



Argonne
NATIONAL
LABORATORY

... for a brighter future

ArCond and parallel data processing at Tier3

Sergei Chekanov (ANL)

June 2010.

USATLAS Tier3 meeting



U.S. Department
of Energy

UChicago ►
Argonne_{LLC}

A U.S. Department of Energy laboratory
managed by UChicago Argonne, LLC

Condor batch system for ATLAS Tier3

- **Condor – a popular batch system for parallel job processing**
- **For ATLAS-specific tasks, it requires additional features:**
 - **better compatibility with IO intensive tasks**
 - data discovery mechanism. Data can be are either on a central file server or uniformly distributed on computers (ANL preferred option)
 - Merging, moving, copying outputs from each core
 - **Requires a better integration with Tier3 data analysis tasks, such as:**
 - Submission of user athena packages
 - MC generation and simulation
 - running ROOT/C++ code over D3PD's or ntuples
 - running custom packages (like NLO QCD etc.)

ArCond (*Argonne's Condor*)

<http://atlaswww.hep.anl.gov/asc/arcond/>

■ A Condor front-end:

- job submission
- data discovery
- checking job status
- merging outputs

■ A PYTHON package which talks to the standard condor commands and provides an interface between Condor and a user for IO intensive jobs with input and output

■ Minimum requirement: PYTHON (any version) and ROOT

■ Designed for analysis of data flatly distributed over multiple PCs

■ Can also be used in a combination with XROOTD

ArCond. What is this?

<http://atlaswww.hep.anl.gov/asc/arcond/>

Optimized for speed: a file-based database. Can work with xrootd.
Fetching 30k file names from 10 computers takes 0.5 sec
Takes ~2-3 sec to update a database with ~10k files. Fully scalable

Submission steps:

- input=/data09
- prepare submission scripts with input option files
- Tar a user package (if needed)
- Submission
- When ready, copy the outputs, move, merge etc..



Check a database, build a list with all files (6 files)

Submits $N(\text{cores}) \times N(\text{computers})$ jobs
Each job has its own input list



A condor batch system

/data09/AOD1.root
/data09/AOD2.root

/data09/AOD3.root
/data09//AOD4.root

/data09/AOD5.root
/data09/AOD6.root

Running ArCond at Tier3s

- Setup ArCond as:
 - For ATLAS cluster (atlasXX.hep.anl.gov), just setup any release
 - <https://atlaswww.hep.anl.gov/twiki/bin/view/Workbook/SettingUpAccount>
 - does not use XROOTD
 - For ASC cluster:
 - `source /export/home/atlasadmin/condor/Arcond/etc/arcond/arcond_setup.sh`
 - uses XROOTD
- Go to some directory (tmp) and do: `arc_setup`
- You will see the structure:
 - `arcond.conf` – user configuration file (look at it!)
 - `user/Analysis_jobOptions_BASIC.py` – analysis job option file. Can be anything
 - `user/ShellScript_BASIC.sh` – a user shell script executed on each node (with `athena.py XX` if you need!)
 - `Job` (directory) where the submission scripts are generated (do not edit)
 - `DataCollector` (directory) where input file lists are collected (do not edit)
 - `patterns` (directory) can specify computers for submission (usually configured by admins, but you can change it)

Some help

- Type “arc_help” for help:

```
--- ArCond help ---  
  
arc_add      >> Merge all output ROOT files located in Job/**/*  
arc_check   >> Check outputs  
arc_clean   >> Clear all submissions from previous runs  
arc_cp      >> Copy and rename all output files located in Job/**/*  
arc_exe     >> Run a shell script. Usage: arc_exe -i script.sh  
arc_get     >>  
arc_ls      >> lists all files in a dataset. Usage: arc_ls <data set>.  
arc_mv      >> Move and rename all output files located in Job/**/*  
arc_setup   >> Setup script. Initialize ArCond directory structure  
arc_setup_admin >> Setup Admin script. Initialize tools for administ  
arc_split   >> Split dataset for multiple nodes (admin tool)  
arc_ssh     >> Parallel ssh to the set of nodes (admin tool)  
arc_update  >> ArCond update script  
arcond     >> Main submission script for a T3g PC farm  
----- Done -----
```

