Parallel Tools Platform for Judge

Carsten Karbach, Forschungszentrum Jülich GmbH

September 20, 2013

Abstract

The Parallel Tools Platform (PTP) represents a development environment for parallel applications. It supports programming languages C/C++ and Fortran. Its main components are system monitoring, code analysis, a parallel debugger for OpenMPI and simple integration of external tools such as profilers. This is a step by step tutorial on the usage of PTP for the JSC supercomputers. It demonstrates how to set up a simple parallel application, how to submit it to the target system and how to monitor the application afterwards.

Contents

1	Overview	2
2	Installation 2.1 Update PTP	2 3
3	Create a synchronized project	3
4	Application development	7
5	Job submission	8
6	System Monitoring	11
7	Stand-alone client	14
8	Trouble-Shooting8.1PTP does not connect to remote system8.2Symbols are not resolved	14 14 15
9	Contact	16

1 Overview

The Parallel Tools Platform(PTP) aims to provide a highly integrated environment specifically designed for parallel application development. It is based on Eclipse. You can download the latest PTP version from http://www.eclipse. org/downloads/. The PTP Eclipse distribution is called *Eclipse for Parallel Application Developers*.



Figure 1: Download PTP

A detailed generic tutorial about PTP can be found at http://wiki.eclipse. org/PTP/tutorials. More information about various PTP related topics are listed at http://wiki.eclipse.org/PTP.

2 Installation

PTP requires a Java Runtime Environment of version 1.6 or higher to be installed. When downloading PTP make sure to select your operating system (Linux, MAC, Windows). Download the eclipse distribution bundle. Unpack the bundle into your preferred installation directory (e.g. /bin). This will create a directory named *eclipse*. Inside this directory you will find the binary called *eclipse*. Start Eclipse by executing that binary. Users within the JSC workstation group can make use of a pre-installed PTP version. They can simply call /usr/local/zam/eclipse-parallel/eclipse to start PTP. The currently installed version in the JSC workstation group is Juno SR2.

After starting PTP choose a workspace directory, in which all your project files will be stored (see figure 2).

Select a workspace	
Eclipse stores your projects in a folder called a workspace. Choose a workspace folder to use for this session.	
Workspace: //private/Programme/PTPTests/JSCTut/workspace	B rowse
$\hfill \Box$ Use this as the default and do not ask again	D
	OK Cancel

Figure 2: Choose workspace

Go to the workbench by clicking on the corresponding icon. The empty workspace is shown in figure 3.



Figure 3: Empty workspace

2.1 Update PTP

You can update the downloaded PTP version with latest bug fixes or new implemented features. If you are experiencing errors at any step of this documentation, update your PTP version first, because the update could already fix your error. In order to update PTP, start Eclipse and click on the *Help* menu. Select *Install new Software*. Add a new update site with the URL http://download. eclipse.org/tools/ptp/builds/juno/nightly like shown in figure 4. If you have downloaded a *Kepler* version of Eclipse, replace *juno* with *kepler* in the mentioned URL.

Follow the instructions in the wizard to apply the update. This update site usually provides new updates immediately after each change of a developer.

3 Create a synchronized project

At first a project has to be configured, which is copied or synchronized to your target supercomputer. The project contains all source files as well as input data and a Makefile. PTP provides so called *synchronized projects*. They are stored locally and can be automatically synchronized with a directory on your remote system (the supercomputer you are working on). Afterwards, the compilation and submission of your application are conducted directly on the remote system. Figure 5 shows how to create a new synchronized project. You can create both a synchronized C/C++ or Fortran project. For further information on parallel Fortran development have a look at http://eclipse.org/photran/.

