클러스터 기반의 단계화된 응용서비스 플랫폼의 평가

준회원 김 태 훈*, 정회원 박 세 명**

Evaluation of the Cluster-based staged Application Service Platform

Tae hoon Kim*, Se myung Park** ***Regular Members***

요 약

본 연구에서는 PVM으로 구성된 클러스터의 공유를 기반으로 하는 단계화된 응용서비스 플랫폼을 기반으로 응용서비스를 구현함으로써 응용서비스 플랫폼의 유용성을 평가하였다. 응용서비스 플랫폼은 요청 처리에 필요한 전 과정을 요청처리 단계와 서비스제공 단계로 나누고, 서비스요청처리 단계를 위한 전위응용서버의 서비스관리자와 요청된 서비스의 분산을 담당하는 부하관리자, 그리고 후위서버에서의 서비스제공을 위한 작업관리자로 구성된다. 구현된 응용서비스 플랫폼은 필요한 처리자원을 동적으로 할당 시스템을 재구성함으로써 기존의 단일서버 시스템에 비해 부하의 변화에 보다 능동적으로 대처할 수 있을 뿐 아니라, 기능의 변경없이 다양한 응용서비스의 구현을 지원함을 확인하였다.

Key Words : cluster, application service platform, PVM, scalable, reconfigurable

ABSTRACT

In this paper, through the implementation of the application service, we evaluated the feasibility and availability of the staged application service platform, which is based on the sharing of the PVM cluster. Applicaton service platform provides three managers for dividing the request processing steps into two stages, such as a request processing stage and a service providing stage. Three managers and its relation to the divided stages are as follows, service manager and load manager to distribute the request in front-end server for a request processing stage, job manager in clustered(back-end) servers for a service providing stage. The experiment shows that the staged application service platform provides more stable and scalable characteristics and better performance improvement on the dynamic load changes than the single server system. And also it shows that real application service system can be implemented easily without modification of the proposed service platform.

Ⅰ. 서 론

인터넷의 보급은 단순히 데이터의 공유단계를 넘어 이제 지리적으로 분산되어 있는 처리자원의 공유를 통한 처리자원의 지역적 한계를 극복할 수 있는 계기를 제공하였다. 지역적으로 산재한 자원의 활용을 위해 생성된 클러스터링의 개념의 확장으로 메타컴퓨팅과 그리드 컴퓨팅[1,2]이라는 개념을 도출하기에 이르렀으며, 근자에 와서는 클러스터 및 그리드 컴퓨팅을 위한 기반구축과 더불어 이러한 환경에 적합한

* 인제대학교 전산학과 대학원 재학중, ** 인제대학교 컴퓨터공학부 교수
논문번호 : KICS2004-12-305, 접수일자 : 2004년 12월 6일
※ 본 연구는 2002년 인제장학재단의 연구비 지원에 의한 것임

다양한 특수목적의 응용의 개발에 관심이 집중되고 있다. 국내에서도 2001년 말에 Grid 관련 연구그룹이 형성되어 연구가 이루어지고 있으며 다양한 응용이 개발되고 있다.[3] 응용개발환경으로는 MPI[4]나 PVM (Parallel Virtual Machine)[5] 그리고 Globus[6]기반의 개발도구인 MPICH[7]등을 들 수 있다.

한편, 현재의 모든 인터넷 응용서비스들에 대한 요구는 부하에 무관하게 균일한 양질의 서비스제공과 서비스의 안정성 및 신뢰성 제공으로 집중되고 있으며, 이러한 요구로 인해 동시 접속자 수를 10,000명까지 확장하고자 하는 서버구성기술 개선(C10K Problem[8])에 관한 연구와 서버의 부하를 클러스터상의 처리기에 분산시키고자 하는 One-IP 서버구성 기술[9]에 대한 연구, 그리고 이벤트기반의 서버구성기술과 쓰레드기반의 서버구성 기술을 적절히 혼합하여 처리과정의 단계를 단계화하고, 단계별 작업처리과정에 적절한 제어를 수행함으로써 과부하시에도 적절한 작업처리를 보장하는 서버구성기술에 대한 연구(SEDA[16])등의 연구가 진행되었다. 한편 C10K Problem은 단일서버기반으로 많은 수의 동시 접속자 수를 허용하기 위한 전략들인 반면, 클러스터기반은 네트워크로 연결된 자원을 사용할 수 있으므로 두 기술이 상호보완작업을 거쳐 궁극적으로는 기존의 단일서버기반의 응용서비스들이 클러스터 기반으로 이전될 것으로 보이며, 이 과정에서 기존의 쓰레드기반 서버구성기술 방식에서 이벤트기반 서버구성 기술이 적절히 혼합되어질 것으로 판단되며 이러한 변화는 기존의 모든 응용서버 시스템의 전반에 걸쳐 적용될 수 있을 것으로 보인다.

