Platform as a Service (PaaS)

Week 4

School of Software

Faculty of Engineering and Information Technology

University of Technology Sydney

Learning Objectives

- Understand Computing Platform
- Define Platform as a Service (PaaS)
- Identify PaaS Components
- PaaS Environment 1 - Google App Engine
  - Architecture and working of Google App Engine
- PaaS Environment 2 - Force.com
  - Multi-tenancy and working of Force.com
  - Demonstration of Force.com
- Conclusion

Computing Platform

- What is a computing platform?
  - Numerous formal definitions
  - A combination of *hardware resources* and *software resources* used for application development
  - Computing platforms can be used
    - to *build* new software applications
    - to *enhance* existing software applications
    - to *execute* software applications

Computing Platform

- Software Frameworks
  - .NET Framework
  - Java Development Kit (JDK)….
- Integrated Development Environments (IDE's) and API's to assist the developer (such as Eclipse)
- Operating System
- The above need hardware resources to run:
  - CPU processing power
  - Memory (Volatile, and Non-Volatile memory)….. etc..

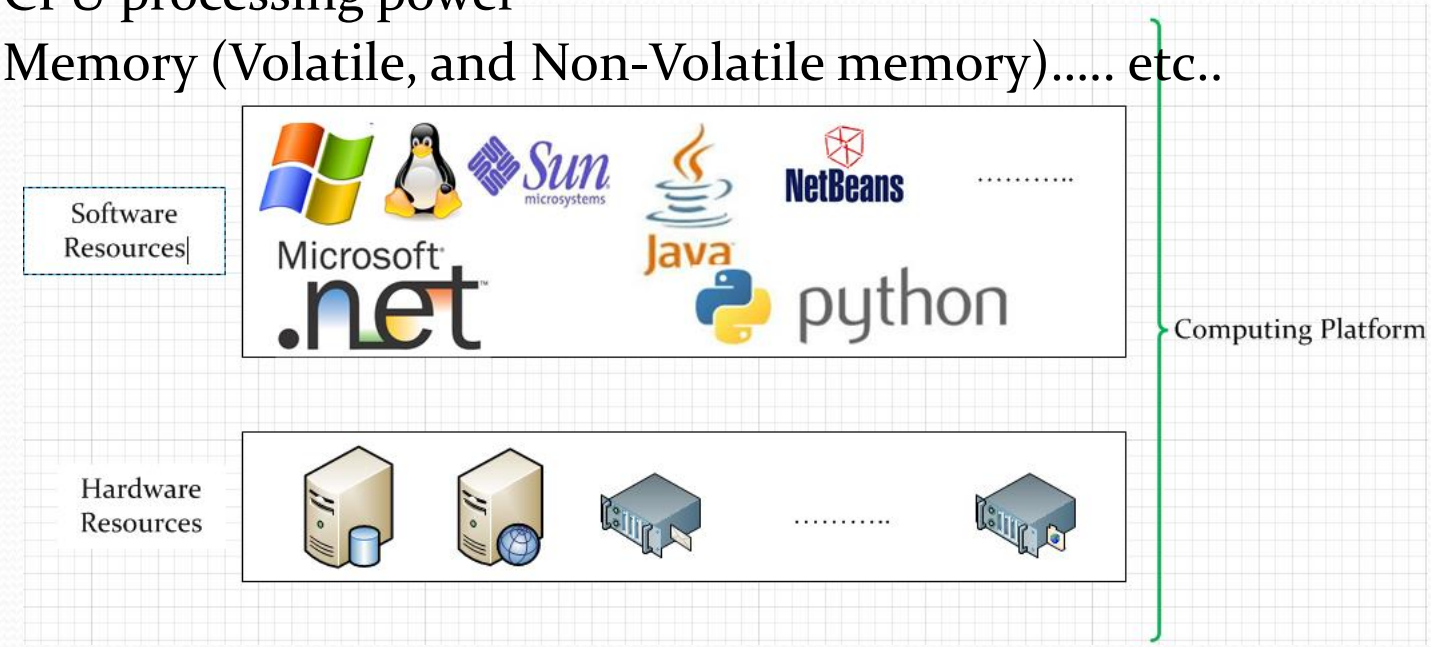


Figure: Computing Platform

"Computing Platform"-as-a-Service

- Computing Platforms, provided over the internet, to cloud consumers, via a web interface
  - Commonly referred to as Platform as a Service (PaaS)
- The NIST Definition of PaaS

  "*The capability provided to the consumer is to create and deploy applications using programming languages, libraries, and tools supported by the provider*". (Mell and Grance 2011)

PaaS Implications to Cloud Consumer

- Implications of PaaS (to the cloud consumer):
  - The consumer *does not* need to pay the up-front capital expenses for the computing platform (software resources and hardware resources)
  - The consumer is *exposed to the Software Framework* only via an interface
    - The user have visibility over the provided software framework only;
    - The consumer *does not* have any visibility over the underlying cloud infrastructure (including servers or storage)
  - The consumer *is not responsible* for the on-going maintenance and upgrades of the platform
  - The consumer *has control* over the developed application

PaaS Components

- Multiple consumers make use of the same computing platform for software development (Multi-tenancy)

