

Multiprocesorski sistemi

Paralelne arhitekture i programski modeli

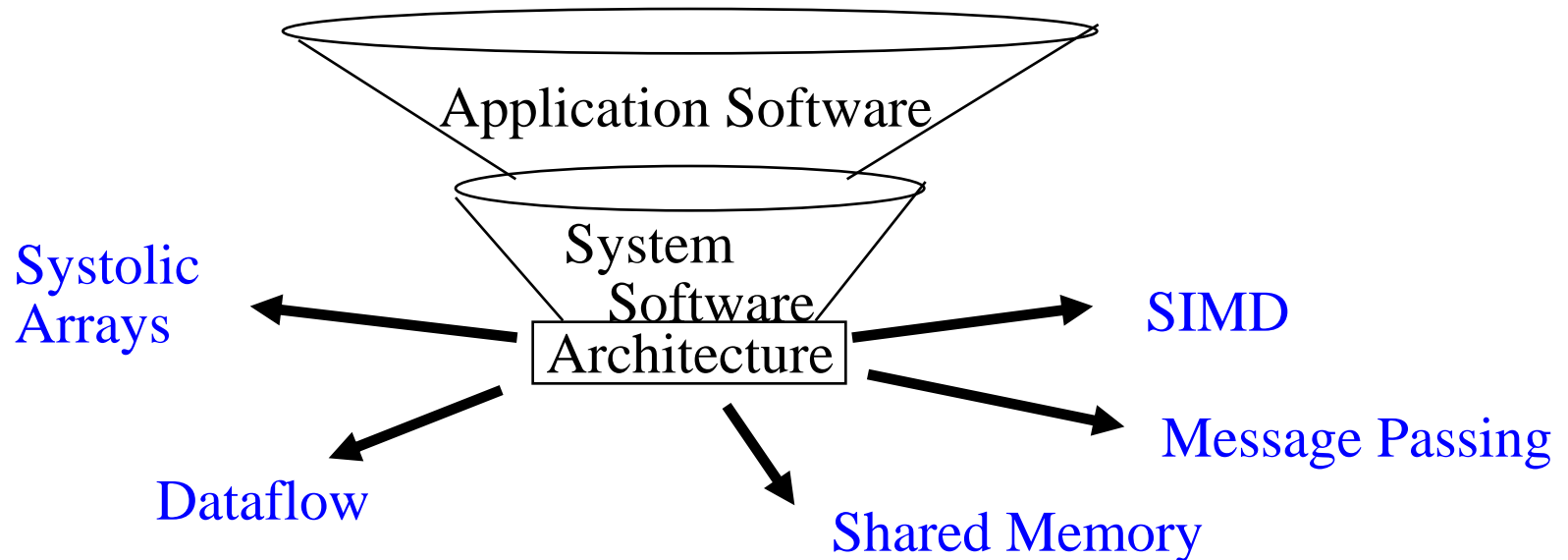
Marko Mišić, Milo Tomašević

13S114MUPS, 13E114MUPS, 13M114MUPS

2024/2025.

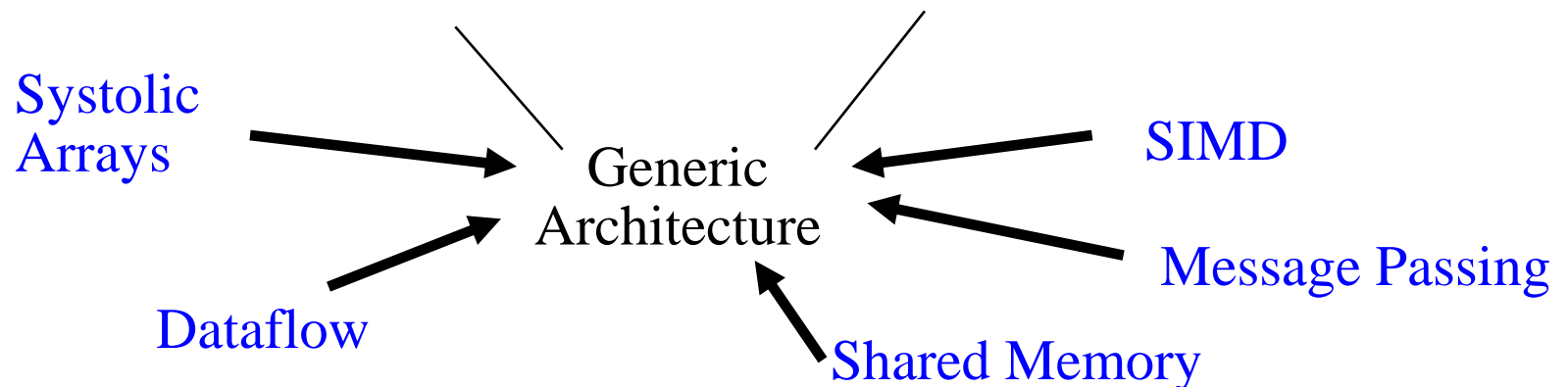
Paralelne arhitekture i programski modeli

- Paralelne arhitekture blisko vezane sa programskim modelima
 - Divergente arhitekture u prošlosti



Paralelne arhitekture i programski modeli

- Konvencionalna arhitektura + arhitektura komunikacije
 - Definicija kritičnih apstrakcija (HW/SW, sistem/korisnik)
 - Organizaciona struktura koja ih implementira
 - Kakva je performansa i cena?
- Isti tehnološki i aplikacioni trendovi utiču na konvergenciju
- Generalna struktura i fundamentalne karakteristike



Paralelne arhitekture

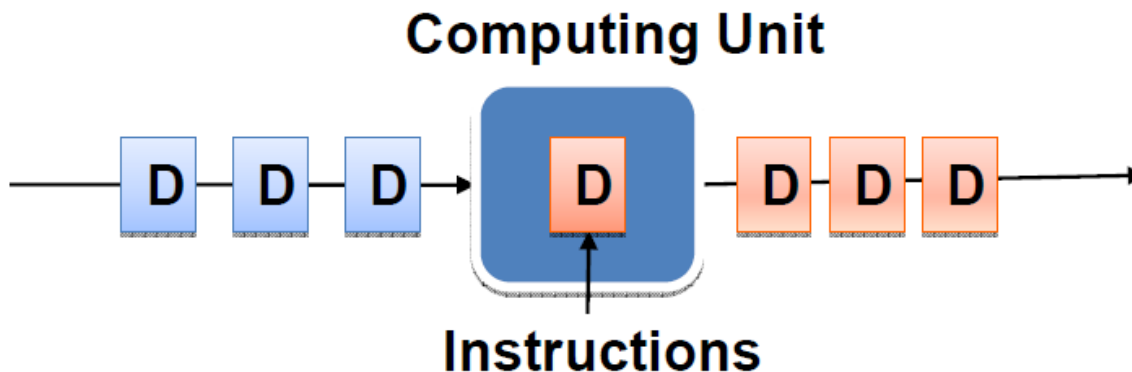
- Flynn-ova klasifikacija (1972)

DS - tok podataka		DS	
		Jedan	Više
IS - tok instrukcija			
IS	Jedan	SISD	SIMD
	Više	MISD	MIMD

- Ponekad teško svrstati konkretnu arhitekturu
- Kuck – ES i IS, skalarni i vektorski

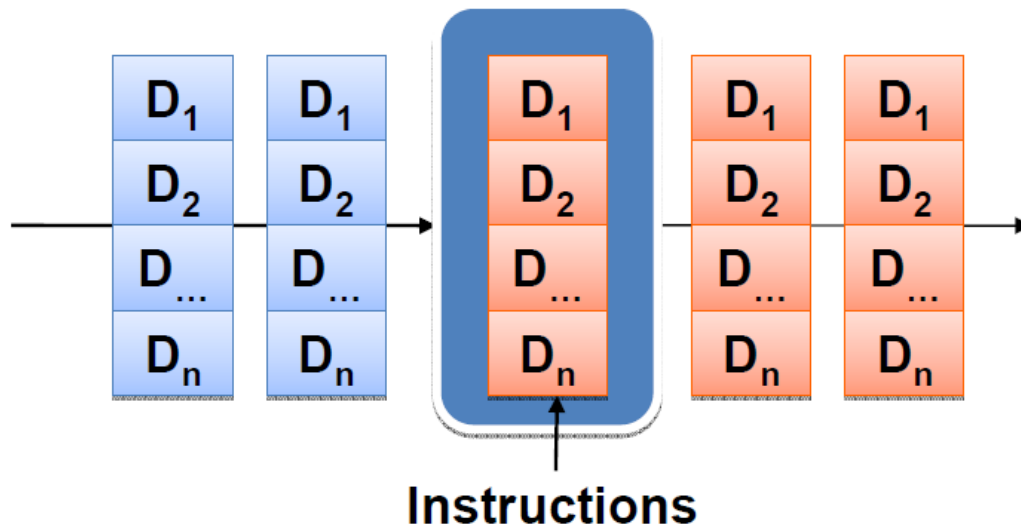
SISD

- Svi podaci u jednoj memoriji
- Jedan procesor izvršava jedan tok instrukcija
- Tipična Von Neumann arhitektura



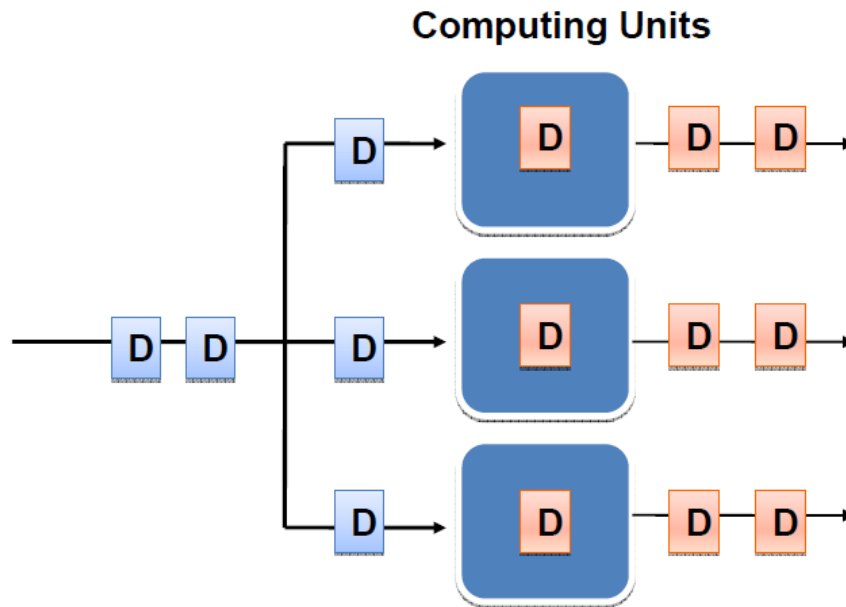
SIMD

- Jedan tok instrukcija nad više tokova podataka
Paralelizam podataka
- Vektorski procesori
Instrukcija se izvršava nad vektorom podataka
- Paralelni SISD
Sinhrono izvršavanje iste instrukcije



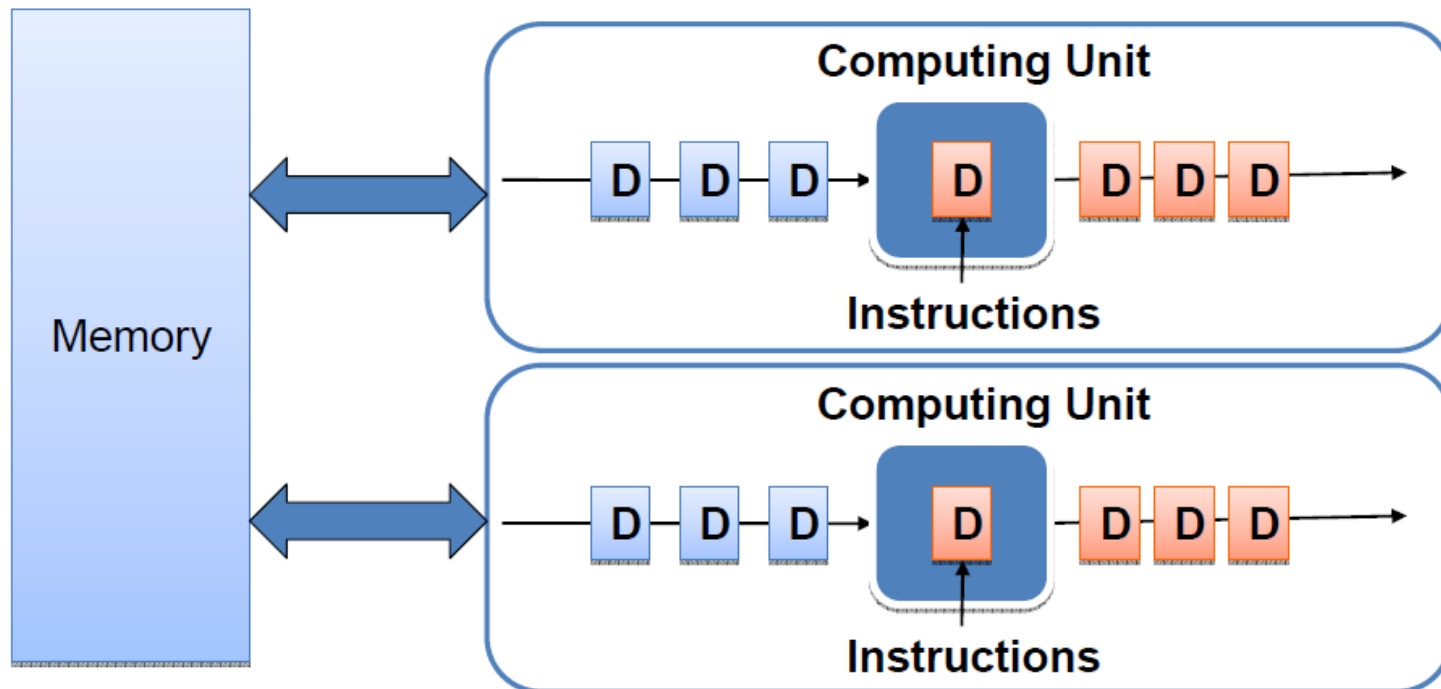
MISD

- Više tokova instrukcija nad jednim tokom podataka
Paralelizam taskova
- Malo realnih primera (npr., sistolni nizovi)
- Smanjuje potreban propusni opseg memorije
- Aplikacije (npr., kriptografija, *3D-raytracing*)



