<학술논문> DOI http://dx.doi.org/10.3795/KSME-A.2014.38.1.017 ISSN 1226-4873(Print)
2288-5226(Online)

그물망과 대량입자의 멀티 스케일 접촉해석

전 철 웅* · 손 정 현**†
* 부경대학교 메카트로닉스공학과, ** 부경대학교 기계자동차공학과

Multi-Scale Contact Analysis Between Net and Numerous Particles

Chul Woong Jun* and Jeong Hyun Sohn**†
* Graduate School of Mechatronics Engineering, Pukyong Nat'l Univ.,
** Dept. of Mechanical and Automotive Engineering, Pukyong Nat'l Univ

(Received May 8, 2013 ; Revised November 12, 2013 ; Accepted November 15, 2013)

Key Words: Graphic Processing Unit(그래픽 처리 장치), Parallel Programming(병렬 프로그래밍), Multi-body Dynamics(다물체동역학), Discrete-Element-Method(이산 요소법)

초록: 그래픽 처리장치(GPU)는 병렬적인 정보를 포함하는 문제를 해결하는데 이상적이다. 본 연구에서는 GPU 는 입자동역학과 함께 다물체 동역학 시뮬레이션을 효율적으로 수행하기 위해 사용되었다. 수치계산을 위해서 HHT 암시적 적분 알고리즘이 사용되었다. 입자들 사이의 접촉을 판별하기 위해서 공간분할 알고리즘과 입자 거동 해석법으로 이산 요소법(DEM)이 사용되었다. 개발된 다물체 동역학 프로그램은 해는 ADAMS 프로그램의 결과와 비교 검증하였다. CPU 기반의 순차해석 프로그램과 GPU 기반 병렬 프로그램은 입자의 수에 따른 수치계산 효율성을 알아보기 위해 서로 비교되었으며, 입자의 수가 많아질수록 계산시간은 단축되었다. 본 예제에서 입자의 수가 1,300 개일 때, 순차 해석 프로그램보다 병렬 프로그램이 약 5 배 가량 빠른 계산 속도를 보였다.

Abstract: Graphics processing units (GPUs) are ideal for solving problems involving parallel data computations. In this study, the GPU is used for effectively carrying out a multi-body dynamic simulation with particle dynamics. The Hilber-Hushes-Taylor (HHT) implicit integration algorithm is used to solve the integral equations. For detecting collisions among particles, the spatial subdivision algorithm and discrete-element methods (DEM) are employed. The developed program is verified by comparing its results with those of ADAMS. The numerical efficiencies of the serial program using the CPU and the parallel program using the GPU are compared in terms of the number of particles, and it is observed that when the number of particles is greater, more computing time is saved by using the GPU. In the present example, when the number of particles is 1,300, the computational speed of the parallel analysis program is about 5 times faster than that of the serial analysis program.

1. 서 론

그래픽 처리 장치(GPU)은 컴퓨터 그래픽을 전문적으로 처리하는 산술 연산 장치이다. GPU 는 3D 그래픽 연산을 처리하여 CPU 의 부담을 줄여줌으로써 빠른 정보 처리를 가능하게 한다. GPU 는 더 자연스러운 3D 이미지를 표현하기 위해서, 많은 양의 데이터를 빠르게 계산하기 위한 코어의 수를 증가시켜가면서 개발되었다. 최근에 고화질의 3D 그래픽을 표현하고 빠른 계산을 필요로 하는 시장의 수요를 충족시키기 위해 매우 병렬적이고, 다중 스레드(Thread)와, 멀티코어(Multi-core) 프로세서를 기반으로 개발이 이루어지고 있다. 이와 같은 GPU 가 지닌 고유의 특징들을 이용하여 기존의 3D 그래픽 처리뿐만 아니라 범용적인 계산 처리 도구로 이용하려는 GPGPU(General Purpose computing on a GPU)에 관한 관심이 증가하고 있다.

2006 년 11 월에 NVIDIA 는 그래픽 카드를 범용적인 계산이 가능하도록 새로운 병렬 프로그래밍 모델인 CUDA™ [1]을 소개했다. 또한 NVIDIA 는 소비자들에게 CUDA 를 적용한 그래픽 카드를 제

† Corresponding Author, jhsohn@pknu.ac.kr

공하기 위해 지속적으로 향상된 성능의 그래픽 카드를 생산하고 있다. 이 뿐만 아니라, 손쉽게 GPU를 이용한 병렬 프로그래밍을 위해 다양한 컴퓨팅 언어를 바탕으로 컴파일러(Compiler), 디버거(Debugger), 라이브러리(Library)[2] 그리고 프로파일러(Profiler)를 제공하고 있다. 현재 CUDA 병렬 프로그래밍은 연예, 디자인, 의학, 교육, 재정 분야뿐만 아니라 공학 분야에서도 활발한 연구가 이루어지고 있으며, Negrut D. 외 5 명[3]은 거대 다물체 동역학 문제에 GPU 를 사용하여 효율성을 비교하였으며, H.Y. Jung 외 2 명[4]은 평면 공간상에서의 다수의 입자와 다물체 모델과의 접촉 문제에 GPU 를 이용하여 효율성을 입증하였다. 하지만 이 논문에서는 평면에서의 문제를 다루었고 입자의 회전을 고려하지 않았다. 본 연구에서는 3 차원 공간상에서의 접촉 문제를 다루었으며, 접촉 평면의 접선 방향 힘에 의한 회전을 고려하였다.

