Best Practices for Storing Your Data

Congyue Cui 01/21/2026

Overview

Motivation

- Reduce the risk of data loss
- Speed up your research
- Increase visibility and collaboration
- Be kind to your future self

Overview

Contents

- General advice
- Storage strategies
- Hands-on
- Slides and code will be available

General Advice

General Advice

Storage data at the right place

- Raw data
- Intermediate data
- Data product
- Archival data

General Advice

Storage data at the right place

- /home: source code, executables, etc.
- /scratch: job input / output
- /projects: final job output (long term)
- /tigerdata: long term storage and for sharing

General Advice

Storage data at the right place

- TigerData
  - Requires sponsorship
  - Not available on the compute nodes
  - Organized by projects
  - https://tigerdata.princeton.edu/how-it-works

General Advice

Princeton Research Data Service (PRDS)

Data Services

- Review **data management plans** (DMPs)
- Advise on federal **funder requirements**
- Consult on file naming conventions and data organization
- **Data curation** assistance (no matter which repository you use!)
- Assistance preparing data for **publication**
- Help find an appropriate **repository**
- Help with **data documentation** and **READMEs**
- Mint **DOIs** for datasets
- **Large data** support (Globus access point)
- Hosts **Princeton Data Commons**

Princeton Research Data Service

Scan to make an appointment
prds@princeton.edu
https://researchdata.princeton.edu/

General Advice

Data safety

- Storage data at 3 places (one original and two copies)
- Two types of storage media (e.g. local disk and Princeton cluster)
- One copy off-site (e.g. cloud storage or external data centers)

General Advice

Scalibility

- Be prepared for larger dataset in the future
- Too many files can crash the file system
- Keep track of storage usage
- Compress data before archiving
- Discard easily reproducible data

General Advice

Reproducibility

- Keep raw data safe
- Document processing steps
- Save processing parameters alongside data
    - Software version may be important
- Append data instead of overwriting if possible
- Perform tests

General Advice

Shareability

- Use mainstream format if possible
- Bundle small files
- Split large files
- Document non-standard notations

Storage strategies

Storage strategies

Formats to consider

- Text
  - CSV - Flat tabular data
  - JSON - Nested metadata, parameters
- Binary
  - Raw binary - Temporary inter-process data
  - NPY / NPZ - Array data
  - HDF5 - Large hierarchical data
  - ADIOS2 - Exascale data emphasizing parallel writing

Storage strategies

Formats to consider

- There is not a format that is best for all use cases
- Understand your needs and choose the right format for your project

Storage strategies

Formats to consider

- There is not a format that is best for all use cases
- Understand your needs and choose the right format for your project
- Case study: SPECFEM++

Storage strategies

Choosing a default format for SPECFEM++

- SPECFEM++ is a C++ numerical simulation software for elastic wave
- We need to choose a default format for storing simulated wavefield
- Requirements:
  - Standard
  - User-friendly
  - Scalable

Storage strategies

Choosing a default format for SPECFEM++

- Text?
  - Not a consideration for large data

Storage strategies

Choosing a default format for SPECFEM++

- HDF5?
  - Widely supported
  - Good for large data

Storage strategies

Choosing a default format for SPECFEM++

- HDF5?
  - Widely supported
  - Good for large data
  - Parallel write issues with a large number of processors
    - Metadata syncing is slow
  - Not always easy for users

Storage strategies

Choosing a default format for SPECFEM++

- ADIOS?
  - Smoother installation process
  - Good parallel write performance

Storage strategies

Choosing a default format for SPECFEM++

- ADIOS2?
  - Smoother installation process
  - Good parallel write performance
  - Official documentation is not so good
  - Not a necessity for most users
  - Limited ecosystem (C / MatLab / Python / Fortran)

