Red Hat Enterprise Linux 6

전원 관리 가이드

Red Hat Enterprise Linux 6에서 전력 소비 관리

엮음 1.0

Last Updated: 2017-10-12

Red Hat Enterprise Linux 6 전원 관리 가이드

Red Hat Enterprise Linux 6에서 전력 소비 관리
엮음 1.0

Don Domingo
Red Hat 엔지니어링 컨텐트 서비스

Rüdiger Landmann
Red Hat 엔지니어링 컨텐트 서비스
r.landmann@redhat.com

Red Hat Inc.

법적 공지

초록

다음 부분에서는 Red Hat Enterprise Linux 6에서 전력 소비를 효율적으로 관리하는 법에 대해 설명합니다. 다음은 (서버나 랩탑 모두에서) 전력 소비를 줄이는 여러 다른 방법들을 다루며, 각각의 기술이 시스템의 전체 성능에 어떤 영향을 미치는지에 대해 설명합니다.

차례

1장. 개요

전원 관리는 Red Hat Enterprise Linux 6를 위한 개선사항 중, 중요한 하나입니다. 컴퓨터 시스템이 사용하는 전력을 제한하는 것은 *그린 IT*(환경 친화적인 컴퓨팅)에 있어 가장 중요한 부분입니다. 그린 IT는 재활용 가능한 재료의 사용, 하드웨어 생산에 있어서 환경에 대한 영향의 고려, 시스템 설계와 설치에 있어 환경적 고려와 같은 제약사항을 함께 고려하는 것입니다. 이 문서에서는 Red Hat Enterprise Linux 6을 실행하는 시스템에 대한 전원 관리 가이드라인과 정보를 제공합니다.

1.1. 전원 관리의 중요성

전원 관리의 핵심은 각각의 시스템 구성요소의 전력 소모를 효율적으로 최소화하는 방법입니다. 따라서 시스템이 수행하는 여러 다른 작업을 연구하고, 각각의 구성요소의 성능이 정확히 그 작업을 수행하는 데 필요한 만큼을 보장하도록 설정하는 것이 필요합니다.

전원 관리와 관련된 주요 동기는 다음과 같습니다:

- 전체 전력 소모 감소를 통한 비용 감소

전원 관리를 적절히 사용해서 얻을 수 있는 이익은 다음과 같습니다:

- 서버와 컴퓨팅 센터의 발열 감소
- 냉각, 공간, 케이블, 발전기, 그리고 *무정전 전원 공급장치(uninterruptible power supplies)* 등의 2차 비용의 감소
- 랩탑의 배터리 수명 연장
- 더 적은 이산화탄소 배출
- 에너지 스타(Energy Star)와 같은 그린 IT와 관련된 정부 규제 또는 요구사항을 만족시킴
- 새로운 시스템에 대한 사내 가이드라인을 만족시킴

일반적으로 말하자면, 특정 구성 요소(혹은 전체 시스템)의 전력 소모를 줄이는 것은 발열을 줄이게 되고, 자연적으로 성능도 감소시키게 됩니다. 따라서, 특히 중요한 임무를 수행하는 시스템(mission critical system)의 경우, 사용자가 설정한 사항이 성능 감소에 어떻게 영향을 끼치는지를 철저히 연구하고, 테스트해야 합니다.

시스템이 수행하는 여러 다른 작업을 연구하고, 각각의 구성요소를 그러한 작업을 만족시키는 데 알맞은 성능만 발휘하도록 설정함으로써, 에너지를 절약하고, 발열을 줄이며, 랩탑의 배터리 수명을 최적화할 수 있습니다. 전력 소모와 관련된 시스템 분석과 튜닝의 원칙 중 많은 것은 성능 튜닝의 원칙과도 유사합니다. 시스템은 보통 성능이나 전원 절약 중 한쪽을 향해 최적화되기 때문에, 전원 관리와 성능 튜닝은 어느 정도는 시스템 설정에 있어 반대방향이라고 할 수 있습니다. 이 문서에서는 최적화 과정에서 Red Hat이 개발한 기술과 도구에 대해 설명합니다.

Red Hat Enterprise Linux 6은 이미 디폴트로 활성화되어 있는 많은 전원 관리 기능을 포함하고 있습니다. 그런 특징들은 전형적인 서버나 데스크탑 사용 환경에서는 성능에 영향을 끼치지 않도록 선택된 것입니다. 하지만, 최대 처리성능이나 최단 응답시간, 또는 최고 CPU 성능을 절대적으로 요구하는 매우 구체적인 사용 예에서는 이러한 디폴트 설정을 검토해야할 필요가 있을 수 있습니다.

이 문서에 나와있는 기술을 사용해 컴퓨터를 최적화해야 할지를 결정하기 위해서, 다음과 같은 질문을 스스로에게 던져보십시오:

질문 최적화를 해야만 하는가?

답변 전원 최적화의 중요성은 회사가 지켜야 하는 가이드라인이나 꼭 만족시켜야만 하는 규제가 있는지 여부에 따라 달라집니다.

질문 **어느 정도까지 최적화할 필요가 있나?**

답변 우리가 제공하는 기술중 몇몇은 시스템에 대한 감사나 분석을 끝까지 자세히 진행하지 않고도 전형적인 경우 전력 소비 효율을 높여줄 수 있는 몇 가지 일반적인 최적화를 제공합니다. 이러한 것들이 물론 수동으로 분석 및 최적화된 시스템에 필적할 만큼 좋지는 않겠지만, 좋은 대안이 될 수 있습니다.

질문 **최적화가 시스템의 성능을 용납할 수 없을 정도로 떨어뜨리지는 않을 것인가?**

답변 이 문서에서 설명하는 대부분의 기술은 시스템 성능에 상당한 영향을 끼칩니다. 만약 Red Hat Enterprise Linux 6에 이미 있는 디폴트 외의 전원 관리를 구현하기로 했다면, 전력 최적화를 한 다음에 시스템의 성능을 모니터링해서 성능 감소가 허용할만한 수준인지를 결정해야만 합니다.

질문 **시스템 최적화를 위해 투자해야 하는 시간과 자원이 달성할 수 있는 이익을 상회하는가?**

답변 단일 시스템을 수동으로 최적화하는 것은 들어가는 시간과 비용이 해당 기계의 수명동안 얻을 수 있는 이익보다 훨씬 높기 때문에 보통은 수행할 만한 가치가 있지 않습니다. 반면, 예를 들어 사무실에서 동일한 설정을 사용하는 10000대의 데스크탑 시스템을 관리하고, 단일한 최적 설정을 만들어서 모든 10000대의 기계에 적용하는 경우라면, 수동 최적화가 좋은 방안이 될것 입니다.

다음 절에서는 최적의 하드웨어 성능이 에너지 소비 측면에서 시스템에 어떤 이익을 가져오는가를 살펴볼 것입니다.

1.2. 전원 관리의 기본

효율적인 전원 관리는 다음 원칙 위에 만들어 집니다:

유휴 CPU는 필요할 때만 깨워야 합니다

Red Hat Enterprise Linux 5 커널은 각각의 CPU에 대한 주기적인 타이머를 사용합니다. 이 타이머로 인해서 CPU는 완전한 유휴상태로 들어갈 수 없습니다. 왜냐하면, CPU가 다른 프로세스가 실행중인지 여부와 관계 없이 각각의 타이머 이벤트(설정에 따라 몇 밀리초당 한 번씩 발생합니다)를 처리해야만 하기 때문입니다. 효과적인 전원 관리의 대부분은 이렇게 CPU가 깨어나야만 하는 주기를 줄이는 것과 관련이 있습니다.

이를 위해서, Red Hat Enterprise Linux 6 Linux 커널은 주기적 타이머를 없앴습니다: 그에 따라서, CPU의 유휴 상태는 이제 *틱이 없는(tickless)* 상황이 되었습니다. 이에 따라 CPU는 유휴 상태일 때 불필요한 전력을 소비하지 않습니다. 하지만, 이러한 특징을 적용해 얻을 수 있는 이득은 시스템이 불필요한 타이머 이벤트를 발생시키는 프로그램을 필요시 감소시킬 수 있다는 것입니다. 이러한 이벤트의 예로는 폴링(polling) 이벤트(예를 들어 볼륨 변화나 마우스 움직임을 감지하는 등의 일)를 들 수 있습니다.

Red Hat Enterprise Linux 6에는 애플리케이션을 CPU 사용 방식에 따라 평가할 수 있는 도구가 포함되어 있습니다. 더 자세한 것은 2장. *전원 관리 감사와 분석*를 참조하십시오.

사용하지 않는 하드웨어와 장치는 완전히 비활성화되어야만 합니다

이것은 움직이는 부품이 있는 장치(예: 하드디스크)의 경우 특별히 더 중요합니다. 또한 일부 애플리케이션은 사용하지 않는 장치를 "열린" 상태로 남겨두고는 합니다; 이러한 일이 발생하면, 커널은 그 장치가 사용중이라 가정하게 되어, 해당 장치를 전원 절약 상태로 전환할 수 없게 됩니다.

낮은 활동은 낮은 전력 소모로 변환되어야만 합니다

하지만, 많은 경우 이는 하드웨어와 BIOS 설정에 달려있습니다. 오래된 시스템의 구성요소들은 Red Hat Enterprise Linux 6에서 현재 지원할 수 있는 새로운 기능 중 일부를 지원하지 못하곤 합니다. 시스템에서 가장 최신의 공식 펌웨어를 사용하고 있는지 확인하시고, BIOS의 전원관리와 장치 설정 부분을 검토해서 전원관리 기능이 활성화 되도록 하십시오. 살펴봐야 하는 특성에는 다음과 같은 것들이 있습니다:

- SpeedStep
- PowerNow!
- Cool'n'Quiet
- ACPI (C state)
- Smart

만약 하드웨어가 이러한 기능을 지원하고, BIOS에서 활성화되어 있다면, Red Hat Enterprise Linux 6는 그런 기능을 디폴트로 사용할 것입니다.

CPU 상태의 차이와 그에 따른 효과

향상된 설정과 전원 인터페이스(Advanced Configuration and Power Interface)(ACPI)를 제공하는 최근의 CPU들은 여러 다른 전원 상태를 제공합니다. 이러한 상태들에는:

- 슬립 (Sleep,C-states)
- 주파수 (Frequency, P-states)
- 열 출력(T-states 또는 "열 상태")

가장 낮은 슬립 상태에서 동작중인 CPU는 가장 적은 전력을 소비합니다. 하지만, 필요에 의해 해당 상태에서 깨어나야 하는 경우 상당한 시간을 필요로 합니다. 아주 드물게는 이로 인해 CPU가 슬립상태가 되자마자 깨어나야 하는 경우도 발생합니다. 이러한 상황은 계속해서 바쁜 CPU와 동일한 효과를 가져오며, 다른 상태를 활용할 경우 얻을 가능성이 있었던 잠재적인 전원 절약을 달성하지 못하는 원인이 됩니다.

꺼진 기계가 가장 전력을 적게 소비합니다

너무나 당연한 이야기이지만, 실제 전원을 절약하는 가장 좋은 방법은 시스템을 끄는 것입니다. 예를 들어 회사가 점심시간이나 퇴근시 컴퓨터를 끄도록 하여, "그린 IT" 정신에 입각한 사내 문화를 발전시킬 수도 있습니다. 또한 Red Hat Enterprise Linux 6에서 제공하는 가상화 기술을 활용해 몇몇 물리적인 서버를 한 대의 큰 서버로 통합하여 이를 가상 서버로 사용할 수 도 있을 것입니다.

2장. 전원 관리 감사와 분석

2.1. 감사와 분석 개요

어느 특정 시스템을 수동으로 감사하고, 분석하고, 튜닝하는 것은 그에 드는 시간과 비용이 해당 시스템을 튜닝해서 최대한 얻을 수 있는 이익을 초과하기 때문에 보통 드문일입니다. 하지만, 동일한 설정을 모두에 적용할 수 있는 동일한 시스템이 많이 있는 경우, 수동으로 이러한 작업을 수행하는 것이 매우 유용할 수 있습니다. 예를 들어 기계가 거의 동일한 수천대의 데스크탑이나, 고성능(HPC) 클러스터를 고려해 볼 수 있습니다. 감사와 분석을 진행하는 또 다른 이유는 미래에 시스템의 동작 방식의 변화나 기존 동작과의 차이를 식별하기 위한 비교의 기준을 만들기 위함일 수 있습니다. 이러한 분석의 결과는 하드웨어, BIOS 또는 소프트웨어 업데이트가 주기적으로 일어나고, 전력 소비와 관련해 급격한 변화를 피하고 싶을 때 도움이 될 수 있습니다. 일반적으로, 철저한 감사와 분석을 통해 실제 특정 시스템에서 어떤 일이 벌어지고 있는가에 대한 더 나은 조망을 가질 수 있습니다.

전력 소비와 관련한 시스템 감사와 분석은 상대적으로 어렵습니다. 이는 사용 가능한 가장 최신의 시스템의 경우에도 그렇습니다. 대부분의 시스템은 소프트웨어로 전력 소비를 측정할 수 있는 수단을 제공하지 않습니다. 하지만, 예외도 있습니다: Hewlett Packard의 ILO 관리 콘솔은 웹을 통해 액세스할 수 있는 전원 관리 모듈을 제공합니다. IBM도 그와 유사한 솔루션을 BladeCenter 전원 관리 모듈에서 제공합니다. 몇몇 Dell의 제품에서는 IT 지원이 전력 모니터링 기능을 제공하기도 합니다. 다른 벤더들도 유사한 기능을 서버 플랫폼에 제공하곤 합니다. 하지만, 이렇듯 모든 벤더에서 지원하는 단일 솔루션은 없습니다. 만약 시스템이 전력 소비를 측정할 수 있는 내부 메커니즘을 제공하지 않는다면, 몇 가지 다른 방법이 있습니다. USB를 통해 전력 소비를 측정하도록 지원하는 특별한 파워 서플라이를 설치할 수 있습니다. Gigabyte의 Odin GT 550 W PC 파워 서플라이가 그러한 제품의 예입니다. 또한 리눅스에서 그런 값을 읽도록 지원하는 소프트웨어는 외부링크 http://mgmt.sth.sze.hu/~andras/dev/gopsu/에서 찾을 수 있습니다. 마지막 방법으로, Watts up? PRO와 같은 몇몇 외부 전력 측정기에는 USB 단자가 있습니다.

