〈 논문 〉

Lagrangian 기법에 의한 충돌 해석 시 접촉처리의 병렬화 및 병렬효율 평가

백승훈† · 김승조* · 이민형**

(2006년 5월 24일 접수, 2006년 9월 5일 심사완료)

Parallel Contact Treatment and Parallel Performance of Impact Simulation Based on Lagrangian Scheme

Paik, Seung-Hoon, Kim, Seung-Jo and Lee, Minhyung

Key Words: Parallel Computing(병렬계산), Explicit FE Method(비선형 외연유한요소법), High Speed Impact(고속 충돌), Parallel Performance(병렬성능)

Abstract

The evaluation of parallel performance of a high speed impact simulation is not an easy task because not only the development of parallel explicit code is difficult but also a large number of processors is not easily accessible. In this paper, the parallel performance of a new Lagrangian FEM impact code carried out on cluster supercomputer has been described in high speed range. In the case of metal sphere impacting to oblique plate, the overall speed-up continuously increases even up to 128 CPUs. Investigation of elapsed time of each part reveals that most of the inefficiency comes from the load imbalance of contact.

1. 서 론

여러 해석 분야 중 특히 고속 충돌해석 분야는 시간에 따른 대 변형 운동 및 재료의 비선형 거동, 그리고 복잡한 접촉 문제를 동시에 다루어야 하기 때문에 많은 계산시간을 필요로 한다. 또한, 모델의 정밀화에 대한 필요성이 점차 증가 하고 있기 때문에, 통상적으로 많은 자유도의 유한요소 모델을 동반하게 된다. 따라서 주어진 기간 내에 충돌현상을 신뢰성 있게 재현하기 위해서는, 큰 계산능력이 요구되며, 단일 CPU로는 이러한 요구사항을 충족시킬 수 없고, 여러 컴퓨터의 능력을 동시에 사용하는 병렬계산 기술을 통해 효과적으로 대응할 수 있다. 고속충돌해석의 병렬화 연구를 위해서는 복잡한 충돌과정의 신뢰성 있게 모사할 수 있는 코드를 이미 개발 혹은 확보 하고 있어야 하고 병렬화 코딩작업이 매우 어렵기 때문에 충돌해석 병렬화 연구에 많은 어려움이 따른다. 또한, 대규모 문제의 병렬 효율성 테스트 및 개선을 위해서는 단독으로 활용할 수 있는 대규모 병렬컴퓨터가 확보되어 있어야 하기 때문에 주로 미국 국립연구소 중심으로 연구 결과가 보고되고 있다. 90년대 초반부터 본격적으로 연구가 시작된 충돌해석 병렬화는 최근 수천 CPU 까지도 성능을 보이는 결과를 보고한 바 있다.[1,2] 그러나, CTH 와 같은 오일러리안 방식의 고속 충돌해석 코드의 경우, 병렬 환경에서 계산시간이 보통 수 일이 소요되는 등[3] 실제 적용 문제에서는 아직도 계산시간에 대한 부담은 크기 때문에 병렬효율을 높이기 위한 연구는 지속적으

† 책임저자, 회원, 서울대학교 대학원 기계항공공학부
* 회원, 서울대학교 기계항공공학부, 비행체특화센터
E-mail : sjkim@snu.ac.kr
** 회원, 세종대학교 기계항공우주공학부

로 필요하다.

고속충돌해석과 관련해서는 국내에서도 약간의 연구가 있었으나,[4] 본 논문에서 다룬 고속충돌 병렬화와 관련하여서는 아직 충분한 연구가 이루어지고 있지 않은 실정이다. 충돌해석 병렬처리와 관련된 국내 연구로는 쉘 요소에 대한 충돌 병렬화 연구가 보고되어 있는데,[5] 간단한 충돌 모델에 대한 테스트로, 대규모 병렬계산 자원에서 지속적인 성능 향상을 보이지는 못하였다.

