FEMA

Department of Civil & Environmental Engineering
COLLEGE OF ENGINEERING
UNIVERSITY OF WISCONSIN–MADISON

THE UNIVERSITY of NORTH CAROLINA at CHAPEL HILL

Auto-regulating Input Files that Save Time when Jobs Fail

Daily Precipitation File Analysis with Item-level retries in an HTCondor / DAGMan pipeline when files are missing

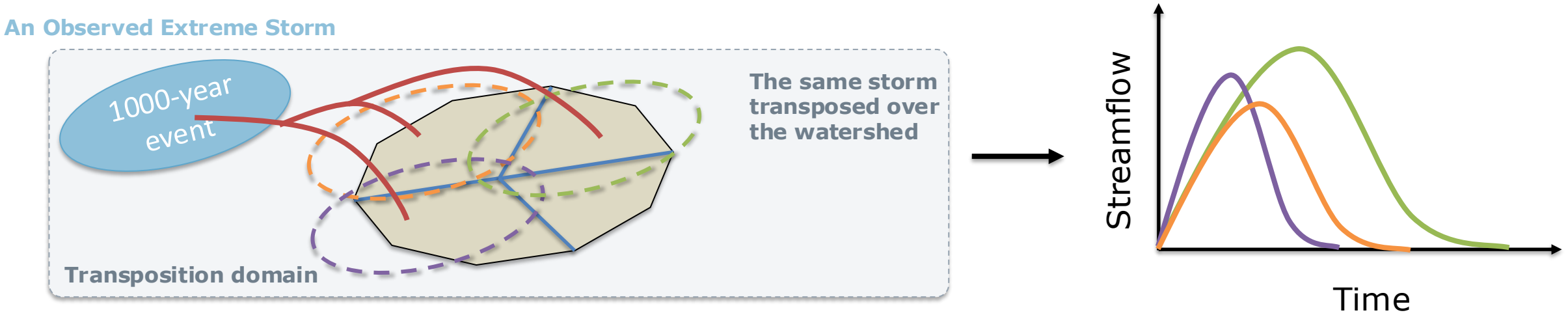
Benjamin FitzGerald

PhD student, Hydroclimate Extremes Research Group (Dr. Daniel Wright)

HTCondor Week

Storm Transposition and Rainfall-runoff modeling

- Engineers design dams, bridges, and floodplains for rare storms (the 100- or 1,000-year event). A rainfall-runoff model turns a storm into the flood it produces, so it needs rainfall estimates far out in the tail.
- The observed record is short, so its biggest storm is nowhere near a 1,000-year event. Stochastic storm transposition (SST) extends the record, by moving observed storms
- Spatial L-Moments of Annual Maxima (SLAM) is a workflow for creating the transposition domain

