ARTICLE IN PRESS

Computer Networks xxx (2014) xxx–xxx

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

Optimal application allocation on multiple public clouds

Simon S. Woo *, Jelena Mirkovic

Computer Science Department, University of Southern California, Los Angeles, CA, USA
Information Science Institute, Marina Del Rey, CA, USA

ARTICLE INFO

Article history:
Received 17 June 2013
Received in revised form 5 November 2013
Accepted 4 December 2013
Available online xxxx

Keywords:
Cloud computing
Resource allocation
Hybrid cloud computing architecture
Federated cloud computing
Multiple public cloud computing

ABSTRACT

Cloud computing customers currently host all of their application components at a single cloud provider. Single-provider hosting eases maintenance tasks, but reduces resilience to failures. Recent research (Li et al., 2010) also shows that providers' offers differ greatly in performance and price, and no single provider is the best in all service categories. In this paper we investigate the benefits of allocating components of a distributed application on multiple public clouds (multi-cloud). We propose a resource allocation algorithm that minimizes the overall cloud operation cost, while satisfying required service-level agreements (SLAs). In spite of the additional delays for inter-cloud communication and the additional costs for inter-cloud data transfer, our simulation study, using real cloud performance and cost data, demonstrates that multi-cloud allocation outperforms single-cloud allocations in a variety of realistic scenarios.

1. Introduction

Cloud providers vie for customers by offering differentiated, reliable, specialized, unique, or cheaper services than their competitors. The performance of generic services, such as Web server hosting and computing, can vary greatly between providers given the same price range [1]. Such an environment makes *multi-cloud resource allocation* appealing. If providers differ greatly in their offers, hosting components of a distributed application with different providers should lead to a better price/performance trade-off than hosting them all at any single provider. In addition to cost and performance considerations, single-provider hosting lowers the reliability of an application in case of a provider-wide outage [2]. Although providers try to guarantee 99.9% availability through vertical and horizontal computing resource scaling, scalability can also be a critical issue for a single-provider approach as the number of applications and demands increase in the future. Furthermore, recent research shows that multi-cloud hosting has lower latency, as seen by end-users, than single-cloud hosting [3].

On the other hand, multi-cloud hosting adds communication delays and extra operational cost for inter-cloud data transfer. Also, the multi-cloud approach adds more complexity to application maintenance due to the lack of standard application programming interfaces (APIs) for application deployment on different clouds.

The main goal of this work is to evaluate performance and cost of multi-cloud resource allocation, and to compare these to the performance and cost of the single-provider approach for a variety of realistic cloud use scenarios. We do not assume anything about an end user's platform (e.g., fixed vs. mobile), nor how a user's application tasks are divided between their end platform and the clouds. The end-user observed delay consists of the time spent to perform tasks on one or more public clouds, and the time to transfer the data between the user's device

* Corresponding author at: Computer Science Department, University of Southern California, Los Angeles, CA, USA. Tel.: +1 310 448 9170; fax: +1 310 448 9300.
E-mail addresses: simonwoo@usc.edu (S.S. Woo), sunshine@isi.edu (J. Mirkovic).

http://dx.doi.org/10.1016/j.comnet.2013.12.001
1389-1286/

Please cite this article in press as: S.S. Woo, J. Mirkovic, Optimal application allocation on multiple public clouds, Comput. Netw. (2014), http://dx.doi.org/10.1016/j.comnet.2013.12.001

and the clouds. Our study focuses on optimizing the first factor in this sum – the time spent to perform tasks on one or more public clouds. This factor remains constant for any end user's platform, but the division of application tasks between the user's device and the cloud, as well as delay requirements, may change. We investigate a wide range of delay requirements in our work and show that for many of them multi-cloud allocation outperforms most of single-cloud allocations. Thus, we believe our results apply to a wide range of end-user's platforms. The delay between end user and the cloud changes based on a cloud's location, thus some allocations that benefit from multi-cloud allocation approach may not be acceptable to users that are physically far away from one or more clouds. We leave consideration of this second factor in end-user observed delay for our future work. Also, mobile users are bandwidth-constrained and their applications may be distributed between their platform and the clouds in a way that minimizes data transfer between the user's device and the cloud. We believe an application developer would decide which application tasks are best performed on clouds. Our algorithm can then be used to find an optimal allocation for this subset of tasks. Also, an end user may have security requirements about which application components can be hosted on which public clouds. We leave consideration of this to our future work.

While others have proposed hybrid or federated cloud computing paradigms [4,5] and have evaluated the benefits of private/public cloud allocations [6,7], ours is the first work that extensively evaluates the benefits of application allocation on multiple public clouds. A recent publication [3] shows that multi-cloud allocation reduces the delay as seen by the end user. That research focuses only on Web service hosting and evaluates only the delay aspect of resource allocation. We, on the other hand, focus on several popular types of cloud applications and evaluate both the delay and the cost of multi-cloud and single-cloud allocations.

Our contributions are:

1. We propose a novel cloud resource allocation algorithm that determines the best allocation of application components over multiple public cloud providers, under a given performance constraint.
2. We show through extensive evaluation, which relies on realistic benchmark data about cloud performance and price, as described in [1], that multi-cloud allocation outperforms single-cloud allocation in a variety of realistic cloud use scenarios. We also investigate the cloud and application features that best bring out the benefits of multi-cloud resource allocation.

2. Related work

A hybrid or a federated cloud is composed of two or more private, community, and/or public clouds which work together to achieve the application objective, while each cloud remains as a unique entity [8]. Typically, in a hybrid cloud, the organization manages and uses in-house computing resources as well as external cloud resources. A multi-cloud is a specific form of the hybrid cloud where all clouds are public clouds.

Migrating parts of the applications to a cloud has been addressed in [9]. Also, [10] provides an approach to dynamically split transactions in different data-centers. However, [9,10] assume allocation on a single cloud provider. Publications [4,5] point out the potential scaling problem of single-provider hosting as the number of users and applications increase, and, thus, champion federation among different cloud providers. These publications, however, focus only on reliability and do not evaluate the potential performance and cost-saving benefits of federated allocation.

Publications [11,12] similarly focus on overcoming inter-cloud interoperability, workload distribution, and inter-cloud policy issues of hybrid or federated cloud hosting. Publications [7,13] considered the application of the public/private clouds to improve the cost and response time of applications. However, they only consider the use of compute machines and private/public cloud allocation. On the other hand, we consider more cloud services (compute, storage and DB resources) and allocation on multiple public clouds. Also, we focus on the evaluation of performance and cost benefits, and we provide a side-by-side comparison between a single-cloud and multiple public clouds approach.

A recent publication [3] shows that multi-cloud allocation reduces the delay as seen by the end user, but focuses only on Web service hosting and delay benefits. We, on the other hand, investigate more cloud applications, and we evaluate both the performance and the cost of each allocation scenario.

3. Multi-cloud resource allocation

While cloud providers try to differentiate themselves from their competitors by offering some unique, specialized services and APIs, most clouds offer generic compute, database (DB), and storage services and have a similar price structure for them (i.e., some combination of flat rate and per usage cost). In [1], the authors benchmarked these generic services over four popular cloud providers and had two major findings:

1. No single provider offered the best performance in all three service categories. For example, one provider offered very fast computing resources but slower storage and no DB service. Another offered fast DB service and fast binary large object (blob) storage (for some blob sizes), but its compute machines were much slower than its competitors'.
2. There was a large difference in price between providers for the same level of performance for a given service.

