Data interfaces between IFS and product generation

Challenges and Opportunities

Simon Smart, Antonino Bonanni, Florian Rathgeber, Tiago Quintino

Forecast department
Development Section
Data Handling Team

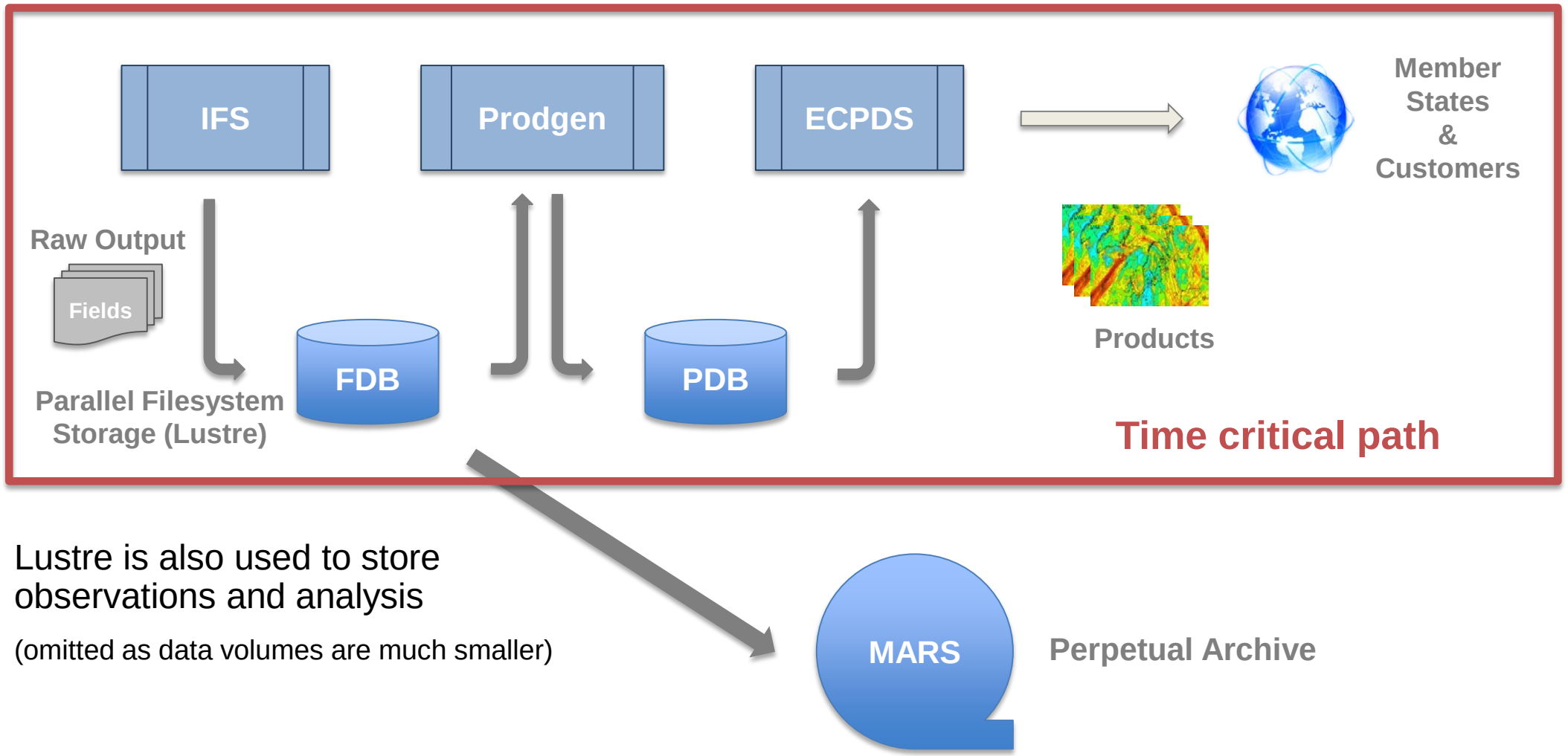
simon.smart@ecmwf.int

1st ECMWF Scalability day, 24 May 2016

ECMWF

nextgenio

Simplified Production Workflow



Lustre is also used to store observations and analysis

(omitted as data volumes are much smaller)

