www.csiro.au

Strategies for data processing on a migrating file system

Robert C. Bell
CSIRO IM&T ASC

CSIRO

Background

- **With Peter Edwards, Gareth Williams, Aaron McDonough, Maciej Golebiewski and several users**
- **ASC Data Store – set up for data intensive computing**
  - Typically accepting and processing of data from climate models
  - Preparing data from climate models for down-scaling runs
- **Large data sets, sometimes large numbers of files, large i/o requirements**
- **Simple to complex workflows**
- **May involve transfers to other systems**
  - e.g. NCI NF

Processing constraints

- **Want best throughput**
  - analysis is a big bottleneck in much of this science
- **Users write own scripts, or get them from group members**
  - Start without any regard to the underlying file system
- **Not an issue with few files and small data**
  - often all on-line
- **Problems come with large numbers of files, and/or large file sizes**
- **Education process then starts: probably more items in our HPCbull on using the DMF HSM than on any other single topic in the last 18 years!**

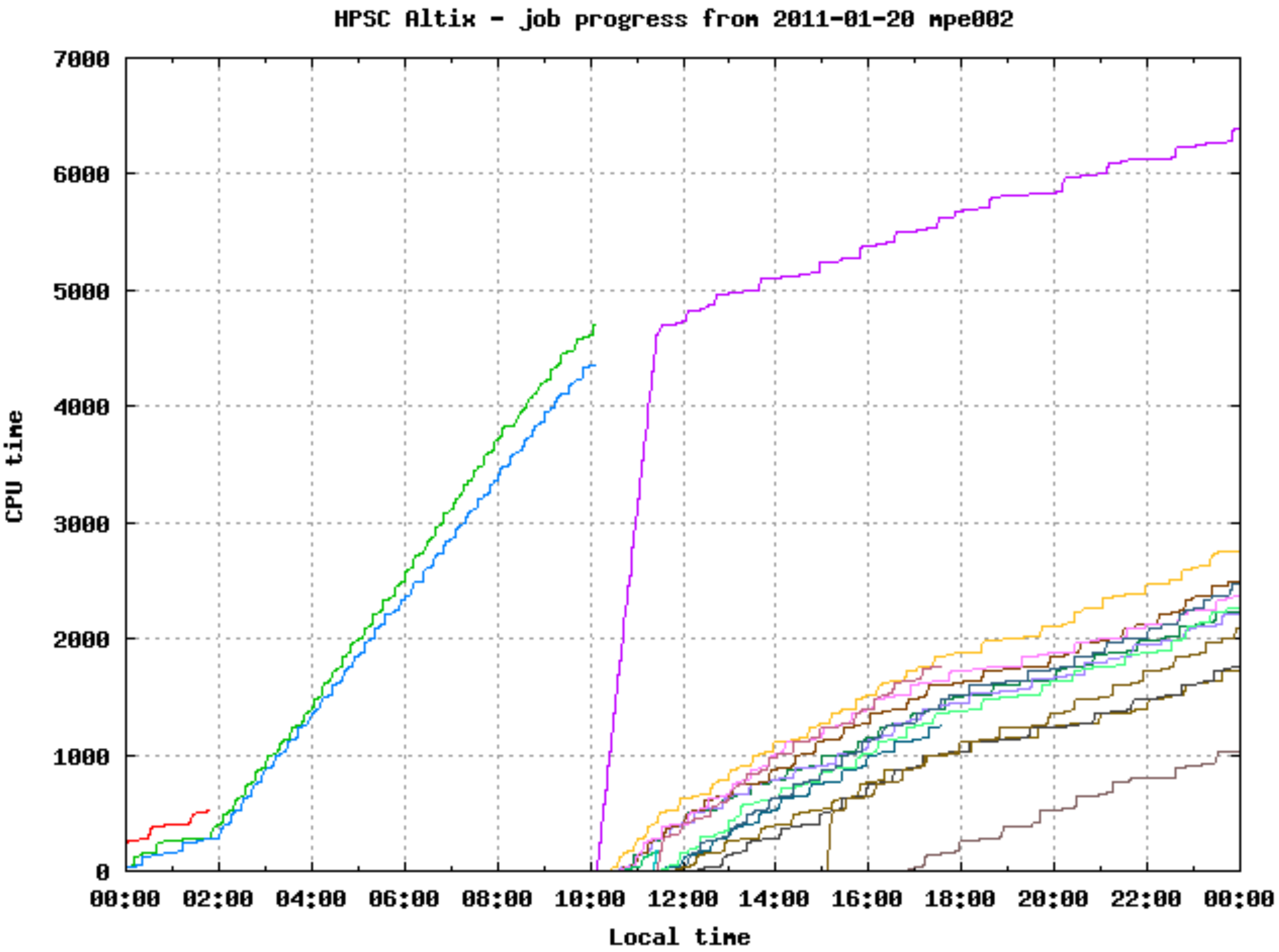
- **Presentation**
  - problems and some solutions to this kind of workflow
  - like passing a magnifying glass over the data

