Improving Performance of LS-DYNA® Crash Simulation with Large Deformation by Modifying Domain Decomposition

Shota Yamada
Technical Computing Solutions Unit, Fujitsu Limited
9-3 Nakase 1Chome Mihama, Chiba, 261-8588, Japan

Abstract

In modern high performance computing era, parallel computing has been a trend to improve the speed of computation. In the past we have found that just simply increasing the number of computing parallelism would not guarantee to achieve better performance especially when simulating large deformation using hundreds or more number of parallel processors. Through our past experience, to improve the computational performance, we had found it was necessary to tackle on the issue of load unbalance of calculation cost among processors and to seek for better strategy in domain decomposition.

In general, calculation cost increases with respect to the extent of deformation. To reduce the unbalance of calculation cost among processors, ideally we would like to decompose domain to subdomains with same extent of deformation on all processors. Even it is possible, it would be difficult to achieve such ideal decomposition for the cases with only local deformation occurred in crash simulation. Therefore we come up with a new enhanced method to decompose the model by distributing calculation cost more uniformly in crash simulation. In this paper, I will reveal this enhanced method, present the results of improved performance of this method using several models of crash simulation, and discuss the efficiency of this method.

1. Introduction

Nowadays the major manufacturers have utilized analytical simulation during the product development stages in order to reduce development cost, to cut down developing time, and to enhance development capability. The engineers have been constantly seeking for better analytical accuracy of the simulation in order to match the experimental data obtained through testing the prototype of products. For such demand results detailed mathematical models with increased number of elements used for simulation, which in turns puts pressure on better computational power for faster analytical job turnaround. Besides refined and more efficient software algorithms, parallel computation using multiple processors to handle a single job becomes the trend of modern crash simulation. It has been experienced that parallel efficiency worsens when number of processors increased. Such adverse effect becomes especially more critical in the highly parallel computing environment with hundreds or more processors.

For the highly parallel system, load unbalance has been the major cause of performance bottleneck. The load unbalance could be caused by unequal calculation amount decomposed to each processor, or due to unbalance in calculation and communication. Such difficulty gets more profound when the analytical models have only local deformation like crash simulation with large deformation. It has been recognized that it is extremely difficult to deal with such load unbalance issue when hundreds or more processors are utilized.

This paper reports the effect of speedup once the load unbalance issues of various models of crash simulation with large deformation has been addressed with a new method; then followed with examination for the efficiency of this method. The issues of load unbalance were examined in Section 2. Section3 discusses mechanism of various domain decomposition methods applied to solve these load unbalance issue. The efficiency in performance improvement with these methods is discussed in Section 4. The future work and summary are discussed in Section 5.

2. Issues of Load Unbalance

An ODB model with about 10 million elements as shown in Figure 1 is used to study how severe that load unbalance can be. The simulation time starts right at the beginning of contact until 40 milliseconds. Such duration is long enough for crash simulation.

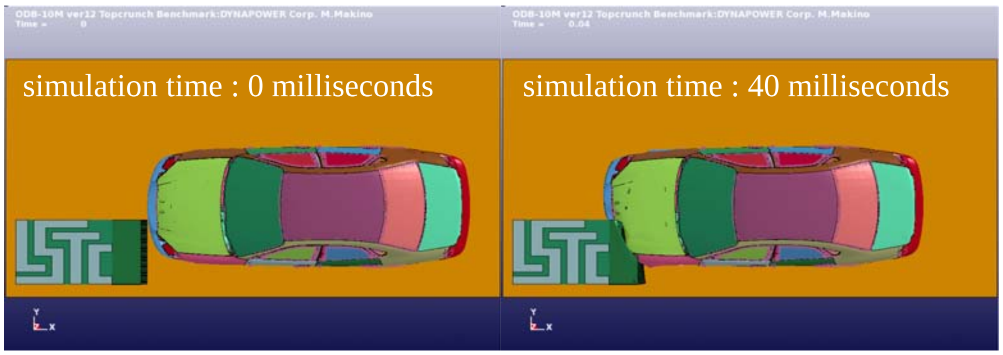

ODB-10M ver12 Topcrunch Benchmark:DYNAPOWER Corp. M.Makino
simulation time : 0 milliseconds
simulation time : 40 milliseconds

Figure 1. Images of ODB Model for Crash Simulation

Figure 2 shows load unbalance of calculation time with respect to domain decomposition using 48 parallel processors. The horizontal axis represents No. of Processors and the vertical axis shows the Calculation Time. The calculation time referred at here does not include those of communication time and wait time. Each subdomain is allocated to one processor sequentially from the bottom of Y-direction in the domain decomposition.

