

# Monitoring system basics, and adding support for a new batch system

September 19, 2012 | Carsten Karbach and Wolfgang Frings

# Content

- 1 PTP System Monitoring
- 2 Architecture
- 3 Large-scale system Markup Language (LML)
- 4 Implementation
- 5 Scalability
- 6 Add new target system
- 7 Conclusion

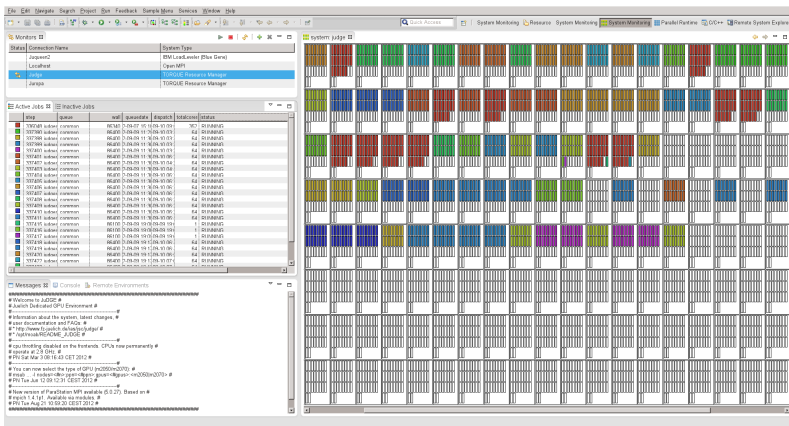
# Part I: PTP System Monitoring

September 19, 2012 | Carsten Karbach and Wolfgang Frings

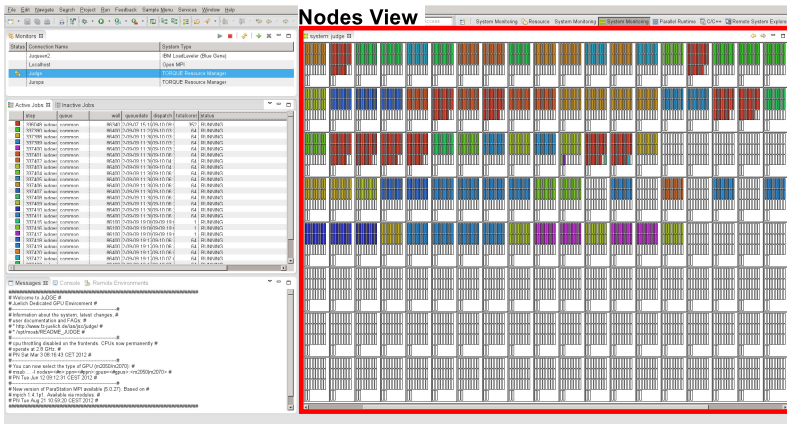
## PTP Monitoring Scope

- job and system monitoring of large-scale supercomputers
- monitoring of **multiple target systems** in one perspective
- support for many batch systems (Grid Engine, LoadLeveler, Open MPI, PBS, Slurm, Torque)
- overview of the system on a **single screen**
- based on monitoring application **LLview**

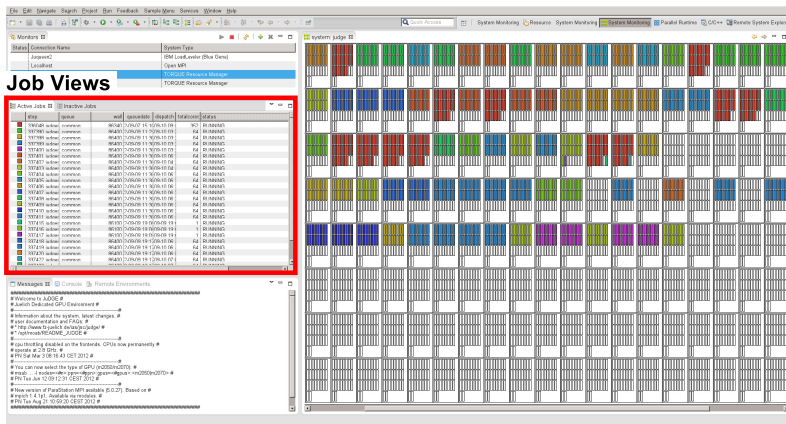
# PTP Monitoring Perspective



# PTP Monitoring Perspective



# PTP Monitoring Perspective

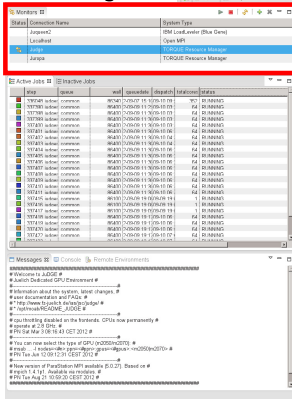


The screenshot displays the PTP System Monitoring interface. On the left, the 'Job Views' window is highlighted with a red border, showing a table of active jobs. The table has columns for Job ID, Name, Status, User, Variables, Dispatch, Subaction, and Status. The jobs listed are numbered from 107161 to 107277, all with a status of 'common' and 'RUNNING'. Below the table, a 'Messages' window shows system logs, including information about the system, local changes, and the availability of various components like the Parallel MPI and PTP Task Aug.

