IPF Performance on Windows 2003 and SqlServer

Larry Kemp
ISS Redmond Performance Group Manager
Hewlett-Packard Company

HP WORLD 2003
Solutions and Technology Conference & Expo

IPF Performance on Windows

- Progress to date: rx2600, rx5670 and Superdome
- Measuring Performance
  - The performance benchmarks
  - Defining scalability
- What's different about IPF performance
  - IPF in general
    - New instruction set, word and cache line sizes
    - Direct interactions with chipset
    - Big memory, 64-bit addressibility
  - Superdome
    - 64 processors, 512gb of memory
    - NUMA

TPCC Non-Clustered Top Five Performance List as of June 24, 2003

www.tpc.org

1		hp superdome Client/Server	707,102	9.13 US $	10/23/03	Microsoft SQL Server 2000 Enterprise Ed. 64-bit	Microsoft Windows Server 2003 Datacenter Edition	Microsoft COM+	05/20/03
2		IBM eServer pSeries 690 Turbo 7040-681	680,613	11.13 US $	11/08/03	IBM DB2 UDB 8.1	IBM AIX 5L V5.2	BEA Tuxedo 8.0	05/09/03
3		NEC Express5800/ 1320Xc	514,034	11.50 US $	10/22/03	Microsoft SQL Server 2000 Enterprise Ed. 64-bit	Microsoft Windows Server 2003 Datacenter Edition	Microsoft COM+	04/23/03
4		PRIMEPOWER 2000 c/s w 66 Front-Ends	455,818	28.58 US $	02/28/02	SymfoWARE Server Enterp. Ed. VLM 3.0	Sun Solaris 8	BEA Tuxedo 6.5 CFS	08/28/01
5		NEC Express5800/ 1320Xc C/S with Express5800/ 120Re-	433,107	12.98 US $	06/30/03	Microsoft SQL Server 2000 Enterprise Ed. 64-bit	Microsoft Windows Server 2003 Datacenter Edition	Microsoft COM+	02/20/03

Industry Recognized Commercial Performance Measurements

- SPEC consortium for measurement of pure computational performance
  - SPECint measures integer computational performance
  - SPECfp measures floating point performance
  - SPECrate measures multi-processor computational performance
- TPC is the Transaction Processing Performance Council
  - Benchmark submissions are AUDITED and subject to challenge
  - Results specified in performance and price per transaction
  - TPCC is Online Transaction Processing
  - TPCH is Decision Support
  - TPCW is web serving
- SAP is the very large market share application software
  - SAP/SD 2-tier measures application and database performance
  - SAP/SD 3-tier measures pure database performance

Just how good is TPCC?

- Primary metric looked at by industry analysts
- TPCC, according to Larry…
  - Transactions are relatively complex. For instance, there is no simple "lookup a customer number" transaction. So by proportion of work, network overhead is under-emphasized. The evidence for this is in the effect of replacing TCP with VIA: early studies on an 8P showed 3-6% improvement for TPCC, but a 20% improvement for SAP SD.
  - OLTP database transactions represent near worst case scaling. By contrast, SPECrate scales near linearly.
  - Disk IO is very high. That is, TPCC is much more harsh on disk IO than most user workloads.
- TPCC is the only benchmark to take price into account.

Scalability: what is it?

A *scale factor* represents the increase in SMP performance when the number of processors is doubled. Generally, SMP systems scale at a factor of 1.6-1.7x, and in many cases less than that. There are many important corollaries:

- If one system has double the number of processors of another, the difference in processor performance is likely to be the scale factor, not 2x.
- We talk about scale factors specifically in TPCC, which is one of the worst cases of scalability. Applications that do little or no data sharing can potentially be much better.
- It is NOT valid to expect rx5670 performance from a four processor Superdome, nor is it valid to expect a Superdome to perform at sixteen times the performance of an rx5670.

What factors inhibit scaling?

Generally, scaling is restricted by data sharing between threads. Interference from data sharing can be caused by simple cache line exchanges, meaning that one processor wants data that is in another processors cache; or it can be from synchronization overhead that is used to enforce that only one process at a time modifies the data.

