KIPS Tr. Comp. and Comm. Sys.
Vol.4, No.2 pp.47~56 pISSN: 2287-5891

http://dx.doi.org/10.3745/KTCCS.2015.4.2.47

Prediction of Power Consumption for Improving QoS in an Energy Saving Server Cluster Environment

Sungchoul Cho† · Sanha Kang†† · Hungsik Moon††† · Hukeun Kwak†††† · Kyusik Chung†††††

ABSTRACT

In an energy saving server cluster environment, the power modes of servers are controlled according to load situation, that is, by making ON only minimum number of servers needed to handle current load while making the other servers OFF. This algorithm works well under normal circumstances, but does not guarantee QoS under abnormal circumstances such as sharply rising or falling loads. This is because the number of ON servers cannot be increased immediately due to the time delay for servers to turn ON from OFF. In this paper, we propose a new prediction algorithm of the power consumption for improving QoS under not only normal but also abnormal circumstances. The proposed prediction algorithm consists of two parts: prediction based on the conventional time series analysis and prediction adjustment based on trend analysis. We performed experiments using 15 PCs and compared performance for 4 types of conventional time series based prediction methods and their modified methods with our prediction algorithm. Experimental results show that Exponential Smoothing with Trend Adjusted (ESTA) and its modified ESTA (MESTA) proposed in this paper are outperforming among 4 types of prediction methods in terms of normalized QoS and number of good reponses per power consumed, and QoS of MESTA proposed in this paper is 7.5% and 3.3% better than that of conventional ESTA for artificial load pattern and real load pattern, respectively.

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

에너지 절감형 서버 클러스터 환경에서 QoS 향상을 위한 소비 전력 예측

조 성 철† · 강 산 하†† · 문 흥 식††† · 곽 후 근†††† · 정 규 식†††††

요 약

에너지 절감형 서버 클러스터 환경에서는 서버 전원 모드가 부하상황에 따라 제어된다. 다시 말하면 현재 부하를 처리하는 데 필요한 대수의 서버들만 ON하고 나머지 서버들은 OFF한다. 이 알고리즘은 정상적인 상황에서는 잘 동작하지만 부하가 급증 또는 급감하는 비정상적인 상황에서는 QoS를 보장할 수 없다. 왜냐하면 서버가 OFF에서 ON으로 바뀌는 데 필요한 지연시간 때문에 ON 서버 대수를 당장 증가시킬 수 없기 때문이다. 본 논문에서는 정상적인 상황뿐만 아니라 비정상적인 상황에서도 QoS를 향상시키는 새로운 소비 전력 예측 알고리즘을 제안한다. 제안된 예측 알고리즘은 기존 시계열 분석에 기반한 예측과 추세를 반영한 예측 조정의 두 부분으로 구성된다. 15대의 서버 클러스터를 이용하여 실험이 수행되었고, 4가지 유형의 기존의 시계열 예측 모델과 본 논문에서 제안하는 4가지 유형의 수정된 모델에 대해 성능을 비교하였다. 실험 결과 4가지 유형 중 추세조정 지수평활법(ESTA)과 본 논문에서 제안된 ESTA(MESTA)가 표준화된 QoS 및 단위전력당 좋은 응답수 측면에서 가장 우수한 성능을 보였으며, 또한 본 논문에서 제안한 MESTA 알고리즘이 기존의 ESTA 알고리즘에 비해 가상 부하패턴과 실제 부하패턴에 대해 QoS가 7.5%, 3.3% 각각 향상됨을 보여주었다.

키워드 : 전원 모드 제어, QoS, 소비 전력, 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 고려 대상이다. 실제로 2012년 ‘뉴욕타임스’의 보도에 따르면 미국 내 데이터 센터의 사용 전력량은 핵발전소 30개 용량에 해당하는 300억 와트(W)를 소비하고 있는데 이 중 90%는 수요가 폭증하거나 정전에 대비하기 위한 예비전력으로 실제로 사용되지 않는다. 최대 규모의 데이터 센터를 운영하고 있는 구글과 페이스북도 각각 3억 와트와 6000만 와트의 전력을 소비한다[1].

데이터 센터는 크게 IT 장비와 이를 안정적으로 운용하

※ 이 논문은 2012년도 정부(교육과학기술부)의 재원으로 한국연구재단의 기초연구사업 지원을 받아 수행된 것임(NRF-2012R1A1A2006602).
† 정 회 원 : 숭실대학교 정보통신전자공학부 박사과정
†† 비 회 원 : 숭실대학교 정보통신전자공학부 학사과정
††† 준 회 원 : 숭실대학교 정보통신전자공학부 학사
†††† 정 회 원 : 유진 CTO
††††† 정 회 원 : 숭실대학교 정보통신전자공학부 교수
Manuscript Received : June 4, 2013
First Revision : March 21, 2014; Second Revision : November 10, 2014
Accepted : November 10, 2014
* Corresponding Author : Kyusik Chung(kchung@q.ssu.ac.kr)

기 위한 기반 설비로 구성된다. 데이터 센터에서 운용되는 IT 장비는 데이터 센터마다 약간의 차이는 있으나 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[2].

데이터 센터에서 IT 장비 중 서버들의 에너지 절감을 위한 방법 중 여러 대의 서버들로 구성된 서버 클러스터에서 서버들 전체의 총 전력 소모를 최소화하도록, 에너지 절감형 부하 분산기를 이용하여 서버의 소비 전력에 따라 동작될 서버의 전원 모드(On/Off)를 제어함으로써 QoS(사용자가 느끼는 응답 속도)와 소비 전력을 절감하는 방법이 존재한다[3]. 그러나 해당 방법은 일반적인 상황에서는 잘 동작하나 비일반적인 상황(요청의 급증/급감)에서는 QoS를 보장할 수 없다. 왜냐하면 요청이 급증할 때는 해당 요청을 처리하기 위해 필요한 서버의 수가 갑자기 증가하는데 서버가 OFF에서 ON으로 바뀌는 데 필요한 지연시간(수초~수십 초 수준) 때문에 해당 요청을 바로 처리할 수 없기 때문이다. 이에 본 논문에서는 비일반적인 상황에서 QoS를 보장하기 위해 소비 전력 예측 알고리즘을 제안한다. 제안된 알고리즘은 필요한 소비 전력을 미리 예측하여 비일반적인 상황에서는 QoS를 보장한다.

