ctools - Bug #2893

Response cache seems to use a lot of memory

05/27/2019 10:32 AM - Knödlseder Jürgen

Status:	Closed	**Start date:**	05/27/2019
Priority:	Normal	**Due date:**	
Assigned To:	Knödlseder Jürgen	**% Done:**	100%
Category:		**Estimated time:**	0.00 hour
Target version:	1.7.0		

Description

There seems to be an issue with the response cache using a lot of memory in ctobssim.

History

#1 - 05/27/2019 03:14 PM - Knödlseder Jürgen

- Priority changed from Immediate to Normal

- Target version changed from 1.6.0 to 1.7.0

Postpone to next release as the change would be very disruptive at this moment.

#2 - 07/23/2019 01:05 AM - Knödlseder Jürgen

It would be good to have a precision description of the problem.

#3 - 10/31/2019 02:00 PM - Knödlseder Jürgen

Still looking for any concrete examples where memory usage is a problem to understand what is going on (and if there actually is an issue). Contributions welcome.

#4 - 11/05/2019 09:53 AM - Knödlseder Jürgen

- Status changed from New to In Progress

Luigi sent me an example from the GPS simulation paper. I created a local folder with the code on dirac using:

```
$ mkdir gps/memory-issue/vela
$ git clone https://github.com/cta-observatory/cta-gps-simulation-paper.git
$ cd cta-gps-simulation-paper
$ git reset --hard 3c61a02c7d26755d26cf82c1236b38f3cf5fa265
HEAD is now at 3c61a02 modified the .xml and .fits name with the same formalism
```

#5 - 11/06/2019 04:32 PM - Knödlseder Jürgen

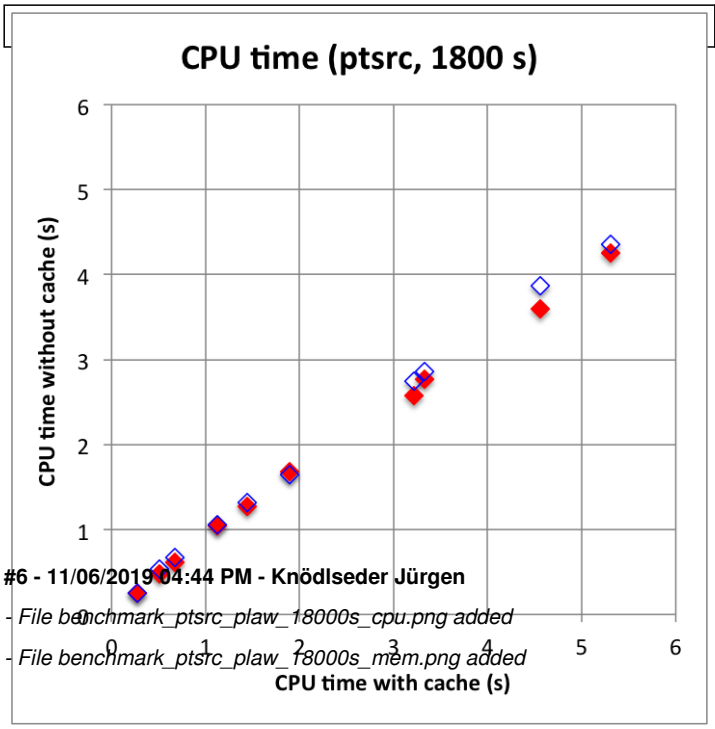
- File benchmark_ptsrc_plaw_1800s_cpu.png added

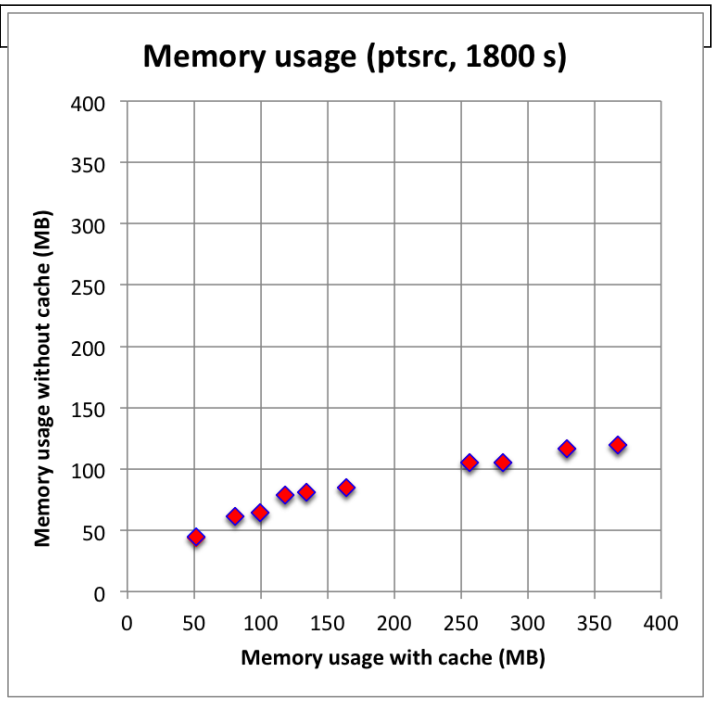
- File benchmark_ptsrc_plaw_1800s_mem.png added

