To Move or Not To Move: The Economics of Cloud Computing

Byung Chul Tak Bhuvan Urgaonkar Anand Sivasubramaniam

Computer Systems Laboratory
Department of Computer Science and Engineering
The Pennsylvania State University, University Park, PA 16802
{tak, bhuvan, anand}@cse.psu.edu

Technical Report CSE-11-002
March 2011

Abstract

Cloud-based hosting promises cost advantages over conventional in-house (on-premise) application deployment. One important question when considering a move to the cloud is whether it makes sense for 'my' application to migrate to the cloud. This question is challenging to answer due to following reasons. Although many potential benefits of migrating to the cloud can be enumerated, some benefits may not apply to my application. Also, there can be multiple ways in which an application might make use of the facilities offered by cloud providers. Answering these questions requires an in-depth understanding of the cost implications of all the possible choices specific to 'my' circumstances. In this study We identify an initial set of key factors affecting the costs of a deployement choice. Using benchmarks representing two different applications (TPC-W and TPC-E) we investigate the evolution of costs for different deployment choices. We show that application characteristics such as workload intensity, growth rate, storage capacity and software licensing costs produce complex combined effect on overall costs. We also discuss issues regarding workload variance and horizontal partitioning.

1 Introduction

Cloud-based hosting promises several advantages over conventional in-house (on-premise) application deployment. First, it claims to offer ease-of-management (although arguments against this have also been made [7]): since the cloud provider assumes management-related responsibilities involving procurement, upgrades, maitenance of hardware/software and supporting infrastructure, the customer is relieved of this burden and can focus on its core expertise. Second, it offers Cap-ex savings: cloud-based hosting eliminates the need for commitment in the purchase of IT equipment and other infrastructure; this may translate into a lowering the business entry barrier for some organizations. Third, it offers Op-ex reduction: this occurs due to the elimination of the need to pay for adminitrator salaries, utility electricy bills, real-estate rents/mortgages, etc. One aspect of Op-ex savings that is highly touted is that this model can allow a cus-

tomer's Op-ex to closely match its evolving resource needs (via usage-based charging) as opposed to being dictated by its worst-case needs (for which an in-house solution must provision).

The quintessential question when considering a move to the cloud is: *should I migrate my application to the cloud?* Whereas there have been several studies into this question, there is no consensus yet on whether the cost of cloud-based hosting is attractive enough compared to in-house hosting [19]. There are several aspects to this basic question that must be considered. First, although many potential benefits of migrating to the cloud can be enumerated for the general case, some benefits may not apply to my application. For example, benefits related to lowered entry barrier may not apply as much to an organization with a pre-existing infrastructural and administrative base. As another example, the benefits of pay-per-use are less attractive for a well-provisioned application whose workload doesnot vary much. Second, there can be multiple ways in which an application might make use of the facilties offered by a cloud provider. For example, using the cloud need not preclude a continued use of the in-house infrastructure. The most cost-effective approach for an organization might, in fact, involve a combination of cloud and in-house resources rather than choosing one over the other. Third, not all elements of the overall cost consideration may be equally easy to quantify. For example, the hardware resource needs and associated costs may be reasonably straightforward to estimate and compare across different hosting options. On the other hand, labor costs may be significantly more complicated: e.g., how should the overall administrators' salaries in an organization be apporitioned among various applications that they manage? As another example, in a cloud-based hosting, how much effort and cost is involved in migrating an application to the cloud? Answering these questions requires an in-depth understanding of the cost implications of all the possible choices specific to *my* circumstances. Given that these answers can vary widely across applications, organizations, and cloud providers, we believe the best way is to explore various applications case-by-case in an attempt to draw generalities or useful rule-of-thumbs.

Research Contributions. We make the following contributions towards answering the questions posed above.

- *Identification and Classification of Cost Contributors*: We identify a comprehensive set of factors affecting the costs of a deployement choice (in-house, cloud, and combinations). We classify these as "quantifiable" and "less quantifiable" based on how amenable they are to precise quantification. We also classify these into the "direct" and "indirect" categories: the former contributes solely towards the costs of my application (e.g., server costs) while the latter contributes to a group and requires that its contribution to my application be teased out (e.g., cooling costs).

- *Identification of Deployment Choices*: Besides the two extreme deployment choices of pure in-house and pure cloud-based hosting available to an application, we identify a spectrum of hybrid choices that can offer the best of both worlds. Our hybrid choices capture both "vertial" and "horizontal" ways of partitioning an application, each with its own pros and cons.

- *Case Studies using NPV-based Cost Analysis*: Using a diverse set of applications (open-source vs. licensed software, interactive vs. throughput-intensive, commercial versus scientific), cloud offerings (IaaS vs. SaaS), and workload characteristics (stagnant vs. growing, relatively constant vs. spikey)

we study the evolution of costs for different deployment choices. We express these evolving costs using the well-regarded Net Present Value (NPV) concept. We conduct two such case-studies (an e-commerce benchmark that uses open-source software and a commerical application that uses Oracle database) for which our analysis offers a number of interesting insights and future directions.

The rest of this paper is organized as follows. Section 2 informs readers with various concepts related to cost analysis and migration, and assumptions we employed in the analysis. We also describe the method we used to determine the required number of hardware units for both in-house and cloud-based options. Section 3 presents the cost analysis using the *quantifiable* and *direct* costs. Then, in Section 3.3, we look at how the workload variance affects the resulf of previous cost analysis. present concluding remarks in Section 4.

