



Save Time with Modern Search Techniques

Mark Jeanmougin, SANS Instructor

© 2022 Mark Jeanmougin | All Rights Reserved | Version 5.0.0.0

Disclaimer

The information presented here is not intended for use by anyone. If you try to follow along at home and get a hangnail, instantiate global thermonuclear war, or have other adverse side effects: You're on your own. Mark, as well as his past, current, or future employers, family members, and pets disclaim any and all responsibility from now until the end of the Universe.

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
```



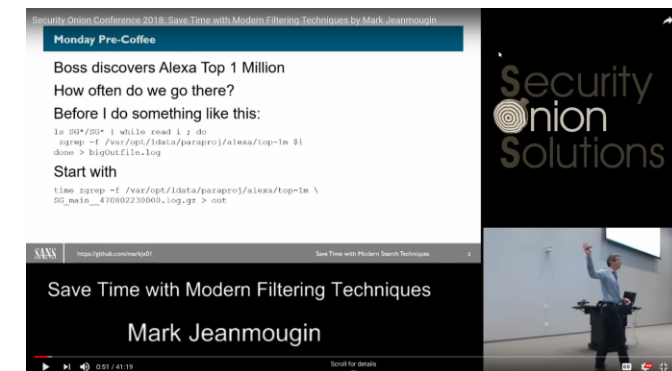
Monday

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



Tuesday Morning

- STILL RUNNING!
- >990min for 1GB of logs. I have 55GB. Doing some maths...
 - This'll take >1 month! (Actually... 3y10m11d18h34m38s)
- Find a YouTube video called “Save Time with Modern Search Techniques”
- Find your boss’s corporate card.
- Overnight shipping is a beautiful thing...



Wednesday Morning

Build the machine

Load the data, 20 minute copy



Wednesday Morning 2

Using what we learn in this presentation...

```
$ time ls SG*/*lz4 | shuf | parallel --nice 14 lz4cat {} \ | grep -a -F -f  
/var/opt/arraytest/alexa/top-1m \ | wc -l | totes1.awk  
751296241
```

```
real    0m43.617s  
user    10m13.816s  
sys     10m47.671s
```



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 SG*/*lz4 | shuf | parallel --nice 14 lz4cat {} \
| grep -v -a -F -f /var/opt/arraytest/alexa/top-1m \
| wc -l | totes1.awk
0
```

```
real    0m35.313s
user    8m13.341s
sys     8m30.060s
```


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 SG*/*lz4 | shuf | parallel --nice 14 lz4cat {} \ | grep -F  
tacobell.com
```

