



Save Time with Modern Search Techniques

Mark Jeanmougin, SANS Community Instructor
© 2018 Mark Jeanmougin | All Rights Reserved | Version 2.1.6

A trip into GNU Parallel, xargs, and other techniques to maximize the parallel capabilities of modern CPU's & disks. Examples focus on search, anti-virus, and image (photo) processing. They are generally applicable.

Author: Mark Jeanmougin / markjx@gmail.com / @markjx01
Supplemental material at <https://github.com/markjx/search2018>

Monday Pre-Coffee

Boss discovers Alexa Top 1 Million

How often do we go there?

Before I do something like this:

```
ls SG*/SG* | while read i ; do
  zgrep -f /var/opt/ldata/paraproj/alexa/top-1m $i
done > bigOutfile.log
```

Start with

```
time zgrep -f /var/opt/ldata/paraproj/alexa/top-1m \
SG_main__470802230000.log.gz > out
```

If these commands don't make sense, don't worry. We'll get there.

The Alexa Top 1 Million may no longer a free service (as of April 15, 2018). See also:

- <https://support.alexa.com/hc/en-us/articles/200449834-Does-Alexa-have-a-list-of-its-top-ranked-websites->
- <https://www.alexa.com/topsites>
- <http://s3.amazonaws.com/alexa-static/top-1m.csv.zip>
- <https://blog.majestic.com/development/alexa-top-1-million-sites-retired-heres-majestic-million/>
- <https://umbrella.cisco.com/blog/2016/12/14/cisco-umbrella-1-million/>

Monday

Get coffee; check email; still running

Check open tickets; still running

Work weekend incidents; still running

Go to lunch; **STILL RUNNING**

Update boss

Get cupcakes



That runs for 30 minutes while you get coffee and check your email.

That runs for another 30 minutes while you check for updates to your open vendor cases about their stuff not working the way they told your boss it would

It runs for another hour while you triage open tickets from the weekend.

It runs through lunch while you finally get to your day job.

You get back from lunch (That fancy Indian place that Hannah likes. Chicken Tikka Masala. It was delicious.) and **IT IS STILL RUNNING!** Update the boss to keep him off your back.

At first, it was nice to have this keep running; it kept your boss off your back. Now, any investigation of employee activity is slower because this stupid query is still running.

Update boss: let it run over night. Remind him that you're only looking at a fraction of the logs.

Time to go get cupcakes.

Tuesday Morning

- STILL RUNNING!
- >990min for 1GB of logs. I have 55GB. Doing some maths...
 - This'll take >1 month!
- Find a YouTube video called "Save Time with Modern Search Techniques"
- Find your boss's corporate card.
- Overnight shipping is a beautiful thing...

That runs all night.

The zgrep has been running all night. At this point, it has taken 990min of CPU time. Your test search is on 1GB of logs, you have 55GB of logs. Do some math...

THIS WILL TAKE OVER A MONTH!

There's gotta be a better way!

Find a YouTube video of a SANS presentation called "Save Time with Modern Search Techniques". And watch it.

Where's that corporate credit card? :)

Amazon Prime next day delivery FTW!!!

Wednesday Morning

Build the machine

Load the data, 20 minute copy

```
$ time ls nvme?/SG*/*lz4 | shuf | parallel -u -j 14 --nice 14 lz4cat {} \\  
grep -F -f /var/opt/ldata/paraproj/alexa/top-1m \| wc -l | totes1.awk  
696951104  
real    6m44.586s  
user    79m3.388s  
sys     11m41.322s
```



SANS

```
$ time ls nvme?/SG*/*lz4 | shuf | parallel -u -j 14 --nice 14 lz4cat {} \\  
grep -F -f /var/opt/ldata/paraproj/alexa/top-1m \| wc -l | totes1.awk  
696951104  
real    6m44.586s  
user    79m3.388s  
sys     11m41.322s
```

That's Not What I Meant!

Don't want to see what **is** on the Top 1million list!
What's **not** on it?

Re-Run with “grep -v”

```
$ time ls nvme?/SG*/*lz4 | shuf | parallel -u -j 110% --nice 14 lz4cat {} \\  
grep -v -F -f /var/opt/ldata/paraproj/alexa/top-1m \|| wc -l | totes1.awk  
53041938  
real    5m40.702s  
user    120m19.058s  
sys     10m6.455s
```

We'll explain the command specifics later...

What about a simple example?

Maybe you get a report from your Threat Intel team saying that a certain URL is bad. So, do we have any hits to that URL from our network?

```
$ time ls nvme?/SG*/*lz4 | shuf | parallel -u -j 110% --nice 14 lz4cat {} \||
grep tacobell.com \|| wc -l | totes1.awk
0
real    0m46.773s
user    10m2.880s
sys     5m28.194s
```

You need to provide a report of all activity for one user going back as far as you can. Or, maybe you get a report from your Threat Intel team saying that a certain URL is bad. So, do we have any hits to that URL from our network?

How big is that data set, anyway?

750 mega logs (750 million logs)

305GB of data. 55GB gzip'ed

$\frac{3}{4}$ of a Billion logs searched in $\frac{3}{4}$ of a minute.

