Introduction to Google app engine

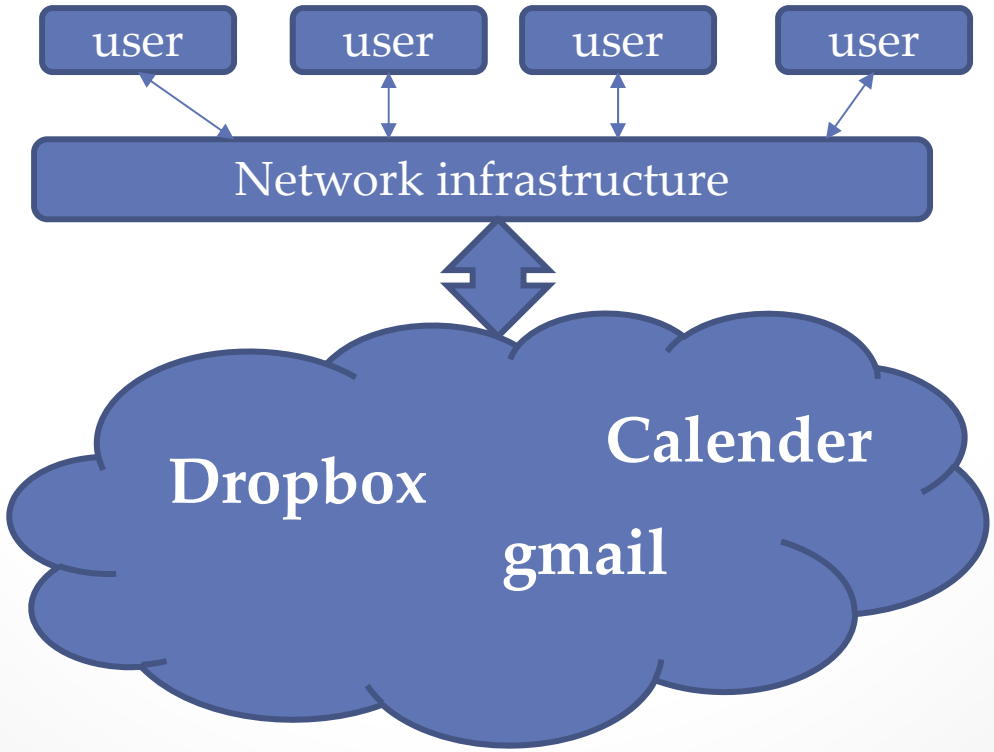
EECE 417 – Software Design

Farid Molazem

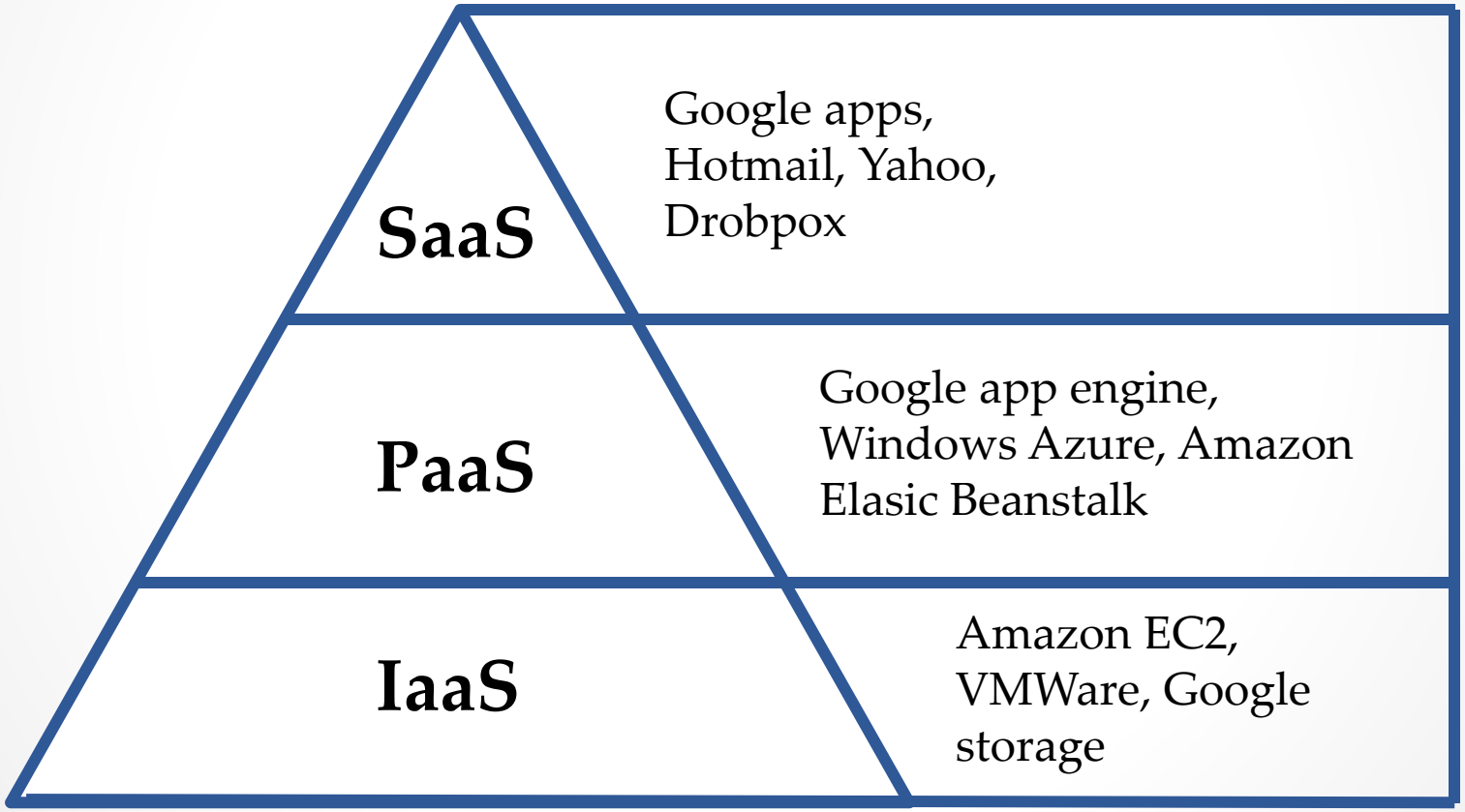
What is cloud computing

