실시간 CORBA 시스템에서 새로운 실시간 스케쥴링 기법

(A New Real-Time Scheduling Scheme on Real-Time CORBA Systems)

백 승 민 † 김 성 천 ††
(Seungmin Baek) (Sungchun Kim)

요 약 오늘날, 이질적인 분산 컴퓨팅 환경을 통합하고 분산 공유 자원을 효율적으로 사용할 수 있는 공통 작업 환경에 대한 요구가 날로 높아지고 있다. 이러한 연구 중에서 주목받는 것 중 하나가 OMG의 CORBA이다. CORBA 시스템에서 실시간 멀티미디어 데이타들을 처리하기 위해 확장된 CORBA 표준을 RT(Real-Time) CORBA라고 한다. RT CORBA의 핵심은 수많은 실시간 태스크들의 블록을 최대한 방지하여 실시간 QoS를 보장해 줄 수 있는 실시간 스케쥴링 기법에 있다고 할 수 있다. 현재 RT CORBA를 위해 제안되는 많은 스케쥴링 기법들은 대개 이전의 단일 시스템에 기반한 전역 우선순위 기반 스케쥴링이 대부분이다. 하지만 이러한 Soft Real-Time 스케쥴링은 다양한 성능 요소가 산재하는 분산 시스템에서는 그 성능을 보장할 수 없다. 본 논문에서는 CORBA와 같은 분산시스템의 보편적 특성을 고려한 Hard Real-Time 스케쥴링 기법인 CII(Cut In Interval) 기법을 제안하였다. 기존의 전역 우선순위 바탕의 스케쥴링이 비효율적인 태스크 할당 및 지역 스케쥴링을 제공하지만, CII 기법의 스케쥴링은 간단하면서도 보다 효과적인 전역 스케쥴링을 제공할 수 있다. CII 기법의 핵심은 이미 스케쥴링된 태스크를 가능한 제한시간까지 연기하여 얻어진 여유시간을 다른 실시간 태스크들의 처리에 활용하자는 것이다. 이러한 여유시간의 활용은 제안한 기법이 기존의 기법과 달리 보다 능동적이고 효율적인 스케쥴링 기법임을 증명한다.

Abstract RT SIG(Real-Time Special Interest Group) of OMG has led many researches for the real-time multimedia data processing on the CORBA system. The extended standard of CORBA for real-time processing is called RT(Real-Time) CORBA. The core of RT CORBA is the real-time scheduling scheme that can guarantee the real-time QoS by minimizing the blocked ratio of real-time tasks. The most of nowadays proposed scheduling schemes are the kinds of previous priority-based scheduling schemes for uni-processor systems. But these soft real-time scheduling schemes cannot guarantee good performance on the distributed computer systems of various performance parameters. This paper proposes a hard real-time scheduling scheme, CII(Cut In Interval). The previous global priority-based scheduling schemes only offer the inefficient task allocation and local scheduling. But CII-based scheduling scheme offers simpler and more efficient global scheduling. The key point of CII is the following. The remained time, resulting from putting off the already scheduled task possibly till the deadline, can be spared for the other tasks. The use of remained-time proves that CII-based scheduling scheme is more active and efficient scheduling scheme than previous priority-based schemes.

1. 서 론

인터넷 사용의 보편화는 단순히 데이타 전송을 위한 네트워크가 아닌, 산재된 멀티미디어 데이타를 분산 처리할 수 있는 시스템 네트워크(SAN; System Area Network)에 대한 요구를 증대시키고 있다. 이러한 요구에 부응하기 위하여 인터넷상에서 이질적인 분산 컴퓨팅 환경을 통합하고 산재한 공유 자원을 효율적으로 활용하여 실시간 서비스를 제공할 수 있는 공통 환경을 제공하

· 이 연구는 2001년도 서강대학교 교내 연구비 지원에 의하여 이루어졌음.
† 비 회 원 : 서강대학교 컴퓨터학과
100seungmin@hanmail.net
†† 종신회원 : 서강대학교 컴퓨터학과 교수
ksc@arqlab1.sogang.ac.kr
논문접수 : 2000년 4월 11일
심사완료 : 2001년 7월 19일

기 위한 많은 연구가 진행되고 있다. 이러한 연구 중에 최근 주목을 받고있는 것이 CORBA(Common Object Request Broker Architecture)이다.[1][2] CORBA는 분산 객체 컴퓨팅 환경의 표준이며, 이러한 CORBA 환경에서 실시간 처리를 위해 확장된 표준을 RT(Real-Time) CORBA라고 한다.

