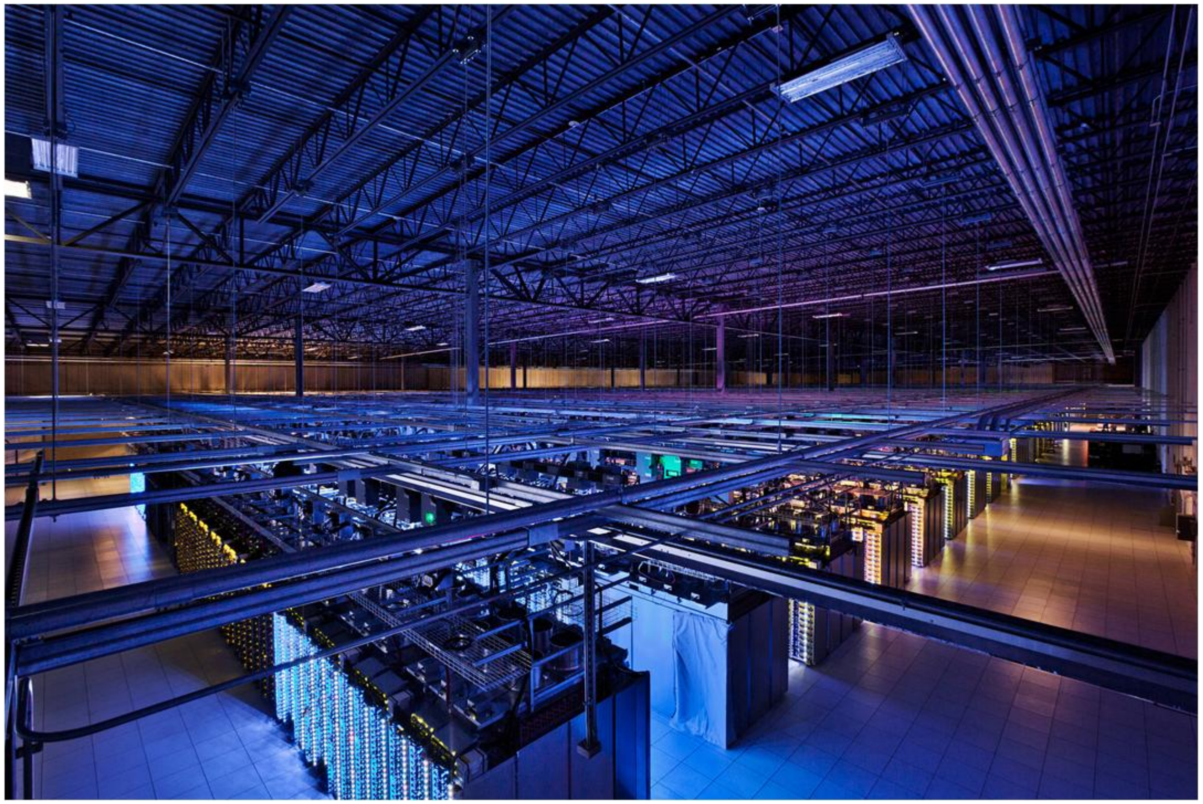
Centrum Wiskunde & Informatica

Big Data for Data Science

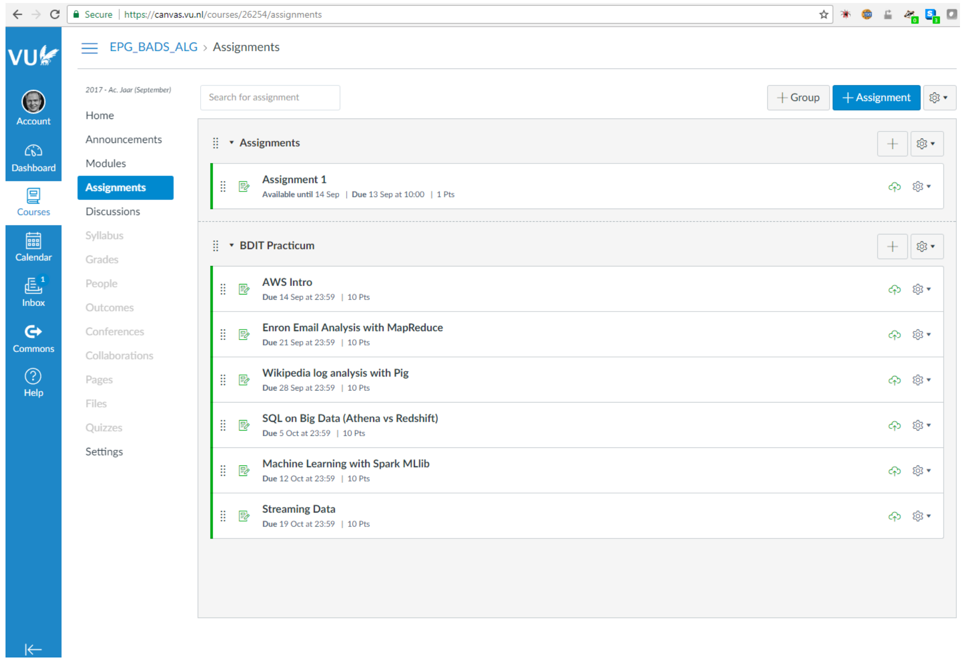
Intro to Big Data and Cloud Computing



www.cwi.nl/~boncz/bads

Administration

- Canvas
  - Announcements, also via email
  - Turning in practicum assignments, Check Grades



Assignments

- Assignment 1 — Available until 14 Sep | Due 13 Sep at 10:00 | 1 Pts

BDIT Practicum

- AWS Intro — Due 14 Sep at 23:59 | 10 Pts
- Enron Email Analysis with MapReduce — Due 21 Sep at 23:59 | 10 Pts
- Wikipedia log analysis with Pig — Due 28 Sep at 23:59 | 10 Pts
- SQL on Big Data (Athena vs Redshift) — Due 5 Oct at 23:59 | 10 Pts
- Machine Learning with Spark MLlib — Due 12 Oct at 23:59 | 10 Pts
- Streaming Data — Due 19 Oct at 23:59 | 10 Pts

- Contact by email: boncz.muehleisen@gmail.com

Goals & Scope

- The goal of the course is to gain insight into and experience with hardware infrastructures and software technologies for managing 'big data'.

- Understanding the concepts

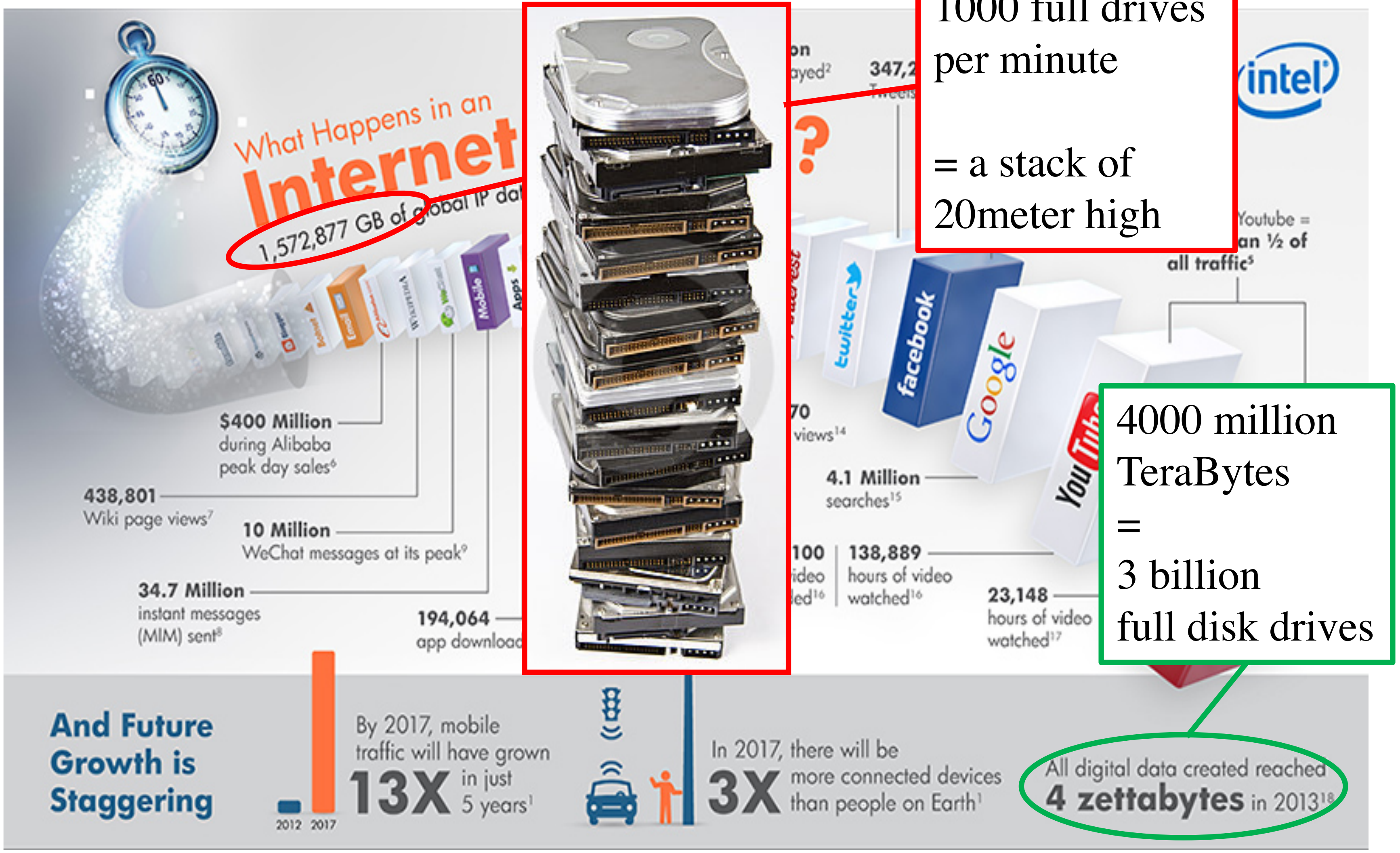
  hardware
  - What components are hardware infrastructures made up of?
  - What are the properties of these hardware components?
  - What does it take to access such hardware?

  software
  - What software layers are used to handle Big Data?
  - What are the principles behind this software?
  - Which kind of software would one use for which data problem?

