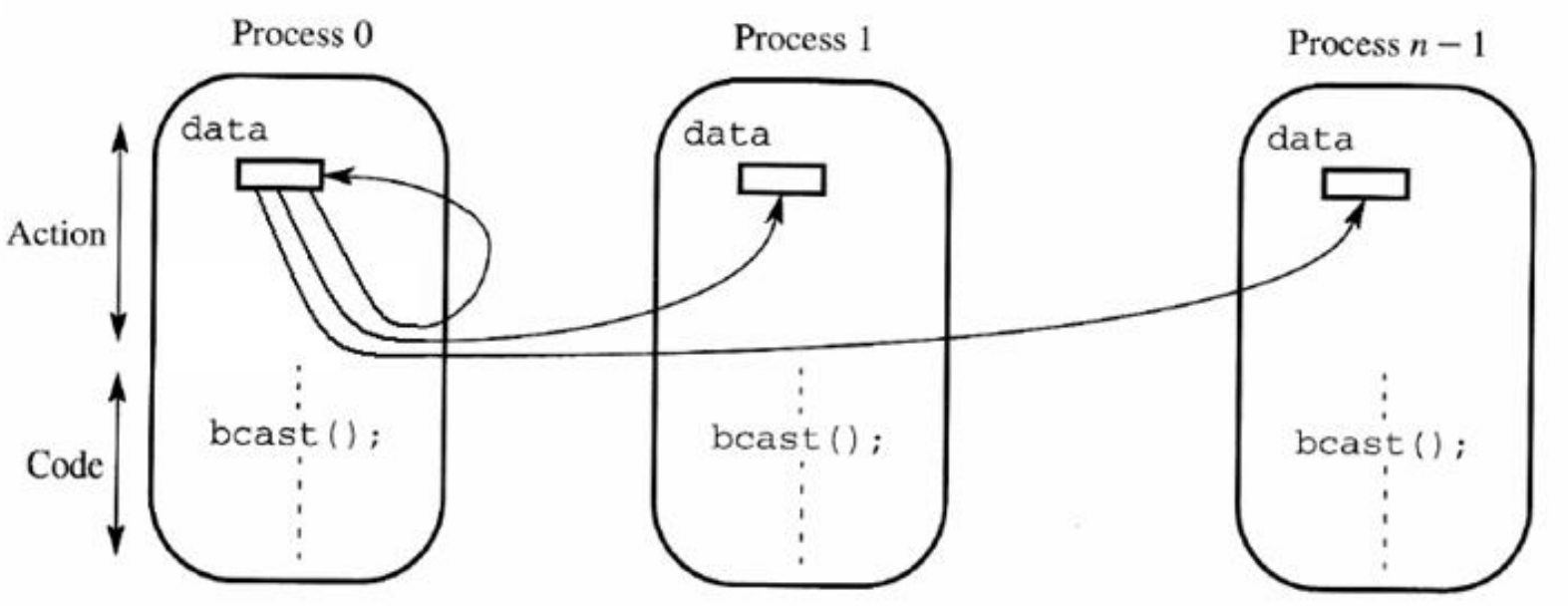
Lygiagretusis programavimas naudojant MPI (2)

