NEC

SX-Aurora TSUBASA

NEC Parallel Debugger User's Guide

(G2AT02E)

Proprietary Notice

Introduction

This document explains how to use NEC Parallel Debugger.

How to Read

This document is composed of the following chapters. The rightmost column is target readers of the corresponding chapter.

Chapter	Title	Description	Target Readers
1	Overview	The overview of NEC Parallel Debugger	Programmers
2	Creation of a Project	Creation of a project in Eclipse PTP	Programmers
3	Building of a Project	Building of a project in Eclipse PTP	Programmers
4	Creation of a Debug Configuration	Creation of a Debug Configuration in Eclipse PTP	Programmers
5	Operations for Debugging	Operations for debugging in Eclipse PTP	Programmers
6	Notices and Restrictions	Notices and restrictions for NEC Parallel Debugger	Programmers

Related Documents

- C/C++ Compiler User's Guide (G2AF01E)
- Fortran Compiler User's Guide (G2AF02E)
- NEC MPI User's Guide (G2AM01E)

Remarks

- This manual conforms to Release 1.0.0 and subsequent releases of the NEC Parallel Debugger.

Glossary

Term	Description
Vector Engine (VE)	Vector Operation Engine implemented as a PCI Express Card attached to an x86 server. This is the core component of the SX-Aurora TSUBASA system.
Vector Host (VH)	An x86 server equipped with VEs.
Node	A VE, which has a shared memory.
NQSV	The NEC Network Queuing System V, which is a batch processing system for high-performance cluster system.
NEC MPI	MPI (Message Passing Interface) implementation by NEC. MPI is A specification for a standard library for communication. It can be used together with OpenMP or automatic parallelization.
Eclipse Parallel Tools Platform (PTP)	An integrated software development environment for parallel applications, which is an open source software.
View	A subwindow displayed in Eclipse window.
Perspective	The name given to an initial collection and arrangement of views and an editor area.
Target Process	A process targeted for debugging.

Contents

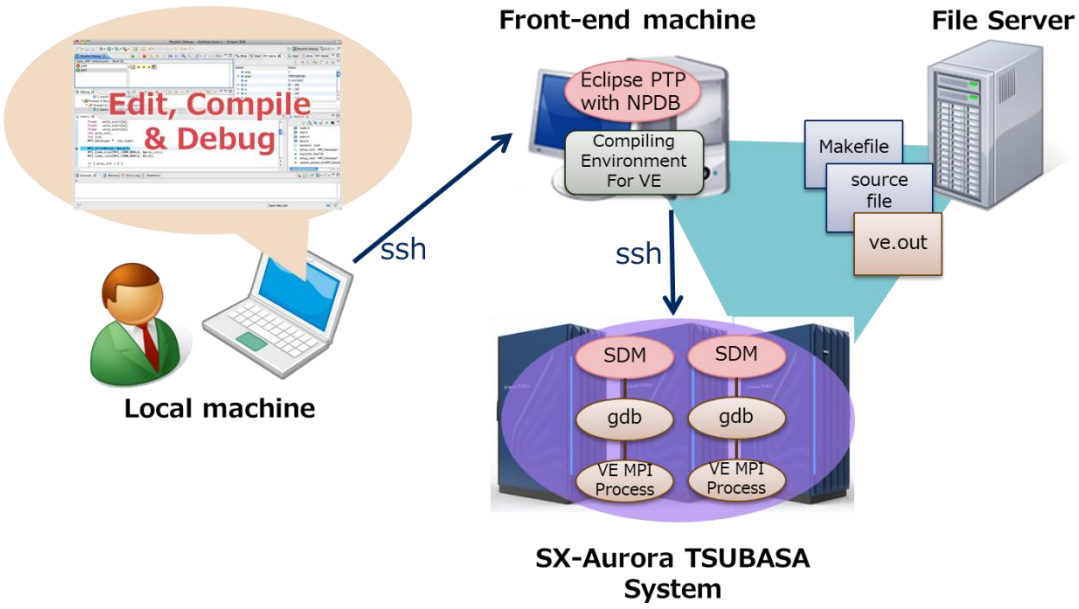
1 Overview

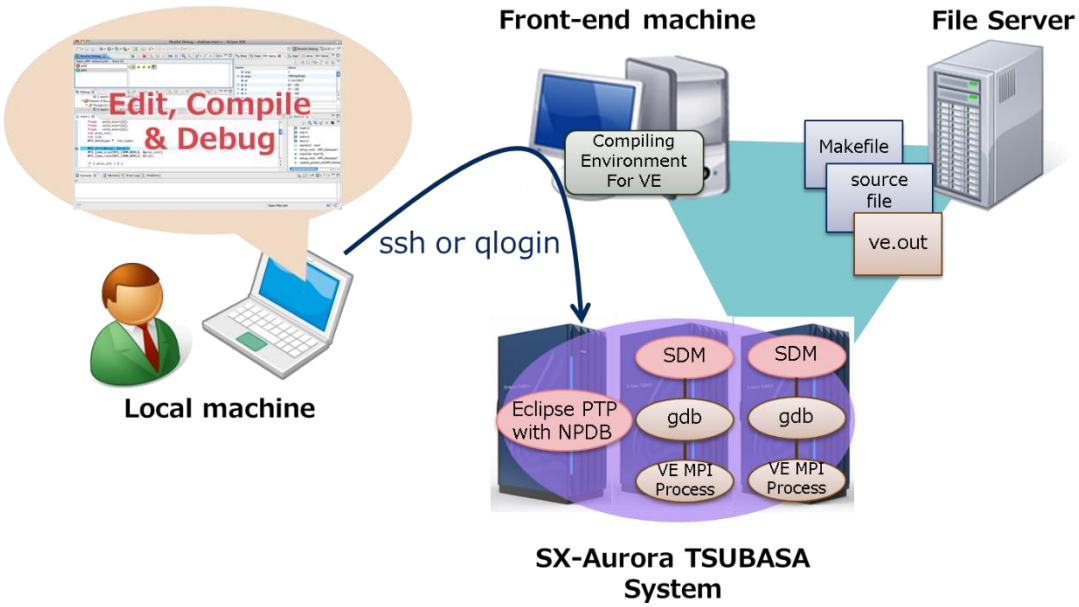
1.1 NEC Parallel Debugger

NEC Parallel Debugger is an Eclipse PTP plugin for debugging of MPI applications. Eclipse PTP is an open source integrated software development environment for parallel programs. Eclipse PTP into which NEC Parallel Debugger is integrated enables effective debugging of distributed-memory parallel programs with MPI in addition to shared-memory parallel programs written in Fortran and C/C++ with OpenMP or automatic parallelization.

1.2 Operating Environment

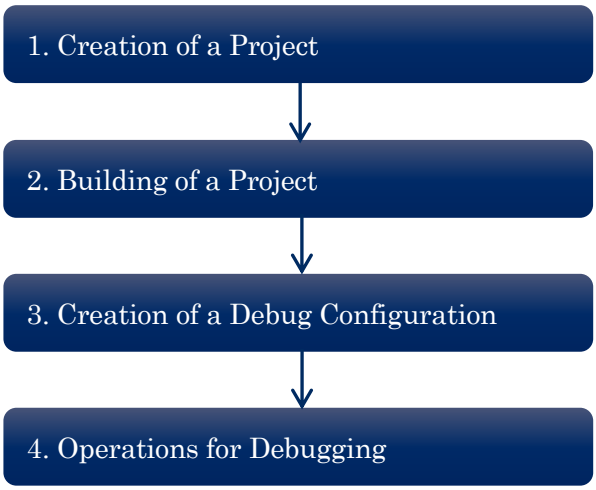
Users invoke NEC Parallel Debugger on a front-end machine to which they have logged in via ssh etc, and can edit, compile, and debug software launched by ssh as shown in the first figure below. It is also possible to invoke NEC Parallel Debugger on the SX-Aurora TSUBASA to which they have logged in via ssh or qlogin which is used for interactive request supported by NQSV as shown in the second figure below. When a debug execution for a MPI application using NEC Parallel Debugger starts, Scalable Debug Manager (SDM) is invoked on the SX-Aurora TSUBASA system. Then SDM initiates gdb and an MPI application targeted for debugging.





1.3 Steps for Debugging

The following figure shows the debugging steps for the first time. The steps one through three are not needed for the second time and afterwards. The following chapters explain each step for debugging applications using NEC MPI launched on VE.



2 Creation of a Project

Development of applications with Eclipse requires creation of a project, which is an environment under which users develop and debug an application. This chapter explains the steps for creating a project and importing an existing make environment for the following two cases:

- Import of a make environment on a remote host
- Import of a make environment on the local host

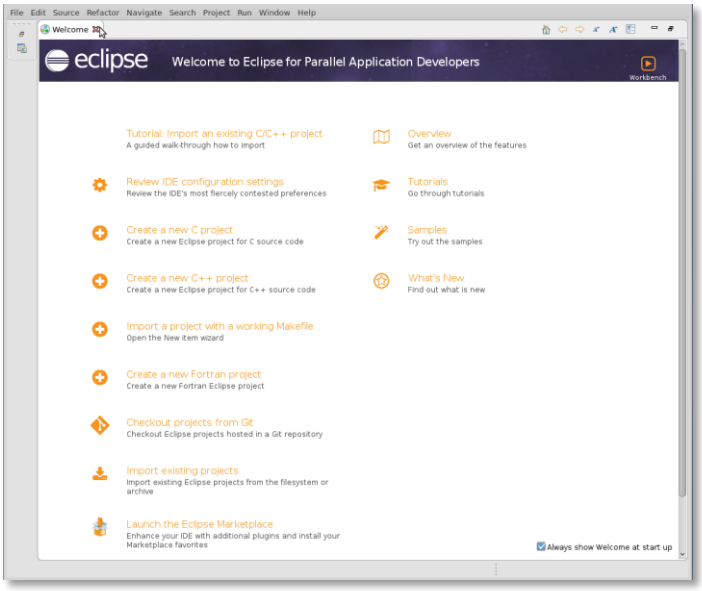
2.1 Invocation of Eclipse

Execution of the following command invokes Eclipse, and a window opens.

