Workload Management: catalogs & co.

Proposal for an CMS catalog architecture and more

Stefano Lacaprara, Nikolai Smirnov

Department of Physics
INFN and University of Padova

DM-WM joint meeting, CERN, 8-9-10 february 2005

Outline

Outline

Outline

Outline

Outline

Motivation

- Starting to really access data in a distributed way
- First important lessons learned
- Should try to evolve/redesign overall architecture
- Data discovery and access is the most critical problem
- not the only one . . .
- Actual prototype works but can be improved and integrated with Data Management activity

Outline

What is a Data Atom

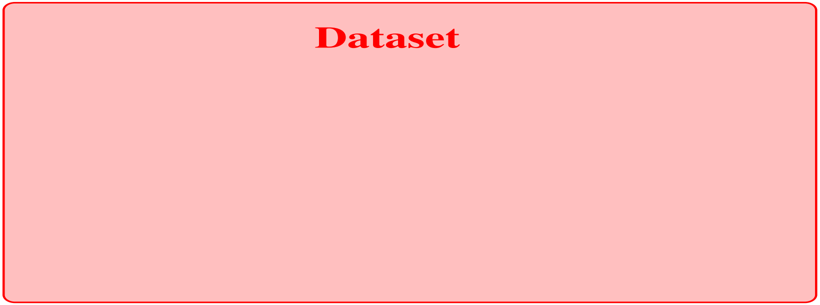
Atom

- Unbreakable unit of data, fully self–consistent
- Must have an unique identifier (key)
- Can be accessed without the need of other data
- Can be different for Data Management and Workload Management!
- From user point of view, the smaller the better: user may want to access a very well defined chunk of data (*e.g.* DT digis for event/run)
- From WM point of view, very small atoms would lead to scalability problems
- Need to compromise

WM vs DM

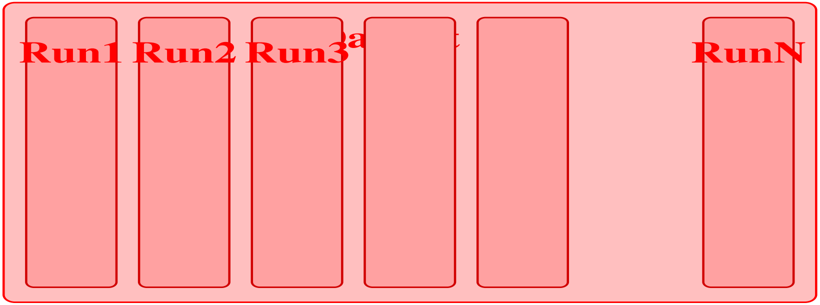
- For DM atom is a file (according to DMRTAG)
- For WM atom is typically a files collection
- For WM atom is an abstract concept, even though it does correspond to a given, fixed and finite set of physical files
- WM does not need to know the files of an atom, DM does
- The analysis Atom is the smaller, unbreakable chunk of data which must be accessed as a whole
- The Data Atom is the smaller, unbreakable chunk of data which is subject to data movement

Data Atom today MetaData



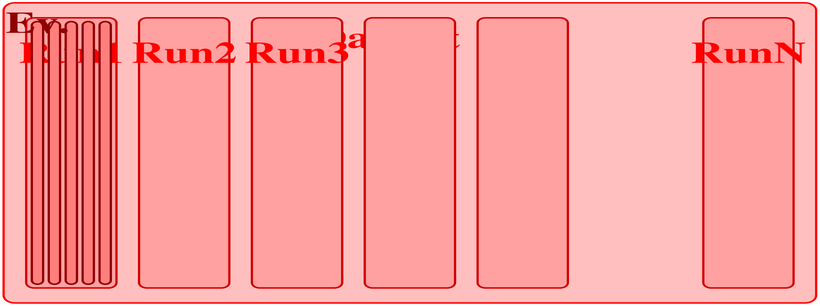
- Today a *de facto* atom is a whole Dataset
- For sure too big!

