

Assessing the Reliability and Cost of Web and Grid Orchestrations *

Alan Stewart, Maurice Clint, Terry Harmer
Peter Kilpatrick, Ron Perrott
School of Computer Science
The Queen's University of Belfast
Belfast BT7 1NN, Northern Ireland
Email: a.stewart@qub.ac.uk

Joaquim Gabarró
Universitat Politècnica de Catalunya
ALBCOM Research Group
Edifici Ω , Campus Nord Jordi Girona, 1-3
Barcelona 08034, Spain
Email: gabarro@lsi.upc.edu

Abstract

Unreliability is a characteristic feature of web and grid based computation: a call to a web or grid site may or may not succeed. An orchestration manager aims to control web or grid behaviour by constructing dynamic threads which acquire and utilise appropriate resources. For example, in an orchestration, a site may be called and may fail to respond after a period of time. Subsequently, the site may be recalled, or an alternative site may be utilised. In this paper approaches to estimating the reliability and cost of complex orchestrations are proposed. It is assumed that information about site reliability can be accessed from empirically acquired information.

1 Introduction

An orchestration is an evaluation that calls multiple service sites and coordinates the results produced by these calls into a composite computation. Typically orchestrations are dynamic and may choose which service sites to activate on the basis of cost or reliability. Suppose that each service site is characterised by a number of parameters which denote the *current* reliability, cost etc. of utilising underlying resources. Grid management involves assessing the effectiveness of different orchestrations based on these parameters, so that the best strategy for utilising resources may be chosen. Consider, for example, a grid manager wishing to acquire a service which is available on two sites s_1 and s_2 . The manager might construct an orchestration which simply calls s_1 . Alternatively, an orchestration might be constructed comprising two parallel threads: one thread calls s_1 and the other s_2 . In the latter case the orchestration will deliver the first result returned and terminate the other thread.

*A preliminary version of this paper was presented at the CoreGRID Integration Workshop, Krakow, 2006.

The latter orchestration is more robust than the former but it may have a higher associated cost. It is desirable to have a means of reasoning about (i) the likelihood of successful termination of an orchestration and (ii) its associated cost. In this paper a framework for reasoning about the reliability and cost of orchestrations is proposed.

Service calls are basic operations from which orchestration expressions are built. Suppose that $\Pr(S_s)$, the probability that site s responds, is available from empirical observation; similarly, suppose that $\Pr(F_s)$ is the probability that s does not respond. These probabilities can be used to estimate the likelihood that an orchestration will behave in a particular way. Conditional probabilities can be used to reason about multiple successful (or unsuccessful) calls to the same site. Let $\Pr(S_s | F_s)$ denote the probability that a call to s succeeds given that the last call to s failed. Let $\Pr(F_s | S_s)$ denote the probability that a call to s fails given that the previous call to s succeeded. Then $\Pr(F_s | F_s) = 1 - \Pr(S_s | F_s)$; $\Pr(S_s | S_s) = 1 - \Pr(F_s | S_s)$. By way of motivation, consider the following scenarios.

Example 1

GeneGrid [8] is a service-based grid platform designed for biologists to access data. **GeneGrid** contains a manager which ensures that required grid sites are (currently) working before it launches an orchestration. However, a site may fail between the initial readiness check and the orchestration launch. The reliability of an individual service call, s , within the overall orchestration can be assessed using the reliability measure $\Pr(S_s | S_s)$ rather than $\Pr(S_s)$. \square

Example 2

Techniques for broadcasters to share distributed media files have been developed in the **GridCast** project [2]. Data transfer has very high bandwidth requirements and involves a mix of *live* and stored events. Typically a broadcaster will have a network with limited bandwidth. Figure 1 shows how the number of live video feeds may vary with time.

