I/O 집약적인 응용의 시뮬레이션 방법론

엄 현 상†

요 약

본 논문에서는 자료 집약적인 분산 또는 병렬 응용의 시뮬레이터들과, 정확도에 대하여 사용자가 정의한 요구 조건이 주어지는 경우에 그 조건을 만족하는 방법들 중에서 가장 효율적인 것을 선택하게 하는 방법론을 제시하고자 한다. 이 방법론은 응용 프로그램의 속성을 기반으로 적당한 시뮬레이션을 선택하는 일련의 시험들로 구성되어 있다. 그리고, 각 시뮬레이터는 응용 프로그램의 실행시간의 두 가지 측정치들, 최소 기대 시간과 최대 기대 시간을 제공한다. 본 논문에서는 현존하는 응용 프로그램들에 이 방법론을 적용한 결과를 제시하고, 각 응용 프로그램의 실행시간보다 수십에서 수백배 빠르면서도 정확하게 그 응용을 시뮬레이션할 수 있다는 것을 보인다.

키워드 : 시뮬레이션 방법론, I/O 집약적인 분산 및 병렬 응용, 성능 예측, 이산 이벤트 시뮬레이션

A Methodology to Simulate I/O-Intensive Applications

Hyeonsang Eom†

ABSTRACT

We introduce a family of simulators for I/O-intensive distributed or parallel applications, and a methodology that permits selecting the most efficient simulator meeting a given user-defined accuracy requirement. This methodology consists of a series of tests to choose an appropriate simulation based on the attributes of the application. In addition, each simulator provides two estimates of application execution time: the minimum expected time and the maximum. We present the results of applying our methodology to existing applications, and show that we can accurately simulate applications tens to hundreds of times faster than the application execution times.

Key Words : Simulation Methodology, I/O-Intensive Distributed and Parallel Applications, Performance Prediction, Discrete-Event Simulation

1. 서 론

저장장치는 컴퓨터 시스템 병목 현상의 주 요인이다. 시스템에 저장되는 자료의 크기가 급속히 커지고 있으므로 이러한 현상은 더욱 악화되고 있다[10]. 예를 들어, 고객 자료는 매 9개월마다 두배가 되고, 많은 위성 자료 저장 장소는 매일 1-2테라바이트의 크기만큼 커진다[14]. 페타바이트 등급의 자료 저장도 몇년 안에 드물지 않게 될 전망이다. 대량의 자료를 저장/사용하는 복잡한 시스템을 효율적으로 사용하기 위해서 I/O 집약적인 응용의 성능 모델을 만드는 것[1, 4, 8, 20]과, 알고리즘의 변경, 하드웨어 구성 요소의 대체, 또는 환경 설정의 변경에 의한 개선을 가능하게 하는 피드백을 주기 위하여 이러한 응용의 성능을 효율적으로 시뮬레이션하는 것이 중요하다.

응용의 성능은 실행할 해당 프로그램과 그 프로그램이 실행될 기계에 의하여 결정된다. 대상으로서의 응용과 기계를 평가하기 위해서는 응용을 실제 하드웨어 상에서 직접 실행할 수 있어야 한다. 그 대안으로, 코드는 기계 아키텍처에 대한 하드웨어 시뮬레이터 상에서 실행될 수 있어야 한다. 그러나, 이러한 실행 구동형의(Execution-Driven) 시뮬레이션은 매우 느리다. 시뮬레이션 속도를 증가시키기 위해서는, 응용 실행 중의 동작에 대하여 포착하는 해당 응용의 추상화된 버전을 구축하는 것과 대단위 수준에서 대상 아키텍처를 모델화하는 시뮬레이터를 그 다음에 사용하는 것이 필요하다.

I/O 집약적인 시스템을 시뮬레이션할 때 해당 시스템을 서로 다른 많은 수준들에서 충실히 고려하는 것이 가능하다. 다양한 시뮬레이션 옵션들은 속도와 정확도 간에 트레이드오프(Trade-Off)를 보인다. 시뮬레이션들 간의 주요한 차이는 시뮬레이션에서의 이벤트 단위수준(Granularity)이다. 예를 들어, 실행 구동형 시뮬레이션에서 명령어들이 기본 블록 내에서 그룹으로 실행되는 동안 각 명령어는 명령어 수

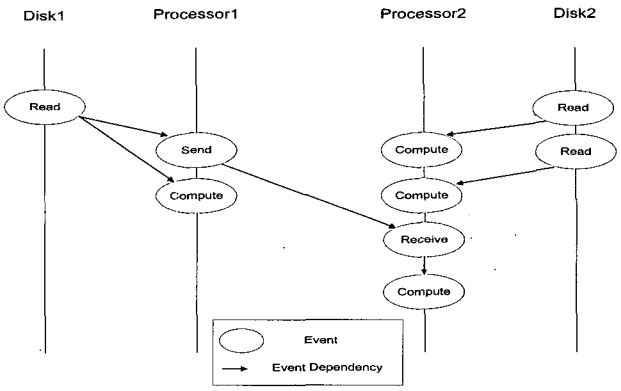
† 정 회 원 : 서울대학교 컴퓨터공학부 교수
논문접수 : 2006년 7월 25일, 심사완료 : 2006년 8월 14일

준의 시뮬레이션에서 에뮬레이션되어야 한다[5]. 따라서, 이러한 시뮬레이션의 결과는 정확하지만 그 속도는 느리다. 더 큰 단위의 수준에서는, 하나의 이벤트는 수백만의 기계 명령어들을 가지는 복잡한 자료 전송 연산들일 수 있다. 이러한 수준에서의 시뮬레이션 결과는 정확하지 않지만 그 속도는 빠르다. 시뮬레이션 충실도(Fidelity) 변화의 두 번째 원인은 이벤트 의존성이다. 어떤 이벤트는 다른 이벤트 후에 일어날 수 있다. 예를 들어, 한 메시지를 전송 받는 연산을 나타내는 수신 이벤트는 해당 송신 연산 이벤트가 발생한 후에 처리되는 것이 필요하다. 정확한 성능 예측을 위하여, 이러한 의존성을 유지하는 것은 중요하고, 이것을 유지를 하기 위해서는 이벤트들이 올바른 순서로 개별 처리되는 것이 요구된다. 그러나, 이러한 유지를 위해서는 해당 비용을 지불해야 한다.

본 논문에서는 자료 집약적인 응용에 대한 이벤트 기반의 시뮬레이션 옵션들과, 사용자가 정확도에 대하여 정의한 요구사항을 바탕으로 그 옵션들 중에서 가장 적당한 것을 선택하게 하여 주는 방법론을 제안한다. 이 방법론은, 응용과 해당 실행의 속성들을 바탕으로, 요구되는 정확도를 제공하는 최소 비용의 시뮬레이션을 선택하는 일련의 시험들로 구성된다. 이 방법론의 주요 장점은 여러 시뮬레이션 옵션들이 주어졌을 때, 더 적은 비용의 옵션이 충분히 정확하면 그것을 선택하여 사용할 수 있게 하는 점이다. 본 논문에서 제안하는 기술은 새로운 시스템을 설정하거나 새로운 하드웨어를 디자인하는데 사용할 수도 있지만, 응용의 개발자들이 서로 다른 응용 옵션을 고르거나 요청된 (예를 들어, 슈퍼컴퓨터 센터에서 해당 할당을 관리하는) 응용 자원들을 선택하는데 도움을 주도록 하는데 우선적으로 사용하기 위한 것이다. 현재의 방법은 응용 이벤트 자료를 처리함으로써 자동적으로 수행되는 많은 시험들이 반자동적으로 연계된 형태로 구성되어 있다.