전력 소비를 직접 측정하는 것은 때로 가능한 한 최대로 전력 소비를 줄이고 싶을때만 필요합니다. 운좋게도, 다른 방법을 통해 변경에 따른 효과나 시스템이 동작하는 방식을 측정할 수 있습니다. 이번 장은 이에 필요한 도구를 설명합니다.

2.2. POWERTOP

Red Hat Enterprise Linux 6에 틱없는 커널이 도입되면서(3.4절. “틱없는 커널”를 참조), CPU가 더 자주 유휴 상태에 들어갈 수 있게 되어, 전력 소비를 줄이고, 전원 관리를 향상 시켰습니다. 새로운 **PowerTOP** 도구는 CPU를 자주 깨우는 커널의 특정 구성요소와 사용자 공간 어플리케이션을 식별합니다. 개발시 **PowerTOP**은 3.11절. “사용자 공간에서의 최적화”에 설명한, CPU를 깨우는 정도를 10배정도 감소하도록 어플리케이션들을 튜닝한, 검사를 실행하기 위해 사용되었습니다.

PowerTOP을 다음 명령으로 설치하십시오:

```
yum install powertop
```

PowerTOP을 다음과 같이 실행하십시오:

```
powertop
```

유용한 결과를 얻으시려면 **PowerTOP**을 root 권한으로 실행해야 한다는 것을 명심하십시오.

실행되면 **PowerTOP**은 시스템에서 정보를 수집하여, CPU를 가장 자주 깨우는 구성요소의 목록을 제시합니다. **PowerTOP**은 또한 절전을 하도록 시스템을 튜닝하는 방법도 제안합니다. 이러한 제안은 화면의 아래쪽에 표시되며, **PowerTOP**의 제안을 받아들이고자할 경우 눌러야 할 키도 함께 표시합니다. **PowerTOP**이 주기적으로 갱신되면서 추가적인 제안사항도 표시될 것입니다. 그림 2.1. “동작중인 PowerTOP”에서 **increase the VM dirty writeback time**라고 되어있는 제안사항과 그 제안을 허용하기 위해서 눌러야 하는 키(**W**)를 확인해 보십시오.

실행이 되면 **PowerTOP**은 시스템에서 통계 정보를 모아서 몇 가지 중요한 정보를 보여줍니다. 맨 위에는 CPU 코어가 얼마나 오래 각각의 C와 P상태에 있었는지를 표시하 줍니다. CPU가 더 높은 C나 P상태에 더 오래 있는 것이 더 나은 것이며(**C4**가 **C3**보다 높음), 이것은 시스템이 CPU 활용도에 있어 얼마나 최적화 되어있는가를 보여주는 좋은 지표입니다. 목표를 시스템이 유휴상태에 있는 동안 CPU가 90% \ 또는 그 이상 최고의 C 또는 P 상태에 있는 것으로 잡아야 합니다.

두번째로 보여지는 정보는 해당 기계의 1초당 깨어난 횟수입니다. 초당 깨어난 횟수는 서비스나 커널의 디바이스 드라이버가 시스템의 전력 사용 측면에서 얼마나 잘 동작하고 있는가를 알 수 있는 척도입니다. 초당 깨어난 횟수가 더 많을 수록, 더 많은 전력을 소모하므로, 이 수치가 더 낮은것이 더 좋습니다.

다음으로 **PowerTOP**은 시스템의 실제 전력 소비량에 대한 추정치를 제공합니다. 가능하다면, 랩탑이 배터리 모드에 있는 동안 **PowerTOP**이 이 수치를 제시하도록 해 보십시오.

전력 소비 예측치 다음에는 CPU를 가장 자주 깨우는 구성요소의 상세 목록이 있습니다. 목록의 맨 위에 있는 구성요소들이 여전력 소모를 줄이기 위해 시스템을 최적화할 때 더 자세히 검토해야만 하는 것들입니다. 만약 그것이 커널 구성요소라면(**<>**사이에 목록이 표시됨), CPU를 깨우는 원인을 제공하는 특정 드라이버와 연관되어 있기도 합니다. 드라이버를 튜닝하기 위해서는 커널을 변경해야 하는 경우가 많고, 이는 이 문서의 범위를 벗어나는 것입니다. 하지만, 사용자 영역 프로세스가 CPU를 자주 깨우는 것은 더 쉽게 처리할 수 있습니다. 우선, 그 서비스나 프로그램이 이 시스템에서 실행될 필요가 있는지를 판단하십시오. 만약 필요가 없다면, 단순히 서비스를 비활성화 합니다. 서비스를 영구히 끄려면, 다음을 실행하십시오:

```
chkconfig 서비스이름 off
```

어떤 콤포넌트가 실제로 어떤 동작을 했는지를 자세히 보고 싶다면, 다음을 실행하십시오:

```
ps -awux | grep 콤포넌트이름
strace -p 프로세스id
```

이 추적 결과, 같은 것이 계속 반복되는 것 처럼 보인다면, 아마도 바쁜 고리(busy loop)에 빠진 것일 수 있습니다. 이러한 문제를 수정하기 위해서는, 해당 구성요소의 코드를 변경해야 하며, 이는 이 문서의 범위를 벗어나는 것입니다.

마지막으로, **PowerTOP**은 더 낮은 전력 소비를 위해 시스템을 어떻게 튜닝할지에 대한 제안을 표시합니다. 이러한 제안사항은 화면의 맨 아래 표시되며, 그 **PowerTOP**의 제안을 받아들이고 싶을 때 눌러야 하는 키도 함께 표시됩니다. **PowerTOP**가 주기적으로 갱신됨에 따라, 다른 제안이 더 표시될 수도 있습니다. 그림 2.1. “동작중인 PowerTOP”에서 **increase the VM dirty writeback time**라는 제안과 그 제안을 받아들이기 위해 사용할 수 있는 **W** 키에 주의하십시오. 이 변경은 다음번 재시작 이전까지만 유효합니다. 이런 변경사항을 시스템에 영구히 적용하기 위해, **PowerTOP**은 최적화를 위해 실행해야 하는 정확한 명령을 표시합니다. 이 명령을 **/etc/rc.local** 파일에 선호하는 텍스트 에디터로 저장해서 시스템 시작시 해당 변경 사항이 적용되도록 하십시오.

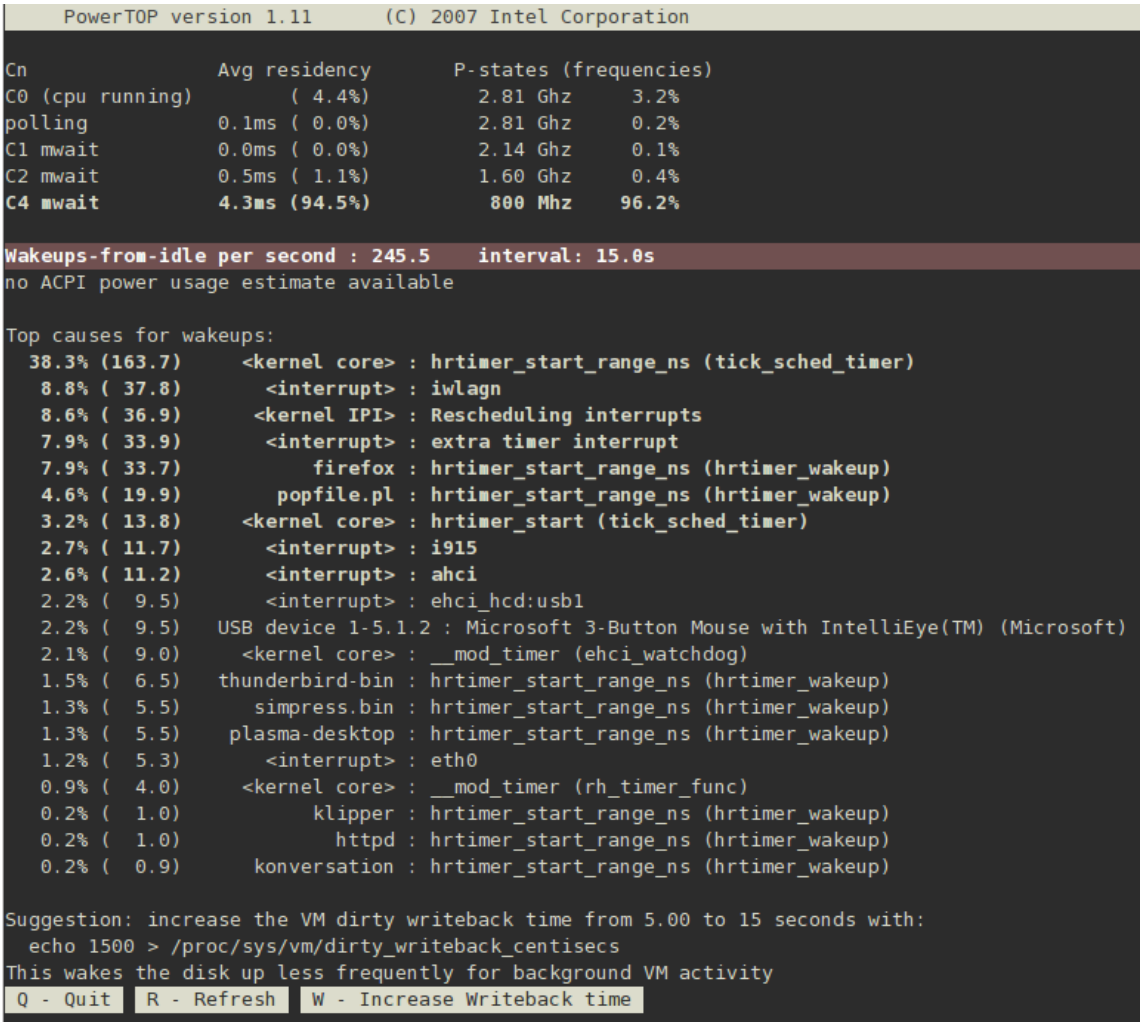


그림 2.1. 동작중인 PowerTOP

Less Watts 웹사이트는 **PowerTOP**이 CPU를 활성화시키는 것을 식별할 수 있는 어플리케이션의 목록을 알려줍니다. http://www.lesswatts.org/projects/powertop/known.php을 참조하십시오.

2.3. DISKDEVSTAT과 NETDEVSTAT

Diskdevstat과 **netdevstat**은 시스템에서 실행중인 모든 프로그램의 디스크와 네트워크 활동상황에 대한 자세한 정보를 수집하는 **SystemTap** 도구입니다. 이러한 도구는 모든 프로그램이 매 초당 CPU를 깨우는 횟수를 보여주는 **PowerTOP**에 의해 영감을 받아 만들여졌습니다(2.2절. “PowerTOP” 참조). 이러한 도구가 수집한 통계 자료를 통해 프로그램에서 큰 크기의 I/O연산을 적게 수행하는 대신에 작은 크기의 연산을 많이 수행해서 전력을 낭비하는 부분을 구별할 수 있습니다. 전송 속도만을 측정하는 다른 모니터링 도구들은 이러한 사용 유형을 구별하는 데는 별 도움이 되지 않습니다.

SystemTap 도구를 다음 명령으로 설치하십시오:

```
yum install systemtap tuned-utils kernel-debuginfo
```

다음 명령을 도구를 실행합니다:

```
diskdevstat
```

또는 다음과 같이 합니다:

```
netdevstat
```

두 명령 모두, 다음과 같이 세개의 매개변수를 받습니다:

diskdevstat *업데이트_주기 전체_시간 히스토그램_표시*

netdevstat *업데이트_주기 전체_시간 히스토그램_표시*

업데이트_주기

각 화면 업데이트 사이의 시간(단위:초). 기본값: **5**

전체_시간

모니터링을 할 전체 시간(단위:초). 기본값: **86400**(하루)

히스토그램_표시

실행이 끝날 때 수집한 전체 데이터를 히스토그램으로 보여줄지 여부

출력은 **PowerTOP**과 유사합니다. 여기에 KDE 4.2를 돌리는 Fedora 10 시스템에서 수행한 **diskdevstat**의 출력 예가 있습니다:

```
  PID   UID DEV     WRITE_CNT WRITE_MIN WRITE_MAX WRITE_AVG    READ_CNT
READ_MIN  READ_MAX  READ_AVG COMMAND
 2789  2903 sda1          854     0.000   120.000    39.836           0
0.000     0.000     0.000 plasma
15494     0 sda1            0     0.000     0.000     0.000         758
0.000     0.012     0.000 0logwatch
15520     0 sda1            0     0.000     0.000     0.000         140
0.000     0.009     0.000 perl
15549     0 sda1            0     0.000     0.000     0.000         140
0.000     0.009     0.000 perl
15585     0 sda1            0     0.000     0.000     0.000         108
0.001     0.002     0.000 perl
 2573     0 sda1           63     0.033  3600.015   515.226           0
0.000     0.000     0.000 auditd
15429     0 sda1            0     0.000     0.000     0.000          62
0.009     0.009     0.000 crond
15379     0 sda1            0     0.000     0.000     0.000          62
0.008     0.008     0.000 crond
15473     0 sda1            0     0.000     0.000     0.000          62
0.008     0.008     0.000 crond
15415     0 sda1            0     0.000     0.000     0.000          62
0.008     0.008     0.000 crond
15433     0 sda1            0     0.000     0.000     0.000          62
0.008     0.008     0.000 crond
15425     0 sda1            0     0.000     0.000     0.000          62
0.007     0.007     0.000 crond
15375     0 sda1            0     0.000     0.000     0.000          62
0.008     0.008     0.000 crond
15477     0 sda1            0     0.000     0.000     0.000          62
0.007     0.007     0.000 crond
15469     0 sda1            0     0.000     0.000     0.000          62
0.007     0.007     0.000 crond
15419     0 sda1            0     0.000     0.000     0.000          62
0.008     0.008     0.000 crond
15481     0 sda1            0     0.000     0.000     0.000          61
```

```
0.000     0.001     0.000 crond
15355     0 sda1            0     0.000     0.000     0.000           37
0.000     0.014     0.001 laptop_mode
 2153     0 sda1           26     0.003  3600.029  1290.730            0
0.000     0.000     0.000 rsyslogd
15575     0 sda1            0     0.000     0.000     0.000           16
0.000     0.000     0.000 cat
15581     0 sda1            0     0.000     0.000     0.000           12
0.001     0.002     0.000 perl
15582     0 sda1            0     0.000     0.000     0.000           12
0.001     0.002     0.000 perl
15579     0 sda1            0     0.000     0.000     0.000           12
0.000     0.001     0.000 perl
15580     0 sda1            0     0.000     0.000     0.000           12
0.001     0.001     0.000 perl
15354     0 sda1            0     0.000     0.000     0.000           12
0.000     0.170     0.014 sh
15584     0 sda1            0     0.000     0.000     0.000           12
0.001     0.002     0.000 perl
15548     0 sda1            0     0.000     0.000     0.000           12
0.001     0.014     0.001 perl
15577     0 sda1            0     0.000     0.000     0.000           12
0.001     0.003     0.000 perl
15519     0 sda1            0     0.000     0.000     0.000           12
0.001     0.005     0.000 perl
15578     0 sda1            0     0.000     0.000     0.000           12
0.001     0.001     0.000 perl
15583     0 sda1            0     0.000     0.000     0.000           12
0.001     0.001     0.000 perl
15547     0 sda1            0     0.000     0.000     0.000           11
0.000     0.002     0.000 perl
15576     0 sda1            0     0.000     0.000     0.000           11
0.001     0.001     0.000 perl
15518     0 sda1            0     0.000     0.000     0.000           11
0.000     0.001     0.000 perl
15354     0 sda1            0     0.000     0.000     0.000           10
0.053     0.053     0.005 lm_lid.sh
```

