http://dx.doi.org/10.3745/KTCCS.2013.2.7.283

Dynamic Shutdown of Server Power Mode Control for Saving Energy in a Server Cluster Environment

Hoyeon Kim† · Chihwan Ham†† · Hukeun Kwak††† · Kyusik Chung††††

ABSTRACT

In order to ensure high performance, all the servers in an existing server cluster are always On regardless of number of real-time requests. They ensure QoS, but waste server power if some of them are idle. To save energy consumed by servers, the server power mode control was developed by shutdowning a server when a server is not needed. There are two types of server power mode control depending on when a server is actually turned off if the server is selected to be off: static or dynamic. In a static mode, the server power is actually turned off after a fixed time delay from the time of the server selection. In a dynamic mode, server power is actually turned off if all the services served in the server are done. This corresponds to a turn off after a variable time delay. The static mdoe has disadvantages. It takes much time to find an optimal shutdown time manually through repeated experiments.

In this paper, we propose a dynamic shutdown method to overcome the disadvantages of static shutdown. The proposed method allows to guarantee user QoS with good power-saving because it automatically approaches an optimal shutdown time. We performed experiments using 30 PCs cluster. Experimental results show that the proposed dynamic shutdown method is almost same as the best static shutdown in terms of power saving, but better than the best static shutdown in terms of QoS.

Keywords : Load Balancing, Static Shutdown, Green IT, Energy Saving, Server Power Mode Control, QoS, Dynamic Shutdown

서버 클러스터 환경에서 에너지 절약을 위한 서버 전원 모드 제어에서의 동적 종료

김 호 연† · 함 치 환†† · 곽 후 근††† · 정 규 식††††

요 약

기존 서버 클러스터에서는 고성능을 보장하기 위해, 실시간 요청 수량에 관계없이 모든 서버를 항상 On 한다. 그 방법에서는 QoS를 보장하지만 일부 서버들이 Idle할 때 서버 전력을 낭비하게 된다. 서버들이 소모하는 에너지를 절약하기 위해, 서버가 필요하지 않을 경우 해당 서버의 전력을 Off 하게 하는 서버 전력 제어 방법이 제안되었다. 서버 전력 제어 방법은 서버의 Power가 실제로 어느 시점에 Off 되느냐에 따라 정적인 방법과 동적인 방법이 있다. 정적인 방법에서는 특정 서버가 Off 하기로 결정된 다음 일정 시간 지연 후 그 서버가 Off 된다. 동적인 방법에서는 그 서버에서 수행중인 모든 서비스가 종료된 다음에 해당 서버가 Off 된다. 이는 가변 시간 지연 후 서버가 Off 되는 방법에 해당된다. 정적 종료방식은 단점이 있다. 반복 실험을 통해 수작업으로 최적의 시간 지연을 알아내기 위해서는 많은 시간이 소요된다.

본 논문에서는 정적 종료 방식의 단점을 극복하는 동적 종료 방식을 제안한다. 제안된 방식은 최적의 지연 시간으로 자동적으로 접근하므로 좋은 전력 절약을 하면서 QoS를 보장하는 것을 가능하게 해준다. 30대의 PC 클러스터를 이용하여 실험이 수행되었다. 실험결과는 제안하는 동적 종료 방법이 기존의 정적 종료 방법과 비교할 때 에너지 절감측면에서는 비슷하지만 QoS 측면에서 우수함을 보여준다.

키워드 : 부하 분산, 정적 종료, 그린IT, 전력절감, 서버 전원 모드 제어, QoS, 동적 종료

1. 서 론

서버, 스토리지 및 네트워크 장비들을 직접 운용하는 데이터 센터는 “전기 먹는 하마”로 불릴 정도로 전력 소비량이 많은 곳으로 그린 IT를 실현하는 데 있어 우선적인 고려대상이다. 데이터 센터는 크게 IT 장비와 이를 안정적으로 운용하기 위한 기반 설비로 구성된다. 데이터 센터에서 운용되는 IT 장비는 데이터 센터마다 약간의 차이는 있으나

※ 이 논문은 2012년도 정부(교육과학기술부)의 재원으로 한국연구재단의 기초연구사업 지원을 받아 수행된 것임(NRF-2012R1A1A2006602).
† 준 회 원 : 숭실대학교 정보통신전자공학부 석사과정
†† 준 회 원 : 숭실대학교 정보통신전자공학부 석사
††† 정 회 원 : 퓸킨네트웍스 기술이사
†††† 정 회 원 : 숭실대학교 정보통신전자공학부 교수
논문접수 : 2013년 1월 9일
수 정 일 : 1차 2013년 4월 23일
심사완료 : 2013년 5월 20일
* Corresponding Author : Hukeun Kwak(gobarian@q.ssu.ac.kr)

사용하는 전체 에너지 소비량의 약 50% 이상을 차지한다. 데이터 센터의 기반 시설인 전력과 냉각 설비의 에너지 효율화도 중요하지만 무엇보다도 IT 장비의 에너지 효율화가 중요한 이유는 IT 장비의 에너지 효율이 데이터 센터의 기반 시설에 비해 낮은 편이기 때문이다. 물론 플랫폼에 따라 차이는 있겠지만 서버의 평균 사용률은 20%를 넘지 않는다. 즉, 데이터 센터에서의 IT 장비에 대한 에너지 절감을 상당히 할 수 있다는 결론이다[1].

데이터 센터에서 IT 장비 중 서버들의 에너지 절감을 위해서는 1) 에너지 절감형 서버 하드웨어를 사용하는 방법[2], 2) 멀티코어/멀티프로세서를 사용하는 서버에서 서버의 CPU 전력 소모를 최소화하도록 서버 운영체제에서 에너지 절감형 스케줄링을 사용하는 방법[3], 3) 여러 대의 서버들로 구성된 서버 클러스터에서 서버들 전체의 총 전력 소모를 최소화하도록, 필요하면 일부 서버를 Off하는 에너지 절감형 부하 분산기를 사용하는 방법[4]이 있다.

본 논문은 에너지 절감형 부하 분산기를 사용하는 방법3)에 해당된다. 데이터 센터에는 규모에 따라 수백/수천/수만 대의 서버로 구성되어 있고, 이 서버들은 각기 다른 성격의 서비스가 제공되는 그룹별로 나뉘는데 이 그룹을 각각 서버 클러스터라고 부른다. 서버 클러스터는 규모에 따라 한 대의 부하 분산기가 하나의 서버 클러스터 또는 여러 서버 클러스터의 부하 분산을 담당한다. 부하 분산기가 다수의 서버를 담당하기 때문에 부하 분산기를 사용하여 얻을 수 있는 에너지 절감 효과가 매우 클 것으로 예측할 수 있다.