In the following screen you have to choose the project name. Then you can configure a new connection to your target system by clicking on the *New* button,



Figure 4: Update PTP with latest version

ی و			C/C++ - Eclipse			\sim \sim \times
Eile Edit Source Refactor	<u>Navigate Search P</u>	roject <u>R</u> un <u>W</u> indow <u>H</u> elp			1991	
	% · • · O ·	× (H) (H) (G) • 63	· © · ⊙ · ⊗ · O · Ø ·	4 109 G / - 19		
					Q Quick Access	System Monitoring
Project Explorer 🛿					- 0	🗄 Ou 🛱 🖲 Ma 🗐 Ta 📮 🗖
	🖹 🛸 🖗 🖻					₽ ▼
						An outline is not available.
	Alexe		- 0 D - 1 - 1			
	Show In	ChiRa Aha)A/	Project	_		
	Silo <u>w</u> III	Shill Party P	C C Project			
	B Copy Qualified Nan		Fortran Project			
	💼 Paste	Ctrl+V	Tracing Project			
	X Delete		🕰 Remote C/C++ Project			
	import		Synchronized C/C++ Project			
	Contraction of the second seco		Synchronized Fortran Project			
	C Rejresn	rb	Example			
	Resource Conligura	ations	Ct Ct	1+N		
		🖹 Problems 🛛 Tasks	Console 🕱 🔲 Properties 🛲 C	all Graph 🌵 GEM Brows	s y GEM Analyz 🧕 GEN	I Consol 🚯 Remote Envi 🦳 🗖
		No consoles to display at	this time			생 암 및 · 🖸 •

Figure 5: Create a synchronized project



which is depicted in figure 6. An example connection configuration for Judge is shown in figure 7.

Figure 6: Configure synchronized project

After configuring your connection you can select the directory on your target system, which should be synchronized with your local source files. Choose this directory by clicking on the *Browse* button. If a *Connection Error* pops up, Eclipse is not able to connect to your target system. Try to create a new connection and double check your configuration.

Finally, the project type has to be configured. For now, just select an empty Makefile project and choose *Linux GCC* as tool chains as shown in figure 8. The toolchains determine, which compilers are used to discover include and library directories. If you have other compilers installed locally such as cygwin, you can also select the corresponding toolchain. You can still decide later in your Makefile, which compiler is used to build your application. Click on the *Finish* button to complete the project creation.

Generic Remote Host	
Properties for connecting to a generic host	
Target name: Judge	-
-Host Information-	
O Localhost ⊙ Remote host	
Host: judge.fz-juelich.de	
User: karbach	
O Password based authentication	
Password:	
● Public key based authentication	
File with private key: /home/zam/karbach/.ssh/id_dsa Browse	
Passphrase:	
Advanced	
\Box	
? <u>Einish</u> Cancel	

Figure 7: Create new connection to Judge

Remote directory: //homeb/zam/karbach/hello		Browse
Project Type ♦ Shared Library (Intel(R) Fortran) ♦ Executable (IBM XL Fortran)	Remote Toolchain Intra Centreley of C Linux CCC	-
 	MacOSX Berkeley UPC MacOSX GCC Local Toolchain	
 Empty Project - Fortran Demo - Hello World - Fortran Demo - Hello World - Fortran using MPI Demo - Calculate Pi - Fortran using MPI 	Linux GCC MacOSX Berkeley UPC MacOSX CCC	
Show project types and toolchains only if they are supported on the p Modify File Filtering	platform	
0	<back next=""> Einish</back>	Cancel

Figure 8: Project type configuration

The synchronization is handled by git in the background. As long as the synchronization of your project works fine you do not have to care about git. Note, that you should avoid changing the same files on both your local system and your remote system at the same time. A following synchronization will probably fail, since git will try to merge both versions. Therefore, you have to exclude especially binary files from synchronization, because binary files are compiled on both systems, which would lead to merge conflicts. To exclude files fom synchronization right-click on your project and select *Synchronization* and click on *Filter* in the sub-menu. The dialog shown in figure 9 shows up.

