Hewlett Packard Enterprise

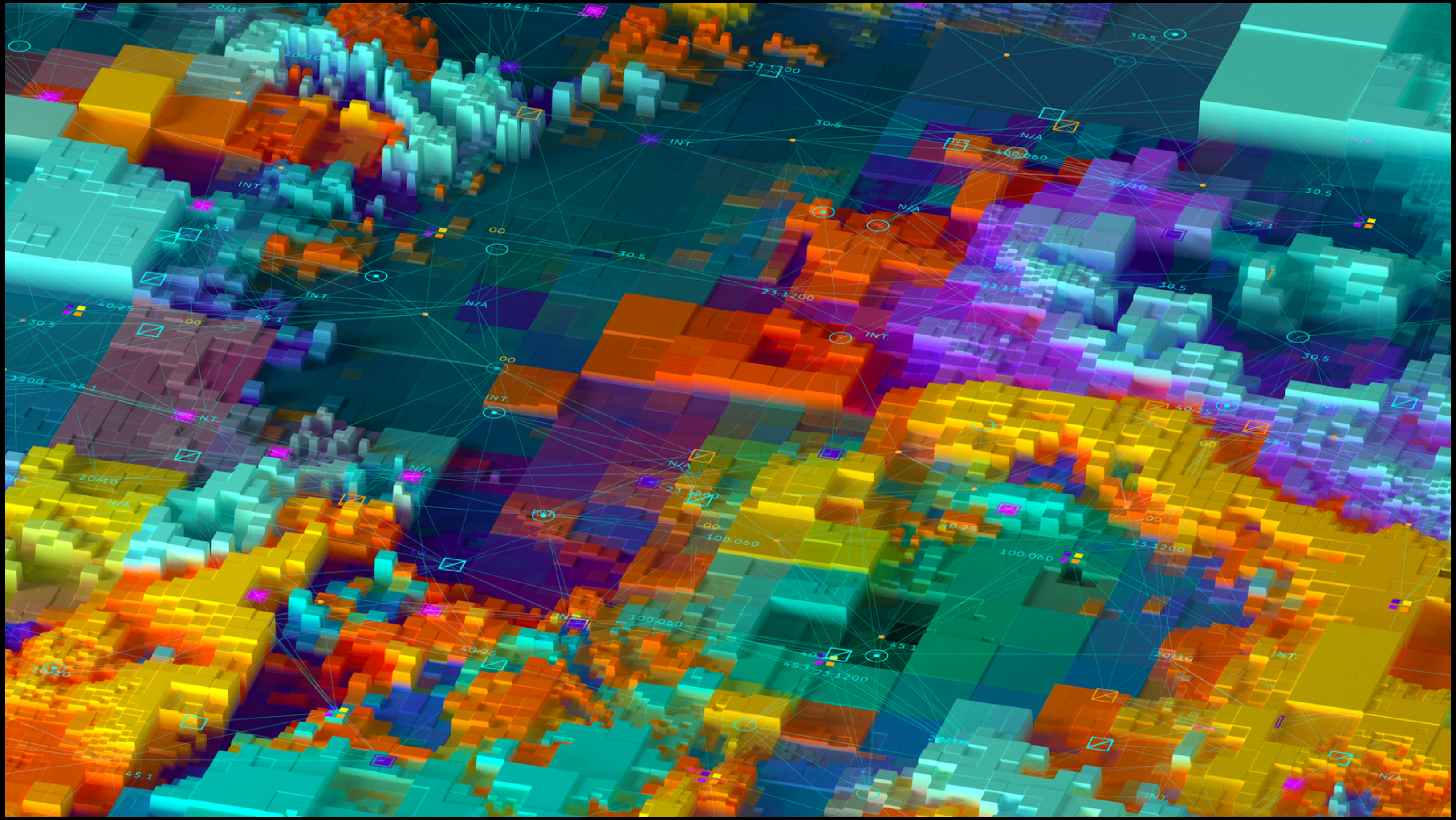
BRIDGING GAPS: THE MAESTRO DATA-AWARE MIDDLEWARE

Utz-Uwe Haus, Head of HPE HPC EMEA Research Lab

joint work with Ali Mohammed, Christopher Haine (HPE) and Domokos Sarmany, Milos Lompar, Simon Smart (ECMWF)

