A Study on the Scalability of Hybrid LS-DYNA® on Multicore Architectures

Yih-Yih Lin
Hewlett-Packard Company

Abstract

In this paper, the effects of differences in problem size, number of cores per processor, and interconnect on the Hybrid LS-DYNA's performance are studied. The result shows that combination of these three factors determines when Hybrid LS-DYNA has a performance advantage over the MPP LS-DYNA.

Introduction

The newly developed Hybrid LS-DYNA is a hybridization of SMP LS-DYNA and MPP LS-DYNA. (We will call Hybrid LS-DYNA the Hybrid Method and MPP LS-DYNA the MPP method from now on in this paper.) Both methods decompose the problem into subdomains that are solved by a single message passing interface (MPI) process. These MPI processes are commonly called ranks. In the MPP Method, ranks are not threaded, while the ranks of the Hybrid method are thread parallel. The following relationship applies for both methods:

Number of ranks = Number of subdomains

For the MPP Method, the number of cores used is the number of ranks:

Number of cores = Number of ranks

while the Hybrid Method uses more cores than ranks:

Number of cores = Number of threads per rank × Number of ranks

Given a number of cores and a given problem, two factors that affect performances of the two methods can be drawn:

- Communication cost: The communication cost is in general proportional to the number of ranks. For a given number of cores, the Hybrid Method uses much fewer ranks than the MPP Method, so the communication cost for the Hybrid method should be much less than that of the MPP Method. Furthermore, the communication cost is affected by the speed of interconnect and the problem size.

- Computation cost on each subdomain: The scalability of the each subdomain's solution should behave similarly to that of the SMP LS-DYNA, which also uses thread parallelism. SMP LS-DYNA is well-known to scale sub-linearly with respect to the number of cores and does not scale well beyond 8 cores. It is also commonly known that the scalability of a server with multiple processors drops substantially across processors. Modern processor chips include memory controllers, so sharing memory on multiple chips means using caches that are not shared and memory access will be slow. Therefore, for optimal performance, the number of threads per rank in the Hybrid Method should equal to the number of cores per processor for a given architecture.

This paper investigates these factors experimentally with two problem sizes, two interconnects with different speeds, and two architectures with different numbers of cores per processor.

Scalability versus Problem Sizes, Architectures, and Interconnects

Two car crash problems from National Crash Analysis Center, http://www.ncac.gwu.edu, described below, are used in the study:

Problem	Number of Elements	Termination Time	Description
Three-car Collision	0.8 M	150 ms	Impact of 3 Caravan at 35 mph
Car-to-car Collision	2.4 M	120 ms	Impact of 2 Caravan at 35 mph

Benchmarks are run on two clusters: One is comprised of HP BL280 nodes containing two Intel Nehalem Xeon processors, and the other of HP BL465c nodes containing two AMD Istanbul processors. A summary of the hardware configuration for the two clusters is shown below:

Node type	HP BL280c	HP BL465c
Architecture	Intel Xeon X5560 (Nehalem) (quadcore) (2.8 GHz)	AMD Opteron 2435 (Istanbul) (sextuple-core) (2.6 GHz)
Processors/Cores per node	2 processors/ 8 cores per node	2 processors/ 12 cores per node
Highest level Cache Configuration	One 8 MB L3 cache shared among 4 cores	One 6 MB L3 cache shared among 6 cores
Memory per node	24 GB	24 GB
Interconnect	InfiniBand QDR or GigE	InfiniBand DDR or GigE
O/S	SLES 11.0	SLES 11.0

The number of threads per rank for the Hybrid Method, used in call cases of this section, is equal to the number of cores per processor, i.e., 4 for the BL 280c cluster and 6 for the BL465c cluster.

Scalability versus Problem Size

Figure 1 shows the elasped times measured for the Car-to-car Collision, as well as their comparison, with both methods, versus number of cores on the BL280 cluster with InfiniBand (IB) QDR; Figure 2 is the performance of the Three-car Collision run on the same cluster, Figures 3 and 4 are the performances run on the BL465c cluster with InfiniBand DDR; Figure 3 is for the Car-to-car Collision, and Figure 4 for the Three-car Collision.

