# 씬-클라이언트 환경에서의 터미널 서비스를 위한 적응적 서버 클러스터링

## (An Adaptive Server Clustering for Terminal Service in a Thin-Client Environment)

정 윤 재 † 곽 후 근 †† 정 규 식 †††
(Yunjae Jung) (Hukeun Kwak) (Kyusik Chung)

**요 약** 수십 대의 PC들로 구성된 학교 PC실 또는 교육 목적 PC실에서는 컴퓨터들이 분산 구조로 되어 있어서 각 컴퓨터별로 셋업, 유지보수, 업그레이드가 각각 따로따로 수행된다. 이러한 분산 구조에 대한 대안으로 씬 클라이언트 컴퓨팅 환경을 고려해 볼 수 있다. 씬 클라이언트 컴퓨팅 환경에서, 클라이언트 쪽 장치는 사용자에게 친숙한 GUI와 멀티미디어 지원과 함께 주로 IO 기능들을 제공하는 반면에 터미널 서버라 불리는 원격 서버들은 컴퓨팅 파워를 제공한다. 이 환경에서는 많은 클라이언트를 지원하기 위해서 터미널 서버들을 클러스터로 구성할 수 있다. 그러나 이러한 구조에서는 터미널 세션의 유지와 사용자의 다양한 컴퓨팅 사용 패턴 요인으로 부하 분산이 어렵고 결과적으로 터미널 서버 자원의 활용도가 낮아지는 단점을 가진다. 이러한 단점을 보완하기 위해 본 논문에서는 적응적 터미널 클러스터를 제안한다. 이 구조에서는 부하가 적은 그룹에 속한 터미널 서버가 부하가 큰 그룹으로 실시간에 동적으로 재 할당될 수 있다. 제안된 적응적 터미널 클러스터를 일반적인 터미널 클러스터와 그룹 기반 비적응적 터미널 클러스터와 비교하고 실험을 통해 제안된 방법의 유효성을 검증하였다.

**키워드** : 적응적 서버 클러스터링, 씬-클라이언트, 터미널 서비스

***Abstract*** In school PC labs or other educational purpose PC labs with a few dozens of PCs, computers are configured in a distributed architecture so that they are set up, maintained and upgraded separately. As an alternative to the distributed architecture, we can consider a thin-client computing environment. In a thin-client computing environment, client side devices provide mainly I/O functions with user friendly GUI and multimedia processing support whereas remote servers called terminal server provide computing power. In order to support many clients in the environment, a cluster of terminal servers can be configured. In this architecture, it is difficult due to the characteristics of terminal session persistence and different pattern of computing usage of users so that the utilization of terminal server resources becomes low. To overcome this disadvantage, we propose an adaptive terminal cluster where terminal servers are partitioned into groups and a terminal server in a light-loaded group can be dynamically reassigned to a heavy-loaded group at run time. The proposed adaptive scheme is compared with a generic terminal service cluster and a group based non-adaptive terminal server cluster. Experimental results show the effectiveness of the proposed scheme.

**Key words** : Adaptive Server Clustering, Thin-Client, Terminal Service

## 1. 서 론

터미널 컴퓨팅 환경의 기본은 사용자에게 단말기와 같은 접속 도구의 부담을 없애고, 컴퓨팅에 필요한 요구사항을 터미널 컴퓨팅 환경을 제공하는 터미널 서버에서 만족시켜주는 것이다. 그림 1과 같은 이러한 환경을 Thin-Client 환경이라 하고, 사용자를 Thin-Client라고 한다. 사용자의 단말기는 가상의 데스크톱을 보여줄 수 있는 디스플레이와 사용자 입력을 받기위한 최소의 입력도구,

· 이 연구는 2003년도 숭실대학교 교내 연구비 지원에 의해 수행되었음
† 정 회 원 : 숭실대학교 정보통신전자공학부
jgyver@q.ssu.ac.kr
†† 비 회 원 : 숭실대학교 정보통신전자공학부
gobarian@q.ssu.ac.kr
††† 종신회원 : 숭실대학교 정보통신전자공학부 교수
kchung@g.ssu.ac.kr
논문접수 : 2004년 5월 24일
심사완료 : 2004년 11월 19일

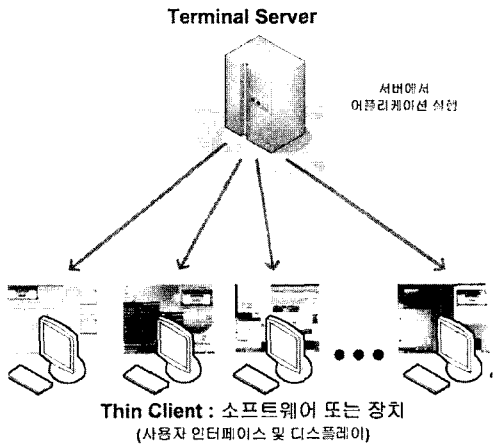


그림 1 터미널 컴퓨팅 환경

네트워크 연결을 위한 인터페이스만을 필요로 한다.

터미널 서버를 통해 다수의 사용자에게 터미널 컴퓨팅 환경을 제공할 수 있다. 이는 별도의 터미널 세션(Session)으로써 사용자와의 연결을 관리하고, 연결 간에 독립성을 유지함으로써 가능하다. 그러나 사용자들은 터미널 서버의 자원을 함께 사용하므로 터미널 서버의 부하는 사용자의 수, 사용자의 작업량에 따라 크게 영향을 받게 된다. 따라서 한대의 터미널 서버로는 자원의 부족 및 성능 한계에 이르게 될 것이다. 이를 해결하기 위해, 여러 대의 터미널 서버를 두고, 서버들을 클러스터로 구성하여 부하 분산을 하게함으로써 확장 가능한 가상의 고성능 터미널 서버를 구축할 수 있다.

서버 클러스터링에는 과학용 연산을 고속으로 처리하는 슈퍼 컴퓨팅분야와 웹서버, 메일서버, 스트리밍서버, 데이타베이스 서버 구축등에 사용되는 인터넷 서버 컴퓨팅분야가 있다[1 5]. 전자에서는 CPU bound 일들을 여러 대의 컴퓨터가 나누어 연산하는 방식이며, 후자에서는 CPU bound 이면서 Network bound 일들을 여러 대의 컴퓨터가 나누어 처리하는 방식이다. 후자의 경우에서는 보통 클러스터링된 서버들 앞단에 L4 스위치를 설치하여 네트워크 트래픽 및 서버들의 부하를 분산하는 기능을 담당하게 된다[6]. 본 논문에서 다루는 터미널 서비스 서버 클러스터링은 후자의 방식에 속하는데 터미널 서비스의 특성이 인터넷 서비스 특성과 다르다. 인터넷 서비스에서는 네트워크 기반 세션(예, HTTP 접속)이므로 세션이 맺어지고 끊어지는 주기가 매우 짧은 편이다. 반면 터미널 서비스에서는 한번 접속이 맺어지면 로그오프할때까지 계속 유지되는 특성으로 말미암아 접속이 맺어지고 끊어지는 주기가 아주 긴편이다. 이러한 특성차이로 말미암아 클러스러링 환경에서의 서버 부하분산은 더 어렵다.

터미널 세션은 웹과 같은 인터넷 서비스의 연결과 달리 한 번의 접속으로 장기간 연결이 유지되며, 사용자는 이 연결 하에서 컴퓨팅을 행하게 된다. 즉, 사용자의 실행 소프트웨어나 작업에 따라 터미널 서버의 자원 사용량이 크게 영향을 받는다. 따라서 단순한 부하 분산구조의 클러스터를 구성 시에는 효율적인 부하 분산이 되지 않고 비효율적 구조의 부하 분산 패턴이 나타나게 된다는 문제점이 발생한다. 이를 개선하기 위해서는 단순 세션 간의 부하분산이 아닌 세션 내에서 이루어지는 컴퓨팅 패턴이나 자원사용량에 따른 부하 분산이 필요하다. 또한 기존 클러스터링 기반 서버 부하분산의 경우 서비스 종류가 여러 개 있을 경우 각 서비스를 담당하는 서버 그룹을 정적으로 할당한 상황에서 각 그룹내 서버들의 부하분산을 다룬다. 터미널 서비스의 특성상 서비스별 부하 불균등화 현상이 발생할 가능성이 크므로 각 서비스를 담당하는 서버 그룹들을 수행시간에 동적으로 조정하면서 서비스별 서버 그룹들사이의 부하까지 조정이 필요하다. 즉, 가변적 상황에 따른 적응적 터미널 서비스 클러스터(Adaptive Terminal Service Cluster)가 구성 되어야 한다.