그러나 최근 제안된 클러스터기반의 서버구성 기술 중 One-IP기반의 응용서비스 시스템들은 유휴 자원들을 하나의 응용서비스를 제공하기 위한 목적으로 클러스터링하고, 동일한 응용서비스를 클러스터된 후위서버(back-end server)마다 중복 설치하고, 요청을 분배 처리하는 구조를 가지고 있으므로 응용서비스를 위해서는 독점적인 사용이 허락된 클러스터가 필요하다. 물론 근자에 와서는 특히 고급 컴퓨팅 자원의 가용성 증대 및 통신 속도의 급격한 개선에 따라 값싼 비용으로 손쉽게 고성능 컴퓨팅 클러스터를 구축할 수 있게 되었으나 클러스터된 자원 관리의 효율성 및 다양한 서비스의 제공을 위해서는 클러스터된 자원을 공유할 수 있는 응용서버 구성기술에 대한 연구 및 개발이 필요하다. SEDA[16]의 경우에는 서비스제공을 위한 단계를 세분화하고 각 단계를 쓰레드와 이벤트기반의 처리모듈로 분류하여 서버의 확장성을 제공하고 있으나 분산된 환경에 대해서 고려하지 못하고 있다. SEDA에서 분산환경에서의 서버구성에 대한 연구가 부족한 이유로는 여전히 기존의 단일 서버의 처리절차의 모듈화를 통한 서버의 확장성에 주안점을 둔 때문이며, 이는 여전히 처리절차와 서비스 단계를 통합하여 고려하기 때문으로 분산환경으로의 확장을 위해서는 서비스와 요청처리절차의 구분이 필요한 것으로 보인다. 따라서 본 연구에서 제안한 서비스를 위한 별도의 모듈을 추가하여 SEDA의 기능을 확장하게 되면 보다 안정적인 서버의 구현이 가능할 것이다.

이상의 연구와는 차이가 있지만 근자에는 Globus 기반의 특수목적의 슈퍼컴퓨터를 개발하고자하는 시도도 보이고 있으나 Globus의 경우 개발목적인 WAN 기반의 자원공유를 위해 개발되었으므로 Globus가 지역 내의 클러스터자원의 활용을 위한 응용분야에 적합한지에 대한 평가도 진행되어야 할 것으로 보인다.

본 논문에서는 유휴 컴퓨팅 자원을 활용한 PVM 기반의 클러스터 상에서 클러스터된 자원을 공유하면서 확장성있는 응용서비스를 제공할 수 있는 응용서비스 플랫폼[16]에 대한 동작특성 평가를 바탕으로 제안된 플랫폼에 적합한 하는 응용서비스를 구현하였다. 응용서비스의 구현을 통해 기 제안된 플랫폼은 서비스 요청 부하에 따라 공유된 클러스터 상의 처리자원을 동적으로 할당 해당 서비스를 처리하는 서비스 시스템의 동적 재구성을 지원함으로써 시스템의 안정적인 응답시간을 보장할 뿐 아니라, 이를 바탕으로 다양한 응용서비스를 보다 효과적으로 설치 운영할 수 있도록 지원함을 확인할 수 있었다.

본 논문의 구성은 다음과 같다. 2장에서는 본 논문에서 이용한 기 구현된 응용서비스 플랫폼에 대한 전반적인 동작과 응용서비스 플랫폼의 구현 환경 및 성능평가를 통한 동작특성에 대해 기술한다. 3장에서는 응용서비스 플랫폼에 적용할 응용서비스의 선정요인 및 구현을 통한 타당성 평가를 기술하고, 마지막으로 5장에서 결론 및 향후과제를 기술한다.

Ⅱ. PVM 기반의 응용서비스 플랫폼

2.1 동작 개요

본 논문에서 응용의 개발을 위해 사용한 PVM기반의 응용서비스 플랫폼의 동작은 다음과 같다.

