Porazdeljeni sistemi

1. Uvod

Predavatelj: izr. prof. Uroš Lotrič
Asistent: Davor Sluga

Cilji predmeta

- Spoznavanje s sistemi za vzporedno in porazdeljeno procesiranje
- Uporaba (programiranje) vzporednih in porazdeljenih sistemov
- Osvojitev načina razmišljanja, potrebnega za učinkovito izkoriščanje omenjenih sistemov
- Naučiti se izbrati in vzpostaviti ustrezen računalniški sistem, ki bi učinkovito reševal realni problem

Organizacija predmeta

- Uvod v paralelno procesiranje
- Sistemi s skupnim pomnilnikom
  - Programiranje z nitmi
  - Programiranje s knjižnico pThread
- Porazdeljeni računalniški sistemi
  - Knjižnica MPI
  - Porazdeljeni algoritmi
- Računanje na grafičnih karticah
  - Ogrodje OpenCL

Pomembni podatki

- Spletna stran predmeta:
  - https://ucilnica.fri.uni-lj.si/ps
- Elektronska pošta:
  - uros.lotric@fri.uni-lj.si
  - davor.sluga@fri.uni-lj.si
- Govorilne ure:
  - Uroš Lotrič: pred ali po predavanjih ali
    po dogovoru po elektronski pošti

Pomembni podatki

- Ocenjevanje:
  - 1/3 + 1/3 + 1/3
    - ocene sprotnih (kratkih) nalog
    - ocena projekta
    - znanje na ustnem izpitu

Pomembni podatki

- Izpitni roki:
  - še niso znani

Pomembni podatki

- Orodja:
  - V veliki meri bomo uporabljali jezik C
  - Okolje Linux
  - VS Code
  - Različne knjižnice in ogrodja za vzporedno in paralelno procesiranje

Pomembni podatki

- Literatura:
  - R. Trobec, B. Slivnik, P. Bulić, B. Robič: Introduction to Parallel Computing, Springer, 2018
  - P. S. Pacheco: An introduction to parallel programming, Morgan Kaufmann, 2011
  - M.J. Quinn: Parallel programming in C with MPI and OpenMPI, Mc Graw Hill, Boston, 2003
  - N. Matloff: Programming on Parallel Machines, http://heather.cs.ucdavis.edu/~matloff/158/PLN/ParProcBook.pdf, University of California, Davis
  - D.B. Kirk, W.W. Hwu: Programming Massively Parallel Processors, Morgan Kaufmann, Burlington, 2010

Uvod

- Vam hitro ni dovolj hitro?
  - Današnji računalniki so 100-krat hitrejši od tistih izpred desetletja, vendar mnogokrat to ni dovolj
  - Današnji računalniki imajo mnogo več spominskih kapacitet kot pred desetletjem, vendar to mnogokrat ni dovolj