이에 본 논문에서는 Thin-Client 환경에서 다수의 사용자에게 터미널 컴퓨팅 환경을 제공하기 위해 클러스터링 구조를 사용하고, 터미널 세션 하에서도 서버 자원의 효율적으로 활용하고 부하 분산을 가능케 하는 적응적 터미널 서비스 클러스터에 대해 연구한다.

기존 클러스터링 기반 서버 부하분산의 경우 서비스 종류가 여러 개 있을 경우 각 서비스를 담당하는 서버 그룹을 정적으로 할당한 상황에서 각 그룹내 서버들의 부하분산을 다룬다. 본 논문에서는 각 서비스를 담당하는 서버 그룹들의 부하 상태를 모니터링하다가 그룹간 부하 불균형이 발생하면 수행시간에 서버 그룹 재설정을 통해 부하분산을 적응적으로 처리하는 부하 분산 방법을 제안한다.

본 논문의 구성은 다음과 같다. 2장에서 터미널 서비스에 대해 설명하고 터미널 세션의 특징을 이해한다. 3장에서는 터미널 서비스 클러스터 구성에 대한 설명 및 문제점을 파악하고 4장에서는 제안된 적응적 터미널 클러스터 구조에 대해 설명한다. 5장에서는 제안된 클러스터의 실험과 결과를, 6장에서는 결론 및 향후 연구 방향을 제시한다.

## 2. 터미널 서비스 환경

터미널 서비스는 사용자에게 터미널 데스크톱 환경을 네트워크를 통해 제공해 주는 서비스를 말한다. 이는 터미널 클라이언트 소프트웨어를 통해 이루어지는데 사용자의 마우스, 키보드 입력에 대해 로컬 컴퓨팅 환경과

동일한 인터페이스를 제공한다. 그림 2는 터미널 서비스의 흐름도를 나타낸다.

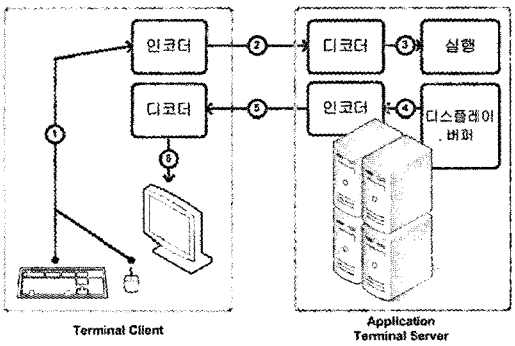


그림 2 터미널 서비스 흐름도

이런 클라이언트가 서버와 연결되어 터미널 컴퓨팅 환경을 제공해 주기 위해서는 프로토콜이 정해져야 한다. 이러한 프로토콜을 RDP(Remote Display Protocol) 라고 한다. 다음은 현재 사용되고 있는 터미널 컴퓨팅 플랫폼들을 나타낸다[7].

- Citrix MetaFrame for Windows[8-10]:
  ICA (Independent Computing Architecture)
- Microsoft Windows Terminal Services[11,12]:
  RDP (Remote Desktop Protocol)
- Tarantella Enterprise Express for Linux[13-15]:
  AIP (Adaptive Internet Protocol)
- AT&T VNC for Linux[16-18]:
  VNC (Virtual Network Computing)
- Sun Ray for Solaris[19,20]
- Xfree86 on Linux[21,22]

터미널 서비스는 접속 기반 통신을 이용한다. 따라서 터미널 데스크톱 환경을 이용하기 위해서는 연결 설정 작업과 소멸작업이 필요하다. 터미널 클라이언트가 서버로 터미널 서비스 접속을 시도하게 되면 터미널 서버는 클라이언트 접속을 받아들이게 된다. 서버가 자원의 부족 등으로 더 이상의 터미널 세션을 생성 못할 경우는 접속을 거부하게 된다. 터미널 서버가 클라이언트의 접속을 받아들이면 클라이언트로 로그인을 위한 화면 정보를 보내주고 사용자 입력을 기다린다.

터미널 세션은 터미널 클라이언트가 터미널 서버에 접속하였을 때 생성된다. 이후에 로그인을 하게 되면 사용자의 환경 복원을 위해 사용자 프로필을 디스크로부터 읽어와 메모리상에 사용자의 데스크톱 환경을 복원한다. 사용자의 작업 상태는 사용자별 프로필 형태로 저장/복원 된다. 터미널 세션의 네트워크 연결은 사용자의 연결 끊김이 있을 때 사라지게 되지만, 터미널 서버가 관리하는 메모리상의 터미널 세션은 사용자가 로그오프인 경우에 소멸된다. 터미널 서비스에서의 터미널 세션은 네트워크 세션의 의미보다는 터미널 서버의 메모리에 상주하는 사용자 터미널 세션을 의미한다. 이는 같은 사용자에 대해 기존 세션과 새 세션의 연관성이 존재하기 때문이다. 즉, 새 세션을 생성하기 위해서는 기존 세션이 로그오프 되어 프로필에 저장되어야 한다. 그렇게 해야 새 세션이 저장된 프로필을 사용하여 사용자 환경을 복원할 수 있기 때문이다. 또한, 터미널 세션 하에서 한 번의 네트워크 연결 후 새로운 연결의 시도 없이 기존 연결을 재사용한다. 이러한 이유들로 인하여 터미널 서버들의 부하 분산에 제약을 가하게 된다.

## 3. 터미널 서비스 클러스터

그림 3은 Windows Terminal Services[11, 12]의 구성을 나타내고 그림 4는 터미널 서비스 클러스터 환경에서 사용자의 터미널 접속 요청 처리 순서를 나타낸다.

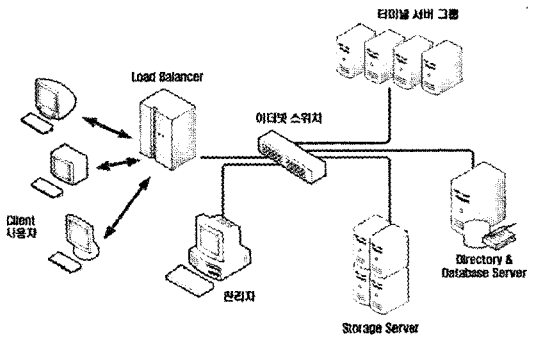


그림 3 터미널 서비스 클러스터의 구성

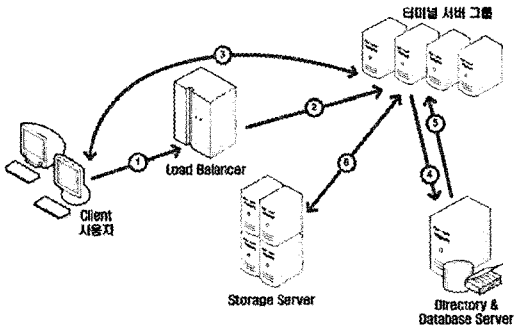


그림 4 터미널 서비스 클러스터의 동작 흐름

접속 요청이 들어왔을 때 부하분산을 위하여 Load Balancer가 동작한다. 접속 요청이 아닌 패킷은 기존 세션의 패킷이므로 현재 기준의 부하 분산이 아닌 기존 부하 분산 기록을 토대로 기존과 동일한 터미널 서버로

전달되어야 한다. 터미널 세션의 특징상 부하분산을 위한 Load Balancer 의 동작시점이 처음 한번 발생 후에는 재접속 시도가 있지 않는 한 부하분산이 불가능하다. 사용자가 컴퓨팅 자원을 사용하는 정도가 컴퓨터 부하에 큰 영향을 준다 하더라도 부하 분산을 할 수 없는 것이다. 또한, 터미널 세션의 종료가 아닌 단순 네트워크 연결 끊김으로 인한 재접속의 경우에는 터미널 세션이 남아있는 터미널 서버로 재접속을 해주어야 한다. 그래야만 기존의 터미널 세션을 계속 사용할 수 있기 때문이다. 결론적으로, Load Balancer가 부하분산을 할 수 있는 경우는 오직 사용자가 터미널 세션을 로그오프를 하고 나서 재접속 하는 경우에만 가능하다.

