Linux for zSeries & S/390 Performance Measurement with RMF PM

Oliver Benke
IBM Deutschland Entwicklung GmbH
D3141 (eServer Systems Management Technology)
Schönaicher Str. 220
D-71003 Böblingen, Germany
Email: benke@de.ibm.com

September 5, 2001

Contents

Chapter 1

RMF PM Server *rmfpms*

1.1 Introduction

rmfpms is a modular data gatherer for Linux. The gathered data can be analyzed using the RMF PM client application. The performance data is accessible through XML over HTTP so you can easily exploit it in your own applications.

Using our enhanced RMF PM client, it is possible to

- generate graphical trend reports
- store the data in spreadsheet format (.WK1, supported by Lotus 1-2-3, StarOffice, MS Excel, etc.)
- persistently store performance analysis scenarios (Performance Desktops, "PerfDesks")
- gather historical performance data
- filter the performance data
- mix OS/390 and Linux performance data in one screen
- use a graphical user client to access performance data in a flexible and highly configurable way

1.2 Installation and Getting Started

1.2.1 Prerequisites

1.2.1.1 Server

- Linux Version 2.2 or 2.4
- IBM zSeries, IBM S/390 or Intel x86 architecture
- IEEE FPU Support, either compiled in kernel or available in hardware
- glibc 2.1, libpthread

1.2.1.2 Client

Windows NT, Windows 9x, Windows 2000 or Linux with Java JRE 1.3

1.3 Usage

1.3.1 Installation

Easy start:

- download RMF PM client from our web site and install it
  - a) Windows desktop:
    http://www.s390.ibm.com/rmf/rmfhtmls/pmweb/gpmwinpm.exe
  - b) Linux desktop:
    http://www.s390.ibm.com/rmf/rmfhtmls/pmweb/gpmlinpm.tgz

  Under Linux, you have to unpack the client using "tar xvfz gpmlinpm.tgz". Then change to the new directory rmfpm and type "./rmfpm". Under Windows, it is a self extracting InstallShield file; just execute it.

- download the gatherer code and unpack it using "tar xvfz rmfpms_arch.tgz"
  - a) Linux for S/390:
    http://www.s390.ibm.com/rmf/rmfhtmls/pmweb/rmfpms_s390.tgz

  - b) x86 Linux:
    http://www.s390.ibm.com/rmf/rmfhtmls/pmweb/rmfpms_x86.tgz

- customize rmfpms/.rmfpms_config if you need to; otherwise you get the defaults:
  - IBM_PERFORMANCE_REPOSITORY=$HOME/.rmfpms/
  - IBM_PERFORMANCE_HOME=$HOME/rmfpms/bin/
  - IBM_PERFORMANCE_MINTIME=60
  - APACHE_ACCESS_LOG=/var/log/httpd/access_log
  - APACHE_SERVER=localhost
  - APACHE_SERVER_PORT=80
- you may customize the HTTP_PORT port number in rmfpms/bin/gpmsrv00.ini; the default port number is 8803
- type "rmfpms/bin/rmfpms start"
- test your installation using RMF PM Linux prototype or Internet Explorer (unfortunately, Netscape has problems with XML)
  http://your_host_name:8803/

1.3.2 Apache Specific Configuration

We have provided many options for you to customize to suite your Apache serve setup.

You need to make some customizations in the `rmfpms/.rmfpms_config` file:

- APACHE_ACCESS_LOG : the location of you Apache log file
  We have assumed that you have *only one* `access_log` file for the entire machine.
- APACHE_SERVER, APACHE_SERVER_PORT : hostname and port address of your Apache server.
- APACHE_SERVERS_CONFIG : if you have more than one Apache server running on a machine, then you can specify a file listing these servers.
  This file's format should be as follows:

```
server1.localdomain:port1:/path/to/access_log1
server2.localdomain:port2:/path/to/access_log2
server3.localdomain:port3:/path/log3
server1.localdomain2:80:/path/log4
```

NOTES:

- You must explicitly list the port numbers – *even the default port*: 80.
- You must explicitly list the `access_log` locations.
- If you specify an APACHE_SERVERS_CONFIG entry in the `.rmfpms_config` file, then the APACHE_ACCESS_LOG, APACHE_SERVER, and APACHE_SERVER_PORT environment variables are ignored.

