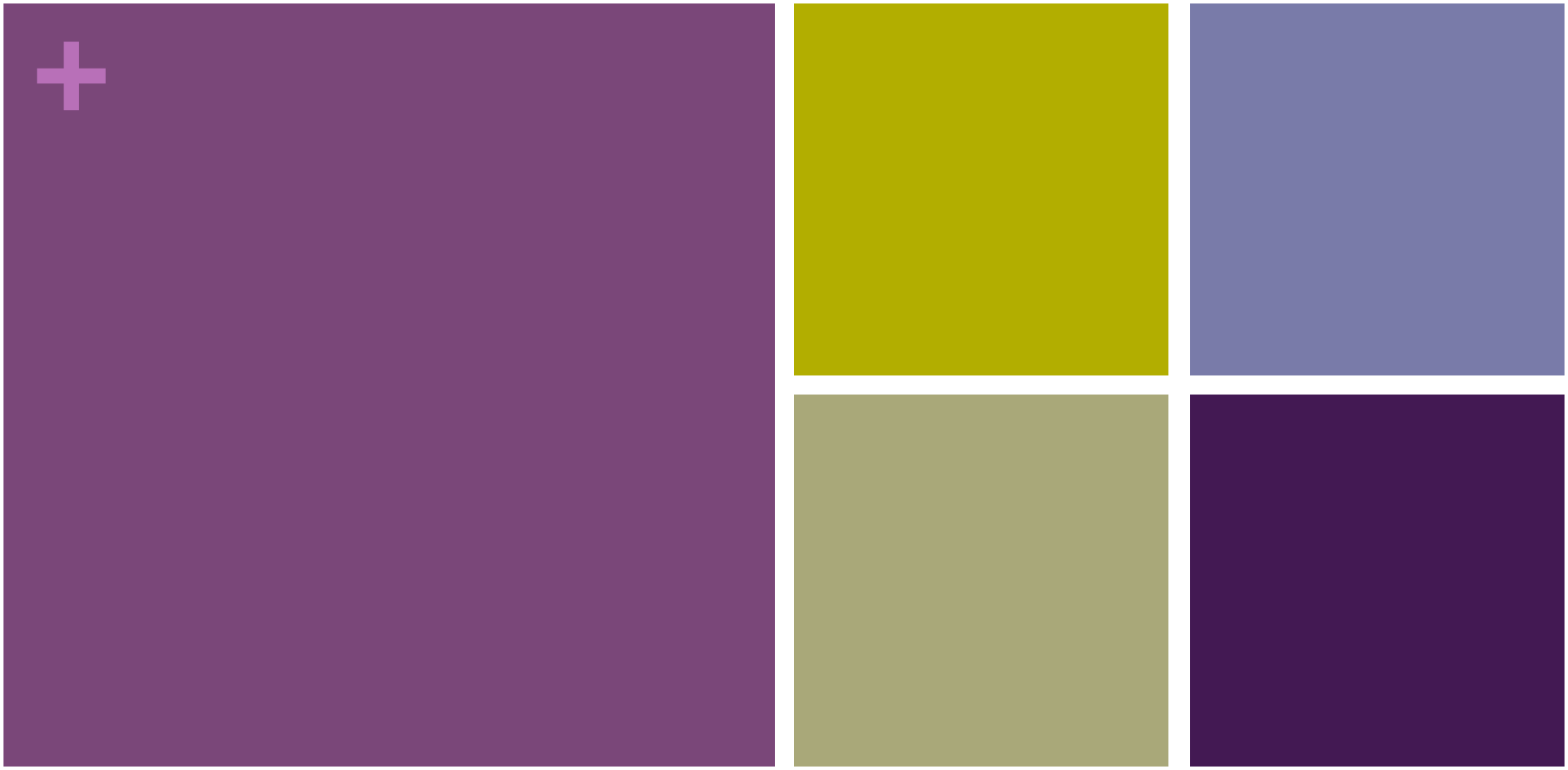


CUDA Assignment, Code Examples and Scaling your App

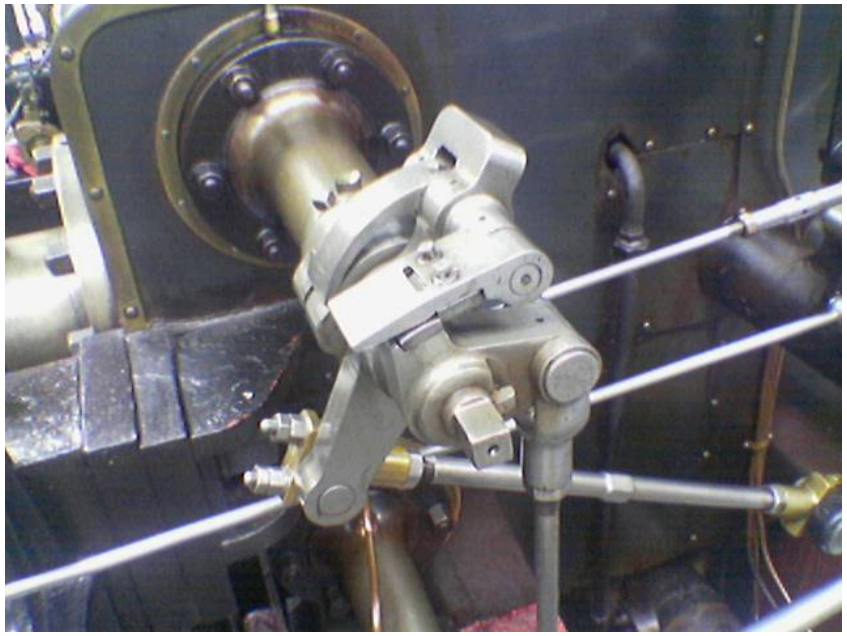
Abhijit Bendale (abendale@vast.uccs.edu)

04/08/2014

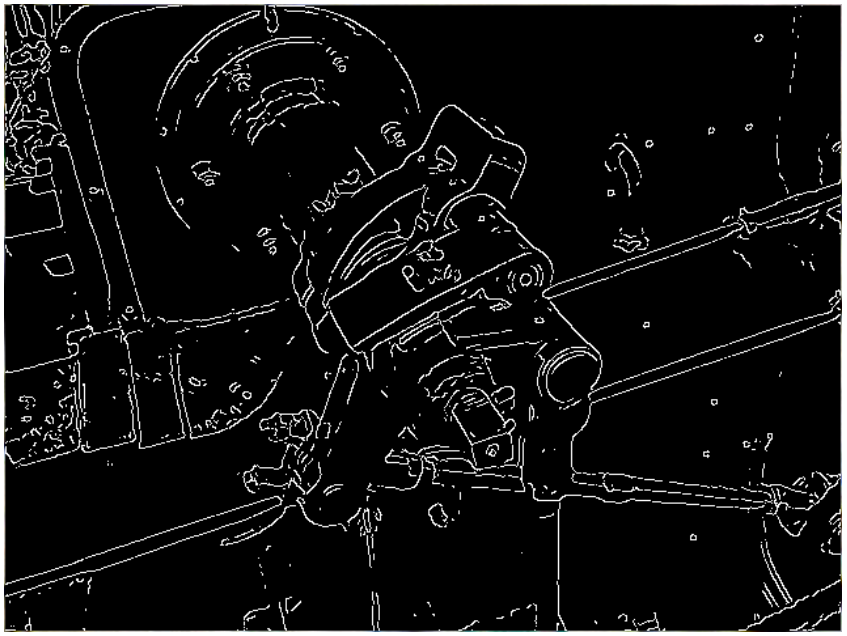
Today's Agenda

- Discussion on CUDA Assignment
- Code example
  - Detailed look at multiple examples from CUDA SDK
- Scaling up your application
  - Amazon EC2, Amazon S3, Auto-Scale
  - Map-Reduce, Apache Hadoop

CUDA Assignment: Implement Canny Edge Detection



Original Image

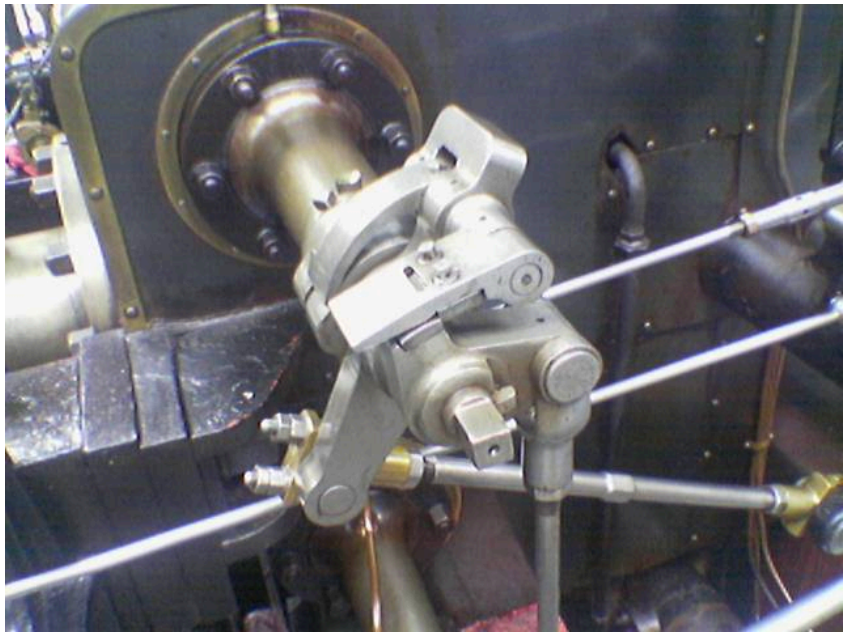


Canny Edge Detection

Steps for Canny Edge Detection: Noise Reduction

$$\mathbf{B} = \frac{1}{159}\begin{bmatrix} 2 & 4 & 5 & 4 & 2 \\ 4 & 9 & 12 & 9 & 4 \\ 5 & 12 & 15 & 12 & 5 \\ 4 & 9 & 12 & 9 & 4 \\ 2 & 4 & 5 & 4 & 2 \end{bmatrix} * \mathbf{A}.$$

Noise Reduction



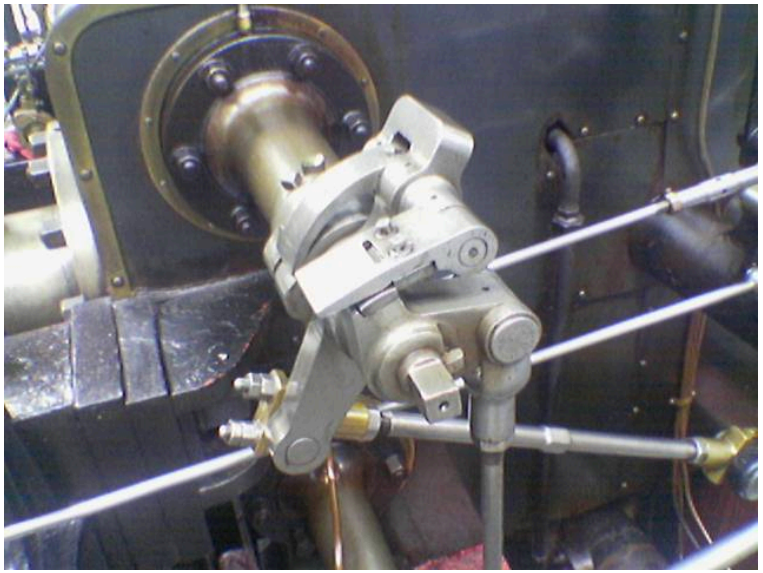
A = Image

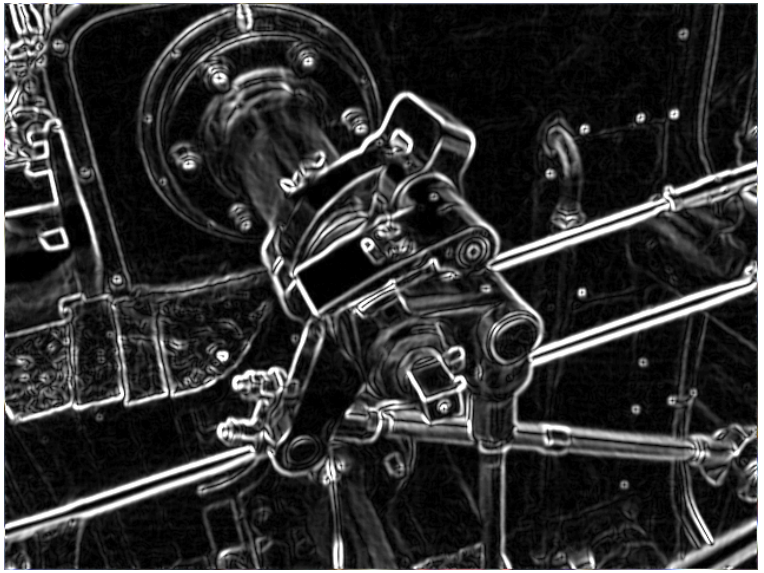
Sobel Filter

$$\mathbf{G}_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} * \mathbf{A} \quad \text{and} \quad \mathbf{G}_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * \mathbf{A}$$

