Memory Consumption in Physics Frameworks

Jeff Arnold

openlab / Intel

Overall objective

Investigate memory consumption in LHC physics frameworks

- part of the investigation into the “how to make use of multi-core” issue
- how to avoid “2 GB of memory per core” for **`ncores=8,12,16,...`**

Specific Objective

Study the memory usage of the LHCb reconstruction program Brunel

- use GaudiPython
- using Copy-on-Write

Use of GaudiPython

- It controls the creation and running of the processes which do the analysis
- Processes are created by forking threads from the main parent process
- Copy-on-Write delays creation of process-private pages in child processes
  - pages which are not changed remain shared

Initialization of the framework

Either

- each child process performs its own initialization

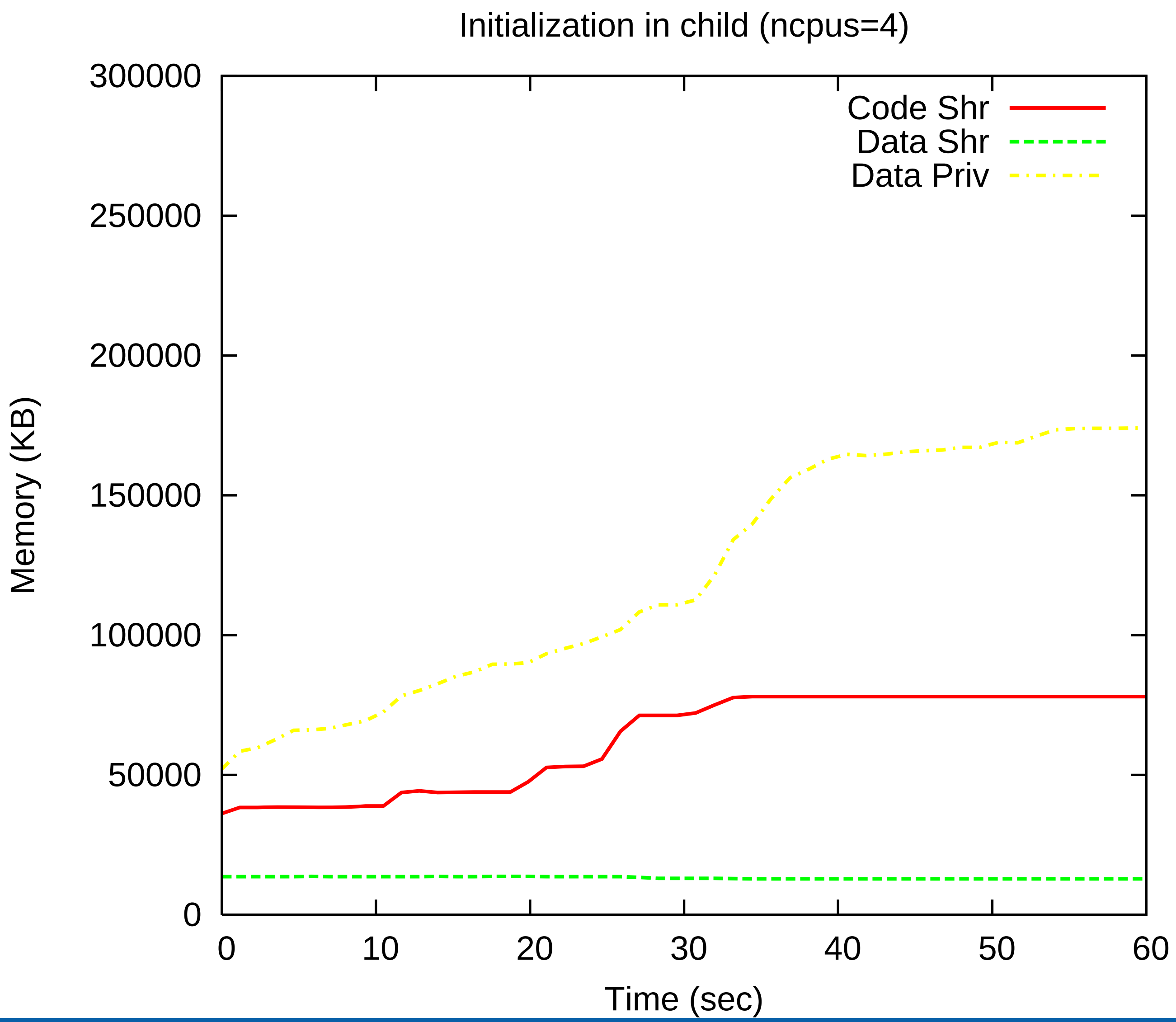
or

- the parent process performs initialization before forking child processes
  - the parent analyzes one event to force complete initialization before forking