컬럼은 다음과 같습니다:

PID

프로그램의 프로세스 ID

UID

실행중인 프로그램의 사용자 ID

DEV

I/O가 일어난 장치

WRITE_CNT

쓰기 동작 횟수

WRITE_MIN

연속된 두 쓰기 동작 사이의 최소 간격(초)

WRITE_MAX

연속된 두 쓰기 동작 사이의 최대 간격(초)

WRITE_AVG

두 연속된 쓰기 동작 사이의 평균 간격(초)

READ_CNT

전체 읽기 동작 횟수

READ_MIN

두 연속된 읽기 동작 사이의 최소 간격(초)

READ_MAX

두 연속된 읽기 동작 사이의 최대 간격(초)

READ_AVG

두 연속된 읽기 동작 사이의 평균 시간(초)

COMMAND

프로세스의 이름

이 예에서 3가지 프로그램이 가장 눈에 띕니다:

```
  PID   UID DEV     WRITE_CNT WRITE_MIN WRITE_MAX WRITE_AVG    READ_CNT
READ_MIN  READ_MAX  READ_AVG COMMAND
 2789  2903 sda1          854     0.000   120.000    39.836           0
0.000     0.000     0.000 plasma
 2573     0 sda1           63     0.033  3600.015   515.226           0
0.000     0.000     0.000 auditd
 2153     0 sda1           26     0.003  3600.029  1290.730           0
0.000     0.000     0.000 rsyslogd
```

이 세개의 프로그램은 **WRITE_CNT**가 **0**보다 큽니다. 이는 이들이 측정하는 동안 어떤 데이터를 기록했다는 것을 의미합니다. 이들 중에서 **plasma**는 상당히 큰 차이로 가장 나쁜 결과를 보여줍니다: 그것은 가장 많은 쓰기 동작을 수행했고, 또한 쓰기 동작 사이의 평균 시간도 가장 짧습니다. **Plasma**는 따라서 여러분이 에너지를 낭비하는 프로그램에 대해 우려하고 있다면 검사해야할 가장 좋은 대상이 됩니다.

strace과 **ltrace**를 사용해 주어진 프로세스 ID의 모든 시스템 호출을 검사함으로써 더 자세히 프로그램을 살펴보십시오. 본 예에서, 다음과 같이 할 수 있습니다:

```
strace -p 2789
```

이 예의 **strace**의 출력에는 45초마다 반복되는 패턴이 있습니다. 그것은 사용자의 KDE 아이콘 캐시 파일을 쓰기위해 연 직후 그 파일을 닫는 것입니다. 이것은 파일 메타정보 부분(특히 마지막 변경 시간 정보)이 바뀌었기 때문에 하드 디스크에 물리적인 쓰기를 야기합니다. 최종 수정본에서는 아이콘이 실제 업데이트 되지 않은 경우 이러한 불필요한 시스템 콜이 일어나지 않도록 변경되었습니다.

2.4. 배터리 수명 도구 키트

Red Hat Enterprise Linux 6에는 **배터리 수명 도구 키트(Battery Life Tool Kit)**(BLTK)가 새로 추가되었습니다. 이는 배터리 수명과 성능을 시뮬레이션하고 분석하는 테스트 도구입니다. BLTK는 개발자나 사무 노동자와 같은 특정 사용자 그룹을 시뮬레이션하는 작업의 모음을 수행하고, 그 결과를 보고함으로써 이런 목적을 달성합니다. 노트북의 성능을 테스트하기 위한 목적으로 개발된 것이기는 하지만, BLTK를 **-a** 옵션으로 시작해서 데스크탑 컴퓨터의 성능에 대해서도 보고할 수 있습니다.

BLTK를 사용해 실제 기계를 사용하는 경우와 비교할 수 있는 부하를 여러번 되풀이해서 만들어낼 수 있습니다. 예를 들어 **office** 부하는 텍스트를 쓰고, 수정하는 작업을 하고, 동일한 일을 스프레드시트에서도 합니다. BLTK와 **PowerTOP**등의 분석/감사 도구를 함께 수행함으로써, 수행한 최적화가 유휴시간 뿐 아니라 컴퓨터를 활동적으로 사용하는 경우에도 효과가 있는지를 테스트할수 있습니다. 서로 다른 설정에서도 동일한 부하량을 발생시킬 수 있기 때문에, 서로 다른 설정하에서의 결과를 비교해 볼 수 있습니다.

BLTK를 다음 명령으로 설치하십시오:

```
yum install bltk
```

BLTK를 다음 명령으로 실행합니다:

```
bltk workload 옵션들
```

예를 들어 **idle** 작업부하를 120초간 실행하려면 다음과 같이 합니다:

```
bltk -I -T 120
```

기본으로 사용 가능한 작업부하들은 다음과 같습니다:

-I, --idle

시스템이 유휴상태입니다. 다른 작업부하와 비교를 위한 기준으로 사용됩니다

-R, --reader

문서를 읽는 작업을 시뮬레이션합니다(디폴트로, **Firefox**를 사용합니다)

-P, --player

CD나 DVD 드라이브에서 멀티미디어 파일을 보는 것을 시뮬레이션합니다(디폴트로, **mplayer**를 사용합니다)

-O, --office

OpenOffice.org 스위트를 사용해 문서를 편집하는 것을 시뮬레이션합니다

지정할 수 있는 다른 옵션은 다음과 같습니다:

-a, --ac-ignore

교류 전원이 있는지 여부를 무시합니다(데스크탑에서 사용시 필요)

-T *시간(초)*, --time *시간(초)*

테스트를 얼마동안 실행할지를 초단위로 정합니다; 이 옵션을 **idle**작업부하와 함께 사용하십시오

-F *파일이름*, --file *파일이름*

특정 부하를 위해 사용할 파일을 지정합니다. 예를 들자면 **player** 부하가 CD나 DVD 드라이브를 읽는 대신 사용할 파일이 되겠습니다.

-W *어플리케이션*, **--prog** *어플리케이션*

특정 작업부하에서 사용할 어플리케이션을 지정합니다. 예를 들자면 **reader** 작업부하에서 **Firefox** 대신 사용할 브라우저가 되겠습니다.

BLTK는 더 자세한 옵션을 많이 제공합니다. 자세한 것은 **bltk** 매뉴얼(man) 페이지를 참조하십시오.

BLTK는 **/etc/bltk.conf** 설정 파일에 지정된 디렉터리에 보고서를 저장합니다. 디폴트는 **~/.bltk/***작업부하*.**results.***숫자*/ 입니다. 예를 들어 **~/.bltk/reader.results.002/** 디렉터리는 **reader** 작업부하를 가지고 수행한 3번째 테스트의 결과가 저장되어 있습니다(첫번째 테스트에는 번호가 붙지 않습니다). 결과는 몇 개의 텍스트 파일에 나뉘어 있습니다. 이러한 결과를 읽기 쉬운 형식으로 축약하려면, 다음을 실행하십시오:

```
bltk_report 결과_디렉터리_경로
```

이제 결과가 결과 디렉터리의 **Report**라는 텍스트파일에 저장될 것입니다. 이 결과를 터미널에서 보려면, **-o** 옵션을 사용하십시오:

```
bltk_report -o 결과_디렉터리_경로
```

2.5. TUNED와 KTUNE

Tuned는 시스템 구성요소를 모니터링하고, 그 결과에 따라서 시스템 설정을 동적으로 튜닝해주는 데몬입니다. 동작인 튜닝에 의해 시스템이 켜져 있는 동안 여러 시스템 구성요소가 사용되는 방식이 달라집니다. 예를 들어, 하드 드라이브는 시스템 시작과 로그인시 매우 많이 사용되지만, 나중에 사용자가 OpenOffice나 이메일 클라이언트와 같은 프로그램으로 작업을 주로 할 때는 거의 사용되지 않습니다. 마찬가지로, CPU와 네트워크 장치도 다른 시간에 다른 방식으로 사용됩니다. **Tuned**는 이러한 구성 요소의 활동을 모니터해서 사용상의 변화에 대응합니다.

실질적인 예로, 전형적인 사무 작업용 컴퓨터를 생각해 봅시다. 대부분의 시간에 이더넷 네트워크 인터페이스는 동작하지 않을 것입니다. 단지 몇개의 이메일만 오고 가고, 몇몇 웹 페이지만 외부에서 읽어올 것입니다. 이런 종류의 작업에 대해서는 디폴트로 설정된 것과 같이 네트워크 인터페이스가 항상 최고 속도로 동작할 필요가 없습니다. **Tuned**는 모니터링을 한 후, 네트워크 장치가 활동이 적음을 파악하고, 자동으로 그 인터페이스의 속도를 낮추며, 보통 이렇게 하면 전력 소모가 줄어듭니다. 만약 해당 인터페이스의 활동이 상당한 기간동안 증가한다면-예를 들어 DVD 이미지를 다운로드 하거나, 첨부 파일이 커다란 이메일을 여는 경우가 그럴 것입니다-**tuned**는 이를 감지해서 인터페이스의 속도를 최고로 변경해 활동 수준이 높은 동안에 최적의 성능을 내도록 해줍니다. 이런 원칙은 CPU나 하드디스크와 같은 다른 플러그인에 대해서도 적용됩니다.

디폴트로 네트워크 장치는 이런 식으로 동작하게 되어 있지 않습니다. 왜냐하면 속도가 변경되는 데 수 초가 걸리고, 사용자 경험에 직접적이고 눈에 보이는 영향을 끼치기 때문입니다. CPU와 하드 드라이브에도 비슷한 고려 사항이 있습니다. 하드 드라이브의 회전 속도를 감소시키면, 다시 회전속도를 높이는 데 몇 초가 걸리게 되고, 이는 해당 주기동안 눈에 띄는 시스템의 반응성 감소를 가져옵니다. CPU 플러그인의 경우 이런 응답 지연이 가장 작긴 합니다만, 비록 사용자가 눈치챌 가능성은 낮더라도, 여전히 이러한 응답 지연을 관찰 가능합니다.

tuned과 함께 우리는 **ktune**를 제안합니다. **Ktune**은 Red Hat Enterprise Linux 5.3에서 특정 사용 유형에 대한 머신의 성능을 최적화 하기 위한 프레임워크와 서비스로 최초로 사용되었습니다. 그 이후 **ktune**도 많은 향상이 이루어졌으며, 이제 우리는 일반적인 튜닝 프레임워크에 꼭 들어가는 한 부분으로 그것을 활용하고 있습니다. 그것은 주로 2.5.2절. “Tuned-adm”에 설명한 여러 미리 정의된 프로파일에 따라 사용됩니다.

tuned 패키지와 그와 관련된 **systemtap** 스크립트를 다음 명령으로 설치하십시오:

```
yum install tuned
```

tuned 패키지를 설치시 **/etc/tuned.conf**에는 몇 가지 예제 설정 파일이 저장되며, 디폴트 프로파일이 활성화됩니다.

tuned를 다음과 같이 시작하십시오:

```
service tuned start
```

tuned를 시스템 부팅시마다 시작하려면, 다음과 같이 합니다:

```
chkconfig tuned on
```

Tuned에는 수동으로 시작할 때 지정할 수 있는 추가 옵션도 있습니다. 가능한 옵션은 다음과 같습니다:

-d, --daemon

tuned를 데몬으로 실행합니다.

-c, --conffile

예를 들자면, **--conffile=/etc/tuned2.conf**과 같이, 지정한 이름과 경로에 있는 설정 파일을 사용합니다. 디폴트는 **/etc/tuned.conf** 입니다.

-D, --debug

가장 높은 로깅(logging) 수준을 사용합니다.

2.5.1. tuned.conf 파일

tuned.conf 파일은 **tuned**의 설정을 저장하고 있습니다. 디폴트로 이 파일은 **/etc/tuned.conf**입니다. 다른 위치나 이름으로 지정하려면, **tuned.conf**를 **--conffile** 옵션을 사용해 실행하면 됩니다.

설정 파일에는 **tuned**의 일반적인 매개변수를 설정하는 **[main]**라는 섹션이 들어가 있어야만 합니다. 이 파일은 또한 각각의 플러그인에 대한 섹션을 포함할 수 있습니다.

[main] 섹션은 다음과 같은 옵션을 포함합니다:

interval

tuned가 시스템을 모니터하고 튜닝해야 하는 주기를 초단위로 지정합니다. 기본값은 **10**입니다.

verbose

자세한 메시지를 출력할지를 정합니다. 기본값은 **False**입니다.

logging

로깅할 메시지의 최소 우선순위를 지정합니다. 지정 가능한 값은 우선순위가 높은 것부터 다음과 같습니다: **critical**, **error**, **warning**, **info**,**debug**. 기본값은 **info**입니다.

logging_disable

로깅할 메시지의 최대 우선순위를 지정합니다; 여기 지정된 우선순위값 이하의 메시지는 로깅하지 않습니다. 지정 가능한 값은 우선순위가 높은 것부터 다음과 같습니다: **critical**, **error**, **warning**, **info**,**debug**. 기본값은 **info**입니다.

