Intermediate Language Design of High-level Language VMs

Towards Comprehensive Concurrency Support

Stefan Marr
Software Languages Lab
Vrije Universiteit Brussel

Michael Haupt
Hasso Plattner Institute
University of Potsdam

Theo D'Hondt
Software Languages Lab
Vrije Universiteit Brussel

VMIL Workshop, 25th October 2009, Orlando, Florida

Agenda

- Motivation
- Survey Design
- Concurrency Support
- Conclusion
- Outlook

Motivation

- VMs support concurrency insufficiently!
  - Why? Because, we want multi-language VMs
    - Fast JITs, great GCs
- How to design an intermediate language?
  - To our knowledge
    - No surveys
    - No overview of design space/tradeoffs

How to design an intermediate language?

SURVEY DESIGN

Survey Design

Dalvik VM 1.0

Dis VM 4th ed.

OTP R12

5.1

1.4

CLI/ECMA-355

JVM spec.

1.3

6.0 svn

2.6

Rubinius

Ruby 1.8/1.9

4.4

Squeak 3.9

Firefox 3.5 TraceMonkey

V8 svn

Survey Criteria

- Specification vs. implementation
- Abstraction level of intermediate language
- Machine model
- Representation, instruction encoding
- Instruction categories
  - Arithmetic & logic, control flow, stack, …
- Optimizations

Survey

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

Survey

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

Survey

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

Survey

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

Survey

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

Survey

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

How to support concurrency in an intermediate language?

