Arhitectura Sistemelor de Calcul Decembrie, 2021

Seminar 0x05

Cristian Rusu

1 Scopul seminarului

În acest seminar vom rezolva niște probleme care implică:

- ierarhizarea memoriei;
- performanța calculatoarelor;
- performanța sistemelor multi-core.

2 Exerciții

1. Presupunem că avem un sistem de calcul pe 32 de biți în care accesul la memoria RAM este $t_{\text{RAM}} = 50\text{ns}$, accesul la memoriile cache sunt: $t_{\text{L1}} = 1\text{ns}$ cu miss rate $m_{\text{L1}} = 10\%$, $t_{\text{L2}} = 5\text{ns}$ cu $m_{\text{L2}} = 1\%$ și $t_{\text{L3}} = 10\text{ns}$ cu $m_{\text{L3}} = 0.2\%$. Răspundeți la următoarele întrebări scurte:

   (a) calculați noua valoare m_{L1} pentru care timpul de acces la memorie este jumătate t_{RAM};
   1 ns + (A x (5 ns + (0.01 x (10 ns + (0.002 x 50 ns)))))

   (b) calculați noua valoare t_{L2} pentru care timpul de acces la memorie este o zecime din t_{RAM};
   1 ns + (0.1 x (A ns + (0.01 x (10 ns + (0.002 x 50 ns)))))

   (c) calculați noua valoare m_{L3} pentru care timpul de acces la memorie este același t_{RAM};
   1 ns + (0.1 x (5 ns + (0.01 x (10 ns + (A x 50 ns)))))

   (d) avem posibilitatea de a îmbunătăți timpii de răspuns pentru memoriile cache cu câte 10% pentru costurile $c_{\text{L1}} = 100\$$, $c_{\text{L2}} = 25\$$ și $c_{\text{L3}} = 5\$$ (costuri mai mari pentru cache mai rapid). Reduceți timpul de acces la memorie de o mie de ori față de t_{RAM} cu cost minim;

   (e) presupunem că avem, în general, memorii cache Li cu $t_{\text{Li}} = \left(\frac{i}{i+1}\right)^2 \times t_{\text{RAM}}$ și $m_{\text{Li}} = \frac{1}{(i+1)^2}$, pentru $i = 1, \ldots, n$. Sunt aceste valori consistente pentru $n \to \infty$? Care este timpul de acces la memorie în acest caz?

2. Presupunem că avem un sistem de calcul despre care vrem să știm anumite lucruri (legate de performanța sa). În scenarii diferite, ne interesează criterii de performanță diferite, fiecare având o metodă proprie de estimare. Rulăm aceleași programe de mai multe ori și trebuie să decidem care este funcția de agregare (media artimetică și geometrică, mediana, maximum și minimum). Ce criteriu de performanță am măsura și cum l-am estima în următoarele cazuri:

   (a) ne interesează să putem răspunde la cât mai multe cereri de date venite de pe Internet; utilizare CPU, media aritmetica

   (b) vrem să ne asigurăm că cererile de date sunt finalizate în 10ms; wall-clock time, media aritmetica

   (c) ne interesează ca cererile să ocupe maxim 100MB în memorie; memoria RAM, maximum

   (d) vrem să rulăm sistemul de calcul cu un cost cât mai mic; performanta per Watt, media airtmetica sau maximum

   (e) vrem să știm că majoritatea cererilor sunt servite în maximum 50ms; wall-clock time, 50/90/99th mediana

   (f) vrem să estimăm timpul total de răspuns al sistemului; wall-clock time speedup, media aritmetica

   (g) este vreodată folositor să estimăm performanța folosind minimul/maximul? zgomote minime/zgomote maxime

   (h) de ce este foarte important să măsurăm cât mai bine/exact performanța unui sistem de calcul?
   daca masuram bine si exact fiecare componenta, putem optimiza cat mai bine

   În general, un sistem de calcul trebuie să îndeplinească un anumit standard de funcționare (Quality of Service QoS sau Service Level Agreement SLA) care implică mai multe (de obicei cel puțin 2–3) dintre criteriile de performanță enumerate de mai sus.

