Preparing the Gaudi-Framework and DIRAC-WMS for Multicore Job Submission

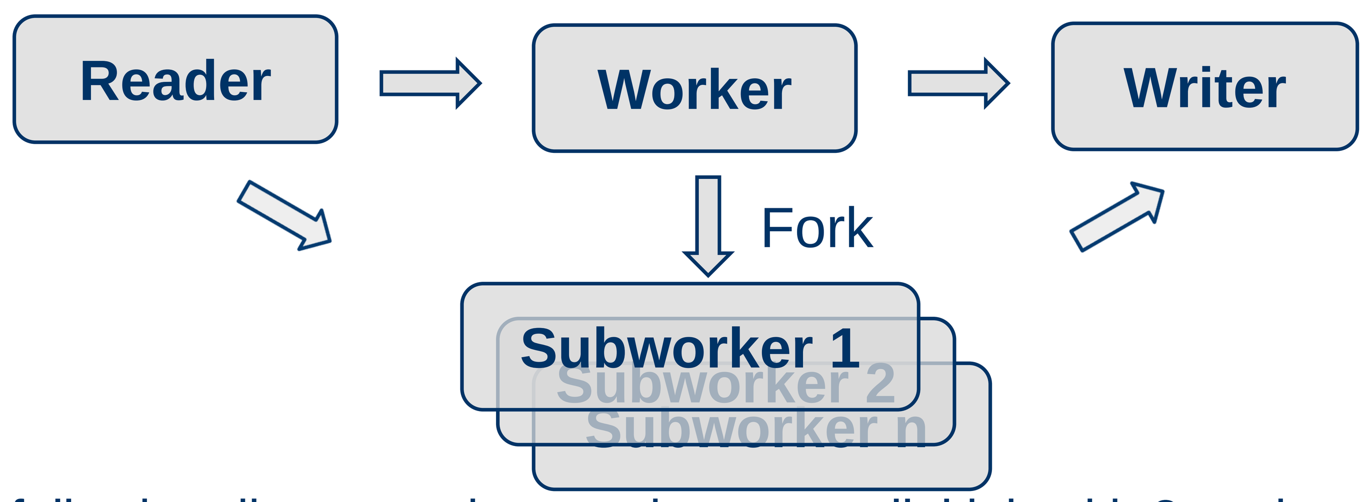
Why parallelization?

State of the art:
- Lower memory ratio on future manycore system
- High memory demand of LHCb applications
- Concurrent access to system resources.

Go Parallel →

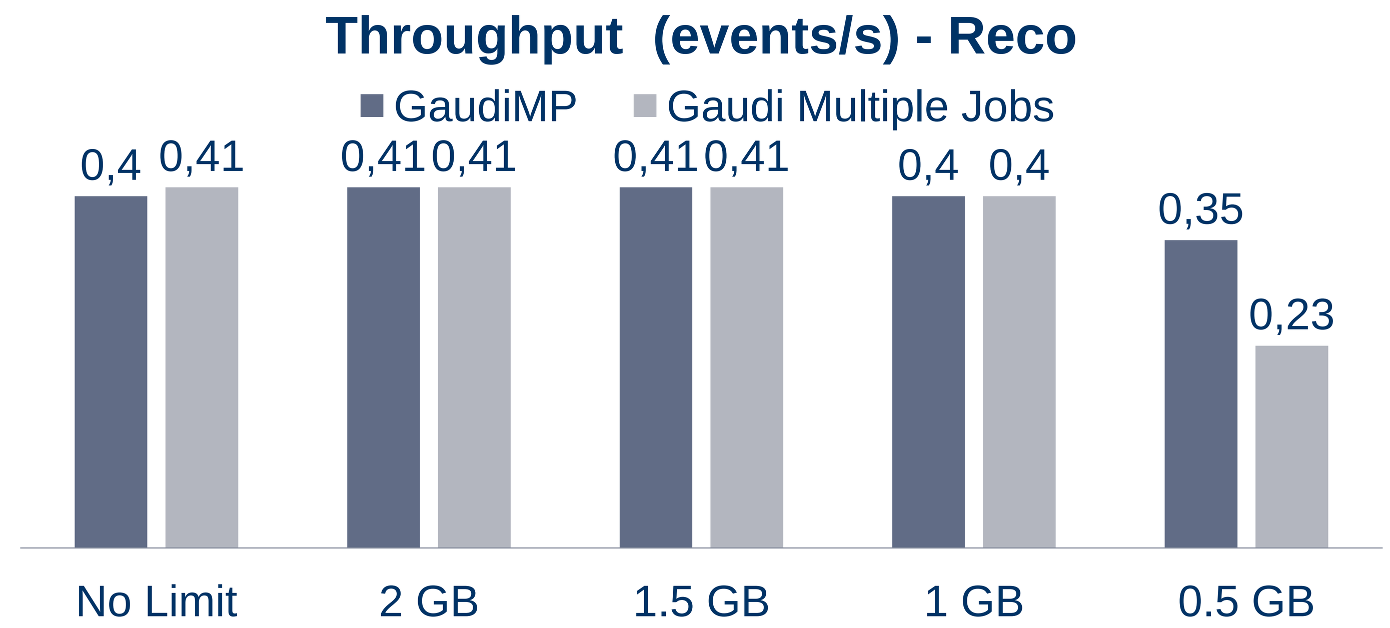
Share datasets and reduce memory
Coordinate access to data, disk and network
Shorten job execution times