Fig. 1은 미국 로렌스 버클리 국가 연구소에서 분석한 PC에서 에너지 사용 패턴을 보여준다. PC의 에너지 사용 패턴은 PC가 Idle한 상태에서도 전력 사용률이 60%가 넘고, Sleep 모드에서는 약 5%의 전력 사용률을 보여주고 있다. 또, 상업적인 서버의 사용률은 15~20% 정도로 분석되었다[5]. 상업 서버의 에너지 사용 패턴은 PC의 에너지 사용 패턴과 동일하게 적용하고, 동작중인 서버 클러스터의 시간별 부하량에 따라 다른 부하 분산 정책을 적용할 수 있다. 예를 들어 전체 부하가 많은 시간대에는 모든 서버를 On하여 기존의 서버 클러스터 방식처럼 부하를 균등하게 분산하고, 전체 부하가 적은 시간대에는 필요한 서버만 부하를 균등하게 분산하고, 필요하지 않은 서버들은 Off 함으로써 서버들의 에너지 절감을 할 수 있다.

부하 분산기가 필요한 서버의 용량을 계산해서 서버를 On 또는 Off하는 방법으로 다음의 3가지 다른 방식으로 구현할 수 있다. 즉, 1대의 서버를 시작으로 사용자의 요청이 늘어나면 서버의 개수를 늘리거나(On) 요청이 줄어들면 서버의 개수를 줄이는(Off) 방법[6], 필요한 서버의 용량을 계산해서 필요한 최소한의 서버만을 On하는 방법[7], 제어 이론을 적용하여 사용자의 요청이 많을 때는 클러스터 내의 서버를 On하고, 사용자의 요청이 적을 때는 클러스터 내의 서버를 Off하는 방법이다[8].

또한, 전력 소비량을 근거로 동작될 서버의 수를 결정하여 서버의 전원 모드(On/Off)를 제어하는 방법이 기존 논문에서 연구되었다[9]. 기존 방법의 경우, 서버의 전원을 제어함으로써 전원을 제어하지 않는 일반클러스터가 갖는 전력 낭비의 문제점을 해결하였으나 서버의 전원 모드를 제어하는 시점이 정적(고정)이다. 이에 반해, 본 논문에서 제안된 방법은 동적으로 전원모드 제어 시점을 변경한다. 이는 사용자 요청의 급감/급증에 따라 서버의 전원 모드(On/Off) 제어 시점이 변할 수 있음을 의미한다.

서버를 Off할 때 중요한 것은 언제 서버를 Off할 것인가에 대한 문제이다. 사용자 요청을 모두 처리하는 순간에 Off하여야 하는데 기존 방법의 경우 이 값이 고정되어 있다. 문제는 이 시간이 너무 길면 소비 전력을 절감할 수 없고, 반대로 이 시간이 너무 짧으면 서버가 사용자의 요청에 모두 응답하지 못한 채 종료되기 때문에 사용자 QoS를 보장할 수 없다. 여기서 QoS란 클라이언트의 요청이 서버에서 무사히 처리되어 클라이언트에게 응답하였는가를 의미한다. 서버의 처리가 요청 처리에 실패하여 발생한 Fail의 수가 많을수록 QoS가 낮다고 표현하며, Fail이 발생하지 않는다면 QoS가 보장되었다고 표현한다.

본 논문의 목적은 서버를 Off할 때 최적의 시간을 자동적으로 찾아내는 동적 종료를 사용하는 것이다. 이를 위해 Graceful Shutdown 방법[10]을 에너지 절약을 위한 서버 종료에 적용한다. Graceful Shutdown 방법은 클러스터 상에서 서버들의 사용자 요청 관리를 위해 주로 사용한 방법으로 아직까지 이를 에너지 절약을 위한 서버 On/Off에 적용한 기존 연구는 없다. 본 논문에서 제안하는 동적 종료 방법은 Graceful Shutdown 개념을 클러스터에 최초로 적용하여, 클러스터의 소비 전력을 절감하고 QoS를 보장하는 것을 목표로 한다.

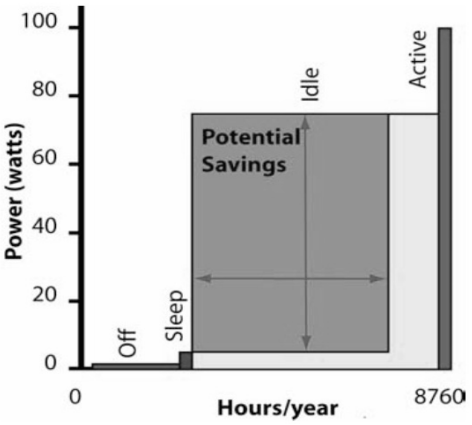


Fig. 1. The usage pattern of PC energy

본 논문의 구성은 다음과 같다. 2장에서는 에너지 절약을 위한 서버 전원 모드 제어에서 기존의 정적 종료 방법에 대한 연구와 문제점을 소개한다. 3장에서는 기존의 종료 방법의 문제점에 대한 해결 방법인 동적 종료에 대해 소개한다. 4장에서는 실험 및 토론을, 5장에서는 결론 및 향후 연구 방향을 제시한다.

2. 연구 배경

2.1 기존의 정적 종료 방법[9]

일반적인 서버 클러스터는 실시간 요청 수량에 관계없이 정해진 서버 수를 항상 최대로 운영한다. 이는 QoS를 보장하지만, 소비 전력 효율이 낮은 문제가 있다. 이 문제를 해결하기 위해 서버의 소비 전력에 근거하여 동작에 필요한 서버 수를 결정하고 서버의 전원을 제어하는 방법이 "정적 종료 방법"이다. 정적 종료 방법은 클러스터의 소비 전력 효율 향상과 QoS 보장을 목표로 한다.

Fig. 2는 종료 시스템의 동작 구조를 나타낸다. Fig. 2에서 일반적인 클러스터에 서버 On/Off 기능이 추가되었음을 확인할 수 있다. 부하 분산기인 LVS(Linux Virtual Server)는 정해진 스케줄링 방식에 따라 클라이언트의 요청을 클러스터내의 서버들에게 분산 하고, 서버의 전원을 제어하는 역할을 한다.

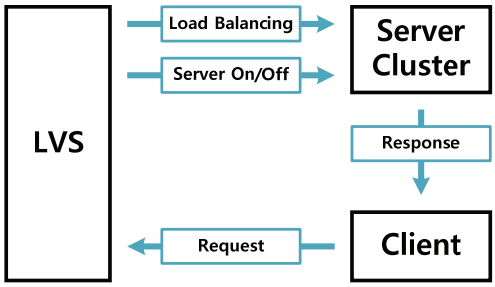


Fig. 2. The functional structure of existing system

기존의 정적 종료방법에서, LVS(부하 분산기)는 각 서버로부터 소비전력 정보를 수집하며, 이 정보를 근거로 하여 필요한 서버의 수를 결정하고 서버들의 전원을 제어한다. 또한, 각 서버의 소비 전력에 근거하여 부하 분산을 수행한다.

