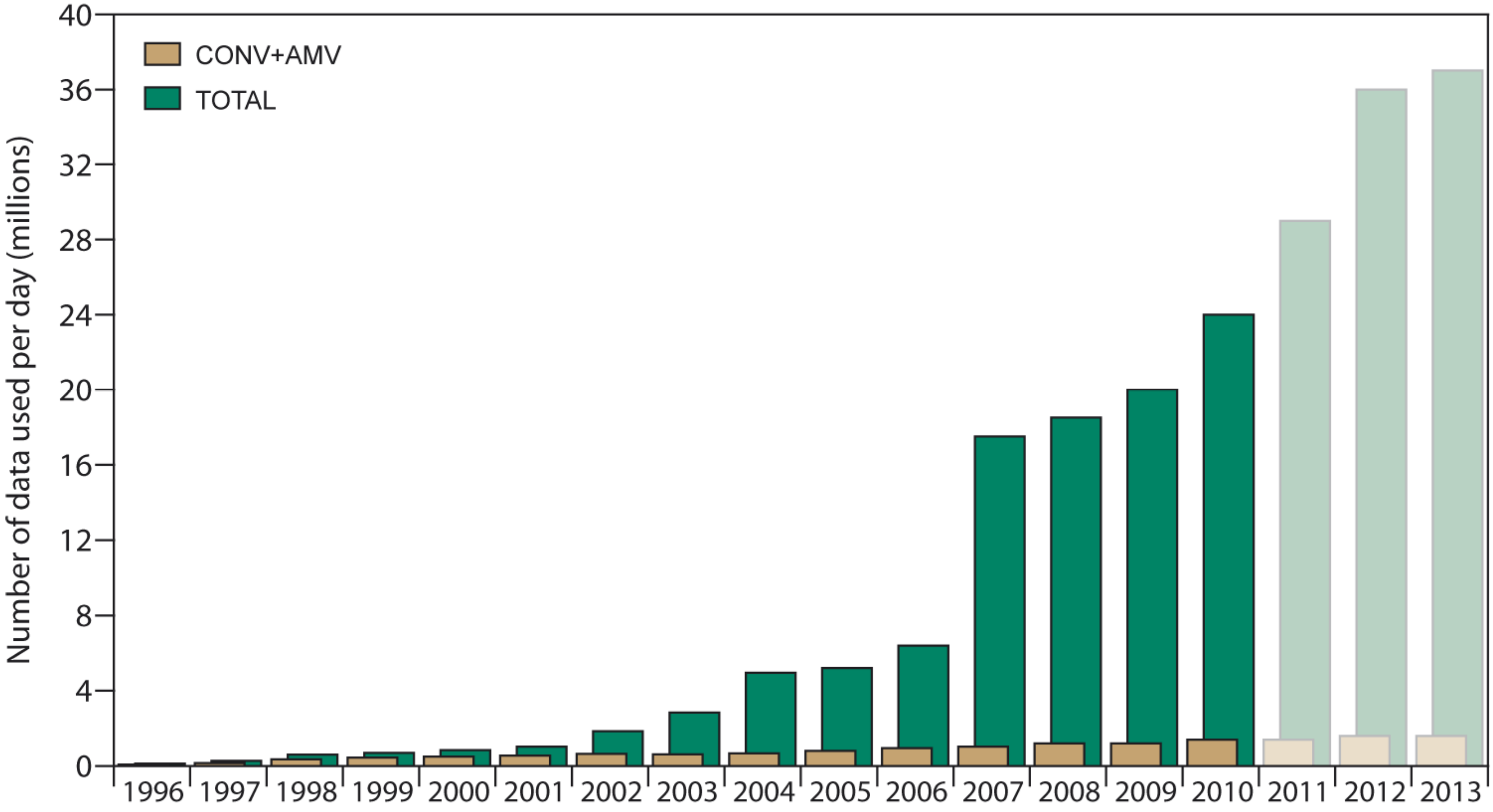
ODB: past, present and future

Or ODB scalability over the past decade...

Scalability: a new challenge for handling observations in meteorological models?



There are two constants about content and data: it will change, and it will grow…

ECMWF

Outline

- What you need to know about ODB…
- Current observation data flow at ECMWF
- ODB usage in our 4D-Var system
- ODB debut
- Switching from vector to scalar architecture
- ODB IO strategy
- Any hope for the future?
- Conclusions

ECMWF

What you need to know about ODB…

- **ODB stands for Observational DataBase**
- **Developments were started by *Sami Saarinen* in 1998 in replacement of the old CMA (Central Memory Array) file structure**
- **ODB is a hierarchical database, a format and a software library:**
  - With a data definition language to describe what data items belong to the database (and their data types and how they are related to each other)
  - And a query language ODB/SQL (subset of ANSI SQL) to query and return a subset of data which satisfies certain user specified conditions.
  - Data can be stored in a distributed fashion (by pools)
  - Managing, manipulating, and analyzing data can be done in parallel (MPI/OpenMP)
- **It has the option of being an *incore database*, but can be used as file based as well**

