The System Management of the ICL GOLDRUSH Parallel Database Server

Paul Watson,
Department of Computing Science, University of Newcastle,
Newcastle-upon-Tyne, NE1 7RU, UK
(this work was carried out while the author was at ICL High Performance Systems)

Mike Ward and Ken Hoyle,
ICL High Performance Systems, West Gorton, Manchester, M12 5DR, UK

About the Authors

Paul Watson joined the Computing Science Department at Newcastle University in 1995, having spent the previous 5 years working on the Goldrush Project in ICL High Performance Systems. It was during this time that the work described in this paper was carried out. Before joining ICL, he was a lecturer in the Computer Science Department at Manchester University from 1986-9.

Ken Hoyle is the System Architect with responsibility for System Management in ICL High Performance Systems.

Mike Ward is a Senior Development Manager with ICL High Performance Systems, currently working on Parallel System Kernels. His previous responsibilities include leading the Goldrush System Management development team.

Abstract

The *GOLDRUSH Mega***SERVER** is a very high performance parallel database server sold by ICL. It consists of up to 64 Nodes each of which can collaborate to speed-up the execution of large, Decision Support queries, or high transaction workloads. Most uses of GOLDRUSH are for business-critical applications which require very high levels of availability, integrity and manageability.

The design of the System Management of GOLDRUSH is the focus of this paper. It differs from conventional System Management as it must support both the underlying parallel machine and the business critical applications which run on it.

We describe the requirements placed on System Management by this class of machine, and how these were met in the design of the GOLDRUSH management system: we explain both the management architecture and the management tools themselves. Finally in the light of experience in the use of these tools we point to future directions for work in this area.

1. INTRODUCTION

System Management is a vitally important but often overlooked component of any computer system to be sold into the commercial marketplace. This is particularly true of systems running business-critical applications in which the ability of the business to function can be undermined by failures in the computer system. Examples of these systems include Telephone Insurance Sales in which customers can phone to receive an insurance quotation. The telephone operator asks the potential customer questions and uses the answers to query a database that holds the rates of premiums for different categories of driver. If the computer system is down then the company cannot issue quotations, and so they lose business. In this environment, System Management software is required both to reduce the risk of a system failure, and to reduce the time it takes to diagnose a problem and get the system working again.

Recently, computer manufacturers have begun to produce parallel systems aimed to the commercial customer. Most of these consist of a set of compute Nodes connected by a fast network. Each Node runs its own instance of an Operating System, usually UNIX. Without System Management tools aimed specifically at the management of parallel systems, each Node has to be managed separately, and as there can be tens of Nodes, the effort required to do this could be very large.

GOLDRUSH is an example of this new class of parallel machines aimed at business-critical applications [Watson & Catlow 1990]. This is a very high performance parallel database server sold by ICL. It consists of up to 64 Nodes each of which can collaborate to speed-up the execution of large, Decision Support queries, or high transaction workloads.

Much work has been carried out into the System Management of computers. However, very little work has been done in the management of systems, such as GOLDRUSH, which are both business-critical and parallel. It is this increasingly important area that is addressed by this paper.

The organisation of the rest of this paper is as follows. Firstly it gives an overview of the GOLDRUSH System (Section 2). Next it describes the requirements placed on System Management by this class of machine, and how these were met in the design of the GOLDRUSH management system - we explain both the management architecture (Section 3) and the management tools themselves (Section 4). Finally in the light of experience in the use of these tools we point to the future directions for work in this area (Section 5).

2. THE ICL GOLDRUSH MEGASERVER

The *GOLDRUSH Mega***SERVER** is a Database Server which runs commercial database back-ends including Ingres, Oracle and Informix. It holds the database, and services SQL queries sent by external clients. The architecture of a GOLDRUSH system is shown in Figure 1. It consists of a set of Processing Elements (PEs), Communications Elements (CEs) and Management Elements (MEs) connected together by a high performance network (DeltaNet) [Watson & Robinson 1990].

The PE is designed to run a Database Back-end. It consists of two SPARC RISC microprocessors, one of which runs the database server while the other is dedicated to delivering high performance message passing over the DeltaNet. A very large amount of RAM store (256 MBytes) is provided in the PE to enable large database caches to be configured. Each PE also has two SCSI-2 (wide and fast) connections, and 12 disks can be connected to each PE.

The Communications Element is identical to the Processing Element except that one of the SCSIs is replaced by two FDDI couplers for connection to Clients. Multiple CEs can be configured in a system for both performance and resilience reasons.

The Management Element is a mid-range UNIX processor which runs the management software. It also contains a "Teleservice" modem connection allowing: problem reports to be sent to a service desk; remote problem diagnosis from the service centre; and software problem fixes to be sent from the service centre to GOLDRUSH.