- Database buffers, and particularly the log file, are common contention points.
- IO structures within the OS and device drivers, are another common source of contention.
- In theory, the bus structure of the platform could be a bottleneck, but in our TPCC work on the Superdome, that was never the case.

Be careful to expand the full system when planning a scalable system. In particular, make sure that memory expands with the processor count, or more specifically, with the database throughput.

Things that help Scaling

- VIA technology for application server to database server communications: avoids TCP and kernel overhead.
- High performance disk drivers: includes interrupt combining, per processor queues, intelligent DPC/APC handling.
- Lots of memory: can be used to avoid disk IO. Nearly always improves price-performance of high end workloads.
- Applications that are NUMA optimized. (A separate subject.) Windows is NUMA-ized, as is the newest version of SqlServer.

IPF in general: what's different?

- At a high level, not a lot: Windows is Windows
- Compiler Optimizations
  - POGO
  - Significant performance improvements from –o and Pogo
- Large Memory
  - Some advantages in OLTP database environments
  - Functional, and resulting performance improvements in applications that can take advantage of large address spaces. Examples include:
    - SqlServer in DSS environments (large hash tables to avoid random disk access)
    - SAP APO LiveCache
- Native Code

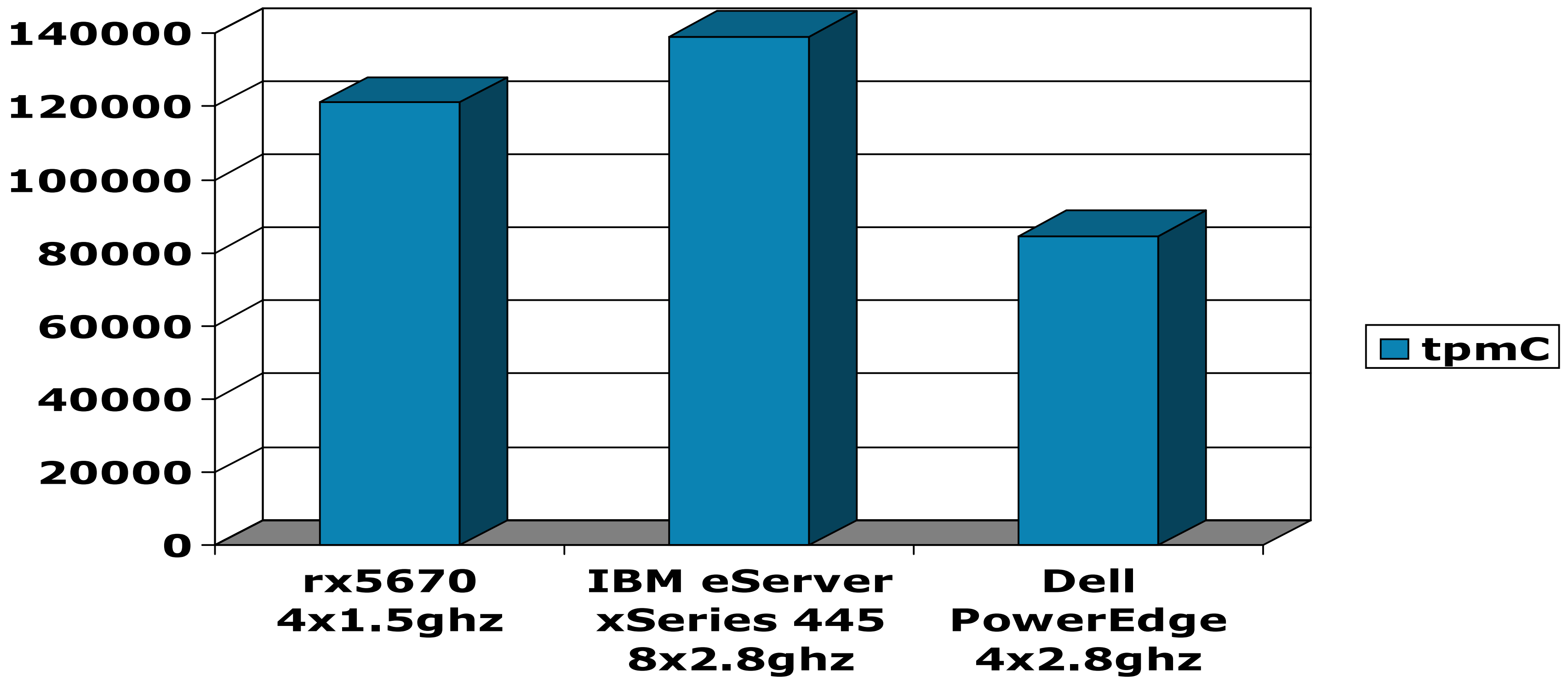