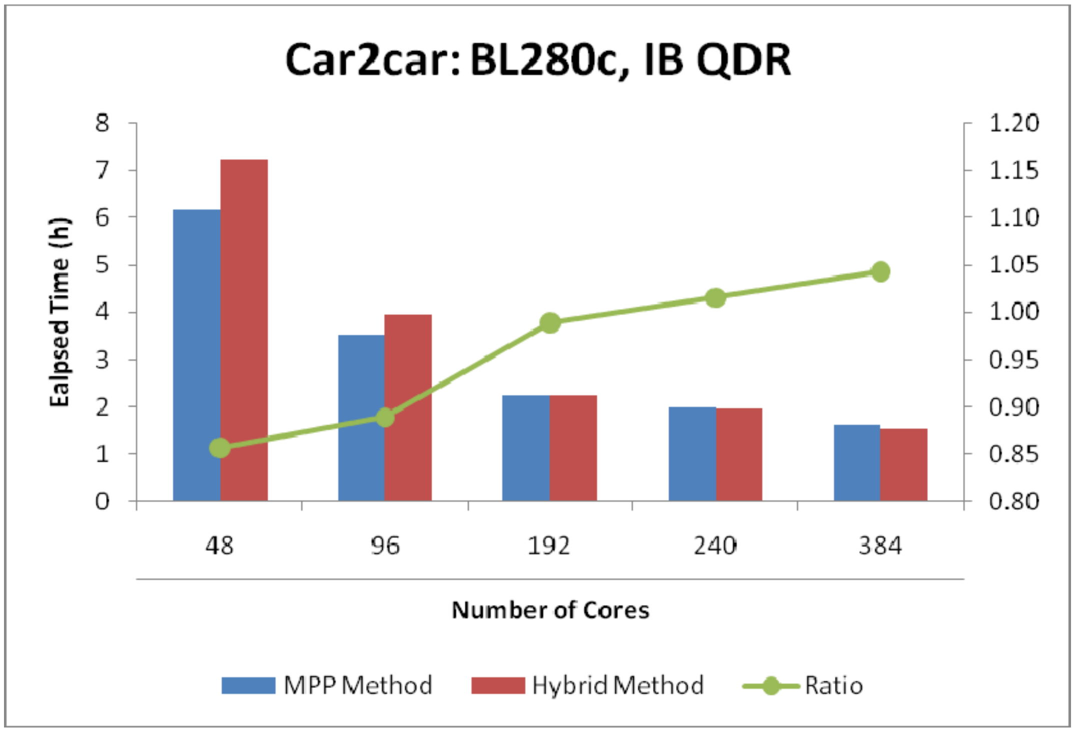

Figure 1. Elapsed times of the Car-to-car Collision with the MPP Method and the Hybrid Method, and their comparison, versus number of cores on the BL280c cluster with IB QDR.

