KiSTi 한국과학기술정보연구원

MPI 프로그래밍 (II)

권오경

okkwon@kisti.re.kr

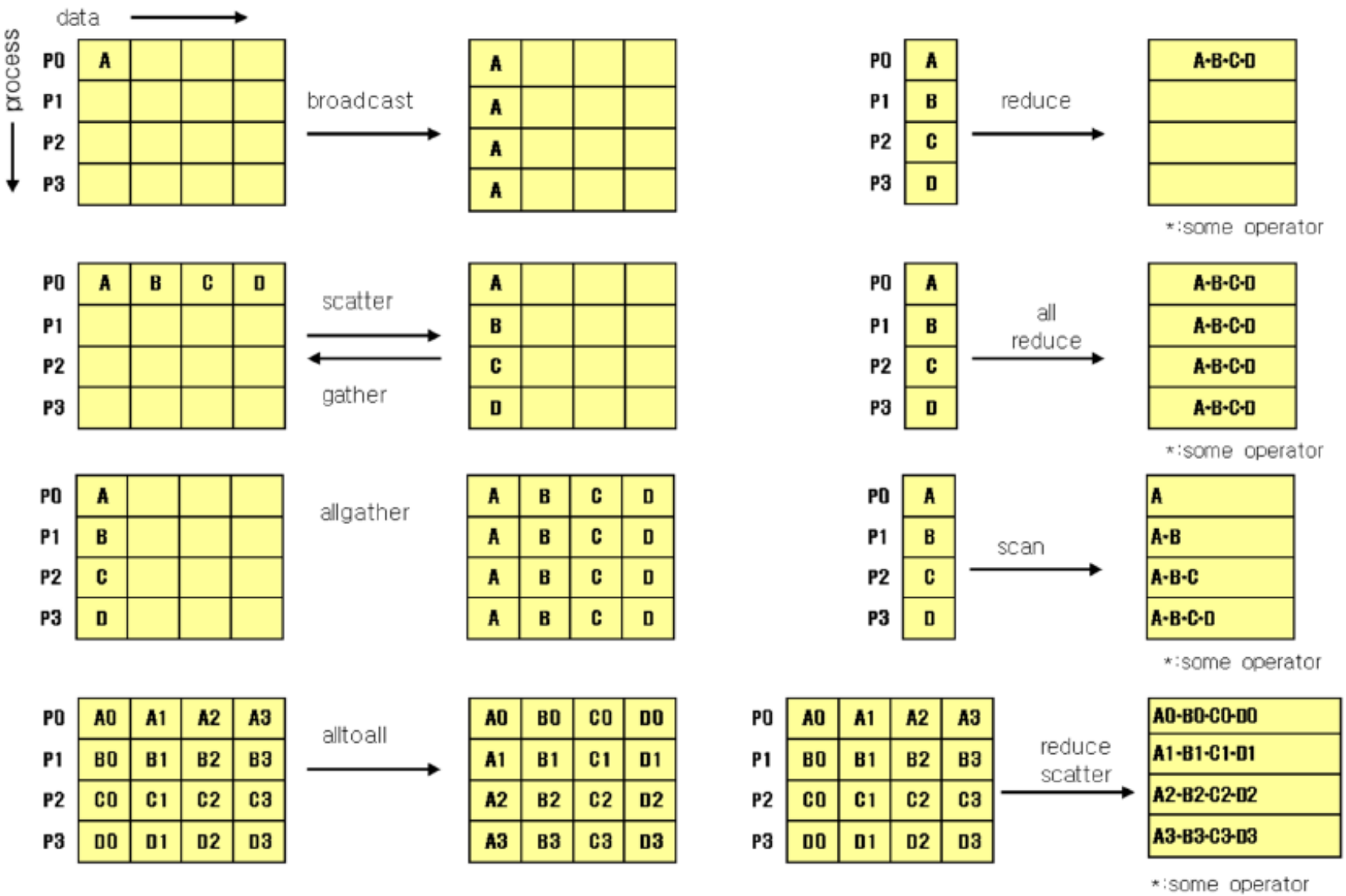
2018년 7월 17일

MPI: 집합 통신(Collective Communication) 함수

- **개요**
  - 2개 이상의 프로세스가 참여하는 그룹 통신
  - 점대점 통신을 기반으로 구현
  - 점대점 통신보다 효율적으로 구현되어 있어 성능이 좋음

분류	함수이름
수신과 송신시 데이터 Buffer(저장 장소)를 하나 사용	MPI_Bcast
수신과 송신기 데이터 Buffer를 별도로 사용	MPI_Gather, MPI_Gatherv, MPI_Allgather, MPI_Alltoall, MPI_Alltoallv MPI_Scatter, MPI_Scatterv, MPI_Allgatherv,
Reduction	MPI_Reduce, MPI_Allreduce
기타	MPI_Barrier