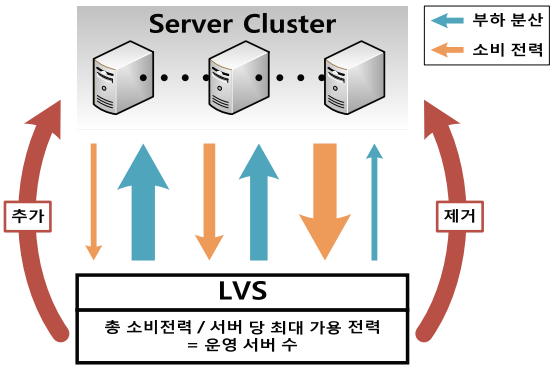


Fig. 3. The load balancing of the existing system

정적 종료 방법의 동작 과정은 다음과 같다.

(1) 운영 중인 각 서버에 대하여, 소비 전력이 많은 서버에게 부하를 적게 분산하고, 소비 전력이 적은 서버에게 부하를 많이 분산한다.

(2) 총 소비 전력, 현재 운영 중인 서버의 수, 서버 당 최대 가용 전력을 근거로 하여 서버의 전원 모드 제어 여부를 결정한다. 총 소비 전력을 서버 당 최대 가용 전력으로 나누어 운영할 서버의 수를 결정하며, 서버의 추가/제거를 결정하게 된다.

(3) 서버의 추가는 전원 모드를 Off 모드 에서 On 모드로 전환한다는 의미이다.

(4) 서버의 제거는 전원 모드를 On 모드에서 Off 모드로 전환한다는 의미이며, 서버가 부하 처리를 완료하도록 미리 정해진 x초 동안 대기한 후에 Off 모드로 전환한다.

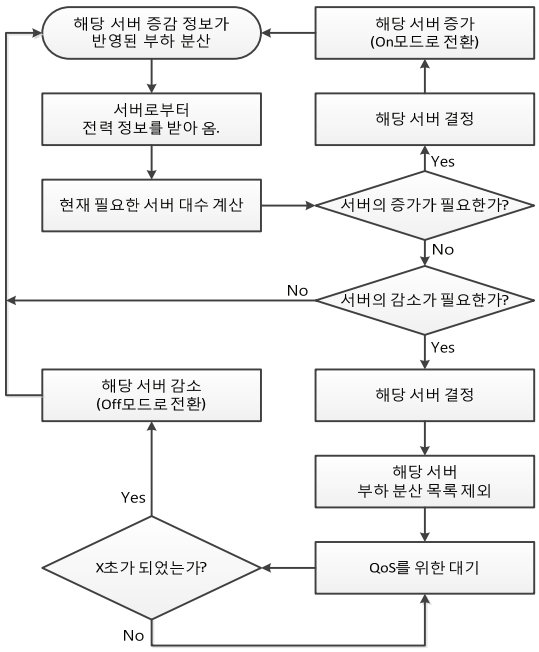


Fig. 4. The operation flow of existing system

기존의 "정적 종료 방법"과 본 논문에서 제안하는 동적 종료 방법의 가장 큰 차이점은 최적의 종료 시간의 변동가능 여부이다. 기존의 방법은 실험을 통하여 최적의 종료 시간(예: 90초)을 미리 결정하여 적용하는 반면, 본 논문에서 제안하는 동적 종료 방법은 클라이언트의 요청을 서버가 모두 처리 완료하였는가 여부에 근거하여 동적으로 최적 종료 시간을 결정한다.

2.2 기존 연구의 문제점

정적 종료 방법의 문제점을 확인하기 위해 실험을 수행하였다. 실험은 30대로 구성된 서버 클러스터 환경에서 기존 클러스터 방법과 비교하여 실행되었다. 기존 클러스터는 30대의 서버가 모두 동작되는 상황이고, 정적 종료 방법은 소비 전력에 따라 전원 모드가 제어되는 상황이다. 실험은 각 120분간 진행되었고, 요청 패턴은 SPECweb(클라이언트에서 서버로 부하를 생성하기 위한 벤치마크 툴)의 Banking Design[9](실제 사용자가 은행 업무를 보는 패턴을 그대로 시뮬레이션)을 사용하였다.

서버들의 소비 전력에 따라 동작될 서버의 전원 모드를 제어함으로써 소비 전력의 효율성을 높이는 기존의 연구 방법은 전원 모드가 On 모드에서 Off 모드로 변경될 때 바로 실행되는 것이 아니고, 기존에 연결된 부하 처리를 위한 시간을 모두 기다린 후 실행하게 된다. 그 시간은 여러 번의 실험을 통해 90초로 정해졌고, 이 방법으로 인해 기존 클러스터와 같은 성능을 유지하면서, 소비 전력은 29% 절감하게 되었다. 물론 이 방법은 기존 클러스터보다 우수한 소비 전력 절감 효과를 내지만, 서버에 연결된 부하 처리 시간이 x초로 정해져 있다는 단점이 있다. x초를 90초라고 가정하고 그 단점의 예를 들면, 부하 처리를 90초보다 빨리 끝낸 서버는 90초까지 기다리는 동안의 소비 전력을 낭비하게 된다. 또는 부하 처리를 90초보다 늦게 끝내야 하는 서버는 부하 처리를 끝까지 마치지 못하고 종료가 됨으로써 QoS를 보장할 수 없다.

Fig. 5는 앞서 설명한 정적 종료의 문제점을 나타내며, 각각을 예를 들어 설명하면 다음과 같다. (1) 실제 부하 처리 필요 시간과 고정된 부하 처리 필요 시간이 같을 때이며, 이 상황에서는 아무 문제가 없다. (2) 실제 부하 처리 필요 시간이 고정된 부하 처리 필요 시간보다 클 때이다. 이 상황에서는 실제 부하 처리 필요 시간이 더 많기 때문에 90초만에 서버를 종료하게 되면 QoS를 보장할 수 없다. (3) 실제 부하 처리 필요 시간이 고정된 부하 처리 필요 시간보다 작을 때이며, 이 상황에서는 부하 처리를 다 끝냈는데도 90초가 되기 전까지 어떠한 처리를 하지 않은 채 소비 전력을 낭비하게 된다.

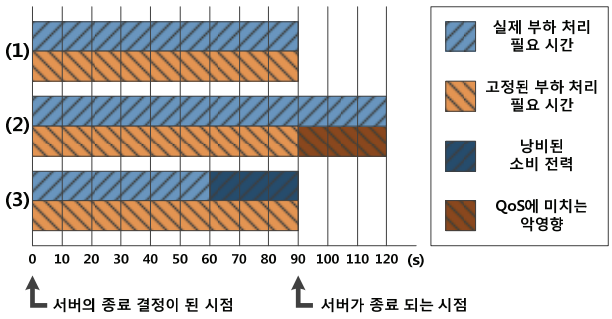


Fig. 5. Server shutdown problems

본 논문에서는 상기의 문제점을 보완하기 위한 동적 종료 방법을 제안한다. 동적 종료 방법은 상황에 따라 부하 처리 필요 시간을 동적으로 운영함으로써 소비 전력과 QoS를 모두 만족시키는 것을 목적으로 한다.

