UNIX의 Decay Usage 알고리즘에서의 지연시간-사용량 정규화 특성 분석

(Analysis of Delay-Bandwidth Normalization Characteristic in Decay Usage Algorithm of UNIX)

박 경 호 † 황 호 영 †† 이 창 건 ††† 민 상 렬 ††††
(Kyeongho Park) (Hoyoung Hwang) (Changgun Lee) (Sanglyul Min)

요 약 Decay usage 알고리즘은 CPU를 최근에 적게 사용한 프로세스를 우선시하는 스케줄링 방법으로, UNIX와 같이 계산 위주의 프로세스와 대화형 프로세스가 혼재한 시분할 시스템에서 널리 사용되어 왔다. 하지만, decay usage의 매개변수들이 어떻게 상호작용하며 결국 어떤 서비스 행태를 보이는 지에 대한 분명한 이해가 없었다. 본 논문에서는 decay usage가 사용하는 매개변수들의 상호작용에 따라 서비스의 사용량 및 지연시간이 보이는 행태를 실험적으로 분석한다. 이러한 실험적 분석을 바탕으로, 각 매개변수가 가지는 의미를 서비스 제공의 관점에서 규명한다. 본 논문의 분석 결과는 decay usage의 매개변수들을 조정하여 응용의 요구에 맞는 서비스를 제공하기 위한 기반이 된다.

키워드 : decay usage, CPU 스케줄러, 공정 스케줄링, 지연시간-사용량 정규화, 과거정보 민감도

Abstract Decay usage scheduling algorithm gives preference to processes that have consumed little CPU in the recent past. It has been widely-used in time-sharing systems such as UNIX, where CPU-intensive processes and interactive processes are mixed. However, there has been no sound understanding about the mixed effects of decay usage parameters on the service performance. This paper empirically analyzes their effects in terms of the resulting service bandwidth and delay. Based on such empirical analysis, we derive the clear meaning of each parameter. Such analysis and understanding provides a basis of controlling decay usage parameters for desirable service provision as required by applications.

Key words : decay usage, CPU scheduler, fair scheduling, delay-bandwidth normalization, history-sensitiveness

1. 서 론

Decay usage 알고리즘은 전통적으로 UNIX 등의 시분할 운영체제에서의 CPU 스케줄러로 널리 사용되어 왔다[1-6]. 그러나 decay usage에 내재된 특성에 대해서는 깊이 있는 분석이 부족했다. 본 논문에서는 decay usage가 각 프로세스의 사용량 정보를 기억하여 이를 추후의 스케줄링에 반영함으로써 서비스 지연시간에 영향을 미치는 특성을 시뮬레이션을 통해 실험적으로 분석하고, 각 매개변수들의 의미를 규명한다.

Decay usage 알고리즘은 각 프로세스에게 주어지는 고정된 값인 기본우선순위(base priority)와 동적으로 변하는 CPU 사용량 정보를 조합하여 우선순위를 표현한다. 이 때, CPU 사용량 정보를 주기적으로 감쇄시켜, 최근 CPU 사용량 정보가 최종우선순위 결정에 보다 많이 반영되도록 한다. 따라서 최근에 서비스를 적게 받은 프로세스가 우선적으로 CPU를 획득하게 되어 서비스의 공정성을 제공하게 된다. 이로 인해, CPU를 자주 사용하지 않는 I/O 위주의 프로세스는 CPU 작업을 빨리 끝마치고 I/O 작업을 계속할 수 있으므로 CPU 작업과

· 본 연구는 '정보통신부 및 정보통신연구진흥원의 IT신성장동력핵심기술개발사업[2006-S-040-01, Flash Memory 기반 임베디드 멀티미디어 소프트웨어 기술 개발]'과 '2007년도 한성대학교 교내연구비' 및 '서울대학교 신임교수 연구정착금'의 지원을 받아 수행되었습니다.

† 학생회원 : 서울대학교 컴퓨터공학부
kalynda@archi.snu.ac.kr
†† 종신회원 : 한성대학교 멀티미디어공학과 교수
hyhwang@hansung.ac.kr
††† 정 회 원 : 서울대학교 컴퓨터공학부 교수
cglee@cse.snu.ac.kr
†††† 종신회원 : 서울대학교 컴퓨터공학부 교수
symin@dandelion.snu.ac.kr

논문접수 : 2007년 3월 16일
심사완료 : 2007년 7월 4일

I/O 작업의 병렬수행 가능성을 높여서 시스템의 전체 사용률이 높아지게 된다. 이는 또한 대화형 프로세스의 응답 시간을 단축하는 효과를 낸다. 즉, decay usage는 계산 위주의 프로세스와 대화형 프로세스가 혼재한 상황에서 대화형 프로세스의 응답 시간을 줄이고 시스템의 사용률을 높이는 특성을 내포하고 있다.

그러나 decay usage 알고리즘을 기술하는 데에 사용되는 시스템 매개변수들은 서비스율이나 지연시간 등의 서비스 품질을 직접적으로 표현하는 것이 아니어서, 성능에 대한 예측을 하거나 매개변수들을 조절해 원하는 서비스를 제공하도록 하기가 어렵다. 과거에 decay usage의 특성을 분석한 몇몇 연구들[7-9]이 있었으나, 이들은 모든 프로세스들이 CPU 작업만을 수행한다는 가정하에 각 프로세스들에 주어지는 CPU 시간의 총량(즉, **서비스 사용량**)만을 주된 관심의 대상으로 삼고 있기 때문에, 다양한 특성의 프로세스들이 혼재하는 환경에서의 decay usage에 내재된 특성을 밝히는 데 한계가 있었다.

본 논문에서는 기존 연구들과 달리 서비스 사용량뿐만 아니라 서비스 지연시간도 프로세스가 받은 서비스를 평가하는 데에 있어서 중요한 요소임을 지적하고, 다양한 CPU 작업부하를 내는 프로세스들이 혼재하는 상황에서 decay usage 알고리즘이 제공하는 서비스의 특성을 "서비스 사용량"과 "서비스 지연시간"을 포괄한 관점에서 이해한다.

Decay usage가 과거의 CPU 사용량 정보를 기억하였다가 사용량이 적은 프로세스를 우선적으로 서비스하여 지연시간을 상대적으로 단축하여 주는 것은, 서비스된 사용량과 서비스 지연시간 사이에 절충(trade-off) 관계가 있음을 암시하며, 본 논문에서는 이 관계를 "지연시간-사용량 정규화"로 표현한다. 본 논문에서는 다음의 특성을 실험적으로 규명한다.

- 기본우선순위가 동일한 프로세스들 간의 지연시간-사용량 정규화 특성: 기본우선순위가 동일한 프로세스들의 그룹을 관찰할 때, 서비스 사용량을 많이 제공 받는 프로세스는, 그렇지 않은 프로세스에 비해, 서비스 지연시간 특성이 나빠진다.
- 기본우선순위에 따른 지연시간-사용량 정규화 정도의 스펙트럼: 다양한 기본우선순위를 가지는 프로세스들이 혼재해 있을 때, 각 그룹의 기본우선순위의 상대적 크기에 따라 지연시간-사용량 정규화 정도가 달라지며 그 정규화 정도가 연속적인 스펙트럼을 형성한다.
- 사용량 정보 감쇄비율에 따른 지연시간-사용량 정규화 정도의 스펙트럼: Decay usage가 프로세스의 우선순위를 결정할 때 과거 CPU 사용량 정보를 얼마나 감쇄시키느냐에 따라 지연시간-사용량 정규화 정도가 달라지며 그 정규화 정도가 또 다른 차원의 연속적인 스펙트럼을 형성한다.