고속 충돌을 수치 모사할 수 있는 상용코드들이 개발되어 있으나, 새로운 수치알고리듬과 재료 모델링해석을 위해 미국 국립연구소를 중심으로 선진 연구기관들은 자체 코드를 개발/보유하고 있다. 본 연구팀도 이러한 목적과 더불어 효율적인 충돌 병렬 알고리듬연구를 위해 IPSAP/Explicit (IPSAP: Internet Parallel Structural Analysis Program) 코드를 자체 개발하고 있다. 코드의 검증을 위해, 여러 참고문헌의 실험 및 다른 해석 코드와의 결과를 비교하여 단일 프로세서 환경에서 개발된 코드의 신뢰성을 확인하였다.[6] 충돌해석 코드의 병렬화는 유한요소부분의 병렬화와 접촉처리 부분의 병렬화로 대별되는데, 유한요소부분의 병렬화 및 병렬효율에 대한 평가는 참고문헌[7]에 자세히 기술하였다. 본 논문에서는 IPSPA/Explicit 에 구현한 접촉 병렬 알고리듬을 기술하고, 고속 충돌 예제에 대해 병렬성능을 평가해 보았다. 접촉병렬 처리는 접촉영역분할을 유한요소 영역분할을 기준으로 접촉영역을 분할하는 방식[8]과, 병렬RCB 방법 등을 이용하여 유한요소영역분할과 별도로 접촉영역을 분할하는 방식[1]이 있다. 기본적으로 유한요소를 기준으로 접촉영역을 분할하는 방식을 따르나, 마스터 세그먼트의 영역분할만 유한요소 영역을 기준으로 분할하고, 마스터세그먼트가 공간에서 차지하는 영역에서 어느 정도 확장된 영역에 들어온 다른 프로세서의 슬레이브노드를 송/수신 하는 방식을 이용하여 접촉병렬화를 구현하였다. 슬레이브 노드는 비구조화 통신방법[1]을 이용하여 송/수신하였다. 이 방법은 마스터와 슬레이브를 따로 지정하는 방식은 물론, 자가접촉(Self Contact)방식과 같이 동일하게 지정하는 방식에 모두 효과적으로 적용할 수 있다.

한편, 미국 국립연수소 등을 중심으로 보고된 병렬성능관련 논문은 주로, 기존에 컴퓨터 제조업체에서 출시되는 모델에서 성능테스트를 하였거나[3,8] 아니면 특수 설계된 OS를 기반으로 하는 경우가 많았다.[1,2] 그러나, 컴퓨터의 성능 대비 부품의 가격이 점차 저렴해 지면서 대학 및 연구실 단위에서 병렬컴퓨터를 자체 구축하여 운용하는 경우가 많아지고 있고, 그 중 가장 대표적인 형태가 리눅스 클러스터 방식의 병렬 컴퓨터이다. 본 논문에서도 이와 같이 자체 구축한 리눅스 클러스터 슈퍼컴퓨터에서 개발된 병렬코드의 병렬성능을 테스트 하였다.

2. 병렬 충돌해석 계산 과정

Table 1에 병렬충돌해석 계산 흐름도를 나타내었다. Table에서 밑줄 친 부분이 병렬화가 되면서 추가로 처리해야 할 부분이다.

Step 1은 Input read 및 초기화 하는 과정이다. 개발된 코드에서는 계산이 시작되면 모든 프로세서가 동일한 입력파일을 읽도록 되어 있으므로, 동일한 경로에 입력파일과 필요한 경우 영역분할에 대한 정보가 있는 파일이 있어야 한다. 실행파일도 동일한 경로에 있어야 한다. 물론, 실행파일의 위치와 입력파일의 위치는 달라도 상관없다. 따라서 NFS(Network File System)으로 묶여져 있으면 상관없지만, 그렇지 않은 경우는, 각 계산노드별로 위의 파일들을 복사해 놓아야 한다.

Step 2의 질량벡터는 처음에 한번 결정되면, 계산 끝까지 변하지 않기 때문에 처음에만 통신을 해주면 된다. 경계면의 절점의 질량을 서로 주고/받은 후 내 영역에서 계산된 질량벡터에 더하면 된다.

Step 3의 유한요소 영역분할은 주로 METIS, Charco와 같은 그래프분할 알고리듬 기법을 사용하게 되며, 다른 유사 코드에서도 유한요소의 병렬화를 위한 부하 균등 시 이러한 그래프 분할 기법을 사용하고 있다.[1,2,8] 본 연구에서도 METIS library 를 사용하여 영역을 분할하였다.

Step 5와 Step 12은 동일한 과정으로 유한요소 계산의 병렬화과정으로 접촉 병렬화 과정과 함께 병렬충돌해석 과정의 두 축을 이루고 있는 부분이다. 유한요소 계산은 각 요소에 대해 절점의 속도 및 변위를 가지고, 응력을 구한 후, 응력값을 이용하여, 절점에 작용하는 내력을 벡터 형태로 구하는 과정이다. 각 요소에 대해 계산을 독

립적으로 수행하므로, 절점을 공유하는 주변 유한요소와 데이터를 연성해서 풀 필요가 없고, 따라서 각 프로세서는 주변 영역과 데이터 교환 없이 독립적으로 내력벡터를 계산할 수 있다. 병렬화 관점에서 이러한 점이 외연적 유한 요소법의 큰 장점이라 할 수 있다. 다만 각 프로세서별로 내력벡터를 구하는 과정이 끝나면, 영역간의 경계면에 위치한 절점들에 대해서는 경계면을 공유한 프로세서간의 통신을 통해서 서로 내력벡터 값을 교환 하고, 받은 내력 값은 내 영역의 경계면의 해당 절점에 합하면 된다.

