Performance Engineering Initiatives for Early Software Test of High Availability Systems

Ed Beck
Sr. Manager

NDIA Test & Evaluation Conference
March 1-4, 2010
Mission Solutions Engineering, LLC

- Mission Solutions Engineering (MSE) is a **full service systems and software engineering provider** with 40 years’ experience in delivering mission systems.

- Formed by CSC to address government concerns for potential OCI, MSE comprises CSC’s former systems and software engineering support to the US Navy and Missile Defense Agency, primarily operating out of Moorestown, New Jersey and with headquarters in Arlington, Virginia.

- The core of our engineering activities is the design, development and integration of mission-critical software. We have been rated at Capability Maturity Model Integration1 Level 5 for software engineering (SW-CMMI®) since 2001 and systems engineering (SE-CMMI®) since 2004.
Aegis Weapon System

- The world’s most advanced shipboard Anti-Air Warfare (AAW) Weapons System
- A highly integrated Combat System capable of Multi-Mission warfare
  - Air, Surface and Subsurface
- Open Architecture migration provides foundation for the modernization effort and future war fighting capabilities

U.S. Navy Photo
Overview

- Performance Engineering (PE), Integration & Test
- Catalyst for PE Initiatives
- Performance Engineering Framework
- Summary
What is Performance Engineering?

- **Performance Engineering** is an emerging Computer Science practice comprised of the following functions:
  - Capacity Planning
    - Projection of future resource needs based on historical data and growth projections
  - Performance Measurements
    - Demonstrating that the system meets performance criteria
  - Performance Analysis
    - Characterizing software behavior and identifying anomalies

- **In practice, as we have defined it at MSE, Performance Engineering** is the process by which software is tested and tuned with the intent of realizing the required performance.
  - Viewed at MSE as an integral function within the Test & Evaluation (T&E) Discipline
What is Systems Integration & Test?

• Systems Integration & Test is the ability to verify and validate the interfaces, functions and performance of two or more computer programs on the target hardware suite and operating environment.

• Multiple Integration & Test levels:
  – Unit
    – Verifies functionality of specific sections of code
  – Integration
    – Verifies the interfaces and interactions of components against a specific design
  – System
    – Verifies the system performs intended functions and integrates with external or 3rd party interfaces
  – Requirements
    – Verifies that a completely integrated systems meets requirements
  – Acceptance
    – Validates the system meets customer needs
• Software testing efforts account for a large part of the software development costs
  – The earlier a defect is found, the cheaper it is to fix

• Incorporating Performance Engineering initiatives in Early Integration & Test efforts results in:
  – Better characterization of software and system performance levels
  – Earlier detection of performance defects which yield lower remediation costs
  – *Timely integration of 3rd party software solutions with better operational performance*
A Migration to Open Architecture

Proprietary Systems

Manufactured hardware and developed software

Open Technology

Emphasis on COTS hardware and software integration

C/C++

Conversion of legacy software to new languages

Heterogeneous operating environments

U.S. Navy Photo
Open Technology

• An Aegis ship is not much different from a large-scale commercial data center.

• The weapons system is composed of a heterogeneous operating environment with unique components not seen in commercial architectures.
An Enterprise Computing Environment

- Networked Computer Systems
- GUNS System
- Sensors
- Vertical Launch System
- Subsurface Systems
- Radar System

U.S. Navy Photo
Transformation to Open Architecture – Numerous challenges with integrating COTS technology for Real-Time High Availability computing requirements

<table>
<thead>
<tr>
<th>SPY, CND, WCS/FCS, ADS, ORTS, ACTS</th>
<th>Tactical Applications</th>
</tr>
</thead>
<tbody>
<tr>
<td>FTWSM, CIGARS</td>
<td>Middleware</td>
</tr>
<tr>
<td>Aegis Tactical Executive System</td>
<td>Operating Systems</td>
</tr>
<tr>
<td>UYK-7 / UYK-43 UYK-20 / UYK-44</td>
<td>Computing Platforms / Network</td>
</tr>
<tr>
<td>Custom NTDS Custom Time-of-Day</td>
<td>Interface Technology</td>
</tr>
<tr>
<td>UYH-3 (Mag Disk) RD-358 (Mag Tape) OL-267 (Data Terminal)</td>
<td>Peripherals</td>
</tr>
</tbody>
</table>

