



CrossGrid Developer Manual Guide

Performance Prediction Component

Task 2.3.2 - PPC

Document Filename:	CG-2.3.2-PPC-DeveloperManual-v1.0-USC.pdf
Workpackage:	Task 2.3.2 - PPC
Partner(s):	USC
Lead Partner:	USC
Config ID:	CG-2.3.2-PPC-DeveloperManual
Document classification:	PUBLIC

Abstract: This is the developer manual for the CrossGrid Performance Prediction Component, task WP 2.3.2.



Delivery Slip

	Name	Partner	Date	Signature
From	T.F. Pena & J.C. Cabaleiro	USC	Jan 2005	
Verified By				
Approved By				

Document Log

Version	Date	Summary of changes	Author
1-0-DRAFT	Jan 21st, 2005	First draft version	Pena/Cabaleiro

Contents

CopyrightNotice	4
1 Introduction	5
1.1 Abbreviations and Acronyms	5
1.2 References and Source Code	6
2 Implementation Structure	7
2.1 Product Use Cases	7
2.2 Product Component Model	7
2.3 Detailed Implementation Model	8
3 Contact Information and Credits	15
4 EDG License Agreement	16

Copyright Notice

Copyright (c) 2005 by *T. F. Pena, J. C. Cabaleiro, M. Boullón and F. F. Rivera*. All rights reserved.

Use of this product is subject to the terms and licenses stated in the EDG license agreement. Please refer to Chapter 5 of the Installation Guide for details.

JFreeChart and *JCommon* (C)opyright 2000-2004, *Object Refinery Limited and Contributors*.

JavaHelp is a registered trademark of *Sun Microsystems, Inc*. All rights reserved.

This research is partly funded by the European Commission IST-2001-32243 Project "CrossGrid".

1 Introduction

The PPC (Performance Prediction Component) is a tool to study the behavior of some selected MPI kernels in the Grid. It is based on a set of models that characterize the behavior of the kernels when they are executed in the Grid. These models were obtained from exhaustive executions on different Grid scenarios. From these measurements a methodology to obtain correlations between performance and monitoring information is applied to obtain easy to use functions to characterize the execution of the kernel. Even though the tool is limited to a set of specific kernels, it can be easily extended to others. Note that some of the kernels are general purpose. i.e. communications, that could be useful for any user.

The tool shows performance information based on predictions taken from the models. The way to show this information is interactive and easy to use. Also the tool can get monitoring information provided by JIMS to show the predictions under the present and real scenario, and then the user can interactively change these values to analyze their effects on the kernel.

Some of the main features of the tool are:

- PPC is an application dependent tool. It is specifically applied to selected kernels that were analyzed in terms of performance to obtain analytical models. Anyway, it can be also broadly used to study the performance of communication routines themselves.
- PPC is an interactive tool in the sense that the user can easily change the parameters that characterize the system or the problem, and then analyze their influence.
- PPC uses analytical models, and therefore the predictions are obtained fast. This feature is very important for the interactivity.
- PPC can be used to understand the behavior of the kernels under different Grid features, selected by the user. This is important in any system in which the features change dynamically, like Grids.
- PPC can be easily extended to any heterogeneous system and kernel.

One of the main contributions of PPC to the state of the art in Grid computing is the methodology to obtain the performance models, and to allow the user to understand the behavior of the kernels under hypothetical Grid features. Please refer to PPC User Manual for details.

1.1 Abbreviations and Acronyms

CE	Computing Element
FLOPs	Floating Point OPERations
GUI	Graphical User Interface
JIMS	JMX-based Infrastructure Monitoring System
JMX	Java Management Extensions
MD	CrossGrid Migrating Desktop
PPC	Performance Prediction Component
SE	Storage Element
UML	Unified Modeling Language

1.2 References and Source Code

PPC is distributed in a simple rpm:

```
cg-wp2.4.2-perfpred-1.3-2.noarch.rpm
```

This rpm can be found in:

[https:// savannah.fzk.de/distribution/crossgrid/autobuilt/i386-rh7.3-gcc3.2.2/wp2/RPMS/](https://savannah.fzk.de/distribution/crossgrid/autobuilt/i386-rh7.3-gcc3.2.2/wp2/RPMS/)

The installation guide is in:

<https://savannah.fzk.de/distribution/crossgrid/crossgrid/wp2/wp2.4-perf/wp2.4.2-perfpred-PPC/docs/CG-2.3.2-PPC-InstallGuide-v1.1-USC.pdf>

The user manual is in:

<https://savannah.fzk.de/distribution/crossgrid/crossgrid/wp2/wp2.4-perf/wp2.4.2-perfpred-PPC/docs/CG-2.3.2-PPC-UserManual-v1.0-USC.pdf>

The developer manual (this manual) is in:

<https://savannah.fzk.de/distribution/crossgrid/crossgrid/wp2/wp2.4-perf/wp2.4.2-perfpred-PPC/docs/CG-2.3.2-PPC-DeveloperManual-v1.0-USC.pdf>

2 Implementation Structure

2.1 Product Use Cases

Figure 2.1 shows the first use case. It involves the user who can obtain the grid parameters from the actor JIMS. JIMS is a monitoring tool from WP 3.3, that provides idle CPU time, latencies and bandwidths of any site, as well as other monitoring information no used by PPC. Then, the user can select the kernel and read its input data from the storage element or local disk. Finally, the results of the prediction built from the monitoring data and the application input data can be shown.

