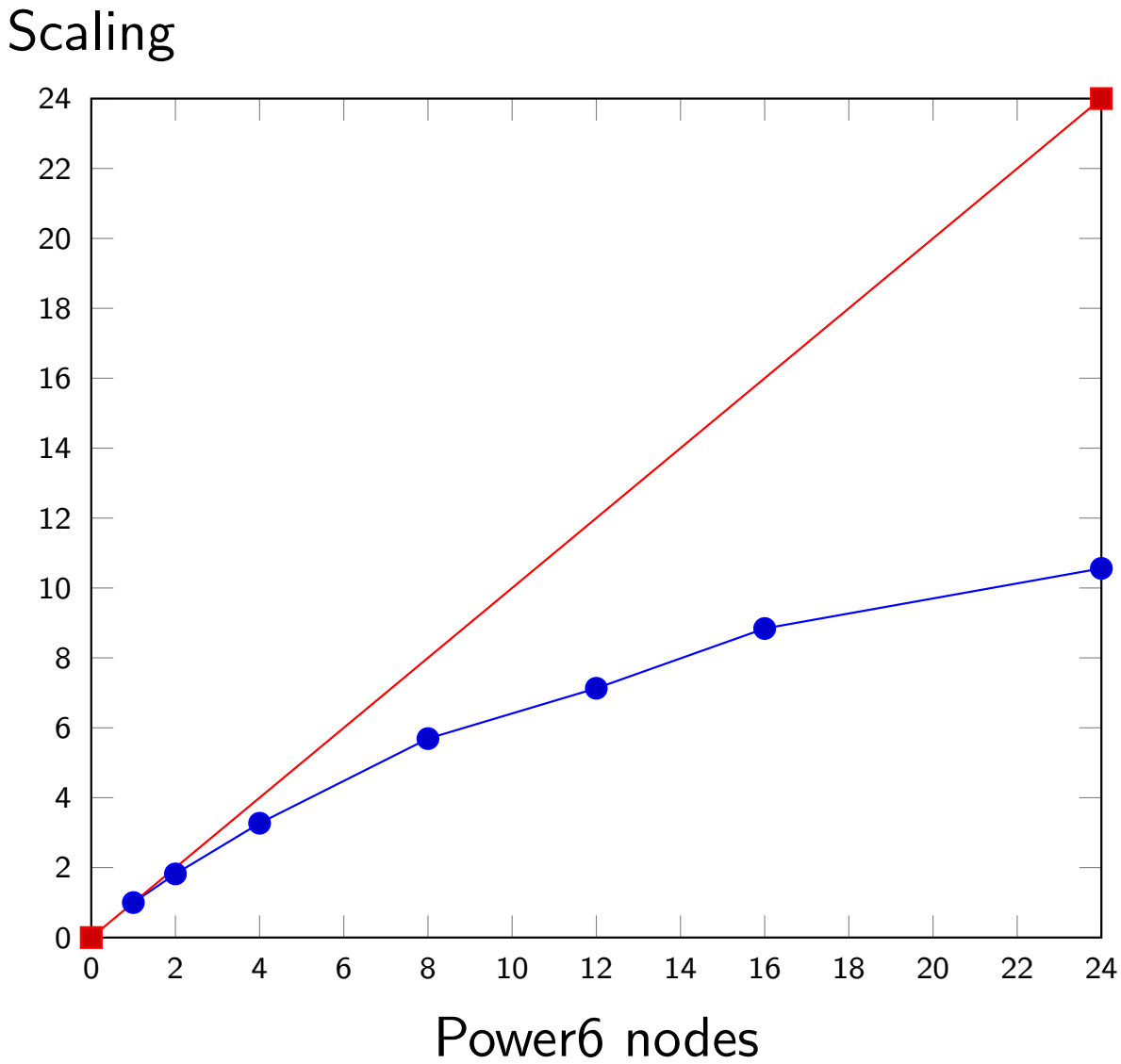
Parallel I/O for Earth System Modelling

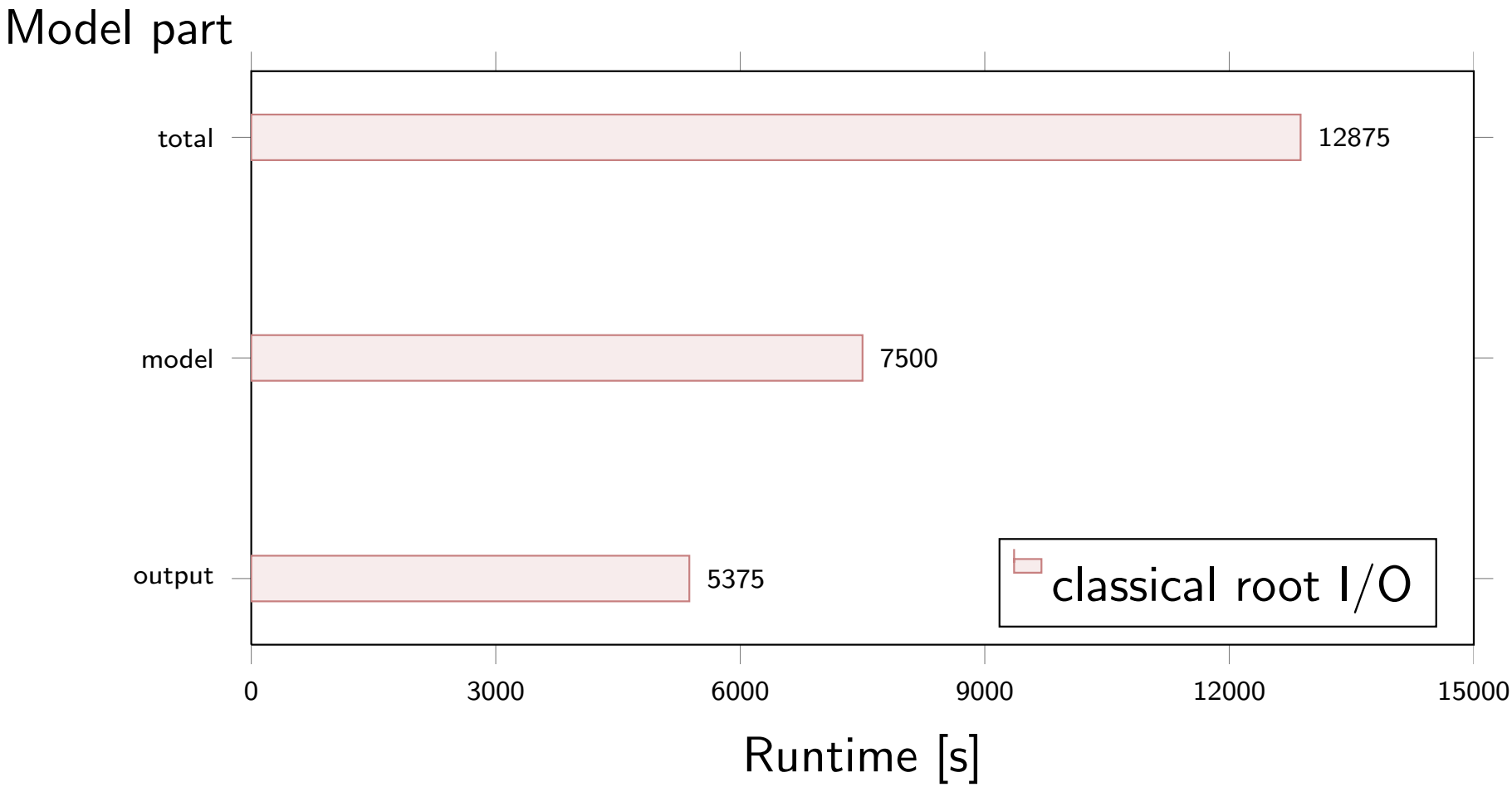
Luis Kornblueh, Deike Kleberg and Uwe Schulzweida
Max Planck Institute for Meteorology
supported by
Mathias Pütz, CRAY
Christoph Pospiech, IBM
Thomas Jahns, Moritz Hanke, Jörg Behrens, and Mathis Rosenhauer, DKRZ