2 Background and Overview

2.1 Net Present Value

In financial analysis, investigating the suitibility of an investment involves assessing the overall costs expected to be incurred over its lifetime. We view the decision-making of whether to migrate the application to the cloud as an investment decision problem. The concept of *Net Present Value* (NPV) is popularly used in financial analysis to calculate the profitability of an investment decision over its expected lifetime considering all the cash inflows and outflows. Walker et al. have recently employed this concept in their work, focusing mainly in separately exploring the feasibility of renting computing [13] and storage [14] from the cloud. While we employ the same NPV concept, we go beyond this work: (i) as opposed to comparing rental vs. in-house costs only for a given hardware base, we compare the costs for hosting specific workloads (for which we determine hardware needs via benchmarking), (ii) we incorporate additional costs (in fact, we find that hardware has a minor effect on the overall decision-making) - IO bandwidth, such as software licenses, and electricity, (iii) we study the impact of workload evolution/variance and cloud models (IaaS vs. Saas), and finally (iv) we consider combinations of in-house and cloud hosting. Borrowing existing notation, we define the NPV of an investment choice spanning Y years into the future as:

$$NPV = \sum_{t=0}^{Y-1} \frac{C_t}{(1+r)^t} \tag{1}$$

where r is the *discount rate* and C_t the cost at time t. The role of the discount rate is to capture the phenomenon that the value of a dollar today is worth more than a dollar in the future, with its value decreased by a factor $(1 + r)$ per year.

As a simple example to understand NPV, assuming $r = 5\%$, consider two choices to purchase 10 items, each costing \$1,000 over a one year span: (i) buy all today: NPV=\$10,000, and (ii) buy half today, and half next year: NPV=\$5,000+\$5,000/1.05=\$9,761. The latter is the preferred choice here since it allows us to spend a lower amount than with choice (i) for the same overall purchase. Whereas the NPV defintion can be enhanced to also incorporate the effect of inflation (e.g., in case (ii) we might need more than \$5,000 to buy 5 items a year from now), we assume it to be small and ignore it in our present work.

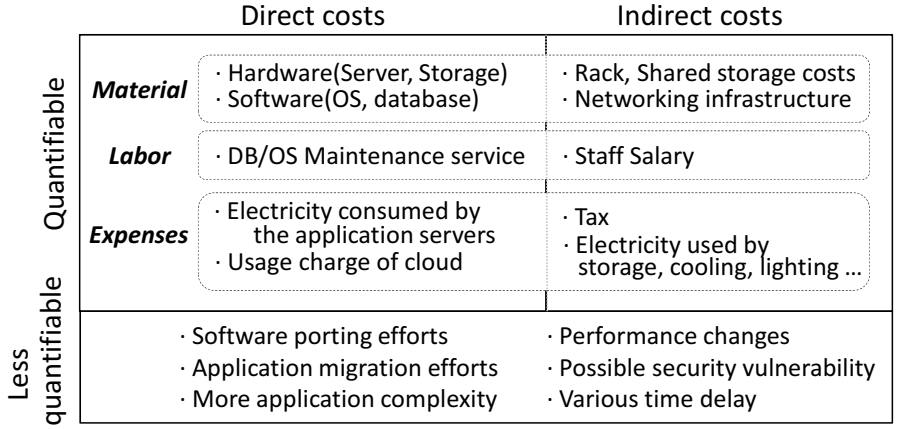


Figure 1: Classification of costs related to migration.

2.2 Cost Components

Fig. 1 presents our classification of costs. Certain cost components are less easy to quantify than others, and we use the phrases "quantifiable" and "less quantifiable" to make this distinction. Examples of less quantifiable costs include effort of migrating an application to the cloud, porting an application to the programming API exposed by a cloud (e.g., as required with Windows Azure), time spent doing the migration/porting, any problems/vulnerabilities that arise due to such porting or migration, etc. Adhering to well-regarded convention in financial analysis, we also employ the classification of costs into the "direct" and "indirect" categories based on their ease of traceability and accounting. If a cost can be clearly traced and accounted to a product/service/personnel, it is a direct cost, else it is an indirect cost. As shown in Fig. 1, examples of direct cost include hardware & software costs; examples of indirect cost include staff salaries. It should be noted that certain costs may be less quantifiable yet direct (e.g., porting an application). Similarly, certain costs may be quantifiable yet indirect (e.g., staff salaries, cooling, etc.) In this work, we restrict our focus to only quantifiable costs and leave less quantifiable costs for future work.

2.3 Application Hosting Choices

Besides pure in-house and pure cloud-based hosting, a number of intermediate/hybrid options have been suggested, and are worth considering [5]. We view these schemes as combinations of different degrees of "vertical" and "horizontal" partitioning of the application. Vertical partitioning splits an application into two subsets (not necessarily mutually exclusive) of components - one is hosted in-house and the other migrated to the cloud - and may be challenging if any porting is required [5]. Horizontal partitioning replicates some components of the application (or the entire application) on the cloud along with suitable workload distribution mechanisms. Such partitioning is already being used as a way to handle unexpected traffic bursts by some businesses (e.g., KBB.com and Domino's Pizza [16]). Such a partitioning scheme must employ mechanisms to maintain consistency among replicas of stateful components (e.g., databases) with associated overheads [1]. Given myriad cloud providers and hosting models (we consider IaaS and SaaS), there can be multiple choices for how a component is migrated to the cloud, each with its own cost implications. In this

[1]Note that pure in-house and pure cloud hosting can be viewed as extreme cases of both these kinds of partitioning.

Migration Type	Stateless Tiers	Database Tier	Possible Deployment Example	
			Non-DB Tier	DB Tier
Fully In-house	(1) In-house (2) Virtualized In-house	(1) Local(In-house) DB server (2) Local database server in virtualized environment	Local servers	Local DB servers
(Vertical) Hybrid migration	In-cloud		EC2 instances	Local DB servers
	(1) In-house (2) Virtualized In-house	(1) Database setup on cloud VM instances (2) Database VM image provided by Cloud (3) SaaS Database Service	Local servers	MySQL installation Amazon RDB AMI Amazon RDS, SQL Azure
Fully In-cloud	In-cloud		EC2 instances	Amazon RDS SQL Azure

Table 1: Hosting options based on vertical migration for an application using some prevelant cloud offerings. We separate out options applicable to the stateless (e.g., Web server) and stateful (e.g., database) components for both in-house and cloud-based hosting. The overall number of hosting options is given by a product of these separate options for components.

work, we choose three such options (in addition to pure in-house and pure cloud-based) that we described next.

