Cloud Computing Jumpstart

Gautham Pai
jnaapti

Setting the Context

How much data are we talking about really?

- 200 million Tweets per day – as of Jun 2011 (10s of GB of Tweets per day)
- Wikipedia dump – 31GB uncompressed
- Common Crawl data – 10s of TBs
- Tumblr – adding 3TB of new data everyday

How much data are we talking about really?

- Google processes 25PB of data per day (as of 2011)
- Facebook – 4.6 million messages get sent every 20 minutes (as of Dec 2010)
- Youtube – 100+ hours of videos uploaded every minute (as of May 2013)

We are dealing with a lot more data...

- Increase in the number of sensor devices
- Larger audience of users using our applications via the web and social networks results in increased data generation
- Cost of storage is falling – so we never discard any of the data

Scrabulous – Case Study

- Scrabulous case study
  - Built by 2 young chaps from Kolkata
  - Both were in their early 20's when they built it
  - One was still in college.
  - 500,000 users daily – back in 2008, 25,000$ in ad-revenues per month

Source: Wikipedia

Imagine this scenario

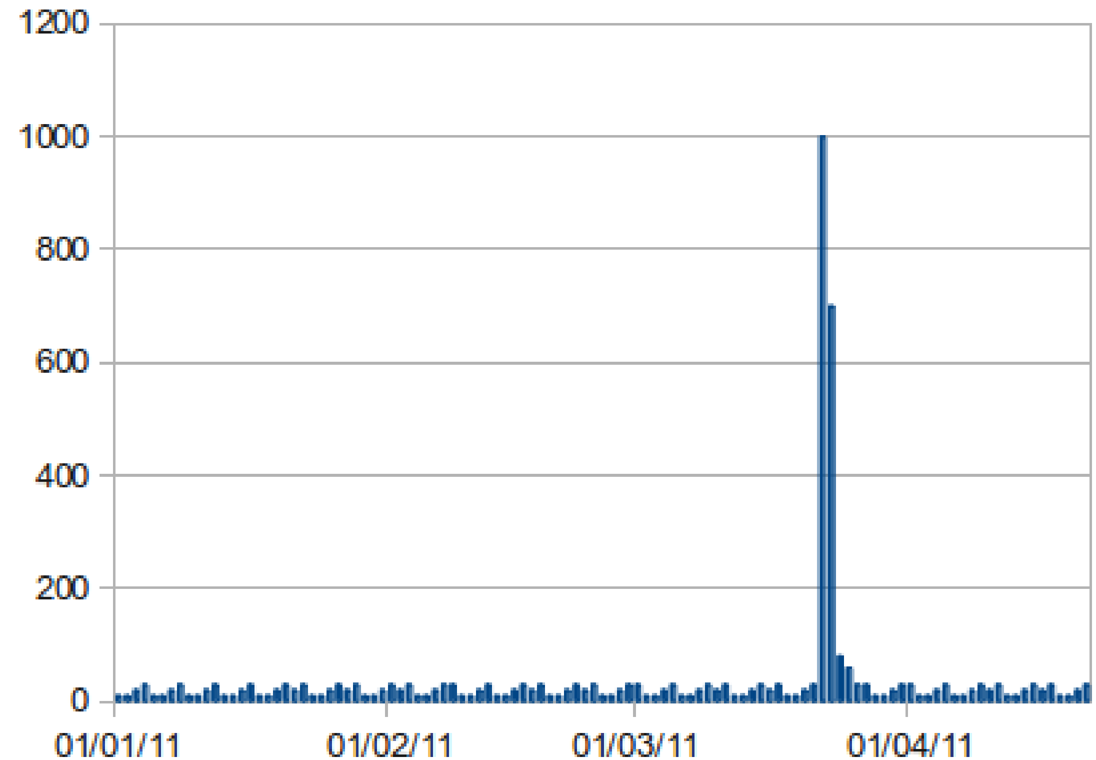
We Have All It Takes

- We have access to a lot of the tools (software) that big corporations use for free
- We have access to a lot of the data for free
- We have computing power (hardware) available cheaply

This last point is because of the “Cloud Computing” revolution

Questions to ask

- Ok, you have the resources
- You build a cool web application
- It is an overnight hit - can you handle it?
- What happens if the server has a disk crash?
- Can we prevent website outages in the account of hardware failures?



Slashdot Effect

Looking for answers

What do technology companies like Google/Facebook/Twitter use to manage data? What challenges do they face in managing such huge volumes of data? How do they analyze such data?