Suppose that surplus bandwidth is made available to a

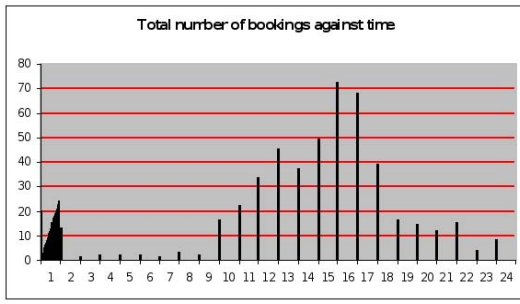


Figure 1. Number of live video feeds per hour

second broadcaster, with the first broadcaster having absolute priority with regard to use of the network. As a consequence, the bandwidth available to the second broadcaster varies during the day. At times of peak usage the second broadcaster may have to switch to an alternative higher cost network. Empirical data such as that in Figure 1 may be used to estimate the likely overall cost incurred by the second broadcaster. \square

In this paper it is demonstrated how site probabilities may be used to reason about the reliability and cost of orchestration executions. First, a brief overview of Orc, the language used here to model orchestrations, is presented.

2 Orc: a language for site orchestration

The orchestration language Orc [3] allows complex dynamic interactions among a collection of sites to be specified. The language provides a means for a user to interact with a web or grid environment and to receive responses from it [6]. A brief summary of the language Orc is given here – see [3, 7] for a complete description.

The simplest kind of Orc expression is a site call, possibly with parameters. A site call returns a single value. More complex orchestrations can be expressed using three combinators and recursion.

The operator $|$ denotes (independent) parallel composition. Evaluation of $s_1 | s_2$ generates calls to sites s_1 and s_2 . If both calls are successful the evaluation will produce two results. More generally, if E_1 and E_2 are Orc expressions (or threads) then the evaluation of $E_1 | E_2$ may produce multiple results.

Example 3

Evaluation of $s_1 | s_2$ is strongly successful only if both threads succeed: $\Pr(S_{s_1 | s_2}) = \Pr(S_{s_1}) \times \Pr(S_{s_2})$. The orchestration evaluation can only fail if *both* threads fail: $\Pr(F_{s_1 | s_2}) = \Pr(F_{s_1}) \times \Pr(F_{s_2})$.

The operator, $>$, denotes sequential composition. Evaluation of the expression $E_1 > x > E_2(x)$ first executes E_1 producing a result x (provided that the call is successful)

and then calls E_2 with parameter x . If the evaluation of E_1 generates n , $n > 1$, results then E_2 is evaluated n times, once for each result produced by evaluation of E_1 .

Example 4

Orc provides a special site *if* which returns a signal if its boolean argument is true and remains silent otherwise. The expression $(if(b) \gg s_1 | if(\neg b) \gg s_2)$ makes a choice between calling s_1 or s_2 , depending on the status of the boolean variable b . \square

The asymmetric parallel composition E_1 where $x : \in E_2$ where E_1 contains an x dependency, is evaluated by executing E_1 and E_2 in parallel until E_1 encounters a dependency; E_1 is suspended until a binding for x becomes available. A binding for x is found by nondeterministically selecting an output produced by E_2 . Evaluation of E_1 is then resumed and evaluation of E_2 is terminated.

Example 5

Consider evaluation of the expression

$$let(x) \text{ where } x : \in \{s_1 | s_2\}$$

where *let* is an internal site call which publishes its argument. The evaluation succeeds if either s_1 or s_2 succeeds. The effect of including redundancy in the orchestration is to increase the likelihood of success. \square

Example 6

Orc provides a timing operation via an internal site call, *Rtimer*(t) (relative timer), which returns a signal after t time units. *Rtimer* may be used to program timeouts. The expression $let(x) \text{ where } x : \in \{s | Rtimer(t)\}$ either returns the result produced by calling s , or, if such a result is not forthcoming within t time units, returns a timing signal (produced by *Rtimer*).