MPI: 집합 통신(Collective Communication) 함수

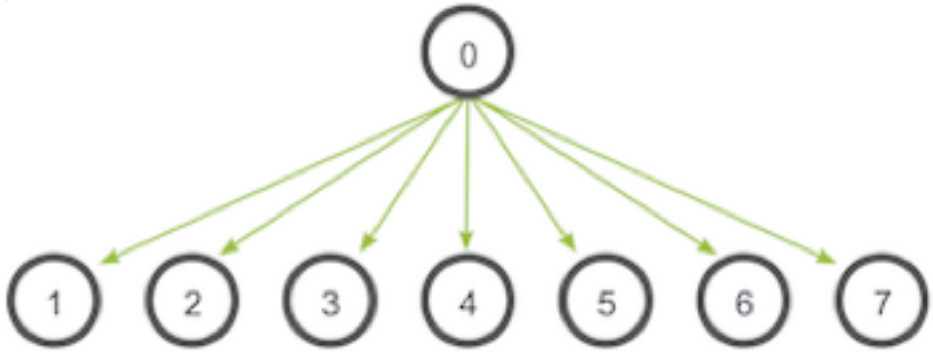


MPI: MPI_Bcast

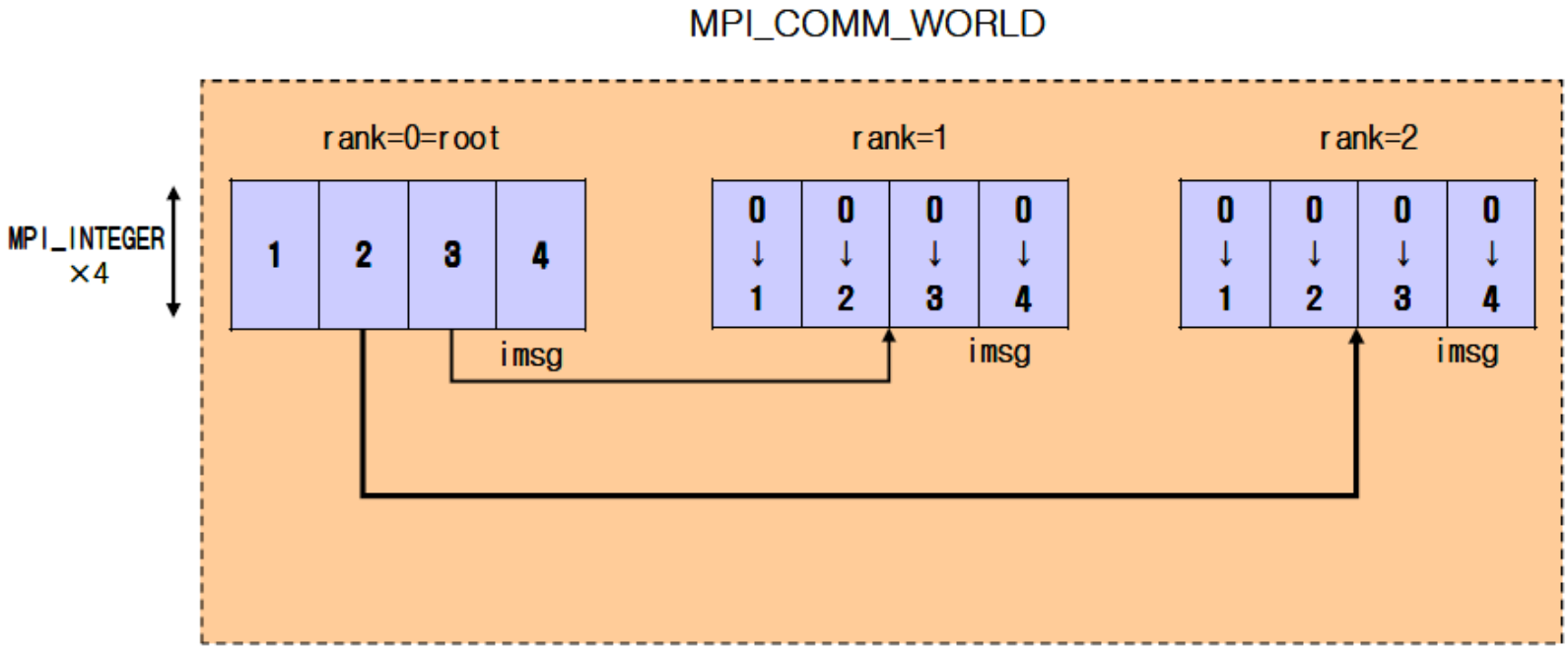
- **개요**
  - 하나의 프로세스에 있는 메시지를 다른 프로세스들에게 일괄적으로 보내기

C	int MPI_Bcast(void *buffer, int count, MPI_Datatype datatype, int root, MPI_Comm comm)

  - **buffer: 보낼 메시지의 데이터 저장소 주소**
  - **count: 보낼 메시지의 데이터 개수**
  - **datatype: 보낼 메시지의 데이터 타입**
  - **root: 보낼 메시지의 랭크 번호**
  - **comm: communicator 이름 (default: MPI_COMM_WORLD)**



MPI: MPI_Bcast example



MPI: MPI_Bcast example

- **코드**

```
#include <stdio.h>
#include <mpi.h>
int main(int argc, char *argv[])
{
  int i, nrank, nprocs, ROOT = 0;
  int buf[4] = { 0, 0, 0, 0 };
  MPI_Init(&argc, &argv);
  MPI_Comm_size(MPI_COMM_WORLD, &nprocs);
  MPI_Comm_rank(MPI_COMM_WORLD, &nrank);
  if (nrank == ROOT) {
     buf[0] = 1; buf[1] = 2; buf[2] = 3; buf[3] = 4;
  }
  printf("rank (%d) : Before : ", nrank);
  for (i=0; i<4; i++) printf(" %d", buf[i]);
  printf("\n");
  MPI_Bcast(buf, 4, MPI_INT, ROOT, MPI_COMM_WORLD);
  printf("rank (%d) : After : ", nrank);
  for (i=0; i<4; i++) printf(" %d", buf[i]);
  printf("\n");
  MPI_Finalize();
  return 0;
}
```

- **컴파일 & 실행**

```
$ mpicc -o bcast bcast.c
$ mpirun -np 3 -host host01,host02,host03 ./bcast
```

MPI: MPI_Bcast example (MPI_Send & MPI_Recv)

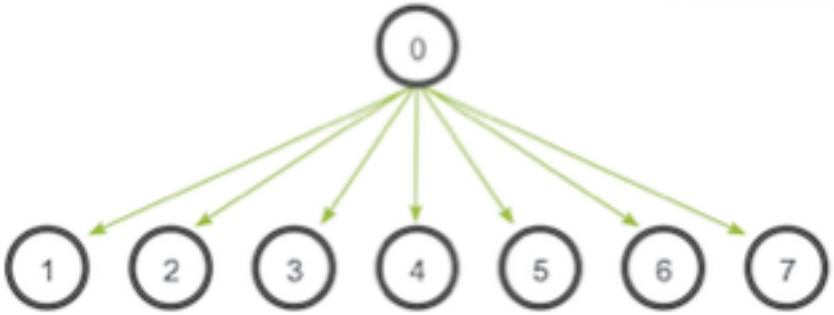
- **코드**

```
#include <stdio.h>
#include <mpi.h>
int main(int argc, char *argv[])
{
  int i, nrank, nprocs, ROOT = 0;
  int buf[4]= { 0, 0, 0, 0 };
  MPI_Init(&argc, &argv);
  MPI_Comm_size(MPI_COMM_WORLD, &nprocs);
  MPI_Comm_rank(MPI_COMM_WORLD, &nrank);
  if (nrank == ROOT) {
    buf[0] = 1; buf[1] = 2; buf[2] = 3; buf[3] = 4;
  }
  printf("rank (%d) : Before : ", nrank);
  for (i=0; i<4; i++) printf(" %d", buf[i]);
  printf("\n");
  if (nrank == ROOT) {
     for (i=0;i<nprocs;i++) {
       if (i!=ROOT) MPI_Send (&buf, 4, MPI_INT, i, ROOT, MPI_COMM_WORLD);
     }
  } else {
     MPI_Recv(&buf, 4, MPI_INT, ROOT, 0, MPI_COMM_WORLD, MPI_STATUS_IGNORE);
  }
  for (i=0; i<4; i++) printf(" %d", buf[i]);
  printf("\n");
  MPI_Finalize();
  return 0;
}
```