서버에 접속한 사용자들 중 어떤 서버의 사용자는 모두 로그오프 한데 반해 어떤 서버는 모두 사용 중이라면 서버 자원의 활용도는 매우 낮아지게 된다. 또한, 어떤 서버의 사용자들은 많은 컴퓨팅 자원을 소모하는 작업을 수행하고, 어떤 서버의 사용자들은 적은 컴퓨팅 자원을 소모하는 경우에도 서버 자원 활용도는 매우 낮아진다. 또한, 터미널 세션의 부하분산측면에서 세션에 따른 자원 할당이 불공평하게 될 것이다. 이러한 다양한 상황으로 인해 단순 터미널 서비스 클러스터의 보완이 필요하다.

## 4. 적응적 터미널 서비스 클러스터

### 4.1 터미널 서비스 클러스터의 문제점 보완

3장에서 언급한 터미널 서비스 클러스터의 문제점을 보완하기 위해서는 먼저 부하분산 관점에서 시점과 정책이 바뀌어야 한다. 접속 요청 기반의 부하분산이 아니라 런타임 환경에서 동적으로 사용자에게 서버가 할당될 수 있어야 한다. 그러나 터미널 서비스의 동작 특성상 터미널 세션을 동적으로 다른 서버로 옮길 수 없고 부하분산 시점은 바뀔 수 없게 된다. 부하 분산 알고리즘에서 이러한 런타임 상황을 예상하여 이를 반영할 수 있는 지능적 부하분산 필요하다. 현재 상태를 기준으로 보는 방식은 미래를 반영하지 못하는 문제가 있으므로, 통계에 기반을 둔 스케줄링을 통해 지능적 분산이 가능토록 한다.

특정 서버로 부하가 집중되고 다른 서버에는 부하가 적게 되는 부하 분산 패턴은 터미널 서비스 클러스터의 자원 활용도를 낮게 한다. 이것의 원인은 사용자의 컴퓨팅 환경 패턴에 있다. 사용자의 컴퓨팅 환경 패턴은 터미널 서버에서 사용하고자 하는 서비스에 따라 틀리게 된다. 즉, 멀티미디어 서비스나 프로그래밍과 같이 컴퓨팅 자원을 많이 소모하는 사용자와 문서 작성이나 인터넷 검색 등 컴퓨팅 자원을 적게 소모하는 사용자들로 사용자 환경 패턴을 나눌 수 있다. 또한, 모든 사용자에게 공평한 성능을 제공하면서도 특정 사용자에게는 차등화 된 성능을 제공해야 할 것이다. 자원 활용도를 높이기 위해서 사용자에 따라 서버 부하를 예상하고 차등화된 자원을 배정 한다. 이를 위해서 서버를 그룹화 시키고 사용자를 그룹에 따라 분산한다. 사용자의 그룹 서비스화는 터미널 서비스 클러스터의 차등화 서비스도 가능케 한다. 그림 5는 그룹 기반의 부하 분산 개념을 나타낸다. 그룹 #1에 속한 사용자의 접속은 그룹 #1의 클러스터 서버에서 부하분산을 행하게 된다.

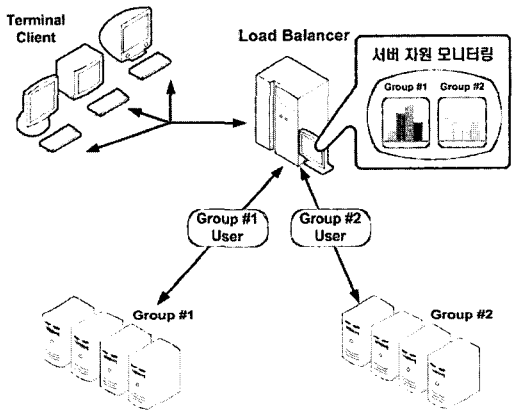


그림 5 그룹 기반의 터미널 서비스 클러스터

그룹 기반의 부하 분산이 가능케 하기 위해서는 Load Balancer가 사용자를 식별하고 사용자의 그룹을 판단하여야 한다. 하지만 터미널 세션이 생성되어야만 사용자 정보를 판단할 수 있으므로, Load Balancer가 원하는 터미널 세션 생성 이전의 판단은 할 수 없다. 이를 위해서는 클라이언트가 그룹에 따라 터미널 서비스 클러스터의 대표 접속 IP인 Virtual IP를 그룹에 따라 선택적으로 접속하는 방식을 취해야 한다. 클라이언트가 자신이 어떤 그룹에 속하는지, 혹은 어떤 VIP를 사용해야 하는지를 알기위해서 먼저 간단한 별도의 UDP 메시지를 Load Balancer로 보낸다. Load Balancer는 이 UDP 메시지의 응답으로 해당 사용자가 접속할 VIP를 알려준다. 터미널 클라이언트는 이 VIP로 접속을 시도한다. 그림 6은 터미널 클러스터 그룹의 VIP 할당을 보여주고, 그림 7은 터미널 클라이언트의 그룹 기반 접속 과정을 보여준다.

### 4.2 적응적 터미널 서비스 클러스터

그룹 기반 터미널 서비스 클러스터를 구성 시에 특정 그룹으로만 사용자가 집중되는 문제가 발생할 수 있다. 이렇게 되면 그룹 #1에는 사용이 별로 없는데 반해 그룹 #2에는 자원 부족 상황이 발생하게 될 수 있다. 이 경우 그룹 #2의 자원 부족 상황을 보충하기 위해 그룹

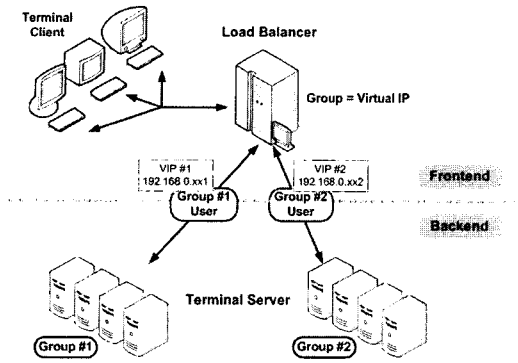


그림 6 터미널 클러스터 그룹의 VIP할당

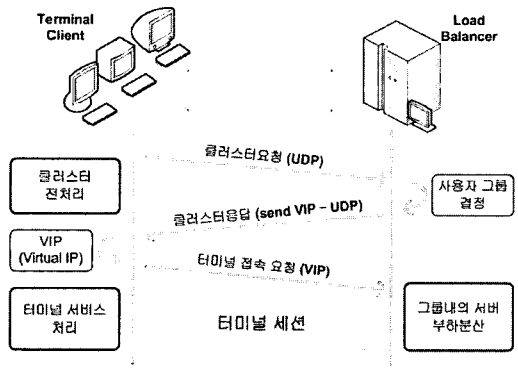


그림 7 터미널 클라이언트의 그룹 기반 접속 절차

#1의 서버를 그룹 #2의 서버로 전환할 수 있다. 이렇게 하기 위해 위해서는 터미널 서버들의 자원 모니터링과 전환 시점의 결정이 필요하다. 적응적 터미널 서비스 클러스터는 이러한 목적으로 제안되고 있다. 터미널 서버들의 자원을 실시간으로 모니터링 하고 룰 기반의 전문가 시스템 로직이 서버 그룹 재설정 시점을 결정하게 된다. 서버 그룹 재설정 시에 재설정되는 서버에 존재하는 활성화 터미널 세션들은 계속 유지된다. 그러나 기존 그룹의 사용자로부터 새 그룹으로 옮겨진 이 서버로의 접속은 제한된다.