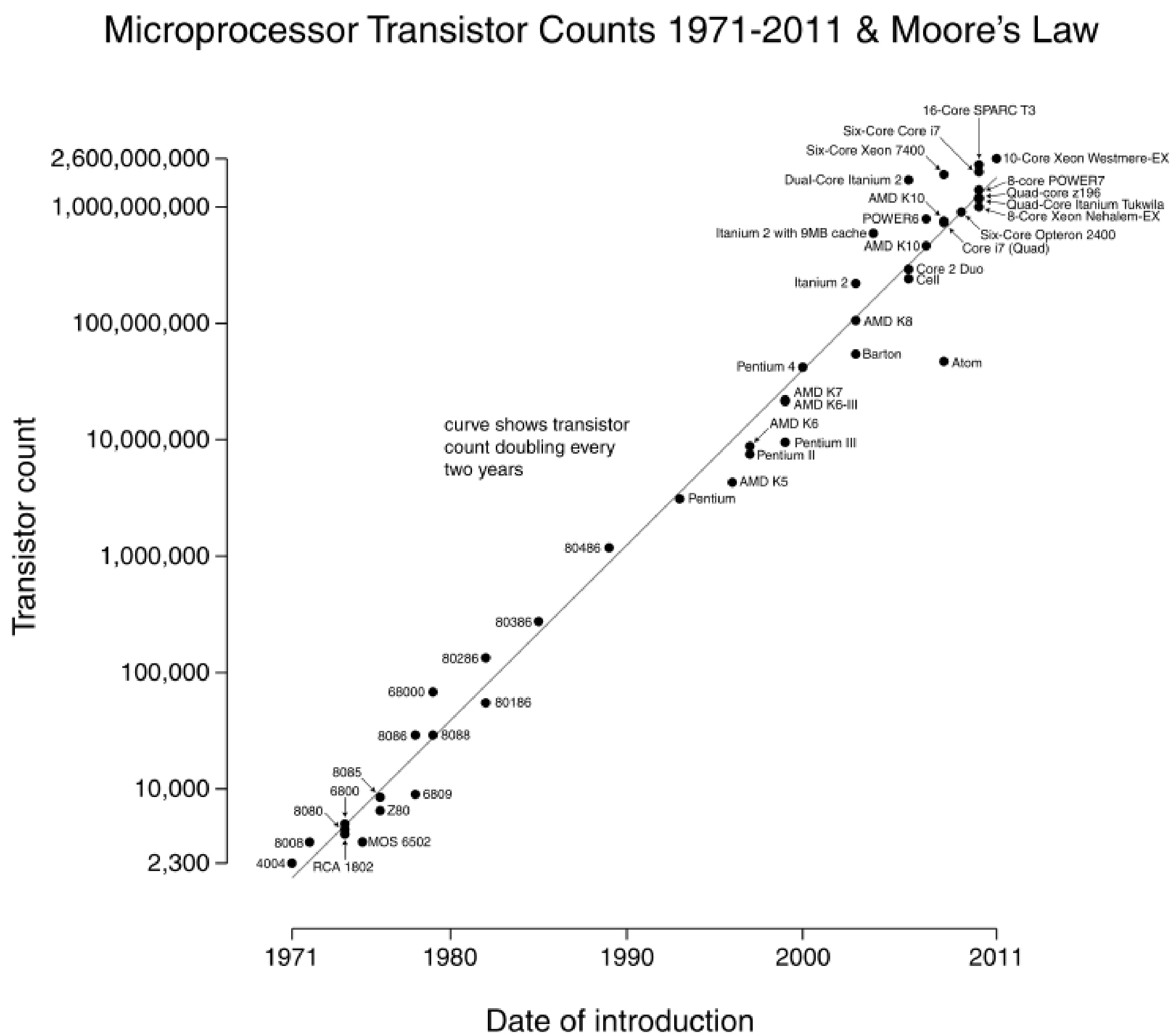
Image Source: http://opencompute.org/

Making Systems Speak

Processing Power over the Years

- Smartphones of today are more capable than desktops that you could buy a decade ago
- In 2002, you could buy a desktop with a single core of 400MHz.
- Today, we can get a dual-core 1GHz mobile phone for the same price

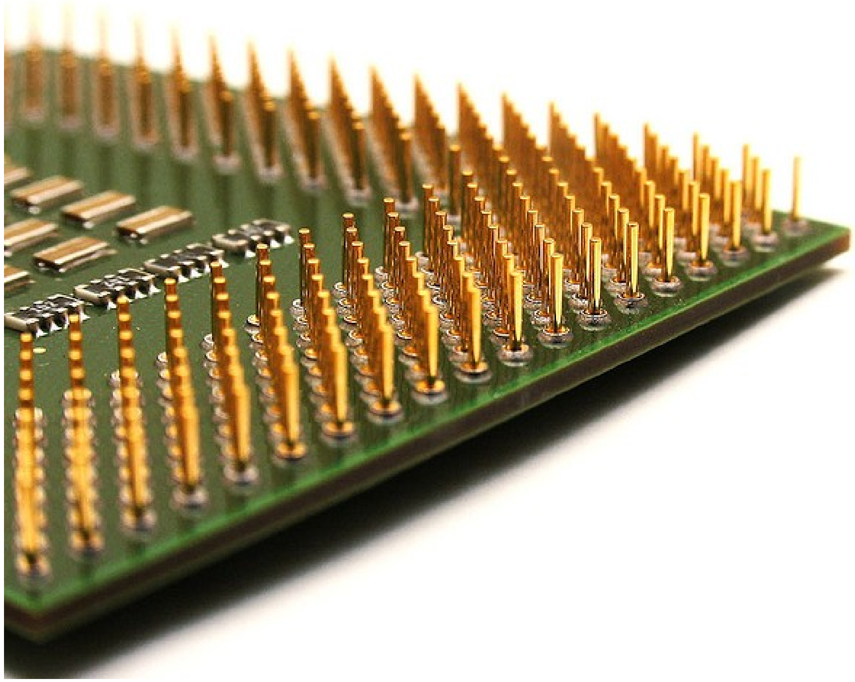
Moore's Law



Source: Wikipedia

Multicore

- We are reaching the limits of silicon
- Manufacturers are now packing more cores on a single chip
- Does a “kilo-core” processor sound weird?



Source: http://www.flickr.com/photos/negativz/41549347/

Issue with Multicore

- Let's say this program takes 'x' seconds to run on a 1GHz processor
- How long does it take to run on:
  - A 2GHz processor?
  - A dual-core processor with 1GHz per core?

```
sum = 0
for i in range(1000000):
    sum += i
print sum
```

Solution?

- Divide the work into smaller chunks that can be distributed across these cores

How about dividing the work across thousands of systems? Wouldn't that be even faster?!

Distributed System

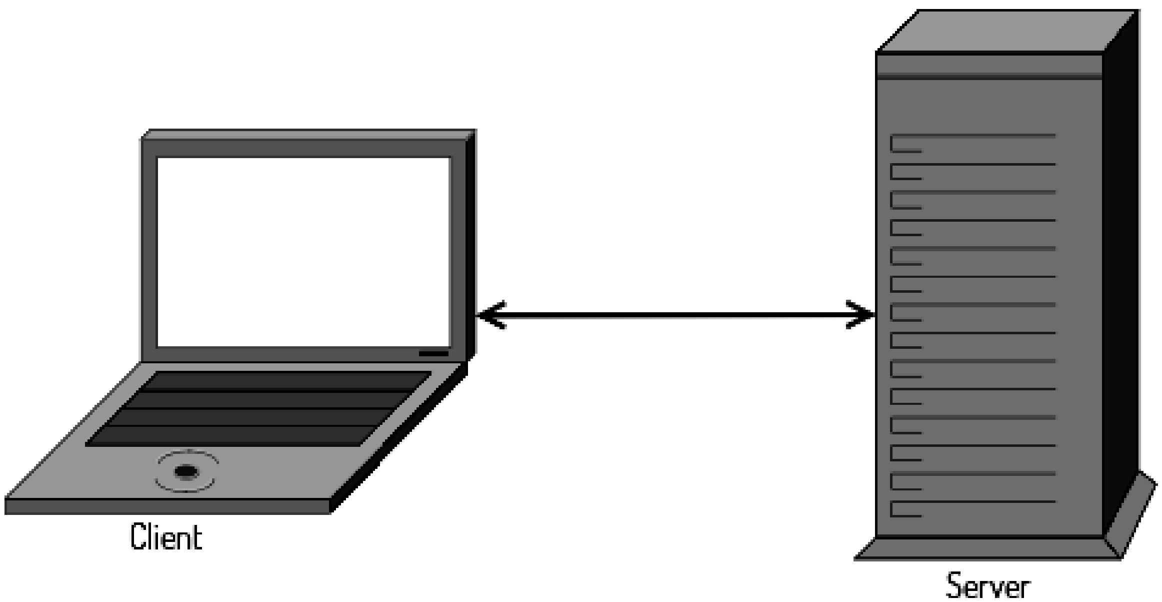
- A collection of computers which work together to achieve some task.
- The computers interact over a network.
- Google/Facebook/Twitter etc have tens of thousands of such systems working together on tasks such as crawling the web, coming up with friend recommendations, indexing your tweets etc.

