하드웨어 지원의 재시도 없는 잠금기법

김미경* · 홍철의*

Efficient Hardware Support: The Lock Mechanism without Retry

Meekyung Kim* · Chuleui Hong*

이 논문은 2003년도 상명대학교 연구비를 지원받았음

요 약

잠금기법은 분산 병렬 시스템의 동기화에 필수적이다. 기존의 큐잉 잠금기법은 최초의 잠금 읽기와 잠금 경합 발생 시 공유 데이터에 대한 잠금이 해제되었을 때 발생하는 잠금 읽기 재시도로 두 개의 트래픽을 발생한다. 본 논문에서는 WPV(Waiting Processor Variable) 잠금기법이라 불리는 새로운 잠금기법을 제안한다. 새로이 제안하는 기법은 오직 한 개의 잠금 읽기 명령을 사용한다. WPV 기법은 파이프라인 전송방식을 사용하여 최초의 잠금 읽기 단계에서 공유 데이터가 전송될 때까지 대기한 후 잠금을 실시한다. 데이터에 대한 잠금을 수행중인 프로세서는 대기 상태의 다음 프로세서에 대한 정보를 저장하고 있으므로, 공유 데이터가 캐쉬 대 캐쉬 데이터 전송 기법에 의하여 대기중인 다음 프로세서로 바로 전송된다. 따라서 대기중인 프로세서에 대한 변수는 연결 리스트 구조를 갖는다. 제안된 기법은 캐쉬 상태의 잠금기법을 사용하여 잠금 오버헤드를 줄이고 다중 잠금 경합 발생시 FIFO를 유지하게 한다. 또한 본 논문에서는 기존의 메모리 및 캐쉬 큐잉 잠금기법에 대한 WPV 잠금기법의 해석적 모델링을 제시한다. WPV 잠금기법에 대한 시뮬레이션의 결과는 기존의 큐잉 잠금기법에 비하여 50%의 접근 시간의 감소를 보여주었다.

ABSTRACT

A lock mechanism is essential for synchronization on the multiprocessor systems. The conventional queuing lock has two bus traffics that are the initial and retry of the lock-read. This paper proposes the new locking protocol, called WPV (Waiting Processor Variable) lock mechanism, which has only one lock-read bus traffic command. The WPV mechanism accesses the shared data in the initial lock-read phase that is held in the pipelined protocol until the shared data is transferred. The WPV mechanism also uses the cache state lock mechanism to reduce the locking overhead and guarantees the FIFO lock operations in the multiple lock contentions. In this paper, we also derive the analytical model of WPV lock mechanism as well as conventional memory and cache queuing lock mechanisms. The simulation results on the WPV lock mechanism show that about 50% of access time is reduced comparing with the conventional queuing lock mechanism.

키워드

병렬 시스템 동기화, 잠금기법, 해석적 모델링, 합성부하 모델 시뮬레이션

* 상명대학교 소프트웨어학부

접수일자 : 2006. 4. 5

Ⅰ. 서 론

공유 데이터에 대한 동기화, 태스크 큐 관리 및 할당, 부하 균형 및 분산은 공유 메모리 병렬 시스템 설계 시 성능에 중요한 영향을 미친다. 이러한 기법들은 다중 프로세스들이 공유 데이터를 동시에 접근하려 하므로 효율적인 잠금기법을 요구한다[1, 2]. 기존의 잠금기법은 공유 데이터에 대하여 잠금 비트를 설정한 후 원자성을 가진 Test&Set 연산자에 의하여 접근을 통제하였다[3]. 향상된 Test&Set 연산자를 사용하는 수누핑 캐쉬 기법은 캐쉬 상태를 감시하여 잠금이 해제될 때에만 잠금 시도에 들어가게 함으로서 불필요한 쓰기를 방지하고 대기중인 프로세서가 캐쉬 상태에 머물러 있도록 한다.

한 개의 프로세서 시스템(uniprocessor system)에서는 일반적으로 버스 홀딩(bus-holding) 데이터 전송 기법이 사용된다. 그러나 다중 프로세서 시스템에서는 버스 경합이 발생하므로 공유 데이터를 얻기 위해서 많은 시간을 기다려야 한다. 반면에 파이프라인드(pipelined) 데이터 전송 기법은 하나의 데이터 전송 단계를 여러 단계로 나누어 이들을 교차시킨다. 이 기법은 대기 시간을 절약하며, 버스 사용량을 늘린다[4].

