XML 기반 그리드 컴퓨팅 환경을 위한 적응적 소프트웨어 구조

°
최창열◊ 박기진◊◊ 김성수◊
◊ 아주대학교 정보통신전문대학원
◊◊ 안양대학교 문리과학대학 컴퓨터학과
{clchoi◊, sskim◊}@kiss.or.kr, kiejin◊◊@anyang.ac.kr

XML-Based Adaptive Software Architecture for Grid Computing

°
Changyeol Choi◊ Kiejin Park◊◊ Sungsoo Kim◊
Graduate School of Information and Communication, Ajou University
Department of Computer Engineering, Anyang University

요 약

그리드 및 P2P 컴퓨팅 시스템과 같이 이종 시스템으로 연결된 환경을 관리하기 위해서 사용되는 미들웨어 및 운영체제의 복잡도의 증가로 인해 시스템 관리 비용이 증가하였다. 따라서 본 연구에서는 시스템 관리 비용을 최소화할 수 있는 시스템 구조적 접근 방식을 채택한다. 이질 컴퓨팅 환경을 통합하기 위해 XML 기반 기술을 사용하여 사람의 개입을 최소화할 수 있는 방법에 대해 제안하며, 적용 사례를 통해 제안된 구조와 기법에 대한 검증을 수행한다.

1. 서 론

IT 기술 진화에 따라 다소 원리적이던 이론들이 실제 구현이 가능할 수 있는 기반구조가 형성되고 있다. 이와 같은 진화를 네트워크 측면에서 살펴보면 기간망으로 시작하여 현재 가입자망이 보편화되었으며 점점 사용자망으로 변화되고 있는데, 대표적인 예가 그리드 컴퓨팅과 P2P(Peer-to-Peer) 컴퓨팅 환경이다. 또한 지역적으로 떨어져 있는 컴퓨팅 자원을 통합하여 웹 서비스와 같은 응용 서비스를 제공하기 위한 연구가 진행되고 있는데, 이와 같은 대형 분산 컴퓨팅 환경을 구현하기 위해서는 분산 공유 환경이 조성되어야만 한다[1]. 하지만 이와 같은 시스템의 대형화로 인해 발생된 문제점이 시스템 관리 비용(TCO, Total Cost of Ownership)의 기하급수적인 증가이다[2]. 시스템 관리 비용이란 하드웨어/소프트웨어 구입 및 유지를 위한 비용을 총 사용자의 비율로 산출한 것으로, 최근 시스템 유지·보수를 위한 비용이 총 비용의 대부분을 차지하고 있다. 이는 그리드 및 P2P 컴퓨팅 시스템과 같이 이종 시스템으로 연결된 환경을 관리하기 위해서는 전통적인 분산 컴퓨팅을 위한 소프트웨어 보다 더욱 복잡한 소프트웨어를 요구한다. 그러므로 미들웨어 및 운영체제의 복잡도가 증가하고 컴퓨팅 자원의 이질성도 늘었으며, 사용자에 대한 예측 불가능성으로 인해 시스템 관리 비용의 증가를 야기하게 된다. 하지만 지금까지 진행된 연구들은 이질성을 통합하여 지리적으로 분산된 자원을 사용 가능케 하기 위한 구조 및 메커니즘에 치중되어 진행되었다[3,4]. 최근 웹 서비스에 대한 스케줄링 전략을 통합하기 위해 XML-RPC(eXtensible Markup Language -Remote Procedure Call) 기반 구조[5]가 제안되었지만, 구조 정의 및 필요한 요소에 대해 명시하기 위해 DTD (Document Type Definitions)를 사용하였기 때문에 서비스 중 필요한 스키마(Schema)를 선택하여 변경할 수 없는 제약조건이 있다. 또한 순차 리스트와 선택 리스트만 제공되므로 구조 및 서비스에 대한 상세한 명세가 불가능하다.

