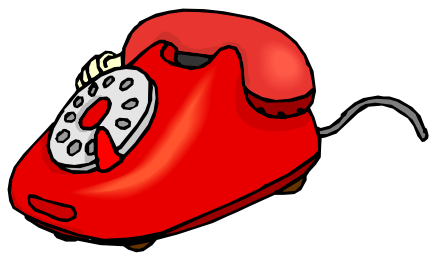
DIAL
Distributed Interactive Analysis of Large datasets

BNL Technology Meeting

David Adams
BNL
May 19, 2003



Contents

Goals of DIAL

What is DIAL?

Design

- Applications
- Results and Tasks
- Schedulers
- Datasets
- Exchange format

Status

Development plans

GRID requirements

Goals of DIAL

1. Demonstrate the feasibility of interactive analysis of large datasets
   - How much data can we analyze interactively?
2. Set requirements for GRID services
   - Datasets, schedulers, jobs, results, resource discovery, authentication, allocation, ...
3. Provide ATLAS with a useful analysis tool
   - For current and upcoming data challenges
   - Real world deliverable
   - Another experiment would show generality

What is DIAL?

Distributed

- Data and processing

Interactive

- Iterative processing with prompt response
  - (seconds rather than hours)

Analysis of

- Fill histograms, select events, …

Large datasets

- Any event data (not just ntuples or tag)

What is DIAL? (cont)

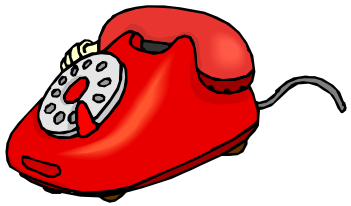
DIAL provides a connection between

- Interactive analysis framework
  - Fitting, presentation graphics, …
  - E.g. ROOT
- and Data processing application
  - Natural to the data of interest
  - E.g. athena for ATLAS

DIAL distributes processing

- Among sites, farms, nodes
- To provide user with desired response time

Interactive analysis
e.g. ROOT, JAS, ...



