Efficient Management of Geographically Distributed Big Data on Clouds

Rui Wang

MASTER RESEARCH INTERNSHIP

REPORT

Efficient Management of Geographically Distributed Big Data on Clouds

Author:
Rui WANG

Supervisors:
Alexandru COSTAN
Radu TUDORAN
Gabriel ANTONIU
Luc BOUGE
KerData, INRIA

TELECOM Bretagne

Contents

Abstract

Nowadays cloud infrastructures allow storing and processing increasing amounts of scientific data. However, most of the existing large scale data management frameworks are based on the assumption that users deploy their data-intensive applications in single data center, few of them focus on the inter data centers data flows. Managing data across geographically distributed data centers is not trivial as it involves high and variable latencies among sites which come at a high monetary cost. In this report, we introduce an uniform data management system for disseminating scientific data across geographically distributed sites. Our solution is environment-aware, as it monitors and models the global cloud infrastructure, and offers predictable data handling performances for transfer cost and time. In terms of efficiency, it leverages for applications the possibility to set the tradeoff to be done between money and time and optimizes the transfer strategy accordingly. A prototype of our system has been implemented in the Windows Azure Cloud, and we obtain some encouraging results from the extensive evaluations.

Keywords: Cloud Computing, Data Transfer, Environment-Aware, Parallel TCP streams

Acknowledgement

I would like to address my thanks to Radu Tudoran and Alexandru Costan. We have had an efficient collaboration environment for this research project: from brainstorming sessions, system architecture conception to the implementation and the performance evaluation. I would like to thank Luc Bougé for his valuable and helpful suggestions that helped me improve this report. I am also grateful to Gabriel Antoniu for his support that helped me to solve many issues. I am thankful to Shadi Ibrahim, Houssem Eddine Chihoub, Zhe Li and Yue Li for their encouragement and helpful suggestions.

1 Introduction

It is generally accepted that we live in a data explosion era. Just as the general manager of Microsoft SQL, Mr. Eron Kelly remarks [18], "In the next five years, we'll generate more data as humankind than we generated in the previous 5,000 years." With data sets continuously growing (from terabytes to petabytes) and more complex, scientists are faced with new challenges. They refer to four essential aspects: data collection from distributed sources, data management, data analysis and data publishing. Nevertheless, the traditional approaches are no longer useful to handle and process those enormous quantities of data. The scientists require new data-driven and data-intensive applications which enable them to unify massive pieces of knowledge, to discover the insight of the science phenomenon and to make decisions. These lead to a new topic in the research domain: *Big Data*.

Cloud computing is another successful paradigm which is globally sweeping across the industrial and research domains over last five years. Cloud can be seen as a possible answer to the Big Data challenges due to the following reasons: *Availability*. To the cloud consumer, the available compute and storage resources often appear to be unlimited and can be appropriated in any quantity at any time. *Accessibility*. The broad network facilitates the cloud consumers to access the cloud resources by heterogeneous thin or thick client platforms. *Elasticity*. The computational and storage resources can be elastically provisioned and released in the cloud. *Low cost*. There is no upfront cost, no hardware investments, no maintenance cost of the infrastructure for the consumers to run their applications in the cloud, they pay only what they use. Above all, cloud becomes a very important step forward in order to deal with the Big Data challenges. Scientists start to leverage the cloud resources to store, manage and process these large data sets by deploying their data-intensive applications in the cloud.

Typically, a Big Data experiment executed in a distributed environment has the following pattern: the scientists run the analysis programs at some facilities which are not located on the same site where the data are generated. There are several reasons for that [4]: Firstly, many experimental facilities, and even some supercomputers, provide only limited storage space and post-processing facilities. Secondly, raw data can be of interest to many researchers other than the individual or team who first generated it. Finally, as science becomes more complex and interdisciplinary, people are eager to combine data from multiple sources. Thus, those challenges motivate scientists to pursue efficient data movement strategies for geographically distributed Big Data.

Let us zoom on the data movement services in the cloud environment. The public cloud providers mainly adopt data center based infrastructure. All the resources are installed at the multiple geographically distributed data centers and each data center is connected via WAN to another. Most scientists mainly focus on the two kinds of data flows in the cloud: the data flow between customers and cloud via internet, the intra data center data flow that enables efficient scheduling of the compute jobs when running some applications. Few of them address the inter data centers data flows. Nevertheless, more and more scientists consider scaling up their applications to be hosted across multiple data centers because of the growing data sets and complex computing algorithms. Thus, an efficient data management service dealing with the inter data centers data flows becomes imperative. The purpose of our work is to fill this gap. There are a set of challenges that need to be taken into account:

WAN environment, data center resource capacity, user-perceived latency and the cost, etc. All of them will be detailed in the following section.

During this internship, we introduce a cloud-based data transfer system that supports large-scale data dissemination across geographically distributed data centers. The key idea of our solution is to accurately and robustly predict I/O and transfer performance in a dynamic cloud environment in order to judiciously decide how to perform transfer optimizations over federated data centers. Estimates must be dynamically updated to reflect changing workloads, varying network-device conditions and configurations due to multi-tenancy. Based on this monitoring feedbacks, we design performance models for the cloud resources. They are further leveraged to predict the best combination of protocol and transfer parameters (e.g., flow count, multicast enhancement, replication degree) in order to maximize throughput or minimize costs, according to application requirements. To validate our approach, we have implemented the above system as part of the Azure Cloud [26] so that applications could use it following a Software as a Service approach.

The rest of this report is organized as follows: In Section 2, we describe the background, including problem statement and some related work. Section 3 presents the motivations and the overview of our contributions. We address the functional descriptions of each components and implementations in Section 4. An extensive evaluation with synthetic benchmarks will be detailed in Section 5. Finally, in Section 6, we summarize the lesson learnt from this internship and the future work.

2 Background

In this section, we explain the geographical data management issues and the background of our work. We present the cloud computing paradigm at first, including its origin and different logical models. Then we address the general data management and data movement issues. By bringing geographically distributed data in the cloud, we detail the problem statement so that the readers could better understand our motivations and challenges. The last part refers to the related work. To sum up, this section generally states that there is no appropriate solution which could handle efficiently the geographically distributed data in the cloud.

2.1 Cloud Computing

Along with the growing demands of computational powers and storages, the new computing systems are required to tackle those capacity barriers. The parallelization of computation on a large amount of machines provides a possible solution. From the Grid Computing to the Peer-to-Peer (P2P) architectures, scientists and researchers proposed number of frameworks to enable machines to cooperate and work in parallel. Over the last decade, the commercial software providers have started to use their available resources in a new way. They provide the facilities available in the data centers as a commercial services, allowing outside customers to rent their available computational and storage resources on-demand as the electricity or tap water is. This is the origin of the cloud computing paradigm. Even though many people use cloud computing, few of them agree on a specific definition. The definition from National Institute of Standards and Technology [13] is one of the most accurate in our opinion:

> Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction.

As shown in Figure 1, cloud computing providers offer their services based on several fundamental models from low level to high level abstractions: Infrastructure as a Service (IaaS), Software as a Service (SaaS), Platform as a Service (PaaS).

Infrastructure as a Service It is the basic cloud model where the cloud providers offer the services as the virtual machine instances. The customers install their operating system images (The OS images can be also provided by the cloud) on the virtual machines to deploy the applications. The customers rent and handle the resources including virtual machines, servers, storages, networks on infrastructure basis. Amazon EC2 [17], Nimbus [22], OpenNebula [23] are some of the examples of the IaaS level providers.

Platform as a Service It delivers customers a platform including programming language execution environment, database, and web server, etc. For the developers who work at the

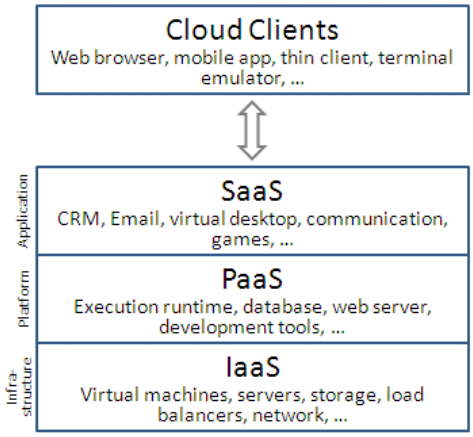

Figure 1: Cloud computing layers [25].

PaaS level, cloud assumes that you write and execute your program in the cloud platform without configuring the program environment or managing any low layer hardware. Here we give some examples of the PaaS providers: Google App Engine [20], Windows Azure, etc.

Software as a Service The SaaS providers are in charge of installing, operating, maintaining and updating the software in the cloud, which frees the customers from managing the software on their own computer. This model offers the flexibility and reliability for the customers to access to the applications from any place at any time, only with one computer and the internet. Some examples include: SalesForce [24], Dropbox [19], etc.

2.2 Data Movement

When enormous quantities of data are being continuously produced today, many scientists raise a critical question: how can researchers efficiently and reliably access these data to do their scientific discovery? This question motivated many research topics about the data movement. In this subsection, we give a brief introduction about the general data movement issues and basic knowledge.

General Approach One general approach for solving the data transmission issue is to partition the complicate communication system into different logic layers based on the OSI model. The OSI model defines a networking framework for implementing protocols. Each layer is functionally independent of the others and typically serves the layer above it. Because of this model, the scientists or engineers concentrate themselves on specific problems at one layer. Here, we only introduce some basic protocols regarding data transfers at two layers: the transport layer protocols and the application layer protocols.

The transport layer provides end-to-end communication services for the applications, draws our attention for understanding the data movement issues. The existing transport layer protocols can be classified into two categories: the connection-oriented and connec-

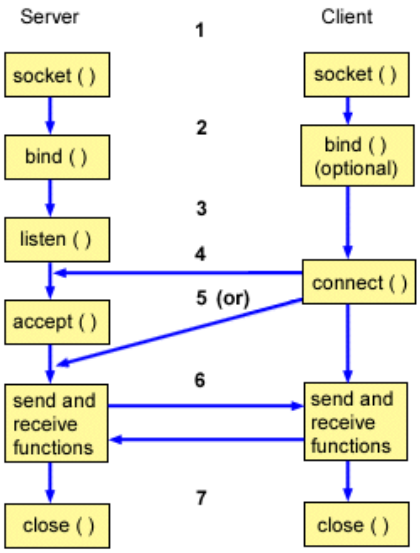

Figure 2: Network sockets event flow [25].

tionless. Transmission Control Protocol (TCP) provides a connection-oriented, reliable, ordered delivery of the byte streams between the processes on different computers. Connections must be properly set up in a three-step handshake process before data transfer. More complex mechanisms are required: congestion control, error checking, etc. In contract, User Datagram Protocol (UDP) is connectionless. With minimum control mechanism, UDP provides a simple, fast, message-based data communication which is an appropriate option for the real time stream systems instead of data transmission.