Problems

1. The sheer volume of data

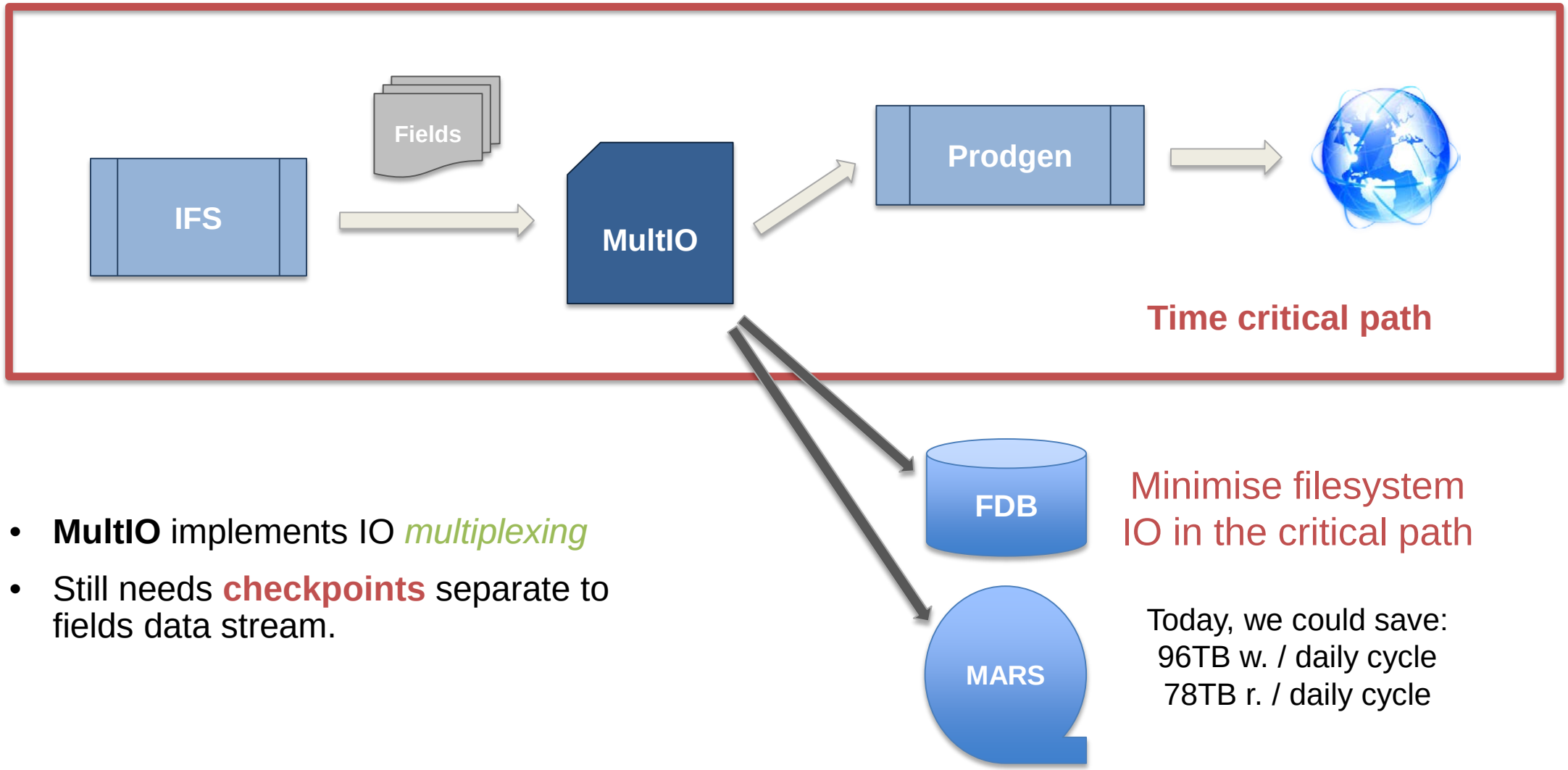
2. Repeated Lustre (5 in this diagram)
   - Note Lustre for checkpointing to retain resilience.

3. POSIX semantics
   - We make use of the assurances given by POSIX semantics.
   - Concurrency assurances generate significant overheads.
   - These overheads don't scale on parallel systems *(see Lustre Metadata server).*

Two pronged approach:

1. Split the data streams into time-critical, and non-time-critical
2. Minimise the Lustre activity in the time critical path.

Redirecting the Data Flows



- **MultIO** implements IO *multiplexing*
- Still needs **checkpoints** separate to fields data stream.

