Performance Evaluation Using LS-DYNA Hybrid Version on the K computer

- Evaluation of Contact Computation on Highly Parallel System -

Kenshiro Kondo

Fujitsu Limited

Kazuo Minami, Yukihiro Hasegawa

RIKEN

Hiroyuki Umetani

Japan Automobile Manufacturers Association, Inc.

Yu Setoyama, Takafumi Horita

Fujitsu Kyushu Systems Limited

Hiroyuki Kanazawa

Fujitsu Limited

1 Introduction

In order to improve the accuracy of the car crash analysis, the number of elements in the analytical model has been increasing rapidly. A large-scale analysis using model with 10 million elements is slowly becoming popular. Some Companies actually has been adapting large models in their crash analysis nowadays. ”The K computer”[1], a highly parallel system, can carry out a car crash analysis in several hours which other supercomputers would have taken several days to complete the analysis in the past. However in order to achieve such efficiency, the analytical jobs have to meet following conditions: use LS-DYNA Hybrid version; deploy Groupable contact function of LS-DYNA; and reduce contact definitions as much as possible.

In this paper, we investigated the performance and behaviour of LS-DYNA Hybrid version using several thousand processes on “The K computer”. More specifically, due to the critical role of contact on the performance in a highly parallel system, we mainly focus on the following two aspects of contact calculation part throughout the discussion: the relationship of computational time with the number of contact definitions; and the effectiveness of Groupable contact.

This paper gives brief descriptions about “The K computer” of Riken, and LS-DYNA Hybrid version used in this study in Sections 2 and 3. Several issues that have been encountered when carrying out crash analysis on such highly parallel computing environment are discussed in section 4. The performance bottleneck and factors that hurt scalability are investigated in the next section using simplified model to reveal the effect on the performance with respect to various contact patterns and different number of contact definitions. The effects of Groupable contact function of LS-DYNA versus different contact patterns are studied in this section, too. In the end it is concluded that Groupable contact functions of LS-DYNA help to improve performance of crash analysis on highly parallel environment. A modification of the model to reduce number of contact definitions can boost the performance of LS-DYNA on the K computer.

2 Summay of "The K computer"

The K computer, a super-large-scale, highly parallel supercomputer, has been ranked as the best performer among all key top high performance computing players. The system was jointly developed by Riken and Fujitsu. The K computer is constructed with more than 80,000 CPUs. With eight cores per CPU, a total of more than 640,000 cores constitute the full system. The CPU is a superscalar type SPARC64^{TM}VIIIfx[2] developed for the K computer specifically. Each CPU provides peak floating point arithmetic processing speed of 128 GFLOPS and has theoretical memory bandwidth of 64 GB/s. Each node is consisted with a CPU, and these 80,000+ computing nodes are connected via a six dimensional mesh/torus network, named as Tofu (Torus Fusion)[3]. The Tofu provides bi-directional 5x2 GB/s per link, 100GB/s throughput bandwidth per node; that offers an aggregate bisection bandwidth of 30TB/s. The Tofu network has hardware global barrier implementation and integrated MPI support for collective operations. These unique features of Tofu interconnect network not only greatly reduce the group communication of global barrier synchronization caused by massive number of processors, but also provides a fault tolerance network due to twelve alternative paths existing in the 6-D network.

3 LS-DYNA Hybrid Version

The single precision LS-DYNA version 971 R6.1 (Hybrid parallel version) is utilized for this performance investigation of LS-DYNA on the K computer.

LS-DYNA Hybrid version is the executable which combines SMP parallel programming and MPP parallel programming paradigms. The SMP parts uses OpenMP language for thread parallelization; The MPP portions uses Message Passing Interface [MPI] language for process parallelization. In a highly parallel computing environment, due to using the same number of computing cores with considerably fewer number of MPI processes invoked, Hybrid version has an advantage in decreasing communication cost comparing with the pure MPP version[4].

