User's Guide

Collect Version 2.0

June 2001

This guide provides information for understanding and using the Collect data collection tool and it's related utilities collgui and cfilt. It is designed primarily for system administrators.

Compaq Computer Corporation
Houston, Texas

Contents

Preface

Collect is a tool that collects operating system data under Compaq Tru64 UNIX Versions 4.*x* and 5.*x*. Collect is designed for high reliability and low system-resource overhead, and can be run in either interactive mode or historical mode.

Collect's tightly integrated associated tools, collgui and cfilt, provide filtering and display capability for collected data.

Introducing `cfilt`

The cfilt utility allows the arbitrary selection of values from the output of Collect. It condenses the output of Collect into 1 line per sample, or per a given number of samples, if using the option to average over a number of samples. The data in this form can then be graphed using gnuplot or Excel.

cfilt can also be used *live*, that is, as a filter to Collect while it's collecting and writing to standard output. This only works if no normalization is being done, as that requires that all samples be seen so that cfilt can determine the highest value, which is then used to normalize.

Introducing `collgui`

The collgui utility is intented to help evaluate data gathered by collect. It operates as a coordinator among collect, cfilt, and gnuplot. collgui automates the extraction of information from a binary data file written by Collect, and directs it to gnuplot to produce a graphical rendition of the data.

Collect features

The following table lists the main features and benefits of Collect.

Feature	Benefit
Lightweight	Collect uses less than 1% of system resources if sampling is performed at 30-second or greater intervals.
High reliability	Even if you suffer a system crash, you will still have the data preceding your last write to disk for analysis.
Highly flexible	You can configure Collect to gather data from all subsystems or any combination of subsystems, run in interactive or historical mode, and use the associated tools to display graphs.
Log scheduling and playback	You can customize both the rollover time for the log files and the length of time that the files are preserved, combine and playback multiple files simultaneously, and select playback samples for a specific time period.

Conventions

This guide uses the following conventions:

%	A percent sign represents the C shell prompt.
$	A dollar sign represents the system prompt for the Bourne, Korn, and POSIX shells.
#	A number sign represents the superuser prompt.
% **cat**	Boldface type in interactive examples indicates typed user input.
file	Italic (slanted) type indicates variable values, placeholders, and function argument names.
[|] { | }	In syntax definitions, brackets indicate items that are optional and braces indicate items that are required. Vertical bars separating items inside brackets or braces indicate that you choose one item from among those listed.
...	In syntax definitions, a horizontal ellipsis indicates that the preceding item can be repeated one or more times.
cat(1)	A cross-reference to a reference page includes the appropriate section number in parentheses. For example, cat(1) indicates that you can find information on the cat command in Section 1 of the reference pages.
Ctrl/ x	This symbol indicates that you hold down the first named key while pressing the key or mouse button that follows the slash. In examples, this key combination is enclosed in a box (for example, Ctrl/C).

1

Collect

1.1 About Collect

The collect utility is a system monitoring tool that records or displays operating system data. Any set of the *subsystems*, such as file systems, message queue, tty, or header can be included in or excluded from data collection. You can display data at the terminal, or store it in either a compressed or uncompressed data file. Data files can be read and manipulated from the command line, or through use of command scripts.

To ensure that the collect utility delivers precise statistics it locks itself into memory using the page locking function plock(), and by default cannot be swapped out by the system. It also raises its priority using the priority function nice(). However, these measures should not have any impact on a system under normal load, and they should have only a minimal impact on a system under extremely high load. If required, you can disable page locking using the -ol command option and disable the collect utility's priority setting using the -on command option.

Some collect operations use kernel data that is only accessible to root. System administration practice should not involve lengthy operations as root, therefore collect is installed with permissions set as 04750. This setting allows group (typically system) members to run collect with owner setuid permissions. If this is inappropriate in your environment, you may reset permissions to fit your needs.

1.2 Automatic Starting on a Reboot

You can configure collect to start automatically when the system is rebooted. This is particularly useful for continuous monitoring. To do this, use the rcmgr command with the set operation to configure the following values in /etc/rc.config*:

```
% rcmgr set COLLECT_AUTORUN 1
```

A value of 1 sets collect to automatically start on reboot. A value of 0 (the default) causes collect to not start on reboot.

```
% rcmgr set COLLECT_ARGS ""
```

A null value causes collect to start with these default values (command options) :

```
-i60,120 -f /var/adm/collect.dated/collect -H d0:5,1w -W 1h -M 10,15
```

You can select other values.

```
% rcmgr set COLLECT_COMPRESSION 1
```

A value of 1 sets compression on. A value of 0 sets compression off.

