A programming model in Cloud: MapReduce

→ Programming model and implementation for processing and generating large data sets

→ Users specify
  - a map function to generate a set of intermediate key/value pairs
  - a reduce function that merge all intermediate values associated with the same intermediate key

→ Programs can be automatically parallelized and executed on a large cluster of machines
  - Partitioning input data
  - Scheduling execution
  - Handling machine failure
  - Managing communication

→ Allow users without much experience to utilize the resources of a large distributed system
An example: Count word frequency

```java
map(String key, String value):
    // key: document name
    // value: document contents
    for each word w in value:
        EmitIntermediate(w, "1");

reduce(String key, Iterator values):
    // key: a word
    // values: a list of counts
    int result = 0;
    for each v in values:
        result += ParseInt(v);
    Emit(AsString(result));
```
MapReduce operations

When the user program calls the MapReduce function, the following sequence of operations occurs:

- Split the input files into M pieces; starts up many copies of the program on a cluster of machines: one master and many workers.
- Master assigns map tasks and reduce tasks to workers.
- A Map worker
  - reads input split,
  - abstracts key/value pairs from the input data,
  - passes each pair to the Map function,
  - produces the intermediate key/value pairs using the Map function
  - Buffers the intermediate key/value pairs
- The buffered pairs are written to local disk; the disk locations of the data are passed back to the master; the master forwards the locations to the reduce workers.
- The reduce workers uses remote procedure calls to read the buffered intermediate key/value pairs from the local disks of the map workers.
- The reduce workers passes the intermediate key/value pairs to the Reduce function.
- After all map and reduce tasks have been completed, the MapReduce call in the user program returns back to the user code.
MapReduce execution flow
MapReduce library

- Partitioning the input data
- Scheduling execution
  - Scheduling map and reduce workers
- Managing communication
  - Reduce workers retrieve the intermediate results from Map workers
- Handling machine failure
  - Ping every worker periodically
  - If no response in a certain amount of time, mark the worker as failed
  - Any map or reduce tasks in progress on a failed worker will be rescheduled
  - The map tasks completed by the failed worker will be rescheduled
  - The reduce tasks completed by the failed worker are rescheduled
Reference papers

- “MapReduce: Simplified Data Processing on Large Clusters”
- “Xen and the Art of Virtualization”
Structured and semi-structured data in google
Popular Cloud Systems

➔ Google
  - App Engine
  - The fundamental techniques: MapReduce, GFS, BigTable

➔ Amazon
  - AWS: providing Cloud services such as EC2, S3, SimpleDB etc

➔ Microsoft
  - Azure
Google provides numerous online services

Search
- Alerts
  - Get email updates on the topics of your choice
- Blog Search
  - Find blogs on your favorite topics
- Book Search
  - Search the full text of books
- Checkout
  - Complete online purchases more quickly and securely
- Desktop
  - Search and personalize your computer
- Earth
  - Explore the world from your computer
- Finance
  - Business info, news, and interactive charts
- Google 411
  - Find and connect with businesses from your phone, for free
- iGoogle
  - Add news, games and more to the Google homepage
- Images
  - Search for images on the web
- Maps
  - View maps and directions
- News - now with archive search
  - Search thousands of news stories
- Notebook
  - Clip and collect information as you surf the web
- Patent Search
  - Search the full text of US Patents
- Product Search
  - Search for stuff to buy
- Scholar
  - Search scholarly papers

Explore and innovate
- Code
  - Download APIs and open source code
- Custom Search
  - Create a personalized search experience for your community
- Labs
  - Explore Google's technology playground

Communicate, show & share
- Blogger
  - Share your life online with a blog – it's fast, easy, and free
- Calendar
  - Organize your schedule and share events with friends
- Docs
  - Create and share your projects online and access them from anywhere
- Gmail
  - Fast, searchable email with less spam
- Groups
  - Create mailing lists and discussion groups
- Orkut
  - Meet new people and stay in touch with friends
- Picasa
  - Find, edit and share your photos
- Reader
  - Get all your blogs and news feeds fast
- SketchUp
  - Build 3D models quickly and easily
- Talk
  - IM and call your friends through your computer
- Translate
  - View web pages in other languages
- YouTube
  - Watch, upload and share videos

Computer Science, University of Warwick
Bigtable: a distributed storage system in Google Cloud

→ Resembles a database

- Aims to scale to a very large size: petabyte of data across thousands of nodes

- Provides a simple data model that
  - Data is indexed using row and column names
  - Data is treated as strings
  - supports dynamic partition of the table
  - Allows clients to reason about the locality properties
Bigtable: a distributed storage system in Google Cloud

