독자-필자 문제를 위한 카운터 기반의 적응적 우선 스케줄링 정책

강 성 일† · 이 흥 규††

요 약

독자-필자 문제(readers-writers problem)를 위한 기존의 스케줄링 정책들은 대부분 독자나 필자 중 어느 한 쪽에 편향된 처리 성향을 가지고 있기 때문에 응답 시간이나 처리량, 둘 중 하나의 성능이 상당히 좋지 않은 특성이 있다. 본 논문에서는 이러한 문제를 개선하고자 새로운 동적 스케줄링 정책인 CAP(Counter-based Adaptive Priority)을 제안한다. CAP은 동시에 수행될 수 없는 필자에게 가중치를 주는 기법과 소수의 독자를 무작정 지연시키는 것을 막기 위한 에이징(aging) 기법을 동적으로 결합하여 사용하고 있다. CAP은 기근(starvation) 문제를 가지고 있지 않으며 주어진 상황에 동적으로 대응하는 능력을 가지고 있기 때문에 처리량과 응답 시간 모두 FCFS에 비하여 더 우수하다. 제안된 정책에 대한 효과성을 입증하기 위하여 세마포어 기반의 해법을 제시하고 사건-기반 시뮬레이션을 사용하여 기존의 정책들과 성능을 비교하였다.

A Counter-based Adaptive Priority Scheduling Policy for the Readers-Writers Problem

Sung-Il Kang† · Heung-Kyu Lee††

ABSTRACT

Most of previous scheduling policies for the readers-writers problem have a problem that they show poor performance in one of throughput or response time because they are biased to one class of readers and writers. In this paper, we propose a new dynamic scheduling policy, Counter-based Adaptive Priority(CAP) to overcome the performance problem. The CAP policy is based on a dynamic combination of *writer-weighting* for non-concurrent writers and *reader-aging* for small waiting readers. CAP does not starve readers or writers and gives better performance than FCFS in both throughput and response time. To show the effectiveness of CAP, we present its semaphore-based solution, and compare the performance of CAP with that of previous policies through the event-driven simulation.

1. 서 론

독자-필자 문제(readers-writers problem)[1]는 운영체제[2]와 데이터베이스[3,10] 분야에서 잘 알려진 전통적인 동기화 문제 중의 하나로써, 최근 들어 널리 쓰이고 있는 다중 쓰레드 프로그래밍(multithreaded programming)[7,8]에서 쉽게 접할 수 있다. 이 문제는 하나의 공유 자료에 독자와 필자가 동시에 접근하고자 할 때 공유 자료의 내용은 하나의 필자에 의해서만 쓰여지도록 하는 것이다. 이것을 위해서 독자와 필자 그리고 필자와 필자는 상호 배제적으로 접근하여야 하지만 다수의 독자는 동시에 접근할 수 있다.

지금까지 독자와 필자의 수행을 조절하는 많은 스

† 준 회 원 : 한국과학기술원 대학원 전산학과
†† 정 회 원 : 한국과학기술원 전산학과 교수
논문접수 : 1997년 10월 21일, 심사완료 : 1998년 9월 22일

케줄링 정책(scheduling policy)이 제시되고 분석되어 왔다. 그 중에서도 필자-우선(writer priority: WP) 정책이 상대적으로 필자의 수가 적다는 가정하에 널리 쓰이고 있으며[7], 이론적으로도 안정 상태(steady-state)에서 큐잉 시간(queuing time)에 대한 최적화 매개 변수 공간(optimal parameter space)이 가장 넓은 것으로 알려져 있다[16]. 그러나 필자-우선의 정책은 독자를 최대한 수집하는 경향이 있기 때문에 처리량(throughput)에서는 상당히 우수하나 독자를 일방적으로 기다리게 함으로써 응답 시간(response time)이 상당히 좋지 않는 특성을 가지고 있다. 그리고 최근에 널리 쓰이고 있는 다중 쓰레드 프로그래밍에서와 같이 독자-필자 문제가 적용되는 많은 실제 환경에서 이론적 분석을 위한 매개 변수를 정확히 알기 힘들 뿐만 아니라 예측하기 어려운 다양한 형태의 도착 패턴(arriving pattern)이 나타나고 있다. 또한, 큐잉 분석적 측면에서 불안정한(unstable) 상태가 될 수 있는 과부하 상태(overloaded state)도 빈번하게 나타난다. 이러한 조건하에서 처리 상황에 관계없이 어느 한 쪽에 우선권을 주거나 고정된 임계치(threshold)를 사용하는 기존의 정책들은 처리 성능이 좋지 않게 된다. 그러므로, 주어진 부하에 동적으로 대응할 수 있는 새로운 정책이 필요하다.

본 논문에서 주어진 부하에 동적으로 대응하는 카운터 기반의 적응적 우선 정책인 CAP(Counter-based Adaptive Priority)을 제안한다. CAP은 바로 전에 동시 수행된 독자의 수를 동시에 수행될 수 없는 필자의 가중치로 사용하고 소수의 독자를 무작정 기다리지 않도록 하기 위한 에이징(aging) 기법을 동적으로 결합하여 사용함으로써 처리 성능을 개선하고 있으며 한쪽을 일방적으로 기다리게 하는 기근(starvation) 문제도 해결하고 있다. 제안된 정책의 효과성을 입증하기 위하여 세마포어 기반의 구현 해법을 제시하고 사건-기반 시뮬레이션(event-driven simulation)으로 기존 정책들과 성능을 비교한다. 시뮬레이션을 사용하여 성능을 비교하는 이유는 동적 스케줄링 정책에 대한 이론적인 큐잉 분석은 상당한 어려운 문제이기 때문이다[12,14].

논문의 나머지 부분은 다음과 같이 구성되어 있다. 먼저 2장에서 기존의 스케줄링 정책에 대한 특성을 분석하고, 3장에서 새로이 제안된 카운터 기반의 스케줄링 정책인 CAP의 동작 원리를 설명하고 세마포어(semaphore) 기반의 구현 해법을 제시한다. 4장에서 CAP의 효과성을 입증하기 위하여 다른 정책들과 함께 실시한 사건-기반 시뮬레이션의 결과를 처리량과 응답 시간을 기준으로 성능을 비교한 다음 응답 시간의 구성 비율을 가지고 처리 시간의 배분에 대한 공평성(fairness)도 비교한다. 마지막으로 5장에서 연구 결과의 요약과 함께 결론을 기술한다.

2. 기존의 스케줄링 정책

독자-필자 문제를 위한 스케줄링 정책의 성능 척도(performance measure)로 처리량(throughput)과 응답 시간(response time)이 고려될 수 있다. 이 둘은 서로 대치(trade-off) 관계에 있기 때문에 두 가지 성능 모두를 동시에 개선하는 것은 쉽지 않다. 이 두 가지 성능 척도와 독자나 필자 중 어느 한쪽이 일방적으로 기다려야 하는 기근(starvation) 문제를 가지고 기존 정책들의 특성을 분석한다.

먼저, 가장 공평한(fair) 정책으로 볼 수 있는 FCFS (First-Come First-Served)[12]는 독자와 필자를 도착하는 순서대로 대기시키고 있다가 그 순서대로 수행시키는데 연이어 대기하고 있는 독자는 동시에 수행시킬 수 있다. FCFS는 시스템 내에 다수의 독자가 있더라도 연이어져 도착하지 않는다면 동시에 수행될 수 없기 때문에 대기하고 있는 독자들을 한꺼번에 수행시킬 수 있는 다른 스케줄링 정책들보다 처리량이 뒤떨어진다. 그렇지만 도착하는 순서대로 처리하기 때문에 부하가 적은 경우 응답 시간이 상당히 좋다.