Table 1 Flowchart of parallel impact simulation

<< 초기화 과정 >>
Step 1. 초기화 및 Input Read
Step 2. 질량 벡터 M 계산 및 질량벡터 통신
Step 3. 영역분할(Domain Decomposition)
Step 4. 초기 외력벡터 계산
Step 5, 초기 유한요소 계산
내력벡터 계산, 내력벡터 통신
임계시간스텝 계산, 임계시간스텝 통신
Step 6. 가속도 계산:
a(n)=M-1*(f(n)-Cdmap*v(n-1/2))

<< 시간 증분 과정 >>
Step 7. 시간 증분 : t(n+1)=t(n)+dt(n+1/2)
Step 8. 1차 속도 증분 : v(n+1/2)=v(n)+a(n)*dt
Step 9. 속도 경계조건 부가
Step10. 변위 증분
d(n+1)=d(n)+dt(n+1/2)*v(n+1/2)
Step11. 외력벡터 계산
Step12. 유한요소 계산의 병렬화
내력벡터 계산, 내력벡터 통신
임계시간스텝 계산, 임계시간스텝 통신
Step13. 접촉처리의 병렬화
(1)접촉영역 분할 (2)접촉점 좌표값 송수신
(3)접촉탐색, 접촉력 계산 (4)접촉력 송수신
Step14. 가속도 계산 a(n+1)
Step15. 2차 속도 증분 :
v(n+1)=v(n+1/2)+[t(n+1)-t(n+1/2)]*a(n+1)
Step16. 결과 값 0번 프로세서로 송신, 출력
Step17. Update Counter n ← n+1
Step18. if (t < termination_time) go to 7

유한 요소 내력 벡터 계산 시 매 스텝 마다 체적을 갱신하기 위해 메쉬 사이즈를 계산 하게 되고, 임계시간 스텝이 메쉬사이즈와 물성치의 함수이므로, 임계스텝을 유한요소 내력벡터를 계산하는 과정에서 구하도록 하였다. 각 영역에서 구한 임계스텝 값을 통신하여, 그 중 최소 값을 임계시간 스텝으로 정한 후, 시간 스텝 진전에 사용한다.

결과를 출력할 때에는 각 계산 노드에서 계산된 결과를 0번 프로세서로 모두 모아서 0번 프로세서가 출력파일에 쓰는 방식을 취하였다. 이때, 영역 간 경계면의 절점의 값들을 그 절점이 공유된 개수만큼 데이터를 중복해서 받게 되므로, 모두 받아서 더한 후, 공유된 개수로 나누어 주었다. 본 논문의 주요 내용인 Step 13의 접촉처리의 병렬화 과정은 다음 장에 자세히 기술하였다.

3. 접촉처리의 병렬화

접촉처리 병렬화 과정을 유한 요소 계산의 병렬화 과정과 함께 Fig. 1에 도시하였다. Fig. 1에서 왼쪽이 유한요소 계산 및 내력벡터를 통신하는 과정이고, 오른 쪽이 접촉처리 병렬화 과정이다. 접촉처리의 병렬화는 (1) 접촉영역 분할, (2) 접촉점 좌표값 송/수신, (3) 접촉력 계산, (4) 접촉력 송수신, 이렇게 네 단계로 이루어진다.

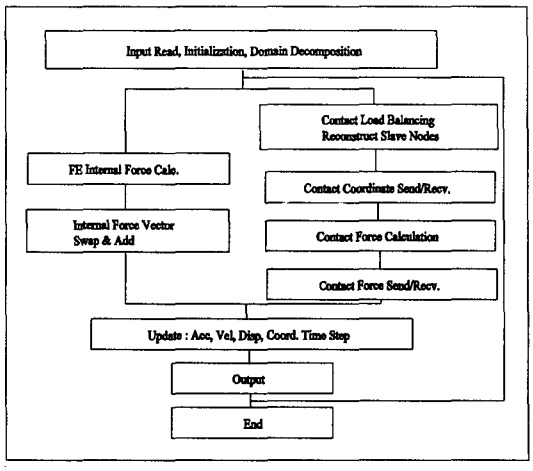


Fig. 1 Flowchart of Parallel FE and Contact computation

3.1 접촉영역 분할

접촉 병렬화 관점에서 가장 중요한 과정은 접촉처리를 위한 부하 균등 과정이다. 부하 균등 과정은 (1) 슬레이브절점을 재구성하는 과정과 (2) 통신을 위한 송/수신 구조를 만드는 것으로 나누었다.

