 CMES, vol.59, no.3, pp.275-299, 2010

High Velocity Impact Simulation of Brittle Materials with Node Separation Scheme in Parallel Computing Environment

Ji Joong Moon[1], **Seung Jo Kim**[1] **and Minhyung Lee**[2]

Abstract: This paper describes the parallelization of contact/impact simulation for fracture modeling of brittle materials using a node separation scheme (NSS). We successfully demonstrated the fracture modeling of brittle materials using a cohesive fracture model. Since a NSS continuously generates new free surfaces as the computation progresses, the methodology requires increased computational time. To perform a simulation within a reasonable time period, a parallelization study is conducted. Particular methods for effective parallelization, especially for brittle materials, are described in detail. The crucial and most difficult strategy is the management of the data structure and communication needed to handle new contact nodes and surfaces. We also developed an efficient domain decomposition definition for the contact calculation, and implemented a user-controlled dynamic load-balancing algorithm. These efforts resulted in enhanced parallel performance as demonstrated by numerical test problems.

Keywords: Parallelization, contact-impact analysis, load balancing, brittle material, node separation.

1 Introduction

By the mechanical properties of brittle material such as high-strength and lightweight, brittle materials have been studied.[Ortiz and Pandolfi (1999); Hohler, Weber, Tham, James, Barker and Picku (2001); Lee (2003)]. An experimental approach, which often provides the most accurate results, is expensive and sometimes does not give detailed information. Simulations with continuum-based failure models [Johnson and Holmquist (1992); Johnson and Holmquist (1994)] have been developed that can only simulate failure of brittle material with the growth of a single dominant

[1] School of Mechanical and Aerospace Engineering, Seoul National University, Seoul, Republic of Korea.
[2] School of Mechanical and Aerospace Engineering, Sejong University, Seoul, Republic of Korea.

crack, and have difficulties with the prediction of dynamic behavior and the discrete nature of brittle failure. Recently, more realistic simulations have been performed using the cohesive-law fracture model [Camacho and Ortiz (1996)] with a node separation scheme (NSS). This model allows the adaptive creation of new surfaces and nodes for fracture modeling. However, a disadvantage is the increased contact calculation time with computational cycles. In some cases, it may not be possible to obtain results for practical problems within a reasonable time period. Addressing this issue is one of the motivations for this work. In order to simulate material fracture in penetration phenomena, the material point method (MPM) [Sulsky and Kaul (2004)] is known to be efficient [Steffen, Walstedt, Guilkey, Kirby and Berzins(2008)], and its parallelization has been studied [Huang, Zhang, Ma and Wang (2008); Ma, Lu, Wang, Roy, Hornung, Wissink and Komanduri (2005)]. MPM is an extension of the particle-in-cell method. Therefore, a Eulerian and a Lagrangian description are considered.

As computer technology has improved, parallelization has been applied to many scientific calculations. Because of material nonlinearity, complex contact calculations, and the high computing resource requirements of contact impact simulations, parallelization has been studied since the early 1990s as a way of overcoming these difficulties [Hoover, DeGroot, Maltby and Procassini (1995); Attaway and Hendrickson (1998); Paik, Moon, Kim and Lee (2006); Jason and Rober (2003)]. Attaway et al. developed a parallel contact simulation code called PRONTO3D and proposed the recursive bisection technique for static and dynamic load balancing to overcome the imbalance of calculation loads of each processor. However, they used a cubic-shaped contact search domain for simplicity, which causes unbalanced loading for practical problems. Their method, therefore, did not guarantee the best parallel efficiency due to the highly dense contact surfaces at the localized loading zone. Further discussion will be given in Section 2. Paik [Paik, Moon, Kim and Lee (2006)] used the same calculation domain for internal force calculations and contact search calculations to reduce the communication cost.

We developed parallel procedures for practical contact/impact problems for both ductile and brittle materials. Our approach includes parallelization of NSS to simulate the fracture of brittle material using the domain decomposition technique, and a static and dynamic load balancing scheme that is highly efficient even for nonuniformly-meshed problems. Two numerical examples were developed to evaluate the proposed method's accuracy and parallel performance. One example is a normal impact onto a metallic multi-layered plate, and the other is a glass-metal bar impact problem. Both computations were compared with experimental results.

2 Difficulties in Parallelization of Contact/Impact for Brittle Materials with NSS

The typical difficulties during parallel simulation of contact/impact problems are domain decomposition and an imbalanced load. Although relatively good parallel performance with respect to these effects was reported in the previous work [Attaway and Hendrickson (1998); Paik, Moon, Kim and Lee (2006)], the work involved only simple contact problems. The impact/penetration of brittle materials using NNS becomes much more complicated because of the imbalanced load associated with highly localized loading and contact zone, and because of the newly-generated contact surfaces and nodes that require sophisticated data management procedures. We now describe certain details about difficulties in the parallel strategy.

First, special consideration has been given to the contact search domain definition. Currently, the recursive coordinate bisection method with a simple cubic shape contact search domain is known to be an efficient domain decomposition method for contact impact problems. However, when we tested this simple cubic shape contact search domain for nonuniformly meshed problems, significant unnecessary contact calculation and communication was observed. The parallel efficiency shown in the previous study [Attaway and Hendrickson (1998); Paik, Moon, Kim and Lee (2006)] is only guaranteed for uniformly-meshed problems, and a uniform mesh system is not common in practical modeling. This unnecessary communication and contact calculation becomes more significant when we use NNS because NNS generates a great number of new nodes in a localized area, causing an increasing unbalance. We propose a new arbitrary shape contact domain definition with recursive coordinate bisection instead of a simple cubic shape contact domain.

Second, the element erosion scheme, which eliminates highly strained elements, is widely applied, especially for large deformation problems. Therefore, we need to redefine contact surfaces due to the new surfaces, and simply we added a contact surface index list for the serial calculation. However, it is necessary to impose synchronization of the new node and new surface index at the domain boundary in a parallel calculation. An efficient synchronization scheme to reduce communication overhead and manage data structure is required, especially we use the NSS. If the newly generated nodes are indexed independently on each domain, the same calculation error observed in the element erosion part can occur for NSS, too. In this study, this difficulty is removed.

Finally, we consider the imbalanced calculation load among processors over time. In general contact/impact problems, the imbalanced load is partly attributed to the dynamic movement of nodes. For brittle materials with the NSS, the newly-

generated nodes cause further significant imbalance in the calculation load. Imbalance due to the new nodes is the most severe. Therefore, the dynamic load balancing is necessary in most of parallel contact/impact problems in every time step. However, it is not necessary to perform dynamic load balancing at each time step with an arbitrary contact domain definition we proposed. The details of arbitrary contact domain definition will be discussed in Section 4.

3 FEM Implementation of the Cohesive Law with NSS

3.1 Numerical schemes

The operate-splitting procedure [Zienkiewicz and Codina (1995)] was applied to the formulation of governing equations. The method was described by Yoo and Lee [Yoo and Lee (2002)]. Instead of the commonly-used hexahedral element, we used a four node tetrahedral element. By using the tetrahedral element, no stabilization scheme is required and contact surface can be defined naturally. A modified central difference scheme, which reduces roundoff error and eliminates the nonzero strain rate for pure rigid body rotation, was applied for time stepping. In addition to elastic and elastic-plastic material models, the Johnson-Cook material model is available in our code for ductile material, and the Mohr-Coulomb [Glenn (1976)] model with a node separation scheme is implemented for brittle material. Contact search and contact enforcement are important procedures in dynamic analysis, and most of the time is consumed by these procedures. In this work, the bucket sorting method and position code algorithm [Oldenburg and Nilsson (1994)] were used for contact search, and the defense node algorithm was implemented for contact enforcement. To prevent large mesh distortion, the element erosion method was applied based on the predefined effective plastic strain. For a ceramic material, edge-to-edge contact is implemented to prevent edge penetration.

