Računalniške delavnice FMF: Delavnica MPI

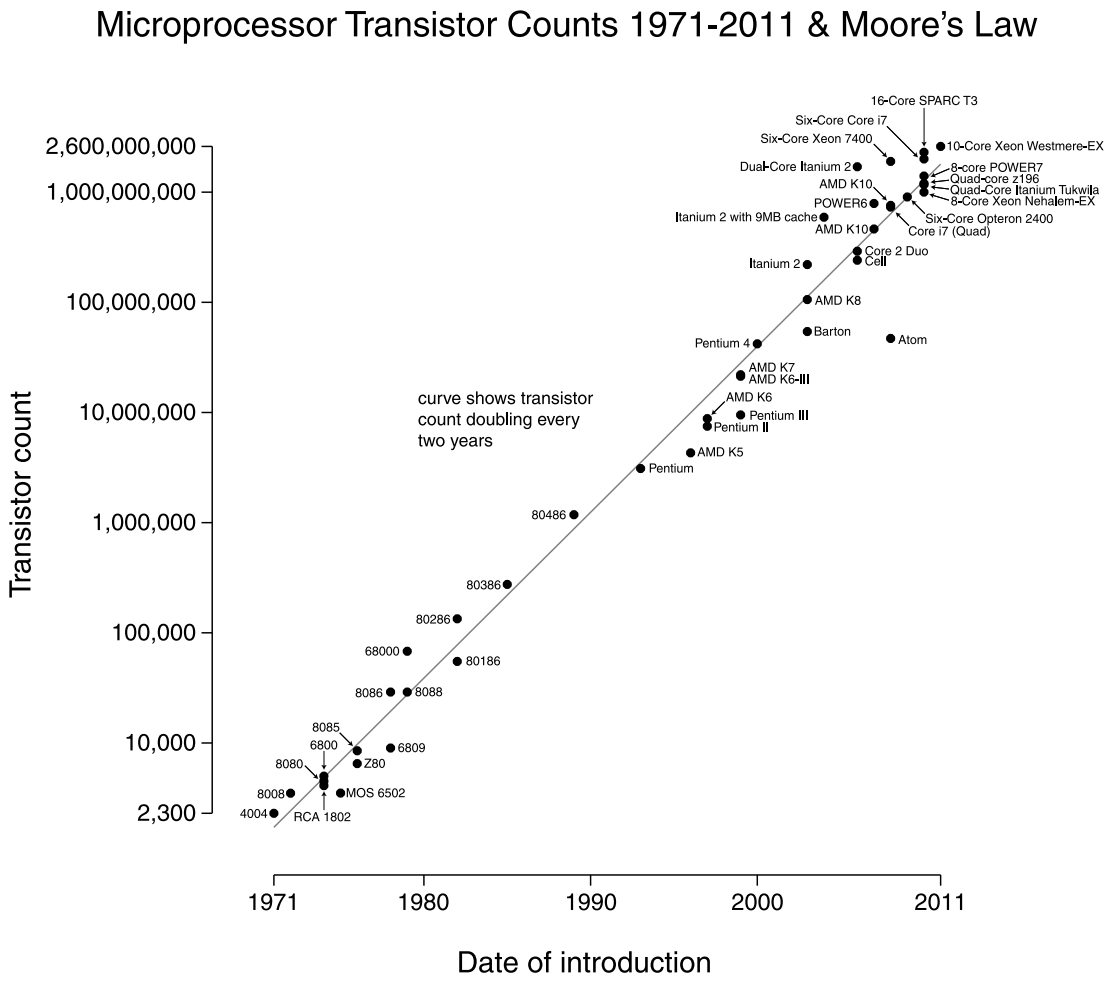
Matevž Jekovec
matevz.jekovec@fri.uni-lj.si

Laboratorij za vseprisotne sisteme
UL FRI
lusy.fri.uni-lj.si

18. april 2014
22. maj 2014

Motivacija

Moorov zakon: število tranzistorjev v integriranih vezjih se podvoji vsaki dve leti.



Motivacija

Kaj pa *zmogljivost*?

Motivacija

Kaj pa *zmogljivost*?
Frekvenca se poveča za faktor 1,4 vsako leto: neposredna pohitritev.

