# Specification and Quantitative Analysis of Probabilistic Cloud Deployment Patterns

Kenneth Johnson, Simon Reed, and Radu Calinescu

Computer Science Research Group, Aston University, Birmingham B4 7ET UK
{k.h.a.johnson,reeds,r.c.calinescu}@aston.ac.uk

*The probable is what usually happens.*

Aristotle

**Abstract.** Cloud computing is a new technological paradigm offering computing infrastructure, software and platforms as a pay-as-you-go, subscription-based service. Many potential customers of cloud services require essential cost assessments to be undertaken before transitioning to the cloud. Current assessment techniques are imprecise as they rely on simplified specifications of resource requirements that fail to account for probabilistic variations in usage. In this paper, we address these problems and propose a new *probabilistic pattern modelling* (PPM) approach to cloud costing and resource usage verification. Our approach is based on a concise expression of probabilistic resource usage patterns translated to *Markov decision processes* (MDPs). Key costing and usage queries are identified and expressed in a probabilistic variant of temporal logic and calculated to a high degree of precision using quantitative verification techniques. The PPM cost assessment approach has been implemented as a Java library and validated with a case study and scalability experiments.

**Keywords:** Cloud computing, formal verification methods, formal specification languages, formal modelling and specification, probabilistic model checking, Markov processes, costing analysis, resource usage patterns.

## 1 Introduction

Cloud computing can be succinctly described as *computing as a service* [1,22,20] where software, platforms and virtualised hardware are available on-demand on a pay-as-you-go basis. The elastic nature of the cloud enables customers to adapt service usage to meet fine-grained variations of their resource requirements by dynamically scaling their computing services up or down. This situation is economically favourable in comparison to making large initial investments on infrastructure based on requirements for peak demand. Despite the envisioned benefits of cloud computing, there are still barriers to its adoption. Alongside concerns such as cloud security [10], reliability and compliance with data protection laws [11], many potential customers are reticent due to an inability to accurately express and analyse their resource requirements.

The most attractive feature of cloud computing is the ability to dynamically scale resources up or down over fine-grained time intervals. As a result, cloud requirements are often thought about in terms of *patterns of usage*, where resource requirements vary over time. Resource usage can change due to small variances in workload or changing economic situations such as fluctuations in a cloud provider's prices and the customer's capital income.

These types of resource usage patterns are inherently *probabilistic* in nature and involve potentially unknown or *non-deterministic* factors, making requirements specification difficult, and existing cost assessment methods less accurate. Current cost modelling tools employ cloud usage patterns that disregard the probabilistic nature of resource usages resulting from the cloud's dynamic scalability [13,3]. This leads to a naive cost assessment, where probabilistic behaviours do not play a role in determining cost. Instead, resource usage is simplified to follow either constant rates of change or variations over coarser time intervals. To overcome these limitations we propose a new *probabilistic pattern modelling* (PPM) approach for the expression of resource requirements as *probabilistic patterns*, and the application of *quantitative verification* techniques to analyse a wide range of cost-related characteristics of potential cloud deployments.

Probability has been used to model unreliable or unknown behaviour in both hardware and software systems. Thanks to the development of effective probabilistic modeller checkers such as PRISM [17], MRMC [12] and RAPTURE [9], quantitative verification has found applicability in a wide range of application domains. Typical applications include verification of QoS properties in service-based systems [5] and run-time model checking to guide self-optimisation strategies in software systems [6,4]. Most recently, probabilistic modelling was used for performance analysis of live migration of virtual machines between physical servers in a cloud data centre [15].

Our approach employs quantitative verification techniques [18,19] to enable potential customers of cloud services to check two classes of quantitative properties of a cloud deployment

- **costs:** to determine a deployment's variation of costs over time, and to calculate the maximum accumulated costs owed to a cloud provider at the end of a billing period, and
- **resource usage:** to determine the maximum and minimum probabilities that a deployment's resource usage exceeds a certain threshold.

The main contributions of our work are:

1. A high-level language for the specification of probabilistic and non-deterministic patterns of cloud resource usage.
2. Techniques to synthesise Markov decision process (MDP) models from resource usage patterns and to formalise resource usage and cost properties as rewards-augmented probabilistic computation tree logic (PCTL) formulae [8].
3. An implementation of our PPM approach as an open-source Java library.
4. A case study and scalability experiments to validate the approach.

The paper is organised as follows. Section 2 provides background information on Markovian models, property specification and probabilistic model checking. Section 3 presents the steps of the PPM approach in detail. A grammar for probabilistic patterns is given and we describe an algorithm for MDP synthesis. Properties for cost and resource usage analysis are formalised as probabilistic temporal logic. Section 4 introduces our prototype implementation of the PPM approach and presents a case study and scalability experiments. Section 5 discusses related research and Section 6 summarises our results and suggests directions for future work.