The application layer is reserved for process-to-process communications across an Internet Protocol computer network. File Transfer Protocol (FTP) and Hypertext Transfer Protocol (HTTP) are two of the widely used fundamental application layer protocols for the data transfer. FTP is the standard application protocol to transfer the files to another host over TCP based network. FTP is designed based on the client-server architecture. HTTP aims to achieve the data transfer specifically for the World Wide Web based on the request-response model.

Network Sockets Network sockets [25] define a general abstraction of the inter process communication flow across the computer network. A socket is characterized by a unique combination of the IP address, the port number and the transport layer protocol. The mechanism ensures that sockets deliver the data packets to the destination machine on the appropriate application process. The Figure 2 describes the typical events flow for a connection-oriented (server and client) socket session.

The endpoint is created via the socket API for the potential communication and data transmission. Application can bind a unique name to the socket. Servers must bind a name in order to be accessible. Then the server activates the listen API to indicate the willingness of accepting the connection requests from the client. After initializing the connect API, the server application issues the accept API. Now the connection is established between stream

sockets and server or client can use the data transfer APIs to communicate. Once any one issues a close command, the connection will be stopped and the system resources will be released.

Performance Metrics for the Data Movement It is important to have some performance metrics to measure the data movement. *Round-trip time* (RTT) refers to the length of time that a data packet to be sent into the network plus the length of time that it takes for an acknowledgement of that packet to be received. In another words, RTT is the time delay of transmission times between two nodes in the network. In the IP network, we usually use Internet Control Message Protocol (ICMP) echo request to obtain the RTT between the hosts. In the computer science, the *bandwidth* is a measurement of the bit-rate for the available resources for the data communications. In other words, it indicates the capacity of the communication links. It can be expressed in bits per second. *Throughput* aims at measuring the average rate of successful message delivery over a communication channel. Bandwidth could be considered as the theoretical maximum throughput of the system, and the throughput essentially means the bandwidth consumption.

Performance Improvement for the Data Transfer One of the approaches to improve the data transfer throughput is by exploiting the network and the end-system *parallelism* or by using a hybrid approach between them. Building on the network parallelism, the transfer performance can be enhanced by routing data via intermediate nodes chosen to increase aggregate bandwidth. Multi-hop path splitting solutions [8] replace a direct TCP connection between the source and destination by a multi-hop chain through some intermediate nodes. Multi-pathing [14] employs multiple independent routes to simultaneously transfer disjoint chunks of a file to its destination. These solutions come at some costs: under heavy load, per-packet latency may increase due to timeouts while more memory is needed for the receive buffers. On the other hand, endsystem parallelism can be exploited to improve utilization of a single path. This can be achieved by means of parallel streams [6] or concurrent transfers [12]. Although using parallelism may improve throughput in certain cases, one should also consider system configuration since specific local constraints (e.g., low disk I/O speeds or over-tasked CPUs) may introduce bottlenecks. Another alternative approache emerged in the context of the BitTorrent based which can aggregate the available bandwidth for transfering the data.

2.3 Bridging Data in the Cloud

As we discovered, most of the existing public cloud providers widely accept the one data center basis for user deployments: the cloud customers are free to select any data center to host their applications, whereas, the deployments are limited to locate in only one data center. There is no seamless /automatic mechanism for scaling their hosted services across multiple geographically distributed data centers [2]. Furthermore, most of the current distributed computing frameworks have been designed with the restriction that all the compute nodes are co-located [5]. In consequence, most of the computer scientists currently focus on the optimization for the intra data center data flows, but few of them address the inter data centers data flows.

As mentioned in the introduction, this single data center basis deployment assumption becomes inadequate in the big data era. Lots of scientists consider scaling up their applications across multiple data centers due to the following facts:

1. In some domains, the commercial or scientific data are typically collected from several geographically distributed sources, nevertheless, people or scientists need to process them with the computational resources hosted in different data centers.

2. For some large scale applications or experiments which require more than available compute resources in one data center (Public cloud providers restricts maximum number of computational virtual machines for users in one data center), scientists need to integrate more resources working in parallel across multiple locations.

It makes sense to manage those amount of data using several data centers. Hence, one needs a system that achieves the efficient data movements in the cloud and considers the following challenges:

WAN Environment Local Area Network (LAN) is designed for interconnecting the computers in a very limited area, for example, the network within one data center. Ethernet is the widely used technology for LAN which provides high transfer rate (100 Gigabit Ethernet), and relatively stable network environment. In contrast, Wide Area Network (WAN) is the data communications network that covers a relatively broad geographic area. The data center is connected via the WAN to each other, and transfer throughput between data centers varies with time. On one hand, in the spatial dimension, the long transmission distance makes the WAN to be more complex and unstable (e.g., different routing path, different technology, different infrastructure condition, etc.). On the other hand, in the time dimension, the transfer bandwidth also depends highly on the peak demand from the geographic area covered by the site and highly correlated with the local time [10].

User-Perceived Latency Large data sets are intuitively stored and processed in the data center which is close to the user's location, so that the user-perceived latency could be reduced. Nevertheless, the decision is not as simple when running large scale data-intensive applications in the cloud. In the distributed computing frameworks, the scientists need to make the wise choice between moving the computations close to the data or moving the data sets close to the computations.

Monetary Cost One of the key features of the cloud paradigm is its pay-as-you-use philosophy. In clouds, the users need to pay for the data transfers between data centers. Moreover, the CPU consumption is also measured in hours of the virtual machines. More resources might reduce the transferring delays for large data sets (e.g., parallel TCP streams), whereas, in this case the monetary cost might be much more higher. Hence, an optimized model is essentially required to make a tradeoff between those factors (delay or cost) for the data movement service in the cloud.

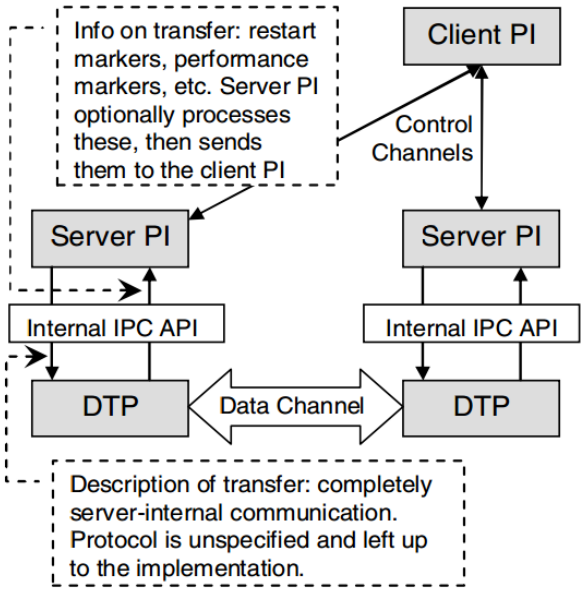

Figure 3: Globus GridFTP architecture [1].

Programming Model The existing public cloud providers do not offer the seamless/ automatic mechanisms for scaling their hosted services across multiple geographically distributed data centers. A new programming model needs to be considered to deal with the data scheduling and data processing issues across multiple data centers.

2.4 State of the Art

In this subsection, we present two complete solutions for the data movement in the distributed environment: Globus and Stork.

Globus [1] is a GridFTP based framework initially targeting the grid environment, which aims at providing secure, reliable, high-performance data movement services. In addition, Globus also released the SaaS based version, Globus Online, which adapts to the cloud environment. Figure 3 illustrates the basic Globus architecture design. The implementation is mainly composed of the three logically distinct components, client and server protocol interpreters (PIs), which are designed for managing the control channel protocol, and the data transfer process (DTP), which is responsible for handling the data access and data movement via the data channel protocol. Globus is one of the high-performance implementation of the GridFTP protocol which supports the following features: third-party control of data transfer; authentication, data integrity, data confidentiality; striped data transfer; parallel data transfer; partial file transfer; automatic negotiation of TCP buffer/window size; support for reliable and restartable data transfer.

Nevertheless, Globus is not the appropriate solution for the problems which we address. For example, Globus only performs data transfers between GridFTP instances (between GridFTP client and server), remains unaware of the environment and therefore its

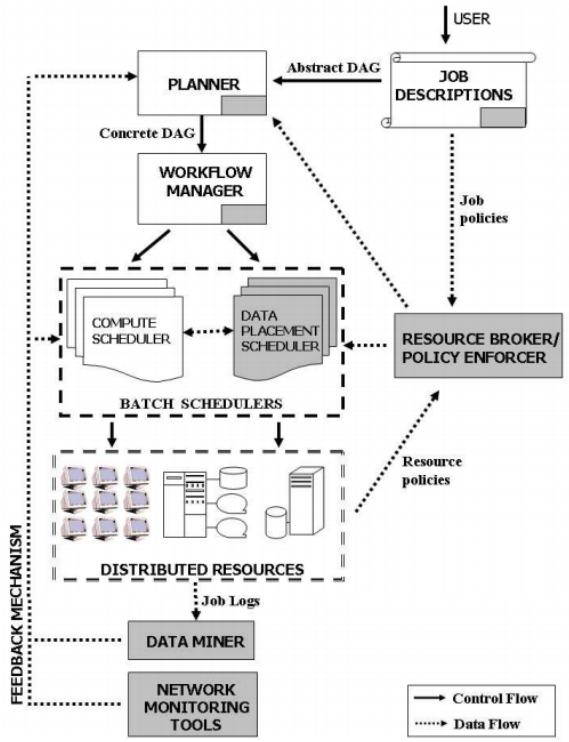

Figure 4: Components of the Data Placement Subsystem(Stork) [9].

transfer optimizations are mostly done statically. Furthermore, it lacks the mechanism to automatically scale up the compute instances for transfers in the cloud, and it did not take the monetary cost into consideration, which is one of the important features in the public cloud environment.

Stork [9] provides a framework for efficient data placement in distributed computing systems. The data movement can be regarded as amount of jobs, as important as the computational jobs, nevertheless, data placement jobs need to be treated in the different way. The key feature of this framework is its specialized scheduler (Stork) for the data movement jobs, which provides the following characteristics: interaction with higher level planners; interaction with heterogeneous resources; flexible job representation and multilevel policy support; dynamic protocol selection; run-time protocol auto-tuning; failure recovery and efficient resource utilization.

