F/s = Hz * N (N etapa kopurua)

Hainbat prozesadore lan komun bat egiten

Prozesua: exekutatzen dagoen programa

Multithreading: hainbat hari prozesadore ezberdinetan edo logikoki ezberdinduak

Haria: sistema eragileakkudeatu/planifikatu dezakeen prozesu minimoa. Beste harirekin batera exekuta daiteke paraleloan; denek partekatzen dute memoria-espazio bera.

Prozesamendu-baliabideak partekatuak, exekuzio tartekatua

Testuinguru-aldaketa, hari batetik beste batera aldatu, oso azkarra, prozesu aldaketa baino gehiago

Hari xeheko multithreading:
- -Ziklo bakoitzean haria aldatu daikete
- -Aginduen exekuzioa tartekatua da, Round Robin
- -ARAZOAK: hari batek arazorik gabe jarrai badezake, abiadura jaitsiko zaioa, tartekatzea dela eta

Hari larriko multithreading:
- -Hari aldaketa gelkite luzeetan bakarrik
- -ARAZOAK: ez dira geldiketa motzak aprobetxatzen

Simultaneous Multithreading:
- -Ziklo bakoitzen edozein haritako edozein agindu exekutatu
- -UFak libre -> hari independienteetarako
- -Erregistro asko testuinguruak kudeatzeko
- -Hari arteko dependentziak ohiko moduan kudeatu
- -Hariak tartekatu daitezke erregistroak nahastu gabe
- -BB (ROB) bat hariko

Multicore:
- -Txip batean hainbat nukleo/core ezberdin
- -Nukleo bakoitzak bere baliabide propioak
- -Prozesu independenteak simultaneoki exekutatzen dituzte
- -Nukleoa multithreading motakoa izan daiteke
- -Memoria partekatua

SIMD:
- -Eragiketa mota bat bat azkar exekutatzeko
- -Bektore prozesadoreak
- -Prozesadore grafikoak, GPU

MIMD:
- -Helburu orokorreko makina paraleloak
- -Hainbat prozesu paralelo problema bat ebazteko
- -Prozesuak elkarlanean -> prozesu arteko komunikazioa
- -Memoria partekatua, komuniakazioa erraza, sinkronizazio beharra.
- -Memoria pribatua, komunikazioa esplizitatu beharra

Maiz, paralelismoa ez da erabiltzen azkarrago exekutatzeko, baizik eta tamaina handiagoko atazak exekutatu ahal izateko.

Eskalagarritasun sendoa -> lan-karga gutxiago prozesadoreko -> Amdhal

Eskalagarritasun ahula -> lan-karga bera prozesadoreko -> Gustafson

Hirugarren aukera ataza independente asko aldi berean prozesatzea. Ataza bakoitzaren abiadura bera, multzoaren jeitsi.

Write-through: idazketak memoriako maila guztietan, kopiak beti berdinak

Write-back: behe mailan bakarrik idatzi. Kopiak ezberdinak, blokea ordezkatzean memoriako hurrengo mailan salbatu

Datu-blokeen kopiak:
- -Aldagai partekatuak
- -Aldagaiek bloke berean koinziditu, nahiz eta pribatuak izan

Memoria sistemak koherentea izan behar du
- -Aldagai bat irakurtzea azkena idatzitako balioa bada
- -Idazketak orden berean edozein prozesadoretan

SMP:
- -Prozesadore gutxi
- -Memoria zentralizatua
- -Busak
- -Zelatatze motako automatak (snoopy)

DSM:
- -Prozesadore asko
- -Memoria banatua
- -Komunikazio-sare banatua
- -Koherentzia-direktoria

SMP batean aldaketak dauden jakiteko busa espiatu (snoopy-az) eta kontrol-seinale bereziak abisatzeko.

Kopia bat aldatzea baliogabetu besteak, ondoren eguneratu balio berriaz

Bost datu egoera
- -I: baliogabea
- -E: esklusiboa, kopia bakarra, koherentea
- -M: aldatua, kopia bakarra, ez koherentea
- -S: partekatua, hainbat kopia, denak koherenteak
- -O: jabea, hainbat kopia, ez koherenteak

Zelatariak bertako prozesadoreko eta busean dauden besteen ekintzak zelatatu

Bertako prozesadorea:

-PR@, hitz bat irakurri, ez badago blokea eskatu: BR @

-PW@,d; hitz bat idatzi, baliogabetu (edo eguneratu) kopiak INV @ (BC @,d). Hutsegitea badago cachean BR @

Beste prozesadorea:

-BR@, bloke bat eskatu dute, cachean badut egoera egokitu

-INV@, bloke bat baliogabetzeko, cachean badut kendu

-BC@,d; hitz at eguneratzeko agindua, cachean badut eguneratu

Protokoloak:

