A Preliminary Study of Scalability of TCP/IP based clusters under Database Workloads

Krishna Kant

Enterprise Technology Labs
Intel Corporation

Abstract. In this paper we study the scalability of non-partitioned, clustered database management systems as a function of inter-process communication (IPC) latency and number of nodes. It is assumed that the clustered DBMS has a fully shared IO subsystem and multiversion concurrency control over the data in various buffer caches. The cluster interconnect fabric is assumed to be TCP/IP over Ethernet with and w/o hardware offload. The main contribution of the paper is to shed some light on the scalability of DBMS workloads in a scaleout environment as a function of number of nodes and interconnect latencies.

1 Introduction

In the e-business environment, mid-tier and backend applications have traditionally been implemented on SMPs (symmetric multiprocessors) because of its easier programming model and efficient inter-process communication (IPC). However, with the emergence of high bandwidth, low-latency cluster interconnect technologies, there is a move afoot towards clustered implementations. In particular, the availability of cost effective 10 Gb/sec Ethernet networking solutions along with hardware offloaded TCP/IP could make clustered implementations even more attractive. In this paper we examine the scalability clustered implementations for DBMS systems since such systems are significantly impacted by IPC overhead and latency. We assume a shared disk type of clustered DBMS (such as the Oracle 9i/10g product) and a TPC-C like workload (http://www.tpc.org/tpcc/default.asp). In order to avoid complexities and idiosyncracies of actual systems, the modeling in this paper is not intended to project performance for any real system; its purpose is merely to study a simple model of clustered DBMS based on a limited set of measurements.

Although several papers in the literature have discussed IPC performance issues, much of the work is concentrated on the high performance computing (HPC) side rather than the commercial workloads. There are some industry papers that show substantial benefits of low latency interconnect technologies on application performance [1], but no performance models or sensitivity analyses are presented.

It is well known that end-to-end IPC latencies via the traditional TCP/IP over Ethernet stack can be almost an order of magnitude higher than other

specialized fabrics such as Myrinet, QsNet, IBA, etc. [1]. A detailed discussion of these along with a study of performance benefits of HW TCP offload for front-end servers is contained in [4], and will be omitted here. Most of these inefficiencies can be addressed by using the Virtual Interface Architecture [VIA] like interface [3] and efficient fast-path processing. The former is supported by the RDMA (remote DMA) protocol [8] which is gaining widespread acceptance. We assume that RDMA/TCP offload is performed in a programmable engine which we call as *packet processing engine* (PPE). This PPE can be located in multiple places in the platform as reported in [4]; however, to avoid clutter, we shall consider only the "north-bridge" implementation that participates in the processor coherence protocol.

2 Clustered Database Overview

Oracle 9i/10g presents a premier example of clustered DBMS and is used as the representative clustering mechanism in this paper. In this architecture, also known as real application cluster (RAC), all nodes share a common disk subsystem that holds the entire database [6]. That is, no partitioning of the database among nodes is required for clustered operation. For efficient access, each node may cache portions of the data or indices in its main memory (normally called "buffer cache"). If a node requires data that is not present in its local buffer cache, it checks if the data is available in the buffer cache of another node and if not, it initiates the disk IO. Given a high BW, low latency and low overhead interconnection fabric, this technique can substantially reduce the IO overhead and thereby improve the scalability of the cluster.

RAC maintains a distributed directory indicating location and status of all database blocks available in various buffer caches. The directory information is migrated dynamically depending upon the data access pattern so that the directory entry is resident at the most frequently used node.

RAC uses *multiversion concurrency control* (MCC) to achieve high scalability [2] where each lockable entity (assumed to be a page here) carries with it the undo log and the version numbers. Thus, a read transaction does not need any locking since it can always get at the correct version of the page. Also, dirty data can be shared among nodes directly. The major additional cost of MCC is in heavier duty directory management and maintaining cascading undo logs that must be carried around. Of course, write locking is still required. In the model, the choice of values for the various RAC related parameters are somewhat arbitrary, since detailed measurements quantifying these are not available. This is adequate since the purpose of the paper is merely to illustrate scalability rather than do performance projections of an actual system.

For the workload, we used TPC-C, which is a popular benchmark for studying on-line transaction processing (OLTP). TPC-C is usually considered an inappropriate benchmark for clustering purposes since it is possible to partition TPC-C database such that the IPC traffic between nodes becomes negligible. However, note that in the RAC context, the database is *not* partitioned. We also assume

that the query processing itself is not partitioned among nodes, although it is reasonable to assume intelligent transaction management/scheduling to take advantage of already cached data in various nodes. In this case, TPC-C becomes a reasonable clustered database workload. These assumptions also allow us to use existing non-clustered TPC-C measurements to calibrate our model.

We assume that all TCP connections used for IPC are persistent, so that connection setup/teardown overhead or latencies do not come into play for IPC. RDMA does require pre-registration, pre-pinning and exposure of user buffers. Registration and pinning require a Kernel call, which is expensive. Buffer exposure requires an explicit message exchange. We assume that control message buffers are registered and pinned at the time of thread creation and exposed to the directory node once per transaction. For the IPC data messages, we assume we assume that both pinning and exposure are done on the basis of adjustable but small "windows" explicitly exposed for every data transfer. Note that buffer exposure message are themselves IPC messages and need to be accounted for.

