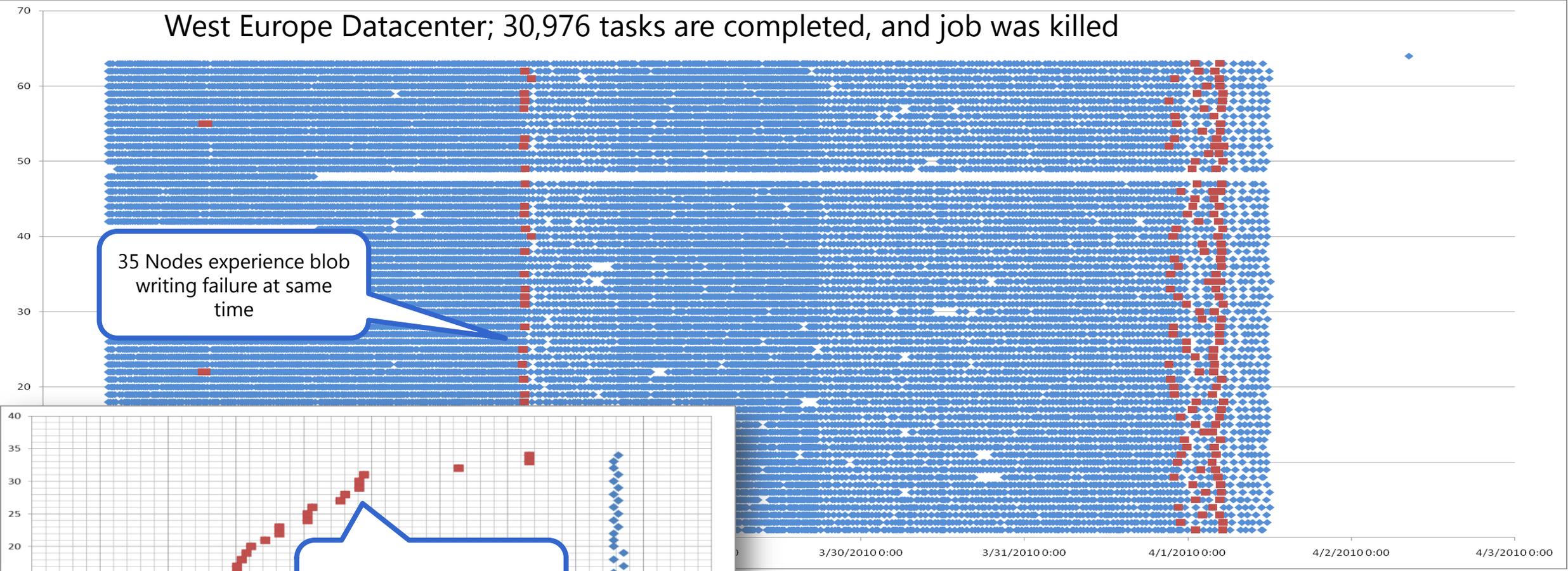


Job hung for 3 days until it was killed; because ***Job Scheduler restarted, in-memory state lost***

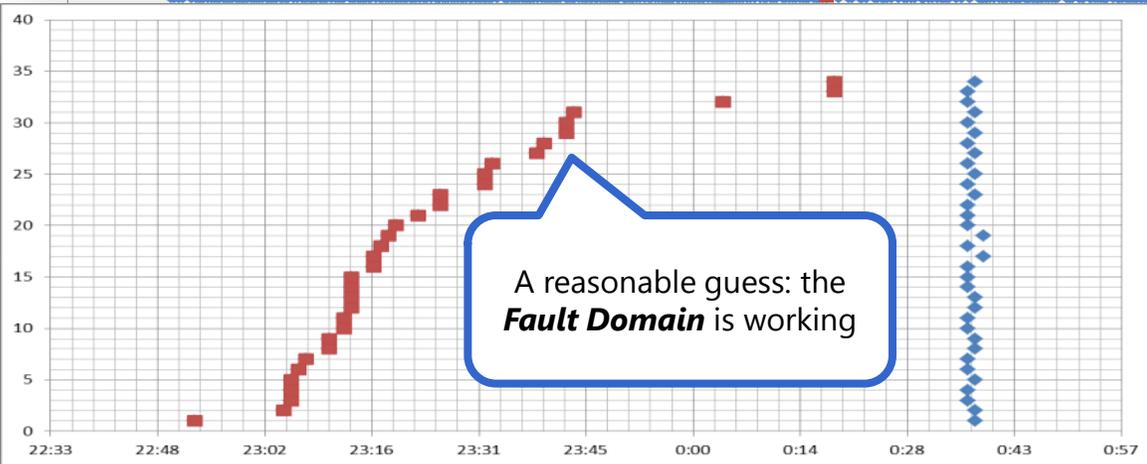
- Domain Upgrade takes a while (~6 hours), and will degrade the throughput.
- Try not to rely on the barrier on instances
- Avoid in-memory session

Surviving Storage Failures

West Europe Datacenter; 30,976 tasks are completed, and job was killed



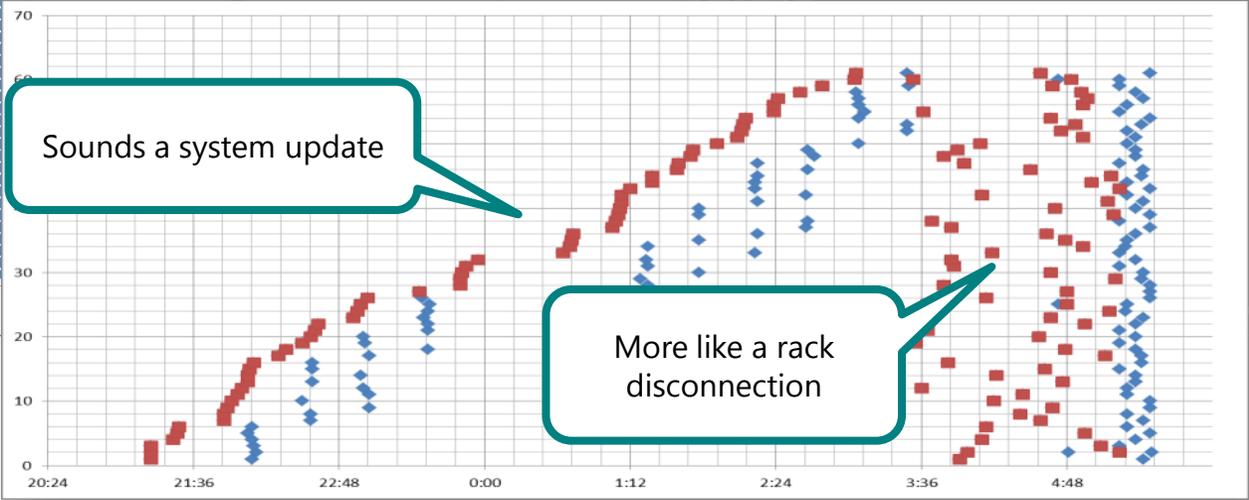
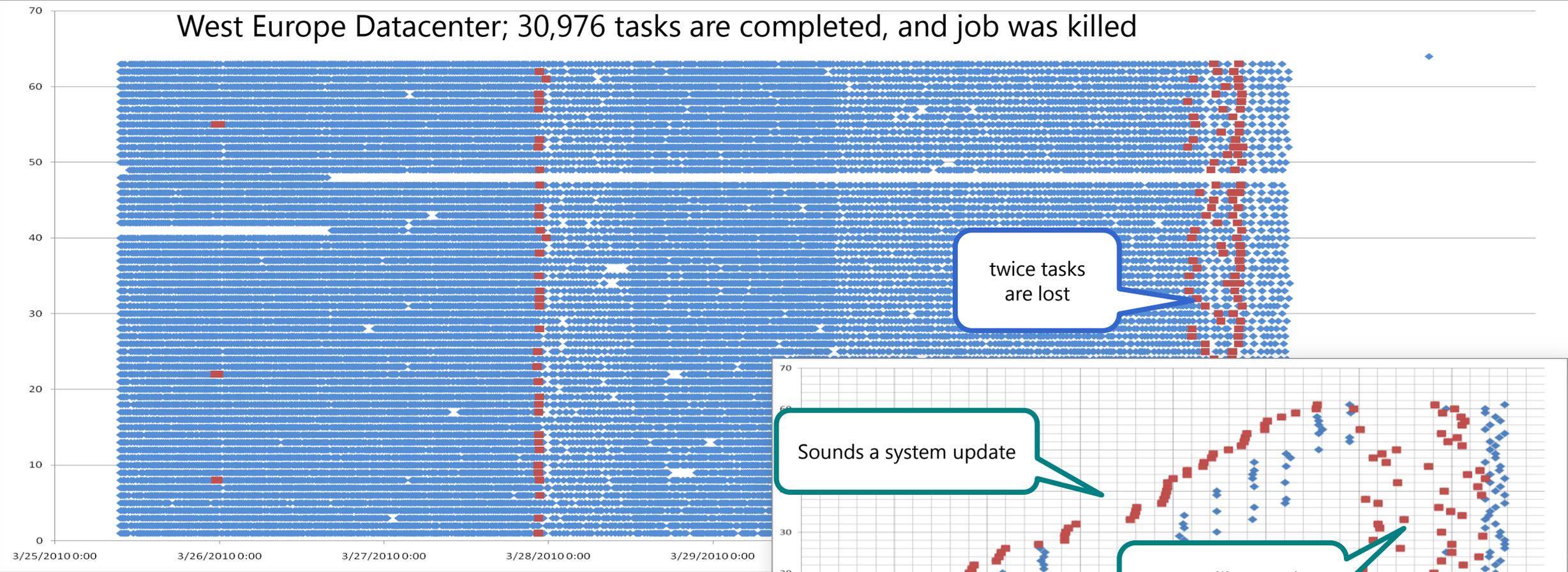
35 Nodes experience blob writing failure at same time



