스타그래프 다중처리시스템을 위한 프로세서 할당방법

이원주, 권소라*, 전창호
두원공과대학 인터넷프로그래밍과, 한양대학교 전자컴퓨터공학부
wonjoo@doowon.ac.kr, {srkwon*, chjeon}@para1.hanyang.ac.kr

A Processor Allocation Strategy for Star Graph Multiprocessor Systems

Won-Joo Lee, So-Ra Kwon*, Chang-Ho Jeon
Department of Internet Programming, Doowon Technical College,
School of Electrical and Computer Engineering, Hanyang University

요 약

본 논문에서는 스타그래프 다중처리시스템을 위한 새로운 프로세서 할당방법을 제안한다. 기존의 할당방법은 프로세서 단편화로 인해 작업을 처리할 서브스타를 형성하지 못하면 프로세서 할당이 지연되는 문제점이 있었다. 이러한 할당 지연은 작업의 대기시간을 증가시키고 시스템의 성능 향상을 제한한다. 본 논문에서 제안하는 할당방법은 프로세서 할당 지연이 발생하면 동적할당테이블을 사용하여 단편화된 프로세서의 주소를 재생성한다. 새로운 주소의 프로세서들로 가용 서브스타를 형성하여 할당함으로써 작업의 대기시간을 줄이고 프로세서 단편화를 최소화한다.

1. 서론

다중처리시스템은 여러 개의 프로세서들을 사용하여 많은 작업을 동시에 처리하는 시스템이다. 다중처리시스템의 각 프로세서들은 링(ring), 트리(tree), 메쉬(mesh), 하이퍼큐브(hypercube), 스타그래프(star graph) 등과 같은 상호연결망으로 연결된다. 이러한 상호연결망중에 스타그래프는 차수(degree)와, 직경(diameter) 등에서 하이퍼큐브 보다 우수하다[1,2]. 스타그래프 다중처리시스템에서는 프로세서들간의 통신비용을 최소화하기 위해 전체 시스템과 같은 구조를 가지는 서브스타를 찾아 작업을 할당한다. 이러한 제약 조건은 프로세서 단편을 발생시키는 원인이 된다. 프로세서 단편화로 인해 서브스타를 형성하지 못하면 작업 할당이 지연되기 때문에 시스템 성능이 저하된다. 스타그래프 다중처리시스템을 위한 기존의 할당방법[3,4]들은 서브스타를 완전하게 인식하고, 기억장소와 탐색 복잡도를 줄이는 방향으로 연구를 진행하여 왔다. 그러나 프로세서 단편화로 인한 할당지연을 최소화함으로써 시스템의 성능 향상시키는 방법에 대한 연구 결과는 아직 발표되지 않았다.

본 논문에서는 스타그래프 다중처리시스템을 위한 새로운 프로세서 할당방법을 제안한다. 이 할당방법에서는 기존의 할당방법과 달리 프로세서의 할당지연을 최소화함으로써 시스템의 성능을 향상시킨다. 할당지연을 최소화하기 위해서는 단편화된 프로세서들의 주소를 재생성하고 새로운 서브스타를 형성하여 할당한다. 이러한 방법은 작업의 대기시간을 줄이기 때문에 시스템의 성능을 향상시킬 수 있다.

본 논문의 구성은 다음과 같다. 제 2장에서는 스타그래프에 대한 용어 정의와 기존의 할당방법에 대하여 설명한다. 제 3장에서는 본 논문에서 제안하는 프로세서 할당 및 할당 해제 방법에 대하여 설명하고, 4장에서 결론을 맺는다.

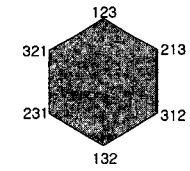
2. 관련 연구

2.1 용어 정의

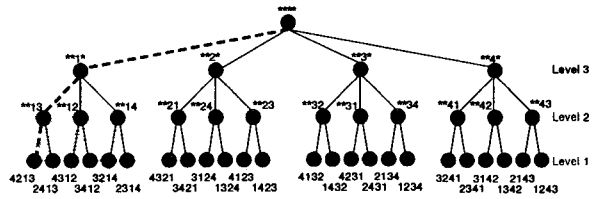
n-차원의 스타그래프 S_n은 n! 노드와 (n-1)n!/2 의 링크로 구성되는 비방향 그래프이다[5,6]. 여기서 노드는 프로세서를 의미한다. S_n의 각 노드 는 n개의 서로 다른 심볼들의 순열(permutation)로 표현된 주소를 가진다. 여기서 심볼은 1≤심볼≤n 조건을 만족하는 정수이다. <그림 1a>에서 노드 123의 첫 번째 심볼 1과 두 번째 심볼 2의 위치를 교환하여 노드 213을 생성한다. 이 두 노드 123과 213은 하나의 링크로 연결되며, 나머지 다른 노드들도 이러한 규칙에 따라 연결된다.

