CERN School of Computing

Data Technologies – CSC 2009

Lectures: Alberto Pace, Bernd Panzer-Steindl

Exercises: Andreas-Joachim Peters

Email Andreas.Joachim.Peters@cern.ch

Exercises Data Technologies

1 IO Measurements Basics

Introduction:

A key point in development for data management (DM) is an objective performance evaluation of the underlying hardware and software data management tools.

Most Linux distributions have simple and very useful command line tools for IO performance measurements and the understanding of IO patterns installed. Some of the tools are actually 'disturbing' the measurements. Therefore we need a good understanding how this tools (might) influence a measurement and how we can use them for performance evaluation and measurements. In the first set of exercises we will use the following Linux commands to do some simple performance measurements and measure some basic IO parameters of our exercise desktops:

- `time`
- `top`
- `iostat`
- `vmstat`
- `yes (dd)`
- `strace`

If you don't know some of these commands, use the man pages to get an idea about syntax and their description:

```
[ csc48 ] man strace
```

A generic mechanism in Linux (you hopefully know) is the redirection of STDOUT (+STDERR) of programs or shell commands. In the following exercises we need to redirect the output of the above listed commands into files for IO measurements or to review the output 'offline' with an editor. In some cases it is also useful to 'dump' all output into a NULL device called '/dev/null'.

Examples:

Redirect STDOUT+STDERR into a file:

```
bash> <cmd> >& /tmp/myfile
tcsh> <cmd> >& /tmp/myfile
```

Just dump STDOUT+STDERR:

```
bash> <cmd> >& /dev/null
tcsh> <cmd> >& /dev/null
```

There are more complex redirections possible which we skip for simplicity.

In all exercises KB,MB,GB etc. express multiples of 1024 not 1000!

Data Technologies – CSC 2009

Lectures: Alberto Pace, Bernd Panzer-Steindl

Exercises: Andreas-Joachim Peters

Email Andreas.Joachim.Peters@cern.ch

Exercises

1.1 TIME

Measure the execution time of the command '*sleep* 1' in the **bash** shell!

1.1.1 Why is the realtime not exactly 1 second?

1.1.2 Give a lower boundary for the precision of a realtime measurement using time in the bash shell!

1.1.3 How would you do a more precise realtime measurement if you implement the sleep command yourself?

1.2 TOP-VMSTAT

Try to startup the *top* command and ...

1.2.1 Identify the process consuming most of your physcial memory sorting the output by memory consumption!

1.2.2 How can you switch back to the task window and change the update frequency of the top window to 2 Hz !

Switch the frequency to 100 Hz and run in a different window a *vmstat* command with a 1Hz update time. In case you don't know *vmstat*, look at the man page!

1.2.3 What percentage of total CPU time is used by your top window? Would you recommend to run top windows with a high update frequency for measurements?

1.3 IOSTAT-YES-STRACE

Run a *vmstat* and an *iostat* (use -x) command with 1 Hz update time in separate windows.

Redirect the output of the *yes* command into a file in */tmp* and let it run for ~4s.

1.3.1

Measure the execution time with the time command and stat the size of the created output file.

What is the IO rate you get in MB/s ? (1M = 1024*1024 bytes)

1.3.2 - Inspect system calls using the strace command in front of the yes command with output to STDOUT
- Inspect system calls using the strace command in front of the yes command redirecting the output to */dev/null* (or a file)

What is the difference between the two or more precisely what seems to happen during shell redirection?

1.3.3 Run again the yes command for ~4s providing the output string '12345678' and redirect the output into a file in */tmp/.* Measure the execution time with the time command and look at the size of the created output file.

Calculate the IO rate in MB/s and compare to the measurement under 1.3.1.

1.3.4 Rerun the previous command and leave it running without interruption.

Which basic information can you extract from the vmstat and iostat windows while it runs?

Which cache strategy is used by the OS:

1. no cache
2. write-through cache
3. write-back cache

Cleanup all the possibly very large files you created in /tmp/ during the exercise!!!!

1.4 CP

Use the strace command to inspect only *open,read,write,close,stat* calls for the following commands:

- `cp /etc/passwd /tmp/passwd`
- `cp /usr/bin/vim /tmp/vim`

1.4.1 Which IO pattern is used to do the copy?

1.4.2 Specify the realtime overhead in percent running the 2nd copy command with strace.

1.5 System-Call Tracing

strace is a very valuable tool to debug IO related problems in executables. Moreover it is usefull to understand binary programs without having source code at hand (e.g. you can attach strace to a running program like you do with a debugger).

Try to investigate using** strace **and** top **what the execution of the mysterious program** /home/peters/public/bash **does. Be careful not to loose track of it!