3. 제안하는 동적 종료 방법

3.1 시스템 구조

본 논문에서 제안하는 시스템의 구조는 기존의 시스템과 라우팅 방식이 같고, Fig. 6과 같이 동작을 한다. 서버와 클라이언트 사이에는 LVS가 존재하고, 클라이언트가 LVS에게 요청을 하면 LVS는 서버들에게 부하 분산을 하게 된다. 그리고 부하 분산을 할당받은 서버는 각 클라이언트에게 요청에 대한 응답을 해줌으로써 연결을 유지하게 된다. 여기까지는 기존의 시스템과 같고, 본 논문에서 제안하는 시스템은 각 서버에서 ESTABLISHED 값을 LVS로 전송함으로써, 실제 서버가 클라이언트와 연결되어 있는 정보를 알 수 있도록 수정하였다.

본 논문에서 제안하는 시스템에서 ESTABLISHED 값을 쓰는 이유는 LVS에서 라우팅 방식으로 DR을 사용하여 서버의 실제 연결 상태를 알 수 없기 때문이다. Fig. 6에서 보는 바와 같이 라우팅 방식에는 DR(Direct Routing) 방식과 NAT(Network Address Translation) 방식이 있는데, DR 방식은 클라이언트의 요청을 서버가 Direct로 클라이언트에게 응답하는 방식이다. 이런 이유로 LVS는 서버의 연결 상태를 자세히 알 수 없다. 만약 NAT 방식처럼 라우팅 방식이 클라이언트의 요청을 LVS를 통해 응답하는 방식이면 연결 상태는 알 수 있지만, LVS를 거치는 동안의 병목 때문에 높은 성능의 네트워크를 구성할 수 없다. 본 논문에서 제안하는 시스템 구조는 Fig. 7에서처럼 서버와 클라이언트의 연결 상태가 LVS와 독립되어 있는 것을 볼 수 있다(DR 방식).

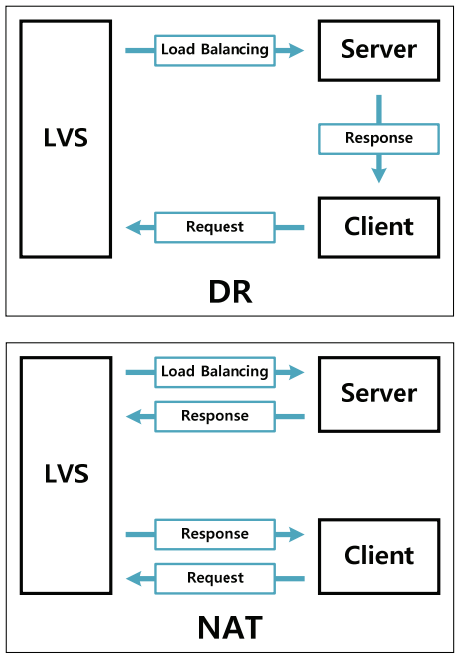


Fig. 6. Routing methods

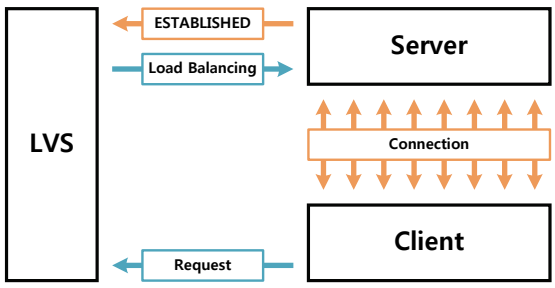


Fig. 7. The functional structure of proposed system(DR)

Fig. 8과 Fig. 9는 동시간대 실행된 본 논문에서 사용한 두 모니터 프로그램의 캡쳐 화면이다. Fig. 8은 LVS에서 제공하는 기본 모니터 프로그램이다. 이 모니터 프로그램은 LVS에서 관리되는 각 서버의 부하 분산 알고리즘 방식, Weight(가중치), ActiveConn, InActConn 값을 모니터 할 수 있다. Fig. 8을 보면 WRR(Weighted Round Robin) 방식(가중치를 가지고 RR을 적용)으로 부하 분산을 하고 있고, 각 서버마다 Weight가 다르게 설정되어 있는 것을 볼 수 있다. 또한 ActiveConn값은 현재 서버와 클라이언트가 연결된 값을 말하고, 이 값은 네트워크 연결 상태가 ESTABLISHED 된 수를 가리킨다. 그 외에 다른 네트워크 연결 상태(CLOSED, LISTEN, SYN_SENT 등)는 InActConn이 된다.

Fig. 9는 본 논문에서 제안하는 시스템을 위해 개발된 모니터 프로그램이다. 이 모니터 프로그램에는 LVS가 관리하는 각 서버의 소비 전력 정보, 서버의 전원 모드 상태, 각 서버의 실제 ESTABLISHED 값을 모니터 할 수 있다. Fig. 8의 첫 번째 PC와 Fig. 9의 PC 1은 같은 서버이다. Fig. 8의 첫 번째 PC의 ActiveConn값이 실제 연결되어 있는 값을 뜻한다. 또, Fig. 9의 PC 1의 ESTABLISHED 값이 실제 연결되어 있는 값이다. 따라서, 두 값은 같아야 하지만 다르다. 즉, DR 방식으로 구성된 라우팅 방식에서의 LVS는 서버의 실제 연결 정보를 알 수 없다는 것을 의미한다[11].

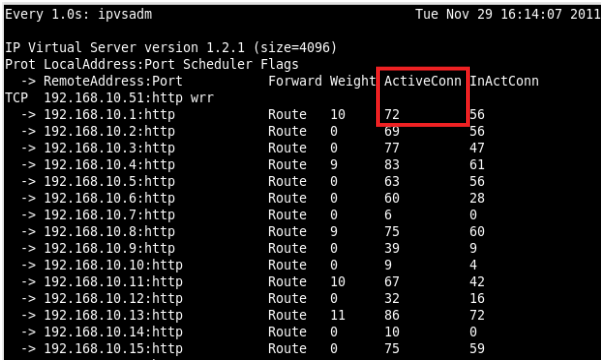


Fig. 8. The basic monitor program for LVS

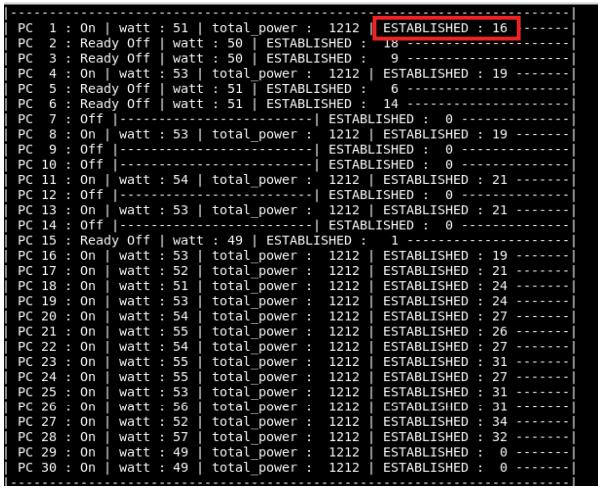


Fig. 9. The monitor program for dynamic shutdown

3.2 동작 과정

우선 QoS를 만족하기 위해서는 각 서버의 연결되어진 부하 처리가 모두 끝났을 때 종료되어야 한다. 그러기 위해서 각 서버의 네트워크 상태 정보를 LVS가 알아야 종료될 서버의 동적인 종료를 할 수 있다. 실제로 서버에서는 "netstat"이라는 명령어를 통해 서버의 실제 연결 개수인 ESTABLISHED 정보를 알아낼 수 있다. 즉, ESTABLISHED 정보가 없다면 모든 부하 처리를 마친 상태이기 때문에 종료를 해도 QoS를 보장할 수 있다.

