

# HP-SEE

## Uvod u MPI programiranje

[www.hp-see.eu](http://www.hp-see.eu)

Mihajlo Savic

Elektrotehnicki fakultet Banja Luka



# HP-SEE

High-Performance Computing Infrastructure  
for South East Europe's Research Communities



- Prezentacija zasnovana na
- University of Notre Dame
  - MPI Tutorial
  - Laboratory for Scientific Computing
  - <http://www.lam-mpi.org/tutorials/nd>
  - M.D. McNally
  - Jeremy Siek



- Vrste paralelnih mašina
  - Višeprocorske mašine
  - Klasteri
- Dijeljena memorija
- Komunikacija porukama
  - **MPI – Message Passing Interface**
  - MPICH, LAM/MPI, MPILite, ...



- Specifikacija biblioteke za prosljeđivanje poruka
- Podrška za brojne programske jezike
  - C, C++, Java, Fortran, ...
- Podrška za
  - Višeprocorske mašine
  - Klastere
  - Mrežno povezane računare



- Podrška za siguran rad sa porukama
- Podrška za niti
- Podrška za point-to-point komunikaciju
  - Blokirajuća, neblokirajuća, baferi, ...
- Podrška za kolektivnu komunikaciju
  - Dijeljenje i prikupljanje podataka, obrada u toku prikupljanja, ...
  - Veliki broj algoritama rada



- MPI ima više od stotinu funkcija/metoda
  - MPI-1 – 128
  - MPI-2 – 152
- Za rješavanje konkretnog problema nam rijetko trebaju sve
  - Mnoge probleme moguće riješiti sa mnogo manjim brojem (min. 6)

# Prvi MPI program



HP-SEE

High-Performance Computing Infrastructure  
for South East Europe's Research Communities

```
#include "mpi.h"  
#include <stdio.h>  
  
int main (int argc, char **argv) {  
    MPI_Init (&argc, &argv);  
    printf ("Moj prvi MPI program!\n");  
    MPI_Finalize ();  
}
```

# Kako dalje?



HP-SEE

High-Performance Computing Infrastructure  
for South East Europe's Research Communities

```
$ mpicc prvi.c -o prvi
```

```
$ mpirun -n 1 prvi
```

Moj prvi MPI program!

```
$ mpirun -n 2 prvi
```

Moj prvi MPI program!

Moj prvi MPI program!

```
$ mpirun -n 4 prvi
```

Moj prvi MPI program!

Moj prvi MPI program!

Moj prvi MPI program!

Moj prvi MPI program!



# Drugi MPI program



HP-SEE

High-Performance Computing Infrastructure  
for South East Europe's Research Communities

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

```
int main (int argc, char **argv) {  
    int ukupno, cvor;
```

```
    MPI_Init (&argc, &argv);
```

```
    MPI_Comm_size (MPI_COMM_WORLD, &ukupno);
```

```
    MPI_Comm_rank (MPI_COMM_WORLD, &cvor);
```

```
    printf ("Ukupno pokrenutih procesa: %d. Moj broj: %d\n", ukupno, cvor);
```

```
    MPI_Finalize ();
```

```
}
```



**\$ mpicc drugi.c -o drugi**

**\$ mpirun -n 4 drugi**

Ukupno pokrenutih procesa: 4. Moj broj: 0

Ukupno pokrenutih procesa: 4. Moj broj: 3

Ukupno pokrenutih procesa: 4. Moj broj: 1

Ukupno pokrenutih procesa: 4. Moj broj: 2



```
#include "mpi.h"
#include <stdio.h>
int main (int argc, char **argv) {
    int ukupno, cvor, rezultat;

    rezultat = MPI_Init (&argc, &argv);
    if (rezultat != MPI_SUCCESS) {
        printf ("Doslo je do greske pri pokretanju MPI programa.\n");
        MPI_Abort (MPI_COMM_WORLD, rezultat);
    }
    MPI_Comm_size (MPI_COMM_WORLD, &ukupno);
    MPI_Comm_rank (MPI_COMM_WORLD, &cvor);
    // Ovdje ide korisni dio koda
    printf ("Ukupno pokrenutih procesa: %d. Moj broj: %d\n", ukupno, cvor);

    MPI_Finalize ();
}
```



- Point-to-point komunikacija
- Šalju se podaci od početne adrese, navedene vrste i broja
- Vrsta – ugrađena ili korisnički izvedena iz ugrađenih (CHAR, INT, ...)
- Fleksibilan način komunikacije
- Isporuka poruka po zadanom redosljedu – ograničena garancija



