Volumetric and capacity studies

Volumetric and capacity studies

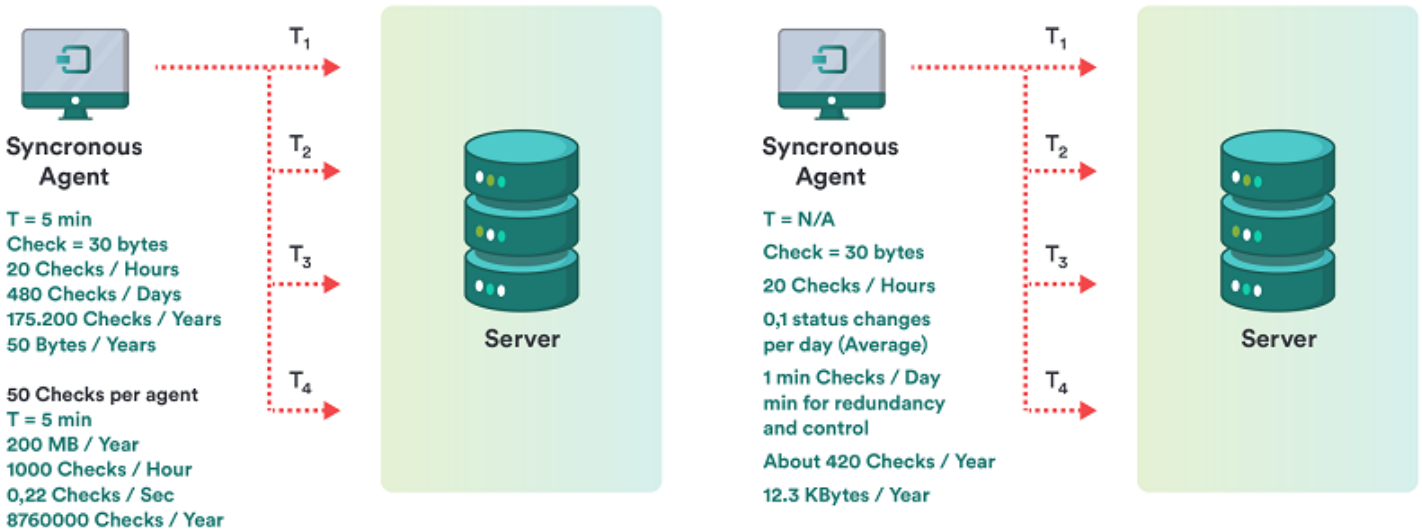
Introduction

Pandora FMS is a complex distributed application that has different key elements, susceptible to represent a bottleneck if it is not sized and configured correctly. The purpose of this chapter is to help to carry out a capacity study, to analyze the *scalability* of Pandora FMS according to a specific set of parameters. This study will help to find out the requirements that the installation should have to be able to support a certain capacity.

Load tests are also used to see the maximum capacity per server. In the current architecture model (version 3.0 or later), with "N" independent servers and a Command Center (Metaconsole) installed, this *scalability* tends to be of linear order, while *scalability* based on centralized models is exponential.

Data storage and compaction

The fact that Pandora FMS compacts data in real time is very relevant in order to calculate the size that they are going to occupy. An initial study was made comparing the way of storing the data of a classic system to the "asynchronous" way of storing Pandora FMS data. This can be seen in the diagram included in this chapter.



In a conventional system

For one check, with an average of 20 repetitions per day, there is a total of 5 MB per year in

occupied space. For 50 checks per agent, that's 250 MB per year.

In a non-conventional system, asynchronous or with compaction, such as Pandora FMS.

For one check, with an average of 0.1 variations per day, you have a total of 12.3 KB per year in occupied space. For 50 checks per agent, that is 615 KB per year.

Specific terminology

A glossary of terms specific to this study is described below:

- Information fragmentation: The information that Pandora FMS manages may have different performance, either constantly changing (for example a CPU percentage meter) or being static (for example the status of a service). Since Pandora FMS takes advantage of this to *compact* the information in the database (DB), it is a critical factor for the performance and the capacity study, since the more fragmentation, the more size in the DB and the more processing capacity will be necessary to process the same information.
- Module: This is the basic piece of information collected for monitoring. In some environments it is known as an Event, in others as a monitor and in others as a metric.
- Interval: The time between data collections from a module. It is usually expressed in minutes or seconds. The “average” value is usually 5 minutes.
- Alert: This is the notification that Pandora FMS runs when a data goes out of the established limits or changes its status to `CRITICAL` or `WARNING`.

