HDF5 I/O Enhancements for Wire-Cell Toolkit

Brett + Claude

2026-02-05

Outline

Overview

Feature 1: Compression and Chunking (wct-6c8)

Feature 2: TensorSink Component (wct-6er)

Feature 3: HDF5 Links (wct-7cl)

Results and Artifacts

Technical Highlights

Lessons Learned

Conclusion

Session Goals

Three major enhancements to the HIO package:

1. Add gzip compression and chunking to HDF5 datasets
2. Create a configurable `ITensorSetSink` component
3. Implement HDF5 linking for meaningful data paths

Motivation

- Compression: Reduce file sizes for large tensor datasets
- Chunking: Optimize HDF5 performance with smart chunk size selection
- Component: Provide a reusable sink for data flow graphs
- Linking: Create human-readable paths that link to actual storage locations

Requirements

Add optional gzip compression and chunking to all HDF5 write operations:

- `gzip` parameter: compression level 0-9 (default: 0)
- `chunks` parameter: user-specified chunk sizes (default: auto)
- Backward compatible through default parameters
- Smart chunk size computation

Implementation Strategy

Added three key components:

1. `compute_chunks()` function
   - Implements specified algorithm
   - Targets 1MB chunks by default
   - Prefers full last dimension (row-major)
   - Fills second-to-last dimension to reach target
2. Extended `write_dataset()` signatures
   - Both raw bytes and template versions
   - Added gzip and chunks parameters
3. Extended tensor write functions
   - `write_itensor()` and `write_itensorset()`
   - Pass through compression parameters

Chunk Size Algorithm

Key design decisions based on discussion:

- Target size: 1MB chunks (good compression efficiency)
- Last dimension: Keep whole if possible (C-order/row-major)
- Second-to-last: Fill to reach target size
- Earlier dimensions: Set to 1

Example: Shape `(10, 8, 1200, 9600)` with 8-byte elements

- Chunks: `(1, 1, 13, 9600)` $\rightarrow$ ~975KB per chunk

Code Changes

Files modified:

- `hio/inc/WireCellHio/HIO.h` - Added `compute_chunks()`, extended signatures
- `hio/src/HIO.cxx` - Implementation with HDF5 deflate filter
- `hio/inc/WireCellHio/Tensors.h` - Extended tensor write signatures
- `hio/src/Tensors.cxx` - Pass through compression parameters
- `hio/test/doctest_hio.cxx` - Added 7 new tests

Test Coverage

All tests passing (36/36):

- `compute_chunks` with various inputs
- Compressed dataset write/read roundtrip
- Custom chunk sizes
- Compression level clamping ([-5,15] $\rightarrow$ [0,9])
- Multi-dimensional arrays with compression

Requirements

Create `HioTensorSink` component:

- Implements `ITensorSetSink`, `IConfigurable`, `INamed`
- Uses boost hana configuration pattern
- Configurable: filename, gzip, chunks, datapath pattern
- Handles EOS (end-of-stream) properly

Configuration Structure

```
struct TensorSinkConfig {
    std::string filename = "";          // Required
    int gzip = 0;                       // Compression level
    std::vector<int> chunks = {};       // Chunk sizes
    std::string datapath_pattern =
        "tensorsets/{ident}";           // Path template
    bool links = true;                  // Create links
};
```

Adapted with `BOOST_HANA_ADAPT_STRUCT` for automatic serialization.

Implementation Details

Key design elements:

- Opens file in truncate mode during `configure()`
- Validates filename is not empty
- Formats datapath using `Fmt::format()` with metadata + ident
- Calls `write_itensorset()` with compression parameters
- Closes file on EOS (nullptr input)
- Factory registered as "HioTensorSink"

Code Artifacts

New files created:

- `hio/inc/WireCellHio/TensorSink.h` - Interface and config
- `hio/src/TensorSink.cxx` - Implementation

Tests modified:

- `hio/test/doctest_hio_tensors.cxx` - Added 5 TensorSink tests

Problems Encountered

Factory Loading Issue:

- Tests fail with "No factory for class HioTensorSink"
- Factory registration code is correct
- Issue: Plugin not loaded in doctest harness
- Workaround: Added `PluginManager::instance().add("WireCellHio")`
- Tests now pass after plugin loading

Note: This is a test infrastructure issue, not implementation problem.

Requirements

Create meaningful HDF5 links from metadata "datapath" entries:

1. Add `write_link(src, dst)` to create HDF5 hard links
2. Add `link_itensor()` - links tensor if metadata has "datapath"
3. Add `link_itensorset()` - links set and all its tensors
4. Extend TensorSink with `links` configuration option

Motivation for Links

Problem: Tensors stored with numeric indices lack semantic meaning

- Actual storage: `/tensorsets/42/0`, `/tensorsets/42/1`
- Desired access: `/frames/0/raw`, `/frames/0/deconvolved`

Solution: HDF5 hard links

- Metadata "datapath" provides meaningful name
- Link from meaningful path to actual storage
- Both paths access same data (no duplication)

Implementation

Three-layer approach:

1. Base function: `write_link(file_id, src, dst)`
   - Creates hard link using `H5Lcreate_hard()`
   - Ensures parent groups exist
2. Tensor function: `link_itensor(file_id, tensor, dst)`
   - Checks for "datapath" in metadata
   - Creates link if present, silent otherwise
3. TensorSet function: `link_itensorset(file_id, tensorset, dst)`
   - Links the set itself
   - Iterates and links each tensor