본 논문의 구성은 다음과 같다. 2절에서는 에너지 절약을 위한 서버 전원 모드 제어에서 기존의 방법에 대한 연구와 문제점을 소개한다. 3절에서는 기존 방법의 성능보장 문제를 해결하기 위한 예측방법을 소개한다. 4절에서는 실험 및 토론을, 5절에서는 결론 및 향후 연구 방향을 제시한다.

2. 연구 배경

2.1 기존 에너지 절감형 연구[4]

Fig. 1은 기존 에너지 절감형 서버 클러스터링 시스템의 구조를 나타낸다. 클라이언트는 부하 분산기인 LVS(Linux Virtual Server, 오픈소스 부하 분산기)[5]에게 요청을 하고, LVS는 서버들에게 설정된 스케줄링에 따라 사용자의 요청을 분산한다. 이때, 요청을 받은 서버(스케줄링에 따라 선택된 서버)는 사용자에게 직접 응답(Direct Routing, 부하 분산기를 거치지 않고 서버가 사용자에게 직접 응답)을 보낸다[6].

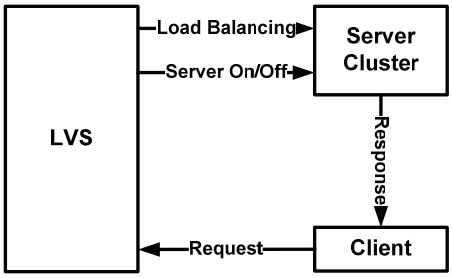

Fig. 1. Structure of the existing system

에너지 절감형 서버 클러스터링 시스템의 동작 과정은 다음과 같다.

1) 현재 운영 중인 각 서버의 현재 소비 전력을 계산하고, 소비 전력이 많은 서버는 부하를 적게 주고, 소비 전력이 적은 서버는 부하를 많이 주어 균등한 부하 분산을 유지한다.
2) 총 소비 전력, 현재 운영 중인 서버의 수, 서버당 최대 가용 전력을 이용하여 서버의 전원 모드 제어의 방향을 결정한다. 이는 총 소비 전력을 서버당 최대 가용 전력으로 나누어 운영될 서버의 수를 결정하여 현재 운영 중인 서버의 수에서 추가/제거를 결정하게 된다.
3) 서버의 추가는 서버의 전원 모드를 Off 모드에서 On 모드로 전환한다는 의미이며, Off 모드의 서버 중에서 종료 시점이 가장 오래된 서버를 On 모드로 전환한다.
4) 서버의 제거는 서버의 전원 모드를 On 모드에서 Off 모드로 전환한다는 의미이며, 이때 서버의 TCP 접속량이 가장 적은 서버를 우선적으로 선정하여 부하 분산을 제거한 뒤 TCP 접속이 모두 종료되기를 기다린 다음 TCP 접속이 모두 종료되면 서버를 Off 모드로 전환한다.

2.2 기존 연구의 문제점

기존 연구의 경우 일반적인 상황(사용자 요청이 일반적인 경우)에서는 QoS와 전력 절감을 보장한다. 그러나 비일반적인 상황(사용자 요청의 급증/급감)에서는 QoS가 보장되지 않거나 일부 서버로 요청이 몰려 서버가 죽는 현상이 발생한다. 기존 연구의 문제는 다음의 2가지로 요약할 수 있다.

1) 서버가 OFF에서 ON으로 바뀌는 데 필요한 지연시간(수초~수십 초 수준) 때문에, 현재 ON된 서버의 처리능력 이상으로 서비스 요청량이 급증할 경우 그 급증된 요청에 대한 서비스를 정상적으로 할 수 없다는 문제점
2) 서버들의 사용량이 급증하는 상황에서 부팅 상태가 길어지면 일부 서버들에게 서비스 요청이 몰릴 수 있는 문제점

2.3 접근 방식

본 논문에서는 기존 연구의 문제점을 해결하기 위해 소비 전력 예측 알고리즘을 제안한다. 제안된 소비 전력 예측 알고리즘에서는 필요한 서버의 수를 미리 예측하여 미리 On 함으로써 비일반적인 상황(사용자 요청의 급증/급감)에서도 QoS를 보장할 있다. 또한 일부 서버로 요청이 몰려 서버가 다운되는 현상을 막을 수 있다.

3. 제안하는 시스템

3.1 소비 전력 예측 알고리즘

제안된 소비 전력 예측 알고리즘은 LVS로 들어오는 InByte(사용자로부터 부하 분산기로 유입되는 요청량의 바

이트 수)의 총량을 조건으로 시계열 모델을 활용하여 증가할 Byte를 예측하며 이를 바탕으로 필요한 서버의 수를 결정한다. 기존의 시계열 모델은 자기유사성을 띈 서비스 요청 패턴일 때의 예측을 위한 모델이며 일반적인 패턴의 경우에 적합하다. 하지만 기존의 시계열 모델은 급작스러운 변화에 둔감한 모델로 실제 급격한 서비스 요청 패턴이나, DDoS 상황에서 추세를 반영하지 못하여 예측값이 정확하지 못한 문제점이 있다[7]. 기존의 시계열 모델들을 정리하면 Table 1과 같다.

Table 1. Existing time series model

Existing Model	Property
이동평균법 (Moving Average)	수열의 일정 범위를 계속적으로 평균을 낸다. 서비스 요청량에 명확한 추세가 없을 경우 가장 유용하다. 변화 추세가 있을 경우 예측값의 오차가 발생하게 된다. 10초 전부터의 InByte의 총량을 이동평균으로 구한다. $Value$는 10초 뒤의 Byte량을 예측한 것이며, n은 첨자의 시점의 Byte량이다. 예를 들면, n_{-9}은 9초 전의 Byte량이다. $Value = \frac{n_{-9}+n_{-8}+n_{-7}+\ldots+n_{-2}+n_{-1}+n}{10}$
가중이동평균법 (Weighted Moving Average)	이동평균법과 비슷하지만 과거의 수요보다 최근의 수요에 더 많은 가중치를 두어 강조할 수 있다. 이동평균법보다 변화 추세에 더 민감하다. 가중치의 총합을 100%로 놓고 10초 전부터 현재까지 매초 InByte량에 가중치를 다르게 두어 이동평균을 구한 뒤 InByte 증가 혹은 감소량에도 각각 가중치를 두어 가감하였다. $Value = n_{-9}\times 4\% + n_{-8}\times 5\% + n_{-7}\times 6\% + n_{-6}\times 7\% + n_{-5}\times 8\% + n_{-4}\times 10\% + n_{-3}\times 12\% + n_{-2}\times 14\% + n_{-1}\times 16\% + n\times 18\%$
지수평활법 (Exponential Smoothing)	가중이동평균법과 비슷하지만 과거로 갈수록 가중치가 지수적으로 감소하여 적용된다. 현재의 값에 더욱 민감하다. 여기서 $Value_{ex}$는 직전 예측치를 의미한다. $Value = n_{-9}\times 70\% + Value_{ex}\times 30\%$
추세조정 지수평활법 (Exponential Smoothing with Trend Adjusted)	지수평활법에 변화추세값을 반영하는 방법으로 수요 증가량 변화를 반영한다. 지수평활모델에 추세조정이 포함된 것이다. $Value = n_{-9} + 0.7\times n_{-9} - 0.4\times n_{-19} - 1.3\times n_{-29}$

