A framework for Grid-based failure detection in an automated laboratory robot system

C. Foulston and A. Clare

Department of Computer Science, University of Wales Aberystwyth,
Penglais, Aberystwyth SY23 3DB
afc@aber.ac.uk

Abstract

We have designed and produced a framework for Grid-based failure detection to monitor and report on our automated laboratory equipment and the Robot Scientist. Its features are distributed agent based monitoring, selective reporting and report dispatch brokering. Monitoring and reporting agents can be distributed due to the loose coupling of Web Services, enabling a large enviroment of parameters to be monitored. Reporting is via different mediums such as e-mail, instant PC alerts and text messaging, though any type of agent can sign up for new reports, making the system expandable to a variety of needs. Dispatch brokering allows agents and humans to sign up for the latest reports for further analysis.

1 Introduction

The "Robot Scientist" is a state-of-the-art e-Science system for automating the scientific process [3]. It comprises automated and integrated laboratory equipment together with intelligent software for creating hypotheses, designing the high throughput experiments and analysing the results. The system is currently used to conduct yeast mutant growth experiments in order to investigate gene function in metabolic pathways. The intelligent software provides closed loop learning, whereby the results of the previous experiment are fed back into the system in order to refine hypotheses and choose and conduct the next round of experiments automatically.

The system must function unaided for long periods of time. The average yeast growth period measured will be 5 days, and experimental batches are overlapped, so that operation is continuous. The equipment is located in a laboratory in a different room to that in which the Robot Scientist project researchers work. The yeast is grown up in a pregrowth phase lasting approximately 24 hours, which is followed by a growth phase lasting a couple of days. Important events may happen at any time of day or night, and our lab technician will not always be around to watch.

We need an remotely accessible automated failure detection system for the Robot Scientist. This should:

- monitor equipment
- log information, errors and warnings
- take intelligent decisions and act upon them
- notify users of problems by a variety of means
- provide users with information (current and historical) and suggest possible actions to be taken
- keep records of previous problems, solutions and actions

In order to make intelligent decisions about whether a potential failure has been detected, and what action to take, we need to employ a variety of reasoning methods in the system, analysing a wide range of data. The detection processes are discussed in Section 3.

The failure detection system has to be available to monitor remotely from a variety of OS platforms, and it has to be secure and reliable. Providing Web Services interfaces to its functionality is a good way to achieve this, and we chose to use the Globus toolkit [2] to provide some of the basic infrastructure. The interfaces and general software architecture is discussed in Section 4.

2 Background

2.1 Robot Scientist

The Robot Scientist uses a large collection of laboratory automation equipment to conduct the experiments that are designed by the artifical intelligence software. The equipment is capable of growing yeast knockout mutant strains under a variety of experimental conditions. Over 1000 experiments can be performed each day, and the main experimental outputs are optical density readings that are used to plot the growth curves of the yeast strains. A large amount of experimental metadata

is also available from each component of the lab automation equipment.

The robot components are controlled and coordinated by software. The whole system is continually supplied with descriptions of experiments that have been designed to test hypotheses created by AI software. The system should run continuously, 24 hours a day, 7 days a week.

Any failure of this system needs to be detected as early as possible, so that action can be taken, and the equipment can be used to its maximum potential. A single fault can cause the entire system to be unusable, and all experiments currently running (which may have been running for days) may need to be abandoned. Faults can range from an obvious show-stopping breakdown to a slight variation of some condition, which could still have a disastrous effect on the results of experiments.

2.2 Existing e-Science labs and failure detection systems

Very little laboratory-based and experimental equipment is actually Grid-enabled yet. The Grid is still an evolving concept and toolkits such as Globus have not yet matured enough for most industrial use.

Ko et al [4] describe a laboratory with a web-based interface, designed to be accessed remotely by students to run coupled tank control experiments. The system provides feedback to the students, by video, audio and by data, such as plots of response curves. The students can access the remote laboratory 24 hours a day. The potential of remote biology labs used for education is explored further by Che [1].

NEESgrid is a large grid-based system for earthquake simulation and experimentation. Laboratories are linked via grid infrastructure to compute resources, data resources and equipment (`http://it.nees.org/`). Various instruments and sensors can be monitored and viewed remotely, and NEESgrid provides teleobservation and telepresence via video streams, data streams and still images.

The DAME project is a major e-Science pilot project providing a distributed decision support system for aircraft maintenance. They proposed a Grid based framework for fault diagnosis and a implementation for gas turbine engines [5]. Their work is perhaps the most similar to ours, though for a very different application, and used Globus Toolkit version 3 to provide web service interfaces. They note the benefits of using a loosely coupled service-oriented architecture to implement a variety of fault diagnostic services. A engine simulation service, case based reasoning services and event sequence/history monitoring services are provided.

