Large Scale Data Warehouses on Grid: Oracle Database 10*g* and HP ProLiant Servers

Meikel Poess

Oracle Corporation
400 Oracle Parkway
Redwood City, CA-94065
USA
meikel.poess@oracle.com

Raghunath K. Othayoth

Hewlett-Packard Company
20555 Tomball Parkway
Houston, TX-77070
USA
raghunath.othayoth@hp.com

Abstract

Grid computing has the potential to drastically change enterprise computing as we know it today. The main concept of grid computing is viewing computing as a utility. It should not matter where data resides, or what computer processes a task. This concept has been applied successfully to academic research. It also has many advantages for commercial data warehouse applications such as virtualization, flexible provisioning, reduced cost due to commodity hardware, high availability and high scale-out. In this paper we show how a large-scale, high-performing and scalable grid-based data warehouse can be implemented using commodity hardware (industry-standard x86-based), Oracle Database 10*g* and the Linux operating system. We further demonstrate this architecture in a recently published TPC-H benchmark.

1. Introduction

Grid computing has the potential to drastically change enterprise computing as we know it today. The main concept of grid computing is viewing computing as a utility. It should not matter where data resides, or what computer processes a task. It should be possible to request information or computation and have it delivered – as much as is needed, and whenever it is needed. This is analogous to the way electric utilities work, in that one does not know where the generator is, or how the electric grid is wired. One just asks for electricity and gets it. The goal is to make computing a utility – a commodity, and ubiquitous. This, however, is the view of utility computing from a user's point of view. From an implementation point of view, grid is much more – it is about data virtualization, resource provisioning and availability.

Virtualization enables components on all levels, such as storage devices, processors and database servers, to collaborate without creating rigidity and brittleness in the system. Rather than statically determining where a database physically locates its data or which exact server the database runs on, virtualization enables each component of the grid to react to change more quickly and to adapt to component failures without compromising the entire system. Provisioning ensures that all those that need or request resources are getting what they need. Once resources are virtualized, they can be dynamically allocated for various tasks based on changing priorities. Both hardware resources and data need to be allocated to databases and application servers. Most importantly, resources are not standing idle while tasks are not being serviced. High availability and scalability ensures that all the data and computation will always be there – just as a utility company must always provide electric power – even when systems are scaled out.

Research institutes have embraced the idea of grid computing for some time. Taking advantage of idle computing resources around the globe, academic grids have been established to solve complex computational problems. For instance, SETI@home, a project at the University of California at Berkeley, uses idle PCs on the Internet to mine radio telescope data for signs of extraterrestrial intelligence [1]. It uses a proprietary architecture consisting of a data server, located at U.C. Berkeley, which sends about 350 KB of radio frequency data at a time to subscribed computers. These computers run a client program as a background process, as a GUI application, or as a screen saver, and results are returned to the data server. In November 2004, IBM, along with representatives of the world's leading science organizations, launched the World Community Grid. Similar to SETI@home, it consolidates the computational power of subscribed computers around the globe. In its first project it is directing its computing

Proceedings of the 31st VLDB Conference, Trondheim, Norway, 2005

power to research designed to identify the proteins that make up the human proteome and, in doing so, to better understand the causes and potential cures for diseases like malaria and tuberculosis [8]. SETI@home and the World Community Grid are architectures that scavenge computing cycles of a completely distributed set of idle, heterogeneous servers. This concept can be applied very successfully to problems that are of common interest to the "world community"– the data (radio waves, human protein) is not subject to any security concerns since it is readily available. Most importantly, the outcome (discovery of extraterrestrial life or cures for diseases) is of everybody's interest. The tasks sent to nodes in these grids involve highly computational problems on a relatively small data set – 350 KB in SETI@home, for example – keeping computers easily busy for days with low network requirements. ObjectGlobe [9], similar to other projects like Jini [10], Mariposa [11], Garlic [12] or Amos [13], have studied distributed query processing. They constitute infrastructures that facilitate distributed processing of complex queries executing multiple operators from different sites while also dealing with security concerns. ObjectGlobe takes this concept further into an open environment. TPC-H-like data warehouse applications execute large joins and sort operations, making it necessary to tightly couple the nodes of a distributed system such as the data warehouse grid we are proposing in this paper. The above projects propose a framework for highly distributed systems rather than tightly connected systems.

Applying the above concepts directly to corporate data warehouses is appealing but difficult because of data security concerns and network performance problems. Corporate data warehouses contain business intelligence that must not be made known to competitors. Also, the results are generally not in the public interest. Although network bandwidth is increasing, computing table joins of terabyte-sized tables between nodes connected via the Internet is not feasible. Businesses rely on having business intelligence delivered at a determined time. The characteristics of grid computing are very appealing for today's corporate data centers. Companies must respond to accelerating business changes fueled by churning market demands, an increasingly dynamic global economy and constant technological innovations. Traditionally, data warehouses have been deployed as a series of islands of systems that cannot easily share resources; this results in significantly underutilized IT systems and soaring costs. Grid-based data warehouses can solve this dilemma.

This paper presents technologies from HP and Oracle that leads the way to large-scale data warehouse grids that deliver high performance while providing flexible provisioning, high availability and resource virtualization. As the prices for Linux-run commodity hardware (industry-standard x86-based) have dropped steadily and as the performance and reliability of these systems have improved enormously, the industry is taking a serious look at industry standard server-based grid configurations for large-scale data warehouse applications. In addition, Oracle's shared-disk architecture is ideal for the key features of data warehouse grids: virtualization, provisioning and availability. The myth that shared-disk implementations have scaling issues for large-scale data warehouses is addressed by a 12-node published 1000-GB scale factor TPC-H benchmark, which delivers performance comparable to equally sized Symmetrical Multi-Processing (SMP) systems. New technologies used in this benchmark, such as InfiniBand, HP high performance storage arrays and Oracle Grid support, overcome these scalability issues. This benchmark demonstrates a milestone on the path toward a true grid. Even though this benchmark did not exercise all of the aspects of grid computing, it addresses scalability, performance and total cost of ownership (TCO).

The organization of this paper is as follows: Section 2 defines the Oracle and HP vision for a data warehouse grid. It further explains how large-scale grid-based data warehouses can be built using industry standard hardware outlining the necessary hardware architectures and hardware features that are supported in HP, such as shared-disk storage, high-performance interconnect, and high-performance computers. Section 3 outlines the features in Oracle Database 10*g* to implement a scalable shared-disk data warehouse grid. Finally, Section 4 gives an in-depth analysis of a published 1000-GB TPC-H benchmark [2] with the emphasis on how the HP and Oracle hardware and software features were used to achieve high performance.