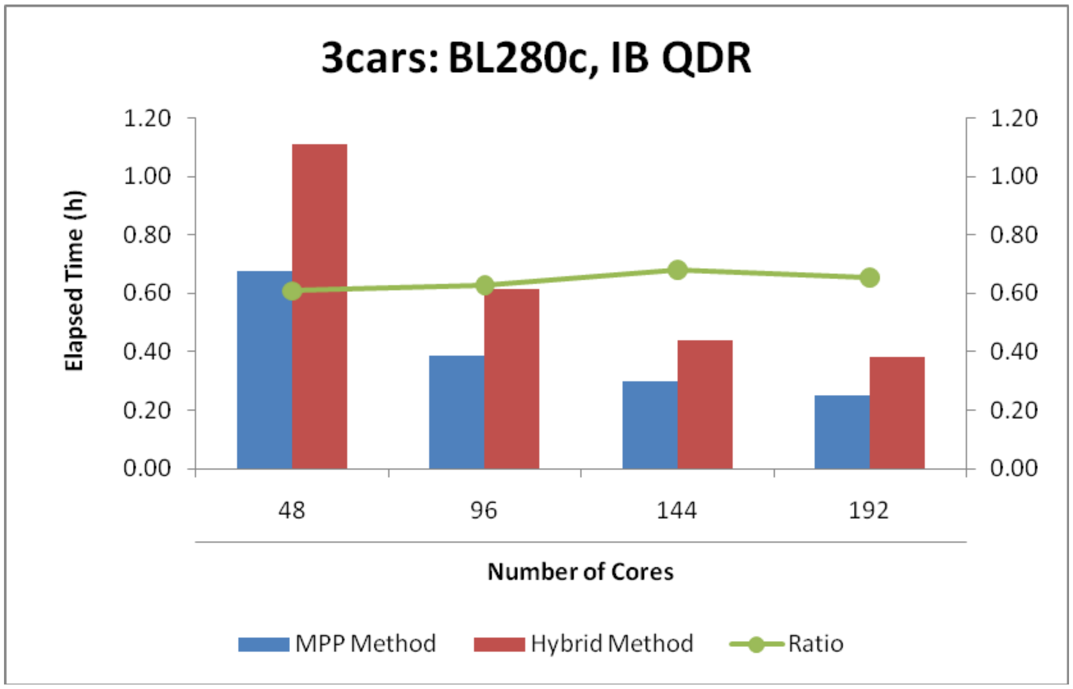

Figure 2. Elapsed times of the Three-car Collision with the MPP Method and the Hybrid Method, and their comparison, versus number of cores on the BL280c cluster with IB QDR.

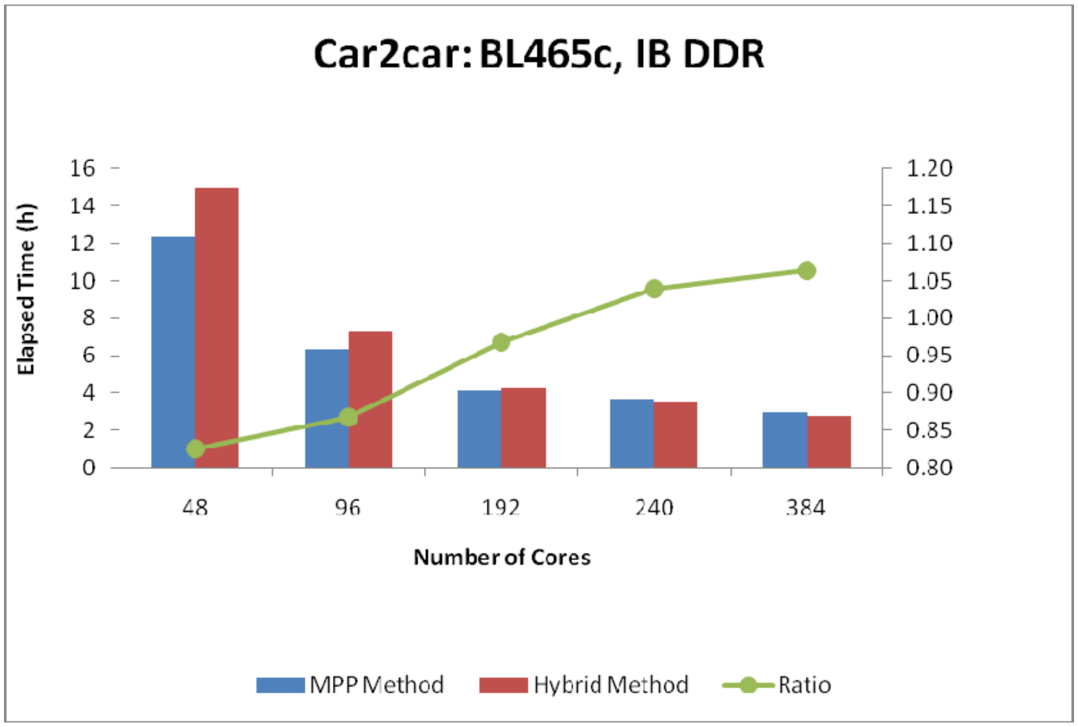

Figure 3. Elapsed times of the Car-to-car Collision with the MPP Method and the Hybrid Method, and their comparison, versus number of cores on the BL465c cluster with IB DDR.