Motivacija

Kaj pa *zmogljivost*?
Frekvenca se poveča za faktor 1,4 vsako leto: neposredna pohitritev.
Frekvenca Pentiuma 4 doseže ≈ 4 GHz leta 2000 (v laboratoriju tudi 10 GHz).

Motivacija

Kaj pa *zmogljivost*?
Frekvenca se poveča za faktor 1,4 vsako leto: neposredna pohitritev.
Frekvenca Pentiuma 4 doseže ≈ 4 GHz leta 2000 (v laboratoriju tudi 10 GHz).
Po tem frekvenca stagnira. Zmogljivost se povečuje z drugimi optimizacijami in *vzporednostjo*.

Družine vzporednih arhitektur

Način komunikacije:

- z deljenim pomnilnikom,
- z izmenjavo sporočil.

Družine vzporednih arhitektur

Način komunikacije:

- z deljenim pomnilnikom,
- z izmenjavo sporočil.

Družine vzporednih arhitektur:

- Večjedrne arhitekture s skupnim pomnilnikom
- Mnogojedrne arhitekture s skupnim pomnilnikom
- Porazdeljene arhitekture z izmenjavo sporočil

Večjedrne arhitekture z deljenim pomnilnikom

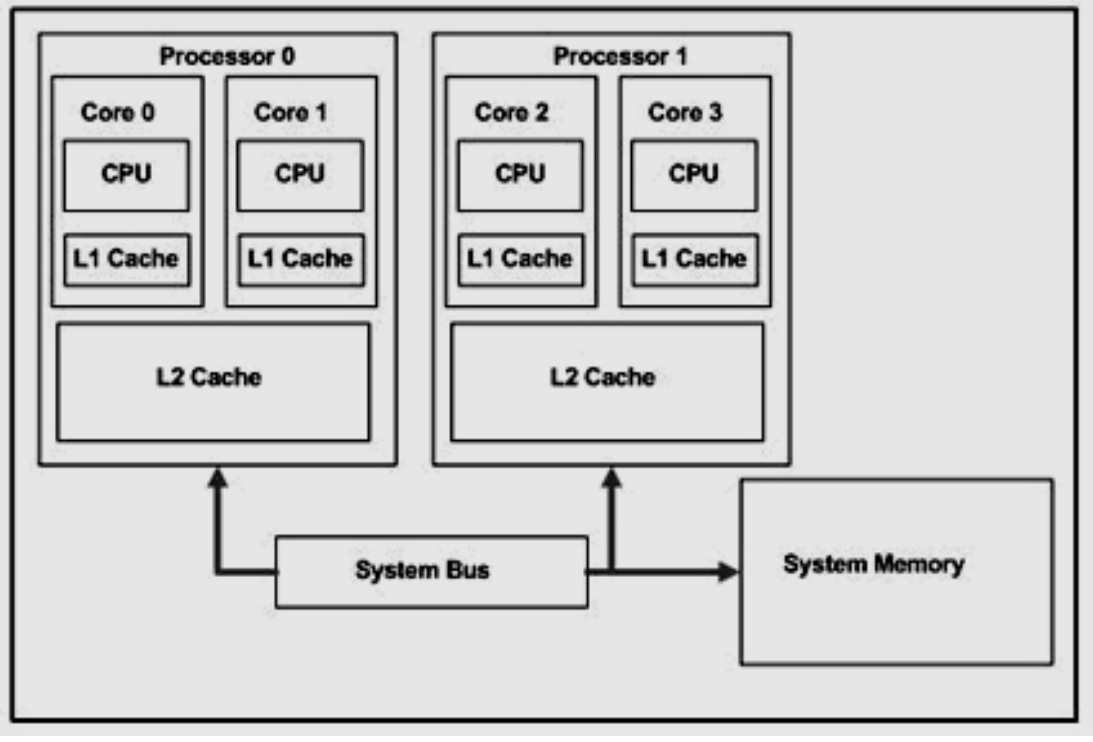
Nekaj 10 jeder na enem čipu.

Večjedrne arhitekture z deljenim pomnilnikom

Nekaj 10 jeder na enem čipu.
Vsa jedra imajo dostop do celotnega pomnilnika.