3.2 Node separation scheme (NSS) based on the cohesive law

Unlike metals, brittle materials generally have very little ductility and high strength. However, it makes difficult to examine its characteristic of fracture by the brittleness and degraded elastic properties of brittle material after failure experimentally and numerically. For our explicit simulation, we implemented the cohesive-law with a node separation scheme in which all the element surfaces were assumed to be potential crack paths. The fracture criterion considering an effective stress intensity factor for mixed mode fracture [Camacho and Ortiz (1996)] is given by

$$\sigma^{eff} = \sqrt{\sigma^2 + \beta_\tau \tau^2} \geq \sigma_{in} \tag{1}$$

where β_τ is a shear stress factor that was set to zero considering only a mode I crack,

σ^{eff} is the effective stress, and σ_{in} is an incident stress amplitude that converges to fracture stress σ_{fr} as time increases. The fracture stress is expressed as

$$\sigma_{fr} = \frac{K_{ic}}{\sqrt{\pi a_0}} \tag{2}$$

where K_{ic} is the fracture toughness and a_0 is the size of a preexisting flaw. Then, the incident stress amplitude is given by

$$\sigma_{in} = \sigma_{fr} \frac{e^{t/t_c}}{e^{t/t_c} - 1} \tag{3}$$

where t_c is a characteristic relaxation time that is a function of the density ρ, the speed of sound speed c, the fracture energy G_e, and the fracture stress σ_{fr}as follows

$$t_c = \frac{\rho c G_e}{2\sigma_{fr}^2} \tag{4}$$

Whenever the fracture criterion in Eq. (1) is satisfied, a new node and surface are generated. Brittle material cannot endure a large amount of plastic deformation. The region, which has undergone plastic deformation, is assumed to be full of microcracks. More details are found in [Camacho and Ortiz (1996)]. Once a fractured surface is detected, the surface must be duplicated by a node separation scheme. Thus, it is required to build an efficient data structure and memory allocation system. A basic data structure concept is explained as follows for a serial procedure, and will be discussed further in Section 4 for the parallel procedure. As shown in Fig. 1, we used the element connectivity as the basis for the NSS.

The detailed procedure is as follows:

(a) The elements that adjoin a specific node (N0) are sorted by connectivity.

(b) Sorted elements are rearranged the order by adjacency of element surface, and mark the fractured surface.

(c) Check if there is a fractured surface from first element, and mark the first fractured surface position. Keep checking the second fractured surface, and if there is a second fractured surface, then a new node index (N1,N2,N3) is given instead of the old node index (N0) for the elements between the first and second fractured surface.

(d) Update the new surface index instead of the fractured surface for the element that has an updated node index.

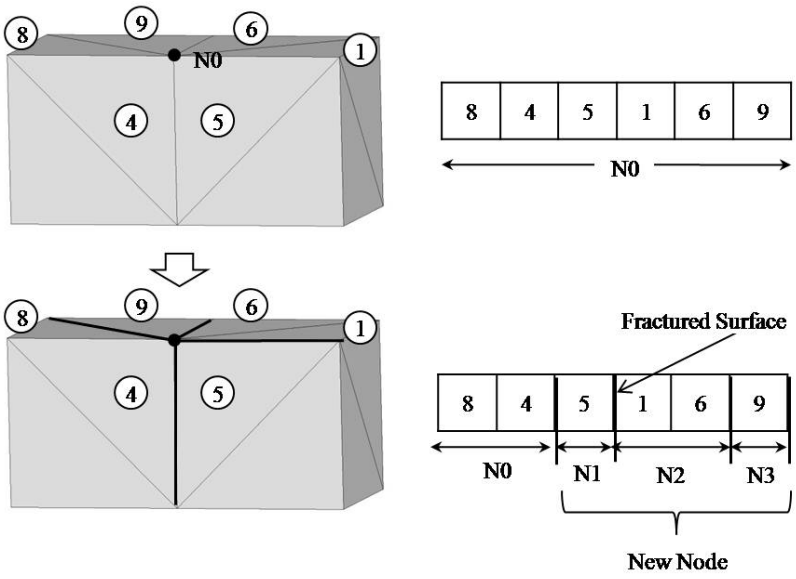


Figure 1: Node separation scheme.

4 Parallelization

4.1 Arbitrary contact search domain definition

The first step in parallelization is the allocation of calculation load to each processor; i.e., the so-called domain decomposition. This should guarantee equally-shared load allocation. In general, this is achieved by the minimization of boundary nodes since for most problems, communication takes place at or around the boundary between each domain. For contact-impact simulations, however, both the internal force calculation domain and contact calculation domain are important [Paik, Moon, Kim and Lee (2006)], and the load for contact calculation is much higher. However, if we make different domains for internal force and contact calculations, extra communication is required. Considering these effects, the internal force calculation domain is decomposed with a consideration of the contact force calculation. And the static load balancing is achieved by a recursive coordinate bisection scheme [Attaway and Hendrickson (1998)], which ensures that each domain has nearly the same number of nodes at the contact search surface.

Once the domain is decomposed, the contact calculation domain must be defined. Paik [Paik, Moon, Kim and Lee (2006)] introduced a 3-D simple box type of contact calculation domain for a uniform mesh size problem. However, it is not effi-

cient for the non-uniform mesh system in which a dense mesh is usually used for a contact region (creased area in Fig. 2). The NNS in our program further increases the load imbalance, which is inevitable when modeling fragmentation phenomena. To resolve this, a new arbitrary shape contact domain definition (rather than a simple cubic shape contact domain) is introduced in two stages.

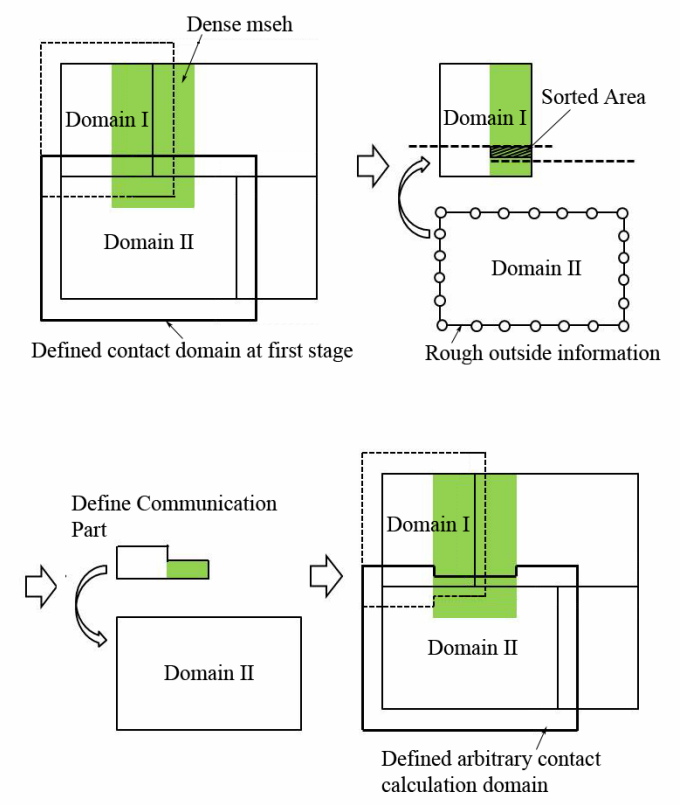


Figure 2: Arbitrary shape contact search domain definition.

In the first stage, Paik's domain definition was used to define a rough domain size and to make a list of communication domains. In the second stage, outside information is communicated between neighboring domains to build the actual communication information. As shown in Fig. 2, the outside information of Domain II was first sent to Domain I and then communication part of Domain I for Domain II is modified. With this procedure, the contact calculation domain of Domain II became an arbitrary shape. This process is very effective to reduce calculation and communication overhead on regions with significant mesh size gradients. The

detailed procedures are as follows:

(a) At the first stage, each domain calculates minimum and maximum coordinate values for a 3-D box ($X_{\min}$, $X_{\max}$, $Y_{\min}$, $Y_{\max}$, $Z_{\min}$, $Z_{\max}$). These variables are broadcast to make a neighboring domain list.

