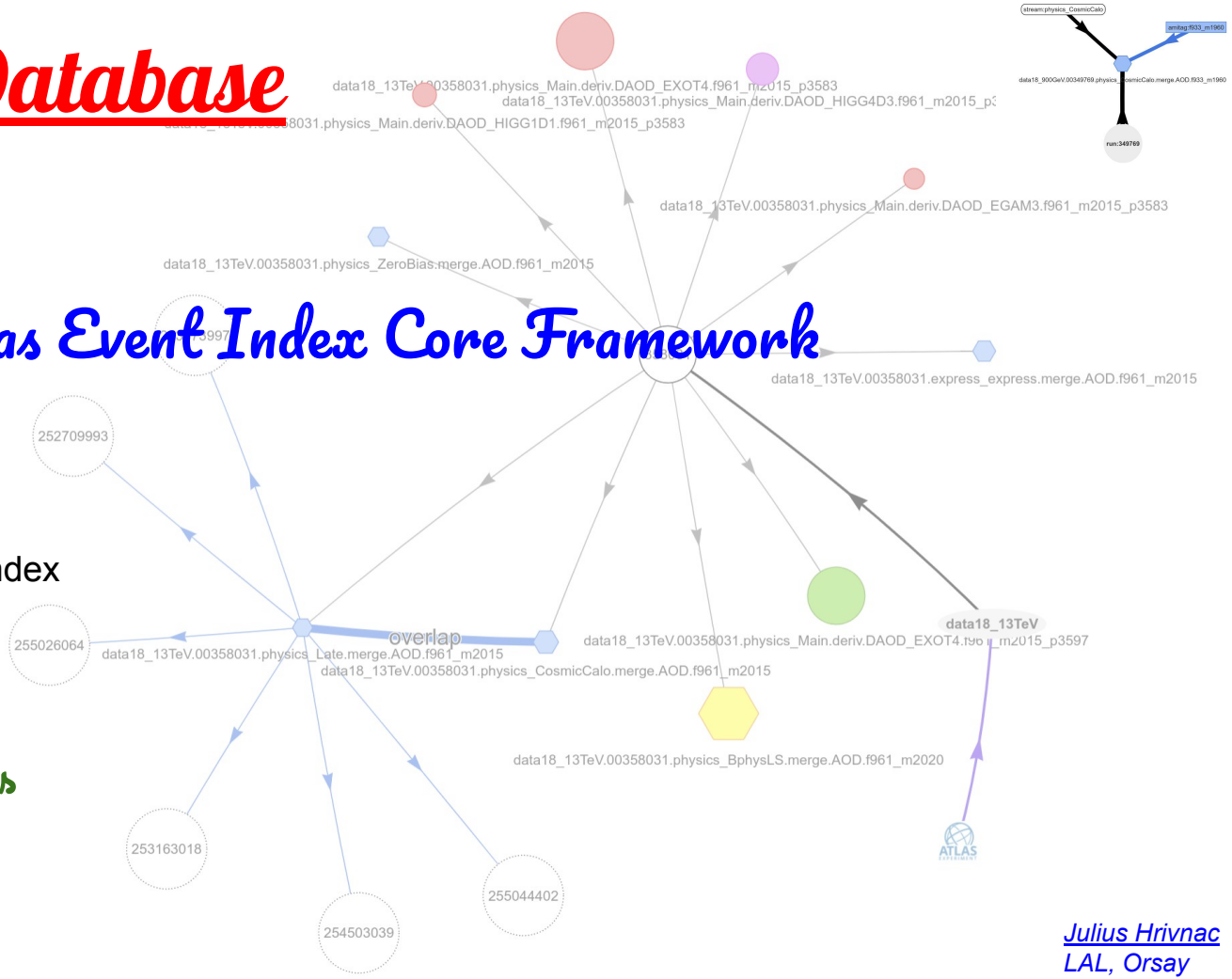
Graph Database



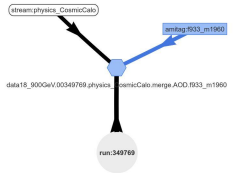
Evolution of the Atlas Event Index Core Framework

- Motivation, History
- What is Graph Database
- Graph DB for Atlas Event Index
  - Database
  - Web Service
  - Migration plans
- *Try It !*

Work in Progress

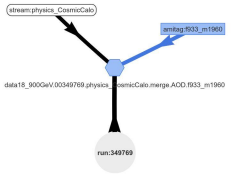
Julius Hrivnac
LAL, Orsay

Graph Databases for HEP



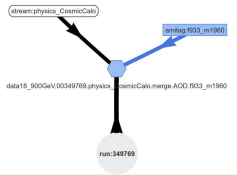
- Traditional data structures in HEP:
  - tuples (tables)
  - trees
  - nested tuples (trees of tuples)
  - relational (SQL-like)
- Schemafull or schemaless
- **But many of our data are graph-like** & schemaless
  - **Entities with relations**
  - G = (V, E) # *graph = (vectors, edges)*
- Not well covered by relational (SQL) databases
  - We don't need only a possibility to add new data with pre-defined relations
  - We need to add new relations
- Graph databases exist since a long time
  - Matured only recently thanks to Big Data & AI (adaptive NN)
  - Very good implementations & (de-facto) standards available
- The difference between SQL and Graph DB is similar as between Fortran and C++/Java
  - On one side, a rigid system, which can be very optimized
  - On the other side, a flexible dynamical system, which allows expressing of complex structures
- GraphDB is a synthesis of OODB and SQLDB
  - Expressing web of objects without fragility of OO world
  - Capturing only essential relations, not an object dump
- Moving structure from data to code
  - Together with migration from imperative to declarative semantics
  - Things don't **happen**, but **exist**