급격한 변화에 대처하기 위해, 본 논문에서는 다음과 같은 보완된 알고리즘을 제시한다. 제안하는 방법은 2개의 파트로 나누어볼 수 있다. 앞부분의 Time Series Part는 기존의 시계열 예측 모델을 적용한 부분이고 Trend Adjusted Part는 추세를 반영한 모델이다.

$$Prediction = (Time\ Series\ Part) + (Trend\ Adjusted\ Part)$$

Time Series Part는 기존의 시계열 모델을 이용한 부분이다. Time Series Part만으로는 비일반적인 상황(요청의 급증/급감)일 때 대응하기 어렵다. 때문에 본 논문에서는 추세를 반영하는 부분을 추가하였다. 여기서 측정 구간 길이는 10초이고, $\Delta n_{-i} = n_{-i} - n_{-i-1}$이다. 여기서, 10초를 선택한 이유는 서버가 Sleep 상태에서 On 상태로 바뀌어 요청을 처리할 때까지 걸리는 시간을 충분히 고려하여 예측하기 위함이다. Trend Adjusted Part는 비일반적인 상황일 때 Prediction에서 차지하는 비중이 커지게 되며 서비스 요청량 급증에 대처 능력이 좋아지게 된다. 각각의 가중치는 10초 전부터 6초 전까지는 30%의 가중치를 두었고, 5초 전부터 현재까지는 70%의 가중치를 두었는데 과거보다 현재에 가중치를 더 두기 위함이다. 본 논문에서는 기존 시계열 모델에 Trend Adjusted Part를 추가하는 방식을 제안하는데 각각을 MMA(Modified Moving Average), MWMA(Modified Weighted Moving Average), MES (Modified Exponential Smoothing), MESTA(Modified Exponential Smoothing with Trend Adjusted)로 부른다. 즉, 제안된 알고리즘(Modified)은 Table 1의 Timer Series Part에 아래의 수식으로 표현되는 Trend Adjusted Part가 각각 더해져서 만들어진다.

$$\text{추세반영} = (k\text{초 전부터 매초사이의 서비스 요청 증가량}) \times (\text{각각의 가중치의 합}) \times k$$

$$Value = [\Delta n_{-9}\times 4\% + \Delta n_{-8}\times 5\% + \ldots + \Delta n_{-1}\times 16\% + \Delta n\times 18\%]\times 10$$

예측된 서비스 요청량(Prediction)의 간단한 예(MWMA)는 다음과 같다. Table 2는 일반적인 상황(요청의 급증/급감이 없음)에서의 Prediction을 보여준다. 표를 보면 일반적인 상황의 경우 Trend Adjusted Part의 비중이 작은 것을 알 수 있다. 이에 반해 Table 3은 비일반적인 상황(요청의 급증/급감이 있음)에서 Prediction을 보여준다. 이 예제들을 보면 비일반적인 상황의 경우 Trend Adjusted Part의 비중이 큰 것을 알 수 있다.

Table 2. Example of normal situation without sharp rise/fall in requests

0초 전	1초 전	2초 전	3초 전	4초 전	5초 전	6초 전	7초 전	8초 전	9초 전	10초 전
35	37	38	34	35	36	39	34	35	37	38

$$Prediction = (Time\ Series\ Part) + (Trend\ Adjusted\ Part) = 35\times 18\% + \ldots + 37\times 4\% + [(35-37)\times 18\% + \ldots + (37-38)\times 4\%]\times 10 = 36 + (-2.7) = 33.3$$

Table 3. Example of abnormal situation with sharp rise/fall in requests

0초 전	1초 전	2초 전	3초 전	4초 전	5초 전	6초 전	7초 전	8초 전	9초 전	10초 전
96	86	76	56	46	36	39	34	35	37	38

$$Prediction = (Time\ Series\ Part) + (Trend\ Adjusted\ Part) = 96\times 18\% + \ldots + 37\times 4\% + [(96-86)\times 18\% + \ldots + (38-37)\times 4\%]\times 10 = 63.88 + (83.1) = 146.98$$

3.2 동작 방식

소비 전력 예측 알고리즘을 이용한 시스템의 전체적인 동작 과정은 Fig. 2와 같다.

Fig. 2. Procedure of our proposed method

1) 단계 1 - InByte량 대비 서버 소비 전력의 증가량 계산

각 서버로부터 소비 전력을 수신하여 초당 측정되는 InByte량 대비 서버 소비 전력 증가량을 일정 시간(5초) 동안 수집하여 평균을 얻는다. 이 값은 매초마다 다음과 같이 구한다.

$$\Sigma \frac{(\text{총소비전력})}{InByte} / (\text{측정시간}) = Coefficient$$

2) 단계 2 - 예측 알고리즘 적용

부하 분산기에서 측정 하는 InByte량을 기준으로 앞에서 설명한 네 가지 예측 알고리즘 중 한 가지를 적용하여 10초 후의 InByte량을 계산하여 이를 단계 1에서 산출한 Coefficient값과 곱하여 10초 후의 서버 소비 전력값을 예측한다.

$$(\text{예측된 } InByte \text{ 량}) \times Coefficient = (\text{예측된서버소비전력})$$

3) 단계 3 - 전원 모드가 변환되어야 할 서버와 해당 서버의 전원 모드 결정

단계 2에서 예측된 서버 소비 전력값을 이용하여 전원 모드 제어 주기(5초)마다 전원 상태가 변환되어야 할 서버와 해당 서버의 On/Off 상태를 결정한다. 예측된 서버 소비 전력값을 매 주기마다 평균값을 계산하여 최소의 서버 수를 결정하고, 각 서버들의 소비 전력을 이용하여 On/Off될 서버를 선정한다.

