Limitations of Azure in GIS Scalability
A performance and migration study

Jonas Bäckström
Abstract

In this study, the cloud platform Windows Azure has been targeted for test implementations of Geographical Information System (GIS) software in the form of map servers and tile caches. The map servers included were GeoServer, MapNik, MapServer and SharpMap, which together with the tile caches, GeoWebCache, MapCache and TileCache, were installed on Windows Azures three different virtual machine roles (Web, Worker and VM). Furthermore, different techniques for scaling applications and internal role communication are presented, followed by four sets of performance tests. The performance tests attempt to highlight the differences in request times, how the different role sizes handle the load from the incoming requests, how the different role sizes handle many concurrent TCP-connections and how well the incoming requests are load balanced in between the worker roles. The test implementations showed that all map servers and tile caches were successfully installed in Azure, which leads to the conclusion that Windows Azure is suitable for hosting GIS software with similar installation requirements to the previously mentioned software. Four different approaches (Direct mapping, Public Internal Endpoints, Queue and Worker Role Request Broker) are presented showing how Azure allows different methods in order to scale the internal role communication as well as the external client requests. The performance tests provided somewhat inconclusive test results due to hardware limitations in the test setup. This made it difficult to draw concluding parallels between the final results and the expected values. Minor tendencies in performance gain can be seen when scaling the VM size as well as the number of VMs.

Keywords: Windows Azure, GIS, Scalability, Migration
Acknowledgments

This project could not have been finished without my university supervisor Martin Kjellqvist (Lecturer at Mid Sweden University), who encouraged, inspired and challenged me throughout my thesis work and also in earlier academic courses.

Secondly I would like to express my sincere gratitude to Joakim Berkebo (Supervisor, Triona) for his supervision and guidance during my time at Triona.

For this thesis I would like to thank my examiner, Professor Ting Ting Zhang and opponent Carl Wikblom, for for their interest, time and helpful comments.

In addition I gratefully acknowledge and extend heartfelt gratitude to the following people at Triona; Peter Löfås (System architect) and Jonas Österberg (System developer) for assistance during the migration and test phase, Henrik Dahlgren (System developer) for assisting me in the test phase, and Fredrik Nordfors and Thomas Höjsgaard for the initial meetings and administrative assistance.

I would also like to thank Emil Gustafsson (Microsoft employee and Windows Azure Team member) for his valuable discussions on the topic of Windows Azure load balancers and scaling semantics.

Finally I would like to thank the Triona for lending me the hardware and software to make this project possible.
Contents

Abstract ii
Acknowledgments iii
Contents iv
Terminology ix

1 Introduction 1
1.1 Background and Problem Motivation .......................... 1
1.2 High-level Problem Statement ................................ 2
1.3 Scope ..................................................................... 2
1.4 Concrete and Verifiable Goals ................................... 2
1.4.1 Goals ................................................................. 2
1.5 Outline ..................................................................... 3

2 Theory 4
2.1 Web Servers ............................................................ 4
2.1.1 Internet Information Services (IIS) ......................... 4
2.1.2 Apache Tomcat ................................................... 4
2.2 Map Servers ............................................................ 5
2.2.1 GeoServer ........................................................... 5
2.2.2 MapNik ............................................................... 5
2.2.3 MapServer .......................................................... 5
2.2.4 SharpMap ............................................................ 5
2.3 Tile Caches .............................................................. 5
2.3.1 GeoWebCache ...................................................... 5
2.3.2 MapCache ........................................................... 5
2.3.3 TileCache ............................................................ 5
2.4 Web Interfaces .......................................................... 6
2.4.1 OpenLayers ......................................................... 6
2.4.2 Polymaps .............................................................. 6
2.4.3 Bing Maps .......................................................... 6
2.4.4 Google Maps ....................................................... 6
2.5 Cloud Computing ...................................................... 6
2.5.1 Infrastructure as a Service (IaaS) ............................. 6
2.5.2 Platform as a Service (PaaS) ................................. 6
2.5.3 Software as a Service (SaaS) ................................. 7
2.7.4 IBM SmartCloud ................................................. 23
2.8 Migration of Existing Application - Cyklereseplaneraren .......... 23
2.9 Related work .................................................. 24

3 Methodology ......................................................... 25
  3.1 Tools ......................................................... 25
    3.1.1 Hardware ............................................... 25
    3.1.2 Development ........................................... 25
  3.2 Map Server and Tile Cache Implementation .......................... 25
    3.2.1 VM Role Implementation ................................ 26
    3.2.2 Web & Worker Role Implementation ....................... 26
  3.3 Internal Role Scalability Options ................................ 26
  3.4 Application Migration & Performance Tests ........................ 27
    3.4.1 Migration Architecture and Scalability Aspects .............. 27
      3.4.1.1 Scalability Aspects ................................. 27
    3.4.2 ASP.NET MVC Web Front-end ................................ 28
    3.4.3 ASP.NET MVC Routehandler ................................ 28
    3.4.4 Windows Service ......................................... 28
    3.4.5 VM Specification ........................................ 28
    3.4.6 The test conditions ...................................... 28
      3.4.6.1 Test 1: Request time differences between Azure VM & local VM ........................................... 28
      3.4.6.2 Test 2: Heavy calculation requests on different VM-sizes ........................................ 29
      3.4.6.3 Test 3: Multiple concurrent request on different VM-sizes ........................................ 29
      3.4.6.4 Test 4: Heavy calculation requests on scaled VM-solutions ........................................ 29

4 Implementation ....................................................... 30
  4.1 Map Server Implementations ..................................... 30
    4.1.1 MapServer ................................................ 30
      4.1.1.1 VM Role ............................................. 30
      4.1.1.2 Worker Role ....................................... 30
    4.1.2 SharpMap ................................................ 32
      4.1.2.1 VM Role ............................................. 32
      4.1.2.2 Worker Role ....................................... 32
    4.1.3 GeoServer ............................................... 32
      4.1.3.1 VM Role ............................................. 32
      4.1.3.2 Worker Role ....................................... 32
    4.1.4 MapNik .................................................. 35
      4.1.4.1 VM Role ............................................. 35
      4.1.4.2 Worker Role ....................................... 35
  4.2 Tile Cache Implementations ....................................... 37
    4.2.1 MapCache ................................................. 37
      4.2.1.1 VM Role ............................................. 37
      4.2.1.2 Worker Role ....................................... 37
    4.2.2 TileCache ............................................... 38
      4.2.2.1 VM Role ............................................. 38
      4.2.2.2 Worker Role ....................................... 38
    4.2.3 GeoWebCache ............................................. 39
      4.2.3.1 VM Role ............................................. 39
## Contents

4.2.3.2 Worker Role ........................................... 39

5 Results ......................................................... 41

5.1 Map Servers on Windows Azure VM Role ....................... 41
  5.1.1 MapServer .................................................. 41
  5.1.2 SharpMap ................................................... 41
  5.1.3 GeoServer .................................................. 41
  5.1.4 MapNik ..................................................... 41

5.2 Map Servers on Windows Azure Worker Role ................... 42
  5.2.1 MapServer ................................................... 42
  5.2.2 SharpMap ................................................... 42
  5.2.3 GeoServer ................................................... 42
  5.2.4 MapNik ..................................................... 42

5.3 Tile Caches on Windows Azure VM Role ......................... 42
  5.3.1 MapCache ................................................... 43
  5.3.2 TileCache ................................................... 43
  5.3.3 GeoWebCache ................................................. 43

5.4 Tile Caches on Windows Azure Worker Role ..................... 43
  5.4.1 MapCache ................................................... 43
  5.4.2 TileCache ................................................... 43
  5.4.3 GeoWebCache ................................................. 43

5.5 Internal Role Scalability Options .............................. 44
  5.5.1 Direct Mapping .............................................. 44
  5.5.1.1 Characteristics ........................................ 44
  5.5.2 Public Internal Endpoints ................................. 44
  5.5.2.1 Characteristics ........................................ 44
  5.5.3 Queue ...................................................... 45
  5.5.3.1 Characteristics ........................................ 45
  5.5.4 Worker Role Request Broker ............................... 46
  5.5.4.1 Characteristics ........................................ 46

5.6 Performance Tests ............................................. 47
  5.6.1 Test 1: Request time differences between Azure VM & local VM 47
  5.6.2 Test 2: Heavy-calculation requests on different VM-sizes .... 48
  5.6.3 Test 3: Multiple concurrent request on different VM-sizes .... 49
  5.6.4 Test 4: Heavy-calculation requests on scaled VM-solutions ... 50

6 Conclusions ..................................................... 51

6.1 Implementation ............................................... 51

6.2 Internal Role Scalability Options .............................. 51

6.3 Performance tests ............................................. 51
  6.3.1 Test 1: Request time differences between Azure VM & local VM 52
  6.3.2 Test 2: Heavy-calculation requests on different VM-sizes .... 52
  6.3.3 Test 3: Multiple concurrent request on different VM-sizes .... 52
  6.3.4 Test 4: Heavy-calculation requests on scaled VM-solutions ... 52
  6.3.5 How to improve the performance tests ........................ 52

6.4 General Azure Observations .................................. 52
  6.4.1 PowerShell and updating 3rd party files .................... 53
  6.4.2 Startup Tasks .............................................. 53

6.5 Difficulties .................................................. 53
CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.5.1 Implementation phase</td>
<td>53</td>
</tr>
<tr>
<td>6.5.2 Migration phase</td>
<td>54</td>
</tr>
<tr>
<td>6.5.3 Performance tests</td>
<td>54</td>
</tr>
<tr>
<td>6.6 Azure Specific Considerations</td>
<td>54</td>
</tr>
<tr>
<td>6.7 Future Work</td>
<td>55</td>
</tr>
</tbody>
</table>