Gradient in X direction

Gradient in Y direction





$$\mathbf{G} = \sqrt{\mathbf{G}_x^{\,2} + \mathbf{G}_y^{\,2}}$$

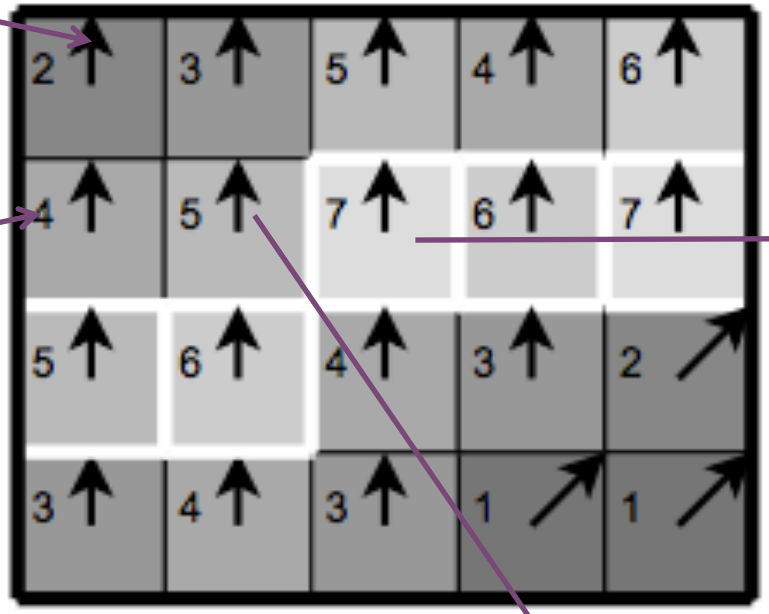
$$\mathbf{\Theta} = \operatorname{atan2}(\mathbf{G}_y, \mathbf{G}_x)$$

Round up angles to (0, 45, 90, 135 degrees angle)

Non-Max Supression

Angle

Gradient Value



Angle = 90
Check top/bottom gradients. In this case 7 is greater. Hence an edge

Angle = 90
Check top/bottom gradients. In this case 5 is not greater than top/bottom Hence not an edge

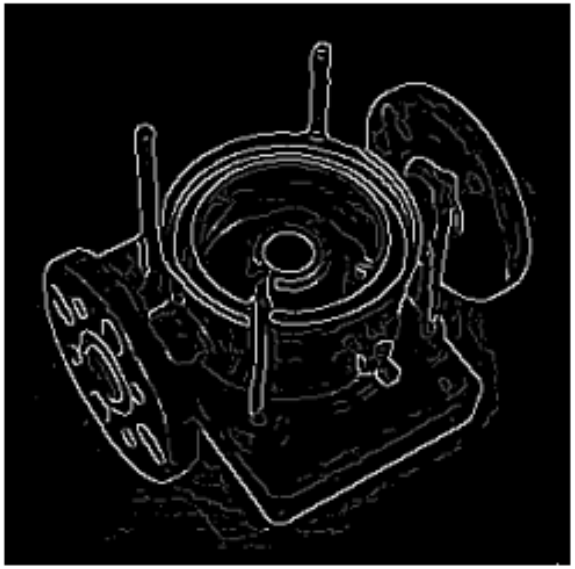
E.g. output of Non-Max Suppression

(a) Gradient values

(b) Edges after non-maximum suppression

Hysteresis (Double) Thresholding

(a) Edges after non-maximum suppression



(b) Double thresholding

Use high and low threshold to determine ideal thickness for edge lines. This is a bit empirical process. It is possible that you wont be able to perform high level of parallelization for this step. Give it a try.

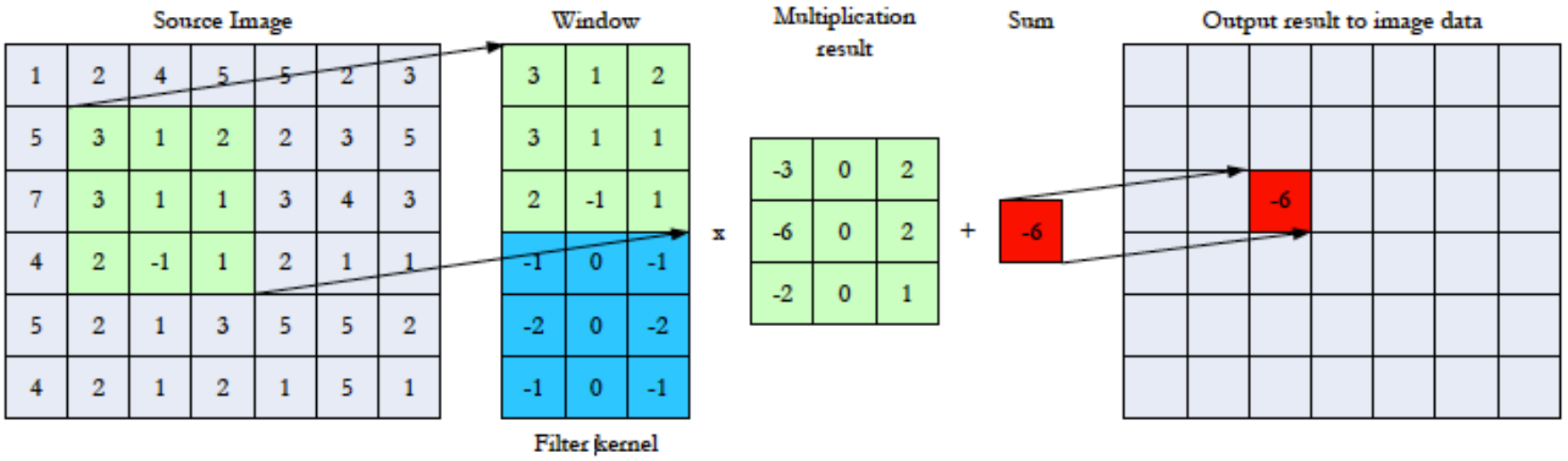
Code base given to you

- Computation of Sobel Edge Filter
- Files
  - SobelFilter.cpp → overall I/O is handled here
  - SobelFilter_kernels.cu → Splitting of image in chunks and computing sobel filter on these chunks
    - sobelFilter() is the global wrapper function
    - ComputeSobel() is where horizontal and vertical filters are defined. __device__ function. Only convolution happens on GPU.
    - SobelShared() splits the image into multiple chunks
  - SobelFilter_kernels.h → header file
  - Makefile → library linking
- Feel free to rename the files, create more files etc.
- Note the different function scope identifiers: __global__, __device__,
- Works on both Linux (command line: type "make") and Windows (Visual Studio). Open visual studio project file

About the code

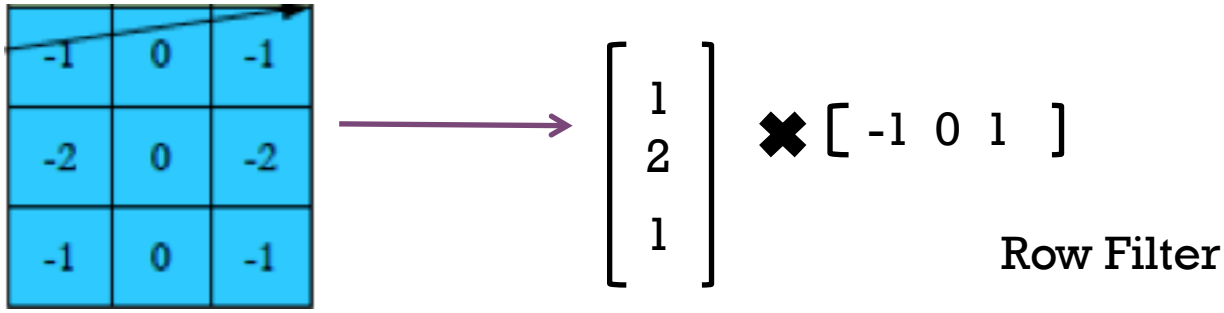
- Make sure you understand data types used:
  - Pixel is just an unsigned char (standard way of representing intensity values in images)
  - Image is organized as texture: which is a 2D vector in C++ (texture<unsigned char, 2> tex)
  - setupTexture allocates memory in the device
  - Contains 2 ways to implement:
    - SobelTex() → Doesn't use shared memory
    - SobelShared() → uses shared memory
- Feel free to get inspiration from existing open-source canny edge detection code.
- Finally.. All the best for your assignment..!

Convolution Separable



Goal: Given a filter kernel, compute convolution with matrix/image

Separable Filters



A separable filter can be divided into 2 consecutive filter operations. They offer flexibility in implementation and reduce mathematical complexity.

Apply row filter and column filter separately

3_Imaging/convolutionSeparable

- main.cpp: main program, allocating host and device memory, generating input data, issuing CUDA computations
- convolutionSeparable.cu: CUDA convolution kernels (contains row and column kernels)
- convolutionSeparable_gold.cpp: reference CPU separable convolution implementation, which is used to validate results from CUDA

Seperable Convolution

3_Imaging/convolutionSeparable/main.cpp

```
for (int i = -1; i < iterations; i++)
    {

        convolutionRowsGPU(
            d_Buffer,
            d_Input,
            imageW,
            imageH
        );

        convolutionColumnsGPU(
            d_Output,
            d_Buffer,
            imageW,
            imageH
        );
    }
```

Output of Rows

Becomes input of columns

```
__global__ void convolutionRowsKernel(
    float *d_Dst,
    float *d_Src,
    int imageW,
    int imageH,
    int pitch
)
```

```
__global__ void convolutionColumnsKernel(
    float *d_Dst,
    float *d_Src,
    int imageW,
    int imageH,
    int pitch
)
```

convolutionSeparable.cu

Histogram

Goal: Given an image, compute its histogram

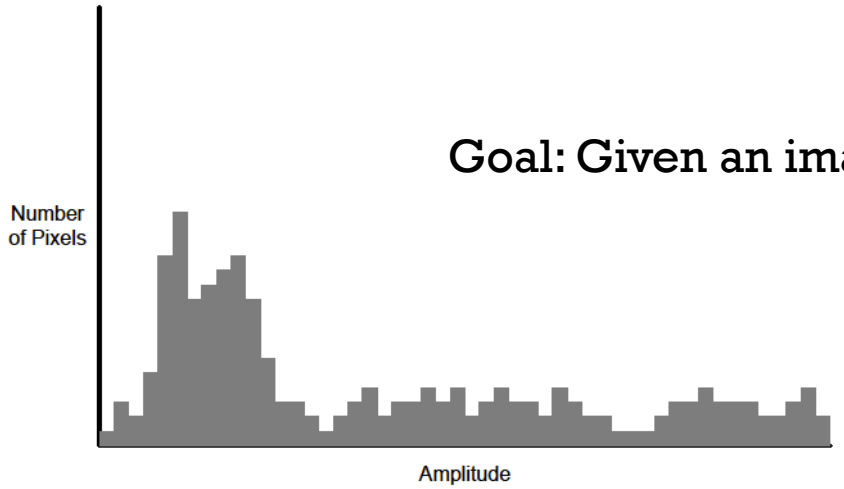


Figure 1: An example of an image histogram

```
for(int i = 0; i < BIN_COUNT; i++)
    result[i] = 0;

for(int i = 0; i < dataN; i++)
    result[data[i]]++;
```

Listing 1. Histogram calculation on a single-threaded device. (pseudocode)

