Emulare, planificare si sabloane

Continut

- Emulare intre arhitecturi
- Problema planificarii sarcinilor
- Algoritmi de planificare
- Incarcare balansare

- Sabloane:
  - descompunerea sarcinilor,
  - descompunerea datelor,
  - sarcini de grup,
  - ordonarea sarcinilor,
  - partajare de date,
  - evaluarea designului

Emulare intre arhitecturi

- Motive:
  - dorința de a dezvolta rapid algoritmi pentru o nouă arhitectură fără a cheltui resursele semnificative care ar fi necesare pentru dezvoltarea algoritmului nativ.
  - dezvolta algoritmi pentru arhitecturi care sunt mai ușor de programat (de exemplu, memorie partajată sau PRAM) și apoi să le execute pe mașini realizabile, ușor accesibile sau accesibile.
- Algoritmul dezvoltat servește apoi ca un „cod sursă” care poate fi „compilat” rapid pentru a rula pe orice arhitectură dată prin emulare.
- Rezultatele emulării sunt uneori utilizate în alte scopuri decât portarea practică a software-ului.
- De exemplu, știm că hipercubul este o arhitectură puternică și poate executa mai mulți algoritmi în mod eficient. Astfel, o modalitate de a arăta că o nouă arhitectură este utilă sau eficientă, fără a fi nevoie să dezvolte un set mare de algoritmi pentru ea, este să arăți că poate emula hipercubul eficient.
- Dacă arhitectura A emulează arhitectura B cu O(f(p)) încetinire și B la rândul său, emulează C cu încetinirea O(g(p)) (presupunând, pentru simplitate, că toate conțin procesoare p), atunci A poate emula C cu încetinirea O(f(p)×g(p)).
- Problemă rezolvată cu teoria graficului: încorporarea unui grafic în altul.

Problema planificarii sarcinilor

- Având în vedere un sistem de sarcini care caracterizează un calcul paralel, se determina cum sarcinile pot fi atribuite resurselor de procesare (programate pe ele) pentru a satisface anumite criterii de optimitate
- Sistemul de sarcini este de obicei definit sub forma unui grafic direcționat, cu noduri care specifică sarcini de calcul și legături care corespund dependențelor de date sau comunicații
- Criteriile de optimitate pot include:
  - minimizarea timpului de execuție,
  - maximizarea utilizării resurselor,
  - minimizarea comunicării interprocesoare,
  - respectarea termenelor sau
  - o combinație a acestor factori.
- Parametrii sarcinii*:*
  - Durata de execuție sau de execuție: cel mai rău caz, cazul mediu sau distribuția probabilităților
  - Creare: un set fix de sarcini la un moment de compilare sau o distribuție a probabilităților pentru timpii de creare a sarcinilor
  - Relația cu alte sarcini: criticitate, ordine de prioritate și / sau dependențe de date.
  - Timpul de început sau de încheiere. Poate fi asociat un termen fix (hard) sau modificabil (soft) cu fiecare sarcină.
- *Resursele sau procesoarele pe care trebuie programate sarcinile sunt caracterizate de obicei prin capacitatea lor de a executa anumite clase de sarcini și de performanța sau viteza lor*.

Caracteristicile algoritmilor de planificare

1. Preventie.
   - Planificare fără prevenție: o sarcină trebuie să fie executată până la finalizare odată ce a început,
   - Planificare preventivă: executarea unei sarcini poate fi suspendată pentru a se acomoda cu o sarcină mai critică sau cu prioritate superioară.
   - În practică, preventia implică unele cheltuieli generale pentru stocarea stării sarcinii parțial finalizate și pentru verificarea continuă a cozii de sarcini pentru sosirea sarcinilor cu prioritate superioară.
     - Cu toate acestea, această depășire este ignorată în majoritatea algoritmilor de planificare din motive de tractabilitate.

2. Granularitate.
   - Problemele de planificare cu granulatie fina, medie sau grosiera se ocupă de sarcini de diferite complexități, de la simplele înmulțiri - adăugarea calculelor la segmente mari de programe complexe, constând poate din mii de instrucțiuni.
   - Planificarea granulara fina este adesea încorporată în alg, deoarece altfel surplusul ar fi prohibitiv.
   - Planificarea cu granulație medie și grosieră nu este fundamental diferită, cu excepția faptului că, cu un număr mai mare de sarcini de granulatie medie care trebuie programate, complexitatea de calcul a algoritmului de planificare poate deveni o problemă, în special cu programarea on-line sau în timpul rulării.