그림 8은 터미널 서버들의 자원 상태를 모니터링 하는 구조이고 그림 9는 터미널 서버 그룹의 재 할당을 보여준다.

터미널 서버 그룹 재할당에 대한 구체적인 알고리즘은 다음과 같다. 서버 그룹은 필요한 서비스의 수와 특성에 따라 나눌 수 있다. 분리하여 부하분산이 필요한 서비스 수에 따라 그룹의 수를 늘리고, 각 서버 그룹마다 해당 그룹에서 수행할 서비스에 필요한 서버를 할당한다. 서비스의 필요 자원에 따라 특정 그룹은 다른 그룹에 비해 고성능의 서버를 할당할 수도 있게 된다. 서

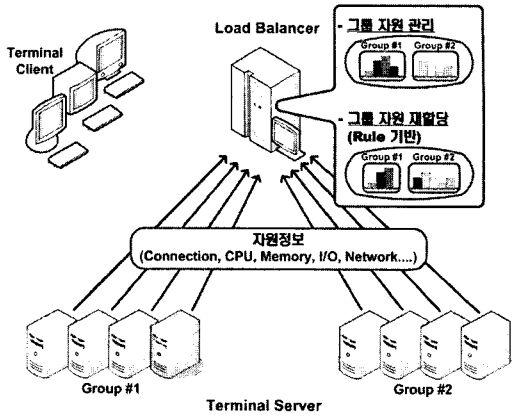


그림 8 자원 정보 모니터링 구조

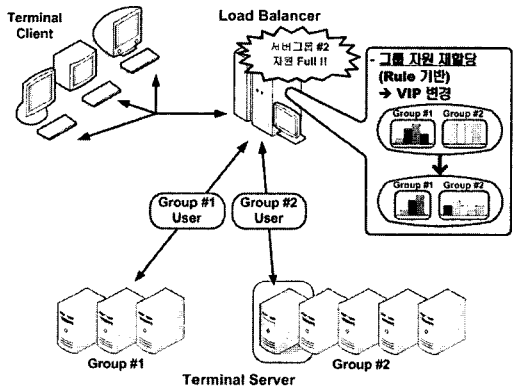


그림 9 터미널 서버 그룹 재할당

버 그룹은 독립된 클러스터로써 Load Balancer에 그림 10과 같은 형태로 저장된다.

그림 10과 같이 서비스 포트 TCP 3389에 대해 두개의 클러스터(그룹)가 존재하고 각 클러스터에는 서버가 4대씩 존재한다. 각 서버의 IP는 서버 ID를 통해 추적할 수 있다. 이와 같은 자료형을 통해 Load Balancer는 서비스와 그룹, 서버와의 관계를 알 수 있다. 적응적 서버 클러스터링 환경에서 클러스터(그룹) 2에서 과부하가 발생하여 그룹 1의 서버를 옮겨갈 경우 자료형은 그림 11과 같이 변경된다. 그림 11은 서버 4가 클러스터 1에서 클러스터 2로 옮겨진 것을 나타낸다.

그림 12는 서버가 두개의 클러스터에 중첩 가능하게 설정될 경우 클러스터 2에 서버 4가 추가된 것을 나타낸다. 서버 4는 클러스터 1에도 존재한다. 이것은 서버 4가 두개의 클러스터에서 서비스를 수행하고 있음을 의미한다.

클라이언트로부터 UDP 메시지를 통해 서비스 클러스터의 요청을 받게 되면 Load Balancer는 UDP 메시지

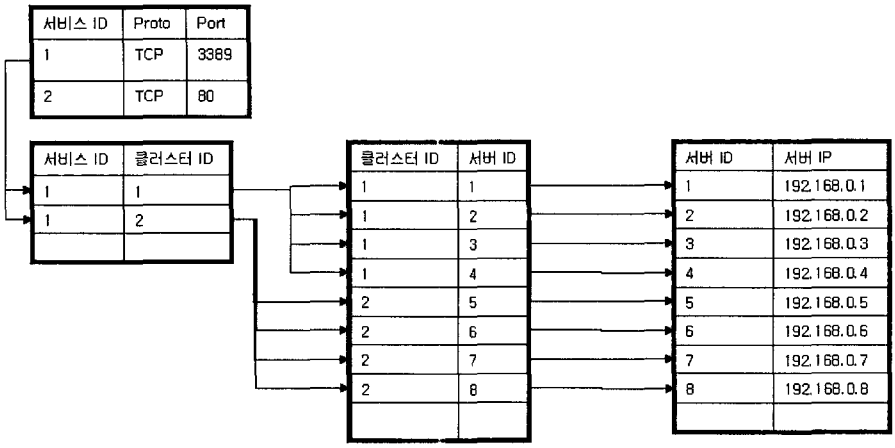


그림 10 Load Balancer에서의 서버 그룹 저장 형태

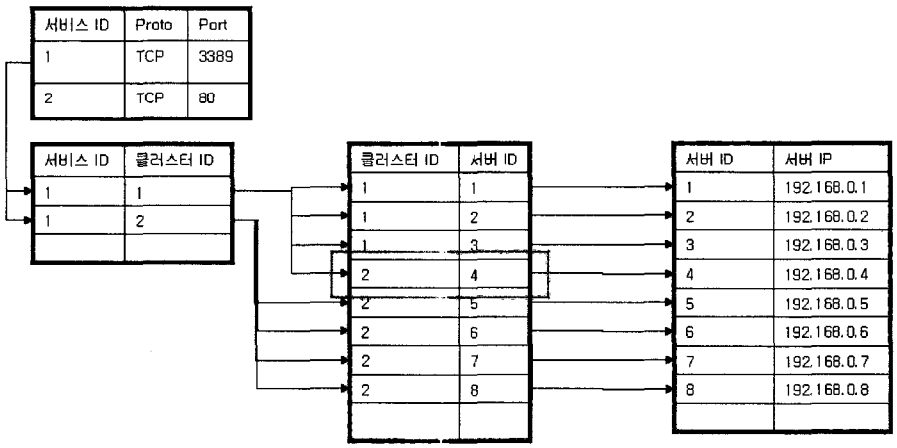


그림 11 그룹 2에서 과부하가 발생하여 그룹 1의 서버가 옮겨간 경우의 자료형 변화

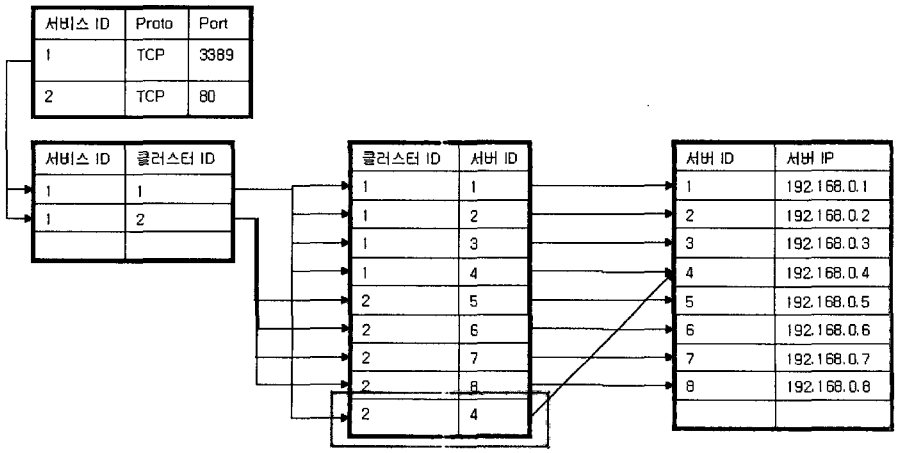


그림 12 하나의 서버가 두 개의 그룹에 중첩된 경우

속에서 사용자의 계정을 확인할 수 있다. 이 계정을 통해 클러스터를 결정하기 위해서는 클라이언트에 대한 그룹 결정을 필요로 한다. Load Balancer는 이 사용자 정보를 얻기 위해 Database 서버로 사용자에 대한 정보를 요청하게 된다. Database에는 아래와 같은 구조의 테이블이 존재한다. Database는 Load Balancer의 요청에 대해 적절한 사용자인지의 여부와 적절한 클러스터 정보를 알려주게 된다. 그림 13은 이러한 상황을 나타낸다.