Timeouts and polling may be used to call site s repeatedly until it returns a result.

$$\begin{aligned} TimeOutPoll(s) &\triangleq \\ &(if(x = signal) \gg TimeOutPoll(s) \\ &| if(x \neq signal) \gg let(x)) \\ &\text{where } x : \in \{s | Rtimer(t)\} \quad \square \end{aligned}$$

3 Reliability of symmetric and sequential composition

In this section techniques for estimating the reliability of Orc expressions are developed. The possible outcomes of a site call s are given by the set $\Omega_s = \{F_s, S_s\}$. Event F_s indicates that the evaluation of the call fails to return a result; S_s indicates that it succeeds in returning a result.

Definition Failure/Success: Single external site call

Let $\Pr(F_s)$ be the probability that the external site call s fails and $\Pr(S_s)$ be the probability that s succeeds and returns a result. It is assumed that $0 < \Pr(S_s) < 1$ and $\Pr(F_s) + \Pr(S_s) = 1$. \square

Orc supports a set of special “internal” sites, including *Rtimer*, that are *guaranteed* to respond if invoked. A call to the special site 0 always behaves as a sink and does not respond. A call to *let* immediately publishes its argument. (*let* is often abbreviated to 1).

Definition Failure/Success: Internal site call

$$\Pr(S_{let}) = \Pr(S_{Rtimer(t)}) = 1 \text{ and } \Pr(S_0) = 0. \quad \square$$

The reliability of an expression is defined in terms of the probabilities of success of sites called during its evaluation. The following definitions provide a means to determine the set of site calls made during an orchestration evaluation.

Definition $\alpha(E)$

Let $\alpha(E)$ denote the alphabet of the expression E (that is the set of site calls that occur in E). \square

Definition $\natural E$

The maximum number of outputs published by E , $\natural E$, is defined over the structure of E as follows:

$$\begin{aligned} \natural 0 &= 0 \\ \natural s &= 1 \text{ if } s \text{ is a service site} \\ \natural i f(b) &= \text{if } b \text{ then } 1 \text{ else } 0 \\ \natural (E_1 | E_2) &= \natural E_1 + \natural E_2 \\ \natural (E_1 \gg E_2) &= \natural E_1 * \natural E_2 \\ \natural (E_1 \text{ where } z : \in E_2) &= \text{if } \natural E_1 \geq 1 \text{ then } \natural E_1 \text{ else } 0 \quad \square \end{aligned}$$

Definition $\natural(s, E)$

The number of times that an *external* site call, s , is made during an evaluation of E , $\natural(s, E)$, is defined over the structure of E as follows:

$$\begin{aligned} \natural(s, s) &= 1 \\ \natural(s, t) &= 0 \text{ if } s \neq t \\ \natural(s, E_1 | E_2) &= \natural(s, E_1) + \natural(s, E_2) \\ \natural(s, E_1 \gg E_2) &= \natural(s, E_1) + \natural(E_1) \cdot \natural(s, E_2) \end{aligned}$$

Note that $\natural(s, E)$ is different from the number of occurrences of s in E . For example, $\natural(r, (p|q) \gg r) = 2$ whereas there is only one textual occurrence of r in $(p|q) \gg r$. \square

The reliability of an orchestration may now be defined. Several variants are given.

Definition $R(E)$ and $R_{uc}(E)$

The reliability, R , of a non-recursive expression, E , involving only the operators symmetric composition and sequential composition is given by:

$$R(E) = \prod_{s \in \alpha(E)} \Pr(S_s) \cdot \Pr(S_s | S_s)^{\natural(s, E) - 1}$$

Here, conditional probability is used to distinguish repeated events from the first occurrence of the event. A corresponding definition of reliability based only on unconditional probability is:

$$R_{uc}(E) = \prod_{s \in \alpha(E)} \Pr(S_s)^{\natural(s, E)}$$

In Section 4 this definition is extended to embrace asymmetric parallel composition. \square