```
real    0m28.711s  
user    5m5.227s  
sys     7m41.065s
```



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 $\frac{1}{2}$ of a minute.

Rate of 1.5 Billion logs / minute

Could your SIEM do that?



What You'll Learn

I'm hear 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!

Agenda

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

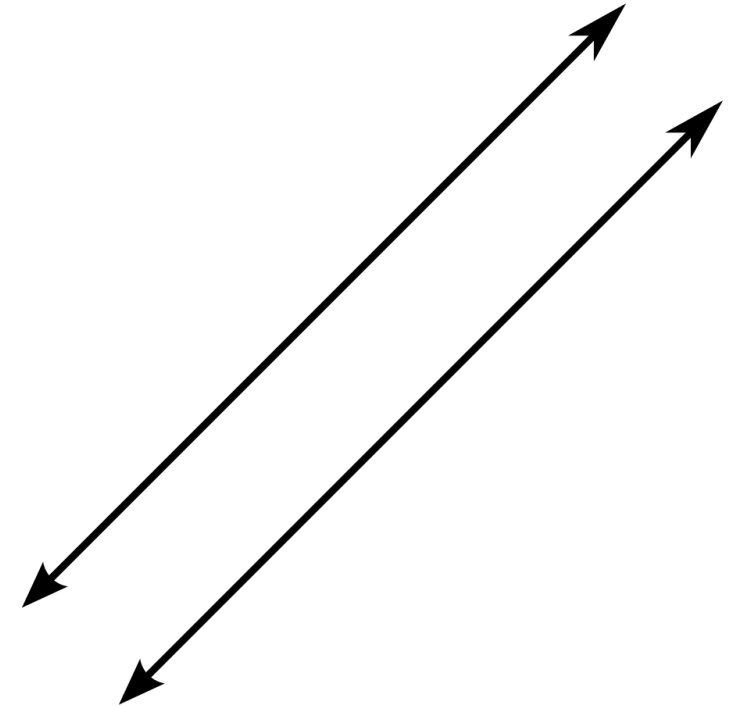
Ask
Questions!

\$ whoami

- Mark Jeanmougin (markjx@gmail.com / @markjx01)
- Career in Blue Team
- SANS Instructor
- Digital Forensics & Incident Response
 - Inappropriate Internet Use & Academic Fraud
- IT for >20 years. Security since 2000.
- Useless Superpowers
 - I can fold a fitted sheet & eat a single Girl Scout cookie

Parallelism is hard

- Change your Code
- Change your Data



Change your Code

- Fine Grained Parallelism
- Cinebench / Blender / Handbrake
- Some Compression / Decompression tools:
 - 7zip / pigz / pbzip2 / xz -T 30
- Coarse Grained Parallelism
 - Not Searching.
 - Fool our search tools by splitting our input 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?

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
```



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
```



Explained on next slide

That's almost four times as fast!!!

xargs – Breakdown!

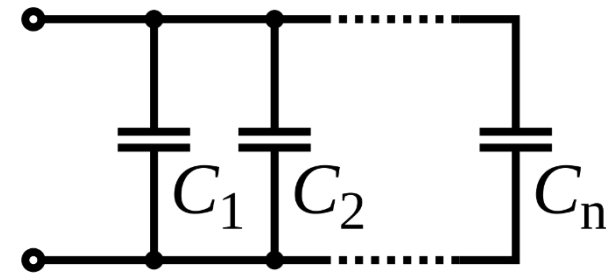
What's that xargs command line?

```
xargs -P 64 -L 8 xzcat
```

- 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

New Tool: GNU Parallel

- Plenty of documentation:
 - 63 page man page (`man -t parallel | ps2pdf - parallel.pdf`)
 - `man parallel_tutorial`: another 44 pages of light reading
 - Total of 193 pages across 11 docs
- Available in most Linux / UNIX environments



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



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!



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 | totes1.awk  
real      0m44.908s  
user      9m42.136s  
sys       5m7.225s
```