A reasonable guess: the **Fault Domain** is working

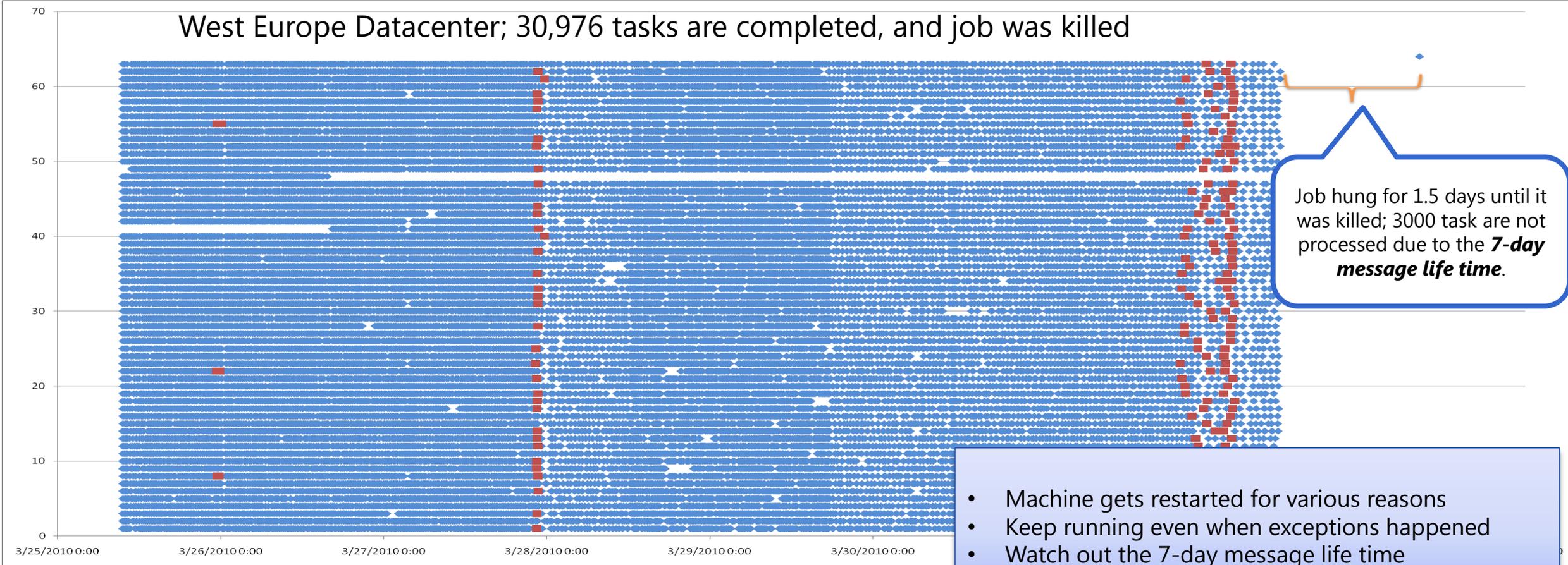
Surviving Storage Failures

West Europe Datacenter; 30,976 tasks are completed, and job was killed



Surviving Storage Failures

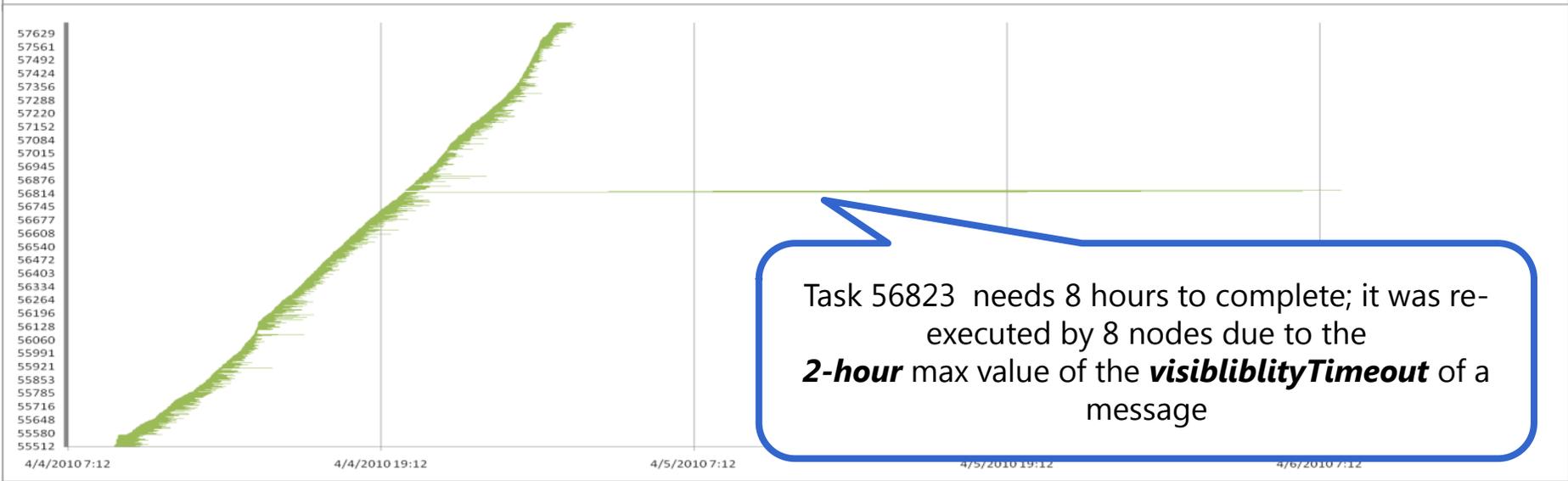
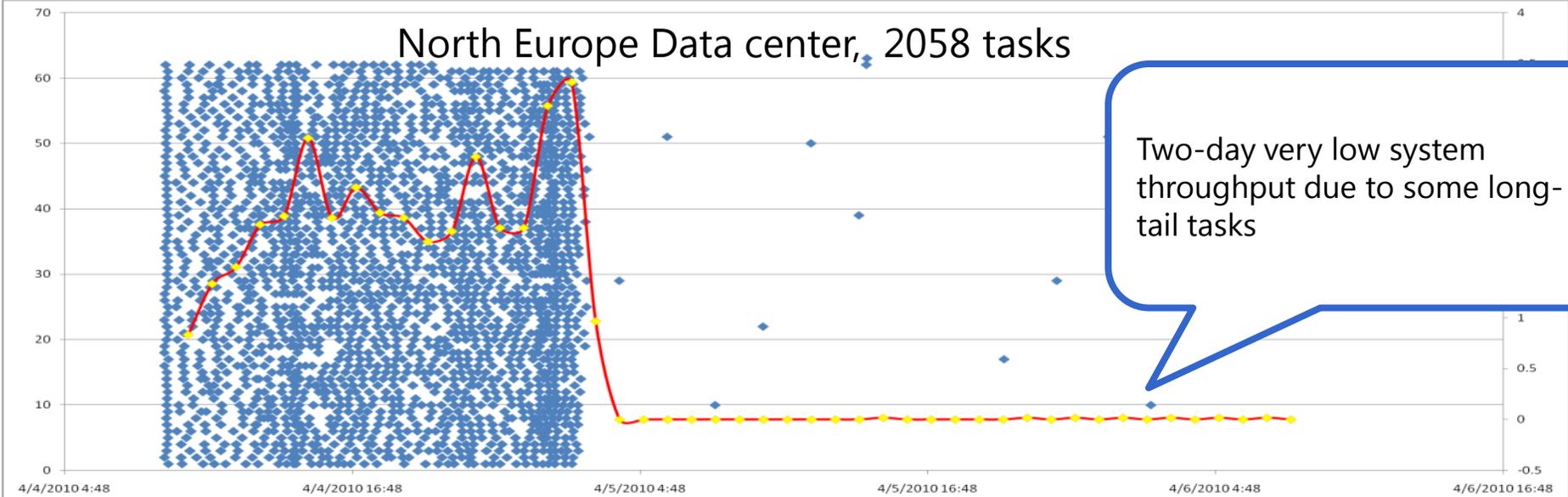
West Europe Datacenter; 30,976 tasks are completed, and job was killed



