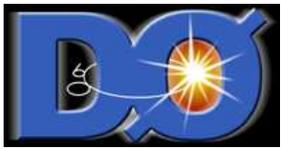


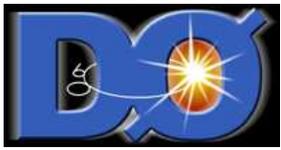
d0tools Primer

Reiner Hauser
Michigan State University
Feb 11, 2003



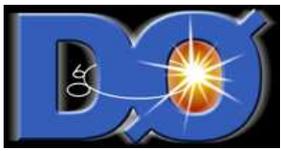
Running framework programs

- You can run any D0 framework program *by hand*
 - Just type the binary name and pass it all required parameters on the command line or via RCP
 - All output and log files will typically appear in your work area.
 - If you run more than one job, you want output separated from each other.
 - You don't want to learn too much about batch systems and SAM if you don't have to...



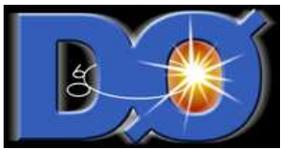
Enter *d0tools*...

- d0tools is a UPS product
 - **setup d0tools** makes it available
 - There is one generic command **rund0exe**, that works with every framework program
 - There are many wrapper scripts around it for most of the standard programs of D0:
 - runreco
 - runD0TrigSim
 - runScriptRunner
 - runchunkanalyze
 - runrecoanalyze



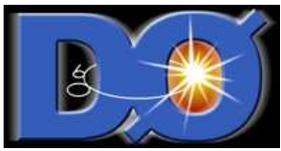
What does it do ?

- Most of these scripts provide some high-level interface that allows you to switch easily between different modes
 - e.g. running with or w/o SAM, running on MC vs. Data, running on reco vs. DST vs. TMB files
- Many of the command line options have a direct counterpart in the framework itself
 - Unfortunately, they are not always the same
 - **-num=100** is a rund0exe option to stop after 100 events
 - **-num_events 100** is the same for ReadEvent



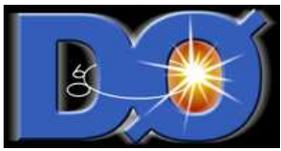
How to get started...

- Setup your working area
- **runreco -filelist=myfiles.data**
- **runreco -input=mydatafile**
 - runs reco over the files you specified
 - Output will go into a directory with a really strange name starting with *D0reco...*
- **runreco -filelist=... -name=MyTest**
 - Output will go into a directory called '*MyTest*'
- Other options:
 - **-format=pXX**
 - use this if you run e.g. over MC produced with a different version



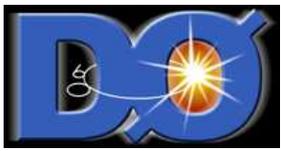
Other standard programs

- Each of the 'standard' executables has a different set of specific options.
- Use the `-h` option to see the details...
 - However, all those programs share the common options that are part of **rund0exe**
 - actually, **-filelist=...** and **-name=...** are such a common option, too...



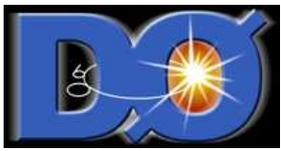
Common options

- Most of the standard programs have an executable in the release.
- You can
 - override RCP parameters with local versions
 - e.g. to change parameters in package X
 - override the framework RCP file with a local version
 - e.g. to change the framework packages which are run
 - use a local build of a prebuilt executable instead
 - e.g. if you made source code modifications to it
 - e.g. if there is no binary of your program in the release



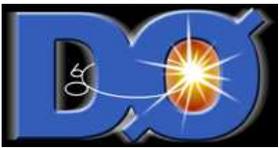
Common options (2)

- The following options specify which executable to use and which framework RCP script to use:
 - **-exe=exename** - Name of executable
 - **-rcp=rcpfile** - Name of framework rcp script
 - **-rcppkg=pkgname** - Name of framework rcp package
- Example from earlier:
 - `rund0exe -exe=tutorial_x -rcp=runMinimal.rcp \
-rcppkg=analysis_tutorial [...]`



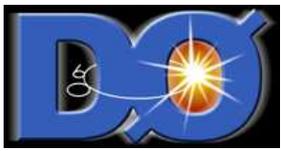
Common Options (3)

- The following options tell *d0tools* to use local versions of the binary and RCP files
 - **-localbuild** - Use local build instead of official version
 - **-localrcp** - Use local rcp's in addition to official ones
 - **-localfwkrp** - Use local version of main framework rcp
- So add to the line on the last slide:
 - **[...] -localbuild -localrcp -localfwkrp [...]**



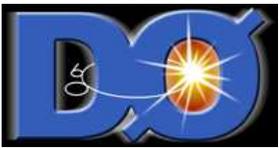
User Scripts

- If your executables needs some special setup or cleanup, you can specify your own scripts to be run before or after the program:
 - **-userscript=script** - User provided initialization script for executable
 - **-userpostscript=script** - User provided finalization script after executable - must be in \$PWD



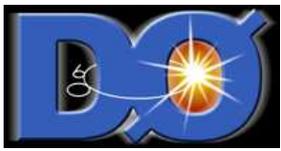
Mode options

- Usually there is both a debug and an optimized build for an executable available
 - **-maxopt** - Use optimized build (you should use this most of the time for production jobs)
- There are other options to run with purify or a profiler, but not all of them are available on clued0 and/or CAB



