분산 시뮬레이션을 위한 DEVS 특성 기반 시뮬레이션 모델 분배 방법

강원석[O†] 김기형[‡]
[O†]대구경북과학기술연구원 지능형임베디드소프트웨어연구팀
[‡]아주대학교 정보통신대학 정보및컴퓨터공학부
[O†]wskang@dgist.ac.kr, [‡]kkim86@gmail.com

Algorithm for Partitioning the Simulation Models Based on DEVS-features for Distributed Simulation Environment

Won-Seok Kang[O†] Ki-Hyung Kim[‡]
[O†]Intelligent Embedded Software research team, DGIST
[‡]Graduates School of Information and Communications, Ajou University

요 약

시뮬레이션 방법론에 있어서 모델기반 시뮬레이션과 프로세스기반 시뮬레이션으로 나눌 수 있는데, 재사용성, 확장성, 시뮬레이터 기술 용이성 등의 장점으로 모델기반 시뮬레이션이 많이 사용되고 있다. 이러한 이유로 근래에는 컴퓨터 시스템, 항공, 자동차 등에서 모델 기반 시뮬레이션 방법이 사용되고 있다. 모델기반 시뮬레이션 방법으로 수학적 이론을 기반으로 모델을 정의하는 DEVS(Discrete Event System Specification) 형식론은 계층적이고 모듈화 된 형태로 이산사건 시스템을 기술한다. 대규모의 복잡한 시뮬레이션 모델을 검증 할 목적으로 분산 시뮬레이션 방법론이 있는데, 이들은 크게 동기적인 방법과 비동기적인 방법이 있다. 동기적 방식보다 빠른 수행을 위해 비동기적 방법은 전체 Time-order 순이 아닌 로컬 Time-order를 가진다. 그러나 비동기적 방식에는 분산된 시뮬레이터들 간의 전체 Time-order를 유지하기 위해 전 처리된 시뮬레이터 결과들을 저장하는데, Time-order 상으로 현재의 시뮬레이션 시간보다 과거의 사건이 왔을 때 그 이벤트를 처리해주어야 되기 때문이다. 이러한 비동기적 분산 시뮬레이션 방법론에서는 전체 Time-order를 유지하기 위해 과거의 Time-order를 가지는 이벤트가 왔을 때 rollback operation을 수행한다. 그러나 rollback operation은 분산 시뮬레이션 방법론에서 성능 장애요소 중 하나이다. 본 논문에서는 rollback operation을 최소할 할 수 있는 DEVS 모델 분배 방법을 제안한다.

1. 서 론

1)일반적으로 이산 사건 시뮬레이션은 시스템의 성능 분석 및 측정에 많이 사용된다. 그러나 대규모 시스템 시뮬레이션에 있어서 시뮬레이션 수행 시간 증가의 문제점이 남아있다. 분산 시뮬레이션은 이들 시뮬레이션 수행 시간을 줄이는 방법들을 제시하고 있다. 또한 분산 시뮬레이션은 대규모 시스템을 다루기 때문에 모델 검증, 모델의 타당성, 모델의 재사용성과 사용자에 대한 투명성 등을 제공하여야한다. Zeigler에 의해 개발된 DEVS(Discrete Event Systems Specification) 형식론(Formalism)은 이산사건을 모듈화 및 계층적으로 기술할 수 있는 정형화된 구조를 가진다[1]. 임베디드 시스템 및 실생활의 논리 시스템 등에서 성능 분석 및 측정을 위해 DEVS 형식론은 많이 사용되는 하나의 수학적 이론이다.

DEVS 형식론은 계층적인 모델을 구성 할 수 있는 특성으로 기본적 구성요소들을 연결 모델 및 더 복잡한 연결 모델 작성을 제공한 다. 연결 모델은 입/출력 포트들을 연결함으로써 구성된다. 계층적 모델 구성은 다음과 같은 특징을 가진다. 첫 번째 기본요소 시스템들은 상위 시스템들과 통합 전에 생성 및 분석을 가능하게 한다. 두 번째 특별한 목적으로 개발된 모델들은 다른 상위 시스템 구성 시 재사용을 가능하게 한다. 세 번째 모듈화 및 계층적 모델들은 실 시스템 모델 반영을 가능하게 한다. 이러한 시스템 이론적 특성 때문에 DEVS 형식론에 기반 한 모델링과 시뮬레이션은 프로그램 언어에 관계없이 구현할 수 있다. 분산 시뮬레이션 방법 또한 이러한 특성에 부합하기에 DEVS 형식론 많이 이용한다.

DEVS 형식론에 기반 한 분산 시뮬레이션은 전통적인 프로세스 기반 분산 시뮬레이션과 다음과 같은 차이점을 가진다[3]. 첫 번째로 DEVS 형식론은 시뮬레이션 모델에 대하여 외부/내부 사건에 대해 구별한다. 두 번째로 계층적인 모델을 구성한다. 이러한 차이점으로 병렬성을 제공하는 DEVS 형식론으로 인해 분산 시뮬레이션은 이를 적용하고 있다[2-7].

