omp_set_num_threads
— " get — "
omp_get_max_threads
omp_get_thread_num //preia id-ul de proces
omp_get_dynamic
omp_get_tick //nr. de secunde intre controlul luat de un thread si
//un altul

SPM.19

Cap. 6: Multiprocesoare

Multiprocesor & multicalculator

Multiprocesor = sist. in care datorită imposibilității de realizare cu met. obișnuite a capacității de calcul ptr. rez. unor pb. complexe, se mloc. calc. supercomp. cu sist. in care m. multe procesoare lucrează impreună ptr. realizarea sarcinilor.

Partea de µP vectorial — cum se realizează supercalc.
↳ greu de real
↳ paralelism de nivel scăzut (la niv. de instr.: ppl. cu dif. tipuri de instr. aflate in dif. etape de prelucrare)

Avantaje ale m. multor calc care lucrează in ||:
→ dc. un calc. al struct. se deteriorează, rămân celelalte N-1 și, cu o programare adecvată, fct. µP. defect sunt sunt preluate de celelalte µPs.;
→ oferă o m. bună fiabilitate și un grad mare de disponibilitate

Pg.4: → sistemele sunt proiectate in funcție de aplicație, nu sunt sisteme generale
→ vtza. µPs. e legată de timpul de răspuns cerut unei astfel de structuri
→ sist. de mem. e gândit in leg. cu volumul de dte. cu care se lucrează, dar și de volumul de dte. fol. in comun; utilizarea in comun pp. cereri simultane de acces la dte. (pb. rez. imediat ptr că timpul de acces la mem. pb. să fie scurt); pb. asigurată păstrarea coerenței info. din memorie (și aici se utiliz. mem. cache rapidă care e fiecărui µP, in ea găsindu-se copii ale dtlor. din mem. principală);
→ dc. op. sunt de citire, nu sunt pb.; dc. se dorește scrierea de către µP. a unui rez., el o inscrie in cache-ul propriu ⇒ celelalte µPs. nu deții modif., și nici mem. principală ⇒ pot apărea rez. eronate
→ alegerea sist. de mem. nu e trivială ⇒ protocoale de comunicație;
→ conectarea µPs se face prin rețele de interconectare ⇒ alg. de dirijare coresp. (toate tipurile de rețele) ↳ se aleg in funcție de topologia rețelei

Met. de cundă:

→ un sist. tb. să aibă o unit. de cundă centrală sau distribuită;
→ strateg. de fct. a μPs. e subord. unui mec. de cundă aflată la niv. rețelei (ex.: hot potato);
→ fiec. nod al rețelei are propriile mec. de cundă. ⇒ mec. de cundă se poate schimba de la un nod la altul (pb. congestionării rețelei);

Fiabilitate:

→ stab. dc anumite trasee tb. să fie redundante, dc. anumite trasee tb. multiplicate sau nu;
→ unitate de rezervă (în cz. de pană de ert.);
→ cum detect. dc. un μP. e defect: fiec. μP., în zona de bkground, se autotestează sau testează o anumită zonă din mem. principală;
→ de obicei ∃ cel puțin 3 unități ce lucrează în ||| ⇒ principiul votării pt. a det. care din ele e defect;
→ unit. de rezervă se află în standby față de unit. care lucrează ⇒ 3 tipuri de standby: cold standby, warm standby, hot standby ⇒ asigură timpul cel mai scurt de substit. a unit. defecte cu cea bună; unit. de rezervă face același tip de op. ca și cea principală, este sincronizată cu ea, însă doar unit. principală scrie în mem.;
→ niv. de paralelism: depinde de nat. pb. care se rez. (nu toate pb. sunt paralelizabile); depinde de nr. de μPs.;
→ scalability ⇒ μP. pe care îl proiect. servește unei fam. de pb date, însă de la a. fam. se adaugă o altă, sist. tb. să se expandeze ușor ⇒ nr. de modif. pt. expandare tb. să fie minim;

Ggv: Ce vrem de fapt?
- performanță ∞ cu cost 0
- fiabilitate 100%
- expandabilitate cât mai ușoară

⇒ grd. rezonabil de fiabilitate, expandabilitate și un raport cost/perf. rezonabil

Două clase:

• Multiprocesor ← din cls. strongly coupled: proc. au sist. com. de memorie (accesib. de toți din proc.)

Modulele de mem. sunt realizate după reguli practice (e de preferat o struct. cu mai multe module pt. că fiabilitatea e mai bună decât la un sist. cu un modul a cărui defectare compromite întreg sist.).

• Sist. puternic cuplate au mem. comună = mem. principală s fiec. µP poate avea ~~o~~ o mem. cache.

