MERCURY DEVELOPER'S MEETING:
ANL ACTIVITIES

PHIL CARNS carns@mcs.anl.gov
JOHN JENKINS jenkins@mcs.anl.gov
Argonne National Laboratory

July 14, 2016

ENABLING DATA SERVICES (MOCHI)

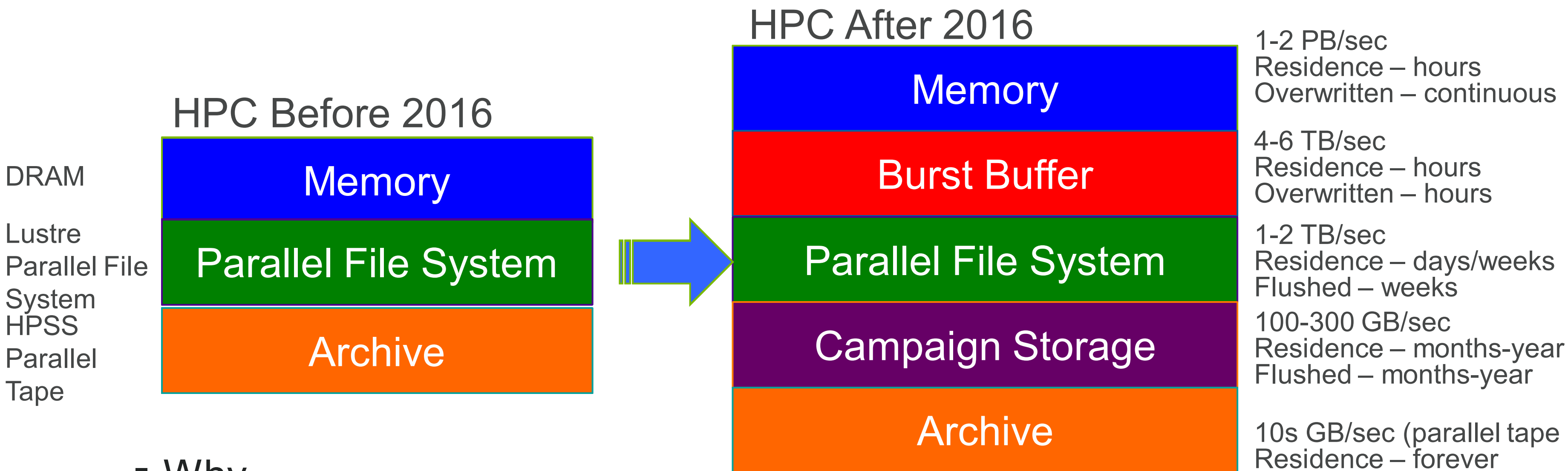
ROB ROSS, PHILIP CARNS, KEVIN HARMS, JOHN JENKINS, AND SHANE SNYDER — Argonne National Laboratory

GARTH GIBSON, CHUCK CRANOR, AND QING ZHENG — Carnegie Mellon University

JEROME SOUMAGNE AND JOE LEE — The HDF Group

GALEN SHIPMAN AND BRAD SETTLEMYER — Los Alamos National Laboratory

ARCHITECTURAL TRENDS: MORE STORAGE/MEMORY LAYERS…



- Why
  - BB: Economics (disk bw/iops too expensive)
  - PFS: Maturity and BB capacity too small
  - Campaign: Economics (tape bw too expensive)
  - Archive: Maturity and we really do need a “forever”

Slide from Gary Grider (LANL).

WHAT COMES NEXT?

- Assumptions
  - New layers in storage hierarchy, lower latencies
  - Storage resources will be highly contended for
  - No “holy grail” emerges that solves everyone’s problems

- Alternative model to the “PFS for data management”
  - Multiple services employed for different classes of data
  - Specialization for scalability/efficiency/productivity
  - In some cases, co-design with applications

Specialized data services are already here!

