The Architecture of the

ICL GOLDRUSH MegaSERVER

Paul Watson & George Catlow

ICL, Corporate Servers, West Gorton, Manchester M12 5DR, UK

Abstract. This paper discusses the requirements which are to be met by a parallel computer system if it is to satisfy the requirements of commercial database processing, and describes how one such system - the ICL GOLDRUSH MegaSERVER - has been designed to meet these requirements.

GOLDRUSH is a distributed store parallel processor consisting of up to 64 elements, each of which can co-operate in database processing, exploiting both the parallelism found within complex queries (intra-query parallelism) and that found between queries in On-Line Transaction Processing workloads (inter-query parallelism). The paper discusses the requirements of business critical database applications including high availability, integrity and manageability. It then details the architecture of GOLDRUSH in order to show how a commercially available system has been designed to meet these requirements; this includes resilience to failure of hardware components such as disks and processors, and the provision of system management applications which allow the parallel machine to be managed as a single system.

1 Introduction

Conventional mainframe computers are being pushed to, and beyond, their performance limits by the needs of today's commercial computing workloads. This, combined with the cost of these large systems, is forcing both customers and manufacturers to look at more radical architectural alternatives. Parallel computers have been used successfully for many years to overcome these same problems for scientific computing [ALA89], but it is only now that parallel machines are starting to be produced for the commercial market by a range of mainstream computer manufacturers. Potentially, this market could become far larger than the scientific market: 80% of the worlds large computer systems (those over $1M) are commercial rather than scientific [IDC92].

Presented at the 13th British National Conference on Databases, Manchester, 1995

As these new commercial parallel systems become available, the unprecedented levels of performance they offer will create new opportunities for businesses which utilize them. Already companies are re-engineering their business processes to fully exploit the new technology. One example of this is a financial organisation with a large number of customers, each of which has a number of different types of accounts. Currently, information on each account is held in a different database, on a different computer. This makes it impossible for the organisation to collate information about all the accounts held by a customer. They wish to do this so as to form an overall picture of the financial state of their customers, allowing them to target their services and marketing more effectively. The solution is to move to a parallel system: the higher levels of performance offered will allow them to unify all their information into a single database, on a single machine. The database can be structured to allow information to be accessed by both account number and customer.

The GOLDRUSH system evolved from earlier work carried out in collaborative projects with Universities and other Companies. The two most important of these projects were: the Alvey Flagship project [WAT88] which produced a distributed store parallel system running both Declarative Languages and Databases; and the ESPRIT EDS project [WAT91] which developed a parallel system for Parallel Databases, Language Translation and Declarative languages.

The rest of this paper focuses on the GOLDRUSH parallel system, and is structured as follows. The requirements of commercial parallel database systems are discussed in Section 2. Section 3 then describes the overall GOLDRUSH architecture, including the hardware. Section 4 details the platform software and 5 describes how databases are supported by it. Sections 6 and 7 give details of how two of the key requirements for commercial parallel computers- high availability and manageability- are met. Finally, section 8 contains some of the conclusions we have drawn from our work.

2 Requirements

The challenge for the designers of parallel machines is to give customers the key attributes they have traditionally found in conventional mainframes, but with greater power and lower cost-performance. Customers will not entrust their business-critical applications and data to a machine which does not offer high availability and manageability. The methods which have been adopted to meet these requirements in GOLDRUSH are described in sections 6 and 7.

Another attraction of parallel machines is their scalability; customers can grow their systems by adding units of processing power and IO capability as required. It is important that the design supports this by ensuring that there are no system bottle-necks which reduce scalability.

The need to run commercial relational database systems (RDBMS) places two major requirements on the platform software of the machine. Firstly, there is a need for a globally shared filestore which all processors in the system can access so that each can be running transactions against the same database tables. Secondly, a Global Lock Manager is required to preserve the integrity of shared database tables when multiple processors are performing transactions simultaneously. The design of these two components is described in Section 4.

It is interesting to note that as yet there is no requirement for distributed shared store: current commercial parallel database systems attain parallelism by running a separate server on each processor. They communicate only through the global lock manager, and the globally shared filestore, both of which can be efficiently implemented by distributed, message passing processes.

The implementation of the lock manager and filestore in GOLDRUSH is described in Section 4. The mapping of the database servers onto the system is described in more detail in Section 5.