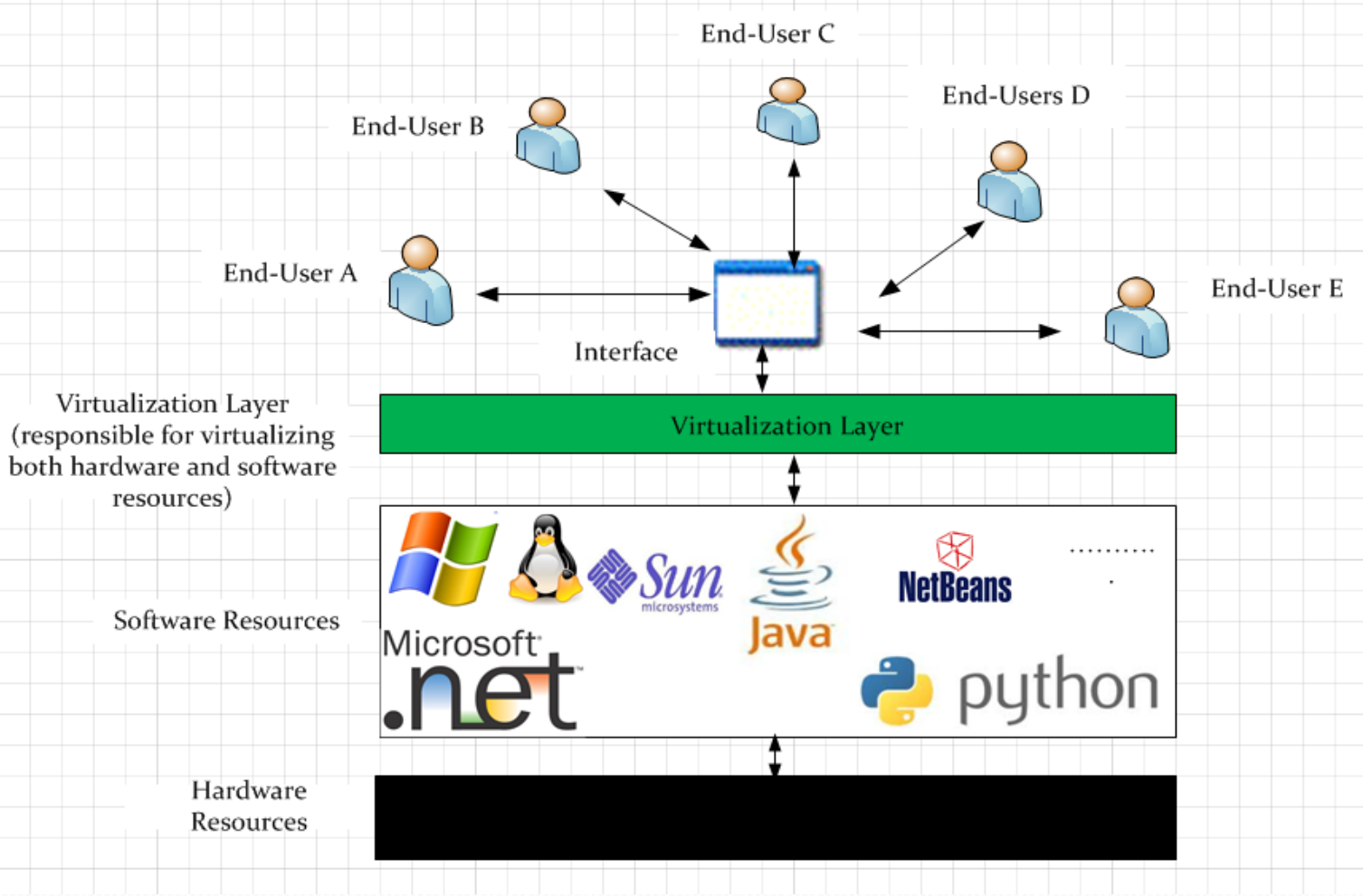


Figure: Multiple users consuming PaaS via an interface

What is provided with PaaS?

- PaaS provides access to all the required resources for building a software application such as:
  - Integrated Development Environments (IDE's);
  - Collaboration tools for distributed application development (such as code merging);
  - Unit testing, stress testing, integration testing, pre-production testing… etc.;
  - Version control for code
  - ……..

Platform as a Service Pricing

- The PaaS consumer (cloud consumer) is charged for the usage of the "computing platform" on
  - The number of lines of code;
  - Number of licences (of the development environment) required or used;
  - Number of applications deployed;
  - Resource usage by the applications deployed;
  - ….

Platform as a Service Implications

- Your code (and the application) and data will reside externally:
  - PaaS provider
  - PaaS provider may in turn be making use of an IaaS provider.
- Accessibility (code, data and application) is governed by the PaaS provider (example: cannot fetch more than 50,000 records in one request)
- Not all languages, development features or run-time features are supported.
  - **Example**: Some PaaS platforms supporting Java SDK do not support in-built Java concurrency constructs
- Problems associated with changing PaaS Provider due to code compatibility between the providers (Vendor lock in)

Platform as a Service Example – Google App Engine (GAE)

- Google App Engine provides a software platform for application development
- Users can develop applications using the provided software platform and deploy them on the Google Cloud
- Preview Version released in April 2008 and came out of the preview version in September 2011
  - Stable Version Release on 11th February 2014
- In Google App Engine, only the provided Software Development Framework is exposed to the user

Google Application Engine (GAE)

- Provides a desktop-client for local application development

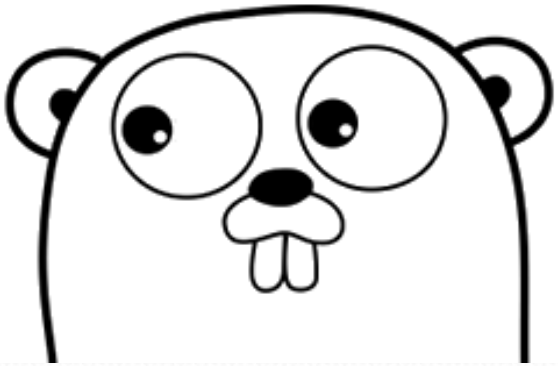
- Originally Python-based environment

- Currently, supports a number of programming languages – PHP, Python, Node.js, .NET and Go

Google Application Engine (GAE)

- From in early 2012, Google App Engine supports application development in "Go" programming language
  - Go has been developed by Google
  - Go supports a number of *concurrency mechanisms* and makes it easy to write programs that leverage *multicore* and *networked machines*

Go is an open source programming environment that makes it easy to build simple, reliable, and efficient software.



Architecture of Google App Engine

- *The infrastructure components of Google Application Engine (GAE) are completely opaque to the PaaS Consumers.*
- *GAE development environment is exposed using an interface*