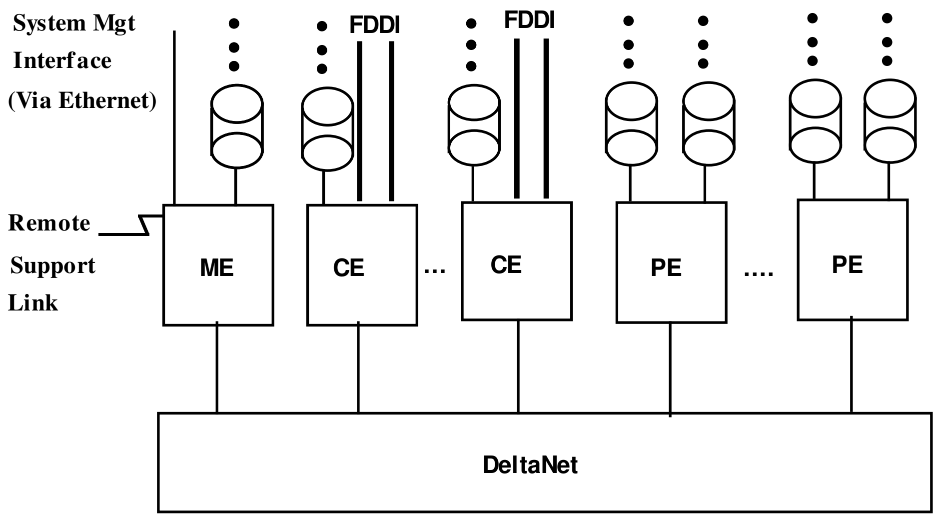

Figure 1 The GOLDRUSH Hardware Architecture

The DeltaNet is a high performance Network built out of 8x8 router chips. 128 Byte Messages are sent through the DeltaNet between Elements over full duplex links delivering up to 25 MBytes per second each way per Element.

Each Processing and Communications Element runs a Chorus micro-kernel based SVR4 UNIX Operating System. On top of this run a set of distributed services which are available to all applications including a Distributed Lock Manager and Distributed Filesystem. It is this software platform which supports the database servers.

There are many levels of resilience built into the system: the failure of an Element does not prevent a database service from continuing to run on the surviving Elements; and the failure of a disk does not prevent the data on it from being accessed by the database server (because all data is Plexed on more than one disk).

3. GOLDRUSH MANAGEMENT ARCHITECTURE

In this section we introduce the architecture of GOLDRUSH System Management and describe how it meets the requirements of managing a system which is both parallel and business-critical.

Parallelism adds an extra dimension of complexity to the management of a system. In GOLDRUSH, each Element could be managed separately using conventional system management tools, but this would be very time consuming and error prone. What is required are specific tools for managing parallel systems which hide the parallelism from the user wherever possible, so providing a single management image of the system to the user. They should allow, for example, groups of Elements to be configured simultaneously, and the aggregate performance of a database service running on a group of Elements to be monitored.

The key concept in the management of GOLDRUSH for achieving this are named sets of components. Users can define sets of Elements and then use the names of these sets in the system management applications, for example to monitor and administer the components of the set.

For example, if there is a 32 Element GOLDRUSH machine with two database services running on it: Service A running on Elements 0 to 10 and Service B on Elements 11-29. Two sets can be defined: ServiceA containing elements 0,1,..10 and ServiceB containing Elements 11,12,..29. The user can then refer to these set names when using the management tools described below to manage the system. For example a new software package can be added to all the Elements of ServiceA in a single command, or the aggregate performance of ServiceB can be easily displayed.

Similarly, sets of disks and volumes can be defined and managed. For example the database tables used by ServiceA may be striped across one disk connected to each processor in order to give high aggregate throughput. By defining a named set containing these disks, the configuration software can create a partition on each disk from a single user action and the performance management software can be instructed to monitor the aggregate performance of the disks without the user having to identify each disk individually.

The concept of sets is also key to resilience and tuning: if one Element in a set running a database service fails then another can be automatically added to the set. Because the management applications refer to the name of the set, and not the Elements in it, any change in the set contents is isolated from the administrator.

The architecture of System Management is based around this idea of sets and is shown in Figure 2. Each Element runs an agent which offers local system management functionality, for example running commands and collecting statistics. The Management Element (ME) contains a layer of software which distributes management requests to the agents on sets of Elements. The agents return results which are filtered and aggregated. Filtering is used to prevent unnecessary duplication. For example, if an element fails then it is possible that all other elements will notice (because they will be unable to communicate with it) and so it is possible that tens of messages to this effect will be sent from the Elements to the ME. It is important that these are not all passed to the user, and so all but one are filtered by this layer of software. Aggregation is used to provide a single management image of the system where it is appropriate; for example, if a service is running on a large number of elements then the user may wish to discover the aggregate performance of the service, and this layer achieves this by combining the performance measures from the set of elements.

