BOUNDED SITE FAILURES: AN APPROACH TO UNRELIABLE GRID ENVIRONMENTS [*]

Joaquim Gabarró[†], Alina García[‡]
Universitat Politècnica de Catalunya
ALBCOM Research Group
Edifici Ω, Campus Nord Jordi Girona, 1-3, Barcelona 08034, Spain
gabarro@lsi.upc.edu

Maurice Clint, Peter Kilpatrick, Alan Stewart
School of Computer Science
The Queen's University of Belfast
Belfast BT7 1NN, Northern Ireland
m.clint@qub.ac.uk

Abstract The abstract behaviour of a grid application management system can be modelled as an Orc expression in which sites are called to perform sub-computations. An Orc expression specifies how a set of site calls are to be orchestrated so as to realise some overall desired computation. In this paper evaluations of Orc expressions in untrusted environments are analysed by means of game theory. The set of sites participating in an orchestration is partitioned into two distinct groups. Sites belonging to the first group are called *angels*: these may fail but when they do they try to minimize damage to the application. Sites belonging to the other group are called *daemons*: when a daemon fails it tries to maximise damage to the application. Neither angels nor daemons can fail excessively because the number of failures, in both cases, is bounded. When angels and daemons act simultaneously a competitive situation arises that can be represented by a so-called angel–daemon game. This game is used to model realistic situations lying between over-optimism and over-pessimism.

Keywords: Fault tolerance, Bounded site failures, Strategic games, Nash equilibria.

[*]This research is carried out under the FP6 Network of Excellence CoreGRID funded by the European Commission (Contract IST-2002-004265).
[†]Partially supported by FET pro-active Integrated Project 15964 (AEOLUS) and by Spanish projects TIN2005-09198-C02-02 (ASCE) and MEC-TIN2005-25859-E. This work has been done in part during a CoreGRID Exchange Program CR38UPC-CR21QUB & CR35UNIPI.
[‡]Supported by a FPI Spanish grant BES-2003-2361.

1. Introduction

A Grid application management system calls sites in order to perform subcomputations. Typically, it is over-optimistic to assume that all site calls made during execution of a grid application will work correctly. While, to a certain extent, such failure may be dealt with by employing time-outs and corrective action, such defensive programming may not always be easy and in some cases not possible. There may be times when the user accepts the possibility of failure, but would like to have an estimate of the likelihood of success. Such an analysis can be obtained by using Orc [4] to describe the orchestration of sites in a grid application [10] and by estimating, using probability theory, the expected number of results that will be published by an expression evaluation [9]; each site S is assumed to have a probability of failure and distinct sites are assumed to be independent. In practice, it may be difficult to provide a meaningful measure of site reliability and the assumption that distinct sites are independent may be too strong. In this paper an alternative approach based on game theory is used to analyse the behaviour of orchestrations over unreliable environments.

Grid sites are partitioned into two disjoint sets, angels $\mathcal{A}$ and daemons $\mathcal{D}$:

- Sites in $\mathcal{A}$ fail in such a way as to minimize damage to an application. This kind of failure is called angelic.
- Sites in $\mathcal{D}$ fail in such a way as to maximise damage to the application. This kind of failure is called daemonic.

It is assumed that the number of possible failures in the sets $\mathcal{A}$ and $\mathcal{D}$ are bounded. $\mathcal{A}$ and $\mathcal{D}$ can be viewed as players in a strategic game. If only angels are present then the problem is a maximization one; if only daemons act then we have a minimization problem. The interesting case lies between the extremes, when both angels and daemons act simultaneously and a competitive situation arises that can be represented by a so-called angel-daemon game. Here, finding a Nash equilibrium gives a solution that may be used to model realistic situations for unreliable grids, where the outcome is found somewhere between over-optimism and over-pessimism.

The study of systems under failures with Nash equilibria is not new. In [3] implementation problems involving unreliable players (who fail to act optimally) are studied. In [5] the authors study distributed systems in which players may exhibit Byzantine behaviour to undermine the correctness of the system. Note that *orchestrations represent control from one party's perspective* [7]. In this sense the analysis of orchestrations is different form the analysis of distributed systems under failures. The analysis of distributed systems is based (at least in part) on the graph properties of the underlying network. In the case of orchestrations we have to abstract the network (or consider it as another web or grid service). It is this "one party perspective" that makes the following analysis new.

