

CAB data handling performance report

abaranov@fnal.gov

The purpose of this document is to understand scalability and performance issues of the cabsrv2 SAM data handling setup. The methodology is based on comparison of the CPU/Wall ratio of the job run in an idealistic single node environment with that been derived from the real-time historical data of SAM. I presume that scalable system, given the resources and the average use, will maintain both ratios numerically close. If rations are close, no changes to SAM data handling will likely improve system performance without scaling cluster itself.

What is measured

In the study I use following parameters to characterize particular type of the CAB job.

- 1) Average file processing time, or CPU consumption as time between file open and close calls.
- 2) Number of jobs run in a day
- 3) Number of files served in a day
- 4) Average file size consumed by the job in MB
- 5) Average waiting time for the job to get next file in sec.

CPU/Wall ratio is assumed as "Average file processing time"/("Average file processing time " + "Average file waiting time") . This formula treats “Wall” time as the sum of the CPU consumption and the time it takes to deliver a file from SAM. In reality, other factors may contribute to the Wall time of the job as well.

Data sample

Minimizing standard error of the study, the study picks the CAF-CS analysis as the most data intensive and parameter consistent job type. Again, the presumption attempts to minimize stddev of the base parameters by avoiding aggregation of the

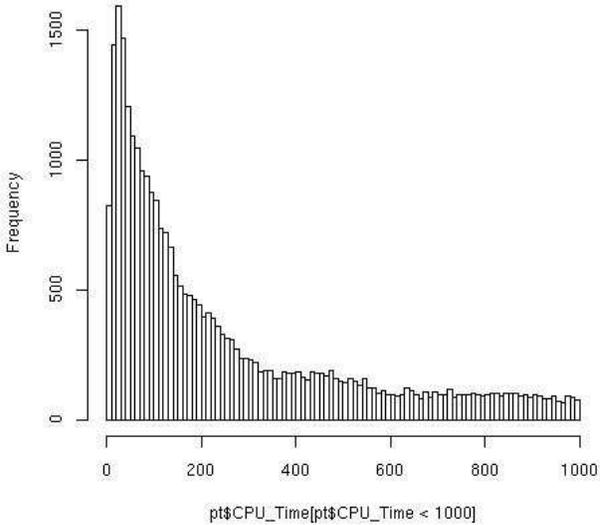
mixture of the job types that have vastly different characteristics. Data were collected on May 3 2006.

What are the most data intensive jobs? The table below compares CAF and TOP files analyses.

	Number of files	Average size MB
CAF-CS	34476	989.932
TOP	4650	171.378

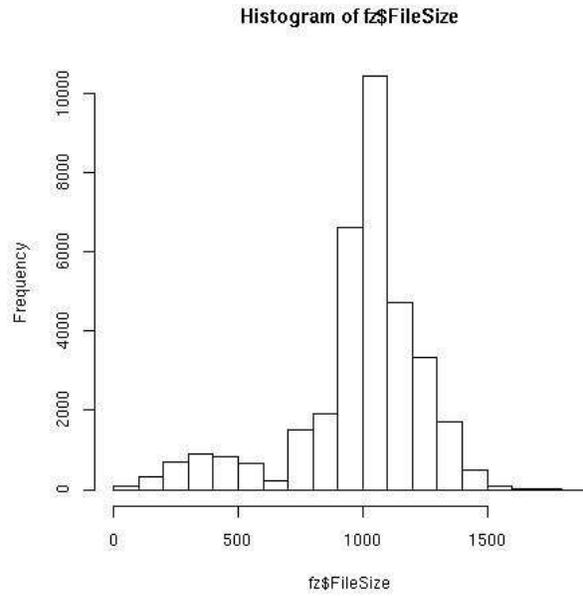
Clearly CAF jobs take the lead.

Now, rest of the parameters:

Avg. CAF file processing time/stddev	<p>Avg. = 504s stddev = 644s</p> <p>Histogram of pt\$CPU_Time[pt\$CPU_Time < 1000]</p> 
Number of CAF jobs	1084
Number of CAF files served	34476