- Blokirajuća komunikacija
- **MPI\_Send**(start, broj, vrsta, odredište, tag, domen)
- **MPI\_Recv**(start, broj, vrsta, izvorište, tag, domen, status)
- MPI\_ANY\_SOURCE i MPI\_ANY\_TAG kod prijema poruka
- Domen je za nas MPI\_COMM\_WORLD

```

#include <string.h>
char poruka_o[255], poruka_d[255];
int posiljalac, primalac, tag, velicina;
MPI_Status mpi_status;
if (broj == 0) {
    strcpy(poruka_o, "Ovo je poruka sa cvora 0.");
    rezultat = MPI_Send(poruka_o, strlen(poruka_o)+1, MPI_CHAR, 1, 9, MPI_COMM_WORLD);
    rezultat = MPI_Recv(poruka_d, 255, MPI_CHAR, 1, 4, MPI_COMM_WORLD, &mpi_status);
}
if (broj == 1) {
    strcpy(poruka_o, "Ovo je poruka sa cvora 1.");
    rezultat = MPI_Recv(poruka_d, 255, MPI_CHAR, 0, 9, MPI_COMM_WORLD, &mpi_status);
    rezultat = MPI_Send(poruka_o, strlen(poruka_o)+1, MPI_CHAR, 0, 4, MPI_COMM_WORLD);
}
if (broj < 2) {
    printf("Proces %d primio poruku od procesa %d sljedeceg sadrzaja: %s\n",
        broj, mpi_status.MPI_SOURCE, poruka_d);

    rezultat = MPI_Get_count(&mpi_status, MPI_CHAR, &velicina);

    printf("Proces %d je primio %d poruka od procesa %d sa tagom %d \n\n",
        broj, velicina, mpi_status.MPI_SOURCE, mpi_status.MPI_TAG);
}

```



\$ mpicc poruke.c -o poruke

\$ mpirun -n 4 poruke

Ukupno pokrenutih procesa: 4 Moj broj: 0

Ukupno pokrenutih procesa: 4 Moj broj: 2

Ukupno pokrenutih procesa: 4 Moj broj: 1

Ukupno pokrenutih procesa: 4 Moj broj: 3

Proces 1 primio poruku od procesa 0 sljedeceg sadrzaja: Ovo je odlazna poruka sa cvora 0.

Proces 1 je primio 34 poruka od procesa 0 sa tagom 9

Proces 0 primio poruku od procesa 1 sljedeceg sadrzaja: Ovo je poruka sa cvora 1!

Proces 0 je primio 26 poruka od procesa 1 sa tagom 4



• Veliki broj programa se može realizovati pomoću samo:

- MPI\_Init
- MPI\_Comm\_size
- MPI\_Comm\_rank
- MPI\_Send
- MPI\_Recv
- MPI\_Finalize





- MPI\_Bcast(start, broj, vrsta, root, domen)
  - Šalje podatke sa jednog čvora svim ostalima
- MPI\_Reduce(start, broj, vrsta, operacija, root, domen)
  - Prikuplja podatke sa svih čvorova i daje ih jednom procesu
  - Operacija: min/max/suma/proizvod/AND/...
  - Ili korisnički definisana



MPI\_Scatter(start1, broj1, vrsta1,  
start2, broj2, vrsta2, root, domen)

Šalje izdijeljene podatke sa jednog čvora svim po dio

MPI\_Gather(start1, broj1, vrsta1,  
start2, broj2, vrsta2, root, domen)

Prikuplja parcijalne podatke sa svih čvorova i kombinuje ih u jednom procesu

MPI\_Barrier(domen)

Sinhronizacija - svi čekaju dok ne stignu na isto mjesto



```
// Pocetak korisnog dijela koda
```

```
rezultat = MPI_Barrier(MPI_COMM_WORLD);
```

```
if (broj == 0) {
```

```
    broj_unosa=32;
```

```
// Saljemo poruku sa roota
```

```
    MPI_Bcast(&broj_unosa,1,MPI_INT,0,MPI_COMM_WORLD);
```

```
} else {
```

```
// Primamo poruku na drugim cvorovima
```

```
    rezultat = MPI_Bcast(&broj_unosa,1,MPI_INT,0,MPI_COMM_WORLD);
```

```
    printf("Cvor %d primljena poruka: %d\n",broj,broj_unosa);
```

```
}
```



```
$ mpicc kol1.c -o kol1
```

```
$ mpirun -n 4 kol1
```

```
Ukupno pokrenutih procesa: 4 Moj broj: 0
```

```
Ukupno pokrenutih procesa: 4 Moj broj: 3
```

```
Ukupno pokrenutih procesa: 4 Moj broj: 1
```

```
Ukupno pokrenutih procesa: 4 Moj broj: 2
```

```
Cvor 1 primljena poruka: 32
```

```
Cvor 2 primljena poruka: 32
```

```
Cvor 3 primljena poruka: 32
```

```
Cvor 3 primljena poruka: 32
```



- Moguća je i komunikacija svi-svi
  - Naziv se mijenja tako da
  - MPI\_Reduce postaje MPI\_**All**reduce
  - U tom slučaju svi čvorovi primaju poruku
  
- Promjenjiva veličina podataka
  - MPI\_Gather postaje MPI\_Gather**v**

# MPI kolektivna komunikacija



HP-SEE

High-Performance Computing Infrastructure  
for South East Europe's Research Communities

A			

BROADCAST



A			
A			
A			
A			

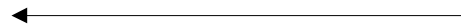
A	B	C	D

SCATTER



A			
B			
C			
D			

GATHER



A			
B			
C			
D			

ALLGATHER



A	B	C	D
A	B	C	D
A	B	C	D
A	B	C	D