NCEDC Data Pipeline

Jeffrey M. Church

Data Sources

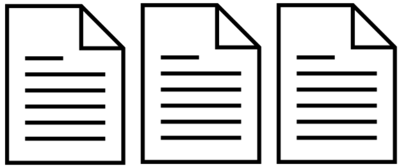
- This pipeline combines hand-picked phases from the Northern California Seismic System (NCSS) Earthquake Phase Catalog with continuous waveforms downloaded from the Northern California Earthquake Data Center (NCEDC) dataselect web service.
  - Phase picks range from 1966 – 2024, but **digital waveforms are available only from March 1984.**

Terminal command to download phase catalog:

- NCEDC documentation
- AWS CLI must be installed

```
>> aws s3 cp s3://ncedc-pds/event_phases . --recursive  --no-sign-request
```

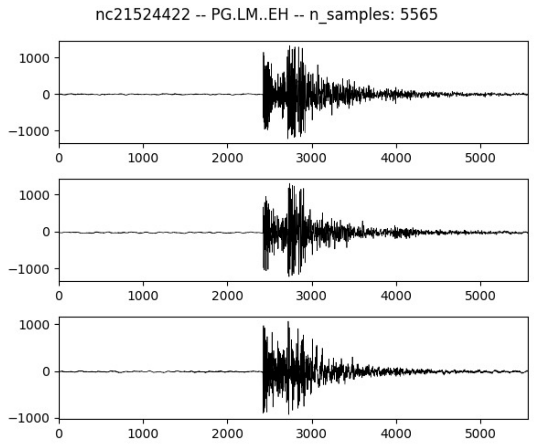
Hypoinverse phase pick files



{year}.{month}.phase (1966 - 2024)

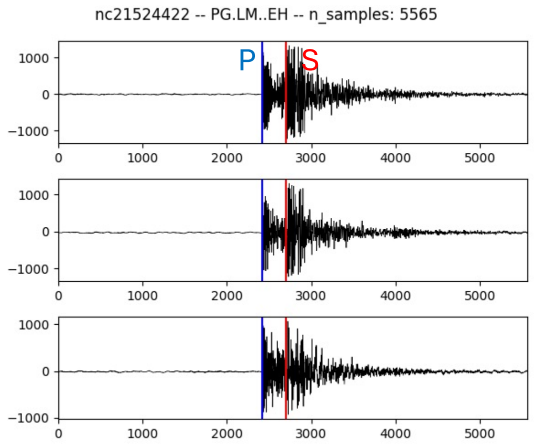
Corresponding waveforms are downloaded from the NCEDC dataselect web service via Obspy.

- In total, **1,367,579 3-component waveforms** and **444,271 Z-component "singletons"** must be downloaded, so bulk requests are necessary to reduce HTTP request overhead.
- With properly implemented bulk requests, this pipeline takes ~2-3 days to complete.



+

End Result



Phase Catalog (Hypoinverse files)

- Each hypoinverse (.phase) file downloaded from the NCSS catalog contains all recorded phase picks for a single month. The file name format is **{year}.{month}.phase**.
- Picks in each file are grouped by event.
  - Header line contain event ID and event time.
  - Pick lines represent a P pick, S pick, or both.
  - Information stored in each column may be found in the [NCEDC documentation]().
  - **Goal:** Find P and S pick pairs with matching event ID, station, and instrument.

Example event (ID: [21366046]()) from 2004.05.phase

```
200405250444382537 3228121 5335  745    10176  0   320944 103 6738  43 87      23    3  73  73   6      20     16       D 12                 21366046D ...
CDV  NC  EHZ IPU1200405250444 4237   5105 4543ES 2  -7          42   9  17 188101           80       700 300N  --
CLC  NC  EHZ IPD1200405250444 4377  -5105 4786ES 2 -31           0  15  27 271 92          326       400   0N  --
CMJ  NC  EHZ IPU0200405250444 3987  -1209 4143ES 2  27           1  -3  -5  24160 1     7.00144100   800   0ND --
CMW1 NC  EP2    4200405250444              4098ES 2   3          42     -11   3177           28            300Y  --
CMW1 NC  HV1 IPU1200405250444 3975  -2105        0                  -6       3177           28       200    A  --
CNI  NC  EHZ IPD0200405250444 4085   3209        0                   0     102119          320       500    N  --
CSU1 NC  HN1 IPD0200405250444 4115  -2209        0                   0     125112          339       500    A  --
CSU1 NC  HN2    4200405250444              4340ES 2  -4          42       0 125112          339            200Y  --
CVL  NC  EHZ EP 2200405250444 4147   22 31       0                  38     106118 2     6.00 26 67   100    ND --
                                                                 21366046
```

station | network | channel | P? | pick time (to min) | P seconds | S seconds | S? | event ID

■ P
■ S
■ Both

NOTE: P and S seconds are *offsets* from the minute-datetime, *not* the actual seconds values. These offsets can be negative or exceed 6000 (60.00 sec). This is because "both" lines share a common minute-datetime, from which negative offsets or offsets exceeding 1 minute are necessary if the P and S picks are far enough apart.

Hypoinverse File Parsing Algorithm

Script name: *parse_phases.py*

1. Add all header line indices to a list (e.g., “chunk” the file by event).
   - 1^{st} event’s lines = lines[event_header_line_ids[0] : event_header_line_ids[1]]
   - 2^{nd} event’s lines = lines[event_header_line_ids[1] : event_header_line_ids[2]], etc.
2. Iterate through each event’s lines and record each line as P, S, or both.
   - For “both” lines, we already have a pair of matching P and S picks.
   - Match P and S picks by iterating through S picks (less numerous than P picks; most efficient) and searching for any matching P picks.
     - First search for picks with matching event ID, station, and instrument. If no matches are found, drop the instrument requirement. Mark these second-pass matches as “non-instrument” matches.
3. Write all pick pairs to a file with name {year}.{month}.{parsed}.txt.
   - Line format: {event ID}|{network}|{station}|{P time}|{S time}|{instrument match?}
   - Henceforth, these files are referred to as **“parsed files.”**

Hypoinverse File Parsing Example

```
CDV   NC   EHZ IPU1200405250444 4237   5105 4543ES 2  -7          42   9  17 188101          80      700 300N  --
CLC   NC   EHZ IPD1200405250444 4377  -5105 4786ES 2 -31           0  15  27 271 92         326      400   0N  --
CMJ   NC   EHZ IPU0200405250444 3987  -1209 4143ES 2  27           1  -3  -5  24160 1    7.00144100  800   0ND --
CMW1  NC   EP2     4200405250444             4098ES 2   3          42     -11   3177          28          300Y  --
CMW1  NC   HV1 IPU1200405250444 3975  -2105         0                  -6       3177          28      200    A  --
CNI   NC   EHZ IPD0200405250444 4085   3209         0                   0      102119         320      500    N  --
CSU1  NC   HN1 IPD0200405250444 4115  -2209         0                   0      125112         339      500    A  --
CSU1  NC   HN2     4200405250444             4340ES 2  -4          42       0 125112         339          200Y  --
CVL   NC   EHZ EP  2200405250444 4147   22 31       0                  38      106118 2    6.00 26 67   100     ND  --
```

- P
- S
- Both

station | network | channel | P? | pick time (to min) | P seconds | S seconds | S?

S picks:
21366046|NC|CMW1|2014-04-05T2004-05-25T04:44:40.98|EP
21366046|NC|CSU1| 2014-04-05T2004-05-25T04:44:43.40|HN

P picks:
21366046|NC|CMW1|2014-04-05T2004-05-25T04:44:39.75|HV
21366046|NC|CNI| 2014-04-05T2004-05-25T04:44:40.85|EH
21366046|NC|CSU1| 2014-04-05T2004-05-25T04:44:41.15|HN
21366046|NC|CVL| 2014-04-05T2004-05-25T04:44:41.47|EH

Pick pairs:
(”both” lines)
21366046|NC|CDV|2004-05-25T04:44:42.37|2014-04-05T2004-05-25T04:44:45.43|1
21366046|NC|CLC|2004-05-25T04:44:43.77|2014-04-05T2004-05-25T04:44:47.86|1
21366046|NC|CMJ|2004-05-25T04:44:39.87|2014-04-05T2004-05-25T04:44:41.43|1

Not instrument match

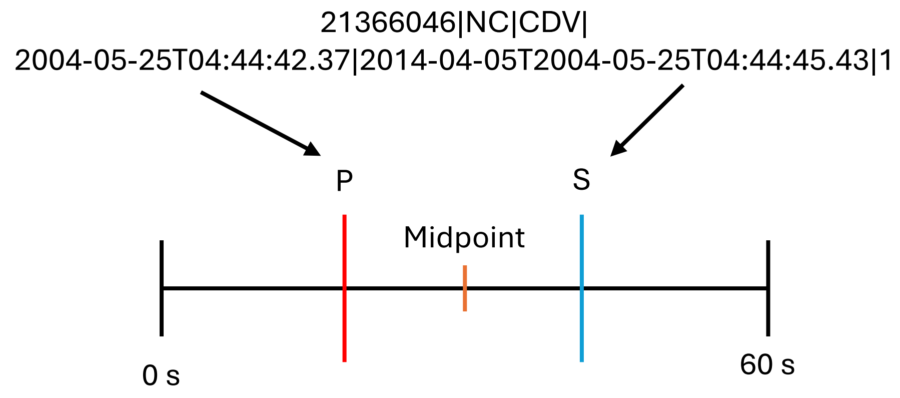
(matched P and S lines)
21366046|NC|CMW1|2004-05-25T04:44:39.75|2014-04-05T2004-05-25T04:44:40.98|0
21366046|NC|CSU1|2004-05-25T04:44:41.15|2014-04-05T2004-05-25T04:44:43.40|1

- There is additional logic in the code to prevent duplicate (event ID, station) pick pairs from being saved. Duplicate sets *should* be identical, because in theory, seismic waves should be detected by all instrument at the same time, with possibly some small discrepancies due to to different sampling rates.

Downloading Waveforms

Script name: *download_waveforms.py*

- 60 s waveforms centered on the midpoint between the P and S times in pick pairs are requested from the NCEDC dataselect web service via Obspy.
  - Randomly positioned 30 s windows must be extracted for PhaseNet training, but downloading this extra data might expand the possible applications of the dataset (GAN training?).
- Data is requested from *all* instruments of interest (see lower-right figure).
  - There is an FDSN “station” web service from which we can obtain a list of available instruments at each station, but this introduces a second HTTP request, greatly slowing the pipeline.
  - [Band/instrument code documentation](#).




```python
bands = ['D', 'E', 'S', 'H', 'B']
instruments = ['H', 'L', 'N', 'P']

for band in bands:
  for instrument in instruments:
    bulk += f'{network} {station} * {band}{instrument}* {dt_start} {dt_end}\n'
```

- There are 20 band/instrument combinations.

