On Performance

markus schnalke <meillo@marmaro.de>

Contents

Some mediations on software performance

- Case studies
- How to and how not to optimize
- A different view angle on performance

What is Performance?

Definition

Computer performance is characterized by the amount of useful work accomplished by a computer system compared to the time and resources used.

Depending on the context, good computer performance may involve one or more of the following:

- Short response time for a given piece of work
- High throughput (rate of processing work)
- Low utilization of computing resource(s)
- High availability of the computing system or application
- Fast (or highly compact) data compression and decompression
- High bandwidth / short data transmission time

— Wikipedia: Computer performance

The Useless Use of Cat Award

Jargon File: UUOC

[from the comp.unix.shell group on Usenet]
Stands for Useless Use of cat; the reference is to the Unix command cat(1), not the feline animal. As received wisdom on comp.unix.shell observes, ‘‘The purpose of cat is to concatenate (or ‘catenate’) files. **If it’s only one file, concatenating it with nothing at all is a waste of time, and costs you a process**.’’ Nevertheless one sees people doing

```
cat file | some_command and its args ...
```

instead of the equivalent and cheaper

```
<file some_command and its args ...
```

or (equivalently and more classically)

```
some_command and its args ... <file
```

Since 1995, occasional awards for UUOC have been given out, usually by Perl luminary **Randal L. Schwartz**.

UUOC example

Newsgroups: comp.unix.shell
Date: 24 Oct 1994 03:53:05 GMT
From: merlyn@stonehenge.com (Randal L. Schwartz)
Subject: Re: Help a newbie with splitting files .....
Message-ID: <MERLYN.94Oct23205305@linda.teleport.com>

>>>>> "Tony" == Tony Nugent <T.Nu...@sct.gu.edu.au> writes:

Tony> `cat file | sed -e '/^==========/,$d' > part.1`

Wow. This week's "**Useless Use of Cat Award**" is being handed out on the first day of the week. I can rest now. :-)

Hint: whenever cat has one argument, or no arguments, it's not *concatenating* anything, and can probably be removed.

In your case:

```
sed -e '/^==============,$d' >part.1 <file
```

Just another UNIX hacker (since 1977, yes, 19*77*),

UUOC example

Newsgroups: comp.unix.shell
Date: 1995/04/26
From: merlyn@stonehenge.com (Randal L. Schwartz)
Subject: Useless Use of Cat Award goes to...
(was Re: bourne shell quoting, solaris, and ufsrestore)
Message-ID: <MERLYN.95Apr26071444@linda.teleport.com>#1/1

This week's useless use of cat award goes to...

>>>>> "S" == S Cowles <sco...@scheffer.Stanford.EDU> [who] writes:

S> An example of a simple command line restore job is:
S> `cat work.dump | ufsrestore xvf - './work/ '`

Which of course as most of you know by now should be written as:

```
ufsrestore xvf - './work/ ' <work.dump
```

Help stamp out **Useless Uses of Cat**!

(And csh scripts, but that's another battle. :-)

Just another Useless Use Of Usenet Bandwidth,

UUOC example

Newsgroups: comp.unix.shell
Date: 07 Oct 1994 14:45:16 GMT
From: merlyn@stonehenge.com (Randal L. Schwartz)
Subject: Re: Csh Programming Considered Harmful
Message-ID: <MERLYN.94Oct7074516@linda.teleport.com>

>>>>> "PLM" == Peter Mutsaers <p...@atcmp.nl> writes:

PLM> In the bourne shell do
PLM> `cat filename | while read line`
PLM> `do`
PLM> .
PLM> .
PLM> `done`

Aha. The winner of this week's "**useless use of cat**" award.

Hint: nearly any time you have just *one* argument to cat, you *probably* don't need the cat.

[...]

Just another would-be shell programmer (if it weren't for Perl :-),

Why UUOC?

- Randal L. Schwartz is JAPH
- Perl's view of the world

Historical background {

- ''v6 shell'' aka. osh(1): separate glob(1), if(1), goto(1)
- Shell text processing: sh + cut + tr + sed + awk + ...
- slow fork(2); length limits → Perl (1987)

}

- Perl's view of the world
- Randal L. Schwartz is JAPH

Why people do waste cats

Myth of the performance difference

Mind model: Pipelines

- Data flow: source → filter... → sink
- Cat as a generic data source
- Syntax should emphasize the semantics

Syntax comparisons

```
cat a | foo | bar | ...
foo a | bar | ...
foo <a | bar | ...
<a foo | bar | ...
cat a b c | foo | bar | ...
```

```
foo < a > b
foo <a >b
```

```
cat a | while read line ; do ... done
<a while read line ; do ... done
while read line ; do ... done <a
```

rss2email

rss2email

- RSS to email gateway
- Queries RSS feeds and delivers new articles by email
- No need for a feedreader; use your email client!

Website: `http://www.allthingsrss.com/rss2email`

→ DEMO (usage, code, storage)

Problem: Object serializing

- Human readability?
- Mixture of static and dynamic data
- Software leverage?

→ To get a feeling: Extensively modify the subscription list!

