8.3 Een Multiprocessor Testbed Systeem

P.H. Hartel, L.O. Hertzberger, W.G.P. Mooij en F. Tuynman,
Vakgroep Informatica, Universiteit van Amsterdam

S A M E N V A T T I N G

Door de vakgroep Informatica van de Universiteit van Amsterdam wordt gewerkt aan een multiple processor testbed systeem, dat zal dienen om het gedrag van parallelle inferentie en reductie machines te bestuderen.
De gebruikersprogrammatuur kan zowel in Prolog, respectievelijk een functionele programmeertaal worden geschreven, als in een imperatieve programmeertaal. Toepassingen worden met name voorzien op het gebied van beeldverwerking en voor het verwerken van sensor gegevens.

Inleiding

Voor steeds meer toepassingen van de informatica is "computing power" vereist. Dit wil zeggen dat bepaalde resultaten slechts kunnen worden berekend met behulp van de krachtigste (en duurste) computersystemen. De vraag naar meer rekencapaciteit en rekensnelheid neemt voortdurend toe.
Aan deze vraag kan in principe op twee manieren worden voldaan.

- Door middel van supercomputers.
  De huidige generatie supercomputers (scalair, vector, pipeline) zal worden opgevolgd door een nieuwe, nog snellere en krachtiger generatie. Het is bovendien waarschijnlijk dat dit soort machines op den duur goedkoper zullen worden.

- Door zogenaamde Multiple Instruction Multiple Datastream (MIMD) machines.
  De huidige generatie microprocessors is goedkoop en krachtig. Een volgende generatie zal minstens even goedkoop en veel krachtiger zijn. Indien een aantal van deze microprocessoren tot een zogenaamd MIMD systeem wordt samengesteld, dan kan tegen relatief geringe kosten een systeem worden verkregen waarvan de prestaties vergelijkbaar zijn met die van de huidige supercomputers.

Bij een keuze tussen één van de bovengenoemde oplossingen mogen niet alleen overwegingen van pragmatische aard een rol spelen. Aan de toename van de rekencapaciteit van de huidige elkaar opvolgende generaties van supercomputers wordt een bovengrens gesteld die enerzijds wordt bepaald door de snelheid van het licht, anderzijds door het feit dat een Von Neumann architectuur wordt toegepast.
Een MIMD systeem wordt opgebouwd met dezelfde technologie als waarmee supercomputers worden opgebouwd. Als volledig systeem kennen zij nog steeds de beperkingen die opgelegd worden door de lichtsnelheid, maar hoeven niet noodzakelijkerwijs de beperkingen te kennen zoals die door de Von Neumann architectuur wordt opgelegd. Dit komt doordat het kenmerkend is voor een MIMD systeem dat meerdere rekenelementen simultaan bijdragen tot het verkrijgen van een bepaald resultaat. Zolang de te verwerken algoritmen voor elk der rekenelementen afzonderlijk te specificeren zijn treden er weinig problemen op, hoewel meestal wel enige coördinatie van de activiteiten gewenst is. De verwerking van niet eenvoudig in onafhankelijke deeltaken uit te splitsen algoritmen, brengt communicatie- en coördinatieproblemen met zich mee. Deze problemen zijn met de ons ten dienste staande programmeertalen en methodieken nog nauwelijks op te lossen. De moeilijkheid schuilt voornamelijk in het feit dat geen goede en correcte notaties voor parallelle problemen te vinden zijn.
Teneinde deze problematiek te kunnen bestuderen wordt een MIMD testbed systeem ontwikkeld.

Model van het Systeem

Het model dat we zullen gebruiken bestaat uit een aantal lagen, te weten de hardware laag, de testbed laag met daar bovenop de virtuele machines voor specifieke talen en een gebruikers interface. Op elk niveau zijn faciliteiten voorzien voor het doen van metingen van de prestaties van de software. Gebruik makend van de virtuele machines kunnen experimenten worden uitgevoerd op het testbed systeem, waarvan de resultaten mede tot gevolg kunnen hebben dat zowel de faciliteiten van het systeem als de definitie van de virtuele machines verder worden ontwikkeld. Indien uit een experiment een

optimale configuratie zowel wat betreft hardware als software gevonden is, wordt het mogelijk voor die specifieke virtuele machine "dedicated" apparatuur en programmatuur te ontwikkelen.
Het testbed is op een zodanige wijze opgebouwd, dat fundamentele realisatie- en applicatie-aspecten van parallelle architecturen optimaal onderzocht kunnen worden.
Het in ontwikkeling zijnde systeem bestaat uit een hardware en een software component, elk opgebouwd uit aparte modulen. De componenten worden zodanig op elkaar afgestemd, dat van zoveel mogelijk modulen de implementatie naar keuze in hardware of in software kan geschieden.
Bij het ontwerp zullen twee overwegingen een belangrijke rol spelen.