## 2 Background

A *Markov decision process* (MDP) is a tuple

$$M = (S, s_0, Act, Dist, step, L) \tag{1}$$

where $S = \{s_0, \ldots, s_n\}$ is a finite set of states, $s_0$ the initial state, $Act$ a set of actions, $Dist$ a set of probability distributions over the states in $S$, $step : S \rightarrow 2^{(Act \times Dist)}$ is a probabilistic transition function that maps the states in $S$ to finite sets of action-distribution pairs, and $L : S \rightarrow 2^{AP}$ is a map labelling individual states with finite subsets of properties from an atomic proposition set $AP$.

*Probabilistic model checking* [18] is a technique for building specifications of systems exhibiting probabilistic behaviour and determining the satisfiability of their quantitative properties. For the analysis of MDP models, properties are specified in a probabilistic temporal logic PCTL extending computation tree logic CTL with a probabilistic operator $P_{\bowtie p}$, for $p \in [0, 1]$ and $\bowtie \in \{<, \leq, \geq, >\}$. A wide range of quantitative properties for Markovian models can be specified in this logic. For example "the probability of a cloud deployment eventually requiring $x$ or more resources is less than $p$" can be specified by the PCTL formula $P_{<p}[F\ res \geq x]$. For model checking MDPs, the operators $Pmin_{\bowtie p}$ and $Pmax_{\bowtie p}$ determine the minimum and maximum probabilities over all adversaries (all resolutions of the non-determinism induced by $Dist$ from (1)), respectively.

A *reward structure* for an MDP assigns values to states and transitions that are interpreted, for example, as resource usage. Formally, a rewards structure is a pair of functions $(r_s, r_a)$ such that $r_s : S \rightarrow \mathbb{R}_{\geq 0}$ is a state-reward function, and $r_a : S \times S \times Act \rightarrow \mathbb{R}_{\geq 0}$ is a transition-reward function. PCTL is extended to include rewards-augmented operators: instantaneous rewards $R_{\bowtie r}[I = t]$ and cumulative rewards $R_{\bowtie r}[C \leq t]$, for $r, t \in \mathbb{R}_{\geq 0}$.

Our PPM approach uses the probabilistic model checker PRISM [17] developed at the Oxford University Computing Laboratory to verify quantitative properties of models that encode probabilistic cloud deployment patterns, and which are synthesised from these patterns automatically.

## 3 Approach

Our approach to assessing the cost and resource usage characteristics of cloud deployments (Figure 1) comprises the following steps:

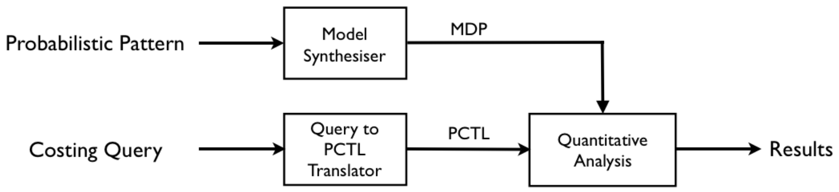

**Fig. 1.** Quantitative analysis of cloud deployments

1. Specification of the resource requirements for the cloud deployment as a high-level probabilistic pattern.
2. MDP Generation. Patterns from Step 1 are accepted as input to a *model synthesiser* that outputs an MDP model formalising probabilistic resource variations over a time interval associated, for example, with a particular day, week, or billing period.
3. PCTL generation. Queries relating to costs or other quantitative properties of the cloud deployment act as input to a *query-to-PCTL translator* that outputs the query formalised as a PCTL formula.
4. Quantitative Analysis. The MDP model generated in Step 2 and the PCTL formula from Step 3 act as input to a probabilistic model checker which performs quantitative analysis. The numerical results of the analysis are returned for further analysis or data visualisation.

### 3.1 A Language for Probabilistic Cloud Usage Patterns

PPM formalises the resource requirements of a cloud deployment as probabilistic patterns, expressed in terms of a simple language[1] based on the declaration of rules that specify elastic variations of cloud resource usage over time. Formally, a *probabilistic pattern*

$$P = BR^* \tag{2}$$

is a high-level syntactic representation of a customer's hourly usage of cloud computing resources, beginning with a baseline declaration $B$, and optionally followed by a finite list of $r$ rule declarations $R_1, \ldots, R_r$[2].

A baseline declaration $B$ has the syntactic form

$$\texttt{Baseline}\ bl \tag{3}$$

[1] We do not expect probabilistic patterns to be composed by hand but rather generated automatically through analysis of application resource usage and request logs.

[2] We list the syntactic form of each declaration in the language with terminals displayed in `fixed-width font`.

specifying that the user requires a constant amount $b$ of cloud computing resources (e.g., virtual machines).

A rule declaration $R$ has the syntactic form

$$\texttt{Rule Start } S \texttt{ Vary } V \tag{4}$$

which enables the specification of a variation $V$ from the resource usage at a certain point in time $S$. For example, a rule might specify an increase in resources to meet computing requirements at peak times when more resources are needed. Such variations are probabilistic in nature to reflect the increase and decrease of resources depending on factors such as workload, capital, economies, and even the fluctuations in the pricing of cloud services themselves.