For the next slides I'll assume the ASC-cluster setup

For ANL cluster, everything is identical only “xrootd” is not included to the file path

Submitting with Arcond

■ Check data availability using “arc_ls”. Example:

- `arc_ls /xrootd/atlastier3/data09_900GeV`

Output:

- `ascwrk1.hep.anl.gov:/xrootd/atlastier3/data09_900GeV/ESD/r988/data09_900GeV.00141994.physics_MinBias.recon.ESD.r988_tid101490_00/ESD.101490._000009.pool.root.1`
- `ascwrk1.hep.anl.gov:/xrootd/atlastier3/data09_900GeV/ESD/r988/data09_900GeV.00141994.physics_MinBias.recon.ESD.r988_tid101490_00/ESD.101490._000118.pool.root.1`
- `ascwrk1.hep.anl.gov:/xrootd/atlastier3/data09_900GeV/ESD/r988/data09_900GeV.00141994.physics_MinBias.recon.ESD.r988_tid101490_00/ESD.101490._000057.pool.root.1`
- `ascwrk1.hep.anl.gov:/xrootd/atlastier3/data09_900GeV/ESD/r988/data09_900GeV.00141994.physics_MinBias.recon.ESD.r988_tid101490_00/ESD.101490._000167.pool.root.2`
- `ascwrk1.hep.anl.gov:/xrootd/atlastier3/data09_900GeV/ESD/r988/data09_900GeV.00141994.physics_MinBias.recon.ESD.r988_tid101490_00/ESD.101490._000177.pool.root.2`
- `ascwrk1.hep.anl.gov:/xrootd/atlastier3/data09_900GeV/ESD/r988/data09_900GeV.001`
- `.....`

Arcond recursively scans all files inside the root directory
`/xrootd/atlastier3/data09_900GeV`

Submitting with Arcond

- Before submitting, check what you are doing. Read “arcond.conf”
 - atlas_release=15.6.9 # atlas release
 - events = 100 # events for each job (-1 for all)
 - input_data = /xrootd/atlastier3/data09_900GeV/ # input file locations
 - *set to empty if no input*
 - *can also be a central storage (not distributed among many computers)*
 - max_jobs_per_node=14 # (for 16-core PowerEdge R710 . put -1 for all cores)
 - package_dir=/users/15.6.5/NtupleMaker # athena user package

Then type “arcond” to submit the package

The default arcond.conf is pre-configured to submit “NtupleMaker” package to run over distributed by xrootd ESD files

Submitting a job..

- Unlike **pathena** and **dq2**, Arcond talks to the user and asks 4 questions
- Say “no” to interrupt the submission

```
chakanau@ascwrk1:tmp$ arcond
Input configuration=/users/chakanau/tmp/arcond.conf
---> Input data located at = /xrootd/atlas tier3/data09_900GeV/ESD/r988/
---> Nr of included PCs = 3
---> Maximum number of cores per node = 14
---> Checking computing cores
-->1 PC node=ascwrk0.hep.anl.gov with=15 cores found, but 14 is used
-->2 PC node=ascwrk1.hep.anl.gov with=15 cores found, but 14 is used
-->3 PC node=ascwrk2.hep.anl.gov with=15 cores found, but 14 is used
---> Total number of found cores= 42
```

We told to use 14 cores in arcond.conf

Start the data discovery tool? (say "y" for yes, "n" for no)

```
--->OK, discover the data using ArCond database
-->1 PC node=ascwrk0.hep.anl.gov with=15 cores found
-->1 PC node=ascwrk2.hep.anl.gov with=15 cores found
-->1 PC node=ascwrk1.hep.anl.gov with=15 cores found
```