The main window shows a Gantt chart for the 'system judge' process. The chart consists of a grid of colored bars representing the execution of different subactions over time. The colors used include red, green, blue, yellow, and purple. The x-axis represents time, and the y-axis represents different subactions or tasks.

# PTP Monitoring Perspective

## Monitoring View



**Monitors**

Name	Connection Name	System Type
Jaune02	IBM LoadLeveler (Blue Gene)	IBM LoadLeveler (Blue Gene)
Localhost	Open MPI	Open MPI
Judge	TORQUE Resource Manager	TORQUE Resource Manager
Junos	TORQUE Resource Manager	TORQUE Resource Manager

**Active Jobs**

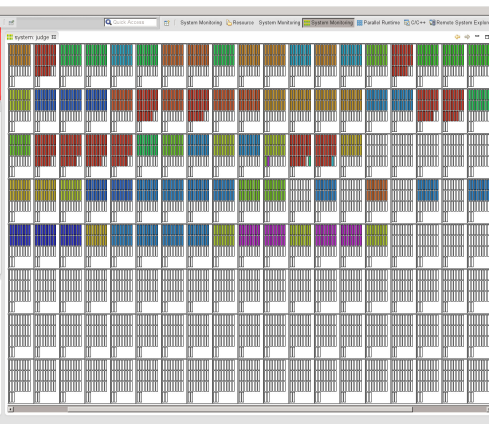
Job	Queue	Nodes	Resources	Status
337848	ibm	common	64	RUNNING
337849	ibm	common	64	RUNNING
337850	ibm	common	64	RUNNING
337851	ibm	common	64	RUNNING
337852	ibm	common	64	RUNNING
337853	ibm	common	64	RUNNING
337854	ibm	common	64	RUNNING
337855	ibm	common	64	RUNNING
337856	ibm	common	64	RUNNING
337857	ibm	common	64	RUNNING
337858	ibm	common	64	RUNNING
337859	ibm	common	64	RUNNING
337860	ibm	common	64	RUNNING
337861	ibm	common	64	RUNNING
337862	ibm	common	64	RUNNING
337863	ibm	common	64	RUNNING
337864	ibm	common	64	RUNNING
337865	ibm	common	64	RUNNING
337866	ibm	common	64	RUNNING
337867	ibm	common	64	RUNNING
337868	ibm	common	64	RUNNING
337869	ibm	common	64	RUNNING
337870	ibm	common	64	RUNNING
337871	ibm	common	64	RUNNING
337872	ibm	common	64	RUNNING
337873	ibm	common	64	RUNNING
337874	ibm	common	64	RUNNING
337875	ibm	common	64	RUNNING

**Messages**

```

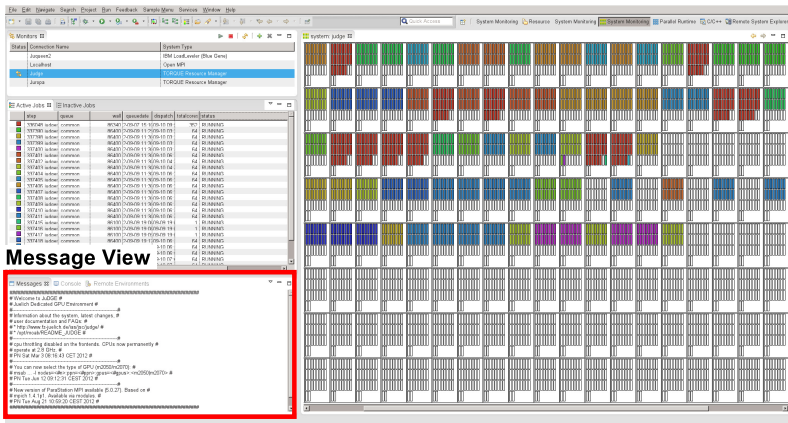
#####
# Welcome to JUDGE #
#####
# Launch Dedicated GPU Environment #
#####
# Information about the system, latest changes, #
# user documentation and FAQs #
# * http://www.fz-juelich.de/ptp #
# * http://www.RESCALE_JUDGE #
#####
# cpu throttling enabled on the frontend. CPUs now permanently #
# operate at 2.0 GHz. #
#####
# Pfu 04 Mar 3 06:16:43 CEST 2012 #
#####
# You can now select the type of GPU (nVidia/nVidia2) #
# usage: ./install-ibm-gpu/ibm-gpu-ibm-ibm-nv200-nv200-CDT2 #
#####
# Pfu Tue Jun 12 09:12:31 CEST 2012 #
#####
# New version of Parastation MPI available (0.0.27). Based on #
# request 14.192. Available via modules. #
# Pfu Tue Aug 21 10:59:20 CEST 2012 #
#####

```





# PTP Monitoring Perspective



The screenshot displays the PTP System Monitoring interface. On the left, there are panels for 'Monitors' and 'Active Jobs'. The 'Active Jobs' panel shows a table of jobs with columns for JobID, Name, User, Status, and others. The main area is a grid of colored squares representing system status for various nodes. A 'Message View' window is open at the bottom left, showing system messages and logs.