RT CORBA에서 채택하고 있는 기존 실시간 스케쥴링 기법의 상당 부분들이 이전의 단일 시스템 환경의 우선순위(priority) 할당에 기초한 실시간 스케쥴링 기법에 바탕을 두고 있다.[3-6] 이것은 모든 시스템 내의 태스크(task)들이 하나의 우선순위를 가지고 마치 한 지역 시스템 상에서처럼 처리될 수 있어야 한다는 요구 때문이다. 본 논문의 목적은 현재 RT CORBA를 위해 제안되고 있는 대부분의 단일 시스템 스케쥴링 기법(RM:Rate Monotonic, EDF:Earliest Deadline First, MRF:Minimum Remained -time First 등등)에 기인한 비효율적인 전역적 우선순위 기반 스케쥴링 기법의 문제점을 지적하고, CORBA가 제공하는 분산 객체 컴퓨팅 환경에 보다 적응된 새로운 전역 스케쥴링 기법을 제안하는데 있다.

본 논문에서 이러한 개선된 실시간 전역 스케쥴링 기법을 위하여 CII(Cut In Interval)기법을 제안한다. 그리고 기존의 가장 안정된 성능이 증명되온 EDF기법에 바탕을 둔 전역적 우선순위 스케쥴링 기법[3][7-9]과 제안한 CII기법에 바탕을 둔 개선된 전역 스케쥴링 기법을 비교하여 제안한 기법의 알고리즘의 효율성을 증명하고 시뮬레이션을 통해 이를 검증한다.

2. RT CORBA의 기존 실시간 스케쥴링 기법과 문제점

CORBA란 이질적 분산 환경의 각각의 시스템 컴포넌트들을 개체화하고 이를 위한 공통된 인터페이스를 제공함으로써 분산 투명성을 실현하여 가상의 단일 시스템 작업 환경을 제공할 수 있는 미들웨어(middleware)를 말한다.[1][2]

OMG는 이러한 CORBA 환경에서 실시간 어플리케이션을 서비스하기 위하여 "CORBA 시스템 내에서 단과 단(end to end) 사이의 실시간 제약에 대하여 표현하고 수행할 수 있어야 한다."와 같은 요구를 수용할 수 있는 RT CORBA를 정의하였다.[10] RT CORBA에는 실시간 태스크들이 ORB에 수행 요구를 전달할 수 있는 TDMI(Timed Distributed Method Invocation)라는 특수화된 인터페이스가 정의된다. 이 TDMI를 통해 ORB에 전달된 태스크들은 실시간성을 표현하기 위해 RT-Environment라는 데이타 구조를 사용한다. RT-Environment에는 각 실시간 태스크의 수행시간 및 수행 제한시간(deadline), 태스크의 QoS에 따른 가중치 등 태스크 수행에 필요한 일련의 정보들이 저장되게 된다. 또한 RT CORBA는 실시간 태스크들을 처리하기 위해 Global Time Service, Real-Time Event Service, Dynamic Scheduling Service, Real-Time Concurrency Control Service와 같은 다양한 객체 서비스를 필요로 한다.[4][5][10]

RT CORBA의 요구사항들을 정리하면 모든 태스크가 하나의 시스템에서와 같이 처리될 수 있어야 한다는 것이다. 그것은 각각의 태스크가 RT CORBA 시스템 내에서 동일한 우선순위를 가지고 처리되어질 수 있어야 한다는 것을 의미한다. RT CORBA에서 새로운 실시간 태스크가 생성되면 RM, EDF, MRF 등 기존의 실시간 스케쥴링 기법에 따라 전역적 우선순위를 할당하고 임의의 부하균등화 원칙에 따라 특정 노드에 태스크를 할당하게 된다. 이 때, 각 프로세싱 노드는 독자적 OS하에 우선순위 할당 규칙이 적용되므로 전역적 우선순위를 각 OS가 제공하는 지역적 우선순위로 매핑(mapping)시킬 필요가 있다. 또한 이미 해당 노드에 할당된 태스크들에 대해서도 우선순위 조정의 과정이 필요할 수도 있다. 이런 경우 다음과 같은 우선순위 매핑 규칙은 매우 중요하다.