In §2 a brief overview of Orc is presented. In §3 a way of assessing the benefits of evaluating an expression in an unreliable environment is proposed. In §4 a means of applying game theory to analyse the outcomes of executing orchestrations on unreliable networks is proposed. In §5 we assume that only one player controls the situation. When the player is $\mathcal{A}$ the damage is minimized. On the other hand, when the player is $\mathcal{D}$ the damage is maximized. These represent the two possible extreme coordinated behaviours, one extremely good and the other extremely bad. In this case there is no competitive activity and we have an optimization problem. In §6 we consider a competitive case defining a zero sum game, the so called angel-daemon game. In this game, both $\mathcal{A}$ and $\mathcal{D}$ play simultaneously. In §7 we apply the angel-daemon game to see how a grid manager assigns macro instructions to angelic and daemonic interpreters. In §8 we conclude and identify some open points.

2. Orc: a brief overview

A set of site calls can be orchestrated into a complex computation by means of an Orc expression [4]. A site call either returns a result or remains silent – silence corresponds to a site failure. The site which always fails (and is useless) is denoted 0. Site calls can be combined by means of three composition operations.

- Sequence: $P \gg Q$. For each output published by P an instance of Q is executed. The notation $P > x > Q(x)$ is used in situations where the computation Q depends on the output of P.
- Symmetric Parallelism: $P|Q$. The published output of $P|Q$ is *any* interleaving of the outputs of P and Q.
- Asymmetric parallelism: P where $x :\in Q$. Threads in P and Q are evaluated in parallel. Some of the threads of P may be blocked by a dependency on x. The first result published by Q is bound to x, the remainder of Q's evaluation is terminated and evaluation of the blocked threads of P is resumed.

3. Value of an orchestration under reliability failures

Web and Grid environments are unreliable. Sites evolve and a user has little (or no) control over the execution environment. Given a complex orchestration E it is *unrealistic to assume that there will be no site failures* when this orchestration is executed.

Reliability assumption. Sites are unreliable and can fail. When a site fails it remains silent and delivers no result at all. When a site does not fail it delivers the correct result. Any kind of byzantine behaviour is excluded. Any kind of behaviour delivering an "approximate" result is also excluded.

Even though some sites fail, orchestration may still produce useful partial results. For example, robust orchestrations may contain a degree of redundant computation so that evaluations may succeed even when a number of site failures occur. Given an orchestration E let $\alpha(E)$ be the set of sites that are called in E. Let $\mathcal{F} \subseteq \alpha(E)$ denote a set of sites that fail during an evaluation of E. The behaviour of the evaluation of E in this environment is given by replacing all occurrences of s, $s \in \mathcal{F}$, by 0. Let $\varphi_{\mathcal{F}}(E)$ denote this expression.

Value assumption. The evaluation of an orchestration has value even if some sites fail. For a particular failure set $\mathcal{F}$ the usefulness of the evaluation of $\varphi_{\mathcal{F}}(E)$ is measured by $v(\varphi_{\mathcal{F}}(E))$, the *value* or *benefit* of the orchestration $\varphi_{\mathcal{F}}(E)$. The range of v should be a non-negative $\mathbb{R}$. The value function v should have the following basic properties:

- $v(\varphi_{\alpha(E)}(E))$ must equal 0 when all sites fail in an evaluation of E,
- $v(\varphi_{\mathcal{F}}(E)) \geq 0$ for all $\mathcal{F} \subseteq \alpha(E)$,
- if $\mathcal{F} \subseteq \mathcal{F}' \subseteq \alpha(E)$ then $v(\varphi_{\mathcal{F}}(E)) \geq v(\varphi_{\mathcal{F}'}(E))$.

In this paper, we measure the benefit by the number of outputs that E publishes,

$$v(E) = \text{numbers of outputs published by } E.$$

An algorithmic definition of $v(E)$, for non-recursive E, is:

$$\begin{array}{l} v(0) = 0 \ , \ \ v(s) = 1 \text{ if } s \text{ is a service site} \ , \ \ v(if(b)) = \text{ if } b \text{ then } 1 \text{ else } 0 \\ v(E_1|E_2) = v(E_1) + v(E_2) \ , \ \ v(E_1 \gg E_2) = v(E_1) * v(E_2) \\ v(E_1 \text{ where } z :\in E_2) = \text{ if } v(E_1) \geq 1 \text{ then } v(E_1) \text{ else } 0 \end{array}$$

$v(E)$ has polynomial time complexity with respect to the length of the expression E.