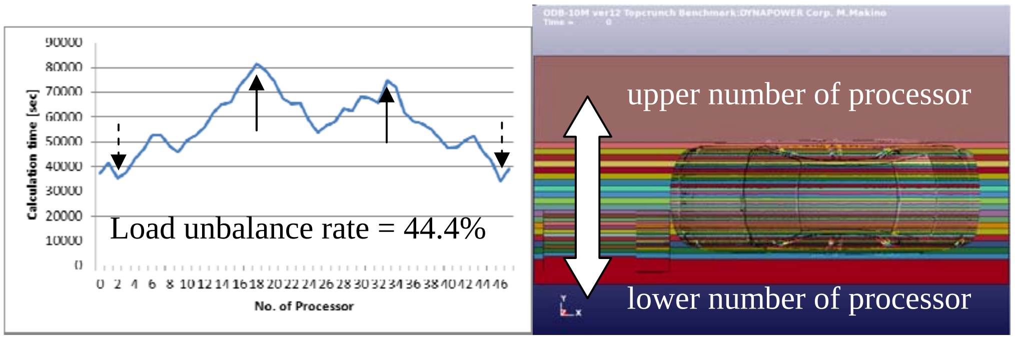


Figure 2. Calculation Load Unbalance and Domain Decomposition

Fig.2 reveals some subdomains have relative high calculation time (No.18 and 33, etc.) and some have short calculation time (No.2 and 46, etc.). Such load unbalance greatly hinders the expected efficiency of parallel computation, since those processors with longer calculation time become the bottleneck for better performance. Perfect domain decomposition would allocate same calculation amount to each processors for better performance. Due to the load unbalance, those processors with shorter calculation time have to wait for those processors allocated with heavier calculation amount to finish. Such wait times within the wall clock time are induced mostly by the calculation load unbalance, therefore it is reasonable to consider the rate of wait time as equivalent to the rate of load unbalance. At here the load unbalance rate can be estimated as following,

Tavg = Σ (Ti)/N,
Load Unbalance Rate = [Max(Ti) – Tavg]/ Tavg
where i=1, N
N is the number of processors.

This load unbalance rate would disappear once the issue of load unbalance has been resolved. Hence it can be regarded as the rate of expected speedup in tuning for better domain decomposition. The load unbalance rate of the above example is 44.4%.

As illustrated in Figure 3, the calculation amount of each processor is tightly relating to the extent of deformation of each subdomain. Those subdomains suffer larger deformation spend more calculation time. Such load unbalance eventually forms the performance bottleneck. The engineer can uniformly distribute the calculation cost among all the subdomains if the extent of deformation of each subdomain is predictable.

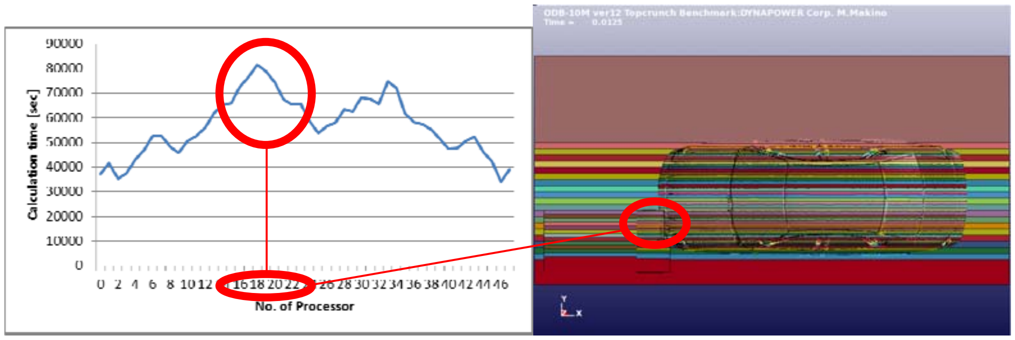


Figure 3. Calculation Amount and Extent of Deformation of Each Subdomain.