Cut 10%

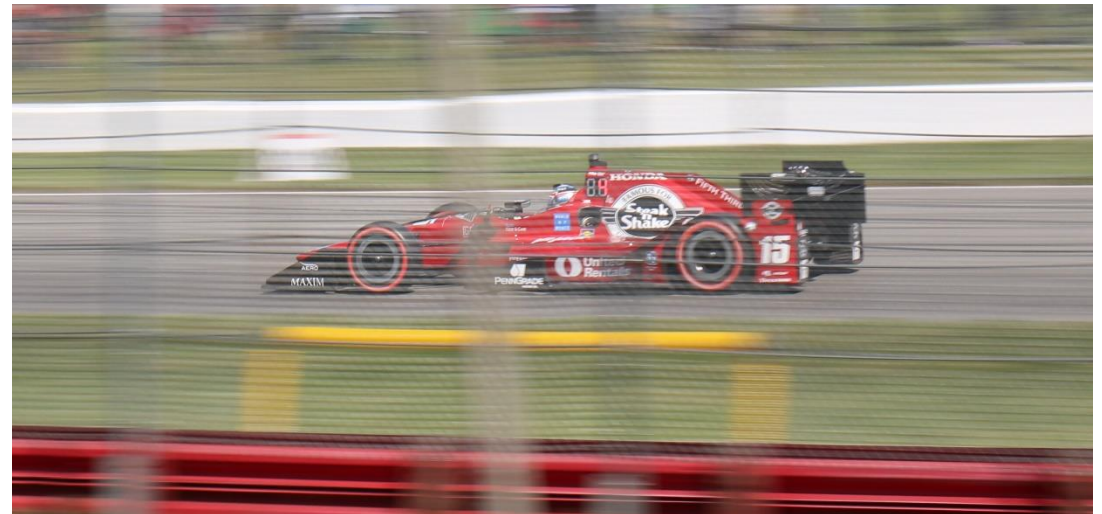


Parallel: Step 3

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%



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%



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)

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
```

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!

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
```


When to Split 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	9.92
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

Two Quick Examples

1. Zeek / Bro
2. ClamAV



What Started All This?

bro!

Threat Intel and “bad” Domain Names



3: Serial Small File Check

78735 bro log files; 231,244MB compressed to 15,379MB

- 637,084,430 log events

Cross Reference with Alexa Top 1million, small files, serial

```
$ time find 2* -type f | grep : | xargs zgrep -c -v -F -f \  
/var/opt/data0/paraproj/alexa/top-1m | totes1.awk
```

```
real    3576m25.396s  
user    3286m22.736s  
sys     295m51.797s  
$
```

59.6h

3: Serial Large File Check

4262 bro log files; 232,062MB compressed to 16,298MB

- 637,084,430 log events

Cross Reference with Alexa Top 1million, large files, serial

```
$ time find 2* -type f | grep -v : | xargs zgrep -c -v -F -f  
/var/opt/data0/paraproj/alexa/top-1m | totes1.awk
```

```
real    403m17.795s  
user    380m6.731s  
sys     30m25.214s  
$
```

6.7h

3: Parallel Small File Check

78735 bro log files; 231,244MB compressed to 15,379MB

- 637,084,430 log events

Cross Reference with Alexa Top 1million, small files, serial

```
$ time find 2* -type f | grep : | shuf | xargs -P 32 -L 100 \  
zgrep -c -v -F -f /var/opt/data0/paraproj/alexa/top-1m | totes1.awk
```

```
real    157m54.528s  
user    3688m42.650s  
sys     646m56.955s  
$
```

2.6h

3: Parallel Large File Check

4262 bro log files; 232,062MB compressed to 16,298MB

- 637,084,430 log events

Cross Reference with Alexa Top 1million, large files, serial

```
$ time find 2* -type f | grep -v : | shuf | xargs -P 32 -L 100 \  
zgrep -c -v -F -f /var/opt/data0/paraproj/alexa/top-1m | totes1.awk
```

```
real      35m40.662s  
user      576m5.104s  
sys       76m35.347s  
$
```

100:1
Speedup

Whatify?

dailyify: The process of converting your bro log files into daily batches rather than tiny hourly files to make searching faster

```
$ cat dailyify.sh
#!/bin/bash

time ls | cut -f 1 -d. | sort -u | while read i
do
    echo $i
    ls ${i}* | xargs zcat | pigz > daily.gz
    mv daily.gz ${i}.gz
done

$
```

bro Extracted Files & ClamAV - Serial

1922 exe files extracted by bro. 11,635MB

```
$ time clamscan -i --log=logfile ./extracted/
```

```
----- SCAN SUMMARY -----
```

```
Known viruses: 6673868
```

```
Engine version: 0.100.1
```

```
Scanned directories: 1
```

```
Scanned files: 1921
```

```
Infected files: 53
```

```
Data scanned: 9044.61 MB
```

```
Data read: 11627.12 MB (ratio 0.78:1)
```

```
Time: 2083.731 sec (34 m 43 s)
```

```
real      34m43.741s
```

```
user      34m16.451s
```

```
sys       0m17.710s
```

```
$
```

bro Extracted Files & ClamAV - Parallel

1922 exe files extracted by bro. 11,635MB

```
$ find extracted -type f > list
$ split -n 1/48 -a 2 -d list list.
$ time ls list.* | parallel clamscan -f {} -i --log={}.log
real      3m47.688s
user      81m27.013s
sys       1m22.020s
$
```