정의 1. S_n에 대한 스타코드(SC: star code)는 n-1 개의 정수를 사용하여 $SC(d_n, d_{n-1},.....d_2,d_1)$로 나타낸다. 이때 각 코드를 구성하는 심볼(symbol)들은 중복하여 사용할 수 없으며 $2 \le d_{n-1} \le n$ 조건을 만족한다.

정의 2. 스타코드 트리는 스타코드를 기반으로 생성하며 T (SC_n) 또는 *SC*-tree로 나타낸다. *SC*-tree는 n-1 레벨을 가지고 있으며 루트 노드의 레벨은 n 이고, 잎(leaf) 노드의 레벨은 1 이다. *SC*-tree의 서브트리를 서브스타라 하며 레벨 k의 서브스타는 S_k로 표기한다. *k* 레벨에 있는 노드의 자식 노드 수는 *k*개이다.



(a) S_3 스타그래프



(b) SC(3, 4, 2) 에 대한 SC-tree

<그림 1> 스타그래프

<그림 1b>는 SC(3,4,2)에서 생성된 SC-tree 이다. SC-tree의 루트 주소는 {****} 이다. level 3의 노드 주소는 심볼셋(symbol set)={1234}에서 하나의 심볼을 선택하여 3번째 심볼로 지정하고 나머지는 "*"로 지정한다. level 2의 노드 주소는 심볼셋={1234}에서 2개의 심볼을 선택하여 순열로 3번째와 4번째 심볼을 지정하고 나머지는 "*"로 지정한다. level 1의 잎(leaf) 노드 주소는 심볼셋={1234}에서 level 2의 주소에서 사용한 심볼을 제외한 나머지 2개의 심볼을 선택하여 순열로 지정한다. 예를들어 잎(leaf) 노드 주소 {4213, 2413}은 level 2의 주소 {**13}에서 사용한 1, 3을 제외한 2와 4를 심볼셋={1234}에서 선택하여 순열로 지정한 것이다.

2.2 기존의 할당방법

스타그래프 다중처리시스템에서는 기존의 할당방법들을 비트-매핑(bit-mapping) 할당방법과 기하학적(geometric) 할당방법으로 분류한다.

비트-매핑 할당방법[3,7]으로 분류하는 MSC(Multi-Star Code) 할당방법[3]은 SC를 이용하여 SC-tree를 생성하고, SC-tree의 잎(leaf) 노드로 서브스타를 구성한다. 이 할당방법은 2^{n-2}개의 SC를 생성하고, 각 SC 마다 n!개의 할당 비트를 사용하기 때문에 기억장소 복잡도는 $2^{n-2} * n!$ 이다. 차원 n 값이 증가하면 할수록 기억장소와 탐색시간 복잡도가 증가하는 문제점이 있다.

기하학적 할당방법으로 분류한 Sg-lattice 할당방법[4]은 MSC 할당방법의 기억장소와 탐색시간 복잡도가 증가하는 문제점을 해결하기 위해 각 노드별로 노드 주소와 할당 프로세서의 수를 저장하는 테이블을 사용한다. 이 할당방법은 가용 서브스타를 찾기 위해 테이블의 데이터를 이용함으로써 서브스타의 탐색시간을 줄인다. 이 할당방법의 기억장소와 탐색 복잡도는 기존의 MSC 할당방법보다 우수하지만 테이블을 관리하는 오버헤드가 발생하는 단점이 있다.

3. 제안한 프로세서 할당방법

본 논문에서는 스타그래프 다중처리시스템을 위한 새로운 DAT (dynamic allocation table) 프로세서 할당방법을 제안한다. 이 할당방법에서는 작업의 할당지연을 최소화하기 위해 단편화된 프로세서들의 주소를 재생성하고 새로운 서브스타를 형성하여 할당한다. 이때 각 노드의 주소와 상태정보를 저장하는 동적할당테이블(DAT)을 사용하여 프로세서들의 주소를 재생성 여부를 결정한다. 동적할당테이블은 다음과 같이 정의한다.

