Imperial College London

Introduction to Cloud Computing

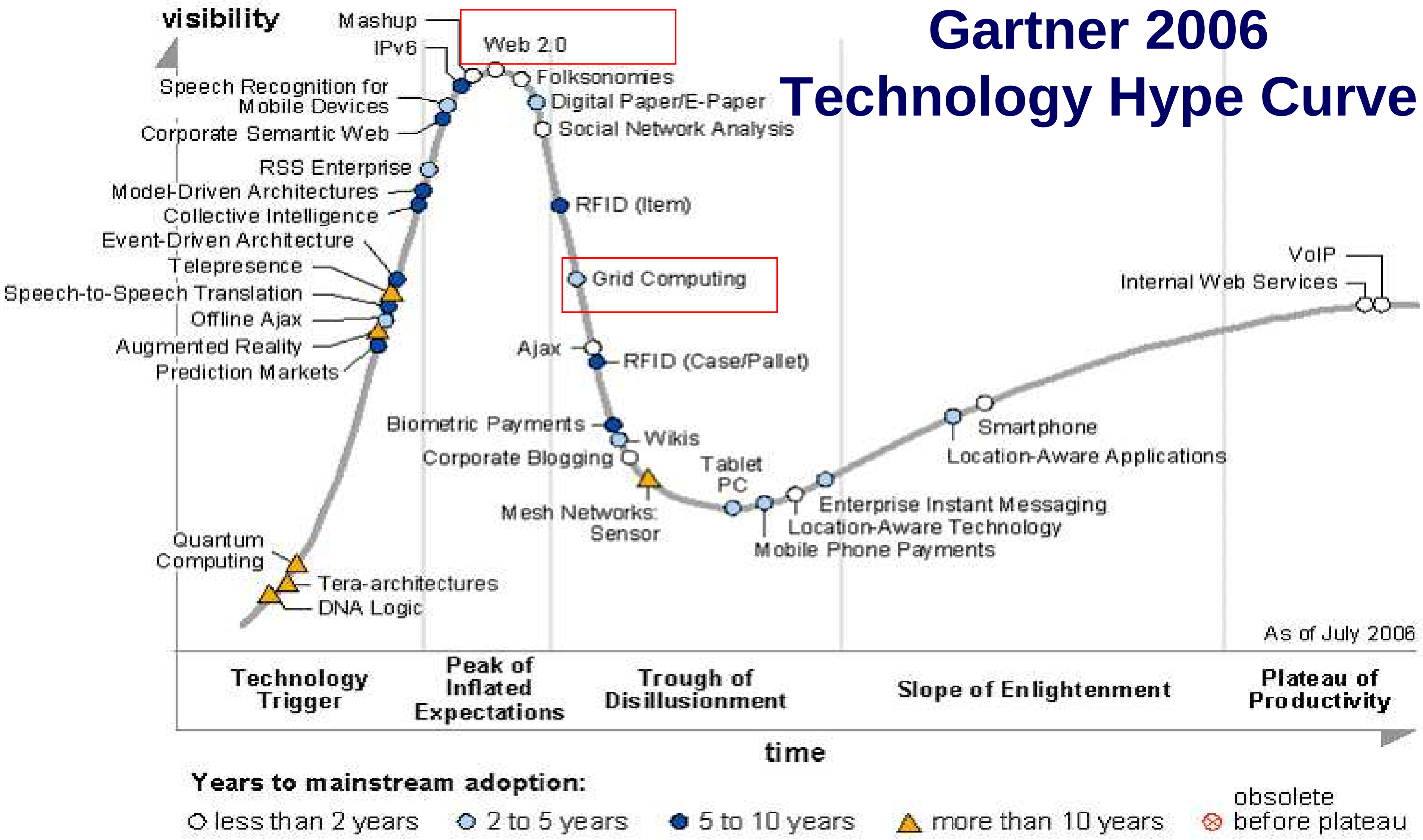
Prof. Yike Guo

Department of Computing
Imperial College London

Content

- Cloud Computing : An Introduction
- An Overview of Cloud Systems
- Programming within the Cloud
- Programming over the Cloud
- Building the Discovery Cloud
- Research Issues of Cloud Computing

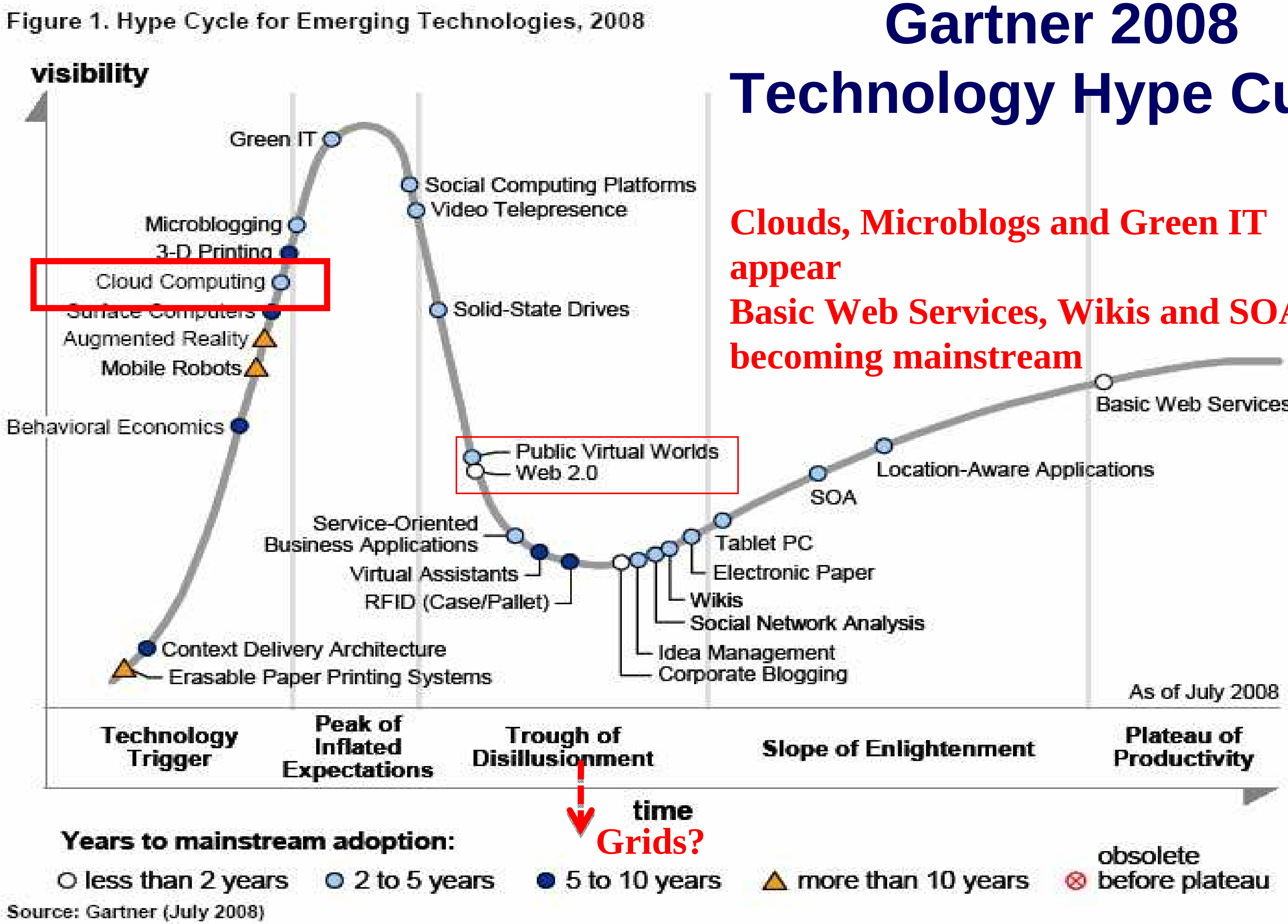
Gartner 2006 Technology Hype Curve



 © The Discovery Sciences Group

Imperial College London

Gartner 2008 Technology Hype Curve



Clouds, Microblogs and Green IT appear

Basic Web Services, Wikis and SOA becoming mainstream

Grids?

Is this Cloud?

- **Cloud computing** is a style of computing in which dynamically scalable and often virtualized resources are provided as a service over the Internet. Users need not have knowledge of, expertise in, or control over the technology infrastructure in the "cloud" that supports them (Wikipedia)

Cloud Computing: My Definition

Cloud computing is a resource delivery and usage model, it means to get resource (Hardware, Software, Applications) via network "on-demand" and "at scale" **as services** in a multi-tenant environment.. The network of providing resource is called 'Cloud'. All resource in the 'Cloud' are scalable infinitely and used whenever as utility

Software as a Service

e.g. Saleforce.com

(Programming over the Cloud)

Platform as a Service

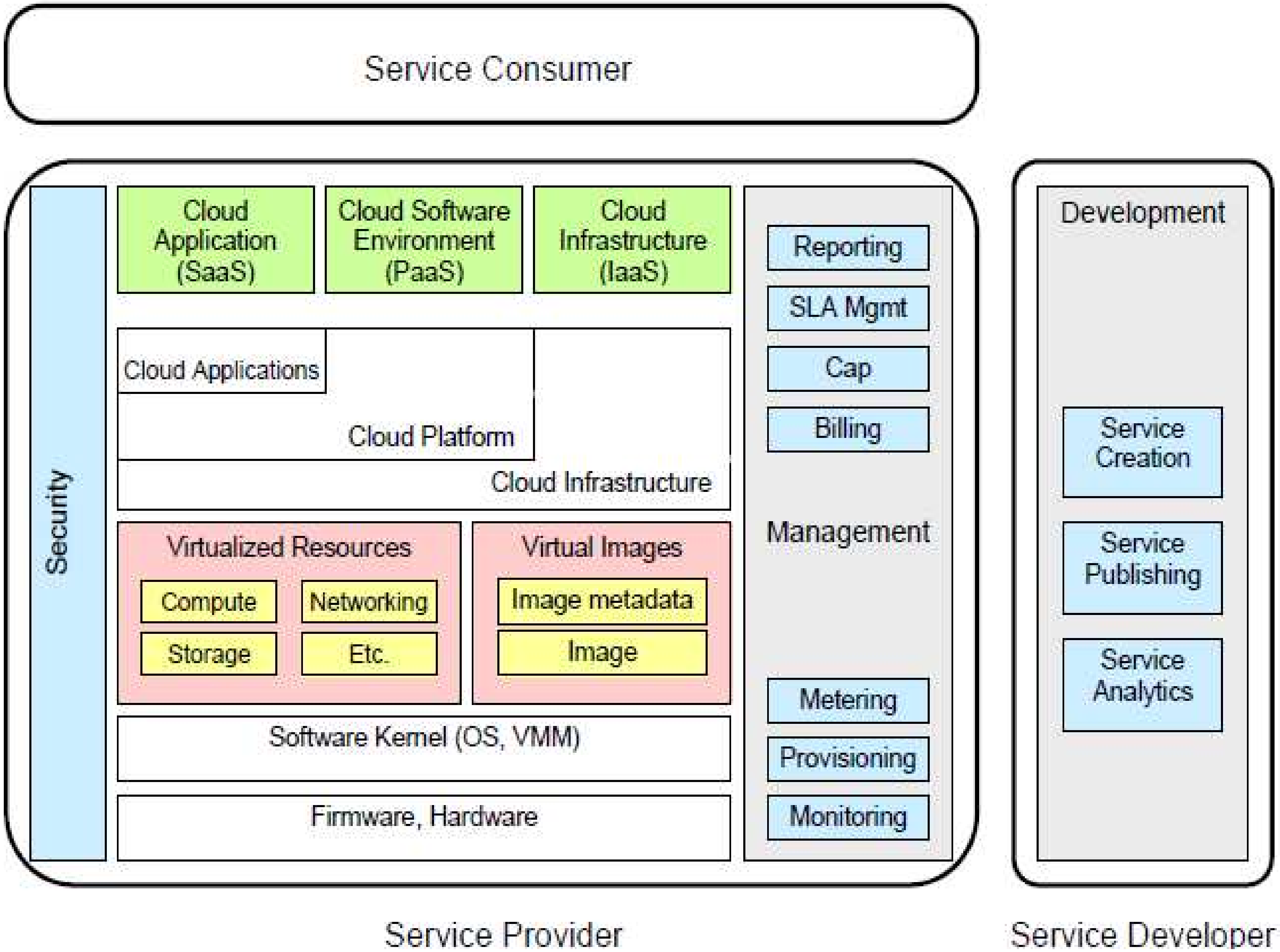
e.g. Google App Engine

(Programming within the Cloud)

Infrastructure as a Service

e.g. Amazon EC2

A Taxonomy for Cloud Computing



Imperial College London

What is Cloud Computing

- A user's view: "No more servers, no more IT managers, no more licensing. All I need is a browser. I will find the service I need and pay it when I use it based how much I used it"--- **Software as a Service**

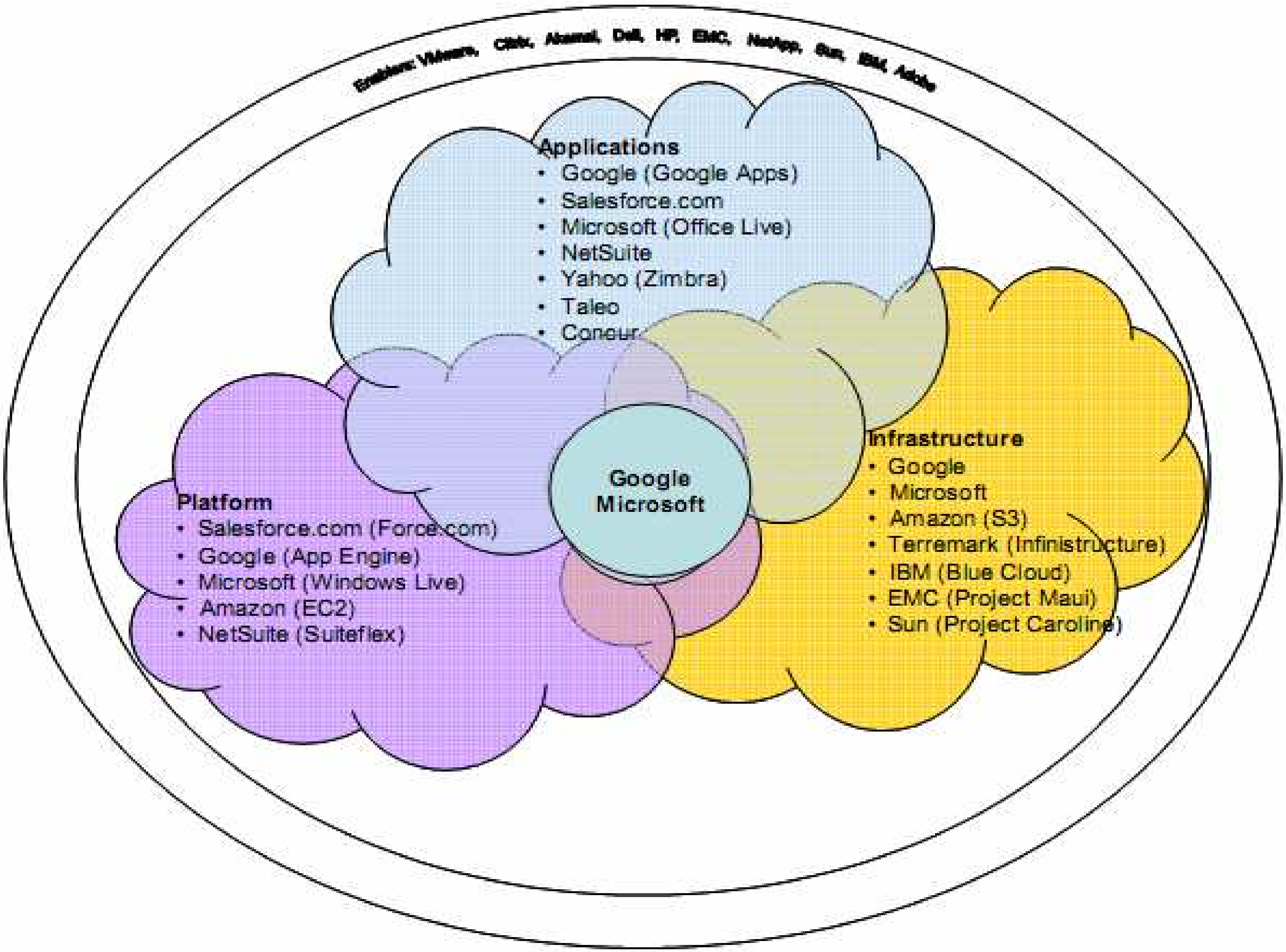
- A developer's view: "Programming an application will not be a 30 man/month job with a stupid and horrible project manager. It will be done by a couple of friends, in a long hard working weekend with PHP on a machine God knows where it is''--- **Platform as a Service**

- A computer scientist's view: " We are not programming a single machine, rather the World Wide Computer''--- **Infrastructure as a Service**

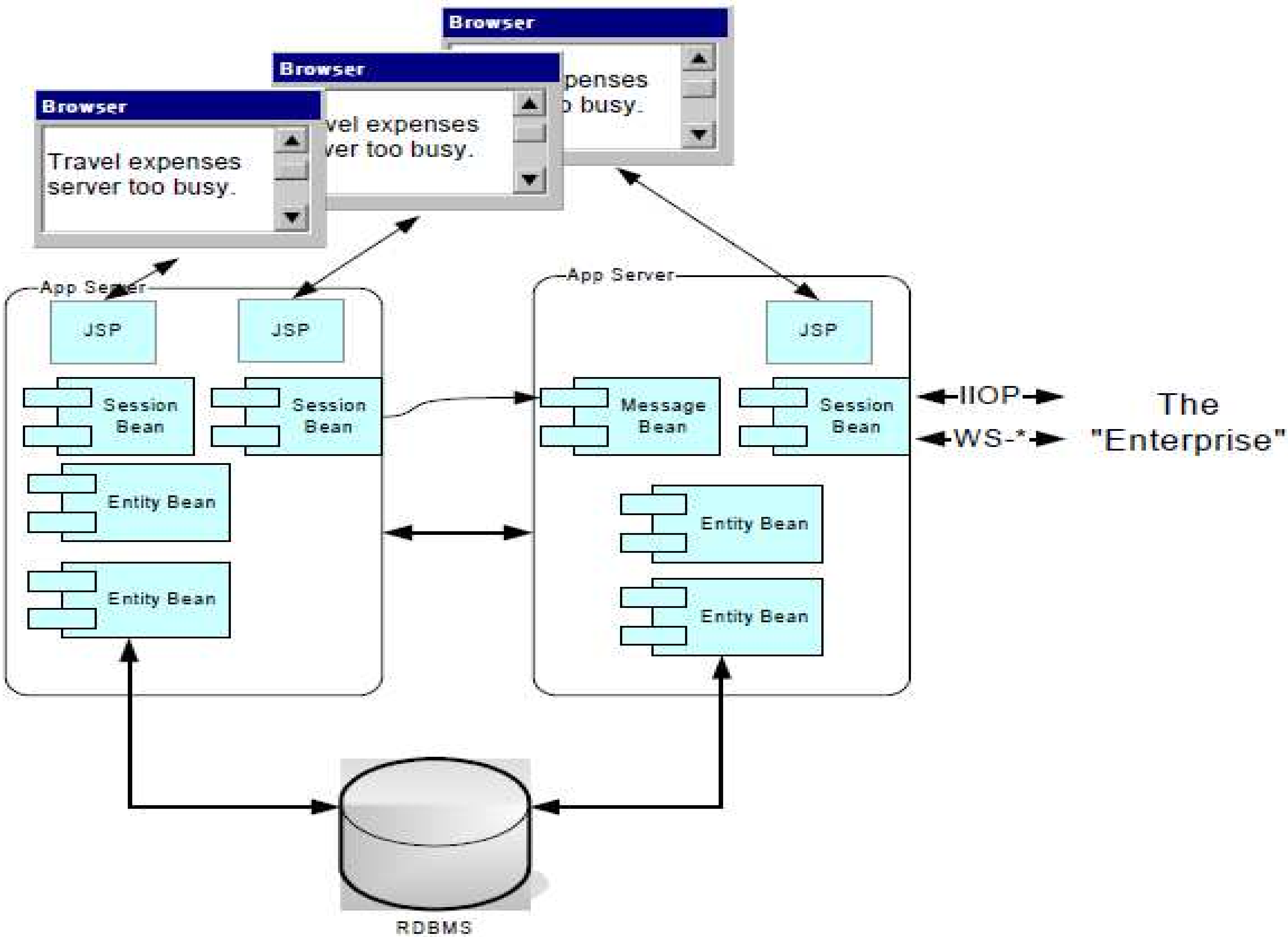
- An economist's view: "The most interesting thing about the cloud computing is not the technology, but the new evolving social standards and business models, and the ramifications of egalitarianism on a global scale'' --- **Everything as a Service**

Imperial College London

Cloud Players Today



- An Enterprise C/S System is Not a cloud !



What is NOT a Cloud?

- A Parallel Computing System (Server Clusters) is NOT a Cloud

Grid Computing is, unfortunately, NOT Cloud Computing

- What does the “Grid” really mean?
  - Gird means the utility : Grid Computing aims to realize cloud computing
  - Grid was interpreted as the network: Grid computing becomes an architecture for “wide-area” clustering computing

- Grid computing focuses on scheduling
  - Cloud
    - Full private (logical) cluster is provisioned
    - Individual user can only get a tiny fraction of the total physical resource pool
    - No support for cloud federation except through the client interface
  - Grid
    - Built so that individual users can get most, if not all of the resources in a single request
    - Middleware approach takes federation as a first principle
    - Resources are exposed, often as bare metal

- Grid computer offers less abstraction on resource
  - Grids require applications to conform to the grid software interfaces
  - An important missing concept in Grid computing is “Virtualization”

- Cloud Computing focuses on the realization of computational utility
  - Virtualization offers the key abstraction of hardware resource: Scheduling over the VM rather than physical resources
  - Scheduling on the VM level : Resource can be dynamically shaped or carved out from its underlying hardware infrastructure and made available to a workload---
  - Utility hierarchy: full exploration of service deployment and delivery for “on demand” and “at scale” use , for hardware, system, development environment and applications

- However:
  - Grid computing sets the technical foundation for clouding computing
  - Cloud computing realizes the goal of Grid computing

A Comparison Between Grid Computing and Cloud Computing

	EGEE Grid	Amazon Cloud
Target Group	Scientific community	Business
Service	short-lived batch-style processing (job execution)	long-lived services based on hardware virtualization
SLA	Local (between the EGEE project and the resource providers)	Global (between Amazon and users)
User Interface	High-level interfaces	HTTP(S), REST, SOAP, Java API, BitTorrent
Resource-side middleware	Open Source (Apache 2.0)	Proprietary
Ease of Use	Heavy	Light
Ease of Deployment	Heavy	Unknown
Resource Management	probably similar	
Funding Model	Publicly funded	Commercial

Summary of „An EGEE Comparative Study: Grids and Clouds Evolution or Revolution" by Markus Klems

Cloud computing infrastructure provider

- Agathon Group - Cloud provider. Services include highly available VPS, virtual private datacenters and ready-to-use LAMP stacks. Self-service ordering. Custom development and managed services available.
- Amazon Web Services - Amazon EC2/S3 (Hardware-a-a-S & Cloud Storage)
- CohesiveFT - CohesiveFT Elastic Server On-Demand
- ElasticHosts - UK-based instant, on-demand servers in the cloud
- Flexiscale - Another instant provisioner of web servers with some advanced features like auto-scaling coming soon.
- GoGrid - instant, on-demand servers offering "control in the cloud". Deploy Windows/Linux servers via web-interface in minutes
- GridLayer - Cloud Provider. A service by Layered Technologies that delivers Virtual Private Datacenters and virtual private servers from grids of commodity servers
- LayeredTechnologies - Cloud Provider. provider of on-demand hosting and cloud and utility computing solutions through its brand GridLayer
- Mosso - Rackspace's cloud hosting service
- Newservers - Instant provisioning of web servers either Windows or Linux

Cloud computing Paas provider

- Bungee Connect - Provides end to end tools and systems required to develop, deploy and host web applications (Platform as a Service)
- Coherence - Oracle Coherence Data Grid for EC2 and other cloud platforms
- Force.com - Salesforce.com's application development platform (PaaS)
- GigaSpaces - middleware for the cloud, "cloudware"
- Google AppEngine - (PaaS)Now support python
- Heroku - Ruby on Rails in their Cloud
- Qrimp - An AJAX based PaaS
- RightScale - RightScale provides a platform and expertise that enable companies to create scalable web applications running on Amazon's Web Services that are reliable, easy to manage, and cost less

- Amazon EC2
- Microsoft Aruze
- Apache
- Eucalyptus Platform

Amazon Web Services for Cloud Competing

- Amazon Elastic Compute Cloud (EC2)
- Amazon Simple Storage Service (S3)
- Amazon Simple Queue Service (SQS)
- Amazon SimpleDB (SDB)
- Amazon Flexible Payment Service
- Amazon DevPay

- Flat object storage model with key
- Bucket as object container (100 buckets per account)
- Standards-based Interfaces: REST and SOAP with URL for each object

- Object-Based Storage
- 1 B – 5 GB / object
- Fast, Reliable, Scalable
- Redundant, Dispersed
- 99.99% Availability Goal
- Private or Public
- Per-object URLs & ACLs
- US & European Locations

$.15 per GB per month storage

$.01 for 1000 to 10000 requests

$.10 - $.18 per GB data transfer

Establish connection

```
require 'S3'

AWS_ACCESS_KEY = '<your key>'
AWS_SECRET_ACCESS_KEY = '<your key>'

conn = S3::AWSAuthConnection.new
(AWS_ACCESS_KEY_ID,
AWS_SECRET_ACCESS_KEY,
false)
```

Create bucket

```
BUCKET_NAME = 'assets.example.com'
conn.create_bucket(BUCKET_NAME)
```

Upload file

```
datafile = File.open(path)
Key = path.basename;

conn.put(BUCKET_NAME, key, datafile.read,
{"Content-Type" => mime,
 "Content-Length" => File.size(path).to_s,
 "x-amz-acl" => "public-read"})

http://assets.sample.com/...
```

Amazon Elastic Compute Cloud (EC2)

Managing scalable virtual private server instances

- Virtual Compute Cloud
- Root-level System Access
- Elastic Capacity
- Management API
- Scale in Minutes
- Multiple Instance Sizes
- Network Security Model

$.10-$.80 per server hour

$.10 - $.18 per GB data transfer

S	L	XL
1.7 GB RAM 1 EC2 Compute Unit 160 GB 32-bit platform	7.5 GB RAM 4 EC2 Compute Units 850 GB 64-bit platform	15 GB RAM 8 EC2 Compute Units 1690 GB storage, 64-bit platform
$0.10/Hr	$0.40/Hr	$0.80/Hr

EC2 Basic Concepts

- Resource delivered as AMI (Amazon Machine Image)
  - Bootable root disk stored in S3
  - AMI are pre-defined or user-built
  - OS support: Fedora, Centos, Gentoo, Debian, Ubentu, Windows Server
- Compute instance
  - Running copy of an AMI
  - Instance invocation is programmable in "event driven" style
- Network Security
  - Explicit access control (code in invocation script)
  - Can establish security group

Images:
- RegisterImage
- DescribeImages
- DeregisterImage

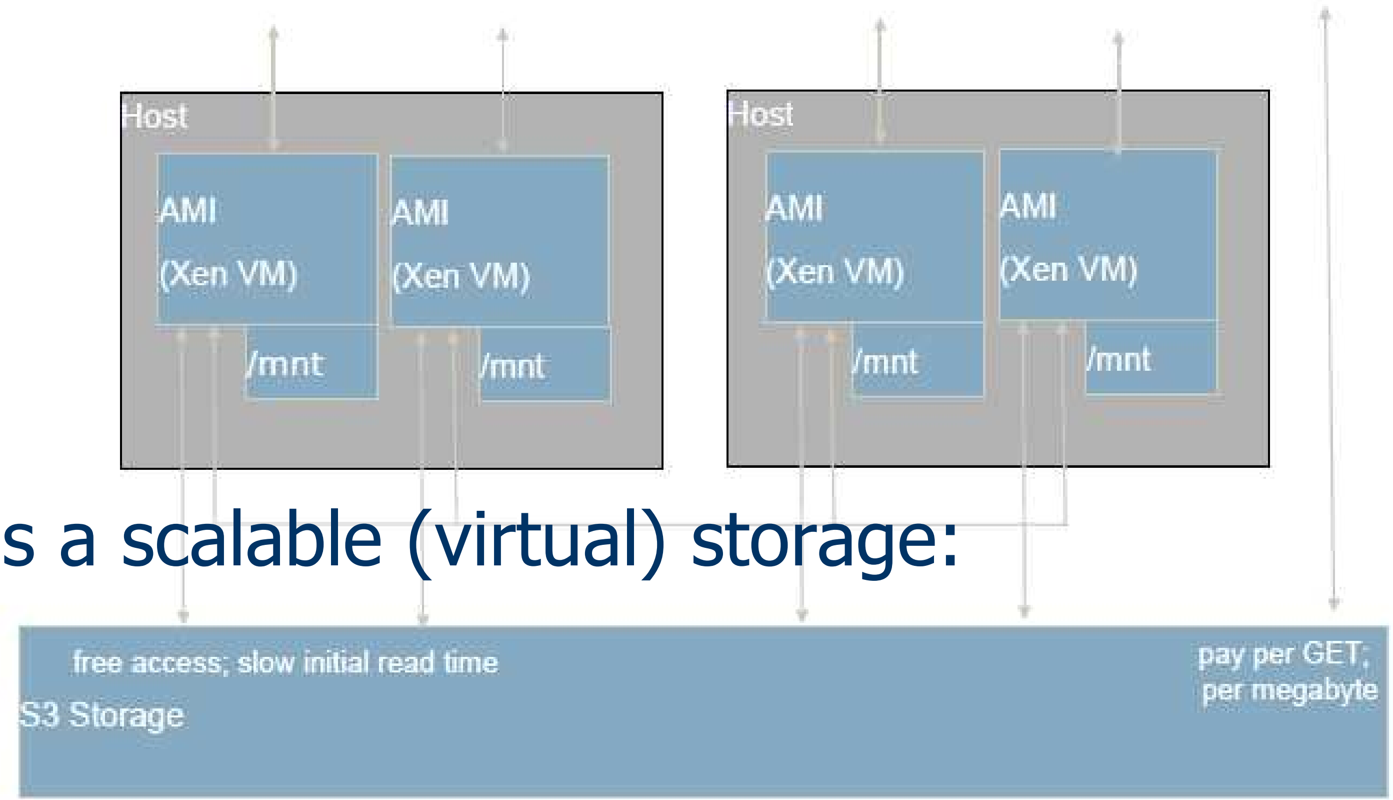
Instances:
- RunInstances
- DescribeInstances
- TerminateInstances
- GetConsoleOutput
- RebootInstances

