SPM.1

Sisteme cu Procesoare Multiple

Sistem cu procesoare multiple ⇒ proc. utiliz. resurse comune și colaborează între ele.

Sistem cu proc. multiple ≠ sist. multiprocesor (f. paralel)
Sist. multiprocesor ⇒ fol. comunicație prin zone comune de memorie
⇒ subcateg. a sist. cu proc. multiple
Sist. distribuite ⇒ proc. sunt sit. la dist. geografice mari ⇒
⇒ comunic. se face prin schimbul de mesaje (format serial)

Parțial → săpt. 8 ⇒ se dă o sg. dată ⇒ dc. e picat ⇒ restanță
Lab → 4 teme:
- 1 referat
- 1 program de simulare
- 2 programe de programare paralelă

ED 308, 312, 321
↳ grafică 2 animație
ED 320, 309

spm.aii.pub.ro

Parțial 2 Exam ⇒ 2 subiecte + 1 pb
30%, min. 5(15) 30%, min. 5 (15)
Lab. → 40%, min. 5 ⇒ 4 teme, min. 24 pcte.

A1: IT practic, SO în timp real
A4: pb. roboților și lucrări conexe (vd. artf., traiectorii etc.)
A3: senzori și traductoare
A2: teorie multă
↑
Mai multe amănunte la cursul de Joi

---

1 [Procesoarele multiple fct. în paralel ⇒ arhitecturi de tip paralel.
  Modalități de a evalua performanțele ⇒ arh. superioare altora, clasif.
2 - Procesoare pipeline ⇒ cresc. vitezei de calcul prin evidențierea unor etape de dim. egle.
⇒ fol. mai ales pt. instrucțiuni ⇒ instrucț. e separată în 5 etape:
- op. fetch (cod instruct.)
- instruction decode
- operand fetch
- execution
- write back

⇒ ~~...~~ se preferă arh. RISC

⇒ instr. de salt șterg pipeline-ul, și fm. o nouă structură pipeline

⇒ pb. de hazard ⇒ când dtle. nu sunt gata pt. exec.

⇒ pipeline e utilizat mult în instr. aritmetice.

3- Rețele de interconectare ⇒ 1. tip magistrală: structuri paralele (sist. multiprocesor)

⇒ 2. rețele din topologiile statice (nu se modif. pe toată durata de viață a sist.)

⇒ 3. topologii dinamice (se reconfig. după condițiile ∃ la un mom. dat)
- rețele cu blocare (la un mom. dat nu e neapărat posibilă conect. între nod. A și nod B, ci depinde de starea rețelei la momentul ant.)
- rețele fără blocare: mai scumpe (hw. spec.) și mai lente
- cu rearanjare: reorganizează soluția la momentul respectiv...

4- Multiprocesoare ⇒ bază: • MultiProcessor Specification (intel) (verif. Google)
• Creating Multiprocessor Nios II System Tutorial (Google)

⇒ scheme de multiprocesoare

⇒ coerența mem. cache ⇒ nu trb. să difere info. din cache MP1 față de MP2

⇒ standard. OpenMP (intel) ⇒ permite programare eficientă a sist. SMP (Shared Memory Processors)

6 - Arh. data-flow și sisto-lice

7 - Exemple

Cap. 1: Clasificări arhitecturale

Cost vs. performanțe
↓
depind de cerințele pb. pe care
tb. să o rezolve sistemul

FLOPS ⇒ Floating Point Operations Per Second
IPS ⇒ Instructions Per Second

Comoditatea programării ⇒ ca să scot aplicații rapid, dar cu o echipă mică de programatori

Disponibilitatea ⇒ proporția de timp în care sist. este disponibil
↳ 100% ⇒ tb. să fie mai multe sisteme care să fct. în comun
- cold standby: se strică una, o schimb cu alta (nepornită)
- warm standby: — " — " — care era pornită și în așteptare
- hot standby: mașina de rezervă urmărește în timp real activitatea mașinii 1 ⇒ ca să fie fiabil tb. să am 3 (răsp. corect e cel dat de majoritate).