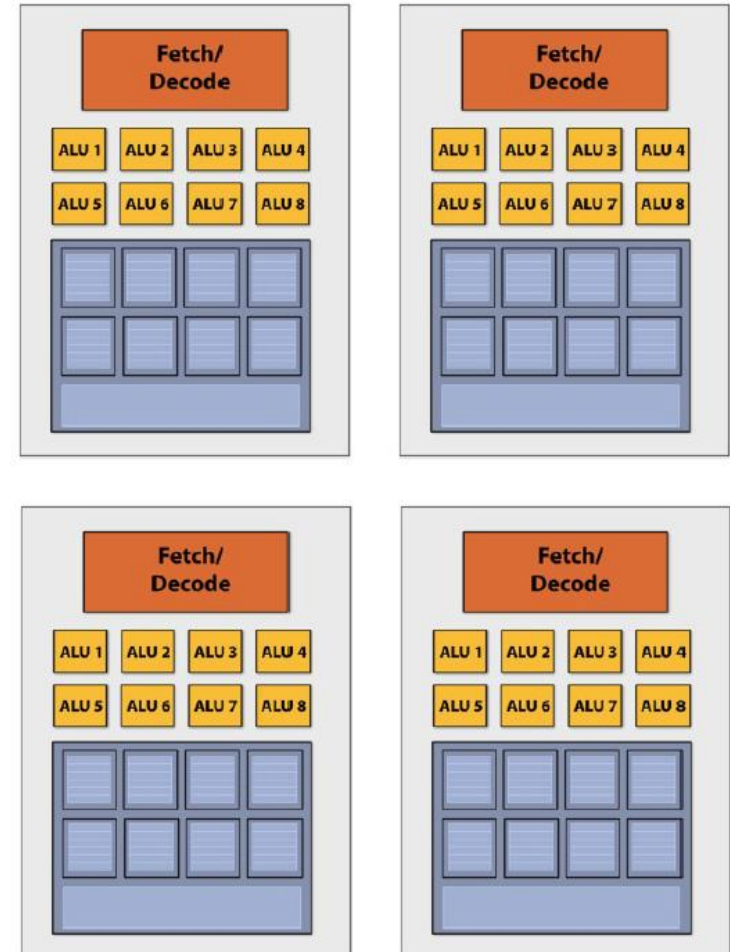
MIMD

- Više tokova instrukcija nad više tokova podataka
- Distribuirani sistemi ili MIMD arhitekture
- Deljeni ili distribuirani memorijski sistem



Savremeni *multicore* CPU

- Hibrid više klasa
- Intel Core i7 (Sandy Bridge)
- SISD (+ ILP)
 - U jednom jezgru
- SIMD
 - 8 ALU po jezgru
 - AVX instrukcije na 256 bita (8x32)
- MIMD
 - 4 nezavisna fizička jezgra
 - 8 logičkih (*hyper threading*)



Novija klasifikacija

○ SPMD

- *Single Program Multiple Data*
- Više autonomnih procesora izvršava isti program nad različitim podacima na različitim mestima
- Za razliku od SIMD nema sinhronizacije na nivou instrukcije i ne zahtevaju se vektorske i SIMD arhitekture

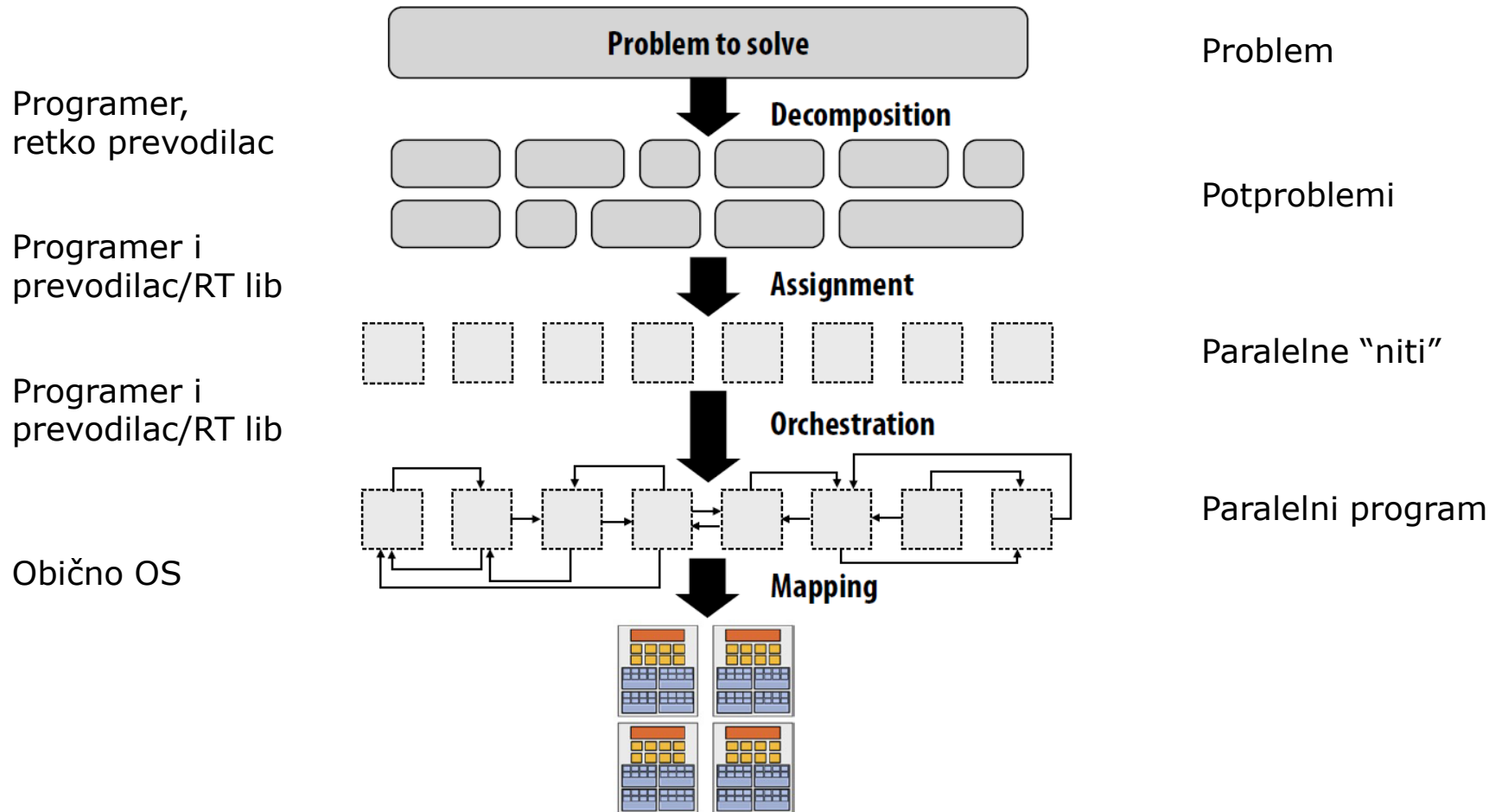
○ MPMD

- *Multiple Program Multiple Data*
- Više autonomnih procesora izvršava bar dva različita programa nad različitim podacima

Programski modeli (1)

- *Skup programskih apstrakcija koje programeru omogućavaju da na uprošćen i transparentan način vidi SW i HW*
 - Jezici i biblioteke koji definišu apstraktni izgled mašine
 - Kooperacija i koordinacija delova programa koji se izvršavaju u paraleli
 - Specificira komunikacione i sinhronizacione operacije
- Kontrola
 - Kako se paralelizam kreira?
 - Koji **poredak** je dozvoljen između operacija?
- Podaci
 - Koji podaci su **privatni**, a koji **deljeni**?
 - Kako se logički deljenim podacima pristupa ili kako se **razmenjuju**?
- Sinhronizacija
 - Koje operacije se koriste za koordinaciju paralelizma?
 - Kakve su **atomične** (nedeljive) operacije?
- Performanse (latencija, propusni opseg)

Programski modeli (2)



Programski modeli (3)

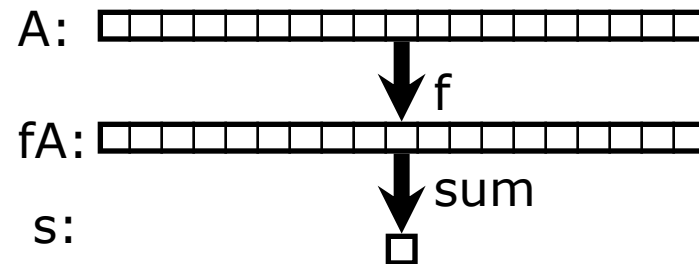
- Npr., izračunati

$$\sum_{i=0}^{n-1} f(A[i])$$

- Pitanja

- Gde se nalazi niz A? U jednoj memoriji? Distribuiran?
- Šta svaki procesor treba da uradi?
- Kako se koordiniraju da se dobije jedan rezultat?

A = niz sa svim podacima
fA = f(A)
s = sum(fA)



Programski modeli (4)

○ Implicitni

- Sekvencijalni model + automatska paralelizacija
- Korisnik ne specificira i ne kontroliše organizaciju izvršavanja i raspored podataka
- Podrška *runtime* biblioteka
- Najlakše za korisnika
- Teško postići efikasne automatske paralelne prevodioce

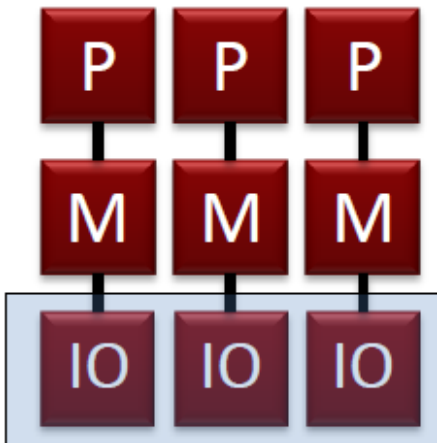
○ Eksplicitni

- Programer odgovoran za paralelizaciju
- Dekompozicija taskova i podataka
- Komunikacija i sinhronizacija paralelnih delova
- Ponekad čak i mapiranje na HW

Programski modeli (5)

- Multiprogramiranje
 - Nema komunikacije na nivou programa
- *Model zajedničke memorija (shared address space)*
 - Zajednički adresni prostor (multiprocesori)
- *Model prenosa poruka (message passing)*
 - Usmereno slanje i primanje poruka (multiračunari)
- *Model paralelne obrade podataka (data parallel)*
 - Koordinacija globalnih, paralelnih operacija nad podacima
 - Karakterističan za SIMD
- *Hibridni modeli*
- *Specifični modeli*

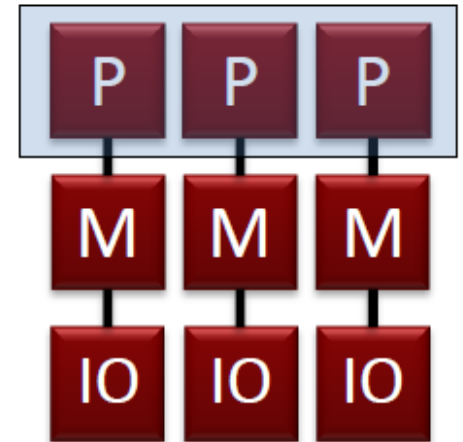
Programski modeli (6)