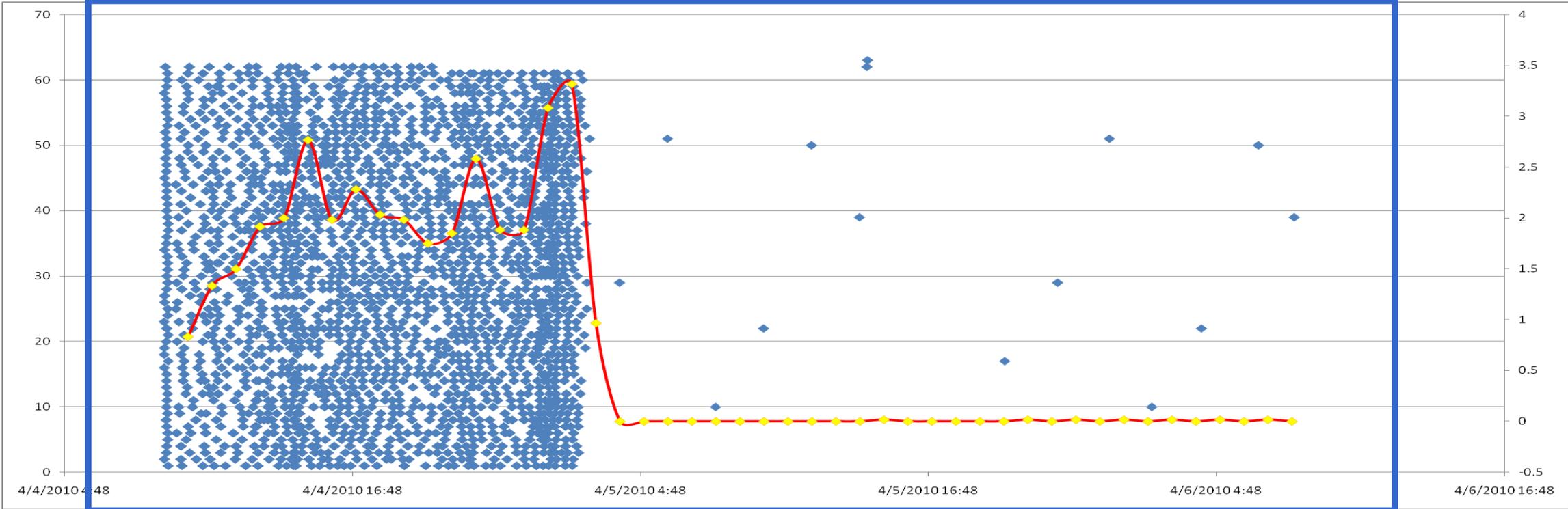
Job hung for 1.5 days until it was killed; 3000 task are not processed due to the **7-day message life time**.

- Machine gets restarted for various reasons
- Keep running even when exceptions happened
- Watch out the 7-day message life time
- An auto-scaling feature can save money

Load Imbalance

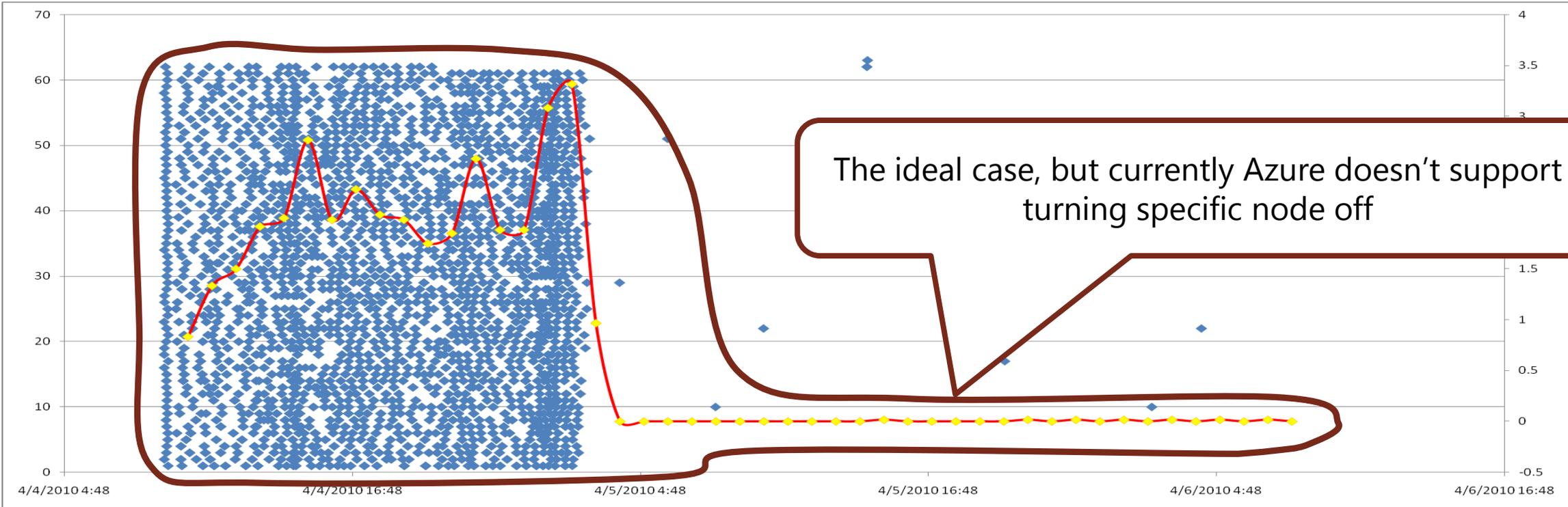


Load Imbalance



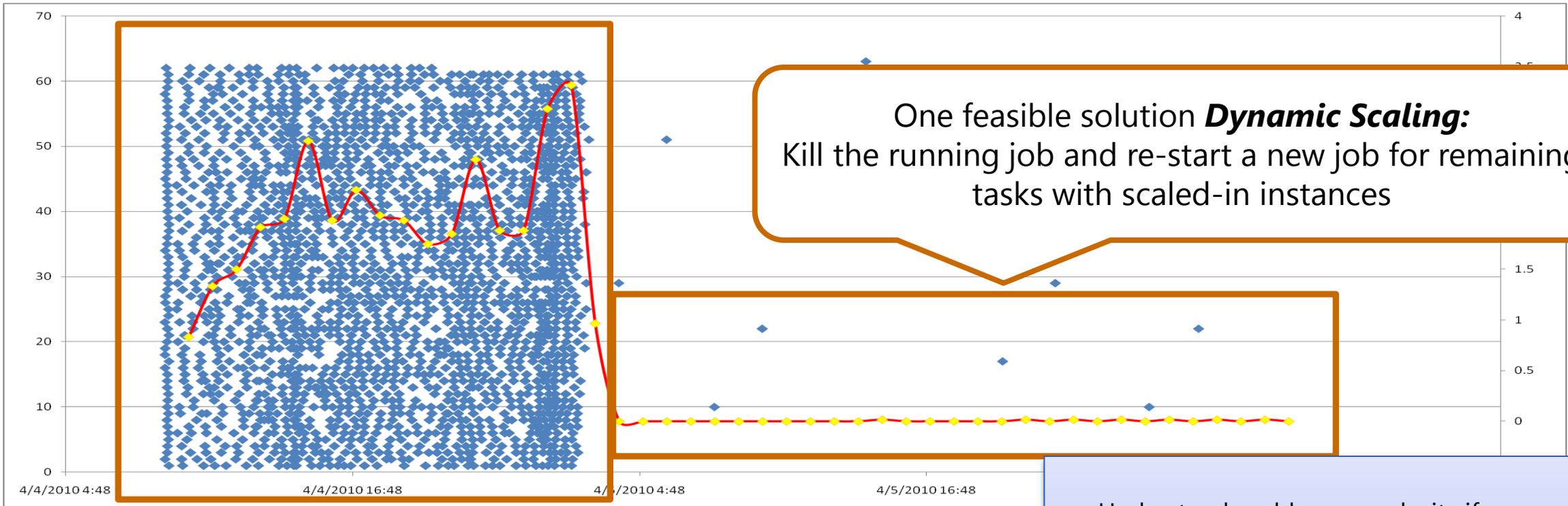
The bounding box is what we were actually charged.