따라서 본 논문에서는 그리드 컴퓨팅의 시스템 관리 비용을 최소화하면서 시스템 운행 중 자원 상태 변화 및 사용자 요구 변화에 따라 적응적으로 구조를 변경할 수 있는 시스템 구조를 도출하고 이를 위한 소프트웨어 구조적 접근 방식을 제안한다. 또한 적용 사례를 통해 접근 방식에 대한 검증을 수행한다.

2. 그리드 컴퓨팅 시스템 구조

그리드 컴퓨팅을 실현하는데 중요한 요소 기술은 자원

본 연구는 한국과학재단 목적기초연구(R05-2003-000-10345-0) 지원으로 수행되었음.
본 연구는 2003년도 두뇌한국21사업에 의하여 지원되었음.

의 다양성을 조정하여 지원할 수 있어야 하며, 사용자의 요구 사항 변화에 순응할 수 있어야 하고 하드웨어 및 소프트웨어 결함을 처리할 수 있어야 하는데, 이 모든 것들이 서비스 수행 시간에 가능해야만 한다. 또한 양질의 웹 서비스를 제공하기 위해서는 대역폭, 서비스 가능한 서버 수와 같은 컴퓨팅 자원의 변화에 효율적인 자원 관리를 제공할 수 있는 메커니즘을 선택하여 동작할 수 있어야 하며, 서버 결함, 네트워크 중지, 외부 장치의 고장 등과 같은 하드웨어적인 결함과 응용 프로그램·운영체제의 에러와 같은 소프트웨어적인 결함에 대해 사람의 개입을 최소화할 수 있어야만 한다. 하지만 지금까지의 구조는 사람에 의존적으로 동작하는데, 이는 프로그램을 설치 또는 제거할 때, 시스템이 공유 파일의 업데이트 또는 삭제에 관해 사용자가 무엇을 원하는지 물어오는 상황에서 확인할 수 있다. 따라서 시스템이 대형화되어 상호 연결성이 증가하여 종속성과 필요한 옵션이 크게 증가되지만 이질 환경을 통합할 수 있는 구조가 필요하다. 그림 1은 웹 서비스를 제공하기 위한 그리드 컴퓨팅 시스템 구조로 특정 미들웨어를 장치에 장착하거나 특정 응용 프로그램을 설치하여 서비스 하는 것이 아니고 탑재하고자 하는 시스템 환경에 따라 필요한 소프트웨어만을 장착할 수 있는 구조이다.