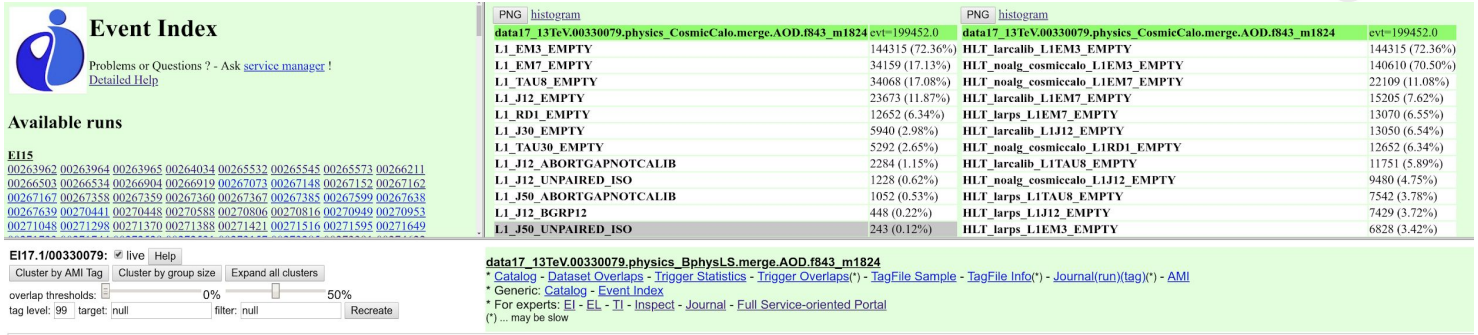
Graph Databases for Event Index

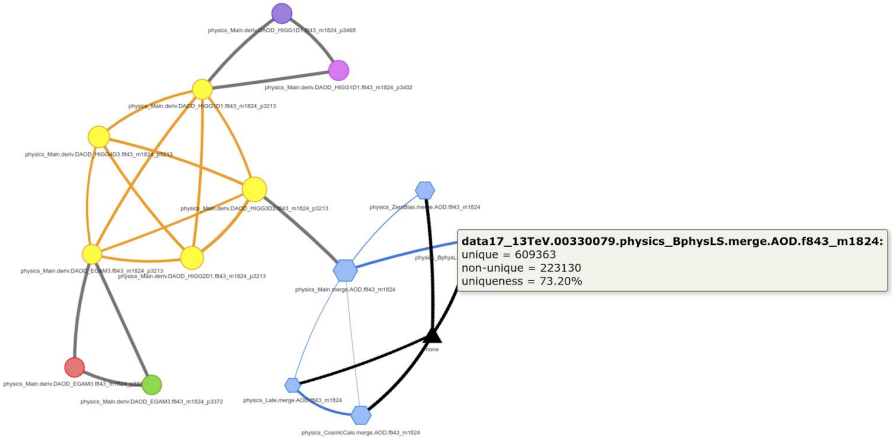


- Original EI in Oracle
  - Too rigid (can't easily add columns, relations), other problems
- Migrated to Hadoop
  - Map files in HDFS
  - Flexible
  - Too slow for searching (ok for processing)
  - Typeless
- Partially migrated to HBase
  - Two tables: Catalog + Events
  - Tables contain a lot of ad-hoc relations (references to other entries)
    - We have in fact implemented a poor-man's GraphDB on top of HBase
- Graphical WS presenting data as graphs

Existing Relational WS







- ***In production***
- ***Simple data interpreted as a Graph by code***
- *To be replaced with GraphDB & generic browser*
- *Aim: Global View of Atlas data with all relations*

Data-oriented Event Index WS

https://atlas-event-index.cern.ch/EIHadoop

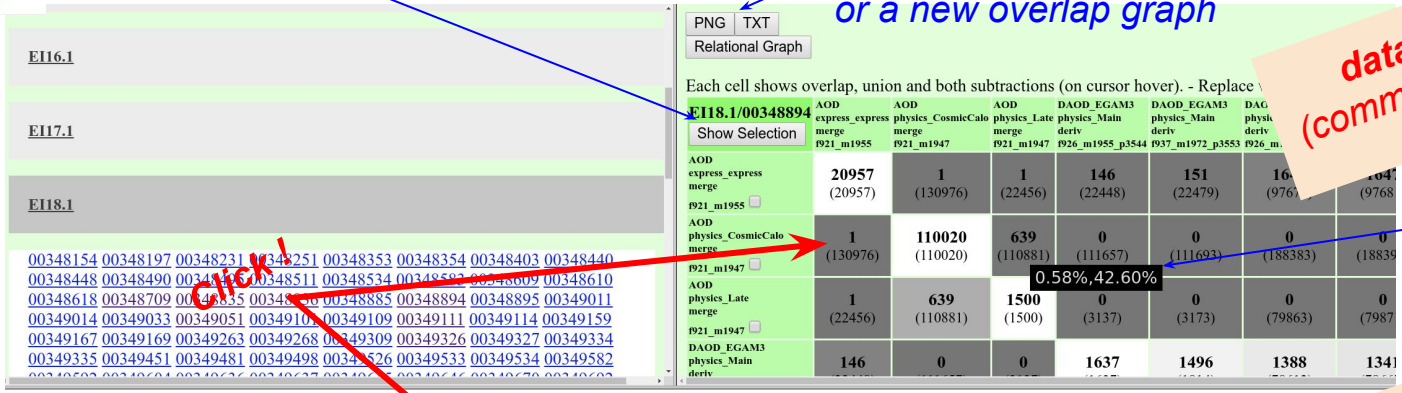
Run Number View

to create a sub-table

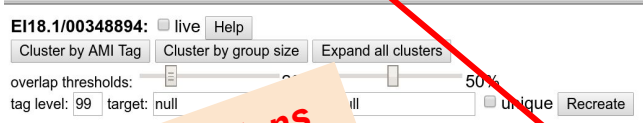
to create a PNG or TXT view of the table
or a new overlap graph

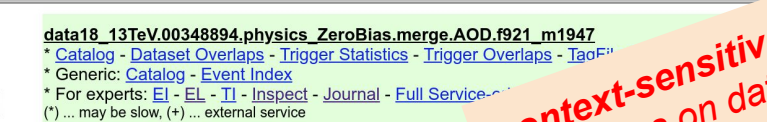
datasets overlap table
(command results go here too)

