GPGPU를 활용한 PDEVS 시뮬레이터 개발 방법론

석문기, 김탁곤

Development Methodology of the PDEVS Simulator using GPGPU

Moongi Seok and Tag Gon Kim

Abstract

GPGPU(General-purpose computing on graphics processing units) is the technique of using a GPU to perform computation in other application. This paper proposes the simulation engine of PDEVS (Parallel DEVS) by using the GPGPU. PDEVS is more appropriate to the multi-core environment than classic DEVS.
Because of the GPGPU characteristic such as SIMD, the simulation engine of PDEVS interoperates with DEVS simulator by the interface model. The result of experiment shows that PDEVS simulator with GPGPU achieves better performance than original DEVS simulator.

Key Words : DEVS, PDEVS, GPGPU, CUDA

1. 서론

게임 소프트웨어 발전과 더불어 그래픽 카드도 발전해왔고 더불어 GPU내 프로세서의 수도 증가하였다. 표 1은 최근의 그래픽 카드의 프로세서의 수를 나타낸다.

표 1. NVIDIA 그래픽카드 프로세서 수[1]

	GTX 280	GTX 260 Core 216	GTX 260	9800 GTX+	9800 GT
Stream Processors	240	216	192	128	112

현재 판매되는 CPU 프로세서의 수가 2~8개임을 고려하면 위의 GPU 프로세서의 수는 엄청나다. 이렇게 많은 GPU의 프로세서를 그래픽 연산에만 사용할 것이 아니라 시뮬레이션에 이용하고자 한다. 이처럼 GPU를 그래픽 이미지 연산이 아닌 그 밖의 목적으로 쓰는 것을 GPGPU(General-Purpose computing on Graphics Processing Units)라 한다.

시뮬레이션을 하기 위해서는 시스템을 기술하기 위한 형식론이 필요하다. 그 중 하나인 DEVS(Discrete Event System Specification) [2] 형식론은 지금까지 활발히 연구되었고 사용되고 있다. DEVS 기반의 시뮬레이션에서는 모델 단위로 스케줄 된 순서대로 모델이 동작한다. DEVS 시뮬레이터는 여러 모델이 동일 시간에 스케줄이 되었을 경우에는 동시에 그 모델들의 동작을 수행하지 않고, 시뮬레이션 시간은 멈춰있는 동안 우선순위 순서대로 모델을 하나 씩 동작시킨다. 이처럼 DEVS 기반의 시뮬레이션에서는 모델이 순차적으로 동작을 하기 때문에 다른 프로세서를 사용할 필요가 없다. 그러므로 멀티 프로세서 환경에서 DEVS 기반 시뮬레이션은 적합하지 않다.

반면, PDEVS(Parallel DEVS) 형식론은 동시에 여러 모델을 수행할 수 있도록 DEVS 형식론을 확장하였다. 이는 이미 병렬, 분산 환경에서 많은 연구가 진행되어 왔다[3].

GPU에서 PDEVS 시뮬레이션이 이루어지게 되면 많은 모델이 같은 시간에 스케줄이 되더라도 모델을 동작시키는 프로세서의 수가 100~260개나 되므로 빠른 시간 내에 동작시킬 수 있다. 이러면 순차적으로 동작하는 DEVS 시뮬레이션에 비해 같은 시간에 스케줄이 되는 모델이 많을수록 시뮬레이션 시간은 크게 단축될 것이다.

하지만 GPU 프로세서는 CPU프로세서와 달리 SIMD (Single Instruction, Multiple Data) 특성을 가진다. 이 특성으로 인해 특정한 조건을 만족하는 모델들만 GPU에서 PDEVS 시뮬레이션이 가능하다.

이와 같은 GPU의 단점을 극복하고자 본 논문에서는 PDEVS 시뮬레이터를 기존의 DEVS 시뮬레이터에 활용한 DEVS-PDEVS 시뮬레이터를 제안한다. 그리고 이 시뮬레이터의 동작 알고리즘과 장점을 설명하고 기존 DEVS 시뮬레이터와 성능을 비교하였다.

본 논문의 구성은 다음과 같다. 2장에서 PDEVS 형식론을 소개하고, 3장에서 GPGPU의 한계를 설명하고, 4장에서는 DEVS-PDEVS 및 동작 알고리즘을 설명한다. 5장에서는 DEVS 시뮬레이터와 DEVS-PDEVS 시뮬레이터의 성능을 비교하는 실험을 하고 6장에서 결론을 맺는다.

2. PDEVS

PDEVS(Parallel DEVS)는 DEVS를 확장한 형식론이다. 대표적인 차이는 DEVS에서는 하나의 입력에 의해 외부 천이 함수가 발생하지만 PDEVS에서는 여러 입력에 의해 외부 천이 함수가 발생했을 때 상태가 어떻게 바뀌어야 할지를 표현할 수 있다.