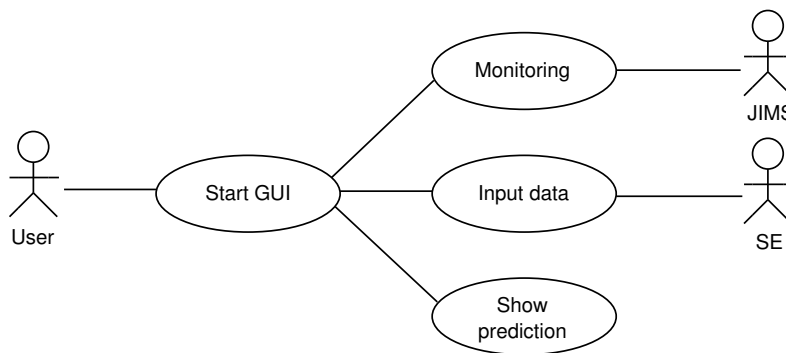


Figure 2.1: PPC Use Case involving JIMS

Figure 2.2 shows the use case in which the user can set the grid parameters by his own instead of reading them from JIMS. This situation is useful in order to study the performance of the application in several hypothetical situations. The rest of the procedure is the same as in the above case.

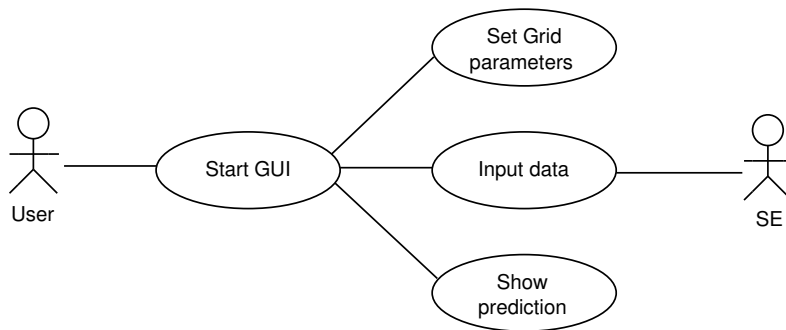


Figure 2.2: PPC Use Case setting grid parameters

2.2 Product Component Model

This section describes the design and architecture of PPC. It can be divided in four main modules (Figure 2.3). The first one is the PPC User Interface, that is the GUI that interacts with the user and launches the other three modules. The Monitoring Client contacts JIMS to obtain the monitoring data of a site. These monitoring data are processed and can be shown graphically by the PPC User Interface. The Event

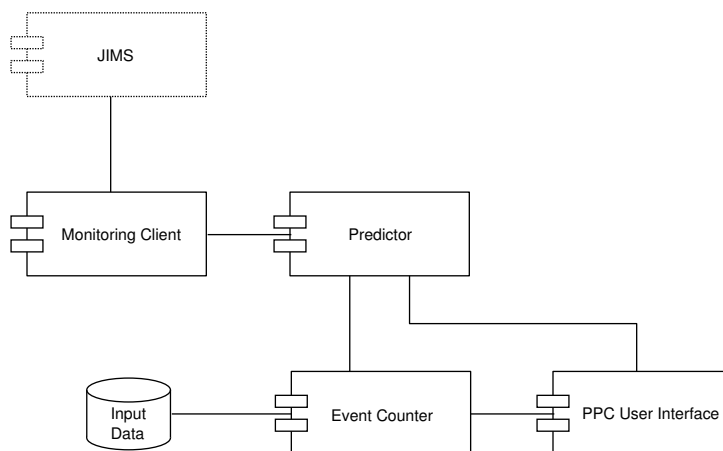


Figure 2.3: PPC component diagram

Counter determines the number of FLOPs, number and type of communications from the kernel selected by the user and the input data. The Predictor establishes a prediction in terms of computing time using all the information (provided by the Monitoring Client and the Event Counter) and applying the analytical models that describe the cost of the kernels. These models were previously developed.