Example 7

Suppose that s_1 and s_2 are distinct sites. Then

$$\begin{aligned} R(s_1 | s_1) &= R(s_1 \gg s_1) = \Pr(S_{s_1}) \cdot \Pr(S_{s_1} | S_{s_1}) \\ R(s_1 | s_2) &= R(s_1 \gg s_2) = \Pr(S_{s_1}) \cdot \Pr(S_{s_2}) \quad \square \end{aligned}$$

Example 8

The operators $|$ and \gg satisfy the following laws [7].

$$\begin{aligned} f | g &= g | f, \quad (f | g) | h = f | (g | h) \\ 0 \gg f &= 0, \quad 1 \gg f = f, \quad f \gg 1 = f \\ (f \gg g) \gg h &= f \gg (g \gg h) \\ (f | g) \gg h &= (f \gg h) | (g \gg h) \end{aligned}$$

The definition of reliability is consistent with these laws.

For example:

$$\begin{aligned} R((f | g) \gg h) &= \Pr(S_f) \cdot \Pr(S_g) \cdot 2 \cdot \Pr(S_h) \\ &= R((f \gg h) | (g \gg h)) \end{aligned}$$

The definition of reliability proposed here is not consistent with the zero law, $f | 0 = f$. However, in §4 a more general notion of k -reliability is introduced. The definition of 1-reliability is consistent with the law $f | 0 = f$. \square

4 Reliability and asymmetric composition

To determine the reliability of E_1 where $x : \in E_2$ it is necessary to determine the probability that one or more threads in E_2 publishes. One way to do this is to model all possible behaviours that might arise during the evaluation of an expression. Let \mathcal{O}_E be a random variable denoting the number of outputs published by a given evaluation of E . \mathcal{O}_E can take values in the range $\{0, \dots, \natural E\}$.

Let $\Pr(\mathcal{O}_E = k)$, the k -reliability of E , denote the probability that evaluation of E publishes exactly k results. $\Pr(\mathcal{O}_E = k)$ can be used to determine the reliability of expressions involving asymmetric composition.

Definition k -reliability

k -reliability is defined over the structure of E . To simplify matters unconditional probability is used.

site call:

$$\begin{aligned} \Pr(\mathcal{O}_s = 0) &= \Pr(F_s), \quad \Pr(\mathcal{O}_s = 1) = \Pr(S_s) \\ \Pr(\mathcal{O}_s = k) &= 0, \quad k > 1 \end{aligned}$$

parallel composition:

$$\Pr(\mathcal{O}_{E_1 | E_2} = k) = \sum_{0 \leq j \leq k} \Pr(\mathcal{O}_{E_1} = j) * \Pr(\mathcal{O}_{E_2} = k - j)$$

Here, if E_1 publishes j results then E_2 must publish $k - j$ results. A special case of parallel composition is $|_i E$, the i -fold parallel composition of E where $i \geq 1$. $|_1 E = E$ and so $\Pr(\mathcal{O}_{|_1 E} = k) = \Pr(\mathcal{O}_E = k)$. Then

$$\Pr(\mathcal{O}_{|_{i+1} E} = k) = \sum_{0 \leq j \leq k} \Pr(\mathcal{O}_E = j) * \Pr(\mathcal{O}_{|_i E} = k - j)$$

sequential composition:

$$\begin{aligned} \Pr(\mathcal{O}_{E_1 \gg E_2} = k) &= \\ &= \sum_{0 \leq i \leq \natural E_1} \Pr(\mathcal{O}_{E_1} = i) * \Pr(\mathcal{O}_{|_i E_2} = k) \end{aligned}$$

asymmetric composition:

$$\Pr(\mathcal{O}_{E_1(z) \text{ where } z: \in E_2 = k}) = \Pr(\mathcal{O}_{E_1} = k) * (1 - \Pr(\mathcal{O}_{E_2} = 0))$$