2. Large-Scale Data Warehouse Grids

The goals of grid computing are closely aligned with capabilities and technologies that Oracle and HP have been developing for years. Oracle's latest RDBMS release, Oracle Database 10*g*, provides substantial grid computing technology. Figure 1 shows how Oracle and HP envision grid computing by orchestrating many small servers and storage subsystems into one virtual computer. There are three levels of abstraction: the first level contains server nodes, the second level contains database applications and the third level contains storage subsystems. This three-level architecture, which allows for a very flexible grid implementation, requires a shared-disk implementation. Shared-disk

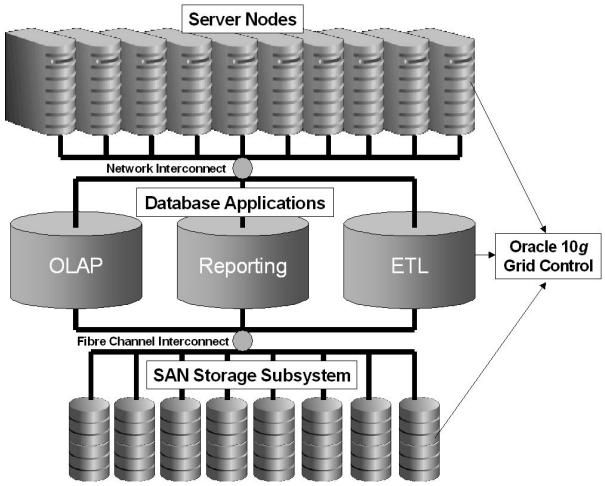

Figure 1: ***Large-Scale Data Warehouse Grid***

architectures are characterized by all nodes having access to all data in the database. Hence, there is only one database – logically and physically. In a shared-nothing architecture the database is divided into partitions. Each server node has exclusive access to a single partition, and only one server node may own and access a particular partition at a time.

Both architectures share many advantages. They are able to use commodity hardware to reduce TCO. They can overcome the natural limitations of SMP systems, which can be scaled up by adding hardware such as processors, memory or disk arrays, but are limited by the resources available on a single server. Today's largest commercially available SMP systems (Sun Fire(TM) E25K servers) are limited to 144 CPUs (72 dual-threaded processors) [14]. In contrast, grid systems can be scaled out virtually without limits by adding additional server nodes.

In addition to the above advantages, shared-disk architectures provide dynamic resource sharing between applications of one database and between databases and provisioning by virtualization of resources. Another advantage of a shared-disk architecture is the increased availability. Availability is critical for today's large-scale data warehouses, which provide business-critical services and must therefore operate 24x7. Adding or removing additional systems (scale-out) can be done without interrupting the system as a whole.

2.1. Advantages of Grid-Based Data Warehouses

2.1.1 Commodity Hardware

Performance and reliability of Linux-run commodity hardware-based systems (industry-standard x86-based) have improved enormously while prices have dropped steadily. TPC-H results show that commodity hardware-based systems can deliver the same performance as large SMP systems at half the price [3,4].

HP ProLiant servers provide features differentiating them from the competition; the number and variety of options and features available for these servers has grown rapidly, and continues to grow. Development of the ProLiant servers illustrates HP's consistent efforts to provide customers with the world's broadest industry-standards-based server portfolio and industry-leading innovation in areas such as management, availability, security and virtualization. For instance, the ProLiant DL585 servers used in the TPC-H benchmark are 4U rack-optimized 4-way servers created for large data center deployments requiring enterprise-class performance, uptime and scalability, plus easy management and expansion. They offer customers running 32-bit applications increased performance and memory addressability. While allowing IT organizations to protect their large x86 investments, the servers also provide a path to more powerful, 64-bit computing to meet evolving business needs for greater processing power and performance.

2.1.2 Scale-Out vs. Scale-Up

Scalability is the ability of a system to maintain desired performance levels as it grows. This is necessary when a system expands to accommodate increased data or a larger number of concurrent users. A system can be either scaled up or scaled out. Traditionally, data warehouse applications have been deployed on scale up (high-end SMP) systems. In recent years, the industry has observed another trend: scale-out configurations. This is fueled by a drop in prices for commodity hardware and its improvement in performance and reliability.

Scale-up is achieved by adding devices – typically processors, memory, disks and network interface cards – to an existing database server. This is also referred to as vertical scaling or vertical growth. Multiple services or applications can be serviced by a single node, which reduces the total administration costs. Server capacity can be easily increased in a server with sufficient expansion capability by adding CPUs, memory and other components. Software licensing costs may be lower, since the software is hosted on only one server. On the other hand, all services will be unavailable if the server is down. The availability of the server is limited and depends on server resources. If the server load is maximized or the server fails, then the services may be discontinued until the server is replaced with a more capable or operational server. The scalability of the server is limited, and depends on server resources. If it is running at maximum capacity, it cannot be scaled up. The only alternative would be to replace the existing server. Typically, the initial expense of scale-up server is higher than the 4-CPU industry-standard server used for scale-out. This is due to the larger capabilities and often more complex architectures of large SMP servers.

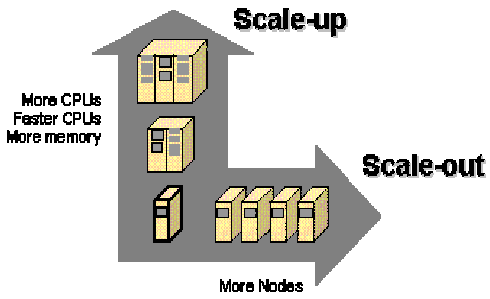

Figure 2: ***Scale-Up vs. Scale-Out***

Scale-out is achieved by adding servers and distributing the existing database application across all of the servers. A scale-out architecture has the potential for scalable and high-hosting highly available services, allowing capacity improvements without interruptions. Servers can be maintained and supported much more easily, since services do not need to go down in order to add or remove a server for repair or replacement. There are linear and predictable costs associated with scaling out an application across multiple servers. On the other hand, depending on the initial performance requirements, the cost per server and the cost

of infrastructure may be higher than the implementation on one server. Software licenses may be higher when licenses are sold on a per-server basis. Also, management may increase for each server added to the array if appropriate best practices are not defined for the environment. This paper explores the potential of a scale-out architecture to host scalable and highly available applications.

2.1.3 Provisioning

Provisioning means allocating resources where they are needed. With Oracle's shared-disk architecture resources can be virtualized. This enables enterprises to dynamically allocate resources for various enterprise tasks based on changing business priorities, reducing underutilized resources and decreasing overcapacities. Data warehouse applications can easily share compute and data resources by migrating between servers of the grid to leverage available resources. Schedulers on the grid track resource availability, and assign resources accordingly.

