멀티코어 가상머신 환경의 실시간 스케줄 가능성 분석

*유시환, **유혁
*고려대학교 컴퓨터학과
*고려대학교 융합전문 S/W 대학원
e-mail : {shyoo,hxy}@os.korea.ac.kr

Real-time Schedulability Analysis for Multi-core Virtual Machine

*Seehwan Yoo, **Hyuck Yoo
*Dept. of Computer Science, Korea University
**Graduate school of Convergence S/W, Korea University

요 약

최근 들어 가상화 기술은 서버의 통합에 뿐만 아니라, 임베디드 시스템에서도 널리 사용되고 있다. 하지만, 가상화 시스템에서는 물리 프로세서가 게스트 운영체제에게 직접 전달되지 않으며, 게스트 운영체제는 가상 프로세서를 통해서 실행할 수 밖에 없다. 따라서, 기존의 처리량 기준의 공평성 스케줄러가 가상머신 모니터에서 동작하는 경우, 실시간 스케줄링이 불가능하다. 본 연구에서는 멀티코어 기반의 가상화 시스템에서 실시간 태스크의 실행을 보장하는 기법을 소개한다. 특히, 본 논문에서는 계층형 스케줄링의 특성과 최대 병렬성 조건을 통하여 멀티코어 가상머신의 스케줄 가능성 분석 기법을 제시한다.

1. 서론[1]

가상머신 환경에서도 실시간 스케줄링은 매우 중요한 이슈이다. 가상화가 임베디드 환경에서도 활발히 적용됨에 따라 실시간 스케줄링 문제는 매우 중요한 문제로 두각되고 있다. 실시간 시스템은 단순히 프로세서의 사용량에 대한 공평성을 제공하는 문제가 아니라, 시간적인 실행의 엄격성을 제공해 주어야 한다. 따라서, 기존의 사용량을 기준으로 한 가상머신 스케줄링 기법들은 실시간 특성을 반영할 수 없다. 이러한 단점을 극복하기 위해서 가상머신 환경에서 실시간성을 보장하기 위한 연구들이 진행되고 있다.

본 논문에서는 멀티코어 가상머신 환경에서 실시간 스케줄링 분석 기법을 제시한다. 멀티코어 프로세서는 높은 처리율 이외에도 전력 효율성을 높일 수 있는 장점이 있어, 최근 임베디드 시스템에서 널리 사용되고 있다. 멀티코어 가상화 환경은 실시간 스케줄링을 훨씬 복잡하게 만든다. 그 이유는 별도의 프로세서가 각각 실행 컨텍스트를 가지고 병렬적으로 실행하며, 실행의 타이밍을 정확하게 맞추기 힘들기 때문이다.

가상화 환경에서 운영체제는 가상머신 모니터 (혹은 VMM)에 의해 제공되는 가상 프로세서를 사용한다. 가상 프로세서는 VMM 에 의해 스케줄링 되므로, 가상 프로세서 상에서 동작하는 게스트 운영체제는 물리 시간에 대한 고려없이 실행되며, 실시간 운영체제가 가지는 스케줄링 정책과는 별도로 동작하게 된다. 이와 마찬가지로, VMM 은 게스트 운영체제의 스케줄링에 대해 이해하지 못하므로 단순히 가상 프로세서의 스케줄링에만 초점을 두며 이들이 다른 가상머신에 대한 접근 제어와 성능 제어 및 고립에 중점을 둔다.

이처럼 계층적으로 분리된 스케줄링은 실시간 게스트 운영체제의 동작에 큰 영향을 미치게 된다. 계층적 스케줄링으로 인해, 실시간 게스트 운영체제는 기존의 실시간 시스템 분석 방법을 사용할 수 없으며, 실시간 성능을 보장할 수 없게 된다. 따라서, 가상화 환경의 실시간 스케줄링은 매우 어렵다. 그 이유는 가상머신 환경에서는 기존의 운영체제의 스케줄링 정책이 보장되지 않기 때문이다.