But it is very difficult to predict deformation of each subdomain in advance. Hence for the case having local deformation like the ODB model as shown above, it is a challenge to achieve well load balanced situation using some domain decomposition methods.

3. Method of Resolving Load Unbalance Issue

This section proposes a new method of domain decomposition which can reduce the load unbalance caused by large local deformation.

The domain decomposition that LS-DYNA currently uses is to distribute elements to each subdomain automatically based on some options that users configure. Based on the built-in calculation amount table, LS-DYNA evaluates some parameters relating to the calculation amount at initial state like data definition of each element, number of elements, etc. then distributes elements to ensure that each subdomain has uniform calculation time. Since LS-DYNA does not know the deformation in advance, following factors are not taken into consideration at the time of domain decomposition at the initial state: extent of element deformation, increment of calculation amount due to deformation, etc. If the users of LS-DYNA do not consider the deformation of model in advance and configure domain decomposition options with assumption that calculation amount of each subdomain remains equally through the whole analysis, as illustrated in Figure 4, the situation of load unbalance eventually becomes worst when deformation of some subdomains progress.

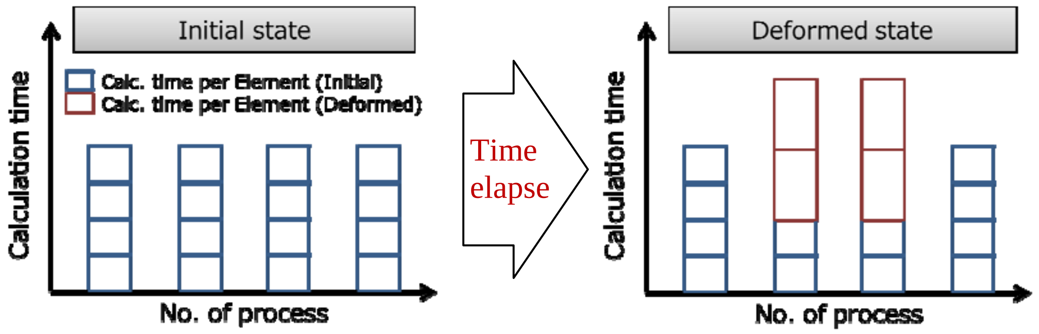


Figure 4. Propagation of Load Unbalance as Deformation Progress

LS-DYNA employs RCB (Recursive Coordinate Bisection) method by carrying out domain decomposition based on the longest evaluated XYZ direction. LS-DYNA provides various options that user can configure to apply this domain decomposition procedure. The SX/SY/SZ options are the most general options used to arrange order of direction to decompose, which recursively halves the model using the longest dimension of the input model. The ODB model of the previous example was configured with SY1000 for decomposition, so the domain decomposition was carried out only into the Y direction. There are several other options that users can utilize, e.g. radial direction decomposition, decompose in each predefined area, etc. In order to achieve a good load balance situation by distributing deformed element uniformly as shown in Figure 5, users have to learn these methods by trial-and-error.