Modus Operandi in this Module

- Presence at the Lectures
  - Lot’s of information on the slides
  - Interact!
- Work in groups of three
  - Each group combines someone with a ‘technical’ background with others who have less such experience
  - Use enrollment in a practicum group via canvas
- During second half of class we start with the practicum
  - If not finished then, you may continue from home
  - Post your solution in blackboard before the next lecture
    - Scores [0-2],1=visible effort, grade = SUM(scores)
- At home, read the background information in the literature
  - required: **lecture summaries**!
  - nice to know: **background** and **extra** material (vetted youtube & slideshare)
  - reference only: scientific literature (probably too technical)

The age of Big Data


What Happens in an
Internet
1,572,877 GB of global IP data
$400 Million during Alibaba peak day sales
438,801 Wiki page views
10 Million WeChat messages at its peak
34.7 Million instant messages (MIM) sent
194,064 app downloads
347,2 Tweets
Youtube = an ½ of all traffic
4.1 Million searches
138,889 hours of video watched
23,148 hours of video watched
And Future Growth is Staggering
2012 2017
By 2017, mobile traffic will have grown 13X in just 5 years
In 2017, there will be 3X more connected devices than people on Earth
All digital data created reached 4 zettabytes in 2013

1500TB/min = 1000 full drives per minute

= a stack of 20meter high

4000 million TeraBytes = 3 billion full disk drives

www.cwi.nl/~boncz/bads

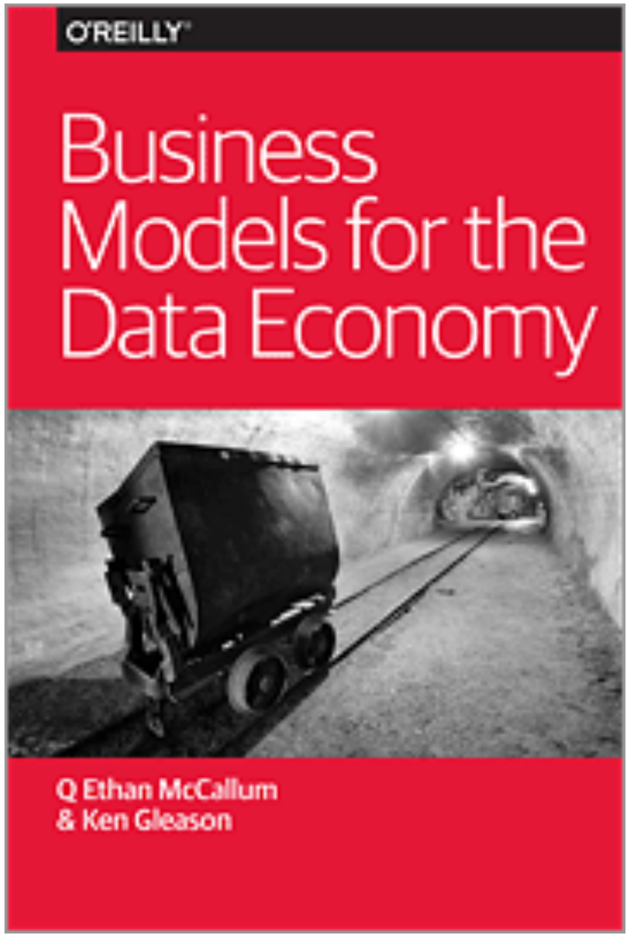
Centrum Wiskunde & Informatica

“Big Data”

www.cwi.nl/~boncz/bads

CWI
Centrum Wiskunde & Informatica

The Data Economy

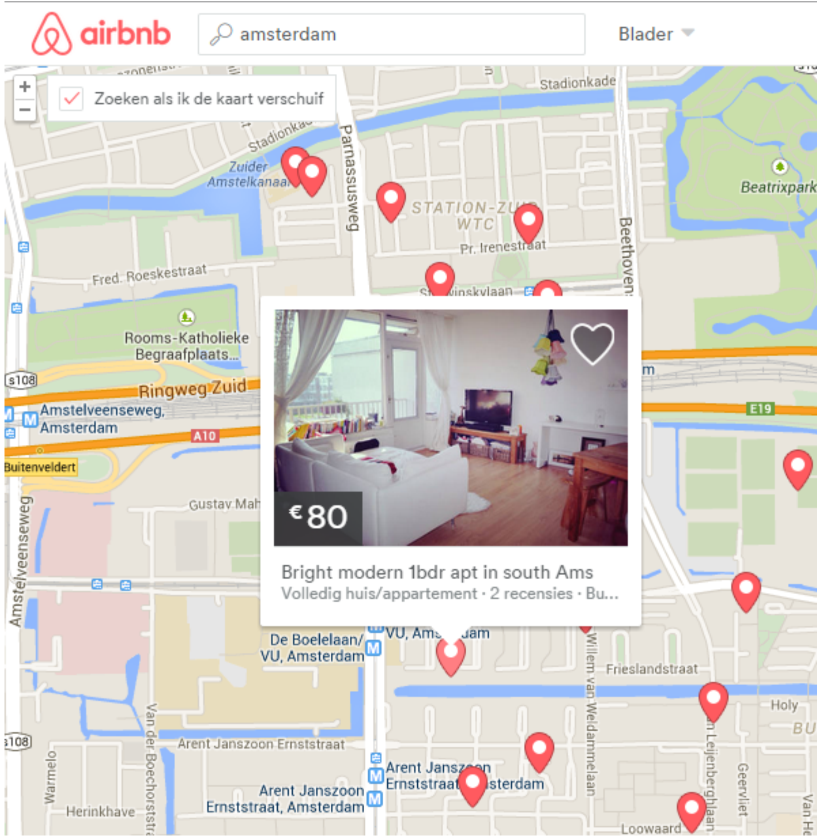


CWI
Centrum Wiskunde & Informatica

Disruptions by the Data Economy

UBER

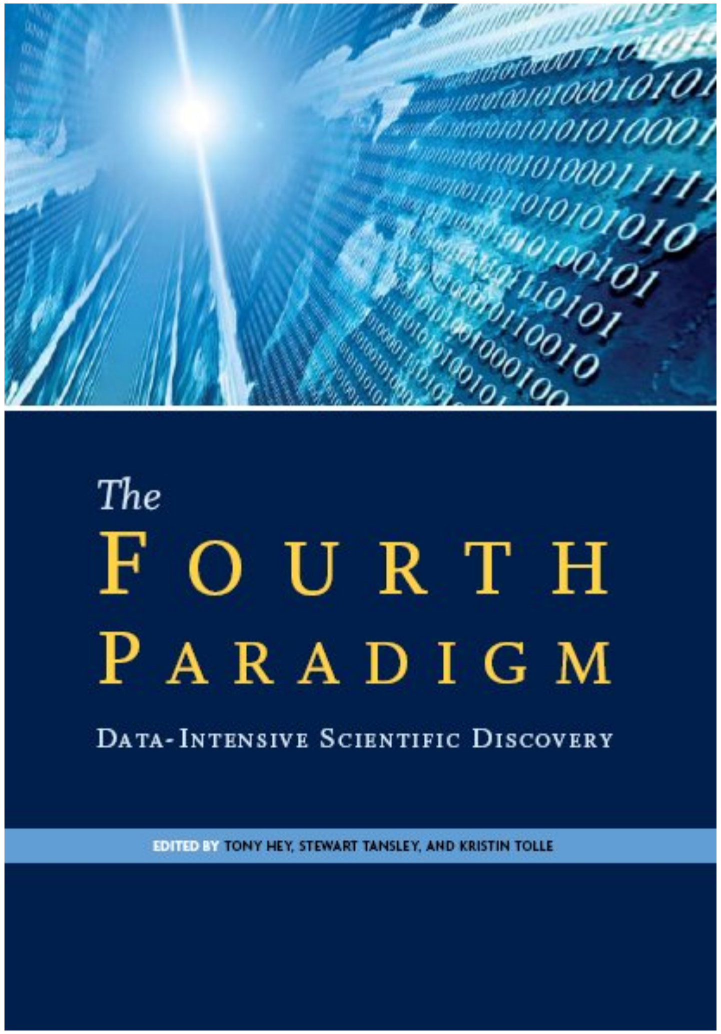
EVERYONE'S PRIVATE DRIVER™



airbnb
amsterdam
Blader
Zoeken als ik de kaart verschuif
€80
Bright modern 1bdr apt in south Ams
Volledig huis/appartement · 2 recensies · Bu...

www.cwi.nl/~boncz/bads

Data Disrupting Science

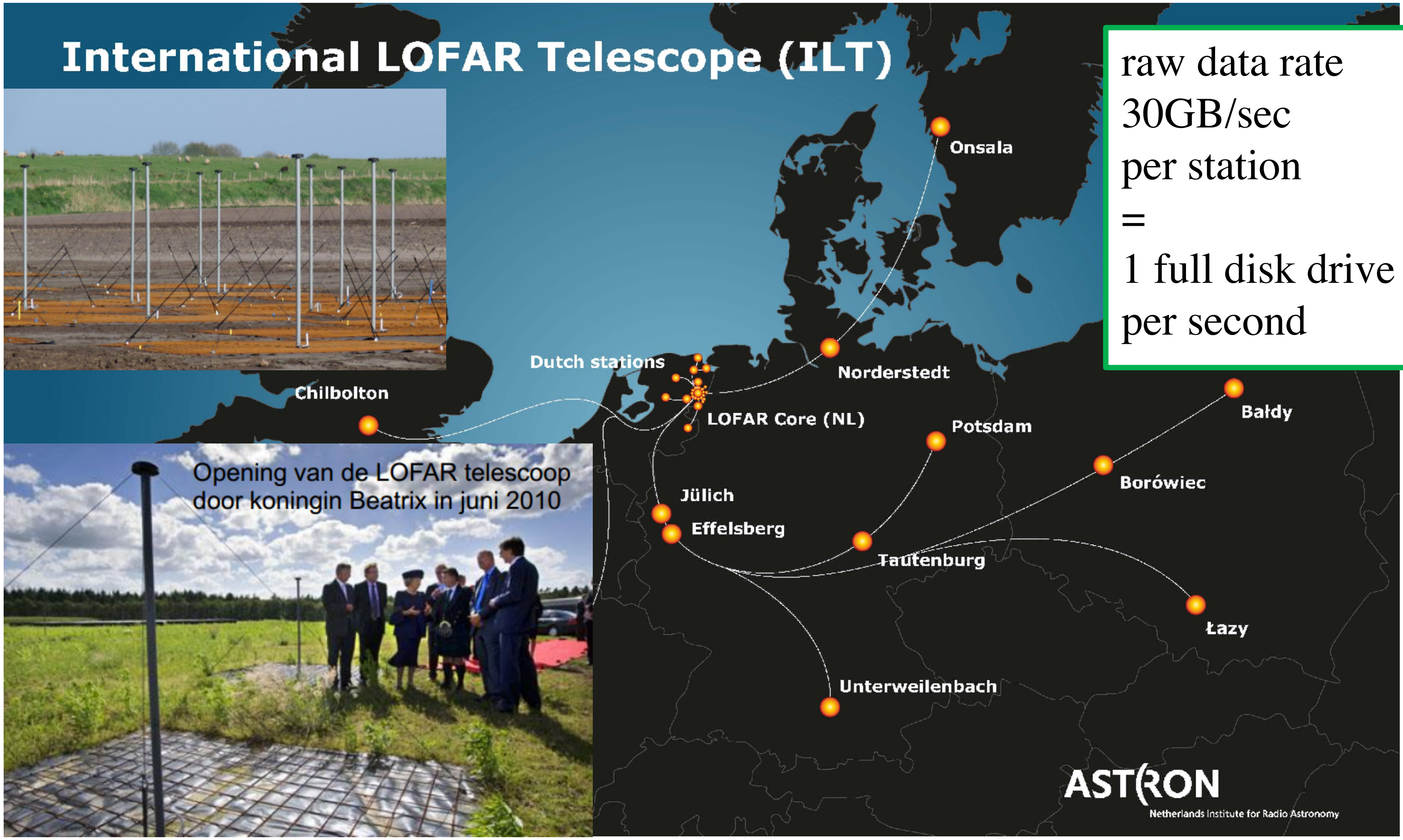

The Fourth Paradigm
Data-Intensive Scientific Discovery
Edited by Tony Hey, Stewart Tansley, and Kristin Tolle

Scientific paradigms:

1. Observing
2. Modeling
3. Simulating
4. Collecting and Analyzing Data

Centrum Wiskunde & Informatica

Data Driven Science


International LOFAR Telescope (ILT)
Onsala
Dutch stations
Norderstedt
Chilbolton
LOFAR Core (NL)
Potsdam
Bałdy
Borówiec
Jülich
Effelsberg
Tautenburg
Łazy
Unterweilenbach
Opening van de LOFAR telescoop door koningin Beatrix in juni 2010
ASTRON
Netherlands Institute for Radio Astronomy

raw data rate
30GB/sec
per station
=
1 full disk drive
per second

www.cwi.nl/~boncz/bads

Big Data

- Big Data is a relative term
  - If things are breaking, you have Big Data
  - Big Data is not always Petabytes in size
  - Big Data for Informatics is not the same as for Google
- Big Data is often hard to understand
  - A model explaining it might be as complicated as the data itself
  - This has implications for Science
- The game may be the same, but the rules are completely different
  - What used to work needs to be reinvented in a different context

Centrum Wiskunde & Informatica

Big Data Challenges (1/3)

- **Volume** ➔ data larger than a single machine (CPU,RAM,disk)
  - Infrastructures and techniques that scale by using more machines
  - Google led the way in mastering “cluster data processing”
- Velocity
- Variety

Supercomputers?

