클라우드 컴퓨팅 인프라 구축을 위한 시스템 부하 및 자원에 관한 연구

정성재* · 배유미** · 장래영** · 성경*** · 소우영**

*마이호스팅, **한남대학교 컴퓨터공학과, ***목원대학교 컴퓨터교육과

A Study on the Linux System Load and Resource for Cloud Computing Infrastructure Development

Sung-Jae Jung* · Yu-Mi Bae** · Rae-Young Jang** · Kyung Sung*** · Woo-Young Soh**

*Myhosting, **Hannam University, ***Mokwon University

E-mail : posein@paran.com, yumidw@hanmail.net, rene402@hnu.kr, skyys04@mokwon.ac.kr, wsoh@hnu.kr

요 약

클라우드 컴퓨팅(Cloud Computing)이 주목받고 활성화되면서 핵심 기술인 서버 가상화(Server Virutualization)를 이용한 시스템 구성이 인기를 얻고 있다. 이러한 영향으로 운영 중인 시스템에 대한 분석 및 서버 가상화 기술을 이용한 자원의 활용이 크게 각광받고 있다.

본 논문에서는 리눅스 시스템을 기반으로 시스템에 발생되는 부하 확인 방법과 종류, CPU의 자원과 부하와의 관계 분석을 통하여 시스템 유휴자원 활용과 클라우드 컴퓨팅 인프라 구축을 위한 기반 기술을 제공하고자 한다.

ABSTRACT

While Cloud Computing catch attention, system configuration using Server Virtualization as a core technology gains popularity. Thus analyzing existing systems and utilizing resources using server virtualization techniques are spotlighted.

This paper presents a solid foundation for utilizing the system idle resources and constructing the cloud computing infrastructure by analyzing means and types of the system load and further analyzing the relationship between CPU resources and loads based on Linux system.

키워드

리눅스(Linux), 부하(Load), 자원(Resource), 프로세스(Process)

Ⅰ. 서 론

클라우드 컴퓨팅(Cloud Computing)이 주목받고 활성화되면서 핵심 기술인 서버 가상화(Server Virtualization) 기술을 활용한 시스템 구성이 보편화되고 있다. 서버 가상화 기술은 클라우드 컴퓨팅 환경을 위한 인프라 구성뿐만 아니라, 운영중인 시스템에 발생하는 유휴자원(Idle Resource)의 활용과 자원 통합을 통한 효율성 증대라는 측면이 강조되면서 더욱 각광을 받고 있다[1].

운영중인 시스템에 서버 가상화 기술을 도입하여 시스템을 통합하려면 시스템의 부하(Load)와 자원(Resource)에 대한 분석은 필수적이다. 시스템에 발생하는 부하는 크게 2가지로 분류하는데, 하나는 CPU 부하이고, 또 다른 하나는 I/O(Input/Output)부하이다. CPU 부하는 대규모의 연산처리처럼 디스크 입출력은 사용하지 않고,

CPU의 계산 속도에 의존하는 경우를 뜻한다. 보통 이러한 종류의 프로그램들이 CPU에 부하는 주는 프로그램이다. I/O 부하는 디스크에 대량으로 저장된 자료로부터 특정한 파일을 검색하는 것처럼 CPU가 아닌 디스크의 읽기 속도, 즉 입출력(I/O: Input/Output)에 의존하여 발생하는 부하를 말하고 보통 디스크가 빠를수록 시간이 짧아진다. 디스크에 의존하는 프로그램들이 I/O부하와 관련된 프로그램이라고 할 수 있다. 시스템에 발생되는 부하에 대한 정확한 판단이 선결되어야 시스템의 자원을 활용할 수가 있다[2].

본 논문에서는 리눅스 시스템을 기반으로 부하를 확인하는 방법과 종류, CPU 자원분석과 부하와의 관계에 대해 연구하여 시스템의 자원 활용 및 통합을 위한 기반 지식을 제공하고자 한다[3].

II. 시스템의 부하

멀티태스킹(MulitTasking) 운영체제는 동시에 서로 다른 여러 태스크(Task)를 처리할 수 있는데, 여러 태스크를 실행한다고 해도 실제는 CPU나 디스크를 공유한다. 즉 매우 짧은 시간 간격으로 여러 태스크를 전환후 처리하면서 멀티태스킹을 실현한다.