The analytical process of LS-DYNA mainly is composed of three parts, element calculation part (Element), contact calculation part (Contact), and rigid calculation part (Rigid). In terms of performance, Contact is the most critical one in the highly parallel computing due to tremendous amount of communications can be generated. Based on the contact definitions of the model, the contact part calculates contact reaction forces. If the contact definition information extends among multiple processes, the MPI collective communication is invoked to exchange information among these processes.

The Groupable function of LS-DYNA, as described in section 5, provides an efficient way to reduce wait time in communication and significantly boost the performance of LS-DYNA in highly parallel computing.

4 Issues of Contact Calculation in Highly Parallel Computing

Figure 1 shows the car crash analysis model of 10 million elements with 49 contact definitions simulating 120ms physical time. This job was tested on the K computer using incremental processes up to 3,000 nodes (24,000 cores). The measured wall clock times were 6.4 hours using 384 processes, and 2.9 hours with 2,048 processes. The elapsed time was decreased 2.2 times, but the process number was increased 5.3 times.

In the subsequent study of performance, we used the same model to evaluate the scalability versus different number of process. Since both cars have contacted ahead of 40ms, significant deformation could be observed and the cost of contact calculation part took heavy proportion, we decided to concentrate our study between 40ms and 45ms physical time. Figure 2 shows the elapsed

times of the slowest process in Contact when jobs were executed with processes of 192, 384, 768, 2048, and 3072. The scalability of the main loop was quite good. When number of process increased from 192 to 3,072, the speedup of Element, Contact, and Rigid were 12.8, 4.2 and 0.2 times respectively.

The combining cost of Contact and Rigid are 52.74%, which suggested this was the priority target for performance improvement. Further study indicated that Rigid and Contact were tightly coupled, and Rigid was heavily influenced by the status of message passing occurred in Contact. Hence it was concluded that Contact was the actual performance bottleneck at this stage.

Figure 3 shows the cumulative time of each process when it was running on 2,048 processes. The horizontal axis is Process ID, and the vertical axis is execution times. The chart reveals the execution times among the processes are not uniformly distributed. Heavy costs were seen among the Process IDs of 900 to 1,000; where Contact occurred and the car deformed the most.

Although there is a way to distribute the contact areas evenly to all processes, such strategy actually increases the amount of communication. The increased cost of message passing has wiped out the effect of evenly distributing the cost of contact calculation. Hence it was concluded that simply relying on various methods of domain decomposition did not improve the scalability of this job. A different approach, as described in next section, was adopted to reach the goal of scalability improvement.

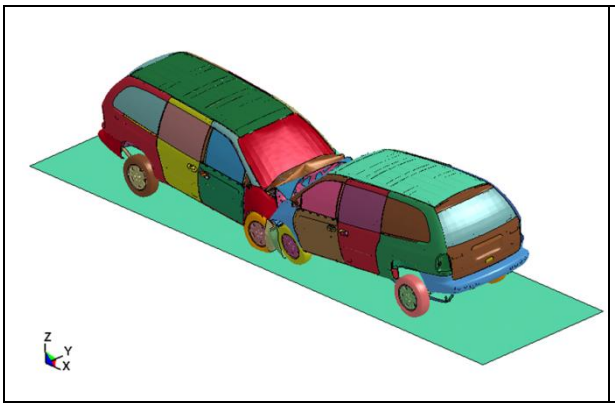


Fig1. Caravan 10M elements (d3plot)

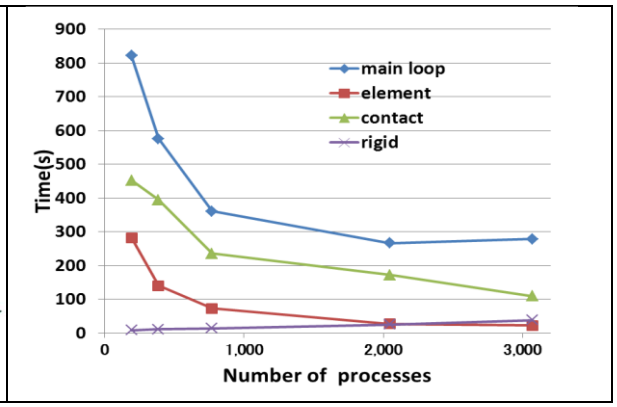


Fig2. Caravan 10M elements (scalability)

