Suez

Chris Jones

CLEO 101

6/03/04

Outline

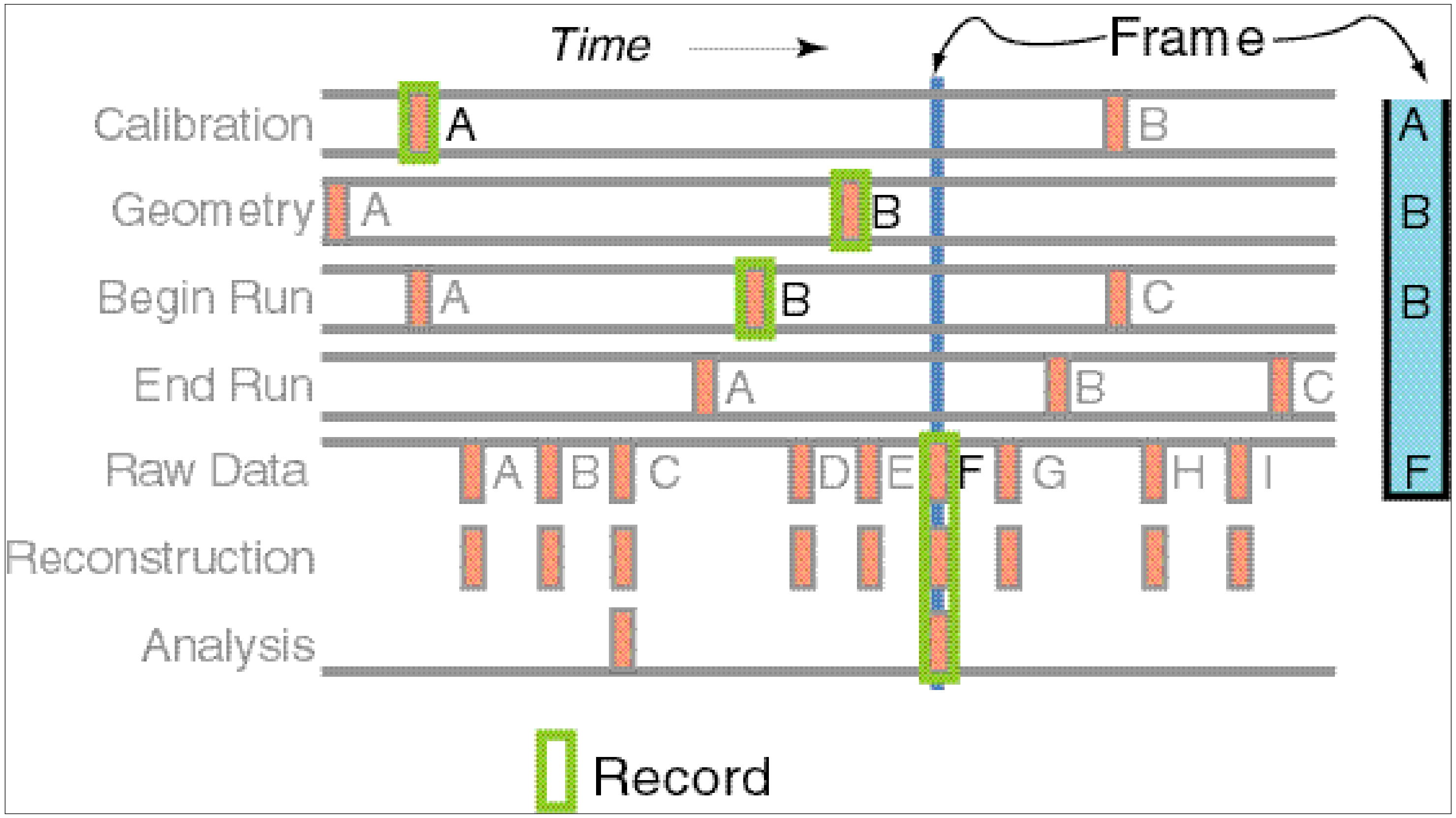
- What is Suez?
- Data Processing
- Starting Suez
- Suez Commands
- Example Jobs

What is Suez?

