연성 실시간 시스템에서 시스템 성능을 높이기 위한 수행시간 결정

정다니엘°, 이진호
포디홈네트 주식회사
{dennis, zino}@4dhome.net
TEL 02-517-1052, FAX 02-517-1054

홍성제
포항공과대학교
sjhong@postech.ac.kr
TEL 054-279-2250

Execution Time Determination for Performance Improvement in Soft Real-Time Systems

Daniel Chung°, Zino Lee
4DHomeNet, Inc.

Sung Je Hong
POSTECH

요 약

기존의 강성 실시간 시스템은 실시간 수행만 고려되어 있어, 실시간 시스템의 성능과 효율적인 시스템 활용과는 거리가 멀다. 따라서 효율적인 시스템 활용을 통하여 보다 많은 일을 수행할 수 있게 하는 연성 실시간 시스템을 사용하게 된다. 여기서는 보상값이라는 것을 사용하여 그것의 총합을 성능평가의 기준으로 사용하며, 보상값의 합을 최대로 할 수 있는 수행시간을 찾아 스케줄링에 사용하게끔 한다. 모의 실험을 통하여 알아낸 수행시간에 따른 보상값의 변화에 대한 경향을 이용하여 최대 보상값을 얻을 수 있는 수행시간을 찾고, 실험을 통하여 얻은 결과와 비교하였다.

1. 서론[1]

1.1 개요

최근 컴퓨터 및 정보통신의 발달로 실시간 음성 및 영상 전송, 휴대용 전화, 원격 제어 및 감시 등에서 실시간 시스템을 사용하는 일이 많아졌다. 여기서 사용하는 것은 대부분 강성 실시간 시스템(Hard Real-Time System)을 기반으로 하고 있다. [6, 7]

이 논문에서는 기존의 강성 실시간 시스템이 가지는 문제점을 보완하고자 연성(soft) 실시간 시스템을 사용한다. 강성 실시간 시스템에서는 실시간으로 일이 수행되는 데 초점이 맞추어져 있으므로 시스템 성능 및 자원 사용의 효율성은 고려되어있지 않다. 따라서 몇몇 일이 실패하더라도 실시간의 특성을 유지하면서 시스템 성능 향상 및 효율적인 시스템 자원 사용 을 고려하고자 한다. 이러한 특성을 만족시키기 위하여 연성 실시간 시스템의 개념을 도입하고자 한다.

1.2 기존 강성 실시간 시스템의 문제점

강성 실시간 시스템에서는 모든 일이 실시간의 특성을 만족해야 한다. 따라서 최악의 경우를 가정하여 스케줄 해야만 한다. 하지만, 대부분의 일은 이보다 빨리 수행을 끝낼 수 있다. 여기서 수행을 끝마치고 남는 시간만큼 시스템은 사용되지 않으므로 그만큼 시스템 자원을 낭비하게 되며, 남는 시간동안 더 많은 일을 수행하지 못함으로 인한 시스템 성능 저하의 문제도 있다. 이러한 문제점은 일의 수행시간의 변화가 클수록 두드러진다. [3]

1.3 연성 실시간 시스템에서의 스케줄링

연성 실시간 시스템에서는 몇몇 일이 실시간의 특성을 만족시키지 못하더라도 전체적으로 실시간의 특성을 유지하고 있으면 된다. 따라서 같은 시간동안 보다 많은 일을 수행하도록 함으로써 시스템 성능을 높일 수 있다. 그러기 위해서는 같은 시간동안 더 많은 일이 수행될 수 있도록 스케줄 해 주어야 하는데, 많은 일을 스케줄하기 위해서는 스케줄 할 때 각 일에 할당하는 시간을 줄여야 한다. 이 시간을 어떻게 조정해 주느냐에 따라 시스템의 성능이 크게 차이가 나는데, 너무 길게 잡으면 강성 실시간 시스템의 경우처럼 시스템 자원의 낭비와 성능 저하를 가져오며 너무 짧게 잡으면 많은 일이 실시간의 특성을 만족시킬 수 없게 되어 오히려 전체적 성능이 떨어진다[6].

[1] 본 논문은 대한민국 정보통신부 2000년 선도기술개발 제 4 차 사업의 일환인 인터넷 정보가전용 내장형 DBMS (과제번호 2000-S-169) 개발과제의 지원으로 작성된 것입니다.