While waiting for a script to do the Vela simulations I wrote a simple script for benchmarking. The script generates a source model with a number of source components (between 1 and 10) and an IRF background model, simulates a single observation with a specific duration, and fits the model to the events in unbinned mode. By increasing the number of sources from 1 to 10 more and more complex sky regions are simulated. The position of the sources are defined randomly within the ROI that is centred around the Crab position and that has a radius of 3 degrees. The source flux and spectral index of the sources also varies.

I started by trying out point sources.

The plots below show the CPU time consumption for the model fitting (left panel) and the maximum memory size for the simulated and fit (right panel). The x-axis show results obtained with the response cache enabled, results on the y-axis were obtained by disabling the response cache. There are in fact two response caches implemented, one for the IRF component and one for the NROI component. The red filled diamonds are for both caches disables, while the blue diamonds are for only the IRF component disabled but with the NROI component enabled. The plots clearly show that the computations are faster and use less memory without a response cache, the response cache slows done a bit the computations, and uses considerably more memory. This indicates that using a response cache for point sources is probably not a good idea. The plot also shows that the IRF component of the cache uses most of the additional CPU time and memory, the NROI component seems to have no significant impact on the performances.



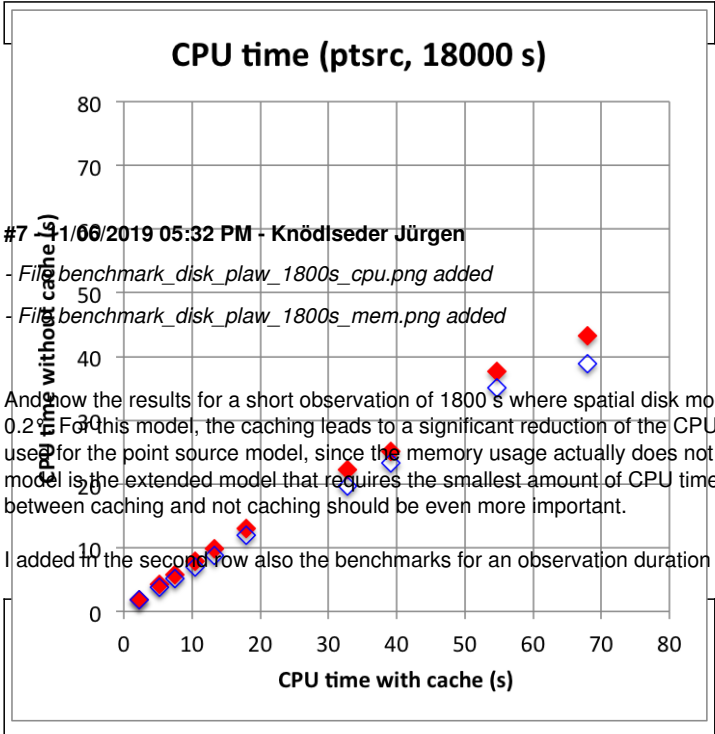


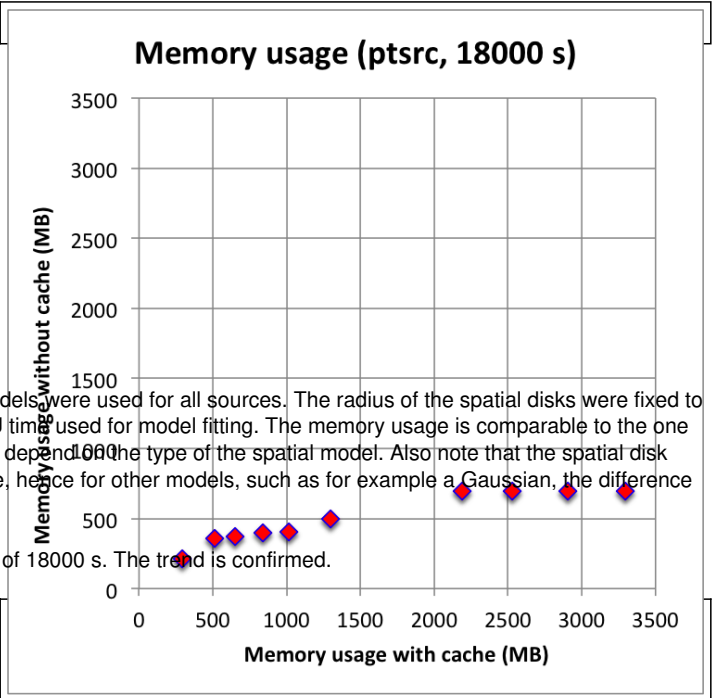
#6 - 11/06/2019 04:44 PM - Knödlseder Jürgen

- File benchmark_ptsrc_plaw_18000s_cpu.png added

- File benchmark_ptsrc_plaw_18000s_mem.png added

Long observation durations confirm the trends: computations without cache, and in particular without the IRF cache component, are faster, and use substantially less memory. For these longer durations, using the NROI cache speeds things a bit up, hence the best combination is no IRF cache component but using a NROI cache component.





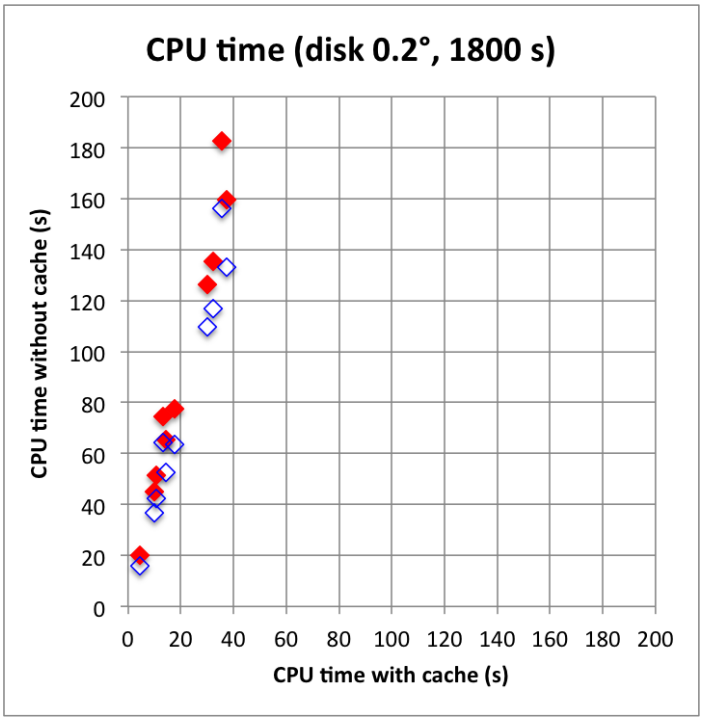
#7 - 11/06/2019 05:32 PM - Knödlseder Jürgen

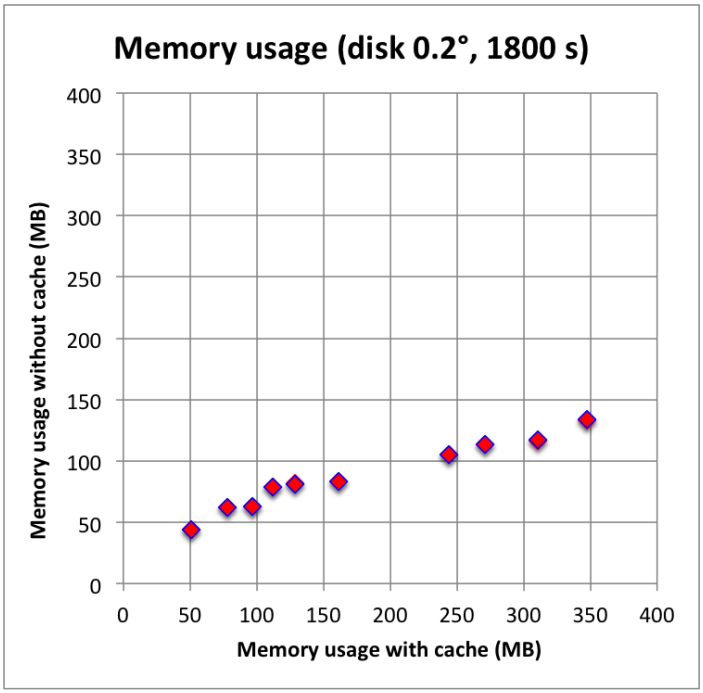
- File benchmark_disk_plaw_1800s_cpu.png added