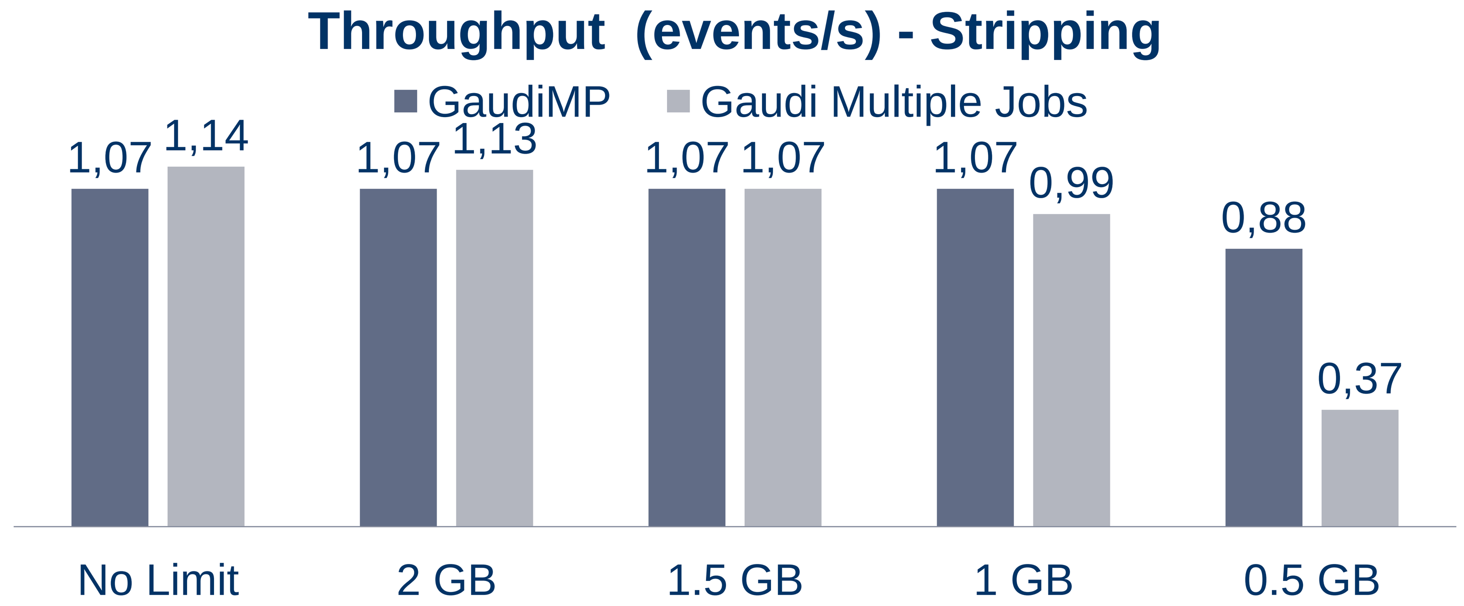
Current GaudiMP implementation



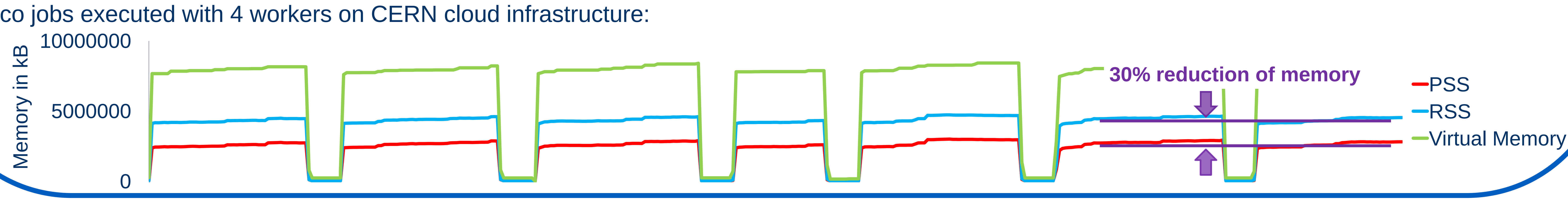
Event parallelism:
- Based on processes due to Python GIL
- Serialization of objects from Transient Event Store (Shared memory cannot be used due to virtual tables)
- Simulation: Reader generates random seeds in order to guarantee reproducibility
- The more events the better the speedup becomes. However, there is an upper limit
- GaudiMP allows a transparent usage of the applications

The following diagrams shows, when a parallel job with 8 workers outperforms multiple single instances, while the memory limit is continuously decreased to 2, 1.5, 1 and 0.5 GB.





