지연시간-대역폭 정규화 기반의 스케줄링 모델

박경호° 황호영* 민상렬
서울대학교 컴퓨터공학부 안양대학교 디지털미디어학부*
{kalynda°, symin}@archi.snu.ac.kr hyhwang@aycc.anyang.ac.kr*

A Scheduling Model Based on Delay-Bandwidth Normalization

Kyeongho Park° Ho Young Hwang* Sang Lyul Min
School of Computer Science & Engineering, Seoul National University
Department of Digital Media, Anyang University*

요 약

이 논문에서는, 과거의 사용량 정보와 서비스 지연시간이 상호 의존관계를 가지는 지연시간-대역폭 정규화 개념을 설명하고, 이에 기반한 스펙트럼 형태의 스케줄링 모델을 제시한다. 이 모델에서는 각 응용이 자원을 획득할 수 있는 권한을 주기적으로 축적하며, 서비스를 받을 경우 그 권한을 소비하게 된다. 사용되지 않고 축적된 권한은 추후의 스케줄링에서 자원 획득 가능성을 높여 지연시간을 단축시키는 효과를 낸다. 이 때 과거의 축적된 정보를 주기적으로 감쇄시킴으로써 과거의 사용 정보를 부분적으로 망각하도록 할 수 있으며, 그 감쇄 정도에 따라 지연시간-대역폭 정규화 정도를 제어할 수 있다. 이 기본적 모델의 세부사항을 조절함으로써 이 모델이 GPS, virtual clock, decay usage 등의 스케줄러와 유사한 특성을 나타낼 수 있음을 보였으며, 이를 통해 기존의 무관해 보이는 스케줄러들이 연속적인 스펙트럼상에 존재함을 설명하였다. 또한 시뮬레이션을 통해 모델의 특성을 관찰하였다.

1. 서 론

이 논문에서는 서비스를 받은 양과 지연시간 사이에 상호 의존관계가 존재하는 지연시간-대역폭 정규화 개념을 설명하고, 이 관점에 기반한 스펙트럼 형태의 스케줄링 모델을 제시한다. 또한, 이 스펙트럼 상에 기존의 서로 무관해 보이는 스케줄링 알고리즘들의 특성들이 포괄될 수 있음을 보인다.

컴퓨터 시스템에서의 자원 배분을 위해 다양한 철학적 배경을 지니는 스케줄러들이 기존에 제시되어 있는데, GPS(generalized processor sharing), virtual clock, decay usage 등의 예를 들 수 있다. GPS[1][2]는 매 시점의 자원의 공정한 배분에 초점을 맞춘 방법으로, 가용한 자원을 현재 적체된(backlogged) 응용들 사이에서만 각각의 가중치에 비례하여 순시적(瞬時的)으로 나누는 방법이다. 반면에 virtual clock[6][1]은 각 응용이 받은 서비스의 총합이 가중치에 비례하도록 하는 방법으로, 과거에 받지 못한 서비스를 미래에 전부 보상하여 장기적인 관점에서 공정한 자원배분을 추구하는 특징을 지닌다. Decay usage[7][8]는 CPU를 최근에 적게 사용한 작업에 높은 우선순위를 주는 방법으로, I/O 위주 작업의 응답시간을 개선함과 동시에 시스템의 사용률을 높이는 것을 지향한다.

위에 언급한 기존의 다양한 스케줄러들은 각각 다른 철학을 지니고 만들어졌으므로 일견 서로 관련이 없는 것처럼 보인다. 그러나 이들의 동작 원리를 살펴보면, 과거의 자원 사용 정보를 어떻게 유지하고 이 정보를 이후의 스케줄링에 얼마나 반영하는가 하는 관점에서 연속적인 속성을 지니고 있음에 주목할 수 있다. 즉, GPS의 경우 현재 적체되어 있는 응용들에게 가용한 대역폭을 모두 나누어 주고 휴지 상태의(idle) 응용에게 이를 나중에 보상해 주지 않는 반면, virtual clock은 과거의 사용하지 않은 대역폭을 기억해 두었다가 미래에 모두 보상해 주는 특성을 지닌다. 한편, decay usage의 경우 I/O 위주의 응용들이 사용하지 않은 대역폭은 우선순위 정보에 반영되어 나중에 CPU 요청에 대한 지연시간을 개선해 주는데, 이 정보는 주기적으로 감쇄되고 이 감쇄 정도에 따라 과거의 정보가 반영되는 정도가 달라진다. 이러한 관찰을 종합해 보면, 과거의 서비스 정보(대역폭)를 유지하는 정도에 의해 미래의 서비스(지연시간)가 영향을 받는 지연시간-대역폭 정규화 관계의 연속된 스펙트럼이 존재함을 알 수 있다.