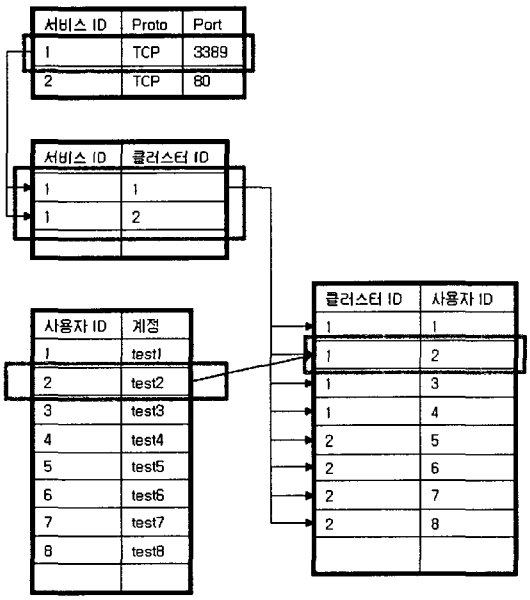


그림 13 사용자 정보를 가진 Database

Load Balancer는 이 정보를 기반으로 사용자의 클러스터 접근 VIP를 UDP 메시지에 대한 응답으로 클라이언트에 보내게 된다. 만약 사용자에 대한 Database로부터의 클러스터 정보가 없게 되면 UDP 메시지를 통해 접속 거부를 알리게 된다. Database에는 해당 사용자의 클러스터 정보가 없을 경우에 접속 가능한 클러스터를 설정할 수도 있다. Load Balancer에서 처리되는 패킷 부하분산은 UDP 메시지 처리와는 별개로 수행되므로 Database로의 요청이 Load Balancer에게 패킷 Block 현상을 초래하지 않는다.

클러스터의 부하 정보를 Load Balancer는 취합하여 특정 클러스터의 부하가 임계값을 넘게 되면 자동으로 Load Balancer의 클러스터 조합 정보를 수정하여 클러스터를 재 할당 하게 된다. 클러스터의 부하 임계값과 클러스터 재 할당 방법을 결정하는 부분에서 지능적인 부분을 사용한다. 예를 들어 클러스터의 부하가 특정 시간동안 유지될 경우 더 이상 기존 자원으로 서비스를 처리할 수 없다고 판단하여 다른 여유 자원을 가진 클러스터로부터 서버를 옮겨오게 된다. 이러한 시나리오는 다양한 조건과 상황이 존재하므로 이를 Rule기반으로 기술하게 된다. 그림 14는 Rule의 간단한 예를 나타낸다.

그림 14의 Rule은 특정 서버의 세션수가 7(임계값)을 넘었을 때 해당 서버가 포함된 클러스터 전체의 세션수가 모든 서버의 임계치 합을 넘었을 때 다른 클러스터로부터 서버를 얻는 것에 대한 기술이다. 위의 룰에서 사용된 서버 변수는 세션 수에 한정되어 있다. 그러나 변수를 서버의 CPU사용량, 메모리 사용량, 디스크 사용량, 네트워크 트래픽 사용량 등 서버로부터 자원 모니터링한 다양한 값으로 사용가능하다.

그림 15는 적응적 터미널 서비스 클러스터를 적용하기 위한 Load Balancer의 내부 기능을 나타낸다. L4 스위칭 모듈은 서버 부하 분산을 수행하고 스위치/클러

```
for(index = 0; index < server_count[cluster_id]; index++) {
    if(session_count[server_id[index]] > 7) {
        // 특정 서버의 세션수가 7을 넘으면
        if(cluster_session_count[cluster_id] > (server_count[cluster_id] * 7)) {
            // 해당 서버가 포함된 클러스터의 총 세션수가
            // 해당 클러스터의 서버수 * 7을 넘으면
            get_backend_except_cluster(cluster_id);
                // 자신의 클러스터를 제외한
                // 다른 서버들로부터 서버를 얻는다.
        }
    }
}
```

그림 14 클러스터 재할당을 위한 Rule의 간단한 예

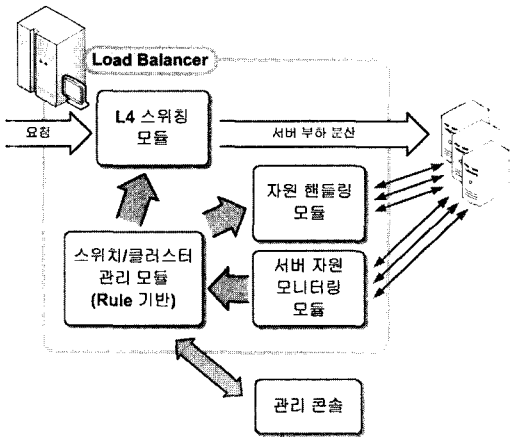


그림 15 적응적 터미널 서비스 클러스터를 위한 Load Balancer의 기능

스터 관리 모듈은 L4 스위칭 모듈 설정하고 Rule 기반으로 동작한다. 서버 자원 모니터링 모듈은 터미널 서버들로부터 자원 정보를 받고 자원 핸들링 모듈은 터미널 서버의 자원을 제어한다.

그림 16은 적응적 터미널 서비스 클러스터를 적용하기 위한 Terminal Server의 내부 기능을 나타낸다. 터미널 서비스 모듈은 Microsoft Windows Terminal Services를 사용하고, 서버 자원 전송 모듈은 터미널 서버의 자원 정보를 Load Balancer로 전송한다. 서버 자원 수집 모듈은 서버가 사용한 CPU, 메모리, 네트워크 트래픽, 디스크 접근, 세션의 수 등을 수집하고, 자원 제어 모듈은 사용자의 자원(CPU, 메모리, 네트워크 대역폭, 디스크 등)을 제한하고 세션을 강제 종료한다.

클러스터 내에서의 부하분산은 Terminal Server로부

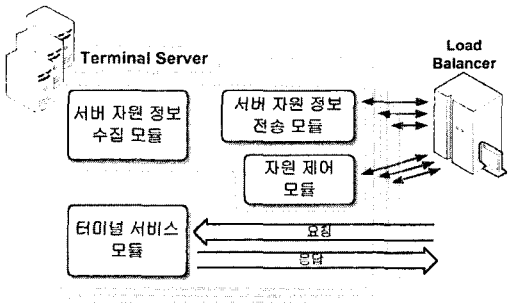


그림 16 적응적 터미널 서비스 클러스터를 위한 Terminal Server 기능

터 수집된 정보를 기반으로 부하분산 하게 된다. 부하분산 알고리즘에도 Rule에 기반을 둔 작성이 가능하고, 이미 정의된 스케줄링 알고리즘을 사용할 수 있다. 이미 정의된 스케줄링 알고리즘의 예는 Least Connection이 있다. 각 서버에 부하 분산된 세션(연결)의 수가 최소가 되는 서버로 부하분산 한다. 그 외에 Rule에 기반을 둔 부하분산의 예는 그림 17과 같다.