Example of a capacity study

Scope definition

The study was carried out with an implementation divided into three main stages:

- Phase 1: Deployment of 500 agents.
- Phase 2: Deployment of 3000 agents.
- Phase 3: Deployment of 6,000 agents.

In order to determine the exact requirements of Pandora FMS in implementations made up by this data volume, it is necessary to know very well what kind of monitoring will be done, with the highest possible accuracy. For the following study we specifically took into account the features of the environment of a fictitious client called “QUASAR TECHNOLOGIES” that can be summarized in the following points:

- Monitoring 90% based on software agents.
- Homogeneous systems with a series of characterizations grouped into technologies / policies.

- Highly variable intervals between the different modules and events to be monitored.
- Large amount of asynchronous information (events, log items).
- Lots of process status information with very little probability of change.
- Little information on yields compared to the total.

After making an exhaustive study of all the technologies and determining the scope of the implementation (identifying the systems and their monitoring profiles), the following conclusions were reached:

- There is an average of 40 modules and events per machine.
- The average monitoring interval is 1200 seconds (20 minutes).
- Having modules that report information every 5 minutes and modules that report information once a week.
- From the entire set of total modules (240,000), the probability of change of each event for each sample was set to 25%.
- The alert rate per module was determined at 1.3 (i.e. 1.3 alerts per module/event).
- It is estimated (in this case it is an estimate based on our experience) that the probability of triggering an alert is 1%.

Capacity measurement

Once the basic requirements for the implementation in each stage (modules/second rate, number of total alerts, modules per day and megabytes per month) are known, a real stress test will be performed on a server relatively similar to the production systems (it was not possible to test on a machine similar to the production ones).

These stress tests will tell which process capacity Pandora FMS has in a server and what its degradation level with time is. This is useful for the following purposes:

1. By means of an extrapolation, to find out if the final volume of the project will be feasible with the hardware provided for this purpose.
2. Know what the limits of online storage are and what should be the cut-off points from which information is moved to history databases.
3. To know the response margins in the event of processing peaks, derived from problems that may arise (service stops, scheduped downtimes) where information pending processing accumulates.
4. To know the impact on performance derived from the different quality (percentage of change) of the monitoring information.
5. To understand the impact of alert processes in large volumes.

Tests were performed on a DELL PowerEdge T100® server with an Intel Core Duo® 2.4 GHz processor and 2 GB of RAM. This server, running on a Ubuntu Server 8.04, provided the study base for the tests in High Capacity environments. The tests were performed on agent configurations relatively similar to those of the QUASAR TECHNOLOGIES project. The intention of the tests is not to replicate exactly the same volume of information that QUASAR TECHNOLOGIES will have, since the same hardware is not available, but to replicate a high capacity environment, similar to QUASAR TECHNOLOGIES to evaluate the impact on performance over time and to determine other

issues (mainly performance) arising from handling large data volumes.

	Agentes	6005	Intervalo	3000	Modulos por Agente	30
	Modulos Tot.	180150	Prob. Cambio %	100	Alertas por Agente	20

	Probabilidad cambio (100%)			Probabilidad cambio (25%)		
	Tasa Paq/Sec	Tasa Mod/Sec	Tasa Procesasdos	Tasa Paq/Sec	Tasa Mod/Sec	Tasa Procesasdos
Dia 1	3	90	149,88%	8	240	399,67%
Dia 2	2,85	85,5	142,38%	7,5	225	374,69%
Dia 3	2,71	81,23	135,26%	6,75	202,5	337,22%
Dia 4	2,57	77,16	128,50%	6,08	182,25	303,50%
Dia 5	2,44	73,31	122,07%	5,47	164,03	273,15%
Dia 6	2,32	69,64	115,97%	4,92	147,62	245,83%
Dia 7	2,21	66,16	110,17%	4,43	132,86	221,25%
Dia 8	2,1	62,85	104,66%	3,99	119,57	199,12%
Dia 9	1,99	59,71	99,43%	3,59	107,62	179,21%
Dia 10	1,89	56,72	94,46%	3,23	96,86	161,29%
Dia 11	1,8	53,89	89,74%	2,91	87,17	145,16%
Dia 12	1,71	51,19	85,25%	2,62	78,45	130,65%
Dia 13	1,62	48,63	80,99%	2,35	70,61	117,58%
Dia 14	1,54	46,2	76,94%	2,12	63,55	105,82%
Dia 15	1,46	43,89	73,09%	1,91	57,19	95,24%
Dia 16	1,39	41,7	69,44%	1,72	51,47	85,72%
Dia 17	1,32	39,61	65,96%	1,54	46,33	77,14%
Dia 18	1,25	37,63	62,67%	1,39	41,69	69,43%
Dia 19	1,19	35,75	59,53%	1,25	37,52	62,49%
Dia 20	1,13	33,96	56,56%	1,13	33,77	56,24%
Dia 21	1,08	32,26	53,73%	1,01	30,39	50,61%

