Optimized Calculation of Timing for Parallel Beam Operation at the FAIR Accelerator Complex

A. Schaller, J. Fitzek - *GSI, Darmstadt, Germany*

Prof. Dr. F. Wolf, Dr. D. Lorenz - *TU Darmstadt, Germany*

Abstract

For the new FAIR accelerator complex at GSI the settings management system LSA is used. It is developed in collaboration with CERN and until now it is executed strictly serial. Nowadays the performance gain of single core processors have nearly stagnated and multicore processors dominate the market. This evolution forces software projects to make use of the parallel hardware to increase their performance. In this thesis LSA is analyzed and parallelized using different parallelization patterns like task and loop parallelization. The most common case of user interaction is to change specific settings so that the accelerator performs at its best. For each changed setting, LSA needs to calculate all child settings of the parameter hierarchy. To maximize the speedup of the calculations, they are also optimized sequentially. The used data structures and algorithms are reviewed to ensure minimal resource usage and maximal compatibility with parallel execution. The overall goal of this thesis is to speed up the calculations so that the results can be shown in an user interface with nearly no noticeable latency.

Motivation

To allow the commissioning and operation of FAIR, the software used today has to be optimized. The Cryring (YR), with its local injector, acts as a test facility for the new control system and in special for the control systems central component, the settings management system LSA. For the last YR commissioning beam time, about 3 700 manual trims were calculated per week with 80 working hours, which is about one trim every 77 seconds. Since the YR is a very small accelerator ring, with a circumference of approximate 54 m, everything worked fine. The waiting time summarizes to about 19 minutes, however, the human reaction time is not much less. But when it comes to calculate the Heavy Ion Synchrotron 18 (SIS18) or SIS100, the calculations get very slow. To calculate 3 700 trims for the SIS18, with its approximate 216 m, an operator would have to wait for over 13 hours. The SIS100, with approximate 1 100 m, calculation would even take over 91 hours.

Speedup

The speedup represents a factor, that shows how two different algorithms perform on the same task. In the context of parallelization, the speedup is a factor that indicates how much the parallel algorithm is faster than the sequential one. It is given by

$$S(P) = \frac{T(1)}{T(P)} \tag{1}$$

where $T(n)$ is the total execution time on a system with n processing units.
$T(1)$ is also representable as

$$T(1) = T_{setup} + T_{compute} + T_{finalize} \tag{2}$$

Since the only part that can benefit from parallel optimization is $T_{compute}$, $T(n)$ can be written as

$$T(n) = T_{setup} + \frac{T_{compute}(1)}{n} + T_{finalize} \tag{3}$$

The efficiency can be expressed as

$$E(P) = \frac{S(P)}{P} \tag{4}$$

where
E = Efficiency
P = Number of Processing Units
S = Speedup
T = Time

Work Depth Model

The Work Depth Model, described by Blelloch, allows to compare the execution time of parallel algorithms. Especially when using trees for parallelization, like the parameter hierarchy in LSA, other comparison mechanism don't fit to the problem. In the context of parallelizing LSA, the work W is expressed by the amount of settings to be calculated and the depth D is expressed by the depth of the parameter hierarchy. Using Equation 5 of Blelloch, a range for time T can be calculated for a given number of processing units P where time T depends on the hardware.

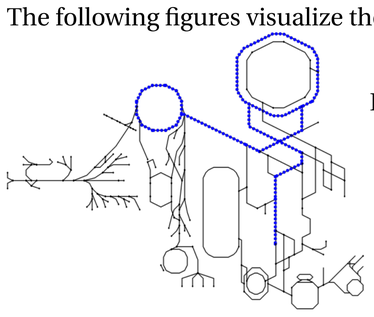
$$\frac{W}{P} \leq T < \frac{W}{P} + D \tag{5}$$

With respect to Equation 1 we can say that the speedup in the Work Depth Model is

$$\frac{WP}{W + PD} \leq S(P) < P \tag{6}$$

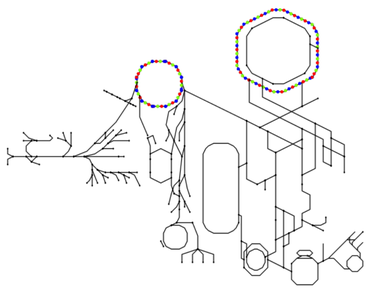
Test Scenarios

The following figures visualize the two patterns used for testing



Pattern 1
- Including one Chain
- Changed one high-level parameter in
  - SIS18 (P1-1)