- Suez is the program used to process data in CLEO III/c
  - Level 3 (software trigger), Pass 1 (online quality monitoring), online event display, calibrating, Pass 2 (pattern recognition), MC generation, offline event display, analysis
- Uses a Frame to pass data
- Uses ‘plug-ins’ loaded at run time to do the work
  - Plug-ins for: reading/writing/creating/filtering data and for adding new commands
- Can read/write multiple formats at the same time
- Uses a command line interface with a full scripting language (Tcl)
- Written in C++

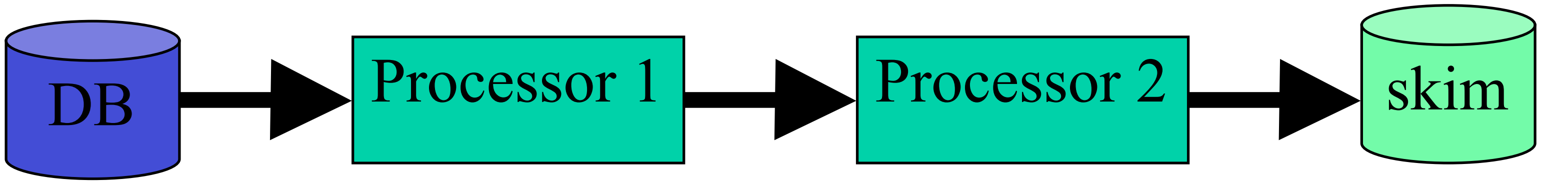
Data in Suez

- All data is accessed through the "Frame"
- Frame: A "snapshot" of CLEO at an instant in time, formed by the most recent Record in each Stream.



Processing Data in Suez

- A new Record is read in from a Source
- A new Frame is created holding all the Records pertinent to that instant in time
- The Frame is passed to Processors which decide if the Record is interesting
- If the Record is interesting to all Processors, it is written out to a Sink



Data Providers

- Data can come from two places: Source or Producer
- Source
  - Read back data from disk or tape
  - E.g. EventStore database or a ‘PDS’ file
- Producers
  - Runs an algorithm when user requests data
  - All data created in Pass2 come from Producers
  - E.g. fitted tracks come from KalmanProd

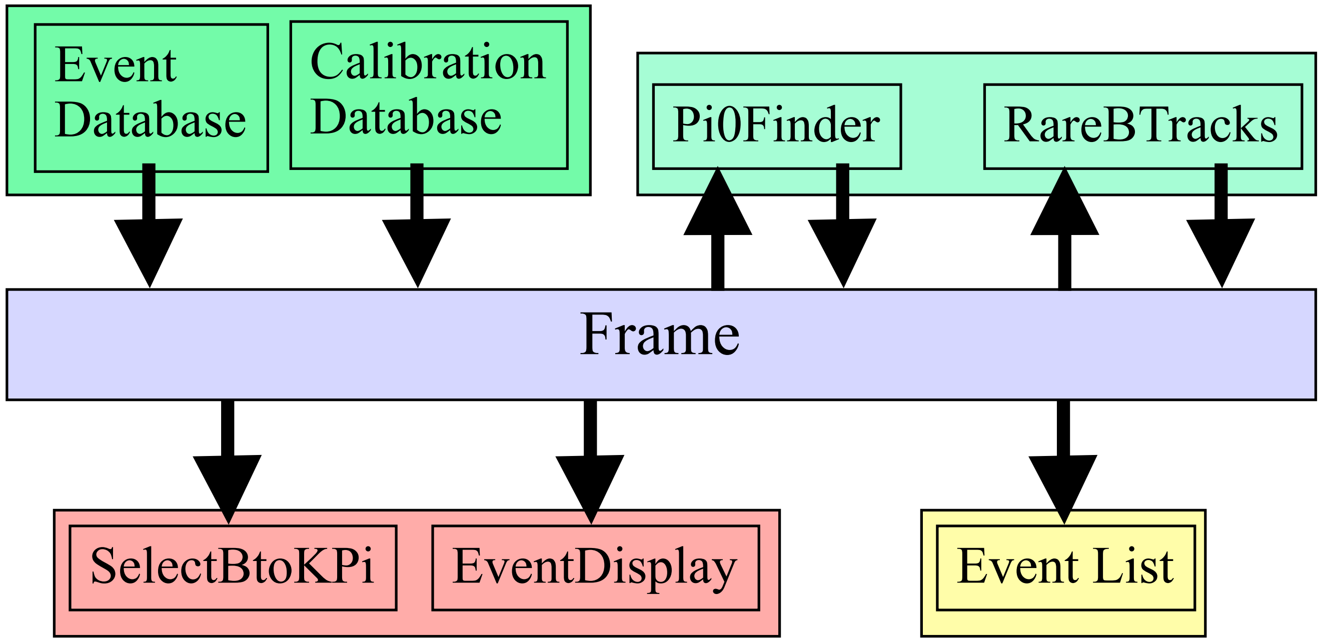
Multiple Sources and Sinks

- Can read from/write to multiple Sources/Sinks at same time
  - Read event list, user data from PDS and reconstruction from database
- Only one Source can be used to decide exactly what Records (e.g. Events) to process
  - Sources that determine what to process are called Active.
  - In above example, suez will prompt you to determine if you want to run over all the events in the event list OR in the database.

