CSC8110: Cloud Computing
Coursework Assignment 2014-15

Matthew Forshaw (`matthew.forshaw@ncl.ac.uk`)

Aims

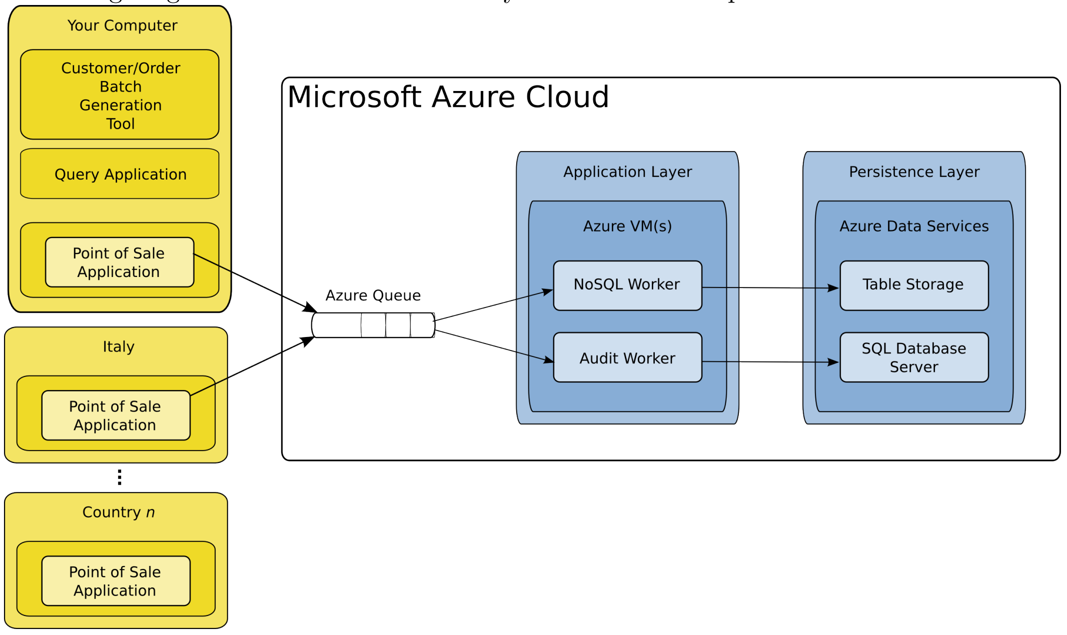
- To use appropriate use of a variety of technologies to build a simple multi-tier retail application in the Microsoft Azure cloud.
- To gain familiarity with, and reflect on, the use of Cloud technologies to tackle a real-world problem.

Learning Skills/Outcomes

- Programming, problem solving and system-design skills.
- To gain experience using Microsoft Azure and the Azure SDK.
- To gain practical experience using message brokering systems in the Cloud.
- To gain experience using relational and NoSQL data stores in the Cloud.

Architecture Overview

The following diagram shows the architecture you will need to implement:



Tasks

1. Point of Sale (POS) Application

You will first develop a *Point of Sale* (POS) or *checkout* where customers' orders may be placed. Your POS application should provide the following basic functionality:

- **Register and list customers.**
  Customers should have a unique identifier, a name, and the country in which they are located (see bullet point three).
- **Place and list orders.**
  Orders should have a unique identifier, a customer identifier, a *"stock keeping unit" (SKU)* (a unique product identifier), an order date and time, and a total order price.
- **Country/city location**
  Each POS application should be aware of the country and city in which it is located. This may either be passed to the command line as an argument, or read from a properties file, but must **not** be hard coded into your application. It is possible for multiple POS applications to run in parallel based in either the same or different locations.
- **Command line interface (CLI)**
  All required operations for your application should be accessible using a command line interface. We strongly recommend using a library such as Apache Commons CLI [1] to develop your command line interface.

  > **Important:** all information should be passed as command line arguments to `args[]` (in the case of Java); your program should **not** require interactive keyboard input from users.