이를 바탕으로, 본 논문은 기본우선순위 및 사용량 정보 감쇄비율을 결정하는 각 매개변수가 지연시간-사용량 관점의 서비스 특성에 미치는 영향에 대한 개념적 이해를 도출한다.

본 연구를 통해 decay usage에 내재되어 있으나 잘 드러나 보이지 않던 특성에 대한 정성적 측면의 이해를 높일 수 있다. 또한 이러한 관찰을 종합하면, 어떤 프로세스가 앞서 언급한 두 스펙트럼상의 어느 부분에 위치하는가에 따라 그 프로세스에 대해 예측할 수 있다. 그리고 이는 decay usage의 특성을 반영하는 분석가능한 모델을 정립하고, 매개변수를 조정하여 응용의 요구에 맞는 서비스를 제공하기 위한 기반이 된다.

이 논문의 이후의 구성은 다음과 같다. 2절에서는 decay usage 알고리즘의 기본적인 동작 원리를 설명하고 관련연구를 정리한다. 3절에서는 decay usage에 내재한 지연시간-사용량 정규화 특성을 규명한다. 4절에서는 지연시간-사용량 정규화 정도가 기본우선순위 및 과거사용량정보 감쇄비율에 따라 두 가지 연속적인 스펙트럼을 형성함을 실험을 통해 보인다. 또한 시스템 매개변수들의 변화에 의해 나타나는 특성들을 분석한다. 5절에서는 이러한 관찰결과의 응용가능성에 대해 토의하고, 끝으로 6절에서 결론을 맺는다.

2. Decay usage 알고리즘 개요 및 관련연구

2.1 Decay usage 알고리즘의 개요

Decay usage 알고리즘은 프로세스들의 우선순위를 동적으로 변화시키면서 스케줄링에 이용하는데, 우선순위는 다음의 두 가지 요소를 조합하여 결정된다.

- 각 프로세스 i 에게 정적으로 주어지는 기본우선순위 ($base_i$)
- 각 프로세스 i 의 과거 CPU 사용량 정보 (cpu_i)

그림 1은 decay usage 알고리즘이 어떻게 위의 두 가지 요소를 조합하여 프로세스 i 의 우선순위 (pri_i)를 결정하는지를 보인다[7]. 실제 구현상의 알고리즘들은 약간의 변형이 있을 수 있으나[1-6], 기본적으로는 그림 1의 알고리즘에 기반하고 있다. 각 프로세스 i 는 고정된 기본우선순위 $base_i$ 를 외부에서 부여받으며, 이 값

- 프로세스 i 가 한 퀀텀을 소비할 때마다:
  $cpu_i = cpu_i + 1$
- 매 T 퀀텀마다 모든 프로세스에 대해:
  $cpu_i = cpu_i / D \quad (D > 1)$
- 스케줄링시 다음 값이 최소인 프로세스 i 를 선택:
  $pri_i = R * cpu_i + base_i$

그림 1 decay usage 기본 알고리즘

은 프로세스들이 시스템 내에서 가지는 상대적인 서비스 획득 권한에 해당한다. $base_i$ 가 작을수록 높은 권한을 나타낸다. cpu_i 는 프로세스 i 의 CPU 사용량 정보를 담고 있는 값으로서 초기값은 0이며, 프로세스 i 가 한 퀀텀을 소비할 때마다 증가하여 CPU 사용량 정보를 누적하게 된다. 모든 프로세스들의 이 cpu_i 값은 매 T 퀀텀마다 $1/D$ 의 비율로 감쇄된다. 이러한 주기적 감쇄는 최근의 CPU 사용량 정보에 비해 과거의 정보의 영향을 줄이기 위한 것이다. 이렇게 결정된 cpu_i 와 정적으로 주어진 $base_i$ 를 그림 1의 마지막 수식에 의해 조합하여 프로세스의 동적 우선순위인 pri_i 를 결정한다. 여기서 R 은 CPU 사용량 정보 cpu_i 를 우선순위 pri_i 에 반영하는 정도를 결정하는 가중치이다. 매 퀀텀에 이렇게 결정된 우선순위에 따라 스케줄러는 pri_i 가 가장 작은 프로세스를 선택하여 CPU를 할당한다. 직관적으로, $base_i$ 값이 작을수록, 또 cpu_i 값이 작을수록 프로세스는 높은 우선순위로 서비스된다.

이러한 스케줄링을 위한 decay usage의 매개변수는 프로세스당 주어지는 개별 변수 $base_i$ 와 시스템 변수 R, T, D 로 요약될 수 있는데, 이 매개변수들의 값은 응용의 요구에 따라 시스템 설계자가 적절히 선택하여야 한다.

그림 2에 세 개의 프로세스 A, B, C 를 decay usage 알고리즘으로 스케줄링하는 예를 보였다. 각 프로세스의 기본우선순위는 0, 4, 4이고, 시스템 매개변수는 D = 2, T = 20, R = 1로 가정하였다. 그림에서 위쪽의 사각형들은 각 프로세스의 요청들이 도착한 시점과 그 길이(요구되는 CPU시간)를 나타낸다. 프로세스 A 와 B 는 CPU 요청을 많이 내는 작업인 반면에, 프로세스 C 는 간헐적으로 CPU 요청을 내는 대화형 작업의 특성을 보이고 있다. 그림의 중간 부분에는 각 프로세스의 우선순위값 pri_i 의 변화와 매 퀀텀에 스케줄러에 의해 선택되어 실행되는 프로세스를 표시하였다. 각 프로세스의 cpu_i 값은 서비스를 받을 때마다 1씩 증가하고 매 T 가 경과한 시점마다 감쇄가 이루어지므로, pri_i 도 이에 따라 증감한다. 그림의 아래쪽에는 각 요청이 서비스를 마치기까지 시스템 안에서 보내는 구간을 화살표로 표시하였다. 이 구간의 길이가 해당 요청에 대한 서비스 지연시간이다. 서비스 지연시간이 요청의 원래 길이에 가까울수록 그 요청은 신속한 서비스를 받는 것으로 해석할 수 있다.

2.2 관련연구

Decay usage의 특성을 분석한 기존의 연구는 그리 많지 않은 편이다. Hellerstein의 연구[7]에서는 CPU 위주의 프로세스들이 경쟁할 때 안정상태(steady-state)에서 각 프로세스에 제공되는 서비스 사용량을 분석하기 위한 모델을 제시하였으며, 그 사용량을 제어하기 위한 알고리즘을 만들었다. Epema의 연구의 경우[8,9], 다중프로세서 시스템에 decay usage 알고리즘을 적용하였을 때의 서비스 사용량에 대하여 유사한 분석을 하였다. 그러나 이러한 연구들은 모든 프로세스들이 CPU 작업만을 수행한다는 가정을 하여 다양한 성격의 프로세스들이 혼재하는 일반적인 시스템의 특성을 반영하지 못하고 있다. 또한, 서비스 사용량만을 주된 관심의 대상으로 삼고 있기 때문에, 서비스를 적게 받은 프로세스의 우선순위를 높여 지연시간의 관점에서 보상을 해주는 decay usage의 중요한 특성을 간과하고 있다.