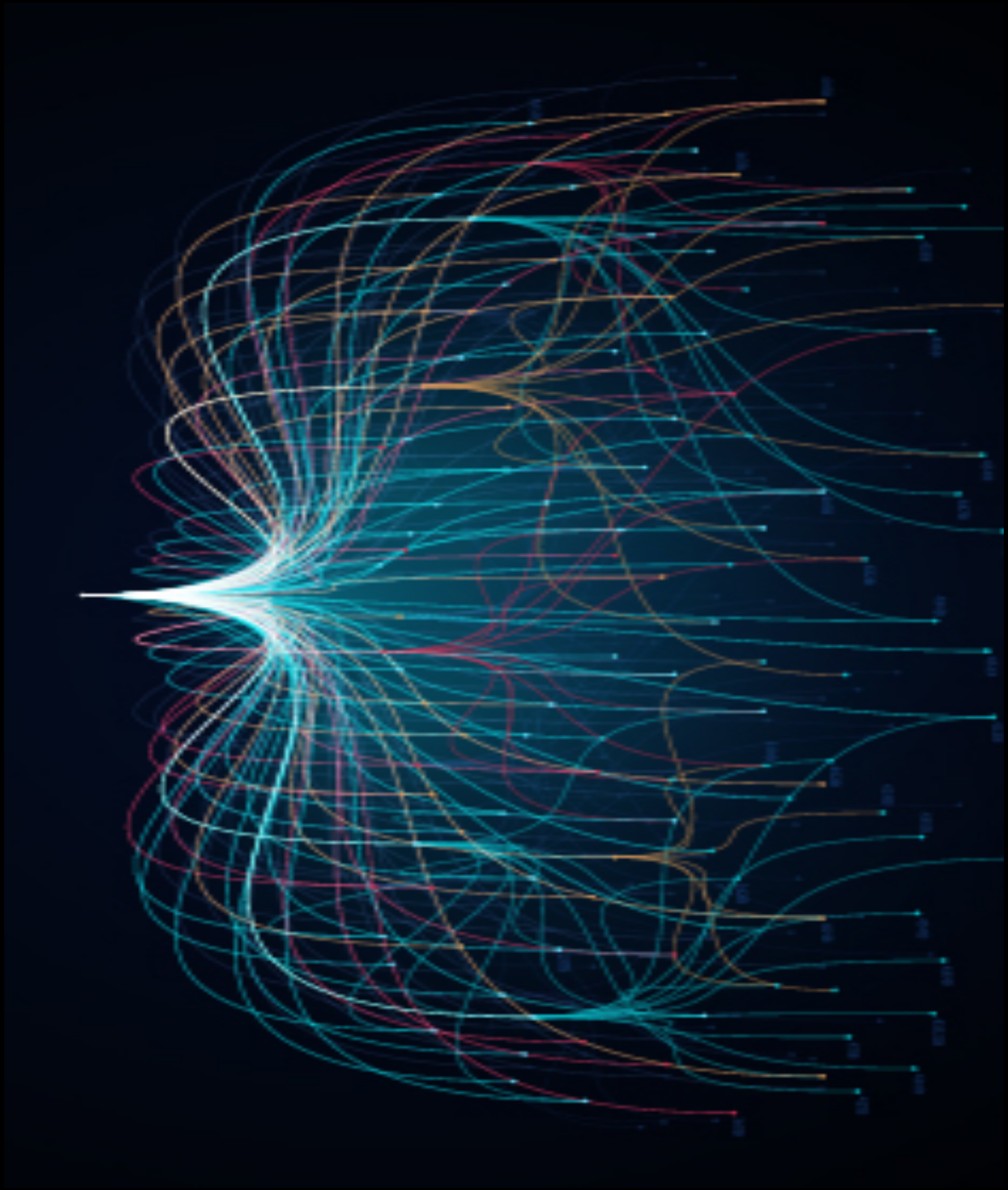
HPCWS2021 – virtual 2021-09-21

This project received funding from the European Union's Horizon 2020 research and innovation program through grant agreement 801101.

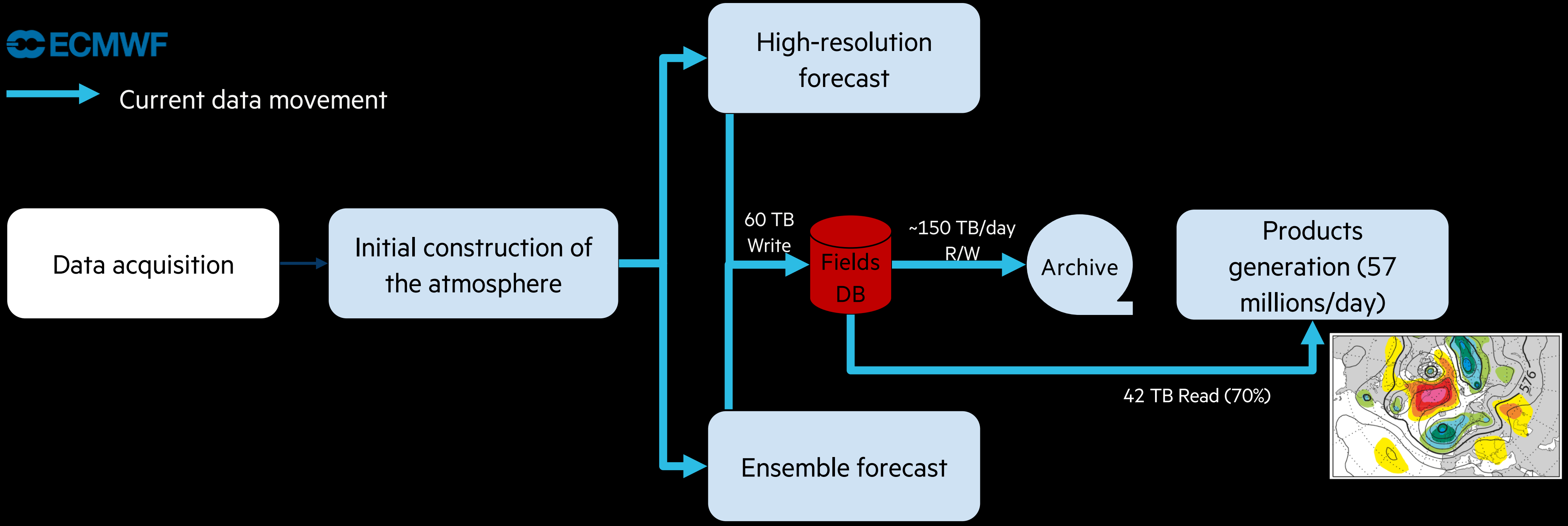


THE CHALLENGES

- Memory is diverse
  - Not really a hierarchy anymore, in terms of non-monotone bandwidth, latency and capacity
  - Some hardware controlled, some programmed as memory, some programmed as storage
- Programmability is hard
  - Abundance of languages, vendor/standardised APIs, programming frameworks, middleware (eg. Umpire, Object Store), device drivers (eg. Unified memory, zocl), OS utilities (eg. PFS, paging)
- Data movement is also hard
  - Workflows are dominated by data movement
  - Inadequacy of the HPC software stack
  - [Scheduling problem]



Example of user issues: Weather Prediction Workflow



Today's bottleneck

- Data movement between forecast stages and product generation
- Archiving via I/O aggregator nodes into PFS
- Each product generation job is reading from PFS

Vision

- Speed up data-movement for the Pgen step (the 42 TiB)
- Exploit multiple storage technologies
- More flexible dependencies

MAESTRO

DATA ORCHESTRATION

CDO (Core Data Object)
It is at the heart of Maestro's design and is used to encapsulate data and metadata. Supports dependencies.