Prenos poruka



Zajednička memorija



Data parallel

Programski modeli (7)

- Realizuje se kroz korisničke komunikacione primitive na nivou paralelnog jezika, a podržane kroz:
 - HW primitive
 - OS primitive
 - Korisnički SW specifičan za mašinu
- Arhitektura komunikacije
 - Skup komunikacionih operacija, formati, strukture podataka
- Istorijski HW organizacije dosta direktno podržavale model
 - Posledica: divergentne arhitekture
- Danas arhitekture konvergiraju
 - Isti tehnološki aspekti nezavisni od modela
 - Bolje implementacione tehnike
 - Prevodioci, biblioteke i OS kao most ka HW
 - Dosta mašina podržava više programskih modela

Programski modeli (8)

*Abstractions for describing
concurrent, parallel, or
independent computation*

*Abstractions for describing
communication*

*“Programming model”
(way of thinking about things)*

Compiler and/or parallel runtime

*Language or library
primitives/mechanisms*

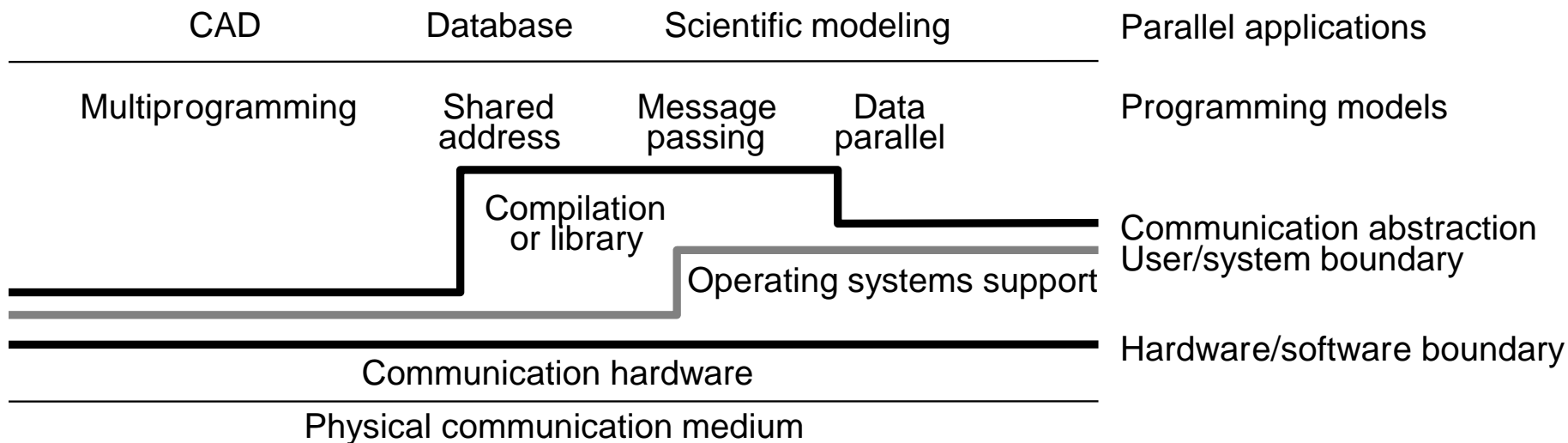
Operating system

OS system call API

Micro-architecture (hardware implementation)

*Hardware Architecture
(HW/SW boundary)*

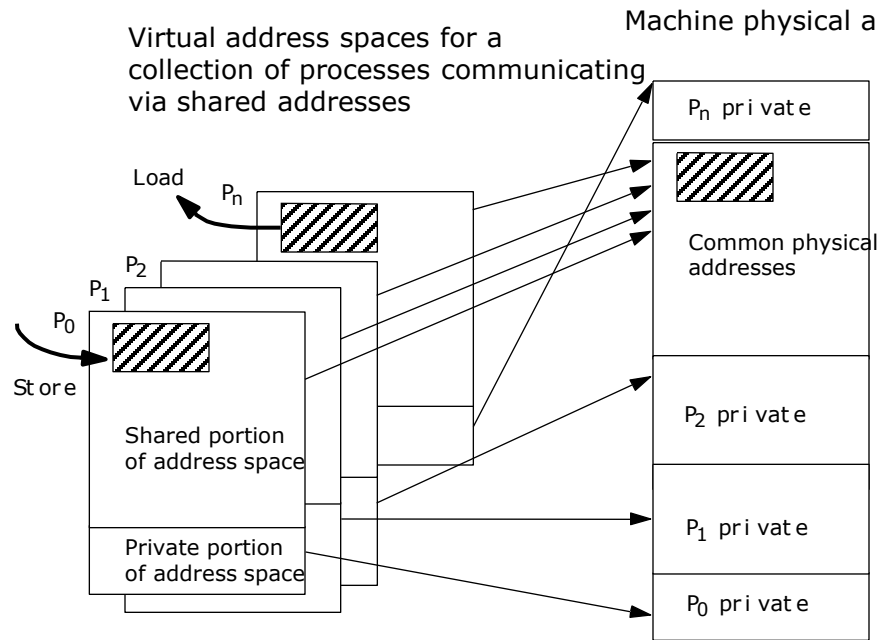
Nivoi apstrakcije u paralelnoj arhitekturi



Model zajedničke memorije

- Jedan od najstarijih, najprostijih i najgeneralnijih
- Velika zastupljenost na svim nivoima sistema
- Proces = adresni prostor + nit kontrole
- Implicitna komunikacija procesa –
upisi i čitanja u zajedničkom adresnom prostoru
- Uobičajene HW R/W primitive (bez intervencije SW)
- Mapiranje iz virtuelnog u fizički prostor
ostvaruje deljivost
 - *User-kernel* ili više procesa
- Upisi u deljivi prostor vidljivi ostalim procesima
- Više tokova kontrole u jednom adresnom prostoru

Model zajedničke memorije



- Mnoge paralelne aplikacije imaju strukturirani SAS
 - Deljivi kod, globalni podaci, zajednički blokovi, globalni *heap*
 - Privatni stek i ostali podaci
 - Npr., paralelne petlje

Model zajedničke memorije

Sumiranje niza A[80000], sistem sa 8 procesora

- 1) $0 \leq P_n \leq 7$
- 2) Sumiranje po procesorima

```
sum[Pn] = 0;
for (i = 10000*Pn ; i < 10000*(Pn+1) ; i++)
    sum[Pn] += A[i];
```
- 3) Sabiranje parcijalnih suma

```
half = 8;
do {
    synch();
    half = half/2;
    if (Pn < half) sum[Pn] += sum[Pn+half];
} while (half != 1) ;
```

Model zajedničke memorije (prednosti)

- Manji komunikacioni “*overhead*” (bez OS)
- Uniforman mehanizam pristupa podacima
 - Nema kopiranja (može po adresi)
- Prirodna nadgradnja sekvencijalnog programiranja
 - Aplikacija izgleda kao *multitasking* na jednoprocesorskom računaru
 - Lakše portovanje postojećeg koda
 - Manje ekstenzije OS
- Pogodan za:
 - Finiju granularnost paralelizma
 - Promenljive i nepredvidljive načina obraćanja podacima
- Olakšava asinhronu komunikaciju
 - Procesi ne moraju ni postojati istovremeno
- Omogućava keširanje
- Može da emulira i druge programske modele (MP)

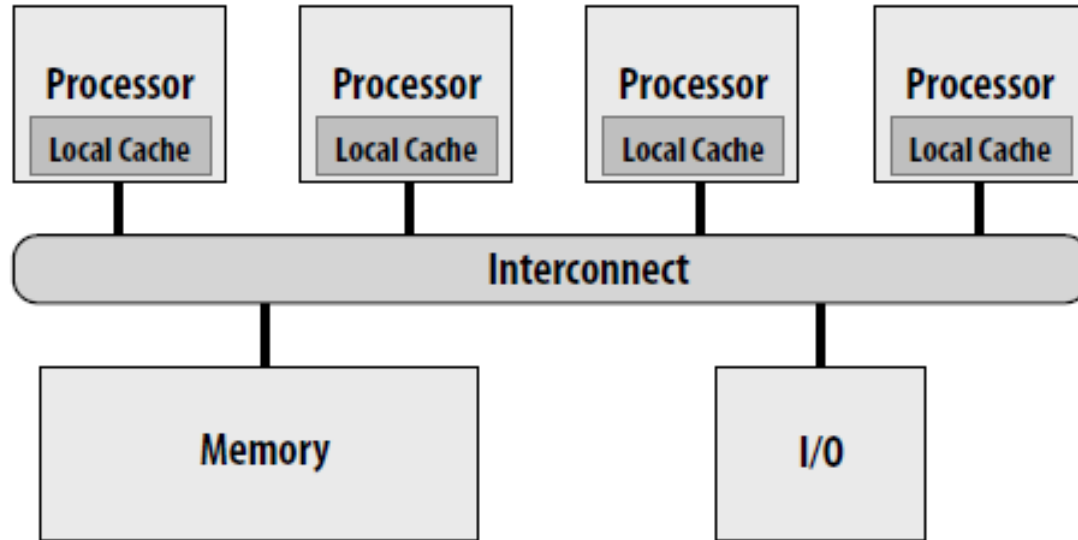
Model zajedničke memorije (nedostaci)

- Posebne atomske operacije za sinhronizaciju
 - zahtevno!
- Lakše za SW, ali složeniji HW koji mora da:
 - obezbedi efikasan pristup deljenim podacima
 - onemogućiti neželjene pristupe
- Zahteva veći propusni opseg memorije
- Implicitnu komunikaciju teže optimizovati
- Teže ostvariti skalabilnost
- Ishod
 - tradicionalni SMP i današnji CMP najuspešnije paralelne platforme
 - ogromno tržište

Model zajedničke memorije

- POSIX niti (Pthreads)
 - Relativno nizak nivo, usluge OS
 - Sistemski pozivi za kreiranje i sinhronizaciju niti
 - Portabilno, ali performanse?
- OpenMP
 - Standardni API zasnovan na direktivama prevodioca
 - Laka paralelizacija petlji
- TBB (*Thread Building Blocks*)
 - Intel, C++ paralelna biblioteka
- CILK
 - Nadskup od C, ugrađene lake niti
- Java niti
 - Objekti u Java jeziku

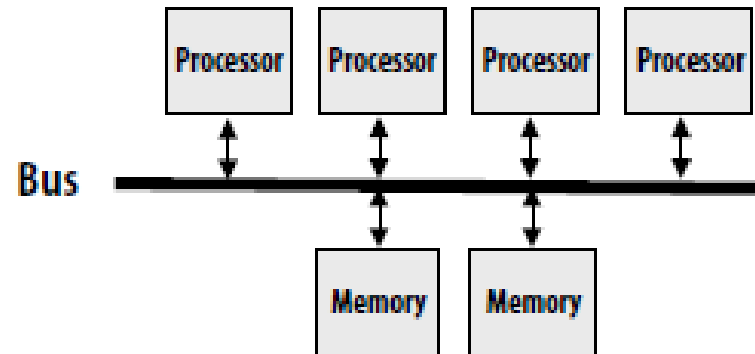
SM arhitekture - UMA



- UMA (*Uniform Memory Access*)
 - Svaki procesor može direktno pristupiti svakoj memorijskoj lokaciji
 - Cena nekeširanog pristupa ista za sve procesore
 - Svaki U/I kontroler može na svaku memoriju (DMA)
 - Za multiprogramiranje ili paralelne aplikacije
 - Skalabilnost (cena, propusni opseg) – ICN?