Večjedrne arhitekture z deljenim pomnilnikom

Nekaj 10 jeder na enem čipu.
Vsa jedra imajo dostop do celotnega pomnilnika.
Primer: Osebni računalniki Intel/AMD.



Mnogojedrne arhitekture s skupnim pomnilnikom

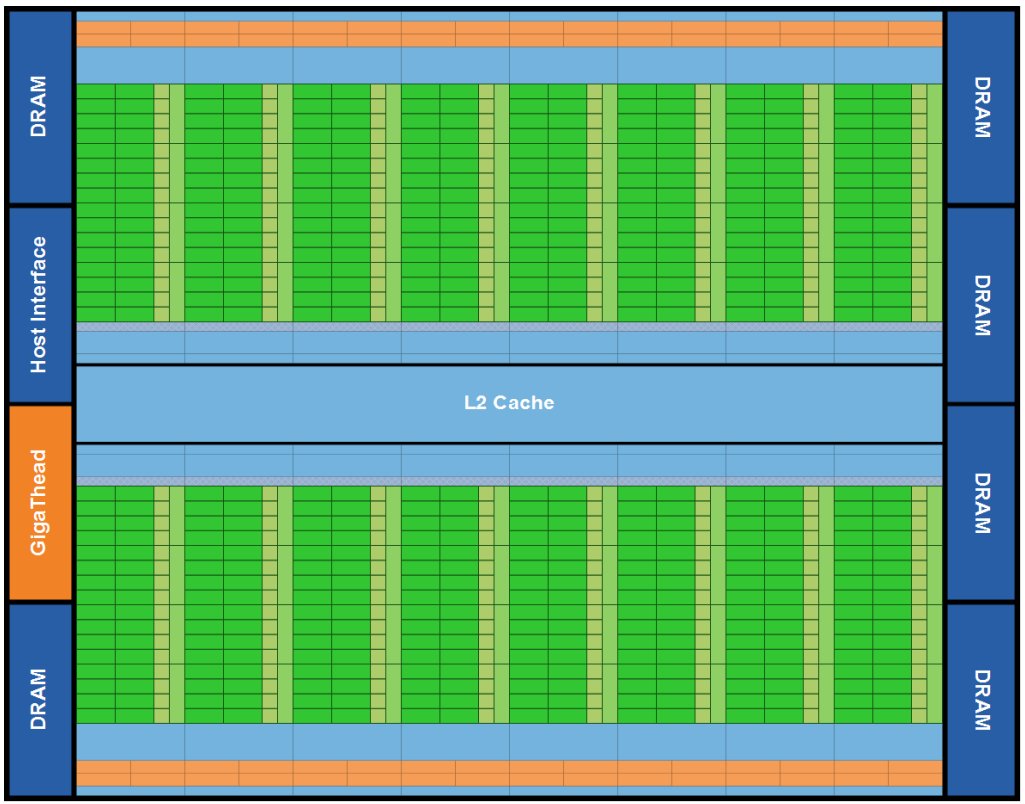
Več 1000 jeder na enem čipu. Jedra so enostavnejša. Nimajo vsi dostopa do vsega pomnilnika.

Mnogojedrne arhitekture s skupnim pomnilnikom

Več 1000 jeder na enem čipu. Jedra so enostavnejša. Nimajo vsi dostopa do vsega pomnilnika.
Predpostavka: Single-Instruction Multiple-Data