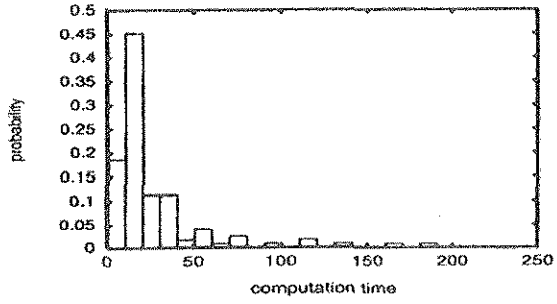
1.4 보상값

이 논문에서는 시스템 성능평가의 기준으로 보상값(reward)이라는 것을 사용한다. 기존의 연성 실시간 시스템에서는 전체에 일에 대한 실시간 실행을 성공적으로 수행한 일의 비율로 성능을 평가하는데, 이것으로는 일마다 가지는 특성을 반영할 수 없다. 따라서 일마다 성공적으로 수행했을 때 값을 부여하면 이러한 특성들을 반영할 수 있다. 여기서는 이것을 보상값이라고 정의한다. 보상값을 가지고 시스템을 평가하는 기준은, 시스템에서 수행되는 모든 일들이 얻은 보상값의 합이 된다.

1.5 수행시간에 따른 확률분포

일반적으로 실시간 시스템에서 수행되는 일은 시스템의 상황이나 프로그램의 특성에 따라 수행시간이 달라질 수 있다. 그러나 같은 일을 여러 번 수행하여 매번 수행시간에 대한 정보를 기록하여 정리하면, 수행시간에는 최소값과 최대값이 있으며[4, 5], 수행시간에 따른 분포가 일정한 경향을 보이는데 그에가 <그림 1>에 나타나 있다.

<그림 1> 수행시간에 따른 확률분포 [8]



2. 시스템 모델

2.1 일의 정의

이 시스템에서 돌아가는 프로그램을 가리켜 일(task)라고 한다. 하나의 일은 일정한 주기를 가지고 반복적으로 수행되며, 다음과 같이 표현할 수 있다. 시스템에서 수행될 일의 모임을 일 집합(set) 이라고 한다. 일은 다음과 같은 변수를 가지고 있다.

- 주기
- 마감시간
- 보상값
- 수행시간
- 수행시간에 다른 확률분포, 최소/최대 수행시간

이렇게 정의되는 일은 다음과 같은 특성을 갖는다.

- 주기적으로 반복된다. [9]
- 다른 일에 대하여 독립적으로 수행된다[7, 10]
- 가장 빠른 마감시간 우선(Earliest Deadline First, EDF) 알고리즘에 의해 스케줄되고, 수행된다.
- 이 시스템에서 수행될 일은 미리 정해져 있으며, 일 집합에서 나열된 순서대로 수행된다.
- 각 주기의 마감시간은 일이 들어온 때부터 다음 번 주기가 시작되기 전까지이다.
- 보상값은 각 주기마다 주어지는데, 마감시간 이내에 수행을 마칠 때 주어진다. 그렇지 않으면 보상값이 주어지지 않는다[1, 3, 8].

2.2 가장 빠른 마감시간 우선(EDF) 알고리즘

이 시스템에서 수행되는 일을 스케줄링 하는데 사용될 수 있는 방법에는 여러 가지가 있으나, 여기서는 EDF 알고리즘을 사용한다. 이 방법은 시스템 사용률이 1 이하이면 스케줄 가능함을 보장하며, 마감시간만 가지고 스케줄 하면 되므로 스케줄러 구현이 간단하다. 또한 수행시간이 확률적으로 분포할 때 최적이라고 알려져 있다[2, 7]. 이 알고리즘은 마감시간이 가장 빠른 일부터 스케줄한다. 그리고 수행하던 일보다 마감시간이 더 빠른 일이 오면 하던 일을 중단하고 마감시간이 빠른 것부터 우선 수행된다.

2.3 최대 보상값을 얻는 수행시간 정하기

보상값이 변하는 경향을 알아보기 위한 모의 실험에서 보상값의 합은 다음과 같은 경향으로 변함을 알 수 있었다.