2.3 Detailed Implementation Model

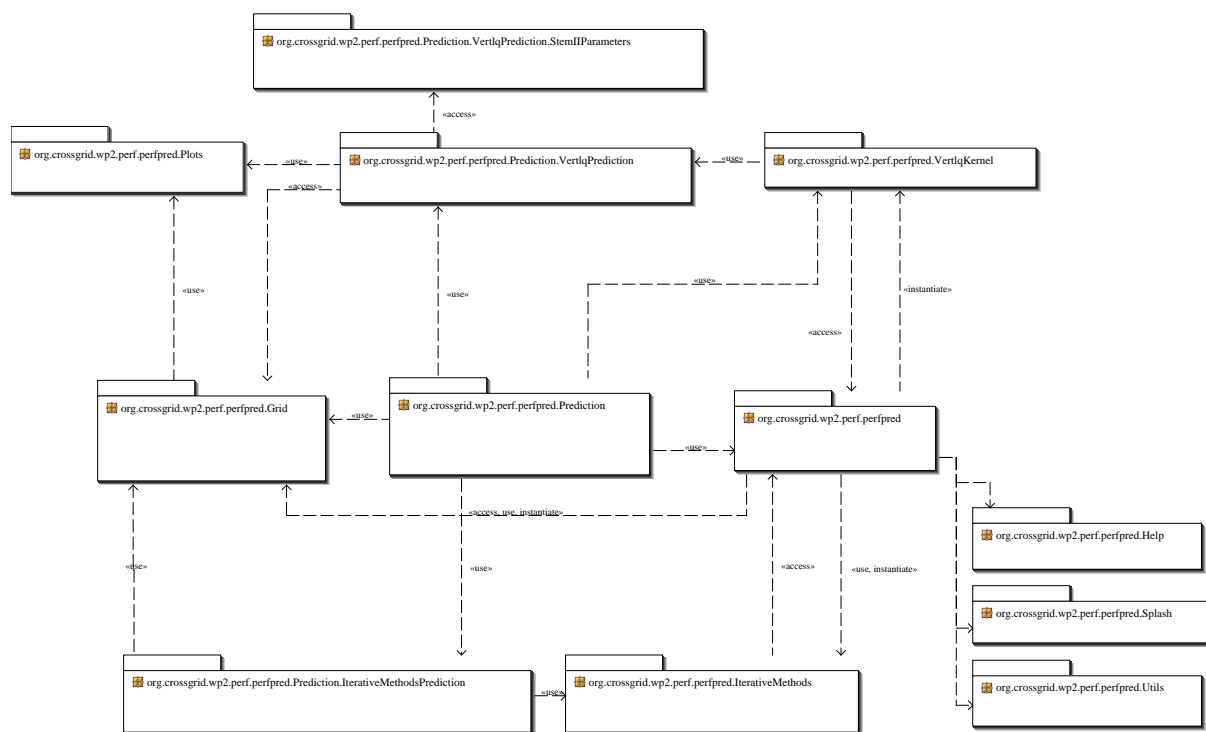


Figure 2.4: PPC package diagram

The PPC source code is organized into 12 packages, as it is shown in Figure 2.4 (for the sake of clarity, in

this figure not all the connections are shown). The three main packages correspond to the main blocks of the component diagram:

org.crossgrid.wp2.perf.perfpred Implements the graphical user interface for PPC.

org.crossgrid.wp2.perf.perfpred.Grid Implements the monitoring client, for contacting JIMS.

org.crossgrid.wp2.perf.perfpred.Prediction: Organized in several subpackages for each kernel. Implements the event counter and the predictor.