See the rcmgr(8) reference page for more information.

1.3 Playing Back Multiple Data Files

Use the `collect` utility with the `-p` option to read multiple binary data files and play them back as one stream, with monotonically increasing sample numbers. You can also combine multiple binary input files into one binary output file, by using the `-p` option with the input files and the `-f` option with the output file.

The `collect` utility will combine input files in whatever order you specify on the command line. This means that the input files must be in strict chronological order if you want to do further processing of the combined output file. You can also combine binary input files from different systems, made at different times, with differing subsets of subsystems for which data has been collected. Filtering options such as `-e`, `-s`, `-P`, and `-D` can be used with this function.

1.4 Normalization of Data

Where appropriate, data is presented in units per second. For example, disk data such as KiloBytes transferred, or the number of transfers, is always normalized for 1 second. This happens no matter what time interval is chosen. The same is true for the following data items:

- CPU interrupts, sytem calls, and context switches.
- Memory pages out, pages in, pages zeroed pages reactivated and pages copied on write.
- Network packets in, packets out, and collisions.
- Process user and system time consumed.

Other data is recorded as a snapshot value. Examples of this are: free memory pages, CPU states, disk queue lengths, and process memory.

1.5 The Data Collection Interval

A collection interval can be specified using the `-i` followed by an integer, optionally followed (without spaces) by comma or colon and another integer. If the optional second integer is given, this is a separate time interval which applies only to the process subsystem. The process interval must be a multiple of the regular interval. Collecting process information is more taxing on system resources than are the other subsystems and is not generally needed at the same frequency. Process data also takes up most space in the binary data-file. Generally, specifying a process-interval greater than 1 will significantly decrease the load the collector places on the system being monitored.

1.6 Specifying What Data to Collect

Use the `-s` (select) option to select subsystems for inclusion in the data collection, or use the `-e` (exclude) option to exclude subsystems from the data collection.

When you are collecting process data, use the `-S` (sort) and `-n`*`X`* (number) options to sort data by percentage of CPU usage and to save only *`X`* processes. Target specific processes using the `-P`*`list`* option, where *`list`* is a list of process identifiers, comma-separated without blanks.

If there are many (greater than 100) disks connected to the system being monitored, use the `-D` option to monitor a particular set of disks.

1.7 Data Compression

The `collect` utility reads and writes `gnuzip` format compressed datafiles. Compressed output is enabled by default but can be disabled using the `-oz` command option. The extension `.cgz` is appended to the output filename, unless you specify the `-oz` command option. Older, uncompressed datafiles can be compressed using `gzip`, and the resulting files can be read by collect in their compressed form.

Compression during collection should not generate any additional CPU load. Because compression uses buffers and therefore does not write to disk after every sample, it makes fewer system calls and its overall impact is negligible. However, because the output is buffered there is one possible drawback. If collect terminates abnormally (perhaps due to a system crash) more data samples will be lost than if compression is not used. This should not be an important consideration for most users, as you can specify how often data is written to the disk.

1.8 Specifying a Time Range from a Playback File

You can select samples from the total period of the time that data collection ran. Use the `-C` option to specify a start time and optionally and end time. The format is as follows:

`[+]Year:Month:Day:Hour:Minute:Second`.

The plus sign (`+`) indicates that the time should be interpreted as relative to the beginning of the collection period. If any of the fields are excluded from the string, the corresponding values from the start time are used in their place as the time-value is parsed from right to left. Thus, field one is interpreted as `Second`, field two (if there is one), as `Minute`, and so on. For example, if the collection period is from October 21, 2000, 16:44:03 to October 21, 2000, 16:54:55, and you wish to extract one minute, all but minutes and seconds can be ommitted from the command option: `-C46:00,47:00` (from `16:46:00` to `16:47:00`). However, if the collection ran overnight, it is necessary to specify the day as well. For example, when the period is Oct 21 16:44 to Oct 22 9:30, enter the following command to specify a time range from `23:00` to `1:00`:

```
# -C 21:23:00:00,22:1:00:00
```

1.9 General Command Options

The following command options are useful:

Use the `-a` option to display simultaneous text (ascii) output to the screen while collecting to a file, .