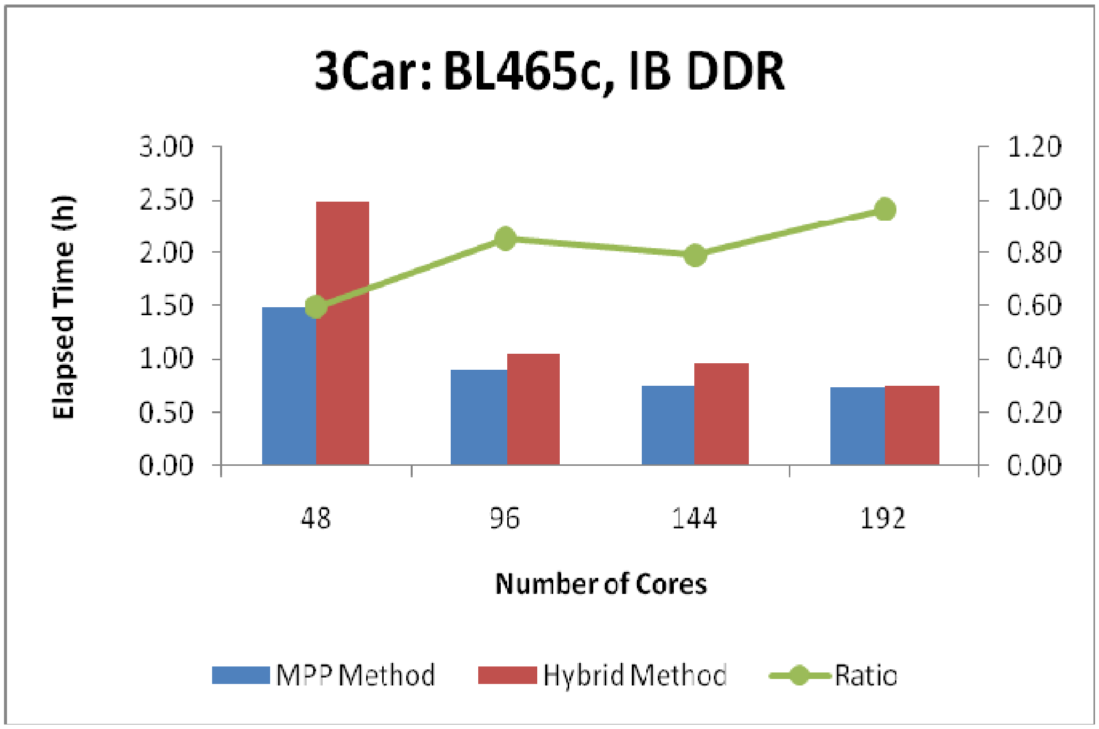

Figure 4. Elapsed times of the Three-car Collision with the MPP Method and the Hybrid Method, and their comparison, versus number of cores on the BL465c cluster with IB DDR.

The following observations can be drawn from the four figures, which are on clusters equipped with the fast interconnect InfiniBand:

- Regardless of architecture, the Hybrid Method has advantage over the MPP Method only if the problem size and the number of cores are large enough.
- For the larger problem, the Car-to-car Collision, the Hybrid Method starts to have a performance advantage over the MPP Method at about 192 cores.
- For the smaller problem, the Three-car Collision, the Hybrid Method performs worse than the MPP Method regardless of the number of cores; so there is no reason to use the Hybrid method.

Scalability versus Architecture

For the Car-to-car Collision, at 384 cores, the Hybrid Method outperforms the MPP Method by 4 percent on the BL280c cluster (Figure 1), whereas the former method outperforms the latter by 6 percent on the BL465c cluster (Figure 3). As explained in an early paper [1], the performance difference can be attributed to difference in number of cores per processor (same as number of threads per rank in the Hybrid Method) between the two architectures: 4 on the BL280c cluster versus 6 on the BL465c cluster.

Scalability versus Interconnects

Figure 5 shows the elasped times measured for the Car-to-car Collision, as well as their comparison, with both methods, versus number of cores on the BL280 cluster with Gigabit Ethernet (GigE); Figure 6 is the performance of the Three-car Collision run on the same cluster. Figures 7 and 8 are the performances run on the BL465 cluster; Figure 7 is for the Car-to-car Collision, and Figure 8 is for the Three-car Collision.