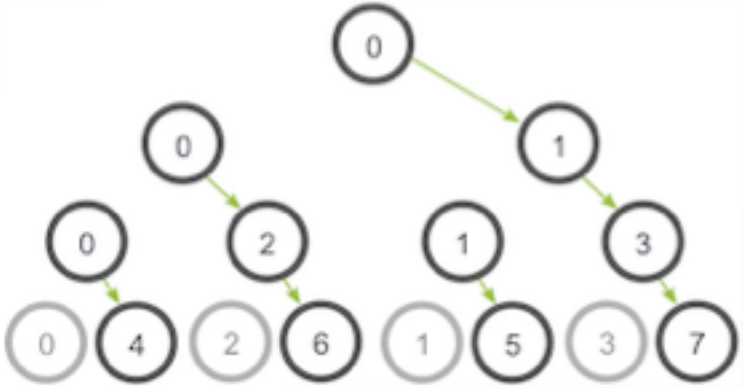
- **컴파일 & 실행**

```
$ mpicc -o bcast2 bcast2.c
$ mpirun -np 3 -host host01,host02,host03 ./bcast2
```

MPI: MPI_Bcast



MPI_Send & MPI_Recv



MPI_Bcast

- **컴파일 & 실행**

```
$ cd ./mpi_bcast_compare
$ make
$ mpirun -np 4 -host host01,host02,host03,host04 ./compare_bcast 1000 10000
Data size = 40000, Trials = 100000
Avg my_bcast time = 0.000013
Avg MPI_Bcast time = 0.000014
$ mpirun -np 4 -host host01,host02,host03,host04 ./compare_bcast 100000 100000
Data size = 400000, Trials = 100000
Avg my_bcast time = 0.000063
Avg MPI_Bcast time = 0.000092
```

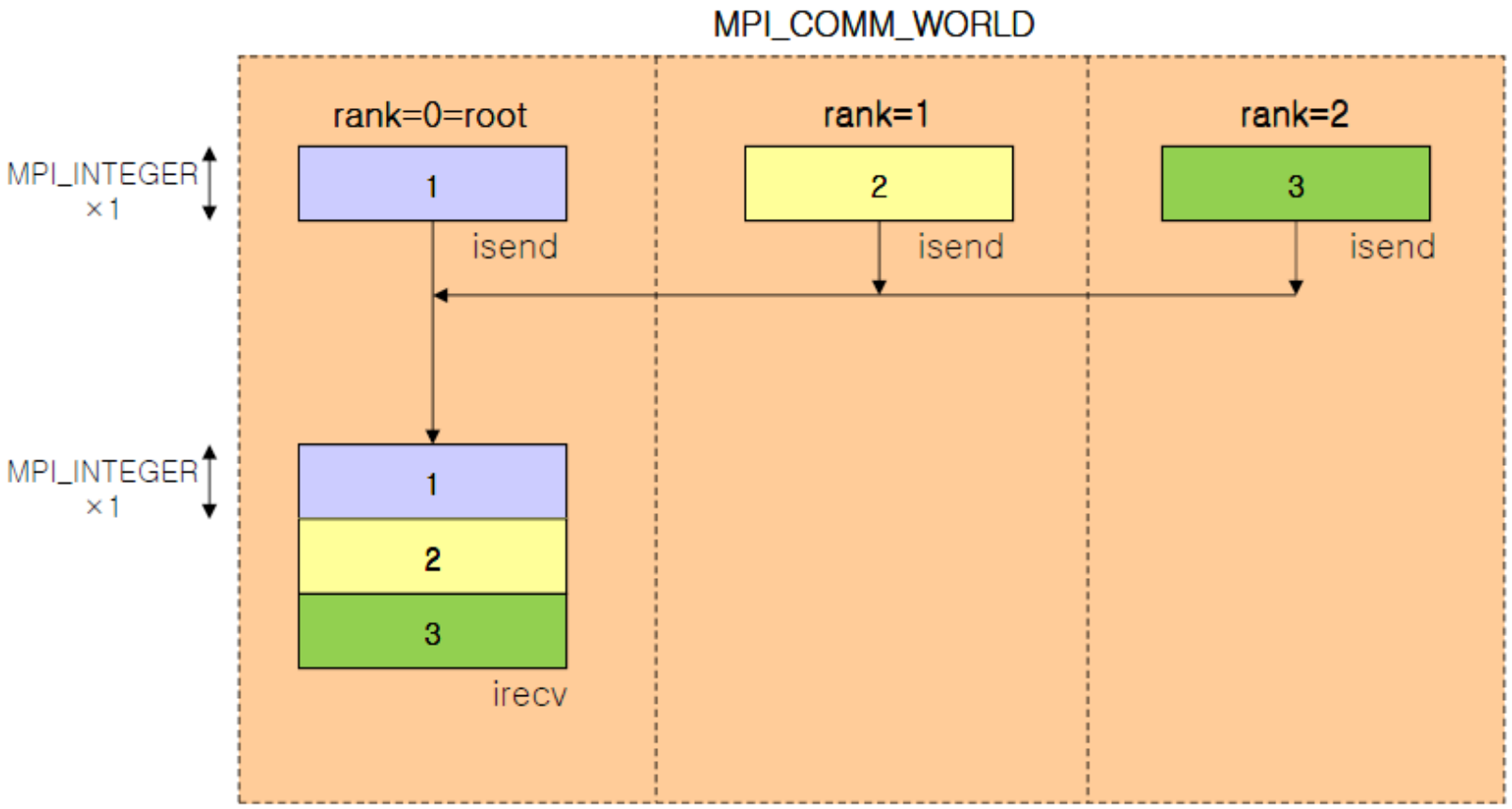
MPI: MPI_Gather

- **개요**
  - 다른 프로세스들에서 메시지를 한번에 받기

C	int MPI_Gather(void *sendbuf, int sendcount, MPI_Datatype sendtype, void *recvbuf, int recvcount, MPI_Datatype recvtype, int root, MPI_Comm comm)

