Daudzprocesoru sistēmas

Kāpēc?

1. Prasības pēc arvien lielākas veiktspējas
2. CPU arī sniedz arvien lielāku veiktspēju:
   - Ātrākas elementu tehnoloģijas
   - Ražīgākas arhitektūras
     - Lielāki keši
     - Vairākas un ātrākas kopnes
     - Konveijerizācija
     - Superskalārās arhitektūras (vairāki vienādi izpildes mezgli)
3. Tanī pat laikā:
   1. Viens CPU ***dažkārt*** vairs nespēj veikt prasītos uzdevumus apmierinošā laikā
      - Zinātniskos lietojumos
      - Stimulācijās
      - CAD
      - Multimediju lietojumos
   2. Šajos gadījumos ir darīšana ar ļoti lieliem skaitļošanas apjomiem un/vai datu apjomiem

Risinājums

- ***Viens no risinājumiem*** šai prasībai ir:
  - Arhitektūras kurās ***vairāki*** CPU darbojas ***viena uzdevuma*** izpildei.
  - Uzdevums tāpēc ir speciāli pārveidojams (paralelizējams)
- Šādus datorus veido ļoti dažādos veidos bet mēs tikai apskatīsim to klasifikācijas veidus:
  - CPU skaits un sarežģītība.
  - Kopējās (koplietošanas) atmiņas esamība
  - Savienojumu sistēmas topoloģija
  - Savienojumu sistēmas veiktspēja
  - I/O iespējas un iekārtas

Paralēlās programmas

- Paralēlā programmēšana ir paralēlas izpildes programmu izveide, ieviešana un skaņošana ar mērķi izmantot paralēlo skaitļotāju iespējas.
- Paralēlā programmēšana pamatā cenšas sadalīt problēmu mazākās daļās (uzdevumos), izplānot uzdevumu izsniegšanu individuāliem CPU un nodrošināt atsevišķu uzdevumu sinhronizāciju.
- Paralēli programmēt var tikai tādas problēmas kuras ***padodas paralelizācijai***!
- Paralēlās programmas veido divos pamata veidos:
  - Netieši (izmantojot kompilatora un/vai sistēmas resursus uzdevuma sadalīšanai)
  - Tieši (programmētājs anotē programmas tekstu ar domu parādīt kā tā jāsadala)
- Paralēlo programmu veiktspēju iespaido daudzi faktori. Viens no galvenajiem ir sistēmas slodzes līdzsvarošanas sistēmas spēja izmantot visus sistēmas resursus maksimāli efektīvi.
- Komunikāciju starp uzdevuma daļām paralēlās sistēmas veic divos veidos:
  - Izmantojot koplietošanas atmiņu (shared memory)
  - Izmantojot ziņojumu apmaiņu (message passing)
- Lietotnes kas korekti strādā vienprocesora sistēmā ***var nestrādāt korekti*** daudzprocesoru sistēmās!

