General openMosix Daemon

Gian Paolo Ghilardi
Matthias Rechemburg
GOMD TEAM

What’s GOMD?

- GOMD stands for " **general openMosix daemon**".
- GOMD is a daemon which executes commands and gets information from the nodes of an openMosix cluster. It has to run on every node in order to collect data, and it waits for commands to execute.
- The purpose of the project is to provide:
  - an integrated way to collect cluster statistics;
  - a simple and well-defined format for statistics and data;
  - easy-to-use APIs to receive raw or already-formatted statistics;
  - new tools and/or mechanisms (i.e. the SCX);
  - support for new interesting openMosix tools.
  - and much more… ☺
- However our primary goal is to serve the openMosix community. ;)

Features (1/3)

- **Access Control List (ACL)**
  This feature allows you to specify:
  - allowed or blocked IPs/subnets
  - what they can do (permissions)
- **clusterSnapshot (CS)**
  This feature collects statistics/informations of all available gomd-enabled openMosix nodes. When enabled, this facility, generates an HTML file with the summary of the cluster, node-by-node.

Features (2/3)

- **Remote Gomd Searcher (RGS)**
  Known also as Autodiscovery.
  The daemon is able to search other daemons.
  Found daemons are cached in a special cache file.

- **Secure Cluster-wide (command) eXecution (SCX)**
  This feature allows you to execute a command on the whole cluster. Obviously you need a daemon running on each openMosix node. ☺
  You can specify exactly the allowed commands but you can also set a MACRO to allow or block all of the commands (this option is also known as "setting the default behaviour"). If you want, you can execute commands also on the local node.

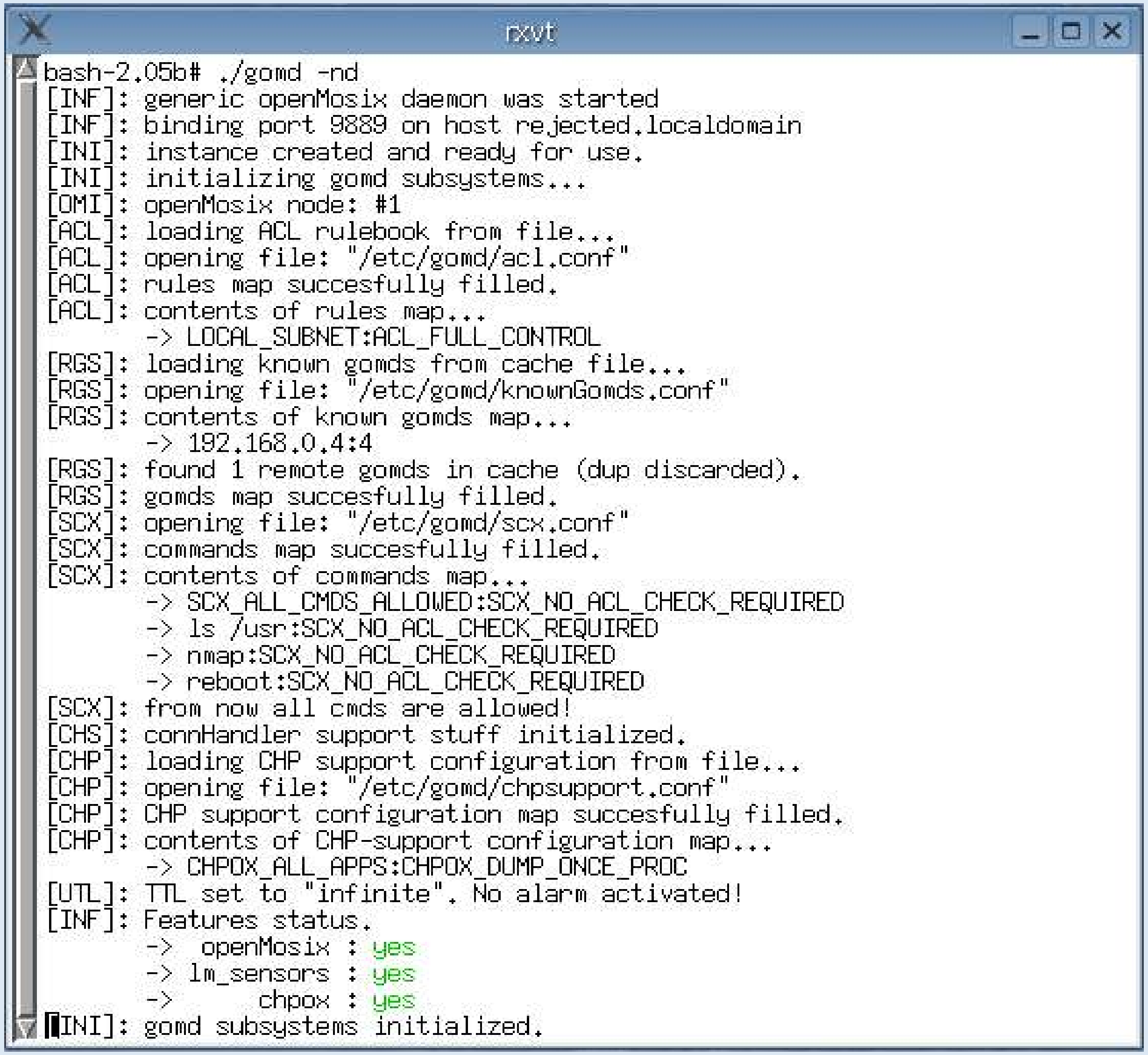
Features (3/3)