Scale of Dataset and Waveform Queries

- In total, we must download 1,367,579 3-component waveforms and 444,271 Z-component "singletons."
- This is too slow to accomplish without batching the HTTP requests to NCEDC.
  - FDSN dataselect documentation (see page 3 regarding POST method).
  - Obspy get_waveforms_bulk documentation (used in this pipeline).
- Waveforms are downloaded via bulk requests using 10 threads (multithreading scheme, see next slide).
  - Each bulk request comprises 100 lines from a parsed file.
    - Increasing the line batch size beyond 100 lines causes occasional timeouts.
- The time required to download all waveforms using properly multithreaded bulk HTTP requests is **~2-3 days.**

```
bands = ['D', 'E', 'S', 'H', 'B']
instruments = ['H', 'L', 'N', 'P']

for band in bands:
  for instrument in instruments:
    bulk += f'{network} {station} * {band}{instrument}* {dt_start} {dt_end}\n'
```

- There are 20 band/instrument combinations.
- Therefore, one batch of 100 parsed phase lines becomes a 2,000 line bulk waveform query.
- Typically, this returns a stream containing 300-500 traces (waveforms).

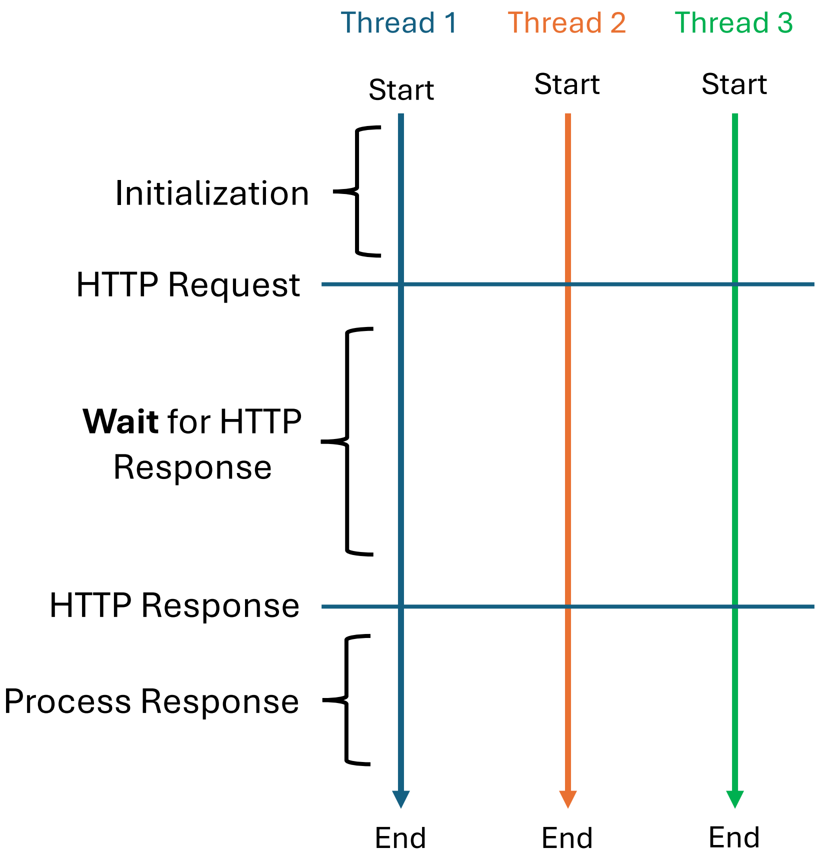
```
NC MCV * DH* 2013-11-02T18:23:13.110000Z 2013-11-02T18:24:13.110000Z
NC MCV * DL* 2013-11-02T18:23:13.110000Z 2013-11-02T18:24:13.110000Z
NC MCV * DN* 2013-11-02T18:23:13.110000Z 2013-11-02T18:24:13.110000Z
NC MCV * DP* 2013-11-02T18:23:13.110000Z 2013-11-02T18:24:13.110000Z
NC MCV * EH* 2013-11-02T18:23:13.110000Z 2013-11-02T18:24:13.110000Z
NC MCV * EL* 2013-11-02T18:23:13.110000Z 2013-11-02T18:24:13.110000Z
NC MCV * EN* 2013-11-02T18:23:13.110000Z 2013-11-02T18:24:13.110000Z
NC MCV * EP* 2013-11-02T18:23:13.110000Z 2013-11-02T18:24:13.110000Z
NC MCV * SH* 2013-11-02T18:23:13.110000Z 2013-11-02T18:24:13.110000Z
NC MCV * SP* 2013-11-02T18:23:13.110000Z 2013-11-02T18:24:13.110000Z
NC MCV * HH* 2013-11-02T18:23:13.110000Z 2013-11-02T18:24:13.110000Z
```

(x 2,000)

Python Multiprocessing vs Multithreading

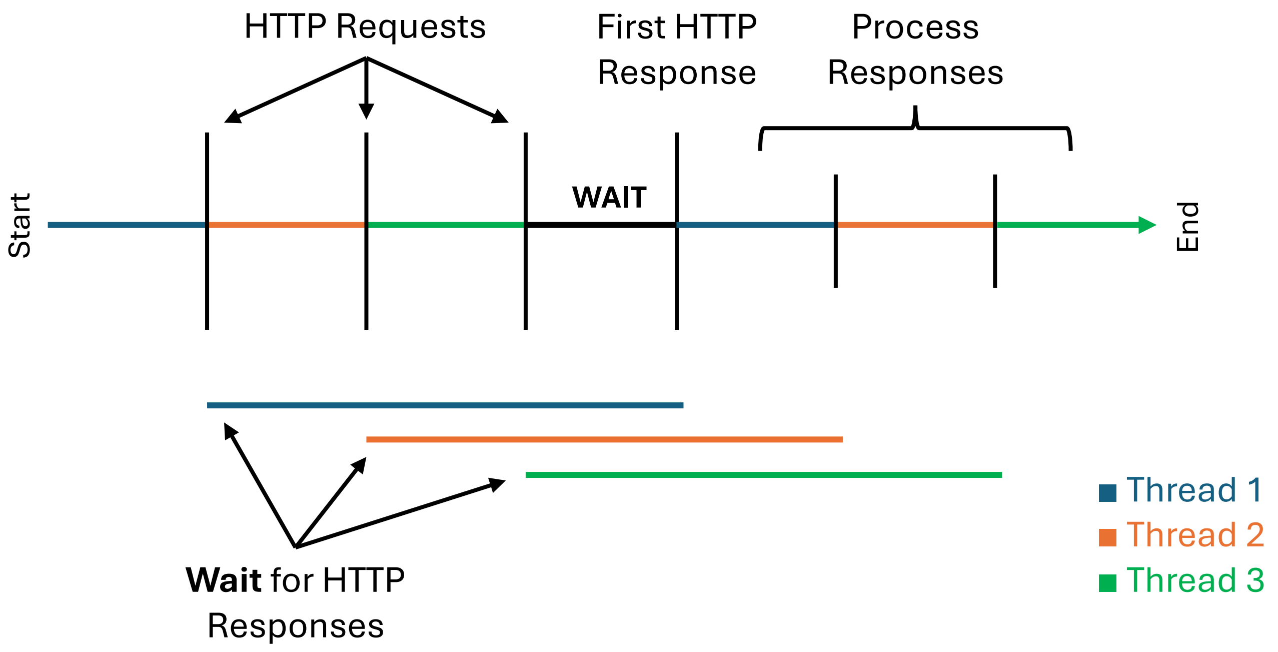
Multiprocessing

- Multiple threads of execution on separate CPU cores.
- Parallel **processing.**



Multithreading

- Multiple threads swapped in and out from single CPU core.
- Parallel **"waiting."**
- Used by NCEDC data pipeline (i/o bound process).



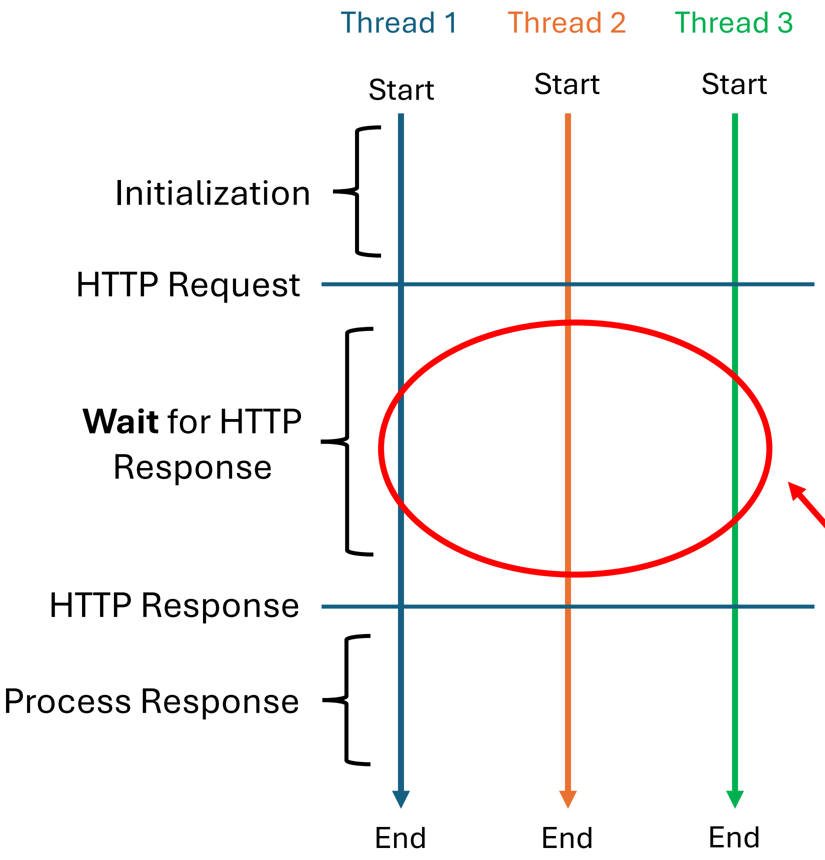
Python Multiprocessing vs Multithreading