For instance, in a business intelligence environment, to increase the productivity of data analysts during the day, most resources should be allocated to online analytical processing (OLAP) queries against data marts, while resources for reporting queries against the data warehouse should be limited. Let's assume that the data warehouse is synchronized twice a day with an operational data store as part of an extraction transformation and load (ETL) task. Hence, during short periods, it is imperative that resources are assigned to the ETL process. With a shared-disk-based business intelligence grid, resources can be assigned to databases and applications without shutting down databases. In our example above, during the day, most nodes of the grid can be assigned the OLAP applications, while only a few are assigned to reporting queries. Twice a day, when the ETL application runs, nodes can be temporarily detached from the OLAP database and assigned to the ETL application. At night, when no OLAP activity exists most nodes can be assigned to answer reporting queries. This can be specified in the query or application.

2.1.4 Availability

Another advantage of shared-disk grid architectures is their increased availability. Availability is critical for today's large-scale data warehouses, which provide business-critical services and must therefore operate 24x7. For instance, Amazon.com's online recommendation system is fed by an industry-standard grid data warehouse system. A shared-disk grid increases availability by employing multiple symmetrical systems sharing the same data. In addition to increased availability, symmetrical systems lead to an increase in computing power. Nodes can fail without compromising the availability of the entire system. They can also be extended or shrunk without bringing down the entire system, increasing system availability through system (hardware and software) upgrades.

2.2 Hardware Requirements of a Shared-Disk Grid

2.2.1 Storage

A shared-disk cluster architecture is characterized by all nodes sharing all the data in the database, which necessitates a storage area Network (SAN). HP SAN provides the data communication infrastructure and management features for most demanding scale-out clusters. In a SAN, server nodes can be added and removed while their data remains in the SAN. Multiple servers can access the same storage for more consistent and rapid processing. The storage itself can be easily increased, changed, or reassigned. In a SAN, multiple compute servers and backup servers can access a common storage pool. Properly designed SAN storage is highly available, allowing many servers to access a common storage pool with the same degree of availability

A typical SAN is assembled from adapters, SAN switches, and storage enclosures. Fibre Channel host bus adapters are installed into hosts like any other PCI host bus adapters. SAN switches provide scalable storage connectivity of almost any size. Storage enclosures place the array controller capabilities close to the physical disk drives.

2.2.2 High-Speed Inter-Node Communication

Data warehouse queries tend to involve complex, long running operations. In a shared-disk grid configuration, they are broken down into smaller sub-queries, which are disseminated to participating nodes of the grid. Upon completion, the results of these sub-queries are forwarded to other nodes for further processing or coalesced into the final result. Performance for these operations is directly correlated to the speed of the inter-node connection.

In a grid configuration, the network connecting the independent computing nodes is called the interconnect. In a data warehouse grid, a low-latency, high-throughput interconnect is critical for achieving high performance. During typical data warehouse queries, such as join operations and data load, it is important to effectively pass messages and to transfer data blocks between nodes. It is not uncommon for queries to require more than 100 MB/s throughput per node. The two most commonly available cluster interconnect technologies on industry-standard hardware are Gigabit Ethernet (GigE) and InfiniBand (IB).

Cluster technology is beginning to be adopted by mainstream customers, and performance between nodes will largely determine the performance scalability. Although Ethernet technology is ubiquitous in IT organizations, a dedicated IB fabric not only significantly increases overall performance, but more than justifies the additional cost.

3. Grid Support Within Oracle Database 10*g*

In Oracle Database 10*g*, users at separate nodes can access the same database while simultaneously sharing resources with all other nodes, yielding benefits such as increased transaction processing power and higher availability of multiple nodes. Also, scaling out with multiple nodes en-

ables a cluster to overcome the limitations of a single node, such as memory, CPU or I/O-bandwidth. This enables the grid configuration to supply much greater computing power. These features make Oracle 10*g* the ideal platform for grid-based data warehouses.

In a grid-based data warehouse, the execution of any particular operation must adapt to the resources available to it at the time it starts (automatic resource allocation). For instance, if a SQL operation is the only operation running in a grid, it should be given all resources. In contrast, if there are multiple SQL operations running, each should be given the same amount of resources without overloading the grid. This is important in scale-out situations – while the grid grows, new resources should be made available to operations without any user intervention, especially without changing the SQL text.

Furthermore, for a grid system, it is important to support features like rapid recovery from failures, support for physical and logical standby databases, online maintenance operations, sophisticated diagnosis and repair of failure conditions and transparent application failover. Oracle Database 10*g* provides these features.

3.1 Dynamic Resource Allocation

Under the term dynamic resource allocation fall many features within the Oracle RDBMS. This paper, however, focuses on those that are applicable to grid-based data warehouse systems – determining the optimal execution model for parallel query processing, choosing the optimal degree of parallelism, minimizing inter-node processing and performing interprocess communication efficiently.

Most data warehouse SQL operations are executed in parallel; that is, operations are divided into smaller portions necessary to run in parallel in multiple processes. This is called parallel execution or parallel processing, a feature most RDBMS implementations offer today. The next sections explain how Oracle Database 10*g* optimally performs parallel processing in grid data warehouses by utilizing all available resources, dynamically choosing the degree of parallelism and minimizing inter process communication.

3.1.1 Parallel Processing in an Oracle Grid System

One of the most challenging tasks for grid-based data warehouse systems is to perform parallel processing efficiently. In Oracle Database 10*g*, processes executing on a portion of the data are called parallel execution servers[1]. One process, known as the parallel execution coordinator, parses each SQL statement to optimize and parallelize its execution. After determining the execution plan of a statement, the parallel execution coordinator decides the parallelization method for each operation in the execution plan. The coordinator must choose whether an operation can be performed in parallel and, if so, how many parallel execution servers on which nodes to enlist. Then, it dispatches the execution of an operation to several parallel execution servers and coordinates the results from all of the server processes.

In detail, the parallel execution coordinator dynamically divides the work into units called granules. One granule at a time is sent to a single parallel execution server. How granules are generated depends on the operation. For instance, for a table scan, granules are generated as ranges of physical blocks of the table to be scanned. The mapping of granules to execution servers is determined dynamically at execution time. When an execution server finishes with one granule, it gets another granule from the coordinator if there are any granules remaining. This continues until all granules are exhausted, that is, the operation is completed.

The number of parallel execution servers used for an operation is the degree of parallelism (DOP). The default DOP for any operation is set to twice the number of CPUs available: **`DOP = 2 x #CPUs`**

3.1.2 Inter- and Intra-Operation Parallelism

Data warehouse SQL statements usually perform complex, multiple table access, join and sort operations. The Oracle RDBMS applies a producer/consumer model to data operations. This means only two operations in a given execution tree need to be performed simultaneously. Each operation is given its own set of parallel execution servers. Therefore, both operations have parallelism. Parallelization of an individual operation where the same operation is performed on smaller sets of rows by parallel execution servers achieves what is termed intra-operation parallelism**.** When

```
SELECT T1.id,MAX(T1.val) maximum
FROM T1, T2
WHERE T1.id= T2.id
GROUP BY T1.id;
```

two operations run concurrently on different sets of parallel execution servers with data flowing from one operation into the other is called inter-operation parallelism**.** Consider the following simple join between two tables T1 and T2 aggregating on one column.

