Introducing HP NonStop Development Environment Version 2.0 for Eclipse (NSDEE 2.0)

Swaroop Dutta
Steve Williams
Seth Hawthorne

May 6, 2010
Welcome

- **Objective**: Provide a “sneak peek” preview of NonStop Development Environment for Eclipse (NSDEE)

- **Audience**: Developers of NonStop applications

- **Topics**
  - Background
  - NSDEE Overview
  - NSDEE Tour: Edit→Build→Deploy→Debug
  - NSDEE Remote Projects
  - NSDEE Debugging
  - Migration
  - Conclusion
Background
Changing Nature of NonStop Software Development

- Gone are the days of a single platform and single technology stack
  - Developers increasingly work with multiple technologies and multiple platforms
  - Open source has become a transformational and driving force
  - Pace of technology evolution continues to increase
  - Organizations value “standard skills” to reduce training and increase flexibility
  - Organizations want to optimize software development cycle time

- Increasingly difficult for proprietary tools to meet these needs

- Response: NonStop Modernization initiative
  - Embrace standards and open source
  - Provide industry standard tools
  - Leverage and extend the industry standard Eclipse Integrated Development Environment (IDE)
The Journey from Proprietary to Industry Standard Tools

- Many have started but are at different points along the way
- The biggest challenge is embracing CHANGE
- Expectations are key
- Requires some “give and take”

  • Challenges
    - Tools familiar to long-time developers are no longer available
    - Sometimes less integrated
    - Often have to rely on documentation from several sources
    - Tools may have some “rough edges” or lack customizations

  • Benefits
    - New capabilities previously unavailable with proprietary tools (debugger memory leak detection)
    - Increased flexibility
    - Faster rate of evolution
    - The ability to use the same tools and skills across platforms
Integrated Development Environment 101

– A GUI for software development
  • Simplifies software development workflows (edit→build→deploy→debug)
  • Integrates development tools
  • Language and API aware editing
  • Provide information about program structure

– Users
  • Some prefer IDEs, some prefer command-line tools
  • Not necessarily an either/or
  • Increasingly important to “next gen” developers

– Industry leading IDEs have converged on:
  • Visual Studio
  • Eclipse and Eclipse derived products
  • NetBeans
The Power of Eclipse

- Leading “open source” IDE
- Rich plug-in ‘ecosystem’
- Architecture
  - Java based
  - OSGi Component model
  - Used as an IDE platform
  - Used as a framework for building stand-alone applications
- Runs on many platforms
- Innumerable IDE frameworks and plug-ins:
  - Java Development (JDT)
  - C/C++ Development (CDT)
  - NSDEE
  - SQL Explorer
  - UML modeling
  - Embedded device development
  - ...
- Yearly major releases
NSDEE Overview
Introducing NSDEE

- “Go forward” NonStop Integrated Development Environment (IDE)
  - Supersedes Enterprise Plug-ins for Eclipse (EPE)
  - Platform for future IDE functionality; Visual Studio-based ETK is mature

- Eclipse-based
  - Industry standard IDE
  - Common development desktop for multiple platforms and technologies
  - Users benefit from rich ecosystem of extensions

- Supports
  - Windows-based (“local”) development
  - NonStop-based (“remote”) development
  - IDE Integrated Debugging for Integrity NonStop systems
NSDEE Targets all NonStop developers

- Windows hosted cross-compiler users
  - Can run existing makefiles from NSDEE
  - Can use the IDE while continuing to use command-line tools

- Enterprise ToolKit (ETK) users
  - Can import existing projects to NSDEE
  - Benefit from the cross-platform nature of eclipse
  - Can leverage more third-party extensions

- NonStop hosted users
  - First NonStop IDE to support NonStop hosted development!
  - Build on OSS or Guardian

- Visual Inspect users
  - Can use NSDEE during the edit->build->deploy->debug cycle
  - Still use Visual Inspect for non-eclipse built applications and when features not yet supported by NSDEE debugging are required

- Enterprise Plug-in For Eclipse (EPE) users
  - Free upgrade to NSDEE!
What’s different from EPE?

- Integrated Integrity NonStop debugging
  - Launch and debug programs from the IDE!

- Launch programs from the IDE

- Improved NonStop connectivity
  - SSH support including certificate based login
  - Browse and edit remote files
  - NonStop connection manager

- Significantly improved eclipse integration
  - Eclipse based packaging and installation (via Update Manager)
  - Improved integration between local and remote development

- Problem view support for NonStop remote builds
  - Click on diagnostic messages to navigate to and fix build issues

- More robust and scalable

- Improvements to standard make build model
The NSDEE Software Stack

- **NSDEE Core**
  - Supports C, C++, COBOL, pTAL
  - Local edit and build
  - Remote edit and build

- **NSDEE Debug**
  - Integrated debugging for Integrity NonStop systems

- **NSDEE builds on eclipse CDT**

- **NonStop Java developers can install the standard eclipse JDT**
NSDEE Build Models

Local Build

– Execute build on Windows
  • Source files reside on Windows
  • Uses NonStop cross-compilers with cygwin make