	Provisioning	Comm.	Local Storage	Fault Mgmt. and Group Membership	Security
ADLB *Data store and pub/sub.*	MPI ranks	MPI	RAM	N/A	N/A
DataSpaces *Data store and pub/sub.*	Indep. job	Dart	RAM (SSD)	Under devel.	N/A
DataWarp *Burst Buffer mgmt.*	Admin./ sched.	DVS/ lnet	XFS, SSD	Ext. monitor	Kernel, lnet
FTI *Checkpoint/restart mgmt.*	MPI ranks	MPI	RAM, SSD	N/A	N/A
Kelpie *Dist. in-mem. key/val store*	MPI ranks	Nessie	RAM (Object)	N/A	Obfusc. IDs
SPINDLE *Exec. and library mgmt.*	Launch MON	TCP	RAMdisk	N/A	Shared secret

OUR GOAL

Enable composition of data services for DOE science and systems

- Application-driven
  - Identify and match to science needs
  - Traditional data roles (e.g., checkpoint, data migration)
  - New roles (e.g., equation of state/opacity databases)
- Composition
  - Develop/adapt building blocks
    - Communication
    - Concurrency
    - Local Storage
    - Resilience
    - Authentication/Authorization
- Enable rapid development of specialized services
- ***Don't built new services from scratch every time***

SERVICE COMPONENTS AND MERCURY INTEGRATION

ROB ROSS, PHILIP CARNS, KEVIN HARMS, JOHN JENKINS, AND SHANE SNYDER — Argonne National Laboratory

GARTH GIBSON, CHUCK CRANOR, AND QING ZHENG — Carnegie Mellon University

JEROME SOUMAGNE AND JOE LEE — The HDF Group

GALEN SHIPMAN AND BRAD SETTLEMYER — Los Alamos National Laboratory

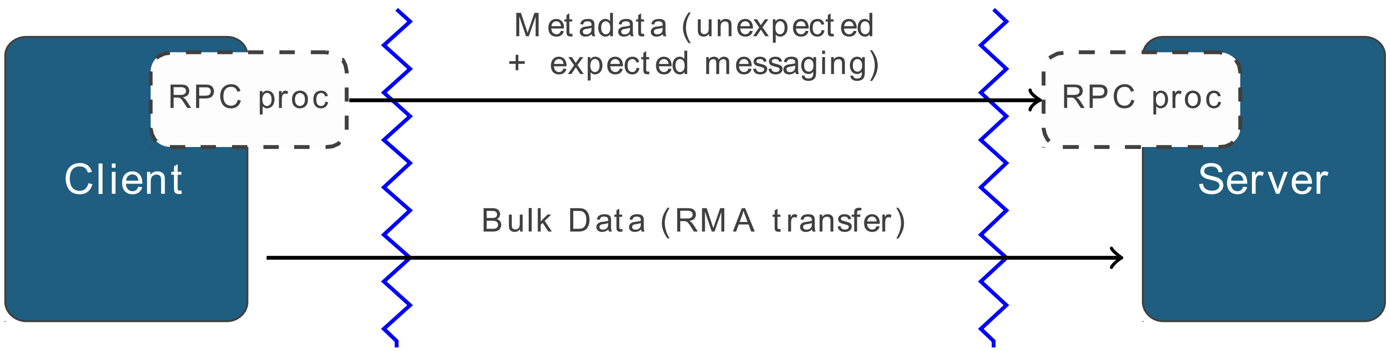
Argonne NATIONAL LABORATORY

COMMUNICATION: MERCURY

https://mercury-hpc.github.io/

The Mercury RPC system is the core building block of all of the services we are building in the Mochi project.

- Portable across systems and network technologies
- Builds on lessons learned from IOFSL, Nessie, lnet, and others
- Efficient bulk data movement to complement control messages