Minimise filesystem IO in the critical path

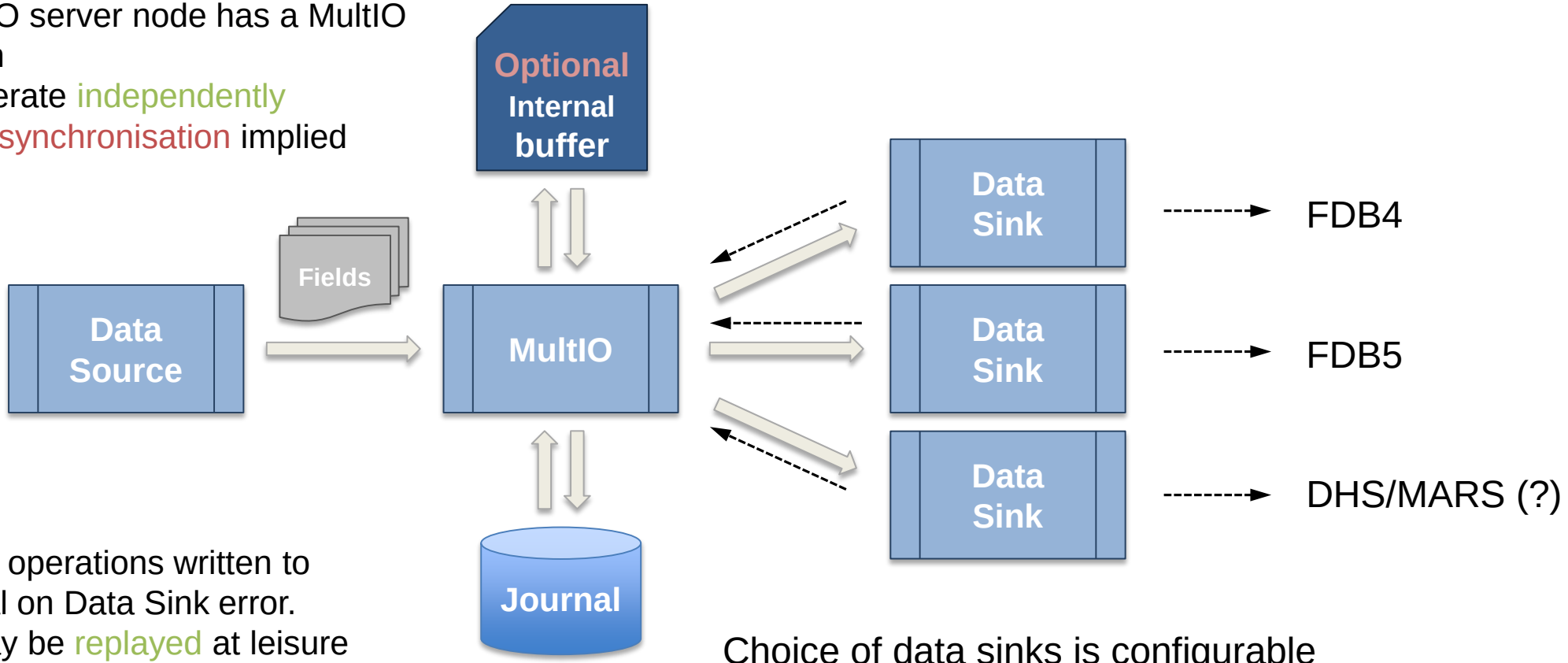
Today, we could save:
96TB w. / daily cycle
78TB r. / daily cycle

MultIO Library Structure

Targeted for potential inclusion in cycle 43r1

Each IO server node has a MultIO system

- Operate independently
- No synchronisation implied



Failed operations written to journal on Data Sink error.

