Institute of Computer Science
Academy of Sciences of the Czech Rep.
Prague, Czech Republic

Fine resolution modeling with CMAQ-adjoint

Jaroslav Resler, Krystof Eben, Pavel Jurus

E-mail: resler@cs.cas.cz

Introduction

As we advance towards simulation of larger domains in finer resolution, parallel efficiency of the CMAQ 4DVar adjoint code starts to be critical for real cases. The code saves the model state after each internal time step of model (this technique is called *checkpointing*). This process has been identified as a major bottleneck for parallel efficiency. We have tested various strategies for parallelization of checkpointing and the results are presented here.

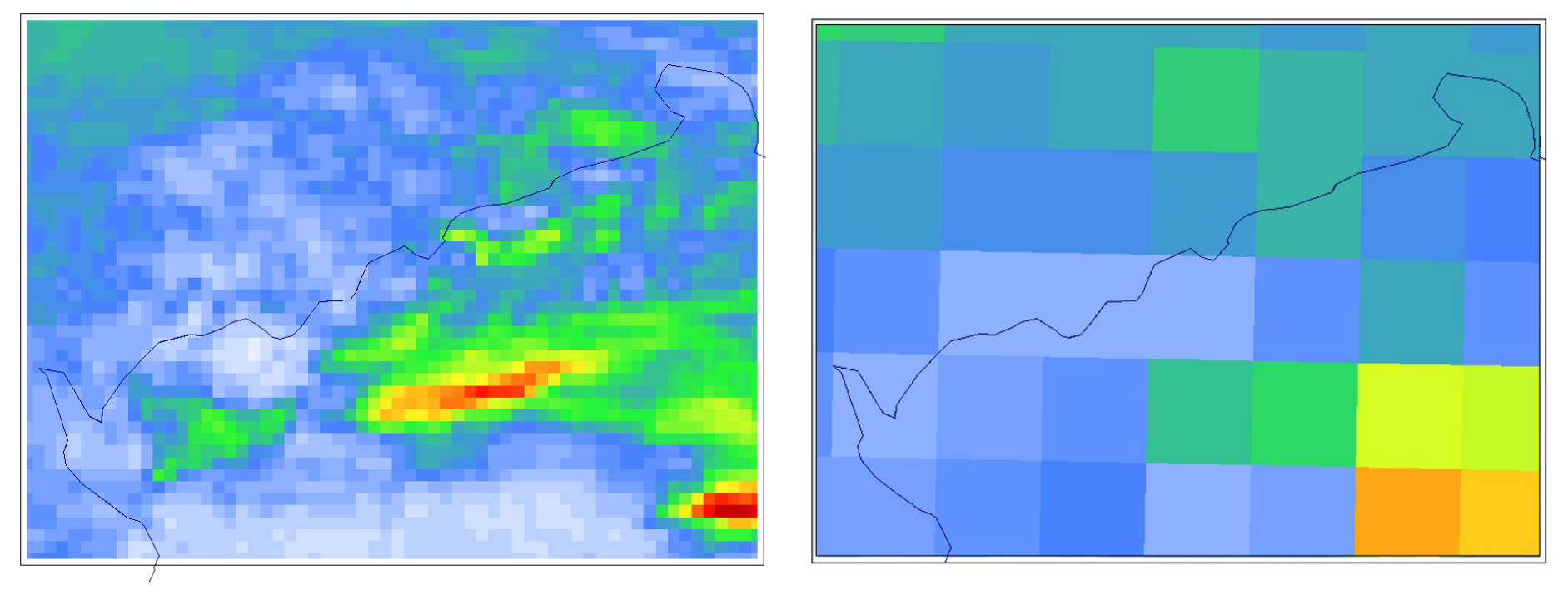
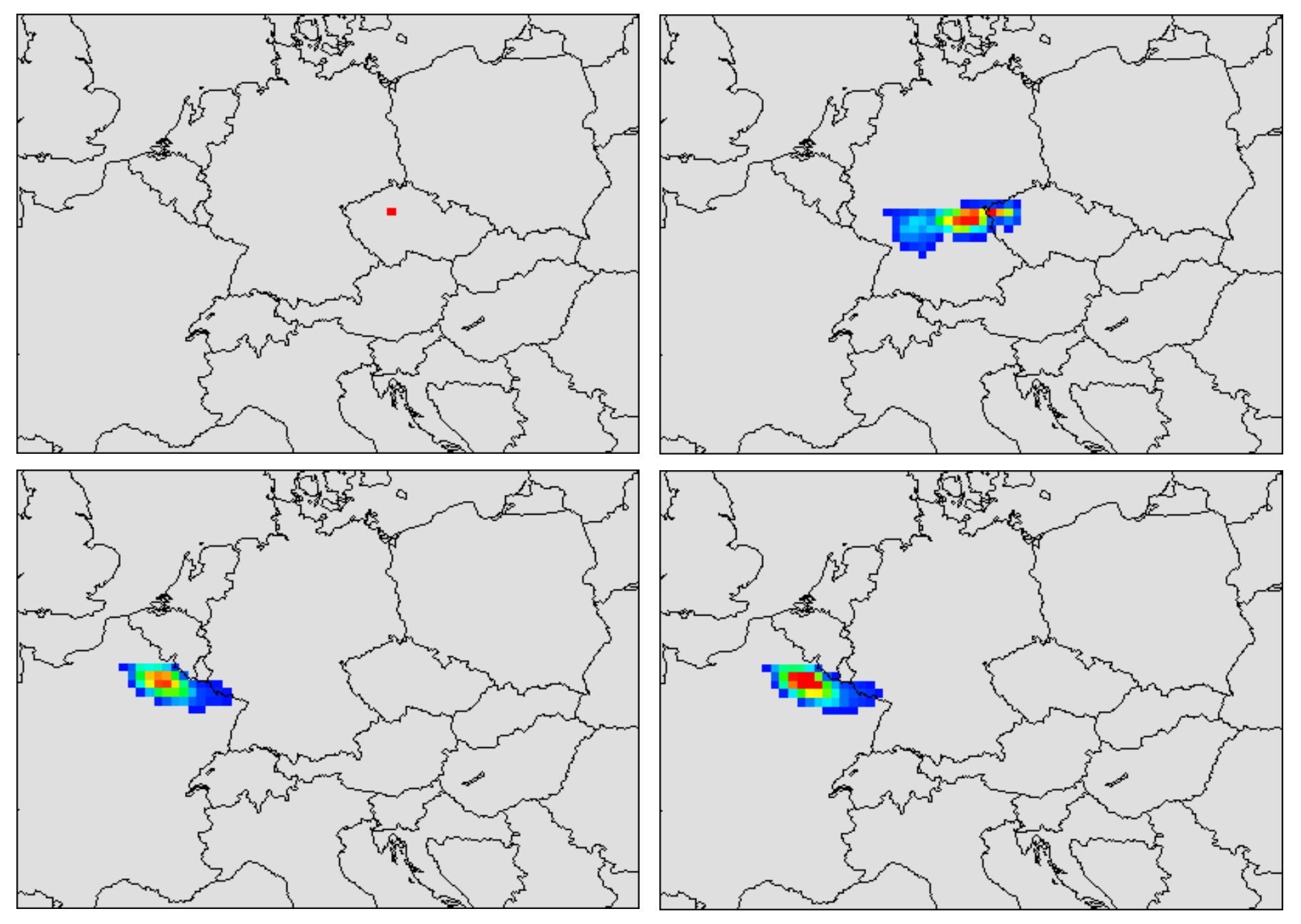


