Regression tests and benchmarking/optimization procedures for Regional climate model RegCM-4.1 climate code.

Tamirat Bekele Jimma
In collaboration with **ICTP** and **CNR-IOM**
Adviser: S. Cozzini & G. Mengistu

Addis Ababa University

May 20, 2011

RegCM climate model

Project goal

Domains used in this test

Experimental setup and procedures

Compilers, Libraries and Compiler Flags

High performance computing clusters

Metrics

Results

Performance differences

Scalability of RegCM-4.1

Additional new feature included in RegCM-4.1

Data compression capability feature of RegCM-4.1

Final comment

Objective:

- to compare performance among different versions of the RegCM model, namely version 3, version 4.0 and version 4.1.
- to measure performance of the latest release v4.1 among compilers, libraries and hardware architectures in order to outline the optimal computing environment for the code

Domains used:

Tested domains

Description	Grid Size
Small European	34×64
Ethiopian	112×128
Medium European	128×128
Big European	160×192
African	250×256

Reference datasets created for future work

Name	Grid size	Data size (Gb)
Small European	34×64	0.241
Ethiopian	112×128	1.7
Big European	160×192	3.6
European	128×128	1.81
East Asian	186×224	4.46
Central America	160×288	5.14
South America (dry)	202×192	4.46
South America (wet)	202×192	4.38
African	250×256	7.16

These are CORDEX domains of 5 years simulations and a monthly average datasets obtained by dividing for the number of months.

Procedures followed:

- One month simulation over January 1989 for all domains.
- Some CORDEX domains 5 year simulations also included.
- Global dataset used 'EIN15'.
- SVN trunk version 1928, 1956, 1976 and 2044 used.
- Only big European domain results presented for this presentation.
- Only parallel version of the code is tested.

Compilers and libraries studied:

Operating system

- CentOS v5.5

Compilers

- Intel 2011
- GNU 4.4.0
- PGI 10.9

Libraries

- NetCDF 4.1.1
- OpenMPI 1.4.3

Compiler flags used:

Default compiler options provided with the package.

Compilers	**F90FLAGS**
GNU	-O3 -fconvert=big-endian
PGI	-O3 -byteswapio
INTEL	-O3 -fp-model precise -convert big-endian -heap-arrays -assume byterecl

Hardwares used:

Name	CPU (GHz)	Cores per node	Infiniband connection
ARGO	Intel E5620 2.4	8	QDR
HG1	Intel E5520 2.27	8	DDR
SP6	IBM Power6 4.7	32	QDR

time measurement utility used:

- **/usr/bin/time** Linux wallclock time measurement utility is used
- RegCM-4.1 has its own internal clock time measurement utility

Overhead by using **/usr/bin/time** utility

Compiler	**RegCM time**	**/usr/bin/time**	**Difference**
GNU	5609	5657	0.8%
Intel	3354	3432	2.3%
PGI	4148	4197	1.2%

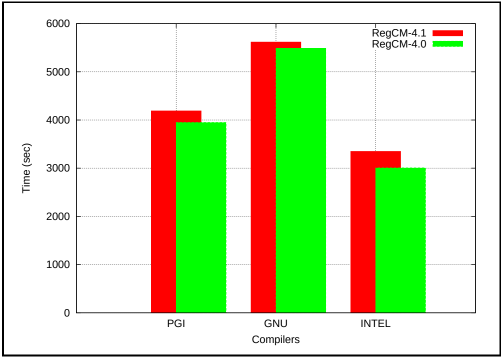
For eight processors in a single node.

Time taken by one month simulation in different domains and compilers.

Compilers	European (34x64)	Ethiopian (112x128)	European (160x192)	African (250x256)
INTEL	256.27	1676.03	3348.86	10030.78
PGI	330.49	1938.76	4189.08	12630.96
GNU	477.02	2653.21	5617.39	16747.22

Performance difference between compilers

- Comparison among compilers in ARGO machine for Big European domain.
- Intel is faster than the others
- GNU is slowest of all



Performance comparison between versions

Version 4.1 is:

- Slower than version 4
- Much slower than version 3.