Figure 4 represents the architecture of the Stork data placement subsystem: the high level planner and workflow manager aim to submit the computational jobs and data placement jobs to the execution queue accordingly. The main component, data placement scheduler, provides an uniform interface for different protocols and storage services to perform as a I/O control system and I/O schedule. It collects information from both the workflow manager and the resource broker/policy enforcer to make the decision for employing the distributed resources to move the data. Moreover, some feedback mechanisms facilitate the scheduler to tune the job scheduling.

Stork provides an efficient data movement framework in distributed systems. It treats

the data placement jobs as important as the computational jobs. The scheduler is able to make the wise decisions according to the job descriptions and existing resources, and it can also adapt to the environment. Nevertheless, this framework can not adapt to the cloud environment based on the data center infrastructure, where the network environment intra and inter data center are different. In addition, it also lacks the models for the adaptive behaviour.

3 Our contribution: An Environment-Aware System for Geographically Distributed Data Transfers

In this section, we introduce our contribution illustrated in the context of a real scientific application, called *A-Brain*. Then we review the design principles of our solution. The overview of the architecture will be presented briefly in this section. The last part introduces our models for the cloud environment.

3.1 Motivation

A-Brain [16] is a project developed in the context of the INRIA-Microsoft Joint Center, aiming at the analysis of neuroimaging and genetic data on large cohorts of subjects. This is a new approach used to assess and understand the variability that exists between individuals, and that has remained poorly understood so far. As both neuroimaging and genetic-domain observations represent a huge amount of variables (of the order of 10^6), performing statistically rigorous analyses on such amounts of data represents a computational challenge that cannot be addressed with conventional computational techniques. On one hand, sophisticated regression techniques need to be used in order to perform sensitive analysis on these large datasets; on the other hand, the cost entailed by parameter optimization and statistical validation procedure (e.g., permutation tests). However, the computational framework can easily by run in parallel.

Taking advantage of the strong experience in the data management for the data-intensive application, KerData team will leverage it on the Microsoft Azure Cloud, and provides a very efficient approach to achieve the experiment computations. The computational model is based on the MapReduce.

The initial experiments from this project stress that for the Azure Cloud, there is a limitation for the maximum number of computational nodes within each data center. The cloud user is strictly limited to use at most 300 cores per data center. Furthermore, there is no existing mechanism to federate the resources across multiple data centers so that we can deploy the applications in several locations and make them work together. On the other hand, some complex computations require more than 1000 cores to cooperate and to work together. This issue motivates the topic of our internship.

3.2 Design Principles

To enable efficient geographically distributed data transfers, we rely on the following ideas:

Environment awareness Cloud infrastructures are exploited using a multi-tenancy model, which leads to variations in the delivered performance of the nodes and the communication links. Monitoring and detecting in real-time such changes is a critical requirement for scientific applications which need prediction performance. The monitoring information can then be fed into higher-level management tools for advanced provisioning and transfer purposes over wide-area networks. This helps removing the performance bottlenecks one-by-one and increase the end-to-end data transfer throughput.

Modeling the cloud performance The complexity of the data centers architecture, topology and network infrastructure make simplistic approaches for dealing with transfer performance (e.g., exploiting system parallelism) less appealing. In a virtualized environment such techniques are at odds with the goal of reducing costs through efficient resource utilization. Accurate performance models are then needed, leveraging the on-line observations of the cloud behaviour. Our goal is to monitor the virtualized infrastructure and the underlying networks and predict performance metrics (e.g., transfer time, costs). As such, we argue for a model that provides sufficiently accuracy for automating the distributed data management tasks.

Cost effectiveness As expected, the cost closely follows performance. Different transfer plans of the same data may result in significantly different costs. In this paper we ask the question: given the clouds interconnect offerings, how can an application use them in a way that strikes the right balance between cost and performance.

No modification of the cloud middleware and loose coupling Data processing in public clouds is done at user level, which restricts the permissions to the virtualized space. Our solution is suitable for both public and private clouds, as no additional privileges are required. Moreover, stand alone components of the data management system increase the fault tolerance and allow an easy deployment across geographically distributed data centers.

3.3 Architecture Overview

The proposed system relies on three components, called agents, including Monitor Agent (MA), Transfer Agent (TA) and Decision Manager (DM) that provide the following services: monitoring, data transfers and decision management. The TA is designed to open multiple TCP connections between the virtual machine instances and transfer the data through those multiple paths. The MA is responsible for tracking the initial cloud environment. The DM implements the cloud models. It plays a role of interaction between MA and TA. On one hand, it conducts the predictable models by collecting the monitor logs from MA. On another hand, it schedules the data movement jobs to the TA in the VM pool based on the model estimators and also continuously tracks the cloud performance from TA. These modules (depicted in Figure 5) are replicated within the Virtual Machines (VMs) of the data centers where the applications are running. The system is self-configurable: the discovery of its other peers is done automatically using the user credentials to query the cloud data centers. The scientific applications simply interact with the data management service using its extended API, in order to perform wide-area data transfers. In Section 4, we will give the detailed functional description of each architectural units.

3.4 Cloud Environment Awareness Modeling

Multiple factors like multi-tenancy, wide-area-networks or the commodity hardware contribute to the variation of the performance in a cloud environment [4]. Figure 6 presents a snapshot of the inter-datacenter throughput in the Azure cloud, as tracked by the Monitoring Agent. There are two options for modeling such complex infrastructures. Analytical models predict the performance of the underlying resources using low-level details about

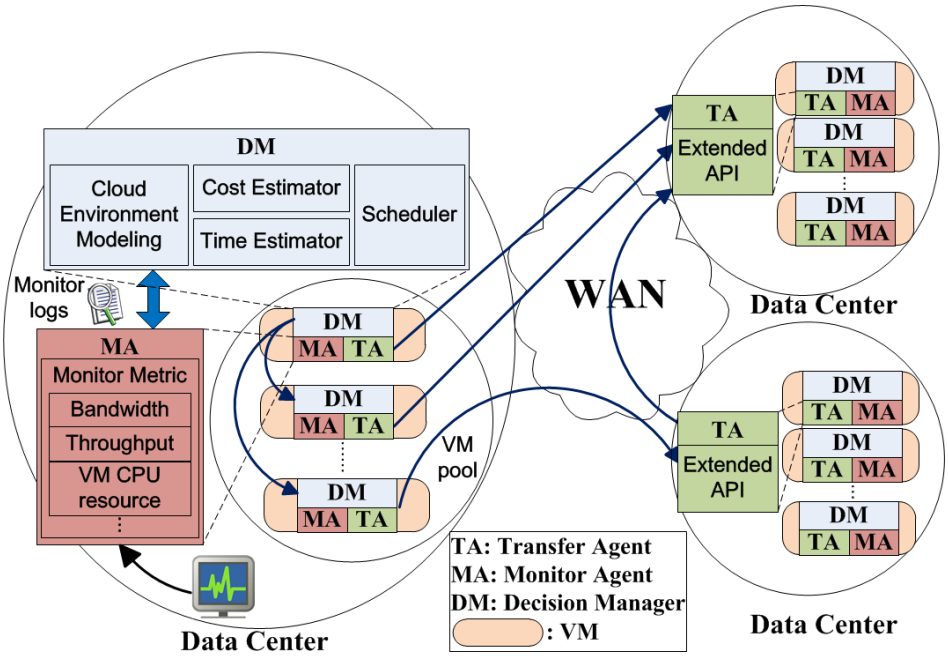

Figure 5: Architectural overview of the geographical distributed data management system (GEO-DMS).

their internals alongside workload characterizations. Although less expensive and faster than empirical models, they rely on simplified assumptions and require complex details for better modeling. Sampling methods perform active measurements of the targeted resources and do not require understanding the increasingly-complex, and often transparent, cloud internals. Our technique falls in the empirical, sample-based category. In this section we describe it and then show how to use it for predicting the transfers cost/completion-time efficiency.

3.4.1 Cloud Data Transfer Model

The monitoring information used for performance estimates is collected in two phases: an initial learning phase, at deployment start-up; and a continuous monitoring phase during the lifetime of the deployment, in which new measurements are done at configurable time intervals. Several user-defined parameters control the update frequencies in order to remain non-intrusive.

We model the cloud performance based on the accumulated trace of the monitoring parameters about the environment (h - history). The cloud average performance (μ - Equation 1) and variability (σ - Equation 2) are estimated at each moment i. These estimations are updated based on the weights (w) given to each new measured sample.

$$\mu_i = \frac{(h-1) * \mu_{i-1} + (1-w) * \mu_{i-1} + w * S}{h} \tag{1}$$

$$\sigma_i = \sqrt{\gamma - \mu_i^2} \tag{2}$$

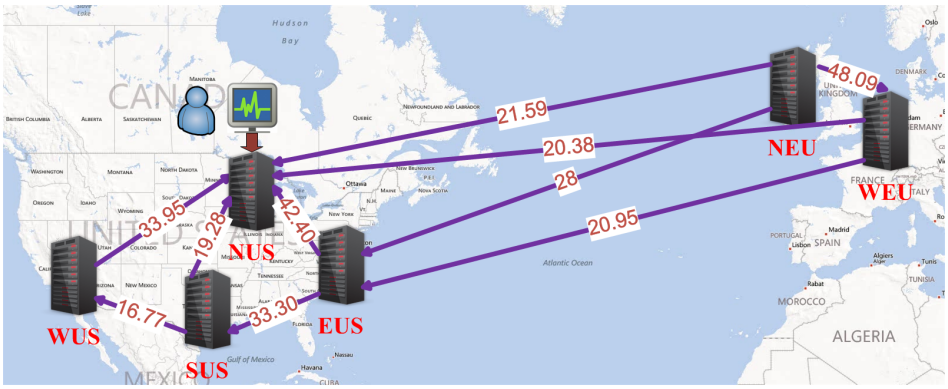

Figure 6: A snapshot of the average inter data centers throughput (in MB/s) map generated by the Monitoring Agent for the Azure cloud

$$\gamma_i = \frac{(h-1) * \gamma_{i-1} + w * \gamma_{i-1} + (1-w) * S^2}{h} \tag{3}$$

where S is the value of the sample to be integrated and γ (Equation 3) is an internal parameter used for iteratively updating the variability (σ).

Our approach weights each sample individually. For instance, an outlier value in a stable environment is most probably a temporary glitch and should not be trusted. To select these weights, we consider the following observations:

- A high standard deviation will favor accepting new samples (even the ones farther from the average);
- A sample far from the average can be an outlier and it is weighted less;
- The time interval between the samples - less frequent samples are weighted higher (as they are rare and thus more valuable).

The weighting formula (Equation 4) then combines the Gaussian distribution with a time reference component:

$$w = \frac{e^{-\frac{(\mu-S)^2}{2\sigma^2}} + (1 - \frac{tf}{T})}{2} \tag{4}$$

where tf is the time frequency of the sample and T is the time reference interval. This normalizes the weights with values from the interval $(0, 1)$: 0 - no trust and 1 - full trust.

