MPI 환경에서의 양방향 병렬 탐색의 구현

차광호[0] 홍정우 곽재승 변옥환
한국과학기술정보연구원 슈퍼컴퓨팅센터
{khocha[0], jwhong, jskwak, ohbyeon}@kisti.re.kr

The Implementation of Parallel Bidirectional Search on MPI environment

Kwangho Cha, Jeongwoo Hong, Jaiseung Kwak, Okhwan Byeon
Korea Institute of Science and Technology Information

요 약

인공 지능 분야 문제의 특성으로 인하여 병렬 처리 기법의 적용이 자주 고려되고 있다. 특히 순차적인 문제 해결 알고리즘이 병렬 처리 개념과 접목되면서 새로운 특징을 갖는 알고리즘으로 발전될 수 있는데 양방향 병렬 탐색을 그 예로 들 수 있으며 특정 슈퍼 컴퓨터를 대상으로 한 구현 결과도 보고 된 바 있다. 본 논문에서는 양방향 병렬 탐색 알고리즘을 보다 보편적인 메시지 패싱 인터페이스 (MPI)를 이용하여 구현하고 두 종류의 병렬 시스템을 대상으로 테스트 함으로서, MPI 환경에서의 양방향 병렬 탐색의 성능을 비교 분석하였다.

1. 서론

인공 지능 분야의 많은 연구들은 주어진 문제를 풀기 위한 일종의 탐색 기법으로 설명되어 질 수 있는데, 이들 탐색 문제가 갖는 가장 중요한 특징은 '조합의 폭발적 증가(combinatorial explosion)'이다. 이는 탐색 되어야 하는 상태의 수가 문제의 크기와 복잡도가 증가함에 따라 급격히 늘어 남을 의미한다[1].

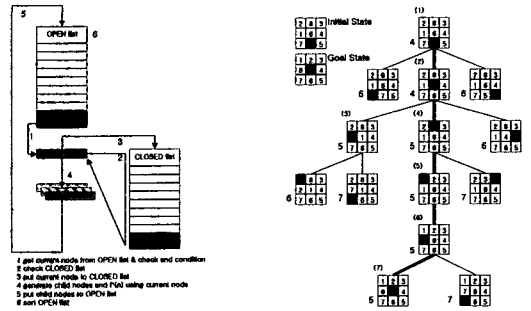
이러한 상황에서 효율적인 탐색을 구현하기 위하여 여러 가지 방법이 제시 되었는데 휴리스틱 기법을 그 예로 들 수 있다. 기존의 획일화 된 방법과는 달리 문제 영역의 이해를 바탕으로 지식 기반의 문제 해결을 시도한다는 차이가 있기는 하지만 역시 많은 자원을 요구하는 문제를 가지고 있는 것이 사실이다. 이러한 제약을 개선하기 위하여 새로운 접근 방법들이 연구되고, 또 다른 관점으로 병렬 처리를 이용하는 방법들이 제안되어 왔다.

병렬 처리를 이용할 때 나타나는 장점 중 하나는 단순히 기존의 순차적인 알고리즘을 병렬화 해서 얻는 효과와 더불어, 기존의 순차적인 환경에서는 적용이 용이하지 않은 알고리즘을 비로서 구현 할 수 있다는 점이다. 그 대표적인 예가 휴리스틱 알고리즘에 바탕을 둔 양방향 병렬 탐색이며, 특정 슈퍼컴퓨터에서의 성능 분석이 진행되어 진 바 있다 [2,3,4].

본 논문에서는 이 양방향 방향 탐색을 보다 보편적인 병렬 프로그래밍 환경인 메시지 패싱 인터페이스 (MPI)를 이용하여 구현하고, CRAY T3E 와 클러스터 시스템에서 성능을 측정함으로서 양방향 병렬 탐색이 MPI 환경에서 갖는 수행 특성을 조사하였다.

