실시간 분산처리 제어기를 위한 객체지향형 시뮬레이터 구조

박재현

인하대학교 자동화공학과

1. 서론

컴퓨터 기술의 비약적인 발전으로 대부분의 아날로그 제어기들은 디지털 제어기로 대체되고 있으며, 전기적인 회로로 구성되었던 제어 알고리즘은 제어기상에서 동작하는 소프트웨어로 바뀌었다. 제어해야 할 시스템의 범위가 커지고 복잡하게 발전함으로 인해 단일 기계에 의한 소규모 제어기로부터 복수의 제어기가 네트웍으로 연결되어있는 분산처리 제어기로 발전되어가고 있다. 이러한 분산처리 제어기의 성능을 검증하기 위해서는 기존의 단일 제어기용 시뮬레이터에의한 모의실험은 한계를 가지며 분산처리 환경을 모델링하고 효과적으로 시뮬레이션할 수 있는 새로운 구조의 시뮬레이터가 필수적이다. 현재 주로 사용되고 있는 시뮬레이션 패키지들은 제어알고리즘의 최적의 해를 구하거나 제어모델의 응답특성을 해석하는데 주 목적이 있기 때문에 실시간 시뮬레이션에는 큰 비중을 두지 않았다. 하지만 실시간으로 동작하는 플랜트와 연계하여 제어기의 구조를 모델링 하거나, 제어기의 실시간 동작 특성을 확인하기 위해서는 시간 제약조건을 만족하는 분산처리 시뮬레이터 및 제어기의 사용을 필요로 한다[1][2].

하나의 대형 시스템 모델을 모듈별로 나누고 각각을 분산된 노드로 부하를 분산시킨다면 실시간 제어 및 시뮬레이션이 가능해진다[3][4][5]. Carnegie Mellon 대학의 Stewart 등은 이를 해결하기위하여 포트객체(port-based object)를 이용한 실시간 처리 소프트웨어 구조를 제안하였다[6]. 하지만 각각의 태스크간에 데이터를 전달하기 위해 전역 상태변수 테이블(global state variable table)을 사용하기 때문에 프로세서간에 비동기적으로 발생하는 메시지의 전달에는 어려운 점이 있으며, 하드웨어와 인터페이스 하기 위해서 포트객체를 이용하는 것은 다양한 종류의 하드웨어 인터페이스 상에서 동작하는 포트객체를 필요로 하여 소프트웨어의 확장성을 떨어뜨린다. Hoger는 분산 시뮬레이션 (conservative distributed simulation)과 공동 시뮬레이션 (concurrent simulation)을 통합한 시뮬레이터 모델을 제안하여 시뮬레이션 모델의 확장성을 높였으나, 이 구조에서는 각 노드의 부하가 고르게 분포하지 않을 때의 시뮬레이터 성능저하를 보여주고 있다[7]. National Institute of Standards and Technology (NIST)에서는 실시간 제어 시스템(real-time control system; RCS)의 참조모델 구조에 관해 연구하였다. RCS는 프로그래머가 실시간 시스템을 프로그래밍 하기 위해 필요로 하는 기본적인 API(application programming interface)를 제공하는 것이다.

가상머신은 컴파일러나 인터프리터 개발에서 이식성을 높이기 위한 수단으로 흔히 사용되어온 기술이다. 이러한 기술을 채택한 예로는 자바에 앞서 얼렝(Erlang)이 있다. 얼렝은 전화교환기 등과 같은 통신 시스템 개발을 위한 프로그래밍 시스템으로 효율성을 적절히 희생하여 이식성을 추구한 대표적 사례다. 자바 기술도 얼랭의 경우와 거의 유사하다. 얼랭이 논리와 함수형 언어를 위한 시스템으로 개발되었다면 자바는 객체지향형 언어와 시스템으로 개발되었으며, 얼랭이 통신 시스템을 위한 기술이라면 자바는 고객주문형 가전제품과 인터넷, 웹 등을 위한 시스템으로 개발되었다는 것 등을 차이점으로 들 수 있다. 자바(Java)는 가상머신을 구현한 임의의 시스템에서 실행될 수 있으므로 특정 하드웨어나 운영체제에 구애받지 않는 공통의 환경을 제공한다[8].

