Software Architectures for Morphing in Polymorphous Computing Architectures

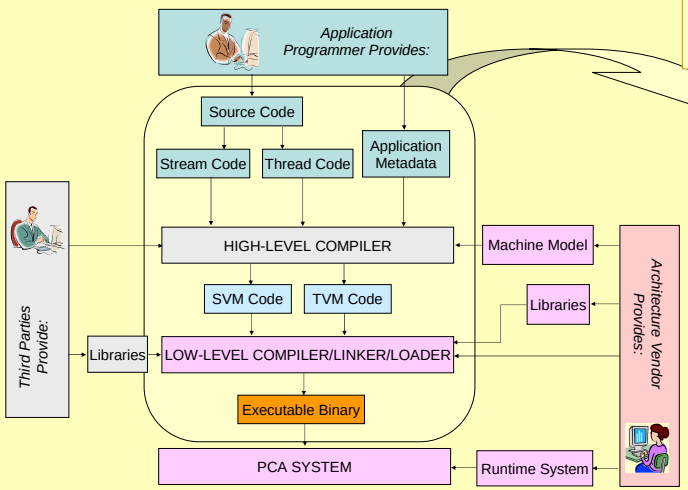
Dan Campbell, Mark Richards
Georgia Institute of Technology

Dennis Cottel, Randall Judd
USN SPAWAR Systems Center, San Diego

Morphware Stable Interface Architecture

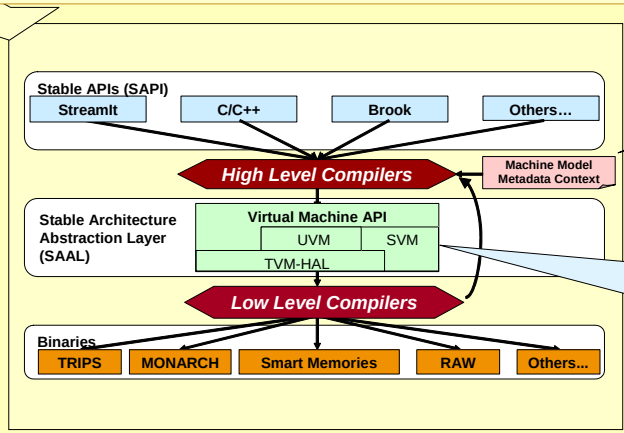
Development Process

- Two-stage compile process enables portable performance across PCA architectures



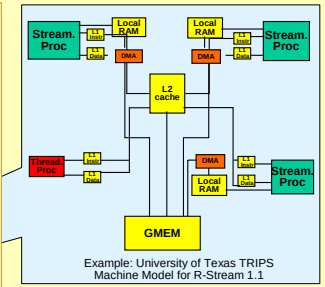
SAPI and SAAL

- Two intermediate representations
  - Stable API: application code in C/C++ and a stream language such as Brook or Streamit
  - Stable Architecture Abstraction Layer: PCA virtual machine code

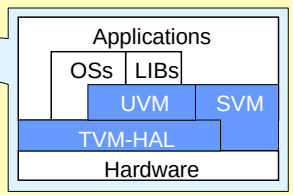


Machine Models

- Used to optimize VM output for different target platforms
  - Coarse grain mapping of application to target resources
- Describes target platform using common dictionary of virtual resources and attributes
  - Processors
  - Memories
  - Net links



Example: University of Texas TRIPS Machine Model for R-Stream 1.1



VM Layers

- User accesses User-level VM for thread code, Stream VM for stream code
- TVM-HAL abstracts low-level hardware to UVM

The Morphware Stable Interface

- Standard PCA Application Environment
  - Defined by a set of open standards documents
- Based on a virtual machine (VM) abstraction layer with standardized metadata and programming languages
- Goals
  - Foster software portability across PCA architectures
  - Dynamically optimize PCA resources for application functionality, service requirements, and constraints
  - Obtain nearly optimal performance from PCA hardware
  - Be highly reactive to PCA hardware and user inputs
  - Manage PCA software complexity
  - Leverage existing and developing technologies
- Cross-project effort, developed in parallel with the hardware

Morphing in the MSI

- MSI assumes component-based architecture
  - natural and intuitive boundaries for compilation and run-time reconfiguration
  - natural support for multiple SWEPT-variant implementations of units of functionality
- Morphing implies changing …
  - component implementations in use;
  - resources assigned to components;
  - or both
- Implies a taxonomy of morph types
- Morphing will be implemented at various levels of MSI
  - compiler
  - run-time system
  - resource manager

Morph Taxonomy

	Run-time System		Application Programmer		Compiling System	
	Components continue	Components change	Components continue	Components change	Components continue	Components change
Resource allocation doesn't change	Type 0a Run-time environment changes transparently to the running application.	Type 1a Run-time system changes components to reconfigured but equivalent set of resources.	Type 2a Application makes API call to make suggestions.	Type 3a Application makes API call to change processing mode but does so within existing resource set.	Type 4a Compiler instructions reconfigure allocated resources.	Type 5a Compiler switches to a different library able to use the same resources.
Resource allocation changes	Type 0b Run-time system changes resource allocation of a running application transparently to the application.	Type 1b Run-time system configures resources and loads components at application startup.	Type 2b Application makes API call to give up or gain some resources.	Type 3b Application makes API call to add or replace one or more components using different resources.	Type 4b Compiler requests different resources to meet change in performance specified by metadata.	Type 5b Compiler switches to a different library that uses different resources.

For more information: www.morphware.org