

CESNET Technical Report x/2008
**Perfmon and Servmon: Monitoring
operational status and resources of
distributed computing systems**

Aleš Friedl, Sven Ubik

November 16, 2008

Keywords: Performance monitoring, Distributed systems, Resource monitoring

1 Introduction

As part of our network monitoring activities, we deployed extensive monitoring systems in our network CESNET2 and in the European academic network GN2 [1]. These systems use active, passive and infrastructure monitoring. They combine multiple tools and monitoring architectures. The infrastructure consists of many hardware and software components. These include PC servers, monitoring cards, operating systems, drivers, libraries, middleware and application software. In such a complex distributed infrastructure, component failures are inevitable. It can be due to hardware failure, resource exhaustion, unexpected network changes, or even software bugs, which unfortunately happen.

In order to secure reliable operation of our monitoring infrastructure, we have developed two distributed service applications named Perfmon and Servmon. Perfmon monitors operational state of components in a distributed hardware & software system and reacts on detected problems. Servmon monitors resource consumption. In this report we describe architecture of these two applications in more detail.

2 Perfmon

The purpose of Perfmon is to support the reliable operation of a distributed hardware & software infrastructure. We use it to check operation of our monitoring system in the CESNET2 and GN2 networks.

2.1 Operation

Perfmon consists of agents running on each monitoring station and a central program with the user interface running on the central station. Perfmon is activated periodically at specified intervals and executes the following types of tests:

- Hardware reachability of monitoring stations (checked from the central station)
- Running status of the specified set of processes (checked by agents)
- Services respond to stimuli correctly (checked by agents or from the central station)

Each test checks one component. Upon detection of a problem, Perfmon executes one or more of the following actions:

- Logs event to a file
- Sends email to specified addresses
- Automatically restarts services

Components often depend on one another. Perfmon can be configured to check or restart services and processes required for another service or process to operate correctly. These dependences can form an acyclic directed graph. Perfmon can be easily extended to check additional services by user-supplied scripts with common interface. A tested component can be in one of the three states:

- Green - operating correctly
- Red - failure
- Yellow - operating correctly, but a failure occurred within the last 24 hours

Aggregated states can be defined. For instance, each monitoring station has a "software state", which is green when all software components are green and red or yellow when one software component is red or yellow.

2.2 Usage Examples

The example figures in this section are for the pilot deployment in part of the GN2 network. Perfmon provides a map-based user interface, see the left part of Figure 1. Each monitoring station is represented by a coloured circle divided into two halves. The left half represents the hardware state (reachability) and the right half the aggregated software state. This enables a quick overview of the current or recent problems on all monitoring stations. On placing the cursor on a monitoring station, an information box opens, which displays the state of individual components, see the right part of Figure 1. Clicking inside this box displays a list of recent events on that monitoring station, see Figure 2. Below the map, Perfmon also displays the state of individual components in a tabular form, see Figure 3.