Use the `-t` option to prefixe each data line with a unique tag. This makes it easier for your scripts to find and to extract data. Tags are superfluous if you use the `perl` script `cfilt`.

Use the `-T` option to shut off collection for all subsystems except disk, and only display a total MB/sec across all disks in system. Use the `-s` option with the `-T` option to override this behavior and collect data for other subsystems.

The `-R` option causes collect to terminate after a specified amount of time.

All flags that can reasonably be applied to both collection and playback will work. The `-Plist` filter option used during collection will collect data only for the processes you specify. During playback it will only display data for the corresponding processes. To save space in the binary data file, you can limit your collection to specific processes, specific disks, or specific subsystems. However, if you want to look at volumes of data and select different chunks at a time, you should collect everything and later use the filter options to select data items during playback.

1.10 Disk Statistics

Note that under certain circumstances the Disk Statistics may be only approximate. For older releases of `collect`, some data fields were zero and data in some fields could be inaccurate under certain circumstances.

1.11 Data Conversion and Filtering

Collect automatically reads older datafile versions when playing back files.

You can convert an older `collect` version datafile to the current version using the `-p` *`collect_datafile`* option with the the `-f` *`file`*. During conversion you can use most command options to extract specific data from the input *`collect_datafile`*. For example:

- Use the `-s` and `-e` options to select data only from particular subsystems.
- Use the `-n`*`X`* and `-S` options to take only *`X`* processes and sort them by CPU usage.
- Use the `-D` option to select disks and the `-L` option to select LSM volumes.
- Use the `-P`, `-PC`, `-PU`, `-PP` options to select processes based on their identifiers.
- Use the `-C` option to extract data according to specified start and stop times.

2

cfilt

With the filtering capability of cfilt you can fine tune Collect to provide the specific values you wish to focus on, without the clutter of all the system information. The utility provides a number of options to allow selection, averaging, and choosing between real-time and historical operation modes.

2.1 Options

The following options are available for cfilt:

- -p

  This option selects only the samples that contain process data. This is useful when a separate process interval was given to Collect, such as -i1,4, but you want to graph process data against some non-process data, such as cpu idle.

- -a[*number*]

  This option averages values for the specified number of samples.

- -f[*input-file*]

  This optionally reads the specified *input-file* instead of standard input. This can also be a binary Collect file, in which case Collect is run as a pre-processor.

Epressions are also accepted as options.

2.1.1 Expressions

An expression has the following syntax:

```
<subsystem>:<selection-criterion>:<tag-expr1>:<tag-expr2>:<...>:<tag-exprN>
```

A subsystem can be one of: *proc*, *disk*, *mem*, *net*, *cpu*, *sin*, *file*, *tty*, and *lsm* (first 3 characters are significant)

If a plus-sign (+) is on the end, or no selection criterion has been given, then numerical values are summed for all lines of a subsystem. If a selection criterion has been provided, and there is no plus sign on the end of the subsystem name, then for each value in the selection criterion, the corresponding values for each <tag-expr> will be printed. For example, given the following output from Collect:

```
# DISK Statistics
#DSK     NAME    B/T/L     R/S    RKB/S    W/S    AVS    QLEN    %BUSY
0        rz1     0/1/0     5      300      10     10     0       70
1        rz2     0/2/0     7      400      11     10     0       80
2        rz3     0/3/0     9      500      12     10     0       90
```

Assuming that *cfilt* is called with the single following expression,

```
disk:r/s
```

cfilt would sum reads/second for all disks. That is, 5+7+9=21. The output of cfilt would be:

```
<epoch-seconds> <sample#> 21
```

The expression `disk+:name=rz1,rz2:r/s` would sum reads/second for disks rz1 and rz2, 5+7=12. (*name=rz1,rz2* is a selection-criterion, which is discussed below.) The output of *cfilt* would be:

```
<epoch-seconds> <sample#> 12
```