IPF and native code

- “Migration” is an overstatement – technically it is trivial.
- The significant difference is that pointers are 64-bits while (conventional) integers are 32-bits. Applications that use the ANYVAR construct, as used in a very small number of system calls, may be affected. Use of ANYVAR is quite rare.
- Structure sizes change due to pointers being 64-bits.
- If you use spinlocks or interlocked operations, there are now more choices.
- The McKinley/Madison cache line size is 128 bytes; which is bigger than its predecessor IA32 cache line sizes. Generally this is transparent, but could show up as a performance factor if an application shares arrays of structures between threads.

IPF and ia32 compatibility

- SqlServer databases are binary compatible – assuming you use the same storage system, you can simply re-cable storage and “sp_attach_db” to the same databases. Backup and Restore are compatible too.
- Data types, structures and the like are identical.
- Client connections can be IPF or ia32. That is transparent to SqlServer on IPF.
- The very few sites that use extended stored (xp) procedures will need to be re-compiled, but that should affect very few customers.

Mid-Range IPF Performance



HP rx5670, available 8/1/2003, 121065 tpmC @ $4.97/tpmC
IBM eServer xSeries 445, available 12/31/2003, 139153 @ $5.07/tpmC
Dell PowerEdge 6600, available 12/30/2003, 84595 @ $3.84/tpmC
see www.tpc.org

Superdome Windows Performance What’s new?

- Lots of Processors (64)
- Lots of Memory (512gb)
- Cells (16)