Rate of 1 Billion logs / minute

Could your SIEM do that?

Many people really like their SIEM. Some people are going to SEC455 or SEC555 later this week and are going to learn awesome ways to build and use SIEM's.

Research Data Sets

- Hard to find so Share Your Data!
- Bluesmote.com
 - 54GB of gzip'ed logs.
 - That's 305,023,727,207 bytes or 305GB of logs comprising 751,296,241 or 751million events.
- CERT Insider Threat Tools:
 - Not as “real world”. Not as useful.

The Bluesmote data set is available at <http://bluesmote.com> . Published by <http://telecomix.org>
CERT Insider Threat Tools: <https://www.cert.org/insider-threat/tools/>

What You'll Learn

I'm here to teach techniques

I'll demo on a few data sets. Think of your data sets!

Slides at: <https://github.com/markjx/search2018/>

Ask questions!

I'm primarily here to teach you some techniques. I'll demonstrate those techniques on some data sets. Throughout this presentation, be thinking of other data sets you have where these techniques may work.

Ask questions! Although, I reserve the right to ask you to hold certain questions until the end.

Agenda

- ✓ Intro
- ☐ whoami
- ☐ Theory
- ☐ Existing Tools: xargs & GNU Parallel
- ☐ Parsing & Splitting
- ☐ At Home
- ☐ Demos
- ☐ New Tools

\$ whoami

- Mark Jeanmougin (markjx@gmail.com / @markjx01)
- Always Blue Team
- SANS Community Instructor for SEC511
- IR & DFIR
 - Inappropriate Internet Use & Academic Fraud
 - Adversary Activity
- IT for >20 years. Security since 2000.
- Useless Superpower: I can eat a Girl Scout Cookie

Mark Jeanmougin

markjx@gmail.com / @markjx01

<https://markjx.blogspot.com/>

Blue Team for my whole career.

SANS Community Instructor for SEC511

DFIR

Inappropriate Internet Use

Academic Fraud

Malware Activity

Started “Experimenting” with UNIX in college.

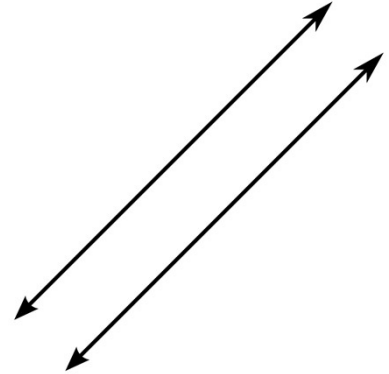
Been doing IT stuff for over 20 years now. Security since 2000.

While I do have a \$DayJob, this work is not endorsed or sponsored by them.

Surprisingly, it looks like there’s no Trademark associated with the phrase “Girl Scout Cookie”. At least, according to: <https://www.girlscouts.org/en/cookies/all-about-cookies.html> on 4/17/2018 .

Parallelism is hard

- Change your Code
- Change your Data



Change your Code

- Fine Grained Parallelism
- Cinebench
- Many Compression / Decompression tools:
 - pigz / pbzip2 / xz -T 30
- Coarse Grained Parallelism
 - Not Searching.
 - Fool our search tools by splitting our input data

Coarse Grained Parallelism

Plenty of things don't; like searching. BUT, if you have huge amounts of data, you can run the same search against multiple pieces of data in parallel. In Cyber Security, we certainly have plenty of data!

Change your Data

- You want “many” input files. >1 per CPU core
 - Not too small: >>1 sec per file
- Only have one multi-GB file? Split to the rescue!

```
$ split -a 2 -d -l 2000000 192.168.1.13-20180113.log 192.168.1.13-20180113.spl
$ ls -al 192.168.1.13-20180113.spl?? | head -3
192.168.1.13-20180113.spl00
192.168.1.13-20180113.spl01
192.168.1.13-20180113.spl02
```

- Compress, too?

split man page:

NAME

split - split a file into pieces

SYNOPSIS

split [OPTION]... [FILE [PREFIX]]

DESCRIPTION

Output pieces of FILE to PREFIXaa, PREFIXab, ...; default size is 1000 lines, and default PREFIX is 'x'.

-a, --suffix-length=N

generate suffixes of length N (default 2)

-d

use numeric suffixes starting at 0, not alphabetic

-l, --lines=NUMBER

put NUMBER lines/records per output file

-n, --number=CHUNKS

generate CHUNKS output files; see explanation below

Old Code

Do you go through logs like this?

```
$ time ls http-201* | while read i
do
  xzcat $i | grep badsite.org
done | wc -l
0
real    7m26.890s
user    8m0.930s
sys     0m14.689s
```



Done on the “CERT-insider r5.2” dataset.

New Code - xargs

Exploit your hardware's parallelism!

```
$ time ls http-201* | xargs -P 64 -L 8 xzcat | grep badsite.org | wc -c
0
real    1m59.148s
user    12m31.649s
sys     10m56.685s
```

That's almost four times as fast!!!

xargs – Breakdown!