There are many ways to migrate the application to the cloud. From the system-topology view point, one can consider either *full* or *hybrid* migration. In the *full* migration, all system nodes move into the cloud. In the *hybrid* migration, subset of the nodes are selected to move to the cloud. The advantages of the *hybrid* migration is explained in *Cloudward Bound* [5]. Authors describe the complexities of current enterprise multi-tiered applications and challenges involved in migrating parts of the applications to the cloud. Their focus is on determining which component to migrate and how to maintain restrictions and security policies. We are more interested in the financial cost aspect of the migration. Workload-based migration is also possible. Instead of considering which nodes to place in the cloud, one can opt to use the cloud only when workload overflows the current capacity of the system. Since it is a form of workload off-loading by horizontally slicing the peaks, we refer to this as a *horizontal migration* scheme. As a reciprocal notion to this, former scheme in which part or all of system nodes migrate to the cloud is called a *vertical migration*. *Horizontal* migration is being widely used as a way to handle unexpected burst of traffics in business such as KBB.com [18] or Domino's Pizza [17].

2.3.1 Deploying Database Applications

Application migration is further complicated by wide range of application deployment options when using the cloud services. Table 1 describes possible deployment configurations. It assumes the *vertical* migration type and the focus is on which tier to place in the cloud and on which database deployment options to select. Assuming the enterprise application in which there is a database server tier, Table 1 differentiates non-database and database tiers. For each, we consider placing them in the cloud or in-house. In-house setting can be divided into virtualized and non-virtualized. This gives rise to large number of possible combinations of deployment options. For the case of *hybrid* migration, if one chooses to keep non-database tiers in house and uses the Cloud for database, there are total of $2 \times 3 = 6$ possibilities.

The number of all identified deployment options demonstrates the complexities of determining the most cost-optimal choice relying only on some rule of thumbs. Our cost analysis explores all possible cases with

actual performance numbers and current cloud prices. For each deployment options we show the expected NPV of costs and provide intuitions for the outcomes.

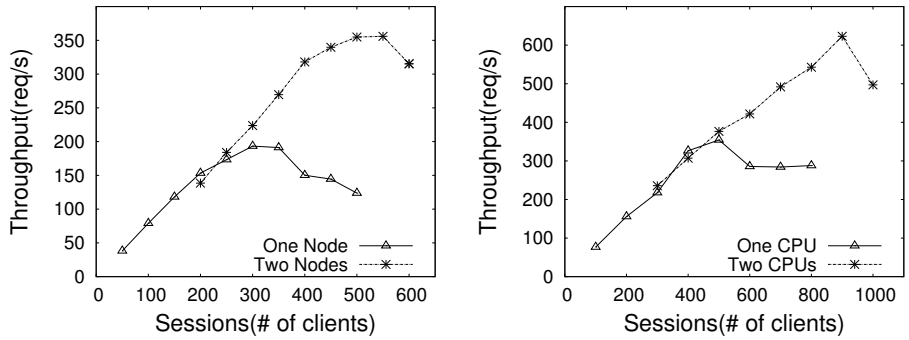
2.4 Our Methodology

2.4.1 Brief Outline

We consider hosting options offered by two prominent cloud providers: Amazon and Windows Azure, including both IaaS (EC2 instances) and SaaS options (Amazon RDS and SQL Azure). We consider the following five hosting options: (i) fully in-house, (ii) fully EC2 (the entire application is migrated to Amazon's EC2 cloud with components hosted within appropriately provisioned EC2 instances), (iii) EC2+RDS (similar to fully EC2 except the database which uses Amazon's RDS SaaS), (iv) in-house+RDS (a vertical partitioning where the database is migrated to Amazon's cloud to use its RDS SaaS while the remaining components are in-house), and (v) in-house+SQL Azure (a vertical partitioning similar to (iv) with RDS replaced with Microsoft's SQL Azure SaaS). We compare these hosting options for the following two applications from TPC [10]: (1) TPC-W (a benchmark that emulates an online bookstore) and (2) TPC-E (a benchmark that emulates online transaction processing in a brokerage firm). We assume that TPC-W is built using open-source software components (Apache, JBoss, Mysql) except for the OS (Windows), whereas TPC-E uses licensed software (SQL Server 2008 and Windows Server 2008). Both applications have three tiers: Web, Java-based application logic, database.

Our cost comparisons require us to make a number of projections/assumptions. We allow for a function that describes workload growth over time (increasing, decreasing, or stagnant in its form) and incorporate this into our NPV calculation. We incorporate both hardware and software upgrades to up-to-date products at typical refresh cycles (4 years for both hardware and software). We project CPU, memory capacities based on Moore's Law (similar to [13]). E.g., we assume CPU speed doubles every two years.

Finally, we need to estimate the hardware needs of our applications for a range of workload intensities (expressed in transactions/second or tps). Our goal is to find configurations across hosting options that offer similar, satisfactory performance. By running TPC-W on machines in our lab (each containing Intel Xeon 3.4GHz dual-processor with 2GB DRAM), we identify *marginal throughput gains* offered by adding an extra server (and CPU) to a tier. Using microbenchmarks, we determine cloud instance configurations that offer "comparable" computing power and memory (encouragingly our results match well with existing work that has benchmarked TPC-W on EC2 [6]). Since EC2 instances come in much smaller sizes, a comparable in-cloud configuration has a larger number of VMs. E.g., the EC2 *small* instance has an effective CPU of 1.1 GHz implying each of our lab machines is equivalent to about three of these. For TPC-E, we are unable to carry out a benchmarking-based estimation since we do not have the license for a MS SQL server. Instead, we employ performance and cost results offered for TPC-E on the TPC Web site for a number of machine, network, and storage configurations [10]. We note that the general problem of modeling resource needs is non-trivial with extensive work for in-house (including ours [11]) and emerging work for the cloud [9], and incorporating such estimates into our costs may be a useful future direction.