**Message View**

```

Messages Console Remote Environments
#####
# Welcome to Jülich #
# Jülich Distributed GPU Environment #
#####
# Information about the system, latest changes, #
# user documentation and FAQs #
# http://www.juelich.de/ptp/ptp.html #
# ** /home/MEADOWS_JUDGE #
#####
# GPU monitoring enabled on the hardware. GPUs now permanently #
# operate at 20 GHz. #
# PTP Sat Mar 9 08:16:43 CEST 2012 #
#####
# You can now select the type of GPU (nVidia or AMD) #
# using _GPU_DEVICE_NAME=your-gpu-name (e.g. nVidia-GTX580) #
# PTP Tue Jul 10 09:12:31 CEST 2012 #
#####
# New version of Parallel-MPI available (0.0.27). Based on #
# version 1.4.141. Available on nodes. #
# PTP Tue Aug 21 10:50:20 CEST 2012 #
#####
  
```

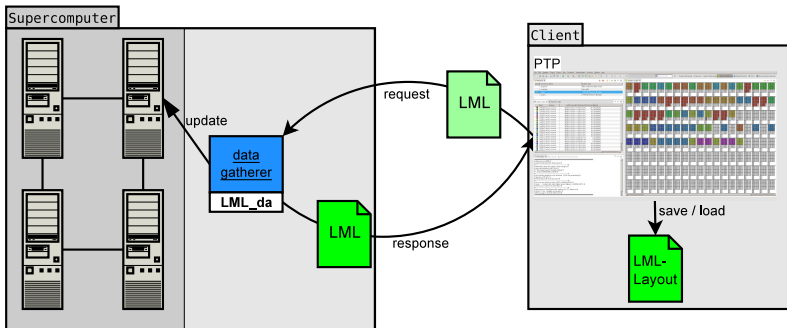
## Monitoring Views

- **Nodes View** renders target system architecture, maps jobs to compute resources
- **Active Jobs View** lists running jobs
- **Inactive Jobs View** lists queued jobs
- **Monitoring View** selects active target system, starts/stops monitoring
- **Message View** shows message of the day

## Part II: Architecture

September 19, 2012 | Carsten Karbach and Wolfgang Frings

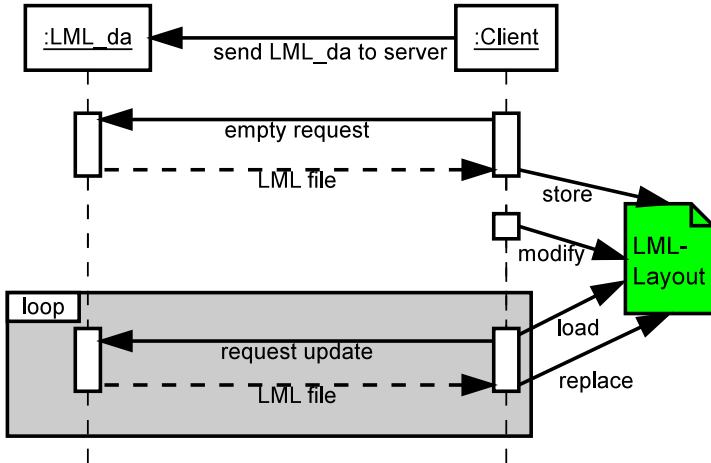
# Monitoring Architecture I



## Monitoring Architecture II

- **LML\_da** gathers status information, calls target system's remote commands, written in Perl
- **LML** is a data format for status information of supercomputers
- LML request: contains table filtering information, visible/hidden columns
- LML response: contains the request and status information
- client stores **current layout** request for successive Eclipse sessions

# Communications Protocol



# Part III: Large-scale system Markup Language (LML)

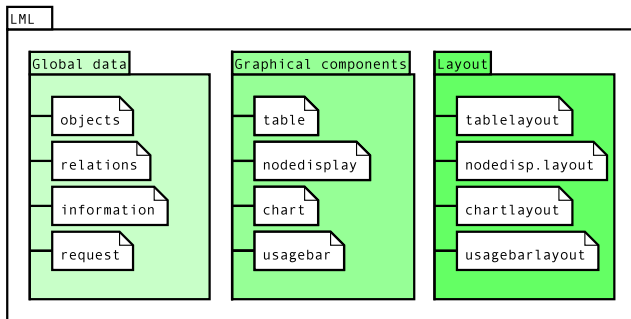
September 19, 2012 | Carsten Karbach and Wolfgang Frings

## Large-scale system Markup Language (LML)

- markup language for description of supercomputer's status
- **interface** between LML\_da and visualization clients
- describes all **graphical components** available in LLview (job-list, node display, charts ...)
- logical and system independent description of current status
- close to graphical representation  
→ thin visualization clients
- implemented in **XML**, validation against **XML-Schema**