3 A simple performance model

The major performance modelling exercise is to accurately estimate the impact of IO (IPC and disk) on the workload as a function of cluster size. To this end, we note that any IO has two major performance impacts:

1. Increased "path-length" (instruction count) of IO handling, which results in *increased utilization* of the host processor.
2. Increased data retrieval latency which leads to *increased stalls* on the host processor. Stalls happen whenever the communication latency cannot be hidden by using multiple threads/processes.

The cluster size primarily affects the shared content (and hence locking & synchronization frequency), amount of management information, and locality properties. Because the available measurements are only for small clusters, we have assumed certain functional behaviors with respect to number of nodes which remain to be validated.

In a typical benchmark performance context, one is usually interested in the achieved throughput when the CPU utilization is as close to 100% as possible. Assuming a multithreaded environment, the effective latency per transaction includes the following components.

1. Latency corresponding to basic per transaction path-length excluding disk IO and IPC, denoted L_{comp}.
2. IPC related latency. This includes code latency L_{eph} and the unhidable latency L_{uh} for each communication.
3. Disk IO related latency. This includes code latency L_{dkc} and the unhidable latency L_{uh} for each IO.

Here, the estimation of the unhidable latency L_{uh}, in turn, requires the estimation of the overall thread stall time, denoted L_{avg}. Let N_{ipc} and N_{disk} denote,

respectively, the number of IPC communications and disk IOs per transaction. Also, let L_{ipc} and L_{disk} denote thread stall times per IPC and disk IO operation respectively. Then L_{avg} can be computed as the following weighted sum:

$$L_{avg} = (L_{ipc}N_{ipc} + L_{disk}N_{disk})/(N_{ipc} + N_{disk}) \quad (1)$$

The estimation of the crucial parameters introduced above (including unhidable latencies) is omitted here due to lack of space and may be found in [5]. With this, the total per transaction latency L_{tot} can be estimated as:

$$L_{tot} = L_{comp} + N_{ipc}[L_{eph} + L_{uh}] + N_{disk}[L_{dkc} + L_{uh}] \quad (2)$$

With this, the achievable throughput *per node* is given by simply $1/L_{tot}$.

Although absolute achievable throughput is interesting, a more important performance metric is *cluster efficiency*, which we define as ratio of nodal throughputs in the clustered and unclustered cases. The unclustered throughput can be obtained by independent (and much more mature) performance projection models; however, we express the unclustered throughput in the same framework as the clustered throughput. For brevity, the details of this calculation are omitted here.

4 Sample Modelling Results

The model calibration proved to be an ardous process because of the lack of consistent set of detailed measurements. The TCP offload related parameters were obtained using a prototype system which uses one processor in a SMP system as the TCP engine. Even here, RDMA related estimates are speculative since currently we do not have a working RDMA prototype. A further difficulty is that the modeled system (assumed to be a platform in 2005-2006 timeframe) is different from the measured system and a translation of parameters was required. The available TPC-C results were are also sketchy and were available only for 1, 2 and 4 node systems. In view of these deficiencies, the results must be treated as merely indicative of trends, rather than as actual achievable results.

For the results presented here, we assume a "rated utilization" model where the utilizations of various resources have been fixed at predetermined values. The practical interpretation of this is that the system uses just enough units (NICs, disk adapters, disk drives, etc.) to keep the utilization almost constant regardless of the cluster size or configuration. Apart from simplicity, the main motivation for this approach is that it does not color the cluster scalability results with system configuration issues. In particular, we assume a rated utilization of 60% for chipset, IO bus, disk adapter, NIC, and switch ports, and 30% for disk drives. All switches were assumed to be 16-port layer-2 and the topologies attempted to distribute the switch-port usage evenly. For the results shown here, we consider the following 3 cases with respect to the IPC transport implementation.

1. Kernel based software RDMA/TCP implementation. TCP parameters are calibrated based the study in [4], but RDMA calibration is speculative. This case represents the lower bound on cluster performance.

2. Memory control hub (MCH) based hardware RDMA/TCP engine, which was calibrated based on the current prototype and internal models of such an engine.
3. An "ideal" case characterized by (a) zero path-length for the PPE, (b) zero port-redirection latency in the switches, and (c) fully optimized host-PPE interface (i.e., no interrupts, no scheduling delays, etc.). This case represents an (almost unachievable) upper bound on performance.