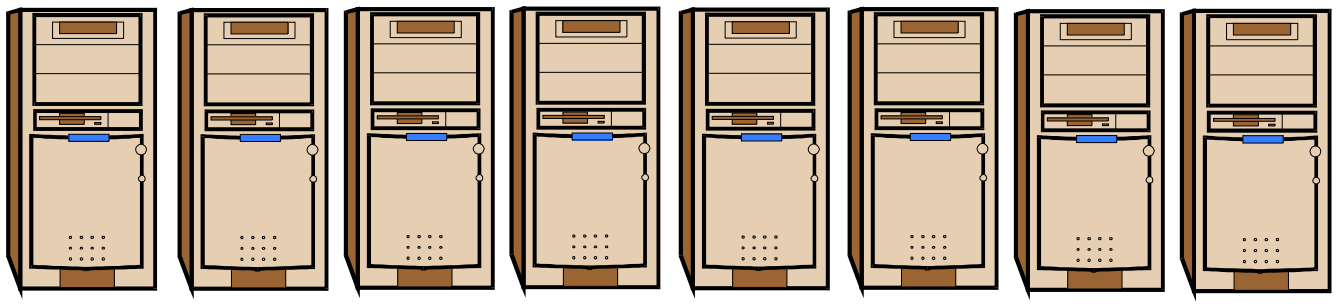
DIAL

Dataset

Job

Scheduler

AAA

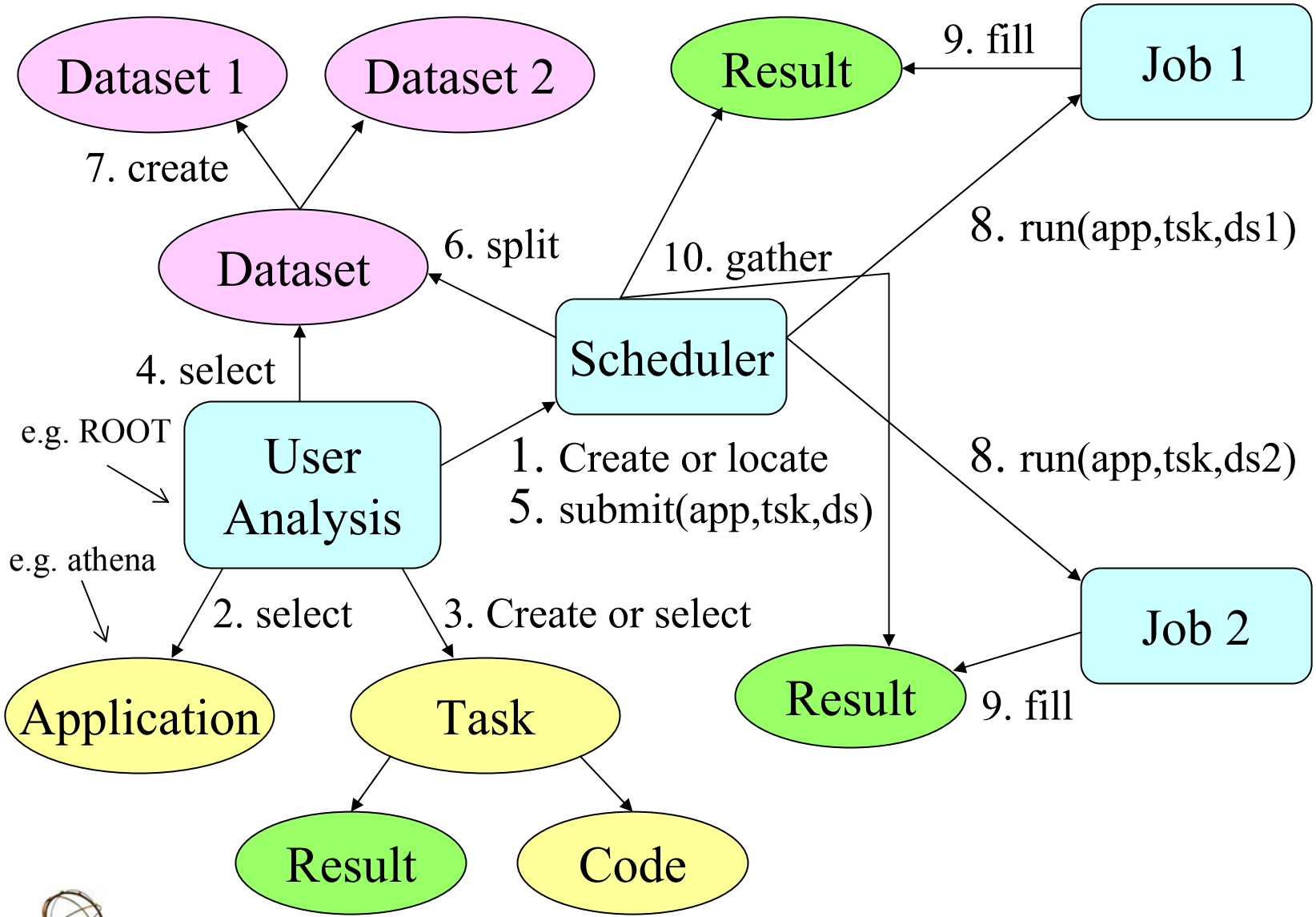


Distributed processing running data-specific application

Design

DIAL has the following components

- Dataset describing the data of interest
  - Organized into events
- Application
  - Event loop providing access to the data
- Result
- Task
  - Empty result plus code process each event
- Scheduler
  - Distributes processing and combines results


Dataset 1
Dataset 2
Result
9. fill
Job 1
7. create
Dataset
6. split
10. gather
8. run(app,tsk,ds1)
Scheduler
4. select
e.g. ROOT
User Analysis
1. Create or locate
5. submit(app,tsk,ds)
8. run(app,tsk,ds2)
e.g. athena
2. select
3. Create or select
Job 2
Application
Task
Result
9. fill
Result
Code

Applications

Current application specification is

- Name
  - E.g. athena
- Version
  - E.g. 6.10.01
- List of shared libraries
  - E.g. libRawData, libInnerDetectorReco

Applications (cont)

Each DIAL compute node provides an application description database

- File-based
  - Location specified by environmental variable
- Indexed by application name and version
- Application description includes
  - Location of executable
  - Run time environment (shared lib path, …)
  - Command to build shared library from task code
- Defined by ChildScheduler
  - Different scheduler could change conventions