Petrou[10]는 lottery 스케줄링[11]을 구현하는 연구에

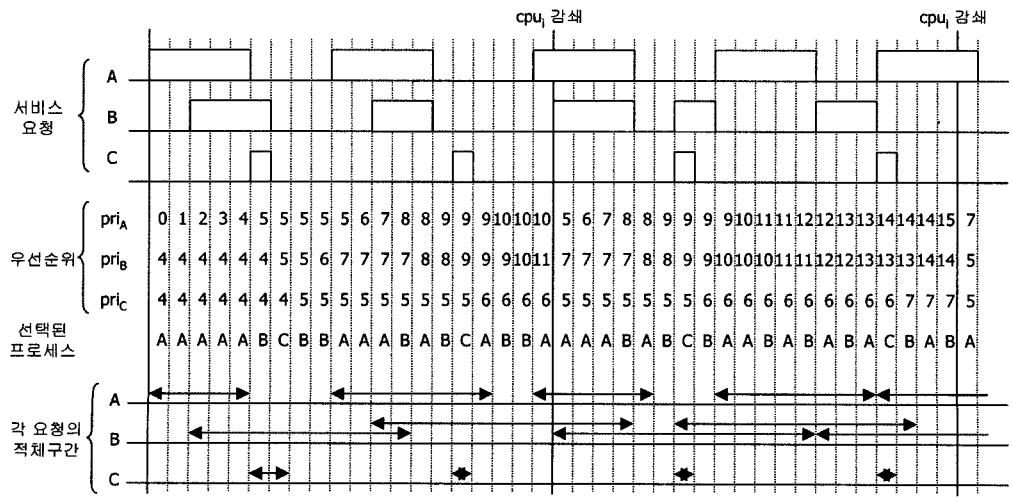


그림 2 decay usage 알고리즘의 동작 예

서 대화형 응용을 우대하는 decay usage의 특성에 대하여 언급한 바 있다. 그러나 decay usage가 우선순위 기반의 스케줄링을 하며 이러한 우선순위가 감쇄동작 등으로 인해 수시로 변화하므로, 큐잉 이론 등의 방법론을 통한 분석이 쉽지 않아 이 특성을 규명하기 위한 추가적인 연구는 거의 없는 실정이다.

한편 이 논문의 분석대상인 decay usage 이외에도 사용량과 지연시간을 함께 고려한 스케줄링 기법들이 있는데, 대표적인 것이 서비스 커브(service curve)이다 [12,13]. 이 방법에서는 개별적 응용이 원하는 서비스의 특성을 서비스 커브의 형태로 지정하면 이를 보장해 주는데, 별도의 수락 제어(admission control)가 필요하고 각 응용에 대해 서비스 커브를 개별적으로 명시해야 하며 응용의 작업부하 특성이 변하지 않고 유지될 때에만 의미가 있다. 이에 비해 decay usage는 서비스 커브와 같은 세밀한 제어를 허용하지는 않지만 단순한 알고리즘만으로 지연시간-사용량 정규화를 실현하고 있으며 응용의 작업 부하 특성을 제한하지 않는다는 데에 그 차별성이 있다. 또한 decay usage와는 별개로 지연시간-사용량 정규화 개념을 내포한 스케줄링 기법에 대한 연구[14]와 기존의 공정 스케줄러들을 포괄하는 과거정보 민감도 기반의 스케줄러에 대한 연구[15] 등도 최근 이루어지고 있다. 하지만, decay usage와 같은 단순화된 스케줄링 틀을 제공하지 못해 실제 구현상에 문제가 있다.

3. 지연시간-사용량 정규화 특성

전술한 바와 같이 decay usage의 특성에 대한 기존의 연구들에서 주요한 분석 대상으로 삼은 것은 각 프로세스에게 주어지는 서비스 사용량이다. 그러나 프로세스가 받은 서비스의 품질을 평가하는 데에 있어서 또 하나의 중요한 요소는 서비스 요청에 대한 지연시간이다. 특히 대화형 프로세스에서는 서비스 요청에 대한 지연시간이 서비스 품질(QoS, quality of service)의 중요한 판단 기준이 된다.

그런데 서비스 요청의 길이가 다양하게 변화하는 경우에는, 서비스 가능한 최소의 지연시간(즉 요청의 길이)에 비해 실제 서비스 받은 지연시간이 어느 정도인지를 따지는 것이 타당하다. 본 논문에서는, 프로세스 i의 j 번째 요청 r_i^j 의 길이를 D_i^j, 서비스에 걸린 지연시간을 R_i^j 라고 할 때, r_i^j 가 겪은 지연시간의 품질을

$$q_i^j = D_i^j / R_i^j \quad (1)$$

로 정의한다. 즉, 최상의 서비스를 받은 경우, 실제 서비스 지연시간은 요청의 길이와 동일하므로 q_i^j 는 1이 되며 지연시간이 늘어날수록 q_i^j 는 0에 가까워진다. 따라서 $0 < q_i^j <= 1$이다. 그리고 관찰 구간 내에서 프로세스 i 가 낸 모든 서비스 요청들 n 개에 대해 q_i^j 의 산술평균을 지연시간 관점의 서비스 품질 $DQoS_i$ 로 표시한다.

$$DQoS_i = \Sigma_{j=1..n}\ q_i^j / n \quad (2)$$

이후의 모든 실험에서 프로세스가 받은 서비스는 관찰 구간 내에서의 받은 서비스의 양 W_i 와 서비스 지연시간의 품질 $DQoS_i$ 로 표현한다.

Decay usage에서는 서비스를 많이 받은 프로세스의 우선순위를 낮춤으로써 서비스를 상대적으로 적게 받은 프로세스의 서비스 요청이 우선적으로 처리될 수 있도록 하므로, 이를 통해 각 프로세스가 받은 서비스의 양과 서비스 품질 사이에는 절충 관계가 성립된다. 즉 많은 양을 사용할수록 cpu_i 의 값이 증가하므로 추후의 스케줄링에서 상대적으로 불리하게 되어 지연시간 측면에서의 불이익을 받게 되며, 적은 양을 사용한 프로세스는 반대로 지연시간의 상대적 이득을 보게 된다. 각 프로세스의 W_i 와 $DQoS_i$ 의 관찰을 통해 이를 확인할 수 있다.

첫 번째 실험은 이러한 사용량과 지연시간 사이의 기본적인 상관관계를 관찰하기 위한 것이다. 일단 pri_i 에 영향을 주는 cpu_i 와 $base_i$ 의 두 가지 요소 중 사용량 정보와 관련있는 cpu_i 의 영향만을 우선적으로 관찰하기 위하여, 이 실험에서는 모든 프로세스들에 동일하게 $base_i$ 를 0으로 부여한다. 또한 cpu_i 를 T 퀀텀마다 D로 나누는 감쇄 동작은 과거사용량 정보를 희석시키는 역할을 하기 때문에, 이 영향을 배제하기 위해 D = 1로 설정하여 감쇄가 일어나지 않도록 한다. 그림 3은 1,000,000 단위시간 동안 동일한 $base_i$ = 0을 가지는 프로세스들 100개가 생성하는 임의의 작업부하들을 스케줄링하여 관찰된 각 프로세스 i 의 (W_i, $DQoS_i$) 쌍을 도시한 것이다. 이 때 CPU 사용률 U 를 0.25, 0.5, 0.75, 1.0으로 변화를 주었으며, T = 1000, R = 0.5로 설정하였다.