"If Global-Priority(task a) > Global-Priority(task b), then Local-Priority(task a) >= Local-Priority(task b)"

현재의 RT CORBA 시스템에는 이러한 우선순위 할당 방법으로 기존의 단일 프로세서 시스템을 위한 기법들이 그대로 적용됨으로써 자원의 비효율적 낭비를 초래한다. 분산시스템에서는 여러 가지 성능 변수가 존재하며 이는 전체적 성능에 큰 영향을 미친다. 기존의 단일 프로세서 시스템을 위한 스케쥴링 기법들은 이러한 요소에 대한 고려가 전무함으로 그 비효율성은 명백하다고 할 수 있다.

또한 전역적 우선순위를 지역적 우선순위로 매핑하는 과정이 현재의 RT CORBA 실시간 스케쥴링 기법에서는 대개 시스템의 상태에 무관한 단순 매핑일 뿐 아니라, 초기 태스크 할당도 단순한 부하균등화 원칙에 따라 수행한다. 시스템 상태를 등한시하는 이런 스케쥴링 기법은 자원 낭비와 시스템 성능 감소를 초래하게 된다. 만약 보다 효율적인 부하균등화 기법을 도입하고자 한다면 시스템의 정보 수집 및 스케쥴링 기법의 개선 등에 많은 오버헤드를 감수해야 한다.

3. RT CORBA 시스템에서 개선된 실시간 스케쥴링 기법

3.1 CII(Cut In Interval) 기법

간단히 CII 기법이란 이미 스케쥴링된 태스크들 사이에 새로운 태스크를 끼어 넣을 수 있는 틈(interval)들을 조사하고, 그 중 최대 또는 최소의 공간에 새로운 태스크를 삽입하는 스케쥴링 기법이라 정의할 수 있다. 이때 각각의 태스크들은 자신의 제한시간과 바로 뒤의 태스크가 더 이상 지연되지 못하고 시작되어야 하는 시간 중에 더 작은 시간에서 태스크의 작업시간을 뺀 시점까지 연기될 수 있다. 이를 식으로 정리하면 다음과 같다.

【가정】 task(n)은 작업 큐에서 n번째 스케쥴링된 태스크이다.(n은 자연수)

f(n) = task(n+1)이 최대 지연시 시작되어야 하는 시간

f(n+1) = task(n+2)이 최대 지연시 시작되어야 하는 시간

s(n+1) = task(n+1)의 작업시간

d(n) = task(n)의 제한시간

f(n) = minimum[f(n+1)-s(n+1), d(n)] or d(n) (if task(n) is the last) 식 (1)

위의 식과 같이 f(n)(='fn')이 계산되면 task(n)과 task(n-1) 사이에 새로운 태스크가 할당될 수 있는지 없는지를 판단할 수 있다. 그림 1은 하나의 작업 큐에서

