High Performance Embedded Computing Software Initiative (HPEC-SI)

Hahn Kim (MIT Lincoln Laboratory) and Sharon Sacco (MITRE)

HPEC-SI Co-chairs

OMG HPEC Working Group Meeting, Cambridge, MA

19 June 2012

Approved for Public Release: 12-2687. Distribution Unlimited
Evolution of Software Support Towards “Write Once, Run Anywhere/Anysize”

- Application software has traditionally been tied to the hardware
- Many acquisition programs are developing stove-piped middleware “standards”
- Open software standards can provide portability, performance, and productivity benefits
- Support “Write Once, Run Anywhere/Anysize”

Chart courtesy of MIT Lincoln Laboratory
Presented at 2001 HPC SIP Forum
Origins of the VSIPL Project

• Hughes Research Laboratory funded by DARPA to facilitate an Industry Working Group to develop a standard Library (and API) for Vector, Signal, and Image Processing
  – 24 months, October 1996 start
  – Principal Investigator: Dr. David Schwartz

• Initial working group included:
  – Hughes Research lab, Lockheed-Sanders, Northrop Grumman
  – Digital, Intel, Silicon Graphics/Cray, TI/Tartan
  – CSPI, Mercury Computers, Sky Computers
  – Alacron, Universal Computing, Sinectomy
  – ORINCON, Khoral Research

• Major Goal: Create a widely (industry) supported standard API/library for vector/signal processing primitives
  – Publicly available API documents and source code
  – Primarily targeted to embedded signal processing
  – API/Library for single processor and parallel version
VSIPPL Features

- VSIPPL stands for Vector Signal Image Processing Library
- Although C was the original language, VSIPPL has always been object based
  - VSIPPL only operates on memory in VSIPPL
  - VSIPPL memory is opaque to the user to allow optimization by the implementation
  - Vectors and matrices are contained within VSIPPL memory but are views of the memory
  - The attributes of the data structures strides and sizes is a part of the VSIPPL object and must be specified by the application.
    - VSIPPL calling sequences do not explicitly use attributes
- VSIPPL has standard function subsets called profiles
  - Profiles encourage new vendors by lowering initial costs
  - Applications written to a profile are portable with implementations that support that profile
VSIPL Functionality

• Vector functions
  – Basic arithmetic, trigonometric, transcendental, logical, bitwise, and selection functions

• Random number generator
  – Designed with parallel systems in mind

• Matrix functions
  – Elementwise functions similar to the vector functions

• Signal Processing
  – FFTs, window, convolution, correlation, histogram

• Linear Algebra
  – Outer and inner products, matrix multiply, matrix-vector multiply
  – Matrix solvers
    • LU, Cholesky, QRD, SVD
    • Special solvers (linear least squares, covariance, Toeplitz)
HPEC-SI
Overall Initiative Goals & Impact

Program Goals
• Develop and integrate software technologies for embedded parallel systems to address portability, productivity, and performance
• Engage acquisition community to promote technology insertion
• Deliver quantifiable benefits

Portability: reduction in lines-of-code to change port/scale to new system
Productivity: reduction in overall lines-of-code
Performance: computation and communication benchmarks

HPEC-SI Capability Phases

Phase 1
Applied Research: Unified Comp/Comm Lib
Development: Object-Oriented Standards
Demonstration: Existing Standards

Phase 2
Applied Research: Fault tolerance
Development: Unified Comp/Comm Lib
Demonstration: Object-Oriented Standards

Phase 3
Applied Research: Hybrid Architectures
Development: Fault tolerance
Demonstration: Unified Comp/Comm Lib

High-level code abstraction
• Reduce code size 3x

Unified embedded computation/communication standard
• Demonstrate scalability


Demonstrate insertions into fielded systems
• Demonstrate 3x portability
VSIPPL++ Features

- VSIPPL++ stands for **Vector Signal Image Processing Library for C++**
- Inherits many concepts from VSIPPL
  - Includes same numerical functions
  - VSIPPL++ memory is opaque to the user
  - Vectors and matrices are views of the memory
- VSIPPL++ is object-oriented
  - Classes and templates help reduce the amount of code needed to be written by the programmer
- Serial and parallel processing
  - Designed to easily scale serial VSIPPL++ code to run on multiple processors with minimal changes to code
VSIPL++ Functionality

• Includes same range of VSIPL functions:
  – Vector functions
  – Random number generator
  – Matrix functions
  – Signal Processing
  – Linear Algebra

• Vector, matrix and tensor views use template to describe:
  – Data types, e.g., int, float, double
  – Memory layout, e.g., row major, column major
  – Data parallelism, e.g., view is allocated on 1 or multiple processors

• Parallel views use map objects to concisely encode how data is distributed
  – List of processors IDs
  – Data distribution, e.g., block, cyclic, block-cyclic
Accomplishments

• VSIPL
  – 1.0 specification released in 2000
  – 1.3 specification released in 2008
  – Multiple vendor implementations (e.g., RunTime Computing Solutions, Mercury, GE Intelligent Platforms) and open source reference
  – Core Lite Test Suite 2002
  – CIP Demonstration 2003
  – GPU VSIPL available 2008

• VSIPL++
  – Serial specification released in 2005, updated in 2010
  – Parallel specification released in 2006
  – Successfully demonstrated insertion into fielded systems
**HPEC-SI Goals**

- **Standardize**: Works with other standards organizations, e.g., OMG, to maintain embedded software standards, e.g., VSIPL and VSIPL++, used in military embedded systems.

- **Develop**: Provides opportunities for researchers and vendors to present, collaborate, and evaluate new directions in embedded software, especially those that improve portability and productivity while retaining high performance. This effort feeds back into standardization activities to help extend existing standards.

- **Advocate**: Provide an opportunity for government organizations to discuss their open architecture requirements. This effort feeds back into standardization and development activities to ensure that the government’s needs are being met.