Working of a Website

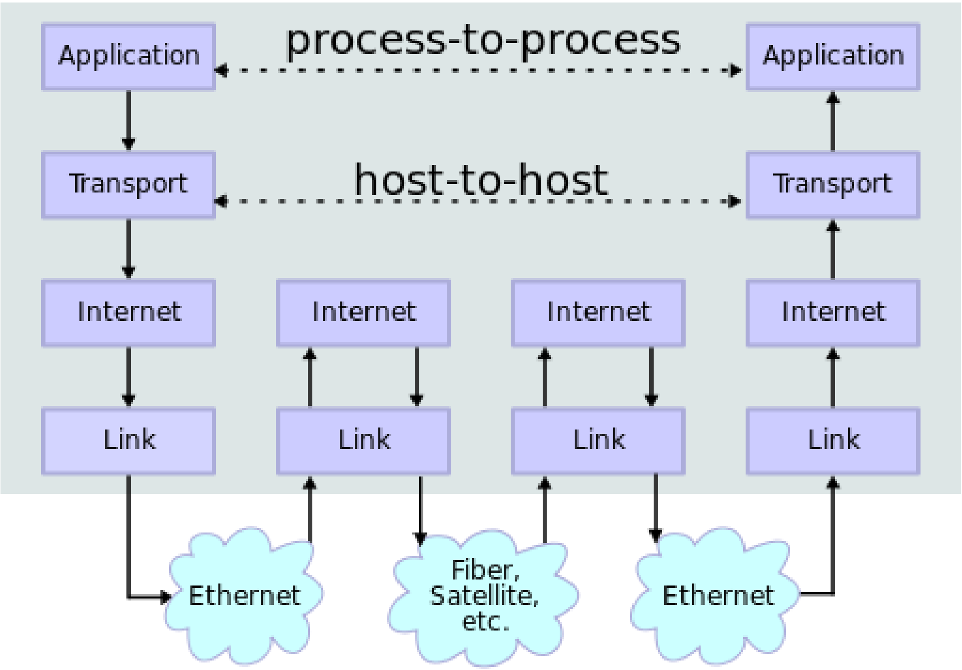
- You enter a URL in the address bar of your browser
- You press ENTER
- You see a webpage

What happens behind the scenes?

Client/Server Model



TCP/IP

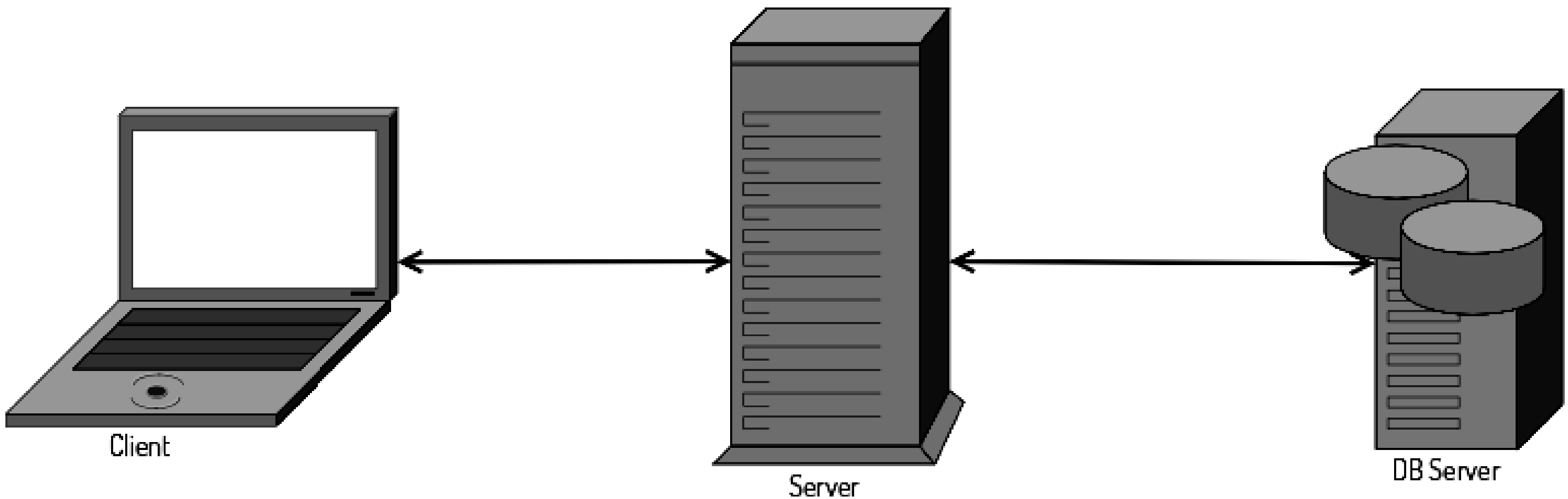


Source: Wikipedia

TCP/IP – Low Level Details

- Tooling
  - DNS – dig
  - Finding your system's IP – ifconfig
  - Routing – traceroute
  - Finding socket information – netstat
  - You can use a tool called Wireshark to get packet information