Bibliography 56
### Terminology

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>.NET</td>
<td>Software framework primarily developed for Microsoft Windows.</td>
</tr>
<tr>
<td>ACS</td>
<td>(Access Control Service)</td>
</tr>
<tr>
<td>AJP</td>
<td>(Apache JServ Protocol)</td>
</tr>
<tr>
<td>ASP</td>
<td>(Application Service Provider)</td>
</tr>
<tr>
<td>ASP</td>
<td>(Active Server Pages)</td>
</tr>
<tr>
<td>AtomPub</td>
<td>(Atom Publishing Protocol)</td>
</tr>
<tr>
<td>BAL</td>
<td>(Base Activity Library)</td>
</tr>
<tr>
<td>BLOB</td>
<td>(Binary Large Object)</td>
</tr>
<tr>
<td>CDN</td>
<td>(Content Delivery/Distribution Network)</td>
</tr>
<tr>
<td>CLR</td>
<td>(Common Language Runtime)</td>
</tr>
<tr>
<td>CMS</td>
<td>(Content Management Systems)</td>
</tr>
<tr>
<td>CTP</td>
<td>(Community Technology Preview)</td>
</tr>
<tr>
<td>DMZ</td>
<td>(DeMilitarized Zone)</td>
</tr>
<tr>
<td>ESP</td>
<td>(Extended Stored Procedures)</td>
</tr>
<tr>
<td>FC</td>
<td>(Fabric Controller)</td>
</tr>
<tr>
<td>GA</td>
<td>(Guest Agent)</td>
</tr>
<tr>
<td>GIS</td>
<td>(Geographic information system)</td>
</tr>
<tr>
<td>CONTENTS</td>
<td></td>
</tr>
<tr>
<td>----------</td>
<td></td>
</tr>
<tr>
<td>Hadoop</td>
<td>A software framework that supports data-intensive distributed applications under a free license.</td>
</tr>
<tr>
<td>HTTP</td>
<td>(HyperText Transfer Protocol)</td>
</tr>
<tr>
<td>HTTPS</td>
<td>(HyperText Transfer Protocol Secure)</td>
</tr>
<tr>
<td>HPC</td>
<td>(High Performance Computing)</td>
</tr>
<tr>
<td>IaaS</td>
<td>(Infrastructure as a Service)</td>
</tr>
<tr>
<td>IIS</td>
<td>(Internet Information Service)</td>
</tr>
<tr>
<td>IIS7</td>
<td>Latest release of the IIS (version 7.0). There exists version 7.5 for Windows Server 2008 R2.</td>
</tr>
<tr>
<td>ISV</td>
<td>(Independent Software Vendor)</td>
</tr>
<tr>
<td>Java</td>
<td>Object-oriented programming language.</td>
</tr>
<tr>
<td>JVM</td>
<td>(Java Virtual Machine)</td>
</tr>
<tr>
<td>LB</td>
<td>(Load Balancer)</td>
</tr>
<tr>
<td>LINQ</td>
<td>(Language Integrated Query)</td>
</tr>
<tr>
<td>MPI</td>
<td>(Message Passing Interface)</td>
</tr>
<tr>
<td>MSMQ</td>
<td>(Microsoft Message Queuing)</td>
</tr>
<tr>
<td>NNTP</td>
<td>(Network News Transfer Protocol)</td>
</tr>
<tr>
<td>Node.js</td>
<td>A JavaScript based software system used for developing scalable web applications.</td>
</tr>
<tr>
<td>NoSQL</td>
<td>(Not Only SQL) - Type of database</td>
</tr>
<tr>
<td>NTFS</td>
<td>(New Technology File System) - Standard file system for Windows NT and later versions.</td>
</tr>
<tr>
<td>OLAP</td>
<td>(Online Analytical Processing)</td>
</tr>
<tr>
<td>PaaS</td>
<td>(Platform as a Service)</td>
</tr>
<tr>
<td>PHP</td>
<td>(PHP: Hypertext Preprocessor) A server-side scripting language for web applications.</td>
</tr>
<tr>
<td>Abbreviation</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------</td>
</tr>
<tr>
<td>RDL</td>
<td>Report Definition Language</td>
</tr>
<tr>
<td>SA</td>
<td>System Administrator</td>
</tr>
<tr>
<td>SaaS</td>
<td>Software as a Service</td>
</tr>
<tr>
<td>SAML</td>
<td>Security Assertion Markup Language</td>
</tr>
<tr>
<td>SLA</td>
<td>Service Level Agreements</td>
</tr>
<tr>
<td>SMTP</td>
<td>Simple Mail Transfer Protocol</td>
</tr>
<tr>
<td>SOA</td>
<td>Service Oriented Architecture</td>
</tr>
<tr>
<td>SSDT</td>
<td>SQL Server Data Tools</td>
</tr>
<tr>
<td>SSL</td>
<td>Secure Sockets Layer</td>
</tr>
<tr>
<td>STS</td>
<td>Security Token Service</td>
</tr>
<tr>
<td>TDS</td>
<td>Tabular Data Stream</td>
</tr>
<tr>
<td>VF</td>
<td>Windows Workflow Foundation</td>
</tr>
<tr>
<td>VHD</td>
<td>Virtual Hard Drive (for running Windows Server 2008 R2 images)</td>
</tr>
<tr>
<td>VIP</td>
<td>Virtual IP</td>
</tr>
<tr>
<td>VM</td>
<td>Virtual Machine</td>
</tr>
<tr>
<td>WCF</td>
<td>Windows Communication Foundation</td>
</tr>
<tr>
<td>WFS</td>
<td>Web Feature Service</td>
</tr>
<tr>
<td>WKS</td>
<td>Well-Known Text</td>
</tr>
<tr>
<td>WMS</td>
<td>Web Map Service</td>
</tr>
<tr>
<td>XML</td>
<td>Extensible Markup Language</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

In this project the suitability of running *Geographic Information System* (GIS) applications on the cloud platform Windows Azure will be tested and analyzed. The most commonly used GIS map servers and tile caches will be test implemented on the Windows Azure VM and Worker Roles. The successful implementations will form part of a performance test against a in-house solution.

1.1 Background and Problem Motivation

The increasing computational complexity of *Geographic information system* (GIS) due to more detailed maps and more information is forcing companies to continuously buy or upgrade their hardware infrastructure. A solution to this problem is urgently required because this can lead to a considerable reduction in the costs and administration for Triona AB\(^1\) and other projects in this area.

*Geographic information system* (GIS) - used for handling, storing and analyzing geographical data - was developed in the 1960s by Dr. Roger Tomlinson [1]. This system concept forms the backbone of the present day online map services e.g. Google Maps, Bing Maps, ViaMichelin. When building GIS applications, two great problems are faced firstly the computational problem associated with rendering the map images data and secondly, distributing these images to a large set of - application consuming - users.

*Cloud computing* is a relatively new term but the idea itself is far from new, especially in the area of Software-as-a-Service (SaaS) in which the significant e-mail services such as *Hotmail* and *Yahoo* have existed for over a decade. In 2006, Amazon released its cloud computing service *Amazon Cloud* [2], in which developers could write their applications in-house and then upload them to the cloud-service for hosting. This was a new means of hosting applications which allowed the companies to focus on writing the code and leave the management of infrastructure to the service provider. This type of concept also provided the possibility for highly scalable systems since the customers could rent more infrastructure, and were able to scale it linearly to the usage of their application. The software giant *Microsoft Inc.* has recently released a new cloud computing service called *Windows Azure*, which is similar to *Amazon Cloud*.

\(^1\)TRIONA is a computer services company combining niche business sector competence, primarily within transport infrastructure, traffic, transportation, forestry, and the energy/vehicle industry with leading edge competence within system development and application management. Website: http://www.triona.se
Combining the two technologies mentioned above namely Cloud Computing and GIS, offers the application developers the possibility of narrowing the focus from managing and developing both the infrastructure and the application, to focusing only on the application development. Looking into the future, it is highly likely that cloud computing will prove to be the means of hosting this type of applications, but - due to the short time since release - very little research has been conducted in this area, particularly in relation to Windows Azure.

1.2 High-level Problem Statement

The project aims to implement and investigate three different relations between GIS and the Window Azure platform. The first step is to implement GIS software in terms of map servers and tile caches on the Azure platform's different roles options, followed by a presentation of different scaling tactics for internal communication and finally a test migration together with performance tests is to be conducted. The outcome of these three steps will then be used to evaluate whether the cloud computing platform Windows Azure is a good contender for hosting GIS applications.

1.3 Scope

This project will not cover the following areas:

- Security aspects of the Windows Azure platform and GIS software.
- Access control via Windows Azure Active Directory or Access Control Service.
- Reporting through SQL Azure Reporting and other business analytics tools.

1.4 Concrete and Verifiable Goals

The project’s objective is to investigate, analyze or implement a solution to the technical problems described in the following subsection.

1.4.1 Goals

Here follows the non-optional demands set for this project.

- Implement the following Map Servers on the Windows Azure VM Role:
  - MapServer
  - SharpMap
  - GeoServer
  - MapNik
- Implement the following Map Servers on the Windows Azure Worker Role:
  - MapServer
  - SharpMap
• GeoServer
• MapNik

• Implement the following Tile Caches on the Windows Azure VM Role:
  – MapCache
  – TileCache
  – GeoWebCache

• Implement the following Tile Caches on the Windows Azure Worker Role:
  – MapCache
  – TileCache
  – GeoWebCache

• Present different means of scaling internal role communication.

• Migrate an existing application to Windows Azure.

• Conduct load and performance tests - measuring the time taken (for given test data) from request to response - versus an existing service hosted on a non-azure VM and different scalable options in Windows Azure.
  – Request time differences between Azure VM & local VM
  – Heavy calculation requests and different VM-sizes
  – Multiple concurrent request and different VM-sizes
  – Heavy calculation requests and scaled VM-solution

1.5 Outline

In the first chapter the user is provided with a brief introduction to the subject and the demands that have been set in relation to this project. Chapter two – Theory – provides a detailed description of Windows Azure, used libraries, techniques, concepts and some brief notes on related work. The methods chapter clarifies the problem statement, the projects attack point and how the tests were performed. In the fourth chapter, the implementation is described in detail. Chapter five – Results - presents how the final application coincides with the statements/demands set in the introduction chapter. In the final chapter the author offers his personal reflections, describes some of the difficulties and problems that occurred during the implementation phase, and discusses how the research could be further extended and improved.
Chapter 2

Theory

The theory chapter provides a presentation of the different techniques and GIS-software used in the study in addition to an in depth description of Windows Azure.

2.1 Web Servers

Web Servers - often running server-side scripts (e.g. PHP and ASP) - are used to deliver HTML documents combined with images, videos, scripts and style sheets. IIS and Apache Tomcat are the two web servers used in this study.

2.1.1 Internet Information Services (IIS)

IIS is a web server developed by Microsoft Inc. and is included in Windows Server 2008 R2 (and a few versions of Windows 7). Currently, on version 7.5, the web server supports access via HTTP, HTTPS, FTP, FTPS, SMTP and NNTP and in addition IIS also allows command line administration via PowerShell. IIS builds upon a modular architecture which allows additional extensions to be installed without additional costs. Security wise, the applications are isolated from each other in a sandbox-like style called Application Pools. [3][4]

2.1.2 Apache Tomcat

Apache Tomcat is an open source implementation - released under Apache License version 2 - of JSP (JavaServer Pages) and Java Servlets. Tomcat requires a Java Runtime Environment (JRE) to be installed and supports access via Apache JServ Protocol (AJP) and the HTTP/1.1 protocol. [5][6]

The web server consists of four main parts [7]:

- **Server** - The server represents a Catalina servlet container.
- **Connector** - The connector handles the communication with the client side via HTTP or AJP.
- **Engine** - The engine handles all requests coming from the connectors and returns them in a processed form.
- **Service** - The service manages all the connectors connected to a given engine.
2.2 Map Servers

Map servers are software applications used for rendering image-tiles from geo-spatial data. These image-tiles are then usually presented to the user via a web or desktop client. A brief description of the map servers used in the study now follows.

2.2.1 GeoServer

GeoServer is a Java-based map server, implementing the reference design of Open Geospatial Consortiums (OGC) Web Coverage Service (WCS), Web Feature Service (WFS) and Web Map Service (WMS) standards. The open source mapping library OpenLayers - described in section 2.4.1 - is included in the default installation. [8]

2.2.2 MapNik

MapNik is written in C++/Python and can be viewed as a drawing API for creating maps. This can be combined with Node.js and is then referred to as Node-MapNik [9].

2.2.3 MapServer

MapServer is an open source lightweight map server written in C/C++. The map server can be run as a CGI-script on a web server or with MapScript on SWIG (Simplified Wrapper and Interface Generator) [10]. The OpenLayers (see section 2.4.1) web interface is normally used to present the data rendered by means of MapServer. [11]

2.2.4 SharpMap

SharpMap is an opens source (GNU Lesser General Public License) map server written in C# and based on the .NET 4.0 platform.

2.3 Tile Caches

Tile caches are used in between map servers and the web/desktop client to increase the performance by caching the tiles and thus avoiding re-rendering of the tiles for every request. The following is a brief description of the tile caches used in the study.

2.3.1 GeoWebCache

GeoWebCache is an open source tile cache formed as a Java web application. The cache accepts most common tile sources (map servers). [12]

2.3.2 MapCache

MapCache is the tile cache included in MapServer (see section 2.2.3). The cache can be installed as either an Apache-specific module or as a CGI/FastCGI script for general web server installation. [13]

2.3.3 TileCache

TileCache is a BSD-licensed, Python-based, WMS-C/TMS server. The cache is run as a python CGI-script on a web server. [14]
2.4 Web Interfaces

The most commonly used web interfaces/APIs for GIS front-ends are: OpenLayers, Polymaps, Bing Maps and Google Maps.

2.4.1 OpenLayers

This JavaScript-library - released in 2006 - is mainly used for displaying and rendering maps on web pages. *OpenLayers* is an open source product released under a modified *FreeBSD License*. [15]

2.4.2 Polymaps

*Polymaps* - similar to *OpenLayers* described above - is a JavaScript-library for vector/image-tiled maps (using *Scalable Vector Graphics - SVG*). [16]

2.4.3 Bing Maps

*Bing Maps* is Microsoft’s own map service that offers developer APIs for building mobile, web and map applications. [17].

Some initial connections for Bing Maps and Windows Azure have been produced, for example, the new support for spatial data in Azure. [18][19]

2.4.4 Google Maps

*Google Maps* offers similar services to those of *Bing Maps* i.e. web-based map services and APIs. [20]

2.5 Cloud Computing

The term *cloud computing* is considered relatively new even though the concept itself is far from new. Cloud computing can be divided into three subcategories depending on the level of abstraction the provider and the consumer require.

