그리드 컴퓨팅 시스템에서의 두 개의 트리를 이용한 집합 통신

(Two-Tree Collective Communication in Grid Computing Systems)

차 광 호 † 이 정 희 †† 한 동 수 ††† 유 찬 수 ††††
(Kwangho Cha) (Junghee Lee) (Dongsoo Han) (Chansu Yu)

요 약 본 논문은 이종 네트워크로 구성되어 통신 시간이 상대적으로 길며 또한 지연(delay)이 일정하지 않은 그리드 환경에서 사용되는 MPI의 집합 통신(collective communication)에 관한 연구이다. 일반적으로 효과적인 집합통신에 있어서는 네트워크 정보를 이용한 통신 스케쥴을 작성하고 이것에 따라서 실제로 통신하는 전략을 선택하고 있다. 전통적인 그리드 집합통신 방식에 있어서도 네트워크 정보가 정확하다는 가정 하에 동일한 접근방식으로 집합통신을 수행하고 있는 상황이다. 하지만 네트워크 환경이 쉽게 변하는 그리드 환경에 있어서는 네트워크 정보가 정확하지 않거나 네트워크 fault가 발생할 경우, 전통적인 방식의 집합 통신은 그 통신 성능이 급격히 감소할 수 있다. 본 논문에서는 그리드에서 집합통신을 위한 스케쥴링의 한 방법으로 TTCC(Two-Tree Collective Communication) 알고리즘을 제안한다. TTCC 알고리즘은 서로 다른 에지들로 구성된 두 개의 통신 트리를 이용해 불확실한 네트워크 상태에서도 효과적이고 신뢰성 있는 스케쥴링을 제공할 수 있다. TTCC 알고리즘의 효과는 시뮬레이션을 통해 검증하였다. 시뮬레이션 결과 지연(latency)이 커짐에 따라 나타날 수 있는 성능 감소량이 TTCC를 이용할 경우 종래의 스케쥴링 알고리즘을 이용하는 것보다 우수한 것으로 판명되었다.

키워드 : 집합 통신, 그리드, 이종 네트워크, 네트워크 정보, 네트워크 웨더 서비스(NWS)

Abstract This paper studies the collective communication in the grid computing environment, which is characterized by the combination of heterogeneous networks as well as uneven, long communication delay. Efficient collective communication requires communication schedule, which in turn requires network information. When the network information is not accurate or network faults occur, the performance of collective communication can be markedly degraded. This paper proposes TTCC(Two-Tree Collective Communication) for scheduling collective communication in the grid. It provides an efficient and reliable schedule even in this unfavorable network condition by maintaining two disjoint communication trees. Benefits of the proposed method are manifested via simulation, where the performance degradation with TTCC is much slower than those using conventional scheduling algorithms.

Key words : broadcast, collective communication, grid, heterogeneous network, network information, and NWS

1. 서 론

다수의 이기종 컴퓨터와 디바이스가 고속 네트워크에 연결된 그리드 컴퓨팅은 최근 원거리 네트워크(WAN)에서 산재되어 있는 컴퓨팅 자원들의 효과적인 이용 방안과 대용량 계산능력의 요구에 대한 효과적인 해결책을 제시함으로써 큰 관심을 끌고 있다[1]. Globus[2], Legion[2]과 같은 그리드 프로젝트에서는 그리드 환경에서 필요한 기술 및 발생가능한 문제들을 제기하고 실험함으로써 이미 실행가능성을 보여주는 노력을 많이 기울였다. 그리드 환경에서 제기되는 대부분의 문제들은 그리드 환경의 특성으로 인해 나타나는 불확실성과 관련되어 있다. 이러한 불확실성을 야기하는 원인으로는, 이종(heterogeneity), 예측불가능 구조(unpredictable

† 정 회 원 : 한국과학기술정보연구원 슈퍼컴퓨팅센터 연구원
khocha@kisti.re.kr
†† 비 회 원 : 한국전자통신연구원 임베디드 S/W센터 연구원
lake@etri.re.kr
††† 종신회원 : 한국정보통신대학교 공학부 교수
dshan@icu.ac.kr
†††† 비 회 원 : 클레베랜드주립대학교 교수
c.yu91@csuohio.edu
논문접수 : 2003년 8월 28일
심사완료 : 2004년 7월 13일

structure), 동적 성향(dynamic behavior), 다수의 관리 도메인(multiple administrative domain) 등을 들 수 있다[4].

자원들이 공유되어 있는 그리드에서는 자원들을 누구나 사용할 수 있으므로, 각 자원이 제공할 수 있는 유효 성능은 일정하지 않은 특징을 가지고 있다. 계산 그리드(computational grid)에는 이러한 불확실성을 극복하기 위해 자원 성능 예측 시스템과 같은 서비스가 필수 불가결한 요소이다. 최근의 한 연구에서는, 시스템 전체 자원의 성능 측정 기능, 각 자원의 성능 예측 기능, 스케쥴러에게 예측된 정보들을 분산시키는 기능의 세 가지 기본 기능을 갖추고 있는 자원 성능 예측 시스템(predictor)을 보여주었다[5]. 그러나 자원 성능 예측 시스템의 서비스 기능을 활용하는 경우에도 예측 시스템과 스케쥴러 간의 공간적 거리 뿐 아니라, 자원들 간의 공간적 거리 또한 무시할 수 없기 때문에, 수집된 정보들의 정확성을 확신할 수 없다. 이것은 자원 성능 예측 시스템이 전체 시스템에 영향을 주지 않는 범위에서 정보를 수집하기 위해서는, 비교적 큰 간격의 주기로 네트워크 정보를 수집해야 하기 때문이다. 정보의 정확성에 관한 문제는 원거리 네트워크(WAN)인 경우 더욱 중요하다. 만약 부정확한 정보를 근거로 스케쥴링할 경우 적절치 못한 결과를 초래할 수 있기 때문이다[6]. 특히 그리드 환경의 메시지 패싱 병렬 응용 프로그램(message-passing parallel application)에 있어서는 계산 자원들 간의 지연이 크기 때문에 통신 스케쥴링의 성능은 해당 응용 프로그램의 전체 성능을 결정짓는 중요한 요소이다. 그 외에도 집합 통신은 프로그램 작업을 단순화시켜 병렬 프로그램의 개발에 있어서도 중요한 기능이므로, 집합통신의 스케쥴링에 대한 고려가 필요하다[7,8].

