DoD Perspective on the Future of Embedded Software Development

Dr. Charles J. Holland
Director, Information Systems
Deputy Under Secretary of Defense (Science & Technology)
Joint Vision 2020
(Key Enabler - Information Superiority)

The joint force of 2020 will use superior information and knowledge to achieve decision superiority in order to reach full spectrum dominance, achieved through the interdependent application of:

The Lenses of Technological Innovation and Information Superiority Integrate and Amplify the Four Operational Concepts
Changing Strategic Environment

Global US Interests
Political - Economic - Humanitarian

Globalization of Technology

Asymmetric Threats
In any domain - Air, Land, Sea, Space or Information
Mission

to ensure that the warfighters today and tomorrow have superior and affordable technology to support their missions, and to give them revolutionary war-winning capabilities.

Defense Science and Technology
Director, Defense Research & Engineering

DIRECTOR, DEFENSE RESEARCH AND ENGINEERING
Honorable Hans Mark

DUSD, SCIENCE & TECHNOLOGY
Dr. Delores M. Etter

DIR, STRATEGIC & TACTICAL SYSTEMS
Dr. George R. Schneiter

DUSD, ADVANCED SYSTEMS & CONCEPTS
Mr. Joseph J. Eash, III

NUCLEAR TREATY PROGRAM
Dr. Ralph Alewine, III

CHEMICAL AND BIOLOGICAL DEFENSE PROGRAMS
Dr. Anna Johnson-Winegar

ODASD, NUCLEAR MATTERS
Dr. Frederick Celec

DIR, DEFENSE ADVANCED RESEARCH PROJECTS AGENCY
Dr. Frank L. Fernandez

DIR, DEFENSE THREAT REDUCTION AGENCY
Dr. Jay C. Davis

DIR, BALLISTIC MISSILE DEFENSE ORGANIZATION
LGen Ronald Kadish, USAF
HPCMP Investments in Real-Time Embedded HPC

Signal/Image Processing CTA
- Radar, Sonar, SAR, ATR, IR and Hyperspectral Image Exploitation

Integrated Modeling and Test CTA
- Tracking, Image Classification, RT Model Validation, Non-Uniformity Correction

PET Activities
- Yearly SIP Forum,
- VSIPL Tiger team
- RTExpress
- SIP Repository

DoD HPC Modernization Program
Outline

• The Challenge of High-Performance Embedded Computing
• The Current State of High-Performance Computing
• A Vision for the Future
• Summary
Embedded Processing Spectrum

<table>
<thead>
<tr>
<th>Platform</th>
<th>Size (liters)</th>
<th>Weight (kg)</th>
<th>Power (W)</th>
<th>Processing (GFLOPS)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Large radar/surveillance platforms (JSTARS, AWACS, AEGIS, THAAD)</td>
<td>1000's</td>
<td>500-1000</td>
<td>10-20 kW</td>
<td>1,000 – 10,000</td>
</tr>
<tr>
<td>Unmanned surveillance platforms (UCAV, Global Hawk, Discoverer II, etc.)</td>
<td>30 – 100</td>
<td>50 – 100</td>
<td>100–500 W</td>
<td>50 - 1,000</td>
</tr>
<tr>
<td>Use-once assets (MALD, torpedos, missiles)</td>
<td>1 – 30</td>
<td>1 – 50</td>
<td>~100 W</td>
<td>1 – 100</td>
</tr>
<tr>
<td>Micro-assets (micro-UAVs, small unit operations, smart dust)</td>
<td>0.01 – 0.1</td>
<td>0.1 – 1</td>
<td>~0 – 10 W</td>
<td>0.1 – 10</td>
</tr>
</tbody>
</table>

- Broad range of applications with very different stressing requirements
- All possess similar software development concerns
Evolution towards HPEC

- Market providing more hardware choices for embedded applications
- Software standards must evolve to support any hardware platform
The Challenge: Software Interoperability

HPC and Networked Computers

Embedded Computing Platforms

Standards-based Software Interoperability

- Move towards “write once, run anywhere”
  - Allows for joint development on workstation class machines
  - Provides easy porting on new machines, allowing for “technology refresh”

- Productivity, portability and scalability without sacrificing performance
  - Productivity: Easier code development and maintenance
  - Portability: Permits migration across platforms
  - Scalability: Allows for easy growth into evolving requirements
Model-Year Portability

COTS HPEC system lifetimes are **short**...