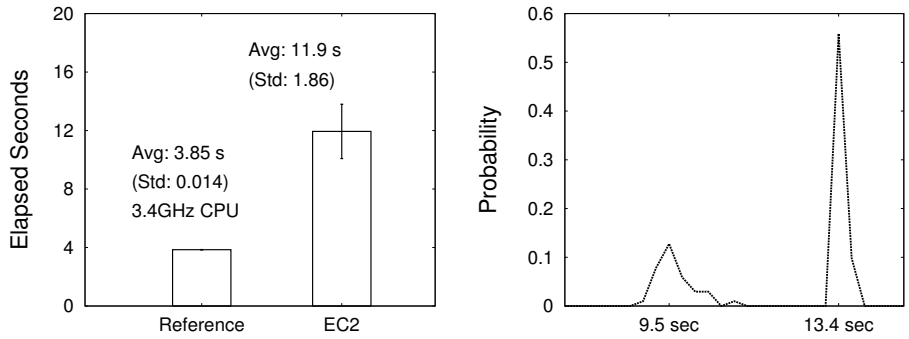
(a) Jboss server (b) MySQL server

Figure 2: Marginal throughput measurements.

2.5 Determining Hardware Needs

The cost of provisioning an application is directly related to the number of hardware units required to handle the target workload. We use an empirical method to determine the hardware resources that each tier needs so the application may sustain a given workload intensity. We are interested in finding configurations across our hosting options that offer same/similar performance (iso-performace). Rather than attempting to find an optimal/minimal sized configuration (which is a non-trivial problem with extensive related work []), we find a "reasonably accurate" configuration - this corresponds to assuming that the IT infrastructure would be sufficiently overprovisioned to avoid any tier from saturating. Enhancing our analysis with more sophisticated techniques for inferring resource needs is part of future work. Our discussion below is concerned only with TPC-W. Since we do not have an Oracle database in our lab, we use projections based on existing results from TPC-E Website for determining the hardware needs of TPC-E for different workload intensities.

In-house Provisioning: We employ a cluster of servers in our lab as our in-house hosting platform all of which have a Intel Xeon 3.4GHz dual-processor with 2GB DRAM and are connected via a 1 Gbps Ethernet. In order to determine the number of machines required to meet the desired throughput, for each tier, we empirically obtain the *marginal throughput gain* offered by adding an extra unit (granularity of CPU core as well as single machine) to it when all other tiers are well-provisioned. Assuming each unit is eventually operated at relatively low utilization (i.e., it is sufficiently over-provisioned), we can use these marginal gains to predict the capacity needs of each tier for a given workload intensity. Once we have the information about marginal throughput we are able to calculate how many computing units we need from T/m_i where T is the target throughput and m_i is the marginal throughput of tier i. Fig. 2(a) and (b) show the marginal throughput offered to TPC-W (i) by an extra machine for JBoss, and (ii) by an extra CPU for Mysql, respectively. For Jboss tier (Figure 2 (a)), we provision sufficient capacity to other tiers to make Jboss the bottleneck. We then measure the throughput as we increase the number of servers (each has only one CPU activated). With one machine, the throughput tops at 159 tps and 300 sessions. When two machines are used, the maximum throughput reaches 293 tps at 550 sessions which is roughly twice the case with one machine. For MySQL tier (Figure 2 (b)), we tested the maximum achievable throughput using one CPU as well as two CPUs and observed the marginal throughput gain. Similar to the Jboss case, adding additional CPU also doubled the maximum throughput. From these empirical results, we obtain the capacity of each of our machines for



(a) Performance Comparison (b) Latency Distribution

Figure 3: EC2 Instance's CPU Microbenchmark Results

Jboss and Mysql tier to be 146.6 tps (Transaction Per Second) and 311.5 tps per machine. As for the Apache tier, the resource consumption was insignificant and we estimated from observed CPU utilization that one machine could handle about 4k tps.

Cloud-based Provisioning: We would like to find cloud-offered resources that are likely to offer performance to TPC-W comparable with that offered in-house. The hosting options that we consider require us to do this exercise for the following: (i) Amazon EC2 instances (IaaS), (ii) Amazon RDS (SaaS) for database, and (iii) SQL Azure (SaaS) for database. Based on existing results, we assume that for (ii) and (iii) the cloud provider internally employs techniques to scale resources to match workload needs and this reflects in the payments []. For (i), we must ourselves determine the resource needs and procure sufficient number of instances. We describe our methodology for estimating this number for each tier of TPC-W where we employ Amazon EC2's *small* instance type (we did not find any improvements in performance/dollar offered by *large* and *extra large* instances, hence we restrict our attention to only *small* instances). Amazon EC2's *small* instance type claims to provide a computing power equivalent to 1.0-1.2 GHz CPU. Interestingly the `/proc/cpuinfo` of such an instance shows that it has a Intel(R) Xeon CPU 2.6GHz CPU. It is known that Amazon EC2 multiplexes two VM instances on one physical core making it effectvely 1.3GHz. We run a simple microbenchmarks to verify this and to establish computing power relative to our machines. Our microbenchmark performs increment operations on an integer variable in a loop as a single thead. We set the loop count to be 2×10^9 times and measured the elapsed time for both the reference machine that had 3.4GHz CPU and the EC2 *small* instance that had 2.6GHz CPU. As shown in Figure 3, we find that EC2 CPU performance shows high variance with bi-modal distribution (Figure 3(b)). EC2 *small* instance is found to operate at about one-third the speed of our machine, which matches well the claim of 1.0-1.2 GHz CPU. Using this benchmark information we set the throughput limit of single EC2 *small* instance for the Jboss and Mysql tiers to be 57.34 tps (Transaction Per Second) and 121.86 tps, respectively. Similar benchamrking for other EC2 instance types as well as offerings from other IaaS providers (e.g., Rackspace, GoGrid. etc.) and including these options into our overall cost calculations is part of ongoing work.