MPI
Collective komunikavimas

MPI Broadcast

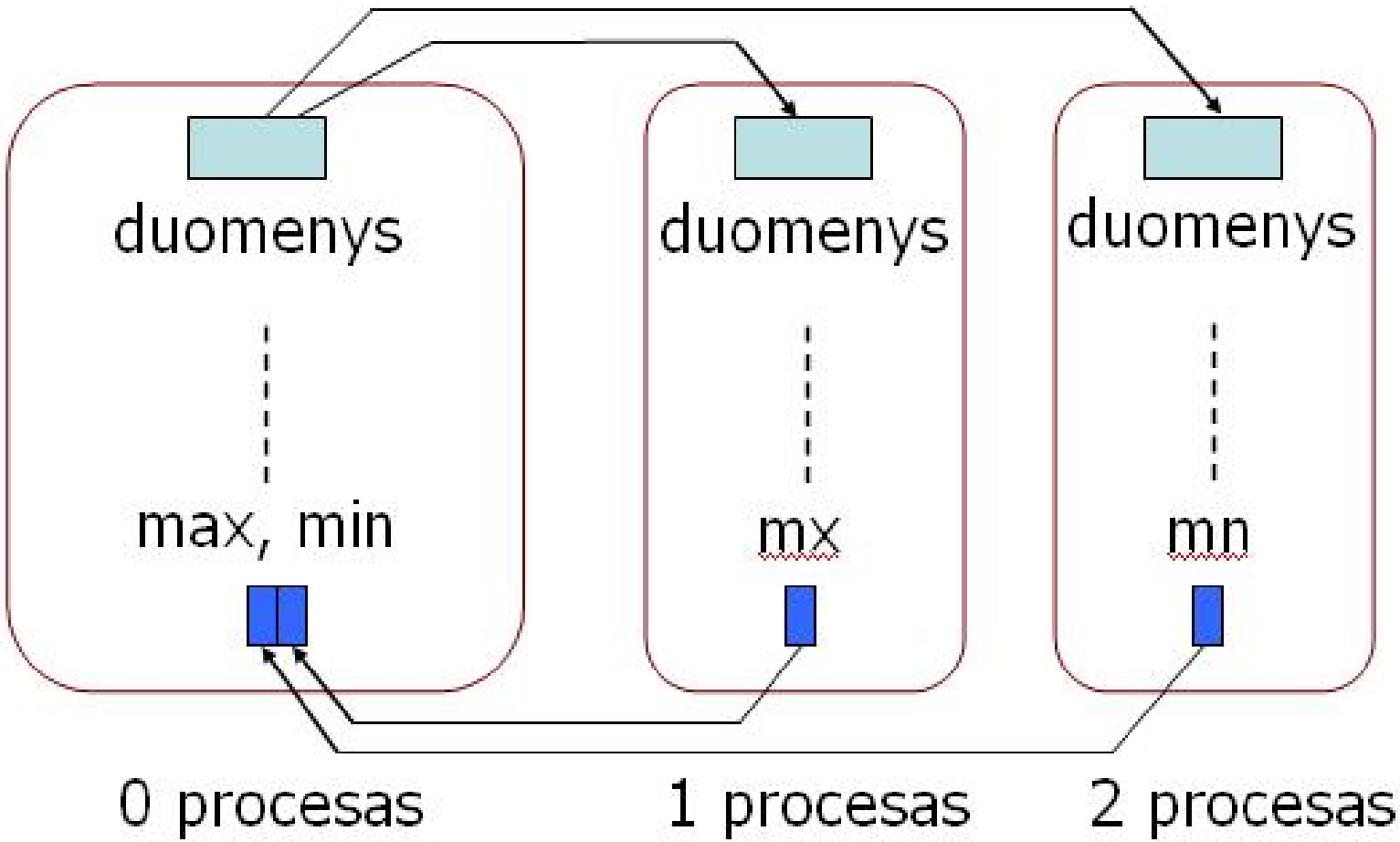


Siuntimas visiems

void Comm::**Bcast**(const void* buf, int count,
const Datatype& datatype, int root) const

- root – siunčiantis procesas;
- kiti parametrai: žiūrėti Send.

Bcast pavyzdys: min, max radimas, size=3



Bcast pavyzdys (a)

```
// MPI Bcast iliustracija (trys procesai).
// Randa masyvo min ir max.
// 0 proc.  - pagr., 1, 2 - dirbantys proc.
...
if (rank == 0) {
  IvestiDuomenis("Duom.txt", Masyvas, m);
}
MPI::COMM_WORLD.Bcast(&m, 1, MPI::INT, 0);
MPI::COMM_WORLD.Bcast(Masyvas, m, MPI::INT, 0);
```