- May be replayed at leisure

Choice of data sinks is configurable

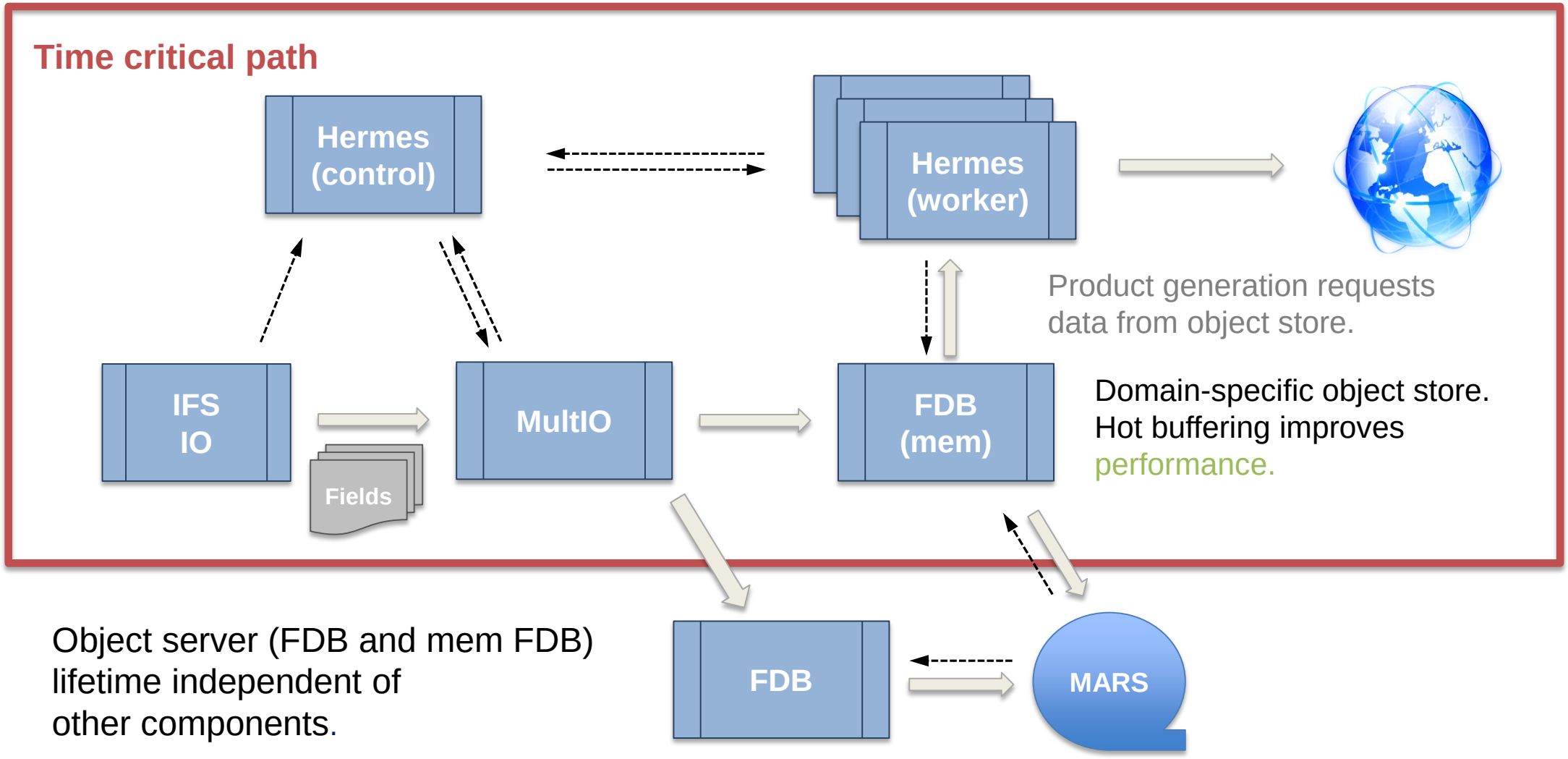
- May be asynchronous
- Errors caught and reported to MultIO

The Opportunity: Object-Store Semantics

- Avoid POSIX filesystem semantics
  - Create and publish objects.
  - Access via globally unique metadata (MARS request)
  - Can be distributed over multiple nodes without synchronization overhead.
- Don’t write to disk (or Lustre)
  - Time-critical hot path remains in memory.
- Object lifetime in cache is independent of IFS, Product Generation, or archiving.
- Improve performance and resilience
  - An additional, high performance layer
  - Additional routes for circumventing hardware failures

A distributed FDB in memory

Overall infrastructure plan



Object server (FDB and mem FDB) lifetime independent of other components.

- Aims to develop a platform and uses for NVRAM technology.

- Key characteristics:
  - Storage density similar to NAND flash memory
  - Better durability, speed and latency than NAND, slower than DRAM
  - Priced between NAND and DRAM

- Usage in operational workflow:
  - Provide the capacity to implement FDB in memory in operations.
  - Provide the speed to be an efficient hot-cache.
  - Provide persistence to enhance resilience.

Thank You!

Questions?