A variation $V$ is a set of $m$ discrete probability distributions

$$\{D_1, \ldots, D_m\} \tag{5}$$

each of which has the syntactic form

$$\texttt{[}p_1\texttt{:}Op_1\texttt{(}z_1\texttt{) + } \ldots \texttt{ + } p_q\texttt{:}Op_q\texttt{(}z_q\texttt{)]} \tag{6}$$

to express the fact that change in resources involves

- a non-deterministic choice of a probability distribution $D_i \in V$, and
- a selection of a resource usage *variant* $Op\texttt{(}z\texttt{)}$ according to the probability distribution $D_i$, with probabilities $p_1, p_2, \ldots, p_q$, $\sum_{j=1}^{q} p_j = 1$.

A variant $Op\texttt{(}z\texttt{)}$ comprises a name $Op$ of an operation and a numerical operand $z$. Variants perform arithmetical operations on the resource amount $res$ at time $S$ to yield a new resource usage value $res'$ for time $S+1$. The types of variants that can be used in PPM patterns are given in Table 1, noting that the "baseline" variant `bl` does not require an operand.

**Table 1.** Pattern Variants

| **Variant** $Op(z)$ | **Description** | **Resources** $res'$ **at time** $S+1$ |
|---|---|---|
| `add(`$z$`)` | Apply arithmetical operation to current resource usage amount | $res + z$ |
| `sub(`$z$`)` | | $res - z$ |
| `mult(`$z$`)` | | $res \times z$ |
| `div(`$z$`)` | | $res \div z$ |
| `bl` | Set resource usage to baseline value | $bl$ |
| `bl-add(`$z$`)` | Set resource usage to baseline value and apply arithmetical operation. | $bl + z$ |
| `bl-sub(`$z$`)` | | $bl - z$ |
| `bl-mult(`$z$`)` | | $bl \times z$ |
| `bl-div(`$z$`)` | | $bl \div z$ |

*Example 1.* Consider a cloud deployment associated with a set of applications whose resource requirements follow a weekly (probabilistic) pattern. Assume, for instance, that less resources are required at the weekend than during the rest of the week as specified by the probabilistic pattern

```
Baseline 10
Rule1 Start Jun 2nd 0h  Vary {[0.6:sub(2) + 0.4:sub(5)],
                              [0.1:add(1) + 0.9:sub(3)]}
Rule2 Start Jun 3rd 23h Vary {[1:bl]}
```

that declares a baseline of 10 resource units followed by two rules. The first rule starts at the beginning of Saturday, June $2^{nd}$ (2012), varying resource usage according to the two stated probability distributions. Variants in these distributions subtract resources from the baseline amount, suggesting that this rule generally decreases resource usage; there is a small probability of selecting the variant `add(1)`. The second rule begins at the end of Sunday June $3^{rd}$, and consists of a single distribution setting the resource usage back to the baseline `bl`.

Of course, one might want to specify that less resources are required for *every* weekend during certain months in the year. To support this scenario, our language allows the specification of an optional repeat declaration for each rule (4):

$$\texttt{Rule Start } S \texttt{ Vary } V \texttt{ Repeat } F \texttt{ Until } U \tag{7}$$

where $F$ is a keyword from the set $\{\texttt{Day}, \texttt{WeekDay}, \texttt{Week}, \texttt{WeekEnd}, \texttt{Month}\}$ setting the frequency of the rule's application, and $U$ specifies the last time the rule is to be applied.

*Example 2.* Using the repeat construct (7), the probabilistic pattern in Example 1 can be extended to

```
Baseline 10
Rule1 Start Jun 2nd  0h Vary {[0.6:sub(2) + 0.4:sub(5)],
                              [0.1:add(1) + 0.9:sub(3)]}
                       Repeat WeekEnds Until Aug 31
Rule2 Start Jun 3rd 23h Vary {[1:bl]}
                       Repeat WeekEnds Until Aug 31
```

This specifies that both rules apply every weekend during the summer months of June, July and August.

## 3.2 Markov Decision Process Synthesis

The model synthesiser takes as input a probabilistic pattern (2) in the form of the concrete syntax described in Section 3.1 and a time interval

$$T = [0, n], \tag{8}$$

and outputs an MDP model that allows the formal analysis of cost- and resource-related characteristics of the pattern.

We say that an MDP $M = (S, s_0, Act, Dist, step, L)$ *models* $P$ over $T$ if it satisfies the following properties:

1. The state space is $S \subseteq \{(res, t) \mid res \geq 0,\ t \in T\}$, contains a state $s = (res, t)$ for each resource amount $res \geq 0$ that the cloud deployment may assume at time instant $t$, and no other states.
2. The initial state is $s_0 = (bl, 0)$ according to the value $bl$ specified in the baseline declaration $B$.
3. $Act$ comprises an action $a_{ij}$ for each rule $R_i$ from $P$, $1 \leq i \leq r$ and each distribution $D_{ij}$ from rule $R_i$.
4. $Dist$ represents the set of all distributions from rules $R_1, \ldots, R_r$ of $P$.
5. $step : S \rightarrow 2^{Act \times Dist}$ specifies state transitions of the form $(res, t) \xrightarrow{a_{ij}, p} (res', t+1)$ where $a_{ij} \in Act$ is the action corresponding to distribution $D_{ij}$ and $p$ the probability that $D_{ij}$ associates with state $(res', t+1)$.
6. $AP = \{\text{"}resources \geq x\text{"} \mid x \in \mathbb{R}_{\geq 0}\}$ and $L(res, t) = \{\text{"}resources \geq x\text{"} \mid x \geq res\}$.

