DC Optimization Service Work

Responsible: Bruno Benkel - Esteban Molina - Antonio Radic
Contacts: Raffaella De Vita - Veronique Ziegler

2022-06-22

Context

- CLAS12 offline reconstruction time is good, but can be improved.
- As seen below, DC reconstruction is the biggest consumer of CPU time.
- Our work plan consists of three areas of work: Refactoring the code, comparing Java releases, and applying algorithmic improvements.

Method	Time (%)
`RungeKuttaDoca.RK4transport()`	29.8%
`StandardSolenoidProbe.field()`	7.9%
`TorusProbe.fieldCylindrical()`	7.4%
`ClusCleanerUtils.ClusterSplitter()`	7.0%
`StateVecsDoca.transport()`	6.8%
`Matrix5x5.add()`	4.9%
...	...

Table: Sampling result for 10.000 data events.

Context

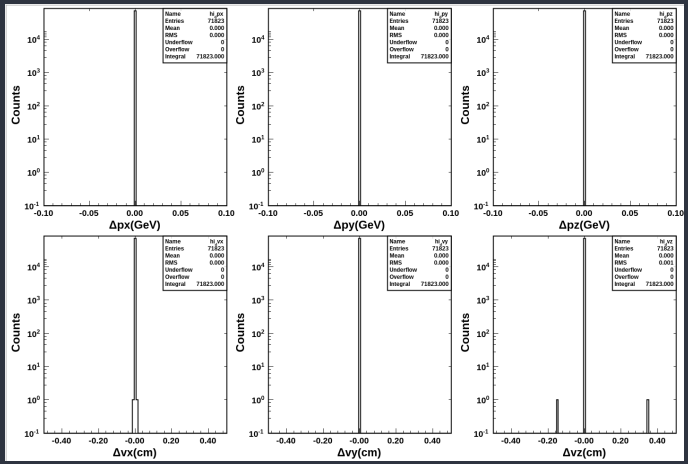
- Our aim is to improve reconstruction speed without damaging its results by a significant margin ($\ll$ than resolution).
- David Heddle is currently working on magfield optimization, so we'll just focus on DC.
- All profiling is done on `jvisualvm` running over `Graalvm`.
- We routinely do error checking by comparing HIPO banks and plotting differences.
- For profiling and error checking, we're using a production data file from RGA Fall 2018 and a 10k events SIDIS sample.

Refactoring the Code

- To refactor is to improve the design of code without changing its behavior.

- First pull request is already out, with a 5.74% reduction in reconstruction time.

- There is a minimal loss in precision associated to the change: 0.4% of events show a difference greater than the chosen threshold.



Refactoring the Code

Before refactoring:

```
swimmer.Bfield(sector, x0+0.5*h*x1, y0+0.5*h*y1, z0+0.5*h, _b);
double x2 = tx0+0.5*h*tx1;
double y2 = ty0+0.5*h*ty1;
double tx2=q*v*Ax((tx0+0.5*h*tx1), (ty0+0.5*h*ty1), _b[0], _b[1], _b[2]);
double ty2=q*v*Ay((tx0+0.5*h*tx1), (ty0+0.5*h*ty1), _b[0], _b[1], _b[2]);
```

After refactoring:

```
swim.Bfield(sector, x0+0.5*h*x1, y0+0.5*h*y1, z0+0.5*h, _b);
double x2   = tx0 + 0.5*h*tx1;
double y2   = ty0 + 0.5*h*ty1;
double tx2  = qv * Ax2(C2, x2, y2);
double ty2  = qv * Ay2(C2, x2, y2);
```

Refactoring the Code

The pull request is mainly focused on `RK4Transport()`, since its the main consumer of computing time.

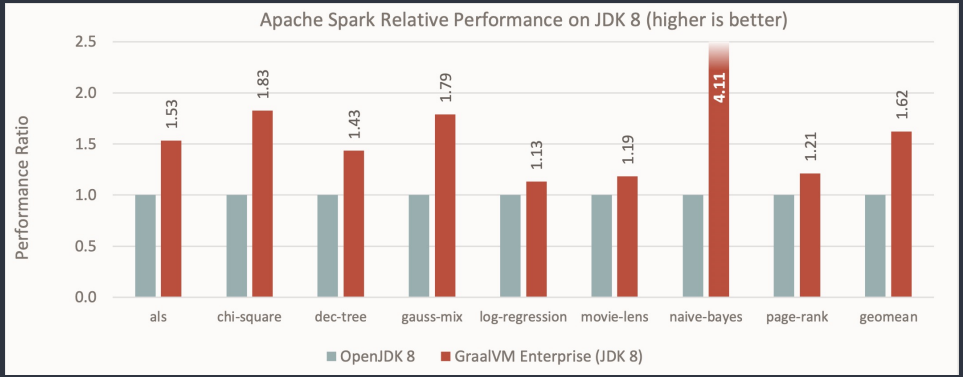
The updated sampling result is:

Method	Prev. Time (%)	New Time (%)
`RungeKuttaDoca.computeCovMat()`	—	18.1%
`RungeKuttaDoca.RK4transport()`	29.8%	10.6%
`StandardSolenoidProbe.field()`	7.9%	8.6%
`TorusProbe.fieldCylindrical()`	7.4%	7.8%
`ClusCleanerUtils.ClusterSplitter()`	7.0%	7.7%
`StateVecsDoca.transport()`	6.8%	1.0%
`Matrix5x5.add()`	4.9%	5.2%
...	...	...

Table: Sampling result for 10.000 data events.

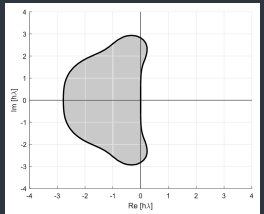
Comparing Java Releases

- Task originally consisted of studying an upgrade from Java 11 to 17.
- The upgrade was trivial to implement, but we found no improvement in computing speed.
- Task has been updated to looking into settings and tools related to Java 17 and `Graalvm`, a performance-oriented Java environment.

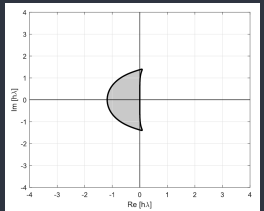


Applying Algorithmic Improvements

- Task consists on changing and/or improving the algorithms implemented in DC reconstruction.
- Insofar, focus has been oriented on switching Runge Kutta 4 (RK4) to Adams-Bashforth-Moulton 4 (ABM4).
- Both methods have the same numerical precision, but ABM4 has less stability – making the switch nontrivial.
- Task is now focused on studying the stability of the methods for the system of equations to improve the choice of step size for both RK4 and ABM4.



(c) Runge-Kutta 4



(c) Adams-Moulton 6

Future Work

Coming work is focused on five areas:

- Work on CVT refactoring and optimization (Bruno).
- Figure out if we can reduce the matrix access time (Bruno).
- Improve tracking parameters tuning via analyzing numerical estimation methods stability (Bruno & Esteban).
- Get ABM4 running and compare it against RK4 (Esteban).
- Figure out how to get `Graalvm`'s Native Compile tool working and measure how much it improves reconstruction speed (Antonio).

Repositories

Fork of the reconstruction software:

`https://github.com/bleaktwig/clas12-offline-software`

Utilities:

`https://github.com/bleaktwig/dc-optimization-utils`