Load Imbalance



The ideal case, but currently Azure doesn't support turning specific node off

Load Imbalance



- Understand problem complexity if possible, A test running is quite useful
- If it is impossible (e.g., BLAST), leveraging the elasticity of cloud to save the cost

MODIS Azure : Computing Evapotranspiration (ET) in The Cloud

You never miss the water till the well has run dry
Irish Proverb

Computing Evapotranspiration (ET)

Evapotranspiration (ET) is the release of water to the atmosphere by evaporation from open water bodies and transpiration, or evaporation through plant membranes, by plants.

$$ET = \frac{\Delta R_n + \rho_a c_p (\delta q) g_a}{(\Delta + \gamma(1 + g_a/g_s)) \lambda_v}$$

Penman-Monteith (1964)

ET = Water volume evapotranspired ($\text{m}^3 \text{s}^{-1} \text{m}^{-2}$)

Δ = Rate of change of saturation specific humidity with air temperature. (Pa K^{-1})

λ_v = Latent heat of vaporization (J/g)

R_n = Net radiation (W m^{-2})

c_p = Specific heat capacity of air ($\text{J kg}^{-1} \text{K}^{-1}$)

ρ_a = dry air density (kg m^{-3})

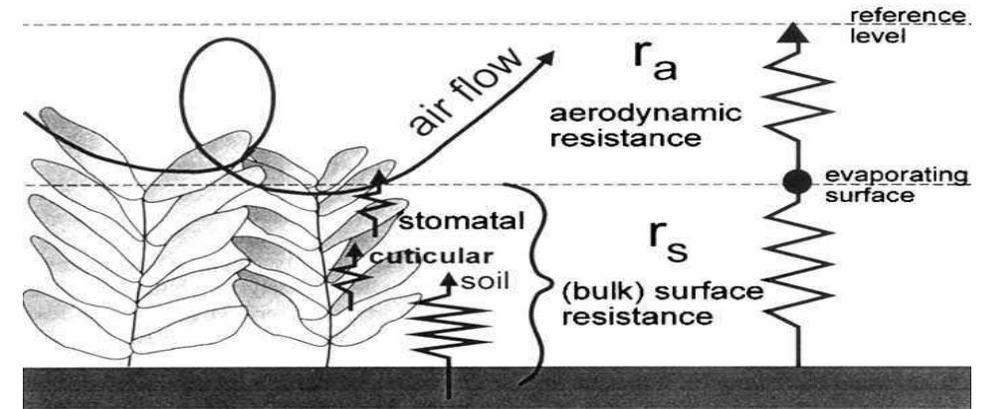
δq = vapor pressure deficit (Pa)

g_a = Conductivity of air (inverse of r_a) (m s^{-1})

g_s = Conductivity of plant stoma, air (inverse of r_s) (m s^{-1})

γ = Psychrometric constant ($\gamma \approx 66 \text{ Pa K}^{-1}$)

- Lots of inputs : big reduction
- Some of the inputs are not so simple

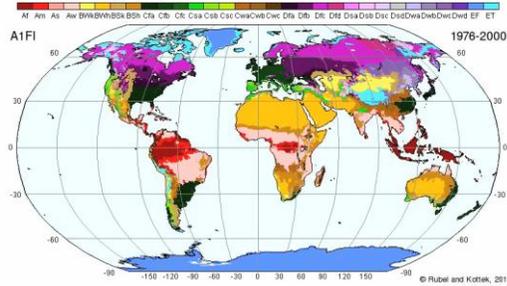


Estimating resistance/conductivity across a catchment can be tricky

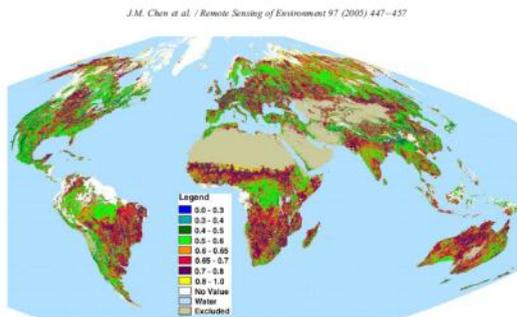


ET Synthesizes Imagery, Sensors, Models and Field Data

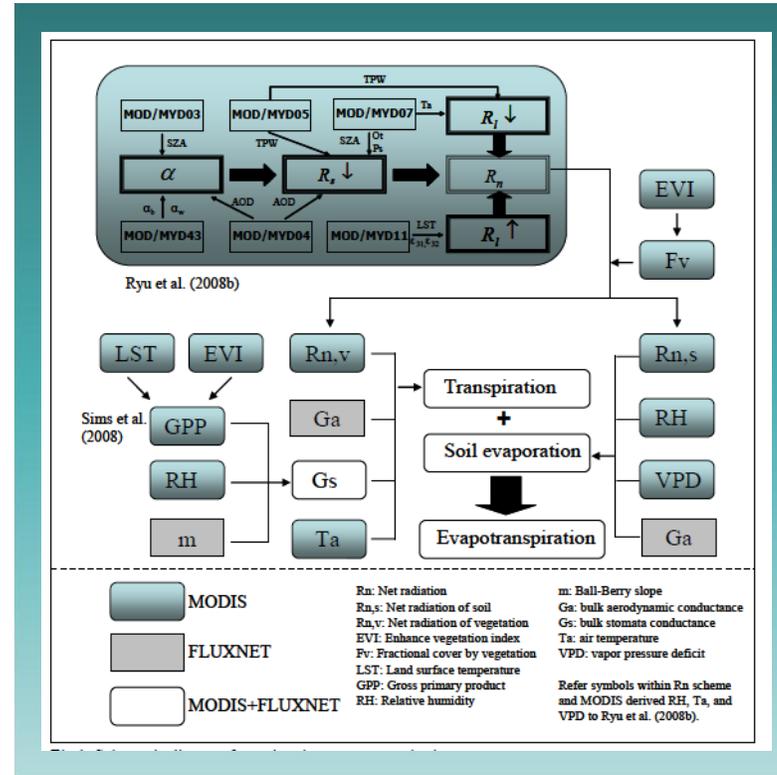
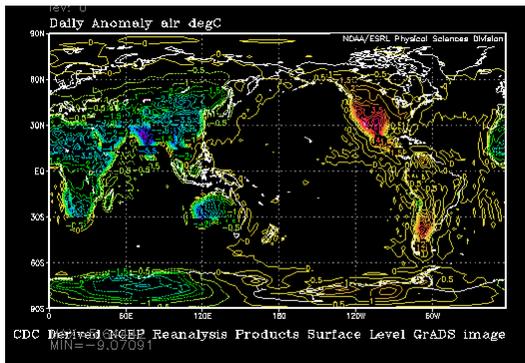
Climate classification
~1MB (1file)



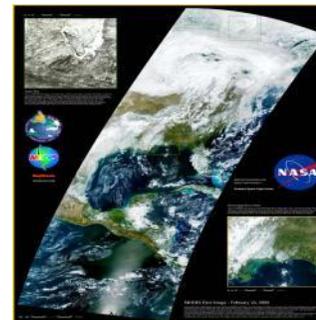
Vegetative clumping
~5MB (1file)



NCEP/NCAR
~100MB
(4K files)



NASA MODIS
imagery source
archives
5 TB (600K files)



20 US year = 1 global year

FLUXNET curated sensor dataset
(30GB, 960 files)



FLUXNET curated field dataset
2 KB (1 file)



MODIS Azure: Four Stage Image Processing Pipeline

Data collection (**map**) stage

- Downloads requested input tiles from NASA ftp sites
- Includes geospatial lookup for non-sinusoidal tiles that will contribute to a reprojected sinusoidal tile