Uvod

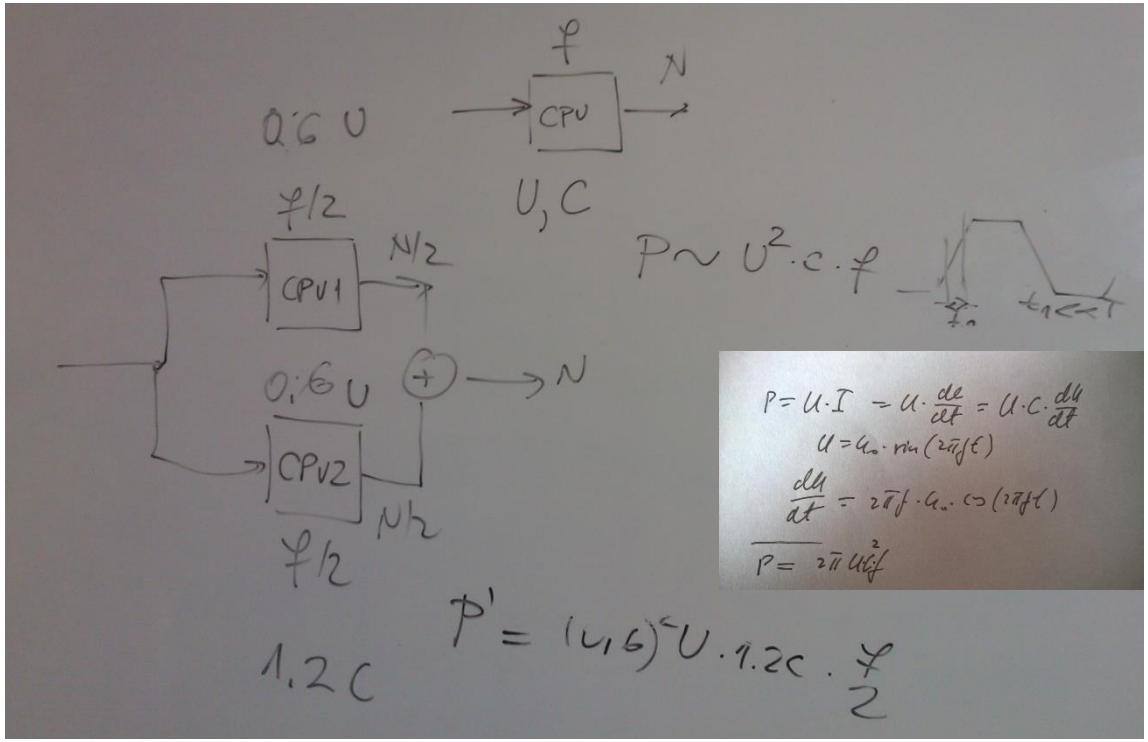
- Vam hitro ni dovolj hitro?
  - Veliki izzivi:
    - Modeliranje vremena in obnašanja okolja
    - Kvantna kemija, statistična mehanika in relativnostna teorija
    - Raziskovanje vesolja in astrofizika
    - Raziskovanje dinamike tekočin in turbulenc
    - Energetika (turbine, vetrnice)
    - Razvoj materialov in superprevodnost
    - Biologija, genetika, genski inženiring, določanje zgradbe proteinov, delovanje encimov, modeliranje celice
    - Medicina, odkrivanje zdravil, modeliranje človeških organov in kosti
    - Podatkovna analitika (podatkovno rudarjenje, iskanje podatkov)

Uvod

- Je vzporedno programiranje res neizogibno?
  - Tehnologija izdelave čipov je (spet) trčila ob oviro - problemi z miniaturizacijo elementov na silicijevi rezini
  - Zmogljivejše sisteme trenutno dobimo predvsem z množenjem gradnikov

Uvod

- Zakaj vzporedni sistemi? Kako povečati hitrost?
  - Monolitne arhitekture, zmanjševanje elementov
    - so bližje skupaj, lahko večje hitrosti
      - do 2005 50 % povečanje vsako leto (60 x v 10 letih)
      - danes 20 % povečanje kapacitet vsako leto (6 x v 10 letih)
    - Energija in moč
      - Hitrejši preklopi → večja poraba
      - Oddajanje toplote → nezanesljivo delovanje
      - P'/P = 2 x 0,2 = 0,4



Uvod

- Kaj je vzporedno računanje?
  - Uporaba vzporednih računalnikov z namenov skrajšanja časa reševanja posameznega problema
  - Standardni pristop pri reševanju zgoraj omenjenih izzivov

Uvod

- Zakaj moramo pisati vzporedne programe?
  - Programi za eno-jedrne sisteme ne znajo izkoriščati več jeder
  - Lahko na vsakem jedru zaženemo svojo instanco programa, večinoma ne pomaga (več instanc igre ali ena z boljšo grafiko?)
  - Prepisati moramo programe v vzporedno obliko

  - Slaba novica: kljub dolgoletnim raziskavam avtomatsko pretvarjanje ne deluje dobro
    - Prevajalniki, ki bi znali naše zaporedne programe samostojno pretvoriti v vzporedno obliko, so v razvojni fazi in so slabi
    - Ideja: iščejo se tipični konstrukti in se jih pretvarja v paralelno obliko

  - Vas slabo delovanje preseneča?
    - Običajno nam vzporejanje po korakih ne da dobrih rezultatov
    - Bolje je algoritem v celoti razmisliti na novo - če želimo hitrost, si ji moramo zagotoviti sami z vzporednim programiranjem

Uvod

- Primer vzporejanja
  - Razdelimo na *p* enot, vsaka izračuna svoj delež
  - Na koncu je treba prispevke sešteti
  - Dva načina
    - master (p=0) dobi vrednost od vsakega posebej
    - Drevesno pokaži na primeru za p = 8 in p = 1024

```
sum = 0;
for(i = 0; i < N; i++)
{
    x = heavyduty(...);
    sum += x;
}

my_sum = 0;
for(i = my_first; i < my_last; i++)
{
    my_x = heavyduty(...);
    my_sum += my_x;
}
```

```
if( p = 0 )
    sum = my_x;
    foreach (p != 0 )
    {
        other_x = receive_value_from_core(p);
        sum += other_x;
    }
}
else
    send_value_to_master(my_x);
```

