KIPS Tr. Comp. and Comm. Sys.
Vol.4, No.6 pp.185~196 pISSN: 2287-5891

http://dx.doi.org/10.3745/KTCCS.2015.4.6.185

An Energy Efficient Cluster Management Method based on Autonomous Learning in a Server Cluster Environment

Sungchul Cho† · Hukeun Kwak†† · Kyusik Chung†††

ABSTRACT

Energy aware server clusters aim to reduce power consumption at maximum while keeping QoS(Quality of Service) compared to energy non-aware server clusters. They adjust the power mode of each server in a fixed or variable time interval to let only the minimum number of servers needed to handle current user requests ON. Previous studies on energy aware server cluster put efforts to reduce power consumption further or to keep QoS, but they do not consider energy efficiency well. In this paper, we propose an energy efficient cluster management based on autonomous learning for energy aware server clusters. Using parameters optimized through autonomous learning, our method adjusts server power mode to achieve maximum performance with respect to power consumption. Our method repeats the following procedure for adjusting the power modes of servers. Firstly, according to the current load and traffic pattern, it classifies current workload pattern type in a predetermined way. Secondly, it searches learning table to check whether learning has been performed for the classified workload pattern type in the past. If yes, it uses the already-stored parameters. Otherwise, it performs learning for the classified workload pattern type to find the best parameters in terms of energy efficiency and stores the optimized parameters. Thirdly, it adjusts server power mode with the parameters.

We implemented the proposed method and performed experiments with a cluster of 16 servers using three different kinds of load patterns. Experimental results show that the proposed method is better than the existing methods in terms of energy efficiency: the numbers of good response per unit power consumed in the proposed method are 99.8%, 107.5% and 141.8% of those in the existing static method, 102.0%, 107.0% and 106.8% of those in the existing prediction method for banking load pattern, real load pattern, and virtual load pattern, respectively.

Keywords : Power Mode Control, QoS, Power Consumption, Autonomous Learning, Prediction Algorithm

서버 클러스터 환경에서 자율학습기반의 에너지 효율적인 클러스터 관리 기법

조 성 철† · 곽 후 근†† · 정 규 식†††

요 약

에너지 절감형 서버 클러스터는 에너지 절감을 고려하지 않는 기존 서버 클러스터에 비해 서비스 품질을 보장하면서 전력소비를 절감하는 것을 목표로 한다. 에너지 절감형 서버 클러스터에서는 현재의 부하를 처리하는 데 필요한 최소수의 서버들만 ON 하도록 고정 또는 가변 주기로 서버들의 전원모드를 조정한다. 이에 대한 기존 연구들은 전력 절감 또는 서비스 품질을 보장하려고 노력해왔지만 에너지 효율성을 잘 고려하지는 못했다. 본 논문에서는 에너지 절감형 클러스터에서 자율학습기반의 에너지 효율적인 클러스터 관리 기법을 제안한다. 자율학습을 통해 최적화된 파라미터들을 이용하여 전력 소모 대비 최고의 성능을 얻을 수 있도록 서버 전원모드를 조정한다. 제안방법은 서버 전원모드 조정을 위해 아래의 과정을 반복 수행한다. 첫째, 현재 부하 및 트래픽 패턴을 보고 현재 워크로드 패턴 유형을 사전에 정의한 대로 분류한다. 둘째, 학습 테이블을 탐색하여 해당 워크로드 패턴 유형에 대해 예전에 학습이 수행되었는지 확인한다. 만일 수행되었다면 이미 저장된 파라미터를 이용한다. 그렇지 않으면, 학습을 수행하여 에너지 효율성 관점에서 최고의 파라미터를 얻어 저장한다. 셋째, 얻어진 파라미터를 이용하여 서버 전원모드를 조정한다.

제안방법을 구현하여 16개의 서버 클러스터 환경에서 3가지 다른 부하 패턴들을 이용하여 실험을 수행하였다. 실험 결과는 제안방법의 에너지 효율성이 뛰어남을 보여주고 있다. 뱅킹 부하패턴, 실제 부하패턴, 가상 부하패턴 각각에 대하여, 제안방법의 단위전력당 good 응답 수가 기존의 정적 서버 전원모드 제어방법의 99.9%, 107.5%, 141.8%이고, 기존의 예측방법의 102.0%, 107.0%, 106.8%이다.

키워드 : 전원모드 제어, QoS, 소비전력, 자율학습, 예측 알고리즘

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

※ 이 논문은 2012년도 정부(교육과학기술부)의 재원으로 한국연구재단의 기초연구사업 지원을 받아 수행된 것임(NRF-2012R1A1A2006602).
† 정 회 원 : 숭실대학교 전자공학과 박사과정
†† 정 회 원 : 유진 CTO
††† 정 회 원 : 숭실대학교 스마트시스템 소프트웨어학과 교수
Manuscript Received : January 13, 2015
First Revision : April 14, 2015
Accepted : May 13, 2015
* Corresponding Author : Kyusik Chung(kchung@q.ssu.ac.kr)

운용하기 위한 기반 설비로 나눌 수 있다. IT 장비는 데이터 센터마다 약간의 차이는 있으나 데이터 센터에서 사용하는 전체 에너지 소비량의 약 50% 이상을 차지한다[1].

데이터 센터에서 에너지 소모 비중이 가장 높은 서버들에서 에너지 소모를 줄이는 방법들로는 1) 에너지 절감형 서버 하드웨어를 사용하는 방법[2], 2) 멀티코어/멀티프로세서를 사용하는 서버를 사용하는 경우 한 서버 내 CPU 전력 소모를 최소화하도록 서버 운영체제에서 에너지 절감형 스케줄링을 사용하는 방법[3], 3) 서버 클러스터에서 부하 상황에 따라 필요한 수만큼의 서버들만 ON 하고 나머지는 OFF 하는 에너지 절감형 서버클러스터 관리 기법[4] 등이 있다.

에너지 절감형 서버 클러스터는 에너지 절감을 고려하지 않는 기존 서버 클러스터에 비해 서비스 품질을 보장하면서 전력소비를 절감하는 것을 목표로 한다. 에너지 절감형 서버 클러스터에서는 현재의 부하를 처리하는 데 필요한 최소수의 서버들만 ON 하도록 고정 또는 가변 주기로 서버들의 전원모드를 조정한다. 기존의 에너지 절감형 서버 클러스터 관리 기법들을 살펴보면 다음과 같이 분류할 수 있다. 클러스터 내 서버 전원모드를 고정주기로 제어하느냐 또는 가변주기로 제어하느냐로 나눌 수 있다[5]. 또한 현재 부하를 처리하는 데 필요한 서버 대수를 계산할 때 서버당 처리 능력에 대한 임계치로 단일값을 사용하느냐 또는 다중값을 사용하느냐로 나눌 수 있다[6]. 또한 트래픽이 급변하게 변화하는 상황에 대처하기 위한 트래픽 예측 기법을 사용하느냐 여부로 나눌 수 있다[7]. 기존의 에너지 절감형 서버 클러스터 연구들은 전력을 절감하거나 서비스 품질을 보장하려고 노력해왔지만 에너지 효율성(소비하는 단위전력당 성능)을 잘 고려하지는 못했다. 본 논문에서는 에너지 절감형 클러스터에서 자율학습기반의 에너지 효율적인 클러스터 관리 기법을 제안한다. 자율학습을 통해 최적화된 파라미터들을 이용하여 전력 소모 대비 최고의 성능을 얻을 수 있도록 서버 전원모드를 조정한다.

본 논문의 구성은 다음과 같다. 2절에서는 에너지 절약을 위한 서버 전원모드 제어에서 기존 방법에 대한 연구를 소개한다. 3절에서는 자율학습을 통한 서버 전원모드 제어방법을 소개한다. 4절에서는 실험 및 토론을, 5절에서는 결론 및 향후 연구 방향을 제시한다.

2. 연구 배경

2.1 서버 클러스터링 시스템

1) 시스템 구조

리눅스 가상서버(LVS: Linux Virtual Server)[8]는 리눅스를 기반으로 독립된 여러 서버들을 하나의 클러스터로 구

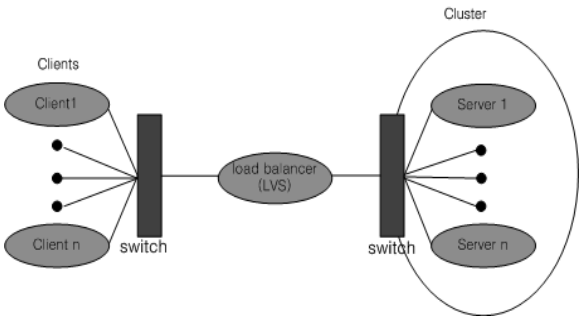


Fig. 1. Cluster Environment

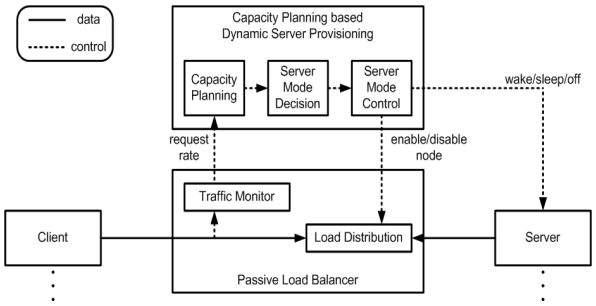


Fig. 2. Procedure of Energy-saving Server Cluster

성하여 뛰어난 확장성과 안정성을 제공한다. Fig. 1은 리눅스 가상시스템의 전체적인 구조를 나타낸다. 더 높은 성능의 클러스터링 시스템의 구축은 높은 비용과 함께 개선의 여지가 필요할 때마다 추가적인 비용이 발생할 수 있는데 부하분산기는 scaleout 형태의 확장을 지원하고 상용서버를 사용하기 유리한 구조를 가진다.