3.1.1 슬레이브 절점 갱신

초기화 과정에서 유한요소 영역 표면에 정의된 마스터 세그먼트를 유한요소 영역분할에 따라 분할한다. 이것은 처음에 한번 수행되고, 계산 종료까지 동일하게 유지된다. 그러나 슬레이브 절점에 대한 정보는 마스터세그먼트가 차지하는 공간상의 위치에 따라 계속 변하므로, 일정 스텝마다 갱신해 주어야 하는데, 다음과 같은 과정을 거쳐 처리하였다.

(1) 새로운 부하 균등을 위해 먼저, 갱신된 접촉점들의 좌표 값을 송/수신한다.
(2) 각 프로세서에 할당된 마스터세그먼트가 공간상에서 차지하는 3차원 박스의 크기 및 확장된 박스의 크기를 구한다. 확장 량은 박스 크기의 약 10 %정도로 하였다.
(3) 확장된 박스영역에 대한 정보(각 방향 최대 최소 좌표 값)를 프로세서 간 통신하여 박스 영역에 대한 정보를 모두 공유한다.
(4) 다른 프로세서의 박스영역 정보를 이용하여, 내 유한요소에 있는 슬레이브 절점 중 다른 프로세서에게 송신해야 할 절점들이 있으면, 그 프로세서 리스트, 각 프로세서 별로 보내주어야 할 슬레이브절점 리스트를 작성한다.
(5) 수신하게 될 슬레이브 절점들의 메모리 할당을 위해 내가 수신하게 될 정보를 알아야 하는데, 내가 송신하는 슬레이브 절점은 누구에게 어느 데이터를 주는지 알고 있지만, 수신 할 데이터는 누구에게 얼마만큼 크기의 데이터가 오는지 알 수가 없다. 따라서 통신을 수행하기 전에 받게 될 정보와 절점 번호를 미리 알게 해 줄 필요가 있는데, 이 과정은 비구조화 (unstructured) 통신 알고리듬[1]을 이용하여 누가 얼마만큼의 데이터를 주는지 알도록 하였다.
(6) 비구조화 통신을 통해 알게 된 정보를 이용해 수신 데이터를 위한 buffer를 구성한다.
(7) 통신해야 할 슬레이브 절점을 송/수신한다.
(8) 수신한 슬레이브 절점과 원래 내 유한요소에 있던 슬레이브 절점을 합하여, 내가 처리해야 할 새로운 슬레이브 절점 리스트를 구성한다.

이렇게 하여, 각 프로세서는 계산에 필요한 마스터 세그먼트 와 슬레이브 절점을 구성한다. 이렇게 구성된 정보는 접촉력 계산 과정에 넘겨지게 된다.

3.1.2 통신을 위한 송/수신 구조 갱신

접촉 부하균등 과정을 통해 구성되는 정보는 다음과 같다.

(1) 데이터를 보낼 프로세서 리스트 exDomains[i] 및 그 개수 NexDomains
(2) 데이터를 받을 프로세서 리스트 imDomains[i] 및 그 개수 NimDomains
(3) 데이터를 보낼 절점리스트 exNodes[i][j] 및 그 개수 NexNodes[i],
개수 중 최대 값 maxNexNodes
(4) 받을 데이터의 절점리스트 imNodes[i][j] 및 그 개수 NimNodes[i],
개수 중 최대 값, maxNimNodes

위의 (1),(2) 항에서 exDomains[i]와 imDomains[i]의 크기는 각각 NexDomains, NimDomains 가 된다. (3)과 (4)항의 exNodes[i][j]와 imNodes[i][j]는 Fig. 2 와 같은 구조를 갖는다. [i]의 크기는 Np로 총 프로세서 개수이다. 만약 [i]번째 프로세서에는 송신할 데이터가 없다면 NULL 을 가리키도록 하였다. 송신용 배열이라면, A[i]가 가리키는 배열이 [i]번째 프로세서에 데이터(좌표값, 접촉력)를 보낼 절점리스트가 된다.

