Introduction to PLAPACK

I. Plan:

- General facts you need to use PLAPACK
- A bit of related algebra
- Code for this problem
- Some older results for how parallelization works with this code.

General Information

On our cluster PLAPACK is installed in:
/cluster/plapack-3.2.1
Notice, that to use PLAPACK we have to work with MPICH, not LAM MPI. We have a version of MPICH. To make MPICH default MPI you would need to write in your .bash_profile something like:

```
export MPIRUN_HOME=/opt/mpich/ch-p4/bin
export PATH=/opt/mpich/ch-p4/bin:$PATH
```

More General Information

Make sure you have right paths in the Make.include file
Look into Makefile used to compile program.
Notice, this is the case when you definitely want to use Makefile: you don't want to type this manually each time. For example:

```
ParaUltraSlice : #$(SLICEOBJS) #$(PLAPACKLIB)
	$(CC) $(CFLAGS) -c -o ParaUltraSlice.o ParaUltraSlice.c
	$(LINKER) -o ./ParaUltraSlice $(LFLAGS) $(SLICEOBJS)
$(PLAPACKLIB) $(RMATH) $(LIB)
```

In place of $(CFLAGS) only you have to specify couple lines of options including location of PLAPACK, type of machine and operating system, etc.

Model

The model proposed :

$$y = X\beta + Z + \in$$

$Z \sim N\left(0,\ \sigma_z^2\,\Omega\left(\phi\right)\right)\ \in \sim N\left(0,\ \sigma_e^2\, I\right)$

So, we have some covariate coefficients β,

Z – a vector capturing the spatial correlation

$\in$ – iid measurement errors

Reparametrization and priors

$Y \sim N\left(X\beta, \sigma_z^2 \Omega(\phi, \rho) + \sigma_e^2 I\right)$ or if $\sigma^2 = \sigma_z^2 + \sigma_e^2$ and $\kappa = \frac{\sigma_e^2}{\sigma^2}$

$Y \sim N\left(X\beta, \sigma^2 \left[(1 - \kappa)\, \Omega(\phi, \rho) + \kappa I\right]\right)$

placing IG priors on σ_z^2 and σ_e^2,

normal on β,

trancated normal on ρ and uniform on ϕ

Algorithm:

So, the algorithm we will look at does following

1. Draw $(\phi^{(k)}, \rho^{(k)}, \kappa^{(k)} \mid Y)$ from $p(\phi, \rho, \kappa \mid Y)$
2. Draw $\sigma^{2\,(k)}$ from $p(\sigma^2 \mid \phi^{(k)}, \rho^{(k)}, \kappa^{(k)}, Y)$
3. Draw $\beta^{(k)}$ from $p(\beta \mid \phi^{(k)}, \rho^{(k)}, \sigma^{2\,(k)}, \kappa^{k}, Y)$

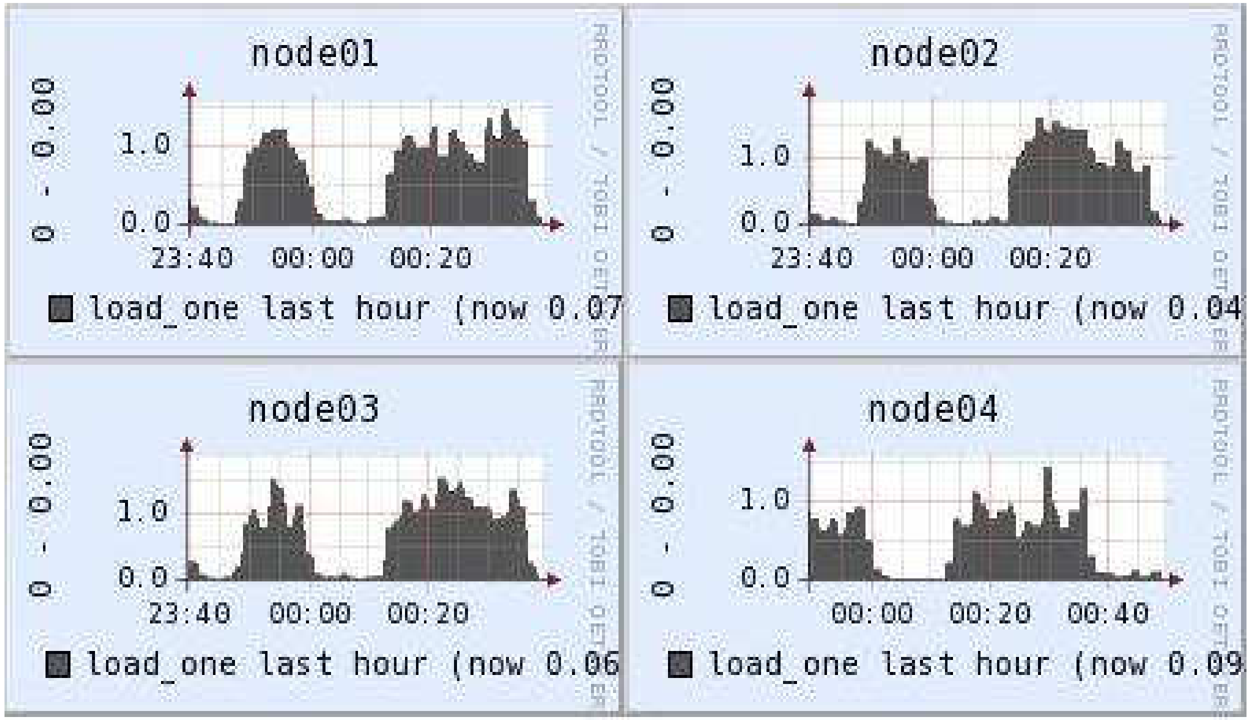
For the third step the slice algorithm is used

Lets move to the program

- The MCMC draws are quite time consuming, and they are sequential.
- The good news is that big part of the computation time is spent on matrix operations. This can be parallelized.
- The general setup of the program:
- a) Parallelize matrix computation at each step
- b) Synchronize, since at the new steps we will need to use new values of parameters.