File View		
D corroiect		
▶ □ 😓 .ptp-sync		
▶ □ 🗁 .settings		
HelloWorld		
🗹 🗎 HelloWorld.cpp		
🔲 🗎 Makefile		
🗹 📄 ptp_job.e489869		
🔲 📄 ptp_job.o489869		
🗹 🗎 script		
Show remote files		
Pattern View		
Exclude regex: .*\.o.*		Up
Exclude path: HelloWorld		Down
Exclude path: Makefile		
Exclude path: .ptp-sync		Edit
Exclude path: .settings		
Exclude path: .cproject		
Exclude path: .project		- 1
		Remove
Enter Path:	Exclude	Include

Figure 9: Filtering of synchronized files

Here you can remove files from synchronization by unchecking the corresponding checkbox in the *File View*. Moreover, you can use regular expressions for advanced filtering rules. E.g. the regular expression ".*\.o.*" matches with all files containing the string ".o". Regular expressions are added by inserting them into the text field labeled with *Enter Regex* and choosing one of the buttons *Exclude* or *Include*. Further information on synchronized projects and in particular on merge conflicts is documented at the URL http://help.eclipse.org/juno. Search for *synchronized project* in the documentation's search. You can also access this documentation directly from Eclipse (Help \Rightarrow Help Contents).

4 Application development

Your workspace should now show your empty synchronized project (see figure 10).

Add your source files and a Makefile for compiling your application to the project. This can be done by either creating these files with the Eclipse ed-



Figure 10: Workspace with synchronized project

itors using for instance the menu $File \rightarrow New \rightarrow Source \ File$. Alternatively, you can import existing source files via $File \rightarrow Import$. Expand in the import dialog the *General* folder and select *File System*. By default the included files are copied into your workspace. It is also possible to create links to the original files. Afterwards, your project could for instance look like shown in figure 11. A simple MPI program in *HelloWorld.cpp* and a Makefile are added. Each time you add or change a file within your synchronized project, these changes are automatically transferred to the corresponding directory of your remote system. If Eclipse is showing errors in your code due to unresolved symbols, try to configure the include paths of your project as described at http://help.eclipse.org/juno/index.jsp?topic=/org.eclipse.cdt.doc.user/tasks/cdt_t_proj_paths.htm. See also section 8.2 for more help on resolving symbols.

You can compile your application by clicking on the *hammer icon* (see figure 12). If you choose *Default_remote* the application is compiled on the remote machine, while *Default_local* compiles your application on your local machine, where you have started the Eclipse client on.

The console view on the bottom (see figure 13) displays the results of the build. It will also show possible compilation errors and link these errors with the corresponding lines of your code.

5 Job submission

Once you have built your application PTP allows for submitting a job to various batch systems. To run your application press on the small triangle next to the



Figure 11: Application development

-	•	010	Ö	-	Ċ	•	6
- <		<u>1</u> D	efault	_loc	al		1
	4	' <u>2</u> D	efault	_rer	note		

Figure 12: Build your application



Figure 13: Build command results

play button in the top toolbar (see figure 14). Click on *Run Configurations*. Start the job submission configuration by double clicking on *Parallel Application* (see figure 15).

For each of the JSC systems Judge, Juropa and Juqueen there is a special *Target* System Configuration. This configuration specifies the interface used by PTP to interact with the target system. It defines configuration parameters for job submissions, the submission command such as *msub* for Moab or *llsubmit* for LoadLeveler or it configures how to retrieve a list of available waiting queues. For Judge choose the Target System Configuration *de.fz-juelich.judge.torque.batch*. As soon as you have selected the previously configured connection, a set of

<u>N</u> aviş	gate	Seg	irch	Pro	ject	<u>R</u> u	1 <u>R</u> u	n !	<u>W</u> in	dow	1	Hel	р										
	100	Ċ	•	<u>63</u> .	•	ð •	6	-	\$	- (D	•	8	·	Q.	•		8	Ö	•	0	ø	A
													<u>R</u> ur	n As	s				•				
3		Llo	lloV	Vorla	lon	n 19							Ruj	n Ci	onfig	jura	itio	ns					1.40
∇		пе	1101	VOIIC	i.cp	pp w							Org	jani	ze F	ay	onit	es				Le	Ivia
			* #in #in	Hell clud clud	oWo e '	mpi.	cpp[. h"					_								ľ			al
			ısi	ng n	ame	espac	e st	d;															He