(b) Make the approximate outside information, and these are sent to neighboring domain.

(c) At the received domain, for the elements that are inside the 3-D box of the neighboring domain, communication elements are sorted by

$$D \leq \alpha \times \min(l_0, l_0, l_0, \cdots, l_m) \tag{5}$$

where α is the safe factor (we usually used 2.0), l_i is the element characteristic length, and D is defined as

$$D = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2} \tag{6}$$

where i is the element index within the 3-D box of the neighboring domain, and j is the approximate outside node index.

(d) Sorted elements are sent to a neighboring domain and independently perform contact searches and contact enforcement.

During the two-stage contact domain definition, an efficient method is adopted in order to reduce communication cost. In our work, instead of communicating all outside node coordinates, only reference cells are communicated. First, the domain is defined as

$$(x_{\min}, x_{\max}, NCX) \tag{7}$$

$$(y_{\min}, y_{\max}, NCY) \tag{8}$$

$$(z_{\min}, z_{\max}, NCZ) \tag{9}$$

where $x_{\min}$ and $x_{\max}$ are the two extreme coordinates, NCX is the number of cells along the x direction, and $NCX \times NCY \times NCZ$ is the total number of cells. The cell index in which node i is located can be found as

$$ICELL = IX + IY \times NCX + IZ \times (NCX \times NCY) \tag{10}$$

where

$$IX = (int)(x_i - x_{\min})/\Delta x \tag{11}$$

$$IY = (int)(y_i - y_{\min})/\Delta y \tag{12}$$

$$IZ = (int)(z_i - z_{\min})/\Delta z \tag{13}$$

$$\Delta x = (x_{\max} - x_{\min})/NCX \; \Delta y = (y_{\max} - y_{\min})/NCY \; \Delta z = (z_{\max} - z_{\min})/NCZ \tag{14}$$

We used a one-dimensional data (1-D) structure ($CELL[\ ]$) of size $NCX \times NCY \times NCZ$. If $CELL[ICELL] = 0$, then the node index is stored in the data structure. Otherwise, this procedure is repeated for every node. The schematic reference cell plane ($x-y$plane) is shown in Fig. 3.

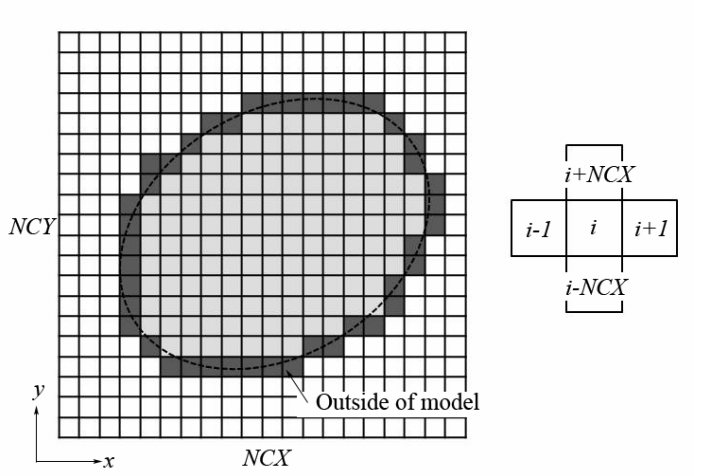


Figure 3: Schematic of reference outside using reference cell.

The outside reference cell is calculated as follows:

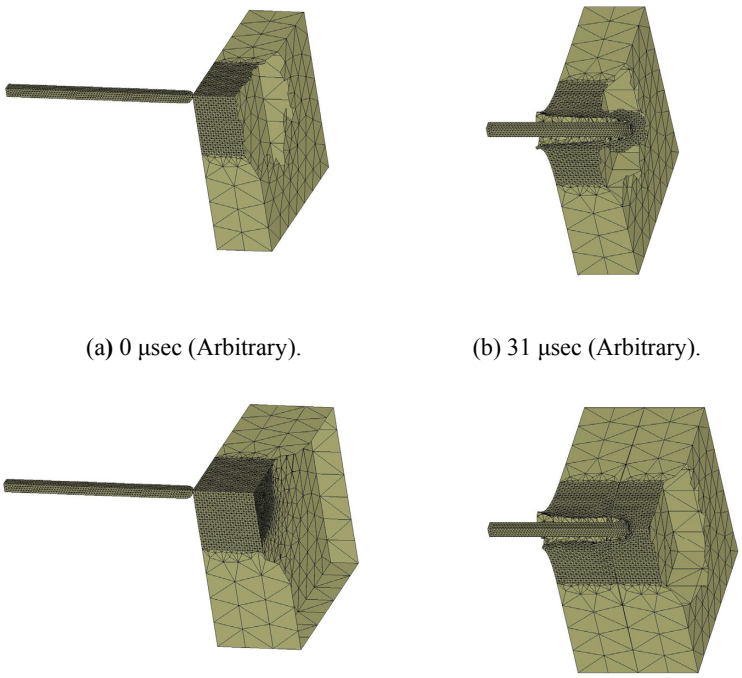
$$C_{i-1} + C_i + C_{i+1} + C_{i-NCX} + C_{i+NCX} < 5 \tag{15}$$

where

$$C_i = \begin{cases} 1 \; if \; CELL[i] > 0 \\ 0 \; else \end{cases} \tag{16}$$

The node coordinate information of only the reference cell is now ready to be sent to the neighboring domain. We tested the efficiency of the new contact domain definition. The test problem was the impact of a long body into a steel block. Eight processors were used to solve this problem. In Figs. 4(a) and (c), the simple cubic shape and arbitrary shape contact search domains are illustrated for comparison.

This is the domain allocated at the early time step to, for example, the 1^{st} processor. As expected, the contact domain was modified such that a certain area was eliminated in the arbitrary shape contact domain. The number of contact surfaces for the arbitrary shape domain was 5,467, and 7,284 for the simple cubic shape domain. A 25% reduction in contact calculations and communication overhead can be achieved.



(a) 0 µsec (Arbitrary). (b) 31 µsec (Arbitrary).

(c) 0 µsec (Cubic). (d) 31 µsec (Cubic).

Figure 4: Comparison of contact search domains between arbitrary shapes and simple cubic shapes at early and mid-stages.

The elapsed computation time used only in the contact calculation is shown in Fig. 5. The pure communication time between processors is also compared in the figure. The computation cost for the simple cubic shape contact domain case was more than twice that of the arbitrary shape contact domain case. We obtained a significant reduction in communication cost by maintaining an optimum arbitrary shape contact domain definition.

Figure 6 compares the calculation and communication overheads of some proces-

sors (e.g., the 1^{st} and 6^{th} processors). Here we used the total number of communication elements as an overhead indicator. The difference between a simple cubic and arbitrary contact domain is more than twofold. It is evident that parallel performance can be improved with the arbitrary shape contact domain definition. For the 1^{st} processor using a simple cubic shape, there was a sudden increase in calculation time and in the total number of elements around 42 μsec. This is attributed to dynamic load balancing since the domain is decomposed along with a dense mesh zone. Thus, we note that without a sophisticated contact domain definition, the effect of dynamic load balancing could be diminished.

Figures 4(b) and (d) compare the contact search domain decomposed at mid-timestep without the dynamic load balancing technique. As the long rod moved forward, the contact calculation load increased for the simple cubic shape contact domain case, causing the imbalance contact calculation load. In this case, the continuous application of the dynamic load balancing scheme is required. This is not necessary in the arbitrary shape domain case. We observed only a slight increase in the contact calculation load. Therefore, one advantage is that there is no need to conduct an expensive dynamic load balancing technique at every timestep.

