Truyền thông cộng tác
(Collective communication)

Hà nội, 6/2008

Đại học Bách khoa Hà Nội

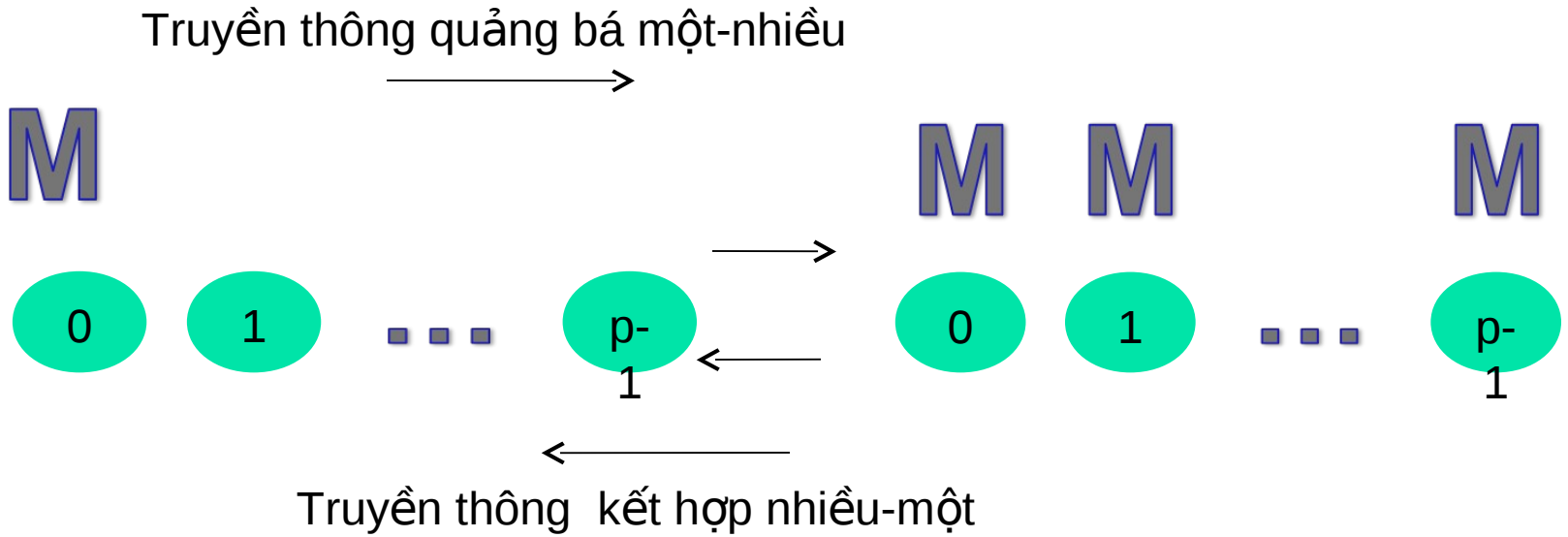
Center of High Performance Computing
Hanoi University of Technology
{hpcc@mail.hut.edu.vn}

Nội dung

- Truyền thông kết hợp kiểu nhiều-một và truyền thông quảng bá kiểu một-nhiều
- Truyền thông kết hợp/quảng bá kiểu Nhiều-Nhiều
- Phép toán All-Reduce và Prefix-Sum
- Phép toán Scatter và Gather
- Truyền thông Nhiều-Nhiều đặc biệt
- Phép dịch vòng
- Cải tiến tốc độ của một số phép toán truyền thông
- Một số định tuyến truyền thông cộng tác trong MPI

Truyền thông kết hợp kiểu nhiều-một/ truyền thông quảng bá kiểu một-nhiều

- Truyền thông kết hợp nhiều-một/truyền thông quảng bá một-nhiều tạo thành một cặp truyền thông.



Được dùng trong nhiều giải thuật quan trọng như: nhân ma trận-vector, phép khử Gause, tìm đường đi ngắn nhất, nhân vector.

Topology cho truyền thông quảng bá một-nhiều

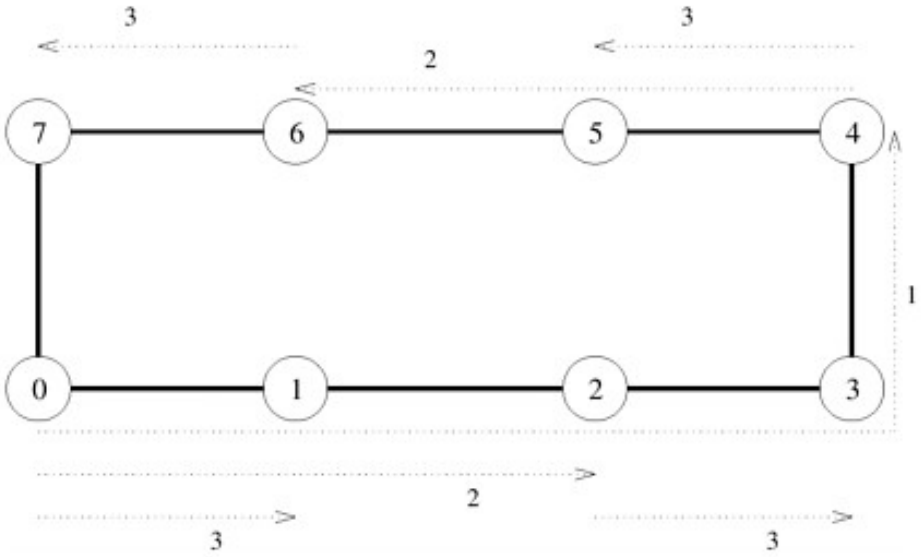
- Một cách tự nhiên, ta thường tiến hành truyền thông quảng bá một-nhiều bằng cách gửi tuần tự (p-1) thông điệp từ nguồn tới (p-1) đích.
- Tuy nhiên, cách trên là không hiệu quả:
  - Tiến trình nguồn bị hiện tượng thắt cổ chai
  - Giảm hiệu suất mạng truyền thông: tại một thời điểm chỉ có một cặp nút hoạt động.

- Xét truyền thông quảng bá một-nhiều trong các topology khác nhau:
  - Topology vòng/ mảng tuyến tính
  - Topology lưới
  - Topology siêu lập phương

Ring or Linear Array Topology

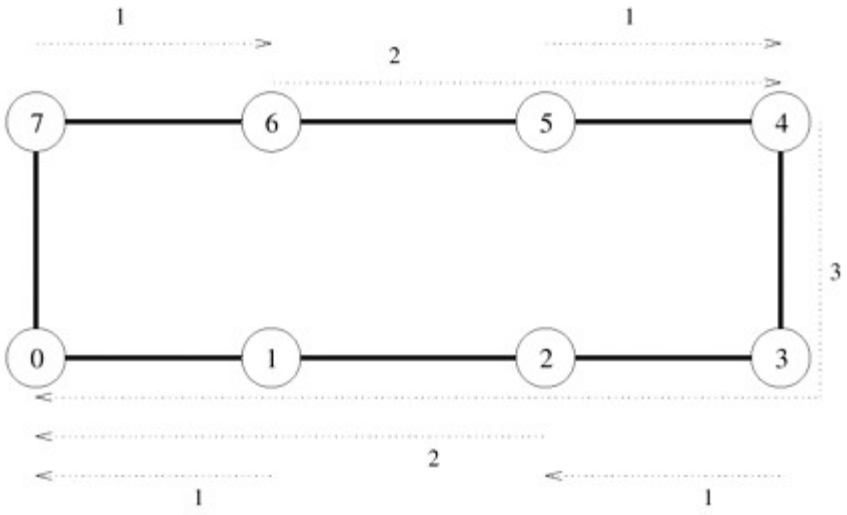
- Sử dụng kỹ thuật *Nhân đôi đệ qui* (recursive doubling) như sau:
  - Tiến trình nguồn gửi một thông điệp đến một tiến trình j bất kỳ
  - Sau khi kết thúc, hai tiến trình có thể *đồng thời* gửi thông điệp cho các tiến trình khác đang đợi
  - Quá trình tiếp tục cho đến khi toàn bộ tiến trình nhận được dữ liệu
  - Dữ liệu có thể được quảng bá chỉ trong *log(p)* bước

Truyền thông quảng bá một-nhiều trên vòng 8 nút



- Trong mỗi bước, chọn đích cẩn thận, đích ảnh hưởng đến hiệu năng.
- Thông điệp đầu tiên gửi từ nút 0 cho nút xa nhất với nó (nút 4)
- Trong bước 2: khoảng cách bị giảm một nửa