- 시스템 사용률이 100% 미만일 때는 수행시간을 줄임에 따라 보상값의 합은 커진다.
- 시스템 사용률이 100%가 되기 시작하면, 과부하가 걸려 실시간 실행에 실패한 일이 늘어나게 되어 보상값의 합은 오히려 작아진다.
- 동시에 수행되는 일의 수가 같으면 보상값의 합은 거의 같다.
- 평균 수행시간을 수행시간으로 정했을 때에는 시스템 사용률이 100%보다 약간 작다.

위의 사실에 근거해서 수행시간을 찾는 법은 아래와 같다.

- 각 일에 대한 평균 수행시간을 구한다.
- 평균 수행시간을 가지고 일을 하나씩 추가해서 스케줄 해 보고, 스케줄할 수 없게 될 때 일의 수를 구한다. (EDF 알고리즘에서 스케줄 가능성 여부 테스트를 할 때 시스템 사용률이 1이 넘어가는 시점)
- 위의 일의 개수만큼 스케줄 가능한 수행시간의 최대값을 구한다.

3. 실험환경

3.1 시스템 환경

여기서 사용되는 시스템은 다음과 같은 특성을 가진다.

- 단일 프로세서 시스템
- 동시에 수행 가능한 최대 일의 수는 100 개
- 다루는 시간은 정수 단위 [9].
- 일 수행 도중에 고장나지 않음

3.2 일의 생성

시스템에 주어지는 일은 일 생성 프로그램을 사용하며, 일이 실제로 수행될 때 수행시간은 매번 수행할 때마다 달라지도록 난수로 주어진다. 이렇게 생성된 수행시간은 스케줄러는 알지 못하며, 실제로 수행될 때 언제 수행이 끝나는가를 알려주는 데에만 쓰인다.

3.3 확률분포

일의 성격에 따라 수행시간에 따른 확률분포가 틀려지는데, 아래와 같은 분포를 사용하였다.

- 균등분포

- 삼각형 분포(대칭/비대칭)
- 종 모양 분포
- 지수 분포
- $y = x e^x$ 에 기초한 분포

지수분포와 $y = x e^x$ 에 기초한 분포의 경우는 범위를 정하여 일부만 사용하였다.

4. 실험 결과

여기서는 3.3에 제시된 확률분포에 따라 일이 수행되었을 때, 최대 보상값을 가질 때 수행시간 및 수행되는 일의 수를 비교해 본다.

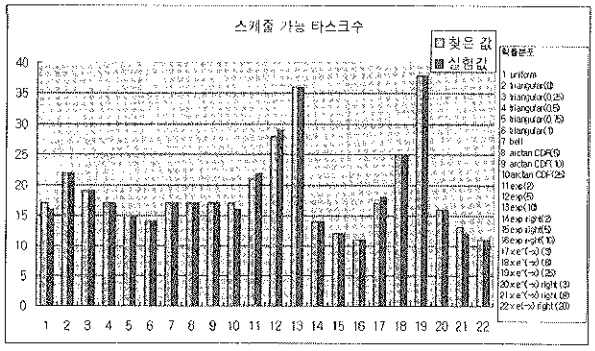
4.1 수행시간

2.3의 방법으로 찾은 수행시간과 실제 최대 보상값을 가질 때 수행시간은 <그림 2>에서와 같이 큰 차이가 없음을 알 수 있다. 최대 오차는 9.92%이다.

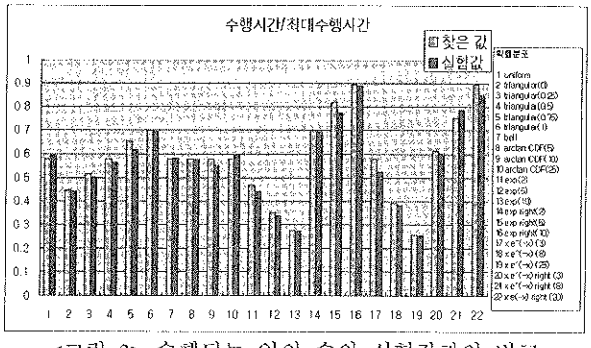
4.2 수행되는 일의 수

대부분의 일이 실시간의 특성을 만족할 때, 보상값의 합이 크기 위해서는 동시에 수행되는 일의 수가 많을수록 좋다. 따라서 여기서는, 수행되는 일의 수를 비교해 보고자 한다. <그림 3> 에서와 같이 대부분의 경우 일치하는 것을 알 수 있다.

