Unmatched Price Performance Gains Using Distributed Computing, ANSYS Solutions, and HP-MPI

Don Mize

Hewlett Packard

Abstract

Distributed ANSYS with HP's HP-MPI increases price/performance benefits to the CAE customer. Clustering or distributed technology is not new, but improvements in interconnects along with message passing technology, have made clusters a viable price/performance alternative to large SMP machines. Developments in ANSYS, which involve the use of HP-MPI in their distributed product, are taking advantage of this paradigm. This document will explore the history of distributed ANSYS, the history of message passing technology, and investigate the advancing power of computing with HP innovation, choice, and performance. It will then show the evolving price and performance gains and benefits for the ANSYS user over the last few years.

Introduction

In the last ten years, the evolution of distributed computing has drastically changed the landscape of high performance computing. The industry has evolved from the very expensive supercomputers and superservers of Cray, HP, and SGI among others, to the very inexpensive, commodity, high performance clusters of today. Distributed ANSYS along with HP-MPI take advantage of this paradigm to deliver unmatched price/performance to the present day ANSYS customer. However, this combined product just didn't appear overnight. Both D-ANSYS, and HP-MPI have their own separate histories before they were combined to create this high performance solution. Another factor in this is the strong partnership between ANSYS and HP throughout the years. Both have supported the other in implementing leading edge technologies such as this.

The History of HP-MPI

In 1993 a company called Convex Computer Corp., began the early work to port MPICH to the Convex Exemplar Scalable Parallel Processor (SPP) server. From these experiences, Convex decided to create an implementation of MPI from scratch. In 1995, Convex was acquired by HP, and the group responsible for doing this MPI implementation became what is now the HP-MPI group.

In the beginning

In those early days, the group was lead by Paco Romero, who is the source of the information about this early history. He is now a section manager, in HP, for the Workload and Resource Management group. In implementing this new version of MPI, Paco and his group decided to do a multi-protocol version, which would use the fastest 'path' or interconnects available on the node (the term used for a single server within a cluster) or cluster. HP-MPI still works this way today. For example, if one has an application which uses HP-MPI and they are running on a single SMP node, the implementation will automatically use shared memory, since that is the best way to pass messages on a SMP system. Likewise if a customer is using this HP-MPI application on a cluster connected with both TCP/IP and Infiniband interconnects, HP-MPI will choose the Infiniband path since that interconnect has much lower latency (< 6 versus microsec vs. ~45 for TCP/IP), and much higher bandwidth (> 700 MB/sec versus ~110 MB/sec for Gigabit Ethernet).

Evolution of the paradigm.

Originally HP-MPI was a product for the HP-UX machines, starting back with the HP C, J, K, and V-Class servers which came out in the middle to late 1990s. Of course vendors and users could use the open source MPICH or PVM (a beast this author never could really grasp) for their applications. The MPI/MPICH protocol is much easier to implement and also much more transportable than PVM. With HP-MPI, a vendor or a customer could get benefits that they would not get from the MPICH open source implementation. For one they would get active support from Hewlett-Packard. They would also get an implementation that was tuned for performance, whereas if they used standard MPICH, they would have to do the tuning themselves. Also to help with the tuning, HP-MPI has profiling capability that can be to report message sizes and lengths, the ratio of compute versus messaging time, breaking all this down to fairly specific detail. See the following example.

Below are two segments of actual output from the HP-MPI profiler. There is more information than is shown here. The first part below shows the distribution of User or Compute and I/O time versus time spent in MPI communication.

```
--------------------------------------------------------------------------------------------------------------------------

Application Summary by Rank (second):


 Rank     Proc Wall Time      User                      MPI

 --------------------------------------------------------------------------------------

  0        230.572972     228.422853( 99.07%)      2.150120(  0.93%)

  1        230.572980      38.034201( 16.50%)    192.538779( 83.50%)

 --------------------------------------------------------------------------------------
```