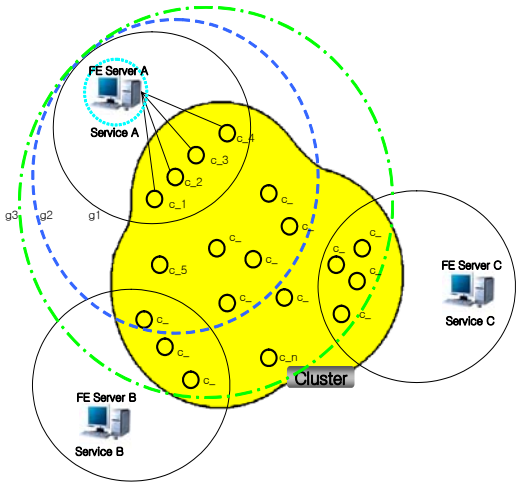

그림 1. 응용서비스 플랫폼의 동작 개요

그림 1은 클러스터 자원을 동적으로 사용하여 각 응용서비스를 제공해주는 응용서비스 플랫폼의 동작 개요를 도식화한 것이다. 그림에서 Service A, Service B, 그리고 Service C는 각 전위응용서버(FE Server)에서 제공되는 고유의 응용서비스로써 클라이언트의 서비스 요청은 저부하인 경우 해당 서비스의 전위응용서버에서 직접 처리된다. c_1, c_2, …, c_n 은 클러스터된 후위서버군으로 전위응용서버들에게 공유된다. 즉 각각의 응용서비스(Service A, Service B, 그리고 Service C) 요청은 해당 전위응용서버의 부하에 따라 해당 전위응용서버에서 직접 처리되거나, 공유된 클러스터의 후위서버를 할당받아 처리된다. 그림에 따라 설명하면, Service A를 제공하고 있는 전위응용서버 A는 현재 4개의 공유 클러스터 자원(g1)을 사용하여 Service A를 제공하고 있다. 부하가 증가하면 더 많은 클러스터 자원을 사용할 수 있고(g1->g2), 부하가 계속 증가하면 할당받는 클러스터 자원이 점차 증가하게 된다.(g3). 또한, 부하가 계속 감소하면 사용하고 있는 4개의 클러스터 자원까지도 반환되고(g3->g2->g1), 결국에는 전위응용서버가 직접 해당 응용서비스를 제공하게 된다.

2.2 응용서비스 플랫폼의 시스템 구성 요소 및 기능

본 연구에서는 사용한 응용서비스 플랫폼은 전위응용서버에 서비스관리자와 부하관리자, 그리고 후위서버에 작업처리자로 구성되며, 구현된 시스템의 구성도는 그림 2와 같다.

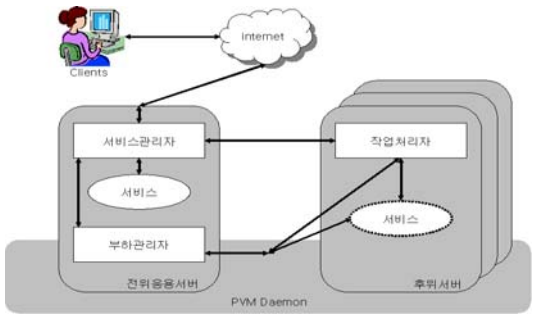

그림 2. 구현된 시스템 구성도

2.2.1 구성요소

구현된 시스템은 크게 전위응용서버와 클러스터된 후위서버군으로 구성되며, 세 개의 관리자를 갖는다. 전위응용서버의 서비스관리자는 클라이언트로부터 요청을 받으면, 자신의 서비스(저부하일 때) 혹은 해당 노드의 작업처리자(고부하일 때)에게 요청을 전달한다. 그리고 반환된 요청 처리 결과를 클라이언트에게 전달한다.

부하관리자는 전위응용서버에만 있으며, 전위응용서버의 부하와 사용 가능한 후위서버의 부하를 감시한다. 전위응용서버의 부하가 적정수준 이상이 되면, 부하관리자는 후위서버 중 저부하인 노드를 선택하고, 선택된 노드는 서비스관리자의 child_list에 등록된다. 이때, 선택된 노드의 작업처리자와 전위응용서버의 서비스와 동일한 응용서비스는 PVM 데몬을 통해 활성화된다. 그리고 리스트에 등록된 노드를 모니터하고 적정 수준 이상이 되거나, 그 이하로 떨어지면 child_list는 갱신된다. 여기에서 child_list는 서비스관리자가 요청을 자체적으로 처리할 것인지 클러스터 자원을 이용하여 처리할 것인지 결정사항이 된다. 즉, 서비스관리자는 child_ list에 등록된 노드가 없으면 지역에 있는 서비스에게 요청을 전달하고, child_list에 등록된 노드가 있으면 해당 노드의 작업처리자에게 요청이 전달된다. 후위서버의 작업처리자는 전위응용서버로부터 받은 요청을 자신의 서비스들 중, 이미 활성화된 서비스에게 요청을 전달하고, 처리 결과를 받아서 전위응용서버에게 전달하는 기능을 수행하며, 부하관리자의 요청에 따라 현재 처리중인 요청의 수를 반환한다.

2.2.2 부하에 따른 부하분산 정책

구현된 응용서비스플랫폼에서 제공하는 부하분산 정책은 시스템의 성능에 주요한 요인으로 응용분야에 적합한 부하분산 정책이 필요하다. 응용서비스

플랫폼은 동적으로 시스템을 재구성하므로 어느 시기에 가용자원을 추가 및 삭제를 하는 것이 효율적인가의 문제는 시스템의 성능에 결정적인 영향을 끼칠 수 있다. 단일 기준으로 고부하와 저부하를 구분하는 경우에는 부하의 변화에 따른 유휴자원의 추가 및 삭제가 빈번하게 발생하므로 이러한 과정에 시스템의 오버헤드로 작용할 가능성이 높으므로 유휴자원의 추가삭제가 빈번하게 발행하는 것을 방지하는 부하분산 정책이 필요하다.[16]

2.2.3 서비스

서비스는 클라이언트에게 실질적으로 제공되는 서비스로서 요청을 처리하고, 처리 결과는 작업관리자를 통해 또는 직접 서비스관리자에게 전달된다.

