Server Architectures: DBMS and Server Architectures

January 2005

René J. Chevance

Foreword

- **This presentation is a part of a set of presentations about server architectures.**
  Presentations are based upon the following book:

  Serveurs Architectures: Multiprocessors, Clusters, Parallel Systems, Web Servers, Storage Solutions
  René J. Chevance
  Digital Press December 2004 ISBN 1-55558-333-4
  http://books.elsevier.com/

  This book has been derived from the following one:

  Serveurs multiprocesseurs, clusters et architectures parallèles
  René J. Chevance
  Eyrolles Avril 2000 ISBN 2-212-09114-1
  http://www.eyrolles.com/

  The English version integrates a lot of updates as well as a new chapter on Storage Solutions.

Contact: www.chevance.com **rjc@chevance.com**

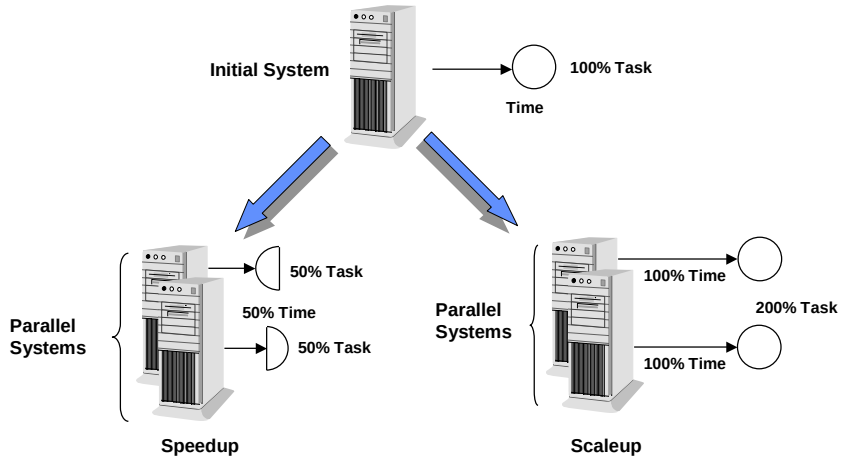
Organization of the Presentations

- Introduction
- Processors and Memories
- Input/Output
- Evolution of Software Technologies
- Symmetric Multi-Processors
- Cluster and Massively Parallel Machines
- Data Storage
- System Performance and Estimation Techniques
- ➔ DBMS and Server Architectures (this presentation)
  - Introduction
  - Architecture Models
  - Problems of Parallel DBMS
  - Architecture of Parallel DBMS
  - Data Partitioning
  - IBM DB2 UDB Enterprise
  - Oracle Real Application Cluster
  - Summary: DBMS Architecture Comparisons
- High Availability Systems
- Selection Criteria and Total Cost of Possession
- Conclusion and Prospects

introduction

- **Searching for performance through the exploitation in parallel of system resources**
- **An ideal parallel system must have two properties (DeWitt/Gray [DEW92]):**
  - **Linear Speedup**
    - **N times more resources make it possible to treat a given task in N times less than the reference system (typical case of DSS)**
  - **Linear Scaleup**
    - **N times more resources make it possible to deal with an N times larger problem in the same time as the reference system (typical case of OLTP: updating a N times larger database by a N times larger number of users)**

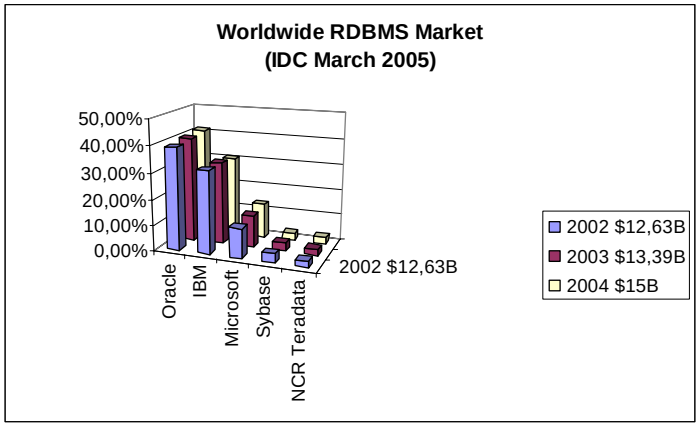
Speedup and Scaleup



Note: For most OLTP applications, the necessary synchronization makes impossible to obtain linear Scaleup or Speedup

