

Grupne operacije-podsetnik

```
int MPI_Reduce (void* send_buffer,void* recv_buffer,  
int count, MPI_Datatype datatype, MPI_Op operation,  
int rank, MPI_Comm comm )
```

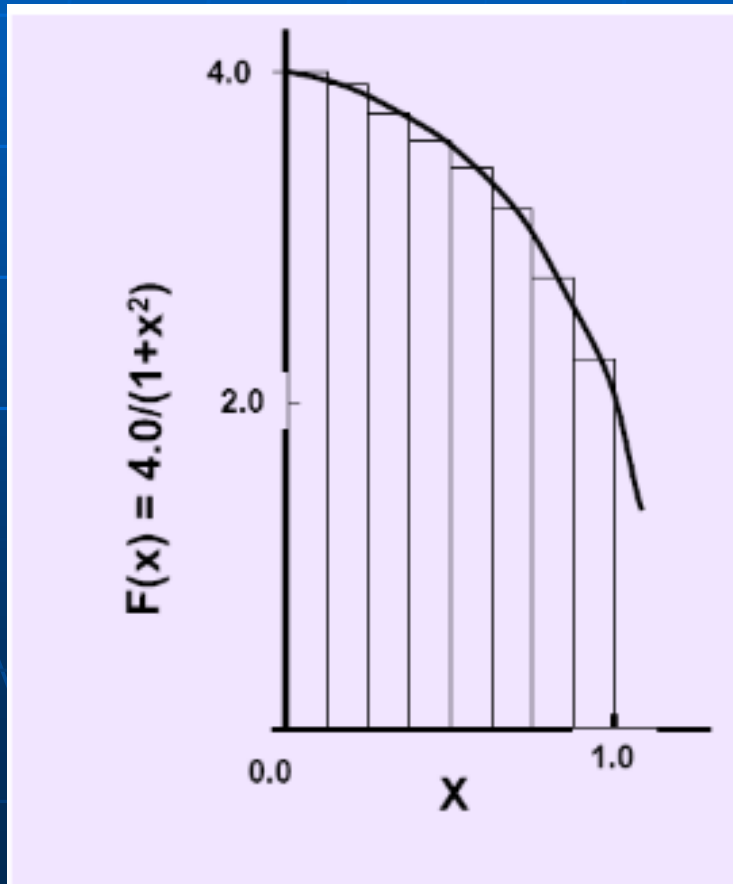
```
int MPI_Bcast ( void* buffer, int count, MPI_Datatype  
datatype, int rank, MPI_Comm comm )
```

```
int MPI_Scatter ( void* send_buffer, int send_count,  
MPI_datatype send_type, void* recv_buffer, int  
recv_count, MPI_Datatype recv_type, int rank,  
MPI_Comm comm )
```

```
int MPI_Gather ( void* send_buffer, int send_count,  
MPI_datatype send_type, void* recv_buffer, int  
recv_count, MPI_Datatype recv_type, int rank,  
MPI_Comm comm )
```

Grupne operacije –zadaci

zad.Napisati MPI program koji izračunava vrednost broja PI kao vrednost integrala funkcije $f(x)=4/(1+x^2)$ na intervalu $[0,1]$.