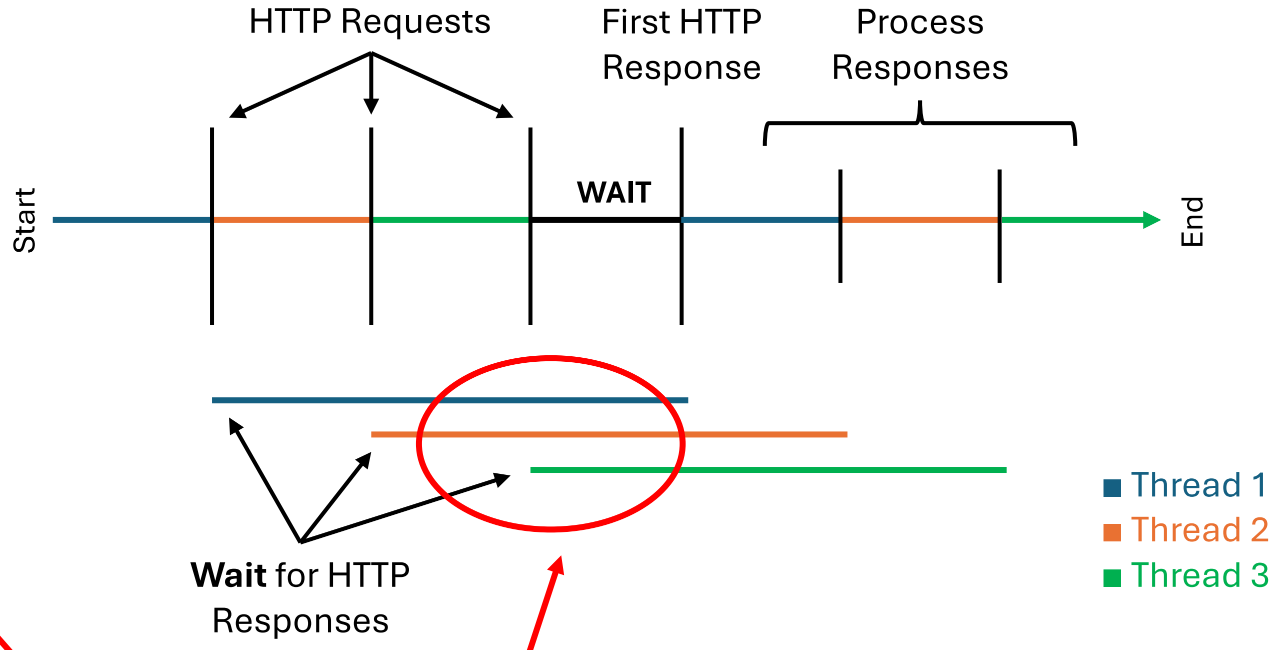
Multiprocessing

- Multiple threads of execution on separate CPU cores.
- Parallel **processing.**

Multithreading

- Multiple threads swapped in and out from single CPU core.
- Parallel **“waiting.”**
- Used by NCEDC data pipeline (i/o bound process).

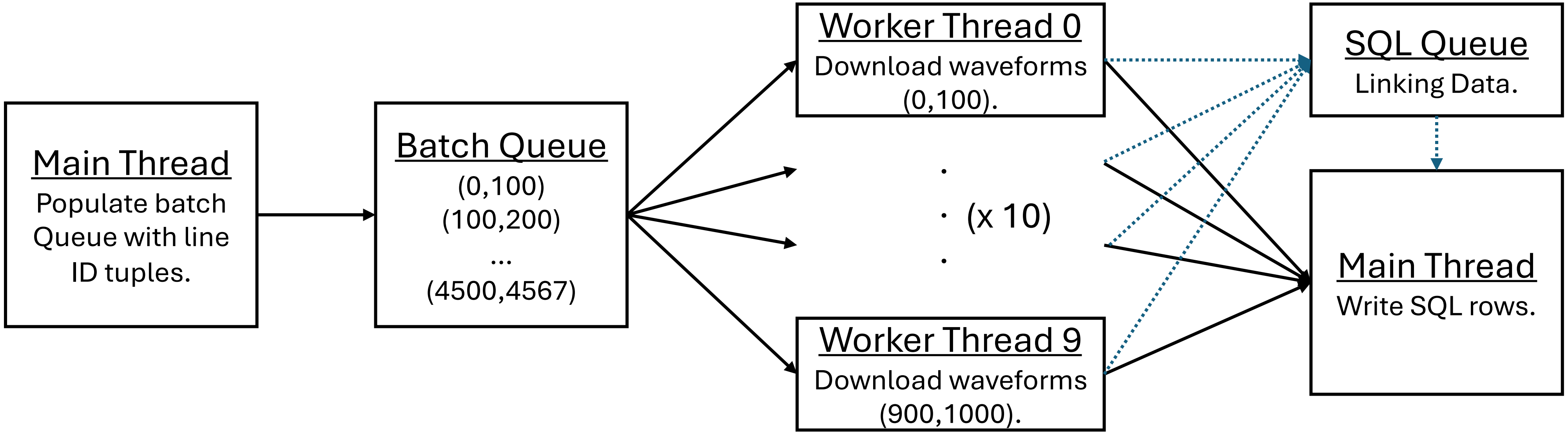




Parallelized waiting for HTTP responses

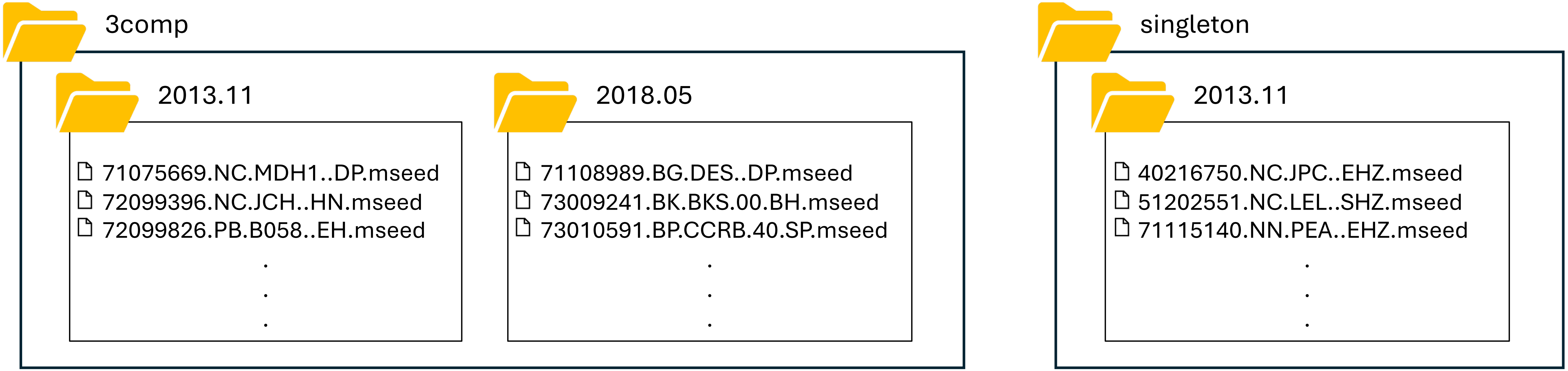
Assigning Work to Threads

- Main thread populates a python Queue with tuples containing the start and end line indicies of a batch until all of a parsed file's lines are exhausted; (0,100), (100,200), etc.
  - Queues are thread-safe containers in Python; data can added or removed safely by multiple threads.
- Worker threads download the waveforms corresponding to 100-line batches by consuming the tuples from the Queue.
  - Worker threads append data to a separate Queue linking the downloaded waveforms (miniseed files) back to the metadata/labels used to construct the queries (see slide 13 for linking details).
    - Main thread consumes this Queue and writes the linking data to SQLite databases using sqlite3.
    - Only the main thread writes SQL rows because SQLite is not suitable for concurrent writes.



Results Thus Far

- After running the *parse_phases.py* and *download_waveforms.py* scripts, the following folders, miniseed files, and database tables will be produced.
  - Two parent directories, *3comp* and *singleton*, containing 3-component waveforms and Z-singletons, respectively.
    - Sub-directories in both folders containing all waveforms for one month.
  - Two SQLite databases, *NCEDC.db* and *NCEDC_s.db*, containing metadata describing the 3-component and Z-singleton waveforms, respectively.
    - Columns: id, event_id, network, station, location, channel, P, S, instrument_match, complete, waveform_path, phase_file.



SQL query: *select * from phase where waveform_path='./waveforms/3comp/2013.11/71075669.NC.MDH1..DP.mseed’*
---> (364540, '71075669', 'NC', 'MDH1', '', 'DP', '2013-11-02T08:10:57.960000', '2013-11-02T08:11:00.190000', 1, 1, './waveforms/3comp/2013.11/71075669.NC.MDH1..DP.mseed', '2013.11.parsed.txt')

Matching NCEDC Traces with Metadata/Labels

- Traces returned from NCEDC in *download_waveforms.py* are labeled only by station, instrument, and start/end times (no P/S pick times, etc.).
- We must link these traces back to the original metadata/labels used to construct the queries. This is the purpose of the SQLite databases discussed on slide 11.
  - First associate 3-component waveforms and then match them with metadata.
- Requesting waveforms in bulk presents some challenges when matching:
  1. No reliable ordering of traces in returned stream.
     - **One safe assumption?** Traces from a single station/instrument are grouped.
  2. Extra Z-singletons without matching E/N channel waveforms.
  3. Incomplete sets (e.g., E/Z, E/N, N/Z).
     - One cause of incomplete sets is the discarding of “very” incomplete waveforms (< 80% expected length). Some returned traces are < 10 samples!
     - “Slightly” incomplete waveforms are retained for manual processing (zero-padding, etc.), but “very” incomplete waveforms cause significant problems (e.g., excessive matching tolerances, discussed later).
- Strategy: maximize retrieved data by saving both complete 3-component waveforms, and Z-singletons.
  - In some studies, models are trained using only vertical channel waveforms.
  - The Z-singletons are unique and not duplicates of Z-channel waveforms in 3-component waveform sets.