October 3, 2012

Scaling optimization required



Classical root I/O explains scaling



Atmospheric and Oceanographic Data

Constraints

- Large amounts of data
- Comparable small spatial extent ($O(2, \ldots, 4)$)
- Large Time extent ($O(4, \ldots, 7)$)
- Large Number of variables ($O(2, \ldots, 3)$)

... continued ...

Requirements

- long term metadata and data storage
- standardization
- compression

Solutions

- WMO GRIB standard
- lowest entropy data subsampling
- two stage compression: lossy entropy based and lossless compression of resulted *image* — metadata uncompressed!

I/O improvements possible

Improvement by additional packing of the BDS data

Resolution Source	GRIBZip2d Frauenhofer	grib-szip MPI-M	gzip (external) GNU
T42 L19	2.32	2.13	2.06
T63 L31	1.85	1.78	1.35
T106 L60	5.17	4.75	3.81
T213 L31	3.09	3.06	2.41
mean	3.03	2.93	2.15

I/O improvements possible

Improvement by additional packing of the BDS data

Resolution Source	GRIBZip2d Frauenhofer	grib-szip MPI-M	gzip (external) GNU
T42 L19	2.32	2.13	2.06
T63 L31	1.85	1.78	1.35
T106 L60	5.17	4.75	3.81
T213 L31	3.09	3.06	2.41
mean	3.03	2.93	2.15

Remark 1

netCDF stores 4 byte, grib in average 2 byte — compression ratio given with respect to the later.

Remark 2

szip has a patent and copyright issue. We (most work by Mathis Rosenhauer) reimplemented the scheme from the CCSDS reference with the extensions from May 2012. The license is the BSD license now!

Solution strategy

1. decompose I/O in a way that all variables are distributed over the collector/concentrator PEs (I/O PEs)

Solution strategy

1. decompose I/O in a way that all variables are distributed over the collector/concentrator PEs (I/O PEs)
2. store each compute PEs data in buffers to be collected by collector PEs (I/O PEs) — buffer should reside in RDMA capable memory areas

Solution strategy

1. decompose I/O in a way that all variables are distributed over the collector/concentrator PEs (I/O PEs)
2. store each compute PEs data in buffers to be collected by collector PEs (I/O PEs) — buffer should reside in RDMA capable memory areas
3. instead of doing I/O, copy data to buffer and continue simulation

Solution strategy

1. decompose I/O in a way that all variables are distributed over the collector/concentrator PEs (I/O PEs)
2. store each compute PEs data in buffers to be collected by collector PEs (I/O PEs) — buffer should reside in RDMA capable memory areas
3. instead of doing I/O, copy data to buffer and continue simulation
4. collectors collect their respectable data (gather) via one-sided (RDMA based) MPI calls and do the transpose

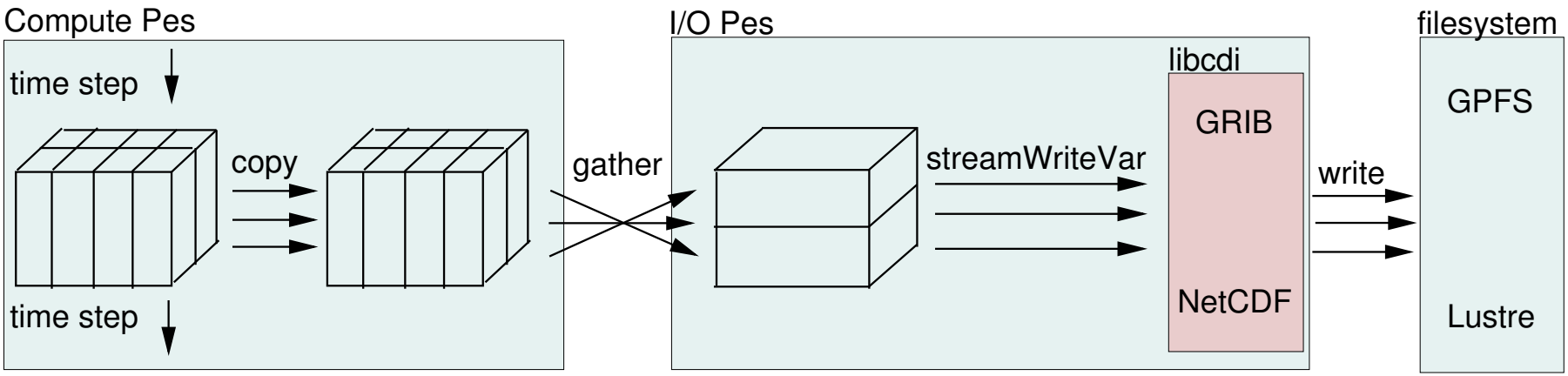
Solution strategy

1. decompose I/O in a way that all variables are distributed over the collector/concentrator PEs (I/O PEs)
2. store each compute PEs data in buffers to be collected by collector PEs (I/O PEs) — buffer should reside in RDMA capable memory areas
3. instead of doing I/O, copy data to buffer and continue simulation
4. collectors collect their respectable data (gather) via one-sided (RDMA based) MPI calls and do the transpose
5. compress each individual record

Solution strategy

1. decompose I/O in a way that all variables are distributed over the collector/concentrator PEs (I/O PEs)
2. store each compute PEs data in buffers to be collected by collector PEs (I/O PEs) — buffer should reside in RDMA capable memory areas
3. instead of doing I/O, copy data to buffer and continue simulation
4. collectors collect their respectable data (gather) via one-sided (RDMA based) MPI calls and do the transpose
5. compress each individual record
6. write ... sort of

File writing in ECHAM based on cdi-pio



- After calculating one I/O timestep the compute processes copy their data to a buffer and go on calculating till the next I/O timestep.
- I/O processes fetch the data using MPI one sided communication.
- Gather and transpose of the data is based on callback routines supplied by the model.