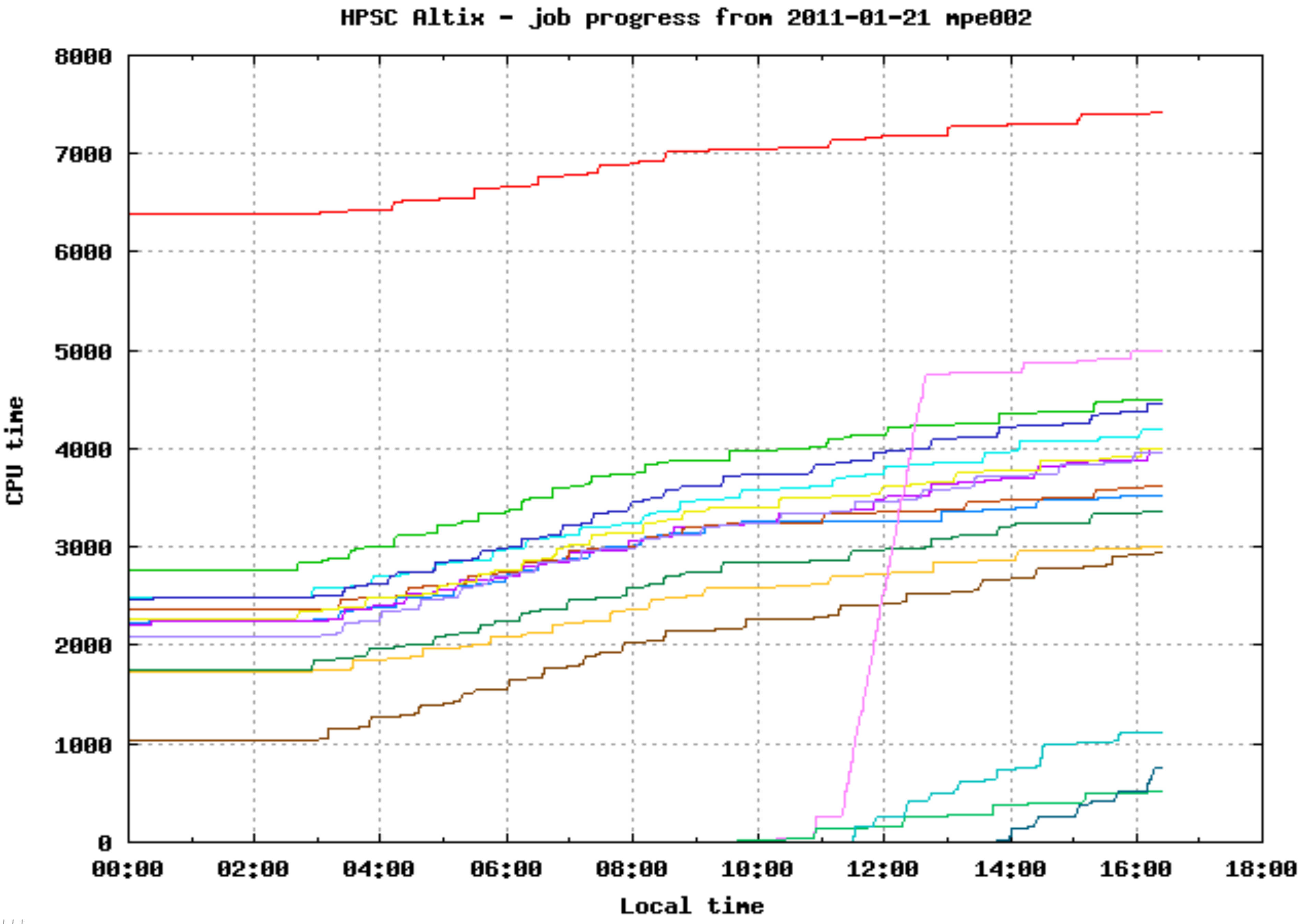
Problem 1: throughput too low

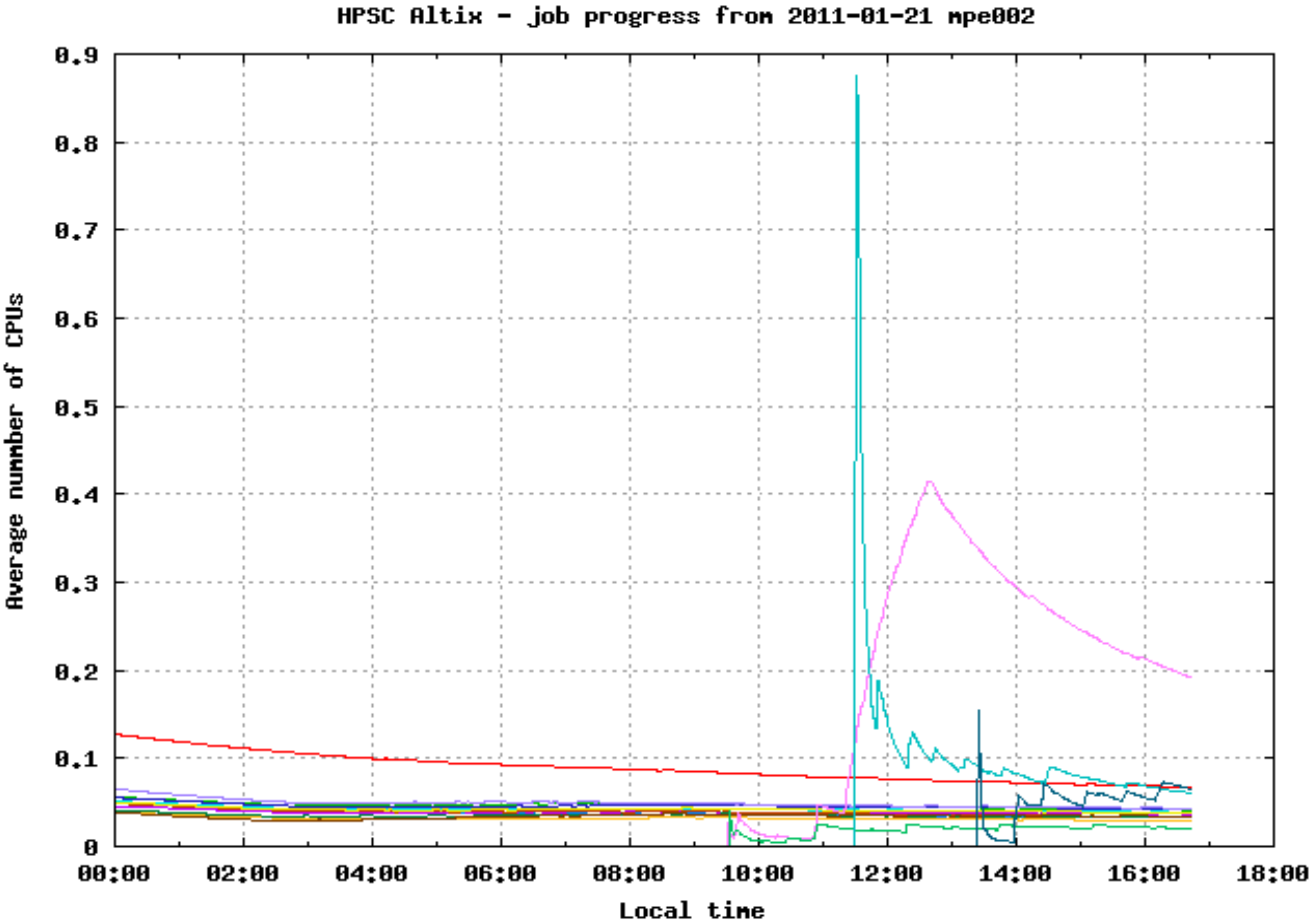
- **With a typical workflow, the pattern is:**

