log2cloud: Log-based Prediction of Cost-Performance Trade-offs for Cloud Deployments

Diego Perez-Palacin
Dpt. de Informática e
Ingeniería de Sistemas
Universidad de Zaragoza
Spain
diegop@unizar.es

Radu Calinescu
Department of
Computer Science
University of York
United Kingdom
radu.calinescu@york.ac.uk

José Merseguer
Dpt. de Informática e
Ingeniería de Sistemas
Universidad de Zaragoza
Spain
jmerse@unizar.es

ABSTRACT

Numerous organisations are considering moving at least some of their existing applications to the cloud. A key motivating factor for this fast-paced adoption of cloud is the expectation of cost savings. Estimating what these cost savings might be requires comparing the known cost of running an application in-house with a predicted cost of its cloud deployment. A major problem with this approach is the lack of suitable techniques for predicting the cost of the virtual machines (VMs) that a cloud-deployed application requires in order to achieve a given service-level agreement. We introduce a technique that addresses this problem by using established results from queueing network theory to predict the minimum VM cost of cloud deployments starting from existing application logs. We describe how this formal technique can be used to predict the cost-performance trade-offs available for the cloud deployment of an application, and presents a case study based on a real-world webmail service.

Categories and Subject Descriptors

C.4 [**Performance of Systems**]: [Modeling Techniques]; D.4.8 [**Performance**]: [Stochastic analysis]

Keywords

Cloud computing, cost, performance, log analysis

1. INTRODUCTION

The pay-as-you-go virtualised infrastructure offered by cloud providers represents an attractive alternative to using in-house resources to run an organisation's software applications. Cloud computing offers not only the prospect of major cost savings, but also a significant reduction in the expertise required to set up and run these applications [2, 18, 20]. Nevertheless, it is still difficult for an organisation to decide whether to choose a cloud versus an in-house deployment for an existing application portfolio. This is due both to concerns over the security, availability and legal implications of cloud deployments [3, 5, 9], and to uncertainties about the cost savings that they can actually achieve [19].

In this paper we focus on the second area of concern mentioned above, namely on the cost implications of moving an existing application to the cloud. Although multiple tools that assess these costs have been developed in recent years by both the research community [6, 8] and by cloud providers [1, 14], they all suffer from a major limitation: they take as input a specification of the resource needs of the analysed applications. Assuming that an organisation can provide such specifications is unrealistic, especially when the aim is to predict the minimum costs that can be achieved by varying the number of virtual machines (VMs) used for a cloud-deployed application in line with the application workload.

Our "log-to-cloud" (log2cloud) approach overcomes this limitation for request-handling applications such as web and email servers by using established queueing network theory results to calculate the VM resources required to achieve a given service-level agreement (SLA) starting from existing application logs. As many of these applications generate such logs, our solution eliminates the need for "guesstimating" the resource usage profile of an existing application when assessing the cost of its cloud deployment. Application logs represent an important source of valuable raw data for the management of IT systems [15] through performance optimisation [17], security analysis [11], and resource usage profiling and capacity planning [4].

Note that log2cloud focuses on VM costs because the other contributors to the overall cost of a cloud deployment are independent of the required SLA, and are trivial to calculate. For example, the data storage cost can be predicted easily based on the in-house data footprint of the application. Likewise, the I/O data transfer cost can be calculated from the overall amount of exchanged network traffic for the application, as recorded in the application log.

The application logs that log2cloud uses as input are logs that record the number of requests handled by the application during successive time intervals of equal duration. The other input parameters used by the approach are: (a) an application SLA that specifies a maximum response time and the probability with which this response time must be achieved; and (b) the cloud infrastructure performance (measured as described later in the paper) and cost.

A special feature of log2cloud is that it supports the calculation of cost estimates associated with a range of SLAs

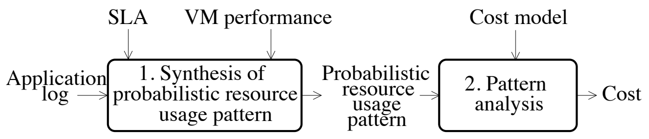

SLA
VM performance
Cost model
Application log
1. Synthesis of probabilistic resource usage pattern
Probabilistic resource usage pattern
2. Pattern analysis
Cost

Figure 1: The two-stage log2cloud approach

for the analysed application. This enables application owners to understand the different performance-cost tradeoffs available for a cloud deployment of their applications, and to choose tradeoffs that suits their needs—or to decide that a cloud deployment is not cost effective.

The main contributions of the paper are:

1. A new technique that uses queuing network theory to derive the (VM) *resource usage profile* of an application from its (a) request handling log and (b) required SLA.
2. An open-source implementation of the technique described above as a Java library that is easy to integrate with existing tools for analysing the cost of cloud deployments.
3. The integration of our Java resource usage profile generator with the tool for the analysis of cloud deployments we previously developed in [6].
4. A case study that illustrates the application of log2cloud to a real-world webmail service.

The rest of the paper starts with an overview of the log2cloud approach in Section 2, followed by the introduction of a running example in Section 3. We then describe the steps of the approach in Section 4, and the case study use to evaluate its effectiveness in Section 5. Section 6 describes related work, and Section 7 concludes the paper with a brief summary.