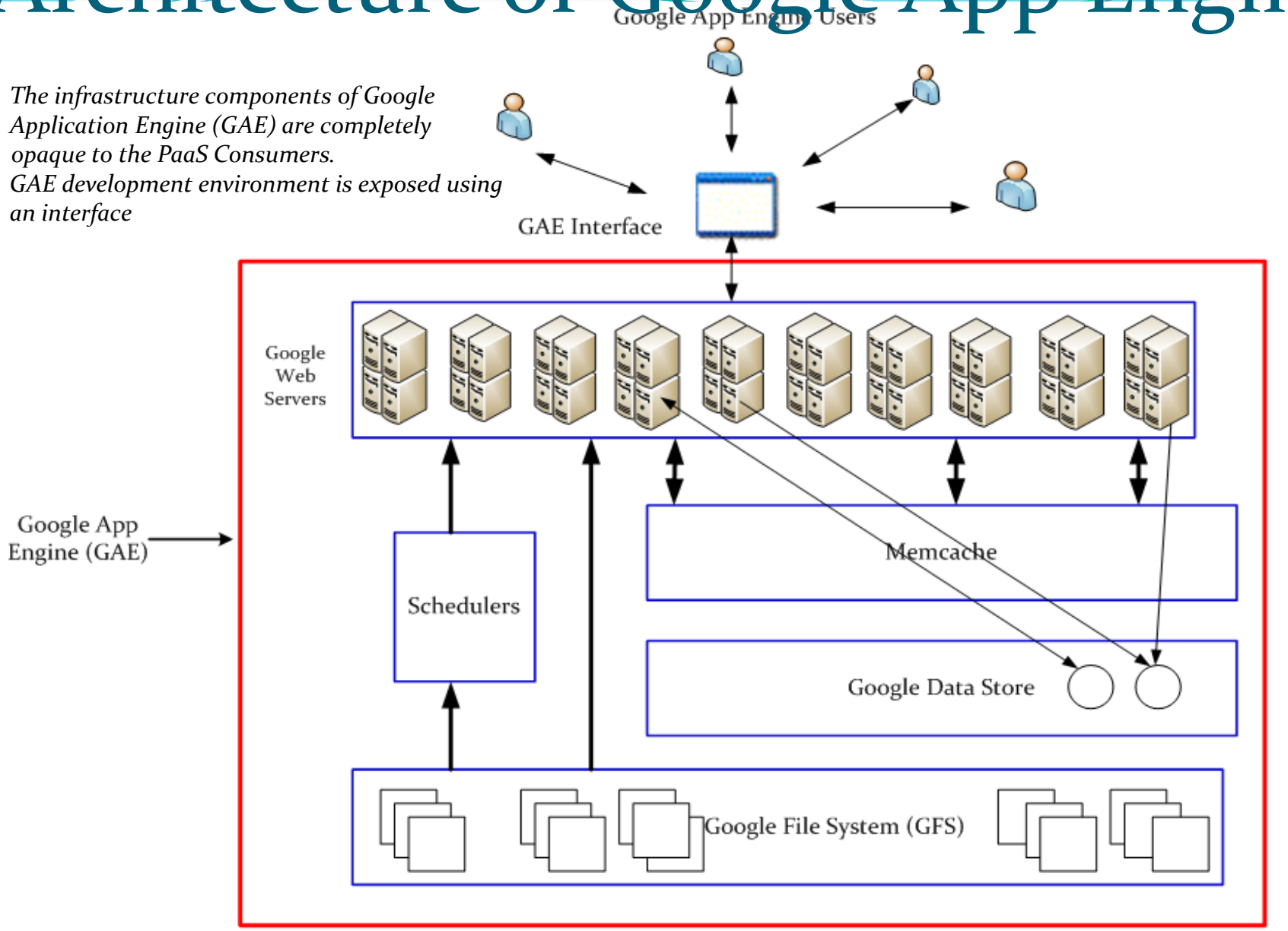
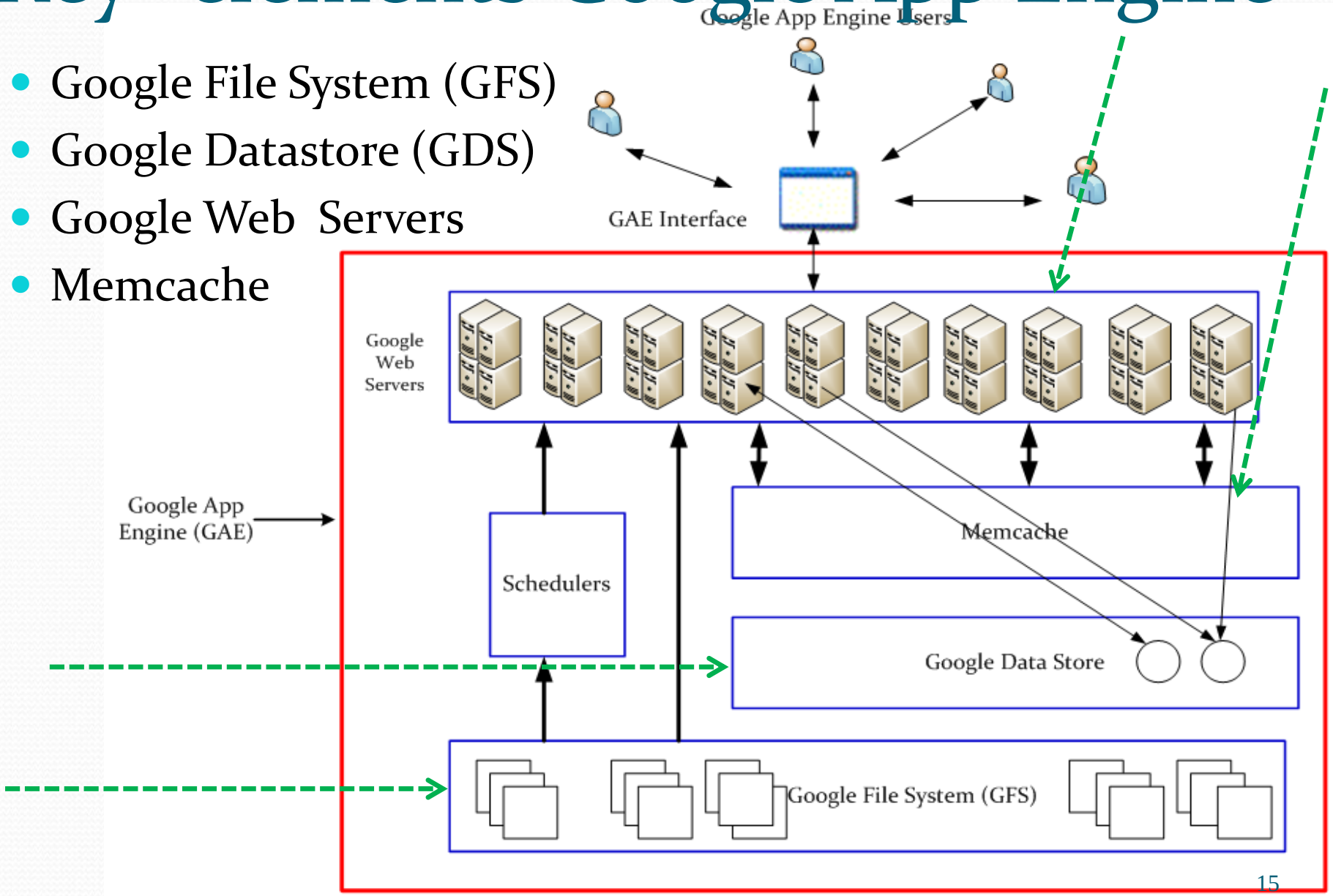


Figure: Architecture of Google App Engine

“Key” elements Google App Engine

- Google File System (GFS)
- Google Datastore (GDS)
- Google Web Servers
- Memcache



Google File System (GFS)

- Proprietary distributed file system developed by Google
- Comprised of hundreds of thousands of servers in Google Data Centre
- Stores data in a fault tolerant manner:
  - Files are stored in chunks (of size 64 MB);
  - The chunks are replicated across a number of servers in GFS
- Used for many Google-based products/services
- In the context of Google Application Engine, users upload code in either Python/Java/Go/.NET in GFS.