자바 가상머신은 자바에게 높은 이식성과 객체지향 개념을 제공한다. 하지만 제어 모델 해석에 주로 사용되는 메트릭스 연산을 하기 위해서는 상당히 많은 자바 바이트코드(Java byte-code)가 필요하며, 특정한 시스템에 설치된 가상머신에서 emulation되어 수행되는 자바 바이트코드는 수행속도가 현저히 느려지게 된다. 따라서 새로 개발된 가상머신에서는 자바 가상머신을 시뮬레이터 및 제어기에 수정 없이 도입한다면 속도가 느린 문제를 보완하기 위해서 사용이 빈번한 기능이나 제어 알고리즘은 가상머신 내부에 내장함수 형태로 제공된다.

본 글에서는 실시간 분산처리 제어기를 위한 시뮬레이터의 구조를 제시한다. 분산처리 환경에서 각각의 노드 간에 비주기적으로 발생하는 메시지와 주기적으로 발생하는 데이터의 교환을 위한 통신 구조와 가상머신

를 이용한 객체지향형 개방형 시뮬레이터 구조를 제시하고자 한다. 그리고 제안된 구조의 소프트웨어를 이용하여 PID 제어기를 설계하는 예제를 보여준다.

2. 객체지향 모델링 및 설계

사용자는 GUI(graphic user interface)환경에서 아이콘으로 표현된 각종 객체들을 끌어 놓은 후 객체들을 선으로 연결하여 모델을 편집하게 된다. 이러한 식의 시각화된 모델은 사용자가 쉽게 이해하는 것이 가능하며 객체별로 모듈화 된 소스코드는 모델의 수정, 추가, 유지보수가 용이하다[9]. 디자인된 모델이 시뮬레이션 되기 위해서는 모델은 각 객체와 연결구조에 해당하는 클래스와 클래스간의 메시지 전달구조를 가지는 소스코드를 만들어낸 후 이를 가상머신(virtual machine)에서 수행되는 기계어 코드로 번역한다. 가상머신 인터페이스(virtual machine interface)는 다른 노드의 가상머신과 통신하기 위해서 메시지 큐(message queue)와 공유변수테이블(shared variable table)을 가진다.

2.1 포트 객체

시각화 된 모델을 구성하는 포트객체(port-based object)는 알고리즘 구성의 최소 단위가 되며 포트객체가 서로 연결되어 하나의 시뮬레이션 모델을 형성한다. 객체의 내부는 외부로부터 숨겨지고 다른 객체와 통신하기 위해 다수의 입출력 포트를 사용한다. 태스크간의 통신을 위해서 포트객체의 입출력 포트는 링크객체(link object)를 거쳐서 다른 태스크를 구성하는 포트 객체와 통신할 수 있다. 링크객체를 통해 한 개의 출력 포트가 여러 개의 입력 포트에 연결될 수 있으며, 두 개 이상의 출력포트는 직접 연결될 수 없고, 서로 연결시킬 때는 joint connector를 이용하여야 한다.

그림 1은 다수의 포트객체가 모여서 하나의 컴바인드 포트객체(combined port-based object)로 구성되는 것을 보여주고 있다. 컴바인드 포트객체를 이용하면 모델을 계층구조로 모델링할 수 있다. 컴바인드 포트객체 의 입출력 포트는 내부를 구성하는 포트객체의 입출력 포트를 외부의 다른 포트 객체들과 연결시켜 준다. 컴바인드 포트객체는 다음과 같은 이점을 제공한다.

① 복잡한 알고리즘을 표현하는 객체들을 작고 단순한 하나의 객체로 묶는 것이 가능하다.

② 객체들은 객체지향 언어의 캡슐화와 상속성을 이용하여 쉽게 객체지향 소스코드로 바뀐다.