The following observations can be drawn from the four figures, which are on clusters equipped with the slow interconnect GigE:

- Regardless of architecture and problem size, the MPP Method stalls at about144 cores. In contrast, the Hybrid Method still scales at about 144 cores.
- For the smaller problem, the Three-car Collision, the Hybrid Method has a performance advantage over the MPP Method with GigE, unlike that with IB. It is notable that Figure 6 shows that the Hybrid Method at 144 cores is 1.4 times faster than the MPP Method at 48 cores on the BL280c cluster, and Figure 8 shows 1.6 times faster on the BL465c cluster.

Discussion

The performance behavior of the Hybrid Method relative to the MPP Method can be explained by difference in message pattern. Shown in Figure 8 are message patterns for the MPP Method, the Hybrid Method with two threads per rank, and the Hybrid Method with four threads per rank.

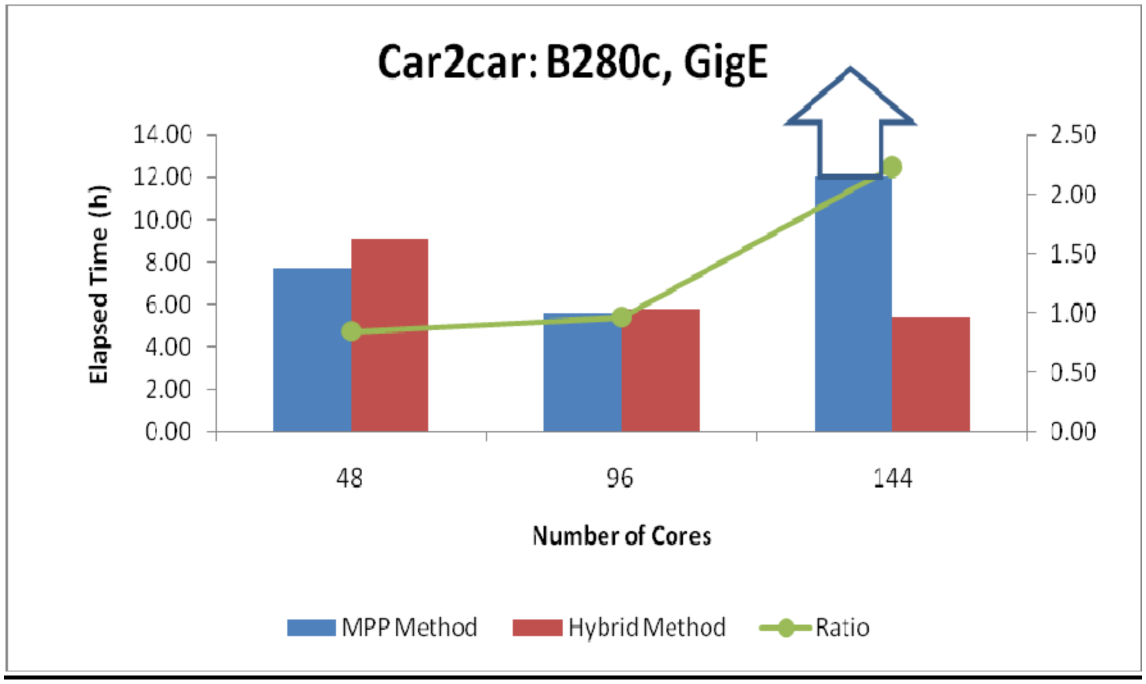

Figure 5. Elapsed times of the Car-to-car Collision with the MPP Method and the Hybrid Method, and their comparison, versus number of cores on the BL280c cluster with GigE. (Arrow indicates out of the axis's scale.)