서버의 네트워크 상태 정보를 이용하여 종료 시점을 결정하는 방법은 다음과 같다.

(1) 각 서버의 상태 정보 중 ESTABLISHED 정보를 LVS로 매초 전송한다. 이 정보는 서버가 클라이언트의 요청에 대한 부하 처리의 개수 즉, 실제로 연결된 개수를 뜻하며, 연결 중인 서버에서는 다수의 연결이 존재한다.

(2) LVS는 종료해야 할 서버를 선택하고, 그 서버의 연결 정보를 계속 체크하면서, 종료 시점을 기다린다.

(3) 선택된 서버의 연결 정보 즉, 연결된 개수가 0이 되는 시점에 서버를 종료하게 된다. 이 때, 시간이 얼마나 걸리든지 QoS는 만족하게 된다.

Fig. 10은 본 논문에서 제안하는 시스템의 동작 흐름이다.

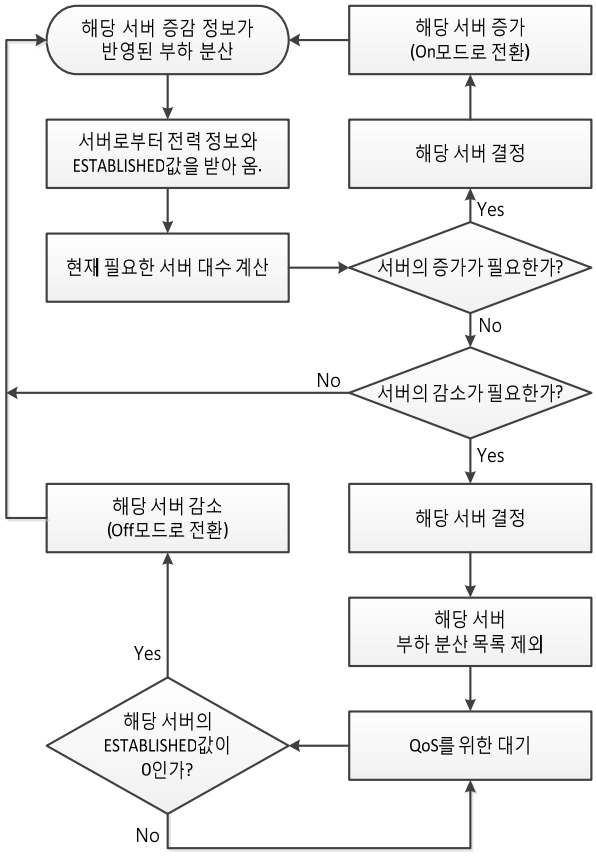


Fig. 10. The operation flow of proposed system

4. 실험 및 토론

4.1 실험 환경

Fig. 11은 실험 전체 구성을 나타낸다. 전력 측정기는 전체 Real Server(실제 사용된 서버)의 전력 값을 측정하여, Zigbee(무선 통신 프로토콜)를 통해 전력 측정 PC로 전송한다. Switch 2대를 이용하여 Real Server 1, Real Server 2, Client(Prime Client), Besim을 각각 분리된 네트워크를 구성하였다. Prime Client는 클라이언트가 여러 대일 때 이를 관리하고, Besim은 Backend Simulator의 약자로 클라이언트의 동작(로그인, 계좌 이체 등)을 제어한다. Switch-LVS-Client간의 네트워크 연결은 1Gb Ethernet을 사용하였고, 나머지는 100Mb Ethernet을 사용하였으며, 네트워크 구성은 Direct Routing(서버가 LVS를 거치지 않고 클라이언트에 직접 응답)을 사용하여 네트워크 내 병목을 최소화하였다.

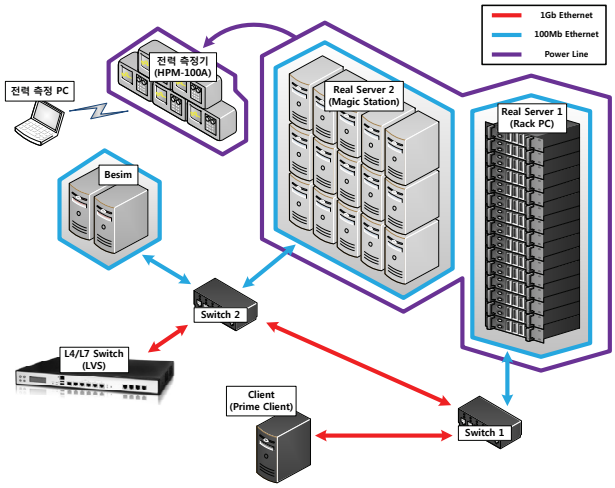


Fig. 11. The overall configuration of experiments

Table 1은 실험에 사용된 하드웨어와 소프트웨어를 나타낸다. 부하 분산 방식은 Round Robin(순서대로 요청을 분배)과 Weighted Round Robin(가중치를 두어 RR을 적용) 방식을 사용하였고, SPECweb의 Client와 Prime Client 패키지를 1대에 설치하였다. 웹 서버는 Apache(오픈 소스 웹 서버)를 사용하였다.

Table 1. Hardware and software for experiments

		Hardware		Software	#
		CPU (Hz)	RAM (MB)		
LVS		Dual Core 2.93G	DDR3 2048	RR/WRR	1
SPEC web	Client (Prime Client)	Quad Q6600 2.4G	DDR2 4096	SPECweb	1
	Besim	P-IV 1.8G	DDR 512	Apache/ SPECweb	2
	Real Server 1	P-IV 2.26G	DDR2 2048	Apache	15
	Real Server 2	P-IV 1.8G	DDR 512	Apache	15

4.2 실험 방법

각 실험은 Table 2와 같이 진행되었으며, 요청 패턴은 SPECweb의 Banking Design[12]을 이용하였고, 실험은 각각 60분간 20번씩 실험하였다. SPECweb에서는 이외에 E-Commerce Design도 제공하지만 비슷한 실험 결과가 예상되므로 실험은 Banking Design으로 한정하였다[9]. 요청 패턴은 InternetTrend™[13]의 카테고리별 방문특성 자료 중 금융권(Fig. 12)의 시간대별 집중도를 참고하여 패턴을 생성하였다. 실험 진행은 기존 클러스터 시스템인 RR과 정적 종료 시스템의 3가지 종료 시간이 적용된 WRR-static(90), WRR-static(60), WRR-static(120) 순서로 각각 실험 하였고, 제안하는 동적 종료 시스템 WRR-dynamic의 순서로 진행되었다. 기존 클러스터 시스템인 RR 방법은 모든 서버를 On하여 실험을 실행하였다.