3 NPV Analysis Results

In this section we present key insights from our NPV analysis. All EC2 and RDS costs are based on the *reserved* instance pricing for TPC-W and TPC-E with all hosting options discussed in Section 2. We

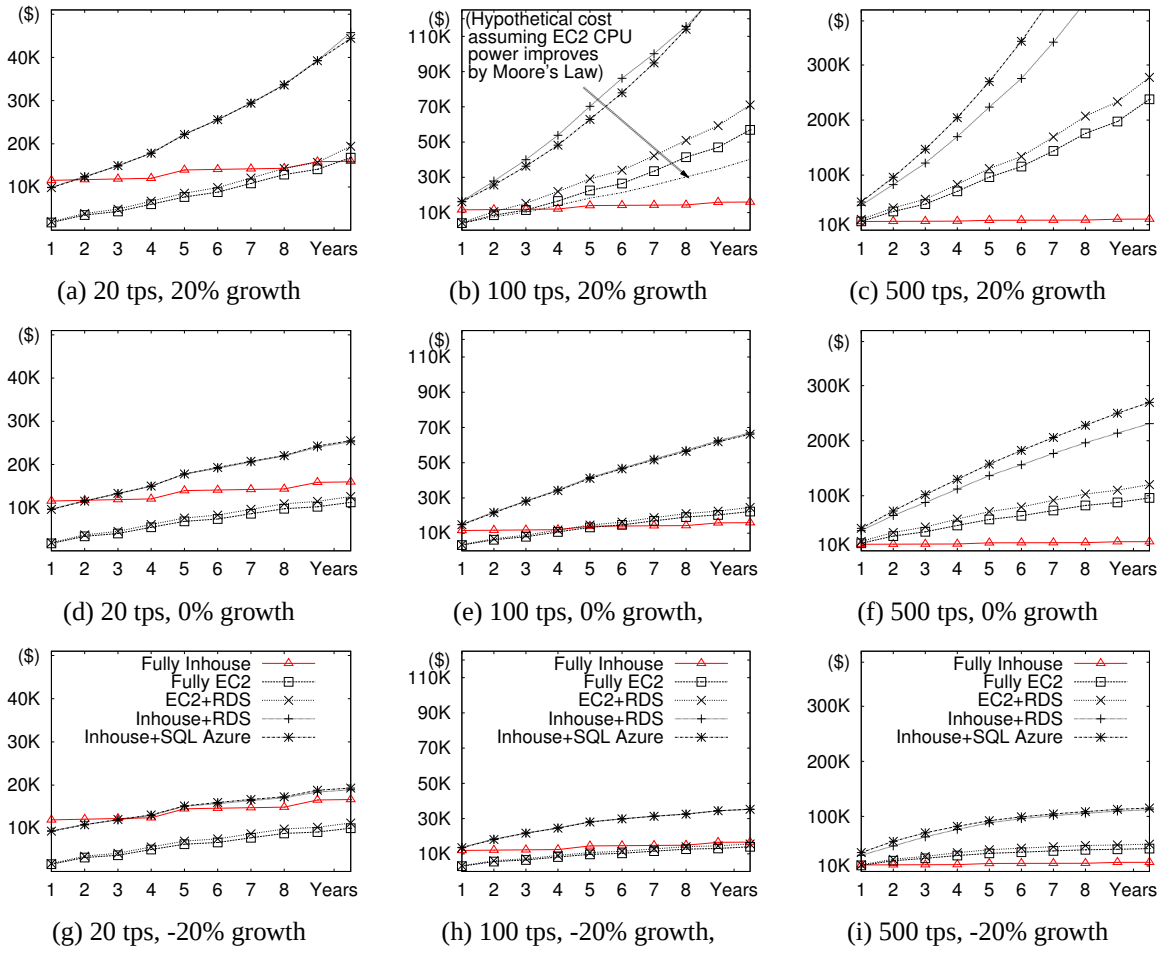


Figure 4: NPV over a 10 year time horizon for TPC-W. We consider three different workload intensity of small (20 tps at t=0), medium workload intensity (100 tps) and high workload intensity (500 tps) We also consider three different workload growth rate of -20%, -% and 20%.

only consider quantifiable and direct cost factors in the present work, and discuss other factors only when appropriate. We list some of the noteworthy points in determining the NPV. (i) For In-house, we assume that hardware is upgraded every 4 years. (ii) When upgrading the hardware, we take into account the salvaged value of existing hardware. (iii)

3.1 Workload Intensity and Growth

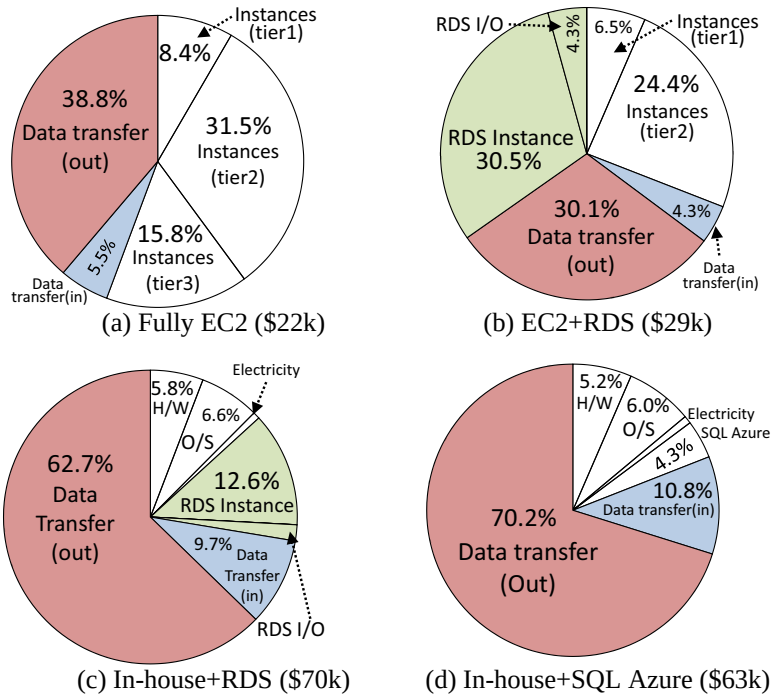
Fig. 4 presents NPV calculations for up to a 10 year time horizon for TPC-W. We present results with two workload intensities at the beginning: (i) 20 tps and (ii) 100 tps, which represent "small" and "medium". We also present two intensity growth scenarios: (i) stagnant and (ii) 20% increase per year; we have also considered other growth rates. As the workload intensity grows, TPC-W requires more servers and higher IO bandwidth but its storage capacity needs do not change. Overall, we find that *in-house provisioning is cost-effective for medium to large workloads, whereas cloud-based options suit small workloads.* For small workloads, the servers procured for in-house provisioning end up having significantly more computational power than needed (and they remain severely under-utilized) since they are the lowest granularity servers available in market today. On the other hand, cloud can offer instances with computational power matching the small workload needs (due to the statistical multiplexing and virtualization it employs). For medium workload intensity, cloud-based options are cost-effective only if the application needs to be supported for 2-3 years, and become expensive for longer-lasting scenarios. These workload intensities are able to utilize well provisioned servers making in-house procurement cost-effective.