What's that xargs command line?

xargs -P 64 -L 8 zcat

- xargs takes a list of arguments and executes a command one or more times with those arguments
- -P: Number of instances to kick off in Parallel
- -L: Number of Lines from the input file to assign to each job

Done on the cert-insider r5.2 dataset on a single NVMe drive.

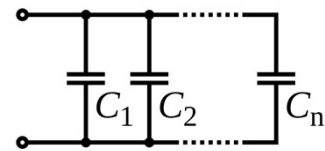
-P 64 was basically chosen at random. I wanted a number greater than my number of CPU cores (16). So I quadrupled it. My gut tells me that there's not much speed improvement going for 32-64, but #YOLO!

-L 8 gave me about 100 jobs to run. I wanted to L to be $> 2 \times P$. This seemed about right.

Had this been a scientific study or a production implementation, I'd have done some more testing here.

New Tool: GNU Parallel

- Plenty of documentation:
 - 58 page man page (`man -t parallel | ps2pdf - parallel.pdf`)
 - `man parallel_tutorial`: another 38 pages of light reading
- Available in most Linux / UNIX environments



I needed something to do multiple downloads in parallel from a video sharing site. I was going to write a shell script to do this, then found parallel.

Available in:

Fedora: `dnf install parallel`

CentOS: `yum install epel-release; yum install parallel`

Ubuntu 16.04 LTS (REMnux, SIFT, etc): `apt install parallel`

Ubuntu 17.10: `apt install parallel`

Parallel: Baseline

How long does it take? The *old* way:

```
time ls nvme[01]/SG*/*lz4 | while read i
do
  lz4cat $i | grep tacobell.com
done | wc -l
real    7m4.406s
user    6m15.146s
sys     1m46.403s
```

750 Megalogs & 305GB in 7m



Photo Credit: My cat, Ollie, just chillin' like a villin.

Parallel: Step 1

```
$ time ls nvme[01]/SG*/*lz4 | \  
parallel -u lz4cat {} \|| grep tacobell.com | wc -l  
  
real    0m49.910s  
user    9m13.037s  
sys     4m34.285s
```

7 min to <1m!



Normally, parallel “chunks” up the output so that it is put out in the same order as it is generated. For many applications, this is the desired behavior. For this run, I just want to know how many people went to tacobell.com in search of tasty tacos. The “-u” option to parallel tells it to output data as it is ready rather than in order. According to the man page, this is faster.

Photo Credit: My cat, Ceili, having recently been shaved.

Parallel: Step 2

Use all drives better & Multiple “wc -l”s

```
$ time ls nvme[01]/SG*/*lz4 | shuf | \  
parallel -u lz4cat {} \|| grep tacobell.com \|| wc -l | totesl.awk  
real    0m44.908s  
user    9m42.136s  
sys     5m7.225s
```

Cut 10%



Photo Credit: An Indy Lights car (I think?) at Mid-Ohio 2018. Taken by Mark Jeanmougin.

Parallel: Step 3

No more Regular Expressions!

```
$ time ls nvme[01]/SG*/*lz4 | shuf | \  
parallel -u lz4cat {} \|| grep -F tacobell.com \|| wc -l | totes1.awk  
real    0m42.402s  
user    9m49.881s  
sys     5m40.397s
```

Cut 5%



We'll do a more dramatic RE / no RE example later...

Photo Credit: An IndyCar at Mid-Ohio 2018. Taken by Mark Jeanmougin.

Parallel: Step 4

Run at >100%

```
$ time ls nvme[01]/SG*/*lz4 | shuf | \  
parallel -j 110% -u lz4cat {} \|| grep -F tacobell.com \|| wc -l | totes1.awk  
real    0m40.149s  
user    10m16.371s  
sys      5m42.126s
```

Cut 5%



I did some testing in 10% increments starting at 100% going to 150%. 110% seemed to be the sweet spot.

- Your mileage may vary.

Photo Credit: Ferrari Formula One car driven by Sebastian Vettel at Montreal 2018. Taken by Mark Jeanmougin.

Parallel: Command Breakdown!

- `$ time ls nvme[01]/SG*/*lz4 | shuf | parallel -j 110% -u lz4cat {} \ | grep -F tacobell.com \ | wc -l | totes1.awk`
- `shuf`: randomize the order of what's passed to it
- `parallel`
 - `-j 110%`: run 11 processes for each 10 CPU threads
 - `-u`: Output is printed as soon as possible (output from multiple jobs may be mixed)
- `lz4cat`: reads lz4 compressed data and dumps it out
- `grep -F` : Search for a string without regular expressions
- `wc -l` : return the number of lines
- `totes1.awk`: sum the first field of input (written by Mark J)

This slide is more for viewing printouts.

Decompression vs. “Real Work” I/2

Threat Intel give you a list of 2320 malicious URL's & IP's.
Do we have any hits?

```
$ time ls nvme[01]/SG*/*lz4 | parallel -u lz4cat {} \|| grep -f  
/var/opt/ldata/paraproj/malwaredomainlist/bad-urls | wc -l
```

```
real    677m29.743s  
user    15936m22.485s  
sys     45m0.255s
```

List is courtesy of: <http://www.malwaredomainlist.com/forums/index.php?topic=3270.0>

Decompression vs. “Real Work” 2/2

Using all our Parallel tricks & no Regular Expressions:

```
$ time ls nvme[01]/SG*/*lz4 | shuf | parallel -j 110% -u lz4cat {} \|| grep -F  
-f /var/opt/ldata/paraproj/malwaredomainlist/bad-urls \|| wc -l | totes1.awk  
  
real    1m36.837s  
user    29m22.391s  
sys     5m9.262s
```

Over ELEVEN HOURS -> 96 seconds!