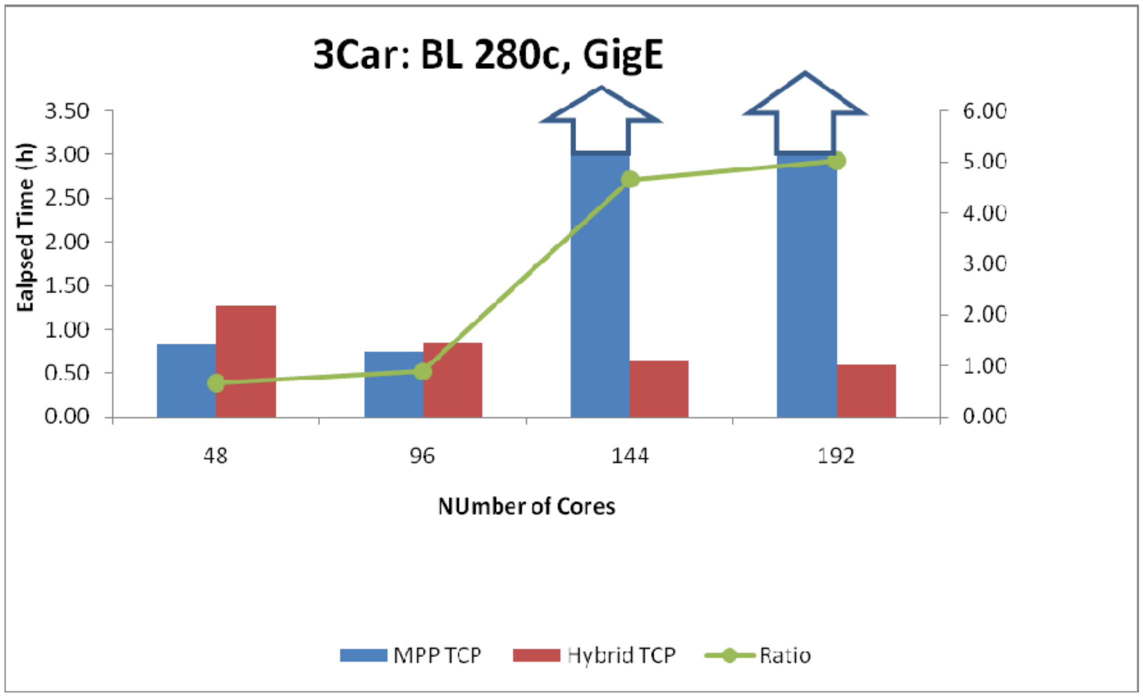

Figure 6. Elapsed times of the Three-car Collision with the MPP Method and the Hybrid Method, and their comparison, versus number of cores on the BL280c cluster with GigE. (Arrow indicates out of the axis's scale.)