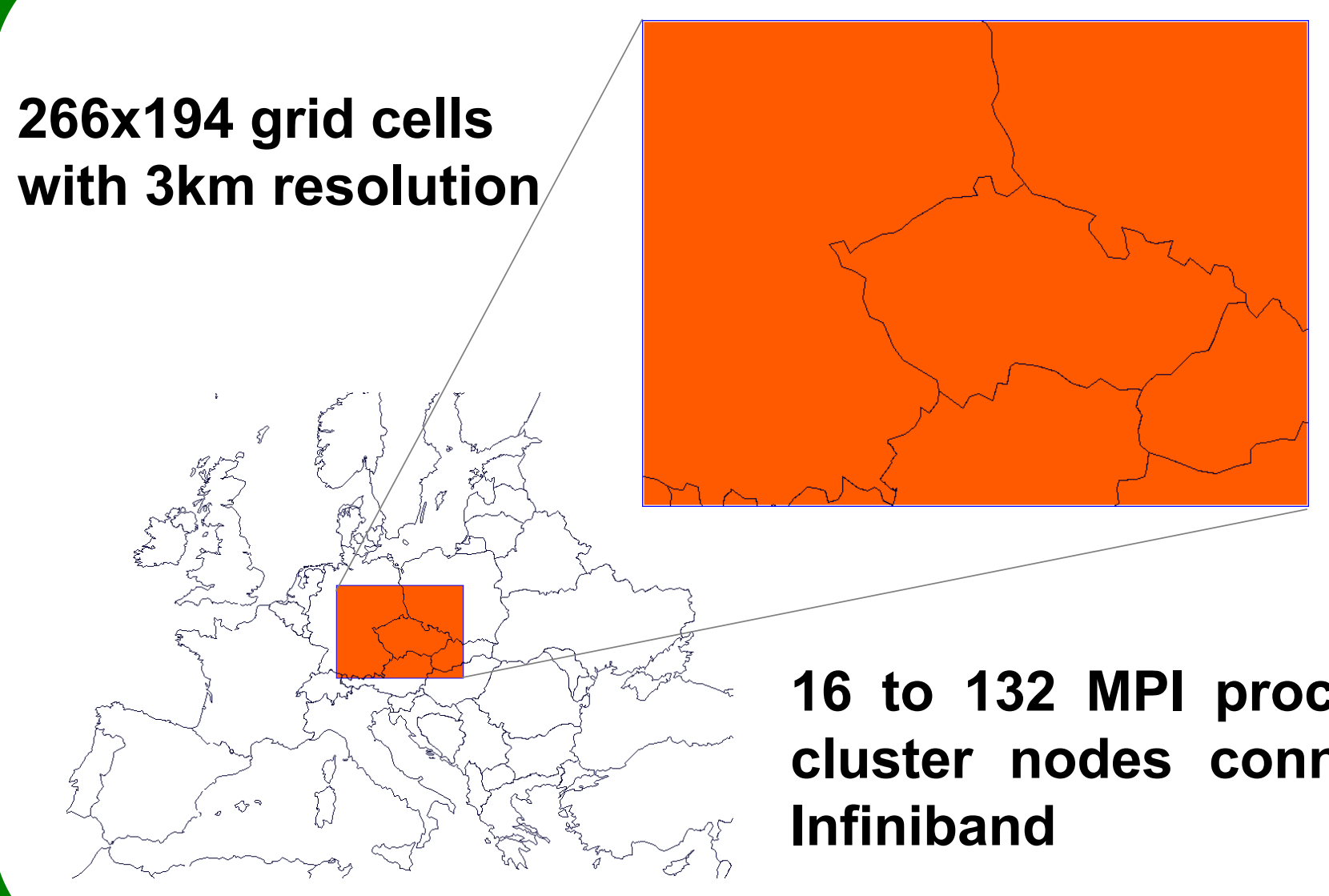
Illustration of differences between coarse (27km) and finer (3km) resolutions for the same domain.