Daudzprocesoru sistēmas

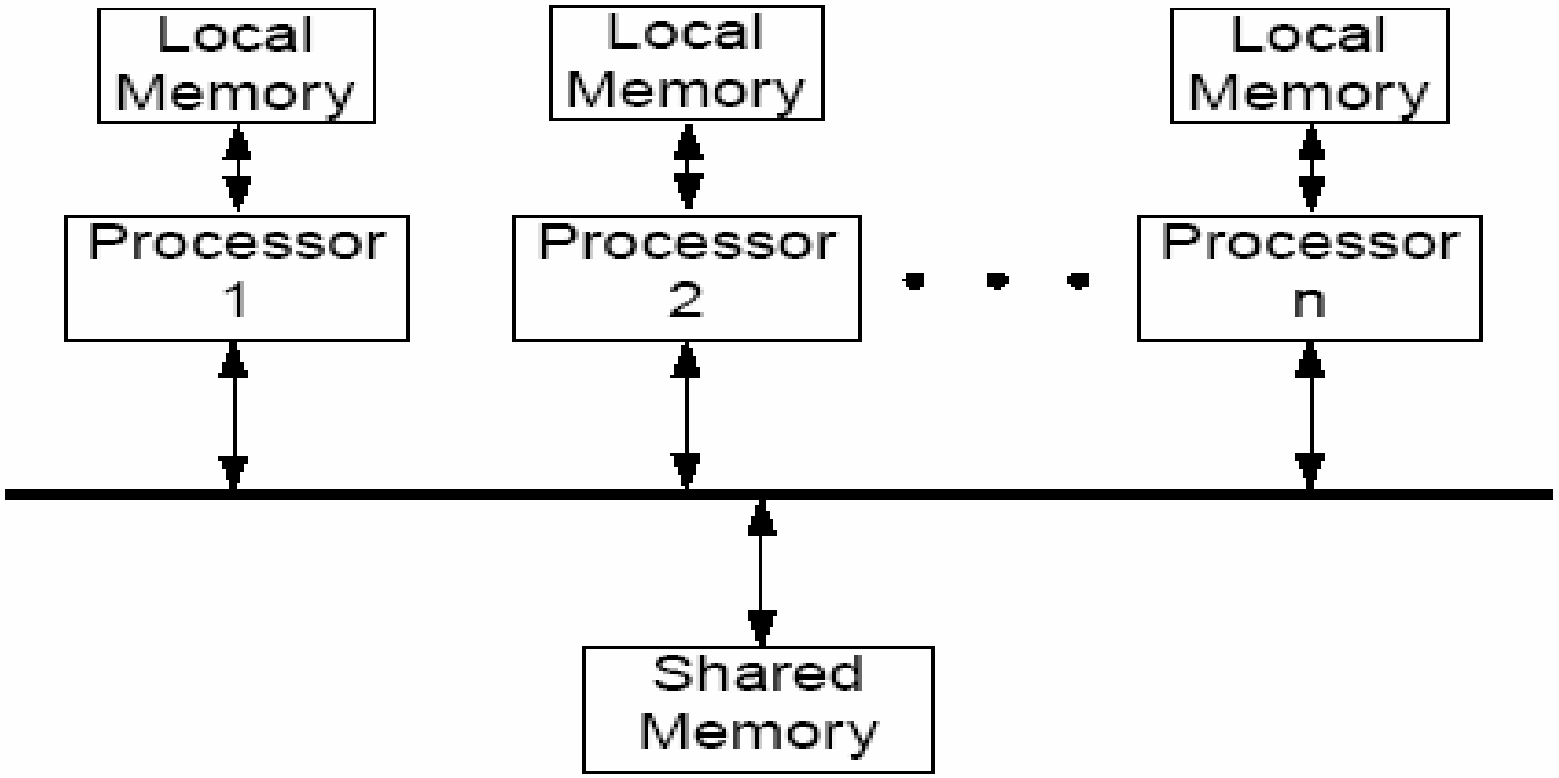
- Paralēlā skaitļošana != multiprogrammu režīms
- Radās dēļ:
  - Veiktspējas prasībām
  - Mērogojamības prasībām
  - Atteikumnoturības prasībām
- Nenovēršamas problēmas
  - Nepietiekams paralēlo darbību apjoms dažādos algoritmos
  - Attālinātas komunikācijas latentums

Daudzprocesoru sistēmu iedalījums

- Flynna (1966) taksonomija:
- – ***SISD***: vienprocesora sistēmas kurās viens CPU izpilda vienu programmu kas atrodas vienā atmiņā (uniprocessors – Von Neuman arhitektūra)
- – ***SIMD***: datu līmeņa paralēlās sistēmas kurās vienu darbību veic ar vairākiem datiem (vektoriem) Pielietojums šodien pamatā multimediju /vektoru procesori /DSP (MMX, HP-MAX), vēsturiski sākotnēji superskaitlotājos

Daudzprocesoru sistēmu iedalījums (atmiņas ziņā)

MIMD skaitļotāji ar kopēju atmiņas lauku (**multiprocessors**)