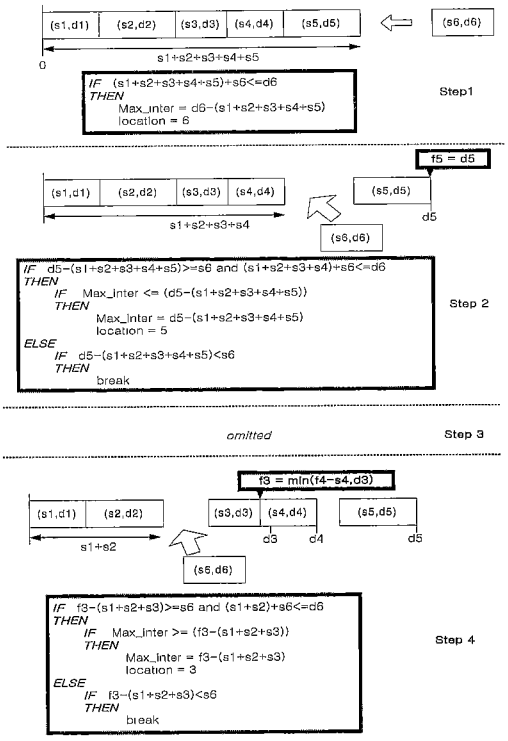


그림 1 CII 기법의 수행 과정

```
Cut_In_Interval( ) {
  now = localqueue.first;
  for ( i = 1  i < localqueue.qlength  i++ )     /* qlength-1 steps */
  { total_jobtime += now->jobtime;                /* the sum of job times */
      now = now->next;}                  /* let now point the rear of the queue */
  location=-1 /* initialize the value of location */
  total_jobtime += now->jobtime;
  f = newdeadline;                    /* f is the deadline of requested task */

  for  ( i = localqueue.qlength+1; i > 0  i-- ) {/* CII steps */
/*check whether the request task can be scheduled in the interval and processed in time*/
            if ( f-total_jobtime >= newjobtime &&
                                  total_jobtime-now->jobtime + newjobtime <= newdeadline )
                { if ( spacesize <= f-total_jobtime )  /*compare with previous location */
                      { spacesize = f-total_jobtime;       location = i; }}
             else if ( newjobtime > f-total_jobtime )    /* next steps are useless */
               break;

            /* update f for the next */
             if ( i == localqueue.qlength + 1 )      /* if the front of the queue */
                    f = now->deadline;
             else {
                    total_jobtime -= now->jobtime;
                    if ( now->prev!=NULL ) {
                            f = min(f-now->jobtime, (now->prev)->deadline);
                            now = now->prev; }
                  }/* else */
               }/ * for */
           }/* end of Cut_In_Interval() */
```

그림 2 CII 알고리즘

최대 간격을 찾는 CII 기법의 알고리즘을 단계적으로 묘사한 것이다. 그림에서 CII 기법의 시간복잡도가 O(q)(q;Avg. of queue length)임을 알 수 있다. 그림 1은 1단계, 2단계, 그리고 4단계의 계산 과정을 대표적으로 보여주고 있다.

CII 알고리즘에서 두 가지 주목할 사항이 존재한다. 첫째는 그림 1의 4단계와 같이 태스크를 자신의 제한시간까지 연기할 수 없는 경우이다. 그림에서 3번째 태스크는 4번째 태스크로 인해 d3까지가 아니라 f3까지 연기될 수 있다. 하지만 이러한 f(n) 값은 이미 위에서 언급한 식(1)과 같이 간단히 구할 수 있다. 둘째는 두 태스크 사이의 간격이 새로운 태스크의 크기보다 작아지는 단계이다. 이런 경우 더 이상의 단계는 불필요하게 된다. 그것은 이후 단계의 태스크들 사이의 간격은 현 단계의 간격보다 커질 수 없기 때문이다. 결국 제안한 CII 기법은 작업큐의 길이가 q라할 때 O(q)의 시간 복잡도를 가지고 스케쥴링 후보지를 검색할 수 있다. 결국 보통 q의 크기가 비교적 무시할 수 있는 상수라고 할 때, 기존의 기법에 대하여 제안한 기법이 갖는 각 프로세싱 노드에서 감수해야하는 정보수집을 위한 부가적인 오버헤드를 거의 무시할 수 있음을 알 수 있다. 그림 2는 CII 기법의 알고리즘을 C 언어로 기술한 것이다.

3.2 RT CORBA 시스템에서 CII 기법에 기초한 실시간 스케쥴링 기법

CII 기법을 사용하여 모든 프로세싱 노드들은 자신의 작업 큐에 새로운 태스크의 스케쥴링 가능 여부를 판단을 할 수 있으며, 만약 가능하다면 큐 상에서 삽입 가능한 공간 중 최대 또는 최소 공간의 위치에 대한 정보를 얻을 수 있다. RT CORBA 시스템에서 CII 기법을 바탕으로 한 실시간 스케쥴링 기법의 과정은 다음과 같다. 임의의 프로세싱 노드에 새로운 태스크가 생성되면 이 태스크가 CII기법에 따라 해당노드에서 처리 가능한지를 판단하게 된다. 만약 처리 불가능하다면 RT ORB에게 태스크 스케쥴링 요청을 한다. 이때 사용되는 인터페이스가 TDMI이다. 그러면 RT ORB는 태스크에 대한 RT-Environment를 생성하고 해당 객체를 활성화시킨다. 그리고 활성화된 객체를 서비스할 수 있는 프로세싱 노드를 객체 참조자를 이용하여 찾아낸 후, 새로운 태스크에 대한 정보(RT-Environment)를 전달하여 처리 가능 여부를 문의한다. 요청을 받은 각각의 프로세싱 노드들은 CII 기법을 사용하여 각 지역 작업 큐에 태스크가 스케쥴링이 가능한가를 판단하고 요청 노드에 자신의 상태 정보를 전달한다. 그러면 요청 노드는 처리 가능 노드들 사이에 간단한 부하균등화 원칙에 따라 최적 노드를 결정하여 태스크를 할당한다. 할당받은 프로세싱 노드는 POA를 통해 서번트를 실현하여 최종적으로 태스크를 수행하게 된다. 만약 처리 가능 노드가 존재하지 않는다면 생성된 태스크는 시스템에서 제거되게 된다. 만약 서비스의 QoS를 정의하여 태스크의 가중치를 정의한 경우 CII 기법 적용시 해당 태스크보다 가중치가 낮은 태스크들은 무시될 수 있으며, 최종적으로 태스크 할당시 가중치가 낮은 태스크와 새로운 태스크가 교체되어진다.

