Building Custom Data Services with Mochi

Approved for public release

EXASCALE COMPUTING PROJECT

Case Study: DataSpaces (and friends)

Philip Davis

National Nuclear Security Administration

U.S. DEPARTMENT OF ENERGY | Office of Science

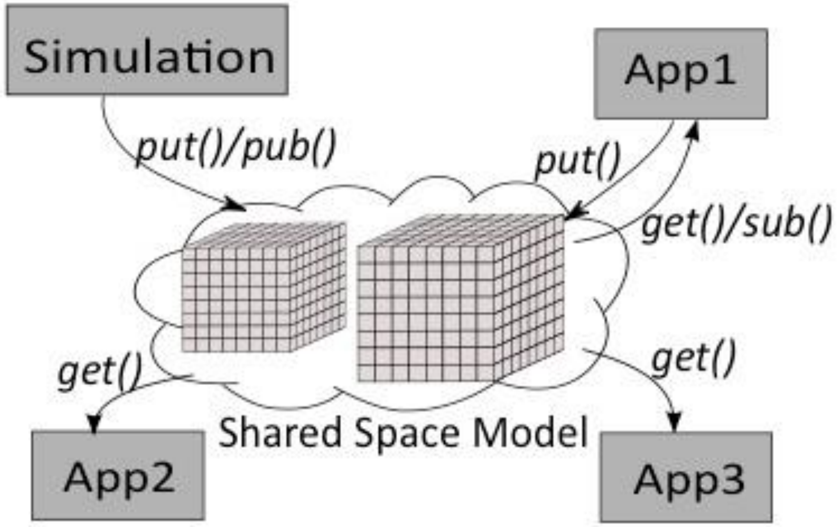
DataSpaces Staging Framework

https://github.com/rdi2dspaces/dspaces

- Simple, powerful abstractions for making data available across all processes in a workflow
  - In-memory data store (either in or out of process)
  - Abstractions for scientific computing
    - N-d array and metadata support optimized for access locality
  - Designed to scale up to thousands of nodes putting and getting
    - Distributed indexing
    - RDMA data transfer

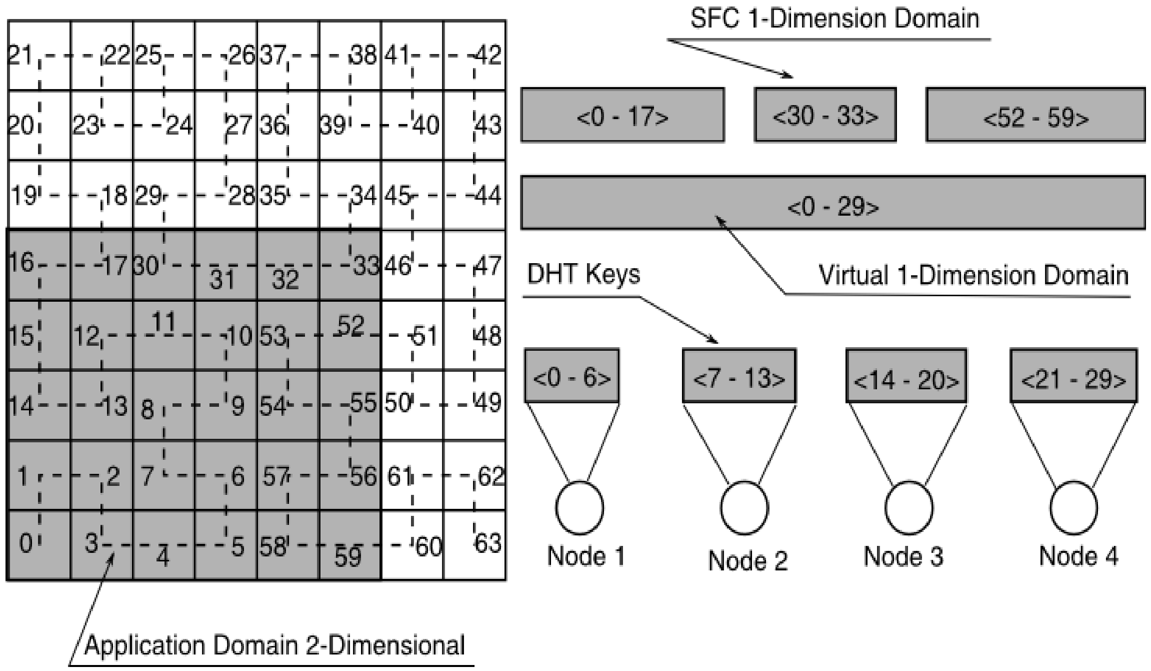
- Useful for in-situ workflows, where multiple process groups are producing and consuming short-lived data products
  - Multi-simulation
  - Simulation / Analysis / Visualization