이 논문에서는 이러한 지연시간-대역폭 정규화 관점에서 스펙트럼 형태의 스케줄링 모델을 제시하고, 이 모델이 기존의 서로 무관해 보이는 스케줄링 알고리즘들을 포괄함을 보인다. 이 모델에서 각 응용은 자원을 획득할 수 있는 권한을 하나의 값으로 관리하며, 스케줄러는 이 값의 크기에 기반하여 서비스할 응용을 선택한다. 이 모델은 이 값이 축적, 감쇄, 소비되는 방법을 정의하고 있

1) [6]에는 여러 가지 버전의 virtual clock이 언급되어 있는데, 여기에서는 이 논문의 문맥에 적합한 첫 번째 버전을 가리킨다.

으며, 이러한 방법의 세부적 결정에 의해 전체 모델의 움직임이 결정된다. 특히 이 값의 주기적인 감쇄 정도의 조절에 의해 지연시간-대역폭 정규화 정도를 제어할 수 있으며, 이 모델은 위에서 언급한 기존의 스케줄러들과 유사한 특성을 보이게 된다.

다양한 스케줄러들을 종합하여 스펙트럼형 모델로 설명하려는 기존의 시도들은 그리 많지 않다. [9]에서는 공정 큐잉에서 사용되는 스케줄러들을 그 특성에 따라 분류, 정리하였으며, [10]의 경우, 서비스율을 보장하는 스케줄러의 클래스를 정의하고 몇몇 알고리즘이 이에 포함된다는 것을 보인 바 있다. 그러나 본 논문에서 제시된 것과 같은 관점의 스펙트럼으로 파악하려는 연구와는 다소 거리가 있다.

본 논문의 구성은 다음과 같다. 2절에서는 지연시간-대역폭 정규화를 표현하기 위한 기본적인 모델을 정의한다. 3절에서는 이 모델의 적절한 세부적 설정을 통해 기존의 특정 스케줄러들 중 GPS, virtual clock, decay usage 등과 유사한 면이 있음을 설명한다. 4절에서는 실험을 통해 이 모델의 특성을 검토하고, 5절에서 결론을 맺는다.

2. 지연시간-대역폭 정규화 모델

이 절에서는 이 논문에서 제시하는 기본적인 모델에 대해서 설명한다(그림 1). 이 모델에서는 과거의 사용 정보가 추후의 스케줄링에 얼마만큼의 영향을 미치는지를 주요 매개변수로 삼고 있으며, 과거의 사용 정보를 스케줄링 권한에 반영하고 이를 감쇄시키는 정도를 제어하는 등의 동작을 수행하게 된다.

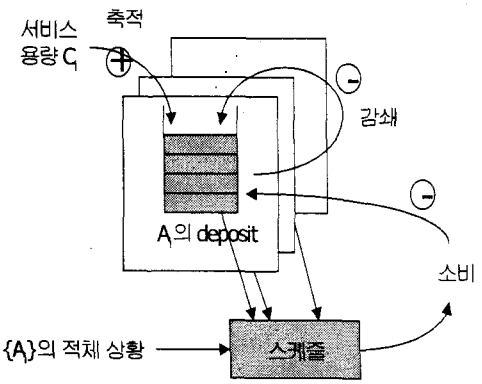

그림 1. 기본적인 모델

이 모델에서 가정하는 서버의 특성은 다음과 같다. 자원을 서비스하는 서버의 용량은 편의상 1로 가정하며, 서비스는 시간 슬롯 단위로 스케줄링된다. 그리고 서버는 작업보전형(work-conserving)으로 동작하는데, 이는 수행할 작업이 있는 한 서버는 중단없이 계속 서비스를 수행함을 의미한다.