독자 우선권을 갖는 RP(Reader Priority)[1,2]는 시스템에 독자가 있는 경우 계속 독자를 수행시키는 것이며 이와 반대로, 필자 우선권의 WP(Writer Priority)[1,7]는 필자가 있는 경우 계속 필자를 수행시키는 것이다. RP와 WP 모두 한 쪽을 무작정 기다리게 하는 이론적인 기근(starvation) 문제를 가지고 있다. 성능 측면에서 볼 때, RP는 동시에 처리될 수 있는 독자의 수가 적기 때문에 처리량은 떨어지나 빠른 시간 내에 독자를 수행시키기 때문에 응답 시간은 우수하다. 반면, WP는 독자를 최대한 수집하고 있다가 동시에 처리하기 때문에 처리량은 우수하나 다수의 독자가 순차적으로 수행되어야 하는 모든 필자를 기다려야 하기 때문에 응답 시간이 좋지 않다. 이들을 변형한 다른 정책들의 성능은 대개 이 두 가지 정책들이 갖는 성능 사이에 존재하게 된다.

교대로 수행하는 ALT(Alternating)[5]는 RP와 WP의 기근 문제를 해결하기 위하여 하나의 필자와 다수의 독자를 교대로 수행시키는 것이다. 이 정책에서 기근 문제는 해결될 수 있지만 독자의 수가 적은 경우 처리량이 떨어진다. 최악의 경우, 독자와 필자가 하나씩 교대로 수행될 수도 있다. ALT의 변형으로 볼 수 있는 교대로 소진 수행하는 AEP(Alternating Exhaustive Priority)[14]는 독자가 수행되면 독자가 없을 때까지, 필자가 수행되면 필자가 없을 때까지 해당 종류의 고객을 계속 수행시키는 수행 모드(run mode)를 가지고 있다. 독자는 한꺼번에 소진될 수 있고 독자에 비해 필자의 수행 시간이 상대적으로 길기 때문에 필자 수행 모드에서 오래 머무를 확률이 높아져 AEP는 WP와 유사한 성격을 띄게 된다. 수행 모드와 시스템 내에 대기하는 필자의 수를 임계치로 사용하는 TFE (Threshold Fastest Emptying)[12]는 대기하고 있는 필자의 수가 주어진 임계치 K에 도달하게 되면 필자 수행 모드로 전환되는데 AEP와 같이 일단 필자가 수행되면 필자가 없을 때까지 수행한다. TFE에서 $K=1$이면 필자가 하나라도 있으면 독자가 수행될 수 없으므로 WP와 거의 동일한 것이 되며, $K\rightarrow\infty$이면 독자가 없을 때만 필자가 수행될 수 있으므로 RP와 동일한 것이 된다. 그러나 임의의 K에 대하여 부하가 커지는 경우 일단 필자 수행 모드로 바뀌면 순차적인 필자를 계속해서 수행할 가능성이 높기 때문에 AEP와 같이 필자 성향을 띄게 된다.

다음은 확률적 선택 개념을 사용하는 정책들이다. 먼저 가장 손쉽게 적용할 수 있는 RAND(Random)는 어느 종류가 나가든지 관계없이 대기하고 있는 독자와 필자 중에서 한 쪽을 임의로 선택하는 것으로 ALT와 유사한 특성을 갖는다. 독자는 동시에 수행될 수 있기 때문에 동일한 선택 조건을 가진다면 잠재적으로 필자가 불리하게 된다. WRP(Weak Reader Priority)[4]와 WWP(Weak Writer Priority)[4]는 절대 우선권을 갖는 RP와 WP를 약화시킨 것으로 우선권이 주어지는 것과 다른 종류가 나가는 경우에만, 두 종류 중 하나를 똑같은 확률로 임의 선택하는 것이다. WRP와 WWP는 RAND와 같이 확률의 특성상 기근 문제를 어느 정도 해결할 수 있으나 처리 성향이 여전히 절대 우선권을 주는 것과 유사하기 때문에 성능 측면에서 별 차이가 없게 된다.

위의 분석 결과를 요약하면 (표 1)과 같다. 처리 성향과 기근 항목의 R과 W는 독자(Reader)와 필자(Writer)를 나타내고 기근 항목에서 RW인 경우는 독자와 필자 모두에 기근이 발생할 수 있음을 나타낸다. 처리 성향이 상황에 따라 바뀔 수 있는 지를 나타내는 동작 형태의 S와 D는 정적(Static)과 동적(Dynamic)을 나타낸다. (P)는 확률적 특성을 사용하고 있음을 나타낸다. 큐잉 분석 항목에서 ○는 분석, △는 근사적인 분석, □는 조건부 분석이 이루어졌음을 나타내고 ×는 이론적으로 분석되지 않았음을 나타낸다.

〈표 1〉 스케줄링 정책의 특성
〈Table 1〉 Features of scheduling policies

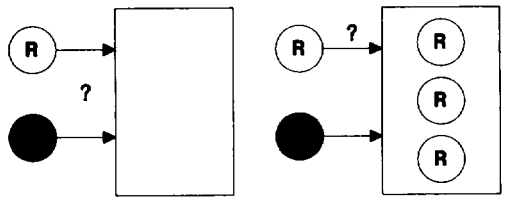
정책	처리성향	기근	동작형태	큐잉분석	참고문헌
FCFS	-	-	S	△	[13]
RP	R	W	S	○	[1]
WP	W	R	S	○	[1]
ALT	R	-	S	×	[5]
AEP	W	RW	S	○	[17]
TFE(K)	W	RW	S	□	[15]
RAND	R	-	S(P)	×	
WRP	R	-	S(P)	×	[4]
WWP	W	-	S(P)	×	[4]

이와 같이 기존의 정책들은 기근 문제를 가지거나 주어진 부하와 도착 패턴에 무관하게 처리 성향이 한 쪽으로 치우쳐 있는 정적인 특성을 가지고 있기 때문에 주어진 부하에 동적으로 반응할 수 없는 문제를 가지고 있다. 임계치를 사용하는 TFE 경우에도 필자 성향이 강하며 임계치를 동적으로 조정할 수 없다는 문제점과 수행 모드에 의한 소진 수행으로 인한 기근 문제도 동시에 가지고 있다.

3. 카운터 기반의 적응적 스케줄링 정책(CAP)

3.1. 동작 원리

스케줄링 입장에서 보면, 독자-필자 문제는 (그림 1)과 같이 독자가 수행되고 있는 상태에서 도착하는 독자를 어떻게 진입시킬 것인 지와 수행 중인 것이 없는 상태에서 대기하고 독자와 필자 중에서 어느 쪽을 어떻게 선택할 것인지에 대한 두 가지 스케줄링 항목(scheduling item)으로 나누어 생각해 볼 수 있다[3].



R: Reader, W: Writer　　R: Reader, W: Writer

(a) 독자 진입 결정 방법　　(b) 선택 방법

(그림 1) 독자-필자 문제의 두 가지 스케줄링 항목
(Fig. 1) Two scheduling items of the readers-writers problem

먼저 한 쪽이 일방적으로 기다리는 기근 문제를 해결하기 위해서는 (a)에서 도착하는 독자를 계속 진입시키지 않아야 하며 (b)의 독자와 필자를 선택하는 방법에 있어서도 한쪽을 계속해서 선택할 수 있는 가능성을 배제하여야 한다. 주어진 부하에 동적으로 대응해야 하는 문제는 독자의 진입과 독자와 필자의 선택 항목에서 주어진 정보에 입각하여 적절히 수행하는 문제로 바꾸어 생각해 볼 수 있다.