이산 요소법(DEM)[5]은 수송 시뮬레이션, 천문학, 컴퓨터 네트워크, 군중 시뮬레이션, 바이오 시스템 상호작용에서의 개별적인 입자들의 거동을 확인하기 위해 널리 사용 되었던 방법이다. 최근에는 다물체 동역학과 입자 동역학의 개별적인 해석이 아닌 두 시스템의 상호작용에 대한 연구의 필요성이 지속적으로 증가하고 있다. 하지만 대량 입자들 간의 접촉과 강체와의 접촉을 판별하고 거동을 표현하기 위해서는 많은 반복 연산과정이 필요로 하기 때문에 CPU 를 이용한 해석 시간의 증가로 인해 능률적인 연구가 어렵다. 하지만 대부분의 반복 연산 과정은 각각의 반복이 독립적인 계산을 포함하기 때문에 병렬화 계산의 적용이 용이하다. 이에 GPGPU 를 활용한다면 해석시간을 감소시킬 수 있다.

본 연구에서는 다물체 동역학 시스템과 많은 입자들간의 상호작용 해석에 대해서 GPGPU 를 사용한 병렬 계산 알고리즘의 적용과 그 효율성을 연구하였다. 또한, 거대 질량-스프링-댐퍼 시스템과 많은 입자들의 GPU 기반 충돌 해석을 수행하였다. 접촉을 검출하기 위해 neighboring-cell 알고리즘[6,7]을 사용하였으며, 다물체 동역학의 운동방정식을 Cartesian 좌표계에서 유도하였다.[8,9] 입자동역학과 다물체 동역학을 해석하기 위해 각각 Velocity-Verlet[10,11]과 Implicit-HHT[12,13] 적분법을 각각 적용하였다. GPU 의 계산 효율성을 연구하기 위해, CPU 기반의 순차 프로그램과 CUDA 병렬 프로그램의 계산 시간이 각각 비교하였다. 운동방정식의 해를 구하기 위해서 GPU 에 의해 병렬화된 선형대수 라이브러리 CULA[14]를 사용하였다.

2. 다물체와 입자 시스템의 동역학 해석

2.1 다물체 시스템 해석 절차

HHT(Hilbert-Hughes-Taylor) 방법은 구조 동역학에서 2 차 선형 상미분 방정식(ODE)의 수치적분에 널리 사용되는 방법이다. HHT 방법은 Newmark 방법에 기반을 두고 있으며, 적분 공식은 두 개의 파라미터 β와 γ에 종속되며, 식 (1)과 (2)에 의해 정의된다.

$$\mathrm{q_{n+1}} = \mathrm{q_n} + \mathrm{h\dot{q}_n} + \frac{\mathrm{h^2}}{2}[(1-2\beta)\ddot{\mathrm{q}}_\mathrm{n} + 2\beta\ddot{\mathrm{q}}_{\mathrm{n+1}}] \quad (1)$$

$$\dot{\mathrm{q}}_{\mathrm{n+1}} = \dot{\mathrm{q}}_\mathrm{n} + \mathrm{h}[(1-\gamma)\ddot{\mathrm{q}}_\mathrm{n} + \gamma\ddot{\mathrm{q}}_{\mathrm{n+1}}] \quad (2)$$

HHT 방법은 $\alpha \in [-1/3,\ 0]$에서 충분한 안정성과 차수 속성들을 지니며, 식 (3)에 의해 파라미터 β와 γ가 결정된다.

$$\gamma = 0.5 \cdot (1-2\alpha) \quad \beta = 0.25 \cdot (1-\alpha)^2 \quad (3)$$

$$\Phi(q,t) = [\Phi_1(q,t) \quad \cdots \quad \Phi_m(q,t)]^T = 0 \quad (4)$$

$$\Phi_\mathrm{q}\dot{\mathrm{q}} + \Phi_\mathrm{t} = 0 \quad (5)$$

$$\Phi_q\ddot{q} + \left(\Phi_q\dot{q}\right)_q\dot{q} + 2\Phi_{qt}\dot{q} + \Phi_{tt} = 0 \quad (6)$$

$$\mathrm{M}(q)\ddot{q} + \Phi_q^T(q)\lambda = Q(\dot{q},q,t) \quad (7)$$

식 (7)은 식 (8)과 같이 다시 표현될 수 있다. 식 (8)에서 H 는 구속력을 포함하는 일반화 힘들을 나타낸다. t_{n+1} 에서 식 (8)은 식 (9)와 같이 다시 표현된다. 식 (9)의 오른쪽 항은 테일러 시리즈(Taylor series)를 적용한 식 (10)으로 표현된다. t_m 은 t_n 과 τ_{n+1} 사이의 시간이고, $\dot{H}(t_m)$ 은 H 가 식 (11)과 같이 선형적으로 변한다고 가정함으로써, 근사화 될 수 있다. 식 (11)을 식 (10)에 대입하고, 식 (9)에서 $\mathrm{H}(\tau_{n+1})$ 에 대한 결과로 사용하면 식 (12)과 같이 표현된다. 식 (8)을 사용하여 식 (12)를 정리하면 식 (13)과 같이 된다.