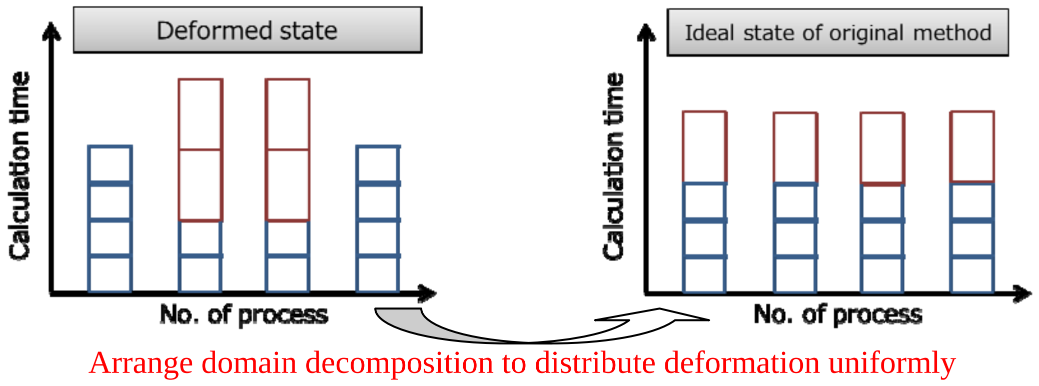

Deformed state
Calculation time
No. of process
Ideal state of original method
Calculation time
No. of process
Arrange domain decomposition to distribute deformation uniformly

Figure 5. Distributing the Extent of Deformation with the Original Method

For those models only has local deformation like the previous ODB model, trying to distribute the deformation extent uniformly on all subdomains become very complicate and difficult to carry out correction using the current options of the original configuration method. The author has enhanced the domain decomposition by correcting element distribution for all subdomains. This method readjusts the domain decomposition by allocating more elements with less deformation in a subdomain, and those subdomains having large deformed elements allocated with less total number of elements. Figure 6 illustrates the distribution of elements using this enhanced domain decomposition.

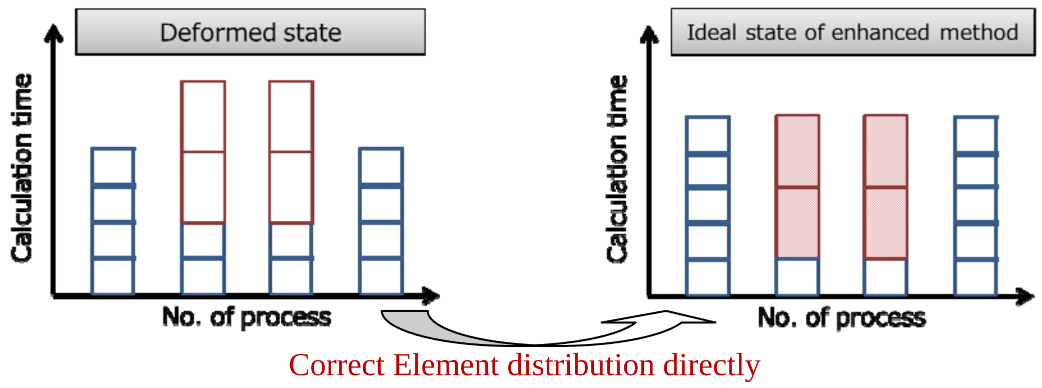


Figure 6. Redistributing Elements with the Newly Enhanced Method

Applying this enhanced domain decomposition on the aforementioned ODB model; the distribution of elements to each subdomain has been readjusted as expected. Figure 7 shows those subdomains in Zone B with larger deformed element have less total number of element per domain; each subdomain in Zones A and C with less deformed element has more total number of element allocated.