Results and Tasks

Result is filled during processing

- Examples
  - Histogram
  - Event list
  - File

Task provided by user

- Empty result plus
- Code to fill the result
  - Language and interface depend on application
  - May need to be compiled

Schedulers

A DIAL scheduler provides means to

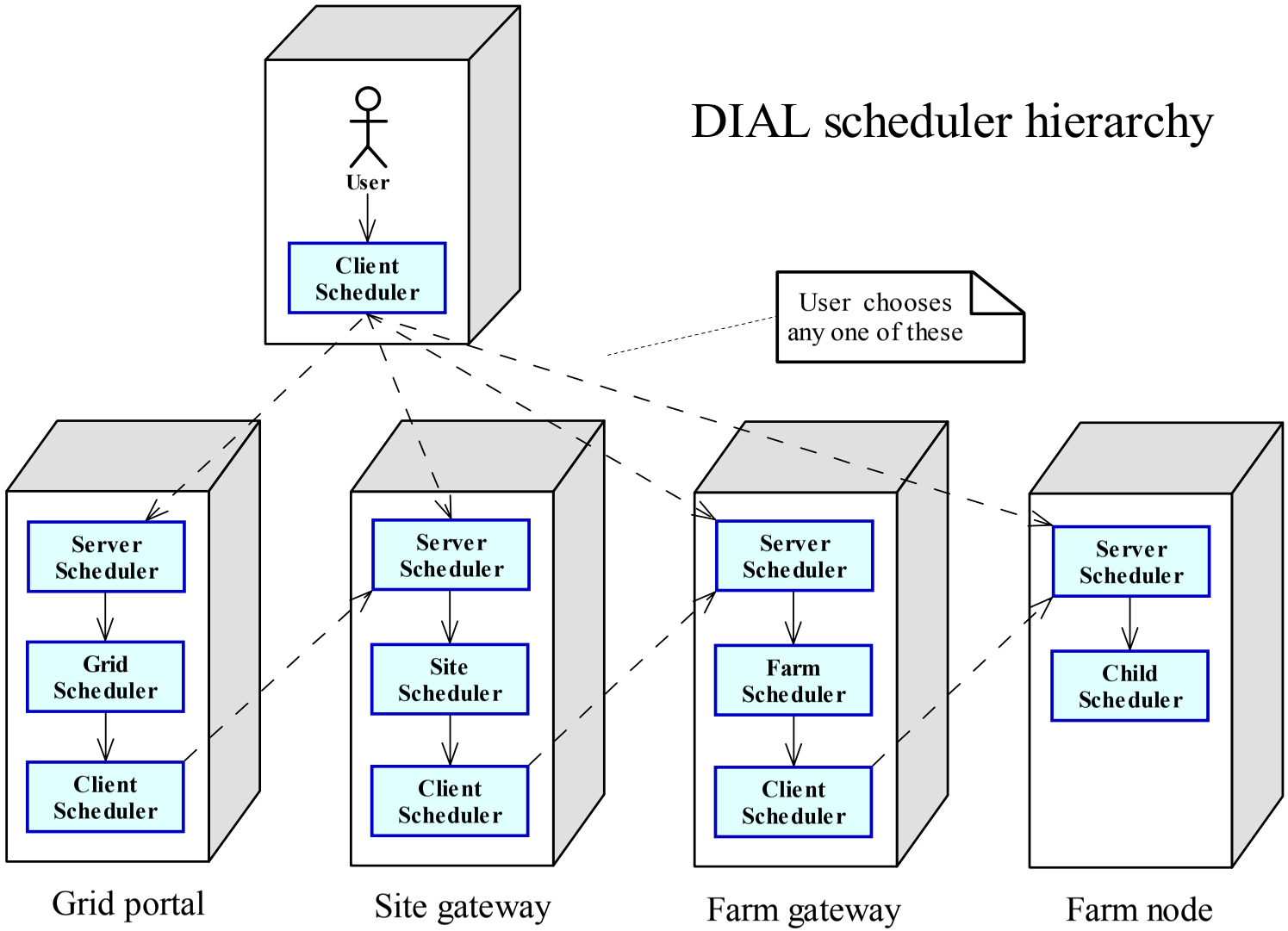
- Submit a job
- Terminate a job
- Monitor a job
  - Status
  - Events processed
  - Partial results
- Verify availability of an application
- Install and verify the presence of a task for a given application