PROGRAMMING MODELS FOR MERCURY SERVICES

Concurrency, low latency, and ease of use

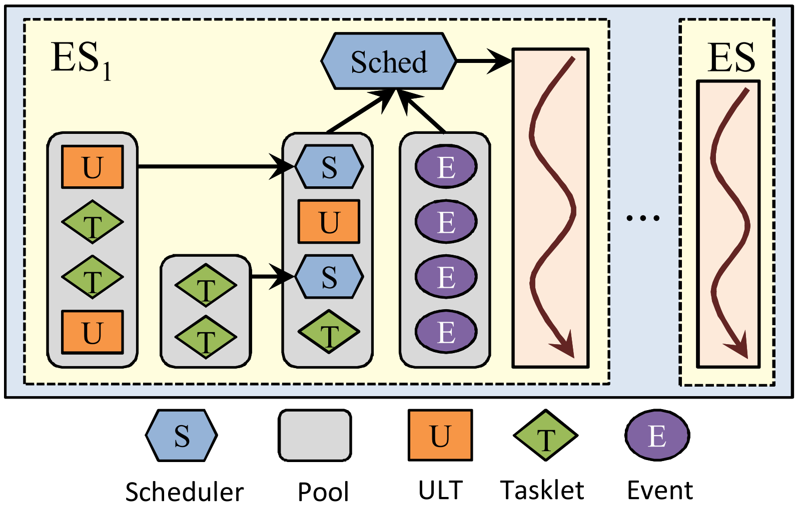
- The Mercury API uses an event-driven callback model that performs well, but we need more than that to build flexible services rapidly
  - Programmability of callback continuation (i.e. avoid stack-ripping)
  - Assignment of work to CPU cores
  - Scheduling for extremely high concurrency
- We've elected to combine Mercury with ***user-level threading***
  - Similar to co-routines, green threads, fibres, etc.
  - Hide callbacks from service developer
  - Combine latency benefits of Mercury's event model with programmability of conventional thread model

CONCURRENCY: ARGOBOTS

Argobots is a lightweight threading/tasking framework

- Features relevant to I/O services:
  - Flexible mapping of work to hardware resources *(which ULTs can run on which cores)*
  - Ability to delegate service work with fine granularity across those resources *(lightweight tasklets and ULTs)*
  - Modular, user-definable scheduling
- We developed asynchronous bindings to:
  - Mercury
  - LevelDB
  - POSIX I/O
- Working with Argobots team to identify needed functionality (e.g., idling) for our use cases

Argobots Execution Model



MARGO

Binding Mercury to Argobots

- Margo is a support library that provides Argobots-aware bindings to Mercury
  - Mercury operations are normally expressed with a post/callback model
  - Margo instead provides blocking functions that will suspend and resume the calling ULT automatically
- Internal progress loop drives Mercury progress and callbacks
  - Can run on dedicated core/thread or share resources of caller
- Incoming RPC handlers are executed as new user level threads
- "_timed()" versions of each function can be used to specify timeouts for cancellation/completion

WHAT DOES THE CODE LOOK LIKE?

A few Margo examples

- Macros are used to define wrappers for RPC handlers that will be launched as user-level-threads
  https://xgitlab.cels.anl.gov/sds/margo/blob/master/examples/my-rpc.c#L111
- Services code (both clients and servers) issue Mercury calls as blocking operations
  https://xgitlab.cels.anl.gov/sds/margo/blob/master/examples/my-rpc.c#L58
- Explicit concurrency is achieved by creating and joining user-level threads
  https://xgitlab.cels.anl.gov/sds/margo/blob/master/examples/client.c#L137
- Boilerplate: after starting Mercury library, set up Argobots and tell Margo what resources to use for driving progress and running RPC handlers
  https://xgitlab.cels.anl.gov/sds/margo/blob/master/examples/server.c#L53

HIGH-LEVEL TUNING OBSERVATIONS

Observations from Argobots/Margo/Mercury/CCI stack

- CPU usage
  - Trade-off between latency and how frequently you are willing to busy-spin on a CPU core
  - CCI level
    - Reduce CPU usage by using CCI poll feature -DNA_CCI_USE_POLL:BOOL=ON), recently upstreamed in Mercury
  - Argobots level
    - Reduce CPU usage with abt-snoozer scheduler (optional add-on for use with Margo and other support libraries that block on I/O)
- Bandwidth
  - So far so good for large transfers

HIGH-LEVEL TUNING OBSERVATIONS (CONT)

Observations from Argobots/Margo/Mercury/CCI stack

- Latency
  - Pay close attention to callback scheduling and context switching points in progress loop (Margo in our case)
  - Disable checksumming in Mercury (-DMERCURY_USE_CHECKSUMS:BOOL=OFF)
  - Tcmalloc reduces memory management latency
- Ongoing
  - Isolating sources of variability
  - Investigating memory allocation and reuse
  - Testing more scenarios