2.3.1 Package org.crossgrid.wp2.perf.perfpred

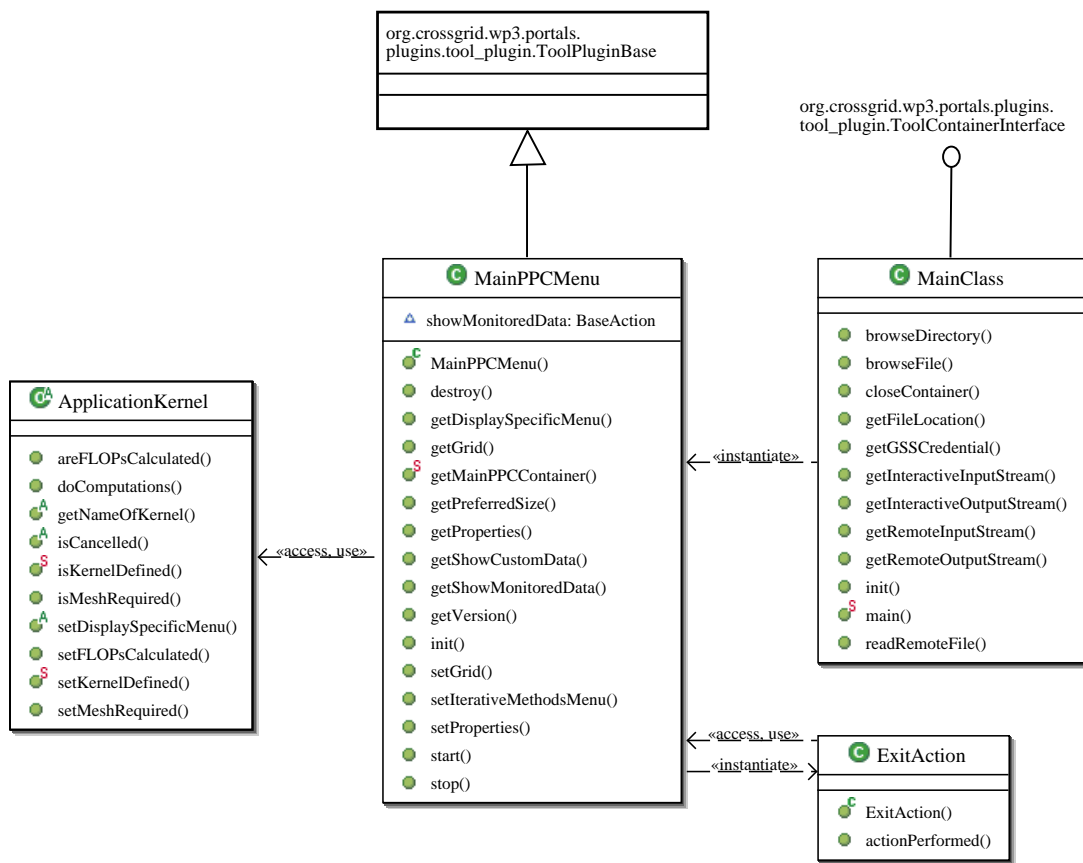


Figure 2.5: PPC class diagram for org.crossgrid.wp2.perf.perfpred

Figure 2.5 shows the class diagram for the org.crossgrid.wp2.perf.perfpred package. There are two main classes, MainClass and MainPPCMenu. The last one implements the GUI, and the former one is used for executing PPC in standalone mode (without the Migrating Desktop).

When in standalone mode, PPC is started using MainClass. This class implements the ToolContainer-Interface, and so emulates the behavior of the MD, creating an MainPPCMenu object that launch the PPC GUI. So, the object MainClass works as the container for the plugin MainPPCMenu.

As an example, when accessing to secondary storage is necessary, the created object call the browseFile method, defined in MainPPCMenu, that opens a window to select a file from secondary storage. When PPC is launched form MD, MainClass is not used, and the browseFile method is obtained from the ToolPluginBase object and opens a window to select a file from MD Virtual Directory. Figure 2.6 shows

the sequence of events that occurs when the PPC is started in standalone mode, and the option for loading a matrix is selected.

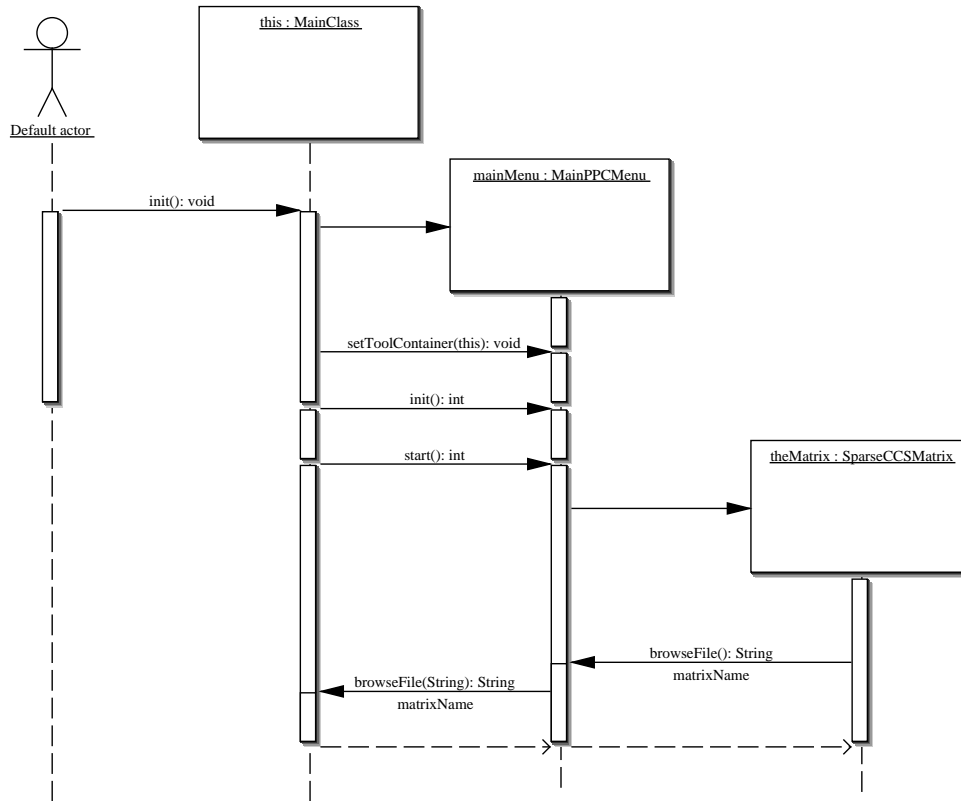


Figure 2.6: Load matrix sequence in standalone mode

When PPC is launched form MD, no MainClass object is created, and MD works as the container of the GUI constructed by the MainPPCMenu plugin.

2.3.2 Package org.crossgrid.wp2.perf.perfpred.Grid

Figure 2.7 shown the class diagram for the org.crossgrid.wp2.perf.perfpred.Grid package (for the sake of clarity, not all the relations between classes are drawn). There are two main classes in this package: Grid and GridPerformanceInfo.

When the option of “Monitoring” or “Custom” is selected in the PPC GUI a new object Grid is created (using the wrapper class GetMonitoringDataGridAction). This object stores all the grid-related information. This information is obtained from the monitoring tool (JIMS) or it is directly specified by the user, when the “Custom” option is selected.

Figure 2.8 shows the sequence of events when the “Monitoring → Get data” option is selected in the GUI. A Grid object is created, which allows the user the possibility of selecting the Grid node to be monitored (using the class JIMSServers).