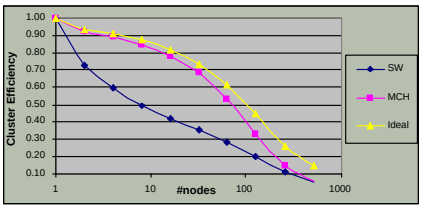

Fig. 1. Cluster Efficiency vs. cluster size

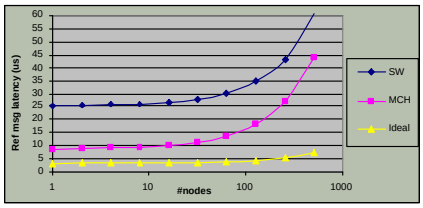

Fig. 2. 256B msg latency vs. cluster size

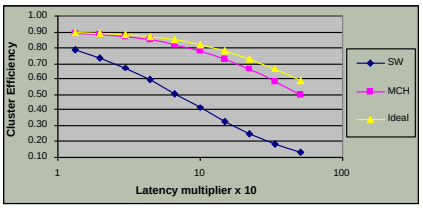

Fig. 3. Latency sensitivity w/ 16 nodes

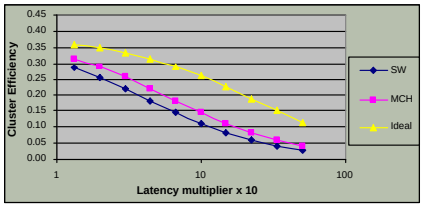

Fig. 4. Latency sensitivity w/ 256 nodes

Figure 1 shows the cluster efficiency as a function of number of nodes (N_{nodes}). The efficiency starts out at 1.0 and generally decreases with N_{nodes}. The efficiency for Ideal and MCH cases remains high with small N_{nodes} but eventually rolls off due to the overhead of managing nodes, duplication in buffer contents of various nodes, and multiple switches in the path. Note that the SW TCP shows a steadily decreasing efficiency because the substantial IPC overhead cannot compensate for reduced IO. In fact, even a 2-node cluster shows an efficiency of only 73%.

It has been well recognized in the literature that IPC latency limits the *scalability*, i.e., the maximum cluster size that one could reasonably build. In the absence of a standardized definition, let's say that scalability refers to the size at which the efficiency drops to 50%. With this definition, Figure 1 shows that HW RDMA allows scalability to > 64 nodes whereas SW RDMA limits it to 8 nodes. The interesting point to note is that even the Ideal curve provides a scalability of only 100 nodes. That is, low IPC latency can only do so much for the scalability – at some point myriad issue of platform latencies, OS overhead, and application interface take over.

Figure 2 attempts to show end-to-end IPC latency $L_{ipc}^{(tot)}$ as a function of number of nodes. For this, we use a *reference message size* of 256B. Figure 2

shows that for small clusters, SW TCP provides a latency of 27 μs whereas HW TCP has a latency of only about 10 μs.[1] The Ideal case shows another factor of 3 reduction in latency (i.e., 3.5 μs). This last result is interesting since it shows that using very fast PPE's/switches still leaves various chipset, wire and OS latencies which may be substantial.

Figures 3 and 4 attempt to directly show the sensitivity of cluster efficiency to end-to-end IPC latency for $N_{nodes} = 16$ and 256. Here, the latency multiplier= 10 corresponds to normal latency and the latency for each successive point is 1.5 times that for the previous point. As expected, the latency sensitivity is small initially, increases in the middle, and then decreases eventually. The major differences between the 3 cases are due to the distinction between code and non-code latencies. Code latencies directly contribute to worse performance since they amount to a change in path-length, whereas non-code latencies can be hidden to large extent by multiple threads. This explains why SW TCP shows the highest sensitivity whereas the Ideal shows the least.

5 Conclusions and Open Issues

In this paper we studied the scalability of Ethernet based clustered non-partitioned DBMS with multiversion concurrency control. The results show that in such an environment, an end-to-end loaded latency of 10 μs is adequate to scale the cluster to 100 nodes and still achieve good cluster efficiency. Although the results are preliminary due to novelty of technologies and other practical challenges, we believe that our model provides a number of interesting insights into cluster performance and scalability that have hitherto been unavailable to the researchers. A more detailed setup and measurement work is currently underway.

References

1. B. Benton, "Infiniband's superiority over Myrinet and QsNet for high performance computing", whitepaper at www.FabricNetworks.com.
2. P.A. Bernstein and N. Goodman, "Multiversion concurrency control — theory and algorithms", ACM Trans on Database Systems ., 8(4):465–483, December 1983.
3. D. Dunning, G. Regnier, et. al., "The virtual interface architecture - a protected, zero-copy user-level interface to networks", IEEE Micro, March 1998, pp66-76.
4. K. Kant, "TCP offload performance for front-end servers", to appear in proc. of GLOBECOM 2003, Dec 2003, San Francisco, CA.
5. K. Kant, "Scalability of TCP/IP based clusters under Database Workloads", Full paper available at kkant.ccwebhost.com/download.html.
6. T. Lahiri, V. Srihari, et. al., "Cach Fusion: Extending shared disk clusters with shared caches", Proc. 27th VLDB conference, Rome, Italy 2001.
7. J. Liedtke, K. Elphinstone, et. al., "Achieved IPC performance", Proc. of 6th workshop on hot topics in operating systems, May 1997, Chatham, MA.
8. J. Pinkerton, `www.rdmaconsortium.org/home/The_Case_for_RDMA020531.pdf`

[1] Recall that these numbers are for 2006 platforms; for current platforms, SW TCP latencies are more like 90 μs and HW offloaded latencies estimated to be 30 μs.