Assuming that the DOP for this operation is 64 this query is parallelized in the following fashion (see Figure 3 for the query's execution plan). Each parallel execution server set (S1 and S2) consists of 64 processes. S1 first scans the table `T1` while S2 concurrently fetches rows from S1 building the hash table for the following hash join operation (inter-operation parallelism).

[1] The term "server" is used to describe a process that works on a portion of the entire SQL operation

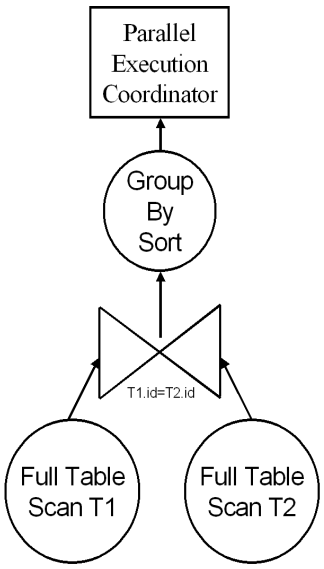

Figure 3: ***Parallel Execution Plan EP1***

After S1 has finished scanning the entire table `T1`, it scans the table `T2` in parallel. It sends its rows to parallel execution servers in S2, which then perform the probes to finish the hash-join in parallel. After S1 is done scanning the table `T2` in parallel and sending the rows to S2, it switches to performing the `GROUP BY` in parallel. This is how two server sets run concurrently to achieve inter-operation parallelism across various operators in the query execution plan while achieving intra-operation parallelism in executing each operation in parallel.

Another important aspect of parallel execution is the re-partitioning of rows while they are sent from parallel execution servers in one server set to another. For the query plan in Figure 2, after a server process in S1 scans a row of `T1`, which server process of S2 should it send it to? The partitioning of rows flowing up the query tree is decided by the operator into which the rows are flowing. In this case, the partitioning of rows flowing up from S1 performing the parallel scan of `T1` into S2 performing the parallel hash-join is done by hash partitioning on the join column value. That is, a server process scanning `T1` computes a hash function of the value of the column `T1.id` to decide the number of the server process in S2 to send it to. Depending on the table partitioning, Oracle can optimize this operation using partial or full partition-wise joins, minimizing inter-process communication (see Section 3.4)

In a grid environment, each node hosts a subset of the parallel server processes. In a 16-node, 4-CPU-per-node configuration, each node holds approximately 64 parallel servers. The entire system may hold 1024 parallel servers. The query coordinator prepares the execution depending on resource availability in the grid. In addition to the usual query parsing, query plan generation and query parallelization steps, query optimization must also determine which DOP to choose and on which node to execute the query.

3.2 Dynamic Degree of Parallelism

In a grid environment, the degree of parallelism cannot be static – it must be adjusted according to resource availability (servers, memory, disk I/O, etc.). Resource availability changes in two ways. At any given time in a grid, more or fewer systems can be available to a user for running SQL operations. This could be because resources are shared between multiple applications and/or different users in a grid, or because the system scaled out. Since the DOP is directly related to the number of CPUs, the DOP adjusts automatically as new nodes are added to the system. The DOP for an 8-node 4-CPU-per-node grid is 64 (see Section 3.1). If the number of nodes doubles to 16, the DOP is automatically adjusted to 128. Similarly, if nodes are eliminated from the grid, the DOP decreases proportionally.

The second way system resources can be different depends on how many operations are actually running on the system. Usually, there are many users connected concurrently to a data warehouse system issuing queries at any given time. To optimize execution time for all users and to utilize all system resources, Oracle Database 10*g* dynamic parallelism allows for adjusting the degree of parallelism before each operation starts (Adaptive Multiuser Algorithm). When a system is overloaded and the input DOP is larger than the default DOP, the Adaptive Multiuser Algorithm uses the default degree as input. The system then calculates a reduction factor that it applies to the input DOP. For instance, using a 16-node 4-CPU-per-node grid, when the first user enters the system and it is idle, it will be granted a DOP of 128. The next user entering the system will be given a DOP of 64, the next 32, and so on. If the system settles into a steady state of, let's assume, 16 concurrent users, all users will be given a DOP of 8, thus dividing the system evenly among all the parallel users.

3.3 Inter-Process Communication

Inter-process communication (IPC) refers to sending data and control messages between parallel execution processors of a grid. IPC is very high during data warehouse operations when join, sort and load operations are performed in parallel. Oracle uses a message-based protocol with its own flow control. Messages between processes on the same node are passed on using shared memory. Messages between processes on different nodes are sent using an operating-system-dependent IPC protocol. Oracle supports a variety of protocols and wires. With Linux, Oracle supports Ethernet and IB. The protocols that can be run on Ethernet are TCP/IP and UDP/IP. IB supports TPC/IP, UDP/IP, and uDAPL. Performance of the different protocol/wire combinations differs significantly.

Grid-based data warehouses demand a high-performance IPC. The amount of interconnect traffic depends on the operation and the number of nodes participating in the operation. Join and sort operations use IPC more than simple aggregations because of possible communication between parallel execution servers. The amount of interconnect traffic varies significantly, depending on the distribution method. Partial partition-wise joins (see next section), in which only one side of the join is redistributed, result in less interconnect traffic, while full partition-wise joins (see next section), in which no side needs to be redistributed, result in the least interconnect traffic.

The amount of interconnect traffic also depends on how many nodes participate in a join operation. The more nodes that participate in a join operation, the more data

needs to be distributed to remote nodes. For instance, in a 4-node grid cluster with 4 CPUs on each node to maximize load performance with external tables, the DOP is set to 32 on both the external and internal tables. This will result in 8 parallel server processes performing read operations from the external table on each node, as well as 8 parallel server processes performing table creation statements on each node. On the other hand, if there are 4 users on average on the systems issuing queries, it is very likely that each user's query will run locally on one node, reducing the number of remote data distributions to almost zero.

3.4 Decreasing Inter-Process Communication

Oracle Database 10*g* minimizes IPC traffic for large join operations, significantly improving performance and scalability for grid-based data warehouse operations. The most important features are partition-wise joins and node locality. In the default case, parallel execution of a join operation by a set of parallel execution servers requires the redistribution of each table on the join column into disjoint subsets of rows. These disjoint subsets of rows are then joined pair-wise by a single parallel execution server. If at least one of the tables is partitioned on the join key, Oracle can avoid redistributing partitions of this table.

3.4.1 Full and Partial Partition-Wise Join

Partition-wise joins minimize the amount of data exchanged between parallel execution servers. In a grid data warehouse, this significantly reduces response time by limiting the data traffic over the interconnect (IPC), which is the key to achieving good scalability for massive join operations. Depending on the partitioning scheme of the tables to be joined partition-wise, joins can be full or partial.