Parallelizing Histograms

- Subdivision of input data array between execution threads
- Processing of the sub-arrays by each dedicated execution thread and sorting the result into a certain number of sub-histograms
- Merge sub histograms into a single histogram

```
extern "C" void histogram64(
    uint *d_Histogram,
    void *d_Data,
    uint byteCount
)
{
    const uint histogramCount = iDivUp(byteCount, HISTOGRAM64_THREADBLOCK_SIZE * iSnapDown(255, sizeof(data_t)));

    assert(byteCount % sizeof(data_t) == 0);
    assert(histogramCount <= MAX_PARTIAL_HISTOGRAM64_COUNT);

    histogram64Kernel<<<histogramCount, HISTOGRAM64_THREADBLOCK_SIZE>>>(
        d_PartialHistograms,
        (data_t *)d_Data,
        byteCount / sizeof(data_t)
    );
    getLastCudaError("histogram64Kernel() execution failed\n");

    mergeHistogram64Kernel<<<HISTOGRAM64_BIN_COUNT, MERGE_THREADBLOCK_SIZE>>>(
        d_Histogram,
        d_PartialHistograms,
        histogramCount
    );
    getLastCudaError("mergeHistogram64() execution failed\n");
}
```

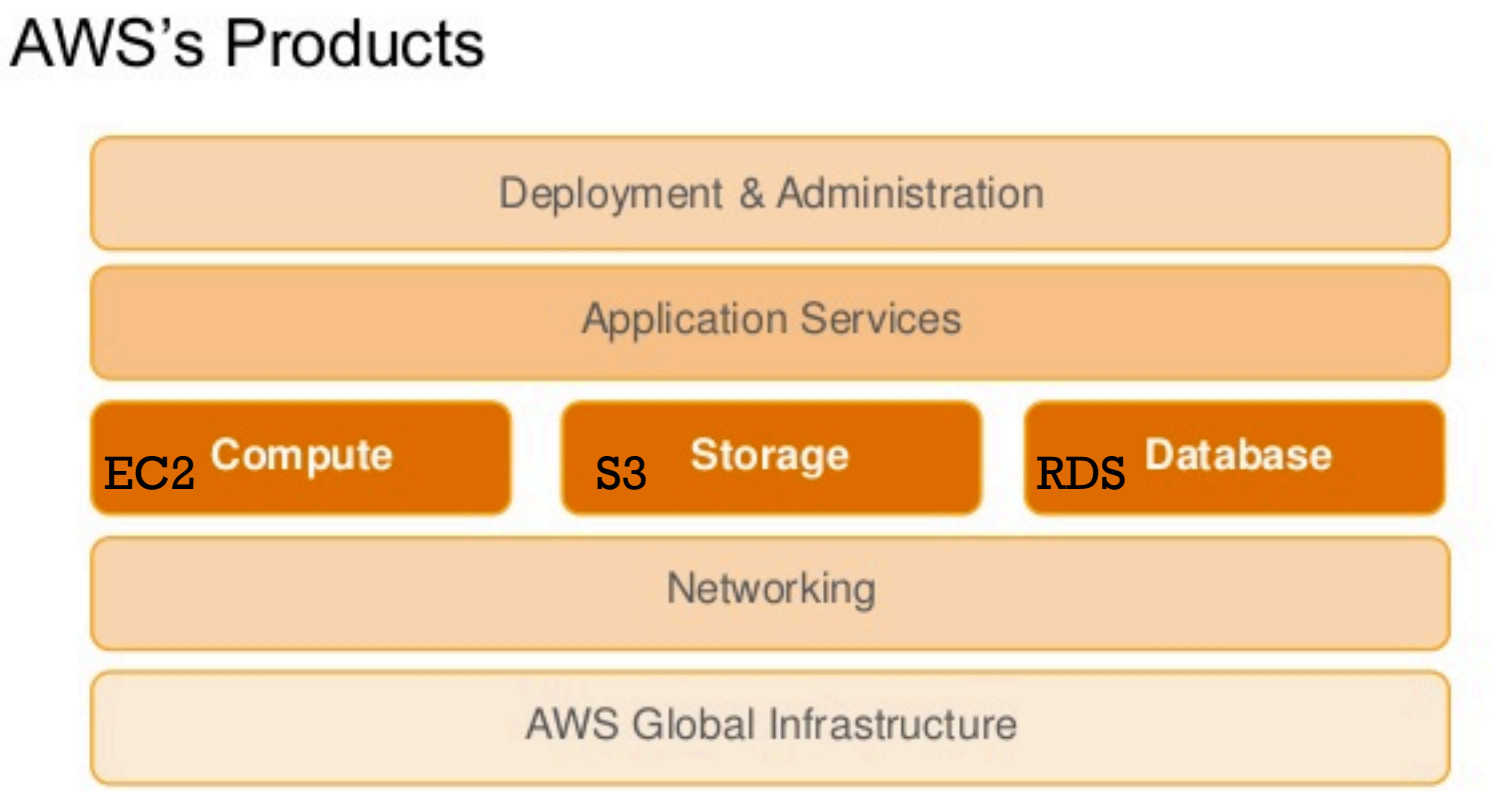
```
__global__ void mergeHistogram64Kernel(
    uint *d_Histogram,
    uint *d_PartialHistograms,
    uint histogramCount
)
{
    __shared__ uint data[MERGE_THREADBLOCK_SIZE];

    uint sum = 0;

    for (uint i = threadIdx.x; i < histogramCount; i += MERGE_THREADBLOCK_SIZE)
    {
        sum += d_PartialHistograms[blockIdx.x + i * HISTOGRAM64_BIN_COUNT];
    }

    data[threadIdx.x] = sum;

    for (uint stride = MERGE_THREADBLOCK_SIZE / 2; stride > 0; stride >>= 1)
    {
        __syncthreads();

        if (threadIdx.x < stride)
        {
            data[threadIdx.x] += data[threadIdx.x + stride];
        }
    }

    if (threadIdx.x == 0)
    {
        d_Histogram[blockIdx.x] = data[0];
    }
}
```

Scaling up your application

Agenda

- Amazon EC2: Scaling Computation, Auto-Scaling
- Amazon S3: Scaling Storage
- Maintaining Large Databases
- Hadoop/MapReduce
- Cassandra, Mongdob, Elasticsearch
- Cost associated with Scaling

Amazon Web Services



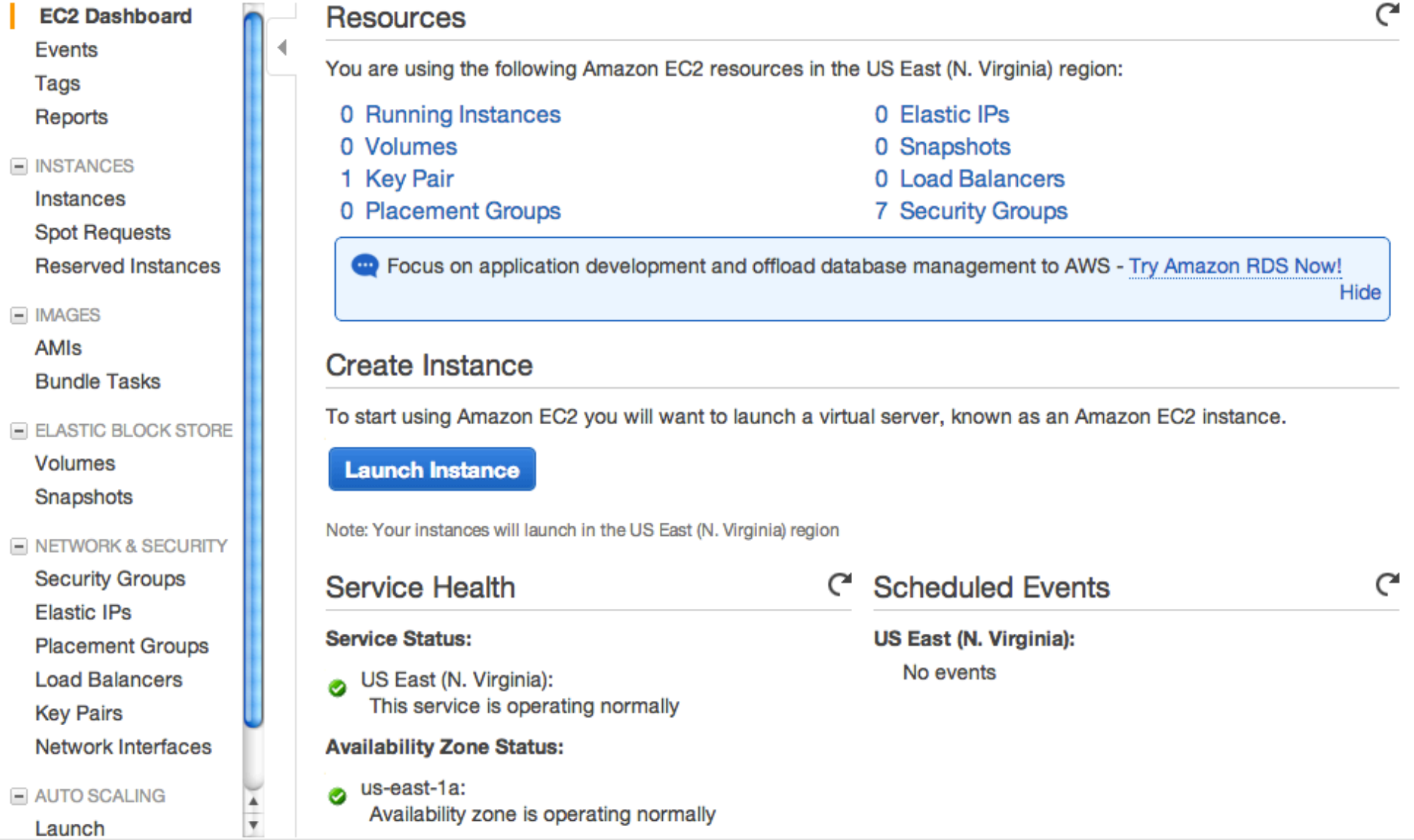
Amazon EC2

- Amazon Elastic Compute Cloud provides compute capacity in the cloud
- EC2 allows users to rent virtual computers on which to run their own computer applications.
- EC2 allows scalable deployment of applications by providing a web service and API
- EC2 provides users control over geographic location for better optimization

Operating Systems		
CentOS	Debian	SUSE Linux Enterprise
Amazon Linux	Oracle Enterprise Linux	Ubuntu
Red Hat Enterprise Linux	Windows Server	

API Name	Memory (GiB)	Cores/Compute Units	Instance Storage (GB)	32bit/64bit	I/O Performance	EBS-Optimizable	Cost (Linux - per hour on US East 1)
m1.small	1.7	1/1	160	32/64	Moderate	No	$0.060
m1.medium	3.75	1/2	410	32/64	Moderate	No	$0.120
m1.large	7.5	2/4	850	64	Moderate	500 Mbit/s	$0.240
m1.xlarge	15	4/8	1600	64	High	1000 Mbit/s	$0.480
m3.xlarge	15	4/13	0 (EBS only)	64	Moderate	500 Mbit/s	$0.500
m3.2xlarge	30	8/26	0 (EBS only)	64	High	1000 Mbit/s	$1.000
t1.micro	0.6	1/(up to 2)	0 (EBS only)	32/64	Low	No	$0.020
m2.xlarge	17.1	2/6.5	420	64	Moderate	No	$0.410
m2.2xlarge	34.2	4/13	850	64	High	500 Mbit/s	$0.820
m2.4xlarge	68.4	8/26	1690	64	High	1000 Mbit/s	$1.640
c1.medium	1.7	2/5	350	32/64	Moderate	No	$0.145
c1.xlarge	7	8/20	1690	64	High	1000 Mbit/s	$0.580
cc1.4xlarge	23	2/33.5 (2 Intel Xeon X5570)	1690	64	Very High (10 Gbit)	?	$1.300
cc2.8xlarge	60.5	2/88 (2 Intel Xeon E5-2670)	3370	64	Very High (10 Gbit)	Not necessary	$2.400
cr1.8xlarge	244	2/88 (2 Intel Xeon E5-2670)	240 (SSD)	64	Very High (10 Gbit)	Not necessary	$3.500
cg1.4xlarge	22	2/33.5 (2 Intel Xeon X5570) + 2 NVIDIA Tesla "Fermi" M2050 GPU	1960	64	Very High (10 Gbit)	Not necessary	$2.100
hi1.4xlarge	60.5	16/35 (8 cores + 8 hyperthreads)	2*1024 (SSD)	64	Very High (10 Gbit)	Not necessary	$3.100
hs1.8xlarge	117	16/35 (8 cores + 8 hyperthreads)	48000 (24 * 2TB drives)	64	Very High (10 Gbit)	Not necessary	$4.600