Complexitate

- Cele mai interesante probleme de planificare a sarcinilor sunt NP-complete.
  - Această dificultate a dat naștere rezultatelor cercetării în multe cazuri speciale care se pretează la soluții analitice și la numeroase proceduri euristice care funcționează bine în circumstanțe adecvate sau cu ajustarea parametrilor de decizie.
- Algoritmii de planificare optimă a timpului polinomial există doar pentru clase foarte limitate de probleme de planificare.
  - Exemplele includ programarea grafirile de sarcini structurate în arbore cu orice număr de procesoare și programarea grafiriloe arbitrare a sarcinilor în timp unitar pe două procesoare.
- Majoritatea problemelor practice de planificare sunt rezolvate prin aplicarea algoritmilor euristici.
  - Exemplu: programare listă.

Planificarea listelor

- Un nivel de prioritate este atribuit fiecărei sarcini.
- O listă de sarcini este construită în care sarcinile apar în ordinea priorităților, cu unele sarcini etichetate ca fiind pregătite pentru execuție. (inițial, sunt etichetate doar sarcinile care nu au o condiție prealabilă).
- Cu fiecare procesor care devine disponibil pentru executarea sarcinilor, sarcina marcată cu prioritate maximă este eliminată din listă și atribuită procesorului.
- Dacă q procesoare devin disponibile simultan, atunci sarcinile până la q pot fi programate simultan.
- Pe măsură ce sarcinile își completează execuțiile, satisfacând astfel condițiile preliminare ale altor sarcini, astfel de sarcini sunt etichetate și devin gata pentru execuție.
- Dacă toate procesoarele sunt identice, programatorii diferă în schemele lor de atribuire prioritare.
- În cazul sarcinilor cu timp unitar, etichetarea noilor sarcini se poate face imediat după finalizarea programării pentru pasul curent.
- Cu timpi de funcționare diferiți, dar deterministi, etichetarea se poate face prin atașarea unui timbru de timp cu fiecare sarcină care va deveni pregătită într-un timp viitor cunoscut.

Alte exemple de planificare

- Algoritmii de planificare sofisticate pot lua în considerare întârzieri în comunicare, termene, timp de lansare, cerințe de resurse, capacități de procesare și alți factori.
- Deoarece majoritatea algoritmilor de planificare nu garantează oricum timpi de funcționare optimi, trebuie să existe un echilibru între complexitatea planificatorului și performanța acestuia în ceea ce privește durata totală a programului obținut.
- Simplitatea programatorului este deosebit de importantă cu programarea on-line sau în timpul rulării, în care algoritmul de planificare trebuie să ruleze pe sistemul paralel însuși, folosind astfel timp și alte resurse precum sarcinile programate.
- În cazul programării off-line sau la timpul compilarii, timpul de rulare al planificatorului este mai puțin problematic.
- Dacă în luarea deciziilor de planificare sunt luați în considerare parametrii de sincronizare, cum ar fi termenele de sarcină și timpii de eliberare, avem o problemă de planificare în timp real.
  - Exemple de strategii de planificare pentru sarcini în timp real includ „primul termen limită cel mai apropiat”.
- Atunci când posibilitatea de a eșua pentru procesoare și alte resurse este luată în considerare în sarcina sau reasignarea sarcinilor, avem o planificare tolerantă la erori.

Planificare distribuita

- Distribuita:
  - distribuie inițial sarcinile între procesoarele disponibile, pe baza unor criterii,
  - permite fiecărui procesor să își facă propria planificare internă (ordonând executarea setului său de sarcini) în funcție de interdependențele sarcinilor și rezultatelor primite de la alte procese.
  - Avantajul acestei abordări este că majoritatea deciziilor de planificare sunt realizate într-o manieră distribuită.
  - Un posibil dezavantaj este că rezultatele pot fi departe de a fi optime.
  - Dacă o astfel de schemă este combinată cu o metodă de redistribuire a sarcinilor atunci când distribuția inițială se dovedește a fi inadecvată, se pot obține rezultate bune.
- Balansarea incarcarii?
  - Să presupunem că sarcinile sunt distribuite procesoarelor în așa fel încât timpul de rulare total preconizat al sarcinilor atribuite fiecărui procesor să fie aproximativ același.
  - Deoarece timpii de execuție a sarcinilor nu sunt constante, un procesor poate rămâne fără a face lucrurile înainte ca alți procesatori să-și termine sarcinile alocate.
  - De asemenea, unele procesoare pot rămâne inactive pentru perioade îndelungate de timp, deoarece așteaptă executarea sarcinilor prealabile pe alte procesoare.
- În aceste cazuri, o politică de balansarea incarcarii poate fi aplicată în încercarea de a uniformiza distribuția încărcării.
  - schimb la sarcini încă neexecutate de la un procesor supraîncărcat la unul mai puțin încărcat.
  - echilibrarea sarcinii poate fi inițiată de un procesor inactiv sau ușor încărcat (inițiat de receptor) sau de un procesor supraîncărcat (inițiat de expeditor).