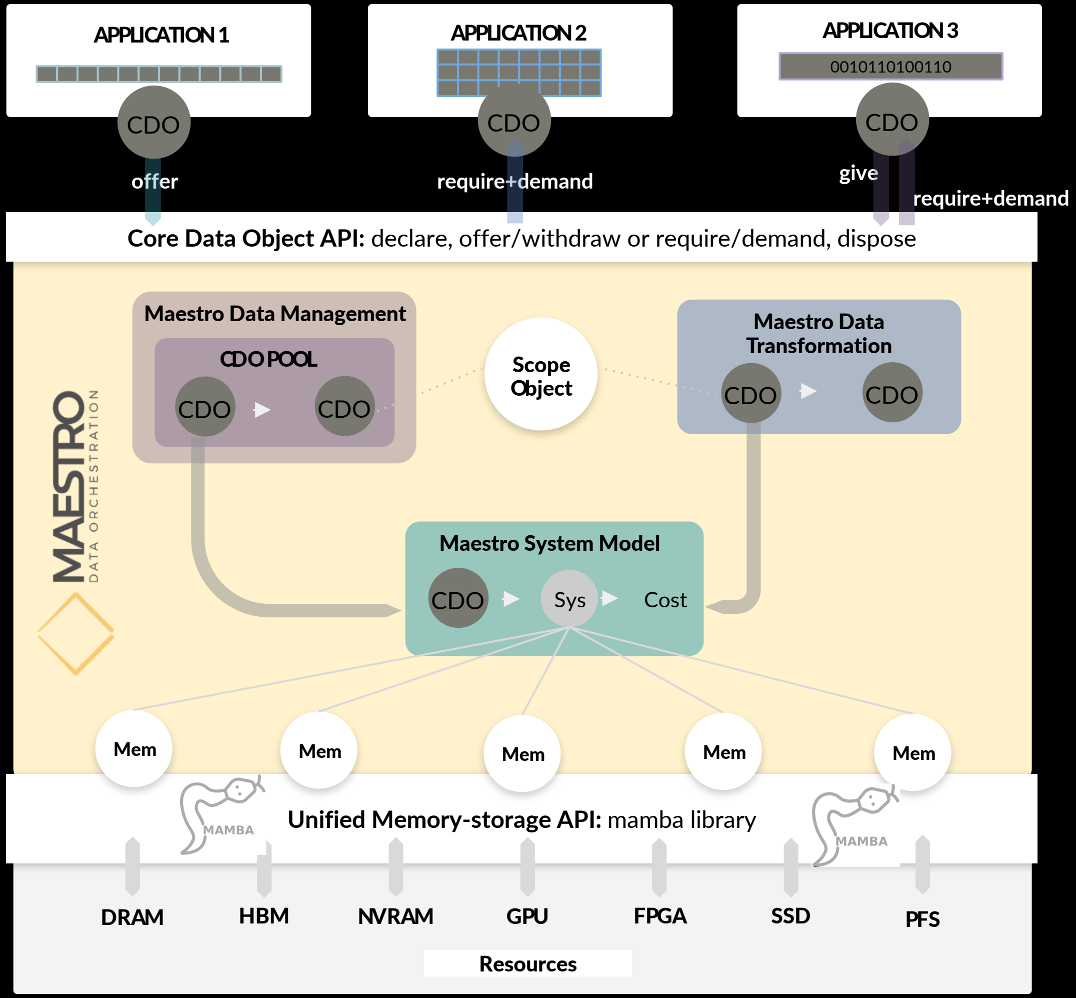
OFFER+WITHDRAW
Applications OFFER CDOs to the management pool. Maestro manages the data, until WITHDRAW occurs.

REQUIRE+DEMAND
When an application REQUIREs a CDO, Maestro makes data available. At DEMAND is hands over resources containing the data and relinquishes all control it.

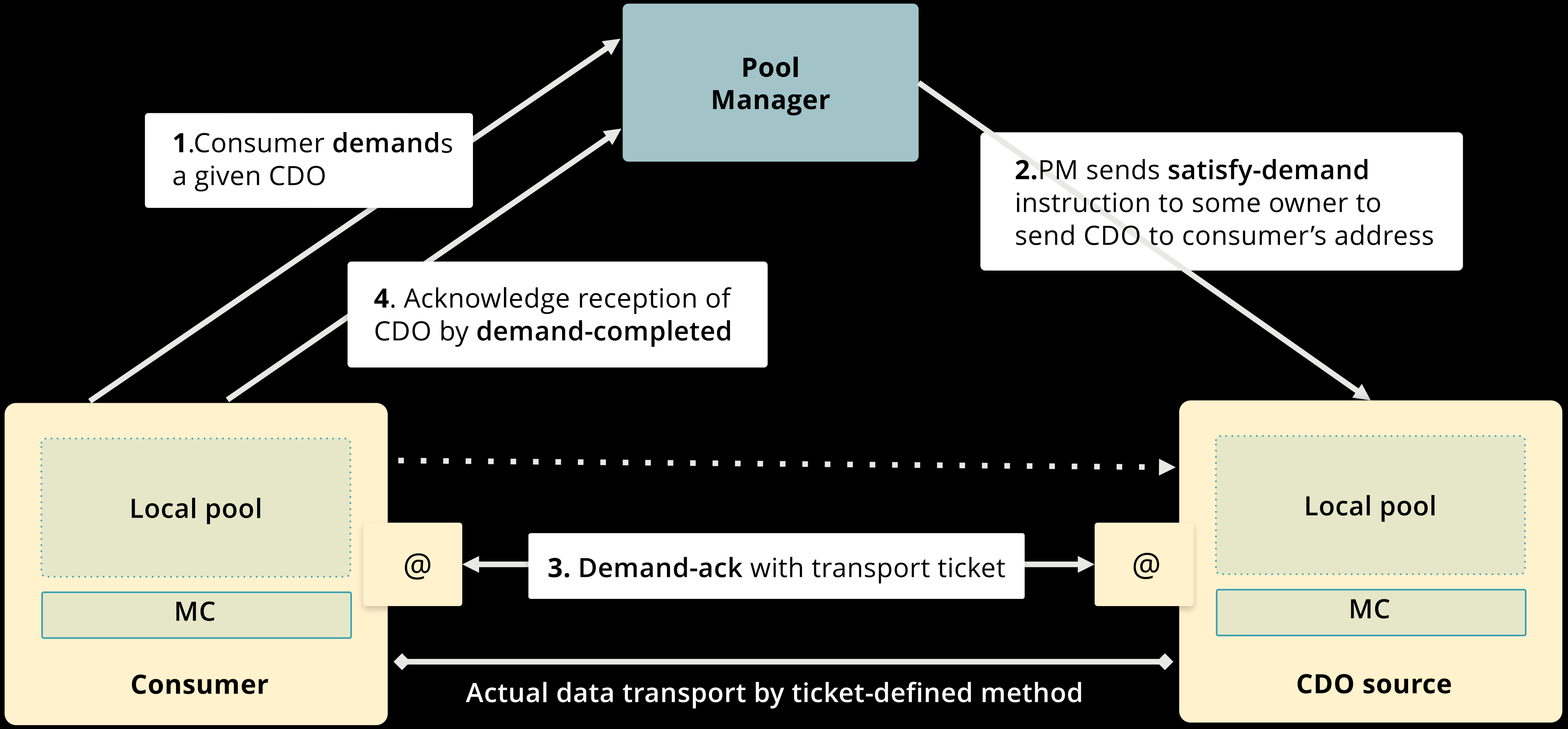
SCOPE OBJECT
Captures information about scope, size, access relations and schedules of the data to enable efficient movement and/or transformation