- **Offline operation**
  It is common for Point of Sale (POS) systems to operate standalone when an internet connection is not available. You should develop your Point of Sale application so it is capable of storing messages to be sent when an internet connection is re-established.

  You should consider a situation where your application crashes while there is no internet connection, and enhance your application to cope with this type of failure.

Your Point of Sale (POS) application should take the provided customer and order information and place it in an Azure Queue for processing.

2. NoSQL Worker

You will then create a *'NoSQL Worker'* which retrieves messages from your queue, and stores the details of customers and orders in a NoSQL data store. In the first instance, you are expected to store customers and orders in separate tables within Table Storage.

Your NoSQL Worker should execute continuously, periodically checking the queue for incoming messages. You will use the Windows Azure Table storage service, a scalable

[1] http://commons.apache.org/proper/commons-cli/

datastore for the storage and querying of structured, non-relational data. A full description of the Azure Table storage service, instructions for creating an Azure storage account from the Azure management portal, and Java source code examples can be found in the practical material.

By default, queues guarantee *At Least Once Processing*, whereby each message will be processed once, but under some conditions a given message may be processed multiple times. You should now fix your NoSQL Worker to prevent a message being processed more than once (At Most Once Processing). In your report, discuss the following:

1. What impact would At Least Once Processing have in our context of a retail system processing customers' orders?
2. How have you fixed this problem in your application?
3. Are there any situations where your system cannot guarantee At Most Once Processing?

3. Generating test customers and orders

Next you are required to develop a program to generate example customers and orders and populate your system. Your application should generate **at least** *1,000* customers, and *10,000* orders within your system. This will mitigate the laborious task of adding test data to your system, and will provide you with the large quantity of data required later in this coursework.

We expect you to run this tool directly on your own computer, and for it to add messages to your queue. Your tool should **not** insert customer and order information directly into your database. You may wish to make use of the code you have already developed by extending Task 1.

Hint: Your tool may contain arrays/lists of first and last names which are selected randomly when creating customers. For your orders you may also generate random dates, times and total order prices.

4. Query Application

The managers of the retail application have specified a number of *"management reports"* (*queries*) they wish to be able to perform on the customer and order information. These are as follows:

4.1. Management Report A: JOINs on data in NoSQL.

You now need to extend your retail application to be able to generate the following report.

> *"Calculate the **total order price** of **all orders** placed by customers from a given country."*

Unlike relational databases, NoSQL does not support JOIN statements, so you will need to execute multiple queries against your NoSQL data store and perform the JOIN between *customers* and *orders* programmatically within your application logic.

You may implement this as a standalone application running on your computer, but marks will be awarded for ensuring this application is *deployable* (e.g. as a JAR file including all dependencies).

4.2. Management Report B

To mitigate the need to conduct the JOINs programmatically, you should consider an additional approach whereby you create an entity containing both order information and country of origin. You should make an appropriate choice of *partition key* to ensure this query works in a performant manner.

Extend your system to generate the following report using this new representation of your data.

> *"Calculate the* ***mean average order price*** *of* ***all orders*** *placed by customers from a given country."*

4.3. Management Report C

We now wish you to extend your Query application to produce the following report.

> *"List the* ***details of all orders*** *placed within the last* ***seven days*** *placed by customers from a given country."*

You should carefully consider your choice of partition and row keys to ensure this query runs as efficiently as possible.

5. Relational databases in Azure

The owners of the retail application have introduced an additional requirement that all customer and order information must stored in a relational database for auditing purposes.

You should now produce an additional application which consumes messages from your queue and places these into an MS SQL Server instance in Azure. You must then select a realistic query of your choice, and extend your *Query Application* to interact with the MS SQL Server database and perform the query.

In your report you should discuss the difference between these approaches (namely Table Storage and relational databases) and discuss situations where each approach may be most appropriate.