Schedulers (cont)

Schedulers form a hierarchy

- Corresponding to that of compute nodes
  - Grid, site, farm, node
- Each scheduler splits job into sub-jobs and distributes these over lower-level schedulers
- Lowest level ChildScheduler starts processes to carry out the sub-jobs
- Scheduler concatenates results for its sub-jobs
- User may enter the hierarchy at any level
- Client-server communication

DIAL scheduler hierarchy

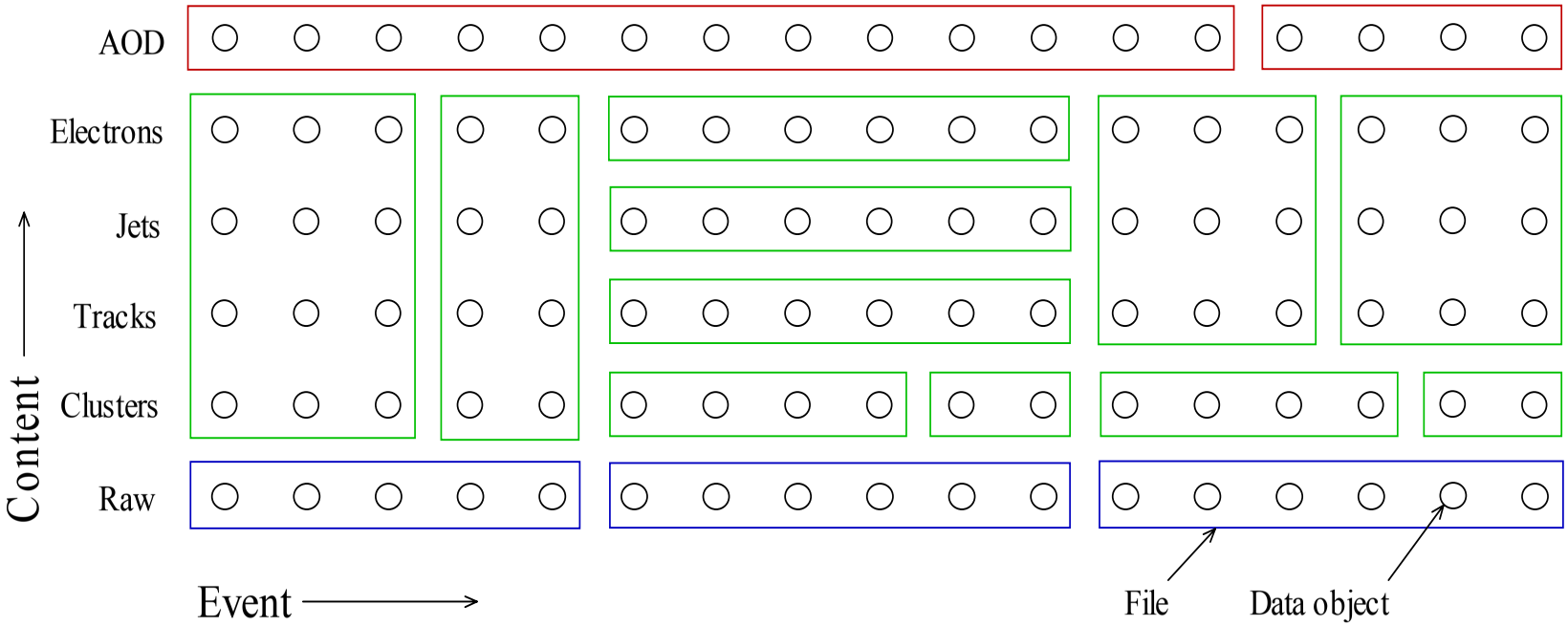


Datasets

Datasets specify event data to be processed

Datasets provide the following

- List of event identifiers
- Content
  - E.g. raw data, refit tracks, cone=0.3 jets, …
- Means to locate the data
  - List of of logical files where data can be found
  - Mapping from event ID and content to a file and a the location in that file where the data may be found
  - Example follows