2. log2cloud OVERVIEW

The log2cloud approach produces a cost prediction for the cloud deployment of an application in two stages (Figure 1). In the first stage, log2cloud generates a formal, compact "profile" of the cloud resources required by the application. This profile is called a *probabilistic resource usage pattern* [6], and abstracts out the irrelevant details of the raw data from the application log into a concise, well defined and easy to manipulate format. The probabilistic resource usage patterns are then analysed in the second log2cloud stage by using the open-source "probabilistic pattern modelling" tool introduced in [6] and freely available on-line at `http://www1.aston.ac.uk/eas/staff/dr-kenneth-johnson/ppm/`. The outcome of this analysis is a cost prediction for the cloud deployment of the application.

Application log The first log2cloud stage handles application logs of the form

$$log = (n_1, n_2, \ldots, n_N), \tag{1}$$

where n_1, n_2, …, n_N represent the number of requests received by the application in N consecutive time intervals of length $t > 0$ each. The time period covered by the application log (i.e., $N \cdot t$) must extend over at least a few days in order to permit the extraction of a meaningful probabilistic resource usage pattern.

Another common type of application log supported by log2cloud is one that records the timestamp of each individual request received by the application. Since this raw data can be converted into the former type of log by counting the number of requests received within each time interval of length t, we will describe the operation of log2cloud only for the first type of log.

Note that many applications (including, for instance, the widely used Apache HTTP web server) generate logs that can be configured to record the request handling data described above.

Service-level agreement (SLA) We consider application SLAs of the form

$$sla = (r, p), \tag{2}$$

where $r > 0$ represents a response time and $p \in [0, 1]$ is a probability. An application or service is deemed to satisfy an SLA $sla = (r, p)$ if the probability of a request being handled within r time units is at least p. As an example, the SLA $sla_1 = (500\text{ms}, 0.95)$ is satisfied by an application if and only if the probability that the application handles a generic request within 500 milliseconds is greater than or equal to 0.95. Note that, in the long term, such an application will handle at least 95% of its requests within 500ms.

Virtual machine (VM) performance The performance of a VM is specified as the mean service rate

$$\mu > 0 \tag{3}$$

of an instance of the analysed application that is running on that VM. Obtaining this information involves installing an instance of the application on each different type of VM considered in the analysis, and carrying out a modest number of tests to measure the service rate on each such VM. The cost of the cloud resources used for these tests is negligible; whereas the effort to install and configure the application to run on cloud infrastructure, although significant, is required anyway (assuming that a cloud deployment is eventually adopted).

Cost model We consider a simple cost model in which the cost of using a VM for a time interval of length $T > 0$ (or for any part of such a time interval) is a constant

$$c > 0. \tag{4}$$

This way of modelling VM cost is consistent with the typical cost models of established cloud providers such as Amazon EC2. For simplicity, we will assume that the *costing time interval* T is a multiple of the *logging time interval* t (i.e., $T = kt$ for some integer value $k \geq 1$). We will also consider that SLA compliance is required for each time interval of duration T. For instance, an application will be deemed compliant with the SLA $sla_1 = (500\text{ms}, 0.95)$ if, within each time interval $[0, T)$, $[T, 2T)$, $[2T, 3T)$, …, it serves 95% or more of its requests in at most 500ms.

Probabilistic resource usage pattern The resource usage profiles synthesised during the first log2cloud stage and used as input by its second stage are encoded as probabilistic patterns with the following syntax taken from [6]:

```
Baseline $bl$
Rule Start $s_1$ Vary $v_1$ Repeat $f_1$ Until $u_1$
Rule Start $s_2$ Vary $v_2$ Repeat $f_2$ Until $u_2$        (5)
...
Rule Start $s_P$ Vary $v_P$ Repeat $f_P$ Until $u_P$
```

where $bl \geq 0$ represents a constant amount of resources that is used as a baseline for the pattern, and the parameters of the i-th rule, $1 \leq i \leq P$, have the following role:

- s_i specifies the (start) time when the rule applies, e.g., `Jul 2, 9am`;
- v_i defines the resource usage from time instant s_i onwards, through specifying the probabilities with which the resource usage varies with respect to either the baseline bl or the resource usage before the application of the rule, e.g., `[0.8:bl_add(10)+0.2:bl_add(15)]` specifies $bl + 10$ resource units with probability 0.8 and $bl + 15$ resource units with probability 0.2;
- $f_i \in \{$`Day`, `WeekDay`, `WeekEnd`, `Week`, `Month`$\}$ is a parameter that indicates the frequency with which the rule is applied;
- u_i is an optional parameter that specifies the end time for the application of rules that include a `Repeat` element, e.g., `Aug 31`.

As an example, consider an application that needs five virtual machines (VMs) at all times and, between 9am and 5pm on week days, 10 additional VMs with probability 0.8 and 15 additional VMs with probability 0.2. This resource usage profile can be encoded (for the period from 2^{nd} July to 31^{st} August) by using the probabilistic resource usage pattern

```
Baseline 5
Rule Start Jul 2, 9am  Vary [0.8:bl_add(10)+
                             0.2:bl_add(15)]
                       Repeat WeekDay Until Aug 31
Rule Start Jul 2, 5pm  Vary [1.0:bl]
                       Repeat WeekDay Until Aug 31
```

For a complete description of the probabilistic resource usage patterns used by log2cloud please see [6].

3. RUNNING EXAMPLE

To illustrate the application of log2cloud we will use the University of Zaragoza's webmail service as a running example. This service handles over one million requests on an average day, and its log entries (available on-line at `https://webmail.unizar.es/mail_monitor.php`) show to the number of requests handled during each minute of operation.