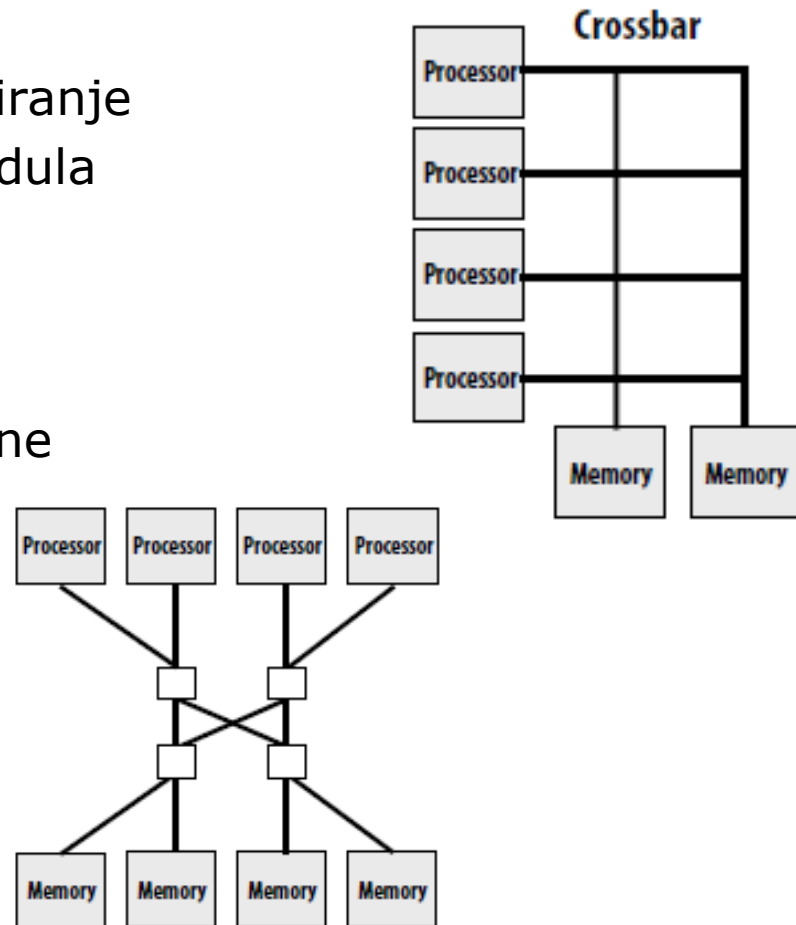
SM arhitekture

- “*Minicomputer*” pristup
 - Zajednička magistrala
 - SMP – simetrični MP
 - Multiprogramiranje i paralelne aplikacije
 - Latencija veća nego kod jednoprocesorskih
 - Manja inkrementalna cena
 - Skalabilnost (do 16)
 - Usko grlo: magistrala
 - Keširanje i koherencija

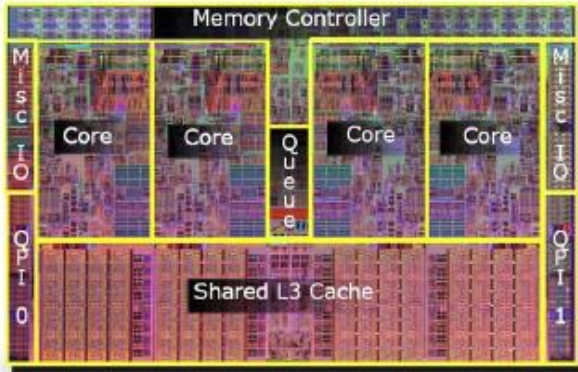


SM arhitekture

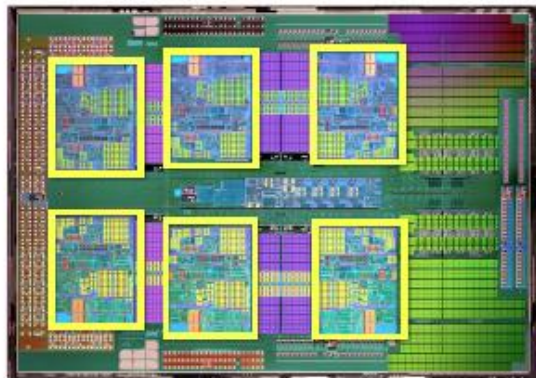
- “*Mainframe*” pristup
 - Namenjene za multiprogramiranje
 - Preklapanje memorijskih modula
 - U/I kanali
 - “*Crossbar*” mreža
 - Nescalabilni po ceni
 - Kompromis - MIN (višestepene sprežne mreže)
 - Veća latencija, manja cena
 - “*Dance hall*”



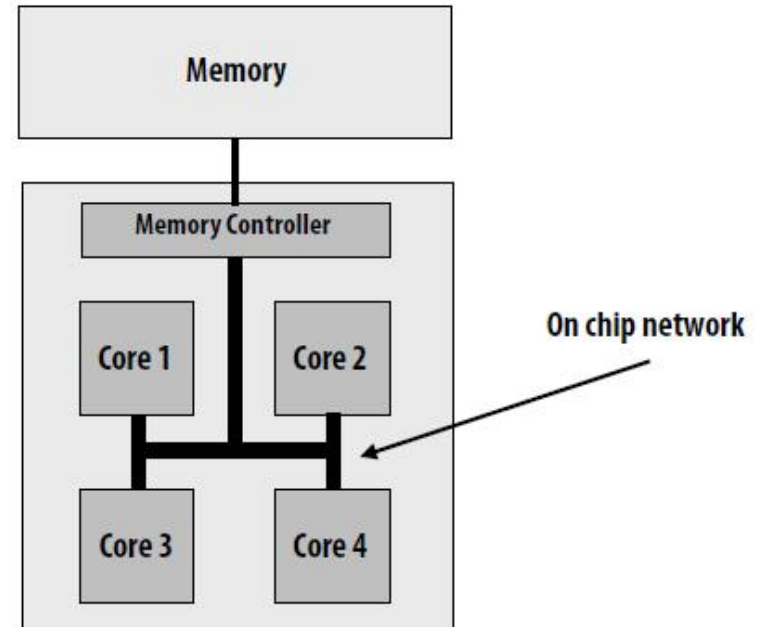
Primeri: x86 CMP



Intel Core i7 (quad core)
(network is a ring)



AMD Phenom II (six core)

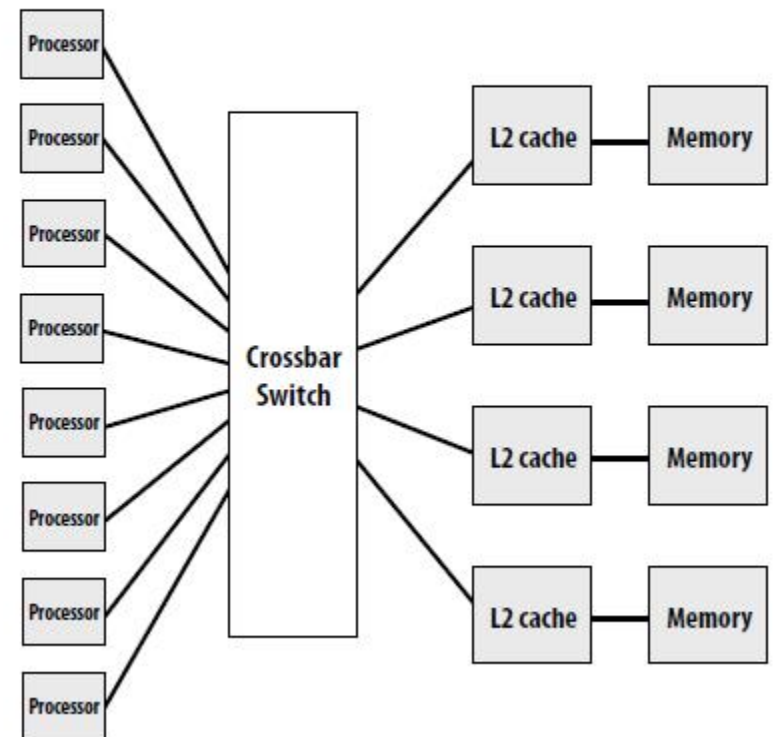


Primer: Sun Niagara 2

Note size of crossbar: about die area of one core



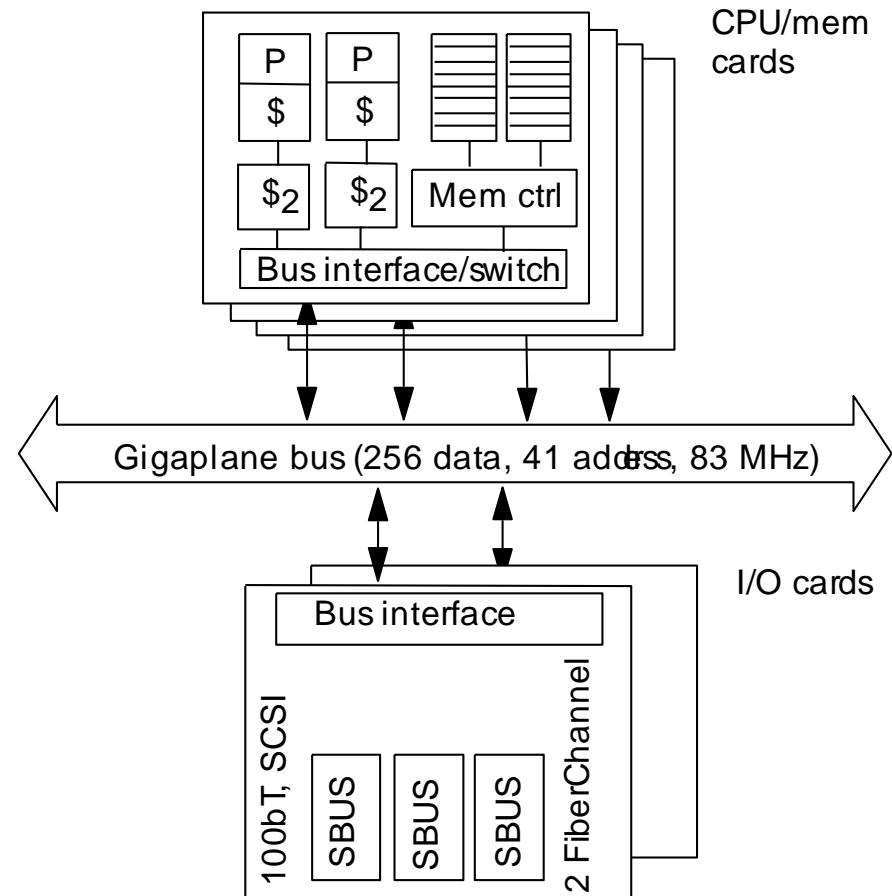
Eight cores



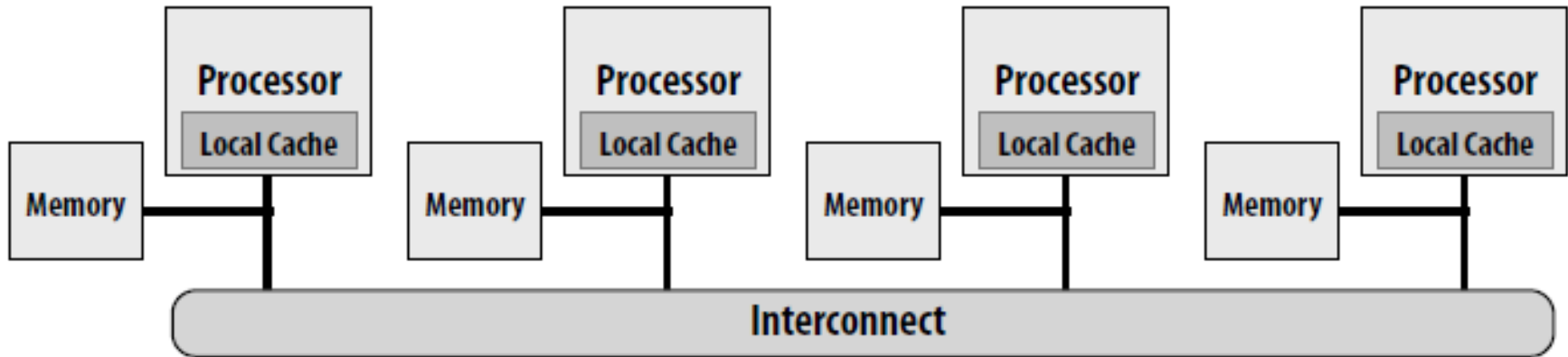
Primer: SUN Enterprise

- Multiprocesorski server
 - Kartice: CPU/memorija ili kompletan U/I
 - Max. 16 kartica
 - CPU/mem kartica = 2 UltraSparc (sa L1, L2) + memorija
 - Magistrala 256-bit, 2.5 Gbit/s
 - Memorija simetrična u odnosu na procesore
 - Veća propusna moć, nešto veća latencija