2. 실행 모델

I/O 집약적인 응용의 실행은 자료 블록이라고 불리는 아이템들의 집합에 대하여 수행되는 일련의 이벤트들에 의하여 모델화된다. 이벤트들은 각 자료 블록에 대하여 읽고, 쓰고, 전송하고, 전송 받고, 그리고 계산하는 연산들을 나타낸다. 이벤트들의 실행은 다른 이벤트들의 실행에 의존적일 수 있다. 예를 들어, 메시지 송신 이벤트는 해당 수신 이벤트보다 선행되어야 한다. 게다가, 다수의 이벤트들이 하나의 이벤트에 의존적일 수도 있다. 특정한 이벤트들과 응용 실행 상의 이벤트들 간의 관계는 동적인 작업 흐름 그래프들(Work-Flow Graphs, WFGs)로 설명될 수 있다. WFG에서 노드(Node)는 이벤트에 대응하고 에지(Edge)는 해당 이벤트들 간의 의존성을 나타낸다. 이 추상적인 모델은 자료 집약적인 응용의 실행시간을 결정하는 주요 활동들을 나타내기 때문에 시뮬레이터를 위한 자연스런 표현 방법으로서의 역할을 한다.



(그림 1) 작업 흐름 그래프

(그림 1)은 두개의 프로세서들과 두개의 디스크들 상에서 자료 집약적 응용의 실행을 나타내는 세가지 WFG들을 보여 준다. 디스크1과 디스크2는 각각 프로세서1과 프로세서2에 부속되어 있다. 프로세서와 디스크에 대한 수직선은 장치에 의한 이벤트들의 처리를 나타낸다. 이 그래프에서 세 개의 자료 블록들은 디스크에서 읽혀진 다음, 지역적인 또는 원격 계산에 사용된다. 여기서는 프로세서1 상에서 계산 이벤트 전에 송신 이벤트가 위치되었지만 이 순서로만 이 이벤트들이 처리되어야 하는 순서적인 제약은 없다.

WFG들은 추가 코드가 삽입된(Instrumented) 프로그램들의 실행에 의하여 생성된다. 이 프로그램은 실제 프로그램 또는 응용 에뮬레이터가 될 수 있다. 응용 에뮬레이터는 응용 전체 또는 한 종류의 응용 집합의 실행 동작을 나타내는 동작을 하는 대상 응용의 커널이다. 에뮬레이터들은 해당 응용들의 계산과 자료 접근 패턴을 제어된 방법으로 모델화한다. 그것들은 응용의 동적 동작을 효율적으로 표현한다. 응용 에뮬레이터에 대한 추가 정보는 [23]에서 찾을 수 있다.

연구개발한 하드웨어 시뮬레이터들은 응용들이나 에뮬레이터들로부터 생성된 WFG들의 이벤트들을 처리함으로써 이산 이벤트 시뮬레이션을 수행한다. 이 시뮬레이터들은 하나의 전역 시뮬레이션 시계(Clock)와 각 하드웨어 구성요소에 대하여 하나씩의 자원 시계를 유지한다. 시뮬레이션 시스템에서 하나의 자원 시계는 각각의 디스크 및 프로세서와 연계되어 있다. 하나의 이벤트가 처리될 때, 해당 자원 시계의 값은 갱신된다. 전역 시계는 가장 큰 자원 시계 값을 계속 유지한다. 프로세서 시계는 네트워크 시계로서도 사용된다. 왜냐하면, 프로토콜 처리가 프로세서에 의하여 수행된다고 가정하기 때문이다. 현재의 클러스터들 상에서는 프로토콜 처리가 하드웨어 지연 시간의 대부분을 차지하기 때문에 메시지의 "비행" 시간은 고려하지 않는다.

3. 시뮬레이션 옵션들

본 장에서는 다른 시뮬레이션 옵션들과 그 상대적인 비용을 설명한다. 여기서, 고려하는 시뮬레이션 옵션들은 모두 이산 이벤트 기반이고 시뮬레이션되는 이벤트들의 단위수준

과 시뮬레이션에 쓰인 하드웨어 자료의 충실도에서 차이가 있다. 이 옵션들은 소단위(Fine-Grained)와 대단위(Coarse-Grained) 두가지 주요 종류로 나뉜다.

대단위 시뮬레이션에는 의존성 보존과 의존성 비보존의 두 가지 종류가 있다. 의존성 보존 방법에서는 이벤트들이 해당 의존 순서에 따라 하나씩 처리된다. 그러나, 의존성을 보존하지 않는 자원 이벤트 시뮬레이션에서는 각 자원에 대한 모든 이벤트들은 하나의 "자원 이벤트"로 통합된다. 핵심적으로, 각 자원에 대한 요청들의 처리 시간을 간단히 합하고 결과 값들 중 가장 큰 것을 보고한다. 극단적으로 간단한 방법이지만, 이 시뮬레이션 옵션은 응용의 최소 실행시간에 대한 아이디어를 제공한다.

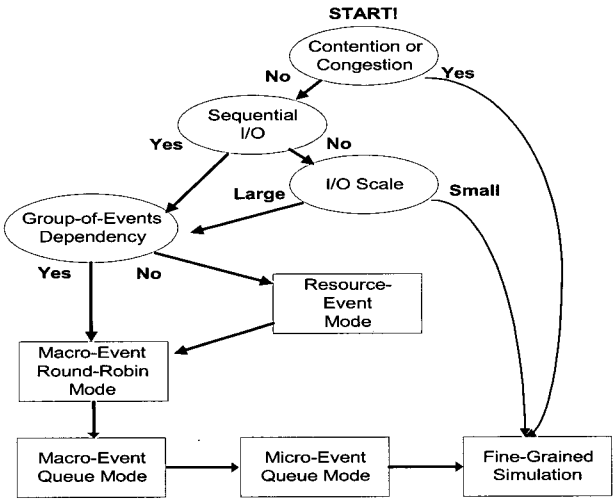
의존성 보존 시뮬레이션에는 매크로-이벤트-라운드-로빈(Macro-Event-Round-Robin), 매크로-이벤트 큐(Macro-Event Queue), 그리고 마이크로-이벤트 큐(Micro-Event Queue)의 세 가지 하위 방법들이 있다. 처음 두 방법들은 효율적이나 세 번째 방법에 비하여 부정확하다. 마이크로-이벤트 큐 기반의 시뮬레이션에서는 전역 이벤트 처리 큐를 사용함으로써 모든 이벤트들을 정확한 순서로 배열한다. 매크로-이벤트 큐 옵션에서는 WFG가 "매크로 이벤트"로 간주됨으로써 간단한 전역 WFG 큐를 사용하여 WFG들의 처리 순서를 결정한다. 다른 한편으로, 매크로-이벤트-라운드-로빈 방법에서는 큐를 사용하지 않고 모든 프로세서들에 대하여 라운드 로빈 방법으로 WFG들을 선택하여 순서를 결정한다.

소단위 시뮬레이션에서는 공유된 자원에 대한 경합/정체, 디스크 탐색(Seek) 시간, I/O 시간의 (예를 들어, 디스크 배치에 따른) 변이, I/O를 위한 CPU 사용, 플랫폼 의존적인 원인에 의한 이벤트 종료 지연, 등의 실행 중에 나타나는 현상들을 모델화한다.