Most important property

Compute processes are not disturbed by file writing.

Known difficulties

- single offload step requires large memory on offload-node (requires eventually changes for Linux cluster and Cray XT architecture type machines, and maybe for IBM BlueGene)
- generates network jitter (RMA access to all compute nodes concurrent with computing nodes internal communication)
- filesystem jitter due to system bottlenecks (total bandwidth 30 GB/s, 2 GB/s per node, but 256 nodes)

What to optimize?

Search strategy

1. get to know your systems I/O capabilities!

What to optimize?

Search strategy

1. get to know your systems I/O capabilities!
2. measure the I/O bandwidth achieved

What to optimize?

Search strategy

1. get to know your systems I/O capabilities!
2. measure the I/O bandwidth achieved
3. build a test case for your I/O library

What to optimize?

Search strategy

1. get to know your systems I/O capabilities!
2. measure the I/O bandwidth achieved
3. build a test case for your I/O library
4. *profile* your testcase

What to optimize?

Search strategy

1. get to know your systems I/O capabilities!
2. measure the I/O bandwidth achieved
3. build a test case for your I/O library
4. *profile* your testcase
5. track down to *offending* level

What to optimize?

Search strategy

1. get to know your systems I/O capabilities!
2. measure the I/O bandwidth achieved
3. build a test case for your I/O library
4. *profile* your testcase
5. track down to *offending* level
6. check selected counters for *offending* code part

I/O capabilities

An example: DKRZ

- 256 compute nodes, 12 file server, 6 PB filesystem, 4 HPSS server, 60 PB tape archive
- total I/O bandwidth to disk 30 GB/s
- per node max. I/O bandwidth 2 GB/s
- 1600 users
- max. 96 post-processing jobs
- unknown number of production jobs

Offending code parts

Legacy in libraries

- portable double to float conversion (199x) taking into account CRAY, VAX , IBM, and IEEE FP formats
- C max/min search loop
- encoding kernel
  - mixed floating point/integer operation
  - very small number of operations

Analysis for optimization strategy

Caution: Assembler reading required

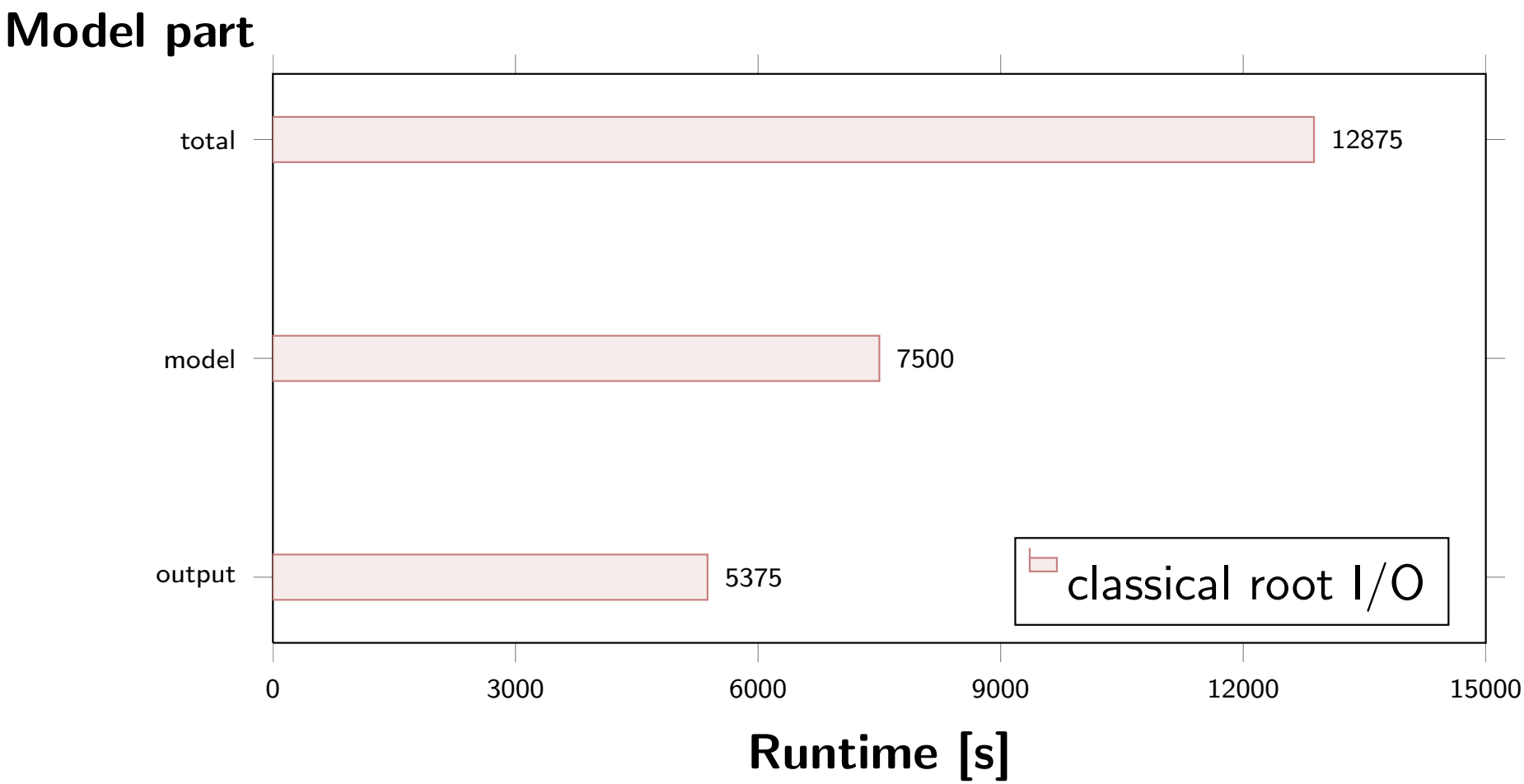
- understand roughly how your CPU works
- need to *read* Assembler (not that bad, feels like using a pocket calculator), you get an idea what the compiler is doing
- compare code of different optimization levels
- try to find the patterns, you would expect for fast code