부하분산기로부터 서버로의 스케줄링 알고리즘으로 라운드로빈, 가중치 라운드로빈, 최소 연결 수, 최단 응답 시간, 최소 부하 서버에 할당하는 방식 등이 있다. 그 밖의 방법으로 해싱, LBLC(Locality Based Least Connection) 등의 기법이 있다[9].

2) 에너지 절감형 서버 클러스터 동작

클러스터에서는 현재의 부하를 처리하는 데 필요한 최소수의 서버들만 ON 하도록 고정 또는 가변주기로 서버들의 전원모드를 조정한다. 이를 지원하기 위해 기존 서버 클러스터에 동적 서버 프로비저닝이 추가된다. Fig. 2에서 하반부는 Fig. 1과 같고 상반부가 동적 서버 프로비저닝이다.

동적 서버 프로비저닝에는 각 서버가 담당할 수 있는 부하량을 사전에 정의하고 용량계획에 의해 현재의 트래픽을 감당할 수준으로 서버를 배정하는 모듈, 서버 전원모드를 결정하는 모듈, 서버 전원모드를 제어하는 모듈이 포함되어 있고, 부하분산기에는 외부로 들어오는 사용자 요청 트래픽을 수집하여 동적 서버 프로비저닝에게 전달하는 모듈, 부하분산 정책에 의해 외부 사용자 요청들을 서버들로 분배하

는 부하분배 모듈이 들어있다.

Fig. 2의 에너지 절감형 서버 클러스터 내부구조의 동작 과정은 다음과 같다[5].

주기적으로 각 서버가 보내는 상태 정보값을 부하분산기에서 수집한다. 수집된 정보에는 각 서버 자신들의 소비전력을 추정값이 포함되어있다. 상태 정보값을 수집하는 주기보다는 긴 일정 주기마다, 부하분산기에서는 수집한 해당값들을 합산하여 클러스터의 총 추정소비전력을 구한다. 이를 서버 한 대가 낼 수 있는 적정 수준의 성능에 해당하는 소비전력값(임계치)으로 나누어 필요한 서버의 수를 구하고 그만큼의 서버만이 ON 되도록 서버 전원모드를 제어한다. 서버를 OFF 해야 할 경우에는 suspend를, 서버를 ON 해야 할 경우에는 wol을 이용하여 서버 전원모드를 제어할 수 있다.

2.2 기존 에너지 절감 연구

1) 전력 절감을 위한 서버의 동적 서버 전원모드 제어

각 서버로부터 받은 추정 소비전력 총합을 특정한 하나의 임계치로 나누어 적정 서버 수를 구하는 방법이다[5]. 또한 이 연구에서는 트래픽이 변화하는 상황에 따라 서버 전원모드 제어 주기를 동적으로 조정한다. 인터넷 서비스의 트래픽 특성이 같은 요일의 같은 시간대에는 비슷한 패턴이 반복된다는 가정하에, 과거(같은 요일의 같은 시간대)의 소비전력 정보를 근거로 현재보다 약간 앞선 시점에서 예측한 서버들의 소비전력 정보와 현재 평균 소비전력 정보를 비교하여 그 차이가 크면 트래픽 변화가 크다고 판단하여, 서버 전원모드 제어주기를 짧게 조정하고 그렇지 않으면 길게 조정한다. 이 연구에서는 전력 절감 효과가 뛰어난 장점이 있지만, 제어주기가 길어진 경우에 급격한 트래픽이 발생하는 경우 QoS를 보장하기 어려운 점과 요청하는 서비스의 유형을 분류하지 못하는 단점이 있다.

2) 예측 기법

짧은 시간에 급격한 트래픽 변화가 발생하는 상황을 다루기 위해 패킷 유입량을 기준으로 시계열 모델을 사용한 예측 알고리즘을 이용해서 서비스 요청 증가, 감소를 예측하여 서버의 ON/OFF상태를 제어하는 알고리즘이다[5]. MMA (Modified Moving Average), MWMA(Modified Weighted Moving Average), MES(Modified Exponential Smoothing), MESTA(Modified Exponential Smoothing with Trend Adjusted) 방식에 대해 비교실험을 수행하였다[7]. 이 연구에서는 정적 서버 전원모드 제어 동작과정에 트래픽을 예측하는 부분을 추가한 방법을 사용한다. Trend Adjusted Part 적용을 통한 트래픽 예측을 통해 5~20초가량 서버를 빠르게 ON 하여 서비스 요청 처리를 준비함으로써 QoS의 향상 효과를 가져왔으나, 요청하는 서비스의 종류에 따라 서버에 미치는 부하가 각기 다른데, 제안하는 방법에서는 트래픽만을 고려하여 실제 상황을 고려하지 못하는 한계가 있다.

3) 자율학습기반의 적응적 클러스터링

클러스터링 기반의 인터넷 프록시 서버에서 Hot-Spot 및 임의의 입력 요청 패턴이 발생하면 일부 서버만 과부하가 되어 전체적인 성능이 떨어지는데, 이런 문제를 해결하기 위한 자율학습기반의 적응적 클러스터링 기법이다[10]. 이 연구는 요청을 처리하는 일부 서버들이 과부하가 되면 해당 요청을 다른 서버들로 재분산하는 방식이다.

이 연구는 일부 서버로 요청이 몰리게 되면 다른 그룹의 서버 중에 부하가 낮은 서버(요청을 적게 처리하는 서버)를 자신의 클러스터로 가지고 와서 요청을 함께 처리하는 방식이다. 이 방식에서는 학습 방법을 적용하여 클러스터 재구성을 위한 운영변수가 자동으로 결정되는 장점이 있지만, 모든 서버가 항상 ON 되어있는 상황에서 서버 활용도를 극대화하기 위해 학습 방법을 적용했는데 에너지 절감이 전혀 고려되지 않았다.

2.3 접근 방식

기존 연구들의 문제점들을 해결하기 위하여 본 논문에서는 다음의 접근 방식으로 연구를 진행한다.

첫째, 기존 방법은 워크로드(작업 부하량)가 단위시간당 사용자 요청 트래픽에 선형적으로 비례한다는 단순한 모델에서 출발한다. 어떤 사용자 요청이 서버에게 정적 콘텐츠를 요청하는가 또는 동적 콘텐츠를 요청하는가, 즉 요청하는 서비스 종류에 따라 서버에 미치는 부하는 각기 다르기 때문에 트래픽만을 고려한 워크로드 정의는 실제 상황을 고려하지 못하는 한계가 있다. 본 논문에서는 서버 클러스터에 걸리는 작업 부하를 정확히 파악하기 위해, 사용자 요청 트래픽뿐만 아니라 현재 동작 중인 서버의 부하 상태(CPU 사용률, 메모리 사용률, IO 처리율, 전력 등)를 같이 고려한 이용률 산정 방식을 사용하였다.

둘째, 기존 방법은 언제, 어떻게 클러스터 재구성을 조정할 것이냐에 영향을 주는 서버 프로비저닝의 운영 변수 값들이 특정 서비스에 맞도록 최적화되어 있다. 서비스 종류가 바뀌면 운영 관리자가 그에 따라 사용하는 운영 변수 값들을 새로 정해주어야 한다. 이 작업은 수동으로 많은 시간 동안 준비해야 하는데 이를 개선하기 위해 본 논문에서는 클러스터 재구성을 위한 운영 변수들이 자율학습을 자동으로 결정되도록 구성하였다.

셋째, 기존 방법에서 모든 서버가 항상 켜져 있는 상태로 동적 클러스터링 할 경우 전력효율이 낮아지는 문제가 있다. 본 논문에서는 이를 해결하는 학습기반의 서버 프로비저닝을 적용하고 클러스터를 ON/OFF 하는 방식을 통해 전력효율을 개선하고, 단위전력당 good 응답 수 지수를 개발하고 이를 효율지수로 이용하였다.