385 AOD files found on all nodes.
All duplicate files are removed (if any)

```
---> Checking for duplicate input data files
--> ## SUMMARY: Total number of input files = 385
Project file:/users/chakanau/tmp/Job/NtupleMaker.tgz was found.
```

For ANL cluster, there is extra option “f” (discover “on-fly”)

Do you want to rebuild it (y/n)? y

```
---> Final checking computing cores
---> Root package directory = /users/chakanau/public/2010_jamb_march/testarea/15.6.5/
---> Package name = NtupleMaker
---> Package submission file = Job/NtupleMaker.tgz
---> Package submission log file = Job/NtupleMaker.log
---> Number of events in one job = 100
---> Atlas release = 15.6.3
---> Nr of available PC nodes = 3
---> Nr of available CPUs in all PC nodes= 42
---> Data splitted for = 42 ranges
---> 42 jobs will be submitted to = 3 PCs
```

Submitting a job.. continue..

Do you want to prepare the submission scripts (y/n)? y

---> Submission scripts in Job/* are ready

Submit all prepared jobs to the PC farm? (y/n)y

ROOT submission directory=/users/chakanau/tmp/Job/

Submitted to:run40_ascwrk0.hep.anl.gov

Submitted to:run3_ascwrk2.hep.anl.gov

Submitted to:run25_ascwrk1.hep.anl.gov

Submitted to:run34_ascwrk0.hep.anl.gov

Submitted to:run36_ascwrk0.hep.anl.gov

.....

.....

Check the status using **condor_q**

Job submitted!

- Run condor commands: **condor_status** or **condor_q**
- Your jobs are in “idle” state?
 - check who is running on the farm as:
 - **condor_status -submitters** (OR) **condor_q -global**
- Check output files as: **arc_check**
- If **arc_check** tells that all output files “Analysis.root” are ready, combine output files to one file using **arc_add**. This creates “Analysis_all.root”
- Use **arc_mv** or **arc_cp** if want just keep the outputs from each core without merging

What jobs can be submitted

- Any. Just define your execution sequence in “`user/ShellScript_BASIC.sh`”
- Can be any sequence of Linux commands, `cp`, `sync`, `untar` etc.
 - For athena jobs runs “`athena.py Analysis.py`”
 - “`Analysis.py`” is rebuilt from the template “`Analysis_jobOptions_BASIC.py`”
- But can also be used for very custom programs:
 - we are currently using for NLO QCD and MC generation.
- Compilation of athena user programs on each processing core is not needed

New features (since March 2010, version 1.6)

- User history file. Keep information about which input data (and which releases) are used by looking at “.arcond_history” in user directories. This is a simple CSV file with the structure:
 - CSV Structure: Input data, Athena version, package, submission date
- More options to scan ArCond database using **arc_ls**:
 - “grep”-type regular expressions to scan the database using **arc_ls**
 - can generate file summaries etc.

```
USAGE:
arc_ls <path> # list all files on all nodes in the <path>
arc_ls -f <path> # as above
arc_ls -e <path> # use <path> as the pattern. Identical to "grep -e <path>"
arc_ls -ef <path> # as above
arc_ls -d <path> # list all directories
arc_ls -s <path> # generate summary of the files
arc_ls -h # show this help
arc_ls -v <path> # list with the information on databases
```

Some useful features and summary

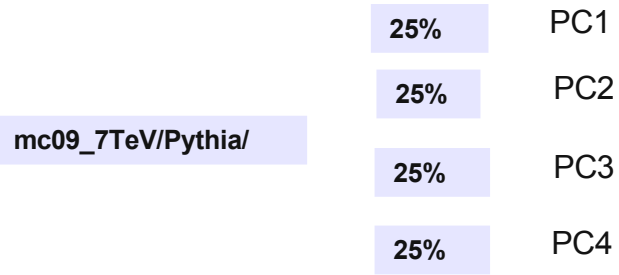
- **Simple. Does not have any server, daemon processes etc..**
 - ArCond is a set of python scripts and does not require maintenance
 - (but Condor does)
- **Designed for easy debugging**
 - Can simulate execution of a certain job on interactive cluster
 - Generate submission scripts (in Job) but say “no” to submit
 - Run the bash shell script “*ShellScript.sh*” locally
- **Well tested**
 - ~500 analysis jobs since 2008 (each 24-200 CPU's) <0.1% fault rate
- **Current operation:**
 - serving 16 PowerEdge 710
 - 6 TB of locally distributed data (900 GeV & 7 TeV)
 - 50% D3PD, 40% AOD, 10% ESD
 - all MinBias1.5 data and MC (AOD/D3PD) , J1-6 samples, Egamma D3PD, etc.