Examples of Matching Challenges

Matching station/instrument group
(the “safe” assumption; slide 13).

```
.
.
.
BK.SCZ.50.BHZ    | 2013-11-07T01:55:29.044539Z - 2013-11-07T01:56:29.019539Z | 40.0 Hz, 2400 samples
1 BK.SCZ.00.HHE  | 2013-11-06T14:17:53.498394Z - 2013-11-06T14:18:53.488394Z | 100.0 Hz, 6000 samples
2 BK.SCZ.00.HHE  | 2013-11-07T01:55:29.028393Z - 2013-11-07T01:56:29.018393Z | 100.0 Hz, 6000 samples
BK.SCZ.00.HHE    | 2013-11-06T15:10:58.038393Z - 2013-11-06T15:11:58.028393Z | 100.0 Hz, 6000 samples
BK.SCZ.00.HHE    | 2013-11-06T16:21:53.288393Z - 2013-11-06T16:22:53.278393Z | 100.0 Hz, 6000 samples
BK.SCZ.00.HHE    | 2013-11-06T22:39:54.278393Z - 2013-11-06T22:40:54.268393Z | 100.0 Hz, 6000 samples
BK.SCZ.00.HHE    | 2013-11-07T03:30:50.018394Z - 2013-11-07T03:31:50.008394Z | 100.0 Hz, 6000 samples
BK.SCZ.50.HHE    | 2013-11-06T14:17:53.498394Z - 2013-11-06T14:18:53.488394Z | 100.0 Hz, 6000 samples
BK.SCZ.50.HHE    | 2013-11-07T01:55:29.028393Z - 2013-11-07T01:56:29.018393Z | 100.0 Hz, 6000 samples
1 BK.SCZ.00.HHN  | 2013-11-06T14:17:53.498394Z - 2013-11-06T14:18:53.488394Z | 100.0 Hz, 6000 samples
2 BK.SCZ.00.HHN  | 2013-11-06T15:10:58.038393Z - 2013-11-06T15:11:58.028393Z | 100.0 Hz, 6000 samples
BK.SCZ.00.HHN    | 2013-11-06T16:21:53.288393Z - 2013-11-06T16:22:53.278393Z | 100.0 Hz, 6000 samples
BK.SCZ.00.HHN    | 2013-11-06T22:39:54.278393Z - 2013-11-06T22:40:54.268393Z | 100.0 Hz, 6000 samples
BK.SCZ.00.HHN    | 2013-11-07T01:55:29.028393Z - 2013-11-07T01:56:29.018393Z | 100.0 Hz, 6000 samples
BK.SCZ.00.HHN    | 2013-11-07T03:30:50.018394Z - 2013-11-07T03:31:50.008394Z | 100.0 Hz, 6000 samples
BK.SCZ.50.HHN    | 2013-11-06T14:17:53.498393Z - 2013-11-06T14:18:53.488393Z | 100.0 Hz, 6000 samples
BK.SCZ.50.HHN    | 2013-11-07T01:55:29.028393Z - 2013-11-07T01:56:29.018393Z | 100.0 Hz, 6000 samples
1 BK.SCZ.00.HHZ  | 2013-11-06T14:17:53.498393Z - 2013-11-06T14:18:53.488393Z | 100.0 Hz, 6000 samples
2 BK.SCZ.00.HHZ  | 2013-11-06T15:10:58.038393Z - 2013-11-06T15:11:58.028393Z | 100.0 Hz, 6000 samples
BK.SCZ.00.HHZ    | 2013-11-06T16:21:53.288393Z - 2013-11-06T16:22:53.278393Z | 100.0 Hz, 6000 samples
BK.SCZ.00.HHZ    | 2013-11-06T22:39:54.278393Z - 2013-11-06T22:40:54.268393Z | 100.0 Hz, 6000 samples
BK.SCZ.00.HHZ    | 2013-11-07T03:30:50.018394Z - 2013-11-07T03:31:50.008394Z | 100.0 Hz, 6000 samples
BK.SCZ.00.HHZ    | 2013-11-07T01:55:29.028394Z - 2013-11-07T01:56:29.018394Z | 100.0 Hz, 6000 samples
BK.SCZ.50.HHZ    | 2013-11-06T14:17:53.498394Z - 2013-11-06T14:18:53.488394Z | 100.0 Hz, 6000 samples
BK.SCZ.50.HHZ    | 2013-11-07T01:55:29.028393Z - 2013-11-07T01:56:29.018393Z | 100.0 Hz, 6000 samples
BK.SCZ.50.HHZ    | 2013-11-06T15:10:58.038393Z - 2013-11-06T15:11:58.028393Z | 100.0 Hz, 6000 samples
BK.SCZ.50.HHZ    | 2013-11-06T16:21:53.288393Z - 2013-11-06T16:22:53.278393Z | 100.0 Hz, 6000 samples
BK.SCZ.50.HHZ    | 2013-11-07T03:30:50.018393Z - 2013-11-07T03:31:50.008393Z | 100.0 Hz, 6000 samples
BP.SMNB.40.DP1   | 2013-11-06T13:25:44.796000Z - 2013-11-06T13:26:44.792000Z | 250.0 Hz, 15000 samples
.
.
.
```

```
1   BK.SCZ.00.HHE | 2013-11-06T14:17:53.498394Z - 2013-11-06T14:18:53.488394Z | 100.0 Hz, 6000 samples
    BK.SCZ.00.HHN | 2013-11-06T14:17:53.498394Z - 2013-11-06T14:18:53.488394Z | 100.0 Hz, 6000 samples
    BK.SCZ.00.HHZ | 2013-11-06T14:17:53.498393Z - 2013-11-06T14:18:53.488393Z | 100.0 Hz, 6000 samples

2   BK.SCZ.00.HHE | 2013-11-07T01:55:29.028393Z - 2013-11-07T01:56:29.018393Z | 100.0 Hz, 6000 samples
    BK.SCZ.00.HHN | 2013-11-06T15:10:58.038393Z - 2013-11-06T15:11:58.028393Z | 100.0 Hz, 6000 samples
    BK.SCZ.00.HHZ | 2013-11-06T15:10:58.038393Z - 2013-11-06T15:11:58.028393Z | 100.0 Hz, 6000 samples
```

Mismatched trace start/end times

```
.
.
.
NC.GDXB..HNN   | 2013-11-06T06:22:43.075000Z - 2013-11-06T06:23:43.065000Z | 100.0 Hz, 6000 samples
NC.GDXB..HNN   | 2013-11-06T08:39:07.995000Z - 2013-11-06T08:40:07.985000Z | 100.0 Hz, 6000 samples
NC.GDXB..HNZ   | 2013-11-06T02:06:18.165000Z - 2013-11-06T02:07:18.155000Z | 100.0 Hz, 6000 samples
NC.GDXB..HNZ   | 2013-11-06T06:22:43.075000Z - 2013-11-06T06:23:43.065000Z | 100.0 Hz, 6000 samples
NC.GDXB..HNZ   | 2013-11-06T08:39:07.995000Z - 2013-11-06T08:40:07.985000Z | 100.0 Hz, 6000 samples
NC.HBT..EHZ    | 2013-11-06T07:24:29.550000Z - 2013-11-06T07:25:29.540000Z | 100.0 Hz, 6000 samples
NC.HBT..ELN    | 2013-11-06T07:24:29.550000Z - 2013-11-06T07:25:29.540000Z | 100.0 Hz, 6000 samples
BG.HVC..DPE    | 2013-11-06T08:39:08.446000Z - 2013-11-06T08:40:08.444000Z | 500.0 Hz, 30000 samples
BG.HVC..DPN    | 2013-11-06T08:39:08.446000Z - 2013-11-06T08:40:08.444000Z | 500.0 Hz, 30000 samples
BG.HVC..DPZ    | 2013-11-06T08:39:08.446000Z - 2013-11-06T08:40:08.444000Z | 500.0 Hz, 30000 samples
NC.JSGB..EHE   | 2013-11-06T00:14:19.827700Z - 2013-11-06T00:15:19.817700Z | 100.0 Hz, 6000 samples
NC.JSGB..EHN   | 2013-11-06T00:14:19.827700Z - 2013-11-06T00:15:19.817700Z | 100.0 Hz, 6000 samples
NC.JSGB..EHZ   | 2013-11-06T00:14:19.827700Z - 2013-11-06T00:15:19.817700Z | 100.0 Hz, 6000 samples
NC.JSGB..HNE   | 2013-11-06T00:14:19.827700Z - 2013-11-06T00:15:19.817700Z | 100.0 Hz, 6000 samples
.
.
.
NC.MDP1..HN3   | 2013-11-06T15:34:13.335000Z - 2013-11-06T15:35:13.330000Z | 200.0 Hz, 12000 samples
NC.MDPB..HHE   | 2013-11-06T15:34:13.380000Z - 2013-11-06T15:35:13.370000Z | 100.0 Hz, 6000 samples
NC.MDPB..HHE   | 2013-11-07T01:22:25.940000Z - 2013-11-07T01:23:25.930000Z | 100.0 Hz, 6000 samples
NC.MDPB..HHE   | 2013-11-06T23:57:30.000000Z - 2013-11-06T23:58:29.310000Z | 100.0 Hz, 5932 samples
NC.MDPB..HHN   | 2013-11-06T15:34:13.380000Z - 2013-11-06T15:35:13.370000Z | 100.0 Hz, 6000 samples
NC.MDPB..HHN   | 2013-11-07T01:22:25.940000Z - 2013-11-07T01:23:25.930000Z | 100.0 Hz, 6000 samples
NC.MDPB..HHN   | 2013-11-06T23:57:30.000000Z - 2013-11-06T23:58:29.310000Z | 100.0 Hz, 5932 samples
NC.MDPB..HHZ   | 2013-11-06T15:34:13.380000Z - 2013-11-06T15:35:13.370000Z | 100.0 Hz, 6000 samples
NC.MDPB..HHZ   | 2013-11-07T01:22:25.940000Z - 2013-11-07T01:23:25.930000Z | 100.0 Hz, 6000 samples
NC.MDPB..HHZ   | 2013-11-06T23:57:30.000000Z - 2013-11-06T23:58:29.310000Z | 100.0 Hz, 5932 samples
NC.MDPB..HNE   | 2013-11-06T15:34:13.380000Z - 2013-11-06T15:35:13.370000Z | 100.0 Hz, 6000 samples
NC.MDPB..HNE   | 2013-11-07T01:22:25.940000Z - 2013-11-07T01:23:25.930000Z | 100.0 Hz, 6000 samples
NC.MDPB..HNN   | 2013-11-06T15:34:13.380000Z - 2013-11-06T15:35:13.370000Z | 100.0 Hz, 6000 samples
NC.MDPB..HNN   | 2013-11-07T01:22:25.940000Z - 2013-11-07T01:23:25.930000Z | 100.0 Hz, 6000 samples
NC.MDPB..HNZ   | 2013-11-06T15:34:13.380000Z - 2013-11-06T15:35:13.370000Z | 100.0 Hz, 6000 samples
NC.MDPB..HNZ   | 2013-11-07T01:22:25.940000Z - 2013-11-07T01:23:25.930000Z | 100.0 Hz, 6000 samples
NC.MDR.02.EHZ  | 2013-11-06T15:59:02.760000Z - 2013-11-06T16:00:02.750000Z | 100.0 Hz, 6000 samples
.
.
.
```

Challenges (#)

- (1) Out of order
- (2) Extra Z ”singletons”
- (3) Missing Component
- Incomplete Trace

Matching Algorithm (1) – Find 3 Component Waveforms

```
for i=0:n_traces_single_station_instrument: (grouping arising from the “safe” assumption; slide 13)
    if group[i] == ‘E’ or ‘1’ trace:
        trace_E = group[i]
        idx_NZ = [];
        expected_trace_length = 60 * trace_E.sampling_rate;
        length_error_E = expected_trace_length – len(trace_E);
        for j=0:n_selected_lines:
            trace_j = group[j]
            length_error_j = expected_trace_length – len(trace_j);
            length_error_max = max(length_error_E, length_error_j);
            if (j != k) && (trace_E.location == group[j].location)
               && (abs(trace_E.starttime – trace_j.starttime) < length_error_max/trace_j.sampling_rate + 0.1):
               idx_nz.append(j);
        if len(idx_NZ) == 2:
            3_component_waveform = [trace_E, group[idx_NZ[0]], group[idx_NZ[1]]]
        else:
            log_error();
```

Tolerance for robustness to incomplete traces (see slide 17).