Useful resources

There are a number of online resources which you may find useful while completing this coursework assignment. These are available at the following URL:
`https://github.com/NewcastleComputingScience/csc8110/blob/master/Resources.md`

If you find an online resource you feel should be included on this list, please don't hesitate to email Matt, or issue a Pull Request against the CSC8110 repository.

Windows Azure Documentation Center
http://azure.microsoft.com/en-us/documentation/

Windows Azure Virtual Machines Documentation
http://azure.microsoft.com/en-us/documentation/services/virtual-machines/

Windows Azure Java Developer Center http://azure.microsoft.com/en-us/develop/java/

Choice of Technologies

We encourage you to make your own decision over your choice of programming language and tools used in completing this assignment. However, there are a few pointers which might help guide your decision:

- **Is there an Azure SDK available for the programming language?**
  Windows Azure offers a number of SDKs in a variety of different languages including Java, Node.js, PHP, .NET, Python and Ruby. While it possible to interact with the management functions of Azure using raw REST requests, this will prove time consuming and error prone and we strongly suggest you select one of the supported languages listed above.
- **Which technologies are you most familiar and comfortable with using?**
  If there are a set of technologies you have more experience of and would feel more comfortable using in the cloud, you may want to select those for this coursework. Alternatively, if you wish to expose yourself to using new technologies, your experiences could form an interesting discussion point in your report.
- **Will the demonstrators have experience using these technologies to be able to help you?**
  While demonstrators will do their best to assist you with any issues you have, they may not be familiar with less common technologies. If in doubt, ask a demonstrator!
- **Have these technologies been used on Azure before?**
  Can you find tutorials or examples of people being successful configuring these technologies in Azure? Are there any pre-defined Azure virtual machine instances which provide these technologies?

If you have any concerns over your choice of implementation language, dont hesitate to contact a demonstrator and they will be able to advise.

Deliverables

Source Code

You are expected to submit all source code and any third-party libraries (or a Maven build file with correctly specified dependencies) required for us to deploy and test your application in the Windows Azure cloud. You should also include a brief *README* document explaining any configuration steps required to run your application. Marks will be awarded for the use of an appropriate build tool, for the quality of code documentation, and for ensuring your application is easily *deployable* in the Azure cloud.

Written Report

You should prepare a written report in **PDF format** containing the following:

- A detailed description of each component in your system.
- A bullet point list of any assumptions which motivated your architectural or system design decisions.

- A description of your NoSQL database representations of customers and orders.
- An detailed explanation of your approach to implementing each of the management reports in Task 4. Please be sure to include content on each of the report discussion points detailed in the tasks.
- Screenshots of the Windows Azure management portal showing your virtual machine, storage and queue dashboards.

Matt Forshaw will be happy to provide initial feedback on your written reports prior to submission. If you would like feedback on your report draft, please email a copy in **PDF format** to `matthew.forshaw@ncl.ac.uk`, allowing **24 hours** for feedback to be provided, though this will often happen sooner.

Practical Sessions

Demonstrators will be available in the computer clusters in the following CSC8110 practical sessions.

Date	Time
Tuesday 2nd December	3-5pm
Friday 5th December	9-11am
Tuesday 9th December	3-5pm
Wednesday 10th December	9-11am
Friday 12th December	9-11am and 1-3pm

Deadline and Submission

Completed assignments including all source code and your written report must be submitted electronically through NESS by **Friday 12th December 2013** at **5pm**.

This assignment is worth **30%** of your final mark for this module.

Questions?

If you have any queries about the coursework assignment or what you are expected to include as part of your submission, you should approach one of the demonstrators during the practical sessions. You may also email Matthew Forshaw (`matthew.forshaw@ncl.ac.uk`) at any time, or use the Discussion Board on Blackboard.

Acknowledgements

We wish to thank the following people for their contributions towards the delivery of this assignment; Sami Alajrami, Hugo Firth, Tudor Miu, Francisco Rocha, Rebecca Simmonds.