I^3: An Architecture for Integrity Management on the Grid

Tope Olukemi and Lionel Sacks

University College London

Abstract: Large scale science is increasingly being carried out through distributed global collaborations. Such global collaborations require distributed computing resources spanning multiple domains, which are available to large numbers of users. The management of such systems raises issues of local integrity and global integrity. This paper attempts to address these using the I^3 (Integrity, Information and Intelligence) architecture. An overview of the architecture is presented in this paper. Preliminary results from simulations of the architecture are also presented.

1 Introduction

E-science is encouraging research in the area of large scale science which is increasingly being carried out through distributed global collaborations enabled by the Internet. Such collaborative scientific enterprises will require access to very large data collections, very large scale computing resources and high performance visualisation back to the individual user scientists [1]. The Grid [2] is an architecture proposed to bring all these issues together.

The Grid can be viewed as a global extension to cluster computing. Clusters, built with commercial off-the-shelf hardware and software components, compared to traditional super-computing usually have the advantages of: a much better price-performance ratio, the flexibility to size the cluster according to requirements and the flexibility to choose the components (hardware and software) which constitute the cluster. Clusters tend to be located in a single location and very often the cluster tends to be homogeneous. The Grid on the other hand is heterogeneous provides wide-spread dynamic, flexible and coordinated sharing of geographically distributed networked resources, among dynamic user groups [3]. The computers that make up a Grid may be distributed across multiple organisations and administrative domains.

Integrity is the quality of an information system reflecting the logical correctness and reliability of the operating system; the logical completeness of the hardware and software implementing the protection mechanisms; and the consistency of the data structures and occurrence of the stored data [4]. From this general definition of integrity it can be divided into three areas – data integrity, system integrity and service integrity. The focus of this paper is on system integrity which is defined as that condition of a system wherein its mandated operational and technical parameters are within the prescribed limits [4]. In other words when a system performs its intended function in an unimpaired manner, free from deliberate or inadvertent unauthorized manipulation of the system [4] it can be said to have system integrity.

With a variety of applications running in such a large, distributed, heterogeneous environment a Grid operator is concerned with integrity from two viewpoints – firstly he wants to ensure that his Grid platform is not compromised by ensuring that processes running on his platform behave the way they're expected to behave. Secondly he wants to be able to provide guarantees that processes running on his platform will not have their integrity compromised by his platform or by other processes.

2 I^3 architecture

Our I^3 architecture consists of light-weight, self-organising and policy controlled management elements. On each node (a node could be a single computer or a cluster of computers) there is an I^3 management element consisting of one or more building blocks (Figure 1). Each building block is a Java class and an XML file is used to describe the component blocks of an I^3 element and how the blocks are linked together. The static policy-based security services were developed in [5] and [6]. The dynamic blocks are being developed in the context of an EPSRC e-science project: *SO-GRM* (Self-organising Grid Resource Management) [3]. Figure 1 shows a typical SO-GRM scenario. A user and a Grid operator negotiate an SLA (service level agreement). The SLA defines the resources the operator is committing to the user. As a result of over provisioning the operator has virtual commitments and actual commitments, these