Benchmarks ⇒ platforme de testare

SPM.2 Benchmarks → tip nucleu (kernel): fragmente de cod extrase din programe reale (care au pus în general probleme) și care se rulează pe post de test
→ locale: teste specifice companiilor producătoare de echipamente IT
→ parțiale: înregistrează trace-ul programului
→ recursive: alg. recursivi
→ Unix SPEC: sistem complex de sw. - 10 scenarii științifice & industriale.
→ synthetic benchmarks: teste artificiale → programe construite speciale pt. testarea µP
- Wetstone: special pt. FPO (floating point operations) → bucle cu mare grad de code locality (de obic. se găsesc în cache)
- drystone: folosesc șiruri, progr. nu au bucle ⇒ cache miss frecvente dc. mem. cache e mică → testează abilitatea µP

de a manipula eficient mem. cache.
→ paralell benchmarks: testarea arh. de tip paralel.

Sisteme de calcul: — serial (1 sg. μP)
pipeline
paralel

Cadența cu care instr. intră în pipeline e dată de durata maximă necesară exec. unui fragm.

Sisteme multiprocesor: → magistrală paralelă
→ compacte
→ sist. „interne cuplate"
→ mem. comună, porturi I/O comune

Sisteme distribuite: → fiec. μP are mem. lui
→ comunicația se face prin schimbul de mesaje
→ sist. „slab cuplate"

conectate prin rețele

SIMD → accesează I e transmisă în // mai multor μP, fiecare μP lucrând asupra unui D ⇒ toate μP execută la un mom. dat aceeași instr.

I = instrucțiune
D = dată stream

MISD → structură dificil de realizat practic → procesoare vectoriale (mai multe instr. acț. asupra aceleiași date)

MIMD → multe μP comunică între ele; nici unul nu e independent
structură f. fiabilă și flexibilă.

dacă un μP se defectează, un altul îi preia fețele.
⇒ sist. funcț. în continuare, dar la perf. mai mici

Clasificări arhitecturale:

A ≤ L ≤ C ≤ H → minimizarea timpului de rulare a aplicației.

Esențial este A (niv. de ||ism permis de aplicație).
(∀) Alg. are o secțiune secvențială și una ||.

Grad de paralelism → proporția între sect. executate || și cele exec. secvențial

În general la sist. puternic cuplate paralelismul apare la niv. taskului, iar ptr. sist. slab — " — " — " — procedură. La pipeline paralelismul apare la nivel de instrucțiune / operație / microoperație.

Gradul de paralelism depinde f. mult de interdependența datelor.

Modele de procese:

Pg. 40 → barieră: tb. să ∃ o structură de sincronizare care să nu permită lansarea în execuție a modulului P înainte ca toate modulele 2÷(P-1) să își fi încheiat execuția.

SPM.3 Granularitate → rel. dintre timpul de exec. a taskului și timpul de comunicare cu alte [illegible] µP. [illegible]

Granularitate fină → sistemele utiliz. ptr. comunicație → de ex. în sistemul bancar (ATM).

• Clasif. Treleaven → în fcție. de factorul care pune în mișcare arh.:

→ control-driven: mașina clasică — ∃ un program, iar fcț. mașinii e comandată de instr. programului; instruct. sunt memorate la locații fixe și aduse cu ProgramCounter; Ex.: arh. RISC, CISC, HLL (High Level Language)

↓ capabile să execute direct instr. scrise în lbj. de niv. înalt, fără a mai necesita compilator, link-editor, asamblor.

→ data-driven: arh. la care nu mai ∃ un program memorat într-o memorie, ci ∃ o multitudine de blocuri fcționale ce au ca scop evaluarea unor fcții; blocurile fcționale se conect. a.î. out. unor blocuri fcț. să repr. in. altor blocuri fcț. din sistem; mașina s.n. data-driven ptr. că un bloc fcț. nu își începe fcț. decât dc. la intrările sale se găsesc toate argumentele (dtle.) de care are nevoie; blocurile Σ interconectarea lor se realiz. prin programare;

→ demand-driven: spre deosebire de data-driven, mai are o caract. suplim. ⇒
⇒ operația nu se execută decât dc. blocul fcțional cere explicit acest lucru; op. nu sunt realizate decât la cerere ptr. a nu încărca µP inutil.

- Clasif. Flynn:

→ SISD: monoprocesor
→ MISD: µP pipeline
→ SIMD: array processors → fluxuri multiple de dte. sunt distrib. unor µP identice, și ∃ un master care are secvența de instr. ce tb. aplicată dtlor. ⇒ cele n µP primesc simultan aceeași instr. și o execută simultan
→ MIMD: procesoare multiple
↓
sisteme [illegible] cu µP multiple
(oferă avantaje importante)

Structura multiprocesor:

→ puternic cuplată
→ leg. µp. cu memoria se face printr-o rețea de interconectare care permite comunicarea paralelă ⇒ ∃ mai multe tipuri de rețele de interconect.