Yes, you need to be multi-threaded, but that’s only the beginning…

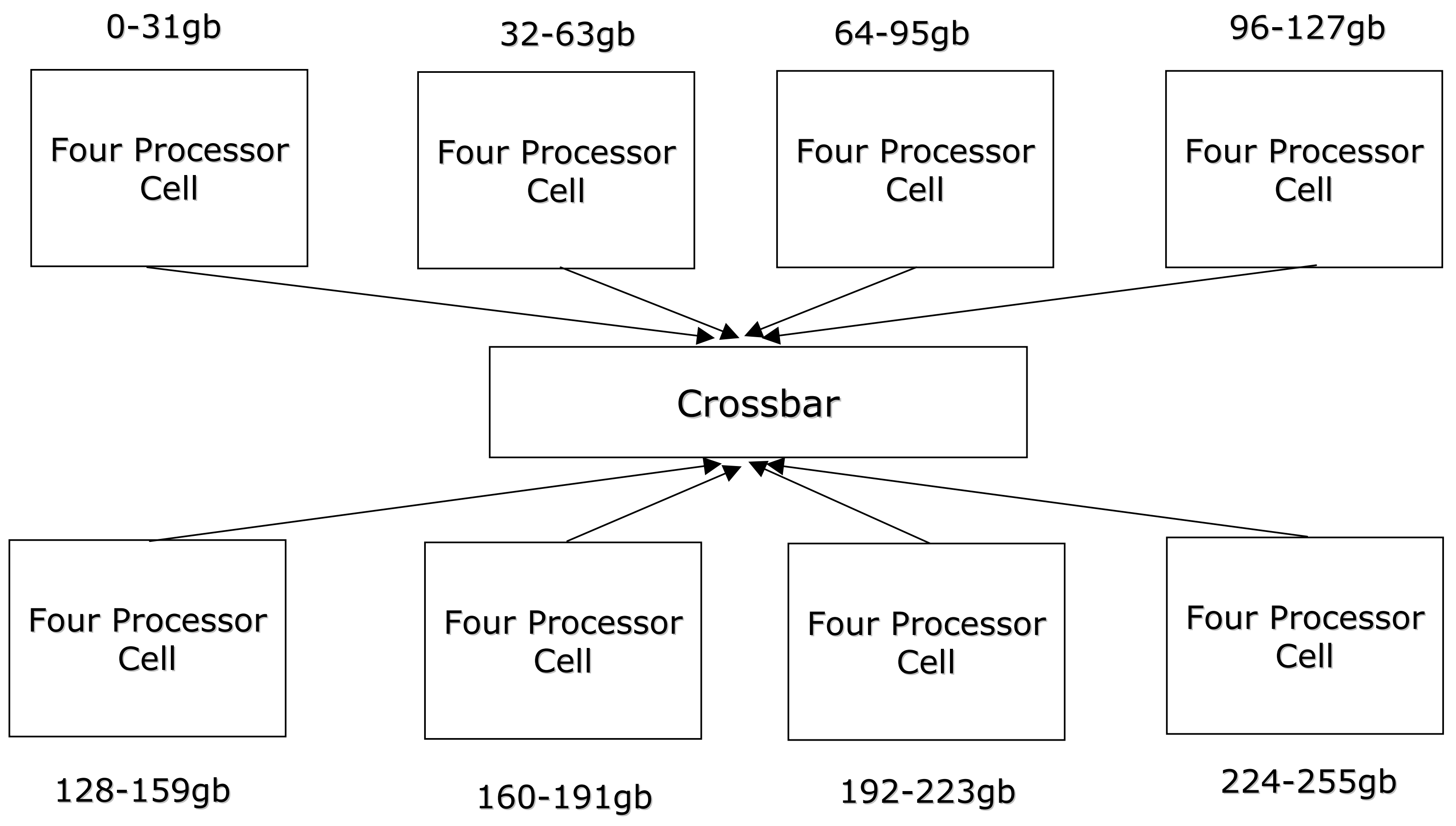
What is a chipset?

The chipset provides memory access to the processor and IO devices. The complicating factor is providing processor cache synchronization.

On smaller systems, like the zx1 chipset that is used in the rx5670, this is a relatively simple technology.

Larger systems, like the Superdome, incorporate a technology commonly referred to as cellular architecture, where each cell contains memory and processors. Cells are connected to one another via a cross-bar. Memory addresses can be assigned such that pages are interleaved across cells, or first in cell 0, cell 1, etc. The latter technique is called cell local memory, and will be described further on.

Superdome Organization



Multi-Threading Basics

```
Struct {
    int             val;
    char            padding[124];
} SharedArray[1000];
```

Static data is shared between threads
Note the relationship between element size and cache line size, with respect to SMP performance