그림 17은 해당 클러스터 내에서 CPU 평균 사용량이 최소가 되는 서버를 선택하여 부하 분산하는 Rule의 예이다. 사용 가능한 서버 모니터링 정보는 CPU 사용량, 메모리 사용량, 네트워크 트래픽 사용량, 디스크 사용량, 세션 수, 디스크 접근 빈도 등이다. 이러한 정보를 이용한 적응적(Adaptive/Dynamic) 부하 분산 방식은 기존 부하 분산 방식의 단점, 즉 서버들의 부하 상황을 고려하지 않는 부하 분산을 방식을 해결하고자 제안된 방식이다. 이 기법은 사용자 요청에 따른 부한 분산 시에 서버들의 부하 상황을 고려하여 가장 작은 부하를 가지는 서버에 요청을 분산함으로써 서버 활용도를 높이는 장

```
min_cpu_usage = 100;
min_cpu_usage_server_id = -1;
for(index = 0; index < server_count[cluster_id]; index++) {
    if(cpu_usage_average[server_id[index]] < min_cpu_usage) {
        // 서버의 CPU 평균 사용량이 서버 최소값보다 작으면
        // 해당 서버를 최소 CPU 사용 서버로 선택
        min_cpu_usage = cpu_usage_average[server_id[index]];
        min_cpu_usage_server_id = server_id[index];
    }
}
select_lb_server(min_cpu_usage_server_id);
// 최소 CPU 평균 사용량을 가진 서버를 스케줄링 한다.
```

그림 17 클러스터 내에서 부하 분산 Rule의 예

점을 가진다[23-25]. 또한 서버에 존재하는 자원 제어 모듈에 의해 각 세션 사용자가 서버에서 사용가능한 자원을 제한할 수 있다. Terminal Server에서 제공하는 API를 통해 이러한 동작을 가능케 한다. 이 또한 Load Balancer 의 Rule에 의해 제어한다. 그림 18의 예는 Load Balancer에서 서버의 자원을 조절하기위한 Rule 의 예이다.

그림 18의 예에서 해당 클러스터의 자원이 임계값을 넘어 다른 클러스터로부터 서버를 얻고자 시도하였으나 다른 서버를 얻을 수 없을 때 해당 클러스터의 해당 서버 내에서 자원을 제한하는 방법을 취하고 있다. 해당 서버에 존재하는 세션이 균등히 나눠가질 수 있는 CPU 사용률의 70%를 세션마다 제한하도록 설정하고 있다. Load Balancer의 이러한 명령을 서버에 존재하는 자원 제어 모듈에게 명령을 내려 실제 Terminal Server의 API를 통해 자원을 제한하도록 설정한다.

터미널 서버 자원 제어는 터미널 서버들이 동일한 성능이 아닐 경우 단순히 세션수와 자원 사용 상태로는 세션 간에 동일한 수행능력을 보장할 수 없게 된다. 즉, 터미널 서버의 성능 차이를 무시할 경우 세션마다의 컴퓨팅 능력차이가 발생하여 공평한 사용성을 제공 못하게 된다. 이를 위해 Load Balancer는 터미널 서버들의 성능을 판단하여 그에 따른 가중치를 부여하여 세션들이 모두 동일한 성능을 가질 수 있도록 제어해 주게 된다.

## 5. 실험 및 결과

### 5.1 실험 환경

본 실험은 Load Balancer를 위한 리눅스 환경 외에 모두 Microsoft Windows 환경에서 수행되었다. 모든 네트워크 환경은 100Mbps의 이더넷 환경을 사용함으로 네트워크 병목은 없다고 가정한다. 구축된 적응적 터미널 서비스 클러스터 환경의 평가에 초점을 맞추었기 때문에 다양한 변수 요인을 단순히 부하 요소로 정의하며, 부하에 따라 클러스터가 어떻게 동작하는지 부하의 상대적 변화량을 중심으로 실험을 수행한다.

실험에 사용된 하드웨어 사양과 설치된 소프트웨어는 표 1과 같다. 저성능 터미널 서버는 터미널 서버들 사이에서 성능 차이를 주기 위한 것으로서, 동일 성능 터미널 서버 중 하나를 이 터미널 서버로 대체하여 실험하였다.

```
for(index = 0; index < server_count[cluster_id]; index++) {
  if(session_count[server_id[index]] > 7) {
      // 특정 서버의 세션수가 7을 넘으면
      if(cluster_session_count[cluster_id] > (server_count[cluster_id] * 7)) {
         // 해당 서버가 포함된 클러스터의 총 세션수가
         // 해당 클러스터의 서버수 * 7을 넘으면

         // 자신의 클러스터를 제외한
         // 다른 서버들로부터 서버를 얻는다.
         if(get_backend_except_cluster(cluster_id) < 0) {
             // 다른 클러스터로부터 서버를 얻을 수 없을 때

             temp=get_backend_cpu(server_id[index])/session_count[server_id[index]];
                   // 해당 서버의 세션당 CPU 사용 가능량을 구한다.
             control_backend(server_id[index], SERVER_CPU_LIMIT,
                             ALL_SESSION,  temp - (temp*0.7));
                  // Backend 의 CPU 사용량 제한을 세션이
                  // 사용가능한 CPU 사용량에 70%로 제한한다.
         }          }
    }
}
```

그림 18 Load Balancer에서 서버의 자원을 조절하기 위한 Rule의 예

표 1 실험용 하드웨어 & 소프트웨어

| 구 분 | Hardware | | | Software | # |
|---|---|---|---|---|---|
| | CPU (MHz) | RAM (MB) | HDD (GB) | | |
| Load Balancer | Intel P-Ⅲ 600 | 128 | IDE 20 | Redhat 9.0 (Linux Kernel : 2.4.20-8) | 1 |
| Terminal Server | Intel P-Ⅲ 700 | 512 | IDE 10 | Microsoft Windows 2000 Server, Terminal Services[26] | 4 |
| 저성능 Terminal Server | Intel P-Ⅲ 600 | 128 | IDE 10 | Microsoft Windows 2000 Server, Terminal Services | 1 |
| Directory Server | Intel P-Ⅲ 700 | 512 | SCSI 10 | Microsoft Windows 2000 Server, Active Directory | 1 |
| Storage Server | Intel P-Ⅲ 700 | 512 | SCSI 10 | Microsoft Windows 2000 Server | 1 |

실험은 기본적으로 터미널 클라이언트로 서버에 접속을 하여 별도 제작한 부하 발생기를 사용하여 사용자의 부하를 생성하고, 서버 상태 모니터링을 통해 서버 세션 변화, 자원 활용도를 측정하였다. 부하 발생기는 CPU, 메모리, 하드 디스크, 디스플레이 사용을 반복적으로 수행하여 시스템 부하를 발생시킨다. 터미널 세션이 성립되면 세션마다 부하 발생기를 실행하여 세션 부하로 동작하게 한다.

실험 조건은 터미널 세션의 특징상 세션 연결이 적을수록 부하 요소도 적음으로 부하 분산 알고리즘은 최소 연결(Least Connection) 알고리즘으로 한다. 세션들의 부하 발생은 3단계를 가지고 있으며 각 단계는 최저 부하 크기만큼씩 차이가 나게 한다. 1 단계가 가장 낮은 부하 발생 단계이며 3 단계가 가장 큰 부하 발생 단계이다. 임의의 세션은 이 단계들 중에 한 단계만을 갖는 부하를 발생시킨다. 세션들은 랜덤하게 부하 발생 단계를 가지고 생성된다.

그림 19는 실험 환경을 나타낸다. TS1, TS2가 그룹 #1, TS3, TS4가 그룹 #2로 구성된다.

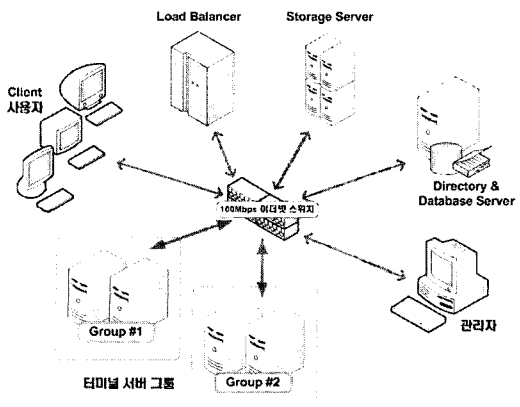


그림 19 실험 환경

## 5.2 실험 결과

본 실험은 터미널 클러스터의 부하 분산과 자원 활용도를 초점으로 하였으며, 4장에서 제안한 시스템을 검증한다. 실험의 진행은 터미널 서버들의 터미널 세션 부하 분산과 자원 활용도를 측정하여 세 개의 클러스터(일반 터미널, 그룹 기반 터미널, 적응적 터미널 서비스)를 비교하였다. 클라이언트의 터미널 클라이언트로 터미널 서버에 접속하여 별도 제작된 부하 생성 프로그램을 실행하고, 터미널 서버들에 대하여 측정한다.