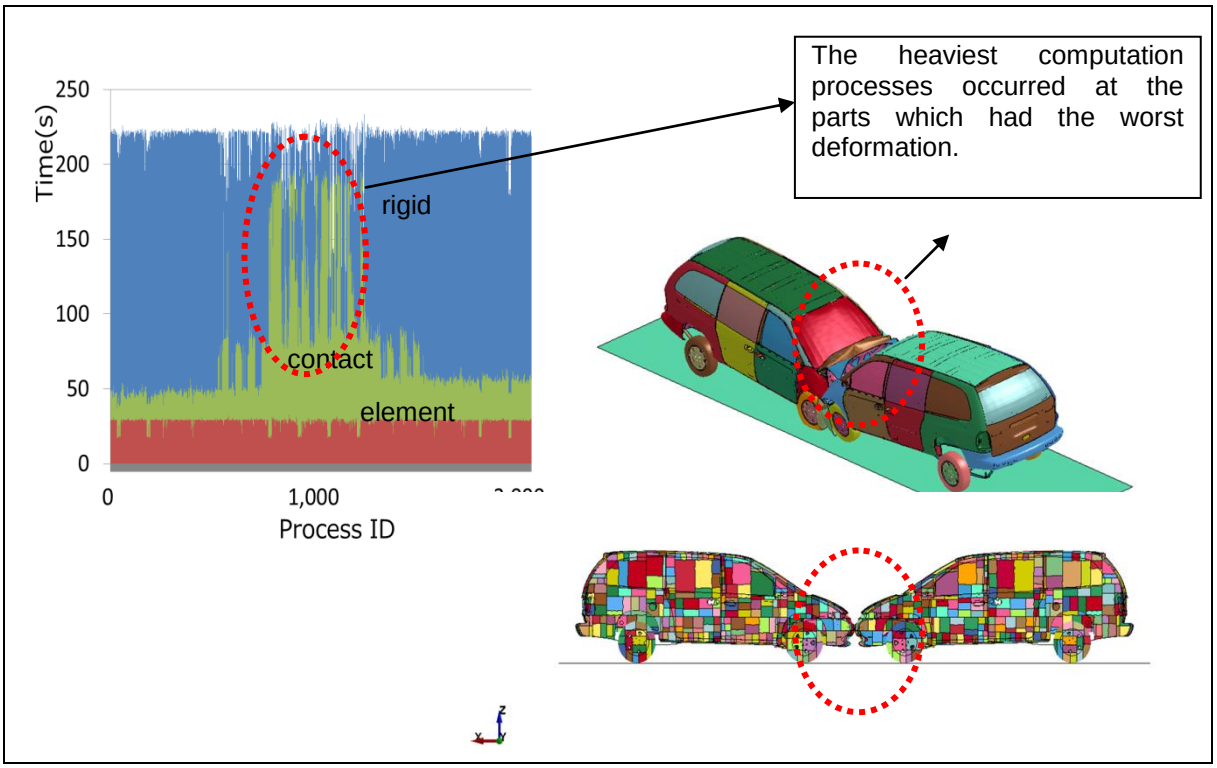


Fig3. all ranks timing costs (left) , deformed parts(right above) and decomposition image(right below) by 2048 process

5 Verification for Improving Contact Calculation

We adapted a simple collision model instead of using the car models to analyze performance improvement in the contact calculation part. There are two objectives in this study: the first is to investigate the extent of impact on the execution time caused by the number of contact definitions; the second one is to understand the effect of Groupable function that LS-DYNA provides. Detail of this simple collision model and its individual effect are presented in the following sections.

(1) simple collision model

To avoid unnecessary factors tangled in the contact calculation part, and to be able to identify the real problem that caused performance bottleneck, we used a simple collision model with behavior closed to what car crash model had. This simple collision model, as shown in Figure 4, pays attention to the contact calculation of the car crash model. The model is composed with two major objects: A car image object and an ODB barrier object.