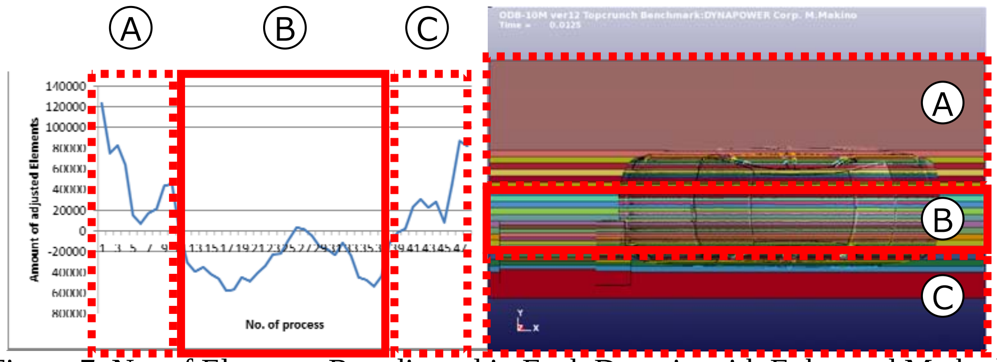


Figure 7. No. of Elements Re-adjusted in Each Domain with Enhanced Method

4. Improving effect

In addition to the above ODB model, four more test models are used to verify the effectiveness of this enhanced decomposition method. The four models used in this study are MDB model, Six Cars model, Neon model and 3Cars model. Detail descriptions about these four models are available to the public and can be found on the websites listed in Reference [1] and [2].

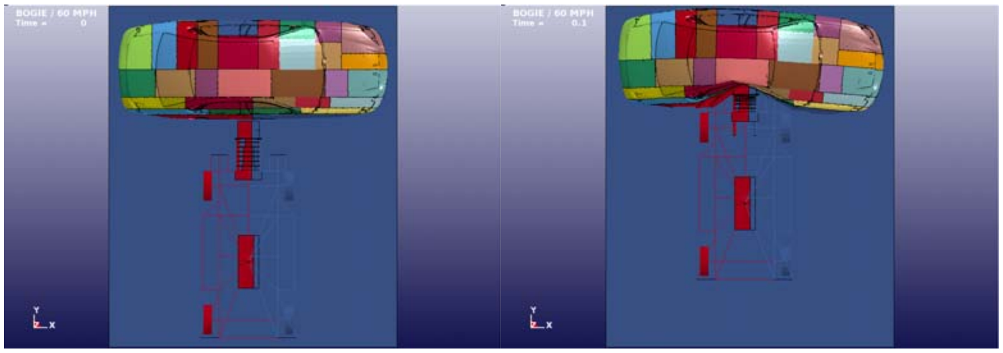
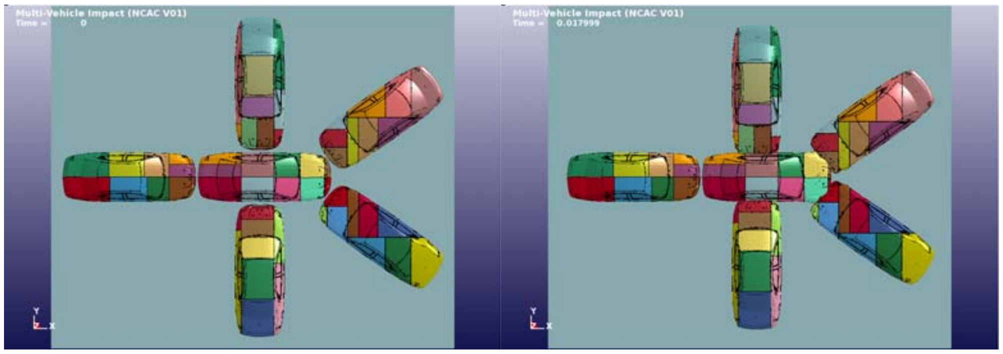


Figure 8. MDB Model


Multi-Vehicle Impact (NCAC V01)
Time = 0
Multi-Vehicle Impact (NCAC V01)
Time = 0.017999
Figure 9. Six Cars Model

Default option of domain decomposition is used for MDB and Six Cars models. So these two models are decomposed from the longest axis in X-Y-Z direction sequentially.