예를 들어, 대표적인 고정 우선 순위 스케줄러인 RM 스케줄러의 경우, 실시간 태스크 스케줄링을 보장하기 위하여 태스크들의 전체 프로세서 활용율을 계산한다. 이 때, 전체 프로세서 활용율이 실시간 보장을 위한 임계값을 넘지 않도록 함으로써 실시간 태스크의 실행을 보장할 수 있다. 임계값은 전체 물리 시간에 대하여, 우선순위에 따라 모든 태스크가 실행을 마칠 수 있는 시간에 대한 비율로 주어진다. 하지만, 가상머신 상에서 동작하는 실시간 게스트 운영체제는 물리적으로 일정 부분 만을 갖게 되므로, 전체 기존의 실시간 보장 방법을 통해서는 모든 태스크의 실행을 보장할 수 없다.

특히, 멀티코어와 같이 다중 프로세서 환경에서는 가

[1] 이 논문은 2010 년도 정부(교육과학기술부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임 (No. 2010-0029180)

상화가 적용되지 않은 환경에서도 실시간 스케줄링이 어려움을 감안할 때, 가상화를 통한 실시간 연구가 매우 필요한 상황이다.
본 연구에서는 계층형 스케줄링을 적용한 가상화 환경에서, 멀티코어 가상머신을 구동하는 경우, 고려해야 할 실시간 게스트 운영체제의 스케줄링 분석 방법을 제공한다. VMM 으로는 xen 을 고려하고 있으며, 실시간 게스트 운영체제는 고정 우선 순위기반 스케줄러를 가진 uC/OS-II 를 가정한다.

2. 관련연구

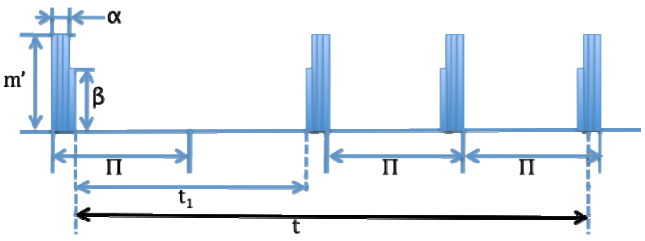
다중 프로세서 환경의 실시간성 보장 스케줄링 방법은 최근 들어 활발히 연구되고 있다. 고정 우선 순위와 가변 우선 순위 스케줄러에 대하여 실시간 보장을 위한 스케줄 가능성 검사 기법이 각각 [1]과 [2]에 소개되었다. 이 기법들은 다중 프로세서 환경의 특성상 프로세서 활용율을 최대로 활용할 수 있도록 하는 가운데, 임계시간을 넘어서는 첫번째 태스크를 가정하여 이 때의 프로세서 요구량을 계산하여, 다중 프로세서 환경의 실시간 스케줄링 보장 기법을 제시한다.
계층형 스케줄링과 관련되어 개방형 시스템에서 사용할 수 있는 실시간 계층형 스케줄링 연구로 조합형 스케줄링 이론이 제시되었다 [3]. 조합형 스케줄링은 하위 계층의 스케줄링 엔티티에 대한 추상화를 제공하여 상위 계층에서 제공하는 리소스 모델에 의해 제공되는 자원량을 정량화한다. 또한, 하위 계층의 스케줄링 엔티티에서 요구하는 자원량을 정량화하여 둘의 비교를 통하여 스케줄 가능한 자원 모델을 계산해 낸다. 이전의 연구에서는[4] 다중 프로세서 환경에서 gEDF 를 사용한 실시간 스케줄 가능성 분석에 대한 기법을 제시하였다. 하지만, 실제 많은 실시간 운영체제들은 고정 우선 순위 스케줄러를 가지고 있으며, 이 경우에 대한 정확한 분석은 이루어지지 않았다.
이러한 스케줄러들은 명시적으로 정의된 실행 시간과 실행 주기, 임계시간 등을 가지고 실시간 요구사항들을 계산해 낸다. 이와 다른 형태인 비율-공평 스케줄러(Proportional Fair 혹은 PFair 스케줄러)[5]는 태스크에 대해 진행율만을 명시하고, 태스크 실행 시간을 실행 윈도(Execution window)로 분할하여 각 태스크의 실행이 진행율을 가능한 한 일정하도록 유지하도록 함으로써 실시간 실행을 제공한다.
Xen 은 대표적인 가상머신 모니터들 중 하나이다[6]. Xen 은 Linux 나 Windows BSD 와 같은 다양한 상용 운영체제를 지원하고 있으며, 파일 서버, DB 서버, 웹서버 등의 네트워크 서버들을 하나로 통합하여 운영하는데 있어 효율적인 방법을 제공한다. Xen 은 가상머신을 생성/관리하며, 물리적 컴퓨팅 자원들을 각각 가상머신에게 할당/해제 하는 역할을 담당한다. Xen 은 물리 프로세서를 시분할 하여 가상 프로세서에게 제공하며, 실시간 지원을 위하여 EDF 기반의 스케줄러인 SEDF 와 크레딧 스케줄러를 탑재하고 있다. 크레딧 스케줄러는 가중치-공평(weighted fair-share) 스케줄러의 한 종류로서 가중치에 따라 실행 크레딧을 각 가상 프로세서에게 제공하며, 실행 크레딧 만큼씩 공평하게 나누어 실행한다[7]. 크레딧 스케줄러는 런 큐를 중간에 정렬하지 않으므로, I/O 응답시간이 예상보다 길어질 수 있다. 따라서, I/O 를 실행한 태스크들의 응답시간을 줄이기 위하여 BOOST 우선 순위를 제공한다[8].