4. 시뮬레이션 옵션들 중에서의 선택

본 장에서는 사용자가 정의한 목적 오류 한계에 부합하는 가장 효율적인 시뮬레이션 방법을 선택하게 하는 I/O 시뮬레이션의 분류론을 제안한다. 주요 아이디어는 일단 대단위 시뮬레이션으로 시작하여 사용자가 지정한 허용 범위 내로 오류 범위가 감소할 때까지 반복적으로 좀더 복잡한 방법을 차례로 사용해 나가는 것이다. (그림 2)는 해당 결정 트리를 보여준다. 이 그림에서 타원은 결정이 이루어지는 시험을 나타낸다. 사각형은 특정 시뮬레이션 기술을 수행하고 그 기술이 정확도 요구에 부합하는지를 시험하는 것을 나타낸다. 이 시험들의 결과를 바탕으로 새로운 시뮬레이션 옵션이 수행되고 해당 정확도가 평가된다.

처음에, 대단위 옵션을 사용하여 실행시간 예측이 가능한지를 판단하기 위하여 WFG들을 시험한다. 이 타당성 시험에서는 대단위 옵션들이 간과할 수 있는, 치명적인 응용 또는 플랫폼 성능 요소가 있는지를 검사한다. 구체적으로, 경합/정체, 순차적인 I/O, 그리고 I/O 스케일의 세 가지에 대하여 시험한다.



(그림 2) 시뮬레이션 방법을 선택하는 결정 트리

- **경합 혹은 정체 시험:** 이 시험은 자원들에 대한 실시간 경합이나 정체가 있는지를 검사한다. 경합이나 정체를 시험할 때 제한된 네트워크 수용 능력을 고려한다. 통신 종점 정체[22]와 링크 경합 현상은 통신비율이 높고 다수의 프로세서들이 다대일 또는 일대다 통신에 참여할 때 발생한다. 제안하는 방법론에서는 시뮬레이션될 응용과 동일한 통신 매개 변수 값을 사용하는, 짧은 두 개의 통신 전용 시험 프로그램들을 실행함으로써 통신 경합이나 정체가 있는지를 결정한다. 첫번째 프로그램은 자료를 서버로부터 수신하거나 서버에게 송신함으로써 경합이나 정체 효과를 포함한 통신 시간을 측정한다. 두번째 프로그램은 단순한 핑퐁 벤치마크를 사용함으로써 경합이나 정체 효과를 제외한 통신 시간을 측정한다. 통신 경합 또는 정체 효과는 두 시험 프로그램의 블록단위 통신 시간의 차와 프로세서당 통신한 자료 블록들의 최대 개수의 곱이다. 경합/정체 시험을 위한 두 통신 프로그램들의 매개 변수들은 다음과 같다.
  - *팬-아웃(Fan-Out)*: 얼마나 많은 프로세서들이 하나의 자료 블록을 수신하는가를 나타내고, 대상 응용의 실행 동안 어떤 프로세서와 통신하는 프로세서들의 최대 개수로서 설정된다.
  - *통신 속도*: 프로세서당 자료 블록 단위 I/O 시간에 대한 전송되는 블록들의 수로서 다음과 같이 계산된다.

$$FO * N_{local\_disk} / T_{IO}$$

여기서, FO는 응용의 실행 상의 최대 팬-아웃 값이고, N_{local_disk}는 실행 동안 사용된 어느 한 노드의 최대 디스크 개수이다. 그리고, T_{IO}는 블록 단위 I/O 시간의 평균 값이다. FO와 N_{local_disk}의 값들은 응용의 입력 WFG들로부터 계산된다. T_{IO}는, 자료 블록들의 크기, 사용된 로컬 디스크들의 개수, 그리고 디스크 (순차적 또는 완전히 랜덤한) 탐색 방법에 대한 매개 변수들을 갖는 간단한 I/O 벤치마크 프로그램을 실행함으로써 측정된다. T_{IO}를 측정하기 위하여 블록 크

기 매개 변수는 응용에 의하여 사용되는 평균 블록 크기로 설정되고 다른 두 매개 변수들은 위에서 정의된 N_{local_disk}과 순차 탐색 방법으로 각각 설정된다.

-*자료 블록 크기*: 시험 프로그램들의 자료 블록 크기이고 응용에 의하여 사용되는 평균 자료 블록 크기로 설정된다.

- **순차 I/O 시험**: 이 시험에서는 자료 블록에 대한 접근이 실행 동안 각 디스크에 대하여 순차적인지를 평가한다. 이 정보는 인접한 디스크 요청들의 파일 오프셋들을 검사함으로써 WFG 기록(Log)에서 추출된다. I/O가 완전히 순차적이지 않다면 디스크 I/O 탐색 시간을 시뮬레이션할지를 결정해야 한다.
- **I/O 스케일 시험**: 이 시험에서는 디스크 탐색 시간이 시뮬레이션에서 고려되어야 할 정도로 충분히 큰지를 결정한다. 시뮬레이션된 응용과 비슷한 크기의 I/O 연산을 수행하는 매개 변수화된 시험 프로그램이 실행된다. 이 시험 프로그램은 한번은 완전히 순차적인 I/O 요청들을 사용하고, 다른 한번은 완전히 랜덤한 I/O 요청들을 사용하여 두차례 실행된다. 랜덤 검색에 때문에 부가되는 시간이 목표로 하는 오류보다 크다면, 탐색 시간이 고려된 소단위 시뮬레이션이 수행되는 것이 필요하다.
- **이벤트-그룹-의존성(Group-of-Events-Dependency) 시험**: 이 시험에서는 의존성 비보존 자원 이벤트 시뮬레이션이 선택될 수 있는지를 결정한다. 이 시험은 이벤트들의 그룹들 간에 의존성이 있는지를 검사한다. 이벤트 그룹 의존성은 하나의 이벤트 그룹이 다음 그룹의 어떤 이벤트가 시작되기 전에 먼저 종료하는 특성을 말한다. 예를 들어, 어떤 데이터베이스 응용에서 하나의 질의에 대한 결과는 다음 질의가 시작될 수 있기 전에 반드시 처리되어야 한다. 이같은 의존성 정보는 WFG들 내에 명시적으로 표시된다.

5. 시뮬레이션의 오류 범위

제안하는 시뮬레이터들은 응용의 기대 실행시간에 대하여 두가지 값을 생성한다. 이것들은, 대응되는 가장 짧은 실행시간의 낙관적인 추정 값인 T_{pl}과 가장 긴 실행시간에 대한 비관적인 값인 T_{ph}이다. 다음과 같이 계산한 최대 오류가 목표로 하는 오류 보다 작으면 시뮬레이션은 선택되고, 그렇지 않으면 다음으로 가장 상세한 시뮬레이션이 수행된다.

$$\frac{|T_{ph} - T_{pl}|}{T_{pl}}$$

T_{ph}는 최종 전역 시계 값과 주어진 시뮬레이터에서 고려되지 않는 측면들로 인한 오류의 보수적인(Conservative) 추정 값의 합이다. T_{pl}은 최종 전역 시계 값에서 해당 시뮬레이터의 최적화로 인하여 야기되는 추가적인 오류의 값을 뺀 차이이다. 본 장에서는 각 시뮬레이터를 세부적으로 설명하고자 한다.