The graph in Figure 2 shows a typical minute-by-minute variation in the number of requests handled by the webmail service over a 12-day time period. The workload peaks and troughs in this graph correspond to the day-time and night-time usage of the service, respectively. The significant difference between the two types of workload suggests that moving the service to cloud may lead to cost savings, as much fewer VMs should be required during workload troughs compared to workload peaks.

The application log in our running example is $log_{webmail} = (n_1, n_2, \ldots, n_{17280})$, where n_i, $1 \leq i \leq 17280$, represents the number of requests handled by the webmail service during the i-th minute of operation from the 12 days shown in Figure 2. Using the notation introduced in the previous section, we have a logging time interval of length $t = 1$ minute, and $N = (12 \text{ days}/1 \text{ minute}) = 17280$. In the remainder of the paper, we will use $log_{webmail}$ to analyse the cost of moving the webmail service to cloud infrastructure. This analysis will be carried out for a range of realistic SLAs and VM parameters.

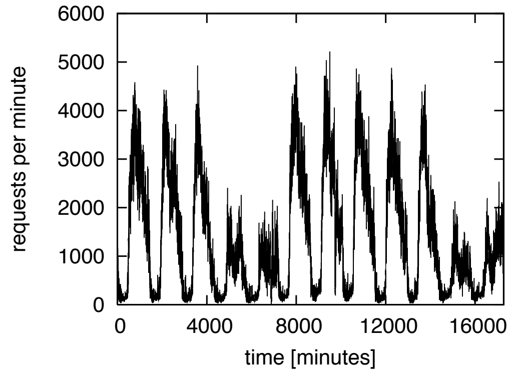


Figure 2: Number of requests per minute handled by the webmail service over the 12-day period between Wed. 22 Feb. and Sun. 4 Mar. 2012.

4. log2cloud DESCRIPTION

4.1 Stage 1: Synthesis of probabilistic resource usage pattern

The pattern synthesis stage of the log2cloud approach comprises three steps:

1a The *resource trace extraction* step calculates how many VMs are required to satisfy the application SLA during each costing time interval of size $T = kt$, over the duration of the analysed log.

1b The *resource trace partition* step splits the resource trace described above into *sub-traces* associated with time periods characterised by similar resource usage.

1c The *resource profile synthesis* step uses the sub-traces in step 1b to build a probabilistic resource usage pattern (5).

4.1.1 Step 1a: Resource trace extraction

This step uses the information provided by the application log (1), the SLA (2) and the VM performance (3) to calculate, if feasible, the resource trace

$$trace = (m_1, m_2, \ldots, m_{N/k}), \tag{6}$$

where $m_i \geq 0$, $1 \leq i \leq N/k$, represents the minimum number of VMs that the application must be deployed on during the i-th costing time interval of size T. Figure 3 shows the relationship between the log entries n_1, n_2, …, n_N and the resource trace entries m_1, m_2, …, $m_{N/k}$.

To calculate the values $m_i \geq 0$, $1 \leq i \leq N/k$, we use a standard queueing theory result. This result states that in an $M/M/1$ queue with request arrival rate λ and service rate $\mu > \lambda$, the probability that the response time τ of a request is less than r is:

$$P(\tau \leq r) = 1 - e^{(\lambda - \mu)r}$$

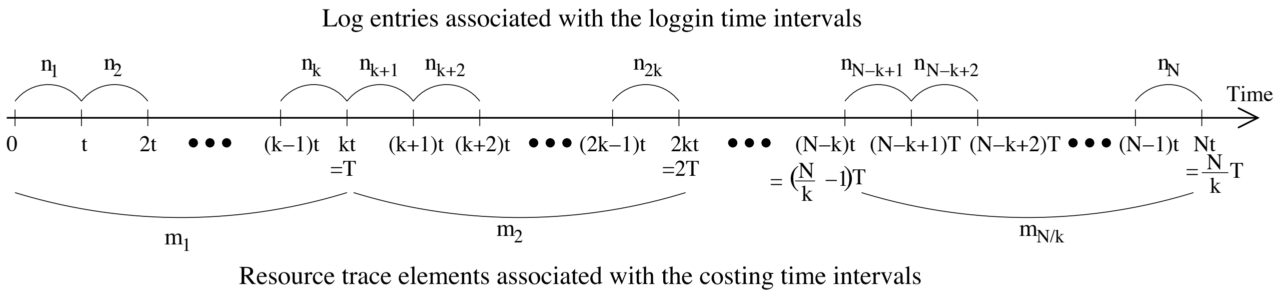


Figure 3: The relationship between the log entries and the resource trace entries.

or, if $m > 0$ independent application instances are used,

$$P(\tau \leq r) = 1 - e^{(\lambda/m - \mu)r}, \tag{7}$$

where λ/m is the request arrival rate for one application instance, assuming that incoming requests are randomly distributed among the independent queues of these instances. Note that the probability in (7) is upper bounded by $1 - e^{-\mu r}$, which corresponds to the scenario in which an "infinite" number of application instances are used. This limits the range of SLAs (r, p) that can be achieved using VMs with service rate μ to those SLAs for which

$$p < 1 - e^{-\mu r} \tag{8}$$

If $n \geq 0$ requests are handled by the application, then the expected number of requests that are handled each in at most r time units is

$$n \cdot P(\tau \leq r) = n\left(1 - e^{(\lambda/m-\mu)r}\right).$$

To use this result in the calculation of m_i, $1 \leq i \leq N/k$, we consider the log entries associated with the i-th costing time interval of size T, namely $n_{(i-1)k+1}$, $n_{(i-1)k+2}$, ..., n_{ik}. We assume that each of these log entries corresponds to a time period during which the requests arrived with exponentially distributed inter-arrival time[1]. The associated arrival rate is:

$$\lambda_j = n_j/t,$$

where $(i-1)k + 1 \leq j \leq ik$. Accordingly, the expected number of requests handled within r time units during the i-th costing time interval when m_i VMs are used is:

$$\sum_{j=(i-1)k+1}^{ik} n_j\left(1 - e^{\left(\frac{\lambda_j}{m_i}-\mu\right)r}\right) = \sum_{j=(i-1)k+1}^{ik} n_j\left(1 - e^{\left(\frac{n_j}{m_i t}-\mu\right)r}\right).$$

and the expected fraction of requests that satisfy the same property is

$$f(m_i) = \frac{\sum_{j=(i-1)k+1}^{ik} n_j\left(1 - e^{\left(\frac{n_j}{m_i t}-\mu\right)r}\right)}{\sum_{j=(i-1)k+1}^{ik} n_j}.$$

As mentioned earlier, we deem that the SLA $sla = (r, p)$ is satisfied by a cloud-deployed version of the application iff

$$f(m_i) \geq p,$$

for $1 \leq i \leq N/k$. We are interested in the smallest number of VMs m_i that satisfies this inequality, so we choose m_i, $1 \leq i \leq N/k$, such that:

$$f(m_i) \geq p > f(m_i - 1).$$

[1]To offer a reliable assessment, this assumption requires a sufficient small t, which is the case in our running example.

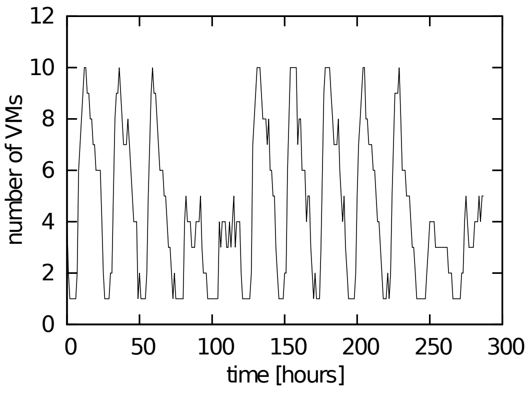


Figure 4: Resource trace corresponding to the $log_{webmail}$ application log from the running example

Assuming that constraint (8) is satisfied, the solution of this inequality belongs to the interval $[\underline{m}_i, \overline{m}_i]$, where $\underline{m}_i$ and $\overline{m}_i$ represent the numbers of VMs required to satisfy the SLA for the minimum and maximum request arrival rates in the i-th costing time interval, respectively. Therefore, we first obtain $\underline{m}_i$ and $\overline{m}_i$ by calculating the value m that satisfies eq. (7) for $\lambda = \min_{(i-1)k+1 \leq j \leq ik} n_j/t$ and $\lambda = \max_{(i-1)k+1 \leq j \leq ik} n_j/t$, respectively, and then solve the inequality above using a standard iterative approximation method based on a binary search within the interval $[\underline{m}_i, \overline{m}_i]$.

Example 1 Consider the webmail service log $log_{webmail}$ from our running example, and suppose that the length of the costing time intervals is $T = 1$ hour. Accordingly, we have $k = T/t = (1 \text{ hour}/1 \text{ minute}) = 60$, so the resource trace generated in this step of the approach will comprise $N/k = 17280/60 = 288$ elements: $trace_{webmail} = (m_1, m_2, \ldots, m_{288})$. For example, assuming a VM service rate $\mu = 10s^{-1}$ and an SLA $sla = (1.5s, 0.99)$, the resource trace will look like the graph in Figure 4.

4.1.2 Step 1b: Resource trace partition

This step partitions the resource trace $(m_1, m_2, \ldots, m_{N/k})$ from eq. (6) into $P \geq 1$ *sub-traces* associated with (disjoint) time intervals with similar resource requirements. The generic form of the i-th sub-trace, $1 \leq i \leq P$, is:

$$subtrace_i = (m_{i_1}, m_{i_2}, \ldots, m_{i_x}), \tag{9}$$

where $1 \leq i_1 < i_2 < \ldots < i_x \leq N/k$. Obtaining these sub-traces involves:

i) dividing the time period into P sets of (costing) time intervals $tset_1$, $tset_2$, ..., $tset_P$, so that the elements in the same set share a common time-related characteristic;

ii) placing the trace elements m_1 to $m_{N/k}$ into disjoint *sub-traces* associated with these P sets of timing intervals.

Several such partitions need to be analysed, to identify a partition with sub-traces comprise elements whose variance does not exceed a threshold. Partitions that we found particularly useful, from our experiments, are those that (a) can be mapped easily to a probabilistic resource usage pattern (5); and (b) correspond to the workload periodicities of typical applications. A (non-exhaustive) set of such partitions is described below:

- *"Time of day"* is suitable for workloads that follow a pattern during each day. In this case $P = (24 \text{ hours}/T)$, and $tset_i$ comprises the i-th costing time interval for each day.[2]
- *"Time of day & type of day"* is suitable for workloads that follow a pattern during each working day, and a different pattern during each weekend day. In this case $P = 2 \times (24 \text{ hours}/T)$, $tset_i$ comprises the i-th costing time interval for each working day, and $tset_i$, $P/2 + 1 \leq i \leq P$, comprises the $(i - P/2)$-th costing time interval for each weekend day.
- *"Time of day & day of month"* partition comprises $(24 \text{ hours}/T)$ sub-traces for each day of month, so $P = 31 \times (24 \text{ hours}/T)$. $tset_i$ comprises the $(i \mod (24 \text{ hours}/T))$-th time interval from each $\lfloor i/(24 \text{ hours}/T) \rfloor$-th month day. Only months with complete traces (i.e., 30 or 31 tracked days) can contribute.