2. 관련 연구

양방향 병렬 탐색은 최적치 탐색 알고리즘의 일종인 A* 알고리즘에 기본을 두고 있으며, 본 논문에서 성능을 측정하는데 사용된 모든 휴리스틱 알고리즘들이 풀고자 하는 문제는 경로 탐색 중 하나인 8 퍼즐 문제로서 초기 상태에서 목표 상태까지의 중간 변화 과정의 최적화를 찾는 문제이다. [그림 1]은 A* 알고리즘의 개요와 A*알고리즘이 구한 8 퍼즐 문제의 해를 보여 주고 있다.

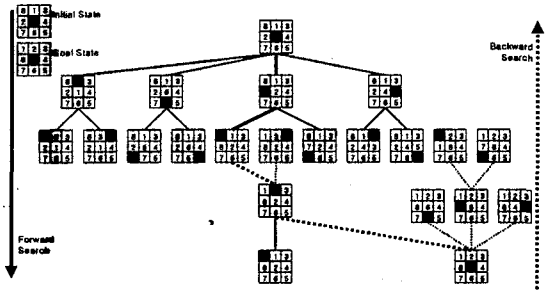


[그림 1] A* 알고리즘의 개요(좌) 및 8퍼즐 문제의 예(우)

순차적인 환경에서의 A* 알고리즘은 초기 상태에서 상태 변환 연산자를 이용하여 새로운 상태를 만든 후, 이를 목표 상태와 비교하면서 휴리스틱 함수를 적용하여 답에 도달할 확률이 높은 상태들을 대상으로 상태 변환 작업을 되풀이하는 알고리즘이다. 이를 그대로 병렬화를 한다면, n 개의 프로세스가 초기 상태로부터 파생된 상태들을 대상으로 연산을 적용하고 목표 상태에 도달했는지를 조사하는 작업을 진행

하게 된다.

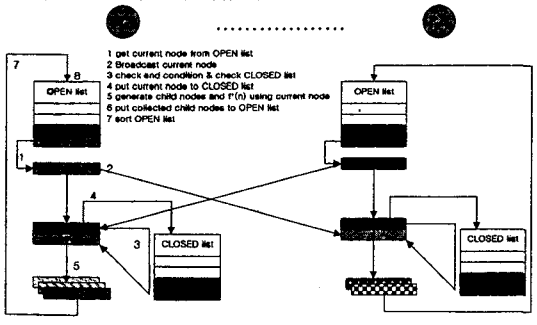
양방향 병렬 탐색은 두 그룹의 프로세서들이 초기 상태와 목표 상태에서 탐색을 시작하는 방법이다. 즉 처음 $n/2$의 프로세스들은 초기 상태에서부터 상태 변환을 시작하고 또 다른 $n/2$ 개의 프로세서들은 목표 상태로부터 상태 변환을 시작해서 탐색을 수행한다. 이때 두 그룹의 프로세서들이 이미 만들어 놓은 상태들 중에 동일한 상태가 존재하면 원하는 답을 구한 것이 된다. [그림 2]는 8퍼즐 문제를 대상으로 한 양방향 탐색의 상태 변이를 보여 준다[2].



