2차원 메쉬에서의 유연성 있는 프로세서 할당기법

서 경 희
성 신 여 자 대 학 교 컴 퓨 터 정 보 학 부
khseo@cs.sungshin.ac.kr

A Flexible Processor Allocation Strategy for 2D Meshes

Kyung-Hee Seo
Dept. of Computer Science, Sungshin Women's University

요 약

상호연결망으로 메쉬 구조를 채택한 대규모 병렬처리 시스템에 대해서 제안된 기존의 프로세서 할당기법들은 직사각형 모양의 서브메쉬 할당 기법으로 제한되어왔다. 그 결과 기존의 기법들은 심각한 시스템의 단편화를 초래하는 문제를 갖고있다. 본 논문에서는 외부 프래그멘테이션과 작업 응답 시간을 동시에 줄이기 위해서, 단편화된 메쉬 시스템에도 적용될 수 있도록 직사각형뿐만 아니라 변형된 L자 모양의 서브메쉬를 할당하는 확장된 LSSA (L-Shaped Submesh Allocation) 기법을 제안한다. LSSA 기법에서 수행되는 모든 서브메쉬 모양의 변형들은 응용 프로그래머에게 투명성을 보장한다. 시뮬레이션 결과를 통해서 LSSA 기법이 작업 응답 시간과 시스템의 활용도 면에서 다른 기법들보다 우수함을 보인다.

1. 서론

MIMD 메쉬 시스템에서 host processor에 있는 task dispatcher는 각 작업이 요구하는 크기를 갖는 서브메쉬를 다른 작업이 수행되는 서브메쉬들과 중첩되지 않도록 할당하며, 그 작업의 수행이 완전히 끝날 때까지 할당된 서브메쉬를 독점적으로 사용하도록 한다. 작업의 수행이 완료되면, deallocation 과정이 진행되어 새로운 작업에게 할당될 기회를 기다리게 된다. 이와 같은 프로세서 할당 · 해제에서 메쉬 시스템으로 들어오는 작업들이 요구하는 서브메쉬들의 크기는 1개에서부터 시스템 전체의 프로세서들의 개수를 요구할 수도 있으므로, 이 시스템을 효율적으로 공유하는 것은 사용자뿐만 아니라 시스템 보유자 입장에서도 매우 중요하다.

그러므로, 계속해서 들어오는 작업들에게 이용가능한 프리 서브메쉬들을 효율적으로 할당하는 프로세서 할당 기법의 개발이 요구된다. 프로세서 할당 기법의 주된 목적은, 작업들이 시스템에 도착되는 순서대로 서비스를 하는 공정성을 유지하면서 시스템의 성능을 높이는 것이다. 이는 작업 응답 시간과 시스템 프래그멘테이션을 줄이면서, 동시에 시스템의 활용도를 증가 시킴으로써 가능해진다. 높은 시스템의 활용도를 얻기 위해서는 발생되는 프래그멘테이션의 양을 최소화 시켜야 한다.

프로세서 할당 기법을 수행하면서 발생할 수 있는 프래그멘테이션의 종류는 내부, 외부, 그리고 가상 프래그멘테이션으로서, 내부 및 가상 프래그멘테이션은 더 이상 문제가 되지 않는다. 여전히 문제가 되는 것은 외부 프래그멘테이션으로서, task dispatcher는 시스템에 도착되는 다음 작업의 크기를 알 수 없기 때문에, 이 외부 프래그멘테이션을 완전히 제거할 수 있는 프로세서 할당 기법은 존재하지 않는다.

Flexfold 할당 기법을 포함해서 이전에 발표되었던 모든 프로세서 할당 알고리즘들은 전체 시스템 구조와 같은 위상의 서브메쉬를 발견하지 못하면, 할당할 수 없었다[1-7]. 바꾸어 표현하면, 요청된 크기를 충분히 수용할 수 있는 개수의 프리 프로세서들이 시스템 내에 존재하더라도, 그 프리 프로세서들이 요청된 크기를 가지는 하나의 직사각형 서브메쉬를 형성하지 못하면, 그 작업은 수행되지 못하고 waiting queue로 보내진다. 그 결과, 외부 프래그멘테이션 문제는 여전히 심각한 상태이며, 이로 인해 34%에서 46%의 시스템의 활용도를 보이고 있다[8]. 이러한 문제를 해결하기 위해서, FCFS방식으로 작업을 처리하면서 효율적인 프로세서 할당 기법을 연구하는 대신에, 다른 작업 스케줄링을 채택하는 것이 더 바람직하다는 연구결과가 하이퍼큐브 시스템에 대해