3.1 동적할당테이블

정의 3. 동적할당테이블은 SC-tree의 각 노드 주소와 노드들의 상태정보를 저장하는 데이터 구조이다. 동적할당테이블의 데이터 구조는 다음과 같다.

```
DAT structure
{
    listpointer ptr_Pre, ptr_Next; // 이중 링크 리스트
    int ASC ;                      //처리중인 작업 수
    bool RSI ;                     //주소 재생성여부 결정 비트
}
```

DAT는 각 노드의 주소와 노드의 상태정보를 기록하는 ASI(Allocated Status Information)로 구성한다. DAT 구조에서 ptr_Pre와ptr_Next는 상위 레벨과 하위 레벨의 노드 주소를 저장한다. ASI는 ASC (Allocated Status Counter)와 RSI(Readdressing Status Information)로 구성한다. ASC는 노드가 처리중인 작업의 수를 저장하고, RSI(Readdressing Status Information)는 주소 재생성 여부를 결정하는 비트를 저장한다.

DAT를 생성하기 위해서는 먼저 스타코드를 생성하고, 그 스타코드를 이용하여 SC-tree 생성한다. SC-tree의 각 노드들은 상위레벨과 하위 레벨의 노드 주소와 자신의 ASI를 저장할 수 있도록 이중리스트로 구현한다. SC-tree를 레벨 단위로 분류하여 DAT에 저장한다. 그리고 저장된 각 노드의 주소와 ASI 데이터를 초기화한다.

3.2 프로세서 할당 알고리즘

프로세서 할당 알고리즘은 <그림 2>와 같다. <그림 2>의 [과정 1]에서 S_k 작업이 프로세서를 요청한다. [과정 2]에서는 전체시스템의 노드 수 n!과 루트레벨의 ASC를 사용하여 요청한 작업을 처리할 만한 노드 수 m을 구한다. [과정 3]에서는 m의 값이 작업을 요청한 크기보다 작다면 작업은 큐로 이동하여 작업 처리가 가능한 노드가 생성될 때까지 대기한다. 그러나 m의 값이 작업을 요청한 크기보다 크다면 첫 번째 노드부터 ASC를 순차적으로 검색한다. DAT[n-k].ASC의 값이 0이면 서브스타에게 작업을 최초(first-fit) 할당한다. 그리고 현재 처리하고 있는 작업의 수만큼 서브스타 노드의 ASC 값을 증가하고, RSL=False로 지정한다. 이때 DAT[n-k].ASC의 값이 0가 아니면 READDRESSING() 모듈을 호출한다. 이것은 시스템에 할당 가능한 노드들이 존재 하지만 프로세서 단편화로 인해 서브스타를 형성할 수 없음을 의미한다.

```
[과정 1] Sk 요청 ;
[과정 2] m = n! - 루트레벨 ASC ;
[과정 3]
 If ( Sk <= m )
 {
    • 첫 번째 노드부터 순차적으로 ASC 검색
    If(DAT[n-k].ASC == 0)
    {
        • FF substar allocate;
        • DAT[n-k].ASC:=DAT[n-k].ASC+k!;
        • DAT[n-k].RSL := False;
        • RELATEDSUB( ); //연관된 노드 검색후 ASC증가
    }else{
        Call READDRESSING();
    }
 }else{
        waiting(); //대기
 }
```