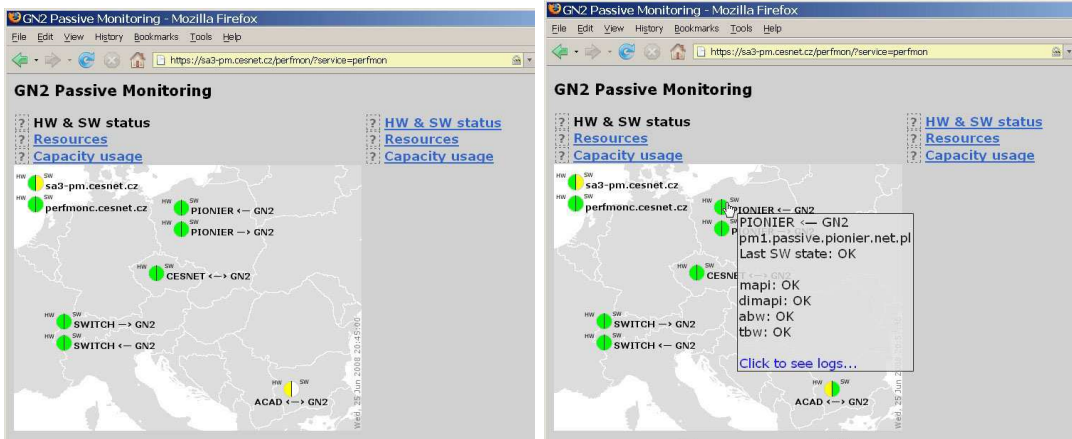


Figure 1: User interface of Perfmon application

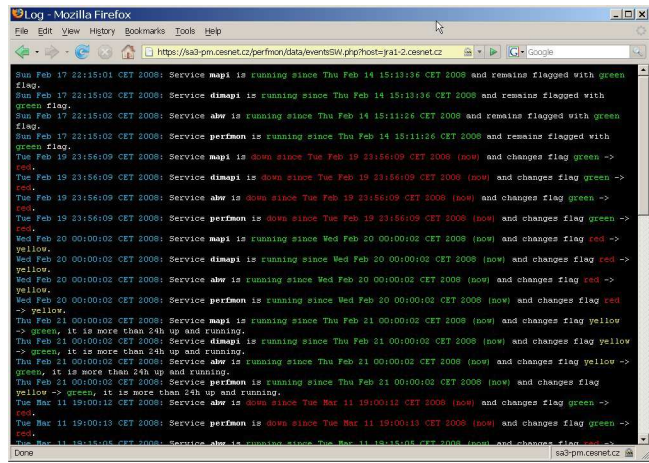


Figure 2: History of test events found by Perfmon

	HW		SW				
	ping	ping6	mapi	dimapi	abw	tbw	packetlos
PIONIER (← GN2)	ping		mapi	dimapi	abw	tbw	
PIONIER (→ GN2)	ping		mapi	dimapi	abw	tbw	
CESNET (← GN2)	ping		mapi	dimapi	abw		
SWITCH (→ GN2)	ping		mapi	dimapi	abw	tbw	
SWITCH (← GN2)	ping		mapi	dimapi	abw	tbw	
ACAD (→ GN2)	ping						
sa3-pm							packetlos
perfmonc	ping						

	Service is UP		Service is UP, but was down within last 24 hours		Service is DOWN		Service state is N/A
--	---------------	--	--	--	-----------------	--	----------------------

Figure 3: Table of component states provided by Perfmon

2.3 Summary

The Perfmon service application provides the following benefits:

- Regular hardware reachability checks
- Regular operation checks for various software components of distributed monitoring applications
- Configurable actions upon detecting problems, including logging, sending emails and automatic restarts
- Configurable component dependencies for checks and actions
- Map-based user interface for a quick overview of the system status with detailed reports on request
- Easily extendable with user-defined tests
- Significant improvement of reliability of a complex distributed hardware & software system

3 Servmon

3.1 Operation

In our network monitoring, we use a combination of active tools which use test packets, passive tools which observe directly existing traffic and infrastructure tools which collect data from network infrastructure, such as routers and switches.

Advantage of passive monitoring is that it can provide almost arbitrary information about real network traffic and about transport characteristics that this traffic experiences. Results of active monitoring are only truly applicable to test packets. Many useful characteristics can only be obtained by passive monitoring, which is however much more compute intensive, because we need to process all network traffic, rather than just a few test packets. Some applications can accept sampling of monitored traffic, such as traffic volume monitoring. Other applications must process all packets, such as passive monitoring of packet loss in real traffic.