The “-F” option to grep tells it to treat data as strings, not as regular expressions. MUCH faster.

Parsing, not just grep'ing

Here's an example of parsing & summarization rather than just searching

3m45s (or so) to get a report of the top 15 sites

```
$ time ls nvme?/SG*/*lz4 | shuf |
parallel -u -j 110% lz4cat {} \| printurl.awk \| sort \> {}.url
real    2m9.690s
user    37m8.133s
sys     7m2.533s

$ time sort --merge nvme?/SG*/*.url > nvme0/allURL
real    0m57.226s
user    0m43.627s
sys     0m13.069s

$ time uniq -c nvme0/allURL | sort -n | tail -15
*SNIP*
real    0m35.003s
user    0m33.049s
sys     0m2.102s
```

printurl.awk is available from the Github site. Written by Mark Jeanmougin.

Splitting Large Files

Split large files into chunks to maximize CPU Utilization

08:28:50 PM	CPU	%user	%nice	%system	%iowait	%steal	%idle
08:28:52 PM	all	0.06	0.09	0.16	0.02	0.00	99.67
08:28:54 PM	all	15.83	0.16	48.48	0.79	0.00	34.74
08:28:56 PM	all	36.67	0.36	57.93	1.56	0.00	3.47
08:28:58 PM	all	33.60	0.42	59.77	2.15	0.00	4.06
08:29:00 PM	all	34.97	0.30	58.12	2.36	0.00	4.26
08:29:02 PM	all	39.97	0.39	55.14	1.05	0.00	3.45
08:29:04 PM	all	40.42	0.47	53.77	1.65	0.00	3.69
08:29:06 PM	all	42.98	0.28	50.95	1.68	0.00	4.10
08:29:08 PM	all	51.32	0.44	43.46	1.31	0.00	3.47
08:29:10 PM	all	53.48	0.53	40.47	1.76	0.00	3.76
08:29:12 PM	all	56.56	0.33	37.39	1.76	0.00	3.96
08:29:14 PM	all	56.26	0.47	37.69	1.79	0.00	3.78
08:29:16 PM	all	56.44	0.57	37.01	1.59	0.00	4.39
08:29:18 PM	all	50.55	0.35	37.07	2.11	0.00	9.32
08:29:20 PM	all	41.07	0.38	33.90	2.57	0.00	22.08
08:29:22 PM	all	39.07	0.47	32.47	2.71	0.00	25.28
08:29:24 PM	all	39.90	0.28	32.76	2.27	0.00	24.79
08:29:26 PM	all	36.63	0.36	29.58	3.11	0.00	30.31
08:29:28 PM	all	34.94	0.41	28.12	3.43	0.00	33.11
08:29:30 PM	all	31.06	0.24	23.25	3.39	0.00	42.07
08:29:32 PM	all	26.16	0.31	17.05	3.42	0.00	53.05
08:29:34 PM	all	18.62	0.36	10.28	3.45	0.00	67.29
08:29:34 PM	CPU	%user	%nice	%system	%iowait	%steal	%idle
08:29:36 PM	all	13.94	0.17	7.52	1.47	0.00	76.89
08:29:38 PM	all	5.13	0.14	2.39	0.25	0.00	92.09
08:29:40 PM	all	0.06	0.08	0.17	0.00	0.00	99.69

Output of “sar 2” command.

See how CPU utilization trails off over time? That’s bad. Split your largest files into chunks so the work is more balanced.

How to do this at \$home?

Get the data

Store the data

Process the data

Get the Data

Syslog

FTP / SCP daily exports

Store the Data

I like having one log file per generator per day. For example:

2018/02/06/firewall1.log

2018/02/06/firewall2.log

2018/02/06/proxy1.log

2018/02/06/proxy2.log

Process the Data

Process the Data

- What do you need?
 - Multi-core CPU (Threadripper FTW!) (64 lanes of PCIe)
 - SSD's (NVMe FTW!)
 - ASCII Logs
- How to get the hardware?
 - Xeon workstation from HP, Dell, Lenovo, etc
 - Build your own Threadripper box. (Gamer on helpdesk?)
 - My build: <https://pcpartpicker.com/list/3d2TCb>

You'll also need a Linux environment. From what I know about PowerShell, it isn't powerful enough for this ... yet.

Organizational Acceptance

- How to justify the cost?
 - Price of a cup of coffee / day over 3y
- Hardware & Software “Support”?
 - Your IT, desktop, etc support teams with react in 1 of 2 ways:
Hatred or Joy
 - Do you have other Linux workstations?
 - “Server”?