- Flexibiliteit
  De structuur van het systeem kan te allen tijden worden aangepast aan de behoefte, zowel van de kant van de gebruiker (dwz. behoefte aan rekencapaciteit) als van de kant van de systeemprogrammatuur (een zo flexibel mogelijke architectuur, bijvoorbeeld memory management, protection schemes en dergelijke).

- Betrouwbaarheid
  Het falen van systeemcomponenten zal geen desastreuze invloed behoren te hebben op de totale prestatie van het systeem.

Het testbed zal dienen als basis voor de implementatie van een aantal virtuele machines met hun taalgerichte gebruikersinterface.
De eerste virtuele machines, die op deze wijze zullen worden onderzocht, zijn een inferentie machine, gebaseerd op Prolog, en een reductiemachine. Daarnaast zal ook aandacht worden besteed aan virtuele machines voor imperatieve talen, zoals ADA en Modula-2.

Het testbedsysteem

Het basissysteem wordt ontwikkeld volgens het objectmodel. Daartoe worden een aantal modulen op zowel hardware als software gebied ontwikkeld, die op verschillende manieren tot één systeem geconfigureerd kunnen worden.
De hardware wordt in eerste instantie opgebouwd uit monoboard computers, gebaseerd op commerciëel verkrijgbare chips.

De basisprogrammatuur van het testbed systeem wordt zo algemeen mogelijk opgezet en zal slechts de minimaal noodzakelijke verzameling faciliteiten aan de bovenliggende software ter beschikking stellen. Wel worden instrumentatiemogelijkheden in zowel de hardware als de software van het systeem ingebouwd.
Gekozen is voor een gesloten systeem, zodat de problemen die een open systeem met zich mee brengt, zoals dynamische protectie en toegangscontrole, in de eerste fase van het project kunnen worden vermeden.
Voor sommige toepassingen is het irrelevant of de onderliggende architectuur een multi- dan wel een uni-processor systeem is. De gebruikte programmeertaal en zijn implementatie dienen in dat geval geen onnodige complicaties voor de applicatieprogrammeur met zich mede te brengen. Voor die applicaties die wel van het aanwezige parallellisme gebruik kunnen maken, moet het voor de programmeur mogelijk zijn om expliciet aan te kunnen geven waar tot parallelle verwerking van algorithmen kan worden overgegaan.

De reductie machine

De talen die algemeen in gebruik zijn op multiprocessor systemen eisen van de programmeur dat hij zich expliciet uitspreekt over het parallellisme dat zijn programma bevat. Nog afgezien van de vraag of dit wel altijd mogelijk is leidt dit tot programma's die architectuur afhankelijk zijn, en waarin kwesties van architectuur optimalisatie een overheersende rol spelen. Het lijkt aantrekkelijk om programmeertalen te hanteren die in principe ongespecificeerd laten wat de volgorde is van de door het programma te ondernemen acties, zodat het aan de compiler of aan het runtime mechanisme kan worden overgelaten om te bepalen welke gedeelten van het programma in aanmerking komen voor parallelle verwerking. In feite betekent dit dat programma's die van zich zelf slechts een partiële ordening hebben niet langer in een totale ordening ingebed hoeven te worden.
In de literatuur zijn verschillende klassen van talen voorgesteld, zoals functionele talen en de talen gebaseerd op eerste orde predicaten logica, die deze eigenschappen bezitten. De bij deze talen horende architecturen verschillen aanzienlijk van de klassieke Von Neumann architecturen. Het ligt in de bedoeling om één of meer van deze talen te implementeren op het testbed. De individuele processors in het multiprocessor systeem zijn weliswaar van het Von Neumann type, maar het systeem als geheel leent zich goed voor de simulatie van niet-Von Neumann architecturen. Deze methode van simulatie kan gebruikt worden om meer inzicht in de mogelijkheden en de toepassing van niet-imperatieve talen te verkrijgen.