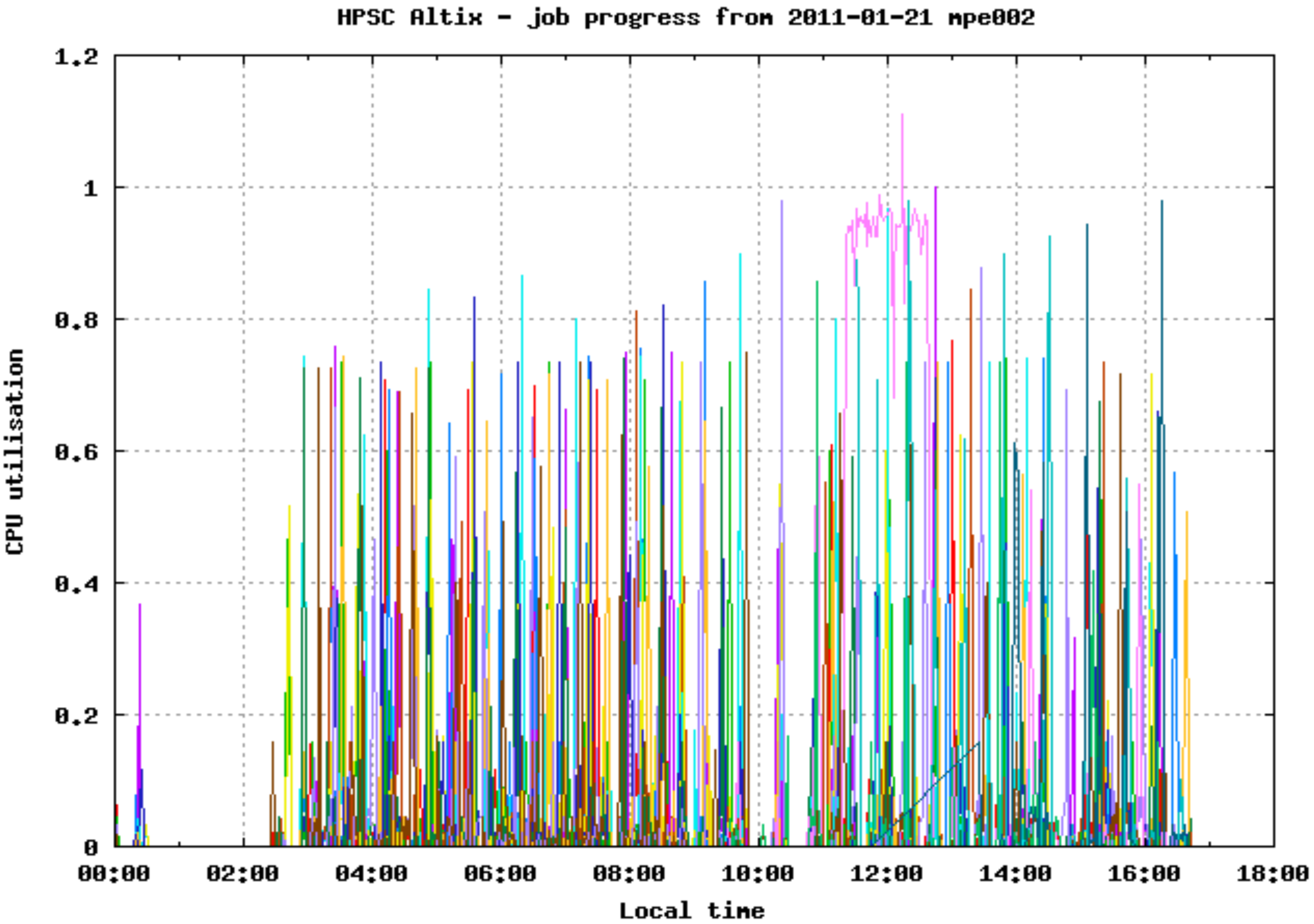
```
Loop over target files
   Read input file
   process
   write output file
End loop
```

- **On an HSM, this leads to low throughput**
  - (measure with average number of CPUs used: CPU time / elapsed time).


HPSC Altix - job progress from 2011-01-20 mpe002
CPU time
0
1000
2000
3000
4000
5000
6000
7000
00:00
02:00
04:00
06:00
08:00
10:00
12:00
14:00
16:00
18:00
20:00
22:00
00:00
Local time


HPSC Altix - job progress from 2011-01-21 mpe002
CPU time
8000
7000
6000
5000
4000
3000
2000
1000
0
00:00
02:00
04:00
06:00
08:00
10:00
12:00
14:00
16:00
18:00
Local time


HPSC Altix - job progress from 2011-01-21 mpe002
Average nummber of CPUs
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
00:00
02:00
04:00
06:00
08:00
10:00
12:00
14:00
16:00
18:00
Local time


HPSC Altix - job progress from 2011-01-21 mpe002
CPU utilisation
1.2
1
0.8
0.6
0.4
0.2
0
00:00
02:00
04:00
06:00
08:00
10:00
12:00
14:00
16:00
18:00
Local time

Problem 2: poor usage of tape drives

- **Each new file read leads to a tape mount**
  - Poor use of drives
  - (1 minute mount and position, 1 sec to read a 100 Mbyte file, another minute to rewind, dismount, replace in library).
  - More wear on tapes and drives
  - Reduced throughput for that user and everyone else

Solution 1: use dmget

- **Issue dmget command to explicitly recall the files**

```
dmget files*
Loop over target files
   Read input file; process; write output file
End loop
```

- **Allows DMF to efficiently recall multiple files from each tape**
- **Wrinkles**
  - Put the dmget command in the background, so that processing can start as files are recalled
  - Insert another dmget command, so that processing aborts when a file can't be recalled.

```
dmget files* &
Loop over target files
   dmget thisfile && read input file; process; write
    output file
End loop
```