2.1 deposit의 역할

이 모델은 기본적으로 과거에 자신에게 주어진 권한에 비해 적게 서비스를 받은 응용의 우선순위를 높게 주는 정책에 기반하고 있는데, 과거에 응용에게 주어졌으나 사용되지 않은 대역폭이 어느 정도인지의 정보를 deposit이라는 하나의 값으로 나타낸다. 따라서 deposit은 각 응용이 현재의 시점에 자원을 획득할 수 있는 상대적인 권한을 나타내며, 스케줄러는 deposit이 가장 큰 응용을 선택한다. deposit은 정해진 규칙에 의해 축적, 감쇄, 소비되며, 이 규칙들을 다르게 설정함으로써 다양한 형태의 스케줄러를 표현할 수 있다.

2.2 deposit의 축적

응용들에게는 자신의 가중치에 맞추어 단위시간당 일정한 자원을 서비스받기 위해 사용할 수 있는 권리가 부여된다. 응용 i는 미리 정해진 고유한 서비스 용량 C_i를 주기적으로 받게 되는데(단, $\Sigma C_i = 1$), C_i는 일반적인 공정 스케줄러 알고리즘들[1][2]에서의 가중치(weight)에 기반하여 얻어지며, decay usage 같은 스케줄러에서는 기본 우선순위(base priority) 및 다른 매개변수들에 의해 유도되어질 수 있다[7]. 응용이 항상 적체되어 있고 서비스가 유체(fluid) 모델 형태로 진행된다면, 각 응용은 자신에게 주어지는 이 권리를 즉시 완전히 소비할 것이다. 그러나 실제 서비스 진행 과정에서 응용은 적체 상태와 휴지 상태를 반복하게 되며, 한 시점에는 한 응용만이 패킷 단위로 서비스된다. 따라서 발생 즉시 사용되지 않은 권리는 deposit에 축적되며, 이로 인해 deposit이 증가한 응용은 다음번 스케줄링에서 선택될 가능성이 상대적으로 높아진다.

권리가 사용되지 않고 계속 축적될 경우, 시간 t에 응용 i의 deposit $D_i(t)$는

$$D_i(t) = D_i(t-1) + C_i$$
$$D_i(0) = 0 \qquad (1)$$

으로 표현된다. 이 때 deposit은 응용이 과거에 부여받았으나 사용하지 못한 권리를 보관해 놓은 것으로 해석가능하다. deposit이 상대적으로 크다는 것은 자원을 획득하기 위한 경쟁에서 유리하기 때문에 지연시간을 줄일 수 있다는 의미이며, 동시에 권리의 미사용분에 대해 그만큼의 보상을 받을 수 있다는 의미가 된다. 결국 이러한 모델에서는 자원의 사용량과 지연시간이 상호 상관관계를 가지게 되며, 이를 지연시간-대역폭 정규화 관계로 해석할 수 있다.

2.3 deposit의 소비

응용이 자원을 사용한 경우 서비스의 비용에 해당하는 만큼을 deposit에서 차감하는데, 이는 일률적으로 적용되거나, 또는 현재 deposit의 크기에 따라 차등적으로 적용될 수 있다. 이 때 deposit의 증감이 균형을 맞추는 흐름균형화(flow balancing)가 이루어져야 할 경우가 있으며, 세부적인 것은 3절에서 언급한다.

2.4 deposit의 감쇄

deposit에는 과거의 자원 사용에 대한 정보가 축적되어

있다. 만일 deposit을 주기적으로 일정비율로 감쇄시키면, 이런 과거의 정보를 부분적으로 망각하도록 할 수 있다. 이 때 감쇄 정도에 변화를 주면 시간의 경과에 따라 보상받는 정도를 조절할 수 있으며, 결과적으로는 지연시간도 이에 영향받게 된다. 즉 deposit의 주기적 감쇄 정도에 따라 지연시간-대역폭 정규화 정도를 제어할 수 있다.

감쇄 정도를 λ라는 상수로 표현한다면 deposit의 감쇄 모델은 다음의 식으로 표현 가능하다.

$$D_i(t) = \lambda D_i(t-1), \quad 0 \le \lambda \le 1 \tag{2}$$

이 때 λ는 과거의 기억을 유지하는 정도를 결정한다. 식 (1)을 감안할 경우 전체적으로 다음과 같이 정리할 수 있다.

$$D_i(t) = \lambda D_i(t-1) + C_i, \quad 0 \le \lambda \le 1 \tag{3}$$

위 식에서 표현된 deposit에는 과거에 사용되지 않은 권리의 정보가 축적되어 있으며, 시간 t에서 멀리 떨어진 과거의 정보일수록 여러 번의 감쇄를 거치기 때문에 전체 deposit 값에 기여하는 정도가 적음을 알 수 있다.

