# Final Scientific Report: A Scalable Development Environment for Peta-Scale Computing

Carsten Karbach and Wolfgang Frings

Jülich Supercomputing Centre
Forschungszentrum Jülich GmbH
c.karbach@fz-juelich.de
w.frings@fz-juelich.de

February 22, 2013

## 1 Overview Information

| | |
|---|---|
| **Project Title** | A Scalable Development Environment for Peta-Scale Computing |
| **Recipient** | Forschungszentrum Jülich |
| **Principal Investigator** | Wolfgang Frings (Co-PI), Forschungszentrum Jülich |
| **Team members** | Wolfgang Frings, Claudia Knobloch and Carsten Karbach |
| **Award No** | DE-SC0001620 |
| **Project Period** | September 15th, 2009 – September 14th, 2012 |

## 2 Objective

The objective of this project is the extension of the Parallel Tools Platform (PTP) for applying it to peta-scale systems. PTP is an integrated development environment for parallel applications. It comprises code analysis, performance tuning, parallel debugging and system monitoring. The contribution of the Jülich Supercomputing Centre (JSC) aims to provide a scalable solution for system monitoring of supercomputers. This includes the development of a new communication protocol for exchanging status data between the target remote system and the client running PTP. The communication has to work for high latency. PTP needs to be implemented robustly and should hide the complexity of the supercomputer's architecture in order to provide a transparent access to various remote systems via a uniform user interface. This simplifies the porting of applications to different systems, because PTP functions as abstraction layer between parallel application developer and compute resources. The common requirement for all PTP components is that they have to interact with the remote supercomputer. E.g. applications are built remotely and performance tools are attached to job submissions and their output data resides on the remote system. Status data has to be collected by evaluating outputs of the remote job scheduler and the parallel debugger needs to control an application executed on the supercomputer. The challenge is to provide this functionality for peta-scale systems in real-time.

The client server architecture of the established monitoring application LLview [1], developed by the JSC, can be applied to PTP's system monitoring. LLview provides a well-arranged overview of the supercomputer's current status. A set of statistics, a list of running and queued jobs as well as a node display mapping running jobs to their compute resources form the user display of LLview. These monitoring features have to be integrated into the development environment. Besides showing the current status PTP's monitoring also needs to allow for submitting and canceling user jobs. Monitoring peta scale systems especially deals with presenting the large amount of status data in a useful manner. Users require to select arbitrary levels of detail. The monitoring views have to provide a quick overview of the system state, but also need to allow for zooming into specific parts of the system, into which the user is interested in. At present, the major batch systems running on supercomputers are PBS, TORQUE, ALPS and LoadLeveler, which have to be supported by both the monitoring and the job controlling component. Finally, PTP needs to be designed as generic as possible, so that it can be extended for future batch systems.

## 3 Achievements

Within the project period a novel system monitoring component has been developed and integrated in PTP. This section outlines the architecture of this component and presents a set of large-scale HPC systems, which are successfully monitored. Afterward, the main features required for the scalability of the

monitoring system are summarized. At last, the integration of the monitoring component in the development cycle of PTP is documented.

## 3.1 Architecture

PTP has been extended by a new System Monitoring Perspective, which works similar to LLview. It allows to connect to a target system via SSH authentication. The monitoring component is split into the visualization client and the server application called *LML_da*. The latter application is written in Perl and instantly deployed on the target system after the first connection is established. Thus, the server component is installed automatically. LML_da is split into independent modules for the different steps of status data collection. The steps are parsing the outputs of batch system commands, deriving higher level data for graphical components, which can easily be visualized, and defining colors, by which jobs can be identified across the entire monitoring perspective. Only the first step is batch system specific and has to be implemented for each new batch system. This module based architecture minimizes the effort for extending PTP for a new batch system. The data format Large-scale system Markup Language (LML) functions as communication layer between client, server and the server modules. It is used to send requests from the client and to respond with a logical description of the current system status from the server. LML strictly defines the interface between status data generating and visualizing applications. It is implemented as XML Schema and a set of additional constraints documented in [2]. LML uncouples LML_da from the visualization client, so that the status data generated by LML_da can be interpreted by both the original LLview visualization client and the PTP system monitoring perspective. An overview of the described monitoring architecture is shown in figure 1.