Experimental stuff: support for external programs.

- **Chpox Support (CHP)**
  This feature allows you to register and dump a process via the nice Chpox program.

- **LM_sensors Support (LMS)**
  This feature allows you to get the status of some hardware sensors including CPU temperature, fan speed,…

How it appears...



```
bash-2.05b# ./gomd -nd
[INF]: generic openMosix daemon was started
[INF]: binding port 9889 on host rejected.localdomain
[INI]: instance created and ready for use.
[INI]: initializing gomd subsystems...
[OMI]: openMosix node: #1
[ACL]: loading ACL rulebook from file...
[ACL]: opening file: "/etc/gomd/acl.conf"
[ACL]: rules map succesfully filled.
[ACL]: contents of rules map...
       -> LOCAL_SUBNET:ACL_FULL_CONTROL
[RGS]: loading known gomds from cache file...
[RGS]: opening file: "/etc/gomd/knownGomds.conf"
[RGS]: contents of known gomds map...
       -> 192.168.0.4:4
[RGS]: found 1 remote gomds in cache (dup discarded).
[RGS]: gomds map succesfully filled.
[SCX]: opening file: "/etc/gomd/scx.conf"
[SCX]: commands map succesfully filled.
[SCX]: contents of commands map...
       -> SCX_ALL_CMDS_ALLOWED:SCX_NO_ACL_CHECK_REQUIRED
       -> ls /usr:SCX_NO_ACL_CHECK_REQUIRED
       -> nmap:SCX_NO_ACL_CHECK_REQUIRED
       -> reboot:SCX_NO_ACL_CHECK_REQUIRED
[SCX]: from now all cmds are allowed!
[CHS]: connHandler support stuff initialized.
[CHP]: loading CHP support configuration from file...
[CHP]: opening file: "/etc/gomd/chpsupport.conf"
[CHP]: CHP support configuration map succesfully filled.
[CHP]: contents of CHP-support configuration map...
       -> CHPOX_ALL_APPS:CHPOX_DUMP_ONCE_PROC
[UTL]: TTL set to "infinite". No alarm activated!
[INF]: Features status.
       ->  openMosix : yes
       -> lm_sensors : yes
       ->      chpox : yes
[INI]: gomd subsystems initialized.
```

