Generating tangent linear and adjoint versions of NASA/GMAO's Fortran-90 global weather forecast model

Ralf Giering[1], Thomas Kaminski[1],
Ricardo Todling[2], Ronal Errico[2], Ronald Gelaro[2],
and Nathan Winslow[2]

[1] FastOpt

[2] 

NASA/GSFC

Copy of presentation at http://www.FastOpt.com

Outline

- Target code : finite volume GCM (fvGCM)
- AD tool : TAF
- AD of fvGCM
- Performance
- Summary

finite volume GCM

- finite volume GCM (fvGCM) is the global weather forecast model of NASA/GMAO
- consists of dynamical core (Navier-Stokes solver) + physics (parametrisations for clouds etc)
- dynamical core by Lin and Rood (1996, 1997) and Lin (1997)
- state comprises two horizontal wind components, pressure difference, potential temperature, and moisture
- various spatial resolutions and integration periods
- GMAO's data assimilation system requires tangent linear (TLM) and adjoint (ADM) codes of fvGCM's dynamical core
- Further applications include singular vector analysis, adjoint sensitivity studies, and inverse modelling of tracers

finite volume GCM

- for most applications, TLM and ADM need to linearise around an external state-trajectory provided by an integration at higher spatial resolution including full physics
- dynamical core comprises 87'000 lines of Fortran 90 code, excluding comments
- uses features such as
  free source form, derived types, allocatable arrays
- parallelises using *OpenMP, MPI*, or both
- good performance TLM and ADM crucial for applications

TAF
Transformation of Algorithms in Fortran

- Source-to-source translator for Fortran-77/95
- Commercial successor of TAMC
- Forward and reverse mode (1^{st} derivatives):
  Tangent linear and adjoint models
- Scalar and vector mode
- Efficient Hessian (2^{nd} derivative) code
  by applying TAF twice (e.g. forward over reverse)
- Command line program with many options
- TAF-Directives are Fortran comments
- Extensive and complex code analyses
  (similar to optimising compilers)
- Generated code is structured and well readable

TAF
More features

- Generation of flexible store/read scheme for required values triggered by TAF init and store directives
- Generation of simple checkpointing scheme (Griewank, 1992) triggered by combination of TAF init and store directives
- Generation of efficient adjoint (Christianson, 1996, 1998) for converging iterations triggered by TAF loop directive
- TAF flow directives for black-box routines,
  or to include user provided derivative code
  (exploit self-adjointness, MPI wrappers, etc...)
- Automatic Sparsity Detection
- Basic support for MPI and OpenMP

TAF
Ongoing Development

- TAF is constantly being adapted to new language standards, next targets:
  - Fortran 2000
  - OpenMP 2
- TAF code analyses are constantly being extended
- TAF algorithms are constantly being improved and adapted to the needs of the users
- FastOpt is giving support for TAF users
- FastOpt is offering consulting for AD and further projects

AD of fvGCM
our general approach

Enhancements of TAF + Modifications of fvGCM source code:

- **Storing required variables: 41 TAF init directives and 75 TAF store directives**
- **204 TAF flow directives for generation of specific call sequences (e.g. for FFT)**
- **11 TAF loop directives to indicate parallel loops**
- **Complex control flows simplified at two places**
- **Initialisation split off the main model code**