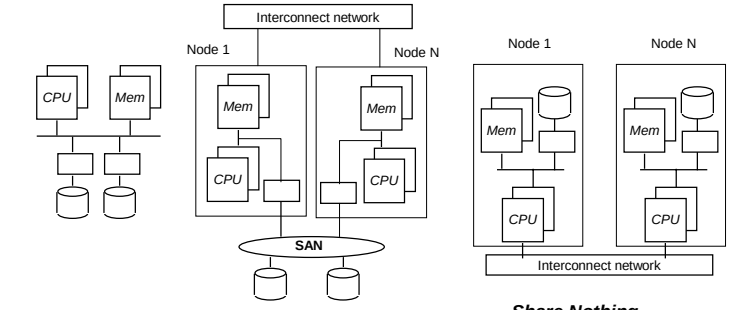
A Quick Look at RDBMS Market

- **Worldwide RDBMS Market Shares**
  (Source IDC March 2005 – 2004 figures are preliminary)



Architecture Models

- **Architecture options in system-disk interconnect**



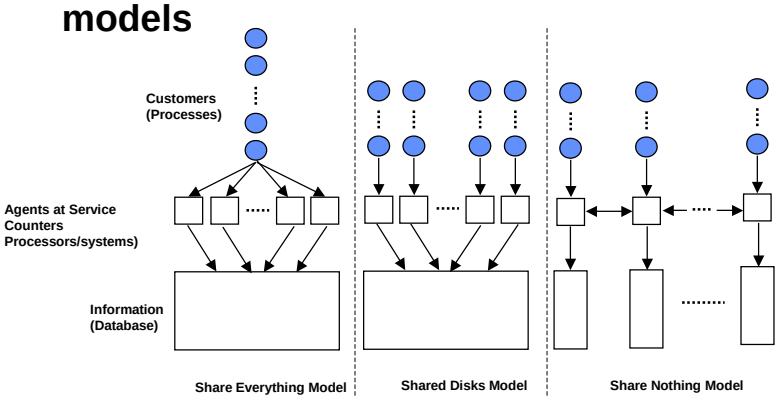
Share Everything (typical SMP)

Shared Disk (some clusters)

Share Nothing (some clusters and typical MPP)

Architecture Models(2)

- **Illustration of the differences between the models**



- **Shared Disks relies on the assumption that the node-to-disk interconnect is fast enough to avoid the need to distribute data**
- **Share Nothing relies on the assumption that the data is distributed so as to optimize I/O bandwidth and data handling capacity.**

Architecture Models(3)

- **Compared Characteristics of the various models**

	Share Everything	Shared Disks	Share Nothing
Advantages	Simplicity for both inter-request and intra-request parallelism Good use of resources Natural load-balancing Efficient interprocess com munication through coher ent shared memory Solution rapidly becoming commoditized at the low end	Good availability Good scalability (100 or more processors) Low cost because of re-use of standard components Good load balancing (data which is heavily shared can be replicated)	Good availability Extremely good scalability (several hundred proces sors) Low cost because of re-use of standard components
Disadvantages	Difficult to ensure system availability Limited scalability (upper limit of a few tens of pro cessors)	Interaction between nodes needed to synchronize data updates Saturation of the intercon nect network because of node-disk traffic Cost of maintaining coher ence across multiple copies of the data (if replication is used) especially if there are frequent updates.	Difficulties in load-balanc ing Difficult to administer and optimize because of data partitioning System performance strongly dependent on inter connect characteristics Cost of parallelizing requests, even for simple requests Cost of maintaining coher ence across multiple copies of the data (if replication is used) especially if there are frequent updates

Architecture Models(4)