Sometimes it is necessary to individually manage an Element, for example to explore why it is doing less work than other Elements in a set. Therefore the management applications all support the concept of mining: starting with the aggregate view of the management of a set of elements, but allowing the user to move down a level to monitor or control an individual component such as an Element or Disk.

These set-based mechanisms allow the Management Element (ME) to offer to Management applications a single, high level interface for managing the system. The applications themselves run on the ME but are controlled from a PC (the System Management Workstation) or an external, possibly Enterprise-wide, Management Server.

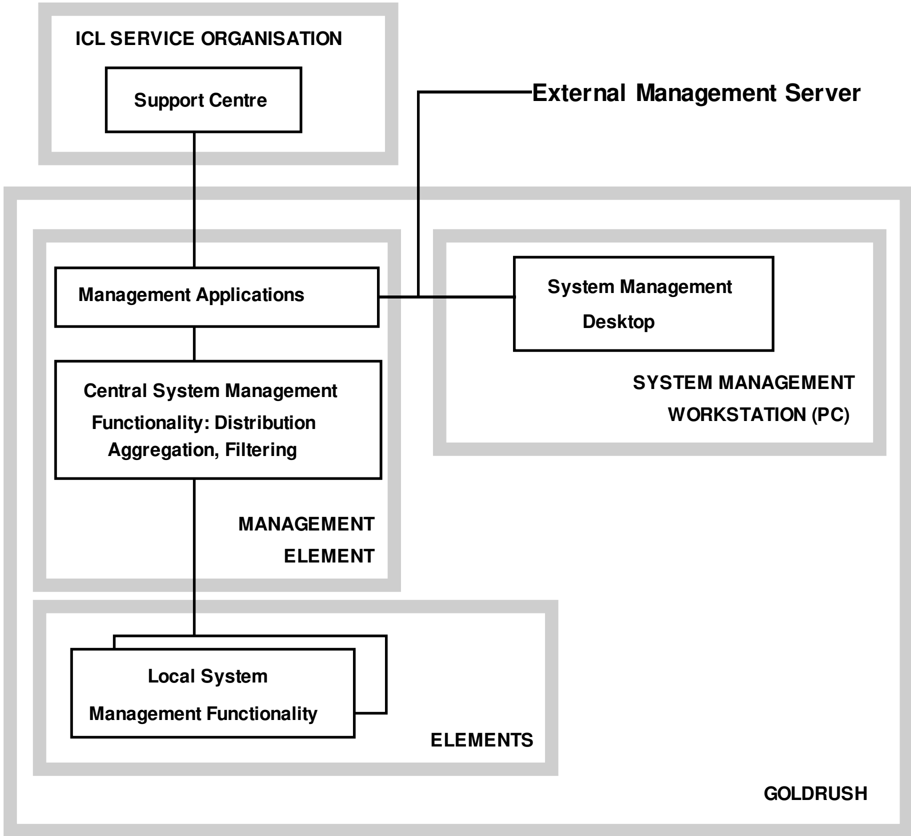

Figure 2 GOLDRUSH Management Architecture

An often overlooked aspect of providing a highly available commercial system is that the system management software must itself be resilient: failure of these tools, or the systems on which they run may otherwise compromise the availability of the system being managed. This is supported in the GOLDRUSH architecture by the ability to switch to a back-up ME if the primary fails.

4. GOLDRUSH MANAGEMENT TOOLS

The Management Applications are designed to offer comprehensive coverage of the key management functions required by a Database Server. The key management applications are:

- *Operations Management:* The aim of Operations Management is to support the day-to-day monitoring and control of the system.

An operator can monitor the current status of all the major hardware and software components (the *managed objects*) of the system through a single pictorial view. Each managed object is represented as an Icon whose colour represents the current status, and so changes of colour alert the operator to events in the system.

A limitation of most current management tools is that they concentrate only on the management of the system hardware. However this is only one aspect of the system, and the failure of the application or system software can cause the loss of service to the user. Therefore in GOLDRUSH all components are monitored. For example, when a database service is created, a managed object is automatically created for it, and the status of the service (starting, running, in error, stopping...) is represented by its Icon's colour.

It is also important that the operator is given warning of problems sufficiently early so that action can be taken to rectify them before they cause a system failure. For example, in GOLDRUSH the key filesystems are monitored and if they are becoming full then the operator is informed so that action can be taken before the system is affected.

Each managed object has a set of actions associated with it, so allowing the Operator to control it. These actions are selected from a menu. For example, the database service object has a set of actions associated with it to allow the Service to be started, stopped etc. These packaged actions remove the need from the operator to directly interact with the system, or edit configuration files. This reduces the risk of errors which may lead to system failures.