We need to standardize on how to measure and tune more of these things…

MERCURY DEVELOPER'S MEETING:
ANL EXAMPLES AND BENCHMARKING

JOHN JENKINS jenkins@mcs.anl.gov
PHIL CARNS carns@mcs.anl.gov
Argonne National Laboratory

July 14, 2016

DEMONSTRATING OUR APPROACH

Argonne NATIONAL LABORATORY

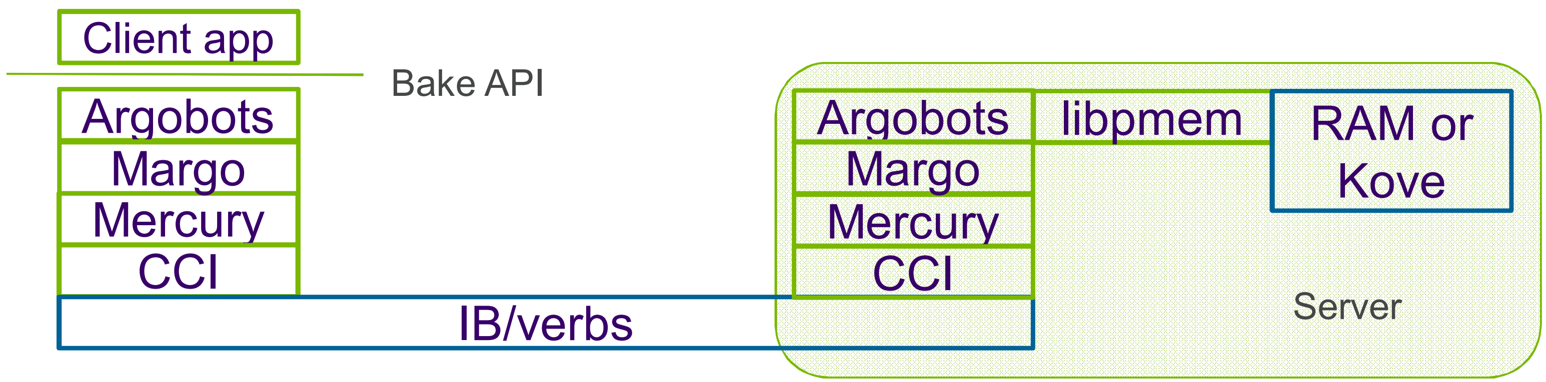
DEPLOYMENT SCENARIOS

When, where, and how to run composable data services

- Use cases will depend on both the system and the application, so we need to be flexible
- Examples:
  - Co-located alongside (or even within) application processes
  - Set-aside nodes in a single job
  - Co-scheduled across jobs
  - Persistent (or semi-persistent) services
- Storage may reside on compute nodes, burst buffer nodes, or off-system
- Duration may be transient, on-demand, or long-running
- Some of these scenarios will rely on resource management integration