- A row key in a table is an arbitrary string

- Bigtable maintains data in lexicographic order by row key
  - Web pages in the same domain are grouped together into contiguous rows by reversing the hostname components of the URLs

- The row range for a table is dynamically partitioned

- Each row range is called a tablet
  - The unit of distribution and load balancing
  - Tablets are distributed and stored in multiple machines
  - Clients can control data locality by naming the row
Column family
  - Category of the column

Column key
  - Family: qualifier

Timestamps
  - 64-bit integers
  - specify only new enough versions be kept

API
  - creating and deleting tables and column families
A Bigtable cluster stores a number of tables.

Each table consists of a set of tablets.

Each tablet contains all data associated with a row range.

Initially, each table consists of just one tablet.

As a table grows, it is automatically split into multiple tablets, each approximately 100-200 MB in size by default.
Infrastructure of BigTable

- If a number of machines are needed to store the a BigTable
  - One machine will be elected as the master;
  - Other machines are called tablet servers

- The master is responsible for assigning tablets to tablet servers, balancing tablet-server load

- Each tablet server manages a set of tablets (typically we have somewhere between ten to a thousand tablets per tablet server)

- Tablet location: three level hierarchy
  - A chubby file contains the location of root tablet
  - Root tablet contains the location of all tablets
Bigtable uses Chubby to keep track of tablet servers

When a table server starts, it creates
Google Cloud Computing: Focusing on User Data

- User data is stored in “Cloud”
- Data access is not constrained by geographic locations
- Data can be conveniently shared
Google App Engine

- Complex Server structure: Apache, SQL, etc
- Endless monitoring and debugging
- Exhausting System upgrading and fault tolerance

Concise development and running platform for web applications

Running on the data centre provided by Google

Managing the entire life cycle of web applications
Google App Engine

- Enable clients to run web applications on Google’s infrastructure

- Clients do not have to maintain servers
  - Users just upload the applications
  - The applications can then serve the users of the web applications

- Supports Java and Python language

- You pay for what you use
  - There are no set-up and maintenance costs
Run your web applications on Google's infrastructure.
Build apps on the same scalable systems that power
Google applications.

Featured: Serving
BuddyPoke's 12 Million
users with App Engine

App Engine has been a
godsend in terms of our
scalability...we saw a 8x jump
in installs overnight and we
are now serving 12 million
users!

Dave Westwood, BuddyPoke!

Who's using App Engine? | Why App Engine?

Getting Started
1. Sign up for an App Engine account.
2. Download the App Engine SDK.
3. Read the Getting Started Guide.
4. Check out the App Gallery to see sample applications.

Demonstration
Developing and deploying an app on Google App Engine.
Watch Now

Recent Articles
Running App Engine Applications on Django
In this article we explain how to port applications that were originally
designed for App Engine with Django
to any environment that supports Django. The helper application
provides a Django based
implementation of the App Engine
APIs...

Building an OpenSocial App with Google App Engine
Google File System

- Provide the traditional file system interface

- Most files are mutated by appending new data rather than overwriting existing data
  - Intermediate results produced in one machine and processed in another
  - Developed atomic append operation so that multiple clients can append data to the same file concurrently
GFS Architecture

- Consists of a single Master and multiple chunkservers
- Files are divided into chunks
Lock service

- Allow its clients to synchronize their activities and reach agreement. For example, electing a leader from a set of equivalent servers

Similar to a simple file system that performs whole-file reads and writes, augmented with advisory locks and with notification of events such as file modification

Both GFS and Bigtable use Chubby to store a small amount of meta-data (e.g., root of their data structure)
- Doesn’t support a full relational data model
- Support client applications that wish to store data
- Allow clients to reason about the locality properties of the data
  - Clients can control the locality of the data
- Data is indexed using row and column names that can be arbitrary string
- Treat data as uninterpreted string
  - Clients often serialize structured and semi-structured data into strings
- Don’t consider the possibility multiple copies of the same data
- Let users tell us what data belongs in memory and what data should stay on disk
- Have no complex queries to execute
The row range for a table can be dynamically partitioned.