$$\int_0^1 \frac{4.0}{(1+x^2)} dx = \pi$$

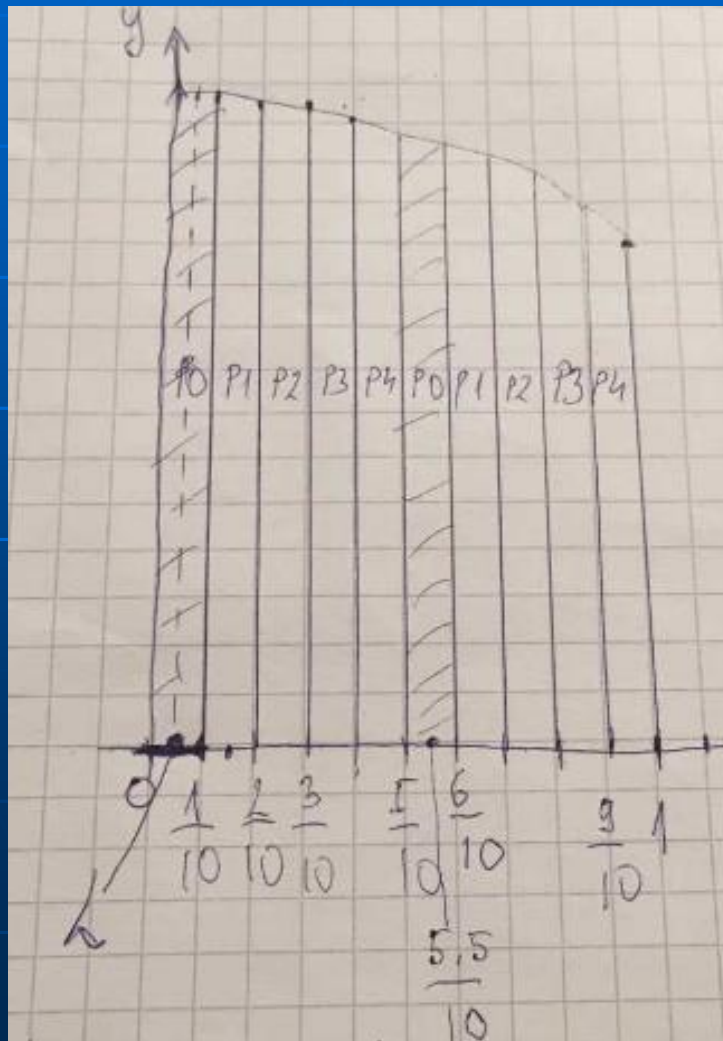
Vrednost ovog integrala funkcije $f(x)=4/(1+x^2)$ na intervalu $[0,1]$ se može aproksimirati sumom N površina pravougaonika:

$$\sum_{i=0}^N F(x_i) \Delta x \approx \pi$$

Što je veće N to je bolja aproksimacija
 Δx -veličina segmenta(podsegmenta)

Grupne operacije – zadaci

Mi ćemo širinu segmenta Δx obeležiti sa h , a vrednost funkcije $F(x_i)$ ćemo tražiti u tačkama x_i , gde je x_i tačka na sredini svakog segmenta.



Pr.N=10, N-broj segmenata
 $h=1/10$, h-veličina segmenta
 $p=5$, p-broj procesa

Grupne operacije – zadaci

Pr.N=10, N-broj segmenata
 $h=1/10$, h-veličina segmenta
 $p=5$, p-broj procesa

	PO	P1	P2	P3	P4
myid	0	1	2	3	4
sum	$f(0.5*h)$ $+f(5.5*h)$	$f(1.5*h)$ $+f(6.5*h)$	$f(2.5*h)$ $+f(7.5*h)$	$f(3.5*h)$ $+f(8.5*h)$	$f(4.5*h)$ $+f(9.5*h)$
mypi	sum*h	sum*h	sum*h	sum*h	sum*h

Grupne operacije – zadaci

```
#include "mpi.h"
#include <stdio.h>
#include <math.h>

void main(int argc, char *argv[])
{
    int n, myid, numprocs, i;
    double PI25DT = 3.141592653589793238462643;
    double mypi, pi, h, sum, x;

    MPI_Init(&argc,&argv);
    MPI_Comm_size(MPI_COMM_WORLD,&numprocs);
    MPI_Comm_rank(MPI_COMM_WORLD,&myid);
    if (myid == 0)
        scanf("%d",&n);
```

Grupne operacije – zadaci

```
MPI_Bcast(&n, 1, MPI_INT, 0, MPI_COMM_WORLD);
h = 1.0 / (double) n;
sum = 0.0;
for (i = myid; i < n; i += numprocs) {
    x = h * ((double)i + 0.5);
    sum += 4.0 / (1.0 + x*x);
}
mypi = h * sum;

MPI_Reduce(&mypi, &pi, 1, MPI_DOUBLE, MPI_SUM, 0,
           MPI_COMM_WORLD);

if (myid == 0)
    printf("pi is approximately %.16f, Error is %.16f\n",
           pi, fabs(pi - PI25DT));
MPI_Finalize();
}
```