- **Architectural choices supported by the principal DBMSes**

Model of architecture	Share Everything (SMP)	Shared Disks (some clusters)	Share Nothing(some clusters and MPP)
DBMS	IBM DB2 Informix Microsoft SQL Server Oracle Teradata (NCR) Sybase	IBM DB2 for OS/390 and z/OS Oracle Real Application Cluster	IBM DB2 for Linux, Unix and Windows Informix Microsoft SQL Server Teradata (NCR)

Note: Informix has been bought by IBM in 2001

Architecture Models(5)

- **Additional comments**
  - **In Share Nothing, alternate connections must be provided so that other nodes can take over in case of node failure. Such connections are not used in the course of normal operation**
  - ***Shared Disk*s and *Shared Nothing* architectures depend on both the architecture of the node-disk interconnect and the operating system. Some operating systems simply do not support the concept of *Shared Disks*. If the OS imposes an undesired model, it is possible to layer the desired architecture on top by partitioning (to convert *Shared Disks* to *Shared Nothing*) or by using Remote I/O (to convert *Shared Nothing* to *Shared Disk*s), although the latter brings substantial inefficiencies**
  - **Given the introduction of SAN storage networks, the distinction between Shared Nothing and Shared Disks becomes strictly functional, since every node connected to the SAN potentially has access to all the storage resources on the network. Thus, any distinction is made at the level of the OS**

Problems of Parallel DBMS

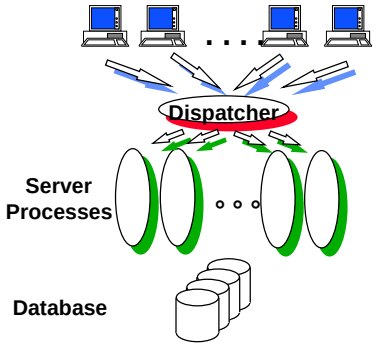
- **Parallel versus Distributed DBMS**
  - A parallel DBMS seeks to make maximum use of the resources of a system through the use of parallelism, while a distributed DBMS aims to make a collection of databases (whether homogeneous or not) supported by different systems look like a single coherent database.
  - Some DBMS versions said to be tuned for parallel architectures are simply distributed databases, in which multiples instances of the DBMS execute on multiple nodes and cooperate to provide the effect of a single database
- **There are two possible parallelization approaches [MOH94]:**
  - **Processing Parallelism.** A request is broken up into atomic requests which are executed in parallel
  - **Data Parallelism.** The data is partitioned into subsets, which are processed concurrently
- **Real life facts**:
  - Processing parallelism is limited by the number of operators involved in the processing of requests and the dependencies between operators
  - Data parallelism offers more possibilities (partitioning a relation into several sub-tables)
  - It is possible to combine both approaches through the parallel execution of requests against several sub-tables

Saturation Situations

- **There two typical cases of saturation of system resources by a DBMS**
  - **Saturation of processing resources.** This situation is called *CPU bound*, since system performance is limited by processors
  - **Saturation of Input/Output resources.** This situation is called *I/O bound*, since system performance is limited by I/Os
- *Taking into account the potential of technologies (i.e. CPU performance is growing more quickly than I/O performance), the « CPU bound » situation is preferable*

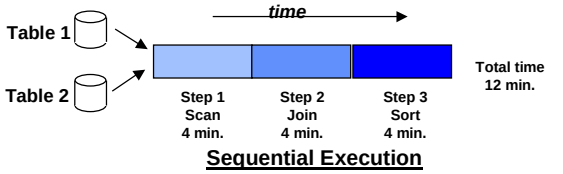
Architecture of Parallel DBMS

- **There are two possible form of parallelism in a server:**
  - **Inter-request parallelism**
  - **Intra-request parallelism**
- **Illustration of inter-request parallelism**

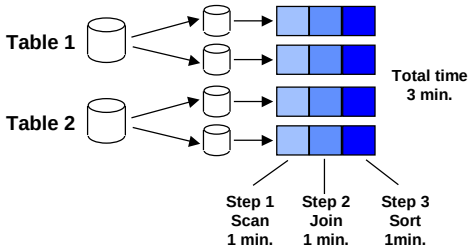


Architecture of Parallel DBMS(2)

