CPU, Disk and Memory probe (cdm)
Technical Brief
Disk information ....................................................................................................................... 10
AIX ............................................................................................................................................. 10
HP-UX ......................................................................................................................................... 10
Linux .......................................................................................................................................... 10
Solaris ........................................................................................................................................ 11
Tru64.......................................................................................................................................... 11

TROUBLESHOOTING ................................................................................................................... 11

GUIDELINES FOR SENDING BUG REPORTS FOR CDM ON THE DIFFERENT PLATFORMS ............... 11
All platforms: ................................................................................................................................ 11
Windows: .................................................................................................................................... 11
Unix (all platforms): ...................................................................................................................... 11
AIX: ............................................................................................................................................. 11
Hp-ux: ......................................................................................................................................... 11
LINUX: ......................................................................................................................................... 11
Solaris: ......................................................................................................................................... 12
Tru64: .......................................................................................................................................... 12
General
The cdm probe collects and monitors CPU, Disk and Memory information from the local machine. It has support for monitoring network disks (shares on Windows and NFS mounted file systems on UNIX/Linux). Alarming is done on the average of a number of samples.

Technical Overview
Information is collected through an API call layer that hides the operating system specifics from the general probe logic.

Measurements are taken at separate intervals for CPU related information (including system load / processor queue length), Memory information (including paging) and Disk information.

Each measurement is stored in a history buffer of user defined length.

QoS messages are generated directly from the measurements, but alarms are based on averaged data.

The configuration tool uses the historic data to present graphically CPU and Memory data.

Disks are discovered and a monitoring profile is created for each based on a default profile (which is currently not configurable through the configuration tool). On discovering disks, the probe restart itself to start using the created profiles.

The probe has a get_export callback that is used in conjunction with the old template discovery, where a dashboard template is extracted from the package and combined with the output from get_export to create a dashboard configured to display information from the discovered probe.

The probe is now "cluster aware" in the sense that it will detect and communicate with a local cluster probe and find current virtual groups and disks.

Configuration and Data Files
The probe only uses the normal configuration and log file.

The configuration file has sections for

**setup**
General settings.

**messages**
All messages are stored here. Each message has a token that identifies which alarm situation it is meant for. For each alarm situation an alarm message is explicitly associated. There is no notion of a default message beyond the original .cfx file settings.

**cpu**
Cpu monitoring settings are stored in two sections: alarm with the generic configuration and specific with platform specific settings, for instance processor queue length for Windows.

**memory**
The memory monitoring settings only contain one section: alarm with a generic configuration.
disk

the disk monitoring settings are also divided into two sections: alarm contains the current profiles and fixed_default contains the default monitoring settings for discovered disks.

computer

Currently only contains QoS configuration for uptime.

cluster

In this section there will be a subsection for each detected virtual cluster group that contains at least one disk resource. The structure underneath the virtual group section is the same as the top level, but can only contain a disk section.

Disk default settings are taken from the top level default.

Cluster integration

The probe works together with the cluster probe to allow monitoring of disk resources. The probe will automatically detect these and register its monitoring profile and messages with the cluster probe.

The user can set alarm and QoS source for each individual cluster group.

Monitoring settings for each cluster group is kept separate in the configuration file so that cluster disks in separate virtual groups can be monitored concurrently.

Cluster disks are automatically excluded from local monitoring.

This functionality needs cluster probe version 2.2x and uses the "discover_cluster", "get_cluster_resources" and "register_section" callbacks to this probe.

Command Interface Description

*cluster_info*

Returns a 'cluster_present' flag and lists of virtual groups and cluster disks present on the current node.

*connection_status*

Returns a table of entries where each entry represents an active share connection monitoring profile. The entry will contain the following fields: fetch_time, map_ok, map_msg, remote_name and user.

*cpu_history*

Returns a table of data where each represents one measurement. When the computer has more than one processor, averaged cpu data is presented at the top level, and cpu information for each individual cpu is present in a cpu subtable.