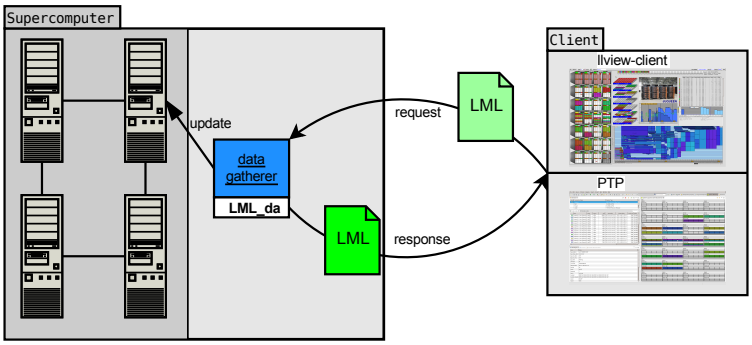


Figure 1: PTP's system monitoring architecture

## 3.2 Scaling examples

PTP's system monitoring component is already in use for a number of peta-scale systems. E.g. the BlueGene/Q system JUQUEEN, which displaces the JSC system JUGENE, with more than 458 thousand cores and a peak performance of 5.9 Petaflops can be displayed. JUQUEEN is listed as number 5 in the November 2012 Top500 List [3]. It runs LoadLeveler as batch system. Figure 2 shows a snapshot of PTP's monitoring perspective for JUQUEEN. The left hand side is divided into the Monitor View, Active Jobs View and Messages View starting from top. The right hand side shows the node display, which renders the system architecture and maps running jobs to the compute resources. In this example each of the smallest rectangles symbolizes a Nodeboard having 512 cores. Colored rectangles are allocated to the corresponding job, while idling resources are left white.

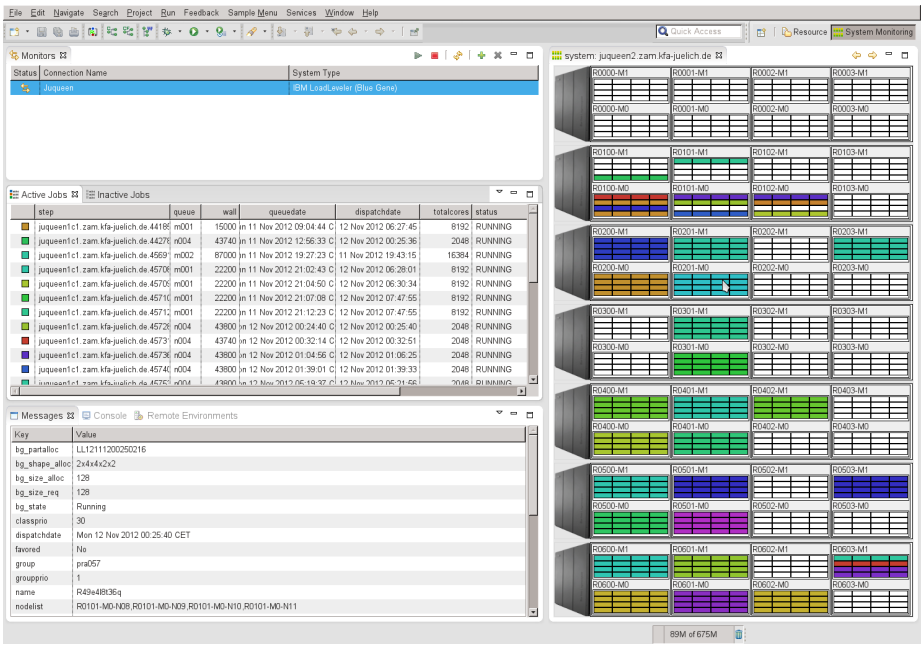


Figure 2: monitoring perspective for JUQUEEN

PTP's system monitoring is also applied to the INCITE system Cray XT Jaguar maintained by the Oak Ridge National Laboratory (ORNL). This supercomputer has 299 thousand processors and runs a combination of Torque and ALPS. As stated in [4] monitoring is successfully tested and used on various XSEDE systems such as "(NICS) Kraken and Keeneland systems, [...] (TACC) Lonestar and Ranger systems, as well as Argonne National Laboratory's Blue Gene/P and Q". Recently PTP was extended to display the status of the DARPA prototype, which is listed as number 10 in the Top500 List

[3]. Next to LoadLeveler the system monitoring also supports the batch system SLURM, which is tested on the Swiss Scientific Computing Center (CSCS) Monte Rosa system.

Moreover, JSC's general purpose cluster JUROPA with more than 26 thousand cores as well as the GPU cluster JUDGE can be monitored with PTP. The node display for JUDGE even allows for monitoring the GPU allocations. Both clusters are running Torque as resource manager, which is triggered to obtain current status data.