Problem 3: users using dmget hog the system

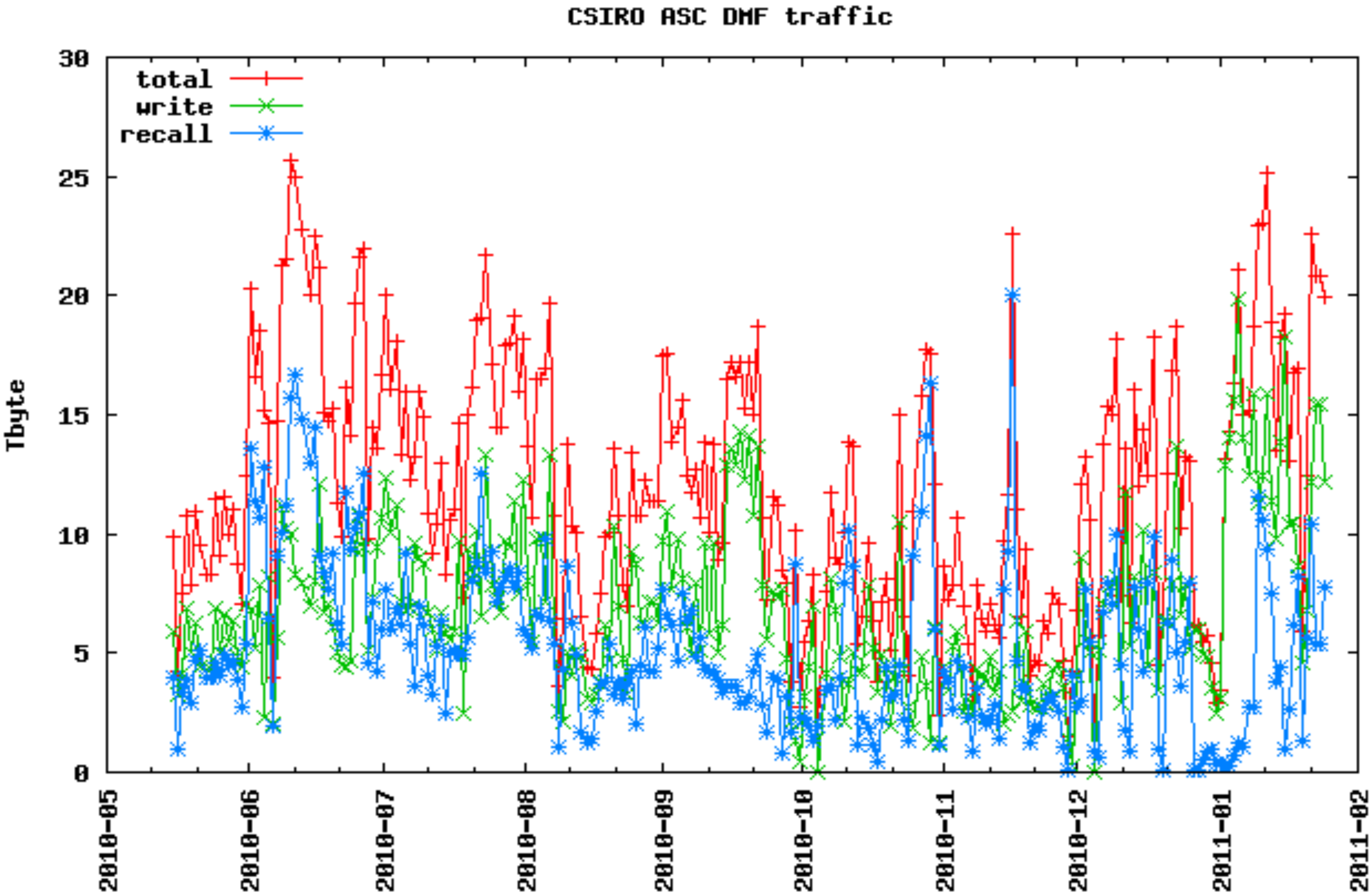
- The DMF request queue is mostly FIFO
- One user's large request can block other users for hours

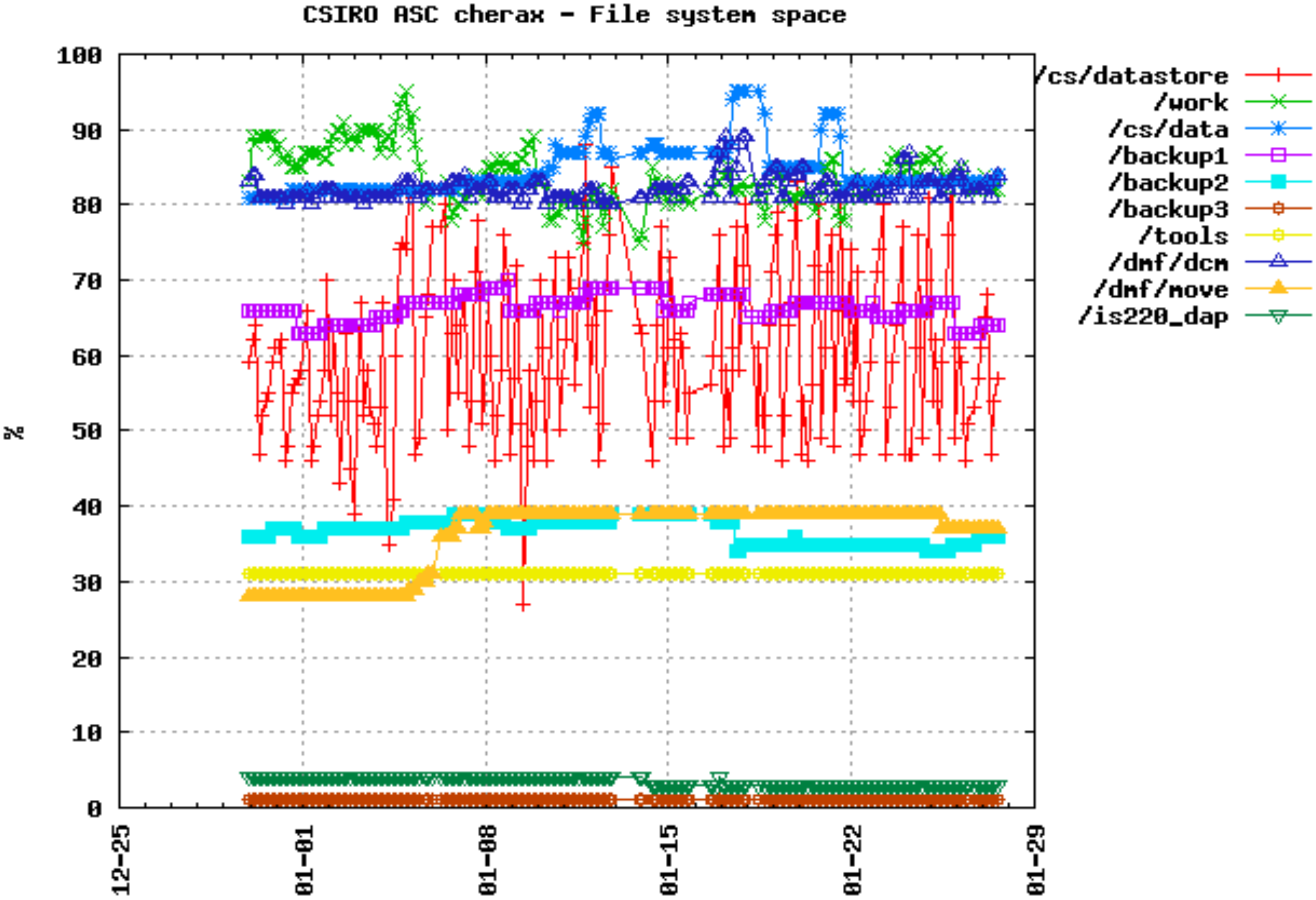
Solution 2: local dmget wrapper

- **First version broke requests up into lumps, based on the number of files and amount of data**
  - Efficiency within lumps
  - Serialised the lumps, to allow other users’ requests to be serviced between lumps
  - Lost some efficiency in tape usage

Problem 4: my files keep getting put away before I can use them

- With a busy system, recalled files can get freed before being used
- To preserve POSIX, a file recall does not count as an access
- Need to update access time (since you are about to read the file anyway)