The results obtained are very positive since the system, although heavily overloaded, was capable of processing a very considerable volume of information (180,000 modules, 6,000 agents and 120,000 alerts). The conclusions drawn from this study are as follows:

1. “Real time” information should be moved to the history database within a maximum of 15 days, optimally for data older than one week. This guarantees a faster operation.
2. The margin of maneuver in the optimal case is almost 50% of processing capacity, higher than expected considering this volume of information.
3. The information fragmentation rate is key to determine the performance and capacity required for the environment where the system needs to be deployed.

Detailed methodology

Although the previous point represented a “quick” study based only on “dataserver” modules, this chapter presents a more complete way of analyzing Pandora FMS capacity.

As a starting point, in all cases the “worst case scenario “ approach will always be used whenever a choice can be made. It is assumed that, if no choice can be made, the “usual case” approach will be used. Nothing will ever be estimated in the “best case” since it is not valid.

Next, you will see how to calculate the system capacity, by type of monitoring or based on the source of the information.

Data server

Based on some goals, calculated according to the previous point, it will be assumed that the estimated goal is to see what the performance is like with a load of 100,000 modules, distributed among a total of 3,000 agents, that is, an average of 33 modules per agent.

It will create a task of `pandora_xmlstress` executed by cron or script manually, containing 33 modules, distributed with a configuration similar to this one:

- 1 chain module.
- 17 `generic_proc` modules.
- 15 `generic_data` modules.

The thresholds of the 17 `generic_proc` modules will be configured in this way:

```
module_begin
module_name Process Status X
module_type generic_proc
module_description Status of my super-important daemon / service / process
module_exec type=RANDOM;variation=1;min=0;max=100
module_end
```

Thresholds must be established in the 15 `generic_data` modules. The procedure to be followed is as follows:

The thresholds of the 15 `generic_data` modules will be configured to generate data like this:

```
module_exec type=SCATTER;prob=20;avg=10;min=0;''m''ax=100
```

The thresholds for these 15 modules will be configured to have this pattern:

```
0-50 normal
50-74 warning
75- critical
```

New tokens will be added to `pandora_xml_stress` configuration file to be able to define the thresholds from the XML generation. Attention: This is because Pandora FMS only “adopts” thresholds definition in module creation, but not in the update with new data.

```
module_min_critical 75
module_min_warning 50
```

Run `pandora_xml_stress`.

It should be left running for at least 48 hours without any kind of interruption and should monitor (with a Pandora FMS agent) the following parameters:

- Number of glued packages:

```
find /var/spool/pandora/data_in | wc -l
```

- `pandora_server` CPU:

```
ps aux | grep "/usr/bin/pandora_server" | grep -v grep | awk '{print $3}'
```

- Total memory of PFMS server:

```
ps aux | grep "/usr/bin/pandora_server" | grep -v grep | awk '{print $4}'
```

- CPU of mysqld (check execution syntax which depends on the MySQL distribution used):

```
ps aux | grep "sbin/mysqld" | grep -v grep | awk '{print $3}'
```

- Average response time of Pandora FMS DB:

```
/usr/share/pandora_server/util/pandora_database_check.pl
/etc/pandora/pandora_server.conf
```

- Number of monitors in unknown status (`unknown`):

```
echo "select SUM(unknown_count) FROM tagente;" | mysql -u pandora -p < password
> -D pandora | tail -1
```

(where < `password` > is the password of `pandora` user)

The first executions should be used to tune the server and MySQL configuration.

The script `/usr/share/pandora_server/util/pandora_count.sh` will be used to count (when there are XML files pending to process) the packet processing rate. The goal is to achieve all generated packets (3000) to be “processed” in an interval lower than 80 % of the time limit (5 minutes). This implies that 3000 packets have to be processed in 4 minutes, then:

```
3000 / (4 x 60) = 12.5
```

A processing rate of at least 12.5 packets must be achieved to make sure to a reasonable degree that Pandora FMS can process that information.

Elements to be adjusted:

- Number of threads.
- Maximum number of elements in intermediate queue (`max_queue_files`).
- Of course, all relevant MySQL parameters (very important).