- Take the top two supercomputers in the world today
  - Tiahne-2 (Guangzhou, China)
    - Cost: US$390 million
  - Titan (Oak Ridge National Laboratory, US)
    - Cost: US$97 million
- Assume an expected lifetime of five years and compute cost per hour
  - Tiahne-2: US$8,220
  - Titan: US$2,214
- This is just for the machine showing up at the door
  - Not factored in operational costs (e.g., running, maintenance, power, etc.)

Let’s rent a supercomputer for an hour!

- Amazon Web Services charge US$1.60 per hour for a large instance
  - An 880 large instance cluster would cost US$1,408
  - Data costs US$0.15 per GB to upload
    - Assume we want to upload 1TB
    - This would cost US$153
  - The resulting setup would be #146 in the world's top-500 machines
  - Total cost: US$1,561 per hour
  - Search for (first hit): LINPACK 880 server

Supercomputing vs Cluster Computing

- Supercomputing
  - Focus on performance (biggest, fastest).. At any cost!
    - Oriented towards the [secret] government sector / scientific computing
  - Programming effort seems less relevant
    - Fortran + MPI: months do develop and **debug** programs
    - GPU, i.e. computing with graphics cards
    - FPGA, i.e. casting computation in hardware circuits
  - Assumes high-quality stable hardware
- Cluster Computing
  - use a network of many computers to create a ‘supercomputer’
  - oriented towards business applications
  - use cheap servers (or even desktops), unreliable hardware
    - software must make the unreliable parts reliable
  - focus on economics (bang for the buck)
    - programming effort counts, a lot! No time to lose on debugging..

Cloud Computing vs Cluster Computing

- Cluster Computing
  - Solving large tasks with more than one machine
    - Parallel database systems (e.g. Teradata, Vertica)
    - noSQL systems
    - Hadoop / MapReduce
- Cloud Computing

Cloud Computing vs Cluster Computing

- Cluster Computing
- Cloud Computing
  - Machines operated by a third party in large data centers
    - sysadmin, electricity, backup, maintenance externalized
  - Rent access by the hour
    - Renting machines (Linux boxes): Infrastructure as a Service
    - Renting systems (Redshift SQL): Platform-as-a-service
    - Renting an software solution (Salesforce): Software-as-a-service
- {Cloud,Cluster} are independent concepts, but they are often combined!
  - We will do so in the practicum (Hadoop on Amazon Web Services)

Economics of Cloud Computing

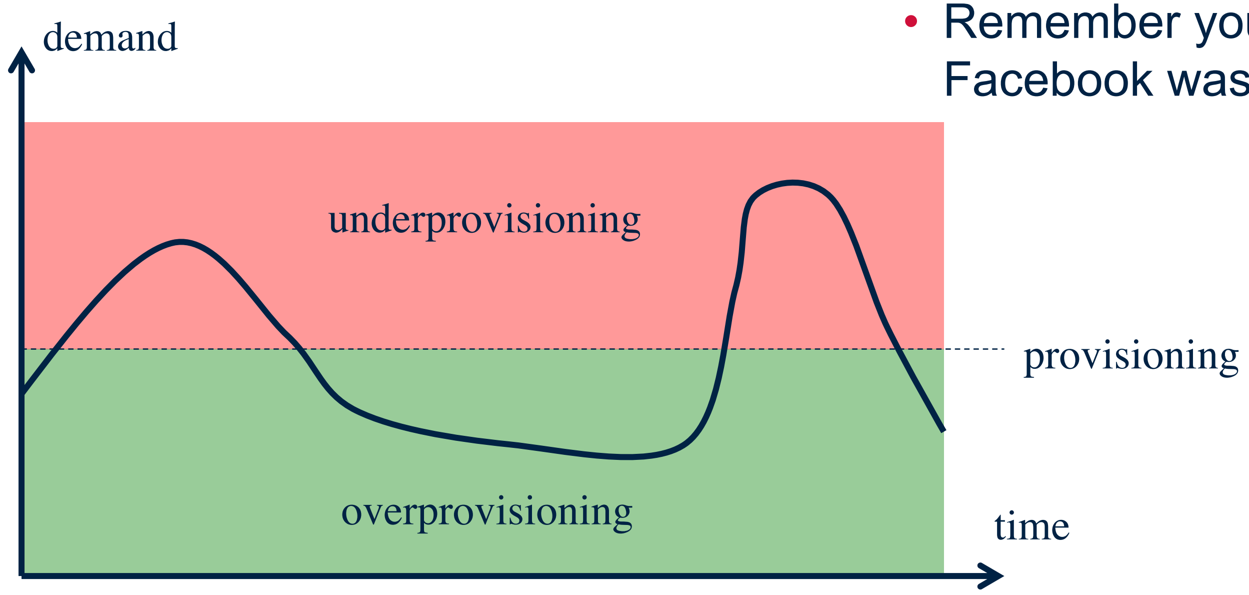
- A major argument for Cloud Computing is pricing:
  - We could own our machines
    - … and pay for electricity, cooling, operators
    - …and allocate enough capacity to deal with peak demand
  - Since machines rarely operate at more than 30% capacity, we are paying for wasted resources
- Pay-as-you-go rental model
  - Rent machine instances by the hour
  - Pay for storage by space/month
  - Pay for bandwidth by space/hour
- No other costs
- This makes computing a commodity
  - Just like other commodity services (sewage, electricity etc.)

CWI
Centrum Wiskunde & Informatica

Cloud Computing: Provisioning

- We can quickly scale resources as demand dictates
  - High demand: more instances
  - Low demand: fewer instances
- Elastic provisioning is crucial

- Target (US retailer) uses Amazon Web Services (AWS) to host `target.com`
  - During massive spikes (November 28 2009 –"Black Friday") `target.com` is unavailable
- Remember your panic when Facebook was down?



Cloud Computing: some rough edges

- Some provider hosts our data
  - But we can only access it using proprietary (non-standard) APIs
  - **Lock-in** makes customers vulnerable to price increases and dependent upon the provider
  - Local laws (e.g. privacy) might prohibit externalizing data processing
- Providers may control our data in unexpected ways:
  - July 2009: Amazon remotely remove books from Kindles
  - Twitter prevents exporting tweets more than 3200 posts back
  - Facebook locks user-data in
  - Paying customers forced off Picasa towards Google Plus
- Anti-terror laws mean that providers have to grant access to governments
  - This privilege can be overused

Big Streaming Data

- Storing it is not really a problem: disk space is cheap
- Efficiently accessing it and deriving results can be hard
- Visualising it can be next to impossible
- Repeated observations
  - What makes Big Data big are repeated observations
  - Mobile phones report their locations every 15 seconds
  - People post on Twitter > 100 million posts a day
  - The Web changes every day
  - Potentially we need unbounded resources
    - Repeated observations motivates streaming algorithms

Privacy and security

- People will not use Cloud Computing if trust is eroded
  - Who can access it?
    - Governments? Other people?
    - Snowden is the Chernobyl of Big Data
  - Privacy guarantees needs to be clearly stated and kept-to
- Privacy breaches
  - Numerous examples of Web mail accounts hacked
  - Many many cases of (NL) governmental data loss
  - TJX Companies Inc. (2007): 45 million credit and debit card numbers stolen
  - Every day there seems to be another instance of private data being leaked to the public

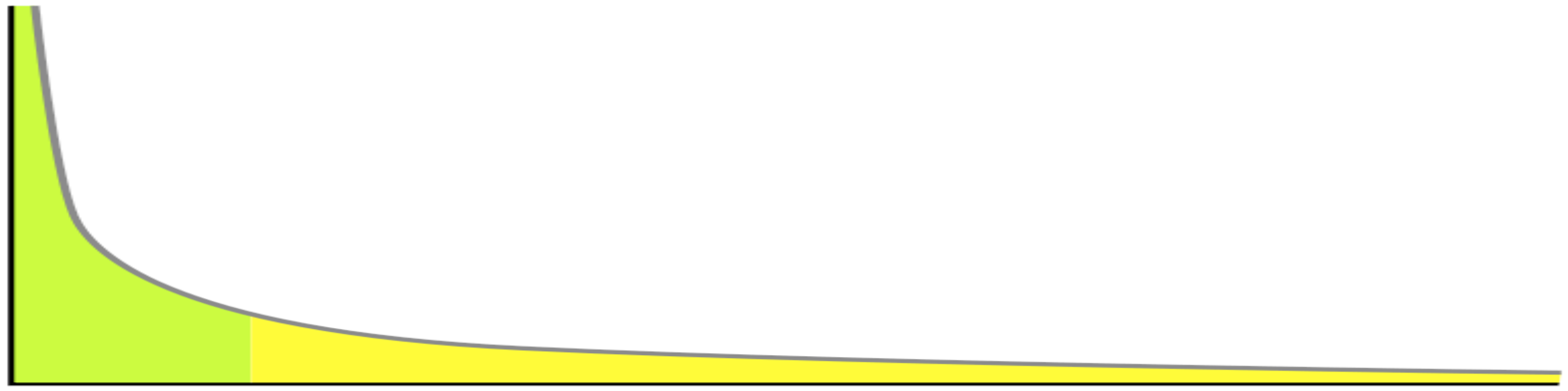
Big Data Challenges (2/3)

- Volume
- **Velocity** ➔ endless stream of new events
  - No time for heavy indexing (new data keeps arriving always)
  - led to development of data stream technologies
- Variety

Big Data Challenges (3/3)

- Volume
- Velocity
- **Variety** ➔ Dirty, incomplete, inconclusive data (e.g. text in tweets)
  - Semantic complications:
    - AI techniques needed, not just database queries
    - Data mining, Data cleaning, text analysis (AI techniques)
  - Technical complications:
    - Skewed Value Distributions and “Power Laws”
    - Complex Graph Structures ➔ Expensive Random Access
    - Complicates cluster data processing (difficult to partition equally)
    - Localizing data by attaching pieces where you need them makes Big Data even bigger

Power laws



- This is **not** the 80/20 rule of skewed data (“80% of the sales are the 20% of the products”)
  - In a power law distribution, 80% of the data sales could well be in the “long tail” (the model is as large as the data).
- Modelling the head is easy, but may not be representative of the full population
  - Dealing with the full population might imply Big Data (e.g., selling all books, not just block busters)
- Processing Big Data might reveal power-laws
  - Most items take a small amount of time to process, individually
  - But there may be very many relevant items (“products”) to keep track of
  - A few items take a lot of time to process

Skewed Data

- Distributed computation is a natural way to tackle Big Data
  - MapReduce encourages sequential, disk-based, localised processing of data
  - MapReduce operates over a cluster of machines
- One consequence of power laws is uneven allocation of data to nodes
  - The head might go to one or two nodes
  - The tail would spread over all other nodes
  - All workers on the tail would finish quickly.
  - The head workers would be a lot slower
- Power laws can turn parallel algorithms into sequential algorithms

CWI
Centrum Wiskunde & Informatica

Big Data Challenges (3/3)