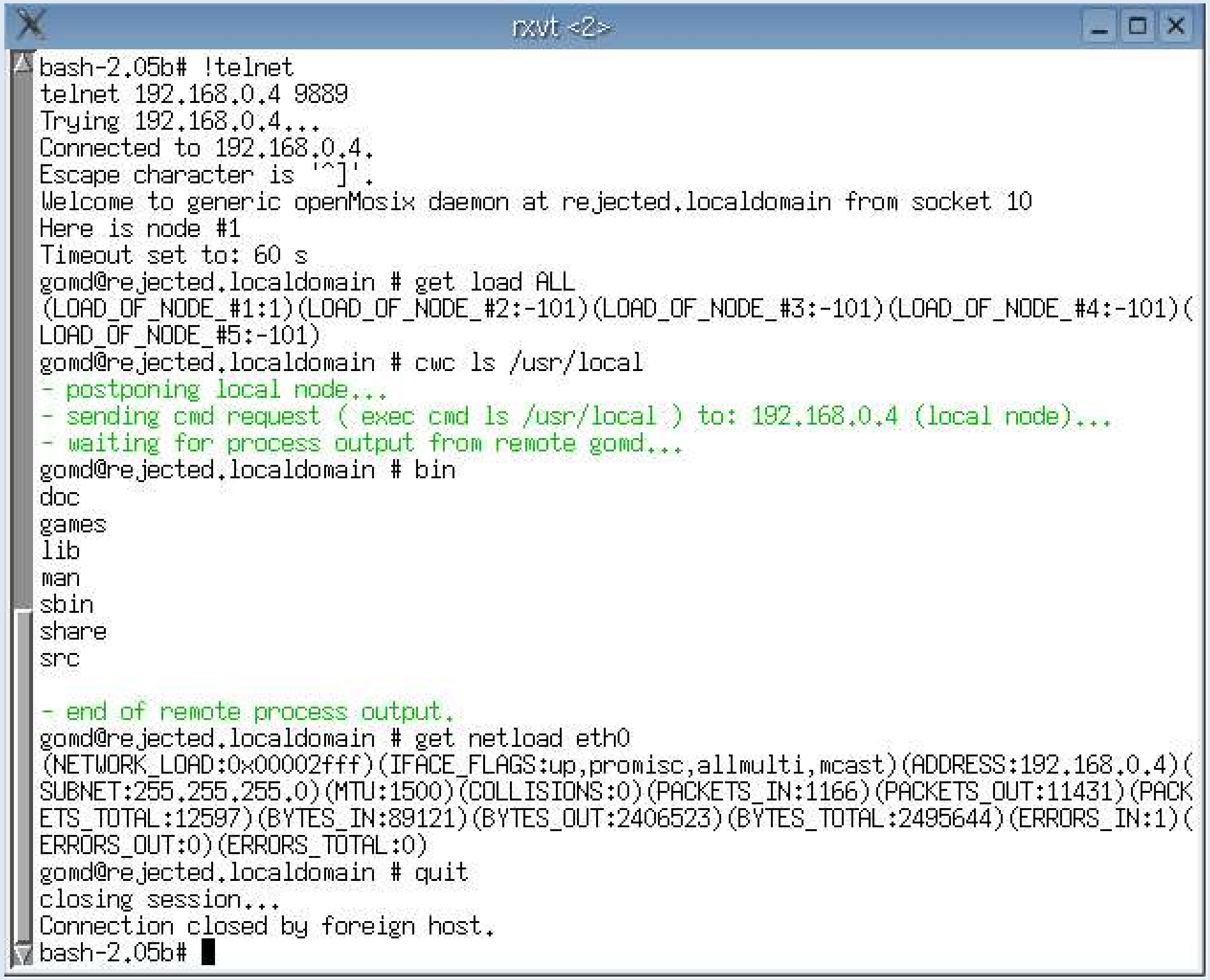
This is how the daemon appears at startup.

You can see the status of each loaded subsystem.