Finally, it is important that the management of the system should conform to the appropriate System Management standards. The most widespread for Operations Management today is the Simple Network Management Protocol (SNMP) [Stallings 1993] which offers a standard for:

- monitoring a remote system
- carrying out actions on it
- receiving asynchronous management messages from it (traps)

The growing demand for the SNMP standard is being aided by the availability of Network Management systems which conform to it: for example the HP OpenView product. These tools allow entire networks of systems to be monitored from a single management station, with visual alerts to problems. Many large computer users are using such systems to centralise their system management, so reducing costs and minimising system specific management differences. Because the SNMP protocol runs over networks, it is possible to centrally monitor systems spread over a wide geographic area. An SNMP agent is provided on GOLDRUSH, giving access to the managed objects described above.

- *Capacity Management:* All the major hardware and software components can be monitored through this application, both in real time and historically. These components include CPU, Disks, Memory, the Distributed Lock Manager, Filesystems and Database Servers. Named sets of components can be defined so allowing them to be monitored and analysed as a unit. The advantage of providing comprehensive monitoring of all levels of the system through a single interface is that it makes it possible to correlate related performance measures, such as transactions per second from the database server and processor utilization from the kernel on which it is running. This aids performance problem identification, tuning and trend analysis. The real-time application will graph meters selected by the user, and apply thresholds to them so that performance problems can be identified. The historic tool stores the values of meters for later analysis. It can collate the meters collected over days, weeks or months to provide aggregated information which is particularly helpful for identifying long term trends which may affect the system: for example the disk utilization may be slowly rising and will eventually reach the point where response times are affected. New forms of presentation tools have had to be designed to allow the user to understand the behaviour of a parallel machine. For example, to monitor the workload of 20 processors running a single database service, it is not sensible to plot 20 line graphs on a single set of axis, but plotting 20 separate line graphs does not allow their relative performance to be easily understood. We have therefore developed a tool - GoldWatch - which collects historic performance data and then, having thresholded it, plots it in the form of a contour map with time on the X axis and Element on the Y axis. The height of the map at any point represents the activity of that processor at that time. The peaks of the map therefore represent areas of high activity which may require investigation.

- *Configuration Management* : This maintains the database of sets (described above), and provides interfaces through which they can be accessed. It also provides a mechanism for configuring filestore. In a machine such as GOLDRUSH which may contain hundreds of disks, it is not feasible for the system administrator to configure them individually. This is particularly the case due to the added complexity of configuring the Striped and Plexed disks which are required by business critical database systems such as GOLDRUSH. Therefore in order to simplify the configuration of GOLDRUSH, the user can use a graphical user interface to design the filestore of one Element and then have it automatically replicated across a set of Elements. This replication can automatically obey to a set of rules; for example the requirement to have disk plexes held on the disks of separate PEs to avoid data loss on Element failure.

- *Problem Management* : Information on all problems observed within the system are passed to the Management Element where they are filtered and logged in a customer accessible database. If necessary they can be passed over a modem connected to the ME to the ICL service centre for action. This information may include problem evidence such as dumps. In the case of software faults, fixes can be passed back to the customer site.

- *Administration & Software Distribution* : This allows commands to be run on sets of Elements, and provides a tool for installing software packages on the Elements. The user specifies the package and the set of Elements. The tool then copies the package onto each element and ensures that it is successfully installed. If an Element is down when this is taking place, then the system records this fact and performs the installation when the element recovers.

5. CONCLUSIONS

We are evaluating methods to further assist the user in managing parallel, business-critical systems. One important area is the development of system models so the effect of proposed changes to the system can be predicted. For example, if the Capacity Management tool shows that the response time of a database is decreasing over time as new users are added, then the user may consider striping the database tables over more disks. By using models of the system, the user could observe the predicted effect of this configuration change. If the predicted performance was acceptable then scripts could be generated which would automatically reconfigure the system. These techniques reduce the risk that changes to the system could adversely affect the performance or availability.

The existing GOLDRUSH System Management tools described in this paper have been developed over the last 4 years and are currently in use in several production systems on customer sites. The central idea of managing sets of components, rather than individual Elements, Disks etc. appears to have been a success as the amount of effort required to manage a GOLDRUSH is roughly the same as that required to manage a powerful Uniprocessor database server.

6. REFERENCES

Stallings 1993: W. StallingsSNMP, SNMPv2 and CMIP. The Practical Guide to Network Management Standards, Addison-Wesley, 1993.

Watson & Catlow 1990: P. Watson & G.W.Catlow, The Architecture of the ICL *GOLDRUSH Mega***SERVER**, in. Advances in Databases, ed. C. Goble and J.A. Keane, Lecture Notes in Computer Science 940, Springer-Verlag, 1995.

Watson & Robinson 1990: P. Watson & E.H. Robinson, The Hardware Architecture of the ICL *GOLDRUSH Mega***SERVER**, in The ICL Technical Journal, November 1990.