In the httpd.conf file of your Apache web server, you need to enable the statistics module if you'd like to have those data incorporated into RMFPMS:

```
LoadModule status_module modules/mod_status.so
[ ... ]
AddModule mod_status.c
[ ... ]
ExtendedStatus on
[ ... ]
<Location /server-status>
    SetHandler server-status
</Location>
```

1.3.3 RMFPMS Logging

All GAT modules work as daemons. Error messages go either to the system log or to some files in "$IBM_PERFORMANCE_REPOSITORY/logs". The default for "$IBM_PERFORMANCE_REPOSITORY" is "$HOME/rmfpms/.rmfpms"; it is set in "rmfpms/.rmfpms_config". We suggest starting the data gatherer at system startup time.

Please note it is possible to start only some of the gatherer modules if you don't need all of them. You can also gather information in raw format without the gpmddsrv web server active; if you want to make a benchmark with minimal interference, just start the "*gat" programs you need and after your benchmark is finished, start the web server in order to analyze it using RMF PM or any other program.

1.3.4 General RMFPMS Setup information

Important Environment Variables

- $IBM_PERFORMANCE_REPOSITORY : The directory in which the performance data is stored
- $IBM_PERFORMANCE_HOME : The directory in which the executables and config files reside
- $IBM_PERFORMANCE_MINTIME : Cycle time, default 60 sec. Each gatherer gathers some raw performance data every MINTIME seconds. The gatherer modules need not be in sync, and they can work with different mintimes

Important configuration files

- rmfpms/bin/gpmddsrv : special Web server for rmfpms
- rmfpms/bin/gpmsrv00.ini : gpmddsrv ini file
- rmfpms/.rmfpms_config : file containing some environment variables which are needed for all other rmfpms programs
- rmfpms/bin/*gat : low level gatherer modules
- rmfpms/bin/*comp : internally used modules

1.3.5 Starting and stopping (Script-based)

The shell script rmfpms in the bin directory is used to start and stop the gatherer modules and the lightweight web server, gpmddsrv.

Commands:

rmfpms start starts gatherer and web server

rmfpms stop stops gatherer and web server

rmfpms status display whether or not the gatherer and DDS web server are active

rmfpms restart first stops, then starts again

1.4 Archiving performance data

We have provided two shell scripts for easy archiving of performance data. The tools are located in the "bin" subdirectory.

rmfpms_archive this tool archives all performance data not created at the same day; additionally, it deletes temporary files

rmfpms_unarchive yyyymmdd if you want to restore the performance data from 01/16/2001, you have to type *rmfpms_archive 20010116*

We suggest to execute rmfpms_archive once a day as a cron job. If you need to analyze old performance data, you can restore them using *rmfpms_unarchive*. The archived data is stored in the *rmfpms/archive* subdirectory.

You might add something like the following line to your cronfile by using the *crontab -e* command:

```
0 23 * * * /home/benke/rmfpms/bin/rmfpms_archive
```

1.5 Exploitation of XML performance data

The following section describes how to exploit the performance data in your own applications.

We are using similar technologies for x86 Linux, Linux for zSeries & S/390, OS/390 and z/OS.

1.5.1 Explore the data format with a Web browser

You can explore the data format using a Stylesheet-capable web browser (currently only Microsoft Internet Explorer Version 5).

To start, type in `http://yourhost.yourdomain:8803` where `yourhost.yourdomain` is the IP name of the Linux host monitored. By the way: for the Microsoft Internet Explorer, `http://` is important and cannot be omitted. Start exploring the resources and metrics presented. To get an idea of what the transfered XML data looks like, view the source in the web browser.

1.5.2 `http://yourhost:8803/gpm/config/index.xml`

This config files contains information regarding all metrics supported by the DDS server. Example:

```
<?xml version="1.0" encoding="UTF-8" ?>
- <ibmgpm:application xmlns:ibmgpm="http://www.ibm.com/servers/eserver/zseries/
  <ibmgpm:server name="RMF-DDS-Server" version="LINUX-BETA" functionality="1670
  <ibmgpm:ostype>Linux</ibmgpm:ostype>
  <ibmgpm:osversion>2.4.2-0tape-dasd</ibmgpm:osversion>
  <ibmgpm:domain identifier="lnxbenk1" idtype="SystemName" />
- <ibmgpm:resource type="LINUX_CPU" expandable="NO" icon="gpmlcpu.gif" helpurl=
  <ibmgpm:metric id="402050" description="% cpu idle time" metric-type="SINGLE"
  <ibmgpm:metric id="4020A0" description="% cpu idle time by processor" metric-
  <ibmgpm:metric id="402030" description="% cpu time in kernel mode" metric-typ
  <ibmgpm:metric id="4020D0" description="% cpu time in kernel mode by process"
...
```