Surplusul balansarii incarcarii

- Balansarea incarcarii poate implica un surplus mare care reduce potențialele câștiguri.
- Dacă mutarea unui sarcini de la un proc. la altul înseamnă copierea unui program mare cu cantități uriașe de date și actualizarea diferitelor tabele de stare pentru a indica noua locație a sarcinii, atunci comunicarea generală este semnificativă
  - Incarcarea balansarii poate să nu merite costul acesteia.
- Dacă sarcinile aparțin unui set de sarcini standard, fiecare fiind invocat cu un set mic de parametri (date) și cu copii deja disponibile local la fiecare procesor, atunci mutarea sarcinilor poate implica doar un mesaj mic de difuzare pentru a trece parametrii. și actualizați tabelele de stare a sistemului.
  - Surplusul va fi minimal.
- În rețelele cu comutare de circuit care utilizează rutarea găurilor de vierme, problema de balansarea incarcarii poate fi formulată ca o problemă de flux in rețea.
  - Excesul (deficiența) sarcinii de lucru la unele noduri poate fi privit ca surse de curgere (chiuvete), iar cerința este de a permite excesul de sarcină de lucru să „curgă” de la surse pe căi care sunt, în măsura posibilului, disjuncte și astfel scutite de conflicte.

Sisteme cu auto-planificare

- Un sistem cu auto-planificare:
  - Încearcă să mențină toate resursele de procesare la o eficiență maximă.
  - Poate exista o locație centrală la care procesatorii se referă și unde își returnează rezultatele.
  - Un procesor inactiv solicită să i se atribuie o nouă lucrare, trimițând un mesaj acestui supraveghetor central și în schimb primește una sau mai multe sarcini pentru a efectua
  - Funcționează bine pentru sarcini cu contexte mici și / sau timp de rulare relativ lung.
- O implementare la nivel hardware a unei astfel de scheme de auto-planificarea, cunoscută sub numele de calculul fluxului de date, si are o istorie lungă.
  - Un calcul de flux de date este caracterizat printr-un grafic de flux de date, care este foarte similar cu grafurile de sarcini, dar poate conține elemente de decizie și bucle.
  - Dacă un arc este restricționat să nu poarte mai mult de un simbol: un sistem de flux de date static.
  - Dacă mai multe jetoane etichetate pot apărea pe arce și sunt „consumate” după potrivirea etichetelor lor, avem un sistem dinamic de flux de date
    - permite pipelinarea calculelor, dar implică o depășire mai mare ca urmare a cerinței de potrivire a etichetelor token.
  - Implementarea hardware a sistemelor de flux de date cu calcule cu granulatie fina (una sau câteva instrucțiuni ale mașinii pe nod), deși posibilă, s-a dovedit impractica.
    - Când fiecare nod sau sarcină este un fir de calcul format din secvențe mai lungi de instrucțiuni ale mașinii, atunci surplus activarii este mai puțin serioasă și conceptul devine destul de practic.

Sabloane in spatial de design

- **Sabloane de descompunere.**
  - Descompunerea sarcinilor
  - Descompunerea datelor
- **Sabloane de analiza dependentelor.**
  - Sarcini de grup,
  - Ordonarea sarcinilor,
  - Partajarea datelor
- **Sabloane de evaluare a designului.**
  - important, deoarece se întâmplă adesea că cel mai bun design nu este găsit la prima încercare, iar cu cat defectele de proiectare anterioare sunt identificate mai repede, cu atât sunt mai ușor de corectat.
- În general, lucrul cu sabloanele în acest spațiu este un proces iterativ.

Descompunerea sarcinilor vs. a datelor

- Dimensiunea de descompunere a sarcinii privește problema ca un flux de instrucțiuni care pot fi defalcate în secvențe numite sarcini care se pot executa simultan.
  - operațiunile care compun sarcina ar trebui să fie în mare măsură independente de operațiunile care se desfășoară în cadrul altor sarcini.
- Dimensiunea de descompunere a datelor se concentrează pe datele cerute de sarcini și cum pot fi descompuse în bucăți distincte.
  - Calculul asociat cu bucățile de date va fi eficient numai dacă bucățile de date pot fi operate relativ relativ independent.
- Vizualizarea descompunerii problemei în termeni de două dimensiuni distincte este oarecum artificială.
  - O descompunere a sarcinilor presupune o descompunere a datelor și invers;
  - Cele două descompuneri sunt fațete cu adevărat diferite ale aceleiași descompuneri fundamentale.
  - Împărțim în dimensiuni separate, deoarece o descompunere problemă se desfășoară, de obicei, cel mai natural, subliniind o dimensiune a descompunerii față de cealaltă.
  - Făcându-le distincte, facem ca acest accent să fie explicit și mai ușor de înțeles de către proiectant.