An installation of Pandora FMS with a GNU/Linux server installed “by default” in a powerful machine, can not pass from 5 to 6 packets per second, in a powerful machine well “optimized” and “conditioned” it can reach 30 to 40 packets per second. This also depends a lot on the number of modules in each agent.

The system is configured so that the database maintenance script in `/usr/share/pandora_server/util/pandora_db.pl` is executed every hour instead of every day:

```
mv /etc/cron.daily/pandora_db /etc/cron.hourly
```

The system is left running with the packet generator for a minimum of 48 hours. Once this time has elapsed, the following points are evaluated:

1. ¿Is the system stable, did it crash? Should there be any problems, look at logs and graphs of the metrics obtained (mainly memory).
2. Evaluate the time trend of the metric “number of monitors in unknown state”. There should be no significant trends or spikes. They should be the exception. If they take place with a regularity of one hour, it is because there are problems with the concurrency of the DB management process.
3. Evaluate the metric “Average response time of Pandora FMS DB”. It should not grow over time but remain constant.
4. Evaluate the metric “CPU of `pandora_server`”: it should have frequent peaks, but with a constant trend, not increasing.
5. Evaluate the metric “MySQL server CPU”, it should remain constant with frequent peaks, but with a constant, not increasing trend.

Evaluation of alert impact

If everything went well, you should now evaluate the performance impact of the alert execution. Apply an alert to five specific modules of each agent (`generic_data`), for the `CRITICAL` condition. Something that is relatively lightweight, such as creating an event or writing to syslog (to avoid the impact that something with high latency such as sending an email message could have).

You may optionally create an event correlation alert to generate an alert for any critical condition of any agent with one of these five modules.

Leave the system operating for 12 hours under these criteria and evaluate the impact, following the above criteria.

Purge evaluation and data transfer

Assuming the data storage policy was:

- Deletion of events older than 72 hours.
- Move data to history that is more than 7 days old.

You should leave the system running “alone” for at least 10 days to evaluate long-term performance. You may see a substantial “spike” after 7 days due to moving data to the history DB. This degradation is important to be taken into account. If you do not have that much time, you may reproduce it (with less “realism”) by changing the purge interval to 2 days for events and 2 days to move data to history, to evaluate said impact.

ICMP Server

It is specifically the ICMP network server. In case of testing for the network server Open version, see the point corresponding to the network server (generic).

Assuming you already have the server up and running and configured. Some key parameters for its operation:

```
block_size X
```

It defines the number of pings that the system will do per run. If most pings will take the same amount of time, you may raise the number to a considerably high number, such as 50 to 70.

If, on the other hand, the number of ping modules is heterogeneous and they are in very different networks, with very different latency times, it is not interesting to set a high number, because the test will take as long as the slowest one takes, so you may use a relatively low number, such as 15 to 20.

```
icmp_threads X
```

Obviously, the more threads you have, the more checks you will be able to execute. If you add all the threads that Pandora FMS executes, they should not reach the range of 30 to 40. You should not use more than 10 threads here, although it depends a lot on the type of hardware and the GNU/Linux version you are using.

Now, you must “create” a fictitious number of ping modules to be tested. It is assumed that you will test a total of 3000 ping modules. To do this, it is best to take a system on the network that is capable of supporting all pings (any GNU/Linux server can handle the task).

Using the Pandora FMS CSV importer, create a file with the following format:

(agent name,IP address,OS identifier,interval,group identifier)

This shellscript can be used to generate such a file (by changing the destination IP address and group ID):

```
A=3000
while [ $A -gt 0 ]
do
    echo "AGENT_$A,192.168.50.1,1,300,10"
    A=`expr $A - 1`
done
```

The main thing is to have Pandora FMS monitored, measuring the metrics from the previous point: CPU consumption (pandora and mysql), number of modules in unknown state and other interesting monitors.

Import the CSV to create 3000 agents, which will take a few minutes. Then go to the first agent (AGENT_3000) and create a PING module in it.

Then go to the bulk operations tool and copy that module to the other remaining 2999 agents.

Pandora should start processing those modules. Measure with the same metrics as the previous case and see how it evolves. The goal is to leave a system operable for the required number of ICMP modules without any of them reaching unknown status.

SNMP Server

This is the SNMP network server. Assuming you already have the server up and running and configured. Some key parameters for its operation:

```
block_size X
```

It defines the number of SNMP requests that the system will make for each execution. Bear in mind that the server groups them by destination IP address, so this block is a guideline. It should not be too large (30 to 40 maximum). When a block element fails, an internal counter causes the PFMS server to retry it.

```
snmp_threads X
```

Obviously, the more threads you have, the more checks you will be able to execute. If you add all the threads that Pandora FMS executes, they should not reach the range of 30 to 40. You should not use more than 10 threads here, although it depends a lot on the type of hardware and the GNU/Linux version you are using.