For each resource, there is a `<ibmgpm:resource type="RESOURCE_NAME"` entry. Contained are the metrics used to record performance data for that resource, e.g. `<ibmgpm:metric id="402050" description="`

1.5.3 HTTP URLs

`By sending HTTP URL requests to the gpmddsrv (DDS) process, a lightweight web server, you can ask for some performance data gathered in the past.`

Example:

```
http://tux390.boeblingen.de.ibm.com:8803/gpm/perform/perform.xml?
resource="tux390,*,LINUX_MEMORY"&id=405100&
RANGE=20010725100000,20010725110000
```

First there is the socket consisting of IP address of the monitored Linux server and the port address of the gpmddsrv web server. After the first "?" character, there are several parameters for the web server:

- *resource=...*
  Name of the resource for which you'd like some information. The metric ID must be useful for this kind of resources. You can find the metric ID and to which resource it belongs to in the config.xml file.

- *id=...*
  Metric ID.The metric ID must be useful for this kind of resources. You can find the metric ID and to which resource it belongs to in http://yourhost:8803/gpm/config/index.xml.

- *RANGE=20010725100000[,20010725110000]*
  gather performance data for the specified interval, format YYYYMMDDhhmmss.

  In the example, the start time is 10:00 AM, 07/25/2001, and the end time is at 11:00 AM at the same day. All times are in local time of the monitored system. If the ending time is omitted, the current time is used.

  RMFPMS gathers data every few seconds, configurable in *.rmfpms_config*. If you'd like a performance value for 10:00:30 AM - 10:03:30 AM, it may be that RMFPMS gathered data at 09:59:00, 10:00:00, ..., 10:03:00, 10:04:00. In this cases, RMFPMS computes the performance value using the gathered counters from 10:00:00 and 11:04:00 (surrounding interval). If there is no data available for a surrounding interval, an error message is returned.

All data gatherers are automatically synchronized[1].

  - YYYY - year
  - MM - month
  - DD - day
  - hh - hour
  - mm - minute
  - ss - second

- *PREVIOUS=nnnn[,mmmm]*
  get the performance data for the interval nnnn sec before current time until mmmm sec before the current time. The second parameter can be omitted.

1.5.4 XML response

Using the URL described above, you get a response from the DDS web server.

There is always an attribute called *value* as well as an attribute called *per* for percentage. The contents of *per* are used by the stylesheets to graphically present the data in a web browser.

1.5.4.1 Example for a simple counter

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="/gpm/include/perform.xsl"?>
<perf-config>
  <server name="RMF-DDS-Server"
version="LINUX-BETA" functionality="1400"/>
  <performance-data>
    <metric id="4050A0" d
escription="% swap space used"
type="single"
```

[1] If a data gatherer is started n seconds after 01/01/1970 and r is the *IBM_PERFORMANCE_MINTIME* specified in *rmfpms/.rmfpms_config*, the gatherer waits for $r - nMODr$ seconds.