3. 제안된 시스템

3.1 시스템 모델

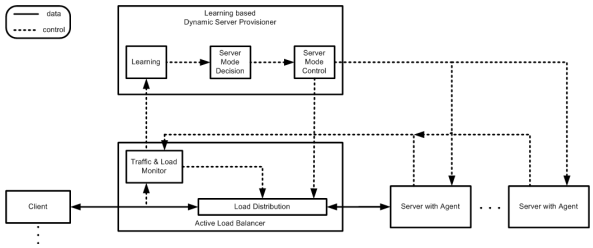


Fig. 3. Internal Structure of Load Balancer

Fig. 3은 부하분산기의 내부 구조를 나타낸다. 부하분산기의 내부 구조를 크게 두 분류로 나는 것은 기존의 부하분산 시스템에 학습기반의 동적 서버 프로비저너를 추가하기 쉽게 하기 위함이다. 부하분산기 상반부의 동적 서버 프로비저너는 학습모듈, 서버 전원모드 결정모듈, 서버 전원모드 제어모듈로 구성되고, 부하분산기 하반부의 부하분산기는 부하 모니터, 부하분산모듈로 구성된다.

1) 부하분산기가 수집하는 서버의 부하정보

Fig. 3 하반부의 부하 모니터(Traffic & Load Monitor)는 전체 클러스터의 부하 상태를 판단하기 위해 각 서버의 부하 정보를 수집한다. 부하분산기를 통해 각 서버의 부하정도를 측정하기 위해 서버 에이전트를 구현하고, 서버 에이전트는 전력, 연결 수, cpu사용량, 송/수신량 등의 정보를 주기적으로 전송하고 python demon[12] 형태로 구현되었다.

Table 1. Metric for Utilization

메트릭	설명	범위
cpu	서버의 CPU 사용률(%)	0~100
memory	서버의 메모리 사용률(%)	0~100
power	서버의 전력 사용률(%)	0~100
connection	TCP 연결수 (ESTABLISHED 상태)	0~ 수천 연결
send/recv bytes	서버의 전체 송/수신량	0~ 수십MByte
server ID	서버의 고유 번호	1~16
LVS input KBytes	부하분산기로 유입되는 데이터양	0~ 수십GByte

Table 1에서 수집된 부하정보와 트래픽정보는 서버 프로비저닝을 결정하는 파라미터로 이용하게 된다. 각 파라미터는 현재 값, 10초, 60초 이동평균 데이터를 가지고 있으며, 급격한 부하를 결정하는 변수로 이용한다.

2) 부하분산기의 동작

부하분산기는 클러스터의 활성화된 서버들 중에서 특정 서버에게 부하가 몰리지 않고 균등하게 배분하도록 역할을 한다. Fig. 3 상반부의 학습기반 동적 서버 프로비저너(Learning based Dynamic Server Provisioner)에서 결정한 서버 전원모드에 따라 활성화할 서버와 비활성화할 서버를 결정하고, 활성화된 서버들에게만 클러스터에 할당된 부하를 나누어 주는 역할을 한다. 비활성화된 서버들에는 부하분산 가중치를 최하로 낮추어 서버를 OFF 할 수 있도록 한다. 부하분산 정책은 각 서버에서 소비하는 전력(추정치)에 역비례하여, 전력소비가 낮은 서버에 더 많은 부하를 나누어 줌으로써 서버들 사이의 부하를 고루 분배할 수 있는 wrr (weighted round robin)을 이용한다[8].

3) 클러스터의 서버 대수 조절

서버 전원모드는 클러스터의 각 서버가 어떠한 상태에 있는지를 알려주기 위해 필요하고, 이를 기반으로 서버 on/off 제어에 활용된다. 서버의 전원모드는 활성화된 상태, 서버를 off하고 있는 상태, 서버가 비활성화된 상태, 서버가 켜지고 있는 상태들로 4가지 정의하고 각 상태 간에 전이가 이루어지도록 구성했다.

클러스터의 서버 수는 켜지고 있는 상태에서는 클러스터의 서버수를 증가시키고 활성화된 서버수에 포함하여 산정되며, 서버를 off하고 있는 상태에서는 클러스터의 서버 수를 감소시킨다. 비활성화된 서버와 off하고 있는 상태의 서버는 클러스터의 서버수 산정에서 제외된다. 서버 동작제어에 사용하는 전원모드 제어모듈은 서버에게 wol을 사용해서 on하고 suspend를 통해 off하는 역할을 수행하게 된다.

4) 동적 서버 프로비저너에서의 자율학습

2.1절에서 설명한 에너지 절감형 서버 클러스터 내 동적 서버 프로비저닝을 보면 각 서버가 담당할 수 있는 부하량을 사전에 정의하고 용량계획에 의해 현재의 트래픽을 감당할 수준으로 서버를 배정한다. 다시 말하면, 클러스터의 총 추정소비전력을 서버 한 대가 낼 수 있는 적정 수준의 성능에 해당하는 소비전력값(임계치)으로 나누어 필요한 서버의 수를 구하고 그만큼의 서버만이 ON 되도록 서버 전원모드를 제어하는 데 이 과정을 반복한다. 본 논문에서는 기존 용량계획 기반의 동적 서버 프로비저닝 대신에 학습 기반으로 최적의 서버 대수를 결정한다. 기존 용량계획 기반에서는 클러스터에 입력되는 워크로드(작업부하) 패턴에 무관하게 동작하는 반면에, 본 논문에서는 입력되는 워크로드 패턴을 분류하고 그에 따라 최적의 서버 대수를 결정할 수 있게 하는 학습 방법을 적용한다.

워크로드 패턴은 각 서버의 네트워크 트래픽 및 cpu 부하값들에 의해 결정되는데 본 연구에서는 워크로드 패턴을 트래픽 및 cpu 부하값들의 표준 편차들에 의해 분류한다[10]. 여기에서 표준 편차를 사용하는 의미는 클러스터 환경에서 각 서버들이 처리하고 있는 워크로드 패턴이 평균에서 얼마나 떨어져 있는 정도를 가지고 워크로드 패턴을 분류한다는 점이다. 각 서버의 현재 트래픽값과 cpu 부하값들이 주

어지면 트래픽값 표준 편차와 cpu 부하값 표준 편차를 구한 뒤 이들 정규화된 값들(각각을 con, cpu 특성값이라고 부름)로 서버 클러스터의 현재 워크로드 패턴을 규정한다.

실시간으로 측정되는 cpu 사용률 및 con 사용률을 가지고, cpu 사용률에 대해 가중치(이를 상태 가중치(ratio)로 부름)를 곱하고 con 사용률에 (1 - 상태가중치)를 곱하여 얻는, 각 사용률에 대한 가중치를 결합한 값으로 이용률(utilization)을 얻는다.

$$utilization = (1-ratio)\times con + ratio\times cpu \quad (1)$$

전체 클러스터에 대한 총 이용률을 구하고, 찾아진 임계치로 나누어 필요한 서버 대수를 결정한다. 본 논문에서는 현재 주어진 워크로드 패턴에 대해 서버 클러스터 전체 소비전력당 서버 클러스터의 성능을 최대로 할 수 있는 최적 서버 대수를 결정하기 위해 학습 방법을 적용한다. 런타임에 상태 가중치를 계속 바꾸어가면서 최적 서버 대수를 찾는다.

일단 학습이 한번 수행되면 그 학습 결과(트래픽 및 cpu 부하에 대한 상태 가중치 및 서버의 워크로드 임계값)를 테이블에 기록하고, 그다음부터는 동일한 워크로드 패턴이 나타나면 기록된 학습 결과를 이용하여 서버 프로비저닝이 동작한다.

본 논문에 적용된 학습 방법은 무감독 학습(unsupervised learning)으로 분류된다. 런타임에 학습이 이루지고 부하분산을 할 수 있도록 하기 위해, 본 논문의 학습 알고리즘에서는 워크로드 패턴을 분류하여 입력 경우의 수를 줄여 단순화하고, 상태가중치와 임계값을 유효한 범위로 조정하여 빠르게 처리할 수 있도록 구성했다. 학습 단계별 설명과 학습 예를 통한 설명은 3.2절에서 설명하도록 한다.

3.2 에너지 절감을 위한 제안 기법

본 논문에서의 학습 방법은 참고문헌[10]의 자율학습 방법을 참고하여 설계 및 구현하였다. 참고문헌에서는 에너지 절감이 전혀 고려되지 않았다[10, 17]. 반면에 본 논문에서는 학습과정에서 전력소모 대비 서버 클러스터의 성능이 최대화될 수 있도록 수정하였다. 또한, 학습 알고리즘의 오버헤드를 줄이기 위해 학습이 이미 이루어진 경우 파라미터 로드하도록 구성하였고, 에너지 효율을 높일 수 있는 학습 파라미터를 선택하도록 하였다. Fig. 4에서 학습부분의 상세한 동작 흐름을 보여주고 있다.

