Cost-driven Service Provisioning in Hybrid Clouds

Mathias Björkqvist
IBM Research Zurich Lab
Rüschlikon, Switzerland

Lydia Y. Chen
IBM Research Zurich Lab
Rüschlikon, Switzerland

Walter Binder
University of Lugano
Lugano, Switzerland

***Abstract*—Hybrid clouds, which comprise nodes both in a private cloud and in a public cloud, have emerged as a new model for service providers to deploy their services. However, given Quality-of-Service requirements for each service, the question of on how many private and public nodes to deploy the services in the most cost-effective way remains to be answered. The challenges faced in the hybrid cloud stem from the disparate time-varying requests across multiple services, the different cost structures of both types of nodes, and the performance characteristics of nodes. In this paper, we propose a novel algorithm to dynamically optimize the allocation of private and public nodes across services, with special focus on the performance-cost tradeoff between private and public nodes. The algorithm is based on an analytical cost-performance framework for service deployment in hybrid clouds. Our evaluation results based on trace-driven simulation show that our proposed node allocation algorithm can effectively achieve a good cost-performance ratio, compared to the deployment of purely public and private cloud.**

***Keywords*-Service provisioning, cloud computing, hybrid cloud, cost model, optimization algorithm, dynamic node allocation, simulation**

I. Introduction

Cloud computing is an emerging computing paradigm, featuring elastic capacity provisioning and ease of operational management for a wide range of services. Resources, such as processors, storage, and network, are provided in an on-demand fashion to multiple service providers (i.e., clients of the cloud), who may deploy multiple services exhibiting disparate workload patterns. Essentially, cloud platforms enable resource sharing among multiple service providers, as well as among multiple services deployed by the same provider.

Cloud computing encompasses three different service models — Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS) [3]. Service providers can use four types of cloud deployment models — private cloud, community cloud, public cloud, and hybrid cloud [3]. The basic computing unit is the virtual node, on which different services can be deployed in an on-demand fashion. Depending on the specific cloud architecture, virtual nodes can correspond to either virtual machines or physical machines. We call the virtual nodes "in-house nodes" for private clouds, and "public nodes" for public clouds. Figure 1 illustrates a hybrid cloud.

Public and private clouds offer complementary advantages in terms of cost and performance. On the one hand, public clouds offer flexibility advantages due to their pay-as-you-go billing features, whereas private clouds are bound by fixed infrastructure cost and variable operational cost depending on system utilization. A highly varying workload may allow for cost savings, if public nodes are dynamically allocated to services on demand. On the other hand, as public clouds are shared by multiple service providers, the public nodes suffer from higher performance variability than in private clouds, as pointed out in [11].

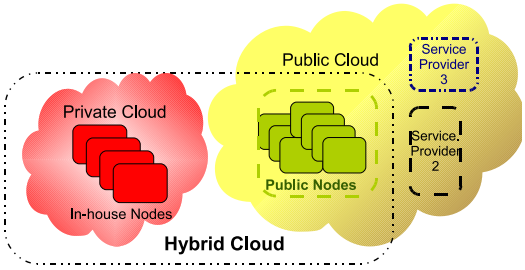


Fig. 1. Illustration of a hybrid cloud.

To leverage the cost and performance advantages of public and private clouds, recent studies have advocated a hybrid deployment model [16], [17], [19]. Related studies first tried to answer if using hybrid clouds was really cost-effective in the long run, when comparing to private and public clouds only. Due to the need for purchasing (or leasing) hardware infrastructure and software for in-house nodes, deployment on in-house nodes requires a longer planning horizon (e.g., several months), whereas public nodes can be purchased and returned in a much shorter period, such as an hour. As such, cost analyses for hybrid clouds tend to consider performance aspects only in a qualitative way. The service provider needs to decide how to allocate the available in-house nodes and public nodes to the different services, whose workloads often exhibit time-variability. Clearly, the answer hinges on the workload dynamics of the deployed services, on the cost structures of both kinds of nodes, on the performance characteristics of the nodes, and on Quality-of-Service (QoS) requirements. However, solutions presented in related studies consider only a subset of these criteria and thus fall short in optimizing the trade-off between cost and performance in hybrid clouds.

In this paper, we consider only the IaaS service model and the hybrid cloud deployment model. We address the following research question: For a service provider, what is the optimal level of service provisioning in a hybrid cloud? That is, how many in-house nodes and public nodes shall be used to minimize service provisioning cost, while meeting given QoS

requirements? To this end, we first develop a quantitative cost-performance optimization framework, which can identify the optimal number of in-house nodes such that the total cost of a hybrid cloud is minimized and a given target utilization and response times are met. Our cost model in particular factors in the long-term fixed cost and short-term operation cost. Second, we develop a novel node allocation algorithm to dynamically allocate available in-house nodes and required public nodes for each deployed service. Our algorithm takes several control actions within a slotted window: reconfiguration of in-house and public nodes from one type of service to another one, allocating and releasing public nodes, and replacing slower public nodes in the hope of getting faster ones. To evaluate the proposed cost-performance model and node allocation algorithm, we build a detailed simulator for services hosted in a hybrid cloud, driven by service utilization traces collected from real IBM production systems.