- Volume
- Velocity
- **Variety** ➔ Dirty, incomplete, inconclusive data (e.g. text in tweets)
  - Semantic complications:
    - AI techniques needed, not just database queries
    - Data mining, Data cleaning, text analysis (AI techniques)
  - Technical complications:
    - Skewed Value Distributions and “Power Laws”
    - Complex Graph Structures ➔ Expensive Random Access
    - Complicates cluster data processing (difficult to partition equally)
    - Localizing data by attaching pieces where you need them makes Big Data even bigger

Summary

- Introduced the notion of Big Data, the three V's
  - Volume, Velocity, Variety
- Explained differences between Super/Cluster/Cloud computing

Cloud computing:

- Computing as a commodity is likely to increase over time
- Cloud Computing adaptation and adoption are driven by economics
- The risks and obstacles behind it are complex
- Three levels:
  - Infrastructure as a service (run machines)
  - Platform as a service (use a database system on the cloud)
  - Software as a service (use software managed by other on the cloud)

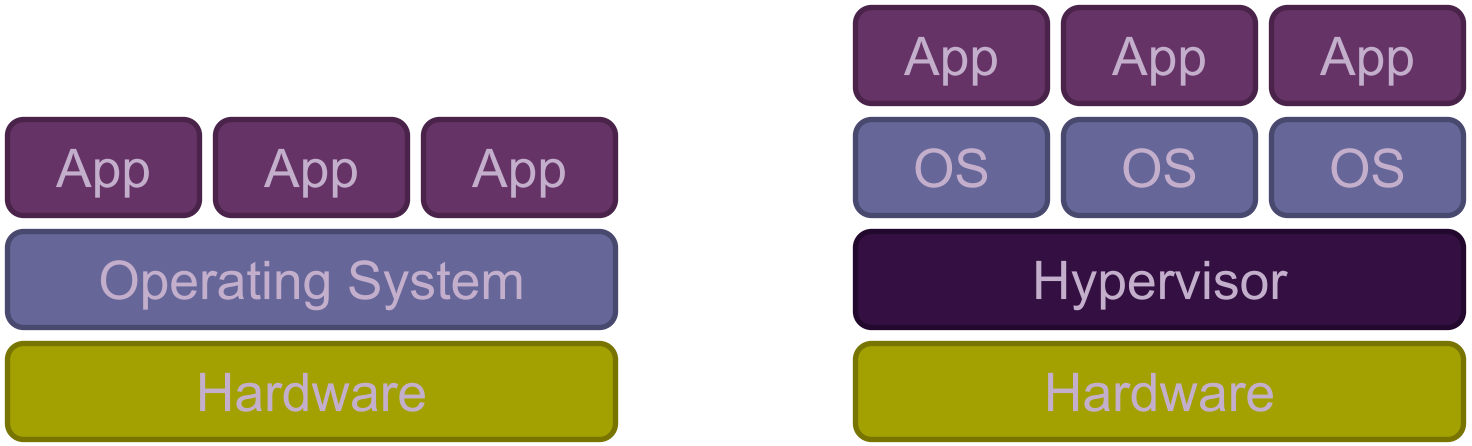
CLOUD COMPUTING

Centrum Wiskunde & Informatica

Cloud computing

- What?
  - Computing resources as a metered service (“pay as you go”)
  - Ability to dynamically provision virtual machines
- Why?
  - Cost: capital vs. operating expenses
  - Scalability: “infinite” capacity
  - Elasticity: scale up or down on demand
- Does it make sense?
  - Benefits to cloud users
  - Business case for cloud providers

Enabling technology: virtualisation



Everything as a service

- Utility computing = Infrastructure as a Service (IaaS)
  - Why buy machines when you can rent cycles?
  - Examples: Amazon’s EC2, Rackspace
- Platform as a Service (PaaS)
  - Give me nice API and take care of the maintenance, upgrades
  - Example: Google App Engine
- Software as a Service (SaaS)
  - Just run it for me!
  - Example: Gmail, Salesforce

Several Historical Trends (1/3)

- Shared Utility Computing
  - 1960s – MULTICS – Concept of a Shared Computing Utility
  - 1970s – IBM Mainframes – rent by the CPU-hour. (Fast/slow switch.)
- Data Center Co-location
  - 1990s-2000s – Rent machines for months/years, keep them close to the network access point and pay a flat rate. Avoid running your own building with utilities!
- Pay as You Go
  - Early 2000s - Submit jobs to a remote service provider where they run on the raw hardware. Sun Cloud ($1/CPU-hour, Solaris +SGE) IBM Deep Capacity Computing on Demand (50 cents/hour)

Several Historical Trends (2/3)

- Virtualization
  - 1960s – OS-VM, VM-360 – Used to split mainframes into logical partitions.
  - 1998 – VMWare – First practical implementation on X86, but at significant performance hit.
  - 2003 – Xen paravirtualization provides much perf, but kernel must assist.
  - Late 2000s – Intel and AMD add hardware support for virtualization.

Several Historical Trends (3/3)

- Minicomputers (1960-1990)
  - IBM AS/400, DEC VAX
- The age of the x86 PC (1990-2010)
  - IBM PC, Windows (1-7)
  - Linux takes the server market (2000-)
  - Hardware innovation focused on Gaming/Video (GPU), Laptop
- Mobile and Server separate (2010-)
  - Ultramobile (tablet,phone) .➔ ARM
  - Server ➔ x86 still but much more influence on hardware design
    - Parallel processing galore (software challenge!)
    - Large utility computing provides build their own hardware
      - Amazon SSD cards (FusionIO)
      - Google network Routers

CWI
Centrum Wiskunde & Informatica

So, you want to build a cloud

- Slightly more complicated than hooking up a bunch of machines with an ethernet cable
  - Physical vs. virtual (or logical) resource management
  - Interface?
- A host of issues to be addressed
  - Connectivity, concurrency, replication, fault tolerance, file access, node access, capabilities, services, …
- We'll tackle as many problems as we can
  - The problems are nothing new
  - Solutions have existed for a long time
  - However, it's the first time we have the challenge of applying them all in a single massively accessible infrastructure

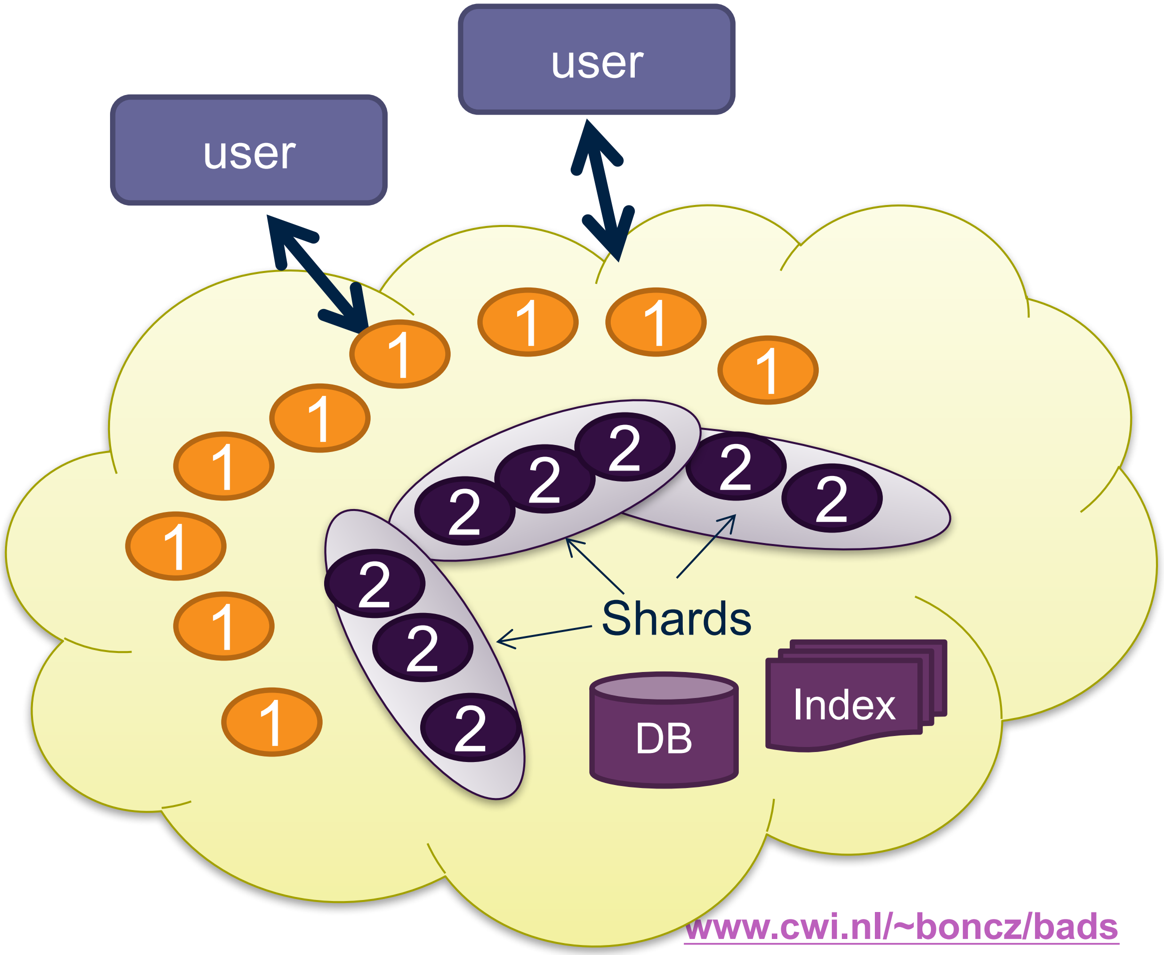
How are clouds structured?

- Clients talk to clouds using web browsers or the web services standards
  - But this only gets us to the outer “skin” of the cloud data center, not the interior
  - Consider Amazon: it can host entire company web sites (like Target.com or Netflix.com), data (S3), servers (EC2) and even user-provided virtual machines!

CWI
Centrum Wiskunde & Informatica

Big picture overview

- Client requests are handled in the first tier by
  - PHP or ASP pages
  - Associated logic
- These lightweight services are fast and very nimble
- Much use of caching: the second tier



www.cwi.nl/~boncz/bads

Many styles of system

- Near the edge of the cloud focus is on vast numbers of clients and rapid response
- Inside we find high volume services that operate in a pipelined manner, asynchronously
- Deep inside the cloud we see a world of virtual computer clusters that are
  - Scheduled to share resources
  - Run applications like MapReduce (Hadoop) are very popular
  - Perform the heavy lifting

In the outer tiers replication is key

- We need to replicate
  - Processing
    - Each client has what seems to be a private, dedicated server (for a little while)
  - Data
    - As much as possible!
    - Server has copies of the data it needs to respond to client requests without any delay at all
  - Control information
    - The entire system is managed in an agreed-upon way by a decentralised cloud management infrastructure

What about the shards?

- The caching components running in tier two are central to the responsiveness of tier-one services
- Basic idea is to always used cached data if at all possible
  - So the inner services (here, a database and a search index stored in a set of files) are shielded from the online load
  - We need to replicate data within our cache to spread loads and provide fault-tolerance
  - But not everything needs to be fully replicated
  - Hence we often use shards with just a few replicas

Sharding used in many ways

- The second tier could be any of a number of caching services:
  - Memcached: a sharable in-memory key-value store
  - Other kinds of Distributed Hash Tables that use key-value APIs
  - Dynamo: A service created by Amazon as a scalable way to represent the shopping cart and similar data
  - BigTable: A very elaborate key-value store created by Google and used not just in tier-two but throughout their “GooglePlex” for sharing information
- Notion of sharding is cross-cutting
  - Most of these systems replicate data to some degree
- We will examine quite a few of these implementations
  - You may have actually used them, do you know how they work?