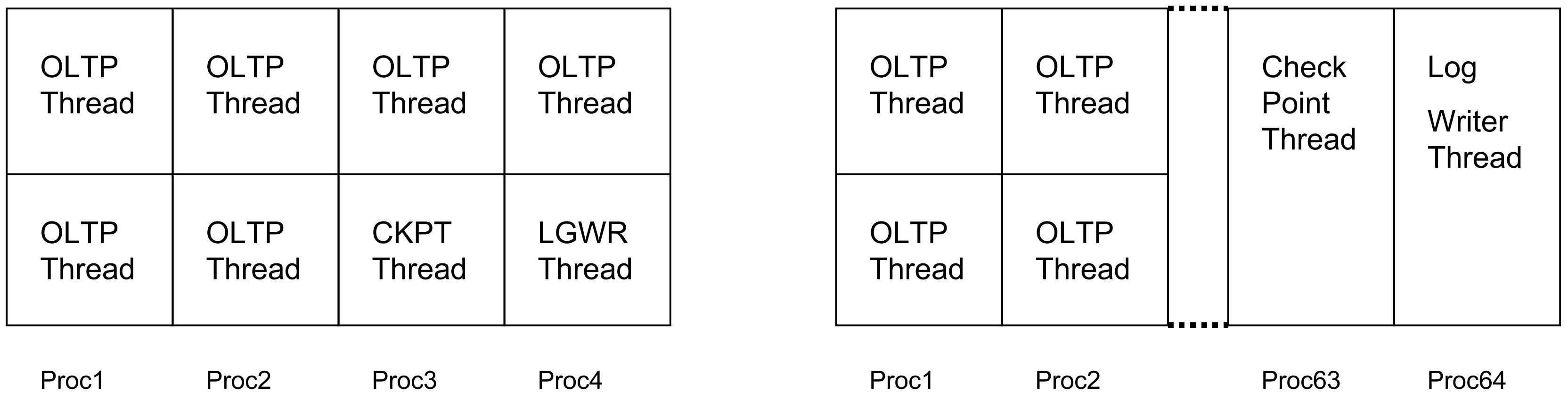
```
void Thread(int i)
{
    int             j;
    for(j=0;j<1000000;j++) SharedArray[i].val=j;
}
```

This is the thread function
It's stack is local

```
int main(int argc, char **argv)
{
    int             j=atoi(argv[1]);
    for(i=0;i<j;i++) CreateThread(…,Thread,…i)
}
```

This is the mainline function

Lots of Processors: the thread starvation problem



- With SMP levels, one fourth or one eighth of a the total system has plenty of processing power. But with one sixty-fourth of the system power in one processor, its possible for a workload to be beyond the processing power of one processor. If this seems bad for 64P, think about 128P.
- We didn't quite have this experience with TPCC, but nearly: both the Checkpoint and LogWriter threads require more than half a processor. Sharing that workload with other threads on the same processor means that some threads will suffer starvation.
- The good news on 64P is that you can give away a processor without much penalty. After all, 1/64th is only 1.5% of the system.
- From a software developers point of view, you need to be very careful of thread-to-thread dependencies. In our case, OLTP threads depend upon the LogWriter thread to empty log buffers. When the LogWriter slows down, the result is system idle time.

Lots of Processors: Interrupt Processing

- IPIs are Inter-Processor Interrupts.
- In a single (non-SMP) implementation, an I/O completion interrupt occurs on the same processor that requested the I/O. On an SMP system, an interrupt may not come back to the same processor that requested the I/O. Therefore, the processor receiving the I/O completion needs to signal the requesting processor. The mechanism for doing so is an IPI.
- How often does this occur, in general? On a 4P system, 75% of the time. On a 64P system, 98.5% of the time. Interrupts generally use less than 3% of processor capacity.
- Given this knowledge, we can dedicate a set of processors to interrupt processing. Provided that the distribution is okay, dedication of processors can be good, since the result is better processor cache locality. That is, interrupt processing doesn't interfere with applications in the use of caches.

Lots of Processors: Semaphores

```
AcquireSpinLock(LockWord);
ManipulateBuffer(Buffer);
ReleaseSpinlock(LockWord);
```

```
AcquireSpinLock(LockWord);
   [spinlock loop]
ManipulateBuffer(Buffer);
ReleaseSpinlock(LockWord);
```

- Spinlock Contention is an issue for all SMP systems. As the number of processors goes up, the probability of contention goes up; at 64P, dramatically.
- A spinlock requires taking ownership of a cache line, and that requires moving the cache line owner. That means that execution of the spinlock memory access itself takes time. And if, as is commonly the case, the cache line containing the lockword is the same as the cache line containing the buffer, then execution of the spinlock will slow down the manipulation of the buffer. This is a best case versus worst case scenario: in the best case (low SMP), taking ownership of the cache line through the spinlock grabs the buffer in the same access, while in the worst case (high SMP) increases the time spent under lock, since every buffer access will become a cache miss, if someone else is contending for the spinlock.
- This is exacerbated on the Superdome, where cache-to-cache accesses take roughly double the time of long memory accesses.