그림 2에 λ의 변화에 따라 deposit의 증가 모습이 달라짐을 보였다(단, 계속 서비스는 받지 않았다고 가정했을 때). λ=0일 때 deposit은 축적되지 못하므로 증가할 수 없다. 반면에 λ=1일 때는 deposit은 무한히 증가하게 된다. 0<λ<1일 때는 deposit이 증가함에 따라 감쇄되는 양도 많아져서 특정한 값에 수렴하게 된다. 개념상 λ=0인 경우는 GPS와 유사한 특성을 가지고, λ=1인 경우는 virtual clock에 해당하는 특성을 가진다고 볼 수 있다. 그리고, 0<λ<1인 경우는 decay usage와 유사하다고 볼 수 있다.

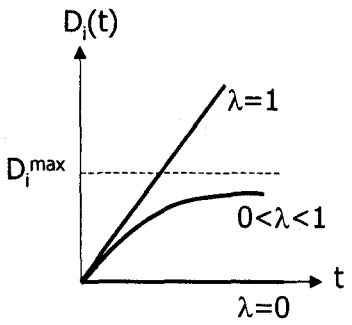

그림 2. λ에 따른 deposit의 변화

λ의 변화에 따라 각 응용의 최대 deposit 값이 결정되며, 이는 서비스 지연시간에 영향을 줄 것이라고 예측할 수 있다. 일반적으로 한 응용의 최악지연시간은 그 응용의 deposit이 다른 응용들의 deposit과 최대의 차이를 가질 때 발생한다. 따라서, λ가 0일 경우 가장 적은 최악지연시간을 가지게 되며 λ가 증가할수록 최악지연시간은 증가할 것이다.

위의 관찰을 통해 이 모델이 지연시간과 서비스양의 홍정(tradeoff) 관계를 나타내고 있음을 알 수 있다. 또한 과거의 정보를 오래 유지할수록 개별 응용의 지연시간 한계는 늘어나지만 장기적 관점에서의 응용들 간의 서비스 양에 공정성을 추구할 수 있으므로, 지연시간과 장기적 공정성 간에도 홍정관계가 성립함을 알 수 있다.

3. 기존의 스케줄러와의 연관성

이 절에서는 본 논문의 모델이 기존의 GPS, virtual clock, decay usage와 유사한 특성을 보일 수 있음을 설명하고, 그 특성들에 대해 고찰한다.

3.1 GPS

GPS는 매 시점에 가용한 서버 용량을 적체된 응용들의 가중치에 비례하여 배분하는 공정 큐잉 방법이다. 가중치가 r_i인 응용 i가 구간 (t_1, t_2)에서 받은 서비스의 양을 $S_i(t_1, t_2)$라고 하면, GPS에서는 응용 i가 (t_1, t_2) 구간에서 연속 적체되어 있을 때

$$\frac{S_i(t_1,t_2)}{S_j(t_1,t_2)} \ge \frac{r_i}{r_j}, \quad j = 1,2,...,N \tag{4}$$

을 만족한다[1]. GPS는 순간적인 관점에서 가장 공정한 스케줄링 방법이라고 할 수 있으나, 서버의 용량을 무한히 나누어 여러 응용에 동시에 서비스 가능하다고 가정하는 유체 모델이다. 따라서 현실의 스케줄러에 그대로 적용할 수는 없으며, 이를 패킷 모델에 응용한 PGPS[1][2], SFQ(start-time fair queueing)[3], stride 스케줄링[4], MCF(most credit first)[5] 등의 다양한 알고리즘들이 제안되어 있다.

GPS는 가용한 용량을 적체되어 있는 응용들간에 가중치의 비율대로 나누어 가지는 모델이므로 적체되어 있지 않고 휴지 상태인 응용들이 받지 못한 서비스는 보상되지 않는다. 따라서 개념적으로는 본 모델에서 λ=0인 경우에 해당한다. 그러나 패킷 모델상에서 GPS를 구현한 알고리즘을 본 모델에서 표현하기 위해서는 모델의 세부적인 사항들을 다음과 같이 변경해 줄 필요가 있다.