If a cloud application needs resources from several of these generic services, the differences in performance and price across providers make a multi-cloud resource allocation an attractive choice. Depending on the exact resources being allocated and the application workload, such multi-cloud allocation has the potential to outperform

Please cite this article in press as: S.S. Woo, J. Mirkovic, Optimal application allocation on multiple public clouds, Comput. Netw. (2014), http://dx.doi.org/10.1016/j.comnet.2013.12.001

single-cloud allocations, both in performance and in price. However, multi-cloud allocation carries the additional costs for inter-cloud data transfer and additional delays for communication between clouds. In this paper, we investigate the benefits and costs of multi-cloud resource allocation and identify cases in which it outperforms single-cloud allocations.

3.1. Cloud performance and cost

In our evaluation, we use the results published in [1] for Microsoft, Amazon and Google cloud providers to calculate the performance and the cost of each resource allocation. We obtained raw data from the authors of [1] and extracted detailed performance data for each service and each provider. We anonymize provider names as A, B, and C in random order. In [1], the authors benchmark compute, DB and storage services using a variety of micro-tasks. For compute services, they use the following sixteen micro-tasks: Security (compression, AES encrypt/decrypt, RSA encrypt/decrypt, sign/verify), Multimedia (mpeg encode/decode), CPU (Fast Fourier Transform, LU decomposition, Monte Carlo simulation, linear equation solving, sparse matrix calculation, 3D rendering), I/O (read, write and read after write, all for a 100 MB file from local disk), and Memory (read/write for <2 MB chunk, read/write for >10 MB memory chunk). For DB services, they measure the time needed for a cloud to process a `put`, a `get`, and a `query` operation. For blob storage, they measure the time needed to upload or download a 1 KB and a 10 MB blob of data. Each of the micro-tasks is measured multiple times for a given cloud and a given class of resources (e.g., slower vs. faster compute machines in the same cloud), and the median and standard deviations for the measured values are shown in [1]. In our calculations, we use only the median value to evaluate the performance since standard deviations are very small for all tasks.

Fig. 1 shows the performance and cost data for each micro-task, class of resources, and each cloud provider. Provider A has three classes of compute resources (A1–A3), one type of DB service (A4), and one type of storage (A5). Provider B has four classes of compute resources (B1–B4), one type of DB service (B5), and one type of storage (B6). Provider C has seven classes of compute resources (C1–C7), no DB service, and one type of storage (C9). The time to complete a micro-task is given in seconds, and the cost is given in US dollars. Micro-tasks `mem`, `io_1KB` and `io_10MB` are introduced by us for purposes of evaluating different application deployment scenarios, as explained in Section 4. The highlighted cells show the best performance (i.e., lowest completion time) for each micro-task.

3.2. Cloud applications and workloads

For our evaluation, we needed a set of typical cloud applications. It is extremely difficult to produce such a set.

First, clouds are used for a multitude of purposes today, such as e-business, search, social networking, high-performance computing, distributed storage, and multimedia streaming. Each of these classes has very different architecture and priorities with regard to minimizing task completion time vs. minimizing cost of resource allocation.

Second, even within the same class of cloud applications (e.g., e-business), the implementation of the application will influence its resource needs for the optimal resource allocation. For example, one application may use compute machines to run both the front-end Web portal and the DB service, while another application may need to use a combination of compute machines for front-end Web portal and a cloud DB service. The first scenario requires only compute resources, and a single-provider allocation will always be the best. The second scenario requires compute and DB resources, and for some SLA constraints, a multi-cloud allocation may outperform allocations on a single cloud.

Third, there is no set of cloud application implementations that is widely used. Usually, cloud applications are developed in-house by stitching together popular implementations for each component (e.g., an Apache Web server and a MySQL database to create an e-business cloud application).

Fourth, the exact performance of the application will depend on its workload. For example, one e-business may perform a query operation on a cloud DB ten times more often than another, thus increasing its task completion time and resource cost.

All of these factors are too complex and varied, and we could not produce an accurate model of "common cloud applications and their tasks" in our study. Instead we aim to define a realistic set of application workflows and use them in our evaluation. We define a detailed workflow of each application, expressing it as a set of transactions, combined serially or in parallel. A transaction in turn consists of micro-tasks from [1], combined serially or in parallel. While a real cloud application may have a different workflow than the one in our evaluation set, we believe that the transactions we have defined, along with their composition of micro-tasks, are generic enough to fit a wide range of cloud applications in a given category, and that differences between a real cloud application's workflow and our representation would be small. In reality, an application developer could either use a profiling tool, such as CloudProphet [14], or his/her knowledge of the application source code to devise an appropriate workflow and feed it into our resource allocation algorithm.

3.2.1. Application workflow

We represent a cloud application's workflow as a sequence of multiple transactions. Each transaction in turn consists of micro-tasks from [1] executed in serial and/or parallel combinations. The transactions we consider in this paper are shown in Fig. 2, and their descriptions are given below:

1. *T1: Displaying a static or dynamic Web page to the user* – this includes just a memory read assuming that the Web page is frequently used and is in memory and not on disk.
2. *T2: Searching for a product or some other information and displaying it* – this includes running a `query` operation on a cloud DB, one or more `get` operations to fetch the data about products, and a Web page generation and display using these results.

Please cite this article in press as: S.S. Woo, J. Mirkovic, Optimal application allocation on multiple public clouds, Comput. Netw. (2014), http://dx.doi.org/10.1016/j.comnet.2013.12.001

	A1	A2	A3	A4	A5
compress	5.84	2.48	2.17		
AES	9.66	1.86	1.73		
RSA	24.61	10.1	7.15		
sign/verify	6.66	1.51	1.44		
mpeg	14.04	4.61	4.17		
fft	4.54	2.35	1.78		
lu	7.26	3.8	1.87		
monte_carlo	1.72	0.35	0.27		
sor	6.01	2.41	2.12		
sparse	7.22	3.9	2.98		
sunflow	14.37	7.41	6.05		
mem_small	2.35	0.93	0.71		
mem_large	12.37	1.84	1.46		
io_read	0.93	0.84	0.34		
io_write	1.34	1.02	0.85		
io_readwrite	2.7	1.96	2.26		
mem	0.001175	0.000465	0.000355		
io_1KB	0.000027	0.0000196	0.0000226		
io_10MB	0.27	0.196	0.226		
get				0.02	
put				0.04	
query				0.05	
st_1KB_up					0.07
st_10MB_up					1.5
st_1KB_down					0.12
st_10MB_down					2.5
flat_rate	61.2	244.80	489.60	129.6	2095
per_trans				0.00001	
st_1TB					95
st_10TB					860
st_100TB					7760

(a) Provider A

	B1	B2	B3	B4	B5	B6
compress	2.65	2.95	2.78	2.7		
AES	5.34	5.93	5.49	5.35		
RSA	11.19	12.52	11.54	11.18		
sign/verify	3.21	3.63	3.36	3.25		
mpeg	7.33	8.1	7.54	7.29		
fft	1.76	1.97	1.82	1.79		
lu	2.16	2.4	2.24	2.2		
monte_carlo	0.96	1.06	1.02	1		
sor	3.4	3.77	3.48	3.4		
sparse	2.76	3.06	2.82	2.76		
sunflow	6.66	7.52	7	6.84		
mem_small	0.88	1.02	1	0.99		
mem_large	1.12	1.3	1.23	1.22		
io_read	0.75	1.01	0.33	0.25		
io_write	5.01	6.11	2.43	2.13		
io_readwrite	2.49	1.99	1.9	2.71		
mem	0.00044	0.00051	0.0005	0.000495		
io_1KB	0.0000501	0.0000611	0.0000243	0.0000271		
io_10MB	0.501	0.611	0.243	0.271		
get					0.04	
put					0.05	
query					7	
st_1KB_up						0.03
st_10MB_up						2
st_1KB_down						0.05
st_10MB_down						4.5
flat_rate	86.4	172.8	345.60	691.20	87.9	1841
per_trans					0.00001	
st_1TB						85
st_10TB						769
st_100TB						6799

