A Hybrid Shared-nothing/Shared-data Storage Architecture for Large Scale Databases

Huaiming Song †, Xian-He Sun †, Yong Chen ‡
†*Department of Computer Science, Illinois Institute of Technology, Chicago, IL 60616, USA*
‡*Department of Computer Science, Texas Tech University, Lubbock, TX 79409, USA*
huaiming.song@iit.edu, sun@iit.edu, yong.chen@ttu.edu

***Abstract*—Shared-nothing and shared-disk are two widely-used storage architectures in current parallel database systems, and each of them has its own merits for different query patterns. However, there is no much effort in investigating the integration of these two architectures and exploiting their merits together. In this study, we propose a novel hybrid shared-nothing/shared-data storage scheme for large-scale databases, to leverage the benefits of both shared-nothing and shared-disk architectures. We adopt a shared-nothing architecture as the hardware layer and leverage a parallel file system as the storage layer. The proposed hybrid storage scheme can provide a high degree of parallelism in both I/O and computing, like that in a shared-nothing system. In the meantime, it can achieve convenient and high-speed data sharing across multiple database nodes, like that in a shared-disk system. The hybrid scheme is more appropriate for large-scale and data-intensive applications than each of the two individual types of systems.**

I. Introduction

Shared-nothing and shared-disk are the two most common storage architectures of parallel databases in the past two decades. In a shared-nothing system, data is partitioned [1][2][3] into several subsets and each node keeps one in its native disks. In a shared-disk system, data is stored in a large centralized storage. Generally, the shared-nothing systems provide a high degree of parallelism for both I/O and computing, while the shared-disk systems can provide data sharing between multiple nodes[1]. The typical shared-nothing systems include IBM DB2 UDB and Mysql Cluster, and shared-disk systems include Oracle 10g RAC products.

Parallel file systems (PFS), such as Lustre[4], PVFS2[5], GPFS[6], are widely used for high I/O performance. Compared with parallel database architectures, the hardware structure of a PFS is similar to the shared-nothing systems, but also it provides a single namespace. Inspired by the design of PFS, we propose a novel hybrid storage scheme for large-scale data processing, by integrating parallel database with parallel file system techniques together. It adopts shared-nothing for the upper layer database instances, but also provides data sharing through a lower-layer PFS.

II. System Design

The proposed hybrid storage scheme can leverage the advantages of both the shared-nothing hardware structure and the shared-data facility of parallel file systems. Figure 1 illustrates the system architecture. It adopts shared-nothing on the hardware layer, but adds a PFS on top of scattered disks to provide data sharing capability. Each node runs a database instance, and serves as both a file server and an I/O client of the PFS. In order to have the merits of both shared-nothing and shared-disk systems, we introduce two data access modes: global mode and local mode in our new architecture.

- **Global Mode**: each database node accesses data via a global file in the PFS. Here a global file is a file represented in the PFS.
- **Local Mode**: each database instance accesses data via a local file on its native disks. Here a local file is a part of a global file in the PFS.

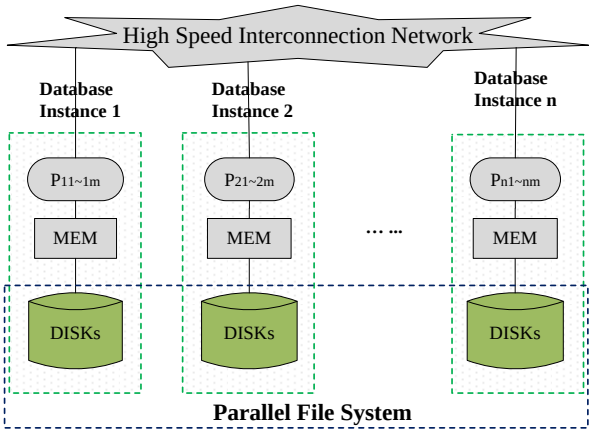

Fig. 1. Architecture of the Hybrid Parallel Database Storage Scheme

Data Organization. We adopt a stripe size of the PFS equaling to the page size of a table, to distribute the relational table across database nodes. We also reserve a small portion in each page for updating records with variable size.

Partitioning. All existing data partitioning methods can be applied in the hybrid system. Moreover, data partitions can be placed either in vertical(each partition placed on one node) or horizontal(each partition striped across all nodes) manner.

Query processing. The hybrid system can simplify query execution by shifting data migration details to the PFS layer. For example, during the execution of multi-join queries, the merge-and-redistribution processes can be simplified.

Transaction processing. A distributed transaction in a shared-nothing system can be converted to a single transaction

TABLE I
DATA ORGANIZATIONS OF TPC-H TABLES

Table Name	Data Size	Strip Size	Reserved(%)
Customer	2.43 GB	64 KB	5
Lineitem	72.42 GB	8 MB	0
Nation	2224 Bytes	–	–
Orders	16.36 GB	8 MB	0
Part	2.42 GB	64 KB	5
Partsupp	11.32 GB	8 MB	0
Region	389 Bytes	–	–
Supplier	142 MB	64 KB	10

conducted by any one database instance, simplifying the transaction control mechanics.