5.1 자원 이벤트 시뮬레이션

자원 이벤트 시뮬레이션에서는 이벤트 간의 의존성을 고려하지 않고 각 자원의 모든 이벤트들을 하나의 "자원 이벤트"로서 처리한다. 이것은 제안하는 기술들 중에서 가장 간단한 방법이다. 이 시뮬레이션이 사용될 수 있는지를 결정하는 오류 한계 시험은 하나의 자원이 실행시간을 결정하면서 다른 활동들의 실행시간에 대한 영향을 없애는지를 확인한다. 하나의 자원이 실행시간을 결정한다면, 정확도를 감소시키지 않으면서도 시뮬레이션에서 다른 부분들을 고려하지 않을 수 있다. 가장 큰 시계 값의 자원을 실행시간 결정 후보로 간주하여 T_{pl}을 이 값으로 설정한다.

$$T_{pl} = \max_{R(esource)}(T^R)$$

여기서, T^R은 자원 R의 최종 시계 값이다. 그러나, 실행시간을 결정하는 자원이 아닌 다른 자원의 활동이 순서적인 제약 때문에 실행시간에 대한 영향이 감추어지지 않을 수 있다. 이를 바탕으로 T_{ph}는 다음과 같이 계산된다.

$$T_{ph} = \begin{cases} \sum_{p(rocessor)} (T^p + E^p_{Overlap}) + \max_{d(isk)}(T^d + E^d), w/\,comm. \\ \max_{\substack{p(rocessor)\& \\ local\_d(isk)}} (T^p + E^p_{Overlap} + T^d + E^d), \quad w/o\ comm, \end{cases}$$

여기서, $E^p_{Overlap}$은 프로세서 p에 의하여 야기된 총 최대 오류이고 E^d는 디스크 d에 대한 해당 오류이다.

최악의 경우에 모든 자원들에 대하여 하나의 자원의 모든 이벤트들은 다른 자원들의 모든 이벤트들 다음에 (예를 들어, 자원 이벤트 다음에 자원 이벤트의 방식으로) 처리되는 것이 필요할 수 있다. 따라서, 간단하면서 가장 보수적인 실행시간의 추정치는 모든 자원 시계들의 값들과 그 자원들에 대한 총 최대 오류의 합이 될 수 있다. 그러나, 이러한 추정 방식은 하나의 디스크가 다른 디스크들과 병렬적으로 동작하는 면을 고려하지 않는 문제가 있다. 그러므로, T_{ph}는 최악의 경우의 병렬 동작 디스크 시간과 최악의 경우의 프로세서 시간의 합이다. 여기서, 프로세서 시간은 서로 통신하는 경우에 모든 프로세서 시계들과 프로세서들의 비중첩 오류 값들의 합으로서 계산된다.

디스크로 인한 최대 오류 E^d를 계산할 때, 랜덤 I/O와 I/O 시간 변이 효과를 고려할 필요가 있다. 그러므로, 디스크 d에 대한 E^d는 다음과 같이 계산된다.

$$E^d = E^d_{Rand\_IO} + E^d_{IO\_Var}$$

랜덤 I/O 효과는 디스크들의 검색 시간을 시뮬레이션하지 않기 때문에 생기는 오류를 나타낸다. I/O 스케일 시험에서 계산된, 순차적인 탐색을 사용한 블록단위 I/O 시간과 랜덤

탐색을 사용한 블록단위 I/O 시간의 차인 T_{Rand_Seek}는 이 현상들을 정량화하기 위하여 사용된다. 디스크 d로 인한 해당 최대 오류는 다음과 같다.

$$E^{d}_{Rand\_IO} = T_{Rand\_Seek} * N^{d}_{IO}$$

여기서, N^{d}_{IO}는 해당 디스크에 대한 I/O 연산들의 총 수이다.

I/O 시간 변이 효과는 프로세서들 간의 I/O 시간의 차이를 나타내고, 어떤 두 프로세서들 간의 블록단위 I/O 시간의 최대차 T_{IO_Diff}로서 측정된다. 이 효과 때문에 발생되는 디스크 d에 대한 최대 오류는 다음과 같다.

$$E^{d}_{IO\_Var} = T_{IO\_Diff} * N^{d}_{IO}$$

프로세서들로부터 야기된 최대 오류 $E^{p}_{Overlap}$에 대해서는, 계산과 I/O의 중첩을 고려해야 한다. 중첩 효과는 I/O를 위한 CPU 사용을 나타내고, 이 현상의 최악의 경우 블록단위 효과는 I/O 루틴에서 사용된 블록단위 벽시계(Wall-Clock) 시간 T_{IO_CPU}로서 계산된다. 이 효과 때문에 발생되는 프로세서 p에 대한 총 오류는 다음과 같다.

$$E^{p}_{Overlap} = T_{IO\_CPU} * N^{p}_{IO}$$

여기서, N^{p}_{IO}는 해당 프로세서에 대한 I/O 연산들의 총 개수이다.

5.2 매크로-이벤트-라운드-로빈 시뮬레이션

매크로-이벤트-라운드-로빈 시뮬레이션에서는 디스크들에 대하여 라운드 로빈 방식으로 다음에 처리할 WFG를 선택하면서 단일 WFG의 이벤트들을 "매크로 이벤트"로서 단번에(Atomically) 처리한다. 해당 오류 한계를 계산하기 위하여 균형, 팬-아웃, 디스크 개수, 이벤트 종료 지연, 랜덤 I/O, I/O 시간 변이, 그리고 계산과 I/O의 중첩을 고려한다. T_{pl}과 T_{ph}는 다음과 같이 계산된다.

$$T_{pl} = T_{p} - E_{Balance} - E_{FO}$$

$$T_{ph} = T_{p} + E_{Event\_Delay} + E_{Rand\_IO} + E_{IO\_Var} + E_{Overlap}$$

여기서, T_{p}는 시뮬레이션 완료 시의 최종 전역 시계 값이다.

불균형적인 통신 오류, $E_{Balance}$는 두 프로세서들 간의 축적된 통신 시간의 최대 차이로서, 매크로 이벤트의 라운드 로빈 처리로 인한 시뮬레이션 오류를 표시한다. I/O와 통신이 중첩되지 않고 서로 다른 디스크에 대하여 같은 수의 I/O 연산이 수행될 때, 어떠한 두 디스크들 간의 I/O 연산 완료시간의 최대 차이가 $E_{Balance}$이다. 이러한 완료시간이 디스크마다 다르고, 어떤 디스크의 I/O 이벤트가 동일 디스크의 이전 I/O 이벤트의 종료시간에 처리되기 시작하므로, 다음 I/O 이벤트가 디스크에 대하여 라운드 로빈 방식으로 선택될 때 I/O 이벤트가 실제 순서에 맞지 않게 처리될 수 있다. 최악의 경우에, I/O 이벤트는 $E_{Balance}$만큼 먼저 처리될 수도 있다. I/O 연산이 완료되었을 때, 블록을 받는 모든 프로세서들의 시계 값들이 이 시간만큼 증가된다(왜냐하면, 메시지 전달은 자원 시계 값을 송신자나 수신자의 시계 값들의 최대값으로 증가시키기 때문이다). WFG의 모든 이벤트들이 단번에 처리되므로, I/O 이벤트의 종료 시간은 어느 다른 이벤트들이 해당 I/O 이벤트와 수신 이벤트 사이에서 처리되지 않는 상태에서 즉시 수신자에게 전달된다. 그리하여, 최악의 경우에, $E_{Balance}$만큼 빨리 처리되는 하나의 I/O 이벤트는 해당 수신자의 "대기" 시간 동안 다른 이벤트들이 처리되지 않도록 하게 되고, 결과적으로 그러한 이벤트들의 처리가 그 시간만큼 지연되게 된다.