발표되었다[9-11]. 또한, 메쉬 시스템에 적용했을 경우에도 좋은 성능을 보인다는 연구 결과가 있다[12]. 그러나, FCFS 방식이 아닌 다른 작업 스케줄링을 채택하는 것은 사용자에게 공정하지 않다는 단점을 가지며, 또한 크기가 큰 작업을 특히 더 차별하게 되는 경향이 있다. 즉, starvation problem을 유발시킬 수 있다

본 논문에서는 사용자에게 공정성을 유지하면서, 외부 프래그멘테이션의 발생 양을 줄임으로써 시스템의 활용도를 높이고, 동시에 작업 응답 시간을 줄일 수 있는 LSSA (L-Shaped Submesh Allocation) 기법[13]을 확장하여 제안한다. LSSA 기법은 기존의 프로세서 할당 기법이 직사각형 모양의 서브메쉬 할당 알고리즘으로 제한되었던 것을 확장시킨다.

2. LSM의 소개

초기의 2차원 Buddy 기법이 고려하는 서브메쉬의 모양은 각 변이 2^n인 정사각형이었으며[1], 그 이후에 제안된 모든 서브메쉬 할당 기법들은 직사각형 모양의 서브메쉬 만을 고려하였다[2-7]. 이제 직사각형의 서브메쉬를 사분면으로 나누어 보았을 때, 각 사분면이 존재하지 않는 유연성 있는 메쉬, 즉, L자 모양의 서브메쉬, LSM (L-shaped submesh)을 고려하고자 한다. LSM, *LS(c,d,e,f)*은 긴 너비가 $c+e$ 이고, 높이 d를 가지는 영문자 L자 모양의 블록이며, 두 개의 서브메쉬 $S_L(c,d)$ 와 $S_R(e,f)$가 서로 연결된 프로세서들의 집합으로 구성된다 (그림 1).

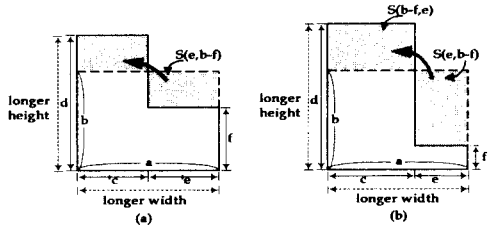

그림 1 서브메쉬 *S(a,b)*로부터 *LS(c,d,e,f)*의 구성 (a) a가 짝수인 경우 (b) a가 홀수인 경우

LSM은 제 1 사분면이 존재하지 않는 서브메쉬로 간주할 수 있다. 따라서, 직사각형의 서브메쉬에서 어느 사분면이 존재하지 않는가에 따라서 네 가지 유형의 LSM들이 존재한다. 그림 2에서와 같이, 제 1 사분면이 없는 서브메쉬는 Type 1 LSM, 제 2 사분면이 없는 Type 2 LSM, 제 3 사분면이 없는 Type 3 LSM, 그리고, 제 4 사분면이 없는 Type 4 LSM이라 한다. Type 2, Type 3, 그리고 Type 4 LSM들은 Type 1 LSM을 시계 반대 방향으로 90°, 180°, 그리고 270° 회전시킨 형태이다.

또한, 직사각형의 서브메쉬에서 두 사분면들이 동시에 존재하지 않는 모양의 서브메쉬도 고려할 수 있다. 이 경우에 네 가지 유형의 형태 변형된 LSM들이 존재한다. 그림 3에서와 같이, 제 1 과 4 사분면이 없는 Type 14 LSM, 제 1 과 2 사분면이 없는 Type 12 LSM, 제 2 와 3 사분면이 없는 Type 23 LSM, 그리고, 제 3 과 4 사분면이 없는 Type 34 LSM이다. 이 형태 변형된 Type 14

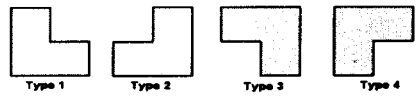

그림 2 직사각형의 서브메쉬들로 구성될 수 있는 LSM들

LSM은 두 개의 서로 이웃하는 Type 1 과 Type 4 LSM이 수직방향으로 붙은 형태이며, 그림에서 Type i LSM을 LSM_i로 표기하였다. Type 12, Type 23, 그리고 Type 34 LSM들은 Type 14 LSM을 시계 반대 방향으로 90°,180°, 그리고 270° 회전시킨 형태이다. 또한, Type 12 LSM은 두 개의 서로 이웃하는 Type 1과 Type 2가 수평방향으로 붙은 형태이며, Type 23 LSM은 두 개의 서로 이웃하는 Type 2 와 Type 3가 수직방향으로, 그리고 Type 34 LSM은 두개의 서로 이웃하는 Type 3 과 Type 4가 수평방향으로 붙은 형태로 간주된다.