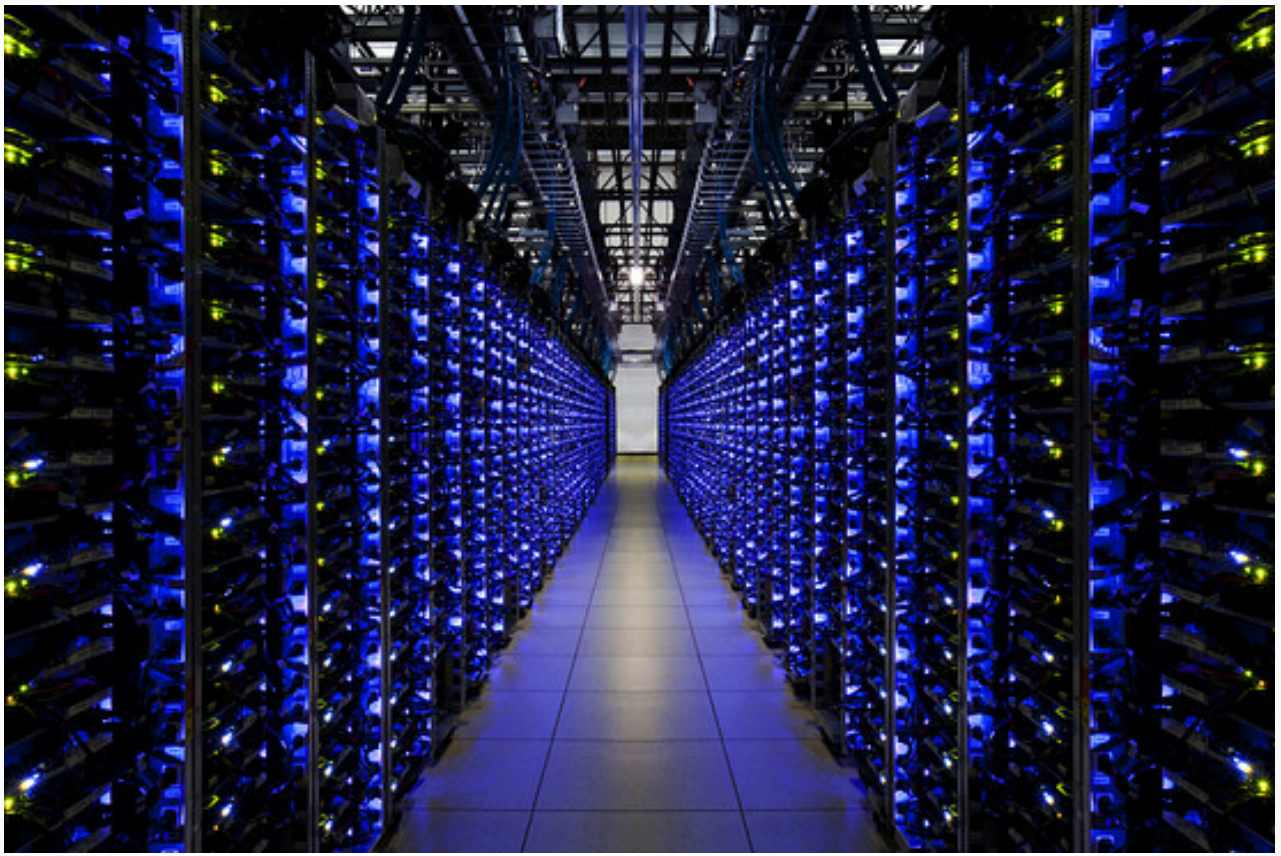
What is cloud computing

- Cloud computing
  - A class of network-based computing that takes place over the internet