DEVS 형식론에 기반 한 분산 시뮬레이션 방법론으로는 크게 동기적인 방법과 비동기적인 방법이 있다. 동기적 방식보다 빠른 수행을 위해 비동기적 방법은 전체 Time-order 순서대로 순차적으로 처리하는 것이 아닌 로컬 Time-order를 유지하면서 각 분산된 시뮬레이터들이 독자적으로 시뮬레이션을 수행한다. 그러나 비동기적 방식에는 분산된 시뮬레이터들 간의 전체 Time-order를

1) 본 연구는 과학기술부 및 대구경북과학기술연구원의 연구개발사업의 일환으로 수행하였음.

유지하기 위해 전 처리된 시뮬레이터 결과들을 저장하는데, Time-order 상으로 현재의 시뮬레이션 시간보다 과거의 사건이 왔을 때 그 이벤트를 처리해주어야 되기 때문이다. 이러한 비동기적 분산 시뮬레이션 방법론에서는 전체 Time-order를 유지하기 위해 과거 사건이 왔을 때 rollback operation을 수행한다[5,7]. 그러나 rollback operation은 비동기적 분산 시뮬레이션 방법론에서 주요 성능 장애요소 중 하나이다.

비동기적 분산 시뮬레이션 시 대규모의 복잡한 시뮬레이션 모델을 여러 대의 컴퓨터 시스템에 효율적으로 분배(partitioning)하는 기법으로 DEVS 형식론 기반 계층적 구조에 근간한 분배 방법이 있다. 그러나 DEVS 형식론의 모델(원소, 연결)들에 대해서 임의의 cost를 기반으로 분배를 수행하고 있기 때문에 실제 분산 시뮬레이션환경으로 모델을 분배 했을 경우 효과적인 방법이 되지 않을 수 있다[8]. 이러한 문제점을 해결하기 위해 실제적인 모델 특성들을 충분히 반영할 수 있는 기법이 필요하다. 이러한 분배 방법은 분산 시뮬레이션 환경 구축 시 핵심 요소 기술 중 하나이다. 분배 방법을 제대로 구축하지 않을 경우 많은 rollback 현상이 발생 할 수 있고 분산 시뮬레이션 장점인 고성능 처리 기능을 크게 저하시킬 수 있다.

본 논문에서는 DEVS 형식론에 기반 한 분산 시뮬레이션 환경 하에서 대규모 시뮬레이션 모델들을 검증할 때 고성능 시뮬레이션 수행을 위해 최소한의 rollback operation이 발생될 수 있게 하는 DEVS 모델 분배(partitioning) 방법을 제안한다.

본 논문에서는 2장에서 분배 방법에 근간이 되는 DEVS 형식론과 이를 기반으로 계층적 분산 시뮬레이션 방법론, 비 계층적 시뮬레이션 방법론에 대해 설명하고 3장에서 본 논문에서 제안하는 DEVS 형식론 특성 기반 분배 방법에 대해서 설명하고 4장에서는 기존 분배 방법과 비교한 효율성에 대해서 설명한다. 마지막 5장에서는 결론 및 향후 계획에 대해 설명한다.

2. DEVS 형식론 기반 계층적 vs 비계층적 시뮬레이션 방법론

DEVS 형식론은 대상 시스템을 모듈화된 여러 작은 모델들로 나누고 그것들을 계층적으로 구성하는데, 이는 DEVS형식론이 대상 시스템의 동작과 구조를 분리하여 기술하기 때문이다. 시스템 동작부분을 기술하는 모델들을 원소 모델이라고 하며 표 1.과 같이 표현한다.

표 1 원소(Atomic) 모델 정의

$$AM =< X, Y, S, \delta_{ext}, \lambda, ta >$$

단, X: 외부 입력사건의 집합
Y: 출력 사건의 집합
S: 순차 상태의 집합
$\delta_{ext} : Q \times X \rightarrow S$: 외부 상태 천이 함수
$Q = \{(s,e) \mid s \in S \text{ and } 0 \le e \le ta(s)\}$
$\delta_{\text{int}} : S \rightarrow S$: 내부 상태천이함수
$\lambda : S \rightarrow Y$: 출력함수
$ta : S \rightarrow R^{+}_{0,\infty}$: 시간 진행함수

표 1.에서 나타난 것과 같이 DEVS 원소 모델은 두 개의 상태전이 함수를 가진다. 그 중에서 내부 상태전이 함수 δ_{int}는 시간이 진행됨에 따라 일어나는 상태의 변화를 나타내기 위한 함수이고, 외부 상태전이 함수 δ_{ext}는 어떤 상태에서 외부로부터 입력(**external input message**)을 받아서 그 상태가 바뀌는 것을 표현하는 함수이다. 그리고 출력 함수는 내부 상태전이가 일어날 때 수행되어서 외부로 출력을 발생하는 함수이다. 이때 발생한 출력은 다른 모델들의 입력이 되므로 한 원소 모델의 내부 상태전이는 다른 원소모델들의 외부 상태전이를 야기하게 된다. 그러나 외부 상태전이에서는 출력이 발생되지 않으므로 다른 모델들에게 영향을 주지 않는다.