An example

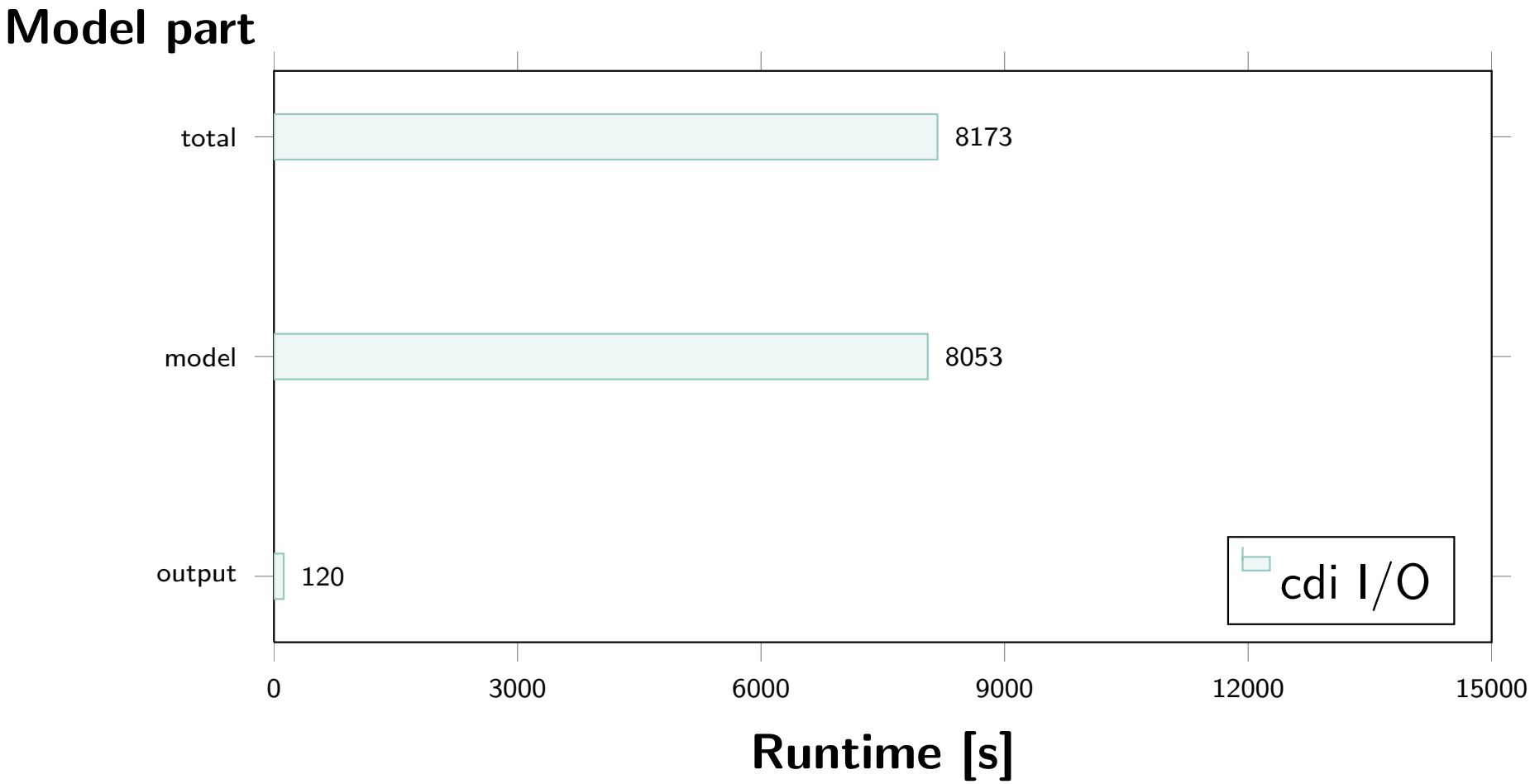
```
/* datatype information only */
long datasize
double data[datasize];
unsigned char *lGrib;
long i, z;
unsigned long ival;
double dval, reference, factor;

/* offending code */
for ( i = 0; i < datasize; i++ )
  {
    dval = ((data[i] - reference) * factor + 0.5);
    ival = (unsigned long) dval;
    lGrib[z  ] = ival >>  8;
    lGrib[z+1] = ival;
    z += 2;
  }
```