– Standard make model
  • User defines and manages the makefile
  • NSDEE defines environment variables

– Managed make model
  • IDE uses knowledge of dependencies to generate and maintain the makefile
  • Useful for small programs, less so for large ones

Remote Build

– Execute build on NonStop system
  • Source files and tools reside on NonStop
  • OSS make
  • Guardian GMAKE

– Standard make model

– Allows the IDE to be used with legacy NonStop applications
NSDEE Tour:
Edit ➔ Build ➔ Deploy ➔ Debug
First a few Key Eclipse Concepts

- **Eclipse Perspective** – A collection of views that are useful for a particular activity. With NSDEE you will use:
  - NonStop Local perspective
  - NonStop Remote perspective
  - Debug perspective

- **Eclipse View** – A tabbed pane within a perspective. You can control which views are displayed, drag-and-drop views to different locations, and double-click a view’s tab to maximize or minimize it.

- **Eclipse Workspace** – A container for organizing a collection of related projects. State that persists between eclipse sessions is managed on a per-workspace basis. NSDEE recommends against combining local and remote projects in the same workspace.

- **CDT** – C/C++ Development Toolkit. Base eclipse framework that NSDEE builds on. Tip: CDT documentation is a useful reference
Getting NSDEE

Prerequisites

– General
  • Windows XP or Vista (32-bit)
  • Java version 5 (or later)
  • ~200 MB disk space

– Local build
  • NonStop cross-compilers
  • Cygwin 3.8.1 (or later)

– Remote build
  • Integrity NonStop system

– Integrated debugging
  • Integrity NonStop system
  • Native Inspect T1237H01^AAT (or later)

Obtaining NSDEE

– Two bundles
  • NSDEE Core
  • NSDEE Core with Debugging

– Customers
  • Try before you by (60-day trial version)
  • Order as independent product

– Internal users
  • Contact presenters
Installation

1. Install the eclipse software stack or use an existing installation
   - Eclipse 3.4.x (Ganymede)
   - C/C++ Development Toolkit (CDT) version 5.0.2 ❗️ exact version required

2. Run Eclipse

3. Use the Update Manager to install NSDEE
   Help ➔ Software Updates…

Eclipse does not alter the Windows registry.
You can easily maintain multiple configurations (disk space permitting)
Open the NonStop Local Perspective

1. Open the NonStop Local Perspective
   Window ⇒ Open Perspective ⇒ Other… ⇒ NonStop Local

![Diagram of Eclipse IDE with NonStop Local perspective open]

- Current perspective
- Workspace project view
- C/C++ outline view
- Source edit view
- Build problems view
Create a Local “Hello NSDEE” Project

2. Create a project: HelloNSDEE
   File ⇒ New Standard Make NonStop Local Project
Create a Local “Hello NSDEE” Project

3. Create folders to organize your files
   Right Mouse Menu: New ⇒ Source Folder…: Source
   Right Mouse Menu: New ⇒ Source Folder…: Build

4. Create a source file: hello.c
   Select the source folder
   Right mouse menu: New ⇒ Source File

5. Enter source text, save, and close the file:

   Missing semicolon error
   Syntax highlighting
   Outline view
Create the makefile and Configure Compilers

6. Edit the makefile

```makefile
TARGET_EXE  = build/hello.exe
TARGET_OBJS = build/hello.o
TARGET_LIBS = -l libc.so -l libc.so -l libld.so
LIB_ROOT    = $(COMP_ROOT)/usr/lib
C_MAIN      = "${LIB_ROOT}/ccplmain.o"
C_OPTS      = -I header -g -Woptimize=0 -Wcall_shared -Wintdir=build/
ELD_OPTS    = -warn -set systype OSS -allow_duplicate_procs "-l $(LIB_ROOT)"

all: $(TARGET_EXE)

$(TARGET_EXE) : $(TARGET_OBJS)
    eld $(C_MAIN) $(ELD_OPTS) $(TARGET_LIBS) $(TARGET_OBJS) -o $(TARGET_EXE)

build/%.o: source/%.c
    c89 $(C_OPTS) -c $< -o $@

.PHONY: clean
    clean:
    rm -f build/*. *
```

7. Configure local compilers

**NonStop Tools ➔ Configure Compilers…**
Build

8. Build

- Disable automatic build (if enabled)
  Project ⇒ Build Automatically (is unchecked)

- Build the project
  Select project
  Right Mouse ⇒ Build Project

9. Oops there was an error

- Double click on the error message in the Problems view to open the source file
- Add the missing semi-colon
- Build the project
Define a Connection to the NonStop System

- Define a connection:

  **NonStop Tools ➔ Configure Connections…**

  • Specify the system
  • Specify user
  • Select protocol
  • Click: **Connect**

Connection information is stored per eclipse Workspace. You can export and import configurations between workspaces.
Create a Launch Configuration

- Defines how to launch (run or debug) a project’s executable on the NonStop system

1. Create
   - Invoke **Debug Configurations**...
   - NonStop Application
   - Complete required fields
   - Optionally specify additional info

2. **Apply** and **Close**

   Copy local executable to NonStop
Run the Program

1. Open the Run Configuration
2. Click Run
3. View output in console window
Debug the Program

1. Launch the Debug Configuration.
   Either:
   • Select previously used configurations from the menu
   • Or select from the Debug Configurations... dialog

   ![Debug Configurations dialog]

   ![Confirm Perspective Switch]

This kind of launch is configured to open the Debug perspective when it suspends.