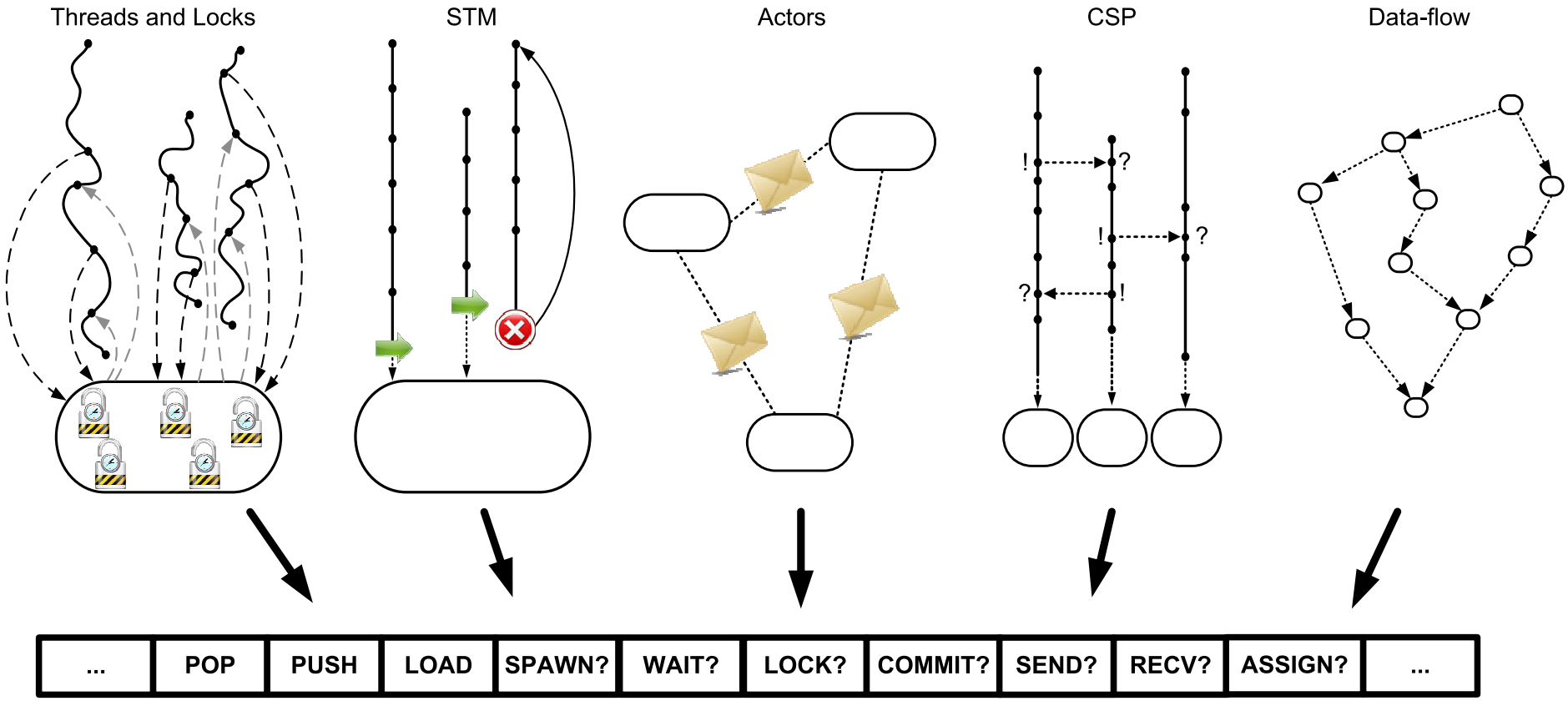
CONCURRENCY SUPPORT

Reminder: Motivation

- VMs support concurrency insufficiently!
  - Why? Because, we want multi-language VMs
    - Fast JITs
    - Great GCs

↳ Add concurrency to intermediate language

Why Concurrency in ILs?



Concurrency Support in the IL?

Dalvik VM 1.0

Dis VM 4th ed.

OTP R12

5.1

1.4

CLI/ECMA-355

JVM spec.

1.3

6.0 svn

2.6

Rubinius

Ruby 1.8/1.9

4.4

Squeak 3.9

Firefox 3.5 TraceMonkey

V8 svn

Concurrency Support in the IL?

Dalvik VM 1.0

Dis VM 4th ed.

OTP R12

Lua 5.1

mozart 1.4

CLI/ECMA-355

JVM spec.

Parrot 1.3

php 6.0 svn

python 2.6

Rubinius

Ruby 1.8/1.9

Self 4.4

Squeak 3.9

Firefox 3.5 TraceMonkey

Google V8 svn

only 6 out of 17

Survey Criteria - Concurrency

- Concurrency support
  - Concurrency model
  - Included instructions
  - Standard library (stdlib)

Common Language Infrastructure

- `volatile.` (prefix instruction)
  - marks a subsequent pointer reference
- Synchronized methods
- Memory model
  - Atomic read or write of certain aligned data
- Standard library
  - Memory barriers, atomic updates
  - Parallel loops, futures,…

Microsoft .NET	
Model	Threads/Locks
IL	Marginal
StdLib	Low-/high-level

Dis VM (spec. 4th ed.)

- Inspired by Communicating Sequential Processes
  - `spawn` – new thread
  - `new*` – new channel
  - `recv`, `send` – on given channel
  - `alt`, `nbalt` – send or receive on ready channel
- Memory model unspecified

inferno 4th EDITION

Model	CSP
IL	High-level
StdLib	High-level

Erlang

- Actors model
  - `send`, `wait`, `wait_timeout`
  - `remove_message`
  - `timeout`
- Pure, functional language
  - No memory model specified
  - Only high-level constructs in stdlib.

ERLANG	
Model	Actors
IL	High-level
StdLib	High-level

Java Virtual Machine

- `monitorenter/-exit`
  - For synchronized blocks
- Synchronized methods
- Memory model
- Standard library
  - Low- and high-level constructs
- DalvikVM promises the same

Model	Threads/Locks
IL	Marginal
StdLib	Low-/high-level

Mozart

- LOCKTHREAD
  - No unlock
- Implicit support
  - Data-flow variables, distribution
- Standard library
  - High-level constructs
  - Futures, stream channels,…