...military development and deployment cycles are **long**.

```
<table>
<thead>
<tr>
<th>System Development/Acquisition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Milestones</td>
</tr>
<tr>
<td>Technology Development</td>
</tr>
<tr>
<td>Field Demonstration</td>
</tr>
<tr>
<td>Engineering/Manufacturing</td>
</tr>
<tr>
<td>Development Insertion</td>
</tr>
<tr>
<td>Signal Processor</td>
</tr>
<tr>
<td>Gen 1</td>
</tr>
<tr>
<td>Gen 2</td>
</tr>
<tr>
<td>Gen 3</td>
</tr>
<tr>
<td>Gen 4</td>
</tr>
<tr>
<td>Gen 5</td>
</tr>
<tr>
<td>Gen 6</td>
</tr>
</tbody>
</table>
```

Processor support limited or dropped in later products

Portative software leverages inevitable advances in network and processor technology

```
Portable software with high performance is the best solution
```

“Point” solutions specific to a single vendor’s hardware or software are long-term cost **ineffective**

Increasing Network Performance
(latency, throughput, bisection bandwidth, ...)

Increasing Processing Capability (generations)

Increasing overall system performance

Vendor A product line
Vendor B product line
Military Software Complexity

- Moore’s Law addresses computation, not complexity!
- In 1995, 85% of military software projects finished over time and/or budget
- 1/2 of projects double cost estimates
- Projects slip an average of 36 months
- 1/3 of projects cancelled due to schedule/cost slips

- Software complexity of signal and image processing applications
  - System development overly complex already
  - Cannot let sensor processing compound the problem

June 9, 1999 “Defense Science and Technology: Preparing for the Future,” Presentation to DARPA; Dr. Delores M. Etter, DUSD (Science & Technology)
Outline

- The Challenge of High-Performance Embedded Computing
- The Current State of High-Performance Computing
- A Vision for the Future
- Summary
# Parallel Architectures and Operating Environments

<table>
<thead>
<tr>
<th>Architecture</th>
<th>Programming Model</th>
<th>Middleware</th>
<th>Ease of Use</th>
<th>Memory I/O Overlap</th>
<th>Real-Time Issues</th>
</tr>
</thead>
<tbody>
<tr>
<td>Symmetric Multiprocessor</td>
<td>Shared Memory / Single System Image</td>
<td>Threads, Compiler Pragmas</td>
<td>Easy</td>
<td>Threads</td>
<td>Non RT-OS</td>
</tr>
<tr>
<td>Hybrid</td>
<td>Shared Memory / Single System Image</td>
<td>Threads, Compiler Pragmas</td>
<td>Easy</td>
<td>Threads</td>
<td>Non RT-OS, Page Migration</td>
</tr>
<tr>
<td>Multicomputer</td>
<td>Distributed Memory Message Passing</td>
<td>Message passing</td>
<td>Hard</td>
<td>Threads</td>
<td>Non RT-OS</td>
</tr>
<tr>
<td>Multicomputer</td>
<td>Distributed Memory Message Passing</td>
<td>One-sided DMA Message passing</td>
<td></td>
<td>DMA engines (mainly)</td>
<td>RT-OS</td>
</tr>
</tbody>
</table>

For most applications, achieving good scalability requires expert architecture knowledge

High-level middleware standards can simplify this!

- Provide abstraction for application-level functions
  - Hide low-level mechanisms for parallel processing / communication
- Unify programming model across architectures
  - Eliminate ease-of-use issues between shared / distributed memory
  - Choose hardware based on mission needs alone
Roles of Current Standard Libraries

- Standards in the HPEC world fill three basic roles
  - Control communication
  - Data communication
  - Single processor computation
- The standards are not tightly integrated

In development:
- High-performance CORBA Extensions
- Data Re-org
# HPC/HPEC Middleware Assessment

## Computation Libraries

<table>
<thead>
<tr>
<th>Middleware</th>
<th>HPEC</th>
<th>HPC</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>VSIPL Standard</td>
<td>BLAS / LAPACK Standard</td>
</tr>
<tr>
<td></td>
<td>Vendor Proprietary</td>
<td>ATLAS / FFTW Vendor Proprietary</td>
</tr>
<tr>
<td>Standards Support</td>
<td>Emerging</td>
<td>Widespread</td>
</tr>
<tr>
<td></td>
<td>User Acceptance TBD</td>
<td>Broad User Acceptance</td>
</tr>
<tr>
<td>Object Oriented</td>
<td>VSIPL Object-Based</td>
<td>R&amp;D</td>
</tr>
<tr>
<td>Automatic Tuning</td>
<td>None</td>
<td>R&amp;D</td>
</tr>
<tr>
<td>Signal Processing</td>
<td>VSIPL API</td>
<td>No Industry Standard</td>
</tr>
<tr>
<td>Full Linear Algebra</td>
<td>VSIPL API Defined Emerging Implementations</td>
<td>BLAS &amp; LAPACK</td>
</tr>
</tbody>
</table>