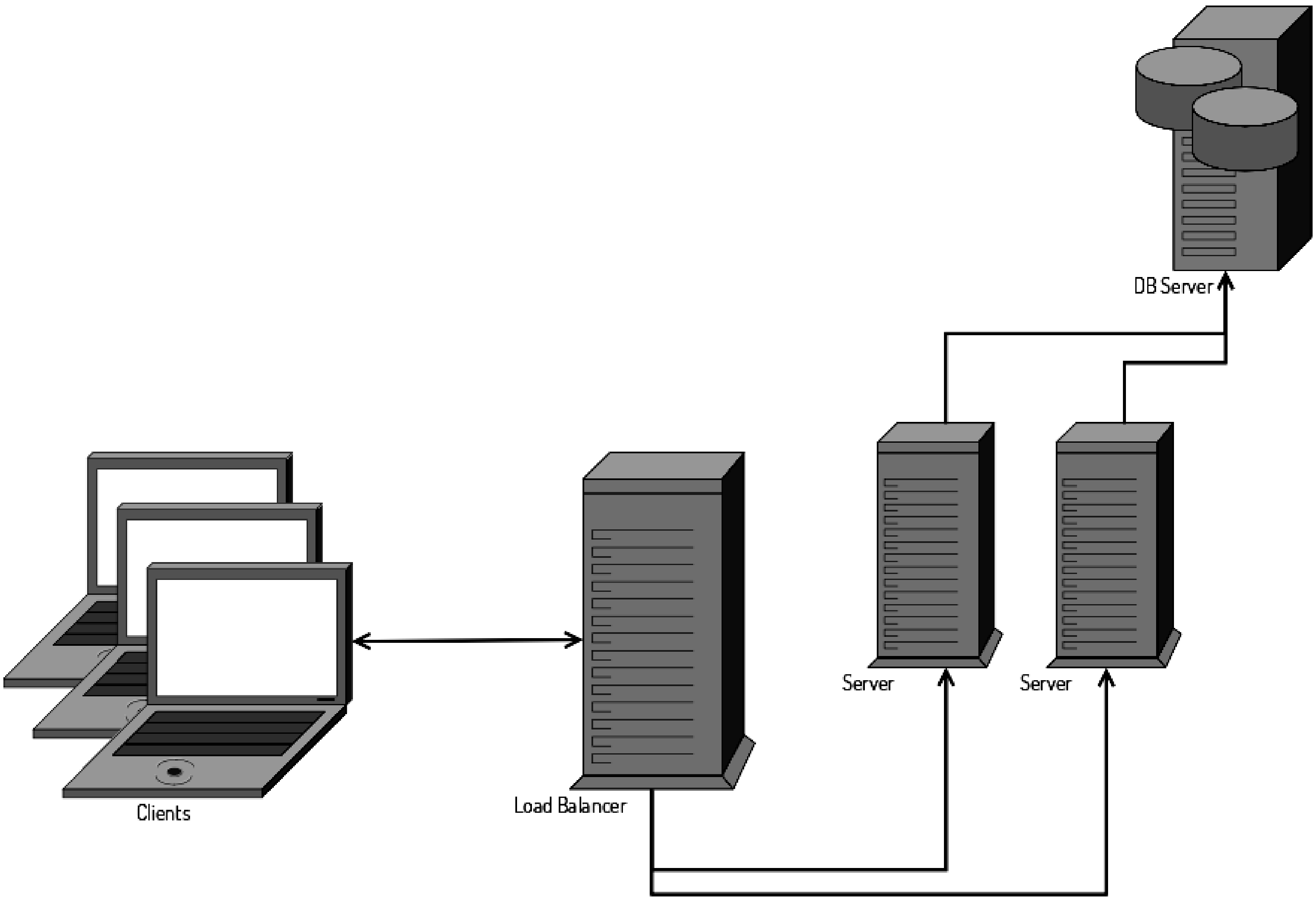
Client/Server Model – Separate DB Layer



Problem 1 – Too Many Requests

- What if a thousand users access my server at the same time?
- If the server can handle 200 such requests in parallel in one second, what if I have 400 requests per second?
  - 1st second → 200 requests
  - 2nd second → 600 requests (200 are from the previous second)
- Results in server thrashing
- Solution: Load Balanced Setup

Load Balancing



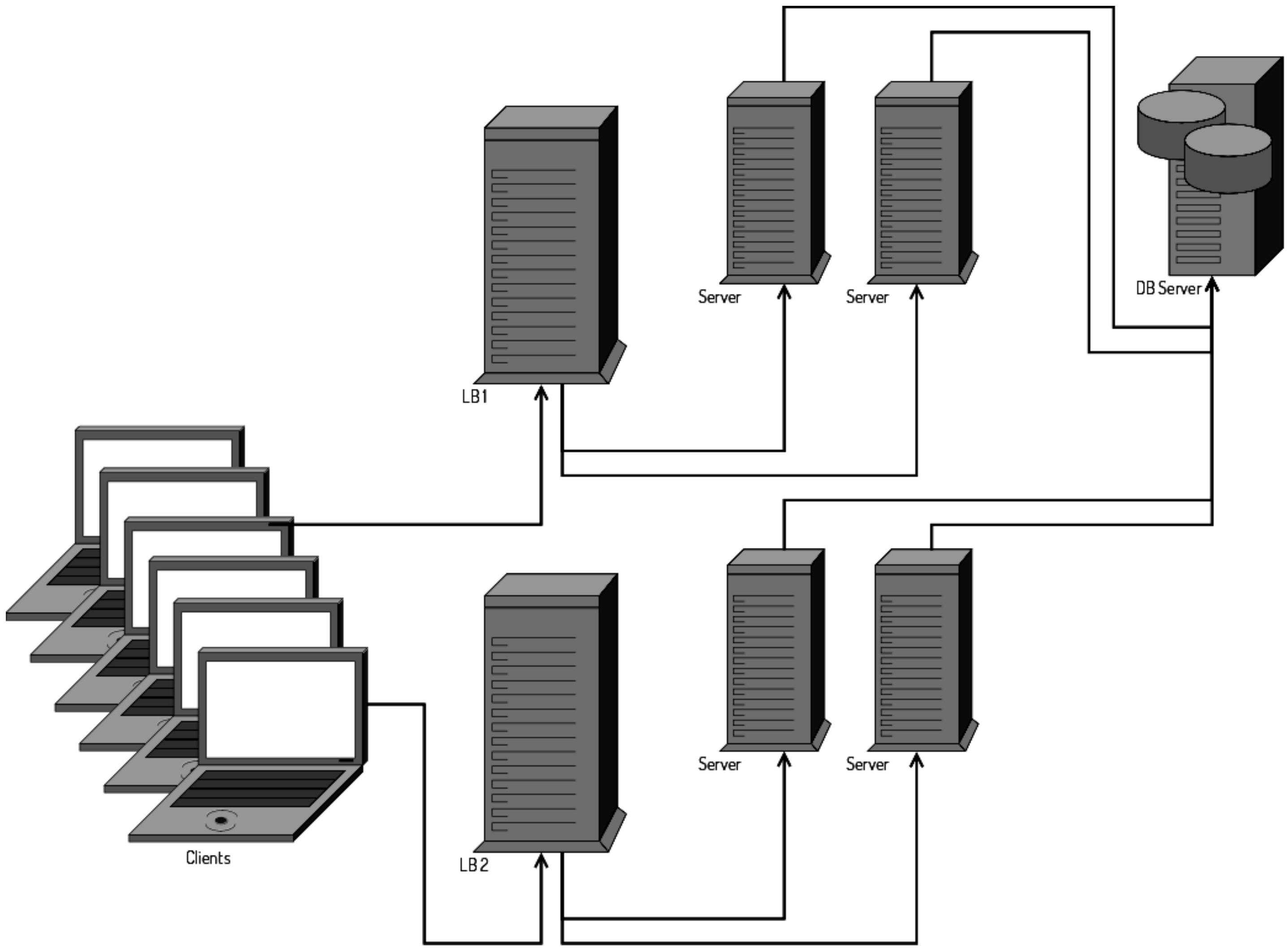
Load Balancing

- Load balancing is a way of parallelizing processing across multiple machines
- The load balancer acts as a proxy that streams requests and responses between the client and the processing server.
- Eg: HAProxy
- Stateful and Stateless Architectures

