Performance of the LOTAAS v.1 pipeline on cartesius

Time taken by the individual pipeline components per beam (24-core node)

fits2fil: 6min
rfifind: 15min
mpiprepsubband (253 trials): 3min
single pulse search: 1min
realfft: 10sec
rednoise: 10sec
accelsearch (-zmax=0 ; –numharm=16): 1min20sec
accelsearch (-zmax=50 ; –numharm=16): 12min
accelsearch (-zmax=50 ; –numharm=8): 5min
accelsearch (-zmax=50 ; –numharm=8): 26min
plots: 20sec
python sifting and folding: 21min
pfd scrunching: 5sec
data copying: a few secs
candidate scoring: a few secs

Total time spent for the first large set of DM trials (0-4000)

mpiprepsubband: 40min
sp: 16min
realfft: 3.5min
rednoise: 3.5min
accelsearch (zmax=0;numharm=16): 21min
accelsearch (zmax=50;numharm=16): 192min
accelsearch (zmax=50;numharm=8): 80min
accelsearch (zmax=200;numharm=8): 416min

Total time spent for the second large set of DM trials (4000-10000)

mpiprepsubband: 24min
sp: 8min
realfft: 2min
rednoise: 2min
accelsearch (zmax=0;numharm=16): 11min
accelsearch (zmax=50;numharm=16): 96min
accelsearch (zmax=50;numharm=8): 40min
accelsearch (zmax=200;numharm=8): 208min

% time alloc.	zmax=0;numharm=16	zmax=50;numharm=16	zmax=50;numharm=8	zmax=200;numharm=8
fil conversion	**3**	**1**	**2**	**<1**
rfifind	**9**	**3**	**6**	**2**
dedispersion	**37**	**16**	**25**	**8**
sp search	**14**	**5**	**9**	**3**
realfft	**3**	**1**	**2**	**<1**
rednoise	**3**	**1**	**2**	**<1**
accelsearch	**18**	**67**	**46**	**81**
folding	**12**	**5**	**8**	**3**
data copying/etc	**1**	**1**	**1**	**<1**

Total processing time per beam (zmax=0;numharm=16): ~3hours
Total processing time per beam (zmax=50;numharm=16): ~7hours
Total processing time per beam (zmax=50;numharm=8): ~5hours
Total processing time per beam (zmax=200;numharm=8): ~13h40m

Performance of the LOTAAS v.1 GPU pipeline on cartesius

mpiprepsubband (253 trials): 38sec

Data transferring (CEP$/LTA)

32-bit to 8-bit downsampling on CEP2 (per observation): 6-8 hours
Transferring from CEP2 to LTA (per observation): 2-3 hours
Observation downloading on cartesius (1-core): ~8hours
Observation downloading on cartesius (home area, 8jobs in parallel.sh):<2hours

Benchmarks for filterbank creation with psrfits2fil

A series of tests were ran on dragnet (drg01), directly (now through slurm), on fits files from a random LOTAAS observation. Number of cores,means number of different fits files processed simultaneously. The total time is an extrapolation of this benchmark.

Only one test was ran for each occasion. Repeating them showed differences in execution time, but I doubt that the results change qualitatively.

Input/Output on the same disk

1-core:354sec
Total time for 16 files:5664sec

3-cores:630sec
Total time for 16 files:3360sec

4-cores:720sec
Total time for 16 files:2880sec

5-cores:1106sec
Total time for 16 files:3539sec

8-cores:1735sec
Total time for 16 files:3360sec

16-cores:3778sec
Total time for 16 files:3778sec

When writing the filterbank in the same disk with the fits file, the best performance is achieved by having 4 psrfits2fil instances running.

Input/output on different disks

1-core:329sec
Total time for 16 files:5264sec

4-cores:523sec
Total time for 16 files:2092sec

8-cores:901sec
Total time for 16 files:1802sec

16-cores:1513sec
Total time for 16 files:1513sec

24-cores (hyperthreaded):2252sec scales perfectly!
32-cores (hyperthreaded):3346sec performance loss already evident;~340 sec slower than expected

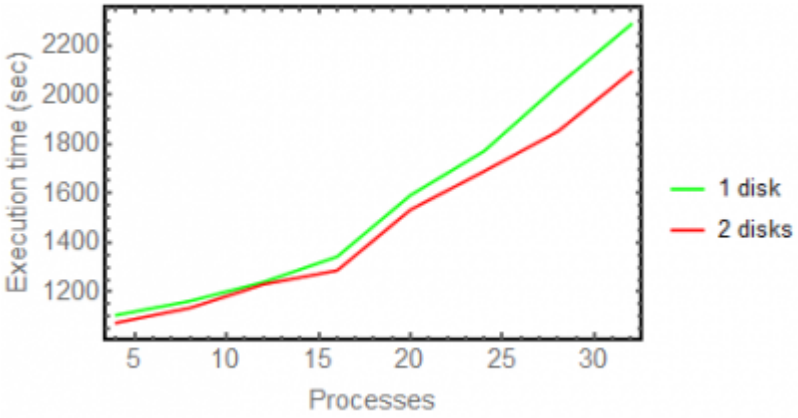
Using all the cores in the dragnet nodes gives the best performance. Moreover, hyperthreading behaves exactly as having the extra cores.