CSIRO ASC DMF traffic
Tbyte
30
25
20
15
10
5
0
total
write
recall
2010-05
2010-06
2010-07
2010-08
2010-09
2010-10
2010-11
2010-12
2011-01
2011-02


CSIRO ASC cherax - File system space
%
100
90
80
70
60
50
40
30
20
10
0
12-25
01-01
01-08
01-15
01-22
01-29
/cs/datastore
/work
/cs/data
/backup1
/backup2
/backup3
/tools
/dmf/dcm
/dmf/move
/is220_dap

Solution 3: add a -a flag to dmget

- **Solution 3: add a -a flag to dmget**
- **(Used to advise using touch -a)**
  - now a flag on dmget
- **Updates access time**
- **Means that recalled files are no longer the prime targets for the DMF freeing process**

Solution 4: recall files in smaller batches

```
dmget batch1* &
Loop over batches
   If (not last batch) dmget batch_next* &
   Loop over files in this batch
      dmget thisfile && read input file; process;
     write output file
   End loop
End loop
```

Problem 5: The file system filled, or I hit my quota limit

- We impose quota limits on on-line space in the /cs/datastore
- In 6 Tbyte filesystem, have default quota of 2 Tbyte, to stop one user dominating to the detriment of others

Solution 5: dmput recalled files after use

- Add a dmput -r onto the recalled files
- Release disc space for dual state files

```
dmget batch1* &
Loop over batches
   If (not last batch) dmget batch_next* &
   Loop over files in this batch
     dmget thisfile && read input file; process; write
    output file; dmput -r thisfile
   End loop
End loop
```

- **Wrinkle**
  - Don’t want users generating dmput request for new files, since we want the system to batch these together.
  - Use local dmput wrapper supporting the -Q flag
    - only dual and partial state files are released.
  - dmput -r -Q thisfile

Problem 6: The user whose files I am recalling hit a quota limit

- Often the user doing the recalling does not own the files
- Lots of recalls cause the owner to hit an on-line space quota limit
- The SGI dmput does not allow users to dmput other people’s files
- Had tried various ruses with .rhosts files to allow limited cross-user access
  - **not satisfactory**

Solution 6: local dmput wrapper allows cross-user dmputting

- Allows dmput on another user’s files, provided the initiating user
  - **has read access to the file**
  - **belongs to the group of the file.**

Problem 7: original dmget wrapper does not allow recall efficiency

- When users call the original dmget wrapper on their batches of files, and this breaks these batches into lumps, then a lot of the built-in DMF tape access optimisation is lost

Solution 7: New local dmget wrapper

- **Restores the DMF tape recall efficiency, by making batches by tape volume**
- **From man dmget:**
  - (CSIRO only) CSIRO's wrapper around the SGI dmget program is intended to prevent one user who requires a large amount of data to be recalled from locking out a following user with more modest demands.
  - It does this by determining which tapes the files reside on, and processes them in batches tape by tape.
  - This minimises tape mounts and multiple passes over tapes, which is kinder to the system and would result in faster processing for the user.
  - Batches by one user may be interleaved with batches from a different one.

http://http://hpsc.csiro.au/users/dmfug/Meeting_Oct2009/Presentations/dmget_wrapper/

Problem 8: dmget wrapper lumps don’t match user batches

- Losing efficiency again
- User breaks work up into convenient batches, e.g. a year at a time
- This does not coincide with optimum tape batches

Solution 8: New local dmget wrapper option --list

- **dmget --list files***
- **A different solution, if the order in which files are processed doesn't matter, is to use a feature of the wrapper where it will perform a dummy run, listing the files in the order in which it would have recalled them, but without actually doing so. (From man dmget)**

```
dmget --list file1 file2 file3 file4 > $TMPDIR/lof
                dmget < $TMPDIR/lof &
                for f in ` cat $TMPDIR/lof `; do
                     process_one_file $f
                     dmput -r $f
                done
```

- **Wrinkles**
- **Local dmget has options:**
  - --list, --defer, --recurse

Problem 9: processing still waits for files

- Can have imbalances, depending on the times to recall files compared with the time to process files
- Use parallel processing with background tasks, and control the number of background tasks depending on how much impact you want on other users, and the extent of the imbalance.

Solution 9: Use parallel

- **Put the commands to be executed into a file: e.g.**

```
Loop over files (in dmget --list order)
    cat < EOF >> $TMPDIR/commands
    dmget -a thisfile && process thisfile ; dmput -r -Q
      thisfile
    EOF
End loop
```

- **Execute the commands in batches in parallel**

```
parallel -j $max_bg < $TMPDIR/commands
```

- **$max_bg should be determined by various factors**
  - number of requested CPUs in the batch job
  - the typical size of the files
  - the desired working set size (of on-line data) for the processing

Problem 10: recalls issued for files that don't need to be processed

- **When reruns are done, or when transferring files, don't want to re-issue recalls for files already processed.**
- **For example, when copying files to a remote location, there are often failures, and the script is re-run, with wasted recalls.**