(b) Provider B

	C1	C2	C3	C4	C5	C6	C7	C9
compress	2.53	2.11	2.12	2.07	2.02			
AES	1.65	1.64	1.65	1.62	1.63	1.7	1.65	
RSA	6.68	6.63	6.62	6.7	6.78			
sign/verify	1.28	1.26	1.26	1.27	1.29			
mpeg	3.86	3.77	3.77	3.78	3.76			
fft	1.61	1.64	1.67	1.79	1.79	1.8	1.8	
lu	1.37	1.38	1.38	1.37	1.36			
monte_carlo	0.32	0.32	0.31	0.31	0.31			
sor	2.27	2.27	2.27	2.27	2.27			
sparse	1.38	2.5	2.49	2.35	2.47	2.4	2.38	
sunflow	5.85	5.88	5.88	5.85	5.73			
mem_small	0.54	0.53	0.54	0.48	0.5	0.9	1.07	
mem_large	0.96	0.98	1.01	0.95	0.93	0.89	1.01	
io_read	0.73	0.65	0.64	0.63	0.76	1.68	0.66	
io_write	1.5	0.8	3.15	1.51	5.65	49.15	0.69	
io_readwrite	1.73	1.19	1.34	1.37				
mem	0.00027	0.000265	0.00027	0.00024	0.00025	0.00045	0.000535	
io_1KB	0.0000173	0.0000119	0.0000315	0.0000151	0.0000565	0.0004915	0.0000069	
io_10MB	0.173	0.119	0.315	0.151	0.565	4.915	0.069	
get								
put								
query								
st_1KB_up								0.018
st_10MB_up								3.2
st_1KB_down								0.24
st_10MB_down								4.2
flat_rate	10.80	21.60	43.20	86.40	172.80	345.60	691.20	2095
per_trans								
st_1TB								79.87
st_10TB								663.25
st_100TB								6261.06

(c) Provider C

Fig. 1. Performance and cost parameters from [1] over different micro-tasks.

3. *T3/T4: Finding and displaying user or product information* – *T3* version of this transaction assumes that a cloud DB only stores information about a file storing user/product details, and a download operation from cloud storage is needed to fetch these details, followed by Web page generation and display to user. Transaction *T4* assumes that user/product details are small enough to be kept in the cloud DB.
4. *T5: Secure sign on* – this transaction uses a sign/verify operation to verify a user's identity and set up a browser cookie.
5. *T6/T7: Finding user information and displaying it securely* – these transactions include a *T3/T4* transaction to obtain user information and an additional encryption (AES) to securely display it.
6. *T8: Processing and logging a purchase order* – this transaction encrypts user input, stores this information in a cloud DB, and displays a confirmation page.
7. *T9: Making a move in an online game and logging it* – this transaction runs three micro-tasks in parallel. First, the new game screen is rendered to the user.

Please cite this article in press as: S.S. Woo, J. Mirkovic, Optimal application allocation on multiple public clouds, Comput. Netw. (2014), http://dx.doi.org/10.1016/j.comnet.2013.12.001

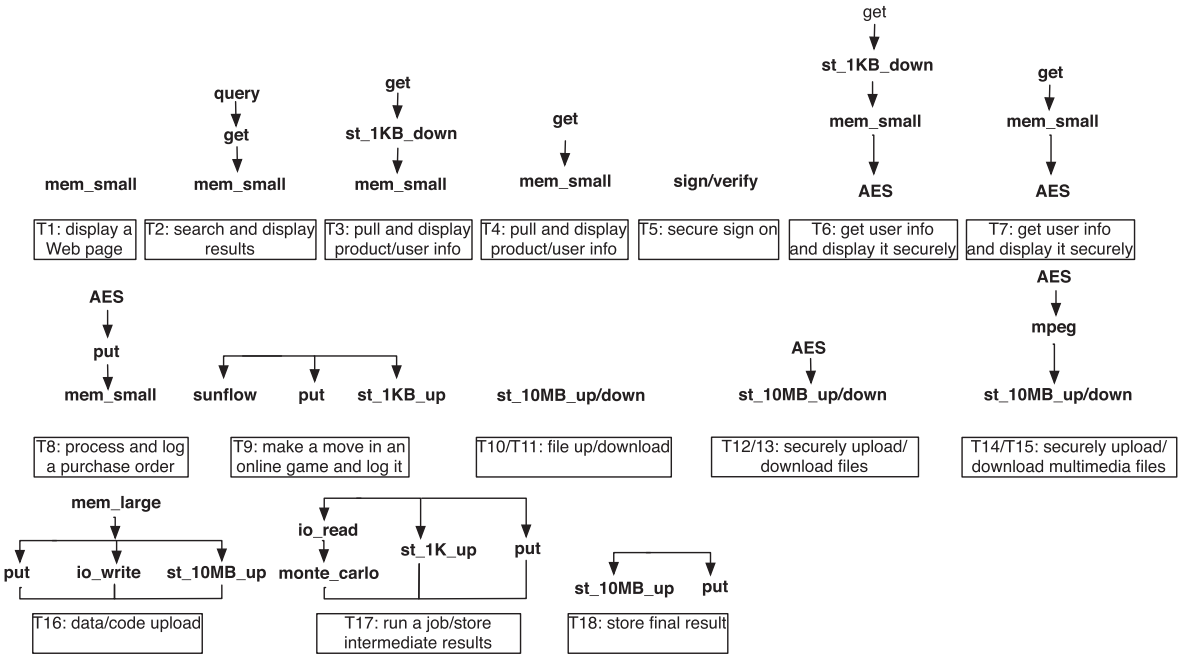


Fig. 2. Transactions we consider in our evaluation.

Second, information about user moves and position is stored in a cloud DB. Third, some game log entries are generated and stored in a cloud storage.

8. *T10/T11: Downloading or uploading a large file* – this transaction uploads/downloads a 10 MB file to/from cloud storage.
9. *T12/T13: Securely uploading/downloading files* – these transactions encrypt/decrypt a file the user wants to upload/download and perform the desired operation with cloud storage.
10. *T14/T15: Securely uploading/downloading multimedia files* – these transactions encrypt/decrypt a file the user wants to upload/download, run MPEG encoding/decoding, and perform the desired operation with cloud storage.
11. *T16: Data/code uploading* – this transaction is used at the start of a scientific computation. It loads large code/data from the user into memory, stores some information about it in a cloud DB, writes it to a local disk (for fast future computation), and stores it in cloud storage for resiliency.
12. *T17: Running a job/store intermediate results* – this transaction runs one map/reduce cycle. In one branch it reads from a local disk and performs some calculation (e.g., Monte Carlo simulation). In the other two parallel branches, it stores some intermediate result into cloud storage and may also log some information into cloud DB.
13. *T18: Storing final result* – this transaction is run at the end of a scientific computation. It stores the final result into cloud storage and may log some information about the job status in a cloud DB.

We then assemble workflows of five typical cloud applications out of our transactions as follows:

1. An e-business site (*ebus*), such as Amazon [15], has transactions *T1*, *T2*, *T3*, *T5*, *T6*, and *T8*.
2. An online gaming site (*game*), such as World of Warcraft Online [16], has transactions *T1*, *T5*, *T6*, and *T9*.
3. A file storage site (*storage*), such as Dropbox [17], has transactions *T1*, *T5*, *T7*, and *T12/T13*.
4. A multimedia storage/sharing site (*multimed*), such as YouTube [18], has transactions *T1*, *T2*, *T4*, *T5*, *T7*, and *T14/T15*.
5. A scientific computation site (*sci comp*), such as Penguin Computing [19], has transactions *T1*, *T5*, *T7*, *T11*, *T16*, *T17*, and *T18*.