The `expression disk+:name=rz1,rz2:rkb/s+wkb/s` would sum KiloBytes read and written for disks rz1 and rz2, 300+400+1000+2000=3700, as follows (`rkb/s+wkb/s` is a *tag-expression*, which is discussed below.):

```
<epoch-seconds> <sample#> 3700
```

The expression `disk:name=rz1,rz2:r/s` would print reads/second for rz1 and reads/second for rz2, as follows:

```
<epoch-seconds> <sample#> 5 7
```

2.2 Selection criterion (optional)

A selection criterion is a field tag (see *tag-expr*) on the left of an equals-sign, and a comma-separated list of values in that field that should be selected.

```
<tag>=<value>[,<value>[,<value>[...]]]
```

Examples: pid=1234,1235,8888, command=init, name=rz0,rz1

2.2.1 tag-expr

Tags are the column labels used by Collect. For example, in the disk subsystem, the tags are *dsk*, *name*, *b/t/l*, *r/s*, and so on. A tag-expr can be anything from a complicated arithmetic expression to simply the name of a collect output field, such as *rss*.

Arithmetic expressions:

tag1+tag2	add values tag1 and tag2
tag1-tag2	subtract
tag1*tag2	multiply
tag1~tag2	divide tag1 by tag2
log(tag1)	functions
(100-tag1)~tag2	constants and grouping

If a number sign (#) is appended to the <tag-expr>, all values are normalized to 100, or if an integer follows the number sign, then it is used instead of 100. This is useful for graphing results simultaneously

The available functions are: cos, sin, tan, sqrt, log, exp, abs, atan2, int, and convtime for converting Minutes:Seconds.TenthsHundreths to Seconds.TenthsHundreths.

If a subsystem has multiple lines/sample, values are added for all lines that match in one record. (if no selection critieron, then all are taken)

Note

No white space is allowed in expressions.

If using parenthesis for grouping or functions, be sure to surround the expression in single-quotes

You can only give one expression per subsystem to cfilt. This means, among other things, that you can't have summed data for a subsystem and individual graphs for all the things you summed.

The first two columns of output are always <epoch-seconds> <sample#>.

2.2.2 Reading output

The first two columns in cfilt's output are always the sample number and the epoch-second. The epoch-second is the internal UNIX time format, the number of seconds since the beginning of the *epoch*, January 1st, 1970. This is extracted directly from the Collect output. At the beginning of each record there is a line similar to the following:

```
#### RECORD 1 (873230968:160) (Tue Sep 2 22:09:28 1997) ####
```

in this example, epoch-seconds is 873230968. The sample number is also extracted from this line. In this example it is 1.

2.3 Examples

The following are examples of cfilt usage:

```
cfilt -fdata.in cpu:user+sys:intr%:sysc%:cs%
```

Output: <seconds> <sample#> <user+sys> <interrupts> <syscalls> <conswitch> (where interrupts,syscalls,conswitch are normalized)

```
cfilt -fdata.in proc+:user=urban:rss
```

Output: <seconds> <sample#> <RSS (resident set size) for all processes owned by urban>

```
cfilt -fdata.in cpu:idle net:inpck+outpck% mem:free%
```

Output: <seconds> <sample#> <cpu:idle> <net:inpck+outpck(normalized)> <mem:free(normalized)>

```
cfilt -fdata.in pro:pid=1234,8888:rss:vsz
```

Output: <seconds> <sample#> <rss(pid=1234)> <vsz(pid=1234)> <rss(pid=8888)> <vsz(pid=8888)>

```
cfilt -fdata.in pro+:pid=1234,8888:rss:vsz
```

Output: <seconds> <sample#> <rss(sum for pid 1234,8888)> <vsz(sum)>

```
cfilt -fdata.in pro+:rss:vsz
```

Output: <seconds> <sample#> <rss(sum all procs)> <vsz(sum for all procs)>

3

collgui

The collgui utility relies on cfilt to evaluate data gathered by Collect. Therefore, understanding cfilt, especially if you want to do complicated or nonstandard operations will help you get the most out collgui.

The data extraction performed by collgui is robust but not quick. Processing the enormous amount of data that Collect can generate takes some time, so collgui offers two different methods for selecting samples for graphing.

