

Speed up your day-to-day work: The Multi-Core Challenge

OpenSourceDays2010
2010-03-06

By: Lars Ole Belhage

The Day-to-Day Scenario

- Day-to-day is inherently Ad-Hoc
- Prepare external (& even internal!) input-files
- Analyze input, logs, archives
- Analyze static data (aka OLAP)
- Tracing
- Bugfixing – with actual fixing ;)
-

The Tools

- The magnificent UNIX tool-chain (coreutils +/-)
- Simple file stats (ls, ls -l)
- Conversion (tar, zip, tr, *iconv*, sed, lame, c)
- Details (wc -c/w, cut, sort, uniq, awk)
- Analysis (grep, awk, perl, c)
- Tracing (ip, nmap, tcpdump, *scp*)
- BugFixing (sed, awk, perl, ...)
-

The Expectations

- Data should be delivered in “my pace”
- Quick answers to easy questions
- Oldboys are used to old computers –
Youth Don't Wait...
(so we'd better start serve them well!)
- What's the difficulty
-

The Problem (real/myth/virtual)

- Computers are slow
- Data Volumes are ever growing (hands up – 1G, 2G, 5G, 1T, 2T, 5T, .. XE, ... - but how much data do you really consume)
- Gui's are slow
- Networks are slow
- Harddisks are slow
- Command lines are really slow
-

The I/O wait

- I/O is cheap (the NAS/NUT-case)
- RAM is cheap
- Disk is cheap
- Flash is cheap
- SSD is cheap'ish (ngD will be even cheaper ;)
- Pretty fast network is cheap
-

Testing

- Same idea – two ends
- Lower end (1*X4 4GB 1*SATA disk)
- Upper end (2*X4 96GB 6*SATA disk)
- “Big Data” seems to shrink
- dd/cat as a baselevel
- Pipes - part of the problem ?
-

File IO timings

- Echo “1” > /proc/sys/vm/drop_caches
- cat or “dd bs=kM” to /dev/null
- cat | cat > /dev/null
- splice-in file | cat > /dev/null

- Times (in secs):

Machine	Filesize	# Records	From disk	From gzip	From bzip2	From cache	thru Pipe	Axboe
Lower	2G	0.5M	22.7	11.8	82.5	0.57	2.3	1.5
Upper	16G	4M	23.2	71.7	141.7	2.21	6.6	4.1
Upper	62G	100M	90.1	382.5-		9.75	29.5	15.9

Tool waiting times

- Direct from disk utf-8, cached utf-8 and 8859
- wc, cut, grep, sort

Machi	Size	Disk	Cache	wc	cut -f60	grep	grep -i	sort -k	sort
Lower	2G	22.7	0.57	24.1/21.2/12.5	120/112/10.3	22.7/2.3/2.2	113/108/2.2	63/62/51	53/-/47
Upper	16G	23.2	2.21	101/96.2/39.6	495/495/35.7	23.8/9.5/9.4	493/480/10.9	-/4241/131.5	99.2/76.2/8.3
Upper	62G	90.1	9.75	805/-/337	1952.6/-/350.8	96.0/-/37.9	1970/-/44.4	-/-/	1976/-/1641

Speedup 1

- Run simultaneously on several files like “make -j” and many “parallel shellscripts”
- Lets split the file (4 and 8 way):

Cache	split	wc	Gain	grep -i	Gain
0.57	43.5	32.1/5.5/3.1	0.3 – 4.0	110/27.5/0.66	0.7 – 3.3
2.21	57.9	179.5/13.2/5.2	0.4 – 7.6	356.5/63.6/1.56	1.1 – 6.9
9.75	331.4	1062/125.5/54.6	0.5 – 6.2	1413/290.7/7.34	1.1 – 5.7

Speedup 2

- Split to pipes instead of files
- Avoids disklatency, simpler syntax
- “ms” - multisplit
- `ms -p# -fcmd [-jcmd] file`

0.57	22.1/6.2/3.1	1.0 – 4.0	56.1/27.0/1.5	1.5 – 2.0
2.21	100.1/15.5/7.3	1.0 – 5.4	65.7/64.1/5.5	1.9 – 7.5
9.75	492.8/106.8/47.1	1.1 – 7.1	258.8/254.1/21.0	2.1 – 7.6

- - note, although much worse in worst-case, the “shellscript” version is better in best-case...