### 3.3 Basis for scalability

Scalability is the key feature of the developed monitoring component in order to apply it for peta-scale systems. This features is achieved by scalable server scripts within LML_da, the scalable data format LML and the scalable visualization of the PTP client. The server scripts only collect the status data requested by the user. E.g. if the user only requires to see the status data of his own jobs, the other job data is omitted. LML is designed to avoid redundant data and can transfer different levels of detail. It allows for mapping running jobs to the compute resources, they are running on. But, instead of mapping a job to each core separately, LML collects coherent cores in ranges. In addition, if an entire compute node is allocated to a single job, it is sufficient to map the job to nodes instead of cores. Therefore, LML has to be aware of the system hierarchy, which defines the number of cores in each node, the number of nodes in each node board and so forth. This hierarchy can be collapsed on any level, so that users are able to resolve the job mapping on a low level providing a good overview or on a detailed level showing exactly, which cores are allocated to their jobs. The system hierarchy is used in LML as well as in the visualization. It allows to zoom into the node display on any level of detail or to select only a specific part of the supercomputer for monitoring. This functionality is depicted in figure 3. It shows one of the 56 midplanes of JUQUEEN, which contains 8192 cores. On the lowest level of detail the midplane is rendered as a usage bar, where jobs are placed in a horizontal bar with each job covering an area of the usage bar proportional to the number of consumed cores. On the next level each rectangle represents a node board. The midplane on the bottom of the figure is zoomed to the compute node level each rectangle symbolizing 16 cores. One node board is extracted by the dashed lines and shown on the highest level of detail, where each rectangle represents a core.

Next to the level of detail functions, users can filter the job data transferred from the target system. On the UI jobs can be filtered by their number of cores, their owner or queue. The filtering information is sent to LML_da via the next LML request. The request is interpreted by LML_da and only the relevant information is returned on the next status update.

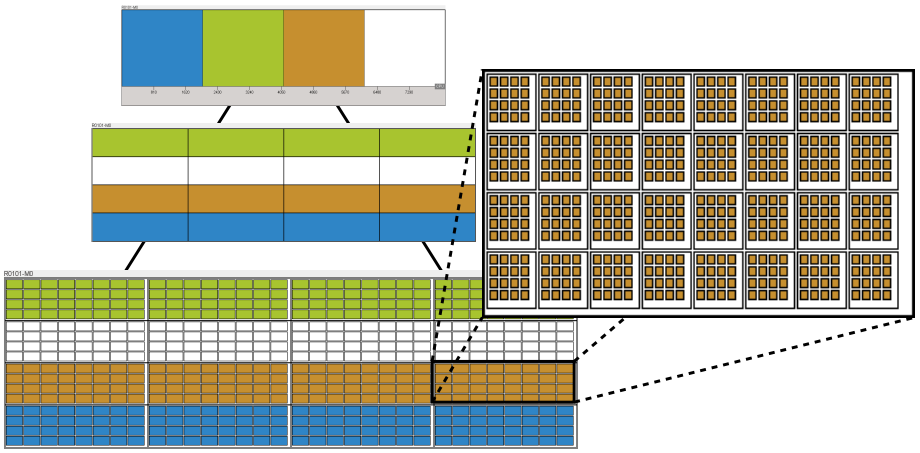

Figure 3: a midplane displayed on different levels of detail

### 3.4 Integration and Deployment

The monitoring system is seamlessly integrated into the development workflow of PTP. As soon as a job is submitted via PTP's Run Configurations the system monitoring perspective is opened. A corresponding monitoring connection is created and started. Users are then able to trace their jobs by switching from submitted over running to completed state. It is easy to check, whether the submission parameters are used correctly by the job scheduler. E.g. the number of requested nodes, cores or GPUs can be compared to the monitoring output. PTP allows for canceling submitted jobs by right-clicking on the job entry in the Job View. Finally, on successful job completion its output can be obtained via the monitoring perspective.

Besides the batch systems LoadLeveler, SLURM and Torque mentioned in above example systems, PTP supports monitoring and job submission for the batch systems Cobalt, Grid Engine, IBM Parallel Environment, IBM Platform LSF and PBS. Monitoring also incorporates status data obtained from the Application Level Placement Scheduler(ALPS), which can be run in combination with SLURM or TORQUE.

The system monitoring implemented by JSC is included in the Eclipse distribution *Eclipse for Parallel Application Developers*, which can be downloaded as top level project from the Eclipse downloads website [5]. It is in production since 2011 and is continuously updated and extended for new target systems. In addition, a stand-alone client only for system monitoring is published [6]. Users only interested in monitoring their target systems can use this distribution in order to simplify the access on the monitoring functionality included in PTP. This client also includes a simple interface to submit jobs, whereas PTP's application development functionality is excluded. It represents an alternative for the monitoring application LLview. Applying LLview to a new target system is

more complicated, since client and server have to be installed separately. However, LLview still provides more statistical diagrams than PTP's monitoring system.

### 3.5 Conclusion

To conclude, the monitoring system developed by the JSC and integrated into PTP is successfully applied for peta-scale supercomputers. An extensible architecture is formed by LML_da and the PTP visualization component. Their interactions are defined by the communication interface provided by LML. Level of detail functions and filtering operations allow for monitoring the status of a wide range of target systems in real-time. The combination of Eclipse based development environment, PTP's job control component and the monitoring system simplifies the access to supercomputers and provides a uniform interface to all supported batch systems.