Data Atom today MetaData



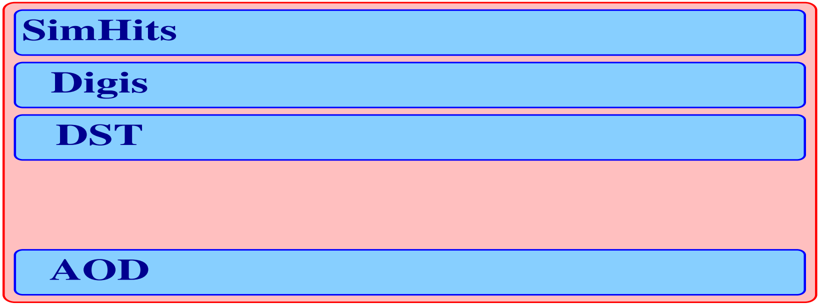
- Better could be a run: a Dataset is made of *N* runs
- Smaller, but not too much

Data Atom today MetaData



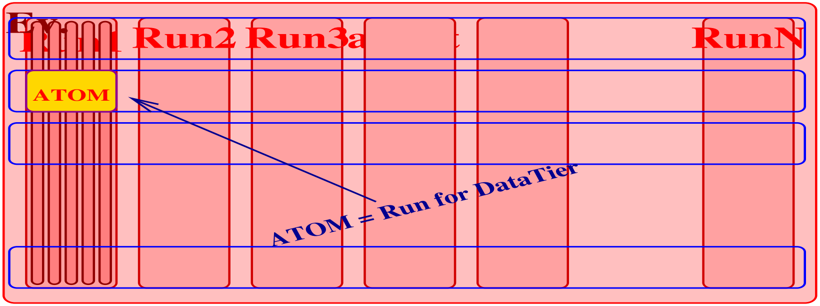
- A run is made by events
- Probably too small: scalability problems

Data Atom today MetaData



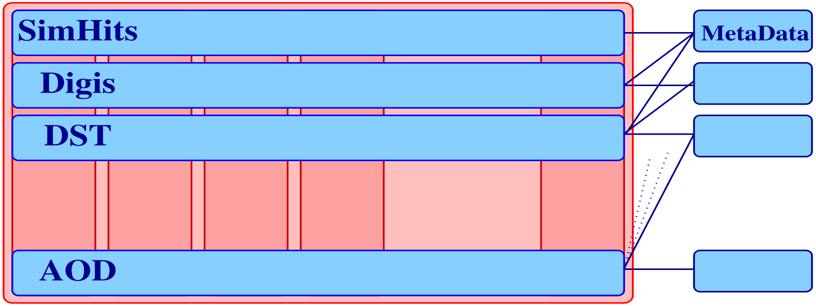
- Not the full story also horizontal division
- Data Tiers

Data Atom today MetaData



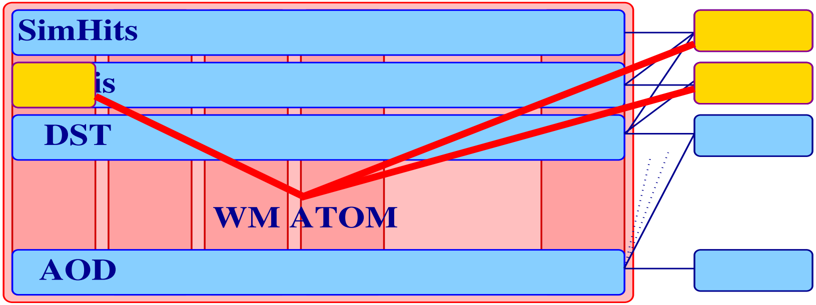
- Good candidate is Run of a given DataTier

Data Atom today MetaData



- Not the full story: to access Data (*e.g.* a Run) need also MetaData

Data Atom today MetaData



- WM Atom must be self-consistent
- Must include also needed COBRA MetaData

Atoms collection

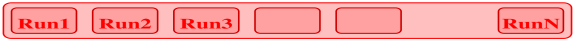
- WM Atom can be a complex object
- However, it does correspond to a finite set of physical files
- WM Atom has a 1 $\longrightarrow N$ wrt DM Atom (files)
- Want to define a kind of hierarchy of atoms
- If a Atom is a Run, what is a Dataset?
- Very important point, since user want to access data with different granularity
- Dataset is **not** and Atom (can be break!)
- Dataset is a collection of Atoms: Molecule, Crystal, Metal??