An interesting trend is the *significantly slower NPV increase for in-house compared to cloud-based options*, which may be partly explained as follows. Since we assume hardware capacity growing according to Moore's Law, unless the workload growth matches or exceeds this rate, the number of servers required in-house will actually shrink each year. E.g., even for medium, eventually the number of servers comes down to 1 per tier. This is why the cost of 'fully In-house' case does not show steep increase. However, things evolve differently with cloud-based options. The computing power as well as price of a cloud instance are intentionally *engineered* to be at a certain level (via virtualization and statistical multiplexing) even though cloud providers may upgrade their hardware regularly (just as in-house). E.g., since the start of EC2 in 2006, the computing power/memory per instance has remained unchanged while there has been only one occasion of instance price reduction. In other words, while in-house hosting enjoys improvement in performance/$ with time, trends over the last 5 years suggest that the performance/$ offered by the cloud has remained unchanged [2]. Even if we assume the performance/$ offered by the cloud improves with time (say, an instance of given capacity becomes cheaper over time), cloud-based provisioning still remains expensive in the long run since data capacity and transfer costs contribute to the costs more significantly than in-house (See Fig. 4(b)).

3.2 Data Transfer, Storage Capacity, Software Licenses

We illustrate in Fig. 5 detailed breakdowns of NPV for five-year long hosting of TPC-W for hosting options involving the cloud (i.e., options (ii)-(v) from Section 2). Overall, we find that *data transfer is a significant contributor to the costs of cloud-based hosting* - between 30%-70% for TPC-W. This suggests

[2] It is important to remember that the improvements in performance/$ for in-house accrue due to investments made in upgrades.



(a) Fully EC2 ($22k) (b) EC2+RDS ($29k)

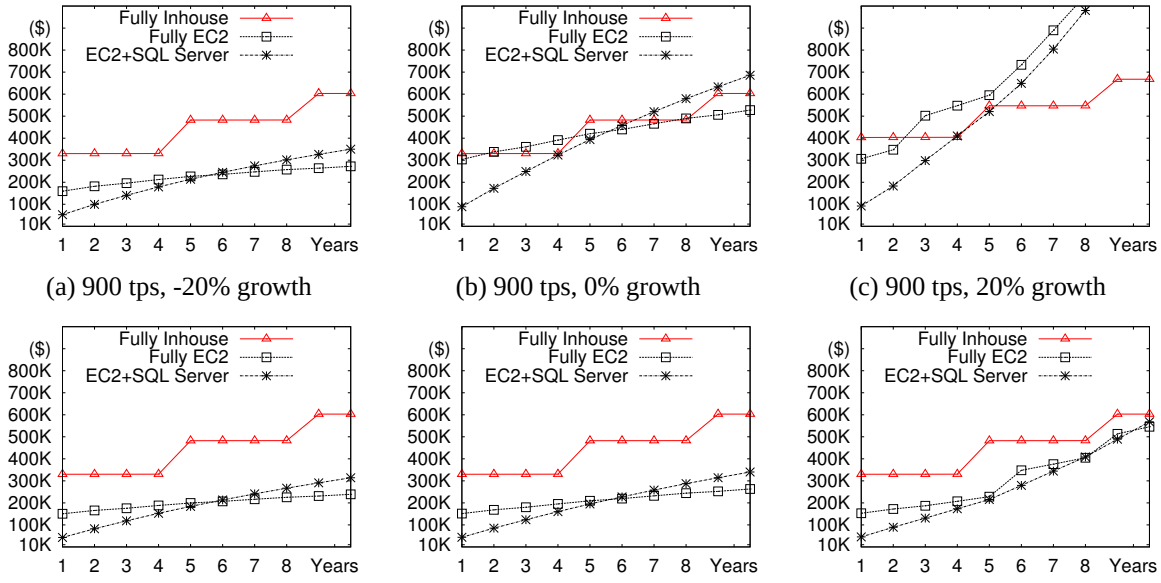
(c) In-house+RDS ($70k) (d) In-house+SQL Azure ($63k)

Figure 5: Closer look at cost components for four cloud-based application deployment options at 5th year. Initial workload is 100 tps and the annual growth rate is 20%.

that *vertical partitioning choices may not be appealing for applications that exchange data with the external world.* Data transfer (in & out) costs in Fig. 5(c),(d) are larger than those in Fig. 5(a),(b) because traffic per transaction between Jboss and MySQL (16KB/tr) is larger than between clients and Apache (3KB/tr).

Another key contributor to costs with cloud-based hosting can be storage capacity. Whereas TPC-W poses relatively small costs for storage capacity (its database only needs a few GB compared to those for servers and IO bandwidth and its storage capacity costs do not even show up in Fig. 5), our other application TPC-E has significant data storage needs (its database requires about 4.5TB). Fig. 7 presents the NPV evolution for TPC-E for two initial intensities - 300 tps (medium) and 900 tps (high). The annual growth rate in both cases is 20%. We only present fully in-house and two cloud (Fully EC2 and EC2+SQL server) options since we have already established the high costs of vertical partitioning. We find that in-house provisioning for TPC-E has to make significant investments in high-end RAID arrays (gap A), that constitute about 75% of overall costs. For initial workload intensity of 300 tps, these costs go down substantially with fully EC2 (i.e., renting storage from EC2 is cheaper than the amortized cost of procuring this much storage in-house), causing the overall costs to improve by 50% (year 1, shown as gap A) and 28% (year 6, shown as gap B in Fig. 7).