2.3 응용서비스 플랫폼 구현환경 및 동작특성

2.3.1 구현 환경 및 시스템 구성도

기 구현된 응용서비스 플랫폼은 LAN 환경에서 3개의 사설IP를 부여한 노드(Slave)들을 후위서버로 갖는 Master 노드와 모든 노드를 연결하는 100Mbps NIC와 dummy Hub로 구성된다. Master 노드는 CPU P4 1.8GHz, RDRAM 512Mbyte, HDD 14 Gbyte이고, Slave 노드는 CPU P2 200MHz, SDRAM 64Mbyte, HDD 2.4 Gbyte이다. 운영체제는 Linux 9.x이고 메시지전달 라이브러리로 PVM 3.4.4를 사용하였다. 구현된 응용서비스 플랫폼의 구현환경은 그림 3과 같다.

구현을 위해 고려한 시스템의 동작은 크게4개의 유형으로 나누어진다. 첫 번째는 사용자의 요청을 전위응용서버에서 처리할 경우이고, 두 번째는 전위응용서버의 부하가 증가하여 클러스터된 후위서버군 중 하나를 후위서버 사용가능 자원으로 등록하고 이를 이용하여 요청을 처리하는 경우이다. 세 번째 경우는 후위서버를 이용하는 중에 지속적으로 부하가 증가하여 또 다른 후위서버를 추가로 사용하는 경우이다. 네 번째는 부하가 감소하여 하나의 노드를 제거하는 경우이다. 즉 하나 이상의 노드를 이용하여 처리하는 중에 부하가 낮아져서 전위응용서버가 가지고 있는 후위서버 중 하나를 다시 유휴자원으로 반환하는 경우이다.[16]

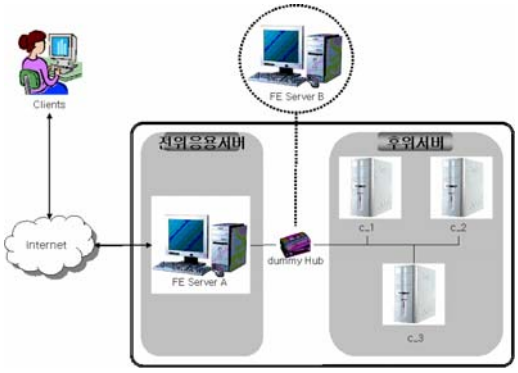

그림 3. 예제 시스템 구성

2.3.2 응용서비스 플랫폼의 특성분석

응용서비스의 성능평가를 위해서 외부와 연결되지 않은 가상IP를 가진 컴퓨터를 클라이언트로 활용하고, 이를 통해 가상적인 요청을 생성하여 서버에 전달하도록 하여 실험하였으며, 웹서버의 성능평가 도구인 httperf[14,15]를 이용하였다. 응용서비스 플랫폼의 동작을 평가하기위한 실험을 통한 응용서비스 플랫폼의 특성에 대한 평가는 다음과 같다.

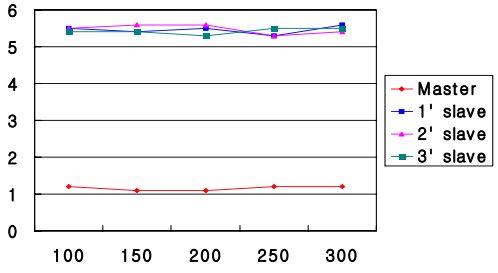

그림 4. Request 수 증가에 따른 응답속도

1) Request 수 증가에 따른 실험

그림 4는 요청파일을 1Kbyte로 고정한 후, 직접 응용서비스 서버에 요청했을 경우와 구현된 응용서비스 시스템에서 후위서버 사용 상태(후위서버의 수 = 0(Master), 1(1 Slave), 2(2 Slave), 3(3 Slave))에 따른 응답속도를 나타낸다.

실험은 요청율을 100에서 300까지 증가시키면서 수행되었다. 그 결과 구현된 시스템의 경우 후위서버를 추가하여 사용한 경우에 따른 이득은 미미하며, 단일서버의 경우에 비해 성능이 더 악화되었음을 알 수 있다. 이것은 하나의 요청을 처리하는 시간이 너무 짧고, 요청율 또한 단일서버에 오버헤더를 일으킬 만큼 충분하지 않아 관리자들의 동작에 따른 오버헤드가 부하의 분산에 따른 이득을 상쇄함을 알 수 있다. 즉 요청 처리시간이 요청 전달시간에 비해 짧은 응용의 경우 구현한 시스템이 적용되기 어려움을 보여주고 있다.