-MSI:

-MESI: E -> M trafikorik gabe, bi S eta bat desagertu bestea S

-MOSI: bi egoera kopia ez koherentekoak M eta O, O arduratu MN eguneratzeaz, M -> O posible Stik pasa gabe

-MOES(I): Ez du inoiz Mna eguneratzen

Busa atomikoa denez arbitro bat dago bi agindurekin BRQ (eskaera) eta BGR (onarpena)

Sinkronizatzeko moduak:

-Sekzio kritikoak: elkarrekiko esklusioan exekutatu behar diren kode zatiak

-Gertaeren bidezkoak: puntutik puntura (gertaerak) eta globalak (hesiak)

Prozesuak gertaera baten zain geratu (denbora galdu), itxarote aktiboa edo blokeoa

Sekzio kritikoa prozesu bakarrak exekuta dezake aldiko, honetarako lock funtzioa erabiliz

SMP sistemetan oso garrantzitsua da ahal den trafiko gutxien sortzea, T&S aginduak beti idazten du sarrailan, sarrailako datu-blokea baliogabetu aldiro -> trafiko handia

Test-and-Test&Set kasu honetan lehenik sarraila irakurtzen da eta irekita badago egiten da T&S

Load Locked-Store Conditional hardwareko latch bat helbide eta adierazle banaz, trafikoa sekziora sartzean soilik

Fetch&OP eragiketa sinpleak modu atomikoan egiteko

Trafikoa murrizteko Txartelak sekzio kritikoko sarrerak ordenatzeko, txartela lortu eta txandaren zain geratu.

Puntutik punturako gertaerak ekoizle batek egin ta beste norbaitek konsumitu (tipikoki Flag-ak)

Hesiekin prozesu multzo bat sinkronizatzen da exekuzioarekin jarraitu aurretik

Hainbat prozesadoretan kontsistentziaren arazoa sortzen da ordena globala ez baita ezagutzen

Koherentzi-protokoloek ez dute aldagai desberdinen aldaketen ordena zehazten

Sinkronizazio-eragiketekin ordena zehaztu daiteke baina ez emaitza (P1 A=1, P2 A=?)

Kontsistentzia ziurtatzeko exekuzio-ordena eta eragiketen atomikotasuna zehaztu behar dira

Kontsistentzia sekuentziala:

- -Aginduen ordena lokala errespetatu behar da
- -Memoria-atzipenen atomikotasuna bermatzen da
- -Baldintza oso gogorrak
- -Ezin dira idazketa bufferrak erabili
- -Ezin dira erregistroak erabili memoria-erabilera optimizatzeko

Kontsistentzia eredu malguak

- -Ordena erlazio batzuk ez dira beti beteko
- -Aukera dago ordena hertsia markatzeko (fence aginduekin)
- -Total Store Ordering-ek load aurreratua
- -Partial Store Ordering-en ez dira wr>>rd eta wr>>wr ordena-erlazioak ziurtatzen
- -Weak Ordering-en sinkronizazio-eragiketetan izan ezik, memoria-aginduen arteko edozein ordena onartzen da

Paralelismo motak:

- -Ale xehea, ataza txikiak, komunikazio asko
- -Ale larria, ataza handiak, komunikazio gutxi

Kode paralelizatuak begiztako aginduen ordena alda dezake soilik iterazioak independenteak badira

i1 eta i2 iterazioko aginduen arteko dependentzia bat badago i2-i1 distanziakoa dela esaten da

Dependentzia-grafoak eta iterazio-espazioa

Dependentziak detektatzeko testak:

- -Begiztaren indiziearen funtzio lineala X[a*i+b] eta X[c*i+d]
- -ZKH testa (d-b/ZKH(c,a))
- -ZKH testat egin ahal izateko begizta normalizatu iteraziosren pausua 1 izateko

Datu-dependentziak errespetatu, arazoa distantzia>0 eta grafoan zikloak

Dependentzi guztiak 0 distantziakoak -> doall

Dependentzi arteko distantziak badaude baina denak aurrera -> forall (doall+barrier)

Dependentzi-zikloak badaude -> doacross

Dependentzien sinkronizazioa:

- -Gertaera-bektorak: post(gA, i) wait(gA, i)
- -Kontagailuak: wait(kA, i-1) post(kA, i)

Optimizazioa:

- -Indukzio-aldagaiak desegin
- -Begizten fisioa
- -Dependentziak ordenatu (aurrerantz)
- -Lerrokatu dependentziak (peeling)
- -Sortu hari independenteak
- -Minimizatu sinkronizazioa
- -Begizta-trukea
- -Noranzko-aldeketa
- -Skew