Optimizing memory consumption within GaudiMP

Nathalie Rauschmayr

Computing group of LHCb

February 5, 2013

Outline

Outline

Possibilities

- Operating system:
  - Automatic tools: KSM
  - Compilation: x32-ABI
- Late Forking:
  - Fork child processes after initialize() and before the first event
    - Close and reopen files
    - Reset DB-connections
    - Reinitialize threads
  - Fork after a few events
    - Reset all histograms
    - Reset all counters ...
- Multicore jobsubmission:
  - Mix of memory-bounded and CPU-bounded jobs

Outline

KSM

- Kernel Same Page Merging for automatic memory sharing between parallel processes
- Absolute and relative memory reduction within LHCb applications:

	Gauss	Brunel	DaVinci
serial mode	183 MB (22 %)	100 MB (8 %)	165 MB (13 %)
2 workers	623 MB (33 %)	448 MB (21 %)	890 MB (26 %)
4 workers	1275 MB (42 %)	990 MB (27 %)	1841 MB (29 %)
8 workers	2659 MB (48 %)	2297 MB (33 %)	3864 MB (32 %)

x32-ABI

x32-ABI:

- Application Binary Interface based on 64-bit x86 architecture
- Uses 32-bit pointers instead of 64-bit
- Takes advantage of many x64-features
- Avoids overhead of 64-bit pointers

Reconstruction of 1000 Events:

- Physical memory: - 20 %
- Total time: - 2 %

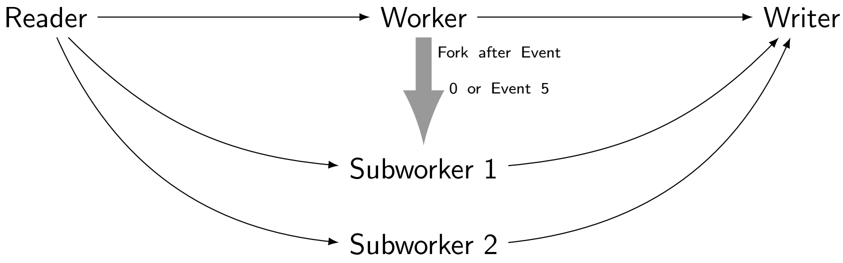
Analysis of 10000 Events:

- Physical memory: - 21 %
- Total time: + 1.5 %

Outline

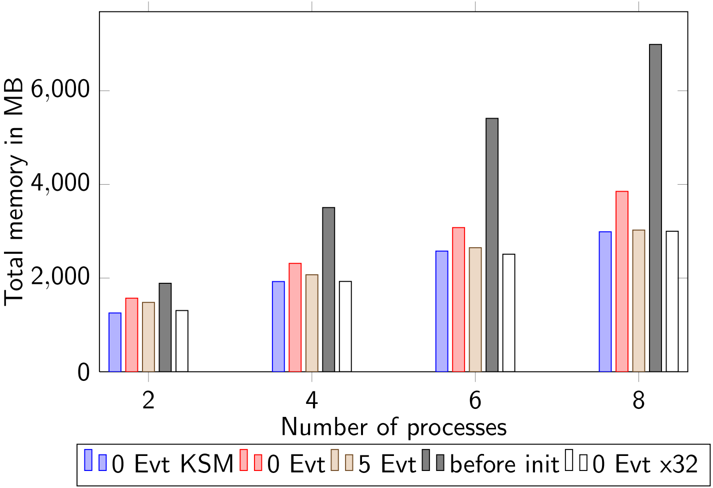
GaudiMP - Implementation Overview

- Main worker forks subworkers
- Main worker sends all necessary information (queues, services, transient event store, etc...)
- As soon as subworkers are spawned queues from subworkers to reader and writer are active



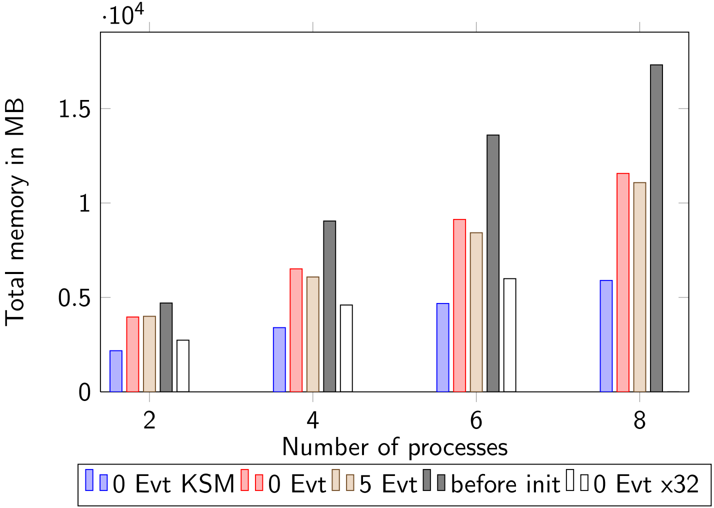
Results within Brunel

Combination of late forking with KSM and x32 opposed to forking before init and after 5 events.