[그림 2] 양방향 병렬 탐색의 상태 변이 예

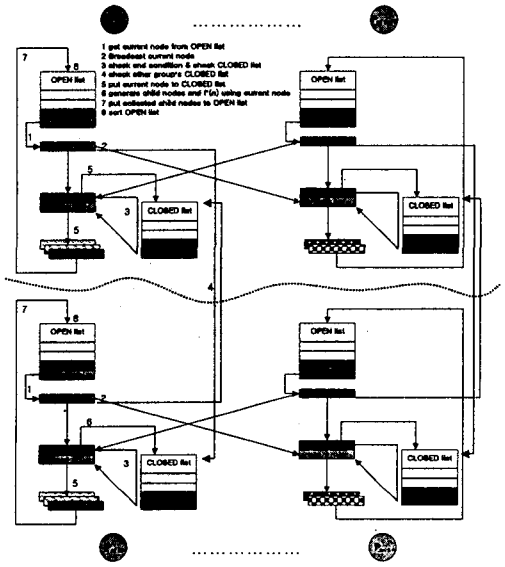
3. 양방향 병렬 탐색의 구현

기존의 A* 알고리즘은 [그림 1]처럼 현재 상태에 연산을 적용하기에 앞서 이미 탐색 된 상태들이 저장되어 있는 클로즈드 리스트를 탐색하는 과정을 거치게 되며 이는 최종 목표 상태에 도달 했는지 여부를 판단하는 중요한 과정이 된다. MPI와 같은 분산 메모리 환경을 고려할 때, 특정 프로세서가 이 클로즈드 리스트를 관리할 경우, 프로세서들 간의 참조 과정으로 인하여 빈번한 메시지 전달 과정이 발생하게 되며, 클로즈드 리스트를 나누어서 관리할 경우에는 확장성에 문제가 있는 것을 발견하였다. 이러한 이유로 단방향 병렬 탐색을 구현하는 과정에서는 [그림 3]과 같이 클로즈드 리스트는 모든 프로세서가 보유하되, 각각의 프로세서가 서로 다른 오픈 리스트를 보유하고 이를 바탕으로 새로운 다음 상태들의 생성을 동시에 진행하도록 하였다.



[그림 3] 단방향 병렬 탐색의 구조

양방향 병렬 탐색을 구현하는 과정에서는, 위에서 설명한 단 방향 병렬 탐색을 기본으로 사용하였다. 전체 프로세서를 두개의 프로세스 그룹으로 나눈 후, 역 방향 탐색을 수행하는 프로세스 그룹은 주어진 문제의 목표 상태를 초기 상태로, 초기 상태를 목표 상태로 설정한 뒤 작업을 수행하도록 하였다. 이와 함께 두 프로세서 그룹이 탐색한 상태들 중에 공통된 상태가 있는지를 확인하기 위하여 타 그룹의 프로세서들이 가지고 있는 오프 리스트에서 현재 상태를 검색하는 기능이 추가 되었다. [그림 4]는 이와 같은 양방향 병렬 탐색의 구조를 보여 주고 있다.



[그림 4] 양방향 병렬 탐색의 구조

4. 성능 측정

성능 측정을 하기 위하여 적용된 문제들은 [그림 5]와 같다. 해를 찾기 위하여 탐색 되어야 하는 트리의 깊이에 따라 문제의 난이도가 결정되는데, 4 종류의 문제를 실험에 이용하였다[2,3].

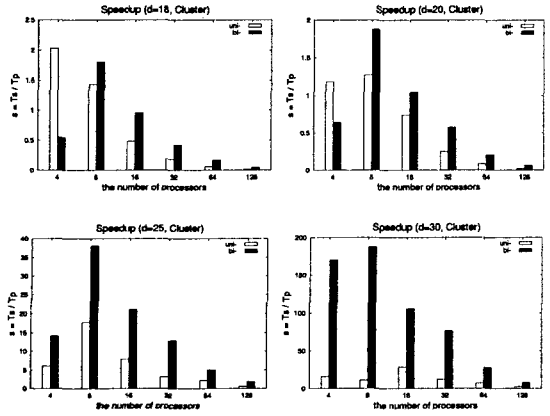
Initial State				Goal State
d = 18	d = 20	d = 25	d = 30	
2 1 6 4 ■ 8 7 5 3	2 1 6 4 5 8 ■ 7 3	1 5 6 2 7 8 4 ■ 3	■ 5 6 1 4 8 7 2 3	1 2 3 8 ■ 4 7 6 5

[그림 5] 성능 측정용 문제들의 초기 상태와 목표 상태 및 난이도

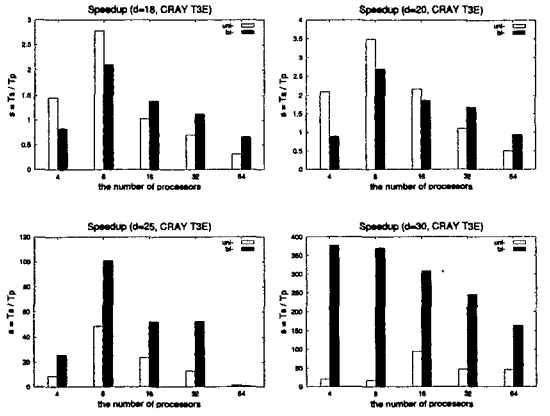
알고리즘이 수행하면서 적용한 휴리스틱 함수 역시 성능을 크게 좌우하는 요소이나, 본 논문에서 확인하고자 하는 바는 휴리스틱 알고리즘간의 성능이므로, 결과 비교에 용이한 수행 시간을 보인 '맨하탄 거리'를 모든 알고리즘의 휴리스틱 함수로 이용하였다.