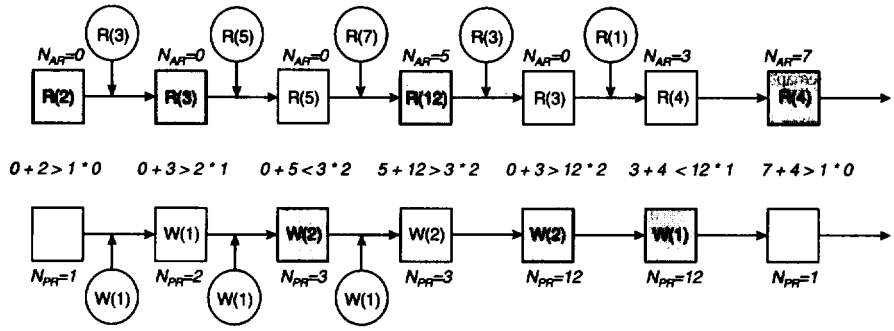
독자-필자 문제에서 풀어야 할 두 가지 숙제인 기근과 동적 대응 문제를 해결하는 새로운 CAP(Counter-based Adaptive Priority) 정책이 만들어진 과정을 설명하면 다음과 같다. 주어진 부하와 도착 패턴에 기초하여 독자와 필자를 적절히 선택하는 문제는 수행 비중이 어느 쪽이 더 큰 가를 결정하는 문제로 해석할 수 있다. 그러면 현재의 수행 비중을 계산하는 방법을 찾아야 하는데 독자와 필자의 수행 시간을 알 수 없기 때문에 현재 알 수 있는 정보인 대기하고 있는 독자와 필자의 수를 가지고 판단해야 한다. 그런데 독자는 동시에 수행될 수 있기 때문에 독자와 필자가 동일한 비중을 가지고 있다고 볼 수 없다. 독자에 대한 필자의 적절한 상대적 비중을 찾기 위해서는 독자의 동시 수행의 정도(degree of reader concurrency)를 알 수 있어야 하지만 계속 변화하는 상황에서 그 값을 정확히 알 수 없다. 따라서, 수행 시간과 동시 수행의 정도를 잘 나타내주는 근사값을 찾아야 한다.

계속 변화하는 부하를 예측할 수 없다는 가정하에서는 가장 최근에 수행된 결과가 현재의 상태를 가장 잘 반영한다고 볼 수 있다. 즉, 동시 수행되는 독자의 수가 많다는 것은 현재의 부하가 그 만큼 크다는 것을 나타내는 것으로 필자의 수행 시간이 길거나 많은 수의 독자가 도착하고 있음을 복합적으로 반영하고 있다. 그러므로 동시 수행된 독자의 수는 상대적으로 필자보다 독자의 수가 많은 상황에서 비중을 계산하는 적절한 척도가 될 수 있다. 이러한 가정하에 이전에 동시 수행된 독자의 수(N_{PR})를 독자에 대한 필자의 가중치로 사용한다. 즉, 이 가중치를 현재 대기하고 있는 필자의 수(N_{WW})와 곱한 후 현재 대기하고 있는 독자의 수(N_{WR})와 비교하여 더 큰 쪽을 선택하는 것이다. 이렇게 되면 일련의 독자 그룹(reader group)이 연속적으로 선택되기 위해서는 이전 보다 더 많은 독자가 대기하고 있어야 하고 그렇지 않으면 필자가 수행되므로 주어진 부하에 맞게 반응하게 된다. 필자가 수행될 수 없으면 가중치가 계속 증가하므로 필자의 기근 문제도 해결된다. 만약 대기하고 있는 필자가 없다면 가중치는 의미가 없어지고 새로운 도착 패턴을 형성할 것이므로 가중치, N_{PR}은 초기 상태인 1로 재설정된다.

독자의 동시 수행의 정도를 가중치로 사용하는 배경은 필자에 비해 독자의 수가 많다는 것이다. 그러나 만약 필자에 비해 독자의 수가 적은 상황이 되면 상대적으로 가중치가 낮은 독자는 수행이 지연되기 때문에 전체적으로 응답 시간이 나쁘게 된다. 이를 극복할 방법이 필요한데 가장 간단한 방법은 수행되지 못하고 기다리고 있는 독자를 에이징(aging)시키는 것이다. 구체적으로 에이징은 독자의 기근 문제를 없애고 반응 시간을 개선하기 위하여 독자가 계속된 필자의 수행을 기다린 횟수 만큼 독자의 에이지(N_{AR})를 반복적으로 증가시키는 방법을 사용한다. 즉, 현재의 에이지(N_{AR}) = 이전의 에이지(N_{AR}') + 이전의 독자 대기수(N_{WR}')가 된다. 만약 다시 독자가 선택되면 기다리는 독자가 없게 되므로 에이지, N_{AR}은 초기값 0으로 재설정된다. 앞에서 설명한 4개의 변수를 사용하여 필자에 대한 독자의 상대적 비중(C_{RW})을 계산하는 방법은 다음과 같다.

$$C_{RW} = \frac{\text{독자의 에이지}(N_{AR}) + \text{현재 대기하고 있는 독자의 수}(N_{WR})}{\text{이전에 동시 수행된 독자의 수}(N_{PR}) * \text{현재 대기하고 있는 필자의 수}(N_{WW})}$$

$C_{RW} > 1$ 이면 독자 그룹이 수행되고 $C_{RW} < 1$이면 하나의 필자가 수행된다. $C_{RW} = 1$인 경우에도 필자를 선택하는 것이 타당하다. 그 이유를 설명하면 다음과 같다. $C_{RW} = 1$인 경우는 필자의 이탈 시점에서 $N_{PR} = 1$, $N_{AR} = 0$이 되고 독자와 필자 각각 하나만 대기하고 있을 경우에 주로 일어난다. 다시 말해, 처리 형태가 W • • • W W R로 되어 W 이전 까지는 주로 필자가



(그림 2) CAP의 스케줄링 과정
(Fig. 2) A scheduling pathpath of CAP

수행되어 가중치 N_{PR}은 1이 되고 독자가 마지막 필자보다 늦게 도착했을 때 발생한다. 그러므로, 필자를 먼저 수행시켜 필자의 응답 시간을 줄이고 추가로 다수의 독자를 수집할 기회를 가질 수 있게 되므로 필자를 선택하는 것이 타당하다. N_{PR} = 1, N_{AR} = 0이 아닌 경우에도 잠재적으로 독자에 비해 필자가 불리하기 때문에 필자에게 우선권을 주는 것이 바람직하다.