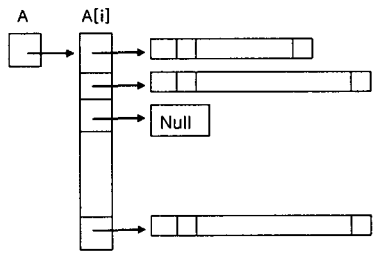


Fig. 2 Structure of contact data send/receive array

위의 항 중 (1)-(4) 항의 10개 변수 및 배열은 접촉좌표 송/수신용으로도 만들어야 하고, 접촉력 데이터 송/수신용으로도 만들어야 하므로 총 2개의 세트가 필요하다. 즉, 20개의 변수 및 배열이 생성되게 된다. 좌표값을 송신했으면, 접촉력을 받아와야 한다는 의미이므로, i 프로세서와 j 프로세서가 서로 접촉데이터를 통신 할 때, i 프로세서에서의 Fig. 2와 같은 좌표값 송신용 배열의 내용 및 절점리스트는 j 프로세서에서의 역시 Fig. 2와 같은 접촉력 송신용 배열 및 절점리스트 내용과 동일해야 한다. 절점번호는 global ID 를 사용하여 통신에 사용하였다. 이러한 절점리스트를 알고 있으면, 데이터 송수신 시 내력벡터에서 구성했던 buffer와 같은 형태의 buffer를 이용하여 접촉점 좌표와 접촉력을 통신한다. 이때에도 역시 받을 때는 어느 프로세서에서 먼저 올지 모르므로 Fig. 2 의 A[i]가 가리키는 배열의 크기를 받을 것들 중 최대 값으로 해야 하는데 이때 이 값이 (3), (4)의 maxNexNode, maxNimNodes 이다.

이 과정까지가 접촉 부하 균등 과정이며, 마스터 세그먼트와 슬레이브 절점의 공간상에서의 위치는 계속 변하므로, 이러한 부하 균등 과정, 즉 위의 정보는 매 스텝 혹은 일정스텝마다 다시 갱신해야 한다. 본 연구에서는 매 스텝마다 갱신하도록 하였다.

3.2 접촉점 좌표값 송/수신

접촉력 계산을 위해, 절점리스트 뿐만 아니라, 그 절점의 좌표 값도 송/수신해야 한다. 이 과정이 끝나면, 접촉처리 과정을 수행하게 되는데, 순차 계산 (serial computing)에서 했던 것과 동일한 과정으로 계산하면 된다. 접촉력 처리과정은 병렬과 독립적으로 계산되므로, 접촉 부하 균등과 접촉력 계산효율 향상은 서로 독립적으로 개산할 수 있다.

3.3 접촉력 계산

접촉력 계산은 마스터세그먼트와 슬레이브 절점만 있으면 계산되므로, 각 프로세서 별로 병렬과 관계없이 독립적으로 계산될 수 있다. 실제로 병렬Code에서 호출하는 접촉력 계산 서브루틴과 MPI library가 없는 단일 프로세서용 Code에서 호출하는 접촉력 계산 서브루틴은 동일하다. 접촉력 계산이 병렬에 영향을 안 받는다는 것은 접촉 알고리듬 개발 및 테스트와 병렬 성능 개선작업을 완전히 독립적으로 수행할 수 있다는 장점이 있다.

3.4 접촉력 송수신

각 프로세서에서 접촉력 벡터가 구해지면, 다시 접촉력을 송/수신해야 한다. 마스터 세그먼트의 접촉력은 경계면이 공유된 프로세서와 경계면에서의 접촉력을 송/수신하면 되고, 슬레이브 절점의 접촉력은 슬레이브 절점의 좌표값을 송수신했던 것과 반대로 송/수신하면 된다. 수신한 접촉력은 내 영역의 접촉력 벡터에 합한다. 이렇게 송수신한 내력벡터와 접촉력 벡터를 이용하여 내유한요소 영역에서 가속, 속도, 변위를 계산한다. 자가접촉(Self Contact)의 경우, 접촉 절점이 어느 쪽 방향으로 접촉되는지를 알려주는 정보가 있는 배열 이 필요한데, 이러한 배열도 같이 통신 해주어야 한다. 이 경우, 각 영역에서 계산된 값을 전체 계산 노드끼리 'SUM' 연산 을 이용해서 통신하여 더해 줌으로써 모든 계산 노드가 정보를 서로 공유하도록 하였다. Fig. 3에 이상에서 설명한 접촉병렬화의 각 단계를 개념적으로 도시하였다.

4. 병렬 성능 평가용 컴퓨터 시스템

병렬 알고리듬 성능 평가에서 어려운 점 중의 하나는 바로 대규모 문제에 대해 병렬 성능을 테스트 할 수 있는 대형 병렬 컴퓨터 시스템의 확보이다. 병렬 Speed-Up 을 측정하기 위해서는 다른 사용자의 방해 없이 수 백 개 이상의 프로세서를 단독으로 활용할 수 있는 환경이 되어야 한다. 본 연구실에서는 병렬 알고리듬 개발 및 테스트를 목적으로 PEGASUS 시스템을 개발하고 활용하고 있다.