```
resource="tux390,*,LINUX_MEMORY"
workscope=",,G" filter="HI=20">
      <timestamp localtime="02/16/2001 19:28:21"
timestring="20010216192821"/>
      <timerange localstart="20010216192721"
localend="20010216192821"
utcstart="20010216182721"
utcend="20010216182821"/>
      <gathererinterval seconds="60"/>
      <row label="" value="0.21946" per="66.6667"/>
    </metric>
  </performance-data>
</perf-config>
```

1.5.4.2 Example for a list-valued counter

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="/gpm/include/perform.xsl"?>
<perf-config>
  <server name="RMF-DDS-Server"
version="LINUX-BETA" functionality="1400"/>
  <performance-data>
    <metric id="405100"
description="resident set size in KB by process"
type="multi"
resource="tux390,*,LINUX_MEMORY"
workscope=",,G" filter="HI=20">
      <timestamp localtime="02/16/2001 19:24:32"
timestring="20010216192432"/>
      <timerange localstart="20010216192332"
localend="20010216192432"
utcstart="20010216182332"
utcend="20010216182432"/>
      <gathererinterval seconds="60"/>
      <row label="httpd.4129" value="8800" per="100"/>
      <row label="httpd.214" value="8480" per="96.3636"/>
      <row label="httpd.287" value="8252" per="93.7727"/>
      <row label="java.16050" value="7172" per="81.5"/>
```

```
      <row label="java.16051" value="7172" per="81.5"/>
      <row label="java.16052" value="7172" per="81.5"/>
      <row label="java.16059" value="7172" per="81.5"/>
      <row label="java.16058" value="7172" per="81.5"/>
      <row label="java.16057" value="7172" per="81.5"/>
      <row label="java.16055" value="7172" per="81.5"/>
      <row label="java.16056" value="7172" per="81.5"/>
      <row label="java.16040" value="7172" per="81.5"/>
      <row label="java.16049" value="7172" per="81.5"/>
      <row label="gpmddsrv.23031" value="1564" per="17.7727"/>
      <row label="gpmddsrv.23032" value="1564" per="17.7727"/>
      <row label="gpmddsrv.23033" value="1564" per="17.7727"/>
      <row label="gpmddsrv.23030" value="1564" per="17.7727"/>
      <row label="sendmail.240" value="1256" per="14.2727"/>
      <row label="sh.16037" value="1036" per="11.7727"/>
      <row label="gpmddsrv.21760" value="948" per="10.7727"/>
    </metric>
  </performance-data>
</perf-config>
```

1.5.5 Filter

You can filter the data transfered to the client by a filter qualifier added to the HTTP request string. Examples for filters are:

- One or more name patterns in form of a simple expression or a regular expression to be matched against the names of the list (simple and regular expressions are discussed below).
- A lower and upper bound threshold to be compared to the values of a list.
- A maximum list length with an indicator to select the pairs with either the highest or the lowest values.
- A sorting order for the either the names or the values of the list (ascending or descending).

If you don't specify a filter explicitly, the default settings used by rmfpms are:

&filter="HI=20;ORD=VD"

meaning rmfpms returns the top 20 values, sorted by value, descending.
You can add the filter to the URL as follows:

http://yourhost.yourdomain:8803/gpm/perform/perform.xml?resource="yourhost,*,LI

Several filter criteria can be seperated by ";".

The following keywords are available for filters:

- PAT=<expression> specifies one or more patterns either as a *regular expression* or as a *simple expression* which must match the NAME part of a list name/value pair. Note that the PID is part of the process name in Linux rmfpms, so you should specify *init.1* or *init.** rather than *init*.

  For example: to specify a filter for all gpmddsrv threads and the *web* process with PID 12345, specify

  &filter="PAT=gpmddsrv*|web.12345

  (see also "Regular Expressions" below).

  For a *simple expression*, you just concatenate the identifiers you're looking for. For example:

  &filter="PAT=httpd.2345|httpd.4567"

- UB=<double> same as LB=, but all list elements with values HIGHER than Upper Bound(UB) are discarded.

- HI=<int> All list elements except the highest n are discarded (mutually exclusive with LO=)

- LO=<int> All list elements except the lowest n are discarded (mutually exclusive with HI=)

- ORD=<xx> sort list name/value pairs by their names in ascending/descending (NA,ND) order, or by their values (VA/VD), or not at all (NN).

1.6 Available Metrics

The available metrics are grouped by *resources*. Under Linux, available resources include SYSTEM, MEMORY, NETWORK, CPU and FILESYSTEM.

There are two major classes of counters: *list-valued* counter and *single* counter. For a single counter, there is exactly one value reported per sample time. For a list valued counter like *"... by process"*, you get a list of value-name pairs for each sample time.

1.6.1 Resource *LINUX_SYSTEM*

- *rate of context switches (per second)*
  Number of context switches per second
  metric id 0x400020

- *rate of processes created (per second)*
  Number of processes created per second
  metric id 0x400010

- *Apache HTTP server: bytes per request*
  Average number of bytes transfered per request to the Apache HTTP Server
  metric id 0x400310

- *Apache HTTP server: number of 404 errors per minute*
  Number of "file not found" errors (404 errors) per minute
  metric id 0x400340

- *Apache HTTP server: number of busy threads*
  Average number of busy threads
  metric id 0x400320

- *Apache HTTP server: number of idle threads*
  Average number of idle threads
  metric id 0x400330

- *Apache HTTP server: rate of requests (per second)*
  Number of requests per second
  metric id 0x400300

1.6.2 Resource *LINUX_MEMORY*

- *% swap space used*
  Percentage of swap space used at the end of a cycle time
  metric id 0x4050A0