WPV(Waiting Processor Variable) 잠금기법은 파이프라인드 프로토콜의 특성을 이용한다. 데이터 대기 상태는 공유 데이터가 전송될 때까지 유지된다. 경합 상태의 공유 데이터는 다음 대기중인 프로세서로 캐쉬 대 캐쉬 데이터 전송 기법에 의하여 대기중인 프로세서의 잠금에 대한 재시도 없이 직접 전송된다. 공유 데이터를 현재 잠금중인 프로세서는 WPV 변수에 공유 데이터를 잠금 하고자하는 프로세서 번호를 저장한다. 따라서 WPV 기법은 기존의 큐잉 잠금기법에 비하여 하나의 트래픽 연산을 절약한다.

Ⅱ. WPV에 의한 잠금

메모리 잠금 방식은 잠금 실패 시에 잠금 성공 시까지 계속적으로 잠금 시도를 하게 되며, 이로 이한 버스의 사용량의 증가로 인해 버스가 포화되고 결국 시스템의 성능 저하를 가져오게 된다. 시스템이 보유하는 프로세서의 수가 많아지면 적어도 프로세서의 수에 해당하는 만큼의 잠금 버퍼를 각 메모리 모듈이 가지고 있어야만 시스템의 교착 현상을 막을 수 있다. 그러나 메모리 모듈 내에 잠금 버퍼의 수가 증가하면 매 번의 메모리 접근 때마다 잠금 버퍼의 내용을 비교해야 하기 때문에 이로 인해 메모리 접근 시간이 길어질 뿐 아니라, 메모리 보드의 값이 증가하게 된다.

캐쉬 동일성 유지 프로토콜을 이용한 잠금 방식은 잠금 시도로 인한 네트워크의 사용률을 혁신적으로 줄일 수 있는 방식일 뿐 아니라 기존의 캐쉬 데이터 일치를 위한 하드웨어를 약간의 수정만으로 이용할 수 있는 장점이 있다[5, 6, 7]. WPV 잠금기법은 수누핑 캐쉬 기법에서와 같은 하드웨어를 필요로 한다.

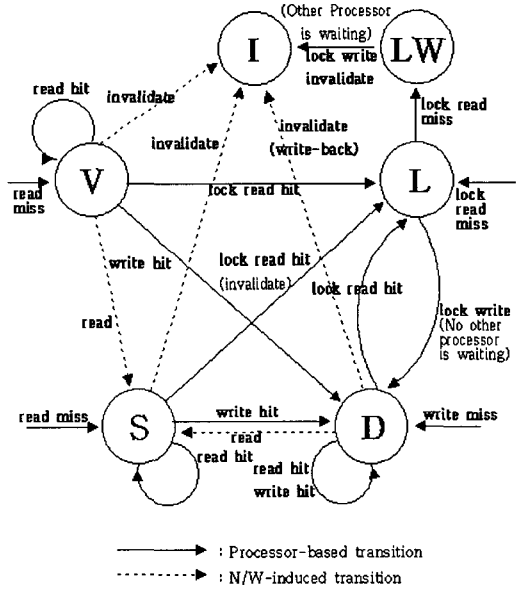


그림 1. WPV 기법의 캐쉬 상태도
Fig. 1. Cache state diagram of WPV

WPV 잠금기법은 캐쉬 일치 프로토콜을 약간만 변경하여 기존의 캐쉬 잠금기법보다 네트워크 사용량을 줄인다. 그림1은 WPV 기법에서의 캐쉬 상태도를 보여준다. WPV 기법에서는 캐쉬 블록은 LOCKED와 LOCK-WAIT 상태를 가질 수 있으므로, 별도의 잠금 비트 없이 데이터 블록은 직접 잠금 될 수 있다. 바로 이런 이유로, 대상 데이터에 대한 읽기/쓰기가 잠금/해제와 동시에 이루어지므로, 잠금/해제는 오버헤드 없이 수행된다. 또한 캐쉬 블록은 현재 잠금 상태인 공유 데이터를 기다리는 다른 프로세서가 존재한다는 것을 나타내는 LOCK-WAIT 상태

를 가질 수 있으므로 불필요한 재시도를 방지하여 네트워크의 트래픽을 줄인다.

WPV 방식에서는 각 프로세서는 WPV라 불리는 크기가 1인 버퍼를 내부에 가지고 있으며, 두 프로세서 A, B 사이의 잠금 동작이 일어나는 과정은 다음과 같다.