Otherwise, fault detection in Grid-based systems has so far been mostly targeted towards analysis of network faults, and detection of problems in compute clusters. Tools for monitoring and fault detection exist, but are currently mostly restricted to monitoring network traffic and general performance of Grid services (response time, availability, etc). Globus has a Monitoring and Discovery System (MDS) component that allows querying and subscription to published data. It is intended to be an interface to other monitoring systems, and to allow a standard interface to the data. Other general grid service monitoring tools include Gridmon[1], Network Weather Service[2], Netlogger[3], Inca[4] and Active Harmony[5]. An annual Grid Performance Workshop[6] reflects current research in these systems.

Our laboratory automation system is complex and contains many potential sources of failure, including human error, hardware error and experimental error. In the next section we describe these and the methods of failure detection in more detail.

3 System requirements

3.1 Sources of information

The primary sources of information for use in detecting failure will be:

Equipment metadata logs: Each piece of equipment logs information such as event timings and internal settings and variation.

Experimental data/observations: The optical density readings of the yeast can show that something is wrong. For example, an oscillating pattern of readings for a microtitre plate lead us to discover that the two incubators between which the plate was being cycled were set to agitate at different speeds.

Experimental expectations: The Robot Scientist work is perhaps unique in having the whole process automated, so we can make use of the fact that the expected result of every experiment is automatically available for comparison to the actual result. Of course, an unexpected result may also be due to the discovery of new scientific knowledge.

[1] Gridmon: `http://www.gridmon.dl.ac.uk/`
[2] NWS: `http://nws.cs.ucsb.edu/`
[3] Netlogger: `http://www-didc.lbl.gov/NetLogger/`
[4] Inca: `http://inca.sdsc.edu/`
[5] Active Harmony: `http://www.dyninst.org/harmony/`
[6] Grid Performance Workshop: `http://www-unix.mcs.anl.gov/~schopf/GPW2005/`

Temperature/humidity sensors: These are scattered throughout the system, monitoring the ambient conditions, rather than instrument-specific conditions.

Webcams: In the future, image recognition should be able to distinguish some fault states, and provide more information on the causes of robotic arms becoming blocked, or tips being dropped. Experiment imaging, such as closeups of plates or cells could also provide valuable information.

3.2 Types of failure

Hardware error: The equipment consists of many integrated automated lab machines. These include incubators, liquid handlers, a washing station, a centrifuge, microplate readers and several robot arms and shuttles for moving plates around the system. Each of these components has a software interface which can report errors that the machine can detect (such as being wrongly initialised, overheating, being full, or dropping what it was carrying). We have already experienced power cuts, freezer motor seizure, dropped tips, misaligned robot arms, and a host of other hardware issues. Human error can also contribute to hardware reports. If the technician did not provide the system with the enough plates for the experiment then at some point there will be none left on the stack to take (and similarly for emptying of waste). While human errors are mostly avoidable, they still occur. Hardware errors due to imperfections in plastic tip mouldings, cable stretch over time, loss of vaccuum suction, etc, are much more difficult to avoid.

Experimental error: Some errors may be less obvious, and may only be noticeable when the experimental results are interpreted in context, or compared to some model of what was expected. For example, if the liquid hander is not aspirating the quantity of liquid that it claims to have aspirated, this may show up in altered plate readings. Similarly, if the plate readings were not as expected this could also point to altered growth temperature, contamination or misplaced strains. If wells at one side of a plate grew less well than wells at the other side it might be suspected that conditions across the plate were uneven, for example air flow, oxygen availability, evaporation or temperature.

Software error: Software errors are much more difficult to detect. There is a huge field of research devoted to the design of error-free software, incuding methods for debugging, refactoring, formal specification, and code development practices. With existing, pre-installed software our aim is simply to detect if a fault has occurred that is likely to have a software origin so that it can be fixed. However, these can be very complex and manifest in unusual ways.

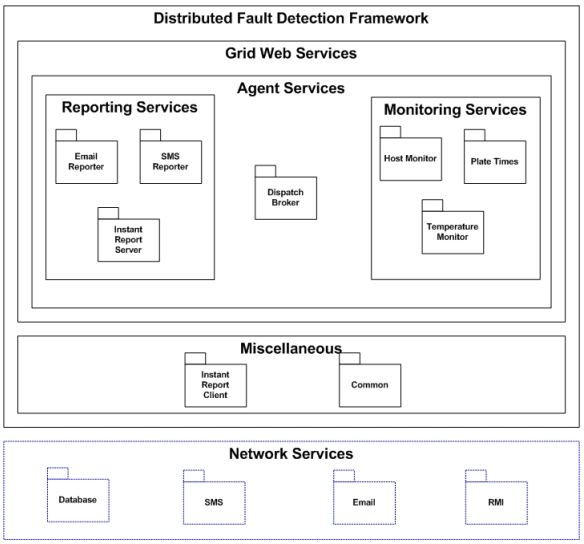

Figure 1: Framework overview, with examples of the Monitoring Agents

3.3 Detection Methods

A variety of different methods will be necessary to detect the different types of faults. These will include case based systems, model based systems, use of historical data, use of experimental expectations and hypotheses, use of event data and precise timings, knowledge of stock control of nutrients, yeast strains and growth media, use of hardware logs and error codes, and many other analyses.