4) 단계 4 - 결정된 모드에 따라 해당 서버의 전원 동작 제어

단계 3에서 결정된 서버의 On/Off 동작을 제어하며, On/Off될 서버에 대한 부하 분산을 조절한다. 이때 제안된 알고리즘은 전력 정보에 기반한 에너지 절감형 부하 분산 알고리즘[8]을 사용함으로써 서버의 부하 분산 가중치를 변경하여 부하 분산을 조절한다. Off 모드의 서버를 On 모드로 변환하는 경우에는 변환된 서버에 더 많은 부하를 분산하여 다른 서버와의 부하량을 균등하게 한다. 한편 On 모드의 서버를 Off 모드로 변환하는 경우에는 서버의 모드를 변환하기 전에 부하 분산 리스트에서 제거함으로써 부하의 분산을 막고, 서버의 연결이 완전히 종료되었을 때 서버의 모드를 Off로 변환한다.

4. 실험 및 토론

4.1 실험 환경

실험을 위해 리눅스 서버 클러스터 환경을 구축하였다. Switch를 이용하여 Real Server, Client(Prime Client), Besim (Backend Simulator)을 각각 분리되도록 네트워크를 구성하였다. Prime Client는 클라이언트가 여러 대일 때 이를 관리하고 Besim은 클라이언트의 동작(로그인, 계좌 이체 등)을 제어한다. Switch-LVS-Client 간의 네트워크 연결은 1GB Ethernet을 사용하였다. 네트워크 구성은 Direct Routing(서버가 LVS를 거치지 않고 클라이언트에 직접 응답하는 방식)을 사용하여 네트워크 내 병목현상을 최소화하였다. 서버 클러스터에 설치된 전력 측정기는 전체 Real Server(실제 사용 중인 서버)의 전력값을 측정하여 Zigbee(무선 통신 프로토콜)[9]를 통해 전력 측정 PC로 전송한다.

Table 4는 실험에 사용된 하드웨어와 소프트웨어를 나타낸다. 부하 분산 방식은 WRR(Weighted Round Robin : 가중치를 두어 라운드 로빈을 적용)[10]을 사용하였고, SPECweb[11]의 Client와 Prime Client패키지를 1대에 설치하였다. 웹 서버는 Apache(오픈 소스 웹 서버)[12]를 사용하였다.

Table 4. Hardware and software specification for experiments

Set	Hardware		Software	서버 대수
	CPU(Hz)	RAM (MB)		
Load Balancer	Intel Core2 Quad Q-6600, 2.4Ghz	4GB	LVS (WRR)	1
Client (Prime Client)	Intel i7-2600k, 3.4GHz	8GB	SPECweb	1
Besim	Intel i5-760, 2.8Ghz	4GB	Apache/ SPECweb	1
Real Server	Pentium-4, 2.26G	2GB	Apache	15

4.2 실험 방법

실험에 사용하고 있는 SPECweb은 원하는 패턴흐름을 생성하여 실제 클러스터 환경에서 발생하는 시간대별 트래픽 양에 맞추어 트래픽을 발생시킬 수 있는 전문 벤치마킹 툴이다. SPECweb에서는 Banking과 E-Commerce, Support 시나리오를 제공함으로써 실제 클러스터에서 사용자 요청

패턴과 유사하게 트래픽의 종류 또한 조절이 가능하다. 본 실험은 SPECweb Banking 시나리오를 이용하여 2가지로 구분하여 진행하였는데, 첫 번째는 급격하게 변화하는 가상 트래픽 패턴을, 두 번째는 실제 부하 패턴을 사용하였다. 가상 트래픽 패턴에 대한 실험 방법은 Table 5와 같고 실험에 사용한 사용자 입력 패턴은 Table 3과 같다. 사용자 입력 패턴은 최악의 입력 상황을 고려하여 임펄스(Impulse) 패턴을 사용하였으며(최소 50 Kbytes-100명의 사용자, 최대 800 Kbytes-1600명의 사용자), 60분간 19번의 부하량 변화를 주었다. 여기서 임펄스 패턴은 SPECweb의 Banking 패턴(온라인 은행 업무에 대한 시뮬레이션)을 이용하였다. 실제 부하 패턴에 대한 실험은 Table 4의 입력 패턴을 이용하여 진행하였다. 가상 트래픽 패턴을 이용한 실험 3가지와 실제 부하 패턴을 이용한 실험 1가지, 총 4가지 실험을 진행하였는데 각각의 단계를 설명하면 다음과 같다.

1) 실험-1

항상 On된 경우와 예측을 적용하지 않은 기존 알고리즘[4]과 추세조정 부분이 빠진 4가지 예측 알고리즘, 추세조정 부분을 포함한 4가지 예측 알고리즘을 비교하였다. 실험에 사용된 “Thread Ramp Up Time”이란 최대 부하가 발생되는 데 걸리는 시간으로 이 시간이 짧으면 최대 부하에 걸리는 시간이 짧은 것을 의미한다. “Warm-up Time”은 최대 부하에서 실제 부하를 측정하기 이전에 유지되는 시간이고, “Thread Ramp Down Time”은 Thread Ramp Up의 반대로 부하 발생을 중지하는 시간을 의미하고, “Warm-down Time”은 비부하 유지 구간의 시간을 의미한다. 실험-1에서는 적용한 알고리즘만 다를 뿐, 실험 조건은 모두 동일하다.

2) 실험-2

추세조정을 포함하지 않는 ESTA 알고리즘과 MESTA 예측 알고리즘을 비교하였다. MESTA 알고리즘은 단위전력당 좋은 응답수 측면에서 가장 우수한 알고리즘이다(4.4절 참조). 실험-2는 이 2개의 알고리즘에 대해 “Thread Ramp Up Time”을 변화시켜가면서 실험을 수행하였다. 이 실험의 수행 목적은 얼마만큼의 급작스러운 부하까지 제안된 알고리즘이 효율적으로 동작하는지를 판단하기 위함이다.

3) 실험-3

추세조정 부분을 포함하지 않는 ESTA 알고리즘과 MESTA 예측 알고리즘을 추가로 비교하였다. 이 실험에서는 2개의 파라미터를 변화하였다. 하나는 “Thread Ramp Up Time”으로 분석을 통해 짧은 시간을 사용하였으며, 나머지 하나는 높은 부하량에도 제안된 예측 알고리즘이 동작하는지를 알아보기 위해 사용자 부하량을 증가시켰다. 사용자 부하량은 서버와 클라이언트와의 Keep-Alive 연결방식을 적용하여 사용자당 서버 접속 시간을 증가시켰다.