The car image object was created with solid and shell elements. The nodes of shell elements and the nodes of solid elements are shared. To simulate ODB locally biases deformation, car image object crashed into the ODB barrier object with initial velocity of 54Km/h. Contacts were defined shell elements next to each other. Three different cases relating to number of contact definitions were used in this study, (A) with 279, (B) with 55, and (C) with 3 contact definitions. Although the number of contact definitions of cases (B) and (C) were fewer, the area of the contact parts remained the same as case (A). All three cases used “automatic_surface_to_surface” contact definitions. In addition, cases (B) and (C) also used “automatic_single_surface” contact definitions. Three optional definitions relating to contact stiffness, soft=0, 1, and 2, were included in the study, too. The physical analytical time were measured from 0 to 10ms. Table 1 lists key parameters used in this simple collision model.

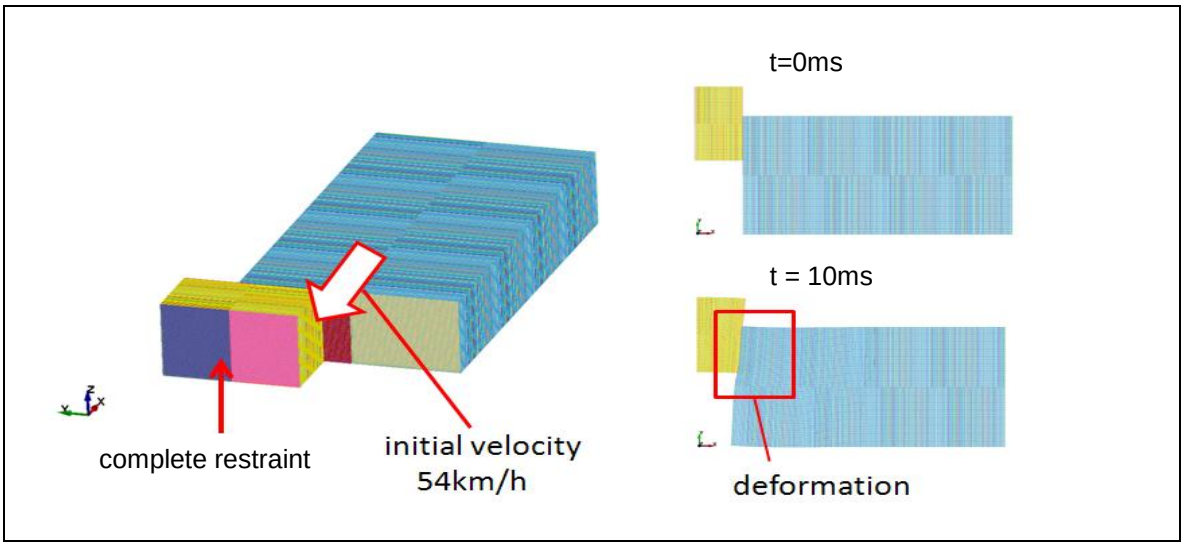


Fig4. simple crash model

Table1 . model information of simple crash model

Number of elements	5,807,070 (shell :1,467,990, solid :4,339,080)
Number of nodes	4,454,680
Number of contact definition	(A)279, (B)55, (C)3
contact optional definition	soft=0: penalty method, node segments contact soft=1: constraint method, node segments contact soft=2: constraint method, pinball segment based contact
Number of parts	286
Shell material	elastic modulus (MPa)=207,000, mass density (t/mm^3)=7.8E-09, Poisson's ratio＝0.28, mild steel , MAT_ELASTIC
Solid material	elastic modulus (MPa)=207, mass density (t/mm^3)=7.8E-09 Poisson's ratio＝ 0.28, MAT_ELASTIC

(2) Performance with respect to number of contact definitions

We used 1,024 processes to investigate performance impact caused by the three cases of different number of contact definitions, (A), (B), and (C), with combination of three contact options of soft=0, soft=1, and soft=2. Figure 5 shows the effect of the performance of the contact definition number. Using case (A) as base performance figure, one can see the relative speed-up of the other two cases with fewer number of contact definition, (B) and (C), in conjunction with soft=0, 1, and 2. Soft=0 and soft=1 shows better performance improvement comparing with the effect of soft=2. Soft=0 and 1 use node segment contact judgment and soft=2 uses pinball segment contact judgment in contact algorithm, which might be the cause on the effect of different performance improvement.