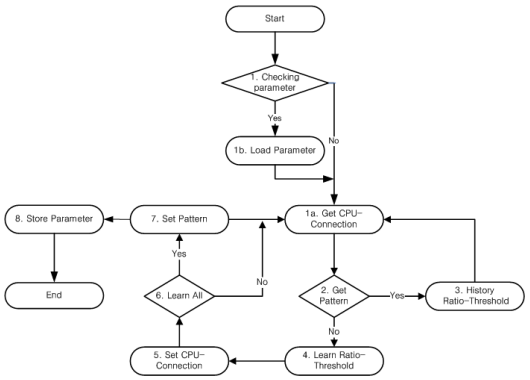


Fig. 4. Autonomous Learning Process

단계 1) 학습 파라미터 기록 확인

자율학습 알고리즘의 런타임 수행을 하기 전에 파라미터가 사전에 기록되었는지 확인 후 기록이 없으면 다음 단계(1a)로 넘어간다. 기록이 있으면 파라미터에 대한 정보를 읽어들인 후 다음 단계(1a)로 진행한다.

단계 1a) CPU-Connection 표준편차 확인

cpu와 con(트래픽)에 대한 표준편차를 구하고, Fig. 5의 정규화값으로 표시한다.

단계 2) 학습 패턴 테이블 확인

학습 패턴 테이블에 cpu와 con에 대한 해당 엔트리가 존재하는지 확인한다. Fig. 6의 학습 패턴 테이블의 해당 엔트리가 비어있는지 혹은 쓰여 있는지 검사한다.

단계 3) 상태 가중치-임계값 선택

학습 패턴 테이블이 채워져 있으면 이미 학습된 것으로 엔트리(cpu, con)에 대한 상태 가중치를 찾고 임계값을 선택한다.

단계 4) 상태 가중치-임계값에 대한 학습

기존 학습 패턴이 없다면, 새로운 패턴이므로 학습을 수행하도록 한다. 엔트리(cpu, con)에 대한 상태가중치(ratio)를 부여하여 이용률(utilization)을 얻고, 임계값을 바꾸어가면서 학습한다.

단계 5) 상태 가중치와 임계값 기록

Fig. 7의 인덱스 테이블 중 해당 엔트리(cpu, con)에서 상태 가중치와 임계값의 조합으로 부하분산을 수행하면서 처리량과 소비전력을 기록한다.

단계 6) 모든 조합에 대한 학습 여부 확인

모든 조합에 대해 기록이 이루어졌으면, Fig. 7의 전체 학습 인덱스 테이블을 검색하여 최적의 단위전력당 처리량을 보인 것을 선정한다.

단계 7) 학습 패턴 테이블에 기록

이전 단계에서 선정된 상태 가중치와 임계값을 Fig. 6의 학습 패턴 테이블에 최적값을 기록한다.

단계 8) 학습 패턴 테이블과 학습 인덱스 테이블을 파일로 기록

학습이 이루어졌고, 부하패턴의 재생시간이 종료된 경우 학습 파라미터를 저장하고 종료한다.

CPU Stdev	Value
< 10	0
< 30	1
...	...
< MAX_CPU	n

CON Stdev	Value
< 1000	0
< 5000	1
...	...
< MAX_CON	n

Fig. 5. Normalized Usage Staus

Fig. 6. Pattern Matrix

CPU STDEV	CON STDEV	Weight			BPS	Total Power
		CPU	CON	Thres.		
0	0	0.0	1.0	50	750	1050
		0.0	1.0	55	840	1140
		0.0	1.0	60	860	1160
		...	...	...	...	...
...	...	...	...	...	...	...
2	1	...	...	...	...	...
		0.6	0.4	65	960	1260
		0.6	0.4	70	820	1290
...	...	...	...	...	...	...
9	9	1.0	0.0	70	820	1190

Max efficiency (BPS/Watt)

Fig. 7. Learning index

위의 3가지 그림은 학습 과정을 설명하기 위한 그림으로, Fig. 5는 CPU의 표준편차와 CON의 표준편차가 정규화되는 값의 범위를 표현한 그림이다. Fig. 6은 학습 패턴 테이블의 엔트리(CPU, CON)가 어떤 조합을 가지고 있는지 기록하기 위한 패턴 테이블이다. Fig. 7의 학습 인덱스 테이블은 어떤 조합에서 최고의 전력당 효율을 가지는지를 기록하고, 찾는데 이용한다.

학습 과정을 예로 들어 설명하면 다음과 같다. Fig. 3에서 클러스터를 구성하는 서버에 설치된 에이전트의 부하 정보를 부하분산기의 하반부 부하 모니터에서 수집한다. 이렇게 수집된 정보 중에서 CPU와 CON의 표준편차를 계산하고, 정규화하면 학습 패턴 테이블의 어떤 엔트리에 속하는지를 찾을 수 있다. Fig. 5와 같이 정규화하여 엔트리(2, 1)로 찾아졌지만, 학습되기 전이라면 Fig. 6의 엔트리(2, 1)가 비어 있는 상태가 되어 어떤 상태 가중치와 임계값을 갖는지 찾아야 한다. 그러기 위해서는, 엔트리(2, 1)에 상태 가중치를 부여하고 임계값을 바꾸면서 Fig. 7의 학습 인덱스 테이블 중 엔트리(2, 1)의 범위에 처리량(BPS)와 소비전력(watt)를 기록한다. Fig. 7 중 엔트리(2, 1)의 모든 조합들에 대해 학습 중 기록된 단위전력당 소비전력(BPS/watt)이 가장 높은 조합을 선택한다. 기록한 학습 인덱스를 검색한 결과 (0.6, 0.4, 65)가 최적값이다. Fig. 3의 하반부 부하분산기에 엔트리(2, 1)의 해당 값을 전달하고 클러스터의 부하분산에 이용하게 된다.

4. 실험 및 토론

4.1 실험 환경

Fig. 8은 실험 전체 구성을 나타낸다. 클러스터는 리얼서버 16대로 구성하고, COTS x86 서버의 성능 향상으로 prime client와 client를 6대로 늘렸다. prime client는 client가 여러 대일 때 동기화 실행을 관리하고, Besim(Backend Simulator)는 client의 동작(로그인, 계좌이체 등)을 제어한다. 부하분산기 1대와 기가비트 스위치 2대를 이용했으며, 부하 테스트환경으로 SPECweb을 이용하였다.

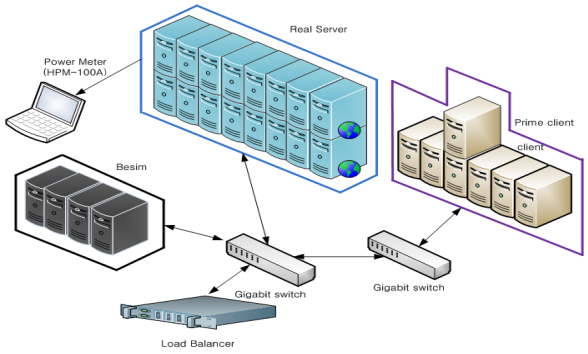


Fig. 8. The Overall Configuration of Experiments

서버에 설치된 아파치[13]의 경우 서버의 웹 처리용량을 늘리기 위해 mpm-event모드를 이용하고, 클러스터는 Fedora를, 클라이언트는 Ubuntu를 설치하여 실험하였다. 전력의 측정은 SPECweb의 warmup구간이 끝나는 시점부터 하여 뱅킹과 실제 부하패턴은 1시간, 가상 부하패턴은 30분간의 클러스터의 전력소비를 측정하였다.

Table 2. Hardware and Software Specification for Experiment

항목	대수	하드웨어	소프트웨어
server	16	i5-3550, 8G, 400G HDD	Fedora 18, PHP 5.5.3, python 3.3.2, Apache 2.4.6
client	6	i5-3550, 8G, 400G HDD	Ubuntu 12.10
PrimeClient	1	i5-3550, 8G, 400G HDD	Ubuntu 12.10
LVS	1	i5-3550, 8G, 400G HDD	Fedora 13, ipvadm
Besim	4	i5-3550, 8G, 400G HDD	Fedora 13, PHP 5.3.6
SPECweb	1	장비에 분산 설치됨	Besim, Client, PrimeClient에 설치
PowerMeter	1	HPM-100A	serial com
network switch	2	SG92-24 Gigabit switch	

Table 2는 클러스터의 구성 하드웨어 요소들과 설치된 소프트웨어에 대한 설명이다. 여기에서 SPECweb은 별도의 하드웨어에서 동작하는 게 아니라 client, primeclient, besim에 나뉘어 분산 설치되었다.

4.2 실험 방법

본 논문에서는 트래픽 변화가 완만한 SPECweb 뱅킹 부하패턴뿐만 아니라 트래픽 변화가 심한 실제 부하패턴 및 가상 부하패턴의 3가지 부하패턴을 가지고 각각에 대해 5종류의 실험을 수행하였다. 5종류의 실험은 전원모드 제어 없

는 방식(always-on), 1초 주기로 전원모드 제어하는 방식(static), 예측(prediction), 자율학습(autonomous learning), 파라미터로드에 의한 자율학습(autonomous learning with parameter loaded)이다.

