Parallella program ger paradigmskifte

Multikärnor gör gamla programverktyg föråldrade

EXPERTARTIKEL

Erik Hagersten (eh@it.uu.se) är professor i datorarkitektur vid Uppsala universitet. Han forskar om nya algoritmer, prestandaverktyg och avlusare för chipmultitrådade CPU:er. Han var tidigare chefsarkitekt för stora servrar på Sun Microsystems. **Lars Albertsson** (lalle@sics.se) är civilingenjör i elektroteknik och forskar på SICS i test- och felsökningsmetoder för parallell programvara. Han har arbetat med verifikation av servermaskiner på Sun Microsystems i USA. **Jakob Engblom** (jakob@virtutech.com) är doktor i realtidssystem och affärsutvecklare på programvaruföretaget Virtutech vars verktyg Simics möjliggör deterministisk avlusning av parallella processorsystem.

Av Erik Hagersten, Uppsala universitet, Lars Albertsson, SICS, Jakob Engblom, Virtutech

De senaste 20 åren har inbyggnadsindustrin kunnat dra nytta av att vanliga processorer blivit allt snabbare tack vare ökande klockfrekvenser och mer avancerad datorarkitektur.

De flesta system har klarat sig med en enda processor. Om prestanda var för låga kunde man vänta på en snabbare.

Nu har trenden brutits. Klockfrekvenserna ökar inte längre. Processortillverkarna har istället börjat placera flera processorkärnor på ett chip för att höja prestanda.

Denna övergång till parallelldatorer riskerar att skapa stora problem för industrin, eftersom gammal kod i de flesta fall inte alls är anpassad för system med flera processorer. För maximal prestanda måste man skriva kod som använder flera processorer parallellt. Det är svårt.

Att de tekniska förutsättningarna för datorer ändras är ingenting nytt. En sak har dock länge varit densamma: den sekventiella programmeringsmodellen. Fram till nu har nästan alla processorer haft som mål att köra ett enda sekventiellt program fort. Morgondagens datorer kommer att kräva parallella program och många parallella arbetsuppgifter för att kunna leverera fulla prestanda.

De nya tillverkningsprocesserna har tvingat leverantörerna till denna utveckling. Hade de kunnat fortsätta utveckla sekventiella datorer enligt utvecklingstakten i Moores lag så hade de gjort det.

Det finns fyra anledningar till att paradigmskiftet sker just nu.

#1: Utbredningshastigheten minskar. Varje gång vi byter teknologi och transistorerna krymper så krymper också ledarna som sammanfogar transistorerna. En krympt ledare medför en ökad resistans som gör att signalutbredningshastigheten minskar. Med andra ord blir transistorerna snabbare och ledarna långsammare för varje generation. Vi är nu i ett skede där ledarnas fördröjning börjar dominera.

Så hjälper parallellism: på ett chips med flera processorkärnor, var och en med en egen förstanivåcache, behöver de flesta minnesåtkomster inte färdas över långa avstånd på chipet.

#2: Det är svårt att hitta oberoende instruktioner. Dagens processorer utför inte en instruktion åt gången, utan jobbar på flera instruktioner samtidigt för att öka prestandan. Detta kräver dock att instruktionerna är oberoende av varandra, det vill säga att resultatet från en instruktion inte används av en annan instruktion. Vi har redan utnyttjat alla uppenbara knep för att hitta oberoende instruktioner. Att hitta ännu fler skulle kräva enorma resurser.

Så hjälper parallellism: när en processor förses med instruktioner från flera parallella trådar så är det lättare att hitta oberoende instruktioner.

#3: Effektbehovet ökar. Effektutvecklingen i en processor är proportionell mot chipsytan, frekvensen samt spänningen i kvadrat. När frekvensen har ökat så har också spänningen ökat för att kunna uppnå dessa frekvenser. Man har alltså inte bara fått dras med det linjära förhållandet mellan frekvens och effekt, utan även fått en ökad effekt på grund av den ökande spänningen.