Arbitrary margin for error.

```
Incomplete set, or set with too many components:
2 Trace(s) in Stream:
NC.MMX1..EP1 | 2013-08-02T04:45:56.390000Z - 2013-08-02T04:46:56.385000Z | 200.0 Hz, 12000 samples
NC.MMX1..EP2 | 2013-08-02T04:45:56.390000Z - 2013-08-02T04:46:56.385000Z | 200.0 Hz, 12000 samples
```

Matching Algorithm (1) Example

Each shaded area corresponds to a complete execution of the outer i-loop on the previous slide.

i j

Find all the E's and 1's

Find matching N/2's and Z/3's

BK.HAST.00.BHE	2013-08-02T01:36:42.494538Z - 2013-08-02T01:37:42.469538Z	40.0 Hz, 2400 samples
BK.HAST.00.BHE	2013-08-02T09:49:24.719539Z - 2013-08-02T09:50:24.694539Z	40.0 Hz, 2400 samples
BK.HAST.00.BHN	2013-08-02T01:36:42.494538Z - 2013-08-02T01:37:42.469538Z	40.0 Hz, 2400 samples
BK.HAST.00.BHN	2013-08-02T09:49:24.719538Z - 2013-08-02T09:50:24.694538Z	40.0 Hz, 2400 samples
BK.HAST.00.BHZ	2013-08-02T01:36:42.494538Z - 2013-08-02T01:37:42.469538Z	40.0 Hz, 2400 samples
BK.HAST.00.BHZ	2013-08-02T09:49:24.719538Z - 2013-08-02T09:50:24.694538Z	40.0 Hz, 2400 samples
BK.HAST.00.HHE	2013-08-02T01:36:42.488393Z - 2013-08-02T01:37:39.068393Z	100.0 Hz, 5659 samples
BK.HAST.00.HHE	2013-08-02T09:49:24.708393Z - 2013-08-02T09:50:24.698393Z	100.0 Hz, 6000 samples
~~BK.HAST.00.HHE~~	~~2013-08-02T01:37:39.078394Z - 2013-08-02T01:37:42.478394Z~~	~~100.0 Hz, 341 samples~~
BK.HAST.00.HHN	2013-08-02T01:36:42.488393Z - 2013-08-02T01:37:42.478393Z	100.0 Hz, 6000 samples
BK.HAST.00.HHN	2013-08-02T09:49:24.708393Z - 2013-08-02T09:50:24.698393Z	100.0 Hz, 6000 samples
BK.HAST.00.HHZ	2013-08-02T01:36:42.488393Z - 2013-08-02T01:37:42.478393Z	100.0 Hz, 6000 samples
BK.HAST.00.HHZ	2013-08-02T09:49:24.708393Z - 2013-08-02T09:50:24.698393Z	100.0 Hz, 6000 samples
BK.HAST.00.HNE	2013-08-02T01:36:42.488393Z - 2013-08-02T01:37:42.478393Z	100.0 Hz, 6000 samples
BK.HAST.00.HNE	2013-08-02T09:49:24.708393Z - 2013-08-02T09:50:24.698393Z	100.0 Hz, 6000 samples
BK.HAST.00.HNN	2013-08-02T01:36:42.488393Z - 2013-08-02T01:37:42.478393Z	100.0 Hz, 6000 samples
BK.HAST.00.HNN	2013-08-02T09:49:24.708393Z - 2013-08-02T09:50:24.698393Z	100.0 Hz, 6000 samples
BK.HAST.00.HNZ	2013-08-02T01:36:42.488393Z - 2013-08-02T01:37:42.478393Z	100.0 Hz, 6000 samples
NC.CMW..EHN	2013-08-01T20:03:37.700300Z - 2013-08-01T20:04:37.695300Z	200.0 Hz, 12000 samples
NC.CMW..EHZ	2013-08-01T20:03:37.700300Z - 2013-08-01T20:04:37.695300Z	200.0 Hz, 12000 samples
NC.CSU1..EP2	2013-08-01T20:03:37.435300Z - 2013-08-01T20:04:37.430300Z	200.0 Hz, 12000 samples
NC.CSU1..EP3	2013-08-01T20:03:37.435300Z - 2013-08-01T20:04:37.430300Z	200.0 Hz, 12000 samples

3-component sets

"slightly" incomplete waveform

"very" incomplete waveform

3-component sets

3-component set

Incomplete E/N set (unconsumed/discarded)

Incomplete N/Z set (N unconsumed/discarded, Z retained as Z-singleton)

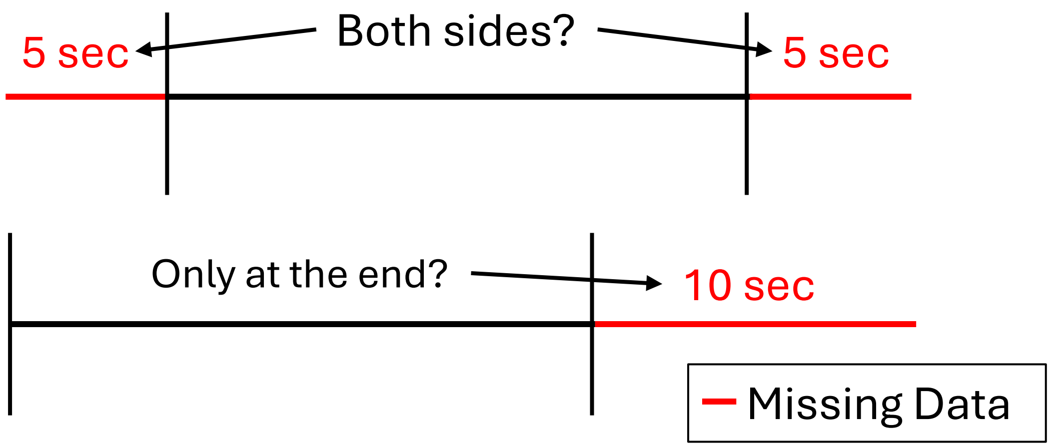
Incomplete 2/3 set (2 unconsumed/discarded, 3 retained as Z-singleton)

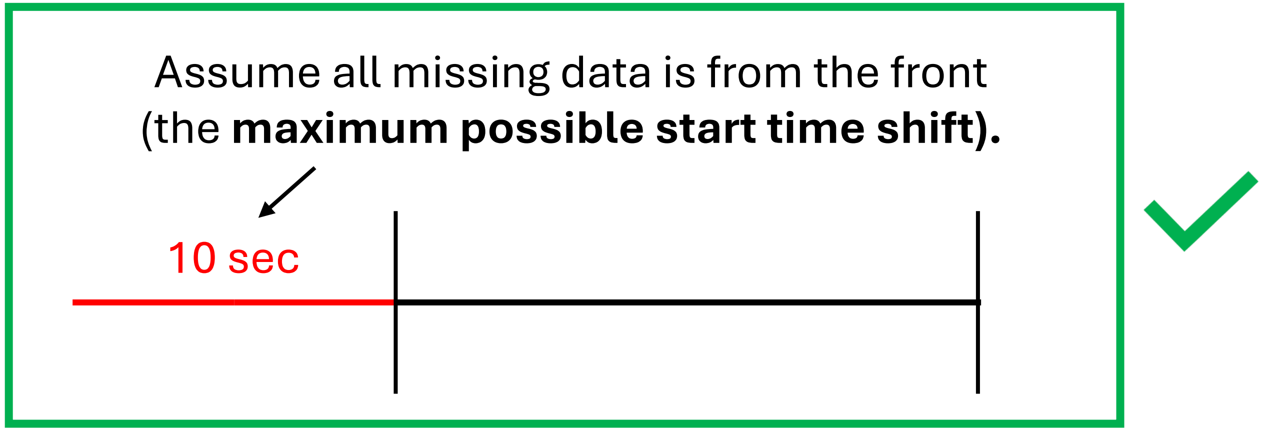
Matching station/instrument groups are color coded: BK/HAST/BH, BK/HAST/HH, BK/HAST/HN, NC/CMW/EH, NC/CSU1/EP.

Matching Algorithm (2) – Match Waveforms to Metadata/Labels

- Waveforms are matched with metadata based on a single component/trace.
  - In the case of 3-component waveforms, this is the longest (most-complete) component/trace. Using the longest trace minimizes the necessary tolerance for missing data (see below).
  - In the case of Z-singletons, this is simply the Z component/trace.
- Traces retrieved from NCEDC are labeled only by network, station, and start/end time.
  - A single query can comprise multiple parsed lines from the same network/station (i.e., different events), so **start time is the only unique data on which matches can be made.**
  - However, incomplete traces can have a shifted start time due to missing data, necessitating a tolerance.
  - Before allowing a tolerance, we search for exact matches (safer) in case all missing data is from the end.

Where is the missing data?





- **This tolerance becomes overly large if "very" incomplete waveforms are retained.**

Matching Algorithm (2) Example

Traces from NCEDC:

```
AZ.KNW..HHE | 2013-08-11T05:19:46.398400Z - 2013-08-11T05:20:44.158400Z | 100.0 Hz, 5777 samples
AZ.KNW..HHN | 2013-08-11T05:19:45.388400Z - 2013-08-11T05:20:44.158400Z | 100.0 Hz, 5878 samples
AZ.KNW..HHZ | 2013-08-11T05:19:45.298400Z - 2013-08-11T05:20:44.158400Z | 100.0 Hz, 5887 samples
```

Longest trace.