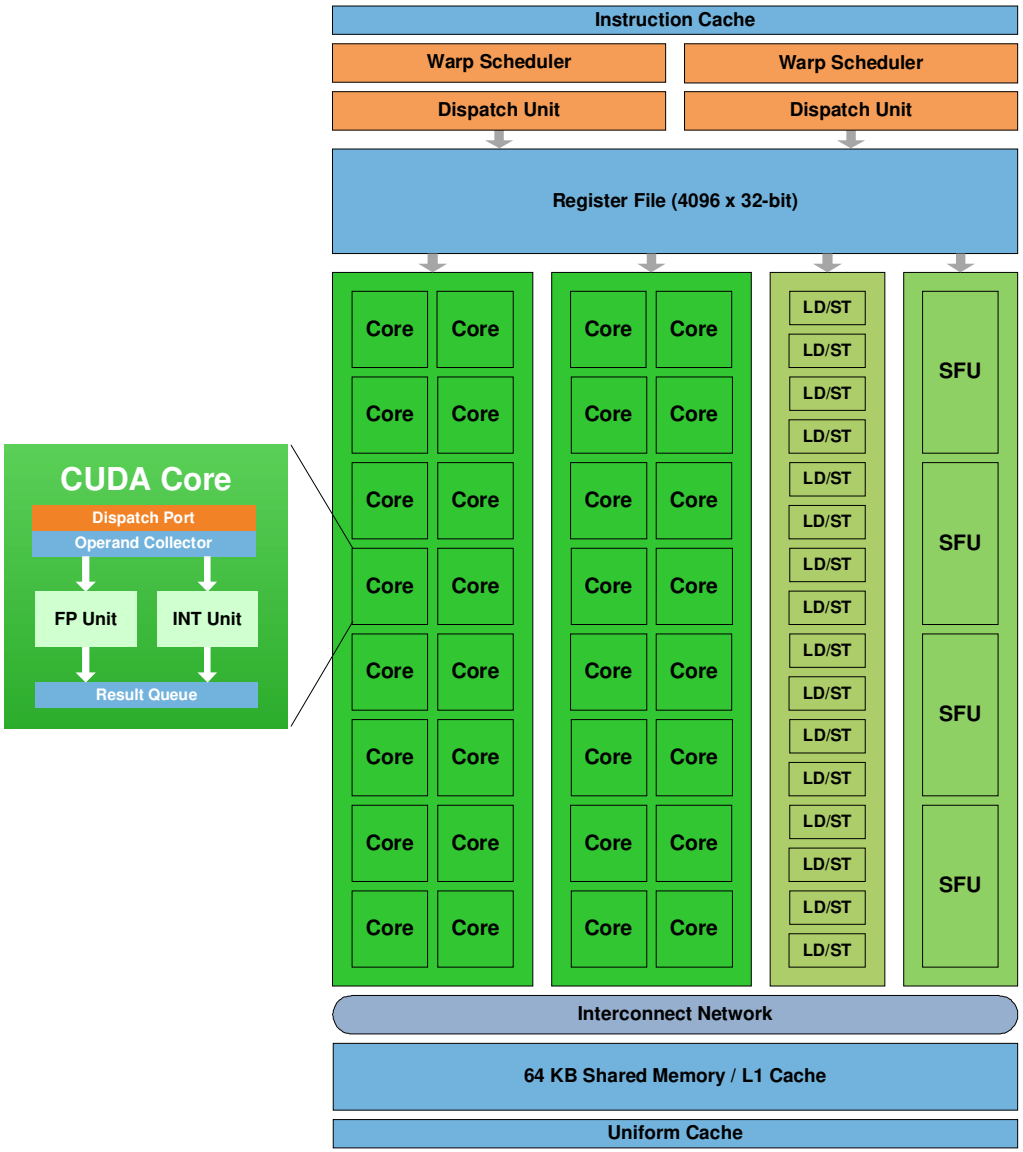
Mnogojedrne arhitekture s skupnim pomnilnikom

Več 1000 jeder na enem čipu. Jedra so enostavnejša. Nimajo vsi dostopa do vsega pomnilnika.
Predpostavka: Single-Instruction Multiple-Data
Primer: NVidia CUDA



Fermi's 16 SM are positioned around a common L2 cache. Each SM is a vertical rectangular strip that contain an orange portion (scheduler and dispatch), a green portion (execution units), and light blue portions (register file and L1 cache).

Mnogojedrne arhitekture s skupnim pomnilnikom



Fermi Streaming Multiprocessor (SM)

Porazdeljene arhitekture z izmenjavo sporočil

Več 1000 vozlišč, povezanih v mrežo.

Porazdeljene arhitekture z izmenjavo sporočil

Več 1000 vozlišč, povezanih v mrežo.

