2013 Third International Conference on Intelligent System Design and Engineering Applications

CloudFTP: A Case Study of Migrating Traditional Applications to the Cloud

Liang Zhou

School of Software, Shanghai Jiao Tong University, Shanghai, 200240, China
brightzhou.cn@gmail.com

***Abstract*—The cloud computing is growing rapidly for it offers on-demand computing power and capacity. The power of cloud enables dynamic scalability of applications facing various business requirements. However, challenges arise when considering the large amount of existing applications. In this work we propose to move the traditional FTP service to the cloud. We implement FTP service on Windows Azure Platform along with the auto-scaling cloud feature. Based on this, we implement a benchmark to measure the performance of our CloudFTP. This case study illustrates the potential benefits and technical issues associated with the migration of the traditional applications to the clouds.**

Keywords- cloud computing; cloud migration; auto-scaling; dynamic scalability; Windows Azure; SaaS

I. INTRODUCTION

Recently, cloud computing[1] has been under a growing spotlight in both industrial and academic areas. Cloud computing is an on-demand and cost saving computing with scalability, high-availability, and reduced management. Amazon's Elastic Compute Cloud (EC2) is an example of IaaS (Infrastructure as a Service)[2] platform. It offers basic infrastructure component such as CPUs, memory, and storage. Google App Engine is an example of PaaS (Platform as a Service) platform. It could deploy and dynamically scale Java and Python based web applications. Based on IaaS and PaaS platforms, a lot of time and money have been saved for start-up companies, such as foursquare and dropbox.

Along with the benefits, cloud computing also raises severe concern when regarding the large amount of existing applications. One major challenge is how to migrate these traditional applications to the cloud. Current research focuses on the migration of specific applications such as high-performance applications[3], but little work has been proposed for the migration of general services.

In this paper, we present a case study moving the traditional FTP server to the cloud. We have implemented the Cloud FTP server on Windows Azure and enabled the auto-scaling feature. Based on this, we have implemented a benchmark to measure the performance of our CloudFTP. We use this case study to illustrate the potential benefits and issues associated with the migration of the traditional applications to the clouds.

The rest of this paper is organized as follows. Next section presents cloud services with Windows Azure. In Section 3, we give a short description of FTP service and details we implement it in Windows Azure. We evaluate our Cloud FTP and discuss advantages and technical issues in Section 4. Finally, we list the related work in Section 5 and conclude this paper in Section 6.

II. CLOUD SERVICES WITH AZURE

Windows Azure is the cloud computing platform offered by Microsoft. Unlike IaaS provided by Amazon and PaaS offered by Google App Engine, Windows Azure uses the mixed PaaS and IaaS strategy, this paradigm makes developing in the cloud fully flexible. Developers could choose PaaS or IaaS depending on their own needs.

When developing cloud applications, three components are mainly used: a Compute service that runs cloud applications, a Storage service providing persistent storage and a Service Bus to exchange messages in a loosely coupled way.

Hosted services in Windows Azure are said to contain roles and there are two types of roles available: a worker role and a web role. Worker roles are frequently used for long-running parallel tasks that non-interactive. Unlike the high level parallel computing framework like Hadoop[4] and Dryad[5], worker roles are not constrained in the way they communicate with each other for each work role stands alone in a virtual server. As to web roles, they are special cases of worker roles that have web access enabled by default. A web role instance responds to user requests and may include an ASP.NET web application.

For persistent storage provided by Windows Azure, there are three types: Tables, Blobs and Queues, all these types could be easily accessed through REST-based web services.

Table Storage is an example of a NoSQL[6] approach called a key/value store. Windows Azure Tables let an application store properties of various types, such as strings, integers and dates. An application can then retrieve a group of properties by providing a unique key for that group.

Blob storage is designed to store unstructured binary data. An application that stores video, for example, or backup data or other binary information can use blobs.

Unlike blobs and tables, which are used to store data, queues serve another purpose. A primary use is to allow web roles to communicate with worker roles, typically for notifications and to schedule work.