Each row range is called a tablet.
- The unit of distribution and load balancing

Maintain bigtables in lexicographic order by row key.
- Clients can select row key to get good locality for data access

For example, pages in the same domain are grouped together into contiguous rows by reversing the hostname components of URLs.
Explore Products

- Infrastructure Services
  - Amazon Elastic Compute Cloud (Amazon EC2)
  - Amazon SimpleDB
  - Amazon Simple Storage Service (Amazon S3)
  - Amazon Simple Queue Service (Amazon SQS)
  - AWS Premium Support
- Payments & Billing
- On-Demand Workforce
- Web Search & Information
- Amazon Fulfillment & Associates

Welcome to the Cloud
Amazon Web Services makes cloud computing a reality for hundreds of thousands of customers looking for a cost-effective infrastructure to deploy highly scalable and dependable solutions.

› Learn how you can benefit from cloud computing

News & Events

<table>
<thead>
<tr>
<th>What's New</th>
<th>Media Coverage</th>
<th>Events</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nov 06, 2008</td>
<td>Announcing the Finalists in the AWS Start-Up Challenge</td>
<td></td>
</tr>
<tr>
<td>Nov 03, 2008</td>
<td>Salesforce.com Announces Force.com Toolkit for AWS</td>
<td></td>
</tr>
<tr>
<td>Oct 23, 2003</td>
<td>Amazon EC2 Running Windows Server Now Available</td>
<td></td>
</tr>
<tr>
<td>Oct 23, 2003</td>
<td>Amazon EC2 Exits Beta and Now Offers a Service Level Agreement</td>
<td></td>
</tr>
<tr>
<td>Oct 23, 2003</td>
<td>AWS Provides Glimpse at New Amazon EC2 Features</td>
<td></td>
</tr>
</tbody>
</table>

› view all
Amazon Elastic Computing Cloud

SQS: Simple Queue Service
EC2: Running Instance of Virtual Machines
EBS: Elastic Block Service, Providing the Block Interface, Storing Virtual Machine Images
S3: Simple Storage Service, SOAP, Object Interface
SimpleDB: Simplified Database
AWS is Production-level Cloud System
Amazon Web Service

→ **Amazon Elastic Compute Cloud (EC2)**
  - A web service that enables the users to launch and manage server instances in Amazon’s data centres using APIs or tools and utilities.

→ **Amazon Simple Storage Service (S3)**

→ **Amazon SimpleDB** is a web service for running queries on structured data in real time
  - **Data model**
    - Customer account, Domain, Items, Attributes, Values
  - **Operations**
    - Subscriber, Request, Response
  - **API**
    - CreateDomain, DeleteDomain, PutAttributes, Select,
  - **Consistency**
    - Keeps multiple copies of each domain
  - **Concurrent applications**
Amazon Simple Storage Service

S3 has a simple web services interface that can be used to store and retrieve any amount of data, at any time, from anywhere on the web.

- **Buckets**
- **Objects**
  - Objects consist of object data and metadata
  - The metadata is a set of name-value pairs that describe the object
- **Keys**
- **Versioning**
  - An object consists of two components: a key and a version ID
- **Operations**
  - Create a Bucket, write an object, listing keys
- **provides a REST and a SOAP interface**
  - The REST API is an HTTP interface to S3
  - The SOAP API uses document literal encoding
- **Regions**: choose the region where S3 will store the bucket user create
  - US, US-West, EU
Amazon Simple Storage Service

Amazon S3 data consistency model

- High availability is achieved by replicating data across multiple servers within Amazon data centre

- A process write a new object to S3 and immediately attempts to read it. Until the change is fully propagated, S3 might return “key does not exist”
Microsoft Cloud Computing
Microsoft Online

- Enterprise-class communicating and cooperative software and services
- Provide ordering services for various scale of enterprise

Business Productivity Online Suite

- SharePoint Online
- Exchange Online
- Office Live Meeting
- Office Communications Online
- Dynamics CRM Online

Computer Science, University of Warwick
Optimizing Resource Consumptions in Cloud Systems

- A physical cluster (Cloud) hosts multiple Virtual Clusters, each Virtual cluster serving a particular type of requests.

- Each Virtual cluster is associated with soft QoS requirements, i.e., at least a certain proportion of requests served by the Virtual cluster should finish by a certain time.

- The objective is to minimize the number of physical nodes used to host the virtual clusters, subject to meeting all Virtual Clusters’ QoS requirements.
Our solution

→ Exploiting the queuing theory to distribute requests at low costs

→ Developing a Genetic algorithm to update the request distribution and resource allocations when necessary

→ Developing a cost model to establish the plan to adjust the resource allocations to Virtual Machines
Two level of Virtual Cluster Managers in the system

- **Local manager**
  - Situated in each Virtual Cluster
  - Applying queuing theory to compute the maximum requests arrival rates that a VM in the VC can accommodate
  - Calculating possible VM combinations that can satisfy the VC’s desired QoS

- **Global manager**
  - One global manager exists for the entire system
  - Managing multiple Virtual Clusters
  - Coordinating the node mapping decisions made by individual local managers
  - Aiming to use as fewer physical nodes as possible to host all VCs in the system and satisfy the desired QoS of all VCs
The fluctuation of resource demands may cause the resource fragmentation in the system.