4.2 Parallelization of node separation scheme (NSS)

For parallelization of the node separation scheme, the data structure management system for sharing the index of the new node and new surface between domains is the key issue because calculations are performed based on the node and surface index. This is illustrated in Fig. 7. For a serial procedure, a new node can be indexed sequentially. However, this is not the case for a parallel procedure that requires synchronization of index by communication. In this work, we implemented a new parallel node counting scheme.

Consider the data structure shown in Fig. 1. Suppose *NFi* fracture surfaces are created on node *i*. Then, the number of new nodes which should be created at the same location of node *i* is

$$(NF_i - 1). \tag{17}$$

This is stored in 1-D array *NC*[ ], and later this domain array is gathered at the master processor by the *MPI_Gather* function. A new start node index is computed by

$$SNID_i = \begin{cases} GN \ (if \ i = 0) \\ SNID_{i-1} + NC[i-1]/ND_i \ (if \ i > 0) \end{cases} \tag{18}$$

where ND_i is the number of domains that are adjacent to the ith node, and GN is the total number of nodes. Once a new start node index$SNID_i$ is computed for all

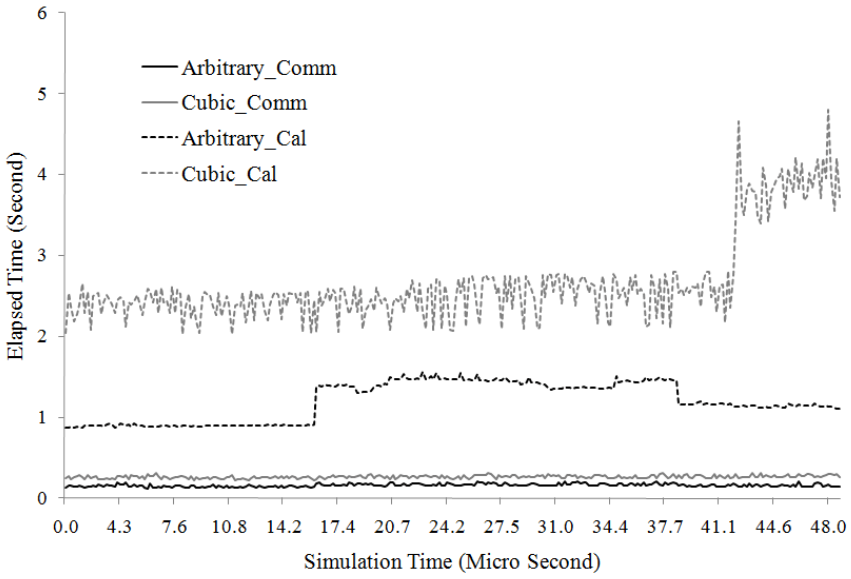


Figure 5: Comparison of elapsed time for contact calculation, and communication between arbitrary shape and simple cubic shape contact domain definition.

nodes, the *NC*[ ] array is sent back to the domains using the *MPI_Bcast* function and a new node index is created. This data structure for the parallel node separation scheme is illustrated in Fig. 8. For example, the node index of N1 is *GN*, N2 is *GN*+1, N11 is *GN*+10, etc.

A similar procedure is required for synchronization of the surface index. This is not so complicated because there is only one new surface for each fractured surface. The new surface index NNS_i is computed by

$$NNS_i = NS_i + GS \tag{19}$$

here NS_i is the i th surface index and GS is the total number of element surfaces. After the index of the new node and surface is defined, the new node and surface are created by the same method of serial procedure without any communication among processors.

4.3 Triggered dynamic load balancing (TDLB)

As a calculation involving large deformation progresses, the mesh structure, and the total number of nodes and surfaces are continuously changed. This causes a calculation load imbalance of the domain, resulting in poor parallel performance.

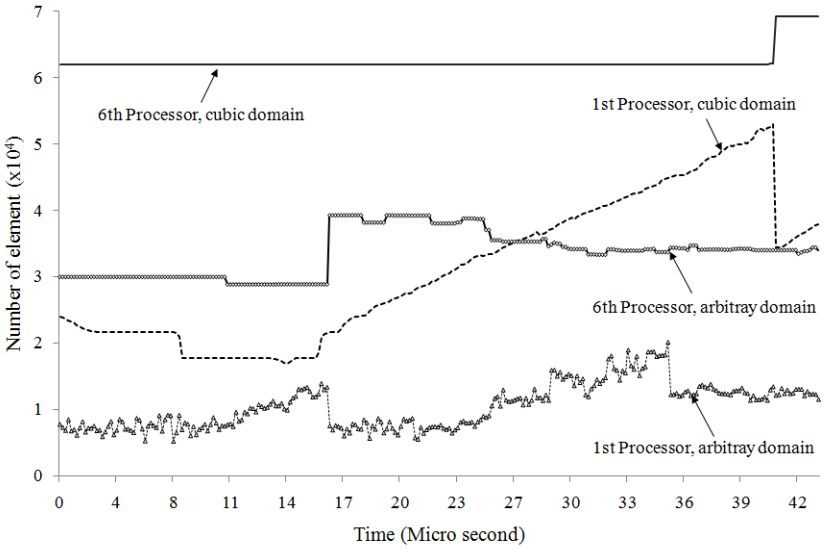


Figure 6: Comparison of calculation load at 1st and 6th processor (cubic vs. arbitrary).

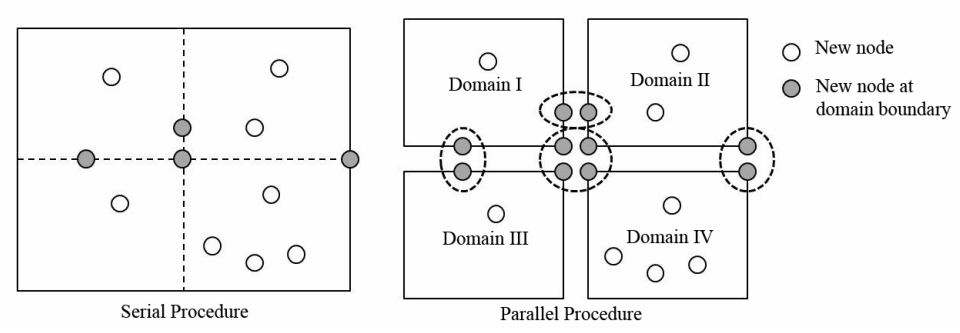


Figure 7: New node indexing of NSS.

The imbalanced calculation load becomes more severe if we model the brittle material fracture with a node separation scheme. To improve parallel performance, a dynamic load balancing scheme to reallocate the calculation load is required. In this study, the parallel recursive coordinate bisection method [Attaway and Hendrickson (1998)] was implemented. We introduced a new ratio value to trigger the initiation of dynamic load balancing to prevent to conduct of dynamic load balancing. The flowchart of this scheme is shown in Fig. 9.