Uvod

- Primer vzporejanja
  - Ne moremo pričakovati, da bo tako zvezo prevajalnik odkril sam
  - Bolj verjetno je, da bo imel pripravljen učinkovit programski konstrukt, ki to dela.
    Lahko bi prepoznal zaporedni vzorec kode in ga zamenjal s programskim konstruktom
  - Bolj kot je program zapleten, teže bo prepoznal segmente, ki se jih da vzporediti

```
sum = 0;
for(i = 0; i < N; i++)
{
    x = heavyduty(...);
    sum += x;
}
```

```
my_sum = 0;
for(i = my_first; i < my_last; i++)
{
    my_x = heavyduty(...);
    my_sum += my_x;
}
```

```
if( p = 0 )
    sum = my_x;
    foreach (p != 0 )
    {
        other_x = receive_value_from_core(p);
        sum += other_x;
    }
}
else
    send_value_to_master(my_x);
```

Uvod

- Programiranje vzporednih sistemov
  - Razširitve prevajalnika
    - Težko, ni še uspešnih komercialnih rešitev
  - Razširitve sekvenčnih programskih jezikov
    - Bolj konzervativno, dodane so funkcije za ustvarjanje in zaključevanje procesov, njihovo sinhronizacijo in komunikacijo med njimi
    - Zelo popularna rešitev
  - Dodajanje vzporedne programske plasti
    - spodnja plast predstavlja paralelno računsko jedro, zgornja plast nadzira uporabo teh jeder
    - GPU
  - Izdelava vzporednih programskih jezikov
    - OCCAM, C*, High Performance Fortran
    - Dodajanje paralelnih konstruktov k obstoječim programskim jezikom

Uvod

- Kaj je vzporedni računalnik?
  - Je računalniški sistem z več procesorji, ki podpira paralelno programiranje.
  - Vzporedne računalnike delimo v dve veliki skupini:
    - Sistemi s skupnim pomnilnikom
      - večprocesorski sistemi
      - Procesori komunicirajo preko skupnega pomnilnika
    - Sistemi s porazdeljenim pomnilnikom
      - večračunalniški sistemi
      - sestavljeni so iz množice računalnikov in povezovalnega omrežja
      - računalniki komunicirajo med seboj z izmenjavanjem sporočil
  - Vzporedno programiranje
    - Programiranje v jeziku, ki omogoča eksplicitno označevanje delov postopka, ki se lahko izvajajo istočasno na različnih procesorjih

Uvod

- Sočasni sistemi
  - V vsakem trenutku se lahko izvaja več opravil
- Vzporedni sistemi
  - Več opravil tesno sodeluje pri reševanju problema
  - procesorji so si blizu, močno povezani, običajno komunicirajo preko skupnega pomnilnika
- Porazdeljeni sistemi
  - Program sodeluje z drugimi programi pri reševanju problema
  - lahko daleč narazen, niso tako močno povezani, zagnanih je več neodvisnih programov

- Primerjava
  - Vzporedni in porazdeljeni sistemi so sočasni
  - Več-opravilni operacijski sistem je sočasen, saj lahko hkrati izvaja več opravil, tudi na eno-jedrnem sistemu

Uvod

- Razvoj superračunalništva
  - Termin superračunalnik se prvič pojavi z razvojem računalnika Cray-1, 1976, 10 mio. USD, Los Alamos National Laboratory
  - Do konca 70. let jih imajo že tudi pomembne firme v industriji (nafta, avtomobilska industrija)
  - Do konca 80. let se pojavijo v poslovnem svetu
    - Hitrejši izračuni podjetjem zagotovijo konkurenčno prednost
    - Manj poskusov pomeni cenejši razvoj
    - Več patentov zaradi hitrejšega razvoja zdravil

Uvod

- Razvoj superračunalništva
  - Prvi superračunalniki so bili vzporedni računalniki
    - Konstantna (visoka) cena, počasno izboljševanje karakteristik
  - Današnji superračunalniki so porazdeljeni računalniški sistemi
    - Masovna proizvodnja, hitrejši razvoj novih zmogljivejših izdelkov
    - Veliko cenejši sistemi, postopno nadgrajevanje, …