Launching EC2 instance

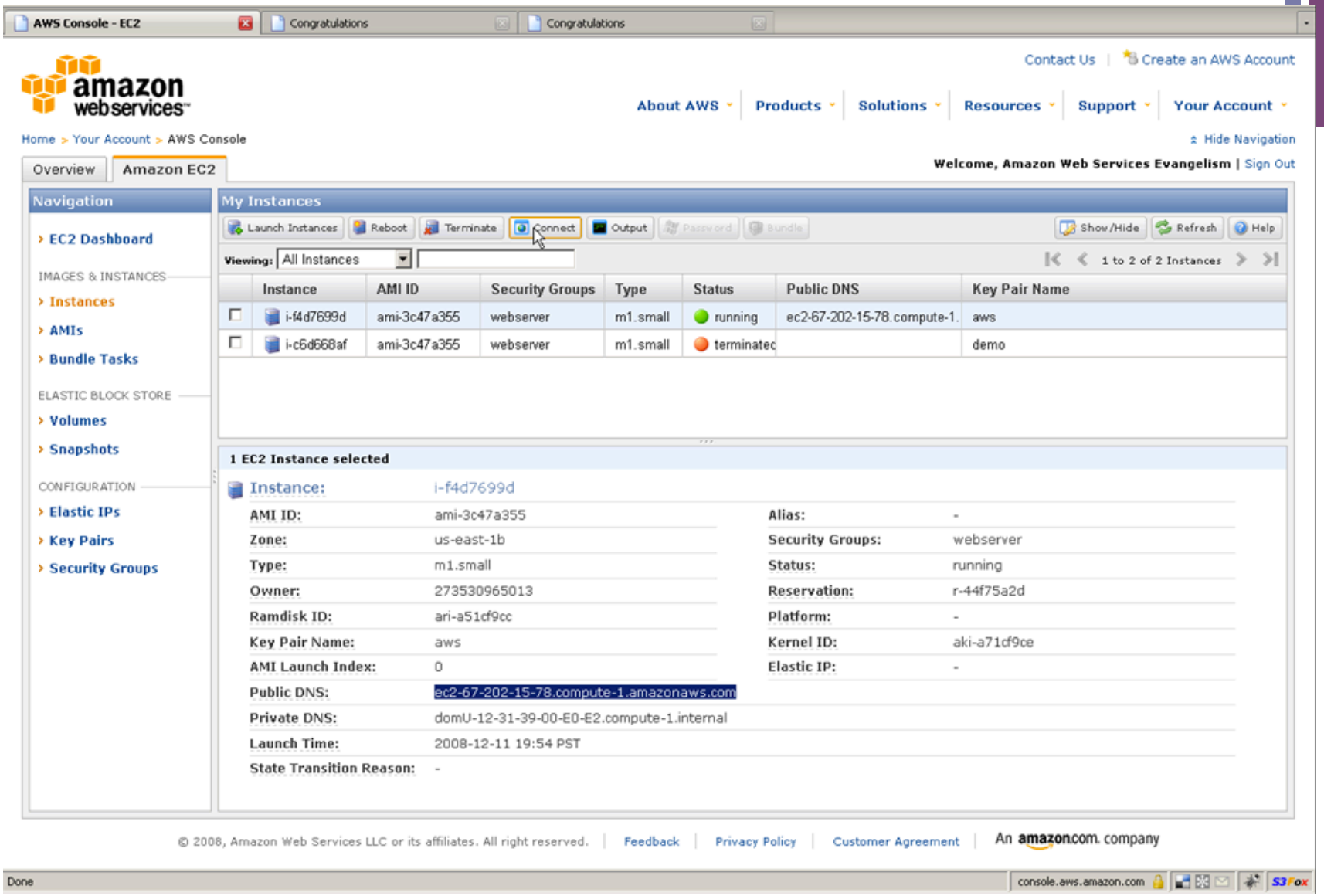


Step 3: Configure Instance Details

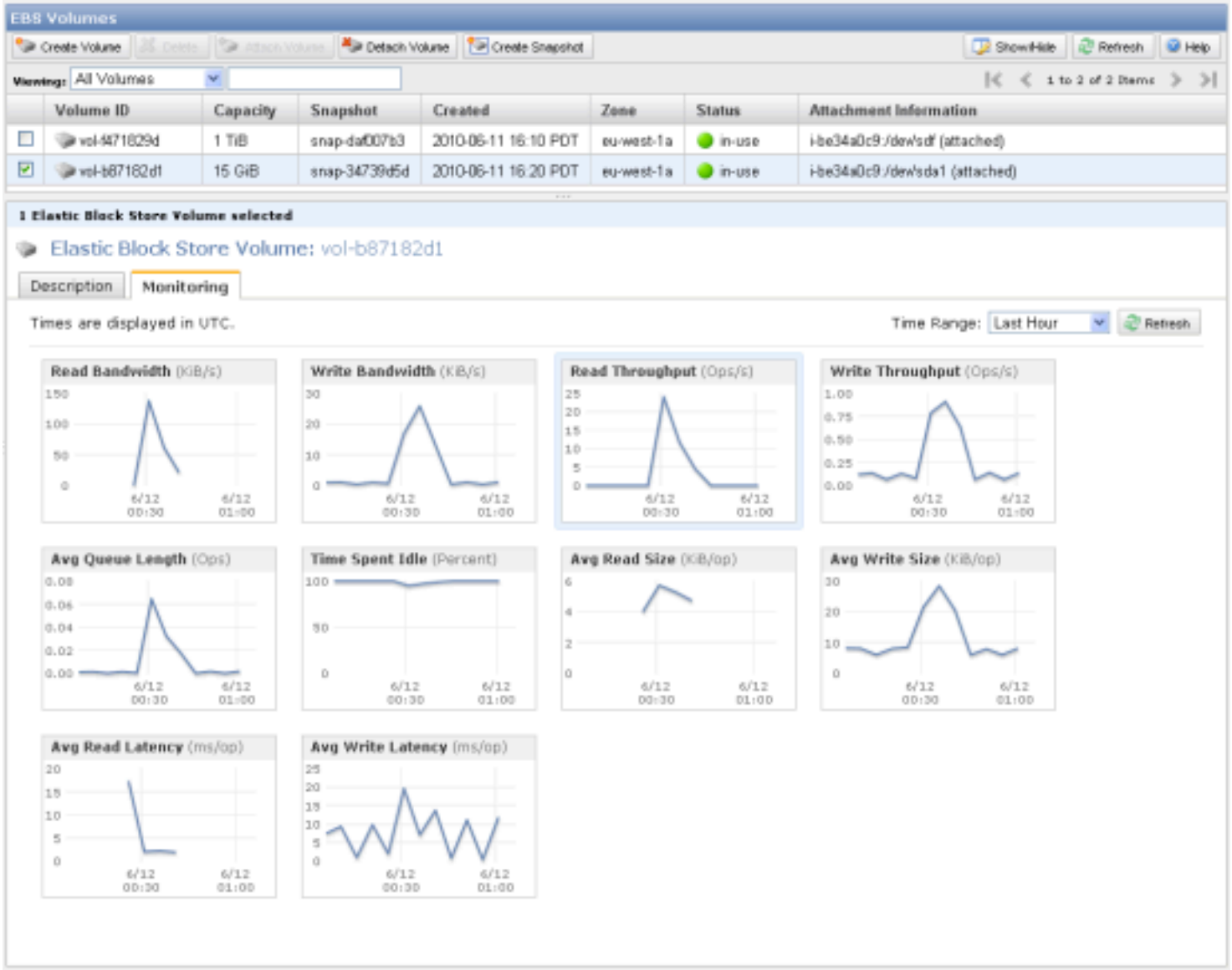
Configure the instance to suit your requirements. You can launch multiple instances from the same AMI, request Spot Instances to take adv
management role to the instance, and more.

Number of instances	1
Purchasing option	☐ Request Spot Instances
Network	Launch into EC2-Classic — Create new VPC
Availability Zone	us-east-1a
IAM role	None
Shutdown behavior	Stop
Enable termination protection	☐ Protect against accidental termination
Monitoring	☐ Enable CloudWatch detailed monitoring Additional charges apply.

Amazon EC2 Web Console



Monitoring activity



Amazon EC2 API

Key Pairs

- CreateKeyPair (p. 77)
- DeleteKeyPair (p. 141)
- DescribeKeyPairs (p. 238)
- ImportKeyPair (p. 379)

Elastic IP Addresses

- AllocateAddress (p. 13)
- AssociateAddress (p. 19)
- DescribeAddresses (p. 180)
- DisassociateAddress (p. 363)
- ReleaseAddress (p. 418)

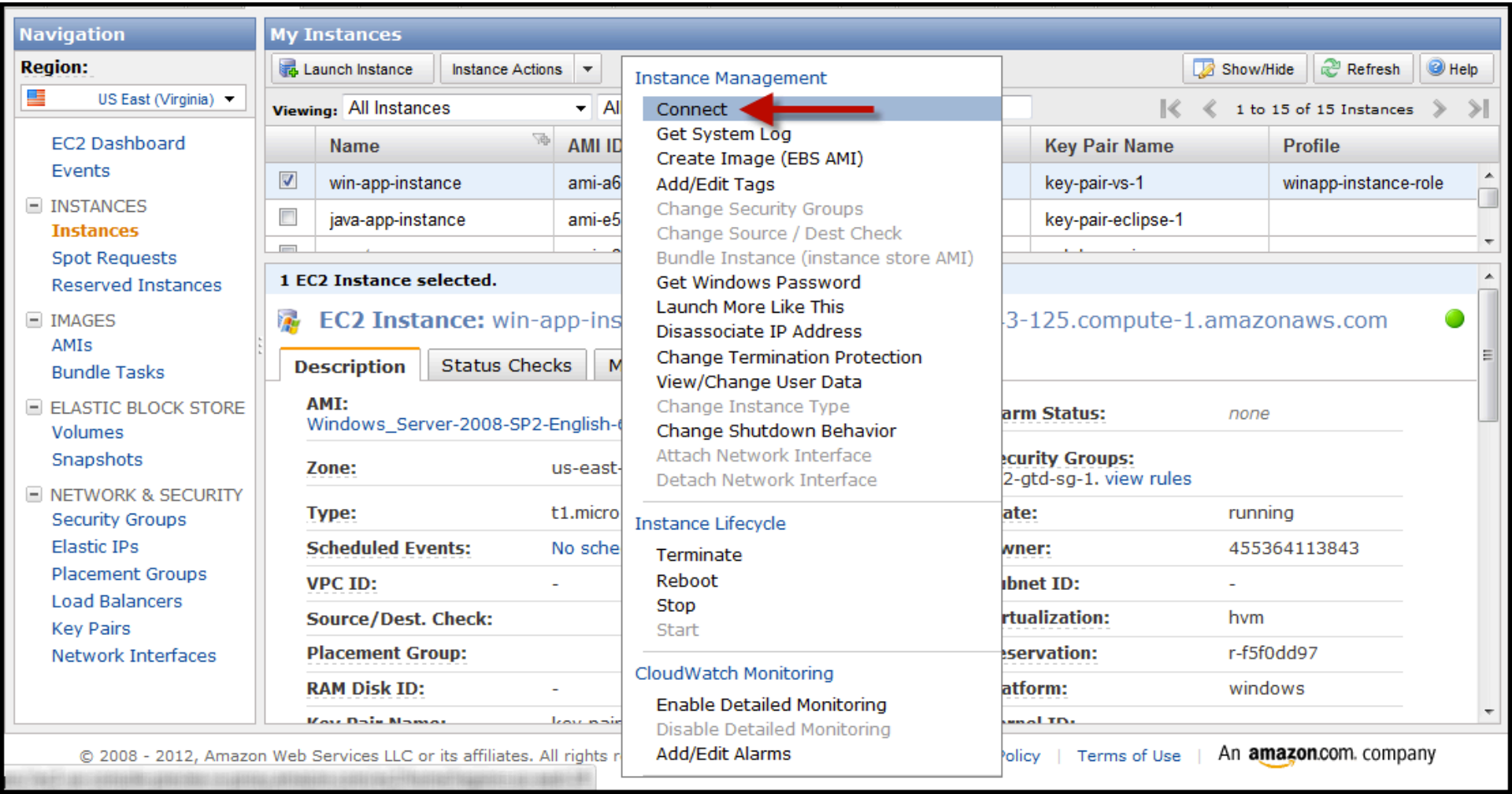
Spot Instances

- CancelSpotInstanceRequests (p. 54)
- CreateSpotDatafeedSubscription (p. 112)
- DeleteSpotDatafeedSubscription (p. 158)
- DescribeSpotDatafeedSubscription (p. 299)
- DescribeSpotInstanceRequests (p. 301)
- DescribeSpotPriceHistory (p. 309)
- RequestSpotInstances (p. 433)