다음의 형태는 연결 모델로 구성요소가 되는 몇 개의 모델들을 결합하여 새로운 모델을 만드는 것을 표현한다. 그런데 이 연결 모델은 또한 다른 모델의 구성요소 모델이 될 수 있기 때문에 이것을 이용하여 복잡한 모델을 계층적으로 구성할 수 있게 된다. 연결 모델 CM(Coupled Model)은 표 2.와 같이 정의된다.

표 2 연결(Coupled) 모델 정의

$$CM =< D, \{M_i\}, \{I_i\}, \{Z_{i,j}\}, SELECT >$$

단,
D: 컴포넌트이름 집합
For each i in D
M_i : 컴포넌트 모델
I_i: i의 출력에 연결된 컴포넌트 집합
For each j in I_i
$Z_{i,j} : Y_i \rightarrow X_j$: i to j 출력 변환

DEVS 형식론 기반 비 계층적 시뮬레이션인 DEVSCluster는 분산 버전에서 기존의 계층적 시뮬레이션 방법론 보다 더욱 간단한 구조로 수행되어질 수 있다. 그림 1.은 기본 DDEVS 분산 시뮬레이션 방법론(a)와 DEVSCluster 분산 시뮬레이션 방법론(b)에 대한 시뮬레이션 방법론을 나타낸다.

DEVSCluster는 기존 DDEVS[2]와 비교 했을 때 메시지[3] 교환 방법에 있어서 중요한 차이를 보인다[9-12]. 기존 DDEVS는 시뮬레이션 모델들 간에 각 외부 입/출력 메시지 교환은 전역 메시지 큐에 추가만 하고 이에 대한 명확한 처리 상황은 판단하지 않는다. 반면에 DEVSCluster는 각 시뮬레이션 모델들 간에 직접 함수 호출을 이용하여 처리한다. 이는 시뮬레이션엔진의 안정성을 높이며 메시지에 대한 분산 처리에 있어서 동기화 알고리즘을 더욱 간편하게 수행 할 수 있게 한다. 실질적으로 기본 DDEVS 방법론은 대규모(large-scale) 시뮬레이션 모델에 대해서 비동기적으로 메시지가 도착할 때 메시지 큐에 저장하여 순차적으로 다른 컴퓨터로부터 전

2) DDEVS는 KAIST에서 개발된 계층적 분산 시뮬레이션 방법론이다. 이는 비동기식 메시지 전달 방법을 수행한다[5-7].
3) 메시지(message) : 시뮬레이션 메시지, 이벤트(event)

달 받은 메시지를 처리하는데 이로 인하여 복잡한 롤백(rollback) 및 제거(:annihilation) 메시지 처리 등의 동기화 작업이 많이 발생하게 된다. 이로 인하여 시뮬레이션 수행 시간 및 정확한 시뮬레이션 수행에 있어서 문제점이 발생하게 된다.

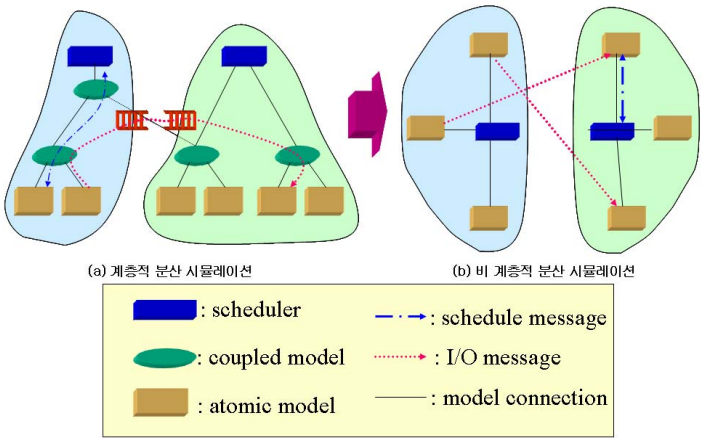

그림 1. 계층적/비 계층적 분산 시뮬레이션 방법론

DEVSCluster는 이러한 메시지 큐를 제거하고 모델들 간에 직접적으로 입/출력 메시지를 함수 호출로 처리한다. DEVSCluster의 분산 시뮬레이션환경은 기존에 설계되지 않았던 멀티스레드 시스템으로 개발되었으며 단일 버전의 DEVSCluster 시뮬레이션 엔진 알고리즘과 비교하여 볼 때 Time Warp 동기화 알고리즘이 원소 모델에 추가되어 기존의 계층적 시뮬레이션 방법론을 간소화하여 시스템 안정성과 수행성능을 높인 시스템이다.