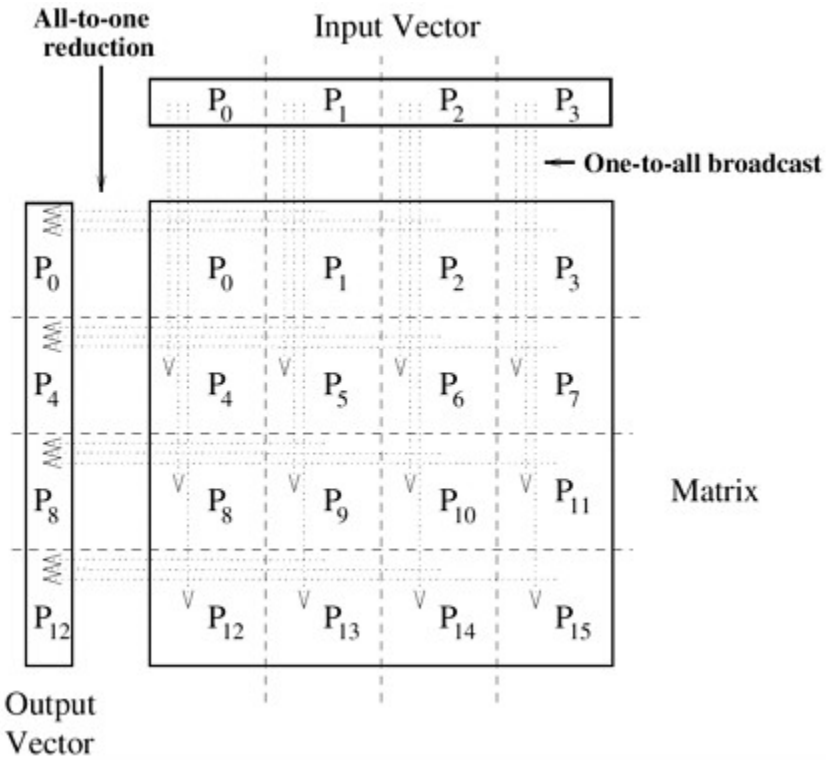
Truyền thông kết hợp nhiều-một trên vòng 8 nút



- Đơn giản, ta đảo ngược hướng và chuỗi truyền thông.
- Đầu tiên, các nút lẻ gửi dữ liệu sang nút chẵn ngay trước nó. Nội dung kết hợp vào nút chẵn
- Còn lại 4 nút: 0,2,4,6
- Nút 0 + nút 2 = nút 0; nút 4 + nút 6 = nút 4;
- Nút 4 + nút 0 = nút 0;

Ví dụ nhân ma trận-vector

- Mỗi dòng của ma trận cần phải nhân với vector
- Bước1: Truyền thông quảng bá một-nhiều:
  - Mỗi phần tử của vector là một nguồn
  - Quảng bá đến cột tương ứng trong ma trận
  - Mỗi cột là một mảng tuyến tính n phần t
- Bước 2: Với mỗi tiến trình
  - Nhân phân tử của ma trận với phẩn tử vừa nhận được
- Bước 3: Tiến hành truyền thông kết hợp nhiều-một:
  - Trên mỗi dòng của ma trận tiến trình
  - Tiến trình đầu tiên của ma trận là đích

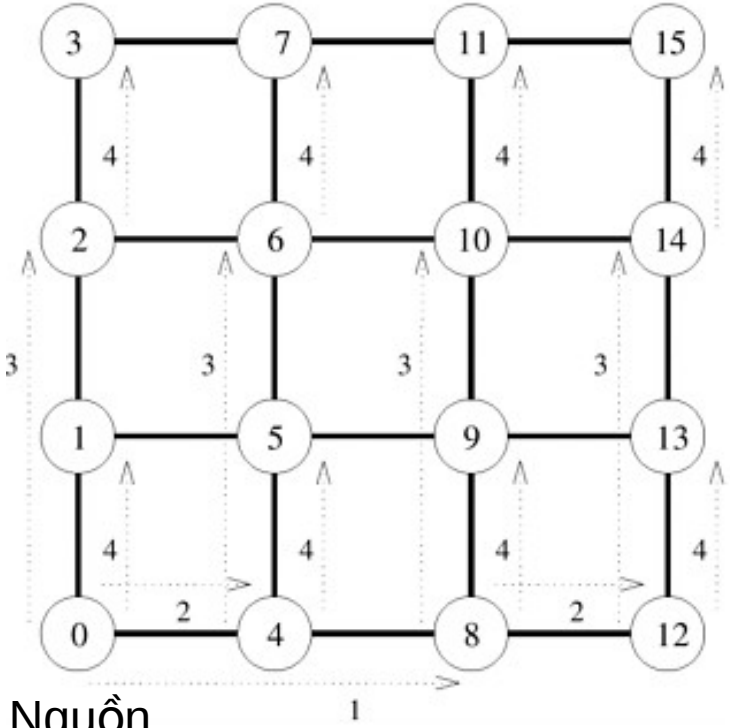


Topology lưới

- Xét lưới vuông có p nút. Mỗi dòng/cột là một mảng tuyến tính $p^{1/2}$ phần tử. Từ lưới này có thể mở rộng cho các lưới khác.
- Toán hạng truyền thông tiến hành theo 2 pha:
  - Pha 1: Tiến hành trên một hoặc nhiều dòng. Mỗi dòng là một mảng tuyến tính.
  - Pha 2: Tiến hành như pha 1, nhưng trên các cột

Truyền thông quảng bá một-nhiều trên lưới vuông 16 nút

- Pha 1: truyền thông quảng bá một-nhiều từ nguồn đến ($p^{1/2}$ – 1) nút cùng hàng
  - Bước 1, 2
- Sau khi các nút trong hàng đã có dữ liệu, tiếp tục bước 2
- Pha 2: truyền thông quảng bá một-nhiều cho các cột tương ứng
  - Bước 3, 4



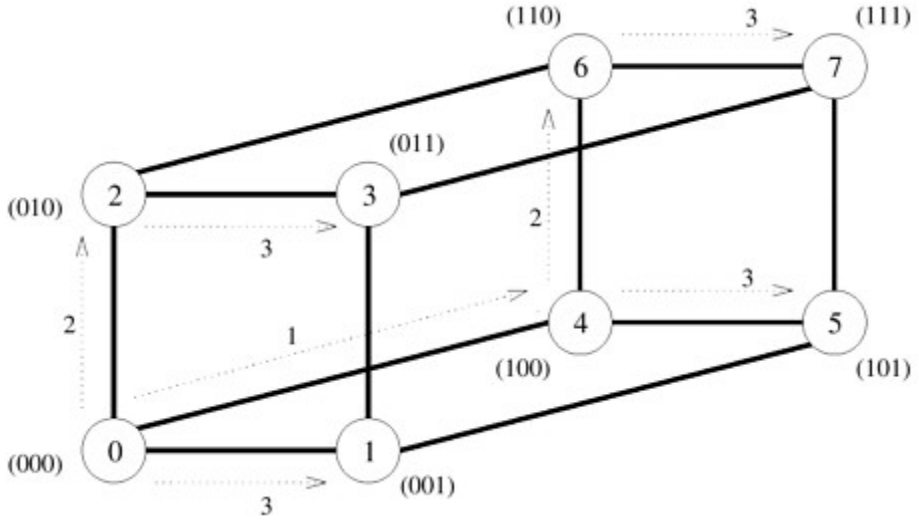
Topology siêu lập phương

- Topology siêu lập phương 2^d nút được coi như một lưới d chiều, trong đó mỗi chiều gồm 2 nút.

- Giải thuật lưới mở rộng cho topology siêu lập phương bằng cách thực hiện trong d bước, hay thực hiện trên từng chiều của lưới.

Truyền thông quảng bá một-nhiều trên siêu lập phương 8 nút

- Siêu lập phương 2^3 chiều
- Coi như lưới 3 chiều, mỗi chiều 2 nút
- Nút 0 là nguồn
- Chiều thể hiện bằng bit có ý nghĩa nhất trong biểu diễn nhị phân của tên nút
- Bắt đầu từ chiều lớn nhất
- Kết quả không phụ thuộc việc chọn chiều truyền thông.

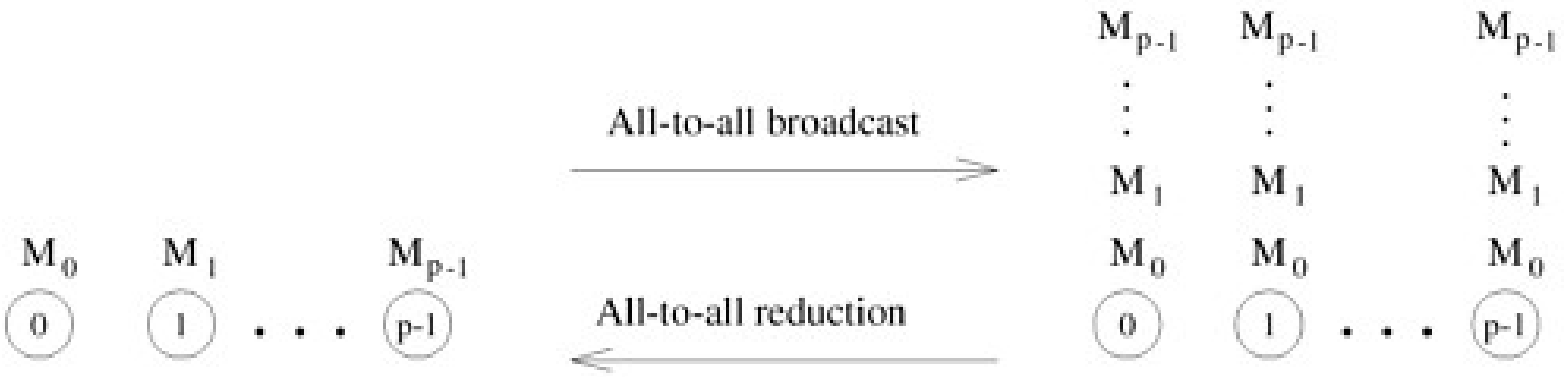


Phân tích chi phí truyền thông

- Giả sử có p tiến trình tham gia quá trình truyền thông
- Dữ liệu quảng bá hoặc kết hợp gồm m từ (word)
- Các thủ tục truyền thông quảng bá một nhiều/kết hợp nhiều-một bao gồm log(p) lần truyền thông điệp đơn
- Thời gian để một truyền thông điệp đơn là: $(t_s+t_w\ m)$
  - T_s : thời gian khởi tạo thông điệp
  - T_w : thời gian truyền một từ (word)
- Tổng thời gian truyền thông:

$(t_s+t_w\ m)\log(p)$

Truyền thông quảng bá/kết hợp kiểu Nhiều-Nhiều



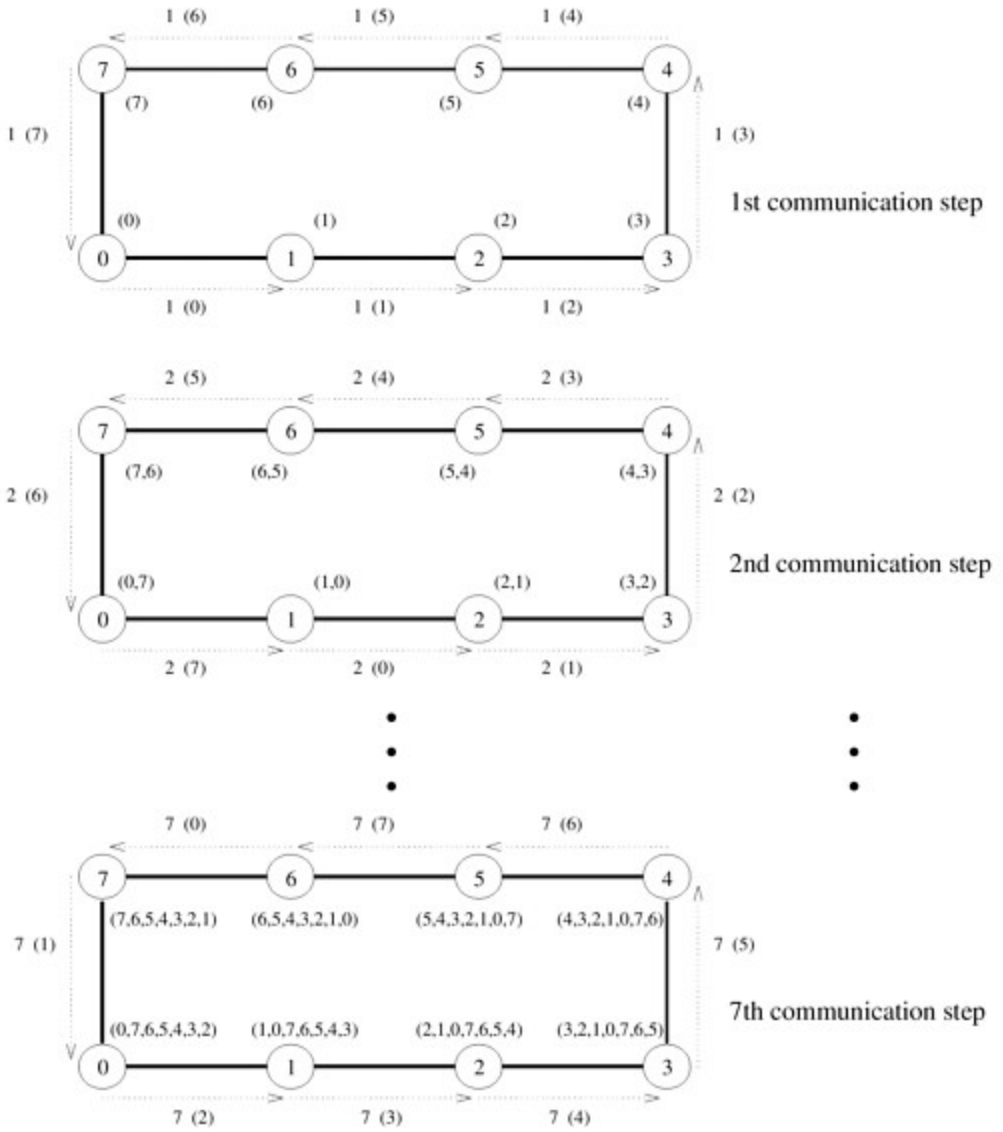
- Truyền thông quảng bá nhiều-nhiều và truyền thông kết hợp nhiều nhiều tạo thành một cặp truyền thông.

Topology tuyến tính

- Các tiến trình liên tục truyền thông đồng thời cho đến khi toàn bộ quá trình truyền thông kết thúc.
- Đầu tiên, mỗi nút gửi dữ liệu của nó cho nút hàng xóm
- Các bước tiếp theo, mỗi nút chuyển tiếp dữ liệu nó nhận được từ nút hàng xóm trong bước trước đến một nút hàng xóm khác.

Truyền thông quảng bá nhiều-nhiều trên vòng 8 nút

- Cách đánh nhãn:
  - 2(7) nằm giữa nút 0 và nút 1: trong bước thứ 2, nút 0 đã nhận dữ liệu của nút 7 từ bước trước
  - (0,7) nằm cạnh nút 0: Gồm nhãn của các nút mà nút 0 đã nhận trong các bước trước đó.



Giải thuật thông điệp quảng bá nhiều-nhiều trên vòng p nút

```
1.   procedure ALL_TO_ALL_BC_RING(my_id, my_msg, p, result)
2.   begin
3.       left := (my_id - 1) mod p;
4.       right := (my_id + 1) mod p;
5.       result := my_msg;
6.       msg := result;
7.       for i := 1 to p - 1 do
8.           send msg to right;
9.           receive msg from left;
10.          result := result ∪ msg;
11.      endfor;
12.  end ALL_TO_ALL_BC_RING
```

Giải thuật thông điệp kết hợp nhiều-nhiều trên vòng p nút

```
1.   procedure ALL_TO_ALL_RED_RING(my_id, my_msg, p, result)
2.   begin
3.       left := (my_id - 1) mod p;
4.       right := (my_id + 1) mod p;
5.       recv := 0;
6.       for i := 1 to p - 1 do
7.           j := (my_id + i) mod p;
8.           temp := msg[j] + recv;
9.           send temp to left;
10.          receive recv from right;
11.      endfor;
12.      result := msg[my_id] + recv;
13.  end ALL_TO_ALL_RED_RING
```

Topology lưới

- Xét lưới 2 chiều, gồm p nút, mỗi chiều gồm $p^{1/2}$ nút.
- Giải thuật tiến hành dựa trên giải thuật cho topology tuyến tính, gồm 2 pha:
  - Pha 1: áp dụng giải thuật tuyến tính cho từng dòng.
  - Pha 2: áp dụng giải thuật tuyến tính cho từng cột.

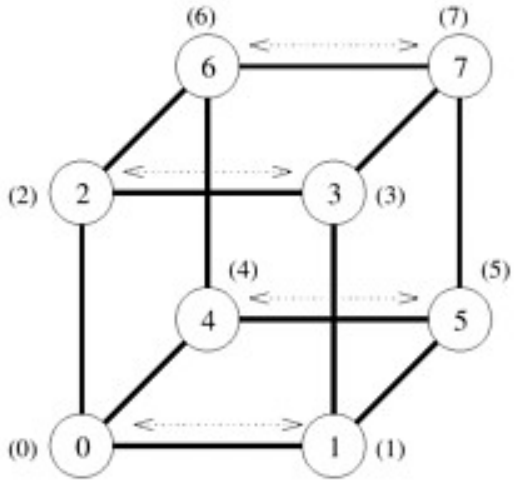
Giải thuật thông điệp quảng bá nhiều-nhiều trên lưới p nút

```
1.   procedure ALL_TO_ALL_BC_MESH(my_id, my_msg, p, result)
2.   begin

/* Communication along rows */
3.       left := my_id - (my_id mod √p) + (my_id - 1)mod √p;
4.       right := my_id - (my_id mod √p) + (my_id + 1) mod √p;
5.       result := my_msg;
6.       msg := result;
7.       for i := 1 to √p - 1 do
8.           send msg to right;
9.           receive msg from left;
10.          result := result ∪ msg;
11.      endfor;

/* Communication along columns */
12.      up := (my_id - √p) mod p;
13.      down := (my_id + √p) mod p;
14.      msg := result;
15.      for i := 1 to √p - 1 do
16.          send msg to down;
17.          receive msg from up;
18.          result := result ∪ msg;
19.      endfor;
20.  end ALL_TO_ALL_BC_MESH
```