4. CII 알고리즘 분석

CII 기법은 보다 자세한 시스템 정보를 가지고 스케쥴링을 하는 것으로 기존의 우선순위에 바탕을 둔 EDF 기법 및 MRF 기법보다 정확한 스케쥴링이 가능하다. 그림 3은 EDF와 MRF 기법 보다 효율적 스케쥴링을 CII 기법이 제공하는 예를 보인 것이다. EDF 기법에서는 태스크 a로 인하여 연쇄적으로 b와 c가 블록되는 결과를 만들지만, CII 기법은 제한시간 내에 처리 불가능한 a를 제거하므로 태스크 b와 c의 처리를 보장할 수 있다. MRF 기법의 경우 태스크 b의 여유시간이 a의 여유시간보다 작기 때문에 b에 우선순위를 두어 스케쥴링을 하므로 태스크 a는 블록되는 결과를 가져온다.[11][12] 하지만 CII 기법에서는 태스크 a, b의 순서로 스케쥴링 되어 a, b의 처리를 보장할 수 있다.

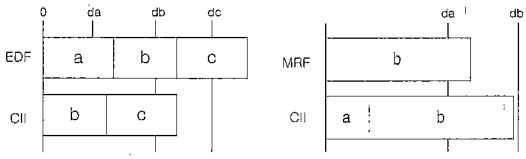


그림 3 CII 기법과 기존의 실시간 스케쥴링 기법의 비교

전 절에서와 같이 CII 알고리즘은 EDF나 MRF보다 효율적인 스케쥴링 결과를 나타낸다는 것은 명확하다. 그것은 스케쥴링에 사용하는 정보량이 CII 기법이 보다 많다는 것에 기인한다. 결국 CII 기법의 핵심은 스케쥴링에 필요한 정보수집 오버헤드에 있다고 말할 수 있다. 본 절에서는 CII 기법이 실제 분산환경에서 효율적으로 적용되기 위한 오버헤드의 제한 조건을 분석한다.

CII 기법의 분산환경 하에서의 효율성을 입증하기 위하여, 먼저 제한시간 **d**와 제한시간의 비율, ***r***(deadline ratio)을 다음과 같이 정의한다.

【가정】 jt= 태스크의 평균 처리시간, t= 현재시간, d= 제한시간

【정의】 $d = jt + jt \times r + t$

$$r = (d - jt - t)/jt \quad \text{식 (2)}$$

식(2)에서 우리는 r이 임의의 태스크가 스케줄링 순서를 양보할 수 있는 태스크 수의 최대값임을 알 수 있다. 만약 시스템의 작업 큐의 길이가 $r+1$ 이상이 될 경우, EDF와 같은 기존의 스케줄링 기법들은 연속적인 태스크 블록을 야기시킬 수 있다. 하지만 CII 기법은 처리 불가능한 태스크들을 제거함으로써 연속적인 태스크의 블록을 방지함은 물론, 작업 큐의 길이를 최적치인 $r+1$에 가깝게 유지함으로써 시스템의 태스크 처리량을 최적화 할 수 있다.

CII 기법이 분산환경에서 그 성능을 보장하기 위해서는 다음과 같은 조건을 만족시켜야 한다. 현재 임의 노드의 작업 큐 마지막에 스케줄링된 태스크의 처리를 보장하기 위해서, 태스크의 제한시간에서 현재 작업 큐에 존재하는 모든 태스크의 처리시간을 빼고 남은 여유시간이 모든 태스크의 스케줄링에 사용된 CII 오버헤드의 합보다 커야 한다. 이는 다음의 식으로 표현될 수 있다.

