GPU server monitoring package documentation

SE Team 10

January 17, 2018

1 What is it?

This package has one main goal: Give users insight into jobs running on the GPU servers at LIACS. This goal is achieved in two ways: 1) It provides a quick and clean view of the running jobs on the GPU's and 2) It allows teachers to automatically notify users of running processes according to adjustable rule sets.

The package consists of three separate parts.
The GPUMonitor gathers the data of the GPU hardware and the processes using them and provides an interface for the other parts of the package to access this data quickly. A single instance of the GPUMonitor is meant to be running at all times on each GPU server.

The GPUView provides a human readable interface in the shell for a selection of the data gathered by the GPUMonitor. Any user on the server should be able to run the GPUView.

The Violation Detector allows teachers to set notification rules for processes utilizing GPU hardware. It enables teachers to notify users of long running jobs, jobs running on multiple GPU's and it allows teachers to ask users to keep the servers free for students enrolled for a specific course.

2 How to use?

2.1 Setup

The package source files should be placed in a location accessible by one of the teachers in order to use the GPUMonitor and the Violation Detector. Preferably the GPUView files should be executable by students from a shared folder.

2.1.1 GPUMonitor

The GPUMonitor needs to be run once and keeps running in the background. The monitor places a socket in the /tmp/ folder providing access to the gathered data through the MonitorClient. Running the GPUMonitor can be done from any account. Trying to run multiple instances of the monitor will result in an error. Starting the monitor can be done by running the /bin/monitor shell script.

2.1.2 Violation Detector

The Violation Detector is preferably run by one user, although it is possible to run multiple instances. It can be run from the /violation_detector_env/ folder in order to access all required files.
The Violation detector should run right of the bat, but in order to configure all the rules properly see 'Teacher use case'.

2.2 Student use case

The student interacts with the system in two separate ways. In the first place the student can request the GPU utilization information. This is done by simply calling the /bin/gpuview shell script. The second

form of interaction is simply receiving e-mail notifications about rule violations and does not require more detailed instructions.

2.3 Teacher use case

The interaction between a Teacher and the system is a little more complex. In general a Teacher sets RULES applying to GROUPS. When a user violates a rule a TEMPLATEd e-mail notification is sent to this user.
RULES, GROUPS and mail TEMPLATES are stored in files. Understanding the syntax of these files enables a teacher to do everything the package is intended to do.
RULES text file, default location: /violation_detector_env/rules.txt
syntax:

```
<rule_type>     <group>     <mail_template>     [parameters]
```

`<rule_type>`

one of the following:
RESERVE
- reserve the server for specified group, all other users are notified when starting a job.
PROC_TIME
- users running jobs longer than the specified time are notified.
Time in seconds must be specified in [parameters].
IDLE_TIME
- Same as PROC_TIME but applying to idling time.
MAX_CLAIMED_GPUS
- users running jobs claiming more than the specified number of devices are notified. Nr of devices must be specified in [parameters].

`<group>`

There are roughly two kinds of groups. Default groups and custom groups.
Default groups: (mostly self explanatory)
NO_ONE
EVERYONE
STUDENTS - all usernames that consist of a lowercase 's' followed by any number of digits (eg. s1122334)
NOT_STUDENTS

Custom groups: Custom groups can be defined in a separate file (see GROUPS). Custom groups can be used to reserve server for a specific course.
Multiple groups can be separated by commas.

`<mail_template>`

Should refer to a .template file in the mailtemplates folder. The .template extension can be omitted. For template syntax see TEMPLATES.
Lines in the rules file starting with '#' are not parsed. Whitespaces are ignored. Example:

```
# notify everyone not in PRACTICAL1 when starting jobs with
# the message template all_in_one.
RESERVE             PRACTICAL1  all_in_one
# notify every user running a job for longer than two minutes
PROC_TIME           EVERYONE    all_in_one                  120
# notify ELGAR and ROB when idling longer than 20 seconds
IDLE_TIME           ELGAR,ROB   all_in_one                  20
# notify every user running a job on 2 or more GPU's
MAX_CLAIMED_GPUS    EVERYONE    all_in_one                  2
```

GROUPS text file, default location: /violation_detector_env/groups.txt
syntax:

```
<groupname>: <users_comma_separated>;
```

Linebreaks and whitespaces are ignored, therefore usernames can be either on separate lines or on the same lines. Groups are closed by a semicolon.
example:

```
PRACTICAL1: s0000000, s0000004, s0000009;
```

is equivalent to:

```
PRACTICAL1:
s0000000,
s0000004,
s0000009;
```

TEMPLATES folder containing .template files, default location: /violation_detector_env/mailtemplates
.template files are files in the following format:

```
From: [FROM_ADDR]
To: [TO_ADDR]
Subject: subject

body
```

keywords between square brackets are replaced by the mailer in order to fit the circumstance. The following keywords can be used: [FROM_ADDR] - sender [TO_ADDR] - receiver/violator [FULLNAME] - full name of violator [SERVER] - server name [TIME] - Current time (moment of sending e-mail)
Example:

```
From: [FROM_ADDR]
To: [TO_ADDR]
Subject: Misuse [SERVER]

Dear [FULLNAME],

At [TIME] you violated a rule by running a job on [SERVER].
This mail was meant for: [TO_ADDR]

Best,
TA

This an automatically generated e-mail.
For comments or concerns please contact [FROM_ADDR].
```