tables shows
overlap & union,
Hover over cell
to see subtractions



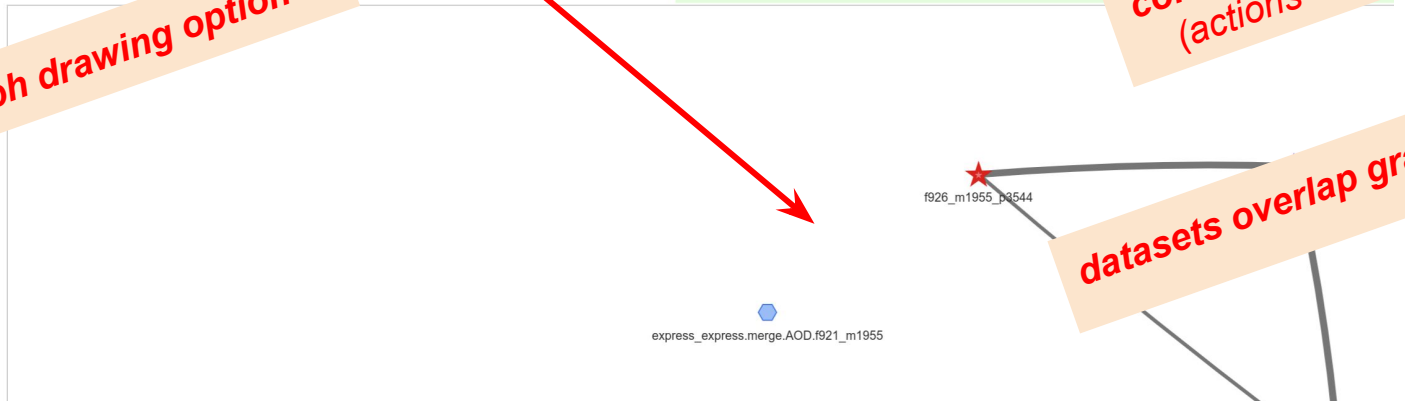
click!





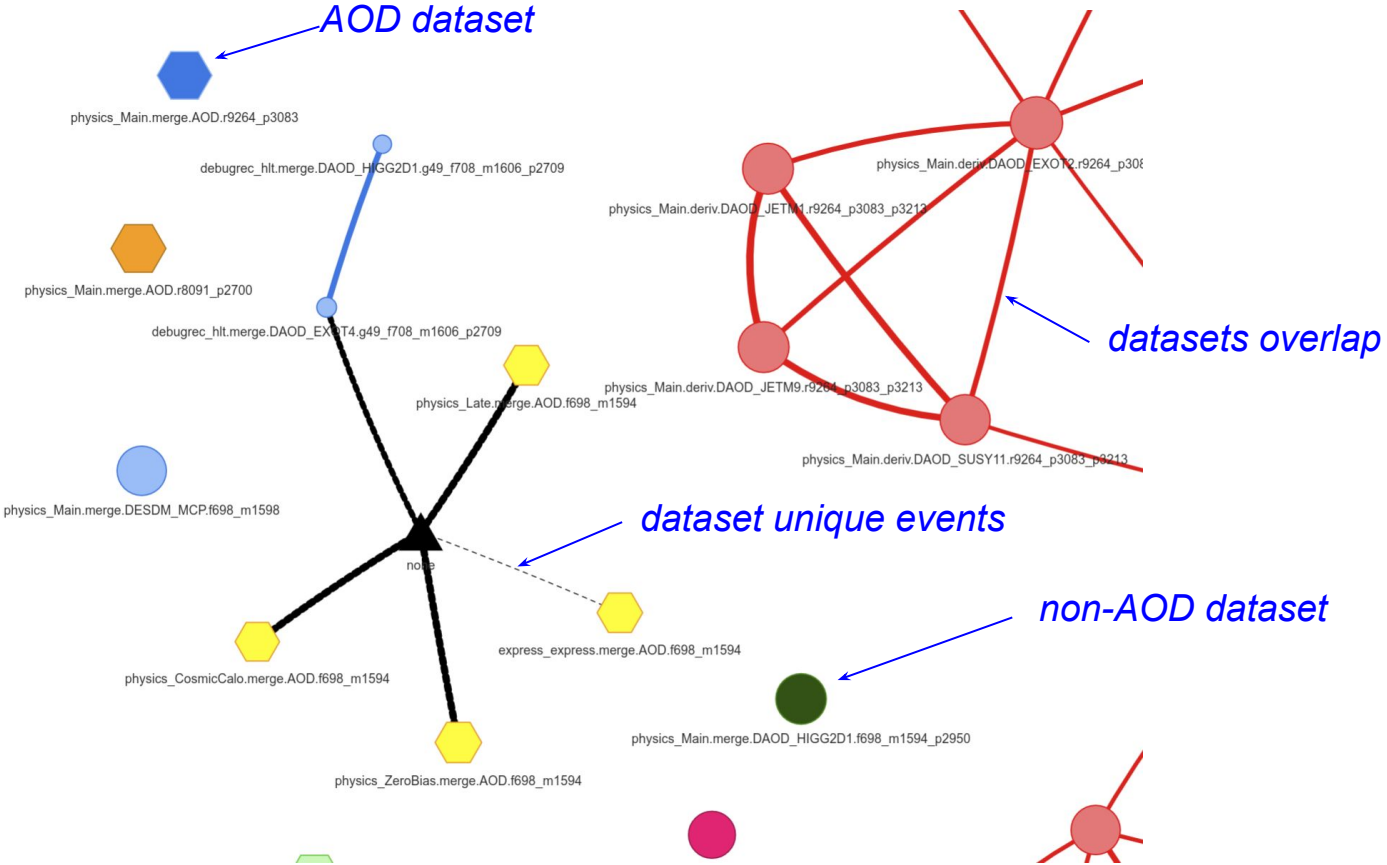
graph drawing options

context-sensitive menu
(actions on datasets)