- **Intra-request parallelism illustrating partitioning a database [RUD98]**



Sequential Execution

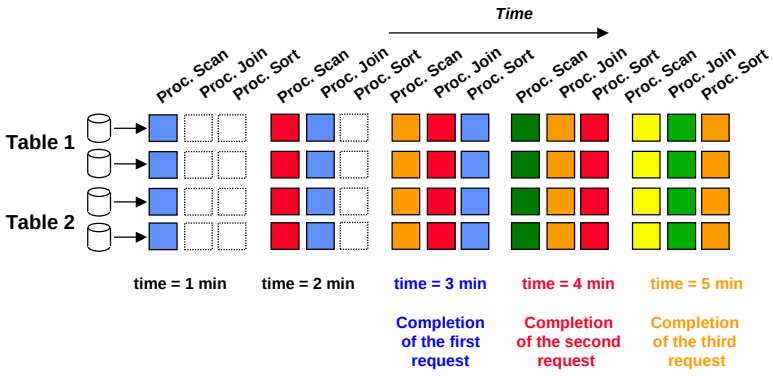


Data Partitioning

Partition the data across several systems (here, 4 systems)

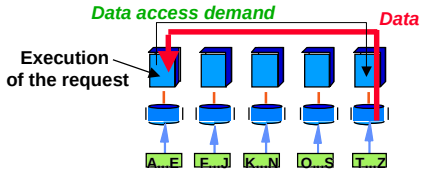
Architecture of Parallel DBMS(3)

- **Combining data partitioning and pipelined execution**

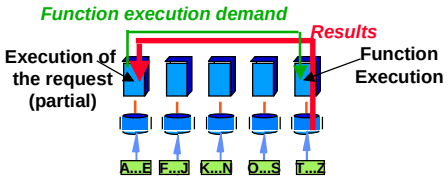


Architecture of Parallel DBMS(4)

- **Functional Models in a Share Nothing Architecture**



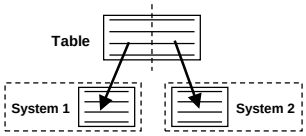
a) *Data Shipping*



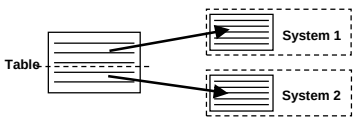
b) *Function Shipping*

Data Partitioning

- Implied by Share Nothing approach
- May be used in a Share Disk approach to improve the performance through judicious directing of requests to nodes (creating an affinity between node and data thus improving the effects of data caching)
- Two ways of data partitioning (based upon an attribute – a column of a tuple- called partitioning key):
  - Vertical partitioning
  - Horizontal partitioning



Vertical Partitioning

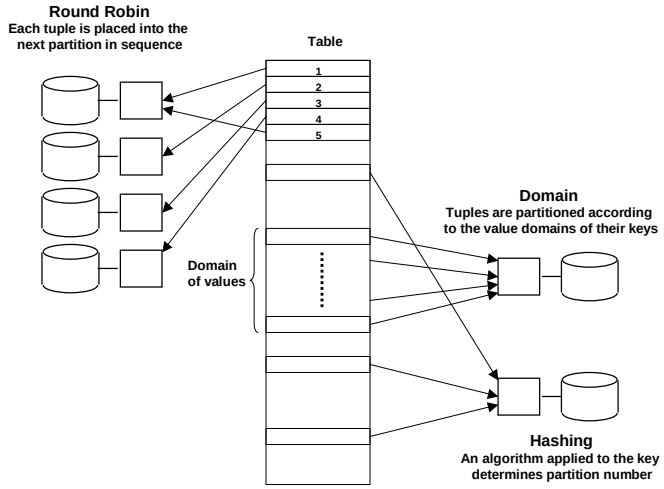


Horizontal Partitioning

  - Vertical partitioning can only be done with application design in mind
  - When a transaction updates data in more than one partition, the DBMS must implement a two-phase commit between its concerned instances

Data Partitioning(2)