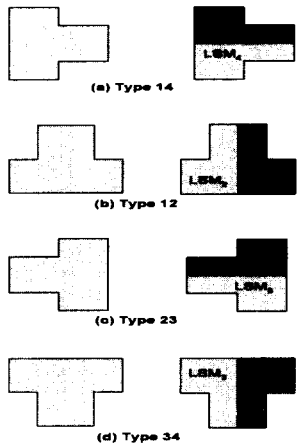

그림 3 두 이웃하는 LSM들이 구성하는 형태변형된 LSM들

3. LSSA 프로세서 할당 기법

LSSA 프로세서 할당 기법은 사용자가 요구하는 직사각형 모양의 서브메쉬를 기존의 기법으로 할당할 수 없을 때, 요청된 서브메쉬를 LSM으로 변환하여 프로세서 할당을 수행한다. 시스템으로 들어오는 작업이 서브메쉬 $a \times b$를 요구할 경우, LSSA 기법은 $a \times b$, $b \times a$, $a/2 \times 2b$, $2a \times b/2$, $2b \times a/2$, 그리고 $b/2 \times 2a$ 들을 탐색할 뿐만 아니라, LSM들도 탐색한다. LSM은 커팅 사이드 a가 짝수일 경우, 서로 연결된 두 개의 서브메쉬 $a/2 \times b+k$ 와 $a/2 \times b-k$ $(1 \le k \le b)$로 구성된다. 커팅 사이드 a가 홀수일 경우, LSM은 다음과 같은 서로 연결된 두 개의 서브메쉬 $\lceil a/2 \rceil + k \times b + \lfloor a/2 \rfloor - k$ 와 $\lfloor a/2 \rfloor - k \times b - \lceil a/2 \rceil - k$ $(0 \le k \le b-1-\lceil a/2 \rceil)$로 구성되며, 할당 가능한 LSM이 발견될 때까지 하나하나 탐색되어진다.

서브메쉬 형태의 변형들이 태스크의 실행시간에 미치는 영향을 분석하기 위해서, 그러한 형태의 변형들을 하나의 서브메쉬에서 다른 서브메쉬로의 그래프 임베딩으로 간주하여[14], 임베딩 코스트를 산출하여, LSSA 기법의 conservative check에서 반영하였다.

LSSA 기법은 변형된 서브메쉬를 할당하는 것이, 예상되는 네트워크 전파 시간의 증가로 인해서, 대기 큐에서 기다리는 것보다도 작업의 응답시간을 증가시키게 되는 것으로 판단되면, 그 작업에게 변형된 모양의 서브메쉬를 할당하지 않는다.

4. 시뮬레이션을 통한 성능 분석

LSSA 기법을 기존의 할당 기법들과 비교하기 위해서 이산 사건 중심의 시뮬레이션을 수행하였다. 할당을 기다리는 태스크들은 대기 큐에서 기다리며, 큐의 헤드에 있는 태스크가 제일 먼저 할당을 받는다. 본 시뮬레이션에서는 시스템에 도착되는 순서대로 큐에서 대기하며, 서비스를 받는 FCFS 방식의 스케쥴링을 사용하였다. 시스템 부하는 작업들이 도착되는 간격과 그 작업들의 체재시간의 비율에 의해 결정하였다[13]. 사용자 입장에서는 평균 작업응답시간(MRT:mean response time)이 가장 중요한 성능평가의 척도이므로, 시스템 부하에 따른 MRT의 변화를 할당 기법에 따라 실험하였다.

메쉬 시스템의 크기는 32×32로 하였고, 시스템 부하는 작업이 도착되는 간격을 변화시킴으로써 조절하였다. 각 작업의 요청된 체재시간은 50단위시간을 평균으로 하는 지수 분포를 갖도록 하였으며, 요구되는 작업들의 크기는 균일 분포로 발생시켰는데, 각 변의 길이는 2에서 20으로 제한하였다. LSSA 기법을 적용하였을 경우의 MRT가 FF, AS, 그리고 Flexfold 기법을 적용하였을 경우보다 작음을 그림 4에서 볼 수 있다. 시스템 부하가 증가할수록, LSSA 기법의 장점은 두드러진다.

LSSA기법을 적용하였을 때의 시스템의 활용도는 그림 5에서 보듯이, 시스템 부하가 0.8 일 때, 62%까지 증가한다. MRT와 시스템의 활용도의 개선도는 특히 시스템 부하가 높을 때 두드러진다.