Keypairs:
- CreateKeyPair
- DescribeKeyPairs
- DeleteKeyPair

Image Attributes:
- ModifyImageAttribute
- DescribeImageAttribute
- ResetImageAttribute

Security Groups:
- CreateSecurityGroup
- DescribeSecurityGroups
- DeleteSecurityGroup
- AuthorizeSecurityGroupIngress
- RevokeSecurityGroupIngress

- EC2 as a scaleable (virtual) cluster:
- S3 as a scalable (virtual) storage:



Imperial College London

Services

- Scalable Queuing
- Elastic Capacity
- Reliable, Simple, Secure

Inter-process messaging, data buffering, scalable architecture component

$.01 per 10000 messages

$.10 - $.18 per GB data transfer

APIs

- **Queues:**
  - ListQueues
  - DeleteQueue
- **Messages:**
  - SendMessage
  - ReceiveMessage
  - DeleteMessage

Imperial College London

- Services

- Distributed Data Store
- Structured Storage
- Fully Indexed
- Redundant
- Scalable
- Elastic Capacity
- Query Language

$1.50 per GB per month

$.10 - $.18 per GB data transfer

$.14 per CPU hour (query processing)

- APIs

- **Domains:**
  - CreateDomain
  - ListDomains
  - DeleteDomain
- **Items:**
  - PutAttributes
  - GetAttributes
  - Query

- **Query Language (samples):**
  - ['Title' = 'The Right Stuff']
  - ['Number of Pages' < '00310']
  - ['Rating' = '***' or 'Rating' = '*****']
  - ['Year' > '1950' and 'Year' < '1960' or 'Year' starts-with '193' or 'Year' = '2007']
  - Keyword' = 'Frank Miller'] union ['Rating' starts-with '****']

- Domain:
  - collection of items (tables)
  - 100 domains per account
- Item:
  - Collection of key-value pairs (table)
  - Up to 256 attributes per item
  - Up to 1024 bytes per value
- Querying Language:
  - Set based querying language
  - A (much) simplified SQL

Pay for Using

- Standard licensing terms
- Commercially usable
- Monthly credit card billing
- Self-serve model:
  - Sign up as developer
  - Choose services
  - Agree to service licenses
  - Enter payment info
  - Start coding

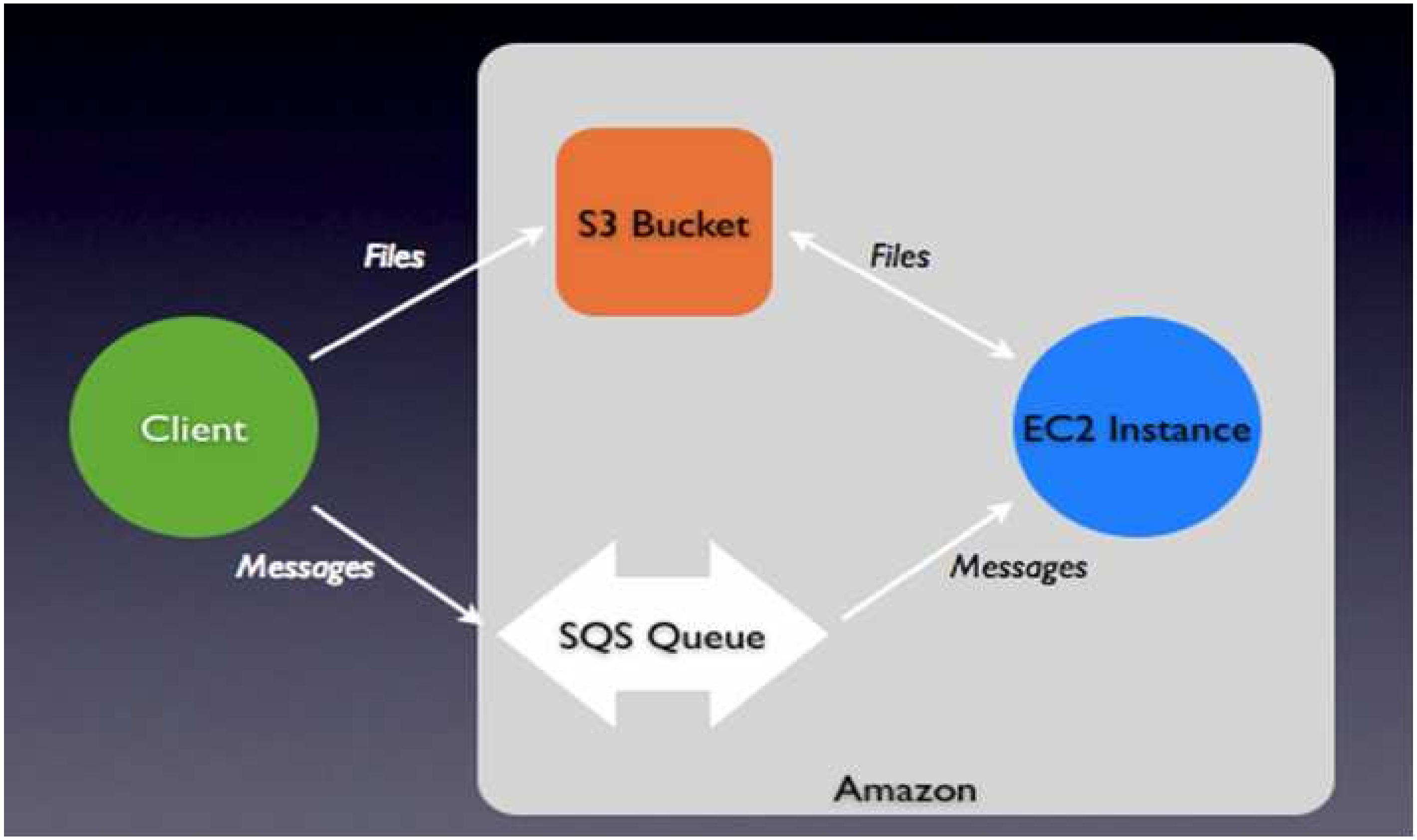
- Flexible Payments Service:
  - Move money between any two people or systems.
  - Credit cards, bank accounts, Amazon Payments.
  - One-time, multiple, or recurring transactions.
  - Payment processing language (Gatekeeper).
  - Aggregated transactions (micropayments).
- DevPay:
  - Wrap custom business models around S3 and EC2.
  - Set custom prices for each charging unit.
  - Charge for your applications.

A Simple Example: 2 hours Student Project (1)

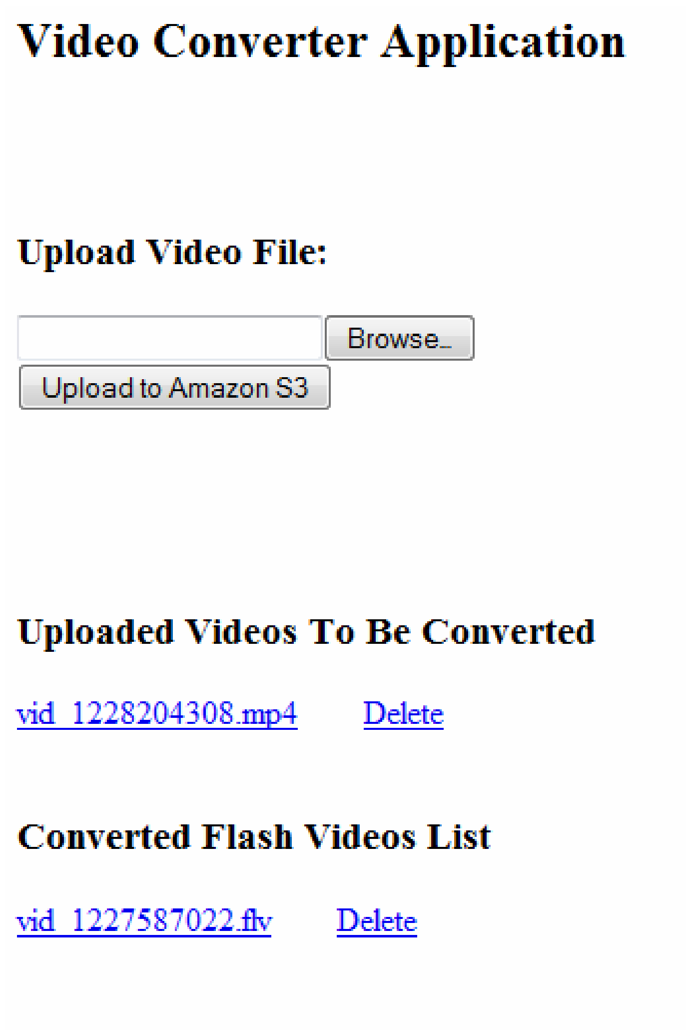
- Use S3, EC2, SQS to create a scalable video converter web application
  - Use computing power of EC2 virtual server instances for video conversion
    - Ubuntu image, installed/configured FFmpeg
  - S3 for media storage
  - SQS for messaging between client and EC2
- Ex: converting for iPod; Youtube

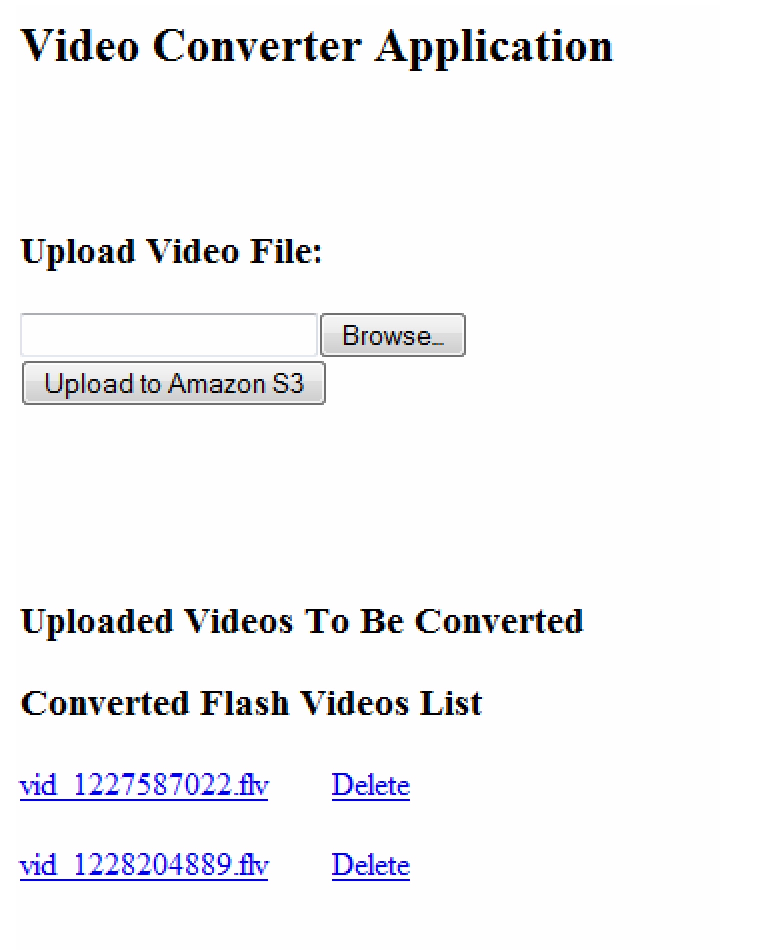
A Simple Example (2)

Video converter architecture

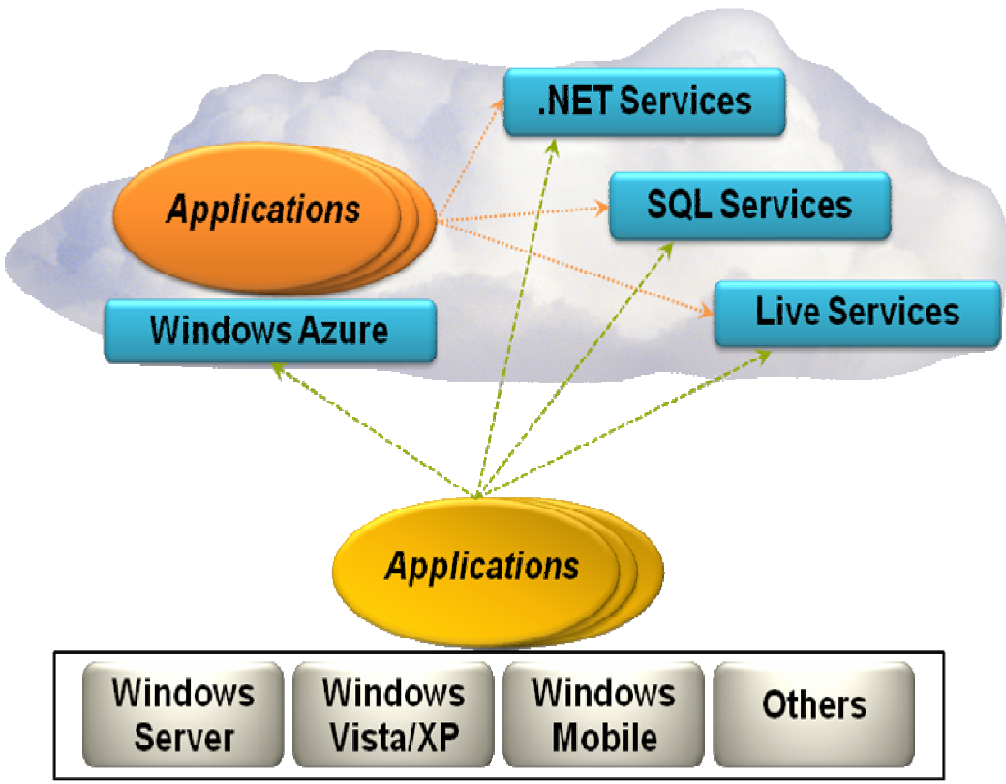


A Simple Example (3)





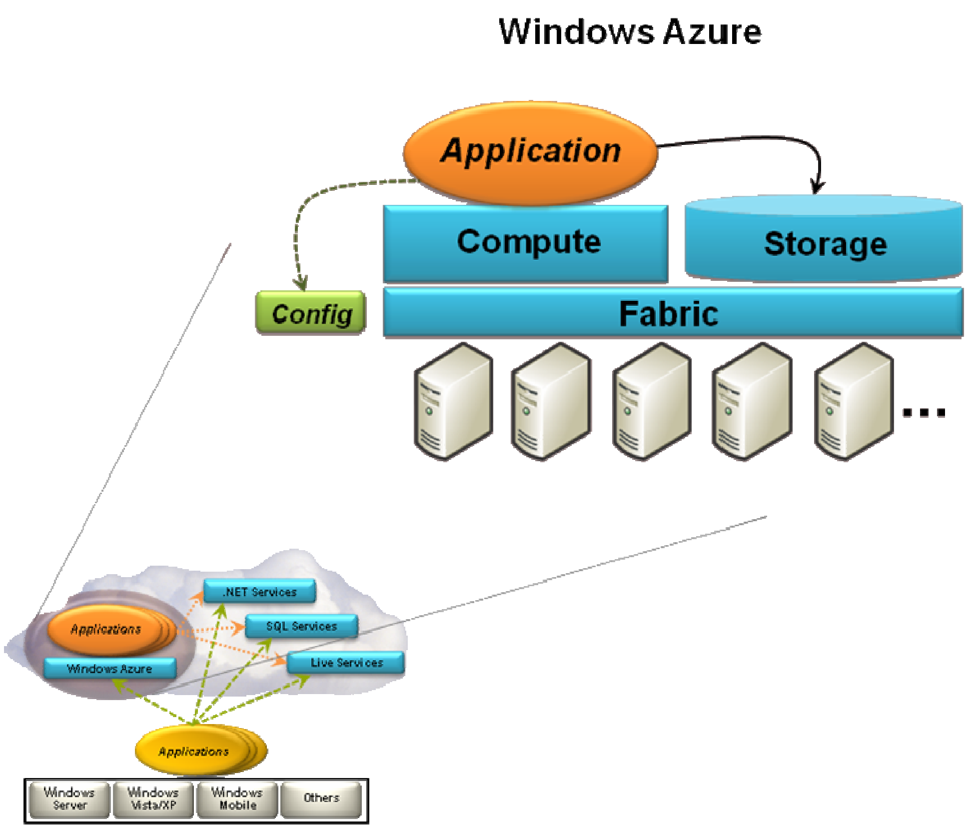
Azure: A Platform for Microsoft Cloud



- *Windows Azure*: Provides a Windows-based environment for running applications and storing data on servers in Microsoft data centers.
- *Microsoft .NET Services*: Offers distributed infrastructure services to cloud-based and local applications.
- *Microsoft SQL Services*: Provides data services in the cloud based on SQL Server.
- *Live Services*: Through the Live Framework, provides access to data from Microsoft's Live applications and others.

  The Live Framework also allows synchronizing this data across desktops and devices, finding and downloading applications, and more.

Window Azure: Supporting Windows-based Compute and Storage Services for Cloud Applications

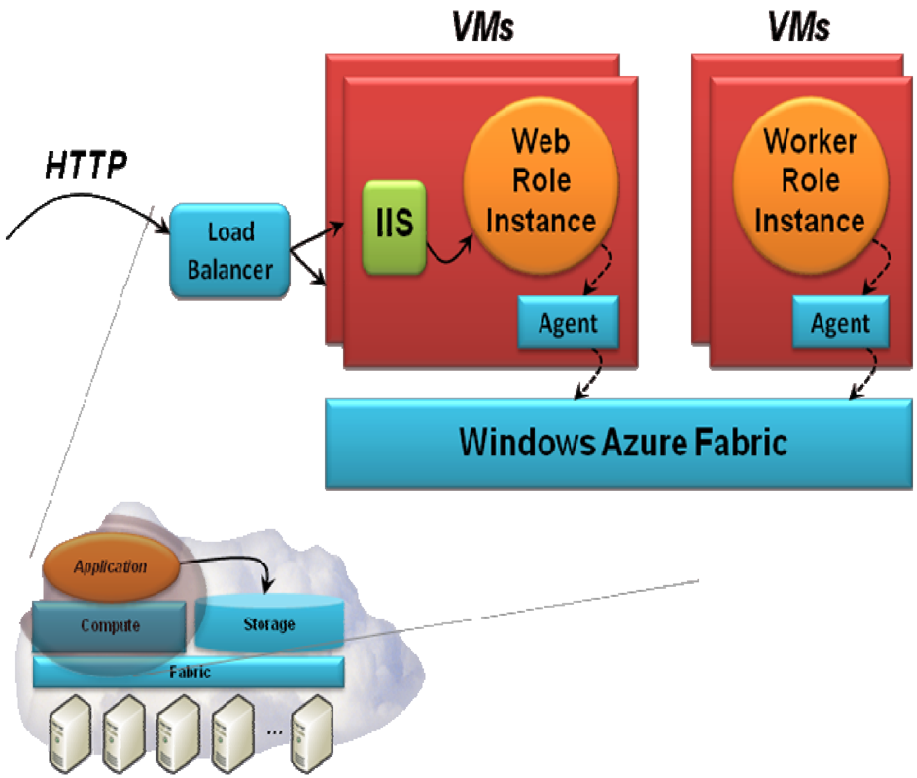


•Windows Azure runs on a large number of machines, all located in Microsoft data centers and accessible via the Internet. A common Windows Azure fabric aggregates this processing power into a unified whole. Windows Azure compute and storage services are built on top of this fabric.

• The Windows Azure compute service is based on Windows. Now ,Windows Azure runs only applications built on the .NET Framework. Non .NET application will be supported in 2009.

•Non SQL storage service provides simpler, more scalable kinds of storage. It allows storing binary large objects (blobs), provides queues for communication between components of Windows Azure applications, and offers a form of tables with a straightforward query language (LINQ Style).

Azure Computation Mechanism



- Each application *instances* runs a copy of all or part of the application's code. Each of these instances runs in its own virtual machine (VM). These VMs run 64-bit Windows Server 2008 with hypervisors.

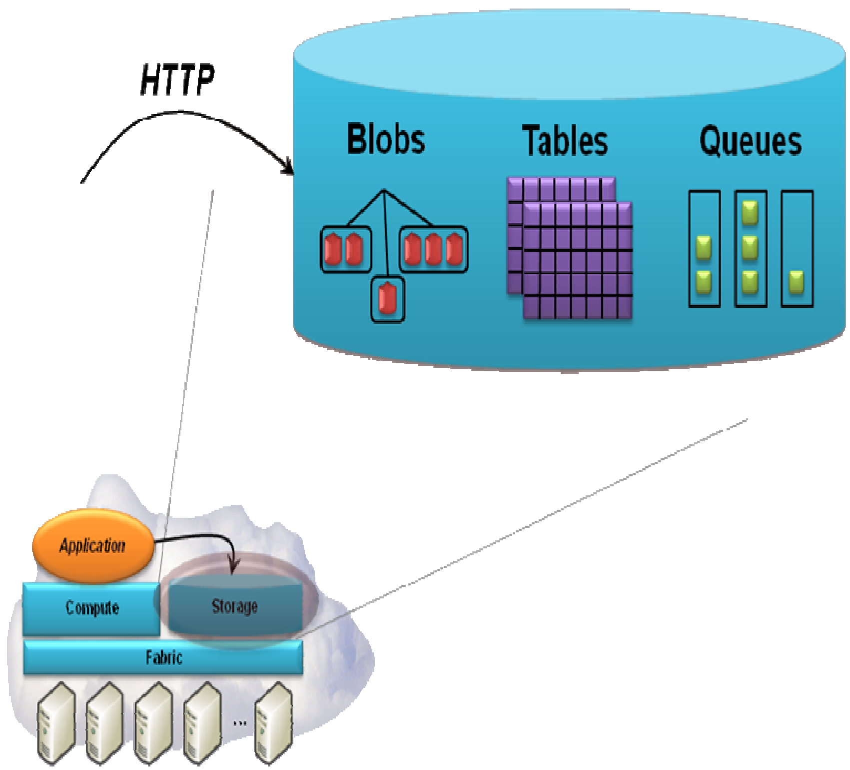
- .NET 3.5 application instances can be created as *Web role* instances and/or *Worker role* instances.

- Web role instance accepts incoming HTTP (or HTTPS) requests. A Web role can be implemented using ASP.NET, WCF, or another .NET Framework technology that works with IIS. It is created to handle an incoming HTTP request and shut down when that request has been processed (short life/event driven)

- A Worker role instance gets its input only from a Web role instance, via a queue in Windows Azure storage. The result of its work can be written to Windows Azure storage. A worker role instance can run indefinitely (long life/ batch job)

Imperial College London

Azure Storage Mechanism



• Designed for efficient data management within MS data center. A storage account can have one or more *containers*, each of which holds one or more blobs. Blobs can be up to 50 gigabytes each

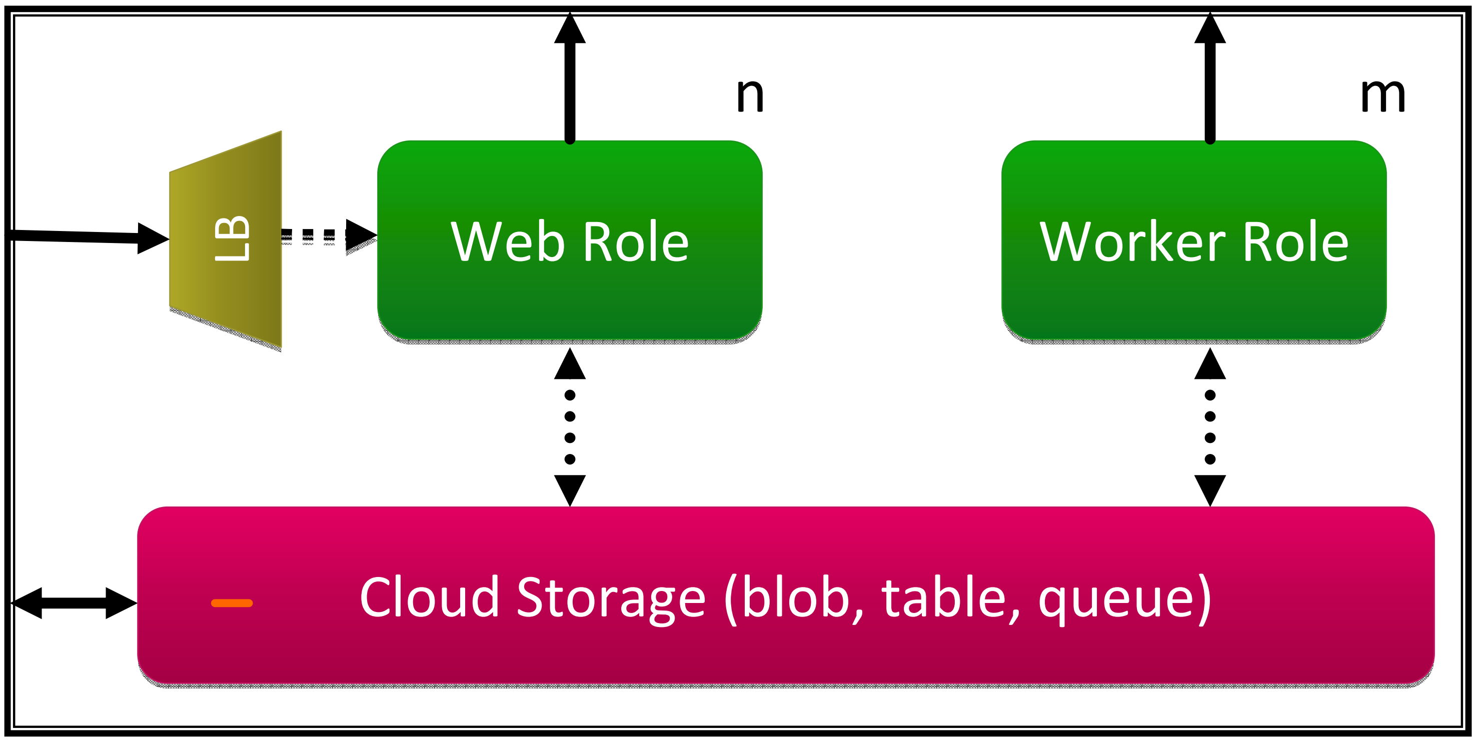
•Windows Azure storage provides tables which is actually stored in a simple hierarchy (entry-property-(name, type value))of entities with properties (rather than with a RDBS). A table has no defined schema; instead, properties can have various types. And rather than using SQL, an application accesses a table's data using a query language with LINQ syntax. A single table can be quite large can partitioned across many servers

•The primary role of queues is to provide a way for Web role instances to communicate with Worker role instances

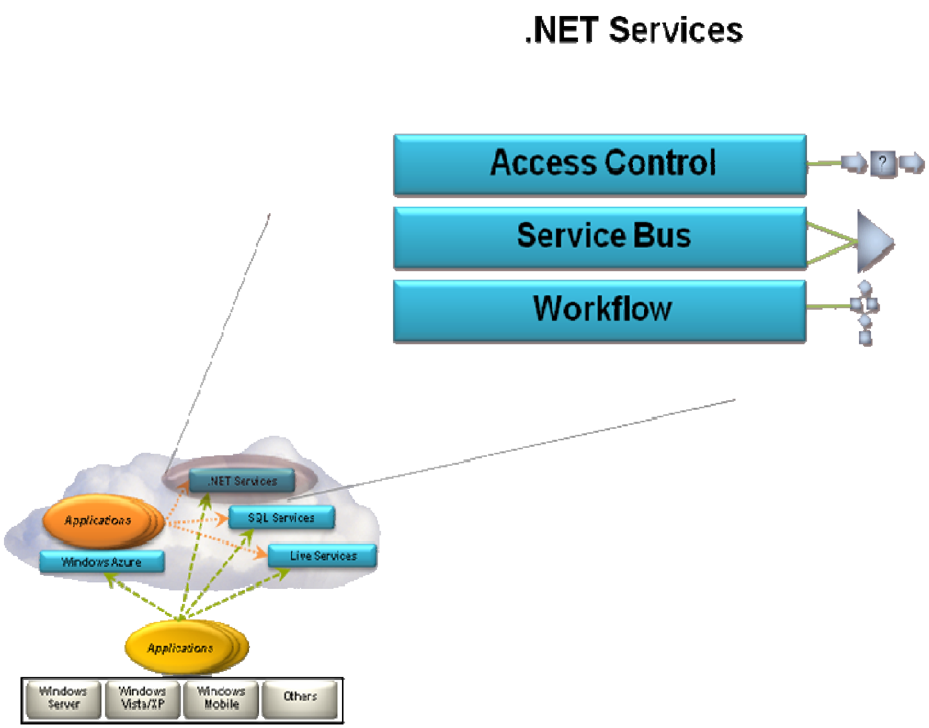
•All Windows Azure storage styles use the conventions of REST to identify and expose data. Everything is named using URIs and accessed with standard HTTP operations

Imperial College London

A Simple Operational Model



.NET Services: Providing Cloud-based Infrastructure for Both Cloud and On-premises Applications.