Communication via the Frame

Data Providers: data returned when requested

Sources: data from storage **Producers:** data from algorithm

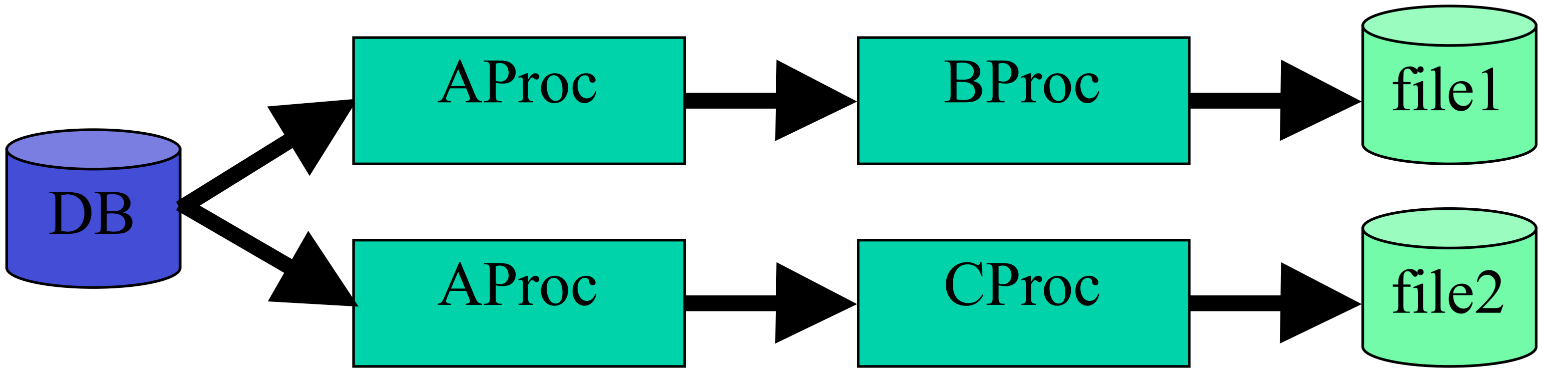


Processors: analyze and filter data **Sinks:** store data

Data Requestor: sequentially run requestors for each new Record from a source

Advanced Data Processing

- Can run several ‘jobs’ in a suez process that all share data (Sources/Producers) with different analysis and/or output
- Each analysis is a ‘path’



Advanced Data Processing

- Path has two parts
  - filter: list of Processors evaluated using boolean expression
    - AProc and BProc or not CProc
    - expression is evaluated left to right and stops running Processors once it knows if the expression will be 'true' or 'false'
      - if AProc and BProc are both true, then CProc is not run
  - operation: list of Processors or Sinks which only get the event if the filter's expression evaluates to 'true'
- The same Processor can be used in two different paths but its code will only be run once per event
- If you do not define a path, Suez will create one
  - all Processors are 'and' together in the order they were loaded
  - all defined Sinks will be added to the operation

Starting Suez

- Default plug-ins
  - `suez`
- No plug-ins
  - `suez -q`
- Your command file
  - `suez -f <filename>`
- See the choices
  - `suez -h`

Command line

- Full 'shell' like editing
  - Tab completion of command name or file name
  - Use left and right arrows to move cursor on the line
  - Use standard shell command keys
    - <control> k: delete to end of line
    - <control> a: go to begin of line
    - <control> e: go to end of line
    - etc
- Full history
  - Use up and down arrows to scroll through command history
  - `history` : returns list of all commands ever typed
  - !<partial command> : runs the last command that matches
  - etc.

Suez Commands: Help

- `Suez> help`
  - Lists all the commands that are available
- `Suez> <command> help`
  - Gives specific information about that command

Suez Command:Event Store

- **Think of it as having one large file holding all data**
- `module sel EventStoreModule`
  - Loads the eventstore commands
- `eventstore in <date> [<grade>] [<skim>] [runs <start> [<end>]]`
  - Creates a source that reads data from EventStore
  - `eventstore in 20040603`
  - <date> YYYYMMDD specify version of data by a date
    - choose start date of analysis
  - <grade> choose processing ‘step’ of code
    - physics is default
  - <skim> choose events based on physics type
    - all (default), qcd, 2photon, bhagam
  - runs Specify what run to process, if do not specify run through all data. If only <start> just process that run.