각각의 플러그인에 해당하는 자체 섹션이 있습니다. 각 섹션은 각괄호로 닫힌 플러그인 이름으로 지정됩니다; 일례로 **[CPUTuning]**. 각각의 플러그인에 자신만의 옵션을 지정할 수 있지만, 다음 옵션들은 모든 플러그인에 사용 가능합니다:

enabled

이 플러그인을 사용할지 여부를 지정합니다. 디폴트 값은 **True**입니다.

verbose

자세한 출력을 표시할지 여부를 지정합니다. 만약 특정 플로그인에 대해 설정하지 않는다면, **[main]**에서 지정된 값에 따르게 됩니다.

logging

로깅할 메시지의 최소 우선순위를 지정합니다. 특정 플로그인에 대해 지정하지 않았다면, **[main]**에서 지정한 값을 따릅니다.

다음은 예제 설정 파일입니다:

```
[main]
interval=10
pidfile=/var/run/tuned.pid
logging=info
logging_disable=notset

# Disk monitoring section

[DiskMonitor]
enabled=True
logging=debug

# Disk tuning section

[DiskTuning]
enabled=True
hdparm=False
alpm=False
logging=debug

# Net monitoring section

[NetMonitor]
enabled=True
logging=debug

# Net tuning section

[NetTuning]
enabled=True
logging=debug
```

```
# CPU monitoring section

[CPUMonitor]
# Enabled or disable the plugin. Default is True. Any other value
# disables it.
enabled=True

# CPU tuning section

[CPUTuning]
# Enabled or disable the plugin. Default is True. Any other value
# disables it.
enabled=True
```

2.5.2. Tuned-adm

때때로, 시스템을 자세히 검사하고 분석하는 것은 매우 많은 시간을 필요로 하고, 겨우 몇 와트를 절약하기 위해 이런 일을 하는게 그리 가치있어 보이지 않기도 합니다. 예전에는 단순히 디폴트를 사용하는 것이 유일한 대안이었습니다. 따라서, Red Hat Enterprise Linux 6는 이러한 양쪽 극단(다 분석하거나, 단순히 디폴트를 사용하거나) 대신 특별한 사용 양식에 따른 별도의 프로파일을 제공합니다. 이와 함께 **tuned-adm** 도구를 사용하면, 명령행에서 쉽게 프로파일을 변경할 수 있습니다. Red Hat Enterprise Linux 6는 전형적인 사용 양식에 따라 미리 정의된 몇 가지 프로파일을 제공하며, **tuned-adm** 명령을 이용해 단순히 이들을 선택하고 활성화할 수 있습니다. 물론, 사용자 자신의 프로파일을 만들고, 변경하고, 삭제하는 것도 가능합니다.

모든 사용 가능한 프로파일의 목록을 얻고, 현재 활성화된 프로파일을 보려면, 다음과 같이 합니다:

```
tuned-adm list
```

현재 활성화된 프로파일만 표시하고 싶다면, 다음과 같이 합니다:

```
tuned-adm active
```

사용 가능한 프로파일 중 하나로 변경하려면, 다음을 실행합니다:

```
tuned-adm profile profile_name
```

예를 들면 다음과 같습니다:

```
tuned-adm profile server-powersave
```

모든 튜닝을 금지하려면 다음처럼 합니다:

```
tuned-adm off
```

tuned를 처음 설치할 때, **default** 프로파일이 활성화될 것입니다. Red Hat Enterprise Linux 6는 또한 다음과 같은 미리 정의된 프로파일을 제공합니다.

default

디폴트 전원절약 프로파일. 이는 사용 가능한 프로파일 중 전원 절약에 가장 적은 영향을 끼치는 것입니다. 오직 **tuned**의 CPU와 디스크 플러그인만을 사용합니다.

desktop-powersave

데스크탑 시스템을 위한 전원 절약 프로파일입니다. SATA 호스트 아답터에 대해서 ALPM 전원 절약을 사용하며(3.6절. "적극적 연결 전원 관리(Aggressive Link Power Management)"를 참조), **tuned**의 CPU, 이더넷, 디스크 플러그인도 함께 사용합니다.

server-powersave

서버 시스템을 위한 전원 절약 프로파일입니다. SATA 호스트 아답터에 대해서 ALPM 전원 절약을 사용하며, **HAL**(hal-disable-polling 매뉴얼 페이지 참조)을 통한 CD-ROM 폴링을 금지하고, **tuned**의 CPU와 디스크 플러그인을 사용합니다.

laptop-ac-powersave

AC전원에서 동작하는 랩탑을 위해 중간정도의 전원 절약을 제공하는 프로파일입니다. SATA 호스트 아답터를 위한 ALPM 전원 절약을 사용하고, WiFi 전원 절약과, **tuned**의 CPU, 이더넷, 디스크 플러그인을 사용합니다.

laptop-battery-powersave

배터리로 작동하는 랩탑을 위해 많은 전원 절약을 제공하는 프로파일입니다. 이전에 설명한 프로파일에서 제공하는 모든 전원 절약 방법을 사용하는 것은 물론이고, CPU를 덜 깨우는 시스템을 위한 멀티코어 전원 절약 스케줄러를 사용하며, 요구불 조정기를 사용도록 설정하고, AC97 오디오 전원 절약도 활성화합니다. 이 프로파일을 사용하면 배터리로 동작하는 랩탑 뿐만 아니고 다른 모든 종류의 시스템에서도 최대로 전원 절약을 하게 됩니다. 이 프로파일을 사용하는 경우의 단점은 성능에 눈에 띄는 영향을 끼친다는 것입니다. 특히 디스크와 네트워크 I/O의 응답 지연 면에서 그렇습니다.

throughput-performance

서버를 위한 전형적인 처리속도 성능에 비중을 둔 튜닝 프로파일입니다. **tuned**와 **ktune** 전원 절약 방법을 사용하지 않도록 하며, **sysctl** 설정을 디스크와 네트워크 I/O의 처리속도(throughput) 성능을 향상시키도록 활성화합니다. 또한 **deadline scheduler**로 스케줄러를 변경합니다.

latency-performance

서버를 위한 응답 시간에 비중을 둔 전형적인 튜닝 프로파일. **tuned**와 **ktune** 전원 절약 메커니즘을 상용 금지하고, 네트워크 I/O의 응답 지연을 향상시키기 위해 **sysctl** 설정을 활성화합니다.

모든 프로파일은 **/etc/tune-profiles** 서브디렉터리에 저장됩니다. 따라서 **/etc/tune-profiles/desktop-powersave**는 해당 프로파일에 필요한 모든 파일을 포함하게 됩니다. 각각의 프로파일 디렉터리는 최대 4개의 파일을 포함합니다:

tuned.conf

이 프로파일을 위해 활성화될 tuned의 설정 파일

sysctl.ktune

ktune가 사용할 **sysctl** 설정 파일. 파일의 포맷은 **/etc/sysconfig/sysctl**과 동일합니다(sysctl와 sysctl.conf 매뉴얼 페이지 참조).

ktune.sysconfig

ktune 자체에 대한 설정 파일. 보통 **/etc/sysconfig/ktune**입니다.

ktune.sh

ktune에 의해 사용되는 **init**-스타일의 셸 스크립트로 시스템 부팅시 시스템을 튜닝하기 위해 사용되는 구체적인 명령들을 담고 있습니다.

새로운 프로파일을 시작하는 가장 좋은 방법은 기존의 것을 복사하는 것입니다. **laptop-battery-powersave** 프로파일에는 매우 풍부한 튜닝 예제가 들어가 있으므로, 유용한 시작점이 될 수 있습니다. 전체 디렉터리를 다음과 같이 통째로 복사하십시오:

```
cp -a /etc/tune-profiles/laptop-battery-powersave/ /etc/tune-
profiles/myprofile
```

새 프로파일에 있는 파일을 필요에 맞게 변경하십시오. 예를 들어 CD가 바뀌었는지 감지하는 기능이 필요하다면, ktune.sh에서 그에 해당하는 줄을 찾아서 커맨트를 제거하면 됩니다:

```
# Disable HAL polling of CDROMS
# for i in /dev/scd*; do hal-disable-polling --device $i; done > /dev/null
2>&1
```

2.6. DEVICEKIT-POWER 및 DEVKIT-POWER

Red Hat Enterprise Linux 6에서 **DeviceKit-power**는 이전 Red Hat Enterprise Linux 버전에서 **GNOME Power Manager**의 일부 기능과 **HAL**의 일부 기능이었던 전원 관리 기능을 담당합니다. (2.7절. “GNOME Power Manager” 참조) **DeviceKit-power**는 데몬, API, 명령행 도구 모음을 제공합니다. 시스템의 각 전원은 물리적 장치 여부에 관계 없이 장치로 표시됩니다. 예를 들어, 랩톱 배터리와 AC 전원은 모두 장치로 표시됩니다.

devkit-power 명령과 다음 옵션을 사용하여 명령행 도구에 액세스할 수 있습니다:

--enumerate, -e

시스템의 각 전원 장치에 대한 객체 경로를 표시합니다. 예:

```
/org/freedesktop/DeviceKit/power/devices/line_power_AC
/org/freedesktop/UPower/DeviceKit/power/battery_BAT0
```

--dump, -d

시스템의 모든 전원 장치에 대한 매개 변수를 표시합니다.

--wakeups, -w

시스템의 CPU wakeups를 표시합니다.

--monitor, -m

AC 전원 연결 및 연결 해제 또는 배터리 부족과 같은 전원 장치의 변화에 대해 시스템을 모니터링합니다. 시스템 모니터링을 중지하려면 **Ctrl+C**를 누릅니다.

--monitor-detail

AC 전원 연결 및 연결 해제 또는 배터리 부족과 같은 전원 장치의 변화에 대해 시스템을 모니터링합니다. **--monitor-detail** 옵션은 **--monitor** 옵션 보다 더 자세한 정보를 제공합니다. 시스템 모니터링을 중지하려면 **Ctrl+C**를 누릅니다.

--show-info *object_path*, -i *object_path*

특정 객체 경로에 대해 사용 가능한 모든 정보를 표시합니다. 예를 들어, 객체 경로 **/org/freedesktop/UPower/DeviceKit/power/battery_BAT0**로 표시되는 시스템 배터리에 대한 정보를 검색하려면 다음을 수행합니다:

-

```
devkit-power -i /org/freedesktop/UPower/DeviceKit/power/battery_BAT0
```

2.7. GNOME POWER MANAGER

GNOME Power Manager는 GNOME 데스크탑의 일부로 설치되는 데몬입니다. 이전 Red Hat Enterprise Linux 버전에서 제공되었던 **GNOME Power Manager**의 전원 관리 기능의 대부분은 Red Hat Enterprise Linux 6에서 **DeviceKit-power**의 구성 요소가 되었습니다 (2.6절. “DeviceKit-power 및 devkit-power ” 참조). 하지만 **GNOME Power Manager**는 그 기능에 대해 프런트 엔드로 남아 있습니다. 시스템 트레이의 애플릿을 통해 **GNOME Power Manager**는 배터리로부터 AC 전원으로 전환 등과 같은 시스템 전원 상태 변경을 알립니다. 또한 배터리의 상태를 보고하는 동시에 배터리가 부족할 경우 경고합니다.

GNOME Power Manager는 기본적인 전원 관리 설정을 구성할 수 있습니다. 이러한 설정에 액세스하려면 시스템 트레이에서 **GNOME Power Manager** 아이콘을 클릭한 후 **기본 설정 (Preferences)**을 클릭합니다.

전원 관리 기본 설정 (Power Management Preferences) 화면에는 다음과 같은 세 개의 탭이 있습니다:

- **AC 전원으로 동작**
- **배터리로 동작**
- **일반**

AC 전원으로 동작 탭 및 **배터리로 동작**탭을 사용하여 동작 종료시 화면이 꺼질때 까지 경과 시간, 동작 종료 시 절전 모드로 될 때 까지 경과 시간, 동작 종료 시 시스템이 하드 디스켓을 스핀 다운할지의 여부를 지정합니다. **배터리로 동작** 탭에서 화면 밝기 설정 및 배터리 저하시 시스템 동작을 선택할 수 있습니다. 예를 들어 디폴트로 **GNOME Power Manager**는 배터리 수준이 매우 낮은 수준에 도달하면 시스템을 절전시킵니다. **일반** 탭에서 시스템의 (물리적) 전원 버튼과 정지 버튼에 대한 동작을 설정하고 어떤 **GNOME Power Manager** 아이콘을 시스템 트레이에 나타나게 할 지를 지정할 수 있습니다.

2.8. 감사를 위한 다른 방법

Red Hat Enterprise Linux 6는 시스템 감사와 분석을 진행하기 위한 몇 가지 다른 도구를 제공합니다. 그들 대부분은 이미 발견한 정보를 검증하고 싶을 때 추가적인 정보원으로 사용하거나, 특정 부분에 대한 더 상세한 정보를 필요로 할 때 사용할 수 있습니다. 이러한 도구 중 많은 수는 또한 성능 튜닝을 위해서도 사용됩니다. 이런 도구에는 다음과 같은 것이 있습니다:

vmstat

vmstat은 프로세스, 메모리, 페이징, 블럭 I/O, 트랩, 그리고 CPU 활동에 대한 상세 정보를 제공합니다. 시스템이 전체적으로 어떤 일을 하고, 어느 부분에서 바쁜지 자세히 살펴보기 위해 이를 사용하십시오.

iostat

iostat은 **vmstat**와 비슷하지만, 블럭 장치에 대한 I/O 정보만 제공합니다. 이는 또한 더 자세한 출력과 통계를 제공합니다.

blktrace

blktrace은 아주 자세한 블럭 I/O 추적 프로그램입니다. 이는 프로그램과 관련된 매 블럭에 대한 정보에 이르기까지 자세한 정보를 제공합니다. **diskdevstat**와 함께 사용하면 매우 유용합니다.

3장. 핵심 인프라와 기법

3.1. CPU 유휴 상태

x86 아키텍처의 CPU는 CPU 내의 어떤 부분이 활성화되거나 낮은 성능 상태로 실행될지를 정하는 여러 상태를 지원합니다. 이러한 상태는 *C-상태*로 알려져 있으며, 시스템이 CPU에서 사용중이 아닌 곳을 부분적으로 비활성화해서 전원을 절약하도록 해줍니다. C-상태는 C0부터 시작되며, 더 높은 번호가 붙을수록 CPU의 기능은 더 줄어들고, 전원 절약은 더 커집니다. 같은 번호의 C-상태는 여러 CPU간에 유사하지만, 각 C-상태의 구체적 특징은 특정 CPU나 CPU 패밀리에 따라 다릅니다. C-상태 0–3은 다음과 같이 정의됩니다:

C0

동작중 또는 실행중인 상태. 이 상태에서 CPU는 동작중이며 전혀 쉬는 일이 없습니다.

