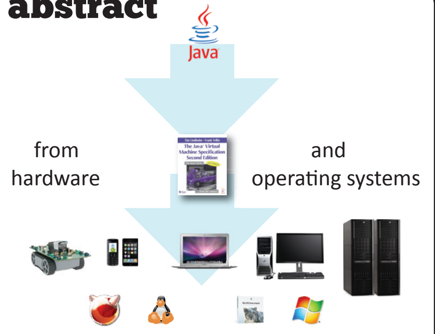
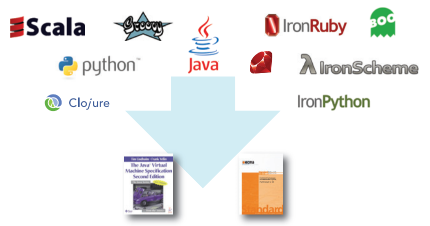
Many-Core Virtual Machines
Decoupling Abstract From Concrete Concurrency

Virtual Machines

abstract



for multiple languages



- 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

Abstraction by ILs

- VM Intermediate Languages (ILs)
  - often defined as bytecodes
  - expressive abstraction for various target languages
  - state of the art is very diverse

	Abstraction	Model	#Register	Execution Mode	Length	in Byte	#Opcodes
CLI	Bytecode	stack	0	-	variable	>= 1	217
CPython	Bytecode	stack	0	switch	variable	1 or 3	102
Dalvik VM	Bytecode	register	∞	threaded	variable	>= 2	218
Dis VM	Bytecode	memory-to-memory	0	-	variable	1 - 33	158
Erlang	Bytecode	register	1024	threaded, JIT	fixed	4	148
JVM	Bytecode	stack	0	-	variable	>= 1	201
Lua	Bytecode	register	255	switch	fixed	4	38
Mozart	Bytecode	register-memory	∞	threaded	variable	4 - 24	97
Parrot	Bytecode	register	∞	switch, threaded, JIT	variable	>= 4	>1200
PHP	Bytecode	register-memory	∞	threaded	fixed	76	136
Rubinius	Bytecode	stack	0	JIT	variable	4 -16	89
Ruby 1.8	AST	stack	0	switch	-	-	105
Ruby 1.9	Bytecode	stack	2	threaded	variable	>= 32	77
Self	Bytecode	stack	0	JIT	fixed	1	17
Squeak	Bytecode	stack	0	switch, threaded	variable	1 or 2	71
TraceMonkey	Bytecode	stack	1	threaded, JIT	variable	>= 1	234
V8	AST	-	-	JIT	-	-	38

but...

Concurrency

concurrency support is limited

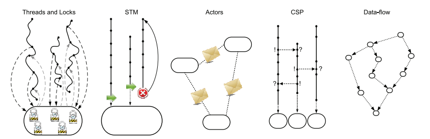
- VM support is minimal
  - only one specific concurrency model is supported
  - only few ILs provide notion of concurrency
  - no comprehensive abstraction

	.NET Java	Inferno	ERLANG	mozart
Model	Threads/Locks	CSP	Actors	Data-flow
IL Support	Marginal	High-level	High-level	Marginal
StdLib	Low/high-level	High-level	High-level	High-level

Stefan Marr, Michael Haupt, and Theo D'Hondt
Intermediate Language Design of High-level Language Virtual Machines:
Towards Comprehensive Concurrency Support
In: Proc. of the 3rd Workshop on Virtual Machines and Intermediate Languages, ACM, October (2009)

a VM has to:
decouple abstract concurrency models

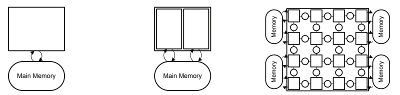
- abstract concurrency modesl are defined by languages or libraries
- used by application developers



- wide range of models supported by VM is necessary
  - implementing unsupported models on top is hard
    - restrictions hinder efficient implementation
  - support at VM-level allows reuse and optimization

and concrete concurrency models

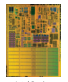
concrete concurrency models are provided by the underlying system

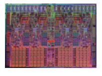


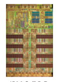
Single-Core
- preemptive OS threads
- instruction-level parallelism
- VM challenges
  - deep cache hierarchies
  - cache-consciousness required

Multi-Core
- uniform memory access
- native support for thread-level parallelism
- and cache coherency
- locality and cache hierarchy must be considered
  - avoid cache thrashing

Many-Core
- non-uniform memory access architectures
- can have explicit core-to-core communication
- very diverse designs
  - with/out cache coher.
  - explicit inter-core com.