Figure 14: Start job submission

8 No target system configuration	has been selected	D
🗋 🗈 🗙 🗌 🚍 🔹	Name: HelloWorld	
type filter text 🛛 🔏	😫 Resources 🛯 Application 🕬= Arguments 🚾 Environment Synchronize 🔲 Common	
C/C++ Application	Target System Configuration: Please select a target system configuration	_ [▼
Fortran Local Application Java Applet	Connection name: Please select a connection (* 1	lew
Java Application		
🕨 Launch Group		
✓ I Parallel Application I HelloWork		
Filter matched 7 of 7 items	Using Parallel Application Launcher - <u>Select other</u> Apply Rega	rt
?	Eun	se

Figure 15: Start new job submission configuration

configuration parameters for your job submission are shown (see figure 16). All of these parameters affect the batch script, which is sent to the scheduler of Judge. You can always check the currently generated script by clicking on the *View Script* button. Many of the parameters, which you can configure in this dialog are also explained here: http://www.fz-juelich.de/ias/jsc/EN/Expertise/Supercomputers/JUDGE/Userinfo/Quick_Introduction.html.

Examples for the most important parameters are shown in figure 16. You can choose the number of nodes, a wallclock limit, the number of processes per node as well as the number of threads started for each MPI task and the number of MPI tasks itself. The meaning of each parameter is explained in the *Description* column. Special parameters for Judge are the number of GPUs per Node and the GPU type. This allows for allocating GPUs to your job.

These configurations lead to the batch script shown in figure 17.

Moreover, the compiled binary has to be chosen in the *Application* tab (see figure 18). Click on the lower *Browse* button to select the binary located at Judge.

Now the *Run* button is enabled and you can submit your batch script by clicking on it. PTP uses *msub* to submit the batch script to the scheduler of Judge. You will be asked, if you would like to switch to the system monitoring perspective.

Create, manage, and run config	urations
Create a configuration to launch a pa	allel application
	Name: India
	Rane, Jieno
	Tarast Sustam Casfiguration: de fa incliné indee terrore bateb
Cit C/C++ Application	raiget System Comiguration. de.nz-Juenen juuget orquet vatem
Eclipse Application	Connection name: judge New
Fortran Local Application	
💌 Java Applet	Basic Settings Advanced Settings Import MSUB Script
🗊 Java Application	Name Value Description
Ju JUnit	Job Name: ptp job The name assigned to the job by the gsub or galter comm
Jij JUnit Plug-in Test	
Launch Group A OSGi Framowork	Account: Account to which to charge this job.
	Queue: Queue: Designation of the queue to which to submit the job.
i⊒ hello	
🍗 Perl CGI	Number of nodes: 1 Number and/or type of nodes to be reserved for exclusive t
🐆 Peri Local	Total Memory Needed: Maximum amount of memory used by all concurrent proce
	Wallclock Time: 00:05:00 Maximum amount of real time during which the job can be
	MPI Command: Which mpi command to use.
	Processes per Node: 8 Processes per node (ppn value)
	GPUs per Node: 0 BPUs per node
	GPU type: GPU architecture
	Threads per Task: 2 Number of threads per task (tpt value)
	MPI Number of Tasks: 4 The '-np' value (Should match with nodes*ppn/tpt = 1*8/2)
	Export Environment:
	View Script View Configuration Restore Defaults
Filter matched 14 of 14 items	Using Parallel Application Launcher - <u>Select other</u> Apply Reget
?	Bun Close

Figure 16: Run configuration for Judge

Click on *yes* in order to monitor the queue status of your job and to obtain its output.

6 System Monitoring

At first, your job will be displayed in the *Inactive Jobs* view as submitted job. As soon as it is dispatched by the scheduler it will be moved to the *Active Jobs* view. After its completion the job switches back to the *Inactive Jobs* view. Here you can check the output of your application via the pop-up menu, which is shown on a right click on your job entry (see figure 19). The Nodes View on the right displays the compute resources of Judge. Each of the small rectangles represents a core. The cores are grouped in larger rectangles, which depict the compute nodes. The two separated cores on the bottom of each node show the status of the node's GPUs. Colored rectangles are allocated to a running job. Therefore, the Nodes View shows the mapping of all currently running jobs to the system's compute resources.