Figure 1 illustrates the typical design pattern of cloud applications on Windows Azure. The web roles are the front end to accept the user request, and then the task messages are written into the message queue and the worker roles reads from the message queue to take their jobs. Both web roles and worker roles take the storage service to store information.

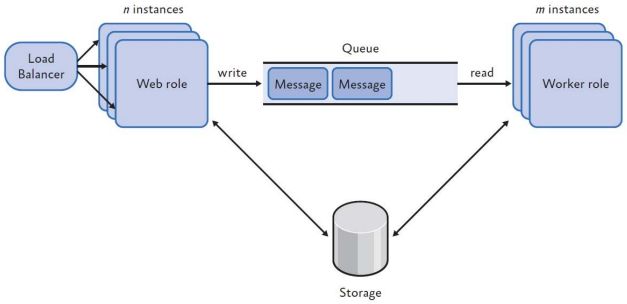

Figure 1. Typical design pattern of cloud applications on Windows Azure

III. CLOUD FTP

A. FTP

File Transfer Protocol (FTP) is a standard network protocol used to transfer files from one host to another host over a TCP-based network, such as the Internet. The protocol is specified in RFC 959 [7].

FTP is built on client server architecture and uses separate control and data connections between the client and the server. FTP may run in active or passive mode, which determines how the data connection is established:

- Active mode: In active mode, the client creates a TCP control connection to the server and sends the server the client's IP address and an arbitrary client port number, and then waits until the server initiates the data connection over TCP to that client IP address and client port number.
- Passive Mode: In passive mode, the client uses the control connection to send a PASV command to the server and then receives a server IP address and server port number from the server, which the client then uses to open a data connection from an arbitrary client port to the server IP address and server port number received. Passive mode may be used in situations where the client is behind a firewall and unable to accept incoming TCP connections.

B. System Architecture

In this section we present the system architecture of CloudFTP. CloudFTP follows the application model suggested for general Azure development as shown in Figure 2.

The users could upload and download files through ftp clients and the administrators could manage the cloud ftp through web portal. After FTP server boot up, whenever a connect request comes, there will be a slave thread spawned to handle all FTP request from the connection. We employed four components of Windows Azure to implement our CloudFTP: web role, worker role, table storage service, and blob storage service.

The web role receives user requests from web portal. The responsibility of the web role is to manage, monitor, and auto-scale. The web role is used to add, delete, update and query ftp users, and it also shows the performance data, such as CPU, memory, network connections and throughput. With this information, the web role could perform auto scaling according to the scaling rules, which is defined in table storage.

The worker role is responsible for transferring data. When an FTP client, such as FileZilla, requests a file, then the worker role sends the requested file data to the FTP client. And the number of worker role instances could be adjusted dynamically without affecting other instances. When a large number of clients attempt to connect, the number of worker roles would be increased. When the clients reduce, the number of worker roles would be decreased.

The table storage service is in charge of storing users' information, configuration information and the performance information. The users' information includes user name, the password, and the home directory. The configuration information includes permissions of users and groups. The performance information is stored in a table called performance counter, which is maintained by Azure platform. All these three types of information are suitable for table storage service, for they are all structured.

The blob storage service is responsible for storing the auto-scaling rule file and data files. The auto-scaling rule file is an xml file defining tuning strategies. For example, when the CPU utilization is greater than 80%, we could increase worker roles to distribute tasks. And the data files, which are uploaded and downloaded from users, are also stored in blob storage. The underlying file storage system is abstracted as an interface to provide basic file operations, such as store, get, mkdir, etc. The interpolated extra abstract layer indicates the FTP server can be set on any file system as long as all the operations defined in the interface are properly implemented. We will explain how we simulate the files system using blob storage in next section.