10:1

ClamAV Breakdown

```
$ find extracted -type f > list
```

Create list of files to be examined

```
$ split -n 1/48 -a 2 -d list list.
```

Split into 48 chunks at line breaks (ell over 48). 2 character decimal suffix.

```
$ time ls list.* | parallel clamscan -f {} -i --log={}.log
```

Kick off clamscan to examine each chunk of files.

How to do this at \$home?

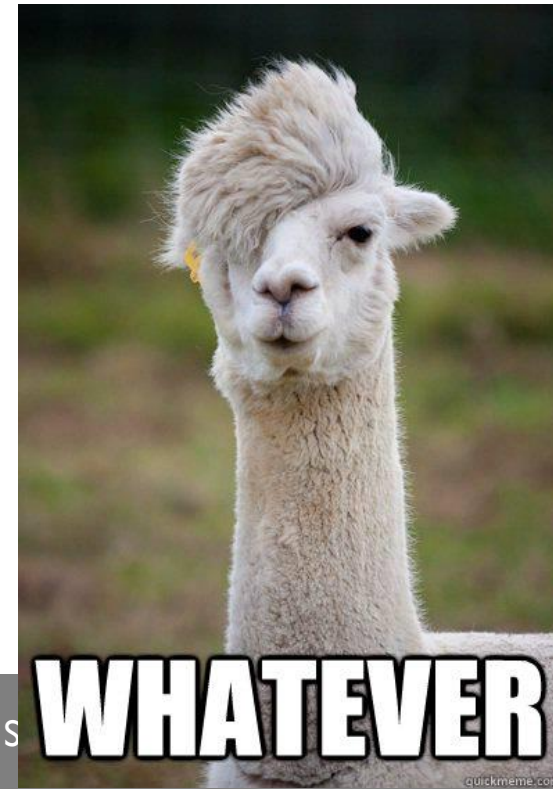
Get the data

Store the data

Process the data

Process the Data

- What do you need?
 - Multi-core CPU (Threadripper|EPYC). SSD's (NVMe FTW!)
 - ASCII Logs (jq, XML, syslog, whatever)
- How to get the hardware?
 - Xeon workstation from HP, Dell, Lenovo, etc
Threadripper Pro Workstation from Lenovo, etc.
 - Build your own Threadripper box. (Gamer on helpdesk?)
 - My build: <https://pcpartpicker.com/list/NLQGBc>



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 will react in 1 of 2 ways:
Hatred or Joy
 - Do you have other Linux workstations?
 - “Server”?
 - Security “Appliance”?
 - (Cloud? ☹️)



Demos!