Grupne operacije – zadaci

zadatak. Napisati MPI program koji izračunava vrednost skalarnog proizvoda dva vektora dimenzije N. Pretpostaviti da je N deljivo sa brojem procesa. Vrednosti vektora a i b se učitavaju u procesu P0.

$$\mathbf{a} \cdot \mathbf{b} = \sum_{i=1}^n a_i b_i = a_1 b_1 + a_2 b_2 + \cdots + a_n b_n$$

```
#include <mpi.h>
#include <stdio.h>
#define n 6
void main(int argc, char* argv[]) {
    float a[n], b[n], dot, local_dot=0;
    int i, n_bar, my_rank, p;
    MPI_Init(&argc, &argv);
    MPI_Comm_size(MPI_COMM_WORLD, &p);
    MPI_Comm_rank(MPI_COMM_WORLD, &my_rank);
    n_bar=n/p;
    float * local_a= (float *)malloc(n_bar*sizeof(float));
    float * local_b= (float *)malloc(n_bar*sizeof(float));
```

```

if (my_rank == 0)
    for (i = 0; i < n; i++)
        scanf("%f", &a[i]);

```

```

MPI_Scatter(a,n_bar,MPI_FLOAT,local_a,n_bar,MPI_FLOAT,0,MPI_COMM_WORLD);

```

```

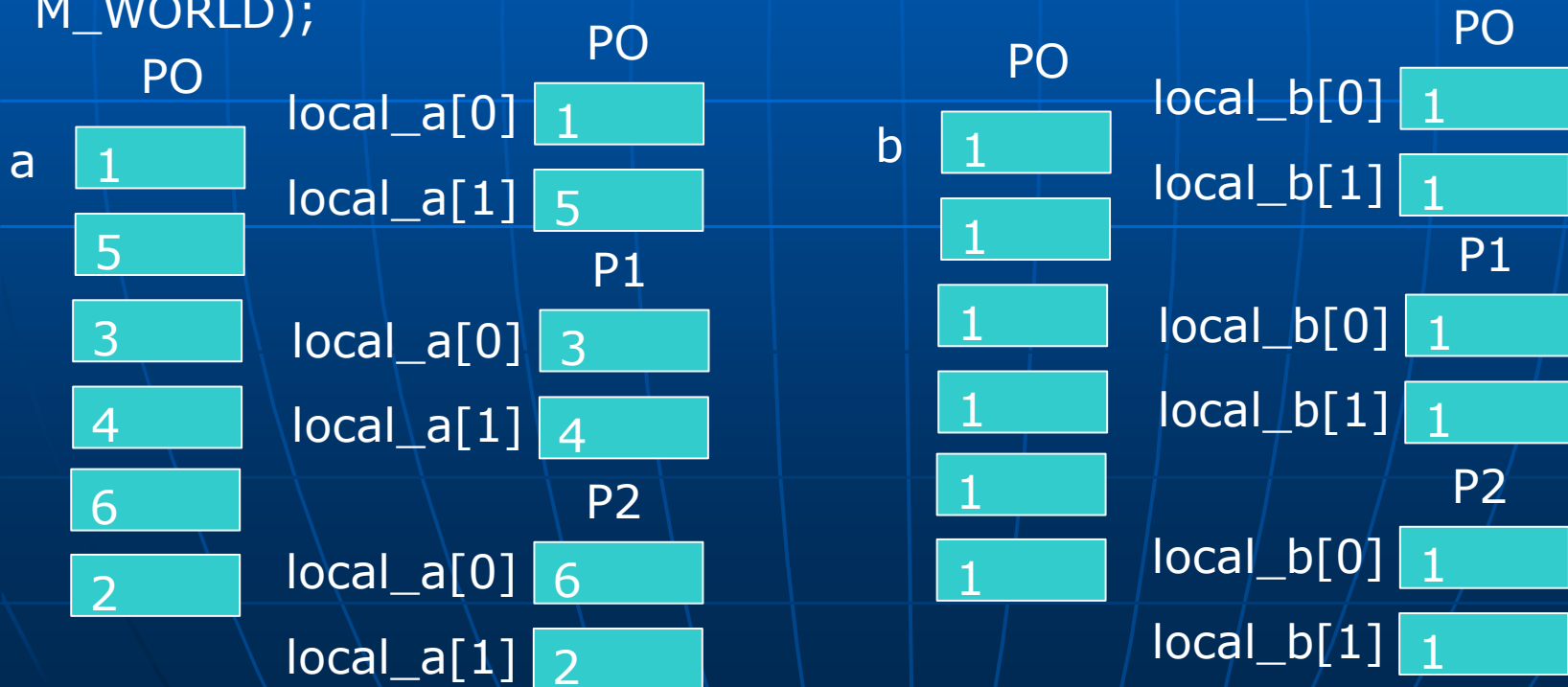
if (my_rank == 0)
    for (i = 0; i < n; i++)
        scanf("%f", &b[i]);

