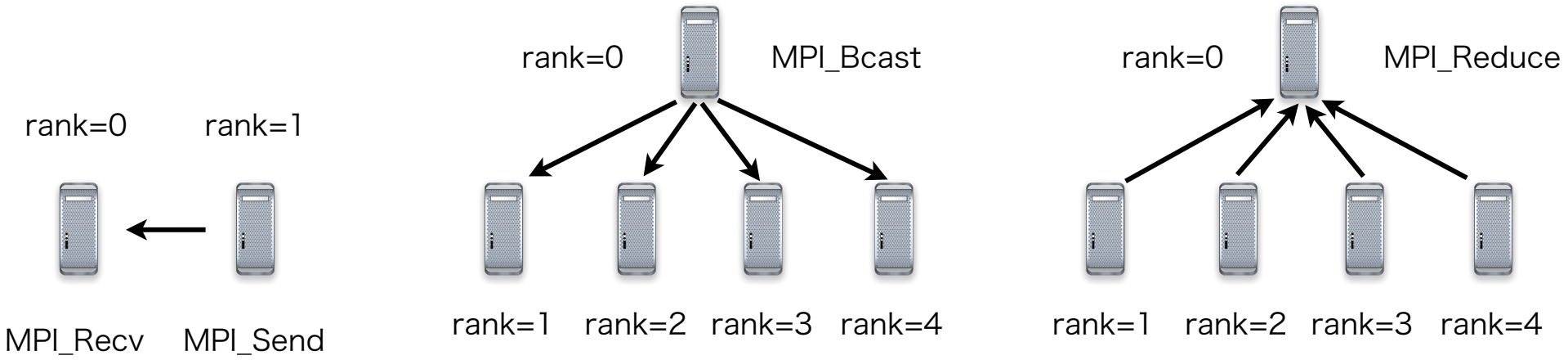
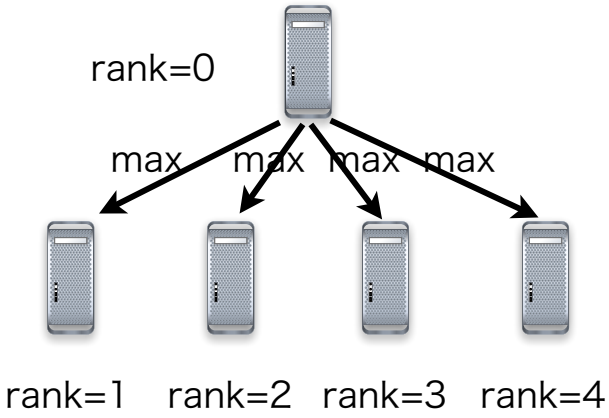
ノード間通信

- 1対1通信を行う関数 MPI_Send と MPI_Recv
- 複数ノードへの一斉送信・受信を行う MPI_Bcast
- 複数ノードのデータをあるノードへ集約する MPI_Reduce



ノード数が増えると、MPI_Send, MPI_Recv による
1対1通信は処理が面倒になる

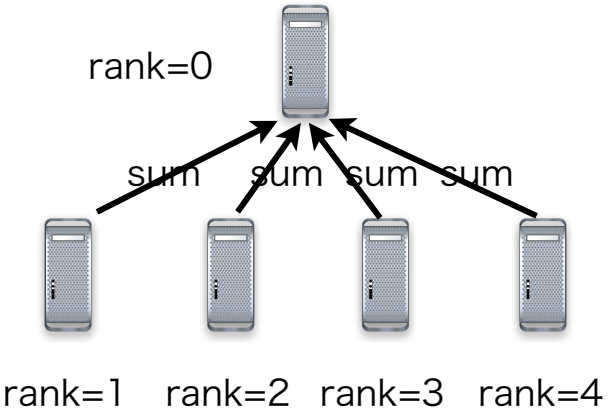
1対1通信で数列和を並列計算



1) rank 0 から 1, 2, 3, 4 へ数列和の上限 max を MPI_Send, MPI_Recv で送信する

2) 各ノードで rank の倍数の和 sum を求める

```
sum = 0;
for(i = rank; i <= max; i += 2) sum += i;
printf("The sub sum in rank %d is %d\n", rank, sum);
```



3) rank 1, 2, 3, 4 から rank 0 へ sum を MPI_Send, MPI_Recvで送信する

コーディングが大変！

MPI_Bcast

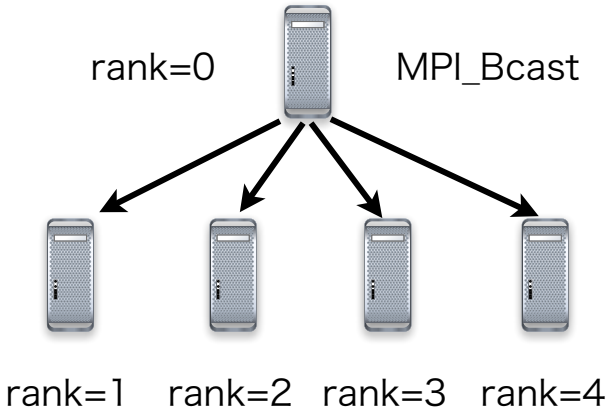
ブロードキャスト通信関数 MPI_Bcast

データ受信

```
MPI_Bcast(
  void *buf,                  受信データが格納されるアドレス
  int count,                  受信するデータの数
  MPI_Datatype datatype,      受信するデータの型 MPI_INT, MPI_DOUBLE
  int source,                 送信元プロセッサ rank
  MPI_Comm comm               送受信グループ MPI_COMM_WORLD
)
```