(그림 2)은 일련의 독자와 필자가 도착하는 상태에서 CAP이 이들을 스케줄링하는 과정을 보여주고 있다. 그림상의 표현을 간단히 하기 위하여 고객은 동시에 진입하고 이탈하는 것으로 가정한다. 사각형 내의 값은 해당 종류의 고객이 대기하는 숫자이고 회색으로 칠해진 것은 해당 시점에서 수행된 것을 나타낸다. 단, 대기하고 있던 독자는 모두 동시에 수행되며 필자는 한번에 하나씩 수행된다. 이러한 조건하에서 (그림 2)의 수행 결과는 R(2) R(3) W(1) R(12) W(1) W(1) R(4)가 된다. 그런데 실제 고객은 동시에 진입하고 이탈하지 않을 수 있기 때문에 구현 해법은 고객의 개별적 도착과 이탈에 대해서 처리할 수 있도록 작성되어야 한다. 이와 관련된 CAP 정책에 대한 자세한 구현 해법은 다음 절에서 설명한다

3.2. 세마포어 기반의 해법

제안된 CAP 정책을 6개의 공유 변수와 상호 배제(mutual exclusion)를 위한 하나의 이진 세마포어[2], 그리고 2개의 카운팅 세마포어를 사용하여 구현한 C유사 구문의 해법[1,5]이 (그림 3)에 나타나 있다. 변수 ER과 EW는 각각 수행 중인 독자와 필자의 수를 나타내고 WR과 WW는 각각 기다리고 있는 독자와 필자의 수를 나타낸다. 이 4개의 변수의 초기값은 0이다. 필요하다면 [1]의 논문에서와 같이 AW는 음수를 사용하는 것으로 하여 ER과 EW를 하나의 변수로 줄일 수 있지만 편의상 구분하기로 한다. 독자에 대한 필자의 가중치로 사용되는 변수 PR은 이전에 동시 수행된 독자의 수를 나타낸다. PR은 1이상의 값을 가지며 초기값은 1이다. AR은 수행되지 못하고 기다리는 독자를 보상해 주기 위한 독자의 에이지를 나타내는 것으로 초기값은 0이다. 이진 세마포어 Mutex는 앞에서 정의한 6개의 공유 변수를 보호하기 위한 것이며 카운팅 세마포어 Reader와 Writer는 각각 독자와 필자를 대기시키기 위한 것으로 FIFO 순서를 유지할 수 있는 것으로 가정한다.

실제 스케줄링이 일어나는 독자와 필자의 진입과 이탈시 수행되는 4개의 프로시져, enter_reader(), enter_writer(), exit_reader(), exit_writer(), 각각에 대해 구체적으로 살펴보자. enter_reader()에서 필자에 기근이 발생하지 않도록 도착하는 독자는 필자가 수행되고 있거나 기다리고 있으면 대기하게 된다. 만약 필자가 없다면 독자는 진입하게 되고 필자를 위해 PR을 증가시킨다. enter_writer()에서 도착하는 필자는 시스템에 고객이 없을 경우에만 수행되고 그 이외에는 대기하게 된다. 독자와 필자가 시스템을 이탈하는 exit_reader()와 exit_writer()에서 WR, WW, PR, AR를 사용하여 상대적 비중을 계산한 후 비중이 높은 쪽을 수행시킨다. 만약 독자가 선택된다면 대기하고 있던 모든 독자는 동시에 수행되고 그 수가 PR에 설정된다. 만약 필자가 선택된다면 대기하던 필자 중 가장 먼저 도착한

```
int ER = EW = WR = WW = 0;
int PR = 1, AR = 0;
semaphore Mutex=1, Reader=0, Writer=0

enter_reader()
{
  P(Mutex);
  /* only if no writers */
  if (EW + WW == 0) {
   ER++;
   PR++;
   V(Mutex);
  } else {
   WR++;
   V(Mutex);
   P(Reader);
  }
} /* end of enter_reader */

exit_reader()
{
  P(Mutex);
  ER--;
  if (PR * WW < AR + WR) {
   if (WW > 0) PR = PR + WR;
   /* wake up all readers */
   while (WR > 0) {
     WR--;
     ER++;
     V(Reader);
   }
  } else {
   /* only if last reader */
   if (ER == 0 && WW > 0) {
     if (WR > 0) AR = WR;
     /* wake up a writer */
     EW++;
     WW--;
     if (WW == 0) PR = 1;
     V(Writer);
   }
  }
  V(mutex);
} /* end of exit_reader */

enter_writer()
{
  P(Mutex);
  /* only if no one */
  if (ER + EW + WR + WW == 0) {
   EW++;
   V(Mutex);
  } else {
   WW++;
   V(Mutex);
   P(Writer);
  }
  V(Mutex);
} /* end of enter_writer */

exit_writer()
{
  P(mutex);
  EW--;
  if (PR * WW < AR + WR) {
   AR = 0;
   if (WW > 0) PR = WR;
   /* wake up all readers */
   while (WR > 0) {
     WR--;
     ER++;
     V(Reader);
   }
  } else {
   if (WW > 0) {
     if (WR > 0) AR = AR + WR;
     /* wake up a writer */
     EW++;
     WW--;
     if (WW == 0) PR = 1;
     V(Writer);
   }
  }
  V(Mutex);
} /* end of exit_reader */
```

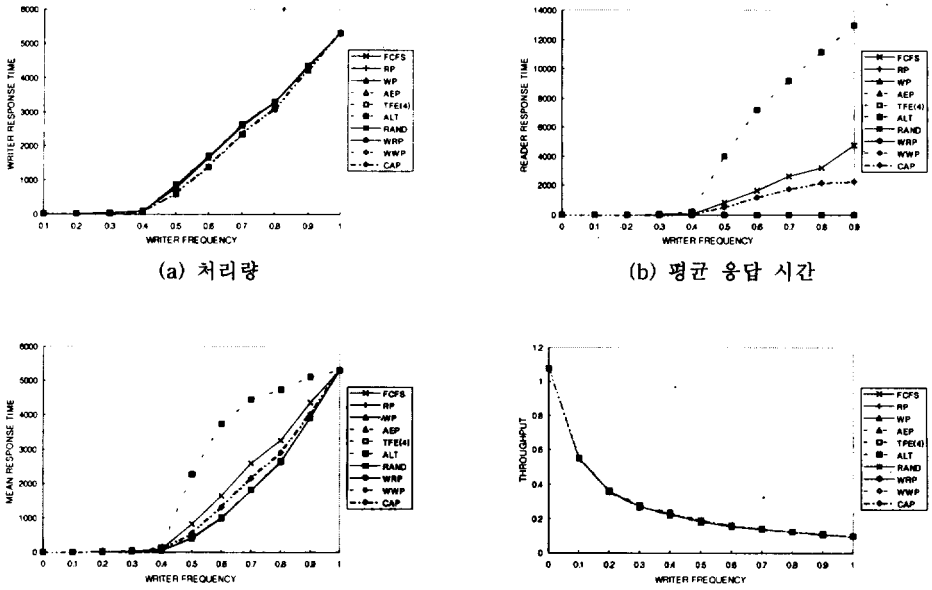
(그림 3) CAP 스케줄링 정책의 세마포어 기반의 해법
(Fig. 3) A semaphore-based solution of CAP

필자가 수행되고 더 이상의 필자가 없으면 PR은 초기값 1로 재설정된다. exit_reader()에서 마지막 독자가 나가는 경우에만 스케줄링 되도록 구현할 수 있지만 마지막이 아니더라도 추가로 독자가 수행될 수 있도록 하는 것이 성능 개선에 더 유리하다. 왜냐하면 일단의 독자 그룹이 수행되는 동안 추가로 도착한 독자에 비해 필자의 수가 많지 않다면 독자를 추가로 수행시키는 것이 더 바람직하다. 또한, 상대적 비중을 비교할 수 있는 기회가 증가하여 좀 더 정확하고 신속하게 부하에 대응할 수 있게 된다. 결과적으로 처리량과 응답시간 모두 부분적으로 개선할 수 있게 된다. 이 때, 추가로 진입한 독자의 수 만큼 PR을 증가시킨다. 비중 계산에서 필자가 선택되는 경우에 마지막 독자가 나가는 경우가 아니면 필자가 수행되지 않도록 하는 것에 주의해야 한다. 그렇지 않으면 독자-필자 문제의 처리 조건에 위배될 수 있다. 제시된 해법은 독자-필자 문제에 대한 일반적인 세마포어 구현 해법[1,5,6]에서 간단히 두 개의 카운터, PR과 AR만을 추가함으로써 손쉽게 구현될 수 있으며 관련 연산이 간단하기 때문에 스케줄링 오버헤드도 크지 않다.