Storage strategies

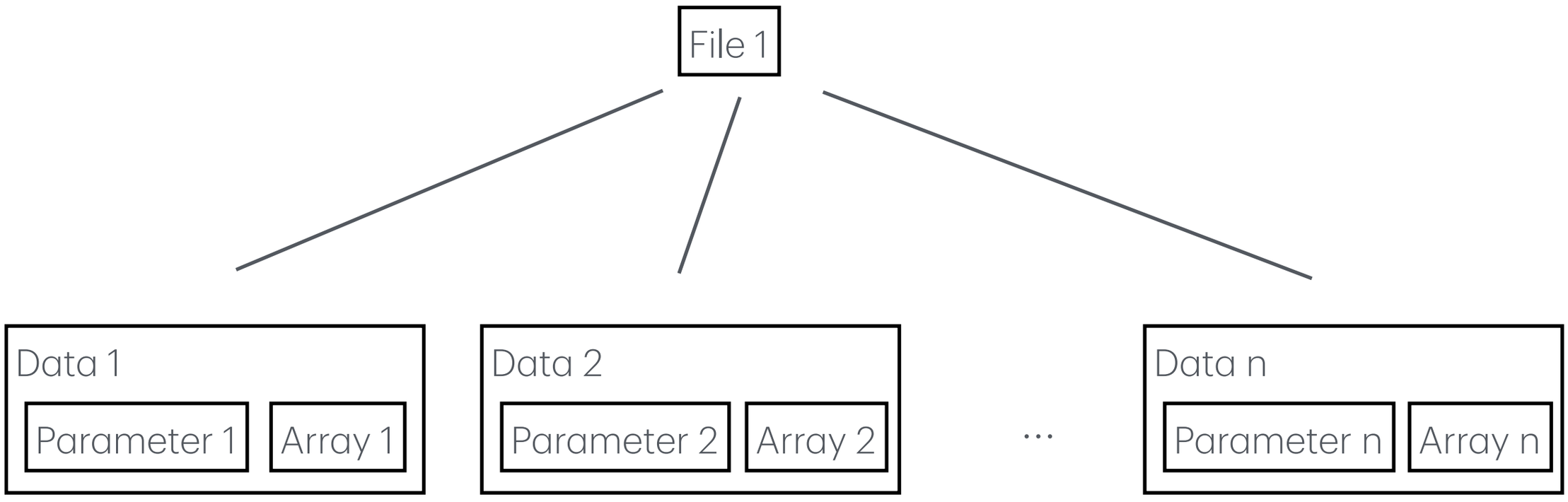
Choosing a default format for SPECFEM++

- NPY / NPZ
  - Simple - just raw binary + a small header
  - Widely supported (Not just Python)
  - Few dependency required (only zlib for NPZ)

Storage strategies

Separate parameter data from array data

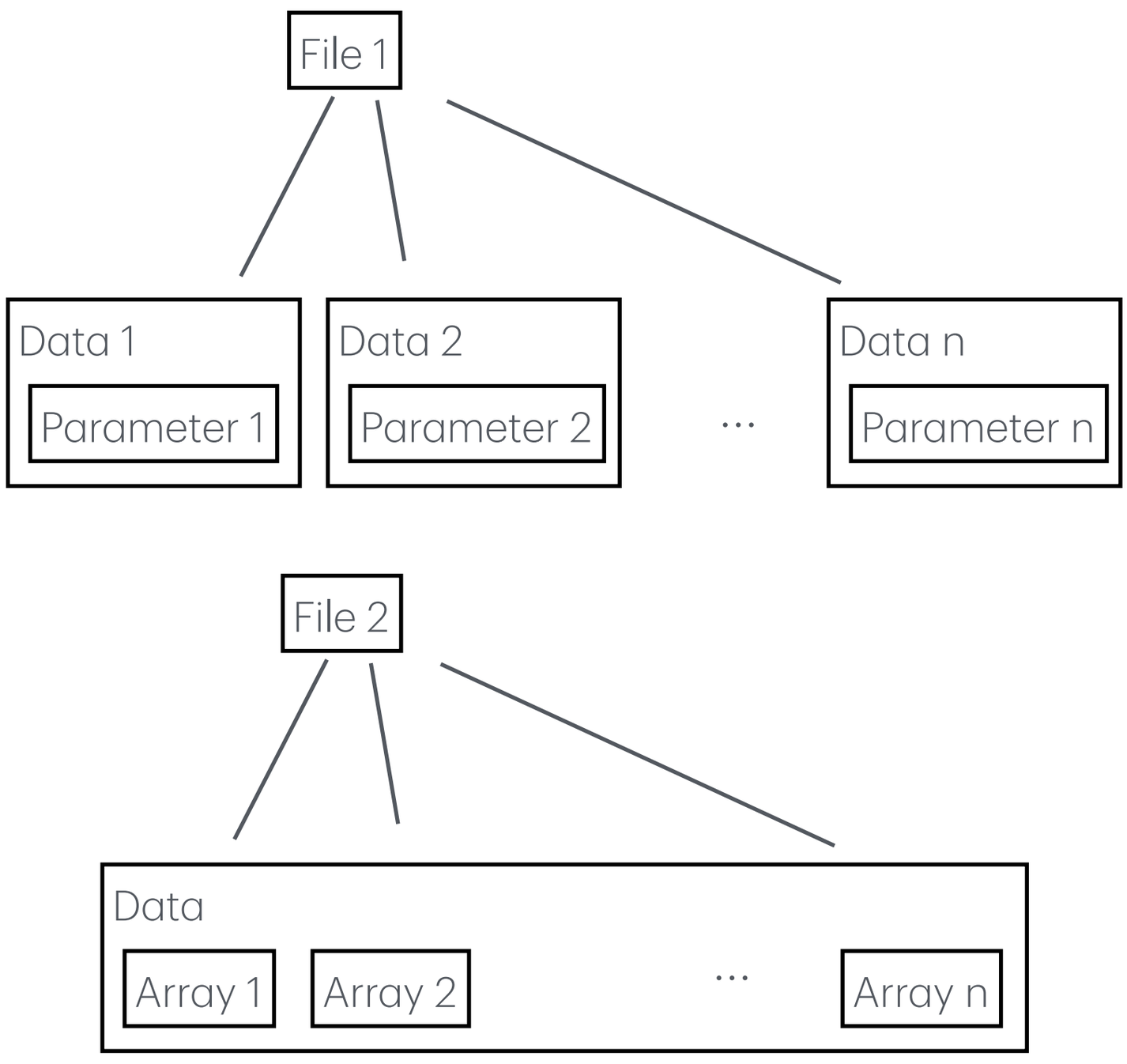
- Before



Storage strategies

Separate parameter data from array data

- After



Storage strategies

Separate parameter data from array data

- Pros
  - More efficient to process
  - Better scalability
  - Easy collaboration
- Cons
  - More complicated to use
  - Not self-contained
  - Potential out of sync