3.4.2 Efficiency in the Context of Data Management

Efficiency can have multiple declinations depending on application context and user requirements. In clouds, the most criticals among these are the transfer time and the monetary cost. Our model estimates these metrics with respect to each transfer setting (used resources used, geographical location, etc.).

The Transfer Time (Tt) is estimated considering the number of nodes (n) that are used in parallel to stream data and the predicted transfer throughput (thr_{model}), obtained from monitoring the respective cloud link:

$$Tt = \frac{Size}{thr_{model}} * \frac{1}{1 + (n-1) * gain} \tag{5}$$

where $gain$ represents the improvement obtained from the parallel transfers (determined empirically and with values less than 1).

The Cost of a geographical transfer is split into 3 components. The first corresponds to the cost charged by the cloud provider for outbound data ($outbound_{Cost}$), as usually inbound data is free. The other two components are derived from the cost of the VMs (n - number of VMs) that are leased: the network bandwidth (VMC_{Band}) and the CPU (VMC_{CPU}) costs. The ratio of the resources used for the transfer is given by the intrusiveness parameter ($Intr$). The final cost equation is:

$$Cost = n * (Tt * VMC_{CPU} * Intr + \frac{\frac{\frac{Size}{n}}{Tt} * Intr}{VMC_{Band}}) + outbound_{Cost} * Size \tag{6}$$

where for simplicity we considered that each of the n nodes sends the same amount of data ($\frac{Size}{n}$).

This model captures the correlation between performance (time) and cost (money) and is able to adjust the tradeoff between them dynamically during transfers. An example of such a tradeoff is setting a maximum cost for a data transfer, based on which our system is able to infer the amount of resources to use. Although the network or end-system performance can drop, the system rapidly detects the new reality and adapts to it in order to satisfy the budget constraint. In fact, we evaluate the *returned benefit* that the resources bring; applications can leverage this knowledge to select their optimal expense.

4 Functional Description and Implementation

In the first part of this Section, we zoom on the each component of our system, to give the detailed functional descriptions. The second part aims to present how we implemented and validated our solution in a public cloud platform: Windows Azure Platform.

4.1 Functional Description

The Monitoring Agent (MA) It has the role of monitoring the cloud environment and reporting the measurements to a decision manager. Using the tracked data, a real-time online map of the cloud network and resource status is continuously constructed and made available to applications. The metrics considered are: available bandwidth, throughput, CPU load, I/O speed and memory status. New metrics can be easily defined and integrated using our pluggable monitoring modules approach. This component further records the monitoring history. Such a functionality is important from two perspectives: on one hand, the tracked logs are used by the scientists to better understand and profile their cloud based applications, and on the other hand, this provides the base functionality for a self-healing system.

The Transfer Agent (TA) It performs the data transfers and exposes a set of functions used to exploit the network parallelism (e.g. direct_send, forward_sending, read, split etc.). These functions are used internally by the decision manager to coordinate data movements, but are also made available to users that might want to integrate the data management in the application logic. Additional transfer optimizations include: data fragmentation and recomposition using chunks of variable sizes, hashing, acknowledgement for avoiding data loss and duplications. One might consider the latter confirmation mechanism redundant at application level, as similar functionality is provided by the underlying TCP protocol. We argue that this can be used to efficiently handle and recover from possible cloud nodes failures, when intermediate nodes are used for transfers. Finally, the component also monitors the ongoing transfers and provides real-time information about the achieved throughput and the progress to completion.

The Decision Manager (DM) It coordinates the transfer from the source(s) to the destination(s), using possibly intermediate nodes, which are used as complementary links with the destination, through which data is forwarded. The applications initialize the data movements by specifying the transfer parameters (e.g., destination, completion time - cost tradeoff or absolute values) to the Decision Manager. Based on these parameters and on the cloud status, it chooses the appropriate resources and paths to perform the transfers, so that they satisfy the efficiency constrains. The choices are made by estimating the delivered performance and its cost as described in Section 3. This selection is updated at specific intervals in order to reflect the inherent cloud variability. The Decision Manager sees the network of Transfer Agents similar to a global peer-to-peer network, which is used to coordinate data flows towards the destination. Although a Decision Manager exists on all nodes for availability reasons, each transfer is handled by a single entity, typically the one contacted by the application to initialize the transfer.

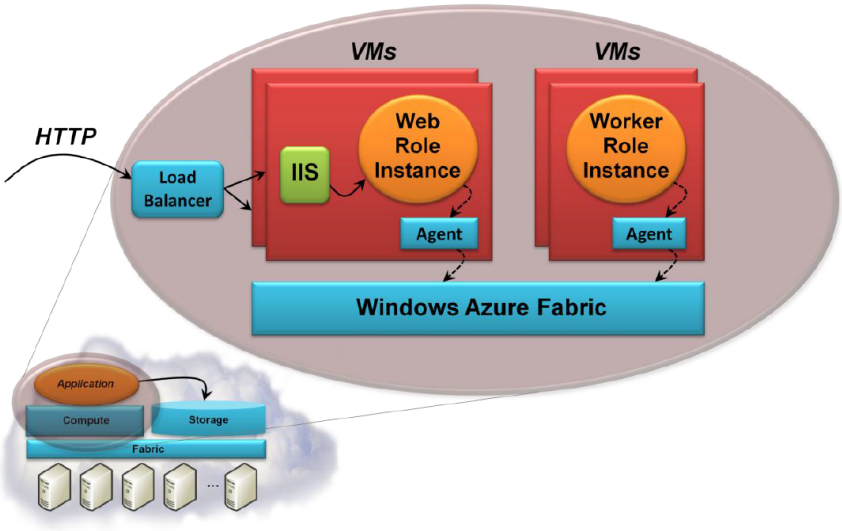

Figure 7: The Windows Azure computational resources [26].

4.2 Implementation

A prototype was built to validate the concepts explained aforementioned. The system was implemented in C# and currently runs on the Microsoft Azure cloud platform. The backbone of the system is a layered class hierarchy with approximately 50 classes, structured in the three main independent modules according to the their different logical functions. This hierarchy is organized in a mostly single-rooted class library (i.e., the grouped classes inherit from a common class) following the abstract factory design pattern. The Monitoring Agent and the Transfer Agent define an application programming interface (API) which specifies the interactions with the Decision Agent.

4.2.1 Windows Azure

Azure Platform, is the Microsoft product for the cloud. Windows Azure provides users with on-demand and scalable computational and storage resources through the geographically distributed data centers in the world. An aspect of the Windows Azure solution is the shift from the PaaS level provider to the IaaS level provider. It allows the users not only to execute and debug their native code by using a specific technology at the distant runtime environment, but also to create the VMs with different operation systems (Windows-based or Linux-based) by specifying their own images. Thus, Windows Azure facilitate the developers to handle the available resources in the Cloud with high flexibility and scalability. In this report, we mainly focus on the PaaS level services. We will address some key features of the Windows Azure in this subsection.

Computational Resources The Windows Azure offers the computational resources generally based on the abstraction of the virtual machines. Windows Azure provides two kinds of virtual machine instances on the PaaS level: Web Roles and Worker Roles. Web Roles, are

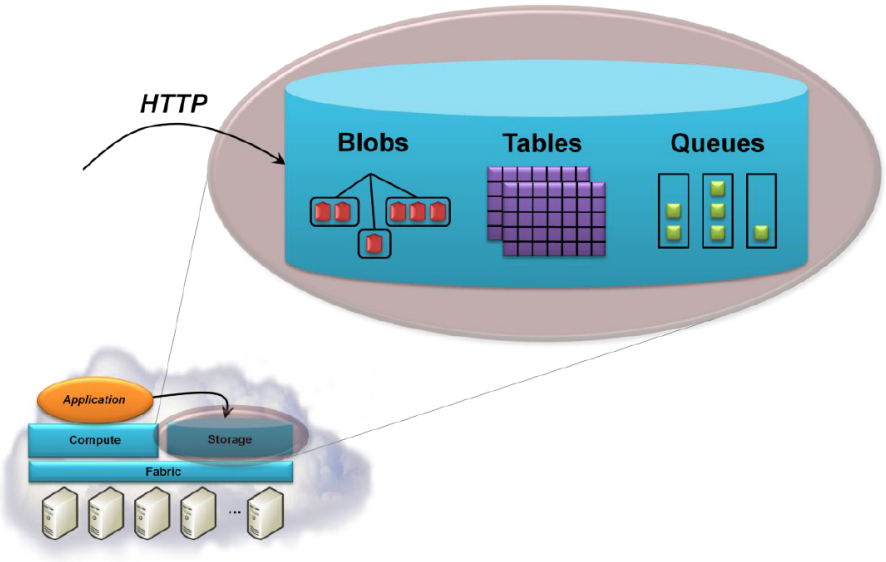

Figure 8: The Windows Azure storage resources [26].

generally pre-configured with Internet Information Services (IIS) and specifically designed to handle the Web applications by interacting with outside HTTP requests. Worker Roles, aim at running the background processes. They facilitate the developers to execute the native code in order to handle the computations as the PaaS level, as depicted in Figure 7. The applications are deployed on one or several Worker Role instances. Worker Roles have the ability to communicate between them via the IP endpoints. Furthermore, the scale of the Worker Roles could be adjusted dynamically which enables developers to handle the computational resources with flexibility. Windows Azure provides also a default load balancer in order to balance the loads between the different Role instances.

Storage Resources Besides those, Windows Azure provides some persistent storage services including Blobs,Tables and Queues, which are designed to manage the data in the different structures as depicted in Figure 8. All of them are accessed via a RESTful HTTP interface. Azure Blobs are the key-value pair storage system which is designed to store large scale unstructured binary data. They are widely used for storing the data, files, video contents, etc. for the running applications. The size of a blob is bounded to 1TBytes (page blob). The blobs are stored on multiple distributed nodes in the data center. The Windows Azure provides some load balancing mechanisms to ensure a strong consistency. Azure Tables is another important key-value pair based storage system which are similar to the NoSQL approach. This technology does not support the relational store, nor complex SQL based queries. Instead, Azure Tables offer a fine grain, fast and very scalable access to typed data. Azure Queues provides a reliable asynchronous message delivery schema between Roles. Azure Queues are only designed for passing small size messages (with maximal size of 64 KB). They do not respect a strictly FIFO rule when processing the messages. Instead, any dequeued message is not deleted, it remains valid a certain amount of time. If we do not carry out the delete operation for this message, this dequeued message will appear again and be processed by other Role. This mechanism guarantees a reliable and fast message passing

Table 1: Compute instance price details [26]

Instance	CPU Cores	Memory	Disk	I/O Perf.	Cost Per Hour
Extra Small	shared	768MB	20GB	Low	$ 0.02
Small	1 x 1.6 GHz	1.75GB	225GB	Moderate	$ 0.12
Medium	2 x 1.6 GHz	3.5GB	490GB	High	$ 0.24
Large	4 x 1.6 GHz	7GB	1000GB	High	$ 0.48
Extra Large	8 x 1.6 GHz	2040B	20GB	High	$ 0.96

communication.