각 프로세스들이 생성하는 서비스 요청의 모델링시, 각 요청의 길이가 상호독립적이라고 가정하여 정규분포 N (μ_i, σ_i^2)를 따르는 것으로 모델링한다. 또 요청의 도착시간 간격은 지수분포 Exp(λ_i)를 따르는 것으로 가정한다. 여기에서 매개변수 μ_i, σ_i 는 각각 균등분포 U (0, 10), U (0, 2)에서 임의로 추출한다. 또한 각 프로세스의 작업부하를 u_i 라고 할 때 u_i 는 균등분포 U(0.01, 0.20)에서 추출한 후 $\Sigma u_i = U$ 가 되도록 정규화하고, 이를 기반으로 $\mu_i = \lambda_i u_i$ 가 되도록 λ_i 를 설정한다. 이후의 실험에서도 서비스 요청 트래픽의 생성은 동일한 방법을 따른다.

그림 3의 결과를 통해 동일한 기본우선순위를 가지는 프로세스들에서 W_i 와 $DQoS_i$ 의 사이에는 기본적으로

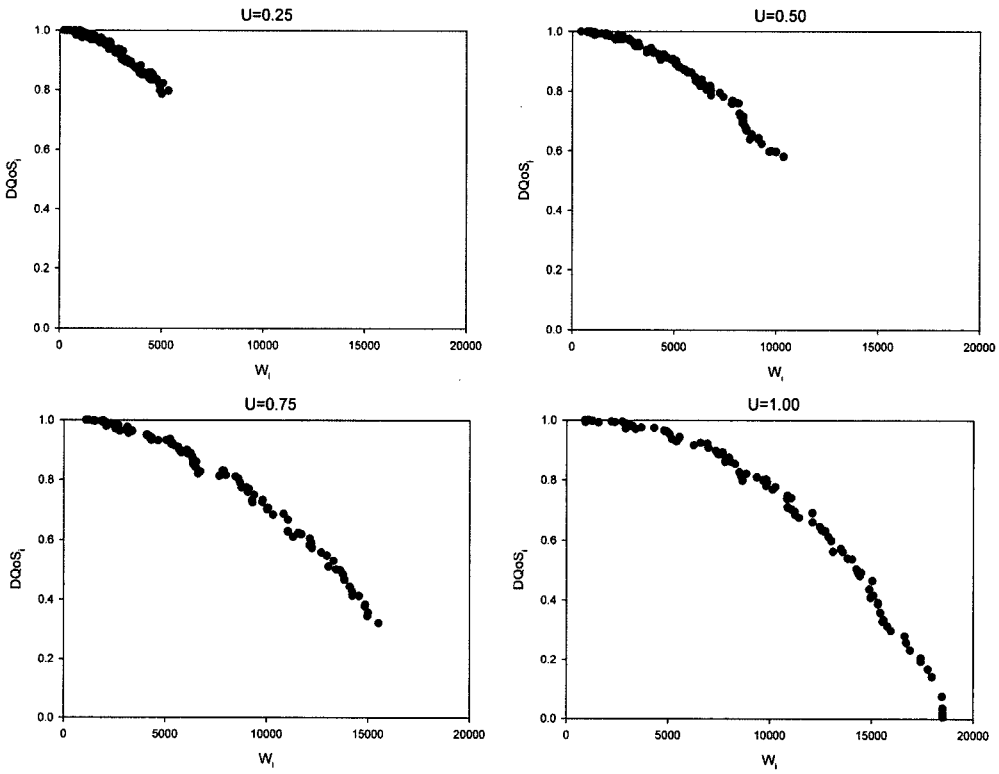


그림 3 동일한 우선순위를 가지는 프로세스들에서의 지연시간-사용량 정규화 특성

서로 절충하는 지연시간-사용량 정규화 관계가 존재함을 확인할 수 있다. 즉, 서비스 받는 양이 많은 프로세스는 지연시간의 관점에서는 나쁜 품질의 서비스를 받게 되어 (사용량, 지연시간) 쌍의 관점에서 정규화가 실현된다. 각 그래프상에 도시된 점들을 연장하는 선은 decay usage에서 동일한 수준의 서비스로 간주되는 $(W_i, DQoS_i)$ 쌍의 궤적이다. 이 정규화 특성은 전체 시스템 사용률 U 와 무관하게 유효하다. 그림에서 U 가 1보다 작을 때 점들이 왼쪽 윗부분에 치우쳐 관찰되는데, 이는 프로세스들이 그만큼 적은 작업부하를 생성하기 때문이다. 또한, decay usage 알고리즘에서 기본우선순위는 상대적인 의미를 지니는 값이므로 $base_i$ 를 0이 아닌 다른 값으로 설정하더라도 이 정규화 관계는 그대로 유지된다.

이 실험을 통해 규명된 특성은 다음과 같이 요약된다.

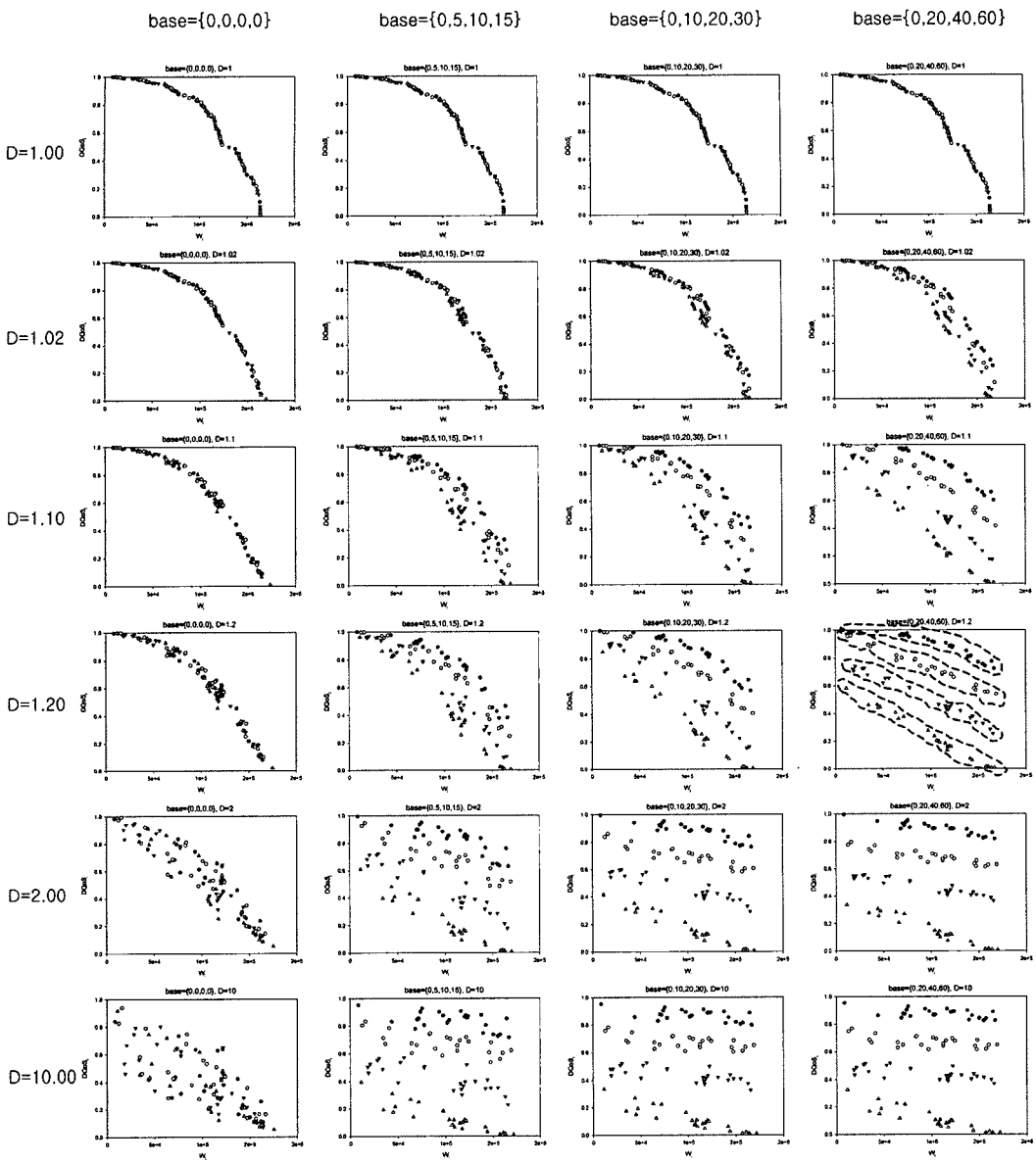
특성 1: 동일한 기본우선순위를 가지는 프로세스들 사이에는 지연시간-사용량 정규화 관계가 존재한다.