Fig. 9에 보이는 SPECweb banking[14]은 온라인 은행 업무에 근거하고, 이 작업 부하에 있는 모든 요청패턴을 만들어 생성하고 워크로드에 대한 실험을 하였다.

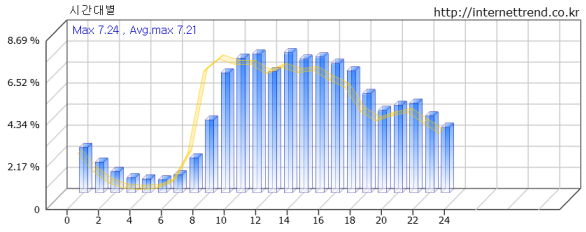


Fig. 9. InternettrendTM - Banking Load Pattern

Session : 2069 1242 730 491 402 446 774 2128 6118 6400 6050 6050 5708 5909 5656 5722 5269 4964 3870 3513 3750 3878 3453 2970
실험시간: 60분(그래프 하단의 24시간은 하루를 나타내며, 전체 실험 소요시간을 60분으로 전환하여 실험), 최대 session : 6400

InternetTrendTM[15]에는 일반사이트 및 모바일WEB/APP의 로그분석 솔루션 및 ASP서비스 LoggerTM에 의해 선별된 실측 로그분석에 의한 데이터 가운데 샘플링 데이터를 통계 처리하여 생성된 웹 분석 데이터의 평균값 데이터들이 카테고리/기간별로 DB화 되어 일반에게 공개되어있다. 따라서 원하는 카테고리와 시간대를 입력하면 해당 트래픽 추세를 확인할 수 있다.

Fig. 10에 보이는 실제 부하패턴은 InternetTrendTM[15]의 Website Performance 카테고리의 주문집중시간대 자료 중 2012/9/1~2012/9/7까지의 금융/부동산/경제 파트의 시간대별 집중도를 따라서 생성하였다.

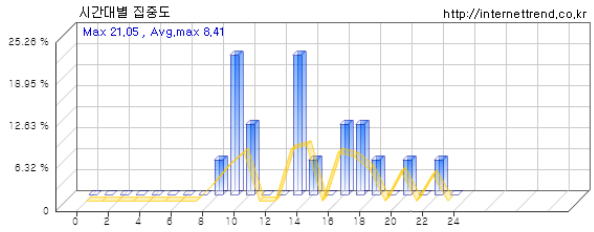


Fig. 10. InternettrendTM - Website Performance

Session : 0 0 0 0 0 0 0 0 1599 6400 3202 0 0 6400 1599 0 3202 3202 1599 0 1599 0 1599 0
실험시간: 60분(그래프 하단의 24시간은 하루를 나타내며, 전체 실험 소요시간을 60분으로 전환하여 실험), 최대 session : 6400

Fig. 11에 보이는 가상 부하패턴은 급격한 부하 변화에 대한 성능을 확인하고자 생성하였다.

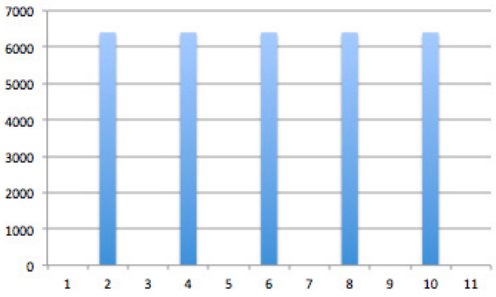


Fig. 11. Virtual Workload Pattern

Session : 0 6400 0 6400 0 6400 0 6400 0 6400 0
실험시간: 30분(그래프 하단의 24시간은 하루를 나타내며, 전체 실험 소요시간을 30분으로 전환하여 실험), 최대 session : 6400

Table 3. Mean of SPECweb Graph Lline

그래프 선 유형	실험 그래프 선의 의미
total request	time good + time tolerable + time fail + validation errors
time good	2초 내의 요구 응답을 만족하는 페이지 요청
time tolerable	4초 내의 요구 응답을 만족하는 페이지 요청
time fail	응답하는 데 4초 이상 걸린 요청
validation error	부하 요청 유형에 따른 각 요청의 에러
QoS	time good / (total request - validation error)

Table 3은 SPECweb그래프 결과로 표현되는 각 파라미터의 의미를 나타내며, 총 유효 응답 수는 4가지 파라미터의 합으로 표현되고, QoS는 유효한 응답 수에서 time good인 응답의 비율로 나타낸다.

5종류의 실험방법을 설명하면 다음과 같다. always-on은 클러스터 구성서버들이 항상 켜져 있는 경우에 대한 실험이다. 정적서버 전원모드 제어는 제어주기를 최소 단위인 1초로 하는 실험이다. 2.2.1절에서 설명한 동적 전원모드 제어[5]에서는 기본적으로 5초 주기로 서버의 전원모드를 제어하면서 일정수준 이상의 순간 트래픽 변화가 감지되면 1초 주기의 전원모드 제어를 추가로 사용한다. 이 방법을 4.1절의 실험 환경에서 적용한 결과, 충분한 QoS를 만족하지 못해 본 실험에서는 제어주기를 최소 단위인 1초로 하였다. 예측기법의 경우 부하분산기로 유입되는 네트워크의 부하에 대해 추세 조정된 지수평활방법(modified exponential smoothing)을 동일하게 1초 단위로 서버 전원모드를 제어하는 방식이다. 여러 가지 추세 조정된 예측 기법들 중에 이 예측 기법이 본 실험에서 채택된 이유는 실험 과정에서 QoS와 단위전력당 good 응답 수가 가장 높은 결과를 보였기 때문이다. 자율학습 방법의 경우는 서버 전원모드 제어는 정적 서버 전원모드 제어와 동일하고 필요한 서버 수를 산정하는 부분에서 자율학습을 적용한 방식에 해당한다. 이 경우 역시 1초 단위의 서버 전원모드 제어를 하고, 3초 단위로 자율학습과 네트워크 연결 가중치를 변경하는 방식이다. 그리고 단일 임계값을

가지고 서버 전원모드 제어를 하는 정적 서버 전원모드 제어와 자율학습 방법에 동일하게 하나의 임계값을 가지고 실험하여 단위전력당 good 응답 수를 비교하여 Table 5, 6, 7의 결과를 얻었다. 파라미터로드에 의한 자율학습 방법은 학습하는 동안에 발생할 수 있는 오버헤드를 제거한 실험에 해당한다. 자율학습에 의해 발생하는 오버헤드로는 실험 초기 warmup구간과 이전에 없던 새로운 부하유형이 생성될 때 발생되는데 임계값 범위*상태가중치 범위*연결가중치의 변경 주기만큼의 시간을 소비하고 학습되지 않은 구간에 해당하므로 연결 수 표준편차, cpu 표준편차에 해당하는 학습 파라미터를 저장하는 동작을 하게 된다.

4.3 실험 결과

Fig. 12의 SPECweb workbench 그래프는 실험 결과를 보여준다. time good으로 표시된 노란색 선은 클라이언트의 요청에 2초 이내 응답받은 수를, time tolerable은 4초 이내 응답 수, time fail은 응답하는 데 4초 이상 걸린 수를 의미하며, 페이지 요청에 대한 처리는 붉은색 선으로 나타낸다. time tolerable와 time fail의 합이 클라이언트 요청에 따른 서버의 유효 응답 총수에 해당한다. Table 5의 실험 결과표에서는 각 요소를 비율(%)로 나타내었는데 이는 유효 응답 총수에 대한 각 요소의 비율을 나타낸 것이다. 실험 결과표의 Power 항목은 해당 실험에서 소모된 서버들의 전력량을 의미한다. 실험 결과표의 처리율(%)은 서버 전원모드 제어 방법마다 유효 응답 총수가 다른데 항상 ON 방법 대비 해당 방법의 유효 응답 총수 처리 비율을 의미한다.

특정 방법에서의 time good 비율은 그 해당 방법의 유효 응답 총수 대비 2초 이내 응답한 수의 비율을 나타내는데 이는 해당 방법에서의 서비스품질에 해당한다. 각 방법에서의 유효 응답 총수가 각기 다르므로 항상 ON 방법 대비 해당 방법의 서비스품질을 정의하여 비교하려고 한다. 이를 실험 결과표에서 표준화된 서비스품질(normalized QoS)이라고 부른다. 이 항목은 해당 방법의 서비스품질에 해당 방법의 처리율을 곱함으로써 계산된다.