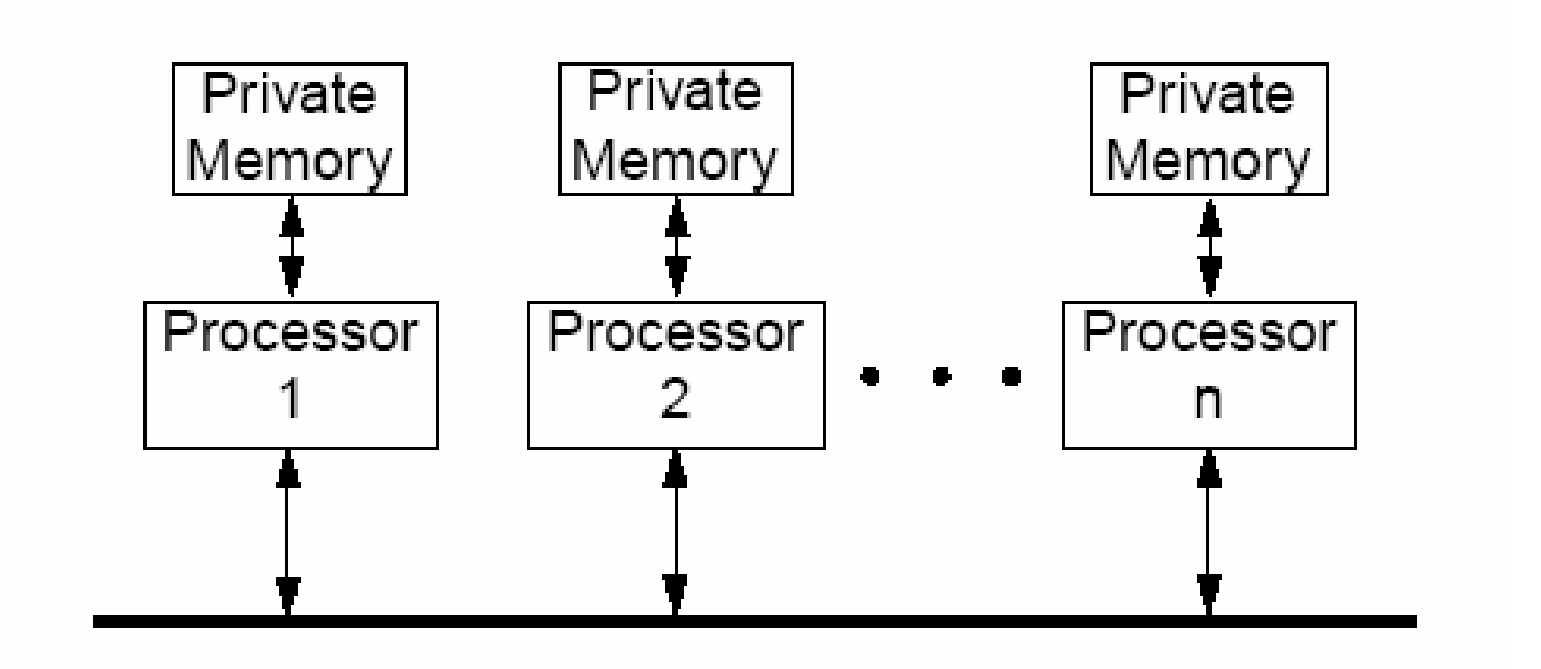


Daudzprocesoru sistēmu iedalījums (atmiņas ziņā)

- Koplietošanas atmiņas arhitektūras iedalās:
  - Uniform memory access (UMA)
  - Non-Uniform memory access (NUMA)
- Dažas daudzprocessoru sistēmas nesatur vienu centrālu koplietošanas atmiņu kurai visi procesori var piekļūt vienādos laikos.
- Visa kopējā atmiņa var būt sadalīta pa atsevišķu procesoru blokiem kā lokālā atmiņa.
- Tomēr jebkurš procesors var piekļūt ne tikai savai lokālai bet arī citu procesoru atmiņām.
- Jebkurā gadījumā ir pieejams ***kopējs globālo fizisko*** adrešu lauks.
- Šāda organizācija tiek saukta par “distributed shared memory”
- Pretējā gadījumā ir SMP “Symmetric Multiprocessing”
- Abos gadījumos paralēlā skaitļošana tiek realizēta izmantojot kopējos mainīgos (shared variables)
- Lielāka CPU skaita gadījumā šādas sistēmas ātri var pārslogot atmiņu un tāpēc SMP neatbalsta ***lielu skaitu*** CPU
- Koplietošanas atmiņas sistēmas prasti ir ***cieši saistītas sistēmas***

Vairākdatoru sistēmas

- MIMD datori ar sadalītu adrešu lauku kur katram procesoram ir sava atmiņas lauka daļa un kurš ***nevar piekļūt*** citu procesoru atmiņai (multicomputers)