BUILDING BLOCKS FOR STORAGE SERVICES

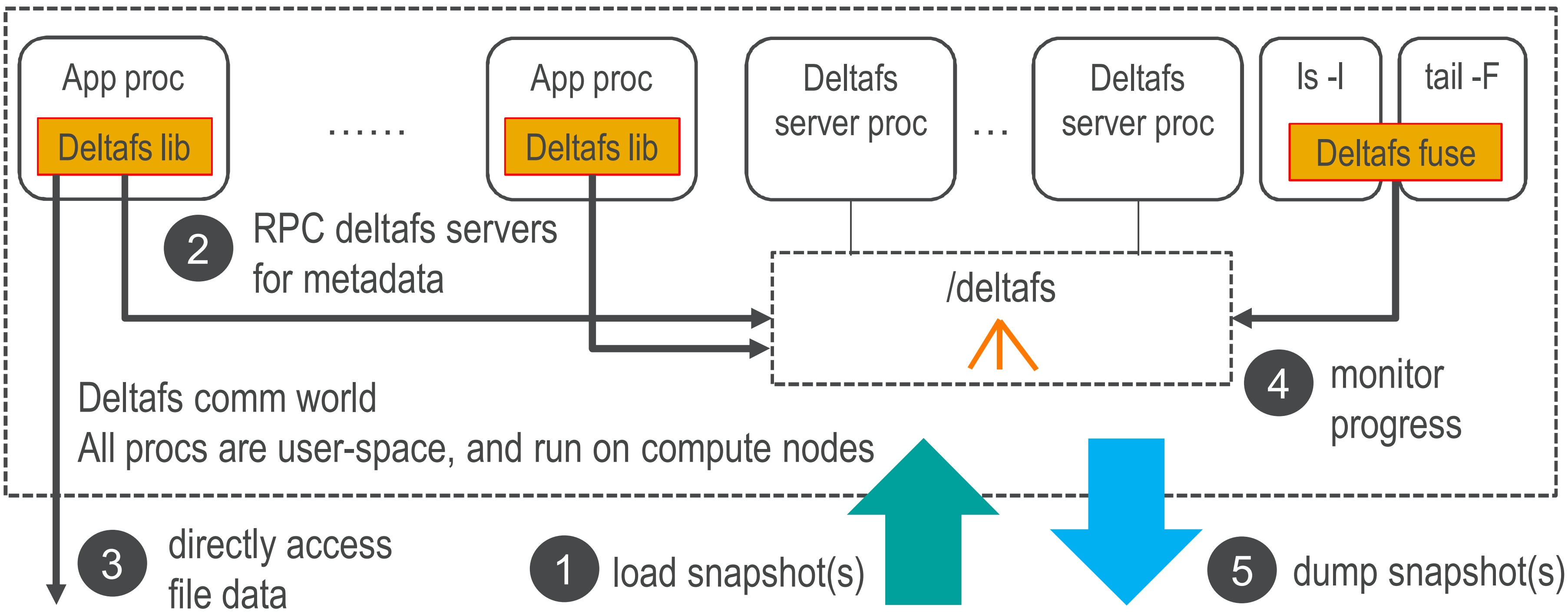
- In-progress low-level storage service, “BAKE”, to serve as a basis for future data services



- Looks like a lot of components
- … but data copies and context switches are rare
- RDMA transfer directly from client app address space to storage device that has been mapped into server address space
- Argobots/Margo/Mercury handle concurrency without any OS threads

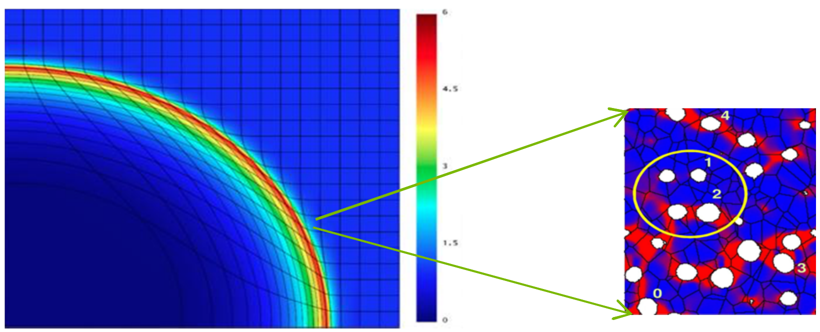
TRANSIENT FILE SYSTEM VIEWS: DELTAFS

Supporting legacy POSIX I/O in a scalable way.