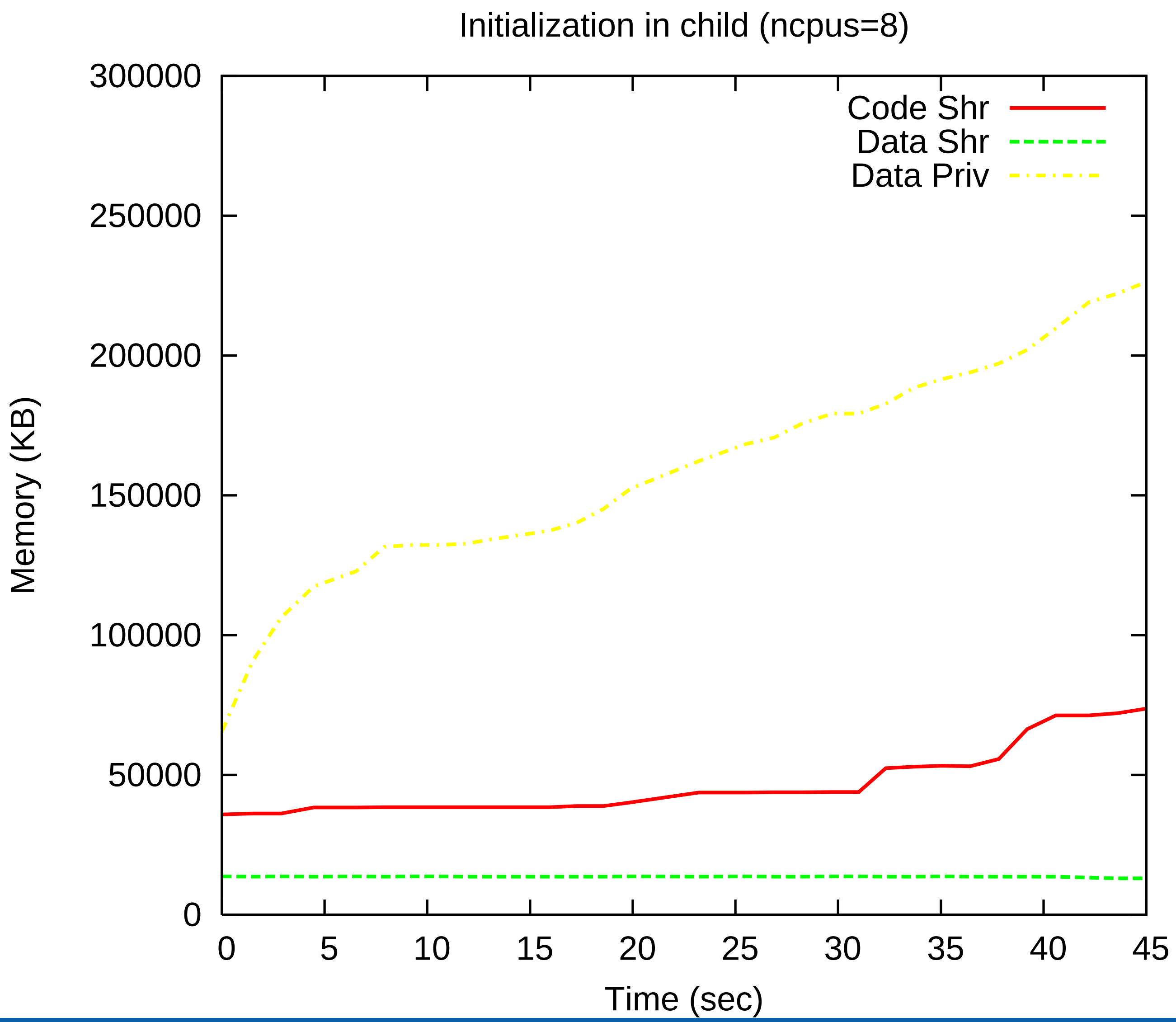
Methodology

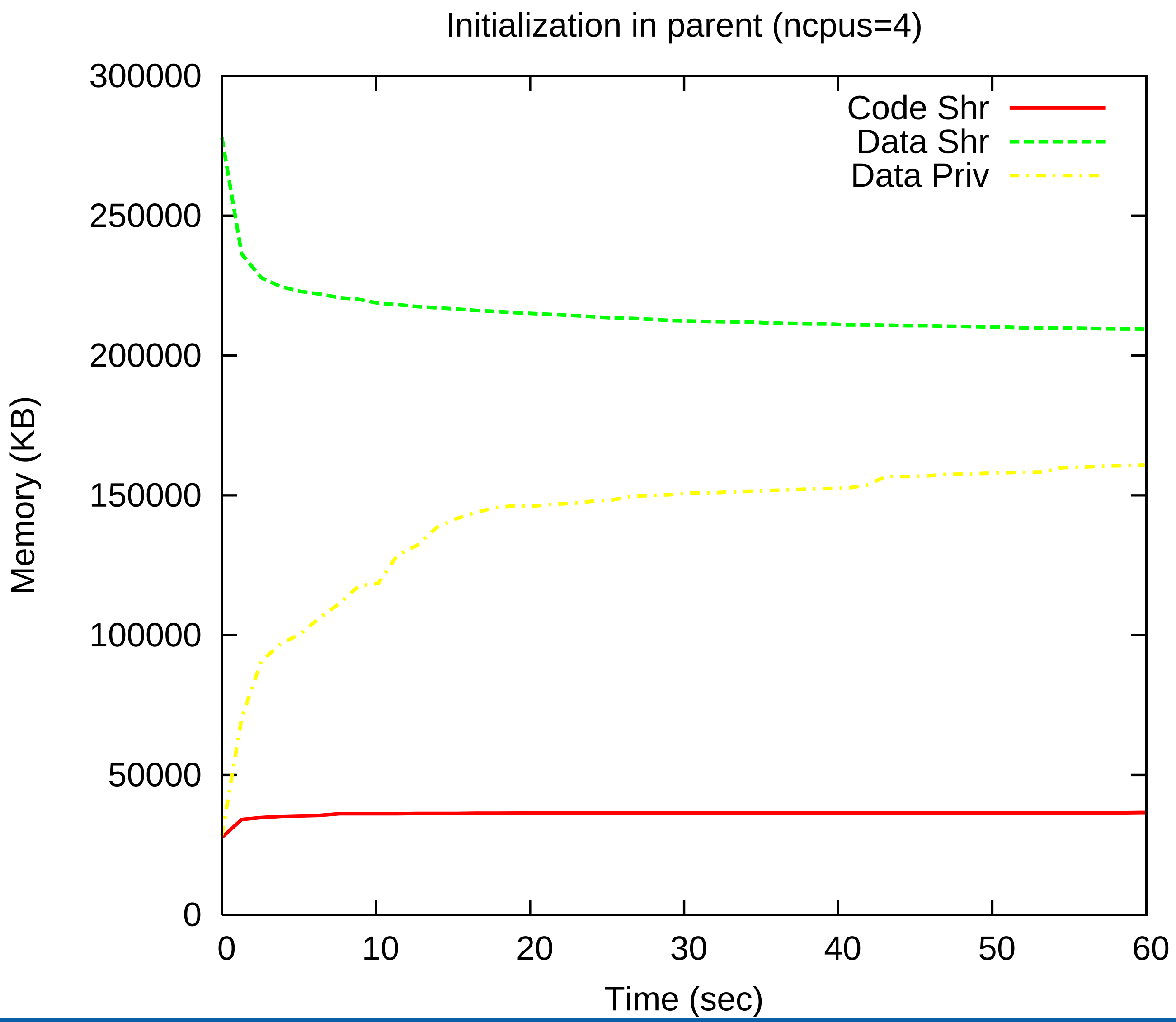
Use **`GaudiPython.Parallel`** to create and manage the child threads