Figure 6 presents cumulative execution times of all processes using these three cases of different number of contact definitions with contact option of soft=0. The horizontal axis is Process ID and the vertical axis is execution times. Graph (a) shows the timing result of contact definition number case (A), graph (b) is the timing result of case (B), and graph (c) is the result of case (C). The Element times in graphs (a), (b), and (c) were almost identical. On the other hand, the Contact times shown in these three graphs differ in wide range among the processes. Graph (a) reveals the maximum and minimum times as well as their difference in Contact as 69.3, 30.5 and 38.8 seconds respectively. Furthermore, the execution times of Contact are much bigger than those of Element. In this case, comparing to the computation time, more communication time was spent to exchange data and for process synchronization comparing to actual number crunching. When more processes were defined

in one contact definition, more time would be spent in wait until data exchange has been completed. That was the main reason that too many contact definitions specified could hurt the performance. It was confirmed that 90% or more of the measured communication time was wait time as a result of a detailed analysis of the cost distribution.

The above arguments can be further confirmed with cases (b) and (c). The contact calculation part decreased as the number of contact definition reduced. The maximum and minimum time of case (b) was 41.8 and 5.5 seconds respectively with difference of 36.3 seconds. As for the case (c), with the least contact definitions, those timings were further reduced to 28.2, 6.7 and 21.7 seconds. These provided good evidences that reducing number of contact definition also cuts down synchronous wait time effectively.

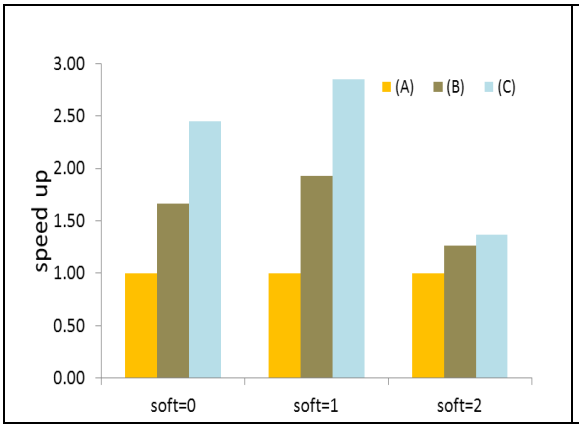


Fig 5 number of contact definitions and computation time (soft=0,soft=1,soft=2)

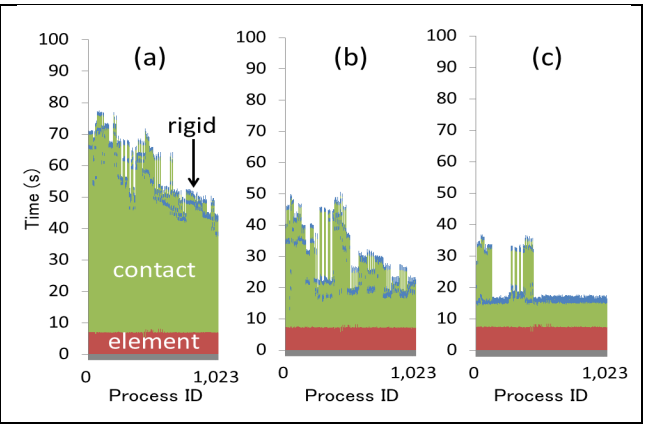


Fig 6 number of contact definitions and computation time(soft=0)

(3) Performance improvement with Groupable contact

The effect of Groupable function of LS-DYNA is studied in this section. Groupable contact is a function of LS-DYNA which improves the efficiency on Contact by easing the process conflict between two or more contact definitions. This feature has been available in releases R5.1.1, R6.1, and R6.1.1. Figures 7a and 7b illustrate the solution flow of Contact with default solution and with Groupable contact solution.