Ex.1: Imagistica medicala

- Scanările PET (tomografie cu emisie de pozitroni) oferă un instrument de diagnostic important, permițând medicilor să observe modul în care o substanță radioactivă se propagă prin corpul pacientului.
  - Imaginile formate din distribuția radiațiilor emise sunt de rezoluție scăzută, datorate în parte împrăștierii radiației pe măsură ce trece prin corp.
  - De asemenea, este dificil să se concluda pe baza intensitățile absolute ale radiațiilor, deoarece diferite căi prin corp atenuează radiația în mod diferit.
- Pentru a rezolva această problemă, sunt utilizate modele de propagare a radiațiilor prin corp pentru a corecta imaginile.
- O abordare comună este construirea unui model Monte Carlo.
  - Se presupune că punctele selectate aleatoriu în interiorul corpului emit radiație (de obicei o rază gamma) și se urmărește traiectoria fiecărei raze.
  - Pe măsură ce o particulă (rază) trece prin corp, este atenuată de diferitele organe pe care le traversează, continuând până când particulele părăsesc corpul și lovește un model de cameră, definind astfel o traiectorie completă.
  - Pentru a crea o simulare semnificativă statistic, sunt urmate mii sau milioane de traiectorii.
- Această problemă poate fi paralizată în două moduri.
  - Deoarece fiecare traiectorie este independentă, este posibilă paralelizarea aplicației prin asocierea fiecărei traiectorii cu o sarcină.
  - O altă abordare ar fi divizarea corpului în secțiuni și atribuirea diferitelor secțiuni diferitelor elemente de procesare.

Ex. 2: inmultirea matricelor

- Dacă A, B și C sunt matrici pătrate de ordinul *N*, înmulțirea matricei este definită astfel încât fiecare element al matricei C rezultă în cazul în care abonamentele notează elemente particulare ale matricelor.
- Cu alte cuvinte, elementul matricei produsului C din rândul *i* și coloana *j* este produsul punct al rândului *i*th din A și coloana a j a lui B.
- Prin urmare, calcularea fiecăruia dintre N^2 elemente a lui *C* necesita *N* multiplicari si *N*-1 adunari, realizând complexitatea generală a înmulțirii matricei $O(N^3)$.
- Există multe modalități de a paralela o operație de înmulțire a matricei.
  - Poate fi paralizat folosind fie o descompunere bazată pe sarcini, fie o descompunere bazată pe date

Ex. 3: Problema celor N corpuri

- Dinamica moleculară este utilizată pentru a simula mișcările unui sistem molecular mare.
  - De exemplu, simulările dinamicii moleculare arată modul în care o proteină mare se mișcă și cum medicamentele cu formă diferită ar putea interacționa cu proteina.
  - Dinamica moleculară este extrem de importantă în industria farmaceutică.
  - De asemenea, este o problemă utilă de testare pentru oamenii de informatică care lucrează la calcul paralel: este simplu de înțeles, relevant pentru știință în general și dificil de paralelizat eficient.
- Ideea de bază este de a trata o moleculă ca o colecție mare de bile conectate prin arcuri
  - bilele reprezintă atomii din moleculă,
  - arcurile reprezintă legăturile chimice dintre atomi.
- Simularea dinamicii moleculare în sine este un proces explicit în timp.
  - La fiecare pas de timp, se calculează forța pe fiecare atom și apoi se folosesc tehnici standard de mecanică clasică pentru a calcula modul în care forța mișcă atomii.
  - Acest proces este realizat în mod repetat pentru a face pas în timp și pentru a calcula o traiectorie pentru sistemul molecular.
- Forțele datorate legăturilor chimice („arcurile”) sunt relativ simple de calculat.
  - Acestea corespund vibrațiilor și rotirilor legăturilor chimice în sine.
  - Acestea sunt forțe cu rază scurtă de acțiune care pot fi calculate cu cunoașterea mîinii de atomi care împărtășesc legături chimice.
  - Dificultatea majoră apare deoarece atomii au sarcini electrice parțiale.
  - Prin urmare, în timp ce atomii interacționează doar cu un mic cartier de atomi prin legăturile lor chimice, sarcinile electrice determină fiecare atom să aplice o forță pe fiecare alt atom.

Aceasta este problema celor N corpuri.