- *cache memory in MB*
  Memory used for cache in MBytes (1048576 Bytes) at the end of a cycle time
  metric id 0x405060

- *free swap space in MB*
  Free swap space in MBytes at the end of a cycle time
  metric id 0x405090

- *major page fault rate by process*
  Number of major page faults by process per second. A major page fault occurs if - from the Linux operating system's perspective - disc access is involved.
  metric id 0x405120

- *major page fault rate including children by process*
  Like *major page fault rate by process*, but including all children processes.
  metric id 0x405140

- *memory used for buffers in MB*
  Memory used for buffers in MBytes at the end of a cycle time. The Linux buffer cache is a disc cache intended to relieve processes from having to wait for relatively slow disks to retrieve or store data. Linux automatically uses otherwise unused memory for disc buffers. When free memory becomes scarce, buffer frames are automatically released.
  The buffer cache kernel thread can be tuned using

/proc/sys/vm/bdflush. Please refer to the file *Documentation/proc.txt* in the Linux kernel source tree.
metric id 0x405040

- *memory used in MB*
  Memory used in MB at the end of a cycle time. This includes memory used for buffer caches.
  metric id 0x405020

- *minor page fault rate by process*
  Number of minor page faults by process per second. If there is (from the kernel's perspective) no disc access involved in the page fault (like Copy on Write for a shared page), the page fault is called a *minor* page fault.
  metric id 0x405110

- *minor page fault rate including children by process*
  Like *minor page fault rate by process*, but including all children processes.
  metric id 0x405130

- *number of pages paged out per second*
  The number of pages paged out of memory per second.
  metric id 0x4050E0

- *number of pages swapped in per second*
  The number of pages swapped in per second.
  Please note the Linux kernel uses swapping only if there is not enough memory available.
  metric id 0x4050B0

- *number of pages swapped out per second*
  The number of pages swapped out per second.
  Please note the Linux kernel uses swapping only if there is not enough memory available. If swapping is not backed by IBM VM on a S/390 or zSeries mainframe, swapping is normally very bad for performance and application responsiveness.
  metric id 0x4050C0

- *resident set size in KB by process*
  Resident set size in KBytes by process. The *resident set size*, *RSS*, is the total size of all parts of a process (code, data, shared libraries) actually resident in memory. This can tell you how much memory the processes are "really" using.
  metric id 0x405100

- *shared memory in MB*
  Memory in MBytes usable by more than one process. If any part of memory could be used by more than one process, it is counted to be shared memory.
  metric id 0x405050

- *size of swap space in MB*
  Size of maximal available swap space in MBytes.
  metric id 0x405070

- *total memory size in MB*
  Total size of memory available.
  metric id 0x405010

- *shared memory in MB*
  Memory in MBytes usable by more than one process at the end of a cycle time. If any part of memory could be used by more than one process, it is counted to be shared memory.
  metric id 0x405050

- *used swap space in MB*
  Total size of memory swapped out in KBytes at the end of a cycle time. metric id 0x405080

- *virtual memory size by process*
  Size of virtual memory in bytes by process at the end of a cycle time. This is normally a very big number, but most parts of the virtual size are often never used (not even paged in).
  metric id 0x4050F0

1.6.3 Resource *LINUX_NETWORK*

- *bytes received per second*
  Sum of bytes received per second through all connected network devices.
  metric id 0x401030

- *bytes received per second by network device*
  Bytes received per second by network device.
  metric id 0x401090

- *bytes transmitted per second*
  Bytes transmitted per second through any network device.
  metric id 0x401040

- *bytes transmitted per second by network device*
  Bytes transmitted per second by network device.
  metric id 0x4010A0

- *packets received per second*
  Packets received per second through all connected network devices. Though the size of a packet is not constant, this can give you an idea about the network processor utilization and the CPU utilization caused by network traffic; if you have many small network packets, you don't have problems with bandwidth but with (network) processor speed. metric id 0x401010

- *packets received per second by network device*
  Number of packets received per second by network device.
  metric id 0x401070

- *packets transmitted per second*
  Number of packets send or received per second over all network devices.
  metric id 0x401020

- *packets transmitted per second by network device*
  Packets transmitted per second by network device.
  metric id 0x401080

- *receive errors per second*
  `Number of receive errors per second.`
  `metric id 0x401050`