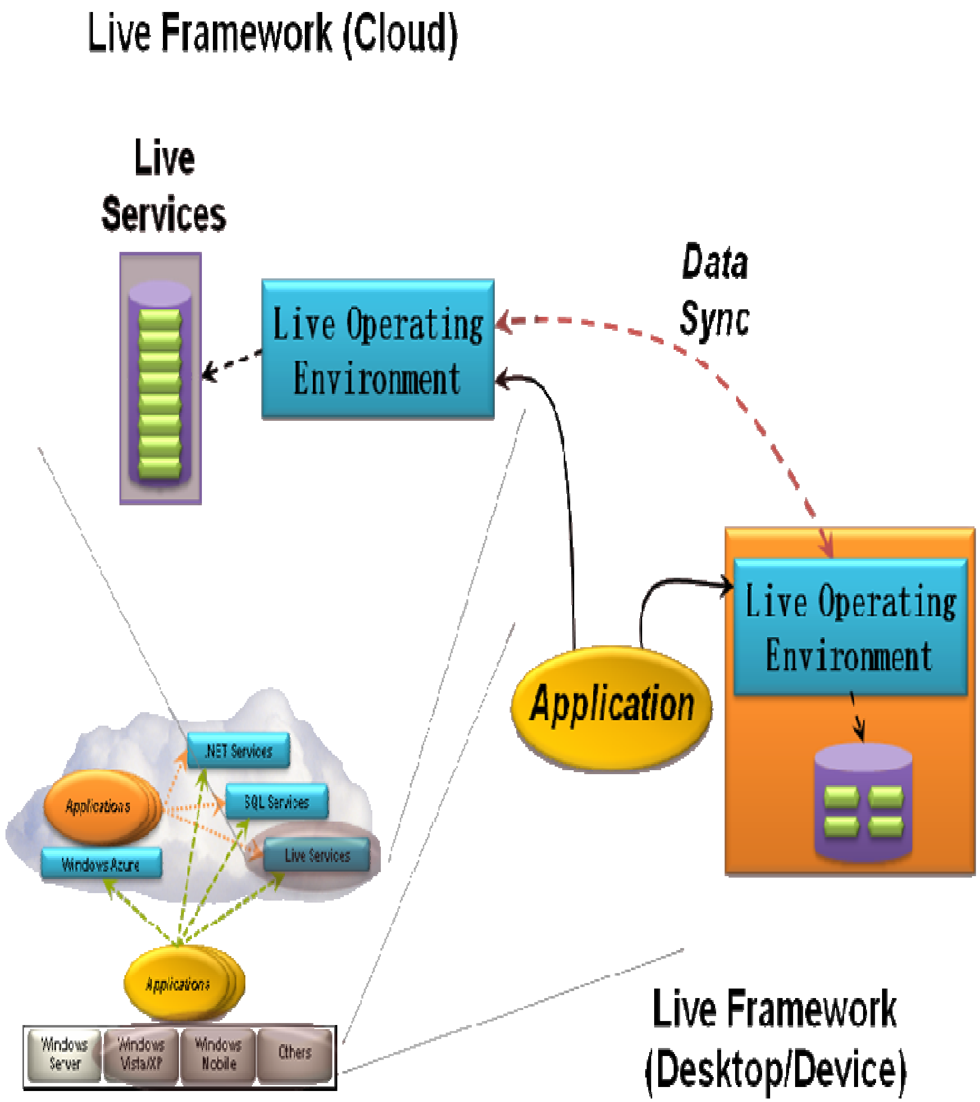
- Three groups of .Net services are provides to form cloud infrastructure services:

- .NET Access Control Service: using Claims-based identity method with STS. To support cloud use, *identity federation is used to* let claims created in one identity scope be accepted in another. *claims transformation* is supported to modify claims when they're passed between identity scopes.

- NET Service Bus: letting an application expose Web services endpoints that can be accessed by other applications. Each exposed endpoint is assigned a URI. Service Bus also deal with network address translation and tunneling .

- .NET Workflow Service: Creating composite applications based a business logic. Built on Windows Workflow Foundation (WF), the Workflow service allows running this kind of logic in the cloud

Live Framework and Live Services: Window's Platform as a Services Infrastructure



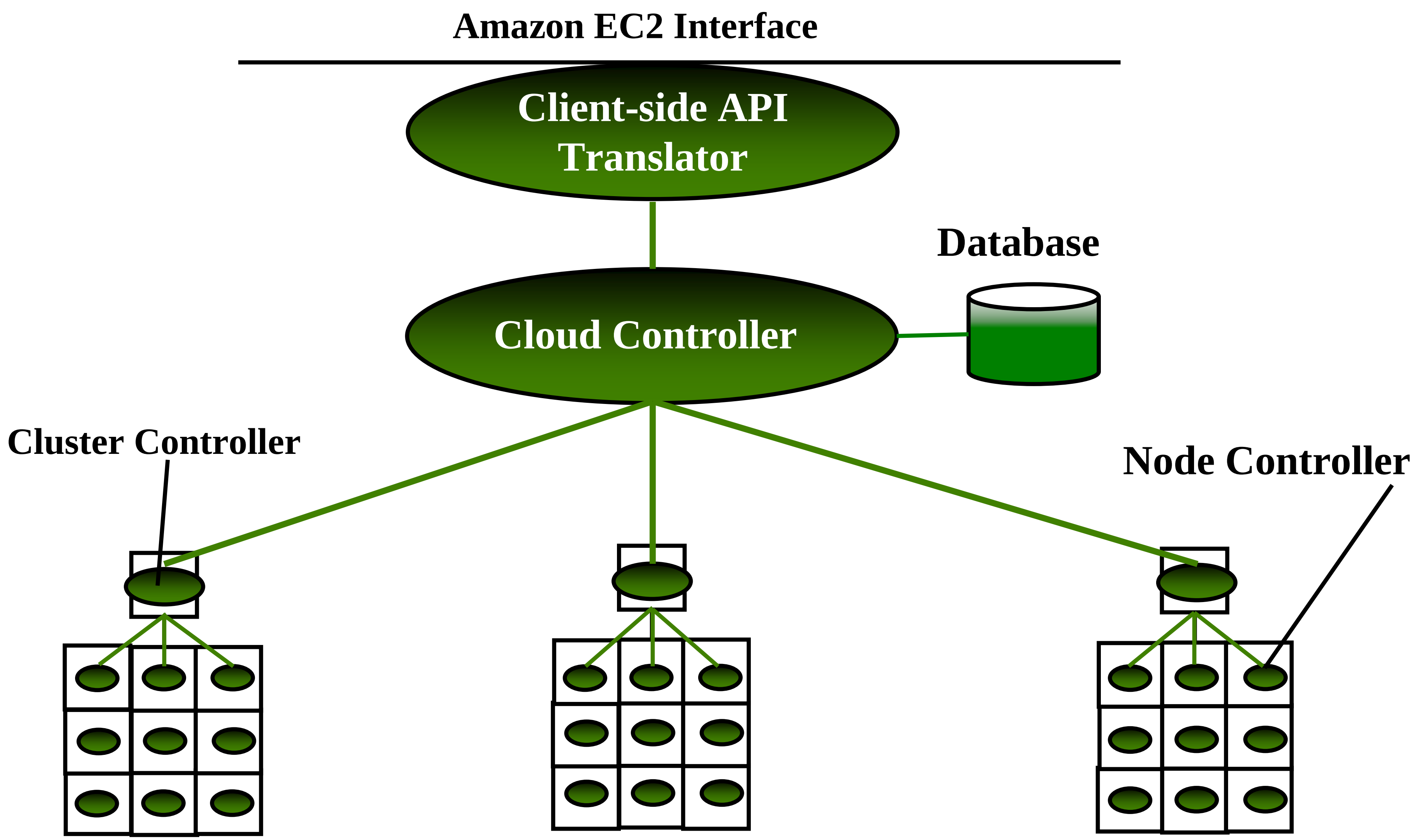
- Great advantage of MS : a large amount of personal/social information available
- MS has wrapped this diverse set of resources (data/application functions) into Live Services. Existing Microsoft applications, such as the Windows Live family, rely on Live Services to store and manage their information
- Live Operating Environment: running both in the cloud, and applications use it to access Live Services data via HTTP using the .NET Framework, JavaScript, Java, or any other language. Information in Live Services can also be accessed as an Atom or RSS feed, letting an application learn about changes to this data. And to set up and manage the Live Services application needs, a developer can use the browser-based Live Services Developer Portal.
- Any application, whether it's running on Windows or some other operating system, can access Live Services data in the cloud via the Live Operating Environment.

- Web services based implementation of cloud computing infrastructure
  - Linux image hosting ala Amazon
- Amazon compatible
  - Try and emulate Amazon cloud: EC2 + S3
  - Works with command-line tools from Amazon w/o modification
  - Enables leverage of emerging EC2 value-added service venues (e.g. Rightscale)
- Functions as a software overlay
  - Existing installation should not be violated
- Simple installation using Rocks

Features

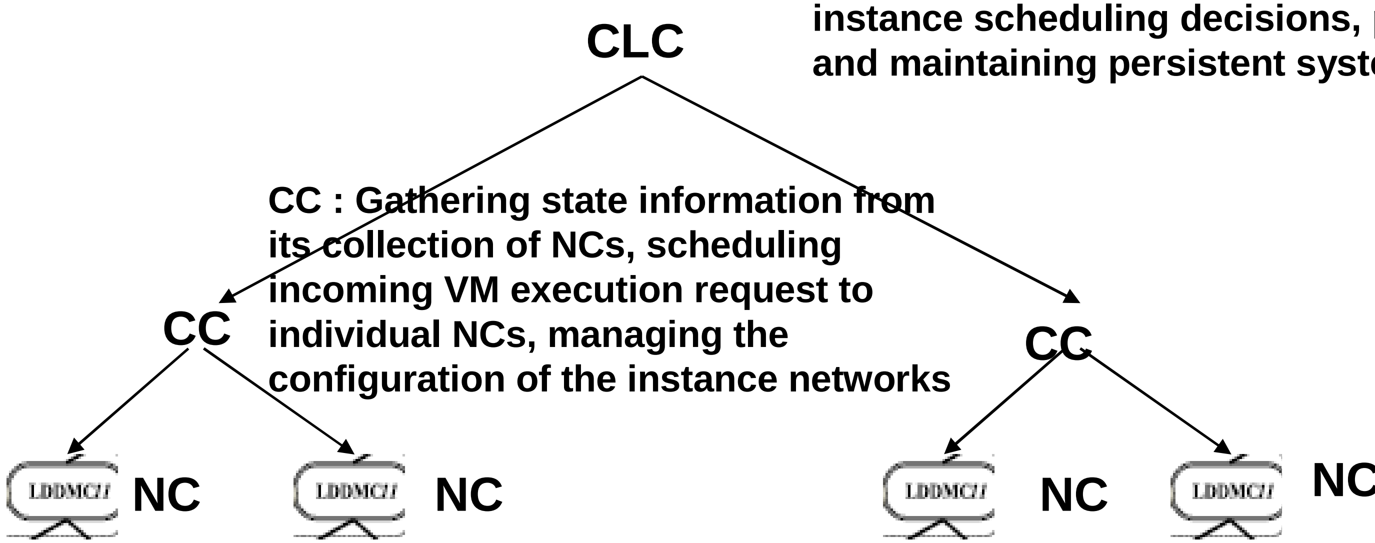
- Extensibility: Simple architecture and open internal APIs
- Client-side interface: Amazon’s EC2 interface and functionality (familiar and testable)
- Networking
  - Virtual private network per cloud based on VDE
  - “Native Network” within physical clusters for performance
- Security: using industry-standard web technologies
- Simple packaging, installation, maintenance

Eucalyptus Architecture: WS-Cloud



Functions of the Nodes

CLC : process user requests, making high level VM instance scheduling decisions, processing SLAs and maintaining persistent system



CC : Gathering state information from its collection of NCs, scheduling incoming VM execution request to individual NCs, managing the configuration of the instance networks

NC : the component that executes on the physical resource that host VM instances and is responsible for instance start up, inspection, shutdown….

Imperial College London

EC2 Compatibility

- Interface is based on Amazon's published WSDL
  - 2008 compliant except for
    - static IP address assignment
    - Security groups
  - Uses the EC2 command-line tools downloaded from Amazon
  - REST interface
- S3 support/emulation
  - Images accessed by file system name instead of S3 handle for the moment
- System administration is different
  - Eucalyptus defines its own Cloud Admin. tool set for user accounting and cloud management

Networking

- Interesting design goal : Connectivity, isolation and performance

- Eucalyptus does not assume that all worker nodes will have publicly routable IP addresses
  - Each cloud allocation will have one or more public IP addresses
  - All cloud images have access to a private network interface (connectivity)

- Two types of networks internal to a cloud allocation
  - Virtual private network
    - Uses VDE interfaced to Xen that is set up dynamically
    - Substantial performance hit within a cluster
    - Allows a cloud allocation to span clusters
    - Key to achieve isolation

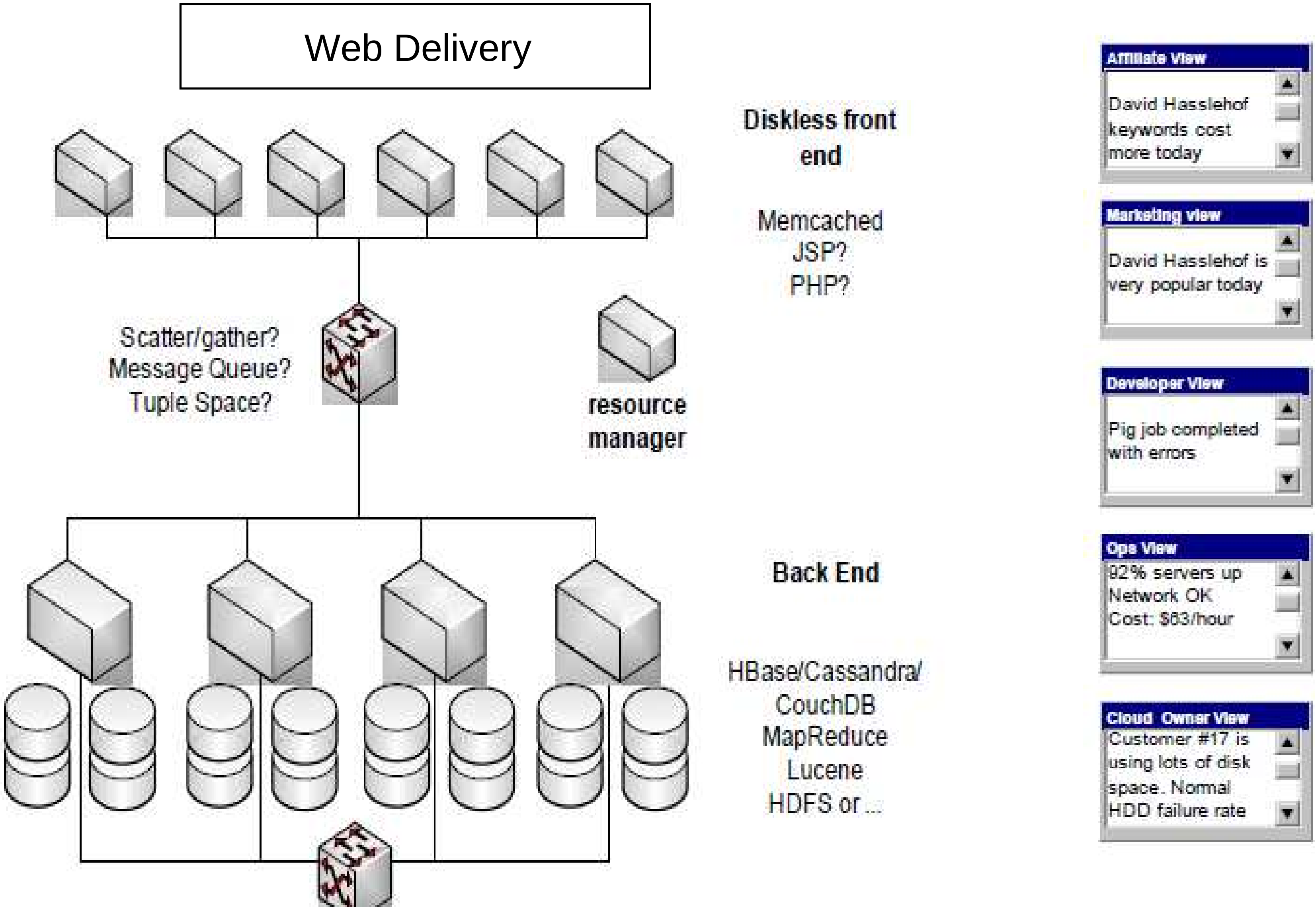
  - High-performance private network (availability zone)
    - Bypasses VDE and uses local cluster network for each allocation
    - Runs at "native" network speed (I.e. with Xen)
    - Cloud allocations cannot span clusters

- Backend : Support scalable and configurable resource management
- Expending standard Web middleware technology to support cloud computing
- Large scalable filestore as the foundation: Hadoop APIs (data organisation), Jave 7 NIO (I/O), Fuse (Virtual FS) , WebDAV (Distributed Authoring and versioning)
- MapReduce phase for post-processing

Apache Cloud Computing Edition (2)

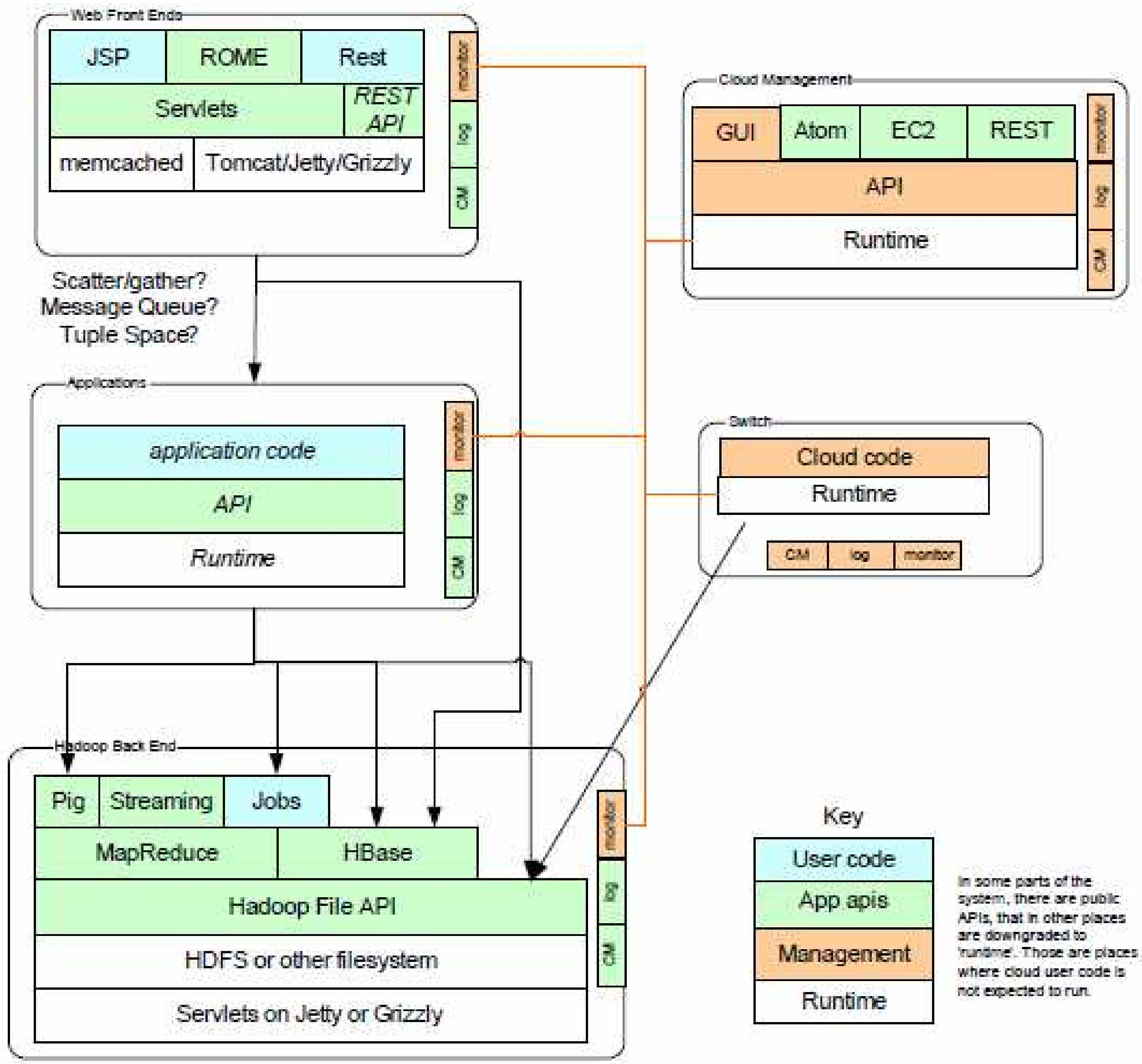
- Front End: Support for Web 2.0 application deployment
- Wide range of front end tools : Servlets, JSP, wicket, PHP and grails
- Glue Mechanism for building scalable applications: queues, scatter-gather, tuple-space, events

Cloud Computing Architecture over Apache

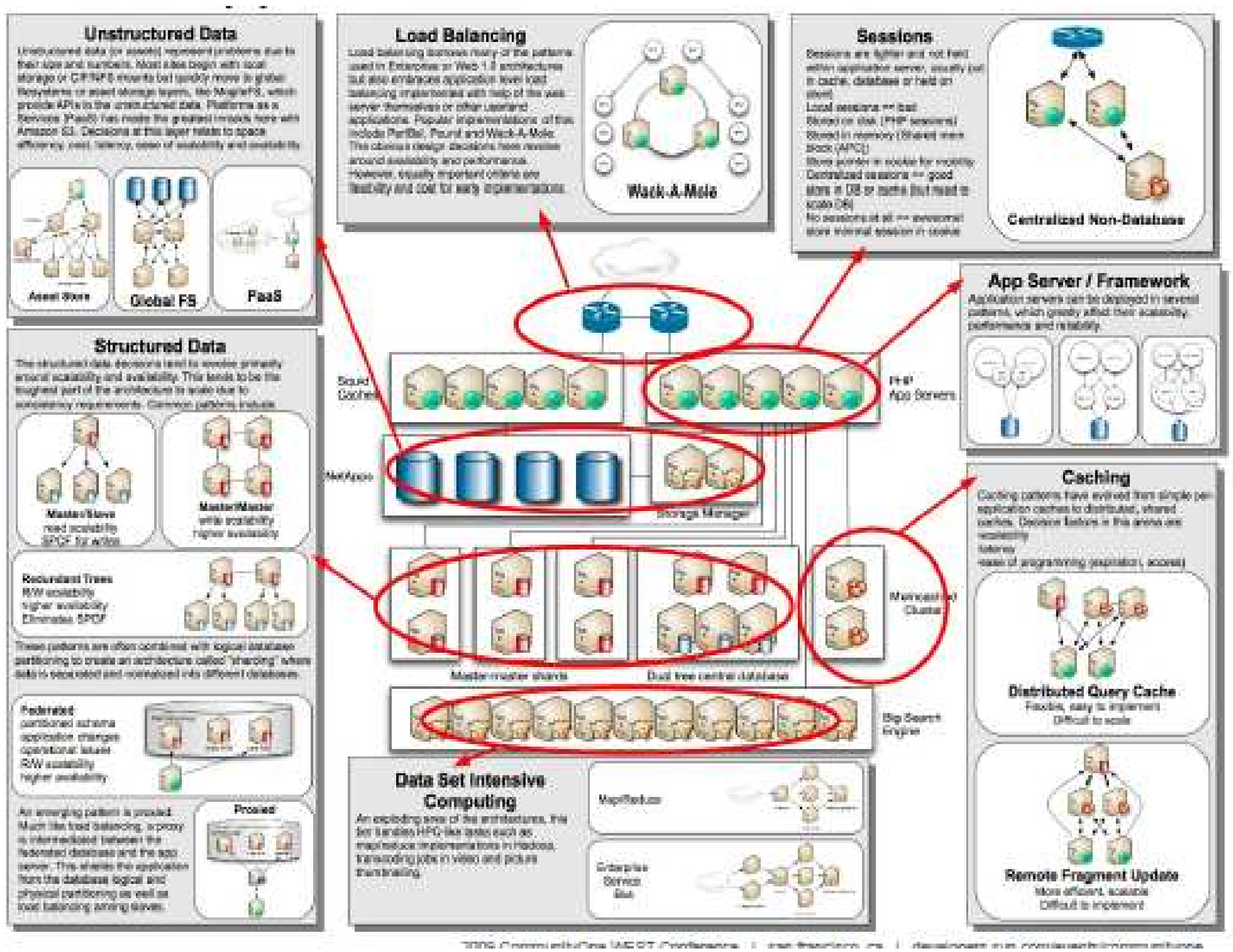


Imperial College London

An Implementation Model



There Are Quite a Lot to do to Programming within the Cloud to Build a Web 2.0 Applications

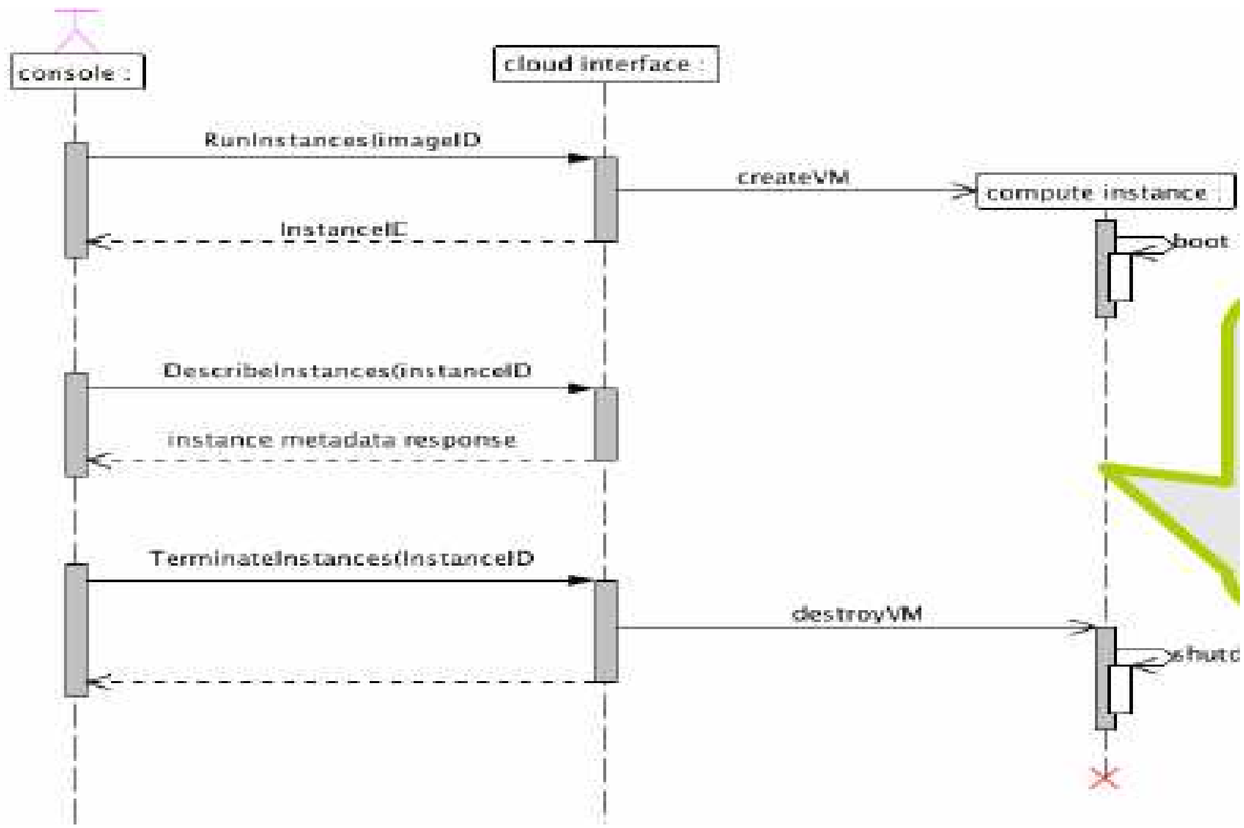


Old Story: Task-Oriented Programming and Data-Oriented Programming

- Programming within a Cloud is for efficient use of the resource to deliver the required services
- Task level programming :
  - Resource provision and monitoring
  - This can be done uniformly as VMM (Managing Images)
  - Similar with Distributed Programming: Distributed Programming Patterns
- Data level programming :
  - Exploring data parallelism for large scale data process and analysis
  - To realize the “on-demand” and “at scale” features of Cloud computing

- A Cloud computing system equips with a set of APIs to provision of subscriber computer resources into the cloud
- Provision mechanism : code executed in response to events
- Need to be flexible to cover all tiers of application architecture
  - Application servers refreshed daily
  - DB servers updated yearly
- Provision pattern: abstraction of commonly used provision mechanisms
- Efficient implementing a provision pattern: Automation for being invoked efficiently and frequently

- Deploys previously constructed and configured server images into the cloud with minimal customization.