The expression $(1 - \Pr(\mathcal{O}_E = 0))$ denotes the probability that at *least* one thread of E will publish a result. \square

Definition (Generalised) $R_{uc}(E)$

The unconditional reliability, R , of a non-recursive expression, E , involving the operators symmetric parallel composition, sequential composition and asymmetric parallel composition is given by:

$$R_{uc}(E) = \Pr(\mathcal{O}_E = \natural E)$$

$R_{uc}(E)$ is the probability that evaluation of E publishes the maximum possible number of results. \square

The probability of E failing is $\Pr(F_E) = \Pr(\mathcal{O}_E = 0)$. In particular:

$$\Pr(F_{E_1|E_2}) = \Pr(F_{E_1}) * \Pr(F_{E_2})$$

$$\Pr(F_{E_1 \gg E_2}) = \Pr(F_{E_1}) + \sum_{1 \leq j \leq \natural E_1} \Pr(\mathcal{O}_{E_1} = j) * \Pr(F_{jE_2})$$

$$\Pr(F_{E_1(z) \text{ where } z: \in E_2}) = \Pr(F_{E_2}) + (1 - \Pr(F_{E_2})) * \Pr(F_{E_1})$$

Determining the reliability of expressions using the definitions above is mechanical, but rather involved. Evaluation trees are used below to illustrate how the reliability of an orchestration is calculated. An *Evaluation Tree* (ET) for expression, E , models possible behaviours that may arise during evaluation of E . Each node has an associated value denoting the number of publications generated so far. Each path to a leaf denotes a sequence of site calls. The reliability of the expression $let(x) \text{ where } x: \in E$ is the sum of the probabilities of all paths in the associated evaluation tree of E which terminate in leaves with non-zero values.

Example 9

Consider the reliability of the expression

$$let(x) \text{ where } x: \in \{P | Q\}$$

Appealing to the definition gives:

$$R_{uc}(let(x) \text{ where } x: \in \{P | Q\}) = 1 - \Pr(F_P) * \Pr(F_Q).$$

Alternatively, the reliability may be determined using the evaluation tree of Figure 2(a). Assume, without loss of generality, that P is called before Q . The following paths have non-zero leaves: $\langle S_P, S_Q \rangle$, $\langle S_P, F_Q \rangle$, $\langle F_P, S_Q \rangle$. Thus:

$$R_{uc}(let(x) \text{ where } x: \in \{P | Q\}) = \Pr(S_P) * \Pr(S_Q) + \Pr(S_P) * \Pr(F_Q) + \Pr(F_P) * \Pr(S_Q)$$

Note that the evaluation tree based reliability may be viewed as $(1 - \text{the sum of the probabilities in all paths terminating in zero values})$, giving a result identical in form to that obtained from the definition of R_{uc} . \square

Example 10

The evaluation tree of the expression

$$let(x) \text{ where } x: \in \{(P | Q) \gg P\}$$

is shown in Figure 2(b). The reliability of the expression is

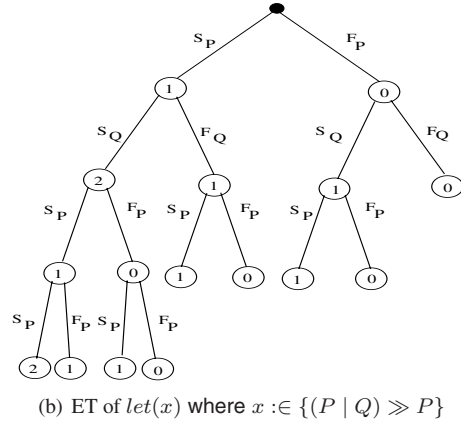
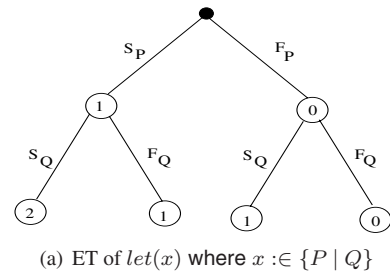