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

Demo

Decompression is CPU intensive

```
markj@tr03:/var/opt/arraytest/bluesmote/20220105/spli ^ _ □ ×
[markj@tr03 xz]$ time ls "xz" | shuf | parallel xzcat {} \; | wc -l | tottest1.awk
```

```
top
top - 17:31:11 up 13 min, 20 users, load average: 1.14, 3.23, 2.51
Tasks: 849 total, 1 running, 848 sleeping, 0 stopped, 0 zombie
%Cpu(s): 0.1 us, 0.1 sy, 0.0 ni, 99.6 id, 0.2 wa, 0.0 hi, 0.0 si, 0.0 st
MiB Mem : 128739.8 total, 102412.2 free, 3485.7 used, 22841.8 buff/cache
MiB Swap: 75728.0 total, 75728.0 free, 0.0 used, 124020.1 avail Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
5451	netdata	20	0	78480	11520	4240	S	3.0	0.0	0:22.97	apps.pl+
5757	root	20	0	24.2g	76120	39772	S	2.3	0.1	0:14.36	Xorg
5123	netdata	20	0	430632	105264	14404	S	2.0	0.1	0:14.63	netdata
9031	markj	20	0	484416	26852	21352	S	2.0	0.0	0:08.89	gkrellm
16145	markj	20	0	1410544	99932	75144	S	2.0	0.1	0:01.39	simplest+
8988	markj	20	0	711216	66588	47756	S	0.7	0.1	0:06.39	xfwm4
9200	markj	20	0	236792	5972	4324	S	0.7	0.0	0:02.89	top
563	root	20	0	0	0	0	I	0.3	0.0	0:00.22	kworker+
4915	root	20	0	278952	39016	37600	S	0.3	0.0	0:00.24	sssd_nss
5452	netdata	20	0	30716	9544	8136	S	0.3	0.0	0:02.37	cups.pl+
9033	markj	20	0	412820	19804	17172	S	0.3	0.0	0:00.30	xfce4-p+
9163	markj	20	0	236792	5928	4284	S	0.3	0.0	0:02.79	top
10712	markj	20	0	236780	5976	4328	R	0.3	0.0	0:02.29	top
1	root	20	0	176968	17308	10140	S	0.0	0.0	0:02.17	systemd
2	root	20	0	0	0	0	S	0.0	0.0	0:00.01	kthreadd
3	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	rcu_gp
4	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	rcu_par+

```
iostat
sdc 0.00 0.00 0.00 0.00 0 0 0
```

avg-cpu:	%user	%nice	%system	%iowait	%steal	%idle
	0.21	0.00	0.26	0.21	0.00	99.32

Device	tps	kB_read/s	kB_wrtn/s	kB_read/s	kB_wrtn/s	kB_read	kB_wrtn	kB_read
nvme0n1	0.00	0.00	0.00	0.00	0.00	0	0	0
nvme10n1	0.00	0.00	0.00	0.00	0.00	0	0	0
nvme11n1	0.00	0.00	0.00	0.00	0.00	0	0	0
nvme1n1	0.00	0.00	0.00	0.00	0.00	0	0	0
nvme2n1	0.00	0.00	0.00	0.00	0.00	0	0	0
nvme3n1	0.00	0.00	0.00	0.00	0.00	0	0	0
nvme4n1	0.00	0.00	0.00	0.00	0.00	0	0	0
nvme5n1	0.00	0.00	0.00	0.00	0.00	0	0	0
nvme6n1	0.00	0.00	0.00	0.00	0.00	0	0	0
nvme7n1	5.00	4.00	36.00	0.00	0.00	8	72	0
nvme8n1	0.00	0.00	0.00	0.00	0.00	0	0	0
nvme9n1	0.00	0.00	0.00	0.00	0.00	0	0	0
sda	0.00	0.00	0.00	0.00	0.00	0	0	0
sdb	0.00	0.00	0.00	0.00	0.00	0	0	0
sdc	0.00	0.00	0.00	0.00	0.00	0	0	0