Primer: SUN Enterprise

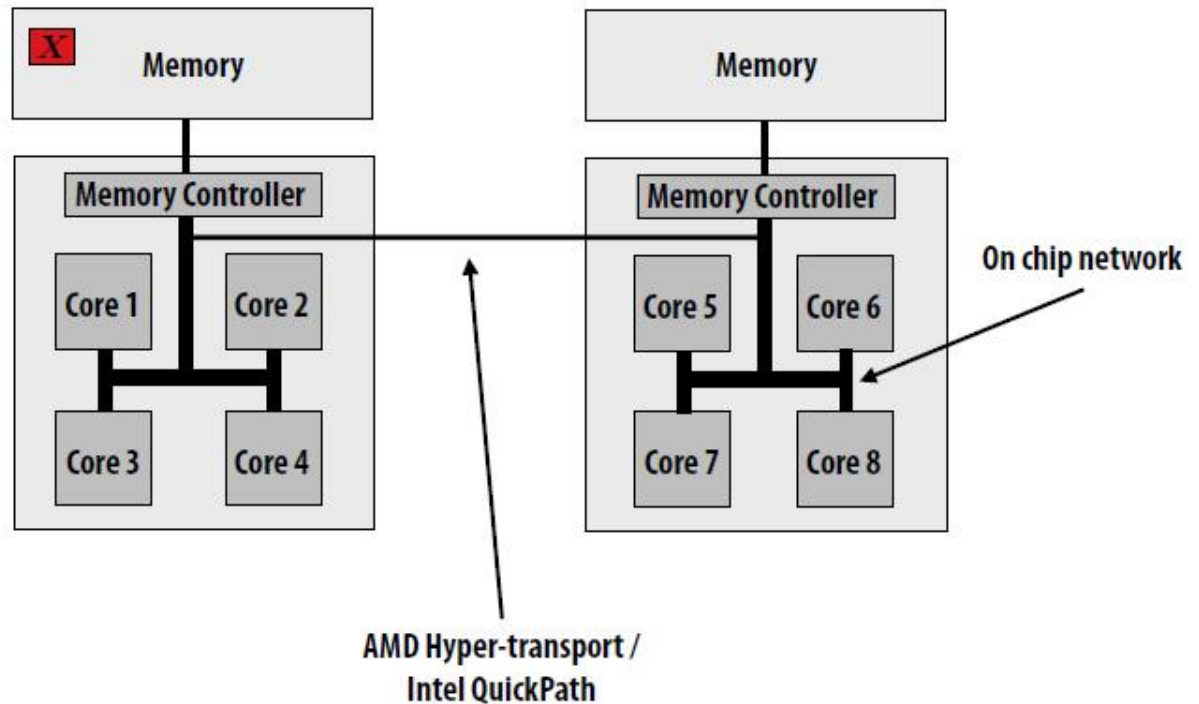


Skalabilne SM arhitekture - NUMA



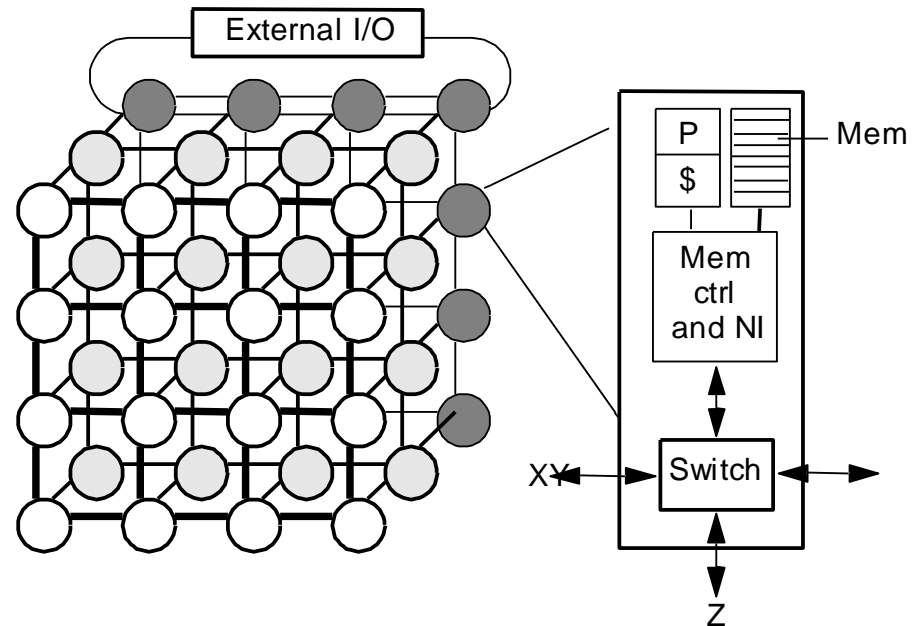
- NUMA (*Non-Uniform Memory Access*)
 - Distribuirana memorija
 - Lokalni i udaljeni pristupi (različite latencije)
 - Proste prouke preko ICN (npr. *read-request*, *read-response*)
 - Manji zahtevi za propusni opseg, bolje performanse
 - Mnogo bolja skalabilnost (hiljade procesora)
 - Teže za programiranje (podešavanje performasni)
 - Keširanje deljenih lokalnih i udaljenih podataka - složeno

Primer



Moderne *dual-socket* konfiguracije

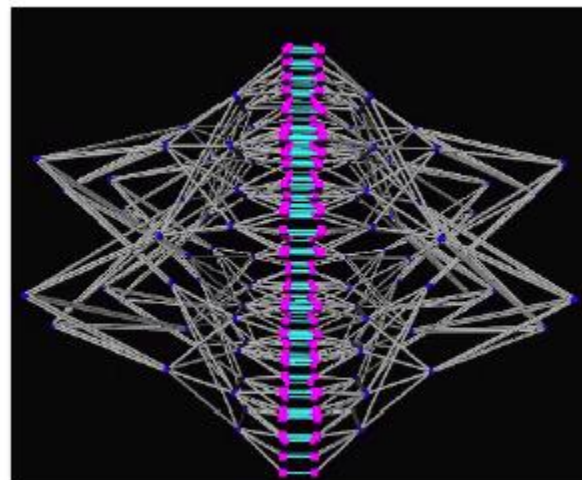
Primer: Cray T3E



- Do 1024 čvora (DEC Alpha + memorija + mrežni kontroler)
- 3D kocka sa 650MB/s linkovima prema susedima
- Sva memorija dostupna svim procesorima
- Memorijski kontroler organizuje prenos sa udaljenim memorijama
- Keširanje samo lokalnih podataka (za razliku od SGI Origin)

Primer: SGI Altix UV 1000

- Do 256 *blade-ova*, 2 CPU/blade, 8 jezgara/CPU = 4096 jezgara
- PSC Blacklight
- Deljeni adresni prostor
- ICN – *fat tree*

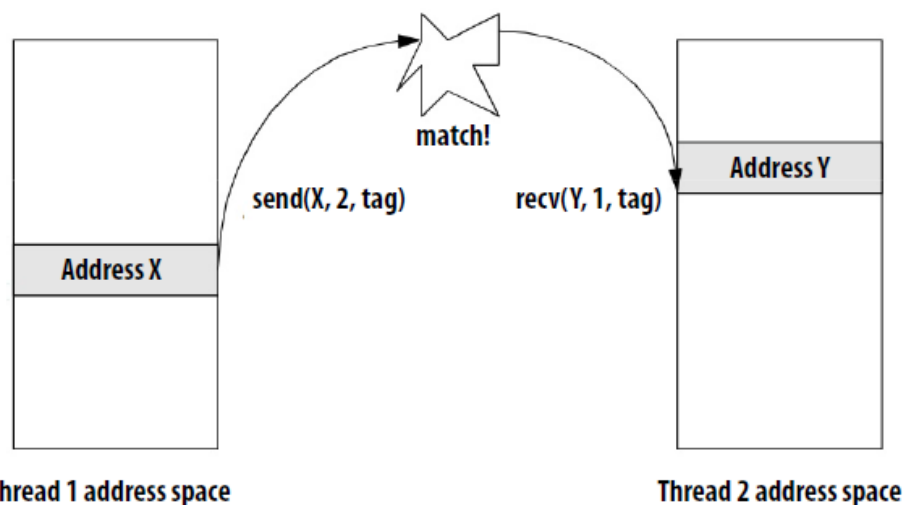


Fat tree

Model prenosa poruka

- Programski model
 - Direktan pristup privatnom adresnom prostoru
 - Logički deljeni podaci raspoređeni po privatnim adresnim prostorima
 - Eksplicitna komunikacija preko poruka (*send/receive*) vidljiva za programera i prevodioca
 - Poruke zahtevaju podatke ili neku akciju
 - Pošiljalac i primalac znaju jedan za drugoga (ne i kod SM!)
 - Mora se znati distribucija podataka
 - Manje pogodno za programera
- Programski model odvojen od osnovnih HW operacija
 - bibliotečke rutine ili OS ostvaruju akcije nižeg nivoa

Model prenosa poruka



- Pošiljalac specificira šta šalje (bafer sa podacima) i primaoca
- Primalac specificira bafer gde prima i pošiljaoca
- Čekanje ili prekid
- Prenos podataka između memorija (*by value*)
- Opcioni tag na predaji i pravilo uparivanja na prijemu

Model prenosa poruka

- Sinhrona komunikacija (uglavnom)
 - Čekanje na obe strane, potvrda
- Ponekad i asinhrona komunikacija
 - Slanje unapred
- Blokirajuća (čeka bafer) i neblokirajuća
- Korišćen čak i na jednoprocesorskim računarima (CSP, Occam)
- “*Overhead*” - kopiranje, rad sa baferima, zaštita
- Potencijalni problemi: blokiranje (*deadlock*), determinističko izvršavanje, performanse
- HW prostiji, SW složeniji

Model prenosa poruka

Sumiranje niza A[80000], sistem sa 8 procesora

- 1) Razdvojiti podatke u lokalne memorije, nizovi A[10000]
- 2) Sumiranje po procesorima

```
sum = 0;
for (i = 0 ; i < 10000 ; i++)
    sum += A[i];
```

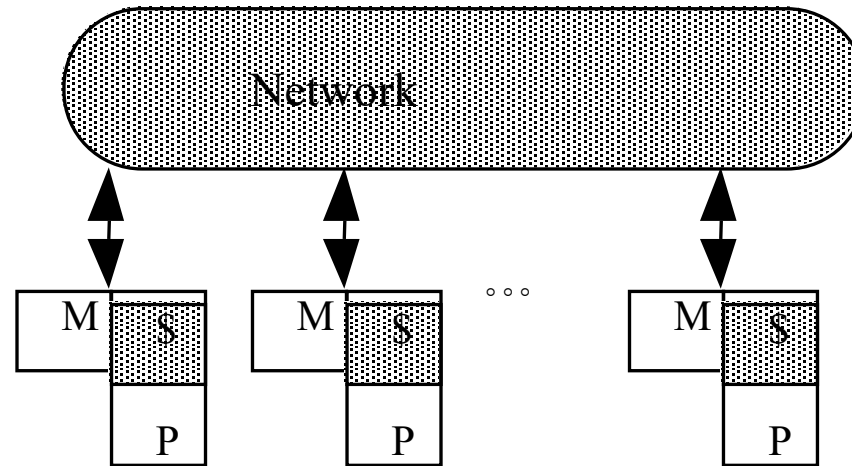
- 3) Sabiranje parcijalnih suma

```
half = 8;
do {
    half = half/2;
    if (Pn >= half) send(sum, Pn - half);
    else sum += receive ();
} while (half != 1) ;
```


Model prenosa poruka

- Operacije pozivi bibliotečkih rutina
 - Komunikacija
 - Usmerena (dva procesa)
 - Kolektivna (grupe procesa)
 - Sinhronizacija (barijera, nema brava)
 - Upiti ispituju okruženje
- MPI (*Message Passing Interface*)
 - Široko rasprostranjen, “*de facto*” industrijski standard
 - Dosta jezika
 - Dosta platformi (klasteri radnih stanica, SMP klasteri, MPP, heterogeni sistemi, DSM)

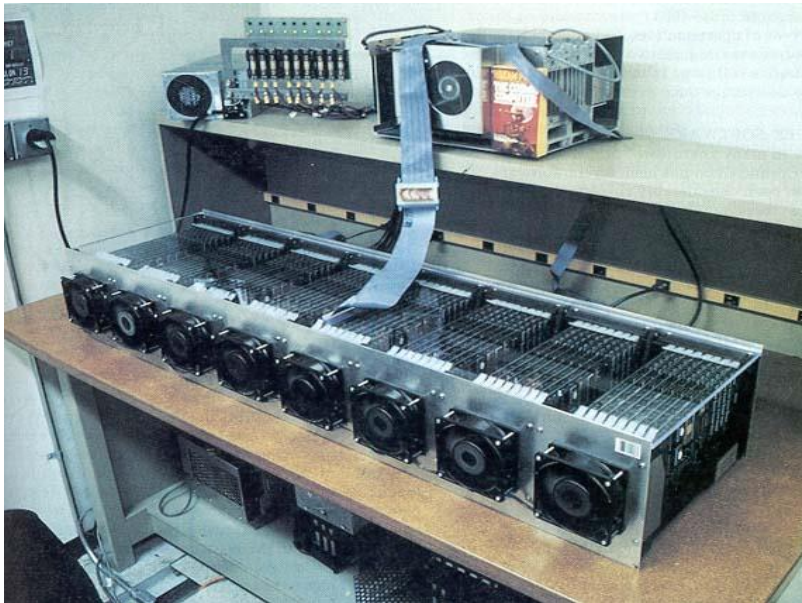
Model prenosa poruka



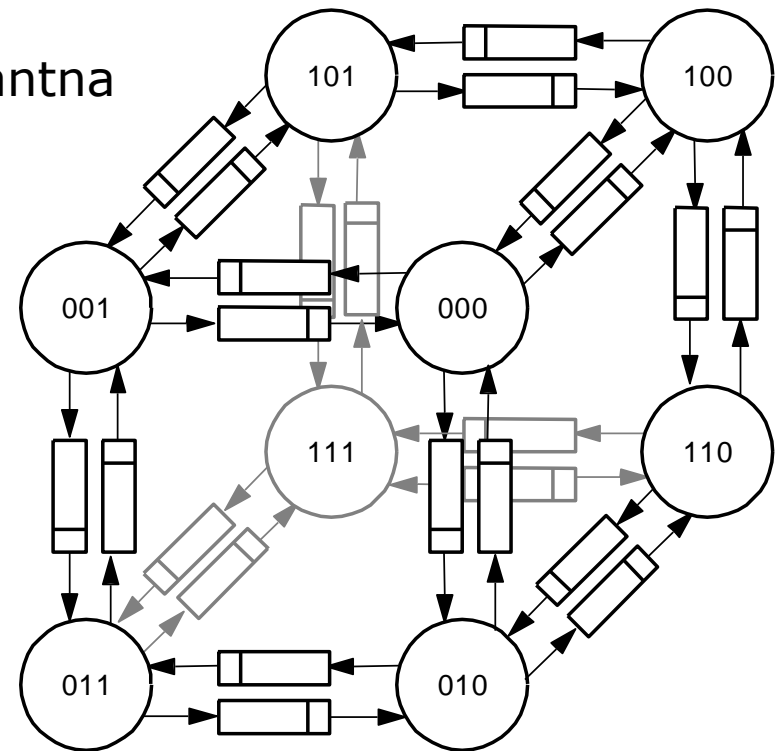
- Gradivni blok – praktično kompletan računar (*multicomputer*)
 - Odvojeni adresni prostori
 - Komunikacija preko eksplicitih U/I operacija
- Organizacija slična NUMA
 - Komunikacija integrisana na nivou U/I, a ne memorije
 - Slično NoW i klasterima, ali jača sprega CPU i mreže

