COMP6205: Web Development

Revision 2
What is Web Testing?

• “Web testing” is the name given to Software Testing that focuses on testing the web applications.

• This could help to address the issues in web application before exposed to public.
Major aspects of Web Testing

- Functional testing
- Usability Testing
- Compatibility Testing
- Security testing
  - Tools for Security Testing
- Performance Testing
  - Load Tests & Load Tests Tools
  - Stress Tests
- Testing Strategy
Web Hosting

- Hosting websites refers to placing websites on web servers to bring them into access by people over internet. Web servers play a significant role in web hosting services as they form the key elements. Following are few functions performed by web servers in hosting:
Role of Web Servers in Web Hosting

- **Stores and secures website data:** In web hosting services, a web server stores all website data and secures it from unauthorized users when it is properly configured.

**Provides web database access:** A web server’s responsibility is to provide access to websites that are hosted. Web hosting service providers own some web servers that are used in variable ways to provide different web hosting services, such as backend database servers.

**Serve the end user requests:** Web servers accept requests from different users connected over the internet and serve them accordingly.
Different Types of Web Hosting

- Virtual Server
- Dedicated Server
- Co-Located
- Shared Server
- Cloud
Web Application Deployment

- Basics of client / server architecture
- More clients => more load => more resources to handle it
- Qualities of a well architected application:
  - Availability
  - Reliability
  - Maintainability
- To manifest these qualities, you need a strategy for expanding compute resources quickly as need arises
  - Scale up
  - Scale out
Scaling up

• You typically choose to scale up when any single request demands more memory and processing power to complete, and

• the bottleneck / latency in the system is the intensive number of software objects created in the computer’s memory or the intensive algorithms and business logic that is performed.

• In this case, the key to completing each incoming request more efficiently is to provide each incoming request with more processing power.
Scaling Out

• Conversely, you typically scale out when any single request requires less memory and processing power to complete, but

• the real **bottleneck / latency is in network communication, disk access, etc.**

• In this case, the key to completing each request more efficiently is to run it in parallel to other requests as each wait on external components to complete -- again, waiting on disk access, network access, etc.
Achieving Redundancy - Virtualization

- Virtualization is a broad term (virtual memory, storage, network, etc)
  - The focus here is on platform virtualization
- Virtualization basically allows one computer to do the job of multiple computers, by sharing the resources of a single hardware across multiple environments

‘Non-virtualized’ system
A single OS controls all hardware platform resources

Virtualized system
It makes it possible to run multiple Virtual Containers on a single physical platform
Virtualization - Container

- VMs are based on a hardware level virtualization
  – basically they are virtualizing hardware resources.

- But container virtualization is done at the operating system level, rather than the hardware level.
The choice between VMs and containers

• Do you need a full platform that can house multiple services? Go with a virtual machine.

• Do you need a single service that can be clustered and deployed at scale? Go with a container.

• Docker is an open-source system of software containers.
  – It allows developers to package all of the things that are needed to run a program inside the container itself.
  – The host, runtime, code, operating system, tools, libraries, and other components are all inside an isolated environment.
  – Everything is self-contained, so programmers will not have to worry about what flavor of Linux is being used.
Microsoft Azure & Azure services

• What is Microsoft Azure?
  – An open and flexible cloud platform that enables you to quickly build, deploy, and manage solutions across a global network of Microsoft-managed datacenters.
  – Microsoft Azure allows you to perform virtually any compute or data storage operation by provisioning and scaling the necessary resources **on demand** on a **pay-as-you-go** basis.

• Deploying ASP.NET Core 2.0 Application in Azure
What is Serverless Architecture?

- **Serverless** architectures refer to applications that significantly depend on
  - third-party services (known as **Backend as a Service** or "BaaS") or
  - on custom code that's run in ephemeral containers (known as **Function as a Service** or "FaaS")

- **By using these ideas, and by moving much behavior to the front end, such architectures remove the need for the traditional 'always on' server system sitting behind an application.**
  - Depending on the circumstances, such systems can significantly reduce operational cost and complexity at a cost of vendor dependencies
Backend as a Service

• Web and mobile apps require a similar set of features on the backend, including push notifications, integration with social networks, and cloud storage.
  – Each of these services has its own API that must be individually incorporated into an app, a process that can be time-consuming and complicated for app developers.

• BaaS providers form a bridge between the frontend of an application and various cloud-based backends via a unified API and SDK.
  – Providing a consistent way to manage backend data means that developers potentially saving both time and money.
Functions as a Service

• Serverless can also mean applications where some amount of server-side logic is still written by the application developer but
  – unlike traditional architectures this is run in stateless compute containers that are **event-triggered, ephemeral** (may only last for one invocation), and fully managed by a 3rd party.

• One way to think of this is ‘Functions as a Service / FaaS’.
  – **AWS Lambda** and **Azure Functions** are examples of implementations of FaaS at present, but there are others.
Advantages of Serverless Architectures

1. Horizontal scaling is completely automatic, elastic, and managed by the provider.
   - For example, if your system needs to be processing 100 requests in parallel the provider will handle that without any extra configuration on your part.