2.2 가상머신 인터페이스

가상머신이 다른 노드의 가상머신과 통신하기 위해서

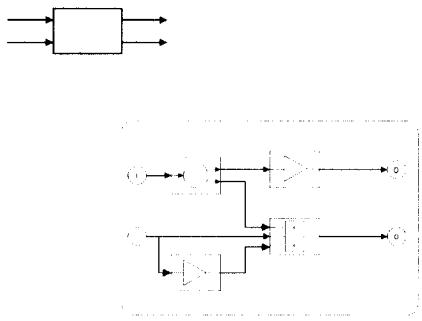

그림 1. 컴바인드 포트 객체

와 같은 가상머신 인터페이스(virtual machine interface)를 사용한다. 가상머신 인터페이스는 노드에서 타스크의 작업을 결정하기 위해서 메시지와 데이터를 주고받는다. 분산환경의 시뮬레이터에서 노드간 데이터와 메시지교환 구조는 공동시뮬레이션에서 프로세서간에 공유되어 사용되는 공유메모리와, 분산시뮬레이션에서의 메시지 큐 구조다. 노드에서 발생하는 연속적인 데이터의 전달에는 공유메모리가 사용되고, 비주기적으로 발생하는 메시지의 처리에는 우선 순위를 가지는 메시지 큐가 사용되었다. 위와 같은 구조는 시뮬레이션 모델의 확장이 용이하고, 각각의 노드에서 수행되는 작업을 다이나믹하게 스케쥴링 가능하므로 한 노드에 작업이 집중되는 병목현상의 발생을 줄일 수 있다[10][11][12].

2.3 실시간 메시지 스케쥴링

분산처리 환경에서 각각의 노드는 구성요소의 특성에 따라 처리되는 프로세서의 주기가 달라진다. 시뮬레이터는 이러한 요구를 만족시키기 위해 각 노드의 수행 주기에 따라 메시지를 발생시키는 클럭을 가지고 있다. 이러한 구조는 각 노드에 일정한 기저부하를 걸어주므로 시뮬레이션이 무리 없이 수행되기 위해서는 일정한 주기로 수행되는 태스크의 평균 수행시간이 메시지의 발생주기보다 짧아야 한다.

주기적인 메시지는 주로 입출력 디바이스로부터 데이터 수집, 모니터링 시스템으로 데이터 출력, 제어 알고리즘의 반복 수행 등의 작업을 통하여 처리되는 연속성을 가지는 데이터 교환을 목적으로 가상머신의 인터페이스를 사용한다. 인터페이스의 지역 공유메모리(local shared memory)와 전역 공유메모리(global shared memory)는 공유메모리(shared memory)와 같은 역할을 하도록 Ethernet, field-bus, 비동기 통신과 같은 네트워 레이어를 추상화시킨 상위 레이어고, 가상머신 인터페이스에 의해서 주기적으로 데이터의 동기를 유지시킨다. 모든 노드는 지역 공유메모리를 가지고 태스크가 수행될 때 필요

한 전역변수들을 저장하고 있다. 마스터 노드는 슬레이브 노드와 달리 전역 공유메모리를 가진다. 전역 공유메모리는 여러 개의 노드로부터 접근되어지지만 하나의 노드만 엑세스 할 수 있기 때문에 테이블의 lock/unlock 하는 과정이 필요하며, 지역 공유메모리와 전역 공유메모리는 네트웍으로 연결되기 때문에 네트웍의 성능에 따라 접근시간을 많이 소비하게 된다. 그러므로 슬레이브 노드에서 수행되는 태스크는 직접적으로 전역 공유메모리의 변수를 접근하지 못하고 지역 공유메모리를 통해서 한 주기의 마지막 사이클에서 한번 접근 되므로 최소한의 오버헤드(overhead)로 수행되는 것이 가능하다.

에러 메시지나 경고 메시지와 같이 비주기적으로 발생하는 이벤트(event)는 타임 스탬프를 가지는 메시지 형식으로 노드에 전달되어 메시지 큐(message queue)에 삽입된다. 그리고 각각의 슬레이브 노드를 동기 시키기 위해서 null 이벤트를 사용한다. 위와 같은 비주기적 메시지들 중에는 중요도가 높은 것이 많으므로 대부분 가장 높은 우선 순위를 가지고 메시지 큐에서 스케쥴링 된다.

메시지는 command와 parameter에 따라 태스크에서 수행되는 작업의 종류가 결정된다. 수행이 끝나고 나면, flag의 상태에 따라 응답 메시지를 송신측으로 되돌려 주기도 하며, 마스터/슬레이브 노드의 메시지 전달 구조에서 한 주기의 요구/응답 사이클을 완료한다.