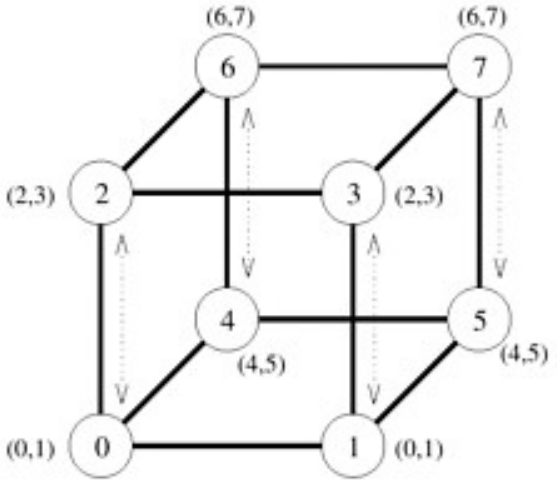
Topology siêu lập phương

- Xét siêu lập phương gồm p nút
- Giải thuật siêu lập phương là mở rộng của giải thuật lưới cho log(p) chiều.
- Thủ tục đòi hỏi log(p) bước
- Mỗi bước tiến hành theo một chiều xác định của siêu lập phương p-node
- Tại mỗi bước:
  - Các cặp nút truyền dữ liệu cho nhau
  - Tạo bản copy để gửi đi trong bước tiếp theo

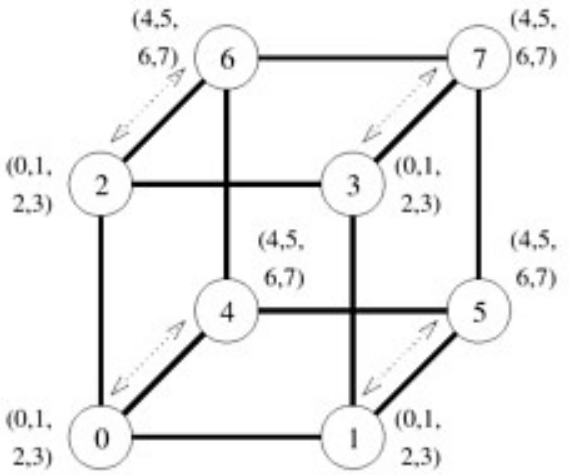
Truyền thông quảng bá nhiều-nhiều trên siêu lập phương 8 nút

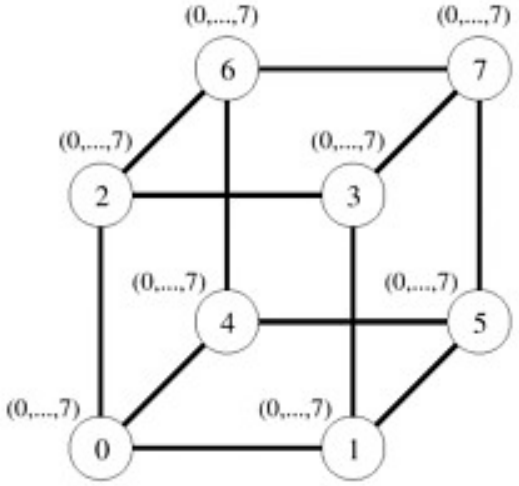


(a) Initial distribution of messages



(b) Distribution before the second step





Giải thuật thông điệp quảng bá nhiều-nhiều trên siêu lập phương p nút

```
1.   procedure ALL_TO_ALL_BC_HCUBE(my_id, my_msg, d, result)
2.   begin
3.       result := my_msg;
4.       for i := 0 to d - 1 do
5.           partner := my id XOR 2^i;
6.           send result to partner;
7.           receive msg from partner;
8.           result := result ∪ msg;
9.       endfor;
10.  end ALL_TO_ALL_BC_HCUBE
```

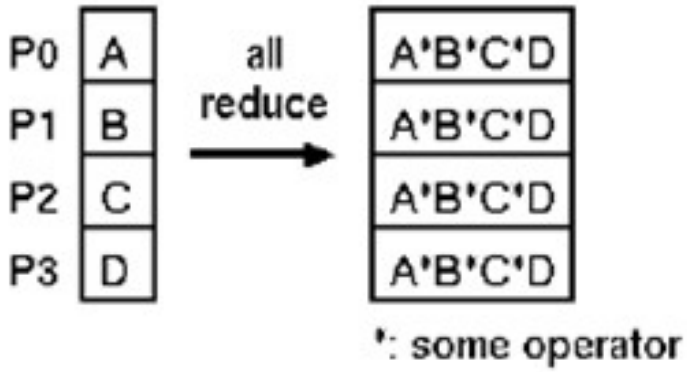
- Truyền thông bắt đầu từ chiều thấp nhất của siêu lập phương, sau đó xử lý theo các chiều lớn hơn
- Tại mỗi vòng lặp, tiến trình truyền thông với tiến trình có bit ý nghĩa nhất thứ i

Phân tích chi phí truyền thông

- Topology tuyến tính p nút:
  - Gồm (p-1) bước truyền thông
  - $T = (t_s + t_w\ m)\ (p-1)$
- Topology lưới p nút
  - Pha 1: có $p^{1/2}$ bước
    - $T_1 = (t_s + t_w\ m)\ (p^{1/2} -1)$
  - Pha 2: mỗi bản tin có kích thước $(m.p^{1/2})$, $(p^{1/2} -1)$ bước
    - $T_2 = (t_s + t_w\ m.p^{1/2})\ (p^{1/2} -1)$
  - Tổng: $T = 2\ t_s\ p^{1/2} + t_w.m.(p-1)$
- Topology siêu lập phương p nút
  - Kích thước thông điệp trao đổi trong bước thứ i là $(2^{i-1} \times m)$
  - Gồm log(p) b

$$T = \sum_{i=1}^{\log p} (t_s + 2^{i-1} t_w m)$$
$$= t_s \log p + t_w m(p-1).$$

Phép toán all-reduce



- Một số cách thực hiện:
  - Một truyền thông kết hợp nhiều-một và truyền thông quảng bá một-nhiều
  - Cách đơn giản hơn là dùng truyền thông quảng bá nhiều-nhiều. Cuối mỗi bước thay vì nối dữ liệu, ta tiến hành phép toán nào đó (cộng, trừ, nhân, chia,..) trên các toán hạng

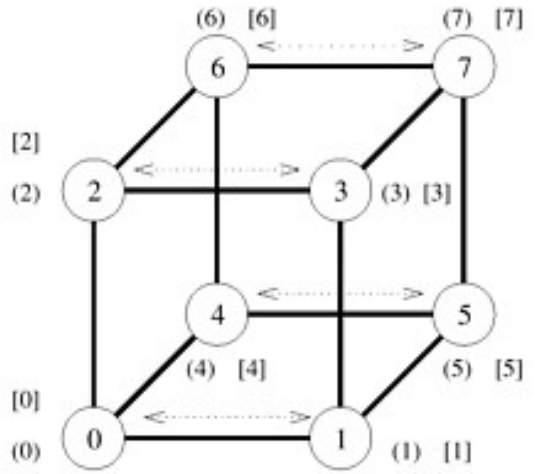
Phép toán prefix-sum (phép toán scan)

- Bài toán:
  - Cho p số n_0, n_1, ... N_{p-1} trên p nút
  - Hãy tính tổng $s_k = \Sigma_{i=0}^{k} n_i$ đối với các giá trị k từ 0 đến p-1

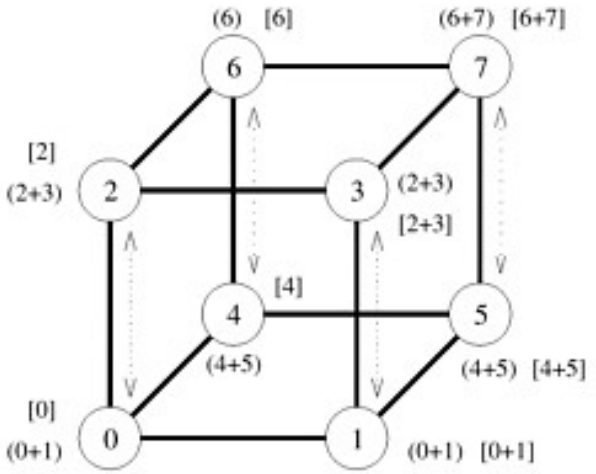
- Ví dụ: cho chuỗi <3,1,4,0,2> thì chuỗi prefix-sum là <3,4,8,8,10>

Ví dụ: prefix-sum trên siêu lập phương 8 nút

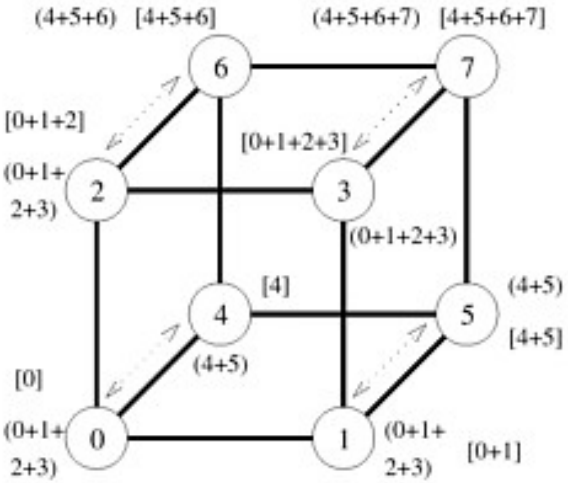
- Ngoặc vuông:
  - Prefix-sum cục bộ
- Ngoặc tròn:
  - Nội dung bộ đệm gửi ra ngoài