Test	Program A	Program B
1	9	3
2	8	2
3	2	20
4	10	2

Table 1: Timpii de execuție (în secunde) pentru Program A și Program B.

3. Presupunem că avem Program A și Program B pe care le rulăm pe același sistem de calcul. Programele rulează de 4 ori iar timpii de execuție sunt prezentați în tabelul de mai sus. Cerințe:
   (a) calculați media aritmetică pentru timpii de execuție pentru fiecare program în parte și pentru fiecare test în parte.
   (b) de câte ori se execută mai repede Program B față de program A? de 3 ori
   (c) de câte ori se execută mai repede Program A față de program B? de 2.7 ori
   (d) ce fel de medie putem să folosim când comparăm cele două programe? media geometrica
   (e) care este cea mai bună metodă de a compara două programe între ele (head-to-head)?

4. Se dau două numere complexe $x = a + bi$ și $y = c + di$. Răspundeți la următoarele întrebări:
   (a) scrieți explicit formula pentru $z = x \times y$;
   (b) câte adunări și înmulțiri se realizează pentru calculul lui z? 2 adunari si 4 inmultiri
   (c) puteți să calculați z cu mai puține înmulțiri? da
   (d) de câte ori ar trebui să fie mai lentă o înmulțire față de a adunare pentru ca rezultatul de la punctul precedent să fie eficient?
   (e) ideea de a înlocui o înmulțire cu mai multe adunări (în general, o operație dificilă cu o serie de operații simple) apare de mai multe ori în algoritmică (vedeți algoritmul lui Strassen).

5. Presupunem că avem vectori $\mathbf{x}$ și $\mathbf{y}$ de dimensiune n și matrice $\mathbf{A}$ și $\mathbf{B}$ de dimensiune $n \times n$. Cerințe:
   (a) discutați cum calculați eficient produsul scalar: $z = \mathbf{x}^T\mathbf{y}$;
   (b) discutați cum calculați eficient produsul matrice-vector $\mathbf{w} = \mathbf{A}\mathbf{x}$;
   (c) discutați cum calculați eficient produsul matrice-matrice $\mathbf{C} = \mathbf{A}\mathbf{B}$ (vezi și Anexa 1).

6. Există multe feluri de a calcula îmbunătățirea de performanță care este posibilă într-un sistem multi-core. Considerăm că avem un program care are $0 \leq p \leq 1$ dintre instrucțiuni (interpretat ca procentaj) paralelizabile. Considerăm că avem la dispoziție $s \geq 1$ procesoare sau că sistemul beneficiază de o accelerare a performanței de δ ori. Două modalități de a calcula accelerarea execuției, legea lui Amdahl și legea lui Gustafson, respectiv:

$$S_{\text{Amdahl}} = \frac{1}{(1-p) + \frac{p}{s}} \quad \text{și} \quad S_{\text{Gustafson}} = 1 - p + \delta p. \tag{1}$$

   Cerințe:
   (a) calculați S_{Amdahl} și $S_{\text{Gustafson}}$ pentru $p = 0.5$, $s = 8$ și $\delta = 8$;
   (b) încercați să deduceți voi legile lui Amdahl și Gustafson;
   (c) ce se întâmplă cu S_{Amdahl} și $S_{\text{Gustafson}}$ dacă $p = 0$? Care este interpretarea rezultatului?

(d) calculați S_{Amdahl} și $S_{\text{Gustafson}}$ dacă $s = 1$ și $\delta = 1$? Care este interpretarea rezultatului?

(e) calculați $L_{\text{Amdahl}} = \lim_{s\to\infty} S_{\text{Amdahl}}$ și verificați dacă S_{Amdahl} este mereu sub sau peste această limită. Ce interpretare are valoarea L_{Amdahl}?

(f) calculați $L_{\text{Gustafson}} = \lim_{\delta\to\infty} S_{\text{Gustafson}}$;

(g) explicați diferențele dintre legile lui Amdahl și Gustafson.