- Opportunities for “smart” storage
  - Analysis, I/O offload, etc.
  - Data-dependent optimizations (pre-delivery, error detection, etc.)

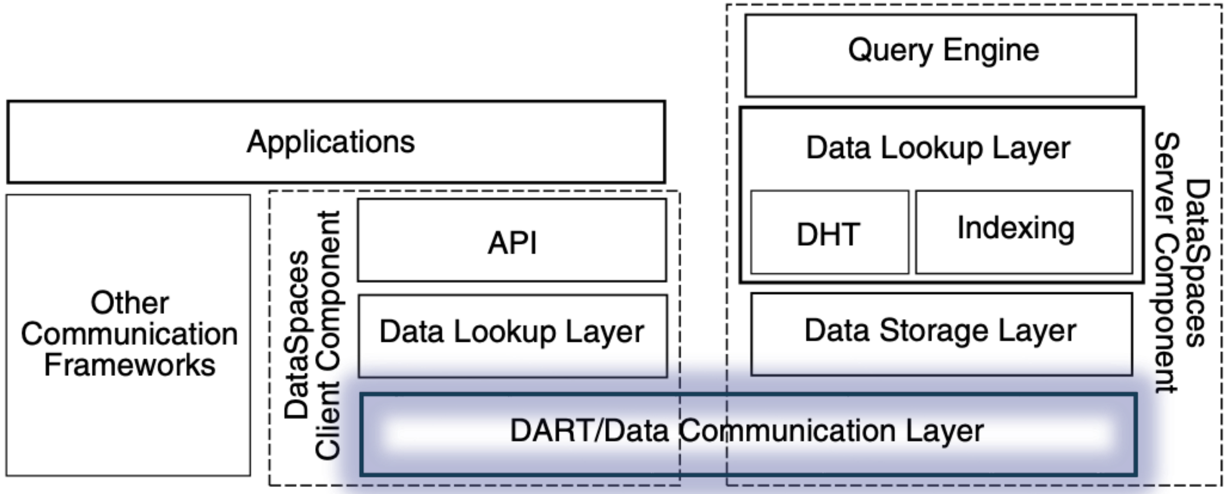


DataSpaces – Architecture

- Client/Server
  - Servers are dedicated threads/processes/jobs.
  - Servers provide indexing and data storage (optionally)
- Index constructed online using SFC mappings or other domain linearization
  - Optimized for locality of indexing (i.e. close data = close index)
- DHT used to maintain indexing metadata
  - indexing domain split across servers by segmenting SFC linearization
  - Locality of SFC reduces multiplication factor of indexing queries
- Communication overlay across DataSpaces clients and servers
  - Independent of, coexists with MPI, etc.
  - Use RDMA transport with RPC-triggered data reorganization



21	22	25	26	37	38	41	42
20	23	24	27	36	39	40	43
19	18	29	28	35	34	45	44
16	17	30	31	32	33	46	47
15	12	11	10	53	52	51	48
14	13	8	9	54	55	50	49
1	2	7	6	57	56	61	62
0	3	4	5	58	59	60	63