3. 멀티코어기반 가상화 시스템의 스케줄가능성 분석

3.1. 프로세서 제공량

가상머신 모니터에서 제공하는 가상 프로세서 시간은 가상머신 모니터의 스케줄링 정책에 따라 달라진다. Xen 의 SEDF 스케줄러는 각 가상 프로세서에 대하여 (주기, 실행 슬라이스, 임계시간)을 정의하도록 하고 있다. 이 때, 주어진 가상 프로세서의 사용량을 보다 쉽게 계산하기 위하여, 가상 프로세서 m'개를 이용하여 실시간 게스트 운영체제에 대해 주어지는 시간을 먼저 구하도록 한다. 단, 마이그레이션 오버헤드는 없으며, 모든 태스크는 임의의 가상 프로세서에서 실행가능하다고 가정한다. Xen 은 gSEDF 스케줄러를 이용하여 게스트 운영체제에게 일정한 주기마다 일정한 실행 슬라이스를 보장할 수 있도록 스케줄링할 수 있다. 이러한 스케줄링 정책은 이전의 gEDF 와 동일하다.
gSEDF 의 스케줄링을 통해 제공받게 되는 물리 프로세서량은 MPR 모델을 이용하여 표시할 수 있다. MPR 모델은 다음과 같이 정의된다.

$$\mu = \langle m', \Pi, \Theta \rangle$$



그림은 MPR 모델에 의해 VMM 으로부터 제공되는 프로세서 시간의 하한을 도출하는 과정을 보여준다. m'은 제공되는 최대 병렬성(maximum concurrency), Π는 주기, Θ는 실행 시간을 의미한다. 이 때, 임의의 t 시간 동안 제공되는 물리 프로세서의 량을 정량화 하여 표시하면 다음과 같이 나타낼 수 있다 [4]. 또한, 이를 통해 얻은 선형 근사함수는 다음과 같이 계산된다.