After the JIMS servers is selected, a GridPerformanceInfo object is created, and this object contacts JIMS in the CE of the remote site in order to obtain the static and dynamic information of the site. This information is stored in an object of class CPUStaticInfo for the static information, and GridPerformanceData for the dynamic one.

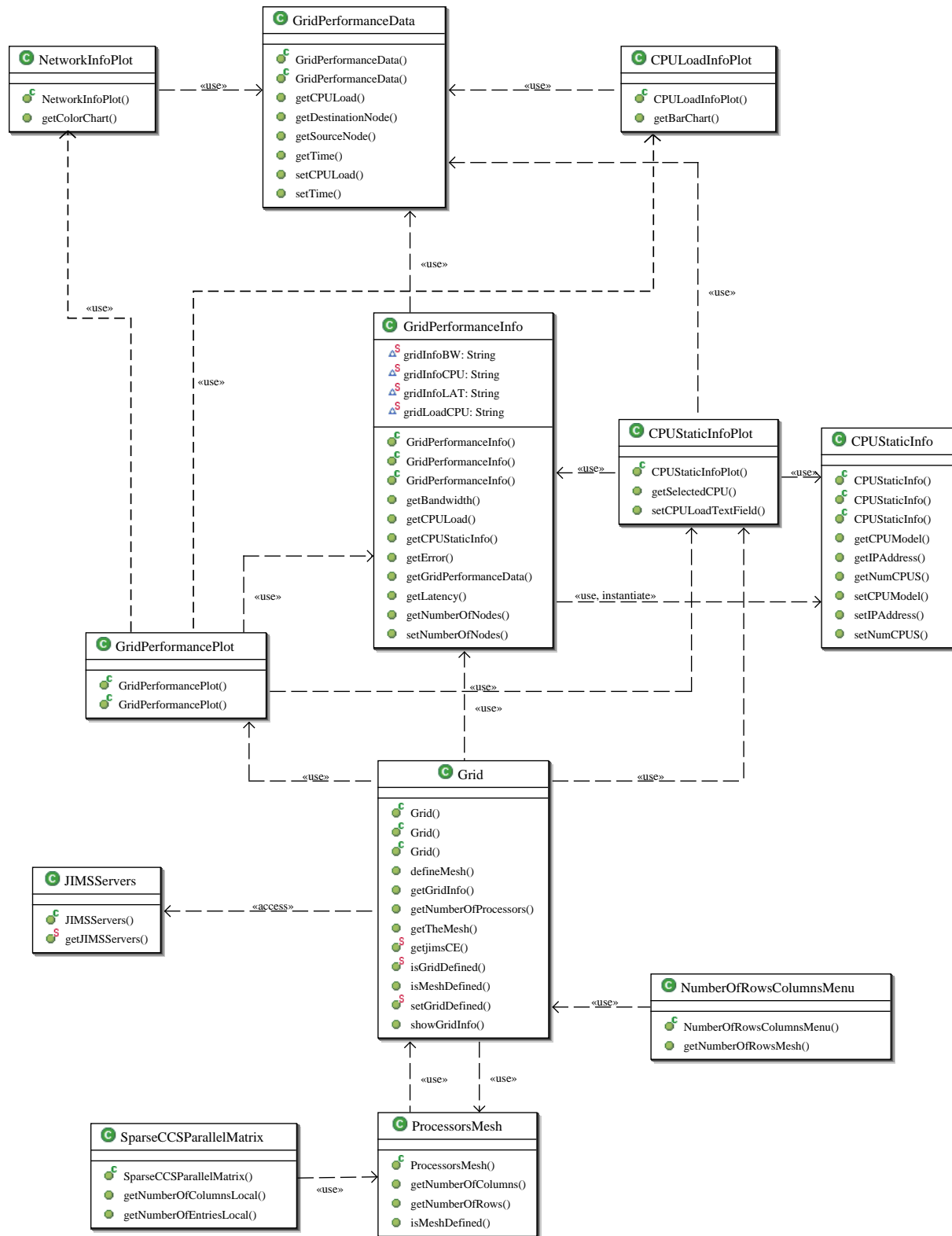


Figure 2.7: PPC class diagram for org.crossgrid.wp2.perf.perfpred.Grid

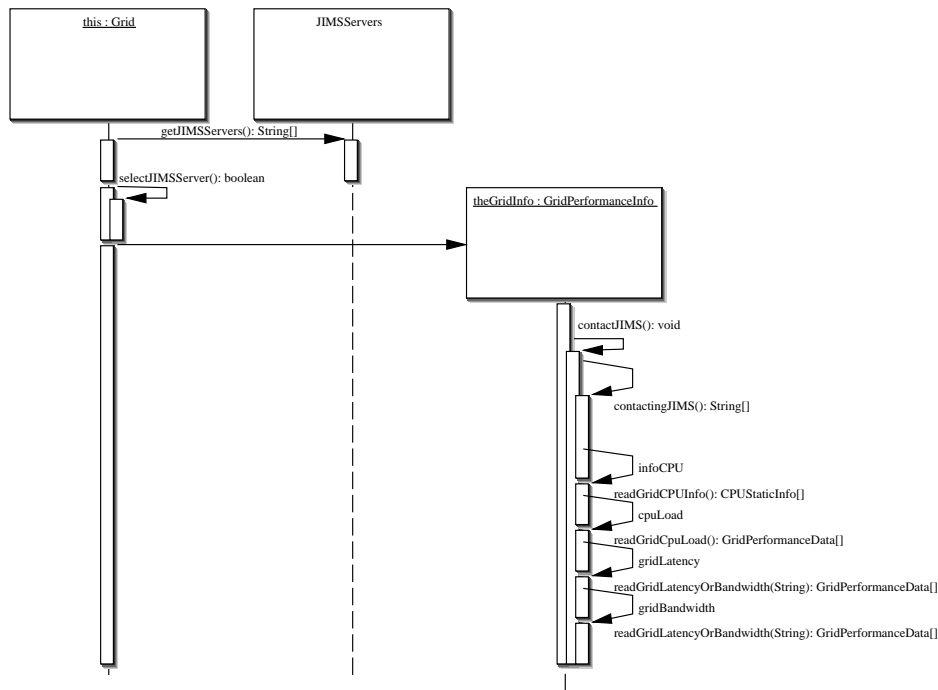


Figure 2.8: Monitoring grid sequence

When the “Monitoring → Show data” option is selected in the PPC GUI, the method showGridInfo of the Grid object is executed. This method opens a new thread to show a graphic representation of the obtained performance data. This representation is implemented through two objects: a CPUStaticInfoPlot object for the static information and a GridPerformancePlot for the dynamic one. This last object creates two new objects of classes CPULoadInfoPlot and NetworkInfoPlot to deal with the graphs for the CPU load and the network latency and bandwidth, respectively.

Finally, some kernels (like the iterative methods) need that a processors mesh were specified in terms of number of rows and number of columns. For manage this question, classes ProcessorsMesh and NumberOfRowsColumnsMenu are used.

2.3.3 Package org.crossgrid.wp2.perf.perfpred.Prediction

Figure 2.9 shows the class diagram for the org.crossgrid.wp2.perf.perfpred.prediction package, only for the verlq kernel. Other kernels have a similar structure.

For each kernel, a class with the name of the kernel has been created (e.g. org.crossgrid.wp2.perf.perfpred.VertlqKernel for the kernel vertlq). When the user selects vertlq as the kernel in the PPC GUI, and object of class VerlqKernel is created. This object implements the Event Counter block of the PPC componet diagram (Figure 2.3). So, it accesses to secondary storage for reading the input data, and computes the “independent-of-grid” data (e.g. the total number of FLOPs for the selected kernel and input data).

As soon as the kernel and the grid have been selected, the prediction can be carried out, creating an object of class org.crossgrid.wp2.perf.perfpred.Prediction.ApplicationKernelPrediction. This is the general class for computing the predictions for all the kernels. This object creates new objects in function of the selected kernels (e.g. for vertlq objects in the packet class org.crossgrid.wp2.perf.perfpred.Prediction.VertlqPrediction are created), that make the prediction for the selected kernel and grid.