Ex. 3: Problema celor N corpuri

- Ordinea in care cei N^2 termenii trebuie calculați pentru a găsi forțele care nu sunt legate.
  - Deoarece N este mare (zeci sau sute de mii) și numărul de pași de timp într-o simulare este uriaș (zeci de mii), timpul necesar pentru calcularea acestor forțe care nu sunt legate domină calculul.
- Au fost propuse mai multe moduri de a reduce efortul necesar pentru rezolvarea problemei.
  - Cea mai simplă: metoda cutoff.
    - Chiar dacă fiecare atom exercită o forță asupra fiecărui alt atom, această forță scade odată cu pătratul distanței dintre atomi.
    - Prin urmare, ar trebui să fie posibil să alegeți o distanță dincolo de care contribuția forței este atât de mică încât poate fi ignorată.
    - Prin ignorarea atomilor care depășesc această limită, problema se reduce la una care se scalează ca O (N x n), unde n este numărul de atomi din volumul cutoff, de obicei sute.
    - Calculul este încă uriaș și domină timpul de rulare total pentru simulare, dar cel puțin problema este tratabilă.
- Structurile de date primare țin pozițiile atomice (atomi), viteza fiecărui atom (viteza), forțele exercitate asupra fiecărui atom (forțele) și listele de atomi în distanța de tăiere a fiecărui atom (vecini).
- Programul în sine este o buclă de pas în timp,
  - În care fiecare iterație calculează termenii de forță de rază scurtă, actualizează listele de vecini și apoi găsește forțele nelegate.
  - După ce forța pe fiecare atom este calculată, o ecuație diferențială simplă ordinară este rezolvată pentru a actualiza pozițiile și vitezele.
  - Proprietățile fizice bazate pe mișcări atomice sunt apoi actualizate

Sablonul de descompunere a sarcinilor

- Cheia pentru o descompunere eficientă a sarcinilor este să:
  - se asigura că sarcinile sunt suficient de independente, astfel încât gestionarea dependențelor ocupă doar o mică parte din timpul total de execuție al programului.
  - se asigura că execuția sarcinilor poate fi distribuită uniform între ansamblul de PE (problema de balansarea incarcarii).
- Realizat manual, bazat pe cunoașterea problemei. & codul necesar pentru rezolvarea acestuia.
- Sarcinile pot fi găsite în multe locuri diferite:
  - În unele cazuri, fiecare sarcină corespunde unui apel distinct la o funcție.
    - Definirea unei sarcini pentru fiecare apel funcțional duce la ceea ce se numește uneori descompunere funcțională.
  - Un alt loc pentru a găsi sarcini este în iterațiile distincte ale buclelor dintr-un algoritm.
    - Dacă iterațiile sunt independente și există suficiente, atunci ar putea funcționa bine să se bazeze o descompunere a sarcinii pe maparea fiecărei iterații pe o sarcină.
    - Acest stil de descompunere bazat pe sarcini duce la ceea ce se numește uneori algoritmi de divizare a buclelor.
  - Sarcinile joacă, de asemenea, un rol cheie în descompunerile bazate pe date.
    - În acest caz, o structură de date mare este descompusă și mai multe unități de execuție actualizează simultan diferite bucăți ale structurii de date.
    - În acest caz, sarcinile sunt acele actualizări pe bucăți individuale.

Descompunerea sarcinilor- exemple

- Ex1 – imagistica medicala
  - Este firesc să asociați o sarcină cu fiecare traiectorie.
  - Aceste sarcini sunt deosebit de simple de gestionat concomitent, deoarece sunt complet independente
  - Sarcinile au nevoie de acces la modelul corpului
  - Modelul caroseriei poate fi extrem de mare.
  - Deoarece este un model cu citire numai, nu există nicio problemă dacă există un sistem eficient de memorie partajată;
  - Dacă platforma țintă se bazează pe o arhitectură de memorie distribuită, totuși, modelul de caroserie va trebui să fie reprodus pe fiecare PE.
    - Acest lucru poate dura foarte mult și poate pierde o mare cantitate de memorie.
    - dacă memoria și / sau lățimea de bandă a rețelei sunt un factor limitativ, o descompunere care se concentrează pe date ar putea fi mai eficientă.
- Ex2 – inmultire matriceala
  - Natural: calculul fiecărui element al matricei produsului ca o sarcină separată
  - Simplu de implementat într-un mediu cu memorie partajată.
  - Timpul de acces la memorie este lent în comparație cu aritmetica cu punct flotant, astfel că lățimea de bandă a subsistemului de memorie ar limita performanța.
  - O abordare mai bună ar fi proiectarea unui algoritm care să maximizeze reutilizarea datelor încărcate în cache-urile unui procesor.