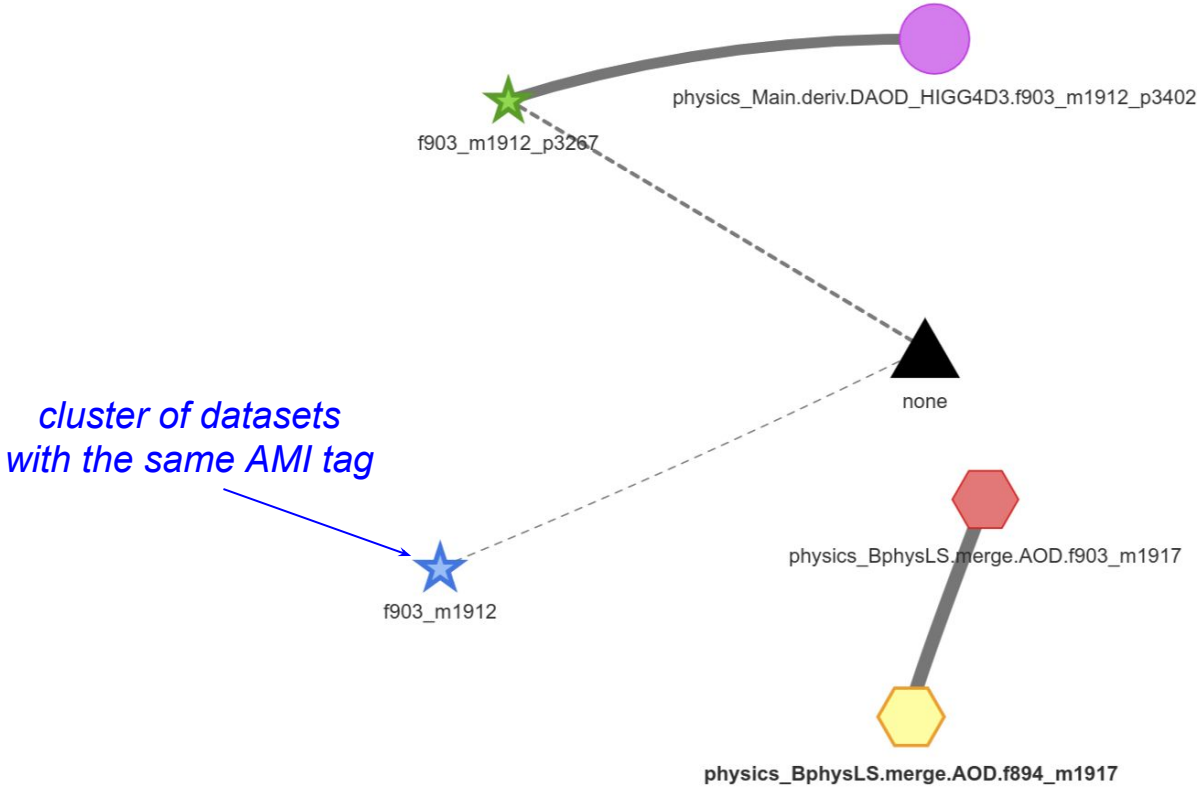


datasets overlap graph

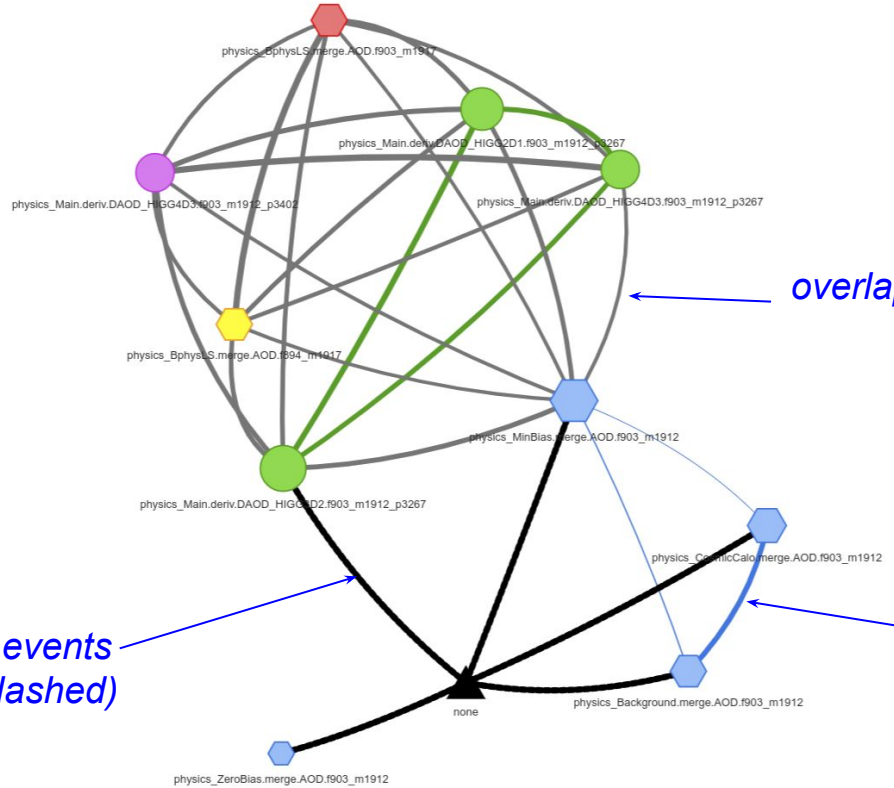
Datasets Graphical View



Datasets Graphical View



Overlaps between Tags, Unique Events