The global manager monitors the resource fragmentation level.

When necessary, applying a genetic algorithm to determine more suitable state (i.e., VM-to-node mapping and resource allocations) in the Virtual Clusters.

The Virtual Machines may be created, deleted and migrated to other nodes in the system, and the resource capacity allocated to a VM may also be adjusted.

The VM operations (creation, deletion, migration and resource adjustment) impose overheads on the system; a cost model is established and used to construct the best reconfiguration plan for reconfigure the Virtual Cluster from the current state to the state calculated by the genetic algorithm.
The Genetic Algorithm

- Representation of the solution, including both VM-to-node mapping and allocation of resource capacities
  - The VM-to-node mapping and the allocations of resource capacity to VMs are represented using a three dimensional array, S, in which an element $S[i, j, k]$ represents the capacity of resource $r_k$ allocated to $VM_{ij}$ in node $n_i$.
  - Crossover and Mutation operations are performed to generate the next generation of solution
Crossover and Mutation

→ Crossover operation

- Assuming two parent solutions are
  - Parent 1: \( S_1[^*, 1, ^*] \), \( ..., S_1[^*, m, ^*] \)
  - Parent 2: \( S_2[^*, 1, ^*] \), \( ..., S_2[^*, m, ^*] \)
- Randomly selecting a VC index \( P \)
- Partitioning both of two parent solutions into two portions at the \( P \) position
- Merging the head (or tail) portion of one parent solution with the tail (or head) portion of the other to generate two children solutions
  - Child 1: \( S_1[^*, 1, ^*] \), \( ..., S_1[^*, p-1, ^*] \), \( S_2[^*, p, ^*] \), \( ..., S_2[^*, m, ^*] \)
  - Child 2: \( S_2[^*, 1, ^*] \), \( ..., S_2[^*, p-1, ^*] \), \( S_1[^*, p, ^*] \), \( ..., S_1[^*, m, ^*] \)

→ Mutation Operation

- Adjusting the capacity allocation of one resource type
Fitness function

à For a certain amount of free capacity, it is desired that the free capacity of the same type of resource converges to as fewer number of physical nodes as possible

- Using the standard deviation of the free capacities of the same resource type in different physical nodes as the positive factor in the fitness function; i.e., the bigger deviation, the better

à In a physical node, it is desired that different type of resources have the similar level of free capacity

- Using the inverse of the standard deviation of the free capacities of different resource types in the same physical node as the weight

à The fitness function is

\[ \sum_{k=1}^{R} \sum_{i=1}^{N} \frac{(rc_{ik} - \overline{rc_k})^2}{W(\sigma_i, rc_i^k)} \]
Reconfiguration Plan

➔ After the calculation of the Genetic algorithm, the system needs to be reconfigured from the current state to the state obtained by the Genetic algorithm.

➔ The time spent in the transition can reflect the transition overhead.
  - Minimise the transition time.

➔ According to the difference between \([i, j, k]\) and \(S'[i, j, k]\), there may be the following operations on Virtual Machines:
  - VM deletion
  - VM creation
  - VM migration
  - Resource capacity adjustment.
The Transition Cost Model

- There might be dependencies among these VM operations
  - For example, the migration of VM\(_{13}\) from node 1 to node 2 requires node 2 releasing the resources first

- The dependencies can be modelled by a DAG graph

- The transition time is the total time of the VM operations on the critical path of the DAG
Bigtable: a distributed storage system in Google Cloud

→ Resembles a database

- Aims to scale to a very large size: petabyte of data across thousands of nodes

- Provides a simple data model that
  - Data is indexed using row and column names
  - Data is treated as strings
  - supports dynamic control over data layout and format
  - Allows clients to reason about the locality properties
Bigtable: a distributed storage system in Google Cloud

→ A row key in a table is an arbitrary string

→ Bigtable maintains data in lexicographic order by row key

→ Each read or write of data under a single row key is atomic

→ The row range for a table is dynamically partitioned

→ Each row range is called a tablet, which is the unit of distribution and load balancing

- Web pages in the same domain are grouped together into contiguous rows by reversing the hostname components of the URLs