**COTS Transition**

- **Middleware**: Aonix, Galilium
- **Operating Systems**: concurrent, HAWK, WIND RIVER
- **Computing Platforms / Network**: IBM, CISCO
- **Interface Technology**: Symmetricom, Fanuc, GE Fanuc
- **Peripherals**: GERMANE SYSTEMS, HILL, AYDIN DISPLAYS, INC.
Performance Engineering Initiatives for Aegis Modernization

- **Analysis of Software Vulnerabilities**
  - Static SW scans and run-time monitoring
  - Cppcheck tool (Open Source Software) for SW scans

- **Infrastructure Performance Measurements**
  - Cpu, Memory, Network Interface utilization
  - Collectd tool (Open Source Software)

- **System/Software Profiling**
  - Process, function and code level graphical & textual reports
  - Oprofile tool (Open Source Software)

- **System-Level Technical Performance Measures (TPMs)**
  - Derived from A-spec requirements
  - Data Extraction and Reduction

- **Product-Area TPMs**
  - Derived from the B5 spec or assumed requirements
  - Data Extraction and Reduction

- **Predictive Engineering**
  - Analysis of planned effects of modifications to the system
  - SW performance based on capacity changes

* Refer to appendix for tool origin
Extensive Use Of Open Source Technology

• Off the shelf system management and performance analysis products are commercially available, but they are:
  – Targeted for commercial IT applications
  – Not easily adaptable to address specific user requirements
  – Highly complex solutions that require significant technical skills and training
  – Not cost effective: high licensing fees, support and training costs

• Open Source Benefits:
  – DoD acceptance of OSS
  – Lower cost ‘generic’ alternative to COTS products
  – OSS available at all levels of the software stack
  – Scalable, extensible components provide for innovative solutions
## Toolset

<table>
<thead>
<tr>
<th>Tool</th>
<th>Description</th>
<th>Network Analysis</th>
<th>CPU Utilization</th>
<th>Memory Utilization</th>
<th>Memory Leak Detection</th>
<th>Software Profiling</th>
</tr>
</thead>
<tbody>
<tr>
<td>cppcheck</td>
<td>Performs static analysis of C/C++ source code for software vulnerabilities</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>valgrind</td>
<td>Performs run-time analysis of executables for instances of memory leaks</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>collectd</td>
<td>Provides dynamic display of system and application performance – cpu, memory and network utilization. Historical data is available for trend analysis.</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>ntop</td>
<td>Provides run-time data on network utilization</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>NightTune</td>
<td>Provides dynamic display of system and application performance – cpu, memory and network utilization. Historical data can be correlated with NightTrace dataset for TPM analysis</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>NightTrace</td>
<td>Provides synchronized graphical or text display of all system activity</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Data</td>
<td>Provides analysis of tactical run-time extraction point data</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Recording</td>
<td>System-wide profiler for Linux systems, capable of profiling all running code at low overhead</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>oprofile</td>
<td>Provides run-time analysis of detailed network packet information</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>wireshark</td>
<td>Providing run-time analysis of detailed network packet information</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>

* Refer to appendix for tool origin
<table>
<thead>
<tr>
<th>Tool</th>
<th>Description</th>
<th>Network Analysis</th>
<th>CPU Utilization</th>
<th>Memory Utilization</th>
<th>Memory Leak Detection</th>
<th>Software Profiling</th>
</tr>
</thead>
<tbody>
<tr>
<td>cppcheck</td>
<td>Performs static analysis of C/C++ source code for software vulnerabilities</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>valgrind</td>
<td>Performs run-time analysis of executables for instances of memory leaks</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>collectd</td>
<td>Provides dynamic display of system and application performance – cpu, memory and network utilization. Historical data is available for trend analysis.</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>ntop</td>
<td>Provides run-time data on network utilization</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NightTune</td>
<td>Provides dynamic display of system and application performance – cpu, memory and network utilization. Historical data can be correlated with NightTrace dataset for TPM analysis</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>NightTrace</td>
<td>Provides synchronized graphical or text display of all system activity</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Data</td>
<td>Provides analysis of tactical run-time extraction point data</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Recording</td>
<td>System-wide profiler for Linux systems, capable of profiling all running code at low overhead</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>oprofile</td>
<td>Provides run-time analysis of detailed network packet information</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>wireshark</td>
<td>Provides run-time analysis of detailed network packet information</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

* Refer to appendix for tool origin
• Cppcheck ([sourceforge.net/projects/cppcheck](sourceforge.net/projects/cppcheck))
  – Source code analyzed for various software vulnerabilities
  – Report identifies specific lines of code with potential issues
  – Does not require software modifications
  – Benefits
    • FOSS
    • Noted issues are identified and resolved prior to run-time testing
    • Significant reduction in memory leak stability issues during initial software integration efforts