In a real cloud computing application, some users may repeat some transactions, skip some transactions, or perform transactions in a different order. Thus, an application may have a number of workflows that need to be evaluated to find the best set of resources for them. The final set of resources would then be a union of the resources found for each workflow. In our evaluation we use only one workflow per application.

A real application may further perform some of our modeled tasks differently. For example, a small e-business site may decide to host its DB locally on a disk of a computation machine, thus, replacing the `query` and `get` micro-tasks from some transactions with a memory or I/O read. Or a scientific computing site may decide to encrypt all user communication, adding a `AES` micro-task to transactions *T16–T18*. While we cannot claim that our workflows match some of the popular cloud applications exactly, we believe that they are reasonably close, such that our conclusions hold in a variety of real cloud use scenarios.

Further, a real application may choose to combine or pipeline some micro-tasks (e.g., an e-business site may

Please cite this article in press as: S.S. Woo, J. Mirkovic, Optimal application allocation on multiple public clouds, Comput. Netw. (2014), http://dx.doi.org/10.1016/j.comnet.2013.12.001

store encrypted user information so it can be served faster), or it may start encrypting the profile information in parallel with signature verification. Thus, the transaction times we use in an evaluation are really an upper bound on the time it would take to perform a given transaction on a given set of cloud resources.

Finally, an application developer may use a cloud solution for storage and/or DB, or he/she may decide to write his/her own distributed storage or database implementation. In our evaluation, we consider all of these possibilities. We start with the application workflows as described above, which all have a combination of computation, DB and storage tasks. We then convert storage and/or DB tasks into I/O and memory access respectively to evaluate alternative workflows where the developer implements his/her own solution instead of using the one offered by the cloud.

We define the time to complete a transaction as the highest sum of times to complete each of micro-tasks *over the serial paths in the transaction*:

$$T^{transaction} = \max_{sp} \sum_{i=0}^{m_{sp}} (t_i + k_i \cdot T_{ic}) \quad (1)$$

where m_{sp} is the number of micro-tasks in each serial path sp, t_i is the time needed for each micro-task on the given cloud resource class for the given cloud provider, T_{ic} is the round-trip time between two clouds, and k_i is 1 whenever there is a transition between resources offered by different providers, either within a transaction or between two transactions. For example, if we had a workflow consisting of *T2* and *T5*, and if we selected B5 and A1 for *T2* and C1 for *T5*, then T_{ic} would be added twice to the time needed for *T2* – once for the "B5-to-A1" transition, and once for the "A1-to-C1" transition.

An application needs some set of resources for the micro-tasks. Most resources have a flat-rate cost, and some resources have the added cost per use. Computation resources only incur a flat-rate cost. DB service has the flat-rate cost plus the usage cost that is charged per transaction. Storage service has the flat-rate cost plus the cost that depends on how much storage is being used. Additionally, multi-cloud resource allocations will pay the data transfer cost for any communication between clouds. We take all these factors into account when we calculate the total cost for allocating compute, DB, and storage resources. We assume that we calculate this cost for one month, and we assume some number of DB transactions and some amount of data stored in cloud storage for that month. We do not directly include data transfer cost in our cost calculations. Instead, we calculate monthly savings between the best multi-cloud allocation and the best single-cloud allocation, and we then calculate the maximum amount of data that could be transferred between clouds for that amount of money. In many cases, we believe that this amount of data is much higher than cloud applications would transfer today between various resources in one month, thus making multi-cloud allocation a better choice than allocation on any single cloud.

3.2.2. Resource allocation goal

We assume that an application provider has a time constraint for each transaction, which denotes the longest transaction completion time that the application's users are willing to tolerate. Each class of cloud applications has different priorities with regard to resource allocation goals. Interactive services, such as search, strive to offer fastest response to a user's request, regardless of the cost of the resources needed to meet this goal. These services will have strict performance requirements for their resource allocation. Non-interactive services, such as high-performance computing, may be interested in a resource allocation on cheaper machines, even if this prolongs the time a user needs to wait for a response. In our evaluation, we assume that the maximum allowable transaction completion times are the SLA constraints on the resource allocation problem. We then run our resource allocation algorithm for several transaction completion targets, spanning a wide range of values. For each target, we identify an allocation strategy that meets all of the SLA constraints and comes at the lowest price.

3.2.3. Resource allocation

We formulate the resource allocation problem as the problem of finding a set of resources for a given application workflow that meets the set of SLA constraints and that achieves the following:

$$\min\ C_{total}\Gamma, \quad (2)$$

where C_{total} is the total monthly cost for the computing resources required to complete all of micro-tasks in the workflow, Γ is 1 if $\forall transaction \in w, T^{transaction} \leqslant SLA^{transaction}$ and ∞ otherwise, w is the set of all transactions in the application workflow, and $T^{transaction}$ is the completion time of *transaction*.

It may seem that simply selecting each service from a provider with minimal cost will lead to the optimum resource allocation. However, there are additional constraints that make this problem non-linear. First, the time for each transaction must meet its SLA constraint. Second, we assume that a cloud application will only use one cloud DB service for all its database micro-tasks, and, similarly, it will only use one cloud storage service for all its storage micro-tasks. This is a reasonable assumption as users tend to allocate a large DB for cloud deployment, and the storage cost per data unit declines as more volume is used. This constraint makes some resource combinations impossible, further adding to the non-linearity of the problem we seek to solve.

We find the best resource allocation in the following manner. First, we compute all possible resource allocations for each transaction that meet their respective SLA constraints. Then, we order these solutions by cost within each transaction. We start building the final allocation by exhaustively combining allocations for each transaction which also meet our one-DB/one-storage constraint. This exhaustive search progresses from the cheapest to the most expensive solution and stops as soon as we find a viable solution that meets all of the constraints.

Although our approach is based on an exhaustive search, the algorithm finds the solution fairly quickly,

Please cite this article in press as: S.S. Woo, J. Mirkovic, Optimal application allocation on multiple public clouds, Comput. Netw. (2014), http://dx.doi.org/10.1016/j.comnet.2013.12.001

because the different compute, DB, and storage machine types, as currently advertised by the cloud providers, is a finite set. In addition, because we used real data measurements taken from [1], our work is not based entirely on simulation results but incorporates the empirical measurement information.

4. Evaluation

In our evaluation, we use the application workflows as defined in Section 3.2.1. We consider four different scenarios, which depend on how the user applications are distributed and deployed over separate compute, DB, and storage services. The first scenario assumes cloud applications use compute, cloud DB, and cloud storage services. Typically, this scenario is desirable when application providers want to clearly separate compute, DB, and storage tasks, e.g., for reliability. For example, managing and storing user session data in a separate DB and storage from the Web server ensures that rebooting a Web server would not disrupt current user sessions. The second scenario is when cloud applications are deployed over compute and cloud storage services only, and DB tasks are run on compute machines. This scenario applies when application providers want to simultaneously manage compute and DB functions for faster access, while storing data in dedicated cloud storage service for fault tolerance. The third scenario assumes cloud applications use compute and cloud DB services only, and storage tasks are managed on compute machines. This scenario applies when application providers want to have fast access to stored data and can implement good solutions for fault tolerance themselves, and they want to use cloud DB services for more complex relational DB operations or for dedicated and faster (*key*, *value*) pair lookup. The fourth scenario assumes cloud applications use only compute machines for computation, DB, and storage tasks for fast access, both to database information and to stored data. This scenario applies to applications with small user base and simple operations.

4.1. Evaluation details

For each scenario, we evaluate e-business, game, storage, multimedia, and scientific computing workloads as described in Section 3.2.1. For each transaction in a workflow, we calculate the minimum time needed to complete it using data from Fig. 1. We then use the following three SLA constraints: (1) minimum + 0.5 s, (2) minimum + 1 s and (3) minimum + 2 s.