팬-아웃 오류, E_{FO}는 WFG 이벤트들을 단번에 처리함으로써 야기되는 이벤트 단위수준 증대로 인한 시뮬레이션 오류를 나타내며 다음과 같이 계산된다.

$$E_{FO} = \begin{cases} E_{FOl}, & I/O\text{와 통신이 중첩되는 경우} \\ E_{FOh}, & \text{반대의경우} \end{cases}$$

I/O가 완벽하게 통신과 중첩된다면 팬-아웃 오류는 다음과 같다.

$$E_{FOl} = FO * T_{comm}$$

여기서, FO는 최대 팬-아웃이고 T_{comm}는 (경합/정체 없는) 블록단위 통신시간이다. 추가적인 잠재적 오류는 모든 WFG 이벤트들이 어떤 다른 WFG 이벤트의 처리에 의하여 중단되지 않고 연속적으로 처리되기 때문에 발생한다. 이것은 이벤트 처리에 있어서 도중에 처리되는 기회를 상실하게 만들 수 있고, 따라서 어떠한 이벤트들을 처리함에 있어 지연을 야기할 수 있다. 매크로 이벤트 시뮬레이션에서 모든 WFG 이벤트들이 한꺼번에 처리되기 때문에, 노드가 "대기" 중인 동안에는 어떠한 다른 이벤트들이 처리되지 못하게 하면서 해당 WFG에 대하여 데이터 블록을 받는 각 노드의 프로세서 시계 값을 증가시키는 결과를 가져온다. 그 대기 시간 동안에 처리되어야 했었던 어떠한 이벤트가 나중에 처리된다면, 시뮬레이션 시간은 그만큼 증가될 수 있다. 따라서, 해당 하한을 계산할 때, T_{p}에서 E_{FOl}을 뺀다.

한편, 만약 I/O가 통신과 완벽하게 중첩될 수 없다면, 팬-아웃 오류는 다음과 같다.

$$E_{FOh} = (FO + N_{disk}) * T_{comm}$$

여기서, N_{disk}는 사용된 디스크들의 총 수이다. 이와 같이 계산하는 이유는 $N_{disk} * T_{comm}$ 만큼 시뮬레이션 오류가 팬-아웃 오류에 더해질 수 있기 때문이다. 모든 WFG 이벤트들이 단번에 처리되기 때문에, 일부 통신 이벤트는 해당 WFG 처리 도중에 다른 WFG 이벤트가 처리되는 경우보다 먼저 처리될 수 있다. 이러한 일이 발생하면, 통신 이벤트들은 그들이 원래 처리되어야 할 시간보다 먼저 처리되게 된다. 최악

의 경우에 각 디스크당 하나씩, 총 N_{disk} 개의 통신 이벤트들은 그들이 원래 처리되어야 할 시간보다 더 일찍 처리될 수도 있다. 이 경우에, I/O 이벤트의 종료는 $N_{disk}*T_{comm}$만큼 부정확하게 지연될 수 있다. 결과적으로, 이 효과는 데이터 블록의 전달 받는 모든 노드들에게 전파될 수 있고 해당 크기의 시뮬레이션 오류를 야기하게 된다.

이벤트 종료 지연, 랜덤 I/O, I/O 시간 변이, 그리고 계산 및 I/O 중첩은 마이크로-이벤트 또는 매크로-이벤트 시뮬레이션을 사용할 때 응용 실행을 과소평가하는 오류의 원인이 된다.

이벤트 종료 지연으로 인한 시뮬레이션 오류, E_{Event_Delay}는 의존적인 연산의 완료될 때까지 어떤 연산의 완료 지연 효과를 나타낸다. 그 지연은 어느 두 프로세서들 간의 시뮬레이션 시간 값의 최대 차이로 측정된다.

$$E_{Event\_Delay} = \max_{p(rocessor)}(T^p) - \min_{p(rocessor)}(T^p)$$

랜덤 I/O, I/O 시간 변이, 그리고 계산 및 I/O 중첩으로 인한 오류인, E_{Rand_IO}, E_{IO_Var}, 그리고 $E_{Overlap}$은 자원 이벤트 시뮬레이션에 대하여 5장 1절에서 기술한 현상들의 효과에 대한 수식들을 이용하여 다음과 같이 계산된다.

$$E_{Rand\_IO} = \max_{d(isk)}(E^d_{Rand\_IO})$$
$$E_{IO\_Var} = \max_{d(isk)}(E^d_{IO\_Var})$$
$$E_{Overlap} = \max_{p(rocessor)}(E^p_{Overlap})$$

5.3 매크로-이벤트 큐 시뮬레이션

매크로-이벤트 큐 시뮬레이션에서는 전역 WFG 큐를 사용하여 WFG를 "매크로 이벤트"로서 처리한다. 오류 범위는, 매크로-이벤트-라운드-로빈 시뮬레이션의 정확도에 영향을 주는 요소들 중에서 불균형 통신을 제외한 같은 오류 원인들을 고려한다. 이러한 오류 원인들은, 5장 2절에서 설명한 바와 같이 통신 불균형이 매크로 이벤트의 실제 순서에 맞는 처리로 인한 시뮬레이션 오류를 야기하지 않기 때문에 고려되지 않는다. T_{pl}과 T_{ph}는 다음과 같이 계산된다.

$$T_{pl} = T_p - E_{FO}$$
$$T_{ph} = T_p + E_{Event\_Delay} + E_{Rand\_IO} + E_{IO\_Var} + E_{Overlap}$$

여기서, T_p 는 시뮬레이션 결과 값이다.

5.4 마이크로-이벤트 큐 시뮬레이션

마이크로-이벤트 큐 시뮬레이션에서는 전역 WFG 큐를 사용하여 WFG 이벤트들을 "마이크로 이벤트들"로서 처리한다. 이 시뮬레이션의 오류 범위를 계산할 때는 시뮬레이션의 정확도를 결정하는 오류 원인들으로서 이벤트 종료 지연, 랜덤 I/O, I/O 시간 변이, 그리고 계산 및 I/O 중첩을 고려한다. T_{pl}과 T_{ph}는 다음과 같이 계산된다.

$$T_{pl} = T_p$$
$$T_{ph} = T_p + E_{Event\_Delay} + E_{Rand\_IO} + E_{IO\_Var} + E_{Overlap}$$

여기서, T_p 는 시뮬레이션 결과 값이다.

5.5 소단위 시뮬레이션

소단위 시뮬레이션에서는 장치 구성 요소 간의 자료 이동과 같은 상세한 수준의 응용 이벤트를 나타내는 소단위 이벤트를 처리하기 위하여 전역 이벤트 큐를 사용한다. 시뮬레이션의 정확도는 대단위 시뮬레이션에 영향을 주는 불균형 통신, 팬-아웃, 디스크 개수, 이벤트 종료 지연, 랜덤 I/O, I/O 시간 변이, 그리고 계산 및 I/O 중첩 중 어떠한 오류 원인에도 영향 받지 않는다. 정확도에 영향을 주는 다른 오류 원인들은 I/O와 통신을 중첩할 수 있는 시스템 능력에서의 한계 등을 포함한다. 그리고, 공유 자원에 대한 경합 및 정체는 오직 제한된 정도로 모델화되었다. 관심있는 응용 타입에 있어서 통신 및 I/O 연산은 큰 자료 블록에 대하여 수행되고 따라서 다른 원인들로부터의 영향은 매우 작은 경향이 있기 때문에, 기타 원인들로 인한 오류도 없다고 가정한다.

6. 방법론 적용의 예

본 장에서는 제안하는 방법론의 정확도를 입증할 수 있는 예들을 보여주고, I/O 집약적인 응용의 실행시간을 효율적으로 예측하는 기능을 입증하고자 한다. Titan과 Virtual Microscope, 두가지의 자료 집약적인 응용에 제안하는 방법론을 어떻게 적용하는지에 대하여 설명한다. Titan은 위성 자료를 저장하고 자료에 대한 질의를 처리하는 병렬 비공유 데이터베이스 서버이다[4]. Virtual Microscope는 시각적인 현미경 슬라이드의 디지털 이미지에 대하여 다수의 클라이언트들로부터의 질의를 처리하는 서버이다[8].