Parsed pick lines:

```
72046971|AZ|KNW|2013-08-11T05:20:12.160000|2013-08-11T05:20:16.170000|True
[Imagine a second AZ.KNW line which prevents us from simply matching on station]
72046971|AZ|LVA2|2013-08-11T05:20:09.080000|2013-08-11T05:20:11.120000|True
72046971|AZ|PFO|2013-08-11T05:20:10.480000|2013-08-11T05:20:13.210000|True
72046971|AZ|RDM|2013-08-11T05:20:12.300000|2013-08-11T05:20:16.370000|True
```

Requested time span:
2013-08-11T05:19:44.165000Z - 2013-08-11T05:20:44.165000Z

Mismatch = 2013-08-11T05:19:45.298400Z - 2013-08-11T05:19:44.165000Z = **1.1334 sec.**

Tolerance = (6000 – 5887)/60 Hz + 0.1 = **1.98 sec.**

Mismatch < tolerance, so we find the match ✓.

NOTES:

- There is a danger we could match multiple pick lines due to the tolerance. So far this hasn't happened, and the code checks for this scenario just in case.
- It is not uncommon for queries to return no data for some of the parsed pick lines (~1/50 lines). If none of the traces in an NCEDC response can be associated with a pick line, that pick line is logged in a "retry" file for manual examination.

Log Files (1) – Progress of Bulk Downloads

- Separate log file created for each thread and {year}.{month}.parsed.txt file.
  - Naming format: thread#_{year}.{month}.parsed.txt_{utc_now}.txt.
  - Prevents multiple threads from concurrently writing to a single text file.
- The following information is logged:
  - {year}.{month}.parsed.txt file line numbers corresponding to current bulk query ①.
  - Full contents of stream returned from NCEDC ②.
  - List of “very” incomplete waveforms that were discarded ③.
  - Results of matching algorithm (E/N/Z sets, Z-singletons, unconsumed traces) ④.
    - As a reminder, unconsumed traces should only be E/N/1/2 as Z/3 channel traces should be retained as Z-singletons.

```
① Processing lines. File: 1996.03.parsed.txt, line ids (n=100): 300-400.

  Downloaded stream.
  221 Trace(s) in Stream:
  NC.AOD..EHZ   | 1996-03-11T22:25:35.830000Z - 1996-03-11T22:26:35.820000Z | 100.0 Hz, 6000 samples
② NC.AOH..EHZ   | 1996-03-10T04:15:18.200000Z - 1996-03-10T04:16:18.190000Z | 100.0 Hz, 6000 samples
  PG.AR..EHE    | 1996-03-12T02:36:38.278000Z - 1996-03-12T02:37:16.448000Z | 100.0 Hz, 3818 samples
  PG.AR..EHN    | 1996-03-12T02:36:38.278000Z - 1996-03-12T02:37:16.448000Z | 100.0 Hz, 3818 samples
  PG.AR..EHZ    | 1996-03-12T02:36:38.278000Z - 1996-03-12T02:37:16.448000Z | 100.0 Hz, 3818 samples
  .
  .
  .

  The following traces were removed because they were < 80% expected length:
  33 Trace(s) in Stream:
③ PG.AR..EHE  | 1996-03-12T02:36:38.278000Z - 1996-03-12T02:37:16.448000Z | 100.0 Hz, 3818 samples
  PG.AR..EHN  | 1996-03-12T02:36:38.278000Z - 1996-03-12T02:37:16.448000Z | 100.0 Hz, 3818 samples
  PG.AR..EHZ  | 1996-03-12T02:36:38.278000Z - 1996-03-12T02:37:16.448000Z | 100.0 Hz, 3818 samples
  .
  .
  .

  Attempting to write all-Z stream:
  (Note that some traces may not be written if they cannot be matched with phase data.)
④ 1 Trace(s) in Stream:
  NC.BSG..EHZ | 1996-03-11T06:03:54.400000Z - 1996-03-11T06:04:54.390000Z | 100.0 Hz, 6000 samples

  Writing E/N/Z set:
  3 Trace(s) in Stream:
  NC.BSG..ELE | 1996-03-11T06:03:54.400000Z - 1996-03-11T06:04:54.390000Z | 100.0 Hz, 6000 samples
④ NC.BSG..ELN | 1996-03-11T06:03:54.400000Z - 1996-03-11T06:04:54.390000Z | 100.0 Hz, 6000 samples
  NC.BSG..ELZ | 1996-03-11T06:03:54.400000Z - 1996-03-11T06:04:54.390000Z | 100.0 Hz, 6000 samples

  Incomplete set, or set with too many components:
  2 Trace(s) in Stream:
④ NC.KHM..ELE | 1996-03-11T15:20:02.720000Z - 1996-03-11T15:21:02.710000Z | 100.0 Hz, 6000 samples
  NC.KHM..ELN | 1996-03-11T15:20:02.720000Z - 1996-03-11T15:21:02.710000Z | 100.0 Hz, 6000 samples
  .
  .
  .

  Unconsumed traces:
  65 Trace(s) in Stream:
④ NC.CCH1..EP2 | 1996-03-11T21:59:02.620000Z - 1996-03-11T22:00:02.615000Z | 200.0 Hz, 12000 samples
  NC.JCH..ELE  | 1996-03-11T04:15:48.340000Z - 1996-03-11T04:16:48.330000Z | 100.0 Hz, 6000 samples
  NC.JCH..ELN  | 1996-03-11T04:15:48.340000Z - 1996-03-11T04:16:48.330000Z | 100.0 Hz, 6000 samples
  .
  .
  .
```

Log Files (2) – Retry Files

- {year}.{month}.parsed.txt file lines to which no traces returned from NCEDC can be associated (see slide 18) are logged to “retry” files.
    - Naming format: {year}.{month}.parsed.txt.{utc_now}.txt.
    - Log file contents are simply the {year}.{month}.parsed.txt file lines (i.e., {event ID}|{network}|{station}|{P time}|{S time}|{instrument match?}; see slide 5).
    - These lines are logged for the purpose of manual processing later if necessary (e.g., attempting to redownload the data with shifted endpoints).

```
1222696|NC|MDC|1996-03-01T04:07:13.010000|1996-03-01T04:07:15.480000|1
1222696|NC|MMP|1996-03-01T04:07:12.730000|1996-03-01T04:07:15.070000|1
1222696|NC|MMT|1996-03-01T04:07:14.850000|1996-03-01T04:07:20.290000|1
30098071|NC|MCD|1996-03-01T04:08:33.430000|1996-03-01T04:08:41.670000|1
30098071|NC|MTU|1996-03-01T04:08:32.770000|1996-03-01T04:08:39.430000|1
30098140|NC|MLR|1996-03-01T17:30:52.310000|1996-03-01T17:31:02.480000|1
30098147|NC|MLR|1996-03-01T18:58:09.280000|1996-03-01T18:58:19.350000|1
30098150|NC|MRF|1996-03-01T19:02:15.610000|1996-03-01T19:02:26.820000|1
30098153|NC|MLR|1996-03-01T20:40:33.010000|1996-03-01T20:40:43.170000|1
30098153|NC|MRF|1996-03-01T20:40:34.720000|1996-03-01T20:40:45.870000|1
30098071|BK|CMB|1996-03-01T04:08:45.100000|1996-03-01T04:08:58.540000|1
1222697|NC|MDP|1996-03-01T04:30:29.590000|1996-03-01T04:30:31.870000|1
1222697|NC|MMS|1996-03-01T04:30:29.560000|1996-03-01T04:30:32.660000|1
1222697|NC|MMT|1996-03-01T04:30:33.280000|1996-03-01T04:30:36.630000|1
                                .
                                .
                                .
```

Rollback Steps

1. Delete all rows in the SQLite databases which originated from the phase file currently being processed .
2. Delete the corresponding waveform folders from the 3comp and singleton directories.

Example exception encountered when processing 1994.08.parsed.txt:

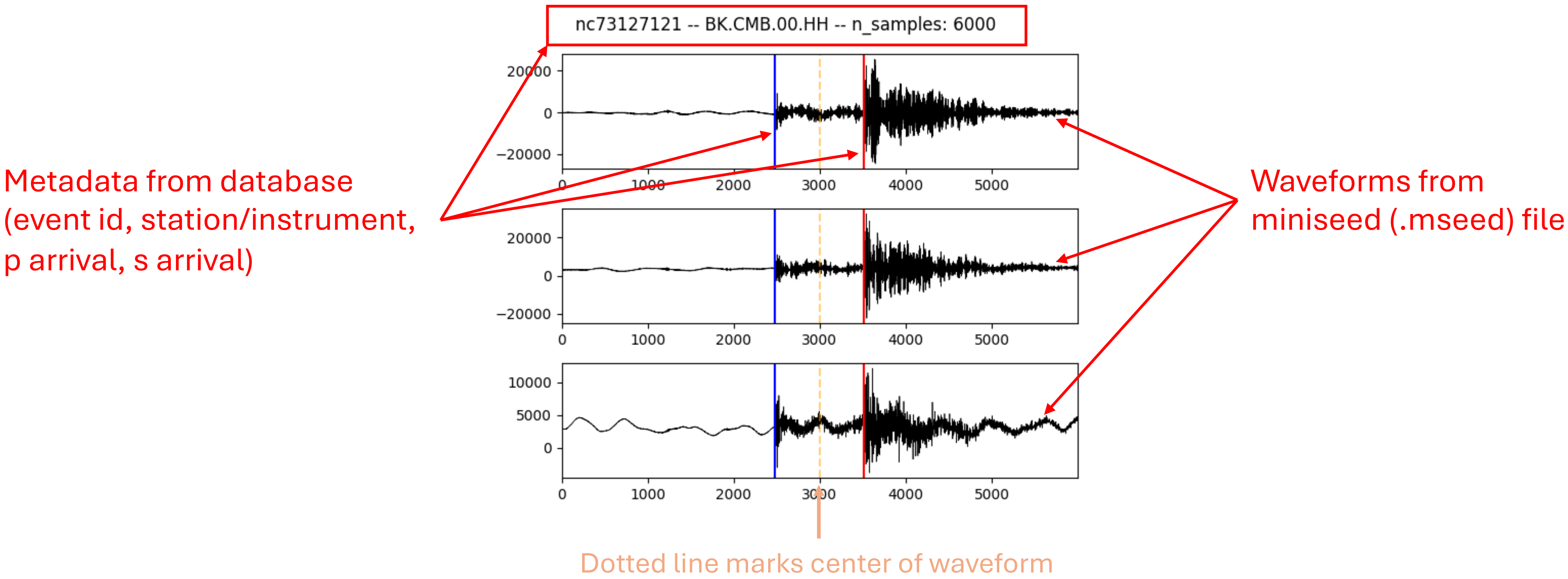
```
Exception in thread Thread-7:
Traceback (most recent call last):
  File "/Applications/Xcode.app/Contents/Developer/Library/Frameworks/Python3.framework/Versions/3.9/lib/python3.9/threading.py", line 973, in _bootstrap_inner
    self.run()
  File "/Users/jeffchurch/Documents/NCEDC/download_waveforms.py", line 29, in run
    process_lines(file, self.log_path, batch_start, batch_end)
  File "/Users/jeffchurch/Documents/NCEDC/download_waveforms.py", line 207, in process_lines
    stream = client.get_waveforms_bulk(bulk)
  File "/Users/jeffchurch/Library/Python/3.9/lib/python/site-packages/obspy/clients/fdsn/client.py", line 1051, in get_waveforms_bulk
    data_stream = self._download(
  File "/Users/jeffchurch/Library/Python/3.9/lib/python/site-packages/obspy/clients/fdsn/client.py", line 1486, in _download
    raise_on_error(code, data)
  File "/Users/jeffchurch/Library/Python/3.9/lib/python/site-packages/obspy/clients/fdsn/client.py", line 1856, in raise_on_error
    raise FDSNException("Unknown Error (%s): %s" % (
obspy.clients.fdsn.header.FDSNException: Unknown Error (URLError): <urlopen error [Errno 8] nodename nor servname provided, or not known>
```

Example steps:

1. SQL query: *delete from phase where phase_file = "1994.08.parsed.txt"*
    - Run against both databases.
2. Delete ./waveforms/3comp/1994.08 and ./waveforms/singleton/1994.08 folders.
3. Rerun *download_waveforms.py* for 1994.08.parsed.txt.