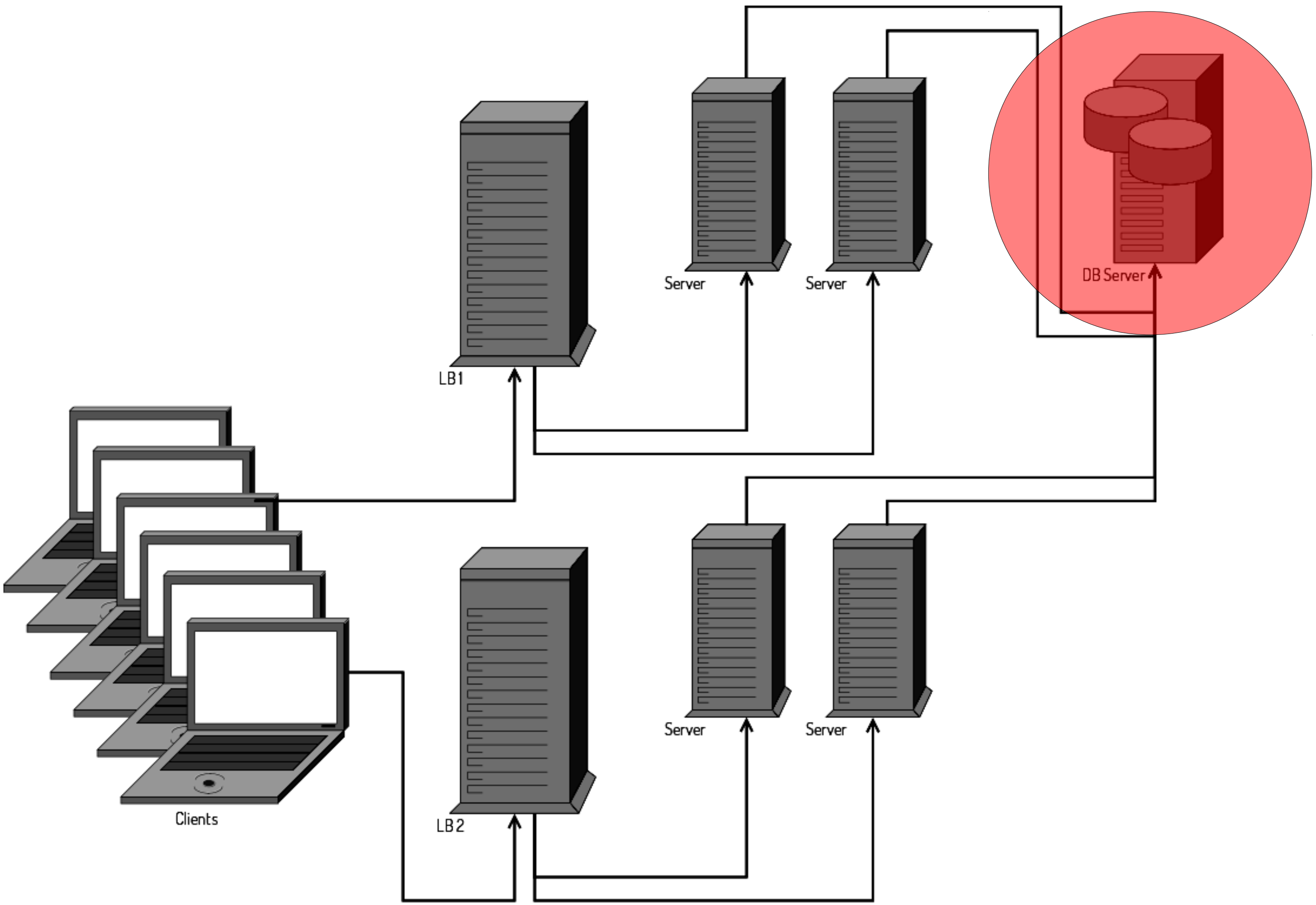
Problem 2 – Even More Requests

- What if the Load Balancer itself becomes the bottleneck?
- Solution:
    - Round Robin DNS
    - Building multiple independent clusters

Clustered Setup



Problem 3



Virtualization

Virtualization

- A way to use something “virtually”
- Eg: Running one operating system within another
  - The OS that runs the other OS is called the Host
  - The OS that runs within the other OS is the Guest

What if we have a way to manage such virtual instances via commands and API's?

Virtualization Types

- Hardware Virtualization
- Application Virtualization
- Operating System Level Virtualization
- Others
  - Memory
  - Network
  - Storage

Virtualization Techniques

- Virtual Machines (eg: KVM)
- Paravirtualization (eg: Xen)
- Containers (eg: LXC)