ECMWF

Example of an ODB database on disk

> ls ECMA.iasi

1/	141/	183/	218/	265/	43/	85/
107/	145/	193/	225/	266/	49/	97/
110/	15/	197/	239/	267/	56/	99/
113/	155/	211/	241/	272/	57/	
121/	164/	212/	25/	281/	71/	
127/	169/	217/	253/	29/	73/	

Pool directories

ECMA.iomap
ECMA.sch
IOASSIGN@
ECMA.IOASSIGN
ECMA.dd
ECMA.flags

Metadata

> ls ECMA.iasi/1

index	sat	radiance	modsurf	update_2
desc	poolmask	cloud_sink	surfemiss_body	update_3
errstat	body	hdr	update_1	timeslot_index

ECMWF

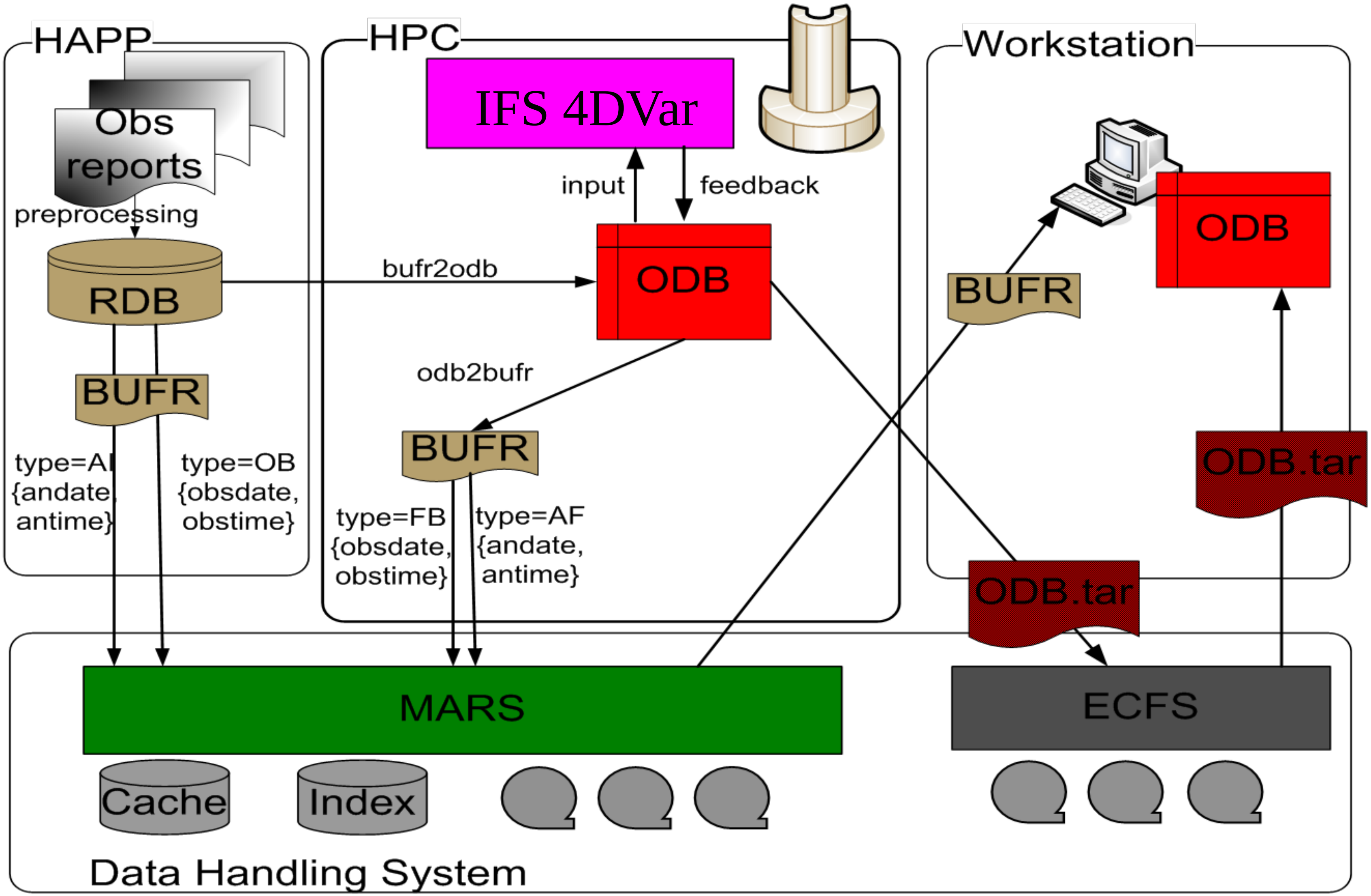
What is a table?