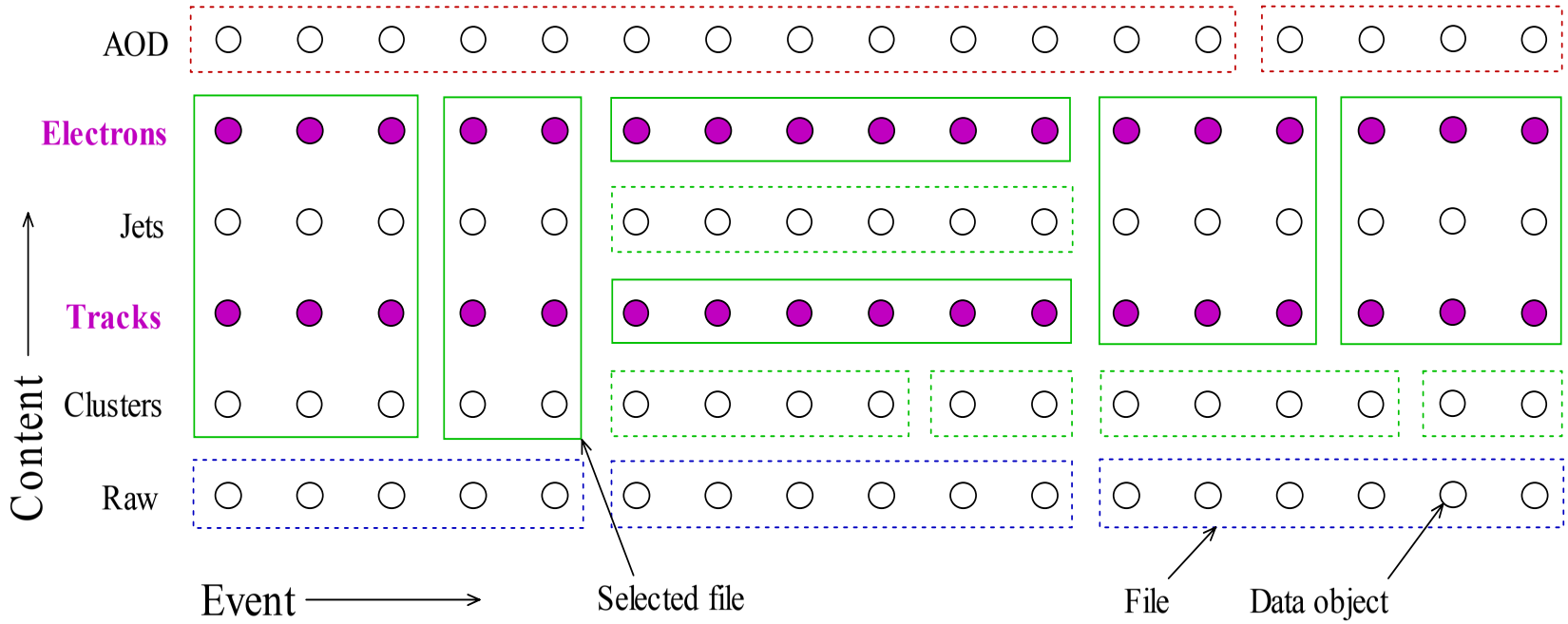


Example dataset with mapping to files

Datasets (cont)

User may specify content of interest

- Dataset plus this content restriction is another dataset
- Event data for the new dataset located in a subset of the files required for the original
- Only this subset required for processing