## LML Structure



- **Global data:** intermediate data format, scheduling objects
- **Graphical components:** data for visualization components
- **Layout:** hints for visualization, node display hierarchy

## Example node display

### LML-description of a node display

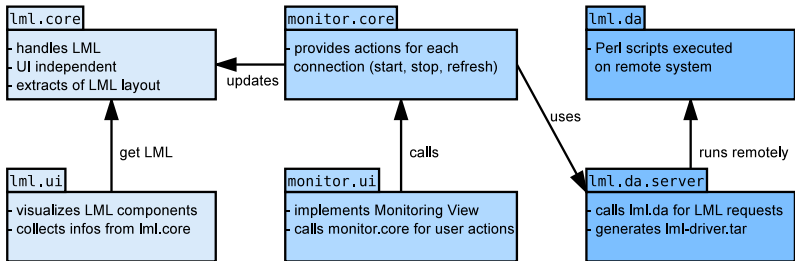
```
<nodedisplay title="Cluster" id="1">
  <scheme>
    <el1 tagname="Node" min="1" max="4">
      <el2 tagname="CPU" min="1" max="3"/>
    </el1>
  </scheme>

  <data>
    <el1 min="1" max="2" oid="job1">
      <el2 min="3" oid="job2"/>
    </el1>
    <el1 min="3" max="4" oid="job2"/>
  </data>
</nodedisplay>
```

# Part IV: Implementation

September 19, 2012 | Carsten Karbach and Wolfgang Frings

## Plug-in Overview



## Plug-in Details I

### lml.core

- uses **JAXB** to manage LML files
- provides **helper classes** for comfortable access on LML
- defines **events and listeners** for all UI actions

### lml.ui

- implements a **ViewPart** for Nodes / Jobs and Info Views
- renders LML components, considers LML layout hints
- UI implementation:
  - Jobs View → JFace TreeViewer
  - Nodes View → nests Composites, PaintListener
  - Info View → SWT Text / Table

## Plug-in Details II

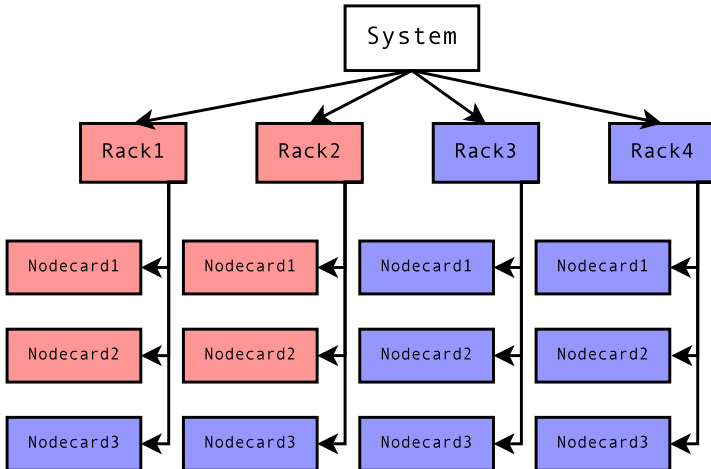
### lml.da

- set of Perl scripts → simple installation
- split into independent modules:
  - 1 **Driver** calls system specific commands to get status infos
  - 2 **Combiner** merges LML files generated by the Driver
  - 3 **AddColor** assigns unique color to each job
  - 4 **LML2LML** converts intermediate format to final LML
- fully configurable workflow
- only Driver scripts and workflow description are target system specific