Drawbacks

- Must be explicitly expressed
- Output is split in several parts
- No memory or linenums cross parts
- May need “joining/summing”:

split “awk '{++t[\$7]}END{for (v in t) print v "\t" t[v]}”

sum “awk '{t[\$0]+=\$1}END{for (v in t) print v "\t" t[v]}”



Speedup 3

- Make the coreutils threaded ?
- Supposedly honed for 20+ years
- Not designed for multicore
 - lots of global variables
 - io/linesplit/operations interleaved
- Current maintainer hard to contact
- So: recreate with same functionality
(choose via /etc/alternatives)

Technology

- OpenMP
- Pthreads with “manual” control
- CSP, PVM, MPI
-

Speedup 3b

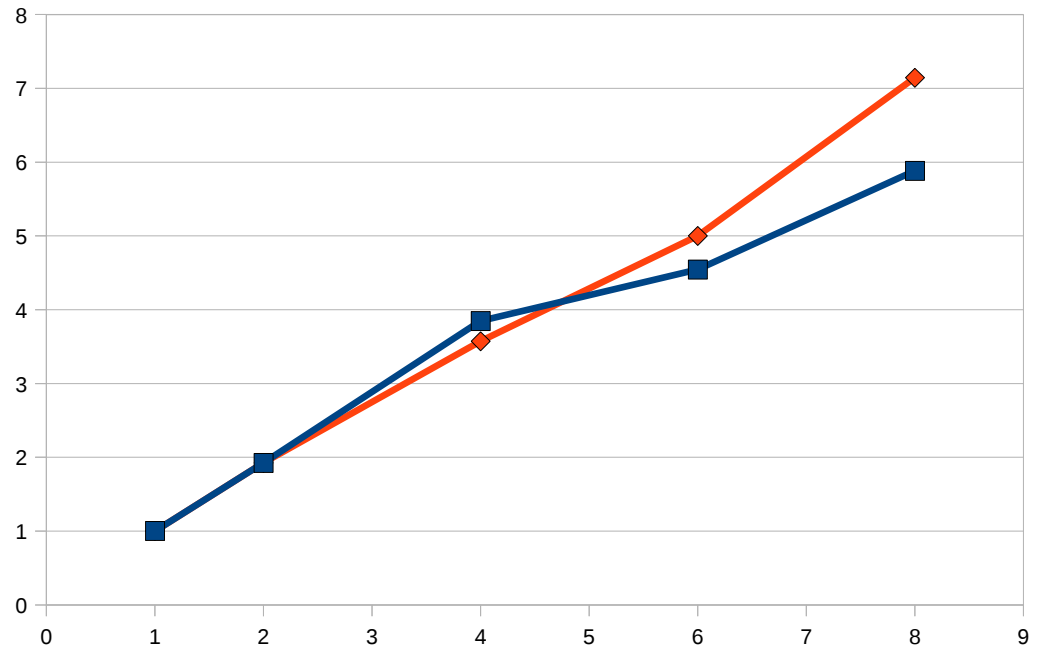
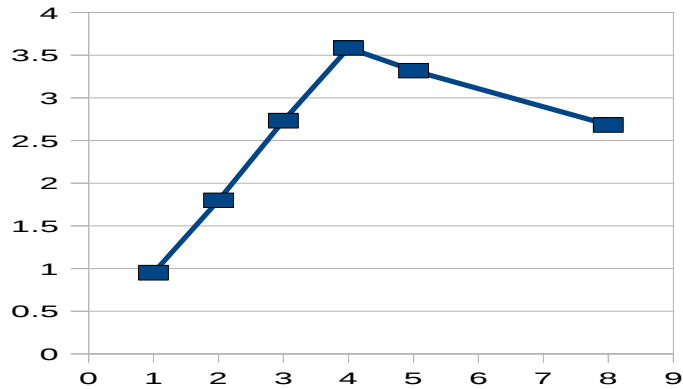
- With naïve inline version of wc and grep (memchr and regcomp/exec)

Cache	wc	Gain	grep -i	Gain
0.57	23.1/-/0.80	1.0 – 4.0	56.1/27.0/0.9	1.5 – 2.4
2.21	100.1/23.1/5.0	1.0 – 7.9	65.7/64.1/5.5	1.9 – 7.5
9.75	90.9/27.9/19.1	1.6 – 7.1	258.1/254.1/13.0	3.4 – 7.6

Does It Scale

- Yes ;) (well, only sort of – due to a bad implementer!)
- Lower: (speedup vs cores)

Better:



Conclusion

- Even simple tools are CPU-bound
- Coreutils should be multicore aware
- Everybody will benefit from faster tools
- Relatively hard to “fix” coreutils, may be more feasible to redesign (selected) tools
- Simplistic “tricks” are problematic
- Please help getting the message to the core!