$$\mathrm{M\ddot{q}} = \mathrm{Q} - \Phi_\mathrm{q}^\mathrm{T}\lambda = \mathrm{H} \quad (8)$$

$$(\mathrm{M\ddot{q}})_{\mathrm{n+1}} = \mathrm{H}(\tau_{\mathrm{n+1}}) \quad (9)$$

$$\mathrm{H}(\mathrm{t_m}) = \mathrm{H}(\mathrm{t_n}) + (1+\alpha)\mathrm{h\dot{H}}(\mathrm{t_m}) \quad (10)$$

$$\dot{\mathrm{H}}(\mathrm{t_m}) = \left(\mathrm{H}(\mathrm{t_{n+1}}) - \mathrm{H}(\mathrm{t_n})\right) / \mathrm{h} \quad (11)$$

$$(\mathrm{M}\ddot{\mathrm{q}})_{\mathrm{n+1}} = (1+\alpha)\mathrm{H}(\mathrm{t}_{\mathrm{n+1}}) - \alpha\mathrm{H}(\mathrm{t}_{\mathrm{n}}) \quad (12)$$

$$(\mathrm{M}\dot{\mathrm{q}})_{\mathrm{n+1}} + (1+\alpha)\left(\Phi_{\mathrm{q}}^{\mathrm{T}}\lambda - \mathrm{Q}\right)_{\mathrm{n+1}} - \alpha\left(\Phi_{\mathrm{q}}^{\mathrm{T}}\lambda - \mathrm{Q}\right)_{\mathrm{n}} = 0 \quad (13)$$

$\ddot{q}_{n+1}$과 λ_{n+1}을 구하기 위해, 뉴턴-랩슨 반복법이 사용되어야 한다. 뉴턴-랩슨 반복법은 식 (14)의 시스템 방정식이 구성되고, k 번째 반복에서 계산된다. Δ은 Δ_{k+1}과 Δ_k사이의 변화량을 나타내며, 행렬 $\widehat{M} = M_1 + M_2$과 벡터 e_1과 e_2는 식 (15), (16), (17) 그리고 식 (18)에 의해 각각 정의되며,식 (17)에서 $Q_{eq} = \Phi_q^T\lambda - Q$이다.

$$\begin{bmatrix} \widehat{M} & \Phi_q^T \\ \Phi_q & 0 \end{bmatrix}\begin{bmatrix} \Delta\ddot{q} \\ \Delta\lambda \end{bmatrix}^{(k)} = \begin{bmatrix} -e_1 \\ -e_2 \end{bmatrix}^{(k)} \quad (14)$$

$$M_1 = \frac{1}{1+\alpha}(\mathrm{M}\ddot{q})_{n+1} - h\gamma\frac{\partial Q}{\partial \dot{q}} \quad (15)$$

$$M_2 = \left[\frac{1}{1+\alpha}(M\ddot{q})_q + \left(\Phi_q^T\lambda\right)_q - \frac{\partial Q}{\partial q}\right]\beta h^2 \quad (16)$$

$$e_1 = \frac{1}{1+\alpha}(M\ddot{q})_{n+1} + \left(Q_{eq}\right)_{n+1} - \frac{\alpha}{1+\alpha}\left(Q_{eq}\right)_n \quad (17)$$

$$e_2 = \frac{1}{\beta h^2}\Phi(q,t)$$

2.2 입자 시스템 동역학해석 절차

입자들 사이의 접촉력 계산과 거동을 계산하기 위해 이산요소법(DEM)을 사용하였다. 이산 요소법은 모래와 같은 질량 입자들의 운동을 계산하기 위한 수치법이며, 입자의 거동을 시뮬레이션에 가장 좋은 방법으로 알려져 있다. 원래 암석 역학 문제를 연구하기 위해 창안되었고, 1979 년 이후로 알갱이 형태의 재료 문제에 적용되고 있다. 각 입자에 대한 운동방정식은 다음과 같다.

$$m_i\dot{\boldsymbol{v}}_i = \boldsymbol{f}_i, \qquad \dot{\boldsymbol{r}}_i = \boldsymbol{v}_i \quad (18)$$

$$I_i\dot{\boldsymbol{\omega}}_i = \boldsymbol{T}_i \quad (19)$$

위 식에서 m_i, $\boldsymbol{r}_i$, $\boldsymbol{v}_i, I_i, \boldsymbol{\omega}_i, \boldsymbol{T}_i$는 각각 질량, 위치, 속도, 관성모멘트, 각속도 그리고 토크를 나타낸다. 시간 $\mathrm{t}+\Delta\mathrm{t}$에서의 위치와 속도를 결정하기 위해 Velocity-Verlet 적분법을 사용하였으며 그 식은 다음과 같다.