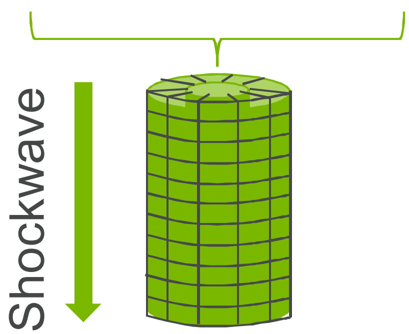
Credit: Qing Liu, CMU

CO-DESIGNING COUPLED MODELS



Lulesh continuum model:
- Lagrangian hydro dynamics
- Unstructured mesh

Viscoplasticity model [1]:
- FFT based PDE solver
- Structured sub-mesh

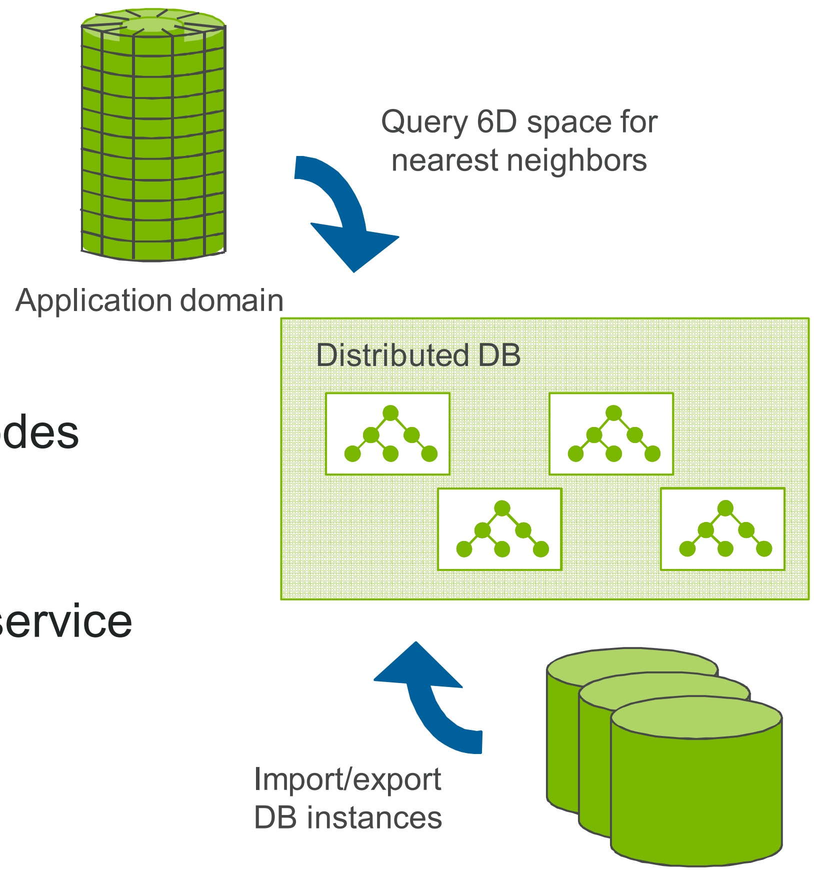


- Future applications are exploring the use of multi-scale modeling
- As an example: Loosely coupling continuum scale models with more realistic constitutive/response properties
  - e.g., Lulesh from ExMatEx
- Fine scale model results can be cached and new values interpolated from similar prior model calculations

R. Lebensohn et al, Modeling void growth in polycrystalline materials, Acta Materialia, http://dx.doi.org/10.1016/j.actamat.2013.08.004.

CO-DESIGNING A FINE SCALE MODEL DATABASE

- Goals
  - Minimize fine scale model executions
  - Minimize query/response time
  - Load balance DB distribution
- Approach
  - Distributed approx. nearest-neighbor query
  - Partitioned spatial data structures across nodes
  - Import/export to persistent store
- Status
  - Mercury-based, centralized in-memory DB service
  - Investigating distributed, incremental nearest-neighbor indexing