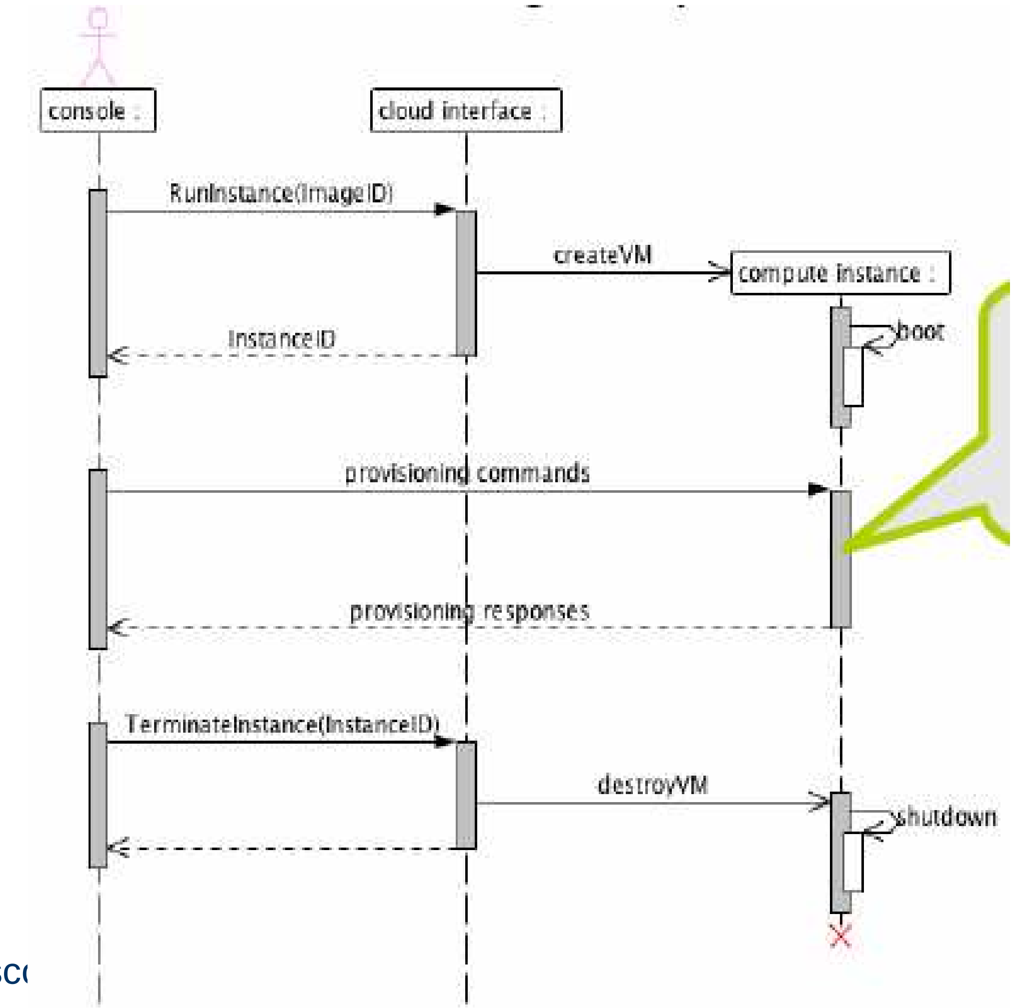
Suitable for code with low complexity, low volatility code bases

Imperial College London

Ruby Script (EC2) for Static Image Provisioning

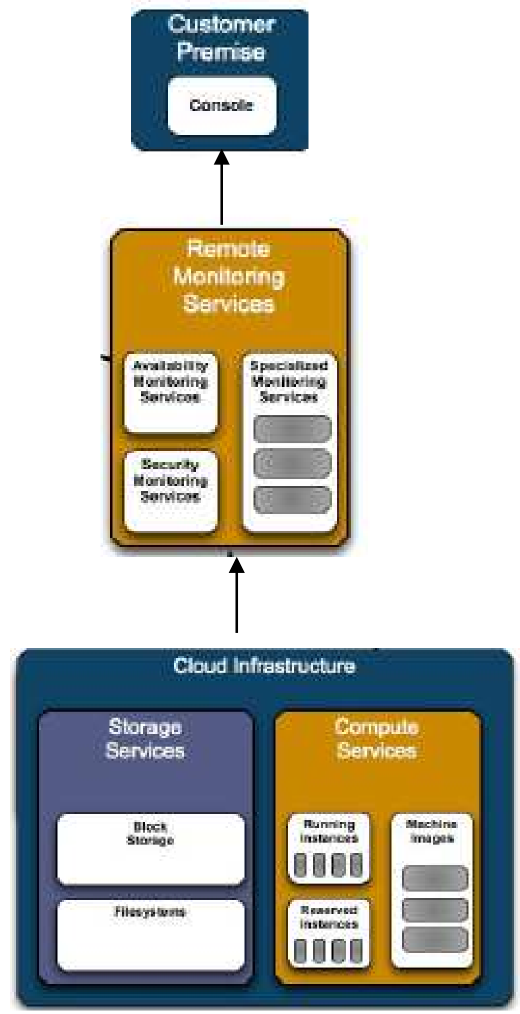
```
#!/usr/bin/env ruby

require 'rubygems'
require 'ec2'
ACCESS_KEY_ID = ENV['AWS_KEY_ID']
SECRET_ACCESS_KEY = ENV['AWS-SEKRIT-ACCESS-KEY']
 IMAGE_ID = 'ami-018e6b68' # identifies image to be launched

ec2 = EC2::Base.new
( :access_key_id => ACCESS_KEY_ID,
:secret_access_key => SECRET_ACCESS_KEY) # creates connection to cloud

ec2.run_instances(   :image_id => IMAGE_ID,
                    :min_count => 1,
                    :max_count => 5,
                    :key_name => nil,
                    :group_id => [],
                    :user_data => nil, # customization by packing payload with scripts
                    :addressing_type => "public",
                    :instance_type => "m1.small",
                    :availability_zone => nil) # launches 1 to 5 small instances of IMAGE_ID
```

- Extending static provision with additional provisioning instructions from console (usually via scripting from SSH)



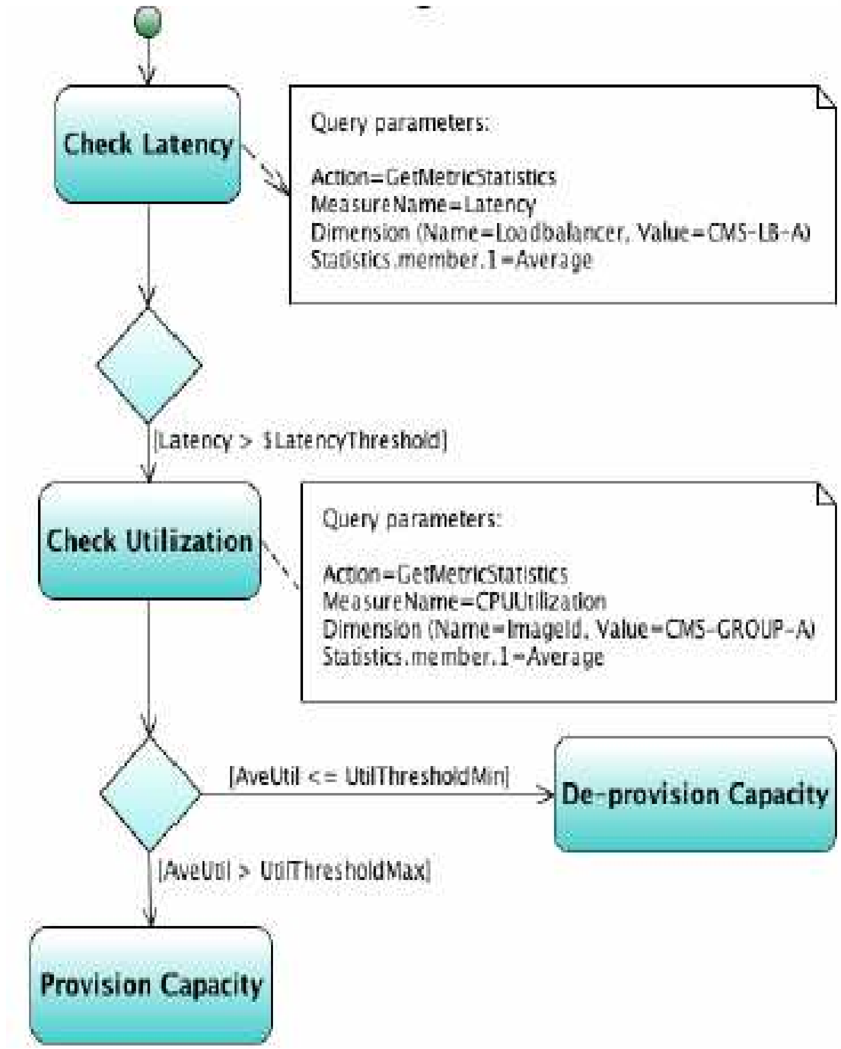
Interaction with other systems, such as pulling down packages from the Internet , is scripted here

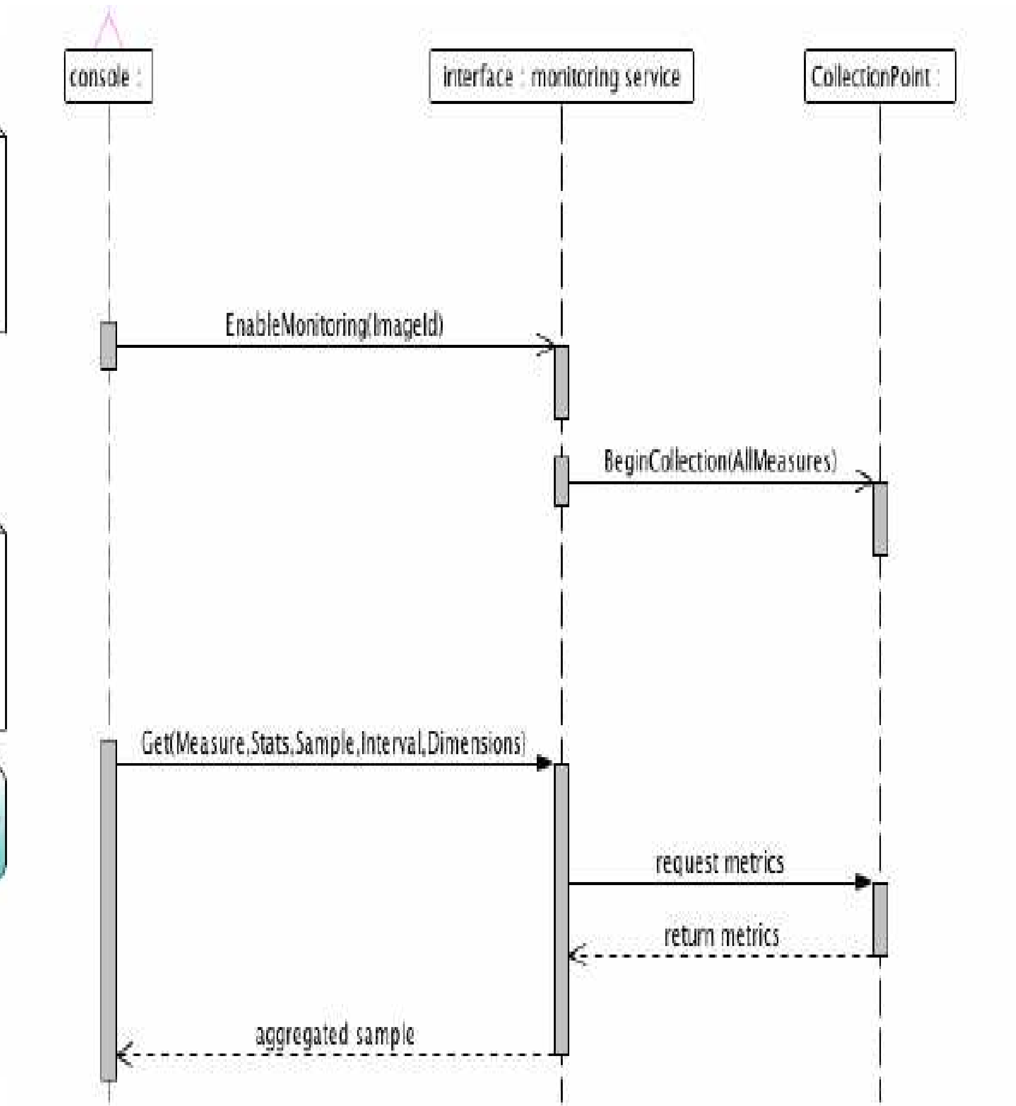
- Key function to Cloud Service Provision
- Required by
  - SLA
  - Economical Analysis
  - System Maintenance and Development



	Running Instances	Images	Object Storage	Network
Availability	√	√	√	√
Utilization	√ *		√	√
Latency / Throughput	√ *		√	√
Integrity	√	√	√	

A Simple Watching Polling Pattern





Imperial College London

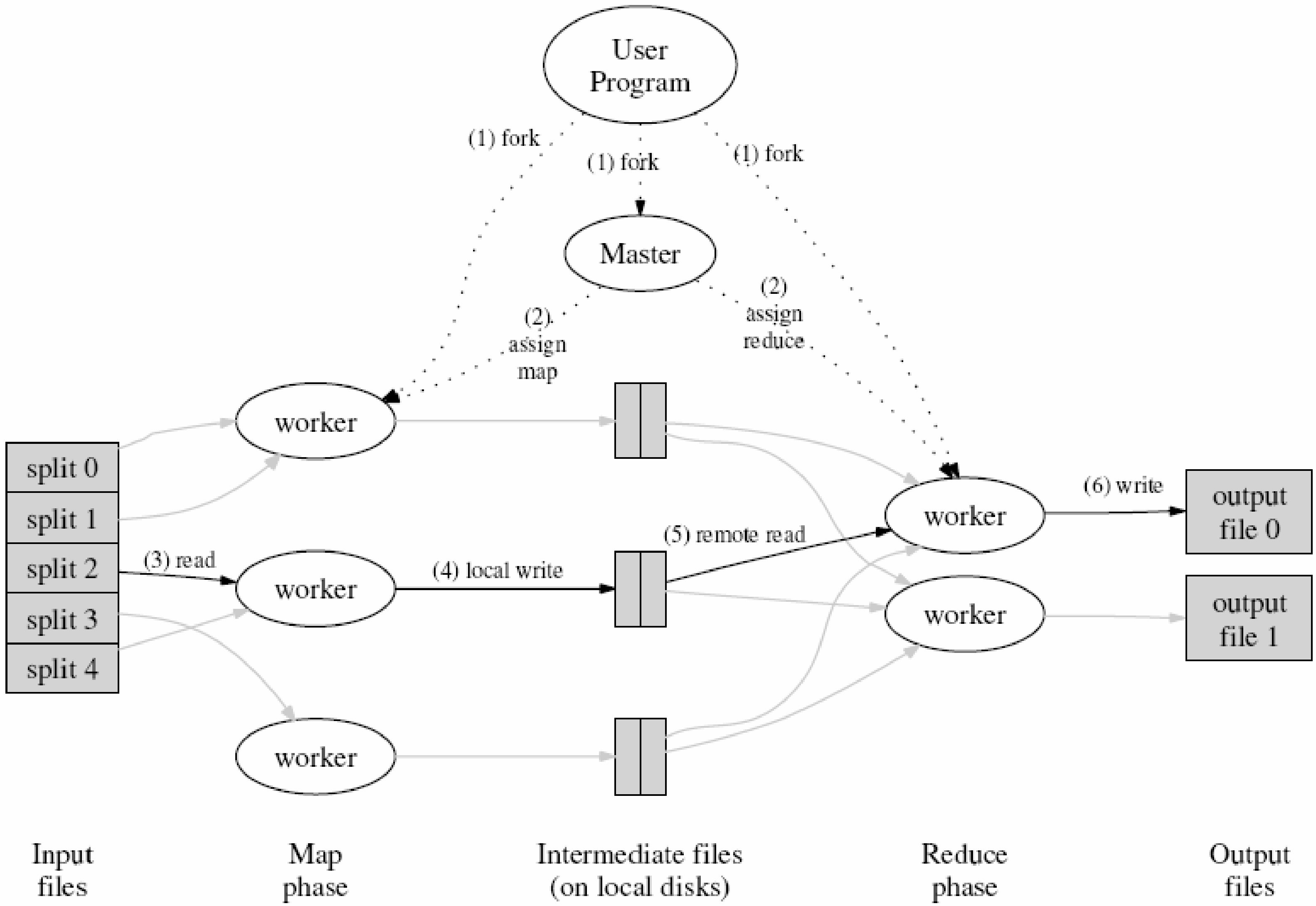
Data, Data, Data…….the Blood of a Cloud

- Cloud computing is originated from the evolution of the concept of data centers
- Centrally managed data provides the benefits of
  - Reduce the cost (hardware, premise and energy)
  - Move form capital to pay-as-you-go (operation)
  - Large scale data management
  - Dynamic information provision by community users
- The motto of Web2.0 business: Who own data owes you
- Web2.0 business need a centralized data utility business
- Cloud infrastructure is the natural model for new generation data center

- Wide class of real-world applications
  - *Environment monitoring using distributed sensors*
  - *Distributed real-time satellite data analysis*
  - *Distributed stock market price prediction*
  - *Large-scale genetic sequence analysis*

Organization	Scale of Data
Walmart	~ 20 million transactions/day
Google	~ 8.2 billion Web pages
Yahoo	~10 GB Web data/hr
NASA satellites	~ 1.2 TB/day
NCBI GenBank	~ 22 million genetic sequences
France Telecom	29.2 TB
UK Land Registry	18.3 TB
AT&T Corp	26.2 TB

Map-reduce Forms a foundation of Data Parallelism for the Cloud



This is NOT New : A Paper 14 Years Ago

Structured Parallel Programming: Theory meets Practice

John Darlington Yi-ke Guo Hing Wing To
Department of Computing
Imperial College
180 Queen's Gate, London SW7 2BZ, U.K.
E-mail: {jd, yg,hwt}@doc.ic.ac.uk

July 21, 1995

Elementary Skeletons: Parallel Arrays Operators In SCL, we use a set of second order functions as `elementary skeletons` to abstract essential data parallel computation and communication patterns. The basic functions specifying data parallelism include:

- `map` which abstracts the behaviour of broadcasting a parallel task to all the elements of an array.
- a variant of `map`, the function `imap` which takes into account the index of an element when mapping a function across an array.
- the reduction operator `fold` which abstracts tree-structured parallel reduction computation over arrays.

Input:
File containing words

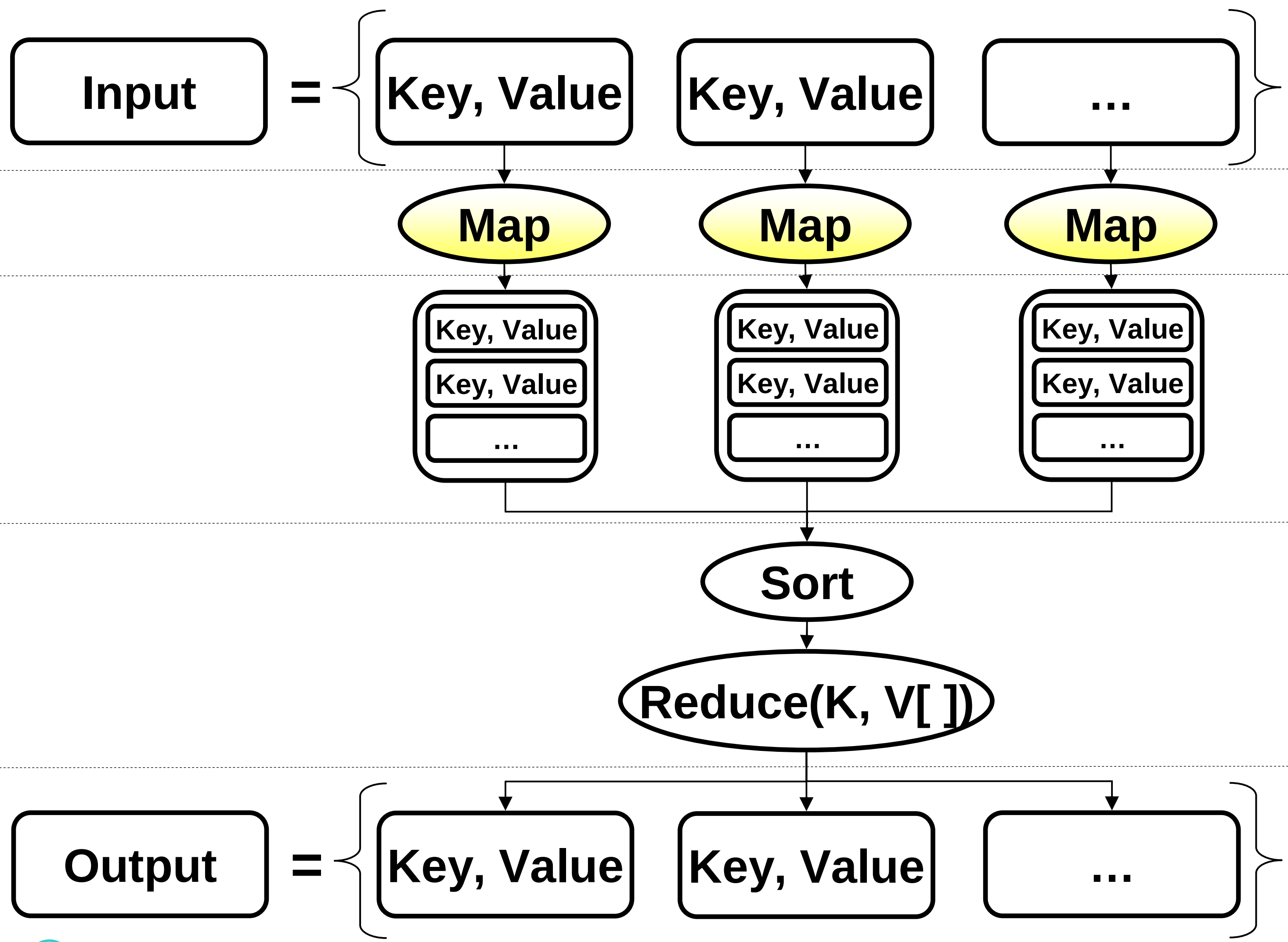
Hello World Bye World
Hello Hadoop Bye Hadoop
Bye Hadoop Hello Hadoop

WordCount

Output:
Number of occurrences of each word

Bye 3
Hadoop 4
Hello 3
World 2

An Map/Reduce illustration of the Word Counting Problem



Split Input into Key-Value pairs.

For each K-V pair call Map.

Each Map produces new set of K-V pairs.

For each distinct key, call reduce. Produces one K-V pair for each distinct key.

Output as a set of Key Value Pairs.

Input

1, “Hello World Bye World”

2, “Hello Hadoop Bye Hadoop”

3, “Bye Hadoop Hello Hadoop”

Map

Map

Map

Map Output

<Hello,1>
<World,1>
<Bye,1>
<World,1>

<Hello,1>
<Hadoop,1>
<Bye,1>
<Hadoop,1>

<Bye,1>
<Hadoop,1>
<Hello,1>
<Hadoop,1>

```
Map(K, V) {
  For each word w in V
    Collect(w, 1);
}
```

The Reduce Phase:

Map Output

<Hello,1>
<World,1>
<Bye,1>
<World,1>

<Hello,1>
<Hadoop,1>
<Bye,1>
<Hadoop,1>

<Bye,1>
<Hadoop,1>
<Hello,1>
<Hadoop,1>

Internal Grouping

<Bye → 1, 1, 1>

<Hadoop → 1, 1, 1, 1>

<Hello → 1, 1, 1>

<World → 1, 1>

Reduce

Reduce

Reduce

Reduce Output

<Bye, 3>
<Hadoop, 4>
<Hello, 3>
<World, 2>

```
Reduce(K, V[ ]) {
  Int count = 0;
  For each v in V
    count += v;
  Collect(K, count);
}
```

Map/Reduce Is a Programming Model and Can Be Implemented in Many Languages

Map/Reduce in PHP

```php
<?php
function map($buffer) {
    $words = preg_split(
      '/\W/', strtolower($buffer),
      0, PREG_SPLIT_NO_EMPTY
    );

    foreach ($words as $word) {
        emit_intermediate($word, 1);
    }
}

function reduce($word, array $values) {
    emit($word, array_sum($values));
}
```

```php
function emit_intermediate($key, $value) {
    global $data;

    if (isset($data[$key])) {
        $data[$key][] = $value;
    } else {
        $data[$key] = array($value);
    }
}
function emit($key, $value) {
    print "$key: $value\n";
}
```

```php
$data = array();

$text = <<<EOT
foo bar baz
bar bar
baz foo
EOT;
```

```php
foreach (explode("\n", $text) as $line) {
    map($line);
}
```

```php
foreach ($data as $key => $values) {
    reduce($key, $values);
}
```

```php
emit_intermediate('foo', 1);
emit_intermediate('bar', 1);
emit_intermediate('baz', 1);
emit_intermediate('bar', 1);
emit_intermediate('bar', 1);
emit_intermediate('baz', 1);
emit_intermediate('foo', 1);
```

```php
reduce('foo', array(1, 1));
reduce('bar', array(1, 1, 1));
reduce('baz', array(1, 1));
```

```
foo: 2
bar: 3
baz: 2
```

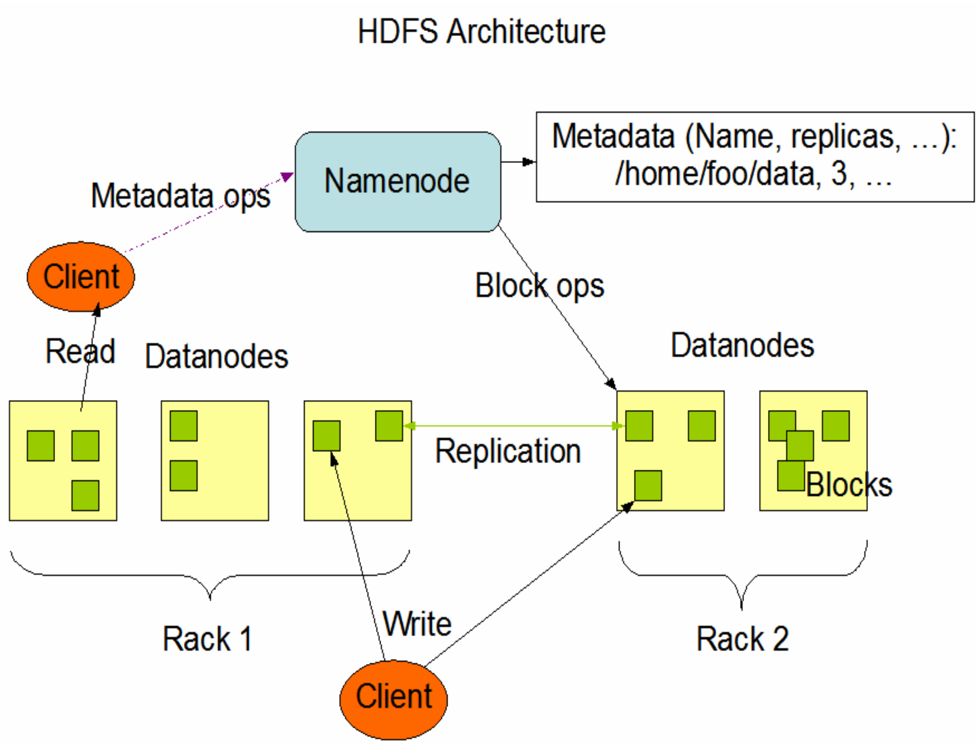
Hadoop: Supporting Map/Reduce over Distributed Files

- Open Source implementation of Map/Reduce by Apache
- Java software framework
- In use and supported by Yahoo!
- Hadoop consists of the following components:
  - Processing : Map/Reduce
  - Storage: HDFS, Hbase (Google Bigtable)

@Yahoo!
Some Webmap size data:
Number of links between pages in the index: **roughly 1 trillion links**
Size of output: **over 300 TB, compressed!**
Number of cores used to run a single Map-Reduce job: **over 10,000**
Raw disk used in the production cluster: **over 5 Petabytes**

(source: http://developer.yahoo.com/blogs/hadoop/2008/02/yahoo-worlds-largest-production-hadoop.html)

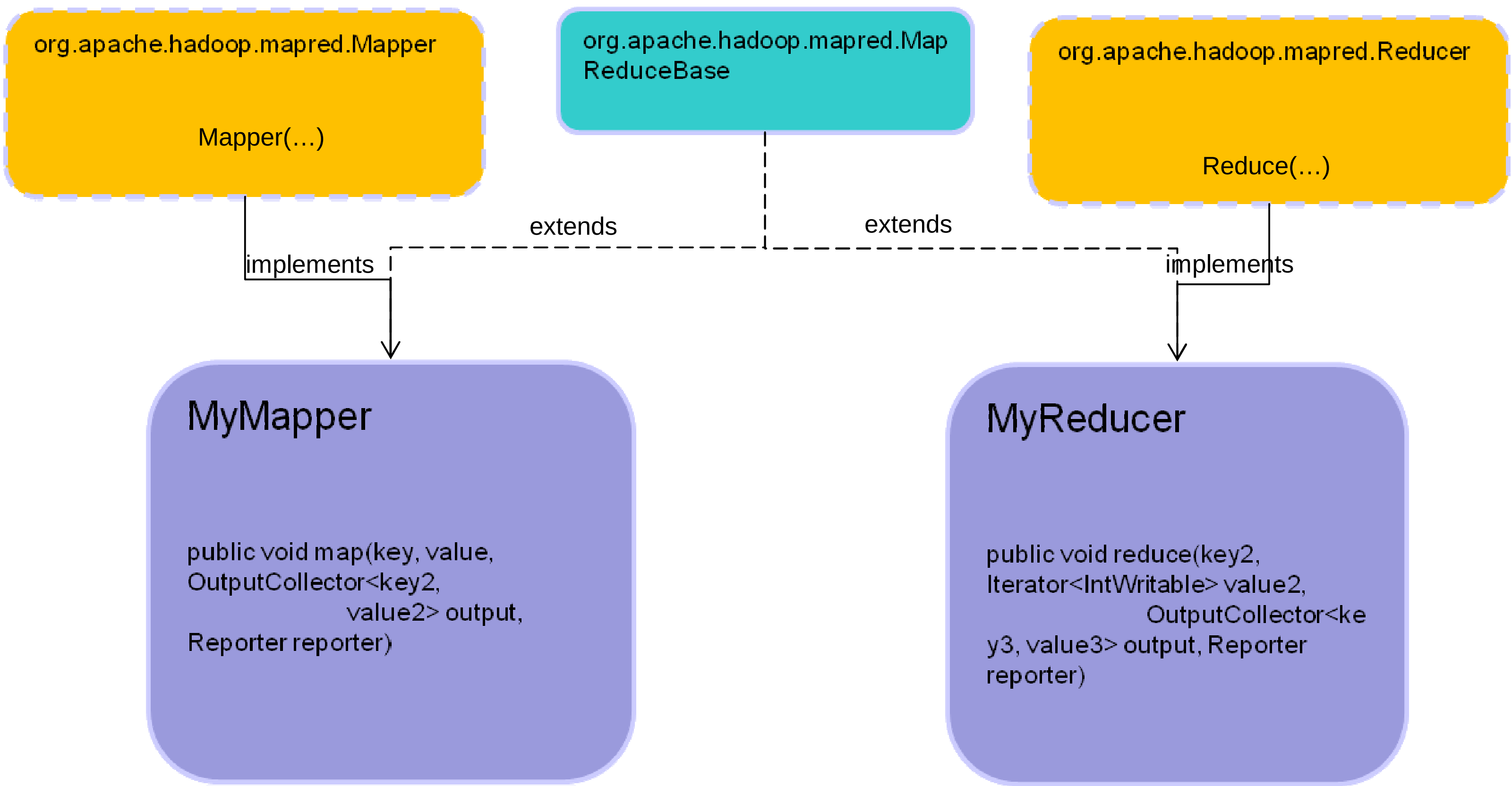
HDFS: A Distributed File System Support Map/reduce



HDFS: Distributed File System with the feature of :
•highly fault-tolerant
•designed to be deployed on low-cost hardware
•high throughput access to application data support map/reduce model for processing applications that have large data sets.

