Using the Cloud to Crunch Your Data

Adrian Cockcroft –
acockcroft@netflix.com
What is Cloud Computing?
What is Capacity Planning

• We care about CPU, Memory, Network and Disk resources, and Application response times
• We need to know how much of each resource we are using now, and will use in the future
• We need to know how much headroom we have to handle higher loads
• We want to understand how headroom varies, and how it relates to application response times and throughput
Capacity Planning Norms

- Capacity is expensive
- Capacity takes time to buy and provision
- Capacity only increases, can’t be shrunk easily
- Capacity comes in big chunks, paid up front
- Planning errors can cause big problems
- Systems are clearly defined assets
- Systems can be instrumented in detail
Capacity Planning in Clouds

- Capacity is expensive
- Capacity takes time to buy and provision
- Capacity only increases, can’t be shrunk easily
- Capacity comes in big chunks, paid up front
- Planning errors can cause big problems
- Systems are clearly defined assets
- Systems can be instrumented in detail
Capacity is expensive


• Storage (Amazon S3)
  – $0.150 per GB – first 50 TB / month of storage used
  – $0.120 per GB – storage used / month over 500 TB

• Data Transfer (Amazon S3)
  – $0.100 per GB – all data transfer in
  – $0.170 per GB – first 10 TB / month data transfer out
  – $0.100 per GB – data transfer out / month over 150 TB

• Requests (Amazon S3 Storage access is via http)
  – $0.01 per 1,000 PUT, COPY, POST, or LIST requests
  – $0.01 per 10,000 GET and all other requests
  – $0 per DELETE

• CPU (Amazon EC2)
  – Small (Default) $0.10 per hour to Extra Large $0.80 per hour

• Network (Amazon EC2)
  – Inbound/Outbound around $0.10 per GB
Capacity comes in big chunks, paid up front

- **Capacity takes time to buy and provision**
  - No minimum price, monthly billing
  - “Amazon EC2 enables you to increase or decrease capacity within minutes, not hours or days. You can commission one, hundreds or even thousands of server instances simultaneously”

- **Capacity only increases, can’t be shrunk easily**
  - Pay for what is actually used

- **Planning errors can cause big problems**
  - Size only for what you need now
Systems are clearly defined assets

• You are running in a “stateless” multi-tenanted virtual image that can die or be taken away and replaced at any time
• You don’t know exactly where it is
• You can choose to locate “USA” or “Europe”
• You can specify zones that will not share components to avoid common mode failures
Systems can be instrumented in detail

- Need to use stateless monitoring tools
- e.g. Ganglia – automatic configuration
  - Multicast replicated monitoring state
  - No need to pre-define metrics and nodes
Use it to learn it...

• Focus on how you can use the cloud yourself to do large scale log processing
• You can upload huge datasets to a cloud and crunch them with a large cluster of computers using Amazon Elastic Map Reduce (EMR)
• Do it all from your web browser for a handful of dollars charged to a credit card.
• Here’s how
Cloud Crunch

The Recipe
You will need:
• A computer connected to the Internet
• The Firefox browser
• A Firefox specific browser extension
• A credit card and less than $1 to spend
• Big log files as ingredients
• Some very processor intensive queries
Recipe

First we will warm up our computer by setting up Firefox and connecting to the cloud.
Then we will upload our ingredients to be crunched, at about 20 minutes per gigabyte.
You should pick between one and twenty processors to crunch with, they are charged by the hour and the cloud takes about 10 minutes to warm up.
The query itself starts by mapping the ingredients so that the mixture separates, then the excess is boiled off to make a nice informative reduction.
Costs

• Firefox and Extension – free
• Upload ingredients – 10 cents/Gigabyte
• Small Processors – 11.5 cents/hour each
• Download results – 17 cents/Gigabyte
• Storage – 15 cents/Gigabyte/month
• Service updates – 1 cent/1000 calls
• Service requests – 1 cent/10,000 calls
• Actual cost to run two example programs as described in this presentation was 26 cents
Faster Results at Same Cost!

• You may have trouble finding enough data and a complex enough query to keep the processors busy for an hour. In that case you can use fewer processors.

• Conversely if you want quicker results you can use more and/or larger processors.