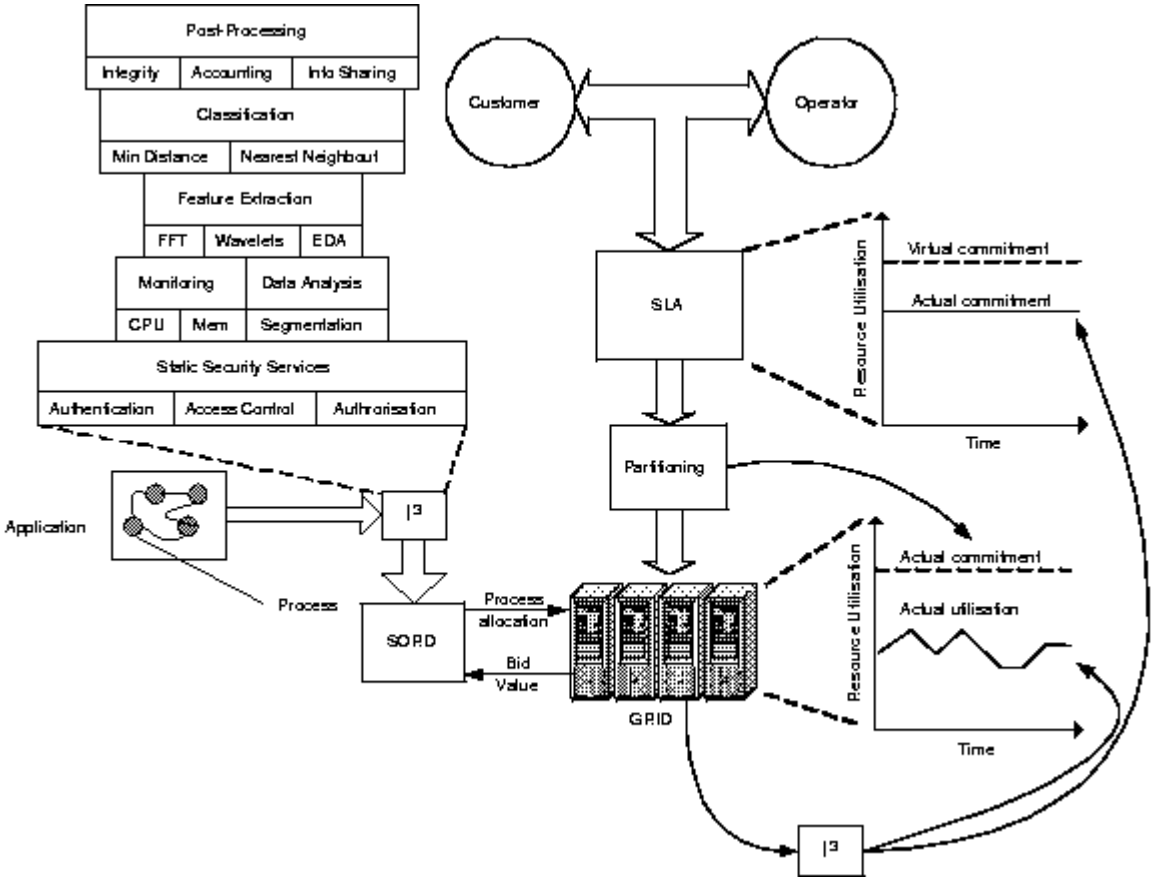

Figure 1: Typical SO-GRM Scenario

commitments are translated into policies which are used to partition resources on individual nodes within the Grid. A user usually wants to run an application which consists of one or more processes. When a new process comes in I^3 provides static security services such as authentication and access control, while *SORD* (self-organising resource discovery) allocates the process to an appropriate node. Once the process is running I^3 monitors the resource utilisation of the process and the data gathered is analysed and classified. This allows the Grid operator at the node level to validate that the process is behaving as it should and to monitor the actual resource utilisation on the node. This ensures integrity of the local node and processes running on the local node. At a higher level information from I^3 about individual processes can give the operator an idea of how an application is behaving, while on the global level an idea of overall system integrity is provided by giving snapshots of global current utilisation, application integrity and process integrity. We can also derive accounting information which can prove useful when negotiating new SLAs.

3 Simulation

In our first simulation scenario we looked at the behaviour of an application consisting of a number of processes. For the classification element of our I^3 element we implemented a minimum distance mahalanobis classifier [7]. The mahalanobis distance was calculated using equation 1.

$$r^2 = (\mathrm{x}-\mu)^{\mathrm{t}}\mathbf{C}^{-1}(\mathrm{x}-\mu)$$
where r is the mahalanobis distance,
x is the test feature vector, (1)
μ is the mean feature vector and
$\mathbf{C}$ is the covariance of the mean feature vector.