Prices Another key feature of the Cloud Computing is its pay-as-you-use philosophy. The Azure users are charged only when their applications are running in the Cloud or when their data are stored in the Azure storage systems. Here we only address two of the pricing policies which are related to our work: role instances and outbound data transfer. The users pay for compute role instances on a per-hour usage basis when they deploy their application code in the cloud. There are five different size of compute instance sizes to choose for running a vast array of applications, as we depicted in the Table 1. Data transfers between Windows Azure services located within the same data center are not subject to any charge. Instead, outbound data is charged based on the total amount of data moving out of the Windows Azure data centers via the Internet. The first 5GB of outbound data transfer is free, extra transfers are charged 0.12$/GB.

4.2.2 Our System

The Monitoring Agent It is designed as an ensemble of self-describing subsystems which are registered as dynamic services, and are able to collaborate and cooperate in performing a wide range of information gathering tasks. As Figure 9 depicted, The Monitoring Agent consists of several subcomponents expressed as the classes. The Monitoring Agent tracks several metrics which can be defined in an abstract class : Metric. By inheriting from this metric class we currently implement some concrete metric contents for measuring the performances: the available bandwidth between the nodes and between the data centers using the iperf software [21]; the throughput, computed by measuring the transfer time of random transfers of data based on the TCP sockets, etc. Those pluggable modules used for collecting different sets of information, or interfacing with other monitoring tools, are dynamically loaded and executed in independent threads. In order to minimize intrusiveness on host systems, a dynamic pool of threads is created once, and the threads are then reused when a task assigned to a thread is completed. This allows one to run concurrently and independently a large number of monitoring tasks, and to dynamically adapt to the load and the response time of the components in the system. We have also set a customizable intrusiveness threshold, which limits the monitoring samples frequency when the VM is highly loaded. This option is used for example by the Decision Manager to suspend the throughput measurements during data transfers, as this information can be collected directly from the Transfer Agent thanks to the command pattern design.

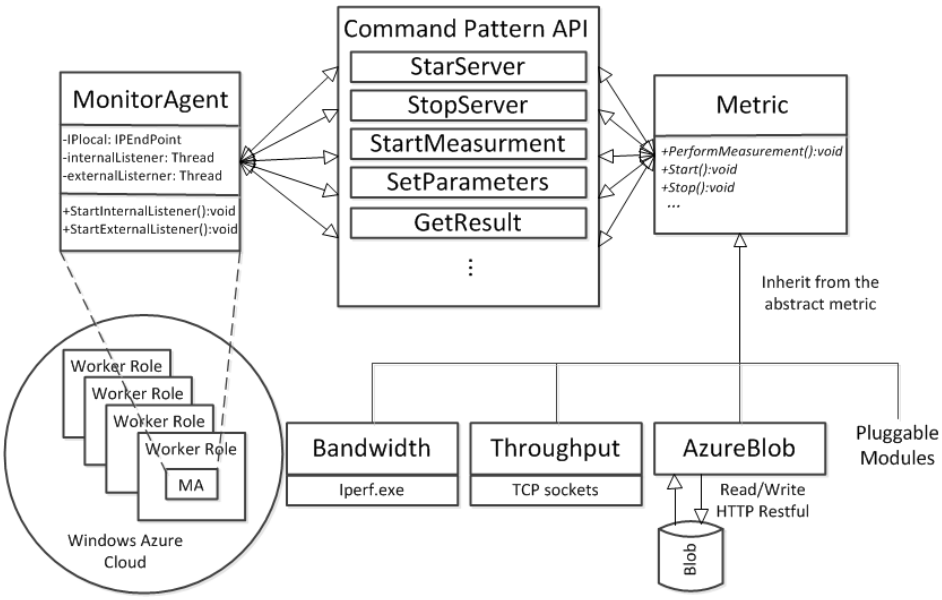

Figure 9: The implementation of the monitoring agent.

The Transfer Agent It is in charge of the data movements using parallel TCP streams, as depicted in the Figure 10. The transferring data is retrieved from local disk of the Worker Role or from the Azure Blobs. DataHandler aims to access to and process the transferring data including virtual splitting. Data is sent as chunks extended with metadata information. Metadata is used for hashing and deduplication, for recomposing the data at destination, as packages can arrive in any order, and for routing the packages and acknowledgements. The parallelization of the transfer is done at *cloud node level*: data is not directly sent from the source to the destination node, but part of it is sent to *intermediate nodes*. These are then forwarding the data towards the destination, exploiting the multiple *parallel paths existing between data centers*. The QueueHandler is responsible for processing the receiving chunks from multiple threads. The disordered chunks wait in a FIFO queue, and the system reconstruct and write the transferring data in the target disks. As future work, we consider adding support for other protocols (HTTP for content-based data or UDP for handling geographical streaming data). The data transfer functions can be accessed via an API or using a set of simple commands similar to the FTP ones. Hence, the Transfer Agents can be controlled both locally and from remote locations (e.g., from other nodes or data centers).

The Decision Manager It implements the modeling and the prediction components as we detailed in the section aforementioned by using the monitoring data. Based on them, the efficiency of the transfer is computed at the beginning at the transfer and then updated iteratively as the transfer progresses. The Decision Manager also coordinates the transfer (i.e. what nodes and resources to provision) based on the time/cost predictions in order to comply with the transfer constraints. For instance, it computes whether a benefit (economy) is brought by a set of resources if the transfer cost is within the limit, or if by using an increased number of nodes for parallel streaming of data the completion time can be significantly reduced. The Manager further communicates with the Transfer Agents to set the chunk size

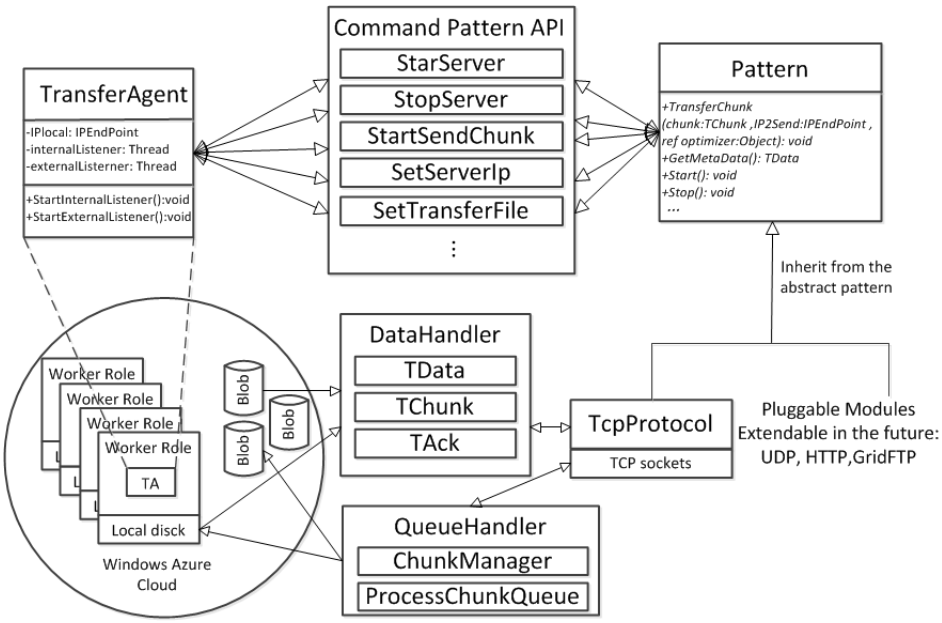

Figure 10: The implementation of the transfer agent.

of the transfers via intermediate nodes in order to maximize the resource usage while preserving the non-intrusiveness constraints. Finally, it detects any resource performance drops and either replaces them or asks the cloud middleware to change them.

5 Evaluation

The section presents the evaluation of our solution on the Azure cloud using synthetic benchmarks and a real life application from A-Brain.

5.1 Experimental Setup

The experiments were carried out on the Azure cloud using data centers from United States (North, South, East, West US) and Europe (North, West EU). We used the Small (1 CPU, 1.75 GB Memory) and Medium (2 CPU, 3.5 GB Memory) VM instances with tens of such machines deployed in each data center, reaching a total of 120 nodes and 220 cores in the global system. To execute the A-Brain application, we used the Extra-Large (8 CPU, 14 GB) VM instances.

5.2 Assessing the Cloud Infrastructure Variability

It is commonly known that clouds offer a variable performance, analyzed in several previous works [15, 11, 7]; the focus there is on the intra-data center performance. As we addressed aforementioned, intra data center environment is usually the local network where the network is relatively stable. When it comes to the global cloud infrastructure, the variability is expected to increase. In this first experimence, we mainly focus on the variability of the cloud network. We report here the results of an evaluation that considers the data transfers between the cloud data centers. We have designed two scenarios: assessing the throughput variability (Figure 11) and the latencies of the AzureBlobs remote storage service (Figure 12). In both cases the measurements were done from the North EU towards the other 5 EU and US data centers during one week, with hundreds of measurements each day.

Figure 11 shows the average throughput and the standard deviation achieved when sending 100MB of data using Small instances. The throughput is obtained as follows: $Th = \frac{DataSize}{Latency}$. As we observed from Figure 11, even in one day, the throughput across data centers varies a lot. For example, we observe that the coefficient of variation for the average throughput between North Europe (NEU) and East US (EUS) at day 1, is higher than 60%. If we compare the same link in different day, we also find the high variability. For instance, considering the same link between North Europe (NEU) and East US (EUS), the average throughput is doubled at day 3 comparing to the value at day 1. Moreover, we do not find any specific pattern for those variabilities and the performance drops or bursts can appear at any time. However, as we explained the complexity of WAN environment in Section 2, we are not surprised by this result. Another observation is that the geographical distance has impact to the transfer latency. As we illustrate in Figure 11 where the average throughput is significant higher between North Europe (NEU) and West Europe (WEU). This experiment result states that it makes sense to have a predictable model of the WAN environment between data centers, so that we can adaptably leverage the available cloud resources to move the data.