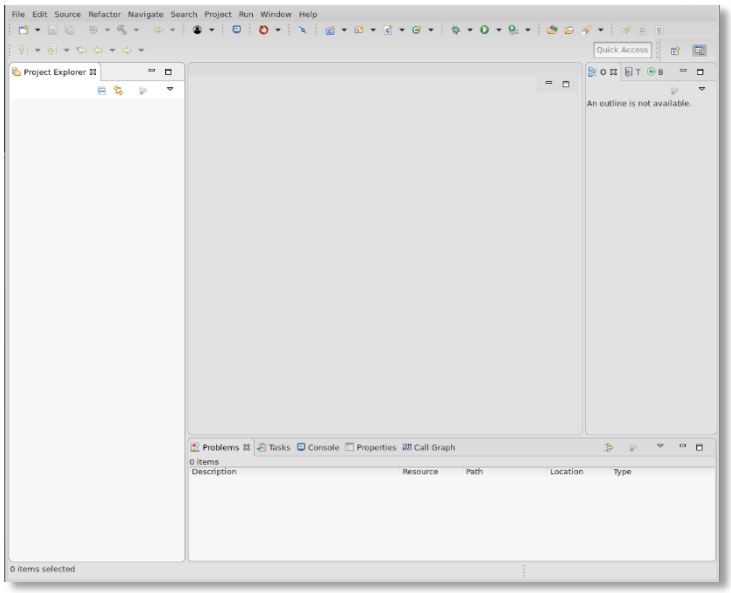
```
% /<INST-PATH>/eclipse/eclipse
```

where *<INST-PATH>* is the path under which Eclipse is installed.

By inputting the Workspace directory, which is the working directory for a project, on the window, the Welcome view is displayed as shown in the following figure.



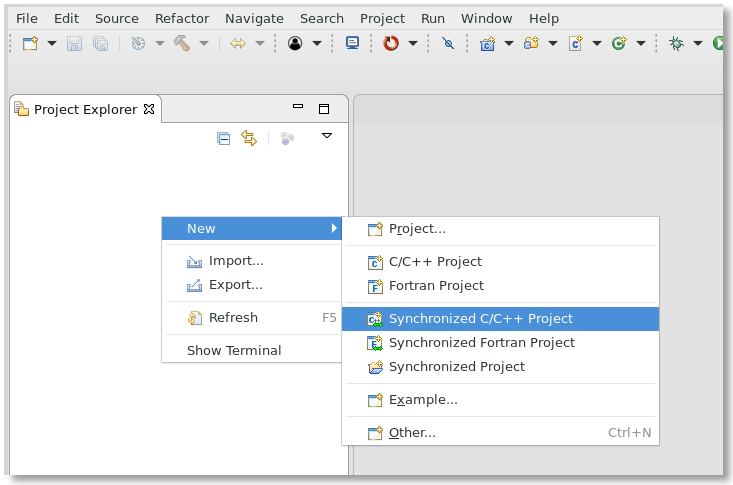
Please close the Welcome view by clicking the X sign on the tab, and the C/C++ perspective, which is a set of views for C/C++, is displayed as shown in the following figure.



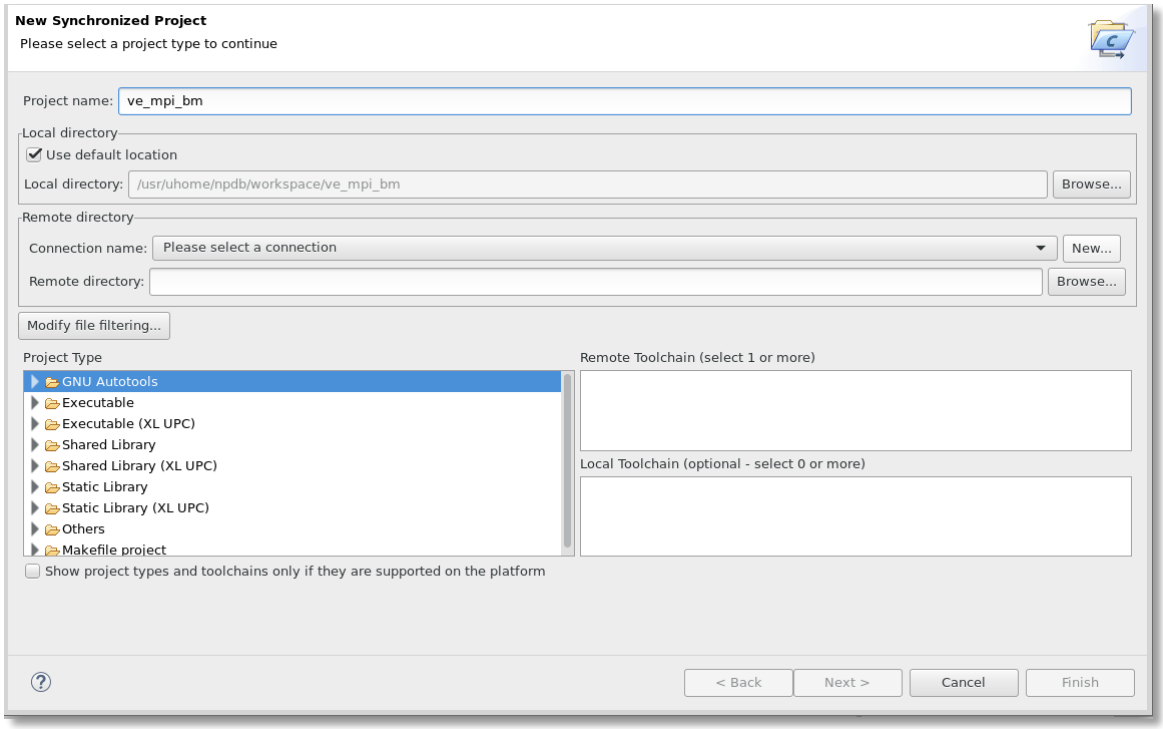
2.2 Import of a Make Environment on a Remote Host

It is recommended to create a synchronized project so that changes under the project such as modifications of source files are automatically reflected into the original environment on the remote host.
The following is the steps for creating a synchronized project and import an existing make environment.

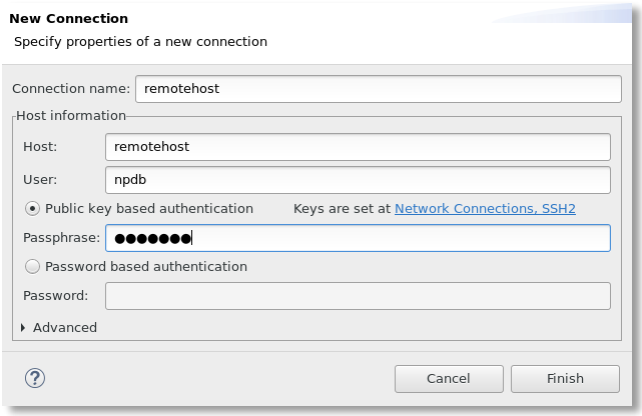
1. Right-click on empty space in the Project Explorer view and select “New > Synchronized C/C++ Project” or “Synchronized Fortran Project” to open the New Synchronized Project window.



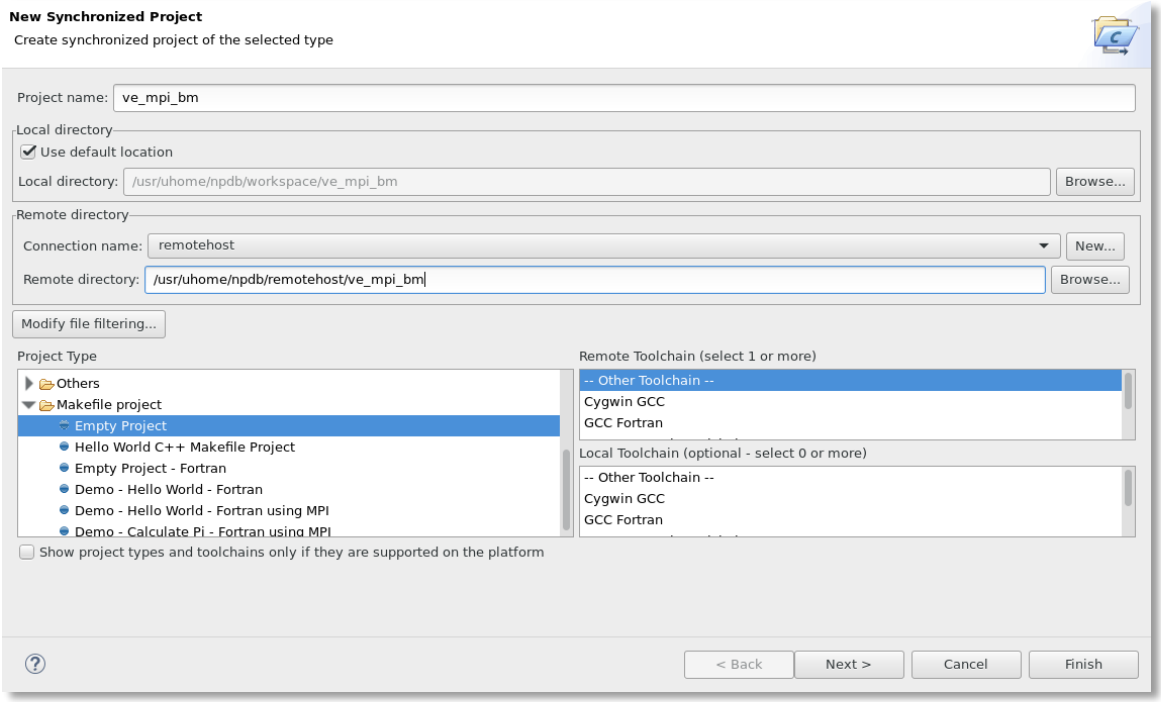
2. Input a project name in the Project name field on the window, and then click on the New button in the Connection name line.