집합통신 프리미티브는 내부적으로 point-to-point 통신 프리미티브들로 구성되어 있다. 이 point-to-point 통신 프리미티브를 어떻게 스케쥴링 하느냐에 따라 집합통신의 성능이 많이 달라지므로 이는 병렬 프로그램의 전체 성능에 큰 영향을 미치게 된다. [8], [9]와 같이 집합통신 프리미티브의 최적의 통신 스케쥴링에 대한 연구가 많이 있다.

이론적으로 이종 환경(heterogeneous environment)에서 최적의 트리생성을 위해서는 네트워크 크기에 비례하여 시간 비용이 기하급수적으로 증가하는 것으로 알려져 있다. 이러한 이론적 한계를 극복하기 위하여 *FEF(Fastest-Edge First)*나 *ECEF(Earliest Completing Edge First)*와 같은 휴리스틱 알고리즘이 제안되기도 하였다[7,9]. 이 알고리즘들은 차선의 트리(suboptimal solution)를 찾는데 polynomial 시간복잡도의 비용을 소요한다. 그러나 이러한 휴리스틱 알고리즘들은 모두 부정확한 정보나 예측 불가능한 네트워크 결점으로 인해 성능에 미칠 영향은 고려하지 않고 있다. 따라서 이 방식은 해당하는 통신 그룹의 통신 시간 동안 예측값과 다르게 네트워크 성능이 변하는 경우에는 심각한 통신 성능 저하를 가져올 가능성이 높다. 이에 반해 본 논문에서 제안하는 TTCC(Two Tree Collective Communicatoin)는 일정한 성능을 제공하기 위해 OT(Original Tree)와 RT(Redundant tree)라는 2개의 통신 트리를 이용하여, 자원 성능에 대한 예측정보가 부정확하더라도 그에 따른 영향을 줄일 수 있다.

TTCC와 ECEF[9]의 성능을 비교하기 위해 두 알고리즘을 이용한 시뮬레이션을 수행한 결과, 네트워크 정보가 부정확하거나 노드에 결점이 많이 발생할수록, ECEF의 성능은 급격히 감소하는 반면, TTCC의 성능은 좀더 완만한 감소를 보여주었다. ECEF가 0%에서 485%까지의 범위로 급격한 성능 감소가 발생하는 반면, TTCC의 성능은 19%에서 104%의 완만한 감소를 보여 빈번하게 변하는 네트워크 환경에서 TTCC 방식의 우수성이 입증되었다.

본 논문의 구성은 다음과 같다. 우선 2장에서는 집합통신에 대한 관련연구와 NWS와 같은 네트워크 정보 서비스를 소개한다. 다음으로 3장에서는 제안된 스킴인 TTCC에 대하여 기술한다. 마지막으로 4장에서는 TTCC의 시뮬레이션 결과로 성능 측정에 대하여 기술하고, 5장에서 결론을 기술하며 끝을 맺는다.

2. 관련 연구(Related Work)

이 장에서는 네트워크 정보를 이용한 스케쥴링에서 필요로 하는 자원 성능 예측에 관한 최근 연구들을 살펴본다. 그리고 계산 그리드에서의 집합 통신의 통신 스케쥴링에 관한 선행 연구들에 대하여 소개한다.

2.1 자원 성능 예측

NWS(Network Weather Service)은 네트워크 정보 수집에 널리 이용하는 툴로서 Globus, AppLes, Legion 프로젝트 등에서 사용한다[5,10]. NWS에서는 네트워크 성능 기록을 분석한 후 9가지 방법론에 기반하여 네트워크 성능을 예측한다. 그 다음으로, 평균 에러를 가장 작게 발생시키고 성능이 가장 높은 프레딕터(predictor)를 동적으로 선별한다. 6개의 슈퍼컴퓨팅 사이트들을 대상으로 한 실험에서, 성능이 가장 좋은 프레딕터는 분명하지 않고 자원마다 다양하게 나타났다. 또한, 일반적인 인터넷 트래픽에서 분리된 이더넷(Ethernet) 단독 사이트에서도 마찬가지로 큰 폭의 성능 변화를 보여주었다[5]. 일반적으로, 원거리 네트워크(WAN)에서는 네트워크 성능의 변동이 훨씬 크고, 예측 결과에 에러가 발생

한다고 알려져 있다.

최근 인터넷의 지연(delay)과 홉수(hop-count)를 측정한 한 연구에서는 지연과 홉수 사이에 밀접한 관계가 없다는 것을 보여주었는데, 이것은 물리적 거리나 홉수 한 가지만으로는 네트워크 성능을 신뢰성 있는 예측을 할 수 없다는 것을 의미한다. 따라서 여러 요소를 고려한 정확한 네트워크 정보 수집의 중요성이 더욱 강조되고 있다. 이것이 NWS와 같은 기능을 가진 툴의 사용에 대한 타당성을 뒷받침한다.

2.2 집합 통신 스케쥴링

집합 통신은 동종 환경[12,13]뿐 아니라 이종의 다양한 컴퓨팅 환경[7,9,14]에 대해서도 연구되어 왔다. 동종 병렬 시스템에서는 지연이나 오버헤드 같은 집합통신 파라미터가 일정하기 때문에 집합통신을 위한 통신 트리가 쉽게 생성된다[12]. 동종 시스템과 달리, 이종 시스템의 통신 파라미터는 다양한 값을 취하며, 이것은 물리적 매체에서 통신규약까지 다양한 컴퓨팅 노드와 네트워크 특성을 나타낸다. 네트워크를 구성하는 요소가 다양하기 때문에 동일한 통신 트리를 구성하기는 어렵다[7]. 이러한 환경에서 최적의 통신 트리를 찾는 문제는 NP-hard로 알려졌고, 기하급수적 시간복잡도(exponential time complexity)인 $O(N^2 2^{2N})$를 요구한다. 여기서 N은 계산 노드들의 수이다. 이러한 문제를 풀기 위해 SPOC(Speed-Partitioned Ordered Chain), FNF (Fastest-Node First)[7], FEF(Fastest_Edge First), ECEF(Earliest Completing Edge First)[9]와 같은 여러 휴리스틱 알고리즘이 제안되었다[9]. FEF 방법은 루트로부터 연결된 노드 중 통신 비용을 최소로 하는 에지를 선택한다. 같은 프로시져를 반복하면서 이미 선택된 노드들로부터 도달가능한 하나의 노드를 선택한다. FEF의 시간복잡도는 $O(N^2 \log N)$을 갖는다. 다른 알고리즘으로 ECEF는 FEF와 유사하지만 에지를 선택할 때 다른 방법을 사용한다. ECEF는 송신자의 준비 시간을 고려해 통신 비용과 준비 시간의 합이 최소가 되는 에지를 선택한다. Look-ahead 알고리즘은 ECEF에 다음 통신 비용을 나타내는 look-ahead 값을 더한다. ECEF와 Look-ahead 알고리즘의 시간복잡도는 각각 $O(N^2 \log N)$와 $O(N^4)$를 나타낸다. 이 알고리즘들의 특징은 모두 스케쥴링에서 사용된 네트워크 정보들이 정확할 것이라는 가정하에 통신트리를 생성한다는 것이다. 이로 인해 네트워크 정보가 부정확할 경우 실제 통신이 발생할 때 성능저하를 초래할 수 있게 된다.