The DiMAPI middleware [2] used in our architecture permits multiple applications to share the same monitoring stations and monitoring cards. This of course further increases resource consumption on monitoring stations. Modern PCs are very powerful hardware with CPU, memory and bus bandwidth sufficient for quite complex 10 Gb/s monitoring [3]. Nevertheless, to make sure that applications operate correctly, resource consumption on monitoring stations need to be observed and actions must be taken when defined thresholds are exceeded. This can happen, for example, when a new application is introduced or when a problem in an application causes excessive resource consumption.

The Servmon application was developed and deployed to deal with this problem. Like Perfmon, it consists of agents running on each monitoring station and a central unit with the user interface running on the central station. Servmon runs continuously and checks

the following resources on all monitoring stations in short intervals (currently every 10 seconds):

- CPU usage separately for each CPU core and each type of usage (system, user, nice, iowait, irq, softirq)
- CPU load (a number of active processes)
- Shared memory usage
- Monitoring card packet drops (not a resource as such, but an important consequence of insufficient resources)

In addition to CPU utilisation, we also monitor CPU load [4], which is the number of processes waiting to be served by a CPU.

Shared memory is an important communication channel used by DiMAPI. When an application does not behave properly, it may consume an increasing number of shared memory segments.

When DiMAPI and applications cannot keep up with the volume of traffic, the circular packet buffer written by the monitoring card can fill up and packets are lost. The monitoring card driver provides information how many packets were lost due to buffer overflow.

3.2 Usage Examples

Servmon provides two types of user interface. A map-based user interface offers a quick overview of resource consumption on all monitoring stations, see the left part of Figure 4. In addition to all monitoring stations, Servmon is configured to monitor resources also on several central stations for applications, user interface and Lookup Service (LS) [5], which registers locations of distributed services.

Each station is represented by a coloured circle consisting of three segments which represent:

- CPU usage (sum over all CPU cores)
- Shared memory usage
- DAG card packet drops

The colour of each segment has a meaning:

- Green - resource consumption is under threshold
- Red - resource consumption exceeds threshold
- Yellow - resource consumption is under threshold, but it exceeded threshold in the past 24 hours

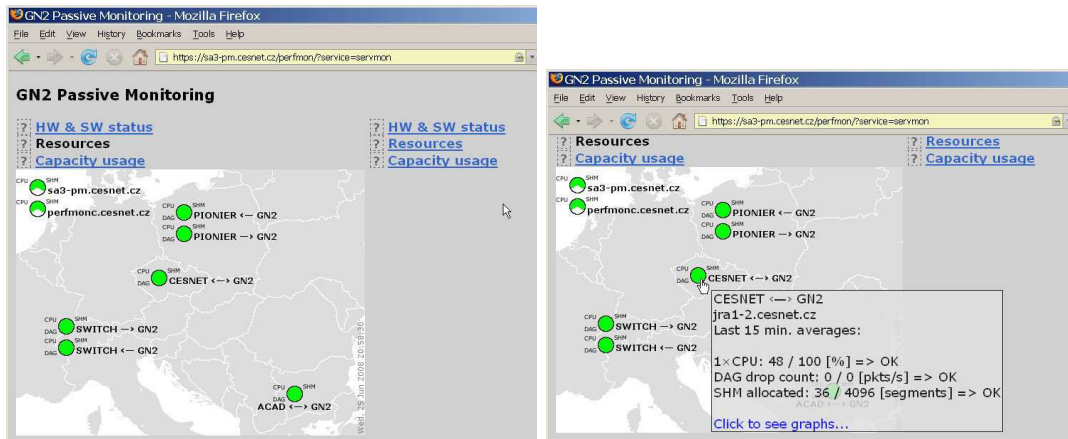


Figure 4: Map-based user interface of Servmon

If the cursor is placed on a monitoring station, an information box displays more detailed resource consumption over the last 15 minutes, see the right part of Figure 4.

Servmon also provides a form which allows users to request detailed graphs of resource consumption. Graphs can be plotted for specified monitoring stations, characteristics, time periods and time resolutions.