4 Implementation

4.1 Framework

The framework is abstract enough to be applicable to most lab automation systems. Figure 1 shows the components of the framework.

4.2 Reporting Agents

The current reporting agents that bind to the dispatch broker are a text message reporter, an email reporter and an instant desktop reporting tool. Reporting to a user's machine is the quickest way but users may not always be at their desks. Therefore text messaging is an important method for urgently attracting attention. Selective reporting is used here to specify times for this as researchers are not always on duty.

4.3 Abstract Monitoring Agents

Example implementations of the abstract monitor are:

- The incubator monitor, which monitors temperature, humidity, O_2 and CO_2 levels of three incubators. This reports to the dispatch broker when thresholds are not met. The yeast must be grown in a stable, controlled and measurable enviroment. Only with a wide array of different environmental state monitors will it be possible to ensure all experiments were done in a similar enviroment.
- A host monitor which simply monitors the main control PC running the robot. The host monitor's job is to ensure that all control PCs are up and running at all times and to report this as a high alert to the dispatch broker. When PCs fail this will waste experimental time and most likely ruin current experiments.

Monitoring Agents are separated from the business of making decisions about who to report to or how to report by the dispatch broker. This allows a Monitoring Agent to focus just on the detail of monitoring. This also allows Monitoring Agents to be composed of other Monitoring Agents in a hierarchical manner. A more complex Monitor that relies on several aspects of the system can be built using the results of several low level Monitoring Agents, hence avoiding time-consuming repetition of low level tests.

Parameters that need monitoring come from many different systems such as databases, file-based data logs, computational tasks or querying serial or USB interfaces to discover continuous parameters, and these may not be in close quarters, therefore it is essential to have a loose coupling for agents.

4.4 Dispatch Broker

The dispatch broker recives reports from any agent and notifies all outbound agents that a new report has been delivered. This ensures the loose coupling between detecting and reporting.

4.5 Selective Reporting

This allows users to select which reports they wish to receive.

This is to ensure that information passed on is relevant in the context of the user it is sent to. On such a large system as the Robot Scientist, researchers with completely different backgrounds will be interested in different reports. Biologists may not want to concern themselves with computer hardware faults and computer science researchers may not want to concern themselves with warnings about low liquid levels. This uses a relational database where the relations between researchers and monitors are stored.

5 Discussion

Detecting and recording errors and failures is essential for any highly complex system of many different parts. Even fully automated biological experiments are prone to noise, and careful monitoring of conditions and experimental metadata is crucial for correct interpretion of results. We need to be able to detect problems as they occur, rather than days later when valuable experimental time and resources have been wasted.

The failure detection system that has been developed is general enough to be applicable to most laboratory automation environments. It uses Globus/Web Services to provide a loose coupling of components. It is in place on the Robot Scientist with the most immediately essential monitoring agents, and we now need to extend its capabilities with a wide variety of more complex monitoring agents. More intelligent agents will provide diagnosis as well as detection, and intelligent, user-friendly diagnosis of biological problems will be an interesting area of research in its own right.

Acknowledgements

The authors would like to thank Dr Andrew Sparkes for valuable discussions.

References

[1] A. Che. Remote biology labs. In *E-ducation Without Borders*, 2005.

[2] I. Foster. Globus toolkit version 4: Software for service-oriented systems. In *IFIP International Conference on Network and Parallel Computing*, pages 2–13. Springer-Verlag LNCS 3779, 2005.

[3] R. D. King, , K. E. Whelan, F. M. Jones, P. G. K. Reiser, C. H. Bryant, S. Muggleton, D. B. Kell, and S. G. Oliver. Functional genomic hypothesis generation and experimentation by a robot scientist. *Nature*, 427(6971):247–252, 2004.

[4] C. C. Ko, B. M. Chen, J. Chen, Y. Zhuang, and K. C Tan. Development of a web-based laboratory for control experiments on a coupled tank apparatus. *IEEE Transactions on Education*, 44(1), 2001.

[5] X. Ren, M. Ong, G. Allan, V. Kadirkamanathan, H. A. Thompson, and P. J. Fleming. Service oriented architecture on the Grid for FDI integration. In *Proc 3rd UK e-Science All Hands Meeting (AHM 2004)*, 2004.