Hands-on

https://github.com/icui/data-workshop

Hands-on

Process an earthquake catalog

- Download a database of all earthquake between 1976 and 2020
- Explore storage options for a subset of the database
- Calculate some statistics for all earthquakes
- Calculate some values of each earthquake
- (What if we do these with MPI?)

Hands-on

Part 1: download and extract data

- https://www.globalcmt.org/CMTsearch.html
  (or search for Global CMT search)

- Click “various ASCII files for direct download”

Catalog in ASCII format

The CMT catalog and Quick CMTs are also available in various ASCII files for direct download.

- Right click link “all events 1976-2020” and copy url

The CMT catalog is available in the ASCII "ndk" format:

- all events 1976-2020 (or gzip compressed).

- Optionally, look at “description of the ndk format”

Note that the "ndk" format is slightly different from the older "dek" format. Please see the description of the ndk format.

Hands-on

Part 1: download and extract data

Copy the url from previous step

```
url='https://www.ldeo.columbia.edu/~gcmt/projects/CMT/catalog/jan76_dec20.ndk'
```

Python

```
import urllib.request

with urllib.request.urlopen(url) as response:
    data = response.read().decode('utf-8')
```

MatLab

```
options = weboptions('ContentType','text');
data = webread(url, options);
```

Hands-on

Part 1: download and extract data

- Extract the following data for processing
  - Event name
  - Event depth (km)
  - Moment tensor (6 components) with exponent
  - Scalar moment (M0)

```
MLI  1976/01/01 01:29:39.6 -28.61 -177.64  59.0 6.2 0.0 KERMADEC ISLANDS REGION
M010176A          B:  0    0   0 S:  0    0   0 M: 12   30 135 CMT: 1 BOXHD:  9.4
CENTROID:     13.8 0.2 -29.25 0.02 -176.96 0.01  47.8  0.6 FREE 0-00000000000000
26  7.680 0.090  0.090 0.060 -7.770 0.070  1.390 0.160  4.520 0.160 -3.260 0.060
V10   8.940 75 283   1.260  2  19 -10.190 15 110   9.560 202 30   93  18 60   88
```

Hands-on

Part 1: download and extract data

- Try it yourself
  - Go to repository https://github.com/icui/data-workshop
  - Finish “Part 1: download and extract data”

Hands-on

Part 1: download and extract data

- Try out these options

  1. Nested dictionary (JSON) `{"M010176A": {"Depth": ..., "M0": ...}}`
  2. Parameter (JSON) + raw array files (.npy or .m)

     `{"Rows": ["M010176A", ...], "Columns": ["Depth", "M0", ...]}`

     `Array2D[[...], [...]]`
  3. Key-value pairs (HDF5 or ADIOS2)

     - Event name as the key, e..g `{"M010176A": Array([...])}`
     - Data class as the key `{"Depth": Array[...], "M0": Array([...]), ...}`

Hands-on

Part 2: process saved data

- Calculate the min, max and median of earthquake depths
- Calculate the moment magnitude of each earthquake and count earthquakes with moment magnitude greater than 8.0

```
Mw = (2/3)*(log10(M0) - 16.1)
```

- Find the maximum absolute value among the moment tensor components of all earthquakes
- Save the results

Hands-on

Part 2: process saved data

- Try it yourself
  - Go to repository https://github.com/icui/data-workshop
  - Finish “Part 2:process saved data”

Hands-on

Part 2: process saved data

	Nested	Parameter + array	Key-value (by event)	Key-value (by data class)
Load time	0.17	0.0084	5.1	0.006
Compute time	0.15	0.0056	0.23	0.0023

Hands-on

Part 2: process saved data

- For a small set of arrays (e.g. 8 arrays in this case), use key value pairs stored as HDF5

- For a large set of arrays, store as separate parameters and array files or ADIOS2

Hands-on

Part 3: accounting for MPI

- Try it yourself
  - Go to repository https://github.com/icui/data-workshop
  - Redo Part 2 with MPI using
    - Parameter + array (two files per MPI process)
    - ADIOS2 (single file)

Hands-on

Part 3: accounting for MPI

- Small dataset: ADIOS2 or HDF5
- Large dataset: ADIOS2
- Performance critical: ADIOS2 or raw binary

Quick recap

Have you followed some good practices?

- Save raw data
- Document what is saved
- Document how Mw is calculated
- Choose reasonable file names

100%