No modifications of generated code

-> TLM/ADM generation automated one-click procedure

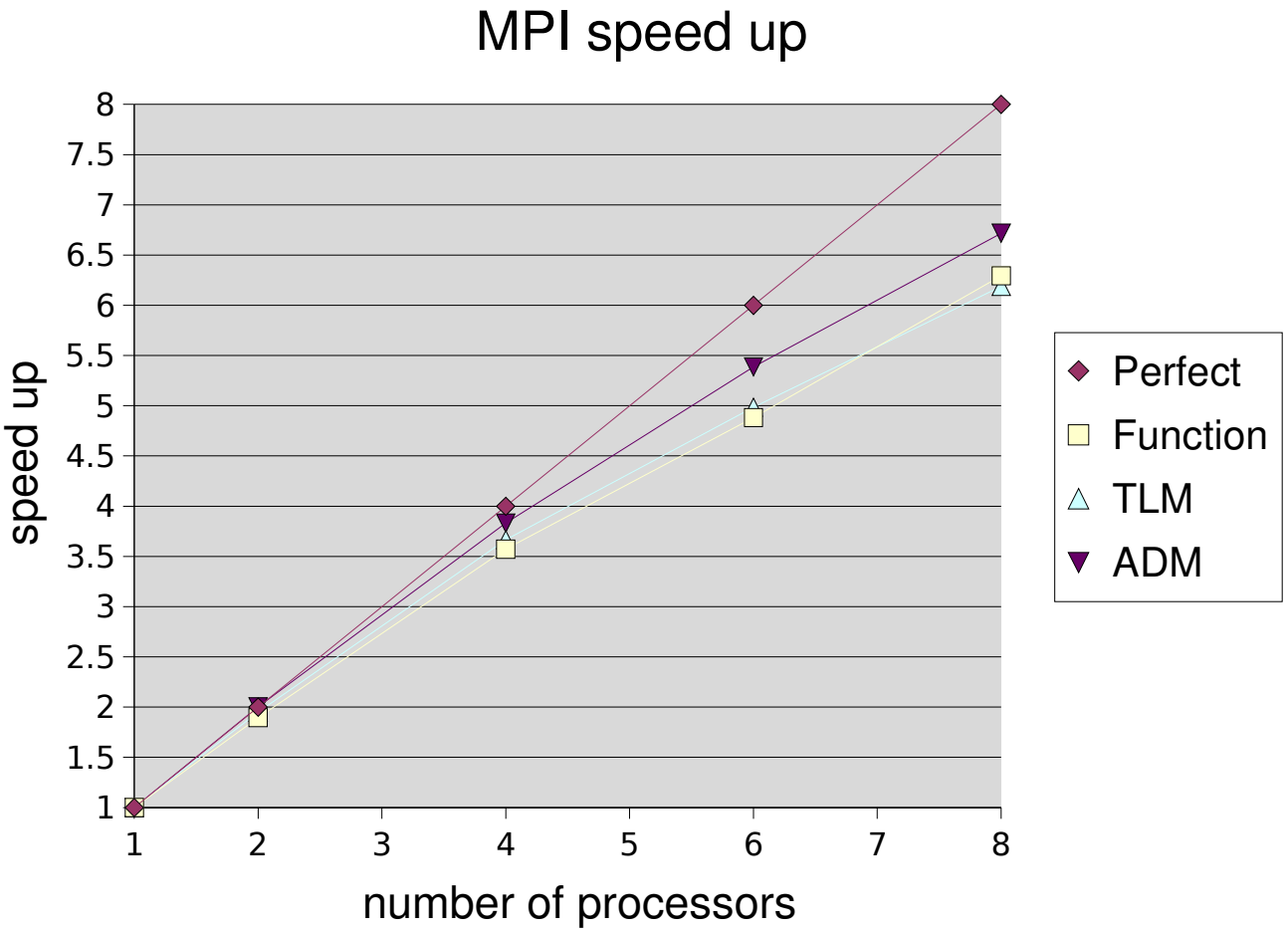
AD of fvGCM
Exploiting TAF flow directives

- TLM and ADM need to linearise around external trajectory
  - function code overwrites state
  - data flow from initial to final state interrupted
  - straight forward use of AD results in erroneous derivatives
  - exploit TAF's flexibility in generation of store/read scheme: trigger generation of desired behaviour by combination of TAF init and store directives
  - generated code is, however, not derivative of function code
- Code uses FFT and its inverse
  - reusing FFT in TLM and inverse FFT for ADM is more efficient than differentiating FFT (Talagrand 1991; Giering et al, 2002)
  - reuse triggered by TAF flow directives

AD of fvGCM
Handling MPI

- Model has wrapper routines (e.g. mp_send3d_ns) that call the respective MPI library routines (e.g. mpi_isend)
- Wrappers are encapsulated in one module
- Decision between MPI-1/2 happens in wrappers
- In forward mode, TAF handles (most) MPI calls.
  We need, however, TLM and ADM
  -> Construction of MPI in TLM and ADM at level of wrappers
    - inserting of TAF flow directives for wrappers
    - TLM and ADM wrapper routines hand written
    - TLM and ADM wrappers reuse model wrappers (easy to maintain)
    - handling of MPI-1 and MPI-2 at once
- Encapsulation helped a lot!