Reprojection (**map**) stage

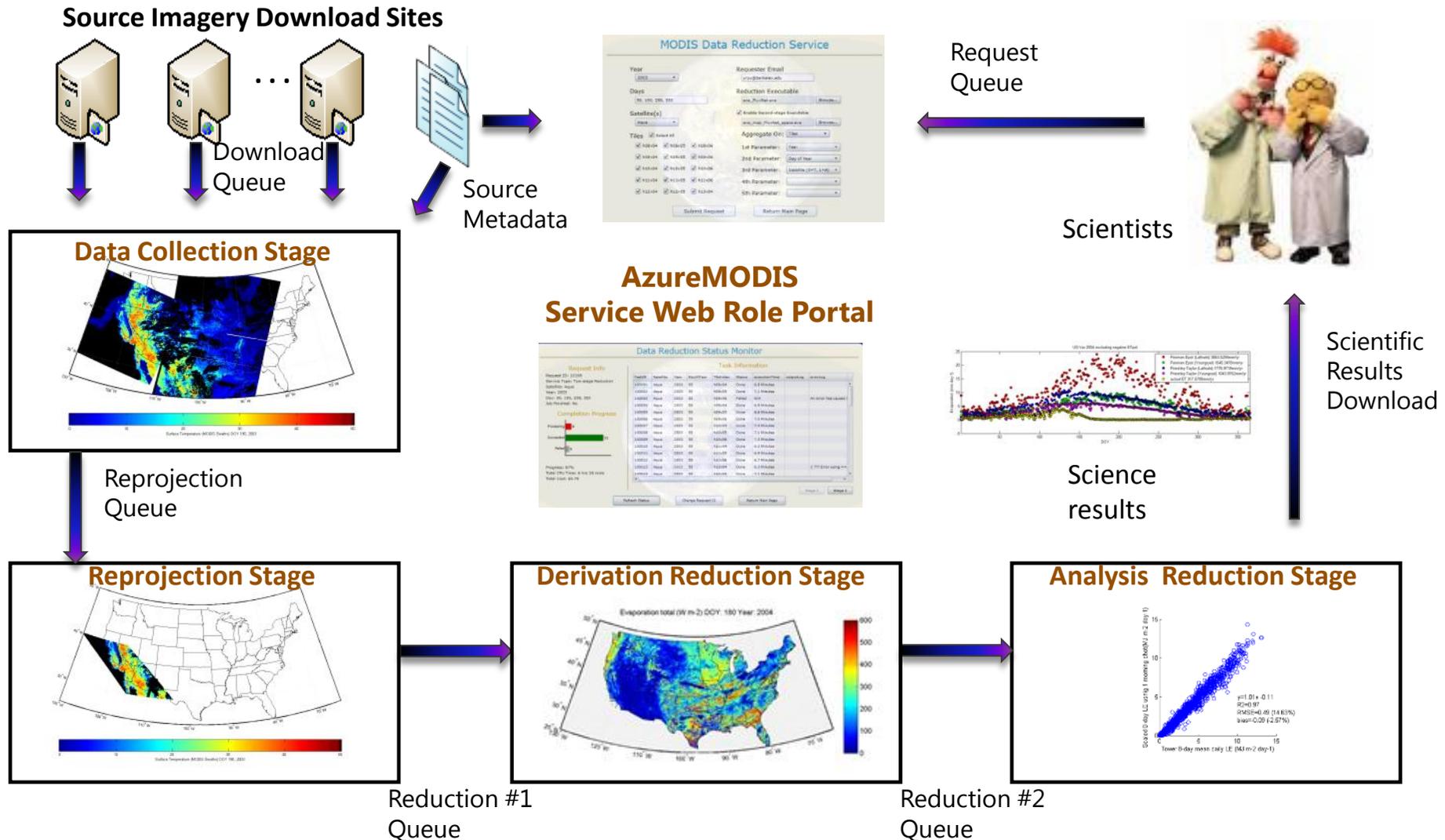
- Converts source tile(s) to intermediate result sinusoidal tiles
- Simple nearest neighbor or spline algorithms

Derivation **reduction** stage

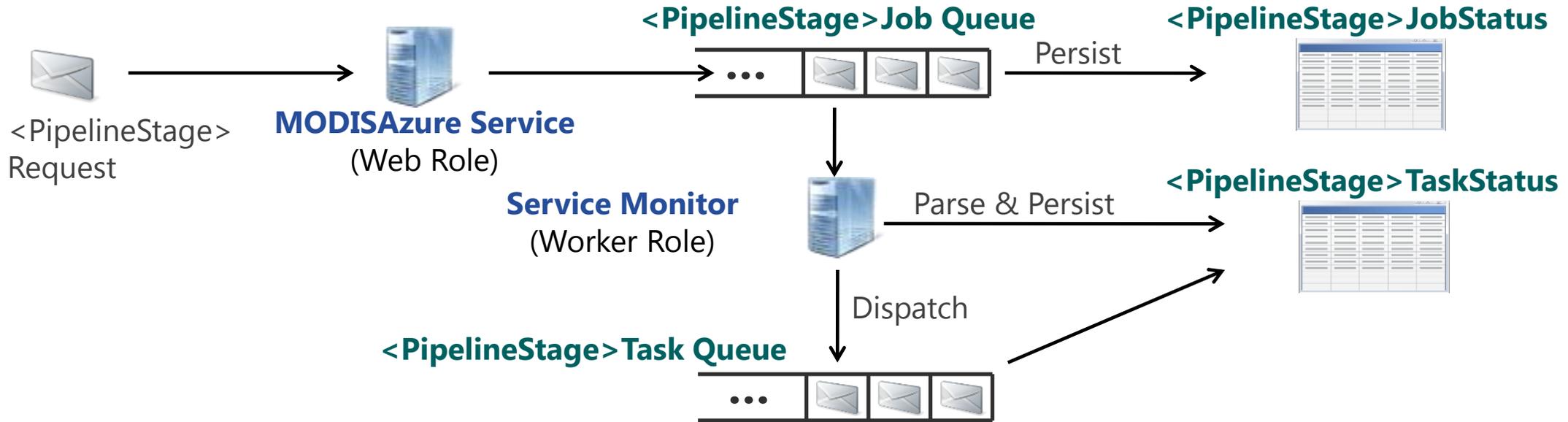
- First stage visible to scientist
- Computes ET in our initial use

Analysis **reduction** stage

- Optional second stage visible to scientist
- Enables production of science analysis artifacts such as maps, tables, virtual sensors

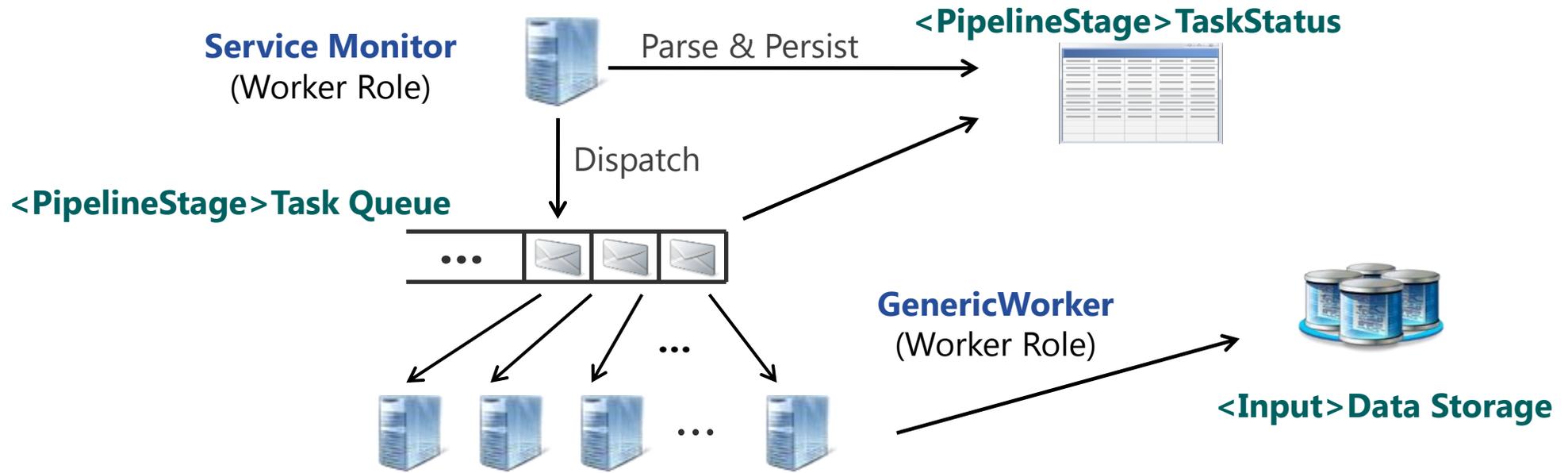


MODIS Azure: Architectural Big Picture (1/2)



- **ModisAzure Service** is the Web Role front door
 - Receives all user requests
 - Queues request to appropriate Download, Reprojection, or Reduction Job Queue
- **Service Monitor** is a dedicated Worker Role
 - Parses all job requests into tasks – recoverable units of work
 - Execution status of all jobs and tasks persisted in Tables