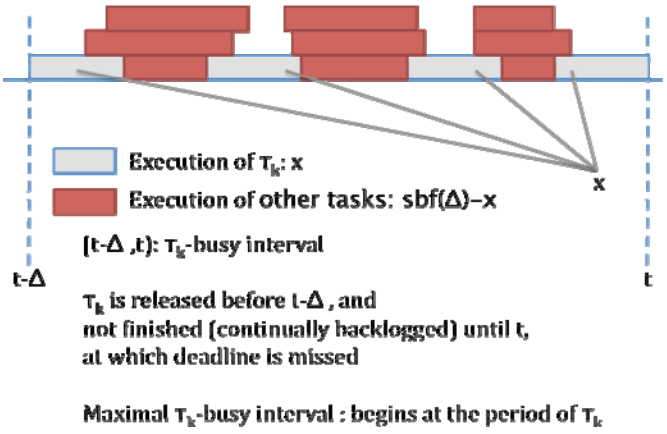
$$sbf(t) = \begin{cases} 0 & t < 0 \\ \left\lfloor \frac{t'}{\Pi} \right\rfloor \Theta + \max\{0, m'x - (m'\Pi - \Theta)\} & t' \geq 0, x \in [1, y] \\ \left\lfloor \frac{t'}{\Pi} \right\rfloor \Theta + \max\{0, m'x - (m'\Pi - \Theta) - (m' - \beta)\} & t' \geq 0, x \notin [1, y] \end{cases}$$

where $t' = t - \left(\Pi - \left\lceil \frac{\Theta}{m'} \right\rceil\right)$, $x = t' - \left\lfloor \frac{t'}{\Pi} \right\rfloor \Pi$, $y = \Pi - \left\lceil \frac{\Theta}{m'} \right\rceil$.

$$lsbf(t) = \frac{\Theta}{\Pi}\left(t - \left\lceil 2\left(\Pi - \frac{\Theta}{m'}\right) + 2 \right\rceil\right)$$

3.2. 작업량의 하한

게스트 운영체제는 다음과 같은 태스크 집합을 가진다. $S = \{\tau_i(p_i, e_i, d_i)\}$ 태스크 τ_i 는 우선 순위가 i 이며, 주기와 실행 시간, 임계 시간은 각각 p_i 와 e_i 이다. 태스크 τ_i 의 시간 구간 $[t-\Delta, t)$ 에서의 작업량(W_i)는 구간에서 실제 수행한 시간으로 정의하며, 부하는 W_i/Δ 와 같이 정의된다. k-수준 부하는 우선 순위가 k 보다 높은 태스크들의 모든 부하의 합으로 정의한다. 이 때, 태스크 τ_k 에 대하여, 최초로 놓친 임계시간 t 를 정의한다. 최초로 놓친 임계시간 t 는 태스크 τ_k 가 t 에서 임계시간이 지났으나, 실행을 마치지 못하였으며, t 이전에는 모든 태스크에 대하여 임계시간을 놓치지 않았음을 의미한다.



이 때, 그림에서 보이는 것처럼 최초로 놓친 임계시간 t 에 대하여 태스크 τ_k 가 실행을 시작하였으나 실행을 마치지 못한 구간을 τ_k-busy 구간으로 정의한다. 이 경우에 대하여, 최대 τ_k-busy 구간에서의 (k-1) 수준 부하는 다음과 같은 하한을 가진다.

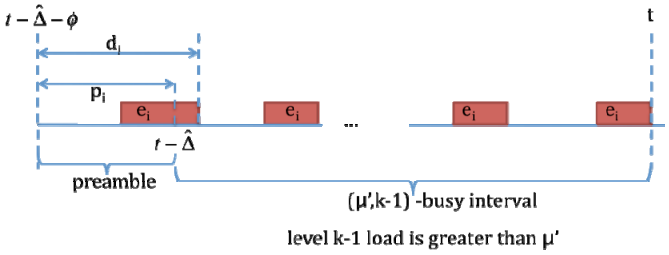
$$\sum_{i<k} \frac{W_i}{\Delta} \geq \mu'$$

$$\mu' = \begin{cases} \dfrac{\Theta}{\Pi} - \dfrac{e_k + 2\Theta\left(1 - \dfrac{\Theta}{m'}\right)}{d_k} & (d_k \leq p_k) \\ \dfrac{\Theta}{\Pi} - \dfrac{e_k}{p_k} & (d_k > p_k) \end{cases}$$

이전 연구에서 작업 부하의 하한을 최대 병렬성을 통해 구했던 것과 달리, 가상머신 환경에서는 MPR 모델에 의해 제공되는 sbf 의 정의와 최초로 놓친 임계시간의 정의로부터 하한을 유도할 수 있다. 내용의 증명은 분량의 문제로 추후 연구에서 자세히 밝히도록 한다.