7. Considerăm un sistem de calcul de 32 de biți. Sistemul poate realiza operațiile următoare: operații aritmetice/logice (1 ciclu), operații de citire/scriere date în memorie (2 cicli) și operații de branch/salt (3 cicli). Avem un program care are în componență 20% operații aritmetice/logice, 50% operații de citire/scriere (împărțite egal) și 30% operații de branch/salt. Cerințe:

   (a) calculați numărul mediu de cicli de ceas pe instrucțiune;

   (b) observăm că operațiile aritmetice/logice sunt mereu grupate cu operații de citire. Ne decidem să adăugăm o nouă instrucțiune care realizează și citirea și operația aritmetică/logică (totul într-un singur ciclu). Care este noul număr mediu de cicli de ceas pe instrucțiune?

   (c) care este timpul de execuție pentru program înainte și după modificarea de la punctul anterior (notăm numărul de instrucțiuni al programului cu N și frecvența sistemului cu f)?

   (d) suplimentar modificării de mai sus, presupunem că îmbunătățim sistemul de calcul și mărim frecvența cu 20%. Care este accelerarea de execuție a programului?

8. Considerăm că rulăm același program pe două sisteme diferite. Pe sistemul X, programul execută 80 de instrucțiuni aritmetice/logice, 40 de operații citire/scriere memorie și 25 de instrucțiuni de branch/salt. Pe sistemul Y, programul execută 50 de instrucțiuni aritmetice/logice, 50 de operații citire/scriere memorie și 40 de instrucțiuni de branch/salt. Pe ambele sisteme, operațiile aritmetice/logice au 1 ciclu, operațiile de citire/scriere au 3 cicli iar operațiile de branch/salt au 5 cicli. Cerințe:

   (a) calculați ponderea tipurilor de operații pentru fiecare sistem;

   (b) calculați numărul mediu de cicli de ceas pe instrucțiune pentru fiecare sistem;

   (c) presupunem că frecvența sistemului Y este cu 20% mai ridicată decât cea a sistemului X. Care sistem execută programul mai repede?

9. Considerăm un sistem de calcul pe 32 de biți care are un cache de 16 Kbytes. Cerințe:

   (a) considerăm că memorile sunt împărțite în blocuri de 32 de bytes. Câte blocuri sunt posibile în memoria principală? Câte blocuri sunt în memoria cache?

   (b) observați că numărul de blocuri din memoria principală poate să fie mult mai mare decât numărul de blocuri din memoria cache (e normal, memoria principală este mult mai multă decât memoria cache). Copiem bucăți din memoria principală în memoria cache pe blocuri. Trebuie să știm unde găsim în cache un byte pe care am vrea să îl citim din memoria principală. Sugerați moduri de a face corespondența între memoria principală și memoria cache.

10. Considerați secvența de cod din Anexa 2 unde vectorii au elemente pe 64 de biți (double float). Codul rulează pe un sistem de calcul de 32 de biți cu memorie cache L1 de 32 Kbytes cu blocuri de 64 bytes și memorie cache L2 de 1MB cu blocuri de 128 bytes. Cerințe:

    (a) ce face codul din Anexa 2?

    (b) câte blocuri sunt posibile în memorile cache L1 și L2?

    (c) cum arată rata de hit/miss pentru codul din Anexa 2?

    (d) rescrieți codul din Anexa 2 astfel încât să fie un singur ciclu for. Care este rata de hit/miss pentru noul cod?

Anexa 1

```
# varianta A
for (int i = 0; i < n; ++i)
  for (int j = 0; j < n; ++j)
     for (int k = 0; k < n; ++k)
        C[i][j] += A[i][k] * B[k][j];

# varianta B
for (int j = 0; j < n; ++j)
  for (int i = 0; i < n; ++i)
     for (int k = 0; k < n; ++k)
        C[i][j] += A[i][k] * B[k][j];

# varianta C
for (int k = 0; k < n; ++k)
  for (int i = 0; i < n; ++i)
     for (int j = 0; j < n; ++j)
        C[i][j] += A[i][k] * B[k][j];
```

Anexa 2

```
for (int i = 0; i < N; ++i)
    x[i] += vx[i] * dt;

for (int i = 0; i < N; ++i)
    y[i] += vy[i] * dt;

for (int i = 0; i < N; ++i)
    z[i] += vz[i] * dt;
```