# Part V: Scalability

September 19, 2012 | Carsten Karbach and Wolfgang Frings

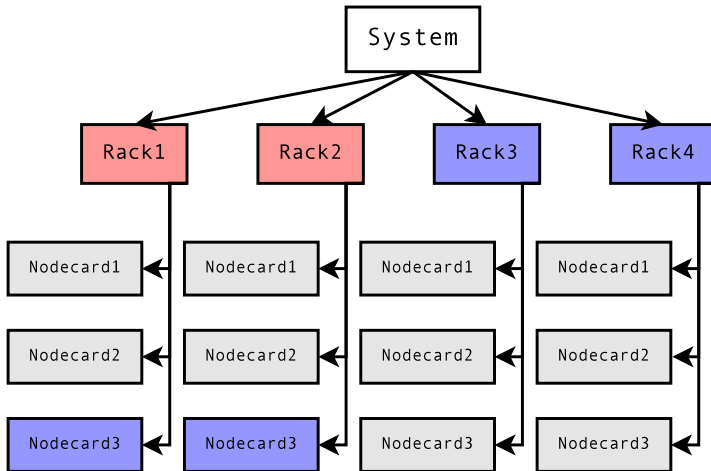
## Node Display – Compression



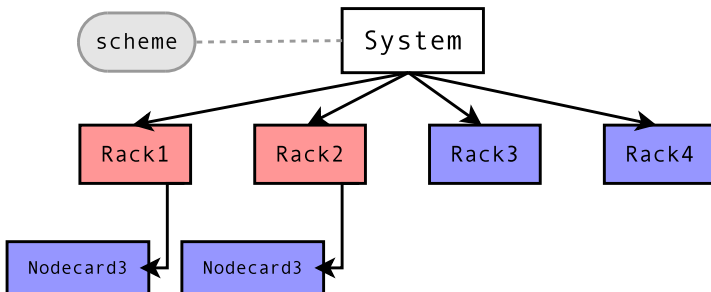


# Node Display – Compression

## Attribute Inheritance



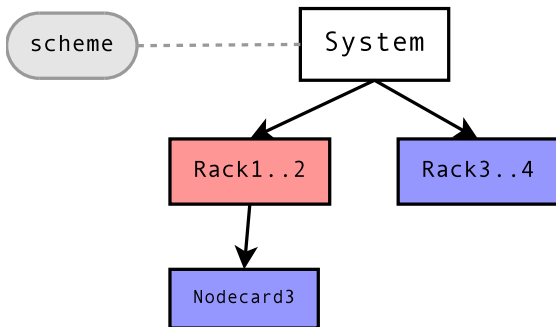
## Node Display – Compression



### Scheme

defines architecture of **empty system**

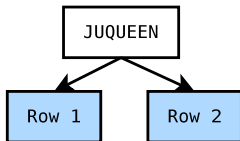
## Node Display – Compression Ranges



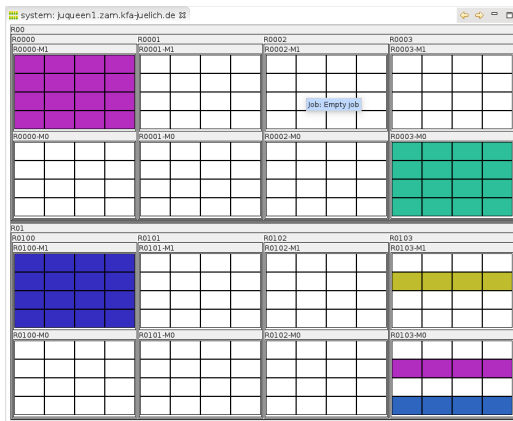
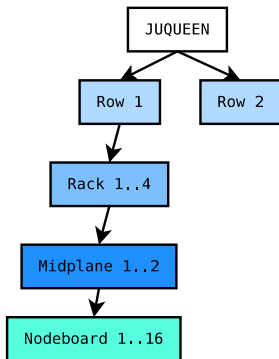
### Result

→ 3 objects instead of 16

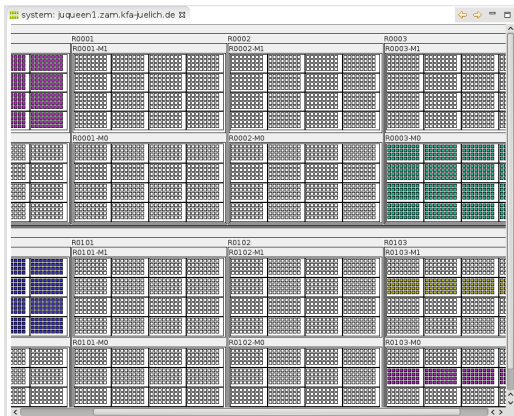
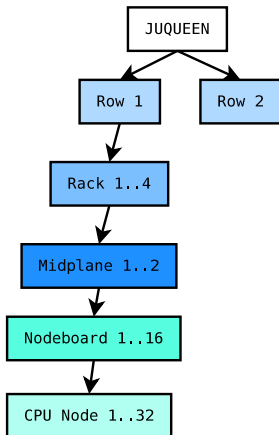
## Scalable Visualization – Row Level



## Scalable Visualization – Nodeboard Level

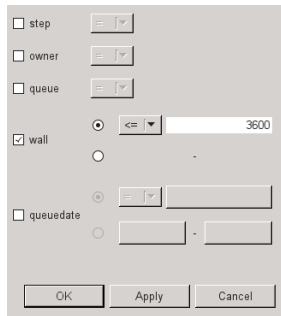


## Scalable Visualization – CPU Node Level



## Table Filtering

- scalability of job data display
- filtering by specification of attribute values / ranges
- **server- and client-side** filtering



The screenshot shows a dialog box for filtering job data. It contains the following elements:

- step: followed by a dropdown menu.
- owner: followed by a dropdown menu.
- queue: followed by a dropdown menu.
- wall: followed by a radio button selected with '<=' and a text input field containing '3600'.
- queuedate: followed by a radio button selected with '=', and two text input fields separated by a hyphen.

At the bottom of the dialog are three buttons: OK, Apply, and Cancel.

# Part VI: Add new target system

September 19, 2012 | Carsten Karbach and Wolfgang Frings



## LML\_da – Data Extraction

### Which status data is required?

- **jobs:** owner, queue, dispatch date, state ...
- **nodes:** memory, cores, state, GPUs ...
- **system:** hostname, date, high messages ...