2.5.1 Infrastructure as a Service (IaaS)

IaaS is where the service only consists of a distributed infrastructure (i.e. computational power, storage and network infrastructure) and where the customer is required to install, update and administer the *operating system* (OS) as well as the applications which run on top. Examples of this include Amazon EC2 and VMWare vCloud [21]

2.5.2 Platform as a Service (PaaS)

If the abstraction layer is moved one step up the concept is called Platform as a Service. In this case the provider administers both the hardware and the platform running on top of it. For example, Windows Azure is considered to be a PaaS in which a variant of Windows Server 2008 R2 runs on top of the hardware. [21]
2.5.3 Software as a Service (SaaS)

The top abstraction layer, for which the provider handles the upper tier software in addition to the hardware and platform. In this case, for the majority of the time, the consumer is the end client i.e. there is no middle-man between the service provider and the end client. Examples of this type include Hotmail, Google Docs, Office 365 and Facebook. [21]

2.6 Windows Azure

Windows Azure is a PaaS cloud computing platform offered by Microsoft Inc. and which was officially released on February 1, 2010 [22]. Many new features have been added, upgraded and refined since the release [23]. Figures 2.1 and 2.2 provide an overview of the different parts of Windows Azure and how they are semantically divided. In the following sections each of the subparts of the platform will be described in detail.

![Figure 2.1: Overview illustrating all the main components in Windows Azure.](image)

2.6.1 Compute

Windows Azure Compute consists of virtual machines (VMs) that can take three different roles, namely the Web Role, Worker Role and Virtual Machine Role (VM Role), on top of which the hosted application runs (as can be seen in figure 2.3). These roles come with different prerequisites and semantics, and are thus suitable for different applications. [24][25]

There are three rules that must be fulfill in order for the applications to run as intended i.e. a Windows Azure application should:

- be built from one or more roles.
- run multiple instances of each role (at least two for the SLA to be valid).
CHAPTER 2. THEORY

Figure 2.2: Structural overview presenting the layering in Windows Azure.

Figure 2.3: Overview of the different parts of Windows Azure Compute.
behave correctly when any role instance fails.

Nott should be noted that Windows Azure does not actually enforce the rules mentioned above, but the underlying platform assumes that the application follows all three.

2.6.1.1 Virtual Machines (VMs)

The compute service relies on virtual machines to perform the underlying processing. There is no difference between the VMs used for the different roles except for the amount of pre-installed software (unique for each role). [26]

When requesting a new VM (for a role instance) the fabric controller - which is described in detail under section 2.6.2 (Fabric) - boots up a VM, makes a fresh installation of Windows Server 2008 R2 (64-bit), installs the prerequisite software depending on role type, and finally deploys the application. The same procedure is performed on VM failure i.e. the fabric controller starts a new VM, makes a fresh Windows installation and deploys the application. This forces the developer to rethink how to store application data, since local data can be deleted at any time due to VM failure i.e. application critical data should NOT be stored in the VM local storage. More information can be found regarding this aspect under section 2.6.3.2 (Windows Azure Storage - Local VM Storage). [24][26][27]

Information regarding the underlying hardware on which the VMs are run, can be found in section 2.6.8 (Hardware).

Fabric Agent
On every VM instance a fabric agent exists which is in charge of delivering close to real-time information regarding the system health, application responsiveness and instance load to the Windows Azure Fabric (i.e. the fabric controller). [24][25]

A more detailed description of the fabric agent is given under section 2.6.2.1 (Fabric Controller).

2.6.1.2 Roles

Windows Azure offers three different role options, namely Web Role, Worker Role and VM Role. These roles are conceptually the same but are shaped in a slightly different manner from each other as there are different intentions for each of them. As the name suggests, the Web Role is intended for hosting web applications, while the worker role can be used as a background service together with a web front-end (Web Role) or for heavy-duty parallel computing. Finally the VM Role is most suitable when an application is unable to be automated and requires a large number of OS customizations on the VM. The developer is free to mix these roles as seen fit. [24][25][26][28]

Due to the non-persistent nature of the VMs, the developer must consider that all roles are required to be stateless i.e. no application critical data should be stored locally [24]. In addition, every role should be run on at least two instances for the SLA to be fulfilled. [29]

Web Role
The web role arrives with a pre-installed IIS 7 web server and is most suitable for web applications. The role supports the use of ASP.NET, Windows Communication Foundation (WCF), native code (e.g. Java, PHP, Node.js) or almost any other web technologies. [24][26][30]
Worker Role
The worker role, as compared to that of the web role, has no pre-installed web server and is mainly targeted towards heavy-duty applications running arbitrary code [26]. It is possible to install web servers and Java Runtime Environment (JRE) on the worker role, taking the statelessness into account [31].

VM Role
The VM role is, in many ways, similar to the standard definition of a VM, allowing developers to deploy a custom image of Windows Server 2008 R2 (Standard/Enterprise). It should be noted that the VM Role is not exactly the same as a standard VM since it is stateless. The process of uploading the image starts with conducting a local on-premises installation, which involves installing the required software and then uploading it to Windows Azure. Management can then be performed on the install after it has been uploaded. The significant difference between the VM role and the web/worker roles is that, in this case, the developer has full control of the OS. This comes at the cost of having to administer and update the OS instead of allowing this to be taken care of by Azure. The license for the hosted VMs are included in the compute-hour price and is not related to the license used for creating the VHD. [32][33]

The VM role is most suitable for applications with long or fragile installs, or when many manual configurations are required. [34]

2.6.1.3 Load Balancing
When running two or more instances of the same role, Windows Azure automatically load balances the incoming traffic across all live instances. However, due to the statelessness of the roles, affinity load balancing\(^1\) is not used and this, in turn, makes Windows Azure unsuitable for sticky sessions. [24][26][30]

The live IP-address given to the application points to the load balancer and not to the actual application. The applications are in fact hiding under virtual IPs behind the load balancer. [35]

The load balancer uses an uneditable session timeout of 60 seconds but, by using a type of session heartbeat it becomes possible to avoid the session timeout. [35]

Windows Azure Traffic Manager (WATM)
The Windows Azure Traffic manager allows the traffic to be distributed both geographically (across different data centers) and across applications (across different applications in your subscription). There are three different distribution policies: [36][37]

- **Performance** - The traffic is routed to the host with the lowest latency.
- **Round-Robin** - The traffic is evenly routed to all hosts.
- **Failover** - The traffic is routed to a predefined host (following a predefined list). If the primary host fails, the next host on the list is chosen.

\(^1\)Affinity load balancing is a technique that enables the load balancer to remember which role was chosen for a certain client, and all subsequent requests are then directed to the same role again.
CHAPTER 2. THEORY

2.6.1.4 VM modes (Staging & Production)

The virtual machines can be set to two different modes namely, production (public) and staging (non-public). [25]

**Staging** - In the staging mode the VM is non-public i.e. only the developer has access via a virtual IP (VIP), making it perfect for in-place testing. [25]

**Production** - The production mode is used when the application is ready to be launched on the Internet. A global URL (<yourApplicationID>.cloudapp.net), which is pointing at the load balancer, is connected via the VIP to the application. The DNS can also be redirected to a domain name specified by the developer (e.g. http://www.yourdomain.com). [25]

2.6.1.5 Service License Agreement (SLA)

The Compute service guarantees a 99.95% monthly SLA given that two or more role instances are deployed i.e. the service should be available 99.95% of the time or a refund will be given. [29]

2.6.2 Fabric

The Windows Azure Fabric is an umbrella name for all underlaying hardware residing the Azure platform. This includes all server racks, load balancers, switches, routers, cables as well as deployment services, cluster management and all the virtual machines. The fabric controller can be viewed as the Windows Azure kernel which monitors, coordinates and administers all these resources. [25]

2.6.2.1 Fabric Controller

The fabric controller manages the connection of the fabric layer with the compute and storage layers, directing which hardware resources are dedicated to each application instance. All communication between the fabric controller and the application instances is conducted via a fabric agent (see section 2.6.1.1 and 2.6.2.1 - Fabric Agent) that is installed in the instantiation phase on all virtual machines. The fabric agent keeps track of the current state and goal state of the running service, forwarding this information to the fabric controller which then takes appropriate measures. The fabric controller uses an internal state machine that compares to the current state given by the fabric agent and that specified in the configuration file, managing the service life cycle accordingly. This loop is referred to as a heartbeat and is the main loop of the state machine. The service life cycle consists of four stages: resource allocation, provisioning, deployment and upgrading, and maintenance. [25][28][38][39]

**Configuration file**

When developing an application an XML configuration file is created, holding information with regards to the required number of VM instances, the VM types and the general application settings. This configuration file is read by the fabric controller which orchestrates the VMs accordingly (e.g. boots up correct number of VMs, creates VM Roles and handles failover). The configuration file can be altered via the management portal (see section 2.6.6.1), which can easily change the number of VM instances. [25][40][41][42]
**Fabric Agent**

The fabric agent is the VM instances interface towards the fabric controller. The fabric agent keeps track of the applications status; goal state (i.e. the state specified in the configuration file) and current state, which is the real time status of the instance (e.g. running, inoperable, initializing or idle). The Windows Azure Storage also has a fabric agent attached to it, keeping track of the storage availability (the fabric controller sees the storage as merely another service running on a VM). \[25\][28][38]

### 2.6.2.2 Fault domains

The fault domains are a physical tightly coupled group of servers in the same data center, which are in the risk zone in case a single node fails (i.e. one failure can affect the whole domain). The fabric controller spreads the VM instances across several fault domains to increase resistance to hardware failure. \[25\]

### 2.6.2.3 Application Updates

When updating an application two choices are available, namely, in-place or VIP swap (Virtual IP swap). In a similar manner to that of the fault domains logical groups exists for updating which are called upgrade domains. \[43\]

**In-place**

The in-place update concept consists of four steps \[43\]:

1. **Stop** - The fabric controller stops all instances in a upgrade domain.
2. **Update** - Updates are performed.
3. **Start** - In the final step the fabric controller starts all the newly updated instances.
4. **Next domain** - The controller returns to step one, continuing to the next upgrade domain.

By following this approach the service, from a consumer point of view, is accessible throughout the whole update (with a temporal reduction in computational power). \[43\]

**VIP Swap**

If the updates require in-place testing before release or the system is sensitive to temporal loss of computational power, VIP Swap (Virtual IP Swap) is the option of choice. The two different VM modes, staging and production, are used for updating the instances (see section 2.6.1.4 for detailed description). \[25\][44][45]

1. **Start** - The fabric controller winds-up fresh VMs (set to staging mode), equal to the amount in the upgrade domain. The new VMs are installed with the latest update and appointed with a non-public virtual IP (see section 2.6.1.3). These VMs can now be accessed and tested by the developer in-place, thus enabling the detection of any bugs not discovered on the local development platform. \[44\][45]
2. **VIP Swap** - Once the developer is satisfied with the updated version, a simple VIP swap is performed between the *staging* and *production* VMs, redirecting the traffic to the newly updated version. [44][45]

3. **Stop** - In the final step, the developer can order the *fabric controller* to stop and to delete the *staging* VMs (those with the non-update application). [44][45]

In this case, from a consumer point of view, the updates will not be noticed i.e. no computation power is lost during any time of the update.

### 2.6.3 Storage

The data storage offered through Windows Azure consists of two different solutions, *SQL Azure* (a relational database) and *Azure Storage* (which can be further subdivided into, BLOB storage, Queues, Tables storage and Azure Drive). This can be seen in figure 2.4. [46]

![Figure 2.4: Overview illustrating the available storage options and functionality.](image)

#### 2.6.3.1 SQL Azure

SQL Azure is a full feathered cloud-based relational *database* (DB) which has evolved from Microsoft SQL Server 2008 and is built on top of the Microsoft CloudDB platform. SQL Azure databases can be developed using Microsoft Visual Studio or SQL Server Management Studio. Administration can be performed using SQL PowerShell, SQL Server Management Studio or by using programmatic access via SQL Server Management Objects (SMO). The DB can be accessed through common technologies such as ODBC, JDBC and ADO.NET. SQL Azure supports all non-deprecated SQL Server 2008 data types. [47][48][49][50]

High scalability is one of the key features associated with SQL Azure, allowing for elastic scaling to 150 GB. For information regarding sizing see section 2.6.8 (Hardware).
Spatial Data Types
At the end of June 2011 the Windows Azure Team presented the face that spatial data types are now supported in SQL Azure [51]. For a detailed list in relation to supported types see [52].

SQL Azure Data Sync
SQL Azure Data Sync, which is built upon Microsoft Sync Framework, is used to synchronize data between SQL Azure and in-house databases, or in between two or more existing SQL Azure/SQL Server databases (Sync Groups). The Sync Group uses a hub-spoke topology in which one of the SQL Azure databases is the hub and the remainder are spokes. Data Sync supports both uni-directional and bi-directional transactions, for which all sensitive data and connections are encrypted using SSL. An agent software must be installed for on-premises clients. [47][53]

Data types supported in SQL Azure Data Sync can be seen in “SQL Azure Data Sync - Supported SQL Azure Data Types” [54].