2. Functions in FaaS are triggered by event types defined by the provider.
   - With Amazon AWS such stimuli include S3 (file) updates, time (scheduled tasks) and messages added to a message bus (e.g. Kinesis).
   - Most providers also allow functions to be triggered as a response to inbound http requests, typically in some kind of API gateway.
Advantages of Serverless Architectures

3. Serverless is about running backend code without managing your own **server systems** or your own **server applications**.
   - The second clause - server applications - is a key difference when comparing with other modern cloud related concepts like containers and PaaS (Platform as a Service.)

4. Since we have no server applications to run, deployment is very different to traditional systems. We upload the code to the FaaS provider and it does everything else.
   - that typically means uploading a new definition of the code (e.g. in a zip or JAR file), and then calling a proprietary API to initiate the update.
Microservices Architecture

Monolithic deployment approach

- A traditional application has most of its functionality within a few processes that are componentized with layers and libraries.
- Scales by cloning the app on multiple servers/VMs

Microservices application approach

- A microservice application segregates functionality into separate smaller services.
- Scales out by deploying each service independently with multiple instances across servers/VMs

Coarse-grained density of apps/services

Need to deploy the full application

Fine-grained density of services

Independent deployment of microservice
An Example of Microservice-based App

**eShopOnContainers reference application**
(Development environment architecture)

<table>
<thead>
<tr>
<th>Client apps</th>
<th>Docker Host</th>
</tr>
</thead>
<tbody>
<tr>
<td>eShop mobile app</td>
<td>Identity microservice (STS+users)</td>
</tr>
<tr>
<td>Xamarin.Forms C# xPlat. OS: iOSS Android Windows</td>
<td>SQL Server database</td>
</tr>
<tr>
<td>eShop traditional Web app</td>
<td>Catalog microservice</td>
</tr>
<tr>
<td>HTML</td>
<td>SQL Server database</td>
</tr>
<tr>
<td>eShop SPA Web app</td>
<td>Ordering microservice</td>
</tr>
<tr>
<td>TypeScript/Angular 2</td>
<td>SQL Server database</td>
</tr>
<tr>
<td>eShop WebApp MVC</td>
<td>Basket microservice</td>
</tr>
<tr>
<td>ASP.NET Core MVC</td>
<td>Redis cache</td>
</tr>
<tr>
<td>eShop SPA Web app</td>
<td>Marketing microservice</td>
</tr>
<tr>
<td>TypeScript/Angular 2</td>
<td>MongoDB / CosmosDB</td>
</tr>
<tr>
<td></td>
<td>SQL Server</td>
</tr>
<tr>
<td></td>
<td>Locations microservice</td>
</tr>
<tr>
<td></td>
<td>MongoDB / CosmosDB</td>
</tr>
</tbody>
</table>

COMP6205 - Serverless Architecture
MEAN stack

- Components of MEAN stack
  - Node, Express, Angular, MongoDB
- MongoDB and Sharding
  - Theory of noSQL: CAP, ACID – BASE
- Middleware-based Request Handling & Routing in Express
- SPA or Non-SPA?
- Node.js Architecture
  - Blocking vs Non-Blocking
  - Node Web Apps - Single Thread, Non-Blocking I/O
Content Management Systems (CMS)

• What is a Content Management System (CMS)?

• Common Features of a CMS

• Different Types of CMS
  – Web Content Management Systems (CMS)
  – Enterprise Content Management System (ECMS)

• ECM Applications

• Benefits of CMS Systems & Key Features

• Examples of WCM, Drupal Basic Concepts/ Building Blocks
Enterprise Content Management

- What is ECM and Why use ECM?
- ECM life cycle and phases
- Feature Sets of An ECM System
- SharePoint Building Blocks
Enterprise Search

- Enterprise Search vs Google Search
- What Enterprise Search Is?
- Enterprise search – Challenges
- Why Enterprise Search is Important?
- Search Algorithms - Recall and precision
- ES: Presenting Results – Clustering
- Search architectures for SharePoint 2013
- Open Source Search Systems
Web Performance

• The Relation Between Revenue and Web Performance

• Web Performance Metrics

• Website Optimisation - Bandwidth vs Latency

• Web Performance Analysis Tools – Waterfall Chart

• Reading & understanding a Waterfall Chart

• Backend and Frontend - Time Spent on Back-end/Frontend
  
  – Golden Rule: 80-90% of the end-user response time is spent on the frontend
Web Performance – Cont.

• Why Focus on Front-end Performance?
• Perceived speed vs. Actual Speed
• Performance Metrics
  – Backend metrics
  – Front-end metrics
    • Minification and reducing asset Weight
  – Content complexity
    • Reducing the Number of HTTP Requests (Inline data, Image Maps, CSS Sprites)
    • Taking the advantages of HTTP2