In Oracle RAC, the application does not need to be aware of where the data resides (in the memory of one of the database instances or on disk), as the RAC protocols automatically bring the data to the instance to which the application is connected (location transparency of data). The algorithm that implements the high-performance distributed caching protocol while maintaining full data consistency is called Oracle Cache Fusion.

Cache Fusion leverages the private cluster interconnect to provide shared access to data cached on any of the Oracle RAC database instances, virtually creating a single "fused" cache across the whole cluster. Cache Fusion implements direct memory-to-memory transfers of data blocks across instances for high-performance. Oracle RAC database instances only need to read data from storage if the block is not already present in the combined caches across the entire Oracle RAC configuration. Cache Fusion also implements locking protocols to prevent multiple instances from updating the same data block simultaneously, ensuring consistency while providing optimal concurrent data access across the Oracle RAC cache distributed across independent instances in the cluster.

By using a combination of function shipping, data shipping, and distributed data caching, Oracle RAC is uniquely able to optimize analytics, OTLP, batch or any combination of these workloads. Oracle has recently implemented significant enhancements to the Oracle RAC algorithms, further enhancing their performance.

ORACLE CLUSTERWARE

Oracle Clusterware is the technology used in the RAC architecture that transforms a collection of servers into a highly available unified system. Oracle Clusterware provides failure detection, node membership, node fencing and optimal resource placement. It provides cluster-wide component inter-dependency management for RAC and other applications in the cluster. Clusterware uses resource models and policies to provide high availability responses to planned and unplanned component downtime.

For more information on Oracle Clusterware visit http://www.oracle.com/goto/clusterware

ORACLE ASM

Automatic Storage Management (ASM) is a file system and volume manager integrated into the database that enables sharing and scaling of storage capacity across thousands of storage devices with 24/7 availability. ASM provides storage management across all servers of a cluster for Oracle RAC databases. It stripes data to prevent hot spots and to maximize I/O performance. ASM allows the online addition or removal of storage capacity. ASM can maintain redundant mirror copies of data to provide fault tolerance, or it can be used on top of vendor supplied reliable storage arrays. Data management is done by selecting the desired reliability and performance characteristics for classes of data rather than with human interaction on a per database file basis.

ASM solves many of the practical management problems of large clustered databases. As the size of a database and associated data increases towards thousands of storage devices, or dozens of servers, the traditional techniques for storage management do not scale efficiently and become prone to human error. Other tasks, such as manual load balancing, also become highly complex. Oracle ASM solves these issues for Oracle RAC as well as for Oracle single instance databases.

For more information on Oracle ASM visit http://www.oracle.com/goto/asm

APPLICATION CONTINUITY 2.3

Oracle RAC is an active-active database management system in which the database service remains continuously available, even after a RAC database instance or server failure. This is in contrast to technologies utilizing an active-passive configuration that must restart and resume processing after the active database instance failed, normally leading to a failover followed by a brownout that can last for many minutes. In both scenarios, SQL and any uncommitted transactions that are being processed at the point of failure are aborted and rolled back by the database management system. Complex application logic must be written for every application to handle these failures. Since failures are rare, however, this logic is often ignored or rarely tested, leading to application failures and the potential for logically corrupted data.