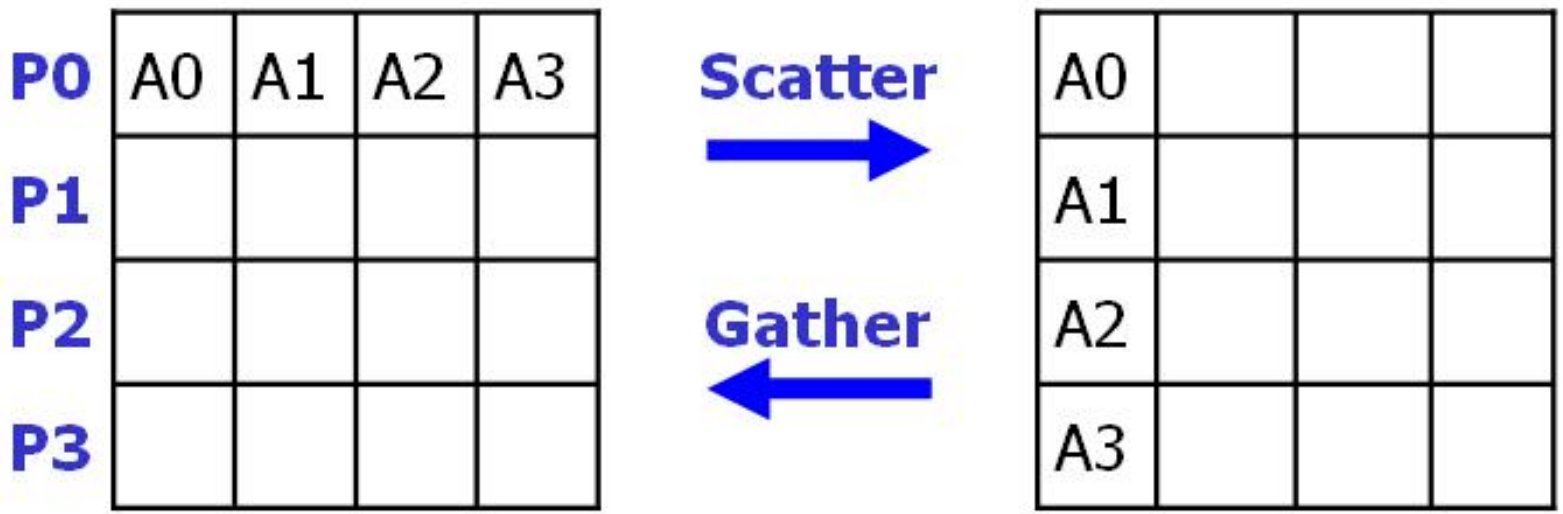
Bcast pavyzdys (b)

```
switch (rank) {
  case 0:  {
    int min = 0, max = 0;
    MPI::COMM_WORLD.Recv(&max, 1, MPI::INT, 1, 1);
    MPI::COMM_WORLD.Recv(&min, 1, MPI::INT, 2, 1);
    break;
  } case 1:  {
    int mx = Max(Masyvas, m);
    MPI::COMM_WORLD.Send(&mx, 1, MPI::INT, 0, 1);
    break;
  } case 2:  {
    int mn = Min(Masyvas, m);
    MPI::COMM_WORLD.Send(&mn, 1, MPI::INT, 0, 1);
} }
```

Scatter ir Gather (a)

- **Scatter** (išsklaidyti): duomenis, saugomus viename procese, persiunčia dalimis visiems komunikatoriaus procesams.
- **Gather** (surinkti): duomenis, saugomus skirtinguose procesuose, surenka viename procese į bendrą duomenų rinkinį.

Scatter ir Gather (b)



Duomenų išskaidymas

void Comm::**Scatter**(const void* sendbuf, int sendcount,
const Datatype& sendtype, void* recvbuf,
int recvcount, const Datatype& recvtype,
int root) const

- root – siunčiantis procesas.

Scatter panaudojimo pavyzdys

- Programos fragmentas:

```
char z[] = "123456"; char A[500] = "******";
if(rank == 0)
  strcpy(A, "vienasdu....trys..keturi");
MPI::COMM_WORLD.Scatter(A, 6, MPI::CHAR,
                        z, 6, MPI::CHAR, 0);
```

- Rezultatas:

```
0 pr:z = "vienas", A = "vienasdu....trys..keturi"
2 pr:z = "trys..", A = "******"
1 pr:z = "du....", A = "******"
3 pr:z = "keturi", A = "******"
```

Duomenų surinkimas

void Comm::**Gather**(const void* sendbuf, int sendcount,
const Datatype& sendtype, void* recvbuf,
int recvcount, const Datatype& recvtype,
int root) const

- root – surenkantis procesas.