6.1 오류 한계의 입증

제안하는 방법론에서 사용된 오류 한계의 정확도를 나타내기 위하여 Titan 및 Virtual Microscope에 대한 실제 측정 값들이 시뮬레이션 옵션들의 오류 범위 안에서 있다는 것을 보여준다. 각 노드마다 4개의 디스크가 부착된 IBM SP2 노드 12개를 대상으로 방법론의 유효성을 입증하는 실험을 수행하였다.

<표 1>은 Titan과 Virtual Microscope에 대한 시뮬레이션 결과를 보여준다. 이 표의 열들은 예측된 시간, 해당하는 실제 측정값에 대한 예측된 시간의 실제 오류(퍼센트), 그리고 원래(Raw) 및 결합된(Combined) 오류의 (상 및 하)한계들과 각 옵션에 대한 오류 범위를 나타낸다. 각 옵션의 상한과 하한은 5장에서 보여준 예측 시간과 최대 오류를 기반한 방법으로 계산된다. 결합된 하한과 상한은 각각 모든 가용한 하한들의 최대 및 모든 상한들의 최소이다. 이것들을 통해 실행된 시뮬레이션의 결합에 기반을 둔 혼합(Hybrid) 오류 범위를 계산할 수 있다. 결합된 오류 한계는 <표 1>의

〈표 1〉 12GB 자료를 가진 Titan과 Virtual Scope 에뮬레이터에 대한 시뮬레이션 결과

Titan 시뮬레이션 옵션	예측 시간 (초)	실제 오류 (%)	시간 범위					
			원래			결합된		
	측정 시간: 336.8		하한	상한	오류 (%)	하한	상한	오류 (%)
Resource Event	185.6	44.9	N/A	N/A	N/A	N/A	N/A	N/A
Macro-Event RR	355.8	5.6	335.9	404.4	20.4	335.9	404.4	20.4
Macro-Event Queue	302.4	10.2	302.2	351.0	16.1	335.9	351.0	4.5
Micro-Event Queue	302.3	10.2	302.3	350.9	16.1	335.9	350.9	4.5

Virtual Microscope 시뮬레이션 옵션	예측 시간 (secs)	실제 오류 (%)	시간 범위					
			원래			결합된		
	측정 시간: 683.4		하한	상한	오류 (%)	하한	상한	오류 (%)
Resource Event	619.2	9.4	619.2	705.0	13.9	619.2	705.0	13.9
Macro-Event RR	619.2	9.4	619.2	705.0	13.9	619.2	705.0	13.9
Macro-Event Queue	619.2	9.4	619.2	705.0	13.9	619.2	705.0	13.9
Micro-Event Queue	619.2	9.4	619.2	705.0	13.9	619.2	705.0	13.9

〈표 2〉 1,200개의 노드 상에 12TB 자료를 가진 Titan에 대한 시뮬레이션 결과

시뮬레이션 옵션	예측 시간 (초)	시간 범위					
		원래			결합된		
		하한	상한	오류 (%)	하한	상한	오류 (%)
Resource Event	2,364.5	N/A	N/A	N/A	N/A	N/A	N/A
Macro-Event RR	5,177.9	4,844.8	7,660.3	58.1	4,844.8	7,660.3	58.1
Macro-Event Queue	*2,650.7*	*2,630.7*	*5,133.1*	*95.1*	*4,844.8*	*5,133.1*	*6.0*
Micro-Event Queue	2,650.6	2,650.6	5,133.0	93.7	4,844.8	5,133.0	5.9

마지막 세개의 열에서 볼 수 있다.

입력 매개 변수 집합은 주로 자료 블록 크기, 입력 자료 파일의 위치, 처리할 자료의 총 볼륨, 2D 처리기 그물 구조, 프로세서당 총 디스크 개수, 질의 윈도우의 위치와 크기, 질의의 개수, 그리고 블록당 계산 시간 등을 포함한다. Virtual Microscope에 대한 추가적인 매개 변수들은 슬라이드의 크기와 개수이다. 모든 경우에, 측정된 실행시간은 Titan의 경우 336.8초이고 Virtual Microscope의 경우 683.4초인데 이 값들은 <표 1>에서 보는 바와 같이 오류 한계 내에 있다. Titan의 경우 <표 1>에서 자원 이벤트 옵션에 대한 결과를 나타냈지만, 이벤트 그룹 의존성 시험이 해당 기술의 결과를 신뢰할 수 없다는 결과를 나타내기 때문에 이 경우에 있어서 오류 한계를 보여주지 않는다. Virtual Microscope의 경우에 모든 대단위 옵션들의 결과가 동일하기 때문에 Virtual Microscope에 대하여 <표 1>에서의 원래 및 결합된 한계는 같다. 그것들은 랜덤 I/O, 이벤트 그룹 의존성, 그리고 통신 특성이 없기 때문에 동일하다.

6.2 방법론의 적용

이 장에서는 연구개발한 방법론을 사용하여 적절한 시뮬레이션 옵션을 선택하는 과정을 보여주고자 한다. 각 응용에 대하여, 이전의 유효성 입증 연구에서보다 더 큰 설정값들을 사용하였다. Titan에 12TB와 1,200 노드수를, Virtual Microscope에는 1.2TB와 12 노드수를 사용하였다. 모든 경우에, 목표 오류 한계를 20%로 설정하였다.

<표 2>는 더 큰 설정에서의 Titan 버전에 대하여 사용될 수 있는 시뮬레이션 옵션들의 결과를 나타낸다. 이 표는 예측 시간, 원래 및 결합된 오류 (상 및 하)한계, 그리고 오류 범위를 나타낸다. 이 표에서 자원 이벤트 옵션에 대한 시뮬레이션 결과들은 이벤트 그룹 의존성이 존재하므로 역시 사용할 수 없는 것들이다.

매크로-이벤트-라운드-로빈 시뮬레이터가 제일 먼저 시도되었다. 이 시뮬레이션 모드에서는 먼저 I/O와 통신의 중첩 여부를 확인한다. 대상 플랫폼에서 I/O는, 그 플랫폼이 이전 절에서 기술한 실험에서 사용되었던 SP2의 스케일이 커진 버전이라고 가정하였기에 통신과 중첩되지 않는다. 결과적으로, 매크로-이벤트-라운드-로빈 시뮬레이터 실행 결과는 해당 옵션이 충분히 정확한지 알아 보기 위하여 확인된다. 요구되는 오류가 20%이기 때문에, 현재 시뮬레이션 오류 범위가 58%인 상황에서는 이 시험은 실패한다.

다음으로, 매크로-이벤트 큐 기반 시뮬레이터가 실행되었다. 이 옵션에 의하여 추정된 상한 및 하한 실행시간은 해당 오류 범위가 목표 오류 내에 존재하는지 여부를 판단하기 위하여 확인되었다. 오류 범위가 6%로 목표 오류보다 작기 때문에 매크로-이벤트 큐 기반 시뮬레이션의 결과는 최

〈표 4〉 12TB 자료를 가진 Virtual Scope에 대한 시뮬레이션 결과

시뮬레이션 옵션	예측 시간 (초)	시간 범위					
		원래			결합된		
		하한	상한	오류 (%)	하한	상한	오류 (%)
Resource Event	*66,126.2*	*66,126.2*	*75,293.1*	*13.9*	*66,126.2*	*75,293.1*	*13.9*
Macro-Event RR	66,126.2	66,126.2	75,293.1	13.9	66,126.2	75,293.1	13.9
Macro-Event Queue	66,126.2	66,126.2	75,293.1	13.9	66,126.2	75,293.1	13.9
Micro-Event Queue	66,126.2	66,126.2	75,293.1	13.9	66,126.2	75,293.1	13.9