Elastic Block Store

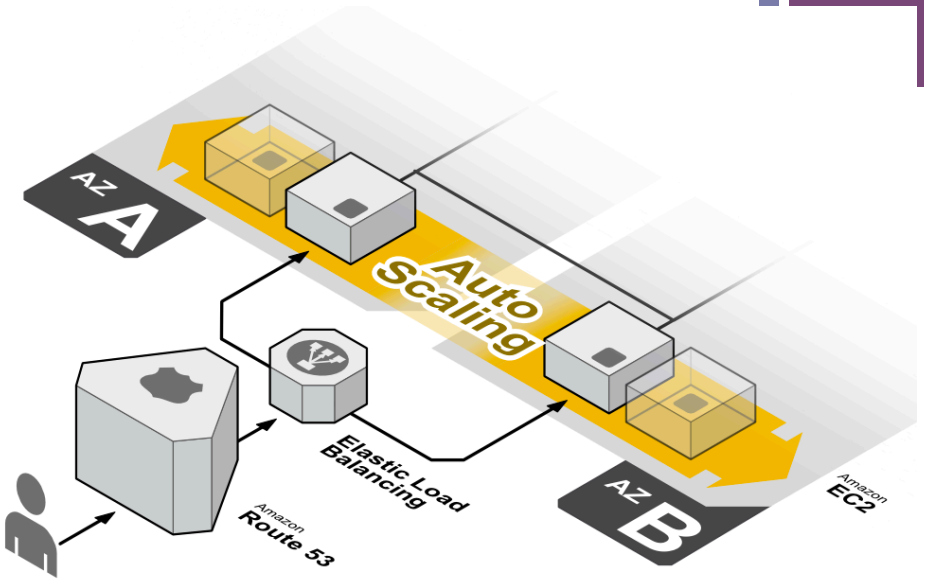
- AttachVolume (p. 30)
- CopySnapshot (p. 61)
- CreateSnapshot (p. 109)
- CreateVolume (p. 119)
- DeleteSnapshot (p. 156)
- DeleteVolume (p. 164)
- DescribeSnapshotAttribute (p. 291)
- DescribeSnapshots (p. 294)

Connecting to Amazon EC2 instance



Auto-Scaling

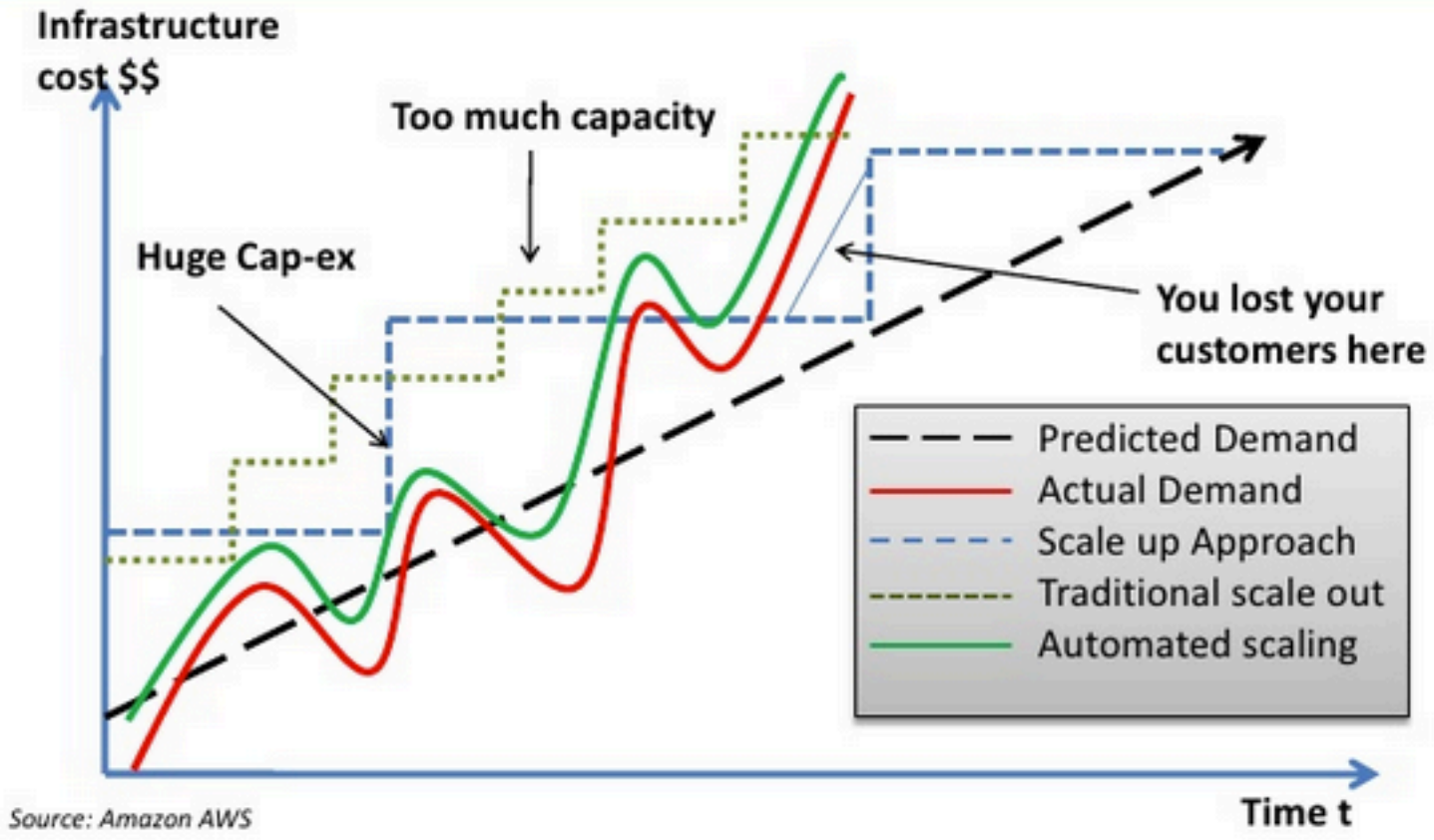
- Automatically adapt computing capacity to site traffic
- Schedule based (e.g. time of the day), rule-based (e.g. CPU utilization thresholds) automated scaling



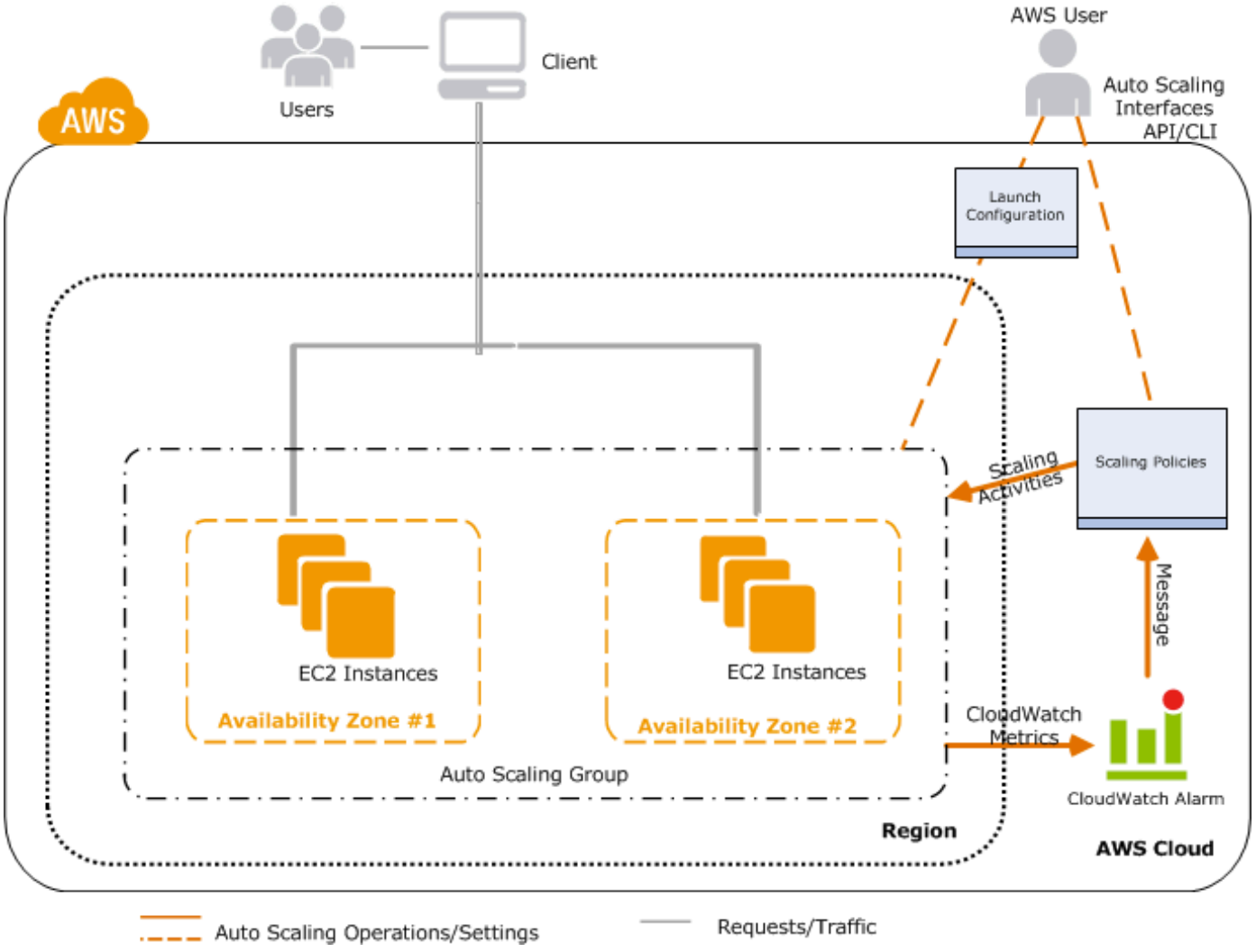

```
home$as-create-or-update-trigger app-trigger –auto-scaling-group webapp
--namespace "AWS/EC2" –measure CPUUtilization --statistic Average
--lower-threshold 40 --upper-threshold 70 --lower-breach-increment=-1
--upper-breach-increment=1 --breach-duration 120
```

Use command line tools to automate the process. For e.g. if CPU utilization goes above 70% for 120 secs, launch 1 machine. If goes below 40% for 120 secs, remove one machine