Reco jobs executed with 4 workers on CERN cloud infrastructure:



Speedup

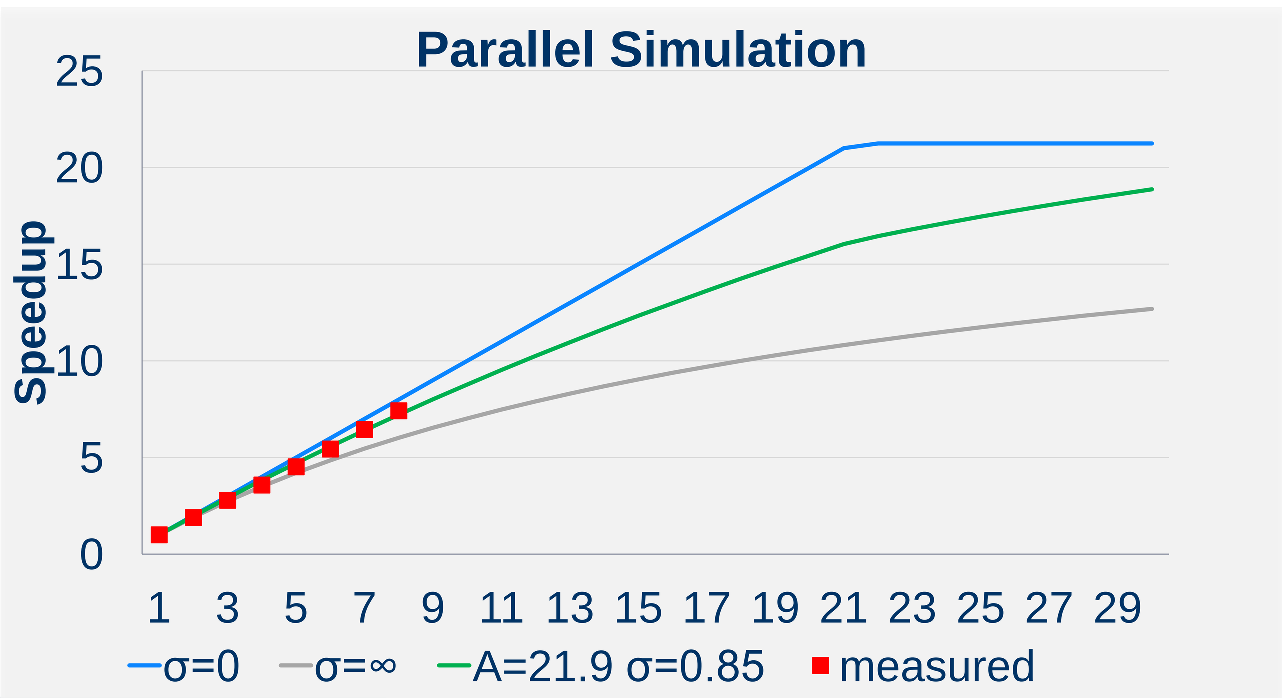
Speedup is limited due to:
- Number of events
- Serialization
- Communication overhead …

The larger the degree of parallelism the better the memory sharing and the worse the speedup. Prediction of speedup via the Downey speedup model:

$$S(n) = \begin{cases} \dfrac{An}{A + \sigma(n-1)/2} & 1 \le n < A \\ \dfrac{An}{\sigma(A - 1/2) + n(1 - \sigma/2)} & A \le n \le 2A - 1 \\ A & n \ge 2A - 1 \end{cases}$$

A = average parallelism
σ = variance in parallelism

Those parameters indicate when cores cannot be used efficiently any longer by an application.



Moldable Job Model

Options:
1. Assign each job the maximum number of available processing units → large loss due to non linear speedup.
2. Assign each job only one processing unit → optimal per job efficiency
3. Assign each job a minimum partition size (defined by memory demand) and mix jobs

Scheduler defines degree of parallelism for each job
- Taking into account characteristics of WNs:
  - Current workload
  - Hardware configuration and CPU architecture
  - Scaling behavior

Aim of a moldable job scheduler:
1. Assign cores such that the least amount of CPU time is wasted due to non linear speedup
2. Modify degree of parallelism of certain jobs if this improves the overall utilization

Future Work

Workload Management System DIRAC must be extended by:
- Self learning algorithms, which gain knowledge about WNs and about scaling behavior of jobs
- Scheduler, which optimizes the overall utilization

The aim is to optimize the LHCb grid jobs in respect to memory and runtime what can be done at the level of:
- Operating system (KSM, x32, auto-vectorization...)
- Software (Late forking …)
- Scheduling (Mix jobs in an appropriate way…)

N. Rauschmayr on behalf of the LHCb Distributed Computing Team