Adjoint sensitivity for a unit ozone receptor in Prague (left upper figure). Cumulative positive adjoint sensitivity to NO emission and adjoint sensitivity to concentrations of NO2 and O3 in 10-th layer 18 hours earlier. During a day, sensitive regions can be hundreds of kilometers far from the receptor. Large simulation domains are therefore necessary.

Simulation domain used for benchmarking

266x194 grid cells with 3km resolution



16 to 132 MPI processes on cluster nodes connected by Infiniband

Changes to I/O in CMAQ adjoint

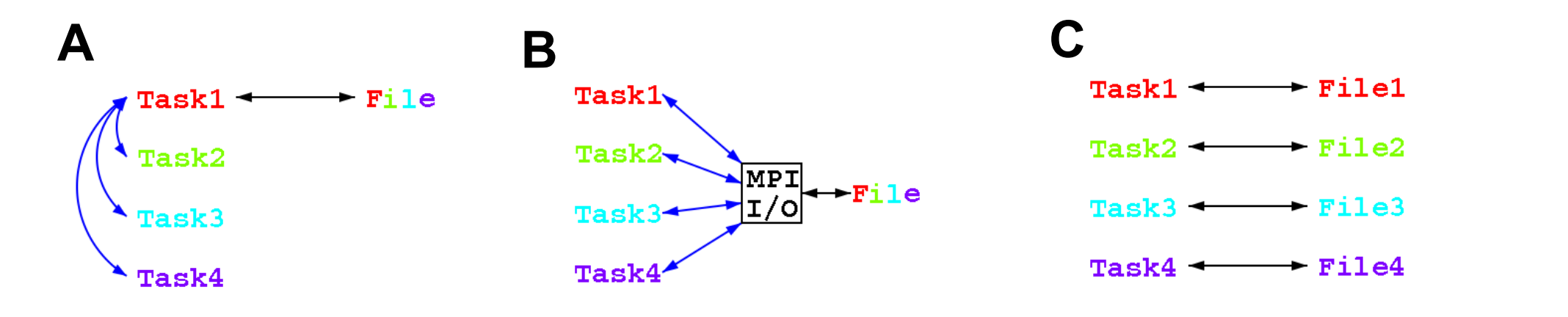
Parallel I/O in CMAQ and CMAQ adjoint is implemented in PARIO library. PARIO collects all of the data into process 0 and then performs serial write to a file system.

- MPI communication "all to one" is not treated as MPI collective operation but is done serially "one after another".
- Serial recording to the file system in process 0 doesn't not allow to utilize the parallel ability of the FS.
- All data are written by one node of the cluster and thus the advantages of the parallel cluster file systems (e.g. PVFS) and local file systems on nodes cannot be utilized.

The aspects of the parallelization of CMAQ model were discussed e.g. in (Kordenbrock 2006)(1) and (Wong 2009)(2).

We have changed the I/O in CMAQ adjoint code so that particular strategy for I/O operations on checkpoint and output files can be chosen at configuration time. At present, we have implemented the following strategies:

- Serial I/O through IOAPI3/PARIO (equivalent to the present state, picture A).
- MPI I/O using Parallel NetCDF library and NetCDF4/HDF5 library (picture B).
- Multiple local files strategy implemented by direct calls of NetCDF (picture C).



Source of the pictures: IBM SP Parallel Scaling Overview - Parallel I/O
(1)Kordenbrock, T.H, Oldfield, R.A.: Parallel I/O advancements in air quality modeling systems, CMAS conference, 2006
(2)Wong, D.: Enhance CMAQ performance to meet future challenges, CMAS conference, 2009

Changes in checkpoint files

4DVar simulation consists of several forward and backward model runs. The checkpoint files in CMAQ adjoint model are repeatedly written and read with very high frequency (i.e. in every model synchronization time step - approximately every 3 minutes of simulation time for the domain with 3km resolution). This contrasts with typically much lower volume of standard I/O operations (e.g. writing model concentrations for each hour of simulation).

The files treated as checkpoint files in CMAQ adjoint model and 4DVar code can be split into two categories :