Figure 12 shows the average time and the standard derivation for sending 100MB data to the Azure Blobs. Here we only consider the write operation time from a VM instance to a blob that is located in another site via the RESTful HTTP interface. The result is close to

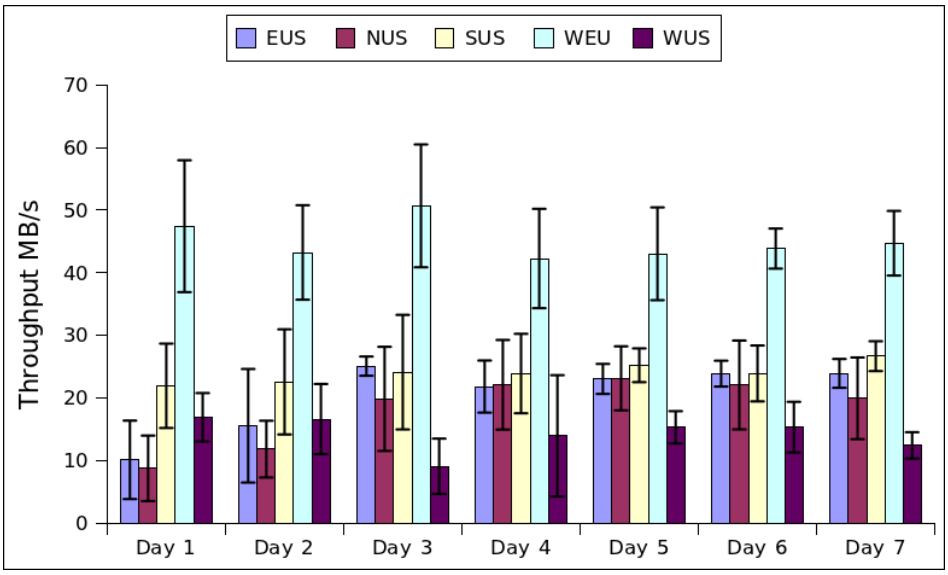

Figure 11: The throughput variation during a week between the North Europe data center and the other European and US data centers.

the result of the data transfer between instances, the WAN environment has high variability for the data movement. Besides this, we find that the latency will not be significant reduced when we write the data into the blob which is located in the neighbouring data center. As we observe in Figure 12, the write time between North Europe (NEU) and West Europe (WEU) is not always the shortest. The observations here hold both for the neighbouring data center and for the remote ones.

5.3 Synthetic Benchmarks

The next series of experiments evaluate the accuracy of the sample-based cloud model, the intrusiveness of our approach and its efficiency in terms of costs and performance, using a set of synthetic benchmarks.

5.3.1 Evaluating the performance prediction

Figures 13 and 14 present the accuracy of the estimations done using the monitoring based model, for a 24 hour interval. The figures show the hourly averages (60 values per hour). The critical aspect about the model's accuracy is how the new samples are integrated within the model. We have compared our solution, based on a weighted integration (*WSI*), with 2 other sample integration strategies. The first one (*Monitor*) considers that the last sample describes the current reality and uses it as expected performance; due to its simplicity and low cost it is widely used. However, in our case it gives the worst results as it is subject to the performance variations. The second strategy considers a linear integration of the samples (*LSI*), computing the future performance as an average between the history and the new sample.

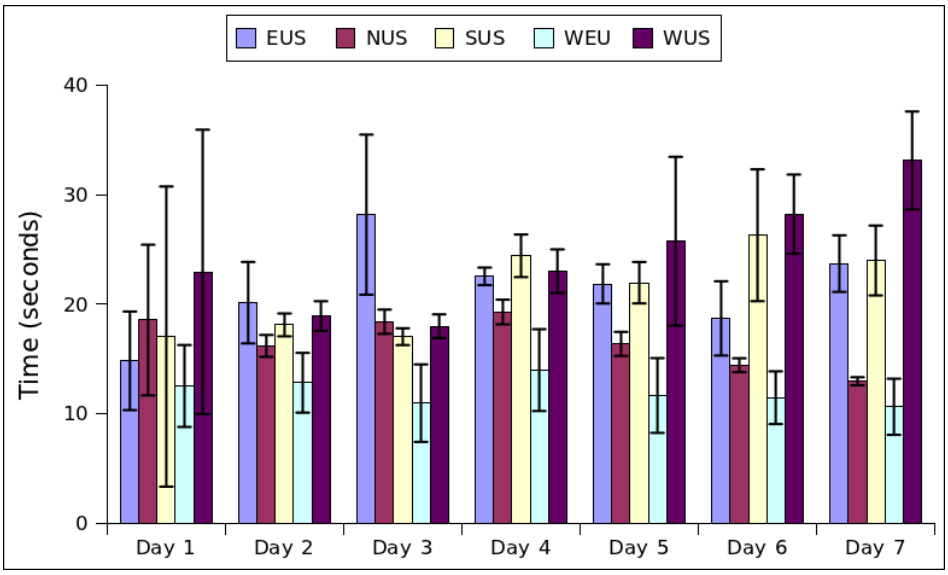

Figure 12: The variation of the time to write in AzureBlobs during a week between a client in the North Europe data center and all the other European and US data centers.

Figure 13 shows how the real value of the throughput between North US and North EU is approximated using the 3 strategies for sample integration. Our weighted approach has a smoother approximation and is not that sensitive to temporary variations as the other two. In Figure 14 we report the difference (i.e. accuracy error) between the estimated value and the real value. With an unstable tracked throughput (e.g., interval 1 to 5 or 18 to 24), weighting the samples seems a better option than using fixed integration strategies (LSI). When the performance variations are less important, both the linear and the weighted strategies give similar results. This is explained by the fact that in a stable environment the weights assigned to the samples converge towards 1, just like in the case of the linear average.

5.3.2 Evaluating the intrusiveness

We now analyze the impact of higher resource utilization due to our management system on the wide-area transfers. We measure the transfer time of 1 GB of data between the North EU and US data centers. The number of nodes that are used for the transfer is varied from 1 to 5. For each node, we also vary the intrusiveness parameter, which gives the percentage of the VM's resources (CPU, Memory and bandwidth) to be used by our system. The highest values from Figure 15 always correspond to the situation when only one node is used for the transfer.

The lower values are obtained using multiple nodes, leading to different transfer time reductions depending on the intrusiveness level and the cloud performance. Adding more resources does not reduce the transfer time with the same percentage because:

1. The network bandwidth is bound
2. Due to the overhead induced by the local transfer from the source node to the intermediate one

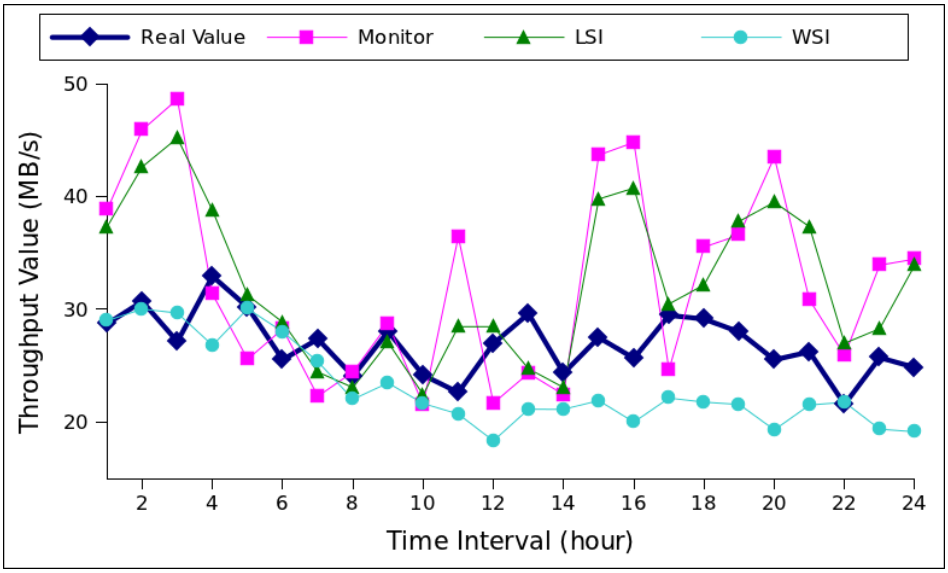

Figure 13: The approximation of the TCP throughput using multiple strategies (Monitor and Update, Linear Sampling Integration and Weighted Sampling Integration).

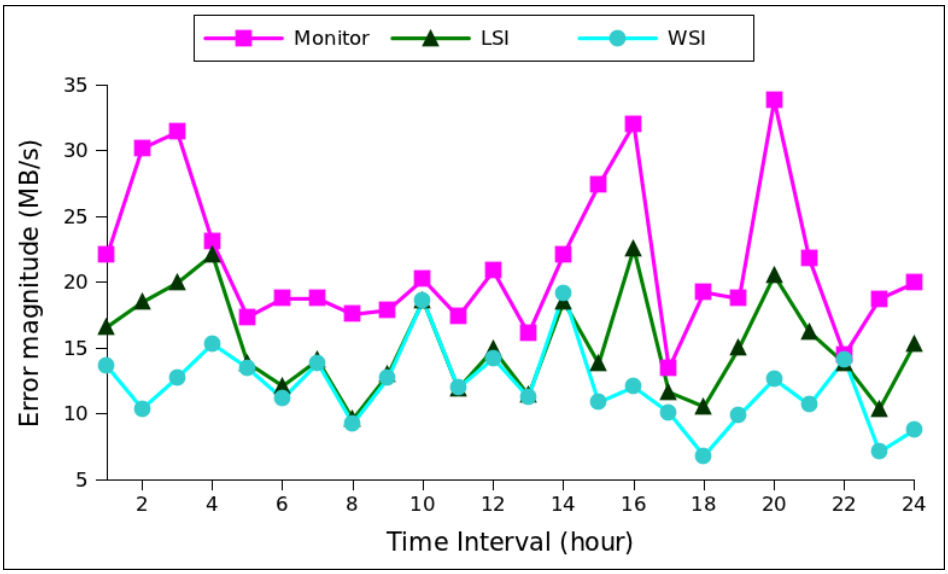

Figure 14: The average aggregated error in approximating the TCP Throughput depending on the strategies.