3. Input in the entry fields such as Connection name, Host, User, and Passphrase, and click on the Finish button.



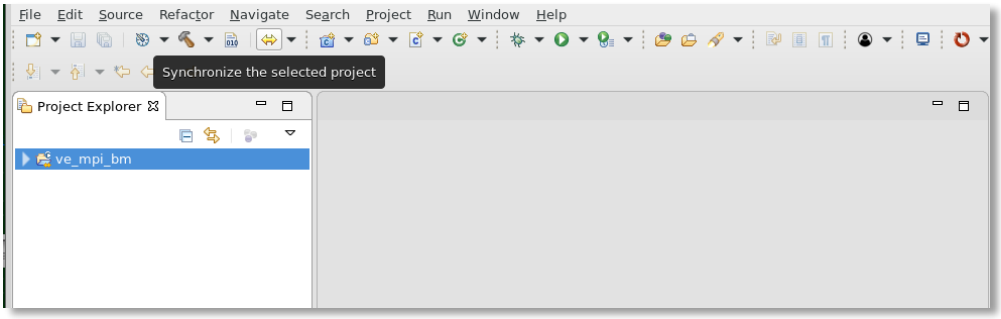
4. Input in the Remote directory field, select “Makefile project > Empty Project” in the Project Type field, and click on the Finish button.



The creation of a synchronized project and import of a make environment are now complete.

Note that changes under the project are not reflected into the original environment at this point.

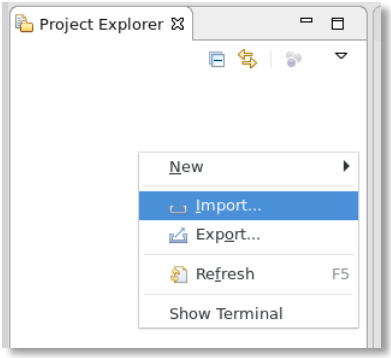
Please select “Synchronized Project” in the Project Explorer view and click on the button in the menu to activate the reflection.



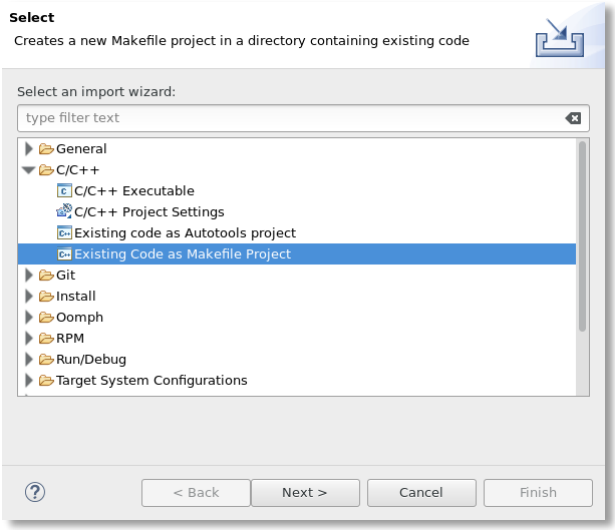
2.3 Import of a Make Environment on the Local Host

The following is the steps for creating a project and import an existing make environment on the local host. The working directory for the project is the same as that of the original make environment.

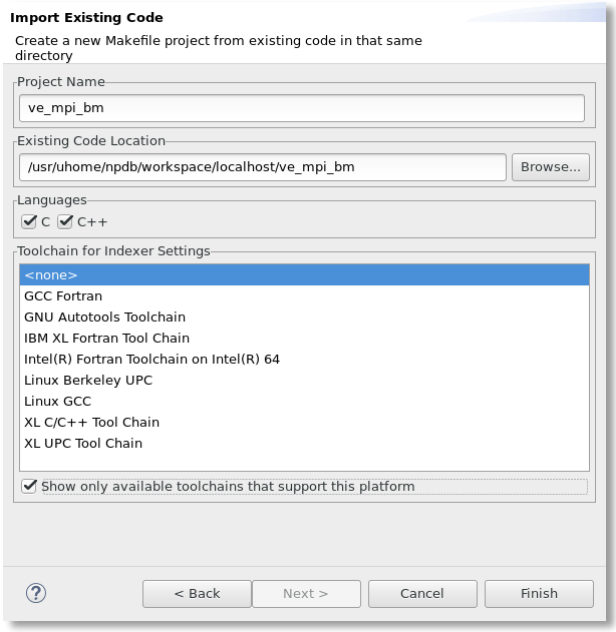
1. Right-click on empty space in the Project Explorer view and select “Import” to open the Import window.



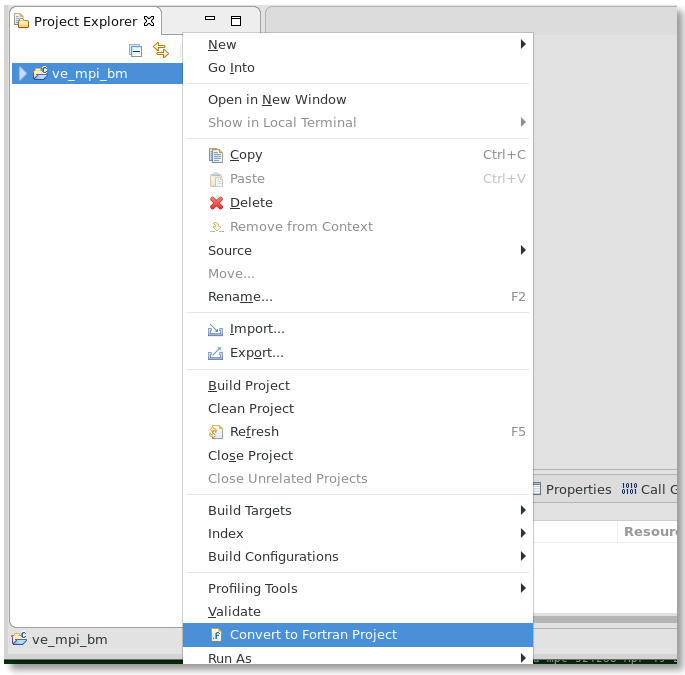
2. Select “C/C++ > Existing Code as Makefile Project” on the window, and click on the Next button to open another window.



3. Input a project name in the Project Name field and the directory where the make environment exists in the Existing Code Location field, and select “none” in the Toolchain field on the window that opened at step 2.



4. If your code is written in Fortran, select “Convert to Fortran Project” in the Project Explorer view to convert from a C/C++ project to a Fortran one.



The creation of a project and import of a make environment are now complete.

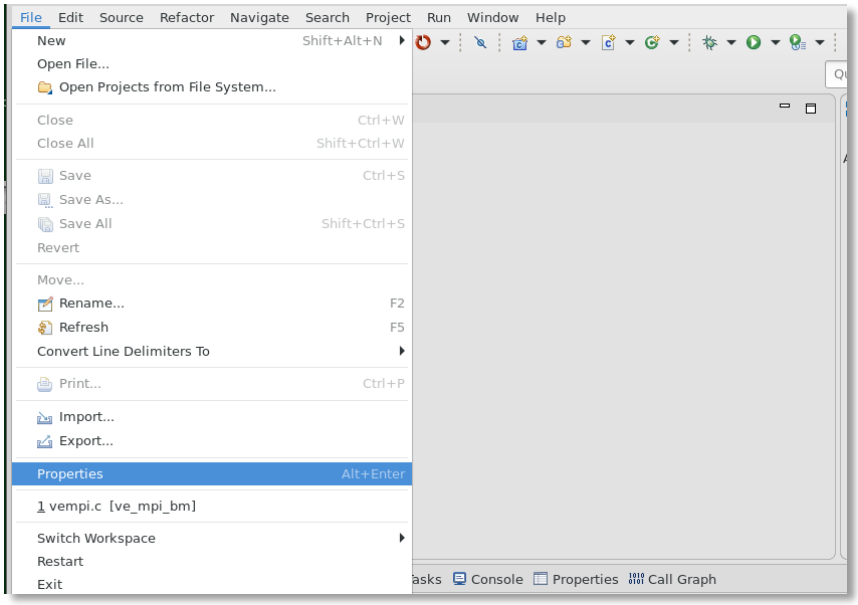
3 Building of a Project

This chapter explains how to configure a project and generate an executable application.

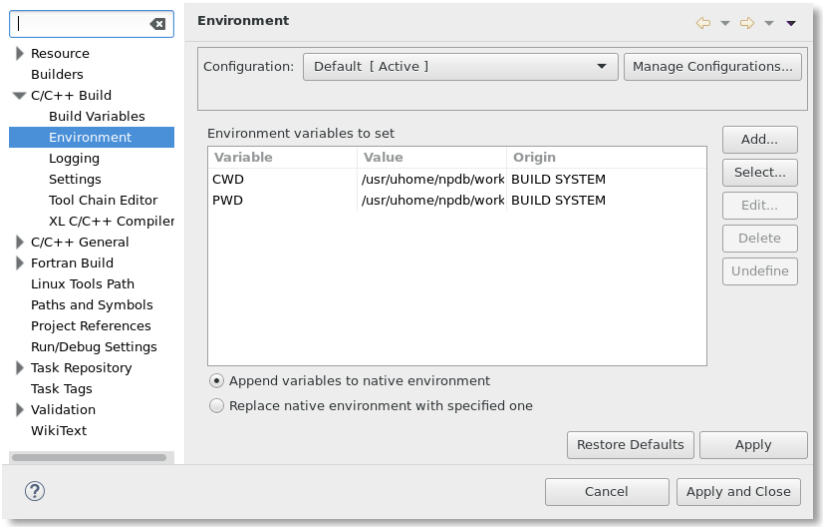
3.1 Setting of the Build Configuration

The following explains the steps for setting the paths to compilers and header files required for generating an application.

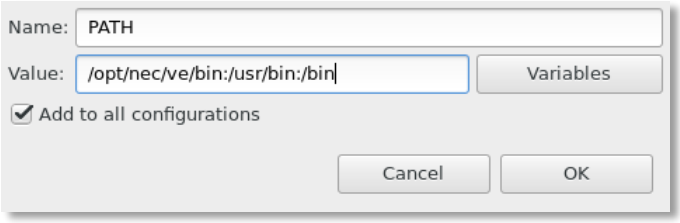
1. Right-click on the project name in the Project Explorer view and select “Properties”.