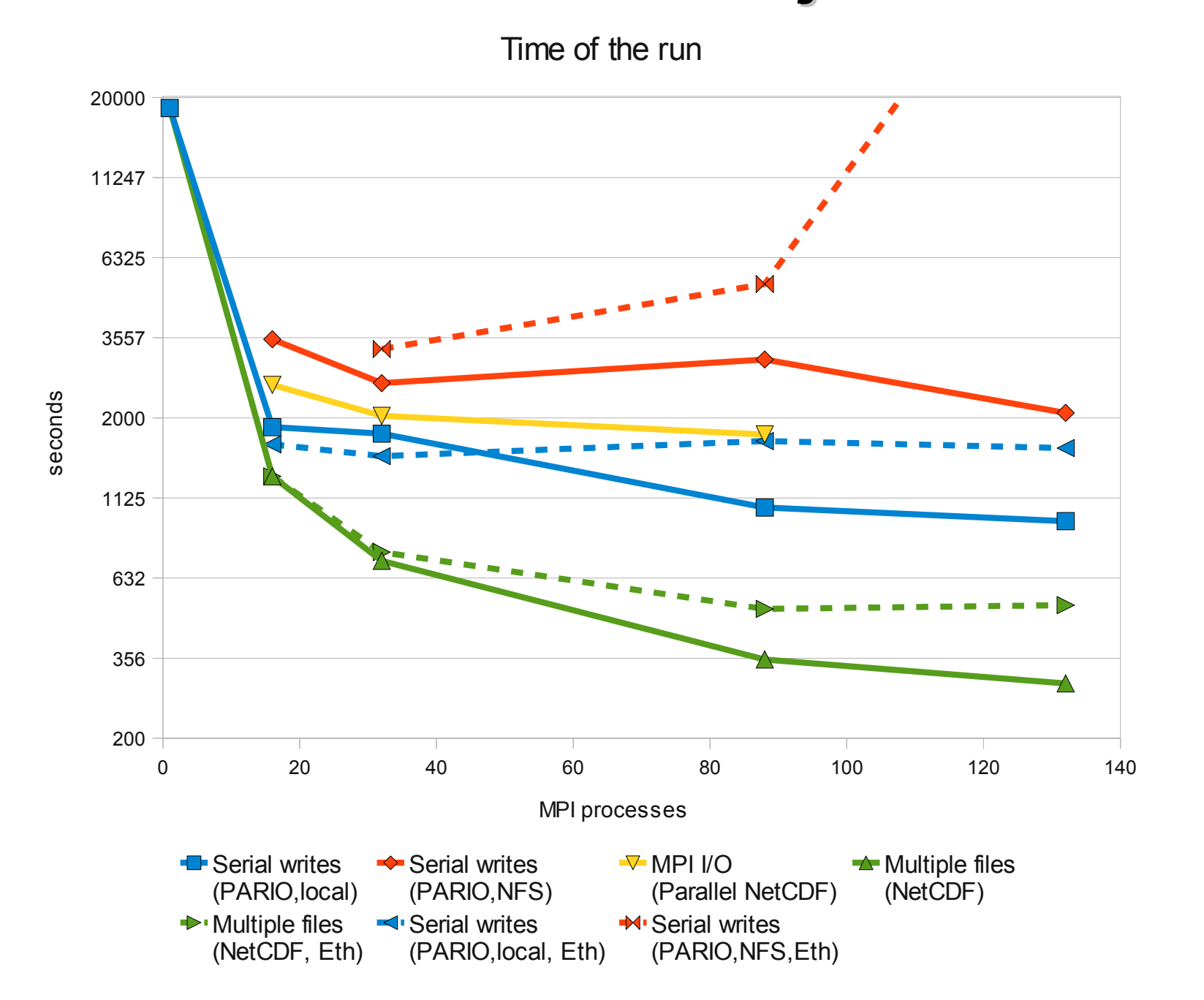
- CMAQ adjoint output and diagnostic files (adjoint variable, sensitivity to emission, emission correction factors, observation operator diagnostic outputs etc.). These files are never read again and they are usually needed only at final step of the backward run (except for debugging).
- The genuine CMAQ adjoint checkpoint files (e.g. checkpoint files for concentrations, level 2 checkpointing). These files can be treated as internal CMAQ adjoint files which need not be read outside the model run.

For the sake of flexibility of the adjoint code, we have extended the code so that different I/O strategies for the output files and checkpoint files can be chosen. Also different frequency of the recording of the output files can be configured. This gives the possibility to choose the best strategy according to the type of application (e.g. model development, offline simulation, operational run) and to the kind of HW configuration (size of the cluster, type of interconnection, type of the storage device).