EXAMPLE 1 *Consider the following expression*

$$E = (M_1|M_2) > x > [(M_3|M_4) > y > (M_5(x) > z > M_6(z) \mid (M_7|M_8) \gg M_9(y))]$$

Then $v(E) = 12$. *If site* M_1 *fails the benefit is* $v(E') = 6$ *where*

$$E' = (0\ |M_2) > x > [(M_3|M_4) > y > (M_5(x) > z > M_6(z) \mid (M_7|M_8) \gg M_9(y))]$$

4. Assessing Orchestrations

In this section a method of partitioning a set of sites into angels and daemons based on ranking is proposed.

Reliability ranking assumption. Given an Orc expression E we assume that a ranking containing $\alpha(E)$ is available. This ranking is a measure (“objective” or “subjective”) of the reliability of the sites. This ranking can be independent of any orchestration E or conversely can depend strongly of the structure of E. Let $rk(s)$ be the rank of site s.

An orchestration assessor may use such a ranking to partition a set of sites $\alpha(E)$ into angel and daemon sets as follows:

$$\mathcal{A} = \{S \mid S \in \alpha(E), rk(S) \geq \lambda_E\} \ , \ \mathcal{D} = \{S \mid S \in \alpha(E), rk(S) < \lambda_E\}$$

λ_E is a ***reliability degree parameter*** fixed by the assessor following the suggestions of the client. We do not consider in this paper how λ_E is determined. The assessor will perform an analysis where sites in $\mathcal{A}$ perform *as well as possible* and sites in $\mathcal{D}$ perform *as badly as possible*. This is a way to perform an analysis lying between the two possible extremes "all is good" or "all is bad". We can argue this as follows. Sites with a rank higher than λ_E are "believed" by the assessor to have non-destructive behaviour. Sites with a rank lower than λ_E are unknown entities as far as the assessor is concerned and can have highly-destructive behaviour. The assessor supposes that during an evaluation of E a number of sites will fail:

- Let a small fraction β_E of angelic sites fail during the evaluation – thus, the number of failing angels is $\beta_E \times \#\mathcal{A} = \mathcal{F}(\mathcal{A})$. When an angel fails it does so in such a way as to *maximise* the value of the orchestration.
- Let a fraction γ_E of daemon sites fail. The number of failing daemons is $\gamma_E \times \#\mathcal{D} = \mathcal{F}(\mathcal{D})$. Failing daemons try to *minimize* the value of the orchestration.

For a given λ_E, β_E and γ_E the behaviour of E can be analysed:

- if λ_E is such that $\alpha(E) = \mathcal{A}$, then evaluation of the behaviour can be determined by solving a maximization problem (see §5).
- conversely, if λ_E is chosen such that $\alpha(E) = \mathcal{D}$, evaluation of the behaviour can be determined by solving a minimization problem (see §5).

If $\mathcal{A} \neq \{\}$ and $\mathcal{D} \neq \{\}$ a competitive situation arises and game theory [6] can be used to analyse system behaviour (see §6). Suppose that the set of failing sites is $a \cup d$ where $a \subseteq \mathcal{A}$, $\#a = \mathcal{F}(\mathcal{A})$ and $d \subseteq \mathcal{D}$, $\#d = \mathcal{F}(\mathcal{D})$. System behaviour is measured by $\varphi_{(a,d)}(E)$. The rewards (or utilities) of the angelic $\mathcal{A}$ and daemonic $\mathcal{D}$ players are $u_{\mathcal{A}}(a, d) = v(\varphi_{(a,d)}(E))$ and $u_{\mathcal{D}}(a, d) = -v(\varphi_{(a,d)}(E))$ (this is a zero sum game as $u_{\mathcal{A}}(a, d) = -u_{\mathcal{D}}(a, d)$). When $\varphi_{(a,d)}(E)$ is executed $\mathcal{A}$ receives $v(\varphi_{(a,d)}(E))$ from $\mathcal{D}$. The strategy a is chosen (by $\mathcal{A}$) to increase the value of $u_{\mathcal{A}}(a, d)$ as much as possible while d is chosen (by $\mathcal{D}$) to decrease this value as much as possible. Stable situations are Nash equilibria: a pure Nash equilibrium is a strategy (a, d) such that $\mathcal{A}$ cannot improve the utility by changing a and $\mathcal{D}$ cannot reduce the utility by changing d. When players choose strategies using probabilities we have mixed Nash equilibria. Let (α, β) be a mixed Nash equilibrium. As in zero sum games all the Nash equilibria have the same utilities, an assessor can measure the value of a program E by the utility of $\mathcal{A}$ on any Nash equilibrium. Given a Nash equilibrium (α, β), the expected benefit of an expression E is given by:

$$Assessment(E, rk, \lambda_E, \beta_E, \gamma_E) = v(\varphi_{(\alpha,\beta)}(E))$$

5. Bounded failures with one player games

The two extremes of behaviour can be determined through angelic and daemonic analysis:

Angelic failures. In an angelic analysis the viewpoint "the world is as good as possible even when failures cannot be avoided" is adopted. An angelic player $\mathcal{A}$ plays the game by choosing a list, $a = (a_1, \ldots, a_n)$, of failing sites for an expression E, when $\mathcal{F}(\mathcal{A}) = n$. Such a tuple a is called the *action* taken by the player $\mathcal{A}$. Defining $\alpha_+(E) = \alpha(E) \setminus \{0\}$ the set of eligible actions for $\mathcal{A}$ is

$$A_{\mathcal{A}} = \{(a_1, \ldots, a_n) | i \neq j \text{ implies } a_i \neq a_j \text{ and } \forall\, i : a_i \in \alpha_+(E)\}$$

To a strategy profile $a = (a_1, \ldots, a_n)$ we associate the set $\mathcal{F}_a = \{a_1, \ldots, a_n\}$ and the mapping $\varphi_{F_a} : \alpha_+(E) \to \alpha(E) \cup \{0\}$

$$\varphi_{F_a}(s) = \begin{cases} 0 & \text{if } s \in F_a \\ s & \text{otherwise} \end{cases}$$

is used to replace failing sites by 0 and to keep working sites unchanged. $\varphi_{F_a}(E)$ denotes the image of E under φ_{F_a}.

$$\text{ANGEL}(E, \mathcal{A}, \mathcal{F}(\mathcal{A})) = (\mathcal{A}, A_{\mathcal{A}}, u_{\mathcal{A}})$$

defines a one player ($\mathcal{A}$) strategic game ($\mathcal{A}$ is used to denote both the player and the set of sites controlled by this player): the set of actions is $A_{\mathcal{A}}$ and the utility $u_{\mathcal{A}} = v(\varphi_{F_d}(E))$. In this game, the angel $\mathcal{A}$ has to choose a strategy profile a giving a maximal utility. As there is only one player, there a maximization problem rather than a strategic conflict.

Daemonic failures. Daemonic failures are in a sense the opposite of angelic failures. In this case there is one player, the daemon $\mathcal{D}$ trying to maximize damage. We define $\text{DAEMON}(E, \mathcal{D}, \mathcal{F}(\mathcal{D})) = (\mathcal{D}, A_{\mathcal{D}}, u_{\mathcal{D}})$ such that $F_d = \{d_1, \ldots d_n\}$, $d = (d_1, \ldots, d_n)$ and

$$A_{\mathcal{D}} = \{(d_1, \ldots, d_n) | i \neq j \text{ implies } d_i \neq d_j \text{ and } \forall i : d_i \in \alpha_+(E)\}$$

Moreover, as $\mathcal{D}$ is intent on maximising damage, a long output is a bad result and thus $u_{\mathcal{D}}(d) = -v(\varphi_{F_d}(E))$. We can imagine $u_{\mathcal{D}}(d) = -v(\varphi_{F_d}(E))$ as a quantity of money that $\mathcal{D}$ has to pay, and naturally it is interested in paying as little as possible. In this case the rational behaviour of the daemon is formalized as a minimization problem.