3 Overall Architecture and Hardware

The GOLDRUSH MegaSERVER is a Database Server, designed to run commercial database back-ends. It stores the database, and services SQL queries sent by external clients such as PCs, workstations and mainframes. Examples of application architectures are shown in Figure 1.

The diagram shows options for Management Information Systems (MIS) generating complex queries, On Line Transaction Processing (OLTP) and Batch Clients. The use of Transaction Processing (TP) protocols in the application architecture is recommended for large systems: Clients connect over a local or wide area network to an Application server, which generates the SQL. This improves performance as TP sends less data over a wide area network than does SQL; it also offers better security and control of application code. Note that various existing systems can be connected to GOLDRUSH. The applications can remain on these systems, but the database migrates to GOLDRUSH.

The internal architecture of a GOLDRUSH system is shown in Figure 2. It consists of a set of Processing Elements (PEs), Communications Elements (CEs) and Management Elements (MEs) connected together by a high performance network (DeltaNet).

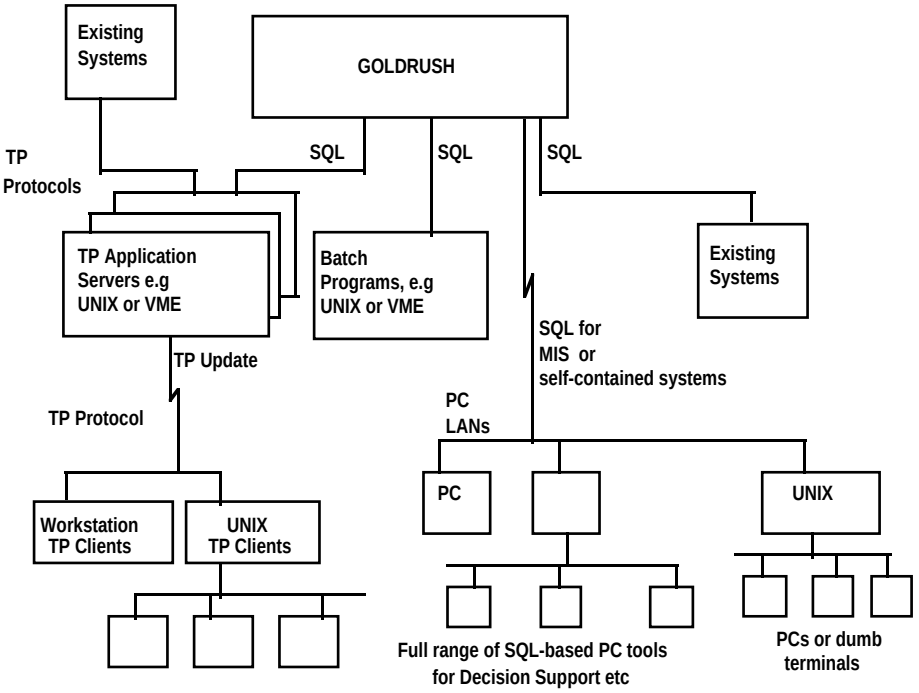

Fig 1. Example GOLDRUSH Application Architectures

The PE is the most common Element and 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 is provided in the PE to enable large database caches to be configured: this is very important for achieving high performance database processing. Each PE also has two SCSI-2 (wide and fast) connections, allowing up to 30 disks to be connected.

The Communications Element is identical to the Processing Element except that one of the SCSIs is replaced by two FDDI couplers for Client connection. FDDI was chosen because of its high performance (100Mb/s), and the availability of bridges allowing connection to all other common types of LAN and WAN, such as Ethernet. Multiple CEs can be configured in a system for both performance and resilience reasons (see Section 6).

The Management Element is a conventional mid-range UNIX processor (currently an ICL DRS6000) which runs the Management Software (which is described in Section 7). 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.