[단계 1] 프로세서 A가 특정 공유 데이터에 대하여 잠금을 시도하였으며 성공을 했다.
프로세서 A: 잠금 변수를 소유한다.
WPV: NULL, 캐쉬 상태: L

[단계 2] 프로세서 B가 A가 가지고 있는 공유 데이터에 대하여 잠금 동작을 시도한다.
프로세스 A: 잠금 상태를 유지한다.
WPV: NULL, 캐쉬 상태: L
프로세스 B: 공유 데이터에 대한 잠금을 시작한다.
WPV: NULL, 캐쉬 상태: Don't care

[단계 3] 수누핑 회로를 통하여 B의 잠금 시도를 안 A는 프로세서 B에게 네트워크를 통하여 잠금 실패임을 알린다. 그리고 프로세서 B의 어드레스를 자신의 내부에 소유하고 있는 WPV에 저장한다.
프로세스 A: 프로세스 B에게 잠금에 대한 응답 신호를 보낸다.
WPV: B, 캐쉬 상태: LW
프로세스 B: 잠금 읽기(Lock-Read) 시도
WPV: NULL, 캐쉬 상태: L

[단계 4] 잠금 실패 신호를 받은 프로세서 B는 버스의 사용을 중지하고 잠금 동작을 시도하던 데이터가 전송될 때까지 대기하고 있다(잠금 읽기 사이클의 연장).
프로세스 A: 잠금 상태를 유지한다.
WPV: B, 캐쉬 상태: LW
프로세스 B: 네트워크에 대한 수누핑을 수행하면서 잠금 읽기(Lock-Read) 상태를 유지한다.
WPV: NULL, 캐쉬 상태: L

[단계 5] 잠금 데이터의 사용을 마친 프로세서 A는 자신의 WPV에 저장된 어드레스인 프로세서 B에게 잠금 데이터를 직접 전달해 준다. 데이터 전달 방식은 캐쉬와 캐쉬 사이의 전송 방식과 유사한 방식이며, 프로세서 A는 쓰기 동작을 수행하고 프로세서 B는 읽기 동작을 수행함으로써 잠금 데이터의 쓰기가 수행된다.
프로세스 A: 잠금 데이터를 프로세서 B에게 전송한다.
WPV: B, 캐쉬 상태: LW
프로세스 B: 잠금 읽기 명령을 종료한다.
WPV: NULL, 캐쉬 상태: L

[단계 6] 프로세서 A는 공유 데이터에 대한 잠금을 해제하고 무효화시킨다. 프로세서 B는 프로세서 A로부터 전달된 데이터에 대한 잠금을 수행한다.
프로세스 A: 잠금 변수를 무효화한다.
WPV: NULL, 캐쉬 상태: I
프로세스 B: 잠금 연산을 수행한다.
WPV: NULL, 캐쉬 상태: L

본 잠금 방식이 캐쉬 동일성 유지 프로토콜을 이용한 방식과 다른 점은 [단계 4]와 [단계 5] 부분이다. 캐쉬의 동일성 유지 프로토콜을 이용하는 캐쉬 잠금 방식에서는 프로세서 A가 공유 데이터에 대하여 잠금을 해제할 때 즉, 잠금 쓰기(LOCK-WRITE) 시에 버스상의 다른 프로세서에게 Invalidate 신호를 보내게 된다. 이 신호를 받은 프로세서 B는 다시 버스를 통하여 잠금 읽기 시도를 하고 프로세서 A는 캐쉬간의 데이터 전송 기능을 이용하여 데이터를 전송하게 된다. 따라서 WPV 방식에 비하여 한 번의 버스 사용이 많게 된다.

다음은 WPV 잠금 방식에 대한 의사 코드를 보여준다.

```
procedure acquire_lock(Lock Variable: lock)
    if (lock is in cache)  //cache hit
        if (cache_state(lock) is S)
            invalidate(lock) to network;
            // if cache state is V or D,
            // we don't need invalidation
        set cache_state(lock) to L;
    else  //cache miss
        Lock-Read(lock) to network
        if (acknowledge is received)
            set cache_state(lock) to L;
            latch data network until data
                is transferred;
```

```
procedure release_lock(Lock Variable: lock)
        if (cache_state(lock) is L)
          set cache_state(lock) to D;
        else //cache state is Lock Wait
          transfer data to WPV processor;
          set cache_state(lock) to I;

procedure snoop(Lock Variable: lock)
        while(TRUE)
          network snooping LCR(lock)
          //LCR is Lock-Read network command
          if (requester of LCR is not my processor
                  and cache_state(lock) is L)
                  send acknowledge to requester;
                  store requester of LCR to WPV;
                  set cache_state(lock) to LW;
```

WPV 잠금 방식은 잠금 재시도를 수행하지 않으므로 기존의 캐쉬 잠금기법에 비하여 하나의 버스 사용을 줄인다. WPV는 일종의 linked list로서 FIFO로 동작한다. WPV에 의해 구성된 linked list에 따라 잠금에 대한 소유권이 옮겨가게 되고, 이러한 방식은 각 프로세서 보드에 대하여 잠금 동작을 수행할 때까지의 대기 시간에 대한 최대 한계(Upper bound)를 제공함으로써 잠금 동작에 확실성을 보장하고, 순차적인 잠금 동작을 보장함으로써 교착상태가 없는(deadlock free) 잠금 동작을 제공한다. 또한 메모리 잠금 방식과 캐쉬 동일성 유지 프로토콜을 위한 방식에서는 버스 프로토콜 및 각 방식의 구현 방식에 따라 발생할 수 있는 기아현상(starvation)을 WPV를 이용한 방식에서는 제거할 수 있다.