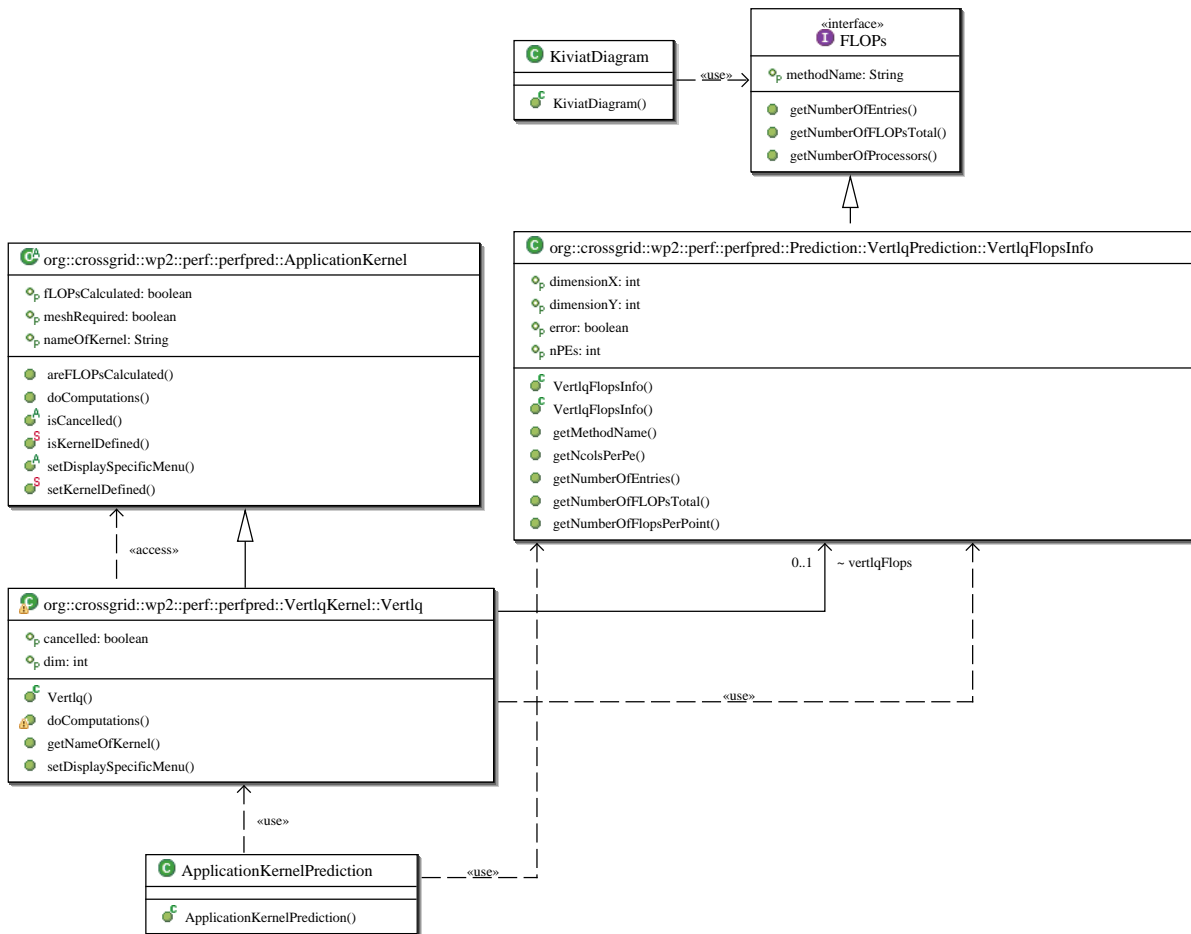


Figure 2.9: PPC class diagram for prediction packages for vertlq kernel

Figure 2.10 shows the sequence of events that occurs when the vertlq kernel is selected and FLOPs, communications and load balance are asked for.

