In order to map system performance to desktop performance, we developed a method for coupling system monitoring with desktop benchmarking. This method consists in observing low-level system parameters while desktop operations take place and describing correlations. For more information see here .

For monitoring VDD system we use dstat , a simple and versatile tool for generating system resource statistics.

```
root@serv1:~# aptitude install dstat
```

Dstat monitors system state instant after instant and generates tables with configurable rows and columns. This command:

```
root@serv1:~# dstat -t -cC 0,1,total -l -ms -n -dD sda1,sda2,total --noheaders --output example.csv
```

dynamically generates a table in the terminal output, one raw per second, having the following columns:

- The first column is time (-t option), given in the form hh:mm:ss. Dstat provides up to one shot per second;
- Second group of columns regards CPU. Six parameters for every CPU (or CPU core) are available: usr, sys, idl, wai, hiq, siq. In the example, in a system with two CPUs, cpu0, cpu1 and the total (-cC 0,1,total) are shown;
- -l option is the classic load avarage in three values: 1m, 5m and 15m (one minute, five minute and fifteen minutes);
- -ms option regards memory (used, buffered, cached, free) plus swap (used, free);
- -n enables network monitoring, a simple byte received, sent scheme;
- The group of columns disk usage is constituted by sda1, sda2 and total (-dD sda1,sda2,total). For each one, bytes read and written are monitored. Of course, it is possible to add sda3, sda4, etc. and it can also be done on a device base instead (sda, sdb, etc.);
- The option --noheader skips some initial output that could interfere with a subsequent process of analysis (not all, though, but thanks to this patch it can be limited);
- The last option (--output) provides a CSV formatted file as output (in addition to the terminal output shown above, that can be redirected to another file too, of course).

If one it is not interesed in monitoring each CPU plus the total, and each disk/partition plus the total, command could simplify this way:

```
root@serv1:~# dstat -t -c -l -ms -n -d --noheaders --output example.csv
```

VDD host and every virtual machine in use (corresponding to one or more VDD desktops dispatched and in use) are monitored with dstat during tests. For every desktop under test, two dstat tables have to be analyzed: the one of the corresponding virtual machine and the one of the host. These are two CSV files (generated by dstat) that have to be imported in OpenOffice Calc for inspection and analysis and to generate graphs if and where needed.

In order to simplify this process, we developed the macro sys_graphs . The macro must be registered inside OpenOffice (tools -> macro -> organize macro -> OpenOffice.org Basic -> New -> copy and paste the macro.

The function sys_graphs can then be called, specifying four parameters:

1. the name and type of desktop (example: Jaunty XFCE);
2. the number of the raw of the sample of interest;
3. how many samples before that one to consider;
4. how many samples after that one to consider.

=SYS_GRAPHS(“Desktop”;sample_number;left_extent;right_extent)

Suppose, for example, that an anomaly or slow down is reported at a particular timestamp (hh:mm:ss). One can see what is the sample number at that timestamp (an unsigned int corresponding to the spreadsheet row number). Say it is 17:00:00.

=SYS_GRAPHS(“Jaunty XFCE”;5785;50;50)

5785 is the number of the raw corresponding to that time. We will get graphs from 50 seconds before 17:00:00 and till 50 seconds after, of Jaunty XFCE desktop and of VDD host.

The table containing the data of the host must be put on a second spreadsheet tab (the table of the virtual machine is on the first tab by default when the CSV file is imported). The macro generates 6 graphs per tab (so, 12 in total) appropriately formatted, corresponding to system perfomance measures (CPU usage, load avarage, memory usage, swap usage, net load, disk load). Despite the use of --headers option and of our dstat patch , it is worthwhile to double check that the second raw of the spreadsheet is the one where the heading of columns is written (the first one must be empty and can be used for launching the function of the macro).

Using this procedure, one can produce graphs to be inspected in a few seconds or re-generated with a larger or smaller range of samples as needed. One or two hundreads of samples (say 50-100 on the left and 50-100 on the right) are usually enough and the resulting graphs are still

quite readable (too many samples make the x axis too crowded).

Desktop benchamrks

We chose the following operations to be performed on VDD desktop during tests.