The remainder of the section describes the algorithm that PPM uses to determine $S$ and *step*. First, we model the progression of time over (8) by the function $tick : S \rightarrow S$ defined by

$$tick((res, t)) = \begin{cases} (res, t+1) & \text{if } 0 \leq t < n, \\ (res, t) & \text{otherwise.} \end{cases} \tag{9}$$

We say that a rule $R_i$ in $P$ *applies* at time $t$ if $R_i$ is

– a rule declaration of the form (4) with start time $t$, or
– a repeat rule declaration of the form (7) with start time $t$ or frequency $F$ such that $t \bmod F = 0$,

with a possible change of resource usage occurring at time $t+1$. We model rule applications by the function $apply : Variant \times S \rightarrow S$ defined by

$$apply(Op(z), (res, t)) = tick((res', t)) \tag{10}$$

where $res'$ is the new resource usage value at time $t+1$ defined according to Table 1.

Letting $S^t$ denote the subset of all states in $S$ associated with time $t$, the state space $S$ of an MDP modelling a pattern over (8) is defined by the equations

$$S^0 = \{s_0\},$$

$$S^{t+1} = \begin{cases} \bigcup_{D_{ij} \in V} \bigcup_{p : Op(z) \in D_{ij}} & \\ \quad \{apply(s, Op(z)) \mid s \in S^t\} & \text{if a rule applies at time } t, \\ \{tick(s) \mid s \in S^t\} & \text{otherwise,} \end{cases} \tag{11}$$

where $S = S^0 \cup S^1 \cup \cdots \cup S^n$ and $S^t \cap S^{t'} = \emptyset$, for $t \neq t'$. Equation 11 defines the states in $S^{t+1}$ by applying variants $Op(z)$ to the states in $S^t$ whenever a rule applies at time $t$. If no rule applies then resources remain unchanged and only $t$ is updated by the *tick* function.

The transition function *step* is defined by the equation
For all $s \in S^t$:

$$step(s) = \begin{cases} \bigcup_{D_{ij} \in V} \bigcup_{p:Op(z) \in D_{ij}} & \\ \quad \{(s \overset{a_{ij},p}{\rightarrow} apply(s, Op(z)))\} & \text{if a rule applies at time } t \\ \{s \overset{a,1}{\rightarrow} tick(s)\} & \text{otherwise,} \end{cases}$$

where $step(s)$ contains state transitions of the form $s \overset{a_{ij},p}{\rightarrow} apply(s, Op(z))$ whenever a rule applies at time $t$. If no rule applies then a unique action $a \in Act$ is chosen and $step(s)$ contains state transitions of the form $s \overset{a,1}{\rightarrow} tick(s)$.

## 3.3 Quantitative Analysis of Cloud Deployment Queries

Using the query-to-PCTL translator, high-level queries relating to cost and other quantitative properties are formalised as rewards-augmented PCTL formulae. In this section, we present a list of such queries and their specification in PCTL and we outline the verification of each property on an MDP pattern model $M$ over time interval $T$ from (8).

1. *What is the maximum probability of the cloud deployment's resource requirements equalling or exceeding the amount $x$?*

This resource usage query is specified by the PCTL formula $Pmax_{=?}[F\ res \geq x]$. Quantitative verification returns the maximum probability of eventually reaching a state in $M$ satisfying the property $res \geq x$. Queries for minimum probabilities or those with probability bounds are also easily specified.

To perform cost analysis we augment $M$ with a rewards structure $r_s : S \rightarrow \mathbb{R}_{\geq 0}$ defined by $r_s((res, t)) = res$, associating every state $(res, t)$ with the value $res$. We interpret rewards as the *cost* of the deployment at time $t$. By using actual resource amounts the customer can scale values according to unit resource prices set by the provider. For example, the cost at time $t$ of using Amazon's Standard On-Demand large instances is calculated $res \times 0.34$¢.[3]

2. *What is the expected maximum cost of a cloud deployment's resource requirements at any point $t$?*

Using MDPs with costs, this high-level cost analysis query is specified by the rewards-augmented PCTL formula $Rmax_{=?}[I = t]$, where $I = t$ denotes the instantaneous cost at time $t$.

3. *What is the expected maximum cumulative cost of a cloud deployment's resource requirements up to time $t$?*

This high-level cost analysis query is specified by the rewards-augmented PCTL formula $Rmax_{=?}[C \leq t]$, where $C \leq t$ denotes the cumulative reward up to time

[3] aws.amazon.com/ec2/pricing (Checked January 2012).