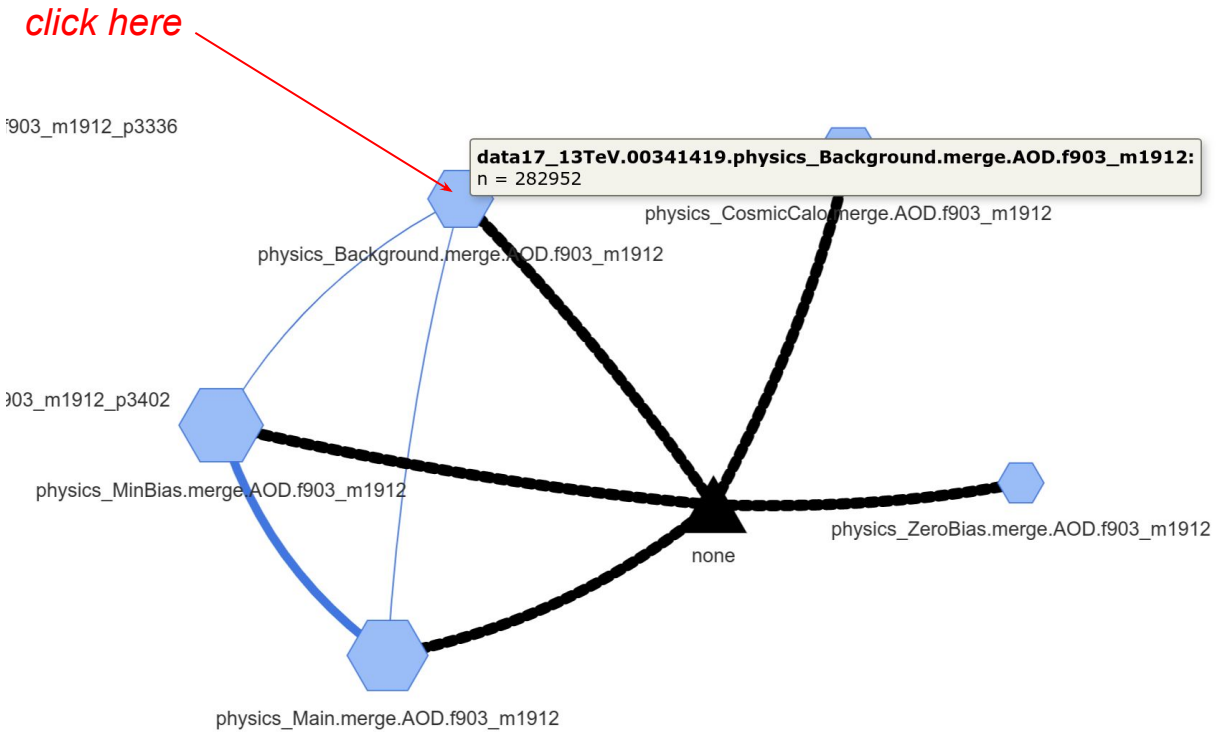


overlap between tags (grey)

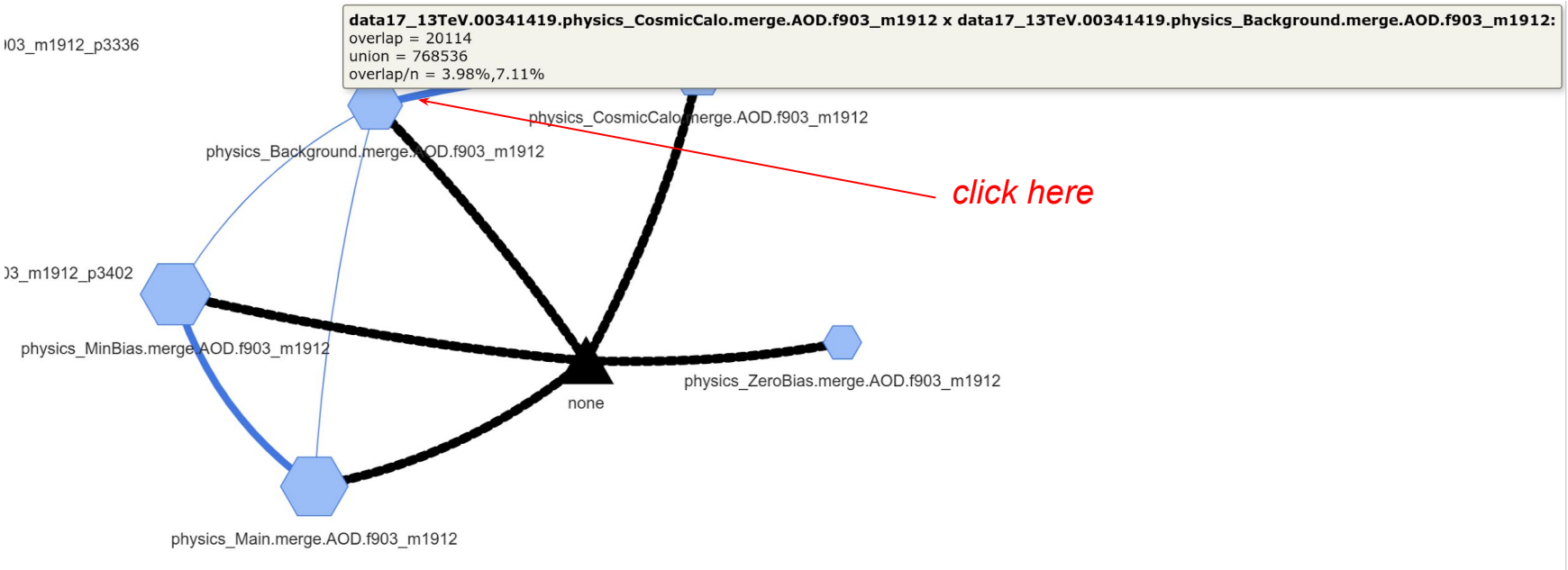
unique events (black dashed)

overlap within tag (with color of the tag)