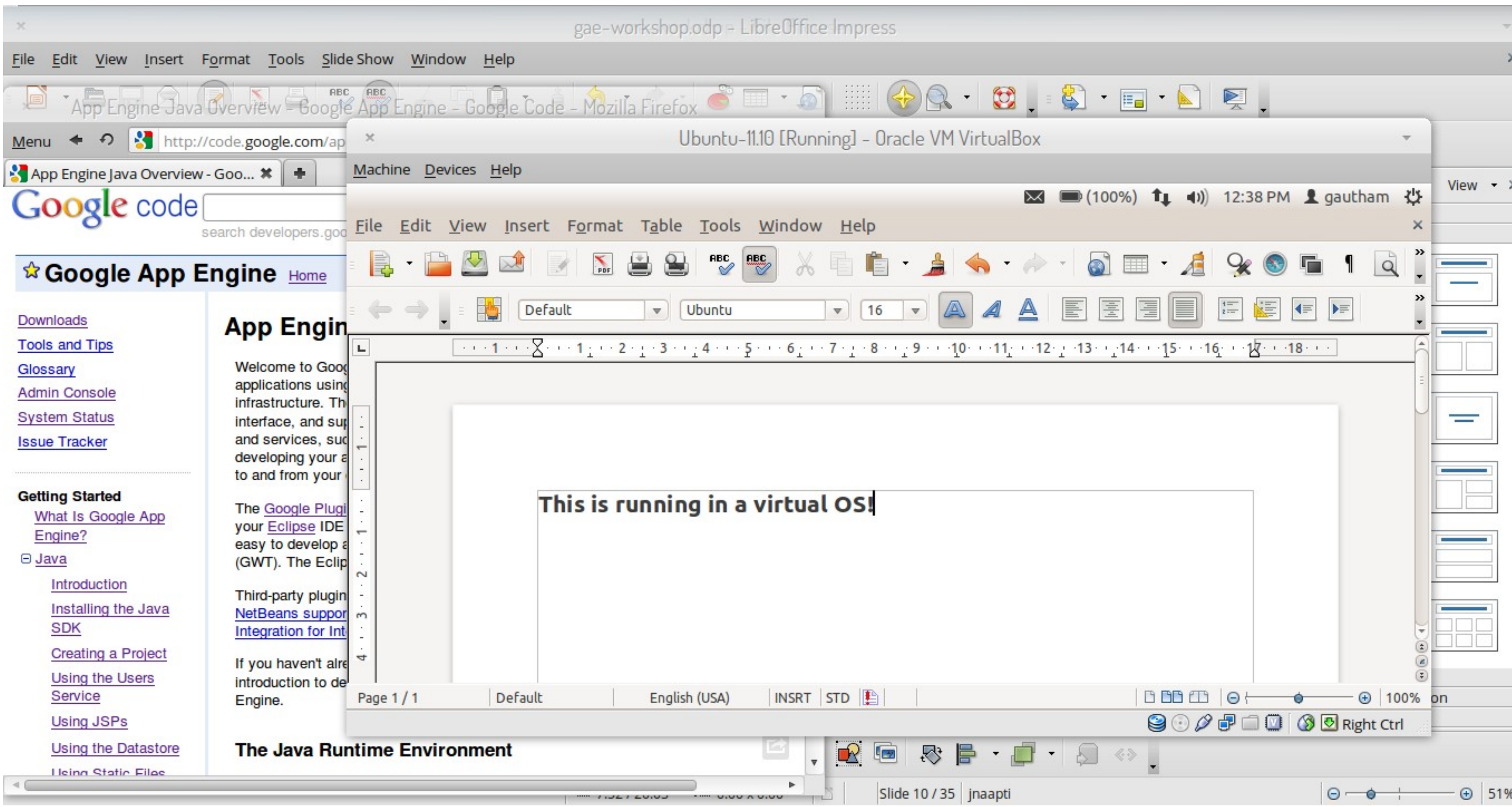
Virtual Machine

- Process Virtual Machine (eg: application virtualization via JVM)
- System Virtual Machine (eg: KVM)

Hypervisor

- Also called Virtual Machine Monitor
- A piece of computer software, firmware or hardware that creates and runs virtual machines
- Type-1 hypervisor or bare-metal hypervisor (eg: KVM)
- Type-2 hypervisor (eg: Virtualbox)

Virtualization with VirtualBox – An Example



Cloud Computing

Cloud Computing

Cloud computing is the use of computing resources (hardware and software) that are delivered as a service over a network (typically the Internet).

- Wikipedia

Have you used the cloud?

- If we talk about use of Cloud in terms of IaaS or PaaS, we may not have used it
- But if we talk about use of Cloud in terms of SaaS, there is a good possibility we have used it

A Little Bit of History

- The pains of procuring systems back in early 2000...
- Virtualization comes into picture
- It's easy to provide resources via the Internet
- Cloud computing becomes a reality

Then v/s Now

- Outlook v/s Gmail
- Local CRM v/s Salesforce
- MS Office v/s Google Docs
- Local storage v/s Dropbox
- Local enterprise storage v/s S3
- Expensive Server provisioning v/s EC2 and other AWS Services

Benefits

- No upfront costs involved
- Lesser ongoing costs (lesser sysadmins required, no hardware failures headaches)
- Short time use of resources costs much lesser
- Meeting peak demands and bursts with pay as you go model

Characteristics of Cloud Computing

- API based access
- Cost
- Device independence
- Virtualization
- Multitenancy

API Based Access

- Being able to manage resources via APIs
- Case study – Boto
  - A Python interface to access AWS Services
  - https://github.com/boto/boto

Multitenancy

- A single instance of a software is used to serve multiple “tenants”
- Advantages
  - Cost savings
  - Data mining/analytics
  - Ease of administration (release/maintenance/bug fixes etc)
- Disadvantage
  - Complexity of customization
- Case study: Google AppEngine Namespaces

IaaS/PaaS/SaaS

Cloud Computing Service Models

- The evolutionary path
  - Software as a Service
  - Platform as a Service
  - Infrastructure as a Service

Examples of IaaS, PaaS, SaaS

- Google AppEngine – PaaS
- Amazon AWS – IaaS and PaaS
- Rackspace – IaaS
- Heroku – PaaS
- Zoho applications – SaaS
- Gmail, Google Docs, Google Maps – SaaS

Cloud Deployment Models

- Public Clouds
- Private Clouds
- Hybrid Clouds
  - Cloud Bursting

Hosting your own Cloud

- Openstack
- Cloudstack
- Eucalyptus

Networking Tools

Networking Tools

- ifconfig
- dig
- ping
- traceroute
- netstat
- tcpdump
- resolv.conf
- ssh
- scp/rsync

Web Engineering in the Cloud

Web Engineering – The Components

- DNS
- Load Balancer
- Web Servers
- Application Servers
- Scaling web and application servers
- Data Stores/Cloud Storage
- Data analysis and the Hadoop ecosystem

Web Engineering - Storage

- Relational Databases
- NoSQL Databases
- Scaling datastores

Architecting for the Cloud

- A case study
- Understanding real scenarios
- Web Applications
- Media Storage
- Advertisement
- Any machine can go down anytime

Architecting for the Cloud

- Avoiding single points of failures
- Simulate scalability by trying with true peak loads
- Chaos monkey
- Monitor and make sure errors don't occur again
- Think about continual automation and automated administration

Introduction to Map/Reduce

Examples of Web Scale Data Analysis

- Distributed Grep - Look for a pattern in all the Tweets
- Inverted Index Building - This is what is used by search engines
- Sentiment Analysis
- Competition Analysis
- Log Analysis

Understanding the problem of Analysis

- Unlike in the case of retrieving data, in the case of analysis, we need to read through everything, but reads are slow in the disk.
- Let's see a simple math:
  - 1 Hard Disk read speed is 100MB/s
  - 100 Hard Disks read in parallel gives 10GB/s!
- Can we exploit this parallelism?

The Coin Counting Example

- You have a sack full of coins, and you are asked to separate them into 1, 2, 5 and 10 Rs coins and tell how many of each are present.
- Now, let's say you have few sacks full of coins and it will take you a lot of time to count it yourself – so you call a few other people to help you out.
- Now, let's say there is few rooms full of coins (like in some large temples in India) – how will you count them?

Coin Counting Problem – in depth

- You can't add more people to the same room – the room is already full.
- You can get a few more rooms, ask people to take some coins to the other room and then do the counting there, and come back with the coins and the final count.
- This will mean a lot of "traffic" in the corridor.
- So what's a better solution?

A Possible Solution to the Coin Counting Problem

- Unload the coins in different rooms rather than in the same room.
- Then get workers in different rooms. With an increase in coins, increase the number of rooms and workers.
- Let the workers in each room work independently.