Effektutvecklingen har idag galopperat in i väggen. Det är inte bara elräkningen som avskräcker, vi börjar närma oss effekter där luftkylning inte längre fungerar.

Så hjälper parallellism: om vi arbetar på flera parallella flöden samtidigt kan vi dra ner frekvensen och därmed också spänningen samtidigt som den totala prestandan ökar.

#4: Minnet är en flaskhals. Idag är det vanligt att en processor tillbringar mer

än hälften av tiden med att vänta på att data från minnet skall anlända. Minnets prestanda har inte utvecklats lika snabbt som processorns.

Så hjälper parallellism: med många samtidigt utestående minnesoperationer som överlappar varandra, ökar minnessystemets prestanda. Detta får vi automatiskt om flera trådar bearbetas parallellt.

Slutligen kan man notera att det är relativt enkelt för tillverkarna att ta fram nya processorer med höga teoretiska prestanda genom att kombinera flera likadana processorer på ett chip. Speciellt jämfört med att ta fram snabbare sekventiella processorer.

Det senaste året har alla stora processortillverkare och många mindre lanserat chips med två eller fler processorkärnor, speciellt för användning inom inbyggnadsvärlden.

Suns kommande ”Niagara” hanterar 32 trådar på ett chip. Freescales 8641D har två PowerPC e600-kärnor på ett chip. ARM:s ARM11 MPCore har fyra kärnor.

Bland nätverksprocessorer från bland annat PMC-Sierra, Broadcom och Cavium är multiprocessorer redan vanliga. Och på x86-sidan har både Intel och AMD börjat leverera processorer med dubbla kärnor.

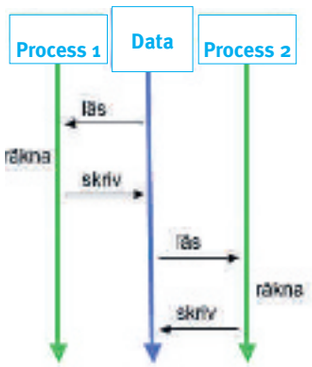
Förändringen kan på ytan kanske inte verka som någon stor sak. Men i själva verket är detta nog vårt årtiondes största paradigmskifte för hela datorindustrin. Skiftet gör inte bara dagens sekventiella program omoderna, utan påverkar hela infrastrukturen: algoritmerna, språken, kompilatorerna, teststrategier och så vidare.

En del forskning inom detta område utfördes på 80-talet, men med inriktning mot massivt parallella superdatorer. Den kommersiella mjukvaruteknologin fick inte mycket uppmärksamhet. Ny forskning krävs för att man ska kunna bygga en ny infrastruktur för utveckling och testning av parallella program för vanliga tillämpningar inom inbyggda system och persondatorer.

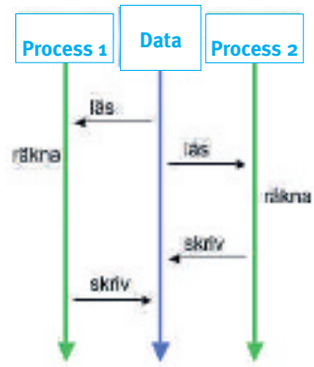
Men, säger vän av ordning, parallella maskiner har ju varit legio inom inbyggnadsområdet i många år. I telekomsystem används hundratals DSP-processorer för att hantera telefonsamtal. En modern bil har uppåt 100 mikroprocessorer i sig. Vanliga mobiltelefoner innehåller minst en styrprocessor och en DSP.

I dessa system har man dock valt att låta varje processor ha sitt eget minne och kommunicera med andra processorer via meddelanden. Fördelen är att man undviker de problem som delat minne ger upphov till. Nackdelen är att man inte kan utnyttja multiprocessorers fulla kapacitet.