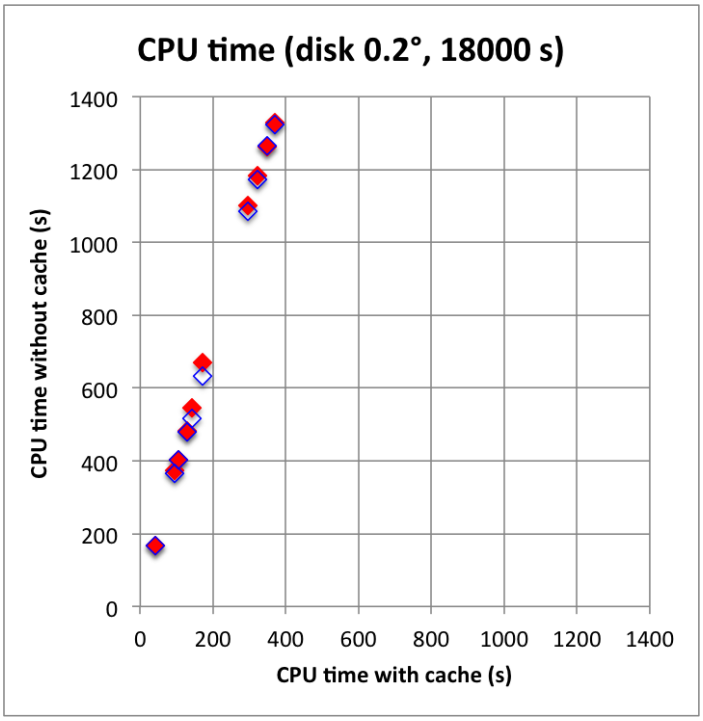
- File benchmark_disk_plaw_1800s_mem.png added

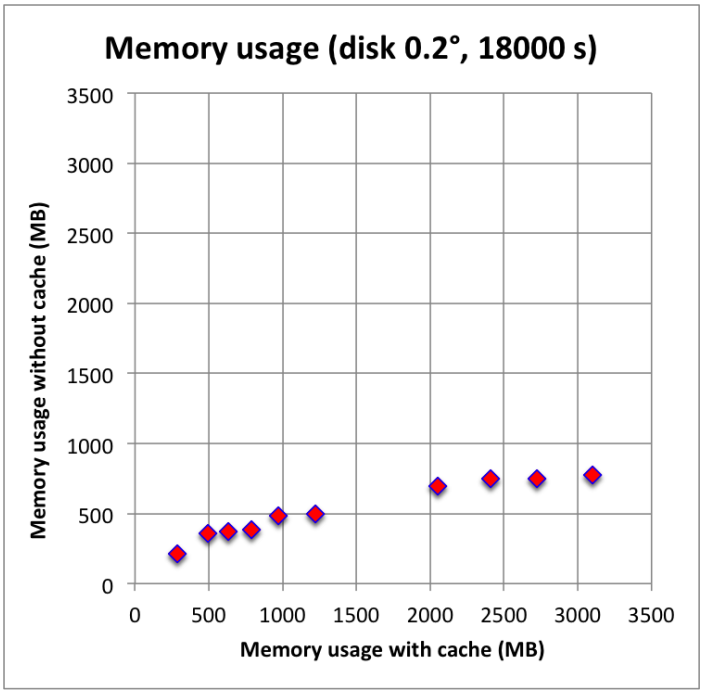
And now the results for a short observation of 1800 s where spatial disk models were used for all sources. The radius of the spatial disks were fixed to 0.2°. For this model, the caching leads to a significant reduction of the CPU time used for model fitting. The memory usage is comparable to the one used for the point source model, since the memory usage actually does not depend on the type of the spatial model. Also note that the spatial disk model is the extended model that requires the smallest amount of CPU time, hence for other models, such as for example a Gaussian, the difference between caching and not caching should be even more important.