    - **sendbuf: 보낼 메시지의 데이터 저장소 주소**
    - **sendcount: 보낼 메시지의 데이터 개수**
    - **sendtype: 보낼 메시지의 데이터 타입**
    - **recvbuf: 받을 메시지의 데이터 저장소 주소**
    - **recvcount: 받을 메시지의 데이터 개수**
    - **recvtype: 보낼 메시지의 데이터 타입**
    - **root: 보낼 메시지의 랭크 번호**
    - **comm: communicator 이름 (default: MPI_COMM_WORLD)**
  - **주의**
    - **sendbuf와 recvbuf는 같은 데이터 저장소를 사용하지 못함**

MPI: MPI_Gather example



MPI: MPI_Gather example

- **코드**

```
#include <stdio.h>
#include <mpi.h>
int main(int argc, char *argv[])
{
  int i, nprocs, nrank;
  int isend, irecv[4];
  MPI_Init(&argc, &argv);
  MPI_Comm_size(MPI_COMM_WORLD, &nprocs);
  MPI_Comm_rank(MPI_COMM_WORLD, &nrank);
  isend = nrank + 1;
  printf("rank (%d) : isend = %d ", nrank, isend);
  MPI_Gather(&isend, 1, MPI_INT, irecv, 1, MPI_INT, 0,
    MPI_COMM_WORLD);
  if (nrank == 0) {
    printf("\n");
    for (i=0; i<3; i++)
    printf("rank (%d) : irecv[%d] = %d\n",
    nrank, i, irecv[i]);
  }
  printf("\n");
  MPI_Finalize();
  return 0;
}
```

- **컴파일 & 실행**

```
$ mpicc -o gather gather.c
$ mpirun -np 3 -host host01,host02,host03 ./gather
```

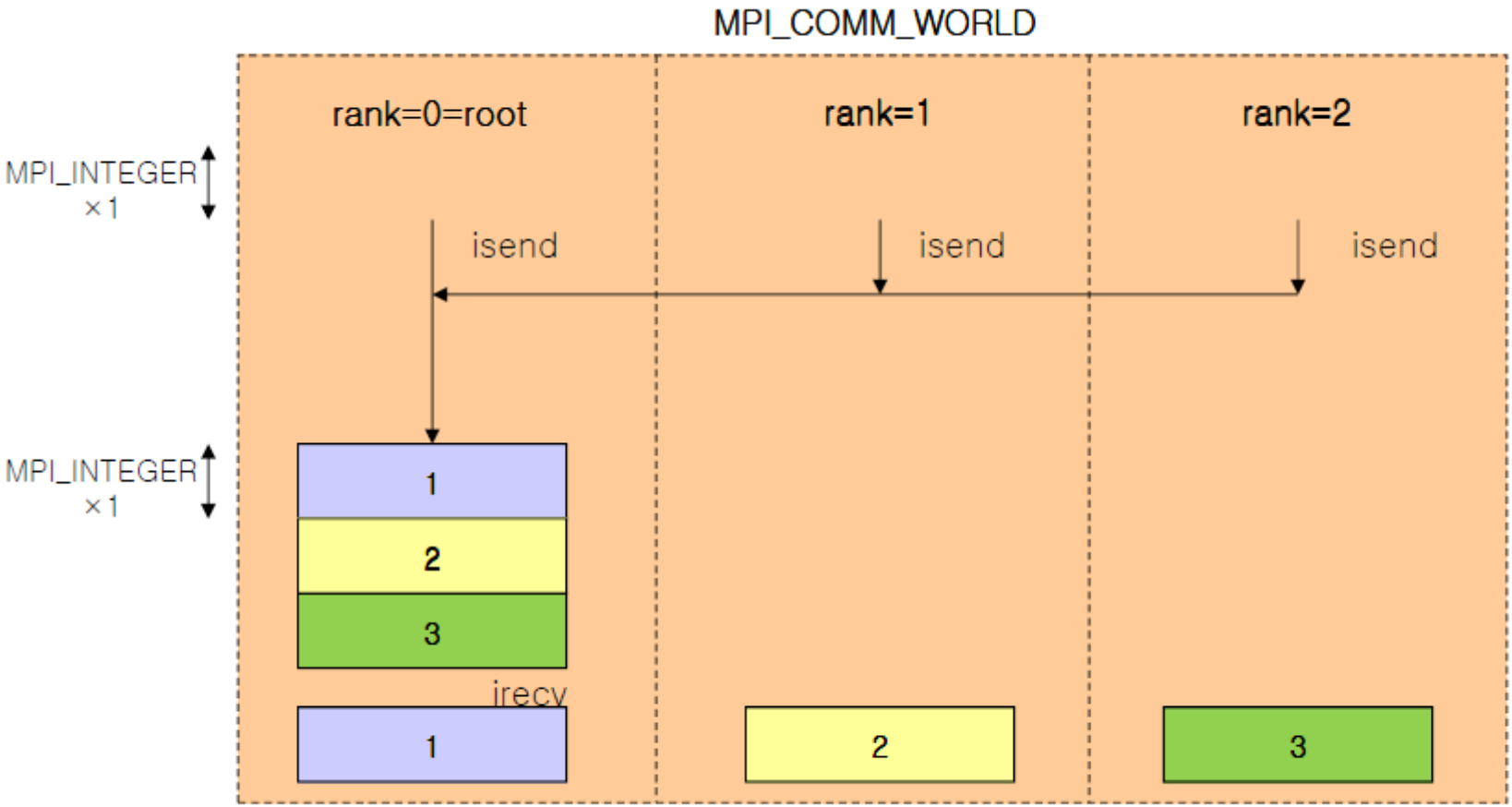
MPI: MPI_Scatter

- **개요**
  - 다른 프로세스들에서 메시지를 뿌리기

C	int MPI_Scatter(void *sendbuf, int sendcount, MPI_Datatype sendtype, void *recvbuf, int recvcount, MPI_Datatype recvtype, int root, MPI_Comm comm)