The software licensing fee for SQL Server and Windows can also be a significant contributor to TPC-E costs: second largest (17.4% of overall) (Fig. **??**(b)) and largest (67%) contributor, respectively, for fully in-house and EC2 options (Fig. **??**(c). Using *pay-per-use* SaaS DB allows the elimination of SQL Server licensing fees (shown as gap C in Fig. 7) and results in even better costs. *SaaS options can be cost-effective for applications built using software with high licensing/maintenance fee*. Note that these concerns did not



(a) 900 tps, -20% growth (b) 900 tps, 0% growth (c) 900 tps, 20% growth

(d) 300 tps, -20% growth (e) 300 tps, 0% growth, (f) 300 tps, 20% growth

Figure 6: NPV over a 10 year time horizon for TPC-E.

arise with TPC-W which employed open-source software, implying a different ordering of cost-efficacy among options.

It is also worth comparing the cost evolution for the two intensities (300 tps and 900 tps) in Fig. 7. With medium intensity (300tps), in-house option is less attractive than cloud-based options for the entire 10 year period without ever having a cross-over. However, at the higher intensity (900tps), cloud-based options quickly (after 2 years for fully EC2 and after 4 years for EC2+SQL server) become more expensive than in-house. This is qualitatively similar to the observations for TPC-W. However, cloud-based options remain attractive for a larger range of workload intensity than for TPC-W (compare Fig. 7 with Fig. 4(b) both of which have the same growth rate but differ in intensity by a factor of 9) - the key reasons for this difference are gaps B and C, i.e., the higher storage costs for in-house TPC-E as well as the contribution of software licenses in non-SaaS options. A final interesting phenomenon arises due to the following: when buying cloud instances for TPC-E database, we do find machines that offer required computational power per-VM but not the requisite degree of parallelism. The most powerful instance has 8 virtual cores. (In-house server uses 6 cores $\times$ 2.) This forces the cloud-based options to procure more instances than in-house with corresponding increase in the SQL server licenses (since Microsoft charges the fee per virtual core regardless of the underlying physical cores). This suggests that *a reconsideration of software licensing structures, particularly as applicable to large-scale parallel machines, may be worthwhile for making cloud-based hosting more appealing.*

3.3 Workload Variance and Cloud Elasticity

Our cost analysis so far were based on *average* workload intensities. Given high burstiness (i.e., high peak-to-average ratio or PAR) in many real workloads, it is common in practice to provision close to the *peak*. Whereas in-house provisioning must continue this practice, the usage-based charging and elasticity

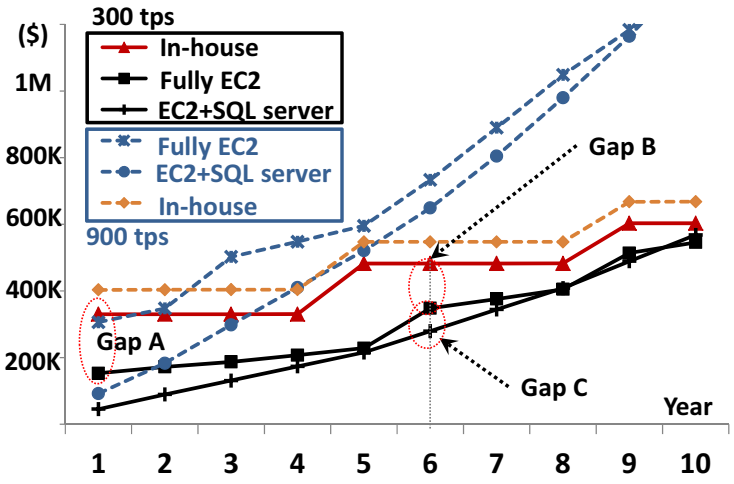
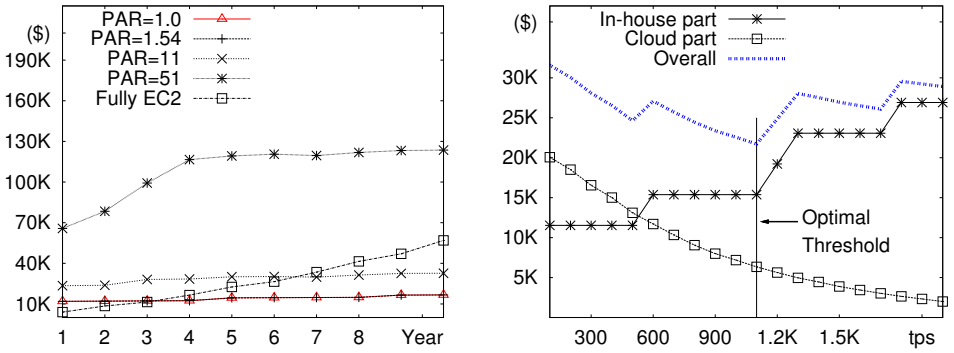


Figure 7: Two sets of TPC-E results at initial workload of 300 tps and 900 tps.

offered by the cloud open new opportunities for savings (for both in-cloud and horizontal partitioning). We investigate costs of variance-aware provisioning for three degrees of burstiness corresponding to time-of-day effects and flash crowds. Researchers have reported the magnitude of daily fluctuation to be around 40% to 50% range [2, 4] for the social networking applications, and about 70% (min:40, max:135 tps) for the e-commerce web site [15]. e-commerce Web site. Flash crowds can cause orders of magnitude higher peaks than the average and become a particularly appealing motivation for considering the use (perhaps partial) of cloud. The logs of World Cpu 1998 has reported 70 times increase of web requests due to flash crowds [1]. There has been many efforts to handle the flash crowds for the enterprise applications [8, 3, 12]. We represent the workload variance using *peak-to-average ratio* (PAR) which we define as $max(x_t)/E(x_t)$ where x_t is the time series of workload. We choose PAR of 1.54 (min=40, max=135 tps) to represent daily variations and PAR values of 11 and 51 to represent two flash crowd scenarios (i.e., peak of 10 and 50 times the average, respectively).