3. Two-Tree Collective Communication(TTCC)

2장에서 소개된 휴리스틱 알고리즘(heuristic algorithm)들은 이종 환경에서의 집합 통신에 대해서는 합리적인 해결책일 수 있지만 부정확한 네트워크 정보와 예측 불가능한 문제에 대해서는 고려하지 않고 있다. 그 결과, 이들 알고리즘을 기반으로 한 스케쥴링의 경우, 정보의 정확성에 대한 오차가 커질수록 성능이 저하된다. 이 장에서는 정보의 불확실성에 직면하고 있는 계산 그리드를 위한 신뢰성 있는 스케쥴링 방법으로 두 개의 트리를 이용한 집합통신(Two-Tree Collective Communication)을 제안한다.

네트워크 성능측정의 빈도가 잦으면 전체 성능 저하를 초래할 수 있으므로 성능측정 빈도수를 제한해야 한다. 따라서 모든 시점에 대하여 정확하게 네트워크 성능을 예측하기는 어렵다. 이로 인해, 네트워크 정보의 예측값에 오차가 발생할 수 있다. TTCC(Two-Tree Collective Communication)는, 네트워크 정보를 적절히 다룸으로써 휴리스틱 알고리즘과 달리 네트워크 정보의 부정확성으로 인해 발생할 수 있는 문제점들을 완화시켜주는 접근 방법이다. TTCC에서 제안하는 방법은, 통신 트리 두 개를 이용해 두 경로로 들어오는 것 중 더 빨리 수신을 완료할 수 있는 노드를 선택하여 수신한다. 이로 인해, 통신이 실제 일어날 때 하나의 통신트리를 생성해 네트워크 정보가 오차가 발생할 경우, 성능이 감소하는 것을 보완하는 효과가 있다. TTCC는 이를 위해, 동시 두 개의 경로로부터 데이타 수신을 받을 경우 TCP/IP 스택에서 최적의 경로를 선택해주는 것을 가정한다.

이 장에서는, TTCC 통신 모델, TTCC의 세부 설명과 논리코드, 그리고 TTCC의 성능 테스트를 위한 시뮬레이션 방법에 대하여 기술한다.

3.1 통신 모델

ECEF의 중요한 네트워크 정보로는 두 노드들 간의 통신 비용이 존재하고, 통신 비용은 $latency + \left(\frac{message\ size}{bandwidth}\right)$로 구할 수 있다. P. B. Bhat[9]의 연구와 마찬가지로 TTCC도 대역폭(bandwidth)과 네트워크의 지연(latency)이 네트워크 모델의 중요한 요소로 작용한다.

이 모델에서 각 노드는 동시에 많아야 한 번의 송신과 한 번의 수신 연산(operation)을 수행하는 것을 가정한다. 여기에 추가로, 먼저 보내는 송신자로부터 데이타를 받는 중에도 각 노드는 다른 송신자로부터 보내진 신호(signal)를 구분할 수 있다고 가정한다. 데이타를 받고 있는 노드가 다른 송신자로부터 신호를 받을 수 있다면, 수신하고 있던 노드는 잠시 데이타 수신을 멈추고, 각 두 송신자 노드들로부터 수신 가능한 속도를 비교한다. 여기서 브로드캐스트 연산을 수행하는 두 노드들 간의 평균 수신 시간의 일정 부분을 차지하는 두 노

드들 중 한 노드를 선택하는데 걸리는 시간을 상수 α로 고려한다.

3.2 Two-Tree Collective Communication 이론

P. B. Bhat의 연구[9]에 따르면, ECEF의 성능은 Look-ahead 알고리즘[9]의 성능과 유사하지만, 시간복잡도 측면에서는 ECEF가 Look-ahead보다 좋게 나타난다. 따라서 TTCC를 설계하는데, Look-ahead보다 ECEF의 알고리즘을 이용하기로 한다.

네트워크 정보의 부정확성 문제로 인한 집합통신 전체 성능의 감소를 방지하기 위하여 ECEF를 이용하여 집합통신 스케쥴링을 위한 통신 트리를 여러 개 보유한다면, 통신에 참가하는 노드들은 루트로부터 여러 개의 경로(path)를 보유하게 된다. 즉, 노드(A)는 여러 개의 수신 경로를 가질 수 있게 된다. 이로 인해 각 노드는 여러 개의 경로로부터 수신이 빨리 완료되는 경로를 선택함으로써 전체 성능을 향상을 도모할 수 있게 된다. 다시 말해 A로 가는 여러 경로 중의 어느 노드에서 심한 부하가 발생한다면, 다른 통신 트리를 이용함으로써 데이타 수신 속도를 보장받는 경로를 선택한다. 따라서 실행 시간 동안 부분적으로 최적의 경로를 선택함으로써 통신성능의 감소를 초래하는 요인으로부터 ECEF의 성능이 급격하게 감소하는 것을 방지할 수 있게 된다.

그림 1에서 특정 상황에서 여러 개의 트리를 사용하여 얻을 수 있는 성능 향상을 보여준다. 트리의 개수가 증가할수록 개선되는 정도는 점점 감소하고, 스케쥴링에 소요하는 시간과 노드 선택의 횟수는 증가한다.

이를 $Completion\ Time = \beta \frac{1}{NmberOfTrees}$의 식으로 나타낼 수 있다. 여기서 Completion Time은 브로드캐스트 함수의 완료시간, NumberOfTrees은 트리의 개수, β는 임의의 상수를 나타낸다. 성능향상 정도가 트리 수가 점차 증가함에 따라 감소하는 것은 트리 생성 시간과 네트워크 부하가 증가가 한 원인이 된다.