Our results are shown in Tables 1–3. Each table shows the best allocation (all SLA constraints are met at the lowest price) for each target, the best single-cloud allocation for each target, and the monthly cost of each test case. We also show which allocation scenarios manage to produce a solution that satisfies the SLA constraints. Since the monthly costs and savings depend on the number of DB transactions and the amount of cloud storage used in a month, we show these for the following combinations of parameters: (1) 100 K DB transactions and 1 TB of cloud storage (light load), (2) 1000 K DB transactions and 10 TB of cloud storage (medium load), and (3) 1000 K DB transactions and 100 TB of cloud storage (heavy load). We assume the round-trip time between clouds is $T_{ic} = 100$ ms which is added in the completion time simulation for multi-cloud. In reality, large clouds are heavily geographically distributed, and the inter-cloud round-trip time should often be lower than 100 ms.

Table 1
Scenario 1: Evaluation results (total monthly cost in dollars) with compute, cloud DB and cloud storage services (Provider C is not available as a single provider).

Application	min + 0.5 s	min + 1 s	min + 2 s
ebus	$Cost_{1MC}$ = 2231 $Cost_{2MC}$ = 2933 $Cost_{3MC}$ = 8963 $Cost_{1A}$ = 4524 $Cost_{2A}$ = 5298 $Cost_{3A}$ = 12,198	$Cost_{1MC}$ = 2231 $Cost_{2MC}$ = 2933 $Cost_{3MC}$ = 8963 $Cost_{1A}$ =4279 $Cost_{2A}$ = 5053 $Cost_{3A}$ = 11,953	$Cost_{1MC}$ = 2231 $Cost_{2MC}$ = 2231 $Cost_{3MC}$ = 8963 $Cost_{1A}$ = 3545 $Cost_{2A}$ = 4319 $Cost_{3A}$ = 11,219
game	$Cost_{1MC}$ = 2069 $Cost_{2MC}$ = 2762 $Cost_{3MC}$ = 8792 $Cost_{1A}$ = 4034 $Cost_{2A}$ = 4808 $Cost_{3A}$ = 11,708	$Cost_{1MC}$ = 2069 $Cost_{2MC}$ = 2762 $Cost_{3MC}$ = 8792 $Cost_{1A}$ =3789 $Cost_{2A}$ = 4563 $Cost_{3A}$ = 11,463	$Cost_{1MC}$ = 2069 $Cost_{2MC}$ = 2762 $Cost_{3MC}$ = 8792 $Cost_{1A}$ = 3361 $Cost_{2A}$ = 4135 $Cost_{3A}$ = 11,035
storage	$Cost_{1MC}$ = 2322 $Cost_{2MC}$ = 3096 $Cost_{3MC}$ = 9996 $Cost_{1A}$ = 3300 $Cost_{2A}$ = 4073 $Cost_{3A}$ = 10,973	$Cost_{1MC}$ = 2322 $Cost_{2MC}$ = 3096 $Cost_{3MC}$ = 9996 $Cost_{1A}$ = 3300 $Cost_{2A}$ = 4073 $Cost_{3A}$ = 10,973	$Cost_{1MC}$ = 2307 $Cost_{2MC}$ = 2899 $Cost_{3MC}$ = 8497 $Cost_{1B}$ = 2933 $Cost_{2B}$ = 3707 $Cost_{3B}$ = 10,607
multimed	$Cost_{1MC}$ = 2518 $Cost_{2MC}$ = 3301 $Cost_{3MC}$ = 10,201 $Cost_{1A}$ = 5993 $Cost_{2A}$ = 6767 $Cost_{3A}$ = 13,667	$Cost_{1MC}$ = 2518 $Cost_{2MC}$ = 3301 $Cost_{3MC}$ = 10,201 $Cost_{1A}$ = 4769 $Cost_{2A}$ = 5543 $Cost_{3A}$ = 12,443	$Cost_{1MC}$ = 2502 $Cost_{2MC}$ = 3104 $Cost_{3MC}$ = 8702 $Cost_{1A}$ = 4218 $Cost_{2A}$ = 4992 $Cost_{3A}$ = 11,892
sci comp	$Cost_{1MC}$ = 2409 $Cost_{2MC}$ = 3183 $Cost_{3MC}$ = 10201 A cannot achieve	$Cost_{1MC}$ = 2387 $Cost_{2MC}$ = 3161 $Cost_{3MC}$ = 10201 $Cost_{1A}$ = 4830 $Cost_{2A}$ = 5604 $Cost_{3A}$ = 12,504	$Cost_{1MC}$ = 2387 $Cost_{2MC}$ = 3161 $Cost_{3MC}$ = 8702 $Cost_{1A}$ = 4218 $Cost_{2A}$ = 4992 $Cost_{3A}$ = 11,892

Scenario 1: Table 1 shows the detailed cost of the best multi-cloud allocation, and the best single-cloud allocations for all three sets of SLA constraints. In Table 1, $Cost_{ab}$ denotes the total monthly operating cost of load a where 1 is the light, 2 is the medium and 3 is the heavy load, and $b \in$ {MC (multi-cloud), A, B, C}.

In this scenario, multi-cloud allocations always have the lower cost than the best single-cloud allocation for all considered cloud applications. Multi-cloud allocations achieve a net-cost savings of $970–3400 per month compared to single-provider allocations. For many of the SLA constraints and application workflows, only allocations on Provider A produce a viable single-cloud solution. Provider B's computation performance is so low for some micro-tasks that it does not even satisfy the relaxed "min + 2 s" constraint. Also, Provider C does not offer DB services, and thus cannot produce viable single-cloud solutions in this scenario. In the scientific computing case, only the multi-cloud allocation provides a viable solution, even with additional inter-cloud delay. Hence, we can

Please cite this article in press as: S.S. Woo, J. Mirkovic, Optimal application allocation on multiple public clouds, Comput. Netw. (2014), http://dx.doi.org/10.1016/j.comnet.2013.12.001

Table 2
Scenario 2: Evaluation results (total monthly cost in dollars) with compute and cloud storage services.