## Overall Assessment

- **HPEC**: follow HPC lead (common interface mentality)
- Full use of object-orientation can reduce API complexity
- HPEC can leverage R&D in automatically tuning software
- Communities should work together to develop unified interfaces
## HPC/HPEC Middleware Assessment

### Communication and Integrated Libraries

<table>
<thead>
<tr>
<th></th>
<th>HPEC</th>
<th>HPC</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Middleware</strong></td>
<td>MPI Standard</td>
<td>MPI Standard</td>
</tr>
<tr>
<td></td>
<td>MPI/RT Standard</td>
<td>ScaLAPACK</td>
</tr>
<tr>
<td></td>
<td>Data Reorg (developing)</td>
<td>Proprietary Message Passing</td>
</tr>
<tr>
<td></td>
<td>Proprietary Message Passing</td>
<td></td>
</tr>
<tr>
<td><strong>Standards Support</strong></td>
<td><strong>MPI: Performance Concerns</strong></td>
<td><strong>MPI: Widespread</strong></td>
</tr>
<tr>
<td></td>
<td><strong>MPI/RT: Slow Adoption</strong></td>
<td><strong>MPI/RT: None</strong></td>
</tr>
<tr>
<td><strong>High-level Objects</strong></td>
<td><strong>Data Reorg (Object-Based)</strong></td>
<td><strong>HPF / ScaLAPACK (Procedural)</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Comm Only</strong></td>
<td><strong>Integrated Comm and Compute</strong></td>
</tr>
<tr>
<td><strong>Addresses</strong></td>
<td><strong>MPI/RT and Data Reorg</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Real-Time Concerns</strong></td>
<td>(Early Binding for Predictable Communication Performance)</td>
<td><strong>No</strong></td>
</tr>
</tbody>
</table>

### Overall Assessment

- Support for MPI is emerging in HPEC arena
- MPI performance for HPEC needs to be evaluated further
- No widely accepted standards that integrate communication and computation
Integrate Computation & Communication

Today’s Applications

- Computation Data (VSIPL)
- Communication Library (1 CPU)

Custom

- Computation Data (MPI)
- Communication Library (N CPU)

Processor

Interconnect

- Software standards developed separately
- Compute & communication data are stored differently
- Application code necessary to pass data between two environments

Tomorrow’s Applications

- Data Object
- Computation + Communication Library

Processors

Interconnect

- Co-develop standards
- Common compute & communication data format
- Seamless transition between two environments
Outline

- The Challenge of High-Performance Embedded Computing
- The Current State of High-Performance Computing
- A Vision for the Future
- Summary
Software for the Next Generation

New Applications Require New Approaches

Today’s Practice
- Low level protocols
- Procedural
- Large platforms
- Large teams
- Static signal flow
- Redundant hardware
- Separate Comp/Comm

Tomorrow’s Applications
- Standards based
- Object oriented
- Small platforms
- Small teams
- Dynamic flow
- Fault tolerant software
- Integrated Comp/Comm
Software for the Next Generation

Today’s Practice
- Low level protocols
- Procedural
- Large platforms
- Large teams
- Static signal flow
- Redundant hardware
- Separate Comp/Comm

Standards
- Allows portable applications
- Provides a common interface for designers
- Separates hardware from software
- Enables innovation in both domains

Tomorrow’s Applications
- Standards based
- Object oriented
- Small platforms
- Small teams
- Dynamic flow
- Fault tolerant software
- Integrated Comp/Comm
Software for the Next Generation

Today’s Practice
- Low level protocols
- Procedural
- Large platforms
- Large teams
- Static signal flow
- Redundant hardware
- Separate Comp/Comm

Object Oriented Programming
- Essential tool for modern software development
- Enables code re-use by hiding complexity
- Allows small teams to be more productive
- Better compiler optimization can achieve better performance

Ready for HPEC

Tomorrow’s Applications
- Standards based
  - Object oriented
- Small platforms
  - Small teams
- Dynamic flow
- Fault tolerant software
- Integrated Comp/Comm