The duration of the processes was assumed to follow a normal distribution. We generated 1000 training sets (of process duration) which were used to train the classification component of an I^3 element. The

training sets were generated from a Gaussian random number generator with a mean of zero and a standard deviation of one. The features we extracted from the training sets were mean, standard deviation and skew. The mean of each of these features was used to generate our mean feature vector (μ). A test set consisting of 1000 data sets generated with the same Gaussian random number generator was used to tune our classifier and establish a threshold mahalanobis distance. Then we generated another series of test sets (still using a Gaussian random number generator with mean of zero and standard deviation of one) varying the skew between 0.01 and 0.2 respectively in each set with a step size of 0.01 and established new threshold mahalanobis distances for each step increase in skew. After establishing these thresholds we tested each threshold with a another series of 1000 test data sets generated with a Gaussian random number generator of mean of zero and standard deviation of one but with a uniformly distributed skew between 0 and 0.3. At each threshold we measured the percentage error which was the percentage of false positives and false negatives measured at that threshold.

In the second simulation scenario we looked at a process running on a local node. We assumed that the process went through the same state transitions every time and as a result we modelled the process as consuming resources in a cyclic manner using a sinusoidal cycle. We generated 1000 training sets and used it to train our mahalanobis classifier. The features we used were the coefficients from taking a FFT (fast fourier transform) of the sinusoid function. We created a number of test data sets of our original function and randomly added white, Gaussian noise to it. We took the FFT of these test sets and measured the error rate by comparing the new mahalanobis distance with the threshold value obtained from our training set. We repeated this while increasing the signal to noise ratio.

4 Results

The graphs in Figure 2 show results obtained from our first simulation scenario. Figure 2a shows the

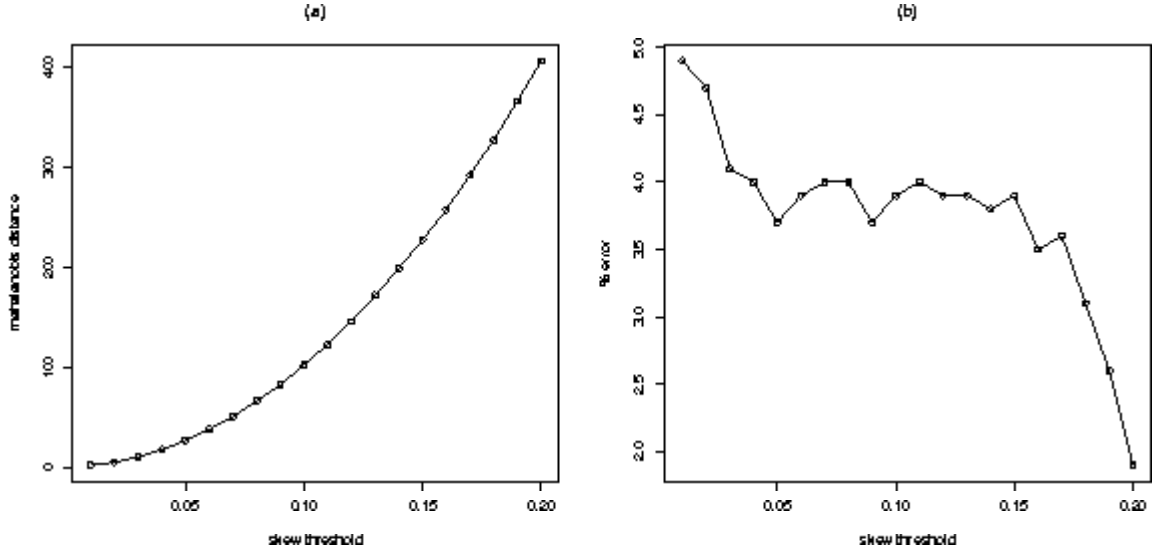

Figure 2: (a) Variation in mahalanobis distance as skew threshold is increased (b) Variation in percentage error as skew threshold is increased

variation in mahalanobis distance for each skew threshold. As we would expect, the mahalanobis distance increases as the skew threshold increases because of the increased distance between the feature vector for the skew threshold we measuring and the mean feature vector. Figure 2b shows the percentage error measured at different skew thresholds. The error is initially high (high number of false positives and negatives) for a low skew threshold and it drops as the skew threshold increases. Of particular interest is a relatively constant region between the threshold values of 0.05 and 0.15. This points to some inherent robustness in the classifier we have used.