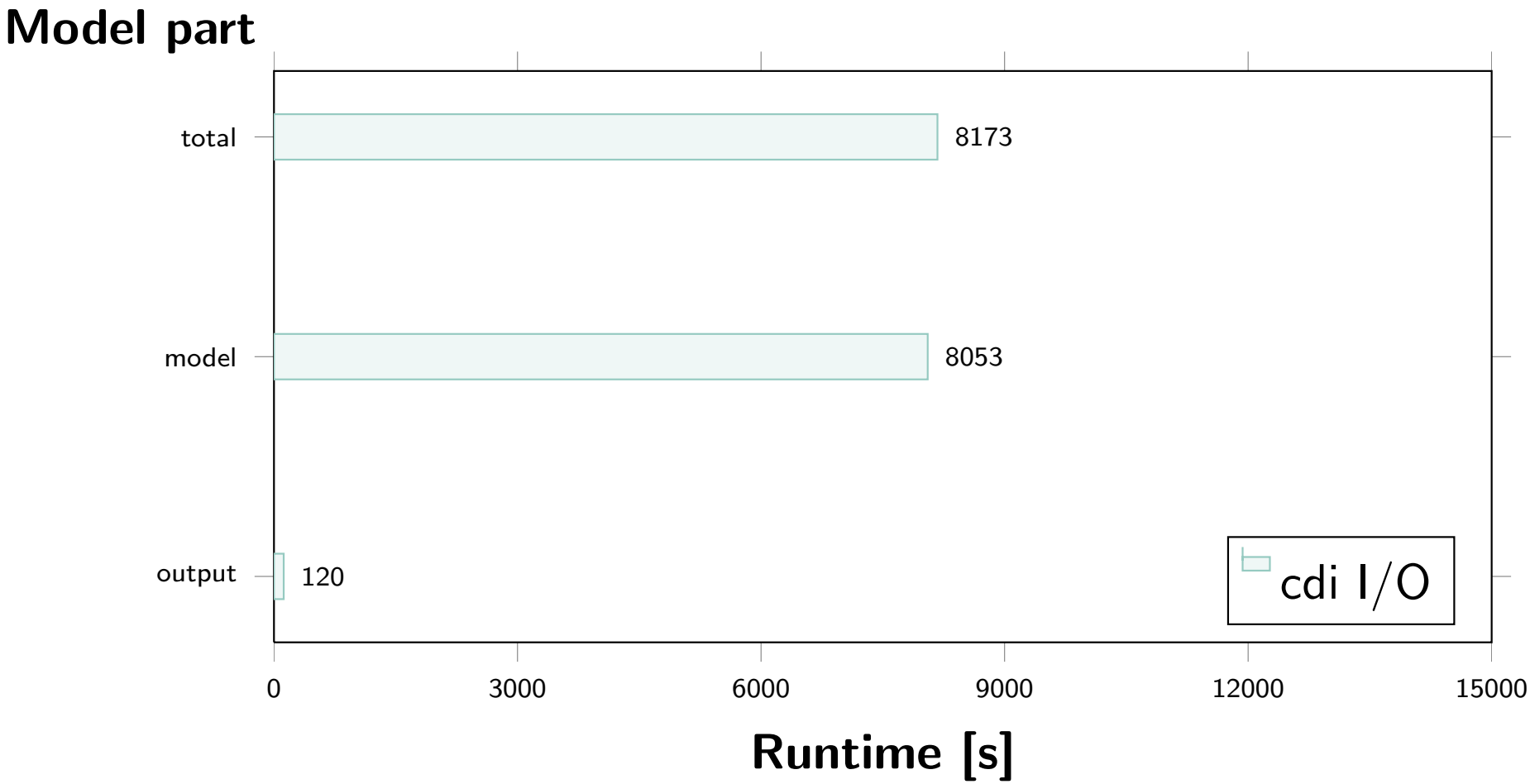
Classical root I/O



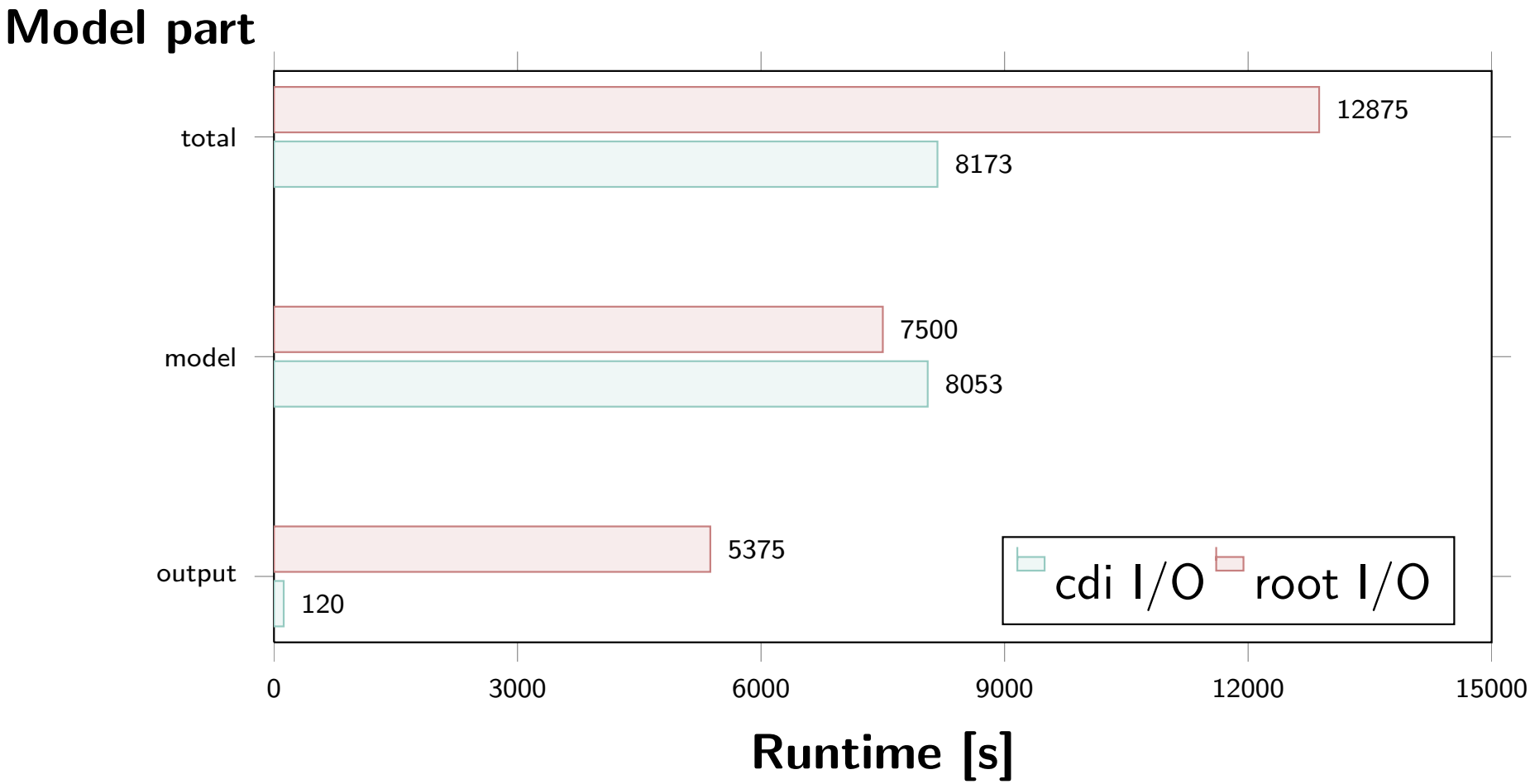
cdi based asynchronous parallel I/O



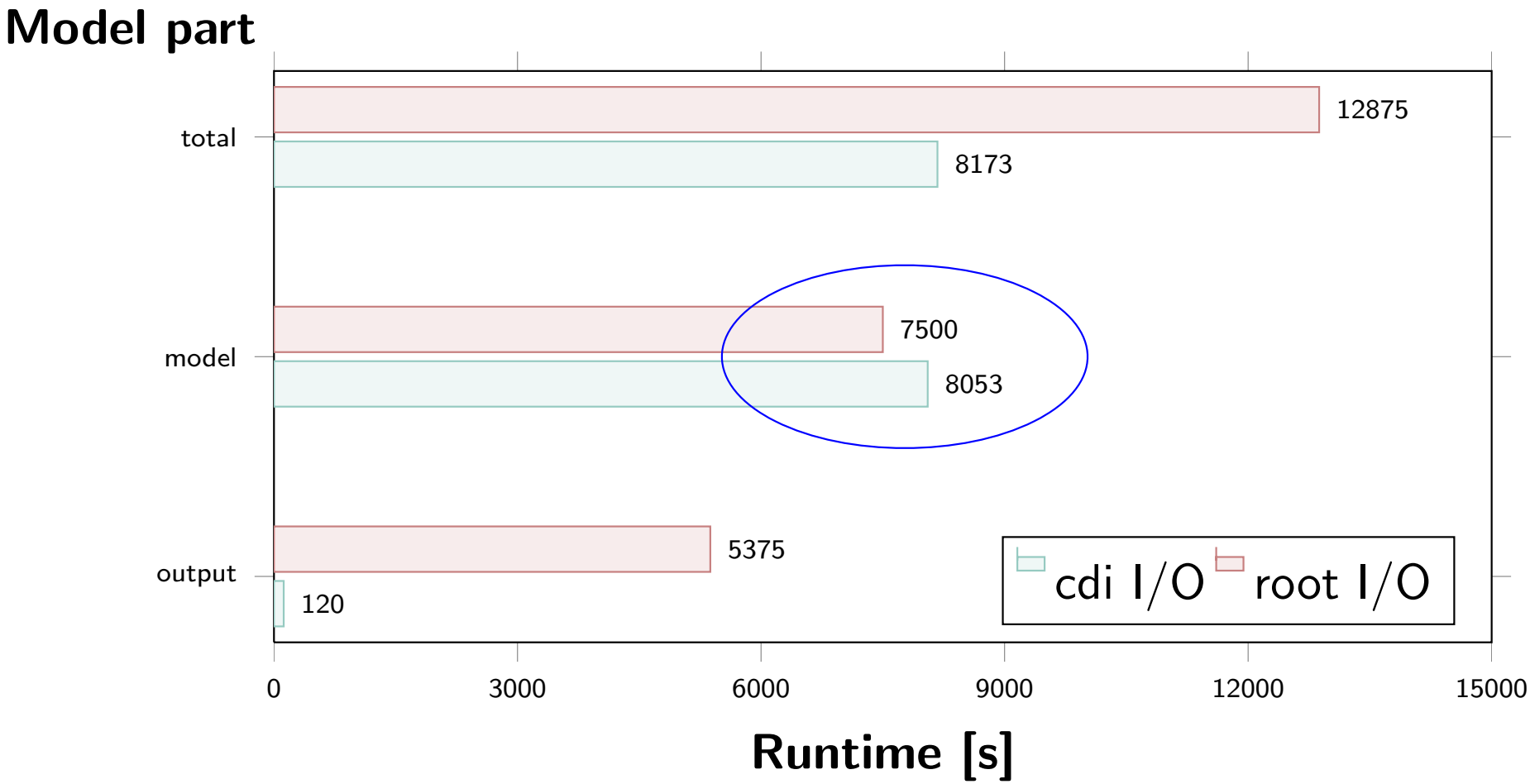
Compare


Model part
total
model
output
8173
8053
120
cdi I/O
0
3000
6000
9000
12000
15000
Runtime [s]

Compare