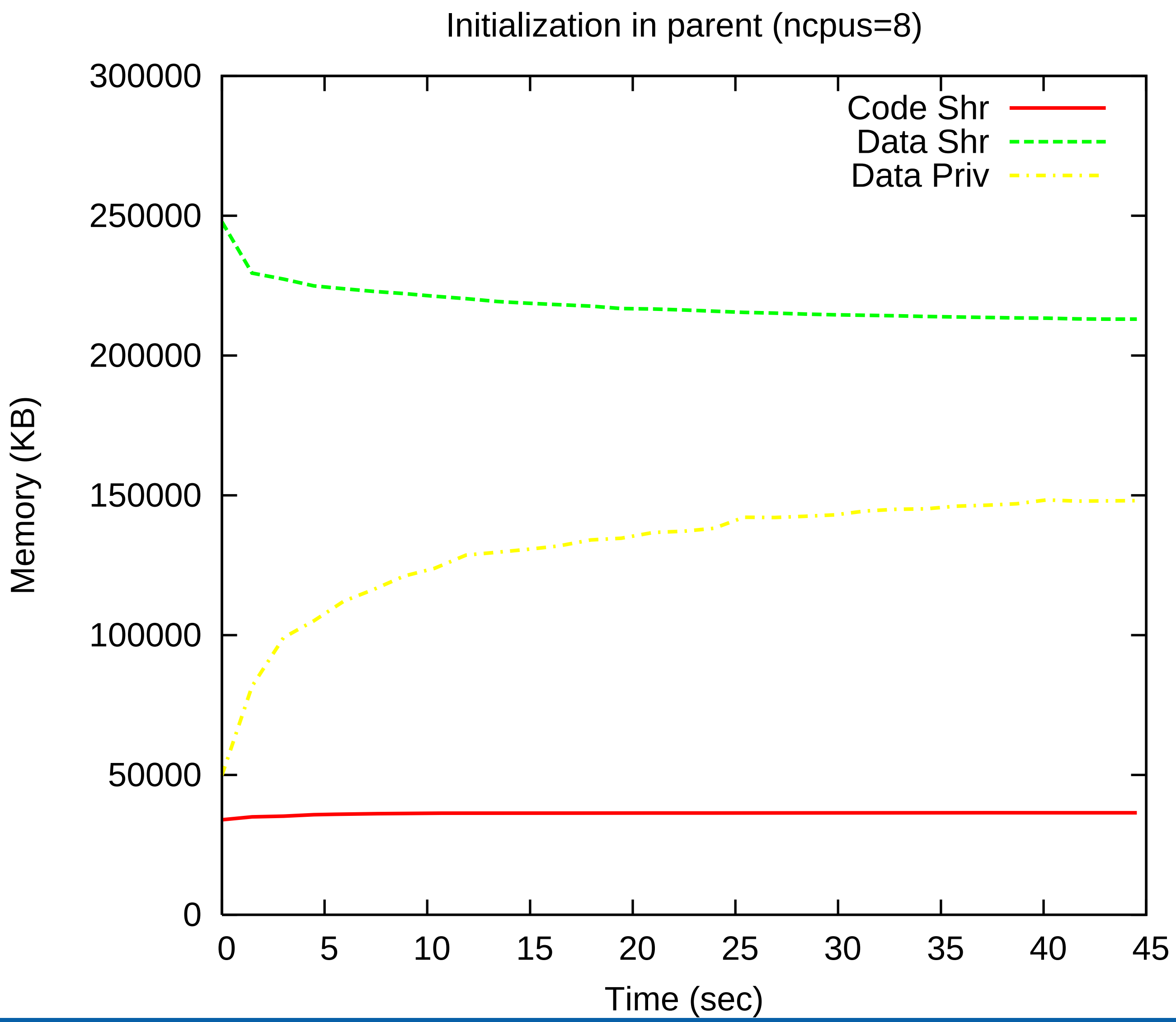
Collect memory information by sampling **`/proc/<pid>/smaps`** every second while threads are running



Initialization in child (ncpus=4)
Memory (KB)
Time (sec)
Code Shr
Data Shr
Data Priv
300000
250000
200000
150000
100000
50000
0
0
10
20
30
40
50
60


Initialization in child (ncpus=8)
Memory (KB)
Time (sec)
Code Shr
Data Shr
Data Priv
0
50000
100000
150000
200000
250000
300000
0
5
10
15
20
25
30
35
40
45


Initialization in parent (ncpus=4)
Memory (KB)
Time (sec)
Code Shr
Data Shr
Data Priv
300000
250000
200000
150000
100000
50000
0
0
10
20
30
40
50
60


Initialization in parent (ncpus=8)
Memory (KB)
Time (sec)
Code Shr
Data Shr
Data Priv
0
50000
100000
150000
200000
250000
300000
0
5
10
15
20
25
30
35
40
45

Observations

- **ncpus=4** and **ncpus=8** are qualitatively the same
- there is much more sharing when the initialization is done in the parent (no surprise)
- Over time:
  - ~13% decrease in shared data
  - ~2.4X increase in private data
- Memory changes appear to be approaching a limit

Observations

- With “child initialization”
  - ~2.9 MB/8 threads
  - 5+ threads/2 GB
- With “parent initialization”
  - ~1.5 MB/8 threads
  - 11+ threads/2 GB

Conclusions

CoW can reduce memory consumption by ~50% when using pre-initialized child threads

No internal application changes required

Furture work

Perform equivalent measurements on other frameworks

Verify asymptotic behavior of memory use