C1, 중단

프로세서가 명령어를 실행하지는 않고 있지만, 전형적인 저전력 상태에 있지는 않은 상태. CPU는 실제적으로 거의 지연 없이 명령어 처리를 계속 할 수 있습니다. C-상태를 지원하는 모든 프로세서들은 이 상태를 제공해야만 합니다. Pentium 4 프로세서들은 C1E라 불리는 실제로는 저전력소모 상태인 확장된 상태를 제공합니다.

C2, 클럭-중단

프로세서의 클럭이 중단되지만, 레지스터와 캐시의 생태는 온전히 유지됩니다. 따라서 클럭만 재개하면 즉시 처리를 시작할 수 있습니다. 이 상태는 옵션입니다.

C3, 슬립

프로세서가 슬립 상태로 들어가서 캐시를 최신 상태로 유지하지 못하는 상태입니다. 이로 인해 이 상태에서 깨어나는 것은 C2 상태에서 깨어나는 것보다 훨씬 시간이 걸립니다. 이 상태도 옵션입니다.

"네할렘(Nehalem)" 마이크로아키텍처의 최신 Intel CPU는 새로운C-State, C6를 포함합니다. 이 상태는 CPU의 전압을 0 볼트로 감소시키며, 전력 소모를 80%에서 90% 사이로 감소시킵니다. Red Hat Enterprise Linux 6의 커널은 이러한 새로운 C-State를 지원합니다.

3.2. CPUFREQ 조정기 사용하기

시스템의 전력 소비와 발열을 줄일 수 있는 가장 효과적인 방법 중 하나는 CPUfreq를 사용하는 것입니다. CPUfreq는 또한 CPU 속도 조절 — 프로세서의 클럭 속도를 실행중에 변경하는 것 — 로도 알려져 있습니다. 이는 시스템이 전원을 절약하기 위해 감소된 클럭 속도로 실행될 수 있도록 해줍니다. 주파수를 변경하는 규칙은, 즉 클럭 속도를 더 빠르게 할지 느리게 할지나 언제 주파수를 변경할지는, CPUfreq 조정기(governor)에 의해 정해집니다.

조정기는 시스템 CPU의 전원 특성을 정의하며, 이 특성은 다시 CPU의 성능에 영향을 미칩니다. 각각의 조정기마다 자신만의 독특한 동작 방식, 목적, 부하에 대한 적합성이 있습니다. 이 섹션은 어떻게 CPUfreq 조정기를 선택하고 설정하는지, 각각의 조정기의 기본적인 성격, 그리고 어떤 종류의 부하가 각각의 조정기에 적합한지에 대해 설명합니다.

3.2.1. CPUfreq 조정기 유형

이 절은 Red Hat Enterprise Linux 6에서 사용 가능한 CPUfreq 조정기의 서로 다른 유형을 나열하고, 설명합니다.

cpufreq_performance

성능 조정기(Performance governor)는 CPU가 가능한 한 최고의 클럭 주파수를 사용하도록 합니다. 이 주파수는 정적으로 설정될 것이며, 변하지 않을 것입니다. 따라서, 이 조정기는 *전원 절약을 제공하지 않습니다*. 이러한 조정기는 오직 부하가 많은 시간에 적합하며, 그런 경우 중에도 CPU가 거의(혹은 전혀) 유휴 상태로 가지 않는 경우에 적합합니다.

cpufreq_powersave

반대로, 전원절약 조정기는 CPU가 가능한 가장 낮은 클럭 주파수를 사용하도록 합니다. 이 주파수는 정적으로 설정될 것이며, 변하지 않을 것입니다. 이 조정기는 전력 소모를 최고로 줄지만, 그 댓가로 *CPU 성능은 가장 낮아집니다*.

하지만, "전원절약"이라는 말은 경우에 따라서는 참이 아닐 수 있습니다. 왜냐하면 (원리로 볼 때) 최고 부하로 도는 낮은 클럭의 CPU가 부하가 없는 높은 클럭의 CPU보다 전력 소모가 많기 때문입니다. 따라서 활동이 적을 것으로 예상되는 시간대에 CPU를 전원절약 조정기를 사용하도록 설정하는 것을 권장할만 하지만, 그 시간대에 예기치 못하게 높은 부하가 걸리는 경우 실제로는 전력을 더 소비할 수 있음을 알아두셔야 합니다.

전원절약 조정기는, 간단히 말해서, CPU의 "전력 절약장치"라기 보다 "속도 제한장치"입니다. 이 조정기는 과열이 문제가 될 수 있는 환경과 시스템에서 가장 유용합니다.

cpufreq_ondemand

요구불(Ondemand) 조정기는 CPU가 시스템의 부하가 높을 때는 최고 클럭 주파수로 동작하고, 시스템이 유휴상태일 때는 CPU가 최저 주파수로 돌도록 하는 조정기입니다. 이렇게 하면 전력 소비를 시스템의 부하에 따라 적절히 조정할 수 있지만, 그 댓가로 *주파수 변경에 따른 지연시간이 발생합니다*. 만약 시스템이 유휴상태와 고부하 상태를 자주 오가는 경우라면, 이러한 변경 지연시간이 요구불 조정기가 제공하는 성능/전력 절약에 따르는 이익을 상쇄할 수 있습니다.

대부분의 시스템에서 요구불 조정기는 발열, 전력 소비, 성능, 그리고 관리 측면에서 가장 좋은 절충점이 될 수 있습니다. 시스템이 하루중 일정 시간에만 바쁜 경우, 요구불 조정기는 더이상의 외부 간섭이 없이도 부하에 따라서 자동으로 최대와 최소 주파수 사이를 변경해 줍니다.

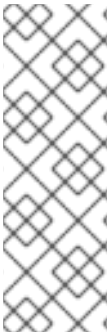
cpufreq_userspace

사용자 공간(Userspace) 조정기는 사용자 공간의 프로그램(또는 root로 실행중인 프로세스)이 주파수를 지정하도록 합니다. 이 조정기는 일반적으로 **cpuspeed** 데몬과 함께 사용합니다. 모든 조정기 중에서, 사용자 공간 조정기가 가장 사용자 설정이 자유로운 것입니다; 또한, 어떻게 설정되느냐에 따라서 시스템에 있어 성능과 전력 소모 간의 균형을 가장 잘 맞춰줄 수 있습니다.

cpufreq_conservative

요구불 조정기와 마찬가지로, 보수적 조정기 또한 클럭 주파수를 사용량(요구불 조정기의 경우와 같음)에 따라 조정합니다. 하지만, 요구불 조정기가 더 적극적인 방식으로 주파수를 조정하는 반면(즉, 최대에서 최소, 최소에서 최대로 변경함), 보수적 조정기는 주파수를 좀 더 점진적으로 변경합니다.

이는 보수적 조정기가, 최고 주파수나 최저 주파수 중 하나를 단순히 고르기 보다, 부하에 들어맞는 주파수로 클럭을 변경할 것임을 의미합니다. 이렇게 하면 더 많이 전력 소비를 절약할 수도 있겠지만, 요구불 조정기보다 *주파수 변경 지연이 더 커지게 됩니다*.



참고

cron 명령을 사용해 조정기를 활성화할 수 있습니다. 그것은 특정 조정기를 하루 중 특정 시간대에 지정할 수 있게 해 줍니다. 유휴 시간대(예: 일과시간 후)에 낮은 주파수의 조정기를 지정하고, 업무 부하가 큰 시간대에는 주파수가 높은 조정기로 돌아가도록 지정할 수 있습니다.

특정 조정기를 어떻게 활성화하는지에 대한 절차는, 3.2.2절. “CPUfreq 설정”의 절차 3.2. “CPUfreq 조정기 활성화하기”를 참조하십시오.

3.2.2. CPUfreq 설정

CPUfreq 조정기를 선택하고 설정하기 전에, 먼저 적절한 CPUfreq 드라이버를 추가해야 합니다.

절차 3.1. CPUfreq 드라이버를 추가하는 방법

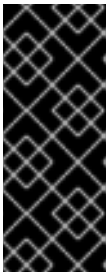
1. 다음 명령을 실행해서 시스템에 사용 가능한 CPUfreq 드라이버를 확인하십시오:

```
ls /lib/modules/[커널 버전]/kernel/arch/[아키텍처]/kernel/cpu/cpufreq/
```

2. 거기에서, **modprobe**를 사용해 적절한 CPUfreq 드라이버를 추가하십시오.

```
modprobe [CPUfreq 드라이버]
```

위의 명령을 사용할 때, **.ko** 파일 확장자를 빼고 입력하셔야 합니다.



중요

적절한 CPUfreq 드라이버를 고를 때, 항상 **p4-clockmod** 보다는 **acpi-cpufreq**를 사용하십시오. **p4-clockmod**를 드라이버를 사용하면, CPU의 클럭 주파수는 줄여주지만 CPU의 전압을 낮춰주지는 못합니다. 반면에, **acpi-cpufreq**는 CPU 클럭 주파수와 함께 전압도 낮춥니다. 따라서 성능을 감소시킬 때, 더 적은 전력을 소비하게 하고, 발열을 더 많이 줄일 수 있습니다.

3. CPUfreq 드라이버가 설정된 다음, 어떤 조정기를 시스템에서 현재 사용중인지를 다음과 같이 알아 볼 수 있습니다:

```
cat /sys/devices/system/cpu/cpu0/cpufreq/scaling_governor
```

또한, 다음 명령으로 특정 CPU에 사용 가능한 조정기가 어떤 것들이 있는지 볼 수 있습니다:

```
cat /sys/devices/system/cpu/[cpu ID]/cpufreq/scaling_available_governors
```

몇몇 CPUfreq 조정기를 사용할 수 없을지도 모릅니다. 그런 경우, **modprobe**를 실행해 사용하고자 하는 특정 CPUfreq 조정기를 활성화하는 데 필요한 커널 모듈을 추가하십시오. 이러한 커널 모듈은 **/lib/modules/[커널 버전]/kernel/drivers/cpufreq/**에 있습니다.

절차 3.2. CPUfreq 조정기 활성화하기

1. 특정 조정기가 CPU에서 사용할 수 있는 것으로 목록에 나타나지 않는다면, **modprobe**를 사용해 사용하고자 하는 조정기를 활성화하십시오. 예를 들어 **ondemand** 조정기를 사용할 수 없다면, 다음 명령을 실행하십시오:

```
modprobe cpufreq_ondemand
```

2. 조정기가 CPU에 사용 가능한 것으로 표시되면, 다음과 같이 그것을 활성화할 수 있습니다:

```
echo [governor] >
/sys/devices/system/cpu/cpu0/cpufreq/scaling_governor
```

3.2.3. CPUfreq 정책과 속도 튜닝하기

일단 적당한 CPUfreq 조정기를 선택했다면, **/sys/devices/system/cpu/*[cpu ID]*/cpufreq/**에 있는 값을 조정해서 각각의 CPU의 속도를 더 세밀하게 튜닝할 수 있습니다. 이러한 값들은 다음과 같습니다:

- **cpuinfo_min_freq** — CPU가 동작 가능한 최소 운용 주파수를 표시합니다. (KHz 단위)
- **cpuinfo_max_freq** — CPU가 동작 가능한 최대 운용 주파수를 표시합니다. (KHz 단위)
- **scaling_driver** — CPU의 주파수를 설정하기 위해서 사용되는 CPUfreq 드라이버.
- **scaling_available_governors** — 이 커널에서 사용 가능한 CPUfreq 조정기를 보여줌. 만약 이 파일에 없는 CPUfreq 조정기를 사용하고 싶다면, 3.2.2절. “CPUfreq 설정”에 있는 3.2.2절. “CPUfreq 설정”을 참조하시면 방법을 알 수 있습니다.
- **scaling_governor** — 현재 사용중인 CPUfreq 조정기를 표시합니다. 다른 조정기를 사용하려면, 단순히 **echo *[governor]* > /sys/devices/system/cpu/*[cpu ID]*/cpufreq/scaling_governor**를 사용하십시오. 자세한 내용은 3.2.2절. “CPUfreq 설정”에서 절차 3.2. “CPUfreq 조정기 활성화하기”를 참조하십시오.
- **cpuinfo_cur_freq** — 현재의 CPU 속도(KHz)
- **scaling_available_frequencies** — 해당 CPU에 설정 가능한 주파수들(KHz단위)
- **scaling_min_freq**와 **scaling_max_freq** — 해당 CPU의 *정책 한계(policy limits)*를 KHz로 지정.

  중요

  정책 한계를 지정할 때, **scaling_min_freq**를 지정하기 전에 **scaling_max_freq**를 설정하셔야만 합니다.

- **affected_cpus** — 주파소 조정 소프트웨어를 필요로 하는 CPU의 목록.
- **scaling_setspeed** — CPU의 클럭 속도를 변경하는 데 사용됨(KHz). 해당 CPU의 정책 한계 내의 속도만 설정할 수 있습니다(**scaling_min_freq**와 **scaling_max_freq**와 마찬 가지임).

각각의 튜닝 가능한 요소의 현재 값을 보려면, **cat *[tunable]***를 사용합니다. 예를 들어 현재 cpu0의 속도를 Khz로 보려면, 다음과 같이 합니다:

```
cat /sys/devices/system/cpu/cpu0/cpufreq/cpuinfo_cur_freq.
```

각각의 튜닝 가능한 항목의 값을 변경하려면, **echo *[value]* > /sys/devices/system/cpu/*[cpu ID]*/cpufreq/*[tunable]***를 사용합니다. 예를 들어, cpu0의 최저 클럭 속도를 360 Khz로 설정하려면 다음과 같이 합니다:

```
echo 360000 > /sys/devices/system/cpu/cpu0/cpufreq/scaling_min_freq
```

3.3. 전원 절약모드와 재개

시스템이 전원절약모드로 들어가면, 커널은 드라이버들에게 각자의 상태를 저장하고 자신을 메모리에서 해제라는 명령을 내립니다. 시스템이 재개될 때, 커널은 이러한 드라이버를 다시 메모리에 적재하며, 이들은 자신의 장치를 재프로그래밍 하게 됩니다. 드라이버가 이러한 작업을 성공적으로 할 수 있느냐가 시스

템이 정상적으로 재개될 수 있는지를 결정합니다.