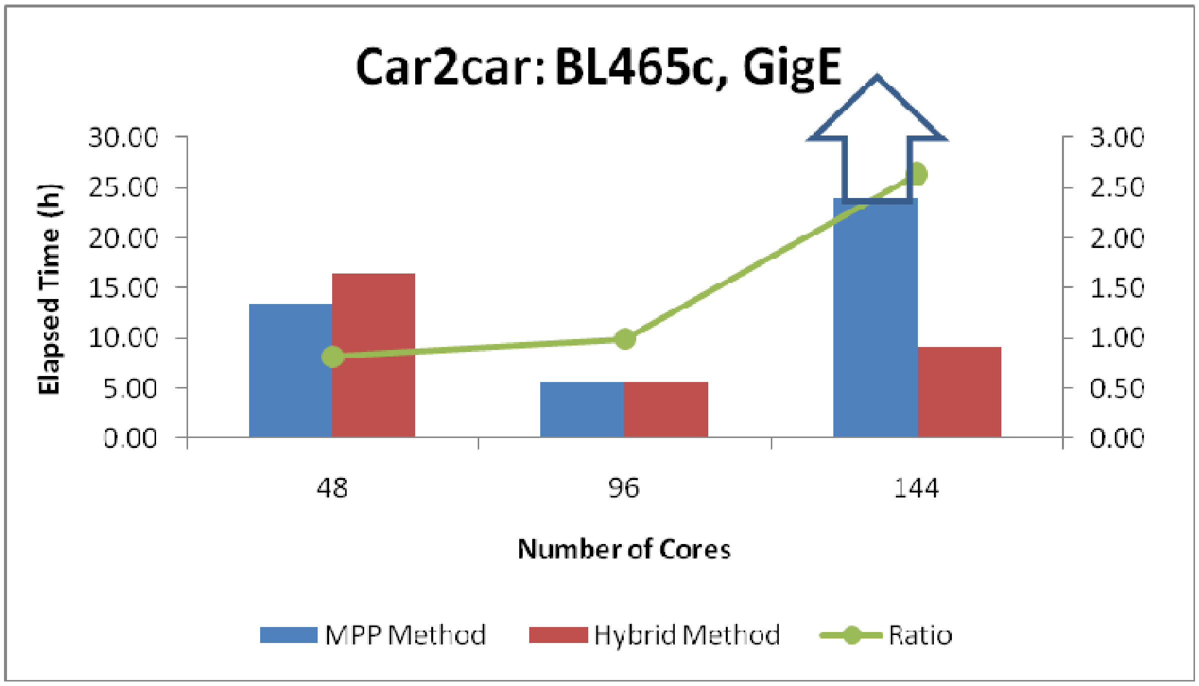

Figure 7. Elapsed times of the Car-to-car Collision with the MPP Method and the Hybrid Method, and their comparison, versus number of cores on the BL465c cluster with GigE. (Arrow indicates out of the axis's scale.)

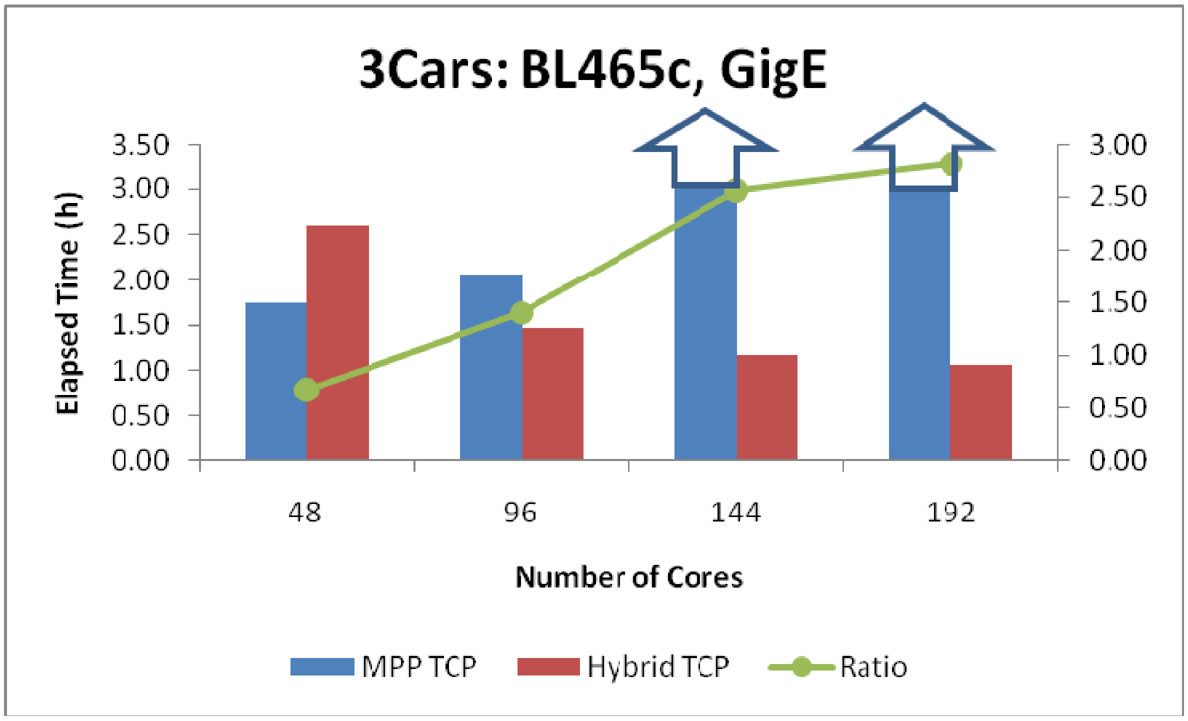

Figure 8. Elapsed times of the Three-car Collision with the MPP Method and the Hybrid Method, and their comparison, versus number of cores on the BL465c cluster with GigE. (Arrow indicates out of the axis's scale.)