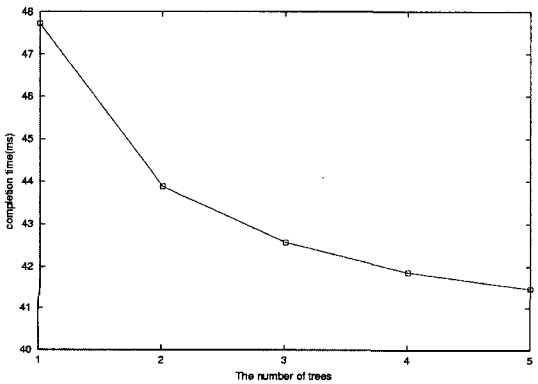

그림 1 트리 수에 따른 성능 향상

개선의 정도를 최대화하는 동시에 여러 개의 통신트리를 사용함으로써 발생하는 스케쥴링 비용과 네트워크 부하를 최소화하기 위해, 그림 1의 실험 결과에 따라 TTCC에서는 2개의 트리를 사용한다. 또한 전체 성능은 통신 트리에 의해 주로 결정되므로 스케쥴링을 하는데 소요한 시간은 무시한다.

두 개의 트리를 이용한 집합통신 스케쥴링 방법에 ECEF 알고리즘을 수정하여 사용하였다. 다시 말해, ECEF의 알고리즘으로 오리지널 트리 T_0를 생성하고, T_0에 포함된 에지를 제외한 나머지 에지들을 이용해 같은 ECEF 방법으로 리던던트 트리 T_1을 생성한다. 이 때 T_1은 T_0 생성된 후에 만들어지기 때문에, 네트워크 정보에 에러가 없다면 T_1을 사용하는 것보다 T_0을 사용하는 것이 성능이 좋을 것이다. 그러나 T1은 네트워크 예측 정보가 부정확할 경우 효과적인 경로를 찾을 때 통신 성능을 보완하는 역할을 한다. 그림 2는 리던던트 트리 생성을 위한 논리코드를 보여준다.

두 개의 트리가 재귀 경로를 갖는다면, 즉, 노드 A에서 노드 B까지 경로와 노드 B로부터 노드 A까지 경로를 모두 갖는다면, 노드가 송신자에게 데이타를 다시 보내주는 것이 가능하다. 그러나 이것은 일반적으로 발생하는 상황이 아니므로, T_0의 에지로 선택할 경우 T_1 생성 시 이미 선택한 에지를 재사용하지 않도록 이 에지는 제거한다. 그림 2의 논리코드를 보면 만약 T_0이 평면트리(flat tree)라면 T_1을 만들 수 없게 된다. 그러나 이러한 경우는 드물게 발생하므로 루트 노드로부터 다른 모든 노드들이 직접 연결된 경우는 고려하지 않았다.

TTCC를 이용할 경우, 이 두 개의 트리가 생성된 후에야 통신을 수행한다. 수신 단계에서 한 노드가 경로를 통해 데이타를 받고 있으면, 그 노드는 다른 수신 경로를 단절시킨다. 송신 단계에서는, 노드가 T_0의 경로에서 T_1의 경로로 두 개의 트리의 모든 경로를 통해서 데이타를 송신하려고 한다. 그림 3은 노드의 각 단계별 수행 동작을 보여준다. 일단 노드 n_i가 송신자로부터 데이타의 수신을 완료하면(a), 다른 입력 경로로부터 들어오는 데이타는 무시하게 된다(b). OT(Original Tree)로부터 데이타 수신을 완료하기 전에 n_i가 데이타를 다른 노드로부터 받게 되면 그 노드는 α 시간동안 결과를 검토한 후 최종적으로 받을 송신자를 선택한다. 노드에서 수신이 끝나면, 송신 단계에 들어간다. 최적의 솔루션을 제공하는 통신트리로 T_0을 생성했으므로, 노드 선택 시 T_1을 사용할 때보다 T_0을 사용할 때 성능이 더 좋을 것으로 기대한다. 따라서 n_i은 T_1에 있는 경로를 선택하기보다 T_0에 있는 경로를 선택할 것이다(e).

```
procedure Redundant Tree (G:weighted bi-directional graph which has
                                                vertice V and edge E)
// G = (V,E)
// E = {e_ij | i∈E,j∈E, and i≠j}
// Y is used for T_0
// R is used for T_1

begin
 Y := ∅
 R := ∅

 Y := Y ∪ {v_R}      // v_R is the vertice representing root node
 V := V - {v_R}

 for i=1 to n-1      // n is the number of vertice of G
 begin
   e_yz = ECEF(Y,V,E) // select the edge which satisfy ECEF rule,
                      // y∈Y, z∈V

   Y := Y ∪{z}
   V := V - {z}

   E := E - {e_yz}
   E := E - {e_xy}    // for redundant tree
 end

 R := R ∪ {v_R}

 for i=1 to n-1
 begin
   e_yz = ECEF(Y,V,E)

   R := R ∪{z}
   V := V - {z}

   E := E - {e_yz}
 end
end
```

그림 2 리던던트 트리(Redundant tree)를 생성하는 논리코드

TTCC와 ECEF를 비교하기 위해 두 노드 간의 통신 시간을 고려해 보자. 두 노드 간의 통신 비용은 앞 절 (3.1)에서 말한대로 $latency + \left(\frac{message\ size}{bandwidth}\right)$라고 할 수 있다. ECEF 알고리즘을 이용해 생성된 통신 트리에서 노드 B에서 데이타 수신이 완료되는 시간을 구해보면, ECEF를 이용해 노드 B는 노드 A로부터 데이타 수신을 받도록 스케쥴링되었다면, 노드 A, B 간의 전송 시간은 $T_{AB} = latency + \frac{messagesize}{bandwidth_{AB}}$의 시간이 소요된다. TTCC의 경우 ECEF를 이용하여 두 개의 통신 트리를 생성하므로 ECEF와 같은 통신 트리와 첫 번째 트리에서 사용한 에지를 제외한 또 다른 트리를 생성하게 된다. 노드 B는 ECEF와 마찬가지로 A로부터 수신을 하거나, 또다른 노드 C로부터 수신하게 될 것이다. 이 때 TTCC는 두 개의 통신 트리로부터 데이타 수신이 빨리 완료되는 것을 선택하기 때문에 노드 B의 전송시간은 $Min\, T_{AB}, T_{CB}$이 된다. 따라서, TTCC의 경우 TTCC로 인한 네트워크 부하가 작다면 항상 ECEF보다 전송시간이 작거나 같게 소요된다. 그러나 여기서 더욱 중요한 것은 네트워크 정보의 오류나 네트워크 결점으로 인하여 스케쥴링 당시의 상황과 통신이 실제 발생할 때의 상황이 다르더라도 TTCC는 안정적인 통신속도를 보장받을 수 있다는 점이다.