```

```

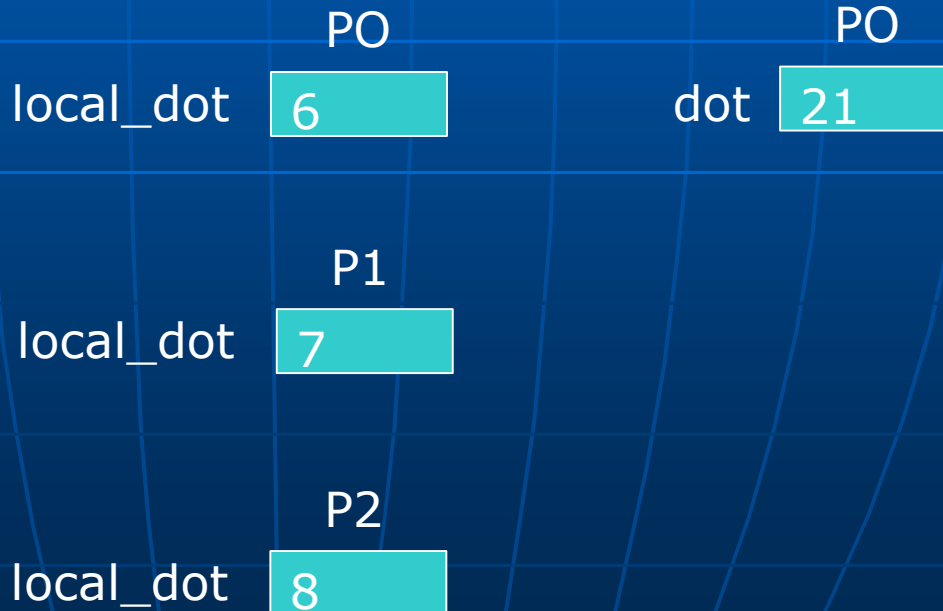
MPI_Scatter(b,n_bar,MPI_FLOAT,local_b,n_bar,MPI_FLOAT,0,MPI_COMM_WORLD);