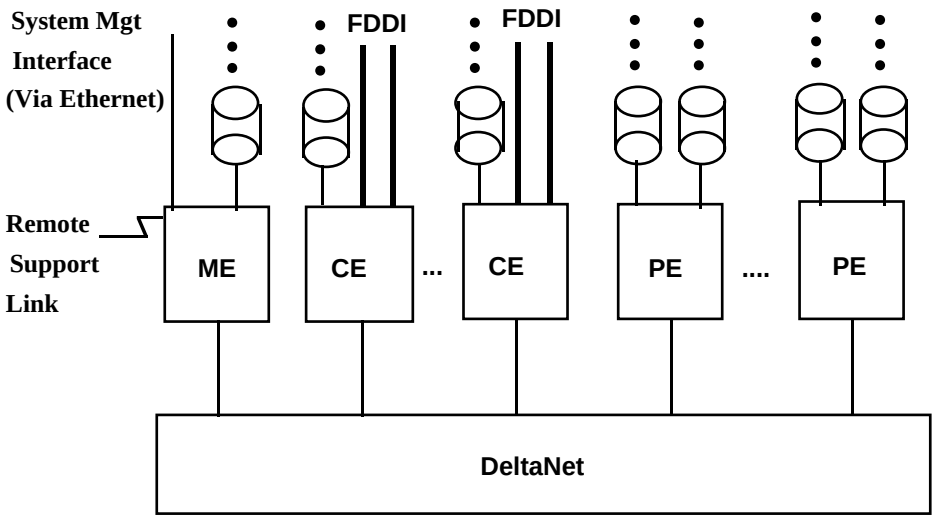

Fig 2. The Internal GOLDRUSH Architecture

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

For high performance archiving, tape libraries can be attached to the Elements via their SCSI connectors. Multiple Element connections provide high performance archiving by allowing multiple data streams to be archived simultaneously to multiple tape drives.

GOLDRUSH systems can contain up to 64 Elements. Each PE and CE has two 90MHz HyperSPARC processors, with 256MBytes of RAM. PEs can each connect to up to 50GBytes of Disk Store. Because of the reliance on industry standard, commodity components, this specification will be continuously upgraded as new versions of the components become available. These upgrades will include faster processors and larger capacity disks. The first release offers the Ingres, Oracle, Adabas and Informix databases.

4 Platform Software

Figure 3 shows the GOLDRUSH platform software architecture. Each Processing and Communications Element runs a Chorus micro-kernel based SVR4 UNIX Operating System. This has been enhanced to support the requirements of the parallel database processing by the addition of the following sub-systems:

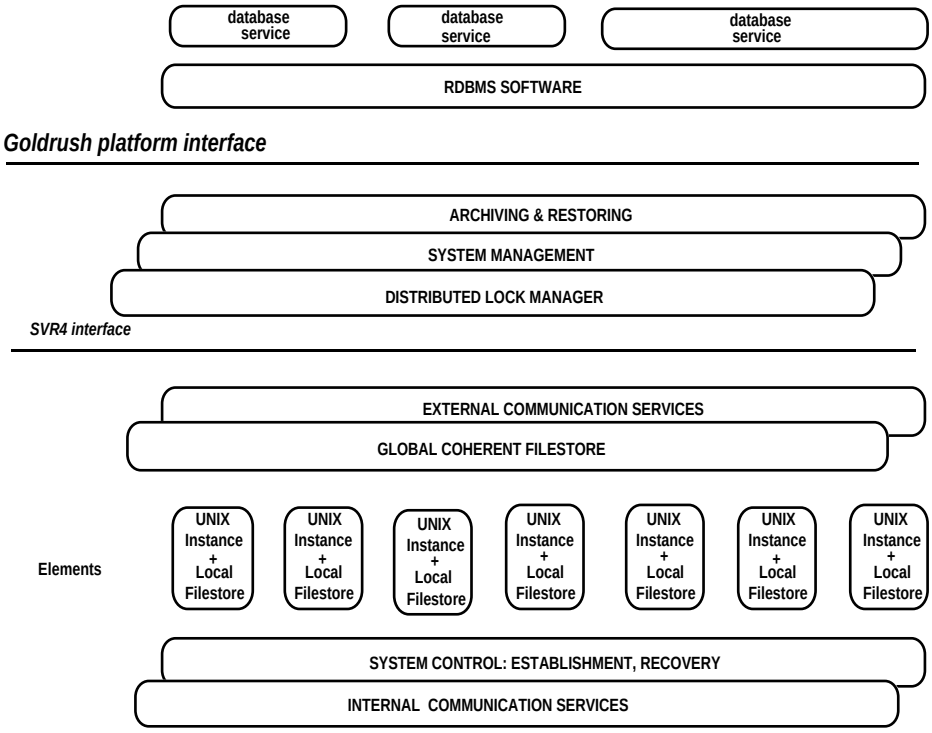

Fig 3. GOLDRUSH Platform Software Architecture

- *Internal Communication Services* : As described in the previous section, the DeltaNet hardware offers fast communication between Elements. In a parallel machine such as GOLDRUSH which relies entirely on message passing for inter-PE communications it is very important that low latency, high throughput message passing is available to software. While conventional transport protocols such as TCP/IP and ISO transport are available for inter-element communications, they add an unacceptably large overhead on top of the basic hardware performance of the DeltaNet. In order to overcome this, a very lightweight communications protocol has been designed. It is made available through UNIX interfaces so that application level software can exploit it; for example database servers can use it for communicating fragments of transactions when exploiting intra-query parallelism. Internal kernel interfaces are also provided, and these are used by a number of the platform sub-systems, for example for remote filestore access and distributed deadlock detection. The bulk of the lightweight comms stack runs on the SPARC processor dedicated to driving the DeltaNet (see Section 3), so minimizing the performance impacts on the other SPARC processor which runs the database server.

- *System Control* : As was described above, each Element runs a separate instance of UNIX SVR4. It is a requirement that it should not be necessary to manage each UNIX instance separately. Therefore, it is the job of the system control layer to support this: it is responsible for establishing and shutting down the Elements in a co-ordinated fashion. The standard Network Information System (NIS) is used to provide a global method of configuring all the Elements, including users and communications connections. If Disks or Elements fail then this layer ensures that the rest of the system can continue and that applications remain running. This is discussed in detail in Section 6.

- *Global Coherent Filestore* : As described earlier, each Processing Element (PE) runs a separate database server. However, they are all able to perform transactions on the same database simultaneously. This requires that each PE can access all the database tables which are held on disk (details of the mapping of tables onto disks is given in section 5). As was described in Section 3, each Processing Element contains two SCSIs for disk connection. Therefore, physically each disk is attached to one Element only. Some of the space on the disks is dedicated to local filestore which is only required to be accessible by that Element: for example the local UNIX kernel. However, the rest of the disk space is used to hold database data (tables, logs etc.) which must be globally accessible by other Elements if parallel database processing is to be supported. Providing this connectivity is the role of the Global Coherent Filestore (GCF). Each Element creates its own portion of this Global Filestore on its own local disks, and then cross-mounts the global filestore from all other Elements, so making it accessible to the database server running on it. In this, the GCF is very similar to the standard UNIX Network File System (NFS), however there are two important differences. Firstly, the GCF has been engineered to provide very high performance remote filestore access. To achieve this it is implemented in the kernel, and exploits the lightweight communications described above. Secondly, unlike NFS, the filestore is completely coherent. This ensures that if one Element updates part of the database, the updated value is seen by all other Elements which subsequently access that part of the database. This is achieved by implementing only server-side filesystem caching (NFS also has client-side caching). The lack of client-side caching does not reduce performance as the database servers themselves manage their own client-side cache in main store on each Element (which is why the large main stores are required).

For efficient database support, both Asynchronous and Raw IO are supported by the GCF, along with conventional UNIX file access. For high availability, the Veritas VxVM Volume Manager is used to allow all data to be mirrored, while for fast recovery after Element failure the Veritas VxFS Filesystem is used. Their use is described in more detail in Section 6.

- *Global Communication Services* : External Communications between GOLDRUSH and external Clients generating SQL is via the FDDI couplers on the CEs. Both TCP/IP and ISO transport protocols are supported.

It was an important criteria for manageability that GOLDRUSH appeared as a single system to Clients, rather than as a set of individual Elements. The Global Communications Services layer provides this in the following way. For ISO communications the GOLDRUSH machine has a single externally visible address independent of the number of CEs and PEs. For TCP/IP communications each FDDI coupler has a separate address, independent of the number of PEs. PEs wishing to set up communications with Clients "listen" for connections to the service they are offering, and when an external Client attempts to connect to the service, the CE which receives the request routes it to a PE offering the service. If multiple PEs are offering the service then the CE will load-balance the connection requests across those PEs. Once a connection between an external Client and a PE is established, the CE relays the data packets from the FDDI coupler to the PE (and vica versa) using the lightweight communications protocol for efficiency. The Global Communications Services layer can maintain connections in the presence of FDDI coupler, CE or (in some cases) external LAN failure. This is described in Section 6.