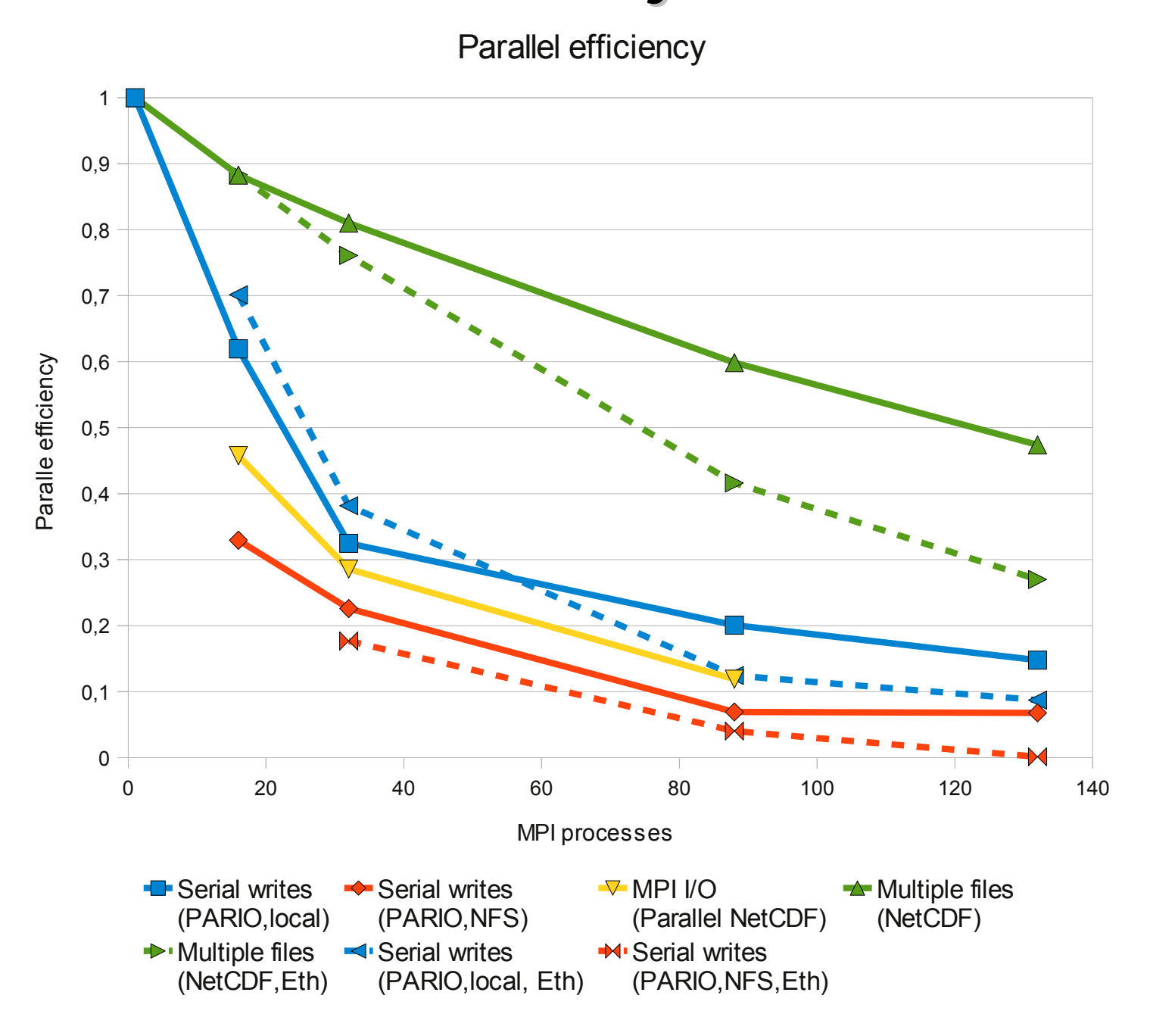
Results

Testing configuration: Linux cluster, NFS cluster storage, fast SAS-2/SCSI local disks on nodes, Infiniband interconnection, MVAPICH2 1.2p1. Some tests were also run for comparison with dedicated 1GB ethernet interconnection and with MPICH2 (dashed lines in graphs). We have performed parallel tests in range from 16 to 132 MPI processes.