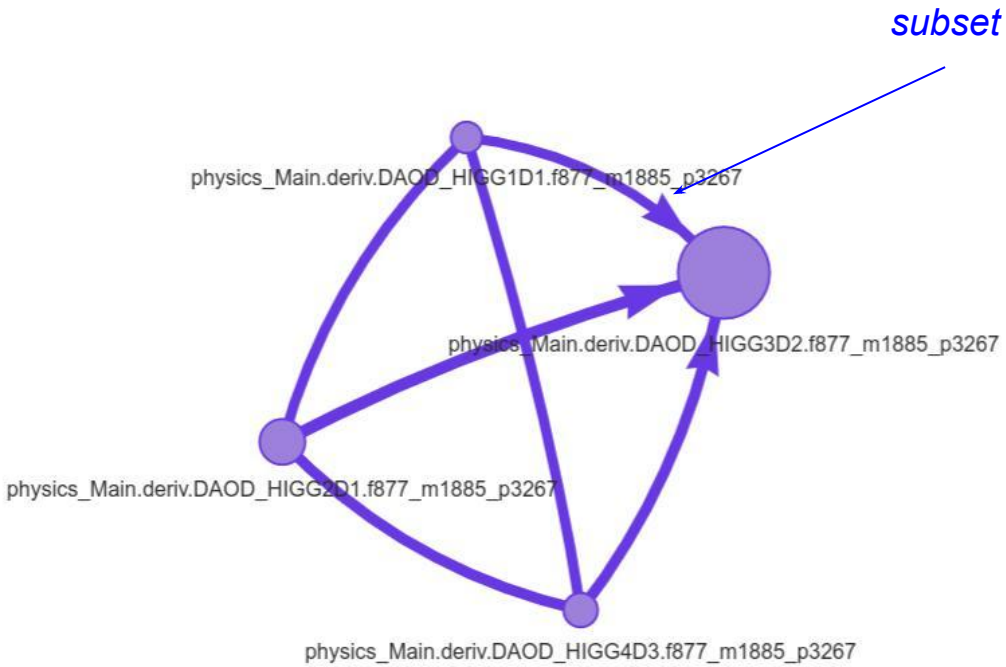
Dataset Details

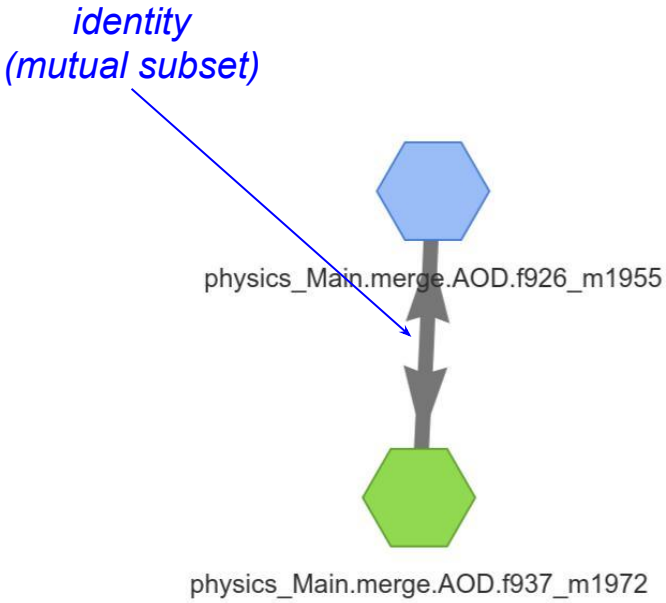


Overlap Details

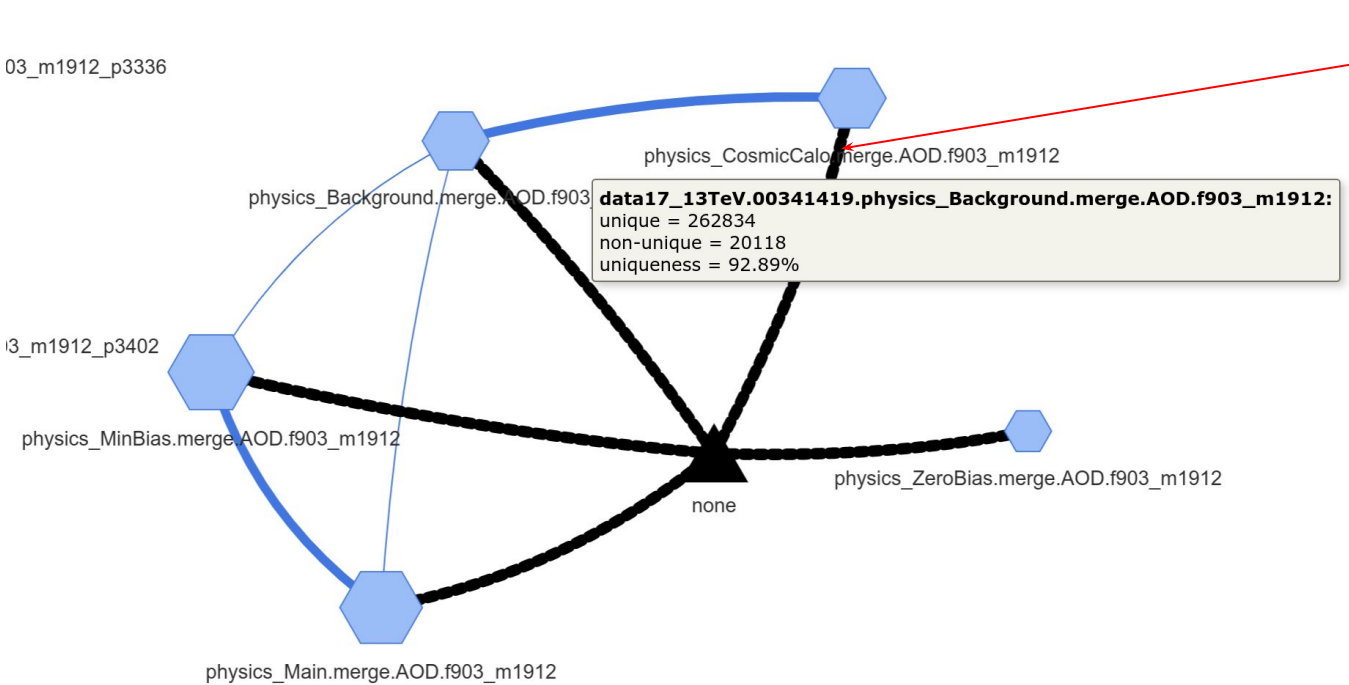


Subsets

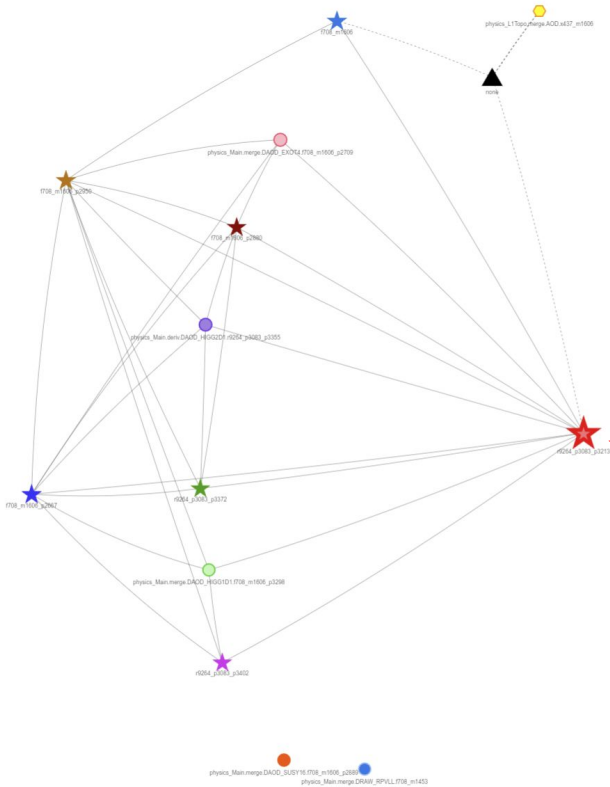

subset
physics_Main.deriv.DAOD_HIGG1D1.f877_m1885_p3267
physics_Main.deriv.DAOD_HIGG3D2.f877_m1885_p3267
physics_Main.deriv.DAOD_HIGG2D1.f877_m1885_p3267
physics_Main.deriv.DAOD_HIGG4D3.f877_m1885_p3267


identity
(mutual subset)
physics_Main.merge.AOD.f926_m1955
physics_Main.merge.AOD.f937_m1972