Uvod

- Razvoj superračunalništva
  - Slovenija
    - CONVEX SPP1000/XA-64
      - 64 procesorjev, 6,19 GFLOPS
      - 8 mio. USD, IJS, 1992

- Mreža delovnih postaj
  - današnja procesorska jedra zmorejo po nekaj 10 GFLOPS

Uvod

- Razvoj superračunalništva
  - Moderni paralelni računalniki
    - Standardni računalnik v inštitutih: DEC VAX 11/780, 1 MFLOPS
    - 1981: Cosmic Cube: 64 Intel 8086 procesorjev (XT), 5 – 10 MFLOPS, za 1/2 cene VAXa
    - 1986: Connection Machine, Thinking Machine Corporation, 1 CPU, veliko ALU enot
    - 1994: Beowulf, NASA, 16 standardnih Intel DX procesorjev povezanih z 10 Mbit Ethernet, Linux
    - 1996: gruča za manj kot 50.000 dolarjev zmore 1 GFLOPS
    - 2013: Tianhe-2a – drugi največji superračunalnik ima vgrajene procesorje Intel Xeon Phi
    - 2016: Sunway TaihuLight –samo procesorje s po 24 jedri
    - 2019: Sierra, Summit (0,15 EFlops) – grafične kartive nVidia

Uvod

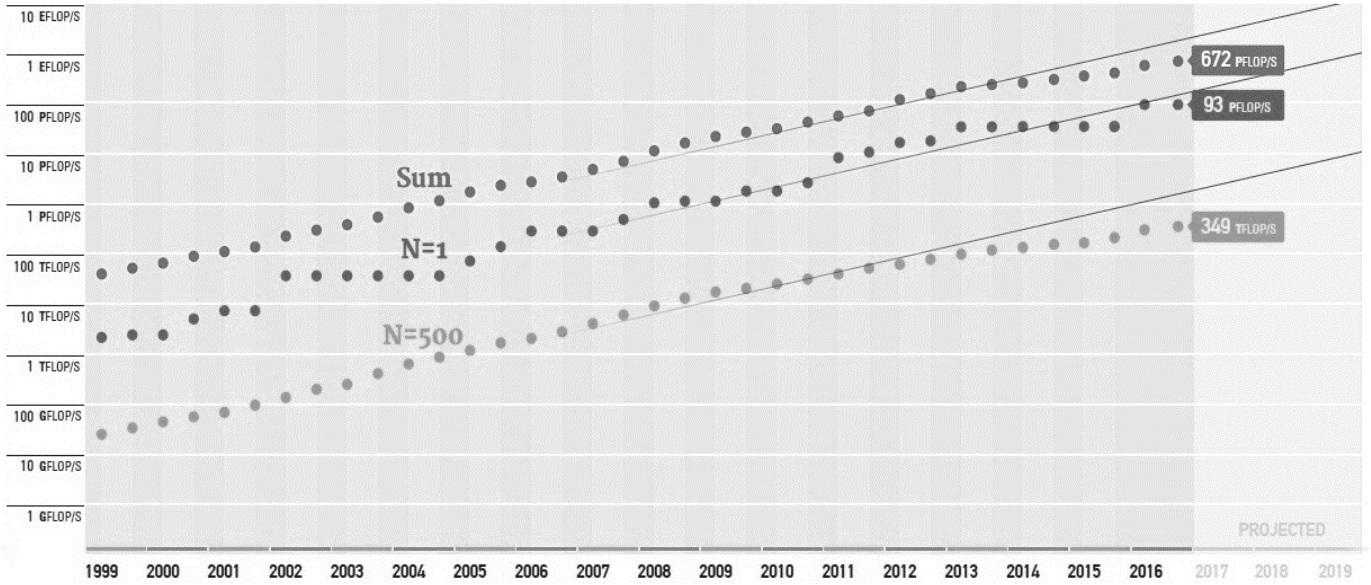
- Superračunalniki:
  - http://www.top500.org, junij 2019
  - 1. Summit,
    - Oak Ridge, USA
    - 2,4 mio jeder, 2,8 PB pomnilnika, 200 PFLOPS
    - 10 MW
  - 2. Sierra
    - USA
    - 1,5 mio jeder, 1,3 PB pomnilnika, 125 PFLOPS
    - 7,4 MW
  - 2. Tianhe-2 (MilkiWay-2)
    - Guangzhou, Kitajska
    - 3,12 mio jeder, 1 PB pomnilnika, 34 PFLOPS
    - 17,8 MW + 24 MW (hlajenje) - zmogljivost HE Blanca
    - 48.000 Intel Xeon Phi
  - 6. Piz Daint
    - Swiss supercompting centre
    - 0,36 mio jeder, 0,34 PN pomnilnika, 27 PFlops
    - 2,2 MW, Cray
  - 9. SuperMUC
    - Munchen
    - 0,3 mio jeder, 0,7 PB pomnilnika, 26 PFLOPS
    - Lenovo