The second part here shows a message summary for each process.

```
---------------------------------------------------------------------------------------------------------------------------

          Message Summary by Rank Pair:


 SRank   DRank     Messages      (minsize,maxsize)/[bin]      Totalbytes

 --------------------------------------------------------------------------------------

  0

              1         221                 (0, 5926608)      24995260

                        141                      [0..64]          1424

                         13                    [65..256]          1736

                         28                  [257..1024]         17520

                          9                 [1025..4096]         21616

                          5                [4097..16384]         62564

                          4               [16385..65536]        129588

                          5              [65537..262144]        723040

                          7            [262145..1048576]       3197316

                          8           [1048577..4194304]      14913848

                          1           [4194305..infinity]       5926608
```

```
-------------------------------------------------------------------------------------
  1
              0          935            (0, 3453384)    293650180
                         102                  [0..64]          596
                           1                [65..256]           80
                           7              [257..1024]         4832
                           1             [1025..4096]         3480
                           3           [16385..65536]       126436
                         578          [65537..262144]    150642512
                         236        [262145..1048576]    123878632
                           7       [1048577..4194304]     18993612


-----------------------------------------------------------------------------------------------------------------------
```

Using this data a developer/user can further tune their MPI based applications without any other diagnostic profilers or development tools.

As time progressed and integrated circuit technology improved, you could continuously build smaller machines with greater compute power per CPU. Here Moore's law came into play: Increasing the number of gates per chip while increasing the clock speed. Also as this technology improved, its cost dropped. As Moore's law states, the compute power of processor technology doubles every 15 months. And the Gordon Bell Prize[1] for price/performance demonstrates that the price of processing power halves every 10 months[2]. So you get more computing power for less cost.

Combine this with increasingly higher performing interconnects. You can then take small cheap machines and cluster them together, making high performance server clusters with the same power as SMP systems but at a lower cost. Obviously the SMP systems in general will handle many workloads better than these clusters of small machines. But in the CAE market, where high compute power is a necessity, the clusters are much more cost-effective. Look at Table 1 below which list prices for various HP Integrity and Proliant servers. As the machine gets smaller, the price per processor goes down. Of course there are variations not listed here, but in general the below is true. Also, the cost of interconnects will come into play, which is about $2K per node. So, if you clustered together 8 HP Integrity rx1620 systems, the cost would be $88,800 plus $16,000 for interconnects. The total would be $104,800, which is almost a third of the price a 16 CPU SMP like the HP Integrity rx8620 below would cost you.

Table 1.

PRICE of HP systems (LIST)

HP Server Model	processor	Number of CPUs	Memory size	List price	List price/ processor
Integrity SuperDome	Intel Itanium2 1.5GHz	64	64GB	$2250000	$35156
Integrity rx8620	Intel Itanium 1.5GHz	16	16GB	$304000	$19000
Integrity rx4640	Intel Itanium 1.5GHz	4	4GB	$36000	$9000
Integrity rx1620	Intel Itanium 1.6GHz	2	2GB	$11100	$5550
Proliant DL145	AMD Opteron 2.6GHz	2	2GB	$5100	$2550

Table 2.

wing job large – 1M DOFs

	Serial	2-way-parallel	4-way-parallel	8-way-parallel
Integrity rx8620	629	405	268	202
Integrity rx1620	490	310	208	158

wing job very large – 2.5M DOFs

	Serial	2-way-parallel	4-way-parallel	8-way-parallel
Integrity rx8620	1360	862	595	463
Integrity rx1620	1108	720	492	376

Now look at both Table 1 and Table 2 above and compare price versus performance for distributed benchmark, which involves a wing built with solid models and solved with the PCG solver. Now understand this is not how a wing model is normally constructed, but this example does show the relative performance.

The Integrity rx8620 is a 16-CPU SMP; the Integrity rx1620 is a 2-CPU SMP. Both systems use the same processor. As is common in SMP design, small SMP's have lower memory latency than do large SMP's, and as a result the serial performance is faster on the Integrity rx1620.

And the cluster of 4 Integrity rx1620's outperforms the single Integrity rx8620 for all levels of parallelism.

So, now we come into the last couple of years, when the CAE applications have completed migrations to cluster computing and industry-standard processors.

Because these machines do not have a single operating system image, thread-parallelism or lightweight process parallelism is not achievable. Applications must use the multiprocess based parallelism, and the CAE market standardized on MPI.

The migration from proprietary processors and operating systems to industry-standard processors (x86-32, x86-64, and Intel Itanium) resulted in a concurrent migration to LINUX. Along with the many good effects of these migrations, there was one negative side-effect: a proliferation of incompatible MPI ports.

As a part of the development environment of the various proprietary systems, there were vendor specific versions of MPI. The move to the commodity paradigm began to obsolete them. The HP planners decided to take a visionary approach to the changes in the high performance computing world. They created a plan to expand HP's existing HP-MPI products (for HP-UX and TRU64) .

