Porting an Application to the Cloud
Using GridGain

Adam Hogan
BSc Computing
Session (e.g., 2010/2011)

The candidate confirms that the work submitted is their own and the appropriate credit has been given where reference has been made to the work of others.

I understand that failure to attribute material which is obtained from another source may be considered as plagiarism.

(Signature of student)__________________________
Contents

1 Summary ......................................................... 4
   1.1 Project Aims .............................................. 5
   1.2 Minimum Requirements .................................... 5
   1.3 Schedule .................................................... 5

2 Aim ............................................................. 7

3 Minimum Requirements ......................................... 7

4 Schedule ....................................................... 9

5 Background Research ............................................ 10
   5.1 Background to the Problem ............................... 10
   5.1.1 Grid Computing and Cloud Computing - Wholly Different? 10
   5.1.2 Grid Architecture ....................................... 10
   5.1.3 Applications of a Grid Environment .................... 11
   5.2 Cluster Computing .......................................... 12
   5.2.1 Applications of a Cluster Computing Environment ...... 12
   5.3 Utility Computing ........................................... 13
   5.3.1 Applications of a Utility Computing Environment ... 13
   5.4 Cloud Computing ............................................ 13
   5.4.1 Virtualisation ............................................ 14
   5.4.2 OpenNebula .............................................. 14
   5.4.3 Eucalyptus .............................................. 15
   5.4.4 Cloud Computing and their levels of Service ............ 16
   5.4.5 Infrastructure as a Service (IaaS) ....................... 16
   5.4.6 Platform as a Service (PaaS) .......................... 16
   5.4.7 Software as a Service (SaaS) .......................... 16
   5.4.8 Benefits of Cloud Computing ........................... 16
   5.5 GridGain ..................................................... 17
   5.5.1 GridGain Success Stories .............................. 20
   5.5.2 Jake2 ....................................................... 20
   5.6 Alternatives to GridGain ................................... 22
   5.6.1 BOOM .................................................... 22
   5.6.2 GridGain vs BOOM ....................................... 23
   5.6.3 Emotive Cloud ............................................ 24
   5.6.4 GridGain vs Emotive Cloud ............................. 25
1 Summary

My project is to deploy a medical application to the cloud using a cloud application framework called GridGain. The application itself deals with the 3D reconstruction of human tissue from the architectural to the cellular level through the use of 2D virtual slides. These slides are then processed in order to produce a 3D reconstruction of some human tissue. This allows the histopathologist to view the tissue from many different angles and orientations in order to observe an accurate representation of the tissue. The histopathology application typically deals with very large, highly detailed images of tissue samples. This high-level of detail is required in order to accurately diagnose any diseases, in real-time, which may be present in the given tissue sample.

Currently, the implementation is that of a classic “client-server” model, in which large (typically gigabyte) images are retrieved from a server by a single client. This has several performance issues pertaining to efficiency, adequate network throughput for image retrieval and application reliability. It is therefore the scope of this project that alternative solutions are researched in order to address these issues; namely in terms of the cloud computing paradigm and related technologies.

The current implementation is essentially sub-standard. Reliability remains a key issue as rendering and reconstruction of the tissue can take many hours to complete which is highly undesirable considering diagnosis should be highly accurate and done within a timely manner. Another factor affecting the reliability of this application is the available network bandwidth between client and server during image transfer and indeed the memory and computational resource limitations inherent in a single machine, single core implementation.

Another key issue is that of efficiency and scalability. Due to the nature of the current implementation, computational and memory resources are exhausted rapidly especially under heavy network load. As numerous histopathologists may wish to view numerous different slides at once, the application performance quickly declines resulting in high latency and increased server/application response time.

In light of these issues, I plan to rectify them by re-deploying this application onto a cloud infrastructure. Due to the nature of the cloud, where there are ample computational and memory resources that can be allocated intelligently through the use of Virtual Machines, current failings in terms of scalability, reliability and efficiency should be eradicated. Furthermore, the nature of GridGain, which allows for intelligent resource allocation through the use of GridGain “nodes”, should further enhance the distributed solution to this problem.
1.1 Project Aims

The aim of this project is to port some existing medical imaging software to the cloud to overcome a number of limitations such as computational and memory limitations of a single machine, which is the current implementation. Therefore, enabling rapid, accurate 3D reconstruction of tissue from the architectural to the cellular level.

1.2 Minimum Requirements

My project minimum requirements were as follows:

- Design a redeployment strategy for redeploying an existing application to the cloud using GridGain.
- Construct a prototype, redeployment application of the current implementation using GridGain.
- Deploy a protoype, redeployment application of an existing application on to the cloud using GridGain.
- Demonstrate a prototype, redeployment application of an existing application within a cloud environment under GridGain.

1.3 Schedule

This section will clarify my schedule as shown in the Schedule section of this report. The schedule was as follows:

- Project Begins: 24/01/2011 - 29/01/2011
- Literature Review: 24/01/2011 - 16/02/2011
- Alternative solutions: 17/02/2011 - 05/03/2011
- Design of a Solution: 07/03/2011 - 19/03/2011
- Implementation of a Solution: 22/03/2011 - 31/03/2011
- Evaluation of a Solution: 04/04/2011 - 10/05/2011

The Project Begins section of my schedule involved the creation and submission of my aims and minimum requirements. The Literature Review section relates to the Background Research section of this report and involved extensive research into significant areas within Cloud Computing including Infrastructure as a Service, Software as a Service and Platform as a Service. This section also involved extensive research into GridGain and how it works. All of the sections within my schedule have sub-tasks to enable better time-management of my project at
each phase. The Alternative Solutions section details middlewares that were researched in relation to GridGain in order to help me make an informed decision as to why GridGain was the most suitable given the abundance of emerging and long-standing middlewares available. The next section of my schedule involved designing an appropriate solution to be deployed using GridGain. In order to tackle this section, I made use of video tutorials supplied on the GridGain website and example code in order to give me the inspiration and ideas I required to produce a suitable solution. Furthermore, I researched areas where GridGain had been successfully used to deploy an application which further helped me to design a solution. Up until this section of my schedule all my goal had been achieved in the allocated timeframe. The Implementation section of my solution was unfortunately extended by a number of weeks due an unresolved issue with being unable to get GridGain and the current Histopathology application to work at the same time from within the same virtual machine. This was a major setback and is detailed more thoroughly in the Implementation section of this report. Furthermore, as a result of this the evaluation section had to be completed in less time and ultimately meant that no testing could be carried out due to my implementation only being partially completed. This has meant that I have detailed what testing would have been carried out had my implementation been successful although a functioning solution was produced albeit without a visual output. This is mentioned in further detail within the Implementation and Evaluation sections of my report.
2 Aim

The aims of this project are firstly to gain a good foundation and understanding of Cloud Computing and it related paradigms. For example Grid Computing, Utility Computing and Cluster computing. Secondly, after obtaining this foundation, this knowledge will then used to consider how to redeploy an existing histopathology application onto a Cloud environment, which will be aided by research into how GridGain has been used successfully in porting applications to the cloud environment. The current application is a single machine, single core implementation which histopathologists use in order to diagnose human tissue samples. The idea of redeploying the current implementation onto a Cloud environment is to combat current issues with reliability, scalability and efficiency. Thirdly, another aim of this project will be to research alternative solutions to this problem in order to justify the choice of GridGain for application redeployment. A final aim is to produce an evaluation of the redeployment in order to effectively compare the current single-server, single-client implementation with the newer, more distributed and parallelised application.

The objectives of the project are to:

- Illustrate how to use GridGain effectively in order to re-deploy an existing application onto a cloud environment.
- Thoroughly research Cloud Computing and its related paradigms.
- Research how GridGain has been used to successfully redeploy an existing application onto a cloud environment.
- Research alternative solutions to this problem in order to justify why GridGain was chosen.

3 Minimum Requirements

The minimum requirements are:

- Design a redeployment strategy for redeploying an existing application to the cloud using GridGain.
- Construct a prototype, redeployment application of the current implementation using GridGain.
- Deploy a prototype, redeployment application of an existing application on to the cloud using GridGain.
- Demonstrate a prototype, redeployment application of an existing application within a cloud environment under GridGain.
The possible extensions are:

- To design a front-end webservice interface for the redeployed application to enable access over the Internet.

- To design a framework for porting medical applications to the Cloud using GridGain, using this project as proof of concept.
4 Schedule

Figure 1: Schedule
5 Background Research

5.1 Background to the Problem

For this section of my report, I am going to talk about various computing paradigms that are related to Cloud Computing. These include Grid Computing, Cluster Computing and Utility Computing. Secondly, I will also be talking about areas of research relating to Virtualisation and how it plays a key role in defining Cloud Computing. I will also be talking about the different levels of Cloud architecture, such as Saas (Software as a Service), PaaS (Platform as a Service) and IaaS (Infrastructure as a Service). Finally, I will also talk about GridGain, its competitors, its benefits and it where it has been successfully implemented.

5.1.1 Grid Computing and Cloud Computing - Wholly Different?

Grids are seen in some respects as predecessors to the Cloud as much of the Cloud technology we see today started in the Grid. Although the two are very similar, there are significant differences which separate the two paradigms. For example, it has been remarked by Ian Foster et al.[11] that the vision of Cloud computing is essentially the same as with Grids in that the two have very similar visions with respect to scalability, reliability and cost. Indeed, the advent of cheaper commodity hardware and the exponential increase in information over the past decade may well have contributed to this vision and the hype of allowing a 3rd party vendor to manage these grids and our information. Furthermore, the challenges faced by both Grid and Cloud Computing appear to be almost uncanny [11]. For example, both paradigms provide computational, storage and memory resources that are required by the end-user. Both also operate on a large scale and face management issues with regard to cost, Service-Level Agreements (SLAs) and resource provisioning to name but a few [11]. A key, unique point about Clouds when compared to Grids is that they make use of virtualisation as a layer on top of hardware in order to scale on demand and provide resources to the end-user as a service [11]. Unfortunately, Grids do not have this capability, as the original idea behind grids was to solve large-scale computation without the expense of using a supercomputer[11].

5.1.2 Grid Architecture

Grids are a type of loosely-coupled, distributed system which delegate resources and which are based upon open-standard protocols in order to provide an accepted quality of service and thereby providing a strong degree of interoperability between nodes within a grid. This idea can be further extended to include the sharing of resources across multiple organisations, which may or may not be geographically disparate.[19] Grids also make extensive use of Virtual Organisations (VOs), which allow recognition of resources as if they came from within the same autonomy/administrative domain[11]. However, each domain itself is subject to its own management and security policies [8]. This is different in nature to Clouds, which offer services
"on-demand" to consumers as and when they are required and do not particularly work within the realm of domains but can have resources that are sourced from multiple geographical locations. Additionally, both Grids and Clouds inherently support heterogeneity, stemming from the fact that differing hardware such as off-the-shelf, cheap, commodity hardware can be used within both Grids and Clouds without any impact upon how either paradigm operates[12].

Both Grids and Clouds currently suffer from a lack of formally recognised standards for how each should operate. However, there does seem to be some progress being made for the Grid paradigm in relation to encapsulating how the Grid operates; more specifically in relation to the "hourglass" model devised by Foster,Kesselman and Tuecke[12]. This model provides a small number of protocols at the "neck" of the hourglass which can be used to devise services, which in turn can be derived by applying them to the technologies that reside at the bottom and top faces of the hourglass. The number of protocols is purposely kept small to allow greater interoperability and sharing of resources between virtual organisations. In this way, such a model allows for a modular approach for creating new Grid services as existing technology from the top and bottom layers can be combined without inventing new protocols/standards.