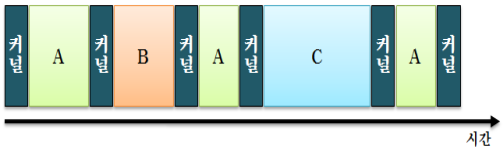

그림 1. CPU의 멀티태스킹 처리

실행 태스크가 적으면 운영체제는 태스크에 대기를 발생하지 않고 전환을 수행할 수 있으나, 실행해야할 태스크가 늘어나면 특정 태스크가 수행되는 동안, 다음 계산을 수행하고자 하는 다른 태스크들은 CPU에 시간이 날 때까지 대기하게 된다. 즉 처리를 실행하려고 해도 대기가 되는 상태를 프로그램의 실행지연이라 하며 리눅스에서는 그 결과를 Load Average라는 값으로 출력한다.

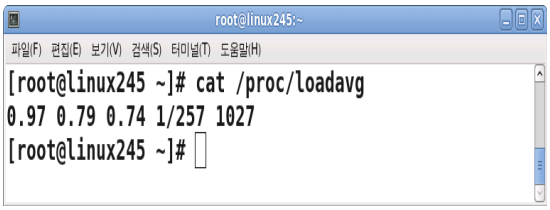

그림 2. 평균 부하의 확인

시스템의 평균 부하율은 top이나 uptime 명령 이외에 /proc/loadavg 파일에서도 확인할 수 있다[4]. 관련 항목은 총 5개의 필드로 구성되어 있고, 첫 번째부터 세 번째 필드까지는 최근 1분, 5분, 15분 동안의 동작중인 프로세스(state R, state D)의 평균 대기 개수를 출력한다. 4번째 필드는 /로 나뉘어져 있는데, 앞부분은 현재 실행된 프로세스 중 동작중인 실행 프로세스 수, 뒷부분은 현재 실행중인 프로세스 수를 나타낸다. 마지막 다섯 번째 필드는 가장 최근에 실행된 프로세스의 PID를 나타낸다. loadavg에 나타나는 값이 클수록 대기하는 태스크의 수가 많다는 것이고 부하가 높다는 것을 뜻하지만, 수치만으로 CPU 부하가 높은지, I/O부하가 높은지를 판단할 수는 없다. 따라서, 서버의 리소스 중 어떤 부분에서 병목현상이 발생하는지는 별도로 조사해야 한다.

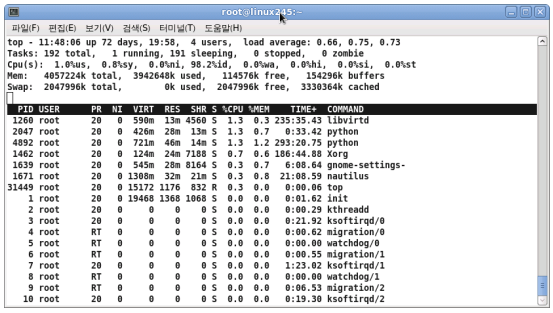

그림 3. top을 이용한 부하 및 자원 확인

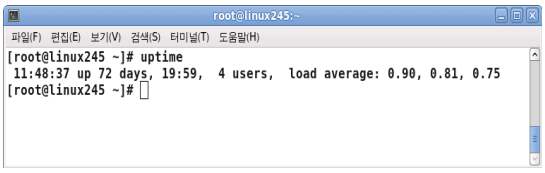

그림 4. uptime을 이용한 부하 확인

III. 프로세스 스케줄러와 상태

시스템의 부하라는 것은 태스크가 대기되는 것을 뜻하는 데, 이러한 대기를 제어하는 것이 리눅스 커널 내부에서 동작하는 프로세스 스케줄러(Process Scheduler)이다. 프로세스 스케줄러는 멀티태스킹의 제어에 있어서 실행할 태스크의 우선순위를 결정해서 태스크를 대기시키거나 재개하는 등 커널의 중추적인 역할을 수행한다. 리눅스 커널은 프로세스마다 프로세스 디스크립터(Process Descriptor)라는 관리용 테이블을 생성하고, 실행과 관련된 정보를 저장한다. 리눅스 커널은 프로세스 디스크립터군을 우선도가 높은 순으로 재배열하고, 프로세스 스케줄러는 알맞은 순서대로 실행되도록 조정하는 역할을 담당한다.