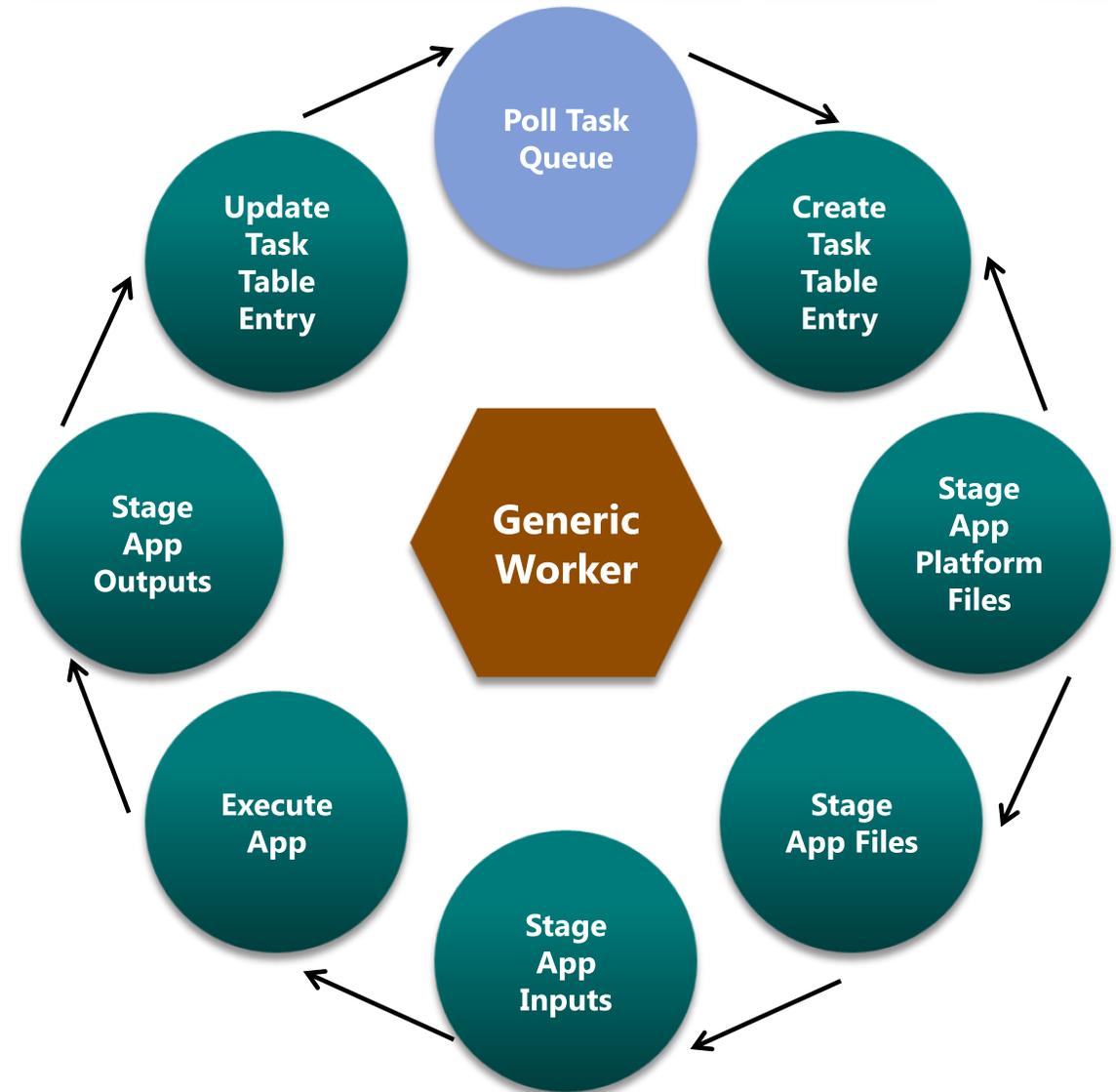
MODIS Azure: Architectural Big Picture (2/2)



- All work actually done by a **GenericWorker** Worker Role
 - Dequeues tasks created by the Service Monitor
 - Retries failed tasks 3 times
 - Maintains all task status
 - Sandboxes science or other executable
 - Marshalls all storage from/to Azure blob storage to/from local Azure Worker instance files

Inside A Generic Worker

- Manages application sandbox
 - Ensures all application binaries such as the MatLab runtime are installed for “known” application types
 - Stages all input blobs from Azure storage to local files
 - Passes any marshalled inputs to uploaded application binary
 - Stages all output blobs to Azure storage from local files
 - Preserves any marshalled outputs to the appropriate Task table
- Simplifies desktop development and cloud deployment



Storage Management



Source

- Original **source** image download
- Can be deleted when all dependent reprojections complete

- **Reprojection** results

- May include the same target tile at different spatial resolution



Reprojection Storage

- **Reduction** results

- Older results can be aged out over time
- A zip file blob is created for each job to simplify download



Reduction Storage

- **Metadata** includes geospatial lookup, known application library binaries, etc

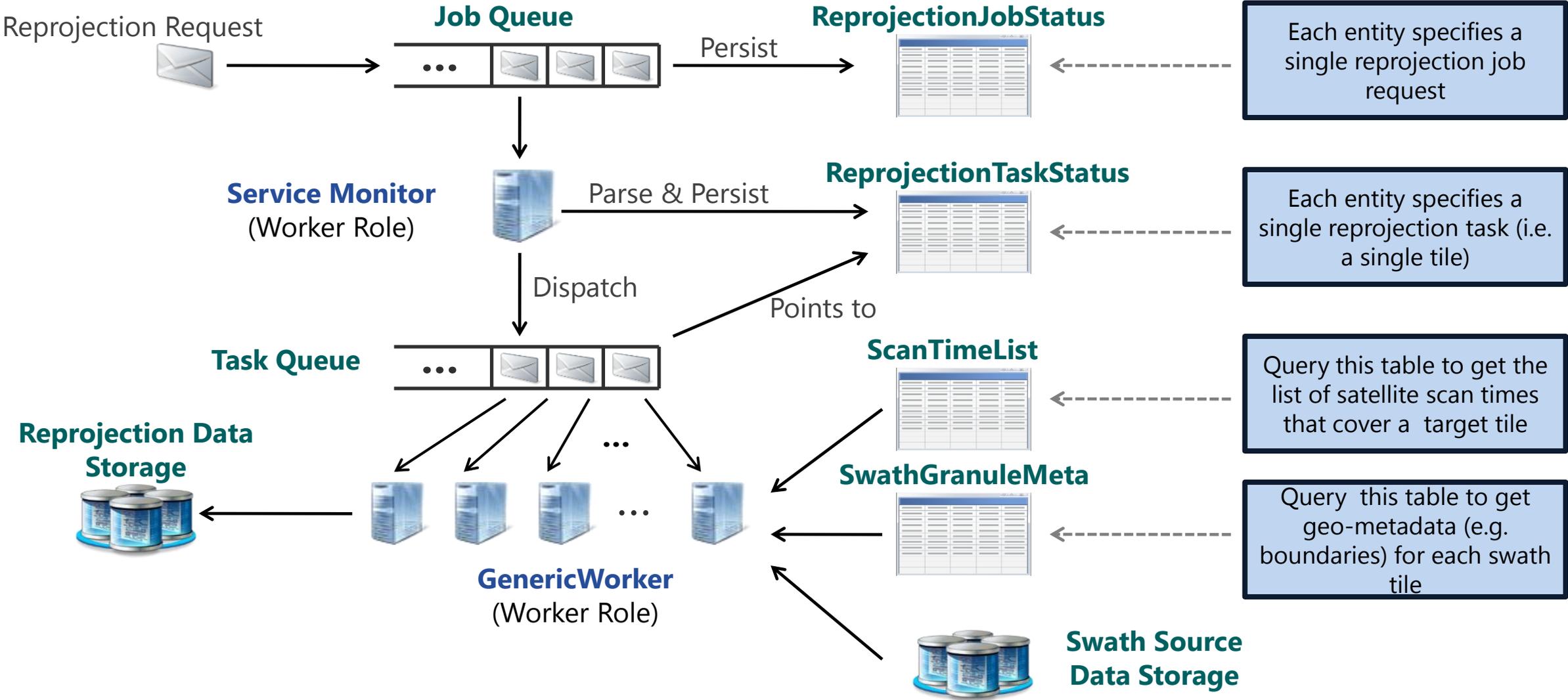
- Necessary for service function
- Never directly accessed by scientist code



Metadata Storage

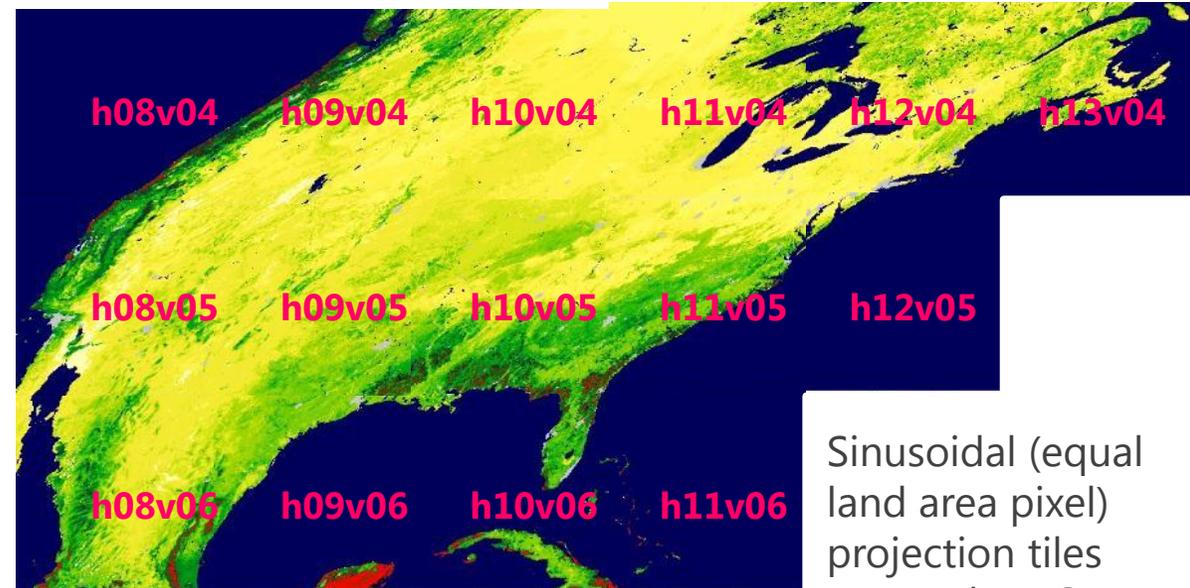
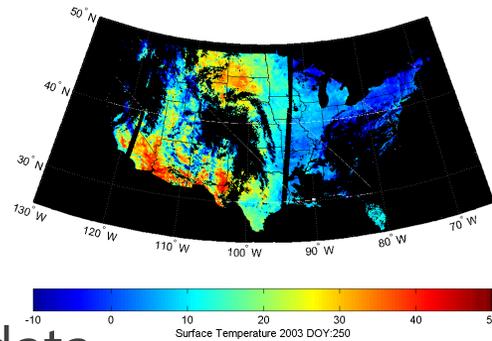
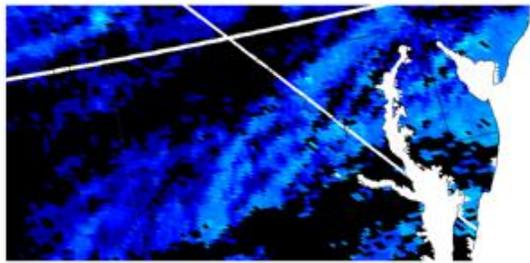
Storage separated by usage to simplify management policies

Example Pipeline Stage: Reprojection Service



Why is Reprojection Tricky?