A full partition-wise join divides a large join into smaller joins between a pair of partitions from the two joined tables. For the optimizer to choose the full partition-wise join method, both tables must be equipartitioned on their join keys. That is, they have to be partitioned on the same column with the same partitioning method. Parallel execution of a full partition-wise join is similar to its serial execution. Instead of joining one partition pair at a time, multiple partition pairs are joined in parallel by multiple parallel query servers. The number of partitions joined in parallel is determined by the DOP.

Figure 4 illustrates the parallel execution of a full partition-wise join between two tables, T1 and T2, on 4 nodes. Both tables have the same degree of parallelism and the same number of partitions (32). As illustrated in the picture, each partition pair is read from the database and joined directly. There is no data redistribution necessary, thus minimizing IPC communication, especially across nodes. Defining more partitions than the degree of parallelism may improve load balancing and limit possible skew in the execution. If there are more partitions than parallel query servers, each time one query server completes the join of one pair of partitions, it requests another pair to join. This process repeats until all pairs have been processed. This method enables the load to be balanced dynamically when the number of partition pairs is greater than the degree of parallelism (for example, 128 partitions with a degree of parallelism of 32).

Unlike full partition-wise joins, partial partition-wise joins can be applied if only one table is partitioned on the join key. Hence, partial partition-wise joins are more common than full partition-wise joins. To execute a partial partition-wise join, Oracle Database 10*g* dynamically repartitions the other table based on the partitioning of the partitioned table. Once the other table is repartitioned, the execution is similar to a full partition-wise join. The redistribution operation involves exchanging rows between parallel execution servers. This operation leads to interconnect traffic in grid environments, since data needs to be repartitioned across node boundaries.

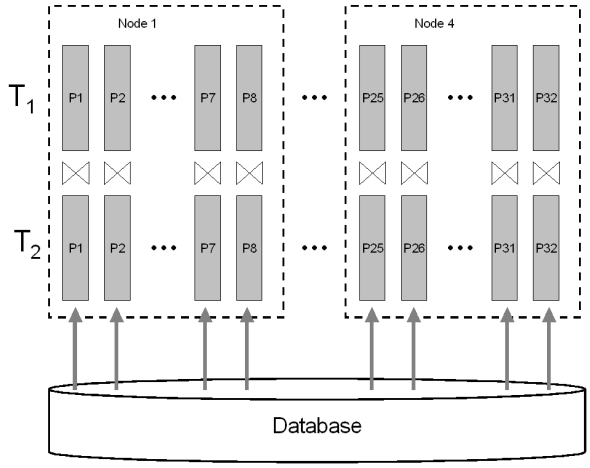

Figure 4: ***Full Partition-Wise Join***

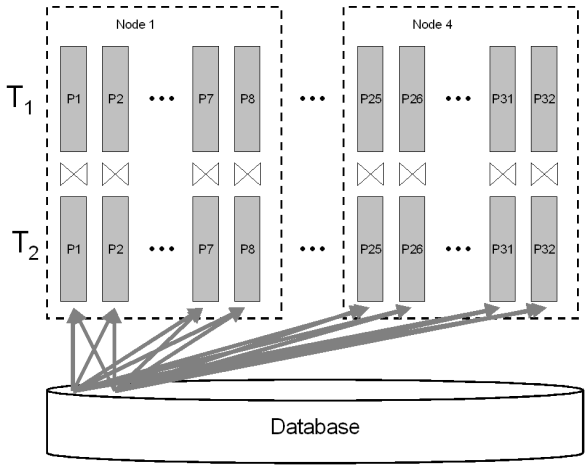

Figure 5: ***Partial Partition-Wise Join***

Figure 5 illustrates a partial partition-wise join. It uses the same example as in Figure 2, except that T2 is not partitioned. Before the join operation is executed, the rows from T2 are dynamically redistributed on the join key as illustrated by the arrows from the database into P1–P32. For

readability, not all arrows are drawn. During this redistribution, data is sent from one parallel server to another parallel server using IPC.

3.4.2 Node Locality

Another feature in Oracle to reduce IPC traffic is node locality. If the Adaptive Multiuser Algorithm determined the DOP to be less than or equal to double the number of CPUs per node, queries are run locally on one node. Which node gets to execute the operation is dynamically determined by the load on each system at that time. As resources are becoming available and the DOP is larger than double the number of CPUs per node, operations are executed on multiple nodes. However, Oracle Database 10*g* always tries to limit the number of nodes executing one operation. This is advantageous because it minimizes the interconnect requirements. For instance, if there are 16 users running on a 16-node 4-CPUs-per-node grid, the DOP for each operation is 8. Hence, each operation runs on one node. If the number of user drops to 8, each operation is run on two systems.

3.4.3 Dynamic Partition of Grid Resources

Types of operations in a data warehouse range from long-running periodic reporting queries over OLAP-type queries to data maintenance operations. For some data warehouses, it is possible to dedicate a specific time period to each of the above operations. In this case, the entire system either calculates reporting queries, or is available for online users or for data maintenance operations. However, some data warehouses, especially globally operating systems, cannot afford to dedicate the entire system to specific tasks, but must run them concurrently. In Oracle Database 10*g* it is possible to dedicate a subset of the grid to specific tasks. This can be done dynamically without shutting down the system. For instance, during peak hours, the system can be available for OLAP users. During off hours, 20% of the system is dedicated to occasional user queries while 40% is dedicated to reporting queries and 40% is dedicated to data maintenance operations.

So far we have assumed that the entire grid runs one database. It is also possible to run multiple databases on the same grid. These databases, sharing the same disk subsystem, can take advantage of further features in Oracle Database 10*g*, such as Transportable Tablespaces. Transportable Tablespaces offers grid users an extremely fast mechanism to move a subset of data from one Oracle database to another. It allows Oracle data files to be unplugged from a database, moved or copied to another location and then plugged into another database. Unplugging or plugging a data file involves only reading or loading a small amount of metadata. Transportable Tablespaces also supports simultaneous mounting of read-only tablespaces by two or more databases.

If data needs to be shared as it is created or changed, rather than occasionally shared in bulk, Oracle Streams can stream data between databases or nodes in a grid. It provides a unique method of information sharing, combining message queuing, replication, events, data warehouse loading, notifications and publishing/subscribing. It also:

- Keeps two or more copies in sync as updates are applied
- Captures database changes
- Propagates database changes to subscribing nodes
- Applies changes
- Detects and resolves any conflicts
- Can be used directly by applications as a message queuing feature, enabling communications between applications in the grid

4. Industry-Standard TPC-H on a Shared-Disk Grid Configuration

TPC-H, the industry's leading decision support benchmark, exemplifies decision support systems (DSSs) that examine large volumes of data, execute queries with a high degree of complexity and provide data used to answer critical business questions. It consists of a suite of business-oriented ad-hoc queries and concurrent data modifications. The queries and the data populating the database have been chosen to have broad industry-wide relevance. Queries are run in two ways to simulate a single-user and multi-user environment: First, queries are submitted by a single stream. In this test, each query has all resources available, running massively in parallel (single-user or power test). Then, multiple streams are run concurrently, each running one query at a time (multi-user or throughput test). Systems that want to excel in a TPC-H benchmark run have to prove that they can handle both the single and multi-user tests.