(1) 패킷 단위의 서비스를 수행할 때, 적체 구간에서 다른 응용과의 경쟁에서 밀려서 사용하지 못하고 축적된 deposit은 스케줄링으로 인해 발생하는 오차라고 볼 수 있으므로 보상할 필요가 있다. 따라서 응용의 적체구간에서는 λ=1을 적용한다. 즉 다음과 같이 정리할 수 있다.

$$D_i(t) = \lambda D_i(t-1) + C_i$$
$$\lambda = 0: \text{ 휴지구간}$$
$$\lambda = 1: \text{ 적체구간} \tag{5}$$

(2) 서비스시마다 고정된 양의 deposit을 소비하면, 적체된 응용들의 deposit이 음의 방향으로 발산하는 문제가 발생할 수 있다. 이 문제를 해결하기 위해 deposit의 전체 증감분이 균형을 이루도록 흐름 균형화를 수행한다. 본 모델에서는 서비스된 응용의 deposit을 다음 값만큼 소비함으로써 이 목적을 달성한다.(단, Q는 적체되어 있는 응용들의 집합)

흐름균형화를 위한 deposit의 소비분 = $\sum_{j \in Q} C_j$ (6)

(3) 서비스를 받고 deposit이 음수가 된 상태의 응용이 잠깐 동안의 휴지상태를 거쳐 적체상태로 바뀔 경우 deposit이 0으로 재설정되므로, 연속적으로 적체되어 있는 다른 응용에 비해 과도한 서비스를 받는 불공정성이 발생할 수 있다. 이를 해결하기 위해서는 휴지 상태에서 무조건 λ=0을 적용하지 않고 deposit이 음수인 구간 동안에 한해 λ=1을 적용한다.

3.2 virtual clock

이 논문에서 모델로 하는 virtual clock은 그림 3과 같은 동작을 수행한다[6].

1. 각 응용은 가중치에 따라 $Vtick_i = 1/r_i$를 가진다.
2. 패킷이 도착할 때마다 VC_i += $Vtick_i$를 수행한다.
   단, 응용의 최초의 패킷에 대해서는
   VC_i = *real_time*으로 설정된다.
3. 적체된 응용들의 VC_i의 크기순서대로 스케줄링된다.

그림 3. virtual clock 알고리즘

virtual clock은 응용이 사용하지 않은 권리를 모두 deposit에 반영하고 보상하는 모델이므로 λ=1에 해당한다. 또한 서비스는 deposit의 크기에 무관하게 동일한 가치를 지니므로 서비스시의 소비량은 항상 1로 고정된다. virtual clock에서는 과거에 받지 못한 서비스에 대해서 요청이 있으면 전부 보상해 주는 방식이므로, 무한대의 최악지연시간을 가질 수 있음을 직관적으로 쉽게 알 수 있다.

3.3 decay usage

그림 4에 decay usage의 기본적인 동작 방식을 보였는데[7], 다양한 UNIX 계열 운영체제들에서 이를 약간씩 변형하여 사용하고 있다. 각 응용은 기본 우선순위 $base_i$를 가지는데 이 값이 작을수록 많은 자원을 획득할 수 있는 기회를 가진다. 실제 자원의 할당 비율은 $base_i$와 함께 감쇄주기의 결정에 영향을 주는 D, T 및 우선순위 값의 증가율에 영향을 주는 R의 값에 의해 결정된다.

응용 i가 한 quantum을 소비할 때마다:
  cpu_i++
매 T quantum마다:
  cpu_i /= D (D>1)
다음 값이 최소인 응용 i를 실행:
  pri_i = R * cpu_i + $base_i$

그림 4. decay usage 알고리즘

decay usage의 경우는 개념상 본 모델에서 $0<\lambda<1$의 경우에 해당한다. 그리고 위 알고리즘에서 서비스를 받을 때마다 pri_i 값이 표면상 동일한 단위만큼 변화하고 있으나, 실제 그 값의 변화분이 가지는 가치는 현재 pri_i 값의 크기에 따라 다르다고 볼 수 있다. 본 모델에서는 이 관계를 반영하기 위해 서비스시에 deposit의 소비분을 현재 시점의 deposit의 값에 비례해서 적용하는 모델을 적용한다. 즉, 서비스에 따른 deposit의 소비분은 $\mu D_i(t)$로 표현된다. ($0 \le \mu \le 1$)