이 실험에서 60초와 120초를 정한 이유는 WRR-static (90)의 90초는 많은 실험을 통한 최적의 종료 시간이며, 60초는 서버가 클라이언트의 요청에 대한 응답을 모두 마치지 못한 상황을 의미하고, 120초는 클라이언트의 요청에 대한 응답은 모두 마쳤지만, 전원 모드가 On모드로 대기하면서 소비 전력이 낭비되고 있는 상황을 의미한다.

Table 2. Experiments procedure

System	Load processing time (s)	Request pattern	Time (min)	Count
RR	X	SPECweb Banking Design	60	20
WRR-static(90)	90			
WRR-static(60)	60			
WRR-static(120)	120			
WRR-dynamic	Dynamic			

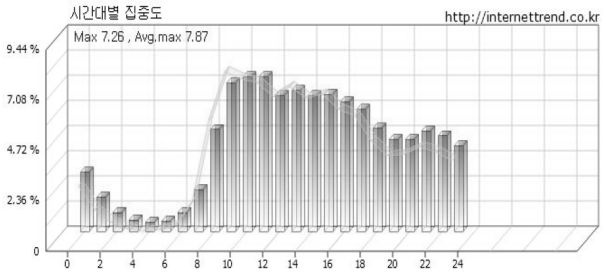


Fig. 12. Banking Pattern
(http://trend.logger.co.kr/trendForward.tsp)

4.3 실험 결과

Table 3은 각 실험의 전체 결과이다. 표에서 Total Request는 Client가 요청한 총 요청 개수를 의미하고, Fail은 요청에 대한 응답이 없음을 의미한다. 또한 Time Tolerable은 응답은 했지만 지정된 시간 안에 하지 못했음을 의미한다.

각 실험은 20회 수행되었으며, 그 결과의 평균은 Table 3과 같다.

Table 3. The overall experimental results

Algorithm	Cumulative power consumption	Time of server shutdown	Number of server shutdown	Average response time	Total Request	Time Tolerable	Fail	Average CPU usage
RR	2235	x	x	0.356	840473	3	0	18.09
WRR-Static(90)	1704	90	42	0.359	839010	38	66	27.21
WRR-static(60)	1614	60	47	0.386	801632	67	3444	26.93
WRR-static(120)	1732	120	36	0.358	840147	18	0	27.38
WRR-dynamic	1697	79	43	0.359	840009	27	0	27.21

Table 4. The mean value of RR, WRR-static(90), and WRR-dynamic

System	Load processing time(s)	Cumulative power consumption (wh)	Time Tolerable	Fail	Number of finish	Time reducing
RR	x	2235	3	0	x	x
WRR-static(90)	90	1704	38	66	42	3780
WRR-dynamic	79	1697	27	0	43	3397

Fig. 13은 RR, WRR-static(90), WRR-dynamic의 누적 소비 전력 평균값을 그래프로 비교한 것이고, Fig. 14의 (a)와 (b)는 RR, WRR-static(90), WRR-dynamic의 Time Tolerable과 Fail의 총 개수를 비교한 것이다. 또한 Table 4는 RR, WRR-static(90), WRR-dynamic의 실험 데이터를 평균한 값이다. 본 논문에서 제안하는 WRR-dynamic은 소비 전력 절감 부분에 있어서 기존 클러스터 시스템 RR보다는 WRR-static(90)와 마찬가지로 만족된 성능과 소비 전력을 절감하는 효과를 볼 수 있었다. 하지만 WRR-static(90)보다 소비 전력 절감 효과는 크지 않다. 이유는 기존의 현재의 테스트 패턴이 정적 종료 시스템에 최적화(90초)되어 있기 때문에 별 차이가 없는 것이다. 하지만 Fig. 14의 (a)와 (b)를 보면 Time Tolerable과 Fail의 총 개수가 WRR-static(90)에 비해 적은 것을 볼 수 있고, QoS에 직접적인 영향을 주는 Fail이 0으로 나오는 것을 볼 수 있다. 그러므로 제안된 방법은 사용자 QoS를 확실하게 보장함을 알 수 있다.

Fig. 15는 WRR-static(90)과 WRR-dynamic의 누적 소비 전력과 Time Tolerable의 분포도를 보여주고 있다. 또, Fig. 16은 WRR-static(90)과 WRR-dynamic의 누적 소비 전력과 Fail의 분포도를 보여주고 있다. Time Tolerable의 경우에는 두 방식의 차이가 한눈에 띄지는 않는다. 하지만 Fail의 경우는 한눈에 봐도 WRR-dynamic의 Fail수가 모두 0으로 되어있다는 것을 볼 수 있으므로, WRR-dynamic의 QoS는 훨씬 뛰어나다는 것을 볼 수 있다.

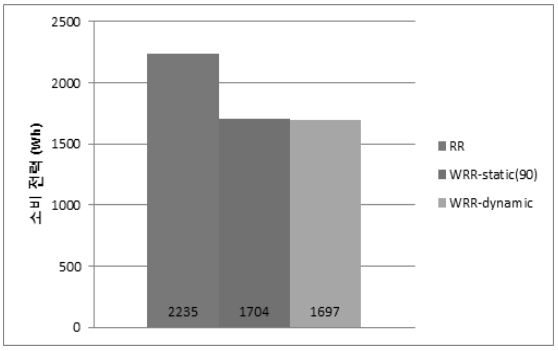
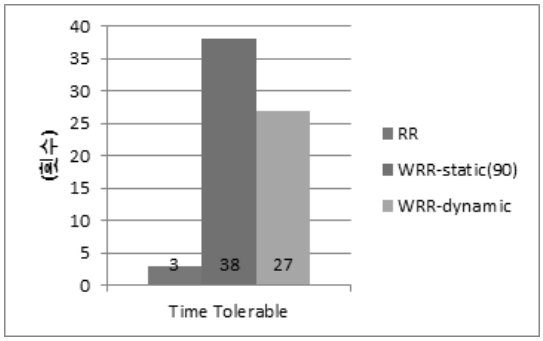
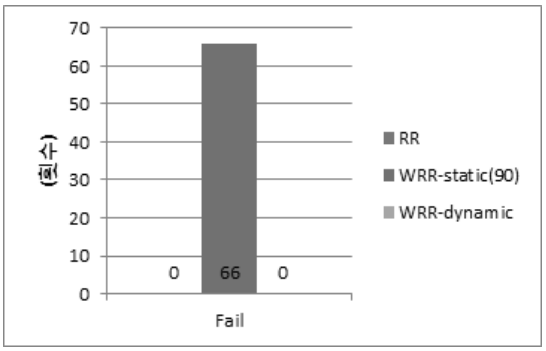