- You can set the `START` and `END` times. These arguments get passed to Collect, so, if your data-file contains lots of samples, but you only want a fraction of them, using `START` and `END` will substantially speed up your extraction because the selection is handled by collect itself.
- You can set an X Range for gnuplot. This has a similar effect as setting `START` and `END`, but Collect provides all samples, and cfilt must extract for all selected subsystems from this data. This is slower, with a difference where the time selection is done. You may want to set `START` and `END`, and still set the X Range in order to give gnuplot explicit instructions as to what should be displayed.

When you save a user-defined setting/configuration, a unique ID is saved with it, consisting of filename (no path) plus file size. When you recall this setting, if the unique ID of your current open data file matches the saved one, things like `START`, `END`, X-range, Y-range, average samples, X- units, and *samples w/process data* are also restored. If the unique ID's don't match, then only the subsystem settings are restored.

3.1 Selection Mechanisms

The mechanism by which one of many objects (such as LSM Volumes, Disks, Tapes, Single CPUs) can be selected is a bit particular. If there are less than a fixed number of objects (~30), a MenuButton is created (when *Add* is pressed a vertical list is presented). If the the number is greater than this constant, a separate window is created with a listbox containing all possible objects. A double-click on an object in the listbox will add it to the selection listbox.

The selection mechanism for processes deserves special mention: this is always a separate window with a listbox and a slider marked *sample* and a button marked *List Processes* next to it. Using the slider, a sample (record) can be selected from the collection period, and double clicking on a process will enter its PID in the selection listbox.Processes are selected using their PIDs. At the top of each column is a button that turns red when the mouse is over it. Pressing the button will sort the list using the values in the button's column.

3.2 Interface

This is a description of the main window of collgui, from top to bottom:

Menu

- File Open

  Opens a collect binary data file and extracts information.

- Exit

  Quits gracefully.

Options

- Legend Position

  Sets the position of the labels for the various lines graphed.

- X Axis Label

  Label the X Axis. This can save valuable graph real estate.

- X Time Format

  Controls the format of the X-Axis tic-mark labels.

- Set Y Label

  Allows users to specify the label for the Y-Axis, rather than using the default, KB/`Transfers/Packets/Pages/etc`.

- Image Format

  Allows users to chose JPEG, PPM (Portable Pixmap - Color), or PBM (Portable Bitmap - Grayscale).

Settings

- Save

  Saves the current configuration — that is, all the information needed to reproduce the current graph. The information is saved in `$HOME/.collguirc` and read in on start.

- Delete

  You are presented a list of user-defined settings. Double clicking on an entry will remove it from the list, clicking on *commit* will save your changes.

- Built In

  Basic settings for looking at data.

- User Defined

  Choose settings previously stored.

`FILE` indicator, `START` and `END`

- FILE indicator
- Shows you the name of the currently open collect binary data file.
- START and END

  These areas allow you to specify a time range for samples to be extracted from the binary data file. They are set to the times of the first and last sample respectively (that is, the whole run) when you open a file. The RESET button at the bottom of the window will restore these to their default values. These values are passed directly to Collect during playback, so this is the fastest method for extracting a sub-range from your collection period.

- X: From: To: and Y: From: To:

These areas allow you to specify gnuplot X and Y ranges. The X entry- windows are wider than those of Y because you can specify a time when *time* is chosen for X Axis Units.

- Average Samples and X Units:

  The average samples area allows you to cause cfilt to take an average over *n* samples. That is, for *n* samples read from the binary data file, cfilt will produce 1 output line with average values. X Units allows you to specify either *time* or *samples* for the horizontal axis of the graph.

- Samples w/Process Data:

  This solves the problem of graphing intermittently gathered process data against constantly gathered other data. For example, if you specified an interval for collect of `-i1,4`, (optional syntax: `-i1:4`) thereby collecting process data only every four seconds, and you try to graph that against cpu idle time, which was gathered every second, you will get zeros for the process data for samples in which no process data appears. Clicking on the *Samples w/Process Data* check button will cause, via `cfilt -p`, only samples with process data to be used. In the above example, it would be the equivalent of taking every 4th sample (actually it would mean takings samples #1,5,9,13,17, and so forth).

- Subsystem Widgets:

  The first 8 subsystems have multiline output from collect. Therefore it possible to select using certain criteria (see cfilt).