$$d - (jt \times q + t) > O \times q$$

(q= 평균 작업큐의 길이, O= CII 오버헤드)

$$jt + jt \times r + t - (jt \times q + t) > O \times q$$

(d= jt + jt×r + t; 식(2))

$$(1 + r - q) \times jt > O \times q$$

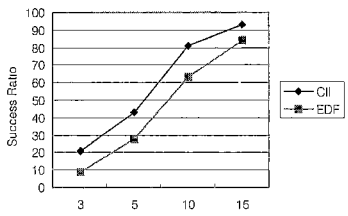
그러므로

$$O < (1 + r - q) \times jt/q \quad \text{식(3)}$$

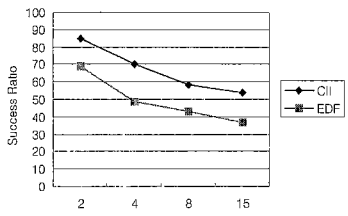
식(3)과 같이 CII 기법의 오버헤드는 제한을 갖는다. CII 기법은 Hard Real-Time 스케줄링을 제공하므로 실제 시스템에서 작업 큐의 길이 q는 $1 + r$ 보다 작은 값을 갖는다. 만약 $q=r$ 이라면 CII의 오버헤드 O는 $O < jt/q$ 의 조건을 만족시켜야 한다. 하지만 일반적으로 CORBA 시스템에서 jt가 태스크 이주에 이득을 얻을 수 있는 크기라고 할 때[2, 3], O의 상한으로서 jt/q는 충분히 큰 값이라 할 수 있다. 또한 O를 최소화하기 위하여 최초 스케줄링 가능 응답 서버에 태스크를 할당하는 방법을 채택할 수도 있다. 이런 경우 부하균등화의 면에서는 손실이 있지만 CII의 오버헤드는 최대 서버의 수에 반비례하여 줄어든다. 5장에서는 본 장에서 고려한 CII 오버헤드의 가정 하에 시뮬레이션을 수행하여 그 CII 기법의 스케줄링 성능을 입증한다.

5. 시뮬레이션 검증

제안한 기법과 기존의 기법에 대하여 큐잉이론(queueing theory)에 따른 이산사건 모델(discrete event model)을 통해 시뮬레이션을 SUN ULTRA 60(Solaris 2.6) 시스템에서 SUN WorkShop Compilers C/C++ 4.2를 이용하여 수행하였다. 먼저 프로세싱 노드 수와 각 기법의 태스크 성공률 사이의 관계를 알아보기 위해, 작업계수를 일정하게 한 뒤 전체 시스템의 프로세싱 노드 수를 변화시켜가면서 EDF와 CII 기법(최소공간 찾기)의 태스크 성공률을 비교하였다. 또한 역으로 프로세싱 노드의 수가 일정할 때 작업계수(∞ queue length)에 따른 두 기법의 태스크 성공률의 변화도 비교하였다. 그림 4의 (a)는 전자의 경우를, (b)는 후자의 경우를 도표로 정리한 것이다. 프로세싱 노드의 수가 증가할수록 시스템의 처리 능력이 증가하므로 태스크 성공률은 두 기법 모두 증가하는 양상을 보였다. 하지만 CII 기법의 스케줄링이 EDF 기법의 스케줄링보다 더 빠른 시스템 성능향상을 보였으며, 최적의 경우 20%의 성능 차이가 발생하는 것을 볼 수 있었다. 큐 길이가 증가한다는 것은 작업계수가 그만큼 커지는 것을 의미한다. 그것은 보다 많은 태스크가 생성되고, 큐에서 대기한다는 것을 의미한다. CII 기법의 스케줄링은 무의미하게 큐에서 대기하는 태스크의 수를 최대한 줄이므로 시스템의 성능 감소를 최대한 막을 수 있다. 하지만 EDF 기법에 따른 스케줄링은 큐의 길이가 증가할수록 무의미하게 큐에서 기다리는 태스크 수의 증가와 그에 따른 자원 낭비가 연쇄적으로 증가하므로 시스템의 성능 악화가 두드러지게 나타나게 되는 것을 볼 수 있었다.