3. 개방형 가상 머신

시뮬레이터 및 제어기의 가상머신(virtual machine)은 컴파일 된 기계어 코드를 수행시키는 추상화 된 컴퓨터이다. '가상'이라는 말이 의미하듯 가상머신은 실제의 하드웨어 플랫폼과 운영체제 상에서 구현 된 소프트웨어를 말한다. 그러므로 컴파일된 기계어 코드가 수행되기 위해서는 특정한 플랫폼 상에 가상머신이 구현되어 있어야 한다.

기계어 코드 사이에서 추상화된 레이어를 제공한다. 그래서 소스코드는 하드웨어 플랫폼이나 운영체제에 무관하게 가상머신에서만 동작하도록 컴파일 된다. 그러므로 가상머신용으로 개발된 프로그램은 하드웨어 플랫폼이나 운영체제가 변하더라도 소스코드를 수정하거나 다시 컴파일 해야 하는 부담을 줄일 수 있다. 즉 가상머신은 시뮬레이터 및 제어기가 이식성을 가지도록 하는 중심적인 역할을 한다. 컴파일 된 기계어 코드는 수행될 노드로 Ethernet, fieldbus, 혹은 비동기 통신을 통해 다운로드 된다.

그림 3은 가상머신의 내부 구조를 보여주고 있다. 가상머신은 크게 CPU, 메모리(memory), 내장 코드(internal code), 입출력부(input/output)로 구성되어 있다. CPU는 레지스터(register), 명령어 패치(instruction fetch), 그리고 수행부(executor)로 구성되어있다. 메모리에는 코드(code), 데이터(data), 스택(stack), 힙(heap) 영역으로 구분되어 있다. 내장 코드는 연산자(operator)와 내부함수(internal function)로 구분된다. 입출력부는 가상머신이 외부와 연결되는 통로 역할을 하고, 입출력 레지스터(I/O register)와 지역 공유메모리를 가진다.

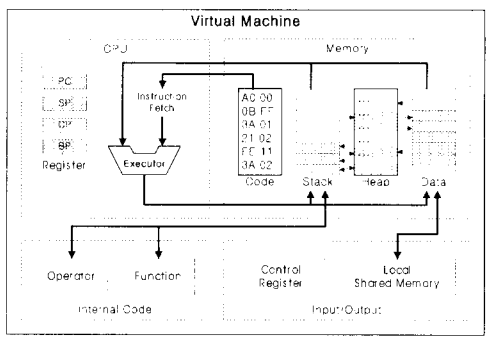

그림 3. 가상머신 내부구조

제어기 및 시뮬레이터는 제어 알고리즘을 수행시키기 위해서 내부에 가상머신을 포함하고 있다. 가상머신은 제어기 및 시뮬레이터와 직접 연결되지 않고 가상머신 인터페이스를 거쳐서 연결된다.

가상머신 인터페이스는 내부에 메시지 큐와 공유메모리를 가지고 있어서 가상머신이 분산환경에서 실시간으로 메시지와 데이터를 주고받으며 동작하도록 한다.

그림 4는 코드를 작성하고 컴파일, 디버깅 하는 통합환경의 MDI 윈도우다. 제어기 및 시뮬레이터와 가상머신은 C++ 언어로 개발되었으며, 현재 소스코드는 Intel사의 386, 486, Pentium CPU를 사용하는 PC 기반의 플랫폼과 Windows 95/NT 운영체제에서 동작하도록 컴파일 되었다. 제어기 및 시뮬레이터는 15,000 라인 분량의 C++언어로 개발되었다. 가상머신 인터페이스는 10,000라인 분량의 C++언어로 개발되었으며, 라이브러리 형태로

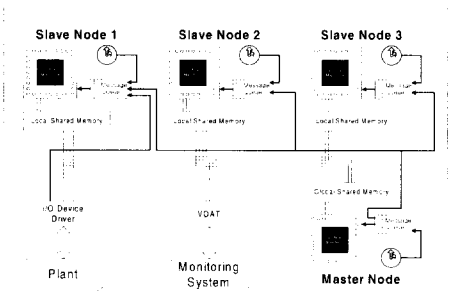

그림 2. 가상머신 인터페이스

컴파일 되어서 제어기 및 시뮬레이터 프로젝트에서 링크된다. 가상머신은 약 30,000 라인 정도의 C++언어로 구현되었으며 Windows 동적 링크 라이브러리(Dynamic Link Library; DLL)로 구현되어 있어서 후에 기능을 확