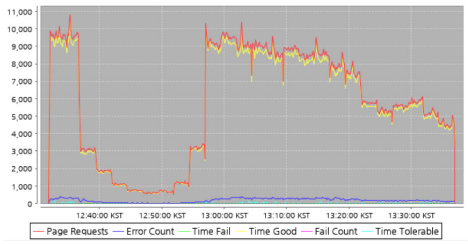
또한, 실험 결과표에서 단위전력당 good 응답 수 항목을 추가로 사용한다. 이는 해당 방법에서 서비스 품질에 해당하는 time good에 해당되는 응답 수를 그것을 처리하기 위해 소모한 전력량으로 나눈 값이다. 이는 전력 소모 측면에서 얼마나 양질의 good 응답인지 구분하기 위해 도입한다. 아래의 실험 결과표에서 표준화된 서비스품질과 단위전력당 good 응답 수를 서버 전원모드 제어방법과 비교할 때 중요한 의미를 갖는다[6].

Table 4. metric of Experiment

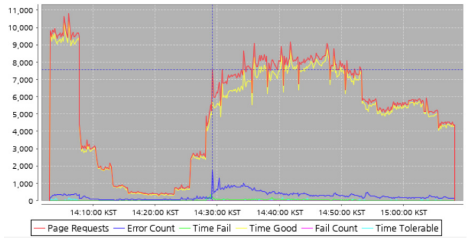
측정지표	지표에 대한 처리
처리율	(해당 유효 응답 총수 / 서버가 항상 켜있을 때 유효 응답 총수) * 100
표준화된 QoS	(time good 비율 * 처리율)/100
단위전력당 good 응답 수	(유효 응답 총수* time good 비율/100)/전력소비

Table 4는 SPECweb 실험 결과에서 얻은 성능지표를 나타낸다. 처리율은 always-on 상황에서의 유효 응답 총수와 각 알고리즘의 유효 응답 총수 비율(%)로 나타내었다. 표준화된 QoS는 always-on 상황에서 처리되는 QoS를 기준으로 하여 얻은 QoS를 의미한다. 또한, 단위전력당 good 응답 수는 전력당 처리되는 time good 비율에 해당하는 유효 응답 총수를 의미한다.

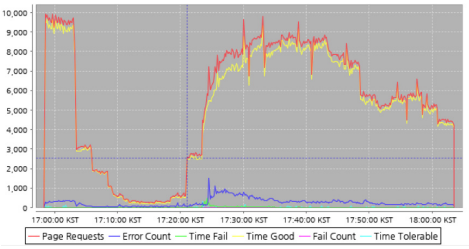
1) 뱅킹 부하패턴



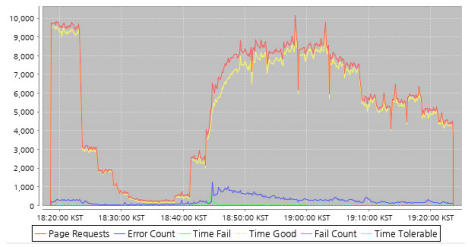
a) always-on



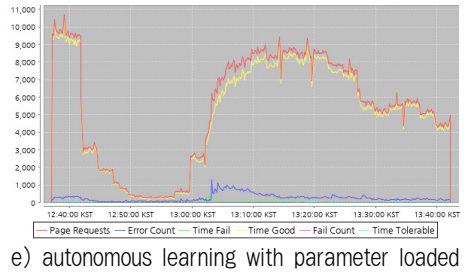
b) static



c) prediction



d) autonomous learning



e) autonomous learning with parameter loaded

Fig. 12. Result of Experiment

Table 5. Results of Experiment(banking load pattern)

서버 전원모드 제어방법	Power (W)	유효 응답 총수	Time (%)			처리율 (%)	표준화된 QoS (%)	단위 전력당 good 응답 수
			Good	Tolerable	Fail			
always-on	645.8	1902342	99.8	100.0	0.0	100.0	99.8	2939.82
static	406.1	1678156	99.3	99.5	0.5	88.22	87.6	4103.44
prediction	430.3	1736408	99.5	99.7	0.3	91.28	90.82	4015.17
autonomous learning	415.5	1712947	99.5	99.6	0.4	90.04	89.59	4102
autonomous learning with param loaded	401.4	1715508	99.5	99.6	0.4	90.18	89.73	4252.44

Table 6. Results of Experiment(real load pattern)

서버 전원모드 제어방법	Power (W)	유효 응답 총수	Time (%)			처리율 (%)	표준화된 QoS (%)	단위 전력당 good 응답 수
			Good	Tolerable	Fail			
always-on	617.3	667531	99.9	100	0	100	99.85	1080.29
static	214.8	459575	98.1	98.3	1.7	68.85	67.54	2098.9
prediction	219.1	471585	98	98.2	1.8	70.65	69.24	2109.33
autonomous learning	203.8	469491	98	98.2	1.8	70.33	68.92	2257.61
autonomous learning with param loaded	208.7	485422	97.9	98.2	1.8	72.72	71.19	2277.09

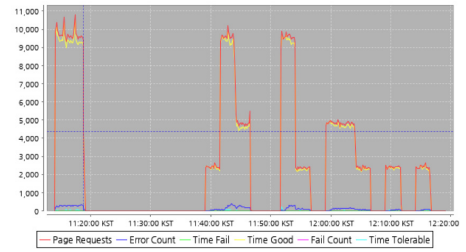
Fig. 12와 Table 5는 뱅킹 부하패턴에 대한 실험 결과의 그림과 비교표이다. 자율학습 방법은 정적 전원모드 제어방법과 비교해 단위전력당 good 응답 수 측면에서 1.4 낮았다. 예측기법과 비교해 단위전력당 good 응답 수 측면에서 86.8 높게 나왔다. 학습 오버헤드를 제외한 파라미터로드에 의한 자율학습 방법은 기존의 정적 전원모드 제어방법과 비교해 단위전력당 good 응답 수 측면에서 149 높게 나왔다. 예측기법과 비교해 단위전력당 good 응답 수 측면에서 237 높게 측정되었다. 실험 결과에 대한 자세한 분석은 4.4절을 보기 바란다.

2) 실제 부하패턴

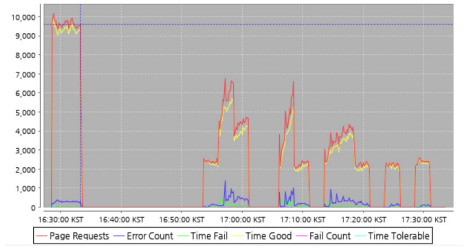
Fig. 13과 Table 6은 실제 부하패턴에 대한 실험 결과의 그림과 비교표이다. 자율학습 방법은 정적 전원모드 제어방법과 비교해 단위전력당 good 응답 수 측면에서 158 높게 나왔다. 예측기법과 비교해 단위전력당 good 응답 수 측면에서 148.2 높게 나왔다. 학습 오버헤드를 제외한 파라미터로드에 의한 자율학습 방법은 기존의 정적 전원모드 제어방법과 비교해 단위전력당 good 응답 수 측면에서 178 높게 나왔다. 예측기법과 비교해 단위전력당 good 응답 수 측면에서 167.7 높게 측정되었다. 실험 결과에 대한 자세한 분석은 4.4절을 보기 바란다.

3) 가상 부하패턴

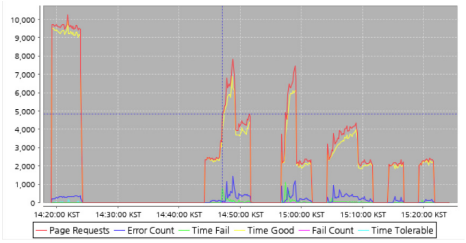
Fig. 14와 Table 7은 가상 부하패턴에 대한 실험 결과의 그림과 비교표이다. 자율학습 방법은 정적 전원모드 제어방



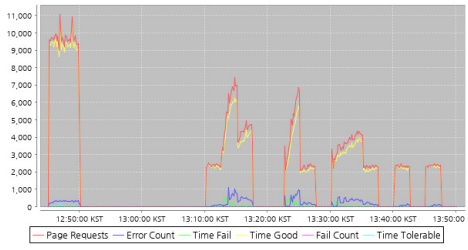
a) always-on



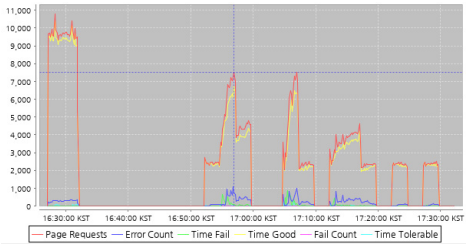
b) static



c) prediction



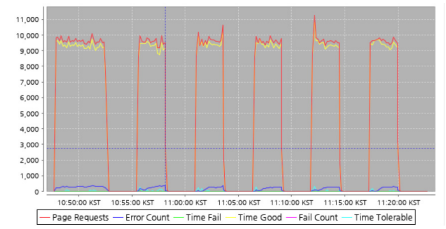
d) autonomous learning



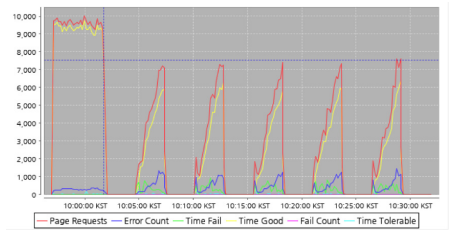
e) autonomous learning with parameter loaded