This is how Map/Reduce frameworks work

Traditional Parallel Processing

- Use of threads, sharing data, synchronization
- Results in Deadlocks, Livelocks, Starvation etc
- Handling failures is complex

Parallel Programming is hard this way.

Requirements from a parallel processing framework

- Higher level programming constructs – don't need to deal with sockets, threading, locking, sharing data etc
- Manage failures - if a task fails or a system breaks down, we want the framework to transparently manage it
- Recoverability - If a system fails, another system must be able to pick up its workload
- Replication – if a system fails, we don't lose data – the framework should replicate data in multiple nodes
- Scalability – Adding more compute nodes should help us increase the compute capacity

Pulling data Or Pushing Computations?

- Pulling data for computation results in a bottleneck
- Every “database store” also has a “processor”.
- Instead of pulling the data for computation, can we think about pushing the computation out to where the data resides?
- Computation is in "bytes", may be a few MB of object code, that is still trivial compared to the data it works on

MapReduce

- Concept introduced by Google in 2004
- Framework is inspired by map and reduce functions found in functional programming languages
- Hadoop is an opensource implementation of MapReduce

MapReduce Frameworks

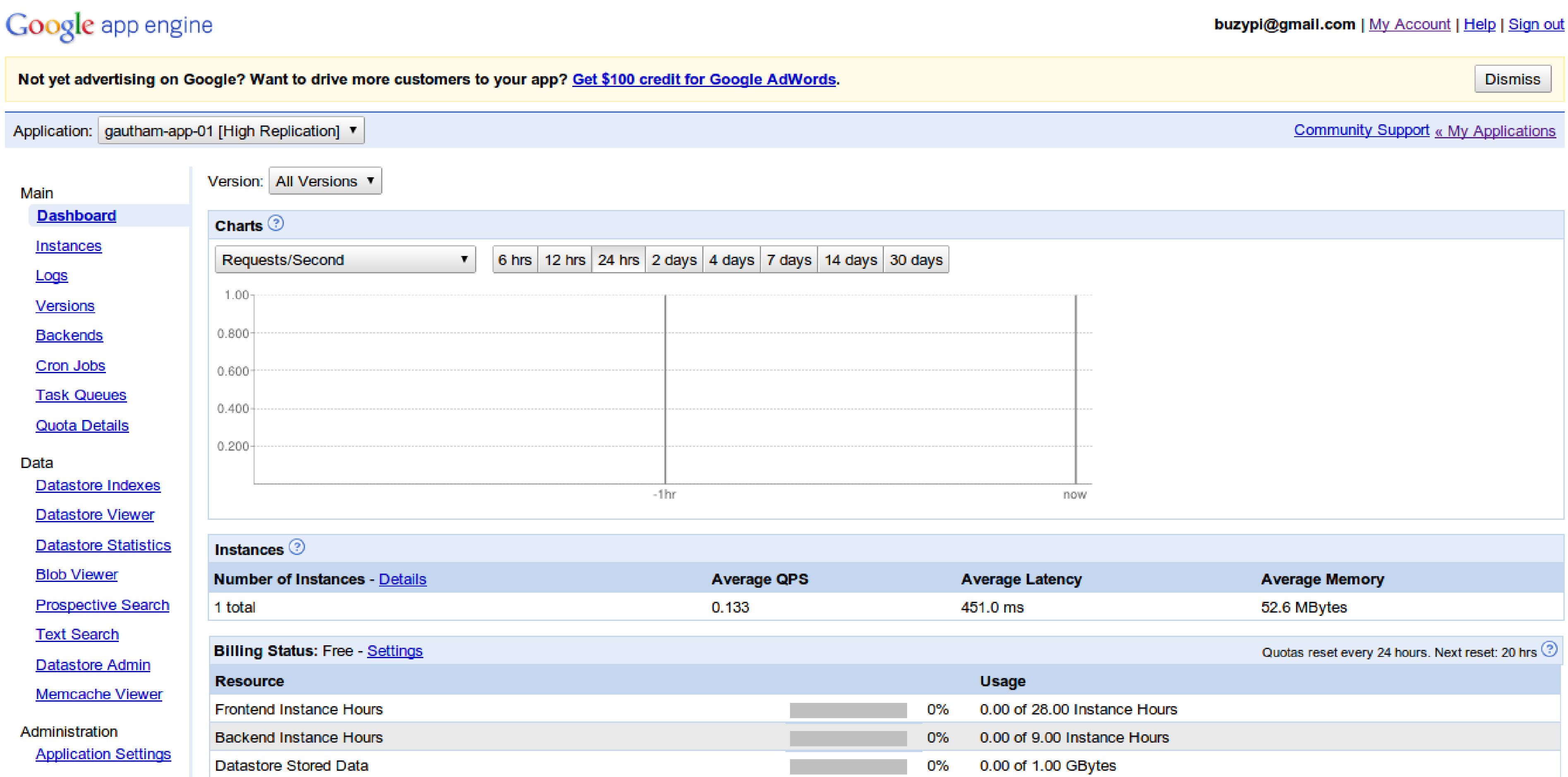
- Data is spread throughout machines before starting the task
- Computation is done in the nodes where data is stored
- Data is replicated in multiple machines to increase reliability
- Tasks are executed on multiple nodes just in case one of them is running slow

Using the Common Crawl Data – A Case Study

- The dump is a few 10s of TBs in size
- Where/How do you download it?
- Answer: You don't need to download it
- Instead you push your computation to where the data exists, perform your computation and then only fetch results you are interested in!

Google AppEngine

Google AppEngine



Google App Engine

- Free to try out
- Supports Java, Python and Go languages
- Does not provide direct access to the system running the application
- Instead allows us to work with application via commands and an admin-console
- You can have your first deployed app in minutes!

Google AppEngine - Handson

- Create an application in the dashboard
- Create the webapp2 Request Handler
- Test your application locally
- Upload your application

Google App Engine contd...

- Provides several useful building blocks that we will need to build applications. Example:
  - Data Storage
  - User Management
  - Mail Services
  - Caching
  - XMPP
  - Image Manipulation
  - Task Queues
  - ...