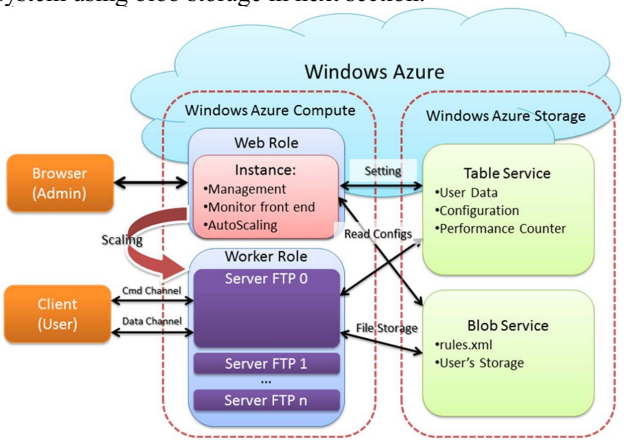

Figure 2. System architecture of Cloud FTP

C. Design of File System

In this section, we explain the design of file system in Windows Azure platform, which is the most important part of our cloud FTP. The Windows Azure Platform offers several storage services, i.e. blobs, tables, queues and SQL servers. Among these 4 storage services, we choose blob storage service as the backend of our file system, because

blob storage service is able to store very large files and the key-value style storage provide us convenient ways to implement file system interfaces, such as find, create and delete files. The design of blob service based file system is shown in Figure 3.

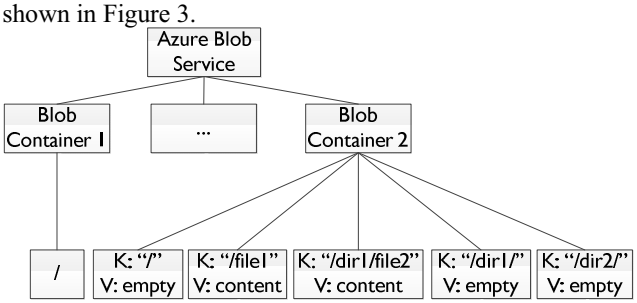

Figure 3. File System Design

In our implementation, a file system corresponds to a blob container in Windows Azure Blob Storage Service. For a file in such a file system, we use a blob to store it and the key of the blob is the absolute path of that file and the value of the blob is file content, besides, we also use a blob to store a directory. The absolute path of a directory is stored as a blob key, which ends with a "/", and the blob that stores directory has an empty value.

D. Design of FTP Server

In this section we present the design of FTP server in Windows Azure platform. We use a worker role with an infinite main loop to listen at the request port and once a client sends a connect request, the server creates a new thread to handle that request. The codes running in the new thread implement all commands in standard FTP. Many storage services can implement our VFS interface. In our project, we use the Azure Storage Service to implement it. Figure 4 depicts the design of FTP server.

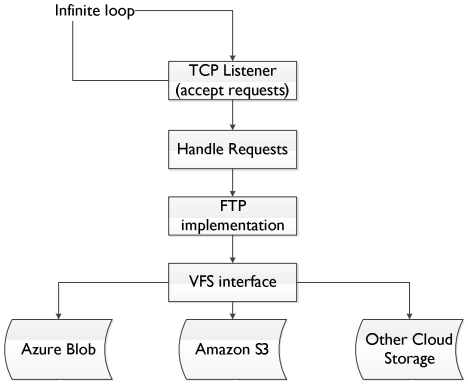

Figure 4. FTP server design

E. Passive Mode

Developing on Windows Azure Storage service is just like playing with a black box, you are not as free as developing on local operating systems. One of the noticeable difficulties is that we cannot allocate ports freely leaving a gap between the specification and implementation for passive-mode data transfer. Given the fact that we can pre-allocate endpoints on Azure, we take an alternative strategy to bridge it so that passive-mode data transfer is integrated into our Azure FTP seamlessly. We set aside a set of free endpoints as an endpoint pool. Once passive-mode is required, available port number is assigned and will be later reclaimed when connection closed. For the sake of synchronization, the procedures acquiring and releasing endpoint are monitored. Since incoming connection requests will be blocked if no endpoints available, the size of endpoint pool will be a factor having effect on concurrency of our FTP. The auto-scaling strategy will be explained in detail in the following section.