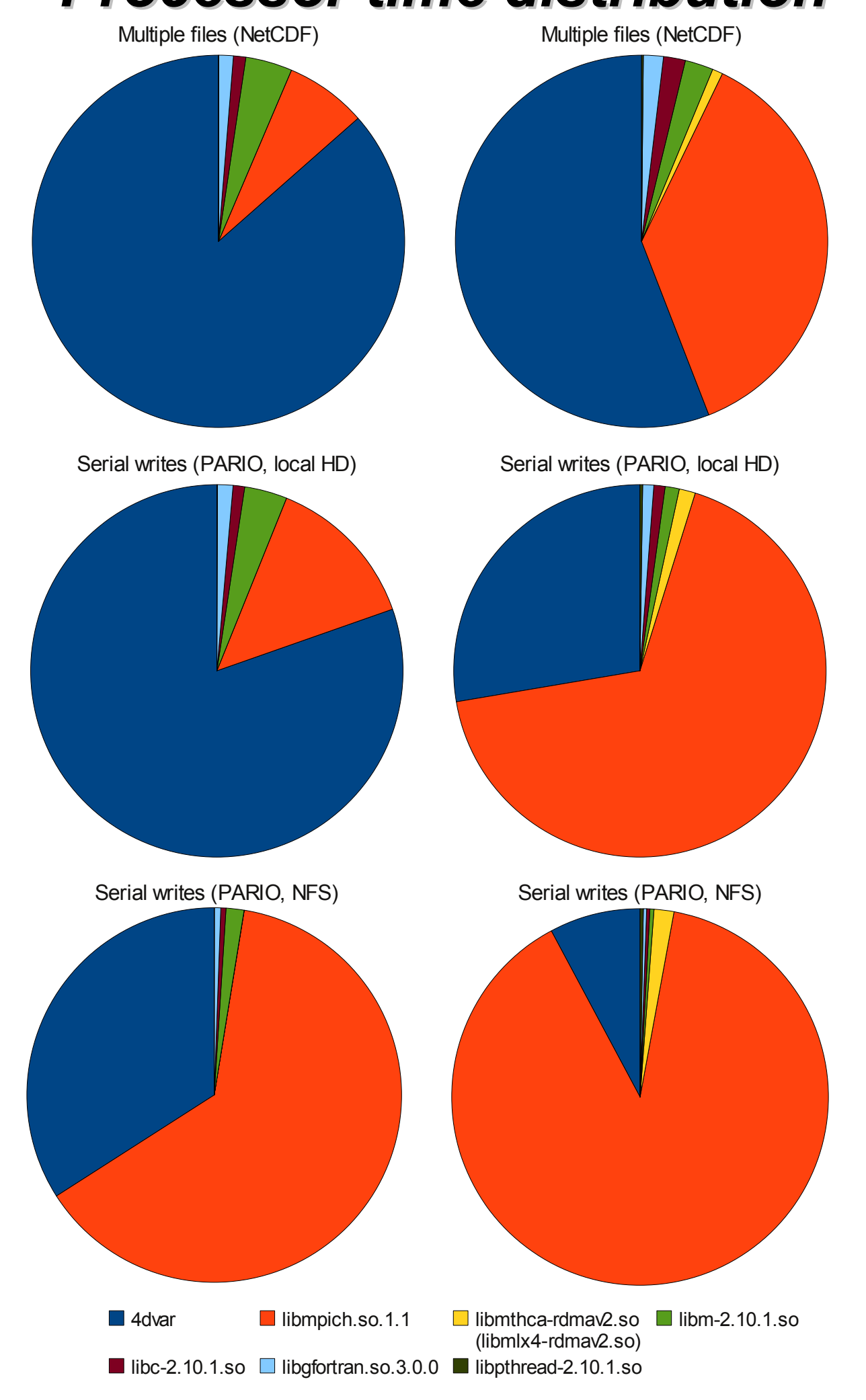
Time of run of CMAQ adjoint model



Parallel efficiency of the model

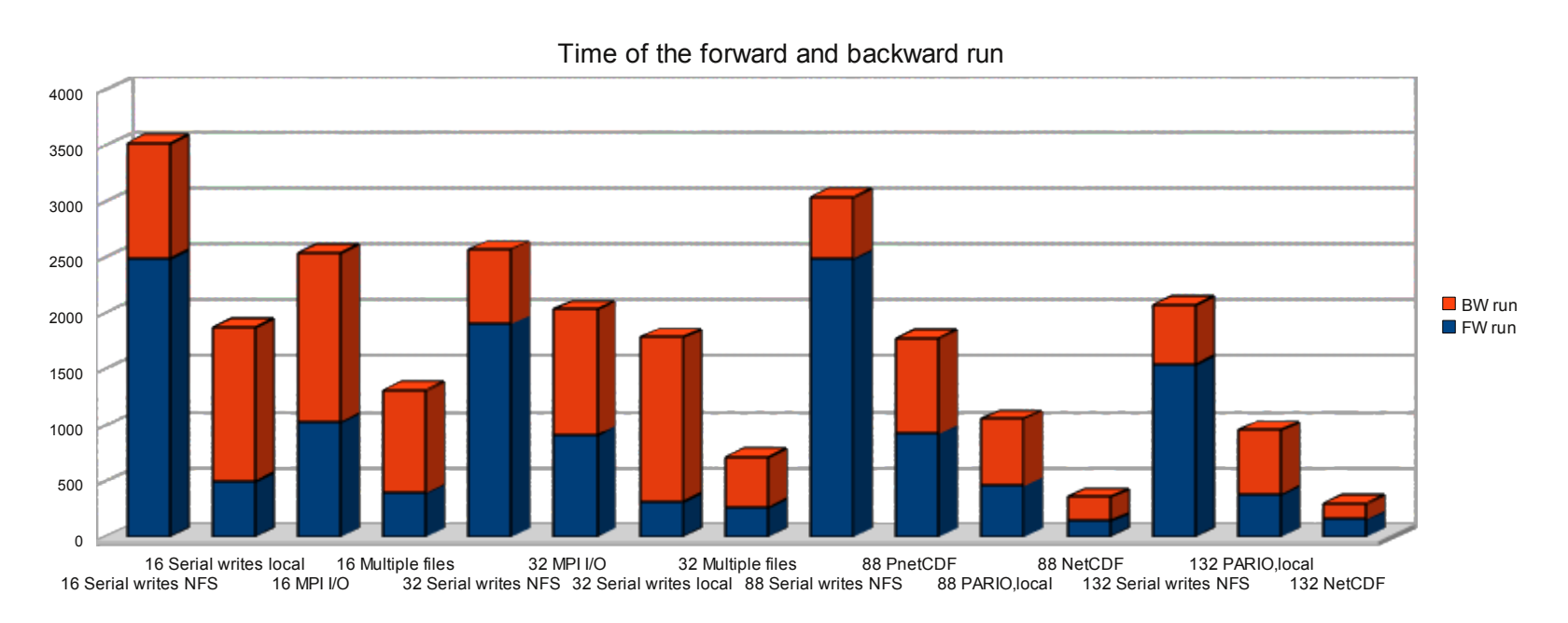


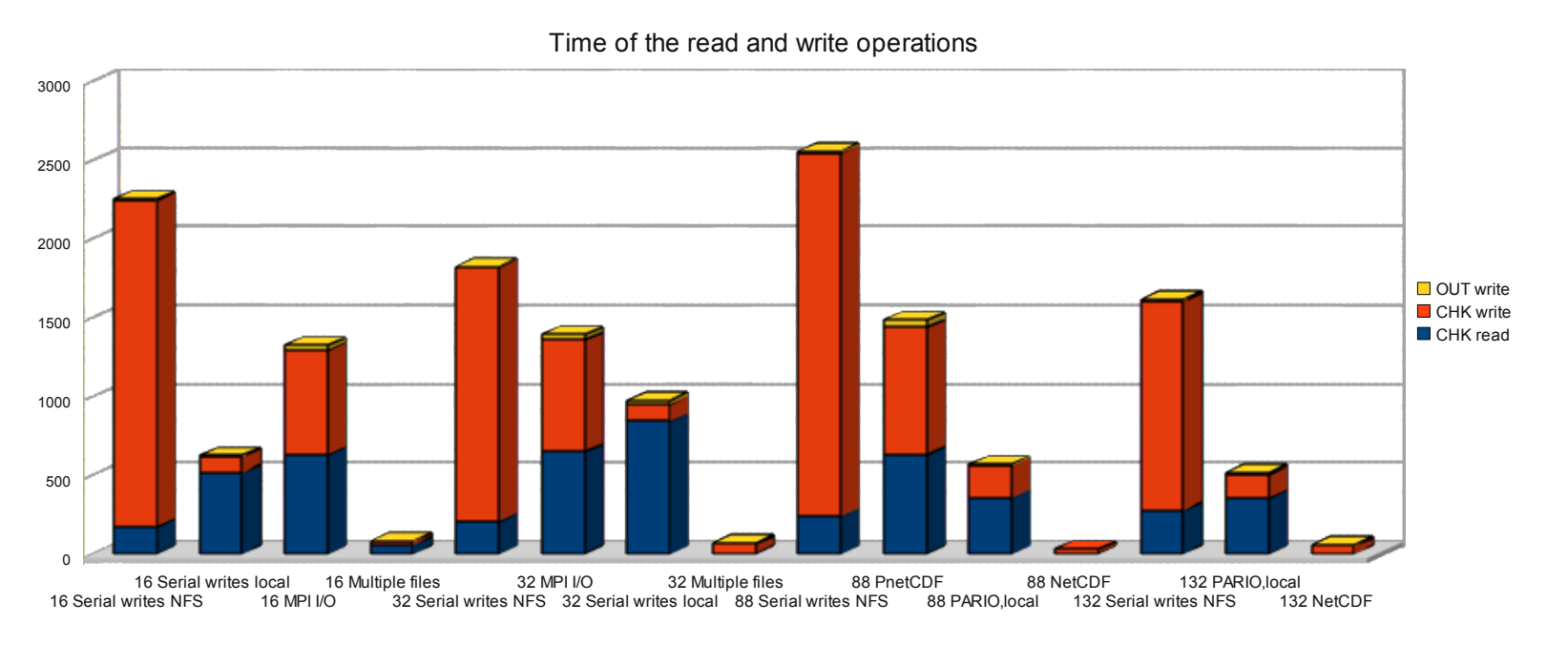
Processor time distribution



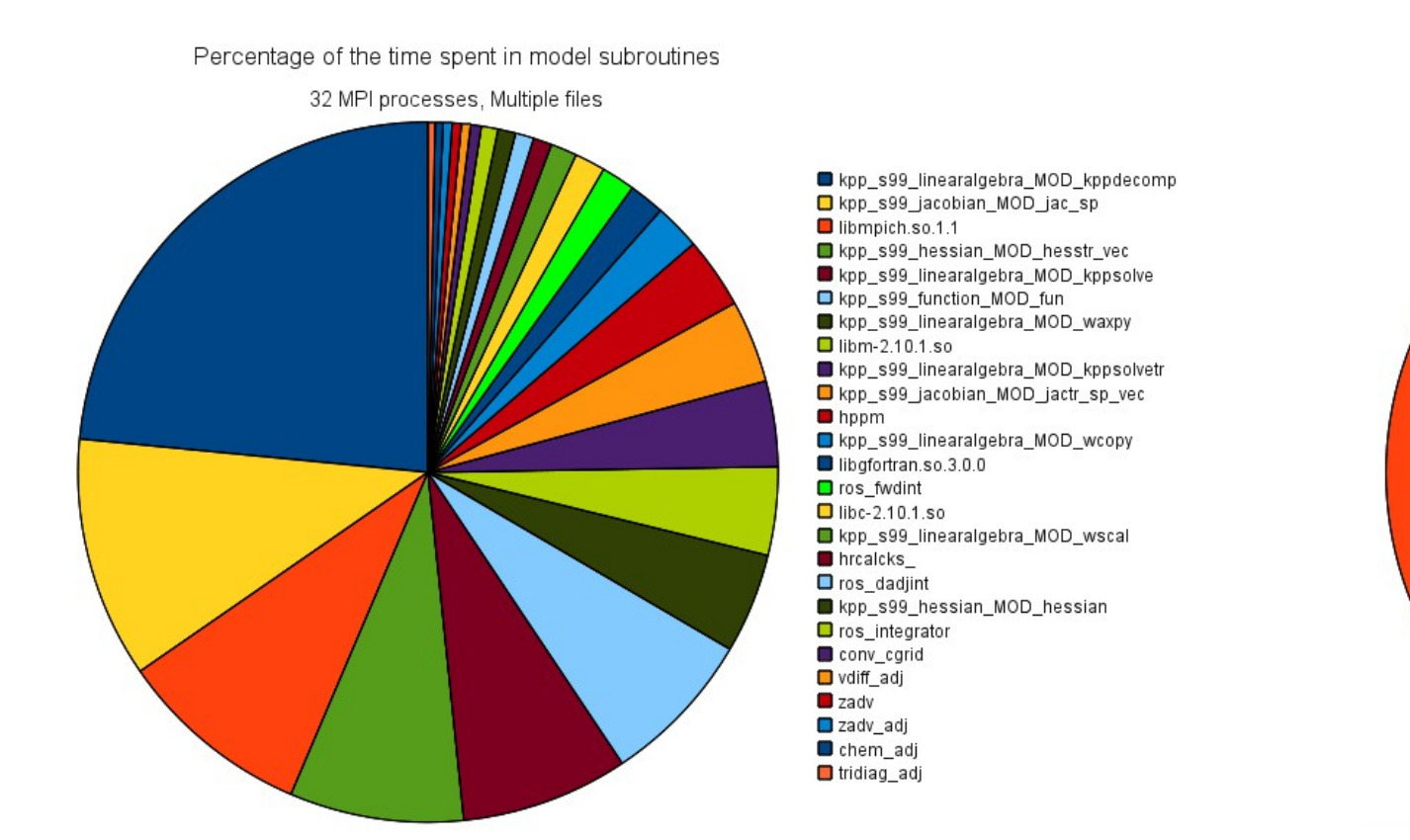
16 processes **132 processes**

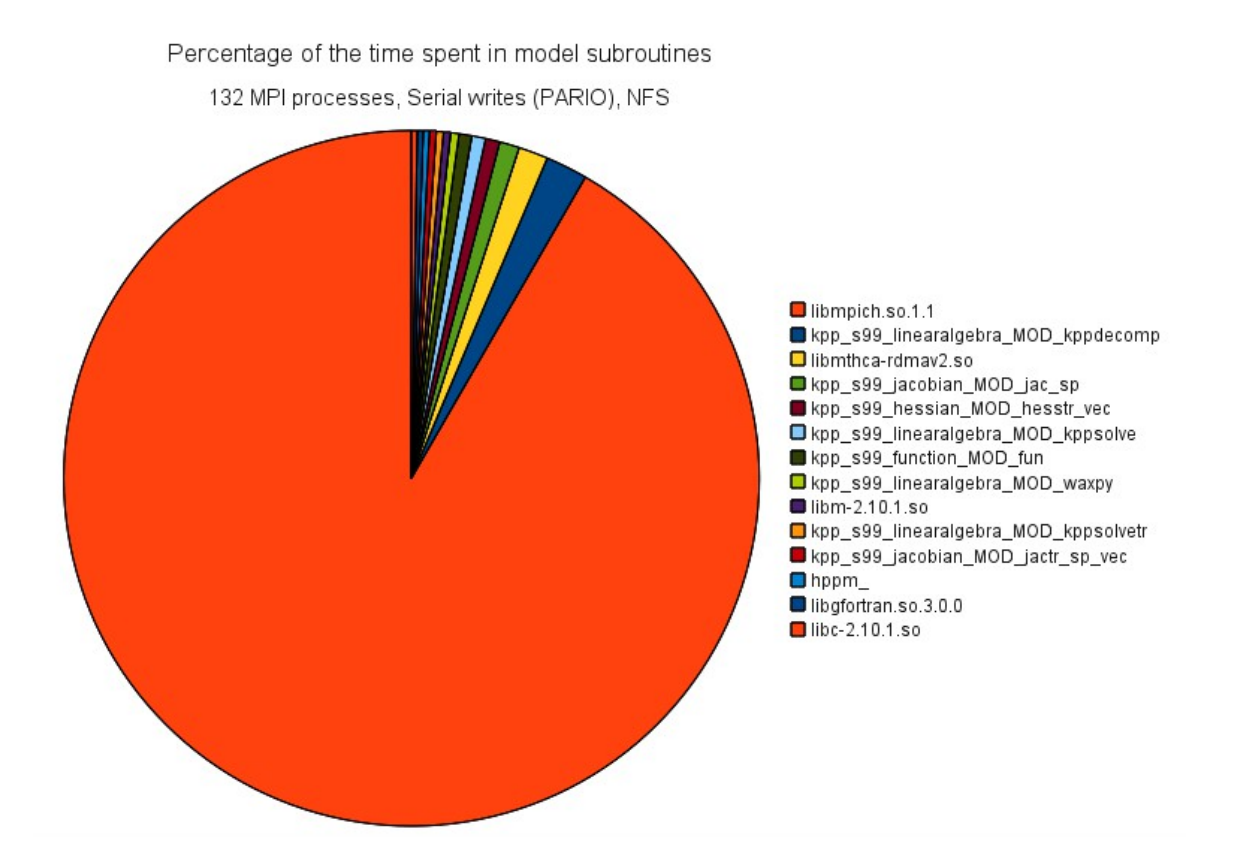
The upper two graphs show the total time of forward and backward run of the model and the parallel efficiency of the CMAQ adjoint model in different tested configurations. The two bar charts below show the distribution of total time between forward and backward run and the time spent in procedures reading and writing checkpoint and output files.
The right series of the graphs demonstrates percentage of the processor time spent in the model and in particular system modules. The results have been obtained by Linux system profiler *oprof* and they are averaged over processors of the cluster nodes. The graphs at bottom right show in more detail the time spent in particular model subroutines and system parts for two selected scenarios.