• Up to 20 systems with 8 CPU cores = 160 cores is immediately available. Oversize on request.
Step by Step

• Walkthrough to get you started
• Run Amazon Elastic MapReduce examples
• Get up and running before this presentation is over....
Step 1 – Get Firefox

Step 2 – Get S3Fox Extension

• Next select the Add-ons option from the Tools menu, select “Get Add-ons” and search for S3Fox.
Step 3 – Learn About AWS

• Bring up [http://aws.amazon.com/](http://aws.amazon.com/) to read about the services.

• Amazon S3 is short for Amazon Simple Storage Service, which is part of the Amazon Web Services product

• We will be using Amazon S3 to store data, and Amazon Elastic Map Reduce (EMR) to process it.

• Underneath EMR there is an Amazon Elastic Compute Cloud (EC2), which is created automatically for you each time you use EMR.
What is S3, EC2, EMR?

- Amazon Simple Storage Service lets you put data into the cloud that is addressed using a URL. Access to it can be private or public.
- Amazon Elastic Compute Cloud lets you pick the size and number of computers in your cloud.
- Amazon Elastic Map Reduce automatically builds a Hadoop cluster on top of EC2, feeds it data from S3, saves the results in S3, then removes the cluster and frees up the EC2 systems.
Step 4 – Sign Up For AWS

- Go to the top-right of the page and sign up at http://aws.amazon.com/
- You can login using the same account you use to buy books!
Step 5 – Sign Up For Amazon S3

• Follow the link to [http://aws.amazon.com/s3/](http://aws.amazon.com/s3/)
Amazon Simple Storage Service

United States

Storage
$0.150 per GB - first 50 TB / month of storage used
$0.140 per GB - next 50 TB / month of storage used
$0.130 per GB - next 400 TB / month of storage used
$0.120 per GB - storage used / month over 500 TB

Data Transfer
$0.030 per GB - all data transfer in April 1, 2009 through June 30, 2009 -
(Note: data transfer in will return to its normal price of $0.100 per GB on July 1)
$0.170 per GB - first 10 TB / month data transfer out
$0.130 per GB - next 40 TB / month data transfer out
$0.110 per GB - next 100 TB / month data transfer out
$0.100 per GB - data transfer out / month over 150 TB
Step 6 – Sign Up For Amazon EMR

EC2 Signup

• The EMR signup combines the other needed services such as EC2 in the sign up process and the rates for all are displayed.

• The EMR costs are in addition to the EC2 costs, so 1.5 cents/hour for EMR is added to the 10 cents/hour for EC2 making 11.5 cents/hour for each small instance running Linux with Hadoop.
Amazon Elastic MapReduce uses Amazon Elastic Compute Cloud to run your job flows and Amazon Simple Storage Service to store and access your data. After completing the sign-up process, you will have signed up to use Amazon Elastic Compute Cloud (You have already signed up to use Amazon Simple Storage Service).

**Amazon Elastic MapReduce**

- $0.015 per Small Instance (m1.small) instance-hour (or partial hour)
- $0.060 per Large Instance (m1.large) instance-hour (or partial hour)
- $0.120 per Extra Large Instance (m1.xlarge) instance-hour (or partial hour)
- $0.030 per High-CPU Medium Instance (c1.medium) instance-hour (or partial hour)
- $0.120 per High-CPU Extra Large Instance (c1.xlarge) instance-hour (or partial hour)

Usage for other Amazon Web Services is billed separately from Amazon Elastic MapReduce.

**Amazon Elastic Compute Cloud**

**United States**

**Amazon EC2 running Linux/UNIX**

- $0.10 per Small Instance (m1.small) instance-hour (or partial hour)
- $0.40 per Large Instance (m1.large) instance-hour (or partial hour)
- $0.80 per Extra Large Instance (m1.xlarge) instance-hour (or partial hour)
- $0.20 per High-CPU Medium Instance (c1.medium) instance-hour (or partial hour)
- $0.80 per High-CPU Extra Large Instance (c1.xlarge) instance-hour (or partial hour)
Step 7 - How Big are EC2 Instances?

- The compute power is specified in a standard unit called an EC2 Compute Unit (ECU).
- Amazon states “One EC2 Compute Unit (ECU) provides the equivalent CPU capacity of a 1.0-1.2 GHz 2007 Opteron or 2007 Xeon processor.”
Standard Instances

• Small Instance (Default) 1.7 GB of memory, 1 EC2 Compute Unit (1 virtual core with 1 EC2 Compute Unit), 160 GB of instance storage, 32-bit platform.