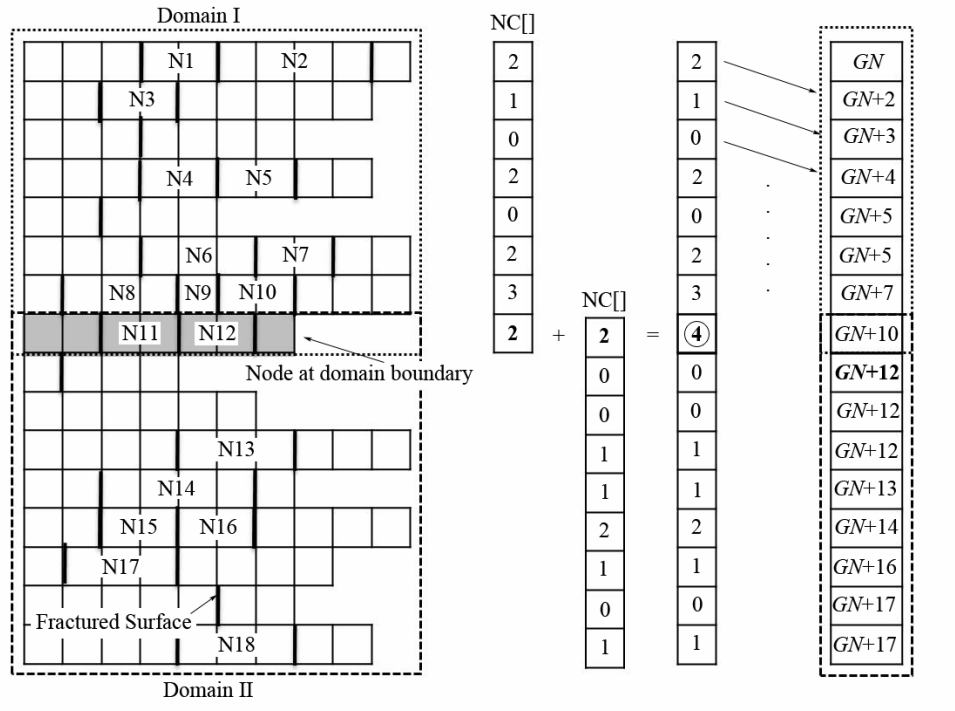


Figure 8: Data structure for parallel counting of new node index.

The total number of outside nodes of each domain stored at the first timestep is $Nref_j$, and at each timestep we calculated the total number of outside nodes of each domain$Nout_i$. If $Nout_i$ was bigger than $\alpha Nref_j$ in any domain, then domain decomposition was performed by broadcasting the value of Ias 1. α is the allowable unbalanced calculation load factor, and 1.2 or 1.5 was used in this study. After decomposition, $Nref_j$ was updated as $Nout_i$.

A master-slave communication model and unstructured communication pattern were implemented for domain decomposition as shown in Fig. 10. It is known that the master-slave model is not recommended for parallel performance; thus, it was used only for node coordinate and position data communication.

The details are as follows:

(a) The coordinate information of the node at the slave processor is sent to the master processor.

(b) The domain is decomposed recursively at mater processor.

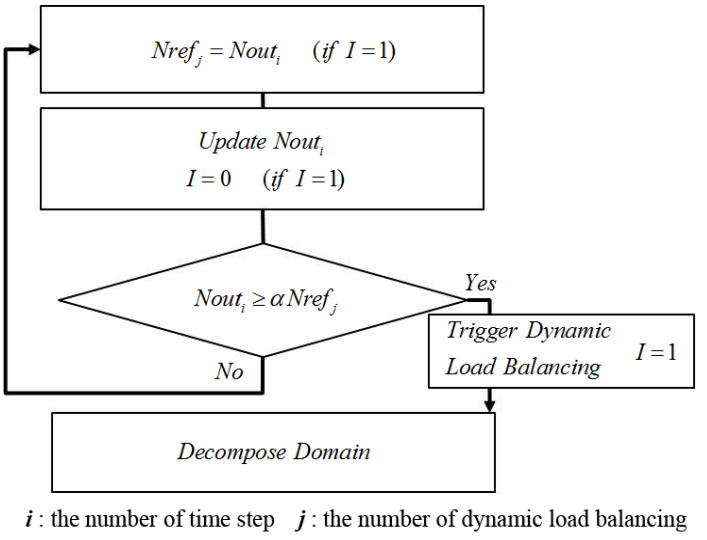


Figure 9: Initiation of dynamic load balancing.

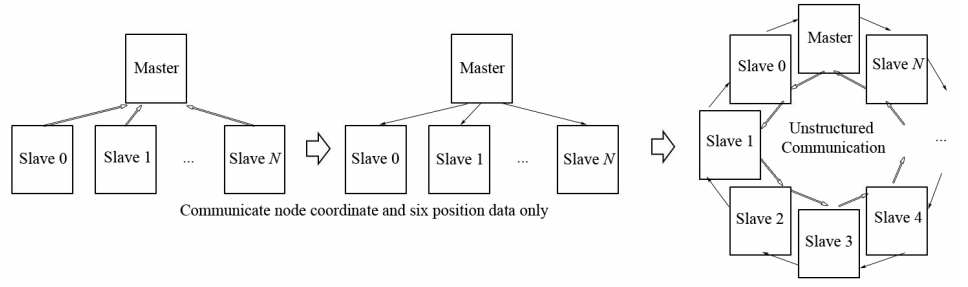


Figure 10: Communication model for dynamic load balancing.

(c) The decomposed domain information (six double variables per domain) is sent to the slave processors to make a neighboring domain list for the unstructured communication [Attaway and Hendrickson (1998)].

(d) Simulation information is updated by unstructured communication.

This dynamic load balancing is closely related to the definition of the contact calculation domain. For most simulations solving metallic media that do not need a node separation scheme, the number of outside node for contact calculation is not usually reached to the number of allowable unbalanced outside node($\alpha Nref_0$)

defined at first timestep because of effect of arbitrary contact domain definition. Sometimes, no dynamic load balancing is triggered; however, this is not the case for brittle materials. This dynamic load balancing method is very effective for modeling brittle materials using a node separation scheme, as will be demonstrated in Section 5.

5 Numerical Examples

5.1 Testbed and scalability

The Linux operating system is very popular for clusters. Another option is the Windows operating system, or so-called Windows HPC Server developed by Microsoft. In this study, the testbed cluster system was built with the Windows operating system and the Infiniband (10 Gbps) network system for interconnecting nodes. A four node (32 core) cluster system was used. The details of the system are given in Table 1. The theoretical peak performance is 409.6 Gflops (giga floating point operations per second).

Table 1: Details of testbed cluster

CPU	Intel Xeon 3.20 GHz quad core (2 processors a node)
Memory	256 Gbytes (64 Gbytes a node)
HDD	2.4 Tbytes (600 Gbytes a node)
Network	Infiniband (10 Gbps)
OS	Windows HPC Server edition
MPI	MS-MPI
Compiler	Visual Studio 2005 with parallel debugger

The parallel performance can be evaluated by a scaled speedup test defined as

$$Scalability = \frac{T_{N processors}}{T_{1 processors}} \tag{20}$$

where $T_{N processors}$ is the computational time with N processors and $T_{1 processors}$ is the time with one processor. There are two tests for measuring speedup: fixed sized speedup and scaled sized speedup. In most cases, it is more difficult to achieve good performance in the fixed sized speedup test due to the fixed problem size, independent of the number of processors. In our case using the node separation scheme, the total number of nodes cannot be estimated initially. Therefore, the fixed sized speed up test was only used for performance evaluation.

To evaluate accuracy and parallel performance, two test problems were simulated as follows.

5.2 Long rod normal penetration

This is a benchmark example for ductile material. The initial mesh model for the first test problem is shown in Fig. 11. The material properties are indicated in Table 2. The striking long rod has an L/D ratio of 10.7 and impacts six piled plates (each with 22 mm thickness). The impact velocities were 1.057, 1.301, and 1.5664 km/s. The final penetration depth is compared with experiment results. The erosion criterion is 200% of the effective plastic strain. The termination time is 140 μsec. We ran this problem with 8 cores.