F. Auto scale

The most powerful feature of cloud platforms is scalability. However, Windows Azure doesn't enable the auto-scaling feature by default due to the accounting problem. In our CloudFTP, we use the Autoscaling Application Block, which is a library, to auto scale our FTP server. We host the Autoscaling Application Block in a web role together with the web portal. In the OnStart method of the web role, we create an auto-scaler instance and the instance would be running once the web role starts. The responsibility of the auto-scaler is to monitor the performance status and scale our FTP Server worker roles automatically according to auto-scaling rules.

There are two types of rules. One is constraint rules and the other is reactive rules.

The constraint rules define some constraint conditions. Figure 5 is an example of constraint rules, which defines the lower and upper bounds on the number of FTP server worker roles.

```
<constraintRules>
  <rule name="default" enabled="true" rank="1">
    <actions>
      <range min="m" max="n" target="AutoscaleRole"/>
    </actions>
  </rule>
</constraintRules>
```

Figure 5. Example of constraint rules

The reactive rules are used to do scale-up and scale-down. We use 4 operands to perform auto-scaling:

- CPU_Avg: The average of CPU Utilization during the last 5 minutes.
- Mem_Available_Avg: The average of available memory during the last 5 minutes.
- FTP_Connections_Avg: The average number of clients connected to the FTP server during the last 5 minutes.
- FTP_WaitClients_Avg: The average number of clients who are waiting to acquire a passive port during the last 5 minutes.

To scale up instances, we use any condition. When the CPU Utilization is greater than 80% or the percentage of available memory is less than 20%, or the number of clients waiting to acquire a passive port is greater than 20 we scale up the instances. Figure 6 depicts the example of scale up rules.

```
<rule name="scale up" rank="2" enabled="true" >
  <when>
    <any>
      <greaterOrEqual operand="CPU_Avg" than="80"/>
      <lessOrEqual operand="Mem_Available_Avg" than="20%"/>
      <greaterOrEqual operand="FTP_WaitClients_Avg" than="20"/>
    </any>
  </when>
  <actions>
    <scale target="AutoscaleRole" by="1"/>
  </actions>
</rule>
```

Figure 6. Example of scale up rules

To scale down instances, we use all condition. When the CPU Utilization is less than 20%, the percentage of available memory is greater than 80%, the number of clients waiting to acquire a passive port is less than 10 and the number of FTP client connections is less than 100, we scale down the instances. Figure 7 depicts the example of scale down rules.

```
<rule name="scale down" rank="2" enabled="true">
  <when>
    <all>
      <less operand="CPU_Avg" than="20"/>
      <greaterOrEqual operand="Mem_Available_Avg" than="20%"/>
      <less operand="FTP_Connections_Avg" than="100">
      <lessOrEqual operand="FTP_WaitClients_Avg" than="10"/>
    </all>
  </when>
  <actions>
    <scale target="AutoscaleRole" by="-1"/>
  </actions>
</rule>
```

Figure 7. Example of scale down rules

IV. Evaluation and Discussion

In this section we present the evaluation of CloudFTP and discuss the advantages and issues of migrating traditional applications to the cloud.

We implement a benchmark to evaluate the performance of our cloud FTP server. The bandwidth is 100Mbps and it is adequate for the network connections to Azure platform.

We record the download speed while increasing the number of work loaders, which continually send pwd commands to the server. We increase the number of work loaders by 10. Figure 8 depicts the download speed of different workloads. It shows that the average download speed is decreasing as the work load increases. When the number of work loaders is 180, the worker role instances increase by one. Thus the download speed arises.

We evaluate the quality of service by recording successful requests per 100 requests. Every request will send pwd, list, store, retrieve, and delete command in order. The transferring data fie is created randomly and uploaded to the server. Then it would be downloaded and MD5 checksum would be calculated to ensure the file is the same as before. Each request with correct checksum is labeled as successful request. Figure 9 depicts successful requests per 100 requests while increasing the workloads. The increasing step is also 10. It shows that the quality of service in Windows Azure platform could keep 100% approximately with the workload increases. But when the workload is up to 180, the successful requests decrease. It is because the server is performing auto-scaling and it requires a certain time to increase the instance.