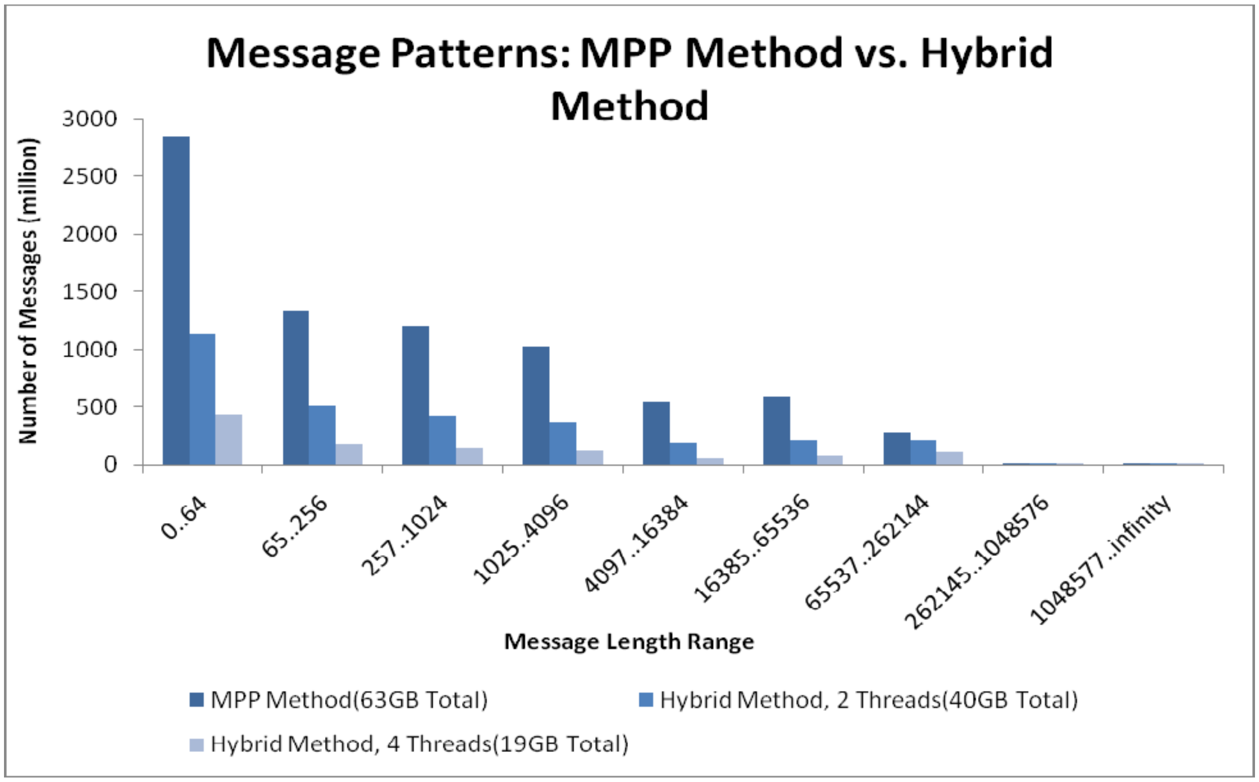

Figure 9. The Car-to-car Collision's 256-core message patterns for the MPP Method, the Hybrid Method with 2 threads per rank, and the Hybrid Method with 4 threads per rank.

Message pattern of the MPP Method is identical to that of the Hybrid Method with one thread per rank. Figure 9 shows the dramatic reduction in the number of messages and message sizes, which reduces the communication cost, as the number of threads is increased. The effectiveness of the Hybrid Method can be attributed to this ability in reducing the communication cost.

Conclusions

To summarize, this paper has established, from actual measurement of elapsed times, the following:

- The relative performance between the MPP Method and the Hybrid Method depends on problem size, number of threads per rank (hence number of cores per processor in an architecture), and interconnect.
- For a fast interconnect, such as InfiniBand, the larger the problem size, the better the scalability of the Hybrid Method relative to the MPP Method.
- For a slow interconnect, such as GigE, the Hybrid Method offers good scalability beyond 96 cores, for a problem size of 0.8 million elements.

References

1. Lin, Y. and Wang, J. Performance of the Hybrid LS-DYNA on Crash Simulation with the Multicore Architecture. 7th European LS-DYNA Conference, Salzburg, May 2009.