SQL Azure Reporting
SQL Azure Reporting is a reporting service built on SQL Server Reporting Services. In the Windows Azure Management Portal it is possible to find the Microsoft SQL Azure Reporting Community Technology Preview (CTP). [55][56]

SQL Azure Federations
The federation concept consists of partitioning a database and splitting it over several database nodes and, thus, also dividing the amount of requests each node will receive. The federation consists of a federation root and one or more federation members. This concept is usually called ‘Sharding’ and is where higher scalability and performance are reached through horizontally partitioning of one or more tables, row by row, across multiple federation members. The federation schema describes the manner in which the data is distributed across the database tables included in the federation and these are called federation tables. Inside these tables a federation member key (federation distribution key) specifies the range of values that each federation member is in charge of. A federation atomic unit is the subset of all rows inside a federation table that match a federation distribution key. All tables not included in the federation are called reference tables. An illustration of a federated DB can be seen in figure 2.5. [57][58]

Using federated databases there are certain aspects that must be consider during the design phase. First of all, carefully choice must be given to the value to federate on and secondly, due to the physical partitioning of data, join operations across databases are not natively supported. [57]

Service License Agreement (SLA)
Microsoft guarantees a 99.9% monthly SLA, for SQL Azure, where time is measured in five minute intervals and the service is deemed to be unavailable if one connection is rejected. [29]

2.6.3.2 Windows Azure Storage
The Windows Azure storage mainly consists of four different options: BLOB storage, Queues, Table storage and Windows Azure Drives. In addition, it is possible to count the
VMs local storage as part of the Azure Storage even though it is non-persistent.

Attributes
There are three common attributes for the storage options under Azure Storage:

- **Region** - Where in the world the data - geographically - will be stored.
- **Quota** - How much entitlement there is to space within the given storage option.
- **Access Keys** - How to securely access the data using a access key.

**BLOB Storage**
The BLOB (Binary Large OBject) storage consists of containers, which, in turn, holds one or more BLOBs. A BLOB consists of a raw byte array to which it is possible map a set of meta data, seen in figure 2.6. The containers can be in one of two states i.e. private or public. In the private state only the account holder has access to the BLOBs URLs while in the public state, the BLOBs can be accessed over the Internet. The maximum total size for all containers is 100TB. [59][60]

With Azure BLOB Storage there is the ability to use Content Delivery Networking (CDN) to cache the BLOBs closer to the end users and thus increasing the performance and reducing the load on the BLOB storage. For more information regarding CDN see section 2.6.4.2. [61][62]

**Queues**
The Windows Azure Queue works in a similar manner to the standard abstract data type it is most related to, following the FIFO (First-In-First-Out) concept. The queue is mainly used for storing messages, which can be reached via an authenticated HTTP/HTTPS
Figure 2.6: Azure BLOB Storage overview.

REST interface. The queue can handle up to 100TB in which each message has a maximum size of 64KB. This offers the ability to store roughly 1.5 billion messages simultaneously. The messages can be stored for a maximum of one week before the garbage collection deletes them. [63][64]

The nature of these queues makes them suitable as the connection between, for example, web front-ends (Web Role) and background services (i.e., a producer/consumer semantic), and thus making it very easy to scale both sides.

**Table Storage**

Azure Table Storage is a NOSQL (Not Only SQL) storage suitable for large amounts of non-relational structured data. The Table Storage contains tables which, in turn, hold entities consisting of properties. The entities have a maximum size of 1MB and can hold up to 252 properties (plus three system properties namely, partition key, row key and timestamp). The tables are accessed via the OData protocol, LINQ (Language Integrated Query) queries, ADO.NET Data Services and REST. [65][66]

**Windows Azure Drives**

Windows Azure Drive is a virtual hard drive (VHD) formatted by means of NTFS and stored in a BLOB. The persistent VHD is then cached on the local file system making it suitable for running applications that must maintain state e.g., third-party database applications. Access to the drive is conducted via existing NTFS APIs. [46][67][68][69]
Local VM Storage
Every virtual machine, independent of role-type, has a local storage (virtual hard drive) connected to it. Due to the non-persistent nature (i.e. all data is lost on VM failure), this storage should be used with great care. The size of the storage depends on the VM size and can be found in section 2.6.8. Since the OS on the VM is Windows Server 2008 R2 the storage is formatted using NTFS. [46]

Service License Agreement (SLA)
Microsoft guarantees a 99.9% monthly SLA with regards to the fact that the requests involving add, update, read and delete data are successfully performed (given correct formatting). [29]

2.6.3.3 Replication
All data stored in Windows Azure BLOBs, Queues and Tables are replicated three times across different fault domains in the same data center, thus ensuring a high availability due to automatic failover. In addition, the replicas are used for load balancing which involves distributing the incoming requests across all three copies, thus significantly increasing the number of concurrent I/O's. [70]

SQL Azure also uses replication with one primary and two secondary copies of the database, spread across different fault domains. Once a primary database goes down, an instant automatic failover is performed, redirecting all traffic to one of the secondary replicas i.e. no load balancing is performed across the replicas. [50]

GEO-Replication
Geo-replication is the concept of retaining an up-to-date copy of data in two or more geographically separated places. At the moment, Windows Azure BLOB and Table storage are geo-replicated to an additional data center located on the same continent e.g. having chosen the 'North Europe' data center for storing the BLOBs, they will also be replicated, free of charge, to the 'West Europe' data center. This will, in a similar manner to the replication described above, greatly reduce the chance of data loss due to a major disaster that affects an entire data center (e.g. power failure, natural disaster and fires). [71]

The Windows Azure Storage Team are currently working on adding geo-replication to the other storage options available in Windows Azure. [70]

2.6.4 Networking

2.6.4.1 Windows Azure Connect
Windows Azure Connect enables the possibility to connect in-house solutions (applications, databases and VMs) with Windows Azure Roles using the end-to-end IPSEC [72] protocol. An software agent is installed on the in-house machine, allowing developers to pass the company firewall without having to add access permission. [73][74][75]
2.6.4.2 Windows Azure Content Delivery Network (CDN)

The Windows Azure CDN is a geographically distributed caching system, holding copies of static data closer to the end users and thus reducing access times and increasing performance. The data can be served both via HTTP and HTTPS. The CDN supports both data from the Azure Storage (BLOBs) and the static output of compute instances. At the time of writing, Windows Azure offers 26 CDN access points across the world [76]. Data stored in the CDN is given a TTL (Time To Live), in which, by default, is set to 72h, before an updated attempt is made. [61][62][77][78]

Service License Agreement (SLA)

Microsoft guarantees a 99.9% monthly SLA that the CDN will respond to client requests and deliver the data requested. [29]

2.6.4.3 AppFabric Caching

In April 2011 Microsoft released the Windows Azure AppFabric Caching service, which is a cloud-based caching service targeted for Windows Azure applications [79]. Compared to a “normal” cache, the application, in this case, is able to decide what items are to be stored in the cache using Add/Put and Get. The default TTL (Time To Live) for cached items is set to 48 hours, but can be changed when adding the items to the cache [80]. Items in the cache must be under 8MB in size when in serialized form. An example of usage is storing session state or a page output for ASP.NET applications. [81][82][83][84][85]

It is also possible to enable caching on the local machine to assist the cloud cache. The TTL in the local cache is five minutes by default and is able to support up to 1000 objects. The positive effect is the increase in performance but, due to the difference in performance, the cloud cache and the local cache might become out of sync. [81]

Service License Agreement (SLA)

Microsoft guarantees a 99.9% monthly SLA for the connection between the caching endpoints and their Internet gateway. [29]

2.6.4.4 AppFabric Access Control Server (ACS)

The Windows Azure AppFabric Access Control Server is a authorization and authentication layer, separated from the application code itself. The access control server is administered via the management portal (see section 2.6.6.1). [86][87][88]

The ACS has support for the following features [89]:

- Windows Identity Foundation (WIF) integration.
- Web identity providers (e.g., Google, Windows Live ID, Yahoo, and Facebook).
- Active Directory Federation Services (AD FS) 2.0.
- OAuth 2.0 (draft 10), WS-Trust and WS-Federation protocols.
- Token formats: SAML 1.1, SAML 2.0 and Simple Web Token (SWT).
- Integrated and customizable Home Realm Discovery allowing for selectable identity provider.
CHAPTER 2. THEORY

- OData based management service providing programmatic access to the ACS configuration.

Service License Agreement (SLA)
Microsoft guarantees a 99.9% monthly SLA for the connection between a customer service endpoint and their Internet gateway. [29]

2.6.4.5 AppFabric Service Bus
The AppFabric Service Bus is a message handling and connectivity feature used internally between Azure hosted applications as well as between hybrid solutions. The main area of usage for the service bus is in relation to brokered (Subscriptions, Queues and Topics) and relayed messaging as well as endpoint connectivity. By using the service bus it is possible to expose WCF-based web services, without having to open ports in the firewall, sitting behind NAT (Network Address Translation). [90][91][92]

The three different relaying modes supported are [93]:

- Direct (One-way)
- Request/Response
- Peer-to-peer

Service License Agreement (SLA)
Microsoft guarantees a 99.9% monthly SLA for the connection between a customer service endpoint and their Internet gateway. [29]

2.6.4.6 AppFabric Integration
The Windows Azure AppFabric Integration was released for connecting Windows Azure with third party applications [94]. For example, Microsoft Dynamics CRM has been given support for registering plugins that can send runtime data to Windows Azure applications [95][96].

2.6.5 Example Solutions
A few case examples regarding how Windows Azure can be used in different scenarios, utilizing the different features now follows.

2.6.5.1 Pure Azure applications
Imagine a ticket ordering system that requires a web front-end presenting the user with different options, a background service taking care of placing the orders, and a database storing the order information. This model could easily be transferred to the web/worker roles and a SQL Azure DB, using the following concept:

- Web front-end -> Windows Azure Web Role
- Background service -> Windows Azure Worker Role
• Database -> SQL Azure

The internal role communication could be handled in a number of different ways depending on the scale of the system and performance requirements (see section 3.3 for more information regarding internal communication options). This system could easily be scaled to larger proportions if the requirement for handling an increased number of users occurs (e.g. release day of highly sought after tickets).

To reduce the load on the web front-end it is possible to initiate the use of Windows Azure CDN which will cache the static requests at a closer proximity to the user, greatly increasing the performance.

2.6.5.2 Hybrid applications

This example is a extension of the previous example with an on-premises administration system used for adding/editing tickets. This on-premises system could be securely connected and integrated using Windows Azure Connect and AppFabric Service Bus.

2.6.5.3 Mobile applications

In relation to mobile application this refers to occasionally-connected clients, such as smartphones, tablets or laptops. In this case a system could be constructed so that it distributes/pushes event notifications or data to these clients whenever they are connected.

2.6.5.4 Loosely coupled applications

Windows Azure could be used to join loosely coupled applications together by removing direct dependencies, connections, and acting as a broker in between the different modular system components.

2.6.5.5 High Performance Computing

Azure contains features such as high scalability, database federation (SQL Azure Federations), and load balancing, making it suitable for high performance computing. For example, having hundreds or thousands of worker roles working towards a greatly federated SQL Azure database, processing the requests incoming from the load balancers.

2.6.6 Administration Tools

The main administrative tool used for handling Windows Azure is the Windows Azure Management Portal (https://windows.azure.com).

2.6.6.1 Windows Azure Management Portal

The portal is a Silverlight based web interface, which was launched on 30th October 2010 [97] and where it is possible to monitor, control and scale the applications run in the Azure cloud. [98]
2.6.6.2 Microsoft Hypervisor (Hyper-V) Server

The hypervisor-based Microsoft Hyper-V Server is used for virtualizing operating systems. In Windows Azure it is used for creating the custom Windows Server 2008 R2 VHD used in the VM role (see section 2.6.1.2). [99][100]

2.6.6.3 Windows Azure VM Assistant

The Windows Azure VM Assistant is a software run inside the VM, providing information about the Role instance (e.g. role details, instance health information). [101]

2.6.7 Development Tools & Languages

The Windows Azure platform allows for development using a variety of different languages, tools and frameworks. The main supported languages and platforms are: .NET, Node.js, Java and PHP and for each of these languages a Windows Azure Client Library exists.

2.6.7.1 Windows Azure SDK

The Windows Azure Software Development Kit (SDK) can be found here [102].