• Large Instance 7.5 GB of memory, 4 EC2 Compute Units (2 virtual cores with 2 EC2 Compute Units each), 850 GB of instance storage, 64-bit platform.

• Extra Large Instance 15 GB of memory, 8 EC2 Compute Units (4 virtual cores with 2 EC2 Compute Units each), 1690 GB of instance storage, 64-bit platform.
Compute Intensive Instances

- High-CPU Medium Instance 1.7 GB of memory, 5 EC2 Compute Units (2 virtual cores with 2.5 EC2 Compute Units each), 350 GB of instance storage, 32-bit platform.
- High-CPU Extra Large Instance 7 GB of memory, 20 EC2 Compute Units (8 virtual cores with 2.5 EC2 Compute Units each), 1690 GB of instance storage, 64-bit platform.
Step 8 – Getting Started


• There is a Getting Started guide and Developer Documentation including sample applications

• We will be working through two of those applications.

• There is also a very helpful FAQ page that is worth reading through.
Step 9 – What is EMR?
Step 10 – How To MapReduce?

• Code directly in Java
• Submit “streaming” command line scripts
  – EMR bundles Perl, Python, Ruby and R
• Code MR sequences in Java using Cascading
• Process log files using Cascading Multitool
• Write dataflow scripts with Pig (not yet in EMR)
• Write SQL queries using Hive (not in EMR)
Step 11 – Setup Access Keys

• Getting Started Guide

The URL to visit to get your key is:

Greetings from Amazon Web Services,

Thank you for signing up for Amazon Simple Storage Service. You now have immediate access to Amazon Simple Storage Service.

You need Access Identifiers to make valid web service requests. Please visit the Access Identifiers section of your account to obtain your identifier and to learn more:
Enter Access Keys in S3Fox

• Open S3 Firefox Organizer and select the Manage Accounts button at the top left.
• Enter your access keys into the popup.
Step 12 – Create An Output Folder

Click Here
Setup Folder Permissions
Step 13 – Run Your First Job

- See the Getting Started Guide at:
  http://docs.amazonwebservices.com/ElasticMapReduce/2009-03-31/
  GettingStartedGuide/gsConsoleRunJob.html

  login to the EMR console at
  https://console.aws.amazon.com/elasticmapreduce/home

  create a new job flow called “Word crunch”, and select the sample
  application word count.
Set the Output S3 Bucket

<table>
<thead>
<tr>
<th>Input Location*</th>
<th>elasticmapreduce/samples/wordcount/input</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output Location*</td>
<td>crunchie/wordcount/output/2009-06-14</td>
</tr>
<tr>
<td>Mapper*</td>
<td>elasticmapreduce/samples/wordcount/wordSplitter.py</td>
</tr>
<tr>
<td>Reducer*</td>
<td>aggregate</td>
</tr>
</tbody>
</table>
Pick the Instance Count

• One small instance (i.e. computer) will do...
• Cost will be 11.5 cents for 1 hour (minimum)
Start the Job Flow

The Job Flow takes a few minutes to get started, then completes in about 5 minutes run time.
Step 14 – View The Results

- In S3Fox click on the refresh icon, then doubleclick on the crunchie folder
- Keep clicking until the output file is visible
Save To Your PC

- S3Fox – click on left arrow
- Save to PC
- Open in TextEdit
- See that the word “a” occurred 14716 times

boring…. so try a more interesting demo!
Step 15 – Crunch Some Log Files

- Create a new output bucket with a new name
- Start a new job flow using the CloudFront Demo
- This uses the Cascading Multitool

```java
Jar Location*: elasticmapreduce/samples/cloudfront/logprocessor.jar
Jar Arguments*: 
- -input s3n://elasticmapreduce/samples/cloudfront/input
- -output s3n://crunchie/cloudfront/output/2009-06-14-20
- -start any
- -end 2009-06-14-20
- -timeBucket 300
- -overallVolumeReport
- -objectPopularityReport
- -clientReport
- -edgeLocationReport
```
Step 16 – How Much Did That Cost?