OpenStack

OpenStack

- Compute - Nova
- Object Storage - Swift
- Block Storage - Cinder
- Image Management - Glance
- Networking - Quantum
- Dashboard - Horizon
- APIs – Python and Shell based access, AWS Compatible

Hypervisors Supported

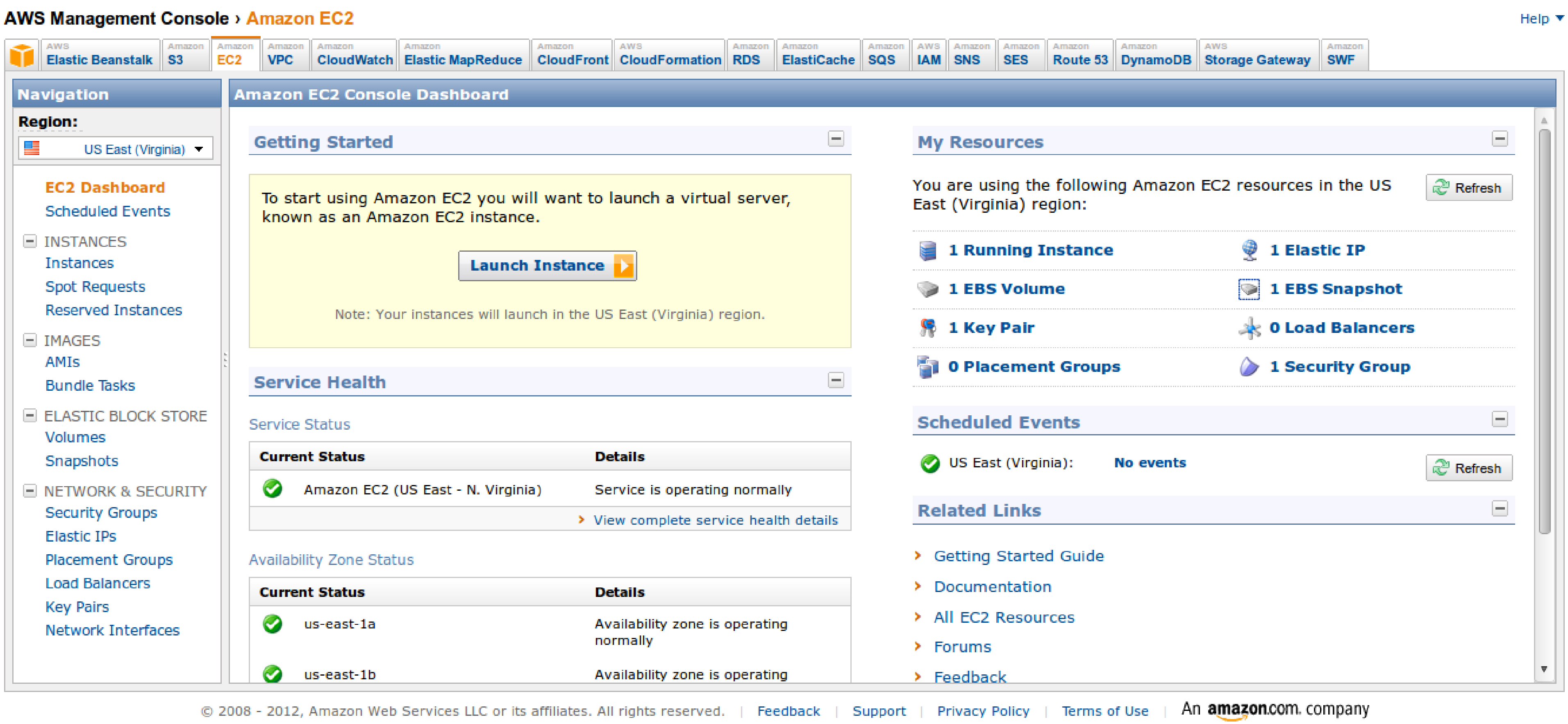
- KVM
- LXC
- QEMU
- UML
- VMware vSphere
- Xen
- PowerVM (IBM)
- Hyper-V (Microsoft)
- Bare Metal

Devstack

- A documented shell script to build complete OpenStack development environments.
- Very simple to install and try out OpenStack
- You can install the cloud in the cloud with this to get started!

Amazon Web Services

AWS Console



AWS Management Console › Amazon EC2
Help
Elastic Beanstalk
S3
EC2
VPC
CloudWatch
Elastic MapReduce
CloudFront
CloudFormation
RDS
ElastiCache
SQS
IAM
SNS
SES
Route 53
DynamoDB
Storage Gateway
SWF
Navigation
Region:
US East (Virginia)
EC2 Dashboard
Scheduled Events
INSTANCES
Instances
Spot Requests
Reserved Instances
IMAGES
AMIs
Bundle Tasks
ELASTIC BLOCK STORE
Volumes
Snapshots
NETWORK & SECURITY
Security Groups
Elastic IPs
Placement Groups
Load Balancers
Key Pairs
Network Interfaces
Amazon EC2 Console Dashboard
Getting Started
To start using Amazon EC2 you will want to launch a virtual server, known as an Amazon EC2 instance.
Launch Instance
Note: Your instances will launch in the US East (Virginia) region.
Service Health
Service Status
Current Status
Details
Amazon EC2 (US East - N. Virginia)
Service is operating normally
View complete service health details
Availability Zone Status
Current Status
Details
us-east-1a
Availability zone is operating normally
us-east-1b
Availability zone is operating
My Resources
You are using the following Amazon EC2 resources in the US East (Virginia) region:
Refresh
1 Running Instance
1 Elastic IP
1 EBS Volume
1 EBS Snapshot
1 Key Pair
0 Load Balancers
0 Placement Groups
1 Security Group
Scheduled Events
US East (Virginia):
No events
Refresh
Related Links
Getting Started Guide
Documentation
All EC2 Resources
Forums
Feedback
© 2008 - 2012, Amazon Web Services LLC or its affiliates. All rights reserved.
Feedback
Support
Privacy Policy
Terms of Use
An amazon.com company

Amazon Web Services (some popular services)

- Amazon Elastic Compute Cloud (EC2)
- Amazon Simple Storage Service (S3)
- Elastic Block Storage (EBS)
- Elastic Load Balancing (ELB)
- Amazon Relational Database Service (RDS)
- Amazon DynamoDB
- Auto Scaling
- Amazon ElastiCache

AWS Management Console

- Introduction to the AWS Management Console
- Working with various AWS Services
  - EC2
  - S3
  - RDS
  - ...

Regions and Availability Zones

- Regions and Availability Zones
  - Each region is in a different geographic area
  - Each Availability Zone is completely isolated from the other and connected to others via low latency links

EC2 Purchasing Models

- On-demand instances
- Reserved instances
- Spot instances

EC2 Instances

- Selecting an AMI
- Choosing the Instance Type
- Choosing the Availability Zone
- Choose on-demand or spot
- Enable Cloudwatch monitoring (if required)

EC2 Instance

- Choose the disk
- Tag your instance
- Associate a key-pair
- Configure the security group
- Review and Launch
- Associating an Elastic IP with the instance

AWS – A Few Case Studies

- Building web applications in AWS
- Ad serving via AWS
- Storing and processing media (sound/video/images) in AWS

Resources

- KVM
- Virtual Machine Manager
- QEMU Images
- AWS Documentation
- AWS Reference Architecture
- Google AppEngine Documentation