De inferentie machine

Prolog is een taal die veel gebruikt wordt voor onderzoek op het gebied van de Kunstmatige Intelligentie. Deze taal lijkt zich, in tegenstelling tot de hiervoor ook veel gebruikte taal Lisp, goed te lenen voor executie op een MIMD systeem. Prolog biedt op twee niveaus mogelijkheden voor parallelle executie, het micro en het macro niveau. Bij het micro-parallellisme worden de statements van Prolog, die bestaan uit Horn-clauses, gelijktijdig uitgevoerd: de conjuncties en eventueel disjuncties in de Horn-clauses worden parallel geëvalueerd. Bij het macro-parallellisme worden de verschillende taken die de Prolog machine uitvoert verdeeld over meerdere processoren. Te denken valt hierbij aan taken als het verzorgen van de user-interface, het ondersteunen van debugging faciliteiten, het executeren van het unificatie-algoritme etc.
Veel aandacht wordt besteed aan het inbedden van een goede programmeeromgeving in het parallelle systeem. Faciliteiten ten aanzien van gestructureerde editors, debugging-systemen en programma-generatoren spelen hierbij een voorname rol.
Het onderzoek beoogt een machine te ontwerpen die het in Prolog aanwezige parallellisme uitbuit. Er zijn een groot aantal parallelle evaluatie mechanismen voor Prolog. Simulatie onderzoek van deze evaluatie mechanismen, geïmplementeerd op het testbed systeem, biedt goede mogelijkheden om de effectiviteit van de verschillende alternatieven te bestuderen.
Het is van belang dat deze simulaties op een multiprocessor systeem uitgevoerd worden, omdat dan belangrijke gegevens over de verschillende rekeneenheden tijdens het parallel evalueren van Prolog programma's toegankelijk zijn voor directe meting. Met name zijn voor dit onderzoek gegevens over de status van processen en communicatiekanalen van groot belang. Op een uni-processor zijn deze gegevens slechts met grote moeite en complexe simulatie programma's te verkrijgen. Een parallelle implementatie, zij het op een systeem dat niet speciaal voor de parallelle executie van Prolog ontworpen is, kan belangrijke informatie verschaffen over de mechanismen die de grondslag vormen van een parallel Prolog systeem.

De **send** primitieve dient om een boodschap naar een (bekende) ontvanger te sturen. De ontvanger gebruikt de **receive** primitieve om aan te geven dat deze bereid is een boodschap (uit een bepaalde categorie) te ontvangen. Zowel de **send** als de **receive** primitieve zijn blokkerend, pas als de rendez-vous optreedt wordt de blokkade van het ontvangende proces opgeheven, zodat dit laatste de ontvangen boodschap kan aannemen en een antwoord versturen, onder gebruikmaking van de **reply** primitieve. Het sturende proces wordt doorgestart zodra het antwoord op de send is ontvangen.
De behandeling van "interrupts" is ingepast in het mechanisme zoals boven beschreven. Een willekeurige interrupt kan op aanvraag door de kernel worden vertaald in een message, zodat een proces kan wachten op het optreden van een interrupt via de **receive** primitieve.
Het FADOS message passing systeem is geïmplementeerd op de boven beschreven hardware. Om een zo efficiënt mogelijke implementatie te verkrijgen is gekozen voor de blokkerende send, zodat het mogelijk is om een bovengrens aan te geven voor de hoeveelheid benodigde bufferruimte waarin boodschappen moeten worden opgeslagen. Dit voorkomt congestie in de communicatie kanalen en de daarmee geassociëerde problemen zoals overlopende wachtrijen. Door de keuze voor een blokkerende send is het niet mogelijk om op efficiënte wijze bijvoorbeeld een broadcast operatie uit te voeren. Met name voor de implementatie van de parallelle Prolog machine is voor het bijwerken van de gedistribueerde database na assert en retract clauses een broadcast faciliteit noodzakelijk.
Een aantal aspecten van het ontwerp van de FADOS kernel zullen in de toekomst worden heroverwogen.

