Stefan MARR

ABSTRACT

During the past decade, software developers widely adopted JVM and CLI as multi-language virtual machines (VMs). At the same time, the multicore revolution burdened developers with increasing complexity. Language implementers devised a wide range of concurrent and parallel programming concepts to address this complexity but struggle to build these concepts on top of common multi-language VMs. Missing support in these VMs leads to tradeoffs between implementation simplicity, correctly implemented language semantics, and performance guarantees.

Departing from the traditional distinction between concurrency and parallelism, this dissertation finds that parallel programming concepts benefit from performance-related VM support, while concurrent programming concepts benefit from VM support that guarantees correct semantics in the presence of reflection, mutable state, and interaction with other languages and libraries.

Focusing on these concurrent programming concepts, this dissertation finds that a VM needs to provide mechanisms for *managed state*, *managed execution*, *ownership*, and *controlled enforcement*. Based on these requirements, this dissertation proposes an *ownership-based metaobject protocol* (OMOP) to build novel multi-language VMs with proper concurrent programming support.

This dissertation demonstrates the OMOP's benefits by building concurrent programming concepts such as agents, software transactional memory, actors, active objects, and communicating sequential processes on top of the OMOP. The performance evaluation shows that OMOP-based implementations of concurrent programming concepts can reach performance on par with that of their conventionally implemented counterparts if the OMOP is supported by the VM.

To conclude, the OMOP proposed in this dissertation provides a unifying and minimal substrate to support concurrent programming on top of multi-language VMs. The OMOP enables language implementers to correctly implement language semantics, while simultaneously enabling VMs to provide efficient implementations.

SAMENVATTING

Over de laatste jaaren hebben softwareontikkelaars de JVM en CLI beginnen gebruiken als multi-language virtual machines (VM). Gelyktydig werd door de multicore revolutie de taak van de softwareontwikkelaar vermoeilijkt. Programmeertaalontwerpers ontwikkelden een grote variëteit aan concurrente en parallelle programmeerconcepten, maar het implementeren van deze concepten bovenop de multi-language VM's blijft een penibel probleem. Gebrekkige ondersteuning hiervoor in de VM's leidt tot afwegingen in de programmeertaalimplementaties tussen simpliciteit, correctheid en performantie.

Vertrekkende van de traditionele verdeling tussen concurrent en parallel programmeren vindt deze verhandeling dat parallelle programmeerconcepten voordeel halen uit performantie-gerelateerde VM ondersteuning, gelykaardig halen concurrente programmeerconcepten voordeel halen uit correctheidsgaranties van semantiek onder reflectie, mutable state en interactie met andere programmeertalen en libraries.

Door het toe te spitsen op deze concurrente programmeerconcepten vindt deze verhandeling dat een VM mechanismen moet aanbieden voor *managed state*, *managed execution*, *ownership* en *controlled enforcement*. Daarop gebaseerd stelt deze verhandeling een *ownership-based metaobject protocol* (OMOP) voor om vernieuwende multi-language VM's te bouwen met fatsoenlijke ondersteuning voor concurrente programmeerconcepten.

We demonstreeren de voordelen van de OMOP door er concurrente programmeerconcepten bovenop te bouwen, zoals agents, software transactional memory, actors, active object en communicating sequential processes. De performantieëvaluatie toont aan dat implementaties bovenop de OMOP van deze concurrente programmeerconcepten de performantie kan evenaren van conventionele implementaties, zolang de OMOP ondersteund is door de VM.

In conclusie, de OMOP biedt een verenigd substraat aan om concurrent programmeren te ondersteunen bovenop multi-language VM's. De OMOP laat programmeertaalontwikkelaars toe om op een correcte manier de semantiek van een taal te implementeren, maar het laat ook deze VM's toe om hiervoor een efficiënte implementatie te voorzien.