2.6.7.2 Windows Azure Tools

The Windows Azure Tools is a Visual Studio extension allowing for the creation and uploading of Windows Azure projects. The download also includes the Windows Azure SDK mentioned above. [103]

2.6.7.3 .NET

Microsoft’s software platform, .NET, presents the opportunity to use an additional set of languages to write applications for the Windows Azure platform and in this case it is possible to use C/C++, C#, Visual Basic (VB), F# and J#. [104][105]

2.6.7.4 Node.js

This non-interpreted JavaScript-based software system was released in 2009 and has grown significantly during the last six months [106]. The main area of this event-driven system is its scalable web servers and the Windows Azure platform provides extensive support for this technology. [107][108]

2.6.7.5 Java

Microsoft gives extensive support for using the object-oriented language Java in Windows Azure. [109][110]

2.6.7.6 PHP

PHP, one of the largest server-side scripting languages, is also given native support in Windows Azure. [111][112]
2.6.8 Hardware

Azure allows different hardware options for the virtual machines that can be changed depending on current needs. The different setups available at the time of this study can be seen in table 2.1.

<table>
<thead>
<tr>
<th>VM Size</th>
<th>CPU Cores</th>
<th>Memory</th>
<th>Web &amp; Worker Roles</th>
<th>VM Role</th>
<th>Bandwidth (Mbps)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ExtraSmall</td>
<td>Shared</td>
<td>768 MB</td>
<td>19,480 MB</td>
<td>20 GB</td>
<td>5</td>
</tr>
<tr>
<td>Small</td>
<td>1</td>
<td>1.75 GB</td>
<td>229,400 MB</td>
<td>165 GB</td>
<td>100</td>
</tr>
<tr>
<td>Medium</td>
<td>2</td>
<td>3.5 GB</td>
<td>500,760 MB</td>
<td>340 GB</td>
<td>200</td>
</tr>
<tr>
<td>Large</td>
<td>4</td>
<td>7 GB</td>
<td>1,023,000 MB</td>
<td>850 GB</td>
<td>400</td>
</tr>
<tr>
<td>ExtraLarge</td>
<td>8</td>
<td>14 GB</td>
<td>2,087,960 MB</td>
<td>1890 GB</td>
<td>800</td>
</tr>
</tbody>
</table>

At the beginning of the study, the CPU clock was, according to Microsoft, set at 1.6 GHz. This number was later removed completely from the Azure homepage where can no longer be found. The reason for this is probably because that Microsoft is upgrading the hardware and thus cannot ensure that all hardwares are exactly the same. Proof of this can be found here [27][114].

For hardware performance see [115].

2.6.9 Migration to Windows Azure

When migrating applications to the Azure, the conditions presented in section 2.6.1 (Compute) must be fulfilled. Microsoft have produced a few tools to aid developers in migrating to Azure. To evaluate the amount of work necessary for a successful migration the Windows Azure Migration Assessment Tool can be used.

More information on migration can be found here [116][117].

2.6.9.1 Windows Azure Migration Assessment Tool (MAT)

By answering a set of binary questions, the assessment tool makes an estimation regarding the required work for a successful migration. [118]

2.6.9.2 SQL Azure Migration Wizard

The amount of work required in relation to migrating databases to SQL Azure mainly depends on what type of database is currently being used. For example, if Microsoft SQL Server 2008 is used, the transition to SQL Azure is straightforward while other DBs require more work. [119][120]

2.7 Related Cloud Computing Services

There are other actors in the market that provide similar services to those of Windows Azure, with Amazon, Google and SalesForce being the main contenders.
CHAPTER 2. THEORY

2.7.1 Amazon Cloud

Amazon Cloud is the popular speech for the combined service of the Amazon Elastic Cloud (EC2) and Amazon Simple Storage i.e. compute and storage services.

2.7.1.1 Amazon Elastic Cloud (EC2)

Amazon Elastic Cloud was released as beta in August 2006 [2] and where it is possible to rent and host applications on virtual machines spread throughout the world. EC2 is considered an Infrastructure as a Service (IaaS) i.e. one abstraction layer lower than Windows Azure. [121]

2.7.1.2 Amazon Simple Storage System (S3)

This is Amazon’s variant of Windows Azure Storage with similar functionality. Data is stored in buckets containing data objects from 1 B to 5 TB. The storage points of these buckets can be designated to points all over the world but the closest to the customer offers reduced spatial latency. The data is retrieved via HTTP over RESTful or SOAP based services and then accessed by means of a unique developer-assigned key. [122]

2.7.2 Google App Engine

Google App Engine is a PaaS concept where it is possible to host web applications built in Go, Java, JavaScript, Python and Ruby. In a similar manner to that for Windows Azure payment is only required for what has been used. Data is stored in App Engine as schemeless data objects. Google offers a free start-up subscription offering 1 GB of storage and up to 5 million page views per month, which is unique among the four actors described here. [123]

2.7.3 Force.com

Force.com is SalesForce[124] SaaS cloud computing service, mainly targeted for social enterprise applications. The service support web services and RESTful API’s. [125]

2.7.4 IBM SmartCloud

IBM SmartCloud offers services at all layers mentioned in section 2.5 (Cloud Computing): IaaS, PaaS and SaaS. The services are offered through three different models: public, hybrid or private. [126]

2.8 Migration of Existing Application - Cykelreseplaneraren

The application migrated for the performance test is called Cykelreseplaneraren (i.e. “Cycling-route-planer”), and is at the moment only available in the Swedish city of Västerås. The system is currently hosted in the Amazon Cloud and, pre-migration, consists of the following components:

- Web front-end (ASP.NET MVC, OpenLayers) handling tile rendering.
- Map Server (SharpMap) running together with the web front-end.
• Secondary ASP.NET MVC application handling route requests.
• Windows Service (C#) performing route calculations.
• Two databases (SQLite) holding map and route data.

The current live version can be seen at [http://vasteras.cyklasakert.nu/].

2.9 Related work

Esri is one of the largest software companies producing competitive GIS solutions to those used in this thesis. In their upcoming release of the map server software ArcGIS server 10.1, they are implementing support for installation on the Windows Azure platform. This shows that the platform is of great commercial interest from a GIS point-of-view.

MatDotNet also sells an Azure/GIS solution in which they promote the economic and scalable advantages over the standard hosting solutions. [127]

Leaving aside the ArcGIS and MapDotNet solutions, there has been almost no relevant, publicly released, research conducted on the topic of Windows Azure and GIS compatibility with the exception of a few theoretical discussions. For example, three Indian researchers published an article in International Journal on Computer Science and Engineering (IJCSE) where they theoretically evaluate cloud computing (Amazon Cloud) and GIS and also offer a proposal for a multi-tiered architecture. [128]
Chapter 3

Methodology

In this chapter the tools used are discussed, how the problems were approached (test-implementations) and how the performance tests were conducted.

3.1 Tools

A description regarding the tools and how they were used in the study now follows.

3.1.1 Hardware

All on-premises development and virtualization were conducted on a laptop, Dell Latitude D630, which supports hardware virtualization (Intel Virtualization Technology) and Data Execution Prevention (DEP), in this case Intel XD bit (eXecute Disable).

The computer was run with dual-boot OS (Windows 7 Professional and Windows Server 2008 R2) with Windows Azure SDK 1.6 and the .NET 4.0 platform installed.

3.1.2 Development

The main IDE used for development was Microsoft Visual Studio 2010 Ultimate together with Windows Azure Tools & SDK, while the open-source IDE Eclipse (Galileo) was used for all Java development.

For all non-supported languages (in Visual Studio/Eclipse), the text editors, Sublime and Notepad++ were used.

To access the Azure BLOB Storage, at first a home-made command-line-based file manager was created, however a better existing solution was found and was used; Azure Blob Studio 2011.

For debugging the services, once they had been uploaded in the cloud, the IntelliTrace plugin was used together with Visual Studio and Windows Azure SDK.

3.2 Map Server and Tile Cache Implementation

The test implementation of the map servers and tile caches are performed to determine the problems that can be encountered and, if unsuccessful, suggest other implementation approaches.
CHAPTER 3. METHODOLOGY

3.2.1 VM Role Implementation

The VM Role instances were created using the following three guides [100][129][130], using Windows Server 2008 R2 Enterprise as OS.

All VM instances had the following software/frameworks installed;

- Windows Azure VM Role Integration Component (SDK 1.6)
- .NET 4.0
- IIS 7
- Java Runtime Environment (JRE) 1.6
- Apache Tomcat 7.0

A shared VHD was used to transfer data (e.g. compiled code and data sets) in between the Hyper-V host and client [131]. Once uploaded to Windows Azure, the VM instances were administered live via Windows Remote Desktop directly from the Management Portal [132].

Implementation specific details regarding the VM Role are presented in chapter 4 (Implementation).

3.2.2 Web & Worker Role Implementation

Implementation specific details regarding the Web/Worker Roles are presented in chapter 4 (Implementation).

3.3 Internal Role Scalability Options

At the time of writing, Windows Azure only supports load balancing on the external endpoints, i.e. no native support of load balancing in between internal endpoints. The scalability issue that is raised when targeting internal role communication is one of the aspects that must be considered when the system architecture is developed. These following four design options which relate to handing this issue, are presented in the result chapter:

- Direct Mapping
- Public Internal Endpoints
- Queues
- Worker Role Request Broker

The “Public Internal Endpoints” solution presented a problem regarding the possibility of being charged twice for the incoming traffic (for details see section 5.5.2). This led to a discussion with a Microsoft employee and Windows Azure Team member, Emil Gustafsson, regarding load balancing and scaling semantics in order to determine whether or not this is the case.
3.4 Application Migration & Performance Tests

The performance test conducted consisted of migrating an existing service (Cykelreseplaneraren), currently running in Amazon Cloud, to Windows Azure and then to perform the test based on a set of given test data.

3.4.1 Migration Architecture and Scalability Aspects

The migrated solution was split into three separate roles (seen in figure 3.1) across two different applications in order to increase the scalability. The two SQLite databases will be stored in Azure BLOB Storage and downloaded to the local VM storage during the Role initialization phase.

![Figure 3.1: System overview for Cykelreseplaneraren in Windows Azure. The TCP communication between the web role and the worker role is performed via “public Internal Endpoints” (Section 5.5.2).](image)

3.4.1.1 Scalability Aspects

The following are some of the scalability aspects taken into consideration when migrating the application:

- The SQLite databases are downloaded from the Azure BLOB Storage at role initialization, which enables easier updates in a highly scaled application (i.e. update file in BLOB Storage and re-image the roles instead of manually updating every role via RDP).
CHAPTER 3. METHODOLOGY

- Directory permissions are set in the Role initialization phase (i.e. inside the startup.cmd shell script) instead of updating every Role via RDP.
- For increased inter-role scalability, option 2 (Section 5.5.2 - Public Internal End-points) was chosen for the communication between the secondary ASP.NET MVC (WebRole : Routehandler) and the Windows Service (Worker Role : Windows Service).

3.4.2 ASP.NET MVC Web Front-end

The front-end migration was straightforward, converting the existing ASP.NET project to a Windows Azure Web Role, opening ports and setting directory permissions. [133]

3.4.3 ASP.NET MVC Routehandler

As with the web front-end the web back-end was converted to an Azure web role using the same procedures as mentioned previously.

3.4.4 Windows Service

To remove some of the extra complexity involved in installing a windows service in Azure, the windows service was stripped of the service installation. The code was then merged with an Azure worker role, the application settings rewritten to fit the Azure model and external libraries and dependencies changed or rebuilt to 64-bit solutions. The internal TCP-communication with the routehandler was exchanged to use the external load balancer to enable modular scalability for all the parts of the application.

3.4.5 VM Specification

The Azure VM instance running the performance tests are based on AMD Opteron 2373 EE (2.1GHz) [134].

3.4.6 The test conditions

The migrated solution was targeted by four different performance tests, with each test being run 1000 times. The test was conducted using a home-made testbed, which queries the routehandler over TCP.

3.4.6.1 Test 1: Request time differences between Azure VM & local VM

The test was conducted by measuring the response time for a query to be handled by the “windows service” (see figure 3.1). The query consists of two, randomly generated points, confined inside a predefined map area, in between which the route is calculated and returned. The requests are run 1000 times sequentially with a predefined delay (estimated from previous test run) in between repetitions in order to avoid any interference effects. The targeted systems is 1) a Windows Azure hosted (extra small VM size) worker role 2) an on-premises VM with very similar hardware configuration.

Aim: To determine the differences in request times which are probably due to: greater spatial distance (RTT), DNS-lookups and instabilities on the Internet affecting the routing.
3.4.6.2 Test 2: Heavy calculation requests on different VM-sizes

In this test the focus is to send heavy calculation requests and to measure the response times. As in the case of the previous test, the requests consists of two randomly generated points inside a predetermined confined area, with the addition of an extra 400'000 via-points to the route.