Uvod

- > 500. Ljubljana Supercomputing Center Adria (LSC Adria)
  - Turboinštitut, Ljubljana, Slovenija
  - 4096 jeder, 0,0367 PFLOPS
  - Poraba: 162,5 kW
- Slovenija 2018 – , projekt RIVR

Motivation

- Ever growing need for enormous computing and data processing resources



Source: A CLOSER LOOK AT 2016 TOP 500 SUPERCOMPUTER RANKINGS, https://www.nextplatform.com/2016/11/14/closer-look-2016-top-500-supercomputer-rankings/

Motivation: huge computer resources

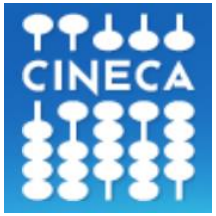
Source: top500.org, November 2019

Motivation: huge computer resources

- Europe is awakening
- EuroHPC initiative
  - Three pre-exascale systems by 2020/21 (>200 Petaflop/s)
  - Five petascale systems by 2020/21 (~10 Petaflop/s)
  - Two exascale systems in 2022/23
  - Post-exascale system afterwards

Motivation: EuroHPC JU – systems

- Finland
  - LUMI
  - Pre-exascale (>200 Petaflop/s)
  - Location: Kajaani
- Italy
  - Leonardo
  - Pre-exascale (>270 Petaflop/s)
  - Location: Bologna
- Spain
  - MareNostrum 5
  - Pre-exascale (>200 Petaflop/s)
  - Location: Barcelona



Motivation: EuroHPC JU – systems

- Bulgaria
  - PetaSC
  - Petascale (~5 Petaflop/s)
  - Location: Sofia
- Czech Republic
  - EURO_IT4I
  - Petascale (~13 Petaflop/s)
  - Location: Ostrava
- Luxembourg
  - MeluXina
  - Petascale (~10 Petaflop/s)
  - Location: Bissen
- Portugal
  - Deucalion
  - Petascale (~10 Petaflop/s)
  - Location: Minho

Motivation: EuroHPC JU – systems

- Slovenia is going Petascale!
- VEGA
  - Performance: ~10 Petaflop/s
  - Location: Maribor
  - Lead organization: Institute of Information Sciences Maribor
  - Total budget: ~30 million €
  - Collaborators: IZUM, UM, FIŠ, SLING
- Goals
  - Upgrade existing HPC capabilities
  - Provide infrastructure for open research data
  - Data storage for Slovenian R&D
  - Offer computational capacities to industry
  - International cooperation

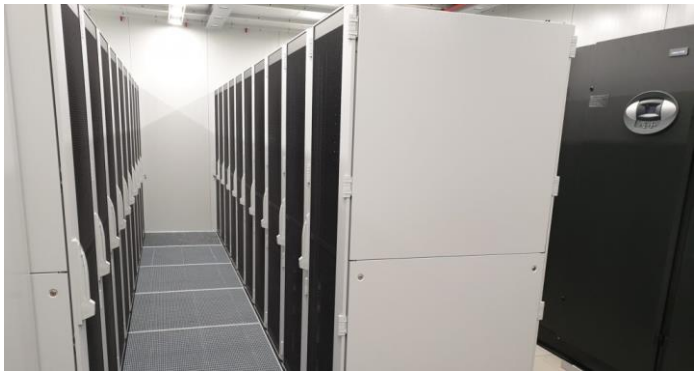
HPC RIVR

IZUM
Institute of Information Science, Maribor, Slovenia

Motivation: HPC Maister@UM

- Prototype system
- Testing in progress
- Configuration
  - ~220 Tflop/s
  - 76 CPU compute nodes with 4864 cores
    - 2 x AMD EPYC 7501, 512GB RAM, 2TB SSD
    - 48 + 28 (connected with Infiniband)
  - 6 GPU compute nodes with 24 GPUs
    - 2 x Intel Xeon Gold 6128, 512GB RAM, 2TB SSD, 4x nVidia V100
  - 3 general purpose servers with 192 cores
  - 3 SSD servers with 140 TB of fast storage
  - Infiniband HDR100 and Ethernet 100Gb/s
- HPC Trdina@FIŠ
  - Small system for educational purposes

