Notes on `hsi_xfer`
General usage documentation:
https://docs.olcf.ornl.gov/systems/2024_olcf_system_changes.html#data-migration

- **What**: hsi_xfer is a wrapper around the standard HPSS hsi tool that optimizes recalls from tape, while minimizing impact to the aging HPSS system and hardware. Additionally, the script provides data integrity protections/checksumming and checkpointing similar to what globus provides.
- **Why**: We created the script to better manage HPSS and DTN resources during this period of mass retrieval and prevent thrashing of the library robotics, and give users a less complicated way to retrieve their data. Additionally, this script provides an optimized way to retrieve data from tape, by batching up all the files you want from any individual tape and streaming them off all at once, minimizing robotics movement and tape loading/seek time and wear and tear on our tapes. This means, the best way to use the script, is to provide larger directory trees, rather than smaller ones, so that you can stream more data off of the tape at any one time.
    - **How it works**:
        - Given an input and output path, it will index recursively all files in the input path in HPSS, and any existing files in the destination path an save this information to a cache file. Any files that it finds in the output path that match in name and tree position, as well as size, will be skipped (this can be overridden with the --overwrite-existing flag
        - Next, it will group files based upon which tape volume the files live on, so that we can get close the to the optimal "streaming all data from a single tape" state, so each tape has its own 'filelist'
        - Next, it will create a checksum for all files it has indexed on HPSS. this can be skipped with the --disable-checksums flag, but we can't guarantee file integrity if it is disabled
        - The actual transfer of files will start. The tool launches 2 parallel hsi's, one tape's filelist per thread
        - After the transfer is complete, it launches a threadpool of 4 threads to md5sum all files it transferred.
        - Finally, the checksums are compared and a report is generated. If any checksum does not match, it will report this as an error and in the JSON report file as well. This report will be written to your current working directory (or the working directory provided with the -f flag)
    - **Common flags:**
        - --overwrite-existing overwrites existing files
        - --disable-checksums disables the checksumming mechanism; this can increase performance at the loss of guaranteed file integrity
        - --preserve-cache: normally, the script will clean up the cache if it completes successfully. If it gets killed or this flag is set, it will keep the cache in your current working directory (or directory defined by the -f flag). This cache can be passed back into the script with the --cache-path flag,

to skip the indexing and hpss checksumming stages and pick back up directly where it was killed/left off. *this is only useful for jobs that take many hours to index hpss files (generally >1million files). Otherwise, it does not buy you anything and could cause confusion and state drift*

- --preserve-timestamps: preserves the timestamp metadata of the files in HPSS, otherwise, it defaults to normal hsi behavior
- --update-interval: during long running transfers, the script defaults to a update interval of 5 minutes, meaning that it will print an update saying 'm/n files transferred' every 5 minutes. This can be used to override this interval to make it longer or shorter
- --enable-log-timestamps: simply adds a timestamp to each line it prints to stdout as it runs. useful for long running jobs when you want to know when the last time an update was printed to the screen
- --dry-run: this will perform the file indexing and filelist generation but skip the checksumming and transfer stages. This is useful with the --preserve-cache flag, in order to index files and create a cache that provides a 'sane' starting point for jobs that are many millions of files

- **Common pitfalls:**
  - You get an error about 'stale file handle' or 'database locked':
    - The cache is built around a sqlite database, which historically does not work very well with networked filesystems, like your NFS home directory. Recommendations are to either run the script from /tmp or kronos, or use the -f flag to change the 'working directory'
  - Advanced globbing:
    - Advanced globbing is not supported, just simple globs. This is a limitation of hsi
  - Checksumming/indexing takes a long time:
    - If you're trying to transfer >1million files, then I suggest either generate a cache with --dry-run --preserve-cache and use that each time you launch a transfer, or break up your transfer into smaller directory trees. Your files will transfer, but we're limited in queue size in HPSS and resources on the DTNs
  - The script dies/failed/gets killed:
    - Simply restart! It will pick back up where it left off, no cache required. The cache simply allows you to skip the indexing and checksumming stages, which doesn't really save you any time unless you're moving >1million files
  - The script gets killed when your ssh session times out:
    - Run your script through screen or tmux
  - Help I can't parallelize this!
    - This is by design. The script handles any parallelization to the level we've deemed acceptable. The script will prevent you from running multiple instances of the script by design. Please don't try to run more than that.