Intel Banias


Intel Nehalem


IBM Cell B.E.

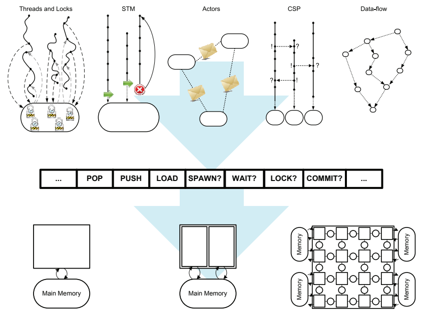
So Many Models

what to include in ILs?

there are various ways to express concurrency

Data Parallelism
Non-shared Memory
Shared Memory
Task Parallelism
Actors
CSP
Non-Determinism
Deterministic
Join Calculus
Data-flow
Reactive Programming
Transactional Memory
Message Passing
Streams
Atomic Operations
Threads and Locks
Fork/Join
π-Calculus
Map/Reduce
SIMD
Chords
Process Calculi
Petri-Nets
Tuple Spaces
Synchronization
Lock-free
Wait-free
Futures
Mutex
Semaphore
Monitor
Barriers
Inbox
Kernel
Token
Spawn
Producer
Consumer

how to combine the various models?



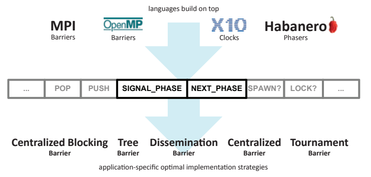
will result in a methodology

How to Build Virtual Machines
Designing Support for Multiple Concurrency Models
Approaches and Tradeoffs
(c) 2013 1st ed. VUB Press

Experiments

extending ILs

for shared memory: fuzzy/split-phase barriers



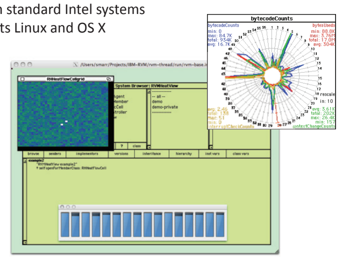
other experiments:
an IL for threads and locks, and an IL for Actors

Stefan Marr et al.
Virtual Machine Support for Many-Core Architectures: Decoupling Abstract From Concrete Concurrency Models
In: 2nd International Workshop on Programming Languages Approaches to Concurrency and Communication-cEntric Software, York, UK, March (2009)

Hans Schippers, Tom Van Cutsem, Stefan Marr, Michael Haupt, and Robert Hirschfeld
Towards an Actor-based Concurrent Machine Model
In: Proc. of the 4th Workshop on the Implementation, Compilation, Optimization of Object-Oriented Languages, Programs and Systems, ACM (2009)

our VM

- A Smalltalk VM for many-core systems
- runs on the 64-core TILE64 chip
- runs on standard Intel systems
- supports Linux and OS X



In cooperation with David Ungard and Sam Adams from
IBM Research

our hardware

for the validation of our research we can use
a wide spectrum of state of the art technology

Apple MacPro
- 2 CPUs
- 4 cores per CPU
- 2 threads per core
- complex cache and memory hierarchy

TILEPro64
- 64 cores
- explicit core-to-core communication
- small caches
- shared coherent memory

Vrije Universiteit Brussel
Software Languages Lab
Stefan Marr, Theo D'Hondt
{stefan.marr, tjdhondt}@vub.ac.be
http://soft.vub.ac.be/~smarr/research/