이와 같이 찾아낸 수행시간과 수행되는 일의 수는 어떤 확률 분포에서든지 실험값과 별 차이가 없으므로, 2.3 에서 서술한 최대 보상값을 얻는 수행시간 찾는 법은 어떤 확률분포에든지 적용 가능함을 알 수 있다.



<그림 2> 수행시간의 실험값과의 비교



<그림 3> 수행되는 일의 수의 실험값과의 비교

5. 결론

이 논문에서는 보상값이 주어지는 연성 실시간 시스템에서 최대 보상값을 얻는 수행시간을 구하는 방법을 제시하였다. 이 방법은 최대 보상값을 얻는 시점이 시스템에 과부하가 걸리기 시작할 때, 이로 인한 일의 성공률 저하와 수행하는 일의 수가 증가함으로 인하여 증가되는 보상값의 영향이 상쇄될 때임을 이용하였다. 이 알고리즘으로 찾은 수행시간과 실험을 통해 수행시켰을 때 얻은 수행시간을 비교해 보면, 9.92% 이하의 오차로 정확하게 찾았음을 알 수 있다. 여기서 사용한 EDF 알고리즘은 시스템에 과부하가 걸릴 때 일이 수행하다가 마감시간을 만나서 실패하는 경우가 많아지는 문제점을 가지고 있으므로 이것을 개선해서 일이 수행하다가 실패하는 경우를 최소화할 수 있는 스케줄링 알고리즘을 사용할 필요가 있다.

6. 참고문헌

[1] A. K Alas and A. Bestavros, *Statistical Rate Monotonic Scheduling*. Technical Report BUCS-TR-98-010, Boston University, 1998.

[2] H. Chetto and M. Chetto, *Some Results of the Earliest Deadline Scheduling Algorithm*, IEEE Transactions on Software Engineering, Vol. 15, No. 10, pp. 1261-1269, October 1989.

[3] M. K. Gardner and J. W. S. Liu, *Analyzing Stochastic Fixed-Priority Real-Time Systems*. In Proceedings of the Fifth International Conference on Tools and Algorithms for the Construction and Analysis of Systems, Vol. 1579, pp. 44-58, 1999.

[4] C. A. Healy, R. D. Arnold, F. Meuller, D. B. Whalley, and M. G. Harmon, *Bounding Pipeline and Instruction Cache Performance*. IEEE Transactions on Computer, Vol. 48, No. 1, pp. 53-70, January 1999.

[5] C. A. Healy, M. Sjödin, V. Rustagi, and D. Whalley *Bounding Loop Iterations for Timing Analysis*, IEEE Transactions on Computers, Vol. 48, No. 1, pp. 53-70 January 1999.

[6] B. C. M. Kao, *Scheduling In Distributed Soft Real-Time Systems with Autonomous Components*, Ph. D. Thesis, Department of Computer Science, Princeton University, 1995

[7] J. A. Stankovic, M. Spuri, M. D. Natale, and G. C. Buttazzo, *Implications of Classical Scheduling Results for Real-Time Systems*, IEEE Computer, pp. 16-25, June 1995.

[8] T. S. Tia, Z. Deng, M. Shankar, M. Storch, J. Sun, L. C. Wu, and J. W. S Liu, *Probabilistic Performance Guarantee for Real-Time Tasks with Varing Computation Times*. In Proceedings of the IEEE Real-Time Technology and Applications Symposium, pp. 164-173, 1995.

[9] J. Xu and D. L. Parnas, *On Satisfying Timing Constraints in Hard Real-Time Systems*, Proceedings of the ACM SIGSOFT '91 Conference on Software for Critical Systems, pp. 132 – 146, 1991.

[10] J. Xu and D. L. Parnas, *Scheduling Processes with Realease Times, Deadlines, Precedence, and Exclusion Relations*, IEEE Transactions on Software Engineering, Vol. 16, No. 3, pp. 360-369, March 1990.