Application	min + 0.5 s	min + 1 s	min + 2 s
ebus	$Cost_{1MC}$ = 2066 $Cost_{2MC}$ = 2750 $Cost_{3MC}$ = 8780 $Cost_{1A}$ = 4944 $Cost_{2A}$ = 5709 $Cost_{3A}$ = 12,609 $Cost_{1C}$ = 2315 $Cost_{2C}$ = 2899 $Cost_{3C}$ = 8496	$Cost_{1MC}$ = 2066 $Cost_{2MC}$ = 2750 $Cost_{3MC}$ = 8780 $Cost_{1A}$ = 4454 $Cost_{2A}$ = 5219 $Cost_{3A}$ = 12,119 $Cost_{1C}$ = 2315 $Cost_{2C}$ = 2899 $Cost_{3C}$ = 8496	$Cost_{1MC}$ = 2066 $Cost_{2MC}$ = 2750 $Cost_{3MC}$ = 8780 $Cost_{1A}$ = 3906 $Cost_{2A}$ = 4669 $Cost_{3A}$ = 11,569 $Cost_{1C}$ = 2315 $Cost_{2C}$ = 2899 $Cost_{3C}$ = 8496
game	$Cost_{1MC}$ = 2002 $Cost_{2MC}$ = 2686 $Cost_{3MC}$ = 8716 $Cost_{1A}$ = 4026 $Cost_{2A}$ = 4791 $Cost_{3A}$ = 11,691 $Cost_{1C}$ = 2250 $Cost_{2C}$ = 2834 $Cost_{3C}$ = 8432	$Cost_{1MC}$ = 2002 $Cost_{2MC}$ = 2686 $Cost_{3MC}$ = 8716 $Cost_{1A}$ = 3781 $Cost_{2A}$ = 4546 $Cost_{3A}$ = 11,446 $Cost_{1C}$ = 2250 $Cost_{2C}$ = 2834 $Cost_{3C}$ = 8432	$Cost_{1MC}$ = 2002 $Cost_{2MC}$ = 2686 $Cost_{3MC}$ = 8716 $Cost_{1A}$ = 3353 $Cost_{2A}$ = 4118 $Cost_{3A}$ = 11,018 $Cost_{1C}$ = 2250 $Cost_{2C}$ = 2834 $Cost_{3C}$ = 8432
storage	$Cost_{1MC}$ = 2244 $Cost_{2MC}$ = 3009 $Cost_{3MC}$ = 9909 $Cost_{1A}$ = 3230 $Cost_{2A}$ = 3995 $Cost_{3A}$ = 10,895 C cannot achieve	$Cost_{1MC}$ = 2244 $Cost_{2MC}$ = 3009 $Cost_{3MC}$ = 9909 $Cost_{1A}$ = 3230 $Cost_{2A}$ = 3995 $Cost_{3A}$ = 10,895 C cannot achieve	$Cost_{1MC}$ = 2229 $Cost_{2MC}$ = 2812 $Cost_{3MC}$ = 8410 $Cost_{1A}$ = 2863 $Cost_{2A}$ = 3628 $Cost_{3A}$ = 10,528 $Cost_{1C}$ = 2229 $Cost_{2C}$ = 2812 $Cost_{3C}$ = 8410
multimed	$Cost_{1MC}$ = 2341 $Cost_{2MC}$ = 3106 $Cost_{3MC}$ = 10,027 $Cost_{1A}$ = 6107 $Cost_{2A}$ = 6872 $Cost_{3A}$ = 13,772 C cannot achieve	$Cost_{1MC}$ = 2341 $Cost_{2MC}$ = 3106 $Cost_{3MC}$ = 10,006 $Cost_{1A}$ = 4883 $Cost_{2A}$ = 5648 $Cost_{3A}$ = 12,548 C cannot achieve	$Cost_{1MC}$ = 2326 $Cost_{2MC}$ = 2909 $Cost_{3MC}$ = 10,006 $Cost_{1A}$ = 4332 $Cost_{2A}$ = 5907 $Cost_{3A}$ = 11,997 $Cost_{1C}$ = 2326 $Cost_{2C}$ = 2909 $Cost_{3C}$ = 8507
sci comp	$Cost_{1MC}$ = 2363 $Cost_{2MC}$ = 3128 $Cost_{3MC}$ = 10,027 A cannot achieve C cannot achieve	$Cost_{1MC}$ = 2341 $Cost_{2MC}$ = 3106 $Cost_{3MC}$ = 10,006 $Cost_{1A}$ = 4944 $Cost_{2A}$ = 5709 $Cost_{3A}$ = 12,609 C cannot achieve	$Cost_{1MC}$ = 2341 $Cost_{2MC}$ = 3106 $Cost_{3MC}$ = 10,006 $Cost_{1A}$ = 4332 $Cost_{2A}$ = 5097 $Cost_{3A}$ = 11,997 $Cost_{1C}$ = 2326 $Cost_{2C}$ = 2909 $Cost_{3C}$ = 8507

clearly observe the benefits of multi-cloud allocation. It not only provides savings in terms of net cost, but it is also able to meet the stringent completion time requirements for complex cloud applications by combining the best machines from multiple public cloud providers.

In *ebus* and *game* cases, 20–50% of relative operational cost savings can be achieved with multi-cloud allocations and 8–29% cost savings for the *storage* case compared to the best single provider allocations. For each *multimed* and *sci comp* case, 15–50% savings and 19–50% of operational cost savings can be achieved with multi-cloud based on the simulation result in Table 1.

Scenario 2: We now examine the alternative implementation of workflows as defined in Section 3.2.1, where the database is implemented on computation resources, instead of using a cloud DB solution. We replace each `query`, `get` and `put` micro-task with the `mem` micro-task. The time for the `mem` micro-task is calculated by taking the time for the `mem_small` micro-task, which measures access to 2 MB of data, and dividing it by 2,000. This approximates the time to read or write 1 KB of data into memory. We assume that all DB transactions transfer up to 1 KB of data. All results are presented in Table 2.

In this scenario, for some application workloads and some SLA constraints, multi-cloud allocation is either the only one that produces a viable solution, or it produces a much cheaper solution than single-cloud allocations. For other application workloads and SLA constraints, however, the best single-cloud allocation using Provider C is cheaper or costs the same as the best multi-cloud allocation. For example, for an e-business application, multi-cloud performs best for (1) 100 K DB transactions/1 TB (light load) and (2) 1000 K DB transactions/10 TB storage (medium load) yielding the relative cost savings from 5% to 50%

Please cite this article in press as: S.S. Woo, J. Mirkovic, Optimal application allocation on multiple public clouds, Comput. Netw. (2014), http://dx.doi.org/10.1016/j.comnet.2013.12.001

Table 3
Scenario 3: Evaluation results (total monthly cost in dollars) with compute and cloud DB services (Provider C is not available as a single provider).

Application	min + 0.5 s	min + 1 s	min + 2 s
ebus	$Cost_{1MC}$ = 328	$Cost_{1MC}$ = 328	$Cost_{1MC}$ = 328
	$Cost_{2MC}$ = 346	$Cost_{2MC}$ = 346	$Cost_{2MC}$ = 346
	$Cost_{3MC}$ = 346	$Cost_{3MC}$ = 346	$Cost_{3MC}$ = 346
	$Cost_{1A}$ = 2456	$Cost_{1A}$ =2211	$Cost_{1A}$ = 1477
	$Cost_{2A}$ = 2456	$Cost_{2A}$ = 2220	$Cost_{2A}$ = 1486
	$Cost_{3A}$ = 2456	$Cost_{3A}$ = 2220	$Cost_{3A}$ = 1486
game	$Cost_{1MC}$ = 165	$Cost_{1MC}$ = 165	$Cost_{1MC}$ = 165
	$Cost_{2MC}$ = 174	$Cost_{2MC}$ = 174	$Cost_{2MC}$ = 174
	$Cost_{3MC}$ = 174	$Cost_{3MC}$ = 174	$Cost_{3MC}$ = 174
	$Cost_{1A}$ = 1967	$Cost_{1A}$ =1722	$Cost_{1A}$ = 1293
	$Cost_{2A}$ = 1976	$Cost_{2A}$ = 1731	$Cost_{2A}$ = 1302
	$Cost_{3A}$ = 1976	$Cost_{3A}$ = 1731	$Cost_{3A}$ = 1302
storage	$Cost_{1MC}$ = 154	$Cost_{1MC}$ = 154	$Cost_{1MC}$ = 154
	$Cost_{2MC}$ = 163	$Cost_{2MC}$ = 163	$Cost_{2MC}$ = 163
	$Cost_{3MC}$ = 163	$Cost_{3MC}$ = 163	$Cost_{3MC}$ = 163
	$Cost_{1A}$ = 1232	$Cost_{1A}$ = 1232	$Cost_{1B}$ = 865
	$Cost_{2A}$ = 1241	$Cost_{2A}$ = 1241	$Cost_{2B}$ = 874
	$Cost_{3A}$ = 1241	$Cost_{3A}$ = 1241	$Cost_{3B}$ = 874
multimed	$Cost_{1MC}$ = 349	$Cost_{1MC}$ = 349	$Cost_{1MC}$ = 349
	$Cost_{2MC}$ = 367	$Cost_{2MC}$ = 367	$Cost_{2MC}$ = 367
	$Cost_{3MC}$ = 367	$Cost_{3MC}$ = 367	$Cost_{3MC}$ = 367
	A cannot achieve	$Cost_{1A}$ = 3191	$Cost_{1A}$ = 2150
		$Cost_{2A}$ = 3191	$Cost_{2A}$ = 2159
		$Cost_{3A}$ = 3191	$Cost_{3A}$ = 2159
sci comp	$Cost_{1MC}$ = 251	$Cost_{1MC}$ = 240	$Cost_{1MC}$ = 240
	$Cost_{2MC}$ = 260	$Cost_{2MC}$ = 249	$Cost_{2MC}$ = 249
	$Cost_{3MC}$ = 260	$Cost_{3MC}$ = 249	$Cost_{3MC}$ = 249
	A cannot achieve	$Cost_{1A}$ = 2885	$Cost_{1A}$ = 2273
		$Cost_{2A}$ = 2894	$Cost_{2A}$ = 2282
		$Cost_{3A}$ = 2894	$Cost_{3A}$ = 2282

