Performance monitoring and tuning in OpenDA

Nils van Velzen

November 30, 2012

1 Introduction

Data assimilation and model calibration techniques can be very computationally intensive. A good performance of your OpenDA configuration is therefore often important. OpenDA contains a number of options that can help you to monitor and tune/improve the performance of your configuration. The currently available tools will be presented in this document.

2 monitoring

A popular Dutch (scientific) saying is "meten = weten", which can be translated into "to measure = to know". The idea is simple, if you measure you will know what is going on and that allows you to improve/change a process when needed. For improving and tuning the performance this counts. Before you can start to improve the performance in a sensible way you need to know the performance of the individual parts.

There are many (Java) profiling tools available to measure the performance of your code. Unfortunately these tools will give you information on the time spend in various routines. These timings are useful for programmers with detailed knowledge of the implementation but it is not practical for a general user. If you profile your configuration you want to have information like the time spend in model steps, getting the state vector, interpolating the observations, multiplication with the gain matrix etc. To provide this type of timing information we have introduced the OdaTiming class in OpenDa.

Programmers can put these timers in the code to time interesting well defined steps in the algorithm. The timers from OdaTiming have labels and there is a way to define sub timers, all to present the result in a human understandable form.

The internal timing mechanism of OpenDA can be switched on and off in the main configuration file of your OpenDA configuration.

```
<timingSettings doTiming="true"></timingSettings>
```

A timing report file will be created at the end of your run when timing in switched on.

The timers are introduced by the programmers. Therefore not all parts of OpenDA are currently timed. At this moment timers are present in:

- Ensemble Kalman Filter.
- Blackbox model.
- rmiModel and threadModel.

At some points programmers should/can extend the timing for other algorithms and parts of OpenDA as well.

3 Precision

The choice of the precision of your computations is an important issue. The most used precisions in scientific computing are double precision (double) and single precision (float). Double precision is sometimes necessary for sufficient accuracy or numerical stability. From the point of performance, single precision is preferable when possible because it halves the storage and communication volume and computations are faster as well.

We have introduced some control over precision in OpenDA. Note the word "some" since OpenDA cannot force the precision of values stored in the user provided classes since it is an object oriented system. In the central input file you can specify

```
<optimizationSettings productionRun="false" VectorPrecisionIsFloat="true">
</optimizationSettings>
```

The option VectorPrecisionIsFloat="true" will mark the wish of the user to use single precision. Programmers of OpenDA classes can introduce dual

precision support by checking the user selected precision in the global static class OdaGlobSettings.

Currently the standard (java) vector implementation in OpenDA (Vector) is implemented in dual precision. Since many classes like the black box model make use of the standard Vector implementation it already works in many configurations.

Note: you can always use the available timing in OpenDA to spot the impact of the "VectorPrecisionIsFloat" option. If you do not see any improvement in linear algebra computations it is likely that the class used in your implementation ignores the user precision preference. In that case you can introduce it yourself or try to convince the programmer of that class to do it for you.

4 Production runs

Data assimilation methods sometimes produce some diagnostics. The amount of time to compute these diagnostics can be quite large. At this moment an algorithms cannot ask a result writer in advance whether data will be written or not. As a result all diagnostics will always be computed whether the user wants them or not. The user can set the options "productionRun" in the global class OdaGlobSettings by specifying it in the central input file.

```
<optimizationSettings productionRun="true" VectorPrecisionIsFloat="true">
</optimizationSettings>
```

Programmers of the assimilation methods can use this user provided option to skip some diagnostics part of the computations.

Note: Currently, this option only impacts the behaviour of the Ensemble based methods in OpenDA.