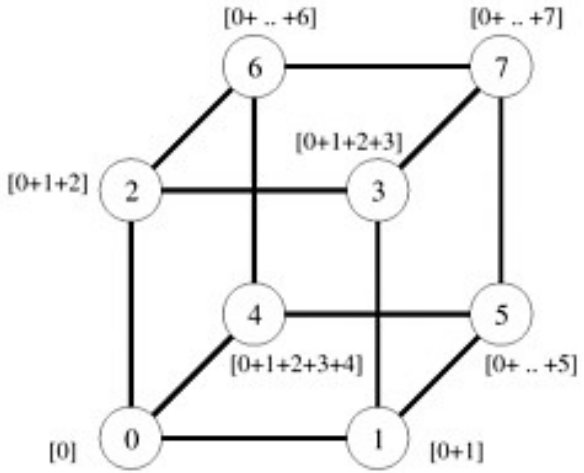


(a) Initial distribution of values

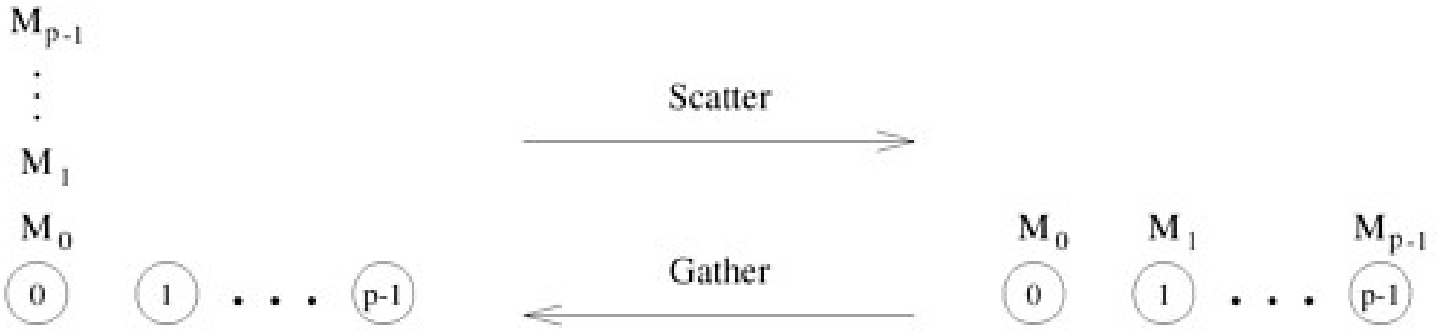


(b) Distribution of sums before second step



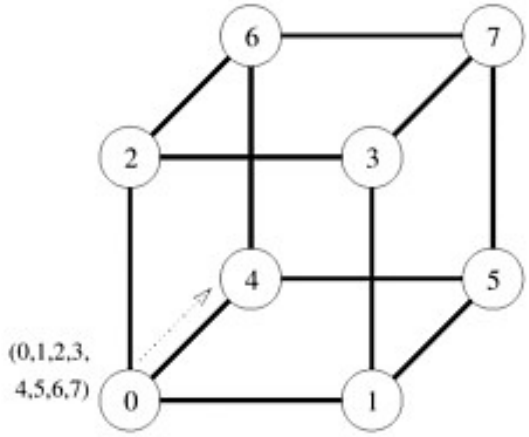


Truyền thông một-nhiều đặc biệt
Phép toán scatter và gather

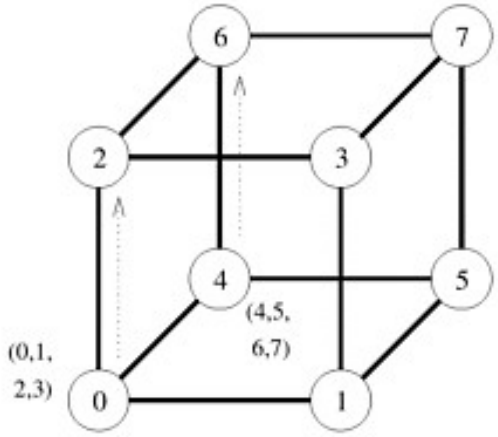


- Khác so với truyền thông quảng bá một-nhiều:
  - Trong truyền thông quảng bá một-nhiều thì nguồn khởi tạo p thông điệp **giống nhau** và gửi cho từng nút các bản sao
  - Phép toán scatter không có sự nhân bản dữ liệu
- Phép toán gather
  - Không bao gồm bất kỳ sự kết hợp dữ liệu nào.
- Phép toán scatter và gather tạo thành một cặp

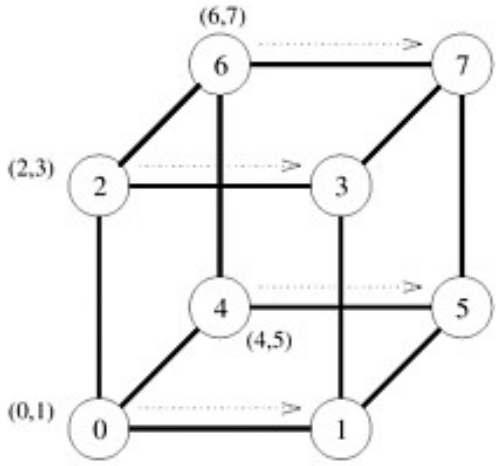
Phép toán scatter với siêu lập phương 8 nút



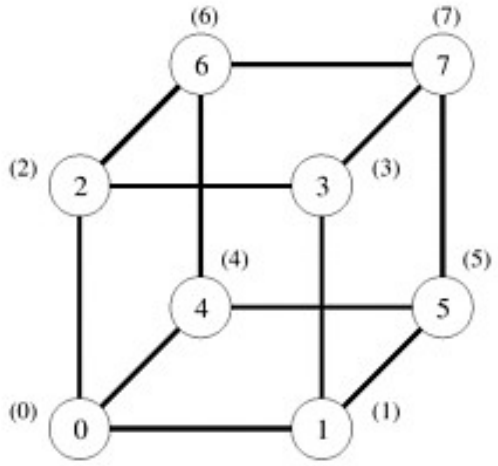
(a) Initial distribution of messages



(b) Distribution before the second step



(c) Distribution before the third step



(d) Final distribution of messages

Truyền thông nhiều-nhiều đặc biệt

- Mỗi nút gửi một thông điệp ***khác nhau*** kích thước m đến mọi nút khác

- Ta thấy, thực chất giống như chuyển vị mảng 2 chiều.
- Còn được gọi là phép **Trao đổi toàn phần**
- Dùng nhiều cho các giải thuật: biến đổi fourier, chuyển vị ma trận, hay một số phép kết nối cơ sở dữ liệu song song.

Phép dịch vòng

- Thuộc lớp các phép toán truyền thông rộng hơn là phép hoán vị.
- Phép hoán vị: sắp xếp lại các dữ liệu một-một một cách đồng thời, trong đó từng nút gửi gói tin m từ cho nút duy nhất khác.
- Phép dịch vòng q: là phép toán trong đó nút thứ i gửi dữ liệu cho nút thứ $((i + q) \bmod p)$ trong số p nút, trong đó $0 < q < p$
- Thường sử dụng trong các ứng dụng liên quan đến ma trận, ứng dụng liên quan đến chuỗi và so khớp mẫu ảnh

Cải tiến tốc độ của một số phép toán truyền thông

- Các phép toán truyền thông đã xét:
  - Thông điệp gốc không chia được thành các phần nhỏ hơn
  - Mỗi nút có một cổng duy nhất để gửi và nhận dữ liệu
- Cải tiến tốc độ:
  - Chia nhỏ thông điệp thành các phần nhỏ và định tuyến cho các thông điệp
  - Truyền thông đa cổng

Chia nhỏ và định tuyến thông điệp

- T_s tăng, t_w giảm
- Xét bài toán truyền thông với p nút
- Truyền thông quảng bá một-nhiều:
  - Chia bộ dữ liệu m thành p phần kích thước m/p:
    - M_0 , M_1 , … , M_{p-1}
    - Chi phí gửi phần dữ liệu M_i cho nút thứ i, dùng phép toán scatter
      - $t_s \log p + t_w(m/p)(p - 1)$
    - Chi phí gửi các phần tử M_i cho tất cả các nút, dùng truyền thông quảng bá nhiều-nhiều trên topology siêu lập phương:
      - $t_s \log p + t_w(m/p)(p - 1)$
    - Tổng chi phí:
      - $2(t_s \log p + t_w(m/p)(p - 1))$
      - xấp xỉ: $2(t_s \log p + t_w m)$
      - So sánh với truyền một thông điệp lớn: $(t_s+t_w m)\log(p)$
        - T_s tăng
        - T_w giảm log(p)/2