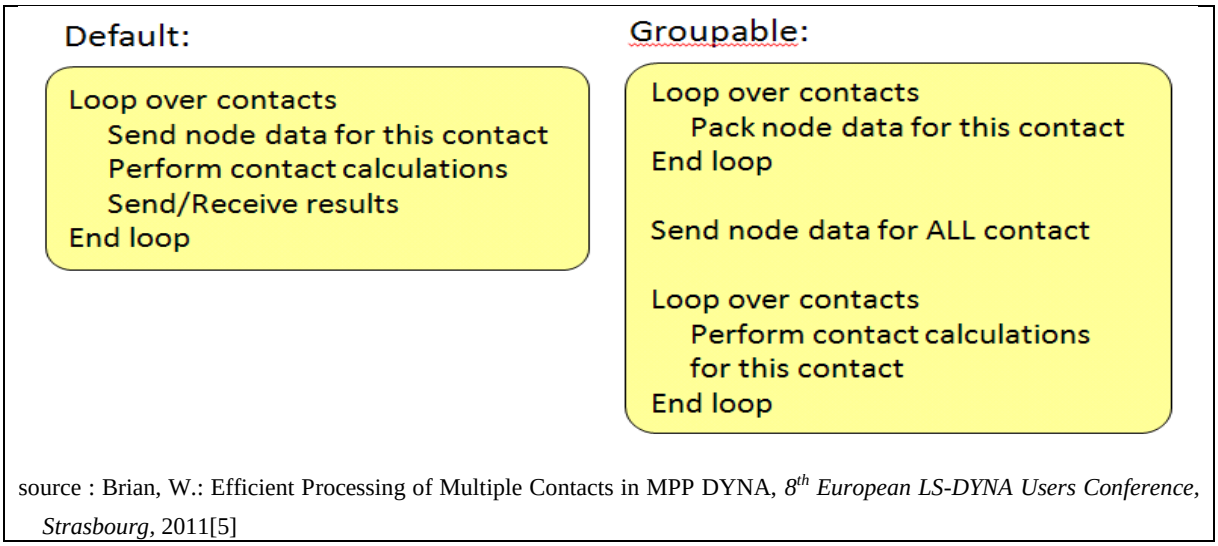
By default LS-DYNA handles contact definition in a repeated sequence as following, each contact requires data exchange among related processes and a synchronous wait occurs.

(process communication of contact 1) → (calculation of contact 1) →
(process communication of contact 2) → (calculation of contact 2) →…..

On the other hand, the Groupable contact method packs nodal information for each contacts first, sends pack nodal data, then performs contact calculations as following,

(process contact nodal data for contacts 1, 2, …) →
(process communication of all contact) →
(calculation of contacts 1, 2, …)

This method improves synchronous wait time by separating data exchange from contact calculation, and minimizes the delay of data exchange.



source : Brian, W.: Efficient Processing of Multiple Contacts in MPP DYNA, *8th European LS-DYNA Users Conference, Strasbourg*, 2011[5]