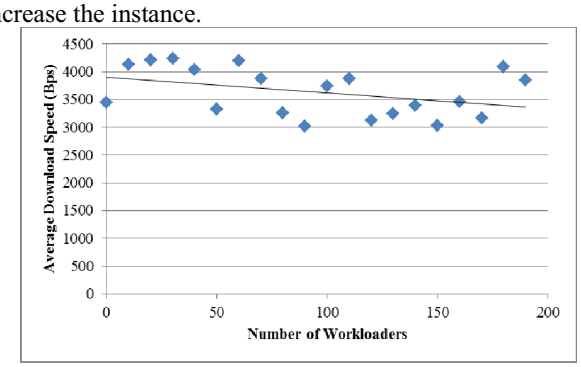

Figure 8. Download speed of different workloads

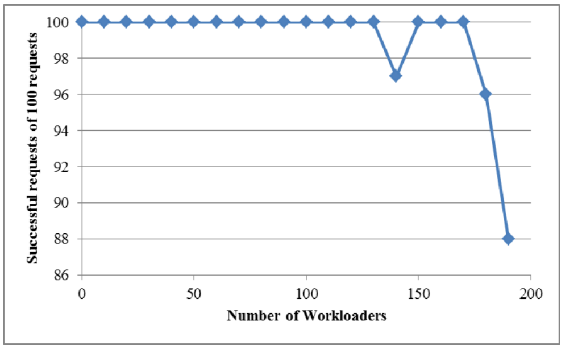

Figure 9. QoS of different workloads

From our design and evaluation results, we could see some advantages and issues migrating traditional applications to the clouds. We list the items below for they are universal problems when we choose to migrate applications to PaaS platforms.

A. Benefits

1) On-demand computing & cost-saving

Cloud computing provides users resources and services on demand. Companies could provide services without spending a great deal of money on in-house resources and technical equipment. And the cost depends on how the infrastructure is used, such as the usage time, the number of computing cores, and the storage size. In our example, we don't need to buy expensive web and storage servers, but we could provide high quality ftp services through high performance hardware.

2) High Availability

Cloud platforms could guarantee QoS for users, such as network bandwidth, CPU speed and storage availability. In our case, the file storage is the most important part of the application. The Azure platform keeps several copies of files on different servers[8]. So even if one server is crashed, other servers could still provide users the requested files.

3) Less Management

The less management means the cloud platforms could be managed by users transparently. There is no need to consider hardware maintain, load balance, and storage management.

4) Scalability

Scalability is one of the greatest benefits to cloud computing. Space, storage and RAM are quick and easy to add. There is no need to wait for quotes to be drafted and equipment to be ordered and shipped. Instead of taking days, a firm's needs can be fulfilled in a matter of hours. And in our case, we could perform auto-scale in only serveral minutes.

B. Issues

In our paper we focus on the technical issues of migration. The issues are as follows:

1) File Systems

In traditional applications, there is always a concept of file system, such as FAT32, NTFS and Ext3. But in PaaS platforms, we use key-value storage instead of file systems. When migrating traditional applications, file-related codes needs to be refactored. And in our case, we provide a good paradigm, virtual file system on the cloud platforms, to refactor the traditional FTP. We replaced every file operation with interfaces and provided the implementations on Windows Azure. In this mode, the migration could be reused on any other cloud platform, such as Google App Engine or VMware Cloud Foundry.

2) Network Issues

Traditional application could use network without limitations. But on cloud platforms, the network port must be controlled for management and security issues. We could only use allocated ports to implement our business model. In some platforms, we could only use port 80 and 443. In Azure, we face the network issues when implementing passive mode of FTP. We make the use of endpoints in Azure to achieve our goal. By pre-allocating some free ports in a pool, passive mode can be implemented. We could also use endpoints when facing protocols like DNS, DHCP, NTP, and SMTP.

V. Related Work

In this section, we discuss some previous work related to cloud migration. As mentioned in [1], cloud migration is still a new topic in research of cloud computing. Only a few papers have proposed the cloud migration. Some of these papers mentioned the cloud migration explicitly, while in others the cloud migration is implicit, but all have indicated the idea of cloud migration.