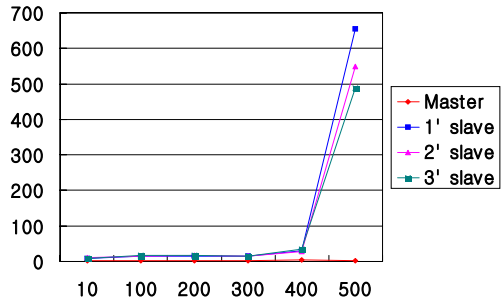

그림 5. File Size 변화에 따른 응답속도

2) File Size 변화에 따른 실험

그림 5는 초당 요청율을 '1'로 고정하고 전체 요청 수가 100일 때, 요청 파일의 크기를 증가시킬 경우(10Kbyte - 500Kbyte) 응답속도를 측정한 것이다. 그 결과, 본 논문에서 구현한 프로세스를 적용한 실험에서 요청 파일의 크기가 400K를 초과하게 되면 오버헤드가 발생해 시스템의 응답속도가 갑자기 증가했다. 그리고 약간의 차이지만 후위서버의 수가 많을수록 응답속도가 약간 빠름을 알 수 있다.

실험결과는 전송되는 자료의 양이 증가하면 결국 구현된 응용서버의 관리자간의 통신비용의 증가가 전체 시스템의 성능을 저하시키는 요인으로 작용함을 보여주고 있다. 실험 1, 2의 결과를 통해 요청율의 증가나 파일 크기 변화의 경우는 현재의 클러스터기반 서버에서는 오버헤드가 부하분산으로 인한 이득을 상쇄함을 알 수 있다. 이러한 원인으로 앞선 두 실험은 CPU 처리시간은 극히 미미하고 네트워크 트래픽 시간이 응답속도의 거의 대부분을 차지하여 전체 응답시간을 증가시키는 요인으로 작용함을 알 수 있었다.

3) CPU 처리시간을 요구하는 Request

실험 3을 위해, 실험결과에 대한 정확한 고찰을 위해 우선적으로 구현한 프로세스가 가지는 오버헤드를 측정해보았다.

그림 6의 왼쪽 그림은 일반적인 단일서버에서

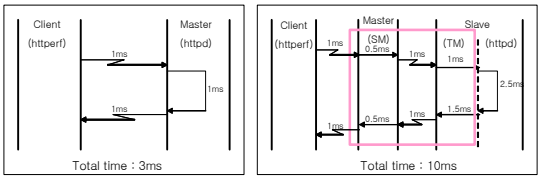

그림 6. 프로세스의 오버헤드 측정

Httpd 서비스를 제공하는 경우 1Kbyte 파일을 요청하는 경우의 오버헤드를 측정한 것이다. 오른쪽 그림은 본 논문에서 구현한 프로세스를 적용했을 경우의 오버헤드를 나타내고 있다. 그림 6에서 왼쪽의 Master와 오른쪽의 Slave의 httpd의 처리속도 차이는 3.1절에서 실험 장비 설명에서 Master와 Slave 후위서버의 성능 차이로 인해 발생하는 차이이다. 여기에서 우리는 단일서버에서 작동하는 경우와 구현된 시스템간의 성능이 약 3배 정도의 차이가 생기는 것을 볼 수 있다. 즉 구현된 시스템은 처리시간이 짧은 응용의 경우이거나 전송되는 문서의 양이 많은 경우 오버헤드가 증가하게 되어 전체적인 시스템의 성능이 저하시키는 요인으로 작용함을 확인할 수 있었다.

실험 1과 2의 경우에는 각 Request 마다 네트워크 사용량에 비해 CPU 처리시간이 너무 짧아서 클러스터 자원을 사용하는 이점을 얻을 수 없었으므로 실험 3에서는 각 요청 당 10ms의 처리시간이 사용되는 경우 구현된 시스템의 요청 증가에 따른 성능을 평가하였다. 그림 7은 10ms의 CPU 처리시간을 추가한 경우의 실험결과이다. 각 Request들이 크기가 1K인 파일을 요청하고, 요청별 지연이 부가된 환경에서 초당 요청율을 '1'에서 '250'까지 증가시키면서 응답속도를 기록한 것이다. 실험결과 요청율이 '80' 미만 일 경우에는 전위응용서버만으로 처리하는 경우(Master)가 3개의 후위서버를 사용하는 경우(3' Slave)보다 응답속도가 더 빠르다는 것을 보이고 있는데, 이는 후위서버와 전위응용서버와의 성능차이와 전위응용서버에서 후위서버로 요청을 전달하는 과정에서 수반되는 오버헤드에 기인한 것으로 판단된다. 요청율이 '80'이상이 되면 3개의 후위서버를 사용하는 경우(3' Slave)가 안정적인 응답속도를 보임을 알 수 있다.