Figure 2. Examples of Evaluation Trees.

given by

$$R_{uc}(let(x) \text{ where } x: \in \{(P | Q) \gg P\}) = 1 - (\Pr(F_P)\Pr(F_Q) + \Pr(F_P)\Pr(S_Q)\Pr(F_P) + \Pr(S_P)\Pr(F_Q)\Pr(F_P) + \Pr(S_P)\Pr(S_Q)\Pr(F_P)\Pr(F_P))$$

Again, the expression corresponds to the sum of the probabilities of all paths in the evaluation tree that terminate in leaves with non-zero values.

5 Estimating the cost of orchestrations

The study of network economy is a well established field [5]. In this paper network managers, expressed using Orc, are directly mapped onto Markov chains [1] which are subsequently used to calculate the expected performance of the network services and to estimate prices. Two examples are considered.

Data Transmission Consider a television station with several regional sites. One of these sites, P , produces programmes. A programme produced by P is transmitted (as a stream of blocks $\underline{x} = (x_1, x_2, \dots, x_n)$) to another site, B , for broadcasting. For the programme to arrive in time for broadcast by B a minimum transmission rate R is required. A program block may be transmitted using one of two avail-

able channels:

- A *low cost channel, low, with variable bandwidth*. The service call $low(x_i)$ places block x_i on this channel for transmission to B . The call returns the estimated transmission rate. This channel has, on average, a transmission rate much higher than R but, during busy periods, this may fall to an unacceptably low rate. The cost of transmitting a block on this channel is L .
- A *high cost, high quality service, high*. The service call $high(x_i)$ places block x_i on this channel and guarantees a transmission rate exceeding R . The cost of transmitting a block on this channel is H .

The transmission from P to B starts using service *low*. However, whenever the transmission rate drops below R , the communication is switched to channel *high*. The transmission manager is defined as

$TransmissionManager(\underline{x}) \triangleq lowCost(\underline{x})$ where

$$\begin{aligned}
 lowCost([\] &\triangleq 0 \\
 lowCost(\underline{x}) &\triangleq low(head(\underline{x})) > r > \\
 & \quad (\text{if}(r \geq R) \gg lowCost(tail(\underline{x})) \\
 & \quad | \text{if}(r < R) \gg highCost(tail(\underline{x}))) \\
 highCost([\] &\triangleq 0 \\
 highCost(\underline{x}) &\triangleq high(head(\underline{x})) \gg highCost(tail(\underline{x}))
 \end{aligned}$$

The producer of the programme may be interested in the transmission cost for n blocks. Suppose that the first k blocks are transmitted through *low* and that the transmission rate for block x_k falls below R . Transmission is switched to service *high* for blocks x_{k+1}, \dots, x_n . The cost of the whole transmission is $Lk + (n-k)H$. To estimate the expected cost we need to consider the probability, $\Pr(S_{low})$, of *low* having a transmission rate in excess of R . Assume that:

$$\begin{aligned}
 \Pr(S_{low}) &= p, \quad \Pr(F_{low}) = 1 - p = q \\
 \Pr(S_{high}) &= 1, \quad \Pr(F_{high}) = 0.
 \end{aligned}$$

Now consider the probabilities associated with the transmission of a file (see example in Figure 3(a)). Note that blocks x_1, \dots, x_k are transmitted through the low cost channel and blocks x_{k+1}, \dots, x_n are transmitted through the high cost channel with probability $p^{k-1}q$ (x_1 is always transmitted via the low cost channel). The expected cost is given by:

$$\begin{aligned}
 Cost(n, L, H, p) &= \sum_{k=1}^{n-1} (kL + (n-k)H)p^{k-1}q \\
 &+ nLp^{n-1} \\
 &= nH - (1-p^n)(H-L)/q
 \end{aligned}$$

There are two limit cases:

$$\begin{aligned}
 p = 1: & \text{ since } \lim_{p \rightarrow 1} (1-p^n)/q = n, \text{ the cost is } nL. \\
 p = 0: & \text{ the cost is } L + (n-1)H.
 \end{aligned}$$

Brokering with bounded delay Consider a broker of-fering access to two supercomputer centres.