MP arhitekture

- Rane mašine:
 - HW primitive bliske S/R
 - Topologija (kocke, mesh) dominantna
 - Mali FIFO linkovi (blokiranje)



CalTech Cosmic Cube (Seitz)



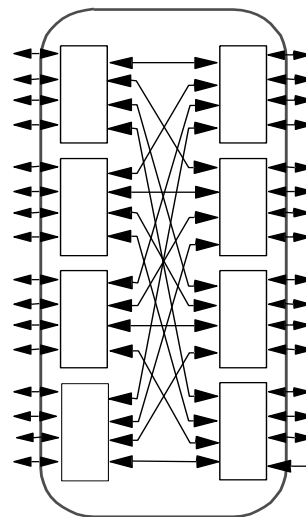
MP arhitekture

- Kasnije mašine:
 - Generalniji linkovi
 - DMA omogućava neblokirajuće slanje
 - Baferovanje na prijemu do Recv
 - Ne ograničava komunikaciju samo na susede
 - *Store&forward* rutiranje u HW
- Smanjena uloga topologije
 - Generalnije mreže
 - Proizvoljno "*pipeline*" rutiranje
 - Dominira vreme od CPU do mreže
 - Jednostavnije programiranje

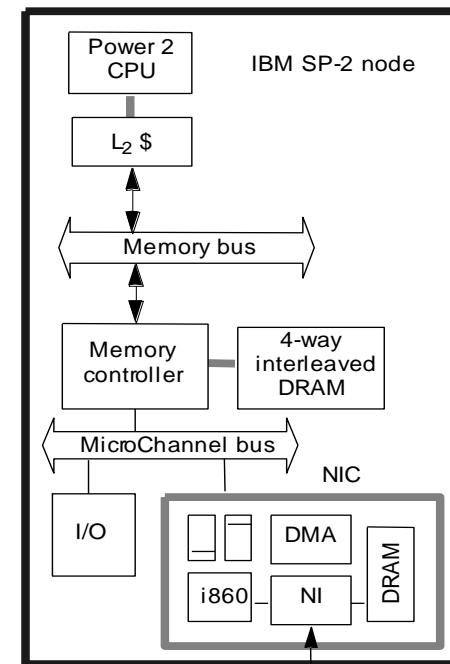
Primer: IBM SP-2



General interconnection network formed from 8-port switches



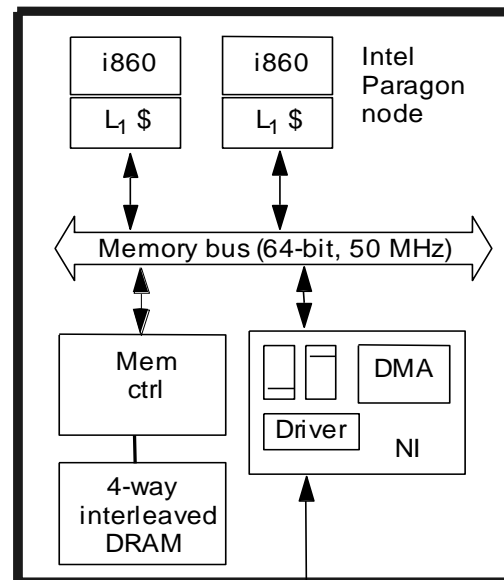
- Sastoji se od RS6000 radnih stanica
- Mrežni interfejs na U/I magistrali (40MB/s)



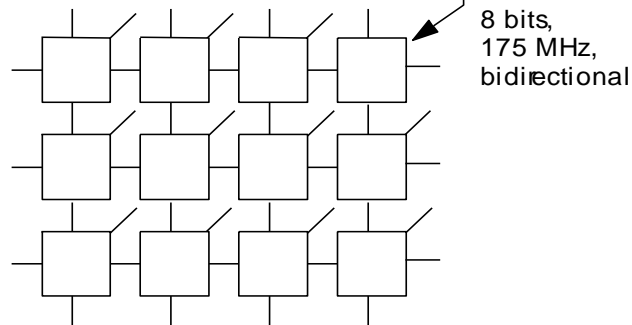
Primer: Intel Paragon



Sandia's Intel Paragon XP/S-based Supercomputer



2D grid network
with processing node
attached to every switch



- Mrežni interfejs na memorijskoj magistrali (175MB/s)

Primer: Berkeley NOW



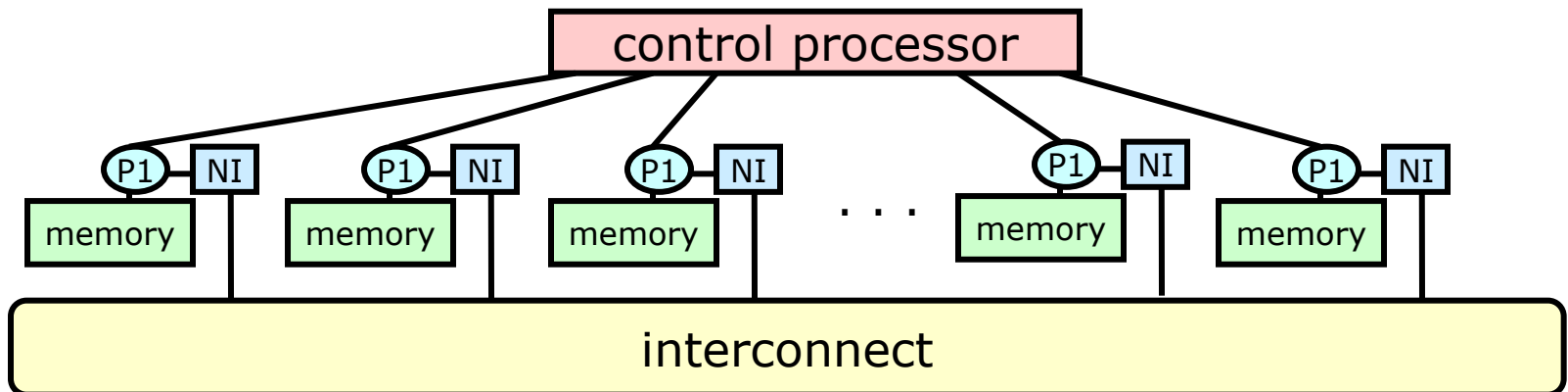
- 100 Sun Ultra2 radnih stanica
- Inteligentni mrežni interfejs
 - CPU + mem
- Myrinet SAN mreža
 - 160 MB/s / link
 - 300 ns / hop
- Globalni OS
- Odnos cena/perfromanse

Model paralelne obrade podataka

- Programski model
 - Operacije se izvršavaju u paraleli na elementima velike regularne strukture podataka
 - Jedan tok kontrole = sekvencijalni + paralelni koraci
 - Prostorni paralelizam
 - Podaci unapred raspoređeni po lokalnim memorijama
 - Sabiranje dva niza $A = B + C$
- Aplikacije
 - Odgovara strukturi nekih aplikacija (ne svih!)
 - "Regularne" sa dobrom lokalnošću
 - Diferencijalne jednačine, linearna algebra, ...
 - Pretraživanje dokumenata, grafika, obrada slike, ...

Model paralelne obrade podataka

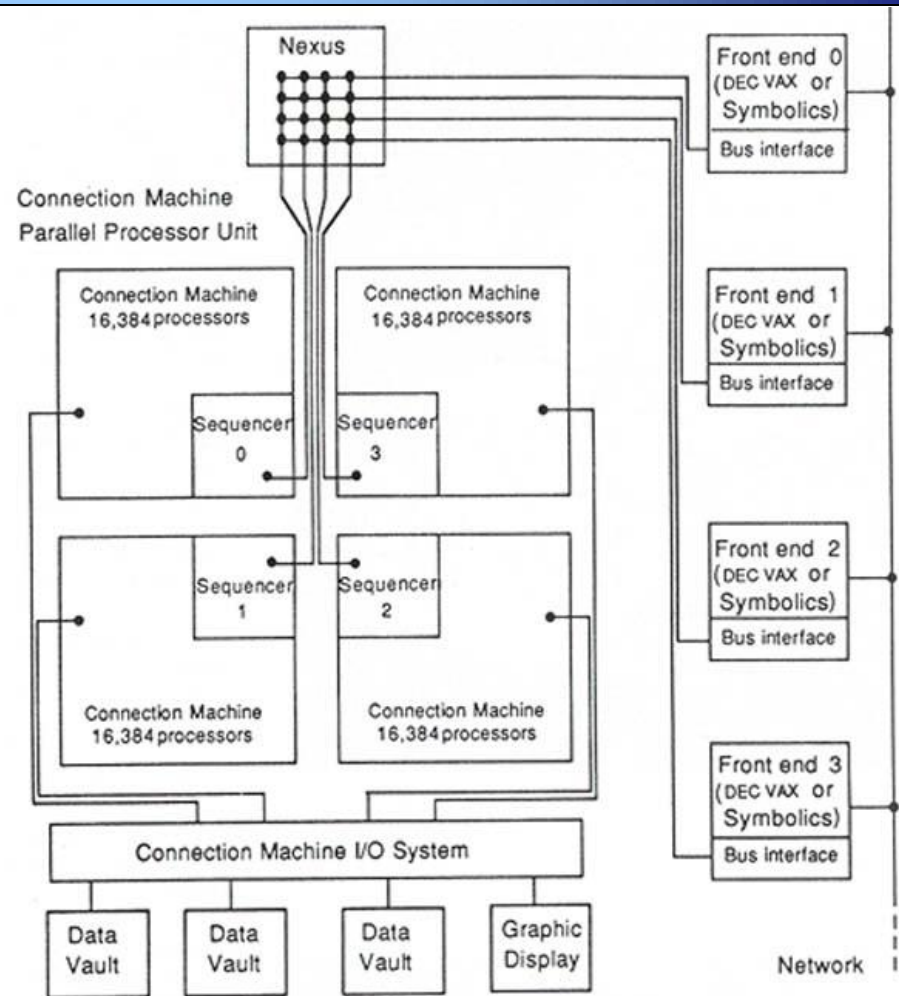
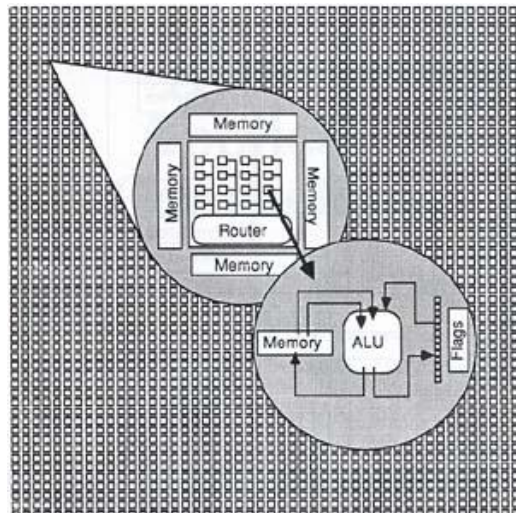
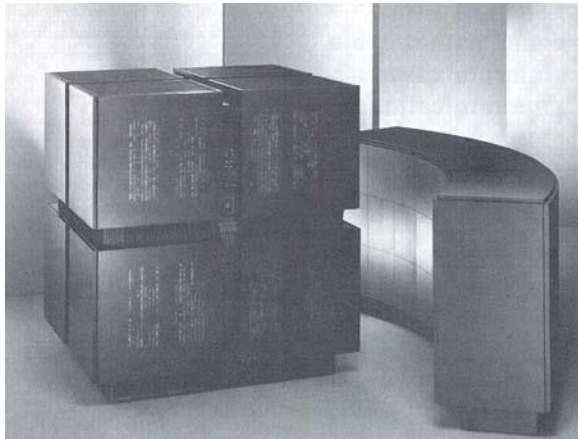
- Arhitektura SIMD (*array processor*)
 - Jedan kontrolni procesor i globalna memorija: kontrola toka i skalarne instrukcije
 - Niz prostih procesora koji rade nad lokalnim podacima
 - Amortizovana cena prihvatanja instrukcija i kontrole toka
 - Regularna sprežna mreža omogućava efikasnu komunikaciju prema susedima
 - Jeftina globalna sinhronizacija