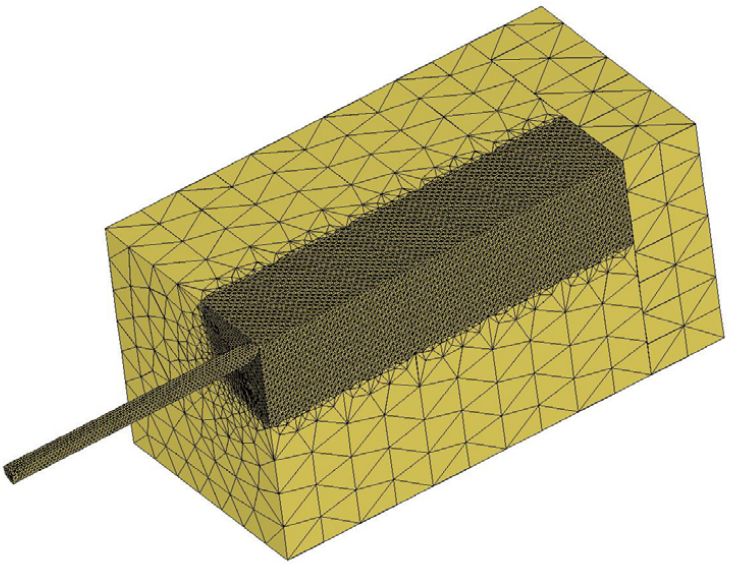
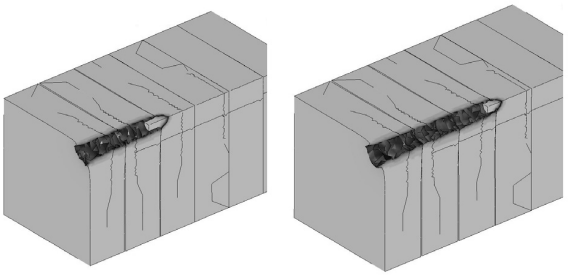


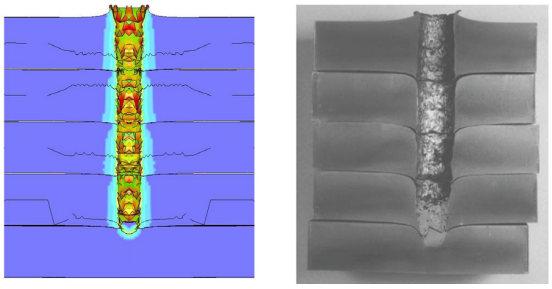
Figure 11: Initial mesh of normal long rod penetration.

Figures 12 and 13 show experimental and simulation results. The errors are within 5%. From the results, we conclude that the numerical formulation, contact enforcement, erosion algorithm, and material model work well for a parallel calculation. The final crater shape in Fig. 12(c) is similar to the experiment results in Fig. 12(d). The crater diameter is slightly smaller than the experimental data. This can be attributed to the uncertainty in the material properties used in the simulation.

Next, parallel performance was evaluated and the results are shown in Fig. 14. The speedup with 32 cores was 23.79, and this corresponds to 74.34% of the ideal speedup performance. Dynamic load balancing was not triggered when the allowable unbalanced calculation load factor was set to 1.5. Good performance was



(a) Equivalent plastic strain (1.057 km/s). (b) Equivalent plastic strain (1.301 km/s).



(c) Equivalent plastic strain (1.5664 km/s). (b) Experimental results(1.5664 km/s).

Figure 12: Results of normal penetration.

Table 2: Material properties for the Johnson-Cook model

	WHA	Mild Steel
Young's Modulus (GPa)	389.16	200.10
Poisson ratio	0.28	0.30
Density (g/cm^3)	17.3	7.87
B (GPa)	0.22326	0.22946
C	0.0221	0.02740
N	0.10841	0.30240
M	1.0	1.0

achieved only with the use of arbitrary contact calculation domain definition in the case. No heavy calculation load changes occurred in this problem. This was not the case for the next example problem.

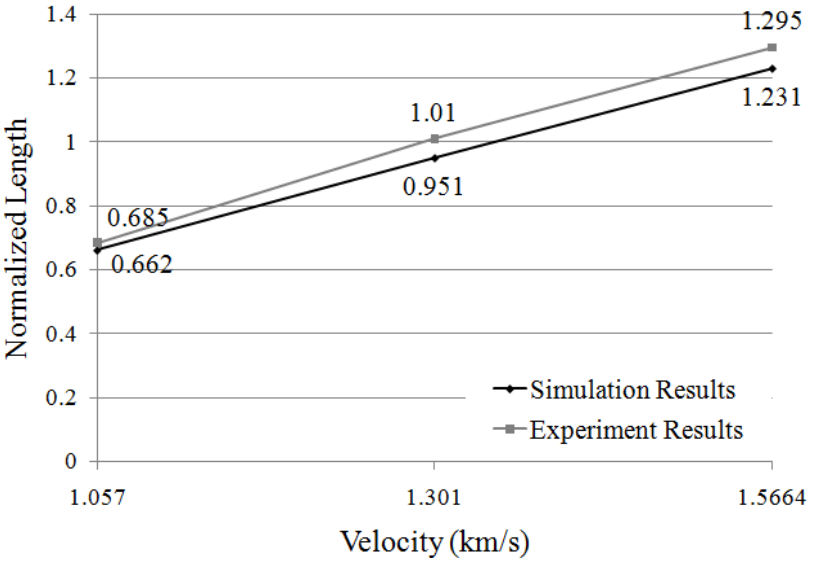


Figure 13: Comparison of penetration depth between simulation and experiment.

5.3 Glass-Metal Bar Impact

To evaluate the parallel cohesive law with NSS, a of glass and metal rod impact numerical experiment was performed [Repetto, Radovitzky and Otriz (2000)]. The glass bar had a diameter of 12.7 mm and a length of 170 mm. The other is a steel bar with the same diameter. On the symmetry plane, only one-half of bars were modeled. The material properties are indicated in Table 3. The initial model configuration decomposed by 16 domains is shown in Fig. 15. The Impact velocity was 210 m/sec, and the fracture stress of the glass was assumed to be 0.1 Gpa. The total number of element was 145,754.

The simulation results are shown in Fig. 16. Shortly after impact, surface cracks on the contacting interface grew into the interior of the bar and axial cracks propagated in the impact direction. As fragmentation progressed, formation of the failure front, which propagates in the direction of the axis of the bar, was observed. The radial expansion of the glass in the failure, which is one of the characteristics of brittle failure, was also observed.

Initially, the total number of nodes was 33,520 and this increased up to 84,821 at termination time. As shown in Fig. 16, the node separation scheme generated most of the new nodes at a certain part of model, thereby causing an imbalance in the computational load. As an example, the total number of outside nodes in the 4^{th}

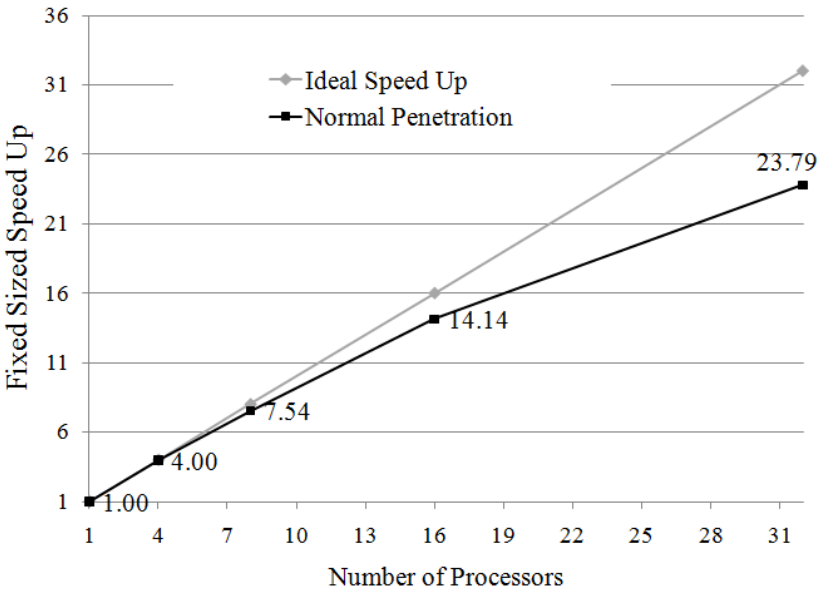


Figure 14: Fixed sized speedup results (1.5664 km/s).