비디오 드라이버는 이런 관점에서 볼 때 가장 문제가 많습니다. 왜냐하면 *고급 설정/전원 인터페이스 (Advanced Configuration and Power Interface)*(ACPI) 규격은 시스템 펌웨어가 비디오 하드웨어를 재 프로그래밍하는 것을 요구하지 않기 때문입니다. 따라서 비디오 드라이버가 완전히 초기화가 안된 상태에서도 하드웨어를 프로그램할 수 있지 않다면, 그로 인해 시스템이 재개하지 못하게 될 수 있습니다.

Red Hat Enterprise Linux 6는 새로운 그래픽 칩셋을 더 훌륭하게 지원합니다. 이로 인해 더 많은 플랫폼에서 전원절약상태와 재개를 사용할 수 있게 되었습니다. 특히 NVIDIA 칩셋에 대한 지원이 매우 많이 향상되었습니다; 특히 GeForce 8800 시리즈가 그렇습니다.

3.4. 틱없는 커널

예전에는 Linux 커널은 플랫폼에 따라 미리 정해진 주파수—100 Hz, 250 Hz, 또는 1000 Hz—로 시스템의 각각의 CPU에 인터럽트를 걸었습니다. 커널은 CPU에게 수행중인 프로세스에 대해 질의하고, 그 결과를 프로세스 어카운팅과 로드 밸런싱에 활용했습니다. 이 인터럽트는 *타이머 틱(timer tick)*으로 알려져 있으며, 커널은 이 인터럽트를 CPU의 전원 상태와 관계 없이 수행했습니다. 따라서, 심지어는 유휴 CPU라도 매 초마다 이러한 요청에 1000번씩 답을 했었습니다. 전원 절약이 구현되어 있는 시스템에서도 유휴 CPU에 대한 측정이 이루어지기 때문에, 타이머 틱은 CPU가 충분히 긴 시간 유휴 상태에 남아있는 것을 방해해서, 전원 절약 효과를 충분히 얻지 못하는 원인이었습니다.

Red Hat Enterprise Linux 6의 커널은 *틱없이(tickless)* 실행됩니다: 이는, 오래된 주기적인 타이머를 요구불(on-demand) 인터럽트로 바꿨다는 의미입니다. 따라서, 유휴 CPU는 새로운 태스크가 처리 대기열에 들어오지 않는 이상 유휴 상태로 남아있게 되고, 저전력 상태에 들어간 CPU는 그 상태에 더 오래 남아있을 수 있습니다.

3.5. 활성 상태 전원 관리(ACTIVE-STATE POWER MANAGEMENT)

*활성 상태 전원 관리(Active-State Power Management)*는 *외장 콤포넌트 고속 연결(Peripheral Component Interconnect Express)*(PCI Express 또는 PCIe) 서브시스템의 전원 소비를 PICe 연결로 접속된 장치가 사용중이 아닐 때 저전력 상태로 변경해서 절약합니다. ASPM은 연결의 양 종단의 전원 상태를 제어하며, 연결의 한쪽 끝에 있는 장치가 완전히 전원이 켜져있는 상태일지라도 전력 소비를 줄여줍니다.

ASPM이 활성화되면, 서로 다른 전원 상태에 있는 연결 양 끝단의 장치들이 상태를 변경해야 하기 때문에 지연 시간이 발생합니다. ASPM은 전원 상태를 결정하는 데 3가지 정책을 사용합니다:

default

PICe 연결의 전원 상태를 시스템의 펌웨어(예: BIOS)에 지정된 디폴트 상태로 설정합니다. 이는 ASPM의 디폴트 상태입니다.

powersave

ASPM을 성능 감소를 감수하고라도 가능한 한 전력을 덜 소비하도록 설정합니다.

performance

ASPM을 비활성화해서 PCIe 연결이 최대 성능을 발휘하도록 합니다.

ASPM 정책은 **/sys/module/pcie_aspm/parameters/policy**에 설정되어 있습니다. 하지만 시스템 부팅시 ***pcie_aspm*** 커널 매개변수를 사용해 설정할 수도 있습니다. **pcie_aspm=off**이라고 하면 ASPM을 비활성화하며, **pcie_aspm=force**는 ASPM을, 심지어는 ASPM을 지원하지 않는 장치에 대해서 까지, 활성화합니다.

주의

pcie_aspm=force를 설정하면, ASPM을 지원하지 않는 하드웨어로 인해 시스템이 멈출 수 있습니다. **pcie_aspm=force**를 지정하기 전에, 시스템의 모든 PCIe 하드웨어가 ASPM을 지원하는지 확인하십시오.

3.6. 적극적 연결 전원 관리(AGGRESSIVE LINK POWER MANAGEMENT)

*적극적 연결 전원 관리(Aggressive Link Power Management)*는 디스크에 대한 SATA 연결을 유휴 기간(즉, I/O가 없을 때)에 저전력으로 설정함으로써, 전원을 절약하는 기술입니다. I/O 요청이 해당 연결의 대기열에 들어오면 ALPM이 자동으로 SATA 연결을 활성 전원 상태로 전환시킵니다.

ALPM으로 전원을 절약하기 위해서는 디스크 액세스 지연이라는 댓가를 치뤄야 합니다. 따라서, 긴 I/O 유휴시간이 예상되는 시스템에서만 ALPM을 사용해야 합니다.

ALPM은 *고급 호스트 컨트롤러 인터페이스(Advanced Host Controller Interface)*(AHCI)를 사용하는 SATA 컨트롤러에서만 사용 가능합니다. AHCI에 대한 더 많은 정보는 http://www.intel.com/technology/serialata/ahci.htm를 참조하십시오.

사용 가능한 경우, ALPM은 디폴트로 활성화 되어 있습니다. ALPM은 3가지 모드가 있습니다:

min_power

이 모드는 링크를 디스크 I/O가 없는 때에 최소 전력 상태(SLUMBER)로 설정합니다. 이 모드는 오랜 유휴시간이 예상되는 경우 유용합니다.

medium_power

이 모드는 디스크 I/O가 없을 때, 연결을 두 번째로 낮은 전원 상태(PARTIAL)로 설정합니다. 이 모드는 가능한 한 성능에 적은 영향을 끼치면서, 연결의 전원 상태를 전환하기 위해 설계되었습니다(예를 들어 간헐적으로 많은 I/O와 유휴 I/O가 일어나는 경우).

medium_power 모드에서는 연결이 부하에 따라 PARTIAL과 최대 전원(즉, "ACTIVE") 상태 사이를 오갈 수 있습니다. 연결의 전원 상태가 PARTIAL에서 SLUMBER로 직접 바뀌거나, 그 반대로 직접 바뀔 수 없다는 것에 주의하십시오; 두 경우 모두 ACTIVE 상태로 일단 전환된 다음에, 다른 상태로 변경될 수 있습니다.

max_performance

ALPM이 비활성화됩니다; 연결은 디스크에 I/O가 없어도 저전력 상태로 전환되지 않습니다.

SATA 호스트 아답터가 실제로 ALPM을 지원하는지 체크하려면, **/sys/class/scsi_host/host*/link_power_management_policy** 파일이 있는지를 봐야 합니다. 설정을 변경하려면 여기서 설명한 값을 이 파일에 기록하고, 현재 설정을 검토하려면 이 파일을 표시해보면 됩니다.

중요

ALPM을 **min_power**나 **medium_power**로 설정하면, "핫플러그(Hot Plug)"특성은 자동으로 사용 불가능하게 됩니다.

3.7. 실시간 드라이브 액세스 최적화

POSIX 표준은 운영 체제가 각각의 파일의 최종 액세스 시간을 기록하는 파일 시스템의 메타정보를 유지할 것을 요구합니다. 이 타임스탬프는 **atime**라 불리며, 이를 유지하기 위해서는 저장소에 계속해서 데이터를 기록할 필요가 있습니다. 이렇게 데이터를 기록하는 것으로 인해 저장소와 그 연결의 상태를 사용중으로, 전원을 계속 켜진 상태로 유지하게 됩니다. **atime** 데이터를 사용하는 어플리케이션이 별로 많지 않기 때문에, 이렇게 저장소 장치가 동작하면 전력을 낭비하게 됩니다. 특히, 심지어는 파일을 저장소에서 읽지 않고 캐시에서 읽은 경우에도 저장소에 기록하는 작업이 이뤄져야 합니다. 때때로 Linux 커널은 **mount**시 **noatime** 옵션을 지원하기도 합니다. 이 옵션과 함께 마운트된 경우에는 **atime** 데이터를 파일시스템에 기록하지 않습니다. 하지만, 이렇게 무작정 해당 기능을 꺼버리는 것은 **atime**에 의존하는 몇몇 어플리케이션이 있고, 그것들에게 오류가 발생하게 되므로, 문제가 될 수 있습니다.

Red Hat Enterprise Linux 6의 커널은 **relatime**라 불리는 대안을 지원합니다. **relatime**은 **atime** 데이터를 유지하지만, 파일 액세스가 있을 때 마다 하지는 않습니다. 이 옵션을 활성화시키면, **atime** 데이터가 파일의 최종 업데이트 날짜(**mtime**)와 같고 그 이후 파일이 변경된 경우거나, 파일의 최종 액세스 시간이 현재로부터 일정 시간 이전(디폴트는 하루)인 경우에만 디스크에 기록됩니다.

디폴트로 모든 파일시스템은 이제 **relatime** 옵션이 활성화된 상태로 마운트됩니다. 이 기능을 전체 시스템에서 사용하지 않으려면, 부트 매개변수로 ***default_relatime=0***를 지정하십시오. 만약 **relatime**이 디폴트로 시스템에서 활성화되어 있고, 특정 파일시스템에서 그 기능을 사용하고 싶지 않은 경우 **norelatime** 옵션으로 파일시스템을 마운트하면 됩니다. 마지막으로, 시스템이 파일의 **atime** 데이터를 업데이트하기 전에 디폴트 시간을 변경하려면, ***relatime_interval=*** 부트 매개변수를 사용해서, 시간을 초 단위로 지정합니다. 기본값은 **86400**입니다.

3.8. 전원 캡핑(POWER CAPPING)

Red Hat Enterprise Linux 6는 HP *동적 전력 캡핑(Dynamic Power Capping)*(DPC)나 Intel의 노드 관리자(Node Manager) 기술과 같은 최근의 하드웨어에서 지원되는 전원 캡핑(capping)을 지원합니다. 전원 캡핑은 관리자가 서버에 의해 소비되는 전력을 제한하도록 해줍니다. 또한 이는 관리자가 데이터 센터의 계획을 더 효율적으로 세울 수 있도록 해줍니다. 왜냐하면 기존의 전원 공급장치에 과부하가 걸릴 확률이 현저히 감소하기 때문입니다. 관리자들은 동일한 물리적인 요구량을 가지는 서버를 더 추가할 수 있고, 서버의 전원 소비가 제한된다면 부하가 많은 경우에도 가용 전력을 초과하지 않을거라는 것에 대해 더 많은 자신감을 가질 수 있습니다.

HP 동적 전력 캡핑(Dynamic Power Capping)

동적 전력 캡핑은 몇몇 ProLiant와 BladeSystem 서버에서 사용 가능한 특징으로, 시스템 관리자가 서버 또는 서버의 그룹에 대한 전력 사용량을 제한할 수 있는 기능입니다. 캡은 부하량과 관계 없이 서버가 넘어설 수 없는 확고한 한계입니다. 캡은 서버가 전원 사용 제한량에 도달가기 전까지는 효력을 나타내지 않습니다. 전력 제한량에 도달하면, 관리 프로세서가 CPU P-상태와 클럭 임계치를 조정해서 소비되는 전력을 제한합니다.

동적 전력 캡핑은 운영 체제와 독립적으로 CPU의 동작을 변경할 수 있지만, HP의 *통합 Lights-Out 2*(iLO2) 펌웨어는 운영 체제가 관리 프로세서를 세스하는 것을 허용하며, 그에 따라 사용자 공간의 어플리케이션이 관리 프로세서에 질의를 할 수 있게 됩니다. Red Hat Enterprise Linux 6에 사용된 커널은 HP iLO와 iLO2 펌웨어에 대한 드라이버를 포함하며, 그에 따라 프로그램들은**/dev/hpilo/d*Xccb*N***에 있는 관리 프로세서에 질의할 수 있습니다. 커널은 또한 전원 캡핑을 지원하기 위한 **hwmon sysfs** 인터페이스의 확장을 포함합니다. 또한, **sysfs** 인터페이스를 사용하는 **hwmon** ACPI 4.0 전력계 드라이버도 제공합니다. 이런 특징들이 한데 어우러져서, 운영 체제와 사용자-영역의 도구를 사용해 전원 캡핑을 위해서 설정된 값을 읽고, 현재 시스템의 전력 소비량을 측정할 수 있도록 해 줍니다.

HP 동작 전원 캡핑에 대한 더 자세한 정보는 http://h20000.www2.hp.com/bc/docs/support/SupportManual/c01549455/c01549455.pdf에 있는 *HP Power Capping and HP Dynamic Power Capping for ProLiant Servers*를 보십시오.

인텔 노드 관리자

인텔 노드 관리자는 시스템에 전력 사용량 제한을 겁니다. 이는 프로세서의 P-상태와 T-상태를 활용해 CPU의 성능을 제약함으로써, 전력 소비를 제한하는 것입니다. 전원 관리 정책을 결정함으로써, 관리자는 밤이나 주말과 같이 시스템이 시스템 부하가 적은 경우, 더 작은 전력을 소비할 수 있도록 할 수 있습니다.

인텔 노드 관리자는 CPU의 상태를 *운영체제를 통한 설정과 전원 관리(Operating System-directed configuration and Power Management)*(OSPM)를 사용해 조정합니다. 인텔 노드 관리자가 OSPM 드라이버에게 T상태로의 변화를 통지하면, 드라이버는 프로세스의 P-상태를 그에 따라 변화시킵니다. 이와 비슷하게, 인텔 노드 관리자가 OSPM드라이버에게 P-상태의 변화를 통지하면, 드라이버는 그에 따라 T상태를 변경시킵니다. 이러한 변경은 자동으로 일어나고, 운영체제에서 추가적인 입력을 필요로 하지 않습니다. 관리자는 인텔 노드 관리자를 *인텔 데이터 센터 관리자(Intel Data Center Manager)*(DCM)소프트웨어를 사용해 설정하고 모니터할 수 있습니다.

