Samenvatting

Het programmeren van many-cores op meerdere abstractieniveaus

De afgelopen decennia hebben we een exponentiële groei van het aantal transistors op een computerchip meegemaakt. Dit ging gepaard met een exponentiële groei van de kloksnelheid van processors. Dit zorgde ervoor dat we automatisch een hogere performance kregen zonder dat de software aangepast hoefde te worden.

Echter, rond 2005 bleek dat de kloksnelheid niet verder omhoog kon, terwijl het aantal transistors door kon blijven groeien. Dit betekende het einde van het "single-core" tijdperk waarin processors een enkele rekenkern of "compute-core" bevatte. Dit type processor was sterk geoptimaliseerd om een stroom van sequentiële instructies, instructies die na elkaar worden uitgevoerd, te verwerken.

Omdat het aantal transistors op een chip door kon blijven groeien, maar dit niet gebruikt kon worden om de kloksnelheid op te voeren, kregen we "multi-core" processoren met meerdere rekenkernen. Echter, dit betekende een fundamentele verandering van hoe men een processor programmeert: Om een processor optimaal te benutten, is het nodig een parallel programma aan te leveren, met meerdere stromen van instructies die mogelijkerwijs met elkaar moeten synchroniseren.

Deze multi-core processors bevatten nog steeds veel optimalisaties om sequentiële instructiestromen te optimaliseren wat veel ruimte op de chip kost, ruimte die ook gebruikt zou kunnen worden om rekenkernen toe te voegen. Dit proefschrift gaat over many-core processoren, waarbij zoveel mogelijk ruimte wordt benut om te rekenen. Dit betekent dat deze processors volledig toegespitst zijn op snel rekenen, maar alleen voor programma's die veel parallellisme

uitdrukken.

Parallel programmeren is moeilijk, vooral als het gaat om many-core processoren omdat deze processoren veel specifieke hardware eigenschappen bevatten om nog meer performance te verkrijgen. Dit resulteert in een ingewikkelde hardware interface naar de programmeur toe, maar zorgt er ook voor dat een programma factors sneller kan draaien, mits men goed rekening houdt met alle hardware-specifieke eigenschappen.

In dit proefschrift nemen wij het volgende standpunt in: Wij beschouwen het *single-core* tijdperk als een fortuinlijke situatie waarin we automatisch snellere programma's kregen met nieuwere generaties processors. Het *multi-core* tijdperk beschouwen wij vervolgens als een overgangsperiode naar een tijdperk waarin we te maken krijgen met allerlei limieten van de hardware, zoals een limiet op de kloksnelheid of een limiet op de snelheid van het geheugen. Om toch performance te krijgen, is het vervolgens nodig om met allerlei hardware-specifieke details rekening te houden om de processor optimaal te benutten. Wij zien *many-core processors* als een eerste manifestatie van deze trend. In dit proefschrift beschouwen wij deze limieten en problemen als een *programmeerprobleem*: Uiteindelijk zullen we allerlei hardwarelimieten tegenkomen wat zal leiden tot een ingewikkelde hardwareinterface waardoor de processoren steeds moeilijker te programmeren zijn. De hoofdvraag van dit proefschrift is hoe we effectief many-core hardware kunnen programmeren terwijl we nog steeds goede performance halen.

In dit proefschrift presenteren wij twee programmeersystemen, Many-Core Levels (MCL) en Cashmere, die een bijdrage leveren aan het programmeerprobleem dat hierboven beschreven is. Cashmere bouwt voort op een al bestaand systeem genaamd Satin, maar integreert hierbij MCL. Hoofdstuk 2 gaat nog niet over many-cores, maar presenteert een analyse op Satin programma's. Deze analyse leidde tot een aantal belanrijke conclusies die het ontwerp van MCL en Cashmere beïnvloed hebben.

Hoofdstuk 3 presenteert Many-Core Levels. MCL is een programmeersysteem voor many-core processors dat het mogelijk maakt om deze processors op meerdere abstractieniveaus te programmeren.

Programmeren, zodanig dat je rekening houdt met allerlei hardware-specifieke details noemen wij programmeren op een *laag* niveau. Aan de andere kant, programmeren op een *hoog* niveau geeft een programmeur de mogelijkheid te abstraheren van allerlei details en het programma op te schrijven zonder rekening te houden met de onderliggende hardware. Dit heeft allerlei voordelen, zoals bijvoorbeeld portabiliteit. Het programma is algemeen genoeg om vertaald te worden naar verschillende soorten hardware. Daarnaast is het programma

vaak eenvoudiger uit te drukken en daardoor beter te onderhouden. Echter, de programmeur verliest ook controle over de hardware. Nu is dit vaak niet een probleem, maar many-core processoren zijn enkel en alleen bedoeld om performance te behalen en in dat geval doen de hardware details er wel toe. Voor het gebied van many-cores vormt dit dan ook een interessante afweging. Hoofdstuk 3 presenteert oplossingen voor deze afweging.

Een andere belangrijke bijdrage van dit hoofdstuk is dat MCL de programmeur een gestructureerde methodologie biedt om many-core processors te programmeren. Wij noemen deze methodologie "stapsgewijs verfijnen voor performance".

Hoofdstuk 4 presenteert het programmeersysteem Cashmere. Cashmere integreert Satin en MCL om een systeem te verkrijgen voor heterogene clustercomputers van many-cores. Hieronder verstaan wij een computer die bestaat uit een groep processors met verschillende types many-cores die aaneengeschakeld zijn met een snel netwerk. Dit soort computers is bedoeld om grote rekenproblemen op te lossen. Een clustercomputer is al moeilijk te programmeren, een clustercomputer met many-cores is nog moeilijker te programmeren, maar een clustercomputer met verschillende typen many-cores nóg veel moeilijker. In dit hoofdstuk laten wij zien dat Cashmere zeer goed schaalt en zeer goede performance behaalt met een elegant programmeermodel dat nauwelijks aangepast is ten opzichte van Satin.

In hoofdstuk 5 concluderen wij dat MCL en Cashmere een belangrijke bijdrage leveren aan de programmeerproblemen die many-core hardware met zich meebrengt, hardware waarbij men rekening moet houden met allerlei hardwarespecifieke details om hoge performance te behalen.