Fig. 8(a) illustrates the effect of three levels of burstiness on the in-house provisioning cost. We select the case of in-house with medium & increasing workloads (Fig. 4(b)). Provisioning for the diurnal fluctuation of 70% (PAR=1.54) does not impact the cost whereas flash crowd noticeably increases costs. Provisioning for PAR=51 shifts the cross-over point with "Fully EC2" from year 2.5 to year 10. The reason why diurnal fluctuation does not affect the cost is because provisioned servers already have enough capacity to embrace the peak of diurnal fluctuation. But, provisioning for flash crowds can substantially increase the cost.

We explore the benefits offered by a horizontal partitioning scheme that sets a threshold of workload intensity over which we create a replica in the cloud to handle the excess. Fig. 8(b) shows the cost change over a range of threshold at year 1. We assume a lognormal(μ:0,σ:1) distribution (mean:500tps) to simulate the bursty traffic. Blue dotted line in Fig. 8(b) is the overall cost, the sum of two components - in-house and cloud part. As the threshold moves to higher workload intensity, in-house cost rises in order to acquire more capacity, and the cloud cost lessens since the probability of overflowing the in-house server capacity diminishes. The equilibrium point where the cost is minimum is found at 1100 tps. This suggests that *horizontal partitioning can be effectively used to eliminate the cost increase from provisioning for the peak*.



(a) In-house Cost Change (b) Horizontal Partitioning

Figure 8: Effect of workload variance and horizontal partitioning on in-house cost.

4 Conclusions and Future Directions

Estimating the cost of application migration to the cloud is critical and for today's businesses since it is directly relates to their profitability. However, it challenging due to large number of variables at hand and complexity of their interactions. In this study we have investigated the migration costs of several deployment options using popular benchmarks. We have shown that application characteristics such as workload intensity, growth rate, storage capacity and software licensing costs produce complex combined effect on overall costs. We have also briefly explained issues regarding workload variance and horizontal partitioning. Overall, we find that (i) complete migration to today's cloud is appealing only for small/stagnant businesses/organizations, (ii) vertical partitioning options are expensive due to high costs of data transfer, and (iii) horizontal partitioning options can offer the best of in-house and cloud deployment for certain applications.

Our work opens up interesting possibilities for future work. We would like to incorporate indirect costs (and also less quantifiable costs in some meaningful way). As immediate work, we are extending our study to a broader set of applications such MapReduce and undergraduate labs at Penn State.

References

[1] Martin Arlitt and Tai Jin. Workload characterization of the 1998 world cup web site. Technical report, IEEE Network, 1999.

[2] Gong Chen, Wenbo He, Jie Liu, Suman Nath, Leonidas Rigas, Lin Xiao, and Feng Zhao. Energy-aware server provisioning and load dispatching for connection-intensive internet services. In *Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation*, NSDI'08, Berkeley, CA, USA, 2008. USENIX Association.

[3] Xuan Chen and John Heidemann. Flash crowd mitigation via adaptive admission control based on application-level observations. *ACM Trans. Internet Technol.*, 5:532–569, August 2005.

[4] Saikat Guha, Neil Daswani, and Ravi Jain. An experimental study of the skype peer-to-peer voip system. In *Proceedings of the 5th International Workshop on Peer-to-Peer Systems, 2006*.

[5] Mohammad Hajjat, Xin Sun, Yu-Wei Eric Sung, David Maltz, Sanjay Rao, Kunwadee Sripanidkulchai, and Mohit Tawarmalani. Cloudward bound: planning for beneficial migration of enterprise applications to the cloud. In *Proceedings of the ACM SIGCOMM 2010 conf.*

[6] Donald Kossmann, Tim Kraska, and Simon Loesing. An evaluation of alternative architectures for transaction processing in the cloud. In *Proceedings of the 2010 international conference on Management of data*, SIGMOD '10, New York, NY, USA, 2010. ACM.

[7] Evangelos Kotsovinos. Virtualization: Blessing or curse? *Queue*, 8:40:40–40:46.

[8] Pratap Ramamurthy, Vyas Sekar, Aditya Akella, Balachander Krishnamurthy, and Anees Shaikh. Remote profiling of resource constraints of web servers using mini-flash crowds. In *USENIX 2008 Annual Technical Conference on Annual Technical Conference*, pages 185–198, Berkeley, CA, USA, 2008. USENIX Association.

[9] Omesh Tickoo, Ravi Iyer, Ramesh Illikkal, and Don Newell. Modeling virtual machine performance: challenges and approaches. *SIGMETRICS Perform. Eval. Rev.*, 37:55–60, January 2010.

[10] Transaction Processing Performance Council. `http://www.tpc.org/`.

[11] Bhuvan Urgaonkar and Abhishek Chandra. Dynamic provisioning of multi-tier internet applications. In *Proceedings of the ICAC 2005*.

[12] Bhuvan Urgaonkar, Prashant Shenoy, Abhishek Chandra, Pawan Goyal, and Timothy Wood. Agile dynamic provisioning of multi-tier internet applications. *ACM Trans. Auton. Adapt. Syst.*, 3:1:1–1:39, March 2008.

[13] Edward Walker. The real cost of a cpu hour. *Computer*, 42, April 2009.

[14] Edward Walker, Walter Brisken, and Jonathan Romney. To lease or not to lease from storage clouds. *Computer*, 43:44–50, April 2010.

[15] Qing Wang, Dwight Makaroff, H. Keith Edwards, and Ryan Thompson. Workload characterization for an e-commerce web site. In *Proceedings of the 2003 conference of the Centre for Advanced Studies on Collaborative research*, CASCON '03. IBM Press, 2003.

[16] Windows Azure Lessons Learned. `http://channel9.msdn.com/Blogs/benriga/`.

[17] Windows Azure Lessons Learned: Domino's Pizza. `http://channel9.msdn.com/Blogs/benriga/Windows-Azure-Lessons-Learned-Dominos-Pizza`.

[18] Windows Azure Lessons Learned: Kelley Blue Book. `http://channel9.msdn.com/Blogs/benriga/Azure-Lessons-Learned-Kelley-Blue-Book`.

[19] Windows Azure's hidden compute costs. `http://searchwindevelopment.techtarget.com/news/1507649/Windows-Azures-hidden-compute-costs`.