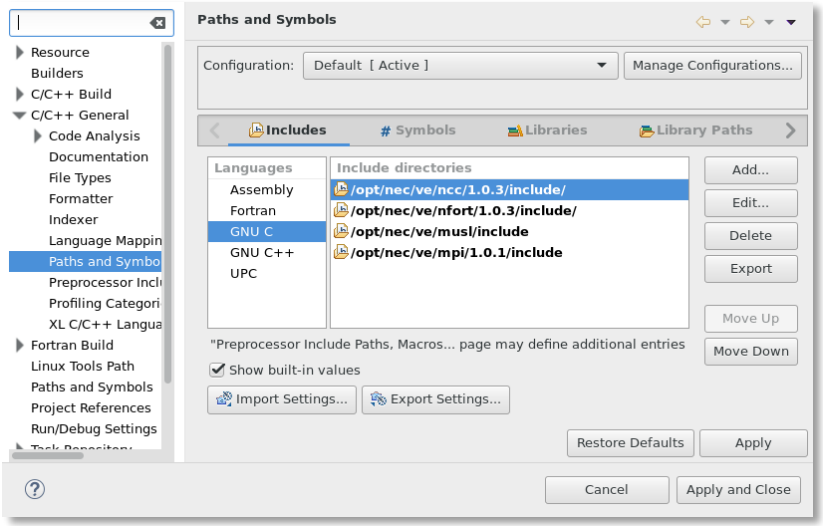
2. Select “C/C++ Build > Environment > Add” and set the value of the environment variable PATH. Please select “C/C++ Build” for a Fortran project, too.



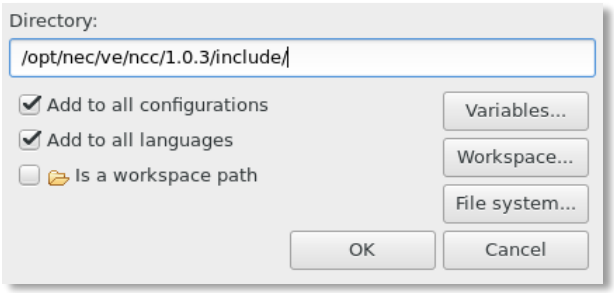
The value of the environment variable PATH needs to include the paths to all the commands required for generating the application.



3. Select “C/C++ General > Paths and Symbols > Add” and set the directories under which include files referenced in the source files are placed. Please select “C/C++ General” for a Fortran project, too.



The following example is the window to input a directory.



3.2 Execution of a Build Project

Right-click on the project name in the Project Explorer view and select “Build Project” to generate the application.

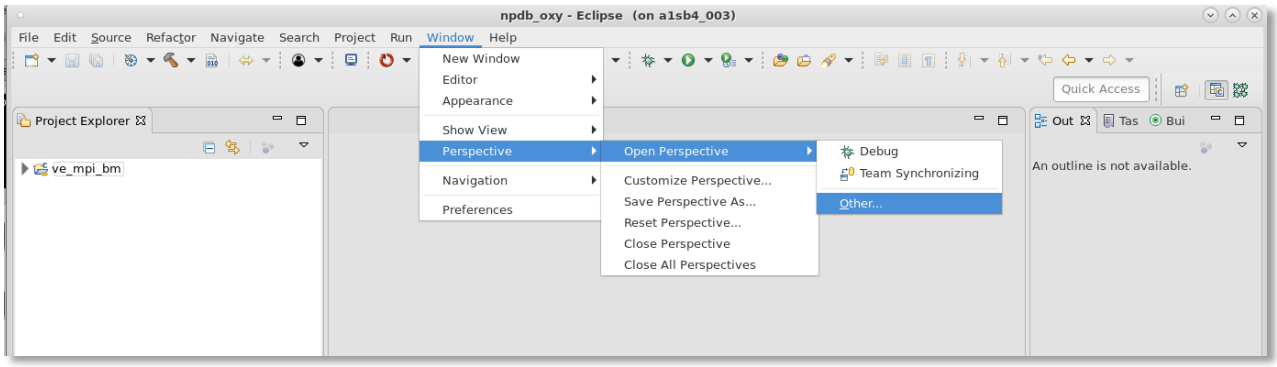
4 Creation of a Debug Configuration

This chapter explains how to configure the settings for debug and start a debug execution.

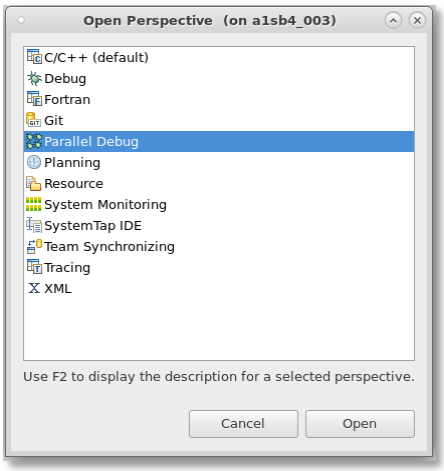
4.1 Creation of a Debug Configuration

The following is the steps for creating a Debug Configuration.

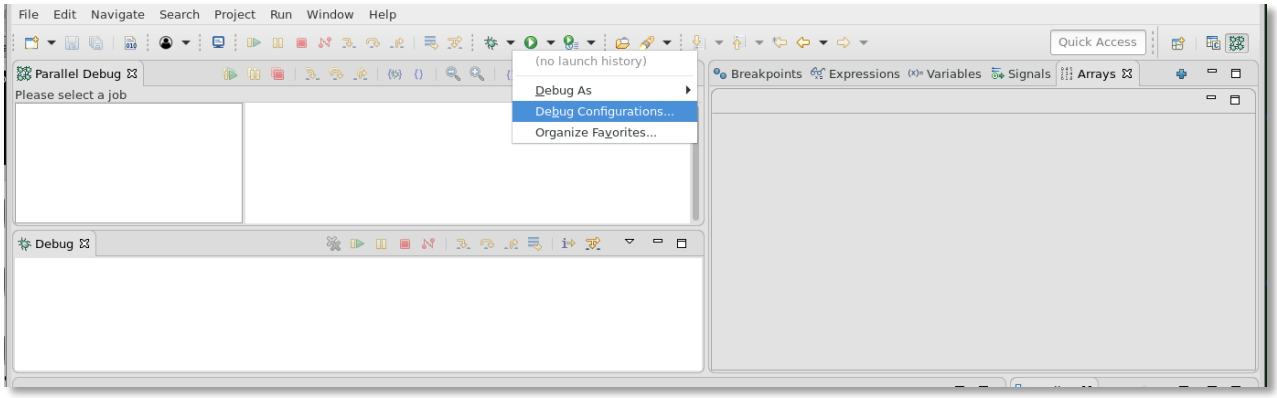
1. Select “Window > Perspective > Open Perspective > Other” in the Project Explorer view.



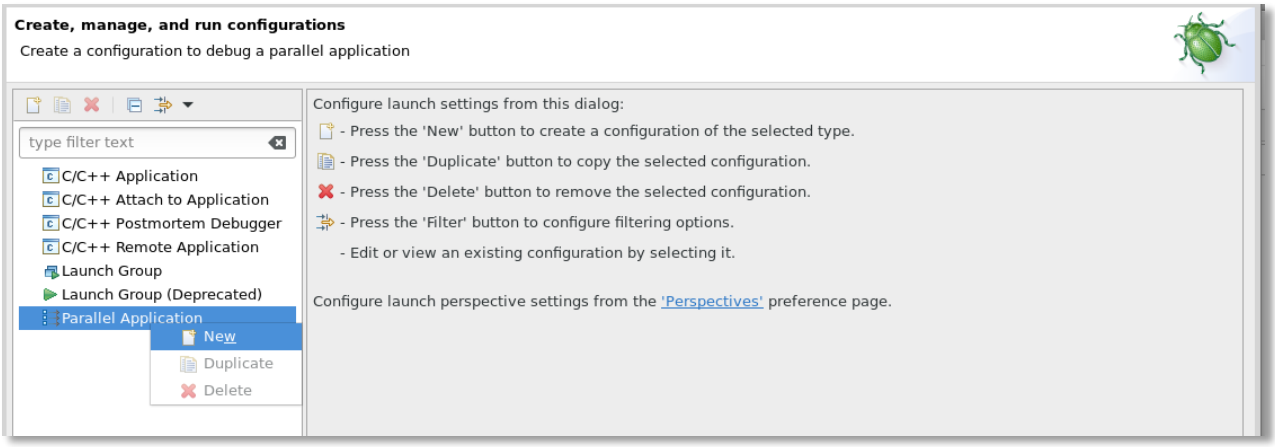
2. Select the Parallel Debug perspective.



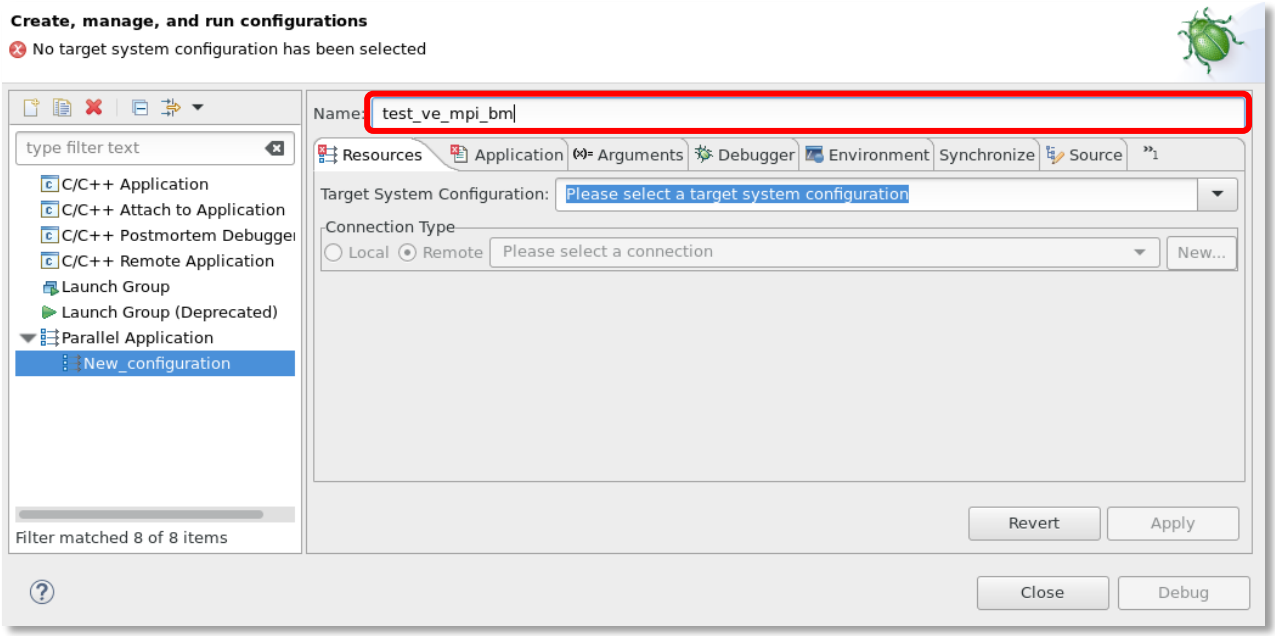
3. Select “Debug (bug icon) > Debug Configurations” to open the Debug Configurations window