Fig. 13. The average cumulative power consumption of RR, WRR-static(90), and WRR-dynamic



(a) Time Tolerable



(b) Fail

Fig. 14. The "Time Tolerable" and "Fail" of RR, WRR-static(90), and WRR-dynamic

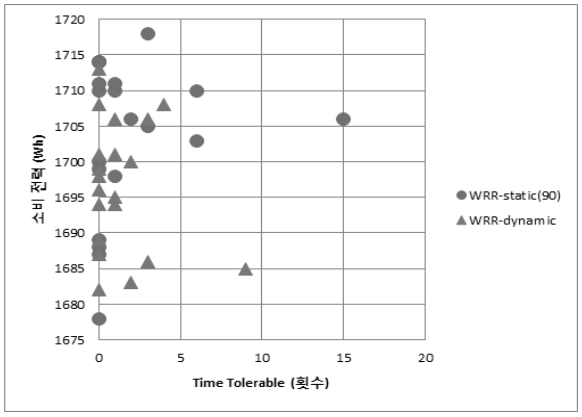


Fig. 15. The power consumption of WRR-static(90) and WRR-dynamic and the distribution of "Time Tolerable"

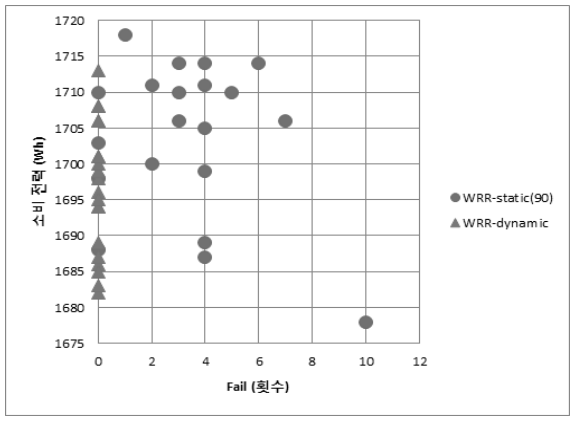


Fig. 16. The power consumption of WRR-static(90) and WRR-dynamic and the distribution of "Fail"

4.4 분석 및 토론

본 논문에서 제안하는 동적 종료 시스템의 실험 결과 중에서 실제로 종료할 때 필요로 하는 시간에 대한 데이터를 Fig. 17에서 그래프를 통해 나타내어 분석해 보았다. Table 5는 WRR-dynamic의 20번 실험 중 첫 번째 데이터에 대한 분석이다. 데이터 분석 결과 총 46번의 서버 종료가 있었고, 60초 이하와 120초 이상에 종료되는 서버는 많지 않다. 또한 90초 이상 종료되는 서버도 많지 않지만, 90초에 종료를 하게 된다면 5개의 Fail이 날 수 있다. 하지만 앞서 말했듯이 최적화된 시간 90초는 거의 모든 서버의 종료 시간을 만족한다. 본 논문에서 제안된 동적 종료 시스템은 90초보다 빨리 종료된 서버들의 소비 전력을 절감할 수 있고, 90초보다 늦게 종료된 서버들의 QoS를 보장할 수 있는 장점이 있다.

Table 5. The analysis for the time of server shutdown

Seconds	Count	Total Count
Less than 60	2	46
More than 90	5	
More than 120	0	
Etc	39	

본 논문에서 제안된 동적 종료 시스템은 기존의 정적 종료 시스템에 비하여 소비 전력 절감 및 QoS를 보장할 수 있다. 기존 정적 종료 시스템은 미리 정해진 시간에 도달하면 요청 처리가 완료되지 않았더라도 서버의 전원을 종료하여 QoS의 손실이 발생할 수 있다. 또한, 모든 요청 처리가 완료된 상황에서 정해진 종료 시간을 기다리는 것은 전력의 낭비를 초래한다. 이와 달리, 본 논문에서 제안한 동적 종료 시스템은 종료 대상인 서버의 요청 처리 여부를 확인하여, 모든 요청 처리가 완료된 시점에서 서버를 종료하므로 QoS를 보장함과 동시에 전력의 낭비를 방지할 수 있다.

또한, 서버의 성능은 요청 처리에 소요되는 시간과 연관되므로, 기존 정적 종료 시스템의 서버 종료 시점 시간단위(예:90초, 120초 등)는 서버의 성능에 따라 유동적이다. 따라서 기존의 정적 종료 시스템은 최적의 종료 시점을 알기 위해서 많은 실험을 필요로 한다. 이에 반하여, 제안된 동적 종료 시스템은 요청 처리의 완료 여부를 확인하여 최적의 서버 종료 시점을 결정하므로 서버의 성능에 독립적으로 동작이 가능하며 최적의 종료 시점을 알기 위한 실험이 불필요하다는 장점을 갖는다. Table 6에 동적 종료 시스템과 정적 종료 시스템을 비교정리 하였다.

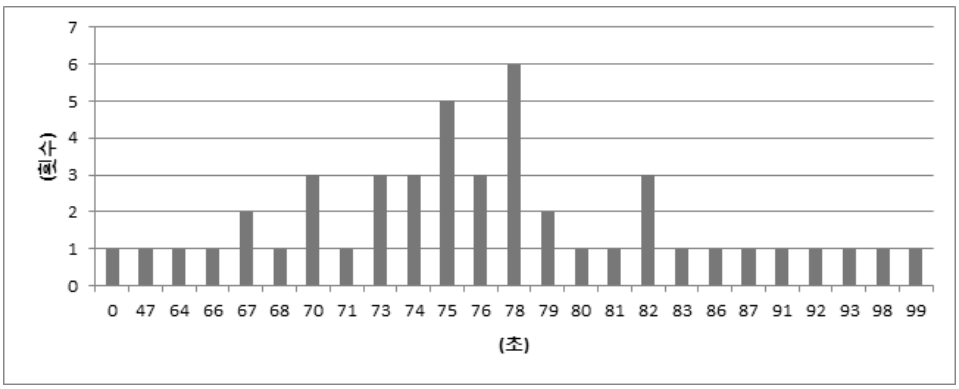


Fig. 17. The time of server shutdown

Table 6. WRR-dynamic vs. WRR-static

Feature	WRR-dynamic	WRR-static
Server shutdown time	dynamic	static
Criteria of server shutdown time	when the request completed	fixed time through experiments
Server dependency	independent	dependent
QoS(# of Fails)	supported(0)	not supported(N)

기존의 정적 종료 방법에서, 최적의 종료 시간은 QoS와 소비전력 절감 사이의 트레이드오프 관계를 갖는다. 서버가 잔여 부하를 처리하는데 90초가 소요된다고 가정한다면, 90초를 기준으로 설정된 최적의 종료시간이 짧아질수록 소비전력의 절감량은 증가하지만 QoS는 감소하게 된다. 예를 들어, 설정된 최적의 종료 시간이 60초일 경우, 90초 보다 30초 더 일찍 서버가 종료되게 되므로, 그 시간동안의 전력이 절감 된다. 하지만, 서버가 60초 동안 모든 요청을 처리하지 못할 것이므로 요청에 대한 Fail이 발생할 것이다. 서버의 성능은 Fail의 수에 영향을 줄 것이다. 다른 예로, 90초를 기준으로 최적의 종료시간을 120초로 설정할 경우, 서버가 30초 더 늦게 종료되므로 그 만큼 전력을 더 소비하게 된다. 서버가 90초 동안 잔여 요청 처리를 완료한 후, 30초가 지난 후에 요청이 종료 될 것이기 때문에 요청에 대한 Fail은 발생하지 않을 것이다.