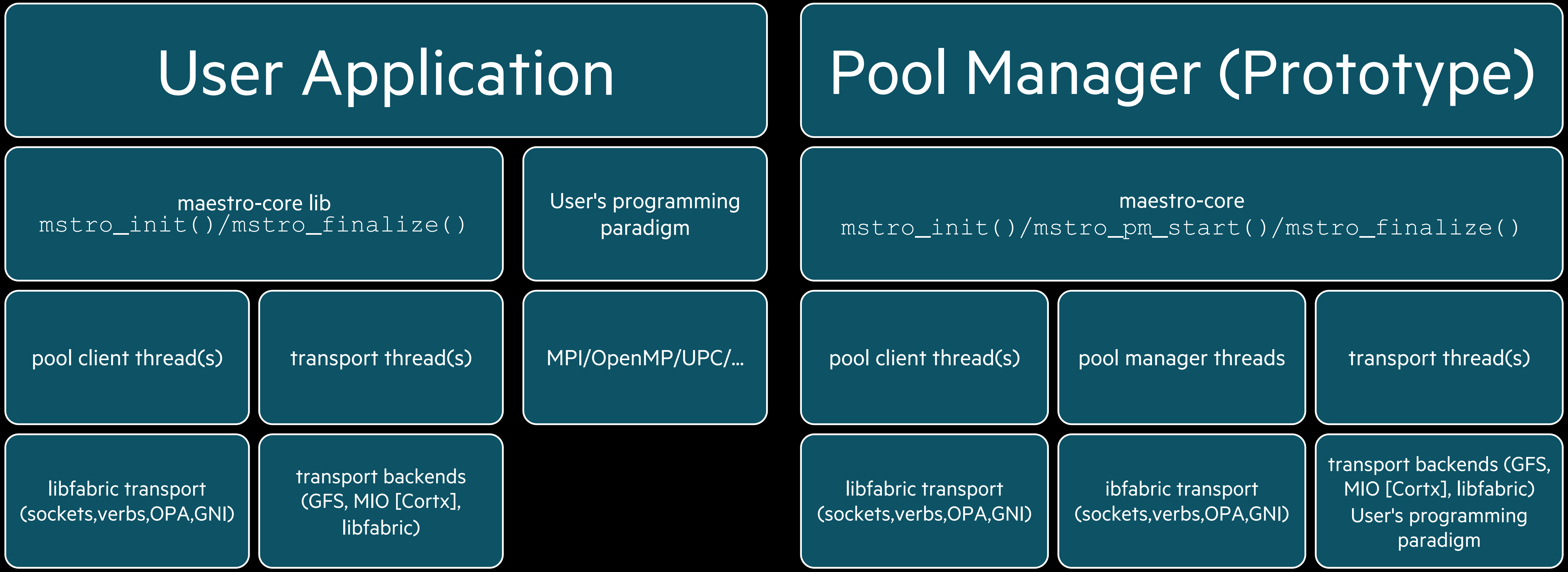
MAESTRO SYSTEM MODEL
Computes the cost of moving, transforming or copying data a CDO

SYS
Interface to every memory level, enabling core functionality of that memory via mamba library.



Mamba developed by https://epigram-hs.eu/


Pool Manager
1.Consumer demands a given CDO
2.PM sends satisfy-demand instruction to some owner to send CDO to consumer's address
4. Acknowledge reception of CDO by demand-completed
Local pool
MC
Consumer
@
3. Demand-ack with transport ticket
@
Local pool
MC
CDO source
Actual data transport by ticket-defined method


User Application
maestro-core lib
mstro_init()/mstro_finalize()
User's programming paradigm
pool client thread(s)
transport thread(s)
MPI/OpenMP/UPC/...
libfabric transport (sockets,verbs,OPA,GNI)
transport backends (GFS, MIO [Cortx], libfabric)
Pool Manager (Prototype)
maestro-core
mstro_init()/mstro_pm_start()/mstro_finalize()
pool client thread(s)
pool manager threads
transport thread(s)
libfabric transport (sockets,verbs,OPA,GNI)
ibfabric transport (sockets,verbs,OPA,GNI)
transport backends (GFS, MIO [Cortx], libfabric) User's programming paradigm

Any application can linked with `-lmaestro`

Current PM is a single-node, multithreaded program
Single-node multithreaded PM in testing
(MPI-based distributed PM TBD)