Lots of Processors: Helping out Semaphore Contention

- VIA: used for network communications. Avoids contention associated with shared TCP buffers. We use the Qlogic 2350 host bus adapter for VIA communications.

- Interlocked Intrinsics, e.g. InterlockedAdd, InterlockedCompareExchange: streamlined instruction sequences that act upon a single word.

- Queued Spinlocks: a two part mechanism for implementing spinlocks. The first access is to a global lockword, where a queue of waiters is kept. The second access is to a local lockword, where the requester spins. The price for a queued spinlock that doesn’t wait is two cache-to-cache accesses; the gain for a queued spinlock that is in contention is that the waiter does not interfere with the lock holder. This type of spinlock is used for the Windows Dispatch lock.

Helping out Semaphore contention, continued

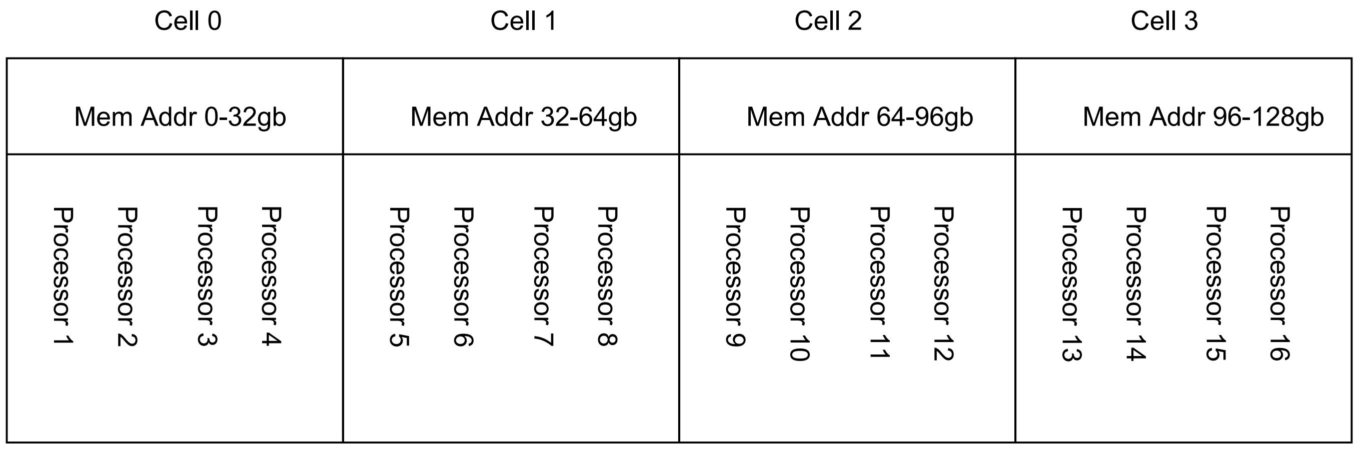
- Per Processor structures and queues: For code that runs in non-preemptive mode, such as SqlServer or any of the high performance device drivers, no locking is required to manipulate structures that are kept on a per processor basis.

- Elimination of false sharing: False sharing refers to two processors inadvertently contending for different parts of the same cache line. One way this can happen is when structures are either not aligned on cache line boundaries, and/or structures that are not multiples of the cache line size. We have also seen it happen through compiler optimization.

Lots of Memory: How do we handle this?

- VirtualAlloc(…) requests that Windows allocate memory, which for security purposes must be zeroed before it can be handed to an application. Do many of these concurrently.
- Some applications, SqlServer among them, allocate static structures based upon the amount of memory available. As an example, SqlServer requires a header for each 8kb page of memory. For 512gb, that is 8gb. Be prepared for such cases.
- Can TLB thrashing be avoided (via large pages) in big memory systems? No.