Our contribution here is two-fold: first, the proposed analytical cost-performance framework is not only generally compatible with detailed cost factor analysis in the related work, but also quantifies the performance constraints. Second, the practical node allocation algorithm considers a complex system dynamic, i.e., workload, node performance, and cost structure variability between two nodes, as well as diverse control actions, especially the reconfiguration to enforce the collaboration across services.

This paper is organized as follows: The system architecture is explained in Section II-A. The proposed cost-performance framework and node allocation algorithm is presented in Section III and IV. Section V contains the experimental results. Related studies are summarized in Section VI. Section VII concludes this paper.

II. Assumptions and System Model

In this section, we first introduce the hybrid cloud architecture considered in this paper. Second, we discuss the cost of deploying in-house and public nodes. Third, we explain the control actions taken by our algorithm to deploy service replicas on different nodes.

A. System Architecture

Figure 1 illustrates the system architecture of the hybrid cloud considered in this paper. It consists of a private and a public cloud. While the private cloud is assumed to have a constant number n^h of in-house nodes, the number of public nodes is changing dynamically. The public cloud is a commercial cloud, such as Amazon EC2 [1], shared with other users who may also be service providers. A service provider deploys I types of atomic services S_i ($1 \leq i \leq I$) in such a cloud. Herein, we assume that only a single service S_i can be configured on a node at any moment (though, each node can be reconfigured with a different service $S_{i'}$). Here, we do not consider composite services. At any moment, each service can be executed on $n_i = n_i^h + n_i^b$ nodes, where n_i^h is the number of in-house nodes and n_i^b is the number of public nodes. Each type of service is always deployed on at least one

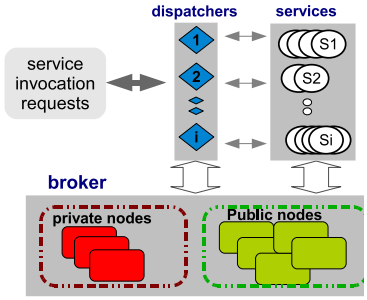


Fig. 2. System architecture of a hybrid cloud.

in-house node or public node. The total number of in-house nodes allocated to services is constrained by the available in-house nodes in the private cloud, i.e., $\sum_i n_i^h \leq n^h$, whereas the total number of public nodes is unbounded, i.e., $n_i^b \geq 0$.

Performance of in-house and public nodes. As the service provider has exclusive control over the in-house nodes, their performance (defined in terms of throughput) is assumed constant (i.e., we assume a private cloud built from homogeneous hardware). In contrast, the public nodes' are assumed to be heterogeneous and have varying performance, with a guarantee to provide at least the minimum throughput specificized in the contract.

To limit the scope of this study, we assume that all services are CPU-bound and multi-threaded, that is, capable of handling several concurrent service requests in parallel. We assume that the service execution time depends on the size of the input parameters passed upon service invocation. We further assume that this size varies following an exponential distribution, and thus the execution times for requests follow an exponential distribution (for a given throughput of a node).

Node management. There are two load-controlling entities in our architecture — (1) dispatchers and (2) a centralized node broker, as shown in Figure 2. All of them are deployed in the private cloud. For each service type, there is one corresponding dispatcher, which dispatches incoming service requests to in-house and public nodes that execute the requested service type; the local dispatcher chooses the node with the fewest outstanding requests. An extra network delay t_{net} is incurred when service requests are dispatched to public nodes. We assume the network delay incurred when dispatching to in-house nodes to be negligible. Additionally, the dispatchers monitor the allocated in-house and public nodes and keep track of statistics about the service invocation rate, node performance, and the billing periods of the public nodes. All dispatchers periodically communicate these statistics to the node broker, on which our proposed algorithm and the control actions are implemented.

B. Cost Model

The private and public parts in a hybrid cloud have disparate cost structures. The service provider is typically charged C^b per public node and billing period [1]. One can thus deploy services on public nodes at a fine granularity, based on workload demand. Turning off a node before the end of a

billing period leads to the waste of resources without any cost saving.

On the contrary, the in-house nodes are associated with a fixed cost and variable operational cost. The fixed cost results from leasing or purchasing of hardware, software and infrastructure for a longer period. To obtain specific values for fixed cost, we direct authors to studies [10], [12], [13], [17], which focus on providing detailed overviews of how to estimate fixed cost from different aspects. For a fair comparison, one can compute the fixed cost per billing period, F, for private nodes by dividing the fixed cost by the leasing period or depreciation period and then multiplying with the length of the billing period. In addition, the service provider needs to maintain operation of the in-house nodes, including cooling and network. Typically, the operational cost is proportional to how in-house nodes are utilized [8], [15]. The operation cost is modeled as a linear or quadratical function of the average utilization of a node [9]. Therefore, we calculate the cost of maintaining one in-house node per billing period, C^h, as the sum of the fixed cost and the operational cost $O(U)$ is $C^h = F + O(U)$.

To develop a general cost-performance framework in the following section, the operational cost $O(U)$ here is a general function in utilization, U, such that one can apply any specific function, depending on system characteristics. Indeed, the utilization value depends on the load and allocated capacity in each billing period, so that U is precisely written as $U(t)$ in the following cost function. We summarize the total cost of maintaining a hybrid cloud for T billing periods, indexed by t, as $\sum_{t=1}^{T}\{C^h(t) + C^b n^b\}$.

