Many-Core Virtual Machines
Decoupling Abstract From Concrete Concurrency

- **Abstraction by ILs**
  - VM Intermediate Languages (ILs)
  - expressive abstraction for various target languages
  - state of the art is very diverse

- **Concurrency Support**
  - VM support is minimal
  - only one specific concurrency model is supported
  - few ILs provide notion of concurrency
  - no comprehensive abstraction

- **A VM Has To:**
  - decouple abstract concurrency models
  - abstract concurrency models are defined by languages or libraries
  - used by application developers

- **Concrete Concurrency Models**
  - concrete concurrency models are provided by the underlying system

- **Approach and Evaluation**
  - Top-Down from a Language Perspective
  - Implement relevant concepts on top of the RoarVM
  - Assess and evaluate benefits of VM Support

- **Virtual Machines**
  - powered by fast JIT compilers, and great GCs
  - foundation for multi-language VMs
  - allow to reuse existing infrastructure
  - require huge investments
  - reuse is economically necessary

- **Hardware and Operating Systems**
  - NUMA is the dominating hardware characteristic
  - locality explicit in shared-memory
  - abstract schemes may have different concrete implementations

- **Virtual Machine Support for Many-Core Architectures:**
  - decoupling abstract from concrete concurrency models

- **Virtual Machine Support for Many-Core Architectures:**
  - challenges:
  - deep cache hierarchies
  - and cache coherency
  - locality

- **However, there are various ways to express concurrency and solutions are domain-specific**

- **Integrated Hardware and Operating Systems**
  - system requirements:
  - performance benefits
  - trade-off to VM complexity
  - language engineering effort

- **Towards Comprehensive Concurrency Support**
  - on-top of the RoarVM
  - core-to-core communication

- **So Many Models?**
  - data parallelism
  - shared mutable state
  - non-uniform memory access
  - protecting and isolating shared state

- **Approach and Evaluation**
  - 1. Implement relevant concepts on top of the RoarVM
  - 2. Assess and evaluate benefits of VM Support
  - 3. Top-Down from a Language Perspective

- **What Are The Fundamental Problems?**
  - non-uniform memory access
  - protecting and isolating shared state
  - non-uniform memory access

- **The Manycore RoarVM**
  - A Smalltalk VM for multi- and manycore systems
  - runs on the 64-core TIE architecture
  - runs on standard ILs systems
  - supports Linux and OS X
  - released under the Eclipse Public License at http://github.com/Smarr/RoarVM

- **Manycore RoarVM Approach and Evaluation**
  - Heterogenous JVMs
  - shared memory
  - non-uniform memory access
  - protected and isolating shared state

- **So Many Models?**
  - data parallelism
  - shared mutable state
  - non-uniform memory access

- **What to Include in ILs?**
  - there are various ways to express concurrency and solutions are domain-specific

- **Locality and Encapsulation?**
  - can have explicit core-to-core communication
  - small caches
  - explicit inter-core communication

- **Experiments**
  - in cooperation with Danilo Zaglia and Sam Ibrahim

- **Virtual Machines**
  - abstract from experiments and extend the VM model

- **Virtual Machines**
  - from hardware and operating systems

- **Virtual Machines**
  - for multiple languages

- **Virtual Machines**
  - abstract from hardware and operating systems

Stefan Marr, Theo D’Hondt
{stefan.marr, tjdhondt}@vub.ac.be
http://soft.vub.ac.be/~smarr/research/