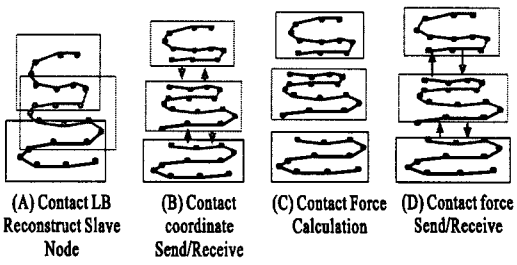


Fig. 3 Process of parallel contact treatment

PEGASUS 시스템은 260개의계산 노드로 구성되어 있으며, 각 노드마다 2개의 Intel Xeon 프로세서를 장착하고 있다. 프로세서는 2.2 Ghz 256개, 2.4 Ghz 112개, 2.8 Ghz 72개, 3.0 Ghz 80개 이며, 본 논문에서는 2.2Ghz에서 성능테스트를 수행하였다.

각 계산 노드는 3~6 GBytes의 메모리와 80~200 GBytes의 하드디스크를 가지고 있으며, Gigabit 네트워크로 연결되어 있고 시간지연(latency)는 약 20 μs이다.

한편, 많은 병렬 코드들이 MPI (Message Passing Interface) Library 를 사용하여 데이터를 송/수신하고 있다. 대표적으로 많이 사용되는 LAM/MPI와 MPICH의 통신성능을 비교한 결과, LAM/MPI의 성능이 좀 더 우수한 것으로 나타났기 때문에, LAM/MPI를 통신을 위한 라이브러리로 사용하고 있으며, 본 연구에서도 이를 사용하였다.

5. 금속구의 경사 충돌

금속구가 금속판재에 경사 충격되는 경우에 대해, 해석 코드의 정확성을 검증하기 위해, 기존의 실험결과[10]와 비교하였다.

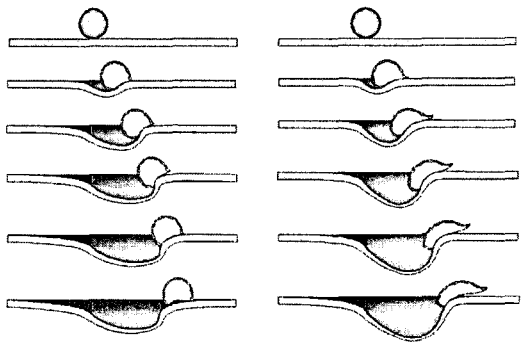
구의 지름은 6.35 mm, 질량은 1.04 g이다. 판재의 길이와 폭은 실험에서는 305 mm, 76 mm이지만, 본 해석에는 50 mm, 40 mm로 하였다. 두께는 1.5 mm 이다.

고속충돌 시 정적 변형에서 고려해야 하는 변형률 경화(strain hardening) 이외에도, 변형률속도에 따른 경화(strain rate dependant hardening) 그리고 단열 변형에 따른 열적 연화(thermal softening) 현상을 동시에 고려하기 위해, 항복조건으로 Johnson-Cook (JC) 모델[11]을 적용하였다.

$$\sigma_Y = (A + B\bar{\varepsilon}^n)\left(1 + C\ln\frac{\dot{\bar{\varepsilon}}}{\dot{\varepsilon}_0}\right)\left[1 - \left(\frac{T - T_{ref}}{T_{melt} - T_{ref}}\right)^m\right] \quad (1)$$

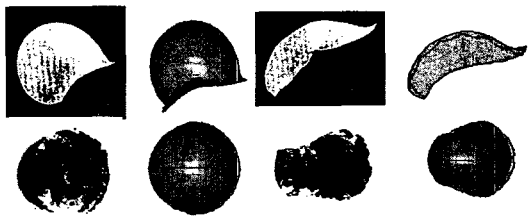
물성치는 문헌[12]에 있는 값을 사용하였는데, 다시 기술하면, 밀도는 7870 kg/m^3, A = 0.53 GPa, B = 0.229 GPa, n = 0.302, C = 0.027, m = 1.0, T_{ref} = 293 K, T_{melt} = 1,836 K이다.

Fig. 4 는 입사각 60도와 충격 속도 610 m/s와 910 m/s 일 때의 변형과정이다.



(a) 610 m/s (b) 910 m/s

Fig. 4 Simulated Impact processes for 60° impact of a mild steel sphere against a mild steel target at 610 m/s (left) and 910 m/s (right)



(a) 610 m/s (b) 910 m/s

Fig. 5 Comparison of simulated and experimented sphere deformation pattern for 60° impact of sphere against a target at 610m/s (left) and 910m/s (right)

Fig. 5에 충격 후 구의 형상을 실험결과와 비교하였다. 구의 변형 형상이 어느 정도 잘 일치하고 있으며, 수치해석을 통해 구의 경사충격에서 나타나는 주요현상들이 잘 나타나는 것을 알 수 있다.