Load Average로 계산되는 대기 태스크의 평균수는 [표 1]에 열거된 여러 상태 중 2가지만 부하로 계산한다. TASK_RUNNING과

TASK_UNINTERRUPTIBLE만 수치로 계산되는데, 2가지 상태 모두 실행하려는 처리가 있어도 CPU의 할당을 받을 때까지 기다려야 한다는 공통점이 있다. TASK_RUNNING은 CPU를 사용하고자 해도 다른 프로세스가 CPU를 사용하고 있어서 기다리는 프로세스를 뜻하고, TASK_UNINTERRUPTIBLE은 계속 처리하고자 해도 디스크 입출력이 끝날 때까지 기다려야 하는 프로세스를 뜻한다. 따라서, 시스템의 부하는 CPU와 I/O중 어느 쪽에 원인이 있는지를 조사해서 대처해야 한다.

표 1. 프로세스 디스크립터의 상태

상태	설명
TASK_RUNNING	실행가능 상태. CPU에 시간이 나면 언제든지 실행이 가능한 상태
TASK_INTERRUPTIBLE	중단(interrupt)가능한 대기상태. 주로 복귀시간이 예측 불가능한 장시간의 대기상태. sleep이나 사용자로부터 입력 대기 등으로 발생.
TASK_UNINTERRUPTIBLE	중단 불가능한 대기상태. 주로 단시간에 복귀한 경우의 대기상태. 디스크 입출력 대기.
TASK_STOPPED	중지(suspend) 시그널을 받아서 실행 중단된 상태. 재개(resume)될 때까지 스케줄링되지 않음
TASK_ZOMBIE	좀비(zombie)상태. 자식 프로세스가 exit해서 부모 프로세스로 반환될 때까지의 상태

Ⅳ. CPU 사용률과 I/O 대기율

시스템의 부하와 연관되어 있는 CPU 사용률과 I/O 대기율은 sysstat라는 패키지에 포함되어 있는 명령어들로 손쉽게 확인할 수 있다[5]. sysstat는 iostat, mpstat, pidstat, sar 등의 명령어를 제공한다. iostat는 CPU의 평균 사용률과 디스크의 상태를 확인할 수 있는 명령어이다.

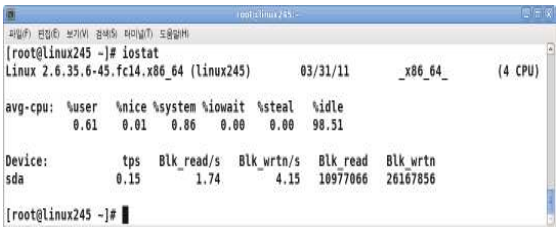

그림 5. iostat를 사용한 검사

mpstat는 실행한 순간의 CPU 사용률을 CPU나 코어별로 확인할 수 있는 명령어이다. 특정한 순간의 부하를 확인할 때 용이하다.

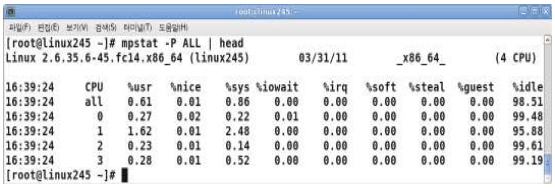

그림 6. mpstat를 이용한 검사

pidstat는 가장 최근에 출시된 Red Hat Enterprise Linux 6 나 Fedora 14에 설치된 sysstat 9 버전에 들어 있는 명령어로 동작중인 프로세스를 PID(Process Identifier)별로 CPU 사용률을 출력해준다. CPU 사용이 많은 프로세스를 검색할 때 유용하다.

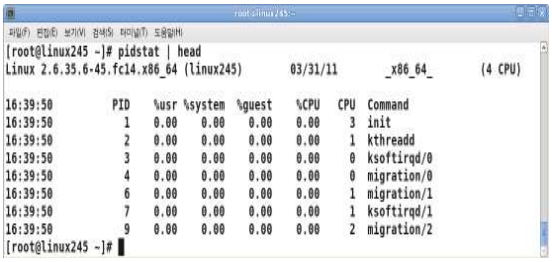

그림 7. pidstat를 이용한 검사

sar(System Activity Reporter)는 CPU 사용률과 I/O 대기율을 통합해서 출력해주는 도구로 프로그램을 실행한 날짜 기준으로 10분 주기로 출력해준다. 시스템의 상태를 수집하고 활동상태를 누적시켜 출력해주는데, 당일의 기록은 스케줄링 프로그램인 크론(Cron)에 의하여 /var/log/sa 디렉토리안에 기록된다. 예를 들어 30일에 누적한 데이터는 /var/log/sa/sar30 이라는 텍스트파일에 기록되어져서 수시로 확인할 수 있다.