**Aim:** To see how the different role sizes handle the load from the incoming requests.

3.4.6.3 Test 3: Multiple concurrent request on different VM-sizes

Same test setup as *Test 1* but with 10 instances of the test-client running concurrently.

**Aim:** To see how the different role sizes handle many concurrent TCP-connections.

3.4.6.4 Test 4: Heavy calculation requests on scaled VM-solutions

This test is a mixture of test cases 2 and 3 and involves 10 concurrent instances of the test-client, each sending 1000 sequential requests containing 100’000 “via-points”. On the receiving side five different cases were tested with 1,2,3,4 and 5 load balanced worker roles, each hosting the “windows service”. The worker role size is set to “Small” to make it easier to distinguish the results (the “Extra small” was discarded due to that it shares one single CPU).

**Aim:** To see how well the incoming requests are load balanced in between the worker roles.
Chapter 4

Implementation

In this chapter the implementation of map servers and tile caches are described.

4.1 Map Server Implementations

The specialized installations, software modifications and measures performed are described under each Map Server.

4.1.1 MapServer

A pre-compiled version of MapServer was downloaded and used for both the VM and Worker Role installation.

4.1.1.1 VM Role

The MapServer Demo was installed and tested with regards to its success using the instructions in the documentation [11].

4.1.1.2 Worker Role

1. Downloaded the following software:
   - MapServer Windows Installation (ms4w_3.0.4.zip) [135]
   - GetFile.exe (Command-line-based software for downloading files) [136]
   - unzip.exe (Command-line-based software for unzipping files) [137]


3. a) In Visual Studio create a Azure Project with a Windows Azure Web Role (or Worker Role with IIS activated i.e IIS is presumed to be activated on the role).
   b) Add TCP Endpoints for ports 80 and 443 (i.e. Tcp80 and TcpSSL).
   c) Open WorkerRole.cs and in the end of the OnStart-function add the following lines:
TcpListener port80Listener = new TcpListener(RoleEnvironment.CurrentRoleInstance.InstanceEndpoints["Tcp80"].IPEndPoint);

TcpListener sslListener = new TcpListener(RoleEnvironment.CurrentRoleInstance.InstanceEndpoints["TcpSSL"].IPEndPoint);

port80Listener.Start();
sslListener.Start();

This will open listeners for incoming requests and forwards them to the defined endpoints.

d) Add the GetFiles.exe and unzip.exe to the project and in the file properties set “Copy to Output Directory” to “Copy if Newer” (This ensures that the file is included in the uploaded package and also removes redundant uploads if they are the same version).

e) Create a CMD Shell-script file (startup.cmd) in the worker role and set “Copy to Output Directory” to “Copy if newer”.

f) Open the CMD Shell-script file and add the following lines:

REM Downloads the ms4w-zip
start /w GetFiles http://trionatestapplications.blob.core.windows.net/mapserver/ms4w.zip C:\ms4w.zip

REM Unzips the file to C:\
start /w unzip C:\ms4w.zip C:\

REM Open port 80 for TCP connections for Apache
netsh firewall add portopening protocol=TCP port=80 name=Apache profile=CURRENT mode=ENABLE

REM Change the scripts active directory to C:\ms4w to call the batch scripts
cd /D "c:\ms4w"

REM Start the MapServer installation script
call apache-install.bat

REM Start script that sets environment variables for MapServer libraries and modules
call setenv.bat

This script will download the files from the BLOB Storage, unzip them to the correct location, set the PATH-variables and install PHP.

g) Open the service definition file (ServiceDefinition.csdef) and just under the section:

<WorkerRole name="WorkerRole" vmsize="VMSizeOfYourChoice">

, add the following row:

<Runtime executionContext="elevated" />
A startup task will also be added that will run the CMD Shell-script as follows:

```xml
<Startup>
  <Task commandLine="startup.cmd" executionContext="elevated"
        taskType="background" />
</Startup>
```

This provides the application with sufficient privilege to run the Windows Command shell-script.

h) Package and upload the solution to Windows Azure.

If the installation is successful, the verification can be conducted via the map server demo that should be able to be reached via `http://NAMEOFYOURAPPLICATION.cloudapp.net/cgi-bin/mapserv.exe`. Entering the URL in a browser should result in the following output: “No query information to decode. QUERY_STRING is set, but empty”.

### 4.1.2 SharpMap

The C# library was downloaded, imported into a Visual Studio project and compiled. The included demo was used to verify that the installation was successful [138].

**4.1.2.1 VM Role**

SharpMap was installed on the VHD using the included demo website.

**4.1.2.2 Worker Role**

SharpMap was used in the migrated application (see section 3.4 “Application Migration & Performance Tests”) as a C# library imported into the ASP.NET MVC project running on a web role.

### 4.1.3 GeoServer

GeoServer was downloaded as a Windows Installer, web archive (*.war), as well as the Java source code.

**4.1.3.1 VM Role**

The GeoServer was installed on the VHD using the Windows Installer and tested with the included demo. The steps followed for the installation can be found here [139].

**4.1.3.2 Worker Role**

The following steps were taken to install and test GeoServer in the Worker Role.

1. Downloaded the following software:
   - Java Runtime Environment (JRE 1.6)
   - Tomcat 7.0.27 (64-bit Windows ZIP)
   - GeoServer Web Archive (*.WAR)
CHAPTER 4. IMPLEMENTATION

- GetFile.exe (Command-line-based software for downloading files) [136]
- unzip.exe (Command-line-based software for unzipping files) [137]

2. a) Install the JRE on local development machine
   b) Go to the installation path (e.g. C:\Program Files\Java\)
   c) Zip the jre6 folder (e.g. jre6.zip)
   d) Upload the zip-file to Azure BLOB Storage using Azure Blob Studio 2011

3. a) Unzip Tomcat on local development machine
    b) Copy the GeoServer WAR-file to %tomcatroot%\webapps
    c) Copy the *.pfx file (PKCS12 certificate used for remote connection to the worker role) to %tomcatroot%\webapps
    d) Open the server.xml file (e.g. %tomcatroot%\conf\server.xml)
    e) Change ports for the Catalina service at the line “<connector port="8080" protocol="HTTP/1.1" connectionTimeout="20000" redirectPort="8443" />” from 8080 -> 80 and 8443 -> 443.
    f) Uncomment and change ports for the SSL connector at the line “<connector port="443" protocol="HTTP/1.1" SSLEnabled="true" maxThreads="150" scheme="https" secure="true" clientAuth="false" ss1Protocol="TLS" keystoreFile="\webapps\NAME_OF_YOUR_CERTIFICATE_FILE.pfx" keystorePass="" keystoreType="PKCS12" />” from 8443 -> 443 and the file-name in keystoreFile.
    g) Re-Zip the Tomcat folder (e.g. apache-tomcat-7.0.27.zip).
    h) Upload the zip-file to Azure BLOB Storage using Azure Blob Studio 2011.

4. a) In Visual Studio create an Azure Project with a Windows Azure Worker Role.
    b) Add TCP Endpoints for ports 80 and 443 (i.e. Tcp80 and TcpSSL).
    c) Open WorkerRole.cs and in the end of the OnStart-function add the following lines:

```
TcpListener port80Listener = new TcpListener(RoleEnvironment.CurrentRoleInstance.InstanceEndpoints["Tcp80"].IP Endpoint);

TcpListener sslListener = new TcpListener(RoleEnvironment.CurrentRoleInstance.InstanceEndpoints["TcpSSL"].IP Endpoint);

port80Listener.Start();

sslListener.Start();
```

```
Process p = new Process();
ProcessStartInfo psi = new ProcessStartInfo();
psi.FileName = "startup.cmd";
psi.CreateNoWindow = true;
p.StartInfo = psi;
p.Start();
p.WaitForExit();
```
This will open listeners for incoming requests and forwards them to the defined endpoints, then creates a new process that will start and hold a “session” for the startup-script created in the next step.

d) Add the `GetFiles.exe` and `unzip.exe` to the Worker Role and in the file properties set “Copy to Output Directory” to “Copy if Newer” (This ensures that the file is included in the uploaded package and also removes redundant uploads if they are the same version).

e) Create a CMD Shell-script file (startup.cmd) in the worker role and set “Copy to Output Directory” to “Copy Always”.

f) Open the CMD Shell-script file and add the following lines:

```bash
REM Downloads the Tomcat zip-file
start /w GetFiles http://trionatestapplications.blob.core.windows.net/geoserver/apache-tomcat-7.0.27.zip C:\apache-tomcat-7.0.27.zip

REM Unzips the file to C:\
start /w unzip.exe c:\apache-tomcat-7.0.27.zip -d c:\

REM Downloads the JRE zip-file
start /w GetFiles http://trionatestapplications.blob.core.windows.net/geoserver/jre6.zip C:\jre6.zip

REM Unzips the file to C:\
start /w unzip.exe c:\jre6.zip -d c:\

REM Sets environment variable for JRE
SET JRE_HOME=C:\jre6

REM Sets environment variable for Catalina
SET CATALINA_HOME=C:\apache-tomcat-7.0.27

REM Open port 80 for TCP connections for Tomcat
netsh firewall add portopening protocol=TCP port=80 name=TomcatGeoServer profile=CURRENT mode=ENABLE

REM Starts the Tomcat web server
C:\apache-tomcat-7.0.27\bin\startup.bat
```

This script will download the files from the BLOB Storage, unzip them to the correct location, set the PATH-variables, add an exception in the Windows Firewall, and start the Tomcat server. The Tomcat server will then automatically install the WAR-file that was previously added in the Tomcat webapps-folder.

g) Open the service definition file (`ServiceDefinition.csdef`) and just under the section:

```xml
<WorkerRole name="WorkerRole" vmsize="VMSizeOfYourChoice">
```

, add the following row:
<Runtime executionContext="elevated" />

This provides the application with sufficient privilege to run the Windows Command shell-script.

h) Package and upload the solution to Windows Azure (Make sure that the same certificate that was in Tomcat is used).

If the installation is successful, the verification can be performed via the map server demo that should be able to be reached via http://NAMEOFYOURAPPLICATION.cloudapp.net/geoserver/.

The amount of memory that Java has dedicated may have to be adjusted if very RAM-consuming applications are to be run.

4.1.4 MapNik

The latest stable version was cloned from the MapNik Git repository and due to MapNik’s requirement of Python on the target system, CPython 2.7.2 was installed. Mapnik was built according to the instructions given here [140].

4.1.4.1 VM Role

The MapNik was installed following this guide [141].

4.1.4.2 Worker Role

The following steps were taken to install and test GeoServer in the Worker Role.

1. Downloaded the following software:
   - Python (In this case Python27) MSI-installer
   - Mapnik source-code
   - GetFile.exe (Command-line-based software for downloading files) [136]
   - unzip.exe (Command-line-based software for unzipping files) [137]

2. Zip the output from the previous step (e.g. mapnik.zip).


5. a) In Visual Studio create an Azure Project with a Windows Azure Worker Role.
   b) Open WorkerRole.cs and in the end of the OnStart-function add the following lines:

   ```csharp
   Process p = new Process();
   ProcessStartInfo psi = new ProcessStartInfo();
   psi.FileName = "startup.cmd";
   psi.CreateNoWindow = true;
   p.StartInfo = psi;
   p.Start();
   p.WaitForExit();
   ```
This will create a new process that will start and hold a “session” for the startup-script created in the next step.

c) Add the GetFiles.exe and unzip.exe to the Worker Role and in the file properties set “Copy to Output Directory” to “Copy if Newer” (This ensures that the file is included in the uploaded package and also removes redundant uploads if they are the same version).

d) Create a CMD Shell-script file (startup.cmd) in the worker role and set “Copy to Output Directory” to “Copy if newer”.

e) Open the CMD Shell-script file and add the following lines:

```bash
REM Downloads the Python msi-installer
start /w GetFiles http://trionatestapplications.blob.core.windows.net/mapnik/python.msi C:\python.msi

REM Installs Python at C:\Python
start /w msiexec /i C:\python.msi /qn TARGETDIR=C:\Python\n
REM Set environment variable for Python
SET PATH=%PATH%;C:\Python\n
REM Downloads the MapNik zip-file
start /w GetFiles http://trionatestapplications.blob.core.windows.net/mapnik/mapnik.zip C:\mapnik.zip

REM Unzips the file to C:\
start /w unzip.exe C:\mapnik.zip -d C:\

REM Set environment variable for mapnik library
SET PATH=%PATH%;C:\mapnik\lib

REM Change the scripts active directory to properly call the demo executable
cd /D "c:\mapnik\demo\c++\"

REM Starts the demo
rundemo.exe ..\..\lib\mapnik
```

This script will download the files from the BLOB Storage, unzip them to the correct location, set the PATH-variables, install python.

f) Open the service definition file (ServiceDefinition.csdef) and just under the section:

```
<WorkerRole name="WorkerRole" vmsize="VMSizeOfYourChoice"/>
```

add the following row:

```
<Runtime executionContext="elevated" />
```

This provides the application with sufficient privilege to run the Windows Command shell-script.

g) Package and upload the solution to Windows Azure.
CHAPTER 4. IMPLEMENTATION

If the installation is successful, the Mapnik server has generated a few tiles of different file-types (pdf, png, svg, jpg and tif) in the demo folder (e.g. C:\mapnik\demo\c++\).