Descompunerea sarcinilor- exemple

- Ex3 – dinamica moleculara
  - o definiție a sarcinii naturale este actualizarea necesară fiecărui atom, care corespunde unei iterații de buclă în versiunea secvențială.
  - După efectuarea descompunerii sarcinii, se obțin următoarele sarcini:
    - Sarcini care găsesc forțele vibraționale pe un atom
    - Sarcini care găsesc forțele de rotație pe un atom
    - Sarcini care găsesc forțele nelegate pe un atom
    - Sarcini care actualizează poziția și viteza unui atom
    - O sarcină pentru a actualiza lista de vecini pentru toți atomii.
  - Structurile cheie de date sunt lista vecinilor, coordonatele atomice, viteza atomică și vectorul de forță.
  - Fiecare iterație care actualizează vectorul de forță are nevoie de coordonatele unui cartier de atomi.
  - Cu toate acestea, calculul forțelor care nu au legătură are nevoie de coordonatele tuturor atomilor.

Sablonul de descompunerea datelor

- În mediile de programare cu memorie partajată, cum ar fi OpenMP, descompunerea datelor va fi frecvent implicată de descompunerea sarcinii.
- În majoritatea cazurilor, însă, descompunerea va trebui făcută manual, deoarece memoria este distribuită fizic, deoarece dependențele de date sunt prea complexe fără a se descompune explicit datele sau pentru a obține o eficiență acceptabilă
- Dacă s-a făcut deja o descompunere bazată pe sarcini, descompunerea datelor este determinată de nevoile fiecărei sarcini.
- Dacă pot fi asociate date bine definite și distincte cu fiecare sarcină, descompunerea ar trebui să fie simplă.
- Când se începe cu o descompunere a datelor, trebuie să ne uităm nu la sarcini, ci la structurile centrale de date care definesc problema și să ne gândim dacă pot fi defalcate în bucăți care pot fi operate simultan.
- Câteva exemple comune includ următoarele.
  - Calcule matriceale.
    - Concurența poate fi definită în termeni de actualizări ale diferitelor segmente ale tabloului.
    - Array multidimensional poate fi descompus într-o varietate de moduri (rânduri, coloane sau blocuri).
  - Structuri de date recursive.
    - De exemplu, descompunerea actualizării paralele a unei structuri de date arbore mari prin descompunerea structurii de date în subtrape care pot fi actualizate concomitent.

Descompunerea datelor - exemple

- Ex1 – imagistica medicala
  - Modelul corpului este marea structură centrală de date în jurul căreia poate fi organizată calcularea.
  - Modelul este împărțit în segmente și unul sau mai multe segmente sunt asociate cu fiecare element de procesare.
  - Segmentele corpului sunt citite, nu scrise, în timpul calculelor de traiectorie, deci nu există dependențe de date create prin descompunerea modelului corpului.
  - Fiecare traiectorie care trece prin segmentul de date definește o sarcină.
  - Traiectoriile sunt inițiate și propagate în cadrul unui segment.
  - Când se întâlnește o graniță de segment, traiectoria trebuie trecută între segmente.
  - Acest transfer este cel care definește dependențele dintre bucățile de date.
- Ex2 – inmultirea matricela
  - descompunerea matricea produsului C într-un set de blocuri de rând (set de rânduri adiacente).
  - o abordare și mai eficientă care nu necesită replicarea matricei complete A este de a descompune toate cele trei matrici în submatrici sau blocuri.

Descompunerea datelor - exemple

- Ex3 – dinamica moleculara
  - Structurile cheie de date sunt:
    - O serie de coordonate de atom, un element pe atom
    - O serie de viteze ale atomului, un element per atom
    - O serie de liste, una pe atom, fiecare definind vecinătatea atomilor în distanța de decuplare a atomului
    - O serie de forțe asupra atomilor, un element pe atom.
  - Un element al tabloului de viteză este utilizat numai de sarcina care deține atomul.
    - Aceste date nu trebuie să fie partajate și pot rămâne locale pentru sarcină.
    - Cu toate acestea, fiecare sarcină are nevoie de acces la o gamă completă de coordonate.
    - Astfel, va avea sens să replicăm aceste date într-un mediu de memorie distribuit sau să le împărtășim între UE într-un mediu de memorie comună.
  - Mai interesantă este gama de forțe.
    - Din a treia lege a lui Newton, forța din atomul i asupra atomului j este negativul forței din atomul j pe atomul i.
      - Exploatați această simetrie: tăiați cantitatea de calcul în jumătate pe măsură ce acumulăm termenii de forță.
    - Valorile din tabloul de forțe nu sunt în calcul până la ultimii pași în care sunt actualizate coordonatele și vitezele.
    - Prin urmare, abordarea folosită este inițializarea întregului tablou de forțe pe fiecare PE și ca sarcinile să acumuleze sume parțiale ale termenilor de forță în acest tablou.
    - După ce s-au finalizat toți termenii de forță parțială, adunăm toate tablele PE pentru a asigura tabloul de forță final.

