Samenvatting

De fysieke beperkingen in de productie van computerhardware en het verlangen van de industrie om de wet van Moore bij te houden, hebben gezorgd voor het gebruik van multicore processoren en gedistribueerde systemen (Cloud) in ons dagelijks leven. Deze technologien van "simultane" verwerking vereisen vaak ingrijpende aanpassingen in de computersoftware om volledig te worden benut. Dit legt een zware last op softwareontwikkeling, zeker als er rekening gehouden wordt met de voortdurend toenemende vraag naar functionelere en waarschijnlijk complexere software. Een van de methodes om softwarecomplexiteit aan te pakken, is door modellering van software. Deze methode laat irrelevante implementatiedetails weg en legt daarbij de nadruk op de functionele juistheid van de software. Echter, om de eerdergenoemde verbeteringen in hardware optimaal te benutten, moet een softwaremodel bewust zijn van de systeembronnen, in ieder geval op een abstracte manier.

In dit proefschrift proberen we deze uitdaging aan te pakken door een eigen modelleertaal te ontwerpen. Deze taal genereert software die kan profiteren van recente hardware ontwikkelingen (multicore, cloud), zonder afbreuk te doen aan zijn abstractieniveau's. Onze taal is gebaseerd op de Abstract Behavioral Specification (ABS). Concurrency in deze modelleertaal focust op de coperatieve multitasking. Programma's geschreven in onze ABS-gebaseerde taal worden vertaald naar de functionele programmeertaal Haskell, vanwege de ingebouwde ondersteuning in Haskell voor zowel de coperatieve multitasking in de vorm van coroutines, als het multicore parralelisme door lichtgewicht uitvoeringsthreads. Daarnaast bewijzen we formeel zowel de juistheid van de vertaling naar Haskell als het behoud van de resource consumptie na vertaling in een deel van onze taal. Om onze oplossing te testen, vergelijken we zijn prestatie met andere bestaande ABS-gebaseerde implementaties.

Om softwaremodellen in staat te stellen om de controle over hun computingbronnen te nemen, breiden we onze taal uit met verschillende constructies die een abstractie maken over de hardware. Deze uitbreiding van de "bronbewuste" taal is onderdeel van een nieuwe methode voor human-in-the-loop simulaties van Clouddiensten. Zo'n simulatie kan gebruikt worden voor training van DevOps-ingenieurs in de cloudomgeving van IT-bedrijven.

Tot slot bieden we een implementatie aan voor gedistribueerde communicatie en

een verbinding met de Cloud infrastructuur, zodat softwaremodellen die geschreven zijn in onze taal uitgevoerd kunnen worden als gedistribueerde applicaties. Omdat modellen “bronbewust” zijn, kunnen ze programmatisch hun eigen cloud resource infrastructuur monitoren en beheren. Onze implementatie is de eerste realisatie van het eerdere concept van “Deployment Componenten” van ABS die een abstractie te representeren van Virtuele Machines in de Cloud en om elke ABS-applicatie in staat te stellen zich te distribueren onder meerdere Cloud-machines.