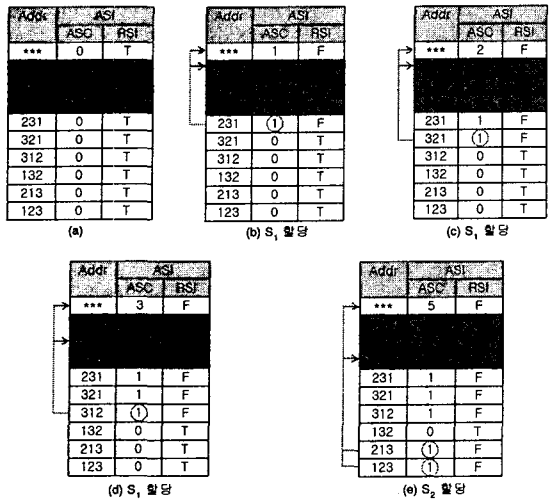
<그림 2> DAT 할당 알고리즘

READDRESSING() 모듈은 단편화로 사용하지 못하는 프로세서의 주소를 재생성하여 할당이 가능하도록 한다. READDRESSING() 모듈은 <그림 3>과 같다.

```
READDRESSING( )
{   //가용 노드의 주소 저장 배열 선언
 POINT  ANG[ ]; //ANG(Avaliable Node Group)
 Do{  //가용 노드그룹 생성하여 ANG[i] 에 저장
      If( DAT[n][i].RSL==TRUE ) I++;
 } While( DAT[n][1] < DAT[n][n] )
 For( i =1 ; n ; 1 ){ //ANG[]의 노드주소로 연관성 검사
   k = i +1 ;
   For( j =1 ; n ; 1 )
   {  // 주소 검색
      if(ANG.ADDRESS[i][j] ◉ ANG.ADDRESS[k][j]==1)
          ADDR.DIFFERENT.ITEM++;
   }
 }
 If(ADDRESS. DIFFERENT. ITEM==2) {
    create  NEWADDRESS;   //주소 재생성
 } else {
    • 대기 큐로 현재의 작성 이동;
 }
}
```

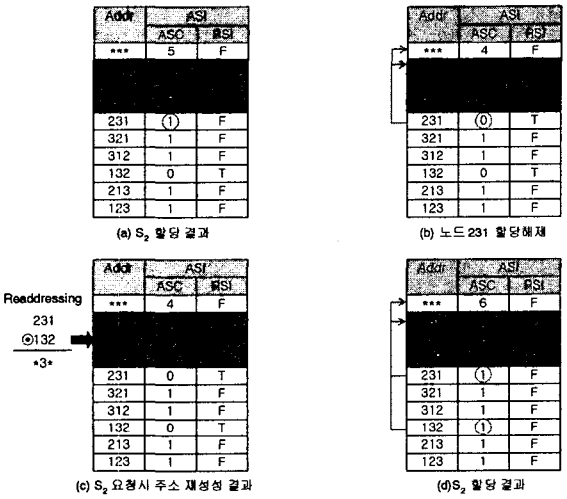
<그림 3> READDRESSING() 모듈

<그림 3>에서는 가용 노드의 주소를 저장할 수 있는 배열 ANG[]를 정의한다. 이 ANG[]에 저장된 값을 통해 노드 주소들의 연관성을 찾는다. 프로세서 주소들이 연관성을 가지면 주소를 재생성하지만 연관성을 갖지 않으면 주소를 재생성 할 수 없기 때문에 요청한 작업은 대기 큐로 이동한다. 노드 주소의 연관성을 찾는 ◉ 연산자는 XNOR 연산을 수행한다. READDRESSING( ) 모듈에서 모든 단편 발생을 해결하는 것은 아니지만 현재 노드들간의 규칙성 여부에 따라 주소를 재생성함으로써 프로세서 단편화를 최소화한다.



<그림 4> 주소 재생성이 필요없는 프로세서 할당 예제

프로세서 할당 알고리즘의 실행 과정을 예로들어 설명한다. <그림 4>는 주소 재생성이 필요없는 프로세서 할당 예제이다, S_3 에 대해 S_1, S_1, S_1, S_2 크기의 작업이 입력되는 경우, <그림 4a>에서 1개의 프로세서 할당 요청를 받는다면 가장 하위 레벨의 노드인 231노드가 할당되고 이와 연결된 노드인 ***와 **1의 상태정보를 변경하게 된다. S_1, S_1, S_2 작업들을 처리할 수 있는 서브스타가 존재하기 때문에 주소를 재생성 할 필요가 없으며 해당 작업들의 할당은 계속 진행된다.



<그림 5> 주소 재성성이 필요한 프로세서 할당 예제

<그림 5>는 주소 재생성이 필요한 프로세서 할당 예제이다, <그림 5a>의 할당 결과에서 노드 231을 해제하면 <그림 5b>와 같다. 이때, 새로운 S_2 의 할당 요청이 있다면 프로세서 단편화로 인해 작업을 할당 할 수 없다. 즉, 시스템에는 요청한 프로세서의 수 2와 같은 2개의 유휴 프로세서가 있지만 서브스타를 형성하지 못하기 때문에 할당 할 수 없다. 이때 노드 주소의 연관성을 조사하여 주소를 재생성하면 <그림 5c>와 같이 노드 (*3*)의 서브스타에 할당이 가능하다. 요청한 S_2 의 할당 결과는 <그림 5d> 이다.