I added in the second row also the benchmarks for an observation duration of 18000 s. The trend is confirmed.


CPU time (disk 0.2°, 1800 s)
CPU time without cache (s)
CPU time with cache (s)
0
20
40
60
80
100
120
140
160
180
200


Memory usage (disk 0.2°, 1800 s)
Memory usage without cache (MB)
Memory usage with cache (MB)
0
50
100
150
200
250
300
350
400


CPU time (disk 0.2°, 18000 s)
CPU time without cache (s)
CPU time with cache (s)
0
200
400
600
800
1000
1200
1400


Memory usage (disk 0.2°, 18000 s)
Memory usage without cache (MB)
Memory usage with cache (MB)
0
500
1000
1500
2000
2500
3000
3500

#8 - 11/07/2019 08:59 AM - Knödlseder Jürgen

- File benchmark_disk_plaw_18000s_cpu.png added

- File benchmark_disk_plaw_18000s_mem.png added

#9 - 11/07/2019 11:59 AM - Knödlseder Jürgen

- File new_benchmark_ptsrc_plaw_1800s_cpu.png added

- File new_benchmark_ptsrc_plaw_1800s_mem.png added

- File new_benchmark_ptsrc_plaw_18000s_cpu.png added

- File new_benchmark_ptsrc_plaw_18000s_mem.png added

- File new_benchmark_disk_plaw_1800s_cpu.png added

- File new_benchmark_disk_plaw_1800s_mem.png added

- File new_benchmark_disk_plaw_18000s_cpu.png added

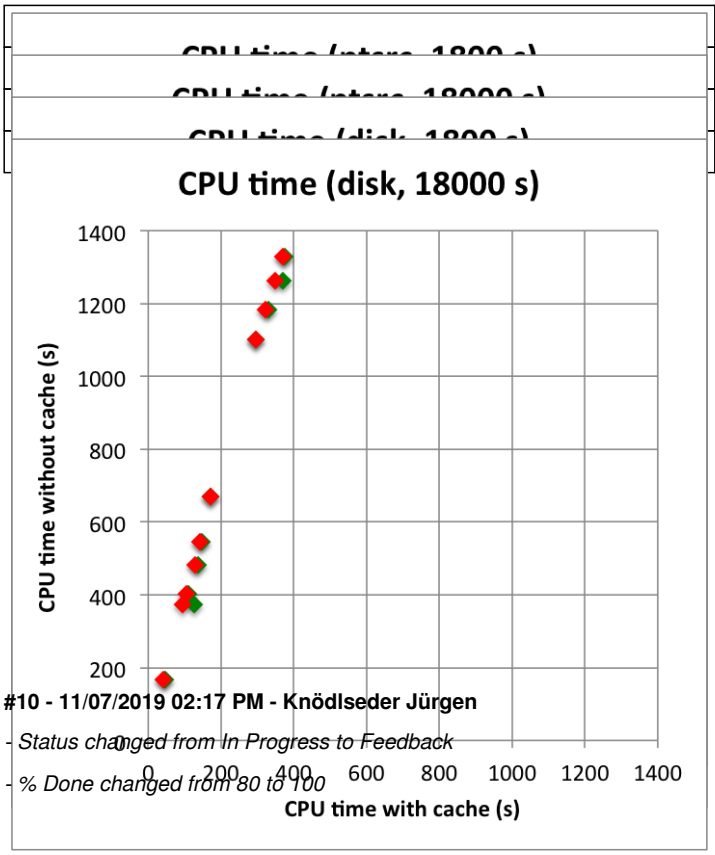
- File new_benchmark_disk_plaw_18000s_mem.png added