    - **sendbuf: 보낼 메시지의 데이터 저장소 주소**
    - **sendcount: 보낼 메시지의 데이터 개수**
    - **sendtype: 보낼 메시지의 데이터 타입**
    - **recvbuf: 받을 메시지의 데이터 저장소 주소**
    - **recvcount: 받을 메시지의 데이터 개수**
    - **recvtype: 보낼 메시지의 데이터 타입**
    - **root: 보낼 메시지의 랭크 번호**
    - **comm: communicator 이름 (default: MPI_COMM_WORLD)**
  - **주의**
    - **sendbuf와 recvbuf는 같은 데이터 저장소를 사용하지 못함**

MPI: MPI_Scatter example



MPI: MPI_Reduce

- **개요**
  - 모든 프로세스의 메시지로부터 하나의 값을 계산해서 모으기

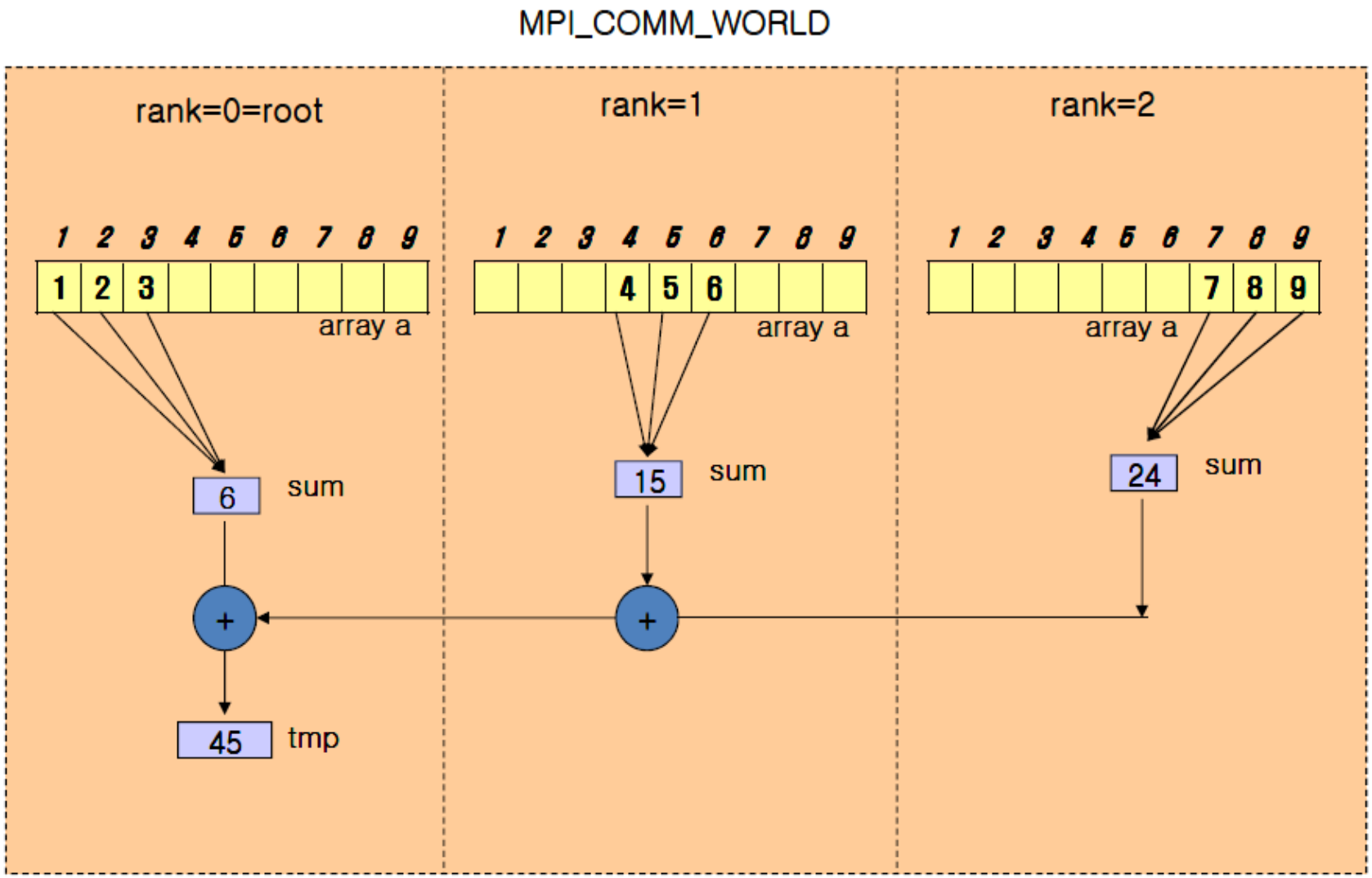
C	int MPI_Reduce(void *sendbuf, void *recvbuf, int count, MPI_Datatype datatype, MPI_Op op, int root, MPI_Comm comm)

  - **sendbuf: 해당 전송 메시지의 데이터 저장소 주소**
  - **recvbuf: 계산 값을 담을 메시지의 데이터 저장소 주소**
  - **count: 전송 메시지의 데이터 개수**
  - **datatype: 전송 메시지의 데이터 타입**
  - **op: 계산 형태 (예, 더하기, 빼기, 곱하기 등)**
  - **root: 계산 값을 담을 메시지를 가진 랭크 번호**
  - **comm: communicator 이름 (default: MPI_COMM_WORLD)**