The TPC-H performance metric is called the TPC-H Composite Query-per-Hour Performance Metric (QphH@Size). It reflects multiple aspects of the system's capability to process queries. The TPC-H Price/Performance metric is expressed as $/QphH@Size, where $ is the 3-year cost of ownership of the hardware and software components. In addition, there is a timed portion of database load time reported as a secondary metric. The TPC-H benchmark specification is available at http://www.tpc.org/tpch/.

This section gives a detailed overview of the recently released 1000-GB TPC-H benchmark by HP and Oracle. It demonstrates that:

- Clustered ProLiant systems with AMD Opteron–-x86 processors deliver performance comparable to large SMP systems
- Large-scale data warehouses can be successfully deployed using an industry-standard hardware grid configuration to deliver world record performance
- The Linux operating system (Red Hat Enterprise Linux AS 3) handles the throughput and processing demands required to achieve the benchmark result
- Oracle Database 10*g* delivers consistent, high-performance query execution in grid environments

This result builds on an earlier 3000-GB TPC-H benchmark result on an 8-node HP ProLiant cluster to demonstrate the commitment of HP and Oracle to this architecture. The benchmark proactively supports current and potential customers that are considering industry-standard hardware running Linux for data warehouses.

4.1 Benchmarked Configuration

The benchmarked configuration was a 12-node ProLiant DL585 cluster connected to a 14.7-TB storage fabric SAN (Figure 6), comprised of 12 HP StorageWorks SAN Switch 2/16s (SAN Switch 2/16) and 48 HP StorageWorks Modular Smart Array 1000 (MSA1000) devices. Each HP ProLiant DL585 server contained 6 dual-port HP StorageWorks Fiber Channel 2214DC Host Bus Adapters (HBAs). Each port connects to one of 12 SAN Switch 2/16s. Each SAN Switch 2/16s has 4 HP StorageWorks MSA1000s connected to it.

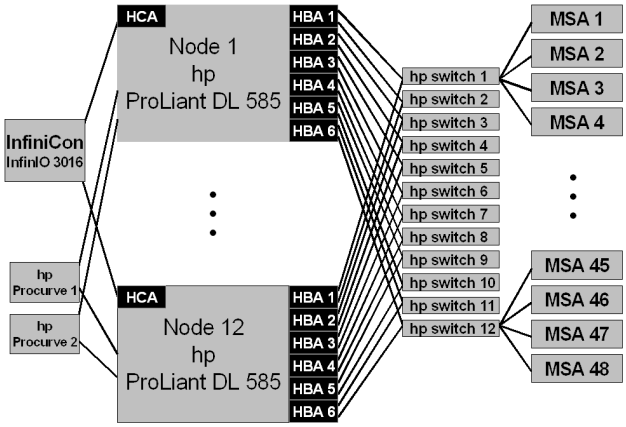

Figure 6: ***TPC-H Benchmark Configuration***

Each server was configured with one InfiniCon Systems InfiniServ 7000 Host Channel Adapter (IB controller) connected to an InfiniCon Systems InfinIO 3016 switch (IB switch), used as cluster interconnect, as shown in Figure 6. The cluster interconnect protocol was UDP/IP over IB. The IB interconnect was chosen because it provides higher performance and lower latency than Gigabit Ethernet. The InfiniCon3016 switch is one of several roughly equivalent products to implement this approach. Each server has two on-board GigE NICs, each connected to a HP ProCurve 4148gl switch. One ProCurve 4148gl switch was used for cluster manager communication (cluster heartbeat) and the other was used for user connectivity.

4.1.1 Server

The ProLiant DL585 is an x86 4-way server, based on AMD Opteron processor technology. Each ProLiant DL585 server was configured with four 2.2-GHz/1-MB AMD Opteron Model 848 processors (Figure 7). The design of the DL585 server is optimized for the AMD Opteron 8000 series chipset and the AMD 800 series of Opteron microprocessors.

The AMD Opteron processor implements the x86 instruction set with a 64-bit memory space. The processor

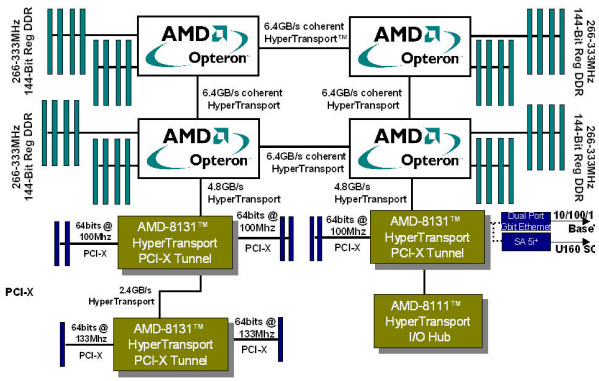

Figure 7: ***Detailed Server Configuration***

runs 32-bit x86 programs in native mode without application code changes, and provides a 64-bit mode for running 64-bit applications. The processor provides program-controlled execution in either 32-bit or 64-bit mode; 32-bit applications can run on top of a 64-bit operating system. The compatibility for 32-bit jobs is in the microcode, which imposes only a small penalty for the conversion to a fixed-length instruction set.

The ProLiant DL585 supports 64 GB of memory. In the benchmarked configuration, 8 GB of memory per server was selected because Oracle Database's System Global Space (SGA) does not require large amounts of memory in decision support systems. For the same reason, a 32-bit version of Oracle Database 10*g* was used.

In ProLiant DL585, HyperTransport links, a universal high-speed chip-to-chip communications technology, is used to interconnect processors and memory. Applications that require high-bandwidth, low-latency access to system memory – for example, contain large sort operations common in data warehouse queries – benefit significantly from fast memory access. Oracle Database inter-process communications within a node, which is implemented using shared memory, also benefit from fast memory access. The ProLiant DL585 supports cache-coherent non-uniform memory access (ccNUMA). As each processor contains its own memory controller, when a processor accesses memory on its own local memory system, the latency is relatively low, especially when compared to a similar SMP system. If a processor accesses memory located on a different processor, then the latency will be higher. Many operating systems and databases servers can take advantage of ccNUMA.