Pattern 2
- Including three Chains
- Changed one high-level parameter in
  - SIS100 (P2-1)
  - SIS18 (P2-2)
  - SIS18 and SIS100 (P2-3)



Overview of the test scenarios

Scenario	Nr. of Settings	Nr. of changed high level Settings	Nr. of calculated dependent Settings	average original time
P1-1	14 538	1	8 707	12.7 s
P2-1	38 943	1	11 184	132.9 s
P2-2	38 943	1	8 466	18.1 s
P2-3	38 943	2	19 650	155.2 s

Optimizations

Sequential

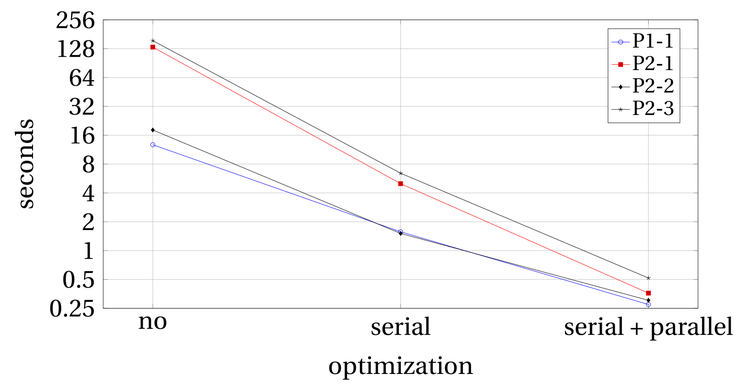
- use caching where possible
- use suitable data structures for the main use case
- reduce array copies when inserting (or deleting) multiple points to a function
- change algorithms with complexity $O(n^2)$ to those with $O(n \log n)$ where possible
- don't calculate a setting for all its parents but only once all its parents have been calculated

Parallel

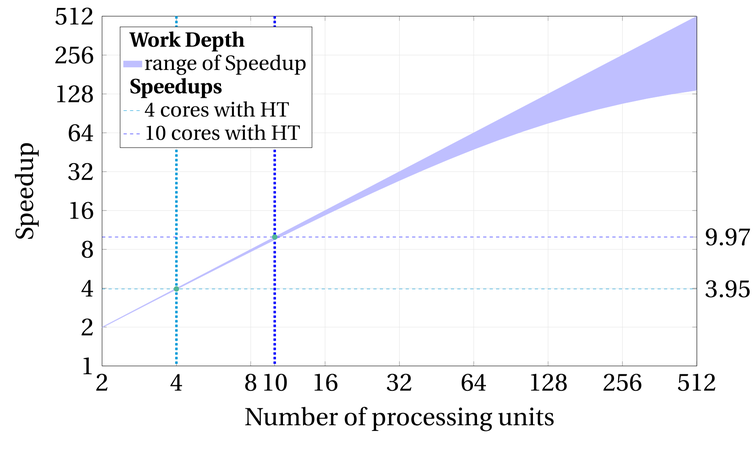
- run static data preparation in parallel
- run calculation loops in parallel where possible
- use parameter hierarchy as a task graph

Optimization Results

Average execution times on the target platform (10 cores with HT, 64 GB RAM) where each scenario was run twice for warmup and five times for measurements. The parallel execution was measured with the default threadpool size of 19 plus the main thread.



Work Depth Model: Equation 6 for $W = 3728$ (changed settings in SIS100) and $D = 20$ (depth of parameter hierarchy for SIS100).