The fastest way to test it is through an SNMP device, applying to all interfaces, all the “basic” monitoring modules as standard. This is done by applying SNMP Explorer (Agent → Administration Mode → SNMP Explorer). Identify the interfaces, and apply all metrics to each interface. On a 24-port switch, this generates about 650 modules.

If you generate another agent with another name, but the same IP address, it will have another 650 modules. Another option could be to copy all the modules to a series of agents that all have the same IP address so that the copied modules work “attacking” the same switch.

Another option is to use an SNMP emulator, such as Jalasoft SNMP Device Simulator.

The goal of this point is to be able to constantly monitor a pool of SNMP modules for at least 48 hours, monitoring the infrastructure, to ensure that the monitoring rate of modules per second is constant, and there are no time periods where the server generates modules in unknown state. This situation could be caused by:

- Scarcity of resources (memory, CPU). You could see a trend of these metrics continuously increasing, which is a bad sign.
- Occasional problems: Daily server reboot (for log rotation), execution of scheduled database maintenance, or other scripts running on the server or the DB server.
- Network problems, arising from unrelated processes (e.g. data backup of a server on the network) that affect the speed and availability of the network at a given time.

Plugins, Network (Open) and HTTP Server

The same concept applies here as above, but in a more simplified form. It will be necessary to control:

- Number of threads.
- Timeouts to calculate the worst-case incidence.
- Average check time.

Size a test set with this data, and verify that the server capacity is constant over time.

Trap reception

Here the assumption is simpler: it is assumed that the system will not receive traps constantly, but rather to evaluate the response to an avalanche of traps, some of which will generate alerts.

To that end, just make a script that generates traps in a controlled manner at high speed:

```
#!/bin/bash
TARGET=192.168.1.1
while [ 1 ]
```

```
do
   snmptrap -v 1 -c public $TARGET .1.3.6.1.4.1.2789.2005 192.168.5.2 6 666
1233433 .1.3.6.1.4.1.2789.2005.1 s "$RANDOM"
done
```

Note: Stop it with the CTRL+C key after a few seconds, as it will generate hundreds of traps quickly.

Once the environment has been set up, the following assumptions must be validated:

1. Trap injection at a constant rate (just enter a `sleep 1` command to the above script inside the while loop, to generate 1 trap per second. The system is left running for 48 hours and the impact on the server is evaluated.
2. Trap storm. Evaluate the situation before and during a trap storm, as well as its recovery.
3. Effects of the system on a very large table of traps (greater than 50 thousand). This includes the effect of passing the DB maintenance.

Events

Similar to SNMP, PFMS events will be evaluated in two scenarios:

1. Normal event reception rate. This has already been tested in the data server, since an event is generated at each state change.
2. Event generation storm. To do this, force event generation through CLI. Using the following command (with an existing group called “Tests”):

```
pandora_manage /etc/pandora/pandora_server.conf --create_event "Event test"
system Tests
```

That command, used in a loop like the one used to generate traps, may be used to generate dozens of events per second. It can be parallelized in a script with several instances to generate a higher number of insertions. This would simulate system performance in an event storm. That way the system could be tested before, during and after an event storm.

User concurrence

For this, another server separate from Pandora FMS will be used, using the WEB monitoring feature. In a user session where it will perform the following tasks in a specific order and measure how long they take to be processed:

1. Login to the web console.
2. See events.
3. Go to the group view.
4. Go to agent detail view.
5. Display a report (in HTML). This report should contain a couple of graphs and a couple of modules

with SUM or AVERAGE reports. The interval for each item should be one week or five days.
6. Display of a combined graph (24 hours).
7. Generation of PDF reports (other different reports).

This test is performed with at least three different users. You may parallelize that task to run it every minute, so that if there are 5 tasks (each with its user), you would be simulating the navigation of five simultaneous users. Once the environment is established, it will take into account:

1. The average speed of each module is relevant in order to identify “bottlenecks” related to other parallel activities, such as the execution of maintenance *script*, etc.
2. CPU and memory impact on the server will be measured for each concurrent session.
3. The impact of each simulated user session will be measured regarding the average time of the rest of the sessions. That is, it should be estimated how many seconds of delay each simultaneous extra session adds.

Back to Pandora FMS Documentation Index