```



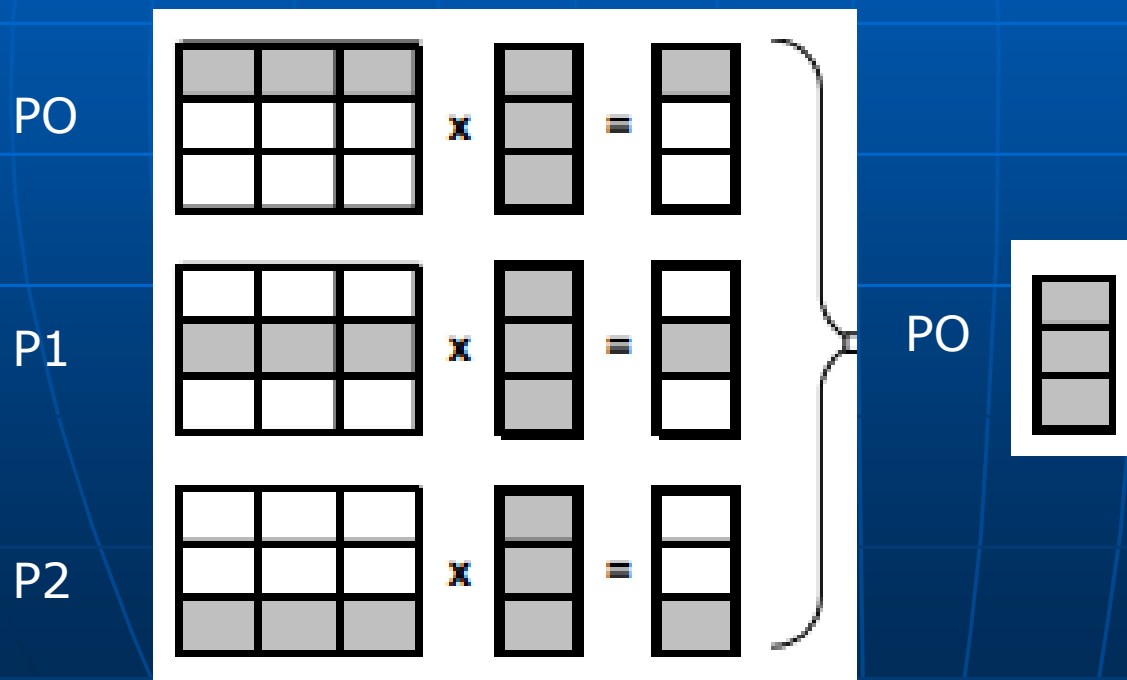
Grupne operacije – zadaci

```
for (i = 0; i < n_bar; i++)  
    local_dot = local_dot + local_a[i]*local_b[i];  
MPI_Reduce(&local_dot, &dot, 1, MPI_FLOAT,  
    MPI_SUM, 0, MPI_COMM_WORLD);  
  
if (my_rank == 0)  
    printf("The dot product is %f\n", dot);  
  
MPI_Finalize();  
}
```



Grupne operacije – zadaci

zad. Napisati MPI program koji pronalazi proizvod matrice $A_{n \times n}$ i vektora b_n . Matrica A i vektor b se inicijalizuju u procesu 0. Izračunavanje se obavlja tako što se svakom procesu distribuira po vrsta matrice A i ceo vektor b . Svi procesi učestvuju u izračunavanju. Rezultat se prokazuje u procesu 0.



Grupne operacije – zadaci

```
#include <stdio.h>
#include <mpi.h>
#define n 6

void main(int argc, char* argv[])
{
    int rank,p,i,j, a[n][n],b[n],local_a[n],local_c,c[n];
    MPI_Init(&argc, &argv);
    MPI_Comm_rank(MPI_COMM_WORLD,&rank);
    MPI_Comm_size(MPI_COMM_WORLD,&p);
    if (rank == 0)
    {
        for(i = 0; i < n; i++)
            for(j = 0; j < n; j++)
                a[i][j] = i+j;

        for(i = 0; i < n; i++)
            b[i] = 1;
    }
}
```

Grupne operacije – zadaci

```
MPI_Scatter(&a[0][0],n,MPI_INT,local_a,n, MPI_INT,0,MPI_COMM_WORLD);
MPI_Bcast(b,n,MPI_INT,0,MPI_COMM_WORLD);

local_c=0;
for (int i= 0; i<n; i++)
    local_c+= local_a[i]*b[i];

MPI_Gather(&local_c,1,MPI_INT,&c[0],1,MPI_INT,0,MPI_COMM_WORLD);

if (rank == 0)
{
    for (i = 0; i< n; i++)
    {
        printf("%d ",c[i]);

        printf("\n");
    }
}
```