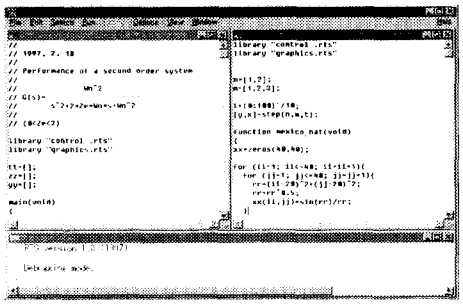

그림 4. 시뮬레이션 윈도우

장하는 것이 용이하다. 가상머신 중 20,000 라인정도가 내장함수를 위한 분량이다. 또한 시뮬레이터와 제어기가 다기종으로 구성된 컴퓨팅 환경에서 동작하기 위해서는 가상머신이 다양한 하드웨어 플랫폼과 운영체제에 따라 개발 되어야 한다. 실시간 분산처리 시뮬레이터가 플랜트 시뮬레이션에 적용되는 구조는 온라인 시뮬레이션(on-line simulation)과 오프라인 시뮬레이션(off-line simula-tion)의 두 가지로 나누어진다. 온라인 시뮬레이션에서는 제어기에 실제의 플랜트가 연결되어 시뮬레이션이 이루어진다. 제어기의 제어 알고리즘(task)은 정해진 시간 내에 수행을 마쳐서 다음 제어 값을 플랜트에 넘겨주어야 하므로 제어 알고리즘의 수행시간이 실시간 제약조건을 만족하는지를 시뮬레이션으로 확인할 수 있다. 오프라인 시뮬레이션은 모델링된 플랜트가 분산시스템의 노드로 대체되어 플랜트의 역할을 한다. 하지만 제어기는 플랜트와 통신하기 위해서 공유메모리를 통해 변수를 입출력 할 뿐 플랜트가 실제이건 모델링된 소프트웨어 알고리즘이건 고려할 바가 아니다. 그러므로 오프라인으로 시뮬레이션한 모델을 가상의 공정만 실제로 대체시키고 제어기 부분은 수정하지 않고도 온라인 제어에 사용할 수 있다.

4. 시뮬레이션 예제

본 연구에서 설계되고 구현된 시뮬레이터의 효용성을 보이기 위하여 분산환경에서 실시간으로 운용되는 시뮬레이터를 이용하여 PID 제어기를 설계해 본다. 그림 5는 시뮬레이션 대상 모델이며 그림 6은 분산환경에서의 시뮬레이션 결과이다.