이러한 특성은 decay usage의 매개변수 값의 모든 범위에서 관찰되지만, 사용량 증가에 따른 지연시간 품질 저하의 정도, 즉 "지연시간-사용량 정규화 정도"는 매개변수 값에 따라 달라진다. 다음 절에서는 매개변수 값들이 정규화 정도에 미치는 영향을 규명한다.

4. 매개변수에 따른 지연시간-사용량 정규화 정도의 스펙트럼

앞의 그림 3의 실험을 통해 동일한 기본우선순위를 지니는 프로세스들 사이의 지연시간-사용량 정규화 특성을 관찰할 수 있었으나, 실제 스케줄링에서는 기본우선순위가 서로 다른 프로세스들이 혼재해 있다. 이런 상황에서는, 프로세스들의 지연시간-사용량 정규화 정도는 decay usage 알고리즘이 프로세스의 우선순위를 결정하는 두 가지 요소, 즉 (1) "기본우선순위 값의 차이" 및 (2) "감쇄를 고려한 CPU 사용량 정보"에 의해 영향을 받는다. 이 절에서는 그러한 영향을 실험을 통해 규명한다.

그림 4는 기본우선순위의 차이와 사용량 정보의 감쇄 정도를 각각 변화시키면서 $(W_i, DQoS_i)$ 쌍을 도시한 것이다. 이 실험에서는 100개의 프로세스를 4개의 그룹으로 나누고, 각 그룹에 속한 25개의 프로세스에 동일한 기본우선순위를 부여한다. 그리고 각 그룹의 기본우선순위들 사이의 간격을 변화시키면서 (그림에서 오른쪽 방향) 프로세스들이 받는 서비스의 특성을 관찰한다. 그림에서 각 열 상단에 base=$\{b_1, b_2, b_3, b_4\}$의 형태로 표시



●: 프로세스 그룹 1, ○: 프로세스 그룹 2, ▼: 프로세스 그룹 3, △: 프로세스 그룹 4

그림 4 기본우선순위에 기반한 지연시간-사용량 정규화 스펙트럼 및 과거서비스 정보 민감도에 따른 스펙트럼의 관찰

한 것은 각 그룹의 프로세스들의 기본우선순위가 b_1, b_2, b_3, b_4 임을 의미한다. 또한 D 를 증가시켜 사용량 정보 감쇄비율의 변화(그림에서 아래쪽 방향)에 따른 추이를 관찰한다. 이 실험에서 CPU 사용률은 모두 1이 되도록 하였으며, 그 밖의 다른 조건들은 앞의 실험과 동일하다.

4.1 기본우선순위에 따른 지연시간-사용량 정규화 정도

우선 CPU 사용량 정보 감쇄비율을 고정하고, 프로세스 그룹 간 기본우선순위의 차를 증가시킬 때의 영향을 규명하기 위해 D = 1.20인 행을 관찰하자. 그룹간 기본우선순위의 차이가 전혀 없을 때에는 (맨 왼쪽 그림), 그룹간의 차이가 없이 모든 프로세스 그룹이 앞에서 관찰한 것과 비슷한 정도의 지연시간-사용량 정규화 정도를 보인다. 하지만, 프로세스 그룹간의 기본우선순위 차이를 점차적으로 벌려감에 따라 각 그룹 간 (사용량, 지연시간) 쌍의 궤적이 점차 벌어진다. 이 결과는 다음과 같이 이해될 수 있다. 우선, 서로 다른 그룹들의 기본우

선순위 차이가 크지 않을 때는 그 차이가 프로세스의 우선순위 pri_i 의 차이에 미치는 영향이 적기 때문에 주로 과거 CPU 사용량 정보가 pri_i 의 차이를 결정한다. 따라서 3절에서 관찰한 바와 같이, 과거 CPU 사용량 정보를 반영함으로써 나타나는 decay-usage의 지연시간-사용량 정규화 특성이 모든 그룹에 비슷하게 나타난다. 그러나 기본우선순위 차이가 증가하면 기본우선순위 차이가 pri_i 차이를 결정하는 주된 요소가 되고, 과거 CPU 사용량 정보의 영향은 점차 줄어든다. 따라서 과거 CPU 사용량 정보를 반영함으로써 나타나는 지연시간-사용량 정규화 특성이 점차 희미해져 서비스 받은 양이 많아도 지연시간품질에서 손해를 보는 정도가 완만해지는 현상이 나타난다. 이런 현상은 기본우선순위가 높은 (즉, 기본우선순위 값이 작은) 그룹에서 먼저 나타나고, 기본우선순위 차이를 키워감에 따라 낮은 기본우선순위 그룹으로 확산된다. 결과적으로 기본우선순위에 의해 주도되는 그룹 간 서비스 차이가 점차 뚜렷이 관찰된다. 즉, 기본우선순위가 작은 그룹은 같은 사용량에 대해 더 나은 품질의 지연시간을 제공받아 그래프 상에서 상대적으로 위쪽에 위치하게 된다. 이는 사용량과 지연시간을 포괄하여 고려한 서비스 품질이 기본우선순위가 커짐에 따라 점차 저하되는 것으로 이해될 수 있다. 이는 D = 1.20 행의 맨 오른쪽 그래프에서처럼 각 그룹을 점선으로 묶어 표시하면 더 명확하게 관찰된다. 이러한 특성은 다음과 같이 요약된다.