2.1 계층적 기반 DEVS 모델 분배 기법

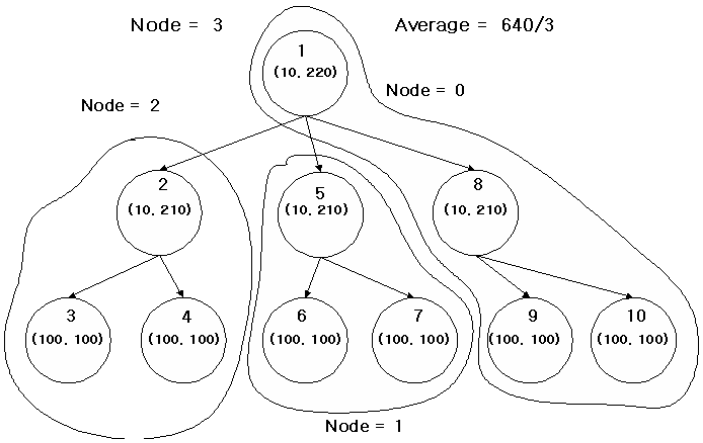

그림 2. 계층적 DEVS 모델 분배 방법

그림 2.는 계층적 DEVS 분산 시뮬레이션 환경 상에서 시뮬레이션 수행 시 시뮬레이션 모델을 효율적으로 분배하기 위해 제안된 DEVS 모델 분배 기법 중 하나이다[11]. 이 기법은 모든 시뮬레이션 노드 즉 시뮬레이터의 load를 가정하여 모델을 분배하는 방법이다. 우선 전체 시뮬레이터 load의 총합을 구하고 분배 할 노드의 개수에 대해서 나눈 값에 가장 가까운 시뮬레이터들을 그룹화 하여 분해하는 방법이다. 이 방법은 부모 노드가 자식 노드의 작업 부하 정보를 모두 더한 부하 정보를 가지고 이를 기반으로 분배 할 컴퓨터 개수(N)를 기반으로 "최상위노드 작업부하/N" 작업 부하 정보를 구하여 계층 구조를 탐색하면서 근접한 시뮬레이터 노드(자식노드들 전체 포함)들을 N로 분배한다. 그러나 이 방법은 전체 각 시뮬레이터들에 대해 **load(processing cost, communication cost)**값을 임의로 설정하여 모델을 분배하기 때문에 실제로 분산 시뮬레이션 환경 상에서 최적의 효과를 주지 못하는 문제점이 있다. 그리고 계층적 구조에 대한 tree depth에 따른 distance 값에 기반으로 하여 시뮬레이션 모델을 분하는 방법도 있다[13]. 이들은 DEVS 형식론의 계층적 구조를 기반 하여 제안된 기술이나 DEVS 모델 세부 속성은 반영하지 못하고 있다. 그리고 비 계층적 시뮬레이션 방법론에서는 실제 시뮬레이션 시 계층적 구조로 수행되지 않기 때문에 적합하지 않을 수 있다.

3. DEVS 특성 기반 시뮬레이션 모델 분배 기법

DEVS형식론에 기반 한 분산 시뮬레이션 방법론은 "계층적 시뮬레이션 방법론"과 "비 계층적 시뮬레이션 방법론"으로 크게 두 가지로 나눌 수 있다.

분배 방법에 있어서 계층적 시뮬레이션 방법론일 때 효율적으로 시뮬레이션 모델을 분배하는 Partitioning 기술이 존재한다. 그러나 2.1절에서 설명한 것과 같이 실제 시뮬레이션 환경 상에서 효율적으로 모델을 분배 할 수 없는 경우가 생기는 문제점이 존재한다. 그리고 계층적 분산 시뮬레이션 방법론을 보다 고성능/안정화를 시키기 위해 제안된 기술 비 계층적 분산 시뮬레이션 방법론에 대한 효율적인 모델 분배 기법에 대한 내용을 아직 나와 있지 않다.

본 장에서는 비 계층적 분산 시뮬레이션 방법론에서 시뮬레이션 모델들이 최대한의 메시지 교환을 가질 수 있게 모델들을 분해하여 그룹핑하는 새로운 분배 방법을 제안한다. 본 논문에서 제안하는 모델 분배의 목적은 비동기적 시뮬레이션 방법에서 볼 때 rollback operation을 일으키는 메시지는 external input message에 의해서 발생하는데, 이러한 external input message 교환수를 최소화 할 수 있는 방법을 개발하는 것이다. 이에 따라 rollback operation이 감소하는 결과를 발생 시킬 수 있다. 그리고 rollback operation이 감소함에 따라 시스템 안정성과 수행속도 또한 더욱 높아 질 것이다.

본 논문에서 제시할 분배 방법 알고리즘을 설명하기 위해 그림 3.과 같은 계층적 DEVS 시뮬레이션 모델 시스템이 존재한다고 가정한다. 각 시뮬레이션 모델의 Computation Cost(:processor occupation)는 본 논문에서는 고려하지 않으며 임의의 상수 값(a)을 부여한다. 그림 4.에서 각 시뮬레이션 모델의 Edge는 시뮬레이션 모델들 간의 시뮬레이션 external input message 교환수를 나타낸다. 메시지 교환은 메시지를 보내는 모델(S)와 메시지를 받는 모델(D)가 있다. 메시지 교환수는 $MSD(S,D)$로 나타낼 수 있다. 본 논문에서는 $MSD(S,D)$와 $MSD(D,S)$의 메시지 교환수는 같은 것으로 간주하여 방향성에 대한 고려는 본 논문에서는 제외한다. 분배 방법을 위한 전체 시뮬레이션 모델 MC개를 가지면 전체

시뮬레이션 모델들에 대한 전체 메시지 교환수($\mathcal{R}$)는 다음과 같이 나타낼 수 있다.