- Pool Manager is market-maker for CDOs
- Pool Manager ensures no provider of a CDO can leave if any REQUIRE is still outstanding; will proactively start transport if needed
- Workflow managers can insert coordination apps into workflow without disturbing or modifying other applications
- 0-length CDOs can be (ab)used as synchronization items

- Simple reusable apps:
  - Archiver
    - Subscribe to CDO OFFER event
    - Immediately post a REQUIRE
    - DEMAND and write to permanent storage
  - Librarian
    - Subscribe to CDO REQUIRE
    - Look into permanent storage and post OFFER if feasible
  - Checkpoint librarian
    - Mix of Archiver and Librarian: Subscribe to OFFERs and REQUIREs with (user-defined) checkpoint attribute
  - Tracer
    - Subscribe to user-defined events
    - Write to logging framework
  - Keep-Alive proxy
    - Post a REQUIRE for CDO1 until an OFFER for CDO2 is seen

A BIT MORE ON METADATA

User-defined metadata

- via YAML schema
- Basic types and composite

Native metadata

- Name, size, lifetime, persistence and more
- Data layout

- **Events**
  - Subscription to pool messages (eg. *declare*, *offer*)
    - Pre- and post-op
    - w/ or w/o ack: permits explicit control of the workflow
      - tracing, librarian service, single-stepping
    - Select by metadata query algebra
      - Boolean, numeric, string-match
  - *poll* or *wait*
  - Allows CDO cherry-picking

Shared knowledge:
- Workflow name

Requirement:
- unique application names

Pool manager (PM)
- Enumerates all OFI interfaces
- Publishes Workflow ID block for RDMA
- Listens on all endpoints
- Provides OOB *“pool manager info”*

User application (PC)
- Enumerates all OFI interfaces

PM-INFO →

- Uses OOB *“pool manager info”* to probe how to best reach PM (reading Workflow ID block)