Our strategy for identifying a good partition assesses the partitions above in increasing P order, and selects the first partition whose sub-traces do not exceed a variance threshold. The rationale is that a partition with few sub-traces yields a probabilistic resource usage pattern that is more concise and therefore easier to analyse. When no partition can be selected, log2cloud uses that whose sub-traces exceed the threshold the least.

Example 2 In our running example, the duration of a costing time interval is $T = 1$ hour. Accordingly, the *"time of day"* partition has $P = 24$ sub-traces (shown in Fig. 5a), the *"time of day & type of day"* partition has 48 sub-traces (shown in Fig. 5b), etc. Suppose that a threshold of 6.0 is chosen for the maximum allowed sub-trace variance. This criterion is not satisfied by the sub-traces from the *"time of day"* partition, e.g., the variance of $subtrace_{13} = (10, 9, 9, 4, 4, 10, 10, 10, 10, 10, 4, 3)$ (corresponding to the time interval between 12pm and 1pm) has a variance of 8.19. However, the variance of each of the 48 sub-traces from the *"time of day & type of day"* partition is below 1.69, so this partition can be selected and used in the next log2cloud step.

4.1.3 Step 1c: Resource profile synthesis

This log2cloud step synthesises a P-rule probabilistic resource usage pattern (5) from the resource trace partition $subtrace_1$, $subtrace_2$, …, $subtrace_P$. The baseline bl and the i-th rule of this pattern, $1 \leq i \leq P$, are obtained as follows:

`Baseline` bl. We select as baseline the statistical mode of one of the sub-traces with the lowest variance:

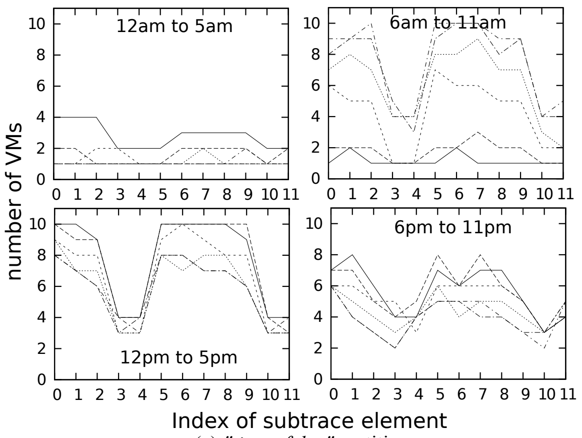
$$bl = mode(subtrace_x),$$

where $x \in \arg\min_{1 \leq i \leq P} Var(subtrace_i)$. This choice ensures that bl represents a value that occurs frequently in the resource trace partition.

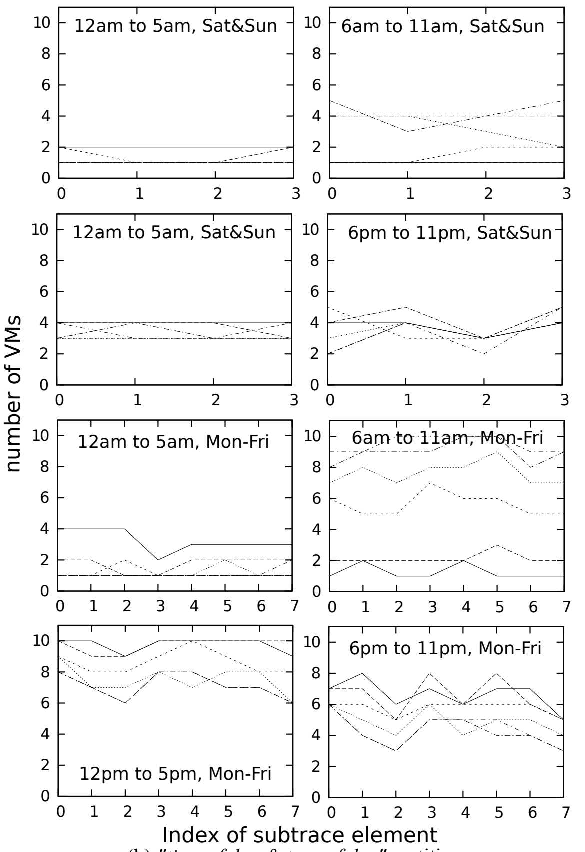
`Rule Start` s_i `Vary` v_i `Repeat` f_i `Until` u_i. The parameters s_i, v_i, f_i and u_i of this rule are extracted from $subtrace_i = (m_{i_1}, m_{i_2}, \ldots, m_{i_x})$ as follows:

- s_i represents the start of the costing time interval associated with m_{i_1}, i.e., the oldest element in the sub-trace;

[2]We assume that 24 hours is a multiple of the costing time period T.



(a) *"time of day"* partition



(b) *"time of day & type of day"* partition

Figure 5: Sub-traces for the running example.