compared to Provider C. As we can see from Table 2, $Cost_{1MC}$ and $Cost_{2MC}$ are the lowest across all SLAs. However, Provider C's cost, $Cost_{3C}$, is the best for (3) 1000 K DB transactions and 100 TB storage (heavy load) with 3–4% cost savings over Multi-Cloud approach. Therefore, we can clearly observe the trade-off between SLA and cost among offered services from different providers for various types of applications. In all of the cases where single-cloud allocation on cloud C outperforms a multi-cloud one, the cost of using a cloud storage service is much higher than the cost of using compute services and the SLA restrict is not stringent, and this dominates the overall application deployment cost. Therefore, we can conclude that for applications that require large storage space, a single-provider allocation may be cheaper than a multi-cloud allocation. This is especially true for the scientific computing case, which utilizes tightly-coupled heavy storage and compute resources, and does not have stringent task completion requirements.

Scenario 3: Next, we examine the alternative implementation of workflows as defined in Section 3.2.1, where the storage is implemented on computation resources, instead of using a dedicated cloud storage solution. We replace each `st_1KB_up`/`st_1KB_down` micro-task with the `io_1KB` micro-task, and we replace each `st_10MB_up`/`st_10MB_down` micro-task with the `io_10MB` micro-task. The time for the `io_1KB` micro-task is calculated by taking the maximum of the `io_read`, `io_write` and `io_readwrite` micro-tasks, which measure the transfer of 100 MB of data, and dividing it by 100,000. Similarly, the time for the `io_10MB` micro-task is calculated by taking the maximum of the `io_read`, `io_write` and `io_readwrite` micro-tasks, which measure the transfer of 100 MB of data, and dividing it by 10. All of the results are presented in the table.

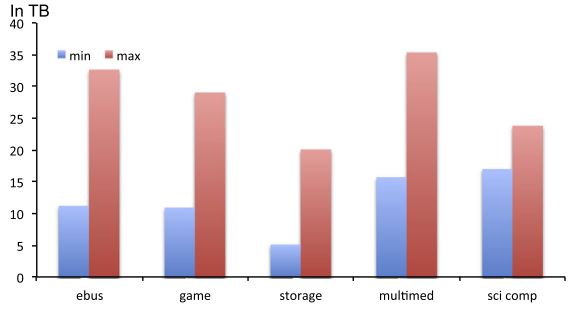
In this scenario, multi-cloud allocations provide significant cost savings for all test cases over all SLAs. The savings range from $700 to $2800 per month. In the multi-media and scientific computing cases, the only viable solution is the multi-cloud one. Often, the resulting resource allocation chooses the best compute machines from C and DB services from A. This scenario clearly demonstrates the benefits of the multi-cloud approach where the best resources are pooled from multiple providers – the obtained performance is potentially better, and the overall cost is clearly substantially cheaper. In particular, in all cases, Multi-Cloud approach provides 80–91% relative cost savings over the best single Provider A as shown in Table 3.

Scenario 4: Finally, we examine the alternative implementation of workflows as defined in Section 3.2.1, where both DB and storage solutions are implemented on compute resources. We replace each `query`, `get` and `put` micro-task with the `mem` micro-task, each `st_1KB_up`/`st_1KB_down` micro-task with the `io_1KB` micro-task, and each `st_10MB_up`/`st_10MB_down` micro-task with the `io_10MB` micro-task. In this scenario, the best solution is always the single-cloud one, regardless of the application workload and SLA constraints. This is because Provider C offers the fastest compute machines at the lowest price. Our algorithm chooses all of resources from Provider C. Even where a micro-task's completion time differs between providers, the difference is not large enough to warrant the added inter-cloud delay that a multi-cloud solution would bring. Therefore, if application needs to be deployed using only compute services, it pays off to stay with a single provider.

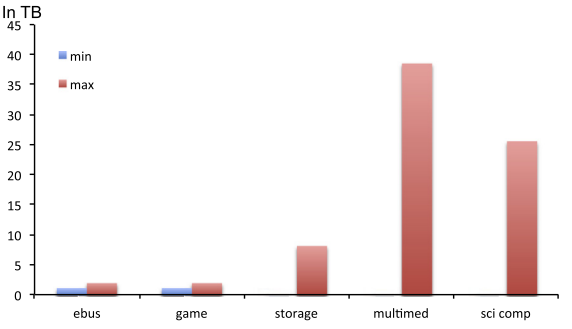
4.2. Inter-cloud data traffic

In the current cloud traffic pricing model, users pay for egress data traffic from a public cloud per bandwidth usage, but ingress traffic to a cloud is free. For the multi-cloud allocations proposed in this paper, an application provider would benefit from cheaper resources but would need to pay the cost for inter-cloud data transfer each time data goes between different public clouds. In this section, we quantify the allowable minimum and maximum inter-cloud traffic that could be supported with the cost savings achieved from the multi-cloud allocation. Using published prices for egress data transfer [20–22], we estimate the minimum and maximum inter-cloud data transfer budget by taking the difference between the best cost of a single provider and the cost of multi-cloud allocations, and translating into an allowable extra network traffic in TB. This is the "break-even" amount of traffic. Anything less would still result in some cost savings for the application provider, while anything more would mean that the application provider needs to pay more for a multi-cloud allocation than if they hosted their application on a single

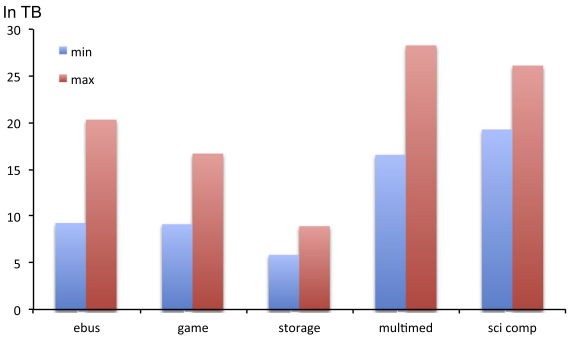
Please cite this article in press as: S.S. Woo, J. Mirkovic, Optimal application allocation on multiple public clouds, Comput. Netw. (2014), http://dx.doi.org/10.1016/j.comnet.2013.12.001



(a) Scenario 1: compute, storage, and DB resources



(b) Scenario 2: compute and storage resources



(c) Scenario 3: compute and DB resources

Fig. 3. Allowable extra data traffic in TB per month between multi-cloud and the best single provider for (a) compute, storage, and DB solution, (b) compute and storage solution, and (c) compute and DB solution.

cloud. The minimum and maximum amount of this "break-even" inter-cloud data that can be transferred in a month is shown in Fig. 3 for the Scenarios 1, 2 and 3. Since there is no advantage to using multi-cloud resource allocation over single-cloud allocation in Scenario 4, we do not show it in this figure.