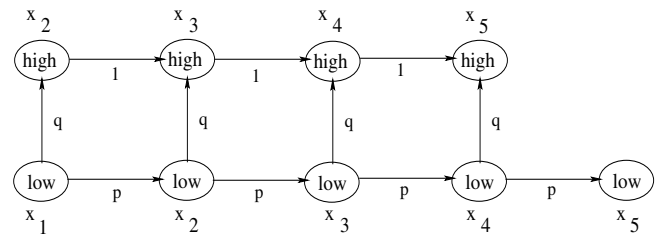
- A *low cost, unreliable supercomputer centre* which, in response to a submission, returns a pair $(done, result)$ – where *done* indicates whether or not the job has been executed and *result* stores the result if available – and returns *void* otherwise. The cost of executing a job on this site is L .
- A *high cost supercomputing centre* which can always execute jobs but at a high cost, H .

In the interests of offering a good service the broker tries to submit the job to the low cost centre several times. To provide bounded delay, the number of such attempts is bounded by a constant T . If the submission has not been successful after T attempts then the broker will submit the job to the high cost site. The broker can be expressed in Orc as

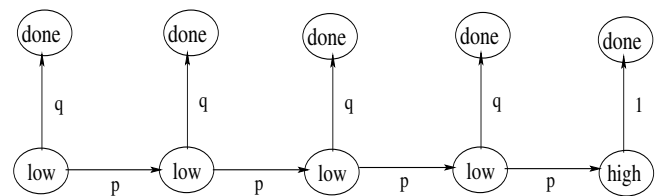
$Broker(\underline{x}) \triangleq lowCost(1, J)$ where

$$\begin{aligned}
 lowCost(i, J) &\triangleq low(J) > (done, result) > \\
 & \quad (\text{if}(done) \gg \text{let}(result) | \\
 & \quad \text{if}(\neg done \wedge i < T) \gg lowCost(i+1, J) | \\
 & \quad \text{if}(\neg done \wedge i = T) \gg highCost(J)) \\
 highCost(J) &\triangleq high(J) > result > \text{let}(result)
 \end{aligned}$$

If p is the probability of a failure for the low cost centre and the Broker replies in at most T steps, then the probability of this centre executing the job is $1 - p^T$; the probability that the high cost centre executes J is p^T . Thus the expected cost of a job determined by the Broker is $(1 - p^T)L + p^T H$. The expected service time is $(1 - p^T)/(1 - p)$.



(a) Behaviour of the data transmission manager, $n = 5$. The states, *low*, correspond to the calls to the *lowCost* service and states, *high*, correspond to *highCost*.



(b) Behaviour of the broker manager.

Figure 3. Markov chains associated with data transmission and broker managers.

6 Experimental evaluation of reliability and cost

The Orc language has an associated system that allows execution of Orc programs [4]. This system was used to validate some of the experimental results presented earlier by using the built-in random number generator to associate probabilities with site calls and then determine the reliability or cost of the related orchestrations.