$$\boldsymbol{r}_i(t+\Delta t) = \boldsymbol{r}_i(t) + \boldsymbol{v}_i(t)\Delta t + \frac{\boldsymbol{f}_i(t)\Delta t^2}{2m_i} \quad (20)$$

$$\boldsymbol{v}_i(t+\Delta t) = \boldsymbol{v}_i(t) + \frac{(\boldsymbol{f}_i(t)+\boldsymbol{f}_i(t+\Delta t))\Delta t}{2m_i} \quad (21)$$

$$\boldsymbol{\omega}_i(t+\Delta t) = \boldsymbol{\omega}_i(t) + \frac{I^{-1}(\boldsymbol{T}_i(t)+\boldsymbol{T}_i(t+\Delta t))\Delta \mathrm{t}}{2} \quad (22)$$

3. 접촉 판별 및 접촉력 계산

3.1 접촉 판별 알고리즘

접촉을 판별하는 부분은 입자 동역학 해석에서 가장 많은 계산시간을 필요로 하는 부분이다. 입자들 사이의 접촉 검출은 두 단계에 의해 이루어진다. 어떤 두 입자들의 충돌 여부를 결정하는 것은 모든 충돌 가능한 입자 쌍을 조사하는 것보다 더 효율적인 과정을 요구한다. 접촉을 판별하는 기법으로 본 연구에서는 neighboring-cell 접촉 판별법을 사용하였다. 이 방법은 공간을 그리드(Grid)로 나누어서 각 공간을 여러 개의 셀(Cell)로 표현한다. 다음으로 각 입자에 대해서 입자가 포함된 셀을 결정하고 자신의 셀과 인접한 셀과의 접촉여부를 비교함으로써 효율적으로 접촉을 판별할 수 있다. 셀의 사이즈는 입자의 크기보다 더 작을 수 있고, 하나의 입자는 여러 개의 셀들을 차지할 수 있다.

3.2 접촉력 계산

접촉력[15]은 법선 방향의 힘과 접선 방향의 힘으로 구성된다. 본 연구에서는 1882 년에 Hertz 에 의해 처음으로 제안된 Hertzian 접촉 모델을 이용하였다. 법선 방향의 힘은 Hertzian 스프링력과 viscous 댐핑력으로 이루어지며 다음과 같이 표현된다.

$$\boldsymbol{f}_{ij}^e = \left(\frac{2}{3}\tilde{E}\sqrt{R_{eff}}h_{ij}^{3/2}\right)\hat{\boldsymbol{r}}_{ij} \quad (23)$$

$$\boldsymbol{f}_{ij}^v = -\left(\gamma_n\sqrt{R_{eff}}h_{ij}(\boldsymbol{v}_i - \boldsymbol{v}_j)\cdot\hat{\boldsymbol{r}}_{ij}\right)\hat{\boldsymbol{r}}_{ij} \quad (24)$$

식 (23)에서 $\tilde{E} = E/(1-v^2)$이고, E 는 영률을 나타내며, v는 푸아송비를 나타낸다. 두 입자 사이

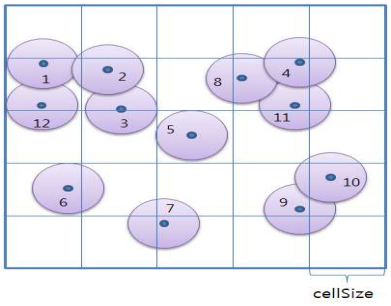

Fig. 1 The arrangement of particle in the grid-cell

의 침투량은 $h_{ij} = R_i + R_j - |\boldsymbol{r}_{ij}|$이고, $\boldsymbol{r}_{ij} = \boldsymbol{r}_i - \boldsymbol{r}_j$ 이다. 그리고 $R_{eff} = R_i R_j/(R_i + R_j)$, $\hat{\boldsymbol{r}}_{ij} = \boldsymbol{r}_{ij}/|\boldsymbol{r}_{ij}|$ 이다. $\hat{\boldsymbol{r}}_{ij}$은 접촉 평면에서 법선 방향의 단위 벡터이다. 식 (24)에서 γ_n은 댐핑 계수이다.

정지 마찰과 미끄럼 마찰은 접선방향의 스프링 력에 의해 표현될 수 있다. 접선 방향의 힘은 다음과 같이 표현된다.

$$\boldsymbol{f}_{ij}^{t} = \min\{k_s\delta_s + v_s(\boldsymbol{v}_{ij})\cdot\hat{\boldsymbol{s}}_{ij},\ \mu_s|\boldsymbol{f}_{ij}^{n}|\}\cdot\hat{\boldsymbol{s}}_{ij} \quad (25)$$

$$\hat{\boldsymbol{s}}_{ij} = \frac{\Delta\boldsymbol{v}_t}{|\Delta\boldsymbol{v}_t|},\ \Delta\boldsymbol{v}_t = (\boldsymbol{v}_{ij} - \boldsymbol{v}_{ij}\cdot\hat{\boldsymbol{n}}_{ij})\hat{\boldsymbol{n}}_{ij} \quad (26)$$

위 식에서, $\boldsymbol{v}_{ij} = \boldsymbol{v}_i - \boldsymbol{v}_j$이고, $\boldsymbol{f}_{ij}^{n} = \boldsymbol{f}_{ij}^{e} + \boldsymbol{f}_{ij}^{v}$이다.