- *receive errors per second by network device*
  `Number of receive errors per second by network device.`
  `metric id 0x4010B0`

- *transmit errors per second*
  `Number of transmit errors per second.`
  `metric id 0x401060`

- *transmit errors per second by network device*
  `Number of transmit errors per second by network device.`
  `metric id 0x4010C0`

1.6.4 Resource *LINUX_CPU*

- *% cpu idle time*
  `Percentage of CPU idle time, averaged over all CPUs.`
  `metric id 0x402050`

- *% cpu idle time by processor*
  `Percentage of CPU idle time by processor.`
  `metric id 0x4020A0`

- *% cpu time in kernel mode*
  `Percentage of CPU time in kernel mode, averaged over all CPUs.`
  `metric id 0x402030`

- *% cpu time in kernel mode by process*
  `Percentage of CPU time in kernel mode by process.`
  `metric id 0x4020D0`

- *% cpu time in kernel mode by processor*
  `Percentage of CPU time in kernel mode by processor.`
  `metric id 0x402080`

- *% cpu time in nice mode*
  Percentage of CPU time in nice mode, averaged over all CPUs. A process runs in "nice mode" if the scheduling priority is lower than normal. If the superuser has increased the scheduling priority of some processes to values higher than normal, this is not considered "nice mode".
  metric id 0x402090

- *% cpu time in user mode*
  Percentage of CPU time in user mode, i.e. executing ordinary user code, averaged over all processors. If a user program calls a system routine like *open()*, the execution time needed for the *open()* system routine is counted as kernel mode, while the normal processing of the user program is counted as user mode.
  metric id 0x402020

- *% cpu time in user mode by process*
  Percentage of CPU time in user mode by process.
  metric id 0x4020C0

- *% cpu time in user mode by processor*
  Percentage of CPU time in user mode by processor.
  metric id 0x402070

- *% cpu time total by process*
  Percentage of all CPU time by process, including CPU time in nice mode, user mode time and kernel mode time.
  metric id 0x402110

- *% cpu total active time*
  Percentage of CPU total active time, averaged over all CPUs.
  metric id 0x402060

- *% cpu total active time by processor*
  Percentage of CPU total active time by processor.
  metric id 0x4020B0

- *accumulated cpu time in kernel mode by process*
  Accumulated CPU time in kernel mode by process, counted since process creation.
  metric id 0x4020F0

- *accumulated cpu time in user mode by process*
  This gives the accumulated CPU time in user mode by process, counted since process creation.
  metric id 0x4020E0

- *load average*
  Average length of the queue of processes ready to run. This is the number of processes which can be executed concurrently provided there are enough CPUs available.
  metric id 0x402010

- *nice value by process*
  Nice value by process. The nice value can be changed with the *nice* operator command. This value ranges from -20 (highest priority) to 19 (lowest priority).
  metric id 0x402100

1.6.5 Resource *LINUX_FILESYSTEM*

For disc related metrics, 1 MB is 1048576 Bytes.

- *% free by file system*
  Percentage of free memory by mount point.
  metric id 0x404080

- *% of space free*
  Percentage of free memory on all mounted filesystems.
  metric id 0x404040

- *% of space used*
  Percentage of memory used on all mounted filesystems.
  metric id 0x404030

- *% used by file system*
  Percentage of memory used by mount point.
  metric id 0x404070

- *available (in 1 MB blocks) by file system*
  Available memory by filesystem.
  metric id 0x404060

- *size (in 1 MB blocks) by file system*
  Used memory by filesystem.
  metric id 0x404050

- *total size of all file systems (in 1 MB blocks)*
  Defined capacity of all mounted filesystems.
  metric id 0x404010

- *total space available (in 1 MB blocks)*
  Total space available on all mounted filesystems.
  metric id 0x404020

- *dasd io average response time per request (in msec)*
  Image-wide average response time per DASD (disc) I/O request, in milli-seconds. This metric is only available for images running 2.4 kernel on IBM S/390 or zSeries mainframes.
  metric id 0x404410

- *dasd io average response time per sector (in msec)*
  Like *dasd io average response time per request (in msec)*, but average response time *per sector*. This metric is only available for images running 2.4 kernel on IBM S/390 or zSeries mainframes.
  metric id 0x404420

- *dasd io requests per second*
  Number of DASD (disc) I/O requests per second. This metric is only available for images running 2.4 kernel on IBM S/390 or zSeries mainframes.
  metric id 0x404400