LEGEND:

INF	Debug informations
ACL	Access Control List
RGS	Remote Gomd Searcher
SCX	Secure Cluster-wide (command) eXecution
CHP	CHPOX Support

...and how to contact it.



```
bash-2.05b# !telnet
telnet 192.168.0.4 9889
Trying 192.168.0.4...
Connected to 192.168.0.4.
Escape character is '^]'.
Welcome to generic openMosix daemon at rejected.localdomain from socket 10
Here is node #1
Timeout set to: 60 s
gomd@rejected.localdomain # get load ALL
(LOAD_OF_NODE_#1:1)(LOAD_OF_NODE_#2:-101)(LOAD_OF_NODE_#3:-101)(LOAD_OF_NODE_#4:-101)(
LOAD_OF_NODE_#5:-101)
gomd@rejected.localdomain # cwc ls /usr/local
- postponing local node...
- sending cmd request ( exec cmd ls /usr/local ) to: 192.168.0.4 (local node)...
- waiting for process output from remote gomd...
gomd@rejected.localdomain # bin
doc
games
lib
man
sbin
share
src

- end of remote process output.
gomd@rejected.localdomain # get netload eth0
(NETWORK_LOAD:0x00002fff)(IFACE_FLAGS:up,promisc,allmulti,mcast)(ADDRESS:192.168.0.4)(
SUBNET:255.255.255.0)(MTU:1500)(COLLISIONS:0)(PACKETS_IN:1166)(PACKETS_OUT:11431)(PACK
ETS_TOTAL:12597)(BYTES_IN:89121)(BYTES_OUT:2406523)(BYTES_TOTAL:2495644)(ERRORS_IN:1)(
ERRORS_OUT:0)(ERRORS_TOTAL:0)
gomd@rejected.localdomain # quit
closing session...
Connection closed by foreign host.
bash-2.05b#
```

You can interact with the daemon via a standard telnet session.

In the image on the left, you can see the execution of some standard commands, including a call to the SCX facility to propagate a command (a simple “ls /usr/local”) to the whole cluster.

The daemon can provide some extra informations via the excellent Libgtop library (optional).

Libgomd

- With the daemon, the GOMD project provides a simple library named libgomd that provides some handy functions to contact a remote gomd daemon and receive the answers.

- The library can be used by both C and C++ programmers. This library actually exposes:
  - some quick functions for C users
  - an handy class for C++ programmers