MPI: MPI_Reduce: operation & data type

Operation	Data Type (C)
MPI_SUM(sum), MPI_PROD(product) MPI_MAX(maximum), MPI_MIN(minimum)	MPI_INT, MPI_LONG, MPI_SHORT, MPI_UNSIGNED_SHORT, MPI_UNSIGNED MPI_UNSIGNED_LONG, MPI_FLOAT, MPI_DOUBLE, MPI_LONG_DOUBLE
MPI_MAXLOC(max value and location), MPI_MINLOC(min value and location)	MPI_FLOAT_INT, MPI_DOUBLE_INT, MPI_LONG_INT, MPI_2INT, MPI_SHORT_INT, MPI_LONG_DOUBLE_INT
MPI_LAND(logical AND), MPI_LOR(logical OR), MPI_LXOR(logical XOR)	MPI_INT, MPI_LONG, MPI_SHORT, MPI_UNSIGNED_SHORT, MPI_UNSIGNED MPI_UNSIGNED_LONG
MPI_BAND(bitwise AND), MPI_BOR(bitwise OR), MPI_BXOR(bitwise XOR)	MPI_INT, MPI_LONG, MPI_SHORT, MPI_UNSIGNED_SHORT, MPI_UNSIGNED MPI_UNSIGNED_LONG, MPI_BYTE

MPI: MPI_Reduce example



MPI: MPI_Reduce example

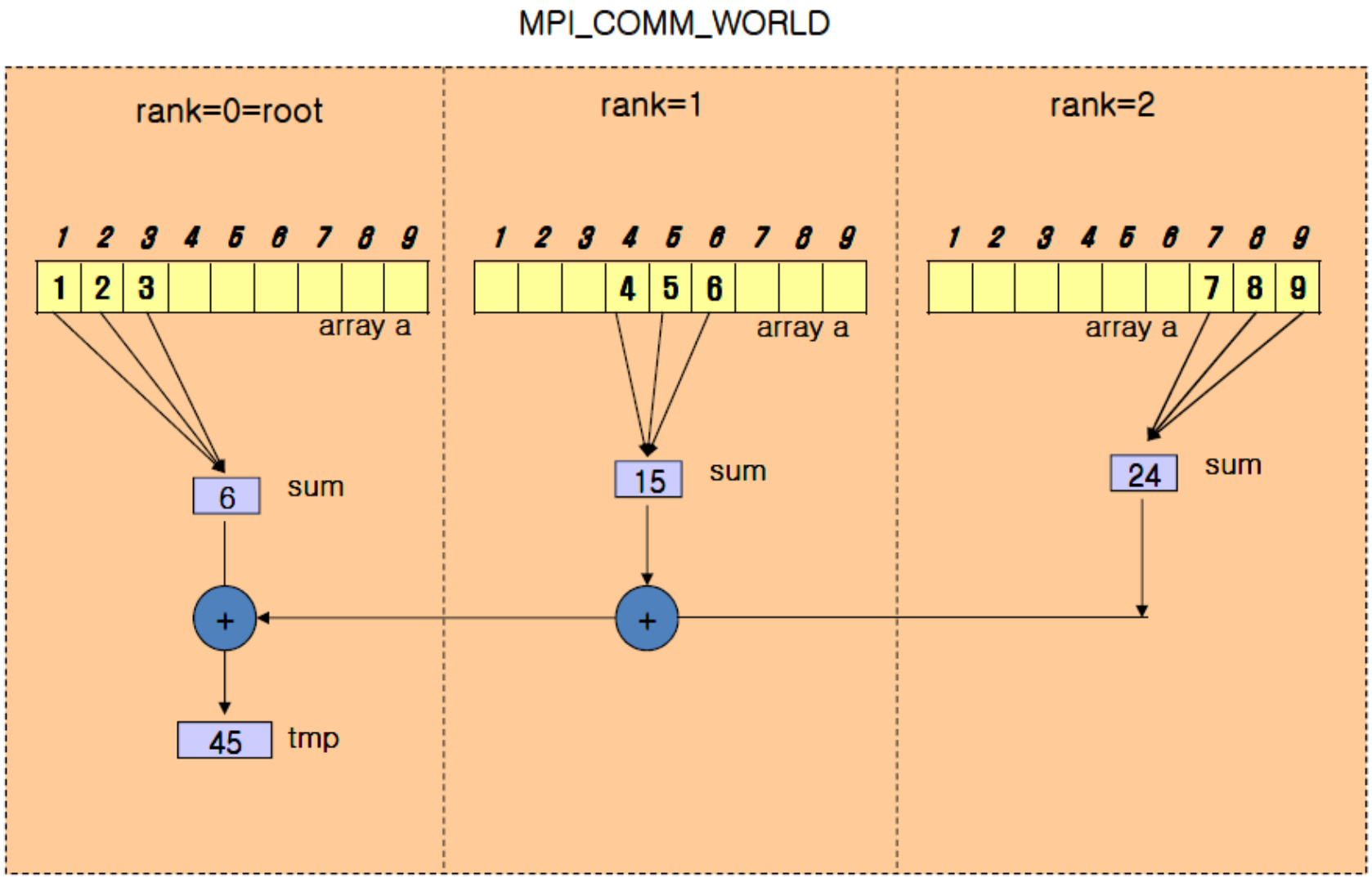
- **코드**

```
#include <stdio.h>
#include <mpi.h>
int main(int argc, char *argv[])
{
  int i, nrank, start, end, ROOT = 0;
  double a[9], sum, tmp;
  MPI_Init(&argc, &argv);
  MPI_Comm_rank(MPI_COMM_WORLD, &nrank);
  start = nrank * 3;
  end = start + 2;
  for (i=start; i<end+1; i++) {
    a[i] = i + 1;
    if (i == start) printf("rank (%d) ", nrank);
    printf("a[%d] = %.2f ", i, a[i]);
  }
  sum = 0.0;
  for (i=start; i<end+1; i++) sum = sum + a[i];
  MPI_Reduce(&sum, &tmp, 1, MPI_DOUBLE, MPI_SUM,ROOT,
     MPI_COMM_WORLD);
  sum = tmp;
  if (nrank == ROOT) printf("\nrank(%d):sum= %.2f.\n",
     nrank, sum);
  printf("\n");
  MPI_Finalize();
  return 0;
}
```

- **컴파일 & 실행**

```
$ mpicc -o reduce reduce.c
$ mpirun -np 3 -host host01,host02,host03 ./reduce
```