3.3 프로세서 할당 해제 알고리즘

프로세서 할당 해제 알고리즘은 <그림 6>과 같이 간단하다.

```
[과정 1] Release Sk ;
[과정 2]
   For(m=k ; 0 ; -1 ){
      • DAT[n-k].ASC = k!-- ;
      • DAT[n-k].RSL := True  ;
      • RelatedSub( );
   }
```

<그림 6> 프로세서 할당 해제 알고리즘 .

[과정 1]에서 해당 노드를 할당 해제한다. 그리고 [과정 2]에서 할당 해제된 노드의 ASC와 RSI를 변경시킨다. 이때, ASC는 현재 할당 해제된 작업의 수만큼 감소하고, RSI는 다른 노드에 의해 할당 가능한 상태인 True로 지정한다. 그리고 RelatedSub() 모듈에서 현재 해제되는 노드와 연관된 모든 노드들의 할당 상태 정보를 수정해 준다. <그림 5b>는 노드 231을 해제하는 하나의 예이다. 노드 231을 할당 해제하면 노드 231의 ASC=0, RSI=True로 지정한다. 그리고 노드 ***의 ASC=4, 노드 **1의 ASC=1로 지정한다.

4. 결론

본 논문에서는 스타그래프 다중처리시스템을 위한 새로운 할당방법으로 DAT 할당방법을 제안하였다. 기존의 할당방법에서는 프로세서의 단편화로 인해 작업을 처리할 서브스타를 찾지 못하면 할당이 지연되는 문제점이 있었다. 이러한 할당지연은 작업의 대기시간 증가시켜 시스템의 성능을 저하시킨다. DAT 할당방법은 프로세서 할당지연이 발생하면 각 프로세서의 주소와 상태정보를 저장한 DAT를 사용하여 프로세서 주소들간의 연관성을 찾아 주소를 재생성한다. 이 DAT를 사용하여 단편화된 프로세서의 주소를 재생성하고, 새로운 가용 서브스타를 형성하여 할당함으로써 작업의 대기시간을 줄인다.

향후 연구과제는 본 논문에서 제안한 DAT 할당방법의 유효성을 검증하는 것이다. 현재는 시뮬레이터를 개발하여 시뮬레이션을 진행하고 있다. 시뮬레이션에서는 기존의 할당방법 보다 DAT 할당방법이 시스템의 성능 향상면에서 우수함을 입증 할 것이다.

5. 참 고 문 헌

[1] Khaled Day, Anand Tripathi, "Comparative Study of Topological Properties of Hypercubes and Star graphs," IEEE Trans. on Parallel and Distributed System, Vol. 5, No. 1, pp. 31-38, Jan. 1994

[2] Sheldon B. Akers, fellow,"A Group-Theoretic Model for Symmetric Interconnection Networks", IEEE Trans. on Computers, Vol. 38, No. 4, pp.555-566, Apr. 1989.

[3] Shahram Latifi, "Task Allocation in the Star Graph," IEEE Trans.and Distributed System,Vol.5, No. 11, pp. 1220- 1224, Nov.1944

[4] Fan Wu and Ching-Chi HSU, "Sg-Lattice : A Model for Processor Allocation for the Star Graph," IEICE Trans. Information&Systems, Vol. E82-D, No. 3, pp. 637-644, Mar. 1999

[5] Yuh-Shyan Chen, Jang-Ping Sheu, "A Fault-Tolerant Reconfiguration Scheme in the Faulty Star graph," In Proceedings of 1996 Second International Conference on Algorithm & Architectures for Parallel Processing, pp. 241-248, 1996

[6] Yordan Rouskov, Shahram Latifi and Pradip K Srimani "Combinatorial Analysis of Star Graph Networks", Technical Report CS-95-101

[7] Ming-Syan Chen and Kang G. Shin, "Processor Allocation in an N-Cube Multiprocessor Using gray codes", IEEE Trans. on COMP. Vol.c-36, No. 12, pp.1396-1407, Dec. 1987.