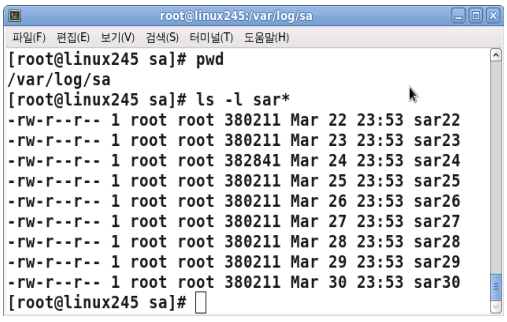

그림 8. sar의 누적 데이터파일

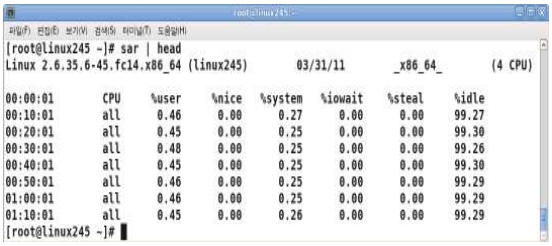

그림 9. sar의 실행 결과

sar의 실행결과인 그림 9에서 보면 CPU 부하와 관련된 부분이 %user와 %system이다. %user는 사용자 프로그램이 동작할 때 사용한 CPU로 보통 응용프로그램이 동작할 때 사용한 CPU라고 보면 된다. %system은 시스템 프로그램, 즉 커널이 동작할 때 사용한 CPU이다. 이 2개의 항목의 수치가 높게 나타나면 대기중인 프로세스의 부하 원인은 CPU 리소스 부족이라고 판단할 수 있다.

sar의 실행 결과중 %iowait의 수치가 높은 경우는 I/O로 인한 부하라고 볼 수 있다. Load Average의 수치가 높은데 %iowait의 수치가 상당히 높다면 대부분의 부하를 디스크 입출력과 관련된 부하라고 볼 수 있다. 이 경우에는 기존의 디스크를 더 빠른 디스크나 RAID 구성, 스토리지 구성 등을 통해 부하를 줄여줘야 한다.

Ⅴ. 결 론

클라우드 컴퓨팅과 서버 가상화 기술의 보편화는 운영중인 시스템들의 자원 활용과 통합이라는 과제를 부여하고 있다. 시스템의 유휴자원을 활용하고 시스템을 통합하기 위해서는 시스템에 발생되는 부하에 대한 정확한 분석이 필요하고, 분석시 발생되는 유휴자원이 많다면 적극 활용해야 한다. 본 논문에서 제시한 시스템 부하의 확인 및 분석 방법을 참고하고, 서버 가상화 기술을 활용한다면 시스템에 발생되는 유휴자원을 활용하여 시스템을 통합하고 자원의 효율성을 높여서 최적화된 시스템 관리를 할 수 있다. 아울러, 최근 각광을 받고 있는 클라우드 컴퓨팅 환경을 위한 인프라 구축에도 밑거름이 될 수 있으리라 여겨진다.

참고문헌

[1] 정성재, 배유미, 소우영, 성경, "x86시스템에 최적화된 서버 가상화 연구", 한국지식정보기술학회, 제 5권 ,제 5호, pp. 131-139, 10월 2010년.

[2] 이토 나오야, 카츠미 유키, 다나카 신지, 히로세 마사아키, 야스이 마사노부, 요코가와 카즈야, “서버/인프라를 지탱하는 기술“, 제이펍, pp. 162-184, 2009년

[3] 정성재, 배유미, 소우영, “보안성이 강화된 리눅스 가상화 기반 효율적인 웹 시스템 연구”, 보안공학연구지원센터, 제7권, 제4호, pp. 335-350, 8월 2010년.

[4] Procps, http://procps.sourceforge.net

[5] Sysstat, http://sebastien.godard.pagesperso-orange.fr