그러나 본 모델이 decay usage와 정확히 동일하게 동작하도록 만들 수는 없는데, 그 이유는 일단 본 논문의 모델이 매 단위시간마다 사용량 정보에 대한 감쇄 작업을 수행하는데 반해 decay usage는 T라는 주기 단위로 그에 해당하는 작업을 수행하기 때문이며, 또한 decay usage는 효율적 구현을 위해 정수만을 사용하기 때문이다. 이는 decay usage 알고리즘을 단순화하기 위한 과정에서 발생하는 양자화 오류(quantization error)에 기인하는 특성으로 간주할 수 있다.

이 모델에서는 감쇄 효과 때문에 deposit이 무한히 증가할 수 없으므로 계속 축적되더라도 특정한 값에 수렴하게 되며, 이 값은 다음과 같이 계산된다.

$$D_i^{\max} = \lambda D_i^{\max} + C_i$$

$$D_i^{\max} = \frac{C_i}{1-\lambda} \quad (7)$$

또한 deposit이 가지는 최소값도 다음과 같이 계산가능하다.

$$D_i^{\min} = (1-\mu)(\lambda D_i^{\min} + C_i)$$

$$D_i^{\min} = \frac{1-\mu}{1-\lambda(1-\mu)} C_i \quad (8)$$

4. 실 험

이 절에서는 간단한 시뮬레이션을 통해 본 모델과 decay usage와의 유사성에 대해 검토한다. decay usage 알고리즘은 크게 다음과 같은 두 가지 특성을 지닌다.

(1) CPU를 최근에 적게 사용한 작업에 높은 우선순위를 줌으로써 I/O 위주 작업의 응답시간을 개선한다
(2) I/O 위주의 작업이 CPU에서 빨리 서비스를 받으므로 I/O 쪽의 이용률이 높아지고 아울러 시스템의 전체적인 이용률을 높일 수 있다.

그림 5는 CPU 위주의 작업 A_1과 I/O가 포함된 작업 A_2를 동시에 수행할 때, 두 응용의 CPU 작업요청에 대한 평균지연시간을 관찰한 것이다. 각 응용의 서비스 용량 C_1, C_2는 각각 0.5씩으로 가정하였으며 μ는 0.3으로 설정하였다. 이 실험에서는 λ를 0.1부터 0.9까지 0.1 단위로 변화시키고, A_2의 전체 요청당 CPU 요청의 비율을 10%에서 90%까지 10% 단위로 변화시키면서 관찰하였다. 일단 전반적으로 I/O 위주의 작업인 A_2가 평균지연시간이 적음을 볼 수 있다. 그리고 λ의 값이 커짐에 따라 A_1의 지연시간은 증가하는데 반해 A_2의 평균지연시간은 감소하는 것을 볼 수 있다. 또한 A_2의 CPU 이용률이 적을수록 A_2의 평균지연시간은 큰 폭으로 감소한다. 이 결과는 CPU 사용률이 낮을수록, 그리고 지연시간-대역

폭 정규화의 정도가 높을수록 I/O 위주의 작업이 지연시간 측면에서 이득을 보고 있음을 나타내며, 이는 위의 특성 (1)에 부합한다.

그림 6은 위 실험과 같은 조건에서 λ의 변화에 따른 I/O 서버의 이용률 변화를 관찰한 것이다. λ가 증가함에 따라 서버의 이용률도 증가함을 볼 수 있는데, 이는 I/O 위주 작업의 CPU 요청에 우선순위를 주면 이 작업의 CPU에서의 대기시간이 줄어들기 때문에 I/O 서버의 이용률이 높아지기 때문이다. 이는 위의 특성 (2)에 부합하는 결과이다.