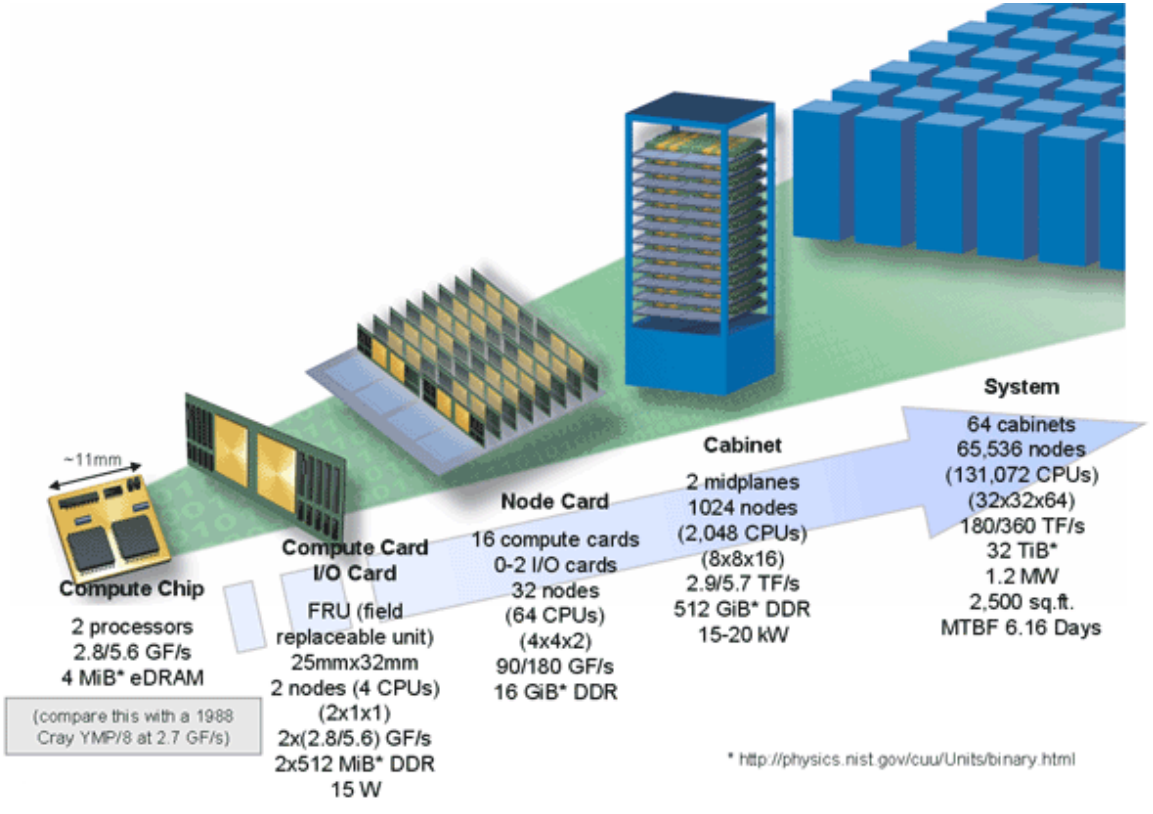
Vsako vozlišče ima neposreden dostop do lokalnega pomnilnika.

Porazdeljene arhitekture z izmenjavo sporočil

Več 1000 vozlišč, povezanih v mrežo.
Vsako vozlišče ima neposreden dostop do lokalnega pomnilnika.
Dostop do “bolj oddaljenega pomnilnika” z izmenjavo sporočil.

Porazdeljene arhitekture z izmenjavo sporočil

Več 1000 vozlišč, povezanih v mrežo.
Vsako vozlišče ima neposreden dostop do lokalnega pomnilnika.
Dostop do "bolj oddaljenega pomnilnika" z izmenjavo sporočil.
Primer: IBM Bluegene/L



Kako programiramo vzporedne aplikacije?

Kako izgledajo vzporedne aplikacije?

Kako programiramo vzporedne aplikacije?

Kako izgledajo vzporedne aplikacije?
Odvisno od arhitekture!

Kako programiramo vzporedne aplikacije?

Kako izgledajo vzporedne aplikacije?
Odvisno od arhitekture!
Deljen pomnilnik:

- en proces, več niti ali
- več procesov, izmenjava sporočil.

Kako programiramo vzporedne aplikacije?

Kako izgledajo vzporedne aplikacije?
Odvisno od arhitekture!
Deljen pomnilnik:

- en proces, več niti ali
- več procesov, izmenjava sporočil.

Porazdeljene arhitekture:

- več procesov, izmenjava sporočil.