특성 2: 프로세스 기본우선순위의 상대적 크기에 따라 지연시간-사용량 정규화 정도가 달라지고 그 정규화 정도의 변이가 연속적인 스펙트럼을 형성한다.

기본우선순위 차이에 따른 정규화 정도의 스펙트럼은 과거 CPU 사용량 정보 감쇄비율이 큰 경우 (그림 4에서는 D 가 큰 경우) 더 분명하게 나타난다. 이는 감쇄비율이 크면 기본우선순위가 프로세스의 우선순위에 미치는 상대적 영향이 커져 기본우선순위 크기에 따른 정규화 정도의 차이가 분명히 드러나기 때문이다. 한 극단의 예로, D = 1.0인 행을 관찰하면 그룹 간 기본우선순위 차이를 늘려도 각 그룹의 지연시간-사용량 정규화 정도에는 차이가 없다. 이는 CPU 사용량 정보를 전혀 감쇄시키지 않아 궁극적으로 CPU 사용량이 프로세스 우선순위를 전적으로 결정하고 기본우선순위의 영향은 거의 없기 때문이다. 반대의 예로, D = 10.0인 행을 관찰하면, 기본우선순위 차이가 증가함에 따라 그룹 간 지연시간-사용량 정규화 정도 차이가 선명하게 나타나는데, 기본우선순위가 프로세스 우선순위 결정에 주도적인 역할을 하고 있기 때문이다.

4.2 CPU 사용량 정보 감쇄비율에 따른 지연시간-사용량 정규화 정도

이번에는 프로세스 그룹 간 기본우선순위 차이를 고정하고, CPU 사용량 정보 감쇄비율을 증가시킬 때의 영향을 규명하기 위해 base={0,0,0,0}인 열을 살펴보자. D 가 증가함에 따라 지연시간-사용량 정규화 특성이 점차 불분명해져 맨 아래 그림에서는 점들이 넓은 영역에 산개함을 관찰할 수 있다. 이는 D 에 의한 감쇄 동작이 과거의 CPU 사용량 정보를 줄이는 것이기 때문에, D 가 증가함에 따라 과거 사용량 정보를 우선순위에 반영함으로써 나타나는 지연시간-사용량 정규화 특성이 희박해지기 때문이다. 이로써 과거 CPU 사용량 정보를 감쇄하는 비율에 따라 지연시간-사용량 정규화 정도의 또 다른 차원의 연속적 스펙트럼이 존재함을 확인할 수 있다. D 의 증가에 따라 정규화가 희석되는 이러한 현상은 기본우선순위 값들에 차이를 두는 다른 열들에서도 유사하게 관찰된다. 단, 그룹 간 기본우선순위 값의 차이가 크면 (예, base={0, 20, 40, 60}인 열) 정규화는 희석되고 기본우선순위 차이에 의해 주도되는 그룹 간 서비스 차별화만 뚜렷하게 관찰된다.

과거 CPU 사용량이 프로세스 우선순위에 반영되는 정도를 "과거 사용량 정보 민감도"라고 정의하면, 이를 조절하는 decay usage의 매개변수로는 D 외에 T 와 R 도 있다. T 는 감쇄를 하는 주기를 결정하는 매개변수이므로 증가할 경우 감쇄를 느리게 하고, 감소할 경우 빠른 감쇄가 이루어지게 한다. R 은 cpu_i 값이 pri_i 에 영향을 미치는 가중치를 나타내므로, 증가할 경우 과거 사용량 정보 민감도를 높이는 역할을 하고 감소할 경우 과거 사용량 정보 민감도를 낮추게 된다. 따라서 T 나 R 을 변화시킬 경우에도 비슷한 스펙트럼을 관찰할 수 있다. 따라서 D, T, R 세 개의 매개변수는 decay usage가 프로세스 우선순위를 결정할 때의 과거 사용량 정보 민감도에 영향을 주며, 이러한 관찰들을 요약하면 다음과 같은 특성으로 정리된다.

특성 3: 과거 사용량 정보 민감도에 따라 지연시간-사용량 정규화 정도가 달라지고 그 정규화 정도의 변이가 또 다른 차원의 연속적인 스펙트럼을 형성한다.

4.3 시스템부하에 따른 서비스 품질 분석

그림 5의 실험은 CPU의 전체 사용률(시스템부하)을 변화시켜 가면서 한 프로세스 그룹이 기본우선순위의 크기에 따라 서비스 품질에 어떤 영향을 받는지를 관찰한 것이다. 관찰 대상이 되는 프로세스 그룹은 동일한 기본우선순위 $base_j$ 를 가지는 10개의 프로세스이며 총 0.3의 작업부하를 생성한다. 그 그룹의 기본우선순위

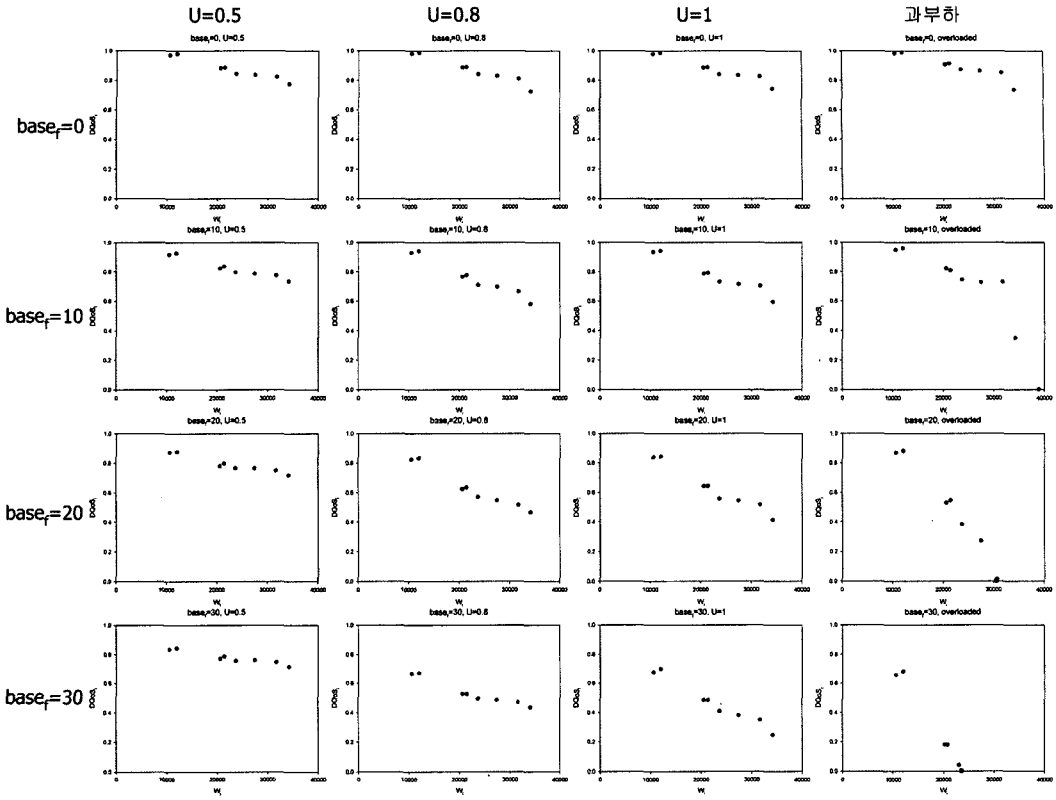


그림 5 사용률 변화에 따른 기본우선순위별 서비스 예측가능성

$base_f$ 를 0, 10, 20, 30으로 변경하면서 시스템부하의 영향을 관찰한다. 시스템부하를 조정하기 위한 배경 트래픽은 0부터 29까지의 기본우선순위를 가지는 30개의 프로세스가 무작위의 작업요청을 내도록 하였다. 이 때 이들의 배경 트래픽의 작업부하는 0.2, 0.5, 0.7, 0.9로 변화시켰으며, 따라서 전체 시스템부하는 0.5, 0.8, 1, 1.2 (과부하) 상태로 변화한다.