$t$. Quantitative verification returns the expected maximum cost of $M$ accumulated over the time interval $[0, t] \subseteq T$.

## 4 Implementation and Validation

We developed a probabilistic pattern modelling tool `PPM` that implements our approach for the analysis of probabilistic cloud deployment patterns. `PPM` is an open-source Java class library[4] that supports the realisation of the workflow in Figure 1. The core component of `PPM` is a `PatternProcessor` class whose constructor takes as parameters the probabilistic pattern (2) to analyse, and the upper bound for (8). The `PatternProcessor` constructor implements the MDP synthesis technique described in Section 3.2 by means of a parser-generator built using the off-the-shelf language tool ANTLR[5]. The result of this model synthesis is an MDP expressed in the PRISM state-based language.

**Table 2.** Analysis methods provided by the `PatternProcessor` `PPM` class

| | |
|---|---|
| Java | `double getMaxProbResourcesExceeds(int x)` <br> `double getMinProbResourcesExceeds(int x)` |
| Input | $x \geq 0$ |
| Output | max(min) probability of resource usage exceeding $x$ at any time |
| PCTL | `Pmax=?[F res >= x]`, `Pmin=?[F res >= x]` |
| Java | `Double[] getMaxResources(int t1, int t2, int step)` <br> `Double[] getMinResources(int t1, int t2, int step)` |
| Input | $t_1, t_2, step > 0$ such that $t_1 < t_2$ |
| Output | List of expected max(min) resource usage over $[t_1, t_2]$ performed for each $t = t_1 + i \cdot step$, $i = 0, 1, 2, \ldots$ such that $t_1 \leq t \leq t_2$. |
| PCTL | `Rmax=?[I=t]`, `Rmin=?[I=t]` |
| Java | `Double[] getMinCumulativeResources(int t1, int t2, int step)` <br> `Double[] getMaxCumulativeResources(int t1, int t2, int step)` |
| Input | $t_1, t_2, step > 0$ such that $t_1 < t_2$ |
| Output | Expected max(min) cumulative resource usage over $[t1, t2]$ performed for each $t = t_1 + i \cdot step$, $i = 0, 1, 2, \ldots$ such that $t_1 \leq t \leq t_2$. |
| PCTL | `R=?[C<=t]` |

The public methods of the `PatternProcessor` class (Table 2) enable the quantitative analysis of a range of cost and resource usage properties of the considered probabilistic pattern. Each such method synthesises the appropriate PCTL property as described in Section 3.3, and runs the probabilistic model checker PRISM in the background to analyse this PCTL property against the MDP model generated by the constructor. The result of the PRISM analysis is parsed and returned to the client that invoked the method.

[4] `PPM` is freely available from `http://www1.aston.ac.uk/eas/staff/dr-kenneth-johnson/ppm/`

[5] `http://www.antlr.org/`

To assess the effectiveness and scalability of PPM, we implemented a simple test tool that was used to carry out the case study and scalability experiments presented in the remainder of this section. The results of this validation exercise are being used to improve PPM, with a view to integrate it into an existing high-level tool for cloud adoption decision [13].

## 4.1 Case Study

The case study described in this section considers a potential cloud customer whose applications require at least three virtual machines at all times in order to maintain an acceptable system response time. This resource requirement can be formalised as a probabilistic pattern with a single baseline declaration:

```
Baseline 3.
```

Each weekday two or three more VMs above the baseline usage are required to be started at 7am. At 9am, four or five virtual machines above the baseline usage are required. These requirements are modelled by two rules:

```
Rule1 Start Jan,1,7 Vary {[0.8:bl-add(2) + 0.2:bl-add(3)]}
                    Repeat WeekDay Until Dec,31
Rule2 Start Jan,1,9 Vary {[0.7:bl-add(4) + 0.3:bl-add(5)]}
                    Repeat WeekDay Until Dec,31.
```

Resource usage is reduced at 5pm and again at 7pm where resources are set back to baseline. These requirements are modelled by two further rules:

```
Rule3 Start Jan,1,17 Vary {[0.8:bl-add(2) + 0.2:bl-add(3)]}
                     Repeat WeekDay Until Dec,31
Rule4 Start Jan,1,19 Vary {[1:bl]}
                     Repeat WeekDay Until Dec,31.
```

We used PPM to analyse the probabilistic cloud deployment pattern described above. The PRISM MDP model generated as a result of building the PPM Pattern Processor object for this pattern is depicted in an abbreviated form in Figure 2.