First-tier parallelism

- Parallelism is vital to speeding up first-tier services
- Key question
  - Request has reached some service instance X
  - Will it be faster
    - For X to just compute the response?
    - Or for X to subdivide the work by asking subservices to do parts of the job?
- Glimpse of an answer
  - Werner Vogels, CTO at Amazon, commented in one talk that many Amazon pages have content from 50 or more parallel subservices that run, in real-time, on the request!

Centrum Wiskunde & Informatica

Read vs. write

- Parallelisation works fine, so long as we are reading
- If we break a large read request into multiple read requests for sub-components to be run in parallel, how long do we need to wait?
  - Answer: as long as the slowest read
- How about breaking a large write request?
  - Duh… we still wait till the slowest write finishes
- But what if these are not sub-components, but alternative copies of the same resource?
  - Also known as replicas
  - We wait the same time, but when do we make the individual writes visible?

Replication solves one problem but introduces another

More on updating replicas in parallel

- Several issues now arise
  - Are all the replicas applying updates in the same order?
    - Might not matter unless the same data item is being changed
    - But then clearly we do need some agreement on order
  - What if the leader replies to the end user but then crashes and it turns out that the updates were lost in the network?
    - Data center networks are surprisingly lossy at times
    - Also, bursts of updates can queue up
- Such issues result in inconsistency

Eric Brewer's CAP theorem

- In a famous 2000 keynote talk at ACM PODC, Eric Brewer proposed that
  - "*You can have just two from Consistency, Availability and Partition Tolerance*"
- He argues that data centres need very fast response, hence availability is paramount
- And they should be responsive even if a transient fault makes it hard to reach some service
- So they should use cached data to respond faster even if the cached entry cannot be validated and might be stale!
- Conclusion: weaken consistency for faster response
- We will revisit this as we go along

Is inconsistency a bad thing?

- How much consistency is really needed in the first tier of the cloud?
  - Think about YouTube videos. Would consistency be an issue here?
  - What about the Amazon “number of units available” counters. Will people notice if those are a bit off?
    - Probably not unless you are buying the last unit
    - End even then, you might be inclined to say “oh, bad luck”

Centrum Wiskunde & Informatica

CASE STUDY: AMAZON WEB SERVICES

Amazon AWS

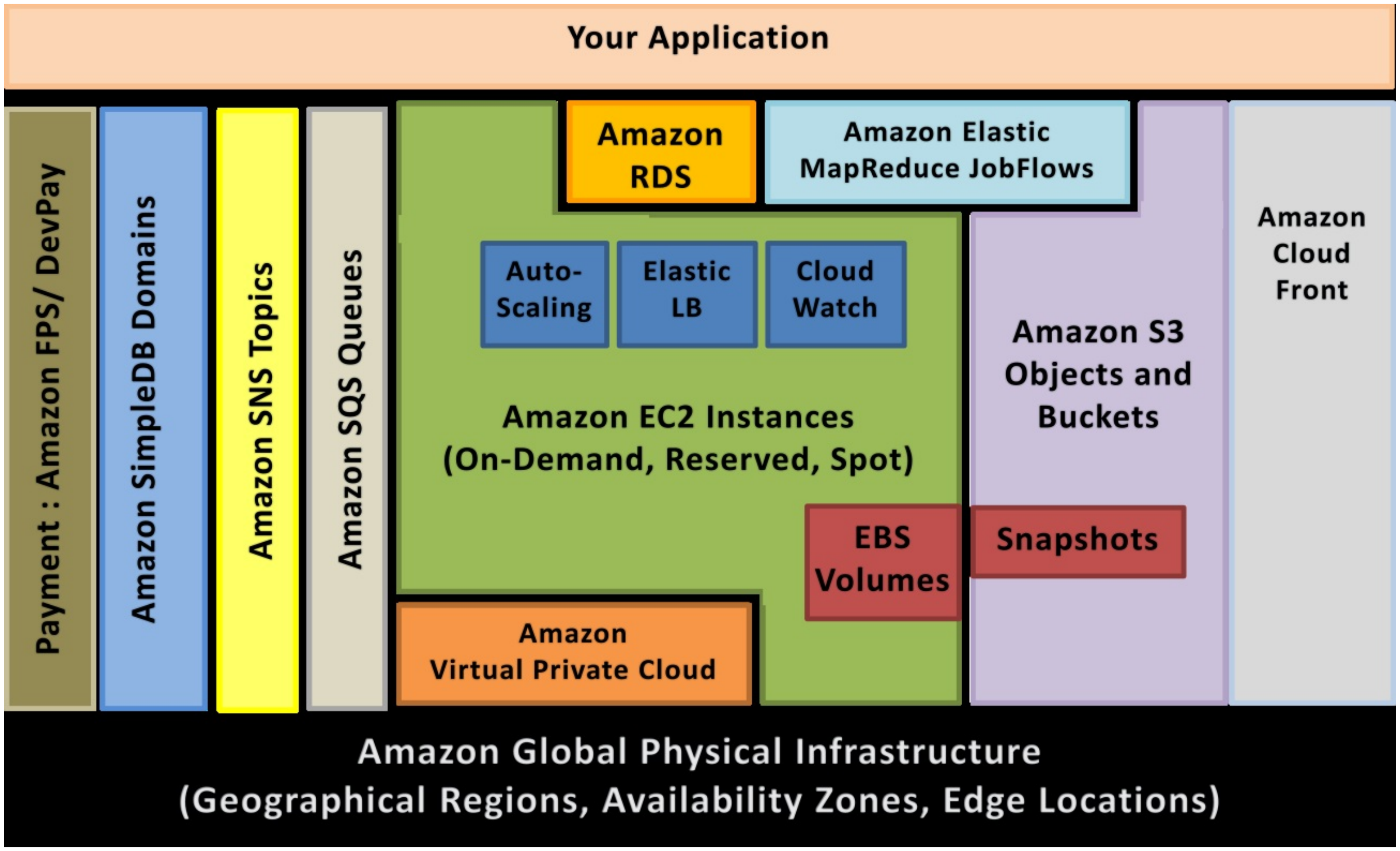
- Grew out of Amazon's need to rapidly provision and configure machines of standard configurations for its own business.
- Early 2000s – Both private and shared data centers began using virtualization to perform "server consolidation"
- 2003 – Internal memo by Chris Pinkham describing an "infrastructure service for the world."
- 2006 – S3 first deployed in the spring, EC2 in the fall
- 2008 – Elastic Block Store available.
- 2009 – Relational Database Service
- 2012 – DynamoDB

Terminology

- Instance = One running virtual machine.
- Instance Type = hardware configuration: cores, memory, disk.
- Instance Store Volume = Temporary disk associated with instance.
- Image (AMI) = Stored bits which can be turned into instances.
- Key Pair = Credentials used to access VM from command line.
- Region = Geographic location, price, laws, network locality.
- Availability Zone = Subdivision of region the is fault-independent.

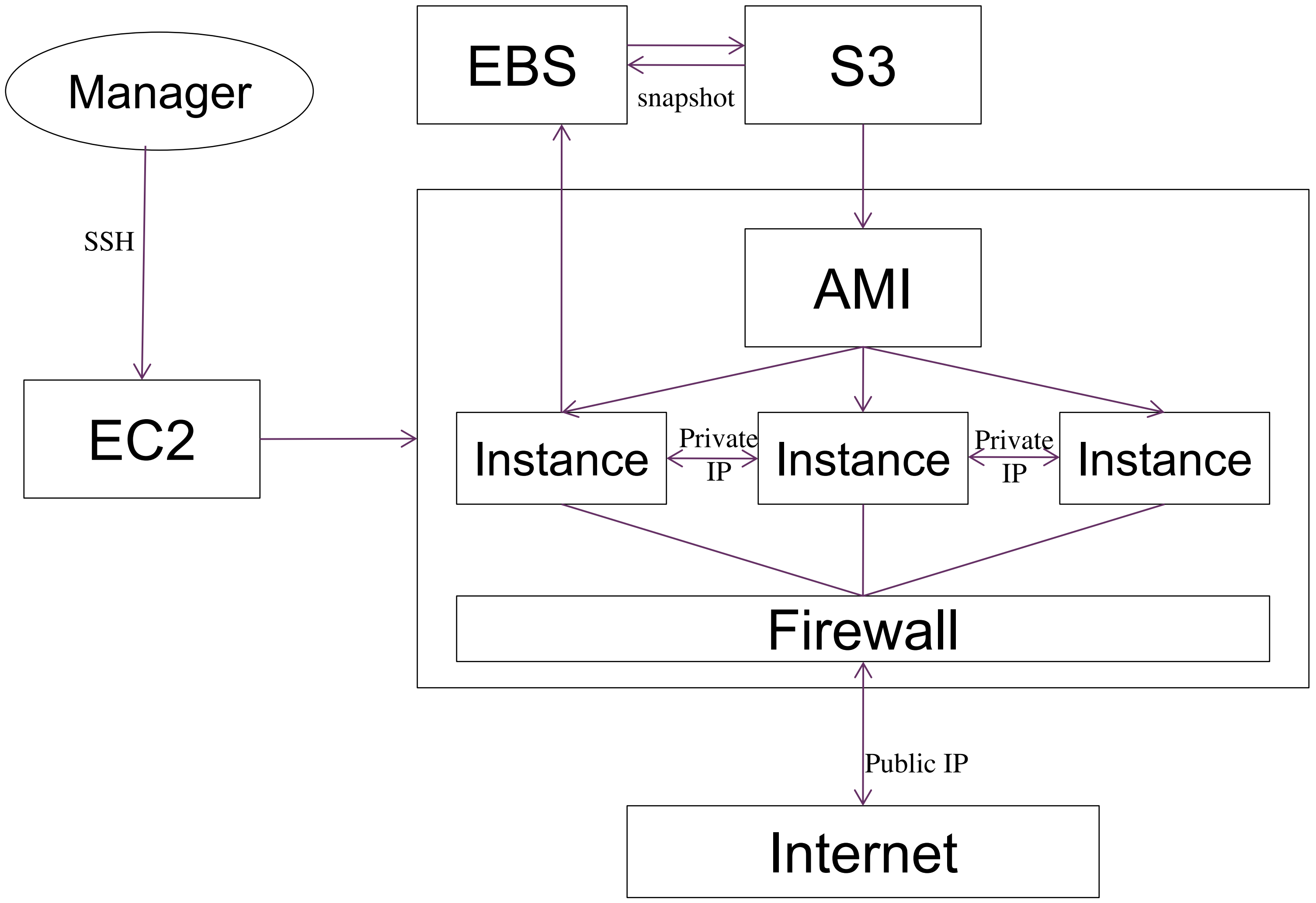
Centrum Wiskunde & Informatica

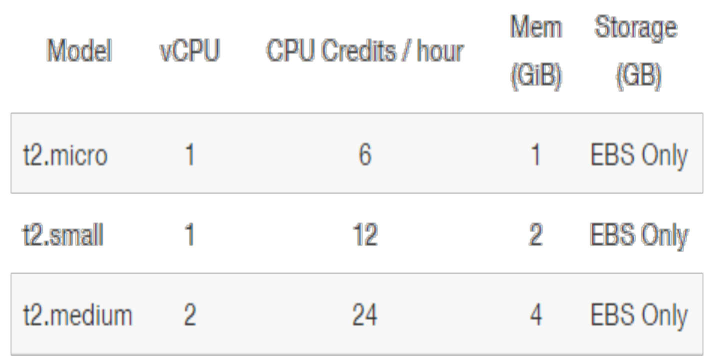
Amazon AWS


Your Application
Payment : Amazon FPS/ DevPay
Amazon SimpleDB Domains
Amazon SNS Topics
Amazon SQS Queues
Amazon RDS
Amazon Elastic MapReduce JobFlows
Auto-Scaling
Elastic LB
Cloud Watch
Amazon EC2 Instances (On-Demand, Reserved, Spot)
EBS Volumes
Amazon Virtual Private Cloud
Amazon S3 Objects and Buckets
Snapshots
Amazon Cloud Front
Amazon Global Physical Infrastructure (Geographical Regions, Availability Zones, Edge Locations)