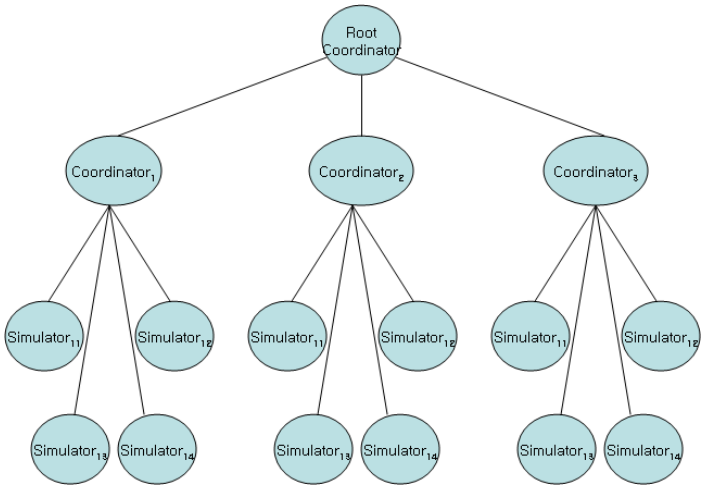

그림 3. DEVS 형식론에 기반 한 계층적 시뮬레이션 모델 구성도

$$\mathcal{R} = \sum_{i=1}^{MC}\sum_{j=1}^{MC} MSD(i,j)$$

본 논문에서 제안하는 알고리즘은 $\mathcal{R}$값을 지정한 분배 개수[4] SC로 나누는데 목적을 둔다. 전체 시뮬레이션 모델[5]을 SC로 나누어 SC 개의 군집화 그룹이 생성되면 이들이 최종 Partitioned Group(PG[6])이다. 본 논문에서는 external input 메시지 교환이 최소로 발생되게 PG를 구하기 위한 방법으로 Heuristic-decision 알고리즘이다. 본 논문의 궁극적인 문제 해결 목적은 $\mathcal{R}$에 대해서 SC개로 최대한 집중된 메시지 교환 시뮬레이션 모델들로 그룹핑된 PG를 구하는 것이다.

본 논문에서 제시하는 DEVS 특성 기반 모델 분배 규칙 및 이에 기반 한 알고리즘은 표 3. ~ 표 5.와 같다. 제안하는 알고리즘은 각 시뮬레이션 모델간의 최소 메시지 교환수를 가지는 시뮬레이션 모델들에 대해 기준 시뮬레이션 모델 그룹(Seed Group)으로 먼저 선택을 한다. 기준 노드가 분배하고자 하는 개수만큼 설정이 되면 나머지 시뮬레이션 모델들에 대해서 각 기준 모델 그룹에 할당한다. 각 시뮬레이션 모델들이 기준 모델 그룹에 분배되는 기본 규칙은 최대의 $MSD(S,D)$ 값을 가지는 시뮬레이션 모델들이다. 기준 시뮬레이션 모델 그룹에 할당을 할 때는 표 3.과 같은 규칙으로 분배가 되어 진다. 규칙에는 Cost 비용이 있는데 본 논문에서 정의하는 Cost 비용은 DEVS 형식론에 정의되어져 있는 *"external input message"* 이며 시뮬레이션 모델들 간에 교환하는 개수를 나타낸다.

4) 최적의 분배 개수는 시뮬레이션 모델 개발자가 사전에 부여하며 거범위는 $SC = \{x \mid x \geq 2 \text{ and } x \in Z\}$ 에 속한다.

5) 전체 시뮬레이션 모델 = $\{ M_i \mid MC \geq i \geq 0 \}$

6) $PG = \{ PG_i \mid sc \geq i \geq 1, \bigcup_{\exists i}^{\exists c} M_i \text{ and } c \in N \}$

표 4.는 DEVS 특성 기반 모델 분배 알고리즘에서 사용하는 데이터 집합 값, 변수 값 들을 정의하며, 입/출력으로 나타나는 결과 값 정의를 나타낸다.

표 3. DEVS 특성 기반 모델 분배 규칙

Rule 1.	어떤 DEVS 모델에 대해 기준 DEVS 모델 그룹과 Cost 비용이 없으면 임의의 기준 DEVS 모델 그룹에 포함함
Rule 2.	어떤 DEVS 모델에 대해 기준 DEVS 모델 그룹과 Cost 비용이 최고 값이면 기준 DEVS 모델 그룹에 포함함
Rule 3.	어떤 DEVS 모델이 특정 DEVS 기준 모델 그룹에 분배되어진 상태에서 이 DEVS 모델이 분배 과정 중 다른 기준 DEVS 모델 그룹에서 더 많은 Cost 비용을 가지면 다른 기준 DEVS 그룹으로 이동
Rule 4.	어떤 DEVS 모델이 둘 이상의 기준 DEVS 모델 그룹과 Cost 비용이 있고 Cost 비용이 각각의 기준 DEVS 모델 그룹과 비교했을 때 그 값이 같을 때 각각의 기준 DEVS 모델 그룹 중 분배된 모델 개수가 최소인 기준 DEVS 모델 그룹으로 이동