MPI_Bcast 例

rank0 から他のノードへ
整数値 int data を送る




```
if (rank == 0){
      printf("I am rank %d. Enter an integer: ", rank);
      scanf("%d", &data);
}

source = 0;
MPI_Bcast(&data, 1, MPI_INT, source, MPI_COMM_WORLD);

if(rank == 0)
   printf("Rank %d has broadcasted an integer %d\n", rank, data);
else
   printf("Rank %d has received an integer %d\n", rank, data);
```

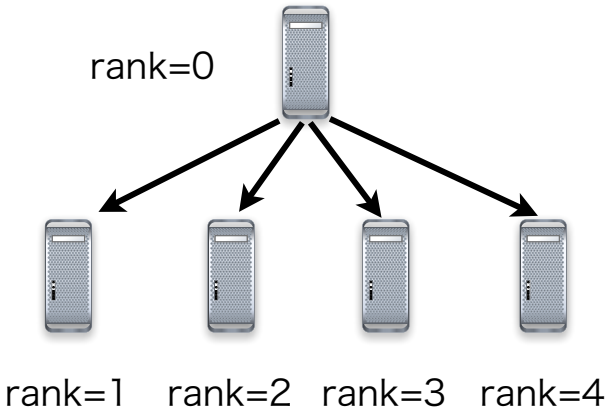
rank0 で整数値を入力

ノード 0 から他のノードへブロードキャスト

他のノードでは受信が行われることに注意

数列和を複数ノード n ≥ 3 で計算

- MPI_Bcast で数列和の上限をすべてのノードに送信する



rank 0 で上限を入力し、他のノードへ一斉送信する

```
int max;

if (rank == 0){
      printf("Enter max: ");
      scanf("%d", &max);
}

source = 0;
MPI_Bcast(&max, 1, MPI_INT, source, MPI_COMM_WORLD);
```

MPI_Reduce

集約関数 MPI_Reduce

データ受信

```
MPI_Reduce(
  void *buf_local,
  void *buf_result,
  int count,
  MPI_Datatype datatype,
  MPI_Op operation,
  int destination,
  MPI_Comm comm
)
```

受信するデータが格納されるアドレス

集約結果を格納するアドレス

受信するデータの数

受信するデータの型 MPI_INT, MPI_DOUBLE

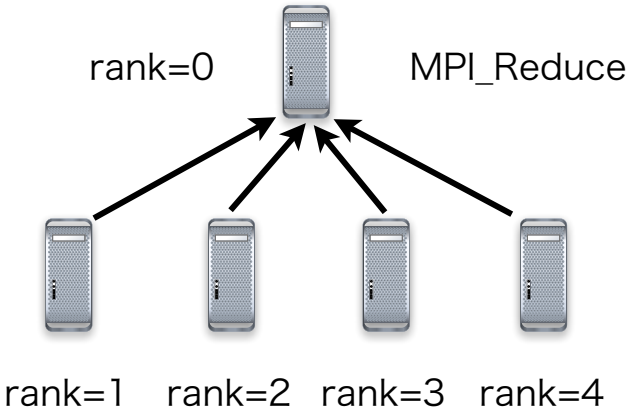
集約処理の指定

送信先プロセッサ rank

送受信グループ MPI_COMM_WORLD

Reduce operation

rank0 へ各ノードの結果を集約する例



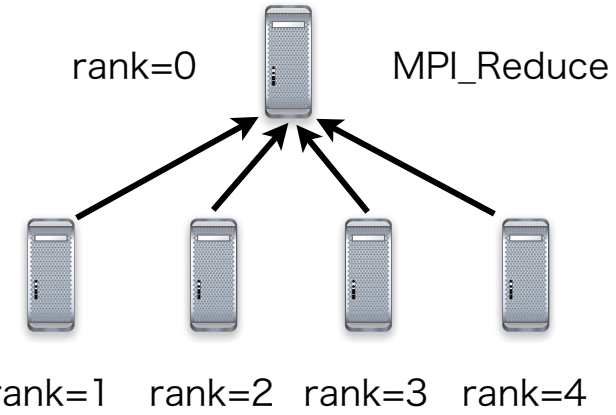
各ノードのデータをどう集約するかを集約 operation で指定

オペレータ	操作
MPI_SUM	和
MPI_PROD	積
MPI_MAX	最大値
MPI_MIN	最小値

MPI_SUM を指定すると、各ノード（自分自身を含む）のデータを足し合わせた結果が送信先の buf_result に格納される。

MPI_Reduce 例

ノード数 5 の場合



各ノードの変数 data に自分のランクを格納し、これをランク 0 の変数 data_reduced に集約処理 MPI_SUM （合計を計算）で格納するプログラム

```
data = rank;
printf("Rank %d has an integer %d\n", rank, data);

dest = 0;
MPI_Reduce(&data, &data_reduced, 1, MPI_INT, MPI_SUM, dest, MPI_COMM_WORLD);

if( rank == dest )
   printf("Reduced data is %d at rank %d\n", data_reduced, rank);
```

課題

- 下記積分をリーマン和として m 個のノードを用いて並列計算するプログラムを作れ。 m はプログラム実行時に $ mpirun -np m ... で指定する。なお、積分区間 [0, 1) の分割数はノード 0 で入力するとする。
- ノード数の増加（m=1, 2, 3, ...）に応じて処理時間がどの程度短縮するかを調べてレポートにまとめよ。

$$\int_0^1 \frac{4}{1+x^2} dx = \pi$$