- *Distributed Lock Manager:* On GOLDRUSH, database management servers run in parallel on many PEs. It is the job of the Distributed Lock Manager to ensure that they run transactions consistently, avoiding concurrent, incompatible updates to the databases. On a uniprocessor, this job is done by the database server's own lock manager which receives requests to take and drop locks and forces transactions to wait until they have the correct lock before data can be accessed. The Distributed Lock Manager (DLM) must act as a Global Lock Manager for all the database servers running against the same database. It is distributed across all Elements, with an instance running on each Element, so that the processing and communications load is shared, and no one Element becomes a bottle-neck. Each lock is managed by one instance of the DLM and all requests for it are sent to that instance by the PE generating the request. The DLM also provides Global Deadlock detection using a novel algorithm to minimize inter-PE communication [HIL89]. For efficiency, all communication with the DLM is by the lightweight communications protocol. The DLM is also resilient to the failure of any process or Element (see Section 6).

- *System Management:* Section 7 describes this layer.

- *Archiving & Restoring:* One of the key attributes found in mainframes but generally not available in UNIX based solutions is fast archive and restore. This is important as the time taken by archiving may reduce the availability of the system for database processing. Also the time taken to restore from archive will limit the recovery time after a major failure. The latter can be a particular problem as a major failure, unlike archiving, cannot be planned in advance.

The large disk capacity of systems such as GOLDRUSH requires very high throughput rates. Therefore for very high performance, this layer supports the integral, SCSI connected archiving described in Section 3, ensuring that data is archived as fast as the available tape drives can accept it. For lower performance archiving, which may for example be acceptable for small databases, the layer will support archiving to external archive servers connected via the FDDI connections.

5 Databases

The initial release of GOLDRUSH supports the Oracle, Ingres, Adabas and Informix databases. There are two basic mappings onto the GOLDRUSH Architecture. These are denoted as the Shared Access and Distributed Access mappings and are described in this section.

The Shared Access mapping is shown in Figure 4.

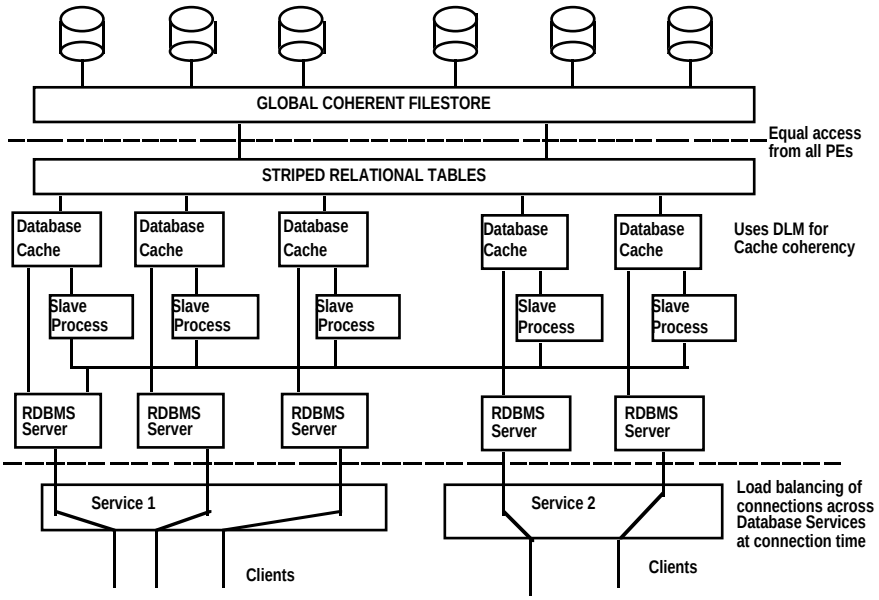

Fig 4. Shared Access Mapping of Databases onto GOLDRUSH

In this mapping, each PE runs its own Database Server. The database tables are striped across disks attached to a number of processing Elements; the Global Coherent Filestore allows all Elements to access all fragments of the tables. Each PE has a local database cache in main store, and the Distributed Lock Manager is used to ensure consistency. Figure 4 shows Service 1 running on 3 PEs, and Service 2 running on 2 PEs. When a Client connects to a database service, the CE uses a load balancing algorithm to decide which of the PEs offering that service should take the connection. Once the connection is made it remains until the Client disconnects; during that time it services all transactions originating from that Client. Where intra-query parallelism is exploited, slave processes in each PE are used to distribute and manage the execution of the query across a set of PEs.