•HDFS has a master/slave architecture. An HDFS cluster consists of a single NameNode managing the file system namespace and regulates access to files by clients. In addition, there are a number of DataNodes, usually one per node in the cluster, which manage storage attached to the nodes that they run on.

•HDFS exposes a file system namespace and allows user data to be stored in files. Internally, a file is split into blocks and these blocks are stored in a set of DataNodes. The NameNode executes file system namespace operations. It also determines the mapping of blocks to DataNodes. The DataNodes are responsible for serving read and write requests from the file system's clients. The DataNodes also perform block creation, deletion, and replication upon instruction from the NameNode.



Imperial College London

Map Class Sample Code

```
public static class Map 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 tokenizer = new StringTokenizer(line);
        while (tokenizer.hasMoreTokens())
        {
                word.set(tokenizer.nextToken());
                    output.collect(word, one);
        }
    }
}
```

Reduce Class Sample Code

```
public static class Reduce 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));
    }
}
```

Find the top 10 most visited pages in each category

Visits

User	Url	Time
Amy	cnn.com	8:00
Amy	bbc.com	10:00
Amy	flickr.com	10:05
Fred	cnn.com	12:00

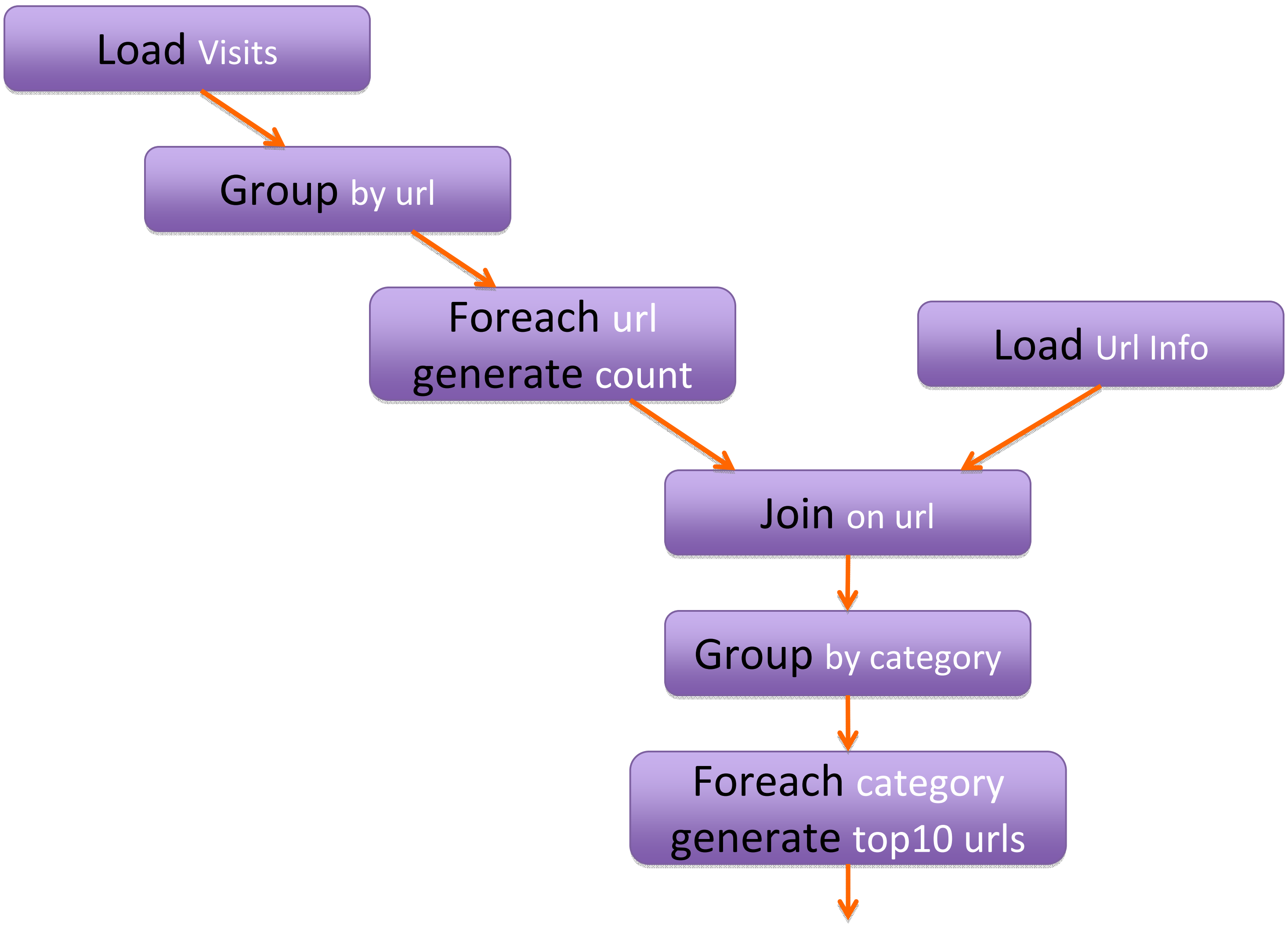
Url Info

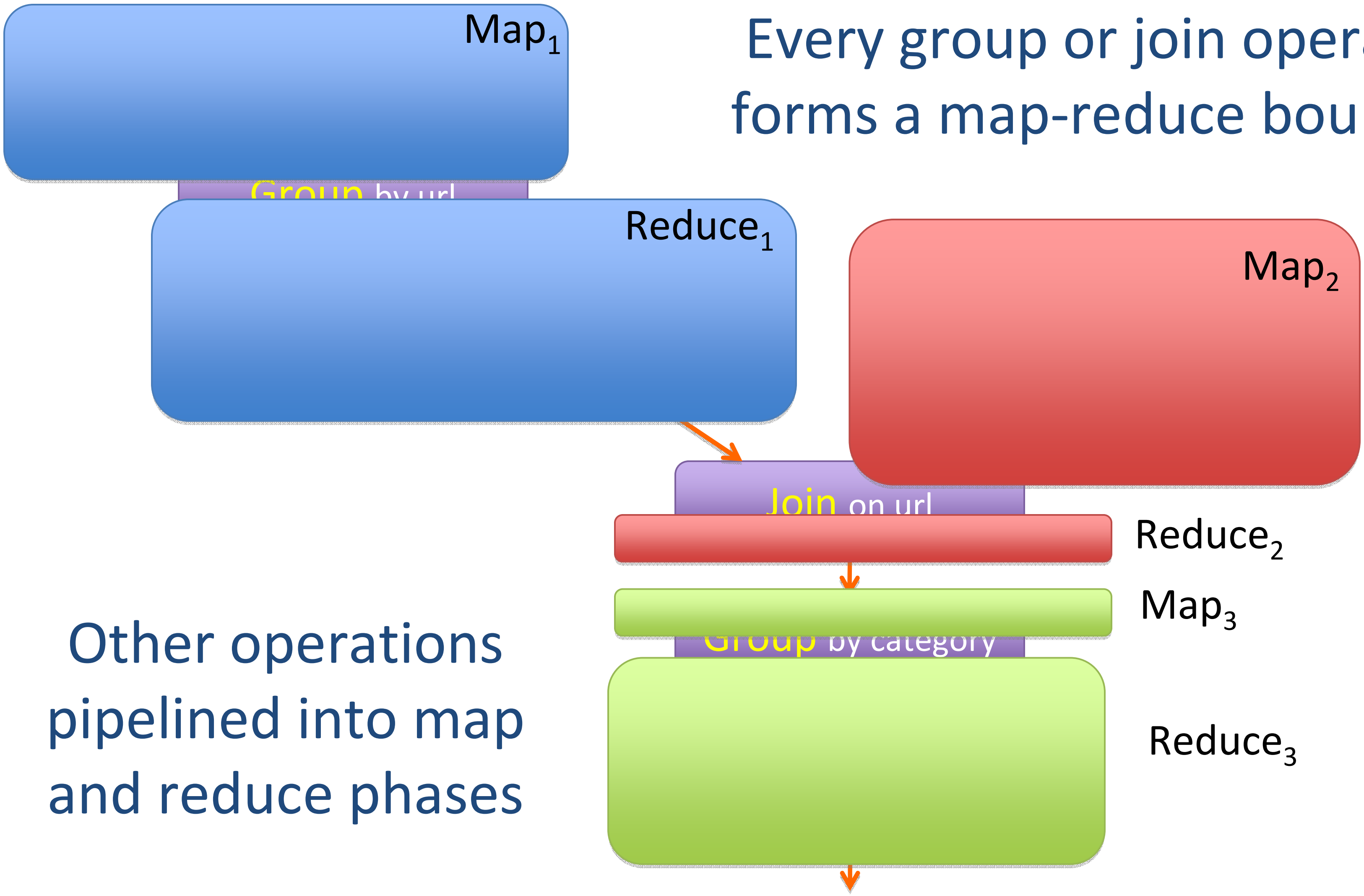
Url	Category	PageRank
cnn.com	News	0.9
bbc.com	News	0.8
flickr.com	Photos	0.7
espn.com	Sports	0.9

Yahoo! Pig Latin: A Functional Language for Hadoop

```
visits            = load '/data/visits' as (user, url, time);
gVisits           = group visits by url;
visitCounts  = foreach gVisits generate url, count(visits);

urlInfo           = load '/data/urlInfo' as (url, category, pRank);
visitCounts  = join visitCounts by url, urlInfo by url;

gCategories = group visitCounts by category;
topUrls = foreach gCategories generate top(visitCounts,10);

store topUrls into '/data/topUrls';
```

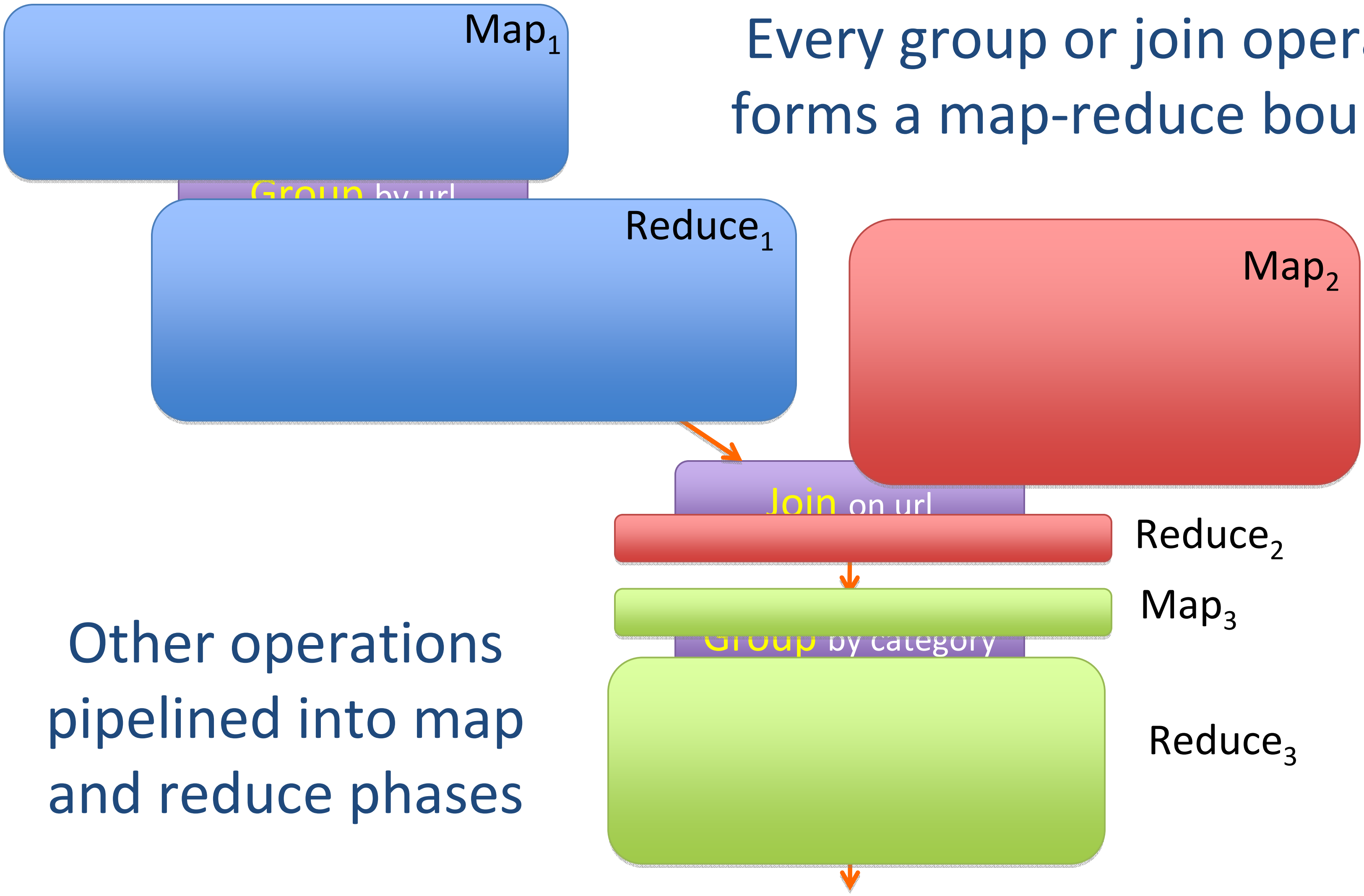
Map/Reduce Can be Implemented via Compiler





Every group or join operation forms a map-reduce boundary

Other operations pipelined into map and reduce phases

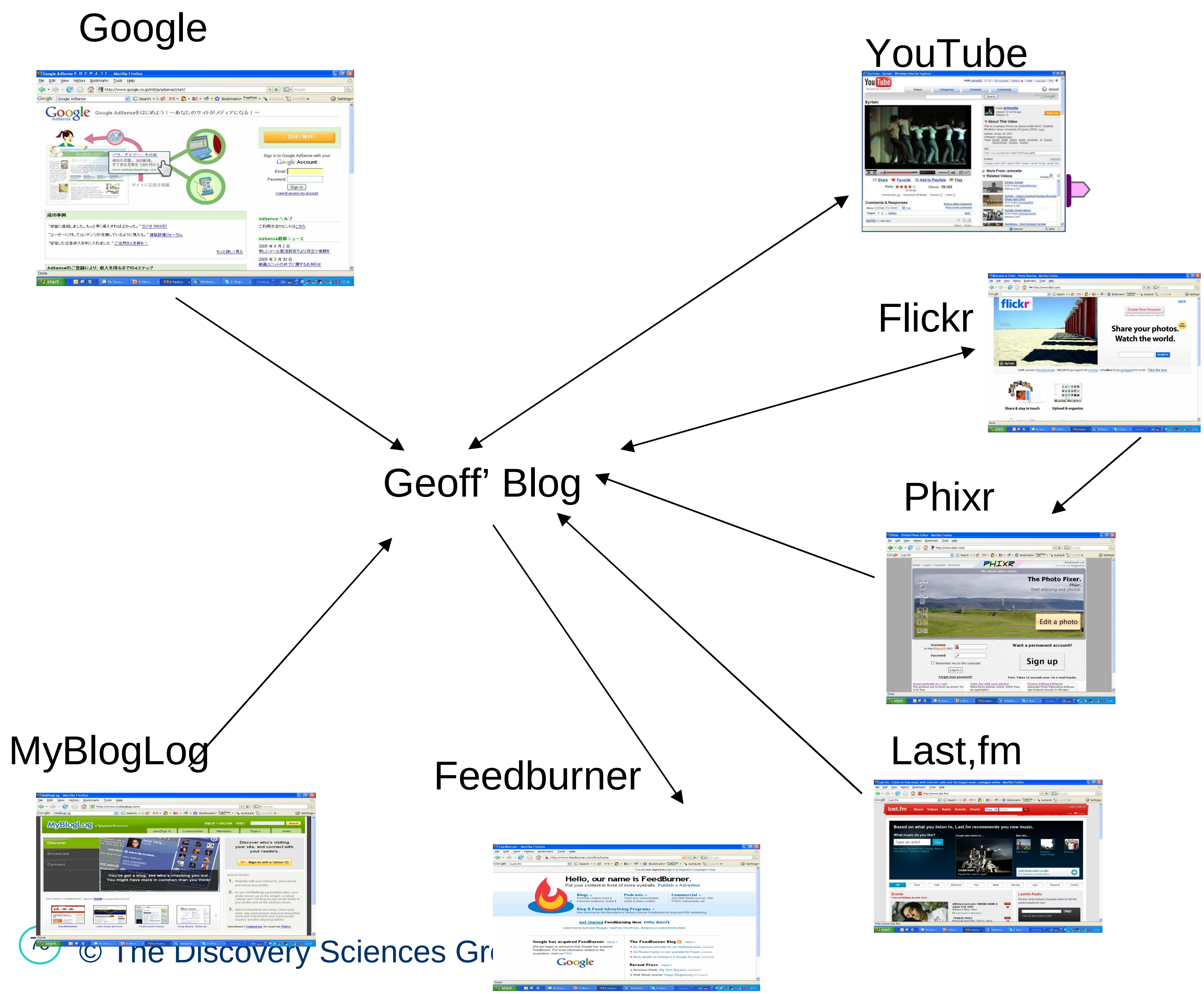


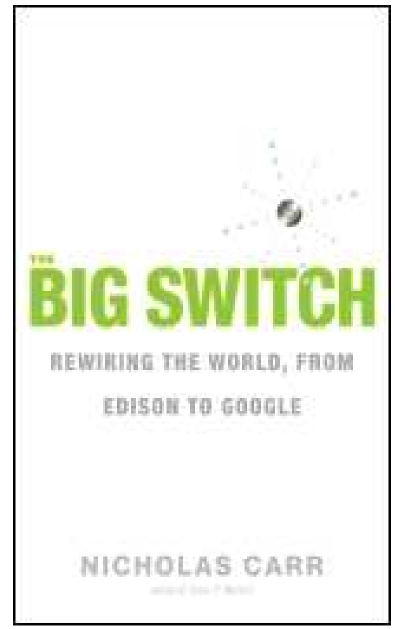
Every group or join operation forms a map-reduce boundary

Other operations pipelined into map and reduce phases

Imperial College London

Programming over the Cloud : A Story of Geoff's Blog





P120

What did Geoff do:

- Geoff's work, which only took him a few hours, gives a sense of how simple it's become to draw and service from various utility suppliers and combine them onto s single Web page. What's remarkable is that he didn't need to install any software or store any data on his PC. The various software applications, and all the data, reside on the utility companies' systems. Using simple tools, he programmed all these far-flung machines to create a multimedia experience for his readers…….P 121, The Big Switch

- Geoff programmed, but he programs over the cloud (a set of services) rather than over computers.

- Programming over cloud means composing services –or Mashup (by our USA friends)

Build a Mashup?

What do you want to do?

Which data you need? APIs/RSS available?
How is your programming skills?

Geek

Sign up for a developer token

http://aws.amazon.com/
http://www.google.com/apis/maps/
http://api.search.yahoo.com/webservices/re

Start coding

Semi-Technical Skills

Use mashup editors

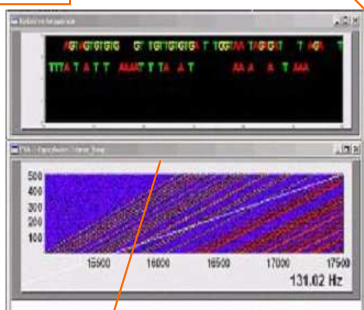
- Microsoft Popfly
- Yahoo! Pipes
- QEDWiki by IBM
- Google Mashup Editor (Coming)
- Serena Business Mashups
- Dapper
- JackBe Presto Wires

Start Configuring

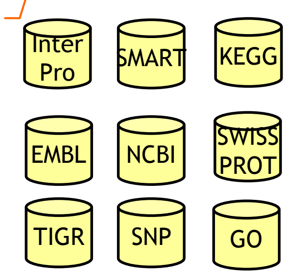
Mash up Is not New, Once We Called it Workflow

Discovery Net 2003 IEEE Supercomputing Award

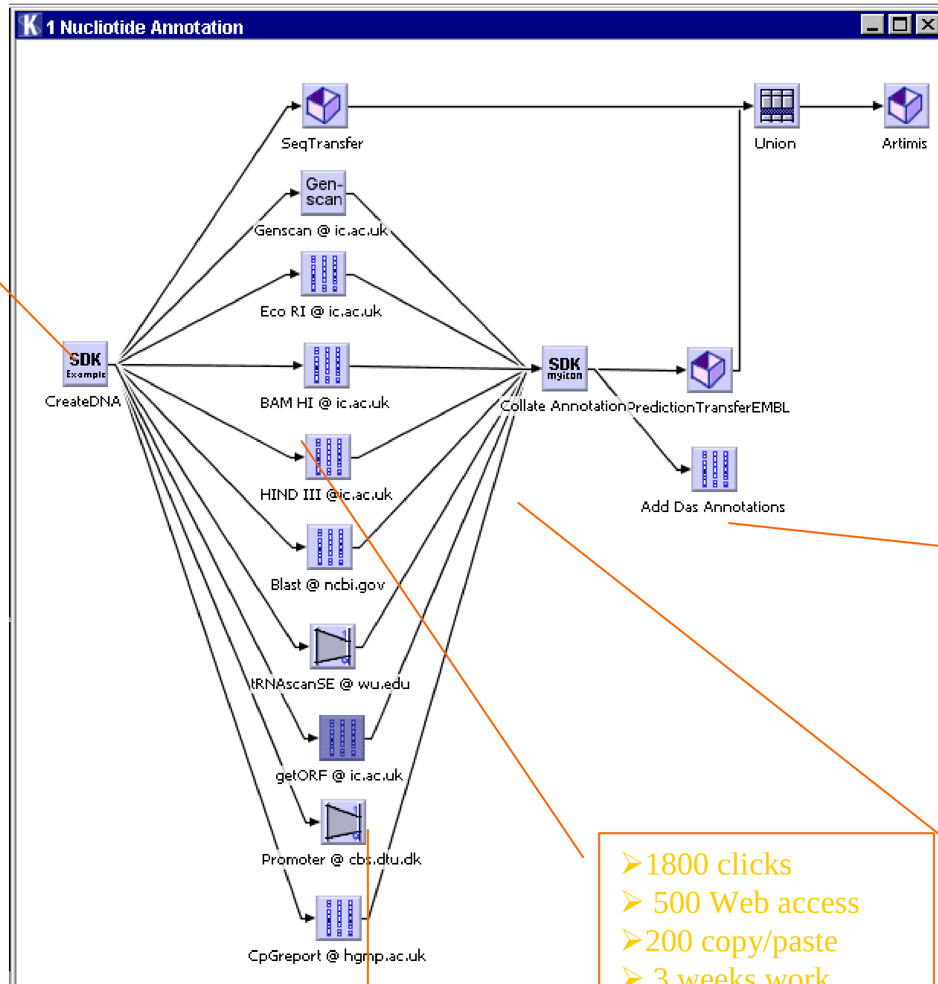
Download sequence from Reference Server



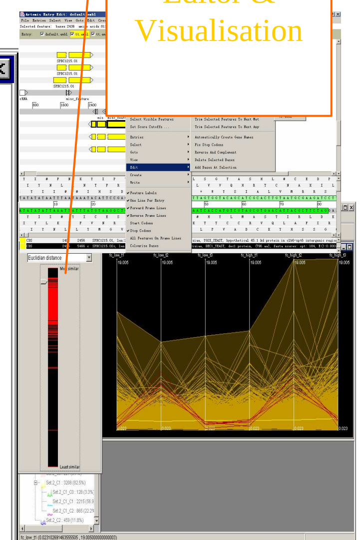
Real-time sequencing in London



Nucleotide Annotation Workflows



Interactive Editor & Visualisation



Save to Distributed Annotation Server

Execute distributed annotation workflow

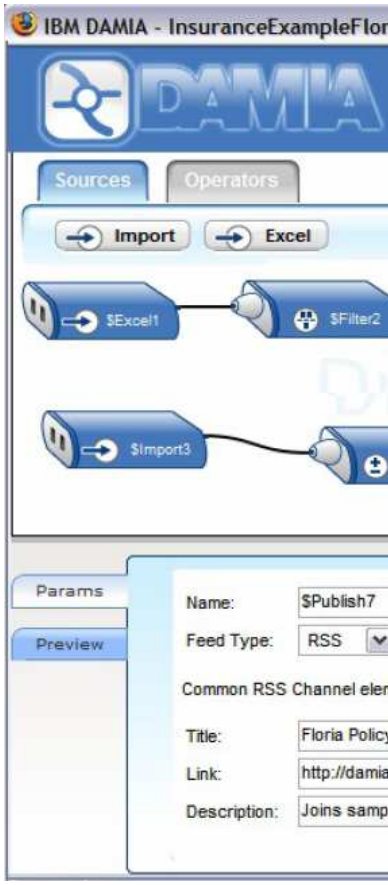
- 1800 clicks
- 500 Web access
- 200 copy/paste
- 3 weeks work