반면에, 동적 종료 방법은 QoS와 소비전력의 트레이드오프 관계가 아닌 두 가지 모두를 만족시키는 방법이다. 각 서버들이 모든 요청을 처리 완료했을 때, 서버가 Off모드로 전환되므로, Fail이 발생하지 않고 소비전력 또한 낭비되지 않는다.

Fig. 13에서, 정적 종료 방법(90초)과 동적 종료 방법의 누적 소비 전력은 7Wh 차이를 보인다. 하지만 서버의 수와 운영 시간에 비례하여 절감되는 전력량이 증가할 것을 추측할 수 있다.

Fig. 14(b)에서, 동적 종료 방법은 Fail이 발생하지 않았지만, 정적 종료 방법(90초)은 66개의 Fail이 발생 했다. 정적 종료 방법에서 66개의 Fail을 방지하기 위해서는(QoS를 보장하기 위해서는) 90초 보다 더 긴 시간을 설정해야 할 것이며, 추가된 시간에 비례하여 소비전력이 증가하게 될 것이다.

5. 결 론

본 논문에서 제안된 동적 서버 종료 시스템은 기존 정적 종료 시스템이 정해진 최적 종료 시점을 이용하여 서버의 전원을 종료하는 것과 다르게 요청 처리 완료 여부를 확인하여 동적으로 최적 종료 시점을 결정할 수 있도록 하였다. 이로 인하여, 기존 정적 종료 시스템에서 미리 정해진 서버 종료시점을 사용함으로써 발생되는 QoS 보장 문제, 전력 낭비 문제, 서버 성능에 의존적인 최적 종료시점 문제 등을 해결하였다.

향후 연구 방향으로는 서버의 대수를 늘리는 실험을 고려해볼 수 있다. 이 경우 서버로부터 정보를 가지고 오는 부하가 발생할 수 있고 이를 해결하기 위해서는 2가지 해결방법이 존재한다. 하나는 부하분산기(LVS)의 사양을 높이는 것이고, 다른 하나는 정보를 가지고 오는 주기를 늘리는 것이다.

참 고 문 헌

[1] S. Lee, S. Mun, J. Kim, S. Shin, Y. Seo, and Y. Choi, “The Establishment Method of Green Data Center in Public Sector”, The Journal of KIISE, Vol.27, No.11, pp.48-57, 2009.

[2] Chenguang Liu, Jianzhong Huang, Qiang Cao, Shenggang Wan, Changsheng Xie, “Evaluating Energy and Performance for Server-Class Hardware Configurations”, 6th IEEE International Conference on Networking, Architecture and Storage, 2011.

[3] J. Mair, K. Leung, Z. Huang, “Metrics and task scheduling policies for energy saving in multicore computers”, 11th IEEE/ACM International Conference on Grid Computing (GRID), 2010.

[4] Xinying Zheng, Yu Cai, “Markov Model Based Power Management in Server Clusters”, IEEE/ACM Int'l Conference on Green Computing and Communications, 2010.

[5] Bruce Nordman, “What the Real World Tells Us about Saving Energy in Electronics”, Symposium on Energy Efficient Electronic Systems, 2009.

[6] K. Rajamani and C. Lefurgy, “On Evaluating Request-Distribution Schemes for Saving Energy in Server Clusters”, Proceedings of the IEEE International Symposium on Performance Analysis of Systems and Software, pp.111-122, 2003.

[7] Kang G. Shin, J. Reumann, S. Singhal, and C. Tsai, “Online Web Cluster Capacity Estimation and Its Application to Energy Conservation”, IEEE Transactions on Parallel and Distributed Systems, Vol.18, No.7, pp.932-945, 2007.

[8] Eduardo Pinheiro, R. Bianchini, E. Carrera, and T. Heath, “Dynamic Cluster Reconfiguration for Power and Performance”, Compilers and operating systems for low power, pp.75-93, 2003.

[9] H. Kim, C. Ham, H. Kwak, H. Kwon, Y. Kim, and K. Chung, “A Dynamic Server Power Mode Control for Saving Energy in a Server Cluster Environment”, The Journal of KIPS, Vol.19-C, No.3, pp.135-144, 2012.

[10] Graceful Shutdown, http://msdn.microsoft.com/en-us/library/ms738547(v=vs.85).aspx.

[11] LVS(Linux Virtual Server), http://www.linuxvirtualserver.org.

[12] SPECweb Banking Design, http://www.spec.org/web2005/docs/BankingDesign.html.

[13] InternetTrend™., http://trend.logger.co.kr

김 호 연

e-mail : hykim@q.ssu.ac.kr
2012년 숭실대학교 정보통신전자공학부 (학사)
2011년~현 재 펌킨네트웍스 개발 엔지니어
2012년~현 재 숭실대학교 정보통신전자공학부 석사과정
관심분야 : 네트워크 컴퓨팅 및 보안

함 치 환

e-mail : hch@q.ssu.ac.kr
2009년 숭실대학교 전자계산원(학사)
2009년~2012년 숭실대학교 정보통신전자공학부(석사)
2012년~현 재 펌킨네트웍스 개발 엔지니어
관심분야 : 네트워크 컴퓨팅 및 보안

곽 후 근

e-mail : gobarian@q.ssu.ac.kr
1998년 숭실대학교 전자공학과(석사)
1998년~2006년 숭실대학교 전자공학과 (박사)
1998년~2000년 (주)3R 부설 연구소 주임 연구원
2007년~2010년 숭실대학교 정보통신전자공학부 겸임교수
2003년~현 재 펌킨네트웍스 기술이사
관심분야 : 네트워크 컴퓨팅 및 보안, 임베디드 소프트웨어

정 규 식

e-mail : kchung@q.ssu.ac.kr
1979년 서울대학교 전자공학과(공학사)
1981년 한국과학기술원 전산학과(이학석사)
1986년 미국 University of Southern California(컴퓨터공학석사)
1990년 미국 University of Southern California(컴퓨터공학박사)
1998년 2월~1999년 2월 미국 IBM Almaden 연구소 방문 연구원
1990년 9월~현 재 숭실대학교 정보통신전자공학부 교수
관심분야 : 네트워크 컴퓨팅 및 보안, 임베디드 소프트웨어