Samenvatting

In dit proefschrift zijn nieuwe methodes beschreven om prestaties van een weersvoorspellingmodel te verbeteren. Hiervoor is een bestaand codegeneratie pakket CTADEL uitgebreid met extra functionaliteit.

Eerder is aangetoond dat codegeneratie met CTADEL erg krachtig is voor finite difference methodes. In hoofdstuk 2 wordt TADEL uitgebreid om code te genereren voor Galerkin finite elements methodes. We hebben deze versie gebruikt om code te genereren voor de ondiepwatervergelijkingen. Om de prestaties van de gegenereerde code met de originele handgeschreven code te vergelijken, zijn beide codes op een computer gerund gebruikmakend van diverse compilers. We hebben gevonden dat de snelheidswinst erg afhankelijk is van de kwaliteit van de compiler. We hebben prestatiewinsten bereikt van een factor 1.2 tot 3 ten opzichte van de handgeschreven code, voor respectievelijk de pathscale 3.0 en gfortran 4.1.2 compiler.

In hoofdstuk 3 hebben we de parallelle implementatie van het HIRLAM weervoorspellingmodel geoptimaliseerd door communicatie te overlappen met berekeningen. In onze aanpak worden de berekeningen al gestart terwijl de communicatie nog bezig is. Het doel is om de communicatie tijd te verbergen achter de rekentijd. Het bereikte voordeel van het samen laten vallen van communicatie met de berekeningen wordt deels tenietgedaan door de toename van de rekentijd ten gevolge van het splitsen van het processordomein. De balans tussen winst en verlies hangt af van de computerhardware. Daarom hebben we verscheidene overlapstrategieën onderzocht, variërend van de aanpak die het snelst is op vectormachines tot de aanpak die een maximale overlap probeert te bereiken. We hebben gevonden dat als het interconnect netwerk erg snel is, de strategie die ontwikkeld is voor vectormachines het snelst is. Dit wordt veroorzaakt door het feit dat tijd die wordt gespendeerd aan communicatie klein is, en daardoor zijn de voordelen van overlappen verwaarloosbaar. Als anderzijds het interconnectienetwerk niet erg snel is, zijn de besparingen aanmerkelijk. Bij het gebruikte computerplatform is de toename van de rekentijd door de splitsing van het computerdomein klein. Daarom is de strategie met de maximale overlap het meest efficiënt. In het algemeen, kunnen we concluderen dat afhankelijk van de netwerkverbinding, weinig tot aanzienlijke winst behaald kan worden op de rekentijd door de communicatie te overlappen met de berekeningen. Met de gebruikte computer platformen zijn de tijdswinsten op een goedkoop interconnect netwerk zo substantieel dat er geen noodzaak is voor het gebruik van een duur

interconnectienetwerk. Als echter een snel interconnectienetwerk beschikbaar is is de te bereiken winst verwaarloosbaar. In dat geval wegen de voordelen niet op tegen de nadelen van de complicaties in de programmatuur. Die complicaties zijn in het algemeen onoverkomelijk als ze met de hand geprogrammeerd moeten worden, en daarom biedt codegeneratie perspectieven.

In hoofdstuk 4 hebben we CTADEL uitgebreid om parallelle programma's te kunnen genereren. We hebben "Single Program Multiple Data" als programmeermodel gekozen. Gebruikmakend van dit model bestaat een parallel programma uit vier stappen: domeindecompositie; datadistributie; berekeningen en communicatie; en het verzamelen van de resultaten. We hebben sjablonen gedefinieerd om code te genereren voor elke stap van een parallel programma. Een voordeel van onze techniek is dat de codegenerator automatisch de hoeveelheid data bepaalt die gecommuniceerd moet worden. De gebruiker hoeft zich hier dus geen zorgen over te maken. Door de toepassing van deze techniek hebben we succesvol parallelle code gegenereerd voor de ondiepwatervergelijkingen.

In hoofdstuk 5 hebben we een methode die gebaseerd is op GPU berekeningen onderzocht om het HIRLAM weervoorspellingmodel te versnellen. We hebben de dynamics routines versneld door deze te optimaliseren voor GPU's. Vanwege data-afhankelijkheden in de berekeningen van de dynamics routine zou synchronisatie nodig zijn tussen blokken, maar CUDA, het programmeerplatform van de GPU's, ondersteunt geen synchronisatie tussen blokken. Daarom hebben we twee methoden onderzocht voor het bouwen van onafhankelijke blokken. De eerste werkt met behulp van herberekening: wanneer een blok informatie nodig heeft van een naburig blok berekent het deze informatie zelf. De tweede methode gebruikt een splitsing van de dynamics routine in verschillende kernels: een kernel bestaat in dat geval uit blokken die onafhankelijk van elkaar kunnen worden uitgevoerd. We hebben gevonden dat deze multi-kernelmethode significant efficiënter is dan de herberekeningmethode. We hebben verdere optimalisaties mogelijk gemaakt door de CUDA code te implementeren met een flexibel aantal grid points per thread en een flexibel aantal threads per blok. We hebben gevonden dat de toewijzing van een verticale kolom van 32 roosterpunten per blok het beste is. Voor wat betreft de blokgrootte hebben we gevonden dat hoe groter het blok is hoe efficiënter het programma. Om de invloed van de transfertijd op de totale executietijd van een CUDA programma te reduceren hebben we meervoudige CUDA streams gebruikt om de dataoverdracht te laten samenvallen met de kernelberekeningen. De resultaten tonen aan dat we door dit overlappen de totale executietijd hebben kunnen reduceren met 36%. In vergelijking met de C code op een state-of-the-art Intel CPU geeft het CUDA programma op een enkele GPU een versnelling van een factor 55. Een enkele GPU heeft echter niet voldoende geheugen voor een productierun. Daarom hebben we een implementatie met meervoudige GPU's onderzocht. Op

4 GPU's hebben we een parallelle versnelling van een factor 2.7 bereikt.

In hoofdstuk 6 hebben we aangetoond dat onze uitbreiding van CTADEL automatisch hoog efficiënte CUDA codes genereert. Eerst hebben we CTADEL uitgebreid om de kernel en hostcode van een CUDA programma te genereren. Daarna, om overlap te bereiken tussen dataoverdracht en kernelberekeningen, hebben we CTADEL aangepast om CUDA streamprogramma's te genereren. Als voorbeeld hebben we een CUDA streamprogramma gegenereerd voor de dynamicsroutine van het HIRLAM weervoorspellingmodel. De resultaten hebben aangetoond dat de gegenereerde code efficiënter is dan het geoptimaliseerde handgeschreven programma.