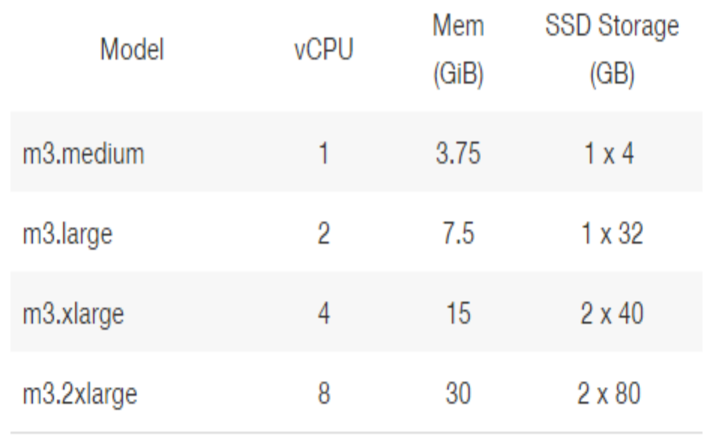
Centrum Wiskunde & Informatica

EC2 Architecture

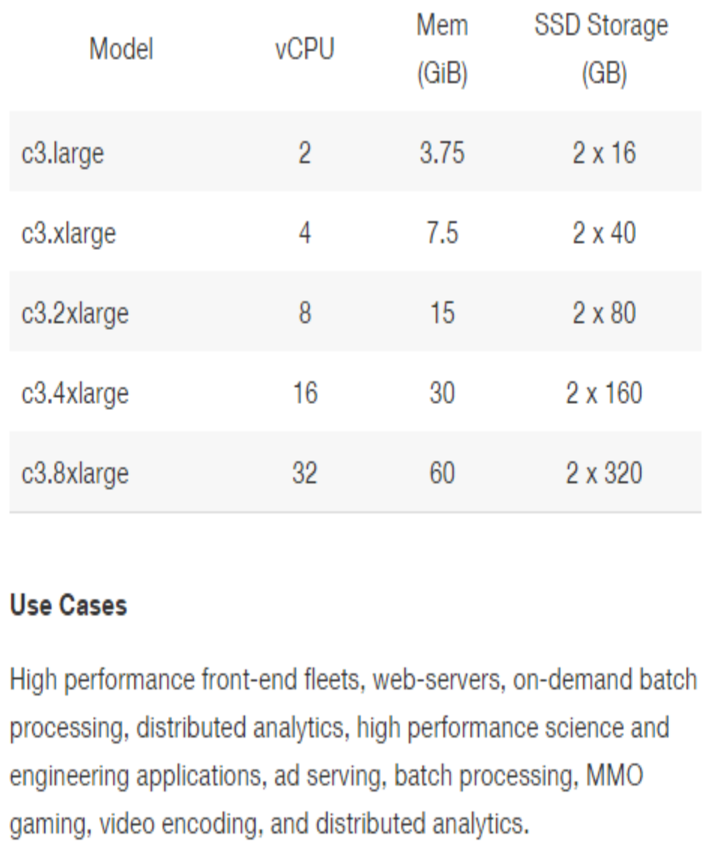




Model	vCPU	CPU Credits / hour	Mem (GiB)	Storage (GB)
t2.micro	1	6	1	EBS Only
t2.small	1	12	2	EBS Only
t2.medium	2	24	4	EBS Only



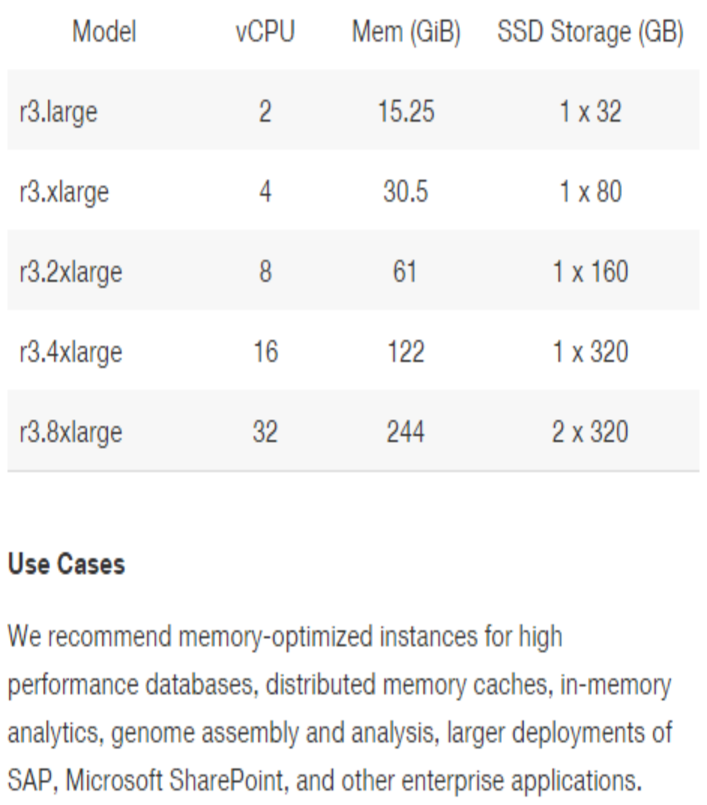
Model	vCPU	Mem (GiB)	SSD Storage (GB)
m3.medium	1	3.75	1 x 4
m3.large	2	7.5	1 x 32
m3.xlarge	4	15	2 x 40
m3.2xlarge	8	30	2 x 80



Model	vCPU	Mem (GiB)	SSD Storage (GB)
c3.large	2	3.75	2 x 16
c3.xlarge	4	7.5	2 x 40
c3.2xlarge	8	15	2 x 80
c3.4xlarge	16	30	2 x 160
c3.8xlarge	32	60	2 x 320

Use Cases

High performance front-end fleets, web-servers, on-demand batch processing, distributed analytics, high performance science and engineering applications, ad serving, batch processing, MMO gaming, video encoding, and distributed analytics.



Model	vCPU	Mem (GiB)	SSD Storage (GB)
r3.large	2	15.25	1 x 32
r3.xlarge	4	30.5	1 x 80
r3.2xlarge	8	61	1 x 160
r3.4xlarge	16	122	1 x 320
r3.8xlarge	32	244	2 x 320

Use Cases

We recommend memory-optimized instances for high performance databases, distributed memory caches, in-memory analytics, genome assembly and analysis, larger deployments of SAP, Microsoft SharePoint, and other enterprise applications.

z/bads

EC2 Pricing Model

- Free Usage Tier
- On-Demand Instances
  - Start and stop instances whenever you like, costs are rounded up to the nearest hour. (Worst price)
- Reserved Instances
  - Pay up front for one/three years in advance. (Best price)
  - Unused instances can be sold on a secondary market.
- Spot Instances
  - Specify the price you are willing to pay, and instances get started and stopped without any warning as the marked changes. (Kind of like Condor!)

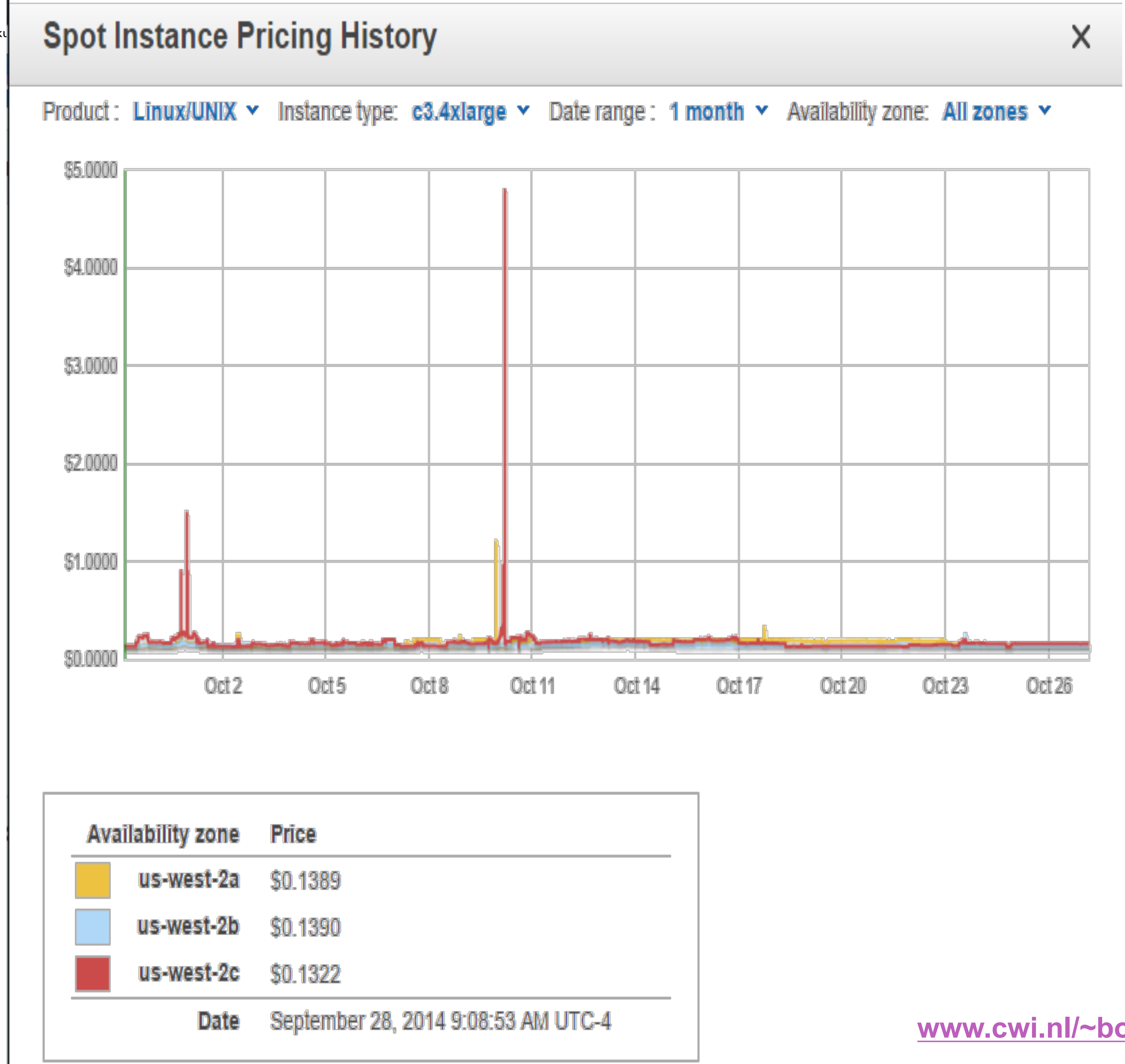
CWI
Centrum Wiskunde & Informatica

Free Usage Tier

- 750 hours of EC2 running Linux, RHEL, or SLES t2.micro instance usage
- 750 hours of EC2 running Microsoft Windows Server t2.micro instance usage
- 750 hours of Elastic Load Balancing plus 15 GB data processing
- 30 GB of Amazon Elastic Block Storage in any combination of General Purpose (SSD) or Magnetic, plus 2 million I/Os (with Magnetic) and 1 GB of snapshot storage
- 15 GB of bandwidth out aggregated across all AWS services
- 1 GB of Regional Data Transfer

CWI

Centrum Wisku



Spot Instance Pricing History

	Availability zone	Price
🟨	us-west-2a	$0.1389
🟦	us-west-2b	$0.1390
🟥	us-west-2c	$0.1322
	Date	September 28, 2014 9:08:53 AM UTC-4

Simple Storage Service (S3)

- A **bucket** is a container for objects and describes location, logging, accounting, and access control. A bucket can hold any number of **objects**, which are files of up to 5TB. A bucket has a name that must be **globally unique**.
- Fundamental operations corresponding to HTTP actions:
  - http://**bucket**.s3.amazonaws.com/**object**
  - POST a new object or update an existing object.
  - GET an existing object from a bucket.
  - DELETE an object from the bucket
  - LIST keys present in a bucket, with a filter.
- A bucket has a **flat directory structure** (despite the appearance given by the interactive web interface.)