Model part
total
model
output
12875
8173
7500
8053
5375
120
cdi I/O
root I/O
0
3000
6000
9000
12000
15000
Runtime [s]

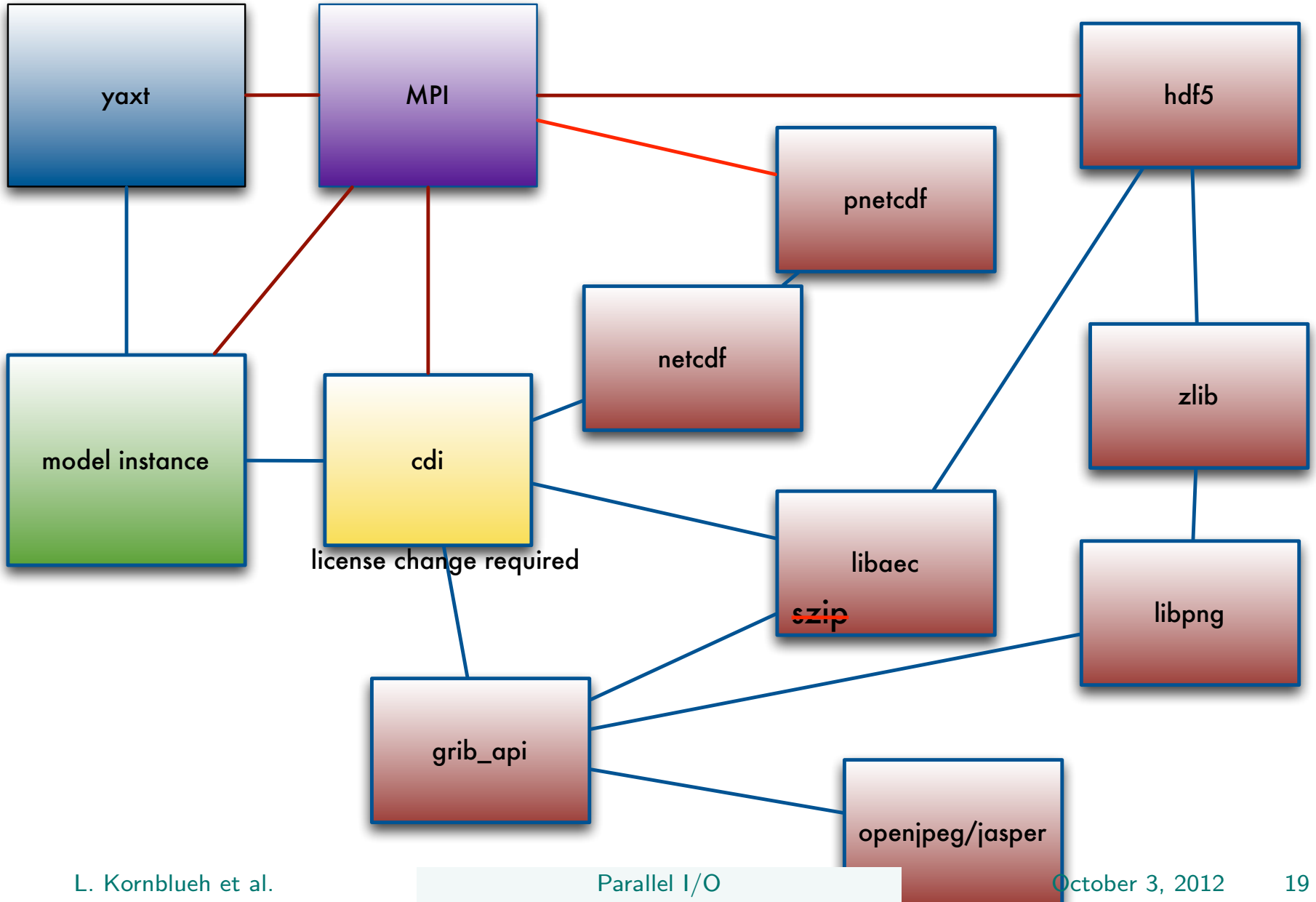
Compare


Model part
total
model
output
12875
8173
7500
8053
5375
120
cdi I/O
root I/O
0
3000
6000
9000
12000
15000
Runtime [s]

What next?