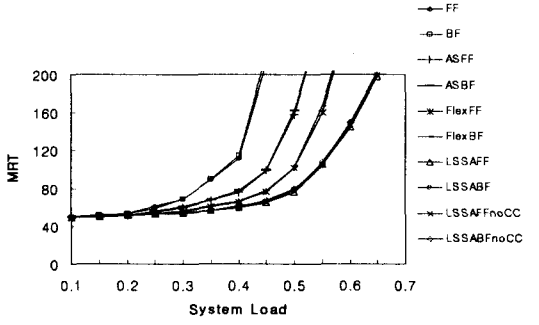

그림 4 시스템의 부하의 변화에 따른 MRT

5. 결론

2차원 메쉬 구조로 구성된 시스템에서 수행될 작업이 요구하는 서브메쉬의 모양을 L자 모양으로 변형하여 할당할 수 있는 유연성 있는 프로세서 할당기법을 제안하였다. 메쉬 시스템에 대한 기존의 프로세서 할당 기법들은 직사각형 모양의 서브메쉬 할당으로 제한함으로써, 심각한 외부 프래그멘테이션 문제를 갖고 있었으며,

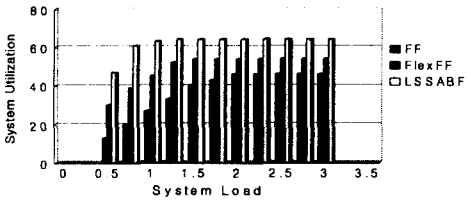

그림 5 시스템의 활용도

이로 인해 시스템의 활용도가 매우 낮았다. 그러나, 제안된 LSSA 기법은 기존의 다른 기법들이 즉시 할당할 수 없는 작업의 요청을 수용할 수 있다. LSSA 기법의 평균 작업 응답 시간이 다른 기법들에 비해서 감소되는 것과 시스템의 활용도가 증가됨을 시뮬레이션 결과를 통해서 보였다.

참고문헌

[1] K. Li and K. H. Cheng, "A Two-Dimensional Buddy System for Dynamic Resource Allocation in a Partitionable Mesh Connected System," *Proc. ACM Computer Science Conf.*, pp. 22-28, 1990.

[2] P. J. Chuang and N. F. Tzeng, "An Efficient Submesh Allocation Strategy for Mesh Computer systems," *Proc. Int'l Conf. on Distributed Computing Systems*, pp. 256-263, 1991.

[3] J. Ding and L. N. Bhuyan, "An Adaptive Submesh Allocation Strategy for Two-Dimensional Mesh Connected systems," *Proc. Int'l Conf. on Parallel Processing*, pp. II-193 - II-200, 1993.

[4] T. Liu *et al.*, "A Submesh Allocation Scheme for Mesh-Connected Multiprocessor Systems," *Proc. Int'l Conf. on Parallel Processing*, pp. II-159 - II-163, 1995.

[5] Y. Zhu, "Efficient Processor Allocation Strategies for Mesh-Connected Parallel Computers," *Journal of Parallel and Distributed Computing*, Vol. 16, No. 12, pp. 328-337, Dec. 1992.

[6] V. Gupta and A. Jayendran, "A Flexible Processor Allocation Strategy For Mesh Connected Parallel Systems," *Proc. Int'l Conf. on Parallel Processing*, pp. III-166 - III-173, 1996.

[7] D. D. Sharma and D. K. Pradhan, "A Fast and Efficient Strategy for Submesh Allocation in Mesh-Connected Parallel Computers," *Proc. IEEE Symposium on Parallel and Distributed Processing*, pp. 682-689, Dec. 1993.

[8] W. Liu, V. Lo, K. Windisch and B. Nitzberg, "*Non-contiguous Processor Allocation Algorithms for Distributed Memory Multicomputers,*" Proc. 1994 Int'l Conf. on Supercomputing, pp. 227-236, June 1994.

[9] P. Krueger, T-H Lai and V. A. Radiya, "Processor Allocation vs. Job Scheduling on Hypercube Computers," *Proc. 11th Int'l Conf. on Distributed Computing Systems*, pp. 394-401, May 1991.

[10] P. Krueger, T-H Lai and V. A. Dixit-Radiya, "Job Scheduling Is More Important than Processor Allocation for Hypercube Computers," *IEEE Trans. on Parallel and Distributed Systems*, Vol. 5, No. 5, pp. 488-497, May 1994.

[11] P. Mohapatra, C. Yu, C. R. Das and J. Kim, "A Lazy Scheduling Scheme for Improving Hypercube Performance," *Proc. Int'l Conf. on Parallel Processing*, pp.I-110-I-117, 1993.

[12] D. D. Sharma and D. K. Pradhan, "Job Scheduling in Mesh Multicomputers," *Proc. Int'l Conf. on Parallel Processing*, pp. II-251 - II-258, 1994.

[13] K. H. Seo and S. C. Kim, "An Adaptive Processor Allocation Strategy for Mesh-Connected Systems," *Proc. IEEE Symposium on Parallel Architectures, Algorithms, And Networks*," pp. 296-303, Dec. 1997.

[14] J. W. Hong, K. Mehlhorn, and A. L. Rosenberg, "Cost Trade-offs in Graph Embeddings, with Applications," *Journal of the ACM*, Vol. 30, No. 4, pp. 709-728, October 1983.