아래 그림 4와 같이 지연값이 주어진 네트워크 상황에서 TTCC의 간단한 예제를 살펴본다. 집합 통신이 수행될 때 네트워크의 실제상황이 그림 4(a)와 같고, 그림 4(b)와 같이 네트워크 정보를 수집해 예측한 데이타가 실제 네트워크 데이타가 일치하지 않는다고 하자. 여기에서 에지에 나타난 수치는 지연 정보를 보여준다. 그림 4의 예제에서 모든 노드들 간에 에지가 존재하는 완전 접속망(fully connected network)을 예로 든 것은, 통신 단계와 네트워크 인터페이스의 충돌 횟수를 최소화하기 위해서이다[8].

이러한 상태에서 ECEF는 그림 5(a)와 같은 트리를

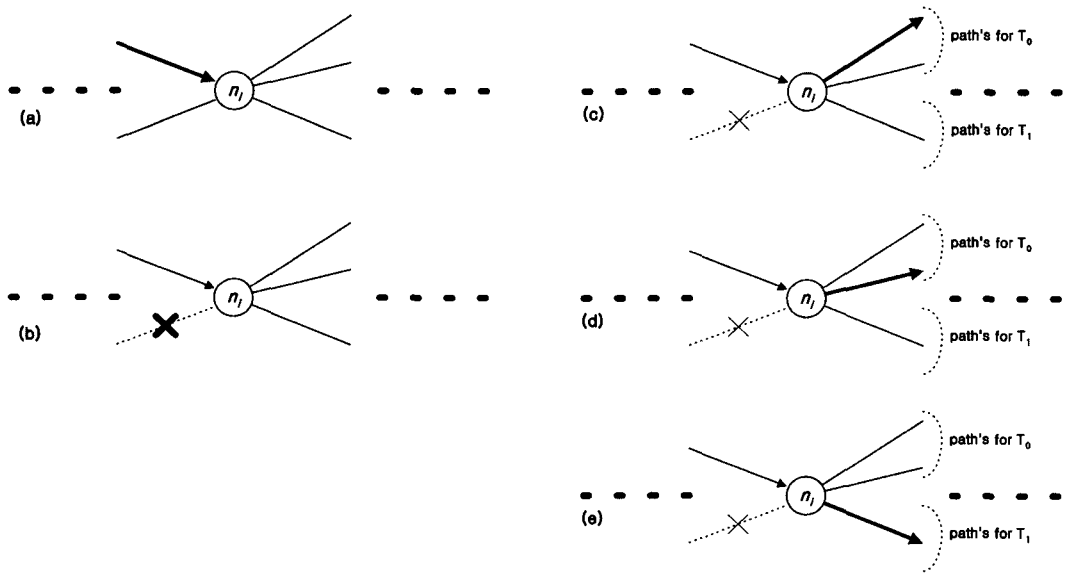

그림 3 TTCC를 수행하는 방법을 보여주는 예제; (a)~(b): 수신단계, (c)~(e): 송신단계

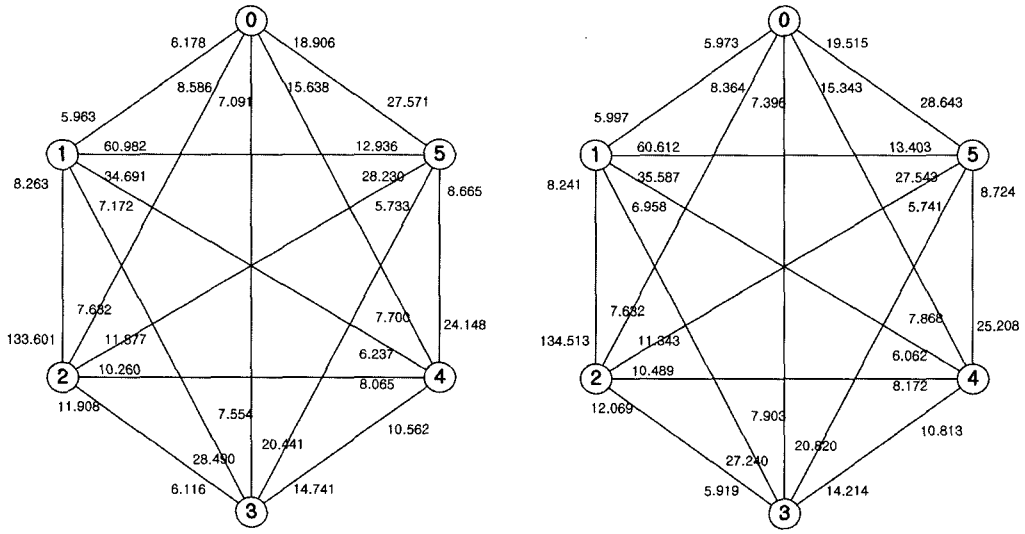

그림 4 실제 네트워크 데이타와 측정된 네트워크 정보: (a) 실제 네트워크 데이타, (b) 네트워크 정보(예측값)

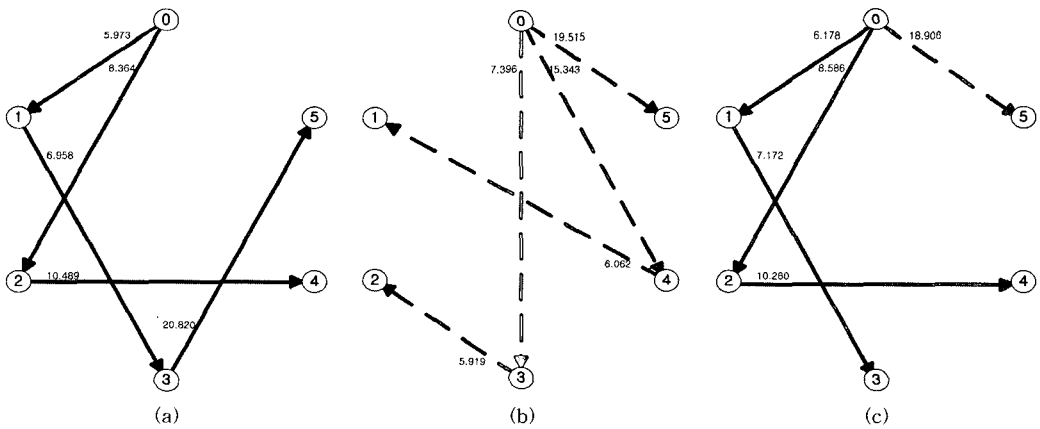

그림 5 그림 4에 대한 TTCC 트리; (a):T_0, (b):T_1, (c): 그림 4의 최종 결과

생성할 것이고 이 트리가 오리지널 트리이고 T0라고 부르자. 그림 5(b)는 TTCC를 이용해 추가 생성된 리던던트 트리 T1라고 하자. 이 두 개의 트리를 이용해 T0과 T1 중 통신 속도가 빠른 에지를 선택하여 최종 통신 트리를 만든다. 노드 0과 노드 5 간의 통신 시간은 노드 3과 노드 5 간의 통신 시간보다 짧기 때문에 시간 α만큼 지난 뒤 노드 5는 노드 3대신 노드 0으로부터 오는 경로를 선택하게 된다.