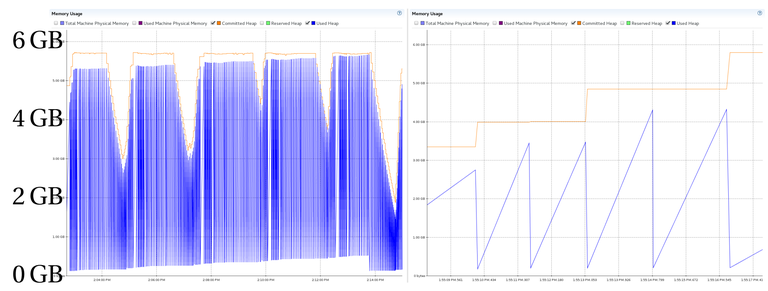
Work Depth Model for

- 4 cores: speedup is between 3.92 and 4.00
  measured speedup is 3.95
- 10 cores: speedup is between 9.49 and 10.00
  measured speedup is 9.97

The parallel speedup on the target platform with 10 cores has an efficiency E of 0.997, on the test platform with 4 cores the efficiency is 0.987 (see Equation 4), which nearly is a so called *perfect linear speedup* where $E = 1$.

The following image shows the memory consumption on the target platform for scenario P2-3. The scenario was run twice for warmup and five times for the measurements.

no optimizations	vs.	serial + parallel optimizations

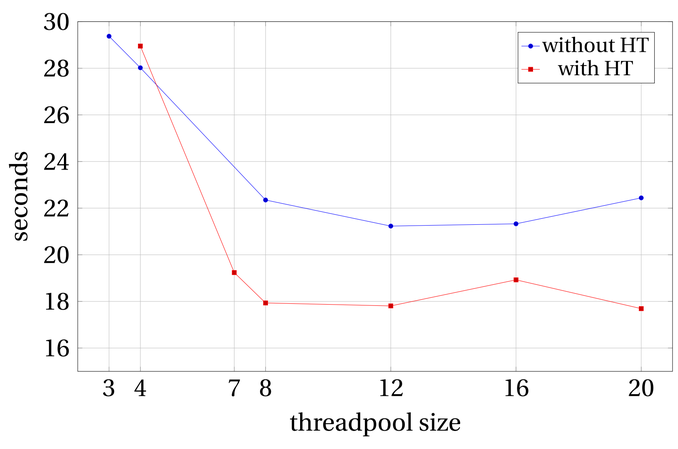


738.6 s	vs.	8.7 s
305 garbage collections	vs.	5 garbage collections
1 353.4 GB total memory allocation	vs.	18.9 GB total memory allocation

Overview of the needed resources for P2-3

optimization	none	sequential	sequential & parallel
trim time	95.6 s	5.4 s	0.7 s
CPU usage avg	6.2 %	9.1 %	62.8 %
CPU usage max	19.9 %	11.1 %	62.8 %
Heap avg	2.5 GB	1.8 GB	2.0 GB
Heap max	5.4 GB	3.4 GB	3.8 GB
GC pause time avg	11.4 ms	29.7 ms	33.8 ms
GC pause time max	70.5 ms	29.7 ms	33.8 ms
Main thread usage	100 %	100 %	8.5 %
Thread pool usage total	0 %	0 %	91.5 %
Thread pool usage avg	0 %	0 %	4.8 %
TLAB size total	147.5 GB	6.4 GB	4.0 GB
TLAB size avg	64.1 MB	56.8 MB	3.6 MB
TLAB size max	105.7 MB	75.0 MB	72.8 MB
Alloc. rate for TLAB	1 495.0 MB/s	731.3 MB/s	672.0 MB/s
Object size total	114.3 kB	10.4 kB	5.0 kB
Object size avg	1.0 kB	10.4 kB	0.8 kB
Object size max	32.0 kB	10.4 kB	2.1 kB
Alloc. rate for Objects	1.1 kB/s	1.1 kB/s	0.8 kB/s

For this chart, each scenario was executed 2 times for warmup and 5 times for measurements on a test platform (4 cores, 12 GB RAM) wth and without Hyper Threading Technology (HT). The default threadpool size is $n-1$, so for the setup without HT, the default threadpool size is 3 and with HT the default threadpool size is 7.



Summary

By reducing the memory consumption and complexity of the most used algorithms of LSA from $O(n^2)$ to $O(n \log n)$ the sequential calculation time could be sped up by 22.00. Parallelizing the DAG containing the parameter hierarchy and some loops in the trim calculations increased the speedup by a factor of 9.97 on the target platform with 10 cores with hyper threading. This leads to an average speedup of 219.23 which now allows the user to seamlessly change the overall accelerator scheduling.

presented at ICALEPCS 2017