The left part of Figure 5 shows, for instance, the CPU usage on the monitoring station on the PIONIER-GN2 link for a one day period with 5-minute resolution and maximum values for 1-hour intervals. Each CPU core is monitored separately. Different colors indicate different types of CPU usage (system, user, etc.). In this case most CPU power was used in the user phase. This is a healthy condition, enabled by using a monitoring card, which does not use CPU power to copy packets from the network to the PC memory. With regular Ethernet adapters, a lot of CPU power is used in the system phase just to copy packets.

Periodic activities of some applications tend to increase CPU load (active processes) temporarily, see Figure 6.

The shared memory allocation over the same period is shown in Figure 7. Only a few segments were allocated. The example graph of monitoring card packets drops is shown Figure 8. A large number of dropped packets can be reported when monitoring applications stop for some reason and packets are not read from the card circular buffer. In the shown example, two singular packet drops happened during the day, probably due to short-term CPU overloads. The graph shows the average number of dropped packets in 10-second counter readouts over 5-minute intervals. Since only one packet was dropped in one 10-second readout, it results in the 5-minute average much less than one, hence the "milipacket" scale on the Y axis.

3.3 Summary

The Servmon service application provides the following benefits:

- Regular resource consumption checks (CPU usage per CPU core and per type of usage, CPU load queue, shared memory usage and monitoring card packet drops)