HyperTransport links are also used to connect a processor to an I/O subsystem. The ProLiant DL585 server contains eight 64-bit PCI-X slots. In the benchmarked configuration, six of the slots were used for Fibre Channel storage adapters, one was used for an IB adapter and one was unused. Typical business-critical queries that require large sequential scans and large multi-table joins gain significantly from ProLiant DL585's high performance I/O subsystem.

4.1.2 Storage

The HP StorageWorks product set provides strong read performance suitable for DSS applications. It can be configured to support a large number of high-performance, low-cost Fibre Channel connections between the nodes and the arrays.

In the benchmarked configuration, the servers and storage arrays were interconnected using the SAN Switch 2/16s, a 16-port Fibre Channel storage switch that offers 2-GB connectivity for an entry-level SAN, and the ability to grow to a large SAN infrastructure. Features such as redundant power supplies and cooling make it ideal for supporting corporate infrastructure implementations. Each SAN Switch 2/16s had 12 servers and 4 MSA1000s connected to it. The benchmarked configuration can be extended to support higher throughput (adding more storage arrays) and larger scale-out configurations (adding more servers) by using SAN switches with higher port counts or by interconnecting multiple SAN Switch 2/16s devices.

The storage array used in the benchmarked configuration was the MSA1000, a 2-GB Fibre Channel storage system designed for entry-level to midrange SAN environments that provides low-cost, scalable and high-performance storage. With the addition of two more drive enclosures, it can control up to 42 drives, at present allowing for a capacity of 12 TB; more spindles result in higher random throughput. In the benchmarked configuration, four 36-GB, 15-krpm Ultra320 disk drives per MSA1000 were choosen to achive the targeted price/performance balance.

One of the key factors that helped in achieving the high I/O throughput was the HP Drive Array Technology used in the MSA1000. The Drive Array Technology distributes data across a series of individual hard drives to unite these physical drives into one or more higher-performance logical arrays and volumes. Distributing the data allows for concurrent access from multiple drives in the array, yielding faster I/O rates with no overhead on the server. HP Drive Array Technology also supports fault-tolerant configurations that protect against data loss due to drive failures. Depending on the performance and application requirements, logical drives in the array can be set to a different level of fault tolerance. The RAID configuration methods supported by the MSA1000 Controller include:

- No fault tolerance/data striping (RAID 0)
- Drive mirroring and striping (RAID 10)
- Distributed data guarding (RAID 5)
- Advanced data guarding (RAID ADG)

Further data protection can be achieved by assigning one or more online spares to any fault-tolerant array. The Automatic Data Recovery feature rebuilds data onto a spare or replacement drive when another drive in the array fails in the background.

HP Drive Array Technology for MSA1000s supports various stripe sizes. The stripe size of the array refers to the size of the stripes written to each disk. Striping improves performance by splitting up files into small pieces and distributing them to multiple hard disks.

In the benchmarked configuration, each MSA1000 had two RAID0[2] (data striping) volumes of four 36-GB, 15-krpm Ultra320 disk drives hosting data files. Eight MSA1000s had two additional 36-GB, 15-krpm Ultra320, RAID10-protected drives hosting database redo log files. The RAID0 volumes were created using a stripe size of 256 KB; this stripe size was chosen to match with the I/O request that Oracle Database 10*g* issues. Oracle Database 10*g* issues I/O requests in chunks of the Oracle Database parameters multi_block_read_count (number of blocks in one request) and db_block_size (size of a database block). To achieve the best read performance, the array IO/request (stripe size [256 KB)] * number of disks [4]=1 MB) and Oracle database I/O request size (multi_block_read_count [64] * db_block_size [16 KB]=1 MB) were set to be the same.

The TPC-H database was partitioned efficiently to reduce overall amount of data required by the queries. The benchmarked configuration had 96 data volumes, two per MSA1000s. All tables of TPC-H schema, except nation and regions, were evenly distributed over these data volumes. Oracle was configured to use a DOP of 96.

The MSA1000 Array Accelerator (battery-backed cache) can increase performance in database configurations. It performs protected posted-write caching and read-ahead caching, allowing data to be accessed much more quickly than from disk storage. In protected posted-write caching, data is written to the cache memory on the Array Accelerator rather than directly to the drives. The read-ahead cache detects sequential accesses to the array, reads ahead data, and stores the data in the cache until the next read access arrives. If the data is of a sequential nature, the data can be loaded immediately into memory, avoiding the latency of a disk access. The MSA1000s in the benchmarked configuration were configured with 256 MB of Array Accelerator Cache, set to 100% read ahead, because of the read-intensive nature of the queries.

In addition, certain Linux operating system and Oracle Database 10*g* features were enabled to improve I/O throughput. Linux operating system and Oracle Database 10*g* engines were configured to support asynchronous I/O, which allows a process to submit an I/O request without waiting for it to complete. Because the system does not put the process to sleep while the I/O request is submitted and processed, the calling process is able to perform other tasks while the I/O proceeds. To further enhance I/O throughput, two of the HP StorageWorks 2214DC driver parameters were changed from their default. The ql2xintrdelaytimer – the delay in milliseconds posted prior to generating an interrupt – was set to 0 (default is 3 ms), resulting in no interrupt migration. The ql2xmaxqdepth – number of out-

[2] Production systems will usually use RAID10 or higher to ensure fault tolerance. This was not used in the test to reduce costs and improve performance.

standing requests per logical unit – was reduced to 24 from the default value of 32.

4.2 Query Scalability

The Oracle Database 10*g* grid shows very good query scalability. Figure 8 shows the elapsed time for the power and throughput runs of the 1000-GB TPC-H benchmark. Bar 0 shows the elapsed time for the power run, while bars 1–7 show the elapsed time for streams 1–7 of the throughput run. The elapsed time of the power run is 3163 seconds. The elapsed time of the throughput run varies between 17375 and 19618 seconds. Hence, the ratio between the power run and the various streams varies between 5.5 and 6.2, indicating that the streams of the throughput run scales super-linearly. The super-linear behavior occurs because during a throughput run, the system is utilized more efficiently due to multiple streams that can overlap I/O with CPU. Also, queries in the throughput run take advantage of node locality. Instead of running across nodes, which might saturate the inter-connect, queries run locally on one node, reducing IPC to a minimum. The good scalability between the power and throughput run shows that the query locality feature in Oracle Database 10*g* works and significantly increases performance for multiple users issuing queries simultaneously.