- v_i is obtained by calculating the frequency of each distinct element from $subtrace_i$. Assuming that the distinct elements from $subtrace_i$ are $y_1, y_2, \ldots, y_q$ for some $1 \le q \le x$, and that their frequencies are $freq_1$, $freq_2$, $\ldots$, $freq_q$, the resulting v_i will have the form

$$v_i = \texttt{[}freq_1\texttt{:}op_1\texttt{+}freq_2\texttt{:}op_2\texttt{+}\ldots\texttt{+}freq_q\texttt{:}op_q\texttt{]}$$

where, for $1 \le j \le q$,

$$op_j = \begin{cases} \texttt{bl\_add}(y_j - bl) & \text{if } y_j > bl \\ \texttt{baseline} & \text{if } y_j = bl \\ \texttt{bl\_sub}(bl - y_j) & \text{otherwise} \end{cases}$$

- f_i depends on the type of partition used to obtain the P sub-traces and, for the *"time of day & type of day"*, on the value of i:

$$f_i = \begin{cases} \texttt{Day} & \text{if partition=}\textit{"time of day"} \\ \texttt{WeekDay} & \text{if partition=}\textit{"time of day \& type of day"} \\ & \quad \text{and } i \le P/2 \\ \texttt{WeekEnd} & \text{if partition=}\textit{"time of day \& type of day"} \\ & \quad \text{and } i > P/2 \\ \texttt{Week} & \text{if partition=}\textit{"time of day \& day of week"} \\ \texttt{Month} & \text{if partition=}\textit{"time of day \& day of month"} \end{cases}$$

The rationale behind *"time of day & day of week"* is analogous to *"time of day & day of month"*.

- The end time parameter u_p is set to a date far enough from the start time s_p, for example six months or one year. This enables the calculation of the expected cost of a cloud deployment over a longer time interval.

Example 3 We return to our running example and the *"time of day and type of day"* resource trace partition selected for it in the previous log2cloud step described in Example 2. This trace partition comprises 48 sub-traces, so the associated probabilistic resource usage pattern has 48 `Rules`. The sub-trace with the lowest variance within the partition is $subtrace_4 = (1, 1, 1, 1, 1, 1, 1, 1)$, so the baseline used for the pattern is $bl = mode(subtrace_4) = 1$. To generate the `Rule` associated with this sub-trace, we note that the distinct elements in $subtrace_4$ are $y_1 = 1$, and that its frequency is $freq_1 = 8/8 = 1.0$. Accordingly, the associated rule is:

```
Rule Start Feb 22, 4am Vary [1.0:baseline]
                              Repeat WeekDay Until Dec 31
```

where `Feb 22` represents the start of the first costing time interval in $subtrace_4$, the end time was chosen to be the end of the year, and the `WeekDay` frequency was obtained using the above definition for f_i. The other 47 `Rules` for this pattern are obtained in a similar way. An excerpt of the overall probabilistic resource usage pattern generated by this step is shown below:

```
Baseline:  1
Rule Start Feb 22, 0am Vary [0.125:baseline_add(1)
    +0.5:baseline_add(2) + 0.375:baseline_add(3)]
                 Repeat WeekDay Until Dec 31
...
Rule Start Feb 22, 4am Vary [1.0:baseline]
                 Repeat WeekDay Until Dec,31
...
Rule Start Feb 22,12pm Vary [0.25:baseline_add(8)
                          + 0.75:baseline_add(9)]
                 Repeat WeekDay Until Dec 31
...
Rule Start Feb 25, 12pm Vary [1.0:baseline_add(3)]
                 Repeat WeekEnd Until Dec 31
Rule Start Feb 25, 1pm Vary [0.25:baseline_add(2)
                         + 0.75:baseline_add(3)]
                 Repeat WeekEnd Until Dec 31
...
```

4.2 Stage 2: Pattern analysis

Once the probabilistic resource usage pattern has been synthesized in the first log2cloud stage, during the second stage the pattern is analyzed to evaluate the VM cost of the cloud deployment. This analysis applies the method introduced in [6], which comprises two steps:

i. translating the probabilistic resource usage pattern into an equivalent Markov decision process (MDP);
ii. using the quantitative model checking techniques presented in [6] to evaluate the overall amount of VM resources associated with this MDP (particularly, we have used methods that calculate the cumulative function), and with the time period for which the cloud deployment of the original application is analyzed.

The result of the evaluation from the second step is the total number of VM costing time intervals for which the application owner needs to pay. Accordingly, the total VM cost is obtained by multiplying this value by the cost $c > 0$ of using one VM for a single time interval. Due to space limitations we do not present these steps here. For a detail description of the method and its underpinning theory, the reader is referred to [6].

Example 4 Consider again the probabilistic resource usage pattern generated for our webmail service in Example 3. We used this pattern as input for the analysis tool from [6], and we selected the time period between 22 February and 31 December as the time period to analyze. The result of the analysis indicated that the application needs to use 34,115.25 hours of VM resources in order to achieve the SLA chosen in Example 1 (i.e., $sla = (1.5s, 0.99)$). Assuming that the cost of using a VM for a one-hour costing time interval is \$0.66, the predicted overall VM cost for the cloud deployment of the webmail service is \$22,516.065. The predicted mean monthly cost for the cloud deployment is \$22,516.065$\times$ (30 days / 313 days) =\$2,158.09 (since there are 313 days between 22 February and 31 December 2012, and assuming that a month has 30 days on average).

5. IMPLEMENTATION AND CASE STUDY

We implemented the log2cloud approach as an open-source Java library, in order to facilitate integration with other tools for assessing the implications of migrating applications to cloud. Also, we developed a prototype log2cloud tool that uses this library to predict the VM costs of deploying an application on cloud infrastructure, starting from an existing application log and an SLA that the application is required to achieve. The log2cloud Java library and the prototype tool are freely available and can be downloaded from [12].