Sablonul sarcinilor de grup

- Problemă: Cum pot fi grupate sarcinile care alcătuiesc o problemă pentru a simplifica munca de gestionare a dependențelor?
- ideea este de a defini grupuri de sarcini care împărtășesc constrângerile și simplifică problema gestionării constrângerilor prin tratarea grupurilor, mai degrabă decât cu sarcini individuale.
- Constrângerile dintre sarcini se încadrează în câteva categorii majore.
  - Cea mai ușoară dependență de înțeles este o dependență temporală - adică o constrângere a ordinii în care se execută o colecție de sarcini..
    - Dacă sarcina A depinde de rezultatele sarcinii B, de exemplu, sarcina A trebuie să aștepte până la finalizarea sarcinii B înainte de a putea executa.
  - Un alt tip de constrângere: când o colecție de sarcini trebuie să fie executată în același timp.
    - Ex: domeniul problemei inițiale este împărțit în mai multe regiuni care pot fi actualizate în paralel și actualizarea oricărei regiuni necesită informații despre limitele regiunilor învecinate.
    - Dacă toate regiunile nu sunt procesate în același timp, programul paralel s-ar putea bloca sau a impune, deoarece unele regiuni așteaptă date din regiuni inactive.
  - În unele cazuri, sarcinile dintr-un grup sunt cu adevărat independente unele de altele.
    - Pentru că înseamnă că se pot executa în orice ordine, inclusiv.
- Scopul acestui model este de a grupa sarcini bazate pe aceste constrângeri, deoarece
  - Prin gruparea sarcinilor, simplificăm stabilirea comenzilor parțiale între sarcini, deoarece constrângerile de comandă pot fi aplicate grupurilor și nu sarcinilor individuale.
  - Gruparea sarcinilor face mai ușor să identificați ce sarcini trebuie executate concomitent.

Sarcini de grup - exemple

- Ex – multiplicarea matriceala
  - actualizarea unui element în C.
  - Organizarea de memorie a majorității calculatoarelor moderne, însă, favorizează sarcini mai mari, precum actualizarea unui bloc de C.
  - Din punct de vedere matematic, aceasta este echivalentă cu gruparea sarcinilor de actualizare elementară în grupul corespunzător blocurilor, iar gruparea sarcinilor în acest fel este potrivită pentru o utilizare optimă a memoriei de sistem.
- Ex – dinamica moleculara
  - Sarcini care găsesc forțele vibraționale pe un atom:
    - Sarcini care găsesc forțele de rotație pe un atom
    - Sarcini care găsesc forțele nebancate pe un atom
    - Sarcini care actualizează poziția și viteza unui atom
    - O sarcină pentru a actualiza lista de vecini pentru toți atomii.
  - Luați în considerare modul în care acestea pot fi grupate.
    - Fiecare element din lista precedentă corespunde unei operațiuni la nivel înalt din problema inițială și definește un grup de sarcini.
    - În fiecare caz, actualizările implicate în funcțiile de forță sunt independente - singura dependență este însumarea forțelor într-un singur tablou de forțe.
    - Sarcinile din primele două grupuri sunt independente, dar au aceleași constrângeri.
    - În ambele cazuri, se citesc coordonatele pentru un cartier mic de atomi și se aduc contribuții locale la matricea de forțe => îmbină acestea într-un singur grup pentru interacțiuni legate.
    - Celelalte grupuri au constrângeri temporale => nu ar trebui să fie comasate.

Sablonul de ordonare a sarcinilor

- Problemă: Având în vedere un mod de descompunere a unei probleme în sarcini și un mod de colectare a acestor sarcini în grupuri legate logic, cum trebuie să fie ordonate aceste grupuri de sarcini pentru a satisface constrângerile dintre sarcini?
- Constrângerile dintre sarcini se încadrează în câteva categorii majore:
  - Dependențe temporale
  - Cerințe pe care anumite sarcini trebuie să le execute în același timp (ex: deoarece fiecare necesită informații care vor fi produse de ceilalți).
  - Lipsa constrângerii, adică independența totală.
- Scopul acestui model este de a ajuta la găsirea și contabilizarea corectă a dependențelor rezultate din constrângerile privind ordinea executării unei colecții de sarcini.
- Există două obiective care trebuie îndeplinite atunci când se identifică constrângerile de comandă între sarcini și se definește o ordine parțială între grupurile de sarcini.
  - Ordonarea trebuie să fie suficient de restrictivă pentru a satisface toate constrângerile, astfel încât designul rezultat să fie corect.
  - Ordonarea nu ar trebui să fie mai restrictivă decât trebuie.
    - Restrângerea excesivă a soluției limitează opțiunile de proiectare și poate afecta eficiența programului;
    - Cu cât sunt mai puține constrângerile, cu atât mai multă flexibilitate trebuie să schimbați sarcinile pentru a echilibra sarcina de calcul între PE.