Note that as the cost of an in-house node depends on the utilization, $C^h(t)$ varies according to the utilization in every billing period. Consequently, we can then obtain the total cost as:

$$TC = \sum_{t=1}^{T}\{F + O(U(t)) + C^b n^b(t)\} \quad (1)$$

The challenge faced by a service provider is how to minimize the aforementioned cost while keeping both types of nodes reasonably utilized and the service response time low. In particular, our aim is to achieve a target utilization value such that a reasonable response time is ensured simultaneously. The specific value of the response time is stipulated in the service level agreement. The mapping between the utilization and response time is very system specific and beyond the scope of this paper.

C. Service Node Allocation

To strike a good balance between cost and performance in a hybrid cloud, we consider the allocation of nodes to services in slotted windows via the node broker. In particular, the broker can consider the following three control actions:

- Turn on/off public nodes. A public node is only turned off at the end of its billing period.
- Reconfigure existing public and in-house nodes to execute services of different types.
- Replace a public node at the end of a billing period (with another newly allocated public node), if the node is suspected of not performing well.

The first and third actions require communication with the public cloud operator. When reconfiguring the private nodes, it is subject to the available nodes in the private cloud. Note that the length of the control window depends on the dynamics of the workload and the parameterization of the node allocation policy. In summary, the private nodes are only subject to reconfiguration, whereas the public nodes are subject to all available control actions.

Time Overhead. We model some time overheads associated with each control action. On the one hand, we assume υ seconds to load and start the required service, when turning on public nodes and reconfiguring both types of nodes. The newly turned-on and reconfigured nodes are published as "available" after the completion of their loading/configuration process. One the other hand, when nodes are about to be turned off and reconfigured, they need to immediately stop receiving service requests and complete the pending invocations. Note that related studies [8] tend to overlook the overhead structure and lead to a simplified (but inappropriate) replication policy.

III. Optimal Number of In-house Nodes

Our objective is to provide sufficient number of in-house and public nodes to cater multiple services such that the total cost is minimized and the entire system is well utilized at the target value, such as 80%, for handling temporary workload variation. Fulfilling the target utilization ensures not only the effectiveness of node allocation, but at the same time, also the response time [6], [14]. The challenge here is how to strike a good balance between the cost, utilization, and response time, combining a constant number of in-house nodes and a dynamically changing number of public nodes.

To guide the search for the optimal number of in-house nodes, we formulate a simple cost-performance optimization, which aims to minimize the total cost in Eq. 1 subject to the target utilization, U^*, at each optimization interval. As the optimal value identified in our optimization depends on certain assumptions, we provide not only the means to achieve solution but also a practical rule of thumb, based on a theoretical lower bound.

A. Cost-Performance Framework

We assume there is a total of T intervals, which are indexed by t. The utilization at interval t, $U(t)$ is defined as the sum of the invocation rate of all services, $\lambda(t)$, divided by the total allocated node capacity, $(n^h + n^b(t))\mu$, where μ denotes the average throughput of a node. As such, we can write the cost-performance optimization in the hybrid cloud as follows:

$$\min_{n^h} \quad TC = \sum_{t=1}^{T}\{F + O(\frac{\lambda(t)}{(n^h + n^b(t))\mu})\}n^h + C^b n^b(t) \quad (2)$$

$$st. \quad \frac{\lambda(t)}{(n^h + n^b(t))\mu} \leq U^* \forall t \quad (3)$$

Note that the variable cost incurred for an in-house node is a function of $U(t)$, and we can thus write $O(U(t))$ as $O(\frac{\lambda(t)}{(n^h+n^b(t))\mu})$ in the objective function.

To solve the aforementioned optimization problem, we make the following assumptions: (1) $\lambda(t)$, the sum of invocation rates for all services, is known for all intervals, and (2) the dynamic allocation of public nodes, $n^b(t)$, is perfectly achieved by the broker, i.e., public nodes are allocated such that they are exactly 80% loaded in every window. Consequently, the proposed cost-performance framework is a non-linear optimization in n^h. One can resort to the standard optimization software, such as C-plex [2], for identifying an optimal allocation of in-house nodes.

Nevertheless, the optimality of n^h obtained in the cost-performance constraint hinges on the validation of those two assumptions. First of all, one can only predict $\lambda(t)$, whose predictability depends on the workload dynamics. In turn, the effectiveness of the node allocation algorithm can be easily hampered, e.g., the allocation of public nodes $n^b(t)$ tends to be higher or lower than the required load. Therefore, we conclude that the optimal number of in-house nodes depends on not only the cost, but also on the workload dynamics and on the effectiveness of the node allocation algorithm.

B. Lower Bound of In-house Nodes

In this subsection, we explain the practical aspect to dimension the number of in-house nodes, based on the lower bound of the proposed cost-performance framework. We first summarize the simple provisioning rule for the in-house nodes, and then describe the theoretical lower bound.