MPI: MPI_Reduce: Array



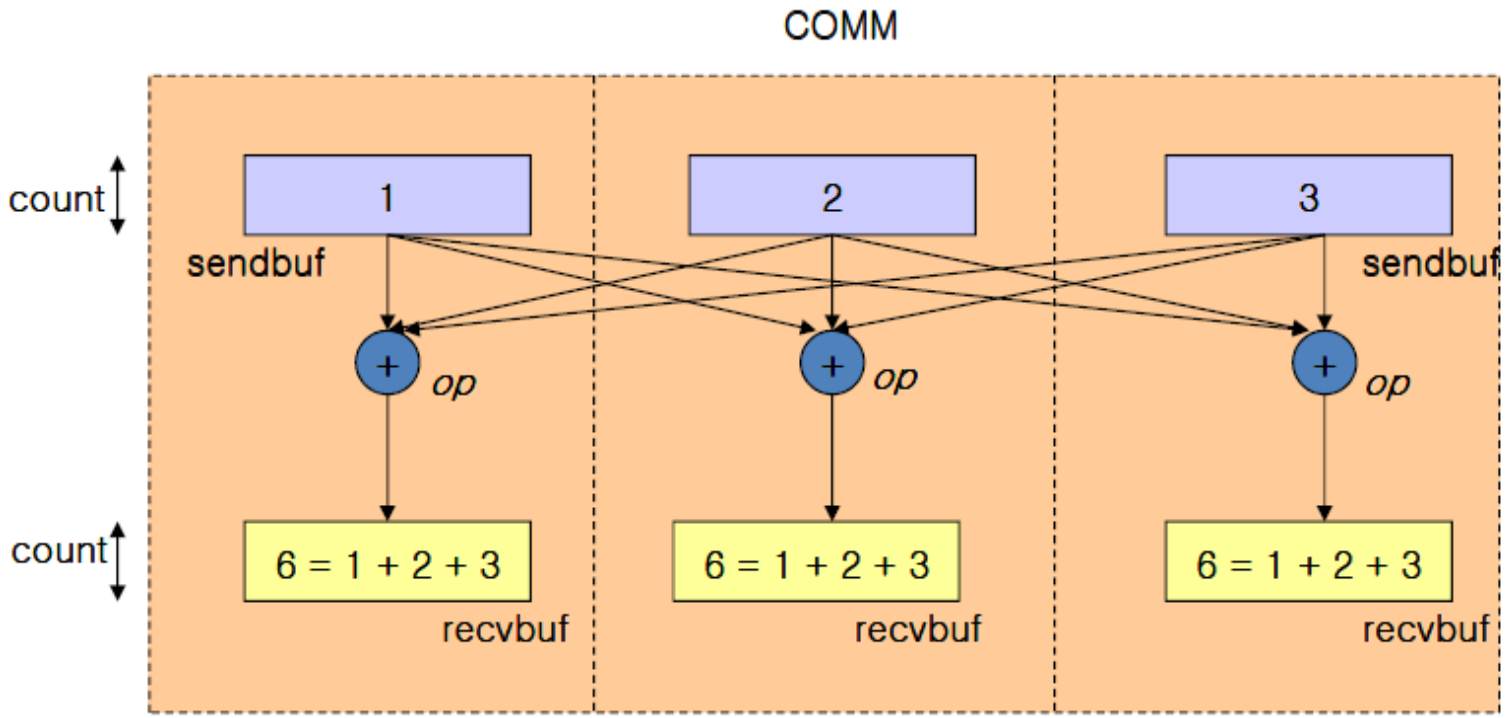
MPI: MPI_Allreduce

- **개요**
  - 모든 프로세스의 메시지로부터 하나의 값을 계산한 다음 모든 프로세스에게 값을 돌려줌

C	`int MPI_Allreduce(void *sendbuf, void *recvbuf, int count, MPI_Datatype datatype, MPI_Op op, MPI_Comm comm)`

  - **sendbuf: 해당 전송 메시지의 데이터 저장소 주소**
  - **recvbuf: 계산 값을 담을 메시지의 데이터 저장소 주소**
  - **count: 전송 메시지의 데이터 개수**
  - **datatype: 전송 메시지의 데이터 타입**
  - **op: 계산 형태 (예, 더하기, 빼기, 곱하기 등)**
  - **comm: communicator 이름 (default: MPI_COMM_WORLD)**