- **A table is a file containing a list of attributes such as *lat*, *lon*, *obsvalue*, *an_depar*, etc. Each of them has a meaningful and unique ODB name, with a short description, and with units or a range of possible values.**
- **We have about 800 different ODB columns defined in our databases but each observation group has its own list of valid ODB attributes (between 60 and 100)**

ODBAttributes < Main <...

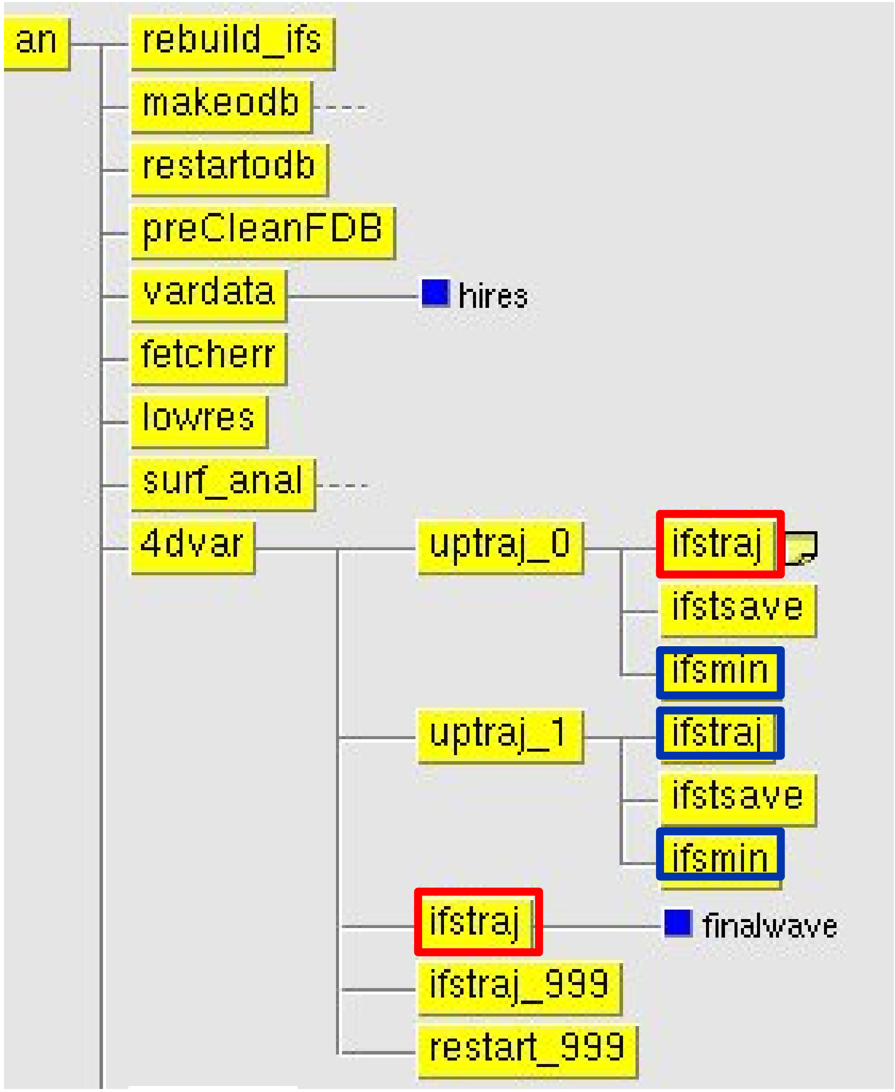
an_cwp	pk9real	Cloud liquid water path @ FG [kg m-2]
an_depar	pk9real	OBSERVED MINUS ANALYSED VALUE
an_iwp	pk9real	Cloud ice water path @ FG [kg m-2]
an_p19	pk9real	19 GHz normalised polarisation difference analysis
an_p37	pk9real	19 GHz normalised polarisation difference analysis
an_rain_rate	pk9real	Surface rain @ FG [mm h-1]
an_rttov_cld_fraction	pk9real	Cloud fraction used in RTTOV_SCATT [0-1]
an_rwp	pk9real	Rain water path @ FG [kg m-2]