Provisions Rule. First, one needs to compare the unit cost of public nodes and the maximum unit cost of in-house nodes operated at the target utilization. When the unit cost of public node is higher, a service provider should then search for the minimum value of λ, using historical data. Finally, the adequate number of in-house nodes is set at the value, computed as the minimum λ divided by the average throughput of a node at the target utilization, i.e., $U^*\mu$. As mentioned in the previous section, the choice of the minimum value of λ should be adjusted according to the knowledge of workload dynamics and effectiveness of the allocation of public nodes.

The aforementioned rule is directly applicable to systems hosting a single service. When it comes to a system with multiple services, we further suggest to aggregate the invocation rate from all services, i.e., $\lambda(t) = \sum_i \lambda_i(t)$. Moreover, as the the average throughput of a node is associated with the hosted services, the average throughput of a node should be precisely defined as μ_i. We then suggest to replace μ by the maximum μ_i in the following theorem.

Theorem 3.1: The theoretical lower bound of n^h in the proposed cost-performance optimization for hybrid cloud is

$$\begin{cases} (1) n^h = 0, & \text{when } C^b < C^h(U^*); \\ (2) n^h = \frac{\min_t \lambda(t)}{U^*\mu}, & \text{when } C^b > C^h(U^*). \end{cases}$$

Proof: We first simplify the objective function by assuming that the target utilization, U^*, is achieved, and so is the upper bound of C^h. One can thus treat C^h as a constant term and obtain the theoretical lower bound of n^h. Secondly, we rewrite $n^b(t)$ as function of n^h, taking the equality constraint of Eq. 3,

$$n^b(t) = \frac{\lambda(t)}{U^*\mu} - n^h \geq 0 \forall t. \tag{4}$$

Substituting Eq. 4 into the objective function Eq. 2, one can then obtain TC as follows (after some straightforward algebraic manipulation):

$$TC = \sum_t (\frac{\lambda(t)}{U^*\mu}) C^b + n^h (C^h - C^b) T. \tag{5}$$

The first term is the summation of known variables, and the second term is a linear function in n^h. Note that as there is a variable cost depending on the utilization of in-house nodes, C^h should written as a function of the utilization, $C^h(U^max)$. One can thus discuss the optimal n^h in the following two cases: (1) $C^b < C^h(U^*)$. When the unit cost of a public node is less than an in-house node operating at the target utilization, the optimal value of n^h is zero. It implies that no in-house nodes shall be allocated in the hybrid cloud, meaning that using public cloud only can achieve the minimum cost. (2) $C^b > C^h(U^*)$. When the unit cost of an in-house node is less than a public node, one shall try to have as many in-house nodes as possible so that the total cost is minimized. However, deriving from Eq. 4, there exists an upper bound for the allocation of n^h. To fulfill such a constraint for all intervals t, we then obtain $n^h \leq \frac{\lambda(t)}{U^*\mu} \forall t \leq \frac{\min_t \lambda(t)}{U^*\mu}$ ■

IV. Collaborative Node Allocation Algorithm

To dynamically allocate in-house and public nodes across multiple services, we modify the opportunistic policy proposed in [7]. The proposed algorithm implemented in the broker enforces the collaboration across services, i.e., nodes can be reconfigured from one service to another, according to the invocation load and billing period. Moreover, our proposed algorithm decides not only on the number of in-house and public nodes per service, but also intends to fully use in-house nodes and acquire public nodes with better performance.

The general idea of our collaborative algorithm is that the broker first decides on the number of nodes needed for each service, based on the information monitored/collected in dispatchers. The second step is to select specific in-house and public nodes for each service, using appropriate control actions. The selection criteria considered are the billing period for public nodes, the difference in the number of nodes in adjacent windows, and the performance of active public nodes.

A. Optimal Total Number of Nodes per Service

To proactively provide a sufficient number of well-utilized nodes for each service at the beginning of every control window, $n_i(t)$, we follow the analysis provided by [7]:

$$n_i(t) = \lceil \frac{\widehat{\lambda_i(t)}}{U^*\widehat{\mu_i(t)}} \rceil, \forall i,t, \tag{6}$$

where the values for the invocation rate and average throughput are based on the last value prediction,

$$\widehat{\lambda_i(t)} = \lambda_i(t-1) \quad (7)$$

$$\widehat{\mu_i(t)} = \frac{\sum_{j=1}^{n_i(t-1)} \mu_{ij}(t-1)}{n_i(t-1)}. \quad (8)$$

Note that due to the performance variability of public nodes, one needs to estimate the average throughput of window t, $\mu_i(t)$, by the measured throughput of all nodes allocated for service i at window $t-1$, $\mu_{ij}(t-1)$. We direct readers to [7] for the detailed derivation. Additionally, as there is an extra network delay for using public nodes, we incorporate such a factor into the computation of node throughput as follows: $\mu_{ij}^{new} = \frac{1}{1/\mu_{ij}+t_{net}}$.

B. Collaborative Algorithm

The objective of allocating nodes is to use as many in-house nodes as possible and maintain as few and as many fast public nodes as possible, such that the return on payment of allocated nodes is maximized. Consequently, the three main ideas of the proposed collaborative algorithm are the following: (1) Due to the billing constraint and performance uncertainty of public nodes, the broker gives high priority to allocating/de-allocating in-house nodes over public nodes. (2) The broker uses the reconfiguration action to enforce the collaboration among services when decreasing and increasing the allocation of in-house and public nodes for each service. (3) The broker only turns on new public node for services when there are no spare in-house or public nodes toe be reconfigured from providing other services, and only turns off or replaces the public nodes whose billing period ends.