**cpu_status (id)**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Req</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>String</td>
<td>no</td>
<td>For Windows only: Measurement id between 1 and 10, used for identifying a particular measurement, where the returned data represents the time interval since the same id was previously requested. Primarily for use in Dashboards.</td>
</tr>
</tbody>
</table>

Returns the last CPU measurement sample without individual CPU information unless the an id is specified.

*disk_history*

Returns a table of data where each represents one measurement. Each sample will have a PDS record for each known disk.
disk_status (filesys)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Req</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>filesys</td>
<td>String</td>
<td>no</td>
<td>File system identification, specifies a particular file system for which to get the last measurement sample.</td>
</tr>
</tbody>
</table>

Returns a data table with the last measurement information for each disk. If a file system is specified, just this measurement information is returned.

get_export (detail)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Req</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Detail</td>
<td>Integer</td>
<td>No</td>
<td>Currently not used.</td>
</tr>
</tbody>
</table>

Returns information about the host and the CDM items that are monitored. For use with dashboard discovery.

get_info

Returns information about the operating system, last sample times and probe.

loglevel (level)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Req</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Level</td>
<td>Integer</td>
<td>Yes</td>
<td>The new log level.</td>
</tr>
</tbody>
</table>

Changes the current loglevel of the probe. The probe reverts to the configured loglevel on restart.

memory_history

Returns a data table with the memory related measurements.

memory_status

Returns the last memory related measurement.

uptime

Returns the uptime of the machine.

Data collection specifics

Windows

The original version of the probe also supported Windows 95/98/ME, and remnants from this support are still present in the code. GetVersionEx is called to be able to distinguish between Windows 95/98/ME and Windows NT/2000/XP/etc.

For Windows NT/etc. Performance objects for Processor, Memory and System are fetched directly.

GetSystemInfo is used to get page size and number of processors.

Host information

The probe uses our own implementation of uname (as used in pltype), that uses various calls to detect operating system and machine specifics.

gethostname is used to get the local machine name.

GlobalMemoryStatusEx is called to get the physical memory size.

For Windows NT/etc. the “System Up Time” counter from the “System” performance object is used to get system uptime.
CPU information

Windows NT > 4 (2000/XP/2003/Vista/etc.)
Total cpu usage = "Processor", "% Processor Time", "_Total"
User = "Processor", "% User Time", pchCpu (for "_Total" as well as individual CPU)
System = "Processor", "% Privileged Time", pchCpu (for "_Total" as well as individual CPU)
Idle = 100 – User – System
For the individual cpus, total usage is calculated by summarizing User and System usage.

NT 4
Total CPU usage = "System", "% Total Processor Time"
User = "System", "% Total User Time"
System = "System", "% Total Privileged Time"
This os version did not support multiple processors in a generic way.

Note that cpu wait time is set to 0, as this is not used in Windows.

Processor queue length is fetched from the performance object "System", "Processor Queue Length" counter.

Memory information

Windows NT > 4 (2000/XP/2003/Vista/etc.)
Total memory = "Memory", "Commit Limit"
Used memory = "Memory", "Committed Bytes"

NT 4
ZwQuerySystemInformation is called to get TotalCommittedPages and TotalCommitLimit
GlobalMemoryStatusEx is used to to get page file size and available pagefile. As well as total and available physical memory.

Swap used = "Paging File", "% Usage", "_Total"
Swap size = page file size – total physical memory

Paging is fetched from performance object "Memory", counter "Pages/sec"

Disk information

Disk volumes are enumerated using the FindFirstVolume, FindNextVolume, and FindVolumeClose calls. GetVolumePathNamesForVolumeName is used to map vloume name to drive letter.

GetDiskFreeSpaceEx or GetDiskFreeSpace are used to collect disk size and usage.

Share connectivity is checked by attempting to map the drive, using netAddConnection2.

Unix/Linux

Host information

All the Unix probes use the uname() system call to gather information about the hostname, machine type, os name, os release and os version.

AIX

Boot time is read from the utmp structure gathered from the getutent() system call, and uptime is calculated based on the formula uptime = now – boot_time.
The number of processors is found from the system call
sysconf(_SC_NPROCESSORS_ONLN).