Ⅲ. 해석적 모델링

본 절에서는 메모리 잠금장치, 캐쉬 동일성 유지 프로토콜을 이용한 잠금장치 및 WPV 잠금 장치의 해석적 모델링을 구하고 각각의 성능을 비교한다. p를 잠금(lock) 참조율을 나타내는 확률이라 가정할 때, 잠금 변수(lock variable) 참조가 독립적이고 동일하게 분포되어 있다면, 특정 프로세서가 잠금을 거는데 걸리는 시간은 기하학분포를 이룬다. 그러므로 평균 수행 간격(execution interval), $x = (1-p)/p$ 이다. 그러나 실제 상황에서의 프로세서당 잠금 참조율, r은 초기 값의 잠금 참조율인 p 보다 크다. 그 이유는 실제 잠금 참조율 r은 처음 시도의 잠금 참조와 재시도의 잠금 참조를 더한 값이기 때문이다. 잠금 참조율, r을 직접 구하기는 어려우나, simple fixed-point iterative algorithm으로 근삿값을 구할 수 있다.

우선 잠금 참조율 r을 갖는 시스템 대역폭, BW을 구해보면 다음과 같다. 총 프로세서 개수를 N, 총 lock variable의 개수를 L, 그리고 라우팅 수를 B개라고 가정하면, 임의의 프로세서 I가 임의의 잠금 변수를 참조할 확률은 r/L이다. 그러므로 임의의 잠금 변수 j를 참조하지 않을 확률은 $1-(r/L)$ 이고, N개중의 어느 프로세서도 임의의 잠금 변수 j를 참조하지 않은 확률은 $(1-(r/L))^N$ 이다. 그러므로 적어도 한 개 이상의 프로세서가 임의의 잠금 변수 j를 참조할 확률은 다음과 같다.

$$q = 1 - \left(1 - \frac{r}{L}\right)^N \tag{1}$$

또한, 잠금 변수 참조가 독립적이고, 동일하게 분포되어 있다면, L개의 잠금 변수 중 i개의 잠금 변수가 참조될 확률 f는 binomial분포를 이룬다.

$$f_i = \binom{L}{i} q^i (1-q)^{L-i} \tag{2}$$

그러므로 1 cycle 동안 참조될 잠금 변수의 평균 개수는 qL이며 참조될 잠금 변수 중 B개만이 성공하여 잠금 변수를 참조하므로, 시스템 Bandwidth는 다음과 같이 된다.

$$BW = B\sum_{i=B}^{L} f_i + \sum_{i=1}^{B-1} (i \cdot f_i) \tag{3}$$

메모리 잠금 장치에서 실제 잠금 변수 참조율, r을 구하는 방법은 다음과 같다. T를 평균 lock interval time이라고 하면, 평균 execution interval time, $x = (1-p)/p$, 잠금을 하기 위해 잠금 변수 및 N/W contention에 의해 지연되는 평균 시간을 d_l, 평균 locking operation 시간을 l_{op}로 정한다. 또한, N/W contention에 의해서만 지연되는 평균 시간을 d_n, lock 및 unlock은 1 cycle에 완료되는 것으로 정한다. 그러므로 평균 lock contention time은 다음 식으로 표현된다.

$$T = x + l_{op} + d_l + d_n + 2 \tag{4}$$

또한, T 시간 동안 네트워크를 통한 참조 개수는 다음과 같다.

$$N_{req} = d_l + d_n + 2 \tag{5}$$

그러므로 한 개 프로세서당 잠금 변수 참조율, r은 다음과 같이 나타낼 수 있다.

$$r=\frac{N_{req}}{T}=\frac{d_l+d_n+2}{x+l_{op}+d_l+d_n+2} \quad (6)$$

또한 프로세서당 잠금 연산(lock operation)의 종료율은 $BW/2N$이다. 평균 lock interval time $T=2N/BW$이다. 그러므로

$$N_{req}=d_l+d_n+2=r\times\frac{2N}{BW} \quad (7)$$

따라서 식 (6)을 식 (7)로 부터 다음 식이 얻어진다.

$$r=\frac{\frac{2rN}{BW}}{x+l_{op}+\frac{2rN}{BW}}=\left(1+\frac{BW\cdot(x+l_{op})}{2rN}\right)^{-1} \quad (8)$$

여기서 r은 r의 식으로 나타내어지므로 시스템 대역폭 BW을 계산하기 위하여 fixed-point iterative algorithm을 사용하면 다음과 같다.