We display in Figure 3 the results of analysing the pattern's maximum expected resource usage over the time interval ($72 \leq t \leq 96$) representing the weekday of January $3^{rd}$. The figure labels each application of a rule by the rule name. As the pattern indicates, the number of VMs is 3 at the beginning of the day and peaks between 9am and 5pm (i.e., for $79 \leq t \leq 90$), when the expected maximum resource usage has value 7.3. The resource usage returns to the baseline outside working hours at 7pm (when $t \geq 91$). Figure 4 depicts the results of cost analysis performed to determine monthly maximum expected accumulated costs from January to April 2012. The four values on the graph are labeled with the cloud resource usage at the end of each month. These values can be used to determine the maximum expenditures for cloud computing services expected at the end of a provider's billing period. For example, supposing the customer uses Amazon's standard EC2 instance at a unit cost of 0.32¢ throughout January, the maximum expected expenditure at the end of January is 3024×0.32¢ = \$967.68.

```
mdp
...
const BASELINE=3;
formula Rule1  = (m=Jan)&(d=3)&(h=6);
const double p1=0.8; const double p2=0.2;
const Amount1 = 2; const Amount2 = 3;
formula Vary1 = min(BASELINE+Amount1,MAXRES);
formula Vary2 = min(BASELINE+Amount2,MAXRES);
...
formula RuleStart = (Rule1| ... );

rewards true : res; endrewards
module pattern
   m : [0..11] init Jan; d : [1..31] init 1; h : [0..23] init 0;
   res: [0..Max] init BASELINE;

[] (!DayEnd)&(!RuleStart) -> (h'=h+1);
[] (!DayEnd)&(Rule1) -> p1:(res'=Vary1)&(h'=h+1)+p2:(res'=Vary2)&(h'=h+1);
...
[] (DayEnd)&(NotMonthEnd)&(!RuleStart) -> (h'=0)&(d'=d+1);
[] (DayEnd)&(NotMonthEnd)&(Rule1) -> p1:(res'=Vary1)&(h'=0)&(d'=d+1)+
                                     p2:(res'=Vary2)&(h'=0)&(d'=d+1);
...
[] (DayEnd)&(MonthEnd)&(m<Dec)&(!RuleStart) -> (m'=m+1)&(h'=0)&(d'=1);
[] (DayEnd)&(MonthEnd)&(m<Dec)&(Rule1)  ->
     p1:(res'=Variation1)&(m'=m+1)&(h'=0)&(d'=1) +
     p2:(res'=Variation2)&(m'=m+1)&(h'=0)&(d'=1);
...
[] (m=Dec)&(DecEnd)&(DayEnd)-> true;
endmodule
```

**Fig. 2.** Abbreviated PRISM file generated by analysis of the case study

## 4.2 Scalability

Experiments were performed on the PPM tool to test the scalability of our approach using a set of different patterns of increasing complexity defined over a single week beginning January $1^{st}$. Between one and five rules formed with the Repeat construct with WeekDay frequency increased resource amounts on each day in the week. Rules started from 12am staggered by four hours, and one or two probability distributions were specified in each rule.

The experiments recorded the number of states and transitions for each MDP model synthesised from the patterns. The analysis speed of PPM was tested by performing two method invocations from the Java library on each pattern:

- getMaxResources(0,168,1), calculating the maximum expected resource usage over a week with a step of one hour, and
- getMaxCumulativeResources(0,168,1), calculating the maximum expected cumulative cost over a week with a step of one hour.

Our experiments were performed using the PPM command-line interface on an Apple MacBook Pro operating on Mac OS X Version 10.7.2 with a 2.66Ghz Intel Core 2 Duo processor and 8GB of 1067MHz DDR3 memory. PRISM version 4.0.2 was used to perform the quantitative analysis on the synthesised MDPs using the sparse matrix PRISM engine [16]. The results of the experiments are listed in Table 3, where method invocation times are averages over several runs,

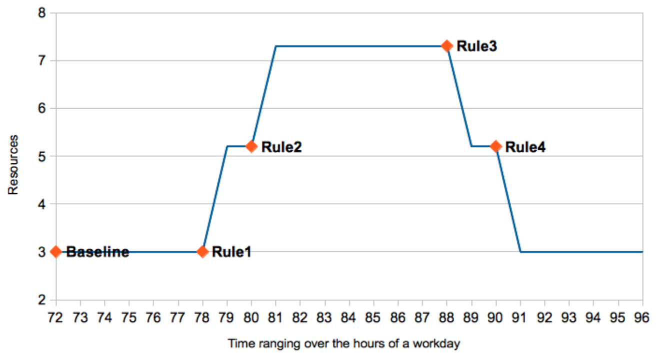

**Fig. 3.** Expected maximum resource usage for January 3$^{rd}$ (hours 72 to 96)

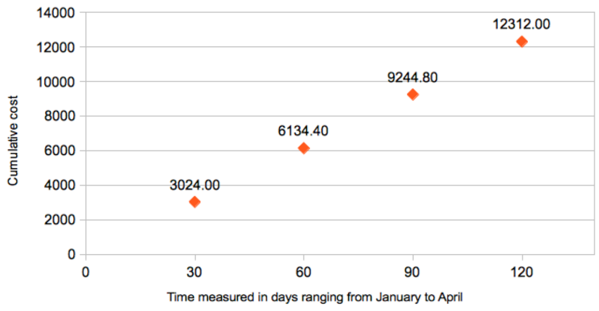

**Fig. 4.** End of month expected accumulated costs from January to April

and include all steps in processing the pattern: parsing, model synthesis and verification.