4. Select “Parallel Application > New” on the window



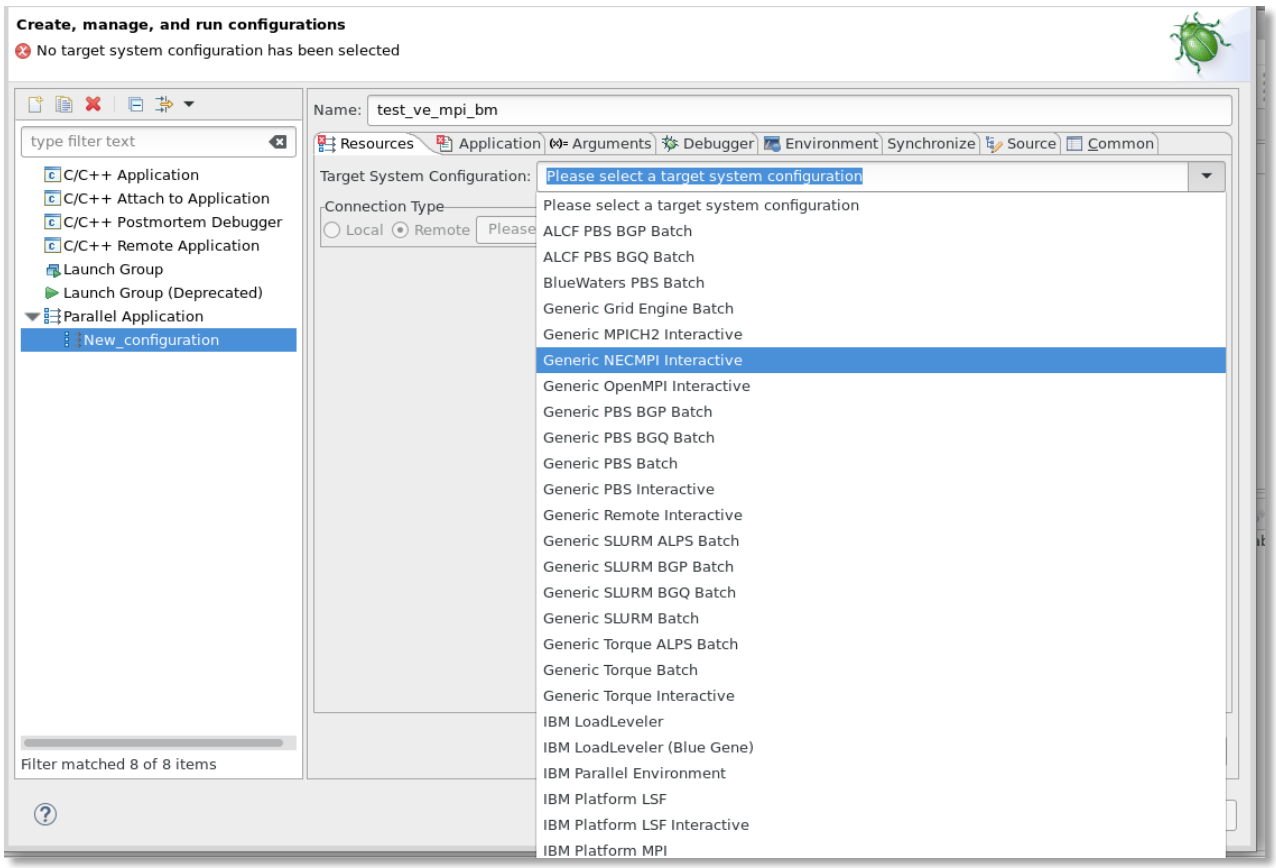
5. Input a debug configuration name in the Name field.



4.2 Setting of Resources

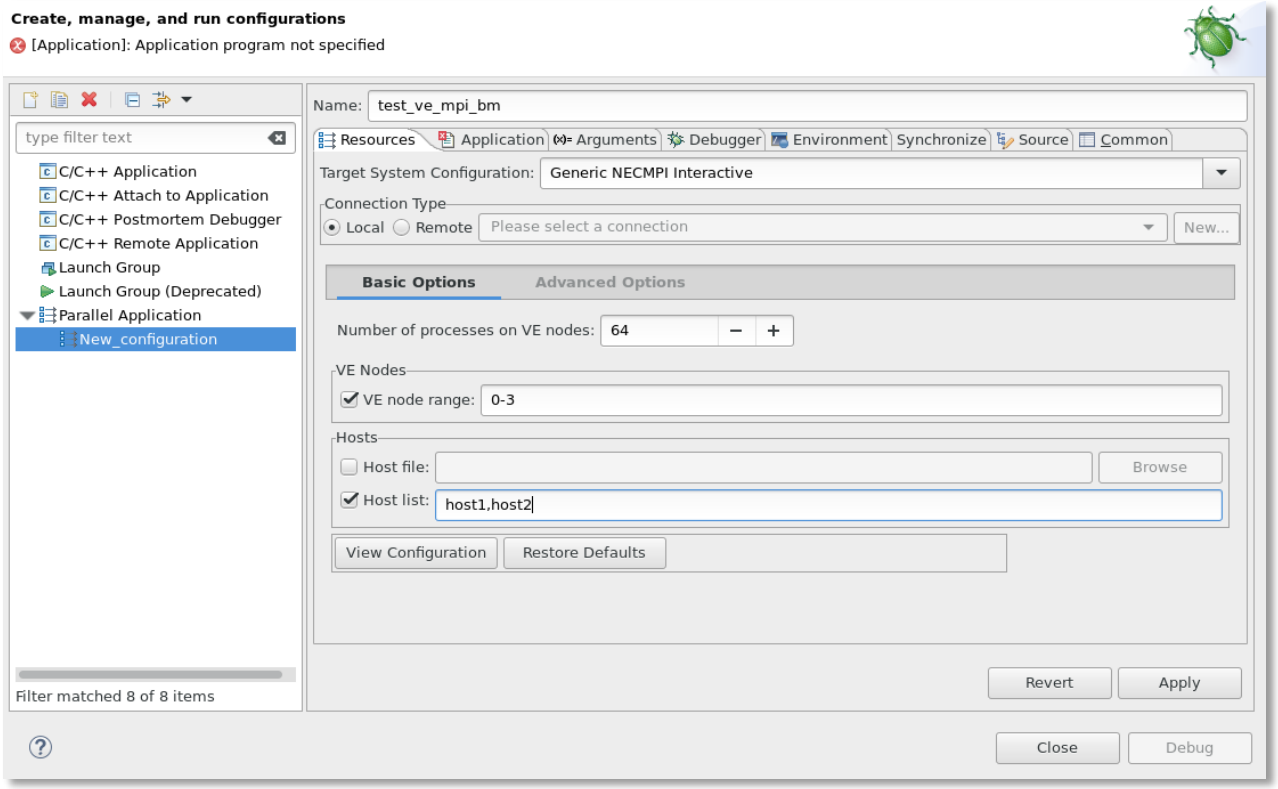
The following is the steps for configuring the settings required for execution of MPI applications such as runtime options.

1. Select “Generic NECMPI Interactive” in the Target System Configuration field on the Resources tab.



2. Select “Local” or “Remote” to specify where to launch an MPI application in the Connection Type field on the Resources tab. If user launches the MPI application on a remote VH, specify the VH information, too.

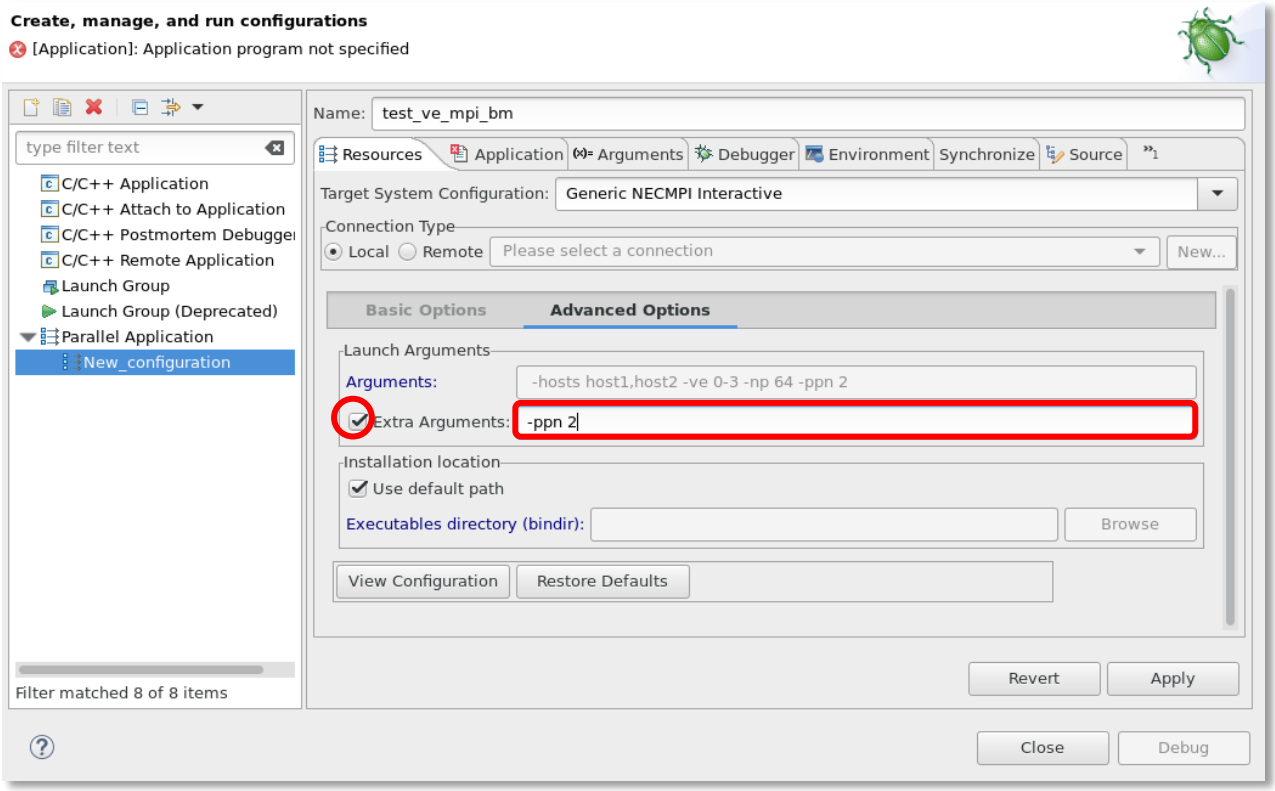
3. Specify the fields on the Basic Options tab to configure runtime options for the mpiexec command, which executes the MPI application.