Unique Events Details



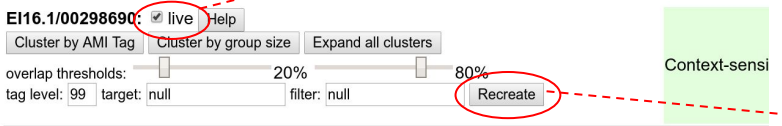
Clusters



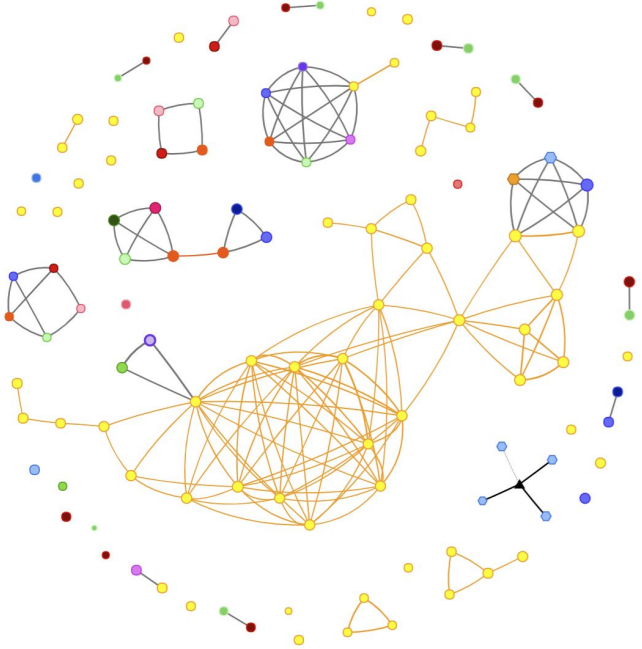
click to expand cluster
(or use menu to expand them all)

Expanded Clusters

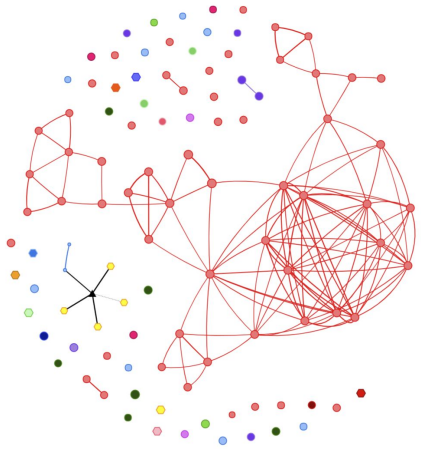
de-select to stop animation

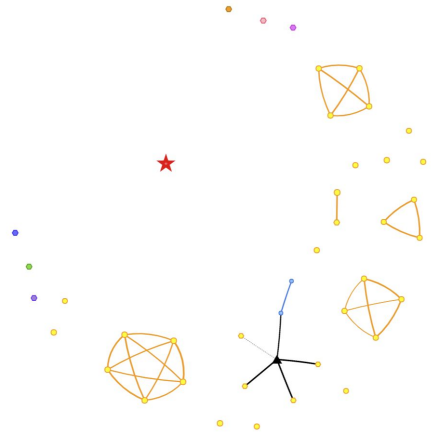


re-create after options change



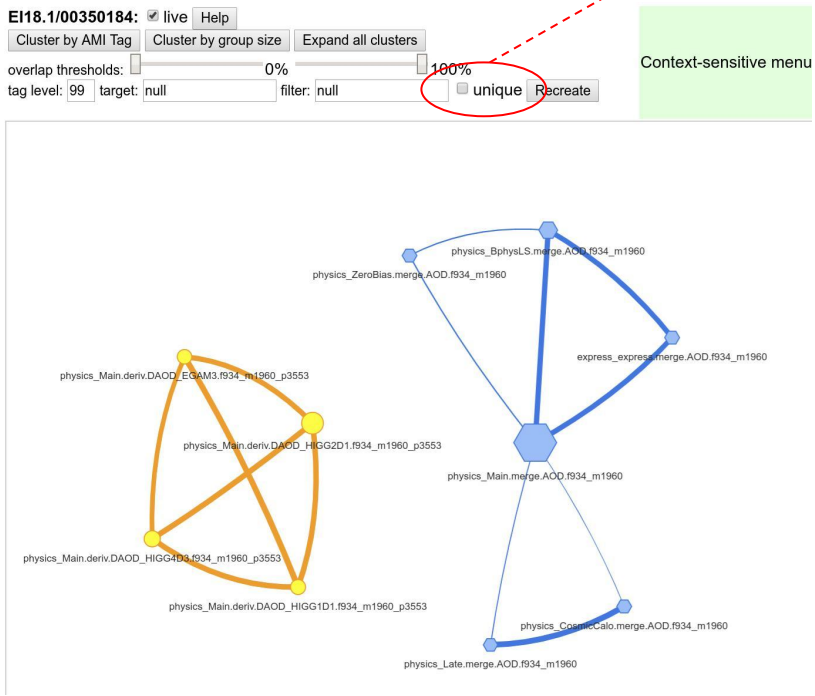
- *zoom to get names & info*
- *move symbols around to re-arrange graph*