- It's not just nearest neighbor vs aggregating spline and nadir vs oblique pixels

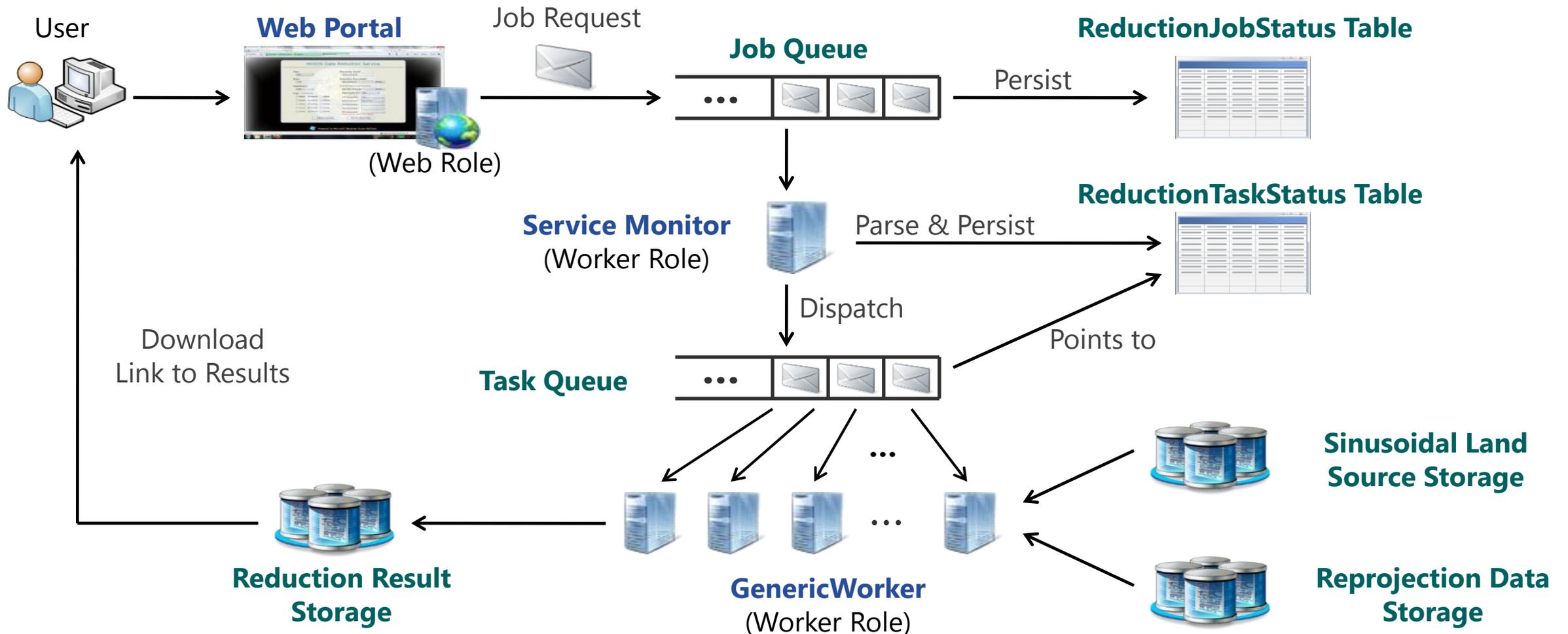


Sinusoidal (equal land area pixel) projection tiles across the US

- Black pixels have no data
 - Non-US land surface masked
 - Vertical bands are gaps between swath tiles; these can be filled by spatial spline or other fit
 - Day/night satellite paths can cause small temporal gaps; clouds cause large spatial and temporal gaps; both must be filled by temporal fit or model result leveraging variables in other products
- White lines have no data
 - Unable to find nearest neighbor at edges of sinusoidal tiles; most likely due to gap between satellite swaths or (early) programming bug
- Processing only the layers of interest makes dramatic savings in compute and storage

Software as a Service means every scientist need not learn to write this code !

Reduction Service (Single Stage Only)

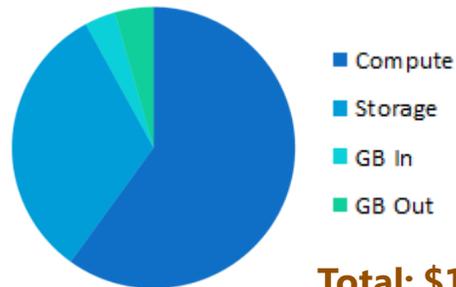


Pipeline Stage Priorities and Interactions

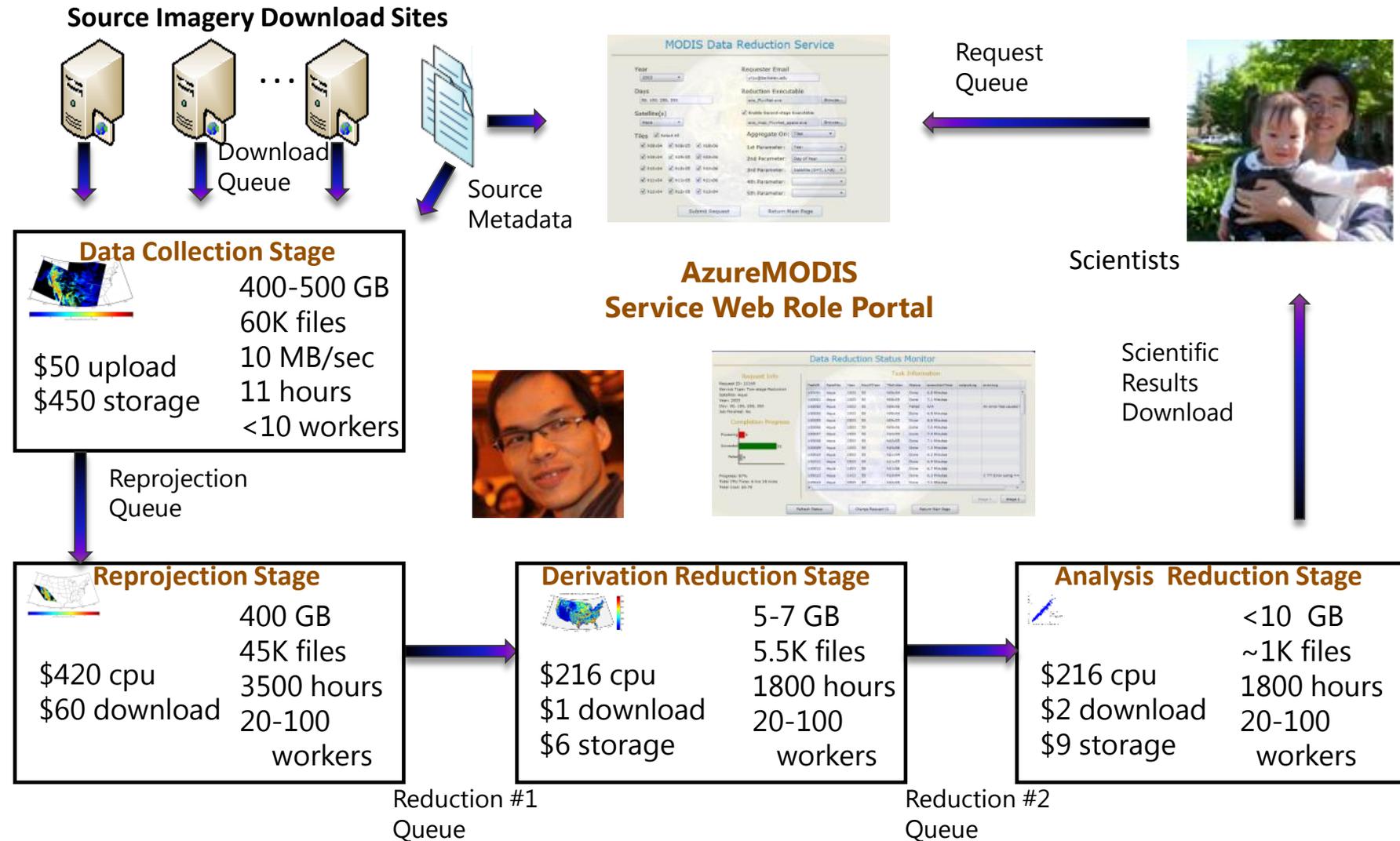
- The Web Portal Role, Service Monitor Role and 5 Generic Worker Roles are deployed at most times
 - 5 Generic Workers are sufficient for reduction algorithm testing and development (\$20/day)
 - Early results returned to scientist while deploying up to 93 additional Generic Workers; such a deployment typically takes 45 minutes
 - Deployment taken down when long periods of idle time are known
 - Heuristic for scaling number of Generic Workers up and down
- Download stage runs in the deep background in all deployed generic worker roles
 - IO, not CPU bound so no competition
- Reduction tasks that have available inputs run preferentially to Reprojection tasks
 - Expedites interactive science result generation
 - If no available inputs and a backlog of reprojection tasks, number of Generic Workers scale up naturally until backlog addressed and reduction can continue
 - Second stage reduction runs only after all first stage reductions have completed