Vairākdatoru sistēmas

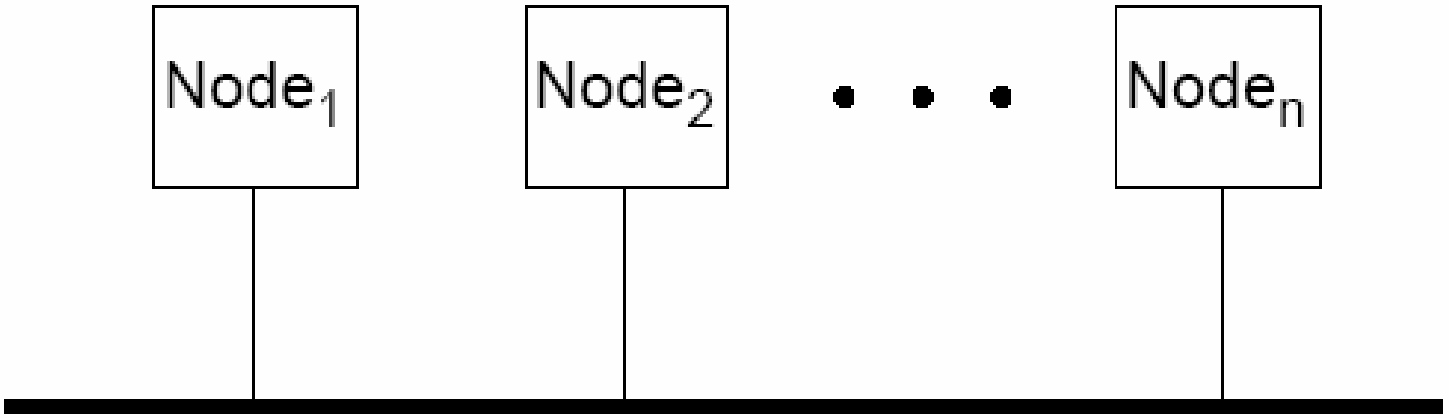
- Komunikācija starp mezgliem ir iespējama tikai izmantojot ***ziņojumus*** kas tiek noraidīti kopējā ***savienojumu tīklā***
- Programmētājam jālieto speciālas komandas un jāveido sakaru kanāli starp programmas daļām (MPI)
- Nav kopējās atmiņas pārslodzes problēmas un tāpēc nav CPU skaita ierobežojuma skatoties no atmiņas problēmu puses
- Pamata ierobežojums ir savienojuma ***tīkls un tā veiktspēja***
- Klasteru sistēmas (***vāji saistītās sistēmas***)
  - HA klasteri
  - LB klasteri
  - HPC klasteri
  - Beowulf klasteri

Savienojumu sistēmas

- Savienojumu sistēmas (SS) ir pamata komponente jebkurai daudzprocesoru sistēmai
- Tā pamatā nosaka kopējās sistēmas veiktspēju un cenu
- Datu plūsmas kas tiek noraidītas SS sastāv no datu un komandu plūsmām
- SS pamata parametri ir:
  - Kopējais joslas platums (biti/sekundē), latentums -?
  - $