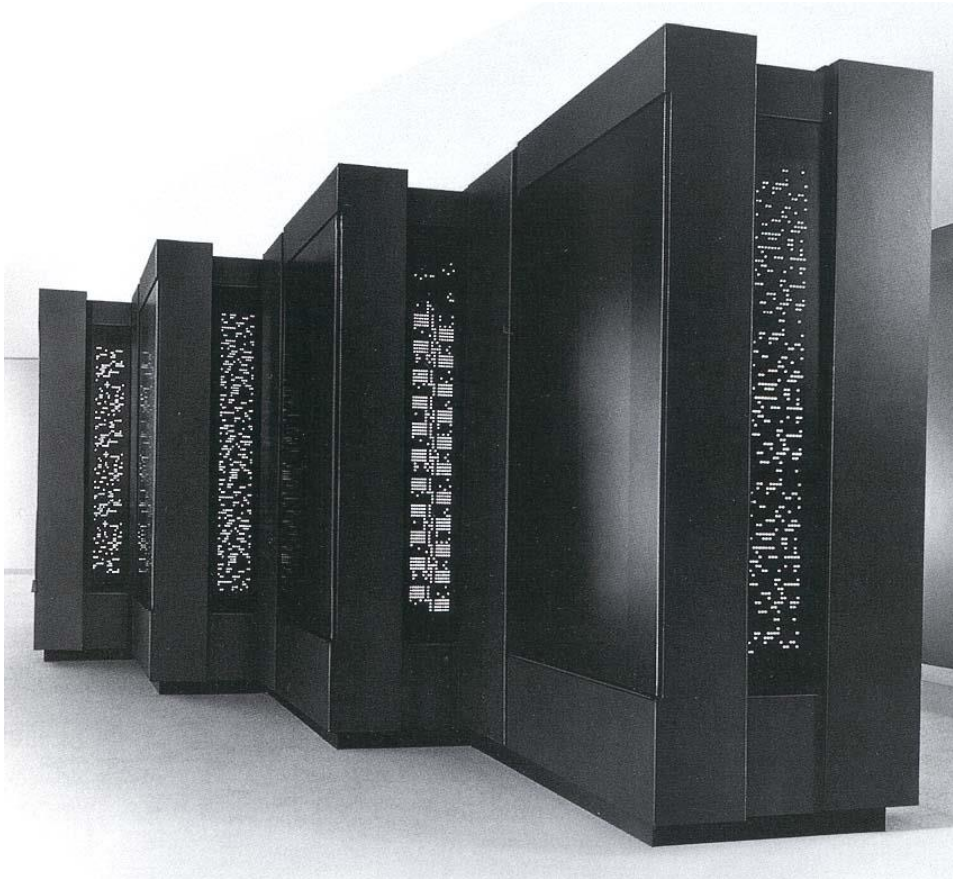
Model paralelne obrade podataka

- Sinhrona paralelna obrada ili komunikacija
 - Pojedini procesori mogu biti onemogućeni u nekom koraku (if, case, ...)
- “Virtuelni” snažni PE sastavljen od mnogo prostih
- Predstavnic
 - ILLIAC (UIUC, 64x64, 30M\$)
 - Thinking Machines CM-1 (64Kx1), CM-2, CM-5
 - MasPar MP-1 and MP-2
 - Sledbenici – vektorski računari CRAY, CDC, .. . (80-te)
 - Tehnološki trendovi (FPU, keš memorije) uslovljavaju evoluciju ka generičkoj arhitekturi

Primer: Connection Machine



Primer: CM-5



- Više MIMD, nego SIMD
- Prepakovana SparcStation
 - 4 po ploči
- “Fat-tree” mreža
- Kontrolna mreža za globalnu sinhronizaciju

Model paralelne obrade podataka

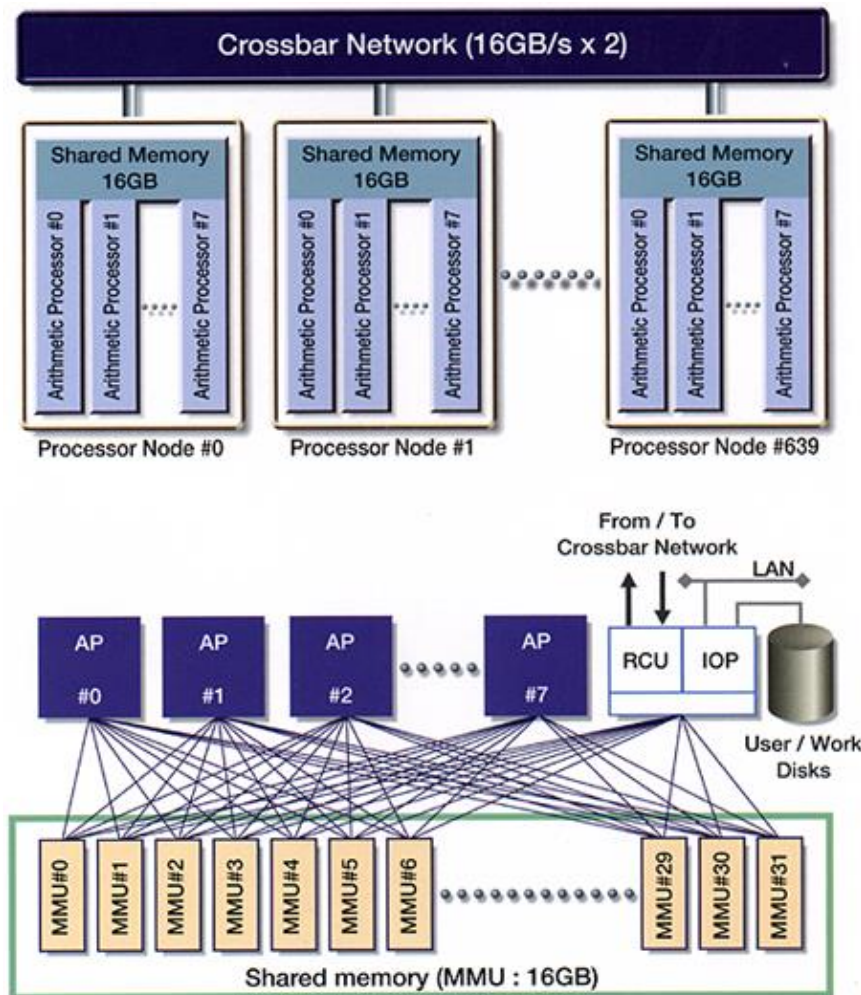
○ Vektorski računari

- Skalarni CPU i više FPU koji rade nad vektorima u memoriji koristeći protočnu obradu
- Vektorski tipovi podataka i instrukcije
- Vektorski registri, prenosi iz memorije
- Potisnuti od MPP u 90-tim

○ Povratak

- Velike mašine - Skalarni CPU Earth Simulator (NEC SX6), Cray X1
- Dodatni skupovi instrukcija
SSE2 u Intel Pentium/IA64, AVX (Intel u Core i7), AltiVec (IBM/Motorola/Apple u PowerPC), VIS (Sun u Sparc)
- GPU !

Primer: Earth Simulator

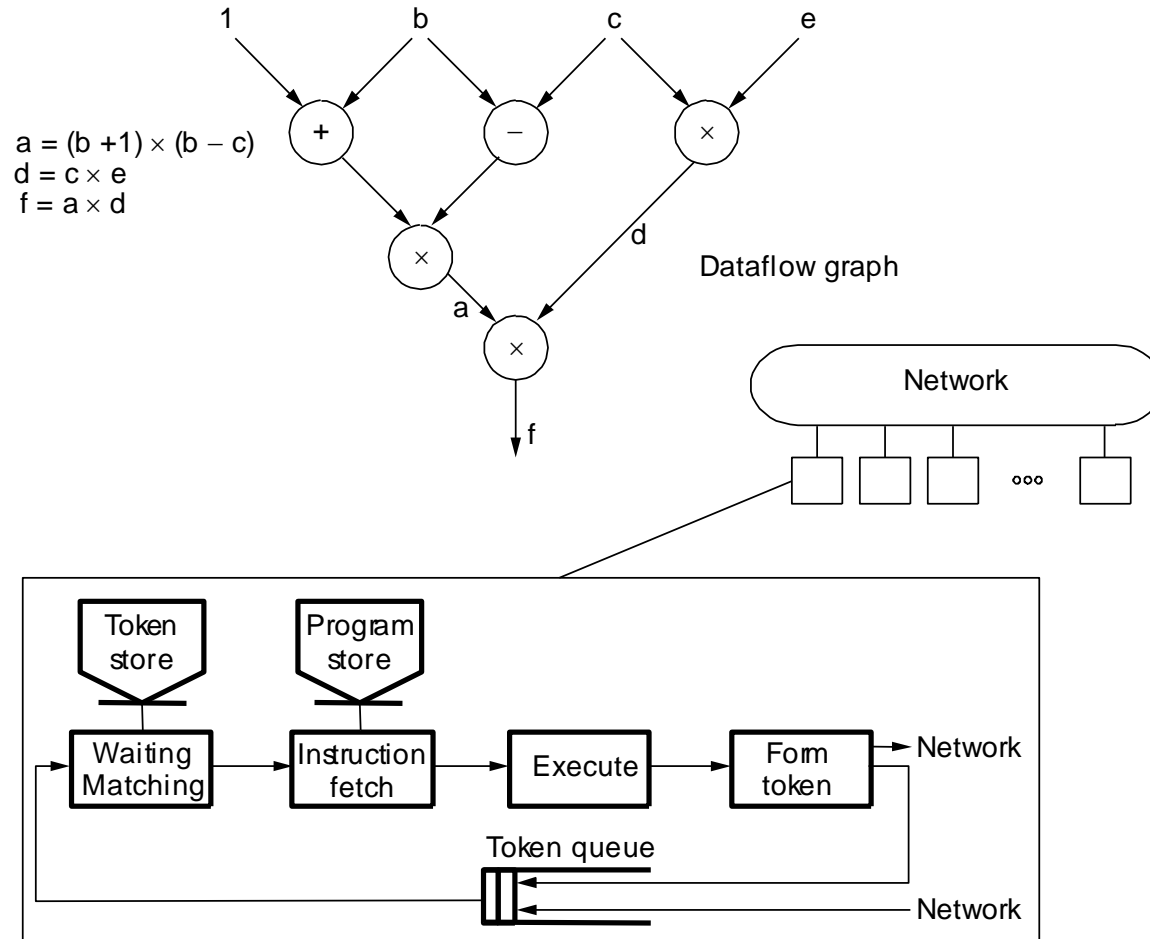


- NEC, Japan
- Klimatski modeli
- 640 čvorova sa brzim vektorskim procesorima
- Brza *crossbar* mreža
- Visok propusni opseg memorijskog sistema

“Dataflow” arhitekture

- Program – graf suštinskih zavisnosti podataka
 - Čvor - operacija i specifikacija odredišta
 - Instrukcija može da se izvrši čim su joj podaci spremni
 - Poruka sa rezultatom (*token*) i tagom odredišnog čvora
 - Poređenjem taga, aktivira se neka instrukcija ili se čeka
 - Može da se koristi ILP (fina granularnost)
- Arhitektura
 - Pogodno za paralelne arhitekture
 - Nema brojača instrukcija, instrukcije nisu uređene
 - Statičke i dinamičke (dinamičko proširivanje grafa)
 - Manchester i MIT mašine