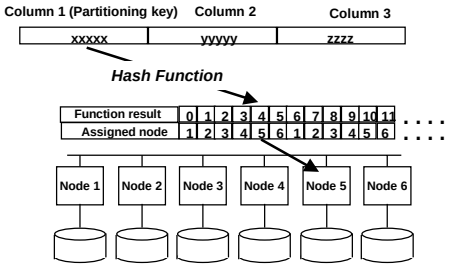
- **Examples of methods of data partitioning**



Page 19

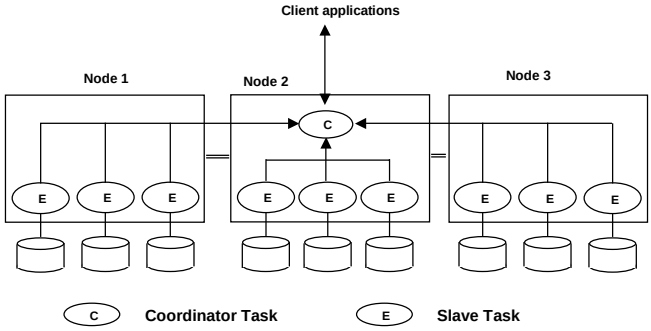
IBM DB2 UDB Enterprise Extended Edition

- **A version with the Data Partioning Option is intended for loosely coupled systems operating under Linux, Unix or Windows (there are equivalent versions for zOS and iSeries). This version (hereafter called DB2) is based on a share nothing approach and function shipping**
- **Data partitioning in DB2**

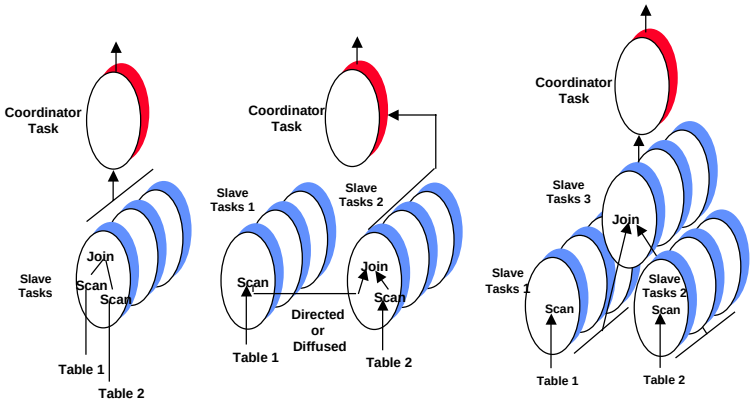


Page 20

Function Shipping in DB2



Join Strategies in DB2

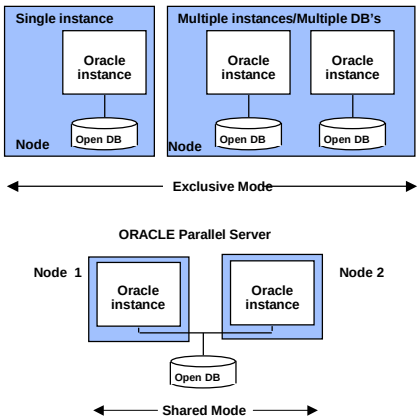


IBM DB2(4)

- **DB2 provides a number of facilities such as:**
  - **Administration tools**
  - **A single system image (as far as the database is concerned)**
  - **A database loading utility which can operate in parallel**
  - **Parallel backup and restore (in parallel with production use)**
  - **Ability to restructure the database concurrently with production use**
  - **High-availability functionality**

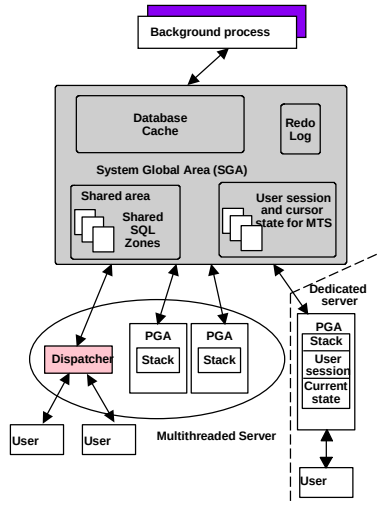
Oracle Real Application Cluster

- **Oracle Real Application Cluster (RAC for short) is based upon the Shared Disks philosophy**
- **Illustration of Oracle terminology**