인텔 노드 관리자에 대한 더 자세한 상황은 http://communities.intel.com/docs/DOC-4766에 있는 *Node Manager — A Dynamic Approach To Managing Power In The Data Center*를 살펴보십시오.

3.9. 향상된 그래픽 전원 관리

Red Hat Enterprise Linux 6는 불필요하게 전력을 소모하는 원인을 몇 가지 제거해서 그래픽과 디스플레이 장치의 전력 소비를 감소시켰습니다.

LVDS 재클럭킹

저전압 차등 신호(Low-voltage differential signalling)(LVDS)는 구리선으로 전기 신호를 전달하는 시스템입니다. 이 시스템의 중요한 응용은 픽셀 정보를 노트북의 *액정 디스플레이(liquid crystal display)*(LCD) 화면에 전달하는 것입니다. 모든 디스플레이는 *갱신 비율(refresh rate)*, 즉 그래픽 컨트롤러에서 새 데이터를 받아와서 화면에 그 이미지를 다시 그려주는 초당 횟수가 있습니다. 전형적인 LCD는 보통 1초에 60번 데이터를 새로 받아서 그려줍니다(주파수로 60 Hz). 화면과 그래픽 컨트롤러가 LVDS로 연결되어 있을 때, LVDS 시스템이 매 갱신 주기마다 전력을 소비합니다. 대부분의 LCD 화면의 갱신 비율을 유휴시간에 30 Hz까지 사용자에게 눈에 띄는 효과를 미치지 않으면서 떨어뜨릴 수 있습니다(갱신 비율을 감소시키면 깜빡임을 야기하는 *음극선관(cathode ray tube)*(CRT) 모니터와는 다릅니다). Red Hat Enterprise Linux 6에 포함된 인텔 그래픽 아답터 드라이버는 이러한 *클럭낮추기(downclocking)*를 자동으로 수행해서, 화면이 유휴상태일 때 소비전력을 0.5 W 근처로 낮출 수 있습니다.

메모리 자체-갱신 활성화

그래픽 아답터에 쓰이는 *동기화된 동적 랜덤 액세스 메모리(Synchronous dynamic random access memory)*(SDRAM)은 개별 메모리 셀에 저장된 데이터를 유지하기 위해 1초에 수천번 갱신됩니다. 원래의 기능인 메모리에서 데이터를 가져오고, 메모리에 데이터를 쓰는 작업 이외에, 이 갱신작업을 시작하는 것도 메모리 컨트롤러의 책임입니다. 하지만, SDRAM은 또한 처전력 *자체 갱신* 모드를 제공합니다. 이 모드에서, 메모리는 자기 자신의 갱신 주기를 만들기 위해 내부 타이머를 활용합니다. 이에 따라 시스템은 메모리 컨트롤러를 메모리에 저장된 데이터를 잃지 않으면서도 중단시킬 수 있습니다. Red Hat Enterprise Linux 6에 사용된 커널은 유휴시 인텔 그래픽 아답터의 메모리 자체 갱신을 활성화 시켜서, 0.8 W 가까운 전력을 절약할 수 있습니다.

GPU 클럭 낮추기

전형적인 그래픽 처리 유니트(GPU)는 내부 회로의 여러 부분을 관장하는 내부 클럭을 포함합니다. Red Hat Enterprise Linux 6에 사용된 커널은 인텔과 ATI GPU 중 일부의 내부 클럭 주파수를 낮출 수 있습니다. 어떤 주어진 시간 동안 GPU 구성요소가 수행하는 작업의 사이클을 줄이는 것을 통해 GPU가 수행하지 않아도 될 사이클에서 소모하는 전력을 줄일 수 있습니다. 커널은 자동으로 GPU가 유휴상태일 때 이러한 클럭의 속도를 줄이며, GPU의 활동이 늘어나면 클럭을 다시 높여줍니다. GPU 클럭을 줄임으로써 최대 5 W의 전력을 절약할 수 있습니다.

GPU 전원 차단

Red Hat Enterprise Linux 6의 인텔과 ATI 그래픽 드라이버는 어답터에 모니터가 연결되지 않은 경우를 감지해서, 그런 경우 GPU를 완전히 꺼줍니다. 이 기능은 특별히 보통 모니터를 연결해 놓지 않는 서버들의 경우 매우 중요합니다.

3.10. RFKILL

많은 컴퓨터 시스템에는 Wi-Fi, 블루투스, 3G 장치와 같은 전파 송수신 장치가 포함되어 있습니다. 이러한 장치들은 전력을 소비합니다. 만약 장치가 사용중이 아니라면, 이는 낭비입니다.

*RFKill*은 리눅스 커널의 서브시스템으로 컴퓨터의 전파 송수신 장치들에 대해 질의하고, 활성화하고, 비활성화 할 수 있도록 해주는 인터페이스를 제공합니다. 송수신 장치가 비활성화되는 경우, 소프트웨어가 그 장치를 재활성화 시킬수 있는 상태(*소프트 블럭(soft block)*)나 소프트웨어로는 그 장치를 재활성화 시킬 수 없는 상태(*하드 블럭(hard block)*) 중 하나로 장치의 상태가 변화됩니다.

RFKill 핵심부는 서브시스템에 API를 제공합니다. RFkill을 지원하는 것으로 서명된 커널 드라이버는 이러한 API를 활용해 커널에 자신을 등록하고, 장치를 활성화하고 비활성화하는 방법을 제공합니다. 또한, RFKill 핵심부는 사용자 어플리케이션이 이해할 수 있는 통지를 제공하며, 사용자 어플리케이션이 송수신 장치의 상태를 질의할 수 있는 방법을 제공합니다.

RFKill 인터페이스는 **`/dev/rfkill`**에 위치하며, 그 안에는 시스템의 모든 전파 송수신 장치의 상태정보가 포함되어 있습니다. 각각의 장치는 현재의 RFKill 상태를 **`sysfs`**에 등록해 둡니다. 추가적으로, RFKill이 활성화되어 있는 장치의 상태가 변화되면 RFKill은 *uevents* 이벤트를 발생시킵니다.

Rfkill은 명령행 도구로 시스템의 RFKill-지원 장치에 질의를 하고, 상태를 변경할 수 있도록 해줍니다. 이 도구를 사용하시려면, rfkill 패키지를 설치하십시오.

`rfkill list`를 사용해 장치의 목록을 얻습니다. 각각의 장치는, **`0`**번 부터, 연관된 *인덱스 번호*를 부여받습니다. 이 인덱스 번호를 사용해 **rfkill**에 어떤 장치를 블럭하거나 블럭을 해제해도록 지정할 수 있습니다. 예를 들면:

```
rfkill block 0
```

은 시스템의 첫번째 RFKill-지원 장치를 블럭합니다.

또한 **rfkill**를 사용해 특정 종류의 장치들이나 모든 RFKill-지원 장치들을 블럭할 수도 있습니다. 예를 들어:

```
rfkill block wifi
```

은 시스템의 모든 Wi-Fi 장치를 블럭합니다. 모든 RFKill-지원 장치를 블럭하려면, 다음과 같이 합니다:

```
rfkill block all
```

블럭된 장치를 해제하려면 **`rfkill block`** 대신 **`rfkill unblock`**를 사용합니다. **rfkill**이 블럭할 수 있는 장치들의 종류를 확인하려면 **`rfkill help`**를 실행하십시오.

3.11. 사용자 공간에서의 최적화

시스템 하드웨어가 수행하는 작업의 양을 줄이는 것은 전력 소모를 줄이는 기본입니다. 3장. *핵심 인프라와 기법*에 설명한 변경이 시스템이 전원을 덜 소비하는 다양한 상태로 동작하도록 해 주지만, 사용자 공간의 어플리케이션이 시스템 하드웨어에 불필요한 작업을 요청한다면, 하드웨어가 그런 에너지 절약 상태로 들어갈 수 없습니다. Red Hat Enterprise Linux 6를 개발하는 동안, 하드웨어에 불필요한 요청을 줄이기 위해 다음과 같은 영역에 대한 분석이 이루어졌습니다.

CPU깨우기의 감소

Red Hat Enterprise Linux 6는 *틱 없는 커널(tickless kernel)*을 사용합니다(3.4절. “틱없는 커널” 참조). 이는 CPU가 더 깊은 유휴 상태에 더 오래 있도록 해 줍니다. 하지만, *타이머 틱(timer tick)*만이 과도하게 CPU를 깨우는 원인은 아니며, 어플리케이션에서 함수를 호출하는 것도, CPU가 유휴 상태에 들어가거나, 그 상태에 남아있지 못하는 원인이 됩니다. 불필요한 함수 호출을 50개 이상의 어플리케이션에서 감소시켰습니다.

저장소와 네트워크 IO의 감소

저장소 장치와 네트워크 인터페이스에 대한 입출력(IO)이 있으면 장치들이 전력을 소비하게 됩니다. 유휴시 저전력 상태를 가질 수 있는 저장소와 네트워크 장치(예: ALPM이나 ASPM)에서, 이러한 트래픽은 해당 장치가 유휴 상태에 들어가거나 남아있는 것을 막게 되며, 하드 드라이브가 사용중이 아닐 때 회전수를 감소시키지 못하게 됩니다. 저장소에 대한 불필요하거나 과도한 요청을 몇몇 어플리케이션에서 최소화 시켰습니다. 특히, 하드 드라이브를 계속 돌게 만드는 몇몇 어플리케이션들을 수정했습니다.

Init 스크립트 분석

사용 여부와 관계 없이 자동으로 시작되는 서비스들은 시스템 자원을 낭비할 가능성이 매우 큽니다. 대신, 서비스들은 가능한 한 디폴트로 "미사용"이거나 "요청시 사용"으로 설정되야 합니다. 예를 들어 Bluetooth를 지원하는 **BlueZ** 서비스는 예전에는 Bluetooth 장치의 존재 여부와 관계 없이 자동으로 시스템 시작시 실행되었습니다. 이제는 **BlueZ** initscript가 Bluetooth 장치가 시스템에 있는지를 서비스 시작 전에 검사하도록 변경되었습니다.

4장. 사용 사례

이번 장은 이 가이드의 다른 부분에서 다룬 분석과 설정 방법을 잘 보여주는 두 가지 사용 예를 다룹니다. 첫번째 예는 전형적인 서버에 대한 것이고, 두번째는 전형적인 랩탑의 경우입니다.

4.1. 예 — 서버

최근의 전형적인 서버는 기본적으로 Red Hat Enterprise Linux 6이 지원하는 모든 필요한 하드웨어 구성을 포함하고 있습니다. 고려해야 할 첫번째 요소는 서버가 주로 사용하게 될 부하의 종류가 무엇인지입니다. 이 정보에 따라서 어떤 구성요소를 전력 소비 절약을 위해 최적화할 지 결정할 수 있습니다.

서버의 중류와 관계 없이, 보통 그래픽 성능은 필요하지 않습니다. 따라서 GPU 전원 절약은 항상 켜둘 수 있습니다.

웹서버

웹서버는 네트워크와 디스크 I/O를 필요로 합니다. 위부 연결 속도에 따라서 100 Mbit/s가 충분할 수도 있습니다. 만약 기계가 대부분 정적인 페이지를 서비스한다면, CPU 성능은 그리 중요하지 않을 수도 있습니다. 따라서 전원 관리 선택 사항은 다음을 포함할 수 있을 것입니다:

- **tuned**에 대해 디스크/네트워크 플러그인을 제외합니다.
- ALPM을 켭니다.
- **ondemand** 조정기를 켭니다.
- 네트워크 카드를 100 Mbit/s로 제한합니다.

계산 서버

계산 서버는 주로 CPU를 필요로 합니다. 전원 관리 선택은 다음과 같이 할 수 있습니다:

- 수행하려는 작업과 데이터를 저장하는 장소에 따라서, **tuned**에 대해 디스크나 네트워크 플로그인을 사용합니다. 배치 모드 시스템의 경우 **tuned**를 완전히 활성화 합니다.
- 활용도에 따라서, 아마도 **performance** 조정기를 사용해야 할 것입니다.

메일서버

메일서버는 주로 디스크 I/O와 CPU를 필요로 합니다. 전원 관리는 다음과 같을 수 있습니다:

- **ondemand** 조정기를 켭니다. 왜냐하면 마지막 몇 %의 CPU 성능을 더 발휘하는 것이 중요하지는 않기 때문입니다.
- **tuned**에 대해 디스크/네트워크 플러그인을 제외합니다.
- 네트워크 속도를 제한하지 않아야 할 것입니다. 왜냐하면 메일은 때로 내부용으로 쓰이며, 이런 경우 1 Gbit/s나 10 Gbit/s 연결의 잇점을 살릴 수 있을 것이기 때문입니다.

파일서버

파일서버는 메일서버와 유사한 요구사항을 필요로 합니다. 하지만, 사용할 프로토콜에 따라서, 더 많은 CPU성능을 필요로 할 수도 있습니다. 보통 삼바 기반의 서버가 NFS에 비해 더 많은 CPU 성능을 필요로 합니다. 또한 NFS는 보통 iSCSI보다 더 많은 CPU 성능을 필요로 합니다. 그럼에도 불구하고, **ondemand** 조정기를 사용할 수 있을 것입니다.

디렉터리 서버

보통 디렉토리 서버는 디스크 I/O에 대한 필요성이 적습니다. 이는 특히 RAM이 많은 경우 더 그렇습니다.

네트워크 응답 시간이 더 중요하지만, 네트워크 I/O는 덜 중요합니다. 사용자는 더 낮은 전송 속도에서 네트워크 응답 속도에 대한 튜닝을 진행하는 것을 검토할 수 있을 것입니다. 하지만, 사용하는 특정 네트워크에 따라서 매우 조심스럽게 테스트해야만 합니다.

4.2. 예 — 랩탑

전원 관리와 절약이 실제 큰 차이를 만들어낼 수 있는 가장 흔한 대상은 랩탑입니다. 랩탑 자체는 이미 설계상 에너지를 서버나 워크스테이션보다 극히 적게 사용하도록 만들어져 있어서 절대적인 에너지 절약 가능성은 다른 기계들보다 적습니다. 하지만 배터리 모드로 있을 때, 어떻게든 에너지를 절약하면 배터리 생명을 몇 분이라도 더 길게할 수 있습니다. 이번 절이 배터리 모드에 있는 랩탑에 초점을 맞추고 있긴 하지만, 여기 있는 튜닝 방법중 일부는 AC 전원에서도 사용할 수 있습니다.