### How is status data extracted?

- one Perl script per data type
- call batch system commands  
(e.g. *qstat* and *pbsnodes* for TORQUE)
- convert output into intermediate LML files

## LML\_da – add new target system

- 1 write scripts for extracting status information e.g.
  - **da\_jobs\_info\_LML.pl** gathers jobs
  - **da\_nodes\_info\_LML.pl** gathers nodes
  - **da\_system\_info\_LML.pl** gets global system information
- 2 write functions for checking remote commands and for composing the workflow → **da\_check\_info\_LML.pl**
- 3 put all scripts into a folder in *lml.da/rms*, folder named after target system
- 4 add *lml.monitor.ui.monitors* extension for the new system type

## Client – add new target system

- LML\_da generates **system independent** LML files
- client visualizes new target systems without any changes
- **but** a system specific LML layout can be configured
- currently this layout configuration is tricky

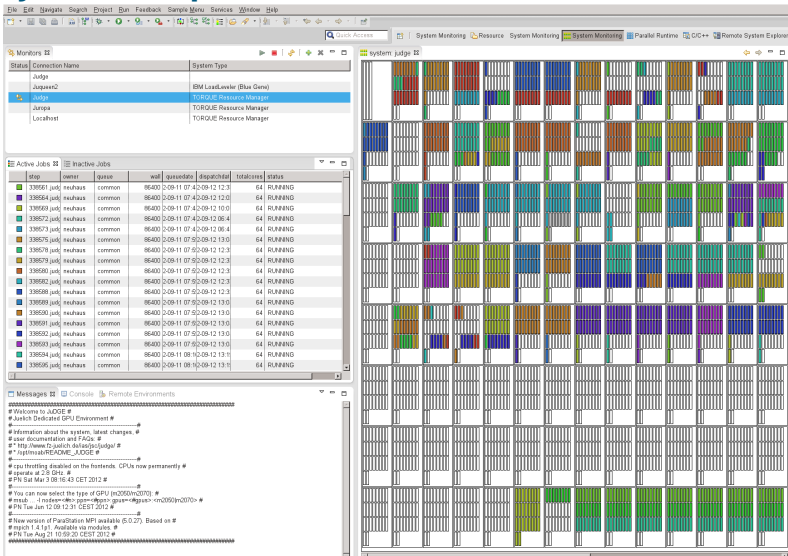
## LML Layout Configuration I

- the following procedure is only meant for **testing**
- **client side configuration** of LML layout is planned
- **long-term target:**  
configure entire layout via GUI, not via XML file
- the layout could be saved together with  
the target system configuration

## LML Layout Configuration II

- 1 write your own LML layout file, or adjust an existing (examples in *lml.da* plug-in)
- 2 start the monitoring connection to the target system
- 3 log-in on the remote machine and switch to the *.eclipsesettings* folder in your home directory
- 4 create the file **.LML\_da\_options** with content **nocheckrequest=1**
- 5 replace **samples/layout\_default.xml** in *.eclipsesettings* with your own layout
- 6 refresh the monitoring perspective on your client

# Layout Example – Default



The screenshot displays the Jülich System Monitoring interface. The top panel shows a list of monitors with columns for Status, Connection Name, and System Type. The 'Active Jobs' panel is expanded, showing a table of running jobs with columns for job ID, owner, queue, wall time, queue date, dispatch time, total cores, and status. The main area contains a grid of resource usage charts for various nodes, with a 'system: judge' tab selected. The bottom panel shows system messages and console output.

**Monitors**

Status	Connection Name	System Type
Judge	Juqeen2	IBM LoadLeveler (Blue Gene)
Judge	Jurpa	TORQUE Resource Manager
Judge	Localhost	TORQUE Resource Manager

**Active Jobs**

jobid	owner	queue	wall	queue date	dispatch time	total cores	status
338561	jud	common	86400	2009-11-07 4:00-12 12 3		64	RUNNING
338564	jud	common	86400	2009-11-07 4:00-12 12 0		64	RUNNING
338569	jud	common	86400	2009-11-07 4:00-12 10 0		64	RUNNING
338572	jud	common	86400	2009-11-07 4:00-12 08 4		64	RUNNING
338573	jud	common	86400	2009-11-07 4:00-12 08 4		64	RUNNING
338575	jud	common	86400	2009-11-07 5:00-12 13 0		64	RUNNING
338578	jud	common	86400	2009-11-07 5:00-12 12 3		64	RUNNING
338579	jud	common	86400	2009-11-07 5:00-12 12 3		64	RUNNING
338580	jud	common	86400	2009-11-07 5:00-12 12 3		64	RUNNING
338582	jud	common	86400	2009-11-07 5:00-12 12 3		64	RUNNING
338588	jud	common	86400	2009-11-07 5:00-12 12 3		64	RUNNING
338589	jud	common	86400	2009-11-07 5:00-12 13 0		64	RUNNING
338590	jud	common	86400	2009-11-07 5:00-12 13 0		64	RUNNING
338591	jud	common	86400	2009-11-07 5:00-12 13 0		64	RUNNING
338592	jud	common	86400	2009-11-07 5:00-12 13 0		64	RUNNING
338593	jud	common	86400	2009-11-07 5:00-12 13 0		64	RUNNING
338594	jud	common	86400	2009-11-08 1:00-12 13 1		64	RUNNING
338595	jud	common	86400	2009-11-08 1:00-12 13 1		64	RUNNING