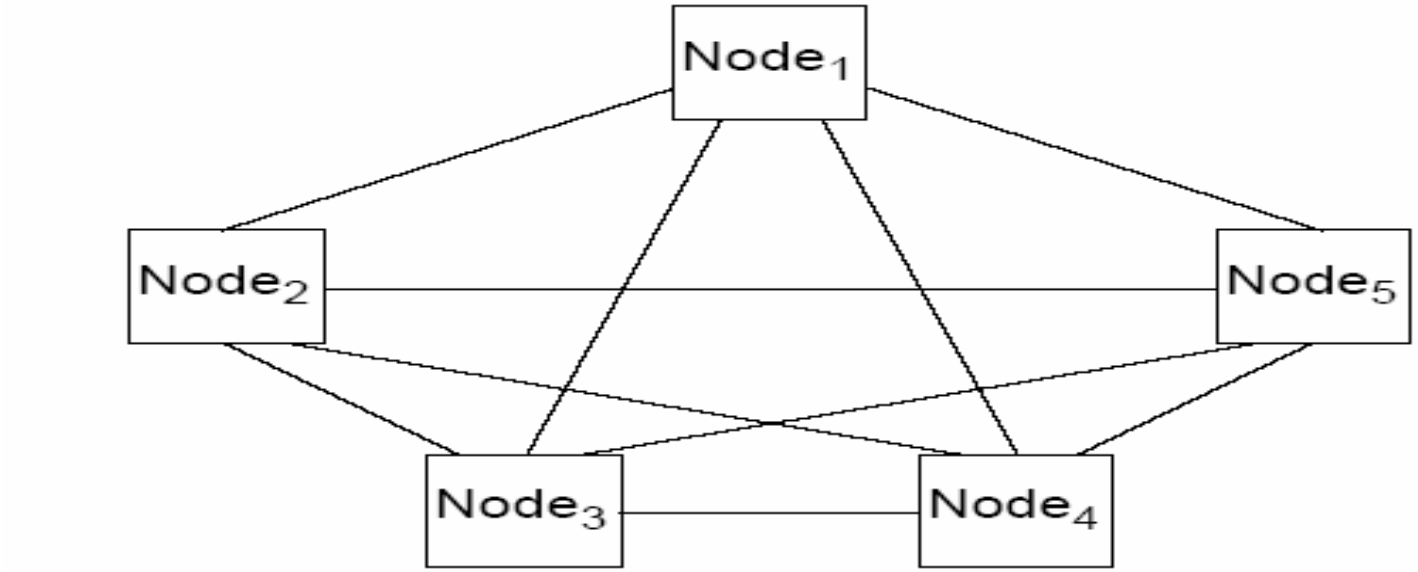
Kopne

- Vienas kopnes sistēmas ir vienkāršas un lētas
- Vienā laika momentā var notikt tikai viena datu apmaiņas darbība
- Joslas platums ir visu mezglu kopējais resurss
  - Veiktspēja ir relatīvi zema
  - Lai garantētu pieņemamu veiktspēju mezglu skaits tiek ierobežots (16 – 20 gab.)



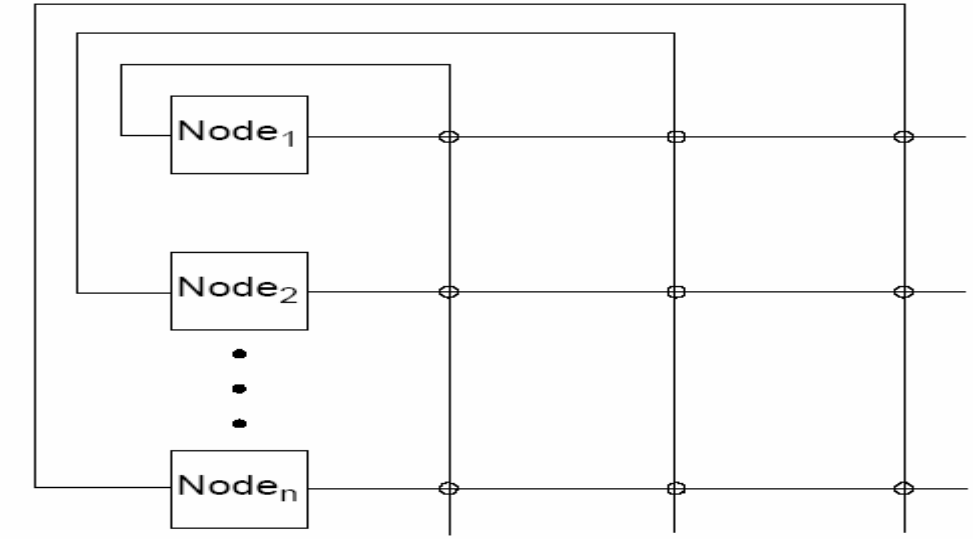
Pilnībā saistītas sistēmas

- Katrs mezgls ir tieši saistīts ar visiem citiem mezgliem
- Komunikācija var notikt paralēli jebkuru mezglu pāru starpā
- Veiktspēja un $ ir ļoti labas
- Palielinot mezglu skaitu $ pieaug ***nelineāri***.