개별 구성요소의 에너지 절약은 랩탑에서 워크스테이션 보다 상대적으로 더 많은 차이를 만들어냅니다. 예를 들어 1 Gbit/s 네트워크 인터페이스가 100 Mbit/s로 도는 경우 3–4 와트를 절약할 수 있습니다. 전체 에너지 소비가 400 와트정도 되는 전형적인 서버의 경우, 이런 에너지 절약은 대략 1 %에 불과합니다. 전체 에너지 소모가 40 와트정도 되는 랩탑에서라면, 이렇게 하나의 구성요소에서 에너지를 절약하기만 해도 전체 에너지 소모를 10 %나 줄일 수 있습니다.

전형적인 랩탑에서 사용할 수 있는 구체적인 에너지 절약 최적화는 다음을 포함합니다:

- 시스템 BIOS에서 사용하지 않는 모든 하드웨어를 비활성화 시킵니다. 예를 들어, 병렬포트나 직렬포트, 카드리더, 웹캠, WiFi, 블루투스등이 가능한 대상이 될 수 있습니다.
- 화면을 편안히 보기 위해 최고 밝기가 필요하지 않은 경우라면, 화면 밝기를 가능한 한 어둡게 합니다. GNOME 데스크탑의 **System+Preferences → Power Management**를 사용하거나, KDE 데스크탑의 **Kickoff Application Launcher+Computer+System Settings+Advanced → Power Management**을 사용하거나, 명령행에서 **xbacklight**을 사용하면 됩니다; 또는 랩탑의 기능키를 사용합니다.
- **tuned-adm**의 **`laptop-battery-powersave`** 프로파일을 사용해 전체 에너지 절약 메카니즘을 활성화합니다. 성능과 하드디스크/네트워크의 응답시간이 영향받을 수 있다는 것을 명심하십시오.

추가적으로(또는 이에 대한 대안으로), 여러 시스템 설정에 작은 조정을 가할 수 있습니다.

- **ondemand** 조정기를 사용합니다(Red Hat Enterprise Linux 6에서는 디폴트)
- 랩탑 모드를 활성화합니다(**`laptop-battery-powersave`** 프로파일의 일부):

```
echo 5 > /proc/sys/vm/laptop_mode
```

- 디스크 플러시 시간 간격을 늘립니다(**`laptop-battery-powersave`** 프로파일의 일부):

```
echo 1500 > /proc/sys/vm/dirty_writeback_centisecs
```

- nmi 와치독을 비활성화합니다(**`laptop-battery-powersave`** 프로파일의 일부):

```
echo 0 > /proc/sys/kernel/nmi_watchdog
```

- AC97 오디오 전원 절약을 활성화(Red Hat Enterprise Linux 6에서는 디폴트로 활성화됨):

```
echo Y > /sys/module/snd_ac97_codec/parameters/power_save
```

- 멀티코어 전원절약을 활성화(**`laptop-battery-powersave`** 프로파일의 일부):

```
echo 1 > /sys/devices/system/cpu/sched_mc_power_savings
```

- USB 자동 중단(auto-suspend을 활성화:

```
for i in /sys/bus/usb/devices/*/power/autosuspend; do echo 1 > $i;
done
```

  USB 자동 중단이 모든 USB 장치에서 잘 동작하는 것은 아님을 알아두십시오.

- ALPM 최소 전력 설정 활성화(**laptop-battery-powersave** 프로파일의 일부):

```
echo min_power >
/sys/class/scsi_host/host*/link_power_management_policy
```

- relatime을 사용해 파일시스템 마운트(Red Hat Enterprise Linux 6에서는 디폴트):

```
mount -o remount,relatime mountpoint
```

- 하드드라이브의 최대 전원 절약 모드를 활성화(**laptop-battery-powersave** 프로파일의 일부):

```
hdparm -B 1 -S 200 /dev/sd*
```

- CD-ROM 폴링 금지(**laptop-battery-powersave** 프로파일의 일부):

```
hal-disable-polling --device /dev/scd*
```

- 화면 밝기를 **50** 또는 그 이하로 감소시키기. 예:

```
xbacklight -set 50
```

- 유휴시 화면의 DPMS를 활성화:

```
xset +dpms; xset dpms 0 0 300
```

- Wi-Fi 전력 수준을 낮춤(**laptop-battery-powersave** 프로파일의 일부):

```
for i in /sys/bus/pci/devices/*/power_level ; do echo 5 > $i ; done
```

- Wi-Fi를 사용하지 않음:

```
echo 1 > /sys/bus/pci/devices/*/rf_kill
```

- 유선 네트워크 속도를 100 Mbit/s로 제한함(**laptop-battery-powersave** 프로파일의 일부):

```
ethtool -s eth0 advertise 0x0F
```

부록 A. 개발자를 위한 팁

좋은 프로그래밍 교재라면 메모리 할당의 문제점과 개별 함수의 성능에 대해 다루기 마련입니다. 소프트웨어를 개발할 때, 이러한 이슈가 소프트웨어를 실행할 시스템의 전력 소모를 증가시킬 수 있다는 것을 인식하고 있어야 합니다. 전력 소모에 대한 고려가 코드의 모든 부분에 영향을 끼치지는 않을지 몰라도, 성능상 병목이 될 수 있는 부분에 대해서는 최적화를 수행할 수 있읍니다.

다음과 같은 기법은 자주 문제를 일으키곤 합니다:

- 쓰레드의 사용.
- 불필요한 CPU 깨우기를 야기하며, 깨어난 CPU를 효율적으로 활용하지 못하는 것. 만약 CPU를 깨워야만 한다면, 해야할 일 전체를 깨어난 CPU에서 한꺼번에 가능한 한 빨리 수행하십시오(race to idle, 역주: CPU 절전기능이 뛰어나기 때문에 CPU가 깨어있을 때는 최대한 빠른 속도로 필요한 작업을 수행하고, 유휴상태로 남는 시간을 최대한 늘려야 전력 소모를 최소화할 수 있다는 것).
- **`[f]sync()`**를 불필요하게 사용하는 것.
- 불필요한 액티브 폴링 또는, 짧게 주기적으로 타임아웃을 사용하는 것. (대신 이벤트에 반응하도록 수정하십시오).
- 깨어난 CPU를 효율적으로 사용하지 않는 것.
- 비효율적인 디스크 액세스. 잦은 디스크 액세스를 방지하기 위해 커다란 버퍼를 사용하십시오. 한번에 커다란 블럭을 쓰도록 하십시오.
- 비효율적인 타이머 사용. 가능한 한 어플리케이션 사이에(가능한 한 시스템 전반에 걸쳐서도) 타이머를 그룹화하도록 하십시오.
- 과도학 I/O, 전력 소모, 또는 메모리 사용(메모리 누수 포함)
- 불필요한 계산 수행.

다음 절에서는 이러한 분야 중 일부를 훨씬 자세히 다룰 것입니다.

A.1. 쓰레드의 사용

쓰레드를 프로그램에 사용하는 것이 성능을 더 좋게하고 빠르게 한다고 널리 알려져 있지만, 항상 그런것은 아닙니다.

파이썬(Python)

Python은 전역 잠금 인터프리터(Global Lock Interpreter)[1]를 사용하며, 이에 따라 쓰레드를 사용하는 것은 오직 커다란 I/O 연산에서만 이득이 있습니다. **Unladen-swallow**[2]가 코드를 최적화하는 데 사용할 수 있는 더 빠른 파이썬 구현입니다.

펄(Perl)

펄 쓰레드는 원래 fork를 하지 않는 시스템(32비트 Windows 운영체제 등)에서 동작하는 프로그램을 위해 만들어졌습니다. 펄 쓰레드에서 데이터는 모든 개별 쓰레드에 복사됩니다(쓰기시 복사,Copy On Write). 사용자들이 데이터 공유의 수준을 결정할 수 있기 때문에, 데이터는 기본적으로 공유되지 않습니다. 데이터를 공유하기 위해서는 **threads::shared** 모듈을 포함시켜야 합니다. 하지만, 그렇게 할 경우 데이터가 (쓰기시 복사에 의해) 복사될 뿐아니라, 공유 모듈에 의해서 해당 데이터와 연계된 변수도 만들어집니다. 이 작업은 더 시간이 걸리고 더 느립니다.[3]

C

C 쓰레드는 메모리를 공유하고, 개별 쓰레드마다 자신만의 스택이 있습니다. 또한 커널은 쓰레드에 대해 새로운 파일 디스크립터를 할당하거나 새로운 메모리 공간을 할당할 필요가 없습니다. C는 실제로 더 많은 쓰레드에 대해 더 많은 CPU지원을 활용할 수 있습니다. 따라서 쓰레드의 성능을 최고로 끌어올리기 위해, C나 C++과 같은 저수준 언어를 사용하십시오. 만약 스크립트 언어를 사용한다면, C 바인딩(binding)을 작성하는 것을 고려해 보십시오. 프로파일러(profiler)를 사용해 코드에서 성능이 낮은 부분을 판별하십시오.[4]

A.2. 디스크 동작

많은 어플리케이션이 설정 파일이 변경되었는지 검사합니다. 이러한 검사는 1분 간격 같이 보통 고정된 주기마다 이루어집니다. 이것은 문제를 야기할 수 있습니다. 왜냐하면 이것은 디스크가 회전속도 감소상태에서 깨어나도록 하기 때문입니다. 가장 좋은 해결책은 적절한 시간 간격을 찾거나, 좋은 검사 방법을 생각해 내거나, 변경 검사를 **inotify**로 바꾸고 이벤트에 반응하도록 만드는 것입니다. **Inotify**는 파일이나 디렉터리에 일어난 여러가지 변경사항을 검사할 수 있습니다.

예를 들면 다음과 같습니다:

```
int fd;
fd = inotify_init();
int wd;
/* checking modification of a file - writing into */
wd = inotify_add_watch(fd, "./myConfig", IN_MODIFY);
if (wd < 0) {
  inotify_cant_be_used();
  switching_back_to_previous_checking();
}
...
fd_set rdfs;
struct timeval tv;
int retval;
FD_ZERO(&rdfs);
FD_SET(0, &rdfs);

tv.tv_sec = 5;
value = select(1, &rdfs, NULL, NULL, &tv);
if (value == -1)
  perror(select);
else {
  do_some_stuff();
}
...
```

이러한 접근 방법의 잇점은 다양한 검사를 할 수 있다는 것입니다.

주요 제약사항은 시스템에서 사용 가능한 감시의 숫자가 제한된다는 것입니다. 이 숫자는 **/proc/sys/fs/inotify/max_user_watches**에서 찾을 수 있으며, 변경할 수 있긴 하지만, 변경을 권장하지는 않습니다. 더구나, **/proc/sys/fs/inotify/max_user_watches**가 제공되지 않는 경우, 코드는 다른 체크 방법으로 복구를 시도해야 하며, 이는 보통 소스 코드의 여러 부분에 **#if #define**가 나타나는 것을 의미합니다.

inotify에 대한 추가 정보는, 다음 inotify 매뉴얼 페이지를 참조하십시오.

A.3. FSYNC

Fsync는 I/O 비용이 많이 드는 시스템 콜로 알려져 있습니다. 하지만 항상 그런것은 아닙니다. 예를 들어 Theodore Ts'o의 글 *Don't fear the fsync!* [5]과 그에 따르는 토론을 살펴보십시오.

Firefox는 **sqlite** 라이브러리를 사용자가 새로운 페이지로 이동하는 링크를 클릭할 때마다 호출하곤 합니다. **Sqlite**은 **fsync**를 부르고, 파일 시스템 설정(주로 ext3에 데이터 정렬 모드)에 따라서는, 아무 일이 일어자니 않더라도 응답 시간이 매우 길었습니다. 만약 다른 프로세스가 동시에 큰 파일을 복사하고 있는 중이라면 이 작업에 아주 긴 시간(최대 30초까지)이 걸릴 수도 있습니다.

하지만, **fsync**를 전혀 사용하지 않는 다른 경우에도 ext4 파일 시스템으로 변경하자 문제가 발생했습니다. Ext3은 데이터-정렬 모드 (data-ordered mode)로 설정되어, 몇초마다 메모리를 플러시해서 디스크에 저장했습니다. 하지만 ext4에서는 랩톱 모드 (laptop_mode)로 설정되어 저장 간격이 더 길어졌고, 시스템이 중간에 예기치 않게 꺼지면 데이터가 손실될 수 있었습니다. 이제 ext4가 패치가 되었지만, 여전히 프로그램을 설계할 때 **fsync**를 적절히 사용하도록 주의를 기울여야만 합니다.

다음의 간단한 설정 파일 읽고 쓰기 예는 파일의 백업을 만드는 방법과, 데이터가 사라질 수 있는 예를 보여줍니다:

```
/* open and read configuration file e.g. ~/.kde/myconfig */
fd = open("./kde/myconfig", O_WRONLY|O_TRUNC|O_CREAT);
read(myconfig);
...
write(fd, bufferOfNewData, sizeof(bufferOfNewData));
close(fd);
```

더 나은 접근 방식은 다음과 같습니다:

```
open("/.kde/myconfig", O_WRONLY|O_TRUNC|O_CREAT);
read(myconfig);
...
fd = open("/.kde/myconfig.suffix", O_WRONLY|O_TRUNC|O_CREAT);
write(fd, bufferOfNewData, sizeof(bufferOfNewData));
fsync; /* paranoia - optional */
...
close(fd);
rename("/.kde/myconfig", "/.kde/myconfig~"); /* paranoia - optional */
rename("/.kde/myconfig.suffix", "/.kde/myconfig");
```

---

[1] http://docs.python.org/c-api/init.html#thread-state-and-the-global-interpreter-lock

[2] http://code.google.com/p/unladen-swallow/

[3] http://www.perlmonks.org/?node_id=288022

[4] http://people.redhat.com/drepper/lt2009.pdf

[5] http://thunk.org/tytso/blog/2009/03/15/dont-fear-the-fsync/

부록 **B.** 개정 내역

고침 1.0-10.400 **2013-10-31** **Rüdiger Landmann**
Rebuild with publican 4.0.0

고침 1.0-10 **2012-07-18** **Anthony Towns**
Rebuild for Publican 3.0

고침 1.0-2 **Fri Oct 22 2010** **Rüdiger Landmann**
텍스트에 있는 작은 오류 수정

고침 1.0-1 **Thu Oct 7 2010** **Rüdiger Landmann**
"draft" 태그 삭제

고침 1.0-0 **Thu Oct 7 2010** **Rüdiger Landmann**
GA 릴리즈