4.2 Tile Cache Implementations

The specialized installations, software modifications and measurements performed are described under each Tile Cache.

4.2.1 MapCache

The MapCache is included in the MapServer repository (See section 4.1.1).

4.2.1.1 VM Role

The MapCache was installed and tested using this guide [142].

4.2.1.2 Worker Role

The worker role installation was performed in combination with the MapServer map server using the same steps as in the MapServer installation with the following exceptions:

1. Downloaded the following software:
2. Unpack the zip-file on the development machine.
3. Open httpd.conf (located in .../ms4w/Apache/conf/)
4. Uncomment line 130 and change the line to:
   ```
   LoadModule mapcache_module "C:/ms4w/Apache/cgi-bin/mod_mapcache.dll"
   ```
5. Change the directory on line 370 to:
   ```
   MapCacheAlias /mapcache "C:/ms4w/apps/mapcache/mapcache.xml"
   ```
6. Open mapcache.xml (filepath can be seen above).
7. Change all lines containing directory mappings to start with:
   ```
   "C:/ms4w/..."
   ```
8. Rezip the MapServer files (i.e. mapserver.zip) and upload to Azure BLOB Storage (using Azure BLOB Studio 2011).

If the installation is successful, the verification can be performed via the MapCache demo that should be able to be reached via http://NAMEOFYOURAPPLICATION.cloudapp.net/mapcache/demo/. Opening and zooming in on any of the demos should generate background pictures that will be stored at “C:/ms4w/tmp/ms_tmp/cache/”.

CHAPTER 4. IMPLEMENTATION

4.2.2 TileCache

The TileCache can be run a few different ways, but the focus in this case is on running it on the two major web servers IIS and Apache.

4.2.2.1 VM Role

Before installation, Python had to be installed (Python 2.7.2) on the target system. The first installation-step was to add the CGI module in the IIS server, then the latest stable release was downloaded, and installed using this guide [143], with the exception that in IIS7 the python script is added via Handler Mappings -> Add Script Map. In the request path field put "*.py" as the script files extension and in Executable select "C:\Python27\Python.exe %s %s".

The cache-installation was verified using the test demo [14].

4.2.2.2 Worker Role

The worker role installation was performed in combination with the MapServer map server. The same steps where taken as in section 4.1.1.2 (MapServer - Worker Role) except for an addition described below.

1. Download the following software:
   - TileCache (ms-tilecache-ms4w-3.0beta10.zip) [144]
   - Gmap ms4w map package (Background maps) [145]

2. Upload the zip-files to Azure BLOB Storage using Azure Blob Studio 2011 (e.g. tilecache.zip and gmap.zip).

3. a) Open the CMD Shell-script file and add the following lines:
   ```
   REM Downloads the ms4w-zip
   start /w GetFiles http://trionatestapplications.blob.core.windows.net/mapserver/ms4w.zip C:\ms4w.zip

   REM Unzips the file to C:\
   start /w unzip C:\ms4w.zip C:\

   REM Downloads the tilecache-zip
   start /w GetFiles http://trionatestapplications.blob.core.windows.net/mapserver/tilecache.zip C:\tilecache.zip

   REM Unzips the file to C:\
   start /w unzip.exe C:\tilecache.zip -d C:\

   REM Downloads the gmap.zip
   start /w GetFiles http://trionatestapplications.blob.core.windows.net/mapserver/gmap.zip C:\gmap.zip

   REM Unzips the file to C:\
   start /w unzip.exe C:\gmap.zip -d C:\
   ```
REM Copys TileCache binaries to apache cgi/bin-folder
  copy "c:\ms4w\apps\tilecache\bin\*.*" "c:\ms4w\Apache\cgi-bin"

REM Open port 80 for TCP connections for Apache
  netsh firewall add portopening protocol=TCP port=80 name=Apache profile=CURRENT mode=ENABLE

REM Change the scripts active directory to C:\ms4w to call the batch scripts
  cd /D "c:\ms4w"

REM Start script that sets environment variables for MapServer libraries and modules
  call setenv.bat

REM Start the MapServer installation script
  call apache-install.bat

b) Package and upload the solution to Windows Azure in the same manner as with MapServer.

If the installation is successful, the verification can be performed via the TileCache demo that should be able to be reached via http://NAMEOFYOURAPPLICATION.cloudapp.net/tilecache/index.html or http://NAMEOFYOURAPPLICATION.cloudapp.net/tilecache/index.fcgi.html (for FastCGI).

If the background images are not showing in this case then go to http://NAMEOFYOURAPPLICATION.cloudapp.net/tilecache/index.html and click on the GMap Interface. If an error message similar to "Fatal error: Call to undefined function ms_GetVersion() in gmap75.inc.php" is received then there is a naming-error on the mapscript file. The solution is to change the filename to the current version of the application (e.g. change /ms4w/Apache/php/ext/php_mapscript_4.8.dll to /ms4w/Apache/php/ext/php_mapscript_4.10.0.dll) and then restart the Apache server. (e.g. change /ms4w/Apache/php/ext/php_mapscript_4.8.dll to /ms4w/Apache/php/ext/php_mapscript_4.10.0.dll) and then restart the Apache server.

4.2.3 GeoWebCache

The GeoWebCache was downloaded both as a web archive (*.war) and as Java source code.

4.2.3.1 VM Role

GeoWebCache was installed and verified on the VHD using this guide [146].

4.2.3.2 Worker Role

Since GeoWebCache is run via a Java WAR-file, the same procedure as in section 4.1.3.2 (GeoServer Worker Role Installation) was performed to successfully install the cache. The
only exception was in step 3b, that instead of using `geoserver.war`, `geowebcache.war` was added in `%tomcatroot%/webapps`. 
Chapter 5

Results

The outcome of the study i.e. results from the goals set in the introduction chapter now follows.

5.1 Map Servers on Windows Azure VM Role

The results regarding whether or not the respective map server was successfully installed and run on the Windows Azure VM Role are presented here.

5.1.1 MapServer

**Demand:** Is MapServer implementable on the Windows Azure VM Role?

**Outcome:** MapServer was successfully installed and run on the Windows Azure VM Role. See section 4.1.1.1.

**Result:** Requirement met.

5.1.2 SharpMap

**Demand:** Is SharpMap implementable on the Windows Azure VM Role?

**Outcome:** SharpMap was successfully installed and run on the Windows Azure VM Role. See section 4.1.2.1.

**Result:** Requirement met.

5.1.3 GeoServer

**Demand:** Is GeoServer implementable on the Windows Azure VM Role?

**Outcome:** GeoServer was successfully installed and run on the Windows Azure VM Role. See section 4.1.3.1.

**Result:** Requirement met.

5.1.4 MapNik

**Demand:** Is MapNik implementable on the Windows Azure VM Role?
CHAPTER 5. RESULTS

Outcome: MapNik was successfully installed and run on the Windows Azure VM Role. See section 4.1.4.1.

Result: Requirement met.

5.2 Map Servers on Windows Azure Worker Role

The results regarding whether or not the respective map server was successfully installed and run on the Windows Azure Worker Role are presented here.

5.2.1 MapServer

Demand: Is MapServer implementable on the Windows Azure Worker Role?

Outcome: MapServer was successfully implemented and run on the Windows Azure Worker Role. See section 4.1.1.2.

Result: Requirement met.

5.2.2 SharpMap

Demand: Is SharpMap implementable on the Windows Azure Worker Role?

Outcome: SharpMap was successfully implemented and run on the Windows Azure Worker Role. See section 4.1.2.2.

Result: Requirement met.

5.2.3 GeoServer

Demand: Is GeoServer implementable on the Windows Azure Worker Role?

Outcome: GeoServer was successfully implemented and run on the Windows Azure Worker Role. See section 4.1.3.2.

Result: Requirement met.

5.2.4 MapNik

Demand: Is MapNik implementable on the Windows Azure Worker Role?

Outcome: MapNik was successfully implemented and run on the Windows Azure Worker Role. See section 4.1.4.2.

Result: Requirement met.

5.3 Tile Caches on Windows Azure VM Role

The results regarding whether or not the respective tile cache was successfully installed and run on the Windows Azure VM Role are presented here.
CHAPTER 5. RESULTS

5.3.1 MapCache

Demand: Is MapCache implementable on the Windows Azure VM Role?

Outcome: MapCache was successfully installed and run on the Windows Azure VM Role. See section 4.2.1.1.

Result: Requirement met.

5.3.2 TileCache

Demand: Is TileCache implementable on the Windows Azure VM Role?

Outcome: TileCache was successfully installed and run on the Windows Azure VM Role. See section 4.2.2.1.

Result: Requirement met.

5.3.3 GeoWebCache

Demand: Is GeoWebCache implementable on the Windows Azure VM Role?

Outcome: GeoWebCache was successfully installed and run on the Windows Azure VM Role. See section 4.2.3.1.

Result: Requirement met.

5.4 Tile Caches on Windows Azure Worker Role

The results regarding whether or not the respective tile cache was successfully installed and run on the Windows Azure Worker Role are presented here.

5.4.1 MapCache

Demand: Is MapCache implementable on the Windows Azure Worker Role?

Outcome: MapCache was successfully implemented and run on the Windows Azure Worker Role. See section 4.2.1.2.

Result: Requirement met.

5.4.2 TileCache

Demand: Is TileCache implementable on the Windows Azure Worker Role?

Outcome: TileCache was successfully implemented and run on the Windows Azure Worker Role. See section 4.2.2.2.

Result: Requirement met.

5.4.3 GeoWebCache

Demand: Is GeoWebCache implementable on the Windows Azure Worker Role?

Outcome: GeoWebCache was successfully implemented and run on the Windows Azure Worker Role. See section 4.2.3.2.

Result: Requirement met.
5.5 Internal Role Scalability Options

The architecture together with the pros and cons of the four different scaling solutions are presented in the following sections.

5.5.1 Direct Mapping

Using the “direct mapping” approach, every web role has one dedicated worker role directly connected via internal TCP endpoints. An architectural overview can be viewed in figure 5.1. This solution can be further extended to support a special case of 1:N or M:1, where one web role is directly connected to several worker roles or the opposite. In this case there is no direct load balancing performed in between the roles, i.e. the load balancing must be performed by the single role (web or worker role).

![Figure 5.1: Architectural overview of the “Direct Mapping” solution.](image)

5.5.1.1 Characteristics

- 1:1 mapping [Web Roles : Worker Roles] (Can be extended to a special case of 1:N or M:1)
- Low latency
- Secure (i.e. internal endpoints not publicly exposed)
- Synchronous

5.5.2 Public Internal Endpoints

The “public internal endpoints” solution (seen in figure 5.2) targets the fact that Azure natively supports load balancing on external endpoints, using it to gain full M:N mapping (many to many). The only drawback found, is the reduced security due to exposing the worker roles via external endpoints instead of internal.

5.5.2.1 Characteristics

- M:N mapping [Web Roles : Worker Roles]
CHAPTER 5. RESULTS

Figure 5.2: Architectural overview of the “Public Interface Endpoints” solution.

- Low latency
- Non-secure (i.e. internal endpoints are publicly exposed)
- Non-synchronous
- More expensive (When data is transferred through the load balancer, this might be considered as inbound traffic and charged accordingly\(^1\))

5.5.3 Queue

The message queues described in section 2.6.3.2 can be used to elevate the scalability by enqueuing outgoing requests, from the web roles, in the queue as messages. Theses messages are then dequeued in an FIFO-manner by the non-occupied worker roles. An overview of this approach is shown in figure 5.3.

Microsoft is encouraging developers to apply this solution in order to increase scalability.

5.5.3.1 Characteristics

- M:N mapping [Web Roles : Worker Roles]
- High latency

\(^1\)This problem was discussed with Emil Gustafsson - a Windows Azure Team member - but no final answer was presented. They had not considered this solution before and therefore simply did not know whether or not this is charged as normal incoming traffic.
CHAPTER 5. RESULTS

Figure 5.3: Architectural overview of the “Queue” solution.