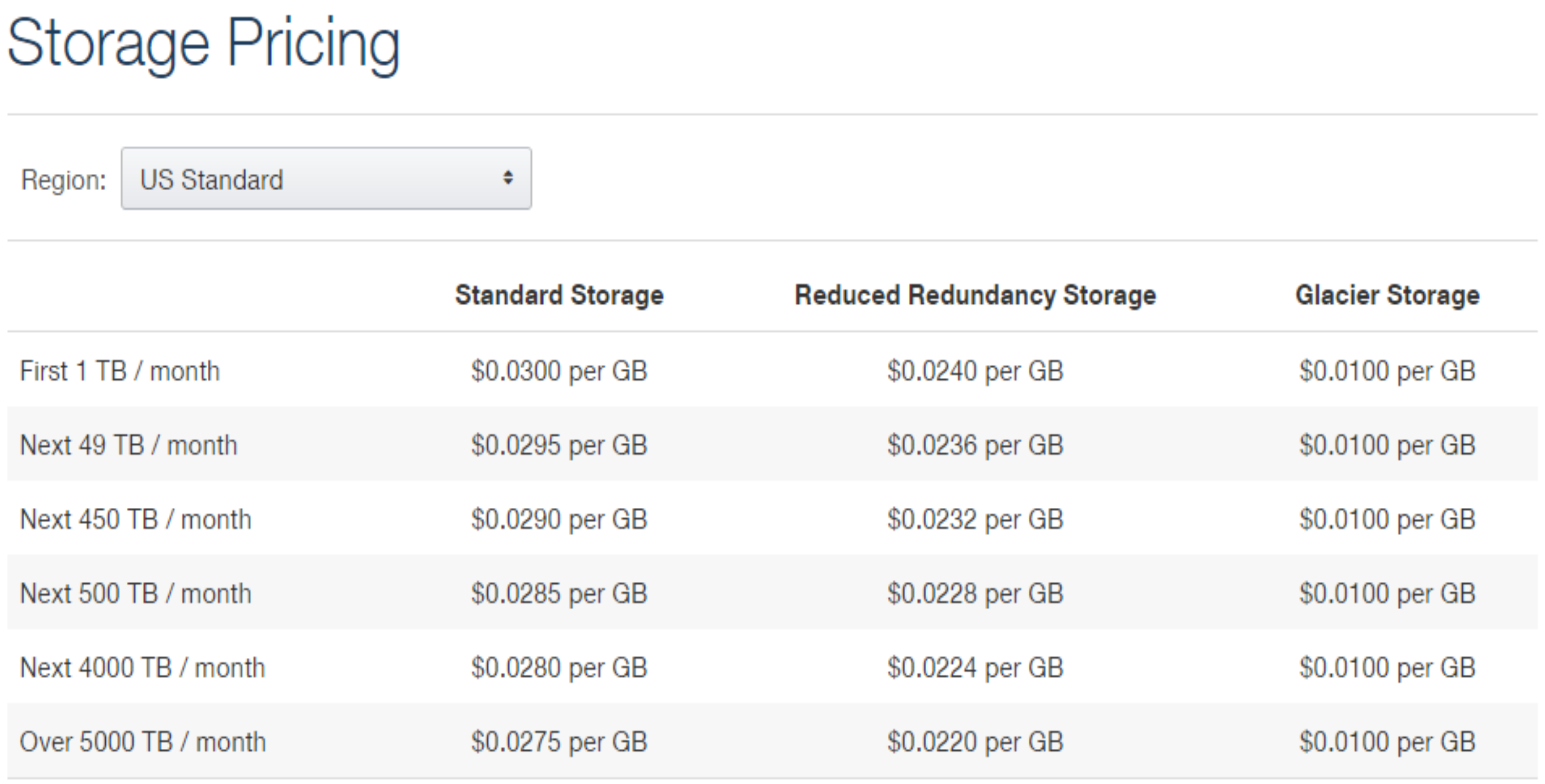
S3 Weak Consistency Model

Direct quote from the Amazon developer API:

“Updates to a single key are **atomic**….”

“Amazon S3 achieves high availability by replicating data across multiple servers within Amazon's data centers. If a PUT request is successful, your data is safely stored. However, information about the changes must replicate across Amazon S3, which can take some time, and so you might observe the following behaviors:

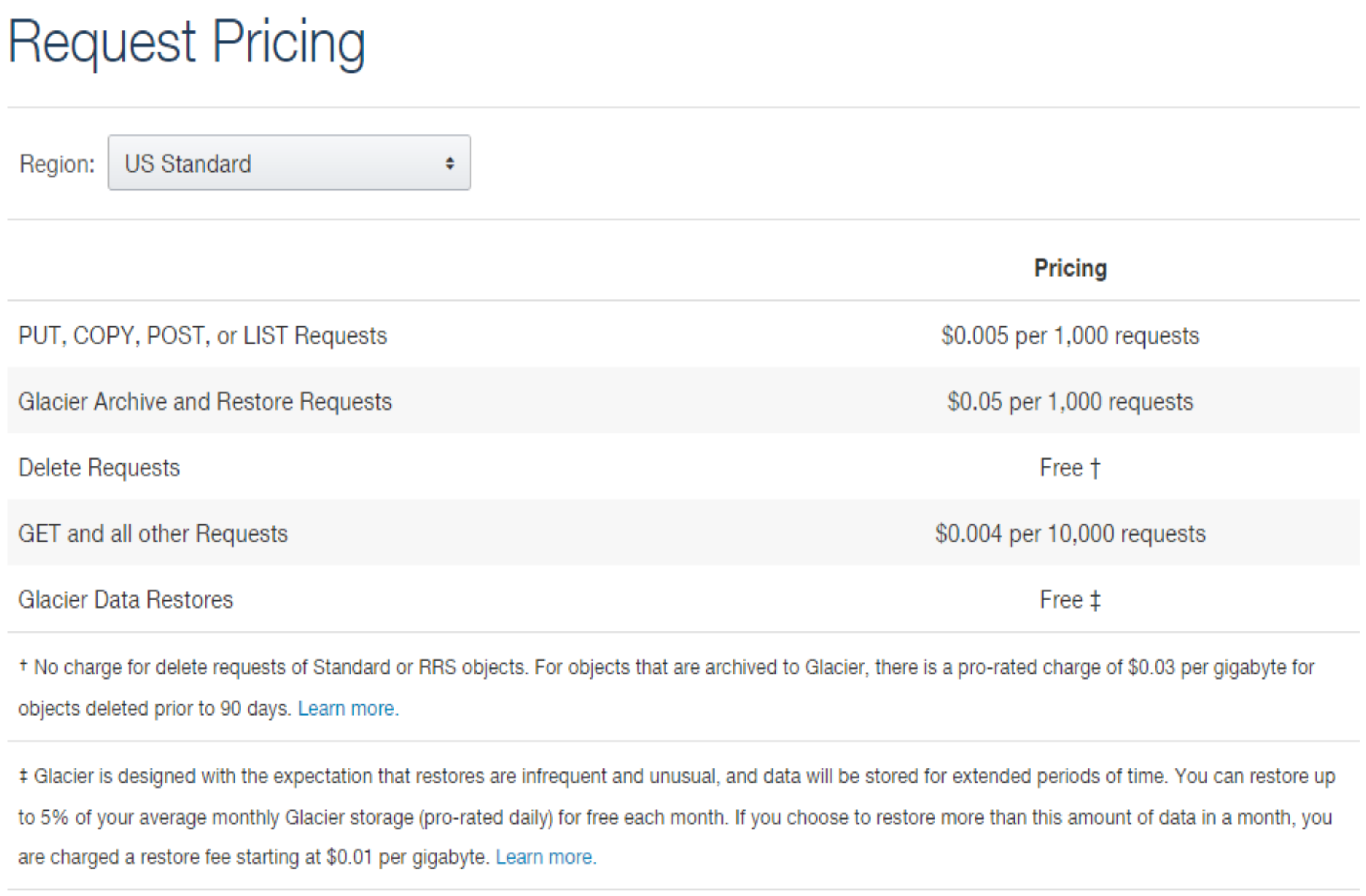
- A process writes a new object to Amazon S3 and immediately attempts to read it. Until the change is fully propagated, Amazon S3 might report "key does not exist."
- A process writes a new object to Amazon S3 and immediately lists keys within its bucket. Until the change is fully propagated, the object might not appear in the list.
- A process replaces an existing object and immediately attempts to read it. Until the change is fully propagated, Amazon S3 might return the prior data.
- A process deletes an existing object and immediately attempts to read it. Until the deletion is fully propagated, Amazon S3 might return the deleted data.”



Storage Pricing

Region: US Standard

	Standard Storage	Reduced Redundancy Storage	Glacier Storage
First 1 TB / month	$0.0300 per GB	$0.0240 per GB	$0.0100 per GB
Next 49 TB / month	$0.0295 per GB	$0.0236 per GB	$0.0100 per GB
Next 450 TB / month	$0.0290 per GB	$0.0232 per GB	$0.0100 per GB
Next 500 TB / month	$0.0285 per GB	$0.0228 per GB	$0.0100 per GB
Next 4000 TB / month	$0.0280 per GB	$0.0224 per GB	$0.0100 per GB
Over 5000 TB / month	$0.0275 per GB	$0.0220 per GB	$0.0100 per GB