EXAMPLE 2 *Let us consider two well-known expressions introduced in [2] . The first is a sequential composition of parallel expressions and the second is a parallel composition of sequential expressions:*

$$SEQ\_of\_PAR \triangleq (P|Q) \gg (R|S)\ ,\ PAR\_of\_SEQ \triangleq (P \gg Q)|(R \gg S)$$

Let us analyse both expressions with two failures. First, consider in detail a pure angelic behaviour. As we identify the player and the possible set of failures we have $\mathcal{A} = \{P, Q, R, S\}$ and the set of strategy profiles is

$$A_{\mathcal{A}} = \{(P, Q), (P, R), (P, S), (Q, R), (Q, S), (R, S)\}$$

The utilities are given in the table. In order to maximize the utility, in the case of SEQ_of_PAR, the angel has to avoid profiles (P, Q) and (R, S). In the case of PAR_of_SEQ the angel has to take precisely (P, Q) or (R, S). As expected, the daemon $\mathcal{D}$ behaves in the opposite way.

STRATEGY PROFILES	(P,Q)	(P,R)	(P,S)	(Q,R)	(Q,S)	*(R,S)*
Angel $\mathcal{A}$						
SEQ_of_PAR	0	1	1	1	1	0
PAR_of_SEQ	1	0	0	0	0	1
Daemon $\mathcal{D}$						
SEQ_of_PAR	0	-1	-1	-1	-1	0
PAR_of_SEQ	-1	0	0	0	0	-1

6. Two player games: the angel-daemon case

A strategic situation will occur when E suffers the effect of two players with opposite behaviour: an angel tries to minimize damage but, at the same time, a daemon tries to increase the damage. Let us consider in more detail this case. Let E be an Orc expression and assume that $\alpha_+(E)$ is partitioned into two disjoint administrative domains, the angel domain $\mathcal{A}$ and the daemon domain $\mathcal{D}$. We assume that $\mathcal{A} \cup \mathcal{D} = \alpha_+(E)$ and $\mathcal{A} \cap \mathcal{D} = \emptyset$. The notation $\mathcal{F}(\mathcal{A}) = p$ means that p sites will fail in $\mathcal{A}$. Similarly we note $\mathcal{F}(\mathcal{D}) = q$. We define

$$\text{ANGELDAEMON}(E, \mathcal{A}, \mathcal{D}, \mathcal{F}(\mathcal{D}), \mathcal{F}(\mathcal{A})) = (\mathcal{A}, \mathcal{D}, A_{\mathcal{A}}, A_{\mathcal{D}}, u_{\mathcal{A}}, u_{\mathcal{D}})$$

as follows. The players are $\mathcal{A}$ and $\mathcal{D}$ (as usual, we use the same letter to denote the player $\mathcal{A}$ and the set of sites controlled by this player). The strategy profiles are defined as follows.

- The *angel* $\mathcal{A}$ chooses p different failing sites $F_a = \{a_1, \dots a_p\} \subseteq \mathcal{A}$. Any call to a site in $\mathcal{A} \setminus F_a$ is successful. We associate with F_a the action

$a = (a_1, \dots, a_p)$. Formally, $\mathcal{A}$ has the following set of actions

$$A_{\mathcal{A}} = \{(a_1, \dots, a_p) | i \neq j \text{ implies } a_i \neq a_j \text{ and } \forall i : a_i \in \mathcal{A}\}$$

- The *daemon* $\mathcal{D}$, chooses q different failing sites $F_d = \{d_1, \dots, d_q\}$. Calls to sites in $\mathcal{A} \backslash F_d$ are successful. The daemon's action is $d = (d_1, \dots, d_p)$ and $A_{\mathcal{D}}$ will be the set of actions.

A *strategy profile* $s = (a, d)$ with $F_s = \{a_1, \dots, a_p, d_1, \dots, d_q\}$ fixes a priori the set of failing sites. Given E and F_s the length of $\varphi_{F_s}(E)$ is used to define the utilities as follows:

$$u_{\mathcal{A}}(s) = v(\varphi_{F_s}(E)) \ , \ u_{\mathcal{D}}(s) = -v(\varphi_{F_s}(E))$$

Note that ANGELDAEMON is a zero sum game because $u_{\mathcal{D}}(s) + u_{\mathcal{A}}(s) = 0$.