• Sist. slab cuplate

→au o mem. care a fost segmentată ~~și~~ se acordă fiec. µP câte o felie ⇒ ⇒mem. nu e comună, e locală ~~și~~ vizibilă doar de µP. căruia i se alocă
→ptr. a accesa un alt sgm. al altui µP e nevoie de timp, treceri prin conectoare, etc. ⇒complicat;
→fiecare nod al sist. e un calculator (comunică prin zone comune de mem., nu prin mesaje);

pg.9: SO → unic (se fol. doar unul)
→ poate fi: centralizat / distribuit / combinație (cea mai bună var.) (din sist.)

Din considerente legate de cost e necesar ca toate µPs să aibă aceeași struct. Fiecare µP. tb. să aibă un modul hw. aferent.

pg.10: µPs. mai multe:
- mem. comună compusă din m. multe module;
- struct. pp. că ~~multe~~ cel mult un sg. procesor are acces la mag. comună;
- (∀) µP tb. să poată avea acces la (∀) modul de mem.;
- sist. de acces simultan la mem. ⇒ se montează un arbitru de mag. care analiz. cererile simultane ⇒acordă priorități dinamice;
- dc. nr. de µPs. crește f. mult accesul la mem. începe să se complice; fiec. din µPs. tb. să ruleze partea de program cu taskul ce i-a fost asignat ⇒arbitraj acces. la mem. prg. nu este neapărat complicat, ci este consumator de timp ⇒ ptr. un nr. de µPs. mai mare de 32 structura cu mag. comună ridică pb. grele de rez. ⇒ tb. aleasă o altă rețea de interconectare;

pg.12: 10 MHz → T = 100 ns
64 biți, ptr. că fiec. unit. de prelucr. are µP cu 32 biți
→la 3 per. de ceas, o per. e fol. ptr. adresare, o per. ptr. verific. info.
→3 linii spec., altele decât mag., fol. ptr. semnaliz. unor ~~efem~~ evenim. infrecvente care au met. speciale de tratare (SLIC);
↳numele liniei

pg.14: Encore Multimax → alt calc antic;
→capac. dublă, lăț. de bandă 100 MB/s;

pg.15: → mag. comună are deci limitările ei prin creșterea nr. de µPs. la a. vtzș. de lucru;

→ dc. introd. mem. cache → crește vtza. de transmitere însă fiec. µP. th. se află modul propriu → coerența nu e real. însă → pb. se rezolvă dc. se utiliz. protocoale de comunic.;

pg.16: → dc. se defect. mag. → sist. compromis;

→ concurență mare → timp de acces mare;

pg.17: → µPs. nu se leagă direct la mag. comună, ci prin Bus Interface, care, pe lângă liniile de dte. adr. (linie groasă) are și alte linii: Bus Request (out.), Bus Grant (in.), Busy (in.);

→ Bus Req. și Grant sunt indiv. (fiecare proc. le are), Busy e comună ptr. toate µPs.;

→ µP1 va avea acces la mag. → analiz. st. lui Busy (spune dc. mag. e deținută de alt µP.) — e activă pe 0 (ușor de utiliz. open-collector)

Dc. Busy e activ → µP. respectiv "se abține" în cerere

Dc. Busy e inactiv → se manif. cerere Bus Req. (Bus Controller îi dă mag. necondiționată de cererea e unică; dc. sunt m. multe cereri pe bza. priorităților, Busy e pus pe 0, se activează Bus Grant pt. acel µP. cu priorit. max.)

→ arbitrul de mag. fct. sincronizat cu un ceas comun al mag.; cererile se analiz. la momente dictate de ceas; mag. comună permite deci acces doar la unul µP la resursele sist.;

pg.19: → altă struct. e cea în care se utiliz. m. multe magistrale (aici 3);

→ avem tot n µPs. și m module de mem.;

→ fiec. din mag. e tratată ca și când ar fi sg. mag. a sist.;

→ se analiz. starea fiec. mag. și se vede care are Busy inactiv și chiar dc. µP. e unic și mag. e liberă e posibil să nu aibă acces ptr. că ∃ sit. în care modulul la care vrea acces P1 să fie ocupat de P2 → chiar dc. P1 e sg. care efect. cererea de acces la o mag. care este liberă, datorită faptului că modulul de mem e ocupat µP. nu i se acordă mag.;

pg.21: struct. cu mag. bidim.

pg.22: → struct. cu ierarhii de memorii → mem. principală are niște copii cache pe un nivel și din cache-uri alte copii pe alt nivel sunt disponibile µPs.;

→ față de sit. anterioară, acum și mem. cache devine comună → accesul la mem. cache e mai rapid decât la mem. principală

→ dezavantaje: • mem. cache ridică pb. de consistență și are un vol. limitat → e necesară frecvent evacuarea / încărcarea de refill lines;