Figure 5 shows the Distributed Access mapping of database servers onto GOLDRUSH.

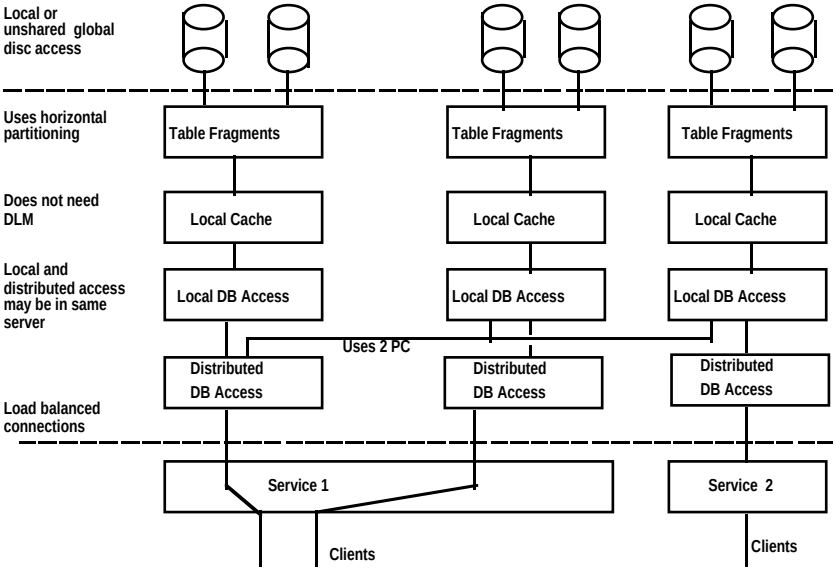

Fig 5. Distributed Access Mapping of Databases onto GOLDRUSH

In this mapping, again, each PE runs its own Database Server and the database tables are horizontally fragmented across PEs. As in the shared access mapping, a Client connects to a PE and sends queries to it. However, each query is decomposed into sub-queries which are distributed such that each PE only has to access the table fragments which are stored on its local disks. The results of these sub-queries are then combined and the result returned to the Client. The main effect of this type of mapping is that there is no use of the Distributed Lock Manager: only one PE (the local PE) can access a particular table fragment and so local locking can be used. However, if a query requires table fragments on more than one PE to be updated then some form of co-ordination such as Two Phase Commit (shown as 2PC in the Figure) is required.

In both the mappings, it is possible for both OLTP and Complex queries to be running against the same database simultaneously. In order to reduce the performance effects of one on the other, different sets of Elements can be used for the two different types of queries- Figures 4 and 5 both show two different services (Service 1 and Service 2) running on seperate sets of elements but against the same database. It is also important that database tables are fragmented across sufficient numbers of disk to ensure that the IO throughput requirements of each type of query are met.

6 High Availability

The GOLDRUSH MegaSERVER is constructed from tens of commodity components such as processors and disks. Machines constructed in this way do not inherently offer the levels of reliability required to meet customers requirements: in a simple parallel machine design, the failure of any one component may bring down the whole machine and halt the customer's application. However, careful design can exploit the parallelism to produce a system in which if one component fails then another takes its place. This may reduce the performance once the failure has occurred, but it will not halt the customer's application.

In this section we discuss how the parallelism of GOLDRUSH is exploited to provide high-availability. The main areas of resilience are:

- *Filesystems:* For high availability, all database data is mirrored; the Veritas Volume Manager VxVM is used for this. Recovery from Disk failure is totally transparent to the database servers accessing the data: when a disk fails, processing continues from the surviving mirror; the data is automatically re-mirrored onto one of the spare disks which are kept in the system for this purpose, and sometime later the failed disk can be replaced (using hot pull and push).

In order to be resilient to Element failures, each mirror is placed on a disk connected to a different Element. Therefore, although when an Element fails the disks locally connected to it become unreachable, all the data on those disks is still available on their mirrors (because they are connected to other Elements).

