The Architecture of the DecentVM
Towards a Decentralized Virtual Machine for Many-Core Computing

17 October 2010

Annette Bieniusa, Johannes Eickhold, and Thomas Fuhrmann

Department of Informatics, Self-Organizing Systems Group
Technische Universität München, Germany
Design Goal of DecentVM

Provide Single System Image
DecentVM’s Origin – Modular Embedded Devices

Bluetooth Module
Ethernet Module
Power-over-Ethernet Module
LiPo Battery Module
Backplane Module
Access Adapter
I/O Module
Peripheral Module

Ambicomp Project, 2005-2009
Current Target Architecture and Prototype
Objects in DecentVM

<table>
<thead>
<tr>
<th>Application level</th>
<th>VM level</th>
</tr>
</thead>
<tbody>
<tr>
<td>Java Object</td>
<td>Code</td>
</tr>
<tr>
<td>Static, Dynamic</td>
<td>Execution Context</td>
</tr>
<tr>
<td>Access via unique name</td>
<td>Access via known reference</td>
</tr>
</tbody>
</table>

- Java Object
  - Static
  - Dynamic
- Java Array
- Code
- Execution Context
Access via Reference and via Name

Saballus, Fuhrmann. Maintaining Reference Graphs of Globally Accessible Objects in Fully Decentralized Distributed Systems. HPDC'09
Consistency with DecentSTM

Bieniusa, Fuhrmann. Consistency in Hindsight, A Fully Decentralized STM Algorithm. IPDPS'10
Exploit STM Left-Over State for Redundancy

Blobs as Suites of Classes

VM internal class IDs

Blob #22CA3F37
Provides classes 0..12

Blob #46BA73B8
Provides classes 13..42
Requires 0..12 from #22CA3F37
Requires 43..59 from #3A815B17
Requires 60..71 from #553B7C8F

Blob #3A815B17
Provides classes 13..29
Requires 0..12 from #22CA3F37
Requires 30..41 from #553B7C8F

Blob #553B7C8F
Provides classes 13..24
Requires 0..12 from #22CA3F37

Mapping of blob class IDs to VM class IDs

Blob #22CA3F37
Provides classes 0..12

Blob #46BA73B8
Provides classes 13..42
Requires 0..12 from #22CA3F37
Requires 43..59 from #3A815B17
Requires 60..71 from #553B7C8F

Blob #3A815B17
Provides classes 13..29
Requires 0..12 from #22CA3F37
Requires 30..41 from #553B7C8F

Blob #553B7C8F
Provides classes 13..24
Requires 0..12 from #22CA3F37
DecentVM’s Memory Architecture

- Private pointer
- Private reference
- Managed pointer
- Mask
- Index
- RefCount
- Pointer
- ID
- Node
- Nonce
- ID
- Outgoing Reference Map
- Nonce
- InRefs
- ID
- Local Heap
- Local Application Instance
- VM Code References
- VM Reference Stack
- Global Memory Layer
- Incoming Reference Map
- to another node
- from another node

DecentVM’s Memory Architecture
Summary

• Fully decentralized Virtual Machine
• Memory Access by Name and via References
• Software Transactional Memory for Consistency
• Exploit STM as Extra Redundancy for Reliability

• Research VM (ca. 2000 lines of code ANSI C)
  – Bare metal implementation, i.e. no OS required
  – Offline transcoder transforms Java bytecode into DecentVM code
  – Traps to system library for float arithmetic, etc.

• References, Class IDs, etc. only Locally Valid
  – Need to map class IDs and runtime type info (RTTI)

• Transactions Exploit Private Memory
  – References within a transaction become public only upon commit
  – References need to be resolved only upon numeric GETFIELD
Thank you!

Thomas Fuhrmann

Department of Informatics
Self-Organizing Systems Group
Technische Universität München, Germany

fuhrmann@in.tum.de