δ_s 은 접선방향의 스프링 변위량을 나타내며, $\delta_s = |\Delta\boldsymbol{v}_t| \times \Delta t$로 계산된다. 입자를 회전시키는 토크는 다음과 같이 계산된다.

$$\boldsymbol{T}_i = (\boldsymbol{r}_i\cdot\hat{\boldsymbol{n}}) \times \boldsymbol{f}_{ij}^{t} \quad (27)$$

4. CUDA 기반 병렬 프로그래밍

다물체동역학 병렬 프로그램은 CUDA C 언어를 사용하여 개발하였다. Device(GPU)의 메모리는 Host(CPU)의 메모리[16] 보다 그 종류가 다양하다. Device 의 메모리의 특성을 제대로 이해하고 사용은 효율적인 병렬 프로그래밍을 위해서 매우 중요하다. 또한 device 에서의 계산을 위해서는 host 의 정보를 device 에의 복사 과정이 필수적이며, 복사 과정은 부하가 큰 작업이기 때문에 이를 최소화 해야 한다. 본 연구에서는 필요한 초기 정보를 host 에서 계산한 후 모든 데이터를 device 로 복사하여 결과값을 출력하는 과정 외에는 host 와 device 간의 메모리 복사 과정을 제거 하여 최대한 효율적인 계산이 되도록 하였다.

Fig. 4 는 다물체 동역학 해석에서의 병렬 프로그래밍의 전체적인 흐름을 보여주고 있다. Fig. 4 에 보여지듯이 초기화 과정이 host 에서 마무리 되