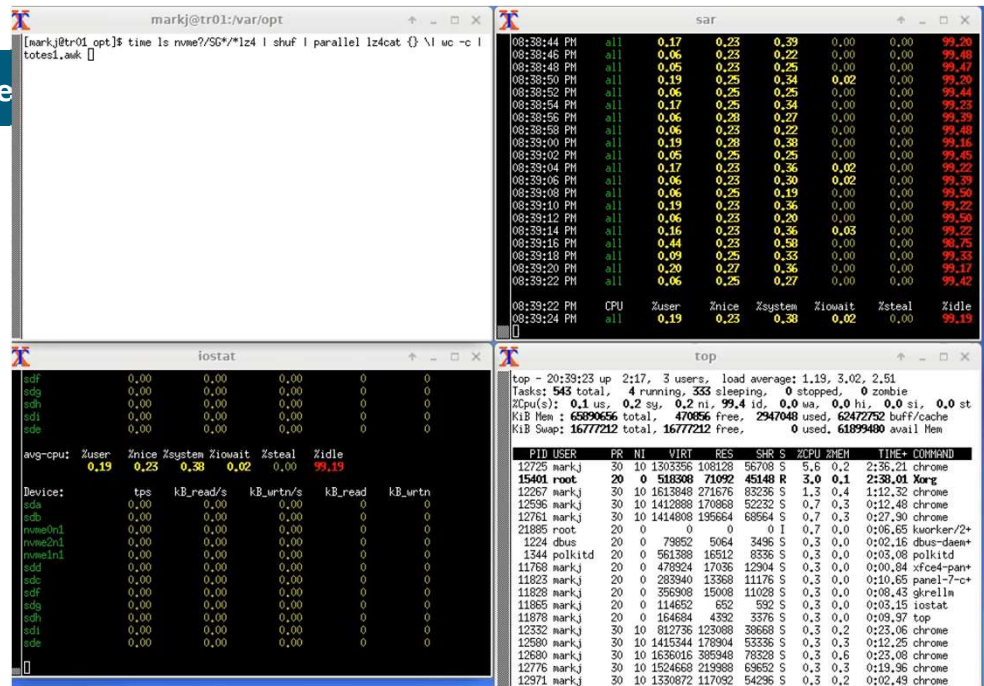


Demos!

1. CPU intensive part is decompression
2. CPU intensive part is searching

Demo: Decompre

Decompression
is CPU
intensive



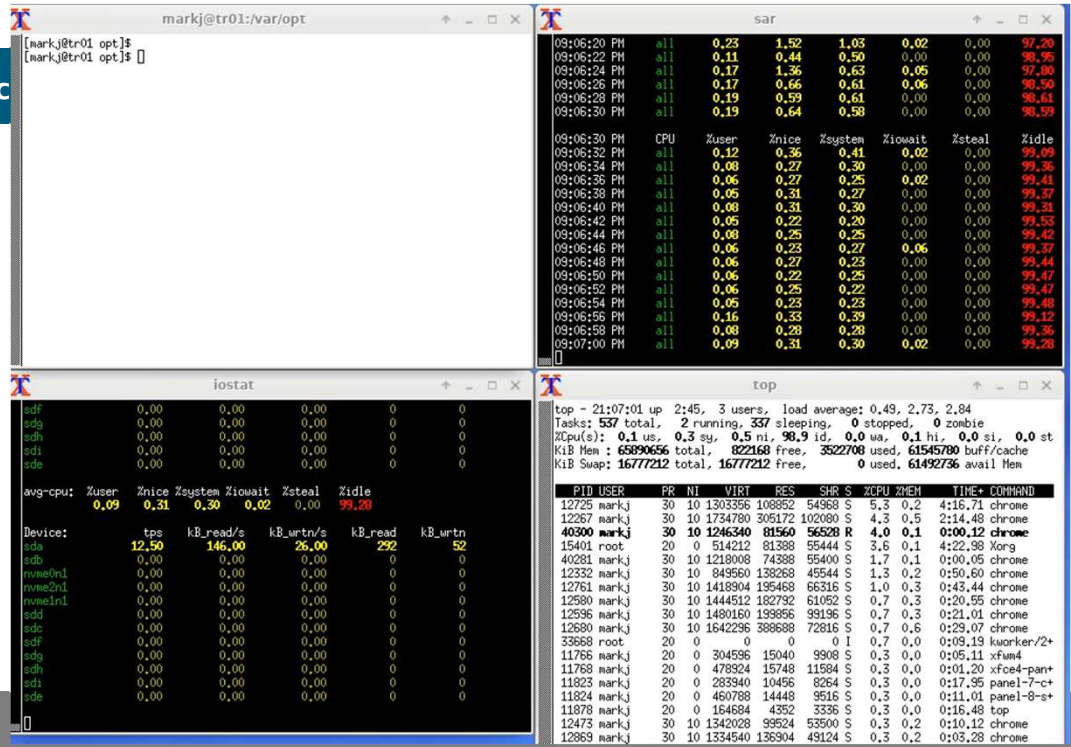
Command:

```
time ls nvme?/SG*/*lz4 | shuf | parallel lz4cat {} \ | wc -c | totes1.awk
```