- Scaling experiments
- Further optimizations of details
- Implementation in all MPIM models
- Finish Implementation in EC-Earth
- Optimize GRIB decoding
- Optimize libaec
- Intensivy collaboration on solving issues in each single component
- Get library zoo manageable

Library zoo


yaxt
MPI
hdf5
pnetcdf
netcdf
zlib
model instance
cdi
license change required
libaec
szip
libpng
grib_api
openjpeg/jasper

What is CDO ?

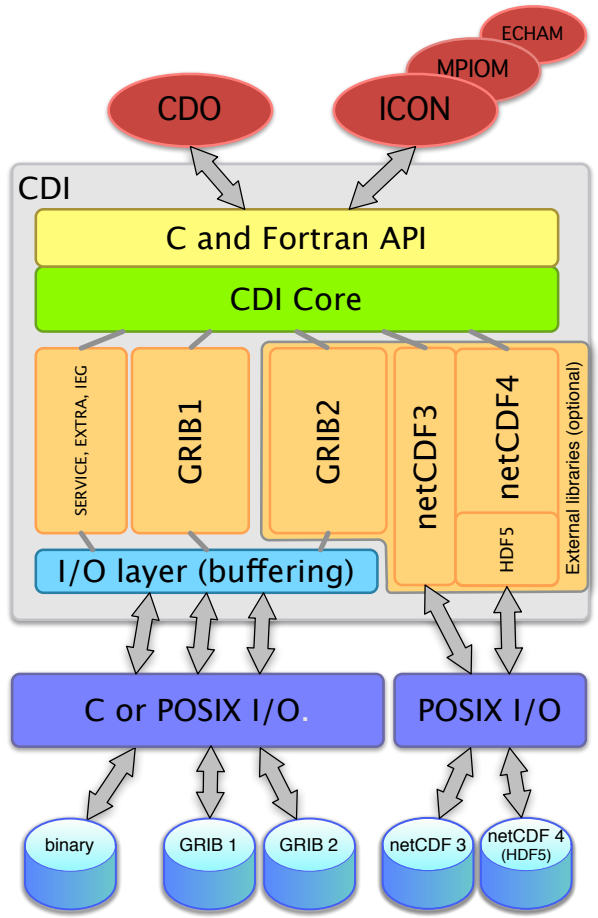
CDO is a collection of tools to process and analyze data from climate and NWP models.

- File format conversion: GRIB $\Leftrightarrow$ netCDF
- Interpolation between different grid types and resolution
- Portability (ANSI C99 with some POSIX extentions)
- Performance (fast processing of large datasets, muti-threaded)
- Modular design and easily extendable with new operators
- UNIX command line interface
- Tested on many UNIX/Linux systems, Cygwin, and MacOS-X

Data I/O Interface

Part of CDO is the I/O interface CDI (Climate Data Interface) which it shares with all major MPI-M climate models.

- GRIB1 via CGRIBEX (MPI-M)
- GRIB2 via GRIB_API (ECMWF)
- netCDF, CF-convention (UNIDATA)
- SERVICE, EXTRA, IEG (MPI-M binary formats)



GRIB support includes highly efficient, fast compression algorithms.

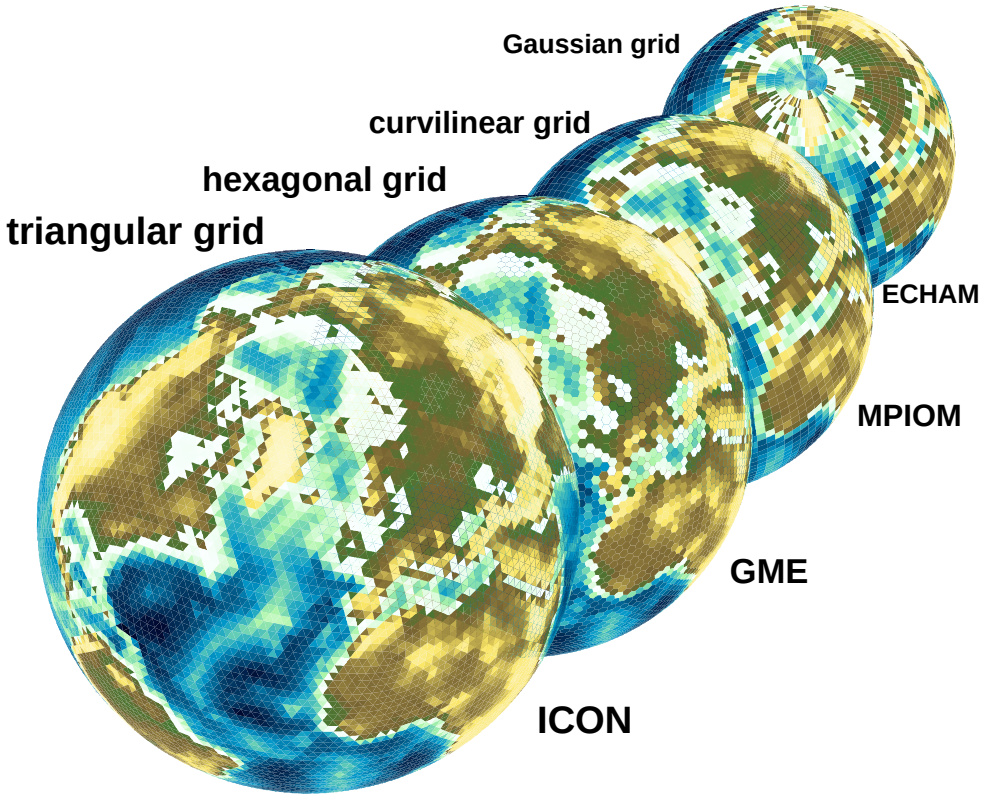
Available Operators

CDO provides more than 400 operators which can be pipelined on thread level. CPU time intensive operators are OpenMP parallelized.