그림 5의 $base_f$ = 0인 행에서 보이는 것처럼 관찰대상이 되는 그룹의 기본우선순위가 상대적으로 작은 경우에는 시스템의 부하가 변하더라도 서비스의 품질에 거의 변화가 없는 반면에, $base_f$ 가 증가할수록 시스템의 부하가 낮을 때와 과부하일 경우의 서비스 품질 차이가 증가한다. 이 때 시스템 부하 변화 정도에 따른 서비스 품질 변화 정도를 "서비스 예측가능성"이라고 정의하면, 기본우선순위가 상대적으로 작은 프로세스는 시스템의 부하 상태와 무관하게 서비스 예측가능성이 높은 반면에, 기본우선순위가 상대적으로 큰 프로세스는 서비스 품질이 시스템의 상태에 종속적이며 그 예측가능성이 낮음을 알 수 있다.

특성 4: 프로세스의 기본우선순위가 상대적으로 작을수록 서비스 품질 예측가능성은 높다.

앞의 실험에서는 동일한 기본우선순위를 가지는 프로세스들을 다수 실행하여 이들의 집단적 행태를 관찰한 것에 반해, 그림 6의 실험에서는 20개의 프로세스를 실행하고, 그 중 하나의 프로세스에 대해 기본우선순위와 작업부하 W_i 를 바꾸면서 서비스 품질을 관찰한다. 나머지 19개의 프로세스는 배경 트래픽으로서 작업요청을 무작위로 발생시키되 이들의 총작업부하가 1이 되도록 하였다. 이 때 이 배경 프로세스들은 31~49 사이의 기본우선순위를 가지도록 설정하였다. 관찰 대상이 되는 프로세스는 기본우선순위 $base_f$ 를 0부터 50까지 10 단위로 변화시켰으며, 각 $base_f$ 에 대해 작업부하 W_i 를 0.01에서 0.3까지 부과하였다. 전체 프로세스의 부하를 더하면 항상 과부하가 걸린 상황이다. 또한 D = 2, T = 1000, R = 0.5를 사용하였다. 이 실험에서 관찰대상이 되는 프로세스의 기본우선순위가 0인 경우, 즉 $base_f$ = 0인 경우, 그 프로세스가 다른 프로세스들에 비해 월등하게 높은 기본우선순위를 지니고 있으므로 항상 최상의 서비스를 받는다. 그러나 $base_f$ 가 커짐에 따라 다른 프로세스의 기본우선순위와의 차이가 좁혀지고 따라

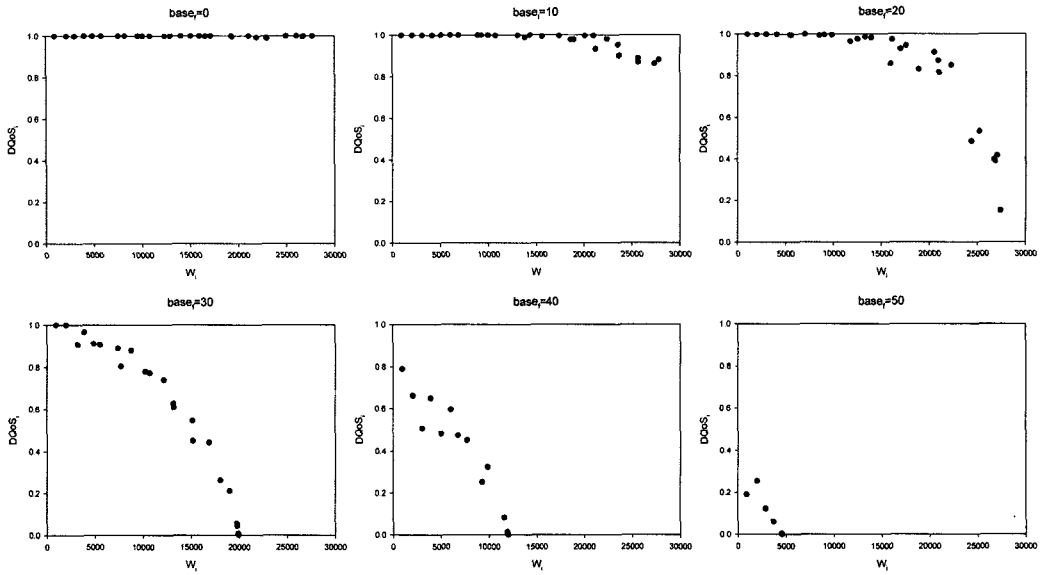


그림 6 과부하시의 기본우선순위 변화에 따른 서비스 품질

서 많은 작업부하를 내는 경우 우선순위 역전 현상이 발생할 수 있다. 그 결과 관찰대상 프로세스의 작업부하 W_i 가 특정한 양 이상이 되는 지점에서부터 서비스 품질의 저하가 보이기 시작한다. $base_j$ 가 커짐에 따라 서비스 품질 저하가 시작되는 지점은 점점 앞당겨져 $base_j$ = 50으로 커진 경우 대부분의 시간 동안 다른 프로세스들에 비해서 우선순위가 현저히 낮기 때문에 서비스를 거의 받지 못한다.

특성 5: 과부하 상황에서도 프로세스의 기본우선순위에 따라 특정한 작업부하량 이내의 요청에 대해서는 서비스 품질이 일정한 수준을 유지하며, 그 양을 넘어서면 서비스 품질이 저하된다.

5. 분석결과의 응용가능성

지금까지 기술한 decay usage의 특성 분석을 통해 매개변수들의 의미를 다음과 같이 정리할 수 있다.

- 각 프로세스에게 할당되는 기본우선순위 $base_i$: 사용량과 지연시간을 포괄적으로 고려한 관점에서의 서비스 품질의 차별화 정도를 결정한다.
- 시스템 매개변수 D, T, R: 과거 CPU 사용량 정보 민감도를 조정하여 지연시간-사용량 정규화 정도를 결정한다. D 를 키울수록, T 를 줄일수록, R 을 줄일수록 과거사용량 정보 민감도를 줄여 지연시간-사용량 정규화 정도를 희석하고 그 결과 기본우선순위 차이에 의한 서비스 품질 차별화를 강조한다.