Truyền thông đa cổng

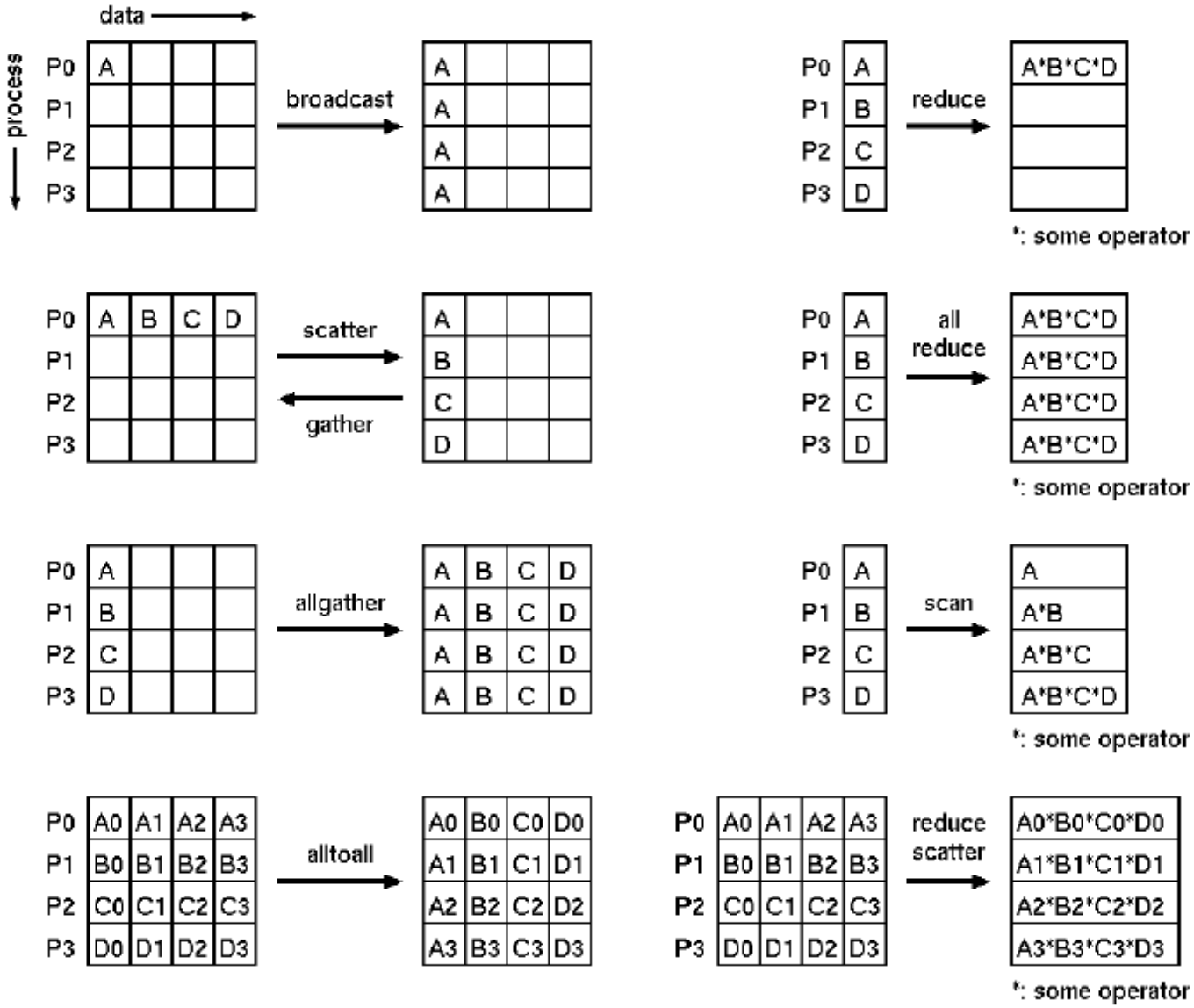
- Trong kiến trúc song song:
  - Một nút có thể có nhiều cổng kết nối với các nút khác nhau:
  - Topology lưới: một nút có 4 cổng
  - Topology siêu lập phương d chiều: mỗi nút có d cổng
- Truyền thông trên 1 cổng: một nút truyền nhận dữ liệu trên một cổng duy nhất tại một thời điểm
- Truyền thông đa cổng: cho phép truyền thông đồng thời trên nhiều kênh của một nút
  - Truyền nhận đồng thời trên cùng một cổng
  - Truyền nhận đồng thời trên nhiều cổng

Truyền thông đa cổng (cont.)

- Topology siêu lập phương p nút:
  - Thời gian truyền thông t_w nhỏ hơn log(p) lần so với dùng một cổng đơn
- Topology tuyến tính và topology lưới:
  - Thời gian truyền thông là không được cải thiện.

- Một số giới hạn:
  - Khó lập trình hơn
  - Thông điệp đủ lớn để chia giữa các kênh
  - Thông điệp lớn, dẫn đến thời gian tính toán cục bộ lớn.
  - Đòi hỏi băng thông bộ nhớ lớn để không ảnh hưởng đến các truyền thông song song
    - Topology siêu lập phương: băng thông bộ nhớ lớn hơn băng thông truyền thông của 1 cổng ít nhất log(p) lần
  - Không phù hợp với bài toán có truyền thông ít

Một số định tuyến truyền thông cộng tác trong MPI

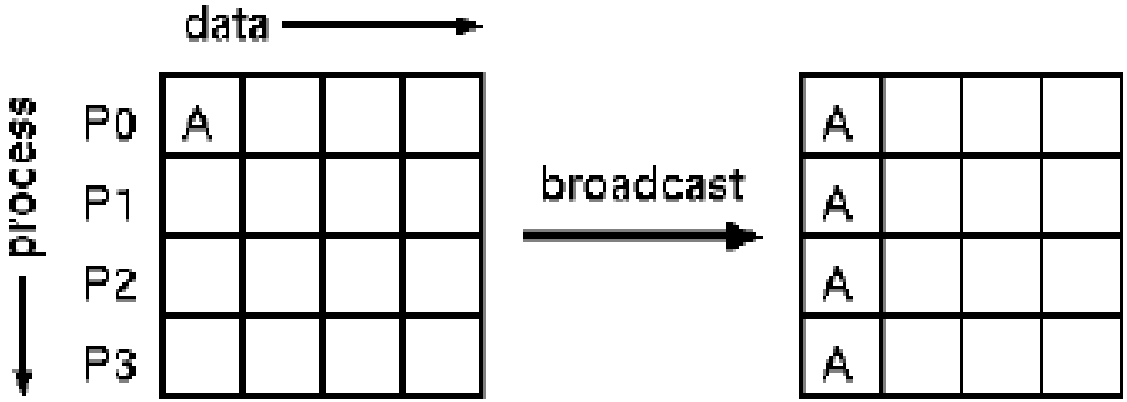


Các dạng truyền thông cộng đồng

- Chia thành 2 nhóm :
  - Nhóm đồng bộ dữ liệu : trao đổi dữ liệu giữa các tiến trình, gồm các phương thức : *broadcast, scatter, gather, allgather* và *alltoall*
  - Nhóm thao tác dữ liệu : thực hiện một thao tác nào đó trên dữ liệu ***của tất cả các tiến trình***. Các thao tác có thể là các phép tính đơn giản hoặc các hàm phức tạp. Các phương thức thuộc nhóm này : *reduce, allreduce, scan, reducescatter*.

Nhóm đồng bộ dữ liệu

- Broadcast :
  - `int MPI_Bcast (void *buffer, int count, MPI_Datatype datatype, int root, MPI_Comm comm)`
  - Gửi một thông điệp từ tiến trình có rank là root tới tất cả các tiến trình trong communicator comm bao gồm chính nó.
    - Rank của tiến trình gọi MPI_Bcast = root: tiến trình gửi
    - Rank của tiến trình gọi MPI_Bcast khác giá trị root: tiến trình nhận

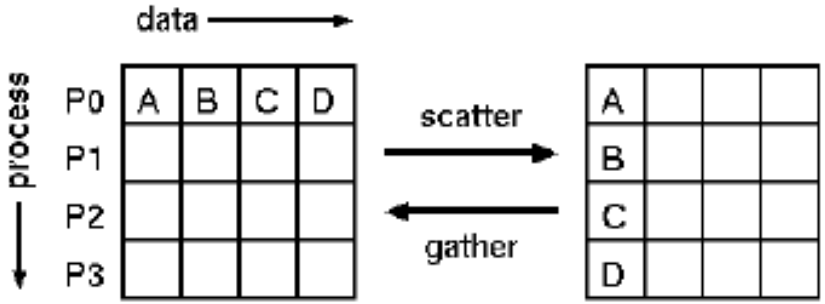


Nhóm đồng bộ dữ liệu

- Gather :

```
int MPI_Gather (void* sbuf, int scount, MPI_Datatype
  stype, void* rebuf,  int rcount, MPI_Datatype
  rtype, int root, MPI_Comm comm)
```

  - Các nội dung này được tiến trình gốc lưu theo đúng trật tự rank.
  - Tiến trình gửi: rebuf có thể null
  - Tiến trình nhận: sbuf có thể null
  - *Tham khảo thêm : MPI_Gatherv()*