mozart	
Model	Data-flow
IL	Marginal
StdLib	High-level

Conclusion

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

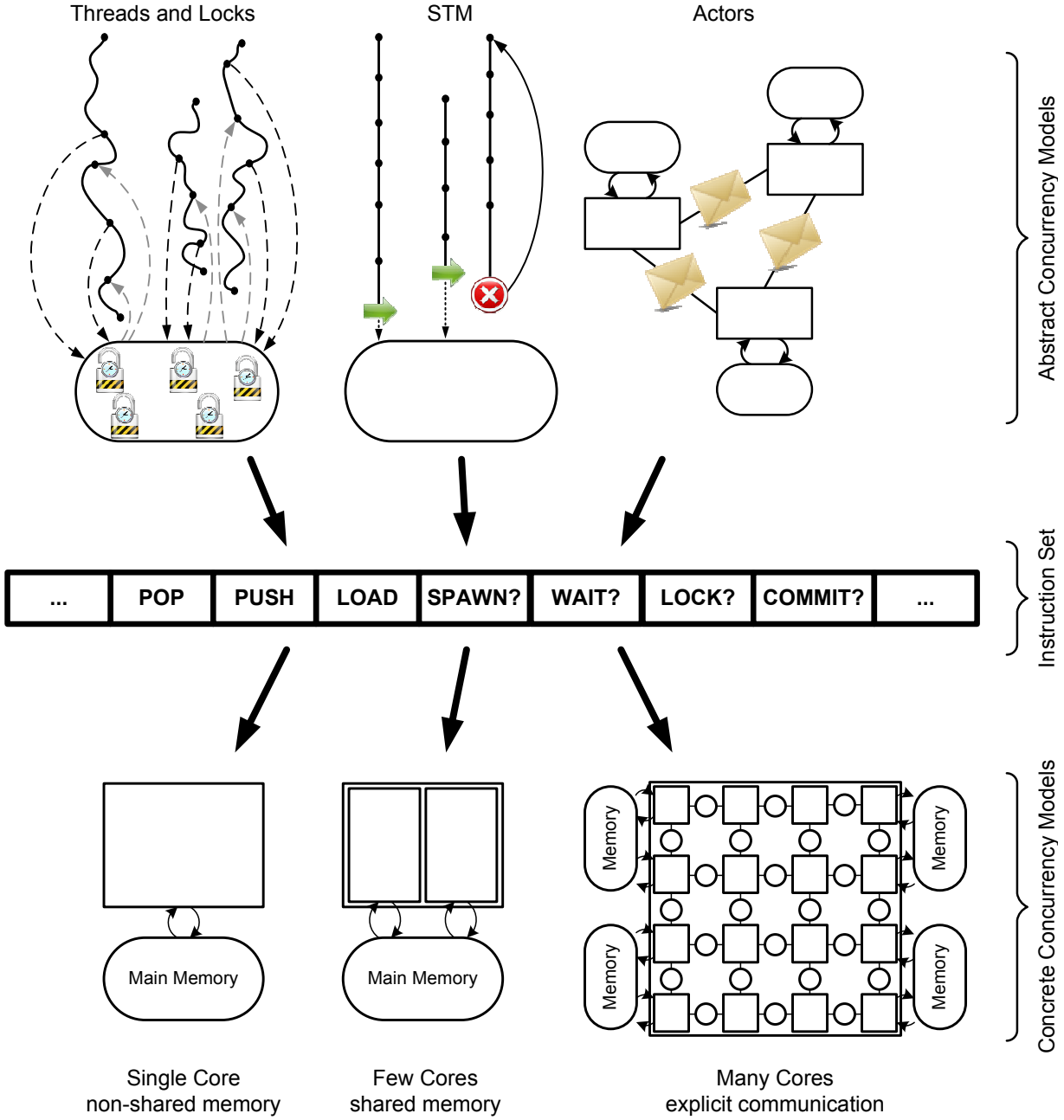
- Wide range of supported models
  - No notion of multiple models
  - Often limited to implicit semantics or guaranties
  - Often most functionality in standard library

Outlook

- Multi-language VMs have to
  - Enable language designer to invent new constructs/concepts
  - Provide low- and high-level constructs
    - Low-level: Memory barriers, atomic updates, …
    - High-level: Tuplespaces, STM, Actors, …

- Open question: tradeoffs IL vs. stdlib.

Discussion


Threads and Locks
STM
Actors
Abstract Concurrency Models
...
POP
PUSH
LOAD
SPAWN?
WAIT?
LOCK?
COMMIT?
...
Instruction Set
Main Memory
Main Memory
Memory
Memory
Memory
Memory
Concrete Concurrency Models
Single Core
non-shared memory
Few Cores
shared memory
Many Cores
explicit communication