Motivation: HPC Vega@IZUM



- Full system (~10 Pflop/s)
  - Deployed at the end of 2020
  - Energy consumption ~1 MW total
  - Computational power

    CPU: GPU = 50: 50
    - > 600 CPU compute nodes with ~40.000 cores
    - > 30 GPU compute nodes with ~120 Nvidia Tesla V100
  - ~30 general purpose servers
    - Login nodes, front ends, databases, data transfer, virtual machines, …
  - ~4PB of SSD storage and ~30PB of HDD storage
  - Network: WAN 500 Gb/s, LAN 100 Gb/s low latency

Motivation: NSC@IJS

- Hardware
  - 1984 cores
    - Vozlišča: 8 x 32 jeder + 27 x 64 jeder
  - 16 GPUs NVidia Tesla Kepler 40
  - 9216 GB RAM
  - LAN
    - 1 Gb/s, 10 Gb/s for storage
    - InfiniBand 56 Gb/s
  - WAN
    - 10 Gb/s
- Software
  - Middleware NorduGrid ARC
  - Batch system: SLURM
  - OS Fedora
    - specific SW in runtime environments
    - Light virtualization (Singularity HPC)

Paralelna strojna oprema

- Ozadje
  - Arhitektura: von Neumannov model
  - Operacijski sistemi: procesi, večopravilnost, niti
- Izboljšave von Neumannovega modela
  - Predpomnilnik
  - Navidezni pomnilnik
  - Vzporejanje ukazov
  - Niti v strojni opremi

von Neumannova arhitektura

- Glavni pomnilnik, CPE (procesor, jedro), povezava
  - CPE = ALE + kontrolna enota + registri
  - povezava (vodilo)
  - delovanje:
    - prevzem + analiza + prenos operandov + izvajanje + shranjevanje
- von Neumannovo ozko grlo
  - CPE in pomnilnik ločena
  - vodilo je ozko grlo

Operacijski sistemi

- Operacijski sistem
  - Zaganja procese
  - Proces sestavlja:
    - primerek izvršljivega programa,
    - rezerviran del pomnilnika (program, sklad, kopica),
    - povezava na dodeljene vire (datoteke),
    - varnostne informacije (dovoljenja za dostop do opreme),
    - stanje procesa (ready, waiting, stopped, …), …
- Večopravilni operacijski sistemi
  - vsak proces se izvaja v časovni rezini,
  - če za neko opravilo OS čaka na vir, lahko tačas izvaja drugo,
  - nitenje omogoča, da proces razbijemo na več neodvisnih opravil,
  - preklapljanje med nitmi je veliko hitrejše kot med procesi, saj si le-te veliko stvari delijo (rabijo svoj programski števec, sklad)

Predpomnilnik

- Predpomnilnik
  - Ne prenaša se beseda za besedo ampak bloki besed (predpomnilniški bloki)
  - Če besede ni, procesor vseeno čaka
    - Manj pogosto, izkorišča lokalnost
      - prostorska (sorodni podatki so blizu skupaj)
      - časovna (bližnje podatke potrebujemo v bližnjem času)
  - primer

```
float z[1000];
sum = 0;
for(i=0; i<1000; i++)
    sum += z[i];
```

  - Večstopenjski predpomnilniki

Predpomnilnik

- Zadetek v predpomnilniku
- Zgrešitev v predpomnilniku
  - Kopiranje iz glavnega pomnilnika
  - Kam ? (zbrišemo najstarejši zapis)
  - Kaj zamenjamo
    - Asociativni (kamorkoli)
    - Set-asociativni (nekaj možnih lokacij)
    - Direktni (točno določene lokacije)
- Branje in pisanje
  - Neskladnost - preprečevanje
    - Ob pisanju v predpomnilnik se vpiše tudi v glavni pomnilnik (pisanje skozi, ang. write-through)
    - Zapis samo v predpomnilnik, postavitev umazanega bita. Ko je treba predpomnilniški blok zamenjati, se zgodi prepis v glavni pomnilnik (pisanje nazaj , ang. write-back)