그림 6은 위의 예제에 대하여 ECEF와 TTCC에 기반한 브로드캐스트 연산에 대한 시간 다이어그램을 나타낸다. 여기서 TTCC를 이용한 집합 통신에 소요한 시간이 ECEF를 사용하는데 소요한 시간보다 작게 나타나 성능이 향상되었음을 보여준다.

3.3 TTCC 시뮬레이터

이 섹션에서는 TTCC 시뮬레이터에 대한 구조의 개념을 설명한다. TTCC의 시뮬레이션은 P.B. Bhat[9]이 사용했던 시뮬레이터에 기반하여 수행한다. 시뮬레이터는 두 개의 프로시져를 가지고 있는데, 하나는 통신트리를 생성하기 위한 것이고, 다른 하나는 생성된 스케쥴의 실행시간을 측정하기 위한 것이다. 각 알고리즘의 성능은 실행시간의 측정값을 이용해 추정한다.

시뮬레이션을 위해 두 개의 통신 비용 행렬(communication cost matrix)을 사용하였다. 실제 네트워크 상태를 나타내는 오리지널 행렬을 M_0라고 하자. M_0를 이

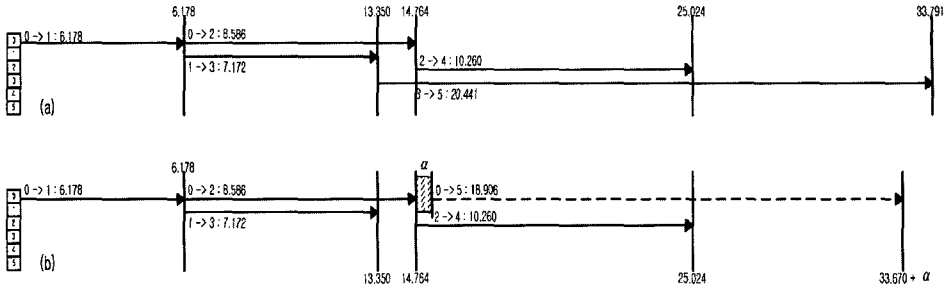

그림 6 타이밍다이어그램; (a) 고유 스케쥴링 트리(ECEF) (b) 최종 스케쥴링 트리(TTCC)

용해 특정 에러율 분포 안에서 행렬 M_1이 만들어진다. 즉, M_0은 집합통신이 실행될 때의 실제 네트워크 정보를 나타내고, M_1은 통신 트리 생성을 위해 예측한 네트워크 정보값을 나타낸다.

첫번째 프로시져로 스케쥴을 만드는 단계에서는 ECEF와 TTCC 모두를 위한 스케쥴을 생성하기 위해 네트워크 예측정보를 표현한 행렬 M_1을 사용하고, 그 다음 프로시져로 주어진 네트워크 상황에서 성능을 구하기 위해서는 각 스케쥴링 알고리즘의 실행시간을 구하는데 M_1이 아닌 실행 시 네트워크 정보를 표현한 M_0을 사용한다.

4. 성능측정(Performance Evaluation)

이 장에서는, 앞에서 설명한 TTCC 시뮬레이터를 이용한 시뮬레이션 결과의 성능 분석에 대하여 기술한다.

네트워크 정보를 모니터링하는데 널리 사용되는 NWS는 유용한 네트워크 정보들을 제공한다. 그러나 NWS는 작은 에러를 자주 발생한다고 전해진다. 에러 분포를 나타내는 막대그래프에 따르면[5], 작은 값의 에러들이 자주 발생하고, 더구나 큰 값의 에러도 다소 빈번히 발생해 네트워크 상황의 변동을 고려해야 한다. TTCC 시뮬레이션에서 이러한 에러 분포를 반영하기