Table 5. Description of experiments

Experiment	Thread Ramp Up Time(s)	Degree	Algorithm	Keep-Alive (s)
Experiment-1	10	89.2	allways-on/ Without prediction	15
			MA/MMA	
			WMA/ MWMA	
			ES/MES	
			ESTA/ MESTA	
Experiment-2	10	89.2	ESTA	15
	30	87.7		
	50	86.2		
	70	84.7		
	90	83.2		
	110	81.7		
	10	89.2	MESTA	
	30	87.7		
	50	86.2		
	70	84.7		
	90	83.2		
	110	81.7		
Experiment-3	10	89.2	ESTA	20
	20	88.5		
	30	87.7		
	10	89.2	MESTA	
	20	88.5		
	30	87.7		

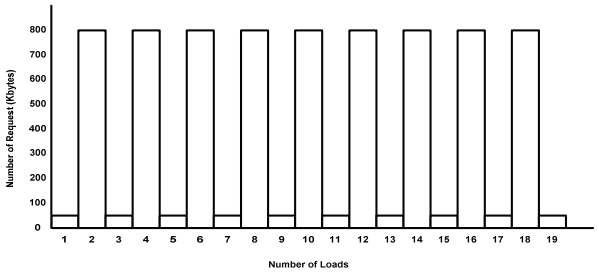

Fig. 3. Impulse pattern of user requests

4) 실험-4

InternetTrendTM[10]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 DB화되어 일반에게 공개되어있다. 따라서 원하는 카테고리와 시간대를 입력하면 해당 트래픽 추세를 확인할 수 있다.

본 실험에서 사용한 실제 부하 패턴은 InternetTrendTM[10]의 Website Performance 카테고리의 주문집중시간대 자료 중 2012/9/1~2012/9/7까지의 금융/부동산/경제 파트의 시간대별 집중도(Fig. 4)를 따라서 생성하였다.

실험시간 : 30분(그래프 하단의 24시간은 하루를 나타내

며, 전체 실험 소요시간을 30분으로 전환하여 실험), 최대 session : 1200

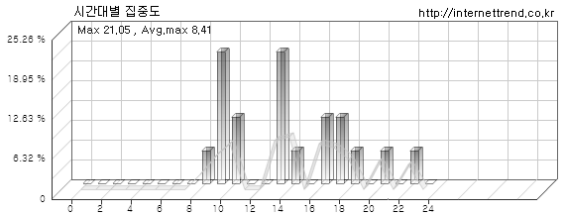

Fig. 4. InternetTrendTM - Website Performance

Session : 0 0 0 0 0 0 0 0 300 1200 600 0 0 1200 300 0 600 600 300 0 300 0 300 0

4.3 실험 결과

1) 실험-1

Table 6은 실험-1의 결과를 나타낸다. 최대 부하를 얻는 데 걸리는 시간(Thread Ramp Up Time)은 10초로서 빠르게 최대 부하를 얻는 것을 의미하며, 이를 기울기로 환산하면 89.2도이다. 실험 결과를 보면 추세조정을 적용하지 않는 기존 알고리즘에 비해 제안된 알고리즘이 모두 QoS가 향상되었음을 알 수 있다. 그러나 소비 전력 측면에서 보면 소비 전력이 증가하였음을 알 수 있다. 이는 QoS가 증가함에 따라 처리량이 늘어나 소비 전력이 증가하였음을 의미한다. 이에 대한 종합적인 분석은 4.4절을 참고하기 바란다.

Table 6. Results of experiment-1

Thread Ramp Up Time(s)	Degree	Algorithm	Measurement		Keep-Alive(s)	유효 응답총수	처리율	표준화된 QOS	단위전력 good
			QoS(%)	Power (W)					
10	89.2	Always -on	98.4	1522	15	238702	100	98.4	154.3
		Without predction	75.1	661		154547	63.3	47.5	169.8
		MA	85.9	683		177076	74.1	63.7	222.7
		WMA	84.9	679		176705	74.0	62.8	220.9
		ES	86.5	686		184875	77.4	66.9	233.1
		ESTA	87.5	691		186749	78.2	68.4	236.4
		MMA	86.6	688		179043	75.0	64.9	225.3
		MWMA	87.2	687		195103	81.7	71.2	247.6
		MES	87.0	690		184883	77.4	67.3	233.1
		MESTA	89.5	705		202541	84.8	75.9	257.1

2) 실험-2

Table 7는 실험-2에 대한 실험 결과를 나타낸다. 여기서 알고리즘은 4.4절의 분석을 토대로 ESTA 알고리즘과 MESTA 알고리즘을 사용하였다. 실험-2는 최대 부하를 얻는 데 걸리는 시간(Thread Ramp Up Time)에 변화를 주면서 실험을 수행하였고, 최대 부하를 얻는 데 걸리는 시간이 빠른 경우(10초)와 느린 경우(110초) 등을 고려하여 실험을 수행하였다. 실험 결과를 보면 최대 부하는 얻는 데 걸리는 시간과 무관하게, 추세보정을 포함하지 않은 ESTA 알고리즘에 비해 MESTA 알고리즘이 표준화된 QoS 측면에서 우수함을 알 수 있다. 그러나 소비 전력 측면에서는 QoS 증가로 인해 소비 전력도 증가하였음을 알 수 있다. 이에 대한 효율 측면에서의 자세한 분석은 4.4절 (2)를 참고하기 바란다.

Table 7. Results of experiment-2

Thread Ramp Up Time(s)	Degree	Algorithm						Keep -Alive(s)
		ESTA						
		QoS(%)	Power(W)	유효 응답총수	처리율	표준화된 QOS	단위전력 good	
10	89.2	87.5	691	186749	78.2	68.4	236.4	15
30	87.7	91.4	715	180925	75.7	69.3	231.2	
50	86.2	93.3	722	182432	76.4	71.3	235.7	
70	84.7	95.6	729	191509	80.2	76.6	241.1	
90	83.2	96.0	733	188401	78.9	75.7	246.7	
110	81.7	97.3	745	194985	81.6	79.4	254.6	
		MESTA						
10	89.2	89.5	705	202541	84.8	75.9	257.1	
30	87.7	88.8	704	204244	85.5	75.9	257.6	
50	86.2	91.1	714	200090	83.8	76.3	255.2	
70	84.7	93.4	721	205097	85.9	80.2	265.6	
90	83.2	95.5	729	202834	84.9	81.1	265.7	
110	81.7	97.0	738	205183	85.9	83.3	269.6	

3) 실험-3

Table 8은 실험-3에 대한 실험 결과를 나타낸다. 실험-3은 최대 부하를 얻는 데 걸리는 시간을 짧게 유지하고, 부하량을 증가시켜 실험을 수행하였다. 부하량의 변화는 Keep-Alive를 20으로 증가하여 사용자가 서버에 접속하는 시간을 늘려서 수행하였다. Table 8의 결과를 Table 7의 같은 Thread Ramp Up Time(10초, 30초)에 대한 결과와 비교해 보면 Keep-Alive가 20초로 증가하여 상대적으로 유효 응답 총수가 증가하고, 이로 인해 표준화된 Qos가 증가함을 알 수 있고, 또한 서버처리 시간이 길어짐에 따라 전력 소모가 증가한 것으로 보인다.