Ordonarea sarcinilor – ex: dinamica moleculara

- Anterior au fost organizat sarcinile pentru această problemă în următoarele grupuri:
  - Un grup de sarcini pentru a găsi „forțele legate” (forțele vibraționale și forțele de rotație) pe fiecare atom
  - Un grup de sarcini care să găsească forțele nebancate pe fiecare atom
  - Un grup de sarcini pentru actualizarea poziției și vitezei fiecărui atom
  - O sarcină pentru a actualiza lista de vecini pentru toți atomii (care constituie în mod banal un grup de sarcini)
- Acum fie ordonarea constrângerilor între grupuri.
  - Actualizarea pozițiilor atomice nu poate avea loc până când calculul forței este complet.
  - Forțele nerezolvate nu pot fi calculate până când lista vecinilor nu este actualizată.
- Cum vor fi aplicate aceste constrângeri de ordonare: un fel de sincronizare pentru a se asigura că sunt respectate cu strictețe.

Sablonul de partajare a datelor

- Problemă. Având în vedere o descompunere de date și sarcini pentru o problemă, cum se împart datele între sarcini?
- Scopul acestui model este
  - pentru a identifica ce date sunt partajate între grupuri de sarcini
  - determina cum să fie gestionat accesul la datele partajate într-un mod corect și eficient.
- Primul pas este identificarea datelor care sunt partajate între sarcini.
  - Acest lucru este cel mai evident atunci când descompunerea este predominant o descompunere bazată pe date.
- După identificarea datelor partajate, acesta trebuie analizat pentru a vedea cum sunt utilizate.
- Datele partajate se încadrează într-una dintre următoarele trei categorii.
  - Numai citit
  - Efectiv local.
  - Citeste si scrie.

Partajarea datelor – ex.: dinamica moleculara

- Principalele elemente de date partajate sunt următoarele.

1. Coordonatele atomice, utilizate de fiecare grup.
   - Aceste coordonate sunt tratate ca date numai în citire de către grupul de forțe legate, grupul de forțe non-afectate și grupul de actualizare a vecinilor.
   - Aceste date sunt citite în scris pentru grupul de actualizare a poziției.
   - Grupul de actualizare a poziției se execută singur după terminarea celorlalte trei grupuri
   - În primele trei grupuri, lăsați accesul la datele de poziție neprotejate sau chiar reproduceți-le.
   - Pentru grupul de actualizare a poziției, datele de poziție aparțin categoriei de scriere citită, iar accesul la aceste date va trebui controlat cu atenție.
2. Matricea de forță, folosită de fiecare grup, cu excepția actualizării listei vecine.
   - Este utilizat ca date numai în citire de către grupul de actualizare a poziției și ca acumulare de date pentru grupurile de forțe legate și nelegate.
   - Acest tablou poate fi introdus în categoria acumulării pentru grupurile de forțe și în categoria numai de citire pentru grupul de actualizare a poziției.
   - Procedura standard pentru simulările dinamicii moleculare începe prin inițializarea tabloului de forțe ca un tablou local pe fiecare UE.
   - Contribuțiile la elemente ale tabloului de forțe sunt apoi calculate de fiecare UE, termenii exacti calculați fiind imprevizibili din cauza modului în care molecula se pliază în spațiu.
   - După ce toate forțele au fost calculate, tablourile locale sunt reduse într-un singur tablou, a cărui copie este plasată pe fiecare UE.
3. Lista de vecini, împărtășită între grupul de forțe nelimitate și grupul de actualizare a listei de vecini.
   - În esență, sunt date locale pentru grupul de actualizare a listei vecine și citesc numai datele pentru calculul forței neobișnuite.
   - Lista poate fi gestionată în stocare locală pe fiecare UE.

Sablonul de evaluare a designului

- Problemă: Analiza descompunerii și dependenței este suficient de bună pentru a trece la următorul spațiu de design sau ar trebui revizuit designul?
- Proiectarea trebuie evaluată din trei perspective.
  1. Potrivire pentru platforma țintă.
     - Probleme precum nr. procesoarele și modul în care structurile de date sunt partajate vor influența eficiența oricărui proiect,
     - Dar cu cât designul depinde de arhitectura țintă, cu atât va fi mai puțin flexibil.
  2. Calitatea designului.
     - Simplitatea, flexibilitatea și eficiența sunt toate atributele de dorit - dar, probabil, contradictorii.
  3. Pregătirea pentru următoarea fază a designului.
     - Sarcinile și dependențele sunt regulate sau neregulate?
       - adică au dimensiuni similare sau variază?
     - Este interacțiunea dintre sarcini sincrone sau asincrone?
       - adică interacțiunile au loc la intervale regulate sau ori foarte variabile sau chiar aleatorii?
     - Sarcinile sunt agregate într-un mod eficient?