Current observation data flow at ECMWF



ECMWF

ODB usage in our 4D-Var system



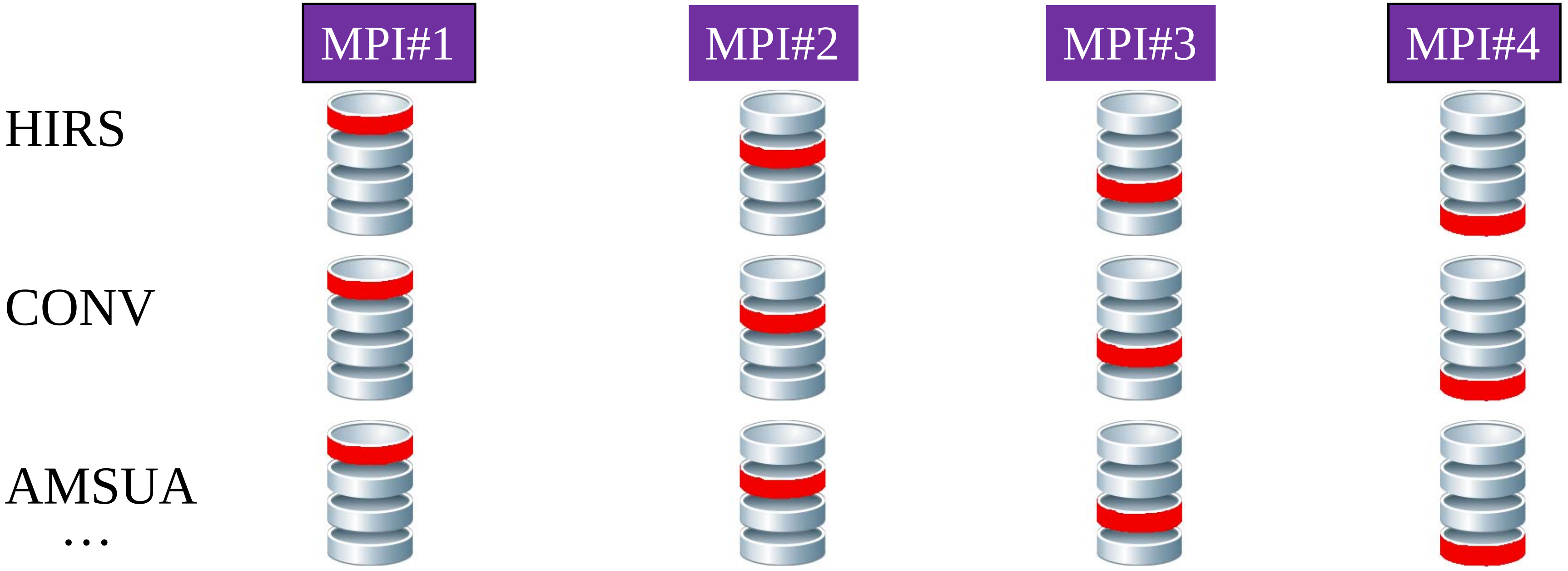
We use two main ODBs:

- **ECMA (Extended CMA):**
  - *All observations*
  - *About 25 ECMAs (one per Observation group)*
  - *≈ 2000 retrievals per thread in the first trajectory*
- **CCMA (Compressed CMA):**
  - *Active observations after IFS screening (< 10% of ECMAs)*
  - *≈ 5000 retrievals per thread in ifsmin*
- **Data randomly distributed among pools**
- **Both are *incore databases* to improve efficiency**

ECMWF

ODB debut

- ODB became operational on our VPP5000 in 2000 (CY22R3, T319, 50 vertical levels). Our 4DVar system was running on 16 MPI tasks.
- ODB IO strategy was fairly simple: each MPI task reads/writes a portion of database and owns ODB data pools in a round-Robin fashion