This plan, which was implemented by the HP-MPI group, created HP-MPI for LINUX, supporting 1) all industry-standard processors, 2) the major distributions of LINUX, and 3) the major cluster interconnect products. HP implemented this plan and invited the CAE ISV's to sign on to use HP-MPI on these various LINUX-based platforms. ANSYS was the first vendor to officially sign the agreement. ANSYS and other vendors who have chosen HP-MPI now have a portable solution that supports 27 combinations of hardware, LINUX versions, and interconnect mechanisms. HP supports the HP-MPI product on all of these configurations and also tunes the performance.

ISV's could use MPICH for all these configurations, but with MPICH, you must build a different MPICH library and link with the main executable for the various LINUX versions and interconnects. With HP-MPI, a developer or a user just needs to link with one library. For example look at the two figures below. The first one shows comparable results on both the Proliant DL145, with the Myrinet interconnect using both HP-MPI and MPICH, and the Integrity rx2600, with the Quadrics interconnect using both HP-MPI and MPICH. With MPICH, this would take two separate builds of the application. HP-MPI does both! The second figure shows much better performance using HP-MPI on an Integrity rx2600 running intranode. This is because that the MPICH version was built to use sockets in the OS. However HP-MPI will determine the best interconnect, which in this case is shared memory, and then use it.

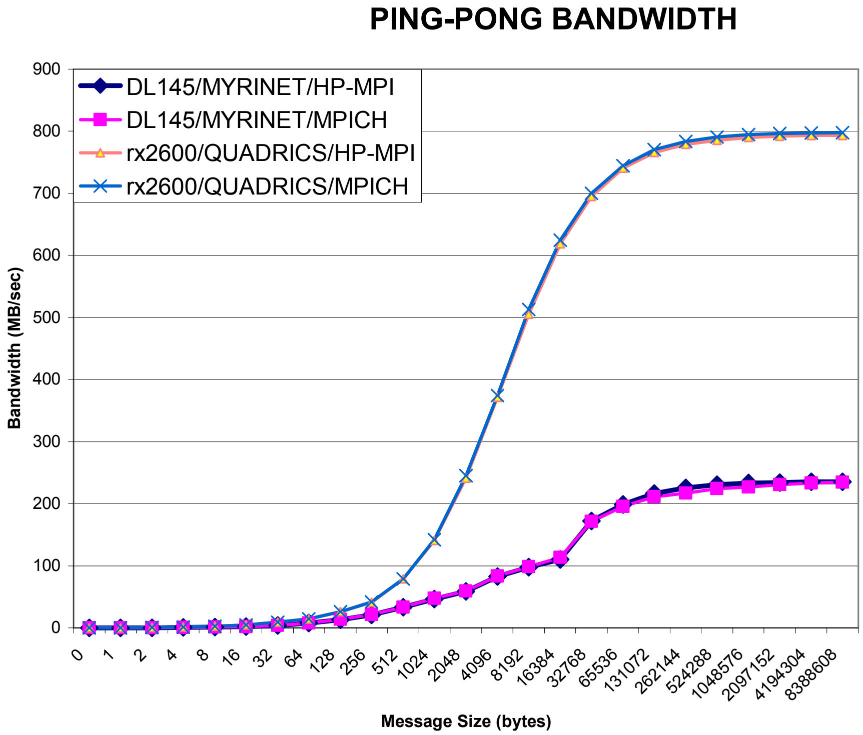

Figure 1.