<table>
<thead>
<tr>
<th>Category</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Memory Leaks</td>
<td>The most basic example of a memory leak is when memory is dynamically allocated but not de-allocated when a function terminates.</td>
</tr>
<tr>
<td>Resource leaks</td>
<td>Occurs when an acquired resource cannot be released.</td>
</tr>
<tr>
<td>Array out of bounds</td>
<td>Occurs when accessing an element of an array using an index that is beyond the size originally allocated for the array.</td>
</tr>
<tr>
<td>Buffer overruns</td>
<td>Can occur when the destination of a copy is smaller than the source.</td>
</tr>
<tr>
<td>Memset on class</td>
<td>Memset is typically used as a convenient way to set all data members of a class to zero. However, if the class contains any virtual functions, it is likely that the C++ internal virtual pointer table will be overwritten causing the virtual pointer to point to NULL.</td>
</tr>
<tr>
<td>Overlapping data buffer</td>
<td>Can occur when specifying the same source and destination for a memory operation</td>
</tr>
<tr>
<td>Missing virtual destructors</td>
<td>In object oriented programming, and in this case C++, it is common to use inheritance to define a “derived” class to extend a previously defined “base” class. This derived class may then perform any valid C/C++ instructions, including dynamically allocating memory in the constructor, and then (hopefully) deleting the memory in the destructor. A problem can occur because it is possible to delete/destroy the object using the destructor of the base class. Doing this will not execute the destructor in the derived class and therefore not delete the dynamically allocated memory which causes a leak.</td>
</tr>
<tr>
<td>Mismatching alloc/dealloc</td>
<td>Using the “new” operator in C++, it is possible to dynamically allocate memory for a pointer to a single object, and a pointer to an array of objects. In both cases, the “delete” operator is used but the syntax differs between the two types of allocations. In the case of the array, “delete” requires the use of the square bracket operators for proper de-allocation to occur.</td>
</tr>
<tr>
<td>Division with signed &amp; unsigned operators</td>
<td>The result of a division will be wrong if one operand is unsigned and the other operand is negative.</td>
</tr>
</tbody>
</table>

**Example Output:**

<table>
<thead>
<tr>
<th>FILE</th>
<th>LINE#</th>
<th>ERROR CATEGORY</th>
<th>ERROR TYPE</th>
<th>VARIABLE NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>file1.cpp</td>
<td>21</td>
<td>(error)</td>
<td>memory leak</td>
<td>test</td>
</tr>
<tr>
<td>file2.cpp</td>
<td>110</td>
<td>(error)</td>
<td>resource leak</td>
<td>class::create</td>
</tr>
<tr>
<td>file10.cpp</td>
<td>204</td>
<td>(error)</td>
<td>overlapping data buffer</td>
<td>template</td>
</tr>
</tbody>
</table>
• **Collectd** (collectd.org)
  
  – Ability to monitor and analyze CPU, memory and network utilization at the system and element level
    
    – Synchronized process-to-infrastructure views
    
    – Per-core and aggregate CPU statistics
  
  – Non-intrusive capability available on-demand
  
  – Benefits
    
    • FOSS
    
    • Provides critical infrastructure TPM’s
    
    • Allows detection and investigation of anomalies early in the development cycle, thus ensuring greater stability in later deliveries
• **NightTrace** ( Concurrent Computer Corporation)

  – Deterministic debugging, monitoring, tracing and tuning
  – Ideal for time-critical applications
  – Synchronized graphical or text display of system and application activity

  – Benefits

  • Easy-to-use graphical user interface
  • Provides user-defined event logging
  • Customizable views
  • Correlate kernel events with application activity

---

**NightStar Tool Suite**

Concurrent Computer Corporation

Provides visibility of application timing issues at the Operating System level
**Data Recording** (U.S. Navy - Aegis / LM / MSE)

- Utilizing extraction point data to implement a TPM “report card” for system and application timing and capacity requirements,

- Extraction point data is recorded during all test events

- Benefits
  - Application instrumentation
  - Ability to toggle EP recording on/off at system consoles
  - Used for timing & capacity measurements, SW studies, and correlation of anomalies