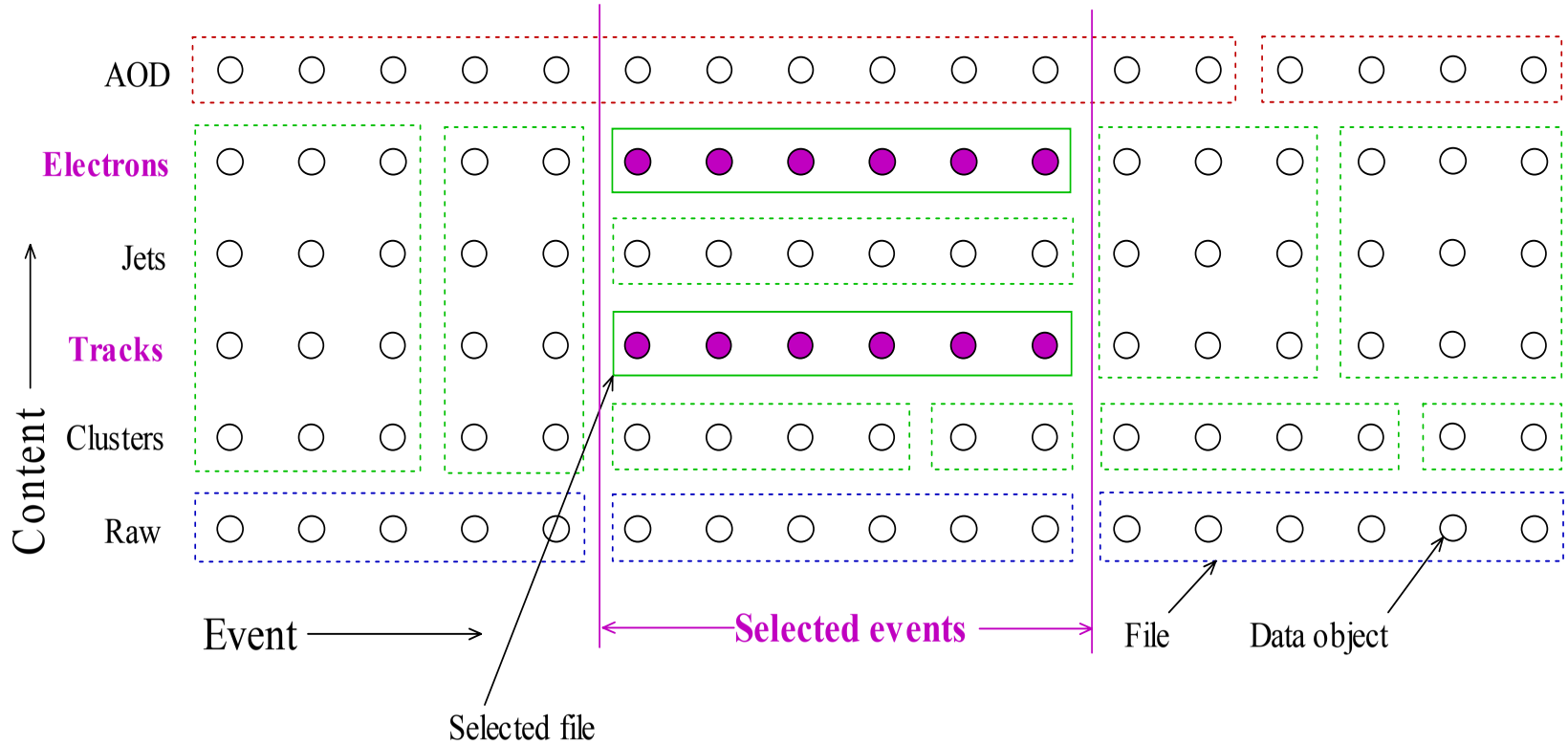


Example dataset with content selection

Datasets (cont)

Distributed analysis requires means to divide a dataset into sub-datasets

- Sub-dataset is a dataset
- Do not split data from any one event
- Split along file boundaries
  - Jobs can be assigned where files are already present
  - Split most likely done at grid level
- May assign different events from one file to different jobs to speed processing
  - Split likely done at farm level



Example sub-dataset with content selection

Exchange format

DIAL components are exchanged

- Between
  - User and scheduler
  - Scheduler and scheduler
  - Scheduler and application executable
- Components have an XML representation
- Exchange mechanism can be
  - C++ objects
  - SOAP
  - Files
- Mechanism defined by scheduler

Status

All DIAL components in place

- http://www.usatlas.bnl.gov/~dladams/dial
- Release 0.20 made in March
- But scheduler is very simple
  - Only local ChildScheduler is implemented
  - Grid, site, farm and client-server schedulers not yet implemented

More details in CHEP paper at

- http://www.usatlas.bnl.gov/~dladams/dial/talks/dial_chep2003.pdf

Status (cont)