The following table shows the runtime options available on the Basic Options tab.

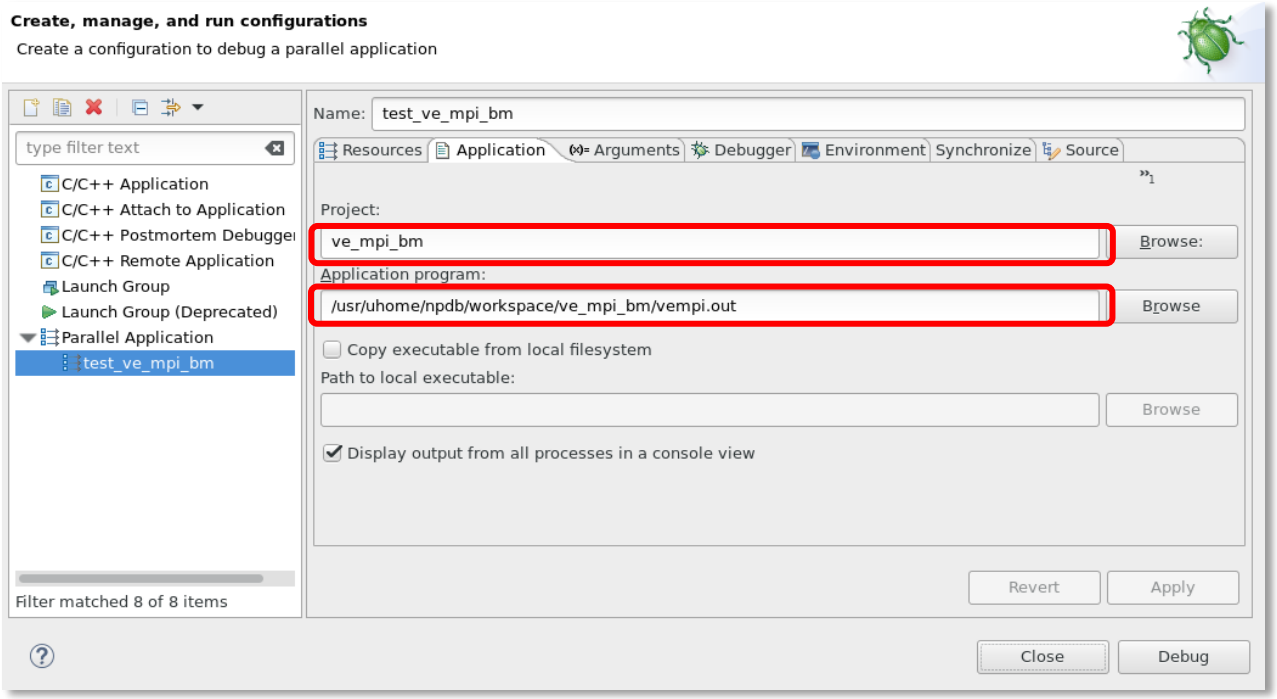
Option	Description
Number of processes on VE nodes	The total number of processes that execute an MPI application on VEs, which is used in the –np option.
VE node range	VE numbers on which an MPI application runs. Specify one VE number or a range of VE numbers, which is used in the –ve option. Without the specification, VE#0 is used. Examples of the specification: • One VE number : 1 (VE#1 is used) • Range of VE numbers: 0-7 (VE#0 through VE#7 are used)
Host file	File name that includes VHs from which an MPI application is launched. In the host file, VH names are descripted line by line. This is used in the -hostfile option. Without the specification of Host file and Host list, the local host is used.
Host list	Specify a comma-delimited list of VHs from which an MPI application is launched, which is used in the –hosts option. Without the specification of Host file and Host list, the local host is used. Example: host1,host2,host3

4. Click on “Advanced Options”, select the checkbox “Extra Arguments”, and input other options than those on the Basic Options tab.



4.3 Setting of an Application

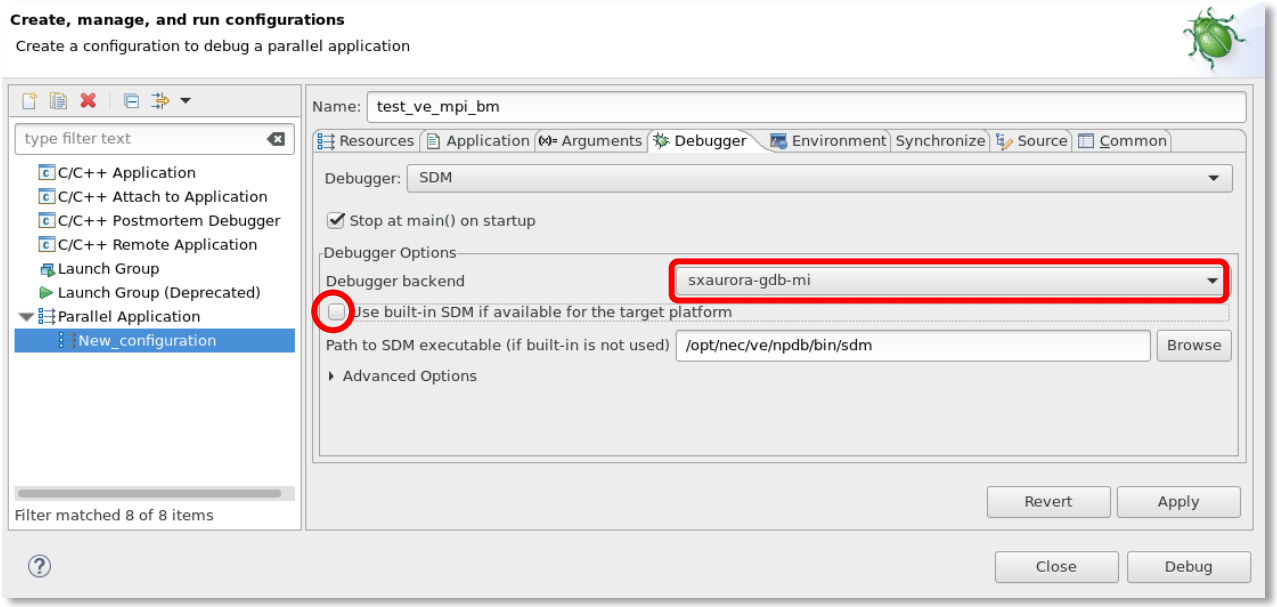
Specify a project name in the Project field and an executable file in the Application program field on the Application tab.



4.4 Setting of the Debugger

1. Select “sxaurora-gdb-mi” in the Debugger backend field on the Debugger tab.

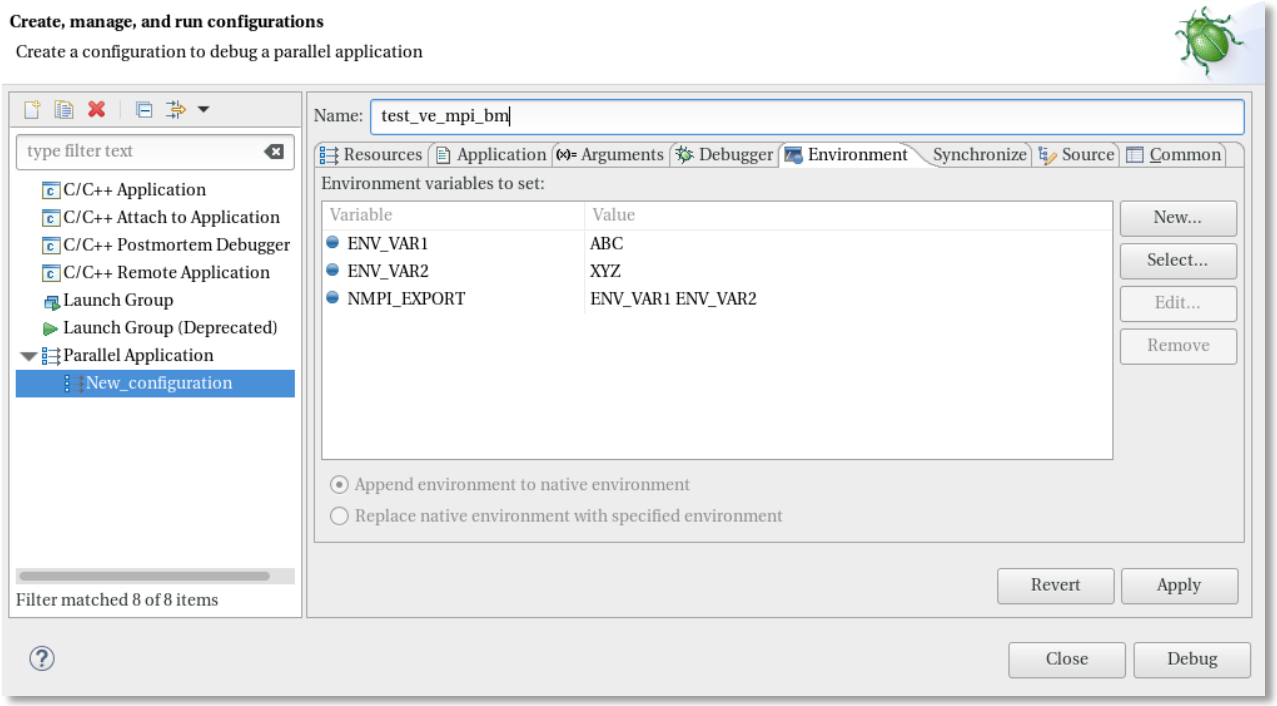
2. Uncheck the checkbox “Use built-in SDM if available for the target platform”.