> Warning: The pickle module is not intended to be secure against erroneous or maliciously constructed data. Never unpickle data received from an untrusted or unauthenticated source.
> — Python Docs: The Python Standard Library

Internal data structure: Tied to some version?

At least, it's plain text ...

Motivation

The most straight-forward implementation:

> In the computer programming language Python, *pickle* is the standard mechanism for object serialization;
> — Wikipedia: Pickle (Python)

Provided by the programming language:

> The *pickle* module implements a fundamental, but powerful algorithm for serializing and de-serializing a Python object structure.
> — Python Docs: The Python Standard Library

No explicit format conversions

Beware

- ‘‘Box thinking’’: Just don’t care about the outside world ...
- Contrast to Unix’ toolchest approach: Write programs to work together!
- Care for the system-perspective!

MH's m_getfld

MH (nmh, mmh)

- Mail Handling tools
- Originate from the late 70s
- Heavily developed in the 80s
- Still used ... ;-)

sbr/m_getfld.c: read/parse an RFC 822 message

Comments on m_getfld

And it seems that while all roads lead to Rome, all data in nmh goes through m_getfld() at some point. And that's where the fun begins ... the function is LITERALLY cursed! :-)
— Ken Hornstein (2012)

/* This module has a long and checkered history. */

my thought is, fire photon torpedoes. m_getfld was the wrong approach when it was new but it worked well enough (especially on slower older machines).
— Paul Vixie (2012)

Reasons ...

Van Jacobson (1986):

> This routine was accounting for 60% of the cpu time used by most mh programs. I spent a bit of time tuning and it now accounts for <10% of the time used.
>
> ...

... and problems

...

Like any heavily tuned routine, it's a bit complex and you want to be sure you understand everything that it's doing before you start hacking on it. Let me try to emphasize that: every line in this atrocity depends on every other line, sometimes in subtle ways. You should understand it all, in detail, before trying to change any part. If you do change it, test the result thoroughly [...]. "Minor" bugs in this routine result in garbaged or lost mail.

If you hack on this and slow it down, I, my children and my children's children will curse you.

A closer look

i've just looked at m_getfld.c, for the first time since 1994 or so. **this was good code in the pdp11 era**.

this is a stateful iterator which does character level processing from an underlying stdio FILE object. **its caller must know internal details** of the state machine. opaqueness is nowhere attempted. **it digs into the underlying FILE object** to effect multi-character "ungetc" which is not supported by POSIX stdio, and it also **returns pointers into the underlying FILE object's buffer** to avoid character copying, **all with #ifdef's** for LINUX_STDIO which presumably works differently. it tries hard to give the compiler hints to use the vax MATCH3 instruction for substring searching. its API and its implementation make UTF-8 impossible and by the time this thing has returned it is not possible for the caller to perform any I18N processing on fields like "subject" that can have same. its indentation has several off-by-one shifts.

those of you who knew me in 1990 know that i used to write code like this; it was an art; **m_getfld.c is high art**.
— Paul Vixie (2012)

Optimiziation

1) Good performance is good

2) Bad performance needs optimization!

... really?

As everyone knows:

> Premature optimization is the root of all evil.
> — Donald Knuth

Optimization is a compromise

We're just ranking the various goals

How to optimize well

Rarely!

After having identified the real problem sources!

As a temporary crutch!

> Okay, the speed concerns mattered a lot back on a VAX; I think everyone agrees that nowadays it's not a big deal.
> — Ken Hornstein (2012)

The User

A broader view on Performance

Computer-oriented performance:

- Runtime performance
- Space performance

User-oriented performance:

- Presentation performance
- Debugging performance
- Maintenance performance
- Code reading performance
- ...

Changes

Back then:

- Computer time and space were expensive
- Users invested work to save computer work

Today:

- Computer time and space are cheap
- Computers should work to reduce **our** work

In consequence:

- Ignore computer performance optimizations
- **Focus on user-oriented performance optimization**

Summary

- Syntax should reflect the mind model (semantics)
- Don’t box think! Enable software leverage!
- View computer performance optimizations as temporary crutches!
- Care about humans, not about computers!

References

Literature

- http://en.wikipedia.org/wiki/Computer_performance

- http://catb.org/jargon/html/U/UUOC.html
- http://partmaps.org/era/unix/award.html
- http://www.in-ulm.de/~mascheck/various/uuoc/

- http://en.wikipedia.org/wiki/Thompson_shell
- http://v6shell.org/
- http://www.in-ulm.de/~mascheck/bourne/#predecessors

- http://www.allthingsrss.com/rss2email
- http://docs.python.org/2/library/pickle.html
- http://en.wikipedia.org/wiki/Pickle_(Python)

- http://marmaro.de/prog/mmh
- http://git.marmaro.de/?p=mmh;a=blob;f=docs/m_getfld.c.humor

This talk was prepared using tools of the Heirloom project:
http://heirloom.sf.net

The slides macros are based on
http://repo.cat-v.org/troff-slider/

The slides are available on my website http://marmaro.de/docs/

2013-11-11 at CCC Ulm, ChaosSeminar