Object Oriented Programming
Essential tool for modern software development
Enables code re-use by hiding complexity
Allows small teams to be more productive
Better compiler optimization can achieve better performance

Ready for HPEC
Software for the Next Generation

Today’s Practice
- Low level protocols
- Procedural
- Large platforms
- Large teams
- Static signal flow
- Redundant hardware
- Separate Comp/Comm

Tomorrow’s Applications
- Standards based
- Object oriented
- Small platforms
- Small teams
- Dynamic flow
- Fault tolerant software
- Integrated Comp/Comm

Reconfigurable Software

- New applications will be more complex and more dynamic
- Fault tolerance will be implemented in software and hardware
- Software needs to adapt to hardware
- Employ self-optimizing software techniques (e.g., FFTW & ATLAS)
Software for the Next Generation

Today’s Practice
- Low level protocols
- Procedural
- Large platforms
- Large teams
- Static signal flow
- Redundant hardware
- Separate Comp/Comm

Unify Computation & Communication
- Eliminates parallel programming code overhead
- Provides a clean interface for developers
- Enables truly scalable programs
- Allows 3rd party parallel software applications

Tomorrow’s Applications
- Standards based
- Object oriented
- Small platforms
- Small teams
- Dynamic flow
- Fault tolerant software
- Integrated Comp/Comm
Middleware as a Tool for Building High-Level Development Tools

Today

- High-level tool
  - (Often) Custom Communication Libraries
  - (Often) Custom Computation Libraries
  - Vendor Libraries / OS
  - Programmable Hardware

Tomorrow

- High-level tool
  - Unified Computation & Communication
  - Object Oriented Programming
  - Standard Communication Libraries
  - Standard Computation Libraries
  - Other Middleware (profiling, data collection, ...)
  - Vendor-Specific Optimizations
  - Programmable Hardware

- Migrate away from building custom infrastructure
- High-level tools need high levels of abstraction in middleware
- Compiler challenges for automatic mapping / code generation tools
Outline

- The Challenge of High-Performance Embedded Computing
- The Current State of High-Performance Computing
- A Vision for the Future
- Summary
Conclusions

• HPEC presents unique software challenges
  • Increasingly complex applications & constraints
  • Technology refresh and lifecycle support
  • Diverse platforms
• Standards are just beginning to emerge
  • HPC leads the way in software innovation
  • HPEC can leverage HPC approach and technology
• Vision: Write once, run anywhere
  • Move towards object-oriented standards
  • Unify computation & communication
DoD S&T is a Partnership

Focused Mission R&D

Expanded Resource Base

New Ideas, Knowledge

Service Labs

DARPA

High Risk, High Payoff

Maximum National Security Payoff

Interagency

Universities

Industries

International

Coalition Capability

Innovation, Transition
Backup slides
### Software and System Costs

<table>
<thead>
<tr>
<th>Software Class</th>
<th>Complexity</th>
<th>Performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Command and Control</td>
<td>Difficult</td>
<td>Moderate</td>
</tr>
<tr>
<td>Signal/Image Processing</td>
<td>Moderate</td>
<td>Demanding</td>
</tr>
</tbody>
</table>

- Software lines of code dominated by command and control, but...
Software and System Costs

<table>
<thead>
<tr>
<th>Software Class</th>
<th>Complexity</th>
<th>Performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Command and Control</td>
<td>Difficult</td>
<td>Moderate</td>
</tr>
<tr>
<td>Signal/Image Processing</td>
<td>Moderate</td>
<td>Demanding</td>
</tr>
</tbody>
</table>

- Software lines of code dominated by command and control, but...
- Hardware costs driven by SIP functions
Software and System Costs

- Software lines of code dominated by command and control, but...
- Hardware costs driven by SIP functions
  - Replication costs (per copy)
### Software and System Costs

<table>
<thead>
<tr>
<th>Software Class</th>
<th>Complexity</th>
<th>Performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Command and Control</td>
<td>Difficult</td>
<td>Moderate</td>
</tr>
<tr>
<td>Signal/Image Processing</td>
<td>Moderate</td>
<td>Demanding</td>
</tr>
</tbody>
</table>

- Software lines of code dominated by command and control, but...
- Hardware costs driven by SIP functions
  - Replication costs (per copy)
- Poor SIP software implementation will decrease architecture efficiency and drive up costs

Signal and Image processing software drives total ownership costs
- Development, hardware costs, maintenance, upgrades