Atoms collection

- WM Atom can be a complex object
- However, it does correspond to a finite set of physical files
- WM Atom has a 1 $\longrightarrow N$ wrt DM Atom (files)
- Want to define a kind of hierarchy of atoms
- If a Atom is a Run, what is a Dataset?
- Very important point, since user want to access data with different granularity
- Dataset is **not** and Atom (can be break!)
- Dataset is a collection of Atoms: Molecule, Crystal, Metal??

Atoms collection

- WM Atom can be a complex object
- However, it does correspond to a finite set of physical files
- WM Atom has a 1 $\longrightarrow N$ wrt DM Atom (files)
- Want to define a kind of hierarchy of atoms
- If a Atom is a Run, what is a Dataset?
- Very important point, since user want to access data with different granularity
- Dataset is **not** and Atom (can be break!)
- Dataset is a collection of Atoms: Molecule, Crystal, Metal??



Outline

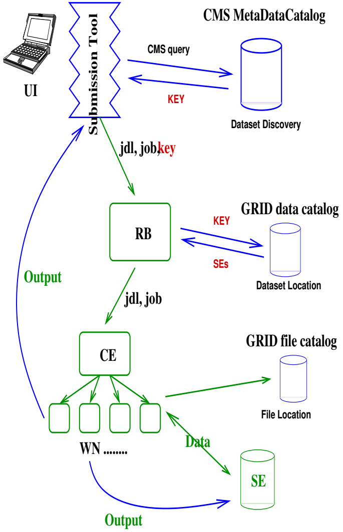
What do we need for WM

WM needs

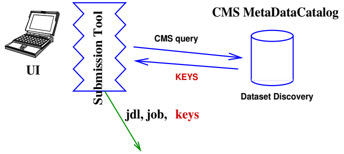
1. We need to know Data exist
2. We need to locate Data (Data Discovery)
3. We need to access Data from remote resource (Data Access)

- What is "Data"
- Even if Data is physically stored on files, we don't need to know that for 1 and 2
- only for 3 files matter
- Key element is WM Atom, as defined above: *abstract* object at level 1 and 2, which does *materialize* into a list of physical files at level 3

Catalogs: proposed architecture



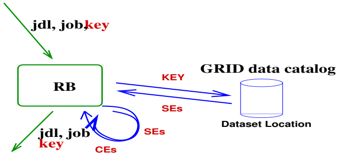
- Three level of catalogs, with defined responsibility and scope
  1. CMS specific MetaData Catalog
     user access point to next step: CMS responsibility, Centralized (replicated)
  2. Grid Data Location Catalog
     should be Grid responsibility, **not** CMS, Global (replicated/distributed)
  3. Grid Local File Catalog
     Grid responsibility, Local



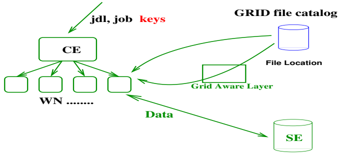
- CMS Meta Data Catalog

- User (or user oriented tool CRAB) access point to Data Discovery and access
- Input user Query: any type, possibly Google like
- Must know about all available data, together with all Data attributes (sw version, calibration, detector condition, processing cards, etc. . .)
- Does not know about data location
- Return list of keys corresponding to WM Atoms or Crystal
- Key list will be passed to next catalog

Proposed architecture



- Grid Data Location Catalog
- Grid responsibility
- Accessed at Resource Broker level: Global
- Input is list of keys, corresponding to Atoms or Crystals
- `inputData = 'key1','key2',...`
- Output is list of Storage Elements hosting Atoms
- Not direct files knowledge is needed
- Data Discovery is done using Atoms and collection of Atoms
- RB finds CEs fine for SEs and choose CE
- keys are sent further down



- Grid Local File Catalog

- Grid responsibility
- Accessed at CE/WN level: Local
- Input is again list of keys, Atoms or Crystals
- Output is list of physical files corresponding to required Atoms
- Directly used by COBRA, if POOL file catalog
- Transformed into POOL format by Grid aware layer (CRAB job wrapper)
- If one POOL catalog (mysql) per site, do not need to extract Atom fragment: COBRA application uses only what is needed

Outline

Job Splitting

Splitting

- Proposed architecture allow for data discovery and data access
- Next issue is about job splitting
- Does this architecture allows smart job splitting as well?