```
sar
05:30:32 PM all 0.07 0.00 0.12 0.20 0.00 99.61
05:30:34 PM all 0.04 0.00 0.13 0.18 0.00 99.66
05:30:36 PM all 0.08 0.00 0.16 0.18 0.00 99.58
05:30:38 PM all 0.05 0.00 0.15 0.20 0.00 99.60
05:30:40 PM all 0.09 0.00 0.15 0.17 0.00 99.59
05:30:42 PM all 0.08 0.00 0.18 0.17 0.00 99.57
05:30:44 PM all 0.08 0.00 0.15 0.18 0.00 99.59
05:30:46 PM all 0.09 0.00 0.20 0.22 0.00 99.50
05:30:48 PM all 0.08 0.00 0.13 0.22 0.00 99.57
05:30:50 PM all 0.06 0.00 0.13 0.21 0.00 99.60
05:30:52 PM all 0.09 0.00 0.16 0.17 0.00 99.58
```

	CPU	%user	%nice	%system	%iowait	%steal	%idle
05:30:52 PM	CPU	%user	%nice	%system	%iowait	%steal	%idle
05:30:54 PM	all	0.06	0.00	0.16	0.20	0.00	99.58
05:30:56 PM	all	0.05	0.00	0.13	0.17	0.00	99.64
05:30:58 PM	all	0.07	0.00	0.22	0.21	0.00	99.50
05:31:00 PM	all	0.05	0.00	0.13	0.16	0.00	99.66
05:31:02 PM	all	0.07	0.00	0.12	0.17	0.00	99.64
05:31:04 PM	all	0.07	0.00	0.17	0.17	0.00	99.59
05:31:06 PM	all	0.06	0.00	0.16	0.17	0.00	99.60
05:31:08 PM	all	0.09	0.00	0.18	0.20	0.00	99.53
05:31:10 PM	all	0.05	0.00	0.18	0.18	0.00	99.59
05:31:12 PM	all	0.20	0.00	0.24	0.21	0.00	99.35

Demo

Searching
is CPU
intensive

```
markj@tr03:/var/opt/arraytest/bluesmote/20220105/splii ^ _ □ ×
[markj@tr03 xz]$ wc -l /var/opt/arraytest/bluesmote/20220105/bad-urls
2320 /var/opt/arraytest/bluesmote/20220105/bad-urls
[markj@tr03 xz]$ time ||s *xz | shuf | parallel -j 110% xzcat {} \I grep -F -f /v
ar/opt/arraytest/bluesmote/20220105/bad-urls \I wc -l | totus1.umk
```

```
top
top - 17:52:53 up 35 min, 21 users, load average: 22.46, 15.19, 7.27
Tasks: 831 total, 1 running, 830 sleeping, 0 stopped, 0 zombie
%Cpu(s): 0.2 us, 0.2 sy, 0.0 ni, 99.4 id, 0.2 wa, 0.0 hi, 0.0 si, 0.0 st
MiB Mem : 128739.8 total, 101006.9 free, 4518.1 used, 23214.8 buff/cache
MiB Swap: 75728.0 total, 75728.0 free, 0.0 used, 122957.8 avail Mem

  PID USER      PR  NI   VIRT   RES   SHR  S  %CPU  %MEM    TIME+  COMMAND
 19396 markj    20   0 3196220 341580 133356 S   7.3   0.3   1:53.72 Isolatu+
 5757 root      20   0    24.2g 77004 40060 S   3.6   0.1   0:38.22 Xorg
 5451 netdata  20   0   79276  12400  4240 S   3.0   0.0   1:02.04 upps.pl+
 5123 netdata  20   0 502520 144284 14412 S   2.0   0.1   0:37.64 netdata
 9001 markj    20   0 645652  37816 27964 S   2.0   0.0   0:05.06 xfcu4-p+
 9031 markj    20   0 484416  26852 21352 S   2.0   0.0   0:27.95 gkrellm
 8988 markj    20   0 711716  67088 48256 S   1.7   0.1   0:17.37 xfwm4
 9012 markj    20   0 998548 184160 30104 S   1.0   0.1   0:03.48 xfdockt+
 8750 markj    20   0 498852  27412 19984 S   0.7   0.0   0:01.90 xfcu4-s+
 9163 markj    20   0 236924  6248  4352 S   0.7   0.0   0:09.35 top
 10712 markj    20   0 236912  6240  4328 R   0.7   0.0   0:08.84 top
    1 root      20   0 176968  17312 10140 S   0.3   0.0   0:02.77 systemd
 4684 root      20   0  18580   9448  7888 S   0.3   0.0   0:00.86 systemd+
 5439 netdata  20   0  13324   3492  2988 S   0.3   0.0   0:02.01 tc-qos-+
 5462 netdata  20   0 203616  87168 31032 S   0.3   0.1   0:06.23 python
 9016 markj    20   0 700300  35032 27912 S   0.3   0.0   0:01.55 panel-8+
 9033 markj    20   0 412820  19804 17172 S   0.3   0.0   0:00.85 xfcu4-p+
```