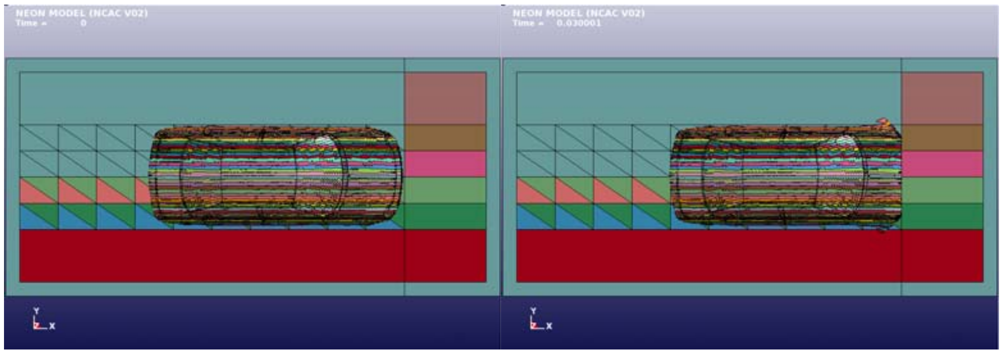


Figure 10. Neon Model

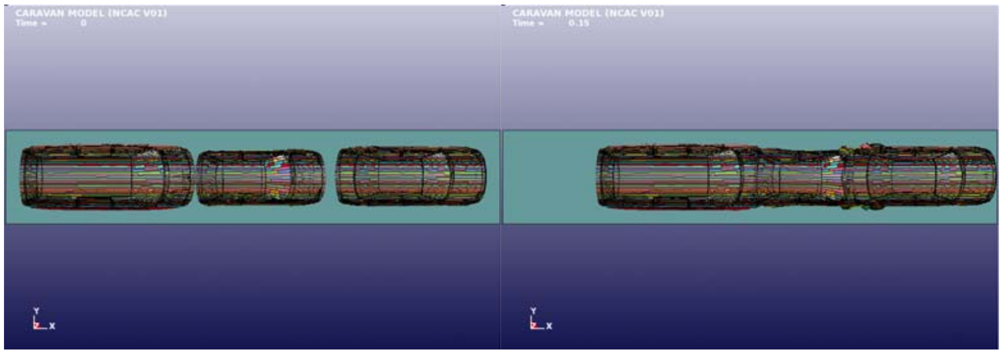
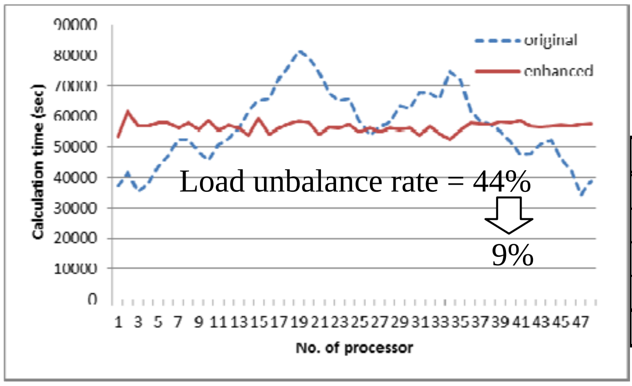


Figure 11. 3Cars Model

The decomposition for Neon and 3Cars models is carried out with SY1000 option. Hence decomposition is done only in the Y-direction similar to the ODB model. These full wrap frontal crash models deform relatively uniformly in one direction, so it is considered rather easy to distribute deformed elements to subdomains equally only by considering order of decomposition.

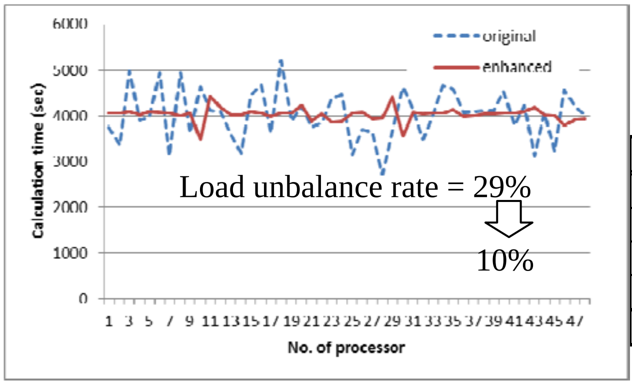
Figures 12 to 16 present the effect of the enhanced domain decomposition on the performance of these five models. The chart on the left side of each figure shows change of load unbalance. Dotted line represents value of original domain decomposition; solid line represents the value of newly enhanced domain decomposition. The table on the right side of each figure presents change of elapsed times of both decomposition method as well as the rate of speedup of the enhanced domain decomposition on each category of computation.