← JOIN

- Sends JOIN

PM:
- Replies to JOIN, assigning app-id

WELCOME →

- Receives WELCOME
- Announces resources
- Performs pool operations

← LEAVE

- Sends LEAVE

PM:
- Replies to LEAVE with BYE when PC can be permitted to leave

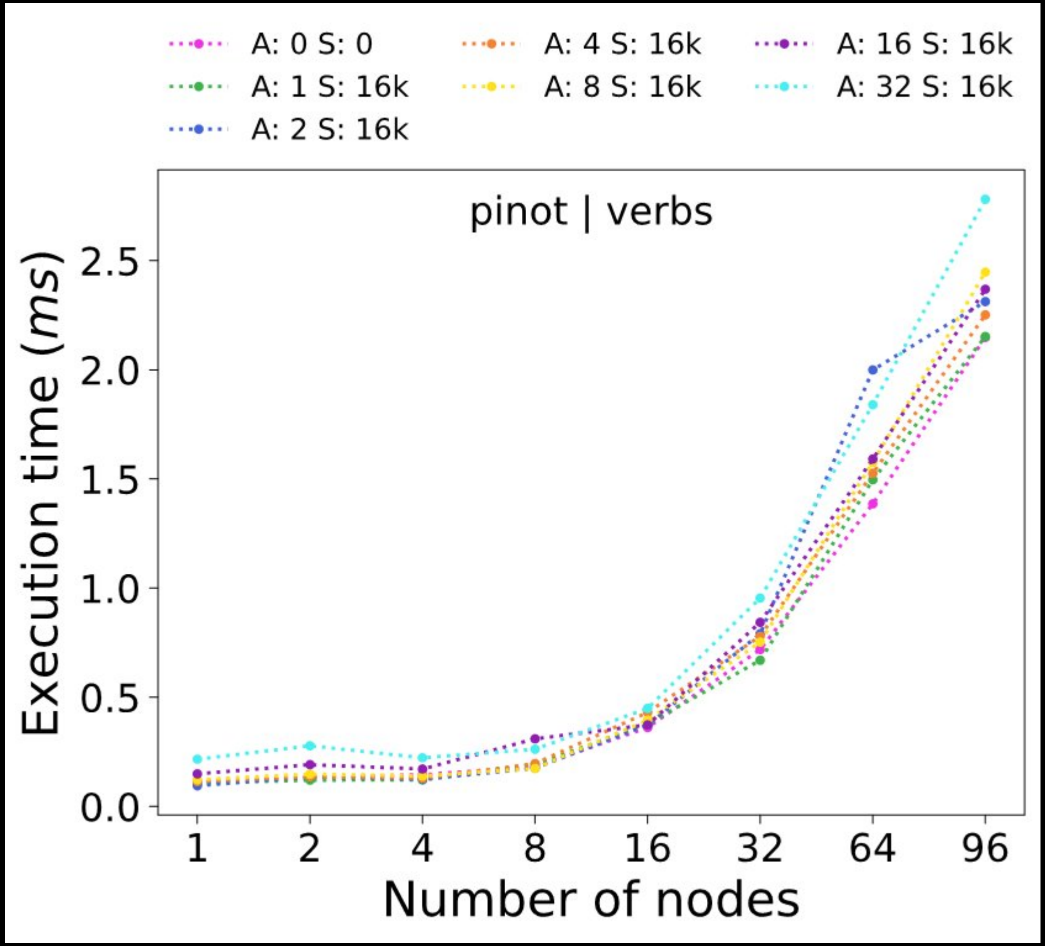
BYE →

- Terminates after BYE

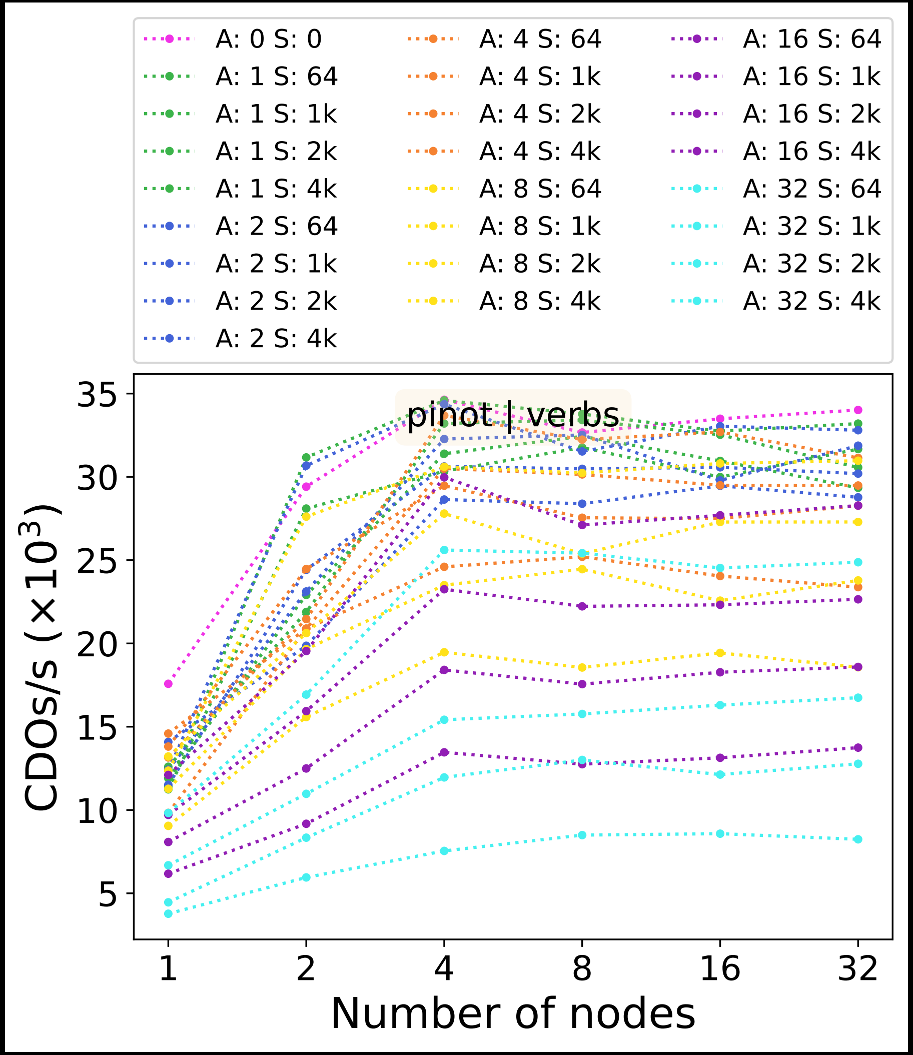
- Works across different WLM allocations
- Works across different network segments as long as PM is multi-homed
- Could support per-app authorization