Understanding Scaling



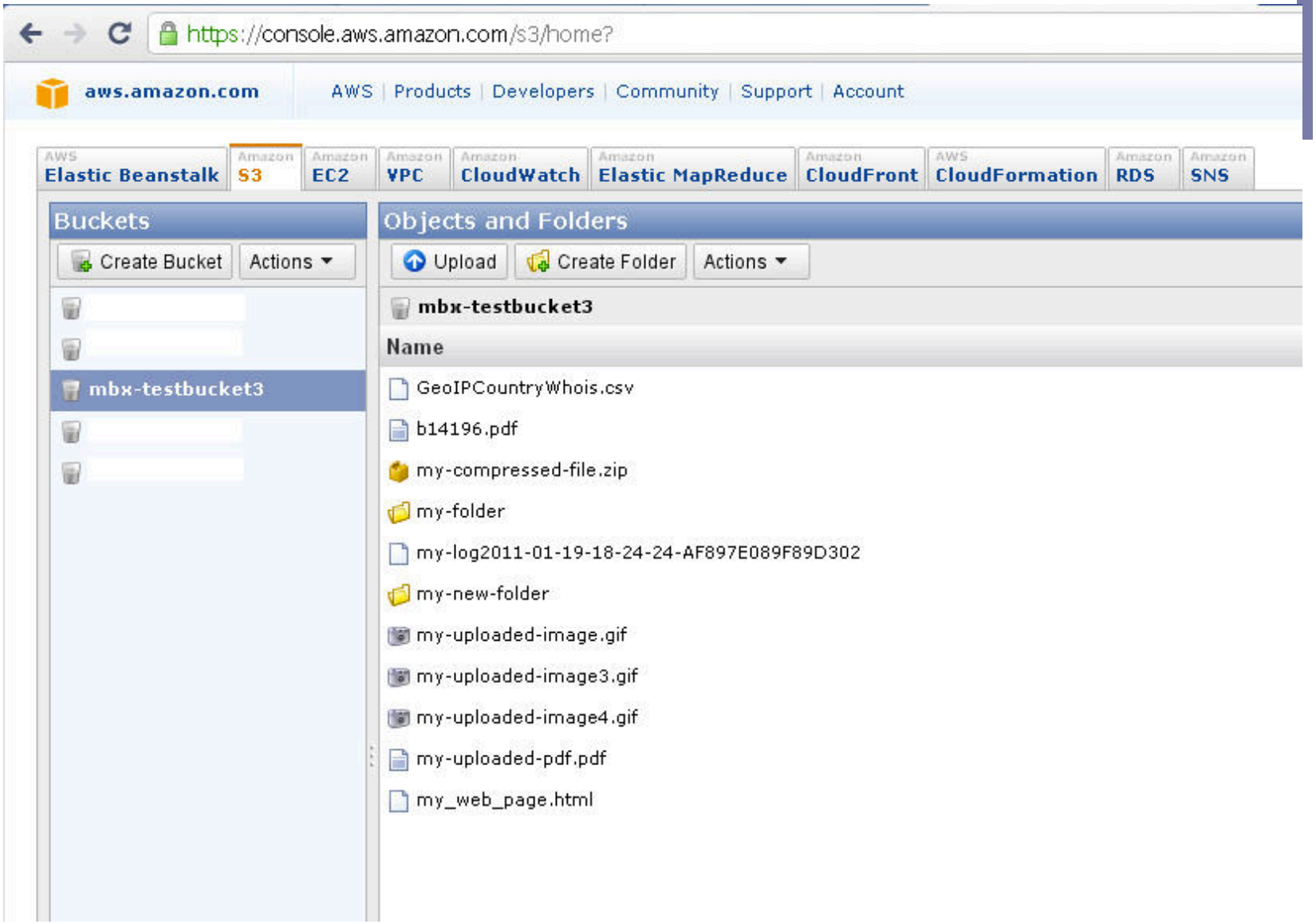
Overview of dynamic Scaling



Amazon S3 : Scaling Storage

- S3 = Simple Storage Service
- Storage in EC2 is destroyed once the instance is terminated. Amazon uses this infrastructure for its own websites
- Data organized in the form of buckets. Accessed as **`bucketname`**`.s3.amazonaws.com`
- Allows unlimited storage: in the increments of 1GB to 5GB
- Objects are stored and retrieved using a developer-assigned key. Can be used along with Amazon EC2 compute instances
- Objects can be made available to public by http or bittorrent protocol

S3 Web Console



Pricing for Amazon S3

Storage Pricing

Region: US Standard

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

If you wanted to back up data from your computer, at 0.01$/GB it will cost you around 5$/month for 500GB.

Data Transfer Pricing

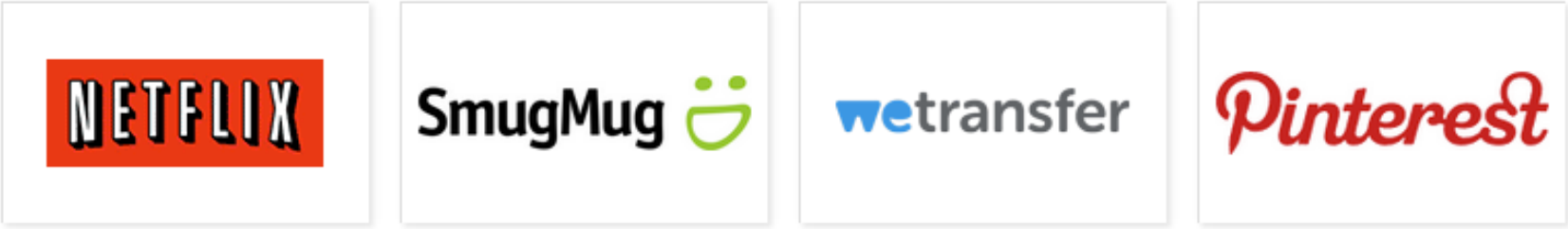
The pricing below is based on data transferred "in" to and "out" of Amazon S3.

Region: US Standard

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

Advantages of Using S3

- Scalability: The amount of storage and bandwidth you need can scale as you like
- Availability, speed, throughput, capacity and robustness is not affected even if you gain 10k users overnight.
- Leaves out lot of system administration overhead
- Seamlessly integrates with other Amazon AWS tools. Could also use it for backing up your data

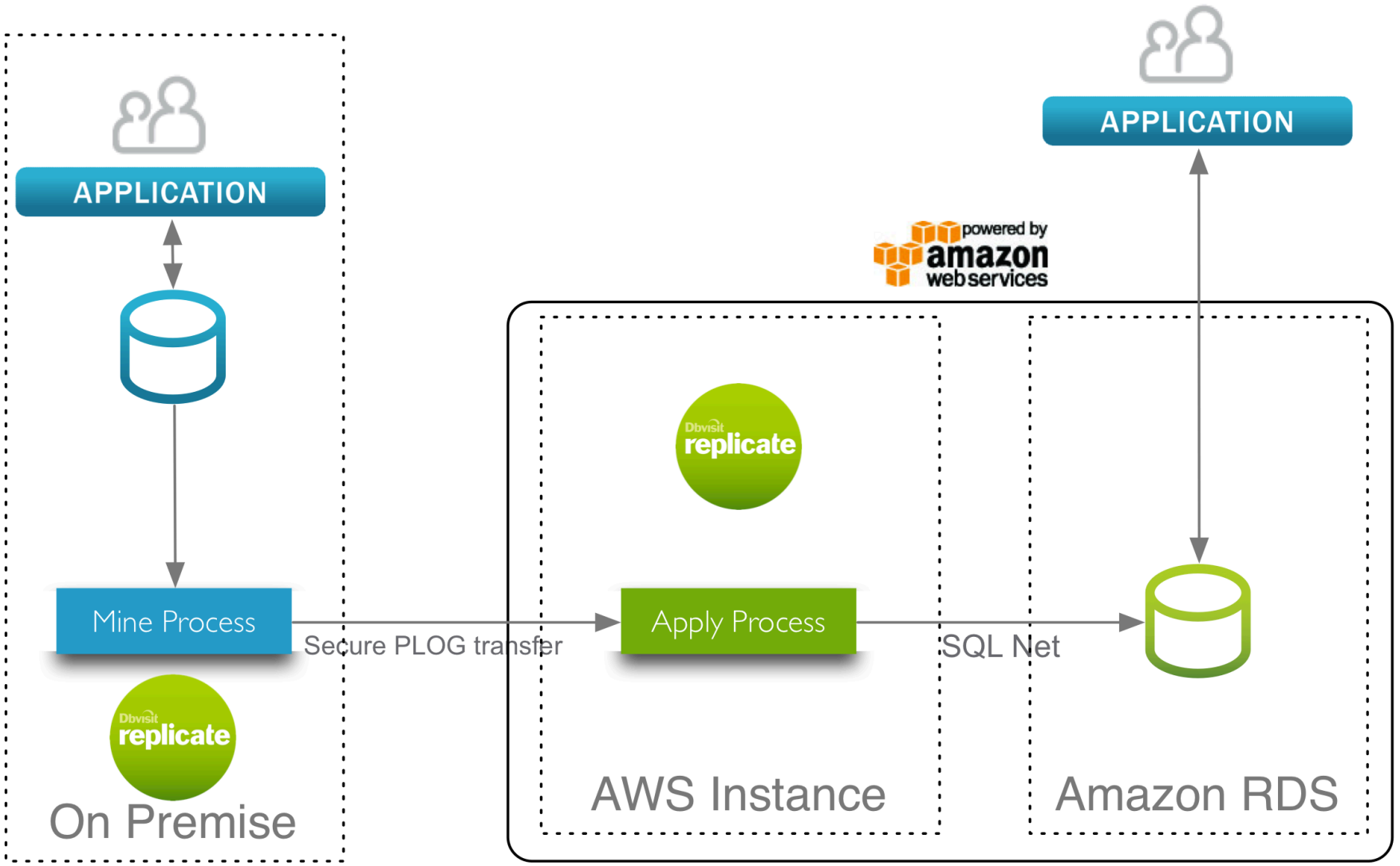


Many well known website use this for their storage requirements

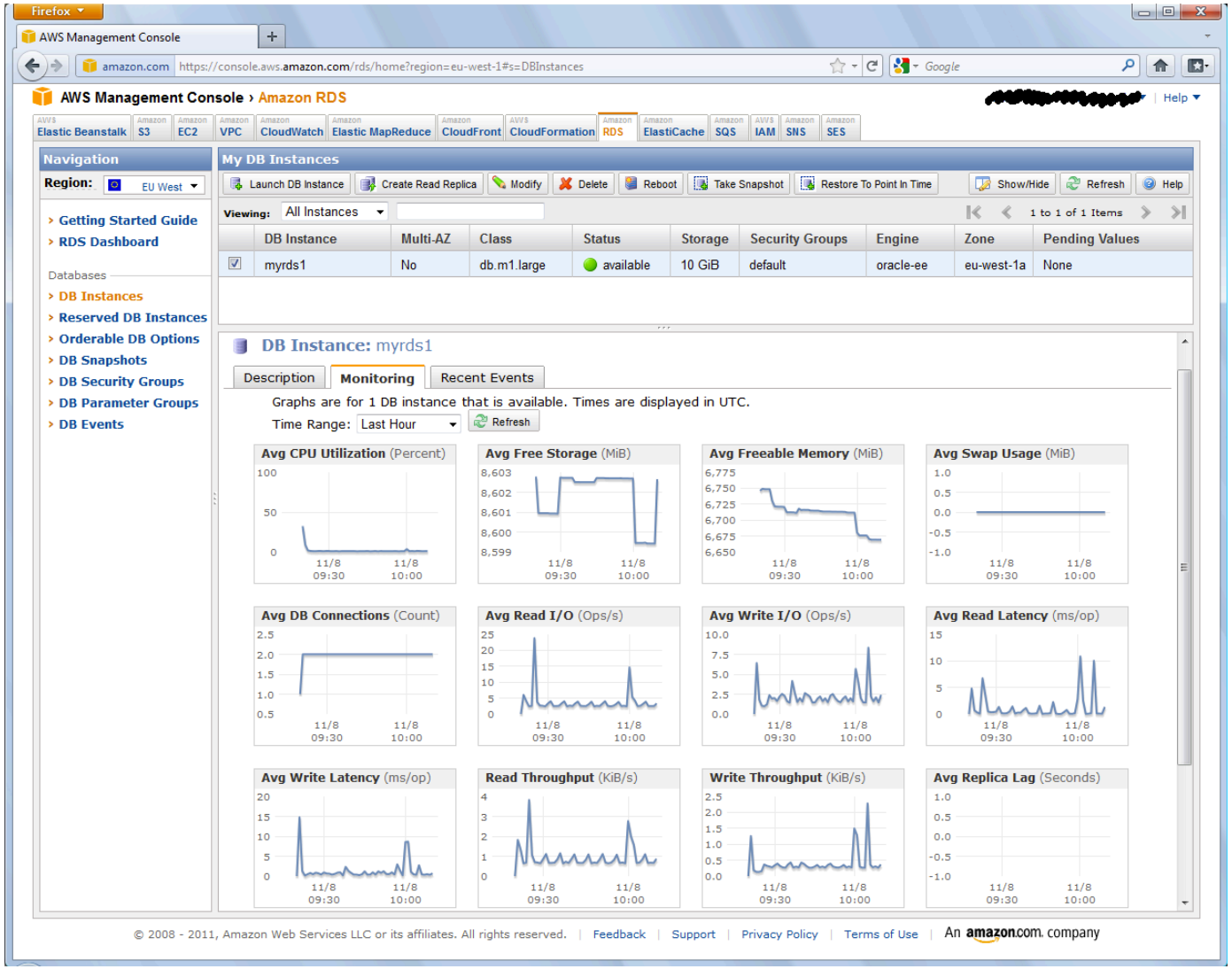
Amazon RDS: Scaling Databases

- Distributed relational database service
- Complex admin processes like patching software, backing up databases are managed automatically
- Can get started in minutes, instead of in days
- Scaling storage and compute resources can be performed by API calls
- Support for MySQL, Microsoft SQL server, Oracale Database and PostgreSQL

RDS Architecture



Amazon RDS Web Console



Other AWS Products

List of products [edit]

Compute [edit]

- Amazon Elastic Compute Cloud (EC2) provides scalable virtual private servers using Xen.
- Amazon Elastic MapReduce (EMR) allows businesses, researchers, data analysts, and developers to easily and cheaply process vast amounts of data. It uses a hosted Hadoop framework running on the web-scale infrastructure of EC2 and Amazon S3.