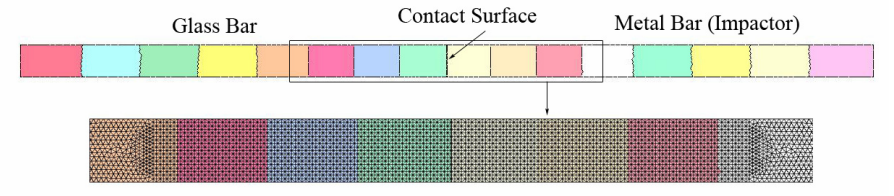


Figure 15: Model configuration and decomposed domain (16 processors).

and 16^{th} domain is indicated in Fig. 17. In this case, 18 dynamic load balancing was performed due to the unbalancing of the calculation load. More nodes were generated in the 4^{th} domain which is located near the impact region. It can be seen that the calculation load is balanced (the total number of outside nodes was reduced) whenever dynamic load balancing was performed. We also note that each domain boundary changed, as indicated in Fig. 16.

The time consumed for each part of the calculation is shown in Fig. 18. As expected, most of the time was used in contact calculation and contact communication. This is because we solved both the node-to-surface contact and the edge-to-edge contact. We note that the overhead for an arbitrary contact calculation

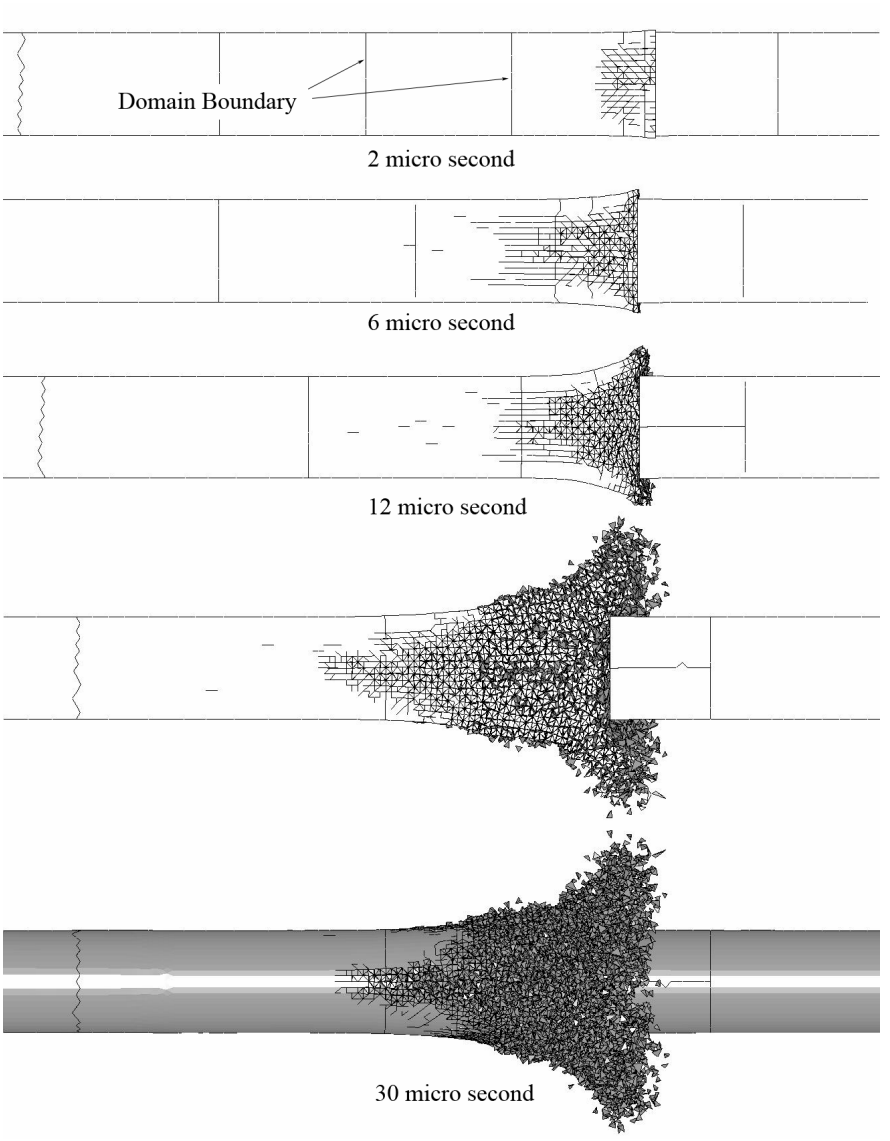


Figure 16: Simulation results.

definition it not significant. The fixed size speedup result is shown in Fig. 19. We obtained 77% of the theoretical performance with 16 processors, and 58% with 32 processors. This is a promising result considering the highly imbalanced calculation load, communication load, and complexity of the calculation.

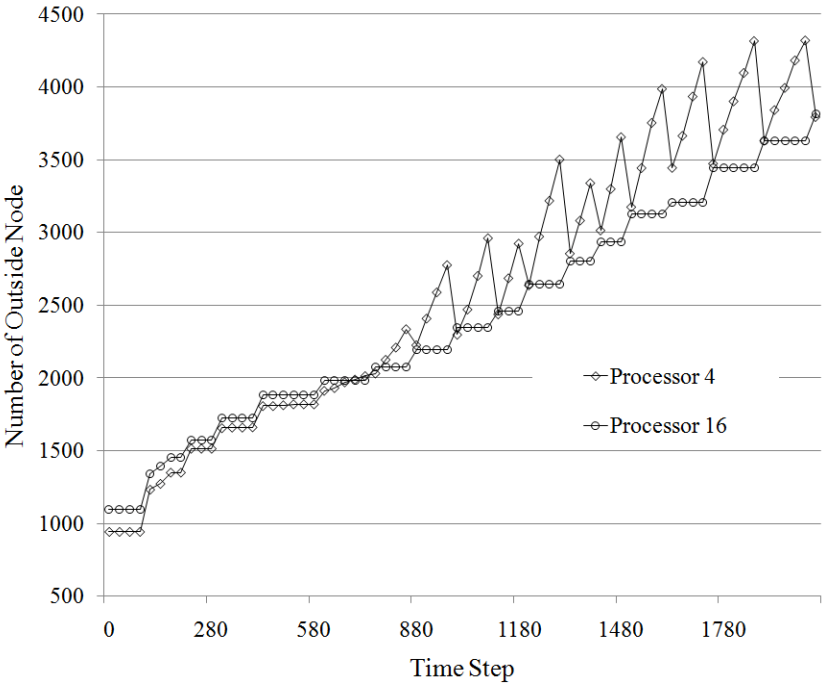


Figure 17: Comparison of number of outside nodes (16 processors).

6 Conclusion

We successfully demonstrated the parallelization of a fracture model for brittle materials using a node separation scheme (NSS). An efficient data structure of new nodes and surfaces was designed, and a parallel new node counting procedure was developed for synchronization of a node and surface index throughout the domain. To enhance parallel performance, arbitrary shape contact domain definition using a two-stage procedure was introduced instead of a simple shape cubic contact domain definition. With this treatment, the communication and calculation overhead of neighboring domains was reduced by more than half. A triggered dynamic load balancing (TDLB) scheme was also developed using a ratio factor to trigger the load balancing. Master-slave communication and an unstructured communication pattern was used for domain decomposition.