Job Splitting Scenarios

1. Splitting done at User Level
2. Splitting done at RB Level
3. Mixed case

Job Splitting Specification

- Set of Jobs with same requirements
- Seen as a single Job Cluster
- Allow bulk operation
  - submission,
  - query,
  - status,
  - cancel,
  - . . .
- Also possible to get access to single sub jobs
- SubJob number available at WN level, to be used by job wrapper
- Perform just one authentication handshaking
- Splitting possible with 3 previous use cases (see after)

Job Splitting At UI Level

- Pretty much what we do today
- The splitting is done according to user specification and not to data distribution
- Data distribution information are **not** available at UI level
- Job splitting does not need to follow data structure
- *e.g.* Access a whole dataset in bunches of 1500 events even if a each run (atom) has 1000 events
- In any case, key hierarchy is crucial
  1. User pass a Crystal key (Dataset key DS_{key}) as InputData
  2. RB match DS_{key} with Grid Data Catalog
  3. At WN Job access files collection corresponding to whole DS_{key}

Job Splitting At RB Level

- Only RB knows about resources available **and** Data location
- Not possible today, even if present in long term LCG/EGEE plan
- It is possible with the proposed architecture

- User define some *restriction* for the splitting, like max number of jobs
- User pass a Crystal key (Dataset key DS_{key}) as InputData
- RB match DS_{key} with Grid Data Catalog
- $DS == \{Atom_1, Atom_2, ..., Atom_n\}$
- $DS_{key} == \{Atom_{key_1}, Atom_{key_2}, ..., Atom_{key_n}\}$
- RB splits according to input Data

Splitting at RB level

- $DS_{key} == \{\{key_1, ..., key_{1_N}\}, \{...\}, \{key_{N_1}, ..., key_{N_N}\}\}$
- So InputData for each SubJob is $\{key_1, ..., key_{1_N}\}$
- RB matches Data Location against SubJob InputData
- At WN, SubInputData is used to get access to files collection
- Problem: how the application (COBRA) uses the SubInputData information as input?
- Namely, how can job wrapper (CRAB) know that $key == key_{N_1}$ must be translated into

```
InputCollections=/System/Owner/Dataset/Evd_RunN
```

Splitting at RB level

- A possibility is that the key_N is just `RunN`
- Very strict requirement!
- In general a `key` is generic
- Who knows the meaning of a `key` in term of CMS Data collection?
- Only the CMS MetaDataCatalog
- **But** the scope of MDC is global, while this information must be available at WN level, that is in a **local** scope
- At WN level, the only info is that a given `key` does correspond to a given files collection
- COBRA input is not a (collection of) files

Splitting at RB level

- Possible solution
- Since the information is available in global scope, collect the needed info at UI level
- Output of query to CMS MDC is not *just* DS_{key} or $DS_{key} == \{Atom_{key_1}, Atom_{key_2}, \dots, Atom_{key_n}\}$
- But also contains the info about correspondence $key \leftrightarrow run$

$$\begin{pmatrix} DS_{key} \\ DS_{name} \end{pmatrix} = \left\{ \begin{pmatrix} Atom_{key_1} \\ Run_1 \end{pmatrix}, \dots, \begin{pmatrix} Atom_{key_n} \\ Run_n \end{pmatrix} \right\}$$

- This information should go from UI to WN to be used by job wrapper (CRAB)
- RB does the splitting and tell to SubJob which `atoms` should access
- CRAB translate `atom` into `COBRA` language

Splitting at RB level

- Important issue
- A Dataset is splitted into *vertical* slices (Runs) and *horizontal* slices (DataTiers)
- An Atom can be the intersection of horizontal and vertical slice (Run with given DataTier)
- Splitting is done at RB according to `Crystal`↔`Atoms` correspondence
- Must be done only for vertical slices!
- This is CMS specific, known at MDC level
- If RB splits according to vector passed by MDC ⇒ ok

Job Splitting At Mixed Level

- The proposed architecture allows for more complex splitting scenarios
- Splitting at UI level plus additional sub-splitting at RB level
- Even more fancy scenarios (if we are interested in, probably not so much. . .)

Summary

- Abstract, unbreakable Atom, Crystal, etc is central.
- Three level catalogs architecture with clear responsibilities and scope
- Job splitting scenarios considered and solution proposed

- Outlook
  - Some components is Grid responsibility
  - Full support from LCG/EGEE have been granted, provided we give clear architecture and components requirements
  - Migration plan from current implementation not yet defined