표 4. DEVS 특성 기반 모델 분배 알고리즘 I

Data : Set ***M*** of DEVS-based simulation models, Value ***SC*** of the entire node count that will be separated, Set ***MSD*** of N-square matrix array for saving the communication cost between simulation models
Input : Hierarchical structure of DEVS Models
Output : Partitioned Group(*PG*)

표 5.는 앞에서 설명한 Seed Group을 설정하는 알고리즘 부분을 나타낸다. 시뮬레이션 모델의 Seed Group을 정할 때는 무조건 최소 cost가 되는 두 모델들을 Seed Group으로 설정하지 않고 어떤 두 시뮬레이션 모델 $MSD(M_i,M_j)$가 최소 cost를 가지더라도 기존에 PG에 속한 모델과의 cost를 계산하여 Seed Group을 설정한다.

표 5. DEVS 특성 기반 모델 분배 알고리즘 II

```
Begin
  SC ← 3 ;
  // SC is the number to do partitioning of the M;
  PG ← null ;
  // PG is initialized into null

  Compute the communication cost for the Set MSD = { msd(i,
  j), for all i, j ∈ M } ;

// Make a decision of the seed nodes be enough to SC
 while SC is full out do
  Get the first min(MSD) for ∀MSD and return Mi, Mj ;
  if the PG is empty then
    Put the Mi as the seed node into ∃scPG ;
```

```
    decrease the value : SC;
    Put the Mj as the seed node into ∃scPG ;
    decrease the value : SC ;
  else
    Put the Mi as the seed into ∃scPG ,
    where Mi ∉ ∀scPG and min(MSD(Mi, all of ∃scPG)) = ∃scPG,
    decrease the value : SC ;

    Put the Mj as the seed into ∃scPG ,
    where Mj ∉ ∀scPG and min(MSD(Mj, all of ∃scPG)) = ∃scPG,
    decrease the value : SC ;
  endif
endwhile
```

표 6.은 분배 규칙에 따른 순차적 실행 알고리즘을 나타낸 것이다.

표 6. DEVS 특성 기반 모델 분배 알고리즘 III

```
// Put All of the remained M into ∀PG, exception of the seed nodes
Sequently, get the max(MSD(Mi, Mj)) for ∀MSD and return Mi, Mj ;
while for all of the remained M, exception of the seed nodes do
    // Process for partitioning the Mi
    // Follows as the Rule 1 :
    R_PG← Get a random ∃scPG, //R_PG is temporary memory;
    Put the Mi into R_PG,
    where Mi ∉ ∀scPG and MSD(Mi, ∀scPG) is equal to 0 ;

    // Follows as the Rule 2 :
    R_PG ← Get a ∃scPG,
    Put the Mi into R_PG,
    where Mi ∉ ∃scPG and MSD(Mi, ∃scPG) is maximum value;

    // Follows as the Rule 3 :
    Move the Mi on ∃scPG to the other ∃scPG,
    where Mi ∈ ∃scPG and MSD(Mi, ∃scPG) is much smaller than MSD(Mi, the others ∃scPG);

    // Follows as the Rule 4 :
    Get the number of allocated model in ∃scPG,
    Get the number of allocated model for the others ∃scPG,
    Move the Mi on ∃scPG having the minimum value of between compared two ∃scPG,
    where (Mi ∈ ∃scPG and Mi ∈ the others ∃scPG) and
    (MSD(Mi, ∃scPG) is equal to MSD(Mi, the others ∃scPG)) ;
    // Process for partitioning the Mj
    // Follows as the Rule 1 :
    R_PG ← Get a random ∃scPG,
    Put the Mj into R_PG,
    where Mj ∉ ∀scPG and MSD(Mj, ∀scPG) is equal to 0 ;


    // Follows as the Rule 2 :
    R_PG ← Get a ∃scPG,
    Put the Mj into R_PG,
    where Mj ∉ ∃scPG and MSD(Mj, ∃scPG) is maximum value;

    // Follows as the Rule 3 :
    Move the Mj on ∃scPG to the other ∃scPG,
    where Mj ∈ ∃scPG and MSD(Mj, ∃scPG) is much smaller than MSD(Mj, the others ∃scPG);

    // Follows as the Rule 4 :
    Get the number of allocated model in ∃scPG,
    Get the number of allocated model for the others ∃scPG,
    Move the Mj on ∃scPG having the minimum value for each ∃scPG,
    where (Mj ∈ ∃scPG and Mj ∈ the others ∃scPG) and
    (MSD(Mj, ∃scPG) is equal to MSD(Mj, the others ∃scPG)) ;
endwhile
begin-end
```

4. DEVS 특성 기반 분배 알고리즘 검증

본 장에서는 제안하는 알고리즘의 효용성을 검증하기 위해 기존 계층적 시뮬레이션 모델 기반 분배 방법과 비교한다.