<table>
<thead>
<tr>
<th>Cornerstones</th>
<th>TPM</th>
<th>Build x</th>
<th>Build y</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>CPU Utilization Normal</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>CPU Utilization Stress</td>
<td>Not Tested</td>
<td>Not Tested</td>
</tr>
<tr>
<td></td>
<td>Memory Utilization</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Network Utilization</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Initialization</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reaction Time</td>
<td>Tracking</td>
<td>Not Tested</td>
<td></td>
</tr>
<tr>
<td></td>
<td>AAW in Adverse</td>
<td>Not Tested</td>
<td>Not Tested</td>
</tr>
<tr>
<td></td>
<td>AAW in Clear</td>
<td>Not Tested</td>
<td>Not Tested</td>
</tr>
<tr>
<td></td>
<td>Display</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Firepower</td>
<td>Engagement</td>
<td>Not Tested</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Firing Rate</td>
<td>Not Tested</td>
<td>Not Tested</td>
</tr>
<tr>
<td></td>
<td>MAX** – AAW</td>
<td>Not Tested</td>
<td>Not Tested</td>
</tr>
<tr>
<td></td>
<td>MAX** – Multimission</td>
<td>Not Tested</td>
<td>Not Tested</td>
</tr>
<tr>
<td>Coverage</td>
<td>Track Capacity</td>
<td>Not Tested</td>
<td></td>
</tr>
</tbody>
</table>

**Example TPM Report Card**
**Oprofile** (sourceforge.net/projects/oprofile)

- Shows most frequently executed software threads
- Very low overhead
- Benefits
  - FOSS
  - Allows engineers to understand the behavior of the software
  - Reveals excessive processing / loops
  - Identifies areas for further instrumentation

---

**Source-Code Annotation**
Adaptable Performance Engineering Initiative

- Highly configurable suite of open source, commercially available, and developed tools that can provide measurements and diagnostics across the enterprise
Performance Engineering Initiatives

- Increasing our ability to characterize and analyze complex systems in a COTS environment

Benefits
- Improves System Stability
- Early Identification & Resolution of Issues
- Reduces System T&E Failure Rates

Summary
- Increasing our ability to characterize and analyze complex systems in a COTS environment

- Improves System Stability
- Early Identification & Resolution of Issues
- Reduces System T&E Failure Rates

Operational
- Reqts Definition

System
- Reqts Definition

Element
- Reqts Definition

CP Performance
- Reqts Definition

Computer
- Program Design

Comp Program
- Detailed Design

Initial Test and Performance
- Measurement

Early I & T, PE
- Activities

Code

Unit Test

Element CP
Integration & Test

Engineering
Test & Evaluation

System Demo &
System Qual Test

System T&E,
MEIT

Operational
Tests

Systems
Integration,
Test and
Performance
Measurement

U.S. Navy Photo
Edward Beck  
Sr. Manager  
Applied Integration Solutions  

MSE, LLC  
304 West Route 38  
Mooresetown, NJ  08057  

(856) 252-2055  
ebeck@missionse.com
# Tool Reference (Origin)

<table>
<thead>
<tr>
<th>Tool</th>
<th>Description</th>
<th>Organization</th>
</tr>
</thead>
<tbody>
<tr>
<td>cppcheck</td>
<td>Performs static analysis of C/C++ source code for software vulnerabilities</td>
<td>Sourceforge/projects/cppcheck</td>
</tr>
<tr>
<td>valgrind</td>
<td>Performs run-time analysis of executables for instances of memory leaks</td>
<td>Valgrind.org</td>
</tr>
<tr>
<td>collectd</td>
<td>Provides dynamic display of system and application performance – cpu, memory and network utilization. Historical data is available for trend analysis.</td>
<td>Collectd.org</td>
</tr>
<tr>
<td>ntop</td>
<td>Provides run-time data on network utilization</td>
<td>Ntop.org</td>
</tr>
<tr>
<td>NightTune</td>
<td>Provides dynamic display of system and application performance – cpu, memory and network utilization. Historical data can be correlated with NightTrace data-set for TPM analysis</td>
<td>Concurrent Computer Corporation</td>
</tr>
<tr>
<td>NightTrace</td>
<td>Provides synchronized graphical or text display of all system activity</td>
<td>Concurrent Computer Corporation</td>
</tr>
<tr>
<td>Data Recording</td>
<td>Provides analysis of tactical run-time extraction point data</td>
<td>U.S. Navy / LM / MSE</td>
</tr>
<tr>
<td>oprofile</td>
<td>System-wide profiler for Linux systems, capable of profiling all running code at low overhead</td>
<td>Sourceforge/projects/oprofile</td>
</tr>
<tr>
<td>wireshark</td>
<td>Provides run-time analysis of detailed network packet information</td>
<td>Wireshark.org</td>
</tr>
</tbody>
</table>