# Current and Planned Development Efforts of Interest to PDT Partners, LLC

# Parallel I/O Efforts

## [Funded] Multi-dataset read/write routines

### Enables a single collective I/O operation on multiple datasets

## [Funded] Property to control independent/collective metadata I/O

### Cuts *way* down on small file I/O for metadata operations

## [Not Funded] Support for writing to compressed datasets with parallel programs

### We know how to make this work for many/most scenarios, just needs funding

## [Not Funded] Support for independent metadata medication operations

### Prototype done, just needs productization

### Uses the new Virtual Object Layer (VOL) interface

## [Not Funded] Avoiding truncates in parallel

### Truncation operations in parallel are very expensive, this change modifies the file format to avoid them

### Needs review & revise, basically finished

# Concurrent I/O Efforts

## [Partially Funded] SWMR

### Background

#### Requires POSIX-compliant file system (namely the atomicity and ordering requirements)

#### NFS will definitely never work

#### Have seen POSIX compliant support on these systems: FreeBSD (UFS2), Mac OSX (?), Linux+GPFS.

#### Currently Linux+ext3/4 does not appear to be POSIX compliant

#### Linux+Lustre may be POSIX compliant, still evaluating

### Funding for “Phase 0” efforts to estimate productizing dataset append operations

### Discussions with another customer to fund “Phase 0” effort for metadata create/append operations

### [Not-funded/planned currently] Support for deleting metadata or shrinking datasets

#### We know how to do this, just doesn’t seem to be demanded yet

## [Funded] Client/Server Network Access

### Funded effort to deliver client/server extensions to allow access to remote data (on server), using TCP/IP

### Can support multiple readers and multiple writers (but data must be shipped to server)

### Uses Virtual Object Layer Interface

## [Not Funded] Multi-Writers to File

### Could use Client/Server solution, but might be too slow

### Could build off of SWMR work, but may require a server to handle fully general solution

### Could avoid server for special circumstances

# Performance Efforts

## [Funded] Metadata aggregation

### Allocate all metadata in file in pages, to avoid small I/O operations

### Needs review/revise, basically finished

### Also needs Page Buffering feature (below) to leverage into high performance

## [Not Funded] Page Buffering

### Access all data in file with at least a page-sized I/O operation

### Prototype evaluation shows that this can eliminate 85+% of file I/O, for metadata intensive operations

## [Not Funded] Scalable Chunk Indices

### Make common dataset chunk configurations have O(1) lookup/append operations

### Needs review/revise, basically finished

## [Not Funded] Asynchronous Low-Level I/O

### Support asynchronous HDF5 I/O operations, enabling overlap of compute & I/O

### Needs revisions, mostly finished

## [Not Funded] Asynchronous API-level Operations

### Allow application by implementing asynchronous HDF5 API calls

## [Not Funded] “Append-only” Property for Writing Data

### Can enable speedups for writing data, when random edits are disallowed

# Multi-Threading

## [Partially Funded] Internal Threading for data parallel operations

### [Funded] Compress dataset chunks with multiple threads

### [Not Funded] Perform datatype conversions, and other operations with multiple threads

## [Not Funded] Improve Library Concurrency

### Convert from single lock on HDF5 library to locks on internal data structures

### Large effort

# Fault Tolerance

## [Not Funded] Journaled HDF5 files

### Protects against application crashes by journaling changes to HDF5 metadata

### Needs revisions, mostly finished

## [Side-effect of SWMR] Ordered Updates to HDF5 files

### Protects against application crashes when making changes to file (but limited to SWMR-safe operations, as they are implemented)

### Free side-effect of SWMR work, for writer

# Other Efforts

## [Done] Persistent free-space tracking in file

### Useful to save space, when modifying file over many open/close cycles

### Finished, ready for 1.10 release

## [Not Funded] Improve String Operations

### Improve storage of variable-length strings by storing all variable-length elements in local storage, for each dataset

#### Infrastructure in place, just needs design & implementation

### Full support for UTF-8 Strings

#### Support for UTF-8 filenames, external links, field names in compound structs, etc.