Google Datastore (GDS)

- Google Datastore is a robust and scalable storage service used for storing application data (*not application code*)
- Application Data is stored as Data Objects (a.k.a., "Entities")
  - The notion of objects in GDS is different to the traditional notion of objects in Object Orientation (OO)
  - Schema-less in contrast to relational data entities or object in OO
  - Data Objects (which are instances of a given "type") may have different properties (attributes)
- Proprietary language to query Google Datastore (GQL)

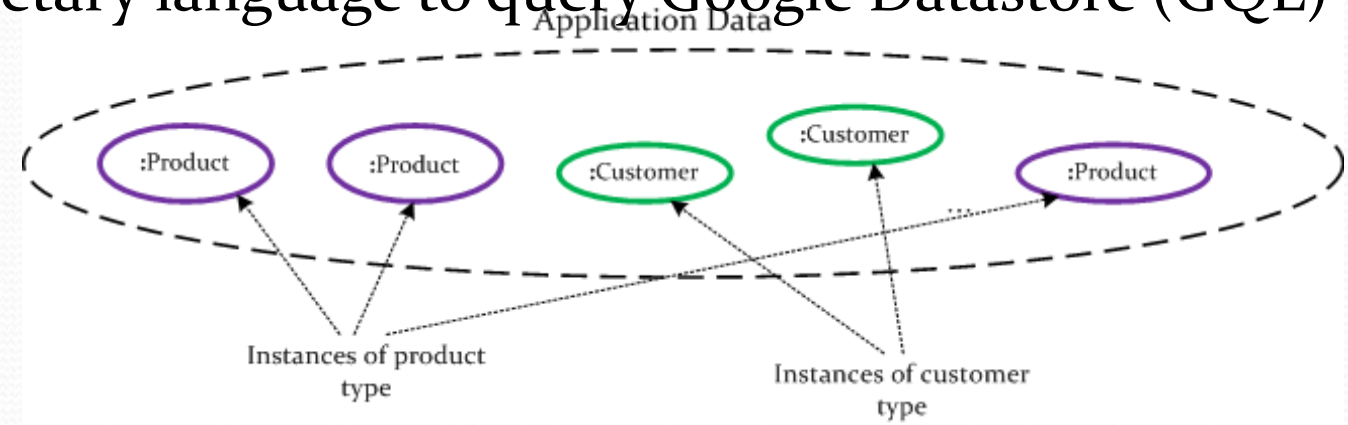


Figure: Example of objects in Google Datastore belonging to an application

Objects in Google Datastore (GDS)

- Fundamental Unit of data storage in GDS
- Each object has data values stored as "attributes"
- Has an immutable unique identifier (key), and zero or more attributes (properties)
  - Unlike traditional relational databases, the Google Datastore doesn't require entities of the same kind to have a consistent property set

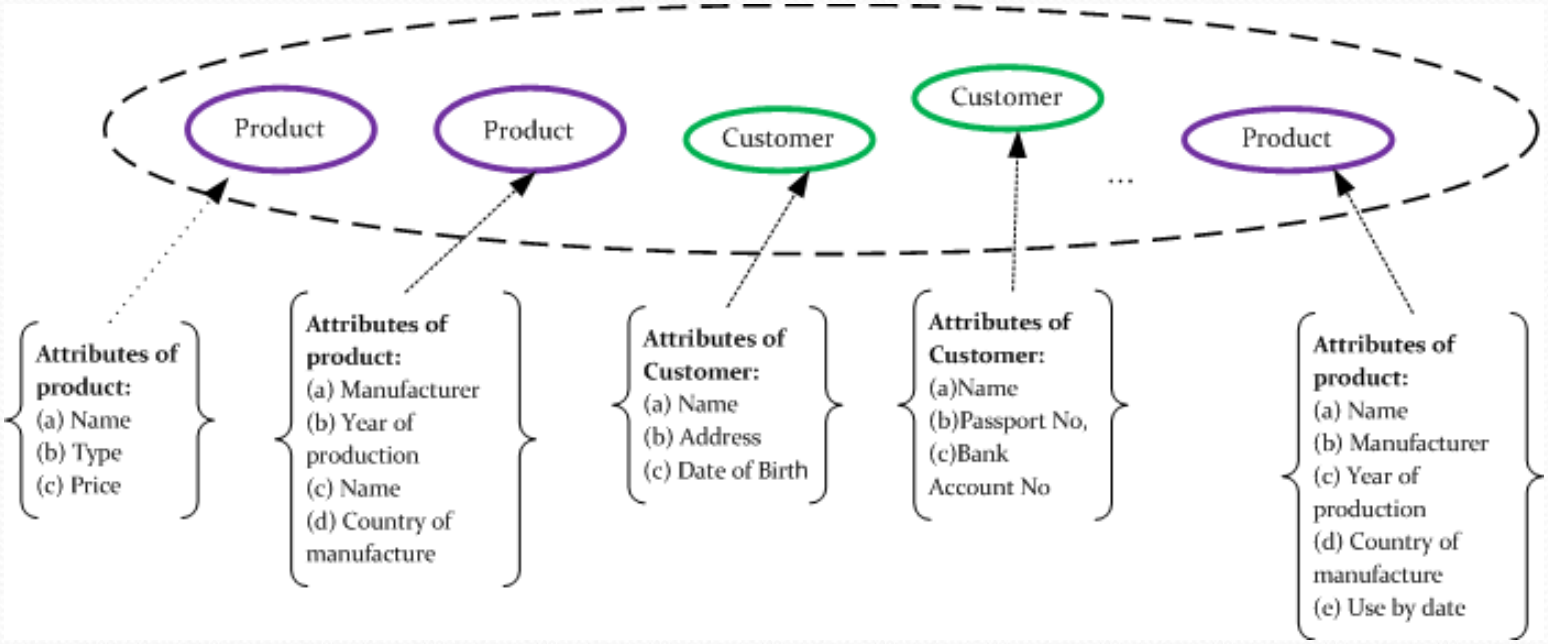


Figure: Example of object's attributes in Google Datastore belonging to an application

Google Cloud SQL

- Google Cloud SQL is a fully-managed database service for relational data integrated within the Google Cloud Platform

- You can select from one of the following two database engines - MySQL or PostgreSQL.