4.5 Setting of the Environment

If necessary, add Environment variables setting on Environment tab. These environment variables set on this tab are passed to the host where mpiexec command is launched (set in Connection Type field in Resources tab). If you want to pass the environment variables to all hosts where MPI

application runs, please use NMPI_EXPORT environment variable supported by NEC MPI.



NEC Parallel Debugger supports the following environment variable.

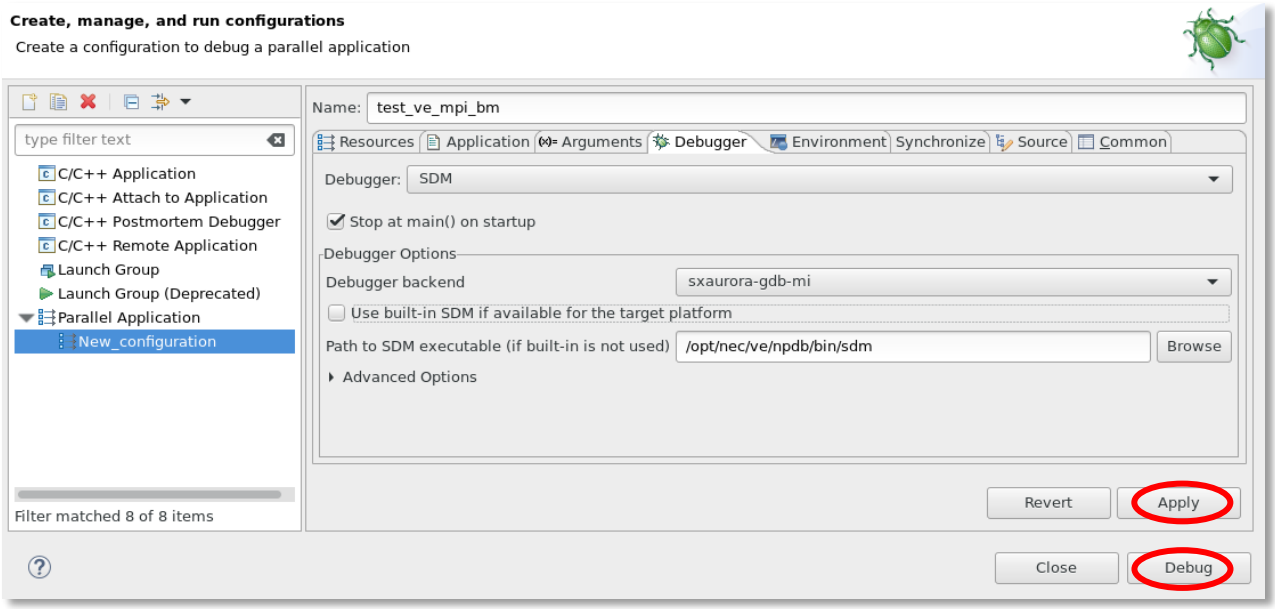
Environment Variable	Contents
NPDB_SDM_PORTRANGE	Range of the port number used for TCPIP connection among SDMs. Ports are separated by a colon. Default is 50000:50079.

4.6 Other Settings

If necessary, add other settings on other tabs such as the Arguments tab.

4.7 Starting of a Debug Execution

Click on the Apply button to save the settings, and then the Debug button to start a debug execution. Please make sure that VHs from which an MPI application is launched have been configured to load the NEC MPI setup script by means like login shell so that the execution of MPI applications is enabled.


Create, manage, and run configurations
Create a configuration to debug a parallel application
type filter text
C/C++ Application
C/C++ Attach to Application
C/C++ Postmortem Debugger
C/C++ Remote Application
Launch Group
Launch Group (Deprecated)
Parallel Application
New_configuration
Filter matched 8 of 8 items
Name: test_ve_mpi_bm
Resources
Application
Arguments
Debugger
Environment
Synchronize
Source
Common
Debugger: SDM
Stop at main() on startup
Debugger Options
Debugger backend
sxaurora-gdb-mi
Use built-in SDM if available for the target platform
Path to SDM executable (if built-in is not used)
/opt/nec/ve/npdb/bin/sdm
Browse
Advanced Options
Revert
Apply
Close
Debug

5 Operations for Debugging

This chapter explains major operations for debugging with Eclipse PTP. For details, please refer to the "Parallel Development User Guide" published on the Eclipse official site.

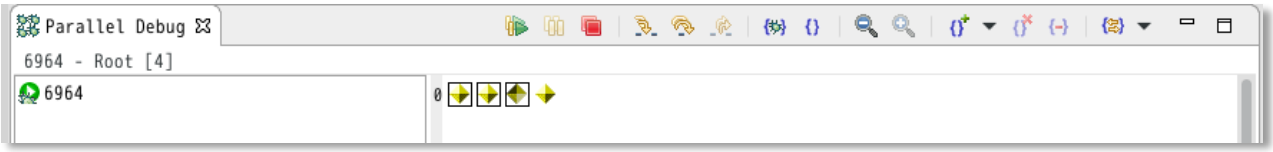
5.1 Available Views

The following table shows the views available for the applications run on VE that are displayed by default in the Parallel Debug perspective. The name of view is displayed on the top of tab. In Editor, the name of the opened file is displayed on the top of tab.

View	Description
Parallel Debug	Perform a debug execution of multiple processes in a collective manner
Debug	Perform a debug execution of a target process and display the stack trace of the process
Editor	Display source code files
Breakpoints	Display the list of breakpoints
Variables	Display arguments and local variables in the stack selected on the target process
Arrays	Display information about arrays
Expressions	Test data by inputting conditional expressions
Signals	Display the list of signals the target process receives and the corresponding actions by the debugger.
Outline	Display definitions of variables and interfaces of functions in the file displayed in the Editor view
Console	Display the standard output and standard error from an application
Problems	Display error messages at the building of a project
Error Log	Display warning and error messages from plugins

5.2 Debugging of Multiple Processes in a Collective Manner

The Parallel Debug view enables debugging of multiple processes in a collective manner.
Firstly, please select a set of processes targeted for debugging in the Parallel Debug view, and then a command to execute. If you do not select the set of processes, all the processes that are running (the root set) are selected by default.



The following table shows the available commands.

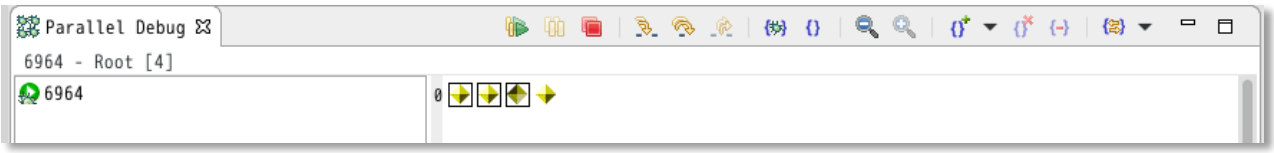
Icon	Command	Description
	Resume Group	Resume execution of the target processes.
	Suspend Group	Suspend execution of the target processes.
	Step Into Group	Execute the current line of the source code on the target processes and suspend the execution at the beginning of the next line. If the current line includes reference of a procedure, suspend at the beginning of the first line of the procedure.
	Step Over Group	Execute the current line of the source code on the target processes and suspend the execution at the beginning of the next line. If the current line includes reference of a procedure, suspend at the beginning of the next line after the execution of the procedure.
	Step Return Group	Resume execution of the procedure on the target processes from the line where the execution has been suspended, and suspend at the beginning of the next line of the line that invoked the procedure.
	Terminate Group	Terminate a debug session of the target processes.

5.3 Debugging of One Process

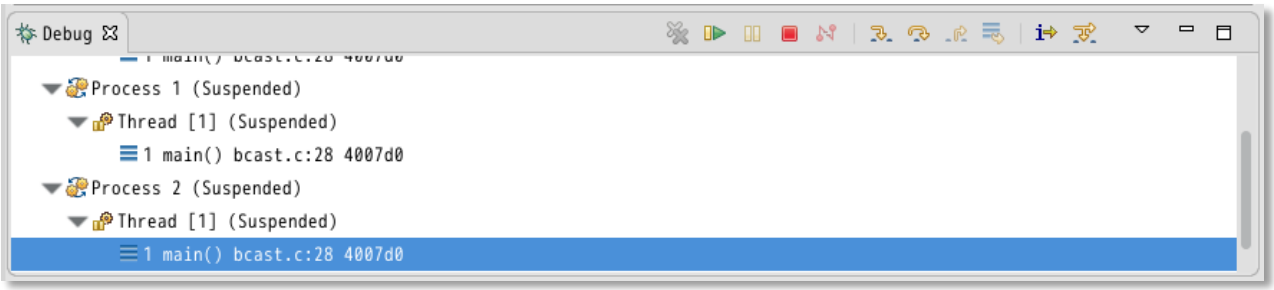
The Debug view enables debugging of a selected process. Firstly, please select a processes targeted for debugging in the Debug view, and then a command to execute.

The following is the detailed steps.

