HTCondor Tips – site configuration

- HTCondor – very flexible batch system
  - FZU twiki provide basic documentation for local users
  - Introduction for HTCondor @ FZU still valid
  - Other sites using HTCondor may provide different user experience / JDL options & features
  - Attempt to use compatible custom job submission options with CERN
    - e.g. +MaxRuntime / +JobFlavour – keep support in future
    - standard HTCondor 10 alternatives (FZU twiki update once we upgrade)
- High Throughput Computing (HTC) – optimize resource usage
- Service administrator vs. users
- Limited service administrators experience with more complex workflows
  - Zero experience with DAG jobs
  - Zero experience with MPI jobs
  - No tuning for unknown / unusual workflows build into our HTCondor configuration
    - Optimized for grid workflows and optimal resource usage

HTCondor Tips – job sumission

- Keep it simple
    - You don't have to use all features provided by HTCondor
    - condor_submit with 5 lines in JDL, condor_q and condor_rm might be sufficient
        - Combine with power of shell scripts or python bindings for HTCondor
    - Only more complex and repeated workflows might benefit from advanced features
- Advanced topics including users themes in a yearly HTCondor Week, e.g.
    - HTCondor CLI tools with links to YouTube tutorials
    - Submitting multiple jobs
    - Data transfers
    - DAG
    - … HTCondor Week 2022 US / EU version …
- Youtube Channel with user tutorials
    - Recordings from (recent) HTCondor Week / Workshop

HTCondor Tips – running jobs

- Only small fraction of resource available for users that are not members of following groups: atlas, alice, auger, dzero, nova, dune
  - Fairshare for these groups based on their contribution
    - `condor_q -af ClusterId ProcId` ***`AcctGroup`*** `AcctGroupUser`
  - 50% of resources usually reserved for grid jobs
  - => one user can get thousands cores (for "reasonable" period of time)
- Always specify reasonable resource requirements
  - request_cpus (1), request_memory (2GB), +MaxRuntime (1800)
  - higher resource requirement => more difficult to find resources for job => longer time to start job
    - idle resources while waiting for jobs to finish and make sufficiently big job slot

HTCondor Tips – resources

- Improved support for arbitrary CPU & memory requirements
  - Still no support for scheduling whole-node jobs
- Be careful specifying your own job `requirements`
  - Might take long time to find matching resources
  - Avoid matching subclusters by name / regex
  - Matching CPU features might be also problematic with current scheduling
    - has_avx, has_avx2, has_sse4_1, has_sse4_2, has_ssse3, … (see `condor_status WN`)
- User jobs without special requirements should start quickly
  - `condor_q –analyze`, `condor_q –better-analyze jobid`
  - ask if jobs gets queued for hours (special job requirements)
- Job priority in JDL is just suggestion and may not be respected in HTC