Pohitritev aplikacije

POZOR: Premisliti moramo, kaj bomo sploh povzporejali!

Pohitritev aplikacije

POZOR: Premisliti moramo, kaj bomo sploh povzporejali!
Če so vsak naslednji korak algoritma odvisen od prejšnjega, je aplikacija *inherentno zaporedna* (inherently sequential) in ni mogoče povzporejanje.
Primer: iskanje po drevesu v globino.

Pohitritev aplikacije

POZOR: Premisliti moramo, kaj bomo sploh povzporejali!
Če so vsak naslednji korak algoritma odvisen od prejšnjega, je aplikacija *inherentno zaporedna* (inherently sequential) in ni mogoče povzporejanje.
Primer: iskanje po drevesu v globino.
Če so koraki popolnoma neodvisni med seboj, je aplikacija *sramotno povzporedljiva* (embarrassingly parallel).
Primer: iskanje števila po neurejenem polju števil.

Pohitritev aplikacije

POZOR: Premisliti moramo, kaj bomo sploh povzporejali!
Če so vsak naslednji korak algoritma odvisen od prejšnjega, je aplikacija *inherentno zaporedna* (inherently sequential) in ni mogoče povzporejanje.
Primer: iskanje po drevesu v globino.
Če so koraki popolnoma neodvisni med seboj, je aplikacija *sramotno povzporedljiva* (embarrassingly parallel).
Primer: iskanje števila po neurejenem polju števil.
Amdahlov zakon: Največja možna pohitritev algoritma je

$$S(P) = \frac{1}{s + \frac{1-s}{P}},$$

kjer je s delež inherentno zaporedne kode in P število procesorjev.

MPI

MPI ali Message Passing Interface je programski standard, ki vsebuje funkcionalnosti, potrebne za razvoj aplikacij za porazdeljen sisteme. Zasnovan je bil leta 1991.

MPI

MPI ali Message Passing Interface je programski standard, ki vsebuje funkcionalnosti, potrebne za razvoj aplikacij za porazdeljen sisteme. Zasnovan je bil leta 1991.
Podpira navidezno topologijo mreže (naslavljanje), sinhronizacijo procesorjev in učinkovito medsebojno komunikacijo.

MPI

MPI ali Message Passing Interface je programski standard, ki vsebuje funkcionalnosti, potrebne za razvoj aplikacij za porazdeljen sisteme. Zasnovan je bil leta 1991.
Podpira navidezno topologijo mreže (naslavljanje), sinhronizacijo procesorjev in učinkovito medsebojno komunikacijo.
Je namenjen porazdeljenim arhitekturam, vendar deluje tudi na večjedrnih.

MPI

MPI ali Message Passing Interface je programski standard, ki vsebuje funkcionalnosti, potrebne za razvoj aplikacij za porazdeljen sisteme. Zasnovan je bil leta 1991.
Podpira navidezno topologijo mreže (naslavljanje), sinhronizacijo procesorjev in učinkovito medsebojno komunikacijo.
Je namenjen porazdeljenim arhitekturam, vendar deluje tudi na večjedrnih.
Je prirejen za številne programske jezike: poleg C, C++ in Fortran tudi za Python, Ocaml, Javo, Matlab, R itd.

MPI

MPI ali Message Passing Interface je programski standard, ki vsebuje funkcionalnosti, potrebne za razvoj aplikacij za porazdeljen sisteme. Zasnovan je bil leta 1991.
Podpira navidezno topologijo mreže (naslavljanje), sinhronizacijo procesorjev in učinkovito medsebojno komunikacijo.
Je namenjen porazdeljenim arhitekturam, vendar deluje tudi na večjedrnih.
Je prirejen za številne programske jezike: poleg C, C++ in Fortran tudi za Python, Ocaml, Javo, Matlab, R itd.
Open MPI je ena izmed knjižnic, ki se podrejajo standardu MPI-2.