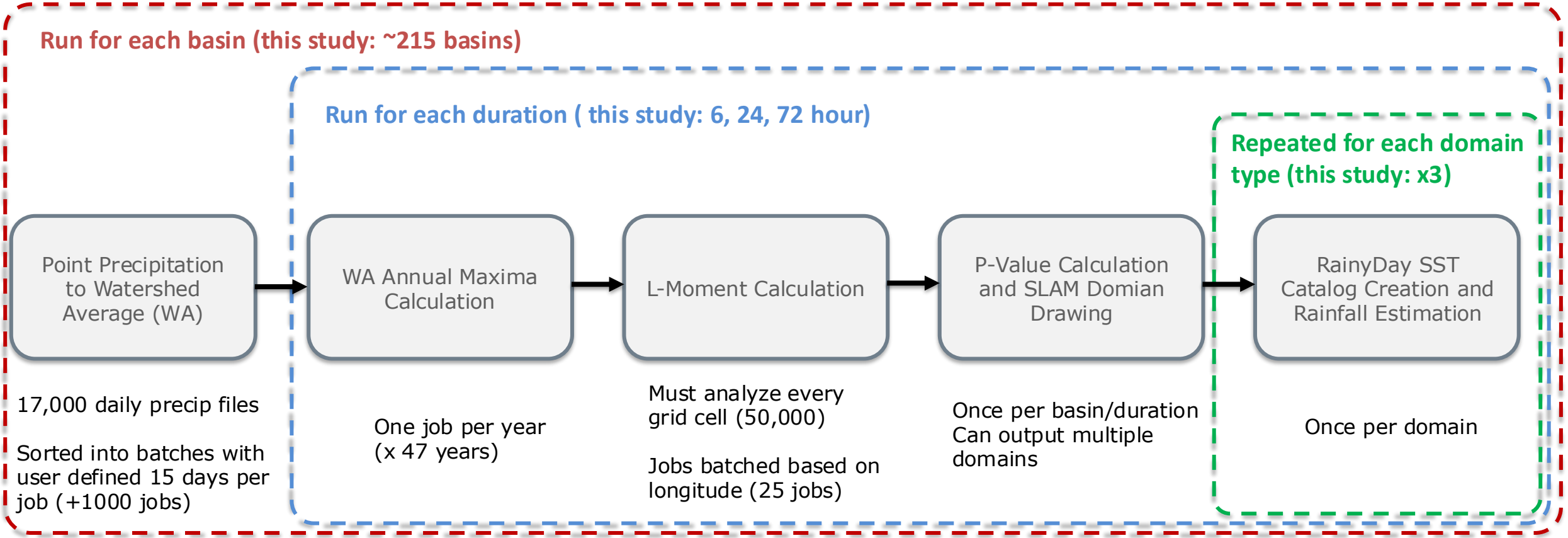


SST shifts real storms to many positions over the basin, building a larger synthetic set of extreme events than the short record alone

This work has been used in FEMA's modernization of National Flood Insurance Program's Rate Maps

SLAM-SST Pipeline

Whole process could be run for any basin and gridded precipitation dataset

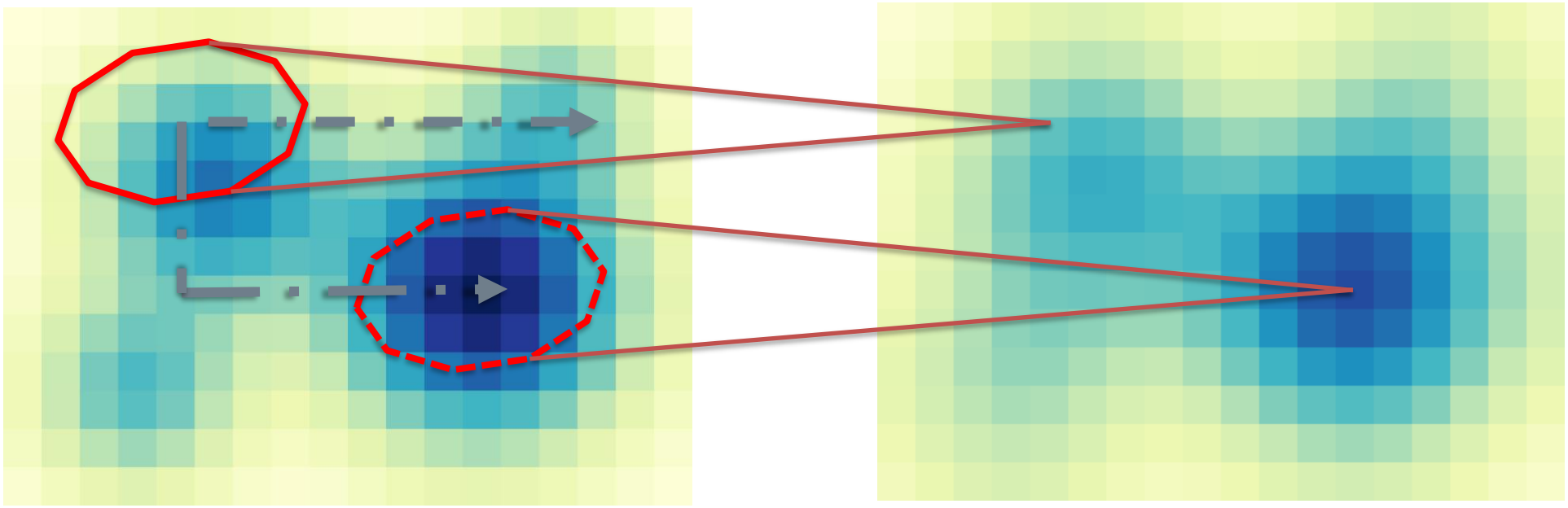


What Point Precipitation to Watershed Average Precipitation (PP2WAP) job computes

Jobs: Watersheds [215] x 400,000 hours of precip data

1-hr Accumulation Precipitation Map

1-hr Watershed Average Precipitation Map



Calculate watershed average precipitation at every possible transposition location. Store in output file.