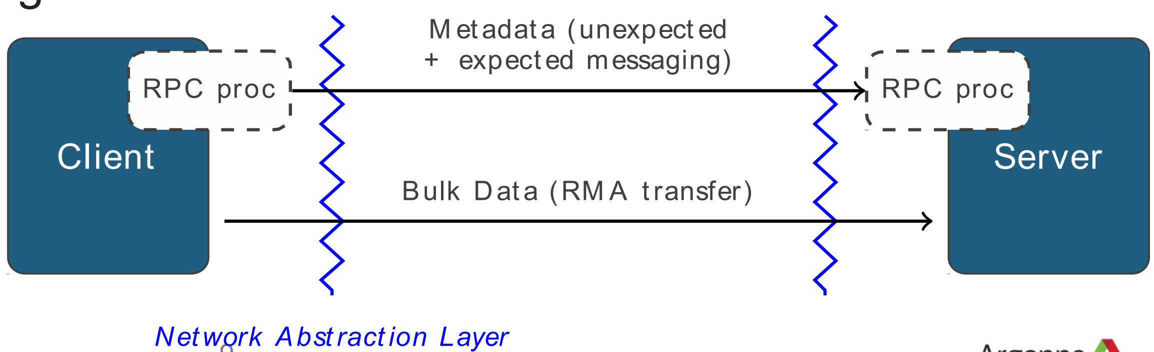
Argonne NATIONAL LABORATORY

BENCHMARKING

MICROBENCHMARKS

Initial code at https://github.com/mercury-hpc/mercury-benchmarks

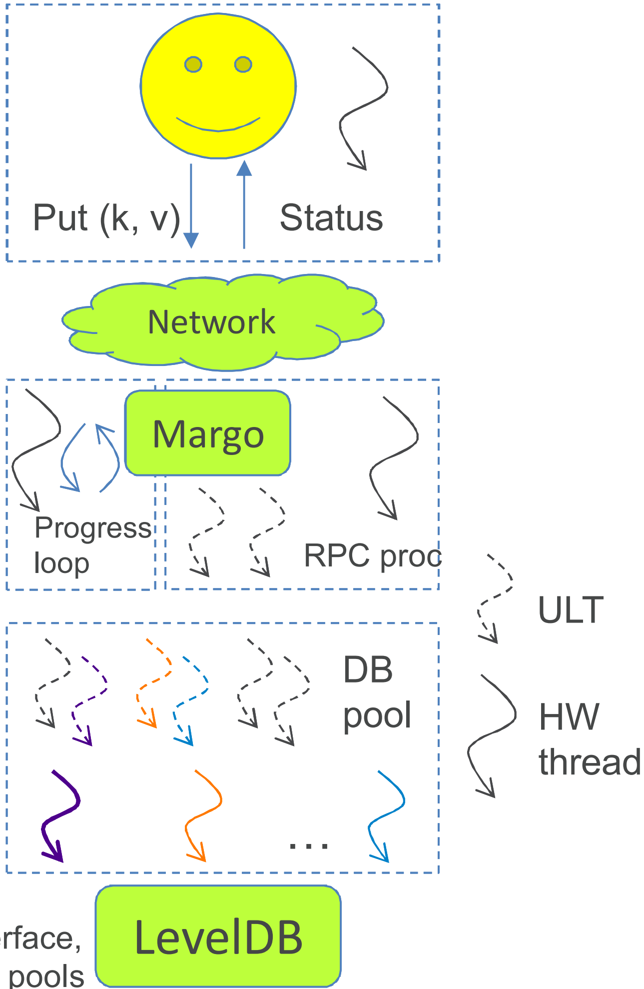
- RPC round-trip latency
- Bulk transfer rate
- Concurrent RPCs/bulks
- Overheads (not yet directly measured)
  - RPC data (un)marshalling
  - HG/NA work queue management
  - NA → plugin translation



BENCHMARKING COMMON DESIGN PATTERNS

Capturing Mercury's surrounding ecosystem

- Example: LevelDB wrapper
- Mixed RPC+bulk workflow
  - k/v pair represented as a bulk handle
  - Server receives "put" RPC, RDMA reads using k/v handle
- Multithreading scenarios
  - HG progress, DB op setup / dispatch
- Interaction with other layers
  - Argobots ULT creation, context switching, thread handoff



NOTE: LevelDB doesn't have an async interface, otherwise could unify the RPC proc and DB pools

CONCLUDING REMARKS

Argonne NATIONAL LABORATORY

TOPICS FOR AFTERNOON DISCUSSION

- Multi-user environments (secure messaging)
- Group membership (dynamic, fault tolerant service participation)
- Transport roadmaps (libfabric)
  – currently using BMI from PVFS, CCI as prototyping vehicles
- Standardizing set of benchmarks?
- Leveraging LNET experience

RESOURCES

- Mercury: https://github.com/mercury-hpc/mercury
- Mercury benchmarks: https://github.com/mercury-hpc/mercury-benchmarks
- Argobots: https://collab.cels.anl.gov/display/ARGOBOTS/Argobots+Home
- Margo: https://xgitlab.cels.anl.gov/sds/margo
- abt-snoozer: https://xgitlab.cels.anl.gov/sds/abt-snoozer

THIS WORK IS SUPPORTED BY THE DIRECTOR, OFFICE OF ADVANCED SCIENTIFIC COMPUTING RESEARCH, OFFICE OF SCIENCE, OF THE U.S. DEPARTMENT OF ENERGY UNDER CONTRACT NO. DE-AC02-06CH11357.

Argonne NATIONAL LABORATORY