The use of Veritas VxFS as the UNIX filesystem allows a filesystem which was owned by a failed PE to be very quickly reconstituted on another PE (by considerably reducing the time needed to restore filesystem consistency).

- *Processing Elements:* Each Element in the system runs its own instance of UNIX. This minimizes the dependencies between Elements and ensures that if an Element does fail then others can continue. Any parts of the Global Coherent Filestore which were owned by that Element are logically moved to be owned by another Element as described above. The database transactions being run on the failed Element are automatically recovered by the database software running on another Element using information found in tables, logs and journals. Any Clients connected to a failing PE have to reconnect to the GOLDRUSH server. Those connected to other PEs see a short delay (up to 2 seconds) while the filestore is re-organised, but then processing continues.

- *Communications Element:* If a CE (or FDDI coupler) fails, then SQL connections from a Client into GOLDRUSH will be automatically routed through another CE (or Coupler), provided that there is another route from GOLDRUSH to the Client (and that the Client provides a full implementation of the transport connection protocols). This is achieved by maintaining configuration tables holding information on external sub-networks. Internal polling ensures that the failure of a CE or coupler is quickly detected. Following this, information in the configuration files is used to cause messages to be re-routed through another coupler, if this offers an alternative route to the Client.

- *Fans and Power Supplies:* Spare fan and power supply capacity is present in the GOLDRUSH cabinets so that if one fails then the system can continue running. Uninterruptable Power Supplies can be used to prevent problems due to electricity supply failure.

- *Distributed Lock Manager* : All lock information is mirrored in more than one PE so that the system is unaffected by Element or process failure.

7 System Management

System Management is a vitally important but often overlooked attribute required by Commercial parallel processors. Experience in the computer industry over recent years suggests that companies who downsize from a mainframe to a number of conventional UNIX systems often find that they make no overall savings in their IT budgets because even though the raw hardware and software costs may be lower, the cost of management increases dramatically. This is due to the fact that management tools for UNIX systems are generally more primitive than those available on mainframes, and because there is a need to manage a number of systems rather than just one. The danger therefore is that a parallel processor containing many UNIX Elements will prove to be extremely difficult and therefore costly to manage.

GOLDRUSH management tools have been designed to allow it to be managed as a single system, rather than as a collection of many UNIX systems. The basic architecture of System Management is shown in Figure 6.

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 the Elements. The agents return results which are filtered and aggregated. Therefore the Management Element (ME) can offer to Management applications a single, high level interface for managing Elements. 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. The Management Applications are designed to offer comprehensive coverage of the key management functions required by a Database Server.

A key concept in the management of GOLDRUSH is the use of 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. Similarly, sets of disks and volumes can be defined and managed. The concept of sets is also key to resilience and tuning: if one Element in a set running a database service fails 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, then the change in the set contents is isolated from the manager.