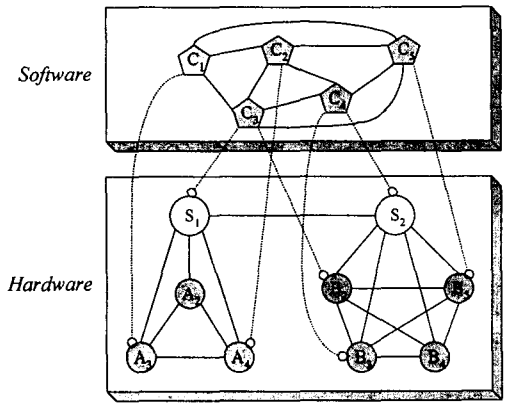

그림 1 웹 서비스를 위한 그리드 컴퓨팅 시스템 구조

서비스 개발자 및 시스템 프로그래머에 의해서 개발된 소프트웨어(C_1,…, C_5) 중 특정 시스템(A_1,…,A_3, B_2, …,B_5) 관리에 적합한 것을 선택하여 설치하며, 관리 노드(S_1, S_2)에 의해서 사용자의 요구 변화에 대한 확인 및 시스템 환경 변화에 대해 양질의 서비스를 지속적으로 제공하기 위한 역할을 담당한다. 이와 같은 구조를 위해서는 웹 서비스를 제공하기 위한 응용 계층과 시스템 구조에 대한 관리를 위한 계층 및 환경 변수의 변화에 대한 감시를 위한 계층으로 나뉘어 특정 계층의 변화가 다른 계층의 변화에 영향을 미치지 않는 프레임구조(Framework)가 필요한데, 이를 도식화하면 그림 2와 같다. 응용 레벨(Application Layer)는 새로 등록된 서비스를 추가하거나 기존 웹 서비스의 업데이트 및 삭제 등과 같은 서비스를 위한 소프트웨어 관리를 위한 것이다. 또한 서비스 관리자(Service Manager)는 응답 시간, 필요한 대역폭, 처리량, 서비스 마감 시간, 작업 우선 순위와 같은 서비스의 특성에 따라 필요로 하는 구조를 요구할 수 있다. 구조 관리자(Architecture Manager)는 응용 관리자의 요구를 수용하기 위해서나 시스템 환경 변수의 변화에 따라 소프트웨어의 선택, 추가, 수정과 같은 구조에 대한 관리를 담당한다. 마지막으로 환경 관리자(Environment Manager)는 사용 가능한 버퍼의 양, 네트워크의 대역폭 및 전송 시간 등과 같은 자원 상태를 감시하고 시스템의 결함 발생 여부에 대해 확인한다.

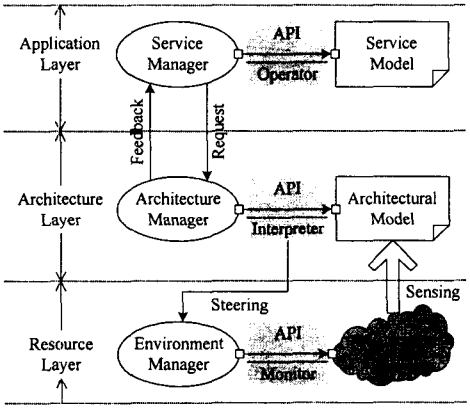

그림 2 적응적 환경 제공을 위한 프레임 구조

3. 적용 사례

본 장에서는 2장에서 제안된 구조를 구현하기 위한 요소 기술을 설명하고 적용 사례를 살펴본다. 먼저, 기본적인 프레젠테이션 도구로 XML 1.0 스펙에 따른 마크업 언어를 사용하며, HTTP(HyperText Transfer Protocol)와 같은 인터넷 전송 프로토콜을 배포 도구로, 자바와 같은 플랫폼 독립적인 언어를 응용 서비스 프로그램의 도구로 사용하면서, Java RMI(Remote Method Invocation)와 같은 분산객체용 미들웨어를 기반구조로 채택한다. 이와 같은 개발 환경에서 시스템 구조에 대해 명시할 수 있는 도구가 필요한데 이를 XML을 통해 개발하였다(그림 3).