To evaluate accuracy and parallel performance, two test problems were simulated: a normal impact of a long rod into multi-layered plates, and a glass-metal bar impact. Good agreement with experimental data was obtained. A fixed sized speed-up test

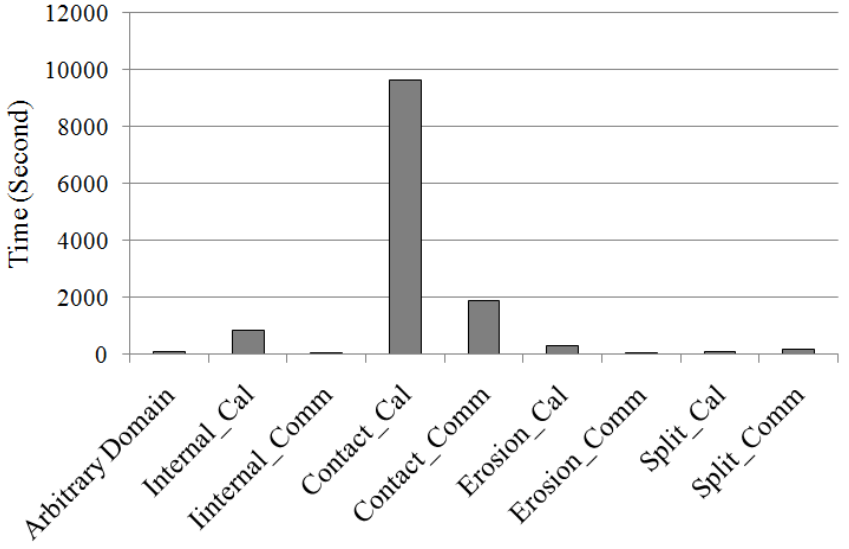


Figure 18: Elapsed time distribution (16 processors).

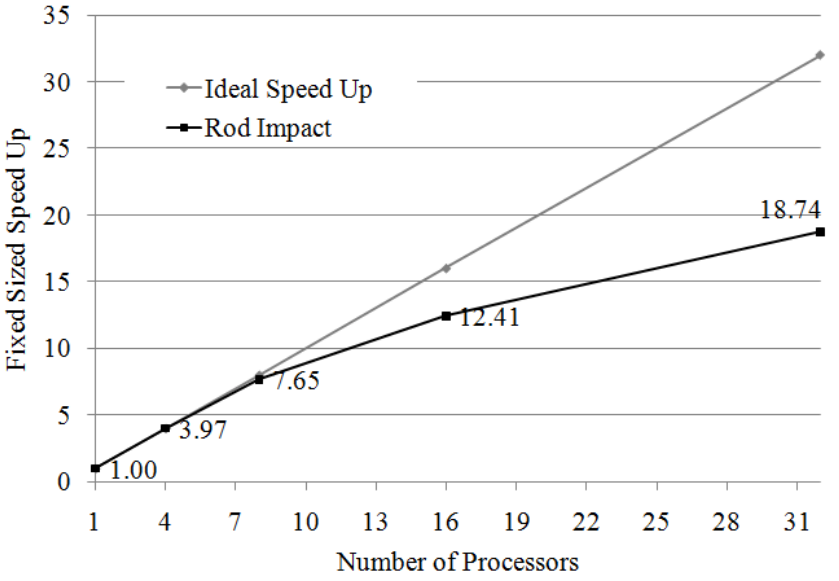


Figure 19: Fixed sized speedup results.

was performed for evaluation of parallel performance. For the first test problem that modeled only ductile materials, we obtained a 23.79 speedup with 32 cores. For the second test problem that modeled brittle materials with NSS, an 18.74 speedup was obtained. This is a promising result, especially considering the highly imbalanced calculation load caused by the continuous addition of new nodes and surfaces from fragmentation.

Acknowledgement: The work was supported by the Agency for Defense Development of the Korean government under the Basic Research Program (Contract UD040012AD).

References

Attaway S.W.; Hendrickson B.A.; Plimpton S.J.; Gardner D.R.; Vaughan C.T. (1998): A Parallel contact detection algorithm for transient solid dynamics simulation using PRONTO3D. *Computational Mechanics*, vol. 22, pp. 143-59.

Berger, M.J.; Bokhari, A. (1987): A partitioning strategy for nonuniform problems on multiprocessors. *IEEE Trans. Computers*, vol. 36, pp. 570-580.

Camacho, G.T.; Ortiz, M. (1996): Computational modeling of impact damage in brittle materials. *Int J Solids Structure.*, vol. 42, pp. 1397-1434.

Glenn, A. (1976): The fracture of a glass half-space by projectile impact. *J Mech. Phys Solids.*, vol. 24, pp. 93-106.

Hohler, V.; Weber, K.; Tham, R.; James, B.; Barker, A.; Picku, I. (2001): Comparative analysis of oblique impact on ceramic composite systems. *Int J Impact Eng*, vol. 26, pp. 333–344.

Hoover, C.G.; DeGroot, A.J.; Maltby, J.D.; Procassini, R.D. (1995): Paradyn: Dyna3d for massively parallel computers. *Presentation at Tri-Laboratory Engineering Conference on Computational Modeling.*

Huang, P.; Zhang, X.; Ma, S.; Wang, H.K. (2008): Shared Memory OpenMP Parallelization of Explicit MPM and Its Application to Hypervelocity Impact. *CMES: Computer Modeling in Engineering and Sciences*, vol. 38, no. 2, pp. 119-147.

Johnson, G.R.; Holmquist, T.J. (1992): A computational constitutive model for brittle materials subjected to large strain, high strain rates and high pressures. *In: Meyers MA, Murr LE, Staudhammer KP, editors. High strain rate phenomina in materials. New York: Marcel Dekker Inc.*, pp. 1075–1085.

Johnson, G.R.; Holmquist, T.J. (1994): An improved computational constitutive model for brittle materials. *In: High Pressure Science and Technology 1993. New York: American Institute of Physics*, pp. 981-984.

Jason, H.; Rober, E. (2003): A parallel finite element procedure for contact-impact problems. *Engineering with Computers*, vol. 19, pp. 67-84.

Lee, M. (2003): Hypervelocity impact into oblique ceramic/metal composite systems. *Int J Impact Eng.*, vol. 29, pp. 417–424.

Ma, J.; Lu, H.; Wang, B.; Roy, S.; Hornung, R.; Wissink, A.; Komanduri, R. (2005): Multiscale Simulations Using Generalized Interpolation Material Point (GIMP) Method and SAMRAI Parallel Processing. *CMES: Computer Modeling in Engineering and Sciences*, vol.8, no. 2, pp. 135-152.

Oldenburg, M.; Nilsson, L. (1994): The position code algorithm for contact searching. *Int J Numer Methods Eng.*, vol. 37, pp. 359-86.

Ortiz, M.; Pandolfi, A. (1999): Finite deformation irreversible cohesive elements for three dimensionl crack propagation analysis. *Int J Solids Structures*, vol. 33, pp. 2899-2938.

Paik ,S. H.; Moon, J. J., Kim, S. J.; Lee, M. (2006): Parallel performance of large scale impact simulations on linux cluster super computer. *Computers and Structures*, vol. 84, no. 10-11, pp. 732-741.

Repetto, E.A.; Radovitzky, R.; Otriz, M. (2000): Finite element simulation of dynamic fracture and fragmentation of glass rods. *Comput. Mehods Appl. Mech. Eng.*, vol. 183, pp. 3-14.

Steffen, M.; Wallstedt, P. C.; Guilkey, J. E.; Kirby, R. M.; Berzins, M. (2008): Examination and analysis of implementation choices within the material point method (MPM). *CMES: Computer Modeling in Engineering and Sciences*, vol. 31, no. 2, pp. 107-127.

Sulsky, D.; Kaul, A. (2004): Implicit dynamics in the material-point method. *Comput. Mehods Appl. Mech. Eng.*, vol. 193, no. 12-14, pp. 1137-1170.

Yoo, Y.H.;Lee, M. (2003): A three-dimensional FE analysis of large deformations for impact loads using tetrahedral elements. *Comput. Mech.*, vol. 30, pp. 96-105.

Zienkiewicz, O.C.; Codina, R. (1995): A general algorithm for compressible and incompressible flow-Part I. The split characteristic-based scheme. *Int J Numerical Methods Eng.*, vol. 20, pp. 865-885.