Implementatietaal

Als de implementatietaal van het testbedsysteem is gekozen voor de programmeertaal Modula-2. Deze taal is eenvoudig genoeg om met niet al te veel moeite een efficiënte compiler te kunnen realiseren (HAR85a, AND84, COA84). De taal is speciaal ontworpen voor het ontwikkelen van betrouwbare systeemprogrammatuur en biedt met name goede mogelijkheden voor afzonderlijke compilatie van modules, waarbij door de compiler en linker de syntactische en semantische consistentie van de programmatuur, ook over de module grenzen heen, zoveel mogelijk wordt gegarandeerd (HAR84). Het ontwerp van de taal Modula-2 is geïnspireerd door Pascal, waaraan de meeste taalelementen zijn ontleend. Een groot aantal tekortkomingen van Pascal komen bij Modula-2 niet voor (SUM82), hoewel een aantal nieuwe problemen zijn gesignaleerd (SPE82).
Een aantal problemen kan op elegante wijze worden opgelost indien gebruik wordt gemaakt van (logisch) parallellisme (BEN82). Voor de implementatie van bedrijfssystemen zijn een aantal faciliteiten beschikbaar, zoals het coroutine concept en type casting. Met behulp van de coroutines kunnen in Modula-2 concurrente processen worden gecreëerd, bijvoorbeeld voor de afhandeling van communicatieprotocollen met de buitenwereld.

Het presentatiesysteem

In een multiprocessor testbed systeem is een geïntegreerd presentatie systeem een onontbeerlijke component (SEG83). Via een dergelijk systeem kan te allen tijden door de experimentator gegevens worden verkregen omtrent de activiteiten van het systeem. Zo zullen een groot aantal systeem parameters moeten worden geïdentificeerd en voorzieningen worden getroffen om deze parameters in te stellen en het effect daarvan te bestuderen. Op alle niveaus zullen aanknopingspunten worden voorzien in het testbed systeem.
Een interessante mogelijkheid biedt het gebruik van hardware spionnen. Deze

component is bedoeld om op de processorbus te worden aangesloten, teneinde informatie te verzamelen omtrent het busverkeer. Een spion kan bijvoorbeeld worden ingezet om het gebruik van bepaalde adressen waar te nemen. Via een buffergeheugen kunnen dergelijke referenties door het monitor systeem worden uitgelezen en verwerkt. Deze methode van meten heeft geen invloed op de executiesnelheid van de op dat moment uitgevoerde applicatie, daar het hier een volledig parasitaire meting betreft. Indien het buffergeheugen overloopt kan eventueel data worden genegeerd of het door te meten systeem worden afgeremd.
Het parallel schakelen van dual-port geheugens kan gebruikt worden voor het meten aan, onder andere, communicatie software. Omdat de protocollen bekend zijn, kan door het toevoegen van wat extra identificatie informatie aan alle op het testbed system verzonden boodschappen de communicatie door het monitorsysteem worden gevolgd.
Op hoog niveau zal ook op de processors welke voor de applicaties nuttig werk verrichten extra programmatuur nodig zijn, teneinde gegevens te verzamelen over bijvoorbeeld het aantal processen wat op een bepaald moment actief is, de tijd welke processen tussen het uitwisselen van boodschappen nuttig werk kunnen verrichten enzovoort. Verwacht mag worden, dat indien alleen voor activiteiten op voldoende hoog niveau extra programmatuur wordt opgenomen, de overhead die aldus wordt geïntroduceerd niet groot is.