Cluster Load

- Here is the picture of nodes activity after setting -np 16 for 4 nodes. Time 00:20-00.35

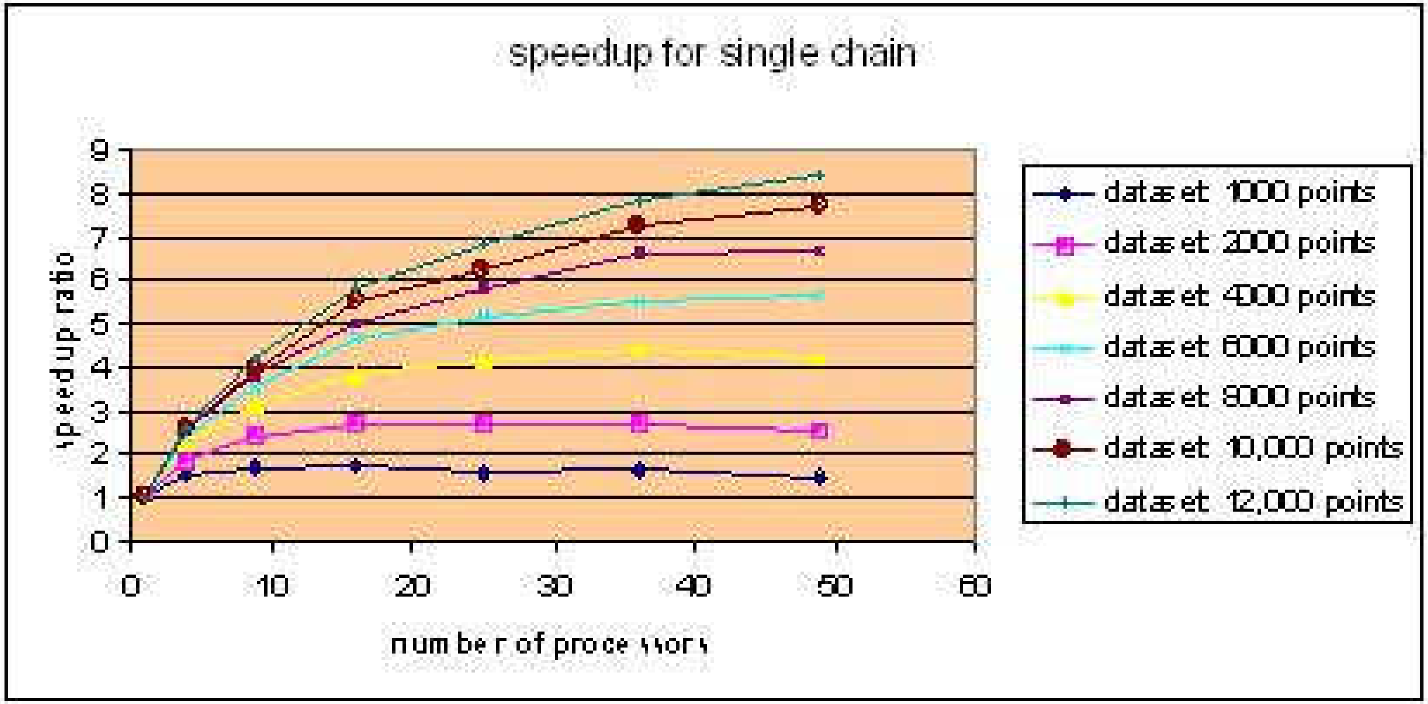


Cluster Load (cont)

- Since I couldn't use xmpi I thought we still may try to get some information from the general ideas.
- First of all, you may see that all four nodes were relatively busy all the time with approximately same load. This at least suggests that PLAPACK did well in terms of load balancing.
- The next picture will be about some previous timing simulations performed for this program.

Timing

- For timing of this application I would like to show the simulation results done by essentially this same algorithm (adapted to Grid) by the GROW in the year 2006.



Improvement

- Notice that here we see quite similar picture to some of the previous topics: for small dataset size there is no improvement and at some point overhead calculations actually take even more time than there is a gain.
- For bigger datasets improvement can be quite substantial: for 12k points dataset it is possible to get results with 50 processors 10 time as quick as with one processor.

References

1. Robert A. Van De Geijn. Using PLAPACK. The MIT Press. 1997
Or online version:
http://www.cs.utexas.edu/~plapack/Guide/Guide_html.html

2. Wenli He and others. Using PLAPACK and MPICH-G2 to Grid-Enable Bayesian Geostatistical Models, 2006
http://www.teragrid.org/library/TG06_WHe-etal_present.ppt