Submit Files

```
Original Submit file piece:
transfer_input_files = Other inputs,
AORC.$(Day1).Precip.nc,
AORC.$(Day2).Precip.nc,
AORC.$(Day3).Precip.nc,
AORC.$(Day4).Precip.nc,
AORC.$(Day5).Precip.nc, …

…
queue Day1, Day2, Day3, Day4, Day5 from dates.csv
```

```
dates.csv:
Day1, Day2, Day3, Day4, Day5
01011979, 01021979, 01031979, 01041979, 01051979
01061979, 01071979, 01081979, 01091979, 01101979
01111979, 01121979, 01131979, 01141979, 01151979
…
```

Problems:

- What if I'm using a different data resolution and need more or less than 5 files per job?
- What if our total files isn't divisible by 5?
- What if I need to rerun a select number?

We can manage submit files and we can manually or automatically populate the .csv file.

But is there a way to have 1 submit file that works with any number of inputs per job?

Dynamic input submit files

```
Remove transfer_input_files from Submit File and add:
queue transfer_input_files from PP2WAP_Input.txt
```

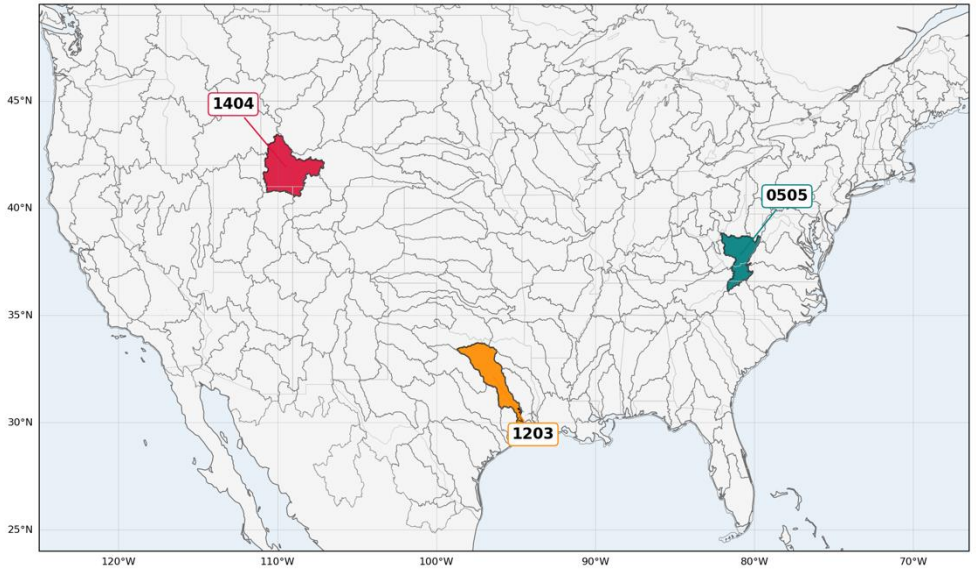
```
PP2WAP_Input.txt:
AORC.19790101.precip.nc, AORC.19790102.precip.nc, …, WS.shp, PP2WAP.py
AORC.19790101.precip.nc, AORC.19790102.precip.nc, …, WS.shp, PP2WAP.py
AORC.19790101.precip.nc, AORC.19790102.precip.nc, …, WS.shp, PP2WAP.py
…
```

PP2WAP_Input.txt is generated by a script that would take user input over what files are necessary, how many to batch per job

- Just allows you to use same job scripts (.sub, .sh, .py) with different batch sizes (useful for different resolution dataset and when rerunning portion).
- Could also be used to run different inputs (i.e watershed shapes) and python scripts for same job