*Pie charts legend: **4dvar** – CMAQ adjoint model, **libmpich.so.1.1** – MPI implementation MVAPICH2, **libmthca-rdmav2.so (libmlx4-rdmav2.so)** – user space part of the driver of the Infiniband card Mellanox Infinihost III Ex (ConnectX), **libm-2.10.1.so** – system mathematical library, **libc-2.10.1.so** – system glibc library, **libgfortran.so.3.0.0** – Fortran runtime, **libpthread-2.10.1.so** – system POSIX threads library.*





Conclusions

Performed tests show substantial influence of checkpointing on parallel efficiency of the CMAQ adjoint model. We were able to reach much faster execution by employing alternative I/O strategies. "Multiple files" strategy is in general the fastest and the computation is 3-10x faster than with the original implementation of I/O, depending on particular hardware and test case. "Multiple files" strategy also allows to obtain benefits from parallelization on Ethernet interconnects for more than 16 processes, in contrary to the original I/O. The tests show substantial benefits from Infiniband interconnection with RDMA based MPI transfers when the number of processors is large. On the other hand, in all tested scenarios the parallel efficiency for 132 MPI processes does not exceed 50%. This degradation of parallel efficiency may be attributed to other bottlenecks in model parallelization and requires further investigation.

Our implementation of the CMAQ adjoint I/O operations is modular. This allows the user to choose the strategy which fits best to the type of his application and to the kind of his HW configuration.

Acknowledgement This work was supported by the grant No. SP/1a4/107/07 of the Ministry of Environment of the Czech Rep., by the COST Action ES0602: "Towards a European Network on Chemical Weather Forecasting and Information Systems" and by the Institutional research plan AV02 10300504 Computer Science for the Information Society "Models, Algorithms, Applications".