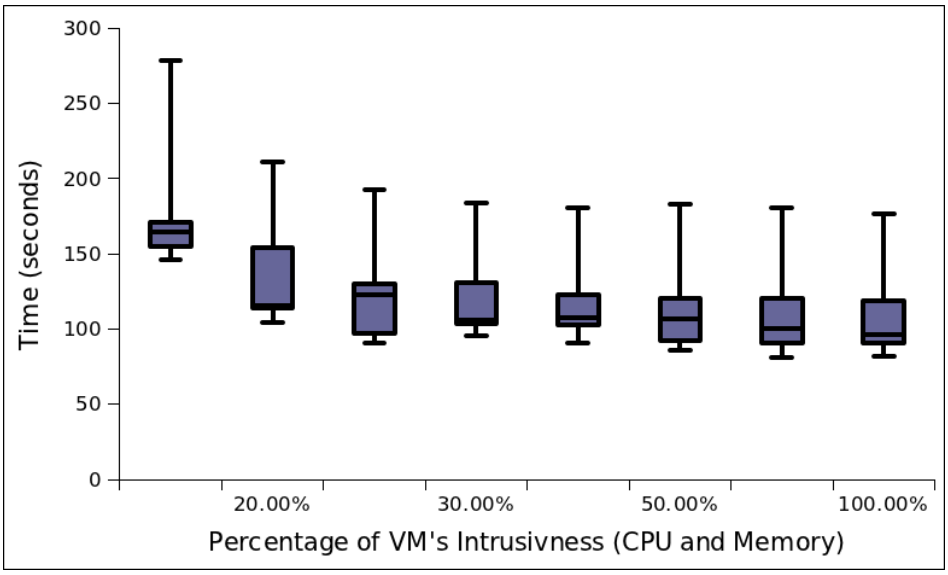

Figure 15: The impact of the intrusiveness on the transfer time of 1 GB of data between NUS and NEU, when varying the number of VMs used (1 to 5)

3. Due to the variable performance of the VMs.

All these lead to a limit in the time reduction that the resources (nodes, bandwidth, CPU or Memory) can bring when transferring data. These results motivate our choice for controlling the amount of resources to be used by the data management system in order to avoid impacting the application and wasting the resources that otherwise could be used for the scientific computation.

5.3.3 Evaluating the transfer efficiency

Our next experiments evaluate the relationship between the transfer efficiency and the monetary cost. We focus on the scenario using additional intermediate nodes to speed up parallel transfers. These extra nodes come at a cost as they have to be either leased or taken from the ones used for the scientific computation. Depending on how urgent the transfer is, this is an acceptable situation. In fact, up to a certain point, the time reduction obtained with more nodes prevents the transfer cost to grow significantly, as observed in Figure 16 when using 3 up to 5 VMs. This happens because the cost depends on the time the nodes are leased for the transfer - smaller transfer times reflect on smaller costs. Looking at the cost/time ratio an optimal point might be with 6 VMs in this case (the maximum time reduction for a minimum cost). However, as different applications can value costs differently, we provide them with the possibility of setting their customized cost/time tradeoff.

5.3.4 Evaluating the environment-aware wide-area transfers

The following experiment illustrates how the transfer efficiency is improved using knowledge about the environment. We consider sending increasing data sizes from a source node

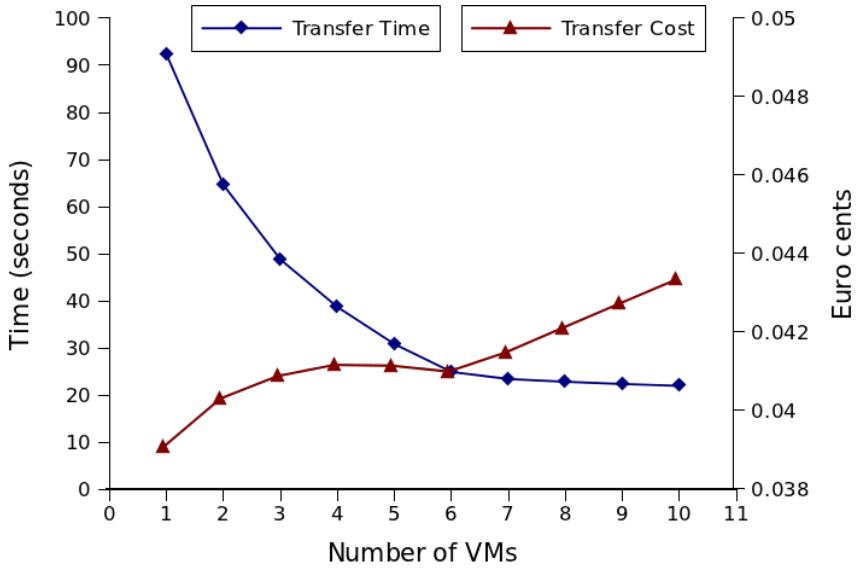

Figure 16: The tradeoff between transfer time and cost when using multiple VMs. The values correspond to 1 GB transferred between North EU and North US

to a destination. As we do the transfer in parallel, our solution uses intermediate nodes, from the same data center as the source node. During the transfer the performance (either CPU or bandwidth) of the used nodes can drop. Aware of these changes, the Decision Manager adapts to them by relying less on the problematic nodes (i.e. sending less data through the respective link) or by choosing some alternate nodes. We compared this environment-aware approach (GEO-DMS) with a strategy in which the transfer is simply done in parallel. The same number of nodes were used to send increasing amounts of data between two close (South and North US) and two farther (North EU and North US) data centers. The results in Figure 17 show that as the size of data and the distance between data center increases, the environment-aware approach reduces the transfer times. This is explained by the fact that the transfers take longer and cloud performance changes can occur during these periods. The results show, with a 95% confidence interval, that we can expect an improvement of up to 20% for large datasets over simple parallel transfers strategies.

5) Comparing the performance with the existing solutions. We compared our system with the Globus Online tool (using a GridFTP backend server), with AzureBlobs (as an intermediate storage for sending data between endpoints) and with direct transfers (Figure 18). AzureBlobs is the slowest option as the transfer has a writing phase with data being written by the source node to the storage, followed by a read phase in which the data is read back by the destination. These steps incur significant latencies due to the geographical remote location (of the source or destination) and the HTTP-based access interfaces. Despite these, AzureBlobs-based transfers are currently the only cloud offering for wide-area data movements. Globus Online is a good alternative but it lacks the cloud-awareness. Our solution reduces the overall transfer time with a factor of 5 over the default cloud offering and with up to 50% over other transfer options that can be adapted for the cloud.

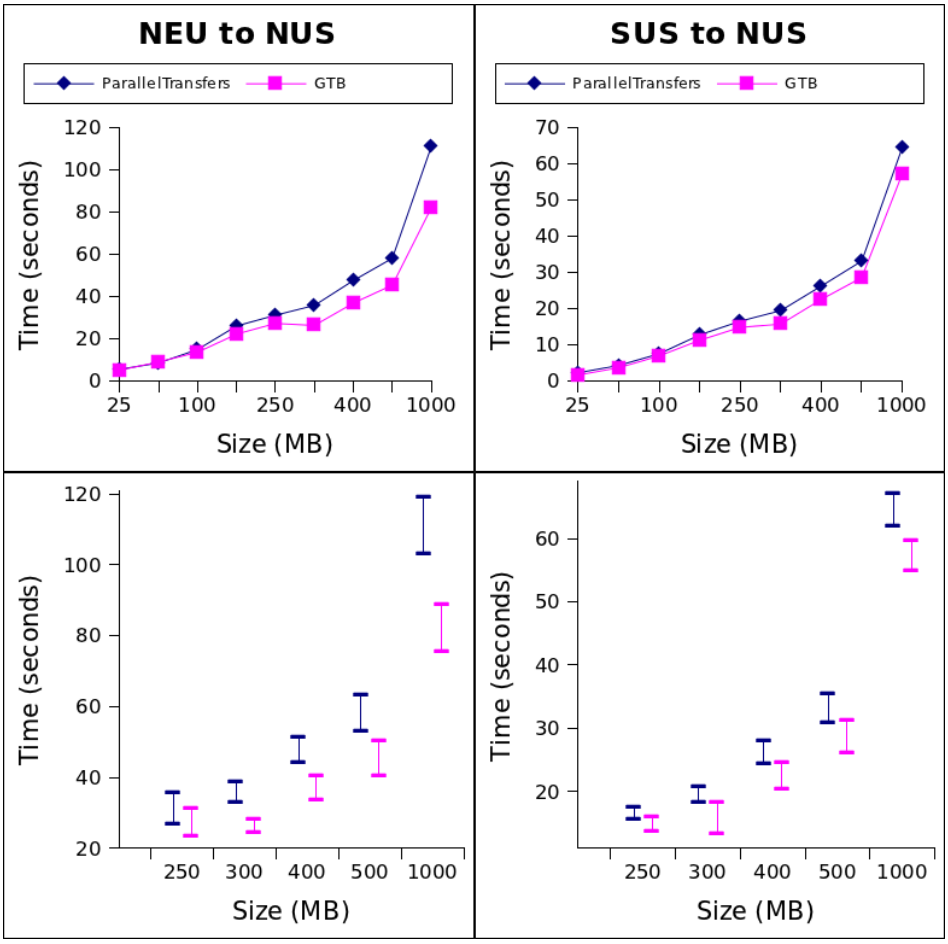

Figure 17: Our approach (GEO-DMS) vs. simple parallel transfers. The bottom figures give the 95% confidence intervals for the transfer time

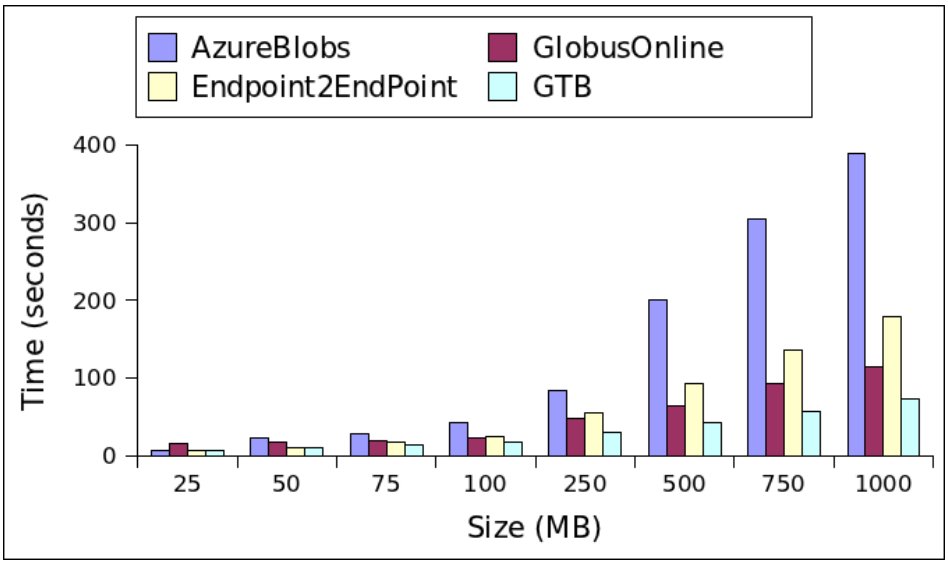

Figure 18: Transfer time when the data size is varied for our approach (GEO-DMS) and other existing options.