Architecture of Google App Engine

- Google Web Servers (in Google Data Centres)
  - Used to deploy the (user's) applications and make them available over the internet
  - Requests to Google Web Servers are *automatically* load balanced
  - Google Web Servers are located in Google Data Centres
  - They access code and application data to execute applications
- Google Data Centre Statistics
  - Data Centers across the globe
  - A Data Center has 100's of cells
  - A cell holds 1160 servers

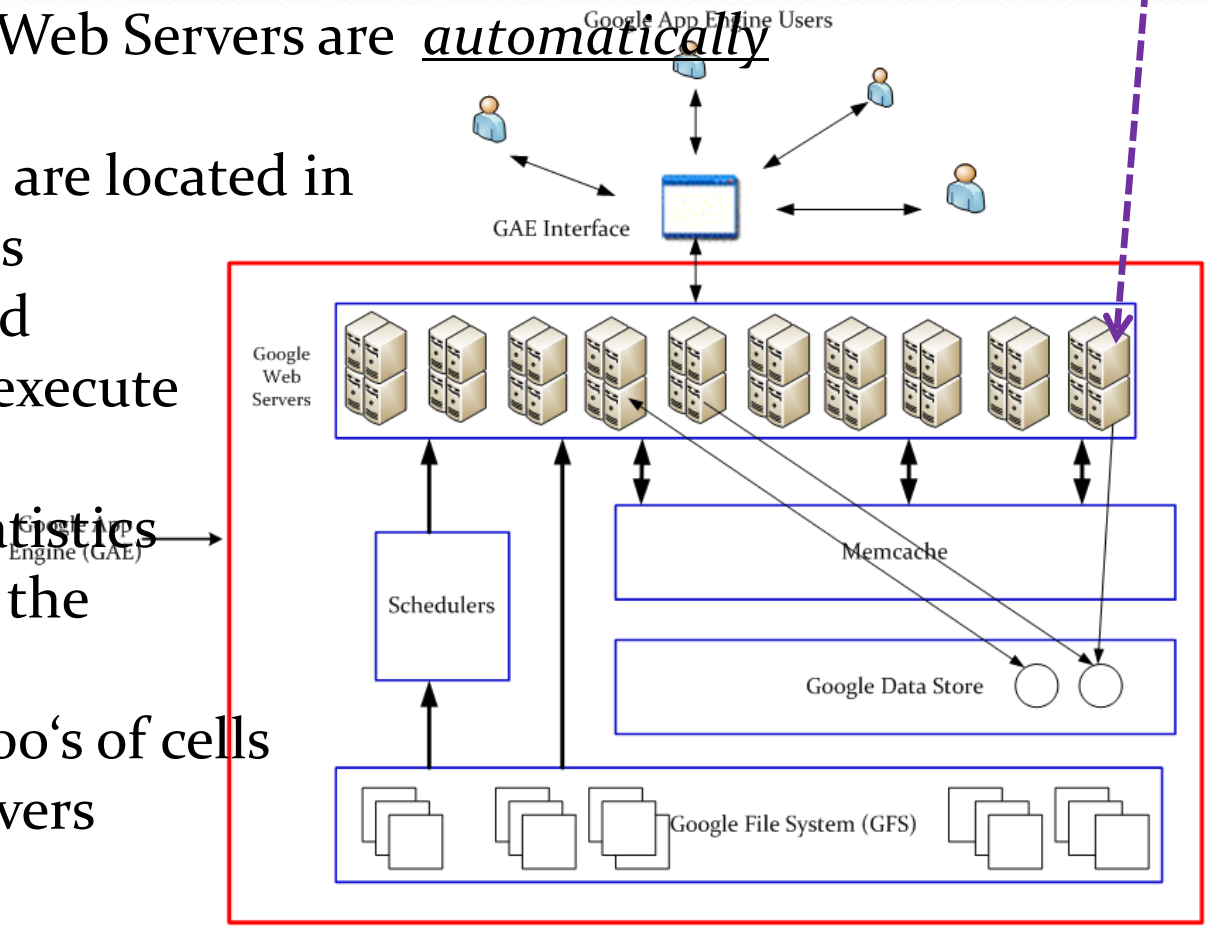


Figure: Google Web Servers in Google App Engine

Architecture of Google App Engine

- Memcache
  - Distributed RAM cache for Google Web Servers
  - "Get" requests to Memcache are five times faster than to Google Datastore
  - "Write" requests to Memcache are ten times faster than to Google Datastore
  - Used to cache application data for easy access across multiple web servers
  - Useful for completing a HTTP Session