Oracle Real Application Cluster(2)

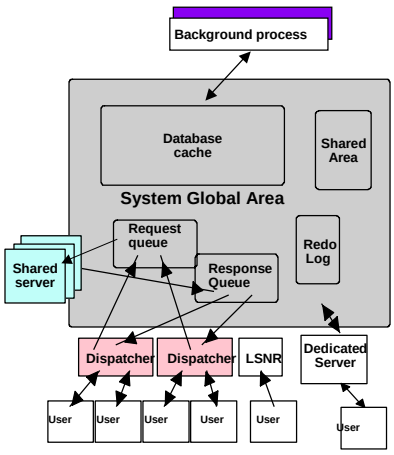
- **General Structure of an Oracle Instance**



Page 25

Oracle Real Application Cluster(3)

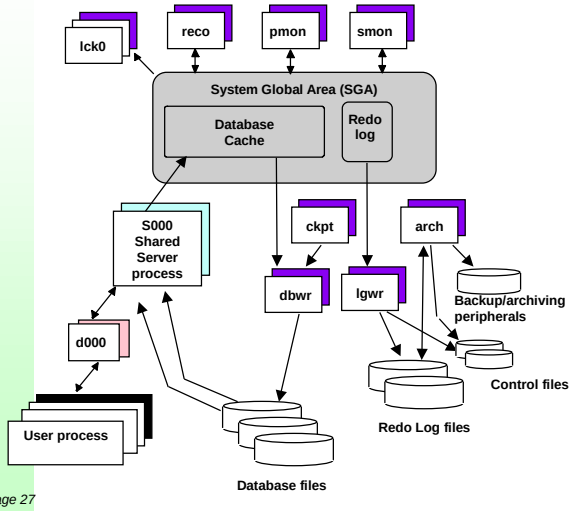
- **Structure of a Multithreaded Oracle Instance**



Page 26

Oracle Real Application Cluster(4)

Oracle Background Processes



The following pair of techniques are used to improve transaction processing performance:

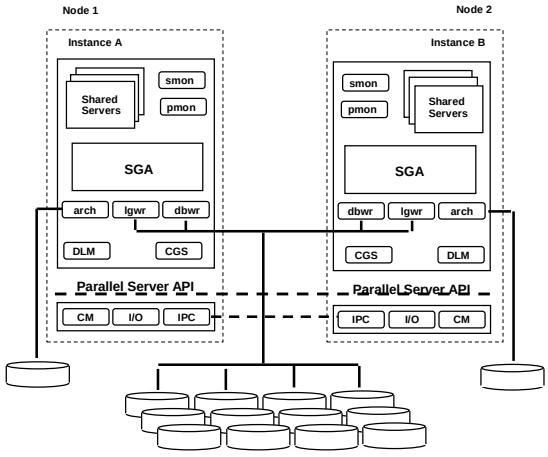
•• Fast Commit/Deferred Write. During a transaction commit, Oracle only updates the log. The database proper is updated only when modified blocks are written back to disk.

•• Group Commit. Log updates for several transactions are gathered up so that recording several commits requires just one log update. In other words, commit is delayed until enough have been encountered, or until a specified time quantum has expired; the log file is then updated with just one I/O operation.

Page 27

Oracle Real Application Cluster(5)

Architecture of RAC (Oracle 9i)



Management of the parallel environment is provided by the two principal components of RAC: PCM (*Parallel Cache Management*) and CGS (*Cluster Group Services*).
PCM uses DLM (*Distributed Lock Management*) to coordinate access to resources required by the instances; it is also used to coordinate access to shared resources such as the data dictionaries (metadata), journals and so on. CGS interworks with the Cluster Manager to supervise cluster state.