Waveform Checker Script

- The *waveform_checker.py* script can be run to verify the contents of the SQLite databases and downloaded waveforms (.mseed files).
- The script reads a random line from the database and plots the corresponding waveforms and metadata.
  - Continues running forever until process is force terminated via ctrl+c.



HDF5 Dataset

- HDF5 is a hierarchical data format, where all data and metadata are conveniently stored together in a single file.
  - A file (.hdf5 or .h5)
    - Groups (with attributes). *Groups are optional.*
      - Datasets (i.e., arrays; with attributes)
- Three HDF5 files are created to find the most efficient implementation.
  - In implementations 1&2 below, all data is stored only once (in the "x_all" and "y_all" groups).
  - Train/val./test splits (i.e., train_x, train_y, validation_x, validation_y, test_x, and test_y groups) link to data stored in x_all and y_all.
    - Links are cheap; HDF5 file size grows only by ~100 mb (e.g., 178.45 gb →178.52 gb) after all splits are added.
  - HDF5 file structure has significant overhead. Mseeds (54.76 gb) → Imp. 1 (178.52 gb). **Can this be improved?**

<u>Implementation 1</u> (178.52 gb)

- Groups containing separate datasets (with attributes) for each 3-comp waveform:
  - x_all: all downloaded waveforms.
  - y_all: corresponding labels.
  - Train/validation/test splits (8 groups total).
- Dataset names are string ids (e.g., "0", "1") to facilitate PyTorch HDF5 Dataset.
- Dataset attributes: event_id, database_id, network, station, location, channel, start_time, end_time, p_time, s_time, p_sample, s_sample.
- No gzip compression.

<u>Implementation 2</u> (72.48 gb)

- Groups containing separate datasets (with attributes) for each 3-comp waveform:
  - Same groups as implementation 1.
- Same dataset names as implementation 1.
- Same attributes as implementation 1.
- gzip compression.

(implementation 1, but with compression)

<u>Implementation 3</u> (56.88 gb)

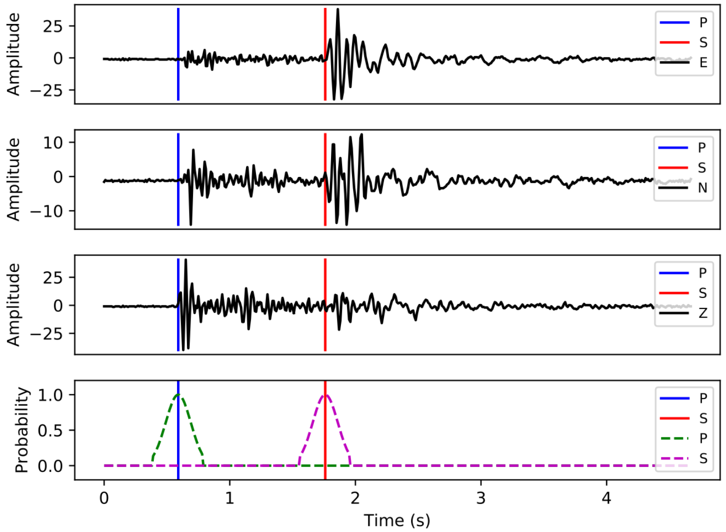
- No groups, but rather standalone large datasets (i.e., arrays) corresponding to groups from implementations 1&2:
  - Does NOT contain all waveforms, but train/validation/test splits only (copied from imp. 1; 6 datasets total).
- Dataset names are group names from imps. 1&2 (e.g., "train_x", "train_y").
- Individual waveforms and labels are indexed by array slicing.
- No attributes.
- No gzip compression.

(the leanest implementation possible)

Dataset Labels

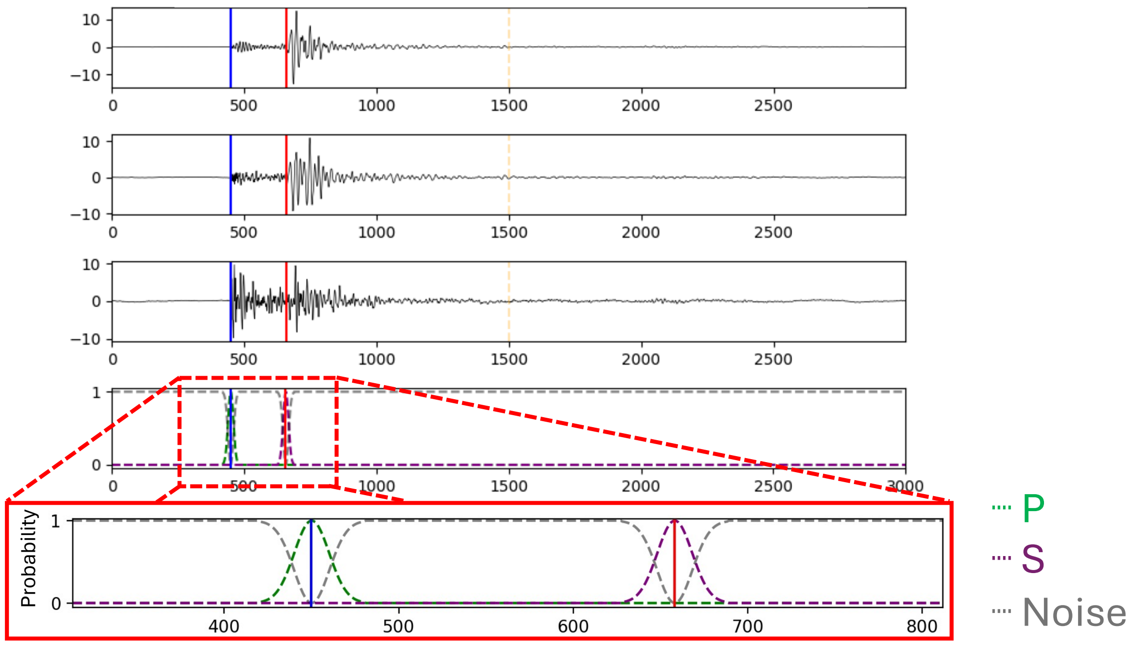
- In the PhaseNet study, P/S arrival times are converted to P/S probability distributions by centering truncated Gaussians with σ = 0.1 s (or 10 samples at 100 Hz) on the arrivals.
  - This is meant to address errors and biases inherent in hand-labeled waveforms, and to accelerate convergence of the model.
- Altogether, PhaseNet labels comprise three time series representing the P, S, and noise probabilities for each sample in the waveform.
  - P(p) and P(s) are given by the truncated Guassians, and are zero elsewhere. P(noise) = 1-(P(p)+P(s)).
  - The P and S probability time series are normalized between 0 and 1.

PhaseNet Article Example



(Zhu and Beroza, 2019)

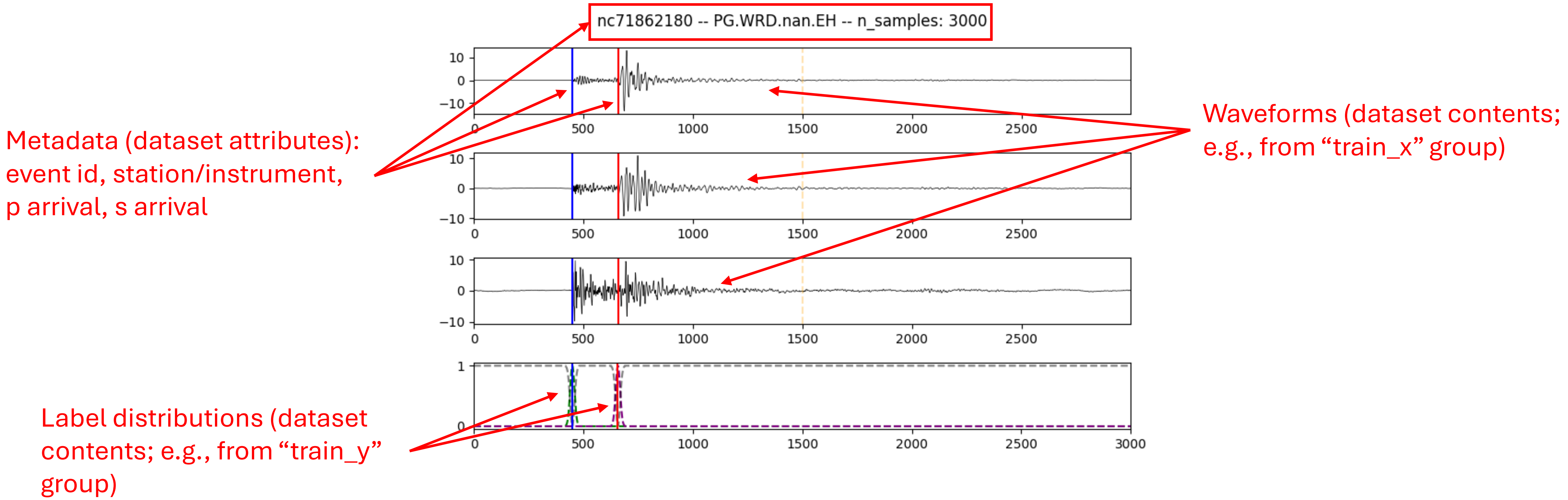
Labeled Data from NCEDC Pipeline



Output from *hdf5_checker.py* script (see next slide).