In the proposed hybrid system, since each database instance can access tables in both global and local modes during query processing, it is much more appropriate for applications with varieties of query patterns. If the query executors work in local data access mode, it can get high degree of parallelism in both I/O and computing, which is like that in a shared-nothing system. While working in global data access mode, it can easily share data among different nodes, and this manner is exactly like that in a shared-disk system. Which data access mode to use in a query execution plan is determined by system query optimizer, based on the costs analysis of the two data access modes. With high-speed interconnection techniques, the proposed storage scheme can obtain the advantages of both shared-nothing and shared-disk systems.

III. PRELIMINARY EXPERIMENTS

We designed experiments to verify benchmark TPC-H[7] tables striped by record pages. We employed PVFS2 as the underlying PFS. Our experiment platform was a 16-node SUN cluster, and each node equipped with 4X InfiniBand interconnection. The data scale of TPC-H was 100GB. Table I shows the data stripe sizes and reserved percentages. For each table, the record page size was equal to the stripe size. Table *nation* and *region* were not striped because the data size was too small. The results demonstrate that it is feasible to stripe tables in the proposed hybrid system.

Figure 2 shows the execution time before *'JOIN'* of Query Q(shown below).

```
select   l_orderkey, l_partkey, l_suppkey,
         l_extendedprice * ( 1 - l_discount ) -
         ps_supplycost * l_quantity   profit
from     lineitem, partsupp
where    l_partkey = ps_partkey
         and l_suppkey = ps_suppkey
         and l_commitdate between(t1,t2);
```

We scaled the total data size according to the number of database nodes, so each node had the same data size on average (table *lineitem* had 12 million rows of records and table *partsupp* had 1.6 million rows for each node). From the results, we can observe that the proposed hybrid scheme can achieve much higher performance in all system scales, about 82% higher on average.

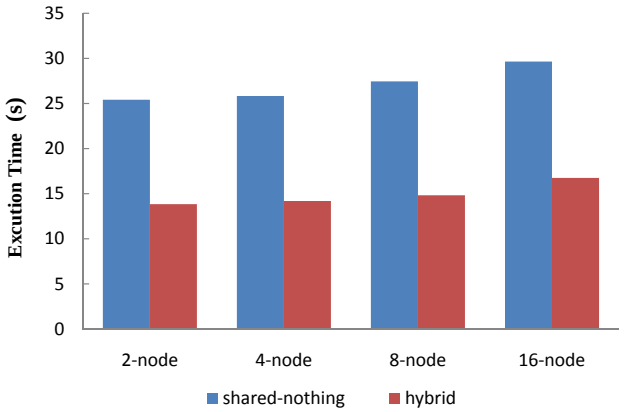

Fig. 2. Execution Time before 'JOIN' of Query Q

IV. CONCLUSION AND FUTURE WORK

We propose a novel hybrid shared-nothing/shared-data storage architecture for large-scale and data-intensive applications in this study. We first present an overview of the system architecture, and then describe the methodology of data organization, data partitioning, query and transaction processing in the proposed system. We compare the proposed system with existing parallel database architectures and conduct experiments to verify our design. The proposed scheme can have the merits of two major parallel database architectures, and it is much more appropriate for applications with complex query patterns. The analytical and experimental results have shown the feasibility of the hybrid architecture.

In the future, we plan to work on detailed design and implementation of the hybrid system, including data partitioning and query processing methods.

ACKNOWLEDGMENT

The authors are thankful to Dr. Rajeev Thakur and Samuel Lang of Argonne National Laboratory for their help and suggestions. This research was supported in part by National Science Foundation under NSF grant CCF-0621435 and CCF-0937877.

REFERENCES

[1] D. DeWitt and J. Gray, "Parallel Database Systems: the Future of High Performance Database Systems," *Commun. ACM*, vol. 35, no. 6, pp. 85–98, 1992.

[2] M. Mehta and D. J. DeWitt, "Data Placement in Shared-nothing Parallel Database Systems," *The VLDB Journal*, vol. 6, no. 1, pp. 53–72, 1997.

[3] D. D. Chamberlin and F. B. Schmuck, "Dynamic Data Distribution (D3) in a Shared-nothing Multiprocessor Data Store," in *VLDB '92: Proceedings of the 18th International Conference on Very Large Data Bases*. 1992, pp. 163–174.

[4] "High-performance Storage Architecture and Scalable Cluster File System," Lustre File System White Paper, December 2007.

[5] I. F. Haddad, "PVFS: A Parallel Virtual File System for Linux Clusters," *Linux Journal*, p. 5, 2000.

[6] F. Schmuck and R. Haskin, "GPFS: A Shared-disk File System for Large Computing Clusters," in *FAST '02: Proceedings of the 1st USENIX Conference on File and Storage Technologies*. Berkeley, CA, USA: USENIX Association, 2002, p. 19.

[7] Transaction Processing Performance Council (TPC) Website. [Online]. Available: http://www.tpc.org/tpch/default.asp