실험 결과를 보면 최대 부하를 얻는 데 걸리는 시간과 무관하게, 추세조정을 하지 않은 ESTA 알고리즘에 비해 MESTA 알고리즘이 표준QoS 측면에서 우수함을 알 수 있다. 그러나 소비 전력 측면에서는 QoS 증가로 인해 소비 전력도 증가

하였음을 알 수 있다. 이에 대한 단위전력당 좋은 응답수 측면에서의 자세한 분석은 4.4절 (3)을 참고하기 바란다.

Table 8. Results of experiment-3

Thread Ramp Up Time(s)	Degree	Algorithm						Keep -Alive(s)
		ESTA						
		QoS(%)	Power(W)	유효 응답 총수	처리율	표준화된 QOS	단위전력 good	
10	89.2	83.6	762	227251	92.7	77.5	249.4	20
20	88.5	85.3	776	236742	96.6	82.5	260.4	
30	87.7	86.5	789	239272	97.6	86.0	262.6	
		MESTA						
10	89.2	85.3	771	235991	96.3	82.1	261.2	
20	88.5	87.2	781	241912	98.7	86.1	270.1	
30	87.7	89.8	798	239551	97.7	87.8	269.6	

4) 실험-4

Table 9는 실험-4에 대한 실험 결과를 나타낸다. 실험-4는 실제 부하를 추세조정을 포함하지 않은 예측과 추세조정을 포함한 예측 알고리즘별로 부하를 발생하여 각각의 QoS와 전력을 측정하였다. Banking workload에서 사용하는 기본 파라미터들을 적용하여 얻은 결과로 Thread Ramp Up time은 180초, warm up time은 300초, think time은 10초로 설정되며, 실험-1에서와 마찬가지로 추세조정을 하지 않은 기존 방법에 비하여 예측방법들이 모두 QoS가 향상된 반면에 전력 소모는 증가하였다. 이에 대한 단위전력당 좋은 응답수 측면에서의 자세한 분석은 4.4절 (4)를 참고하기 바란다.

Table 9. Results of experiment-4

Thread Ramp Up Time(s)	Degree	Algorithm	Measurement		Keep -Alive(s)	유효 응답총수	처리율	표준화된 QOS	단위전력 good
			QoS (%)	Power (W)					
180	76.45	Always-on	98.6	679	15	126903	100.0	98.6	184.2
		Without predction	92.7	322		68365	53.8	49.9	196.8
		MA	93.4	325		108516	85.5	79.8	311.8
		WMA	94.8	331		95008	74.8	70.9	272.1
		ES	95.0	331		106834	84.1	80.4	306.7
		ESTA	95.6	333		116740	91.0	87.3	335.0
		MMA	94.9	329		110730	87.2	82.8	319.4
		MWMA	96.2	334		93544	73.7	70.0	269.4
		MES	96.3	337		118509	93.3	89.9	328.5
		MESTA	96.7	345		119027	93.7	90.6	333.6

4.4 분석 및 토론

Fig. 5의 SPECweb workbench 그래프는 실험 결과를 보여준다. Time good으로 표시된 노란색 선은 클라이언트의 요청에 2초 이내 응답받은 수를, Timer Tolerable은 4초 이내 응답수, 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 10의 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 전원 모드 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 응답수 항목을 추가로 사용한다. 이는 해당 방법에서 서비스품질에 해당하는 Time good에 해당되는 응답수를 그것을 처리하기 위해 소모한 전력량으로 나눈 값이다. 이는 전력 소모 측면에서 얼마나 양질의 good 응답인지 구분하기 위해 도입한다. 아래의 실험 결과표에서 표준화된 서비스품질과 단위전력당 좋은 응답수가 서버 전원 모드 제어 방법과 비교할 때 중요한 의미를 갖는다.

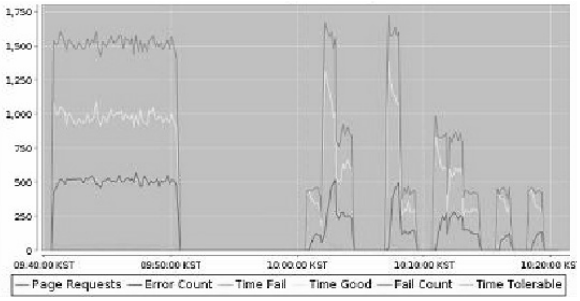

Fig. 5. Result of real workload - MESTA

1) 실험-1

Table 10은 실험-1의 가상 부하에 대한 실험 결과이고 단위전력당 좋은 응답수에 대한 비교를 보여준다. 가상 부하에 thread ramup 파라미터를 적용하고, 추세조정하지 않은 알고리즘과 추세조정한 알고리즘의 실험 결과 차이를 통해 향상 정도를 확인할 수 있다.

Table 10은 MESTA가 표준화된 Qos 측면에서 가장 우수하고 단위전력당 좋은 응답수 면에서 가장 우수함을 보여준다. 추세조정하지 않은 알고리즘과 추세조정한 알고리즘의 차이를 통해 전력, 표준화된 QoS, 단위전력당 좋은 응답수의 증감 정도를 확인할 수 있다. 표준화된 QoS 측면에서 MESTA가 가장 우수한 QoS를 보여주고 표준화된 QoS의 향상 정도가 상대적으로 높아 전력 증가와 무관하게 단위전력당 좋은 응답수 면에서 역시 가장 우수함을 보여준다.