Security in Grids and Clouds is an area in which there appears to be a significant difference. Within a Grid environment, a user would make use of a security credential which would validate his/her use of the Grid from within a particular virtual organisation. In this way, there appears to be a distinct lack of specific end-user security within Grids when compared to Clouds [19]. However, this is no doubt counterbalanced with the application of a single-user sign on, which would presumably be administered by the relevant authorised user.[12]. Within a Cloud environment however, due to the use of virtualisation this inherently applies another layer of security, which would shield a particular user from a particular organisation from a different user and organisation; thereby offering a certain amount of isolation.

5.1.3 Applications of a Grid Environment

As mentioned during the introduction of this project, Grids were initially driven by a need to solve large-scale computing problems without the added expense of a supercomputer and which to run on a single machine was just not economically viable in terms of time to complete, which would inevitably extend to weeks or even years for some applications and large data sets. Such applications and data sets would find pride of place within a Grid environment, with some being "embarrassingly parallel" and noticeably more efficient within a Grid.

An example of a recent Grid application would be the Large Hadron Collider developed at CERN in Switzerland. This highly ambitious project to try and discover a particle known as the "Higgs Boson", produces an absolutely staggering 15 million gigabytes of data per year. The Grid itself is made up of over 140 computing sites arranged into tiers. Tier 1 sites are responsible for processing the data collected by the LHC whilst Tier 2 sites are responsible for
storage and other computing resources in relation to this project. The data itself is requested to be analysed by any number of scientists working on this project. The Grid delegates the tasks into jobs to other Tier 2 sites within the grid, which returns a result to the within days[7]. The results are themselves sent along dedicated fibre optic links to the relevant centre for analysis.

Another example of a Grid application is the use of a Grid within computer animation. The paper by Chong et al. proposes a framework for the rendering and subsequent retrieval of rendered images from a Grid environment.[9]. As part of this framework, Chong outlines the theoretical transfer times of data across a network and discusses the problems that arise with regard to contention and how to overcome this with use of a specific compression algorithm to reduce the file sizes of large 3D animation image files[9].

5.2 Cluster Computing

Cluster computing is a paradigm that is somewhat closely related to Grid computing. They are in actual fact a type of distributed system that supports the execution of jobs in parallel but in a homogeneous environment, unlike the Grid paradigm.[8] Furthermore, clusters are predominantly under the same autonomy, are more tightly coupled and have dedicated networks unlike Grids which are usually accessed over the Internet and are subject to high-latency.[8] In addition, clusters are less secure, relying upon the traditional logins with username and passwords and user-privileges. This is converse to Grids, which use highly encrypted security credentials such as public/private key authentication from trusted certificate authorities.[8]. Clusters tend to operate on a much more local basis, with resources being less geographically disparate and of much higher bandwidth when compared to Grids.[8].

5.2.1 Applications of a Cluster Computing Environment

A real-world example of cluster computing would be the Google search engine; more specifically the search function it employs, which returns results of webpages from keywords entered by the user. Google routes queries to the cluster, which is nearest to the location of the query. Google's indexing function is also consulted, which is constructed by Google's web crawlers, which then returns a list of matching documents. The actual distribution of the search and results and the results for the index are both functions which can be executed in parallel to increase the efficiency and optimise the speed of the search [17].

Another example of cluster computing is the Folding@Home project. The Folding@Home project from the University of Stanford makes use of distributed computing techniques in order to solve problems related to diseases such as Huntingtons disease, cystic fibrosis and Mad Cow disease. All these diseases have proteins, which misfold and subsequently cause these diseases. It is this process of misfolding into 3D shapes from a single chain of amino acids that is investigated and requires massive computational resource in order to obtain results. Folding@Home
uses redundant CPU cycles of those users that have chosen to become participants of this project. How the molecules of the proteins interact is what is parallelised between participants. This vastly optimises the speed at which these parallel computations are done [17].

5.3 Utility Computing

Utility computing is a paradigm that aims to provide computing as a resource, much in the same way that Water, Gas and Electricity are provided. Indeed, Buyya et.al. remark and layout a framework for establishing a market-oriented architecture for trading and selling computing as a kind of commodity, which is regulated by supply and demand [8]. Utility computing itself can be closely compared with Cloud computing in that it makes use of virtualisation to abstract physical resources such as memory, computational resource and storage and provide them as a rented and metered facility to consumers for a fee.

5.3.1 Applications of a Utility Computing Environment

A shining example of Utility Computing would be Amazon’s EC2 (Elastic Compute) cloud. The Amazon EC2 is a service, which may be rented by the user for a nominal fee. Such entities as CPU time/resource and memory can be leased for fixed number of cents or American dollars per hour. It successfully does this by making extensive use of virtualisation using the Xen hypervisor in order abstract and provide virtual machines that provide fixed capacity resources to customers as a service [20]. Furthermore, the EC2 infrastructure is organised into regions and availability zones, which gives the user the flexibility to choose which physical location to execute a particular task or computation with respect to minimising cost, execution time and with regard to fault-tolerance and reliability of service [20].

5.4 Cloud Computing

As Grids have been defined to tackle large-scale problems, it may well be appropriate to consider Cloud computing as a possible extension to Grid Computing, with a layer of Utility computing operating over the top. In this way, Cloud Computing has been remarked as tackling even larger scale computational challenges; perhaps even Internet-scale as remarked by Foster et. al. [11]. Furthermore, it would be reasonable to consider the Clouds are the natural successor to Grids as they face similar challenges in terms of computational, memory and storage capabilities. Clouds are essentially driven by economies of scale according to supply and demand from users; which is akin to the way Amazon EC2 operates by charging consumers by the hour in terms of instance run-time regardless of data transferred and computational resource used [20].

Clouds themselves, like Grids, offer multiple types of resources. However, within a Cloud, this usually done through virtual machines as with virtual machines, specific amounts of memory, storage and computational resource can be virtually abstracted and allocated to a specific virtual
machines. Within a Cloud environment, these may be accessed via Web Service technologies such as SOAP and REST.

5.4.1 Virtualisation

Virtualisation is the process of abstracting physically available resources such as storage, memory and CPU time and allocating them in such a way that allows them to be used virtually by software. This is usually done through one or more Virtual Machines (VMs). In this way virtualisation allows for the partitioning and aggregation of resources in order to harness all resources currently available for a given task which maybe data or or computationally intensive [17]. However, as well as providing emulation at the hardware layer, virtual machines can be used to emulate operating systems such Windows and Linux through the use of virtual images. These virtual machines can be managed through the use of infrastructure managers such as OpenNebula and Nimbus. Using virtualisation allows an organisation to offer specific virtual machines as services to users on top of a fully-managed cloud infrastructure.

5.4.2 OpenNebula

![OpenNebula architecture](image)

OpenNebula is effectively a Cloud management tool that efficiently manages numerous virtual machines to give the appearance of one entire, elastic Cloud environment [14]. As depicted in figure 4, OpenNebula operates on a separate layer above the network infrastructure allowing for greater interoperability between different Cloud environments and ensuring no vendor lock-in occurs. Due to OpenNebula operating at a separate layer, it allows full use of currently available resources such as storage, security and computational resources to be abstracted and delegated.
to users in the form of various virtual machines. Also depicted in the figure shown are image repositories that make use of virtual storage, which will be a portion of the physical hard drive storage that has been abstracted for use by a particular virtual machine. Such requirements as storage, memory and computational resources can be scaled up and down on demand as required.

5.4.3 Eucalyptus

Eucalyptus is predominantly an academic research and exploratory concept that addresses the differences in how Cloud Computing is implemented by various providers through various levels of abstraction such as IaaS, PaaS and SaaS; focusing specifically on Infrastructure as a Service [15]. Eucalyptus has the main objectives of "extensibility" and "non-intrusiveness" so as to provide a truly open and flexible framework upon which to experiment with the concepts of Cloud Computing [15]. Figure 3 depicts the Eucalyptus architecture. The Eucalyptus architecture is made of various node controllers which are responsible for managing the use of virtual machines, which in this case use the Xen hypervisor. Furthermore, the node controllers make use of WSDL documents to define the virtual machine data and control the operations which the node controllers supports [15]. The CC (Cluster Controller) nodes are responsible for managing logical groups of node controllers through delegating virtual machine execution requests to node controllers and managing public and private networks within the Eucalyptus environment. [15]. The nodes in the figure labelled CLC (Cloud Controller) is the user’s entry into the Cloud environment. The Cloud controller is responsible for processing many high-level functions such as Service Level Agreements, virtual machine creation and user authentication [15]. The Eucalyptus framework itself is very modular to accommodate for extensibility and purposefully makes use of existing technologies such as WSDL to enhance interoperability with other cloud infrastructures.
5.4.4 Cloud Computing and their levels of Service

Another distinction between Clouds and Grids is the Clouds inherent partitioning of services. This partitioning of services relates to 3 different levels known as Infrastructure as a Service (IaaS), Software as a Service (Saas) and Platform as a Service (Paas).

5.4.5 Infrastructure as a Service (IaaS)

Infrastructure as a Service (IaaS) operates as the name suggests by providing the underlying infrastructure as a service to the consumer. This can include routers, switches, servers (most likely virtualised servers) and any other additional resources. An example of this would be Joyent [6], which provides an extensive number of virtualised servers that are exposed to the consumer. Another example would be the Amazon EC2 cloud, which provides scalable levels of storage and computation that operates on a per-per usage model [11].

5.4.6 Platform as a Service (PaaS)

Platform as a Service provides access to a development environment without the user having to consider the underlying infrastructure and hardware. PaaS usually facilitates the support of various programming languages such as Python or Perl [11]. Usually, for a consumer to make use of PaaS, they would usually access it through an API or front-end in order to configure and deploy their required service [6]. In some instances, the consumer may well be constrained to the type of software they can produce due to the providers infrastructure, in exchange for scalability [11].

5.4.7 Software as a Service (SaaS)

Software as a Service provides applications to the consumer as a Web Service. These applications can be accessed through a Web Service interface over the Internet.

5.4.8 Benefits of Cloud Computing

As stated earlier, Cloud Computing inherently comes from a need to process increasing amounts of data. As organisations become increasingly more complex in terms of computing infrastructure, the number of services they offer and the number of applications they require. Organisations are increasingly looking for a viable way of reducing and even migrating the cost of managing their own infrastructure to that of a 3rd part provider[6].

Indeed, the benefits of migrating to a cloud solution are not just limited to operational costs of managing a cloud infrastructure. Due to the vast amounts of resources available within a Cloud environment when compared to a single-machine implementation, the potential to reduce running and response time is gargantuan [6]. For example, the New York Times newspaper
decided to use a cloud in order to batch process data across 1000 servers; effectively reducing
the run time to 1/1000th of the time a single server implementation would require [6]. This
example epitomises the amount of resources a Cloud environment could offer to an organisation.

Another example of optimal cloud usage is an application known as Animoto [6]. This applica-
tion processes numerous images and music and proceeds to make a video from these two types
of data. Impressively, within 72 hours this application scaled from double-digit server usage to
a few thousand without impacting performance. This is a good example of how an application
can scale "on-demand" and still reduce response time to its users cite [6]. This was done
through the use of virtual machines whose job it was to process any computationally-intensive
data, thereby effectively helping to improve the application response time.

Another important aspect of Cloud Computing to consider is risk; or more importantly in-
frastructure risk. It is important for an organisation to assess whether the applications they
currently use will make full use of a cloud solution and not lie dormant, which would increase
overall cost. Furthermore, it is important for an organisation to assess its future requirements
also. For example, will an organisation have applications that require a scalable solution in the
future in order to meet consumer demand? If not, then what other viable requirements does
an organisation have that warrants a migration to the cloud?