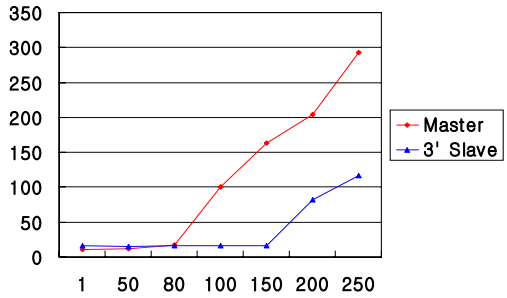

그림 7. CPU의 처리시간을 추가한 응답속도

이러한 결과를 통해서 요청의 특성, 즉 처리시간, 통신비용이 응용서비스의 성능에 결정적인 영향을 끼침을 알 수 있으며, 본 논문에서 구현한 응용서비스 시스템은 처리시간을 필요로 하는 응용의 경우에 적합함을 알 수 있다. 또한 응용서비스 플랫폼이 적용될수 있는 환경으로 사용자의 작업처리시간이 10ms인 경우 최대 150명의 사용자가 동시에 작업처리를 요청하더라도 시스템의 부하를 적절히 조절하여 적절한 응답시간을 보장할 수 있음을 보여준다.

Ⅲ. 응용서비스의 구현

3.1 응용서비스 선정시 고려 사항

본 논문에서 구현된 클러스터기반의 응용서비스 플랫폼을 실용성을 평가하기위해 실험을 통해 분석된 결과를 바탕으로 적절한 응용을 선택, 구현하였다. 구현될 응용은 적절한 처리시간(10ms 이하)을 가지고 사용자의 수가 150명 이내인 경우에 적용될 수 있는 응용으로 제한함으로써 응용서비스에 대한 별도의 실험없이 결과를 추정할 수 있도록 하였다. 중요한 점검사항은 새로운 응용서비스의 구현과정에서 기 구현된 응용서비스 플랫폼의 기능에 추가 없이 적용될 수 있어 다양한 응용서비스에 적용될 수 있다는 점을 확인하여야 한다. 또한 현재 구현 중인 많은 처리시간을 요구하는 응용[19]의 경우에는 보다 좋은 성능을 보일 것으로 예상된다.

Ⅳ. 응용서비스 구현 및 평가

4.1 응용서비스 구현시 고려사항

본 논문에서는 웹기반 C프로그램제출(컴파일/실행 확인 후 제출)/평가(동작확인) 서비스를 제공할 수 있는 응용서비스를 구현하였다. 해당 서비스의 경우 컴파일/실행을 위해 적절한 처리시간을 요구하며, 처리결과는 오류메시지나 실행결과를 NFS기반의 파일 시스템을 통해 전송함으로써 통신비용의 과다발생으로 인한 성능저하가 발생하지 않는다. 구현된 응용서비스는 웹기반으로 프로그래밍 리포트 제출시 컴파일/실행을 통한 정상동작 확인이 실시간으로 가능하며, 평가를 위한 배치작업의 지원도 가능하다.

그림 8은 응용서비스 플랫폼에 웹기반 C프로그램 제출/평가를 가능하게 하는 응용서비스 시스템의 구성도이다. 그림 8에서 점선으로 표시된 영역은 C프로그램강의 지원을 위해 추가된 부분에 해당하며, 본 논문에서는 전위응용서버 상에 웹서버를 설치하고 클라이언트 인터페이스를 위한 JSP 페이지 구성하였으며, 서비스관리자를 서버 측의 JSP모듈로 구현하였다. 따라서 구현된 응용서비스 시스템은 기 구현된 응용서비스 플랫폼에 웹기반 사용자 인터페이스를 제공하기 위한 기능만 추가됨을 알 수 있다. 즉 기 구현된 응용서비스 플랫폼의 주요동작에는 어떠한 변화도 끼치지 않고 새로운 서비스를 위한 기능이 부가될 수 있음을 보이고 있다. 이러한 결과는 다양한 응용서비스가 구현된 응용서비스가 제시된 응용서비스 플랫폼상에 적용될 수 있음을 확인할 수 있다.

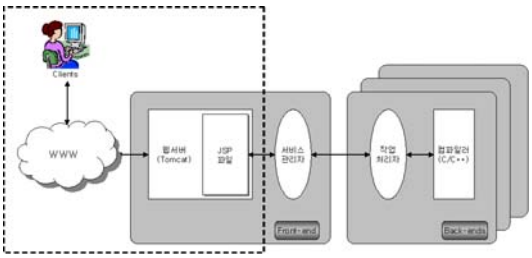

그림 8. 응용서비스 시스템 구성

본 논문에서 응용서비스로 구현한 웹기반 C프로그램제출/평가 시스템의 주요 기능은 과제의 제출 및 저장(제출자), 컴파일(제출자/평가자), 그리고 실행(제출자/평가자) 및 평가(평가자) 단계로 나뉘며, 각각 작업의 실행과정에 대한 설명은 아래와 같다.

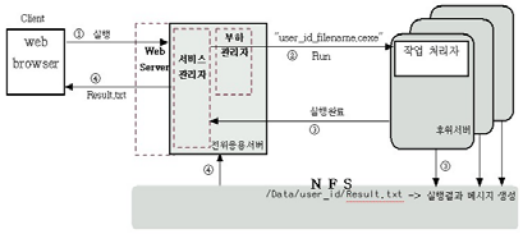

그림 9. 컴파일된 과제의 실행 절차

그림 9는 사용자가 JSP 인터페이스를 이용하여 제출할 프로그램을 작성한 후 다음 동작을 위해 작성된 프로그램을 저장하는 과정을 보이고 있다. 사용자가 C프로그램을 작성 후 저장버튼을 클릭하면 클라이언트의 jsp페이지는 사용자가 작성한 코드의 내용과 파일의 이름을 서버로 전송하며, 서버측 서비스관리자는 파일을 "/Data/user_id/ user_id_filename.c"로 저장한다.