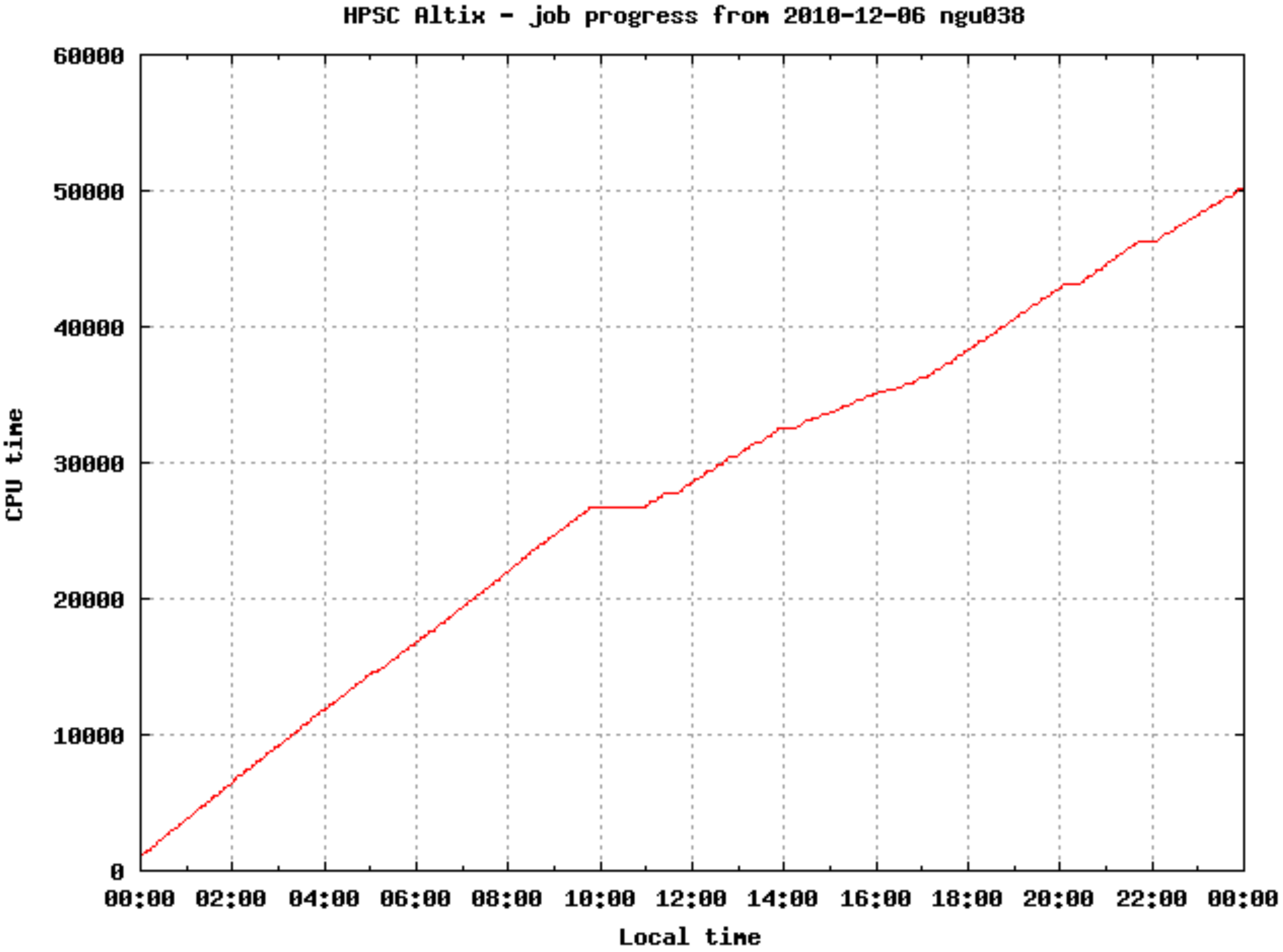
Solution 10: Use rsync dry-run

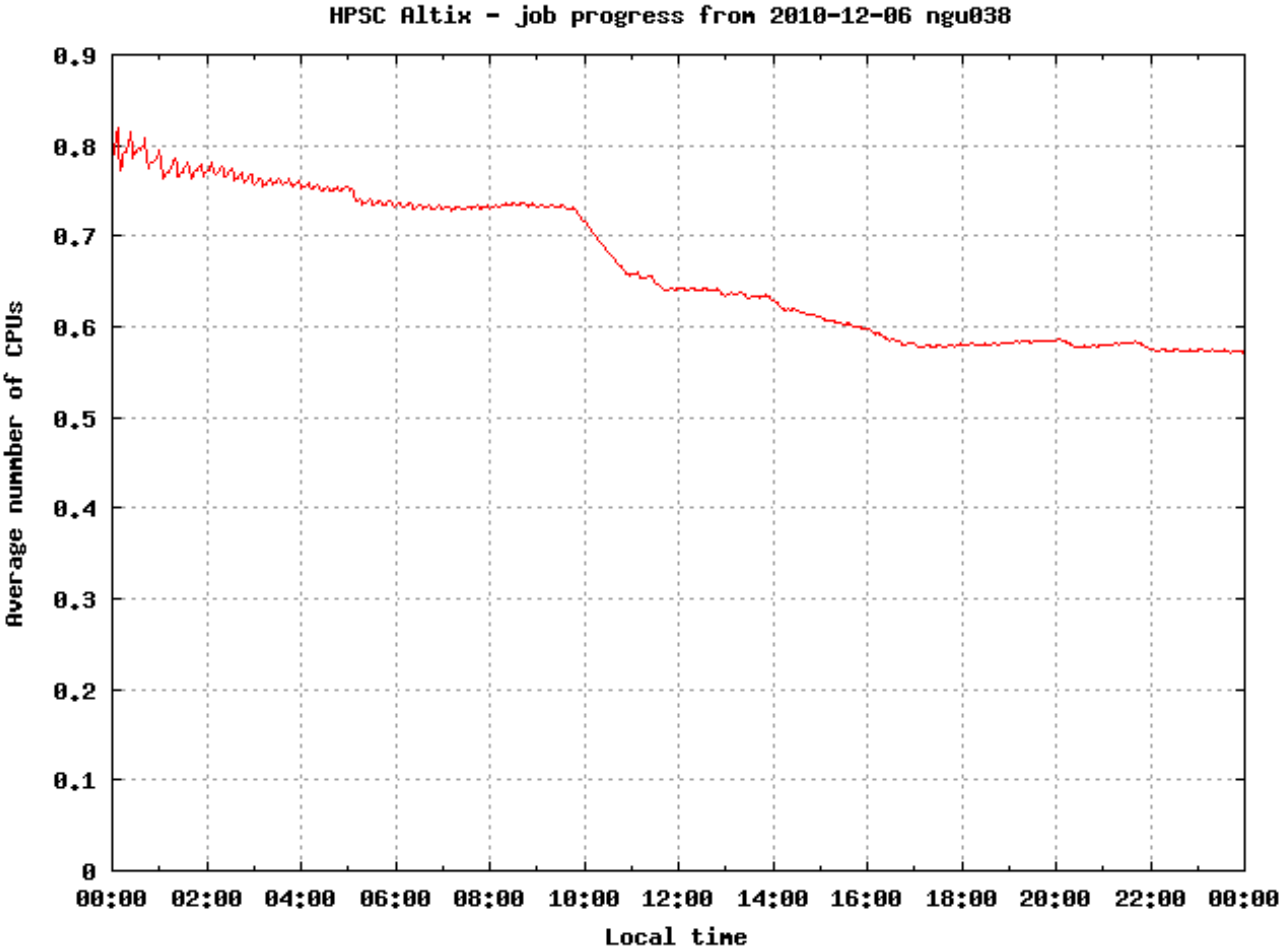
- **rsync --archive --verbose --dry-run \**