Additionally, The three levels of the cloud as mentioned above (IaaS, PaaS and SaaS), together
provide a full infrastructure on which to design, implement and use services on the Cloud. This
modular approach also allows cloud providers to expose as many of the 3 layers mentioned
as they wish according to their individual consumer requirements. For example, the company
Salesforce.com [11] provides Customer Relationship Management (CRM) services by making us
of its own PaaS known as Force.com.

Essentially moving to a cloud solution removes most if not all infrastructure risk [6]. In the
vast majority of Cloud provider cases, they have surplus infrastructure to anticipate and meet
growing demand from consumers and organisations. This greatly reduces the financial implica-
tions of an organisation as there is no requirement to invest in any new hardware or software
[6].

5.5 GridGain

GridGain is a Java and Scala based Cloud middleware that combines the potential of data and
computational grids with support for auto-scaling based upon demand. GridGain can be used
either within a Windows or Linux environment due to its native Java support. Amongst its
numerous features, GridGain allows for the even distribution of tasks across all nodes and for
nodes to be added automatically to any existing GridGain topology without the need for any
extra classes or packages to be loaded; known effectively as "zero deployment"[13]
Figure 4 outlines the architecture of GridGain. Figure 3 illustrates how GridGain works, with the smaller clouds being the "jobs", which are a smaller, sub-unit of work of the overall task which are executed in parallel on the "Worker" nodes and then each job is amalgamated and passed back to the "Control" node at the beginning. As figure 2 depicts, GridGain is built upon existing technologies allowing for greater interoperability between heterogeneous environments, in line with the vision of how a Cloud environment functions such as providing virtualised resources to users on demand and load balancing those resources so as to provide greater levels of efficiency.[19] Furthermore, GridGain makes use of a technology known as MapReduce, which has the responsibility of delegating Grid jobs to specific Grid nodes in the GridGain topology and assumes that all nodes at that moment have equal work capacity available in terms of computational time and memory available.[4] Additionally, the discovery of other GridGain nodes is done through multicasting and each multicast is known as a "heartbeat" in GridGain.
terminology. The number of heartbeats which can be sent out can be altered to provide a more accurate and current map of the present GridGain topology. This can help the user to accurately obtained which nodes are available however, care must be taken to prevent unnecessary network traffic generated from too many multicast packets; thereby making inefficient use of the resources available. In addition, GridGain makes use of distributed caching allowing for data that is used regularly to be placed closer to where it is most likely to be used; effectively reducing the amount of network bandwidth and Grid node resources used during task execution.

GridGain also makes use of different programming paradigms such as Aspect-Oriented Programming and functional programming, provided through the functional programming language Scala. Functional programming languages allow state to be attributed to data and are thus seen as being ideal for data within Grid environments. Furthermore, Scala is fully interoperable with Java as it runs within Java Virtual Machine (JVM). This gives the developer the freedom of choice to use either object-oriented, functional or a mixture of both programming paradigms. Aspect-Oriented Programming is a paradigm that extends the Object-Oriented paradigm in that it further separates functions from each other without crossover between the functions. This helps to provide greater modularity in the form of code reuse and also aids in future code improvement. Scala itself provides a higher-level way of working and interacting with GridGain in the sense that a lot of the “boilerplate” code specifying how Grid nodes should function and how tasks should be split and delegated in the form of jobs is hidden from the developer. This greatly simplifies code and rapidly improves development time.

GridGain also makes extensive use of load-balancing in order to fully utilise available resources and provide greater efficiency and fault-tolerance whilst tasks are being split into jobs and
subsequently executed on the Grid. One such flavour of load-balancing, which GridGain uses is known as Round-Robin. Round-Robin load balancing is used in GridGain by default and works by sending out the first Grid job from a particular task to the first node in the topology, the second Grid node receives the second job and so on until GridGain gets back to the first Grid node, hence task jobs are distributed in a circular, round-robin fashion.[13] Another flavour of load-balancing used by GridGain, known as adaptive load-balancing, makes use of various Grid node metrics such as CPU load, processing time and job waiting time, allowing for greater accuracy in determining which node should execute which job to allow more efficient use of available resources on the Grid.

In addition to load balancing, GridGain further supports fault-tolerance on a cloud through two mechanisms; fail-slow and fail-fast. Fail-fast stops a task and returns an error if any of it’s subsequent jobs fail to execute on a Grid node. Conversely, fail-slow will try and delegate a failed job from a particular task until either the job executes or until all available nodes have been exhausted. This provides a robust platform on which to perform large-scale computations in a manner which fails gracefully should an error occur.[13]

GridGain has also recently introduced APIs for use within hybrid cloud environments; more specifically to support Amazon EC2, RackSpace and in-memory cloud.[13] This enables the user to manage cloud instances such as those provided through virtual machines from within application and removes the requirement of virtual infrastructure managers such as OpenNebula and Nimbus.[13]

5.5.1 GridGain Success Stories

5.5.2 Jake2

GridGain has been successfully used to port a Java-based multiplayer game known as Jake2 to a Grid environment. This a Java based implementation of the popular online multiplayer game Quake2. The aim of this project was to re-engineer the current implementation so that it would work within a Grid environment. In order to be able to do this, the developers of this project first had to discover the modules that make up the structure of this game and how each interact in order to decide which functions would benefit from being "gridified". As figure 6 shows the logic of how Jake2 works, which aids in deciding which methods and events may be gridified so as to gain efficiency in performance and speed. Figure 5 illustrates the GridGain architecture of Jake2. As the figure illustrates, Jake2 delegates tasks, which are split into jobs to be executed in parallel. This is done via a Control node, which decides which how tasks are split into jobs and which Worker nodes, which do the actual processing of the data, will get which specific job. This is done using a MapReduce function, which assumes that worker nodes will have differing levels of computational, memory and storage capacity depending on their current load and will efficiently delegate either more or less jobs to nodes depending upon
Figure 6: Jake2 architecture

[4]

these metrics.
5.6 Alternatives to GridGain

This section will detail the alternatives to GridGain, their attributes and how they compare to GridGain.

5.6.1 BOOM

BOOM (Berkley Order of Magnitude), is very much an approach to how a Cloud software stack should be built to facilitate the construction of distributed software. It has the motto of producing "orders of magnitude bigger systems with orders of magnitude less effort" [2]. It takes a declarative approach to designing a distributed system and treats everything within a Cloud environment as collections of data; thereby making it much simpler and easier to model system state over time and is therefore more accurately associated with the temporal differences of system state (i.e. how the data itself changes overtime).[2] Due to the nature of this design, system state is easily replicated and partitioned across nodes within a Cloud environment due to data representation as sets and relations [2]. This is very much a high-level approach aimed at reducing code complexity by abstracting away details such as data layout and removes the requirement of coding parallel data mechanisms such as threads, concurrency and serialisation, which need to be accounted for in languages such as Java.

To try and hypothesize upon and demonstrate the use of BOOM, the BOOM project implemented a complete rewrite of the Hadoop filesystem known as HDFS and the MapReduce
This involved the use of Overlog, a declarative language intended to model system state similar to how relations are represented in SQL. This is shown below [2]:

```
path (@From, To, To, Cost )
:- link (@From, To, Cost);
path (@From, End, To, Cost1 + Cost2)
:- link (@From, To, Cost1),
    path (@To, End, NextHop, Cost2);
WITH RECURSIVE path (Start, End, NextHop, Cost) AS
(
SELECT From, To, To, Cost FROM link
UNION
SELECT link .From, path.End, link.To,
    link.Cost + path.Cost
FROM link, path
WHERE link.To = path.Start);
```

As you can see, the code at the top is the Overlog code for computing all possible paths from links. The code below is the SQL translation to ease understanding of the Overlog code. The Overlog code is very declarative and functional and effectively hides all logic regarding how the path computation is done, which would significantly reduce development time. The BOOM project has since evolved from using Overlog and has implemented its own language known as BLOOM, which is based on a temporal, functional logic language known as Dedalus [3]. It was argued by Alvaro et al. that Overlog lacked a clear definition of how time could be modelled. This is essential in distributed programming where key concepts such as concurrency, asynchronisation and parallelisation are all subject to time such as the length of time a task/job has executed, which can be further used to decide if a task/job should be stopped or passed to another node[3]. Dedalus functions much in the same way as any other logic language such as Prolog and allows for inference upon time from rules and predicates, which relate to specific domains.

### 5.6.2 GridGain vs BOOM

BOOM is very much associated with an overall Cloud software stack design to facilitate the construction of “true” distributed software. Here, the word “true” is used to denote software that has been entirely constructed within a distributed environment such as BOOM and not merely ported and redeployed on a Cloud from a single machine implementation, which maybe somewhat true of GridGain. However, GridGain does take a functional approach to distributed software design as in terms of redeployment and distributed software design from the ground up through its combined use of Java and Scala. With GridGain, there is greater scope for
modularity in terms of code reuse and GridGain offers different approaches to distributed software redeployment/design through the use of Aspect-Oriented Programming (AOP) and functional programming using Java and Scala [13].

Furthermore, BOOM appears to be very theoretical in nature and deals predominantly with predicates, relations and rules of inference [3]. Although this is a sound approach to dealing with data within a system, it does appear to lack practicality and flexibility in relation to constructing business logic and is very much a framework for constructing distributed software. Conversely to this, GridGain is a complete data and compute Grid that facilitates the redeployment and construction of single implementation and distributed software and appears to be more of a complete and practical package in terms of porting an application to the Cloud.

5.6.3 Emotive Cloud

Emotive (Elastic Management of Tasks in Virtualised Environment) Cloud is a middleware who’s main use is as a Virtual Machine manager incorporating scheduling and customised virtual machines through the use of different hypervisors such as Xen and KVM [1], and makes use of a Java API known as Libvirt in order to support the many virtualisation hypervisors that exist. It is the aim of Emotive Cloud to provide fully customised Virtual Machines that satisfy the resource requirements of the user [1]. Figure 8 illustrates the Emotive Cloud architecture.

![Figure 8: Emotive Cloud architecture](image)

The architecture is split into three layers; Scheduling layer, Node Management layer and Data Infrastructure layer [1]. The scheduling layer is responsible for the management of the different nodes and virtual machines within Emotive Cloud such as choosing specific nodes to execute tasks based upon their current load and the delegation of virtual machines to physical nodes.
This layer also makes use of different scheduling policies such as Round-Robin and SERA (Semantically-Enhanced Resource Allocator), which supports meaningful descriptions for allocating resources to VMs as well as the use of AI agents to aid this function [1]. The Node Management layer (characterised by yellow and blue border containing the acronyms VRMM (Virtual Resource Management and Monitoring and RM (Resource Management) respectively), is responsible for the efficient creation and management of VMs so as to reduce any overhead associated with this task. This is done by the Virtualisation Manager (VtM) as shown in figure 8. The Resource Manager is responsible for the intelligent allocation and efficient usage of available resources to each virtual machine to ensure adherence to specified SLAs [1].

In addition to the aforementioned attributes, Emotive Cloud has very robust mechanisms for dealing with failed tasks such as a checkpoint system to allow other VMs with available resources to take over any tasks from a failed Virtual Machine [1]. Furthermore, Emotive Cloud provides a certain degree of persistence by allowing data to be saved to a VM including between different user sessions. This allows for migration of VMs between physical hosts so as to make full and efficient use of all resources available. In addition, each VMs support for task execution allows it to execute task as soon as it is initialised by the VRMM layer. This substantially reduces idle time and allows greater efficiency throughout the whole VM lifecycle [1].