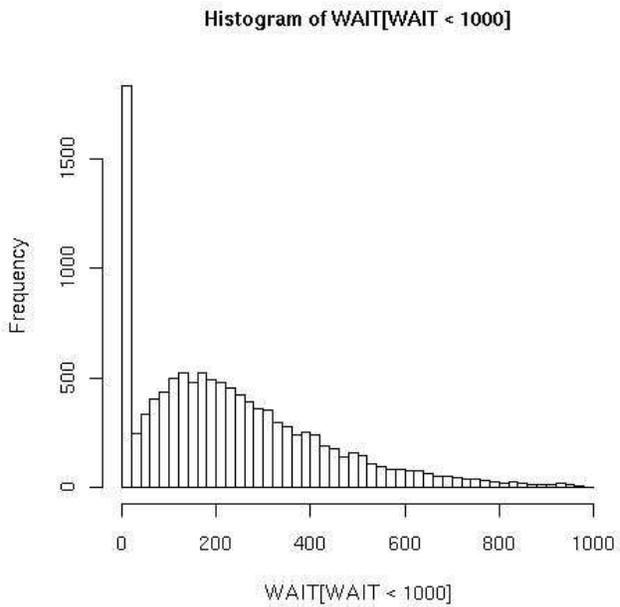
Avg. CAF file size / STD

Avg. = 989.9 MB
stddev=257MB



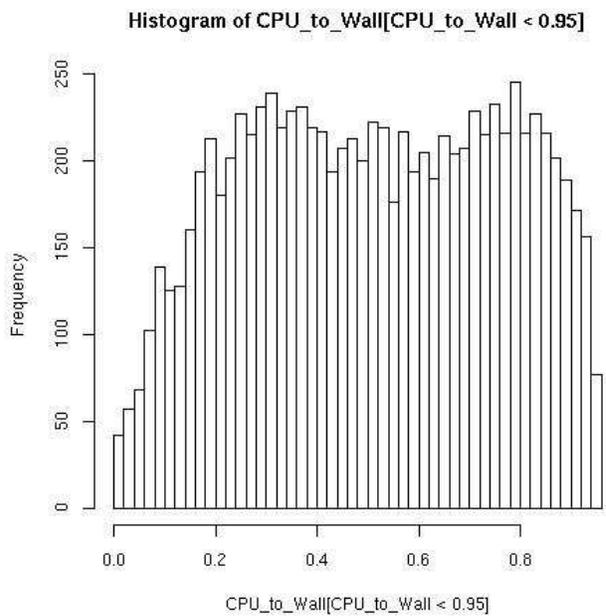
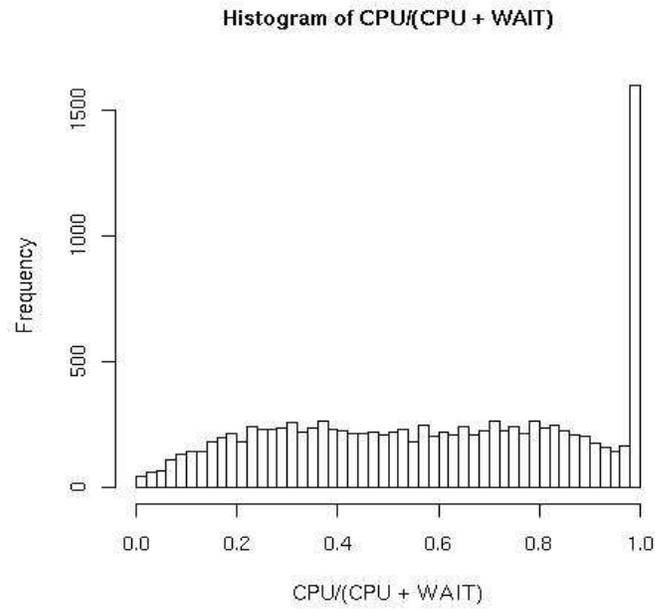
Avg. file waiting
time/STD

248 seconds



CPU/Wall

$\text{mean(CPU)/}(\text{mean(CPU)+mean(WAIT)}) = 0.6680397$
 $\text{mean(CPU/(CPU+WAIT))}=0.5849793$



* The distributions noticeably peaked at efficiency 1 and 0 wait times. These peaks reflect the cabsrv2 station setup that is tuned to deliver files in pairs. Thus, every other file is available at instant.

Ideal CPU/Wall (no presaging)

Idealistic data handling CAF job environment complies with following constraints:.

- 1) All job data resides on a remote server.
- 2) Job has an exclusive access to the data.
- 3) Files are requested sequentially.

Average transfer speed per file: 5Mb/s

Average file size: 989.932 MB

Ideal CPU/IO efficiency for a single CAF job

Data rate 5Mb/s -> 200 sec per file -> 200 + 60 = 260 sec per file with CRC check

CPU / Wall = 0.70

Results

The simulated CPU/Wall is the "same" as the one measured in production. This result suggests that, in fact, the existing data handling model on CAB is optimal (or within acceptable limits). However, surprisingly enough calculated CPU/Wall is at the conflict with plots generated by the batch system monitoring software. At that time (May 03), the reported CPU/Wall ratio was actually ~0.15. One way to look at the discrepancy is to question the assumption of CPU utilization by the job at times when data is available from SAM.