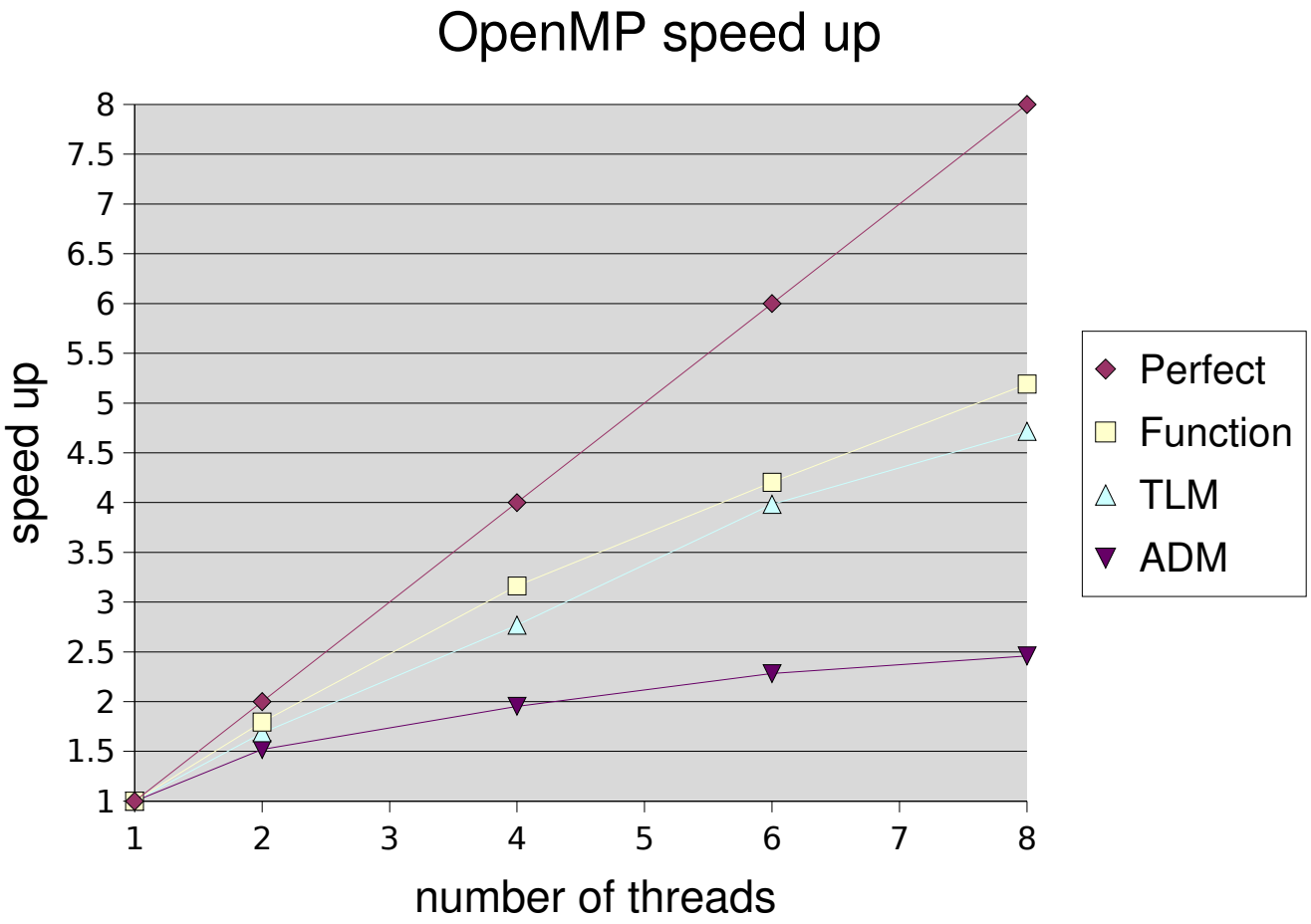
MPI


MPI speed up
speed up
8
7.5
7
6.5
6
5.5
5
4.5
4
3.5
3
2.5
2
1.5
1
1
2
3
4
5
6
7
8
number of processors
Perfect
Function
TLM
ADM