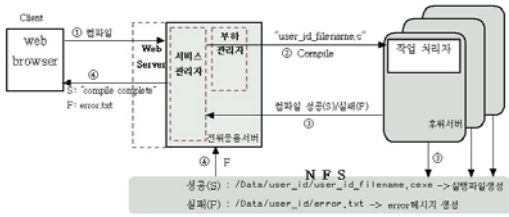

그림 10. 저장된 과제의 컴파일 수행 절차

그림 10은 사용자가 파일을 저장한 후 컴파일을 하는 단계이다. 사용자가 컴파일 요청하면 전위 응용서버의 서비스관리자는 부하분산 정책(round robin)에 의해 선택된 서버의 작업 관리자를 구동시키고 사용자의 컴파일 요청을 처리할 파일이름과 함께 전달한다. 작업을 지시 받은 서버는 서비스(gcc 컴파일러)를 이용하여 처리(컴파일)한다. 에러가 발생시는 에러 메시지는 저장 후, 사용자에게 전송되며, 정상실행시는 사용자에게 컴파일 성공 메시지를 보여주고 실행파일을 서버에 저장한다.

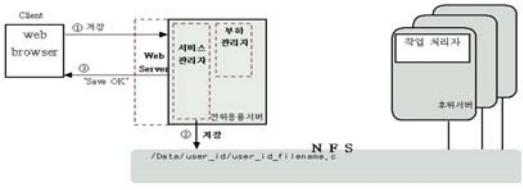

그림 11. 작성된 과제의 저장 절차

그림 11은 실행하는 단계를 나타낸다. 사용자가 실행을 요청하였을 경우에는 컴파일 요청 시와 같은 방법으로 서버를 선택 후 해당서버의 서비스 관리자에게 사용자의 실행 요청을 전달하게 된다. 요청을 받은 서버의 작업관리자는 작성된 실행 파일을 실행시키고 실행된 결과를 파일로 저장한 후 이를 서비스 관리자를 통해 웹기반 사용자 인터페이스로 전달된다.

그림 12는 jSP로 구현된 사용자 인터페이스이며, 이를 통해 제출할 C프로그램을 입력/편집한 후 동작(컴파일/실행)을 확인한 후에 제출(저장)한다. 이처럼 응용서비스가 웹기반의 인터페이스를 지원하게 되므로 관리자의 편의를 위해 과제제출 기한이 종료되면 과제제출자를 확인, 과제를 평가할 수 있는 과제평가 인터페이스뿐 아니라 사용자의 편의를 위해 제출된 과제의 평가여부와 평가등급, 그리고 평가사유등을 웹에서 확인할 수 있는 다양한 인터페이스의 지원이 가능하다.

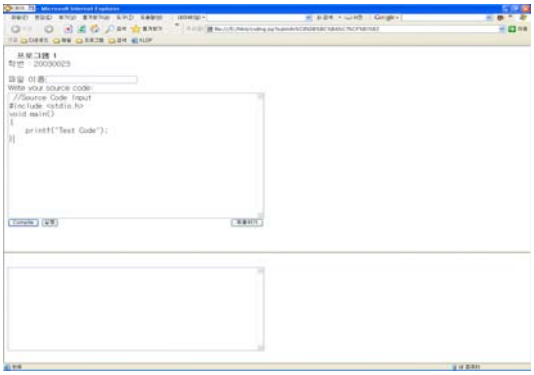

그림 12. 과제제출 인터페이스

이상에서 구현된 웹기반 C프로그램 제출/평가시스템을 통해서 알 수 있듯이 많은 처리용량을 요구하는 응용서비스인 경우 구현된 응용서비스 플랫폼상에서 구현되면 여러 사용자가 동시에 작업의 처리를 요청하더라도 만족할 만한 응용서비스를 제공할 수 있을 뿐 아니라 기 구현된 응용서비스 플랫폼은 모듈화된 단위로서 사용될 수 있으므로 손쉽게 응용서비스에 적용될 수 있음을 보이고 있다. 이러한 시스템의 구현은 많은 처리자원을 요구하는 사용자의 요청이 공유된 클러스터 처리자원을 통해 충분히 제공될 수 있다는 가능성을 보이고 있다. 이는 다양한 형태로 구현된 객체 간에 연동방식이 제안[18]되고 있는 현실을 감안해볼 때 다양한 응용서비스를 제공할 수 있는 서비스플랫폼으로 발전될 것이라 판단된다.