Fig7a. Groupable contact procedure

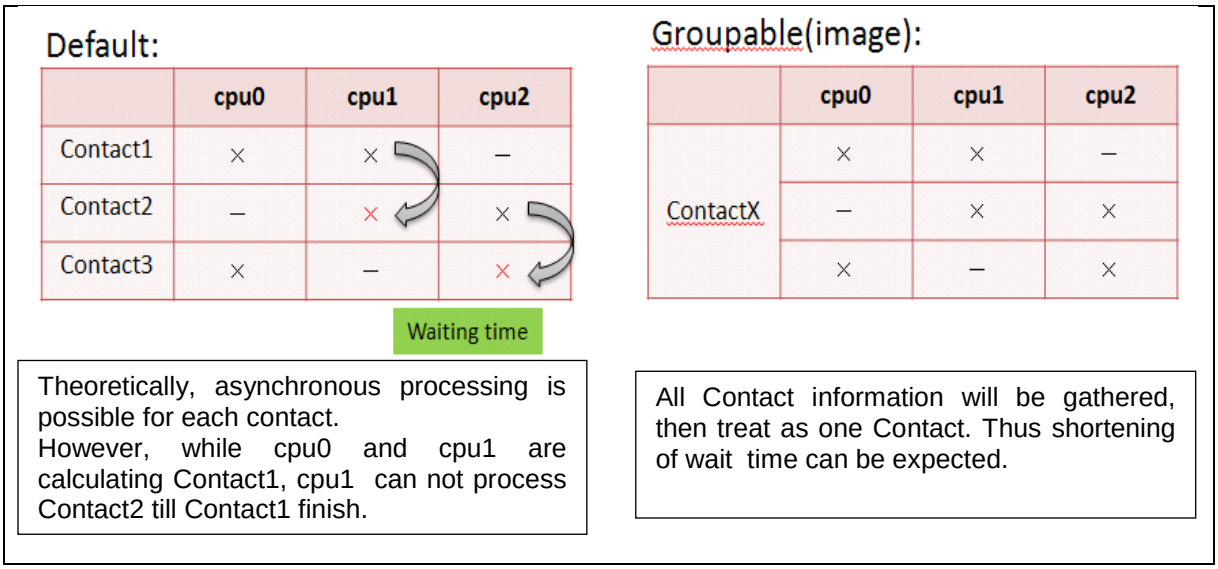


Fig7b. Default and Groupable contact procedure

We used the aforementioned simple collision model to study the effect of Groupable contact. Current released LS-DYNA only supports Groupable contact with contact options soft=0 and soft=1; soft=2 is not supported at this moment. In order to learn the effect of Groupable contact with respect to different amount of messages, we tested the three contact definitions of cases (A), (B), and (C) using 256 and 1,024 processes. Figure 8a1 shows the performance of Groupable contact relative to default (without Groupable) on 1,024 processes. We saw speedup of (A) 2.74 times, (B) 1.52 times, and (C) 1.19 times with soft=0; and speedup of (A) 2.61 times, (B) 1.49 times, and (C) 1.25 times with soft=1. Due to more number of contact definition and more number of processes being tested, the communication amount increased significantly. Case (A) enjoyed much better performance improvement with Groupable contact. Soft=0 performed slightly better than soft=1, although the difference was insignificant. Figure 8a2 shows the effect of Groupable contact with soft=0 running on 256 processes. The speedup of using Groupable was (A) 1.6 times, (B) 1.14 times and (C) 0.87 times. Slower performance was observed when using Groupable contact with Case (C) on 256 processes. Due to smaller amount of data in communication, those saved in data exchange and wait time did not compensate the overhead of packing nodal data for Groupable contact.

Figure 8b shows cumulative execution time of each of 1,024 process using contact definition case (A) with soft=0. The horizontal axis is Process ID and the vertical axis is execution time. The results of using default option (no-Groupable) is presented in graph 8b-(a); and the result of using Groupable contact is shown in graph 8b-(b). As expected, the Element part of both runs remained the same, and the Contact part was significantly improved when Groupable contact was activated. The maximum and minimum execution times of the default (no-Groupable) were 69.3 and 30.5 seconds with difference of 38.8 seconds. With Groupable contact, these three numbers reduced to 25.3, 10.6 and 14.7 seconds respectively. This proves that Groupable contact not only improved load balance of this job, but also reduced a lot of wait time in communication.