Dataset implemented as a separate system

- http://www.usatlas.bnl.gov/~dladams/dataset
- Implementations:
  - ATLAS AthenaRoot file
    - > Exists
    - > Holds Monte Carlo generator information
  - ATLAS combined ntuple hbook file
    - > Under development
  - Athena-Pool files
    - > when they are available

Status (cont)

DIAL and dataset classes imported to ROOT

- ROOT can be used as interactive user interface
  - All DIAL and dataset classes and methods available at command prompt
  - DIAL and dataset libraries must be loaded
- Import done with ACLiC
- Only preliminary testing done
- Need to add result for TH1 and any other classes of interest

Status (cont)

No application integrated to process jobs

- Except test program dialproc counts events
- Plans for ATLAS:
  - Dialpaw to run paw to process combined ntuple
    - > Under development
  - Athena to process Athena-Pool event data files
    - > When Athena-Pool is available later this year
  - Perhaps a ROOT backend to process ntuples
    - > Or is this better handled with PROOF?
    - > Or use PROOF to implement a farm scheduler?

Status (cont)

Interface for logical files was added recently

- Includes abstract interface for a file catalog
  - Local directory has been implemented
  - Plan to add AFS catalog
    - > Good enough for immediate ATLAS needs
  - Eventually add Magda and/or RLS

Status (cont)

Results

- Existing implementations
  - Counter
  - Event ID list
- Under development
  - HBOOK (logical) file
    - > Holding histograms
- Planned
  - Collection of other results
  - ROOT objects (TH1, …)
  - AIDA instead of ROOT?

Development plans

Highlighted items in

- red required for useful ATLAS tool and
- green to use it to analyze Athena-Pool data

Schedulers

- Client-server schedulers
- Farm scheduler
  - Allows large-scale test
- Site and grid schedulers
  - GRID integration
  - Interact with dataset, file and replica catalogs
  - Authentication and authorization

Development plans (cont)

Datasets

- Hbook combined ntuple
  - in development
- Interface to ATLAS POOL event collections
  - expected in summer
- ROOT ntuples ??

Applications

- PAW (with C++ wrapper)
- Athena for ATLAS
- ROOT ??

Development plans (cont)

Analysis environment

- ROOT implementation needs testing
- JAS?
  - Java based
  - Add DIAL java binding?
- One or more of the LHC Python busses?
  - SEAL, Athena-ASK, GANGA, …
  - These can use ROOT via PyRoot
  - Import DIAL into Python
    - > With SWIG or boost

GRID requirements

A major goal of DIAL is to identify components and services to share with

- Other distributed interactive analysis projects
  - PROOF, JAS, …
- Distributed batch projects
  - Production
  - Analysis
- Non-HEP event-oriented problems
  - Data organized into a collection of "events" that are each processed in the same way

GRID requirements (cont)

Candidates for shared components include

- Dataset
  - Event ID list
  - Content
  - File mapping
  - Splitting
- Application
  - Specification
  - Installation
- Task
  - Transformation = Application + Task

GRID requirements (cont)

Shared components (cont)

- Job
  - Specification (application, task, dataset)
  - Response time
  - Hierarchy (split into sub-jobs)
    - > DAG?
- Scheduler
  - Accepts job submission
  - Splits, submits and monitors sub-jobs
  - Gathers and concatenates results
  - Returns status including results and partial results

GRID requirements (cont)

Shared components (cont)

- Logical files
  - Catalogs
  - Replication
- Authentication and authorization
- Resource location and allocation
  - Data, processing and matching

GRID requirements (cont)

Important aspect is ***latency***

- Interactive system provides means for user to specify maximum acceptable response time
- All actions must take place within this time
  - Locate data and resources
  - Splitting and matchmaking
  - Job submission
  - Gathering of results
- Longer latency for first pass over a dataset
  - Record state for later passes
  - Still must be able to adjust to changing conditions

GRID requirements (cont)

Interactive and batch must share resources

- Share implies more available resources for both
- Interactive use varies significantly
  - Time of day
  - Time to the next conference
  - Discovery of interesting events
- Interactive request must be able to preempt long-running batch jobs
  - But allocation determined by sites, experiments, …