The collaborative algorithm used by the broker is structured into two parts: node allocation, and node de-allocation, using three available control actions explained in Section II-C. The algorithm first focuses on services which have a negative difference in the number of nodes in adjacent windows, $\delta_i(t) = n_i(t) - n_i(t-1) \leq 0$, and thus requires node de-allocation. Thereafter it focuses on services which require additional node allocation compared to current allocation, i.e., $\delta_i(t) > 0$.

To facilitate selecting control actions for allocating and de-allocating nodes, we keep a reconfiguration list which keeps track of the reconfigurable nodes. The list is filled up during the de-allocating part of the algorithm. To maximize the utilization of in-house nodes, a higher priority is given to in-house nodes and slower public nodes when flushing out the list during the allocation phase of the algorithm.

1) De-allocating Nodes: For services with $\delta_i(t) < 0$, the broker greedily optimizes the aggregate node capacity of individual services by turning off its expiring public nodes or reconfiguring its in-house/public nodes. We let the number of expiring public nodes for service i at interval t be $E_i(t)$. When there is not a sufficient number of expiring public nodes to be de-allocated, the broker first turns off the expiring public nodes and then selects $\{|\delta_i(t)| - E_i(t)\}$ nodes for the reconfiguration list as follows: The in-house nodes are given a higher preference than non-expiring public nodes, and slower non-expiring public nodes are more preferable than faster ones. The reconfiguration list is first filled up by services with $\delta_i(t) < 0$ in a sequential order of service index, and then flushed out by services with $\delta_i(t) > 0$ in a round robin fashion for reasons of fairness. As such, the time overheads associated with replacing and reconfiguring nodes can be minimized.

When there are more expiring nodes than nodes to be de-allocated, the broker first turns off the slowest $|\delta_i(t)|$ expiring public nodes. Then, the broker tries to replace remaining expiring public nodes by comparing the corresponding cost and benefit as proposed in [7]. The cost of replacing a public node is the unavailability of its capacity during the time a replacement node is being configured, and the potential benefit is the chance of obtaining a public node with better performance.

2) Allocating Nodes: Once the broker completes the process of de-allocating nodes, it proceeds to the services requiring additional allocation of nodes. The broker first tries to distribute available nodes on the reconfiguration list, and then turn on new public nodes where required. The allocation of nodes on the reconfiguration list starts with in-house nodes, followed by the slowest public nodes, and distributes them to services with $\delta_i(t) > 0$ in a round-robin fashion.

When there is a sufficient number of nodes on the reconfiguration list, meaning that the total number of reconfigurable nodes is greater than the number of nodes needed to be allocated, the remaining nodes on the reconfiguration list are returned to their original services. On the other hand, when there is not a sufficient number of nodes on the reconfiguration list to meet the requirements for node allocation, the broker first distributes the reconfigurable nodes in a round-robin fashion, and then turns on additional public nodes to meet the unfulfilled demand.

V. Evaluation

In this section, we evaluate the provisioning of service systems hosted on hybrid clouds. We vary the number of in-house nodes, and use trace-driven simulation, in conjunction with the proposed collaborative allocation algorithm.

The performance metrics evaluated are the total cost, the average utilization of in-house nodes, and the average response time of an invocation for all services. Our evaluation results, based on the average of ten simulation runs, show how to adjust theoretical lower bound based on the proposed cost-performance framework such that the optimal number of in-house nodes is identified.

A. System Configuration

We built a trace-driven simulator of service-oriented systems in the hybrid cloud using Java. Invocation requests are generated for each service with time-varying arrival rates. A node dispatcher for each service type forwards requests to the appropriate in-house nodes or public nodes. Forwarding a request to a public node incurs an extra network delay of $t_{net} = 0.075$ seconds. Moreover, we assume a public node can

have three different levels of performance to process requests of each service in our simulated cloud environment. The specific values of three levels are $\{1, 1.1, 1.25\}$, representing the normalized throughput with respect to the throughput of an in-house node, which has a performance level of one. The probabilities of obtaining public nodes with different α values are 0.6, 0.25, and 0.15, respectively. The aforementioned values can be configured according to values measured in different cloud platforms.

The node controller collects the required statistics 20 seconds before every control window of length 5 minutes and the node broker immediately computes and implements the control actions, some of which have a time overhead of $\upsilon = 20$ seconds. The specific length of the control window is chosen according to workload characteristics and prediction schemes. The discussion of the optimality of those values is beyond the scope of this paper.

B. Cost Calculation

We follow the convention in today's commercial cloud [1] and use an hour as the billing period for public nodes and as the basic unit for in-house nodes. To simplify the comparison, we present our results in normalized cost with respect to the in-house nodes at maximum utilization, i.e., 100%, per billing period. That is the maximum cost of an in-house per billing period, calculated as one unit of normalized cost. When an in-house node is not fully utilized per billing period, it is charged less than one unit of normalized cost. Using a public node incurs a premium compared to using an in-house node. The specific premium values are given in the different scenarios. In particular, we assume that the variable cost per in-house node, $O(U)$, accounts to 40% of the maximum cost and follows a linear function, i.e., $O(U(t)) = 0.4U(t)$, and quadratic function , i.e., $O(U(t)) = 0.4U(t)^2$, for the single and three service case respectively. The fixed cost of in-house node, encompassing leasing, cooling, and constant power term [8], accounts for the rest of the cost. Note that the choice of cost values depends very much on the deployment and business models. The applicability of our analysis is not restricted by the choice of cost values.