<u>But it is:</u>

- More user friendly than others
- Much portable than previous versions
- Stable

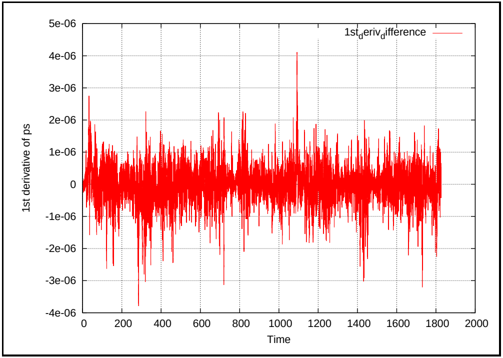
Version	Abs. WCT	Time difference
3	2892	
4.0	3142	+7.96%
3	2892	
4.1	3492	+17.18%

Floating point relaxation (FP)

- Relaxing FP accuracy can improve performance by about 10% for Intel compiler
- Aggressive optimizations have impact on the accuracy and precision loss.

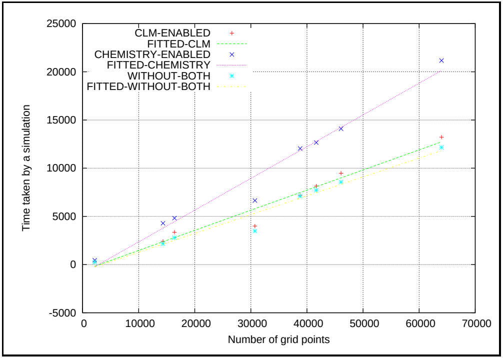
Compiler	FP flags	WCT
Intel 12.0	fp -precise	865
	fp -fast=1	793
	fp -fast=2	786
Intel 11.1	fp -precise	893
	fp -fast=1	819
	fp -fast=2	807

For instance:



- We compared the output for **-fp-model precise** and **-fp-model fast=1** in Intel compiler
- First derivative output of the model varies in the order of 10^{-06}.

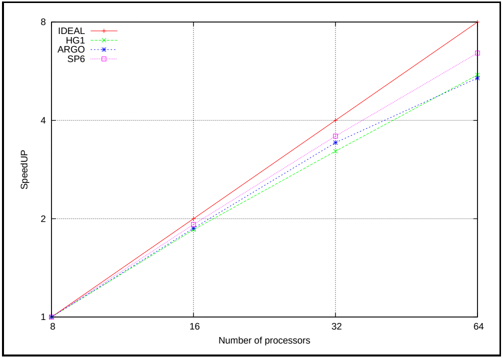
Computational cost of different physical parameterizations



- Results show us the performance difference between chemistry enabled and disabled simulation can go up to 43%.

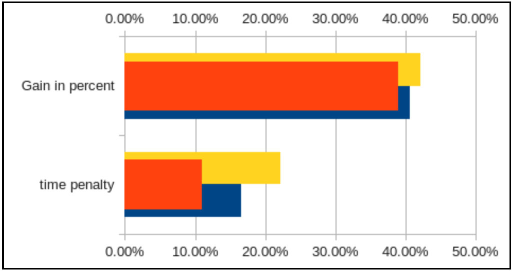
Speedup of RegCM-4.1

- Relative speedup plotted as a function of the number of processors
- Speedup falls beyond optimal when using more than 32 processors



Storage and performance comparison

Use of HDF5 through NetCDF libraries for data compression.



- One can save about 40% storage space
- But he/she will loss 16% in performance
- So we advise compression should be used with caution

Conclusion

We have found out that:

- Intel is the fastest compiler suitable for RegCM.
- PGI can be used as an alternative.
- Some most time consuming MPI communications and FORTRAN modules to be improved.
- The model can scale up.
- Data compression capability can be an option for limited storage spaces.

Future works

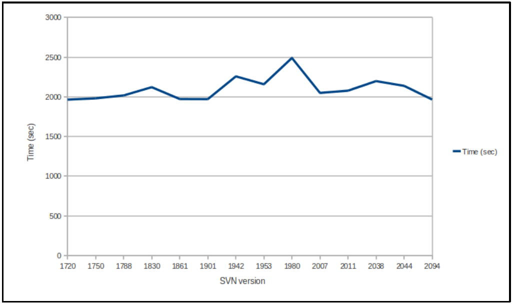
Tasks remain to be done

- Optimization and Regression tests
- 2D domain decomposition

SVN comparison

Simulations we've done on the big European domains with 16 processors shows:

- We were working on the svn version of higher peaks relative to others.
- Shows a variation of upto 500 seconds.



Acknowledgment

Thanks all

Special thanks to:

Stefano Cozzini
Graziano Giuliani
Martin Scarcia
Clement Onime
Moreno Baricevic
Iztok Gregori
Antonio Messina