	current	new	rate of speedup	
Element	72090	54112	-17978	-21%
Contact	13211	15329	2118	2%
Rigid	1962	2209	247	0%
Other	386	402	16	0%
Elapsed	87649	72051	-15598	-18%

Unit : second

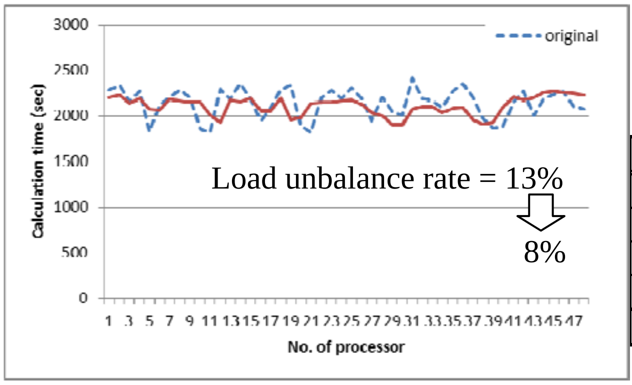
Figure 12. Change of load unbalance and elapsed time (ODB)



	current	new	rate of speedup	
Element	4596	3794	-802	-12%
Contact	1872	1670	-202	-3%
Rigid	374	414	39	1%
Other	38	36	-2	0%
Elapsed	6880	5913	-967	-14%

Unit : second

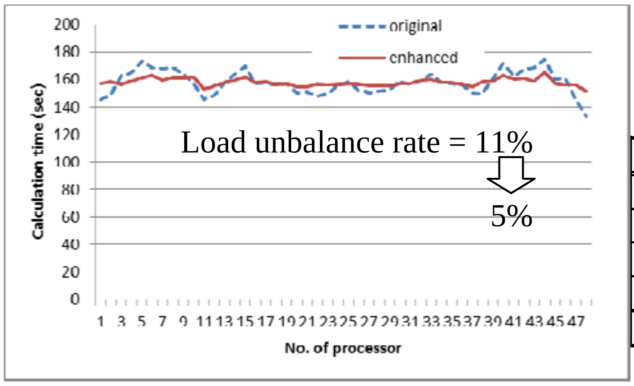
Figure 13. Change of load unbalance and elapsed time (MDB)



	current	new	rate of speedup	
Element	1747	1736	-10	0%
Contact	739	666	-72	-3%
Rigid	92	95	3	0%
Other	133	130	-3	0%
Elapsed	2710	2627	-83	-3%

Unit : second

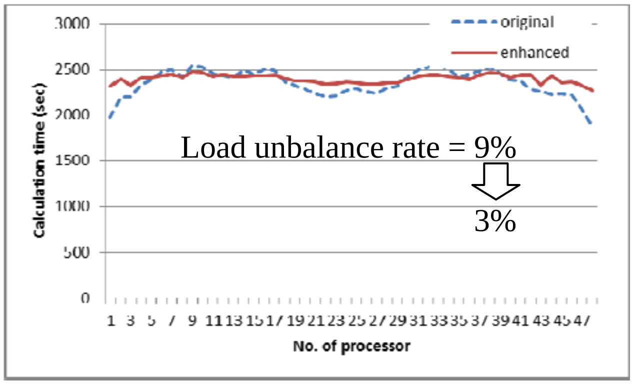
Figure 14. Change of load unbalance and elapsed time (Six Cars)



	current	new	rate of speedup	
Element	125	126	1	**1%**
Contact	51	49	-1	-1%
Rigid	33	33	0	0%
Other	10	10	0	0%
Elapsed	219	219	0	**0%**

Unit : second

Figure 15. Change of load unbalance and elapsed time (Neon)



	current	new	rate of speedup	
Element	1528	1558	31	**1%**
Contact	1043	1045	2	0%
Rigid	353	353	0	0%
Other	29	31	2	0%
Elapsed	2953	2987	34	**1%**

Unit : second

Figure 16. Change of load unbalance and elapsed time (3Cars)

Overall, depending on the severity of load unbalance, this enhanced domain decomposition method is able to reduce the load unbalance more than 50%. For those with large local deformation like ODB and MDB models, the above results demonstrate the enhanced domain decomposition speeds up job turnaround time by 15-20% through improving load unbalance. For the full frontal contact cases like Neon and 3Cars models, the original domain decomposition has already distributed the large deformation elements to each subdomain uniformly, therefore the benefit of the enhanced domain decomposition do not show on the elapsed time.