AD of fvGCM
Handling of OpenMP

- Model uses only a single directive:
  !$omp parallel do
- TAF analyses the loop-carried dependencies
- For ADM loop, according to the dependencies, TAF generates the proper !$omp directive for the adjoint loop and (if necessary) additional statements to preserve parallelism
- Can generate code for OpenMP-1 or OpenMP-2
- OpenMP-1 adjoint of fvGCM need many critical sections, because OpenMP-1 does not support array reductions.
- OpenMP-2 does and thus yields faster code.
- For TLM loop, TAF uses the similar directive

OpenMP-1


OpenMP speed up
speed up
number of threads
1
1.5
2
2.5
3
3.5
4
4.5
5
5.5
6
6.5
7
7.5
8
1
2
3
4
5
6
7
8
Perfect
Function
TLM
ADM

Performance summary

- Performance tests on:
  Linux Intel 4 : coarse spatial resolution (72 x 46 x 18)
  SGI O2000 /w OpenMP1, 8 threads : medium resolution (144 x 91 x 55)
  SGI O2000 /w MP1-1, 8 processors : medium resolution (144 x 91 x 55)
- Bug in SGI Fortran-compiler when optimisation switched on causes ADM errors of a few percent:
  - ADM : Optimisation on
  - ADM-noopt : Optimisation off

Platform/Setup	TLM	ADM	ADM-noopt
Linux Intel 4	1.5	7.0	-
SGI OpenMP-1 / 8 threads	1.5	10.8	20.6
SGI MPI-1 / 8 processors	1.5	3.9	12.6

some larger TAF Derivatives

Model (Who)	Lines	Lang	TLM	ADM	Ckp	HES
NASA/DAO (w. Todling & Lin)	87'000	F90	1.5	7.0	2 lev	-
MOM3 (Galanti & Tziperman)	50'000	F77	Yes	4.6	2 lev	-
MITGCM (ECCO Consortium)	100'000	F77	1.8	5.5	3 lev	11.0/1
BETHY (w. Knorr, Rayner, Scholze)	5'400	F90	1.5	3.6	2 lev	12.5/5
Nav.-Stokes-Solver (Hinze, Slawig)	450	F77	-	2.0	steady	-
NSC2KE (w. Slawig)	2'500	F77	2.4	3.4	steady	9.8/1
HB_AIRFOIL (Thomas & Hall)	8'000	F90	-	3.0		-

- Lines: total number of Fortran lines without comments
- Numbers for TLM and ADM give CPU time for (function + gradient) relative to forward model
- HES format: CPU time for Hessian * n vectors rel. t. forw. model/ n
- 2 (3) level checkpointing costs 1 (2) additional model run(s)

Performance ratios
TAF vs hand coded adjoints

Code	Hand	TAF/TAMC	relativ
EPT (MINPACK-2)	1.5	1.9	-27%
GL1 (MINPACK-2)	1.5	1.7	-13%
GL2 (MINPACK-2)	2.1	1.3	38%
MSA (MINPACK-2)	1.8	1.6	9%
PJB (MINPACK-2)	1.8	2.2	-26%
SSC (MINPACK-2)	1.2	1.1	4%
Nav.-Stokes (Hinze and Slawig)	1.9	2.0	-5%
EULOSOLDO (Cusdin and Müller)	2.4	2.7	-12%
3D NS CFD-code (Cusdin and Müller	1.2	1.2	-4%

=> Performance of TAF generated adjoints is comparable to that of hand written adjoints

Summary

- TLM and ADM of large Fortran 90-code for OpenMP and MPI
- TAF can update derivative in one-click procedure
- Concept can be generalised to other modelling systems
- AD helps to reduce the delay from model development to data assimilation and related applications
- Concepts are being transferred from Fortran to C(++)

More Info:

- Posters on
  -> AD of fvGCM
  -> TACC++ : C++ counterpart of TAF
  -> CCDAS : another AD-based F90 modeling system
- http://www.FastOpt.com/