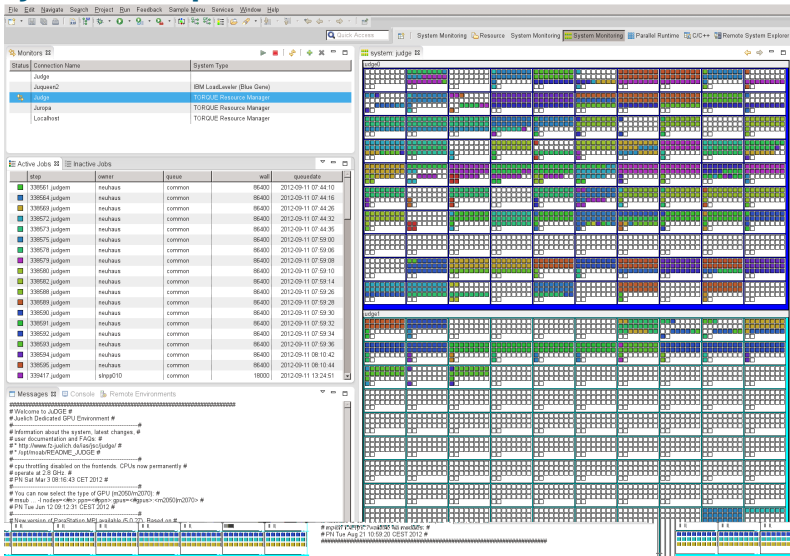
**Messages**

```

#####
# Welcome to JUDGE #
# Jülich Dedicated GPU Environment #
#
# Information about the system, latest changes, #
# user documentation and FAQs. #
# * http://www.fz-juelich.de/parastation/judge/ #
# * http://www.fz-juelich.de/parastation/judge/ #
#
# GPU throttling disabled on the host nodes. CPUs now permanently #
# operate at 2.8 GHz. #
# PN Sat Mar 3 09:16:43 CET 2012 #
#
# You can now select the type of GPU (m2050/m2070). #
# # m2050: /bin/rdmbo-gpu-m2050-gpu-m2050-m2050-m2070 #
# # m2070: /bin/rdmbo-gpu-m2070-gpu-m2070-m2070-m2070 #
# # PN Tue Jan 12 09:12:31 CET 2012 #
#
# New version of ParaStation MPI available (5.0.27). Based on #
# mpich 1.4.1g1. Available via modules. #
# PN Tue Aug 21 10:59:20 CEST 2012 #
#####

```

# Layout Example – Custom I



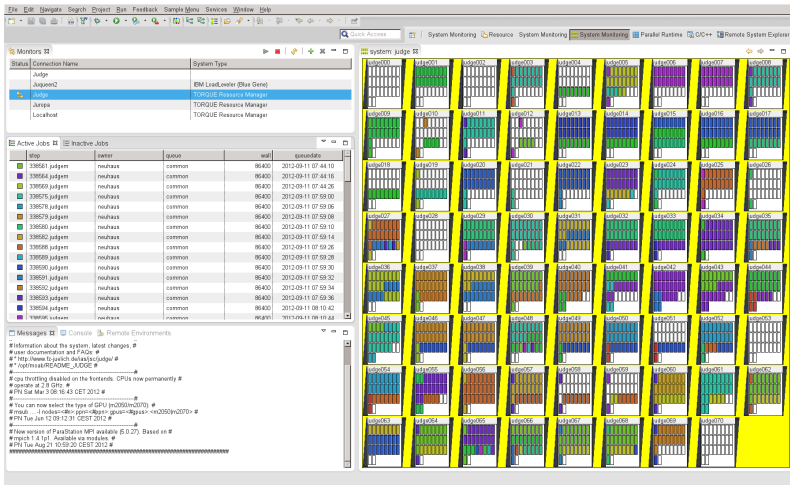
The screenshot displays the JUDGE system monitoring interface. The top menu includes File, Edit, Navigata, Search, Project, Run, Feedback, Sample Menu, Services, Window, and Help. The main window is divided into several panes:

- Monitors:** A table listing connection names and system types.
 

Status	Connection Name	System Type
Judge	Juquean2	IBM LoadLewer (Blue Gene)
Judge	Jurpa	TORQUE Resource Manager
Judge	Localhost	TORQUE Resource Manager
- Active Jobs:** A table listing job IDs, owners, queues, wall times, and queue dates.
 