The Data Infrastructure layer makes use of a Network File System (NFS) to provide persistent access to shared amongst related VMs [1]. This done through a shared storage and repository mechanism enabling VMS to be moved seamlessly between physical hosts without loss of data. Furthermore, this approach enables the VRMM mechanism to optimise the overall cloud structure and thus make greater, more efficient use of available memory and CPU resources through VM consolidation between physical hosts [1].

5.6.4 GridGain vs Emotive Cloud

Emotive Cloud is a very comprehensive middleware for the creation and management of Virtual Machines for operation within a cloud environment. Indeed, Emotive Cloud shares some common attributes with GridGain such as Round-Robin scheduling [1, 13], task delegation and even similar failover methods for failed tasks. Furthermore, both GridGain and Emotive Cloud make use of Java to provide functionality and both support web technologies such as REST and both can even interface with Amazon EC2; thereby giving way to different types of hybrid Cloud infrastructures [1, 13].

However, the key, significant difference between the GridGain and Emotive Cloud is portability and development. Although Emotive Cloud is perhaps more aesthetically pleasing with regard to having a GUI for setting up VMS and tasks, GridGain allows for existing applications not only to be deployed to a Cloud environment, but to also be ”gridified” in such a way which will improve performance and increase application efficiency; thereby each application which is
deployed to the Cloud runs in such a way that it is suited to a Cloud environment and makes full use of all resources available [13].

Furthermore, GridGain provides fully-customisable interfaces for producing distributed applications such as, which I have not encountered with Emotive Cloud. Also, In relation to the aims of this project, I believe GridGain to have a clear advantage over Emotive Cloud with regards to application portability. In balance, Emotive Cloud is perhaps better suited to the management of VMs and resources than GridGain due to its advanced scheduling capabilities including machine learning and agent-based scheduling.
6 Solution Design

This section will detail how I plan to go about designing an application with respect to how GridGain operates. It will also consider any other technologies I intend to use in order to design, deploy, test and evaluate my application within a cloud environment. For this to be done effectively, a cloud-based environment will need to be used.

6.1 Solution Design - Methodologies

In order to plan how I am going design my application, I first need to know about the current implementation, how the application functions, it’s benefits and it’s drawbacks. I also need to know who the current users are, what their level of knowledge is and how they interact with the current application. To gain a full overview of how the current implementation works, I have decided to use UML diagrams and activity diagrams to show the flow of information and to depict how each part of the program functions. These will be constructed by viewing the source code of the program to ascertain which functions/behaviours would benefit from being redeployed within a Cloud environment and from the work by Djemame et al., which gives a higher-level overview of how the application functions[10]. Furthermore, the application needs to be tested within a Cloud environment which will involve testing how long it takes for the application to function across varying numbers of GridGain nodes and virtual machines when compared to the current implementation.

6.2 The Histopathology Application

The histopathology application currently has a simple and easy to use interface. Figure 4 illustrates this along with a simple toolbar for zooming in and out of images, image rotation, image panning and even an image thumbnail so that the user can gauge what they are viewing in relation to their position within the overall image. Such a simple interface would appear to create an ideal training tool for trainee and even expert Histopathologists. Indeed, Treanor et. al. have researched ways in which the conventional microscope could one day become superseded by high-resolution, powerwall displays that depict very high resolution virtual images; allowing for greater overall detail, usability and a greater all-round view of human tissue to aid diagnosis[18]. The digital slide images themselves are created by allowing a scanner to pass over them slowly at an extremely high resolution to retain as much detail as possible. These are then saved to an image server for later retrieval.

As the images are stored on one image server, the application itself becomes quickly overloaded with image requests especially if there are numerous histopathologists using the application at any one time. It is under this kind of environment in which the application is mostly likely to be used. However, due to the drawbacks of the current implementation, the application
inevitably fails with regard to scalability and reliability due to its inability to efficiently process all image requests it receives and as a result the histopathologists experience significant delays when receiving processed images[10]. All of these drawbacks are compounded due to a distinct lack of available bandwidth, computational and memory resources between the client (which will process the image) and the image server itself. Furthermore, the greatest problem occurs due to the amount of resources required for executing the algorithms that are used in reconstruction of the 2D slides into a 3D volume[10]. In the current implementation this is all done on a single client machine with limited resources available, which would increase execution time. For example, this application processes images serially one after another and execution takes greater than 100 hours of processing time[10]. Therefore, it would be reasonable to infer that the more slides that are required to construct a 3D volume, the more resources and processing time that is required for complete execution. Therefore, for this application to be used effectively within a Medical environment, these issues need to be addressed as patients would require a quick diagnosis.
6.3 Solution Design - Application

The redesign of the application had to take into account the Histopathologists familiarity with the current application and the fact that there would be varying degrees of technical ability and technical expertise amongst them. Therefore, it was decided to keep the redeployment as simple as possible in terms of usability. Furthermore, it was also decided to redeploy a subset of the overall functionality of the original histopathology application using GridGain as a proof of concept prototype application. This was done to simplify the overall design and redeployment of the original application rather than attempt to redeploy the original application in its entirety. This would have involved complex calls to C libraries and would have inevitably added greater complexity to the redeployed application. Furthermore, it was decided to make use of UML and Activity diagrams to illustrate the users who may be involved with the redeployed application and how it will work. Finally, it was decided that the functionality of processing an image at a specified resolution and zoom level would be loosely emulated in the redeployed application using GridGain by creating an application that would either scale up or down any specified number of images by the user, whereby the actual scaling operation would be distributed across a number of GridGain nodes in a Split-Reduce fashion.

![Figure 10: How the current application operates](image)

6.4 Solution Implementation

To implement this application, it was decided that in order to make full use of the redeployed application then a cloud environment would be required to make full use of the computational and memory resources available. Therefore, the Cloud testbed in the School of Computing was

29
chosen in order to emulate the real-world Cloud environment and to demonstrate the application as a proof-of-concept.

6.4.1 The University of Leeds Cloud Testbed

The University of Leeds maintains a testbed of commodity hardware that functions to emulate a Cloud environment. It was decided that this environment would be suitable for the purposes of redeploying an application to the Cloud as this would be the closest environment to a real-world, fully-functional Cloud for implementation and testing purposes. Indeed, it was remarked by Djemame and Armstrong that the testbed was not sufficient to be classified as a fully-functional Cloud [5]. I believe this strongly to be an accurate representation of the Cloud testbed. However, given that this is a testbed, inherently it is only used for testing purposes and is therefore well-suited to be used for testing proof-of-concept applications as is the nature of my redeployed application.

The testbed is made up of 4 physical hosts with the capacity to run numerous virtual machines across the testbed as required [5]. Furthermore, the testbed makes use of OpenNebula and Nimbus as Virtual Infrastructure Managers for managing the Cloud environment and the KVM and Xen hypervisors for the management of each virtual machine. Additionally, as this is a Cloud testbed it also provides provisions for operation at the IaaS, SaaS and PaaS abstractions of operation. This would be beneficial if the testbed were to be used to investigate and experiment with the different levels of Cloud operation. As I had been given and instruction
to the Cloud testbed and also an introduction to the Xen hypervisor, this was a natural choice for my project.

6.4.2 Application Redeployment

As the purpose of this project is to redeploy an existing application, it’s logical to examine the existing code of the original implementation as it contains all necessary functionality to be implemented in the redeployed solution. However, after issues with setting up both GridGain and the original implementation to work at the same time, I decided to examine the C code of the original implementation and choose which functionality was most critical to the redeployed application and also which functionality could easily be replicated and emulated within my redeployed application; most notably altering the image resolution and scaling the image up or down as specified by the user. The predominant, overall functionality of reconstructing a 3D volume from a set of images was not emulated as this would have required much complex coding and additional external libraries due to GridGain be heavily Java based and such mathematical libraries for volume reconstruction in Java were not readily available. Furthermore, due to the aforementioned problems regarding getting GridGain and the original implementation to work at the same time, I was unable to make use of any existing code as it was written in C/C++ and GridGain predominantly deals with Java as this is the language in which it is written. Although external calls to C libraries could have perhaps been implemented, this would have added too much complexity and overhead to my proposed solution and in the interests of time I decided not to pursue this idea. Greater detail of the problems I encountered with my deployed application is covered within the Application Implementation section of this report. In contrast, as the original implementation resides and operates within a single client-server capacity, this makes it easier for evaluation purposes as my solution will operate within a Cloud environment and is therefore easier to compare both applications.

The original implementation consists of a single client and server, which is accessible through the Internet. This is advantageous as my proposed solution will make use of virtual machines, each of which will have their own IP addresses and will all reside on the same network. In this way, the HTTP communication between my proposed application and the GridGain nodes on each virtual machine correctly emulates that of original implementation. Furthermore, as GridGain makes use of a broadcast mechanism to ascertain all currently running GridGain nodes, using HTTP as the preferred form network communication is ideal and requires no extra coding as the broadcast and also the multicast mechanisms are inherently performed by GridGain by default.

As part of my design, I opted to make use of a pre-existing virtual machine based upon the Fedora distribution of Linux. This also included the Histopathology application installed as default. This was very useful as it meant that there was no requirement to install the Pathology...
application prior to beginning the design phase and therefore decreasing the overall administrative time spent setting up the virtual machine as all that was required was to set up the GridGain middleware. Furthermore, this meant the virtual machine could be cloned with exactly the same configuration and allowed me to run as many virtual machines as were required to explore the impact of running a number of virtual machines with each virtual machine running a number of GridGain nodes and monitoring if any increase in this overhead results in a decrease in my proposed solutions performance. Additionally, as each virtual machine would be on the same network, monitoring which virtual machine and GridGain nodes are involved in any computation whilst the application is running would be simple as it would just require the user to login to each virtual machine and check each GridGain node. However, within a large scale deployment of this application this method would be impractical. As an additional point to consider is the dynamic tear-down and initialisation of virtual machines and GridGain nodes as and when required. However, due to time constraints these areas of consideration shall be left as pointers for future work.

6.5 Solution Testing

Testing my proposed solution would involve two categories; Technical testing and User testing. The technical testing involves ensuring that my proposed solution is more reliable and efficient than the current solution and that the functionality is sufficient for a proof-of-concept application. The user testing relates to how simple my proposed solution is to use when compared with the original implementation. Carrying out the technical testing would involve submitting a set number images for processing as input and observing the actual output against the expected output. A measurement of the actual and the expected output could perhaps be as a function over time along with a benchmark of an acceptable level of execution time. This benchmark could possibly come from the length of execution time of the original Histopathology application with the aim of reducing how long the execution takes in order to increase the reliability efficiency of my solution when compared the original implementation. Furthermore, examining the image detail from the actual output of each implementation with respect to clarity will help to compare which implementation produces the most acceptable level of detail in the least amount of execution time. Another test to measure the scalability of each solution would be to increase the number of image requests for each implementation in tandem and observe the execution time. These tests will help to prove if my solution is better in terms of efficiency, reliability and scalability when compared the current implementation.

The user testing phase would involve assessing the ease of use of my solution by its intended users, primarily the Histopathology department at St James Hospital in Leeds and other stakeholders including researchers from the University of Leeds. My intention would have been to design a suitable questionnaire aimed at the aforementioned stakeholders and users that would
detail questions relating to my solutions reliability, scalability and efficiency as these are the main areas in which the current implementation is lacking. The questions would allow all concerned parties to rate my solution using a Likert scale and compare these results to that of the current implementation. I feel that using a questionnaire to test my solution would be an effective way of determining if my solution addresses the aforementioned and is indeed an improvement upon the current implementation. Furthermore, use of a questionnaire would be a more practical way of assessing user satisfaction with my solution when compared an alternative such as holding a meeting with all concerned stakeholders within the current system. This is because a questionnaire allows for more detailed and structured answers especially through use of an additional comments section where the users can suggest ways to improve the system based upon their own requirements as these may change over time. The questionnaire, therefore is a good alternative and provides a source of valued feedback based upon a user’s own experiences.