Backup

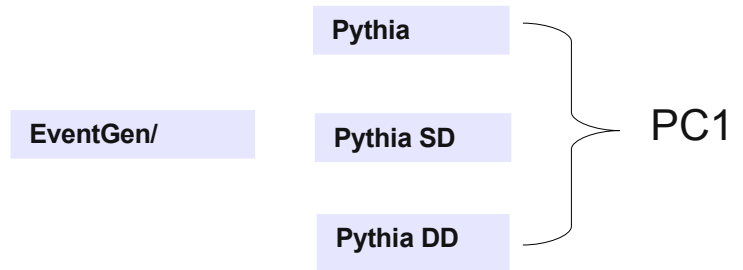
How to distribute data on multiple nodes..

- All depends on the user wishes..
- So far 3 types of data deployments:
 - 1) A single data set evenly distributed on all nodes. **Best for MC**
 - 2) A data set should be located on a single computer
 - *Example: Do not distribute single runs on multiple computers.*

Typically used for data



Usually for MC



ArCond has some simple tools (python+rsync) to copy data. For 3) just use rsync or scp!

Example II. Generation of MC truth

- Login on ASC cluster
- Setup: `source /export/home/atlasadmin/condor/Arcond/etc/arcond/arcond_setup.sh`
- Copy `~chakanau/public/2010_jamb_march/MCtruthASC` to some directory
- Study the directory `user/`:
 - **Analysis_jobOptions_BASIC.py** – the standard option file for Pythia
 - **ShellScript_BASIC.sh** – has the usual command for event generation:

```
get_files PDGTABLE.MeV
get_files -scripts csc_evgen_trf.py

PhysicsScript='Analysis.py'
runno=5144
maxevents=5000
firstevent=1
# random seed. Change for each run!
rseed=$RANDOM
evgenfile=$PACKAGE_NAME'.evgen.pool.root'
histogramfile=$PACKAGE_NAME'.evgen_histo.root'
ntuplefile=$PACKAGE_NAME'.evgen_ntuple.root'

ls -la
echo `date`
echo "ArCond: Starting: csc_evgen_trf.py"
csc_evgen_trf.py $runno $firstevent $maxevents $rseed \
    $PhysicsScript $evgenfile \
    $histogramfile $ntuplefile > $PACKAGE_NAME'_gen.log' 2>&1
```

- Type “arcond”
- This time Arcond does not ask for data input (`input_data=` is empty!)
- When ready (check `arc_check`) use “./arc_cp” (local version!) to copy ROOT output files to the current directory

Tips for administrators: How to redistribute data between many computers

- **Option 1.** Use XROOTD (cannot comment, not an expert)
- **Option 2.**
 - Copy data to some central storage using `dq2-get`
 - Make a passwordless login using `ssh-agent` to all PC nodes
 - *Most of us have this already*
 - Type “`arc_setup_admin`”.
 - It will fetch several simple PYTHON scripts based on the Linux `rsync` comand
 - Specify hosts in “`host.py`” and define input and output directories (on all nodes) in the “`arc_sync`” script
 - Run this script. The script builds a file list from central storage, splits it and copy equal fractions of files on each node.
 - If
 - *data are corrupted;*
 - *a new node is added*
 - *something went wrong on a salve PC*
 - **execute the same script again. It checks file timestamp and size**
 - **For ANL claster, 100 GB can be redistributed between 3 PCs for ~1h**