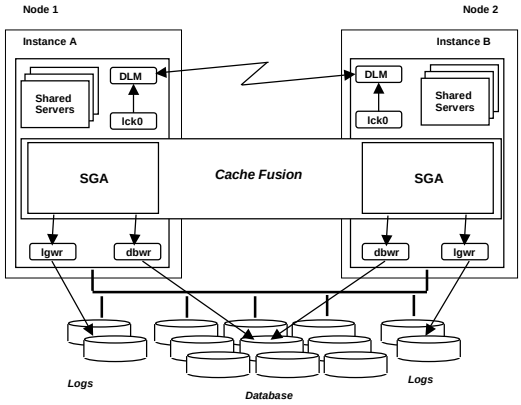
Page 28

Oracle Real Application Cluster(6)

- From a conceptual point of view, RAC implements a cache coherence protocol. The basic idea behind the protocol is that only one instance at a time can modify a storage block. Thus, a block being modified by one instance and needed by another instance will be communicated to the second instance . In RAC, the name *ping* is used to denote this operation.
- A lock protects one or more storage blocks. Intuitively, the more blocks protected by one lock, the more likely it is that a request for a block will require a ping. Since this operation means that all the modified blocks covered by the lock will be communicated to the requesting instance - and not just the wanted block - it is said to result in a *false ping*. Thus, a balance must be struck between the cost of managing many locks and the cost of having too many false pings.
- The optimization of RAC was based upon the idea of reducing the cost of managing cache coherency (the term cache synchronization is also used). This is the key objective of the Cache Fusion concept.

Oracle Real Application Cluster(7)

- **Cache Fusion Concept**



Oracle Real Application Cluster(8)

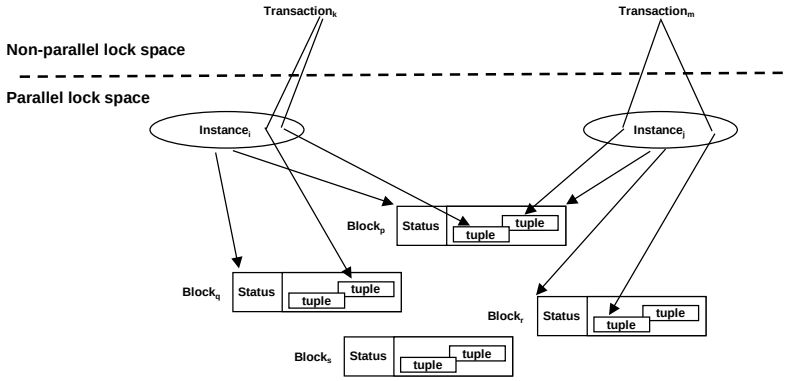
- **Four cases are handled to reduce overall ping cost:**
  - **Read/Read. Instance A wishes to read, on behalf of a user, a storage block recently read by Instance B and which is in B's SGA;**
  - **Read/Write. Instance A wishes to read, on behalf of a user, a storage block recently written by Instance B and which is in B's SGA;**
  - **Write/Read. Instance A wishes to write, on behalf of a user, a storage block recently read by Instance B and which is in B's SGA;**
  - **Write/Write. Instance A wishes to write, on behalf of a user, a storage block recently written by Instance B and which is in B's SGA.**

Oracle Real Application Cluster(9)

- **The Read/Read case does not require any coordination between the instances - instance A may simply read the storage blocks from disk into its own SGA, from B's SGA (i.e. using the interconnection network) or from a disk read operation (the choice depends on the respective performance of the interconnect and the disk read).**
- **In the Read/Write and Write/Write cases, a coordination between the instances is necessary to ensure consistency (read/write) or integrity (write/write). In either of these situations, the current possessor of an up to date version of the data sends it to the requester over the interconnect network. This save the two disk I/Os implied by a ping. RAC implements appropriate redundancies to ensure recoverability in the event of a failure.**
- **In the case of Write/Read, the instance that last read the data block (and has a copy in its SGA) sends it over the interconnect network to the instance that wants to modify the data. This avoids a read access to the disk.**

Oracle Real Application Cluster(10)