5.4 Experimenting with a Real-life Neuroimaging Application

We present an evaluation of the time gains that can be obtained for wide-area transfers in the context of a scientific application from bio-informatics, A-Brain, aiming at joint genetic and neuro-imaging data analysis. Due to the large resource requirements that could not be obtained from the cloud provider within a single data center, the application runs a MapReduce-based processing across 3 data centers; the final global result is computed using a Meta-Reducer [3] that aggregates results from all the datacenters. We compare the transfer times of 1000 files representing partial data, sent from each datacenter towards the Meta-Reducer, using AzureBlobs as a transfer backend and our solution (GEO-DMS). The results are shown in Figure 19 for multiple file sizes, resulted from different input data sets and configurations. For small datasets (108 MB from the $3 \times 1000 \times 36$ KB files), the overhead introduced by our solution, due to the extra acknowledgements, makes the transfer inefficient. However, as the data size grows (120 GB), the total transfer time is reduced by a factor of 3.

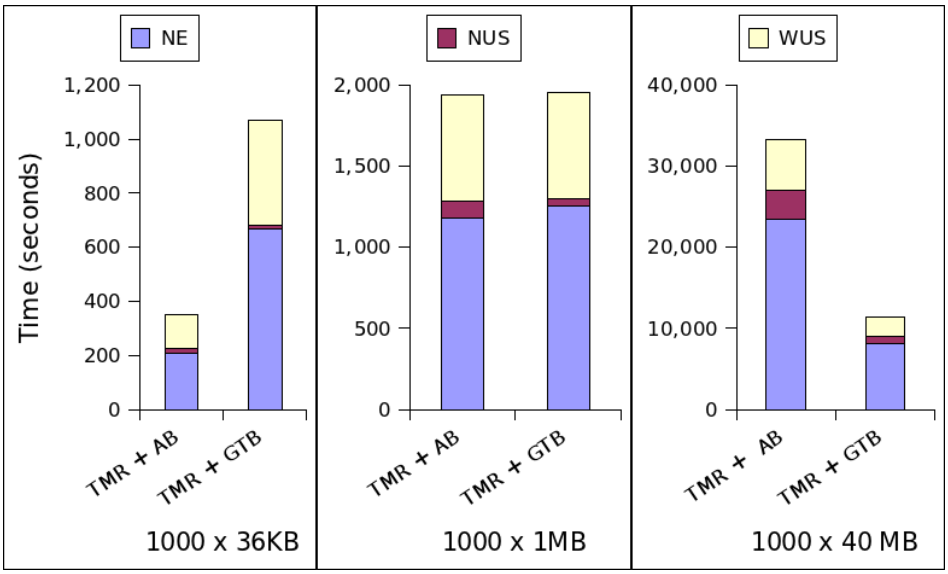

Figure 19: Execution times of the A-Brain application across 3 datacenters, using AzureBlobs and GEO-DMS as a transfer backend. The bar indicate the total time of transferring the files towards the Meta-Reducer located in NUS

6 Conclusion

6.1 Contributions

This thesis introduces a cloud-based data management system for big data science applications which run in large, federated and highly dynamic environments. Our solution is able to effectively use the high-speed networks connecting the cloud data centers through optimized protocol tuning and bottleneck avoidance, while remaining non-intrusive and easy to deploy. At its core, it uses a sampling-based model for cost-performance in a cloud setting to enable efficient transfer operations across a group of geographically distributed data centers. As an example, by distributing data locally, it enables high wide-area data throughput when the network core is underutilized, at minimal cost. Our experiments show that the system can achieve high performance in a variety of settings: it substantially improves throughput and it reduces the execution time for real applications.

Encouraged by these results, we have started to explore other research data-management issues, from a cloud providers perspective. Apart from the practical applications shown in this paper, our approach can be used to study the performance of inter-data center or inter-clouds transfers. This is especially useful for cloud users, which do not have visibility into the supported service levels. We believe that the cloud providers could benefit from using this tool as a metric to describe the performance of resources with particular configurations. Further, they could provide Introspection-as-a-Service to reveal information about the cloud internals to the interested applications.

6.2 Future Works

There are still several directions that worth exploring in the future.

With respect to the modeling part, we are planning to take more performance metrics into consideration. The available CPU load might be one of the potential metrics which helps us to build the CPU usage prediction models for the data transfer in the cloud. In addition, we would like to improve the weight of the prediction algorithm in order to get more accuracy predictions.

We are also planning to enhance the transfer strategies. For example, we consider adding some error detection and checkpoint mechanisms. With those strategies, the system could detect the transfer fails by itself and recovery the transfer jobs to the point where fails occured instead of restart the entire transmissions.

Another possible future work direction is to support more underlayer transfer protocols. Currently the transfer agent only implements TCP sockets for the data transfer which is not enough in the future. For instance, we could apply HTTP for content-based data or UDP for handling geographical streaming data. Moreover, some high layer transfer protocols (FTP, GridFTP) can be also reused. With this enhanced support, our system adapts to more types of data and achieve more efficient data transfers.

6.3 Lessons Learnt From the Internship

This internship provides me a good opportunity to gain the research experiences in the field of distributed systems and cloud computing. Through this five-month internship, I have enhanced the knowledge regarding the networks and cloud computing. I have the opportunity to manage and handle the cloud resources, to execute my native code in the cloud plateform, to deploy the applications in the cloud. I have adopted a rigorous approach to learn new knowledge for the research, ranging from bibliophical study, brainstorming sessions to the implementation and the evaluation. Furthermore, I have nurtured my interests and passions in starting the research career. I am also engaged in writing a part of the paper regarding all of our contributions with a PhD student to the IEEE Cluster 2013 conference.

References

[1] William Allcock, John Bresnahan, Rajkumar Kettimuthu, Michael Link, Catalin Dumitrescu, Ioan Raicu, and Ian Foster. The Globus Striped GridFTP Framework and Server. In *Proceedings of the 2005 ACM/IEEE conference on Supercomputing*, SC '05, pages 54–. IEEE Computer Society, 2005.

[2] Rajkumar Buyya, Rajiv Ranjan, and Rodrigo N. Calheiros. InterCloud: utility-oriented federation of cloud computing environments for scaling of application services. In *Proceedings of the 10th international conference on Algorithms and Architectures for Parallel Processing - Volume Part I*, ICA3PP'10, pages 13–31. Springer-Verlag, 2010.

[3] Alexandru Costan, Radu Tudoran, Gabriel Antoniu, and Goetz Brasche. TomusBlobs: Scalable Data-intensive Processing on Azure Clouds. *Journal of Concurrency and computation: practice and experience*, 2013.

[4] Ian Foster, Ann Chervenak, Dan Gunter, Kate Keahey, Ravi Madduri, and Raj Kettimuthu. Enabling PETASCALE Data Movement and Analysis. *Scidac Review*, Winter 2009.

[5] Y. Gu and R.L. Grossman. Sector and sphere: the design and implementation of a high-performance data cloud. In *Philosphical Transactions A Special Issue associated with the 2008 UK e-Science All Hands Meeting*, 2009.

[6] T. J. Hacker, B. D. Noble, and B. D. Athey. Adaptive data block scheduling for parallel TCP streams. In *Proceedings of the High Performance Distributed Computing, 2005. HPDC-14. Proceedings. 14th IEEE International Symposium*, HPDC '05, pages 265–275. IEEE Computer Society, 2005.

[7] Gideon Juve, Ewa Deelman, Karan Vahi, Gaurang Mehta, Bruce Berriman, Benjamin P. Berman, and Phil Maechling. Data Sharing Options for Scientific Workflows on Amazon EC2. In *Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '10, pages 1–9. IEEE Computer Society, 2010.

[8] Gaurav Khanna, Umit Catalyurek, Tahsin Kurc, Rajkumar Kettimuthu, P. Sadayappan, Ian Foster, and Joel Saltz. Using overlays for efficient data transfer over shared wide-area networks. In *Proceedings of the 2008 ACM/IEEE conference on Supercomputing*, SC '08, pages 47:1–47:12. IEEE Press, 2008.

[9] Tevfik Kosar and Miron Livny. A framework for reliable and efficient data placement in distributed computing systems. *J. Parallel Distrib. Comput.*, pages 1146–1157, 2005.

[10] Nikolaos Laoutaris, Michael Sirivianos, Xiaoyuan Yang, and Pablo Rodriguez. Inter-datacenter bulk transfers with netstitcher. *SIGCOMM Comput. Commun. Rev.*, pages 74–85, 2011.

[11] Hill J. Li, M. Mao, A. Ruiz-Alvarez, and M. Humphrey. Early Observations on the Performance of Windows Azure. In *High Performance Distributed Computing*, pages 367–376, 2010.

[12] Wantao Liu, Brian Tieman, Rajkumar Kettimuthu, and Ian Foster. A data transfer framework for large-scale science experiments. In *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing*, HPDC '10, pages 717–724. ACM, 2010.

[13] P. Mell and T. Grance. The NIST Definition of Cloud Computing. Technical report, National Institute of Standards and Technology, 2011.

[14] Costin Raiciu, Christopher Pluntke, Sebastien Barre, Adam Greenhalgh, Damon Wischik, and Mark Handley. Data center networking with multipath TCP. In *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*, Hotnets-IX, pages 10:1–10:6, New York, NY, USA, 2010. ACM.

[15] Radu Tudoran, Alexandru Costan, Gabriel Antoniu, and Luc Bougé. A Performance Evaluation of Azure and Nimbus Clouds for Scientific Applications. In *CloudCP 2012 – 2nd International Workshop on Cloud Computing Platforms, Held in conjunction with the ACM SIGOPS Eurosys 12 conference*, Bern, Switzerland, 2012.

[16] A-Brain. `http://www.msr-inria.inria.fr/Projects/a-brain`.

[17] Amazon EC2. `http://aws.amazon.com/`.

[18] The Big Bang: How the Big Data Explosion Is Changing the World. `http://www.microsoft.com/en-us/news/features/2013/feb13/02-11BigData.aspx/`.

[19] DropBox. `www.dropbox.com/`.

[20] Google App Engine. `https://appengine.google.com/`.

[21] Iperf. `http://iperf.fr/`.

[22] Nimbus. `http://www.nimbusproject.org/`.

[23] OpenNebula. `http://opennebula.org/`.

[24] SalesForce. `http://www.salesforce.com/`.

[25] How sockets work. `http://pic.dhe.ibm.com/infocenter/iseries/v7r1m0/index.jsp?topic=%2Frzab6%2Fhowdosockets.htm`.

[26] Windows Azure. `http://www.windowsazure.com/en-us/`.