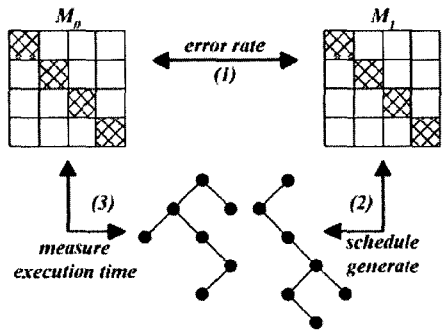

그림 7 TTCC 시뮬레이션 생성단계

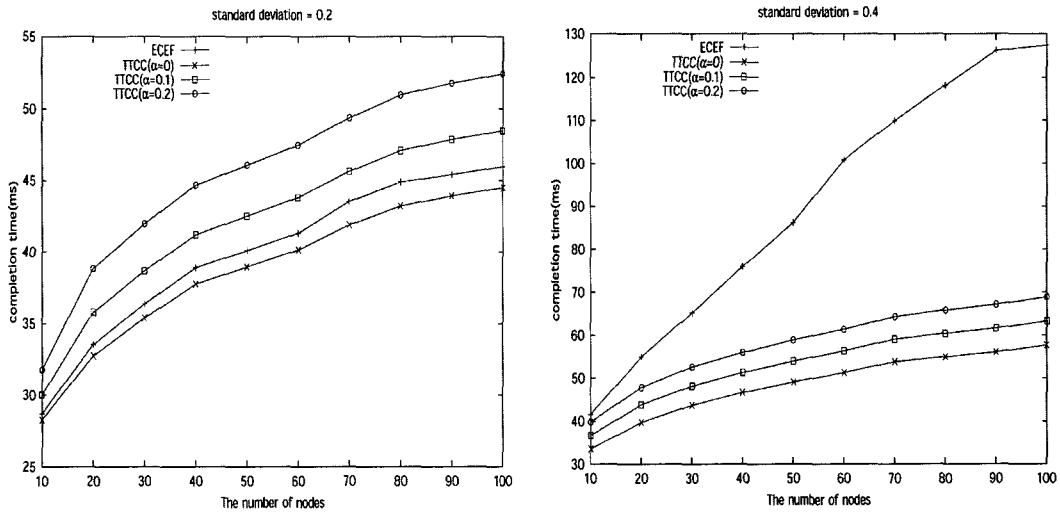

그림 8 에러 분포의 다양한 표준편차에 따른 TTCC 성능 비교

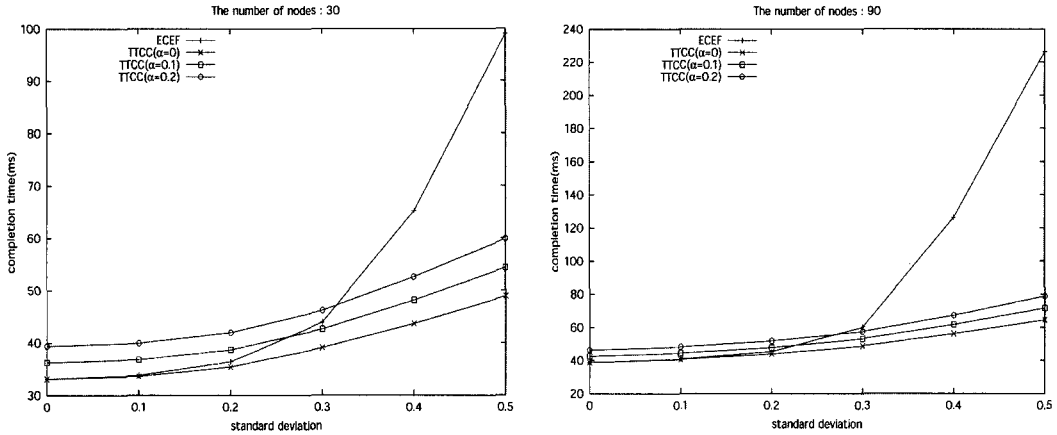

그림 9 노드 수에 따른 TTCC 성능 비교

위해 왜곡된 네트워크 비용 행렬에 대한 임의의 수(random number) 생성 모듈에 정규분포 $N(\mu, \sigma)$가 적용됐다. 에러 범위는 표준편차(σ)로 반영되고 0부터 0.5까지 다양한 표준 편차 값을 시뮬레이션에서 사용했다. [15]의 연구 결과를 볼 때, 에러 정보에 대한 표준편차의 일반적인 상황은 0부터 0.3까지의 범위에 있으므로 네트워크 결점을 일반적인 범위를 넘는 0.3부터 0.5까지의 표준편차 범위에서 발생한다고 가정했다.

이번 실험에서 사용한 시뮬레이터는 ECEF[9]를 수정하여 사용한 것으로, [9]에서 FEF, ECEF, Look-ahead를 비교하는데 사용한 시뮬레이션 파라미터들을 그대로 이용하였다.

- 메시지 크기는 1 MB이다.
- 네트워크 지연은 10 μsec에서 1 msec 범위에 있다.
- 네트워크 대역폭(bandwidth)은 10 KB/s에서 200 MB/s에 범위에 있다.

이종 네트워크를 구성하는 내부 클러스터 네트워크(intra-cluster network)에의 지연과 대역폭은 각각 10 μsec에서 1 msec, 10MB/s에서 200MB/s의 범위에 나타나며, 외부 클러스터 네트워크(inter-cluster network)에서는 지연이 1 msec에서 20 msec, 대역폭이 10KB/s에서 50KB/s의 범위에 있다[9]. 그런데, 지연이 1 msec보다 큰 경우 성능의 변화가 완만하게 나타나므로 성능의 변화가 큰 10 μsec에서 1 msec의 범위에 있는 지연값을 파라미터로 이용하였다.

이러한 파라미터들로 노드 수를 변화시키며 실험을 했고, 실험은 1000 번까지 반복되었다. 그림 8은 에러 분포에서 2 개의 표준 편차 아래 ECEF와 TTCC의 비교결과를 나타낸다. 노드 수가 증가할수록 실행시간은 완만하게 증가하는 반면, 그림 9에서는 표준편차가 증가할수록 실행시간이 급격하게 변화하는 것을 볼 수 있다. 이것은 에러 분포의 표준편차가 증가할수록 성능이 감소하는 것을 보여준다. 이 결과에 따르면 노드 수보다는 에러분포의 변화가 성능감소에 더 큰 영향을 미친다.

네트워크 정보가 정확할 때는 ECEF가 TTCC보다 더 좋은 결과를 보여주지만, 네트워크 정보가 부정확할 경우 ECEF보다 TTCC의 성능이 좋게 나타난다. 특히, 에러 분포의 표준편차가 증가할 경우, TTCC로 스케줄링을 할 때 ECEF보다 훨씬 좋은 성능을 보여준다. 이것을 그림 10에서 다시 확인할 수 있다.

그림 10은 표준편차에 따른 비율로 표현된 지연(delay)의 변화를 보여준다. 지연률(dealy ratio)은 $\frac{P(E_i)-P(E_0)}{P(E_0)}$로 표현할 수 있고, 여기서 $P(E_i)$는 에러 분포의 표준편차가 i일 때 실행시간을 말한다. 그림 11의 그래프는 에러율의 표준편차가 증가할수록 나타나는 성능 감소의 정도를 보여준다. 예를 들면, 표준편차가 0.3일 경우, TTCC의 성능감소 비율은 49%인 반면, ECEF의 성능감소 비율은 55%이다. 표준편차가 0.4일 경우는 감소비율은 각각 TTCC가 76%, ECEF가 226%이다.