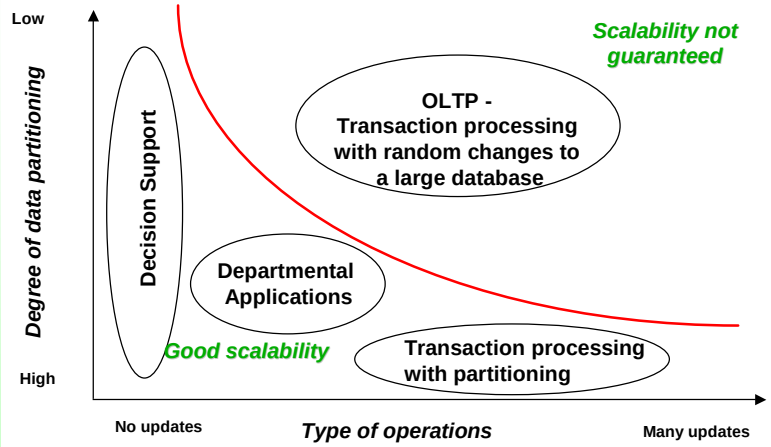
- **RAC Locking Concepts**
  - **Two types of locks:**
    - **Non-PCM Locks (or nonparallel locks): Transaction Locks, Table Locks, System Change Number, Library Cache Locks, Data Dictionary Cache Locks, Data Base Mount Lock**
    - **Locks used to manage the parallel cache (or parallel locks)**



Oracle Real Application Cluster(11)

- **RAC Recovery Logic**
  1. **Several OPS instances are active on a number of nodes**
  2. **A node fails, its failure is detected, and connections between the clients of that node and its instance are migrated to instances on surviving nodes**
  3. **The Cluster Manager (CM) and the Cluster Group Services (CGS) reconfigure themselves to eliminate the failing node. During this process, new service requests from clients are suspended, as are requests in the process of being handled (until the recovery is completed).**
  4. **The lock database is reconstituted, with the database and the locks associated with resources being redistributed to the surviving nodes.**
  5. **Cache recovery and use exploitation of the logs are done**
  6. **Access to the database is once more permitted while recovery continues with Fast-Start Rollback.**

DBMS Architecture Comparisons



Page 35

DBMS Architecture Comparisons(2)

		Decision Support	Departmental Transaction Processing	Partitioned Transaction Processing	Transaction Processing
Share Everything	Advantages	. Intra-request parallelism provides speedup . Requests may easily be optimized	. Inter-request parallelism provides scale up	*Not applicable*	. Efficiency (scale up)
	Disadvantages	. Limited maxi mum number of processors	. Limited maximum number of processors . Not failure-resistant	*Not applicable*	. Limited maximum number of processors . Not failure-resistant
Shared Disks	Advantages	. Intra-request parallelism provides speedup . Data partitioning not needed	. Intra-request parallelism provides speedup . Data partitioning not needed	. Efficiency of transaction processing	. Intra-request parallelism provides scaleup . Data partition ing not needed
	Disadvantages	. Disk interconnect is a potential bottleneck . Data partitioning needed for performance increase . Difficult to optimize requests	. Disk interconnect is a potential bottleneck	. Partitioning makes applications difficult to design	. Disk interconnect is a potential bottleneck . Data partitioning needed for performance increase . Difficult to optimize requests
Share Nothing	Advantages	. Intra-request parallelism provides speedup	. Inter-request parallelism provides scaleup	. Efficiently of transaction processing	. Inter-request parallelism provides scaleup
	Disadvantages	. Data partition ing needed for performance increase . Difficult to optimize requests	. Difficult to optimize requests	. Partitioning makes applications difficult to design	. Partitioning makes applications difficult to design . Controlling data distribution makes database administration difficult . Difficult to optimize requests

Page 36

References

[DEW92] David DeWitt, Jim Gray, «Parallel Data Base Systems: The Future of High Performance Data Base Systems»
CACM, June 1992, Vol. 35, Nº 6, pp. 85-98

[MOH94] C. Mohan, H. Pirahesh, W. G. Tang, Y. Wang, «Parallelism in Relational Database Management Systems»,
IBM Systems Journal, Vol. 33, Nº 2, 1994, pp. 349-371.

[RUD98] Ken Rudin, «When Parallel Lines Meet»,
Byte, May 1998, pp. 81-88.

Product documentation available from IBM and Oracle