Networking [edit]

- Amazon Route 53 provides a highly available and scalable Domain Name System (DNS) web service.
- Amazon Virtual Private Cloud (VPC) creates a logically isolated set of Amazon EC2 instances which can be connected to an existing network using a VPN connection.
- AWS Direct Connect provides dedicated network connections into AWS data centers, providing faster and cheaper data throughput.

Content delivery [edit]

- Amazon CloudFront, a content delivery network (CDN) for distributing objects to so-called "edge locations" near the requester.

Storage and content delivery [edit]

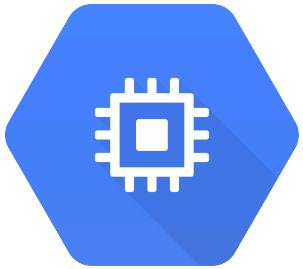
- Amazon Simple Storage Service (S3) provides Web Service based storage.
- Amazon Glacier provides a low-cost, long-term storage option (compared to S3). High redundancy and availability, but low-frequent access times. Ideal for archiving data.
- AWS Storage Gateway, an iSCSI block storage virtual appliance with cloud-based backup.
- Amazon Elastic Block Store (EBS) provides persistent block-level storage volumes for EC2.
- AWS Import/Export, accelerates moving large amounts of data into and out of AWS using portable storage devices for transport.

Database [edit]

- Amazon DynamoDB provides a scalable, low-latency NoSQL online Database Service backed by SSDs.
- Amazon ElastiCache provides in-memory caching for web applications. This is Amazon's implementation of Memcached and Redis.
- Amazon Relational Database Service (RDS) provides a scalable database server with MySQL, Informix,[20] Oracle, SQL Server, and PostgreSQL support.[21]
- Amazon Redshift provides petabyte-scale data warehousing with column-based storage and multi-node compute.
- Amazon SimpleDB allows developers to run queries on structured data. It operates in concert with EC2 and S3 to provide "the core functionality of a database".

Infrastructure as a Service

Red Hat



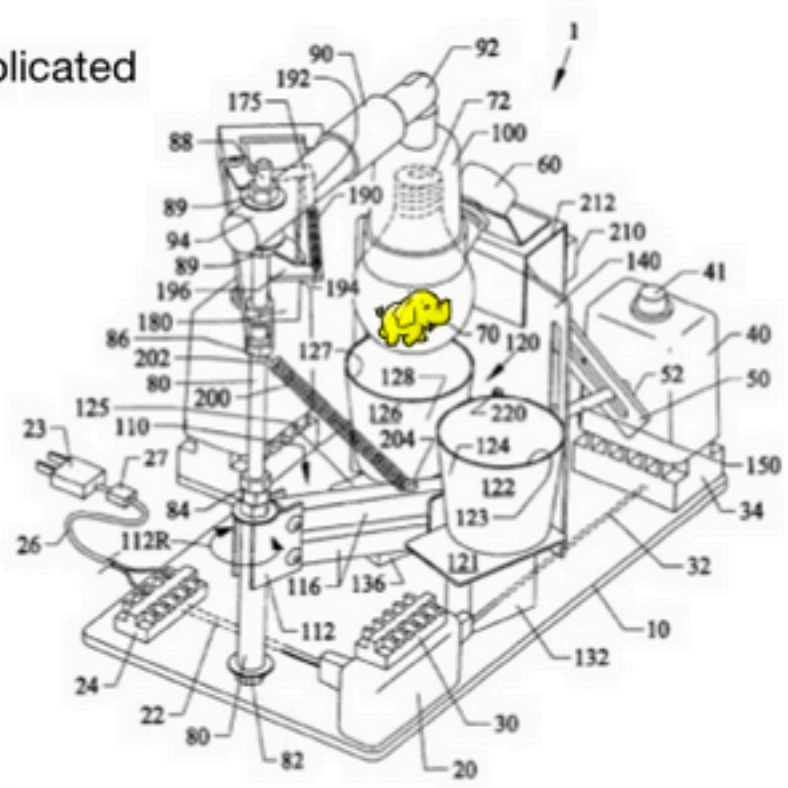
Google Compute Engine

Hadoop, MapReduce ….

Hadoop distributes data and computation across a large number of computers.

DISCLAIMER

- Don't use Hadoop if your data and computation fit on one machine
- Getting easier to use, but still complicated



http://www.wired.com/gadgetlab/2008/07/patent-crazines/

What exactly is hadoop ?

- Actually a growing collection of subprojects

hadoop Common

HBASE

HIVE

AVRO

hadoop MapReduce

hadoop HDFS

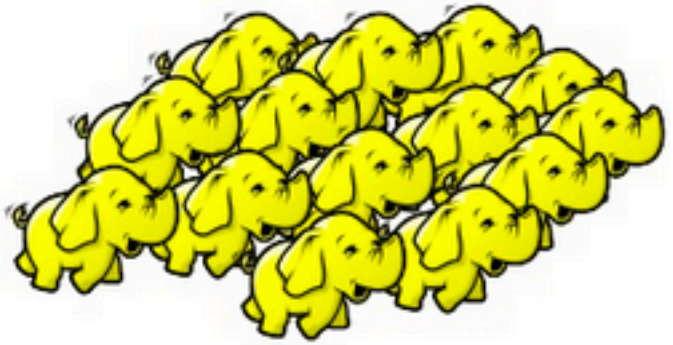
chukwa

An overview of Hadoop Map-Reduce

Traditional Computing

(one computer)

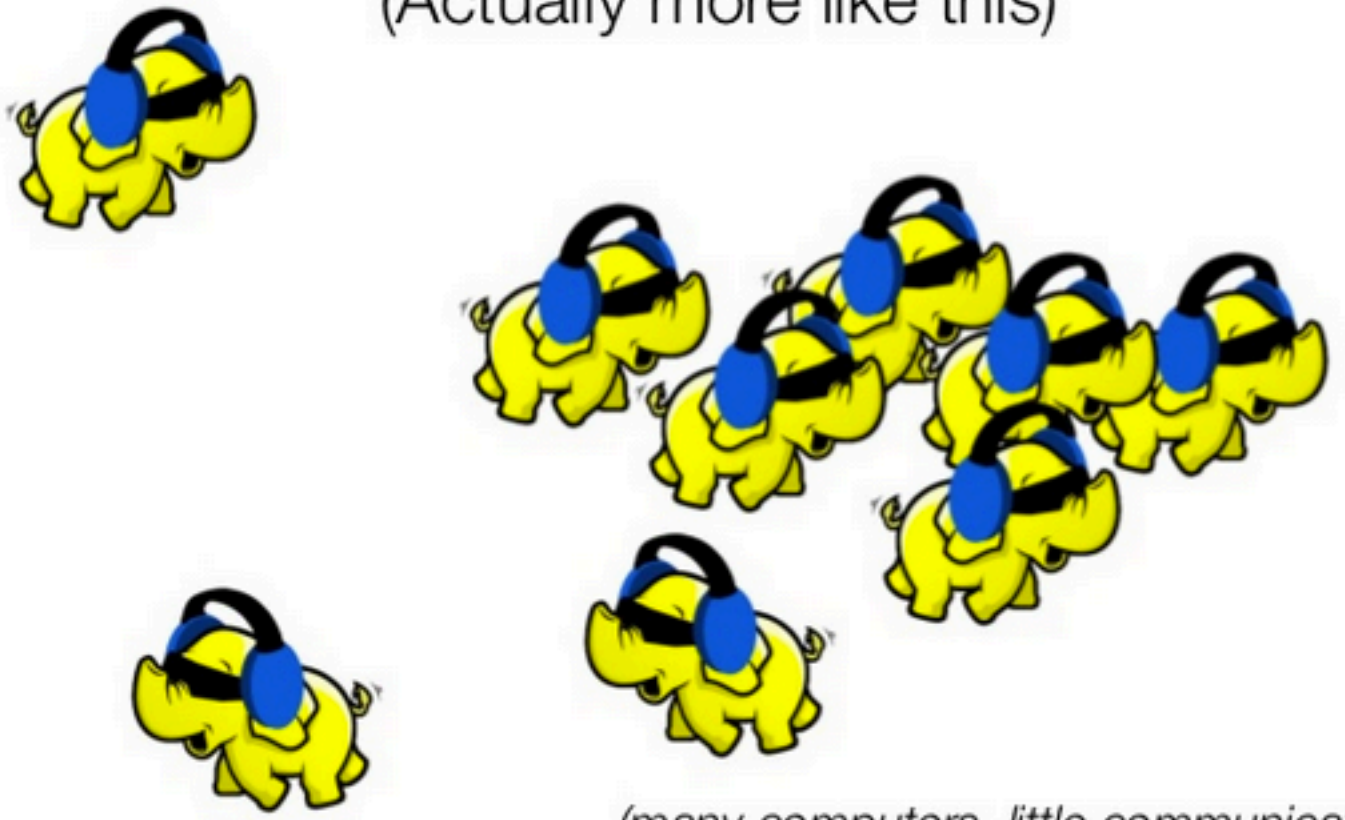
Hadoop



(many computers)

An overview of Hadoop Map-Reduce

(Actually more like this)



(many computers, little communication, stragglers and failures)

What is Hadoop?