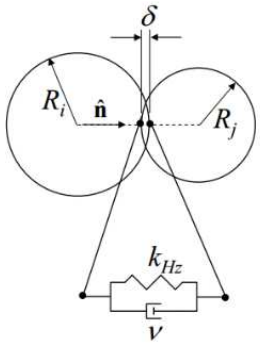

Fig. 2 Model of the normal force between two particles

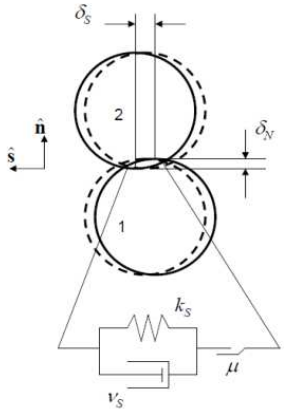

Fig. 3 Model of the tangential force between two particles

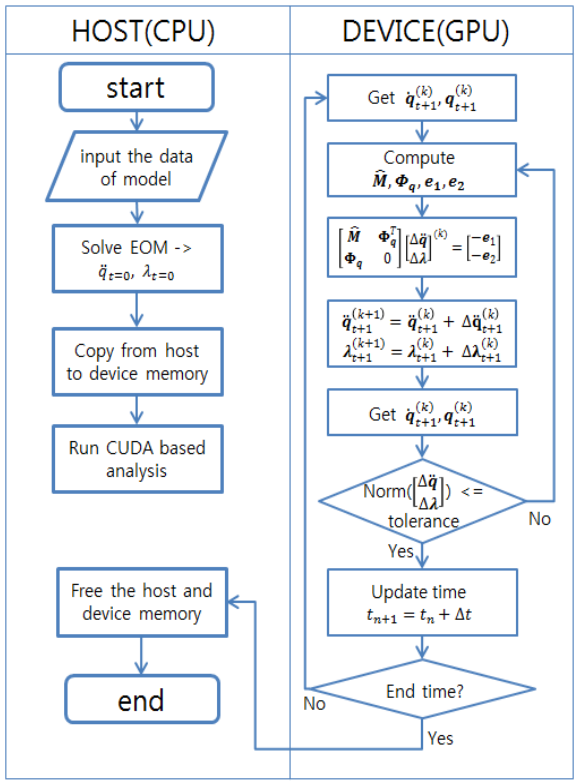

Fig. 4 Flow diagram for parallel programming

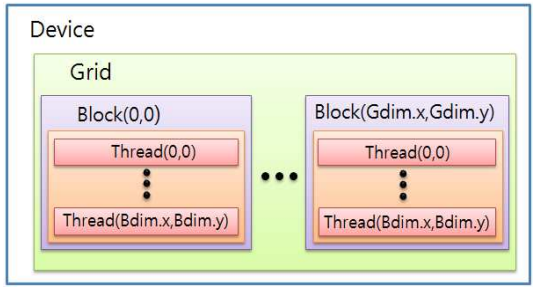

Fig. 5 Generation of grid with blocks and threads

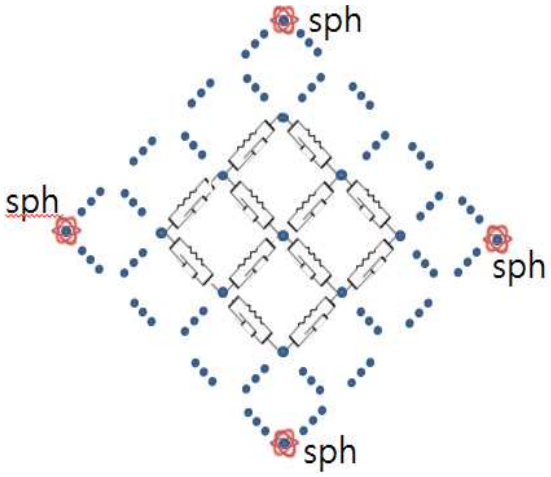

Fig. 6 The net-shape model with the spring-damper

면 device 로 메모리로 데이터의 복사가 이루어지고 전체적인 계산 과정은 device 에서 이루어짐을 알 수 있다. 순차해석의 경우에 시스템을 이루는 구성요소의 수만큼의 반복계산이 이루어지고, 반복 계산은 전 단계의 계산 값이 다음 반복에 영향을 주지 않는 독립적인 성향을 가지기 때문에 병렬화가 가능하다. GPU 에서의 병렬계산은 하나의 커널(kernel)함수에 정의된 계산과정을 할당된 여러 개의 스레드가 각각 커널 함수에 정의된 계산을 수행한다. 커널 함수를 실행하면 device 는 하나의 그리드를 생성하고 명시된 수만큼 블록(Block)과 스레드를 할당한다. Fig. 5 는 device 내의 생성된 그리드-블록-스레드를 구조를 나타내고 있다.

5. 수치시뮬레이션의 효율성 비교

개발된 해석 프로그램의 정확성을 검증하기 위해서 간단한 그물형상 모델을 구성하여 시뮬레이션하였다. 시스템을 구성하는 구성요소(강체, 조인트, 힘 요소 등)의 수에 따른 해석 프로그램의 효율성을 비교하고자, Fig. 6 과 같은 그물 형상의 시스템을 구성하였고 강체 수와 스프링-댐퍼의 수를 증가 시켜가면서 1 초 동안의 해석시간을 비교하였다. 그물모양의 꼭지점 4 군데에 구면조인트(Spherical joint)를 지면과 연결하여 구속하였다. 입자 동역학의 경우 입자의 수를 점차 증가시켜가면서 순차 해석과 병렬 해석과의 1 초 동안의 계산 시간을 서로 비교하였다.

Fig. 7 은 스프링-댐퍼 힘 요소의 수에 따른 힘을 구하는 함수의 계산 시간의 비교 결과를 나타내고 있으며, Fig. 8 은 강체 수에 따른 1 초 동안의 해석 시간을 보여주고 있다. 강체가 280 개, 스프링-댐퍼가 360 개 일 때 병렬 프로그램은 순차 프로

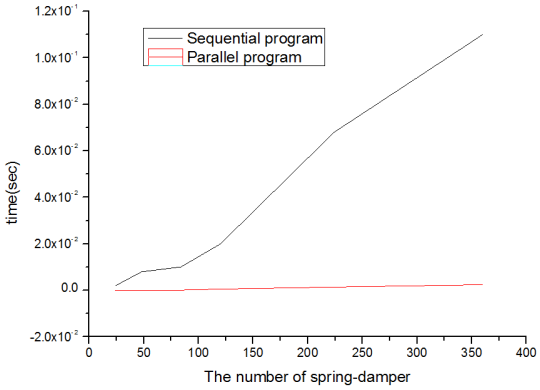

Fig. 7 Comparison of numerical efficiency according to the number of spring-dampers.

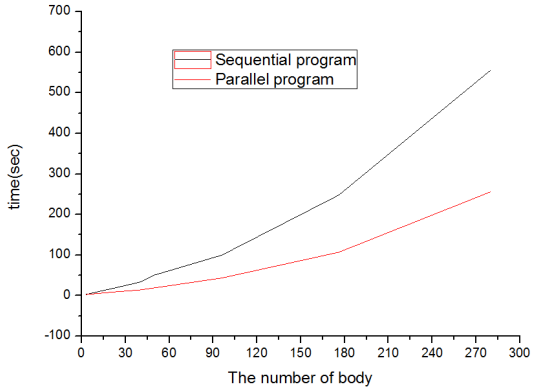

Fig. 8 Comparison of numerical efficiency according to the number of bodies.

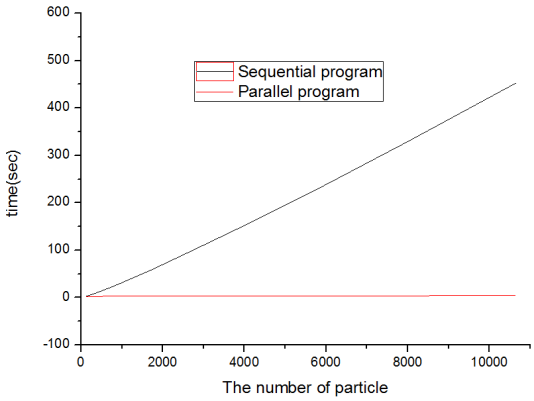

Fig. 9 Comparison of numerical efficiency according to the number of particles.