2.1 PDEVS 원자모델 표현

원자 모델(Atomic Model)은 PDEVS 형식

론을 구성하는 가장 기본적인 모듈로서 시스템의 행동을 기술하는 모델이다. 원자 모델 M의 수학적인 표현은 다음과 같다.

M = <X, Y, S, δext, δint, δcon, λ, ta>
X : 이산사건 입력 집합
Y : 이산사건 출력 집합
S : 일련의 이산사건 상태의 집합
δint : S → S : 내부 상태 천이 함수
δext : Q × X^b → S:외부 상태 천이 함수
X^b는 X의 원소들로 이루어진 bag들의 집합 bag이란 {a,b,a,c}와 같이 원소 중복을 허용하는 유사 집합이다.
Q={(s,e)|s∈S, 0≤e≤ta(s)}
δcon : S × X^b → S : 융합 천이 함수
λ : S → Y : 외부 상태 천이 함수
ta : S → R0,∞+ : 시간 진행 함수

2.2 PDEVS 결합 모델 표현

결합 모델(Coupled Model)은 여러 모델을 내부적으로 연결하여 만든 모델로 DEVS 결합모델에서 SELECT 함수가 제외되었다. 결합 모델 CM의 수학적인 표현은 다음과 같다.

CM = <X, Y, {Mi}, EIC, EOC, IC >
X : 이산사건 입력 집합
Y : 이산사건 출력 집합
{M_i}: 모든 이산사건 콤퍼넌트 모델들의 집합
EIC : 외부 입력 연결 관계
EOC : 외부 출력 연결 관계
IC : 내부 연결 관계

3. PDEVS 시뮬레이터로 GPGPU 응용

GPU는 CPU보다 많은 프로세서를 가지고 있지만 SIMD 특성을 가진다. SIMD(single instruction, Multiple data)란 하나의 명령어로 여러 자료를 처리하는 것을 말한다.

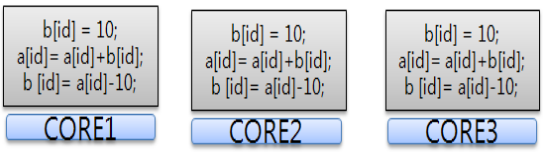

그림 1. 같은 명령어를 참조하고 있는 GPU 프로세서

그림 1과 같이 SIMD 특성 때문에 GPU의 여러 프로세서는 같은 명령어를 수행하고 있다. 실제 GPU에서 서로 다른 모델 M1, M2가 PDEVS 기반으로 시뮬레이션이 되고 있다고 가정하자. 그리고 어느 시점에 모델 M1과 M2가 동시에 다른 내부상태천이함수가 동작해야 한다. 이때 모델 1의 내부상태천이함수 명령어들은 프로세서1에서 처리되고 모델 2의 내부상태천이함수 명령어들은 프로세서2에서 처리된다면 서로 다른 두 모델의 내부상태천이함수는 병렬적으로 동시에 수행이 가능할 것이다. 그렇지만 앞서 밝힌 SIMD 특성에 의해 서로 다른 프로세서가 다른 명령어들을 처리하는 것은 불가능하다. 반면에 두 모델이 같은 명령어를 가지는 내부상태천이함수를 수행해야 한다면 서로 다른 프로세서에서 동시에 수행하는 것이 가능하다.

따라서 GPU를 활용하여 PDEVS 시뮬레이션을 하기 위해서는 모델들의 PDEVS 원자 모델 형식론의 함수의 내부 명령어가 같아야 한다. 예를 들면 cellular automata에서 cell이나 워게임 시뮬레이션에서 동일한 교리를 따르는 군인들, 동일한 운동방정식을 가지는 미사일 등이 있다.

4. DEVS-PDEVS 시뮬레이터

실제 같은 종류의 모델들로만 시뮬레이션할 경우는 흔치 않다. 따라서 PDEVS 시뮬레이터가 독자적으로 동작하는 방식이 아니라 기존 개발된 DEVS 시뮬레이터와 함께 동작하는 DEVS-PDEVS 시뮬레이터를 제안한다.