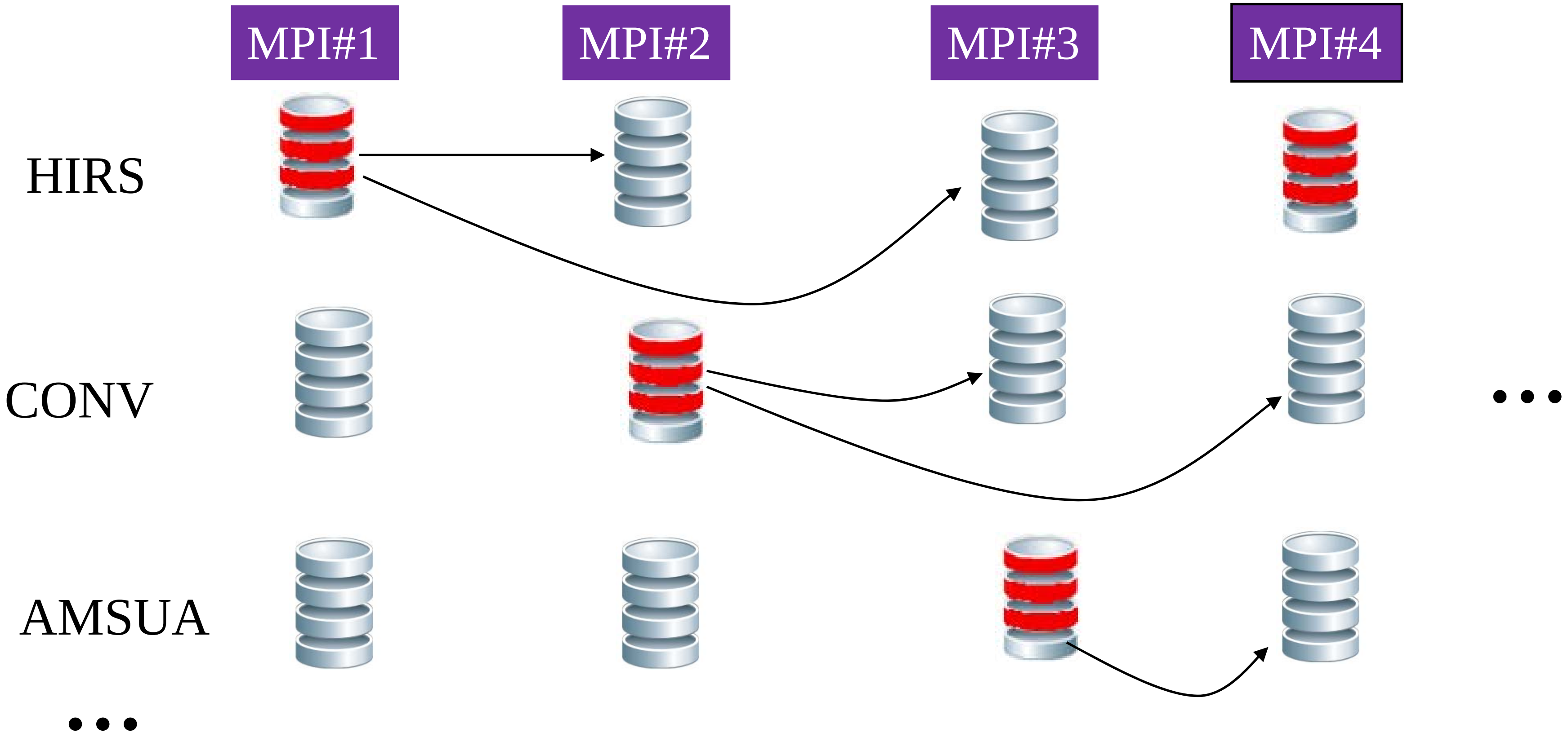
→ ODB retrievals scale well but this simple ODB IO strategy doesn't…

Switching from vector to scalar architecture

- **To improve performance and better scale on platforms with increasing number of processors:**
  - **Only a subset of pools is selected to perform I/O (read/write ODB on disk).**
  - **Similar files (tables) are then concatenated together (reduces the total number of files).**
  - **ODB I/O pools distribute data to other processors via MPI communications**
- **The number of I/O pools is fully configurable via environment variables**
  - **At least every ODB_IO_GRPSIZE -MPI-task performs I/O -- up to a certain file size limit (MB) defined by the parameter ODB_IO_FILESIZE**

Parallel I/O strategy

- **A loop over tables and for each table:**



- **Walltime goes from 663 s to 550 s for the first trajectory (15% improvement)**

ODB-I/O strategy: recent optimizations

- **It was first identified that MPI communications were costly when storing the database (because we write about 20GB…).**
- **John Hague has improved message passing involved when writing the database: we collect what has to be written and send/receive larger MPI messages.**
- **ODB_IO_FILESIZE=32, ODB_IO_GRPSIZE=$NPES_AN**
- **Message passing I/O *included* in the timings**

(T1279, 48 nodes)	STORE			STORE optimized
Step	WALLTIME (seconds)	Size (GB)	# of files	WALLTIME (seconds)
Traj_0	20.70	20.0	922	9.82

Cost for loading/storing ECMAs/CCMAs

(T1279, 48 nodes)	LOAD			STORE
Step	WALLTIME (seconds)	Size (GB)	# of files	WALLTIME (seconds)
Traj_0	6.33	2.45	166	9.82 + 2.25
Min_0 (T159)	1.71	1.8	88	2.59
Traj_1	2.45	1.9	91	2.29
Min_1 (T255)	1.75	2.0	94	2.46
Traj_2	2.38	2.1	97	3.58
Min_2 (T255)	2.48	2.2	99	3.96
Traj_3	22.33 + 7.66	19.4+2.3	928+102	17.76 + 1.55

- **ODB_IO_FILESIZE=32, ODB_IO_GRPSIZE=$NPES_AN**
- **Message passing I/O *included* in the timings**

traj_0: impact of ODB_IO_FILESIZE on ECMAs

ODB_IO_FILESIZE	LOAD WALLTIME (seconds)	# of files	STORE WALLTIME (seconds)	# of files
8	7.65	414	22.66	2514
16	6.45	240	25.97	1723
32	6.33	166	9.82	928
64	11.91	130	17.23	587
128	12.36	118	20.35	448

- **We load about 3 GB and store about 20 GB in the first trajectory**
- **Runs done with T1279 on 48 nodes**
- **ODB_IO_FILESIZE=32 is optimal on our current supercomputer (Power 6)**

ECMWF

Any hope for the future?