Microbenchmark

- VDD Desktop start up: select a Desktop to be dispatched on your terminal;
- Responsiveness of DE menus: browse menus using a fixed sequence;
- Start up user programs: OpenOffice Write, OpenOffice Calc, OpenOffice Impress, Mozilla Firefox, Gimp, Mplayer, Thunderbird, acroread;
- Copy and move files of fixed dimension: 100 KB, 10MB, 100 MB;
- Switching between 2 VDD desktops;
- Switching VDD desktops with full load (switch forward and backward all desktops).

Task-oriented benchmark

- Multimedia multitasking test: convert a file containg a video with a defined quality (encoding, fps, resolutions and megabyte) from mov to mpeg. The video conversion takes place in the foreground (in order to test how long it takes a system to perform only the conversion). This test exercises almost every major subsystem, including CPU, memory, and hard drive. Desktops with multicore CPUs are likely to perform better than comparable systems that use CPUs with fewer cores or single-core CPUs;
- GIMP image-processing test: apply a filter to an image of defined quality (camera RAW image files captured from a X-megapixel camera). Examples of filters could be Unsharp Mask, Lens Correction, and Dust and Scratches filters; as well as reducing image noise and converting the images to grayscale JPEGs. This test primarily exercises a system's CPU, memory, and chipset subsystem, but it also utilizes the graphics and hard drive subsystems to some extent. Some of the filters we use in the GIMP test can use multithreading, so desktops with multicore CPUs are likely to perform better than comparable systems that use CPUs with fewer cores or single-core CPUs;
- Encoding test: convert tracks of a music album (e.g. 10 tracks) to MP3 files. This test almost exclusively exercises a system's CPU capabilities. It supports multithreading, so desktops with multicore CPUs are likely to perform better than comparable systems that use CPUs with fewer cores or single-core CPUs;
- OpenOffice productivity test: execute defined operations (using macros can be a good

option). Operations under OpenOffice Write could be searching and replacing, changing font sizes, and creating columns. About OpenOffice Calc, operations could be performing functions on a spreadsheet, such as solving formulas and creating charts. In OpenOffice Impress, we could add graphics and text and moves images around in a presentation. This test exercises nearly every major subsystem, including CPU, memory and hard drive;

- Cinelerra test: perform rendering. This test almost exclusively exercises a system's CPU capabilities. Cinelerra supports multithreading, so desktops with multicore CPUs are likely to perform better than comparable systems that use CPUs with fewer cores or single-core CPUs.

- Note: this set of tests is inspired by http://reviews.cnet.com/4520-6603_7-5020816-1.html

Application microbenchmark

- OpenOffice Write: Search and replace (e.g. "a" arial at 12 → "A" bold verdana at 14)
- OpenOffice Calc: Sum up a certain amount of numbers in a column
- OpenOffice Impress: Open an animated presentation
- Firefox: Open page with fixed content and dimension (either local file system or apache web server could be used)
- Gimp: Open an image with defined quality
- Mplayer: Play a song with defined quality
- Thunderbird: Import address book
- Acroread: Search a word in a document

Batteries of tests

- Battery 1: one user for one desktop at a time. All operations in Section 2.3 are performed twice (to avoid caching distortions). This battery verifies if peformance of a VDD desktop differ from performance of a standard local and physical desktop. Another result of the battery is the comparison of performance among desktops.
- Battery 2: ten different users on the same desktop from ten terminals. Repeat same tests, in a random order for each user, and without taking care of caching distortions (irrelevant in a random situation).
- Battery 3: ten different users on ten different desktops from ten terminals. Repeat same tests, in a random order for each user, and without taking care of caching distortions (irrelevant in a random situation). Battery 2 and 3 must not be done the same day (in order to keep the experience as random as possible).

- Battery 4: one user on one desktop. Repeat tests with artificial stressing of VDD. The traffic for stressing must occur from terminals: nine terminals using the same desktop or nine terminals using different desktops. Three stressing profiles are defined (low, intermediate, high), to be calibrated during tests, based on levels of I/O, memory access, CPU usage and network traffic. Tools for stressing systems: CPUBurn (http://pages.sbcglobal.net/redelm/), Mersenne Prime (GIMPS) (http://mersenne.org/freesoft/).

The full report can be downloaded here .

{backbutton}