A more detailed documentation of all functions of the system monitoring perspective can be found at http://help.eclipse.org/juno/index.jsp. Select

0	Script with current values
Script with current values	
<pre>#/bin/bash #MSUB -N ptp_job #MSUB -N ptp_job MSUB - I nodes=1:ppn=8:gpus=0: #MSUB - V tpt=2 #MSUB -V MPI_ARGS="-np 4" if ["-np" == "\${MPI_ARGS}"]; then MPI_ARGS= fi COMMAND=mpiexec if [-n "\${COMMAND}"]; then COMMAND="\${COMMAND} \${MPI_ARGS} /homeb/zam/karbach/hello. else COMMAND="/homeb/zam/karbach/hello/hello " fi cd /homeb/zam/karbach \${COMMAND}}</pre>	/hello "

Figure 17: Generated batch script

Create, manage, and run configue Create a configuration to launch a paral	ations el application	
Type filter text C/C++ Application Fortran Local Application Java Applet	Name Helloword Image: Best and the second sec	<u>B</u> rowse:
Java Application ▶ Launch Group ▼ ∰ Parallel Application B∰ HelloWorld	Thomab/zam/katachHelleWorld2HelleWorld Copy executable from local filesystem Path to local executable: Display output from all processes in a console wew	Browse
Filter matched 7 of 7 items	Using Parallel Application Launcher - <u>Select other</u>	Reyert

Figure 18: Application tab of run configuration

Parallel Tools Platform (PTP) User Guide \rightarrow Monitoring Jobs and Systems in order to navigate to the documentation.

🍃 💿 Sr	stem Monitoring - HelloWorld/HelloWorld.cpp - Eclipse		\odot \otimes \otimes
<u>File Edit Source Refactor Navigate Search Project Run Window Hel</u>	P		
		Q Quick Access	C/C++ System Monitoring
🛠 Monitors 🛛 🕨 📕 😵 🗍 🕂 🗶 📟 🗖	system: judge 🖾		(+ + = = =
Status Connection Name System Type			
S Judge TORQUE Resource Manager			
🔚 Active Jobs 🔠 Inactive Jobs 🕱 📃 🗖		1001001 000000 000000 000000	00000 0000000 000000
step owr queue wall queue dispat totalcc stat			
465154.judgem karbac ? ? ? ? CO			
Resource Configurations	<u> Diregnes (concorr inconcor ionconco inco</u>		
Resume Job Gancel Job			
O Hold Job			
Release Job			
Get Joh Error			
C Get Job Output			
😂 Refresh Job Status		non accouct ponone accouct	0000 000000 0000
Remove Job Entry			
🗆 Messages 🛢 Console 🕱 🐁 Remote Environments 🛛 🗖 🗖			
🗟 (🕞 🖬 (🖻 = 📑 -	<u> </u>		
/homeb/zam/karbach/HelloWorld2/ptp_job.o467005 My rank is 0			
We are 4 tasks My rank is 1			
My rank is 2 My rank is 3			
	<u> 00 00 00 00 00 00 00 00 00 00</u>		0 0 0
			Þ

Figure 19: System monitoring for Judge

7 Stand-alone client

If you are only interested in the monitoring functionality of PTP you can use the following tools:

- LLview: graphical monitoring of batch system controlled cluster, available at http://www.fz-juelich.de/ias/jsc/EN/Expertise/Support/ Software/LLview/_node.html.
- SysMon: Stand-alone system monitoring client based on the system monitoring perspective of PTP, available at http://download.eclipse.org/ tools/ptp/builds/kepler/nightly/index.html

While SysMon is directly derived from the PTP IDE presented above, LLview is a monitoring application especially and only designed for system monitoring. Both applications look and work similar, since a large part of LLview is also contributed to PTP's monitoring system. However, LLview still provides a larger set of statistical diagrams evaluating the load and usage of the particular supercomputer.