〈표 3〉 1,200개의 노드 상에 12TB 자료를 가진 Titan

	시뮬레이션 시간	예측 시간에 대한 스피드업
Resource Event	32.8	72.1
Macro-Event RR	177.3	29.2
Macro-Event Queue	*275.8*	*9.6*
Micro-Event Queue	1,343.8	2.0

〈표 5〉 12TB 자료를 가진 Virtual Microscope

	시뮬레이션 시간	예측 시간에 대한 스피드업
Resource Event	*70.5*	*917.1*
Macro-Event RR	175.0	377.9
Macro-Event Queue	260.7	253.6
Micro-Event Queue	325.2	203.3

종 예측된 시간으로서 사용된다(표에서 해당 행을 기울인 글자로 표시한다). 상대적인 시뮬레이션 오류 95.1%는, 이 옵션의 하한이 대단위 매크로-이벤트-라운드-로빈 옵션보다 매우 작기 때문에 매크로-이벤트-라운드-로빈 옵션의 그것보다 크다. 그러나, 절대적인 시뮬레이션 오류인 이 옵션의 하한과 상한의 차이는 매크로-이벤트-라운드-로빈 옵션의 그것보다 작다. 여기서, 이전에 시도된 기술을 사용하여 얻어진 오류 한계를 바탕으로 판단하면 마이크로-이벤트 큐 기반 시뮬레이터를 실행시킬 필요가 없지만 참고로 그 시뮬레이터의 결과도 보여준다.

<표 3>은 초 단위의 시뮬레이션 소요 시간과 예측되는 시간에 대한 스피드업을 보여준다. 스피드업 요인은 실제 수행하는 것보다 얼마나 빨리 시뮬레이션할 수 있는가를 나타낸다. 이 예에서는, <표 3>에서 보여주는 바와 같이 최종 시뮬레이션 옵션 실행에 5분 미만의 시간이 소요됐다. 최선의 옵션을 선택하는 방법론을 적용함으로써, 20분 이상의 시간이 소요되지만 유사한 시뮬레이션 결과를 내는 마이크로-이벤트 큐 옵션을 사용하는 것을 피할 수 있다.

<표 4>는, 12개의 노드 상에 1.2TB(노드당 100GB)의 자료를 가진 Virtual Microscope에 대하여 각 시뮬레이션 옵션의 예측 시간, 오류의 상 및 하한계, 그리고 오류 범위를 나타낸다. 모든 4개의 옵션에 대하여 예측된 실행 시간은 18시간보다 약간 큰 정도(66126초)이다. 그 시뮬레이션의 예측 오류 한계는 14%이다.

이 예에서, 네트워크 경합/정체 또는 랜덤 I/O 특성은 존재하지 않는다. 그리고, 이벤트 그룹 의존성도 존재하지 않는다. 그 결과, 자원 이벤트 시뮬레이터가 실행되었다. 목표 오류보다 작은 해당 옵션의 오류 범위는 14%이다. 따라서, 이 옵션은 최선의 방법으로서 선택된다. 다른 대단위 옵션은 이 옵션과 동일한 시뮬레이션 결과를 보여주고 따라서 상대적인 장점이 없다. 하지만, <표 4>에서는 선택된 방법 이외의 다른 대단위 시뮬레이터들의 결과도 참고로 보여준다. 자원 이벤트 기술로 얻어진 오류 한계를 사용하면 충분하므로 그 시뮬레이터들을 실행시킬 필요는 없다.

<표 5>는 초 단위의 시뮬레이션 시간과 예측되는 시간에 대한 스피드업을 보여준다. 자원 이벤트 옵션 대한 스피드업 요소는 917이다. 이것은 응용을 실행시키는 것보다 900배 빠르게 프로그램의 실행시간을 예측할 수 있다는 것을 의미한다. 그리고, 제안하는 방법론은 5분 이상의 시간이 소요되는 다른 대단위 옵션들 대신에 약 1분만이 걸리는 자원 이벤트 옵션을 선택하게 해 준다. 이 예에서의 결과는 제안하는 시뮬레이터 선택 방법론을 사용하는 장점을 보여준다. 가장 간단한 시뮬레이션 옵션이 충분히 정확할 때 이 방법론은 이 사실을 파악하여 더 복잡한 것들을 실행하지 않게 해준다.

7. 관련 연구

많은 시뮬레이션 연구에서 시뮬레이션 속도와 세밀함의 트레이드오프에 대하여 언급했다. 전체의 컴퓨터 시스템 시뮬레이터인 SimOS[19]는 세개의 전환 가능한 시뮬레이션 모드들을 제공한다. 위치선정, 간략한 특성화, 그리고 정확한 모드가 그것들이다. 이것은 동적으로 모드를 선택할 수 있게 함으로써 실행 중 관심있는 부분만을 세밀하게 시뮬레이션할 수 있게 해준다. 그러나, 이것은 오류 요구사항에 대하여 최선의 성능을 얻을 수 있도록 시뮬레이션 모드를 선택하게 하는 구체적인 방법론을 제공하지는 않는다.

다단계의 시뮬레이션을 허용하는 서로 다른 시뮬레이션 시스템들은 이미 개발되었다[6, 9, 13]. 하지만, 이들 중 어느 것도 목표 오류 조건을 만족시키는 가장 효율적인 수준의 시뮬레이션을 선택하게 해 주지 않는다. 그러나, 이러한 시스템들은 제안하는 방법론이 포괄하지 않는 캐싱이나 버퍼링과 같은 시스템 활동의 영향을 명시적으로 고려한다.

Wisconsin Wind Tunnel[18]을 사용한 병렬 시뮬레이션 연구[3]는 6개의 서로 다른 네트워크 시뮬레이션 모델들에 대하여 다양한 성능 트레이드오프를 보여준다. 그러나, 이연

구에서는 최선의 모델을 선택할 수 있는 방법론을 제공하지 않는다. 다른 한편으로, 직접적인 실행이나 병렬 시뮬레이션에 의한 효율적이고도 정확한 시뮬레이터를 제공하는데 초점을 맞춘 연구들[2, 18, 24]도 있다.

POEMS[7]는 서로 다른 대상 구성 요소들이 여러가지 패러다임들에 의하여 다양한 상세함의 수준들로 모델화되게 해 주는 통합된 End-to-End 성능 모델화 시스템이다. 이것은 시뮬레이션 속도와 정확도 간에 선택할 수 있게 해주는 다양한 옵션들을 제공한다는 점에서 제안하는 시뮬레이션 시스템과 비슷하다. 그러나, 이것은 각 옵션에 대한 오류 한계를 추정할 수 없고, 이는 목표 오류를 고려하는 옵션 선택 방법론을 제공할 수 없게 한다. COMPASS[2]는 POEMS 프로젝트에서 개발된, 다양한 시뮬레이션을 위하여 쓰이는 직접적인 실행 구동형 병렬 시뮬레이터이다.

시간 워프 시뮬레이션[11, 16, 21]은 메시지를 통하여 LP(Logical Process)들이 통신하는 동안 타임스탬프 순서에 따라서 이벤트들에 대한 LP의 시계 값을 비동기적으로 증가시킨다. 이 시뮬레이션에서는 순서가 어긋난 메시지들에 대하여 롤백을 제공함으로써 올바른 타임스탬프 순서로 메시지에 대한 이벤트들을 처리하도록 시뮬레이션을 재구성한다. 제안하는 시스템에서 이벤트들은 이벤트 통합으로 인하여 순서가 어긋나게 처리될 수 있다. 하지만, 비용이 큰 롤백을 수행함으로써 이 문제를 해결하려고 하기보다는 통합으로부터 야기되는 오류 범위를 수량화하였다. 이러한 접근 방법과는 다르게 롤백의 비용을 줄이기 위하여 다양한 기술들이 개발되어 왔다. LP 상태, GVT(Global Virtual Time) 추정, 그리고 화석(이전 이벤트) 수집 비용을 절약하기 위한 효율적인 체크포인팅 구조는 시뮬레이션 성능을 증가시키고 요구되는 메모리 양을 줄이기 위하여 고안되었다[15, 17]. 병렬 이산 이벤트 시뮬레이션의 언어와 라이브러리에 대한 연구 결과는 [12]에서 찾을 수 있다.