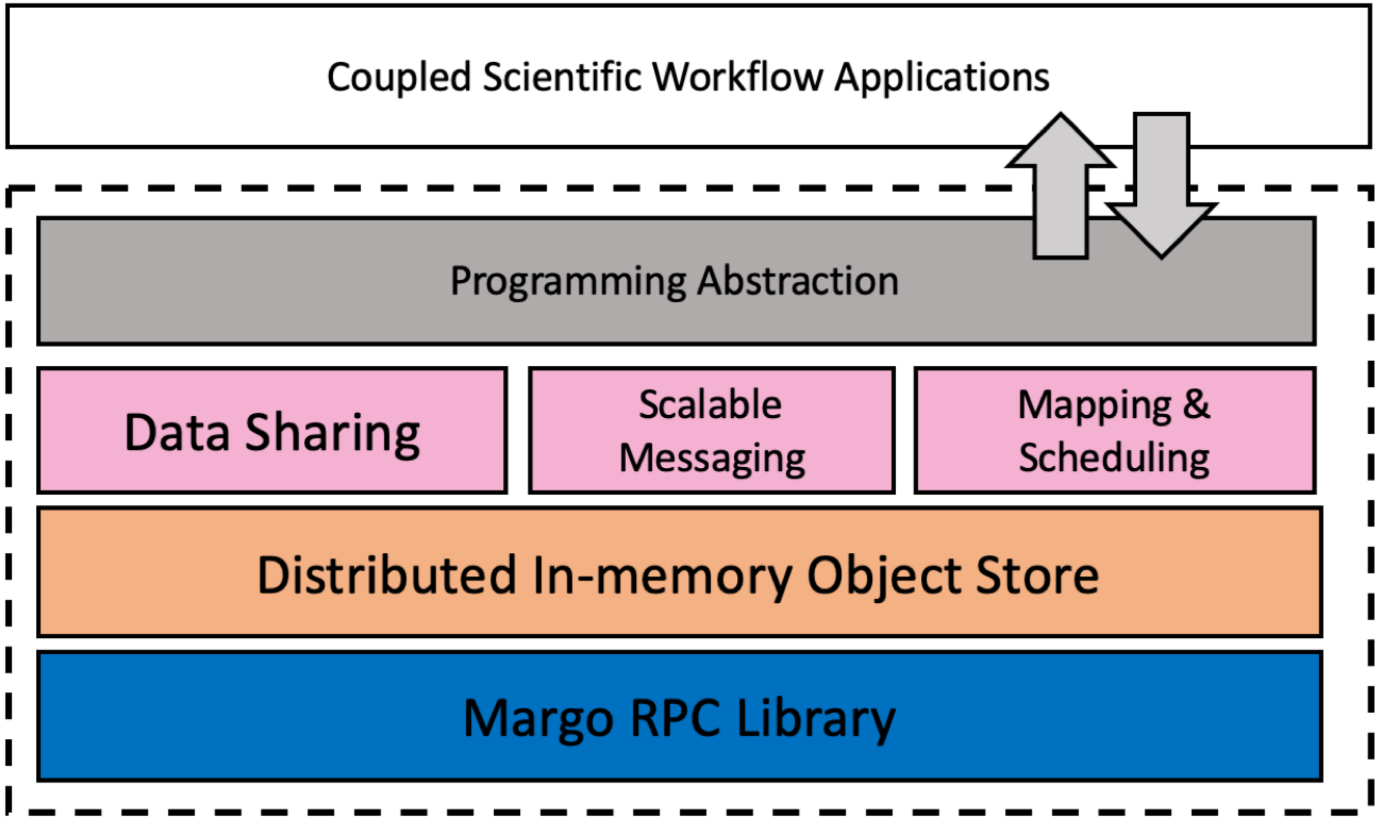
Query Engine
Applications
Data Lookup Layer
DHT
Indexing
Other Communication Frameworks
DataSpaces Client Component
API
Data Lookup Layer
Data Storage Layer
DataSpaces Server Component
DART/Data Communication Layer

DataSpaces 2.0

- API extensions
  - Publish/Subscribe interface
  - Simplified concurrency control
  - Python bindings
  - Hybrid client/server processes
  - Application-informed data placement