Results within DaVinci

Combination of late forking with KSM and x32 opposed to forking before init and after 5 events.



Current state

Fork new subworkers before the first event:

- Only small modifications necessary
- Open files are handled by reader and writer

Following modifications have been necessary:

- Threads which are initialized before the main loop
- Disconnect from conditions database and close threads
- Reopen and restart threads within each worker process

Outline

Multicore jobsubmission

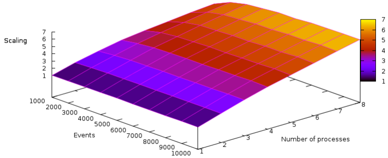
Given: certain number of available cores

How to schedule multicore jobs:

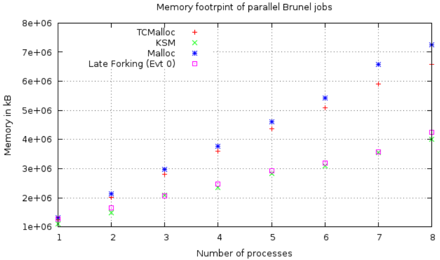
- Such that memory limit is not reached
- n free cores $\rightarrow$ run one job with n processes
- mix of parallel jobs $\rightarrow$ Decision making problem
    - Main limitation is memory
    - Improvement in memory consumption $\rightarrow$ influence on decision making
    - Search of global optimum
    - Fairness

Multicore jobsubmission

Example:



(a) Scalability Graph for Brunel



(b) Maximum of memory consumption

Multicore jobsubmission

Iterative greedy algorithm [1] [2] [3]

- Each job gets at least one core
- Next core is given to the job which will profit the most from it
- Repeated until all cores are assigned

If memory limit is reached:

- Job with lowest memory footprint get all cores
- Give away one core, such that scalability increases but memory limit is still not reached
- Repeated until limit is reached

[1] G. Sabin, M. Lang, P. Sadayappan: Moldable Parallel Job Scheduling Using Job Efficiency: An Iterative Approach

[2] S. Srinivasan, V. Subrammi, R, Kettimuthu, P.Holenarsipur, P. Sadayappan: Effective Selection of Partition Sizes for Moldable Scheduling of Parallel Jobs

[3] S. Kobbe, L. Bauer, D. Lohmann, W. Schröder-Preikschat, J. and Henkel: DistRM: distributed resource management for on-chip many-core systems

Multicore jobsubmission

Caveats:

- Producing scalability graphs is quite time consuming
- Scalability can change (version of a software, compiler flags ...)

Solution:

- Instrumenting production jobs to get information
- Provide training set and improve with recorded output
- Scalability DB
- Prediction with speedup model

Multicore jobsubmission

Downey Speedup Model [4]:

$$S(n) = \begin{cases} \frac{An}{A+\sigma(n-1)/2} & 1 \leq n \leq A \\ \frac{An}{\sigma(A-1/2)+n(1-\sigma/2)} & A \leq n \leq 2A-1 \\ A & n \geq 2A-1 \end{cases}$$

, where A is the average parallelism and σ the variance in parallelism

Estimation of A and σ:

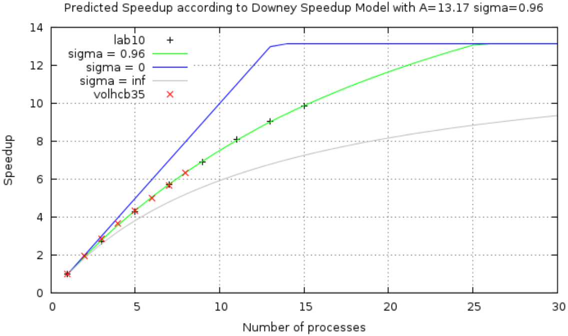
- Given points via measurements $\rightarrow$ curve-fitting
- 2-dimensional minimization problem

[4] Allen B. Downey: A Model for Speedup of Parallel Programs

Multicore jobsubmission

Example:

- volhcb35 with 8 cores and lab10 with 16 cores



(c) Parallel Brunel

Outline

Future Plans

A lot of further investigations necessary:

- Mixing parallel jobs $\rightarrow$ requires modifications on the level of WMS
  - Influence on overall throughput (concrete numbers)
  - Definition of limitations
  - Job duration
- Limitations in the speedup of GaudiMP
- Influence of optimization techniques on overall memory consumption
- Speedup model

Questions?