6.6 Solution Architecture

![Solution Architecture Diagram](image)  

Figure 12: The proposed solution architecture
Figure 12 depicts how my proposed solution will operate and illustrates a very high-level overview of how my solution will work. Although the actors Database and Image Server will not be implemented in my proof-of-concept solution, I have included them for completeness and clarity as it is hoped that they will be implemented at a later date and it provides a clearer illustration of the functionality of the original implementation, which is what my solution will emulate and improve upon. The main actors within my solution will be the Histopathologists, either fully-qualified or even trainee histopathologists that may wish to use this solution as a training tool. The digital slides will be retrieved from the image server, which will consult its own database of available images and then return the required image at a specific resolution and zoom level as specified by the Histopathologist. An issue to be addressed at this stage would be that of Security and confidentiality. In a fully completed implementation, this would ideally cover the use of passwords and perhaps the use of security certificates within a formal Medical environment. This would ensure that all data remains confidential and only viewed by authorised Medical staff not limited to Histopathologists as illustrated in figure 12. This functionality is purposely absent in my solution as it deals with images that are not urgently required for diagnosis nor are they representative of any patients currently being dealt with by the Histopathology department at St James University Hospital, which is where the current implementation resides. Therefore, there is no requirement at this stage of development for such stringent Security policies and as such the login credentials to the University of Leeds Cloud testbed are sufficient. Additionally, the GridGain Nodes actor consists of a Master node and a number of Worker nodes. The Master node delegates work to worker nodes that have sufficient resources free such as memory and computational resources in order to process any images which have been supplied to the solution application. The GridGain Nodes actor is just a generalisation of these attributes.

In order to further clarify and illustrate how my solution application will operate, I have decided to implement an Activity diagram which will show the flow of information within my solution and convey the overall functionality. Figure 13 illustrates the flow of information of my proposed solution. Again, the Database and Image Server actors have been included for clarity and completeness. The Histopathologist requests their desired slides at a specific resolution, image region and zoom level for the 3D volume reconstruction by typing a set of meaningful image names at the commandline and these names are queried by the image server within a database, which also resides on the image server itself. Although the use of a GUI to aid in using this application would probably be beneficial in the long term, in the interests of time I decided to keep a simple, easy to use text interface whilst concentrating on the functionality of my solution. After successfully looking up the required images, the database forwards the requested images back to the image server, which then proceeds to forward the requested images to the GridGain head node for processing. The GridGain head node oversees the processing of this set of images as a task, which is broken down into equal sub-units of work known as jobs. Each job is
Figure 13: Activity diagram showing flow of information

forwarded to a GridGain worker node that performs the actual processing of the image, in parallel, and returns the result back to the GridGain head node. The GridGain head node is then responsible for the aggregation of the results of each completed job received from each GridGain worker node and then passes the processed set of images and completed 3D volume reconstruction back to the user application, which then displays the completed 3D volume to the user. The idea is that as the actual work is done in a distributed fashion, the user should see the benefit from reduced processing time and more efficient use of available resources as GridGain monitors available resources automatically. More detail of how this is done such as load-balancing is explained within the GridGain subsection contained within the Background Research section of this report.
7 Solution Implementation

This section will detail how I went about implementing my solution using the University of Leeds Cloud testbed and the Java-based middleware GridGain. It will detail any issues that occurred with my implementation how I managed to overcome these issues.

My implementation first began by getting to grips with University of Leeds Cloud Testbed. This firstly involved a group session with other final year project students on learning to use the Cloud testbed along with how to install relevant software and learning how to use the hypervisor Xen and the Virtual Infrastructure Manager OpenNebula. Furthermore, I also used a pre-made virtual machine image based on the Debian version of Linux which already had the Histopathology application pre-installed on it with all available libraries. This was advantageous as it meant that I did not have to consume time by setting up the current Histopathology application manually and also spend time transferring virtual machine images to the Cloud testbed, which were quite large in size and would have unnecessarily used bandwidth resources available to the Cloud testbed. I then had to download and install the GridGain middleware from their website.

Trying to install GridGain onto the actual testbed was more difficult than anticipated as although the virtual machine and the testbed both had access to the Internet, due to the way in which the GridGain website was constructed I was not able to use a command such as “wget” in Linux in order to easily download the GridGain source code. In order to get the GridGain source code onto my Debian virtual machine, I had to resort using the FTP protocol from my desktop in order to transfer the GridGain source code to cslin-gps and then from cslin-gps to the Cloud testbed and then finally from the Cloud testbed to my Debian virtual machine image. Although this was an adequate workaround it was quite time-consuming and did delay the implementation of my application slightly.

Setting up the virtual machines required for the redeployed application was difficult as it required research on how to set the required Java classpaths for both GridGain in order to construct the redeployed application and the current implementation of the Histopathology application. Furthermore, it also required much administrative work in terms of setting up each virtual machine to work correctly each time they were initialised. To counteract the time required to set up each virtual machine, I created a shell script which would run each time a virtual machine was initialised and would automatically set all required classpaths without having to manually reference each JAR file before even beginning to construct the redeployed application. This had the advantage of cutting down on the development time, which would have been extended considerably had a shell script not been created. Additionally, as the shell script took care of the administration of the virtual machines, this meant that additional virtual machines could be created and reused as required without any time-consuming administrative
work and allowing the application to be scaled up or down across virtual machines as needed.

Unfortunately, working directly on the virtual machine whilst it was running remotely on the Cloud testbed did incur certain disadvantages. One such disadvantage was the amount of resources available for use at any one time was limited as understandably the Cloud testbed is used for testing purposes and does not realistically resemble the dynamic scalability and reliability of a commercial Cloud nor is it subject to the stringent Service Level Agreements that may be adhered to by real-world Cloud environments such as Amazon EC2. Another limiting factor was the fact that at least two other final year projects were being conducted on the Cloud testbed which further limited the amount of available computational and memory resources and which did in fact result in downtime to the Cloud testbed due to an unforeseen error in allowing a user to create more virtual machines and allowing them to use more resources than there were currently available. Furthermore, the Cloud testbed also sustained another error in which the Cloud testbed failed again. Although these problems were quickly resolved, this resulted in extended development time and although some of the redeployed application coding could be done on a single machine, the amount of resources are severely limited when compared to the resources available on the Cloud testbed and development on a single machine would not be as comprehensive in terms of testing and evaluation when compared to the Cloud testbed.

After successfully setting up GridGain to work, I encountered quite a large issue from which I had to conduct a workaround for and effectively change how I would redeploy my application using GridGain. I found that although GridGain worked quite well on each virtual machine and each GridGain node could effectively recognise each other regardless of which virtual machine they resided on, the current implementation of the Histopathology application had ceased to function and would not recognise the libraries that were present in order to execute successfully. This was a substantial issue as I required the current implementation to work in order to evaluate it against my redeployed solution and to effectively redeploy the current implementation within a Cloud environment using GridGain.

After setting a classpath I found that the current implementation began to work however, GridGain and more specifically Java, which GridGain relies upon, began to display segmentation fault errors. As this is quite a major issue I initially tried setting different classpaths for both GridGain and the current implementation but I was unsuccessful in getting both the current implementation and GridGain to work correctly at the same time. As I had spent much time trying to unsuccessfully correct this problem, I decided to produce a solution that would emulate some of the functionality from the current implementation; more specifically processing a digital slide at a specific resolution. This would be sufficient enough to provide a proof-of-concept application that would function as an emulation of the original histopathology application without any unnecessary calls to C libraries as inevitably would have been the case if I had decided to port the original histopathology application to the Cloud using GridGain.
Furthermore, the revised application reduces any unnecessary overhead that would have been created by function calls to C libraries.

After deciding upon how I was going to implement my application, I then approached the problem of how I was going to emulate the functionality of the original Histopathology application. After studying the original implementation, I realised that much of the functionality was to provide a digital slide to the user at a specified resolution and zoom level. To emulate this, I decided to produce a solution which would take any number of images as specified by the user at the command line and then either scale the images up or down as specified by the user and subsequently save this new image in JPG format to disk in order to be viewed by the user. As the GridGain middleware comes with many examples of Cloud and distributed applications, it allowed for some code reuse for certain functions in the implementation of my solution and therefore, reduced the overall development time. Furthermore, GridGain comes with very comprehensive and informative Java documentation which further enhanced my understanding of how GridGain operates.

Another stage within my implementation was how I was to approach implementing my solution with respect to coding style and which programming paradigm I was to follow. GridGain offers much flexibility to the user and supports many programming paradigms including functional and imperative styles. Initially, I had decided upon a functional paradigm as this reduces much of the boilerplate code required to implement any required functionality and significantly reduces development time. However, after considering what I wanted my solution to implement, I decided on an imperative style of development. This was because it allowed me the flexibility to implement the functionality I required, which in this case was a Split-Reduce kind of functionality and also provided the efficiency of importing any required functions as included with the GridGain middleware.

After coding my solution, I encountered a problem with viewing the newly created, processed image due to producing my solution remotely on the Cloud testbed. Although I was given instructions on how to forward the virtual machine desktop to the desktop I was working from, I was unsuccessful in forwarding the desktop. As a workaround, I had copy the newly created image to the desktop I was working from and view it locally. This is a potential drawback as it created unnecessary overhead in transferring the newly created image and undesirably uses available Cloud resources and network bandwidth.

The resulting solution as described above and within the Design section of this report does emulate some of the functionality of the original Histopathology application and is sufficient as a proof-of-concept application. However, due to unresolved issues with easily viewing the processed images and with issues relating to executing the original implementation and my own solution at the same time, in the interests of time these will need to be left unresolved and may perhaps provide a pointer for improving and evaluating this application at a later date.
8 Solution Evaluation

This section will detail my planned testing and evaluation for my solution had it been completed with the full functionality of the original implementation. This section which will be split into further relevant sections relating to the, technical, system and user evaluation of my solution including evaluating why GridGain was chosen. Furthermore, this section will detail the conclusions I have drawn from my solution and suggest further work to be done to enhance my application, perhaps as part of another project.

8.1 Solution Testing

In order to fully test my application, there are a number of areas that would need to be tested; most notably the functionality. This would test that the application works as desired with all valid inputs. This simply involves choosing valid digital slides for processing at the commandline and the application returning an image that has either been scaled up or down as specified in the code by the user. An improvement to this would be to allow the scale factor be specified at the commandline by the user rather than within the code. This wastes much needed time in the diagnosis of tissue as time is no doubt a key factor in the efficient and early diagnosis of a very serious disease such as Cancer.

In addition to testing the functionality of my solution, other key factors to be tested are the accuracy, efficiency, scalability and reliability of my solution. These factors are very significant in a realistic Medical context as the solution has to be robust enough to consistently deliver multiple slides at a user-specified resolution within an acceptable time frame in order to be practical for Histopathologists to use on a daily basis. As my solution is a proof-of-concept application it does not implement all of the functionality of the original implementation such as retrieving an image at a specific zoom level. However, had my implementation been fully completed with the same functionality as that of the original implementation, I would test that my solution produces an output image that is of the same accuracy if not more accurate than the returned image of the original implementation. This would involve using a selection of images of varying sizes that would be submitted for processing on the original implementation and my solution and the accuracy of each return image compared. For example, in a pair of digital slides that contain the same cancerous region, how detailed and accurate is each image and how long does it take for the Histopathologist to diagnose the tissue seen using each method.