Grupne operacije –zadaci

zad. Napisati MPI program koji pronalazi i prikazuje minimalni neparan broj sa zadatom osobinom i identifikator procesa koji ga sadrži. Neparni brojevi se nalaze u intervalu $[a,b]$ (a i b su zadate konstante). Osobina koju broj treba da poseduje je da je deljiv zadatom vrednošću x . Prilikom ispitivanja (da li broj poseduje zadatu osobinu ili ne) svaki proces **generiše** i ispituje odgovarajuće neparne brojeve na način prikazan na slici (za primer broj_procesa=4 i $a=3$, $b=31$, $x=5$). Konačne rezultate treba da prikaže proces koji sadrži najmanji broj takvih brojeva. Zadatak rešiti korišćenjem grupnih operacija.

P0	P1	P2	P3
3	5	7	9
11	13	15	17
19	21	23	25
27	29	31	

$broj_procesa = 4$ i $a = 3, b = 31, x = 5 \Rightarrow min = 5, id = 1, broj_brojeva_sa_zadatom_osobinom = 3$
 $id_procesa_koji_prikazuje_rezultate = 0$

Grupne operacije – zadaci

```
#include "mpi.h"
#include <stdio.h>
#include <limits.h>
#define a 5
#define b 31
#define x 5
void main(int argc, char* argv[]) {
    struct {int val;int rank;} f,c,d,e;
    int      my_rank, p,b1=0, z, min=INT_MAX;
    MPI_Init(&argc, &argv);
    MPI_Comm_size(MPI_COMM_WORLD, &p);
    MPI_Comm_rank(MPI_COMM_WORLD, &my_rank);
    for (z=a+2*my_rank;z<=b;z+=p*2)
        {
            if (z%x==0)
            {
                b1++;
                if (z<min)
                    min=z;
            }
        }
}
```

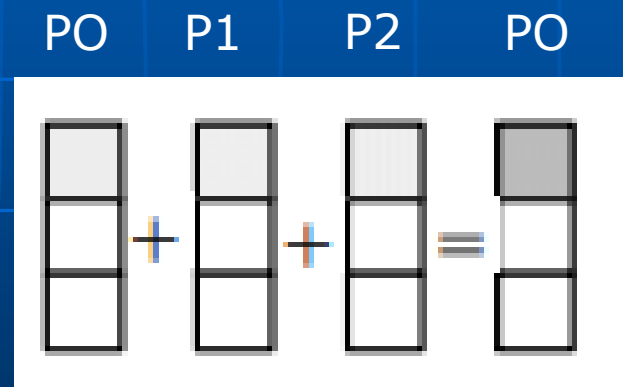
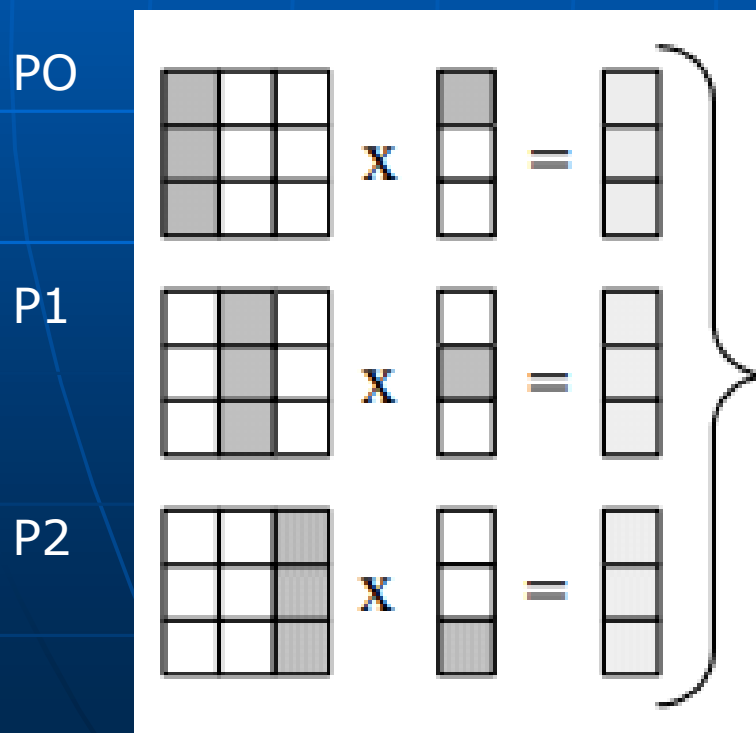
Grupne operacije – zadaci

```
c.val=min;
c.rank=my_rank;
d.val=b1;
d.rank=my_rank;
MPI_Reduce(&d,&e,1, MPI_2INT,MPI_MINLOC,0,MPI_COMM_WORLD);
MPI_Bcast(&e,1, MPI_2INT,0,MPI_COMM_WORLD);
MPI_Reduce(&c,&f,1, MPI_2INT,MPI_MINLOC,e.rank,MPI_COMM_WORLD);
if (my_rank == e.rank)
    printf("%d %d", f.val,f.rank);

MPI_Finalize();
}
```