그림 4.는 그림 1.의 계층적 시뮬레이션 구조를 비 계층적으로 구성을 하고 각 시뮬레이션 모델들 간의 external input 메시지 개수를 Edge에 표현한 그림이다. 실제적으로 계층적 시뮬레이션 모델 구조도 rollback에 영향을 받는 메시지는 그림 4.와 같은 external input 메시지로 계층적 및 비 계층적 구조 모두 메시지 개수는 동일하다. 이에 본 논문에서 제안하는 방법은 모델 기반 분산 시뮬레이션 방법론에 모두 적용이 가능하다.

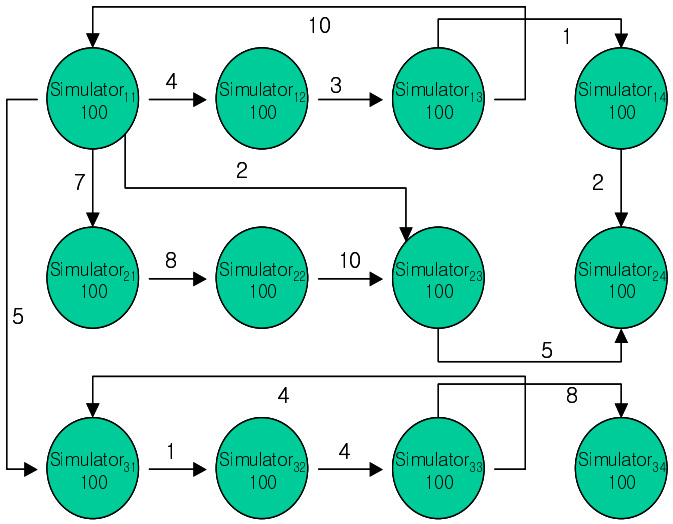

그림 4. DEVS 형식론에 기반 한 비 계층적 시뮬레이션 모델 구성도

그림 5.는 기존 DEVS 모델 기반 계층적 시뮬레이션 모델 분배 방법(***HIPART***)으로 모델들을 분해한 결과이고 그림 6.은 본 논문에서 제안한 DEVS 특성 기반 모델 분배 방법으로 모델들을 분배한 결과이다. 본 논문의 목적은 컴퓨터 노드에 분배된 DEVS 모델들로 구성된 ***PG*** 들 간에 최소 external input message 교환을 목적으로 하고 있다고 앞장에서 설명하였다. 그림 5.에서 보는 것

과 같이 기존 방법으로 시뮬레이션 모델들을 분배하면 전체 external input message가 총 16번이 수행되어진다. 반면에 본 논문에서 제안한 방법은 그림 6.에서 보이는 것과 같이 전체 external input message가 총 12번이 수행되어진다. 이러한 추정 결과에 따라 기존 방법에 비해 본 논문에서 제시한 알고리즘에 대해서 약 25%의 성능 개선 효과를 추정할 수 있다.

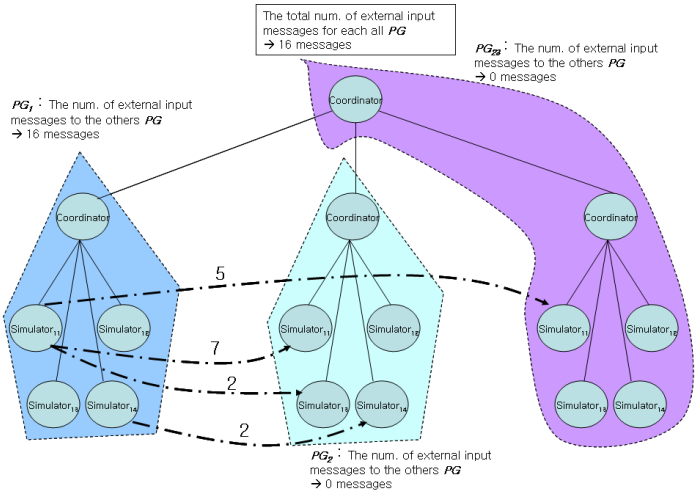

그림 5. 기존 계층적 시뮬레이션 모델 분배[11]에 대한 external input message 수(:case of SC = 3)

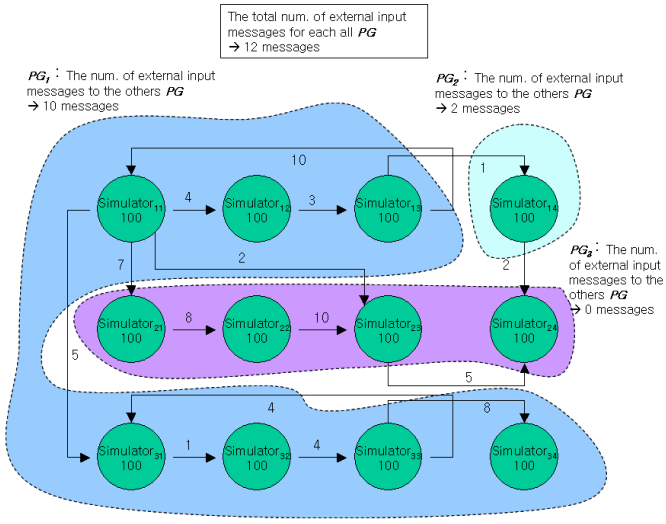

그림 6. DEVS 특성 기반 분배에 대한 external input message 수(:case of SC = 3)