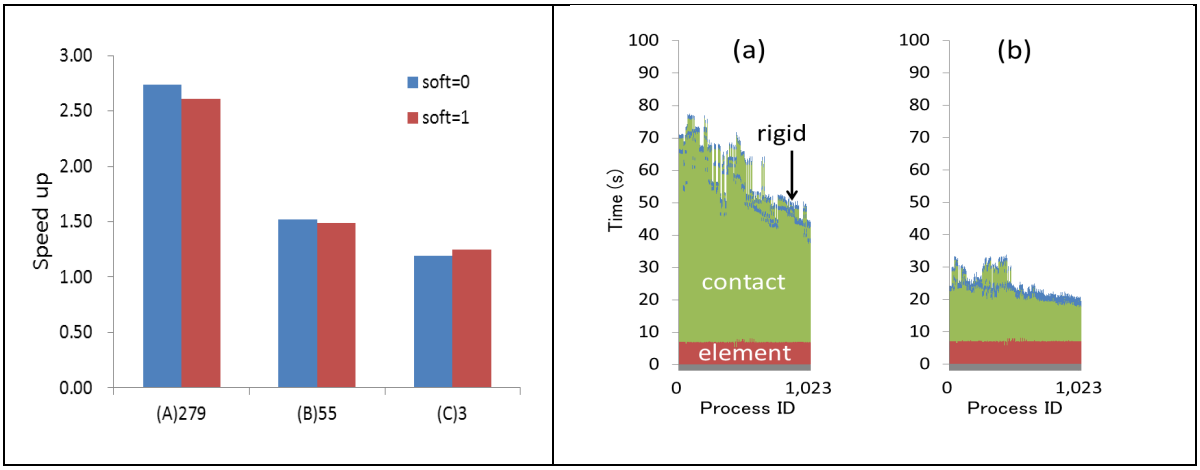


Fig8a1.Groupable performance (1024process) Fig 8b. Groupable performance case (A) (1024process)

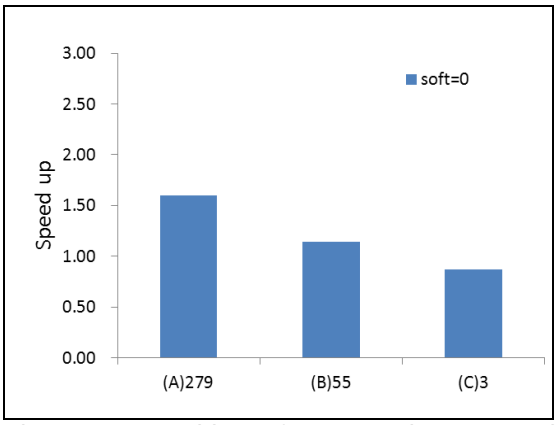


Fig 8a2. Groupable performance (256process)

6 Conclusion

The hybrid parallel version of LS-DYNA was capable to handle car crash analysis model of 10 million elements with forty-nine contact definitions running on RIKEN K computer up to 3,000 processes (24,000 cores). It has been proved that contact calculation part took a critical role in performance speed for auto crash analysis on such highly parallel system. There are two ways to improve contact analysis part when dealing with humongous model on highly parallel system: invoke Groupable contact feature of LS-DYNA and reduce number of contact definitions. Using Groupable contact has been proven quite straight forward and does not require users to modify the input file. Currently LS-DYNA only supports Groupable contact with soft=0 and 1. Reducing number of contact definition has been proven to be effective, but it would require users to modify and to improve the mathematical model. Since most of the models have been growing quite complicate and big,

modifying the model could put tremendous burden on the users. But the improvement on the performance can be very rewarding.

Acknowledgement

The authors would like to express their gratitude to several different parties. This is a joint research project among RIKEN, Japanese Automotive Manufactures Association and Fujitsu. We constantly exchanged opinions and information within these three parties. The test results of this study were obtained by testing LSTC's LS-DYNA on RIKEN's K supercomputer. The suggestions and cooperation from Drs. Jason Wang and Brian Wainscott of LSTC and Dr. Clifford Chen of Fujitsu America have been extremely valuable.

References

[1] Mitsuo Yokokawa and others : Special feature Super computer " "The K computer", Information Processing, Vol.53, No.8, Consecutive number of volumes 569, 754-807, 2012.

[2] Fujitsu Ltd.: *SPARC64VIIIfx Extensions*, Architecture manual, Fujitsu Ltd. , 2008.

[3] Ajima, Y. et al.: Tofu: A 6D Mesh/Torus Interconnect for Exascale Computers. *IEEE Computer*, 36-40. DOI=10.1109/MC.2009.370., 2009.

[4] Kenshiro Kondo: The Performance of Car Crash Simulation by LS-DYNA Hybrid Parallel Version on Fujitsu FX1, 11th International LS-DYNA Users Conference

[5] Brian, W.: Efficient Processing of Multiple Contacts in MPP DYNA, *8th European LS-DYNA Users Conference, Strasbourg*, 2011