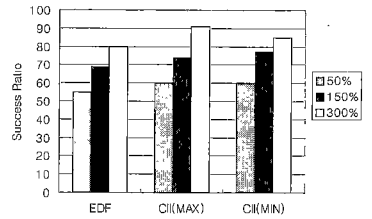


(a) 노드 수와 태스크 성공률의 변화

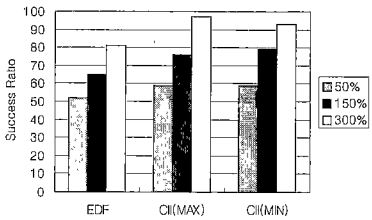


(b) 큐 깅이와 태스크 성공률의 변화

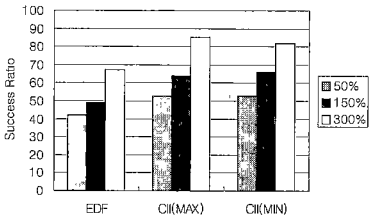
그림 4 노드 수와 작업계수에 따른 성능비교



(a) 태스크비 3:4:3



(b) 태스크비 5:3:2



(c) 태스크비 2:3:5

그림 5 태스크 트래픽에 따른 성능비교

다음으로 태스크의 트래픽 양상에 따른 두 기법의 성능을 비교하였다. 각 태스크의 처리시간을 10000ms를 기준으로 하여 1~400ms, 400~3000ms, 3000~10000ms의 세 구간으로 나누어 구간 비율이 3:4:3, 2:3:5, 5:3:2인 경우에 각각 제한시간 비율이 최대 50%, 150%, 300%인 태스크들로 평균 큐 길이가 15가 되도록 시스템의 태스크 트래픽을 구성하고 총 노드 수가 100개인 시스템에서 반복하여 시뮬레이션을 수행하여 EDF, CII(MIN;최소공간 찾기), CII(MAX;최대공간 찾기) 기법에 따른 스케쥴링의 평균 태스크 성공률을 구하였다. 그림 5는 각각의 결과를 도표로 정리한 것이다. 세 기법 모두 제한시간의 비율이 클수록, 또 태스크 수행시간이 짧은 작업의 비율이 클수록 좀더 나은 태스크 성공률을 보였다. CII(MIN)과 CII(MAX) 기법은 제한시간의 비율이 작을수록 공간 선택의 폭이 좁아지는 이유로 거의 같은 성능을 보였다. 하지만 실험에 의해 평균 제한시간의 비율이 250%이내일 때는 트래픽의 양상에 따라 다르지만 CII(MIN) 기법이 좋은 성능을 보일 때가 많은 것을, 250% 이상이면 CII(MAX) 기법이 뚜렷이 더 나은 성능을 보이는 것을 알 수 있었다. 이는 새로운 태스크의 여유시간과 이미 작업 큐에 대기 중인 태스크의 여유시간 중 어느 것을 활용하는가에 대한 문제라고 할 수 있다. CII(MIN) 기법의 경우 새로운 태스크가 삽입되는 곳은 여유가 작은 태스크들 사이일 경우가 높기 때문에 자신의 여유시간의 사용가능성이 작아지게 된다. 그러므로 새로운 태스크의 제한시간의 비율이 커질수록 CII(MAX) 기법이 더욱 유리하게 된다. 전반적으로 EDF 기법의 스케쥴링보다 CII 기법의 스케쥴링이 최소 5%에서 최대 20%의 성능향상을 보였다. 이것은 노드 수와 작업계수에 따른 비교에서의 성능차와 일치하는 것을 알 수 있다.

5. 결 론

기존의 단일 프로세서 시스템에서는 스케쥴링을 적용하는 태스크들의 수행시간의 단위가 작고 전문화된 태스크들이 아닌 범용적인 태스크들로서, 실행 시간의 정확한 예측 및 태스크간 미세한 스케쥴링 조정이 사실 불가능하여 Hard Real-Time 스케쥴링을 적용하는데 무리가 있었다. 하지만 CORBA 시스템에서는 각 서버가 전문화되어 있을 뿐만 아니라, 수행되는 각 객체들에 대한 메타 데이타들이 관리되고 있다. 또한 CORBA 시스템에서 수행되는 태스크들은 대개 부하이주 가치가 있는 비교적 작업 크기가 큰 태스크들로 구성된다. 그러므로 CORBA 시스템은 Hard Real-Time 스케쥴링을 위한 충분한 조건을 갖추고 있다고 할 수 있다. Hard Real-Time 스케쥴링을 통한 보다 정확한 자원 관리는 비교적 큰 통신 부담을 감수해야 하는 CORBA 시스템에서 불필요한 부하이주 및 프로세싱 자원 낭비를 최대한 줄일 수 있다.