Uporaba MPI v C

Začnimo s Hello world v programskem jeziku C:

```
1 #include <stdio.h>
2
3 int main(int argc, char* argv[]) {
4         printf("Pozdravljen, svet!\n");
5
6         return 0;
7 }
```

Uporaba MPI v C

Dodamo inicializacijo MPI in izpis procesa:

```
 1 #include <stdio.h>
 2 #include <mpi.h>
 3
 4 int main(int argc, char* argv[]) {
 5   int rank, size;
 6
 7   MPI_Init(&argc, &argv);
 8   MPI_Comm_rank(MPI_COMM_WORLD, &rank);
 9   MPI_Comm_size(MPI_COMM_WORLD, &size);
10
11   printf("Svet, jaz sem proces %d od %d!\n", rank,
         size);
12
13   MPI_Finalize();
14
15   return 0;
16 }
```

Uporaba MPI v C

Aplikacijo prevedemo z ukazom

```
mpicc helloworld-mpi.c -o helloworld-mpi
```

Aplikacijo zaženemo z ukazom

```
mpirun -n <število procesov> ./helloworld-mpi
```

```
mpirun -n 5 ./helloworld-mpi
Svet, jaz sem proces 1 od 5!
Svet, jaz sem proces 0 od 5!
Svet, jaz sem proces 4 od 5!
Svet, jaz sem proces 3 od 5!
Svet, jaz sem proces 2 od 5!
```

Uporaba MPI v C

Aplikacijo prevedemo z ukazom

```
mpicc helloworld-mpi.c -o helloworld-mpi
```

Aplikacijo zaženemo z ukazom

```
mpirun -n <število procesov> ./helloworld-mpi
```

```
mpirun -n 5 ./helloworld-mpi
Svet, jaz sem proces 1 od 5!
Svet, jaz sem proces 0 od 5!
Svet, jaz sem proces 4 od 5!
Svet, jaz sem proces 3 od 5!
Svet, jaz sem proces 2 od 5!
```

Opomba: Vrstni red izvajanja ni deterministično!

Komunikacija

Komunikacija lahko poteka na dva načina:

Komunikacija

Komunikacija lahko poteka na dva načina:

- prek mrežnega diska,

Komunikacija

Komunikacija lahko poteka na dva načina:

- prek mrežnega diska,
- prek MPI.

Komunikacija

Komunikacija lahko poteka na dva načina:

- prek mrežnega diska,
- prek MPI.

Funkciji MPI za komunikacijo:

```
int MPI_Send(const void *buf, int count, MPI_Datatype,
             int dest, int tag, MPI_Comm comm)
int MPI_Recv(void *buf, int count, MPI_Datatype,
             int source, int tag, MPI_Comm comm,
             MPI_Status *status)
```

Komunikacija

Komunikacija lahko poteka na dva načina:

- prek mrežnega diska,
- prek MPI.

Funkciji MPI za komunikacijo:

```
int MPI_Send(const void *buf, int count, MPI_Datatype,
             int dest, int tag, MPI_Comm comm)
int MPI_Recv(void *buf, int count, MPI_Datatype,
             int source, int tag, MPI_Comm comm,
             MPI_Status *status)
```

Opomba: Funkciji sta sinhroni in čakata, dokler se prenos ne konča.

Naloge

Današnje naloge: Napišite program, ki s standardnega vhoda prebere

1. vrednosti polja in vzporedno sešteje njegove elemente.
2. vrednosti dveh matrik in jih vzporedno sešteje.
3. vrednosti dveh matrik in jih vzporedno zmnoži.

Algoritme najprej preizkusite lokalno, nato pa na gruči Raspberry π.

Naloge

Današnje naloge: Napišite program, ki s standardnega vhoda prebere

1. vrednosti polja in vzporedno sešteje njegove elemente.
2. vrednosti dveh matrik in jih vzporedno sešteje.
3. vrednosti dveh matrik in jih vzporedno zmnoži.

Algoritme najprej preizkusite lokalno, nato pa na gruči Raspberry π.

