Parallel in ADMB via pthread and in RcppAD via openMP

Anders Nielsen

September 21, 2013

Background

Enabling AD Model Builder to benefit from multiple cores in modern computers has been flagged as one of the highest priorities for the ADMB foundation. It is important simply because ADMB will not continue to be the fastest general purpose optimizer otherwise.

At the Seattle meeting and in the following months a great effort was made and some examples were produced, where ADMB computations were made parallel using the pthread technology. In order for this to be possible many internal changes had to be made.

The automatic differentiation tool RcppAD presented by invited expert Kasper Kristensen also supports parallel processing, but uses the openMP technology.

The following documents the experiences maed during the developers meeting.

The pthread setup in ADMB

To use pthreads in ADMB a the threaded branch of ADMB must be cheked out. The name and location of the branch is:

`svn+ssh://www.admb-project.org/branches/threaded2`

On a vanilla linux system it did not work in the first go. Two things were missing. 1) In the threaded branch the compiler from template to c++ code, the file `tpl2cpp` was missing, so that was copied from an existing standard ADMB installation. 2) Also the threaded approach requires the `gcc-multilib` to be installed, so that was installed.

In addition the a Makefiles in the examples had hard coded paths to compilers, which had to be changed.

The openMP setup in RcppAD

The `RcppAD` is available from `https://github.com/kaskr/adcomp` and installed by typing `make install`. The package includes the tools needed to parallel runs via openMP.

The Multisimple example

The multisimple example is also described in Fournier & Sibert (2013) is a linear regression example with 10^6 simulated pairs of (x,Y). The data are simulated such that the slope is 2, the intercept is 4 and the standard deviation is 7. The x-points are uniformly distributed between 0 and 100.

The pthread approach

The likelihood used in AD model builder is the so-called concentrated likelihood, where the maximum likelihood estimate for the variance is plugged-in. The function to minimize w.r.t. α and β is:

$$\frac{N}{2}\log\left(\sum(Y_i-\alpha x_i-\beta)^2\right)$$

Besides data and parameter section the template file specifying this model is simply:

```
  f=nobs/2.*log(norm2(Y-a*x-b));
```

The multi-threaded version is a bit more cumbersome. It is more than 200 lines, and those are mainly fairly complicated calls to the pthread_manager. Before the minimization starts the relevant chunks of data has to be sent to the different cores and received by the cores in the same order. When the minimization starts, every iteration starts by sending the new parameter values to each core, and have each core receive it, then the subset of the calculation is done in each core, and finally the results are send back combined. Notice that data is only send to the cores ones.

The part of the calculation which is done in parallel is the sum of squares SS. The final calculation of $\frac{N}{2}\log(SS)$ is done after all terms are collected.

The openMP approach

To use the parallel implementation in RcppAD it is necessary to use the full likelihood, so the function to minimize w.r.t. α, β, and σ is:

$$\frac{N}{2}\log(2\pi\sigma^2)+\frac{\sum(Y_i-\alpha x_i-\beta)^2}{2\sigma^2}$$

The model implemented in RcppAD to use only a single core is:

```
#include <RcppAD.hpp>
template<class Type>
Type objective_function<Type>::operator() ()
{
  DATA_VECTOR(Y);
  DATA_VECTOR(x);
  PARAMETER(a);
  PARAMETER(b);
  PARAMETER(logSigma);
  Type nll=0;
  for(int i=0;i<x.size();i++)nll-=dnorm(Y[i],a+b*x[i],exp(logSigma),true);
  return nll;
}
```

To make to invoke parallel processing of this code only one line need to be changed. Instead of declaring `Type nll=0;` the `nll` must be declared of type `parallel_accumulator` the changed line is:

```
  parallel_accumulator<Type> nll(this);
```

To use this simple parallel constructor it is required that the negative log likelihood is computed as a sum of terms. Notice that this requirement prevents us from computing the concentrated likelihood we used in ADMB.

Timings

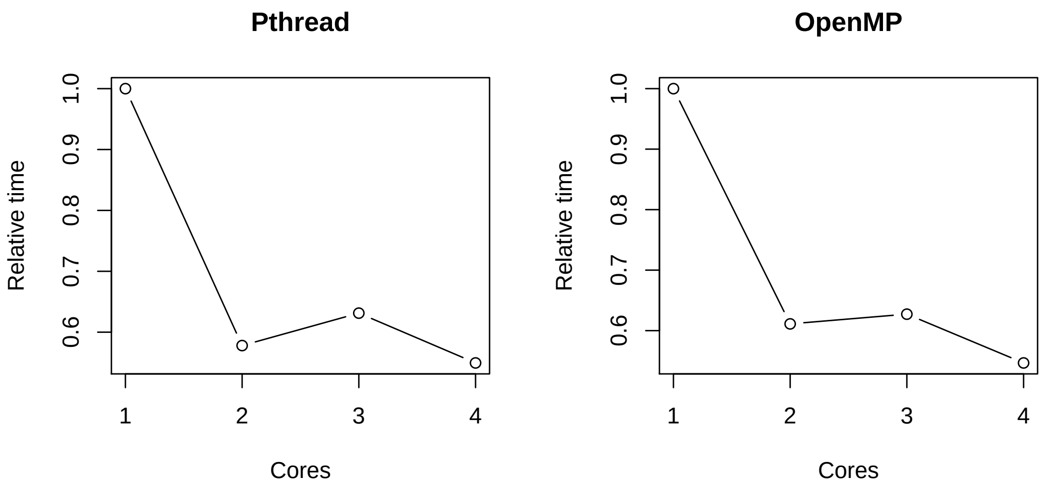
On a simple linux laptop with 4 cores the basic example without any parallel preparation ran in 3.3 seconds in ADMB and 7.4 seconds in RcppAD. Notice that ADMB is twice as fast in this case, but that is because RcppAD is not using the concentrated likelihood. A single core RcppAD version which uses the concentrated likelihood runs in 1.7 seconds, but cannot be improved be parallel computations.

The run times for the parallel versions are:

System	1 core	2 cores	3 cores	4 cores
Pthread (ADMB)	9.4	5.4	5.9	5.2
OpenMP (RcppAD)	8.0	4.9	5.0	4.4

The Pthread parallel setup running on one core increases the run time 2.8 times, so there is an overhead in this setup, which in this simple example is substantial. From one to two cores the run time drops almost to half (56%), but from 2 to 4 core there is little improvement.

The OpenMP parallel setup running on one core seems to have very little overhead, as the run time only increased by 8%. From one to two cores the run time drops almost to half (61%), but from 2 to 4 cores there is little improvement.



The two different approaches to parallel the computations scaled in almost exactly the same way on this architecture, but the pthread had a greater overhead.

Comments

The pthread model is general and flexible, but very manual in its present form. The overhead is a concern, as the speedup on a four core machine was not enough to overcome initial overhead. On larger machine, or larger problems with bigger chunks this should be less important.

The openMP is very simple, but requires a certain structure of the problem, but negative log likelihoods which are a constructed purely by a sum of terms are very common and include most random effects models where speedup is often needed.

References

Fournier, D. & Sibert, J. (2013). The ADMB pthread_manager Class. Available at: `http://www.admb-project.org/developers/parallel/threaded2.pdf`