“Dataflow” arhitekture

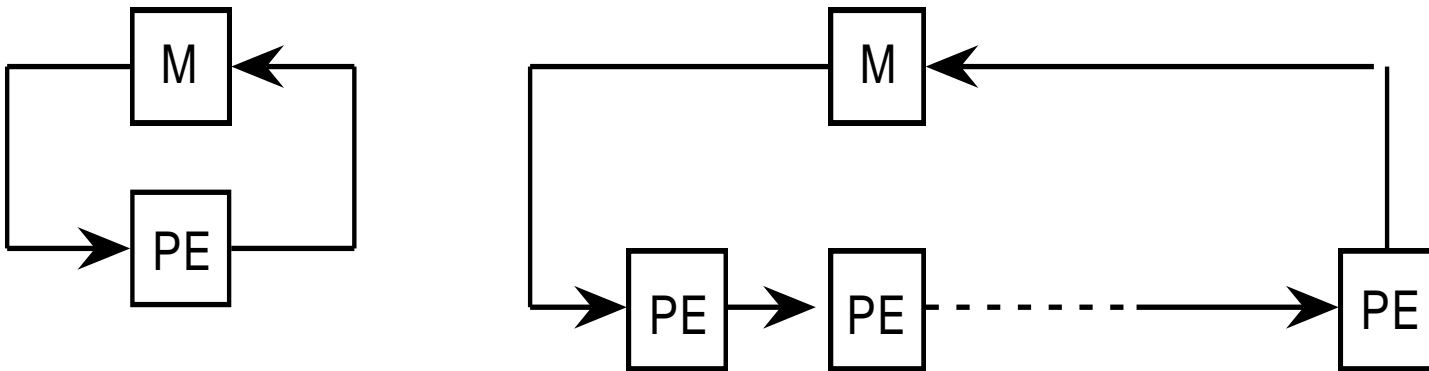


“Dataflow” arhitekture

- Problemi
 - Ne koristi lokalnost operacija
 - Obrada složenih struktura podataka
 - Složenost poređenja tagova i memorijske jedinice
 - Nepotrebna računanja (bottom-up, eager evaluation)
- Konvergencija ka konvencionalnoj NUMA arhitekturi
 - Podrška za veći, dinamički skup niti (mapiranje na PE)
 - Integracija sinhronizacije sa kreiranjem niti
 - Integracija PE i mreže (poruke u registrima)
 - Tipično se koristi globalni adresni prostor, ali sinhronizacija finija (na nivou elementa)
 - Programski model se odvajaja od specifične HW strukture

Sistolne arhitekture

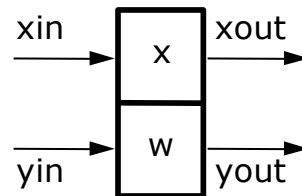
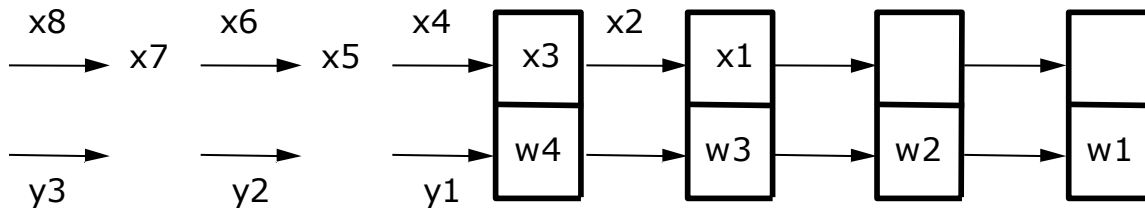
- VLSI omogućio jeftine čipove posebne namene
 - Nizovi prostijih PE povezanih regularnom 2-D sprežnom mrežom (linearna, trougaona, hexa, mesh, ...)
 - Tok podataka između PE -> mnogo veća propusnost za isti propusni opseg memorije (nije usko grlo)
 - PE mogu imati lokalnu memoriju i raditi različite operacije
 - Primer: CMU Warp



Sistolne arhitekture

Primer: Systolni niz za 1-D konvoluciju

$$y(i) = w1 \times x(i) + w2 \times x(i + 1) + w3 \times x(i + 2) + w4 \times x(i + 3)$$



$$\begin{aligned}x_{out} &= x \\x &= x_{in} \\y_{out} &= y_{in} + w \times x_{in}\end{aligned}$$

- Aplikacije slične kao u SIMD
 - Obrada slike i govora, linearna algebra, grafika

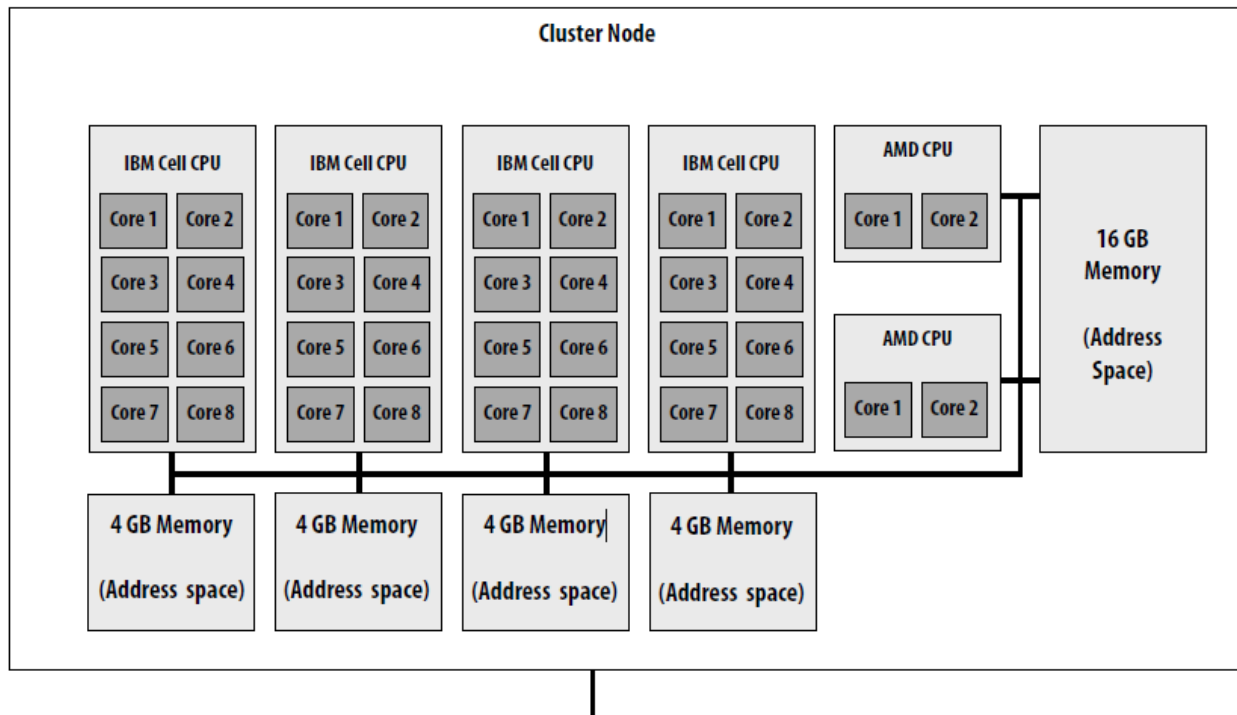
Sistolne arhitekture

- Mapiranje algoritma na HW
 - Mogu i generalniji procesori za više algoritama na istom HW
 - Sprežne veze odgovaraju modelu komunikacije, direktan prenos
 - MISD ?
- U/I na topološkim granicama strukture
- Globalni takt sinhronizuje izračunavanja i prenos podataka (problem "*clock skew*")
- Proširljivost, fleksibilnost, jeftina implementacija
- "*Wavefront array*" procesori
 - Asinhroni – nema globalnog takta, rukovanje u HW
 - Pogodniji za složenije PE i aplikacije kod kojih vreme izvršavanja zavisi od podataka (npr. retke matrice)

Hibridni programski modeli

- SM & MP vrlo često
 - Zajednički adresni prostor u CMP čvora klastera
 - Prenos poruka između čvorova
- DP podržava sinhronizacione primitive u kernelima (CUDA, OpenCL)
 - Ograničen vid komunikacije
- SM & DP (kod CUDA/OpenCL)
 - DP za skaliranje na *manycore*
 - SM adresni prostor dozvoljava da niti na istom čvoru komuniciraju
- DARPA HPCS jezici
 - Hibrid DP i niti u SM

Hibridni programski modeli



Konvergenција arhitektura

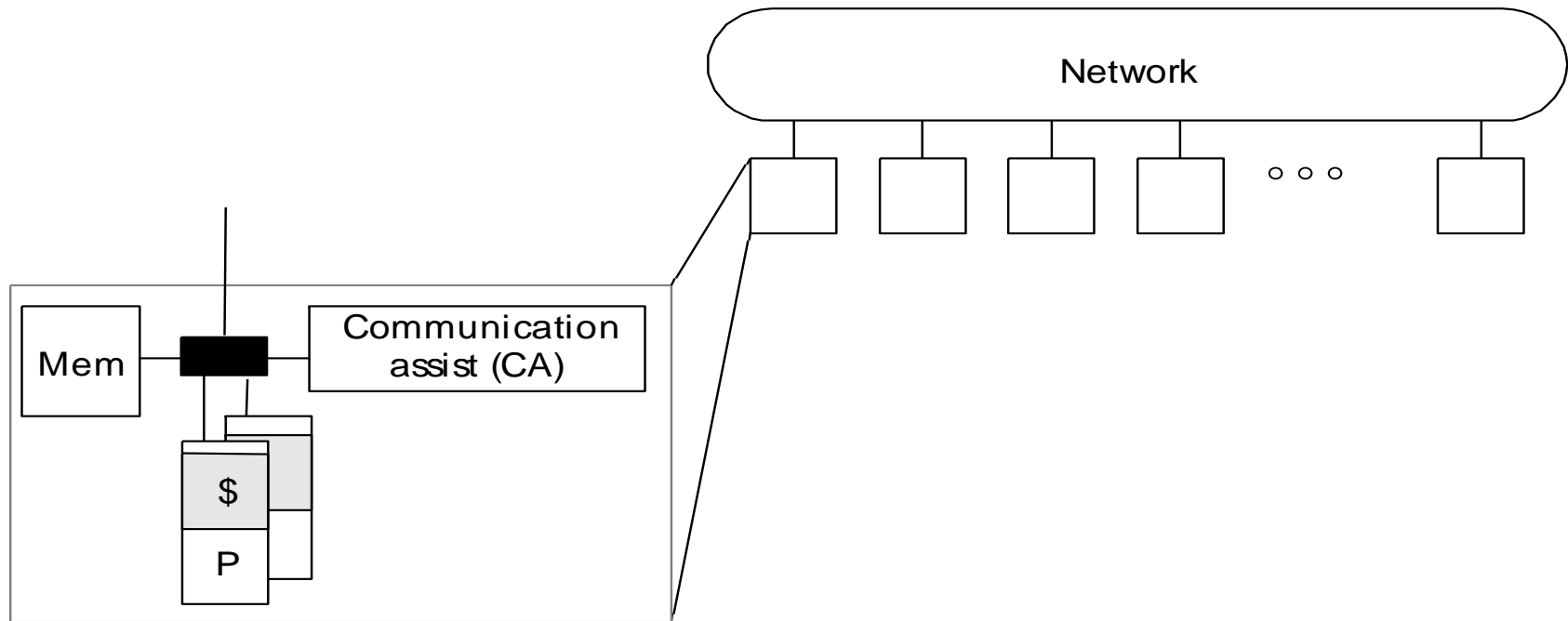
- Konvergenција SM i MP
 - *Send/recv* na SM mašinama kroz deljene bafere
 - Globalni adresni prostor na MP (GA -> P & LVA)
 - Deljena virtuelna memorija na nivou stranice u MP mašinama
- DSM (*Distributed Shared Memory*) = SM programski model + HW multiračunara
- Konvergenција HW organizacija
 - Čvršća integracija mrežnog interfejsa u memorijsku hijerarhiju, čak i na MP mašinama (manja latencija, veći BW)
 - Veće SM mašine razmenjuju poruke

Konvergencija arhitektura

- Klasteri radnih stanica/SMP kao paralelni sistemi
 - Termin CLUMP (cluster of SMPs)
 - Brze SAN (Ethernet, ATM, FiberChannel)
 - Multiprogramiranje ili paralelne aplikacije
 - Heterogeni sistemi sa GPU

- Programski modeli različiti, ali organizacije konvergiraju
 - Čvor povezan na ICN preko podrške za komunikaciju
 - Implementacije konvergiraju, posebno u mašinama većeg obima

Konvergencija paralelnih arhitektura



- Generička NUMA paralelna arhitektura
 - Podrška za komunikaciju (CA) u čvoru
- Skalabilna sprežna mreža

Konvergencija paralelnih arhitektura

- CA – mrežni interfejs sa komunikacionim kontrolerom ili posebnim procesorom
 - Različitosti integracije i implementacija ...
- Specifičnosti CA uslovljene programskim modelom
 - SM – jaka integracija u memorijski sistem (podržava udaljene pristupe i protokole koherencije)
 - MP – efikasno slanje i prijem eksplicitnih poruka
 - DP – brza globalna sinhronizacija
 - DF – dinamičko raspoređivanje izračunavanja
- Razdvajanje programsokg modela od arhitekture
 - Npr. implementacija HPF ili MPI