To analyse the size complexity of the state space $S$ of an MDP modelling a pattern, we introduce the following notation. The *size* of a rule $R$ with $m$ probability distributions each with $q_i$ probabilities, $1 \le i \le m$ is defined as $size(R) = q_1 + \cdots + q_m$. We assume that the maximum number of rule applications at time $t$ is bounded by the value $K$. For a pattern with rules $R_1, \ldots, R_r$ we set this value to satisfy the inequality $size(R_1) + \cdots + size(R_r) \le K$. We adapt Equation 11 to give the inequality

$$|S^{t+1}| \le \begin{cases} |S^t| \cdot K & \text{if a rule applies at time } t, \\ |S^t| & \text{otherwise,} \end{cases} \tag{12}$$

for all $t \ge 1$ and $|S^0| = 1$. Using Inequality 12, and noting the inequality $|S| \le |S^0| + |S^1| + |S^2| \cdots + |S^n|$ we calculate a bound on the size $|S|$ of the state space over the interval $[0, n]$. For patterns consisting of only a baseline declaration, we have $n+1 \le |S|$, e.g. a single state models each point in the time interval. Using the `Repeat` construct, rules can be applied repeated on a frequency $F$ yielding an upper bound $|S| \le \sum_{t=0}^{n} K^{\lceil \frac{t}{F} \rceil}$.

**Table 3.** Model size and method invocation time according to pattern complexity

| Pattern size | | Model size | | Avg. analysis time (minutes) | |
|---|---|---|---|---|---|
| Rules | Dists | States | Transitions | Max. Usage | Max. Cost |
| 1 | 1 | 654 | 669 | 3.67 | 2.81 |
| 2 | 1 | 1119 | 1174 | 3 | 6.22 |
| 3 | 1 | 1427 | 1532 | 7.41 | 6.23 |
| 4 | 1 | 1610 | 1763 | 13.47 | 4.94 |
| 5 | 1 | 1700 | 1890 | 10.54 | 4.21 |
| 1 | 2 | 1624 | 1729 | 7.96 | 7.85 |
| 2 | 2 | 3019 | 3454 | 15.58 | 16.22 |
| 3 | 2 | 3943 | 4804 | 18.29 | 19.47 |
| 4 | 2 | 4492 | 5767 | 26.81 | 27.11 |
| 5 | 2 | 4762 | 6358 | 29.87 | 25.58 |

## 5 Related Work

Several research projects focused on developing tools and techniques to assist organisations in assessing the cost-savings of transitioning to the cloud. Specific aspects of cloud technology such as Infrastructure as a Service (IaaS) have been formally modelled to analyse cost-savings of leasing virtualised hardware from remote data centres [21], while case studies assessing feasibility of cloud computing has been carried out for specific industries [14] and applications [7].

Advanced tools such as CloudSim [2] model components of cloud computing data centres for fine-tuning applications deployed on the cloud. CloudSim enables users to improve application performance by simulating resource provisioning policies, and work is in progress to extend support to include simulated costing-analysis of deployment on public clouds [3].

Research undertaken as part of the Large Scale Complex IT Systems (LSC-ITS)[6] initiative in the United Kingdom has developed the Cloud Adoption Toolkit [13] which is an organisational framework identifying key concerns of cloud services adoption, and comprising tools that support the decision making process of potential cloud customers. In particular, the framework's cost modelling tool allows cost-analysis of cloud deployments to be performed with resource requirements expressed in a notation similar to the language developed in our approach, but does not support the specification of probabilistic and non-deterministic characteristics.

Our work complements and improves upon these approaches by accounting for probabilistic behaviour of cloud deployments, and using precise techniques for cost and resource usage analysis.

[6] `http://lscits.cs.bris.ac.uk/`

## 6 Conclusion and Future Work

The results presented in this paper target the growing need for precise cost analysis techniques that address both uncertainty and probability in using cloud computing services. The probabilistic pattern modelling approach introduced in the paper formalises cloud computing resources as probabilistic patterns and synthesises Markov decision processes. Quantitative verification performed on the model providing accurate costing and usage results. The approach has been implemented as an open-source Java library using the probabilistic model checker PRISM. We have validated our approach with a case study, and carried out a number of preliminary scalability experiments.

Our future work aims at improving the PPM approach and tool in a number of ways. The scalability performance of the tool can be improved by exploiting the periodical nature of some patterns to eliminate redundant calculations when performing analysis. We plan to extend the PPM workflow to include software components to synthesises probabilistic cloud deployment patterns using data mining techniques on application resource request logs. Lastly, we plan to integrate the `PPM` tool with existing toolkits such as [13].

**Acknowledgements.** This work was partly supported by the UK Engineering and Physical Sciences Research Council grant EP/H042644/1.