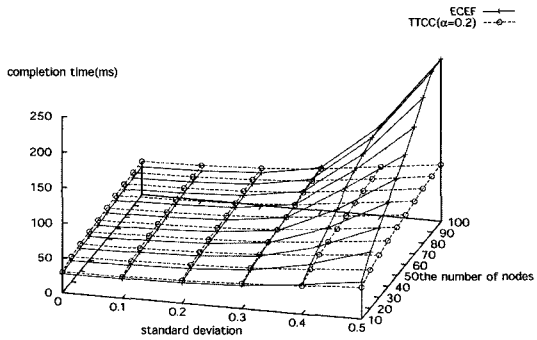

그림 10 TTCC 성능의 실험 결과

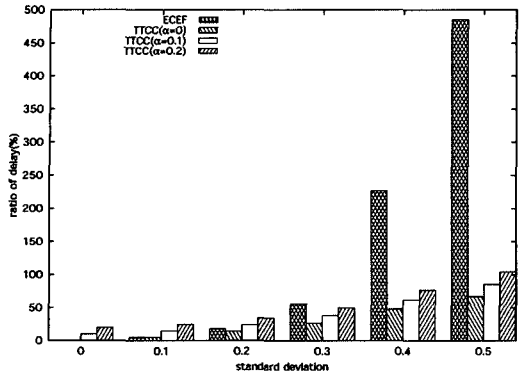

그림 11 노드 수가 100일 때의 지연률

5. 결론 및 향후과제

분산된 이종 시스템 환경에서 집합 통신을 다룬 대부분의 기존 연구들은 사용되는 네트워크 예측 정보가 정확하다는 가정 아래 이루어진 통신 효율성에 대한 연구들이었다. 그러나 네트워크 정보의 예측값이 항상 정확하다는 것을 보장하기는 어려우므로, 통신이 발생할 당시엔 예측정보를 이용해 생성한 통신 트리가 최적의 트리가 아닐 수 있다. 이 연구에서는 TTCC는 좋지 않은 네트워크 상황이거나 네트워크 예측정보가 실제 통신이 일어날 경우의 네트워크 정보와 다를 경우에도 좀더 신뢰성 있는 통신 스케줄링하는 방법을 제안한다. 이를 위해 리던던트 트리를 추가로 생성해 실행시간 동안 좀더 최적의 스케줄링 경로를 찾아내는데 사용한다. 시뮬레이션 결과로 TTCC가 일반적인 상황에서는 작지 않은 오버헤드를 갖지만 네트워크 정보가 부정확한 경우 좀더 나은 스케줄링 트리를 생성시킬 수 있다. 특히, 노드 수가 많을수록, 에러율이 높을수록, ECEF는 급격히 성능이 감소하는 반면, TTCC는 안정적인 성능을 보여준다.

이 논문은 시뮬레이션을 통한 실험의 결과에 기반하며, 스케줄 생성을 위한 시간이나 스케줄 생성주기와 같은 다른 요소들을 모두 고려한 것은 아니다. 이러한 요소들은 정확한 결과를 얻기 위해 중요한 사항이므로 향후 이를 고려한 연구를 진행하여야 할 것이다.

참 고 문 헌

[1] I. Foster, C. Kesselman, and S. Tuecke, "The anatomy of the grid: enabling scalable virtual organizations," The International Journal of Supercomputer Applications, 2001.

[2] I. Foster, and C. Kesselman, "The Globus Project: A status report," 7th Heterogeneous Computing Workshop, pp. 4~18, 1998.

[3] A.S. Grimshaw, and W. A. Wulf, "Legion-A view from 50,000 feet," 5th IEEE International Symposisum on High Performance Distributed Computing, pp. 89~99, 1996.

[4] G. Aloisio, M. Cafaro, C. Kesselman and R. Williams, "Web access to supercomputing," IEEE Computing in Science and Enginerring, pp. 66~72, November/December 2001.

[5] R. Wolski, "Dynamically forecasting network performance using the network weather service," Journal of Cluster Computing, 1998.

[6] M. Mitzenmacher, "How useful is old information," IEEE Transactions on Parallel and Distributed Systems, Vol. 11, No. 1, pp. 6~20, Jan, 2000.

[7] M. Banikazemi, V. Moorthy, and D. K. Panda, "Efficient collective communication on heterogeneous network of workstations," International Conference on Parallel Processing, pp. 460~467, 1998.

[8] M. Bernaschi and G. Iannello, "Collective communication operations: experimental results vs. theory," Concurrency: Practice and Experience, pp. 10(5):359~386, 1998.

[9] P. B. Bhat, C. S. Raghavendra, and V. K. Prasanna, "Efficient collective communication in distributed heterogeneous systems," 19th IEEE International Conference on Distributed Computing Systems, pp. 15~24, 1999.

[10] R. Wolski, N. T. Spring, and J. Hayes, "The network weather service: a distributed resource performance forecasting service for metacomputing," Journal of Future Generation Computing Systems, pp. 15(5~6):757~768, 1999.

[11] A. Fei, G. Pei, R. Liu and L. Zhang, "Measurements on delay and hop-count of the internet," 3rd Global Internet Mini-Conference in conjunction with IEEE Globecom, 1998.

[12] A. Bar-Noy and S. Kipnis, "Designing broadcasting algorithms in the postal model for message passing systems," 4th Annual Acm Symposium on P.

[13] D. Culler, R. Karp, D. Patterson, A. Sahay, K. E. Schauser, E. Santos, R. Subramonian, and T. von Eicken, "LogP: Towards a realistic model of parallel computation," 4th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming, pp. 1~12, 1993.

[14] M. Banikazemi, J. Sampathkumar, S. Prabhu, D. K. Panda, and P. Sadayappan, "Communication modeling of heterogeneous networks of workstations for performance characterization of collective operations," International Workshop on Heterogeneous Computing, pp. 125~131, 1999.

[15] C. Krintz, R. Wolski, "JavaNws: the network weather for the desktop," Proceedings of Java Grande, pp 116~125, 1999.

[16] CSIM18 web page, http://www.mesquite.com/.

차 광 호

1999년 8월 숭실대학교 컴퓨터학부(공학사). 2002년 2월 한국정보통신대학교(ICU) 정보공학부(공학석사). 2002년 4월~현재, 한국과학기술정보연구원 슈퍼컴퓨팅센터 연구원. 관심분야는 병렬 컴퓨팅, 클러스터 컴퓨팅, 병렬파일시스템

이 정 희

1999년 2월 서강대학교 전산학과(공학사) 2004년 2월 한국정보통신대학교(ICU) 정보공학부(공학석사). 2004년 4월~현재, 한국전자통신연구원 임베디드 S/W연구단 연구원. 관심분야는 병렬 컴퓨팅, 그리드 컴퓨팅, 임베디드 OS

한 동 수

1989년 서울대학교 계산통계학과(학사) 1991년 서울대학교 계산통계학과(석사) 1996년 일본 교토대학교 정보공학과(박사). 1996년 4월~1996년 7월 일본 NEC C&C 중앙연구소 연구원. 1996년 9월~1997년 10월 ㈜현대정보기술 정보기술연구소 책임연구원. 1997년 11월~현재 한국정보통신대학교 공학부 부교수

유 찬 수

서울대학교 전기공학(공학사), 서울대학교 전기공학(공학석사). 1994년 the Department of Computer Science and Engineering of The Pennsylvania State University(박사학위). 1994년~1997년 LG전자. 1998년~2001년 ICU(Information and Communications University) 교수. 2002년~현재 Cleveland State University 교수. 관심분야는 Mobile and Embedded Computing, Parallel and Cluster Systems, Computer Architecture and Performance Evaluation

e-mail: c.yu91@csuohio.edu