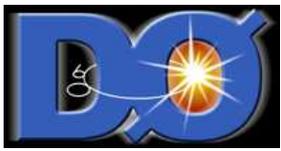
Input/Output

- To determine your input and output files:
 - **-filelist=filelist.dat** - List of files/datasets to process
 - **-input=filename** - single file
 - **-defname=defname** - SAM project definition name
 - That's all you have to do for most programs to use SAM
 - **-out=file** - Write processed events to specified file



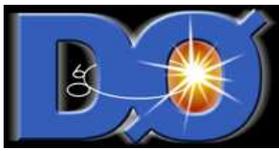
Event processing

- Skip the processing of certain events:
 - `-num=num` - Specify number of events to process
 - `-skip=num` - Specify number of events to skip
 - `-eventlist=eventlist.dat` - Process only these specific collision ids
 - `-skipruns=badruns.dat` - Skip these runs
 - `-skipevents=badevents.dat` - Skip these specific collision ids



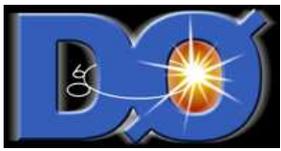
Batch options

- Apart from setting up a nice run-time environment, *d0tools* knows how to submit batch jobs for you....
 - On d0mino, clued0 and in Karlsruhe...
 - **-batch** submits your job to the batch system
 - **-q=<queue>** selects a queue (except that a SAM job usually picks the queue for you)
 - **-mem=<memory>** specifies your memory requirement
 - To run reco on your SAM dataset in the batch system:
 - **runreco -batch -defname=MySAMDefinition**



Batch Jobs on CAB

- You know that on d0mino and clued0 you can only access files that are visible via NFS (e.g. in /rooms/... or /work/machine-clued0/...)
- For CAB, d0tools does even more for you:
 - It packages up your local working directory and copies it over to CAB
 - It makes sure that your data files are either accessible from CAB or copies them over, too !
 - By default, the result comes back the machine you submitted your job from. But it can also put the results at any place on d0mino or clued0.
 - e.g. submit from your workstation, have the results on d0mino project disk



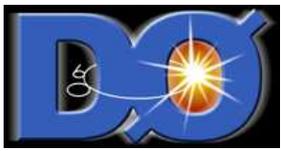
Before you do that....

- Login to d0mino:
 - setup kcroninit
 - kcroninit
- Put into your .k5login file the following two lines, replacing *rhauser* with your login ID

```
rhauser@FNAL.GOV  
rhauser/cron/d0mino.fnal.gov@FNAL.GOV
```

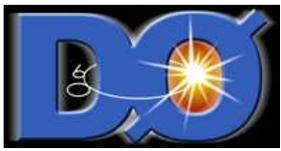
Make sure there are no additional spaces or newlines in the file !!!!!

Copy the file to your clued0 home area. You need it in both your domin0 and clued0 home directory if they are different !!!!



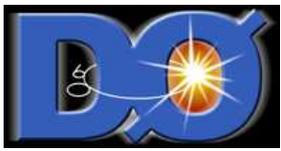
Batch Jobs on CAB (2)

- To do all this, you must provide d0tools with a scratch directory that is big enough (several hundred megabyte at least)
- Then just say you want your job executed on CAB:
 - `runreco -cab -scratch=/work/.../scratch \`
`-defname=MySAMDefinition -name=MyTest`
 - This will create a .tar file with a funny name in your scratch area. If you don't make any changes in your work area, you can reuse that .tar file:
 - `runreco -cab`
`-cabt看ar=/work/.../scratch/NNNN.tar`
`-scratch=/work/.../scratch/ -defname=...`



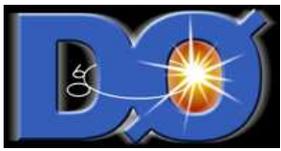
Output and Input

- If you don't want your output in your scratch area (because you're copying it somewhere else anyway), just specify the final destination:
 - `runreco -cab -cabouthost=d0mino -caboutpath=/tmp_root/729/...`
- If you're running from a file list and your data is on a d0mino project disk, you can avoid copying it over to CAB (not recommended, NFS access is not very efficient compared to a local file):
 - `runreco -cab -cabusenfs -filelist=files.dat`



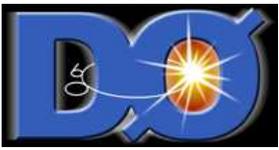
Parallel SAM Jobs

- You have a giant SAM data set ?
 - Use the `-job` option when submitting a job to CAB
 - `runreco -cab -scratch=... \`
`-defname=MySAMDef -jobs=20`
 - This will start 20 batch jobs for you, all working on the same SAM project !
 - SAM ensures that no file will be processed more than once !
 - 20 is an arbitrary limit which we put into *d0tools*
 - Even so you should be able to process all skimmed thumbnail files in a matter of hours/days without creating artificial SAM datasets.



How parallel jobs look in qstat

- For parallel jobs a number of additional scripts are started for you:
 - one called *start- \langle jobname \rangle* to start the SAM project
 - N jobs called *wkX- \langle jobname \rangle* where X goes from 0 to N
 - one called *stop- \langle jobname \rangle* to stop the SAM project.
- They are dependent on each other. If you cancel the *start* job, all the other jobs will automatically abort.
- Once the *start* and the *worker* jobs have finished (successfully or aborted), the *stop* job will run.



Status in qstat

- Parallel jobs are started in such a way to make sure that everything is properly initialized when they run.
 - All *wk** and *stop** jobs are in state H until the *start** job finishes successfully (for *hold*)
 - The *stop** job is in state H until all worker jobs are finished.
 - So don't panic if *qstat* doesn't show you a Q or R state !!
 - If *start** is in R, the others in H, you are waiting for SAM...