Additionally, it would be reasonable to stress test my solution under a high load of image requests, thereby testing the reliability and efficiency of my solution. This would be done by allowing a group of Histopathologists to request a minimum of 4 images each at varying resolutions and recording how long it take for my solution to return the processed images when compared to the original implementation. I would also increase the number of images submitted
to the Cloud testbed by one for each Histopathologist and record the length of time it takes to return a result. This would give an idea as to the maximum number of images that can be submitted to Cloud before we begin to see a reduction in performance. It would also be reasonable to infer that a Histopathologist will work on a set of digital slides at anyone time, thereby making this a valid test to conduct. Furthermore, another test would be to compare the speed of diagnosis using a digital slide to that of a conventional microscope. As this is currently outside the scope of this project, this is could perhaps be a potential extension in future years.

In addition to the aforementioned tests, I would also introduce a test to measure the scalability of my solution whilst under an excessive load. This excessive load would be determined by the execution timing results as detailed in the Application Testing section of this report. This test would measure my solutions ability to adapt and more efficiently manage the resources that are available and would involve the dynamic introduction of extra virtual machines and GridGain nodes as are optimally required to deal with this extra load. All of the aforementioned tests for reliability, accuracy, efficiency, scalability and reliability should be carried out on the current implementation and compared against the results of my solution. This will help to establish if my proposed solution is a reasonable improvement upon the current implementation.

8.2 Technical Evaluation

This section will detail the planned evaluation the technical testing of my solution and will focus on the main points of reliability, scalability and efficiency. One such test would be evaluate the execution time of both my proposed solution and the current implementation. To begin with, this will involve submitting two images to each implementation, which each image at a fixed resolution and zoom level and recording the execution time. The setup of my solution prior to this test will involve 3 GridGain nodes, consisting of one head node and two worker nodes, all of which will be on one virtual machine. This will allow the processing of the 2 images to be distributed between the two GridGain nodes. The number of images requested for each implementation will then be increased by one for each execution and the execution times will be noted accordingly. I anticipate that, due to the distributed nature of my solution it should perform better with a greater number of image requests when compared to the current implementation due to its distributed nature of operation and the automatic management of resources by the GridGain middleware between the GridGain nodes. However, I would expect the execution time under this load to plateau under a certain number of image requests and a specific number of GridGain nodes before the system has to consider initialising another virtual machine to cope with a high load. Additionally, at this point other factors pertaining to network communication between the GridGain nodes on each virtual machine will have to be considered including available bandwidth and network latency as these may heavily influence the execution
time. Conducting such a test will allow both implementations to be compared in detail under varying load conditions. Furthermore, this allows us to see if a test fails and therefore, if a system fails to address any of the current issues relating to scalability, reliability and efficiency.

Testing the reliability of my solution would be conducted as an extension to the tests mentioned previously. As mentioned in the Application Design section of this report, it is important that both implementations make use of a benchmark of execution time in order to determine what execution time is acceptable. This is paramount as realistically this system may be adopted for use in a real-world Histopathology environment and speed and accuracy of diagnosis is obviously significant in a patient’s overall recovery, welfare and well-being. Using this benchmark as a limit upon overall execution time will help to determine which implementation is more suitable in a real-world Histopathology environment. As this is a proof-of-concept application, its evaluation and testing is to be done upon the University of Leeds Cloud testbed, which is subject to limited available resources. Upon completion with the full functionality of the original implementation, my solution would benefit from more rigorous and thorough testing within a real-world, commercially used Cloud environment such as Amazon EC2. This would give a more robust evaluation of my solution when compared with the current implementation as there are a greater number of available computational, memory and storage resources.

8.3 User Evaluation

This section will detail the planned user evaluation tests that would have been carried out had my implementation been completed including the full functionality of the current Histopathology implementation. To effectively evaluate a user’s experience using both the current implementation and my solution, a number of tests would need to be conducted involving the Histopathologists as the end users. These would include similar tasks being conducted on both my solution and the current implementation involving exercises and routines that would be carried out within a real-world diagnosis. This will effectively establish which implementation is the easiest to use and diagnose slides with. Furthermore, Treanor et al. proposed use of a training set of slides to enable the users to familiarise themselves with the Powerwall as a tool for tissue diagnosis [18]. I believe this to be an effective way of allowing the Histopathologists to familiarise themselves with a new system of diagnosis prior to real-world use. To that end I propose that a limited amount of time be allowed for the Histopathologists involved with the current implementation to use and familiarise themselves with my proposed solution to prevent any inaccuracies in its usage from affecting the overall indication as to which implementation is favoured whilst these tests are conducted. Additionally, a key factor in the user evaluation is the time in which diagnosis would take when compared to the current implementation and the introduction of an average diagnosis time for each task whilst using implementation along with corresponding error bars to indicate the confidence interval during each task [18]. Fur-
thermore, the introduction of a median and possibly standard deviation of the times taken for diagnosis during completion of each task would help to illustrate the spread of data and give a clearer illustration as to the effectiveness and practicality of each implementation. Also, it may be useful to ask the Histopathologists how aesthetically pleasing the user interface is of each implementation and how this could be improved in order to aid and ultimately speed up the diagnosis process. This may be through the inclusion of additional functionality such as a simple point-and-click interface and may well draw upon the domain knowledge of the Histopathologists.

This section has focused particularly on the user experience whilst using each implementation and how this can be effectively evaluated and improved. Ultimately, the speed and ease of accurate diagnosis are the key factors and the ultimate goals of both solutions and these have been focused upon overall in this section. This is in relation to how it can be improved through an enhanced user experience and therefore, an increase in the ease of use and diagnosis of diseased human tissue.

9 Evaluation

My evaluation will focus primarily on the development phase of my project and the work completed as my application is not complete as it does not visually show the user the newly created images when processed on the cloud as originally intended although the newly created images are saved to disk and they are processed either by scaling the image up or down as specified by the user on the commandline. However, my solution does emulate some of the functionality of the original implementation by returning an image at a specific resolution and therefore is suitable as a proof-of-concept application. Furthermore, due to this problem and time constraints I have been unable to conduct a thorough user evaluation of my solution. Furthermore, I was successful in getting my application to work across multiple virtual machines and a future intention of this solution would be to offer it as a service to the Histopathologists rather than as a standalone application thereby utilising the Software as a Service cloud computing paradigm.

9.1 Development and Middleware Evaluation

This section will focus on why GridGain was chosen and what it’s advantages were during the development of my solution. Furthermore, I can also evaluate the work done with regard to the functionality of my solution and how that relates to the original implementation. These are all major factors within my project to be considered and should provide an indication as to how my solution can be improved upon in future years. Additionally, GridGain has never before been used within a Cloud Computing project in the School of Computing and it would be interesting to learn from the advantages and disadvantages of this middleware and how effective it is as a development tool in the redeployment of future applications to a Cloud environment.
9.1.1 Virtual Machines and GridGain

GridGain was the selected middleware for this project and has never before been used within the School of Computing. GridGain is a relatively new middleware when compared with other similar technologies such as Globus Toolkit and is therefore not as widely adopted or supported. However, during my research phase I had downloaded and installed GridGain on my own laptop prior to designing and implementing my solution, which therefore gave me some experience with learning how to use GridGain. Furthermore, there is also an online book, which is currently a draft version available to read on the GridGain website as well as a user forum. These resources provide valuable information on how GridGain works including an open community of users with which to ask questions and solve any problems that may be encountered whilst developing with GridGain.

A distinct advantage to using GridGain was the availability of online and interactive video tutorials on the GridGain website and a downloadable whitepaper which illustrated useful information regarding how to use GridGain effectively and highlighting its strengths. Furthermore, these video tutorials provided and explained detailed demonstration code which reflected applications that were successfully deployed and used within an Cloud environment. I made extensive use of this tutorial during my design phase in order to learn how GridGain worked and even successfully constructed a simple program involving typing in a phrase or a word that was equally split up and processed between each GridGain node in a MapReduce fashion. This was an invaluable part of the tutorial as it enabled me to learn functions that were similar to my design and provided a clearer illustration as to what functionality could be implemented using the classes, interfaces and functions supplied by GridGain. With respect to the GridGain documentation, further additional online documentation explaining the different classes, objects and functions that make up the GridGain middleware. However, a drawback of this tutorial while being an invaluable tool to learning GridGain does make some assumptions on the users knowledge of Java and requires that the user has beyond a basic knowledge of Java in order to use and understand how GridGain functions. This is one phase of my project where I felt I had to build upon my current knowledge of Java prior to gaining a good understanding of how to use GridGain effectively in order to achieve my project aims.

As GridGain is heavily Java based, it requires no special compilers or interpreters in order start working with GridGain. As Java is installed by default on many Linux and Windows machines, it only requires the use of a basic text editor in which to start developing GridGain applications and therefore reduces the administration time required to get GridGain functioning correctly. Furthermore due to GridGain's Java foundation allowed for support for standard Java libraries including those relating to image manipulation. This was a strong advantage as this meant that part of the original implementation's functionality could be emulated using this standard library and eradicated any requirement to setup and install any external libraries.
in order to achieve the same functionality. Furthermore, GridGain also provides interfaces to well-established commercial cloud environments including Amazon EC2 and even mobile device firmware such as Google’s Android operating system [13]. This functionality could be very useful in redeploying my solution to a commercial environment, thereby increasing its reliability, efficiency and scalability when compared to redeployment on the Cloud testbed due to the abundance of computational, memory and storage resources offered by these Cloud environments.

Another advantage of GridGain was that fact that it could be installed and used on a virtual machine. This meant development could be done remotely on the virtual machine allowing for easier deployment and removing the need for transferring development files across the network. This saved on much needed network and bandwidth resources and prevented them from being used unnecessarily. However, as mentioned in the Implementation section of this report, GridGain was required to be downloaded locally and then transferred using sftp to the Cloud testbed. Although this was a temporary use of network resources it unfortunately resulted in a slight delay in development time. Additionally, as the original Histopathology application was located on a virtual machine this meant that it could be duplicated along with its configuration without the need for constructing a new virtual machine. This also reduced administration time spent setting up the virtual machine as the only configuration that was required to be altered was the IP address and allowed GridGain to be transferred directly to the Histopathology virtual machine using sftp. However, a disadvantage of installing GridGain directly to the Histopathology virtual machine meant that various Java classpaths were required to be set in order for GridGain to function correctly. This inevitably increased the administration time spent getting GridGain to work.

A disadvantage of using the virtual machine was that it was only accessible through a terminal interface and therefore any modification that were required within the virtual machine were done using a text-based interface. This added complexity to the development of my solution as I was unable to use any kind of visual IDE such as Netbeans or Eclipse in order to speed up the development process. Although I was given instructions on how to forward my virtual machine desktop to the local desktop I was working from using VNC, I was unsuccessful in getting this method to work. Furthermore, due to this drawback, I was restricted to using text-based editors such as Emacs and VIM and I found it difficult to set up all required jar files in order to get my solution to work correctly with GridGain. In order to solve this problem some development was done locally using an IDE and then later transferred to my virtual machine and compiled using Java. Additionally the lack of a good GUI made certain administrative tasks more time-consuming such as setting up all required Java and GridGain class paths that were essential to the function of my application. Furthermore, as there were numerous jar files which had to be referenced explicitly within the JAVA HOME system variable within Linux,
doing this at the commandline proved to be a tedious task that would benefit from being automated when GridGain is first installed and is a suggested improvement for future releases of GridGain. Also, as I was unable to forward my virtual machine desktop I was also unable to visually demonstrate the images that had been processed by GridGain and it was therefore difficult to demonstrate the results of my solution without first transferring the images from my virtual machine to the Cloud testbed and then from the Cloud testbed to my local desktop in the School of Computing. Again, this unnecessarily uses additional network bandwidth.