- The bottlenecks are the first and the last trajectories in our 4D-Var where ECMAs are involved.
- Poor performance of ODB I/O in traj_3 may show that we may have reached some limits…Tools to monitor I/Os like those developed by John and Oliver would help us to better understand what is going on.
- However, the best way to improve I/Os is to reduce I/Os… We need to better organise our databases (ECMA/CCMA) to avoid unecessary I/Os:
  - create readonly tables and use ODB_WRITE_TABLES
  - use ODB_CONSIDER_TABLES to load in memory only the necessary tables for a given step
- Change our strategy: do not try to load the entire database at the beginning or store the entire database at the end (i.e. try to overlap I/Os with computations)

New strategies for the first trajectory

- **The first trajectory is very expensive because of the screening (about 150 millions observations used): only 10% retained for the assimilation.**
- **Screening of observations may not be needed at high-resolution (Scientists have to tell us)**
- **Therefore, it may become cheaper to run several "trajectories", as soon as data is available to eliminate as much data as possible when we start our 4D-Var.**

Other scope for improvement

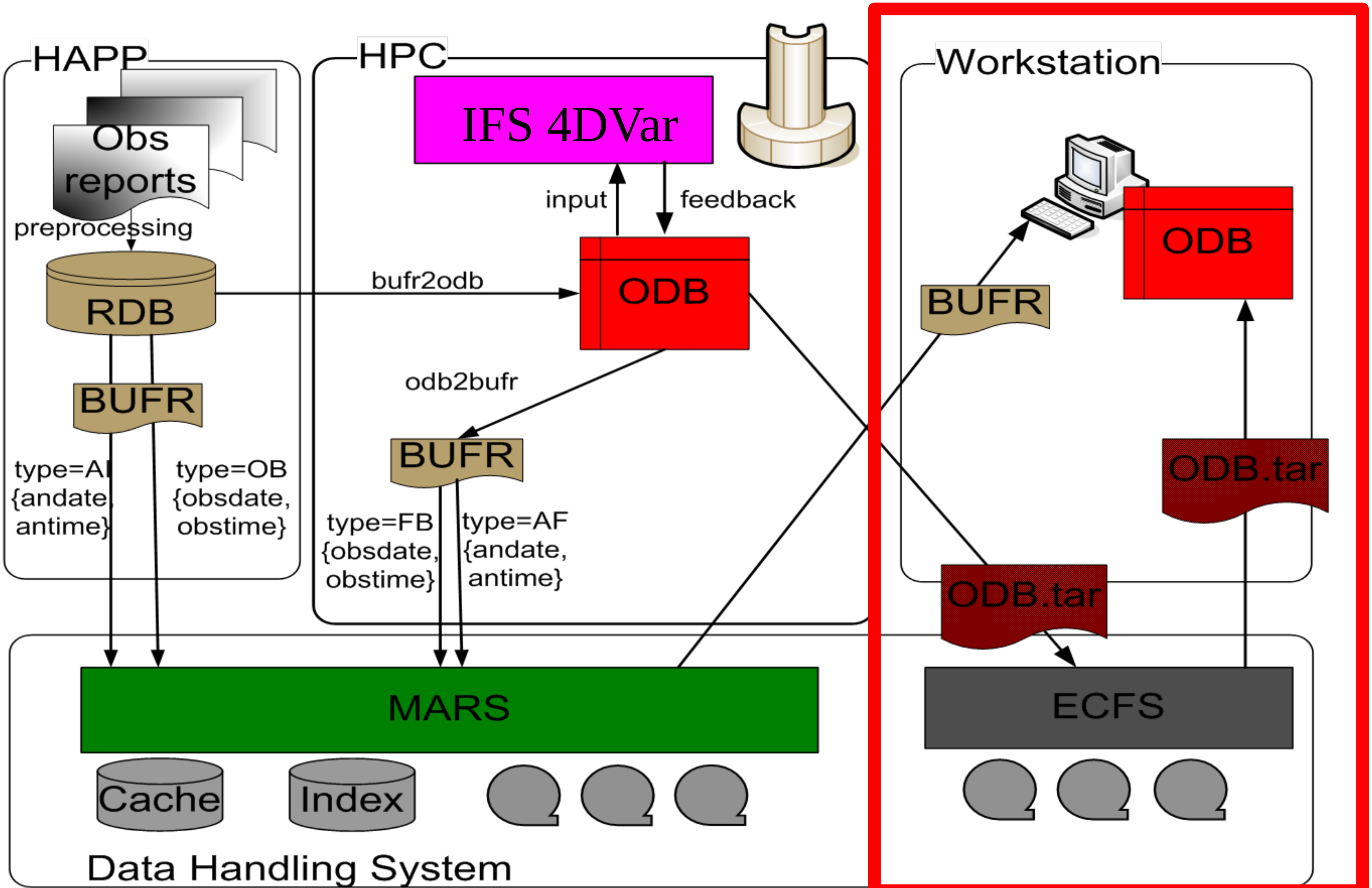
- **The last trajectory is even more expensive than the first one because it involves all observations. About 20 Gb have to be loaded in memory!**
- **Could we run two last trajectories?**
  - **One with CCMA only (this one would be in the critical path)**
  - **Another for ECMA-CCMA which would not be in the critical path (for diagnostics only)**
- **Use vertical partitioning for each step (traj_0, ifsmin_0, etc.) i.e. write in different files (tables) and create an "incremental ECMA/CCMA". This approach is successfully used for our ensemble kalman filter suite (Mats Hamrud)**