Voor de implementatie van het presentatiesysteem zullen zoveel mogelijk dezelfde hard- en software componenten worden gebruikt als voor de implementatie van het testbed zelf. Dit wordt geïllustreerd door het bovengenoemde gebruik van dual-port geheugens. Op deze wijze kan een aanzienlijke besparing op de ontwikkelingskosten van het systeem worden bereikt. Speciale componenten, zoals de hardware voor de spionnen en window handling software, zullen eventueel moeten worden ontwikkeld voor zover dergelijke componenten niet verkrijgbaar zijn. Met name lijkt de software van het Medos besturingssysteem (WIR81, GEI82) geschikt voor de implementatie van de basisprogrammatuur op het monitor systeem.
Naar de experimentator toe dient het presentatie systeem zoveel mogelijk nuttige informatie op overzichtelijke wijze te presenteren. Veel ervaring is opgedaan met bijvoorbeeld het weergeven van boomstructuren die de status van een programma weergeven (HAR85b). Daarbij is gebleken dat het gebruik van kleur een belangrijk hulpmiddel kan zijn bij het overzichtelijk weergeven van gecompliceerde structuren, zoals bomen en grafen. Aangezien vergelijkbare structuren een prominente rol spelen bij de implementatie van zowel de inferentie als de reductie machine ligt het voor de hand op deze lijn voort te gaan. Ook elders zijn met name bij de bestudering van het gedrag van experimentele al dan niet gesimuleerde reductiemachines uitstekende ervaringen opgedaan met geïntegreerde intelligente presentatiesystemen (KEN83).

Conclusie

De ontwikkelingen op hardware gebied maken het mogelijk een veelheid aan nieuwe computerarchitecturen te construeren, die in theorie zeer grote rekenefficiëntie kunnen bereiken. Een centraal probleem is echter het feit dat slechts weinig toepassingen effectief gebruik kunnen maken van de mogelijkheden van parallelle systemen. Dit komt voornamelijk doordat deze rekenprocessen moeilijk op te delen zijn in een aantal subprocessen die weinig onderlinge interactie hebben. Dergelijke problemen kunnen alleen worden opgelost door middel van het uitvoeren van experimenten op parallele architecturen waarbij zoveel mogelijk gegevens over de applicatie en de implementatie aspecten ervan kunnen worden verzameld. Een testbed systeem als hier beschreven is een machtig hulpmiddel voor het doen van dergelijk architectuur onderzoek.

Referenties

AND84 Anderson, T.L.
- Seven Modula compilers reviewed
- Journal of Pascal, Ada & Modula-2, March-April 1984.

BEN82 Ben-Ari, M.
- Principles of Concurrent Programming
- Prentice Hall, Englewood Cliffs, New Jersey, 1982.

COA84 Coar, D.
- Pascal, ADA and Modula-2
- Byte Magazine, Vol. 9, 8, Aug. 1984.

GEI82 Geissmann, L., J. Hoppe, Ch. Jacobi, S.E. Knudsen, W. Winiger & N. Wirth
- Lilith Handbook : A Guide for Lilith Users and Programmers
- Institut für Informatik, ETH Zürich, 1982.

HAR84 Hartel, P.H. & F.A.H. van Harmelen
- The Analysis of Modular Structures with respect to their Interconnection Topology
- Interfaces in Computing, 2, Jan. 1984.

HAR85a Hartel, P.H. & D. Starreveld
- Modula-2 Implementation Overview
- Nog te verschijnen.

HAR85b Hartel, P.H., M.J. Plasmeijer & J. van den Bos
- WTOOL : An Implementation of Input Tools in Modula-2 and Pascal
- Nog te verschijnen.

HER82 L.O. Hertzberger
- The Fast Amsterdam Multi Processor (FAMP) system
- Comp. Phys. Comm. 26 (1982) 79.

KEN83 Kennaway, J.R. & M.R. Sleep
- Novel Architectures for declarative languages
- Software and Microsystems, Vol. 2, 3, Jun. 1983.

SEG83 Segall, Z., A. Singh, R.T. Snodgrass, A.K. Jones & D.P. Siewiorek
- An Integrated Instumentation Environment for Multiprocessors
- IEEE Transactions on Computers, C-32, 1, Jan. 1983.

SPE82 Spector D.
- Ambiguities and insecurities in Modula-2
- ACM Sigplan Notices, Vol. 17, 8, Aug. 1982.

SUM82 Sumner R.T. & R.E. Gleaves
- Modula-2 : A solution to Pascal's Problems
- ACM Sigplan Notices, Vol. 17, 9, Sep. 1982.

TUY84 Tuynman F.
- A distributed real-time operating system
- Nog te verschijnen.

WIR81 Wirth, N.
- The Personal Computer Lilith
- Report No. 40, Institut für Informatik, ETH Zürich, 1981.

WIR82 Wirth, N.
- Programming in Modula-2
- Springer Verlag, Berlin, 1982.