Another potential advantage of GridGain is that it allows a user to monitor jobs and task currently running on the current topology remotely as well as locally by logging into a remote GridGain node using the Java console. Although I made limited use of this functionality, it is worth noting that this functionality could be used to efficiently monitor GridGain node metrics such as CPU, memory and storage utilisation as they are visually illustrated to the user in the form of a graph for each different metric. These graphs could be used to tweak my solution and provide an insight into how my solution could be improved to work more efficiently and accurately especially during the processing of images to reconstruct a 3D volume by monitoring the length of execution of each job on each Grid node. However, a potential disadvantage of GridGain is the lack of Security provided especially relating to user and system security and permissions. Although my application was a proof-of-concept application and was only to be used by myself, within a Medical environment issues relating to data protection, data security and patient confidentiality are usually subject to stringent government legislation and it is worth noting that GridGain does not appear to provide any kind of user or data security. For example, there is no provision of allowing a user to login with a secure password to the current GridGain topology prior to submitting a task for processing. Therefore, there is no way of preventing unauthorised access to data submitted to the GridGain topology. Although this may be provided by the organisation that is running a GridGain topology, providing a default solution to the user that may be customised as required would be advantageous.

Using GridGain as the chosen middleware for my project appears to be a well-informed choice. For example the GridGain documentation including the online user guide, community forum and wiki were major focal points as they provided me with invaluable information and guidance with how to use GridGain effectively and to ultimately produce my solution. Furthermore, as I had never worked with a Cloud environment before, the provision of online video tutorials and demonstration code helped to enhance my understanding of how Clouds and Grid environments operate and also provided me with the initial ideas and inspiration of how I was to undergo constructing my solution application. Although, as mentioned within my Implementation, I was unable redeploy the original implementation as originally intended, GridGain provided ample resources and examples to construct a proof of concept application that emulated some of the functionality of the original implementation and thus enabled me to fulfill
my minimum requirements. This was further helped due to GridGain’s native support of Java and therefore supported many of the standard libraries I could make use of to construct my application. Although GridGain is not without its problems such as the lack of an effective GUI to aid development and debugging and also the lack of a default Security framework, it’s core functionality appears to be very effective and efficient in producing Cloud applications, which is further enhanced with it’s support for commercial Cloud offerings such as Amazon EC2 and also the mobile computing market with its support for the Android operating system[13]. Another clear advantage is its native support for virtual machines as it can be installed and used as if running on a local desktop. However, support for virtualisation is very much a standard offering for middleware although this seems to be integrated with ease using GridGain as automatically manages all available resources dynamically between GridGain nodes within a GridGain topology. This is further complemented by GridGain’s native support for networking, therefore allowing it to scale to geographically disparate proportions. Finally, after considering the advantages and disadvantages of GridGain I am certain that GridGain was the correct choice for this project taking into consideration the project’s aims and minimum requirements. However, for future projects that may be of a similar nature to this it may be worth undertaking training and experimentation with different middlewares prior to the design phase of the project. This is to ensure all potential problems with any middleware are realised early on within the project and that adequate solutions are implemented, thereby enabling a better-informed choice of middleware.

9.1.2 Development Evaluation

This section will focus on the significant steps followed within my development phase in order to evaluate the development process as a whole. I began my development phase by setting up the pre-created Histopathology virtual machine to work with GridGain. This involved installing GridGain onto the virtual machine and setting up all required classpaths to allow GridGain to function properly. Then I had originally intended to redeploy the existing Histopathology application to the cloud testbed using GridGain. However, due to a problem that I was unable to solve as detailed in the Implementation section of my report, I was unable to this. Therefore, I created an application that would emulate some of the functionality of the original implementation and act as a prototype application that would operate within a Cloud environment. Finally, after designing and constructing my application, the application would then be implemented and deployed to the Cloud testbed from within the virtual machine. However, as I was unable to display the resulting images after being processed by the Grid nodes within the GridGain topology, the implementation remains partially complete.

At the beginning of my design phase, I required a virtual machine on which to install and setup GridGain alongside the original Histopathology implementation. As the virtual machine containing the Histopathology application was already setup prior to use, I decided to install
GridGain on this virtual machine, which helped to reduce the overall development time as much of the administration required to setup the virtual machine had already been done. Deciding to use a pre-created virtual machine appears to be a good decision as much of the administration required to setup the virtual machine was eradicated. Furthermore, if I required another virtual machine in order to increase the scalability of my application I would have only needed to make a copy of the existing virtual machine and it’s configuration file and make changes to the hostname and IP address within the virtual machine whilst keeping all existing software and setup parameters from the first virtual machine. This further reduces any required setup time that would have been dedicated to the setup of each virtual machine and is overall beneficial to the time taken to produce a solution.

The virtual machine was located remotely on the Cloud testbed, which brought a number of benefits. Firstly, the development of my solution was done directly on the virtual machine, which allowed for any code edits to be done directly and the application to be compiled and ran instantly on the Cloud testbed. This prevented any need for data transfers between the local desktop and the Cloud testbed and thus freed much required bandwidth and memory resources on the Cloud testbed. Although as a notable drawback to developing remotely on the Cloud testbed was that it was subject to Cloud testbed failure. This did unfortunately occur at one stage, which resulted in lost development time. Another benefit to developing within a virtual machine located on the Cloud testbed was the availability of more computational, storage and memory resources when compared to developing on a single machine and allowed for a more comprehensive and thorough testing and evaluation phase. Another advantage of using a virtual machine was that in the case of this project the virtual machine was pre-created prior to use. This removed any requirement to create a new virtual image from scratch and therefore, removed any requirement to setup and administer the virtual machine and only required that GridGain was installed and setup as needed. This enabled greater time to be spent developing the solution rather than wasting time trying to overcome administration issues such as those relating to network connectivity prior to development. Another advantage to using a remote virtual machine is that there is no requirement for it to be transferred across a network from a local machine to the Cloud testbed. This is advantageous as the virtual machine is GBs in size and removing this preserves the available network bandwidth on the Cloud testbed.

However, use of a virtual machine that is hosted remotely on the Cloud testbed is not without it’s disadvantages. Firstly, as mentioned previously developing remotely on a virtual machine can be subject to failure of the network or the Cloud testbed; the latter being the case during my development process which resulted in lost development time. Furthermore, the Cloud testbed can be subject to planned maintenance at any which would further delay development. Secondly, the availability of resources can be limited depending on the number of users who are working on the testbed. In the case of my project, there were two other project students making
use of the available resources on the Cloud testbed as the same time. Consequently, this can result in reduced availability of resources which may affect the testing of my solution. Thirdly, hosting a virtual machine remotely on the Cloud testbed would mean the virtual machine would be subject to the network and security policies of the Cloud testbed and is therefore not a fully managed host under the complete control of the user. This may be a drawback in terms of manageability as it may be desirable to implement bespoke network and Security policies for the purposes of testing and evaluating my solution. In light of these drawbacks, I believe developing my application directly on the Cloud testbed was a good choice not least because it enabled me to instantly edit and compile code when required but that it also saved time by allowing me to make use of a pre-created virtual machine.

As the virtual machine used was created with the original Histopathology application pre-installed, this allowed me to execute the application and to observe its functionality. Furthermore, as the source code was made available this further enhanced my understanding of how the Histopathology application functioned and allowed me to make some informed judgements as to how I would proceed to develop my own solution. However, it was at this stage of my project during the design and implementation phases that I ran into problems getting both GridGain and the original Histopathology application to work at the same time. It was my original intention to redeploy the original Histopathology application which would have involved external calls to C libraries as this was the language in which the original Histopathology application was written. Therefore, my design and implementation were heavily influenced due to these problems and because of the significant amount of time spent trying to solve the problem of getting both GridGain and the Histopathology application to work, I decided to implement a proof of concept and prototype application. In hindsight, this may well have been a positive decision because GridGain is heavily Java based and the Histopathology application is based in C language and trying to redeploy a C program onto a Cloud using a Java-based middleware would have undoubtedly involved complex calls to external libraries. This would have added much unrequired complexity to my solution. After making this decision, I considered how I would go about trying to emulate some of the functionality of the original Histopathology application. As GridGain comes with demonstration code of programs that have been successfully implemented using GridGain, it was beneficial to observe how these applications work and apply the same principals of design to my own solution. Furthermore, designing and implementing my solution in this way allowed me to make use of standard Java libraries for the functionality of my solution and removed any requirement for external libraries and ultimately resulted in less taken to design and develop an appropriate solution.

Finally, after creating a solution the last part of my project involved deploying it to the Cloud testbed. This involved executing my solution on the Cloud with all relevant inputs and observing the output of each GridGain node to observe which GridGain nodes were involved in the
execution of each task. This stage went fairly well as I could clearly observe which GridGain nodes has completed each task during the processing of the input images. However, although the resulting images were successfully saved to the Cloud testbed, I was unable to visually observe the resulting images and therefore, I could not observe how the resulting image had been changed in terms of the resulting resolution and scale when compared to the original images. This was due to me being unable to successfully set up VNC in order to observe the resulting images remotely. However, I am confident that the resulting images would display successfully if I had local access to the Cloud testbed.

A point worth noting is how I approached the management of my project overall. I decided to implement tasks and key milestones within a modular fashion and further break down tasks into sub-tasks. This allowed for better time management and the creation of clear and structured goals at each stage beginning with my Background Research section all the way through to my evaluation. I believe approaching my project in this way was key to motivating myself to consider each goal and milestone at each stage of my project. Overall I feel the project has gone fairly well albeit in the absence of a fully working solution. I feel I carried out each task within time with the exception of my implementation and design sections. In order to plan for future errors and obstacles I believe I would benefit from further research and training in subject of any future projects I may undertake in order to be better prepared for any eventualities that may hinder future progress. The implementation and design phase of my project has also given me a sound foundation into how Cloud technologies and applications function and has provided an invaluable wealth knowledge that will benefit me in any similar future projects. Furthermore, as I did manage to deploy my application to the Cloud testbed it does at the very least prove that redeployment of an application can be done although any future project students undertaking similar work may wish to use this project as an example of the advantages and disadvantages of redeploying an application to a Cloud and learn from the potential pitfalls. As an element of advice for future project students, it may be worthwhile to research and investigate examples of C applications that have been successfully ported to a Cloud environment using a middleware written in another language.

9.2 System Evaluation

This section will evaluate what was completed within my application and discuss what would have been the overall outcome of my solution. In relation to my project this involved the creation of a prototype application which successfully ran from within a virtual machine. The Activity diagram within the Application Design section of my report best illustrates how the application would have functioned had its implementation been completed. As I successfully managed to operate my solution from within a virtual machine it would be reasonable to infer that my solution would have the capability of being deployed across all of the hosts within the
cloud testbed and could therefore be scaled up and down as demand requires and therefore allow it to be operated as a service. Although I was unable to successfully view the resulting images my solution does produce the output as it is saved to the relevant storage media and the problem of viewing the images remotely when not directly accessing the Cloud testbed is something to be considered as an unresolved problem for resolution in the future. Additionally, some of the functionality emulated in my solution could be compared to some of the actual functionality of the original Histopathology application and should therefore serve as proof that the original implementation would indeed benefit from redeployment to a Cloud environment. However, for future work it would be worth considering how beneficial it would be redeploying an existing application to a Cloud environment and exploring if any of the current functionality would benefit from being rewritten to advocate functioning within a distributed manner inside a Cloud environment.

My project minimum requirements were as follows:

- Design a redeployment strategy for redeploying an existing application to the cloud using GridGain.
- Construct a prototype, redeployment application of the current implementation using GridGain.
- Deploy a prototype, redeployment application of an existing application on to the cloud using GridGain.
- Demonstrate a prototype, redeployment application of an existing application within a cloud environment under GridGain.