SLAM-CONUS with DAGman

SLAM-CONUS: Goal to create transposition domains for all 215 HUC4 basins



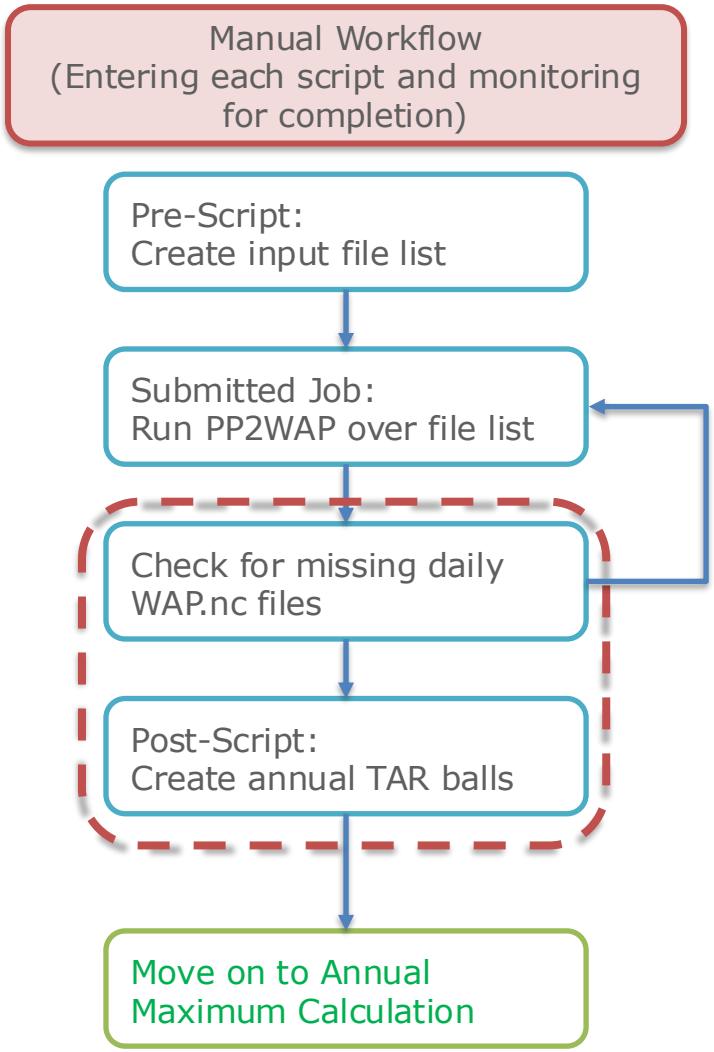
Hard to manually run per basin workflow with proper error checking for 300,000 PP2WAP jobs

DAGman breaks up pipeline into a series of nodes and manages pre-processing, running jobs and post-processing.

Issue:

We need to have the complete processed dataset and sometimes outputs go missing

Original Workflow



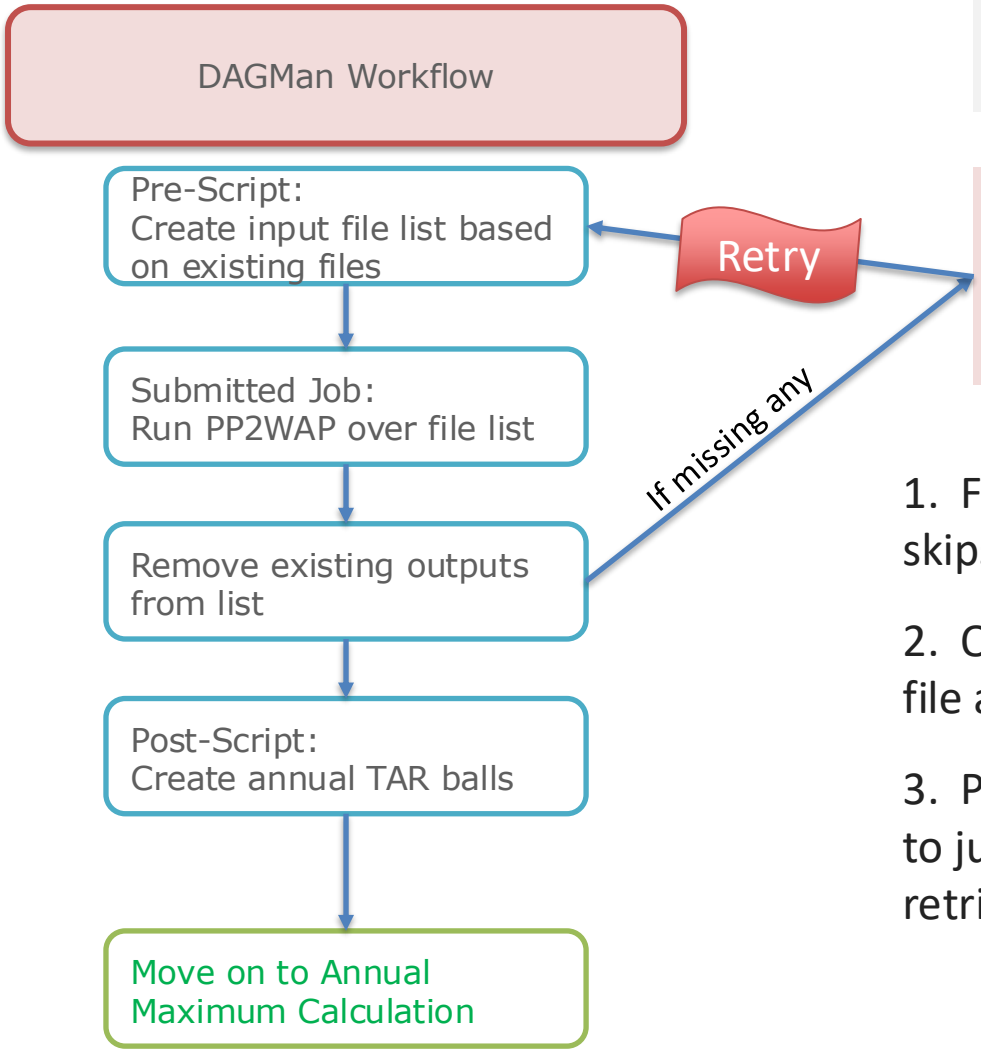
Can manually alter file list to only rerun missing days

With shift to DAGMan, can't manually check before telling it to move on.

DAGMan has some capabilities for covering jobs failing (retry, submit file max_retries, periodic_release)

No transfer_output_files line in submit file

The pattern




```
PP2WAP_Input.txt:
AORC.19790101.precip.nc, AORC.19790102.precip.nc, …, WS.shp, PP2WAP.py
AORC.19790101.precip.nc, AORC.19790102.precip.nc, …, WS.shp, PP2WAP.py
AORC.19790101.precip.nc, AORC.19790102.precip.nc, …, WS.shp, PP2WAP.py
…

1,143 Lines for 17,136 daily files
```

```
Edit PP2WAP_Input.txt:
AORC.19830714.precip.nc, AORC.20010322.precip.nc, …, WS.shp, PP2WAP.py
…

1 Lines for 2 daily files
```

1. First run: the PRE script writes every item that still needs doing, and skips any whose output is already on disk (unless over-write flag).

2. On a retry: if a .retry flag is present, the PRE script keeps the shortened file as it is, then clears the flag.

3. POST script: it checks the outputs. If any are missing, it rewrites the file to just those items, creates the .retry flag, and exits non-zero so DAGMan retries the node.

   - If no shrinking file, DAGMan retries nodes with all files

The same scripts in every stage

The same PRE and POST logic runs in all five stages. Only the unit of work changes:

Stage	Item unit
PP2WAP	Daily Precip files
AMC	one year
LMC	Longitude chunk
CLMPV	one domain
RDCatalog	one year

Final Thoughts

Cautions:

- The set of job codes needs to be written to be compatible with files given.
  - If processing code is looking for AORC.YYYYMMDD.nc but input_file.txt can give C404.YYYYMMDD.nc
- An empty queue file does not submit zero jobs. queue x from <empty file> submits one job with x empty. When a stage is finished, I write a DONE line and exit early, or that one job ends up held.

Summary:

Have each batch stage read its list of work from a file.

The PRE script fills that list from what is missing on disk. The POST script trims it to the failures.