Cell Local Memory



- Most modern large scale systems use a cell concept. A cell is a unit containing memory and processors. The Superdome declares that there are four processors and thirty-two DIMM slots in a cell.
- Access from a processor to memory within a cell is called local or near memory access, and is generally fast; while access to memory outside the cell is called remote or far memory. In the Superdome remote memory accesses are generally 1.5-2x the latency of cell local memory accesses.
- The trick, of course, is to get most of your accesses to be local.

Just how much effect does NUMA have?

Percentage throughput change depending upon percent of accesses to cell local memory

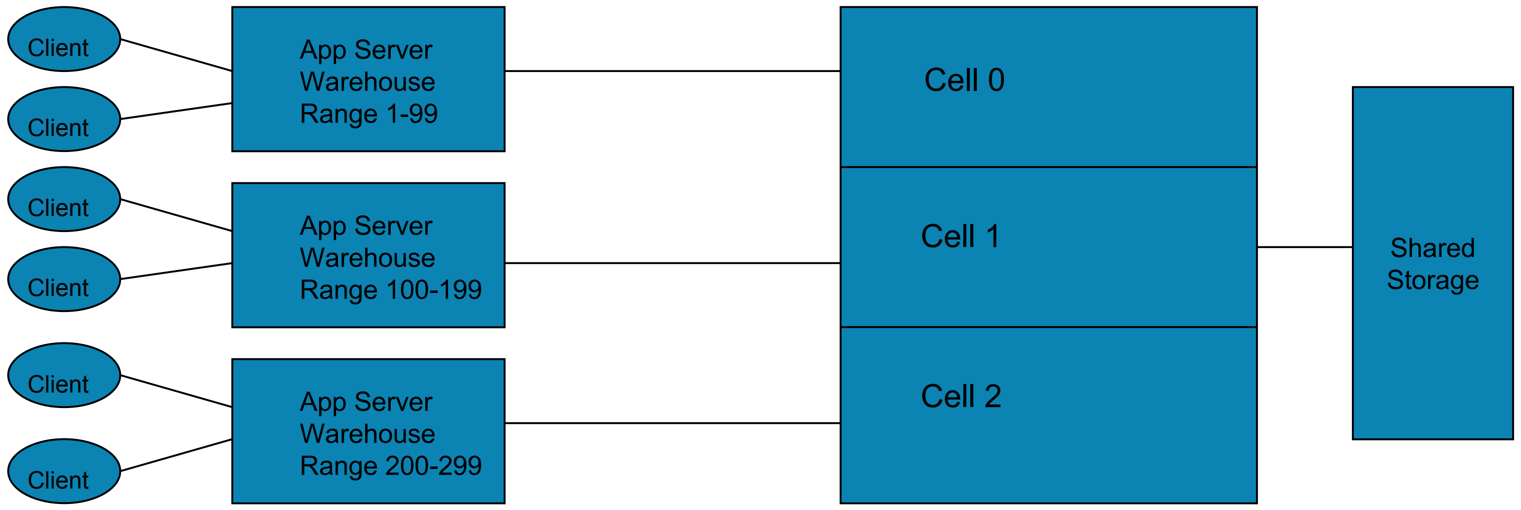
Hit	8-way	16-way	32-way	64-way	128-way
100%	100%	115%	142%	175%	203%
90%	100%	114%	138%	167%	193%
80%	100%	112%	134%	160%	182%
70%	100%	111%	130%	152%	172%
60%	100%	109%	125%	145%	162%
50%	100%	108%	121%	137%	152%
40%	100%	106%	117%	130%	141%
30%	100%	105%	113%	122%	131%
20%	100%	103%	108%	115%	121%
10%	100%	102%	104%	107%	110%
0%	100%	100%	100%	100%	100%

Data is from HP Fort Collins system performance models

NUMA-ized Applications
aka How do I make my application NUMA-ized?

- The Windows system call VirtualAlloc() attempts to allocate memory on the cell containing the processor making the request. The actual allocation happens at page fault time, that is, the first time the page is actually referenced. So one thread, even a global one, can allocate memory, while the threads performing the work will force the actual allocation.
- If you manage your own memory, as SqlServer does, you'll need to have per cell free memory lists.
- If you have data that is sharable between threads, make up a locality scheme, with a goal of having most shared access happen within cells.