부하 절대 기준량은 부하를 터미널 서버 성능과 관계있는 절대적인 값으로 표현되어 각 터미널 서버들에서 발생하는 부하를 서로 비교할 수 있다. 부하 절대 기준량은 성능이 좋은 환경에서는 상대적으로 낮은 값이 되고, 성능이 안 좋은 환경에서는 상대적으로 높은 값이 된다. 적절한 부하 분산은 터미널 서버들의 성능과 관계없이 터미널 서버들 사이의 부하 절대 기준량이 비슷해야 한다.

- 터미널 서버 부하 절대 기준량 = 세션수 $\Sigma$ (세션 부하량 × 터미널 서버 부하 가중치)
- 터미널 서버 부하 가중치 = 기준 터미널 서버 가중치 / 터미널 서버 성능치

터미널 서버의 성능치는 터미널 서버가 필요로 하는 다양한 기준치를 반영하여 수치화 한다. 터미널 서버 부하 가중치는 해당 서버가 기준 터미널 서버보다 성능이 부족할 때 큰 값을 가진다. 실험에 사용된 터미널 서버들에 사용된 터미널 서버 부하 가중치에서 일반 터미널 서버는 기준 터미널 서버 가중치를 갖게 하여 1이고, 저성능 터미널 서버는 1.5이다. 터미널 서버 부하 절대 기준량을 구하기 위한 세션 부하량은 1단계 부하 세션은 1, 2단계는 2, 3단계는 3을 갖는다.

그림 20은 랜덤 부하 세션을 가지는 일반 터미널 서비스 클러스터에서 터미널 세션 수의 증가에 따른 터미널 서버 간 부하 비교를 나타낸다. 그림에서 보이듯이

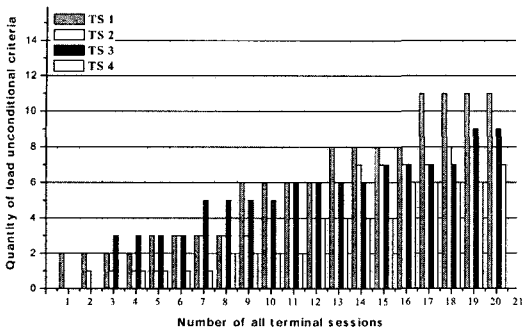


그림 20 터미널 세션 수 증가에 따른 터미널 서버 간 부하 비교 (랜덤 부하 세션, 일반 터미널)

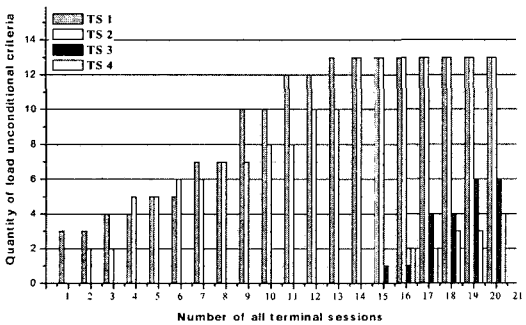


그림 21 터미널 세션 수 증가에 따른 터미널 서버 간 부하 비교 (랜덤 부하 세션, 그룹 기반 터미널)

터미널 서버들의 성능이 동일한 상황에서 부하 절대 기준량이 차이가 있음은 부하 분산이 적절히 이루어지고 있지 않음을 의미한다.

그림 21은 랜덤 부하 세션을 가지는 그룹 기반 터미널 서비스 클러스터에서 터미널 세션 수 증가에 따른 터미널 서버 간 부하 비교를 나타낸다. 그림에서 보이듯이 그룹 #1로 계속적으로 세션이 생성되고 전체 세션수가 15번째 되는 지점부터는 그룹 #2로의 사용자 접속을 시작한다. 즉, 그룹 간에 어떠한 상호 작용 없이 분리된 부하 분산이 이루어지고 있음을 알 수 있다.

그림 22는 동일 3단계 부하 세션과 TS 그룹 2개를 가지는 적응적 터미널 서비스 클러스터에서 터미널 세션 수 증가에 따른 터미널 서버 간 부하 비교를 나타낸다. 그룹 #1의 사용자 세션이 증가함에 따라 그룹 #1 터미널 서버의 세션수가 증가하고 있다. 전체 세션수가 7이 되었을 때 그룹 #1 의 세션수가 임계점에 도달하게 되고 이때 그룹 #2 의 터미널 서버를 그룹 #1 으로 옮겨 TS3을 포함한 부하분산을 수행하고 있다. 터미널 세션수가 10이 되었을 때 TS1, TS2, TS3 모두가 세션수 임계점에 도달함에 따라 TS4 까지도 그룹 #1 으로

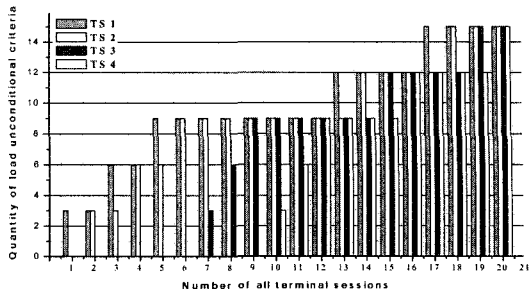


그림 22 터미널 세션 수 증가에 따른 터미널 서버 간 부하 비교 (동일 3단계 부하 세션, TS 그룹 2개)

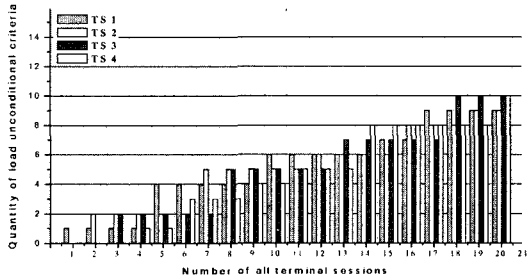


그림 23 터미널 세션 수 증가에 따른 터미널 서버 간 부하 비교 (랜덤 부하 세션, TS 그룹 1개)

옮겨 부하분산을 수행하게 된다. 그룹 간 부하에 따라 적응적으로 그룹을 형성하여 부하 분산이 됨을 알 수 있다.

그림 23은 랜덤 부하 세션과 TS 그룹 1개를 가지는 적응적 터미널 서비스 클러스터에서 터미널 세션 수 증가에 따른 터미널 서버 간 부하 비교를 나타낸다. 그림에서 보는 바와 같이 세션마다 부하 발생량이 다르지만, 동일한 성능의 터미널 서버들에 비슷한 수준의 부하량을 나타내고 있다. 이는 서버 부하량에 따른 부하 분산을 함으로써 가능한 것이다.

그림 24에서 TS1을 저성능 터미널 서버로 하고 랜덤 부하 세션과 TS 그룹 1개를 가지는 적응적 터미널 서비스 클러스터에서 터미널 세션 수 증가에 따른 터미널 서버 간 부하 비교를 나타낸다. 그림에서 보는 바와 같이 랜덤한 부하 세션 상황에서 저성능 터미널 서버가 존재함에도 불구하고 고른 터미널 서버의 성능에 따른 적절한 부하 분산을 하고 있다.