The players can choose the actions using probabilities. A mixed strategy for $\mathcal{D}$ is a probability distribution $\alpha : A_{\mathcal{A}} \to [0, 1]$ such that $\sum_{a \in A_{\mathcal{A}}} \alpha(a) = 1$. Similarly, a mixed strategy for the daemon player $\mathcal{D}$ is a probability distribution $\beta : A_{\mathcal{D}} \to [0, 1]$. A mixed strategy profile is a tuple (α, β) and the utilities are

$$u_{\mathcal{A}}(\alpha, \beta) = \sum_{(a,d) \in A_{\mathcal{A}} \times A_{\mathcal{D}}} \alpha(a)\beta(d)u_{\mathcal{A}}(a, b)$$
$$u_{\mathcal{D}}(\alpha, \beta) = \sum_{(a,d) \in A_{\mathcal{A}} \times A_{\mathcal{D}}} \alpha(a)\beta(d)u_{\mathcal{D}}(a, b)$$

As $\mathcal{A}$ and $\mathcal{D}$ have opposing interests there is a strategic situation and we recall the definition of Nash equilibrium as a concept solution.

DEFINITION 3 *A* pure Nash equilibrium *is a pair $s = (a, d)$ such that, for any a' it holds $u_{\mathcal{A}}(a, d) \geq u_{\mathcal{A}}(a', d)$ and for any d' it holds $u_{\mathcal{D}}(a, d) \geq u_{\mathcal{D}}(a, d')$. A* mixed Nash strategy *is a pair (α, β) with similar conditions.*

EXAMPLE 4 *Let us consider how to assess SEQ_of_PAR and PAR_of_SEQ under two different situations. In the first sites are ranked*

$$rk(P) > rk(Q) > rk(R) > rk(S).$$

Assume that the reliability parameter is such that $\mathcal{A} = \{P, Q\}$ and $\mathcal{D} = \{R, S\}$ and, moreover, $1/2$ of angelic sites will fail and $1/2$ of the demonic sites will also fail (therefore $\mathcal{F}(\mathcal{A}) = \mathcal{F}(\mathcal{D}) = 1$). Then $A_{\mathcal{A}} = \{P, Q\}$, $A_{\mathcal{D}} = \{R, S\}$ and the bimatrix games are

$$\begin{array}{cc|c|c|} & & \multicolumn{2}{c}{\mathcal{D}} \\ & & R & S \\ \hline \mathcal{A} & P & 1,-1 & 1,-1 \\ \hline & Q & 1,-1 & 1,-1 \\ \hline \end{array} \qquad \begin{array}{cc|c|c|} & & \multicolumn{2}{c}{\mathcal{D}} \\ & & R & S \\ \hline \mathcal{A} & P & 0,0 & 0,0 \\ \hline & Q & 0,0 & 0,0 \\ \hline \end{array}$$

SEQ_of_PAR $\qquad$ PAR_of_SEQ

In both cases any strategy profile (pure or mixed) is a Nash equilibrium and

$$Assessment(SEQ\_of\_PAR, rk, 1/2, 1/2, 1/2) = 1$$
$$Assessment(PAR\_of\_SEQ, rk, 1/2, 1/2, 1/2) = 0$$

Thus, for example, this assessment indicates to the client that, in the present environment, there is a reasonable expectation[1] *of obtaining* 1 *output when executing* PAR_of_SEQ.

Consider a second case with $rk'(P) > rk'(R) > rk'(Q) > rk'(S)$ *such that* $\mathcal{A} = \{P, R\}$ *and* $\mathcal{D} = \{Q, S\}$ *with* $\mathcal{F}(\mathcal{A}) = \mathcal{F}(\mathcal{D}) = 1$.

$$\begin{array}{cc|c|c|} & & \multicolumn{2}{c}{\mathcal{D}} \\ & & Q & S \\ \hline \mathcal{A} & P & 0,0 & 1,-1 \\ \hline & R & 1,-1 & 0,0 \\ \hline \end{array} \qquad \begin{array}{cc|c|c|} & & \multicolumn{2}{c}{\mathcal{D}} \\ & & Q & S \\ \hline \mathcal{A} & P & 1,-1 & 0,0 \\ \hline & R & 0,0 & 1,-1 \\ \hline \end{array}$$