The installed physical memory is found using the system call perfstat_memory_total() from the (required) packages bos.perf.libperfstat and bos.perf.perfstat.

The page size, which is used to convert quantities measured in pages (and not MB which the probe typically uses) is found with the getpagesize() system call.

**HP-UX**

Boot time is found using the system call pstat_getstatic(), and uptime is calculated based on the formula uptime = now − boot_time.

The number of processors is found using the system call pstat_getdynamic().

The installed physical memory is found using the system call pstat_getstatic().

The page size, which is used to convert quantities measured in pages (and not MB which the probe typically uses) is found using the system call pstat_getstatic().

**Linux**

Uptime is read directly from the file /proc/uptime.

The number of processors is found using the system call get_nprocs().

The installed physical memory is found in the file /proc/meminfo.

**Solaris**

Boot time is read from the utmp structure gathered from the getutent() system call, and uptime is calculated based on the formula uptime = now − boot_time.

The number of processors is found from the system call sysconf(_SC_NPROCESSORS_ONLN).

The installed physical memory is found using the system call sysconf(_SC_PHYS_PAGES) and the pages are converted to MB based on the page size.

The page size, which is used to convert quantities measured in pages (and not MB which the probe typically uses) is found using the system call sysconf(_SC_PAGESIZE).

**Tru64**

Boot time is read using the system call table(), and uptime is calculated based on the formula uptime = now − boot_time.

The number of processors is found using the system call getsysinfo()

The installed physical memory is found using the system call table().

The page size, which is used to convert quantities measured in pages (and not MB which the probe typically uses) is found using the system call getpagesize().
**CPU information**

**AIX**

The CPU usage information is found by running the command "/usr/sbin/sar -P ALL" in a separate thread and parsing the data it provides. For each individual cpu and for the system as a whole user, system, wait and idle are collected.

The load data is found by running the command "/usr/bin/uptime". Based on the data gathering interval either the 1, 5 or 15 minute average is used.

**HP-UX**

The CPU usage information is found using the system call pstat_getdynamic(). The usage information is cumulative, so the difference between the last measurement and the current measurement is used.

The load data is found using the system call pstat_getdynamic().

**Linux**

The CPU usage information is found by reading the file "/proc/stat". Based on the data gathering interval either the 1, 5 or 15 minute average is used.

The load data is found by reading the file "/proc/loadavg". Based on the data gathering interval either the 1, 5 or 15 minute average is used.

**Solaris**

The CPU usage information is found by running the command "/usr/bin/mpstat" in a separate thread and parsing the data it provides.

The load data is found by running the command "/usr/bin/uptime". Based on the data gathering interval either the 1, 5 or 15 minute average is used.

**Tru64**

The CPU usage information is found by running the command "/usr/sbin/collect -s c -t -i" in a separate thread and parsing the data it provides.

The load data is found using the system call table(). Based on the data gathering interval either the 1, 5 or 15 minute average is used.

**Memory information**

**AIX**

The memory usage is found using the system call perfstat_memory_total() .

The swap usage is found using the system call perfstat_memory_total().

The amount of physical memory used can be calculated to either count memory used by buffers in the OS as used memory, or as we prefer to do, without these buffers counting as used memory since they are instantly available to programs that request more memory. Please note that vmstat and other utilities generally include the buffers as part of the used memory in the system, and if compatibility with these utilities is desired the appropriate flag in the configuration file should be set. To do this add the key "mem_buffer_used = yes" to the "setup" section of the cdm.cfg file.

The paging data is found by running the command "/usr/bin/vmstat" in a separate thread and parsing the data it provides. Paging is based in the "pi" and "po" columns which contain averages for the interval, and the paging is converted to KB/s based on the page size using the formula \( \text{paging_kbps} = \text{paging} \times \text{pagesize} \).
HP-UX

The memory usage is found using the system call pstat_getdynamic().

The swap usage is found using the system call pstat_getswap().