C. The Workloads: Invocation Requests

Following approaches used in [6], [7], we adopt the utilization traces from current IBM production systems as workload input for each service, i.e., to generate the invocation requests. In particular, utilization traces are collected from three large multi-processor servers engaging in web services in financial, airline and media industries in late January, 2012. The utilization values are the average computed over 15 minutes. Fig. 3 depicts the collected request rate for four services, which show clearly the different characteristics of time-variability.

The execution times of each service are assumed to follow the exponential distribution with mean $\frac{1}{\mu_1} = \frac{1}{0.4}, \frac{1}{\mu_2} = \frac{1}{1}, \frac{1}{\mu_3} = \frac{1}{2}, \frac{1}{\mu_4} = \frac{1}{2.5}$ seconds respectively. Note that due to the performance variability of public nodes, the execution times there can be scaled down by the normalized performance, namely

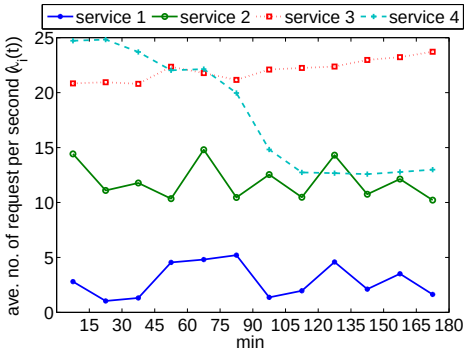


Fig. 3. Average request rates of services, $\lambda_i(t)$.

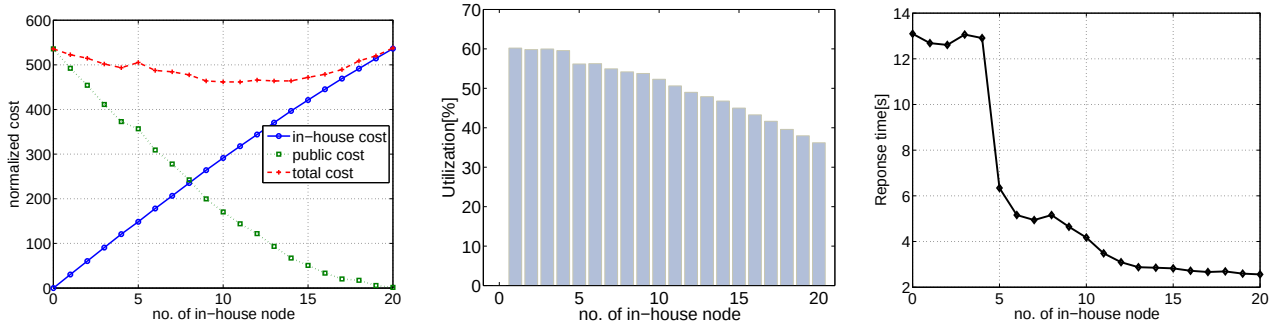
by 1.1 or 1.25, whereas the execution times of in-house nodes follow the aforementioned values.

D. Single Service Case

We first evaluate a simple scenario, where a service provider only caters service 1 shown in Fig. 3. In this scenario, the public node premium is set to 10%, compared to the maximum cost of an in-house node. That is, a public node is charged with 1.1 units of normalized cost per billing period. We summarize the cost, average utilization of in-house nodes, and the average response time, with respect to different provisioning of in-house nodes in Figs. 4 (a), (b) and (c), respectively. The hybrid cloud, combining in-house and public nodes, can achieve a better tradeoff between performance and cost, i.e., lower response time at lower cost, compared to purely public or private cloud.

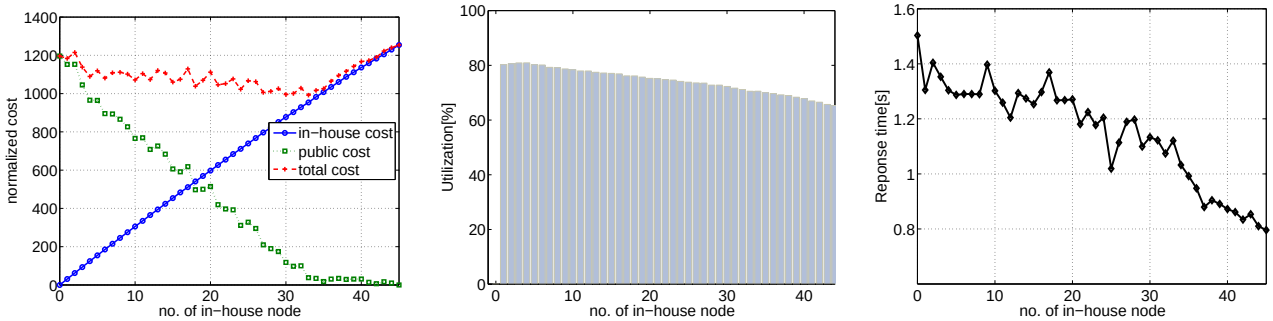
One can see that the trend for the cost of in-house node is not exactly linear in the number of in-house nodes, due to the decreasing utilization. On the other hand, the cost for public nodes roughly decreases with increasing in-house nodes, with a less smooth trend. This is partially because of the inaccuracy of the proposed algorithm in predicting workloads and allocating nodes, and partly because of the billing period restriction when de-allocating nodes. The overall cost trend of public nodes essentially depends on the workload dynamics and node allocation algorithm, as mentioned in Section III. Consequently, the total cost first decreases when the number of (less expensive) in-house node increases, and then increases when there are too many under utilized in-house nodes.