- A large scale distributed batch processing infrastructure
- True power lies in its ability to scale to hundreds or thousands of machines
- Hadoop includes distributed file system which breaks in input data and sends fractions of the original data to several machines in your cluster
- It includes a distributed file system which breaks up input data and sends fractions of the original data to several machines in your cluster
- Similar to NFS but lot more efficient

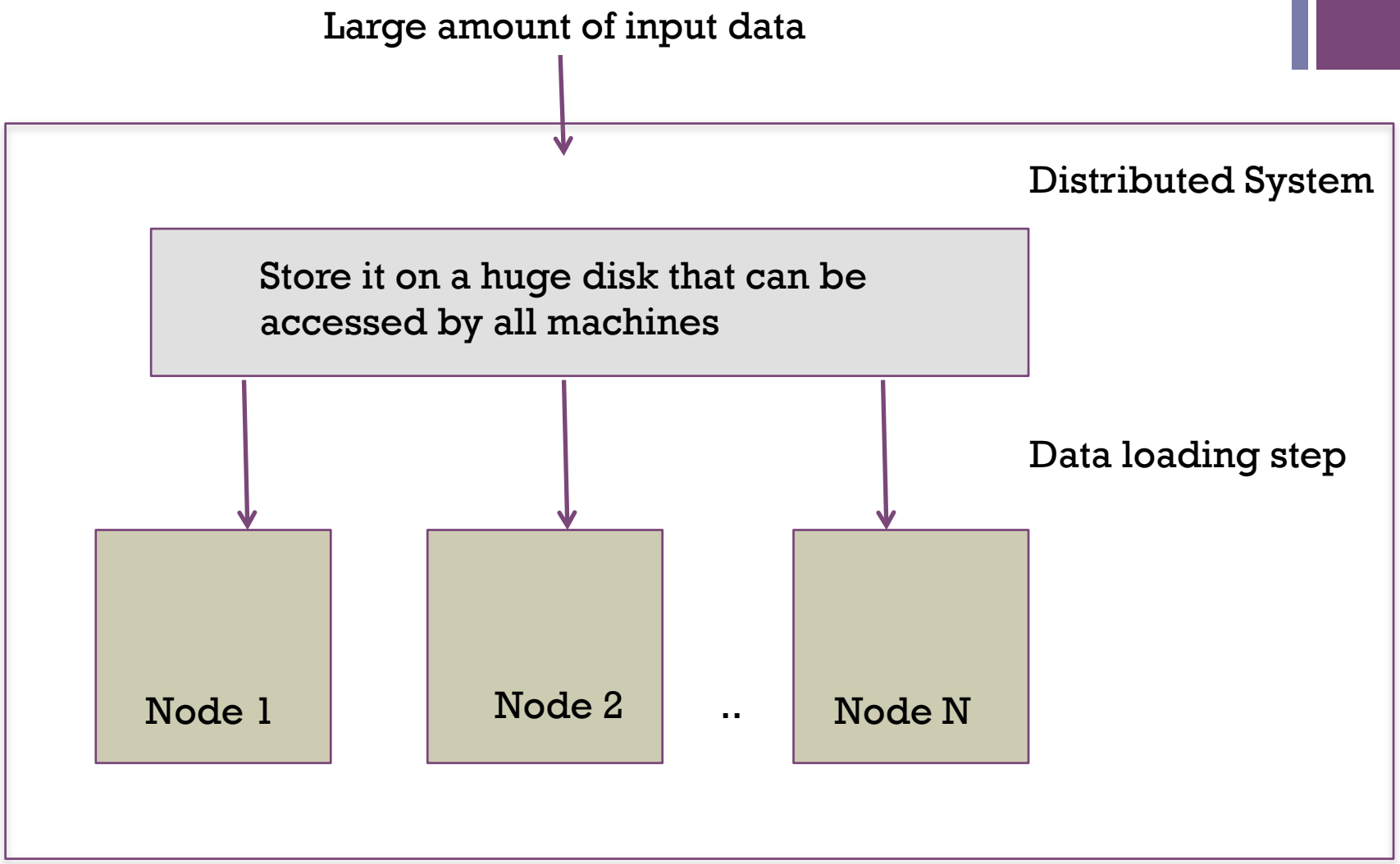
Challenges at Large Scale

- Data distributed over multiple machines
  - Increases probability of failure
  - Network failure, machine failure, router failure etc.
  - Drive failures, desynchronized clocks etc.
- Synchronization between multiple machines remains biggest challenge for distributed systems
- For e.g. in a system with 100 machines, if 1 fails it should be equivalent to loss of 1% of the work and not 100% of work

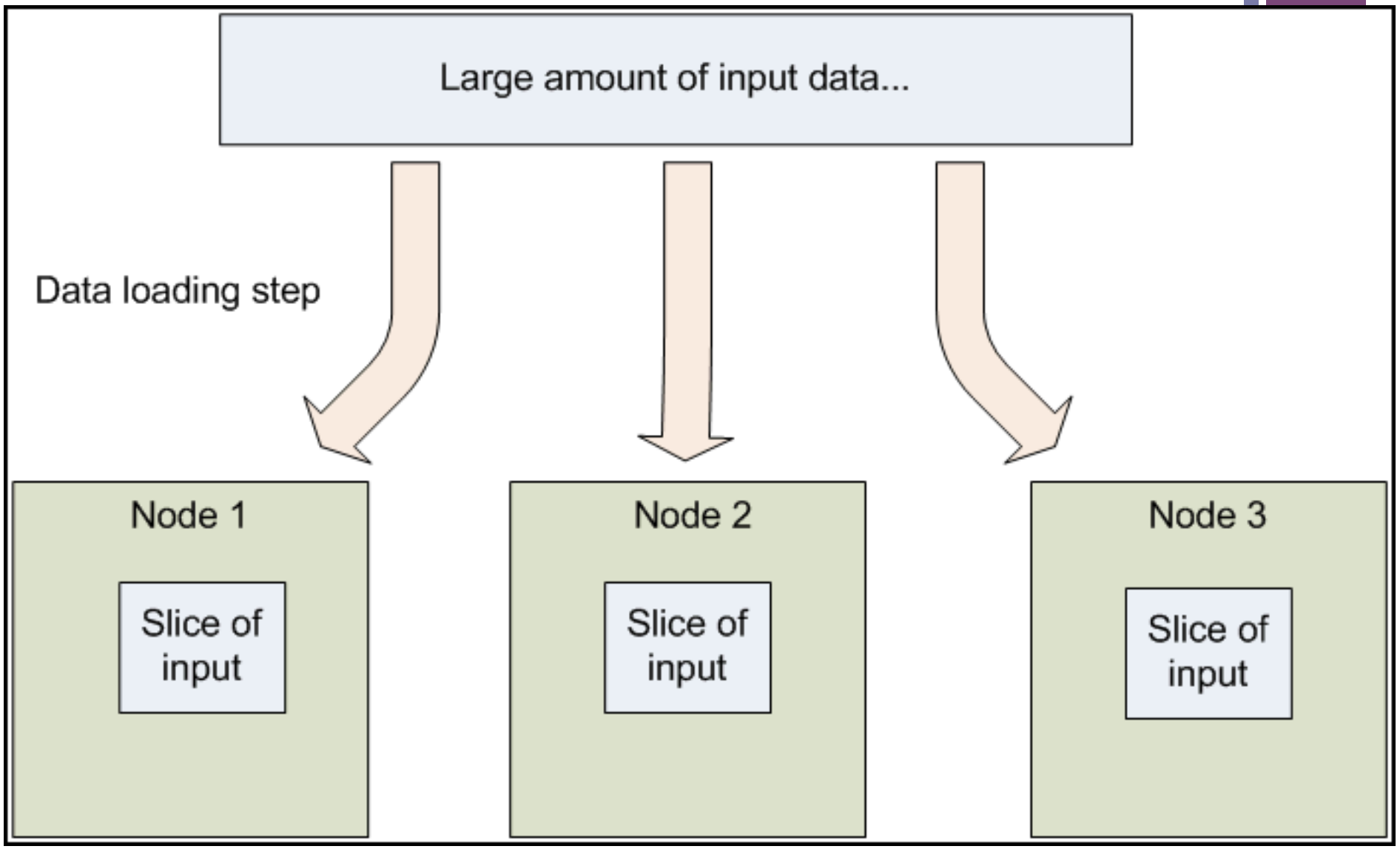
Hadoop Approach

- Connect multiple computers together and efficiently process large volumes of information by using commodity machines
- A theoretical 1000-core CPU costs more than 1000 single CPU machines of 250 quad-core machines.
- Hadoop will tie smaller and more reasonably priced machines together into a single cost-effective compute cluster
- Data is distributed to different nodes in the cluster instead of a single NFS drive

Network File System (NFS)Approach



Hadoop Data Approach



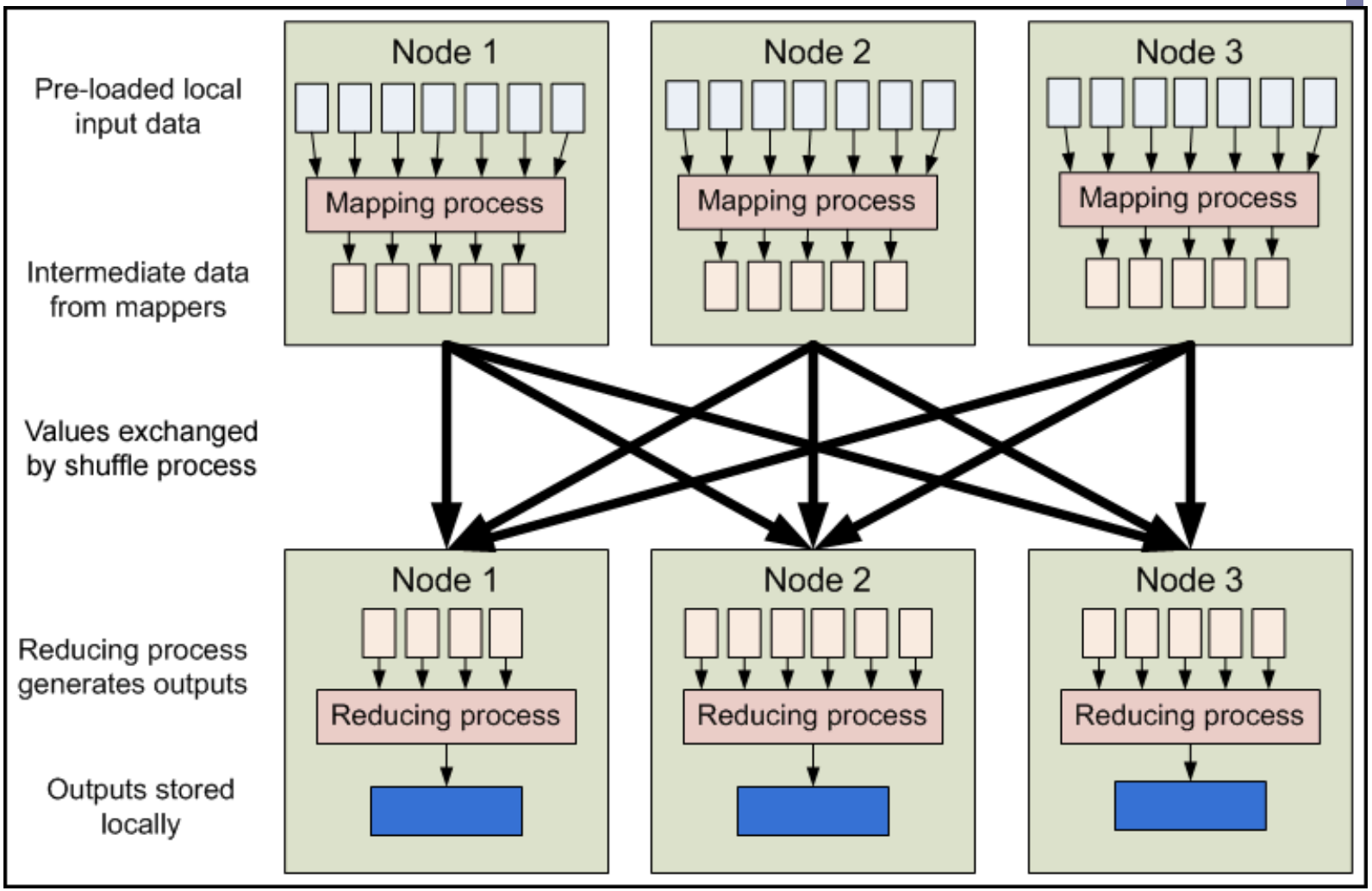
Data Management in Hadoop

- Data is conceptually record-oriented in Hadoop.
- Individual input files are broken into lines or into other formats specific to application logic
- Each node in the cluster processes subset of the record
- Data is stored on local disks: thus reduces overhead on network bandwidth and transfers
- **Moving computation to the data** instead of moving data to computation

MapReduce: Isolated Processes

- Hadoop limits the amount of communication which can be performed by the processes
- Though this looks like limitation, it increases reliability of the system.
- Hadoop will not run just any program and distribute it across cluster (just like GPUs)
- Programs must be written to conform "MapReduce" programming model
- Records are processed in isolation by tasks called *Mappers*
- The output of mappers is then brought together into second set of tasks called *Reducers,* where results from different mappers can be merged

MapReduce Model



MapReduce: Isolated Processes

- Communication between nodes is implicit
- Pieces of data are tagged with key names, which inform Hadoop how to send related bits of information to common destination node
- Hadoop internally manages all of the data transfer and cluster topology issues.
- Less message passing between nodes compared to MPI (Message Passing Interface).
- Offers flat scalability compared to MPI. In MPI there is significant overhead for large scale systems. Have to manually engineer how message passing between all the machines work.
- Same code works for MBs of data and TBs of data. No overhead for refactoring, I/O, node failure etc. Hadoop takes care of it.