in 1 workflow and few second execution

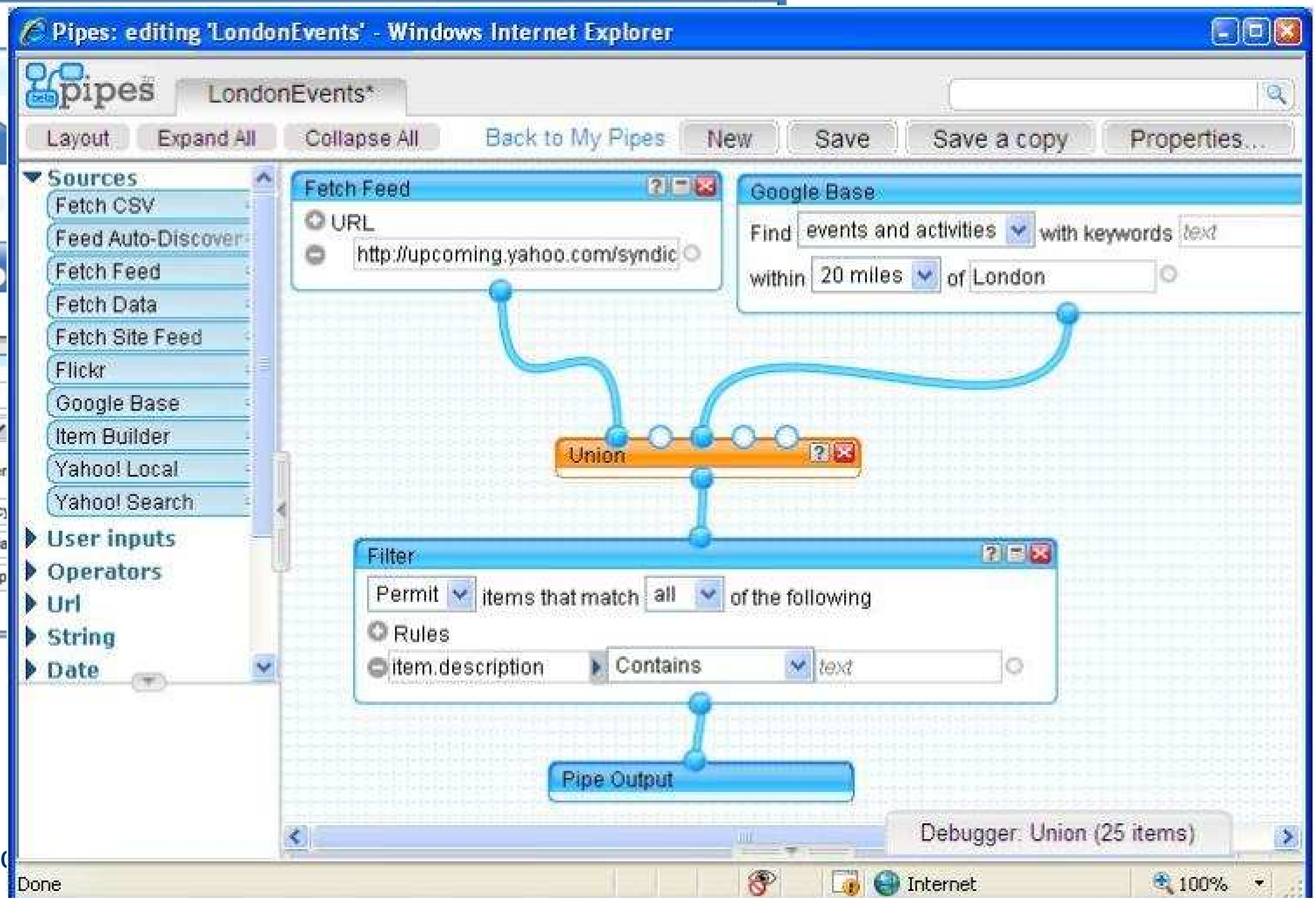
Distributed data and computation

Imperial College London

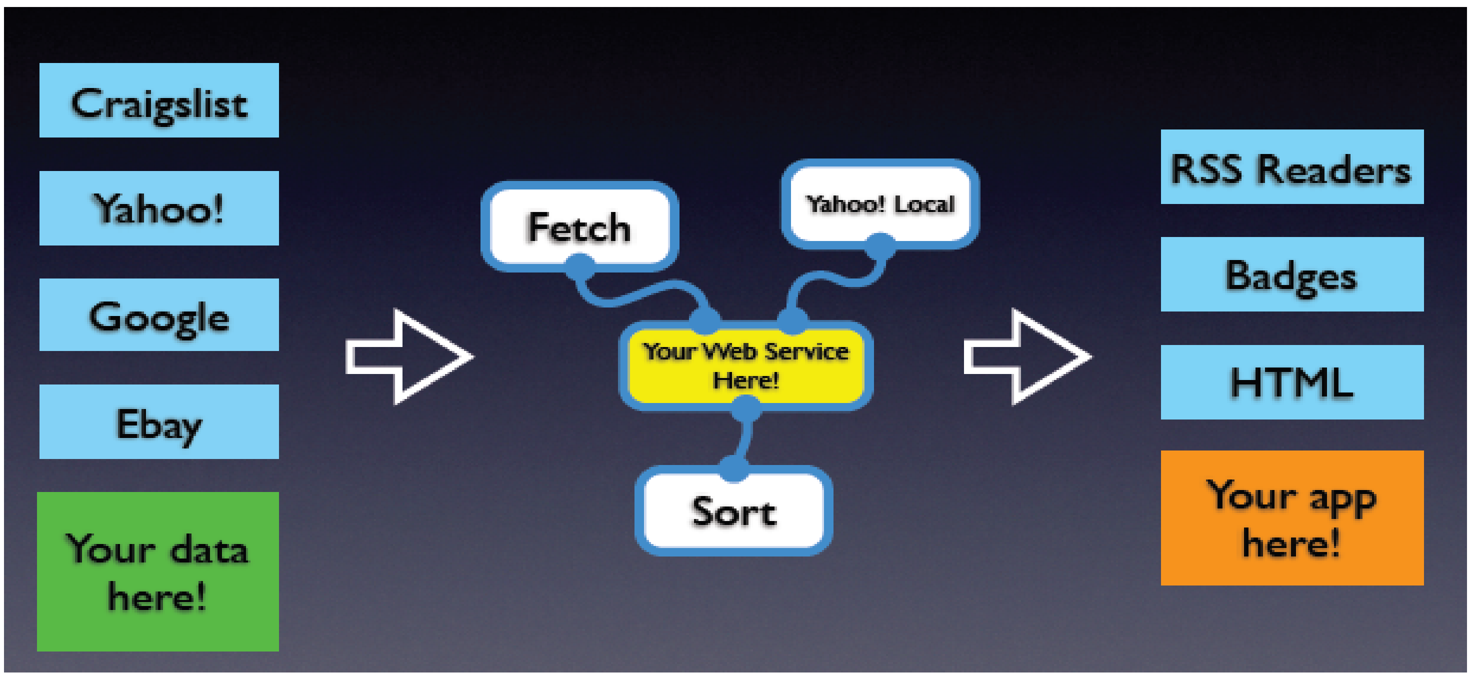
IBM DAMIA



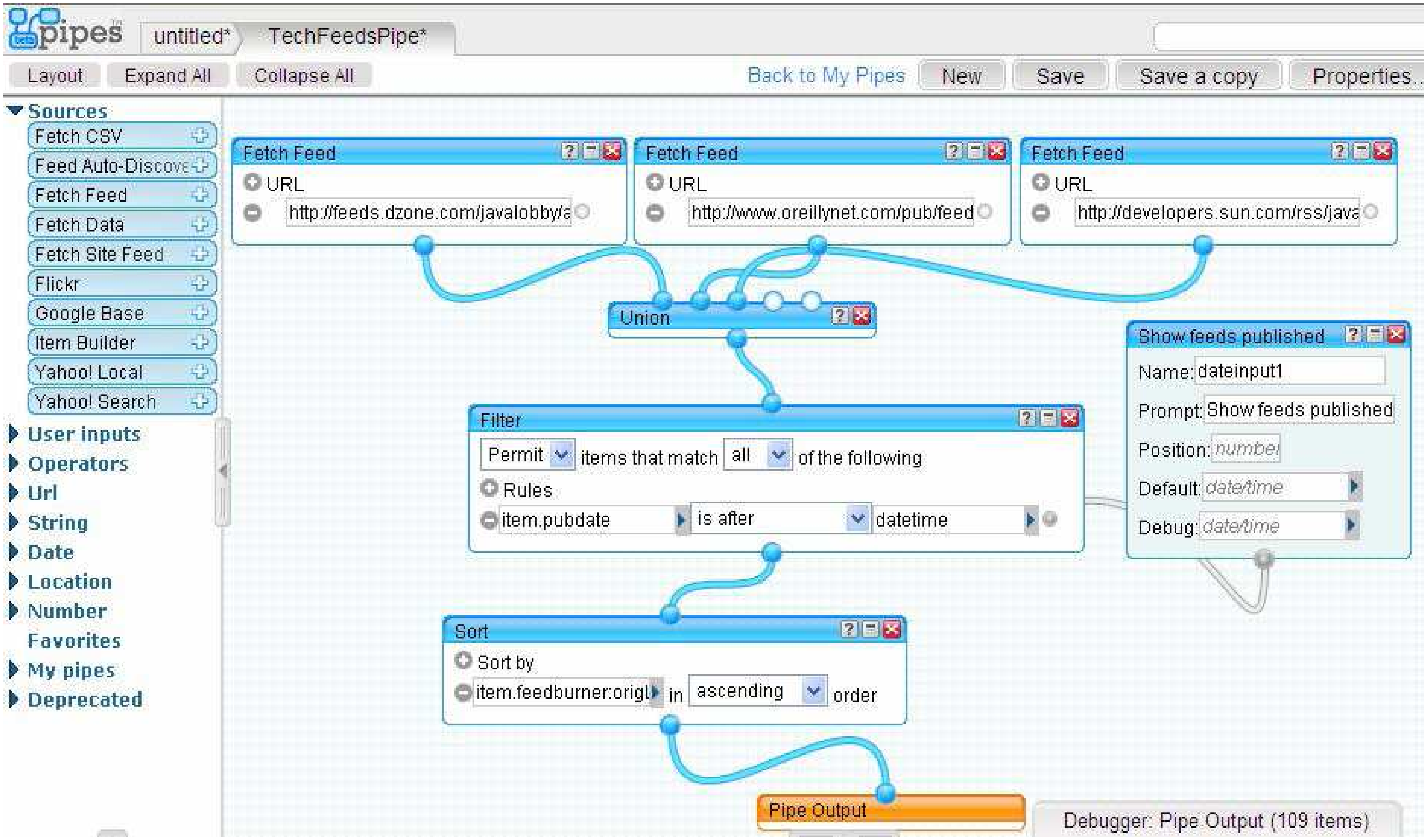
Yahoo! Pipe



- Motivation: Unix Pipes for the Web (or Visual Perl based on RSS)
- Design Principle:
  - mashup Web applications with few basic operators to form new applications
  - A compositional mechanism of the PaaS framework for Yahoo!

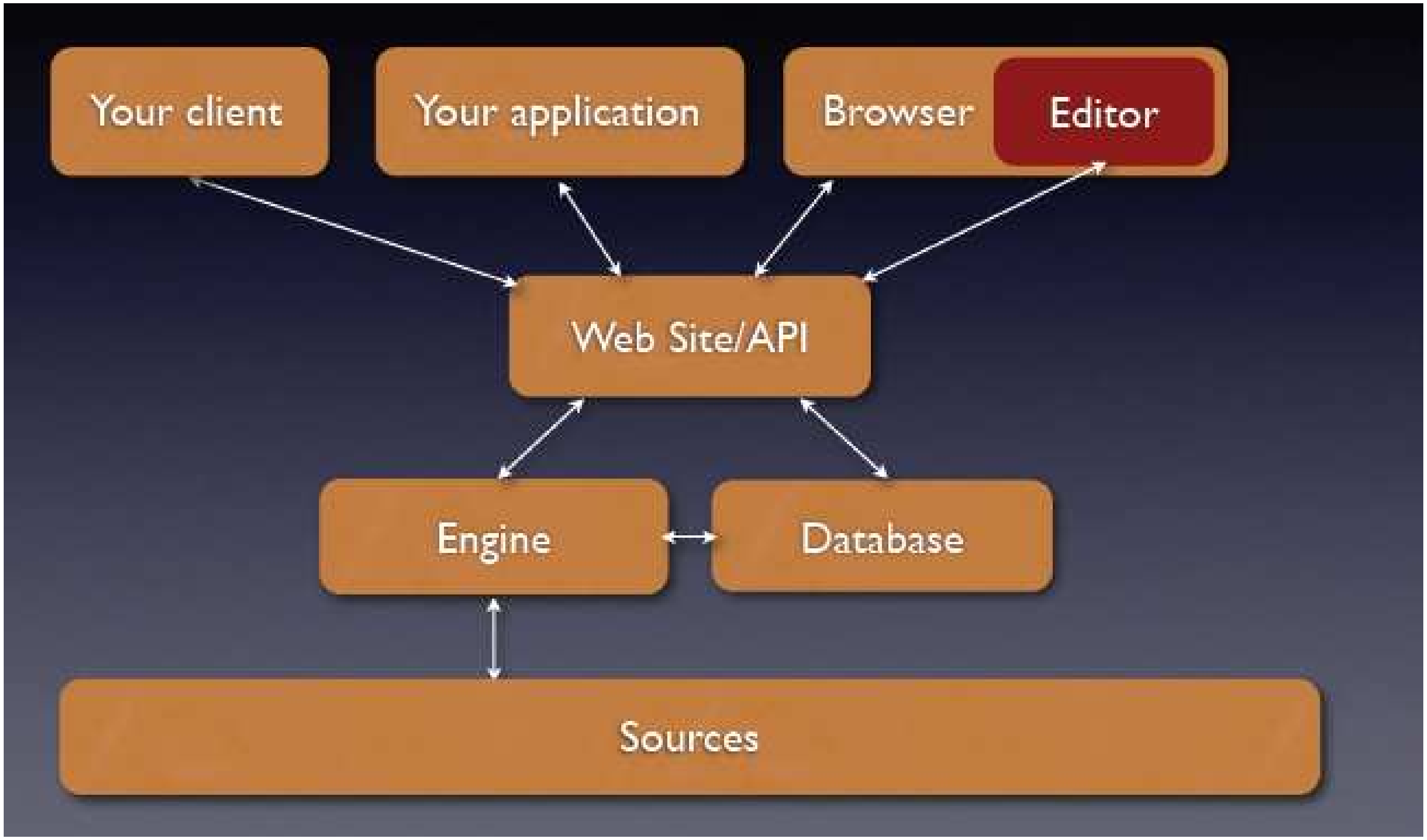


Yahoo! Pipe : Example

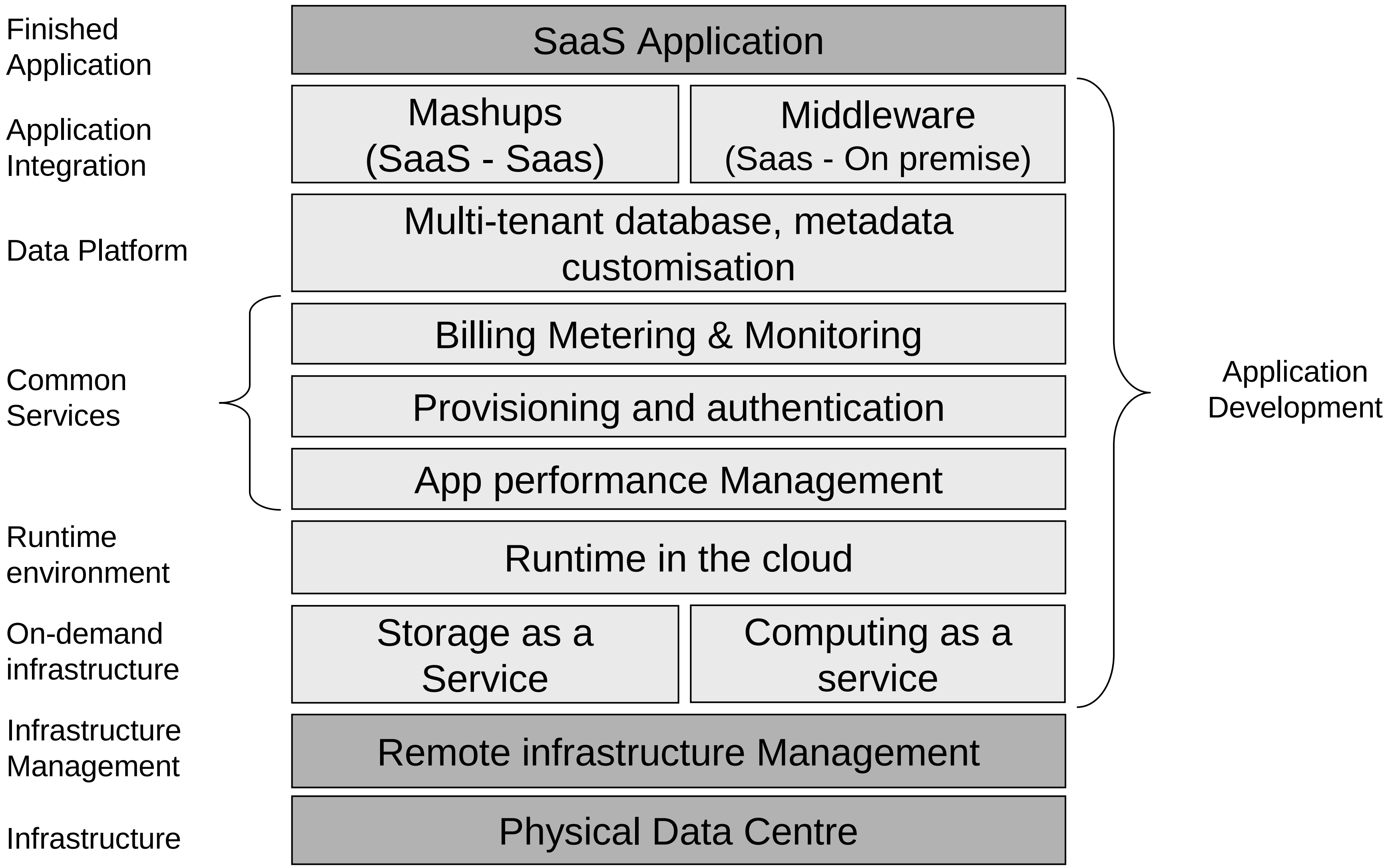


Imperial College London

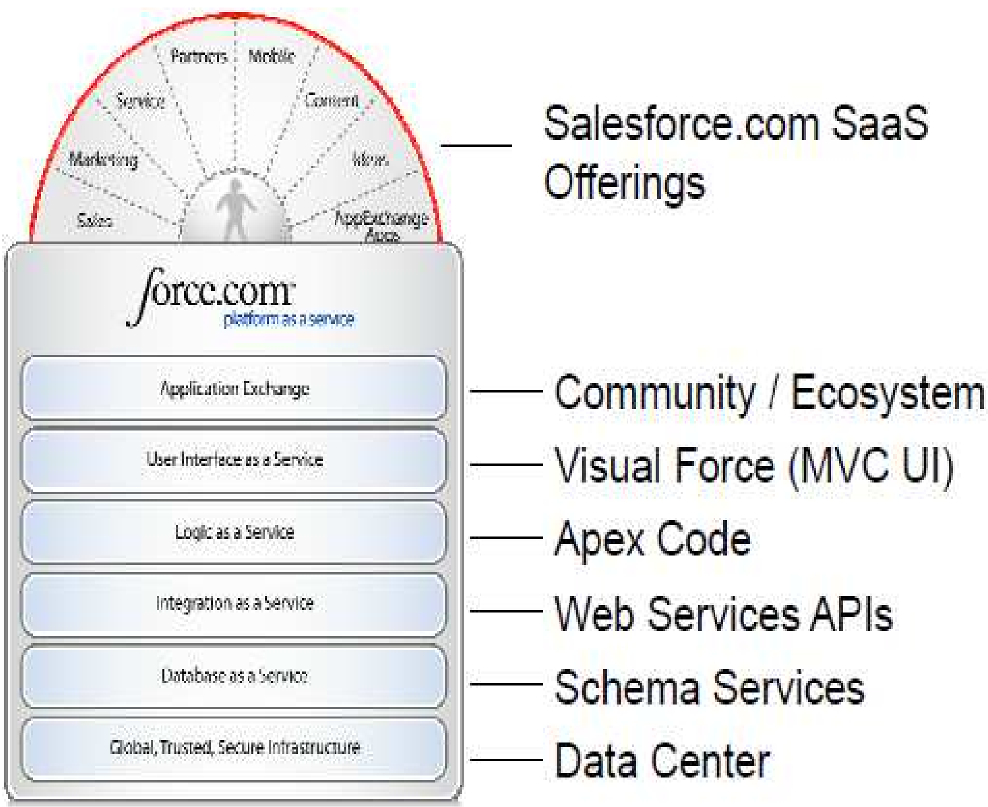
Yahoo! Pipe Architecture



Building Enterprise Cloud Needs More Sophisticated PaaS Platform



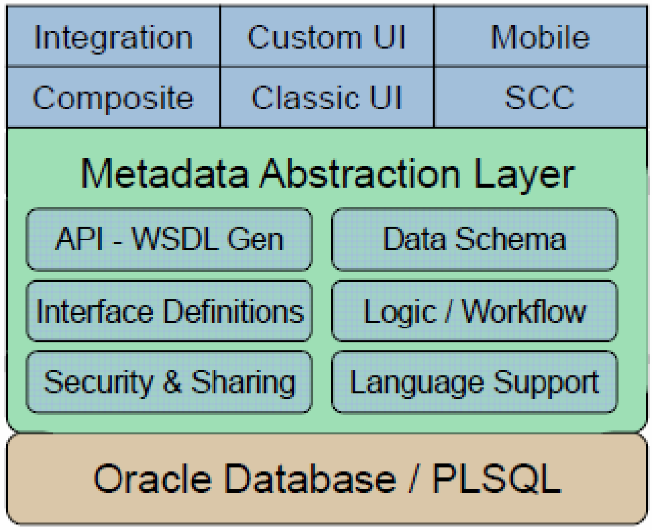
PaaS example: Force.com



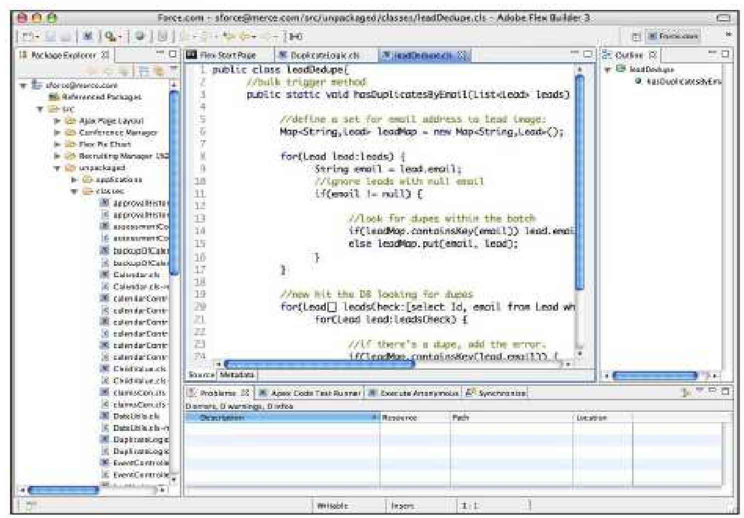
Building Applications with Force.com Platform

High Level System Abstraction

- Single low level schema
- Abstraction Layer
  - Data schemas
  - Workflow and Logic
  - User interface
  - Describable as metadata
- Enables
  - Integration Svcs
  - UI Svcs
  - Mobile Svcs
  - Security Svcs

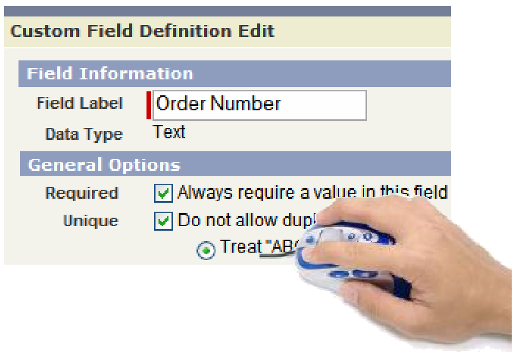


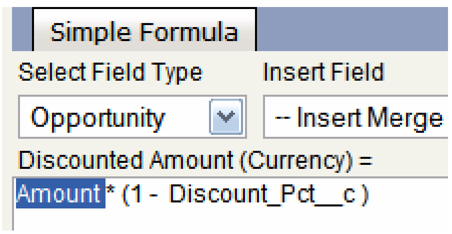
Low Level System Abstraction
(APEX Programming)

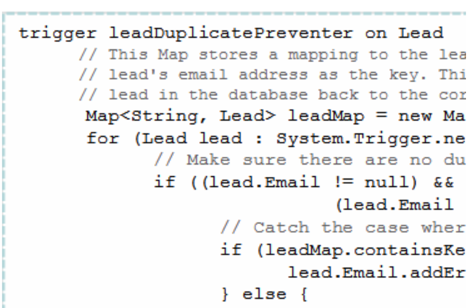


Imperial College London

Multi-tier Logic for Application Building in Force.com







Declarative Logic (point and click)

Audit History Tracking

Assignment Rules

Escalation Rules

Workflows Rules

Approval Processes

Formula-Based Logic (similar to excel)

Formula Fields

Data Validation Rules

Procedural Logic (code)

Apex Triggers (logic before or after a save, update or delete)

Apex Classes

Apex Web Services (logic that can be called by an external system)

InforSense Platform for BI Cloud Application

Reporting and Analytics SaaS

Mashups (SaaS - Saas)	Middleware (Saas - On premise)

Multi-tenant database, metadata customisation

Billing Metering & Monitoring

Provisioning and authentication

App performance Management

Runtime in the cloud

Storage as a Service	Computing as a service

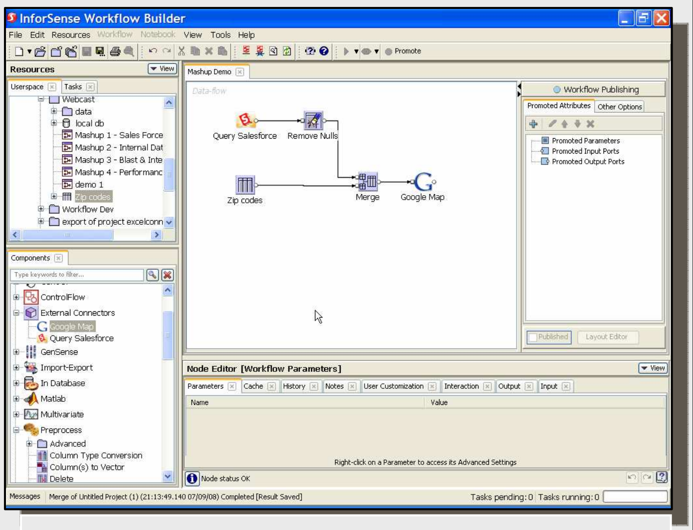
Remote infrastructure Management

Physical Data Centre

- InforSense BI application builder
  - Supports Mash-up applications
  - Supports Middleware applications
  - Enables developers to rapidly build BI applications
  - Deploy BI applications onto the cloud

InforSense PaaS Platform - Architecture

InforSense Application Builder Environment



-Graphical IDE for building BI Cloud applications.

-Workflow based virtual warehousing
-Analytical mashup

InforSense Analytical PaaS Server

Interactive Web Reporting Engine (VisualSense)

Enterprise Analytics Workflow Engine

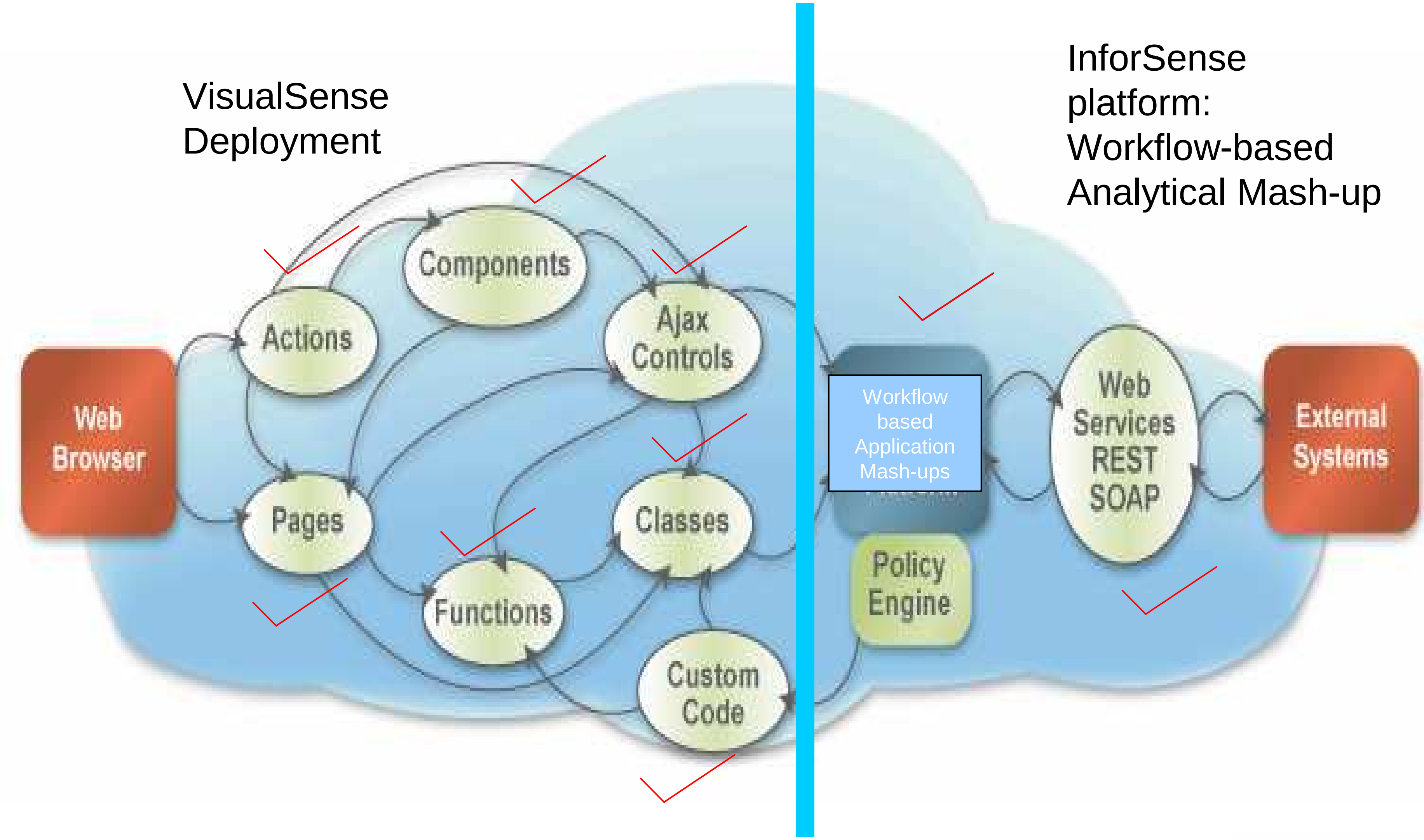
Data Processing Operators | **Analytics Operators**

Web Service Connector | **Database Connector**

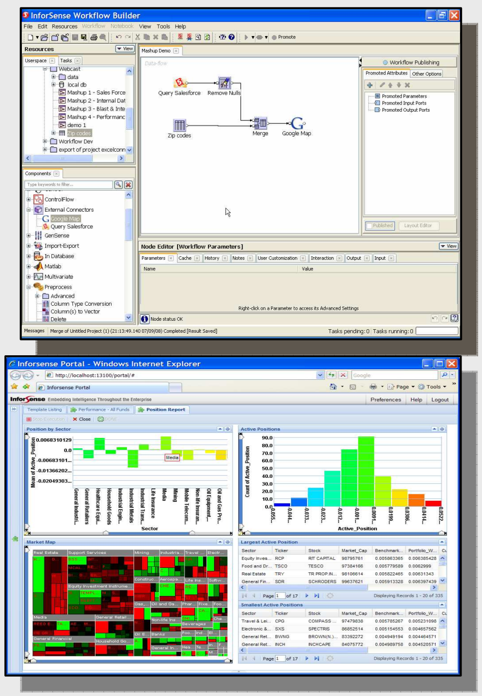
Enterprise Analytics / Data Management

Deployable on Cloud or Virtualised Infrastructures

InforSense PaaS Technology Map

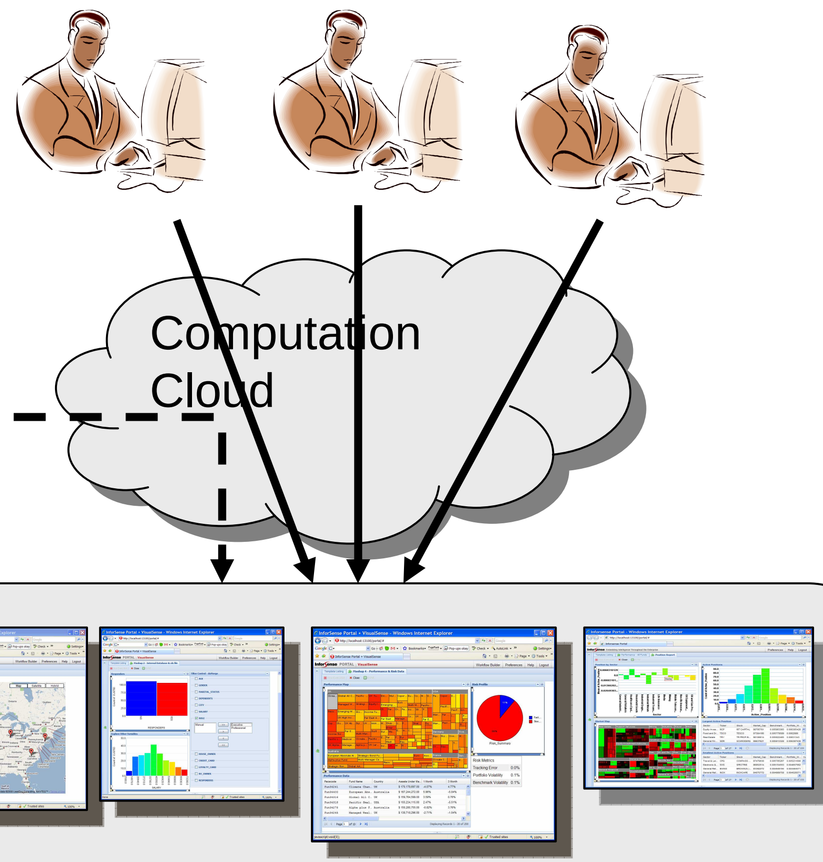


InforSense PaaS Platform - Life Cycle



1) Application developer builds and tests BI could application locally using InforSense Builder Environment