단계 1: 초기의 시스템 대역폭, BWO을 계산한 후 초기 참조율 r0 = p로 놓는다.

단계 2: 식 (8)을 사용하여 근삿값 r을 구한다.

$$r_i=\left(1+\frac{BW_{i-1}\cdot(x+l_{op})}{2\cdot r_{i-1}\cdot N}\right)^{-1} \quad (9)$$

단계 3: 적어도 한 개 이상의 프로세서가 임의의 잠금 변수 j를 참조할 확률 $q=1-(1-r_i/L)^N$을 계산하여 새로운 시스템 대역폭 BWi를 계산한다.

단계 4: $|BW_i-BW_{i-1}|<\epsilon$ 이면 수행을 종료하며, 원하는 범위, ε안으로 접근하지 못했을 경우, 단계 2로 되돌아가 수행을 계속한다.

메모리 잠금 장치의 해석적 모델을 기본으로 캐쉬 큐잉 잠금 및 WPV 잠금기법의 해석적 모델을 나타내면 다음과 같다.

캐쉬 큐잉 기법 :

$$r=\left(1+\frac{BW\cdot(x+l_{op}+w)}{r\cdot(\alpha+2)\cdot N}\right)^{-1} \quad (10)$$

WPV 기법 :

$$r=\left(1+\frac{BW\cdot(x+l_{op}+w)}{r\cdot 2\cdot N}\right)^{-1} \quad (11)$$

여기서 α 는 잠금 변수 경합 확률 (lock variable contention probability)을 나타낸다.

$$\alpha=\frac{r}{N}\left(1-\left(\frac{1-r}{L}\right)^{N-1}\right)$$

그러므로 $w=\alpha\cdot d_l$ 로 나타내어질 수 있다.

마지막으로 그림 2는 메모리 잠금기법, 캐쉬 동일성 유지 프로토콜을 이용한 잠금기법 및 WPV 잠금기법의 잠금 수행에 대한 시간 경과를 보여준다.

<메모리 잠금기법>

execution interval	blocked req. for lock	lock	locking operation	blocked req for N/W	unlock
x	d_l	l	l_{op}	d_n	l

<캐쉬 잠금기법>

execution interval	blocked for N/W	lock	wait	additional req.	lock	locking operation	blocked req. for N/W	unlock
x	d_n	l	w	a	l	l_{op}	d_n	l

<WPV 잠금기법>

execution interval	blocked for N/W	lock	wait	locking operation	blocked req for N/W	unlock
x	d_n	l	w	l_{op}	d_n	l

그림 2. 여러 가지 잠금기법의 시간 경과 비교
Fig. 2 Time comparison of lock mechanisms

Ⅳ. 시뮬레이션 및 결과 분석

시스템 시뮬레이션은 시스템 설계 단계에서 병목현상 제거 및 용량 설계 (capacity planning) 등에 유용하게 사용되며, 시스템이 완성된 후에도 시스템을 실제로 작동하지 않고 저렴하고 신속하게 성능 측정을 가능하게 한다. 시스템 시뮬레이션 시 시스템 모델 및 부하 모델을 실제 환경에 맞게 설계하는 것은 정확한 성능 측정을 가능하게 한다. 부하모델은 실제 트레이스를 이용한 부하 모델 (trace-driven workload model)과 합성 부하 모델 (synthetic workload model)로 구분할 수 있다.

트레이스 부하 모델을 이용한 시뮬레이션은 실제적이지만 그 데이터양이 방대하여 전체 시스템을 시뮬레이션하기에는 적합하지 못하다. 반면에, 합성 부하 모델은 사실성은 부족하나 유연성을 제공하여, 서로 다른 부하 변수를 조정함으로서 그 변수가 미치는 영향을 명확하게 알 수 있다. 합성 부하모델이 실제적이기 위해서는 시스템의 세세한 부분 및 사용되는 입력 변수를 정확하게 설정하여야 한다. 입력 부하 모델로서 다음의 식이 사용되었다[8].

$$M(r)=A\times r^{\frac{1}{\theta}}$$

여기서 M(r)은 r번째 접근 시까지의 참조 실패의 수를 나타내며, Θ는 시간 지역성을 나타내는 변수로, Θ가 1에

접근하면 대부분의 참조가 실패로 나타나게 되고, θ가 커질수록 참조 성공 확률이 올라간다. A는 공간 지역성을 나타내는 변수이다.