Grupne operacije – zadaci

zad. Napisati MPI program koji pronalazi proizvod matrice $A_{m \times n}$ i vektora b_n . Matrica A i vektor b se inicijalizuju u procesu 0. Izračunavanje se obavlja tako što se svakom procesu distribuira po kolona matrice A i po 1 element vektora b . Za distribuciju kolona po procesima koristiti P-t-P operacije, za sve ostalo grupne operacije. Svi procesi učestvuju u izračunavanju. Rezultat se prikazuje u procesu koji, nakon distribuiranja kolona matrice A , sadrži minimum svih elemenata matrice A .



$$\begin{array}{rcl}
 a_{00} * b_0 & a_{01} * b_1 & a_{02} * b_2 \\
 a_{10} * b_0 & + a_{11} * b_1 & + a_{12} * b_2 \\
 a_{20} * b_0 & a_{21} * b_1 & a_{22} * b_2
 \end{array}$$

Grupne operacije – zadaci

```
#include <mpi.h>
#include <limits.h>
#define m 4
#define n 3
void main(int argc, char* argv[])
{
    int a[m][n], b[n], rank, p, i, j;
    int lc[m], c[m], y[m], x[m], z;
    struct {int val; int rank;} min, gmin;
    MPI_Status status;

    MPI_Init(&argc, &argv);
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    MPI_Comm_size(MPI_COMM_WORLD, &p);
    if (rank == 0)
    {
        for(i = 0; i < m; i++)
            for(j = 0; j < n; j++)
                a[i][j] = i+j;
```

```
for(j = 0; j < n; j++)  
    b[j] = 1;  
}
```

```
if (rank==0)  
{  
    for (i = 0; i < m; i++){x[i]=a[i][0];}  
  
    for(j = 1; j < p; j++)  
    {  
        for(i = 0; i < m; i++)  
  
            y[i] = a[i][j];  
  
        MPI_Send(y,m,MPI_INT,j,0,MPI_COMM_WORLD);  
    }  
}  
else  
    MPI_Recv(x,m,MPI_INT,0,0,MPI_COMM_WORLD,&status);
```

```
MPI_Scatter(&b[0],1,MPI_INT,&z,1,MPI_INT,0,MPI_COMM_WORLD);
for (int i= 0;i<m;i++)
    lc[i]= x[i]*z;
min.val=INT_MAX;
for (int i= 0;i<m;i++)
    if(x[i]<min.val)
        { min.val=x[i];min.rank=rank;}
MPI_Reduce(&min,&gmin,1,MPI_2INT,MPI_MINLOC,0,MPI_COMM_WORLD);
MPI_Bcast(&gmin,1,MPI_2INT,0,MPI_COMM_WORLD);
MPI_Reduce(lc,c,m,MPI_INT,MPI_SUM,gmin.rank,MPI_COMM_WORLD);
if (rank == gmin.rank)
{
    for (i = 0; i < m; i++)

        printf("c[%d]=%d\n",i,c[i]);
        printf("gmin=%d",gmin.val);

}

MPI_Finalize();

}
```