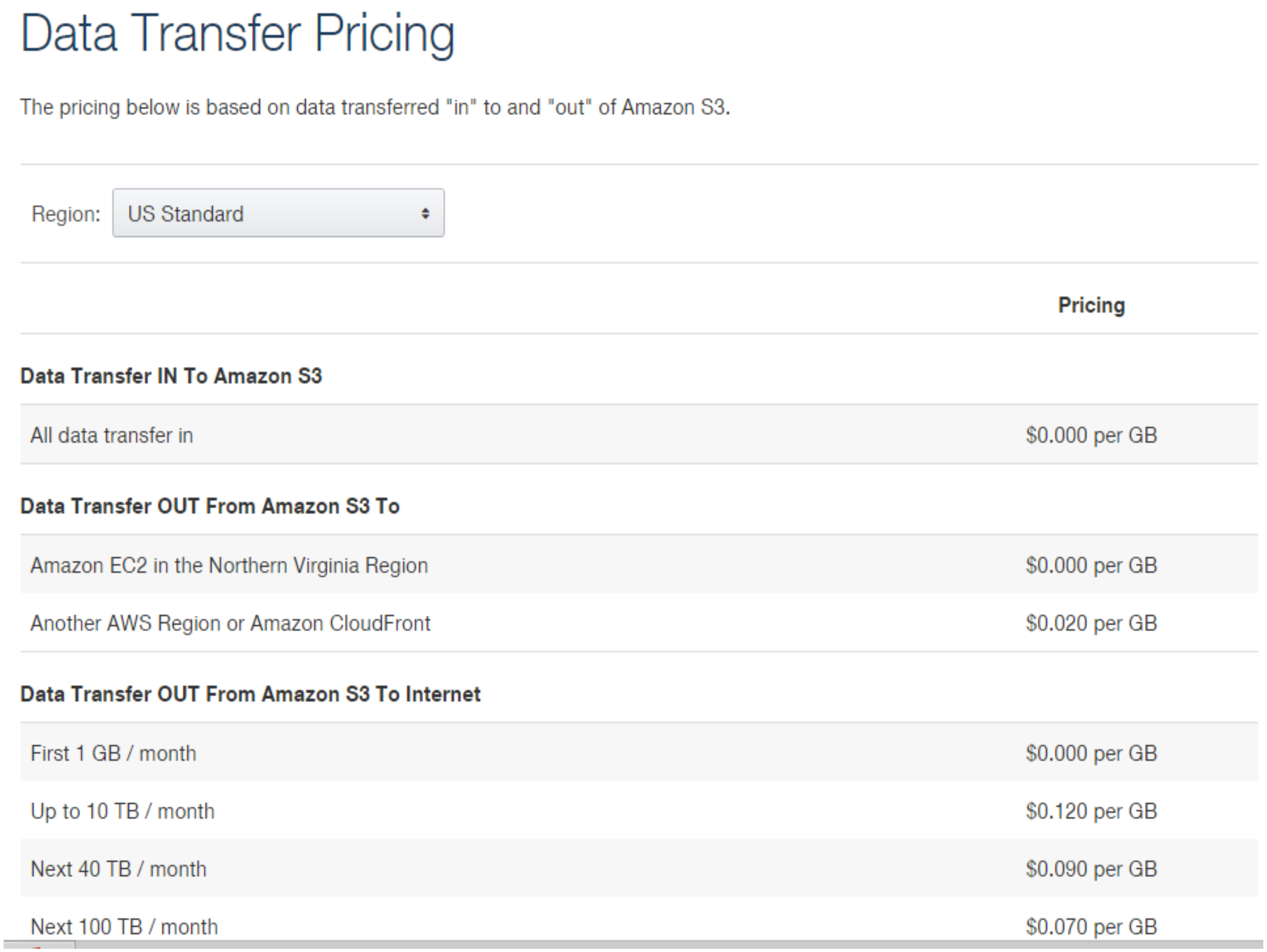
Request Pricing

Region: US Standard

	Pricing
PUT, COPY, POST, or LIST Requests	$0.005 per 1,000 requests
Glacier Archive and Restore Requests	$0.05 per 1,000 requests
Delete Requests	Free †
GET and all other Requests	$0.004 per 10,000 requests
Glacier Data Restores	Free ‡

† No charge for delete requests of Standard or RRS objects. For objects that are archived to Glacier, there is a pro-rated charge of $0.03 per gigabyte for objects deleted prior to 90 days. Learn more.

‡ Glacier is designed with the expectation that restores are infrequent and unusual, and data will be stored for extended periods of time. You can restore up to 5% of your average monthly Glacier storage (pro-rated daily) for free each month. If you choose to restore more than this amount of data in a month, you are charged a restore fee starting at $0.01 per gigabyte. Learn more.



Data Transfer Pricing

The pricing below is based on data transferred "in" to and "out" of Amazon S3.

Region: US Standard

	Pricing
Data Transfer IN To Amazon S3	
All data transfer in	$0.000 per GB
Data Transfer OUT From Amazon S3 To	
Amazon EC2 in the Northern Virginia Region	$0.000 per GB
Another AWS Region or Amazon CloudFront	$0.020 per GB
Data Transfer OUT From Amazon S3 To Internet	
First 1 GB / month	$0.000 per GB
Up to 10 TB / month	$0.120 per GB
Next 40 TB / month	$0.090 per GB
Next 100 TB / month	$0.070 per GB

Elastic Block Store

- An EBS volume is a **virtual disk** of a fixed size with a block read/write interface. It can be **mounted** as a filesystem on a running EC2 instance where it can be **updated incrementally.** Unlike an instance store, an EBS volume is **persistent.**
- (Compare to an S3 object, which is essentially a file that must be accessed in its entirety.)
- Fundamental operations:
  - CREATE a new volume (1GB-1TB)
  - COPY a volume from an existing EBS volume or S3 object.
  - MOUNT on one instance at a time.
  - SNAPSHOT current state to an S3 object.

S3 Weak Consistency Model

Direct quote from the Amazon developer API:

“Updates to a single key are **atomic**….”

“Amazon S3 achieves high availability by replicating data across multiple servers within Amazon's data centers. If a PUT request is successful, your data is safely stored. However, information about the changes must replicate across Amazon S3, which can take some time, and so you might observe the following behaviors:

- A process writes a new object to Amazon S3 and immediately attempts to read it. Until the change is fully propagated, Amazon S3 might report "key does not exist."
- A process writes a new object to Amazon S3 and immediately lists keys within its bucket. Until the change is fully propagated, the object might not appear in the list.
- A process replaces an existing object and immediately attempts to read it. Until the change is fully propagated, Amazon S3 might return the prior data.
- A process deletes an existing object and immediately attempts to read it. Until the deletion is fully propagated, Amazon S3 might return the deleted data.”

Elastic Block Store

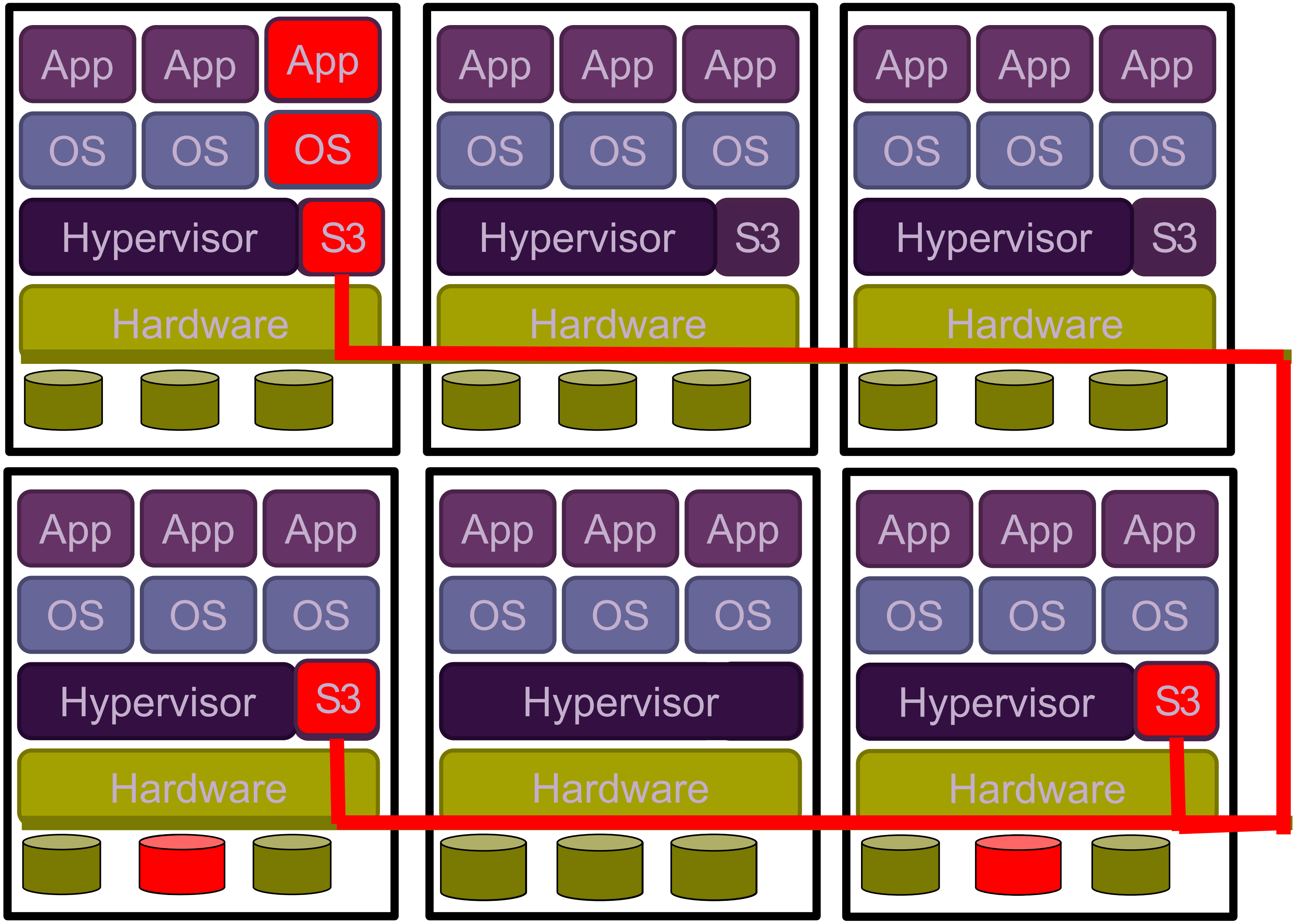
- An EBS volume is a **virtual disk** of a fixed size with a block read/write interface. It can be **mounted** as a filesystem on a running EC2 instance where it can be **updated incrementally.** Unlike an instance store, an EBS volume is **persistent.**
- (Compare to an S3 object, which is essentially a file that must be accessed in its entirety.)
- Fundamental operations:
  - CREATE a new volume (1GB-1TB)
  - COPY a volume from an existing EBS volume or S3 object.
  - MOUNT on one instance at a time.
  - SNAPSHOT current state to an S3 object.

Centrum Wiskunde & Informatica

EBS is approx. 3x more expensive by volume and 10x more expensive by IOPS than S3.

CWI
Centrum Wiskunde & Informatica

How does storage work on AWS?



How does AWS work

- S3 the cornerstone for reliable storage
  - Data blocks are replicated over the cluster
  - All (many) nodes in the Amazon clusters contribute their disk to S3
    - S3 is not the only Amazon service on the machines
    - Instead of ‘hypervisor’ likely a barebones Linux-a-like OS
- File-system (EBS and also HDFS) is an abstraction on top S3
  - if nodes go down or instances hang, the data is safe in S3
  - disk access speed depends on network and caching (!)
- Private disk storage is an expensive extra
  - One can get access to dedicated (flash) disk, for controllable I/O
  - Amazon makes no persistence or reliability guarantees (when you boot, it is empty). Makes it hard to use.

Use Glacier for Cold Data

- Glacier is structured like S3: a **vault** is a container for an arbitrary number of **archives**. Policies, accounting, and access control are associated with vaults, while an archive is a single object.
- However:
  - All operations are asynchronous and notified via SNS.
  - Vault listings are updated once per day.
  - Archive downloads may take up to four hours.
  - Only 5% of total data can be accessed in a given month.
- Pricing:
  - Storage: $0.01 per GB-month
  - Operations: $0.05 per 1000 requests
  - Data Transfer: Like S3, free within AWS.
- S3 Policies can be set up to automatically move data into Glacier.

CWI
Centrum Wiskunde & Informatica

Summary

- Utility Computing – Cloud Computing for rent
- Cloud Computing infrastructures
  - Virtualization
  - Caching, Replication, Sharding
  - Eric Brewer’ CAP theorem: can’t have:
    - Consistency & Availability & Partition-tolerance
- Amazon Web Services tour
  - EC2: Instance types, pricing models, & other terminology
  - Storage: EBS/S3/Glacier