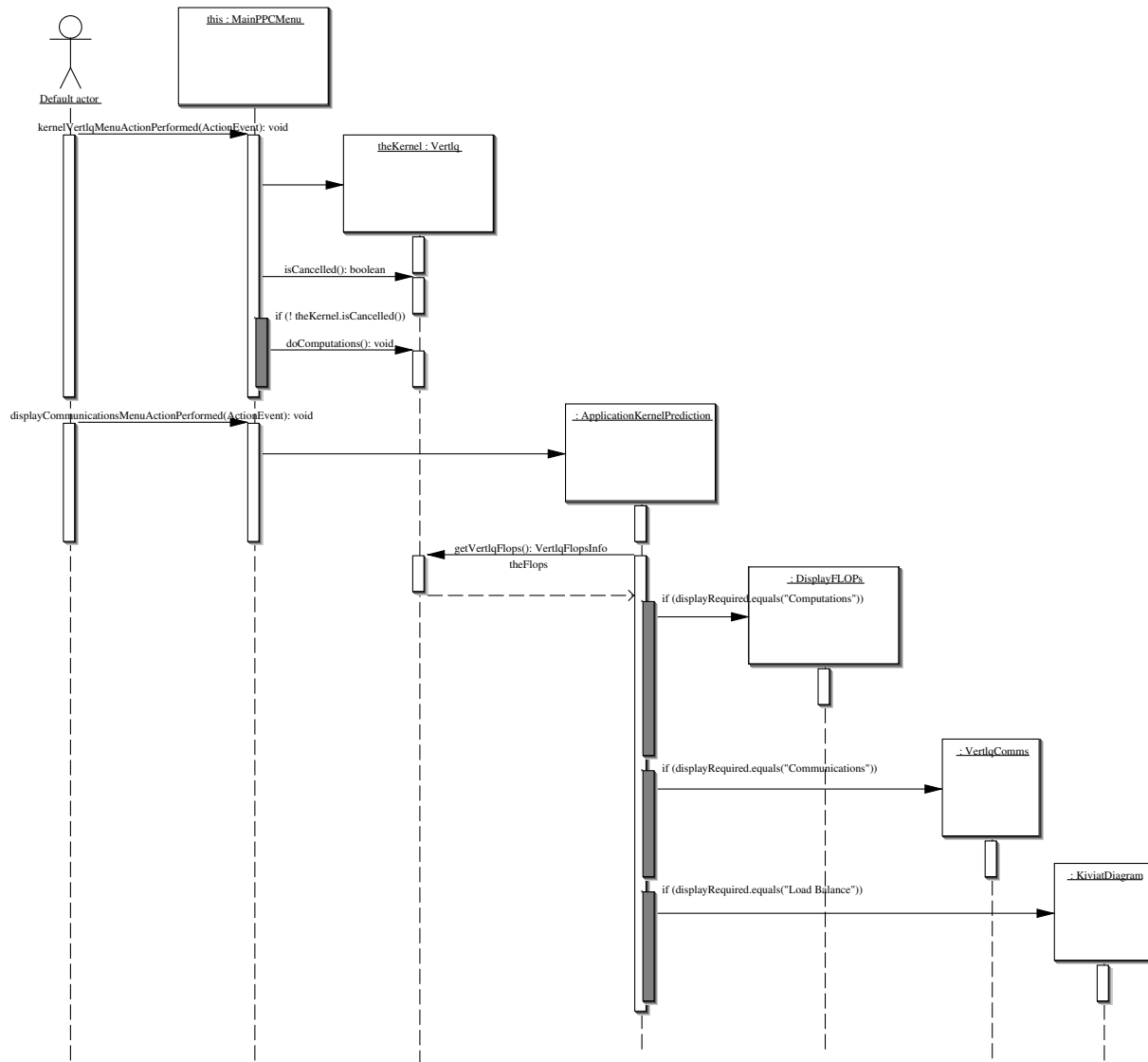


Figure 2.10: Vertlq prediction sequence

3 Contact Information and Credits

People involved in PPC project are:

- Marcos Boullón Magán <marcos@dec.usc.es>
- J. Carlos Cabaleiro <caba@dec.usc.es>
- Tomás F. Pena <tomas@dec.usc.es>
- Francisco F. Rivera <fran@dec.usc.es>

For bug reporting and commenting on the PPC, please contact us in <ppc@dec.usc.es>

4 EDG License Agreement

This section should contain the EDG agreement, under which CrossGrid software is being licensed. If your software follows a different licensing pattern, replace this text with another license, appropriate for your software.

Copyright (c) 2005 CrossGrid. All rights reserved.

This software includes voluntary contributions made to the CrossGrid Project. For more information on CrossGrid, please see <http://www.eu-crossgrid.org>.

Installation, use, reproduction, display, modification and redistribution of this software, with or without modification, in source and binary forms, are permitted. Any exercise of rights under this license by you or your sub-licensees is subject to the following conditions:

1. Redistributions of this software, with or without modification, must reproduce the above copyright notice and the above license statement as well as this list of conditions, in the software, the user documentation and any other materials provided with the software.
2. The user documentation, if any, included with a redistribution, must include the following notice:

This product includes software developed by the CrossGrid Project (<http://www.eu-crossgrid.org>).

Alternatively, if that is where third-party acknowledgments normally appear, this acknowledgment must be reproduced in the software itself.

3. The names CrossGrid and CG may not be used to endorse or promote software, or products derived therefrom, except with prior written permission by cgooffice@cyfronet.krakow.pl.