그림 25는 랜덤 부하 세션, 랜덤 그룹 세션, TS 그룹 2개를 가지는 적응적 터미널 서비스 클러스터에서 터미널 세션 수의 증가에 따른 터미널 서버 간 부하 비교를 나타낸다. 실험시 각 그룹의 부하가 증가함에 따라 그룹 간 터미널 서버 재할당이 발생하지 않았다. 그림에서 보는 바와 같이 세션마다 부하 발생량이 다르지만, 동일한

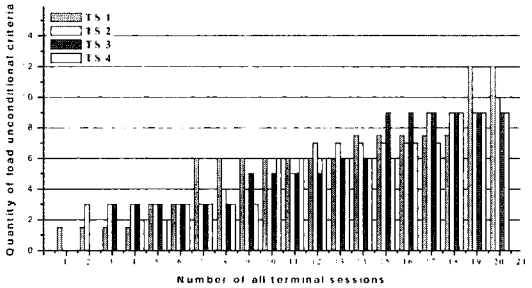


그림 24 터미널 세션 수 증가에 따른 터미널 서버 간 부하 비교 (TS1 저성능 터미널 서버, 랜덤 부하 세션, TS 그룹 1개)

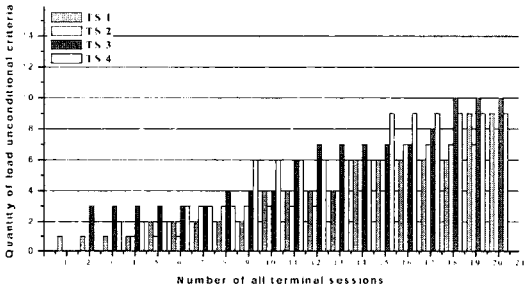


그림 25 터미널 세션 수 증가에 따른 터미널 서버 간 부하 비교 (랜덤 부하 세션, 랜덤 그룹 세션, TS 그룹 2개)

성능의 터미널 서버들에 비슷한 수준의 부하량을 타나내고 있다. 이는 서버 부하량에 따른 부하 분산을 함으로써 가능하게된 것이다.

그림 26은 TS1을 저성능 터미널 서버로 하고 랜덤 부하 세션, 랜덤 그룹 세션, TS 그룹 2개를 가지는 적응적 터미널 서비스 클러스터에서 터미널 세션 수 증가에 따른 터미널 서버 간 부하 비교를 나타낸다. 각 그룹의 부하가 증가함에 따라 그룹간 터미널 서버 재할당이 발생하지 않았다. 사용자의 부하가 랜덤 하므로 저성능 터미널 서버에 대한 세션수 만으로는 저성능 터미널 서버에 대한 적응적 부하 분산을 확인할 수 없다. 그림에서 보는 바와 같이 저성능 터미널 서버에 대해서도 다른 서버와 마찬가지로 부하 절대 기준량이 비슷하게 부하 분산되고 있다.

## 6. 결 론

본 논문에서는 터미널 서비스 서버 환경에서 확장성을 보장해주는 클러스터를 구성하였고, 터미널서비스로 인하여 발생하는 클러스터 비효율성 문제를 해결하기 위해 적응적 터미널 클러스터 구조를 제안하였다. 실험을 통해 제안된 방법이 기존 시스템의 성능 향상에 기여하였음을 검증하였다.

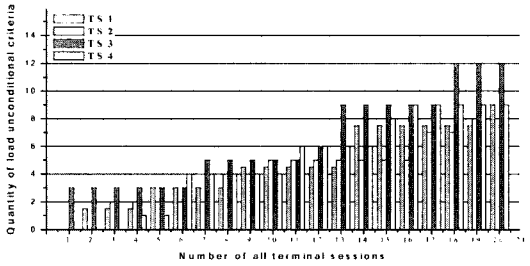


그림 26 터미널 세션 수 증가에 따른 터미널 서버 간 부하 비교 (TS1 저성능 터미널 서버, 랜덤 부하 세션, 랜덤 그룹 세션, TS 그룹 2개)

향후 연구 방향으로는 터미널 세션의 실시간 Migration, 패턴 분석, RDP의 클러스터 최적화 지원 및 터미널 클러스터의 관리 등을 고려해 볼 수 있다.

## 참 고 문 헌

[1] C. Kopparapu, "Load Balancing Servers, Firewalls, and Caches," Wiley Computer Publishing, 2002.
[2] T. Schroeder, S. Goddard, and B. Ramamurthy, "Scalable Web Server Clustering Technologies," IEEE Network, Vol. 14, No. 3, pp. 38-45, 2000.
[3] D. A. Menasce, "Trade-offs in Designing Web Clusters," IEEE Internet Computing, Vol. 6, No. 5, pp. 76-80, 2002.
[4] S. Testa and W. Chou, "The Distributed Data Center: Front-End Solutions," IEEE IT Professional, Vol. 6, No. 3, pp. 26-32, 2004.
[5] A. Volchkov, "Server-based Computing Opportunities," IEEE IT Professional, Vol. 4, No. 2, pp. 18-23, 2002.
[6] R. Buyya, "High Performance Cluster Computing," Vol 1. Prentice-Hall.
[7] S. Jae Yang, Jason Nieh, Matt Selsky, Nikhil Tiwari, "The Performance of Remote Display Mechanisms for Thin-Client Computing," USENIX Annual Technical Conference, 2002.
[8] Citrix MetaFrame, http://www.citrix.com
[9] "Citrix ICA Technology Brief," Technical White Paper, Boca Research, Boca Raton, FL, 1999.
[10] T. W. Mathers and S. P. Genoway, Windows NT Thin Client Solutions: Implementing Terminal Server and Citrix MetaFrame, Macmillan Technical Publishing, Indianapolis, IN, Nov. 1998.
[11] Microsoft Windows Terminal Services, http://www.microsoft.com/terminal
[12] B. C. Cumberland, G. Carius, and A. Muir, Microsoft Windows NT Server 4.0, Terminal Server Edition: Technical Reference, Micorsoft Press, Redmond, WA, Aug. 1999.

[13] Tarantella Enterprise, http://www.tarantella.com
[14] "Tarantella Web-Enabling Software: The Adaptive Internet Protocol," SCO Technical White Paper, Santa Cruz Operation, Dec. 1998.
[15] A. Shaw, K. R. Burgess, J. M. Pullan, and P. C. Cartwright, "Method of Displaying an Application on a Variety of Client Devices in a Client/Server Network," US Patent US6104392, Aug. 2000.
[16] AT&T VNC, http://www.realvnc.com
[17] T. Richardson, Q. Stafford-Fraser, K. R. Wood, and A. Hopper, "Virtual Network Computing," IEEE Internet Computing, Vol. 2, No. 1, pp. 33-38, 1998.
[18] Sheng Feng Li, Q. Stafford-Fraser, and A. Hopper, "Integrating Synchronous and Asynchronous Collaboration with Virtual Network Computing," IEEE Internet Computing, Vol. 4, No. 3, pp. 26-33, 2000.
[19] Sun Ray, http://www.sun.com/sunray
[20] B. K. Schmidt, M. S. Lam, and J. D. Northcutt, "The Interactive Performance of SLIM: A Stateless, Thin-Client Architecture," Proceedings of the 17th ACM Symposium on Operating Systems Principles, Kiawah Island Resort, SC, Dec. 1999.
[21] Xfree86, http://www.xfree86.org
[22] R. W. Scheifler and J. Gettys, "The X Window System," ACM Transactions on Graphics, 5(2), Apr. 1986.
[23] Huican Zhu, Hong Tang, and Tao Yang, "Demand-driven Service Differentiation in Cluster-based Network Servers," IEEE INFOCOM 2001, Vol. 2, pp. 679-688, 2001.
[24] E. Casalicchio and S. Tucci, "Static and Dynamic Scheduling Algorithms for Scalable Web Server Farm," The 9th Euromicro Workshop on Parallel Distributing Processing, pp. 369-376, 2001.
[25] Suntae Hwang and Naksoo Jung, "Dynamic Scheduling of Web Server Cluster," The 9th International Conference on Parallel and Distributed Systems, pp. 563-568, 2002.
[26] M. Minasi, C. Anderson, B. Smith, and D. Toombs, "Mastering Windows 2000 Server," Sybex, 4th Edition, 2002.

정 윤 재
정보과학회논문지 : 정보통신
제 31 권 제 1 호 참조

곽 후 근
정보과학회논문지 : 정보통신
제 31 권 제 1 호 참조

정 규 식
정보과학회논문지 : 정보통신
제 31 권 제 1 호 참조