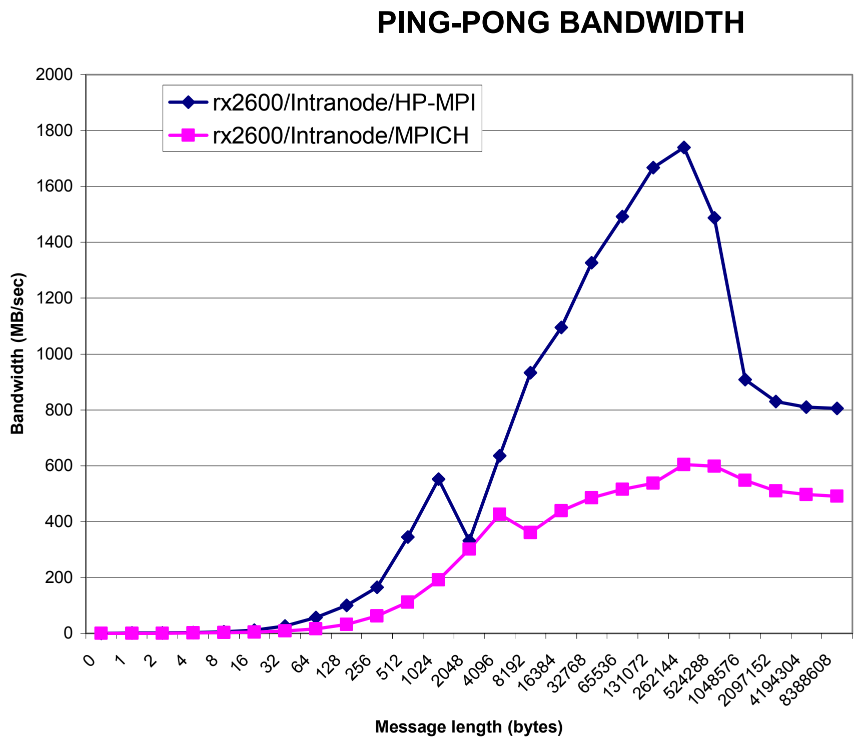

Figure 2.

History of D-ANSYS

Distributed ANSYS or D-ANSYS was developed to take advantage of the cluster computing migration. With ANSYS 5.7, ANSYS came out with the Domain Decomposition Solver. This solver originally was for the user with small machines, the idea being that those small machines could be clustered together to run the large jobs that previously required large SMPs. The scalability was quite good: up to 22x[3] speedup. This of course is dependent on the parallel architecture. The next step was done with what is known as the PowerSolver, which is ANSYS' best iterative solver, otherwise known as the PCG or Preconditioned Conjugate Gradient solver. This solver has been around since version 5.2 and was thread-parallel. With ANSYS 8.0, a distributed version of this solver was released.

By this time, the trend for the High Performance Computing market was moving from large scale SMPs with proprietary architectures to smaller commodity based systems, which could be clustered together for comparable computing power at a fraction of the cost. However, this required that the applications must be multiprocess parallel. Also, the multiprocess parallel technologies must be tuned for these systems and the various types of interconnects that are used for clustering these systems. So with version 9.0, ANSYS released their D-ANSYS product, which included distributed flavors of the PCG, JCG(Jacobian Conjugate Gradient) solver, and the distributed sparse solver. However the SMP-shared-memory-threaded versions of the iterative solvers, plus a highly tuned sparse solver, were still available for customer use.

It all comes together.

For ANSYS 10.0 the two lines of D-ANSYS and HP-MPI were to cross. ANSYS was the first software vendor to sign the agreement to use HP-MPI in its LINUX based products. So, ANSYS 10.0 is available with HP-MPI on LINUX running on industry-standard servers. Of course it is also running on both HP 9000 PA-RISC and Integrity servers running HP-UX. Also as the table below indicates, ANSYS has made performance improvements going from the distributed versions of 9.0 to 10.0. This improvement is accomplished by reducing the number of MPI messages by packing more info into these messages, thereby increasing the compute to communication ratio. This information by was acquired by the use the HP-MPI instrumentation which reports number and size of messages per process as shown earlier in this paper.

Table 3.

	Wing Job Medium	500KDOFs		
	Single Process	Two Process Parallel	Four Process Parallel	Eight Process Parallel
Ansys 9.0	500	415	488	436
Ansys10.0	499	318	196	149

Conclusion

To conclude, D-ANSYS with HP-MPI takes advantage of the paradigm of commodity processors, along with open source LINUX to bring the customer unmatched price performance gains using distributed computing. The working relationship between HP and ANSYS, which this writer has been part of for more than a decade, is one of the best in the business. As Lisa Fordanich, who is a Senior Systems Specialist for ANSYS, and who I've worked with most of the last 10 years says, "One of the top reasons that we went with HP-MPI is that we've had a great working relationship with HP. It was a win-win for ANSYS, HP and our customers - in terms of cost, interconnects, support and performance compared to other message passing interfaces for LINUX and UNIX. In addition I've always had great turnaround from HP in response to hardware and software issues." So, along with unmatched price/performance, the customer also will have support from two great organizations renowned for their commitment to customer satisfaction.

References

[1] www.sc2000.org/bell

[2] "Effective Moore's Laws in High Performance Computing Based on Gordon Bell Prize Winners", Xiaofeng Gao, University of California at San Diego.

[3] "Ansys Solvers: 30 Year Development – A look Into the Future", Ansys solver Team, April 2002, Revised October 2005.

Intel® Xeon® Processor, Intel® Itanium® 2 Processor, and AMD Opteron™ are trademarked products of Intel® and AMD.