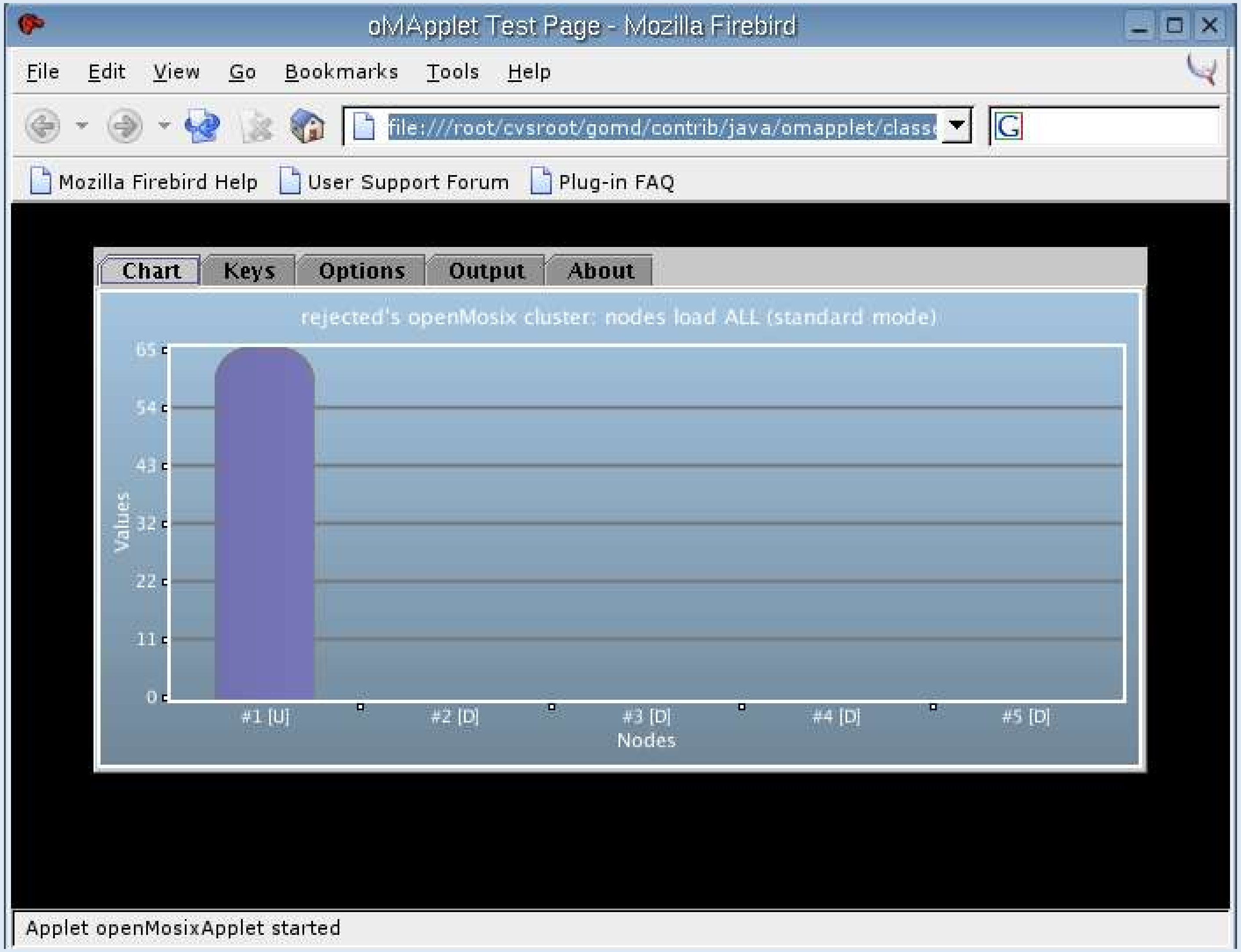
- The daemon itself uses this library for daemon-to-daemon communication (for statistics/informations exchange,…). ☺

Subprojects (Contrib)

- The key idea for the gomd project is to assure bindings for all the languages.