The concept of cloud computing has been proposed for about six years, a lot of works [9-12] have discussed the definition, the scope, and the advantages and challenges of cloud computing. And current works [3, 13-15] focus on scientific computing, which needs high performance calculating. The mode of cloud computing is natural for scientific calculations, but we can't ignore the existing large amount of general applications. In [16], Ali et al. presented a case study to migrate an Enterprise system to IaaS. In their work, they evaluated the cost and summarized the benefits and risks. But they didn't analyze the risks from the technical view.

In summary, though cloud migration is not a totally new idea, there is no case study to present potential advantages and technical issues on cloud migration, especially on Windows Azure platform.

VI. Conclusions

This paper presents a case study to migrate traditional applications to the cloud. We implement CloudFTP on Windows Azure along with the auto-scaling feature. We also implement a benchmark to evaluate the performance of the cloud ftp server. From the design and the evaluation results, we summarized potential benefits and risks to migrate traditional applications to the cloud. The summary could help cloud developers migrate traditional applications quickly and safely, especially on Windows Azure platform.

References

[1] M. Armbrust, A. Fox, R. Griffith *et al.*, "A view of cloud computing," *Communications of the ACM,* vol. 53, no. 4, pp. 50-58, 2010.

[2] S. Bhardwaj, L. Jain, and S. Jain, "Cloud computing: A study of infrastructure as a service (IAAS)," *International Journal of engineering and information Technology,* vol. 2, no. 1, pp. 60-63, 2010.

[3] W. Lu, J. Jackson, and R. Barga, "Azureblast: a case study of developing science applications on the cloud." pp. 413-420.

[4] T. White, *Hadoop: The definitive guide*: Yahoo Press, 2010.

[5] M. Isard, M. Budiu, Y. Yu *et al.*, "Dryad: distributed data-parallel programs from sequential building blocks," *ACM SIGOPS Operating Systems Review,* vol. 41, no. 3, pp. 59-72, 2007.

[6] M. Stonebraker, "SQL databases v. NoSQL databases," *Communications of the ACM,* vol. 53, no. 4, pp. 10-11, 2010.

[7] J. Postel, and J. Reynolds, "Rfc 959: File transfer protocol (ftp)," *InterNet Network Working Group*, 1985.

[8] B. Calder, J. Wang, A. Ogus *et al.*, "Windows Azure Storage: a highly available cloud storage service with strong consistency." pp. 143-157.

[9] R. Buyya, C. S. Yeo, S. Venugopal *et al.*, "Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility," *Future Generation computer systems,* vol. 25, no. 6, pp. 599-616, 2009.

[10] N. Leavitt, "Is cloud computing really ready for prime time?," *Computer,* vol. 42, no. 1, pp. 15-20, 2009.

[11] P. Mell, and T. Grance, "The NIST definition of cloud computing," *National Institute of Standards and Technology,* vol. 53, no. 6, pp. 50, 2009.

[12] L. M. Vaquero, L. Rodero-Merino, J. Caceres *et al.*, "A break in the clouds: towards a cloud definition," *SIGCOMM Comput. Commun. Rev.,* vol. 39, no. 1, pp. 50-55, 2008.

[13] C. Evangelinos, and C. N. Hill, "Cloud Computing for parallel Scientific HPC Applications: Feasibility of running Coupled Atmosphere-Ocean Climate Models on Amazon's EC2," *ratio,* vol. 2, no. 2.40, pp. 2.34, 2008.

[14] C. Hoffa, G. Mehta, T. Freeman *et al.*, "On the use of cloud computing for scientific workflows." pp. 640-645.

[15] S. Ostermann, A. Iosup, N. Yigitbasi *et al.*, "A performance analysis of EC2 cloud computing services for scientific computing," *Cloud Computing*, pp. 115-131, 2010.

[16] A. Khajeh-Hosseini, D. Greenwood, and I. Sommerville, "Cloud migration: A case study of migrating an enterprise it system to iaas." pp. 450-457.