Strukturen med meddelanden som skickas mellan isolerade noder kommer nog att bestå som systemarkitektur, men varje nod kommer i sig att vara en multiprocessor. I längden kommer ingen undan multiprocessorerna.



Parallellprogrammering är svårt. I övre diagrammet arbetar process 2 med uppdaterade data från process 1 och allt fungerar som det är tänkt.



Men när process 2 startar tidigare än förutsatt blir resultatet fel. Processerna utgår från samma data och process 1 skriver över de ändringar som process 2 gjort.

Man kan kanske tänka sig att kompilatorer eller virtuella maskiner skulle kunna skriva om programmen för att utnyttja hårdvaran optimalt. De teknikområdena är dock omogna och inom överskådlig framtid kommer det att vara programmerarnas uppgift att avgöra vilka delar av ett program som kan utföras parallellt.

De flesta människor gör tyvärr ofta misstag när de försöker hantera komplexa samtidiga förlopp och lista ut i vilken ordning händelser kan tänkas ske. Parallella program tenderar därför att innehålla fler fel än liknande sekventiella program, och felen är svårare att återskapa och åtgärda. När ett program sprids över flera processorkärnor vet man inte i vilken ordning de olika delarna utför sitt arbete. Även om ett program beter sig som förväntat under en testkörning kan det misslyckas med samma uppgifter i en annan testkörning eller när det har tagits i bruk, om slumpen gör så att processerna kommunicerar i en annan ordning eller vid andra tidpunkter än tidigare.

Detta kommer öka mängden intermittenta fel i programvaran. Och om man någon gång lyckas provocera ett intermittent fel är det svårt att felsöka, då felet tenderar att försvinna igen om man försöker observera programkörningen. De metoder och verktyg som vi använder idag för programmering och kvalitetskontroll hanterar helt enkelt inte parallellism tillräckligt bra.

Det kan tyckas pessimistiskt att tro att programvara plötsligt kommer att bli mycket sämre, men det förestående paradigmskiftet ska inte underskattas. Det är en av de största förändringarna i mjukvaruhistorien, och har större påverkan än till exempel när tidsdelning och virtuellt minne introducerades. Till skillnad från tidigare skiften blir tyvärr programmering svårare av denna förändring och vi måste vänja oss av med de prestandaökningar vi är vana vid eller betala med försämrad kvalitet och högre utvecklingskostnader.

Problemet gäller inte bara maximala prestanda på nya system, utan drabbar också befintlig programvara. Många standardtekniker som används för att programmera inbyggda system med flera program som delar på en processor fungerar inte när man går över till att köra på flera processorer samtidigt.

Exempelvis kan man inte anta att man har exklusiv tillgång till data som flera processer delar på bara för att man stängt av interrupten på den egna processorn. Eller att processer som har samma prioritet aldrig körs samtidigt – de kan mycket väl köras samtidigt, på olika processorer.

Vi behöver ägna mycket forskning åt problemet med parallell mjukvara och utveckla både beprövade och nya metoder. Våra utbildningar i datorteknik måste förändras så att nyutexaminerade ingenjörer förstår problemet och har kunskap om parallella programsystem.

Det behövs verktyg som kan hjälpa oss hantera kvalitet i parallell mjukvara. Det behövs verktyg som kan hjälpa till att provocera fram felen. Och det behövs verktyg för att skapa parallella program och för att optimera prestanda. ■

ORDLISTA

Multiprocessor, en dator med mer än en processor, traditionellt paketerad med ett chip per processor.
Multitrådad processor, en processor som använder speciell teknik för att se ut som flera processorer mot mjukvaran.
Multikärna, ett processorchip som innehåller mer än en komplett processor. Man kallar varje enskild processor för en ”kärna” för att skilja detta från multitrådning. En multikärna är alltså en multiprocessor på ett chip.
Chipmultitrådning, att köra flera programtrådar på ett chip. Man använder antingen multitrådning eller multikärnor. Eller båda om varje kärna är multitrådad.