Main categories	Description
File information	Print information about datasets
File operations	Copy, split and merge datasets
Selection	Select parts of a dataset
Comparision	Compare datasets
Modification	Modify data and metadata
Arithmetic	Arithmeticly process datasets
Statistical values	Ensemble, field, vertical and time statistic
Interpolation	Horizontal, vertical and time interpolation
Import/Export	HDF5, binary, ASCII
Climate indices	ECA Indices

Supported Grids

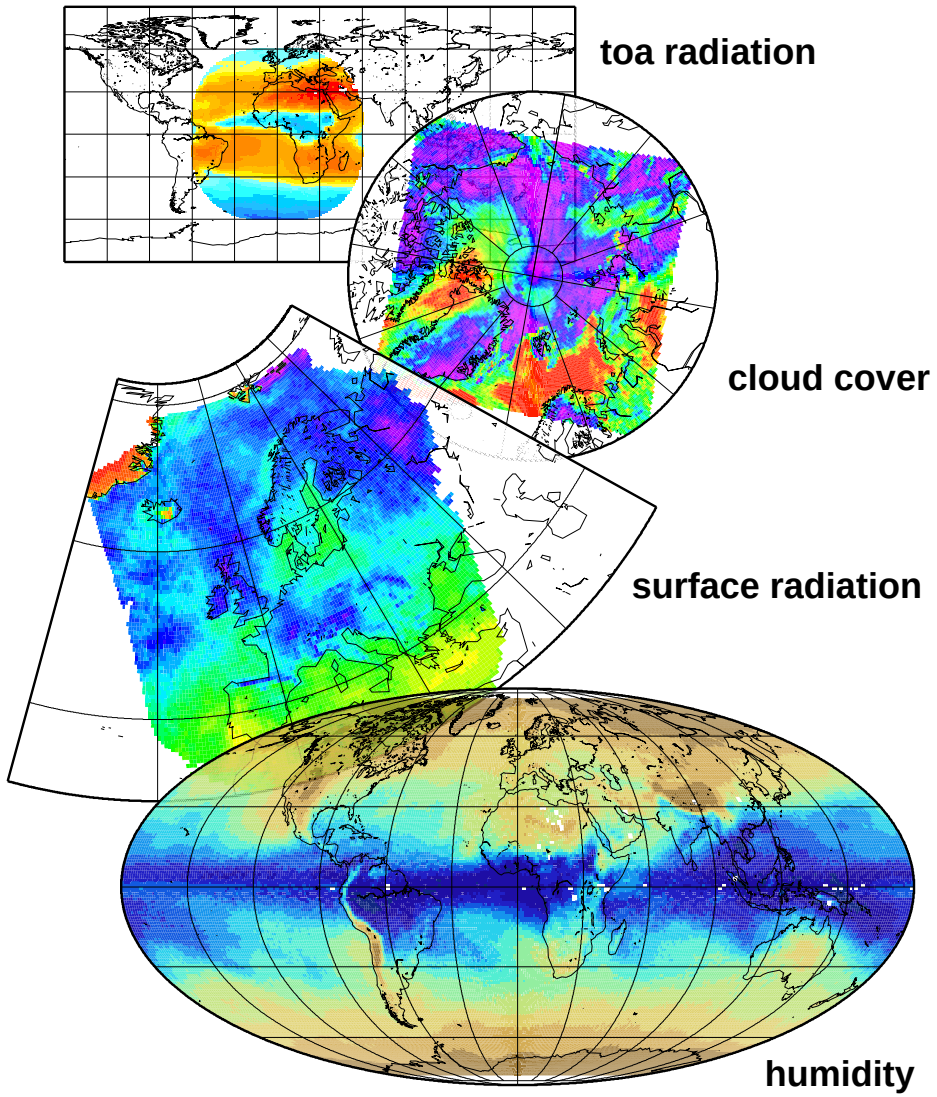
A large set of grids is supported including spectral- and Fourier-coefficients. Gaussian grids, regular and rotated lat-lon grids, conformal mapped quadrilateral grids, and finally general unstructured grids.



All major models world wide are supported (COSMOS, CLM, ECHAM, GME, HIRLAM, ICON, IFS, MPIOM, NEMO, and REMO — only to mention those used mostly in Germany).

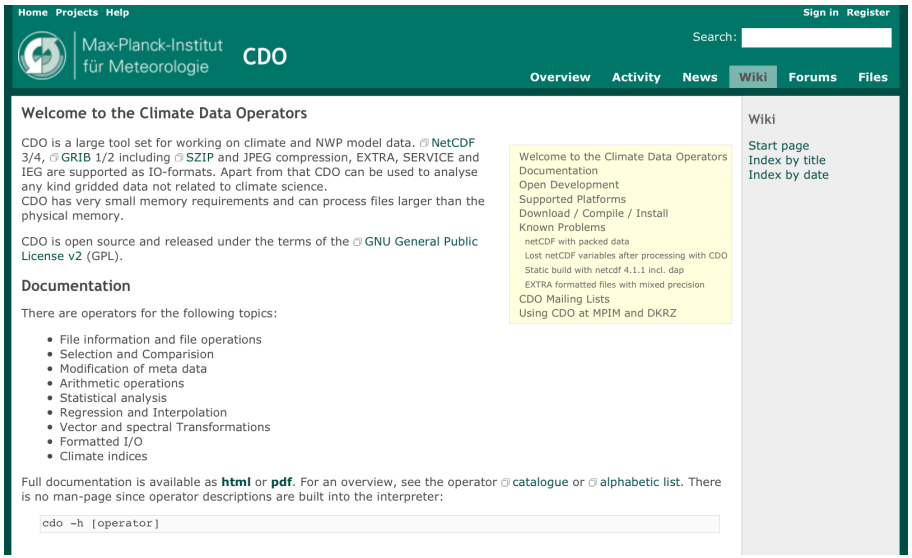
Satellite-data Support

EUMETSAT's Climate Monitoring Satellite Application Facility provides satellite-derived geophysical parameter for climate monitoring. Data sets contain several cloud parameters, surface albedo, radiation fluxes, temperatur and humidity profiles. These products are stored in HDF5. DWD has funded an import CDO operator `import_cmsaf`.



Community Support

The rapidly increasing number of CDO installations and users create a very high demand of support. A fully featured development platform is available to support the community. The CDO community page was funded by the European Commission infrastructure project IS-ENES.



- User wiki
- Documentation
- Bug tracking system
- User forums
- Download area
- Repository access

http://code.zmaw.de/projects/cdo

What comes next?

- Web Services (EUDAT, EU funded)
- Script-language interface (Python, Perl, Ruby, ...)
- Add simple standardized plotting capabilities (Magics++, ECMWF)
- Add more functionalities
- Performance improvements (ENES funded)
- Parallel asynchronous CDI, Deike Kleberg and Luis Kornblueh (ScaLES, BMBF funded)