ECMWF

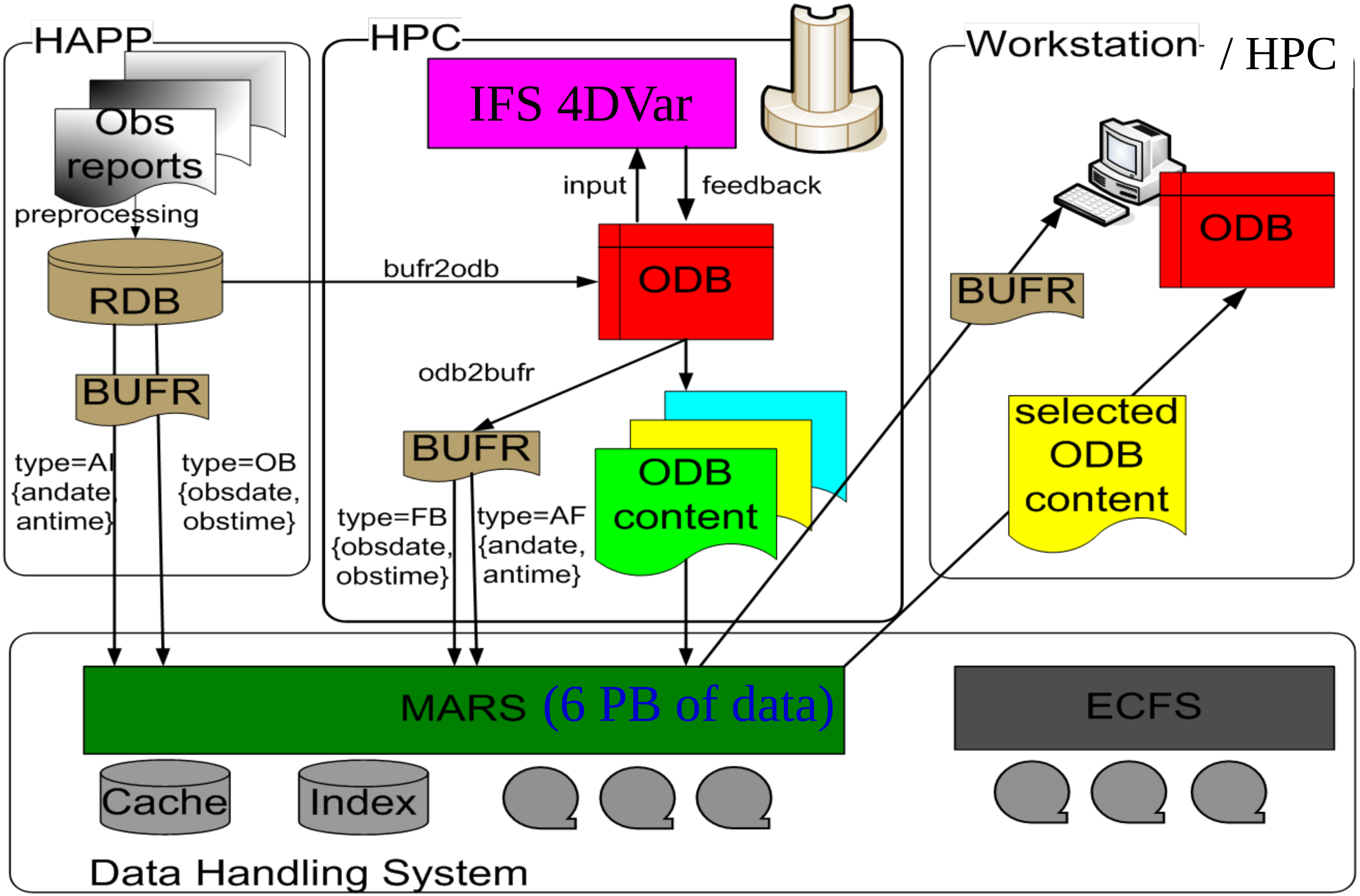
Is scalability the only issue?