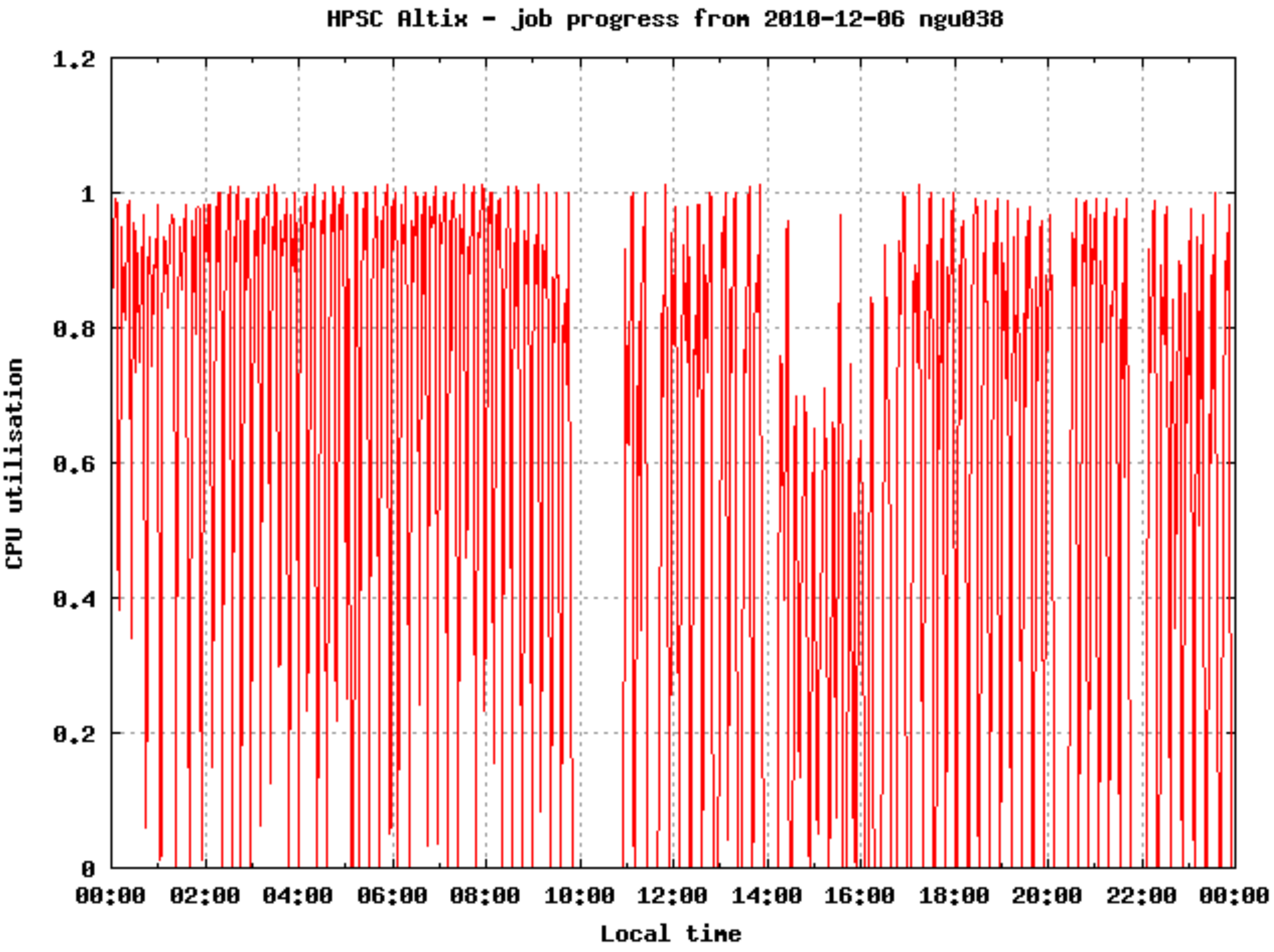
  [other options] files destination:

  (rsync –anv ...)

and capture the list of files that would need to be transferred to do the real update.


HPSC Altix - job progress from 2010-12-06 ngu038
CPU time
Local time
0
10000
20000
30000
40000
50000
60000
00:00
02:00
04:00
06:00
08:00
10:00
12:00
14:00
16:00
18:00
20:00
22:00
00:00


HPSC Altix - job progress from 2010-12-06 ngu038
Average nunmber of CPUs
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
00:00
02:00
04:00
06:00
08:00
10:00
12:00
14:00
16:00
18:00
20:00
22:00
00:00
Local time


HPSC Altix - job progress from 2010-12-06 ngu038
CPU utilisation
Local time
1.2
1
0.8
0.6
0.4
0.2
0
00:00
02:00
04:00
06:00
08:00
10:00
12:00
14:00
16:00
18:00
20:00
22:00
00:00

Example: transferring files to NCI NF. Part 1

```
# Set up:
source_dir=~bel107/tests-cherax
file_mask=job.progress.2011-01-1'*'
dest_dir=test-transf
dest_host=vayu.nci.org.au
dest_user=rcb599
transf_cmd='rsync --archive --rsh=ssh --partial \
   --whole-file'
```

Example: transferring files to NCI NF. Part 2

```
cd $source_dir
max_bg=5  # Max. number of background processes.
max_Mbyte=40000  # Max. amount of data to be recalling at any time.
# Find a sample file size, and ensure that the amount of data being
# recalled does not exceed $max_Mbyte, nor the number of files (and
# hence processes) does not exceed the initial value of $max_bg.
file_sizeM=$(ls -al $file_mask | tail -n 1 | \
   awk '{print int($5/1000000)}')
if [ $file_sizeM -le 0 ] ; then
   file_sizeM=1
fi
((nfiles=$max_Mbyte/$file_sizeM))
if [ $nfiles -lt $max_bg ] ; then
   max_bg=$nfiles
fi
print \$max_bg is $max_bg
```

Example: transferring files to NCI NF. Part 3

# Collect a list of only those files that need updating.

```
$transf_cmd -n --out-format='%n' $file_mask \
   ${dest_user}@${dest_host}:${dest_dir} > \
   $TMPDIR/file.list.1
```

# Set up the files in order by tape volume.

```
dmget --list < $TMPDIR/file.list.1 > $TMPDIR/file.list.2
```

# Initiate a dmget for the first batch.

```
head -n $max_bg $TMPDIR/file.list.2 | $dmget_cmd -a &

/bin/rm $TMPDIR/jobqueue
touch $TMPDIR/jobqueue
```

Example: transferring files to NCI NF. Part 4

```
# Set up commands in a file, in optimum DMF order.
for ifil in $(cat $TMPDIR/file.list.2) ; do
  print setup for $ifil
  #  Ensure the required file is present before proceeding.
  # Pin the dmget to the processing.
/bin/echo " \
  echo processing $ifil ; $dmget -a $ifil ; \
   ($transf_cmd -v $ifil \
   ${dest_user}@${dest_host}:${dest_dir} &&\
    $dmput -r -Q $ifil ) 1> $TMPDIR/out.$ifil.1  2>&1  \
" >> $TMPDIR/jobqueue
done
```

Example: transferring files to NCI NF. Part 5

# Add a dummy command to finish the pipe.

```
echo "echo last command executed, but earlier ones may \
be still running" >> $TMPDIR/jobqueue
```