Ⅴ. 결론 및 향후과제

본 논문에서 구현한 응용서비스 시스템은 부하의 증감에 관계없이 모든 클라이언트들에게 안정된 서비스를 제공할 수 있음을 확인하였다. 그리고 본 시스템의 경우에는 서비스처리를 위해 요청처리와 서비스제공의 2단계로 전 단계를 분류함으로써 작업의 부하를 시스템내에서 분산시키는 효과를 가짐으로써 확장성있는 서비스의 제공도 가능하다. 그러나 요청처리단계에서의 병목현상의 가능성은 여전히 상존하고 있으므로 시스템의 확장성과 안정성을 제고하기 위한 연구가 계속되어야 할 것이다. 또한 작업처리 부하가 서비스관리자로 집중되는 문제점을 개선하기 위한 연구가 추가로 수행되어야 한다. 이는 결국 응용서비스 플랫폼의 구현을 위해 사용된 PVM의 성능분석을 통해 응용에 적합한 기능을 제

공하는 효율적인 미들웨어에 대한 연구가 필요함을 보이고 있다.

참 고 문 헌

[1] Ian Foster and Carl Kessleman, "The Grid : Blueprint for a New Computing Infrastructure," Morgan Kaufmann Pub. Inc., 1999.

[2] Rajkumar Buyya, "High Performance Cluster Computing : Architecture and Systems, Volume 1," Prentice Hall, 1999.

[3] 윤찬현 외 4인, "인공심장의 혈류해석을 위한 Globus 기반 협업 기술 개발," ITRC Forum 2002, Seoul, Korea, May. 2002.

[4] W. Gropp, "Learning from the success of MPI," High Performance Computing - HiPC 2001, number 2228 in Lecture Notes in Computer Science, pp 81-92, Dec. 2001.

[5] A. Geist, A. Beguelin, J. Dongarra, W. Jiang, R. Manchek, and V. Sunderam, "PVM : Parallel Virtual Machine A User s Guide and Tutorial for Network Parallel Computing," MIT Press, Cambridge, MA, 1994.

[6] A. Geist, A. Beguelin, J. Dongarra, W. Jiang, R. Manchek, and V. Sunderam, "PVM : Parallel Virtual Machine: A Users Guide and Tutorial for Networked Parallel Computing. Scientific and Engineering Computation," MIT Press, Cambridge, MA, USA, 1994.

[7] W. Gropp, E. Lusk, N. Doss, and A. Skjellum, "A High Performance, Portable Implementation of the MPI Message Passing Interface Standard," Parallel Computing, Volume 22, number 6, pp 789-828, Sep. 1996.

[8] The C10K Problem, "http://www.kegel.com/c10k.html"

[9] Om P. Damani, P. Emerald Chung, Yennun Huang, "ONE-IP: Techniques for Hosting a Service on a Cluster of Machines," In Proc. the Sixth International WWW Conference, http://decweb.ethz.ch/WWW6/Technical/Paper196/Paper196.html, Apr. 1997.

[10] "http://www.pdl.cmu.edu/Pasis/survivablestor-ageindex.html"

[11] 권세오, 김상식, "리눅스 클러스터형 웹 서버 설계," 한국정보과학회지 제18권 3호, pp. 48-56, Mar. 2000.

[12] Greg Regnier, "CSP: A System-Level Architecture for Scalable Communication Services," Intel Technology Journal Q2, http://developer.intel.com/technology/itj/q22001.htm , May, 2001.

[13] 서대화, 민병준, 이기욱, "네트워크 연결형 스토리지의 기술 동향," 한국정보과학회지, 제19권 3호, pp 6-13, Mar. 2001.

[14] The httperf Tool, "ftp://ftp.hpl.hp.com/ pub /httperf/"

[15] David Mosberger and Tai Jin, "httperf - A Tool for Measuring Web Server Performance," In Proc. the SIGMETRICS Workshop on Internet Server Performance, pp 59-67, Jun. 1998.

[16] Matt Welsh,"SEDA:An Architecture for Highly Concurrent Server Application", http://www.eecs.harvard.edu/~mdw/proj/seda/

[17] 김동근외 1, "클러스터기반의 확장과 동적재구성 가능한 인터넷서비스 시스템," 멀티미디어학회논문지, 제7권 10호, 2005.

[18] 박철우외 1,"분산객체들의 통합을 위한 확장성 있는 SOAP Bridge 제안", 인제논총 제출

[19] Y. Han, J. Kim, and M. Kim, "A New Moire Smoothing Method for Color Inverse Halftoning," Proceeding of ICIP 2002, Vol. 1. pp. 820-823, Sep. 2002

김 태 훈(Tae hoon Kim) 준회원

2003년 2월 인제대학교 컴퓨터공학부졸업(공학사)

2005년~현재 인제대학교 전산학과 대학원

<관심분야> 분산처리, 유비쿼터스컴퓨팅

박 세 명(Se myung Park) 정회원

1994년 8월 경북대학교 전자공학과 전산공학 전공(Ph.D)

1990년 3월~현재 인제대학교 컴퓨터공학부 교수

<관심분야> 분산처리, 유비쿼터스컴퓨팅, Grid컴퓨팅 지능형홈시스템