1. Double-click on a process in the Parallel Debug view to select the target of debugging. The stack trace of the selected process is added into the Debug view.
   - A rhombus stands for a process. The rank of the process is displayed by placing the cursor over the rhombus.
   - The rhombus corresponding to the process added into the Debug view is surrounded by a black border.



2. Select a stack in the Debug view.



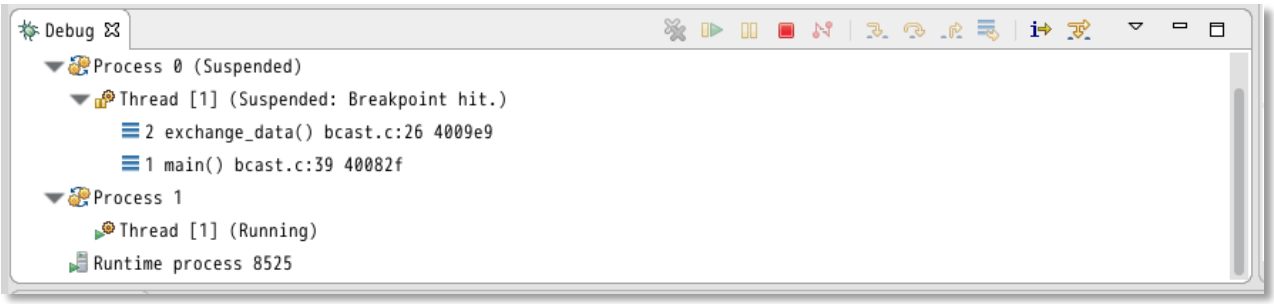
3. Select a command to execute.

The following table shows the available commands.

Icon	Command	Description
	Resume	Resume execution of the target process.
	Suspend	Suspend execution of the target process.
	Step Into	Execute the current line of the source code on the target process and suspend the execution at the beginning of the next line. If the current line includes reference of a procedure, suspend at the beginning of the first line of the procedure.
	Step Over	Execute the current line of the source code on the target process and suspend the execution at the beginning of the next line. If the current line includes reference of a procedure, suspend at the beginning of the next line after the execution of the procedure.
	Step Return	Resume execution of the procedure on the target process from the line where the execution has been suspended, and suspend at the beginning of the next line of the line that invoked the procedure.
	Terminate	Terminate a debug session of the target process.
	Remove All Terminated Launches	Delete the terminated debug session from the Debug view.

5.4 Display of the Stack Trace

The Debug view displays the stack trace of the target process.

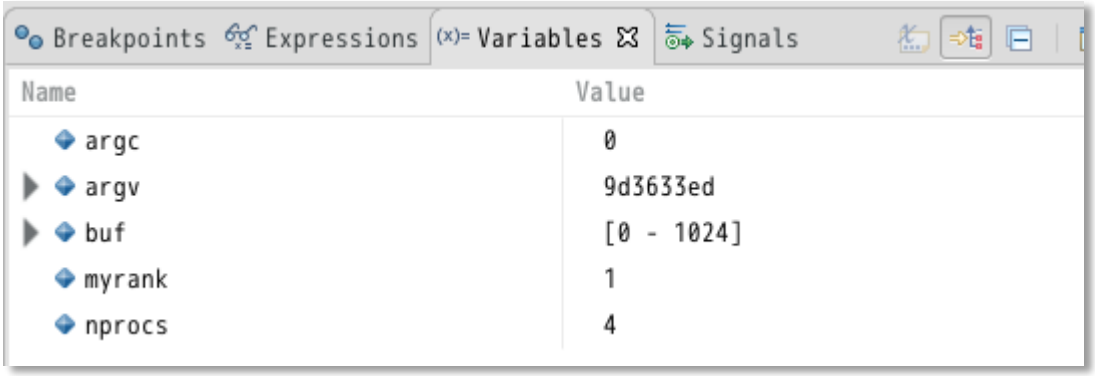


The displayed information above is as follows:

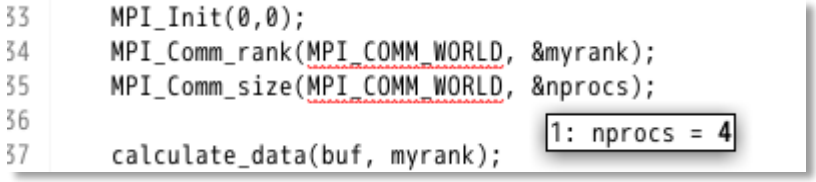
```
Session Name
  Rank of the MPI process
      Thread [Thread number] (Execution status: The reason for suspension, if suspended)
         Stack number   Procedure name   File name: Line number   Address
         Stack number   Procedure name   File name: Line number   Address
             :
```

5.5 Reference of Variable Information

Select a target process in the Parallel Debug view or Debug view, and then a stack of the process in the Debug view. The Variables view displays arguments and local variables in the selected stack.

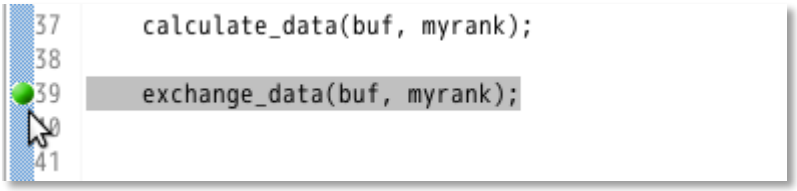


Information about a variable including external one is displayed on the source code in the Edit view by placing the cursor over the variable.

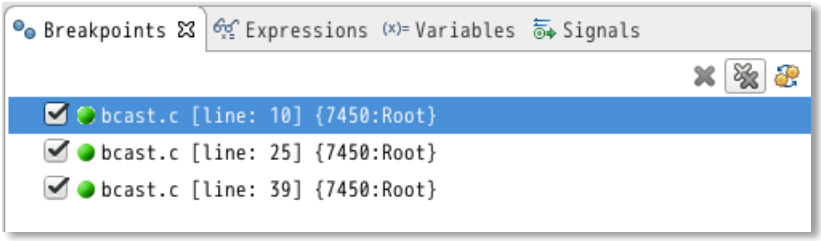


5.6 Setting of Breakpoints

A breakpoint is enabled or disabled by double-clicking on the left of a source line number in the Edit view. The breakpoint is set for all processes by default.

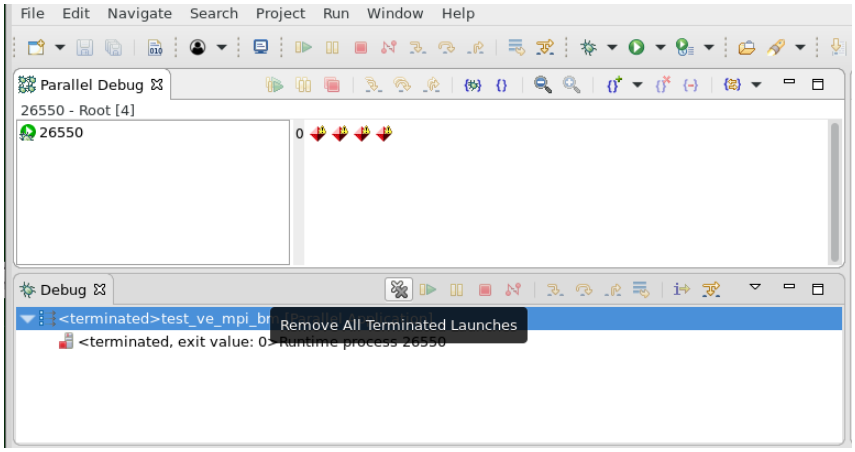


The Breakpoints view displays the list of enabled breakpoints.

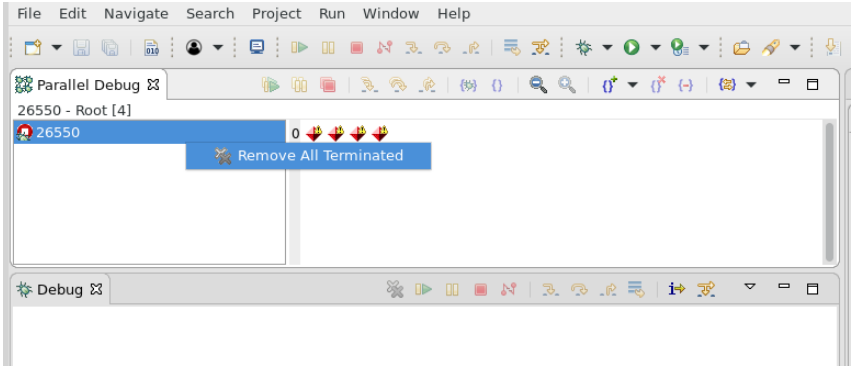


5.7 Termination of a Debug Session

To terminate a debug session, click on the  button in the Debug view to terminate the target processes, and then click on the  button to delete the debug session.



Next, click on the  button in the Parallel Debug view to terminate the processes not targeted for the debug, and then right-click on the debugging job to delete the debug session.


File Edit Navigate Search Project Run Window Help
Parallel Debug
26550 - Root [4]
26550
0
Remove All Terminated
Debug

6 Notices and Restrictions

6.1 Notices

- Please use the Oxygen v3 version of Eclipse PTP as NEC Parallel Debugger supports only the version.

- The following message would be displayed in the Console view at the beginning of debugging. Please ignore it as there is no effect on debugging operations. (This message appears when gdb attaches to the mpiexec command at the beginning of debugging)

```
Missing separate debuginfos, use: debuginfo-install glibc-2.17-157.el7.x86_64
```

- The watch point feature and trace point feature are not supported.

6.2 Miscellaneous

- It is also possible to debug non-MPI applications for VE, using the CDT plugin or Fortran plugin for Eclipse. Please specify gdb for VE (/opt/nec/ve/bin/gdb) as a debugger on the Debugger tab at the creation of a Debug Configuration. For details, please refer to the following documents published on the Eclipse official site.

  - C/C++ Development User Guide
  - Fortran Development User Guide

Appendix A: Change Log

Edition	Issue	Category	Modified Item	Chapter/Section
1	May　2018			
2	Dec　2018	DELETE	Restriction for referring variables.	6.2
3	May　2019	UPDATE	Additional description	1.2