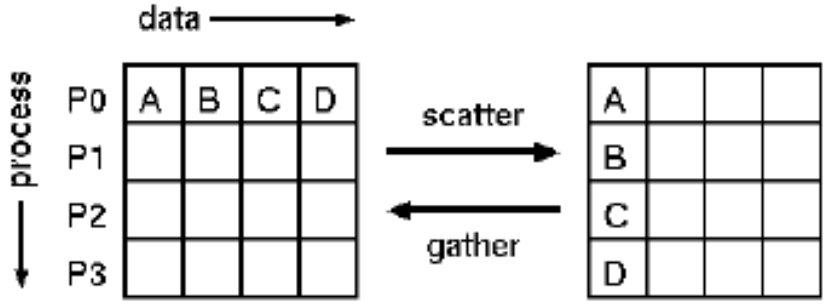


Định tuyến MPI_Scatter

- Scatter:

```
int MPI_Scatter (void* sbuf, int scount,
  MPI_Datatype stype, void* rebuf, int rcount,
  MPI_Datatype rtype, int root, MPI_Comm comm)
```

  - Thực hiện thao tác ngược với gather
  - Tiến trình gốc gửi nội dung send buffer tới các tiến trình khác. Mỗi tiến trình nhận và lưu lại theo thứ tự rank.
  - Ý nghĩa các tham số giống thao tác gather.
  - *Tham khảo thêm : MPI_Scatterv()*

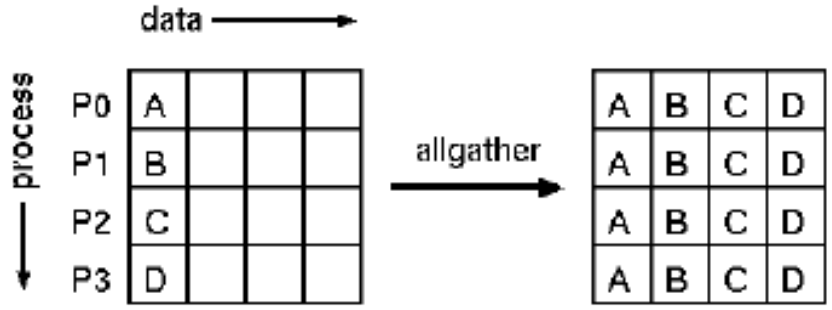


Định tuyến MPI_Allgather

- Gather-to-all :

```
int MPI_Allgather (void* sbuf, int scount,
  MPI_Datatype stype, void* rebuf, int rcount,
  MPI_Datatype rtype, MPI_Comm comm)
```

  - Tác dụng giống như thao tác Gather nhưng tất cả các tiến trình đều nhận kết quả thay vì chỉ tiến trình gốc.
  - Ý nghĩa tham số tương tự thao tác Gather.
  - *Tham khảo thêm : MPI_Allgatherv()*

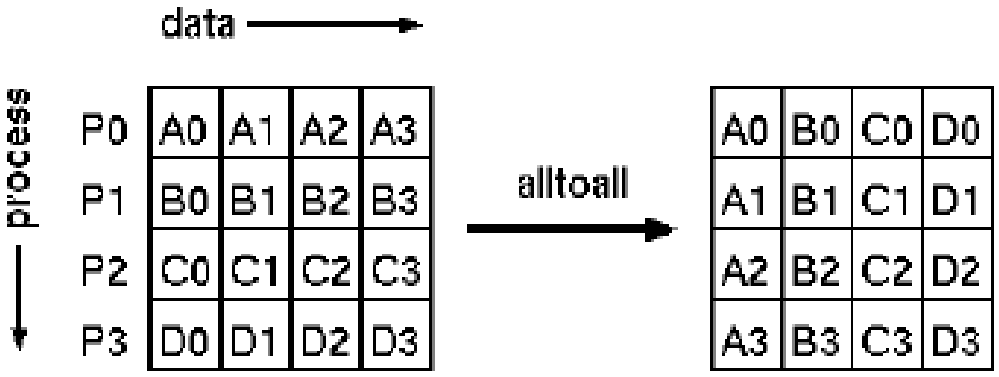


Định tuyến MPI_Alltoall

- All to all scatter/gather:

```
int MPI_Alltoall (void* sbuf, int scount,
  MPI_Datatype stype, void* rbuf, int rcount,
  MPI_Datatype rtype, MPI_Comm comm)
```

  - Chính là truyền thông quảng bá/kết hợp nhiều nhiều đặc biệt
  - Block j gửi bởi tiến trình i sẽ được đặt vào block i của tiến trình j.
  - *Tham khảo thêm : MPI_Alltoallv()*



Nhóm thao tác dữ liệu

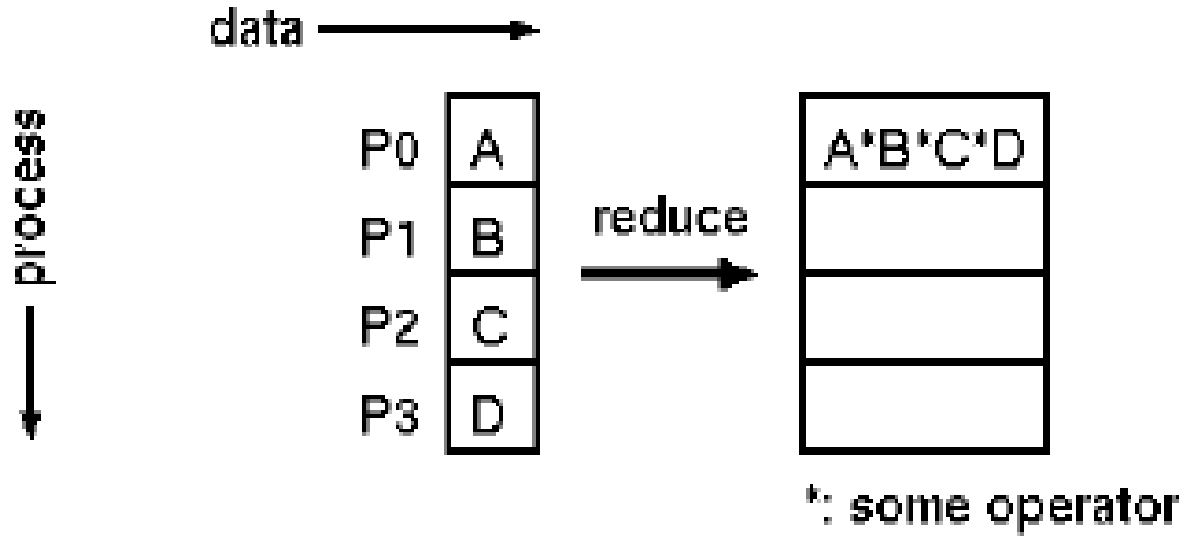
- Thực thi một tác vụ thu gọn toàn cục, ví dụ tính tổng, tìm số lớn nhất, tìm số nhỏ nhất, ...
- Các tác vụ này có thể
  - Tác vụ có sẵn
  - Hoặc do người dùng tự định nghĩa.

Định tuyến MPI_Reduce

- Reduce :

```
int MPI_Reduce (void* sbuf, void* rbuf, int count,
  MPI_Datatype datatype, MPI_Op op, int root,
  MPI_Comm comm)
```

  - Kết hợp giá trị các phần tử từ các tiến trình gửi theo thao tác op, kết quả lưu tại tiến trình root.

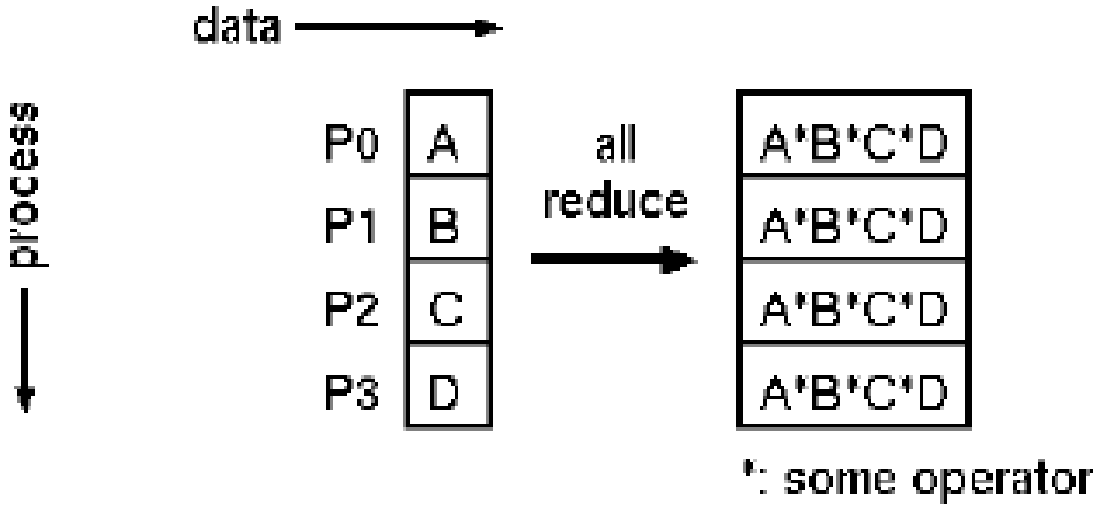


Định tuyến MPI_Allreduce

- All reduce:

```
int MPI_Allreduce (void* sbuf, void* rbuf, int
  count, MPI_Datatype  datatype, MPI_Op op, MPI_Comm
  comm)
```