Hierarchy


SaaS
Google apps, Hotmail, Yahoo, Drobpox
PaaS
Google app engine, Windows Azure, Amazon Elasic Beanstalk
IaaS
Amazon EC2, VMWare, Google storage



http://blogs.wsj.com/

Google app engine

- Easy to use, scale and manage
- Run your application on Google's infrastructure
- Forgot worries of managing your servers, network, cooling, etc.
- Focus on functionality

Cloud development in a box

- Downloadable SDK
- Application runtimes
  - Java, Python
- Local development tools
  - Eclipse plugin
  - AppEngine Launcher
- Specialized application services
- Cloud-based dashboard
- Ready to scale
- Built-in fault tolerance, load balancing

What you need

- Eclipse
- Google plugin for Eclipse

- Eclipse: a multi-language software development environment
  - Open source
  - Available at : http://www.eclipse.org/downloads

Scaling

- No configuration is required
- Resources are assigned to your app automatically
  - As long as CPUs and resources are available

Security

- Constraints on functionality
  - No process or threads
  - No sockets (use urlfetch API)
  - No files (use datastore)
- Constraint on resrouces
  - Hard time limit of 30 sec per request
  - Hard limit (about 1MB) on request, response size
  - Quota system for number of requests, API calls, bandwidth, etc.

Preserving fairness through quotas

- Everything an app does is limited by quotas
  - Request count, bandwidth used, CPU usage, datastore call count, disk space used, etc.
- Per-minute and per-day
- If you run out a quota for a specific operation, it will be blocked for a while until replenished
- Provides support for a large number of apps

Datastore

- Entities

```
Entity employee = new Entity("Employee");
employee.setProperty("firstName", "Antonio");
employee.setProperty("lastName", "Salieri");
```

- Key: kind, identifier

```
Entity employee = new Entity("Employee", "asalieri");
```

- Transaction
- Entity groups

```
Entity address = new Entity("Address", "addr1", employee.getKey());
```

Project Layout

- Project directory
  - A single directory named PROJECT_NAME/ contains your project files
  - Subdirectory named src/ contains your Java source code
  - Build process compiles the Java source files and puts the compiled classes in the directory war/
- The servlet class
  - App engine Java applications use the Java servlet API to interact with the webserver
  - An HTTP servlet is an application class that can process and respond to web requests

Project layout

- web.xml file
  - To determine which servlet class to call when a request is received by the web server
  - Resides in war/web-inf/ directory
- Appengine-web.xml file
  - Resides alongside web.xml file
  - Includes the registered ID of your application, version number, etc.

Services

- Memcache API – high performance in-memory key-value cache
- Datastore – database storage and operations
- URLFetch – invoking external URLs
- Mail – sending mail from your application
- Task Queues – for invoking background processes
- Images – for image manipulation
- Cron Jobs – scheduled tasks on defined time
- User Accounts – using Google accounts for authentication

http://code.google.com/appengine/docs/java/apis.html

URLFetch API

- Invoking external URLs from your application over HTTP and HTTPs

```
import java.net.*;
import java.io.*;

URL url = new URL("http://www.example.com/atom.xml")
BufferedReader reader = new BufferedReader(new
    inputStreamReader(url.openStream()));
 String line;
while ((line = reader.readLine()) != null) {
        // ...
}
reader.close();
```

http://code.google.com/appengine/docs/java/urlfetch/

Memcache API

- Storage on memory rather on disk

```
import static java.util.Collections.emptyMap;
import javax.cache.*;

Cache cache;
try {
    CacheFactory cacheFactory =
    CacheManager.getInstance().getCacheFactory();
    cache = cacheFactory.createCache(Collections.emptyMap())
} catch (CacheException e) {
            // ...
}
```

http://code.google.com/appengine/docs/java/memcache/

Datastore API

- Data storage and retrieval
- Not a relational database

http://code.google.com/appengine/docs/java/datastore/

Task Queues API

- Launching background processes by inserting tasks into queues using queue.xml, in the WEB-INF/ dir

```
import com.google.appengine.api.labs.taskqueue.Queue;
import com.google.appengine.api.labs.taskqueue.QueueFactory;
import com.google.appengine.api.labs.taskqueue.TaskOptions;

 DatastoreService ds = DatastoreServiceFactory.getDatastoreService();
 Queue queue = QueueFactory.getDefaultQueue();
 try {
    Transaction txn = ds.beginTransaction();

    // ...

    queue.add(TaskOptions.Builder.withUrl("/path/to/my/worker"));

    // ...
    txn.commit();
 } catch (DatastoreFailureException e) {
 }
```

http://code.google.com/appengine/docs/java/config/queue.html

Users API

- Authentication using Google acc

```
import com.google.appengine.api.users.*;

 UserService userService = UserServiceFactory.getUserService();
       User user = userService.getCurrentUser();

       if (user != null) {
          resp.setContentType("text/plain");
          resp.getWriter().println("Hello, " + user.getNickname());
       } else {

resp.sendRedirect(userService.createLoginURL(req.getRequestURI()))
;
       }
```

http://code.google.com/appengine/docs/java/users/

Business Examples

Demo

?