Gather panaudojimo pavyzdys

- Programos fragmentas:

```
char z[] = "******"; char A[500] = "123456";
if(rank == 0) strcpy(z, "vienas");
if(rank == 1) strcpy(z, "du....");
if(rank == 2) strcpy(z, "trys..");
if(rank == 3) strcpy(z, "keturi");
MPI::COMM_WORLD.Gather(z, 6, MPI::CHAR,
                       A, 6, MPI::CHAR, 0);
```

- Rezultatas:

```
1 pr:z = "du....", A = "123456"
2 pr:z = "trys..", A = "123456"
0 pr:z = "vienas", A = "vienasdu....trys..keturi"
3 pr:z = "keturi", A = "123456"
```

MPI
Duomenų perdavimas

Pagrindinės problemos

- Siųsti galima tik MPI tipų duomenis.
- Struktūros tipas – ne MPI tipas.

Struktūros kintamasis

```
struct TStruct {
  char   d1[20];
  double d2;
  int    d3;
};
  // p - paskirtos atminties adresas
TStruct *p = new TStruct;
```

Struktūros siuntimas

```
int a = sizeof(TStruct); // baitų sk.
MPI::COMM_WORLD.Send(&a, 1, MPI::INT, 1, 1);
 // konvertuojamas adresas (char - 1 baitas)
char *r = (char*)p;
 // siunčiama simbolių (baitų) seka
MPI::COMM_WORLD.Send(r, a, MPI::CHAR, 1, 2);
```

Struktūros priėmimas

```
int c;
MPI::COMM_WORLD.Recv(&c, 1, MPI::INT, 0, 1);
 // rezervuojama atmintis simbolių (baitų) sekai
char *r = new char[c];
 // priimama simbolių (baitų) seka
MPI::COMM_WORLD.Recv(r, c, MPI::CHAR, 0, 2);
 // konvertuojamas adresas (p1 - struktūros adr.)
TStruct *p1 = (TStruct *)r;
```

Struktūrų masyvas

```
struct TStruct {
  char   d1[20];
  double d2;
  int    d3;
};
int n = ...;
  // p - paskirtos atminties adresas
TStruct *p = new TStruct[n];
```

Struktūrų masyvo siuntimas

```
int a = sizeof(TStruct) * n; // baitų sk.
MPI::COMM_WORLD.Send(&a, 1, MPI::INT, 1, 1);
 // konvertuojamas adresas (char - 1 baitas)
char *r = (char*)p;
 // siunčiama simbolių (baitų) seka
MPI::COMM_WORLD.Send(r, a, MPI::CHAR, 1, 2);
```

Struktūrų masyvo priėmimas

```
int c;
MPI::COMM_WORLD.Recv(&c, 1, MPI::INT, 0, 1);
 // rezervuojama atmintis simbolių (baitų) sekai
char *r = new char[c];
 // priimama simbolių (baitų) seka
MPI::COMM_WORLD.Recv(r, c, MPI::CHAR, 0, 2);
 // konvertuojamas adresas (p1 - str.masyvo adr.)
TStruct *p1 = (TStruct *)r;
int n = c / sizeof(TStruct);
```

???

- Aprašytas duomenų perdavimo būdas tinka C++ versijose, kuriose **char** tipas užima 1 baitą.
- Ar galima šį būdą taikyti ten, kur **char** užima > už 1 baitą?
- Ar galima pritaikyti kokį nors kitą duomenų, kurių tipas nesutampa su MPI tipais, perdavimo būdą?

Asinchroninis pranešimo perdavimas

- Siuntimas:
  `Comm::Isend()`
- Priėmimas:
  `Comm::Irecv()`

`MPI_Wtime()` (programos laiko skaičiavimas)

- Returns a double giving time in seconds from a fixed time in the past.
- To time a program, record MPI_Wtime() in a variable at start, then again at finish, difference is elapsed time.
- Pavyzdys:

```
startime = MPI_Wtime();
...  // Programos dalis, kurios vykdymo
...  // laikas yra skaičiuojamas
stoptime = MPI_Wtime();
time = stoptime - starttime;
```

Klausimai pakartojimui

1. Kiek procesų dalyvauja MPI Broadcast siuntime?
2. Kokiose situacijose naudingas MPI Broadcast siuntimas?
3. Kuriame procese turi būti rašomas kreipinys į Bcast siuntimo funkciją?
4. Kada naudingas Scatter siuntimas?
5. Kada naudingas Gather siuntimas?
6. Kokių tipų duomenys gali būti siunčiami?
7. Kokiu būdu galima siųsti ne MPI tipų duomenis?