The design and implementation phases of my project illustrated that a prototype was successfully designed to be redeployed on to a cloud environment. As this was successfully achieved it allowed for progress to be made to construct a working solution, which in this case was a prototype Java program that emulated some of the functionality of the original Histopathology application; albeit a solution that does not successfully display the resulting images as processed by the GridGain topology. As these requirements were completed it allowed for deployment of my solution within a virtual machine. This should serve as proof that my solution would have the ability to be deployed onto the Cloud testbed and operate from within the virtual machine. Furthermore the fulfillment of these requirements would allow my code to be demonstrated and it was demonstrated albeit in the absence of any significant output which would have been required to show the resulting processed images. This is unfortunate but at the very least it demonstrates that my solution does function and that the output is saved to the relevant storage for viewing. My code listing in Appendix B along with comments will highlight the strides taken to construct an application with meaningful output.
9.3 Project Conclusions

Conclusively, I believe that the planning and design phase of the project overall went quite well. I feel that the ideas and structure that I placed upon my solution were beneficial to achieving the aims of the project. This was due in part to approaching the planning and preparation of my project in a modular and measured way and allowed for achievable and demonstrable goals and milestones. I also feel that choosing GridGain to produce such a solution to the specified problem was a well-informed decision. This is because GridGain allowed for the flexibility of approach in how to design, construct and ultimately implement a solution through its choice of programming paradigms and an active user community who were approachable for help regarding any difficulties in using GridGain. Furthermore, the addition of example code along with a user guide helped to improve my understanding of how GridGain operates as a whole.

However, due to the problems encountered whilst implementing my solution, I believe it would be of huge benefit to investigate and research successful implementations of applications written in a language that is different to the language used to construct the middleware. This should illustrate a wealth of information highlighting the advantages and disadvantages of this approach and to avoid the problem of being unable to get both the middleware and the application to work at the same time under the same virtual machine which is identical to the problem I encountered during my implementation phase. However, overall I believe Clouds are of great use within this kind of project due to their abundance of resources and the parallel nature in which they operate that speeds up the execution time of an application that clearly functions in a parallel nature but that its current implementation prevents it from doing so; thereby resulting in a decrease in performance as is demonstrated by the disadvantages of the current Histopathology implementation.

10 Further Work

There are numerous issues which maybe tackled relating to the current implementation; most notably the size of the medical images being used and the complex nature of reconstructing a 3D volume of images. Due to the way in which the 3D reconstruction is carried out and the amount of data involved in doing so I believe extending the resources of the Cloud environment upon which it would ultimately operate would improve overall execution time and produce more accurate results. It may also be worth redeploying my solution within a commercial Cloud setting to test this theory prior to investing in a large Cloud infrastructure. This suggestion would also address the large amount of data involved in processing medical images GBs in size as there would also be an abundance of network bandwidth in relation to the transfer of these images. However, network latency would also have to be addressed in this case and how this would impact upon the overall execution time of my solution. Using the colocation
functionality within GridGain whereby the execution process is located close to data that is used regularly may help to elevate the problem of network latency as less transfers of data between the GridGain nodes involved in the execution process would occur and therefore more bandwidth would be available for use if required.

Another suggestion of future work to improve this project could be to evaluate the algorithm being used to reconstruct and ultimately produce the 3D image volume. It may be worth reviewing how the algorithm operates and if it can be optimised for use within a Cloud environment in order to make maximum use of the resources available within a Cloud environment. Another suggestion to be made would be to improve the GUI currently used within my solution. It would be easier for the Histopathologists to make use of a point and click mechanism when choosing the required images to be involved within the 3D reconstruction process. This would be more beneficial that the current commandline system as it would decrease the amount of time required to use the application and ultimately speed up the diagnosis process. Furthermore, a better, more elaborate and useful user interface would introduce a limited amount of automation when choosing images that are used regularly as the images that are chosen by the Histopathologist could be saved to a cache to enable quicker choice from a drop down menu in the future.
11 Appendix A - Reflections

I began my project not long after completing the January examinations after much deliberation over which project I would be most suited to given my academic attainment and the modules I enjoyed the most. I believe this was a good decision as it enabled me to gain experience within an area I was most interested and would allow me to gain the skills required for me to gain employment within this relatively new and exciting area of Computing. Furthermore, as Cloud Computing as an area of interest to me, my concentration was less likely to waiver in the latter stages of my project as my interest and enthusiasm would be constantly engaged each time I come across new material especially within the Background Research section of my report, which was very true with respect to my project.

The meeting with my supervisor occurred not long after the January exams had finished and was early enough so that I could make a well-grounded start on my Background Research section after being given indications as to what to include in my Background Research section by my supervisor. My project plan I believe was well structured and contained clear objectives as to what to achieve each week. However, I do believe that was perhaps too optimistic with the time frames for each objective as they did not take into account any problems that might occur during my training on the Cloud testbed and the initial administration time spent getting GridGain onto the Cloud testbed and fully functioning. Furthermore, I do not believe I accounted for the learning curve I encountered when getting to grips with GridGain and I believe allowing for more time to learn GridGain in more detail would have been key to overcoming this problem. This unfortunately meant I had to learn aspects of GridGain ”on the fly” without much consideration for how well I understood certain functions. However, I do believe that this would a partial advantage overall as it meant that I could focus my attention on the functions of GridGain I would make the most use of throughout my project with feeling too overloaded with all of the many different objects and attributes of GridGain. Additionally, I do not believe my project outline was of enough detail to enable me have a clear understanding as to what was expected throughout this project. Due to this I found myself trying to plan each section of my report as and when I came across it, which wasted much time during the write up of my report which I believe could have been better spent on other, more important sections of my project such as my design and implementation phase. As a lesson to be learned I would urge any final year project student to spend time after their initial supervisor meeting planning and drafting the plan for their project overall and to modularise it in such a way that each section flows to the next. Furthermore, another area I would urge all final year project students to consider is that of asking relevant and detailed questions about the projects to their supervisors; an area which I feel I let myself down considerably. I believe if I had done this I would have had a more successful design and implementation stage, which would have ultimately resulted in a
fully-functioning solution.

The Background Research section was on the whole a positive experience. As I had been given training in the resources available to look for relevant research papers I found it relatively easy to find journals relevant to the subject matter of my project. Furthermore, I was given a useful headstart by my supervisor who initially provided me with many interesting and useful papers relating to virtualisation, Cloud technologies and relevant middlewares. Furthermore, I was given useful training on how to use the Cloud testbed prior to starting my design phase which helped me to learn how to use a Cloud environment and provided me with initial ideas of how I would go about designing and constructing my solution with respect to the Cloud resources available to me. However, unplanned downtime of the Cloud testbed during my design phase did hinder my progress somewhat as I had planned each stage of my project to flow serially one after the other with the exception of some stages of my project. On hindsight, it may have been better to plan my project overall to include stages that would have benefit being done in parallel in the event of such an occurrence as this would have kept any lost project time to a minimum. Additionally, the practical, hands-on experience I gained whilst developing on the Cloud testbed was invaluable to my understanding of how Clouds operate and I would recommend that any kind of theory work undertaken be complemented by practical, hands-on work in order to enhance understanding if possible.

The design section of my report also went quite well. Initially, I struggled to come up with ideas of how to redeploy an application to a Cloud environment and had to move away from the thinking of constructing a single application to be ran on a single host. Fortunately, I came across an example of an application that had been successfully ported to a Cloud environment using GridGain with detailed images and examples of the work that had been done along with how it was done. This proved invaluable in enhancing my understanding of porting applications to a Cloud environment and gave me concrete and viable ideas on which to work with in order to design my implementation. I would recommend that any final year project student thoroughly researches examples of work that has been done previously, which relates to the subject of their project. This will help to enhance understanding and formulate ideas as to how to proceed with the design and implementation of their project. Furthermore, due to this I was in a much better position to plan my evaluation as I had a clear and detailed idea as to how my solution was to function and how it would be an improvement to the current implementation.

My implementation was initially okay. However, overall I am extremely disappointed with how this section turned out practically and also with myself. I became increasingly frustrated with not being able to get both GridGain and the current Histopathology implementation to work at the same time and spent much time trying to solve this issue. Personally, I believe I would have benefited by asking questions and seeking help from my supervisor instead of struggling and suffering in silence. Although, conversely I am happy that I at least managed to get a
solution working albeit not in a fully-functional capacity. Therefore, I advise future project students to seek help in the first instance and not get increasingly frustrated as I did.

My evaluation I feel could have benefited from more detail which would have come a better working solution and more detailed implementation. I think the time I spent trying to solve my implementation problems ultimately resulted in my having less time to spend analysing my solution in greater detail and therefore, drawing more relevant and accurate conclusions from it. In this way, I believe my evaluation section could be improved considerably and again, I would urge all final year project students to spend more time getting help from their supervisor in solving pertinent problems and therefore seeking out adequate solutions in a timely manner. Furthermore, I would urge any final year project student to consider their project wholly in terms of the design, implementation and evaluation stages and to consider making smaller sub-tasks in order to achieve better time-management and ultimately save time for resolving issues they may have in their projects.

Overall, considering the problems encountered in the previously mentioned sections of my report, I am disappointed with myself for not seeking out worthwhile help from my supervisor with whom I am confident would have done his utmost to help me and ultimately improve the overall quality of my project. However, I am incredibly grateful for this experience as I have learn a great amount about myself as a person and also about the subject area of Cloud Computing as a whole and feel that as this area of Computing is relatively new that I have contributed something worthwhile and firmly believe that Cloud Computing is the way forward for how people will interact with technology in the future; which will hopefully be as a pervasive, commodity service much in the same way as we access water, electricity and other related utilities.
package org.gridgain.examples.executor;
import org.gridgain.grid.∗;
import org.gridgain.grid.typedef.∗;
import java.util.∗;
import java.util.concurrent.∗;

public final class GridExecutorExample {
    // Constructor
    private GridExecutorExample() {
        //empty constructor
    }

    @SuppressWarnings("TooBroadScope")
    public static void main(String[] args) throws Exception {
        //Start the grid
        G.start();

        try {
            Grid grid = G.grid();
            //An object to allow command execution on the grid
            ExecutorService exec = grid.executor();
            //Used to save resulting image to disk (stored in /tmp folder)
            boolean archive = true;

            //Creates a collection ready to add subsequent commands
            Collection<Callable<GridExecutorImage>> cmds = new ArrayList<Callable<GridExecutorImage>>(5);
            //Adding subsequent commands to be executed
            for (int i = 0; i < args.length; i++) {
                cmds.add(new GridExecutorImageScaleCommand(0.5d, args[i], archive));
                Future<GridExecutorImage> futureimages = exec.submit((new GridExecutorImageScaleCommand(2.0d, args[i], archive));
                System.out.println("Received command submission for image: \" + args[i]);

        }
    }
}
// executes all commands in the list in parallel across all nodes
List<Future<GridExecutorImage>> futs = exec.invokeAll(cmds);
// A loop which loops through the invoked commands
// and retrieves the most recent resulting image
for (Future<GridExecutorImage> scaledimages : futs) {
    GridExecutorImage scaledimage = scaledimages.get();
}
for (int j=0; j<args.length; j++){
    System.out.println(
        "Received execution for image: " + args[j]);
}
System.out.println(">>>");
System.out.println(">>> Retrieved execution of all commands.");
System.out.println(">>> Check all nodes for output (this node is also part of the grid)");
// Close execution object
exec.shutdown();

} finally {
    // Stop the grid
    // G.stop(true);

}
References