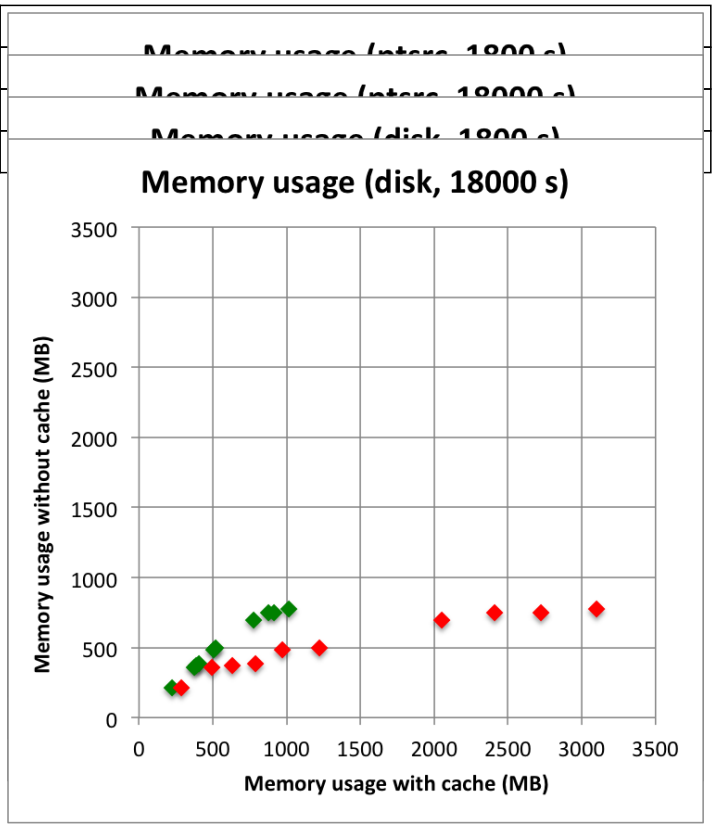
- % Done changed from 0 to 80

I looked into the current implementation of the cache. It turned our that the nesting of the event attributes Right Ascension, Declination, reconstructued and true energy led to quite some memory usage. The managed to reduce the memory usage by encoding these four attributes in a single double precision value. I checked that the fit results were still the same after using the new encoding. I did this check also for a run with the energy dispersion enabled.

Below I reproduced the timing and memory plots that compare the performance with cache (x-axis) to the performance without the cache (y-axis). The red dots are the performance values before changing the code, the greed dots are after removing the nested levels in the cache implementation. For the point source the fit is now done a bit faster, since the removing of the nested levels removes some of the bookkeeping, which apparently led to a speed penalty before. The memory usage (right column) is now drastically reduced.

I merged the code into a new bugfix-1.6.3 branch.





#10 - 11/07/2019 02:17 PM - Knödlseder Jürgen

- Status changed from In Progress to Feedback

- % Done changed from 80 to 100

I also merged the code into devel. Now waiting for feedback.

#11 - 11/08/2019 05:17 PM - Knödlseder Jürgen

- Status changed from Feedback to Closed

Checked one further test case. Also got positive feedback from Josh, close issue now.

Files

benchmark_ptsrc_plaw_1800s_cpu.png	50.9 KB	11/06/2019	Knödlseder Jürgen
benchmark_ptsrc_plaw_1800s_mem.png	58.5 KB	11/06/2019	Knödlseder Jürgen
benchmark_ptsrc_plaw_18000s_cpu.png	56.8 KB	11/06/2019	Knödlseder Jürgen
benchmark_ptsrc_plaw_18000s_mem.png	57.1 KB	11/06/2019	Knödlseder Jürgen
benchmark_disk_plaw_1800s_cpu.png	63.1 KB	11/06/2019	Knödlseder Jürgen
benchmark_disk_plaw_1800s_mem.png	61 KB	11/06/2019	Knödlseder Jürgen
benchmark_disk_plaw_18000s_cpu.png	56.5 KB	11/07/2019	Knödlseder Jürgen
benchmark_disk_plaw_18000s_mem.png	58.8 KB	11/07/2019	Knödlseder Jürgen
new_benchmark_ptsrc_plaw_1800s_cpu.png	49.8 KB	11/07/2019	Knödlseder Jürgen
new_benchmark_ptsrc_plaw_1800s_mem.png	62.1 KB	11/07/2019	Knödlseder Jürgen
new_benchmark_ptsrc_plaw_18000s_cpu.png	54.2 KB	11/07/2019	Knödlseder Jürgen
new_benchmark_ptsrc_plaw_18000s_mem.png	60 KB	11/07/2019	Knödlseder Jürgen
new_benchmark_disk_plaw_1800s_cpu.png	55.9 KB	11/07/2019	Knödlseder Jürgen
new_benchmark_disk_plaw_1800s_mem.png	62.4 KB	11/07/2019	Knödlseder Jürgen
new_benchmark_disk_plaw_18000s_cpu.png	53.3 KB	11/07/2019	Knödlseder Jürgen
new_benchmark_disk_plaw_18000s_mem.png	61.4 KB	11/07/2019	Knödlseder Jürgen