WPV를 이용한 잠금 방식의 성능을 예측하기 위하여, 메모리 잠금 방식과 현재 많이 사용되고 있는 큐(Queue)를 이용한 잠금 방식, 그리고 WPV를 이용한 잠금 방식에 대한 시뮬레이터를 각각 구성하였다. 캐쉬의 실패율(miss rate)은 3%로 가정하였다[9]. 각 프로세서가 수행하는 프로그램 모델은 아래와 같다.

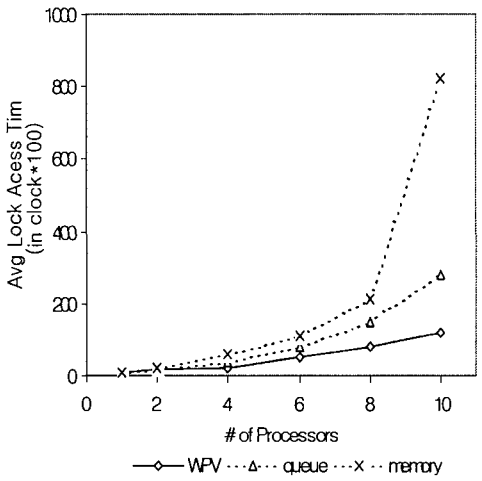
1. 비임계 구역 (Non-Critical section);
   각 프로세서는 비임계 구역을 수행할 때, 캐쉬의 참조 성공/실패 여부에 따라 네트워크를 통하거나, 캐쉬로부터 데이터를 참조하게 된다.
2. while(NOT TAS(&lock));
   본 프로그램 모델에서 TAS(test-and-set)명령어는 공유변수 lock에 대하여 변수를 읽고, 최상위 비트를 1로 쓰는 명령어로써, 읽기와 쓰기 동작 시에 다른 프로세서로부터 방해를 받지 않도록 구현되어야 한다. TAS 명령어를 처리하는 방식의 차이에 따라, 메모리 잠금, 큐를 이용한 잠금, WPV를 이용한 잠금 방식으로 구분된다.
3. 임계 구역 (Critical Section);
   잠금 동작을 통하여, 수행 권을 얻은 프로세서는 임계 구역을 수행하게 되는데, 이 부분에 해당하는 동작도 캐쉬의 참조 성공/실패에 따라 네트워크의 사용 여부가 결정된다. 한편 임계 크기(수행되는 명령어 수)에 따라, 잠금 방식의 성능의 차이가 있으며, 본 시뮬레이션에서는, 각각 10개와 5개의 명령어 수행에 대하여 실시한다.
4. lock = FALSE;
   임계 구역의 수행을 마친 프로세서는 잠금 변수에 대하여 대기하고 있는 프로세서가 임계 구역을 수행할 수 있도록 잠금을 풀어준다. 이 동작도 세 가지 형태의 잠금 방식에 따라 방식이 달라진다.
5. 비임계 구역 (Non-Critical section);
   비임계 구역의 수행 시에는 다시 캐쉬의 데이터 참조 성공/실패에 따라 네트워크를 사용한다.

본 시뮬레이션을 통하여 각 잠금 방식에 대한 평균 잠금 수행 시간(Average Lock Access Time), 네트워크 사용률(Network Utilization), 프로세서 사용률(Processor utilization)을 알 수 있다.

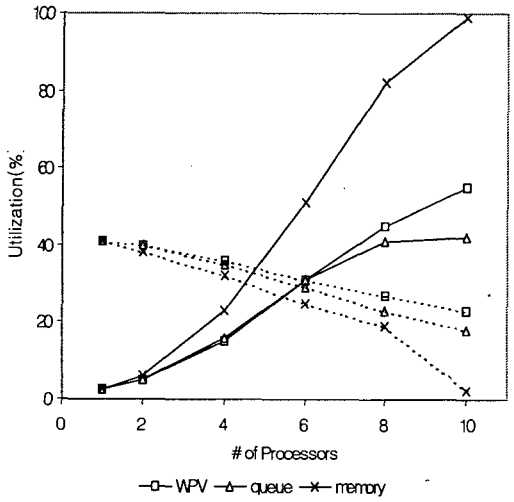
[단계 1] 평균 잠금 빈도를 1% (평균적으로 100개의 명령어 중 1개의 잠금 (TAS 명령어)이 있는 입력 부하)과 임계 영역의 명령어 수를 10개로 고정하고, 프로세서의 수를 1개에서 10개까지 변화시키면서, 평균 잠금 수행 시간(그림 3), 버스 사용률, 프로세서 사용률(그림 4)을 측정한다.

[단계 2] 프로세서의 수를 8개, 임계영역의 수를 10개로 고정하고, 잠금 빈도를 1%에서 10%로 변화시키면서 세 가지 잠금 방식에 대한 평균 잠금 수행 시간(그림 5), 버스 사용률(그림 6)을 측정한다.