MAESTRO-CORE PERFORMANCE

Declared/offered CDOs/s



128 threads per node injecting at 0.2…2.5ms/CDO for each

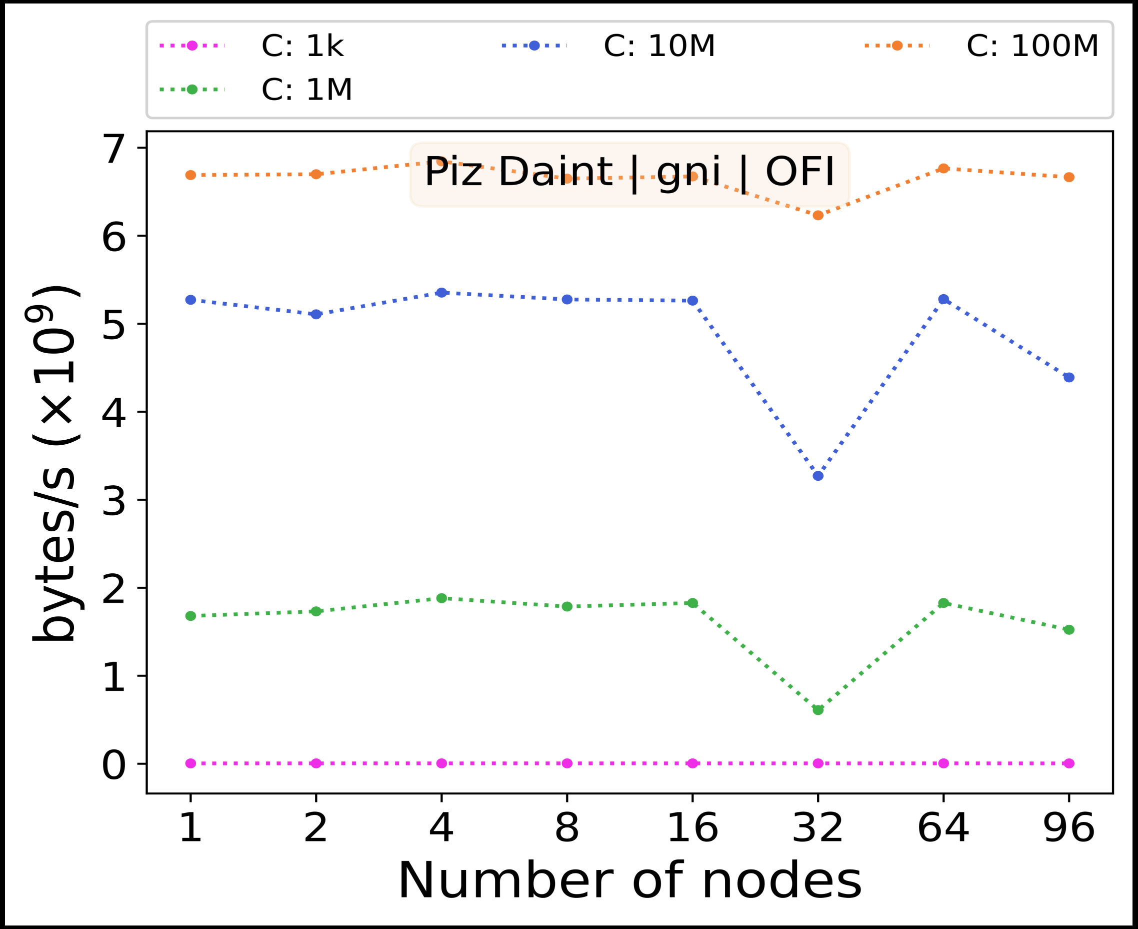


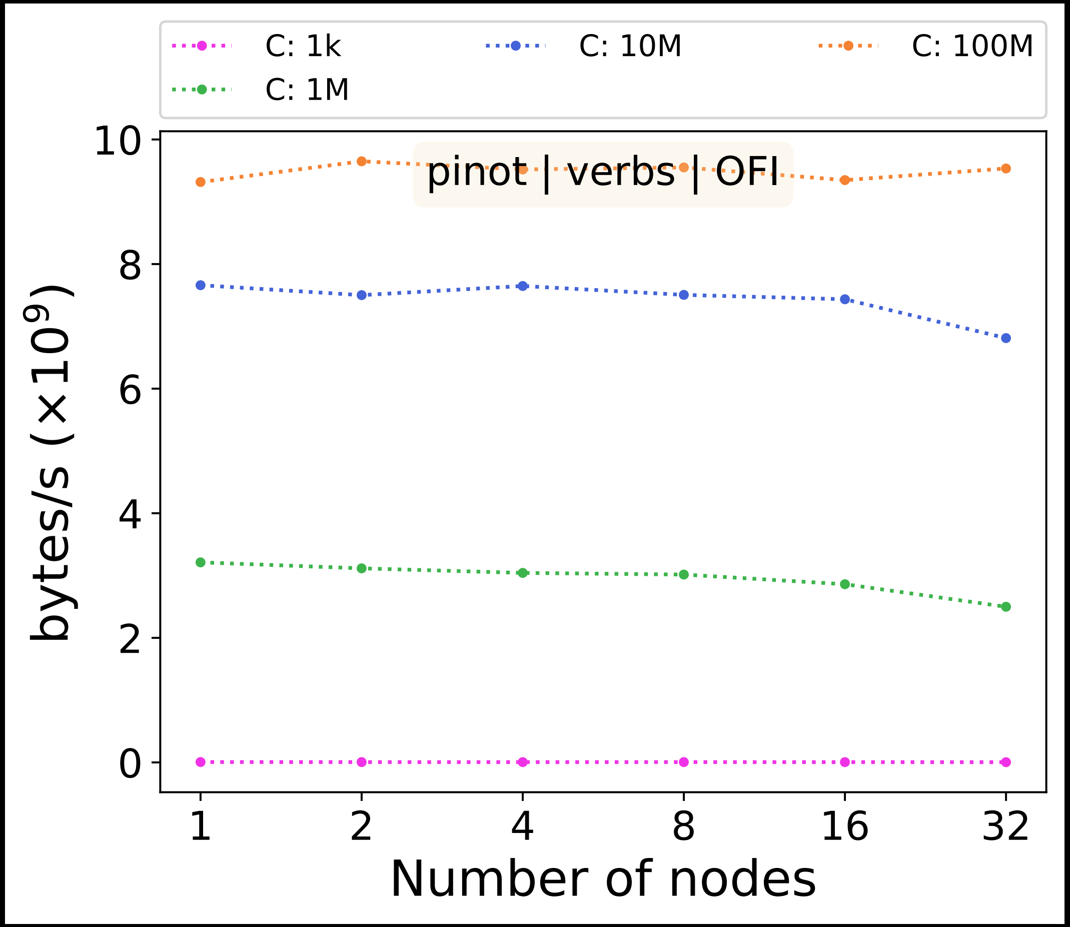
35k CDO/s through 1-thread PM
4 injectors (of 128 threads each) saturate

MAESTRO-CORE PERFORMANCE

RDMA TRANSPORT

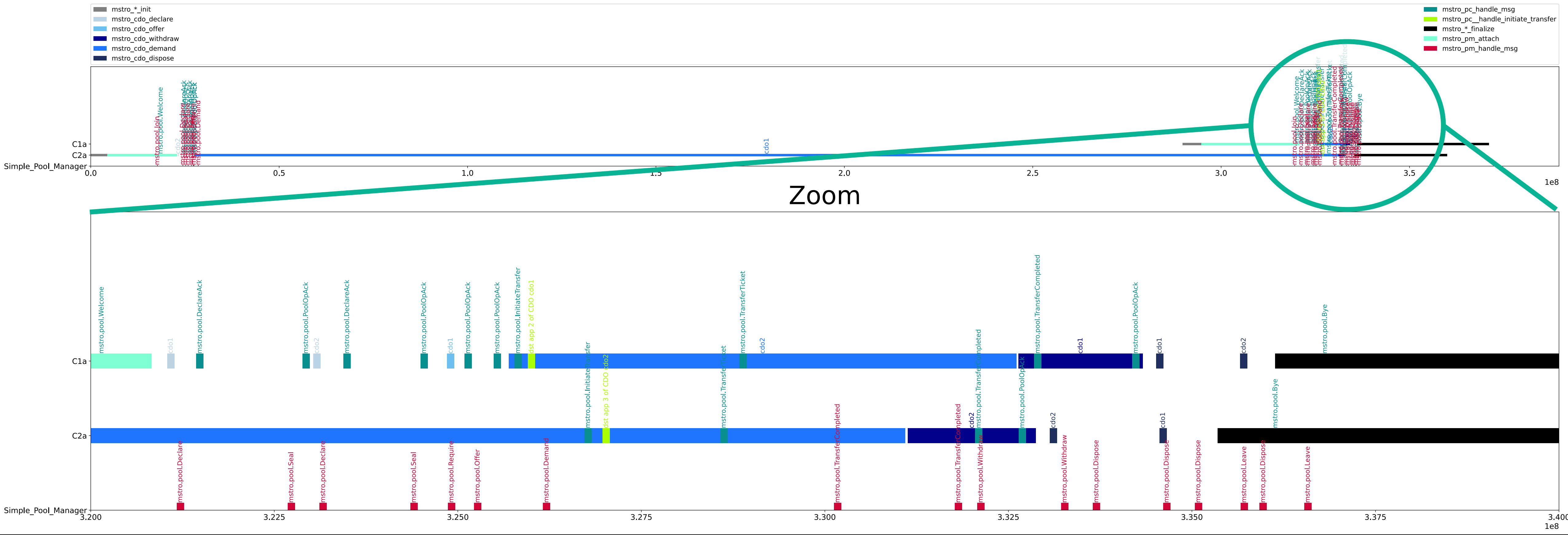
MPI application: N producer ranks, 1 PM rank, 1 consumer rank (sinks all CDOs)