I recorded this on my Fedora workstation with “recordmydesktop -x 2570 -y 1 --width 1000 --height 700”. The output is ogv, which PowerPoint doesn’t like. I converted to mp4 with “ffmpeg -i demo.ogv -f mp4 demo.mp4”. You can also use the “--windowid” option to only record a single window. You find out the windowid with the “xwininfo” command

Demo: Search

Searching
is CPU
intensive



Command:

squishycat



cat compressed files

- **gzip**
- **bzip2**
- **lz4**
- **xz**

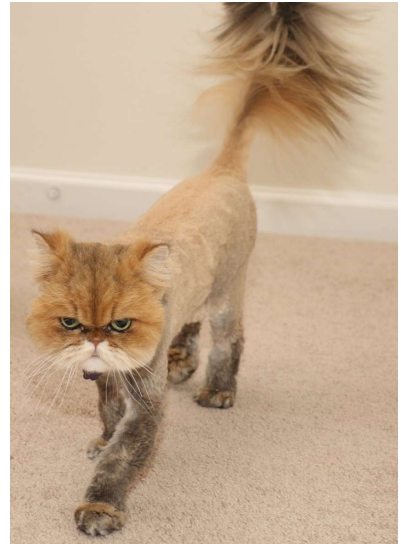


Photo Credit: My cat, Ceili, just before and after being shaved. Taken by Mark Jeanmougin.

<https://github.com/markjx/search2018/>

squishycat is like the normal UNIX cat command except: When dealing with normal ASCII text, it just cats it. When dealing with data compressed, it decompresses it first, then cat's it. It currently supports gzip, bzip2, lz4, and xz.

squishycat: Use

```
[markj@tr01 20180415]$ ls S* | while read fn ; do file $fn ; squishycat $fn |
sha1sum ; echo . ; done
SG_main__470802230000.log: Non-ISO extended-ASCII text, with very long lines
6ddafe913bf160a0a6c2c4a949de5270e19682e9  -
.
SG_main__470802230000.log.bzip2: bzip2 compressed data, block size = 900k
6ddafe913bf160a0a6c2c4a949de5270e19682e9  -
.
SG_main__470802230000.log.gzip: gzip compressed data
6ddafe913bf160a0a6c2c4a949de5270e19682e9  -
.
SG_main__470802230000.log.lz4: LZ4 compressed data (v1.4+)
6ddafe913bf160a0a6c2c4a949de5270e19682e9  -
.
SG_main__470802230000.log.xz: XZ compressed data
6ddafe913bf160a0a6c2c4a949de5270e19682e9  -
.
[markj@tr01 20180415]$ █
```

Generated compressed files:

```
ifn=SG_main__470802230000.log ;
for i in gzip bzip2 xz lz4
do
  ofn=${i}.out ;
  (time cat $ifn | $i > ${ifn}.$i) > $ofn 2>&1 &
done
```

grepwide

Rounds up all the search techniques discussed in this paper

Files in your home directory:

- look4me: What you're searching for
 - No blank lines!
- outfile: Saves output here

```
[markj@tr01 lz4links]$ cat ~/look4me
yum.com
kfc.com
kfc.co.uk
pizzahut.com
tacobell.com
wingstreet
[markj@tr01 lz4links]$ time grepwide

real    0m49.802s
user    13m32.079s
sys     4m7.599s
[markj@tr01 lz4links]$ wc -l ~/outfile
1982 /home/markj/outfile
[markj@tr01 lz4links]$ █
```

<https://github.com/markjx/search2018/>

grepwide implements the parallelization techniques in this presentation. It uses two files in your home directory:

- look4me: list of regular expressions, one per line, that you're looking for. NO BLANK LINES!
- outfile: whatever lines match the RE's in look4me are saved in this file

Questions?

markjx@gmail.com

@markjx01 <https://markjx.blogspot.com/>

Slide Deck & Scripts:

<https://github.com/markjx/search2018/>

Hardware Stopped Getting Faster

MHz **stopped** increasing in 2000. Core Count **started** increasing in 2006.

For my work: AMD ThredRipper & Samsung NVMe

This works with

- Any multi-core CPU
- Any SSD

MHz stopped increasing in 2000. Core count started increasing in 2006.

I'm Using: CPU: AMD 1950X "ThreadRipper" & Samsung NVMe drives

But really, these techniques work with: Any multi-core CPU & Any SSD

Stop thinking that VM's are just as good as bare hardware. Stop thinking that you need "server class" hardware.

The Pentium 4 (2000) was the last CPU where Intel tried to chase MHz. It was replaced by the Core architecture (2006), itself highly based on the P6 architecture of the Pentium Pro (1995). That was an excellent architecture, but as of 2018, the only thing people will remember about is that it was Intel's first CPU with Speculative Execution.

Fastest MHz Offered:

Pentium 4 HT 3.8F: 3.80GHz / Nov 2004

Ryzen 7 1800X: 4.1GHz / Mar 2017

Intel i9-7940X: 4.3GHz / Q3 2017

Threadripper 1950X: 4.2GHz / Aug 2017

What is Hyper-Threading? Or Simultaneous Multi-Threading?

- One execution core with multiple register sets
- Two queues, two registers, one cashier.
- When someone goes "uh...", the cashier pays attention the person in the other queue.