This Debug perspective is designed to support application debugging. It incorporates views for displaying the debug stack, variables and breakpoint management.

Do you want to open this perspective now?

Remember my decision

Yes
No
The Debug Perspective

Stack View

Source View

Current loc

Execution Control

Variables

Breakpoints

Consoles
NSDEE Remote Projects
NSDEE Remote Projects

- Use the NonStop Remote perspective
  - Edit NonStop hosted files (OSS or Guardian)
  - Use makefile or build script to build with NonStop hosted tools
    - OSS make (a GNU make)
    - Guardian GMAKE (GNU make subset)
  - Use the NSDEE debugger

- Improves EPE capabilities
  - Problems view lists remote build errors
  - Edit arbitrary remote files
  - Browse NonStop hosted files
  - Improved handling of file types
NonStop Remote Perspective

Remote Projects

Remote Directories

Projects
NSDEE Debugging
NSDEE Debugging

- **Focus**: debug phase of the IDE workflow
  - Debugs eclipse-built applications during development
  - NSDEE is not yet a stand-alone or production debugger
  - Does not yet support some NonStop debugging features (info opens, …)

- Supports debugging C, C++, and COBOL on Integrity NonStop systems

- Recommend compiling at optimization level O0 during development

- Uses Native Inspect as “back-end”

- Some Debug Perspective features are not available with NonStop
  - Watchpoints, Step Filters, …

- Controlling GUI update/step performance
  - Use debug configuration to set defaults
  - Enable/disable items

*Note: Enabling automatic tracking slows stepping performance.*
Session and Execution Control

- Execution control
  - Resume, Suspend, Step in, Step Over, Step Out
  - Run To Line

- Right Mouse menu operations
  - Run To Line
  - Resume At Line

- Attach to a running process using a NonStop Attach Process Debug Configuration

- Session control
  - Stop
  - Disconnect
Breakpoints

- Setting code breakpoints
  - Click breakpoint column in source view
  - or use Right Mouse: **Toggle Breakpoint**
- Breakpoints View lists breakpoints
- Enable/disable breakpoints
- Breakpoint properties
- Breakpoints persist per-workspace between sessions
- Can import/export
Variables View

- Displays local variables for selected stack frame
- Displays specified globals
  - Add
  - Remove
- Changed values are highlighted
- Operations
  - Expand/collapse
  - Enable/disable
  - Cast To Type…
  - Display As Array…
  - Change Value…
  - Format
- Can also display C/C++ variables using the mouse pointer in the source view
Expressions View

- Displays specified variables and expressions

- To add an expression either
  - Right Mouse: **Add Watch Expression**...
  - or drag-and-drop text from source view
  - or drag-and-drop an item from the variables view

- Expressions are reevaluated in the current context each time the program is suspended.
Other Debugging Views

- Debug view displays
  - Program execution stack
  - Debugger process
  - Process being debugged

- Modules view displays DLLs (shared libraries)

- Signals view
  not supported in this release
Machine-Level Debugging (1 of 2)

- Disassembly view
- Step instructions
- Instruction breakpoints

- Memory view
  - Monitors display memory in different formats
  - Modify memory
  - Changes highlighted
Machine-Level Debugging (2of2)

- Registers view
  - Enable/disable to control performance
  - Changes highlighted

- Operations
  - Change value...
  - Cast To Type...
  - Create Watch Expression...
  - Format...
Console commands

- A sub-set of Native Inspect commands can be entered from the debugger console
  - `b location`
  - `catch load | unload | stop | abend`
  - `info open`
  - `ih` and `mh`
  - `mab` and `dmab`
  - `print`
  - `save`
  - `Vq`

1. Click debugger Debug view item
2. Enter command in Console view
Migration
Migrating from EPE

1. Select the NonStop perspective that matches the type of EPE project (local or remote)

2. Use the Import wizard to import EPE projects
Migrating from ETK

1. Select the NonStop Local perspective

2. Choose **Import ETK Project**...

PATH variables and macros are not migrated
Conclusion
Conclusion

– NSDEE extends the industry standard Eclipse IDE to support NonStop
  • Builds on an IDE that is available for many platforms
  • Offers a rich ecosystem of open source and third-party extensions

– NSDEE targets all modes of NonStop development

– NSDEE is the first NonStop IDE to support
  • NonStop hosted (“remote”) development
  • Integrated debugging

– NSDEE offers significant functional and integration improvements over EPE

– NSDEE is the future platform for NonStop IDE and debugger GUI enhancements
We Value Your Feedback

– We are always interested in your feedback
– Especially on:
  • Do you expect to use eclipse with other platforms?
  • The Importance of Linux hosting?
  • ETK features that you would like to see in NSDEE?
  • Additional debugging capabilities
Thanks!