```
iostat
sdc          0.00      0.00      0.00      0.00      0      0      0
avg-cpu:  %user   %nice %system %iowait  %steal   %idle
           0.24    0.00    0.18    0.17    0.00   99.41

Device            tps    kB_read/s    kB_wrtn/s    kB_dscr/s    kB_read    kB_wrtn    kB_dscr
nvme0n1           0.00         0.00         0.00         0.00         0         0         0
nvme10n1          0.00         0.00         0.00         0.00         0         0         0
nvme11n1          0.00         0.00         0.00         0.00         0         0         0
nvme1n1           0.00         0.00         0.00         0.00         0         0         0
nvme2n1           0.00         0.00         0.00         0.00         0         0         0
nvme3n1           0.00         0.00         0.00         0.00         0         0         0
nvme4n1           0.00         0.00         0.00         0.00         0         0         0
nvme5n1           0.00         0.00         0.00         0.00         0         0         0
nvme6n1           0.00         0.00         0.00         0.00         0         0         0
nvme7n1          10.50         0.00        102.00         0.00         0        204         0
nvme8n1           0.00         0.00         0.00         0.00         0         0         0
nvme9n1           0.00         0.00         0.00         0.00         0         0         0
sda              0.00         0.00         0.00         0.00         0         0         0
sdb              0.00         0.00         0.00         0.00         0         0         0
sdc              0.00         0.00         0.00         0.00         0         0         0
```

```
sar
05:52:08 PM    CPU      %user      %nice      %system      %iowait      %steal      %idle
05:52:10 PM    all        0.27        0.94        0.20        0.20        0.00      98.39
05:52:12 PM    all        0.23        0.89        0.15        0.17        0.00      98.56
05:52:14 PM    all        0.23        0.96        0.20        0.20        0.00      98.40
05:52:16 PM    all        0.25        0.95        0.20        0.20        0.00      98.40
05:52:18 PM    all        0.23        0.95        0.19        0.20        0.00      98.44
05:52:20 PM    all        0.17        0.92        0.20        0.21        0.00      98.50
05:52:22 PM    all        0.22        0.95        0.16        0.21        0.00      98.47
05:52:24 PM    all        0.35        0.93        0.25        0.21        0.00      98.26
05:52:26 PM    all        0.37        1.00        0.24        0.16        0.00      98.22
05:52:28 PM    all        0.48        0.98        0.34        0.20        0.00      98.01
05:52:30 PM    all        0.25        0.99        0.21        0.20        0.00      98.34
05:52:32 PM    all        0.50        0.94        0.32        0.16        0.00      98.08
05:52:34 PM    all        0.41        0.94        0.27        0.20        0.00      98.17
05:52:36 PM    all        0.44        0.97        0.33        0.20        0.00      98.07
05:52:38 PM    all        0.34        1.01        0.21        0.20        0.00      98.24
05:52:40 PM    all        0.34        0.80        0.29        0.20        0.00      98.37
05:52:42 PM    all        0.20        0.00        0.13        0.20        0.00      99.46
05:52:44 PM    all        0.43        0.00        0.24        0.20        0.00      99.13
05:52:46 PM    all        0.20        0.00        0.20        0.17        0.00      99.44
05:52:48 PM    all        0.18        0.00        0.20        0.18        0.00      99.45
05:52:50 PM    all        0.20        0.00        0.17        0.29        0.00      99.34
05:52:52 PM    all        0.25        0.00        0.18        0.17        0.00      99.40
```


squishycat



cat compressed files

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

**Or
uncompressed!**



squishycat: Use

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

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]$ █
```