5. 결론 및 향후 계획

본 논문에서 DEVS 특성에 따른 시뮬레이션 모델 분배 방법을 제시하였다. 본 제안 알고리즘의 효용성을 검증하기 위해서 기존 개발된 계층적 시뮬레이션 모델 분배 방법과 비교하여 그 효용성을 나타내었다. 본 논문에서 제안하는 기법은 단순히 DEVS 형식론 기반의 시뮬레이션 방법론뿐만 아니라 DEVS 형식론과 같은 모델 기반 시뮬레이터들에 대해 분산 시뮬레이션 방법론으로 수행할 때 효과적으로 시뮬레이션 모델을 분배할 수 있는 방법이다.

최근에 실생활 환경, 자동차, 항공, 임베디드 시스템 등의 산업 전반에 실시간성 모델 기반 소프트웨어 개발 이슈들이 증가하고 있으며 국가적으로 지원을 하고 있는 상황이다. 이들 시스템들은 특히 더욱 복잡하고 방대한 시뮬레이션 모델로 구성되는데 이들을 검증하기위한 시간 비용을 줄이기 위해 분산 시뮬레이션 방법을 사용하는데 여기서 실시간성 보장, 고성능의 검증 등을 보장하기 위해 본 논문에서 제안하는 방법이 효과적으로 활용될 수 있을 것이다.

본 논문에서는 실제적으로 DEVSCluster 환경에서 기존 계층적 분배 방법과 제안하는 알고리즘을 적용하여 테스트를 수행하지 못하였다. 향후 large-scale 시뮬레이션 모델에 대해서 기존 계층적 분배 방법과 제안하는 알고리즘을 적용하여 implementation하여 명확한 성능 검증을 수행할 것이다.

- 참고문헌 -

[1]. Zeigler, B.P., Praehofer, H., and Kim, T.G.,"Theory of Modeling and Simulation"
[2]. Chandy, K.M. and Misra, J.,"Distributed Simulation: A Case Study in Design and Verification of Distributed Programs", IEEE Trans. on Software Eng. vol. 5, no. 5, (1978) 440-452
[3]. Fujimoto, R.M.,"Optimistic approaches to parallel discrete event simulation", Transactions of the Society for Computer Simulation International, vol. 7, no. 2, October, (1990) 153-1-91
[4]. Fujimoto, R.M.,"Parallel and Distributed Simulation Systems", Proceedings of the 2001 Winter Simulation Conference (2001) 147-157
[5]. Kim, K.H., Seong, Y.R., Kim, T.G., and Park, K.H.,"Distributed Simulation of Hierarchical DEVS Models: Hierarchical Scheduling Locally and Time Warp Globally.", Trans. of SCS vol. 13. no.3. (1996) 135-154
[6]. Kim, K.H., Seong, Y.R., Kim, T.G., and Park, K.H,"Ordering of Simultaneous Events in Distributed DEVS Simulation", Practice and Theory, vol. 5, (1997) 253-268
[7]. Kim, K.H., Distributed Simulation Methodology Based on System Theoretic Formalism: An Asynchronous Approach. Doctoral Dissertation, EE Department, KAIST, 1996.
[8]. Kim, K.H., Kim, T.G., and Park, K.H. Hierarchical partitioning algorithm for optimistic distributed simulation of DEVS models. Journal of Systems Architecture, vol. 44, no. 6-7, pp433-455, 1997.
[9]. Kihyung Kim, Wonseok Kang, Bong Sagong, Hyungon Seo, "Efficient Distributed Simulation of Hierarchical DEVS Models: Transforming Model Structure into a Non-Hierarchical One",33rd Annual Simulation Symposium, (2000), 227-236
[10]. K. Kim and W. Kang, "CORBA-based, Multi-threaded Distributed Simulation of Hierarchical DEVS Models: Transforming Model Structure into a Non-Hierarchical One," LNCS vol. 3046, pp. 167-176, 2004.
[11]. Ki-Hyung Kim, Won-Seok Kang, "A Web Services-based Distributed Simulation Architecture for Hierarchical DEVS Models," LNCS vol. 3397, pp.370-379, 2005.
[12]. 강원석, 김기형, "CORBA를 이용한 멀티스레드 분산 시뮬레이션 환경". 03년 한국정보과학회 추계학술대회논문집, pp.406-408, 2003.10.
[13]. Roland Ewald, Jan Himmelspach, Adelinde M. Uhrmacher, "A NON-FRAGMENTING PARTITIONG ALGORITHM FOR HIERARCHICAL MODELS", Proceedings of the 2006 Winter Simulation Conference, pp.848-855, 2006.