### Summary of This Month's Activity as of June 15, 2009

**Billing Cycle for this Report:** June 1 - June 30, 2009  
*AWS service usage charges on this page currently show activity through approximately 06/14/2009 23:59 GMT.*

<table>
<thead>
<tr>
<th>Rate</th>
<th>Usage</th>
<th>Totals</th>
</tr>
</thead>
</table>
| **Amazon Elastic Compute Cloud**  
View/Edit Service | | |
| **Amazon EC2 running Linux/UNIX**  
View/Edit Service | | |
| $0.10 per Small Instance (m1.small) instance-hour (or partial hour) | 2 Hrs | 0.20 |
| **Amazon Elastic MapReduce**  
View/Edit Service | | |
| $0.015 per Small Instance (m1.small) instance-hour (or partial hour) | 2 Hrs | 0.03 |
| **Amazon Simple Storage Service**  
View/Edit Service | | |
| $0.170 per GB - first 10 TB / month data transfer out | 0.000339 GB | 0.01 |
| $0.01 per 1,000 PUT, COPY, POST, or LIST requests | 243 Requests | 0.01 |
| $0.01 per 10,000 GET and all other requests | 586 Requests | 0.01 |
| **Taxes**  
Estimated Taxes  
(Due July 1, 2009) | | |
| **Charges due on July 1, 2009**† | | 0.26 |
Wrap Up

• Easy, cheap, anyone can use it
• Now let’s look at how to write code...
Log & Data Analysis using Hadoop

Based on slides by
Shashi Madappa
smadappa@netflix.com
Agenda

1. Hadoop
   - Background
   - Ecosystem
   - HDFS & Map/Reduce
   - Example
2. Log & Data Analysis @ Netflix
   - Problem / Data Volume
   - Current & Future Projects
3. Hadoop on Amazon EC2
   - Storage options
   - Deployment options
Hadoop

• Apache open-source software for reliable, scalable, distributed computing.
• Originally a sub-project of the Lucene search engine.
• Used to analyze and transform large data sets.
• Supports partial failure, Recoverability, Consistency & Scalability
Hadoop Sub-Projects

- **Core**
  - **HDFS**: A distributed file system that provides high throughput access to data.
  - **Map/Reduce**: A framework for processing large data sets.
- **HBase**: A distributed database that supports structured data storage for large tables
- **Hive**: An infrastructure for ad hoc querying (sql like)
- **Pig**: A data-flow language and execution framework
- **Avro**: A data serialization system that provides dynamic integration with scripting languages. Similar to Thrift & Protocol Buffers
- **Cascading**: Executing data processing workflow
- **Chukwa, Mahout, ZooKeeper**, and many more
Hadoop Distributed File System (HDFS)

Features
• Cannot be mounted as a “file system”
• Access via command line or Java API
• Prefers large files (multiple terabytes) to many small files
• Files are write once, read many (append coming soon)
• Users, Groups and Permissions
• Name and Space Quotas
• Blocksize and Replication factor are configurable per file

Commands: hadoop dfs -ls, -du, -cp, -mv, -rm, -rmr

Uploading files
- hadoop dfs -put foo mydata/foo
- cat ReallyBigFile | hadoop dfs -put - mydata/ReallyBigFile

Downloading files
- hadoop dfs -get mydata/foo foo
- hadoop dfs –tail [-f] mydata/foo
Map/Reduce

Map/Reduce is a programming model for efficient distributed computing

- Data processing of large dataset
- Massively parallel (hundreds or thousands of CPUs)
- Easy to use
  - Programmers don’t worry about socket(), etc.
- It works like a Unix pipeline:
  - `cat * | grep | sort | uniq -c | cat > output`
  - `Input | Map | Shuffle & Sort | Reduce | Output`
- Efficiency from streaming through data, reducing seeks
- A good fit for a lot of applications
  - Log processing
  - Index building
  - Data mining and machine learning
Map/Reduce

• Simple Dataflow
• Detailed Dataflow
Input & Output Formats

• The application also chooses input and output formats, which define how the persistent data is read and written. These are interfaces and can be defined by the application.