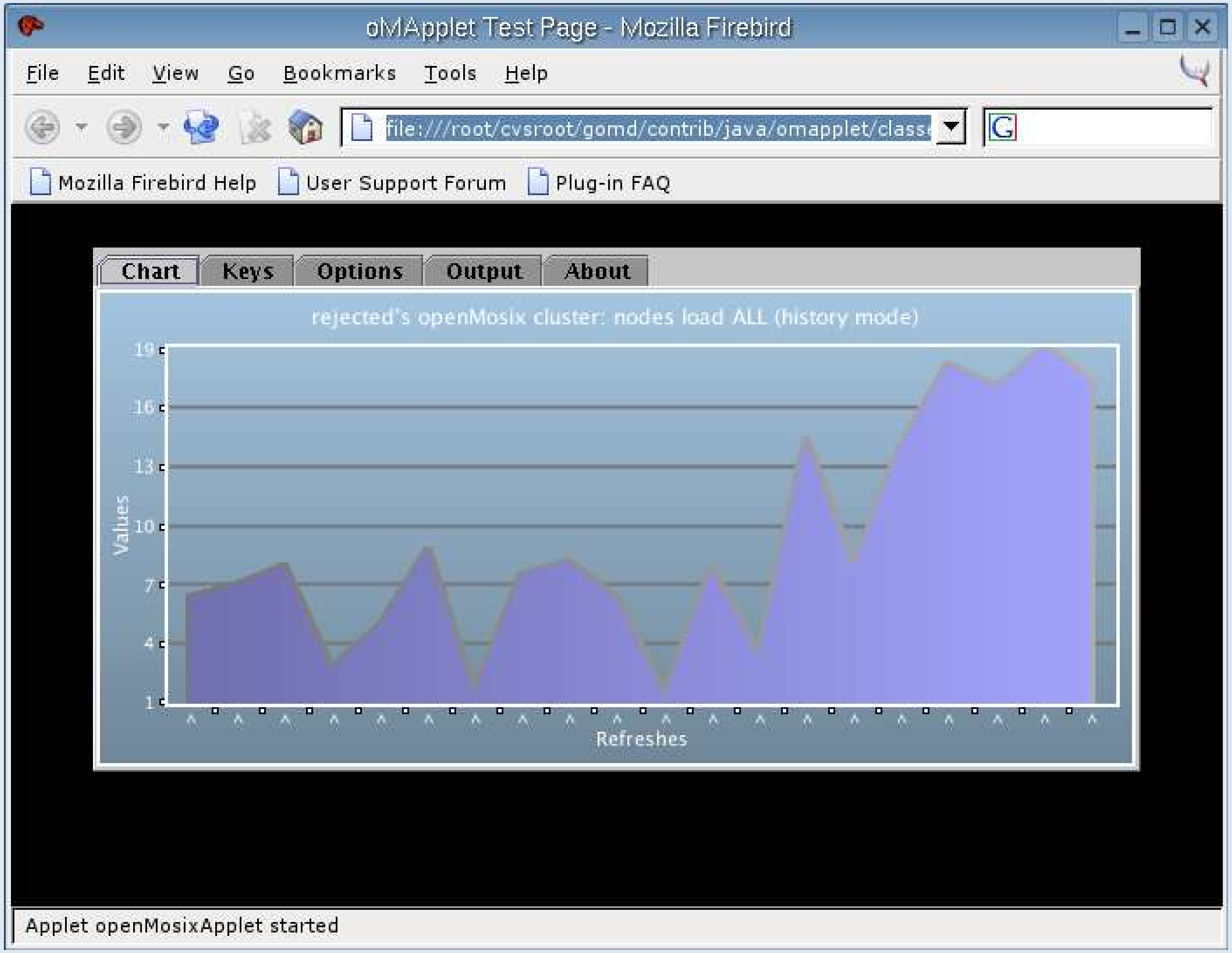
- Actually we've 4 active sub-projects for this purpose:
  - java2gomd (maintainer: roeles)
  - perl2gomd (maintainer: CaloRE)
  - oracle2gomd (maintainer: CaloRE) ☺
  - openMosixApplet (maintainer: JP)

openMosixApplet


oMApplet Test Page - Mozilla Firebird
File Edit View Go Bookmarks Tools Help
file:///root/cvsroot/gomd/contrib/java/omapplet/class
Mozilla Firebird Help
User Support Forum
Plug-in FAQ
Chart
Keys
Options
Output
About
rejected's openMosix cluster: nodes load ALL (standard mode)
Values
65
54
43
32
22
11
0
#1 [U]
#2 [D]
#3 [D]
#4 [D]
#5 [D]
Nodes
Applet openMosixApplet started

This is the port for GOMD of the openMosixApplet.
This Java2 applet uses oustanding Chart2D library to monitor the status of an openMosix cluster, in REAL TIME! This is the real-time ("standard") mode...

openMosixApplet


oMApplet Test Page - Mozilla Firebird
File Edit View Go Bookmarks Tools Help
file:///root/cvsroot/gomd/contrib/java/omapplet/class
Mozilla Firebird Help
User Support Forum
Plug-in FAQ
Chart
Keys
Options
Output
About
rejected's openMosix cluster: nodes load ALL (history mode)
Values
Refreshes
Applet openMosixApplet started

...while this is the history modes (values are averages of all nodes).
The applet supports all basic openMosix infos (cpus, loads, memory status,...).
It's provided in a single JAR file for easy deployment.