  - Giống thao tác reduce, nhưng kết quả được gửi đến cho tất cả các tiến trình trong communicator.
  - Ý nghĩa các tham số tương tự hàm reduce



Các phép toán thao tác dữ liệu

MPI Name	Function
MPI_MAX	Maximum
MPI_MIN	Minimum
MPI_SUM	Sum
MPI_PROD	Product
MPI_LAND	Logical AND
MPI_BAND	Bitwise AND
MPI_LOR	Logical OR
MPI_BOR	Bitwise OR
MPI_LXOR	Logical exclusive OR
MPI_BXOR	Bitwise exclusive OR
MPI_MAXLOC	Maximum and location
MPI_MINLOC	Minimum and location

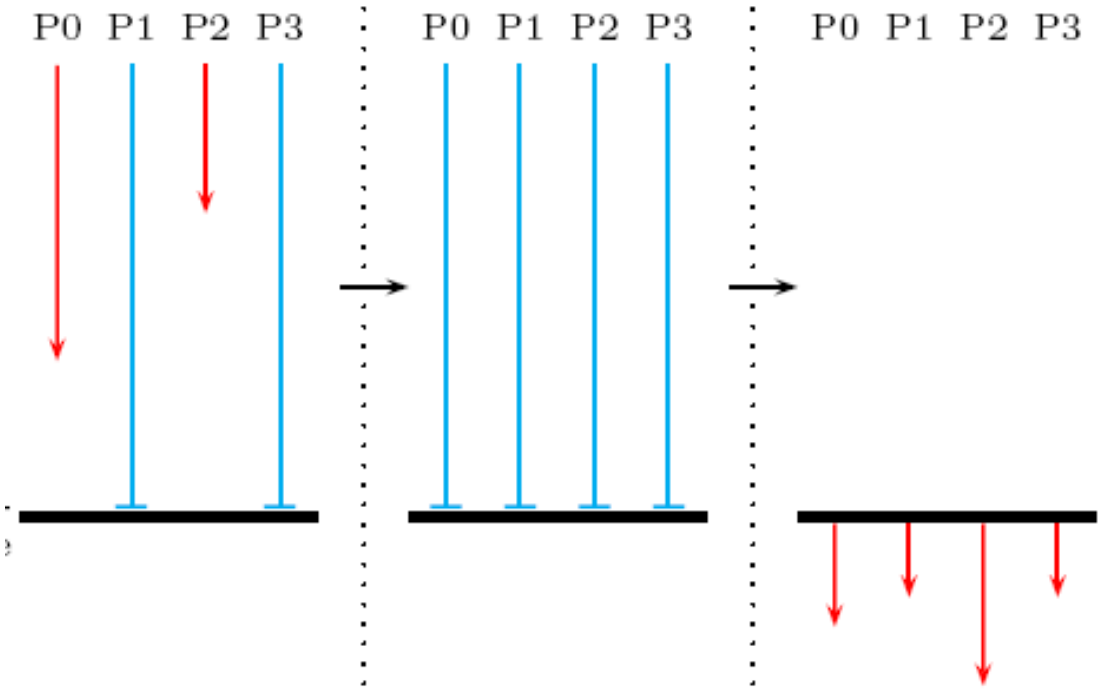
Tham khảo thêm

- MPI_Reduce_scatter()
- MPI_Scan()
- Các phép toán do người dùng tự định nghĩa

```
void my_operator (void *invec, void *inoutvec, int *len,
                  MPI_Datatype *datatype)

    for (i=1 to len)
        inoutvec(i) = inoutvec(i) χ invec(i)

int MPI_Op_create (MPI_User_function *function,
  int commute, MPI_Op *op)
```

Barrier Synchronizations

- Là cơ chế đồng bộ, tạm ngừng tiến trình.
- Tiến trình chỉ có thể tiếp tục thực hiện khi tất cả các tiến trình khác đã sẵn sàng.
- Đảm bảo các tiến trình cùng bắt đầu thực hiện một (chuỗi) các thao tác nào đó.



Case study: tìm kiếm song song

- Lập trình đếm số lần xuất hiện của một khoá (key) trong một dãy cho trước, sử dụng các hàm truyền thông cộng tác.
- Dãy số cho trước được lưu trong một file text data.in gồm N phần tử có định dạng :
  - Số_phần tử_N
  - Phần_tử_thứ_1 phần_tử_thứ_2 .... phần_tử_thứ_N
- Các phần tử phân biệt bằng khoảng trắng.
- Yêu cầu chọn số tiến trình thực hiện là ước của số phần tử trong dãy.
- Khoá cần tìm kiếm được truyền vào dưới dạng tham số của chương trình. Ví dụ:
  - `$ mpirun -np 4 program 23`
- Sử dụng các hàm truyền thông cộng đồng : MPI_Bcast(), MPI_Scatter(), MPI_Reduce().

Case study: tìm kiếm song song

#

#

Case study: tìm kiếm song song

Hàm đọc dữ liệu từ file data.in :

```
#define FILE_NAME   "data.in"

int * loadData(int * NoElements){
     FILE * dataf;
     int * dataSet;
     int i;

     /* mo file */
     dataf = fopen(FILE_NAME, "r");
     if (!dataf){ /* loi mo file */
          printf("Loi khi mo file %s\n", FILE_NAME);
          return NULL;
     }

     /* doc so phan tu */
     fscanf(dataf, "%d", NoElements);

     /* cap phat vung nho cho mang du lieu */
     dataSet = (int*) malloc(sizeof(int) * (*NoElements));

     /* doc mang du lieu */
     for ( i=0 ; i< (*NoElements) ; i++ )
          fscanf(dataf, "%d", dataSet+i);

     fclose(dataf);
     return dataSet;
}
```

Case study: tìm kiếm song song

Hàm đếm số lần xuất hiện khoá key trong dãy arr

```
int search(int *arr, int ne, int key){
    int i;
    int count = 0;

    for ( i=0 ; i<ne ; i++ )
        if ( arr[i] == key )
            count++;

    return count;
}
```

Case study: tìm kiếm song song

Hàm rank0():

```
int rank0(int key){
    int tne;            // so phan tu cua day;
    int *dataSet;   // mang du lieu, duoc cap phat dong
    int np;                     // tong so tien trinh
    int sne;            // so phan tu gui den cho moi tien trinh
    int count;          // so lan xuat hien cua khoa key
    int tmp;
    int * buff;

    /* lay ve so tien trinh */
    MPI_Comm_size(MPI_COMM_WORLD, &np);

    /* doc du lieu tu file */
    dataSet = loadData(&tne);

    /* tinh so phan tu gui cho moi tien trinh */
    sne = tne / np;

    /* cap phat du lieu cho buff*/
    buff = (int*) malloc(sizeof(int) * sne);
```

Case study: tìm kiếm song song

```
1. /* phan tan du lieu */
2. MPI_Bcast(&sne, 1, MPI_INT, 0, MPI_COMM_WORLD);
3. MPI_Scatter(dataSet, sne, MPI_INT, buff, sne, MPI_INT, 0,
   MPI_COMM_WORLD);
4. /* dem so lan xuat hien khoa tren mien con ma no dam nhiem */
5. tmp = search(buff, sne, key);

6. /* nhan du lieu tu cac tien trinh con gui ve */
7. MPI_Reduce(&tmp, &count, 1, MPI_INT, MPI_SUM, 0,
   MPI_COMM_WORLD);
8. /* thong bao ket qua */
9. printf("So lan xuat hien khoa key trong day la %d\n", count);

10.free(dataSet);
11.free(buff);
12.return 0;
13.}
```

Case study: tìm kiếm song song

Hàm ranki()

```
1.  int ranki(int key){
2.       int ne;         // so phan tu cua mang con
3.       int *buff;      // mang con
4.       int count;
5.       /* Nhan du lieu tu tien trinh 0 */
6.       MPI_Bcast(&ne, 1, MPI_INT, 0, MPI_COMM_WORLD);
7.       buff = (int*) malloc(sizeof(int) * ne);
8.       MPI_Scatter(NULL, ne, MPI_INT, buff, ne, MPI_INT, 0, MPI_COMM_WORLD);
9.
10.      /* thuc hien thao tac tim kiem */
11.      count = search(buff, ne, key);
12.
13.      /* gui ket qua ve cho tien trinh goc */
14.      MPI_Reduce(&count, NULL, 1, MPI_INT, MPI_SUM, 0, MPI_COMM_WORLD);
15.
16.      return 0;
17. }
```

Kết luận

- Các kiểu truyền thông cộng tác
  - Truyền thông kết hợp
    - Nhiều-một
    - Nhiều-Nhiều
  - Truyền thông quảng bá
    - Một-Nhiều
    - Nhiều-Nhiều
- Phân tích chi phí truyền thông
- Ảnh hưởng các topology kết nối mạng đến chi phí truyền thông
- Đồng bộ barrier

Truyền thông điểm điểm