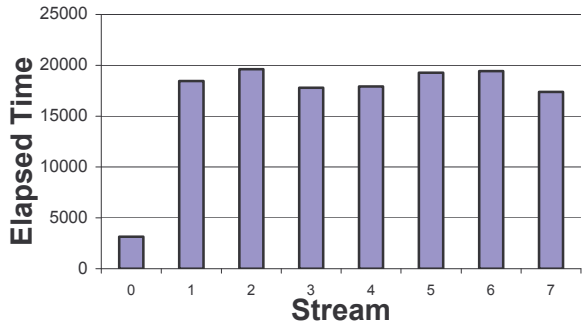

Figure 8: ***Power vs. Throughput Run***

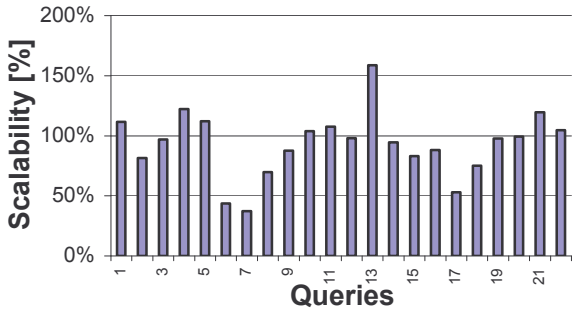

*Figure 9****: Query Scalability***

Figure 9 shows the query elapsed time scalability of all 22 TPC-H queries from an 8- to a 12-node grid. Please note that the elapsed times presented in Figure 9 are not taken from any published TPC result. They were obtained during a lab exercise to obtain empirical data as evidence for grid scalability. Furthermore, the numbers in the x-axis do not correspond directly to the TPC-H query numbers. The y-axis shows the scalability in percent. 100 percent indicates perfect scalability. Any scalability larger than 100% indicates super-scalar behavior. As the chart shows, some queries scale poorly while other queries show super-scalar behavior. Overall, the scalability is at about 90%.

5. Conclusion

In this paper, we have shown that the commodity hardware (industry-standard x86-based) and software components used in this benchmark are proven cost-effective building blocks for large-scale DSS grids. Enterprise customers can now deploy the Oracle Database 10*g* on HP ProLiant hardware with Red Hat Linux and obtain high performance at very low cost compared to an equivalent SMP scale-up configuration.

With the 1000-GB TPC-H publication, we have demonstrated that an IB-enabled grid scales to 12 nodes. However, the architecture and database system support scale-out to many more nodes, including failover nodes. The MSA1000 storage fabric and ProLiant cluster size can be extended to support the higher throughput requirements of larger scale-out configurations by using SAN switches with higher port counts or by inter-connecting them. Each MSA1000 has the capacity of accommodating a maximum of 42, 300-GB disk drives so that the total storage can be expanded by a factor of more than 87 from the configured system without adding extra arrays.

With its 48 AMD Opteron CPUs, the grid demonstrated exceptional performance. The MSA1000-based storage SAN enabled throughput of 7.2 GB/sec; additional throughput is available as disk drives, nodes and faster SAN switches are added to the cluster. More spindles will result in higher random throughput. In real-world terms, this means that customers can extend the warehouse and add data marts within the same centrally managed storage environment. The IB technology, compared with the ability to add I/O throughput, means that the overall configuration can easily reach higher levels of performance.

Customers demand a reasonable return on investment (ROI) from their data warehouses, as well as a low TCO. The industry-standard AMD Opteron-based ProLiant servers help to reduce overall solution cost. Nodes can be added inexpensively to improve performance or provide failover redundancy. Oracle Database 10*g* can then seamlessly integrate the extra nodes into the cluster. The MSA1000 delivers a cost-effective storage subsystem with a high degree of parallelism, redundancy and performance.

HP and Oracle offer powerful grid management tools. HP Systems Insight Manager software provides powerful monitoring and control of HP ProLiant Servers and storage. Oracle Enterprise Manager 10*g,* with its grid control capability, enables all database instances to be managed in par-

allel. This reduces costs and ensures consistent management across the cluster.

Typically, DSS at these sizes provides business-critical services and requires 24x7 availability. The ProLiant servers and MSA storage offer strong availability and reliability capabilities. The MSA1000 supports many fault-tolerant capabilities and features, including advanced RAID protection, online spares and automatic data recovery. ProLiant systems and StorageWorks SANs can be configured with redundant adapters, SAN switches and arrays. Along with Oracle Database 10*g* RAC node failover capability, the benchmark configuration can be extended to better support mission-critical business intelligence applications.

6. Acknowledgement

The authors would like to thank Bryon Georgson, Jay Workman, Anne Kohlstaedt, George Lumkin and Alex Morgan for their valuable input in writing this paper. Special thanks to Mike Nikolaiev, Tracey Stewart and Ray Glasstone for providing guidance and resources to write this paper.

7. REFERENCES

[1] David P. Anderson, Jeff Cobb, Eric Korpela, Matt Lebofsky, Dan Werthimer: SETI@home: an experiment in public-resource computing. Commun. ACM 45(11): 56-61 (2002)

[2] HP ProLiant DL585 Cluster 48P Benchmark Result ID: 104102501: http://www.tpc.org/tpch/results/tpch_result_detail.asp?id=104102501

[3] PRIMEPOWER 2500 10309080: http://www.tpc.org/tpch/results/tpch_result_detail.asp?id=103090803

[4] HP ProLiant DL740 Cluster 32P 3000GB TPC-H 1041030202 http://www.tpc.org/tpch/results/tpch_result_detail.asp?id=1041030202

[5] Ralph Kimball, Kevin Strehlo: Why Decision Support Fails and How To Fix It. SIGMOD Record 24(3): 92-97 (1995)

[6] Ralph Kimball: The Data Warehouse Toolkit: Practical Techniques for Building Dimensional Data Warehouses. John Wiley 1996

[7] E. F. Codd: A Relational Model of Data for Large Shared Data Banks. Commun. ACM 13(6): 377-387 (1970)

[8] World Community Grid: www.worldcommunitygrid.org and www-1.ibm.com/grid/grid_press/pr_1116.shtml

[9] R. Braumandl, M. Keidl, A. Kemper, D. Kossmann, A. Kreutz, S. Seltzsam, K. Stocker: ObjectGlobe, Ubiquitous query processing on the Internet. VLDB J. 10(1): 48-71 (2001)

[10] J. Waldo. The Jini Architecture for Networkcentric Computing. Communications of the ACM, 42(7):76–82, 1999.

[11] M. Stonebraker, P. Aoki, W. Litwin, A. Pfeffer, A. Sah, J. Sidell, C. Staelin, and A. Yu. Mariposa: A wide-area distributed database system. The VLDB Journal, 5(1):48–63, January 1996.

[12] L. Haas, D. Kossmann, E. Wimmers, and J. Yang. Optimizing queries across diverse data sources. In Proc. of the Conf. on Very Large Data Bases (VLDB), pages 276–285, Athens, Greece, August 1997.

[13] Vanja Josifovski and Tore Risch. Integrating heterogeneous overlapping databases through object-oriented transformations. In Proc. of the Conf. on Very Large Data Bases (VLDB), pages 435–446, Edinburgh, GB, September 1999.

[14] Sun Microsystems, http://www.sun.com/servers/highend/sunfire_e25k/index.xml