Deployment of application over cloud

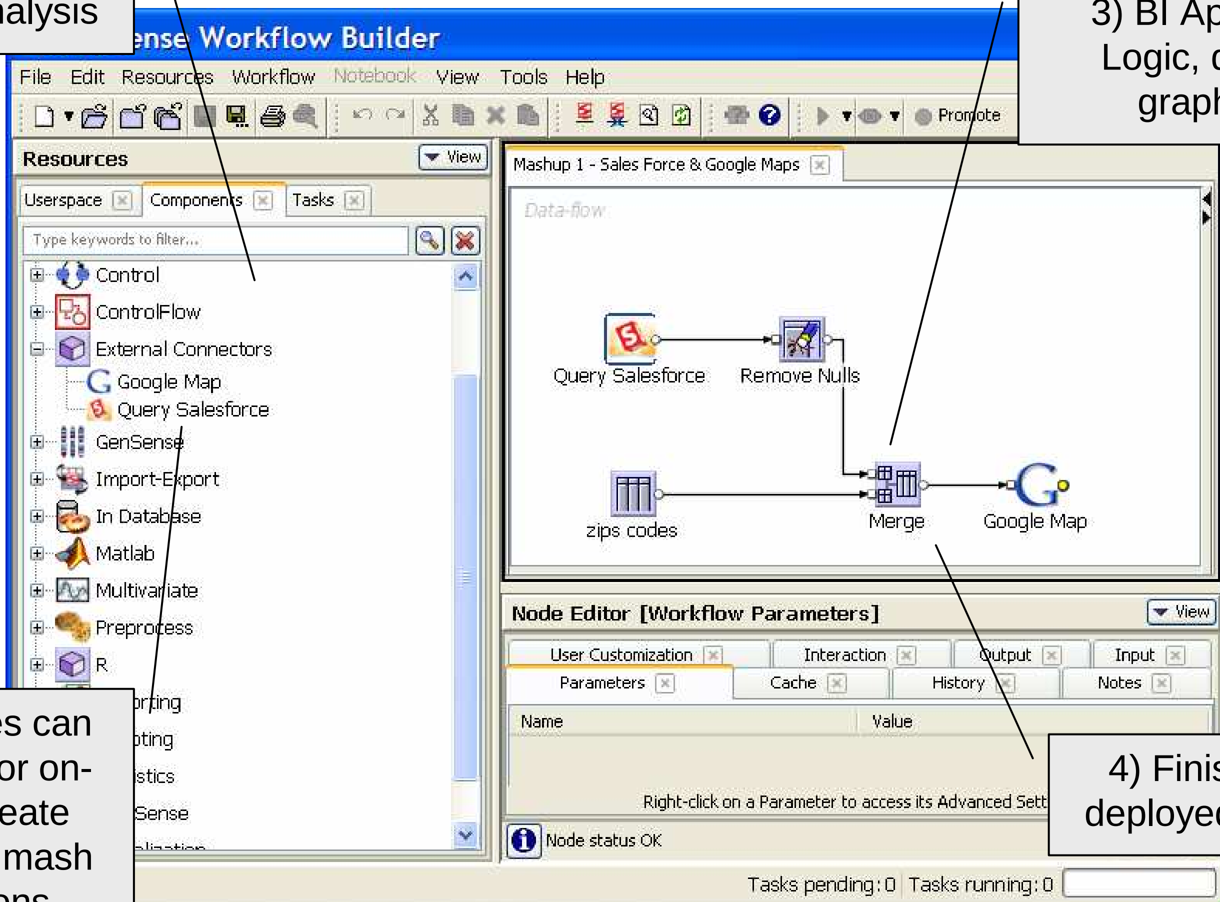


2) End User community accesses deployed BI cloud applications which are hosted and managed on Service/cloud Based Infrastructure

InforSense SaaS Platform – Cloud Application Builder

1) Components to query data sources and perform analysis

3) BI Application Logic, describe graphically

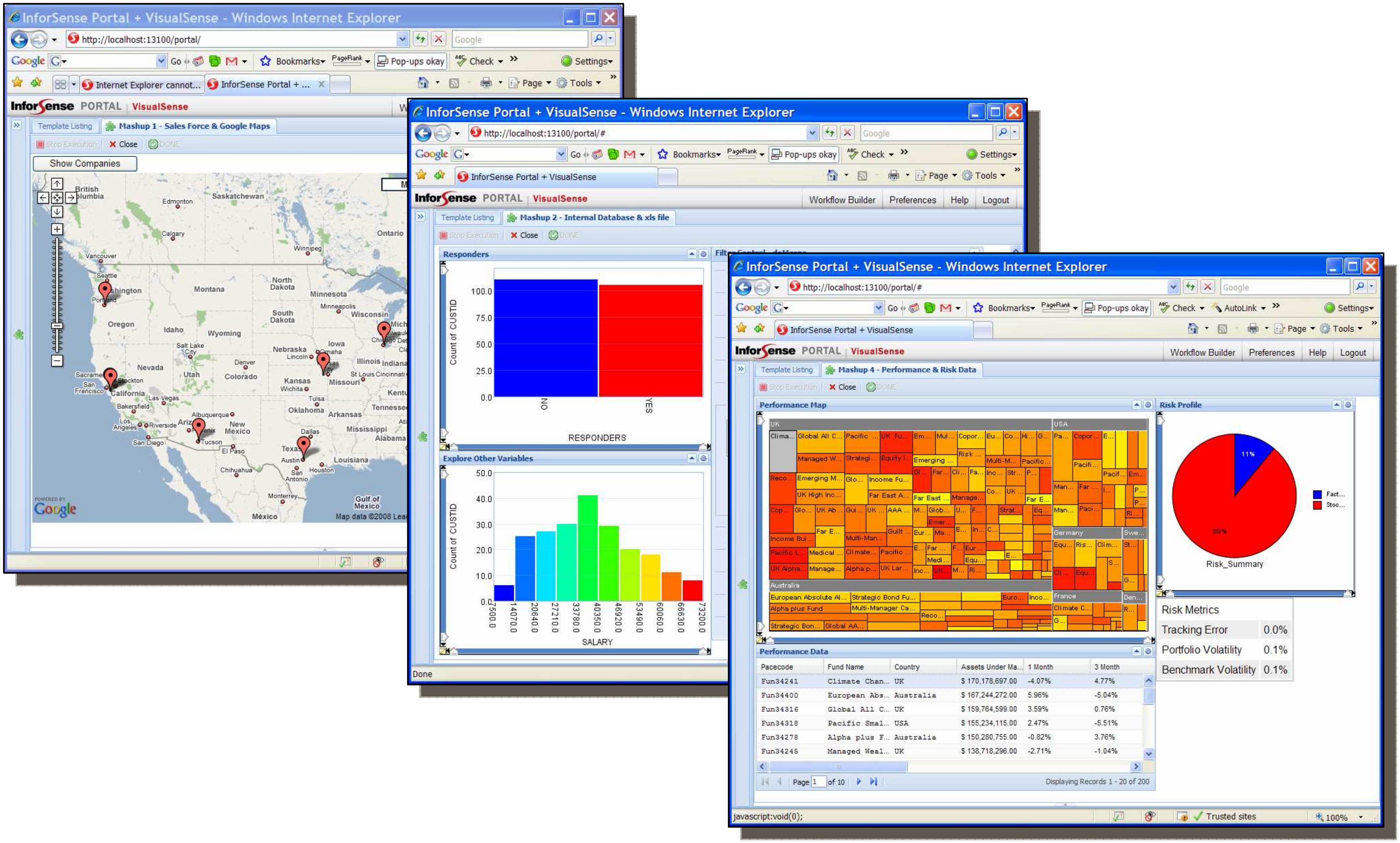


2) Data sources can be both SaaS or on-premise to create middleware or mash up applications

4) Finished application deployed to Cloud Server

Imperial College London

InforSense SaaS Platform – BI Cloud Application Presentation

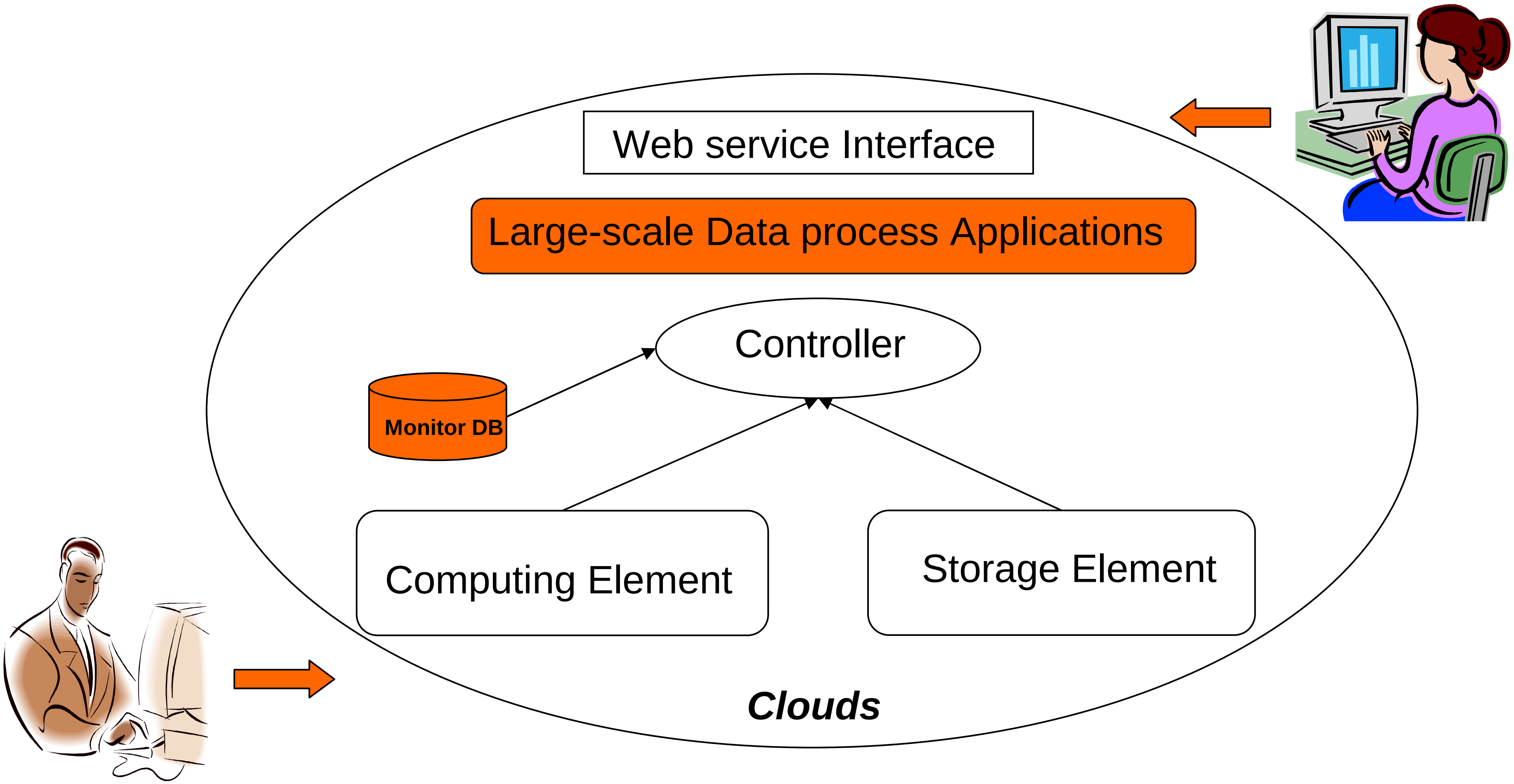


Imperial College London

Goals for Discovery Cloud (DC)

- Concrete hands-on research in cloud computing: Investigating cloud computing mechanisms for service provisioning, scheduling, SLA formulation, hypervisor portability and feature enhancement, etc.
- Application research of cloud computing
  - Applicability of cloud computing for scientific computing and scientific data analysis
- Investigating building large scale cloud computing system using open source software
- Education use

An Abstract Model of DC

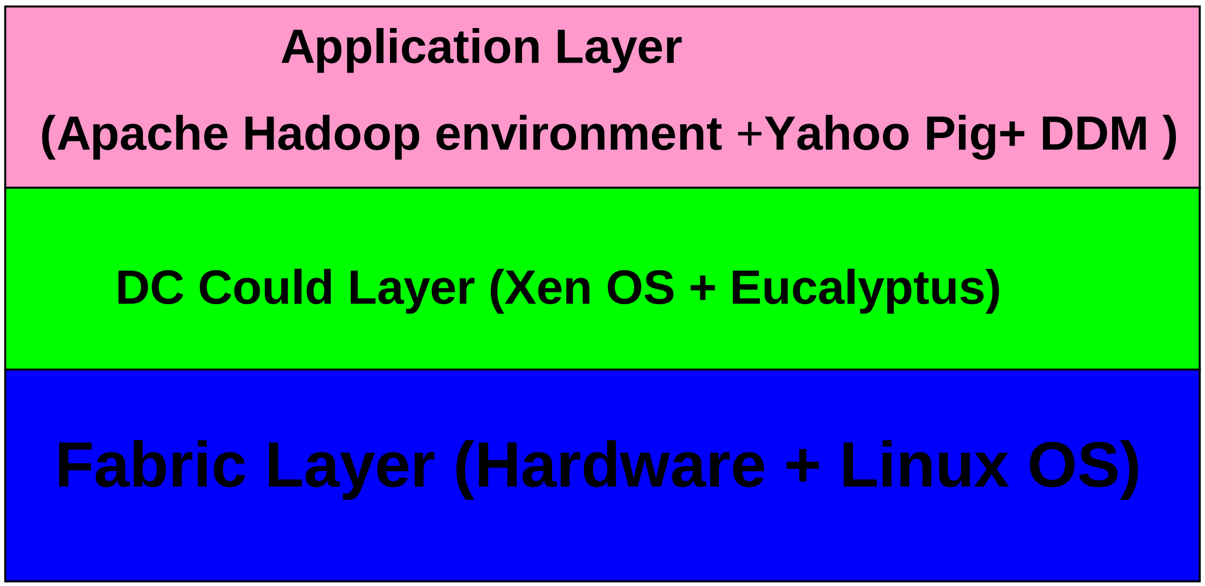


Imperial College London

Outline

- Overview of Discovery Science Clouds (DSC)
- DSC Open Source Cloud-ware
  - Ubuntu + Xen
  - Eucalyptus
- Applications on DSC
  - Apache Hadoop
  - Running Pig
- DSC testbed
  - Installation, configuration, and deployment
- Lessons Learnt and Remarks

Layered Structure of Discovery Cloud



Overview of Discovery Cloud (DC)

Application Layer

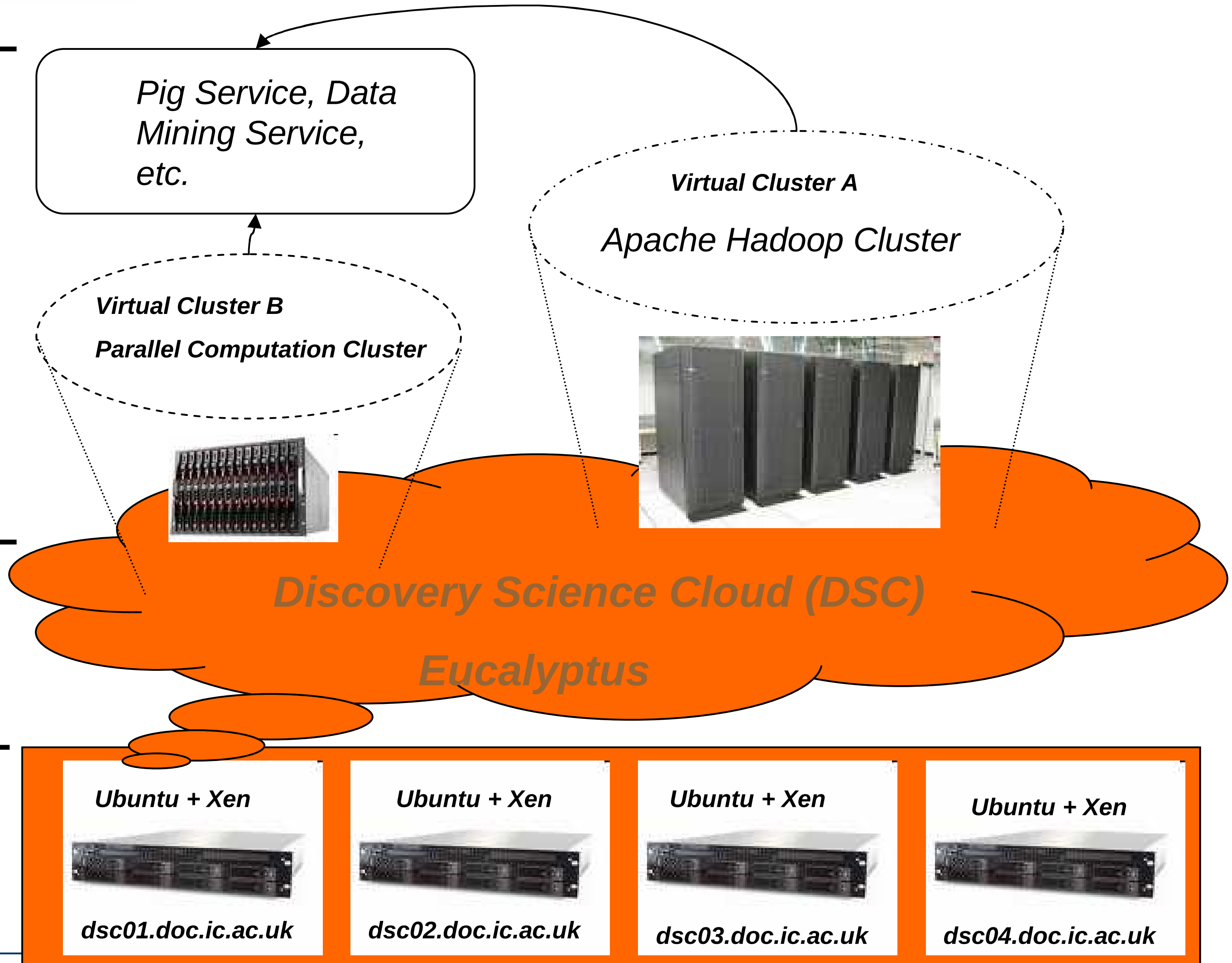
(Applications)

Virtualisation Layer

(Middleware)

Fabric Layer

(Hardware)



Applications on DSC

- Hadoop + Pig
  - Hadoop for providing Map/Reduce-based data parallelism
  - Pig for declarative coding for analyzing large data sets
  - Operations in Pig are extended to support data mining
- Enabling high performance mining for large-scale data process
  - Parallelisation of data mining algorithms are implemented through Map/reduce model
- Mapping dynamic data mining models to the could infrastructure to deal with real time analytics management

- Three stages:
  - Fabric Installation
  - Cloud middleware
  - Application environments

- Step-by-step:
  - Physical machines
  - OS
  - Middleware
  - Application environments
  - Applications

Step by step (1)

- Basic Linux OS:
  - Install Ubuntu 8.04
- Install Xen
  - *apt-get install ubuntu-xen-server*
- *vi /etc/modules*
  - Add one line: loop max_loop=64
- *vi /etc/xen-tools/xen-tools.conf*
  - Modify accordingly
- Reboot

Step by step (2)

- Create Virtual Machines (*domU*)
  - *xen-create-image --hostname=dsc01.doc.ic.ac.uk --size=2Gb --swap=256Mb --ide |*
    --ip=192.x.x.101 --netmask=255.255.255.0 --gateway=192.x.x.1 --force |
    --dir=/home/xen --memory=64Mb --kernel=/boot/vmlinuz-2.6.24-16-xen |
    --initrd=/boot/initrd.img-2.6.24-16-xen --install-method=debootstrap --dist=hardy |
    --mirror=http://archive.ubuntu.com/ubuntu/ --passwd

- Check log file
  - */var/log/xen-tools/dsc01.doc.ic.ac.uk.log*

Step by step (3)

- Cat config file
  - *cat /etc/xen/dsc01.doc.ic.ac.uk.cfg*
- Start the virtual machine
  - *xm create /etc/xen/dsc01.doc.ic.ac.uk.cfg*
  - *xm console dsc01.doc.ic.ac.uk*
- Check vm
  - xm list
- Login
  - ssh root@*dsc01.doc.ic.ac.uk*
- Shutdown
  - *xm shutdown dsc01.doc.ic.ac.uk*

Step by step (4)

- Setup other three machines following same procedure
  - dsc02, dsc03, and dsc04
- Check Xen v-machines
  - xm list

Ubuntu OS+Xen done!

Go to *Eucalyptus setup!*

Step by step (5)

- ***Eucalyptus setup!***
- Download the appropriate installation files from http://open.eucalyptus.com/downloads
- Prerequisites
  - synchronized all machines (a NTP server)
  - Open ports needed (iptables)
    - ports 8443, 8773, 8774, 8775 (node)
  - configured with a bridge as the primary interface
    - dsc02, dsc03, dsc04
    - auto br0
    - iface br0 inet dhcp
    - bridge_ports all

Step by step (6)

- Install master node (dsc01)
  - the cloud controller (-cloud package)
  - the cluster controller (-cc package)

- Install three nodes (dsc02, dsc03, dsc04)
  - the node controller (-nc package)

- To connect the Eucalyptus components together
  - $EUCALYPTUS/usr/sbin/euca_conf -addcluster <clustername> <clusterhost> $EUCALYPTUS/etc/eucalyptus/eucalyptus.conf
  - $EUCALYPTUS/usr/sbin/euca_conf -addnode <nodehost> $EUCALYPTUS/etc/eucalyptus/eucalyptus.conf

- Starting Eucalyptus
  - Master node (dsc01)
    - o $EUCALYPTUS/etc/init.d/eucalyptus-cloud start
    - o $EUCALYPTUS/etc/init.d/eucalyptus-cc start
  - Slave nodes (dsc02, dsc03, dsc04)
    - o $EUCALYPTUS/etc/init.d/eucalyptus-nc start
- Configuring eucalyptus
  - admin portal: https://localhost:8443

Two steps for setting-up Hadoop

- Create a v-Cluster for Apache Hadoop
  - Apache Hadoop Cluster setup
  - A Virtual Cluster managed by Eucalyptus

- Install/config Apache Hadoop
  - Install Hadoop on the v-Cluster
  - Configure Hadoop
  - Start Hadoop

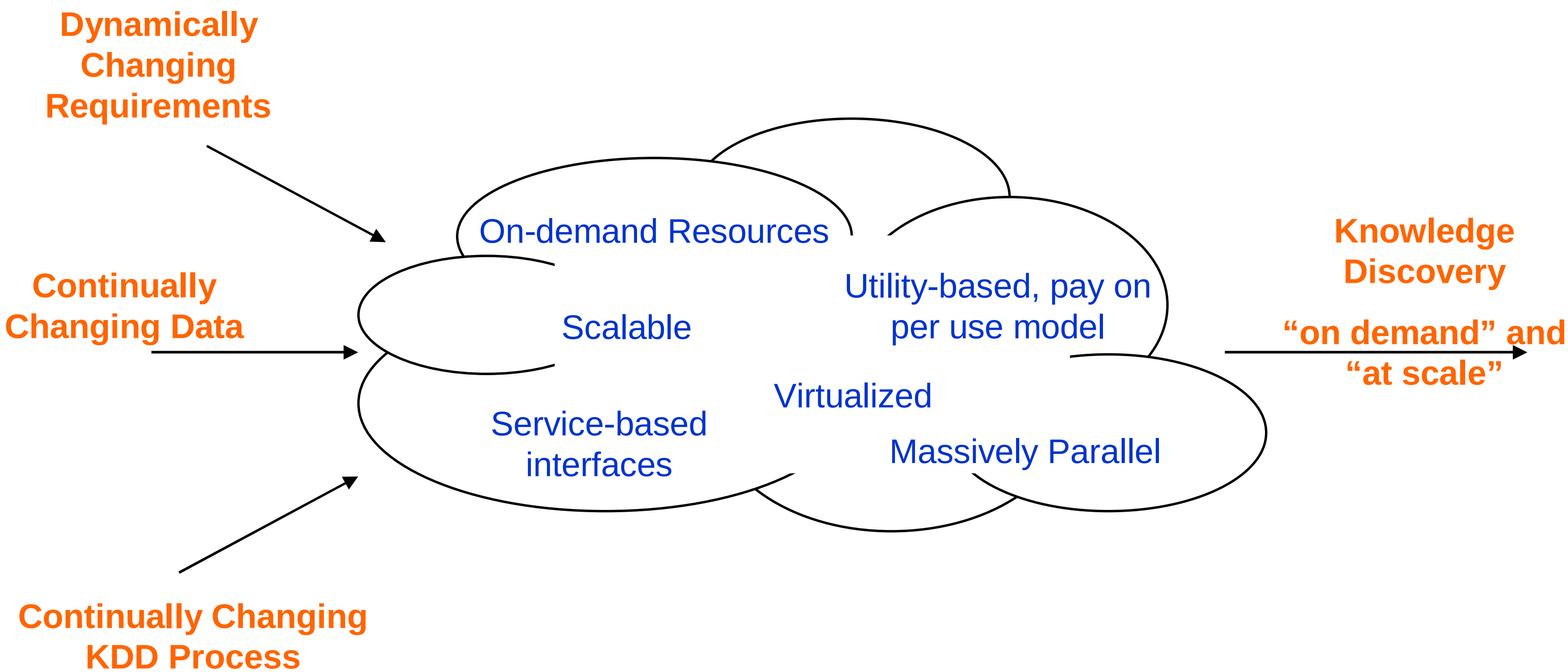
- Create an V-Cluster for Apache Hadoop
  - Adding Images
    - ec2-bundle-image -i <kernel file> --kernel true
    - ec2-upload-bundle -b <kernel bucket> -m /tmp/<kernel file>.manifest.xml
    - ec2-register <kernel-bucket>/<kernel file>.manifest.xml
  - Associating kernels and ramdisks with instances
    - ec2-bundle-image -i <vm image file> --kernel <eki-XXXXXXXX> --ramdisk <eri-XXXXXXXX>

Step-by-step (2)

- Install/config Apache Hadoop
  - download software: http://hadoop.apache.org/core/docs/current/quickstart.html#Download
  - install the packages on all nodes of v-Cluster.
  - edit three config files: conf/core-site.xml; conf/hdfs-site.xml; conf/mapred-site.xml

- Startup Hadoop
  - bin/hadoop namenode -format
  - bin/start-dfs.sh
  - bin/start-mapred.sh

Now, We are in the position to build a Knowledge Discovery Cloud: A Buzzword or a Reality



- MESSAGE aims to build a dynamic monitoring system for urban pollution based on real time analysis of wireless mobile sensors
- MESSAGE has developed an e-Science architecture and the associated infrastructure to support:

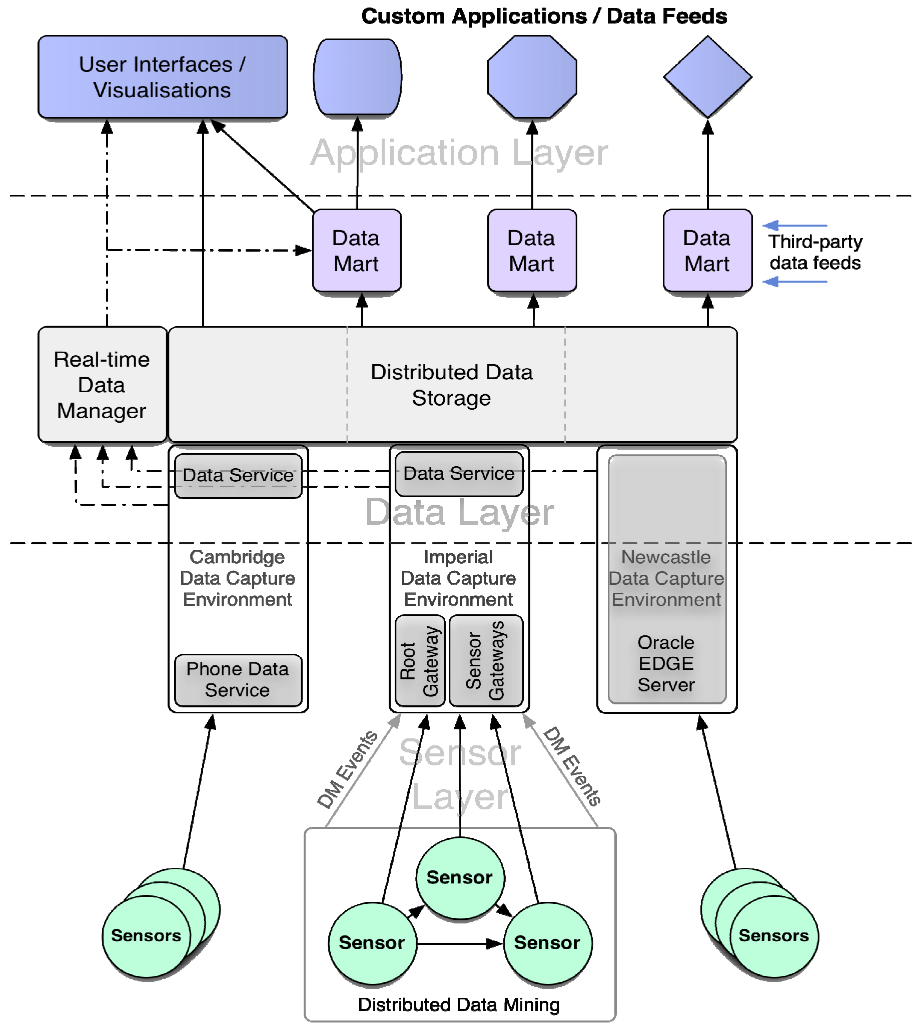
- Flexibility
  - Heterogeneous sensors
  - Heterogeneous databases
- Scalability
  - Connect new sensors, data stores, applications
  - Opportunity to use on-demand computing platforms
  - Different data pre/post-processing approaches
- Interoperability
  - Interoperable data transmission formats – JSON, XML
  - Federated database infrastructure

Is this Architecture Good Enough?