6. 병렬 성능

대규모 문제에서 병렬 효율을 측정하기 위해, 약 150만 자유도의 금속구 경사 충격 모델 (유한요소 개수 521,600 절점 개수 627,865) 에 대해 병렬효율을 테스트 하였다. 610 m/s 속도로 60도 경사 충격되는 조건이다. 현재 개발된 코드에서 요소소진에 대한 병렬화는 아직 구현되지 않은 상태라, 요소소진은 적용하지 않았다. 접촉면은 판재의 윗면을 마스터 세그먼트로 구의 아랫부분을 슬레이브 절점으로 지정하였다. 마스터 세그먼트는 20,000개이고, 슬레이브 절점은 2,125

개이다. 계산은 20 μs까지 하였고, 이 때 까지의 계산 스텝 수는 약 30,000 정도이다. Fig. 6은 METIS를 이용해, 32개로 영역 분할 된 유한요소 모델이다.

Fig. 7에서 유한요소와 접촉처리 계산의 Speed-Up 향상을 보여주고 있다. 여기서, Speed-Up은 단일 프로세서에서의 속도(Elapsed time의 역수) 대비 병렬 계산에서의 속도 비율이다. 128개 CPU 까지도 지속적인 Speed-Up 이 나타나고 있음을 알 수 있다. 유한요소계산 부분은 유한요소 영역 간 경계면에서의 통신시간을 포함하더라도 병렬효율이 좋게 나타나고 있으나, 접촉처리 부분은 CPU 증가에 따라 Speed-Up 이 점점 감소하고 있어, 전체 병렬효율 감소의 주된 원인으로 나타났다.

각 계산 부분이 전체 계산시간에서 차지하는 비중이 클수록 전체 Speed Up 에 미치는 영향이 크기 때문에, Table 2 에 CPU 증가에 따른 접촉처리부분을 포함한 각 부분별 계산시간(Elapsed time)을 전체 계산시간에 대한 백분율로 표시하였다. 접촉처리의 병렬계산은 접촉처리를 위한 부하균등과정(Contact Load Balancing), 접촉처리, 접촉데이터의 통신으로 구성되는데, 이 중, 부하균등 및 접촉데이터의 통신은 사용 CPU 개수와 관계없이 전체 계산시간에서 차지하는 비중이 미약하여 전체 Speed-Up 에 미치는 영향이 거의 없을 것으로 판단된다. 그러나, 접촉력 계산 부분은 사용 CPU가 적을 때에는 10 % 미만이었던 것이 128 CPU 인 경우에는 36 %에 이르고 있어, CPU 가 증가할수록 전체 Speed-Up 에 미치는 영향이 점차 커지는 것으로 나타났다. 이렇게 나오는 이유는 본 예제가 접촉영역이 전 범위에 걸쳐있지 않고, 충격부위에서 국부적으로 발생하기 때문에, Fig. 6의 영역분할에서 보듯이 접촉계산을 하지 않는 프로세서와 접촉계산을 많이 하는 프로세서들 간의 부하 차이가 크게 나기 때문일 것으로 보인다.

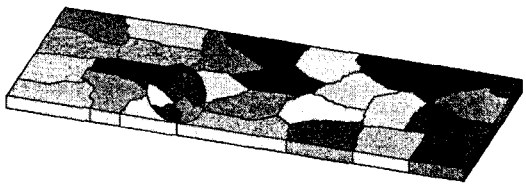


Fig. 6 Decomposed domains of metal sphere impact model using METIS

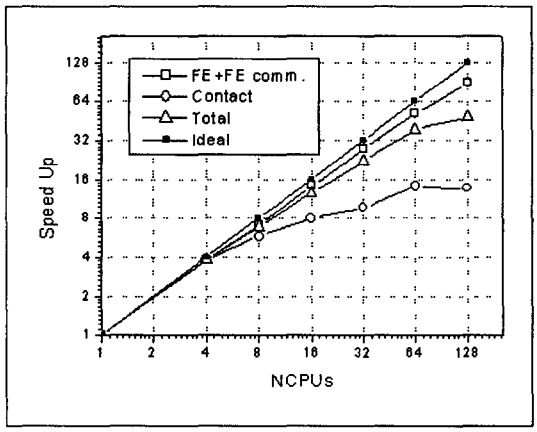


Fig. 7 Speed-Up of each sub function of metal sphere impact model

Table 2 Percent of elapsed time of each sub function of metal sphere impact model