Table 10. Results of Experiment-1

서버 전원 모드 제어 방법	Power (W)	유효 응답 총수	Time (%)			처리율 (%)	표준화된 QoS (%)	단위전력당 good 응답수
			Good	Tolerable	Fail			
allways-on	1522	238702	98.4	98.9	1.1	100	98.4	154.3
without predition	661	154547	75.1	92.2	7.8	63.3	47.5	169.8
MA	683	177076	85.9	95.9	4.1	74.1	63.7	222.7
WMA	679	176705	84.9	95.2	4.8	74.0	62.8	220.9
ES	686	184875	86.5	95.6	4.4	77.4	66.9	233.1
ESTA	691	186749	87.5	96.4	3.6	78.2	68.4	236.4
MMA	688	179043	86.6	93.3	6.7	75.0	64.9	225.3
MWMA	687	195103	87.2	94.1	5.9	81.7	71.2	247.6
MES	690	184883	87.0	95.0	5.0	77.4	67.3	233.1
MESTA	705	202541	89.5	96.8	3.2	84.8	75.9	257.1

2) 실험-2

Table 11은 최대 부하를 얻는 데 걸리는 시간(Thread Ramp Up Time)의 변화에 대한 실험 결과로 실험-2에 대한 단위전력당 좋은 응답수에 대한 비교를 보여준다. thread ramup 파라미터의 변화에 따른 추세조정하지 않은 알고리즘과 추세조정한 알고리즘의 실험 결과 차이를 통해 향상 정도를 확인할 수 있다.

Table 11의 ESTA와 MESTA에서 thread ramup 110에서 각각 QoS가 최대가 됨을 확인할 수 있고, 함께 증가한 전력 대비 단위전력당 좋은 응답수가 가장 우수함을 확인할 수 있다. 동일 thread ramup 시점에 추세조정하지 않은 알고리즘과 추세조정한 알고리즘 간의 차이를 통해 전력, 표준화된 QoS, 단위전력당 좋은 응답수의 증감 정도를 확인할 수 있다.

Table 11에서 ESTA와 MESTA를 비교해보면 유효 응답 총수가 상대적으로 높아 표준화된 QoS 측면에서 MESTA가 더 우수함을 확인할 수 있으나, thread ramup이 최대까지 증가하면서 표준화된 QoS 향상 정도와 단위전력당 좋은 응답수가 감소함을 볼 수 있다. 이를 통해 기울기가 완만해져갈수록 Qos와 단위전력당 좋은 응답수 차이가 축소되어감을 확인할 수 있다.

Table 11. Results of Experiment-2

Thread Ramp Up Time(s)	Power (W)	유효 응답 총수	Time (%)			처리율 (%)	표준화된 QoS (%)	단위전력당 good 응답수
			Good	Tolerable	Fail			
ESTA								
10 Sec	691	186749	87.5	96.4	3.6	78.2	68.4	236.4
30 Sec	715	180925	91.4	95.9	4.1	75.7	69.3	231.2
50 Sec	722	182432	93.3	96.4	3.6	76.4	71.3	235.7
70 Sec	729	191509	95.6	97.2	2.8	80.2	76.6	241.1
90 Sec	733	188401	96.0	97.9	2.1	78.9	75.7	246.7
110 Sec	745	194985	97.3	98.2	1.8	81.6	79.4	254.6
MESTA								
10 Sec	705	202541	89.5	96.8	3.2	84.8	75.9	257.1
30 Sec	704	204244	88.8	97.2	2.8	85.5	75.9	257.6
50 Sec	714	200090	91.1	94.1	5.9	83.8	76.3	255.2
70 Sec	721	205097	93.4	95.3	4.7	85.9	80.2	265.6
90 Sec	729	202834	95.5	98.0	2.0	84.9	81.1	265.7
110 Sec	738	205183	97.0	98.1	1.9	85.9	83.3	269.6

3) 실험-3

Table 12는 실험-3에서 thread ramup 시간이 짧은 구간 중에서 Keep-Alive가 20으로 증가한 경우의 성능비교표이다. 실험-3에 대한 알고리즘 차이에 따른 전력, 표준화된 Qos, 단위전력당 좋은 응답수 향상 정도를 보여준다.

Table 12는 최대 부하를 얻는 데 걸리는 시간이 짧은 경우(10초, 20초, 30초)에는 표준화된 Qos가 증가하고, 단위전력당 좋은 응답수가 증가함을 보여준다. ESTA와 MESTA 간의 알고리즘 차이를 보여주는 결과로 thread ramup 시간이 증가함에 따라 표준화된 Qos의 차이와 단위전력당 좋은 응답수 향상 정도 차이가 감소함을 알 수 있다.

Keep-Alive를 20으로 늘린 경우 유효 응답 총수와 표준화된 QoS가 증가하고 전력 소비 역시 증가함을 볼 수 있다. Table 12에서 볼 수 있듯이 기울기가 완만해져갈수록 QoS와 단위전력당 좋은 응답 차이가 축소되어감을 확인할 수 있다.

Table 12. Results of Experiment-3

Thread Ramp Up Time(s)	Power (W)	유효 응답 총수	Time (%)			처리율 (%)	표준화된 QoS (%)	단위전력당 good 응답수
			Good	Tolerable	Fail			
ESTA								
10 Sec	762	227251	83.6	95.2	4.8	92.7	77.5	249.4
20 Sec	776	236742	85.3	95.9	4.1	96.6	82.5	260.4
30 Sec	789	239272	86.5	96.4	3.6	97.6	86.0	262.6
MESTA								
10 Sec	771	235991	85.3	96.8	3.2	96.3	82.1	261.2
20 Sec	781	241912	87.2	95.0	5.0	98.7	86.1	270.1
30 Sec	798	239551	89.8	96.3	3.7	97.7	87.8	269.6

4) 실험-4

Table 13은 실험-4의 실제 부하에 대한 알고리즘별 효율성에 대한 비교표를 보여준다. 뱅킹워크로드의 기본 실험 파라미터를 이용하여 30분 동안 수행된 실험에서 모든 서버가 On 상태일 때 소모되는 소비 전력량은 679Wh이었다. MESTA의 표준화된 QoS는 90.6으로 가장 우수하게 나왔고, ESTA의 표준화된 QoS는 87.3의 결과를 얻었다.