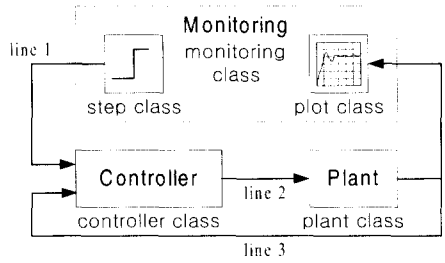

그림 5. 시뮬레이션 모델

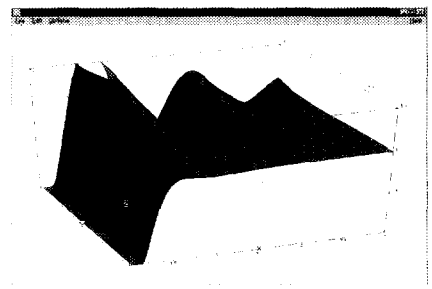

그림 6. 시뮬레이션 결과

5. 결론

본 글에서는 실시간 분산처리 제어기의 동작을 검증할 수 있는 분산처리 시뮬레이터의 내부구조를 설명하고 이를 구성하는 가상머신의 구조와 실시간 분산처리 환경에 대해서 설명하였다. 본 시뮬레이터의 장점은 다음과 같다. 첫째, 실시간 분산처리 시뮬레이터 및 제어기에서 수행되는 제어 알고리즘의 개발에 객체지향 프로그래밍 방법을 사용하여 모델을 모듈별로 나누어 프로그램을 작성하여 모듈간의 결합도를 낮추고 모듈의 재사용성을 높일 수 있다. 둘째, 제어기 구조에 가상머신을 도입하여 다양한 컴퓨터로 구성된 분산환경에서의 이식성을 제공한다. 그리고 사용이 빈번한 기능이나 제어 알고리즘은 가상머신 내부에 내부함수 형태로 제공된다. 셋째, 분산처리 시뮬레이터의 제어 알고리즘은 두 시스템 간의 공통된 인터페이스를 제공하는 메시지 큐와 공유메모리를 통해서 제어대상과 연결되므로, 제어대상을 실제의 플랜트가 아닌 가상의 플랜트 모델을 설정하여 제어 알고리즘의 성능을 분석할 수 있다. 위에서 제시하는 실시간 분산처리 시뮬레이터 및 제어기 구조는 다기종으로 구성된 지역 네트워 환경을 수정 없이 이용가능하고 제어 알고리즘의 개발에는 객체지향 방법을 사용하므로, 시스템 개발기간을 단축할 수 있으며 개발된 시스템의 유지보수 비용을 줄일 수 있다.

참고문헌

[1] S. Bennetntrol, Prentice Hall, Maylands Avenue Hemel Hempstead, 1994.

[2] Miquel Angel Piera, Cesar de Prada, "Modeling Tools in the Continuous Simulation Field", vol. 66, no. 3, pp. 179-189, March, 1996.

[3] K. H. Kim, "Toward new-generation object-oriented real-time software and system engineering", SERI Journal, vol. 1, no. 1, pp. 1-23, 1997.

[4] Michel de Champlain, "A Small and Expressive Objectbased Real-time Programming Language", SIGPLAN Notices, vol. 25, no. 5, pp. 124-134, 1990.

[5] C. G. Cassandras, Discrete Event Systems:modeling and performance analysis. Richard D. Irwin, Inc., and Aksen Associates, Inc., 1993.

[6] D. B. Stewart, R. A. Volpe, and P. K. Khosla, "Design of dynamically reconfigurable real-time software using port-based objects", Technical report, Car- negie Mellon University, Pittsburgh, PA 15213, July.1993.

[7] H. R. Hoger and D. W. Jones, "Integrating concurrent and conservative distributed discrete-event simulators", Simulation, vol. 66, pp. 303-314, Nov 1996.

[8] James Gosling and Henry McGilton, "The Java Language Environment A White Paper", Sun Microsystems Computer Company, October, 1995.

[9] D.R.Gentner and J. Grudin, "Design models for computer human interfaces", IEEE Computer, vol. 29, no. 6, pp. 28-35, June 1996.

[10] A. Pullafito, S. Riccobene and M. Scarra, "Evaluation of Performability Parameters in Client-Server Environments", The Computer Journal, vol. 39, no. 8, pp. 647-662, 1996.

[11] Krithi Ramamritham, John A. Stankovic, "Efficient Scheduling Algorithms for Real-time Multiprocessor Systems", vol. 1, no. 2, pp. 184-194, April, 1990.

[12] Helen D. Karatza, "Simulation Study of Task Scheduling and Resequencing in a Multiprocessing System", Simulation, vol. 68, no. 4, pp. 241-247, April, 1997.

♣ ICASE home page 안내 ♣

1. 주 소 : http://icat.snu.ac.kr/icase/
2. 구 성 : 본 학회의 논문 작성요령 및 사진을 포함한 회원 D/B를 구축하였습니다.
3. home page에 방문하여 신상명세에 오류가 발견될 시에는 아래의 사무국 전화번호를 참고하시어 정정하여 주시기 바랍니다.
4. 또한 회원 D/B에 사진이 포함되어 있지 않은 경우는 사무국으로 반명함판 사진을 보내주시면 입력하도록 하겠습니다.

♥ 기타의 자세한 사항은 본 학회 사무국으로 문의하여 주시기 바랍니다

제어·자동화·시스템공학회 사무국.

주 소 : (우)135-703 서울시 강남구 역삼동 635-4 과학기술회관 본관 406호

Tel. 508-5801, 508-5830, Fax. 555-4746, E-mail. icase@chollian.dacom.co.kr