We can observe that the minimum total cost occurs when the number of in-house nodes is at 9, which is higher than the theoretical lower bound (i.e., 4 in-house nodes), using the lowest value of average invocation request rate in the trace as the minimum $\lambda(t)(= 1.1)$ and $\mu = 0.4$ in Theorem 3.1. However, as the broker uses simple last value prediction, which induces an ineffective allocation of public nodes, a higher allocation of in-house nodes not only reduces the cost, but also the response time. To compensate for such an effect here, one should set the minimum $\lambda(t)$ equal to the value of 60^{th} percentile of the invocation rate trace, when using formula 3.1 to decide the provisioning of in-house nodes. Nevertheless, finding the optimal n^h requires off-line



(a) Cost (b) Utilization (c)Response time

Fig. 4. Performance summary of one service.



(a) Cost (b) Utilization (c)Response time

Fig. 5. Performance summary of three services.

statistical training to choose an appropriate invocation rate. In the following, we provide another practical method combining two simulation runs to identify n^h.

Practical Recommendation. Herein, we suggest that to empirically and quickly find the optimal of number of in-house nodes, one first should run simulation with historical traces to extrapolate the actual trends of public node cost and in-house node cost with respect to increasing in-house nodes. The required simulation runs are the $n^h = 0$ and theoretical lower bound of n^h, using the lowest the average invocation rate. After obtaining the estimated curve of public node cost, one can thus straightforwardly project the curve of total cost with respect to increasing in-house nodes.

E. Three Services

In this subsection, we consider a scenario where a service provider hosts three types of services, which correspond to service 2, 3, and 4 in Fig. 3, respectively. Fig. 5 summarizes the related performance metrics of interests. In this scenario, the public node premium is set to 75%. That is, the cost of a public node is calculated as 1.75 units of normalized cost.

Similar to the single service case, the total cost first decreases and then increases, with an increasing number of in-house nodes. The minimum cost is reached when n^h is around 27 and 30. The theoretical lower bound in Theorem 3.1 is around 23, using the lowest value of total invocation request rate as the minimum $\lambda(t)(=45.46)$ and the maximum value of $\mu_i(=2.5)$ as the average throughput. In contrast to the single service case, the allocation of nodes here can be shared across different services, via the control action of re-configuration. Consequently, the node allocation algorithm is more effective and the cost of public nodes thus decreases faster here than in the single service case. Moreover, the overall utilization is well maintained around the target value of 80%, and the response time shows a smoother decreasing trend with respect to increasing number of in-house nodes. The overall observations lead us to conclude that our proposed algorithm can indeed effectively allocate both types of nodes according to the workload dynamics of three services, their inherent performance variability, and their disparate cost features.

VI. Related Work

Cloud computing is an emerging paradigm, with flexibility and cost advantages stemming from pay-as-you-go billing schemes. To answer the question whether or not to move into cloud, several studies [10], [12], [13], [17] conduct in-depth analysis to quantify the cost of cloud computing. Bittencourt et al. [5] design a scheduling algorithm for the workflow of composed services in hybrid clouds, with the objective to minimize the total cost, resulting from the public nodes only. Kashef and Altmann [12] presented an overview of detailed cost factors of deployment of in-house nodes and proposed a cost model for hybrid cloud. Tak et al [17] compared different deployment models of cloud computing, using benchmark experiments and the concept of net present value. Their results show that allocating replicas of services on in-house nodes and public nodes can offer the best for hybrid cloud, especially for certain applications.

However, the aforementioned studies fall short in quantitatively incorporating the performance aspect, which may lead to a different decision in selecting the cloud deployment model and dimensioning the hybrid cloud. We provide a simple cost performance framework, which not only tries to minimize the cost but also maintains the target utilization and response times. Our framework is so general that cost factors identified in the aforementioned studies can be readily applied to our framework.

There is a growing interest to study the deployment of services on hybrid cloud, combining the private cloud and public cloud, especially from a cloud provider's perspective. The focus centers on the technologies enabling hybrid cloud and policies dispatching requests to a given set of static in-house nodes and dynamic nodes. Sotomayor et al. [16] discuss various virtual infrastructure and management tool kits for hybrid clouds, from a cloud provider's perspective. In particular, they advocate the OpenNebula architecture [4] combined with Haizea lease manager, which aims to meet SLA commitments with best-effort and configurable VM placement. Calheiros et al. [18] study how to dispatch a request among given in-house nodes and public nodes in hybrid clouds, such that the execution deadline of scientific applications is met. The in-house nodes in their study are based on computing grids and thus considered as a "free and unlimited resource". Zhang et al. [19] propose a factoring policy, which separate the requests into base and trespassing workloads and send them to in-house and public nodes respectively. The objective is to minimize the data cache/replication overhead.