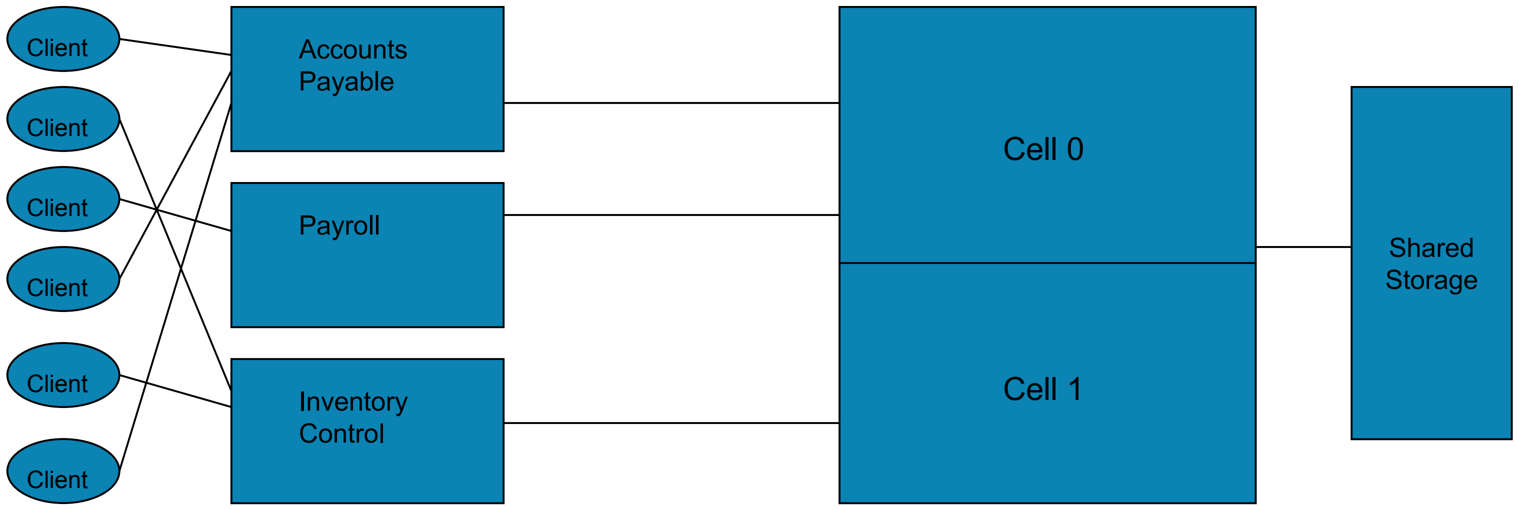
NUMA scheme for TPCC


Client
Client
Client
Client
Client
Client
App Server Warehouse Range 1-99
App Server Warehouse Range 100-199
App Server Warehouse Range 200-299
Cell 0
Cell 1
Cell 2
Shared Storage

- Each client primarily accesses a warehouse, i.e. 90% of accesses are to its home warehouse, the other 10% are random.
- An application server provides access for clients within a home warehouse range.
- VIA networking affinity is used to assign application servers to NUMA cells.
- The clustered index configuration of key tables is used to physically group warehouse ranges together.

As a result, the cell memory is predominantly for the range of warehouses accessed by it's clients.

General NUMA scheme



- Most large scale applications have modules, and each module can be fitted to one or potential more servers.
- In this arrangement, any user has access to any function. Servers are designated by function, although that could easily be based upon usage pattern rather than strict assignment.
- VIA networking affinity is used to assign application servers to NUMA cells.
- Natural separation of tables does effectively take advantage of cell local memory. Some tables are shared between applications, which means they are shared across cells.

Superdome, what’s next?

- Many different customers are testing their own applications.
- We will be working on different workloads.
- We will start working with the SqlServer Yukon release soon.