NCPU	FE Processing		Contact Treatment				Etc	Total Elapsed Time (seconds)
	Internal Force (%)	Data Comm. (%)	Load Balancing (%)	Contact Force (%)	Data Comm. (%)	Total Contact Time(%)	Time Integ. + Etc (%)	
1	77.7	1.4	0.0	8.7	0.0	8.8	12.2	8.35E+04
4	77.7	1.4	0.0	8.7	0.0	8.8	12.2	2.20E+04
8	74.7	2.4	0.0	10.4	0.0	10.4	12.4	1.21E+04
16	70.5	2.5	0.1	14.3	0.0	14.4	12.7	6.37E+03
32	64.1	2.7	0.1	20.8	0.1	21.0	12.2	3.62E+03
64	59.0	3.7	0.4	24.8	0.1	25.3	12.0	2.04E+03
128	46.4	5.5	1.1	36.0	0.1	37.2	10.9	1.44E+03

7. 결 론

Lagrangian 기법을 적용한 외연 유한요소 코드의 접촉 병렬화를 위한 알고리듬을 제안하였다. 접촉영역 분할 방식은, 마스터 세그먼트를 유한요소와 함께 분할하되, 마스터 세그먼트가 공간상 차지하는 확장영역에 들어오는 슬레이브 노드에 대한 정보를 비구조화 통신을 이용해 송/수신하는 방식을 적용하였다. 이 방법을 사용했을 때, 약 50만 유한요소의 금속구 충돌예제의 경우, 128 CPU 까지 지속적인 속도 증가를 보이기는 하였으나, 사용 CPU 가 증가 할수록 접촉처리의 병렬효율 감소하는 경향을 보였다. 병렬효율 감소 원인을 분석하기 위해 각 서브루틴 별로 계산시간을 분석해 본 결과 병렬효율 감소의 주된 원인은 접촉력 계산 과정의 병렬 효율 감소가 주된 원인으로 나타났다. 접촉처리 부하 균등화 과정을 개선하면 더 좋은 병렬효율을 확보할 수 있을 것으로 보이며, 이를 위해 접촉면의 유한요소와 분리하여 처리하는 방식에 대한 연구가 필요하다. 한편, 접촉 영역의 크기 및 복잡성에 따라 병렬효율도 다르게 나타날 것이므로, 향후 제안된 알고리듬의 객관적인 성능 평가를 위해 다양한 접촉문제에 적용하고 비교, 검토하여 알고리듬의 효율성을 검토해 보고자 한다.

후 기

이 연구는 ADD 장기 기초 과제(UD040012AD)의 지원을 받아 수행되었습니다.

참고문헌

(1) 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.

(2) Kevin Brown, Steve Attaway, Steve Plimpton, Bruce Hendrickson, 2000, "Parallel Strategies for Crash and Impact Simulations," *Comput. Methods Appl. Engng* Vol. 184, pp. 375~390

(3) Gee D. J., 2003, "Plate Perforation by Eroding Rod Projectile," *Int. J. Impact Engng.* Vol. 28, No. 4, pp. 377~390.

(4) Yoo, Y. H, 2002, "Numerical Simulation of High-Velocity Oblique Impacts of Yawed Long Rod Projectile Against Thin-Plate," *Trans. of the KSME(A)*, Vol. 27, No. 7, pp. 1426~1437.

(6) Har, J, 2003, "A Parallel Finite Element Procedure for Contact-Impact Problems," *Proc. of KSME conference*, pp.1286~1290.

(7) Paik, S. H., Kim, S. J. and Lee, M., 2006, "Parallel Computing of Large Scale FE Model Based on Explicit Lagrangian FEM," *Journal of the Korean Society for Aeronautical and Space Science,* Vol. 43, No. 8, pp. 33~40.

(8) Paik, S. H., Kim, S. J., 2005, "High Speed Impact and Penetration Analysis Using Explicit Finite Element Method," *Journal of the Korea Institute of Military Science and Technology,* Vol. 8, No. 4, pp. 5~13

(9) Malone, J. G., Johnson, N. L., 1994, "A Parallel Finite Element Contact/Impact Algorithm for Non-Linear Explicit Transient Analysis : Part Ⅱ Parallel Implementation," *Int. J. Num. Meth. Eng.*, Vol. 37, pp. 591~603.

(10) Finnegan, S. A., Dimaranan, L. G., Heimdahl OER, Pringle, J. K., 1993, "A Study of Obliquity Effects on Perforation and Ricochet Processes in thin Plate Impacted by Compact Fragments," *Proc. of 14th Int. Symp. on Ballistics*, Quebec, Canada, pp. 661~670.

(11) Johnson, G. R. and Cook, W. H., 1985, "Fracture Characteristics of Three Metals Subjected to Various Strains, Strain Rates, Temperatures and Pressures," *Engng Fracture Mech.* Vol. 21, pp. 31~48.

(12) Yoo, Y. H. and Lee, M., 2003, "A Three-Dimensional FE Analysis of Large Deformations in Contact Impacts Using Tetrahedral Elements," *Computational Mechanics*, Vol.30, No. 2, pp. 96~105.