Fig. 13. Result of Experiment

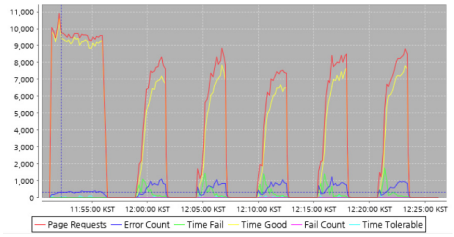
법과 비교해 단위전력당 good 응답 수 측면에서 775 높게 나왔다. 예측기법과 비교해 단위전력당 good 응답 수 측면에서 168.2 높게 나왔다. 학습 오버헤드를 제외한 파라미터로드에 의한 자율학습 방법은 기존의 정적 전원모드 제어방법과 비교해 단위전력당 good 응답 수 측면에서 822 높게 나왔다. 예측기법과 비교해 단위전력당 good 응답 수 측면



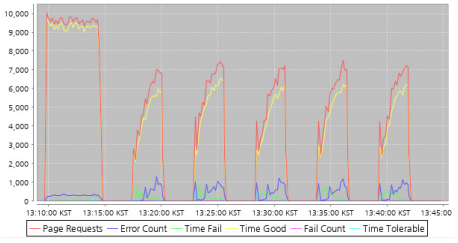
a) always-on



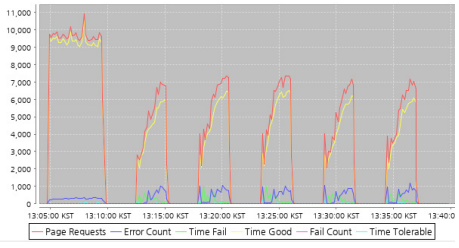
b) static



c) prediction



d) autonomous learning



e) autonomous learning with parameter loaded

Fig. 14. Result of Experiment

에서 216 높게 측정되었다. 실험 결과에 대한 자세한 분석은 4.4절을 보기 바란다.

4.4 분석 및 토론

1) QoS 측면에서의 결과 분석

자율학습 방법은 뱅킹 부하패턴에서 정적 전원모드 제어방법에 비해 1.99% 증가, 예측기법에 비해 1.23% 감소되었으며, 실제 부하패턴에서 정적 전원모드 제어방법에 비해 1.38% 증가, 예측기법에 비해 0.32% 감소하였다. 가상 부하패턴에서 정적 전원모드 제어방법에 비해 15.38% 증가, 예측기법에 비해 0.28% 감소하였다. 자율학습 방법이 뱅킹 부하패턴에서 예측기법에 비해 QoS 측면에서 낮은 이유는 부하 증가하는 구간의 기울기가 예측기법에 적합하도록 구현되었기 때문으로, 가상 부하패턴과 같은 급증하는 구간에서는 QoS가 더 높음을 확인할 수 있다[7]. 자율학습은 정적 전원모드 제어방법에 비해 3가지 부하패턴에서 QoS 면에서 더 우수하지만, 예측기법에 비해서는 다소 떨어짐을 확인할 수 있다. 자율학습 방법의 Qos를 예측기법보다 개선하기 위해선 다중 임계값 자율학습을 적용을 통해 QoS와 단위전력당 good 응답 수를 개선하여야 한다.

Table 7. Results of Experiment(virtual load pattern)

서버 전원모드 제어방법	Power (W)	유효 응답 총수	Time (%)			처리율 (%)	표준화된 QoS (%)	단위 전력 당 good 응답 수
			Good	Tolerable	Fail			
always-on	325.6	764140	99.6	100	0	100	99.63	2338.48
static	149.6	308126	90	91	9	40.32	36.29	1853.7
prediction	171.6	454491	92.9	93.5	6.5	59.48	55.26	2460.5
autonomous learning	150.2	415167	95.1	95.6	4.4	54.33	51.67	2628.65
autonomous learning with param loaded	143.7	403973	95.2	95.6	4.4	52.87	50.33	2676.29

2) 전력 측면에서의 결과 분석

자율학습 방법은 뱅킹 부하패턴에서 정적 전원모드 제어방법에 비해 9.4Wh 더 소비하였고, 예측에 비해서 14.8Wh 감소하였다. 실제 부하패턴에서 정적 전원모드 제어방법에 비해 11Wh 감소하였고, 예측에 비해서 15.3Wh 감소하였다. 가상 부하패턴에서 정적 전원모드 제어방법에 비해 0.6Wh 더 소비하였으나, 예측에 비해서 21.4Wh 적게 소비하였다. 실제 부하패턴에서 정적 전원모드 제어방법에 비해 자율학습이 전력소비가 감소한 이유는 급격한 증가와 급격한 감소가 반복되는 워크로드에서 부하가 최하까지 떨어지기 때문에 네트워크연결 유지 시간이 짧아 서버가 빠르게 OFF 되었기 때문이다. 예측기법에 비해서 3가지 부하패턴 모두에서 QoS 감소와 함께 전력소비가 감소하였다. 이것은 알고리즘에서 소비전력당 처리량이 많은 상태를 학습 조건으로 사용하였고, 단일 임계 자율학습으로는 급격히 증가하는 부하에서 QoS 측면에서 예측보다 앞서지 못하기 때문이다.

3) 단위전력당 good 응답 수 측면에서의 결과 분석

자율학습 방법은 뱅킹 부하패턴에서 정적 전원모드 제어방법에 비해 1.4 낮고, 예측기법에 비해 86.8 높게 나왔다. 실제 부하패턴에서 정적 전원모드 제어방법에 비해 자율학습이 158.7 높고, 예측기법에 비해 148.3 높게 나왔다. 가상

Table 8. Results of Experiment

부하의 유형	표준화된 QoS(%)	전력(Wh)	단위전력당 good 응답 수
banking load pattern	0.14% 증가	14.1Wh 감소	150.4 증가
real load pattern	2.27% 증가	4.9Wh 증가	19.48 증가
virtual load pattern	1.34% 감소	6.5Wh 감소	47.64 증가

부하패턴의 경우 정적 전원모드 제어방법에 비해 774.9 높고, 예측에 비해 168.1 높게 측정되었다. 뱅킹 부하패턴의 정적 전원모드 제어에서만 낮게 나타난 이유는 워크로드의 부하 급증 후 부하의 감소가 비교적 천천히 감소하고 상태 가중치가 트래픽에 일부 할당되어 이용률이 정적 전원모드 제어에 비해 느리게 OFF 되기 때문이다. 네트워크 유입량이 감소하면서 서버의 부하가 감소하지만, 네트워크연결 수는 keepalive로 인해 세션 유지기간이 길기 때문에 부하의 감소 시간보다 느리게 연결 수가 떨어지기 때문이다[16].

4) 파라미터 로드 된 자율학습의 결과 분석

학습 방법의 경우 계속 적용되어 학습이 충분히 이루어지면 런타임에 학습된 결과를 이용하여(추가 학습 과정 없이) 수행될 수 있다. 이러한 경우 성능 향상이 기대되는데 이를 검증하기 위해 파라미터 로드에 의한 자율학습 방법에 대한 실험을 수행하였다. 자율학습 방법 대비 파라미터 로드에 의한 자율학습 방법이 얼마만큼 향상되었는지를 Table 6, 7, 8에 있는 데이터를 기준으로 계산 정리한 것이 Table 8이다.

Table 8은 자율학습과 파라미터 로드에 의한 전원모드 제어 간의 차이를 통해 런타임 실행에 의한 오버헤드를 알아보기 위한 실험의 결과이다. 실험 결과는 3가지 부하패턴에 대해서 모두 단위전력당 good 응답 수면에서는 모두 향상됨을 보였다.

Fig. 15는 부하의 입력에 따른 서버 수의 변화를 나타낸 그림으로 런타임 실행에 자율학습을 한 경우와 실행시점에 파라미터 로드 된 전원모드 제어를 한 경우의 서버 수 변화에 대한 비교이다. 런타임 실행에 자율학습한 경우는 파라미터 로드 된 전원모드 제어보다 추가적으로 서버를 더 켜거나, 끄는 경우가 발생하는데, 실험 초기 warmup구간에서 학습으로 인해 서버 수가 조정되고, 급격한 상승 구간 등에서 서버 수 변화가 더 발생함을 확인할 수 있다.