이러한 이해를 바탕으로 decay usage 알고리즘의 매개변수를 적절히 설정함으로써 스케줄러가 특정한 형태로 동작하도록 설정하는 것이 가능하다. 예를 들어, R = 1, T = 실행 큐 안의 프로세스의 수, D = ∞로 설정하고 모든 $base_i$ 를 0으로 하면 decay usage는 라운드로빈 스케줄러로 동작한다. 또한, 그림 3에서 보인 바와 같이 동일한 기본우선순위를 가진 프로세스들 간에 지연시간-사용량 정규화 특성이 존재한다는 점과, 그림 4에서 보인 바와 같이 서로 다른 기본우선순위를 가지는 프로세스들이 서로 구분되는 띠(band)를 형성하는 특성을 보이는 것은, 기본우선순위를 매개변수로 하는 서비스 클래스의 구분 가능성을 보여준다. 이 때 같은 서비스 클래스에 속한 프로세스들은 지연시간과 사용량을 포괄한 관점에서 동일한 서비스를 받으며, 기본우선순위가 작은 클래스일수록 더 좋은 서비스를 받는다. 이러한 관계를 체계적으로 모델링하면 서비스 사용량과 지연시간의 정규화 관점에서의 새로운 과금 체계를 도출할 수 있을 것으로 보인다.

또한 그림 5의 실험에서 보듯이 기본우선순위의 상대적 크기에 따라 서비스의 예측가능성이 달라지기 때문에 완벽한 보장성 서비스를 제공할 수는 없지만, 시스템의 사용률과 상대적 기본우선순위에 기반한 확률적인 모델을 설립할 수 있을 것으로 보인다. 특히 그림 6의 실험결과를 보면 기본우선순위에 무관하게 작업부하가 특정한 양에 도달하기까지는 일정한 서비스 품질을 보이다가 그 양을 넘어서는 지점부터 서비스 품질이 서서히 저하되는 것을 관찰할 수 있는데, 이러한 특성은 사용자가 예약한 작업부하 이내의 양을 발생할 때는 서비

스 품질을 보장해 주다가 계약된 양을 넘어서면 품질을 서서히 저하시키는 형태의 서비스 가능성을 시사한다. 좀 더 구체적으로, 프로세스 간 기본우선순위 $base_i$ 의 차이를 현저히 넓히고 D, T, R 을 조정하여 과거 사용량 정보 민감도를 작게 하면, decay usage가 고정우선순위 스케줄링과 완전히 동일하게 동작하도록 만들 수 있다. 더 나아가, 기본우선순위 $base_i$ 의 차이와 D, T, R 을 조정하여, 계약된 길이 이내의 요청에 대해서는 고정우선순위 스케줄링에서와 같은 보장 서비스를 제공하고 그 이상에 대해서는 CPU 사용량에 따라 점차적으로 서비스 품질을 저하시키는 방안도 가능하다.

이러한 구체적 모델의 정립은 본 논문의 범위를 넘어서는 것이나, 앞에서 제시한 가능성을 토대로 체계적인 모델을 정립하기 위한 추가적인 연구를 진행하고 있다.

6. 결 론

이 논문에서는 UNIX 상의 CPU 스케줄러로 널리 사용되어 온 decay usage 알고리즘에 지연시간-사용량 정규화 특성이 존재하고 그 정규화 정도가 매개변수에 따라 연속적 스펙트럼들을 형성함을 밝히고 이를 분석하였다. 과거에 CPU를 적게 사용한 프로세스를 우대하는 decay usage의 동작 방식으로 인해 동일한 기본우선순위를 지니는 프로세스들 사이에는 지연시간-사용량 정규화 관계가 존재하며, 기본우선순위의 상대적인 크기에 따라 연속된 스펙트럼이 형성된다. 또한 사용량 정보의 감쇄 동작시 그 감쇄 정도에 의해 결정되는 과거정보 민감도에 따라 또 다른 차원의 스펙트럼이 존재한다. 본 논문에서는 실험을 통해 이러한 관계를 입증하고 이들의 의미를 규명하였다. 그리고 이러한 분석결과를 토대로 지연시간-사용량 정규화 개념에 기반한 새로운 서비스 제공 모델의 가능성에 대해 토의하였다.

참 고 문 헌

[1] M. J. Bach, *The Design of the UNIX Operating System*, Prentice-Hall, 1986.

[2] M. K. McKusick, K. Bostic, M. J. Karels, and J. S. Quarterman, *The Design and Implementation of the 4.4BSD Operating System*, Addison-Wesley, 1996.

[3] M. K. McKusick and G. V. Neville-Neil, *The Design and Implementation of the FreeBSD Operating System*, Addison-Wesley, 2004.

[4] D. L. Black, "Processors, Priority, and Policy: Mach Scheduling for New Environments," *Proceedings of USENIX '91*, pp. 1-9, 1991.

[5] B. Goodheart and J. Cox, *The Magic Garden Explained: The Internals of UNIX System V Release 4, an Open Systems Design*, Prentice-Hall, 1994.

[6] U. Vahalia, *UNIX Internals: The New Frontiers*, Prentice-Hall, 1996.

[7] J. L. Hellerstein, "Achieving Service Rate Objectives with Decay Usage Scheduling," *IEEE Transactions on Software Engineering*, Vol.19, No.8, pp. 813-825, August 1993.

[8] D. H. J. Epema, "Decay Usage Scheduling in Multiprocessors," *ACM Transactions on Computer Systems*, Vol.16, No.4, pp. 367-416, November 1998.

[9] D. H. J. Epema and J. F. C. M. de Jongh, "Proportional-Share Scheduling in Single-Server and Multiple-Server Computing Systems," *ACM SIGMETRICS Performance Evaluation Review*, Vol.27, No.3, pp. 7-10, December 1999.

[10] D. Petrou, J. W. Milford, and G. A. Gibson, "Implementing Lottery Scheduling: Matching the Specializations in Traditional Schedulers," *Proceedings of USENIX '99*, pp. 1-14, 1999.

[11] C. A. Waldspurger and W. E. Weihl, "Lottery Scheduling: Flexible Proportional-Share Resource Management," *Proceedings of the First Symposium on Operating Systems Design and Implementation*, pp. 1-11, 1994.

[12] R. L. Cruz, "Quality of Service Guarantees in Virtual Circuit Switched Network," *Journal on Selected Areas in Communications*, Vol.13, No.6, pp. 1048-1056, August 1995.

[13] H. Sariowan, R. L. Cruz, and G. C. Polyzos, "SCED: A Generalized Scheduling Policy for Guaranteed Quality-of-Service," *IEEE/ACM Transactions on Networking*, Vol.7, No.5, pp. 669-684, October 1999.

[14] 이주현, 황호영, 민상렬, "지연-대역폭 정규화 관점에서의 출력링크 서비스 알고리즘", *한국정보과학회 제33회 추계학술발표회 논문집*, pp. 259-262, 2006.

[15] 박경호, 황호영, 민상렬, "지연시간-대역폭 정규화 기반의 스케줄링 모델", *한국정보과학회 제33회 추계학술발표회 논문집*, pp. 176-180, 2006.

박 경 호
정보과학회논문지 : 시스템 및 이론
제 34 권 제 6 호 참조

황 호 영
정보과학회논문지 : 시스템 및 이론
제 34 권 제 6 호 참조

이 창 건
정보과학회논문지 : 시스템 및 이론
제 34 권 제 6 호 참조

민 상 렬
정보과학회논문지 : 시스템 및 이론
제 34 권 제 6 호 참조