Table 13에서 단위전력당 가장 우수하게 나온 ESTA와 MESTA를 비교해보면, MESTA가 ESTA에 비해 전력을 12Wh 더 소비하였고, QoS 측면에서는 3.3% 우수하였으며, 단위전력당 좋은 응답수 면에서는 1.4만큼 낮아 근소한 차이를 보였다. Table 13에서 WMA와 MWMA에서 표준화된 QoS 측면에서 -0.9%, ESTA와 MESTA에서 단위전력당 좋은 응답수 측면에서 -1.4만큼 낮게 나온 것은 실제 워크로드에서 부하의 상승하강 정도의 변화가 심해 발생한 결과로 보인다. 이전의 실험-1에서 MESTA가 가상 워크로드에서 단위전력당 좋은 응답수 측면에서 가장 우수함을 보여주었고, 실험-4의 실제 워크로드에서 MESTA가 QoS 측면에서 가장 우수했던 결과를 보면, ESTA와 단위전력당 좋은 응답 측면에서 차이가 크지 않음을 알 수 있다.

Table 13. Results of Experiment-4

서버 전원 모드 제어 방법	Power (W)	유효 응답 총수	Time (%)			처리율 (%)	표준화된 QoS (%)	단위전력당 good 응답수
			Good	Tolerable	Fail			
allways-on	679	126903	98.6	99.0	1.0	100.0	98.6	184.2
without predition	322	68365	92.7	96.4	3.6	53.8	49.9	196.8
MA	325	108516	93.4	96.8	3.2	85.5	79.8	311.8
WMA	331	95008	94.8	96.2	3.8	74.8	70.9	272.1
ES	331	106834	95.0	97.2	2.8	84.1	80.4	306.7
ESTA	333	116740	95.6	96.9	3.1	91.0	87.3	335.0
MMA	329	110730	94.9	96.3	3.7	87.2	82.8	319.4
MWMA	334	93544	96.2	97.2	2.8	73.7	70.0	269.4
MES	337	118509	96.3	97.2	2.8	93.3	89.9	328.5
MESTA	345	119027	96.7	97.3	2.7	93.7	90.6	333.6

실험을 종합적으로 고려할 때, 제안된 알고리즘에서 MESTA 알고리즘이 가상 워크로드와 실제 워크로드에서 우수한 성능을 보여줌을 확인할 수 있었다. MESTA 알고리즘은 ESTA 알고리즘에 비해 실험-1의 가상 워크로드와 실험-4의 실제 부하패턴에 대해 평균적으로 좋은 응답수 측면에서 9.65, QoS 측면에서 5.4% 우수함을 보여준다.

Fig. 5는 MESTA 방법을 이용한 경우의 실험에 대한 SPECweb 그래프로, 표준화된 QoS는 90.6%, 소비 전력은 345Wh이었다.

5. 결론 및 향후 연구 방향

일반적인 데이터 센터에서는 서버들의 소비 전력을 절감하기 위해 필요한 서버만큼만 On하고 나머지 서버는 Off하는 방식으로 제어하는 전원 모드 제어 알고리즘을 사용한다. 해당 알고리즘은 일반적인 상황에서는 잘 동작하나, 비일반적인 상황에서는 QoS를 보장할 수 없다. 이에 본 논문에서는 비일반적인 상황에서 QoS 보장을 위해 소비 전력 예측 알고리즘을 제안하였다.

4가지 유형의 기존의 시계열 예측 모델과 본 논문에서 제안하는 4가지 유형의 수정된 모델에 대해 실험을 통해 성능을 비교하였다. 실험 결과 4가지 유형 중 추세조정 지수평활법(ESTA)과 본 논문에서 제안하는 수정된 ESTA(MESTA)가 표준화된 QoS 및 단위전력당 좋은 응답수 측면에서 가장 우수한 성능을 보였으며, 또한 본 논문에서 제안한 MESTA 알고리즘이 기존의 ESTA 알고리즘에 비해 가상 부하패턴과 실제 부하패턴에 대해 QoS가 7.5%, 3.3% 각각 향상됨을 보여주었다

향후 연구 방향으로는 하이브리드 예측 알고리즘의 연구가 필요하다. 다시 말하면, 일반적인 상황에서는 기존의 예측을 적용하지 않는 방법을 사용하고, 사용량이 급등하는 비일반적인 상황에서는 예측을 적용하는 방법을 적용한다. 이럴 경우 QoS와 소비 전력 측면에서 최적의 상태를 유지할 수 있을 것으로 보인다.

References

[1] The New York Times, http://www.nytimes.com/

[2] 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.

[3] 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.

[4] 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. (under review)

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

[6] Direct Routing, http://www.linuxvirtualserver.org/VS-DRouting.html

[7] S. Chung, Y. Chung, and J. Kim, "Network Routing by Traffic Prediction on Time Series Models," Vol.32, No.4, pp.433-442, 2005.

[8] Y. Kim, D. Kim, N. Kang, H. Kwon, H. Kwak, and K. Chung, "The Server Load Balancing Algorithm for Energy Saving," *The Journal of KIPM,* Vol.3, No.2, 2011.

[9] Zigbee, http://en.wikipedia.org/wiki/ZigBee

[10] WRR(Weighted Round Robin), http://en.wikipedia.org/wiki/Weighted_round_robin

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

[12] Apache, http://www.apache.org/

조 성 철

e-mail : sccho@q.ssu.ac.kr

1998년 호서대학교 전자공학과(학사)

2000년 숭실대학교 전자공학과(석사)

2002년~2004년 한단정보통신 과장

2007년~2008년 아시아나IDT 과장

2009년~2011년 스트라텍 차장

2011년~현 재 숭실대학교 전자공학과 박사과정

관심분야 : 임베디드 및 네트워크 컴퓨팅

강 산 하

e-mail : ncsana@q.ssu.ac.kr
2014년 숭실대학교 정보통신전자공학부 학사과정
관심분야 : 임베디드 및 네트워크 컴퓨팅

곽 후 근

e-mail : gobarian@q.ssu.ac.kr
1998년 숭실대학교 전자공학과(석사)
1998년~2006년 전자공학과(박사)
1998년~2000년 (주) 3R 부설연구소 주임연구원
2003년~20013년 펌킨네트워크 기술이사
2014년~현 재 유진 CTO
관심분야 : 네트워크 컴퓨팅 및 보안

문 흥 식

e-mail : moonhs85@q.ssu.ac.kr
2013년 숭실대학교 정보통신전자공학부(학사)
2013년~현 재 Seetech 연구원
관심분야 : 임베디드 및 네트워크 컴퓨팅

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~1999년 미국 IBM Almaden 연구소 방문연구원
1990년~현 재 숭실대학교 정보통신전자공학부 교수
관심분야 : 임베디드 및 네트워크 컴퓨팅