일반적으로 파라미터로드를 이용한 방법이 전력을 덜 소비하게 되어, 단위전력당 good 응답 수 향상에 반영되었다. 다만, Table 8의 실제 부하패턴에서 소비전력이 더 증가한 것을 볼 수 있는데, 상승구간에서 적합한 상태 가중치가 찾아져 서버가 더 빨리 켜지고 QoS는 더 향상되었지만, Fig. 13의 실제 부하패턴과 같이 부하의 하강 후 부하 유지기간이 짧아, 서버를 더 켠 만큼 소비전력이 더 증가했기 때문이다. 이는 서버의 부하의 감소속도보다 keepalive시간으로 인해 네트워크 연결 수 감소가 더 느리기 때문이다. 이를 개선하기 위해선 graceful shutdown과 같은 동적 종료방법을 통해 서버를 빨리 끌 수 있게 된다면 소비전력이 더 감소하게 되어 단위전력당 good 응답 수가 더 향상될 것으로 보인다.

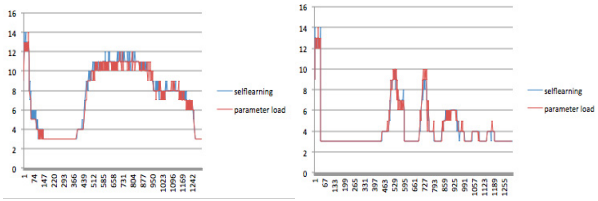


Fig. 15. Number of Activated Servers (banking load pattern, real load pattern)

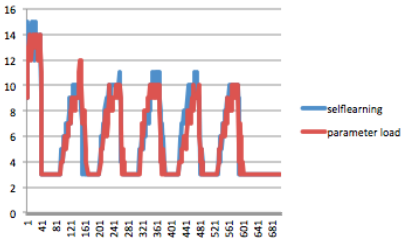


Fig. 16. Number of Activated Servers (virtual load pattern)

가상 부하패턴과 같은 급격한 부하의 변화가 있는 경우에서 표준화된 QoS가 감소한 이유는 Fig. 16에서와 같이 런타임 학습 과정 중에 고정된 임계값에서 상태 가중치를 찾아가는 동안 서버를 추가 활성화하는 동작으로 인해 파라미터 로드 된 전원모드 제어보다 QoS가 높게 된 경우에 해당한다. 이러한 현상은 다단계 임계값 자율학습을 적용할 경우 QoS를 더 높게 하므로 개선되고, 파라미터 로드 된 다단계 전원모드 제어에서 오버헤드 제거 효과가 더 커져 문제가 해결된다. 다만 여러 형태의 부하패턴에 적용할 경우 전력소비가 함께 증가할 수 있으므로 이를 고려한 세심한 설계를 통해 단위전력당 good 응답 수를 더 향상시킬 수 있을 것으로 본다.

5. 결론 및 향후 연구 방향

본 논문에서는 자율학습 방법에 단위전력당 good 응답 수를 높이기 위한 서버 전원모드 제어를 제안하였다. 부하의 형태를 일반적인 경우의 뱅킹 부하패턴, 실제 부하패턴, 특별한 경우의 가상적인 경우로 나누어 실험을 진행하였다. 제안방법을 구현하여 16개의 서버 클러스터 환경에서 3가지 다른 부하패턴들을 이용하여 실험을 수행하였다. 실험 결과는 제안방법의 에너지 효율성이 뛰어남을 보여주고 있다. 기존의 정적 서버 전원모드 제어방법과 비교할 때 뱅킹 부하패턴, 실제 부하패턴, 가상 부하패턴에서 단위전력당 good 응답 수가 각각 99.9%, 107.5%, 141.8%이고, 기존의 예측방법과 비교할 때 뱅킹 부하패턴, 실제 부하패턴, 가상부하 패턴에서 단위전력당 good 응답 수가 각각 102.0%, 107.0%, 106.8%이다. 학습오버헤드를 제외하여 비교하면, 실험 결과는 기존의 정적 서버 전원모드 제어방법과 비교할 때 뱅킹 부하패턴, 실제 부하패턴, 가상 부하패턴에서 단위전력당

good 응답 수가 각각 103.6%, 108.4%, 144.3%이고, 기존의 예측방법과 비교할 때 뱅킹 부하패턴, 실제 부하패턴, 가상 부하 패턴에서 단위전력당 good 응답 수가 각각 105.9%, 107.9%, 108.7%이다.

현재의 실험은 리눅스 서버클러스터를 구성하여 부하의 유형에 따라 동적 재구성을 하도록 구성하였으나, 상용서버의 cpu의 고속화, 메모리의 고용량화, 디스크의 단위용량당 비용이 저렴해져감에 따라 가상화 환경에서 서버 클러스터를 지원하도록 할 필요가 증가하고 있다. 가상환경에서 동적 마이그레이션을 지원하는 실험을 통해 좀 더 많은 가상 서버환경에서 동적 재구성 및 전력을 절감하는 연구가 필요할 것으로 본다.

References

[1] SangHak Lee, SungJun Mun, JinHwan Kim, SangYong Shin, YongWon Seo, and Young-Jin Choi, "The Establishment Method of Green Data Center in Public Sector," *Journal of Korea Information Science Society*, Vol.27 No.11, pp.48-57, 2009.

[2] Chenguang Liu, Jianzhong Huang, Qiang Cao, Shenggang Wan, and 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, and Z. Huang, "Metrics and task scheduling policies for energy saving in multicore computers," 11th IEEE/ACM International Conference on Grid Computing(GRID), 2010.

[4] G. Chen et. al, "Energy-Aware Server Provisioning and Load Dispatching for Connection-Intensive Internet Services," NSDI'08 Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation, 2008.

[5] Hoyeon Kim, Chihwan Ham, Hukeun Kwak, Hulung Kwon, Youngjoung Kim, and Kyusik 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.

[6] Taejune Ahn, Sungchoul Cho, Seokkoo Kim, Kyongho Chun, and Kyusik Chung, "A Flexible Multi-Threshold Based Control of Server Power Mode for Handling Rapidly Changing Loads in an Energy Aware Server Cluster," *The Journal of KIPS*, Vol.3, No.9 pp.279-292, 2014.

[7] Heungsik Moon, Sungchul Cho, Hukeun Kwak, and Kyusik Chung, "The Expectation of Power Consumption for Improving QoS in a Server Cluster Environment," JCCI, 2013.

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

[9] [Internet], http://www.austintek.com/LVS/LVS-HOWTO/HOWTO/LVS-HOWTO.ipvsadm.html

[10] H. Kwak, A. Sohn, and K. Chung, "Autonomous Learning of Load and Traffic Patterns to Improve Cluster Utilization," *Cluster Computing*, Vol.14, Issue.4, Dec., 2011.

[11] Dongjun Kim, Hukeun Kwak, Hujung Kwon, Youngjong Kim, and Kyusik Chung, "An Improved Estimation Model of Server Power Consumption for Saving Energy in a Server Cluster Environment," *Journal of Korea Information Processing Society*, Vol.19A, Issue.3, pp.139-146, 2012.

[12] python [Internet], https://www.python.org

[13] Apache [Internet], http://www.apache.org/.

[14] SPECweb [Internet], http://www.spec.org/benchmarks.html/.

[15] InternetTrend [Internet], http://www.internettrend.co.kr

[16] H. Kim, C. Ham, H. Kwak, and K. Chung, "Dynamic Shutdown of Server Power Mode Control for Saving Energy in a Server Cluster Environment," *The Journal of KIPS*, 2013.

[17] Hukeun Kwak, Kyusik Chung, Hyung Won Choi, and Andrew Sohn, "Enabling Scalabe Cloud Infrastructure using Autonomous VM Migration," 2012 IEEE 14th International Conference on High Performance Computing and Communications.

조 성 철

e-mail : sccho@q.ssu.ac.kr
1998년 호서대학교 전자공학과(학사)
2000년 숭실대학교 전자공학과(석사)
2002년~2004년 한단정보통신
2007년~2008년 아시아나IDT
2009년~2011년 스트라텍
2011년~현 재 숭실대학교 전자공학과 박사과정
관심분야 : 임베디드 시스템 및 네트워크 컴퓨팅

곽 후 근

e-mail : gobarian@q.ssu.ac.kr
1998년 숭실대학교 전자공학과(석사)
1998년~2006년 숭실대학교 전자공학과 (박사)
1998년~2000년 (주) 3R 부설연구소 주임연구원
2003년~2013년 펌킨네트워크 기술이사
2014년~현 재 유진 CTO
관심분야 : 네트워크 컴퓨팅 및 보안

정 규 식

e-mail : kchung@q.ssu.ac.kr
1979년 서울대학교 전자공학과(학사)
1981년 한국과학기술원 전산학과(석사)
1986년 미국 University of Southern California(컴퓨터공학석사)
1990년 미국 University of Southern California(컴퓨터공학박사)
1998년~1999년 미국 IBM Almaden 연구소 방문 연구원
1990년~현 재 숭실대학교 스마트시스템 소프트웨어학과 교수
관심분야 : 임베디드 및 네트워크 컴퓨팅