Performance analysis, development and improvement of programs, commands and BASH scripts in GNU/Linux systems

Erion ÇANO

Prof. Dr Betim ÇIÇO

11TH WORKSHOP "SOFTWARE ENGINEERING EDUCATION AND REVERSE ENGINEERING"

OHRID, REPUBLIC OF MACEDONIA 22TH AUGUST – 27TH AUGUST 2011

Outline

- Motivation
- Shell performance bottlenecks
- Some performance improvement techniques
- Preliminary results

Why to improve Bash performance

- Bash is the default shell in all GNU/Linux systems which are gaining significant popularity
- Shell language is slower than other powerful scripting languages like Perl and Python
- Administrative scripts are getting larger

Shell Performance Bottlenecks

As soon as a user types a commend or the execution flow of a script reaches an external commend, the shell slows down by:

- Searching through the PATH directories to find the corresponding binary
- Loading the binary in the memory
- Starting a new process (forking)

Searching through the PATH

The PATH environment variable is a chained list of directories that contain the most important binaries of the system. The PATH in my system is:

/home/cano/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games

The shell has to search the contents of all these directories to find the matching command.

How considerable is the average time cost of the search?

How to reduce search time

A Hash Table with the recently used commands is kept in memory. Each entry keeps the name of the command, its path and the number of hits.

If the required command is found in the hash table there is no need for searching $PATH variable.

What is the average performance improvement of the hash table?

Loading and Forking

More costly than searching the PATH.

Any possible access (when there is no caching) to the filesystem creates delays in the order of milliseconds.

System calls (fork and exec) have the same effect (kernel overhead).

Techniques to reduce execution time

- Preloading code in memory (reducing I/O)
- Averting subprocess creation or command substitution (averting fork)
- Using the best syntactic features of the language for script performance-oriented optimizations (style guides)

Reducing I/O activity

- The shell supports the creation of functions which contain commands just like any script.
- The primary reason for using functions is the modularization of the large scripts (problems) same as in conventional programming.
- As soon as the interpreter reaches their definition, their content is loaded in the memory. This reduces the I/O activity (potential speed up)

Reducing I/O activity (2)

Function libraries can also be built. This functions are sourced from ".bashrc" file.

What is the performance effect of using functions?

Is the relation between binary size preloaded in the memory and execution time reduction (almost) linearly proportional?

Using shell built-ins

Built-in commands are part of the shell code. No need to search the path, no need to load in memory, no process creation.

Maximal performance achievable.

Problem: A very small part of the binaries are shell built-ins…!

Dynamic-Loadable built-ins

External programs compiled and linked as shared objects in memory.

Can be enabled and disabled with “enable” command .

Are loaded in the memory from the shell when it starts up and perform the same as static built-ins.

Converting external commands into dynamic-loadable built-ins significantly improves their performance.

External commands vs. Functions

A function with just one command inside runs the same as the external command it contains (for 100 executions)

A function with 10 commands inside runs 30% faster than the script with the same commands.

Why? What about functions containing more commands or scripts with many functions

External Commands vs. Built-ins

Echo is one of the most used commands in shell scripts. It is both a shell built-in and an external Command (/bin/echo)

"echo" time: 20 ms (100 executions)
"/bin/echo" time: 330 ms (100 executions)

Speedup factor: 16.5 !

SUGGESTIONS / QUESTIONS ?

THANK YOU!