To evaluate the log2cloud approach and its implementation, we used the log2cloud tool in a case study. Our case study assessed the VM costs of deploying the webmail service from our running example on different types of Amazon

Table 1: VM performance and cost

	Amazon EC2 instance type		
	Standard Small (S)	*High-CPU Medium* (M)	*High-CPU Extra Large* (XL)
performance μ	2.5s^{-1}	5s^{-1}	10s^{-1}
cost c	\$0.08	\$0.165	\$0.66

Table 2: Mean monthly VM costs for a cloud deployment of the webmail service. The lowest cost for each SLA is shaded, and dashes ('—') are used to mark VM–SLA combinations that are unfeasible (i.e., the constraint in eq. (8) is not satisfied).

(a) results for $sla = (r, 0.99)$

	response time r		
VM type	1.0s	1.5s	2.0s
S	—	—	\$4025.09
M	\$6353.42	▨ \$1628.75	▨ \$1228.39
XL	▨ \$2561.33	\$2157.41	\$2033.86

(b) results for $sla = (r, 0.999)$

	response time r		
VM type	1.0s	1.5s	2.0s
S	—	—	—
M	—	\$6373.62	▨ \$2018.65
XL	▨ \$4153.25	▨ \$2637.36	\$2323.73

EC2[3] *instances* (i.e., virtual machines), for a range of SLAs. The three types of Amazon EC2 instances considered in our experiments are shown in Table 1. The VM costs in this table are the actual costs of Amazon EC2 instances (as of 26th September 2012). A plausible value was chosen for the VM performance of the *Standard Small* EC2 instance ([7] suggests best practices for measuring the actual VM performance). The VM performance for each of the other VM types was estimated based on the relative difference in performance between that VM type and the Standard Small VM type (e.g., as measured in some of the benchmarking experiments in [7]).

We considered six possible SLAs, corresponding to achieving a response time of at most 1.0s, 1.5s and 2.0s with probability 0.99 and 0.999, i.e.,

$$sla = (r, p) \in \{1.0, 1.5, 2.0\} \times \{0.99, 0.999\}$$

and we used the application log from the running example (shown in Figure 2).

Table 2 shows the results of applying the log2cloud VM cost-performance prediction approach to the application log, range of SLAs and VM types described above, i.e., probabilistic pattern synthesis and subsequent pattern analysis using theoretical results in [6]. Analyzing these results we could draw the following useful conclusions about a possible cloud deployment of the webmail service, and about deploying an application on cloud infrastructure in general:

1. Demanding SLAs cannot be achieved using low-performance VMs, irrespective of how many such VMs are used. For instance, *Standard Small* EC2 instances (i.e., VMs of type S in Table 2) can achieve only the least demanding of the six SLAs in our case study, sla = (2.0s,0.99). This is due to the fact that the performance μ of this type of VM does not satisfy the constraint in eq. (8) for any of the other five SLAs.
2. The lowest cost for a given SLA does not necessarily correspond to the least expensive type of VM that can be used to achieve that SLA. For example, the lowest cost to achieve the SLA $sla = (1.0\text{s}, 0.99)$ for the webmail service in our case study is obtained when VMs of type XL are used; using VMs of type M to achieve the same SLA would cost almost three times as much.
3. Reducing the response time r of the application SLA for a fixed probability p leads to an exponential increase in VM costs. As an example, reducing the response time r of our webmail service from 2.0s to 1.5s, and from 1.5s to 1.0s for $p = 0.999$ corresponds to cost increases of slightly over \$600 and \$1,500, respectively.
4. The cost implications of not choosing the most cost-effective type of VM for a cloud deployment are significant. All three types of VM in our case study can achieve the SLA $sla = (2.0\text{s}, 0.99)$, but the differences between the optimal monthly cost (corresponding to VMs of type M) and the monthly costs for deployments using the other two VM types are almost \$2,800 and \$800, respectively. Note that these costs could be much higher in the worst-case scenario in which a VM type that cannot satisfy an SLA is chosen for a deployment, and the number of such VMs used to run the application is scaled up automatically in an (impossible) attempt to achieve this SLA.

These insights demonstrate the usefulness of the log2cloud approach to assessing the (VM) cost-performance trade-offs of cloud deployments.

6. RELATED WORK

Very limited support is currently available for organizations interested in analyzing the possible costs of running their existing applications in the cloud. Although all major cloud providers offer calculators to estimate the costs of such deployments (e.g., Amazon [1], Windows Azure [14], and Rackspace [16]), the capabilities of these calculators is limited. In particular, they tend to consider that a fixed number of active VMs are used at all times, although a key characteristic of cloud deployments is to save costs by varying the number of active VMs in line with application demand. Although the Amazon [1] and Rackspace [16] calculators also allow the specification of the number of hours per month when a set of VMs is estimated to be active, no support is available for users to obtain these estimates for their applications. Our approach avoids the need for guessing the VM needs of an existing application by using the logs of that application to predict the cost of its cloud deployment.