We observe that the "break-even" amount of traffic is significant in Scenarios 1 and 3, across all tested cloud applications. This is because these two Scenarios use compute and DB services, where there is a significant difference in performance and price between Providers A (best for DB, poor for compute) and C (best for compute, no DB services available). We expect that realistic cloud applications would easily transfer less than this "break-even" amount of traffic between their components in a month, making the multi-cloud resource allocation a profitable choice for application providers. On the other hand, in Scenario 2, the "break-even" amount of traffic is significant only for the highest workload in *multimed*, *storage* and *sci comp* applications. These are the applications that heavily use storage service, transferring large blobs, which dominates the allocation cost. Since storage services are similarly priced between providers, gains of multi-cloud allocation are evident only when storage use is large, exposing the price differences. Again, we expect that for realistic cloud applications which provide cloud storage, multimedia or scientific computing services, the gains of the multi-cloud resource allocation would be more than sufficient to cover the cost of inter-cloud data transfer.

5. Conclusion

With increased differentiation in cloud providers' service offers, allocation of application components on multiple clouds becomes an attractive option. Our evaluation shows that such allocation provides significant cost savings over single-cloud allocations in a variety of realistic cloud use scenarios. While multi-cloud allocation carries additional delay and cost due to inter-cloud communication, its performance gains and cost savings are still significant. In particular, we have observed that the cost benefit of multi-cloud is substantial when applications are deployed over separate and dedicated compute, DB, and storage services.

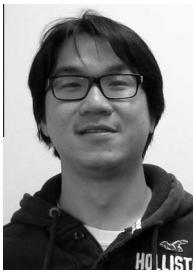
Further, application providers may have different priorities – some may want to minimize task completion time at any cost (e.g., a search service), while others may want to minimize the cost and accept higher service delays (e.g., an image processing or a genome sequencing service). In mobile cloud applications, compute/storage intensive tasks can be offloaded to mobile devices instead of to a cloud infrastructure if bandwidth is very limited. Our proposed framework is general and can be applied to multiple optimization goals. Our allocation algorithm exposes

Please cite this article in press as: S.S. Woo, J. Mirkovic, Optimal application allocation on multiple public clouds, Comput. Netw. (2014), http://dx.doi.org/10.1016/j.comnet.2013.12.001

sufficient information to let application providers find their own "sweet spot" in the price/performance trade-off.

References

[1] A. Li, X. Yang, S. Kandula, M. Zhang, Cloudcmp: comparing public cloud providers, in: Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement, ACM, 2010, pp. 1–14.
[2] Netflix Hit by Outage, Blames Amazon, The Wall Street Journal, December 25, 2012.
[3] Z. Wu, H.V. Madhyastha, Understanding the latency benefits of multi-cloud webservice deployments, ACM SIGCOMM Comput. Commun. Rev. (2013) 13–20.
[4] A. Celesti, F. Tusa, M. Villari, A. Puliafito, How to enhance cloud architectures to enable cross-federation, in: Proceedings of the 3rd IEEE International Conference on Cloud Computing (CLOUD), 2010, pp. 337–345.
[5] B. Rochwerger, D. Breitgand, D. Hadas, I. Llorente, R. Montero, P. Massonet, E. Levy, A. Galis, M. Villari, Y. Wolfsthal, et al., The reservoir model and architecture for open federated cloud computing, IBM J. Res. Dev. 53 (4) (2009) 4–11.
[6] L.F. Bittencourt, E.R. Madeira, N.L. da Fonseca, Impact of communication uncertainties on workflow scheduling in hybrid clouds, in: Proceedings of the IEEE Global Communications Conference (GLOBECOM), 2012, pp. 1623-1628.
[7] M. Bjorkqvist, L.Y. Chen, W. Binder, Cost-driven service provisioning in hybrid clouds, in: Proceedings of the 5th IEEE International Conference on Service-Oriented Computing and Applications (SOCA), 2012, pp. 1–8.
[8] P. Mell, T. Grance, The NIST Definition of Cloud Computing (draft), vol. 80, NIST Special Publication, 2011, p. 145.
[9] M. Hajjat, X. Sun, Y.-W.E. Sung, D. Maltz, S. Rao, K. Sripanidkulchai, M. Tawarmalani, Cloudward bound: planning for beneficial migration of enterprise applications to the cloud, ACM SIGCOMM Comput. Commun. Rev. 40 (2010).
[10] M. Hajjat, D. Maltz, S. Rao, K. Sripanidkulchai, et al., Dealer: application-aware request splitting for interactive cloud applications, in: Proceedings of the 8th International Conference on Emerging Networking Experiments and Technologies, ACM, 2012, pp. 157–168.
[11] C. Vecchiola, S. Pandey, R. Buyya, High-performance cloud computing: a view of scientific applications, in: Proceedings of the IEEE 10th International Symposium on Pervasive Systems, Algorithms, and Networks (ISPAN), 2009, pp. 4–16.
[12] H. Zhang, G. Jiang, K. Yoshihira, H. Chen, A. Saxena, Intelligent workload factoring for a hybrid cloud computing model, in: Proceedings of the IEEE International Workshop on Cloud Services (IWCS), 2009, pp. 701–708.
[13] R. Buyya, R. Ranjan, R.N. Calheiros, Intercloud: utility-oriented federation of cloud computing environments for scaling of application services, in: Algorithms and Architectures for Parallel Processing, Springer, 2010, pp. 13–31.
[14] A. Li, X. Zong, S. Kandula, X. Yang, M. Zhang, Cloudprophet: towards application performance prediction in cloud, ACM SIGCOMM Comput. Commun. Rev. 41 (2011) 426–427.
[15] Amazon.com. <http://www.amazon.com>.
[16] World of Warcraft Online. <https://us.battle.net>.
[17] Dropbox. <http://www.dropbox.com>.
[18] Youtube. <http://www.youtube.com>.
[19] Penguin Computing. <http://www.penguincomputing.com>.
[20] Amazon Web Services Pricing information. <http://aws.amazon.com/ec2/pricing/>.
[21] Windows Azure Pricing information. <http://www.windowsazure.com/en-us/pricing/calculator/>.
[22] Google App Engine Pricing information. <https://cloud.google.com/pricing/>.



Simon Woo is a Ph.D. candidate at USC/ISI supervised by Prof. Jelena Mirkovic. His current research interests include cloud computing, network security, and cybersecurity education. Also, he is a Member of Technical Staff at the Jet Propulsion Laboratory (JPL), Pasadena, CA. As a technologist, he conducts networking and cybersecurity research and development. He has been involved in network and protocol simulation for NASA Constellation Program and time synchronization and QoS services development for Deep Space Networks. Prior to joining JPL in 2005, he interned at Entropic Communications in San Diego to evaluate the protocol performance. Also, he worked at Intel Corp. He earned his BSEE degree from Univ. of Washington, Seattle, MSEE degree from Univ. of California, San Diego, specializing in communication theory and systems, and MSCS degree from USC.

Jelena Mirkovic is Computer Scientist at USC/ISI and research faculty at USC. She received her MS and PhD from UCLA, working in the LASR group, lead by Prof. Peter Reiher. She received BS in Computer Science and Engineering from School of Electrical Engineering, University of Belgrade, Serbia. Her research interests span networking and security fields. Her current research is focused on several network security problems: botnets, denial-of-service attacks, and IP spoofing. Additionally, she is interested in methodologies for conducting security experiments and she is working with colleagues at USC/ISI on improving DeterLab testbed.

Please cite this article in press as: S.S. Woo, J. Mirkovic, Optimal application allocation on multiple public clouds, Comput. Netw. (2014), http://dx.doi.org/10.1016/j.comnet.2013.12.001