SEQ_of_PAR PAR_of_SEQ

Here no game has pure Nash equilibria. There is only one mixed Nash equilibrium with $\alpha(P) = \alpha(R) = \beta(Q) = \beta(S) = 1/2$, *and in this case the angel has utility* $1/2$ *and the daemon has utility* $-1/2$. *In this case*

$$Assessment(SEQ\_of\_PAR, rk', 1/2, 1/2, 1/2) = 1/2$$

and similarly for PAR_of_SEQ. *This assessment indicates to the client that, in the present environment, an output of* 1 *or* 0 *results (with equal likelihood) is a reasonable expectation.*

EXAMPLE 5 *Consider the expression* $(P \mid Q \mid R) \gg (S \mid T \mid U)$ *with* $\mathcal{A} = \{P, Q, S\}$ *and* $\mathcal{D} = \{R, T, U\}$ *with* $\mathcal{F}(\mathcal{A}) = 2$ *and* $\mathcal{F}(\mathcal{D}) = 1$

$$\begin{array}{cc|c|c|c|} & & \multicolumn{3}{c}{\mathcal{D}} \\ & & R & T & U \\ \hline & (P,Q) & 0,0 & 2,-2 & 2,-2 \\ \hline \mathcal{A} & (P,S) & 2,-2 & 2,-2 & 2,-2 \\ \hline & (Q,S) & 2,-2 & 2,-2 & 2,-2 \\ \hline \end{array}$$

$(P \mid Q \mid R) \gg (S \mid T \mid U)$

The pure Nash equilibria are $((P,S),R)$, $((P,S),T)$, $((P,S),U)$, $((Q,S),R)$, $((Q,S),T)$ *and* $((Q,S),U)$. *In this case the assessment says that* 2 *outputs is the reasonable expectation.*

[1]Here "reasonable expectation" is meant in the everyday sense of the phrase and is not intended to represent a probabilistic outcome.

7. Manager's placement problem

The muskel [1] skeleton-based programming system provides an example of the kind of application that can be analysed using this approach. In muskel skeletons are implemented using macro data flow instruction graphs. Macro data flow instructions are placed on (potentially unreliable) remote worker sites for execution and the results are returned to the muskel manager for consolidation. A macro data flow graph may be modelled using an Orc expression. The behaviour of the muskel system in an untrusted environment may be analysed by considering its operation under the following assumptions:

- Data flow interpreters are unreliable.
- There is no recovery mechanism.
- Some sites may fail, but all cannot fail simultaneously. This means that the number of failures is bounded.
- An application manager tries to maximize the number of outputs that are generated.

Suppose that n remote data flow interpreters $\mathcal{I} = \{I_1, \ldots I_n\}$ are available. The manager has to place the set $\alpha_+(E)$ of macro insh tructions on the different macro data flow interpreters. Assume that interpreters are partitioned into two groups: the first group tries to minimize damage and we call this group the angel; the other group of interpreters behaves like a daemon. Moreover, in both cases the number of failures is bounded. A basic question in this case is, which is the best placement? Here we do not develop a general answer, but just consider a worked example.

EXAMPLE 6 *Consider the $SEQ\_of\_PAR \triangleq (P \mid Q) \gg (R \mid S)$ with $\mathcal{I} = \{I_1, I_2\}$ where I_1 is very reliable (angelic) and I_2 is very unreliable (daemonic). Assume that the manager has to place two macro instructions on one interpreter and the other two on the other interpreter. We assume that in both cases the interpreters will execute half of the assigned load. What is the best placement? There are several possibilities.*

- *Consider the outputs when the manager places P and R in I_1 (the angel) and Q and S in I_2 (the daemon). If I_1 makes P fail, I_2 will force Q to fail and there is no output. This situation is not ideal for I_1 (the angel) because changing the failure from P to R the output improves to 1. This new situation is not too bad for I_2 (the daemon) because changing the failure from Q to S worsens the output to 0. This situation is also unstable. This never ending behaviour can be analyzed by a bimatrix game having a mixed Nash equilibrium with expected output $1/2$.*

- *If P and Q are placed in one interpreter and R and S are placed in the other, the output is* 1. *This is the best we can obtain.*

8. Conclusions

One of the defining characteristics of grid programming is its dynamicity. Typically, the grid user has significantly less control over resources employed than in traditional scenarios: sites used for the execution of application components may fail. Thus, an important consideration for practical grid applications is the provision of an assessment of the quality of the application based on the expected performance of its constituent execution sites. In terms of an Orc expression, E, used to model the application, an ordered list for $\alpha(E)$ is needed. How this list should be built is unclear and is perhaps a controversial question: the likely behaviour of a site may depend on subjective perceptions of its qualities. In drawing up the list two distinct classes of consideration may be identified:

- Aspects independent of the application. Here we consider "stand-alone" qualities of a site. For example, the availability of proxies for a site may be regarded as enhancing its reliability. We might also take into account "the reputation" of the sites [8].
- Aspects depending of the application. The designer has *a priori* knowledge of the available potential sites, $\mathcal{S}$. >From $\mathcal{S}$ the designer has to choose $\alpha(E)$. Once $\alpha(E)$ has been determined the orchestration has to be developed. We suggest that in many cases the development of the application and the rank of $\alpha(E)$ are inextricably linked. For example, a site used only as a "thread" in a parallel search may fail with little consequence for the application; failure of a site which forms a constituent of a sequential backbone of an application will be catastrophic for the application.

The ordering of $\alpha(E)$ depends also on the perception of the risk. Different people have different perceptions of risk and will rank sites accordingly. For instance, consider a database D with no back-up available. Assume that D is crucial for the application so that a failure in D significantly harms the application. There are two possibilities for ranking D:

- Moderate optimism. As a failure of D harms the application, an optimistic view will rank D among the angels. In this way the angel will try to avoid having D fail, but if it does fail then the outcome will fall far short of expectation.
- Safe pessimism. Since D is crucial to the application, D is ranked among the daemons, so that the outcome (likely failure of D), although far from ideal, is at least predictable and uncertainty is removed.

To better understand these points consider the following “gedanken experiment”. Imagine that in the sixties you are asked to build an orchestration for a global war involving nuclear weapons and conventional arms. In order to assess how pessimistic is the orchestration, should you place the nuclear weapons among the daemons or among the angels? If you choose the former nuclear catastrophe ensues (in the simulation); if you choose the latter there is probability of survival.

Failure of grid sites is a reality of grid computing and forms a significant part of its challenge. Assessing the likelihood of success of an application requires both an evaluation of the quality of its constituent sites and a means of combining the results to measure the quality of the assembly. We propose the use of Orc together with game theory as a way of addressing the latter point; the assessment of individual sites and the establishment of a ranking among them remain open questions, touching as they do upon issues such as degree of trust and perception of risk, issues which remain largely subjective.

References

[1] Aldinucci, M., Danelutto, M.: Algorithmic skeletons meeting grids. *Parallel Computing* 32 (7): 449–462, 2006.

[2] Bougé, L.: Le mòdele de programmation à parallélisme de donés: une perspective sémantique. *Techniques et science informatiques*, Vol 12, 5, 541–562, 1993.

[3] Eliaz, K: Fault Tolerant Implementation. *Review of Economic Studies*, 2002, vol. 69, issue 3, pages 589-610.

[4] Misra J., Cook, W.: Computation Orchestration: A basis for wide-area computing. *Software & Systems Modeling*, 2006. DOI 10.1007/s10270-006-0012-1.

[5] Moscibroda, T., Schmid, S., Wattenhofer, R: When selfish meets evil: byzantine players in a virus inoculation game. PODC’06, 35 - 44, 2006.

[6] Osborne, M., Rubinstein, A.: *A Course on Game Theory*, MIT Press, 1994.

[7] Peltz, C: Web Services Orchestration and Choreography. IEEE Computer 36(10): 46-52 (2003)

[8] Silaghi, G.C., Arenas, A., Silva, L.: Reputation-based trust management systems and their applicability to grids. Technical Report TR-0064, Institute on Programming Models, CoreGRID-Network of Excellence, March 2007. http://www.coregrid.net/mambo/images/stories/TechnicalReports/tr-0064.pdf 21 May 2007.

[9] Stewart, A., Gabarró, J., Clint, M., Harmer, T., Kilpatrick, P., Perrott, R.: Estimating the reliability of web and grid orchestrations. In Gorlatch, S., Bubak, M., Priol, T., eds.: Integrated Research in Grid Computing, Krakóv. Poland, CoreGRID.

[10] Stewart, A., Gabarró, J., Clint, M., Harmer, T., Kilpatrick, P., Perrott, R.: Managing Grid Computations: An ORC-Based Approach. In ISPA 2006, LNCS Vol. 4330, pp. 278–291, 2006.