Costs for 1 US Year ET Computation

- Computational costs driven by data scale and need to run reduction multiple times
- Storage costs driven by data scale and 6 month project duration
- Small with respect to the people costs even at graduate student rates !



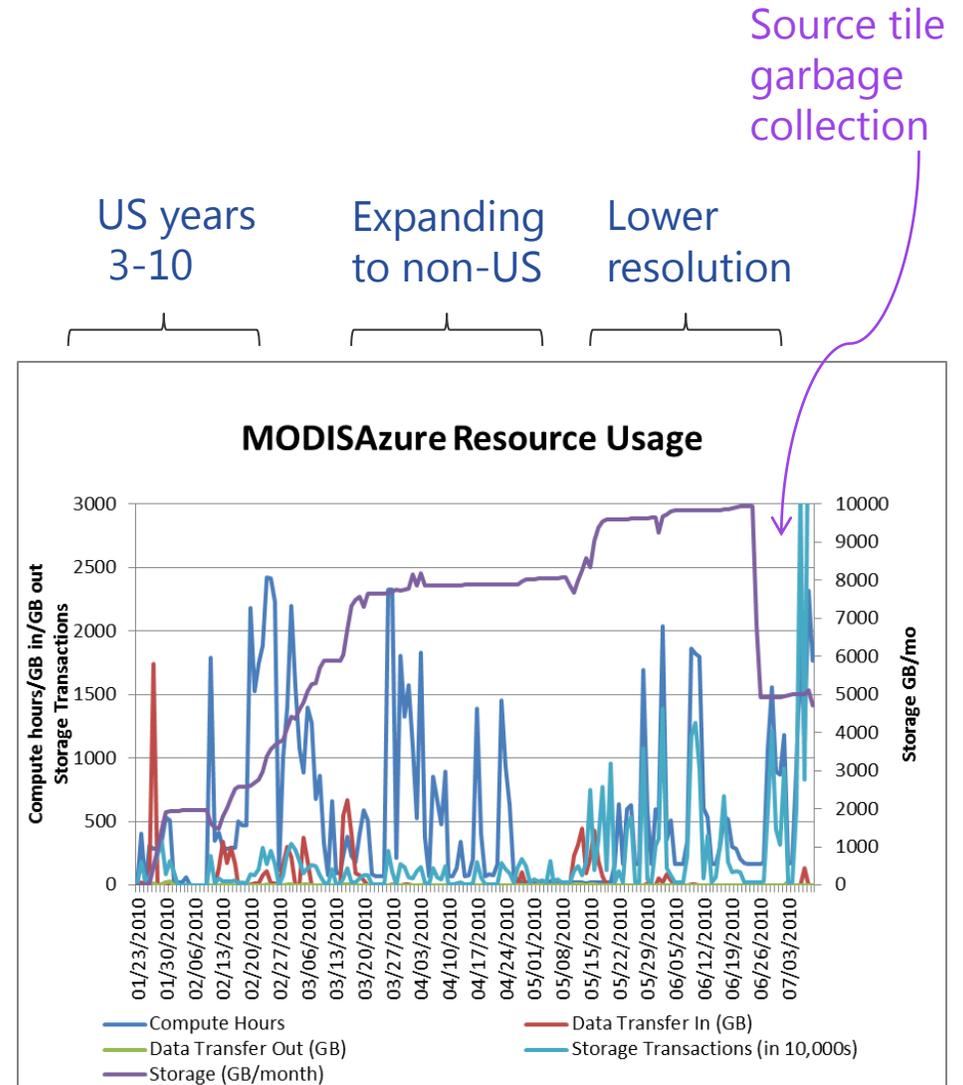
Total: \$1420



Current Status (7/10/2010)

Encouraging science results lead to changing resource needs

- Fine scale computation expanded to cover more of the globe: 2x compute requirements and 2x (transient) storage requirements
- Lower resolution global computation added: .5x compute requirements and 2x (transient) storage and higher IOP/cpu reprojection
- Now underway: geo-spatial validation with yearly aggregate: shifts reduction to IO intensive



Summary

*I can see clearly now, the rain has gone. I can see all obstacles in my way.
Johnny Nash*

Learnings

- Clouds are the **largest scale computer centers ever constructed** and have the potential to be important to both large and small scale science problems.
- Clouds **suitable for “loosely coupled” data parallel applications**, but tightly coupled low-latency applications perform poorly on clouds today.
- Clouds as amplifier for familiar client tools and on premise compute.
- Clouds exploit economies of scale, healthy commercial competition, and an active research community.
- Impedance mismatch between science apps and today’s cloud platforms.
 - Long running tasks, task diversity
 - Performance reliability much different (design for failure)
- Provide **valuable fault tolerance** and **scalability** abstractions

Learnings, cont'd

- Science and algorithm debugging benefit from the same infrastructure as both need to scale up and down
 - Debugging an algorithm on the desktop isn't enough – you have to debug in the cloud too
 - Whenever running at scale in the cloud, you must reduce down to the desktop to understand the results
- Putting all your eggs in the cloud basket means watching that basket
 - Cloud scale resources often mean you still manage small numbers of resources: 100 instances over 24 hours = \$288 even if idle
 - Where is the long term archive for any results ?
- Azure is a rapidly moving target and unlike the Grid
 - Commercial cloud backed by large commercial development team
 - Bake in the faults for scaling and resilience

Acknowledgements

Microsoft Research

- Dan Reed
- Tony Hey
- Dennis Gannon
- David Heckerman
- Nelson Araujo
- Dan Fay
- Jared Jackson
- Wei Liu
- Jaliya Ekanayake
- Simon Mercer
- Yogesh Simmhan
- Michael Zyskowski

Berkeley Water Center, University of California, Berkeley, Lawrence Berkeley Laboratory

- Deb Agarwal
- Dennis Baldocchi
- Jim Hunt
- Monte Goode
- Susan Hubbard
- Keith Jackson
- Rebecca Leonardson (student)
- Carolyn Remick

University of Virginia

- Marty Humphrey
- Norm Beekwilder
- Jie Li (student)

Indiana University

- You-Wei Cheah (student)

Fluxnet Collaboration

- Dennis Baldocchi
- Youngryel Ryu (postdoc)
- Dario Papale (CarboEurope)
- Markus Reichstein (CarboEurope)
- Alan Barr (Fluxnet-Canada)
- Bob Cook
- Dorothea Frank
- Susan Holladay
- Hank Margolis (Fluxnet-Canada)
- Rodrigo Vargas (postdoc)

Ameriflux Collaboration

- Beverly Law
- Tom Boden
- Mattias Falk
- Tara Hudiburg (student)
- Hongyan Luo (postdoc)
- Gretchen Miller (student)
- Lucie Ploude (student)
- Andrew Richardson
- Andrea Scheutz (student)
- Christophe Thomas



<http://azurescope.cloudapp.net/>

<http://research.microsoft.com/cloud>



<http://www.fluxdata.org>

The Microsoft logo is centered on the page. It consists of the word "Microsoft" in a bold, italicized, black sans-serif font. A registered trademark symbol (®) is located at the top right of the word.

© 2010 Microsoft Corporation. All rights reserved. Microsoft, Windows, Windows Vista and other product names are or may be registered trademarks and/or trademarks in the U.S. and/or other countries.
The information herein is for informational purposes only and represents the current view of Microsoft Corporation as of the date of this presentation. Because Microsoft must respond to changing market conditions, it should not be interpreted to be a commitment on the part of Microsoft, and Microsoft cannot guarantee the accuracy of any information provided after the date of this presentation.
MICROSOFT MAKES NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, AS TO THE INFORMATION IN THIS PRESENTATION.