구현된 양방향 병렬 탐색은 두 종류의 병렬 시스템에서 테스트 되었는데, KISTI 에서 보유하고 있는 128 노드 클러스터 시스템과 CRAY T3E 에서 각각 성능을 측정하였다.

단일 순차 알고리즘과 두 종류의 병렬 알고리즘들의 실행 시간을 바탕으로 성능 향상 비률 구하였다. [그림 6,7]은 각



[그림 6] 128 노드 클러스터 시스템에서의 성능



[그림 7] CRAY T3E 에서의 성능

시스템에서의 성능 향상 비를 보여 주고 있다.

두 시스템에서의 성능 변화의 전반적인 추세는 유사하였지만, 성능 향상의 정도가 CRAY T3E가 더 우수한 것으로 나타났다. 이는 각 프로세스들의 클로즈드 리스트를 업데이트하는 과정에서 프로세서들 간에 집합 통신(Collective Communication)을 수행하는데, CRAY T3E의 프로세서간 통신 성능이 클러스터 시스템 보다 우수하기 때문인 것으로 예상된다.

5. 결론

두 시스템에서 공통적으로, 문제의 난이도가 어려워질수록 양방향 병렬 탐색의 성능이 우수한 것으로 나타났으며, 프로세서 수가 증가함에 따라 양방향 병렬 탐색이 갖는 성능 향상비가 감소됨을 발견할 수 있었다. 따라서 문제의 난이도가 높은 경우에는 단 방향 탐색 보다 양 방향 탐색이 효과적임을 알 수 있었다.

성능 향상 비에서 기종간에 차이를 가지고는 있으나, 클러스터 시스템에서는 대부분의 경우 양방향 병렬 탐색이 우수한 성능을 가지고 있다는 사실을 확인 할 수 있었으며, 특히 프로세서 수가 적은 클러스터 시스템에서 난이도가 높은 문제의 풀이를 시도하고자 하는 경우에는 양방향 병렬 탐색이 유용하다는 결론을 내릴 수 있었다.

본 논문에서는 MPI-1 을 기준으로 실험을 진행하여, MPI-2 가 가지고 있는 원격 메모리 접근과 같은 기능을 이용하지는 않았으나, 원격 메모리를 이용한 구현과, 집합 통신을 이용한 구현간의 성능 비교도 추후 필요할 것으로 여겨진다.

참고문헌

[1] Morris W. Firebaugh, "Artificial Intelligence : A knowledge-based Approach," 1988, PWS-KENT Publishing company.

[2] David Nassimi, Milind Joshi, and Andrew Sohn, "H-PBS: A hash-based scalable technique for parallel bidirectional search," Proc. 7th IEEE Symposium on Parallel and Distributed Processing, 1995, pp 414 ~ 421.

[3] Andrew Sohn, Mitsuhisa Sato, Shuichi Sakai, Yuetsu Kodama, and Yoshinori Yamaguchi, "Parallel Bidirectional Heuristic Search on the EM-4 Multiprocessor," Proc. 6th IEEE Symposium on Parallel and Distributed Processing, 1994. pp 100 ~ 107.

[4] Andrew Sohn, "Parallel Bidirectional A* Search on a Symmetric Multiprocessor," Proc. 5th IEEE Symposium on Parallel and Distributed Processing, 1993, pp 19 ~ 22.

[5] Peter S. Pacheco, "Parallel Programming with MPI," 1997, Morgan Kaufmann Publishers, Inc .

[6] Marc Snir, Steve Otto, Steven Huss-Lederman, David Walker, and Jack Dongarra, "MPI-The Complete Reference," 1998, The MIT Press.