Structura multicomputer:

→ slab cuplată
→ leg. între computere se face printr-o rețea de interconectare; este necesară ∃ unui protocol care să guverneze schimbul de informații

Structura multi-multiprocesor:

→ multicomputer, dar fiecare 'computer' este de fapt o structură multiproc.

Instr. scalare: utiliz. un sg. elem. de dte.
— " — vectoriale: instr. similare cu cele dintr-o mașină SIMD ⇒ avem un vector de dte. organizate pe . . . . . . , iar dtle. sunt distribuite µP din sist. SIMD. Array Processor.

Structura procesor vectorial:

→ instr. sunt aplic. aceluiași flux de dte., văzt. ca un vector
→ structură pipe = 

→ sistem monoprocesor

Structura Matrice Sistolică (Systolic Array)

→ ∃ un val de dte. și un baraj de µP → la un mom. dat valul de dte. pompat de sistolă atinge simultan pr. baraj de processing elements (PE) → la acest mom. are loc procesarea, iar apoi dte. sunt date toate o dată către urm. baraj de PE → când pr. val de dte. atinge pr. baraj de PE, primul val de rezultate e eliberat de pr. baraj de PE (simultan).

• Clasif. Skillcorn:

→ f. multe niveluri, marea maj. teoretice.

SPM. 4 Skillcorn:

→ computational model → extrem de abstract; f. puține legături cu pb. pt. care e destinată mașina

- → modelul mașinii abstracte → ...
- → modelul bzt. pe performanță → se referă la arhitectura considerată ca umr. finit de stări → mașina stă atâta timp cât nu au loc evenim.;
- → modelul implementării → mașina ASM e implementată tehnologic

4 tipuri unități fcționale:
- IP = instruction processor = generează comenzi care se dau celorlalte unități
- DP = data processor
- DM & IM = data & instruction memory
- rețea de interconectare → struct. dinamică (lég. fixe → se modif. în fcție. de circumstanțe) → rețeaua e ca un switch

4 tipuri de comutatoare

• 1 to 1 → un obiect poate fi conectat la alt obiect; de regulă leg. sunt bidirecționale → la conectarea procesoarelor cu memoria (dtle. sunt citite de µP din memorie, modif. (dc e cazul) și puse la loc în memorie)
- → half-duplex → procedura de urmărire nu poate avea loc simultan, în ambele direcții
- → duplex → ar fi necesare 2 canale de comunicație

• n to n → mai multe 1 la 1 juxtapuse

• 1 to n → µP e conectat la toate memoriile

• n by n → (∀) µP e conectat la (∀) memorie

n-n = n to n
n x n = n by n
→ pg. 64.

• Taxonomia Duncan:

→ arhitectura paralel: pp ∃ (în cadrul de nivel înalt) a μP (cu memorii) ce permit execuția ∥ a secvențelor din programele complexe.
nivel înalt

Arhitecturi sincrone ⇒ dtele. introduse cu o cadență
⇒ ex.: • μP vectoriale
• arh. SIMD
• matr. sistolică

Arh. MIMD ⇒ loosely coupled & strongly coupled

μP vectoriale: n procesoare în succesiune, care efs. prelucrări diferite asupra unui flux de dte. ce trece prin ele ⇒ paralelism la nivel de task sau la niv. de procedură.

Arh. asociative ⇒ utilizează memorii asociative (gen cache) → ∃ un reg. mască care arată care dintre biții de dte. sunt folosiți de mem. asociativă → pg. 104

Arh. sistolice ⇒ se folosește str. dte. uniforme x op. ce se execută
⇒ avantaj ⇒ rez. intermediare nu sunt stocate în mem., ci sunt transmise de la μP la μP
⇒ inițial în acumulatoarele μP se găsește '0'.
pg. 102 ⇒ la intrările fiecărui μP ∃ o coadă; avem 4 μP → se dă tactul ⇒ pri μP face axe, coada se micș., '0' se propagă la μP2; '0' din μP 2 intră la procesare ⇒ coada se micș. la [g-h]; vine al 2-lea tact →
(sunt niște greșeli în slide, în fig de la log.)

μP1 ⇒ ae+cf
μP2 ⇒ 0+gh
μP3 ⇒ 0+0
μP4 ⇒ 0+be

la pasul urm. → noul conținut e vechiul conținut * coada * ce s-a propagat ...

→ execută rapid operații simple dar multe (de ex. calcul cu matr.)