그램보다 약 2.2 배 가량 높은 효율성을 나타내었다. Fig. 9 은 입자들의 수에 따른 입자들의 접촉 및 거동을 1 초동안의 순차 해석과 병렬 해석의 시간 비교를 나타낸 결과이다.

입자의 수가 증가함에 따라 두 해석 시간의 차이가 상당히 커짐을 확인하였으며, 입자의 수가 10,640 개 일 때, 병렬 프로그램은 순차 프로그램보다 약 100 배의 효율성을 나타내었다. 또한 입자

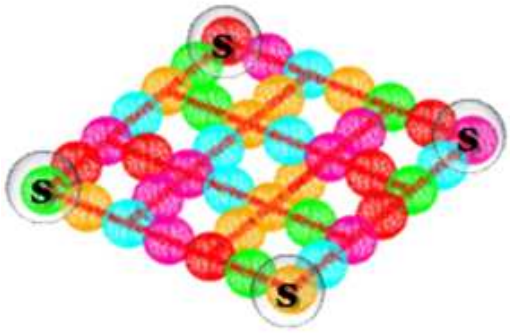

Fig. 10 The net-shape multi-body-model

의 수가 520,000 개 일 때, 순차 프로그램에 비해 병렬 프로그램이 약 360 배의 빠름을 확인하여 입자의 수가 많아질수록 병렬 해석의 효율성이 더 좋아지는 것을 확인할 수 있다.

6. 다물체와 대량입자와의 접촉 해석

다물체 동역학과 입자 동역학은 각기 다른 적분과 계산법을 사용하여 해석 된다. 따라서 두 역학 간의 연성 해석을 위해서는 두 해석 과정이 통합되어야 하는 부분들이 존재한다. 다물체 동역학은 암시적 적분법을 사용함으로 인해 뉴턴-랩슨 반복 계산이 이루어 지기 때문에 뉴턴-랩슨 반복 계산 전에 입자 동역학의 상태들이 미리 정의되어야 한다. 정의된 입자 동역학의 상태들은 뉴턴-랩슨 반복 계산 과정에서 고정된 값으로 사용되어 강체와 접촉 검출을 통해 접촉력이 계산된다.

본 연구에서는 다물체와 입자간의 정보를 공유하고 공유된 데이터를 사용하여 필요한 계산을 수행하기 위해 통합 시스템을 따로 구성하였다. 연성해석에 대한 효율성 비교 대상으로 스프링-댐퍼로 연결된 강체의 수가 433 개, 스프링-댐퍼의 수가 628 로 구성된 그물 형상 모델을 사용하였으며, Fig. 10 은 그물 형상 모델을 표현하고 있다.

Fig. 11-12 는 그물 형상과 입자의 충돌 모습을 나타내고 있으며, 그물 형상 위에서 입자가 떨어지면서 충돌로 인해 그물에 쌓이는 입자와 사방으로 튀는 입자의 모습을 확인하였다. CUDA 병렬 프로그래밍을 사용하여 병렬 프로그램을 구성하였으며, 그물 형상의 모델을 고정하고, 입자의 수를 점점 증가 시켜가면서 1 초 동안의 계산 시간을 순차해석프로그램과 각각 비교하였다.

