Is DeWitt Wrong? We Have Evidence that Parallel Databases Do Not Scale.

But Are These Problems Fundamental or Simple Engineering Hurdles?

Daniel Abadi
dna@cs.yale.edu

1. POSITION PAPER

The amount of data that needs to be stored and processed by analytical database systems is exploding. This is partly due to the increased automation with which data can be produced (more business processes are becoming digitized), the proliferation of sensors and data-producing devices, Web-scale interactions with customers, and government compliance demands along with strategic corporate initiatives requiring more historical data to be kept online for analysis. It is no longer uncommon to hear of companies claiming to load more than a terabyte of structured data per day into their analytical database system and claiming data warehouses of size more than a petabyte [6, 1].

Given the exploding data problem, many people believe that parallel databases will soon become the only reasonable solution for performing data analysis. This is due to the fact that parallel database systems partition data across a collection of independent, possibly virtual, machines, each with local disk and local main memory, connected together on a high-speed network, and automatically parallelize SQL queries over these partitions. Since data analysis workloads tend to consist of many large scan operations, multidimensional aggregations, and star schema joins, much of the needed parallelization is fairly straight-forward.

Parallel databases have been proven to scale really well into the tens of nodes (near linear scalability is not uncommon). However, there are very few known parallel databases deployments consisting of more than one hundred nodes, and to the best of our knowledge, there exists no published deployment of a parallel database with nodes numbering into the thousands. We believe that this is due to the fact that parallel database systems, as engineered today, cannot scale to such a large number of nodes.

There are a variety of reasons why we believe parallel databases generally do not scale well at this level. First, parallel databases tend to be designed with the assumption that failures are a rare event and restart queries if any of the nodes involved in query processing fails. Yet, given the proven operational benefits and resource consumption savings of using cheap, unreliable commodity hardware to build a shared-nothing cluster of machines, and the trend towards extremely low-end hardware in data centers [5], the probability of a node failure occurring during query processing is increasing rapidly. This problem only gets worse at scale: the larger the amount of data that needs to be accessed for analytical queries, the more nodes are required to participate in query processing. This further increases probability of least least one node failing during query execution. Google, for example, reports an average of 1.2 failures per analysis job [3]. If a query must restart each time a node fails, then long, complex queries are difficult to complete.

Second, parallel databases generally assume a homogeneous array of machines, yet it is nearly impossible to get homogeneous performance across hundreds or thousands of compute nodes, even if each node runs on identical hardware or on an identical virtual machine. Part failures that do not cause complete node failure, but result in degraded hardware performance become more common at scale. Individual node disk fragmentation and software configuration errors also can cause degraded performance on some nodes. Concurrent queries (or, in some cases, concurrent processes) further reduce the homogeneity of cluster performance. On virtualized machines, concurrent activities performed by different virtual machines located on the same physical machine can cause 2-4% variation in performance [2]. Parallel databases tend to divide the amount of work needed to execute a query equally amongst the nodes in a shared-nothing cluster, and thus are vulburable to the danger that the time to complete the query will be approximately equal to time for the slowest compute node to complete its assigned task. A node observing degraded performance thus has a disproportionate affect on total query time.

Third, until recently, there have only been a handful of applications that required deployment on more than a few dozen nodes for reasonable performance, so parallel databases have not been tested at larger scales, and unforeseen engineering hurdles await.

We plan to look at the scalability issues of parallel databases is more detail, and present some initial results we've been running on Amazon's Web Services platform (EC2) that show some of the scalability limitations of parallel databases. We will then discuss whether these issues are fundamental or just engineering challenges that future versions of parallel databases will need to overcome. This will lead to our revealing our position on the MapReduce vs. Parallel Database debate [4].

2. REFERENCES

[1] http://www.sybase.com/detail?id=1054047.

[2] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield. Xen and the art of virtualization. In *Proc. of SOSP*, pages 164–177, 2003.

[3] J. Dean and S. Ghemawat. MapReduce: Simplified Data Processing on Large Clusters. In *OSDI '04*, pages 137–150, 2004.

[4] D. DeWitt and M. Stonebraker. MapReduce: A major step backwards. DatabaseColumn Blog. http://www.databasecolumn.com/2008/01/mapreduce-a-major-step-back.html.

[5] J. Hamilton. Cooperative expendable micro-slice servers (cems): Low cost, low power servers for internet-scale services. In *Proc. of CIDR*, 2009.

[6] C. Monash. The 1-petabyte barrier is crumbling. http://www.networkworld.com/community/node/31439.