slip	owner	queue	wall	queue date	
338561	judgem	neuhais	common	86400	2012-09-11 07:44:10
338564	judgem	neuhais	common	86400	2012-09-11 07:44:16
338569	judgem	neuhais	common	86400	2012-09-11 07:44:26
338572	judgem	neuhais	common	86400	2012-09-11 07:44:32
338573	judgem	neuhais	common	86400	2012-09-11 07:44:35
338575	judgem	neuhais	common	86400	2012-09-11 07:53:00
338578	judgem	neuhais	common	86400	2012-09-11 07:53:06
338579	judgem	neuhais	common	86400	2012-09-11 07:53:08
338580	judgem	neuhais	common	86400	2012-09-11 07:53:10
338582	judgem	neuhais	common	86400	2012-09-11 07:53:14
338588	judgem	neuhais	common	86400	2012-09-11 07:53:26
338589	judgem	neuhais	common	86400	2012-09-11 07:53:28
338590	judgem	neuhais	common	86400	2012-09-11 07:53:30
338591	judgem	neuhais	common	86400	2012-09-11 07:53:32
338592	judgem	neuhais	common	86400	2012-09-11 07:53:34
338593	judgem	neuhais	common	86400	2012-09-11 07:53:36
338594	judgem	neuhais	common	86400	2012-09-11 08:10:42
338595	judgem	neuhais	common	86400	2012-09-11 08:10:44
338417	judgem	slipq710	common	18000	2012-09-11 13:24:51
- Messages:** A console window showing system messages, including a welcome message and information about the system, user documentation, and GPU throttling status.
- Resource Usage Grids:** Multiple small grid charts showing resource usage for various components like 'judgem' and 'slipq710'.

# Layout Example – Custom II



The screenshot displays a software interface for system monitoring, likely on a Linux-based system. The interface is divided into several panes:

- Monitors List:** A table showing connection names and system types.
 

Status	Connection Name	System Type
	Juqum2	EM Lead_Levier (Blue Gene)
	Juqum	TORQUE Resource Manager
	Localhost	TORQUE Resource Manager
- Active Jobs List:** A table listing active jobs with columns for job ID, owner, queue, wall time, and queue date.
 

jobid	owner	queue	wall	queue date	
338581	judgen	neuhdas	common	86400	2012-09-11 07:44:10
338584	judgen	neuhdas	common	86400	2012-09-11 07:44:16
338589	judgen	neuhdas	common	86400	2012-09-11 07:44:26
338576	judgen	neuhdas	common	86400	2012-09-11 07:59:09
338579	judgen	neuhdas	common	86400	2012-09-11 07:59:05
338582	judgen	neuhdas	common	86400	2012-09-11 07:59:08
338583	judgen	neuhdas	common	86400	2012-09-11 07:59:14
338608	judgen	neuhdas	common	86400	2012-09-11 07:59:26
338609	judgen	neuhdas	common	86400	2012-09-11 07:59:28
338580	judgen	neuhdas	common	86400	2012-09-11 07:59:30
338591	judgen	neuhdas	common	86400	2012-09-11 07:59:32
338592	judgen	neuhdas	common	86400	2012-09-11 07:59:34
338593	judgen	neuhdas	common	86400	2012-09-11 07:59:36
338594	judgen	neuhdas	common	86400	2012-09-11 08:10:42
338606	judgen	neuhdas	common	86400	2012-09-11 08:10:42
- Messages:** A log pane showing system status messages, including information about system changes, user documentation, and GPU configuration updates.
- Job Status Grid:** A large grid of 80 small monitors (labeled judgn000 to judgn079) arranged in 8 rows and 10 columns. Each monitor displays a visual representation of job progress or resource usage, with colored bars indicating different components.



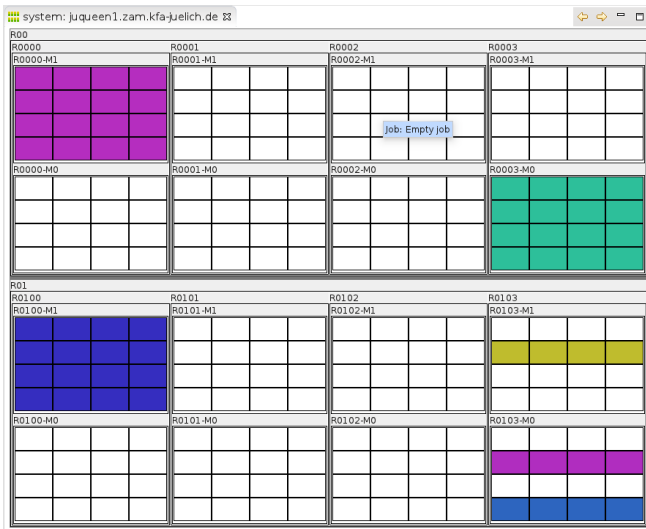
# Part VII: Conclusion

September 19, 2012 | Carsten Karbach and Wolfgang Frings

## Conclusion

- 5 interacting views form the **monitoring perspective**
- **LML** as data interface between LML\_da and client
- LML is divided into global data, graphical components and layout
- composition of 6 Eclipse plug-ins (lml.core/ui, monitor.core/ui, lml.da/lml.da.server)
- **new target system:**  
write data extraction scripts + configure layout
- Scalability
  - scalable data **acquisition**
  - scalable data **format**
  - scalable **presentation**

# Questions?



## Contact

- **E-mail:**  
c.karbach@fz-juelich.de, w.frings@fz-juelich.de
- **LML** → <http://llview.zam.kfa-juelich.de/LML>
- **LLview** → <http://www.fz-juelich.de/jsc/llview>