[단계 1]의 결과인 그림 3을 보면 프로세서의 수가 8개를 넘어갈 때, 메모리 잠금 방식의 경우, 평균 잠금 수행시간이 급격히 증가함을 알 수 있다. 이러한 이유는 그림 4에서 보는 바와 같이 네트워크의 사용률이 급속히 증가하는 현상으로 설명될 수 있는데, 잠금 변수에 대한 공유 율이 증가함에 따라 버스의 사용률이 많아짐으로 인해, 임계영역의 수행 시와 잠금 해제를 위한 쓰기 동작 시에, 네트워크에서 충돌로 인한 시간지연으로 인해 평균수행시간이 증가하기 때문이다. 또한 그림 4에서 보는 바와 같이 프로세서의 사용률이 급격히 저하되어 자원 사용의 효율을 떨어뜨리는 요인이 된다.



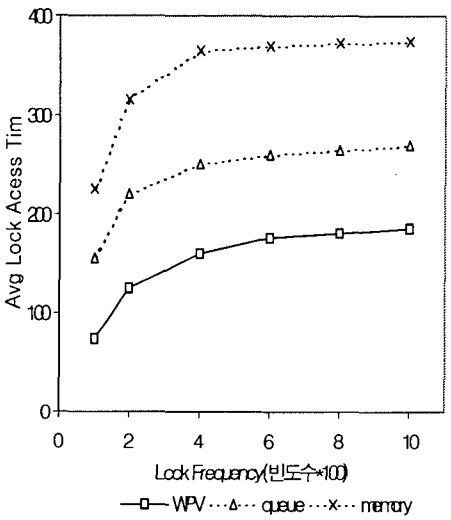
Critical section : 10 instructions
Cache miss ratio : 3%, Lock frequency : 0.01
그림 3. 프로세서 수에 대한 WPV, 큐, 메모리 기법의 평균 잠금 수행 시간
Fig. 3 Average lock access times on number of processors



—— : N/W , ····· : Processor
Critical section : 10 instructions, Cache miss ratio : 3%
Lock frequency : 0.01

그림 4. 프로세서 수에 대한 WPV, 큐, 메모리기법의 네트워크 사용율과 프로세서 사용률

Fig. 4 Utilizations of network and processor on number of processors

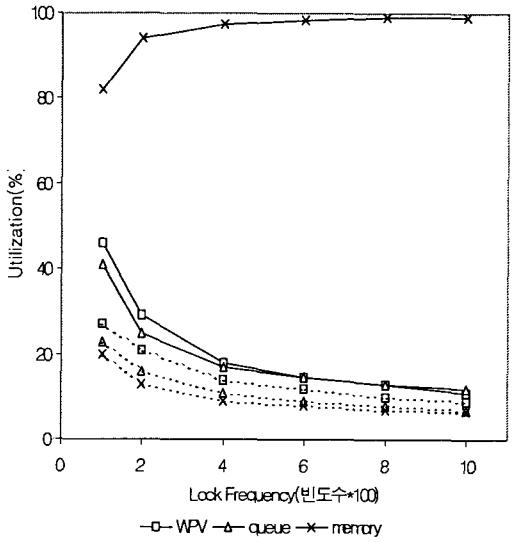


Critical section : 10 instructions, Cache miss ratio : 3%
# of processors: 8

그림 5. 잠금 빈도수에 대한 WPV, 큐, 메모리 기법의 평균 잠금 수행 시간

Fig. 5 Average lock access times on lock frequency

이러한 잠금 동작으로 인한 네트워크의 트래픽(traffic)증가는 잠금 동작에 수행시간뿐 아니라 다른 버스 동작의 수행시간도 지연시켜, 시스템의 성능을 저하시키는 요소가 된다. 큐를 이용한 방식과 WPV를 이용한 방식은 프로세서 6개 미만에서는 성능상의 큰 차이를 보이지 않으나, 8개를 넘어가면서 차이를 보이기 시작한다.



—— : N/W , ····· : Processor
Critical section : 10 instructions, Cache miss ratio : 3%
# of processors: 8

그림 6. 잠금 빈도수에 대한 WPV, 큐, 메모리 기법의 네트워크 사용율과 프로세서 사용률

Fig. 6 Utilizations of N/w and processor on lock frequency

[단계 2]에서는 프로세서가 8일 때 잠금 빈도를 변화시키며 세 가지 방식의 성능차이를 측정한다. 그림 5에서 보는 바와 같이 잠금 빈도에 관계없이 항상 일정한 성능의 차이를 보여 주고 있다. 이는 그림 6에서 보는 바와 같이 네트워크 사용율과 프로세서 사용률이 잠금 빈도에 영향없이 일정한 차이를 유지하고 있는 것으로도 일치한다. 이로써, 잠금 방식에 따른 성능의 차이는 잠금 빈도에는 영향을 받지 않고, 잠금의 공유 율에 크게 영향을 받는다는 결론에 도달한다.