기존의 스케쥴링 시스템은 우선순위라는 간접적인 스케쥴링 척도를 계산하여 전역 스케쥴링을 수행하므로, 다양한 시스템 변수가 작용하는 분산시스템에서는 각 태스크의 우선순위 조정을 위한 복잡한 관리 수단이 필요로 한다. 또한 우선순위 할당과 분산 시스템에서의 부하할당 및 부하균등화 과정이 분리되어 있어, 우선순위에 따라 부하할당 및 부하균등화를 이행하는데 복잡한 단계를 필요로 한다. 하지만 제안한 CII 기법에 바탕을 둔 실시간 스케쥴링 기법은 전역 우선순위의 지역 우선순위로의 단순 매핑에 따른 스케쥴링이 아니라, 시스템 상태에 따른 유연한 스케쥴링을 제공하므로 부가적인 복잡한 부하할당 및 부하균등화 과정이 불필요하다. 또한 우선순위 위주의 할당이 아니므로 비실시간 태스크들에 대해서도 응답시간을 최대한 단축시킬 수 있다.

참 고 문 헌

[1] Steve Vinoski, "CORBA: Integrating Diverse Applications Within Distributed Heterogeneous Environments," IEEE Communications Magazine, pp.1-12, Feb. 1997.

[2] Paul Haggerty, Krishnan Seetharaman, "The Benefits of CORBA-Based Network Management," Communication of the ACM, Vol. 41 No. 10, pp. 73-79, Oct. 1998.

[3] Lisa Cingiser Dipippo, Victor Fay Wolfe, Thomas Wheeler, Russell Johnston, "A Scheduling Service for a Dynamic Real-Time CORBA System," Proceedings of the Twenty-Second Annual International Computer Software & Applications Conference, pp.608-613, Aug. 1998.

[4] Victor Fay Wolfe, Lisa Cingiser Dipippo, "Real-Time CORBA," Proceedings of the Real-Time Technology and Applications, pp.148-157, Jun. 1997.

[5] Victor Fay-Wolfe, Lisa C. DiPippo, Gregory Cooper, Russell Johnston, Peter Kortmann, Bhavani Thuraisingham, "Real-Time CORBA," IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, Vol. 11, No. 10, pp. 1073-1089, Oct. 2000.

[6] Douglas C. Schmidt, Aniruddha S. Gohale, Timothy H. Harrison, and Guru Parulkar, "A High-Performance End System Architecture for Real-Time CORBA," IEEE Communications Magazine, Vol. 35 No. 2, pp.72-77, Feb. 1997.

[7] Zhonghua Yang, Chengzheng Sun, "CORBA FOR HARD REAL-TIME APPLICATIONS: SOME CRITICAL ISSUES," Operating Systems Review, Vol. 32 No. 3, pp.64-71, Jul. 1998.

[8] Maurizio A. Bonuccelli, M. Claudia Clo, "EDD Algorithm Performance Guarantee for Periodic Hard Real-Time Scheduling in Distributed Systems," Proceedings of the 13th International Parallel Processing Symposium & 10th Symposium on Parallel and Distributed Processing, pp.668-677, Apr. 1999.

[9] Giorgio C. Buttazzo, Fabrizio Sensini, "Optimal Deadline Assignment for Scheduling Soft Aperiodic Tasks in Hard Real-Time Environments," IEEE TRANSACTIONS ON COMPUTERS, Vol. 48, No. 10, pp.1035-1052, Oct. 1999.

[10] OMG, "Realtime CORBA," electronic document, http://www.omg.org/docs/orbos/98-10 -05.pdf.

[11] Lichen Zhang, Jiwu Huang, Yi Zheng, "Scheduling Algorithms for Multiprocessor Real-Time Systems," Proceedings of the 1997 International Conference on Information, Communications & Signal Processing, Vol. 3, pp.1470-1474, Sep. 1997.

[12] J.W.-S. Liu, Real-Time Systems. Prentice-Hall, Mar. 2000.

백 승 민

1998년 2월 서강대학교 컴퓨터학과 졸업. 2000년 2월 서강대학교 컴퓨터학과 대학원 석사과정 졸업. 2000년 3월 ~ 현재 서강대학교 컴퓨터학과 대학원 박사과정 재학중

김 성 천

1975년 서울대학교 학사. 1979년 Wayne State University Computer Engineering MS. 1982년 Wayne State University Computer Engineering Ph.D. 1982년 ~ 1984년 California State University Computer Engineering 조교수. 1984년 ~ 1985년 금성반도체(주) 책임연구원. 1985년 ~ 현재 서강대학교 컴퓨터학과 교수. 관심분야는 Parallel Computer System, Interconnection Network, Process Scheduling, Faulttolerance System