Reliability

Consider again Example 10. With $\Pr(S_P) = 0.6$; $\Pr(S_Q) = 0.3$ the formula predicts

$$R_{uc}(let(x) \text{ where } x : \in (P \mid Q) \gg P) = 0.48.$$

In the following program **random**(n) returns a random integer in the range $0 \dots n - 1$. The behaviour of site P is defined as:

```
def P =
  random(10) > n >
  (if(1e(n, 5)) >> let(1) | if(gt(n, 5)) >> let(0))
```

P returns either a 1, indicating success, or a 0, indicating failure. Site Q is defined similarly. The behaviour of the entire expression is then defined by:

```
def R = let(r) where r in
  { {let(x, y) where x in P; y in Q}
    >>
    { if(and(eq(x, 1), eq(y, 1))) >> PP
      | if(and(eq(x, 1), eq(y, 0))) >> P
      | if(and(eq(x, 0), eq(y, 1))) >> P
      | if(and(eq(x, 0), eq(y, 0))) >> let(0)
    }
  }
```

The expression PP represents two parallel calls to P : it will succeed (and output 1) if either (or both) calls to P produce 1; otherwise it outputs 0 indicating failure:

```
def PP = { if(or(eq(x, 1), eq(y, 1))) >> let(1)
          | if(not(or(eq(x, 1), eq(y, 1)))) >> let(0) }
  where x in P; y in P
```

The experimental reliability value obtained by averaging 10,000 calls to the expression $let(x) \text{ where } x : \in \{(P \mid Q) \gg P\}$ was 0.47.

Cost

Orc programs were written to test experimentally the results obtained in §5.

The Data Transmission Broker:

Setting $n = 6$; $L = 5$; $H = 20$; $p = 0.3$; $q = 0.7$.

Expected Cost: Theory gives 98.59; Expt. gives 98.47.

The Network Manager:

Setting $p = 0.6$; $T = 5$; $L = 5$; $H = 20$.

Theory gives: Cost = 6.17; Time = 2.31.

Expt. gives: Cost = 6.17; Time = 2.29.

7 Conclusions

The Grid is an untrustworthy resource dispenser. A probabilistic approach has been developed to allow assessments of the reliability and cost of orchestrations to be made. Two small, but realistic, examples have been used to illustrate how the proposed probabilistic approach can be applied. There is much scope for further development of the work reported upon here. It is anticipated that components to be executed on Grids will incorporate local managers to provide a degree of autonomic capability. Among other things, these managers will have to decide upon the most reliable and cost-effective mixture of computing resources for the task for which they are responsible. The approach outlined here might provide a basis upon which autonomic component managers can be designed.

Acknowledgment

This research is carried out under the FP6 Network of Excellence CoreGRID funded by the European Commission (Contract IST-2002-004265). J. Gabarro is partially supported by FET pro-active Integrated Project 15964 (AE-OLUS) and by Spanish projects TIN2005-09198-C02-02 (ASCE) and MEC-TIN2005-25859-E.

References

- [1] A. Engel. *Wahrscheinlichkeitsrechnung und Statistik, Bd.2*. Klett Verlag, 1973.
- [2] T. J. Harmer. Gridcast - a next generation broadcast infrastructure? *Cluster Computing*, 10(3):277–285, 2007.
- [3] J. Misra and W. Cook. Computation orchestration: A basis for wide-area computing. *Software and Systems Modeling (SoSyM)*, 6(1):83–110, March 2007.
- [4] Orc Home Page, 2007. <https://sourceforge.net/projects/orc/>.
- [5] C. Shapiro and R. Varian. *Information rules, a strategic guide to network economy*. Harvard Business School Press, 1999.
- [6] A. Stewart, J. Gabarró, M. Clint, T. J. Harmer, P. Kilpatrick, and R. Perrott. Managing grid computations: An orc-based approach. In *ISPA'06*, pages 278–291, 2006.
- [7] G. M. T. Hoare and J. Misra. A tree semantics of an orchestration language. In *Proc. of the NATO Advanced Study Institute, Engineering Theories of Software Intensive Systems*, NATO ASI Series, Jan. 2004.
- [8] S. Wasnik, P. Donachy, T. J. Harmer, R. H. Perrott, P. V. Jithesh, M. McCurley, J. Johnston, M. Townsley, and S. McKee. Genegrid: From “virtual” bioinformatics laboratory to “smart” bioinformatics laboratory. In *CBMS*, pages 768–776, 2006.