# Now set off the parallel execution.

```
parallel -j $max_bg < $TMPDIR/jobqueue
```

Example: transferring files to NCI NF. Part 6

# Loop to ensure all is done (may not be needed).

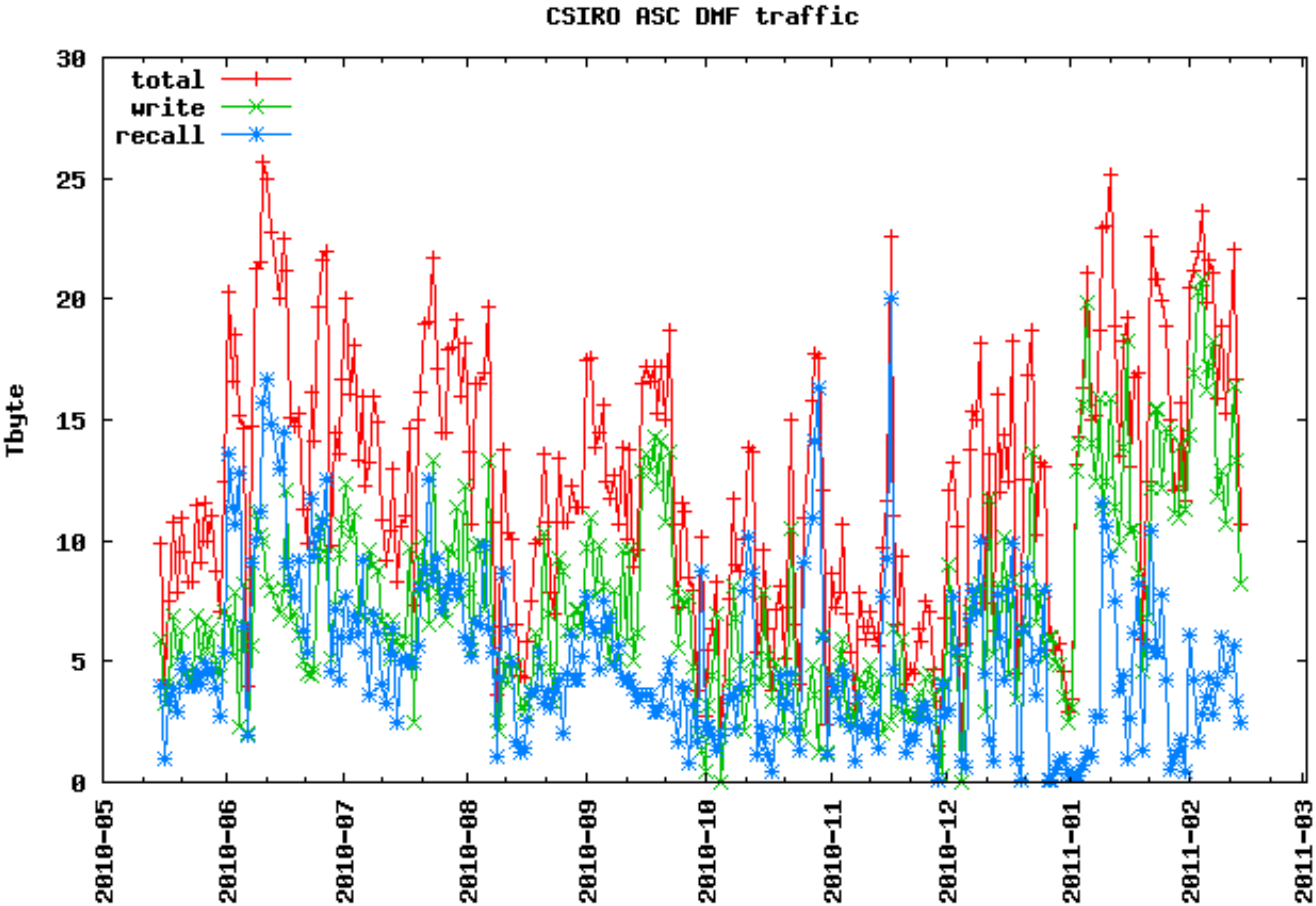
```
max_loops=10
 i_loop=1
 wait_time=10
 while [ $i_loop -lt $max_loops ] ; do
   njobs=$(ps --no-headers | tee $TMPDIR/ps.out | wc -l)
   cat $TMPDIR/ps.out
   if [ $njobs -gt 5 ] ; then
     echo $njobs processes in total - sleep for \
       $wait_time seconds
     sleep $wait_time ; (( wait_time = $wait_time * 2 ))
   else
     break
   fi
   (( i_loop = $i_loop + 1 ))
 done
```

Example: transferring files to NCI NF. Part 7

```
wait
# Collect the output files.
cat $TMPDIR/out.*
# Optional - files should be removed by system if run as batch job.
# However, for repeated interactive execution, do the remove.
/bin/rm $TMPDIR/out.*
```

Further notes and work

- **The above relies on passwordless ssh being set up**
- **Uses hpn-ssh by default**
- **Does not use gridftp (anti-social?), but does do parallel recalls, parallel processing, parallel file transfers**
- **Key techniques are:**
  - to try to do the recalls in the best possible order
  - to tie the processing/transfer to the individual recall, and
  - to do as much as reasonable in parallel
- **Could check quota usage**
  - have done in another version
- **Could use this script to transfer files from migrating file system to scratch/work/flush area, and then work in there**
- **Could set up utility**
  - mcp like scp, but aware of source being on migrating file system
- **Hard to make a general facility**
  - Big differences in techniques depending on the sizes and numbers of files, and the relative amount of processing involved.


CSIRO ASC DMF traffic
total
write
recall
Tbyte
30
25
20
15
10
5
0
2010-05
2010-06
2010-07
2010-08
2010-09
2010-10
2010-11
2010-12
2011-01
2011-02
2011-03

Conclusion

- HSM
  - key technology, but hard to build efficient workflows
- Applies to ‘archive’ services as well
  - need intelligence in the recall process
- User education!!!
- dmgets are the key
- locally-written version provides important enhancements
- enhanced dmput as well
- The process of putting data into an HSM is not so difficult!

CSIRO IM&T
Robert Bell
Technical Services Manager, Advanced Scientific Computing

Phone: (03) 9545 2979
Email: Robert.Bell@csiro.au
hpchelp@csiro.au
Web: http://intranet.csiro.au/intranet/imt/eResearch/asc.htm
http://www.hpsc.csiro.au/contact
ASC Helpdesk: (03) 8601 3800

Thank you

Contact Us
Phone: 1300 363 400 **or** +61 3 9545 2176
Email: Enquiries@csiro.au **Web:** www.csiro.au

CSIRO