Predpomnilnik

- Množenje vektorja in matrike

```
double A[MAX][MAX], x[MAX], y[MAX];

for(i=0; i<MAX; i++)
    for(j=0; j<MAX; j++)
        y[i] += A[i][j]*x[j];

for(j=0; j<MAX; j++)
    for(i=0; i<MAX; i++)
        y[i] += A[i][j]*x[j];
```

  - Poglejmo kako je z zgrešitvami (samo matrika A)
    - Primer: MAX = 4, velikost predpomnilnika je 4
    - Pomnilnik je običajno organiziran po vrsticah, v drugem primeru je en kup zgrešitev

Navidezni pomnilnik

- Pomanjkanje glavnega pomnilnika
- Lokalnost
- Namesto o blokih se govori o straneh
- Shema pisanje-nazaj na straneh, diski so prepočasni za pisanje skozi
- Za razliko od predpomnilnika, ki ga nadzira HW, navidezni pomnilnik nadzira OS

- Kot programerji nimamo kaj dosti vpliva na delovanje predpomnilnika in navideznega pomnilnika, moramo pa ga upoštevati

Vzporejanje ukazov

- Paralelizem na nivoju ukazov se deli na
  - cevovodne rešitve, kjer so funkcijske enote urejene v stopnje
  - hkratno izvajanje več različnih ukazov v programu

  - Cevovod
    - Izdelovanje avtomobilov
    - Na enem ukazu se ne pozna, če jih je več, je razlika očitna – prejšnji primer vsote – 1 vsota ali 100 vsot do s-kratno povečanje, če je s stopnja cevovoda, hitrost določa najpočasnejša stopnja
    - Primer:
      - `x[i] = 9,87e4, y[i]=6,54e3: z[i] = x[i]+y[i]`
      - Branje, primerjava eksponentov, poravnavanje eksponentov, seštevanje, normalizacija, zaokroževanje, shranjevanje (7 stopenj)
      - Koliko urinih ciklov traja izvajanje zanke (ena stopnja en urin cikel):

```
for(i=0; i<1000; i++)
        z[i] = x[i] + y[i]
```

Vzporejanje ukazov

- Hkratno izvajanje različnih ukazov v programu
  - Več enakih enot, recimo dva seštevalnika v plavajoči vejici
  - for(i=0; i<1000; i++)
    - z[i] = x[i] + y[i]
    - Prvi seštevalnik: 0, 2, 4, 6, … , drugi: 1, 3, 5, 7, …
  - Dodeljevanje
    - statično (ob prevajanju) in
    - dinamično (med izvajanjem)
      - Pri dinamičnem govorimo o superskalarnih procesorjih (v sekvenčni niti najde kaj lahko vzporeja)
      - Procesor sam najde ukaze, ki se lahko izvajajo paralelno – špekulativno izvajanje

```
z = x + y;
if( z > 0)
   w = x;
else
   w = y;
```

- Hkrati preverja pogoj if in priredi w=x. Če se zmoti, mora popraviti.

Niti v strojni opremi

- Večnitenje (angl. multi-threading)
  - Bolj grobo zrnato vzporejanje, kot na nivoju ukazov
  - Ni treba iskati vzporednosti znotraj iste niti
    - Kdaj ga tudi ni mogoče najti

```
f[0] = f[1] = 1;
for(i=2; i<=n; i++)
    f[i] = f[i-1] + f[i-2];
```

  - Če se ena od niti ustavi (čaka na IO), sistem zažene drugo nit
  - Sistem mora podpirati hitro preklapljanje med nitmi
    - Fino zrnato nitenje
      - Preklapljanje po vsakem ukazu in preskakovanje ustavljenih niti
      - Prednost: procesor ne čaka,
      - Slabost: preklapljamo tudi niti, ki bi se lahko izvajale brez zaustavitev
    - Grobo zrnato nitenje
      - Zamenjevanje samo čakajočih niti,
      - Prednost: manj pogosti preklopi,
      - Slabost: procesor lahko občasno čaka, preklapljanje niti je počasnejše in povzroča zakasnitve
    - Simultano večnitenje:
      - Variacija fino zrnatega nitenja
      - Več niti lahko uporablja hkrati več funkcijskih enot v superskalarnem procesorju