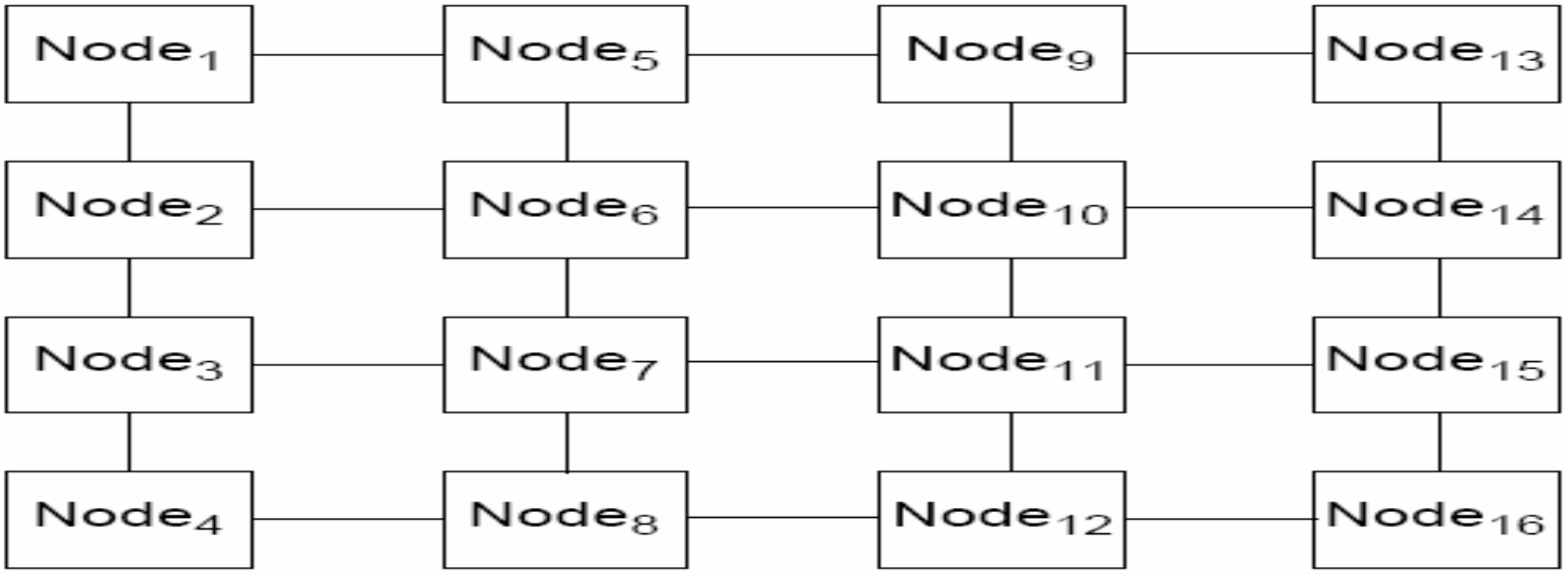
Crossbar

- Crossbar tīkls ir dinamiski maināms komutējot savienojumus
- Crossbar komutators ir pilnībā saistīta sistēma (jebkurš mezgls var tieši sazināties ar jebkuru citu mezglu)
- Salīdzinājumā ar statiski pilnībā saistītu tīklu ir nepieciešams daudzkārt mazāk savienojumu bet daudzkārt vairāk komutatoru
- Liels apjoms komunikācijas var tikt veikts paralēli (lai gan viens mezgls var veikt komunikāciju tikai ar kādu citu vienu mezglu)



Režģtīkli

- Režģtīkli (mesh networks) ir lētāki par pilnībā saistītiem tīkliem un nodrošina relatīvi labu veiktspēju
- Lai noraidītu informāciju no viena mezgla otram ir jāveic maršrutēšana caur citiem mezgliem (sliktākajā gadījumā 2*(n-1) ceļa mezgli n*n režģtīklā).
- Var veidot arī cilindriskus un 3D režģtīklus



Vektoru procesori

- Vectoru procesori savās komandu kopu arhitektūrās satur arī speciālas komandas kas darbojas ar vektoriem
- Vektoru procesori ir SIMD skaitļotāji kuros katru vektoru elementu pāri apstrādā savs izpildes mezgls
- Vairākas datoru arhitektūras ir ieviesušas vektoru komandas izmantojot paralēlās izpildes realizācijas mezglus
- Šādas arhitektūras sauc par vektoru procesoriem
- Vektoru procesori nav daudzprocessoru sistēmas

Kopumā

- Ne vienmēr prasības ir iespējas apmierināt ar viena CPU palīdzību.
- Paralēlās skaitļošanas gadījumā vairāki CPU darbojas viena uzdevuma izpildei.
- Lai lietotu paralēlās sistēmas ir nepieciešamas paralēlas programmas (vai liels skaits paralēli izpildāmu programmu)
- Skaitļotājus var klasificēt pēc vadības un datu plūsmu kopdarbības veidiem SISD, SIMD,MIMD...
- Veiktspēja ko var iegūt no daudzprocesoru sistēmām ir atkarīga no lietotnēm ***nevis CPU skaita***.
- Paralēlā skaitļotāja lietošanas efektivitāte ir atkarīga no paralēlās programmas veida (paralēlās izpildes apjoma, starpprocesu komunikācijas apjoma ...)
- Veiktspēju ļoti iespaido savienojumu siostēmas
- MM lietotnes satur lielu daļu darbību kas var tikt veiktas kā SIMD paralēla izpilde
- Moderno CPU ISA (Pentium, Sparc...) satur SIMD komandas kas apstrādā neliela apjoma vektorus

Mājās

- http://en.wikipedia.org/wiki/Parallel_computing
- http://www.cs.rtu.lv/Pubs/Cipa/Arhit2003/WPad/ARH14_1.doc