  – Single CPU – Processes – Disks – LSM Volumes

  – Tapes – Message Queues – Network – File Systems

  The last 3 only have a single output line from collect.

  – CPU Summary – Memory or Swap – Terminal I/O

  When you open a binary data file, the Add button and collected data are shown. Without the file, Expressions will be grayed-out, and the Add button will say “No Data”. The widget controls give you a graphical front-end to cfilt. The Expressions menu offers a confortable way of choosing cfilt expressions. The sum check button corresponds to the plus sign at the end of the subsystem name in an expression. The values in the list box correspond to the selection criterion for cfilt, that is =,,...,. You may type in the expression window itself. If you specify an illegal expression, you will get an error message.

3.2.1 Subsystem Expressions

Single CPU

- SMP Stack

  This will create a graph consisting of a set of horizontal bands, one per CPU. Best practice is to select all CPUs when using this display option.

CPU Summary

- State Stack

  This graphs four values for all CPUs: USER+SYS+IDLE+WAIT, USER+SYS+IDLE, USER+SYS, and USER, as in SMP Stack under Single CPU.

3.2.2 DISPLAY, ALL, PRINT, {JPEG|PPM|PBM}, and RESET buttons:

- DISPLAY

  Graphs your selections to the gnuplot X11 output device.

- ALL

  Refreshes the display of all Gnuplot windows. Use it after changing the time range.

- PRINT

  Sets the device to postscript, and asks you for an output filename (which may be routed directly to the printer).

- {JPEG | PPM | PBM}

  Produces an image file in the corresponding format.

- RESET

  Clears most settings, and sets the rest to reasonable startup values (like `START` and `END` time).

3.3 Quick Start

The following example serves as a quick guide for those who don't need to learn cfilt first.

1. Choose *Disks*.
2. Click on *Expressions* and select *KB/Sec*. Without selecting any specific disks, click on the *DISPLAY* button, and you will get the TOTAL KiloBytes/Second throughput for all disks for which data was collected.
   - Data is totalled because the list on the right is empty. cfilt assumes, since you have not selected any particular disk(s), you want a grand total.
   - If, however, you now add *rz0* and *rz1* (assuming these disks exist on your system, and you collected data for them), you will now get two lines graphed, KB/Sec for rz0 and KB/Sec for rz1.
   - Click on *Expressions* and select *%Busy*. You will get 4 lines: KB/Sec and %Busy for rz0, and KB/Sec and %Busy for rz1.
   - If you click on the *sum* checkbutton, (and the *DISPLAY*), you will get only 2 lines this time: KB/Sec for rz0+rz1, and %Busy for rz0+rz1. *sum* sums over all objects in the list box, or over all objects for which data was collected if no specific object has been selected (that is, the list box is empty). It is sometimes useful to graph dissimilar data together, for example cpu idle and disk KB/sec.
3. Gnuplot has one vertical scale. In order to get such incongruous data together in a reasonable fashion on the same graph, data may have to be normalized (scaled to fit into a particular range, typically 0-100). Placing a number sign (#) on the end of an expression will cause this data to be normalized. For typical expression possibilities, Normalized and Raw options are offered. The only difference is the percent sign on the end. You can also choose the end of the normalized range yourself by giving that value after the number sign, for example: `disk:rkb/s+wkb/s#150`.

3.3.1 X Resources

collgui relies on the default colors of Tk, which is usually OK. However, under CDE there are problems. If you have a problem seeing text in the entry widgets, try putting the following line in your ~/.Xdefaults file.

```
Collgui*foreground: black Collgui*background: white
```

Merge this change into your in-memory resource database with this command:

```
xrdb -merge ~/.Xdefaults
```

3.3.2 Environment Variables

- COLLGUI_PRINT

  The default printer file, can also be `|lpr -lpr -flags -here` (see gnuplot help *set output*)

- COLLECT

  The name/path of collect, if not collect, or if collect is not in your path (for example `collect3` or `/usr/foo/bin/collect4`)

- CFILT

  The name/path for cfilt, if not cfilt, or if cfilt is not in your path .

- GNUPLOT

  The name/path for gnuplot, if not gnuplot, or if gnuplot is not in your path.

- CJPEG

  The name/path for the cjpeg program, used to convert PPM image files to JPEG.