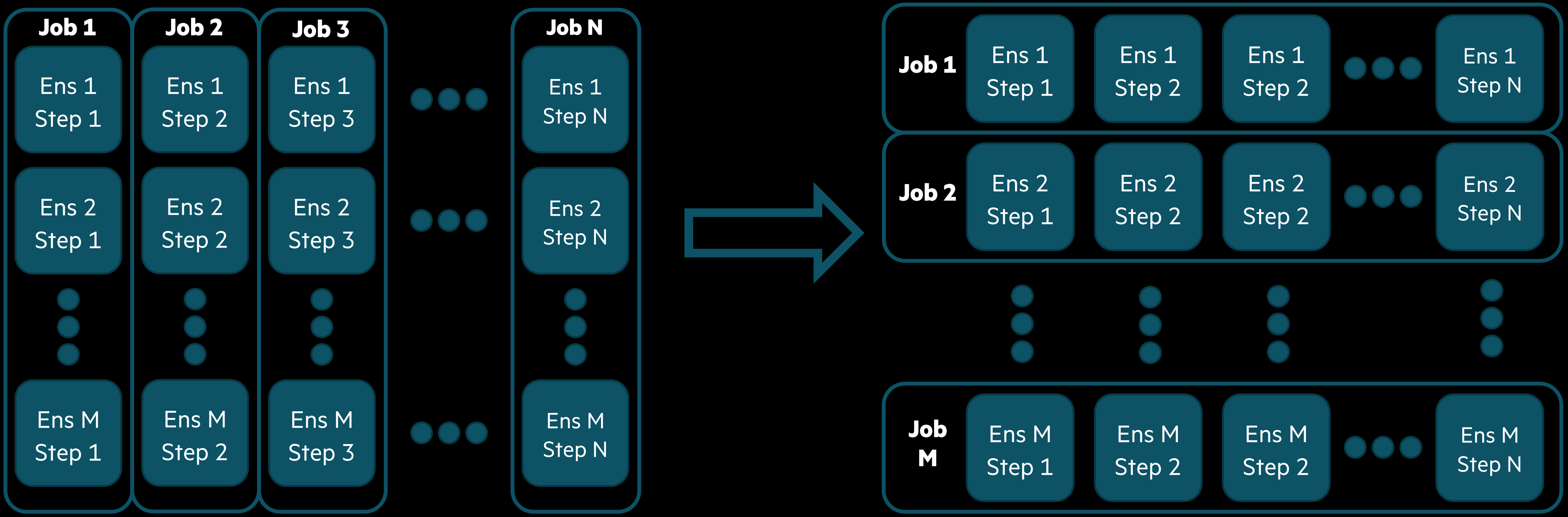


250 CDO/node are offered

1260 CDO/node are



mstro_*_init
mstro_cdo_declare
mstro_cdo_offer
mstro_cdo_withdraw
mstro_cdo_demand
mstro_cdo_dispose
mstro_pc_handle_msg
mstro_pc__handle_initiate_transfer
mstro_*_finalize
mstro_pm_attach
mstro_pm_handle_msg
C1a
C2a
Simple_Pool_Manager
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
1e8
cdo1
Zoom
mstro.pool.Welcome
cdo1
mstro.pool.DeclareAck
mstro.pool.PoolOpAck
cdo2
mstro.pool.DeclareAck
mstro.pool.PoolOpAck
cdo1
mstro.pool.PoolOpAck
mstro.pool.PoolOpAck
mstro.pool.InitiateTransfer
dst app 2 of CDO cdo1
mstro.pool.TransferTicket
cdo2
mstro.pool.TransferCompleted
cdo1
mstro.pool.PoolOpAck
cdo1
cdo2
mstro.pool.Bye
mstro.pool.InitiateTransfer
dst app 3 of CDO cdo2
mstro.pool.TransferTicket
cdo2
mstro.pool.TransferCompleted
mstro.pool.PoolOpAck
cdo2
cdo1
mstro.pool.Bye
mstro.pool.Declare
mstro.pool.Seal
mstro.pool.Declare
mstro.pool.Seal
mstro.pool.Require
mstro.pool.Offer
mstro.pool.Demand
mstro.pool.TransferCompleted
mstro.pool.TransferCompleted
mstro.pool.Withdraw
mstro.pool.Withdraw
mstro.pool.Dispose
mstro.pool.Dispose
mstro.pool.Dispose
mstro.pool.Leave
mstro.pool.Dispose
mstro.pool.Leave
3.200
3.225
3.250
3.275
3.300
3.325
3.350
3.375
3.400
1e8



- Product-generation (consumer) workflow is adapted to use the maestro-core library
- Original workflow: each Pgen job can only be triggered once all data from a corresponding step is persisited
- Maestro-enabled workflow: each Pgen job can consume data as soon as made available
- Maestro-enabled workflow is past the correctness test; performance testing is ongoing
- Uses: PM events to track job state, trigger Pgen jobs (selected by attributes), RDMA transport (libfabric)

THANK YOU

uhaus@hpe.com

Information: https://maestro-data.eu/

Code: https://gitlab.jsc.fz-juelich.de/maestro/maestro-core

Docs: https://maestro-core.readthedocs.io/

Release 0.2 is out now -- BSD 3-clause licensed