Figure 17 reveals the effect of job speedup relative to the original load unbalance using the information gathered from the above five models. Although the enhanced domain decomposition method can reduce load unbalance for all five models, this enhanced method has better impact in speeding up job turnaround on those models having more than 15% of load unbalance.

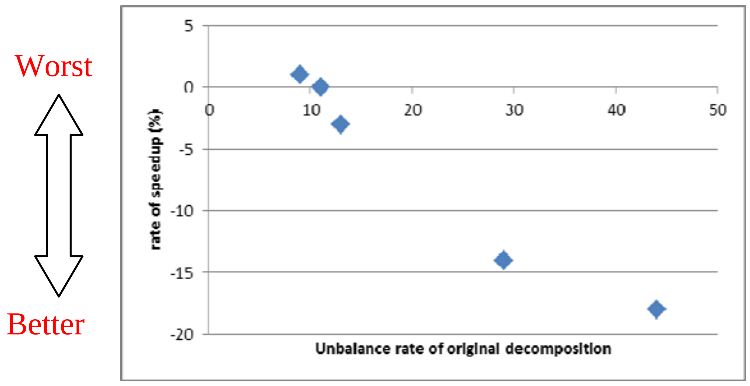


Figure 17. Relationship of Speedup Rate and Original Load Unbalance

5. Summary and Future Work

The enhanced domain decomposition developed by the author at Fujitsu can reduce load unbalance significantly. Such improvement leads to better job turnaround for the problems with large local deformation like ODB and MDB models. In analytical simulations for optimization of product structure, user executes a huge number of cases by changing data definition little by little for the same model. The basic characteristics of deformation and load balance of the model should not change dramatically. Hence this enhanced method would provide huge advantage to all the jobs during the product design cycle.

IIHS added small overlap front test for the evaluation of a vehicle's crashworthiness in 2012 [3]. Such test primarily affects a vehicle's outer edges not protected by the crush zone structures and resulting extensive local deformation, too. This enhanced decomposition method should offer certain benefit to the LS-DYNA users when handling the small overlap front crash simulation.

Unfortunately not all models can enjoy such benefit. Those models with uniform deformation in one direction, such as Neon model and 3Cars model, already have relative low load unbalance with the original decomposition method. Hence users of LS-DYNA cannot expect speedup with this enhanced method.

This enhanced method provides a critical step to distribute equal amount of calculation amount to all computing processors. The times used in this paper are the accumulation of calculation time of all processors after many iterative cycles of analysis. In order to improve job turnaround time beyond this study, it is necessary to look into detail of the iteration, investigate the mechanism of synchronization at each analytical cycle, investigate the communication patterns of LS-DYNA on the stages of Element, Contact, and Rigid in detail. LSTC has been constantly enhancing LS-DYNA over the years. Several features have been developed to improve the performance of these computing stages. For example, the feature of 'groupable contact' [4] improves the performance when a model involves multiple contact definitions. It is worthwhile to go through these features and check their impact on the performance by combining these features and to seek for enhancement.

The demand of computing power to deal with much more complicate and bigger model from the users of LS-DYNA has been growing continuously. Using a parallel system with hundreds or more processors to tackle one job has become a norm among the community of LS-DYNA. The author will continue his effort to enhance the performance of LS-DYNA.

References

[1] http://www.topcrunch.org/
[2] http://www.ncac.gwu.edu/
[3] http://www.iihs.org/iihs/ratings/ratings-info/frontal-crash-tests
[4] Brian, W.: Efficient Processing of Multiple Contacts in MPP DYNA, 8th European LS-DYNA Users Conference, Strasbourg, 2011Text

Acknowledgement

The author would like to express his gratitude to LSTC for their constant support.