In contrast, our study takes a service provider's perspective and aims to answer the orthogonal question of how to decide the optimal number of in-house nodes and public nodes dynamically, given a certain deployment architecture and request dispatching policy.

VII. Conclusion

There is a a growing interest to deploy service-oriented systems in hybrid clouds, which consist of a constant number of in-house nodes, as well as public nodes allocated in an on-demand fashion. In this paper, we develop a general cost-performance optimization framework to dimension the number of in-house nodes in hybrid clouds, such that the total cost is minimized and the desired QoS is met. Moreover, we design a dynamic collaborative node allocation algorithm, which not only leverages disparate characteristics of both nodes, i.e., performance variability, network delay, and cost structure, but also enforces the collaboration across services exhibiting different workload variabilities, via reconfiguration control actions.

Our evaluation results using production traces show that the proposed algorithm can achieve minimum cost for given target utilization and response time values in a hybrid cloud, where the number of in-house nodes is dimensioned based on the recommended design rules following the theoretical lower bound of the cost-performance framework.

Regarding ongoing research, we are exploring more complex service-oriented systems including composite services and different request dispatching policies.

References

[1] Amazon EC2. http://www.amazon.com/.

[2] ILOG CPEX Optimizer. http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/.

[3] National Institute of Science and Technology. The NIST Definition of Cloud Computing.

[4] OpenNebula. http://opennebula.org/.

[5] L. F. Bittencourt and E. R. M. Madeira. HCOC: A cost optimization algorithm for workflow scheduling in hybrid clouds. *J. Internet Services and Applications*, 2(3):207–227, 2011.

[6] M. Björkqvist, L. Y. Chen, and W. Binder. Dynamic Replication in Service-Oriented Systems. In *Proceedings of IEEE CCgrid*, 2012.

[7] M. Björkqvist, L. Y. Chen, and W. Binder. Opportunistic Service Provisioning in the Cloud. In *Proceedings of IEEE Cloud*, 2012.

[8] Y. Chen, A. Das, W. Qin, A. Sivasubramaniam, Q. Wang, and N. Gautam. Managing Server Energy and Operational Costs in Hosting Centers. In *Proceedings of ACM SIGMETRICS*, pages 303–314, 2005.

[9] A. Gandhi, M. Harchol-Balter, R. Das, and C. Lefurgy. Optimal power allocation in server farms. In *SIGMETRICS/Performance*, pages 157–168, 2009.

[10] M. Y. Hajjat, X. Sun, Y.-W. E. Sung, D. A. Maltz, S. G. Rao, K. Sripanidkulchai, and M. Tawarmalani. Cloudward bound: planning for beneficial migration of enterprise applications to the cloud. In *SIGCOMM*, pages 243–254, 2010.

[11] K. R. Jackson, L. Ramakrishnan, K. Muriki, S. Canon, S. Cholia, J. Shalf, H. J. Wasserman, and N. J. Wright. Performance analysis of high performance computing applications on the amazon web services cloud. In *CloudCom*, pages 159–168, 2010.

[12] M. M. Kashef and J. Altmann. A cost model for hybrid clouds. In *GECON*, pages 46–60, 2011.

[13] D. Kondo, B. Javadi, P. Malecot, F. Cappello, and D. P. Anderson. Cost-benefit analysis of cloud computing versus desktop grids. In *IPDPS*, pages 1–12, 2009.

[14] V. Petrucci, E. V. Carrera, O. Loques, J. C. B. Leite, and D. Mossé. Optimized management of power and performance for virtualized heterogeneous server clusters. In *CCGRID*, pages 23–32, 2011.

[15] H. Qian and D. Medhi. Server operational cost optimization for cloud computing service providers over a time horizon. In *Proceedings of the 11th USENIX conference on Hot topics in management of internet, cloud, and enterprise networks and services*, Hot-ICE'11, pages 4–4, 2011.

[16] B. Sotomayor, R. S. Montero, I. M. Llorente, and I. T. Foster. Virtual infrastructure management in private and hybrid clouds. *IEEE Internet Computing*, 13(5):14–22, 2009.

[17] B. C. Tak, B. Urgaonkar, and A. Sivasubramaniam. To move or not to move: The economics of cloud computing. In *Proceedings of the Third USENIX Workshop on Hot Topics in in Cloud Computing*, HOTCLOUD 2011, 2011.

[18] C. Vecchiola, R. Calheiros, D. Karunamoorthy, and R. Buyya. Deadline-driven provisioning of resources for scientific applications in hybrid clouds with aneka. *Future Gener. Comput. Syst.*, 28(1):58–65, Jan. 2012.

[19] H. Zhang, G. Jiang, K. Yoshihira, H. Chen, and A. Saxena. Intelligent workload factoring for a hybrid cloud computing model. In *SERVICES I*, pages 701–708, 2009.