Bibliography

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

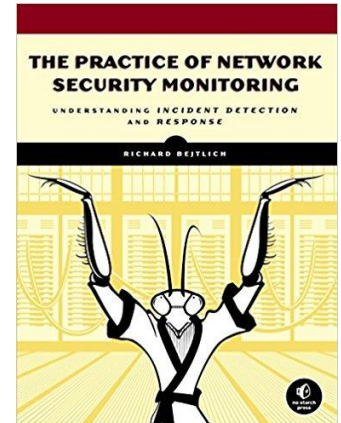
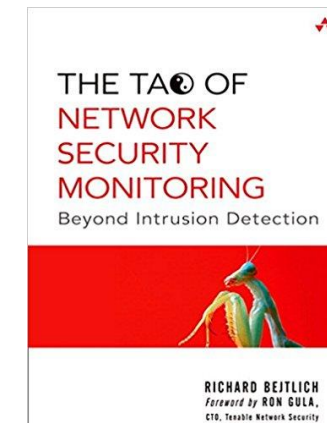
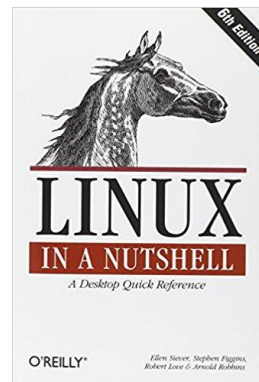
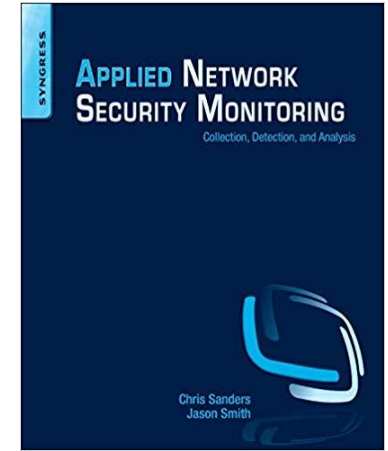
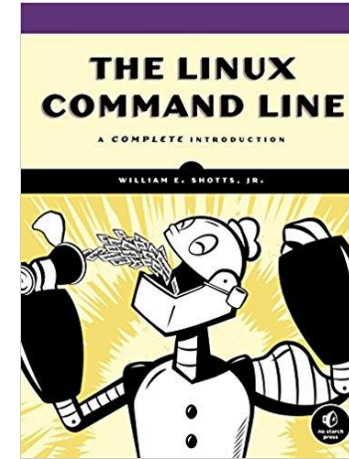
Questions?

markjx@gmail.com

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

Slide Deck & Scripts:

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



Appendix

~~Appendix~~

~~Appendixes~~

~~Appendices~~

Bonus Information!

Hardware Stopped Getting Faster

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

For my work: AMD Threadripper & NVMe PCIe g4 (or 5!)

This works with

- Any multi-core CPU
- Any SSD

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 \l 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 \l wc -l | totes1.awk  
real    0m44.252s  
user    9m39.629s  
sys     5m14.836s
```

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

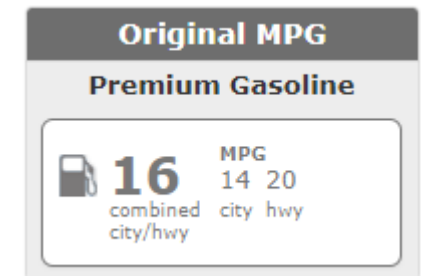
Compression Test – CERT Insider r6.2

			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
 - Downside? xz compression is much slower.
- 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 {} \l 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

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
```


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
```

parallel & ClamAV 2

An early run before I optimized the CPU usage

Another example of the importance of balancing CPU usage

See notes below

GPU's?

nVidia, AMD Radeon, Intel Xe are all processing units that specialize in SIMD. Can that help?

Maybe, but probably not.

I'm a scripter. Not a developer. Don't wanna. Feel free!

Overhead of copying to/from GPU's

People have looked into this, but not many results