3.3. 작업 부하의 상한

최초로 놓친 임계시간 t 에 대하여 각 태스크 τ_i 의 작업부하의 상한을 별도로 계산하도록 한다. 이를 위해 (μ', k-1)-busy 구간을 정의한다. (μ', k-1)-busy 구간은 t 로부터 k-1 수준 부하가 μ' 이상인 $[t-\hat{\Delta}, t)$ 이다. 이 중 가장 긴 구간을 최대 (μ', k-1)-busy 구간이라 한다.



이 구간을 기준으로 그림에서와 같이 t 로부터 각 태스크 τ_i 의 해당 주기의 시작까지 구간을 확장하여 preamble 을 정의하고, 주어진 구간에서 각 태스크의 부하의 상한은 다음과 같이 구할 수 있다.

$$\frac{W_i}{\hat{\Delta}} \leq \beta'(i)$$

$$\beta'(i) = \begin{cases} \min\left\{1, \ \dfrac{e_i}{p_i}\left(1 + \dfrac{p_i - e_i}{d_k}\right)\right\} & \dfrac{m'-\mu'}{m'-1} \geq \dfrac{e_i}{p_i} \\ \min\left\{1, \ \dfrac{e_i}{p_i}\left(1 + \dfrac{p_i - e_i}{d_k}\right) + \dfrac{d_i}{d_k}\left(\dfrac{e_i}{p_i} - \dfrac{m'-\mu'}{m'-1}\right)\right\} & \dfrac{m'-\mu'}{m'-1} < \dfrac{e_i}{p_i} \end{cases}$$

작업 부하의 상한은 최대 (μ', k-1)-busy 구간의 정의와 최대 병렬성 조건 및 앞서 구한 작업 부하 하한 조건으로 부터 유도 할 수 있다.

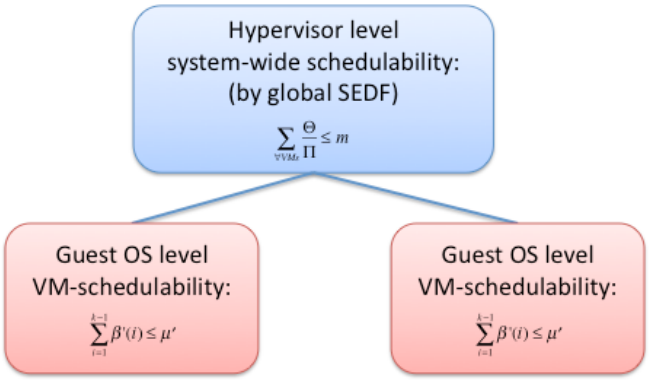
Preamble 을 포함한 구간에서의 작업량은 $\mu'(\phi + \hat{\Delta})$ 보다 작으므로, 이러한 조건을 활용하여 최대 병렬성을 고려하여 실행할 수 있는 작업량을 구하면, 전체 $[t-(\phi+\hat{\Delta}), t)$ 구간에서 실행하는 작업량의 상한을 얻을 수 있다.

3.4. 스케줄 가능성 검사 방법

모든 태스크 τ_i 의 부하의 상한의 합이 최대 τ_k-busy 구간에서의 부하의 하한 보다 작거나 같으면, (k) 보다 낮은 우선 순위의 모든 태스크가 스케줄 가능하다. 이를 식으로 나타내면 다음과 같다.

$$\sum_{i=1}^{k-1} \beta'(i) \leq \mu'$$

스케줄 가능성 검사는 주어진 가상머신에 대하여 가상머신 내의 모든 태스크가 스케줄 가능한지를 확인해 준다. 따라서, 이 스케줄 가능성 검사 기법은 개별적인 실시간 가상머신에 대해서 적용되지만, 다중 가상머신에 대한 보장을 의미하지는 않는다.