Application Layer

Data Layer

Sensor Layer



A New Analytics Infrastructure is Needed

- The Analysis needs to be in real time, on dynamically available information
- Global information is not always needed: federating a central data base is not suitable
- Local models contribute to the global knowledge
- Model management is more important than data management
- Model life cycle becomes essential: Long life model vs short life data
- On demand evolution of knowledge:
  - Model level scale-up----Learning a better model : hotspots need to be investigated in details
  - Model level scale down---using less sensors for normal situation

In Fact, the Information Infrastructure Can Be Easily Done in a Cloud Way

- Applications, Sensors, Data Repositories as Services
- Computing via Amazon EC2
- Portals as Gadgets
- Metadata by tagging
- Data sharing as in YouTube mashed up with Google Maps
- Alerts by RSS
- Virtual Organizations via Social Networking
- Performance by multicore
- Interfaces via iPhone, Blackberry etc.

- The main challenge is to have a Cloud Infrastructure for Analysis !

Dynamic Data Mining Model : Analysis on Demand, at Scale

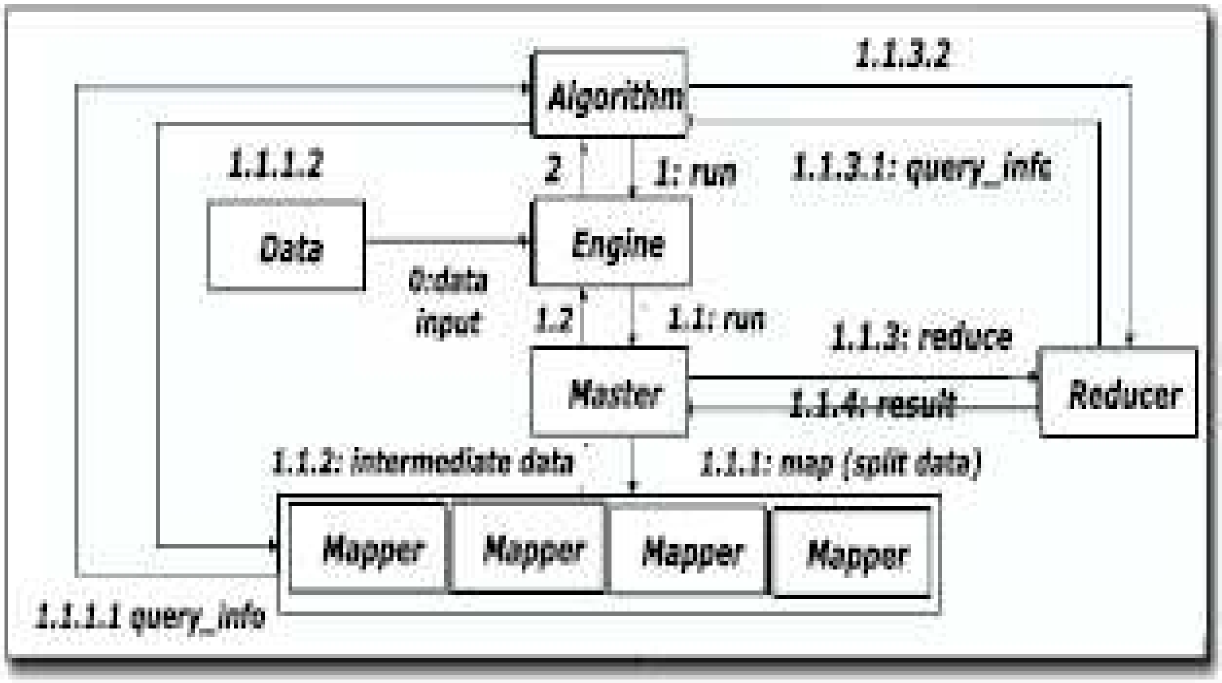
- Dynamic Data Mining Model : An Paradigm of Analysis Dynamically Changing Information with Different Degree of Interests

- New data are dynamically acquired which may change the models learned from historical information

- Models are managed with its life cycle:
  - Model created
  - Model selected for use
  - Model evaluated
  - Model updated
  - Model combined

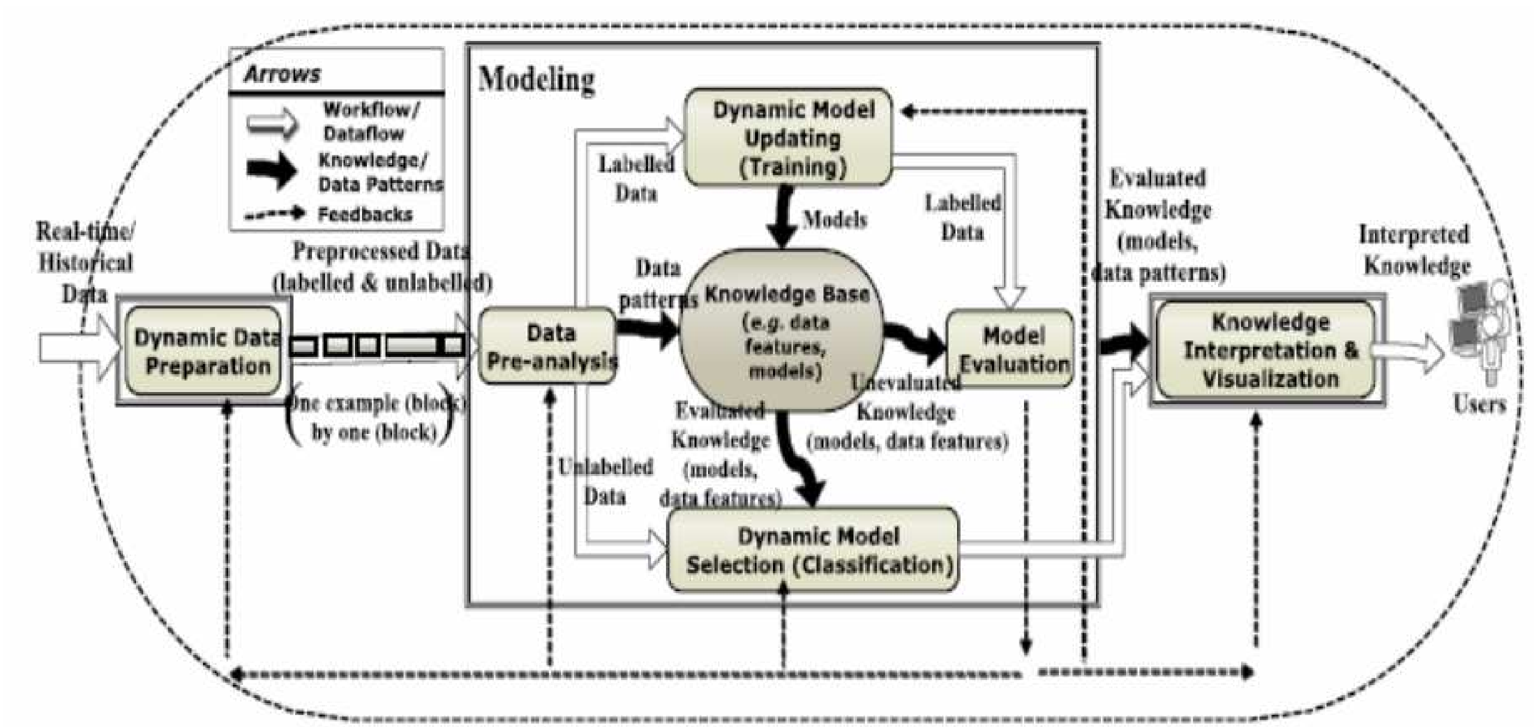
- On demand updating : data driven or business requirement
- At scale learning : A MAP/REDUCE based execution model

Data Mining Algorithms with Map/Reduce Model

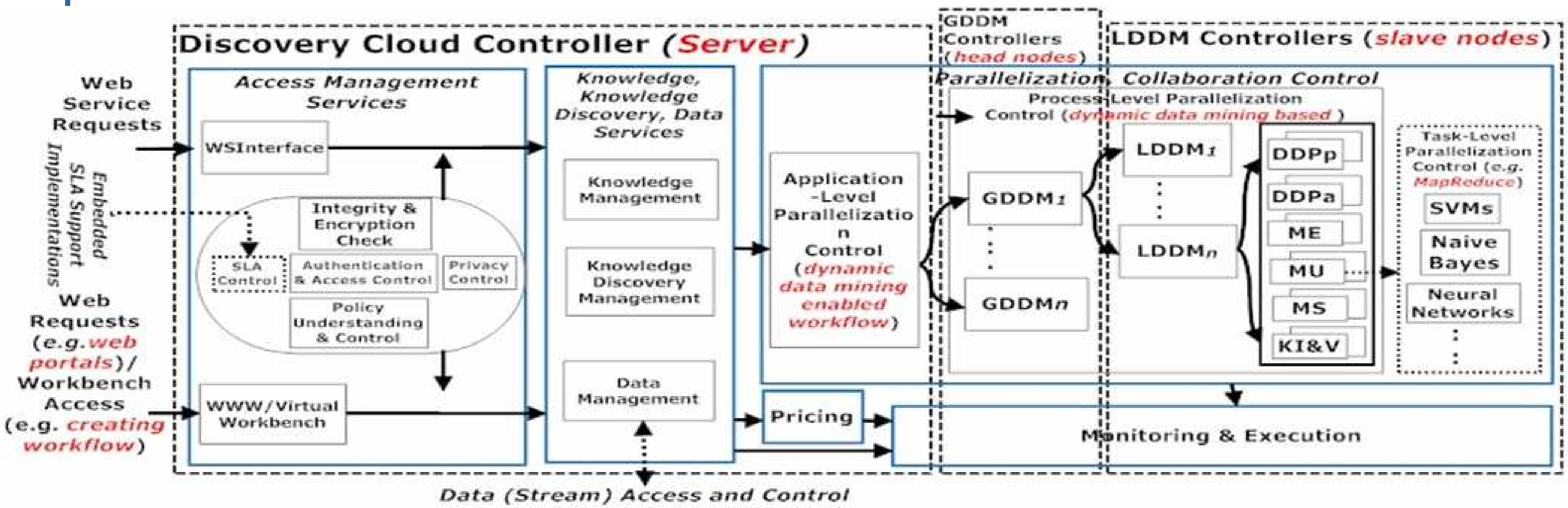


Map-reduce for Machine Learning on Multicore :
Cheng-Tao Chu et.al,
CS Department Stanford University

➢Dynamic Data Mining Model Illustration:

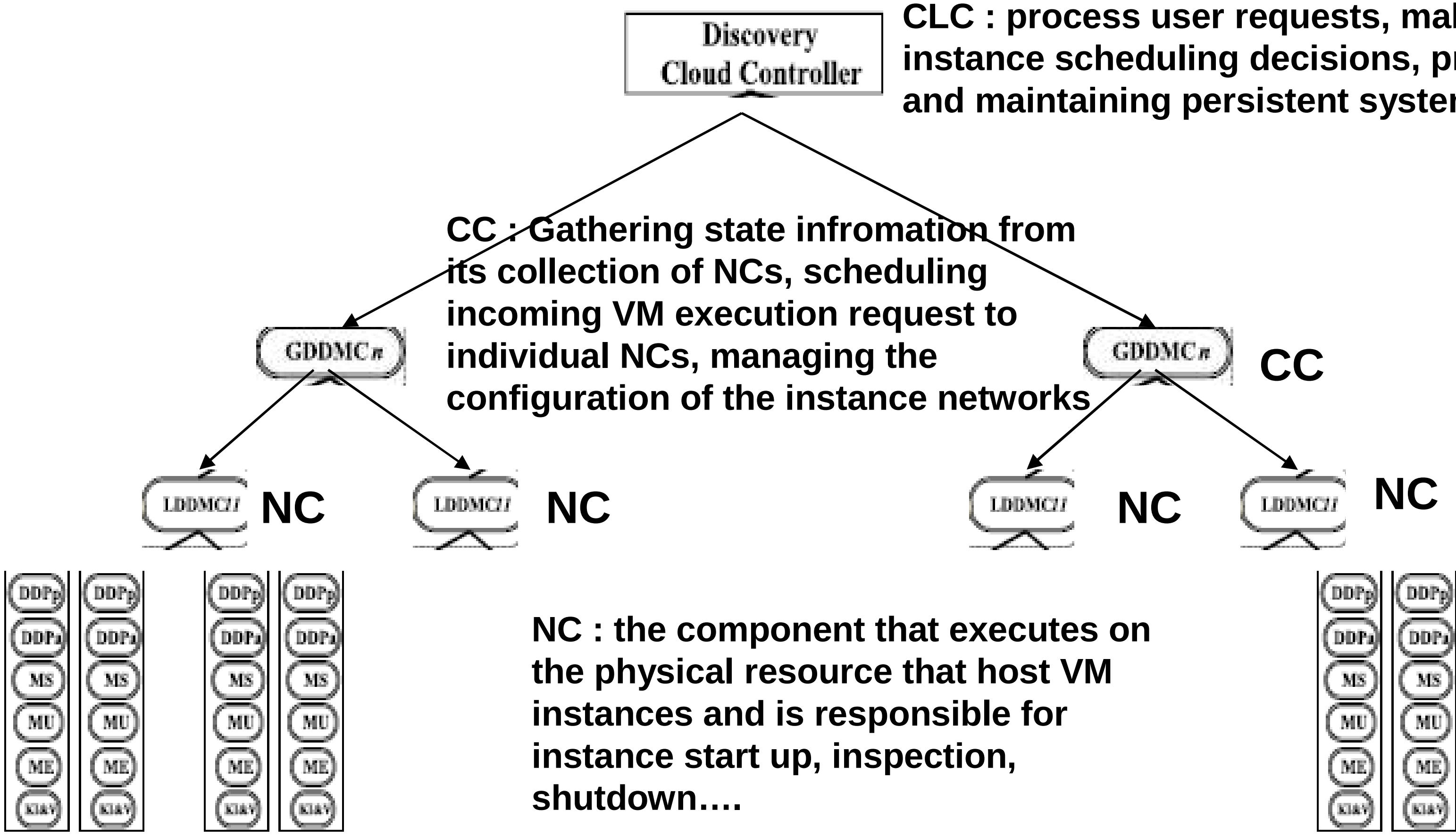


Dynamic data preparation (DDPp), Dynamic data pre-analysis (DDPa),
Dynamic model selection (MS), Dynamic model updating (MU),
Model evaluation (ME), Knowledge interpretation and visualization (KI&V)



- ➢ DCC: User entry point, KM/DB persistent systems interface, high level problem decomposition
- ➢ GDDM: General Map/Reduce Style Organiser for DM
- ➢ LDDM : DM on the local data

Map the Dynamic Data Mining Model to DC



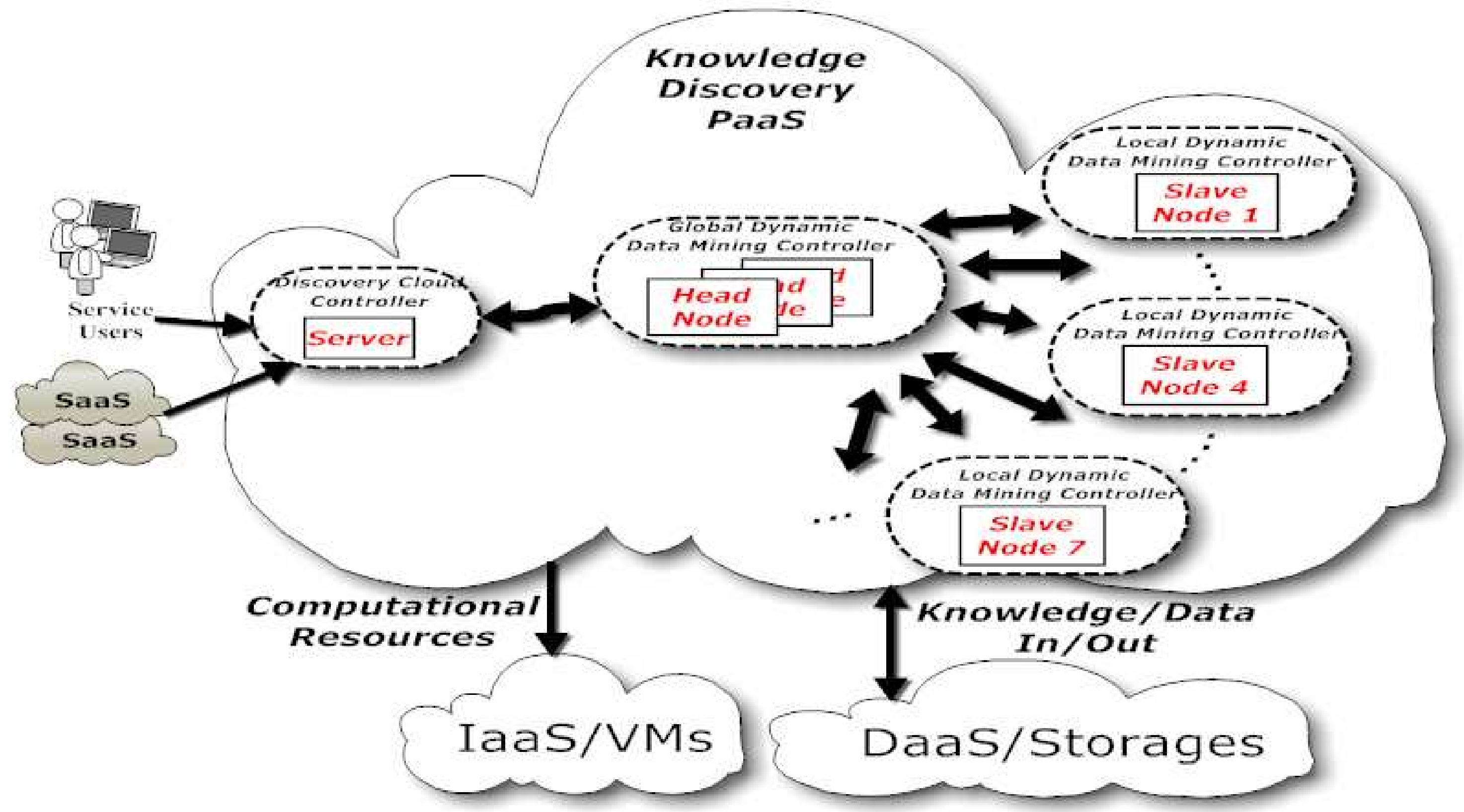
CLC : process user requests, making high level VM instance scheduling decisions, processing SLAs and maintaining persistent system

CC : Gathering state infromation from its collection of NCs, scheduling incoming VM execution request to individual NCs, managing the configuration of the instance networks

NC : the component that executes on the physical resource that host VM instances and is responsible for instance start up, inspection, shutdown….

A PaaS platform to build a Discovery Cloud

- Abstracting commonly used DM functionality as services
- Each service has a Map/Reduce implementation
- Use workflow to compose services
- Deploy composed services based on the DDM/Cloud mapping



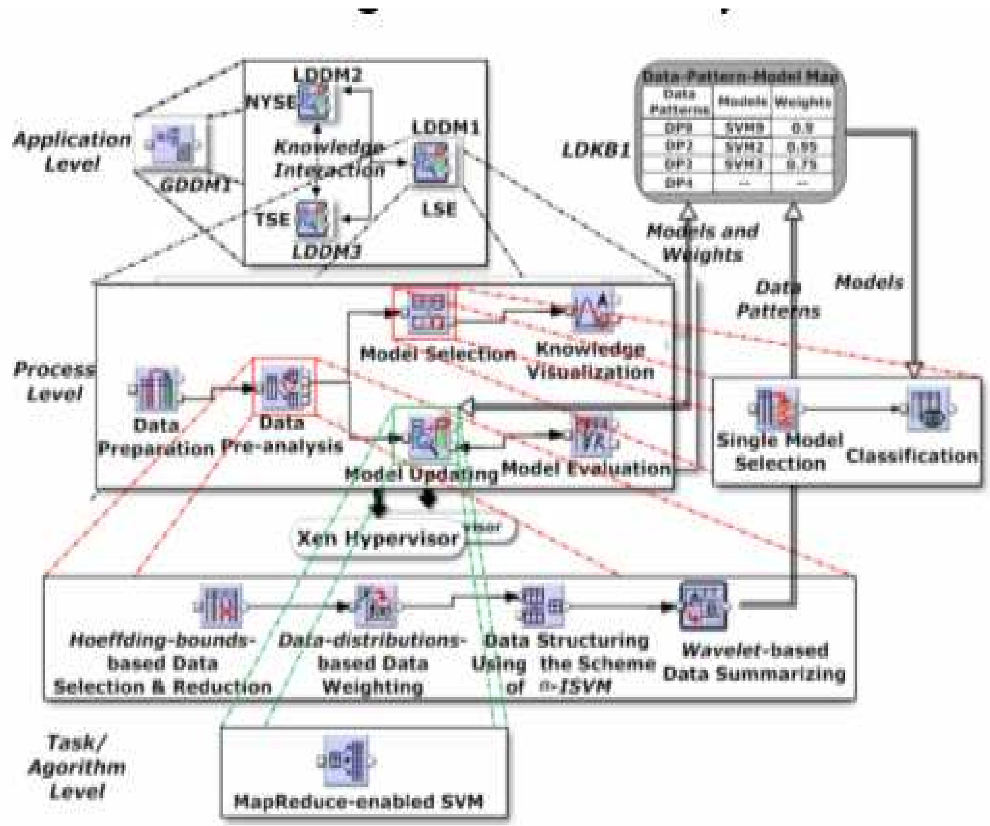
Example

Distributed Real-time stock price prediction

When and how much to buy or sell stocks through predicting the trends and extents of intra-day stock price movements within a changeable short time interval (e.g. less than every 10 minutes) on London Stock Exchange (LSE).

The next time-interval (i.e. a time window) price of a stock depends on not only various economic indicators but also political and international events.

Influenced by some other major stock markets in the world such as the New York Stock Exchange (NYSE) and the Tokyo Stock Exchange (TSE)



- On Demand :
  - New source of information
  - New criteria of models
  - New business need: e.x. a portfolio design
- At Scale :
  - Data size
  - Time interval
  - Precision of the models
- Delivered as a service:
  - Not only the data (DaaS)
  - Not only the model (KaaS)
  - But also the process (PaaS)

Imperial College London

Cloud Computing: New Challenges and Opportunities to Computer Science (1)

- **Towards to a mathematics of distributed computing**: virtualization provides the uniformity to the resource. Thus, a compute instance can be associated with a matrix to abstract its resource and cost information. Resource scheduling for services provision is then becoming an optimization issue, i.e. constraint programming. Moreover, if we combine the pricing model with the resource optimisation, the whole theory of Computational utility theory can be developed just like we did in the past 20 years for the power grid. Further more, an economics of cloud computing can be developed when the computing resource becomes a trading commodity.

- **Towards to a social knowledge economy**: Cloud computing provides a global platform for exploring individual creativity with social collaboration. It breaks the enterprise barriers in building knowledge products. The Apple application store business model is the first example that a new software production model is becoming a reality. Such a model of global collaborative knowledge production and trading is not only an revolution in software industry but also a great opportunity for computer science. A new software engineering principle need to be established for such a new industry

- **Towards to a social knowledge economy**: Cloud computing provides a global platform for exploring individual creativity with social collaboration. It breaks the enterprise barriers in building knowledge products. The Apple application store business model is the first example that a new software production model is becoming a reality. Such a model of global collaborative knowledge production and trading is not only an revolution in software industry but also a great opportunity for computer science. A new software engineering principle need to be established for such a new industry

- **Towards programming of the Flat World** : Programming a Cloud is completely different from coding a stand alone computer system, or a networked computing system. Programming a Cloud can only be done at the service level. Thus, the declarative style of programming offers a promising paradigm. However, a simple reuse of old technology is not going to work. We need to start with the declarative programming principle but investigate new abstraction mechanisms to realise a flexible, expressive and adoptable end user programming model for building personal cloud applications.

Projects for tomorrow : Over the Cloud

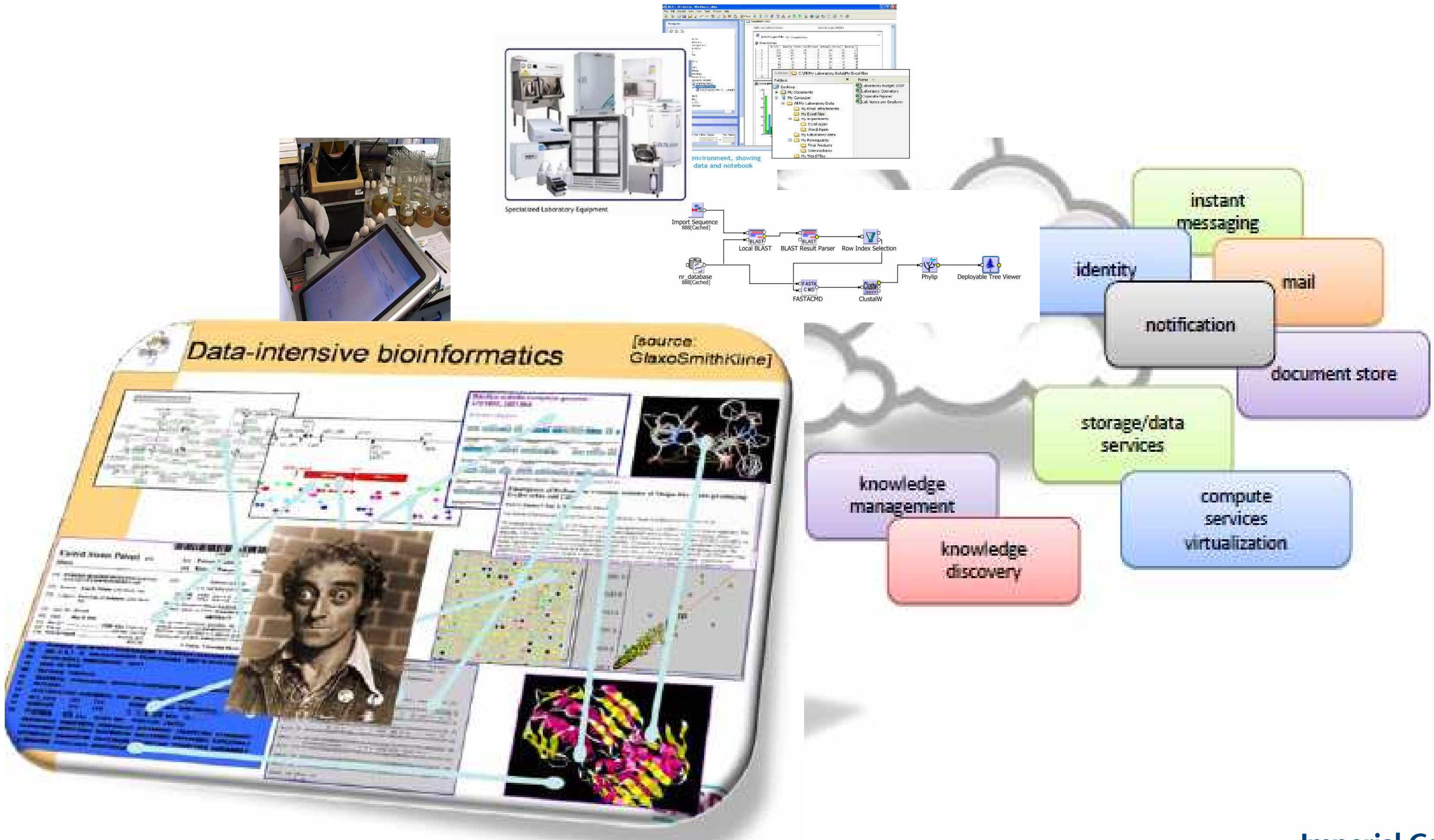
- What can and cannot easily be hosted in a cloud?
- What extensions or modifications are required to support a wider variety of services and applications?
  - Scientific computing
  - Data assimilation
  - Multiplayer gaming
- How can cloud computing be coupled with other distributed software systems and infrastructure?
  - How should clouds and mobile devices (e.g. cell phones) interact?

Projects for tomorrow: Within the Cloud

- Management of services (**groups of VMs**) with inter-connection taking into accoount the relationship and placement constraints within and across sites
- • **Dynamic and scalable management of VMs** and physical resources and **elasticity support** to meet variations in service workload—An optimization problem.
- • **Advanced placement algorithms** with policies for SLA commitment…
- • Architectures for **federation of sites and** heuristics for **capacity**

Projects for Tomorrow: Make a Cloud

Go to the computer centre of your university, convince them to work with you to build a campus Cloud so that the instrument, experiment information, literatures, computing resource are all come be provided as shared resources



Imperial College London

打造中国云: 为人民服务

- Cloud computing is based on the concept of centralised provisioning of resources
- Cloud computing transfers the IT industry from product manufacturing industry into utility service industry
- Cloud computing changes the focus of software from one for all to one for each
- Personalised computing and computing as a service should be one of the key focuses in our future research

- 为人民计算+计算即服务 = 为人民服务

天高云淡，

望断南飞雁。

不到长城非好汉，

屈指行程二万。

六盘山上高峰，

红旗漫卷西风。

今日长缨在手，

何时缚住苍龙。