Suez Command: Reading File

- **`source_format sel <format>`**
  - Load into suez the plug-in for that format
  - `source_format sel PDSSourceFormat`
- **`file in <filename>.<extension> [<stream> [<stream>]]`**
  - Reads those streams from the file
    - If not streams given, use default for that file
  - Uses the file extension to determine what format to use
  - `file in test.pds`

Suez Command: Writing File

- `sink_format sel <format>`
  - Load into suez the plug-in for that format
  - sink_format sel PDSSinkFormat
- `file out <filename>.<extension> <stream> [<stream>]`
  - Write those streams to the file
  - Uses the file extension to determine what format to use
  - `file out test.pds beginrun startrun event`

Suez Command: Sources and Sinks

- `source ls`
  - Lists sources loaded into Suez
- `source status`
  - Lists the status (e.g. OK or something went wrong)
- `sink ls`
  - Lists sinks loaded into Suez
- `source activate <source name> <stream> [<stream>]`
  - Make this source the Active source for the list of streams
  - `source activate test beginrun startrun event`
- `default prompt <off>|<on>`
  - Tells suez to “not prompt” or “prompt” users for input
  - With default prompt off, it will automatically choose the active sources
  - Useful for batch scripts

Suez Command: Processor

- `proc sel <processor> [production <tag>]`
  - Loads the Processor into Suez
  - Processors are run in loading order (can be reordered)
  - `proc sel NTrackFilterProc`
  - Can load the same Processor multiple times
  - `proc sel NTrackFilterProc production 8tracks`
- `proc lss`
  - Lists all processors that have been loaded
- **param <processor> <command>**
  - Run a command of the Processor
  - `param NTrackFilterProc help`
  - `param NTrackFilterProc MinNumberOfTracks`
  - `param NTrackFilterProc MinNumberOfTracks 5`

Suez Command: Producer

- **`prod sel <producer> [production <tag>]`**
  - Loads the Producer into Suez
  - Loading order does not matter for Producers
  - `prod sel TrackDeliveryProd`
  - Can load the same Producer multiple times by specifying a production tag. (Use the tag when getting data)
  - `prod sel TrackDeliveryProd production myTracks`
- **`prod lss`**
  - Lists all processors that have been loaded
- **`param <producer> <command>`**
  - Run a command of the Producer

Suez Command: Looping

- **`go [<number of stops>  [<streams to count>] ]`**
  - `go`
    - Processes all Records in the source
  - `go 10`
    - Process 10 event
  - `go 10 endrun`
    - Keeps processing until it sees 10 endruns
- **`goto <run number> [<event number>]`**
  - Jumps to that particular run or event

Suez Command: Tcl

- `run_file <filename>`
  - Loads a series of scripting commands
  - `run_file test.tcl`
  - `run_file $env(C3_SCRIPTS)/runOnPass2.tcl`
- Tcl commands
  - `set <variable name> <value>`
    - Creates a new variable and sets it's "value"
    - Use $<variable name> to access the variable's value
  - `exec <shell command>`
    - Runs a shell command
    - `exec ls`
    - `exec echo [proc ls] | grep Sp`
  - etc

Suez Command: Ending Job

- `exit`
- `quit`
  - Both will cause Suez to end
- <control> c
  - During looping, this will bring up a prompt asking you if you want to stop the loop or end the Suez job.

Suez Command: Path

- path create <name> [<filter spec>] >> [<proc>] [<sink>]
  - create path with name <name> and filter defined by <filter spec> with operation defined in <proc> <sink>
  - <filter spec> specifies the definition of the filter to use
    - [not] <proc>| <filter> [and|or|xor [not] <proc>|<filter>]
    - <proc> name of a Processor
    - <filter> name of a defined Filter
  - path create Good AProc and BProc >> good.pds
- path ls
  - list all created paths
- path filter create <name> [<filter spec>]
  - create a filter with name <name> defined by <filter spec>
- path filter ls
  - list all created filters

Simple Example Job

```
#setup to read file
source_format sel AsciiSourceFormat
file in /nfs/cleo3/Offline/data/runBeginRunEvent.asc
#print out each event number in file
proc select RunEventNumberProc
#process file
go
```

More Advanced Example

```
#read from the event store only qcd related events
module sel EventStoreModule
eventstore in 20040603 qcd

#load standard stuff for analysis
run_file $env(C3_SCRIPTS)/runOnPass2.tcl
#filter out all events that have less than 5 tracks
proc select NTrackFilterProc
param NTrackFilterProc MinNumberOfTracks 5
#printout the event numbers
proc sel RunEventNumberProc
#look at only the first run
go 2 beginrun
```