4. 성능 평가

각 스케줄링 정책의 성능 평가를 위한 실험은 Simpack[9]을 이용하여 (그림 3)의 4개의 프로시져를 사건(event)으로 정의한 사건-기반 시뮬레이션으로 수행되었다. 도착 과정은 독자와 필자의 평균 도착율(arrival rate)이 λ_R과 λ_W인 포아송 분포(Poisson distribution)를 하고, 수행 시간(service time)은 각각의 평균이 $1/\mu_R$과 $1/\mu_W$ 기하 분포(exponential distribution)를 하는 것으로 가정하였다. 여러 가지 도착 패턴을 가정하기 위하여 독자의 도착 수(A_R)와 필자의 도착 수(A_W)를 합친 총 도착 수(A_T)가 2000개이고 이것이 모두 처리될 때까지 실시하였다. 성능 척도(performance measure)로 처리량(throughput)과 평균 대기 시간(mean response time), 그리고 상대적 공평성(relative fairness)을 사용하였고, 세부적인 분석을 위하여 독자와 필자 각각의 평균 응답 시간도 함께 비교하였다. TFE(K)의 모든 임계치 K에 대하여 성능을 비교할 수 없기 때문에 임의로 선정한 K=4를 가지고 실험하였다.

성능 평가를 위한 실험은 크게 4부분으로 구성되어 있다. 첫 번째 실험은 다양한 형태의 도착 패턴에 대한 성능을 비교하기 위하여 독자와 필자의 구성 비율을 변화시켜 나가면서 독자의 평균 수행 시간이 필자에 비해 짧은 경우, 같은 경우, 긴 경우에 대하여 수행한 것이다. 독자의 수행 시간이 긴 경우는 동시에 처리되는 독자의 수에 비례하여 시스템의 부하가 증가하는 경우와 공유되는 자료가 복잡한 구조를 가지고 있는 경우에 주로 발생한다. 두 번째 실험은 부하의 증가에 따른 성능을 알아보기 위하여 독자와 필자의 수행 시간을 고정시킨 상태에서 독자와 필자의 도착율을 합친 총 도착율(λ_T)을 비례적으로 증가시켜 나가면서 수행한 것이다. 세 번째 실험은 버스트 현상을 갖는 도착 패턴에 대한 반응을 분석하기 위하여 도착 간격을 초기하 분포(hyper-exponential distribution)로 설정하고 분산 계수를 증가시키면서 실험을 한 것이다. 마지막 네 번째 실험은 각 스케줄링 정책의 성능 향상 정도와 상대적 공평성을 비교하기 위하여 독자와 필자의 구성 비율을 10:1로 설정하고 3가지 부하에 대하여 수행한 것이다. 다음은 각 실험의 결과를 분석한 것이다.



(a) 처리량 (b) 평균 응답 시간

(c) 독자 응답 시간 (d) 필자 응답 시간

(그림 4) 독자와 필자의 구성 비율에 따른 성능
(Fig. 4) Performance on the component ratio of readers and writers
[A_T=2000, λ_T =0.2, λ_R=0.2~0.0, λ_W=0.0~0.2, μ_R=1.0, μ_W=0.1]

(그림 4)는 독자의 수행 시간이 짧은 경우로 독자와 필자의 수행 시간을 각각 $1/\mu_R$=1, $1/\mu_W$ =10으로 고정시켜 놓고 독자와 필자의 구성비율을 조정하면서 성능을 측정한 것이다. 동시 수행될 수 있는 독자의 수에 제한이 없는 경우 필자의 수행 시간이 전체 부하의 대부분을 차지하기 때문에 처리량에는 거의 차이가 나지 않고 있다. 필자에 의한 부하, $\rho_W=\lambda_W/\mu_W=1$인 $\lambda_R=0.1$, $\lambda_W=0.11$에서부터 대기 현상이 심화되면서 응답 시간에 많은 차이를 나타내고 있으며 필자 성향의 정책들의 독자의 응답 시간이 급격히 나빠지고 있다. 평균 응답 시간과 독자의 응답 시간에서 FCFS를 기준으로 독자 성향의 정책과 필자 성향의 정책으로 명확히 구분되고 있다. 이와 같이 대기 현상이 독자에 의해 발생되지 않고 주로 필자에 의해 발생되는 경우는 독자 성향의 정책들이 전체적으로 좋은 성능을 보이고 있다. CAP은 RP와 같이 독자 성향을 보이고 있으며 처리량과 응답 시간이 모두 FCFS보다 우수하다.

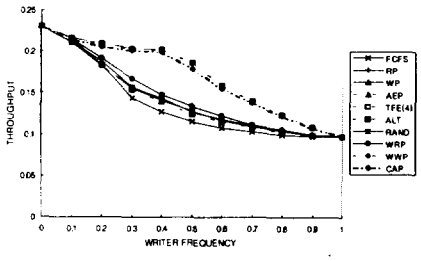
(그림 5)는 독자와 필자의 수행 시간을 동일하게 $1/\mu_R$= 10, $1/\mu_W$=10으로 가정한 실험의 결과이다. 처리량에서 독자 성향의 정책과 필자 성향의 정책 사이의 성능이 뚜렷이 구분되고 있다. 이것은 독자의 수행이 전체 부하에 어느 정도 영향을 주기 때문이다. 독자와 필자의 수행 시간이 동일한 경우 FCFS, ALT, RAND가 유사한 성능을 보이고 있다. 필자 성향의 정책들은 필자 구성 비율이 0.4인 점에서부터 대기 현상이 심화되면서 응답 시간이 급격히 나빠지고 있다. CAP은 처리량에서는 WP, WWP, TFE와 같이 필자 성향의 정책들의 성능에 근접하고 있으며 응답 시간에서는 RP에 근접한 성능을 보이고 있다. 또한, CAP은 갑작스러운 상승이나 하강 없이 안정된 성능을 보이고 있으며 처리량과 응답 시간 모두에서 FCFS보다 더 나은 성능을 보이고 있다.

(그림 6)은 독자의 수행 시간이 필자의 수행 시간보다 더 긴 경우에 대한 결과로서 정책들 간에 처리량과 응답 시간, 둘 다에서 많은 차이를 보이고 있다. 즉, 독자와 필자 모두가 전체 부하에 큰 영향을 미치고 있음을 알 수 있다. 이와 같이 독자의 수행 시간이 필자에 비해 긴 경우는 동시에 수행될 수 있는 독자의 수가 제한된 경우에 많이 나타나는데 일정 수의 독자가 순차적으로 수행되어 평균 수행 시간이 길어지는 것과 같은 효과를 나타내기 때문이다[14]. 예상했던 바와 같이 대기하고 있던 독자를 모두 수행시킬 수 없는 FCFS가 처리량과 응답 시간 모두에서 가장 좋지 않는 성능을 보이고 있다. 처리량에서 TFE(4)가 WP보다 부분적으로 더 나은 성능을 보이고 있다. CAP은 처리량과 응답 시간 모두에서 우수한 성능을 보이고 있으며 갑작스러운 변화 없이 주어진 부하와 도착 패턴에 안정적으로 반응하고 있다.