Paging data is found using the system call pstat_getvminfo(). Since the data is cumulative the difference since the last data gathering is used as the basis for calculating paging in KB/s based on the formula paging_kbps = paging_delta * pagesize / interval.

Linux

Memory and swap usage is found by reading the file="/proc/meminfo".

The amount of physical memory used is based on the formula MemTotal - MemFree - Buffers – Cached. This because buffers and cached data are instantly available to any process which requests more memory.

Paging data is found by reading the file="/proc/stat" on older Linux Kernels, and "/proc/vmstat" on newer systems. Since the data is cumulative the difference since the last data gathering is used as the basis for calculating paging in KB/s based on the formula paging_kbps = paging_delta * pagesize / interval.

Solaris

Memory usage is found using the system call sysconf(_SC_AVPHYS_PAGES). The result is the number of free pages of physical memory which is multiplied by the page size and subtracted from the physical memory.

Swap usage is found using the system call swapctl().

Paging data is found by running either "/usr/bin/vmstat –s" or "/usr/sbin/vmstat –s" (depending on which is installed on the system) and parsing the data it provides. Since the data is cumulative the difference since the last data gathering is used as the basis for calculating paging in KB/s based on the formula paging_kbps = paging_delta * pagesize / interval.

Tru64

Memory usage is found using the system call vm_statistics(). The result is the number of free pages of physical memory which is multiplied by the page size and subtracted from the physical memory.

Swap usage is found using the system call table().

Paging data is found using the system call vm_statistics(). Since the data is cumulative the difference since the last data gathering is used as the basis for calculating paging in KB/s based on the formula paging_kbps = paging_delta * pagesize / interval.

Disk information

AIX

The list of mounted filesystems is found using the system call mntctl(). Space usage on the filesystems is found using the system call statfs().

HP-UX

The list of mounted filesystems is found using the system call getmntent(). Space usage on the filesystems is found using the system call statvfs().

Linux

The list of mounted filesystems is found using the system call getmntent(). Space usage on the filesystems is found using the system call statvfs().
Solaris
The list of mounted filesystems is found using the system call getmntent(). Space usage on
the filesystems is found using the system calls statvfs() on 64bit platforms and
statvfs64() on 32bit platforms.

Tru64
The list of mounted filesystems is found using the system call getfsstat(). Space usage on
the filesystems is found using the system call statfs().

Troubleshooting

Guidelines for sending bug reports for CDM on the different platforms
When sending a bug report on the CDM probe, there is a wealth of information that is very
important to attach to the report. The following is a list of files and commands that should be run
on the affected system. The files and output from the commands should always be attached to a
bug report if at all possible. If this information is present when the bug report is filed we can save
quite a bit of time, since the first thing R&D will ask for is the information listed below!

All platforms:
cdm.cfg

Note: Loglevel should be set to 3 and the error reproduced prior to sending these files! Since this
can/will generate quite a lot of log information you can either increase the size of the log files in
the CDM gui or set the Loglevel to 13 which will not truncate the log file.

cdm.log
_cdcm.log

Windows:
The version of Windows and SP level. Individual patches applied are not necessary unless
explicitly requested by the developer. A screenshot of the Performance tab in the Task Manager
would be helpful if the problem is CPU or Memory related. A screenshot of the disks in the
Windows Explorer would be helpful for disk issues.

Unix (all platforms):
# uname –a
# mount
# df –k (on systems that support the –k option)

AIX:
# /usr/bin/vmstat
# /usr/sbin/sar –P ALL
# /usr/bin/uptime

Hp-ux:
No commands need to be run on this platform.

LINUX:
# cat /proc/stat
# cat /proc/vmstat (if available)
# cat /proc/meminfo
# cat /proc/loadavg

**Solaris:**
# /usr/bin/mpstat 60 (note: runs until stopped with Ctrl-C, get at least two iterations!)
# /usr/bin/uptime

**Tru64:**
# /usr/sbin/collect -s c -t -i0 (note: runs until stopped with Ctrl-C, get at least two iterations!)