8. 결론 및 미래 연구방향

본 논문에서는 I/O 집약적인 응용을 대상으로 하는 이벤트에 기반을 둔 시뮬레이션 옵션들의 집합에 대하여 기술하였고 이벤트 통합 정도에 따라서 시간과 정확도 간에 트레이드오프가 있다는 것을 보였다. 주어진 옵션들에 대한 오류 한계 조건을 만족하면서 가장 효율적인 시뮬레이션 옵션을 선택하게 하는 방법론을 제시하였고 두개의 현존하는 자료 집약적인 응용에 대하여 그 효과를 입증하였다.

향후 연구 과제로는 동적으로 옵션 선택 방법론을 적용하여 최선의 옵션을 실행시간 동안 선택하게 하는 연구가 필요하다. 이러한 선택을 위하여, 입력 이벤트들은 온라인으로 생성되어야만 한다. 또한, 옵션 변경 시점에서 시뮬레이션 상태 정보를 서로 다른 시뮬레이터들 간에 정확하게 전달하는 메커니즘이 구현되어야 한다. 최선의 옵션이 선택되면 그것은 실행의 완료 시점까지 사용될 수 있고, 또는 시뮬레이션 방법은 대상 응용과 해당 실행의 특성의 변화에 따라 주기적으로 교체될 수 있다.

참 고 문 헌

[1] R. Agrawal and J. Shafer, "Parallel Mining of Association Rules," IEEE Trans. on Knowl. Data Eng. Vol.8. No.6, pp.962-969, 1996.

[2] R. Bagrodia, E. Deelman, S. Docy, and T. Phan, "Performance Prediction of Large Parallel Applications Using Parallel Simulations," ACM PPoPP, pp.151-161, Atlanta, GA, May 1999.

[3] D. C. Burger and D. A. Wood, "Accuracy vs. Performance in Parallel Simulation of Interconnection Network," 9th ACM/IEEE IPPS, pp.22-31, Santa Barbara, CA, Apr., 1995.

[4] C. Chang, et al., "Titan: A High-Performance Remote-Sensing Database," 13th ICDE, pp.375-384, UK, Apr., 1997.

[5] R. G. Covington, et al., "The Efficient Simulation of Parallel Computer Systems," International Journal in Comp. Simul. Vol.1, pp.31-58, 1991.

[6] H. Davis, S. R. Goldschmidt, and J. Hennessy, "Multiprocessor Simulation and Tracing Using Tango," 1991 ICPP, pp.99-107, St. Charles, IL, Aug., 1991.

[7] E. Deelman, et al., "POEMS: End-to-end Performance Design of Large Parallel Adaptive Computational System," International Workshop on Software and Performance, pp.18-30, Santa Fe, NM, Oct., 1998.

[8] R. Ferreira, et al., "The Virtual Microscope," 1997 AMIA Ann. Fall Symp., pp.449-453, Nashville, TN, Oct., 1997.

[9] R. S. Francis and I. D. Mathieson, "Compiler-Integrated Multiprocessor Simulation," International Journal in Comp. Simul. Vol.1 No.2, pp.169-188, 1991.

[10] G. A. Gibson and R. V. Meter, "Network Attached Storage Architecture," CACM, Vol.43 No.11, pp.37-45, Nov., 2000.

[11] D. Jefferson, "Virtual Time," ACM TPLS Vol.7 No.3, pp.405-425, 1985.

[12] Y. H. Low, et al., "Survey of Language and Runtime Libraries for Parallel Discrete-Event Simulation," The Journal of the Society for Comp. Simul., Vol.72 No.3, pp.170-186, 1999.

[13] P. S. Magnusson, et al., "SimICS/sun4m: A Virtual Workstation," Usenix 1998 Ann. Technical Conference, pp.119-130, New Orleans, LA, June, 15-18, 1998.

[14] G. Papadopolous, 'The Future of Computing,' Unpublished Talk at NOW Workshop, 1997.

[15] F. Quaglia and A. Santoro, "Modeling and Optimization of Non-Blocking Checkpointing for Optimistic Simulation on

Myrinet Clusters," JPDC,Vol.65 No.6, pp.667-677, June, 2005.

[16] R. Radhakrishnan, N. Abu-Ghazaleh, M. Chetlur, and P. A. Wilsey, "On-line Configuration of a Time Warp Parallel Discrete Event Simulator," 1998 ICPP, pp.28-35, Minneapolis, MN, Aug., 1998.

[17] P. L. Reiher, "Parallel Simulation Using the Time Warp Operating System," 1990 Winter Simul. Conference, pp.38-45, New Orleans, LA, Dec., 1990.

[18] S. K. Reinhardt, J. R. Larus, and D. A. Wood, "The Wisconsin Wind Tunnel: Virtual Prototyping of Parallel Computers," ACM SIGMETIRCS, pp.46-60, Santa Clara, CA, May, 1993.

[19] M. Rosenblum, E. Bugnion, S. Devine, and S. Herrod, "Using the SimOS Machine Simulator to Study Complex Computer Systems," ACM Trans. Model. Comp. Simul. Vol.7 No.1, pp.78-103, 1997.

[20] E. Rosti, G. Serazzi, E. Smirni, and M. S. Squillante, "Models of Parallel Applications with Large Computation and I/O Requirements," IEEE Trans. on Soft. Eng., Vol.28 No.3, pp.286-307, Mar., 2003.

[21] J. S. Steinman, "Incremental state saving in SPEEDES using C++," 1993 Winter Simul. Conference, pp.687 - 696, Los Angeles, CA, Dec. 13-16, 1993.

[22] M. Uysal, A. Acharya, R. Bennett, and J. Saltz, "A Customizable Simulator for Workstation Networks," 11th IPPS, pp.249-254, Geneva, Switzerland, Apr., 1997.

[23] M. Uysal, T. M. Kurc, A. Sussman, and J. Saltz, "A Performance Prediction Framework for Data Intensive Applications on Large Scale Parallel Machines," 4th Workshop on Language, Compiler and Run-Time Systems for Scalable Computers, pp.243 -258, Pittsburgh, PA, May, 1998.

[24] T. L. Wilmarth, G. Zheng, E. J. Bohm, Y. Mehta, N. Choudhury, P. Jagadishprasad, and L. V. Kale, "Performance Prediction Using Simulation of Large-Scale Interconnection Networks in POSE," 19th Workshop on Principles of Advanced and Distributed Simul.," pp.109-118, Monterey, CA, June, 2005.

엄 현 상

e-mail : hseom@cse.snu.ac.kr

1992년 서울대학교 계산통계학과(학사).

1996년 미국 University of Maryland 컴퓨터과학과(석사)

1997년 미국 Sun Microsystems, Inc. Data Engineering Group, 인턴

2003년 미국 University of Maryland 컴퓨터과학과(박사)

2003년~2005년 삼성전자 정보통신총괄 책임연구원

2005년~현재 서울대학교 컴퓨터공학부 교수

관심분야 : 컴퓨터 시스템/네트워크/응용 소프트웨어 성능공학, 모바일 응용/미들웨어(보안 포함), 임베디드 소프트웨어