Integration with TensorSink

Added `links` boolean to `TensorSinkConfig`:

- Default: `true` (links enabled)
- After writing tensorset, conditionally call `link_itensorset()`

```
write_itensorset(m_file_id, in, datapath,
                 m_config.gzip, m_config.chunks);

if (m_config.links) {
    link_itensorset(m_file_id, in, datapath);
}
```

Code Changes

Files modified:

- `hio/inc/WireCellHio/HIO.h` - Added write$_{\text{link}}$()
- `hio/src/HIO.cxx` - Implementation
- `hio/inc/WireCellHio/Tensors.h` - Added link functions
- `hio/src/Tensors.cxx` - Implementation
- `hio/inc/WireCellHio/TensorSink.h` - Added links config
- `hio/src/TensorSink.cxx` - Call link$_{\text{itensorset}}$()
- `hio/test/doctest_hio_tensors.cxx` - Added 5 link tests

Test Coverage

All tests passing (53/53 total):

- Link creation from metadata
- No-op when metadata absent
- TensorSet with multiple tensor links
- TensorSink with links enabled
- TensorSink with links disabled

Test Results Summary

Feature	Tests Added	Status
Compression/Chunking	7	Pass
TensorSink Component	5	Pass
HDF5 Links	5	Pass
Total	17	53/53

All doctests pass successfully!

Files Created

New components:

- `hio/inc/WireCellHio/TensorSink.h`
- `hio/src/TensorSink.cxx`

Files Modified

Core functionality:

- `hio/inc/WireCellHio/HIO.h`
- `hio/src/HIO.cxx`
- `hio/inc/WireCellHio/Tensors.h`
- `hio/src/Tensors.cxx`

Test coverage:

- `hio/test/doctest_hio.cxx`
- `hio/test/doctest_hio_tensors.cxx`

Issues Closed

All beads issues successfully completed:

- wct-6c8: Add gzip and chunking support
- wct-6er: Create ITensorSetSink component
- wct-7cl: Add HDF5 linking functionality

Updated with detailed implementation notes.

Smart Chunk Size Selection

The `compute_chunks()` algorithm provides intelligent defaults:

1. Empty input: Full last dimension
2. One value: Apply to last dim, compute second-to-last
3. Multiple values: Apply to last dims, prepend 1s
4. Always: Clamp to actual dimension sizes

Target: ~1MB chunks for optimal compression efficiency.

Backward Compatibility

All changes maintain backward compatibility:

- Optional parameters with sensible defaults
- `gzip=0` $\rightarrow$ no compression (existing behavior)
- `chunks={}` $\rightarrow$ automatic chunk selection
- `links=true` $\rightarrow$ links created only if metadata present

Existing code continues to work unchanged.

Boost Hana Configuration Pattern

TensorSink demonstrates modern configuration approach:

```
BOOST_HANA_ADAPT_STRUCT(TensorSinkConfig,
    filename, gzip, chunks,
    datapath_pattern, links);
```

Benefits:

- Automatic JSON serialization
- Type-safe configuration
- Easy to extend with new parameters

Factory Pattern Integration

Component properly registered with Wire-Cell factory system:

```
WIRECELL_FACTORY(HioTensorSink,
    WireCell::Hio::TensorSink,
    WireCell::ITensorSetSink,
    WireCell::IConfigurable,
    WireCell::INamed);
```

Enables dynamic instantiation in data flow graphs.

HDF5 Best Practices

Key insights from implementation:

1. Chunking required for compression - HDF5 deflate needs chunked storage
2. Property lists - Use `H5Pcreate(H5P_DATASET_CREATE)` for compression
3. Hard links - Use `H5Lcreate_hard()` for aliasing without duplication
4. Parent groups - Always ensure parent paths exist before creating objects

Design Patterns

Successful patterns applied:

- Optional parameters: Provide sensible defaults
- Graceful degradation: Missing metadata $\rightarrow$ no links created
- Separation of concerns: Base functions $\rightarrow$ higher-level wrappers
- Comprehensive testing: Test happy path, edge cases, and error conditions

Testing Strategy

Effective test design:

- Unit tests: Individual function behavior
- Integration tests: TensorSink with file I/O
- Roundtrip tests: Write then read verification
- Edge cases: Empty metadata, invalid parameters
- Plugin loading: Explicit setup in test harness

Summary of Achievements

Three significant enhancements delivered:

1. Compression and chunking for reduced file sizes
2. Reusable TensorSink component for data flow
3. HDF5 linking for semantic data access

All with comprehensive test coverage and documentation.

Impact

Benefits to Wire-Cell Toolkit:

- Storage efficiency: Compressed HDF5 datasets
- Performance: Optimized chunk sizes
- Usability: Meaningful data paths via links
- Flexibility: Configurable sink component
- Maintainability: Well-tested, documented code

Future Work

Potential enhancements:

- Performance benchmarking of compression levels
- Additional chunk size heuristics for different access patterns
- Soft links vs hard links for external file references
- Parallel HDF5 I/O for multi-threaded applications
- Additional sink components (streaming, network, etc.)

Thank You

Questions?

- Repository: Wire-Cell Toolkit `spng` branch
- Package: `hio/` (HDF5 I/O wrapper)
- Tests: `./build/hio/wcdoctest-hio`

All code ready for ~~review and integration~~ committed.