## References

1. Armbrust, M., Fox, A., Griffith, R., Joseph, A.D., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I., Zaharia, M.: A view of cloud computing. Commun. ACM 53, 50–58 (2010)
2. Buyya, R., Ranjan, R., Calheiros, R.: Modeling and simulation of scalable cloud computing environments and the CloudSim toolkit: Challenges and opportunities. In: Intl. Conf. on High Performance Computing Simulation, pp. 1–11 (2009)
3. Calheiros, R.N., Ranjan, R., Beloglazov, A., De Rose, C.A.F., Buyya, R.: CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Software: Practice and Experience 41(1), 23–50 (2011)
4. Calinescu, R., Grunske, L., Kwiatkowska, M., Mirandola, R., Tamburrelli, G.: Dynamic QoS management and optimization in service-based systems. IEEE Transactions on Software Engineering 37, 387–409 (2011)
5. Calinescu, R., Johnson, K., Rafiq, Y.: Using observation ageing to improve Markovian model learning in QoS engineering. In: Proc. of the 2nd Joint Intl. Conf. on Performance Engineering, pp. 505–510. ACM (2011)
6. Calinescu, R., Kikuchi, S.: Formal Methods @ Runtime. In: Calinescu, R., Jackson, E. (eds.) Monterey Workshop 2010. LNCS, vol. 6662, pp. 122–135. Springer, Heidelberg (2011)
7. Deelman, E., Singh, G., Livny, M., Berriman, B., Good, J.: The cost of doing science on the cloud: the Montage example. In: Proc. of the 2008 ACM/IEEE Conf. on Supercomputing, pp. 50:1–50:12 (2008)

8. Hansson, H., Jonsson, B.: A logic for reasoning about time and reliability. Formal Aspects of Computing 6, 512–535 (1994)
9. Jeannet, B., D'Argenio, P., Larsen, K.: Rapture: a tool for verifying Markov decision processes. In: Cerna, I. (ed.) Proc. Tools Day, Affiliated to 13th Int. Conf. Concurrency Theory (CONCUR 2002). Technical Report FIMU-RS-2002-05, Faculty of Informatics, Masaryk University, pp. 84–98 (2002)
10. Jensen, M., Schwenk, J., Gruschka, N., Iacono, L.: On technical security issues in cloud computing. In: IEEE Intl. Conf. on Cloud Computing, pp. 109–116. IEEE Computer Society (2009)
11. Joint, A., Baker, E., Eccles, E.: Hey, you, get off of that cloud? Computer Law & Security Review 25(3), 270–274 (2009)
12. Katoen, J.P., Zapreev, I.S., Hahn, E.M., Hermanns, H., Jansen, D.N.: The ins and outs of the probabilistic model checker MRMC. In: Proc. of the 6th Intl. Conf. on the Quantitative Evaluation of Systems, QEST 2009, pp. 167–176. IEEE Computer Society Press, Los Alamitos (2009)
13. Khajeh-Hosseini, A., Greenwood, D., Smith, J.W., Sommerville, I.: The cloud adoption toolkit: supporting cloud adoption decisions in the enterprise. Software: Practice and Experience (2011)
14. Khajeh-Hosseini, A., Greenwood, D., Sommerville, I.: Cloud migration: A case study of migrating an enterprise IT system to IaaS. In: IEEE 3rd Intl. Conf. on Cloud Computing, pp. 450–457. IEEE Computer Society (2010)
15. Kikuchi, S., Matsumoto, Y.: Performance modeling of concurrent live migration operations in cloud computing systems using PRISM probabilistic model checker. In: Proc. 4th Intl. Conf. on Cloud Computing (2011)
16. Kwiatkowska, M., Norman, G., Parker, D.: PRISM: Probabilistic Symbolic Model Checker. In: Field, T., Harrison, P., Bradley, J., Harder, U. (eds.) TOOLS 2002. LNCS, vol. 2324, pp. 200–204. Springer, Heidelberg (2002)
17. Kwiatkowska, M., Norman, G., Parker, D.: PRISM 4.0: Verification of Probabilistic Real-Time Systems. In: Gopalakrishnan, G., Qadeer, S. (eds.) CAV 2011. LNCS, vol. 6806, pp. 585–591. Springer, Heidelberg (2011)
18. Kwiatkowska, M.: Quantitative verification: models techniques and tools. In: Proceedings of the the 6th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering, pp. 449–458 (2007)
19. Rutten, J., Kwiatkowska, M., Norman, G., Parker, D.: Mathematical Techniques for Analyzing Concurrent and Probabilistic Systems. In: Panangaden, P., van Breugel, F. (eds.) CRM Monograph Series, vol. 23. AMS (2004)
20. Vaquero, L.M., Rodero-Merino, L., Caceres, J., Lindner, M.: A break in the clouds: towards a cloud definition. SIGCOMM Comput. Commun. Rev. 39, 50–55 (2008)
21. Walker, E., Brisken, W., Romney, J.: To lease or not to lease from storage clouds. Computer 43(4), 44–50 (2010)
22. Youseff, L., Butrico, M., Da Silva, D.: Toward a unified ontology of cloud computing. In: Grid Computing Environments Workshop (GCE 2008), pp. 1–10 (2008)