A framework for detailed multiphase cloud modeling on HPC systems

ParCo 2009, 3. September 2009, ENS Lyon, France

Matthias Lieber\textsuperscript{a}, Ralf Wolke\textsuperscript{b}, Verena Grützun\textsuperscript{b}, Matthias S. Müller\textsuperscript{a}, Wolfgang E. Nagel\textsuperscript{a}

\textsuperscript{a} Center for Information Services and High Performance Computing (ZIH), TU Dresden, Germany

\textsuperscript{b} Leibniz Institute for Tropospheric Research (IfT), Leipzig, Germany
Outline

- Introduction to cloud modeling
- Framework FD4
- Overhead benchmark
- Conclusion & outlook
Introduction: Cloud modeling

- Clouds play a major role for climate and weather: They
  - influence the radiation budget of the planet,
  - are part of the hydrological cycle,
  - interact with aerosol particles and pollution

- Clouds represent one of the major uncertainties in climate and weather models

- Unsatisfying improvements in precipitation forecast during the last decades
Cloud droplets are described by their bulk mass only.
Actual size distribution is neglected.
Computationally fast.
Used in most weather models.
Introduction: Spectral microphysics schemes

- Bin-wise discretization of size distribution
- Allow more detailed modeling of interactions between aerosol particles, cloud formation, and precipitation
- Computationally very expensive (runtime & memory)
- Only used for process studies up to now
Introduction: Spectral microphysics in weather models

- Spectral microphysics schemes have been implemented in the weather models MM5, WRF, and COSMO

- High computational costs, e.g. COSMO-SPECS [1]:
  - Spectral cloud model SPECS and related communication take up more than 90% of the total runtime
  - High load imbalance between cloudy and cloud-free areas
  - A simple artificial case (80x80x48 grid, 30 min forecast) runs 4.5 hours on 100 CPUs of an SGI Altix 4700

How can we make such model systems run faster to be more applicable for practical cases?

- Take heterogeneity of cloud processes into account
  - Do not run the spectral model in cloud-free grid cells
  - Dynamic load balancing

- High memory demands (100-1000 variables per grid cell)
  - Adapt memory allocation to spatial cloud structure
  - Optimize data structures for high number of variables

➔ New framework specialized on coupling multiphase cloud processes to a weather model:

**FD4 = Four-Dimensional Distributed Dynamic Data structures**
Framework FD4: Basic Idea

- Present approaches:
  - Cloud model is implemented as a sub-module within the weather model
  - Uses (static) data structures of the weather model
  - No dynamic load balancing

- Our idea:
  - Separate cloud model data from weather model data structures
  - This allows dynamic adaption and load balancing for the cloud model
  - Couple cloud model as if it was a self-contained model (like in climate models)
Framework FD4: Adaption to cloud structure

- Block-based 3D decomposition of rectangular numerical grid
- Only those blocks are allocated which are required to capture the clouds
- Determined via a threshold for the cloud variables
- FD4 ensures appropriate data for stencil operations
Framework FD4: Dynamic load balancing

- Two methods implemented:
  - Hilbert space-filling curve (SFC)
  - Graph repartitioning using ParMETIS

- Partition calculation will be called very frequently
  - Must be very fast
  - SFC much faster than ParMETIS
Framework FD4: SFC vs. ParMETIS Benchmark

- Advection of an 'abstract cloud'
- FD4 adapts to cloud structure
- Continuous rebalancing of blocks required
- Comparable partition quality (edge-cut)
- ParMETIS calculation takes very long at high CPU numbers
Spatially small blocks are good:

- With 100s of variables per cell, only small blocks will not exceed the cache size, e.g. 
  \(8^3 \text{ cells} \times 1024 \text{ DP vars} = 4 \text{MB}\)
- Fine-grained adaption to cloud structure and load balancing

But usual way to allocate block with ghost cells is very costly

FD4's optimization: Allocate ghost cells only at the partition border in the form of additional *ghost blocks*
How ghostcells waste memory

$2^2$: 800%
$2^3$: 2600%
$4^2$: 300%
$4^3$: 700%
$8^2$: 125%
$8^3$: 237%

Additional memory costs due to ghost cells for different block sizes in 2D and 3D
Spatially small blocks are good:

- With 100s of variables per cell, only small blocks will not exceed the cache size, e.g. $8^3 \text{ cells} \times 1024 \text{ DP vars} = 4\text{MB}$
- Fine-grained adaption to cloud structure and load balancing

But usual way to allocate block with ghost cells is very costly

FD4's optimization: Allocate ghost cells only at the partition border in the form of additional *ghost blocks*
Framework FD4: Other features

- Coupling interface to exchange data fields between cloud model and meteorological model
- Output to Vis5D and NetCDF4 (not yet optimized)
- Fortran 95 + MPI (like most weather models)
- General applicable to multiphase modeling, i.e. no assumptions specific to meteorology / cloud modeling
Overhead benchmark

- COSMO real-life scenario, 249 x 174 x 50 grid, 1h forecast
- Cloud variables (water, ice) are transmitted to FD4, which adapts block structure and partitioning dynamically
- Adaption is required every time step (10s)
- FD4 overhead measured on an AMD Opteron cluster with SDR Infiniband

Cloud structure

Block partitioning
Overhead benchmark

- Determination of required blocks scales only little
- SFC partitioning does not scale (serial algorithm)
- Coupling data transfer scales well to 256 processes (mainly communication-bound)
- Overhead highly depends on the number of blocks
Conclusion & outlook

- Spectral cloud microphysics schemes have the potential to increase the understanding of the complex interactions of aerosol particles, clouds, and precipitation.
- Innovative HPC methods are required to lower their computational demands.
- The framework FD4 specialized for an efficient coupling of multiphase cloud processes is developed.
- Challenge: Balance between adaptivity and overhead.

Next steps:
- Use FD4 in COSMO-SPECS
- Implement & tune parallel I/O (based on NetCDF4)
The end

Thank you for your attention!

This work has been funded by the German Research Foundation