성능 측정의 결과로 WPV 및 큐잉 잠금 방식은 메모리를 이용한 잠금 방식에 비하여 월등한 성능향상을 보인다. 또한 WPV 잠금 방식은 큐를 이용한 잠금 방식에 비하

여 잠금 경합 발생 시 재시도의 잠금 읽기를 하지 않으므로 2배 정도의 성능향상을 보인다. 잠금 경합이 없을 시는 큐를 이용한 잠금 방법과 똑같은 수의 트래픽을 발생하므로 큐를 이용한 방법과 같은 성능을 보인다. 이상과 같이 WPV 잠금 방식은 잠금 경합의 유무에 상관없이 항상 기존의 잠금 방식에 비하여 좋은 성능을 나타낸다.

V. 결 론

WPV 잠금기법은 파이프라인드 데이터 전송 프로토콜을 이용하였으며, 작은 크기(fine grain granularity)의 잠금 연산에 있어서 효율적인 성능을 보였다. 본 기법은 경합 공유 데이터에 대하여 재시도를 행하지 않으므로, 기존의 캐쉬 큐잉 잠금기법의 두 번의 트래픽 명령에 비하여 한 번의 트래픽 명령만을 수행한다. 분산 병렬 시스템에서 일반적으로 네트워크가 병목현상으로 작용하므로, WPV 기법은 전체 시스템 성능을 크게 높인다. 시뮬레이션 결과 기존의 기법에 비하여 2배의 성능 향상이 있음을 보여주었다. 더구나, 본 기법은 캐쉬 상태의 잠금기법을 사용함으로서 잠금 연산에 대한 오버헤드를 낮추었다. 또한 잠금 대기 시간에 대한 한계치를 예상할 수 있으며 교착상태 없는 연산을 보장한다.

참고문헌

[1] Moon, E., Thang, S., Jhon, C., "Adjacency preferred hardware synchronization method for CC-NUMA systems", Proceedings of International Conference on Electronics, Informations and Communications. 1998

[2] Moon, E., Thang, S., Jhon, C., "Analysis of the Relation of Synchronization Algorithm and Parallel Programs in Shared-Memory Multiprocessor Systems", Proceedings of High Performance Computing Symposium, 2000

[3] Tauri, T., Nakagawa, T., Ido, N., "Evaluation of the Lock Mechanism in a Snooping Cache", Proceedings of International Conference on Supercomputing, 1992

[4] An Do Ki, Byung Kwan Park, "Highly Pipelined Bus : HiPi-Bus," Proceedings JTC-CSCC '91, 1991

[5] Cho, H., "Cache coherency and concurrency control in a multisystem data sharing", IEICE Transactions on Information & Systems, Vol. E82-D. No. 6, 1999 pp. 1042-1050

[6] Nikolopoulos, D., Papatheodorou, S., "Fast Synchronization on Scalable Cache Coherent Multiprocessors using Hybrid", Proceedings of the 14th International Parallel & Distributed Processing, 2000 pp. 711-719

[7] Stenstrom, P., "A Survey of Cache Coherence Schemes for Multiprocessors", IEEE Computer, June 1990 pp. 12-24

[8] Thiebaut, D., "On fractal dimension of computer programs and its application to the prediction of cache miss ratio", IEEE Trans. on Computers, Vol. 38. No. 7, 1989, pp. 1012-1026

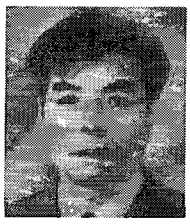
[9] Yamamura, S., Hirai, A., Yamamoto, M, "Speeding Up Kernel Scheduler by Reducing Cache Misses", Proceedings of the Freenix Track 2002 USENIX Annual Technical Conference, 2002, pp. 275-285

저자소개

김 미 경(Meekyung Kim)

1998년 상명대학교 정보과학과 (학사)
2000년 상명대학교 컴퓨터과학과(석사)
1999년 ~ 현재 상명대학교 컴퓨터과학과(박사)

※관심분야 : 분산시스템 및 알고리즘, 원격 교육, 멀티미디어 응용



홍 철 의(Chuleui Hong)

1989년 미국 New Jersey Institute of Technology 전산학(석사)
1992년 미국 Univ. of Missouri - Rolla 전산학(박사)
1992년 ~ 1997년 한국전자통신연구원 선임연구원
1997년 ~ 현재 상명대학교 소프트웨어학부 부교수

※관심분야 : 분산시스템 및 알고리즘, 최적화 기법, 멀티미디어 응용