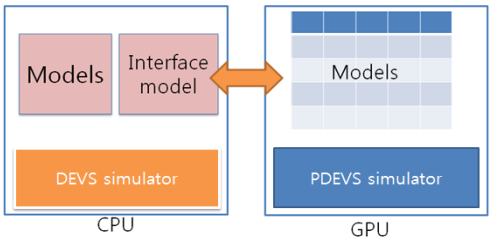

그림 2. DEVS-PDEVS 시뮬레이터

DEVS-PDEVS 시뮬레이터 전체 구성은 그림 2와 같다. CPU에는 DEVS 시뮬레이터가 동작하고 GPU에는 PDEVS 시뮬레이터가 동작한다. GPU 위의 PDEVS 시뮬레이터에서 동작이 가능한 모델은 GPU에 두고, 그 외의 모델은 DEVS 시뮬레이터 위에서 동작시킨다. 그리고 DEVS 시뮬레이터 IF(인터페이스) 모델을 통해서 두 시뮬레이터 간 데이터를 교환한다.

이 방법을 통해 기존의 DEVS 시뮬레이터를 재사용하면서 제약이 있는 GPU의 PDEVS 시뮬레이터를 다른 모델들과 시뮬레이션이 가능하게 하였다.

4.1 IF(Interface) 모델

IF 모델은 PDEVS 시뮬레이터 모델들을 대표하고 DEVS-PDEVS 두 시뮬레이터 사이의 시간 동기화를 한다. 그리고 PDEVS와 DEVS의 형식론 차에 의해서 발생하는 문제를 해결해주어야 한다. 예를 들면 PDEVS 모델에서 DEVS 모델로 향하는 출력이 {y1, y2, y3}라면 IF 모델은 ta=0인 3번의 내부천이함수가 이루어지면서 y1, y2, y3 를 한 번씩 출력한다. 이렇게 PDEVS 의 출력 bag이 시간 진행하지 않으면서 순차적으로 3번 나오기 때문에 DEVS 시뮬레이터에서 모델은 순차적으로 외부상태천이함수 동작이 가능해졌다.

4.2 PDEVS 시뮬레이터 동작 알고리즘

전체 시뮬레이션 시작은 DEVS 시뮬레이터에 의해 이루어지고 DEVS 시뮬레이터에서 종료가 된다. PDEV 시뮬레이터는 IF 모델의 δext, δint, λ, ta가 호출되면 동작한다. 이렇게 PDEVS 시뮬레이터는 DEVS 시뮬레이터에 의존적으로 동작을 하기 때문에 PDEVS 형식론이 다음과 같이 변경하였다.

M=<X, Y, S, δexti, δexto , δint, δcon, λ, ta>

δexti는 GPU 내부 모델로부터 입력을 받았을 때 수행되는 외부상태천이함수이고, δexto는 DEVS 모델로부터 입력을 받았을 때 호출되는 외부상태천이함수를 말한다.

PDEV 시뮬레이터가 IF 모델로부터 호출되어 동작하면 DEVS 시뮬레이터는 PDEVS 시뮬레이터 동작이 끝날 때까지 기다리고, 동작이 끝나면 DEVS 시뮬레이터는 계속 동작한다. PDEVS 시뮬레이터 동작 알고리즘은 다음과 같다.

① IF 모델의 δext가 호출된 경우

(가) PDEVS 시뮬레이터 현재 시간을 IF 모델의 함수가 호출된 DEVS 시뮬레이터 시간에 맞추고, 각 모델의 스케줄까지 남은 시간을 수정한다.
(나) IF 모델의 입력을 확인하고 그 입력이 실제로 PDEVS의 어느 모델로 향하는지 확인한다. 그리고 그 PDEVS 모델의 δexto를 수행한다. 이 때 남은 시간이 0이면 δcon을 호출하여 δexto와 δint중 우선순위를 결정한다. 그리고 우선순위에 맞게 δexto, δint를 수행한다.
(다) 상태가 바뀐 모델의 ta를 호출하고 스케줄까지 남은 시간을 ta 값으로 수정한다.

② IF 모델의 δint가 호출된 경우

(가) PDEVS 시뮬레이터 현재 시간을 IF 모델의 함수가 호출된 DEVS 시뮬레이터 시간에 맞추고, 각 모델의 스케줄까지 남은 시간을 수정한다.
(나) 남은 시간이 0인 모델들에 대해서 PDEVS의 δint를 수행한다.
(다) 상태가 바뀐 모델의 ta를 호출하고 스케줄까지 남은 시간을 ta 값으로 수정한다.

③ IF 모델의 λ,가 호출된 경우

(가) ① 알고리즘 (가)와 λ동일
(나) 스케줄까지 남은 시간이 0인 모델들에 대해서 PDEVS의 λ를 수행한다.
(다) 이 때 발생한 출력 중 DEVS 모델로 가는 출력들을 IF 모델에게 알려준다.
(라) 출력이 PDEVS 모델의 입력으로 들어갈 때에는 해당 모델에 대해 δexti를 수행한다. 이 때 그 모델이 스케줄까지 남은 시간이 0인 경우 δcon을 호출하여 δexti와 δint중 우선순위를 결정한다. 그리고 우선순위에 맞게 δexti, δint를 수행한다.
(마) ① 알고리즘 (다)와 동일

④ IF 모델의 ta가 호출된 경우

(가) ① 알고리즘 (가)와 동일
(나) PDEVS 전체 모델 중 스케줄까지 남은 시간이 가장 작은 것은 값을 IF모델에게 알려준다. 이 값을 GPGPU의 병렬 연산을 이용해서 발견할 수 있다.