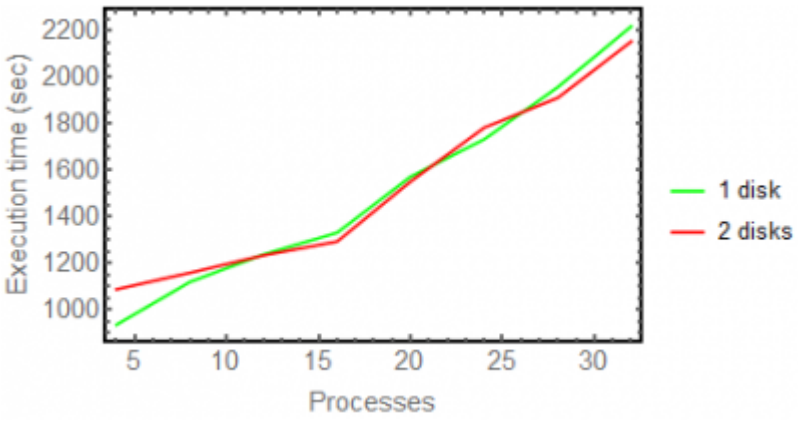
Moving a fil file from /data2 to /data1 takes 1 minute.

rfifind benchmarks

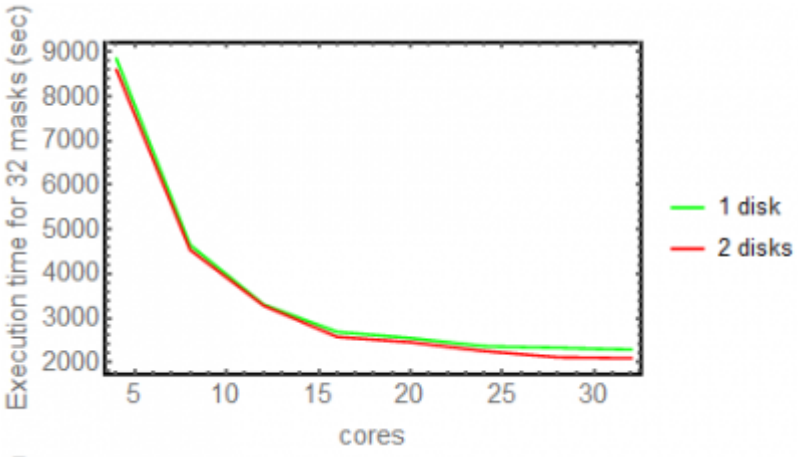
I ran the same tests twice.

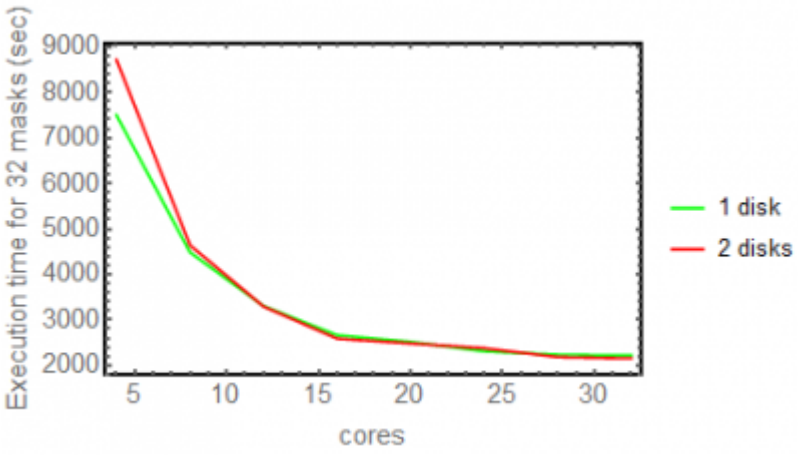
I created rfi masks running rfifind in parallel for 4,8,12,16,20,24,28 and 32 cores (>16 hyperthreaded).
In the following plots I plot the number of parallel instances of rfifind executed (x-axis) and the time taken for these to be completed (y-axis).





In the following plots, I extrapolated the above results in order to find the optimal number of parallel jobs in order to compute 32 rfi masks





From the above, we can conclude that using 1 or 2 disks does not make a big difference. Also, hyperthreading works smoothly, and indeed the best strategy is to have the maximum possible number of rfifind instances running in parallel .