Convenience of HDF5

- The *waveform_checker.py* script requires many files to work.
    - SQLite database (metadata).
    - Different miniseed file for each 3-component waveform.
- The *hdf5_checker.py* script accomplishes the same thing, reading only a single hdf5 file.
    - Note that waveforms are randomly shifted and trimmed to 30 s to match PhaseNet specifications.



Details of HDF5 Dataset Creation (1)

Script name: *build_hdf5.py*

- <u>Step 1: filtering.</u>
  - Remove incomplete waveforms (those with missing samples).
  - Keep only waveforms where 0.35 s ≤ P/S distance ≤ 26 s.
    - If P/S distance is < 0.35 s, then (normalized) P(p) + P(s) > 1.0, meaning P(noise) < 0.0 (see slide 24).
    - If P/S distance is > 26 s then both phases can no longer comfortably fit within a 30 s waveform.
  - Remove waveforms containing a component with standard deviation < 0.1 (i.e., a channel with no data).
    - Std. dev. is used instead of mean because there are "flat" waveforms with non-zero mean.
  - All filtering is performed using pandas.
    - To avoid repeating lengthy computations, filtered data is saved to disk (*df_complete.csv*).
- <u>Step 2: processing and saving filtered waveforms to "x_all," and "y_all" HDF5 groups.</u>
  - Waveforms are resampled to 100 Hz using the Obspy resample function.
  - Randomly select a 30 s window from the downloaded 60 s waveform that contains both phases.
    - This is done by shifting the center of the 60 s waveform up to a maximum distance given by:

$$\underbrace{\boxed{P + (P/S\ \text{offset})/2}}_{\text{60 s waveform mid point}} + \underbrace{\boxed{15 - S}}_{\text{distance from S to 30 s endpoint (pre-shift)}} - \underbrace{\boxed{2}}_{\text{small extra margin (e.g., to account for P/S distribution width)}}$$

  - Normalize waveforms by subtracting the mean and dividing by the standard deviation.
  - Create dataset attributes: *event id, database id, network, station, location, channel, waveform start/end time, p/s time, p/s sample*.

Details of HDF5 Dataset Creation (2)

- Step 3: create stratified train/validation/test splits.
  - Train/validation/test splits are saved in the “train_x,” “train_y,” “validation_x,” “validation_y,” “test_x,” and “test_y” HDF5 groups.
    - Waveform data is not copied; rather, links to the “x_all” and “y_all” groups are created, only marginally increasing HDF5 file size (e.g., 178.45 gb →178.52 gb; see slide 23).
  - Split sizes (632,000, 79,000, and 79,000, respectively) are in line with the PhaseNet article dataset (623,054, 77,866 and 78,592, respectively).
  - Splits are created by calling the scikit-learn train_test_split function on the filtered pandas dataframe from step 1.
    - Splits are stratified on station, as specified by the PhaseNet article. To facilitate stratifying on station, only waveforms whose station recorded ≥ 10 waveforms are retained.
    - Stratified dataframes are saved to avoid repeated computations (*df_train.csv, df_val.csv, df_test.csv*).

Stratified Sampling

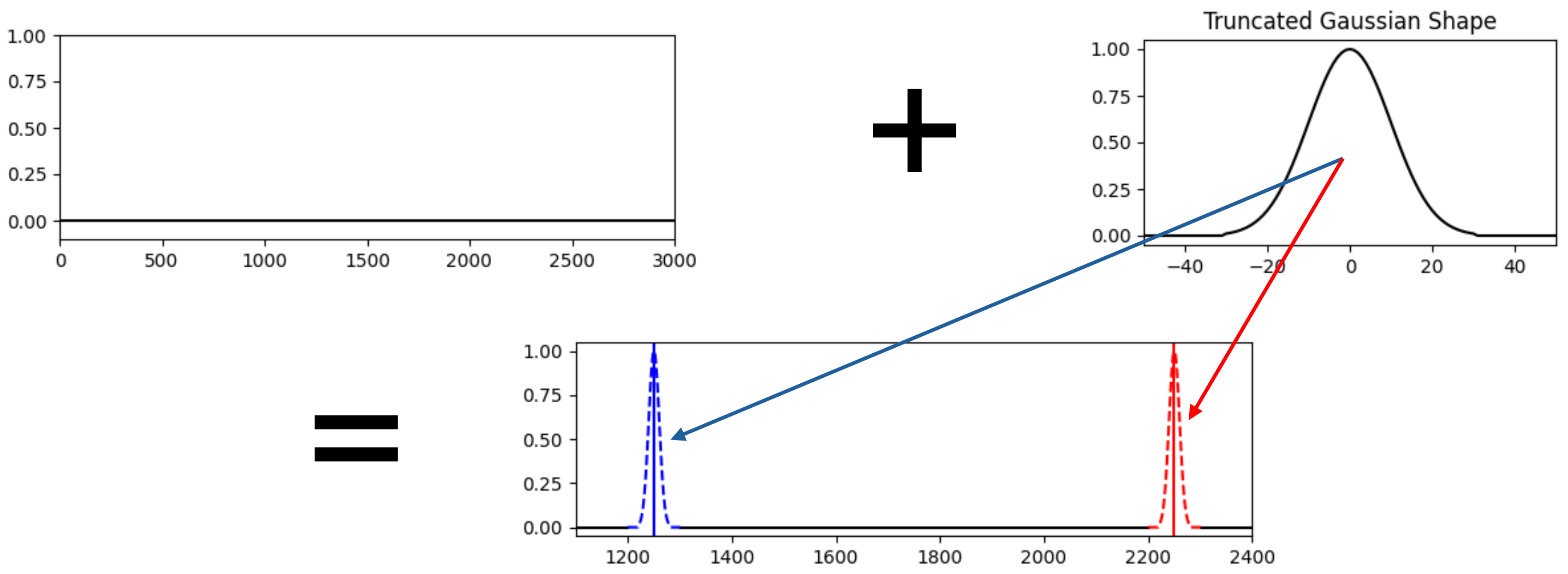
- The purpose of stratified sampling is to retain the proportions (of stations, in this case) of the original dataset/population when taking a smaller sample.
  - In other words, the proportion of each station should be the same in the full dataset, the train split, the validation split, and the test split.
  - For example, the proportion of NC.GDXB is 25,173/632,000 = **0.0398** in the train split, 3,145/79,000 = **0.0398** in the validation split, and 3,145/79,000 = **0.0398** in the test split.
  - This ensures that models are trained on the same data distribution that they are tested on.

network.station	train_count	val_count	test_count
NC.GDXB	25173	3147	3147
NC.MDH1	22157	2770	2770
BK.SCZ	12414	1552	1552
NC.BJOB	11829	1479	1479
BK.SAO	11476	1434	1434
NC.MDPB	10840	1355	1355
NC.MCB	10360	1295	1295
NC.MINS	8976	1122	1122
BK.MHC	8884	1110	1110
BG.SQK	8850	1106	1106
NC.MMX1	8645	1081	1081
NN.OMMB	7889	986	986
CI.MLAC	7760	970	970
NC.BBGB	7691	961	961
BG.SB4	7324	915	915
BG.CLV	6923	865	865
BK.PKD	6663	833	833
NC.MQ1P	5648	706	706
BG.HVC	5598	700	700
BG.FUM	5267	658	658

The 20 most represented stations in the dataset.

HDF5 PyTorch Dataset Benchmarks

- The *pytorch_hdf5_dataset.py* script was created to test the efficiency of various HDF5 settings when training deep learning models.
- All three HDF5 implementations from slide 22 are evaluated by fully iterating through a PyTorch Dataset (i.e., benchmarking the data load time of one epoch).
- Because the dataset labels (slide 23) are derived from only the p_sample and s_sample attributes, the script tests both loading the saved labels, and computing the labels on the fly.
    - The exception is implementation 3, which does not store any attributes.
    - To maximize efficiency, the truncated Gaussian is computed only once and inserted into zero arrays at the p and s sample locations (see figure below).



HDF5 PyTorch Dataset Benchmark Results

- Benchmark results are shown in the table to the right.
- We can make a few definite conclusions:
  1. The computational cost of computing the dataset labels is far lower than the cost of reading more data from disk.
  2. There is a significant overhead (2.5 x) when ucompressing gzip compressed datasets.

	Saved Labels	Computed Labels
Implementation 1	412 s	242 s
Implementation 2	1,030 s	674 s
Implementation 3	248 s	N/A

- Although gzip compression introduces significant overhead, additional testing is required to determine if this overhead would actually slow down model training.
  - With computed labels, the load time for one batch (size 32) is approximately 674/19,750 = .034 s, which could very well be faster than the time needed to train one batch on the GPU.
  - This overhead could be further mitigated by using multiple CPU cores to load data.
  - On the other hand, much of the value of gzip compression could be lost if labels, which are mostly zeros and easily compressible, are no longer stored.
- In conclusion, we know that computing labels on the fly is more efficient than saving them in the HDF5 file, but that further investigation is needed to determine the value of using gzip compression.

List of Assumptions and Remaining Questions

- Assumptions:
  1. It's okay to match a P arrival recorded on one instrument with an S arrival recorded on another (slide 4).
  2. When associating 3-component waveforms (*download_waveforms.py;* slide 12), we assume that all traces returned from NCEDC with a matching station and instrument appear together in the stream.
     - If this assumption no longer held, the pipeline would seriously break.
  3. Waveforms with P/S distance < 0.35 s were filtered out by the PhaseNet authors (slide 25).
     - Since the P(p) and P(s) time series are normalized 0-1, this seems like the only option, but maybe the PhaseNet authors employed an additional processing step to address the problem.
  4. Random (stratified) sampling from the 1,223,376 complete 3-component waveforms retrieved from NCEDC produces an equivalent dataset to the PhaseNet article (slide 26).
     - It is very possible that the PhaseNet authors employed additional filtering criteria to the waveforms, since their dataset (N=779,514) is much smaller than the 1.2 million waveforms retrieved by the pipeline.
- Remaining questions:
  1. Can anything be done to reduce the HDF5 overhead? (slide 22)
     - Some of this overhead may be due to losing the Steim2 compression employed by miniseed files, but the relatively small file size of implementation 3 seems to somewhat contradict this.
     - Another cause of data size increase is the resample function (slide 25), which converts int32s to float 64s. What are the implications of storing float32s instead? How much accuracy is lost?
  2. Does the overhead introduced by gzip compression of the HDF5 datasets actually slow down model training?