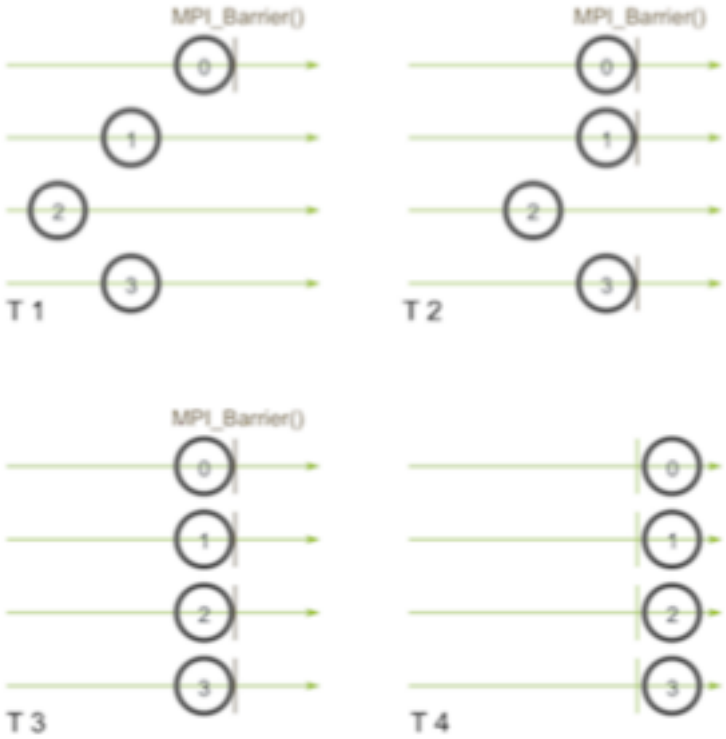
MPI: MPI_Allreduce example



MPI: MPI_Barrier

- **개요**
  - 모든 프로세스가 현 시점까지 도달할 때까지 기다림
  - 병렬프로그램 실행중 일부분이 정확하게 같이 실행되길 원할때 사용

C	`int MPI_Barrier(MPI_Comm comm)`



MPI: MPI_Barrier example

- **코드**

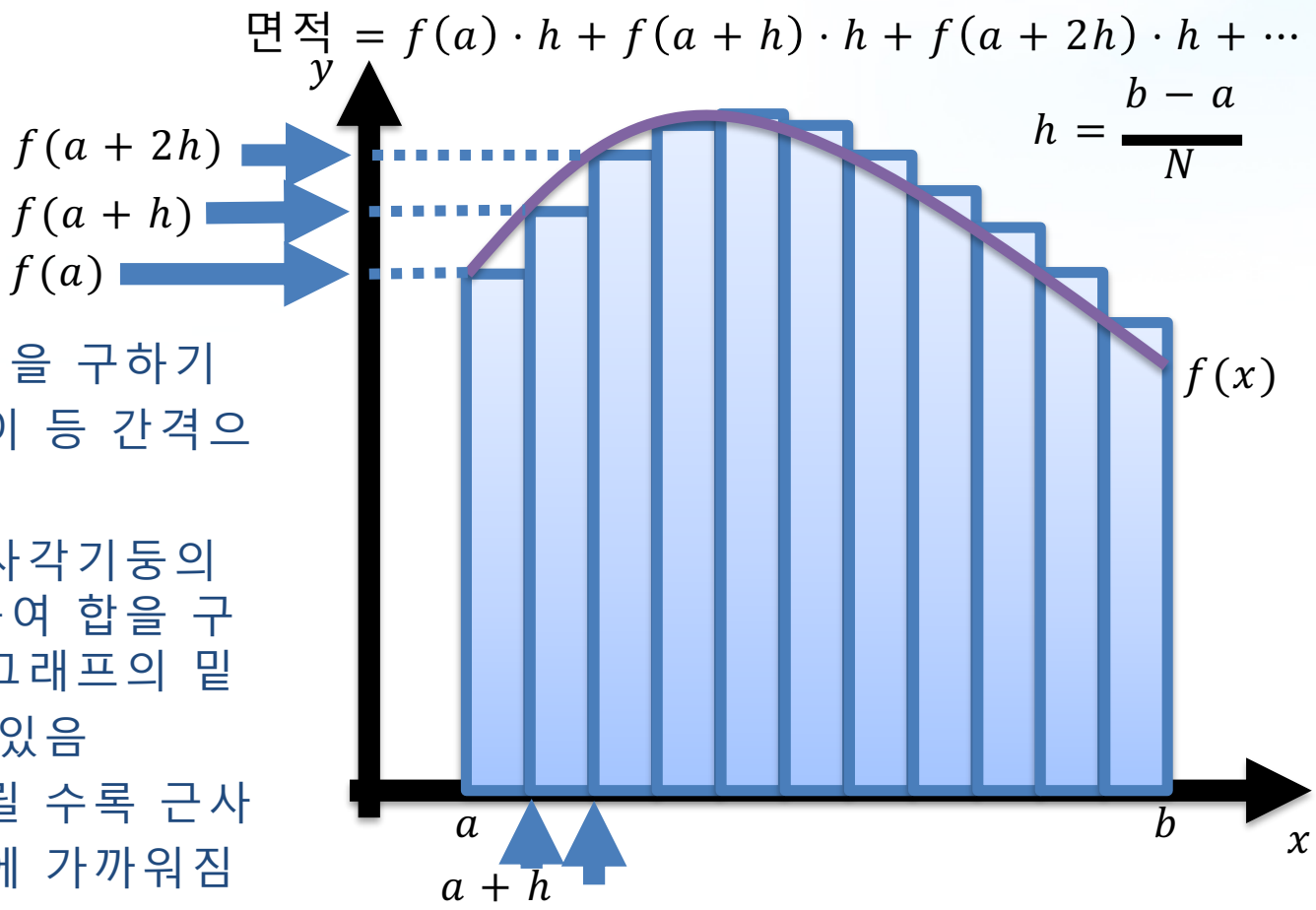
```
#include <stdio.h>
#include <mpi.h>
int main (int argc, char *argv[])
{
  int nRank, nProcs;
  MPI_Init(&argc, &argv);
  MPI_Comm_rank(MPI_COMM_WORLD, &nRank);
  MPI_Comm_size(MPI_COMM_WORLD, &nProcs);
  if (nRank == 0) {
     printf("############\n");
  }
  MPI_Barrier(MPI_COMM_WORLD);
  printf("nRank(%d) : Hello World.\n", nRank);
  MPI_Finalize();
  return 0;
}
```

- **컴파일 & 실행**

```
$ mpicc -o barrier barrier.c
$ mpirun -np 4 -host host01, host02, host03, host04 ./barrier
```

Practice: 그래프 밑면적 구하기

점대점통신으로 구현한 코드를 집합통신을 이용해 구현하기



- 그래프의 밑면적을 구하기 위해 다음과 같이 등 간격으로 분할
- 분할된 면적을 사각기둥의 면적으로 근사하여 합을 구하면 근사적인 그래프의 밑면적을 구할 수 있음
- 분할 개수를 늘릴 수록 근사값이 실제 면적에 가까워짐

MPI: 그래프 밑면적 구하기

- **코드**

```
…
    for (i = rank; i < N; i += size) {
        my_sum += f(i*step);
    }
    if (0 != rank) {
      MPI_Send(&my_sum, 1, MPI_DOUBLE, 0, 0, MPI_COMM_WORLD);
    } else {
      sum += my_sum;
      for (i = 1; i < size; i++) {
         MPI_Recv(&my_sum, 1, MPI_DOUBLE, i, 0, MPI_COMM_WORLD, MPI_STATUS_IGNORE);
         sum += my_sum;
      }
      sum *= step;
    }
    end = MPI_Wtime();

    if (0 == rank) {
      sum = tmp*step;
      printf("result=%e\n", sum);
      printf("error=%e\n", fabs(sum - solution(A, B)));
      printf("elapsed time=%f\n", end - start);
    }
…
```

Q&A