- Secure (i.e. internal endpoints not publicly exposed)
- Non-synchronous

5.5.4 Worker Role Request Broker

This approach uses a worker role instance to act as a request broker, redirecting incoming web role requests to non-occupied worker roles. This solution also enables full M:N mapping (many to many) but adds complexity (developer have to construct the broker) and latency. This solution is useful if the reduced security in “Public Internal Endpoints” and the redesign required for the “Queue” is unacceptable.

Figure 5.4: Architectural overview of the “Public Interface Endpoints” solution.

5.5.4.1 Characteristics

- M:N mapping [Web Roles : Worker Roles]
- High latency
- Secure (i.e. internal endpoints not publicly exposed)
- Non-synchronous
- Higher complexity
CHAPTER 5. RESULTS

5.6 Performance Tests

The results from the performances tests conducted according to the conditions set in section 3.4.6 (Test Conditions) are now provided.

5.6.1 Test 1: Request time differences between Azure VM & local VM

The test condition can be seen in section 3.4.6.1, the resulting plot in figure 5.5, and the arithmetic mean and median values for every test case in table 5.1. The column of plots to the left are the results from the local VM and those to the right are the Windows Azure, as can be seen in the plot headers i.e. [VM] or [Azure].

Figure 5.5: Resulting plots from Test1 - Request time differences between Azure VM & local VM.

Table 5.1: Arithmetic Mean and Median for Test 1

<table>
<thead>
<tr>
<th>VM Type</th>
<th>Arithmetic Mean(ms)</th>
<th>Median(ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Local VM</td>
<td>477</td>
<td>225</td>
</tr>
<tr>
<td>Azure VM</td>
<td>630</td>
<td>612</td>
</tr>
</tbody>
</table>
5.6.2 Test 2: Heavy-calculation requests on different VM-sizes

The test condition can be seen in section 3.4.6.2, the resulting plots in figures 5.6, and the arithmetic mean and median values for every test case in table 5.2.

![Histogram plots](image)

**Figure 5.6**: Resulting plots from Test2 - Heavy calculation requests on different VM-sizes.

<table>
<thead>
<tr>
<th>VM Size</th>
<th>Arithmetic Mean(ms)</th>
<th>Median(ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extra Small</td>
<td>9429</td>
<td>9968</td>
</tr>
<tr>
<td>Small</td>
<td>9405</td>
<td>9823</td>
</tr>
<tr>
<td>Medium</td>
<td>9380</td>
<td>9686</td>
</tr>
<tr>
<td>Large</td>
<td>9303</td>
<td>9643</td>
</tr>
<tr>
<td>Extra Large</td>
<td>9139</td>
<td>9597</td>
</tr>
</tbody>
</table>
5.6.3 Test 3: Multiple concurrent request on different VM-sizes

The test condition can be seen in section 3.4.6.3, the resulting plot in figure 5.7, and the arithmetic mean and median values for every test case in table 5.3.

Figure 5.7: Resulting plots from Test 3 - Multiple concurrent request on different VM-sizes.

Table 5.3: Arithmetic Mean and Median for Test 3

<table>
<thead>
<tr>
<th>VM Size</th>
<th>Arithmetic Mean(ms)</th>
<th>Median(ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extra Small</td>
<td>612</td>
<td>622</td>
</tr>
<tr>
<td>Small</td>
<td>691</td>
<td>621</td>
</tr>
<tr>
<td>Medium</td>
<td>691</td>
<td>617</td>
</tr>
<tr>
<td>Large</td>
<td>617</td>
<td>615</td>
</tr>
<tr>
<td>Extra Large</td>
<td>618</td>
<td>611</td>
</tr>
</tbody>
</table>
5.6.4 Test 4: Heavy-calculation requests on scaled VM-solutions

The test condition can be seen in section 3.4.6.4, the resulting plot in figure 5.8, and the arithmetic mean and median values for every test case in table 5.4.

Figure 5.8: Resulting plots from Test4 - Heavy calculation requests on scaled VM-solutions.

<table>
<thead>
<tr>
<th>VM Size</th>
<th>Arithmetic Mean(ms)</th>
<th>Median(ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 x Small</td>
<td>1650</td>
<td>1197</td>
</tr>
<tr>
<td>2 x Small</td>
<td>602</td>
<td>546</td>
</tr>
<tr>
<td>3 x Small</td>
<td>578</td>
<td>546</td>
</tr>
<tr>
<td>4 x Small</td>
<td>564</td>
<td>546</td>
</tr>
<tr>
<td>5 x Small</td>
<td>557</td>
<td>546</td>
</tr>
</tbody>
</table>
Chapter 6

Conclusions

The conclusions regarding GIS implementations on the Windows Azure platform now follow as well as discussions concerning the results of the performance tests. Furthermore some of the difficulties experienced throughout the project are highlighted and finally some Azure specific considerations and how to further extend this study are presented.

6.1 Implementation

All map servers and tile caches included in the test were successfully installed on both the VM Role as well as the Worker/Web role. Thus from an installation compatibility point-of-view there are no significant obstacles that cause Windows Azure to be unsuitable for GIS map servers or tile caches.

6.2 Internal Role Scalability Options

In the result chapter four, different options for scaling the internal role communication are presented. During the discussions with Azure Team member, Emil Gustafsson, regarding load balancing semantics, the four different solutions were presented to him. He took great interest in the “Public Internal Endpoints” option since this way of using the external load balancer was unknown to the team. Furthermore he attempted to investigate the potential economical impact of leading the data through the external load balancer twice, but no conclusive answer was presented. Thus, using this option may lead to unexpected costs.

6.3 Performance tests

As can be seen in the graphs and tables under section 5.6 (Performance Tests), the hardware limitations meant that the different test-sets ups were unable to place sufficient stress on the system so that the significant differences in between the role sizes could be easily seen.
6.3.1 Test 1: Request time differences between Azure VM & local VM

One of the reasons for testing this was that the VMs do not answer simple PING-messages. The differences in mean and median values in table 5.1 are the products of the greater spatial distance and more time consuming DNS-lookup for the Azure VM which was expected. The factor two difference between the mean and median values for the local VM is due to the hardware sometimes being pushed to its limits. The reason why this behavior is not visible on the Azure VM is the slightly better hardware (see section 2.6.8).

6.3.2 Test 2: Heavy-calculation requests on different VM-sizes

The resulting graphs and table indicate less difference in between VM sizes than was expected. This is most likely due to the message containing the VIA-points became so large that the transmission time greatly affects the result. The reason for using this amount of VIA-points was that the preliminary investigations (conducted on a local VM) showed that this magnitude was necessary in order to sufficiently stress the system to obtain viable readings.

The readings in table 5.2 gives a hint of the increasing computational power for the different VM sizes.

6.3.3 Test 3: Multiple concurrent request on different VM-sizes

Similar problems as witnessed in Test 2 were also reproduced here i.e. the hardware limited the amount of concurrent connections to the point such that only minor fluctuations can be seen in between the different VM sizes. The anomalies seen for the “Small” and “Medium” sized VM’s in the histogram are probably due to fluctuations in either the local area network or the Internet. However, because of the time constraints based on the project, and the extensive time required for another round of tests, it was not feasible to redo the test.

6.3.4 Test 4: Heavy-calculation requests on scaled VM-solutions

Table 5.1 initially offers great gains between using one or two VMs behind the load balancer but, from this point, the same problems, as described for test 2, can be seen due to the similar test conditions.

6.3.5 How to improve the performance tests

In order to produce more fair and impartial test results, a larger scale system should be used, enabling higher computational ability and bandwidth, or creating a more optimized test suite. Another solution would be to measure directly on the virtual machines, removing the effect of the large message size. The downside of using this approach is that the test is not conducted from a user point-of-view.

6.4 General Azure Observations

During the study some Azure specific observations have been noticed, for example, limitations in using PowerShell for startup-scripts and how to handle startup tasks.
6.4.1 PowerShell and updating 3rd party files

When installing MapNik in the worker role, the initial tests were conducted using PowerShell to download the python msi-installer directly from source, but, due to limitations with quotations marks in PowerShell that approach was discarded.

Initial command used in the startup.cmd:

```
```

The later solution is also preferred if scalability is highly prioritized, since it enables easy updates of Python, because updating the line above requires repackaging and uploading the project, while the latter is performed by following these steps:

- Replace the old python.msi in the BLOB-storage with the new version.
- Re-image the Role-instances.
- The CMD Shell-script will automatically download and install the newer version.

6.4.2 Startup Tasks

During the first test-installations of map servers a new process was created that handled the execution of the startup.cmd shell script. A better method was later found which involved using a built-in function in the ServiceDefinition-file - that leaves all the installation-phase-steps to Azure instead of retaining it in the actual application code.

Thus instead of calling starting the script via code, as follows (WorkerRole.cs):

```
Process p = new Process();
ProcessStartInfo psi = new ProcessStartInfo();
psi.FileName = "startup.cmd";
psi.CreateNoWindow = true;
p.StartInfo = psi;
p.Start();
p.WaitForExit();
```

this - more refined - solution was used (ServiceDefinition.csdef):

```
<Startup>
<Task commandLine="startup.cmd" executionContext="elevated"
  taskType="background" />
</Startup>
```

6.5 Difficulties

6.5.1 Implementation phase

As Mapnik only comes in a 32-bit pre-compiled package, a 64-bit had to be built from the source code which proved to be more challenging than expected. In order to build this
code easily experience with Visual Studio and general C++ development is required. The existing Windows 32-bit binaries [147], built with Visual C++ 2008 Express, requires vc90 runtime (included in Azure Roles) and was also successfully tested following the same steps as in section 4.1.4.2 (Mapnik - Worker Role).

6.5.2 Migration phase

Due to Azures requirement of x64 compatible code many of the external dependencies and libraries had to be recompiled. The majority of the libraries passed through the compiler without problems but when the built program was uploaded and installed on the Azure roles they caused havoc. By using the IntelliTrace support, the installation logs and stack traces could be gathered and used to discover the cause.

Another problem was discovered when attempting to read role-runtime-information (e.g. reading the appsettings) from inside the application. To enable this, the System.Core, System.XML.Linq and Microsoft.WindowsAzure.ServiceRuntime namespaces are required which, in turn, requires version 4.0 of the .NET platform. This led to few of the projects in the solution having to be upgraded from earlier version in order to fit the compatibility requirements.

6.5.3 Performance tests

Since the public IP-address of the load balancer can change at any time, the only way for external clients to connect, is to perform a DNS-lookup on the URL/DNS-name (YourApplicationName.cloudapp.net). However, in the following scenario this created connection problems. When re-deploying (not updating) a solution to Windows Azure, using the same upload-profile (i.e. using the URL/DNS-name), the old IP-address is still mapped to the URL, i.e. the TTL of the DNS-translation in the cache has not expired or been updated. This makes the application unreachable until such an update or TTL-expiration occurs. The administrator is able to temporarily circumvent this problem, using the VIP in the Management portal. This solution is, however, applicable only during a short time space since the VIP can, at any given time, be changed due to VM failover.

Furthermore, the initial tests were performed an on-premises hosted client, but, the limited bandwidth (100 Mbps) proved to be a bottleneck that interfered with the results.

6.6 Azure Specific Considerations

The most important consideration to bear in mind is that the storage on the local VMs is non-persistent which otherwise can cause problems as shown by the following examples:

- Directory permission and firewall rules must be granted in the startup script (or in the application code).
- When logging events for administrative use, the logs should be stored in the Azure BLOB storage instead of the local file-storage.
- External files needed for the application or dependent installations should be uploaded in Azure BLOB storage and downloaded automatically via the startup-script at the role initialization phase.
6.7 Future Work

This study could be extended by investigating the following areas;

• Investigate whether or not the “Public Internal Endpoints” solution is charged twice for incoming data.

• Test different “tile-caching” solutions;
  – VM’s local storage.
  – Windows Azure Cache
  – Windows Azure BLOB storage.
  – Windows Azure Content Delivery Network (CDN).

• Conduct performance tests on the different caching solutions to reveal which is the fastest.

• Test how 3rd-party software/databases can be integrated with the Azure-hosted application using Azure Service Bus and SQL Azure Data Sync.

• Produce a test database with Azure SQL Federations and conduct performance tests against the non-federated Azure SQL database to see how the reduced access time affects the overall system.

• Investigate GIS possibilities around Windows Azure HPC (High-Performance Computing).

• Investigate GIS possibilities around Windows Azure Hadoop connection.
Bibliography


BIBLIOGRAPHY