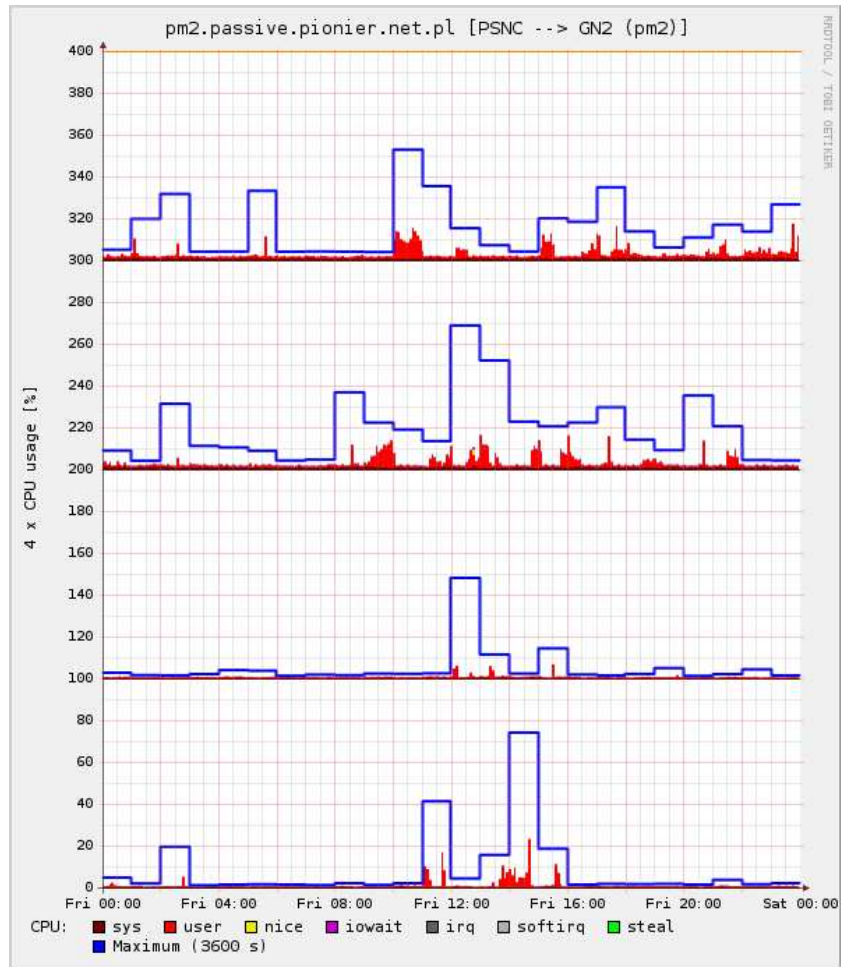


Figure 5: CPU utilisation monitoring

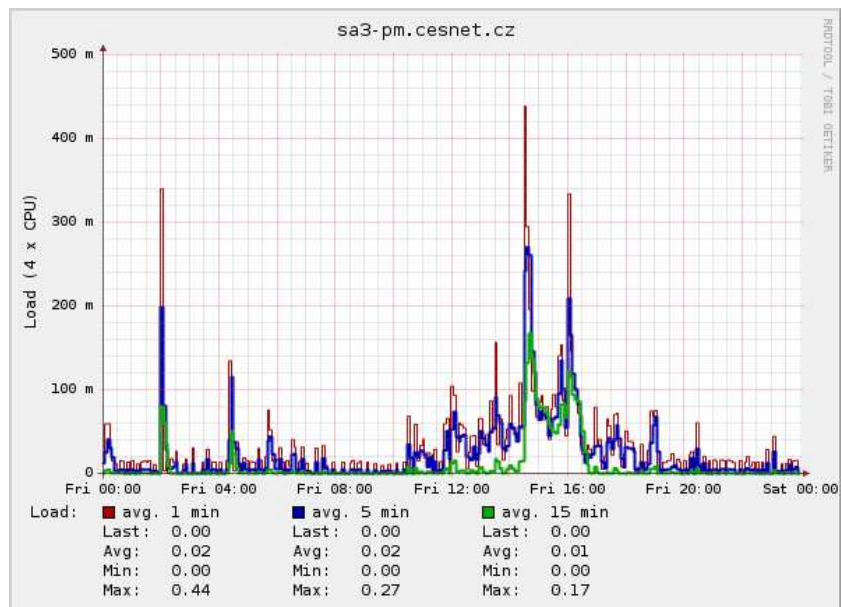


Figure 6: CPU load monitoring

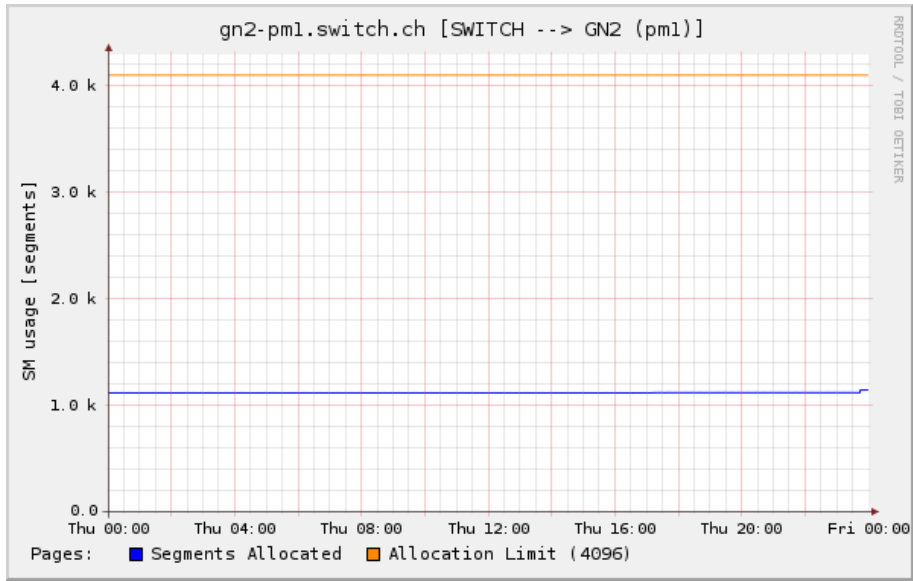


Figure 7: Shared memory usage monitoring

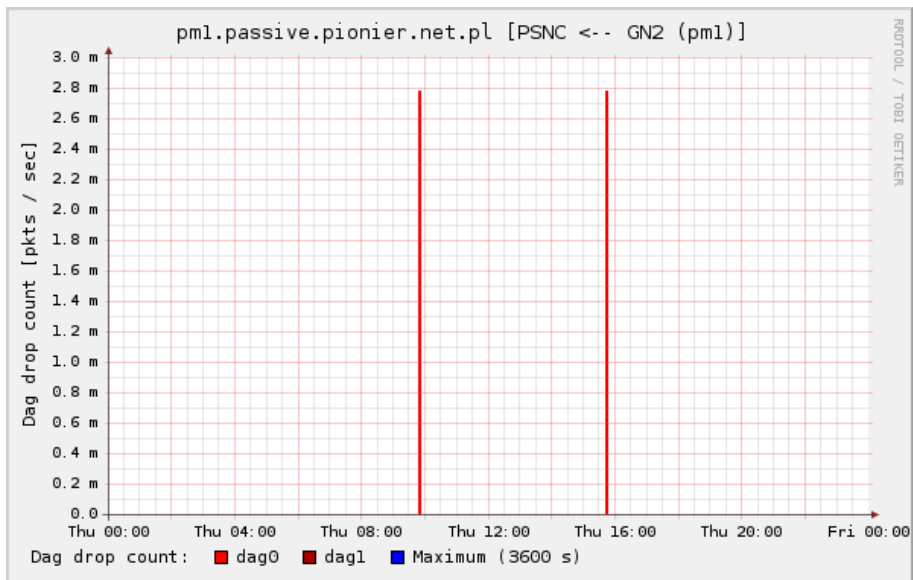


Figure 8: Monitoring card packet drops

- Map-based user interface for a quick overview of the resource status and a form to request detailed resource consumption graphs for specified time periods
- Extendable for monitoring of more resources
- Can be used with Perfmon to increase reliability of a complex distributed hardware & software system

4 Related Tools

Various CPU utilization monitoring tools exist, particularly for the MRTG (Multi Router Traffic Grapher) [6] framework. The UNIX utility *top* [7] is commonly used to observe CPU utilization. We use the same mechanism as *top* does, that is reading CPU ticks from the */proc/stats* file. We combine CPU utilization with CPU load, shared memory usage and monitoring card packet drops and present all characteristics in a common user interface. CPU utilization is presented in such a way that overload of a single CPU core is clearly observable. For general computing applications temporary overload of some CPU cores just increases time to completion and may be even a desirable condition (we fully utilize the CPU). For monitoring applications, which must process live stream of packets without drops, it is a wrong condition resulting in packet loss.

An industry standard package for status and resources monitoring is Nagios [8]. It provides a large number of features and options. On the other hand, the exact characteristics and their presentation that we required are not by default available in Nagios. Since we would need to develop plugins for Nagios that would provide them, we decided to use our own framework for status and resource monitoring, which is simpler and very easy to configure and use.

References

- [1] Deliverable MS.3.7.5: Report on Passive Monitoring Pilot, SA3 activity, GN2 project, August 2008.
- [2] DiMAPI - Distributed Monitoring Application Programmable Interface, <http://mapi.uninett.no>.
- [3] S. Ubik, P. Žejdl, *Passive monitoring of 10 Gb/s lines with PC hardware*, TNC2008, Bruges, May 2008.
- [4] Neil Gunthe. *UNIX Load Average*, <http://www.teamquest.com/resources/gunther/ldavg1.shtml>.
- [5] LS - perfSONAR Lookup Service, http://wiki.perfsonar.net/jra1-wiki/index.php/Lookup_Service.
- [6] Tobi Oetiker. *MRTG - Multi Router Traffic Grapher*, <http://oss.oetiker.ch/mrtg>.
- [7] TOP Linux utility and */proc* file system, <http://procps.sourceforge.net>.
- [8] Nagios, <http://www.nagios.org>.