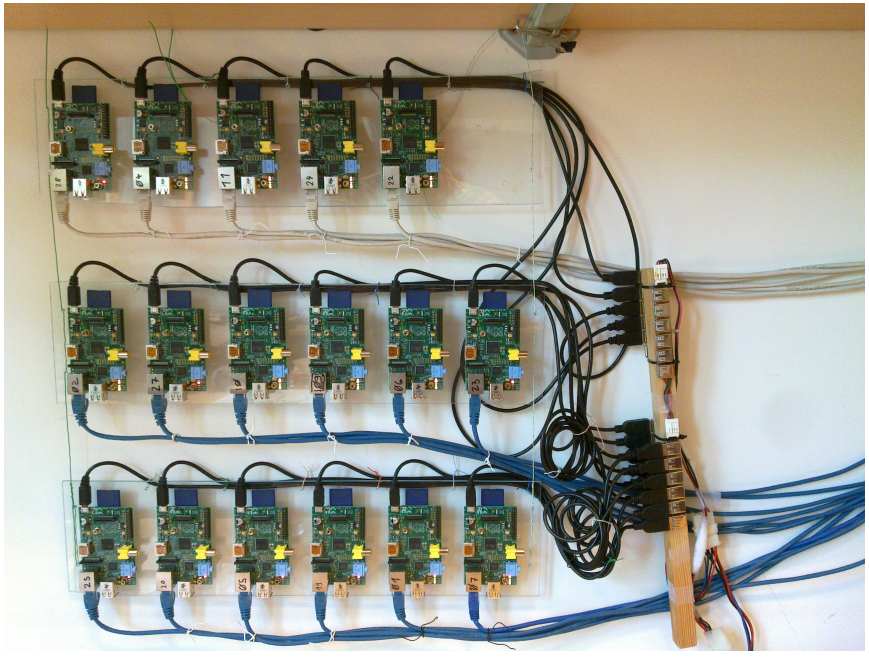
Za pridne:

1. Izmerite razliko v hitrosti izvajanja, če preberete matriko z mrežnega diska ali prek MPI.
2. Poglejte Cannonov algoritem za množenje matrik.

Dodatno: Zagon MPI v gruči

Ustvarimo datoteko z naslovi računalnikov:

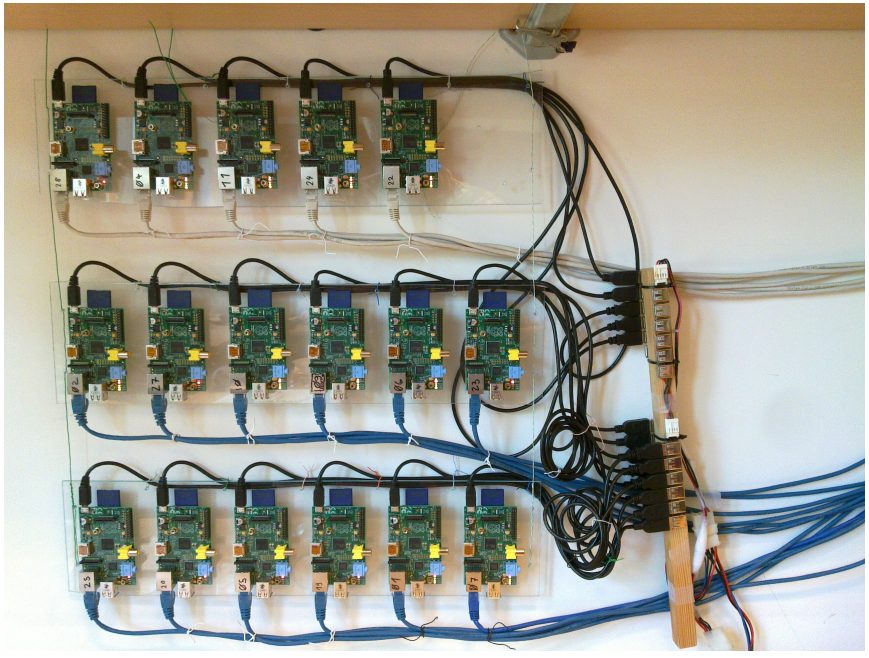
```
pi@10.100.0.10
pi@10.100.0.11
pi@10.100.0.12
pi@10.100.0.13
pi@10.100.0.14
pi@10.100.0.15
```



Dodatno: Zagon MPI v gruči

Ustvarimo datoteko z naslovi računalnikov:

```
pi@10.100.0.10
pi@10.100.0.11
pi@10.100.0.12
pi@10.100.0.13
pi@10.100.0.14
pi@10.100.0.15
```



Aplikacijo zaženemo z ukazom:

```
mpirun --hostfile my_hostfile helloworld-mpi
```