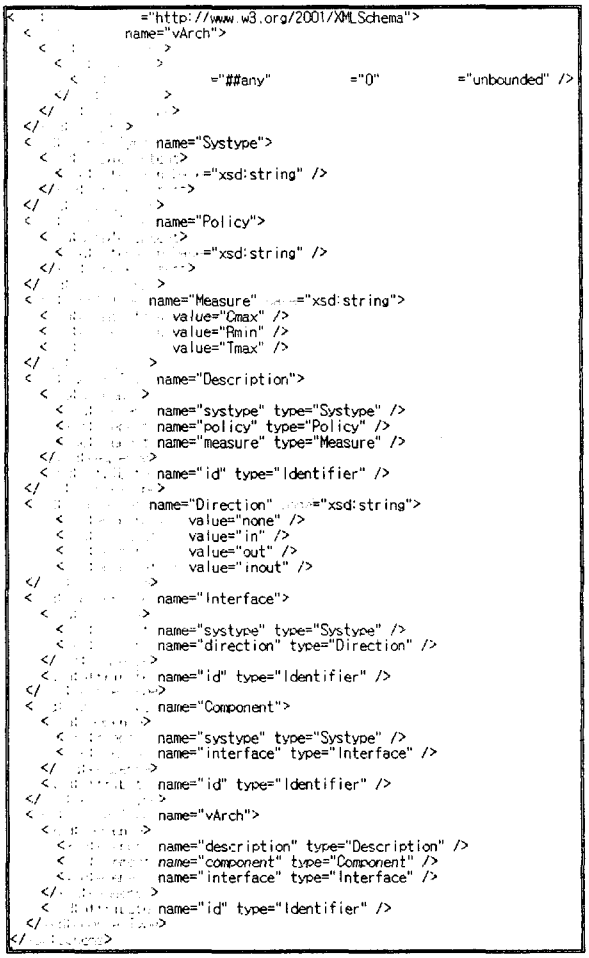

그림 3 시스템 구조 표현을 위한 기본 XML 스키마

서비스 관리자가 원하는 구조를 구조 관리자에게 전달하기 위해서는 필요한 시스템(systype)과 스케줄링 및 자원 관리 메커니즘(policy) 및 필요한 성능 및 의존도 관련 요구사항(measure)을 명시한 도큐먼트 인스턴스(Document Instance)를 생성하여 구조 관리자에게 전송한다. 이때 구조 관리자는 구조 변경을 위해 필요한 컴포넌트(Component)를 찾은 후 이를 탑재하기 위한 인터페이스(Interface)를 선택한다. 기본 XML 스키마에서 요소 타입 중 Direction은 기존 구조에 컴포넌트를 추가(in)할 것인지, 삭제(out)할 것인지, 아니면 변경(inout)할 것인지에 대한 것을 명시하기 위한 것이다. 또한 Measure 요소 타입은 최대한의 저장 용량(Cmax)을 필요로 하는 서비스와 응답 지연 시간을 최소화(Rmin)해야 하는 서비스, 단위 시간당 처리량을 최대화(Tmax)하기 위한 서비스를 정의하기 위한 타입이다.

또한 서비스 관리자는 웹 서비스 검색 및 등록을 위한 도구로 WSDL(Web Service Description Language)[6]를 사용하며, 예를 들면 그림 4와 같다.

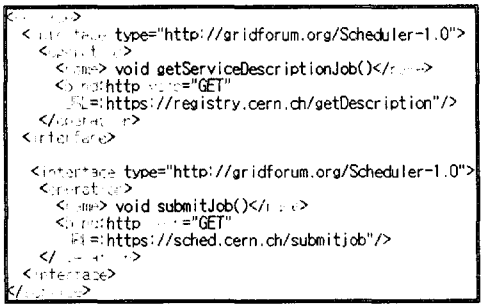

그림 4 웹 서비스 등록 및 검색을 위한 인스턴스

4. 결론

본 논문에서는 그리드 컴퓨팅 환경에서 웹 서비스를 제공하기 위한 소프트웨어 구조를 도출하고 환경 변화 및 사용자 요구사항에 따라 적응적으로 구조를 변경할 때 필요한 명세서를 작성하기 위한 메커니즘을 제안하였다. 향후 웹 서비스 기반 그리드 컴퓨팅의 데이터 관리를 위한 기법을 추가하여 구조를 확장할 것이다.

참고 문헌

[1] I. Foster, C. Kesselman, J. M. Nick, and S. Tuecke, "Grid Services for Distributed System Integration," IEEE Computer, Vol. 35, No. 6, pp. 37-46, June 2002.

[2] A. Gillen, D. Kusnetzky, and S. McLarnon, "The Role of Linux in Reducing the Cost of Enterprise Computing," IDC White Paper, Jan. 2002.

[3] B. Schmerl and D. Garlan, "Exploiting Architectural Design Knowledge to Support Self-repairing Systems," 4th International Conference on Software Engineering and Knowledge Engineering, pp. 241-248, July 2002.

[4] P. Oreizy, et.al., "An Architecture-Based Approach to Self-Adaptive Software," IEEE Intelligent Systems, Vol. 14, No. 3, pp. 54-62, May 1999.

[5] R. V. Maria, N. A. Joaquim, and C. S. Silvio, "A Scheduling Web Service based on XML-RPC," 13th International Conference on Automated Planning & Scheduling, June 2003.

[6] E. Christensen, F. Curbera, G. Meredith, and S. Weerawarana, "Web Service Description Language 1.1.," W3C Note 15, 2001.