• InputFormat
  – Splits the input to determine the input to each map task.
  – Defines a RecordReader that reads key, value pairs that are passed to the map task

• OutputFormat
  – Given the key, value pairs and a filename, writes the reduce task output to persistent store.
Map/Reduce Processes

• Launching Application
  – User application code
  – Submits a specific kind of Map/Reduce job
• JobTracker
  – Handles all jobs
  – Makes all scheduling decisions
• TaskTracker
  – Manager for all tasks on a given node
• Task
  – Runs an individual map or reduce fragment for a given job
  – Forks from the TaskTracker
Word Count Example

• Mapper
  – Input: value: lines of text of input
  – Output: key: word, value: 1

• Reducer
  – Input: key: word, value: set of counts
  – Output: key: word, value: sum

• Launching program
  – Defines the job
  – Submits job to cluster
public static class WordCountMapper extends MapReduceBase implements Mapper<LongWritable, Text, Text, IntWritable> {
    private final static IntWritable one = new IntWritable(1);
    private Text word = new Text();
    public void map(LongWritable key, Text value, OutputCollector<Text, IntWritable> output, Reporter reporter) throws IOException {
        String line = value.toString();
        StringTokenizer itr = new StringTokenizer(line, ",");
        while (itr.hasMoreTokens()) {
            word.set(itr.nextToken());
            output.collect(word, one);
        }
    }
}
public static class WordCountReducer extends MapReduceBase implements
Reducer<Text, IntWritable, Text, IntWritable> {
    public void reduce(Text key, Iterator<IntWritable> values, OutputCollector<Text,
IntWritable> output, Reporter reporter) throws IOException {
        int sum = 0;
        while (values.hasNext()) {
            sum += values.next().get();
        }
        output.collect(key, new IntWritable(sum));
    }
}
public class WordCount {

    public static void main(String[] args) throws IOException {
        JobConf conf = new JobConf(WordCount.class);
        // the keys are words (strings)
        conf.setOutputKeyClass(Text.class);
        // the values are counts (ints)
        conf.setOutputValueClass(IntWritable.class);
        conf.setMapperClass(WordCountMapper.class);
        conf.setReducerClass(WordCountReducer.class);
        conf.setInputPath(new Path(args[0]));
        conf.setOutputPath(new Path(args[1]));
        JobClient.runJob(conf);
    }
}
Running the Example

• Input
Welcome to Netflix
This is a great place to work

• Output
Netflix 1
This 1
Welcome 1
a 1
great 1
is 1
place 1
to 2
work 1
### WWW Access Log Structure

<table>
<thead>
<tr>
<th>Event Time</th>
<th>2009-06-23 16:20:58</th>
</tr>
</thead>
<tbody>
<tr>
<td>IP Address</td>
<td>xxx.xxx.xxx.xxx</td>
</tr>
<tr>
<td>Method</td>
<td>GET</td>
</tr>
<tr>
<td>URL</td>
<td>/SetRating</td>
</tr>
<tr>
<td>QUERY</td>
<td>?widgetid=M70071602&amp;pageId=page-MemberHomeQT</td>
</tr>
<tr>
<td>Status</td>
<td>200</td>
</tr>
<tr>
<td>Bytes</td>
<td>13010</td>
</tr>
<tr>
<td>Agent</td>
<td>Mozilla/5.0 (Macintosh; U; PPC Mac OS X 10_4_11; en) AppleWebKit/525.27.1 (KHTML, like Gecko) Version/3.2.1 Safari/525.27.1</td>
</tr>
<tr>
<td>Cookies</td>
<td>lastHitTime=1240528857147; etc...</td>
</tr>
<tr>
<td>Referrer</td>
<td><a href="http://www.netflix.com/MemberHome">http://www.netflix.com/MemberHome</a></td>
</tr>
<tr>
<td>ID</td>
<td>SfD32grAIGoAGftkcAAAADu</td>
</tr>
</tbody>
</table>
Analysis using access log

• Top
  – URLs
  – Users
  – IP
  – Size
• Time based analysis
• Session
  – Duration
  – Visits per session
• Identify attacks (DOS, Invalid Plug-in, etc)
• Study Visit patterns for
  – Resource Planning
  – Preload relevant data
• Impact of WWW call on middle tier services
Why run Hadoop in the cloud?

• “Infinite” resources
  – Hadoop scales linearly
  – Elasticity
  – Run a large cluster for a short time
  – Grow or shrink a cluster on demand
Common sense and safety

• Amazon security is good
  – But you are a few clicks away from making a data set public by mistake

• Common sense precautions
  – Before you try it yourself...
  – Get permission to move data to the cloud
  – Scrub data to remove sensitive information

• System performance monitoring logs are a good choice for analysis
Questions?
acockcroft@netflix.com