그림에서 보는 바와 같이 가상머신 기반 시스템에서 스케줄 가능성 검사를 위해서는, 각 계층에서 수행해야 할 스케줄 가능성 검사를 별도로 수행해야 한다. 즉, 가상머신 모니터 수준에서 global SEDF 를 통하여 각 가상머신 단위의 스케줄 가능성 검사를 별도로 수행해야 하며, VM 단위의 스케줄 가능성 검사를 위해서는 3.4 절의 앞에서 제시한 내용과 같은 별도의 스케줄 가능성 검사를 수행해야 한다.

4. 결론

멀티코어 가상머신에서 실시간 스케줄링을 보장하기 위해서는 멀티코어가 가지는 병렬성을 최대한 활용할 수 있어야 할 뿐 아니라, 가상화로 인한 계층적 스케줄링의 특성을 고려해야 한다. 본 논문에서는 가상머신 모니터의 스케줄러가 가지는 자원 모델을 바탕으로 자원 제공량의 하한을 구하고, 이를 기반으로 하여, 병렬성을 최대화 할 수 있는 실시간 스케줄 가능성 조건을 분석해 내었다. 이를 통하여, 멀티코어 가상머신 환경에서도 우선 순위 기반의 실시간 게스트 운영체제가 동작할 수 있음을 이론적으로 보였다.

참고문헌

[1] Baker, T. P. 2006. An Analysis of Fixed-Priority Schedulability on a Multiprocessor. Real-Time Syst. 32, 1-2 (Feb. 2006), 49-71. DOI= http://dx.doi.org/10.1007/S11241-005-4686-1

[2] Baruah, S. and Baker, T. 2008. Schedulability analysis of global EDF. Real-Time Syst. 38, 3 (Apr. 2008), 223-235. DOI= http://dx.doi.org/10.1007/s11241-007-9047-9

[3] Shin, I. and Lee, I. 2008. Compositional real-time scheduling framework with periodic model. ACM Trans. Embed. Comput. Syst. 7, 3 (Apr. 2008), 1-39. DOI= http://doi.acm.org/10.1145/1347375.1347383

[4] Easwaran, A., Shin, I., and Lee, I. 2009. Optimal virtual cluster-based multiprocessor scheduling. Real-Time Syst. 43, 1 (Sep. 2009), 25-59. DOI= http://dx.doi.org/10.1007/s11241-009-9073-x

[5] Anderson, J. H. and Srinivasan, A. 2000. Pfair scheduling: beyond periodic task systems. In Proceedings of the Seventh international Conference on Real-Time Systems and Applications (December 12 - 14, 2000). RTCSA. IEEE Computer Society, Washington, DC, 297.

[6] Barham, P., Dragovic, B., Fraser, K., Hand, S., Harris, T., Ho, A., Neugebauer, R., Pratt, I., and Warfield, A. 2003. Xen and the art of virtualization. In Proceedings of the Nineteenth ACM Symposium on Operating Systems Principles (Bolton Landing, NY, USA, October 19 - 22, 2003). SOSP '03. ACM, New York, NY, 164-177. DOI= http://doi.acm.org/10.1145/945445.945462

[7] Cherkasova, L., Gupta, D., and Vahdat, A. 2007. Comparison of the three CPU schedulers in Xen. SIGMETRICS Perform. Eval. Rev. 35, 2 (Sep. 2007), 42-51. DOI= http://doi.acm.org/10.1145/1330555.1330556

[8] Ongaro, D., Cox, A. L., and Rixner, S. 2008. Scheduling I/O in virtual machine monitors. In Proceedings of the Fourth ACM SIGPLAN/SIGOPS international Conference on Virtual Execution Environments (Seattle, WA, USA, March 05 - 07, 2008). VEE '08. ACM, New York, NY, 1-10. DOI= http://doi.acm.org/10.1145/1346256.1346258