- And what happens if we can make it?
  - We will use more and more observations…
  - We will write more and more feedback information from our assimilation system…
- And what do we do with GBs of observation feedbacks?
- Scientists write ODBs to monitor observations and analyze data
  - The final goal is to improve our forecasting system…
- How do we store this feedback information? What tools, visualization facilities do we offer to users?

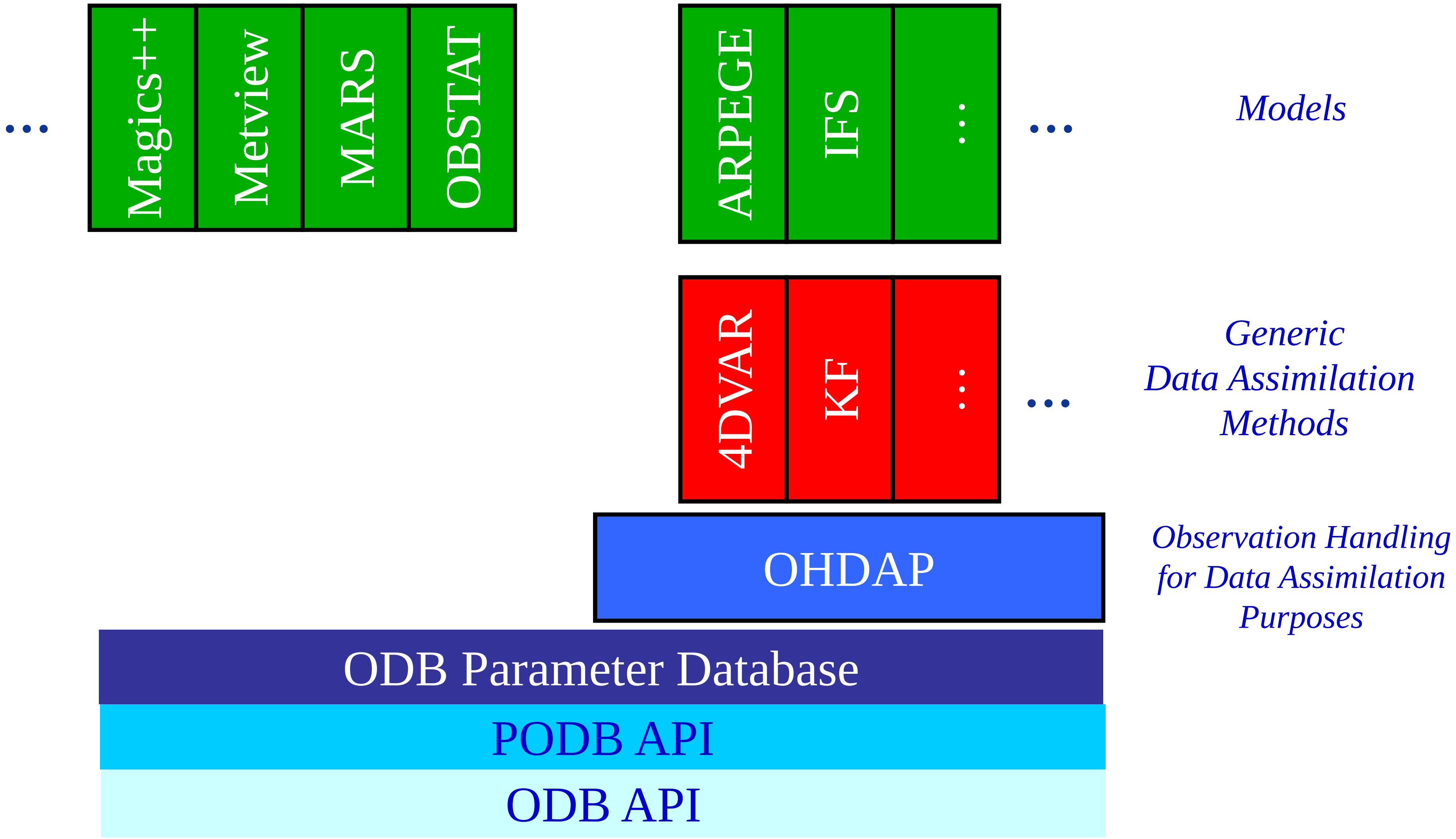
Current observation data flow at ECMWF



Proposed observation data flow at ECMWF



Proposed framework for Observation handling



ECMWF

Conclusion

- **The ODB software will still evolve: a new C++ library is under development (Peter Kuchta). We believe that an object-oriented approach will help to improve the scalability of both ODB and its usage in IFS**
- **Diagnostic tools (to monitor I/Os, debug, analyze runtime applications, etc.) are necessary**
- **As well as tools for scientists to analyze, visualize and monitor feedback data (ODB-tools, MARS, obstat, magics++, metview, etc.)**
- **Optimizations will be a common effort between scientists, analysts, computer vendors, etc.**

It can't be the work of a single man/woman!!!

Questions?

With many thanks to John Hague, Peter Kuchta and Manuel Fuentes