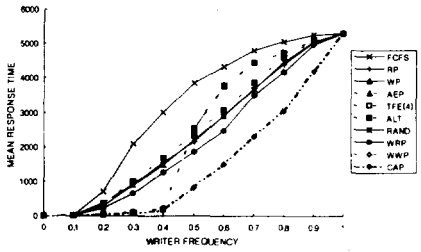
(그림 7)은 고 부하 상태에서의 성능을 확인하기 위한 것으로 독자의 도착율보다 필자의 도착율이 10배 정도 큰 상태에서 총 도착율을 0.11에서 1.21로 증가시키며 실험한 결과이다. 처리량에서 FCFS는 총 도착율이 0.33이 되는 시점에서 부터 한계에 도달하고 있으며 WP, WWP, TFE(4), 그리고 CAP은 1.1부터 한계에 도달하고 있다. 대기 현상이 심화되는 0.55에서 수행 모드에 기초한 AEP는 상대적으로 독자의 수가 필자보다 많기 때문에 독자 수행 모드에 수행될 확률이 높아져 처리 성향이 필자에서 독자로 바뀌고 있다. 대기하고 있는 모든 독자를 동시에 수행시킬 수 없는 FCFS가 처리량과 응답 시간에서 가장 좋지 않는 성능을 보이고 있다. CAP의 처리량은 필자 성향의 정책들처럼 부하에 비례하여 증가하고 있으며 응답 시간에서도 독자 성향을 보이는 정책들과 같이 상승폭이 상당히 적게 나타나고 있다. 즉, CAP은 부하의 증가에도 안정적으로 동작하고 있음을 알 수 있다.

분산 계수, 즉, 버스트 현상에 대한 처리 성능을 보여주는 (그림 8, 9, 10)에서 버스트 현상이 커짐에 따라 다수의 독자를 동시에 수행할 기회가 커져서 전체적으로 처리량은 증가하고 응답 시간은 줄어들고 있다. (그림 9)에서 극단적인 모드 수행을 하는 AEP는 버스트 현상에 대한 불안정한 모습을 나타내고 있다. (그림 10)에서와 같이 버스트 현상이 심하고 부하가 적은 경우는 독자 성향의 정책이 오히려 FCFS보다 좋지 않은 처리량을 보이고 있다. 개념적으로 볼 때, CAP은 버스트 현상이 심하면 이전의 수행된 정보의 정확성이 떨어지기 때문에 성능이 좋지 않아야 하지만 에이징으로 이를 보상해 주기 때문에 안정적인 성능을 보여 주고 있다.

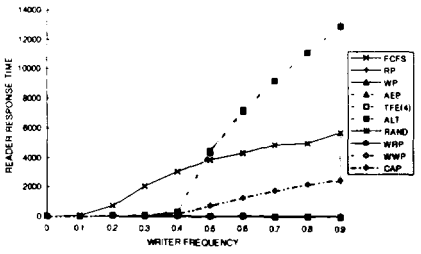
독자-필자 문제에는 두 종류의 고객이 있기 때문에 처리 성능 뿐만 아니라 스케줄링 정책이 이 두 종류의



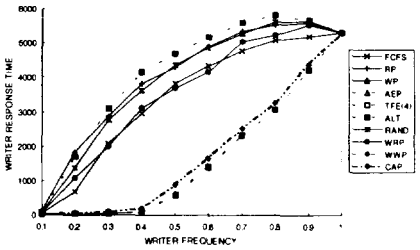
(a) 처리량



(b) 평균 응답 시간



(c) 독자 응답 시간

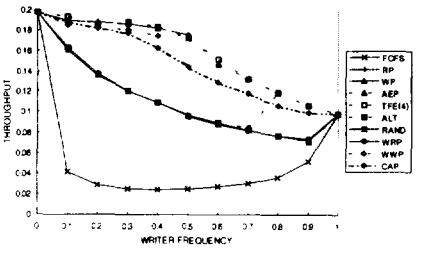


(d) 필자 응답 시간

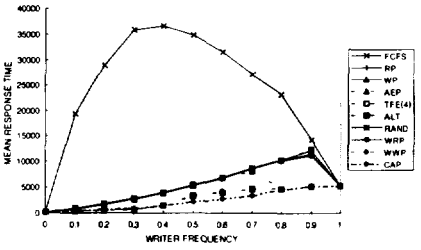
(그림 5) 독자와 필자의 구성 비율에 따른 성능

(Fig. 5) Performance on the component ratio of readers and writers

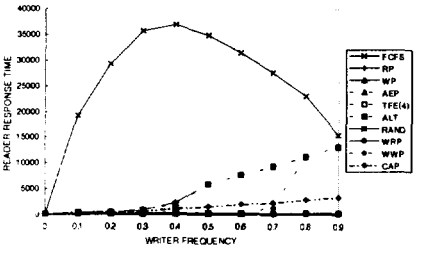
[A_T=2000, λ_T =0.2, λ_R=0.2~0.0, λ_W=0.0~0.2, μ_R=0.1, μ_W=0.1]



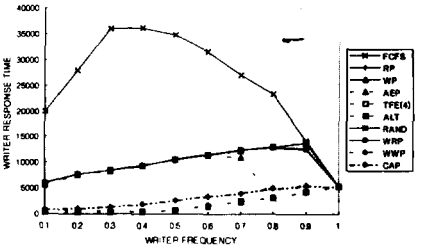
(a) 처리량



(b) 평균 응답 시간



(c) 독자 응답 시간

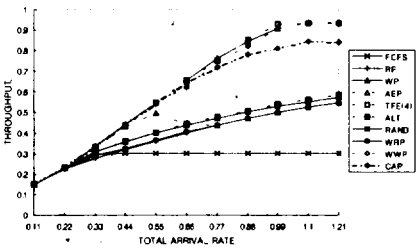


(d) 필자 응답 시간

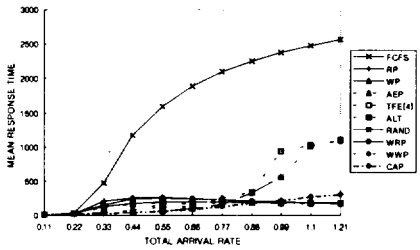
(그림 6) 독자와 필자의 구성 비율에 따른 성능

(Fig. 6) Performance on the component ratio of readers and writers

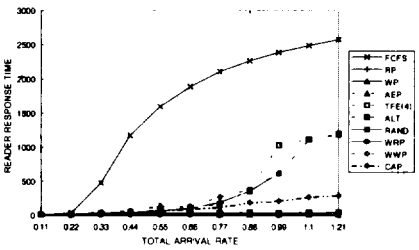
[A_T=2000, λ_T =0.2, λ_R=0.2~0.0, λ_W=0.0~0.2, μ_R=0.01, μ_W=0.1]