To analyze other factors that may contribute to the job "Wall" clock we've taken a snapshot of the machine state. See below the snapshot of "top" command on the single disk/dual processor CAF analysis worker node.

CPU	CPU_user	nice	system	irq	Softirq	iowait	Idle
Total	74.6%	0.0%	6.6%	0.0%	0.0%	118.6%	0.0%
cpu00	52.0%	0.0%	4.0%	0.0%	0.0%	44.0%	0.0%
cpu01	22.6%	0.0%	2.6%	0.0%	0.0%	74.6%	0.0%

CAF process states:

PID	USER	STAT	%CPU	%MEM	TIME	CPU	COMMAND
8456	Begel	D	52.2	38.7	1989m	0	cafe
18645	mundal	D	24.1	7.7	77:35	1	café

The node did not have neither incoming nor outgoing IO, yet the iowait was 118%.

In the model, analysis jobs read data from local disks. For the analysis at rates of 20Mb/s (see job processing time histogram plot), the sole disk becomes more expensive resource than CPU. Thus, disk not CPU effectively limits number of processes running on the machine.

Extrapolating the case, the machine equipped with one hard drive translates CPU utilization of dual nodes running CAF analysis to **%50**. That combined with **0.58** data handling efficiency yields cumulative CPU/Wall ratio of **0.28** or less. The statement should manifest more clearly as the job data consumption rates grows. Let fetch batch system wall time by user running CAF jobs

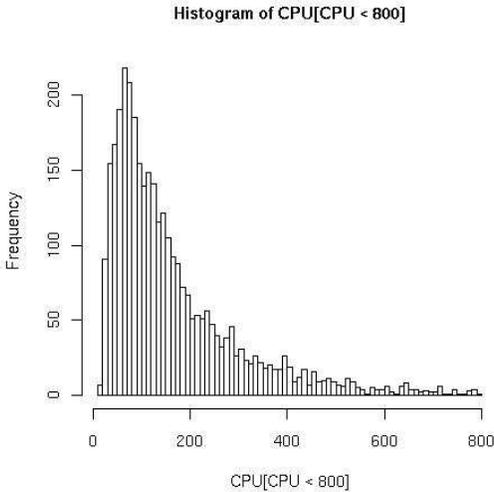
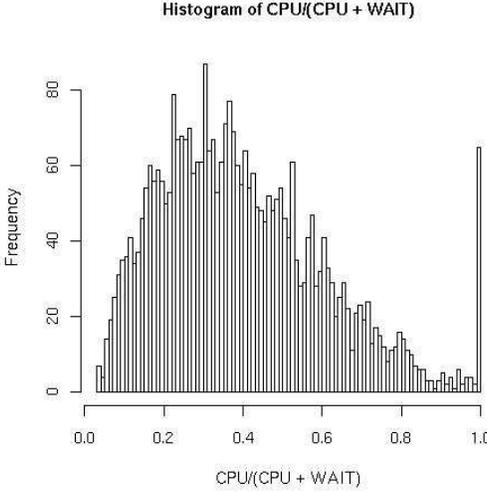
```
mysql> select sum(time_to_sec(walltime)) as wall ,
sum(time_to_sec(cput))/sum(time_to_sec(walltime)),user, count(*) from
pbs_accounting where rec_date = '2006-05-03' and user in
('begel','calfayan','chadj','charly','chunxuyu','duflot','fox','ghesketh','juste','mansoor','mar
kown','miruna','oleksiy','shfu','tuchming','venkat') and cluster='d0cabsrv2' and
queue='sam_lo' group by user order by wall;
```

wall	ratio	user	count(*)
703413	0.06	mansoor	57
785128	0.35	ghesketh	14
1624243	0.23	charly	97
2076586	0.14	chunxuyu	192
2292501	0.14	juste	26
2926798	0.16	miruna	684

4576242	0.22	oleksiy	48
6249425	0.06	fox	103

The average CPU/Wall is 0.15. Let's look at the biggest contributor:

User fox :

<p>Avg. CAF file processing time/stddev</p>	<p>Avg. = 150secs stddev = 265secs</p> <p>Histogram of CPU[CPU < 800]</p> 
<p>Number of CAF files served</p>	<p>10000</p>
<p>CPU/Wall</p>	<p>Avg 0.41 sd 0.26</p> <p>Histogram of CPU/(CPU + WAIT)</p> 

Again, there is a difference between reported and calculated CPU/Wall. Let see if disk read access might be an issue:

A test of the concurrent read of the 2 files on worker node:

```
rw-r--r-- 1 sam ods 1041292439 May 10 03:24 CAF-CSGv3-CSskim-MUinclusive-20060208-155015-2019758_p17.09.03_p18.05.00.root
```

```
-rw-r--r-- 1 sam ods 1028313869 May 10 03:26 CAF-CSGv3-CSskim-MUinclusive-20060208-181848-2019476_p17.09.03_p18.05.00.root
```

```
<d0cs346> date; cat CAF-CSGv3-CSskim-MUinclusive-20060208-155015-2019758_p17.09.03_p18.05.00.root > /dev/null ; date &
```

```
Thu May 11 07:38:04 CDT 2006
```

```
[1] 13298
```

```
<d0cs346> date ; cat CAF-CSGv3-CSskim-MUinclusive-20060208-181848-2019476_p17.09.03_p18.05.00.root > /dev/null ; date
```

```
Thu May 11 07:38:24 CDT 2006
```

```
Thu May 11 07:40:35 CDT 2006
```

```
Thu May 11 07:41:07 CDT 2006
```

```
[1] + Done (date; cat ...
```

```
<d0cs346>
```

It took 140 seconds to read 1Gb file. Now, let's assume this number as an average IO time per job and discount it against average "CPU" time reported by SAM.:

$$(170 - 140) / (170 + 250) = 0.07$$

Conclusion

The report can be succinctly summarized by stating that the single disk configuration of the CAB worker nodes is not optimal for jobs where the analysis duration is comparable to disk access time.

Action items

- a) Continue monitoring individual workers to collect more usage statistics
- b) Prohibit/control/reduce use of the worker nodes as an alternative source of the data. In CAB setup worker nodes are part of the cache. Thus, it is possible (and likely) that there can be many replicas of the same file scattered among primary

and worker cache nodes. By not having explicit preference in choosing a particular replica, station is more likely to select. Worker nodes as the data source for a given job/file.

Worker caches are not designed to sustain heavy IO and their use as storage resource may aggravate CPU utilization to an average less than %50.

* Station development effort ~1 week.

- c) Increase cumulative disk / network IO throughput. Disk fragmentation, network driver. Current values are (cumulative): disk access (16Mb/s), network 15 Mb/s. It was suggested to look into fragmentation and TCP setups of the worker nodes.
- d) Decouple storage and computing resources to increase flexibility in independent scheduling of the data / CPU intensive jobs.

Scripts:

CPU and WAIT times as reported in SAM log files:

```
function getCPID(line){
    split(line,tmp,"Consumer Process"); split(tmp[2],cpidA,"[()]"); return cpidA[2];
};

function getCPID2(line){
    split(line,tmp,"acquired CPID="); return tmp[2];
}

function tmstamp(){
    dt=$1; split(dt,dtA,"/"); timeStamp=$2; split(timeStamp,timeA,"."); return
int(dtA[2])*24*3600+3600*int(timeA[1])+60*int(timeA[2])+int(timeA[3]);
}

BEGIN { print "CPU","WAIT"; }

/acquired CPID={
    cpid = getCPID2($0);
    procTimeMap[cpid] = tmstamp();
    count[cpid] = 0;
}

/opens.* CAF-CS/{
    cpid = getCPID($0);
```

```

tm = tmstamp();
if ( cpid in procTimeMap ){
    timeDiff = tm - procTimeMap[cpid];
    if ( cpid in waitTimeMap )
        waitTimeMap[cpid] += timeDiff;
    else
        waitTimeMap[cpid] = timeDiff;
}
procTimeMap[cpid] = tm;
}
/closes.* CAF-CS/{
    tm = tmstamp();
    cpid = getCPID($0);
    if ( cpid in procTimeMap )
    {
        timeDiff = tm - procTimeMap[cpid];
        totalTime += timeDiff; numCached += 1;
        if ( cpid in waitTimeMap){
            if ( cpid in cpuTime )
                cpuTime[cpid] += timeDiff;
            else
                cpuTime[cpid] = timeDiff;
            count[cpid] += 1;
            if ( count[cpid] == 3 ){
                print cpuTime[cpid]/count[cpid], waitTimeMap[cpid]/count[cpid];
                waitTimeMap[cpid] = 0;
                cpuTime[cpid] = 0;
                count[cpid] = 0;
            }
        }
    }
}
procTimeMap[cpid] = tm;
}
END {

```

```
timeMap1 = 0;
for ( i in timeMap ) { timeMap1 +=1 ; } print totalTime/numCached,timeMap1;
}
```

**Average CAF file size , average CAF file transfer speed
(queued) , number of CAF files transferred.**

```
awk '/executed process.*\CAF.*/{ ok=1; } /Transferred/{ if ( ok == 1 ) {
totalTime += strtonum($5); totalSize += strtonum($2); n+=1; ok = 0; } } END { print
totalSize/n,totalSize/totalTime,n; }' ./sm_log__05_03_06
```