Operating System

- Many Options!!!
 - Linux VM's or bare hardware
 - Windows Subsystem for Linux
 - Docker
- Test Yo'self!
- What's important?
 - Your skills / Institutional Support
 - Cost / Performance

7200rpm RAID vs. NVMe

7200rpm RAID

```
time ls SG*/*lz4 | shuf | parallel -u -j 110% --nice 14 lz4cat {} \\  
grep tacobell.com \\  
wc -l | totes1.awk  
real    11m5.992s  
user    6m35.496s  
sys     2m1.593s
```

NVMe

```
time ls nvme[01]/SG*/*lz4 | shuf | parallel -u -j 110% --nice 14 lz4cat {} \\  
grep tacobell.com \\  
wc -l | totes1.awk  
real    0m44.252s  
user    9m39.629s  
sys     5m14.836s
```

The RAID I used is five 7200rpm 2TB drives in RAID 5. This is meant to be representative of an Enterprise configuration.

Compressed or Uncompressed?

Types of Compression

Compression vs. Decompression

What does your “off hours” usage look like?

Know your Data

Don't be afraid to “transcompress”

Test, Test, Test

Types of Compression

Performance results

Compression vs. Decompression

What does your “off hours” usage look like?

Conclusion: “It Depends”

Know your Data

Don't be afraid to “transcompress”

Test, Test, Test

Most important thing to take away from this section: Small compression differences have HUGE impacts. Test for your environment. Different data sets may want different compression schemes.

Compression Test – CERT Insider r6.2

		Space		wc -l		Time	grep -F -f		Time
		MB	Savings	real	user+sys	Savings	real	user+sys	Savings
raid5	uncompressed	86054	0.00%	563.815	71.4251	0.00%	563.154	217.85	0.00%
nvme	uncompressed	86054	0.00%	43.672	28.362	92.25%	133.874	112.378	76.23%
nvme	split	86054	0.00%	67.590	41.942	88.01%	70.403	250.432	87.50%
nvme	gzip	35375	58.89%	29.763	881.843	94.72%	37.356	1087.088	93.37%
nvme	bz2	19507	77.33%	353.441	10994.801	37.31%	425.696	12579.695	24.41%
nvme	lz4	53965	37.29%	44.730	411.786	92.07%	46.242	316.816	91.79%
nvme	xz	4519	94.75%	21.332	637.354	96.22%	27.974	853.265	95.03%

wc -l, grep -F -f (2320 lines)

Compression – Winner!

- Winner: xz
 - fast decompression & very little space on disk
 - Compared to uncompressed: 95% space & speed
 - Compared to gzip: 77% space & 27% speed
- Your Mileage May Vary
 - Other data sets work better with other algorithms



Transcompression

I don't know if that's a word, but I'm using it.

It is trivial to convert from one compression type to another.

Something like this:

```
$ time ls nvme?/S*/*lz4 | shuf | parallel -u lz4cat {} \; | gzip \> {}.gz
real    6m39.899s
user    126m0.536s
sys     3m58.523s
```

That's 6 and a half minutes to move 305GB of data from lz4 to gz

Scripting

Round this out with some ideas on how to script some of this to do... ? something useful to someone.

Daily “top 10” reports from some of the bro logs from pcaps?

parallel & Pictures

Resize 5,558 jpg's from 20MP -> 2.6MP

```
[markj@tr01 all]$ time make-picasa.sh ./
```

```
real    36m1.123s
user    226m15.221s
sys     142m12.921s
```

And... in parallel

```
$ mv ../picasa ../picasa.serial ; mkdir ../picasa ; time ls | \
parallel make-picasa1
real    9m50.470s
user    287m49.904s
sys     20m52.173s
```

The dataset is 5558 jpg files from my vacation to Montreal in Summer 2017 which total to about 30GB of data.

The script converts the ~20MP files from my Canon 7D Mark II to ~2.6MP files with higher compression rates suitable for sharing on social media.

```
[markj@tr01 all]$ time make-picasa.sh ./
```

```
real    36m1.123s
user    226m15.221s
sys     142m12.921s
```

And... in parallel

```
[markj@tr01 all]$ mv ../picasa ../picasa.serial ; mkdir ../picasa ; time ls | parallel make-picasa1
```

```
real    9m50.470s
user    287m49.904s
sys     20m52.173s
```


parallel & ClamAV

Scan 80,168 files, taking 39,292MB of disk space

```
real    177m58.320s
user    174m36.915s
sys     1m45.777s
```

And... in parallel

```
$ time ls -S | shuf | xargs -L 600 -P 32 clamscan > parallel
real    13m52.956s
user    397m23.623s
sys     4m29.457s
```

Approximately 80,168 files taking up 39,292MB of disk space. Files came from https://archive.org/download/virusshare_malware_collection_000 They are basically all malicious.