## 4 Publications

In 2011 the paper [7] has been published, which describes the architecture of the Parallel Tools Platform. It focuses on the scalability of the monitoring system developed by the JSC. Examples of the data format LML and its visualization are outlined. Moreover, the target system configuration is documented, which allows to quickly adapt PTP to new batch system interfaces.

The paper [4] extends the previous one by additional functions for level-of-detail visualization and provides example supercomputers monitored by PTP in order to showcase scalability.

The bachelor thesis [2] deals with the design of LML. It documents design decisions and describes how to map the LML data to a corresponding graphical representation. It also outlines LML's features for avoiding redundancies and for implementing various levels of detail. A generic job scheduler simulator is designed and implemented in the master thesis [8]. It can be applied as efficient on-line prediction for future job dispatch dates. The target system's job scheduler is simulated based on current status data obtained from the server component of PTP's system monitoring. Thus, a valuable tool is provided to the users for better understanding the scheduling algorithms. It also allows to detect future idling resources, which can be used by appropriately sized job requests.

A full day tutorial on using PTP as development environment including hands-on sessions on the JSC supercomputer JUDGE was given in June 2012 [9]. Within the PTP User-Developer Workshop a talk documenting how to extend the system monitoring for further target systems was presented [10]. Moreover, a summary of the results obtained from this project especially focusing on the development work of the JSC was provided by the talk [11].

A couple of web sites document the usage of PTP as system monitoring application. A large list of relevant documentation, which is continuously updated,

is provided by PTP's wiki page [12]. This page especially contains a summary of the resource manager and monitoring framework [13] outlining their architecture and interactions. Finally, the web site [14] documents the data format LML. It provides the entire XML Schema definition, a couple of example LML files and an on-line validation tool for generated LML files.

# References


[1] Jülich Research Centre. LLVIEW: graphical monitoring of batch system controlled cluster. `http://www.fz-juelich.de/jsc/llview/`, March 2005.

[2] Carsten Karbach. Konzeption und Umsetzung einer Beschreibungssprache für Statusinformationen von Parallelrechnern als Basis einer Webschnittstelle für LLview, August 2010.

[3] Hans Meuer, Erich Strohmaier, Jack Dongarra, and Horst Simon. Top 500 list. `http://www.top500.org/list/2012/11/`, November 2012.

[4] G.R. Watson, W. Frings, C. Knobloch, C. Karbach, and A.L. Rossi. A scalable control and monitoring framework to aid the development of supercomputer applications. `http://wwwx.cs.unc.edu/~tgamblin/whist-2012/papers/whist-2012-watson.pdf`, 2012.

[5] Eclipse Foundation. Eclipse downloads. `http://www.eclipse.org/downloads/`, February 2013.

[6] G.R. Watson. Stand-alone SysMon Application. `http://download.eclipse.org/tools/ptp/builds/kepler/nightly/index.html`, February 2013.

[7] G.R. Watson, W. Frings, C. Knobloch, C. Karbach, and A.L. Rossi. Scalable control and monitoring of supercomputer applications using an integrated tool framework. In *Parallel Processing Workshops (ICPPW), 2011, 40th International Conference on, Edited by Jang-Ping Sheu, National Tsing Hua University, Taiwan; Cho-Li Wang, The University of Hong Kong, China, IEEE, 2011, 978-1-4577-1337-8, 457 - 466*, 2011. Record converted from VDB: 12.11.2012.

[8] Carsten Karbach. Design and implementation of a highly configurable and efficient simulator for job schedulers on supercomputers. Master's thesis, FH-Aachen, University of Applied Sciences, `http://hdl.handle.net/2128/4703`, August 2012.

[9] G. Watson, Carsten Karbach, and Claudia Knobloch. Developing Scientific Applications with the Eclipse Parallel Tools Platform (PTP). In *PTP-Tutorial at JSC*, 2012. Record converted from VDB: 12.11.2012.

[10] Carsten Karbach and Wolfgang Frings. Monitoring system basics, and adding support for a new batch system. NCSA Eclipse PTP User-Developer Workshop, Schiller Park, IL, USA, and Jülich, Germany, 09/18 - 09/20 2012.

[11] Carsten Karbach. PTP is watching your supercomputer. JSC-Jahresabschlusskolloquium 2012, Jülich, 12/13 2012.

[12] The Eclipse Foundation. PTP. `http://wiki.eclipse.org/PTP`, February 2013.

[13] The Eclipse Foundation. PTP Design document: scalability. `http://wiki.eclipse.org/PTP/designs/scalability`, February 2013.

[14] Carsten Karbach. LML: Large-scale system Markup Language. `http://llview.zam.kfa-juelich.de/LML`, February 2013.