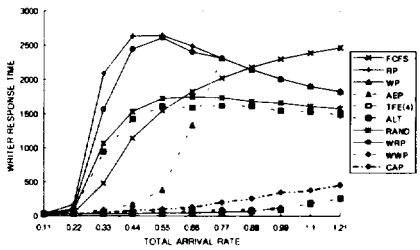
(a) 처리량



(b) 평균 응답 시간



(c) 독자 응답 시간

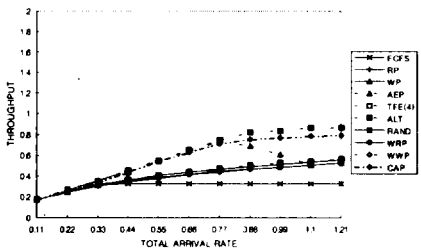


(d) 필자 응답 시간

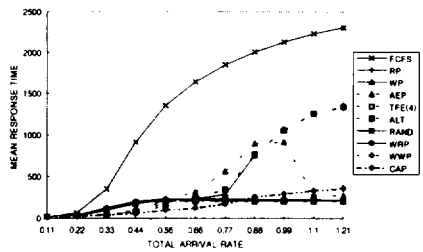
(그림 7) 부하의 증가에 따른 성능

(Fig. 7) Performance for the increasing load

[A_T=2000, λ_T =0.11~1.21, λ_R=0.1~1.1, λ_W=0.01~0.11, μ_R=1.0, μ_W=0.1]



(a) 처리량

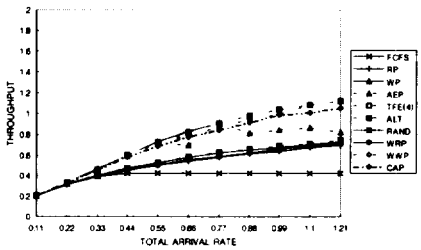


(b) 평균 응답 시간

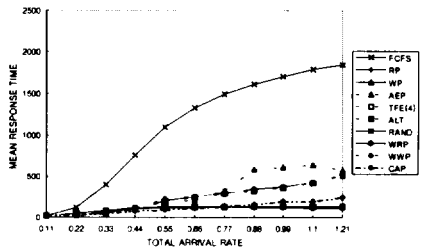
(그림 8) 버스트를 갖는 부하의 증가에 따른 성능

(Fig. 8) Performance for the increasing bursty load

[A_T=2000, λ_T =0.11~1.21, CV=2, μ_R=1.0, μ_W=0.1]



(a) 처리량



(b) 평균 응답 시간

(그림 9) 버스트를 갖는 부하의 증가에 따른 성능

(Fig. 9) Performance for the increasing bursty load

[A_T=2000, λ_T =0.11~1.21, CV=4, μ_R=1.0, μ_W=0.1]

고객을 공평하게 처리하고 있는 지에 대해서도 비교할 필요가 있다. 그런데 독자와 필자의 수행 특성이 서로 다르기 때문에 도착 패턴이나 부하에 관계없이 공평성(fairness)을 측정할 절대 기준을 가질 수 없다. 따라서, 상대적 기준을 사용할 수 밖에 없는데, 본 논문에서는 처리 방법이 가장 공평한 FCFS를 기준으로 처리 시간의 배분에 대한 **상대적 공평성(relative fairness)**을 비교하였다. 상대적 공평성은 어떤 도착 패턴에 대하여 FCFS가 독자와 필자에 배분하는 처리 시간의 비율에 가장 근접하는 정책이 가장 공평하다고 판단하는 것이다. 공평성에 대한 비교는 이전의 연구에서는 없던 것으로 본 논문에서 처음으로 수행된 것이다.

(그림 11, 12, 13)은 필자의 수행 시간이 독자에 비해 10배 길고 독자의 수가 10배 많은 일반적인 경우를 가정하고 전체의 부하를 증가시키면서 FCFS를 기준으로 한 (a)처리량과 응답 시간의 성능 향상과 (b)독자와 필자의 응답 시간 구성 비율로 처리 시간의 배분에 대한 공평성을 비교한 것이다. 성능 향상 그래프인 (a)에서 음수는 FCFS에 비해 그 만큼 성능이 떨어짐을 나타내고 응답 시간의 구성 그래프인 (b)에서 응답 시간의 구성이 크다는 것은 상대적으로 처리 시간이 적게 배분되고 있음을 나타낸다.

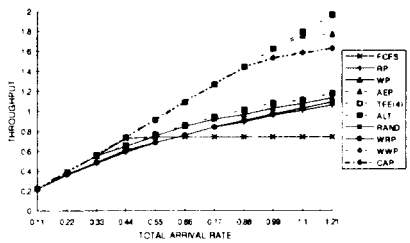
결과 그래프에서 부하가 커짐에 따라 각 정책들의 처리 성향과 성능 개선 정도가 뚜렷이 구분되고 있다. 성능 향상에서 처리량의 작은 차이가 응답 시간에서는 큰 차이로 나타나고 있다. 성능 향상에 대한 비교에서 CAP은 부하와 관계없이 일정 비율로 처리량과 응답 시간 둘 다 개선하고 있다. WP, AEP, TFE와 같이 필자 성향의 정책들은 처리량은 우수하나 응답 시간은 오히려 FCFS에 비하여 상당히 떨어지고 있다. 반면 RP, ALT, RAND, WRP와 같은 독자 성향의 정책들은 응답 시간을 상당히 개선하고 있으나 처리량의 향상은 거의 없으며 (그림 12)의 (a)에서 RP와 ALT는 오히려 FCFS보다 처리량이 더 떨어지고 있다. 이와 같은 응답 시간의 구성에 대한 비교에서 처리 성향이 있는 기존의 정책들은 FCFS에 비하여 처리 시간을 배분하는 데 있어 문제가 많음을 알 수 있다. 즉, 독자 성향의 정책들은 상대적으로 필자를 많이 대기시키고 있고 필자 성향의 정책들은 독자를 많이 대기시키고 있다. CAP은 응답 시간의 구성에 있어서도 FCFS와 비슷한 처리 시간의 배분을 보여 주고 있다. 즉, 다른 정책에 비해 공평성이 더 우수하다고 할 수 있다.

5. 결 론

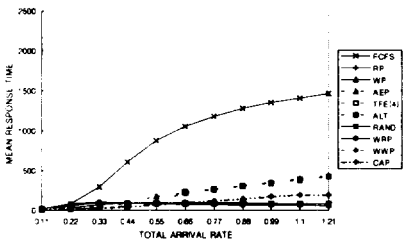
독자-필자 문제를 위한 기존의 스케줄링 정책들은 기초적인 해법을 제시하거나 큐잉 분석이 용이하도록 극단적인 처리 방식을 택하고 있다. 성능 측면에서 볼 때, 일방적으로 한 쪽에만 처리 우선권을 부여하거나 전체적으로 처리 성향이 한쪽으로 치우쳐 있는 기존의 정책들은 부하가 다양한 형태로 계속 변화하는 실제 환경에 적합하지 않다.

본 논문에서는 이러한 문제를 개선하고자 변화하는 부하와 도착 패턴에 동적으로 대응하는 카운터 기반의 적응적 우선권을 갖는 새로운 CAP(Counter-based Adaptive Priority) 정책을 제안하였다. 그 효과성을 입증하기 위하여 기존의 일반 해법에서 두 개의 카운터만을 추가로 사용한 세마포어 기반의 해법을 제시하였으며 시뮬레이션으로 기존의 정책들과 성능을 비교하였다.

시뮬레이션 결과에서 기존의 정책들의 처리 성향을 확인할 수 있었으며, 전반적으로 필자 성향의 정책들은 독자를 최대한 수집하기 때문에 처리량이 우수하고 독자 성향의 정책들은 빠른 시간 내에 독자를 수행시키기 때문에 응답 시간이 우수하게 나타났다. 부하가 적은 경우는 정책에 따른 처리량에는 거의 차이가 없으나 응답 시간에는 다소 차이를 보이고 있다. 부하에 버스트 현상이 심할수록 정책에 따른 처리량에는 더 큰 차이를 나타내지만 응답 시간에서의 차이는 작아지는 것으로 나타났다. CAP은 그 처리 성향을 부하에 맞게 조절함으로써 처리량과 응답 시간, 두 가지 성능 척도에서 다른 정책들에 비하여 우수한 것으로 나타났다. 또한, 부하의 크기나 도착 패턴에 크게 영향을 받지 않고 주어진 상황에 부합하는 처리 성능을 나타내면서 공평하게 처리 시간도 배분하고 있는 것으로 밝혀졌다. 따라서, CAP은 부하의 크기나 도착 패턴이 불명확한 곳이나 처리 시간이 가변적인 곳에서 유용하게 사용될 수 있을 것이다.