Going through sequentially:

```
[markj@tr01 virusshare]$ time clamscan -l serial -r .
```

```
----- SCAN SUMMARY -----
Known viruses: 6470742
Engine version: 0.99.4
Scanned directories: 20
Scanned files: 80148
Infected files: 46706
Data scanned: 59250.00 MB
Data read: 39007.12 MB (ratio 1.52:1)
Time: 10678.307 sec (177 m 58 s)
```

```
real    177m58.320s
user    174m36.915s
sys     1m45.777s
```

And, in parallel...

```
$ time ls -S | shuf | xargs -L 600 -P 32 clamscan > parallel
real    13m52.956s
user    397m23.623s
sys     4m29.457s
```

parallel & ClamAV 2

An early run before I optimized the CPU usage

Another example of the importance of balancing CPU usage

See notes below

```
$ time find . -type f | xargs -L 400 -P 32 clamscan | tee parallel
real    28m17.375s
user    370m27.567s
sys     4m56.347s
```

the job finished at about 21:21. Here's what sar recorded in that time:

08:54:27 PM	all	47.96	0.00	1.58	0.05	0.00	50.42
08:56:17 PM	all	97.37	0.00	2.48	0.04	0.00	0.12
08:58:27 PM	all	98.10	0.00	1.84	0.02	0.00	0.05
09:00:25 PM	all	97.81	0.00	2.10	0.02	0.00	0.06
09:02:12 PM	all	97.25	0.00	2.61	0.03	0.00	0.10
09:04:17 PM	all	81.68	0.00	1.77	0.04	0.00	16.51
09:06:27 PM	all	13.06	0.01	0.31	0.05	0.00	86.57
09:08:17 PM	all	9.19	0.00	0.15	0.01	0.00	90.65
09:10:27 PM	all	6.16	0.00	0.10	0.01	0.00	93.73
09:12:27 PM	all	6.16	0.01	0.11	0.02	0.00	93.70
09:14:17 PM	all	6.17	0.00	0.10	0.04	0.00	93.68
09:16:27 PM	all	6.17	0.00	0.11	0.01	0.00	93.70
09:18:27 PM	all	6.17	0.00	0.10	0.00	0.00	93.72
09:20:17 PM	all	6.18	0.00	0.12	0.01	0.00	93.69
09:22:09 PM	all	3.59	0.00	0.53	0.07	0.00	95.81
Average:	all	3.43	0.13	1.00	0.05	0.00	95.40

The box worked hard for about 10 minutes. Then was only running a few threads for 12 minutes.

Bibliography

See notes for some of the sites that I found useful in this research, in no particular order

Some of the sites that proved useful in this research, in no particular order:

<http://www.secrepo.com/>

<https://www.netresec.com/?page=PcapFiles>

<https://virusshare.com/about.4n6>

<https://archive.org/details/datasets>

<http://www.unb.ca/cic/datasets/index.html>

<https://www.unsw.adfa.edu.au/australian-centre-for-cyber-security/cybersecurity/ADFA-NB15-Datasets/>

<https://cloudstor.aarnet.edu.au/plus/index.php/s/2DhnLGDdEECo4ys?path=%2FUNSW-NB15%20-%20pcap%20files>

<http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html>

<http://commoncrawl.org/the-data/get-started/>

<https://registry.opendata.aws/>

<https://mcfp.felk.cvut.cz/publicDatasets/CTU-Malware-Capture-Botnet-318-1/>

<https://ictf.cs.ucsb.edu/pages/the-2016-2017-ictf.html>

<https://download.netresec.com/pcap/>

ftp://download.iwlab.foi.se/dataset/smia2012/network_traffic/pcap/

ftp://download.iwlab.foi.se/dataset/smia2011/Network_traffic/

ftp://download.iwlab.foi.se/dataset/smia2012/network_traffic/pcap/
<ftp://ftp.bro-ids.org/enterprise-traces/hdr-traces05/>
<http://cybercrime-tracker.net/>
<http://cybercrime-tracker.net/all.php>
<http://dfir.to/DFIRCON-Challenge-15>
<http://dfir.to/FOR572-Challenge-Data>
<http://downloads.digitalcorpora.org/corpora/files/govdocs1/zipfiles/>
<http://log-sharing.dreamhosters.com/>
<http://osint.bambenekconsulting.com/feeds/dga-feed.txt>
https://archive.org/download/2011_04_02_enron_email_dataset
<https://download.netresec.com/pcap/maccdc-2012/>
<https://download.netresec.com/pcap/smia-2011/>
<https://download.netresec.com/pcap/smia-2012/>
https://drive.google.com/file/d/0B_IN6RzP69b2TkNrYVdOMnQ4LVE/view
<https://ictf.cs.ucsb.edu/pages/the-2016-2017-ictf.html>
<https://ransomwaretracker.abuse.ch/feeds/csv/>
<https://www.ll.mit.edu/ideval/data/1999data.html>
<https://www.ll.mit.edu/ideval/data/1999/training/week1/index.html>
<https://www.uvic.ca/engineering/ece/isot/datasets/index.php#section0-0>
<https://zeustracker.abuse.ch/blocklist.php>
<https://zeustracker.abuse.ch/blocklist.php?download=baddomains>
<https://zeustracker.abuse.ch/blocklist.php?download=badips>
<http://www.gwern.net/DNM-archives>
<http://www.malwaredomainlist.com/forums/index.php?topic=3270.0>
<http://www.netresec.com/?page=PcapFiles>