5. 실험

cellular automata의 한 예인 산불이 퍼지고 이를 헬리콥터로 진압하는 것을 DEVS-PDEVS 시뮬레이터를 사용했을 때와 DEVS 시뮬레이터만을 사용했을 때 시뮬레이션 종료시간을 비교하였다. 실험 컴퓨터 사양은 아래와 같다.

표 3. 시뮬레이션 컴퓨터 사양

CPU	Intel® Core™2 Quad Processor Q9550
RAM	4GB
GPU	Geforce GTX 260

DEVS-CUDA 시뮬레이션의 경우, 숲 cell들은 PDEVS 시뮬레이터에서 동작하고 헬리콥터 모델은 DEVS 시뮬레이터에서 동작한다. 그리고 DEVS 시뮬레이터의 경우 모든 모델이 DEVS에서 동작한다. DEVS 시뮬레이터는 기존에 개발되었던 DEVSim++[5]을 이용하였고, PDEVS 시뮬레이터는 본 논문에서 제시한 알고리즘을 기반으로 CUDA 환경에서 만들어졌다.

두 시뮬레이션 종료까지의 걸리는 시간은 다음과 같다.

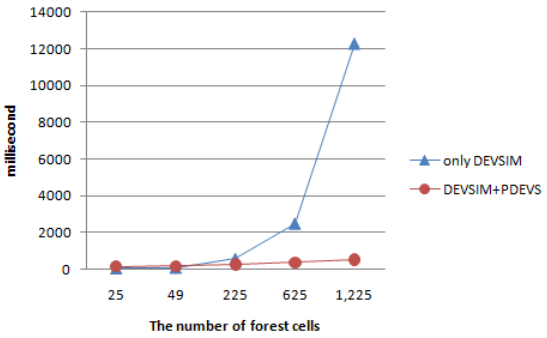

그림 3. 시뮬레이션 종료 시간

결과를 보면 처음 숲 cell의 숫자가 적을 때는 DEVS 시뮬레이터로만 시뮬레이션 했을 때에는 더 좋은 성능이 나왔지만 cell의 숫자가 커질수록 DEVS-PDEVS를 함께 사용한 것이 더 좋음을 볼 수 있다. 이 이유는 cell 수가 적을 때에는 PDEVS를 사용해서 생기는 이득보다 CPU, GPU 간의 통신시간 증가에

의한 손해가 더 크기 때문이다. 그렇지만 cell 수가 증가하면서 동시 다발적으로 불이 발생하는 지점이 늘어나고, 그만큼 PDEV에 의해 동시에 수행되는 모델의 수가 늘면서 DEVS로만 시뮬레이션 한 것보다 성능이 좋음을 확인할 수 있다.

6. 결론

이 논문을 GPGPU를 이용해서 PDEVS 시뮬레이터로 활용하는 방법을 제시했다. 그리고 GPU의 SIMD 특성으로 인해 PDEVS 시뮬레이터 모델의 제약 있지만 기존 DEVS 시뮬레이터에 연동하여 동작하게 함으로써 기존 DEVS 시뮬레이터도 재사용하고 PDEVS 시뮬레이터도 활용성을 높였다.

GPU에서 동작하는 모델의 수가 크지 않으면 잦은 CPU, GPU간 통신오버헤드로 인해 성능이 저하 될 수 있다. 그렇지만 실험에서처럼 동시에 수행되는 작업이 많다면 큰 성능 향상을 얻을 수가 있다.

참고문헌

1. http://www.anandtech.com/show/2619
2. Bernard P. Zeigler, Herbert Praehofer and Tag Gon Kim, *Theory of Modeling and Simulation.* ACADEMIC PRESS, 2001.
3. Jan Himmelspach, Roland Ewald, Stefan Leye and Adelinde M. Uhrmacher. Parallel and distributed simulation of Parallel DEVS models" in *proceedings of the 2007 spring simulation multiconference*, pages 249-256 ,2007
4. NVIDIA CUDA Compute Unified Device Architecture- Programming Guide, 2008.
5.Tag Gon Kim, *DEVSim++* v3.0 Developer''s Manual, 2006, http://smslab.kaist.ac.kr/