The graph in figure 3 shows results from our second scenario. As the SNR is increased, the percentage error initially drops rapidly and then it levels off giving us a constant SNR from 6dB to 9dB.

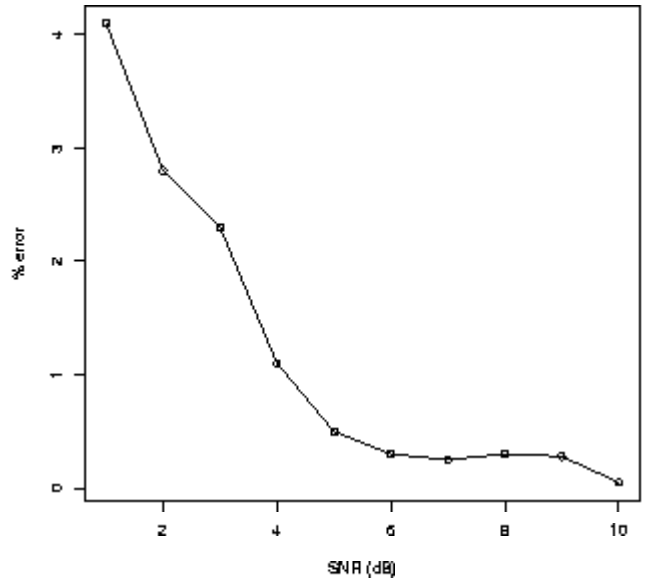

Figure 3: Variation in error rate with increasing signal-to-noise ratio

5 Conclusion

Management of Grid environments will require mechanisms which enable a Grid operator to maintain the integrity of his Grid and to provide guarantees to users regarding the integrity of their applications. Such mechanisms must be flexible, autonomous, adaptive and decentralised. In this paper we have presented a policy-based architecture which is inherently distributed, adaptive and decentralised. The classification component of our system was simulated in two different scenarios and the initial results hold promise and indicate that our architecture could provide integrity in a Grid environment.

References

[1] UK Research Councils. Research Council e-science Core Programme.

[2] I. Foster and C. Kesselman, editors. *The Grid: Blueprint for a New Computing Infrastructure*. Morgan-Kaufmann, 1999.

[3] Ioannis Liabotis, Ognjen Prnjat, Tope Olukemi, Adrian Li Mow Ching, Aleksandar Lazarevic, Lionel Sacks, Mike Fisher, and Paul Mckee. Self-organising management of Grid environments. 2003.

[4] National Security Telecommunications and Information Systems Security Instruction. National Information Systems Security (INFOSEC) Glossary. NSTISSI No. 4009, January 1999. Revision 1.

[5] Tope Olukemi, Ioannis Liabotis, Ognjen Prnjat, and Lionel Sacks. Security and Resource Policy-based Management Architecture for ALAN Servers. In Dominique Gaiti and Nadia Boukhatem, editors, *Network Control and Engineering for QoS, Security and Mobility*, pages 91–102. IFIP, Kluwer Academic Publishers Group, October 2002.

[6] Ognjen Prnjat, Ioannis Liabotis, Tope Olukemi, Lionel Sacks, Mike Fisher, Paul Mckee, Ken Carlberg, and Grigorio Martinez. Policy-based management for alan-enabled networks. In *Proceedings of the third international workshop on policies for distributed systems and networks (Policy 2002)*, pages 181–192. IEEE, June 2002.

[7] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification*. John Wiley and Sons, 2nd edition, 2001.