4. You are under no obligation to provide anyone with any bug fixes, patches, upgrades or other modifications, enhancements or derivatives of the features, functionality or performance of this software that you may develop. However, if you publish or distribute your modifications, enhancements or derivative works without contemporaneously requiring users to enter into a separate written license agreement, then you are deemed to have granted participants in the CrossGrid Project a worldwide, non-exclusive, royalty-free, perpetual license to install, use, reproduce, display, modify, redistribute and sub-license your modifications, enhancements or derivative works, whether in binary or source code form, under the license conditions stated in this list of conditions.

5. DISCLAIMER

THIS SOFTWARE IS PROVIDED BY THE CROSSGRID PROJECT AND CONTRIBUTORS AS IS AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, OF SATISFACTORY QUALITY, AND FITNESS FOR A PARTICULAR PURPOSE OR USE ARE DISCLAIMED. THE CROSSGRID PROJECT AND CONTRIBUTORS MAKE NO REPRESENTATION THAT THE SOFTWARE, MODIFICATIONS, ENHANCEMENTS OR DERIVATIVE WORKS THEREOF, WILL NOT INFRINGE ANY PATENT, COPYRIGHT, TRADE SECRET OR OTHER PROPRIETARY RIGHT.

6. LIMITATION OF LIABILITY

THE CROSSGRID PROJECT AND CONTRIBUTORS SHALL HAVE NO LIABILITY TO LICENSEE OR OTHER PERSONS FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, CONSEQUENTIAL, EXEMPLARY, OR PUNITIVE DAMAGES OF ANY CHARACTER INCLUDING, WITHOUT LIMITATION, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES, LOSS OF USE, DATA OR PROFITS, OR BUSINESS INTERRUPTION, HOWEVER CAUSED AND ON ANY THEORY OF CONTRACT, WARRANTY, TORT (INCLUDING NEGLIGENCE), PRODUCT LIABILITY OR OTHERWISE, ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Bibliography

[QAP] WP5, CYRFRONET; **Quality Assurance Plan**; Evolving document

[PPCINSTALL] T.F. Pena, J.C. Cabaleiro; **PPC Installation Guide**;

<https://savannah.fzk.de/distribution/crossgrid/crossgrid/wp2/wp2.4-perf/wp2.4.2-perfpred-PPC/docs/CG-2.3.2-PPC-InstallGuide-v1.1-USC.pdf>

[PPCUSER] T.F. Pena, J.C. Cabaleiro; **PPC User Manual**;

<https://savannah.fzk.de/distribution/crossgrid/crossgrid/wp2/wp2.4-perf/wp2.4.2-perfpred-PPC/docs/CG-2.3.2-PPC-UserManual-v1.0-USC.pdf>

[JIMSUSER] **JIMS Users Guide**;

http://www.eu-crossgrid.org/user_manuals/jims-1.5.20-usermanual.pdf