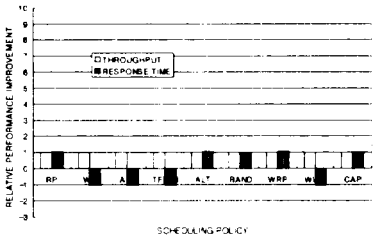


(a) 처리량

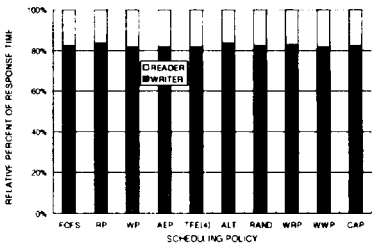


(b) 평균 응답 시간

(그림 10) 버스트를 갖는 부하의 증가에 따른 성능
(Fig. 10) Performance for the increasing bursty load
[A_T=2000, λ_T =0.11~1.21, CV=8, μ_R=1.0, μ_W=0.1]

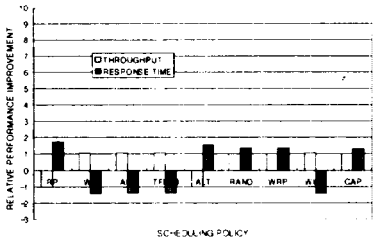


(a) 성능 향상

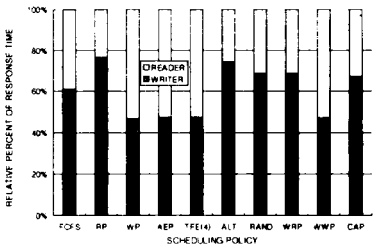


(b) 응답 시간의 구성

(그림 11) FCFS를 기준으로 한 성능 향상과 상대적 공평성
(Fig. 11) Performance Improvements and relative fairness based on FCFS)
[A_T=2000, λ_R =0.1, λ_W =0.01, μ_R=1.0, μ_W=0.1]

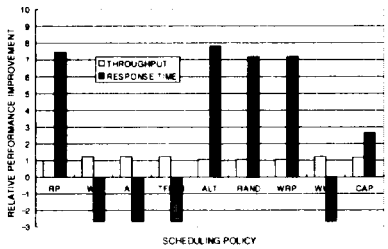


(a) 성능 향상

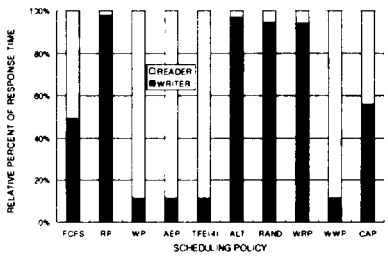


(b) 응답 시간의 구성

(그림 12) FCFS기준으로 한 성능 향상과 상대적 공평성
(Fig. 12) Performance Improvements and relative fairness based on FCFS)
[A_T=2000, λ_R =0.5, λ_W =0.05, μ_R=1.0, μ_W=0.1]



(a) 성능 향상



(b) 응답 시간의 구성

(그림 13) FCFS를 기준으로 한 성능 향상과 상대적 공평성
(Fig. 13) Performance Improvements and relative fairness based on FCFS)
[A_T=2000, λ_R =1.0, λ_W =0.1, μ_R=1.0, μ_W=0.1]

참 고 문 헌

[1] P. J. Courtois, F. Heymans, and D. L. Parnas, "Concurrent control with readers and writers," Communications of the ACM, Vol.14, No.10, pp. 667-668, 1971.

[2] L. Bic and A. C. Shaw, 'The logical design of operating systems', 2nd Ed., Prentice-Hall, 1988.

[3] J. Bacon, 'Concurrent Systems: an integrated approach to operating systems, database, and di stributed systems', Addison-Wesley, 1993.

[4] M. Singhal and N. G. Shivaratri, 'Advanced concepts in operating systems: distributed, database, and multiprocessor operating systems', McGraw -Hill, 1994.

[5] C. A. R Hoare, "Monitors: an operating system structuring concepts," Communications of ACM, Vol.21, No.8, pp.549-557, 1974.

[6] P. A. Buhr, M. Fortier, and M. H. Coffin, "Monitor classification," ACM Computing Surveys, Vol. 27, No.1, pp.64-107, March 1995.

[7] SunSoft, 'Multithreaded programming guide', Sun Microsystems Manual No.801-6659-10, 1994.

[8] B. Nicols, D. Buttler, and J. P. Farrell, 'Pthreads programming', O'Reilly & Associates, 1996.

[9] P. A. Fishwick, "Simpack: getting started with simulation programming and C and C++," Proc. of 1992 Winter Simulation Conference, 1992.

[10] E. G. Coffman Jr, H. O. Pollak, E. Gelenbe and R. C. Wood, "An analysis of parallel-read sequential-write," Performance Evaluation, Vol.1, pp.62-69, 1981.

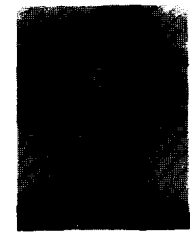
[11] T. Johnson, "Approximate analysis of reader and writer access to a shared resource," Proc. of ACM SIGMETRICS Conf. on Measuring and Modeling of Computer Systems, pp.106-114, 1990.

[12] A. Thomasian and V. F. Nicola, "Performance analysis of a threshold policy for scheduling readers and writers," IEEE Trans. on Computers, Vol.42, No.1, pp.83-98, 1993.

[13] V. G. Kulkarni and L. C. Puryear, "A reader-writer queue with reader preference," Queueing Systems, Vol.15, pp.81-97, 1994.

[14] V. G. Kulkarni and L. C. Puryear, "Stability and queueing time analysis of a reader-writer queue with alternating exhaustive priorities," Queueing Systems, Vol.19, pp.81-103, 1995.

[15] L. C. Puryear and V. G. Kulkarni, "Comparison of stability and queueing time for reader-writer queues," Performance Evaluation, Vol.30, pp.195-215, 1997.



강 성 일

sikang@casatum.kaist.ac.kr

1986년 부산대학교 계산통계학과 졸업(학사)

1988년 부산대학교 대학원 계산통계학과(이학석사)

1988년~현재 국방과학연구소 선임연구원

1996년~현재 한국과학기술원 대학원 전산학과 박사과정

관심분야 : 운영체제, 실시간 처리, 멀티미디어 통신

이 홍 규

hklee@cs.kaist.ac.kr

1978년 서울대학교 전자공학과 졸업(학사)

1981년 한국과학원 전산학과 졸업(이학석사)

1984년 한국과학기술원 전산학과 졸업(공학박사)

1985년~1986년 U. of Michigan, U.S.A(Research Scientist)

1986년~현재 한국과학기술원 전산학과 교수(부교수)

1991년~1995년공업진흥청 JTC1/SC29(전문위원)

1994년~1996년 한국통신기술협회 ITU-T SC-9 VOD 위원회(위원장)

관심분야 : 실시간 처리, 고장허용 시스템, 멀티미디어 서비스