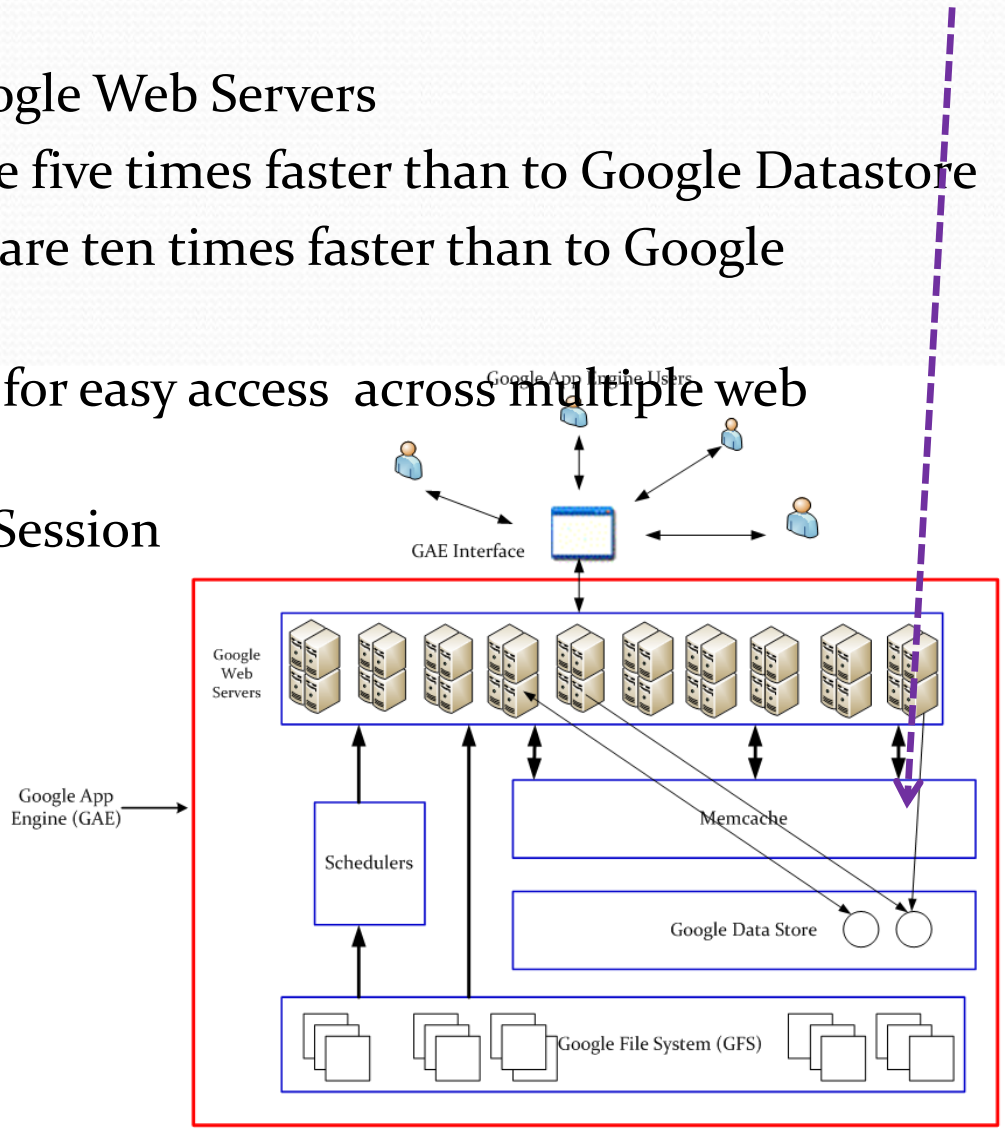


Figure: Memcache in Google App Engine

Illustration of Memcache

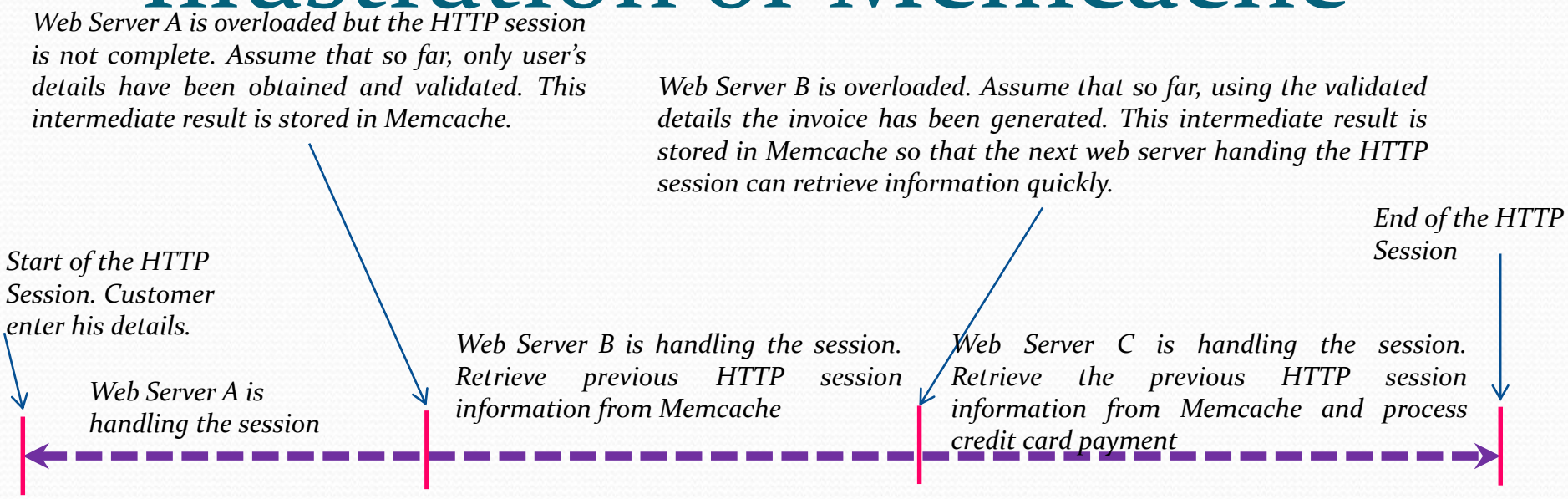


Figure: Example HTTP Session: Enter the customer details, validating them, retrieve invoice for the customer, and process payment using credit-card payment on a GAE hosted application

Deployed Applications on Google Web Server

- An application is made available on the internet as soon as it is deployed.
  - This is unlike IaaS, where virtual machines will need to be explicitly provisioned
  - In Google PaaS, users cannot specify the configuration and number of VM's
- GAE Applications are load balanced across a number of servers automatically
  - This is unlike IaaS, where users need to explicitly set up load balancers
- Resource usage for an application is metered in terms of number of web requests served, CPU-hours actually spent executing requests… etc.

Google App Engine Constraints

- Google's resource usage for deployed applications is free within certain limits (An application on a free account can use up to 1 GB of storage and up to 5 million page views a month)
- Constraints on resource usage by applications as (Quotas) (Quotas)
- Some constraints may be waived by purchasing premium services
- Constraints intended to ensure security of deployed application or fair usage of free available resources