HTCondor Tips – I/O

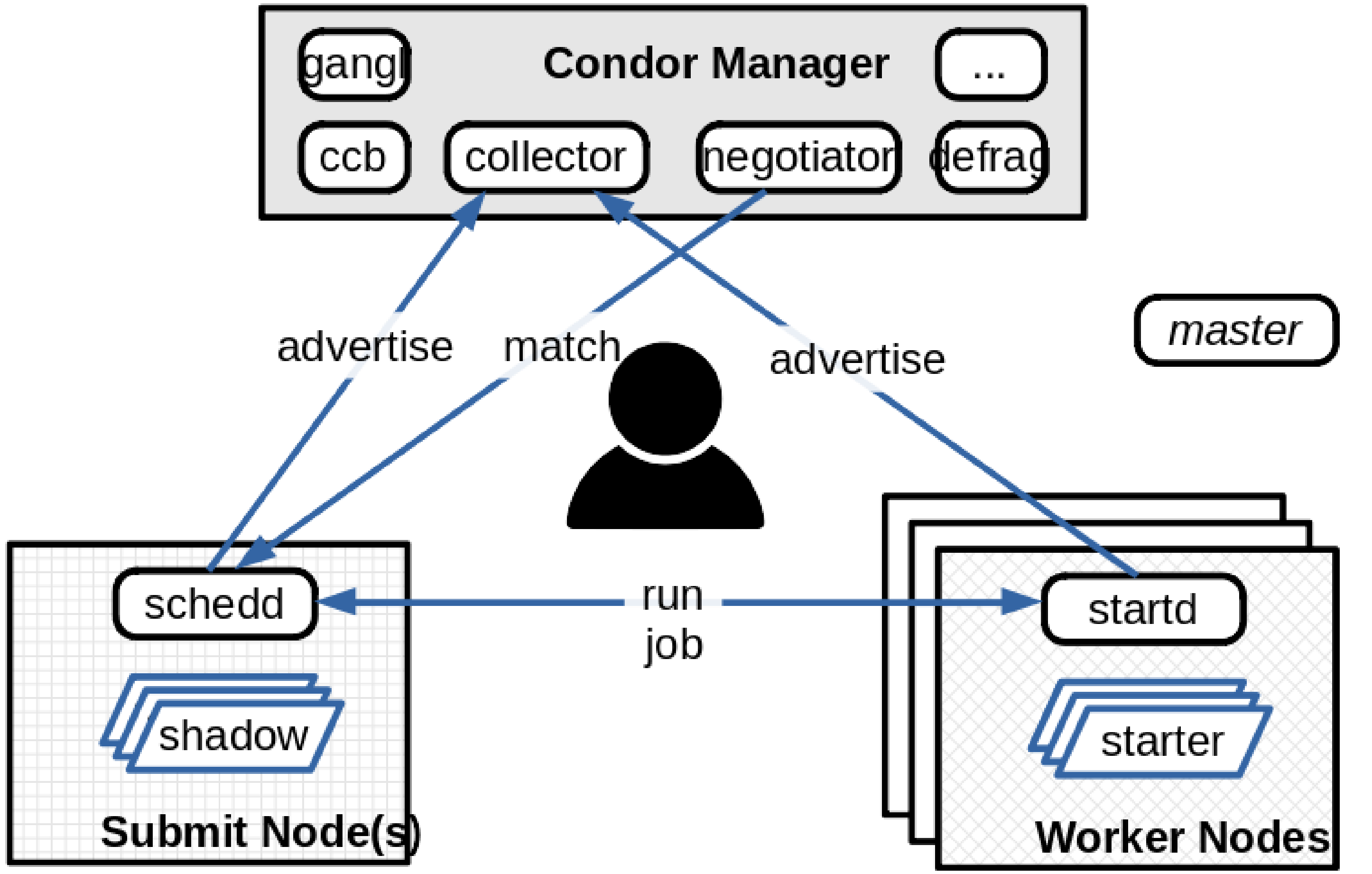
- Not difficult to I/O overload (random data access) your own machine
  - Shared NFS filesystem hammered by thousands of jobs
  - Most of our storage servers build for capacity and not I/O
    - Your laptop have 2 order of magnitude better I/O then our NFS servers
  - Concurency limits – details in FZU HTCondor TWiki
- Use local WN scratch
  - Random I/O much more stressful for shared filesystem
  - Prefer whole file upload at the end of your jobs
  - Never use NFS for temporary files (even small ones, 8 bytes MC counter can kill NFS)
  - Jobs should not depend on files from $HOME
- Try to avoid (big) file transfers with transfer_{input,output}_files
  - Our batch not tuned for this use-case => overload job submission node
  - Unnecessary transfers: NFS <-> submission node <-> worker node
- Use local grid storage `se1.farm.particle.cz` (WebDAV, xroot, GridFTP protocols)
  - Load distributed between 10+ servers (but still with single file replica)
- Distribute load over multiple NFS servers (11) by using multiple file replicas
  - Big read-only files with random access (e.g. sqlite DB)
  - CVMFS much more scalable with local WN cache

HTCondor Tips – limits

- Batch configuration: `condor_config_val condor.farm.particle.cz -dump`
  - Maximum jobs per user 100k
  - Maximum jobs per submission 20k
  - Maximum running jobs per user 10k
  - Maximum 30 days of Walltime time, but only 28 days of CPU time (multicore sum)
    - only ~ 5 hours with 128 core jobs
  - Automatic job re-try 10x
- User submission node resources (16GB RAM)
  - Should deal with 5k running (50% user share) and 100k queued jobs
  - Scheduling individual jobs increase CPU load (`queue N` in JDL)
- Total cores ~ 12k, 160k HEPSPEC06 (13.5k till the end of 2022), hierartical fairshare (1day halflife) for VOs
  - ATLAS ~ 40% => ~ 3600 user jobs (assuming there are always grid jobs)
  - Auger+CTA ~ 30% => ~ 1800 user jobs (3600 if there are no grid jobs)
  - ALICE ~ 20% => ~ 1200 user jobs (assuming there are always grid jobs)
  - NoVA+DUNE ~ 10% => 600 user jobs (assuming there are always grid jobs)
  - No jobs for some VO => its share divided among the other VOs
- Maximum worker node HW limits
  - 128 cores (hyperthreading enabled)
  - 512GB RAM

BACKUP

HTCondor Tips


gangl
Condor Manager
...
ccb
collector
negotiator
defrag
advertise
match
advertise
master
schedd
run
job
startd
shadow
starter
Submit Node(s)
Worker Nodes