Research on predicting the cost-performance trade-offs of deploying an existing application on cloud infrastructure is, to the best of our knowledge, very limited. The approach in [10] estimates the costs of the cloud and in-house delopyments of an application by considering the expected and unexpected changes in the application workload. The cost metric used in [10]—although cumulative over the analyzed time period like the one used by log2cloud—is too coarse-

[3]`http://aws.amazon.com/ec2/`

grained to support the detailed analysis that is possible with log2cloud (cf. Section 5). In [13], the authors present an experiment that assesses the VM costs for multiple SLAs (expressed as utility functions) when capacity planning techniques are used to vary the number of VMs so that these SLAs are achieved. This approach can provide accurate results, but is limited to scenarios in which it is possible to actually run the application on the cloud, which makes it unsuitable for the scenarios targetted by log2cloud i.e., assessing cloud deployment costs for existing applications prior to deploying them on cloud infrastructure. Finally, the cost calculators from the "cloud adoption toolkit" in [8] and from the project in [6] are sophisticated versions of the calculators made available by cloud providers, since they also require the user to specify their VM resource needs as input. Our log2cloud used the results from [6] in its second stage, but starts from an existing application log in its calculation of the costs of a cloud deployment.

7. CONCLUSION

We introduced a new method that, based on queueing network theory, extracts a resource usage profile from an existing application log, and uses this profile to predict the VM costs for a cloud deployment of the application. We implemented the method as a Java library, and developed a prototype tool for analyzing the (VM) cost-performance trade-offs of cloud deployments. The case study presented in the paper showed that this tool can be used to gain useful insights into the options available for the cloud deployment of an existing application. An assumption of this study is that VMs operate independently and do not impact each other, even if it is not always the case nowadays. However, significant progress has been made in virtualization technology to improve this issue, and more progress is expected.

The overall cost of running an application in the cloud includes many other components, for example, data storage costs and I/O data transfer costs. This paper focused only on VM costs because varying the type and number of VMs used for a cloud-deployed application has a major impact on the SLA that the application will achieve. In future work, we aim to experiment with real applications. Moreover, we plan to extend the log2cloud analysis to also cover the other components of the cost of cloud deployments, e.g., through integrating it with the cost calculator from the toolkit in [8]. Finally, we will apply the extended toolkit to the cloud deployment of a real benchmark application.

Acknowledgements

This work was partly supported by the UK Engineering and Physical Sciences Research Council grant EP/H042644/1 and by Spanish CICYT DPI2010-20413

8. REFERENCES

[1] AWS. Simple Monthly Calculator, 2012. `http://calculator.s3.amazonaws.com/calc5.html`.

[2] M. Armbrust et al. A view of cloud computing. *Commun. ACM*, 53:50–58, 2010.

[3] R. Buyya et al. Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Generation Computer Systems*, 25(6):599–616, 2009.

[4] A. Ganapathi et al. Statistics-driven workload modeling for the cloud. In *IEEE Intl. Conf. on Data Engineering Workshops*, pages 87–92, 2010.

[5] M. Jensen, J. Schwenk, N. Gruschka, and L. Iacono. On Technical Security Issues in Cloud Computing. In *IEEE Intl. Conf. on Cloud Computing*, pages 109–116, 2009.

[6] K. Johnson, S. Reed, and R. Calinescu. Specification and quantitative analysis of probabilistic cloud deployment patterns. In *Hardware and Software: Verification and Testing*, volume 7261 of *LNCS*, pages 145–159. Springer, 2012.

[7] K. Sarda, S. Sanghrajka, and R. Sion. Cloud Performance Benchmark Series: Amazon E2C CPU Speed Benchmarks. Technical report, Stony Brook University, 2010.

[8] A. Khajeh-Hosseini, D. Greenwood, J. W. Smith, and I. Sommerville. The cloud adoption toolkit: supporting cloud adoption decisions in the enterprise. *Softw.: Practice and Exper.*, 42(4):447–465, 2012.

[9] A. Khajeh-Hosseini, D. Greenwood, and I. Sommerville. Cloud migration: A case study of migrating an enterprise IT system to IaaS. In *IEEE Intl. Conf. on Cloud Computing*, pages 450–457, 2010.

[10] M. Klems, J. Nimis, and S. Tai. Do Clouds Compute? A Framework for Estimating the Value of Cloud Computing. In *Designing E-Business Systems. Markets, Services, and Networks*, volume 22 of *LNBIP*, pages 110–123. Springer Berlin Heidelberg, 2009.

[11] K. Kowalski and M. Beheshti. Analysis of Log Files Intersections for Security Enhancement. In *3rd Intl. Conf. on Information Technology: New Generations*, pages 452–457. IEEE Computer Society, 2006.

[12] Log2cloud tool 2012. `http://webdiis.unizar.es/GISED/?q=tool/log2cloud`.

[13] D. A. Menascé and P. Ngo. Understanding cloud computing: Experimentation and capacity planning. In *Computer Measurement Group Conference*, 2009.

[14] Microsoft. Windows Azure Calculator, 2012. `http://www.windowsazure.com/en-us/pricing/calculator/?scenario=full`.

[15] A. Oliner, A. Ganapathi, and W. Xu. Advances and challenges in log analysis. *Commun. ACM*, 55(2):55–61, Feb. 2012.

[16] Rackspace. Pricing and calculator, 2012. `http://www.rackspace.com/cloud/public/servers/pricing`.

[17] J. Schaefer et al. Performance-aware design and optimization of enterprise applications. In *6th Intl. Conf. on Network and Service Management*, pages 306–309, 2010.

[18] L. M. Vaquero, L. Rodero-Merino, J. Caceres, and M. Lindner. A break in the clouds: towards a cloud definition. *SIGCOMM Comput. Commun. Rev.*, 39:50–55, 2008.

[19] E. Walker. The real cost of a CPU hour. *Computer*, 42(4):35–41, 2009.

[20] L. Youseff, M. Butrico, and D. Da Silva. Toward a unified ontology of cloud computing. In *Grid Computing Environments Workshop*, pages 1–10, 2008.