Google App Engine Screenshot

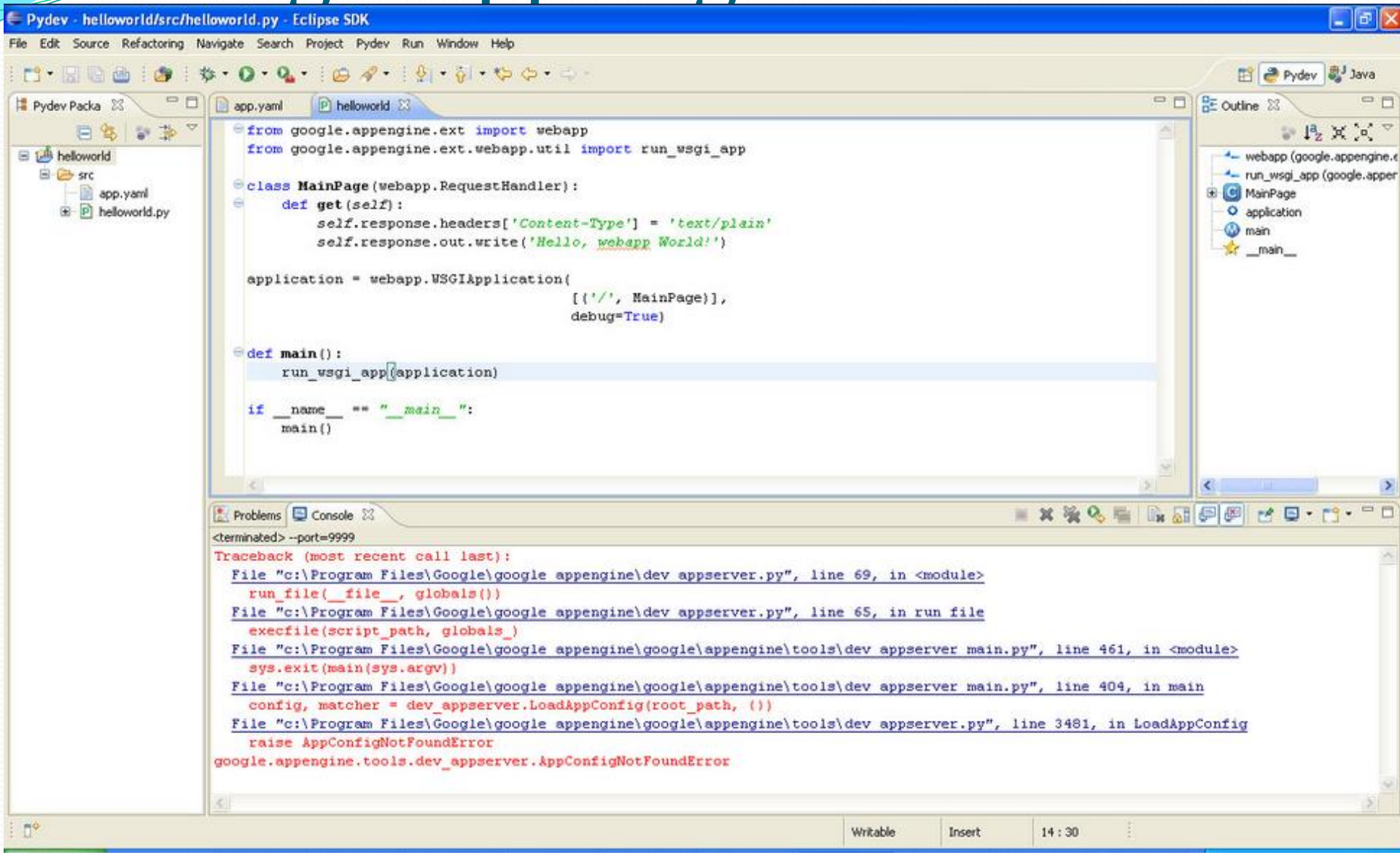


Figure: Application Development in Google App Engine using Eclipse IDE

Platform as a Service – Example 2

- Force.com PaaS
  - Originated to host its Customer Relationship Management (CRM) SaaS application
  - Subsequently offered a platform for application development, deployment and hosting (Force.com)
  - Exposes its software development platform via (very) simple drag, drop and click APIs
    - Provides a simple web-interface for application development
    - Initially Force.com application development was with Eclipse IDE

How does Force.com work?

- The users in Force.com share a single stack of resources (such as the database)
- Different users of the Force platform see customized views
- Force.com *does not* maintain customized views for all the users (rather it generates these views on the fly)

- In the shared database, Force.com provides clear separation between:
  (a) Core application data and functionality (Pre-defined Data and Logic)
  (b) User-defined application data and functionality (Custom Data and Logic)
- Core-application data and logic is stored as metadata
  - Comes pre-packaged with every user
- User-defined application data and logic is also stored as metadata
  - Unique to every user on the Force platform