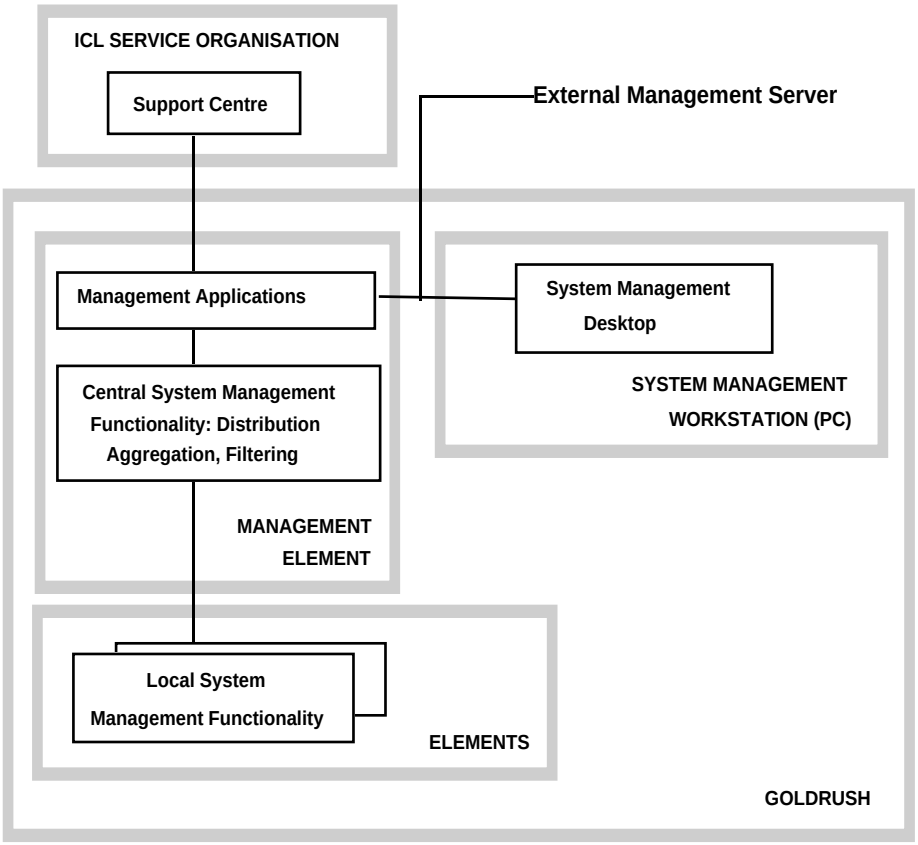

Fig 6. GOLDRUSH Management Architecture

The key management applications designed to provide single-image management of the parallel machine are:

- *Operations Management:* This allows an operator to monitor the current status of all the major hardware and software components (the *managed objects*) of the system through a single pictorial representation. Each managed object is represented as an Icon whose colour represents the current status, and so changes of colour alert the operator of events in the system. Each managed object also has a set of actions associated with it, so allowing the Operator to control it. 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. The object also has a set of actions associated with it to allow the Service to be started, stopped etc.

- *Capacity Management:* All the major hardware and software components can be monitored through this application, both in real time and historically. These components include Disks, Processors, the DLM, Filesystems, Database Servers. 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 both performance problem identification, tuning and trend analysis.

- *Configuration Management* : This maintains the database of sets (described above), and provides interfaces through which they can be accessed. It also provides a graphical interface 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 one at a time, as in standard UNIX systems. This is particularly true when the complexity of configuring mirroring is added. Therefore in order to simplify the configuration, the system manager can set up the filestore of one Element and have it automatically replicated across a set of Elements. This replication will automatically take into account the need to have disk mirrors held on separate PEs (see section 6).

- *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* : This allows commands to be run on sets of Elements, and provides a tool for installing software packages on a set of Elements.

8 Conclusions

This paper has described the design of a commercial parallel computer system for running business critical database applications. The requirements for such systems are evolving as customers begin to take advantage of the new machines now available in the marketplace. We envisage a positive feedback effect in which these machines will open up new opportunities to users, who will then exploit them and in so doing place new requirements on the machines themselves. In particular, we expect that customers' realisations of the business advantages they can gain through the use of complex queries to analyse their corporate information will lead to increasing performance demands. Similarly, the moves towards multi-media databases, and the advantages of centralizing information in these servers, so making it all available for analysis, will increase the data capacity requirements.

GOLDRUSH is designed to meet these evolving requirements due to its scalable parallel architecture, and its utilization of commodity components whose capacity and performance are continually improving.

9 Acknowledgements

The GOLDRUSH MegaSERVER is the result of the work of a large number of people at ICL, Corporate Servers Division. We would also like to acknowledge our debt to our partners in the Alvey Flagship and ESPRIT EDS projects.

References

ALA89 *Highly Parallel Computing,* G.S. Alamasi & A.J. Gotlieb, Benjamin/Cummings, 1989

HIL89 *Distributed Deadlock Detection: Algorithms and Proofs,* S. Hilditch & C.M. Thomson, Dept. of Computer Science, University of Manchester, Technical Report Series UMCS-89-6-1, 1989

IDC92 Source: International Data Corporation, 1992

WAT88 *Flagship: A Parallel Architecture for Declarative Programming,* I. Watson, V. Woods, P. Watson, R. Banach, M. Greenberg & J. Sargeant. in Proceedings of the 15th Annual International Symposium on Computer Architecture, Honolulu, Hawaii, May 1988.

WAT91 *The EDS Parallel Relational Datbase System,* P. Watson & P. Townsend, in Parallel Database Systems, ed. P. America, Lecture Notes in Computer Science 503, Springer-Verlag 1991.