8 Trouble-Shooting

This section documents possible problems experienced by PTP users. Each problem is outlined and possible solutions are documented.

8.1 PTP does not connect to remote system

The connection to the target supercomputer cannot be established. This problem will occur as soon as you are trying to invoke any commands remotely. E.g. the remote file browser is trying to start for choosing the location for your synchronized project. Or you are starting a monitoring connection. Depending on your configuration you might get the error dialog shown in figure 20.



Figure 20: Error dialog when trying to connect to the target supercomputer

Solution

Try the following approaches to get your connection working:

1. When creating your connection (see figure 7) click on the *Advanced* button and enable the checkbox for *Use login shell*. This tells PTP to use the user's login shell when running remote commands. This approach might also work vice versa. If the checkbox was selected by default, when you configured your connection, try to create a new connection with the login shell checkbox deactivated.

2. If the previous approach is not working, check if you have additional commands in your .profile or .bashrc file of your remote system, which are printing to STDERR/STDOUT. You have to remove these commands for the PTP connection to work correctly. In the same manner PTP will not be able to connect, if these scripts are creating a new shell on start-up for instance by calling the bash command.

8.2 Symbols are not resolved

Symbols, which are available in the include files of your remote system, are not resolved. E.g. all MPI functions are marked with error signs saying that the corresponding "Symbol 'symbolname' could not be resolved".

Solution

The most reliable solution is to copy the MPI include files to your local machine. Afterwards, add the local path to your include paths in the project properties as shown in figure 21.

type filter text 🛛 🕥	Paths and Symbols		↓ ↓ ↓
- Builders	Configuration: Default_re	note [Active]	e Configurations[
> C/C++ Build			
C/C++ General			
> Code Analysis	Bincludes # Symbols	Libraries 👼 Library Paths 😕 Source Location 🤔 Output Location 🗟 References	
- Documentation			
- File Types	Languages	Include directories	Add
- Formatter	 Assembly 	/private/PTPWF/openmpi-1.4.3/ompi/include	Edit
- Indexer	- GNU C		
- Language Mappings	GNU C++		Delete
Paths and Symbols			Export
- Preprocessor Include Paths, Ma			
- Profiling Categories			Move Up
XL C/C++ Language Options			Maus Down
Fortran Build			TALOAG DOMIL
Paths and Symbols			
Project References			
	<	P	$ \rightarrow $
?		С	Cancel

Figure 21: Add include path

You can find the MPI includes by calling the MPI compiler with your input file including MPI headers like this:

Listing 1: Find MPI include path

picc -E HelloWorld.cpp grep mpi.h

Copy the localized include directory to your local machine and add the local path to your include paths. Then rebuild the indexer by right-clicking on your project and choosing $Indexing \rightarrow Rebuild$. You can check, if your include files are found by Eclipse by selecting the #include statement and pressing F3, which will open the include file, if it is found. If it is not found, no file is opened. Then you would have to check your configured include paths again.

If that did not help, try to follow the instructions on http://help.eclipse. org/juno/index.jsp?topic=%2Forg.eclipse.ptp.doc.user%2Fhtml%2Fsync. html.

9 Contact

Your feedback on this tutorial and PTP is highly appreciated. For questions or suggestions directly concerning this tutorial you can write an e-mail to *c.karbach@fz-juelich.de*. For more general questions you can also use PTP's user mailing list available at http://dev.eclipse.org/mailman/listinfo/ptp-user. If you would like to get more information on PTP, the following links are helpful:

- Articles on PTP: http://wiki.eclipse.org/PTP/articles
- Mailing Lists:
 - Major announcements: http://dev.eclipse.org/mailman/listinfo/ ptp-announce
 - Developer discussions: http://dev.eclipse.org/mailman/listinfo/ ptp-dev

In order to file a bug you are experiencing by using PTP use https://bugs.eclipse.org/bugs/enter_bug.cgi?product=PTP.