Force.com Architecture

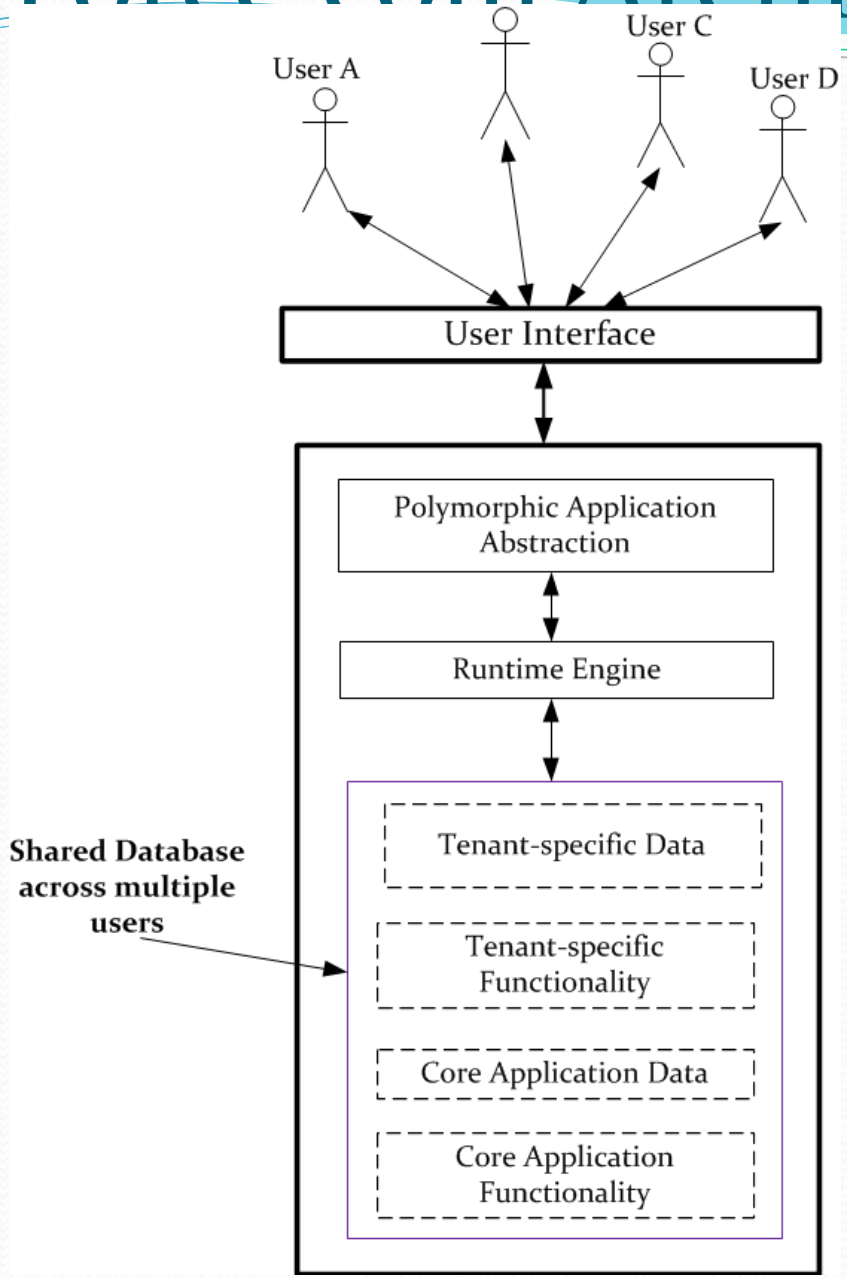
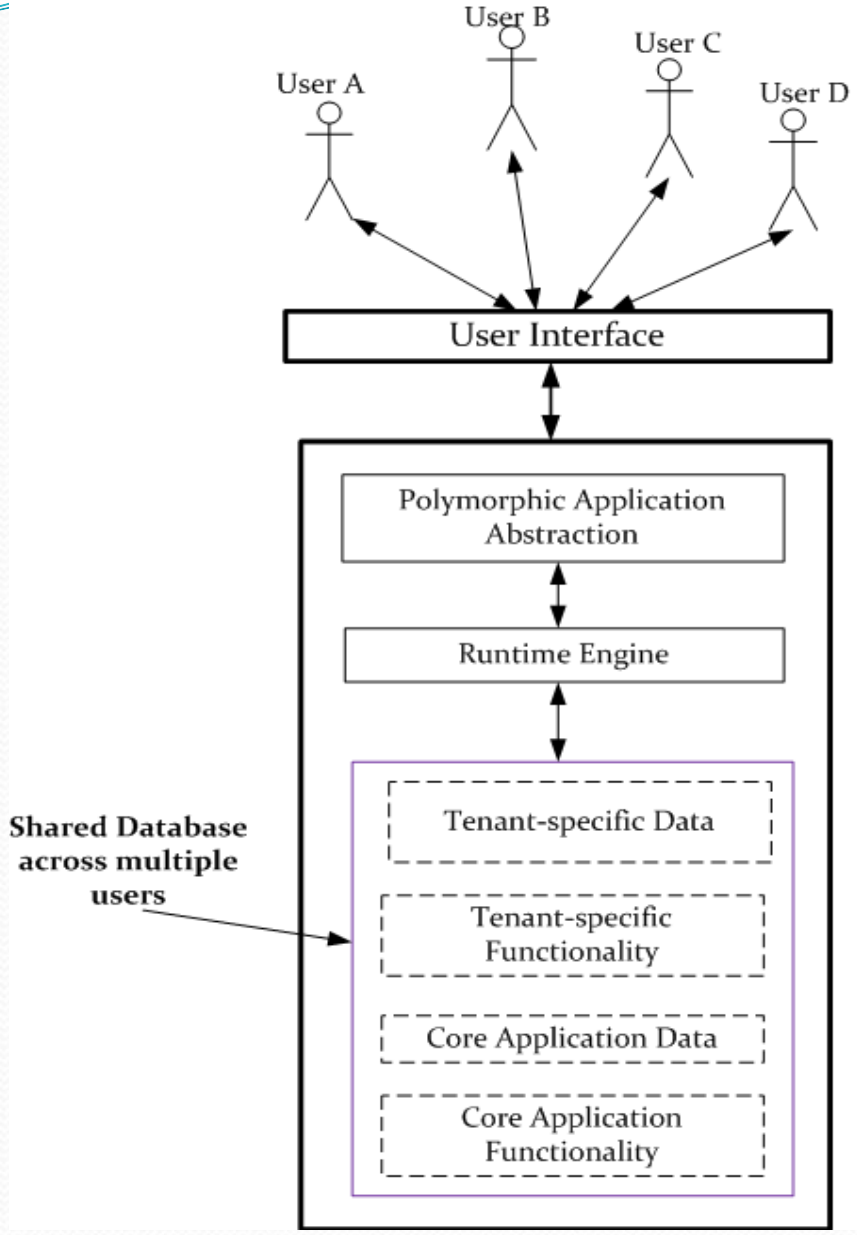


Figure: Architecture of Force.com

Working of Force.com

- Different users (tenants) see customized (tenant-specific) view of the application (depending on the user-defined logic and data)
- "*Polymorphic Abstraction*" process determines which user is requesting data and passes this information to the runtime engine
- Runtime engine accesses:
  - core application metadata; and
  - custom metadata of the corresponding user
- Runtime engine compiles them at run-time to generate custom views for the corresponding user
- Users can change core data, core functionality, or user-defined data without impacting on each other.

Step-wise working of Force.com



Step 1: A given user (assume User A) interacts via a user interface and provides authentication credentials

Step 2: The user interface passes User A's credentials and/or request for information from User A to the Polymorphic Application Abstraction Layer

Step 3: The polymorphic application abstraction determines the user requesting the data and the type of data that is being requested and passes this information to the run time engine

Step 4: The run-time engine takes input from the polymorphic application abstraction, retrieves relevant data, compiles it, and passes it back.

Figure: Step-wise working of Force.com

Demo of Force.com

- Please visit http://developer.force.com and create free developer account
- Creating applications using Force.com platform will be covered during Week 7, Week 8, Week 9 and Week 10 during the lectures.
- We will make use of Force.com (Platform as a Service) during the labs in these weeks.
- Assignment 2 will be done using the Force.com PaaS

Summary

- Computing Platform and its components
- Platform as a Service (PaaS)
- Architecture and working of Google App Engine
- Force.com Platform
  - Multi-tenancy in Force.com
  - Working of Force.com
  - Demonstration of Force.com platform
- Traditional vs. PaaS execution environments

Reading

Books

1. Rhoton, J. (2010), Cloud computing explained, Recursive Press, UK – Chapter 4

2. Shroff, G. (2010), Enterprise cloud computing: technology, architecture, application, Cambridge University Press, UK – Chapter 5

Reading

Papers, Website

1. Weissman, C.D., and Bobrowski, S. (2009), The Design of the Force.com Multitenant Internet Application Development Platform. SIGMOD'09, Providence, Rhode Island, USA.

2. P. Mell and T. Grance. (2011), The NIST Definition of Cloud Computing.

3. http://www.salesforce.com/