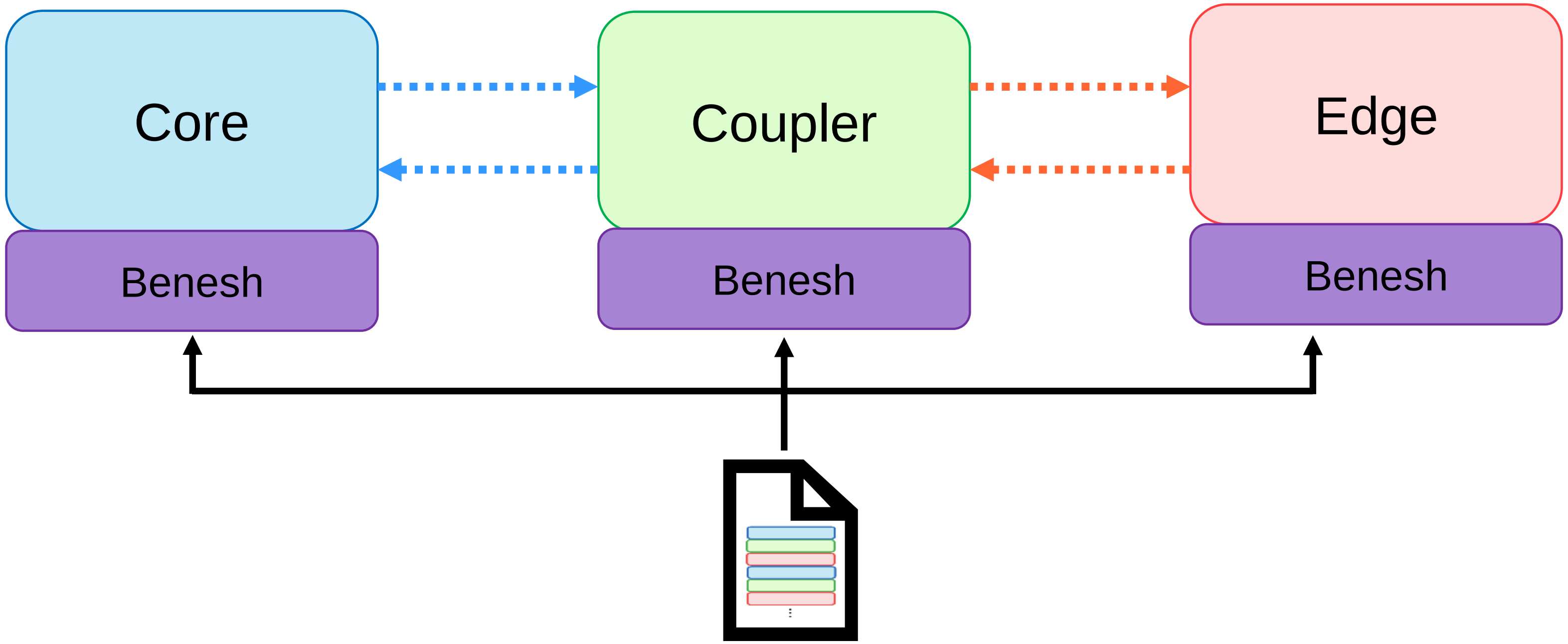
- Architectural enhancements
  - Unified data transport layer
    - Replaces parallel DIMES/DART objects storage models
  - Replace RPC layer with Margo

- Software enhancements
  - Cmake integration
  - Reduced configuration complexity
  - Low overhead running modes



Benesh

- A programming model for developing in-situ workflows
  - Take existing codes and make them work together
  - Abstractions aimed at supporting multiphysics use cases
- Programming-language hooks for a preparing an existing code for use in a Benesh workflow
- Workflow description language for specifying the interactions of workflow components
  - Provide enough information about the workflow to make interactions flexible
- Middleware for instantiating Benesh workflows



EKT – Everyone Knows That

- Benesh components need to send notifications that well-known events have occurred
  - Components are process groups (i.e. each component is an mpirun instance)
  - All ranks of a given component generate the same events (eventually)
  - All ranks of a peer need to know about these events (eventually)
- Everyone Knows That (EKT)
  - Precompute fan-out / fan-in overlay networks between components
  - Broadcast messages with predefined structures using these overlay networks
- Use RPC (mercury) for network bootstrapping and message broadcast