Fig. 13 에서 입자의 수가 증가함에 따라 계산 시간의 차이가 점점 커지는 것을 확인하였으며,

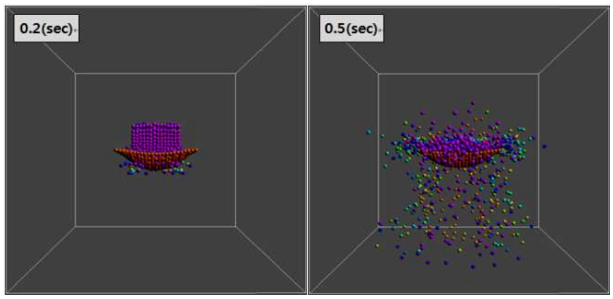

Fig. 11 Animation of the integration simulation at 0.2 sec and 0.5 sec

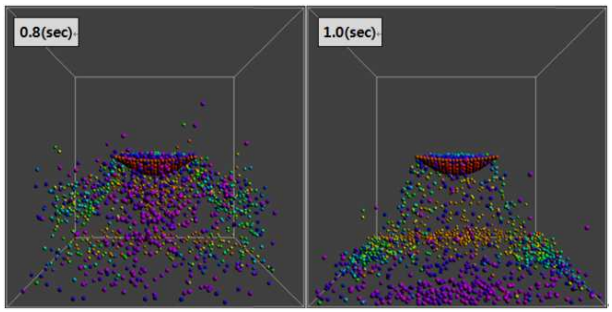

Fig.12 Animation of the integration simulation at 0.8 sec and 1.0 sec

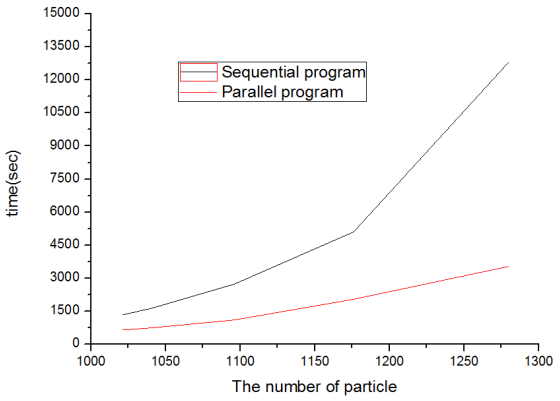

Fig. 13 Comparison of computational times in terms of the number of particles

입자의 수가 1300 개 일 때 병렬 프로그램의 계산 시간은 순차 프로그램보다 약 5 배 가량 빨랐다. 다물체와 입자간의 접촉 해석 시뮬레이션에서도 마찬가지로 요소의 수가 증가할수록 병렬 해석의 효율성이 더 좋아질 것으로 기대된다.

시뮬레이션 결과의 검증을 위해 모래와 같은 알갱이 형태의 요소를 사용한 시간 별 알갱이의 이동경로 및 쌓인 형태를 확인할 수 있는 실험을 통해 입자간의 접촉 계수 값 결정과 함께 입자간의 거동을 검증할 수 있을 것으로 사료된다.

7. 결 론

본 연구에서는, 다물체 시스템과 수많은 입자들

과의 접촉을 시뮬레이션 하였고, GPU 기반의 CUDA 병렬 프로그래밍을 적용하여 시스템의 구성요소 수의 증가에 따른 계산 시간을 비교하였다.

다물체 동역학의 경우, 강체가 280 개, 스프링-댐퍼가 360 개 일 때 병렬 프로그램은 순차 프로그램보다 약 2.2 배 가량 높은 효율성을 나타내었다.

입자의 수가 520,000 개 일 때, 계산시간면에서 병렬 프로그램은 순차 프로그램보다 약 360 배 빠름을 나타내었다.

다물체와 입자 동역학의 연성해석의 경우 강체가 433 개, 스프링-댐퍼 628 개, 입자의 수가 1,300 개 일 때, 병렬 프로그램이 순차 프로그램보다 약 5 배 가량의 높은 효율성을 보였다.

후 기

이 논문은 2011 년도 부경대학교 연구년교수지원사업에 의하여 연구되었음 (CD-2011-0303)

참고문헌

(1) NVIDIA CUDA™, 2011, “NVIDIA CUDA C Programming Guide,” Version 4.1,

(2) NVIDIA CUDATM, 2011, “CUDA Toolkit 4.1 CUBLAS Library,” PG-05326-041_v01.

(3) Negrut, D., Tasora, A., Anitescu, M., Mazhar, H., Heyn T. and Pazouki A., 2010, “Solving Large Multibody Dynamics Problems on the GPU,” *Math, and Comp Science Division*, ANL/MCS-P1777-0710.

(4) Jeong, H. Y., Jun, C. W. and Sohn, J. H., 2013, “GPU-Based Collision Analysis Between a Multi-body System and Numerous Particles,” Vol. 27, No. 4, pp 973~980.

(5) Tupy, M., 2010, “A Study on the Dynamics of Granular Material with a Comparison of DVI and DEM Approaches,” UNIVERSITY OF WISCONSIN-MADISON.

(6) Grand, S. L., 2007, “Broad-Phase Collision detection with CUDA,” Addison Wesley.

(7) Mio, H., Shimosaka, A., Shirakawa, Y. and Hidaka, J., 2005, “Optimum Cell Size for Contact Detection in the Algorithm of the Discrete Element Method,” *Journal of Chemical Engineering of Japan*, Vol. 38, No. 12, pp. 969~975.

(8) Nikravesh, P.E., 1988, “Computer-Aided Analysis of Mechanical Systems,” Prentice-Hall International Inc.

(9) Shabana, A.A., 2011, “Computational Dynamics 2Ed,” Wiley-Interscience.

(10) Schafer, N. and Negrut, D., 2009, “On the Potential of Implicit Integration Methods for Molecular Dynamics Simulation,” *Journal of Computational Physics*.

(11) Schafer, N. and Negrut, D., “On the Potential of Implicit Integration Methods for Molecular Dynamics Simulation,” pp.5~6.

(12) Negrut D., Ottarsson G., Rampalli R. and Sajdak A., 2006, “On an Implementation of the Hilber-Hushes-Taylor Method in the Context of Index 3 Differential-Algebraic Equations of Multibody Dynamics,” DET2005-85096.

(13) MSC Software, “ADAMS help,” MSC Software.

(14) EM Photonics, Inc., 2011, “CULA Reference Manual Release R13(CUDA4.0).

(15) Sarkar, A., 2009, “Discrete Element Method(DEM) Course Module,” PHARMAHUB, Purdue University, Lecture 7-9.

(16) Sanders, J. and Kandrot, E., 2011, “CUDA by Examples : An Introduction to General-Purpose GPU Programming,” Addison Wesley.