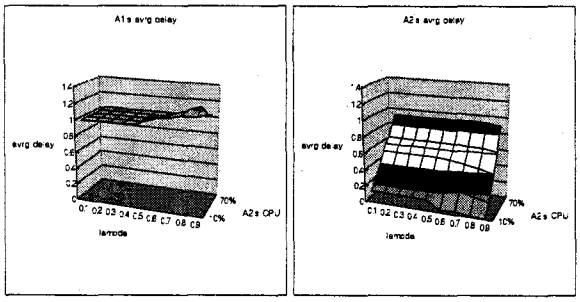

그림 5. 두 응용의 평균지연시간 관찰

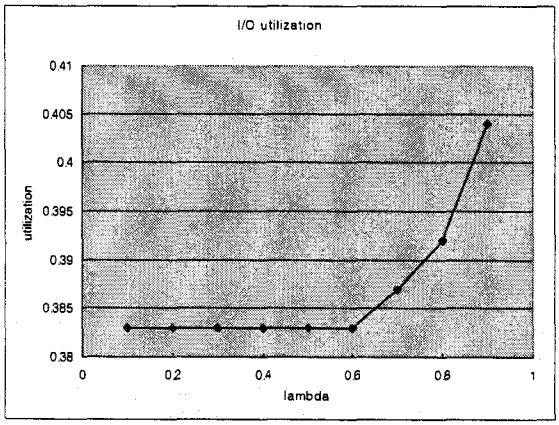

그림 6. I/O 서버의 이용률

5. 결 론

이 논문에서는 기존의 무관해 보이는 스케줄링 기법들을 지연시간-대역폭 정규화 관점에서 포괄하는 모델을 제시하였다. 이 모델에서는 각 응용이 자원을 획득할 수 있는 권한을 deposit이라는 값으로 나타내고, deposit은 응용별로 미리 정해진 고유한 비율로 계속 축적된다. 응용이 자원을 사용한 경우 deposit에서 차감하며, 차감되는 양은 일률적으로 적용되거나, 또는 서비스 지연시간이나 현재 deposit의 정도에 따라 차등적으로 적용된다. 서비스에 사용되지 않고 쌓여 있는 deposit을 주기적으로 감쇄시킴으로써 과거의 정보를 부분적으로 망각하도록 할 수 있는데, 이 때 감쇄 정도에 따라 지연시간-대역폭 정규화 정도를 제어할 수 있다. 본 모델의 세부적 사항들의 설정에 따라 기존의 GPS, virtual clock, decay usage 등의 스케줄러들과 유사한 특성을 나타냄을 관찰할 수 있었으며, 이는 이런 스케줄러들을 일관성 있게 포괄하는 스펙트럼이 존재하는 것을 의미한다.

향후 이 모델의 특성을 좀더 정확히 이해하기 위해서 수학적 분석과 추가적인 실험이 필요하다. 또한 이 모델을 실제 운영체제에 적용함으로써 지연시간-대역폭 정규화 개념을 체계적으로 구현한 스케줄러를 만들 수 있을 것으로 기대된다.

참고문헌

[1] A. K. Parekh and R. G. Gallager, "A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks: The Single-Node Case," IEEE/ACM Trans. on Networking, vol. 1, no. 3, Jun. 1993.

[2] A. Demers, S. Keshav, and S. Shenker, "Analysis and Simulation of a Fair Queueing Algorithm," SIGCOMM '89, 1989.

[3] P. Goyal, H. M. Vin, and H. Cheng, "Start-Time Fair Queueing: A Scheduling Algorithm for Integrated Services Packet Switching Networks," IEEE/ACM Trans. on Networking, vol. 5, no. 5, Oct. 1997.

[4] C. A. Waldspurger and W. E. Weihl, "Stride Scheduling: Deterministic Proportional-Share Resource Management," Tech. Rep. MIT/LCS/TM-528, MIT, 1995.

[5] D. Pan and Y. Yang, "Credit Based Fair Scheduling for Packet Switched Networks," IEEE INFOCOM 2005, 2005.

[6] L. Zhang, "VirtualClock: A New Traffic Control Algorithm for Packet Switching Networks," SIGCOMM '90, 1990.

[7] J. L. Hellerstein, "Achieving Service Rate Objectives with Decay Usage Scheduling," IEEE TOSE, vol. 19, no. 8, Aug. 1993.

[8] D. H. J. Epema, "Decay-Usage Scheduling in Multiprocessors," ACM Trans. on Computer Systems, vol. 16, no. 4, Nov. 1998.

[9] H. Zhang, "Service Disciplines for Guaranteed Performance Service in Packet-Switching Networks," Proc. of IEEE, vol. 83, no. 10, Oct. 1995.

[10] P. Goyal and H. M. Vin, "Generalized Guaranteed Rate Scheduling Algorithms: A Framework," IEEE/ACM Trans. on Networking, vol. 5, no. 4, Aug. 1997.