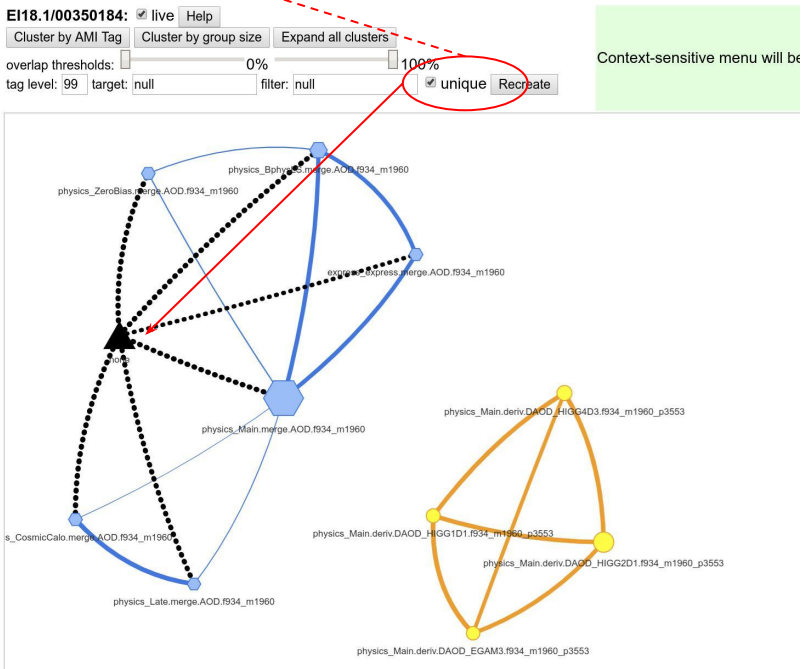




Uniqueness

show uniqueness triangle



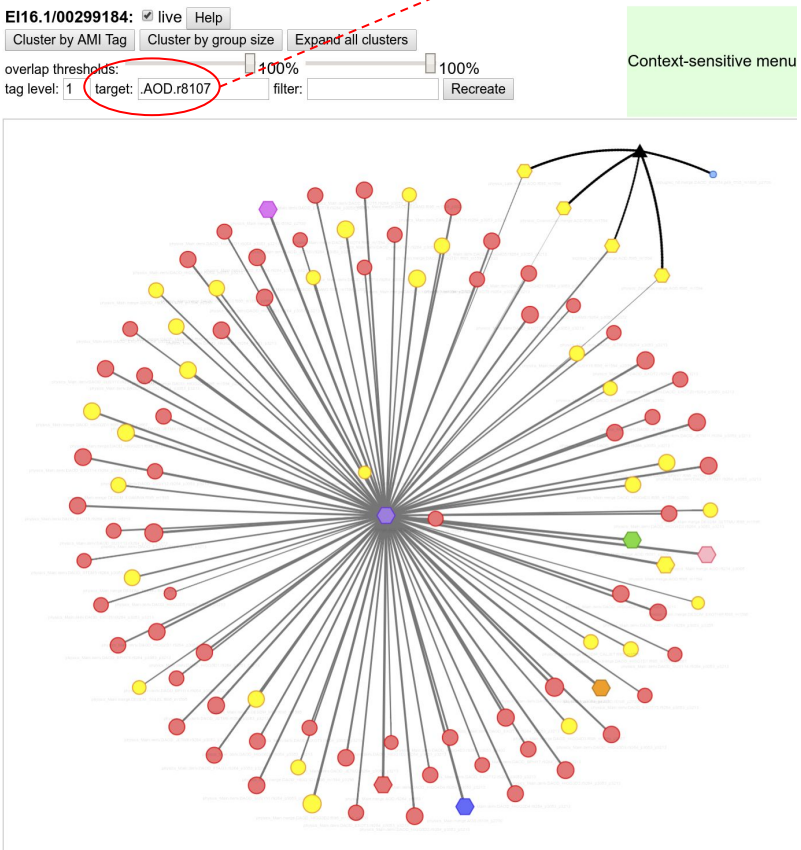


Filter


EI16.1/00299184:
live
Help
Cluster by AMI Tag
Cluster by group size
Expand all clusters
overlap thresholds:
0%
100%
tag level: 1
target:
filter: .AOD.
Recreate
Context-sensitive menu
show only AODs
physics_CosmicCalo.merge.AOD.f698_m1594
physics_Late.merge.AOD.f698_m1594
physics_Main.merge.AOD.f698_m1594
physics_Main.merge.AOD.r8109_p2700
physics_ZeroBias.merge.AOD.f698_m1594
express_express.merge.AOD.f698_m1594
physics_Main.merge.AOD.r9214_p3068
physics_Main.merge.AOD.r8108_p2700
physics_Main.merge.AOD.r8092_p2700
physics_Main.merge.AOD.r8091_p2700

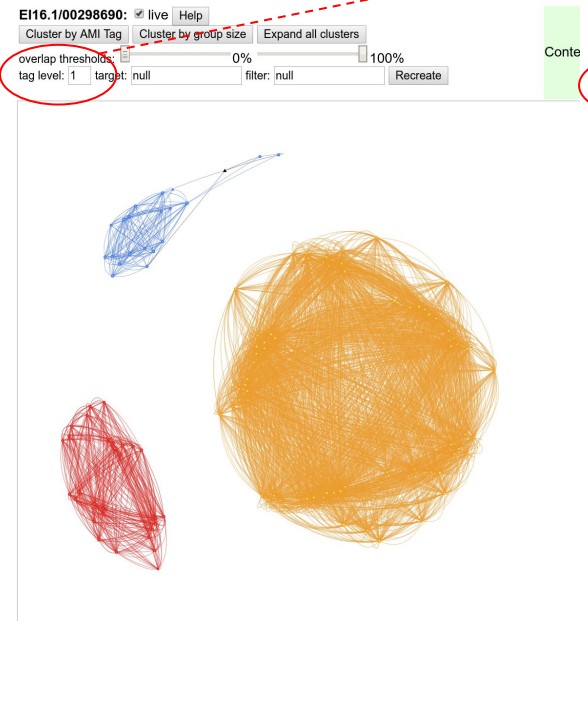
Target

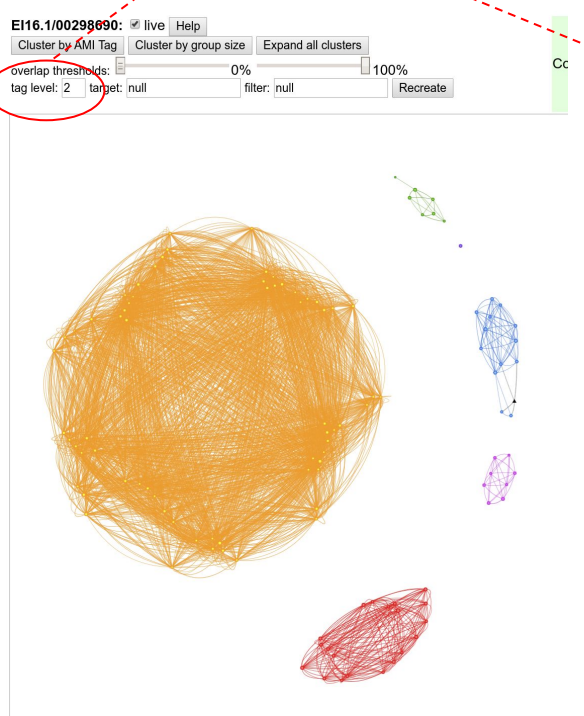
show all overlaps to AOD & r8107

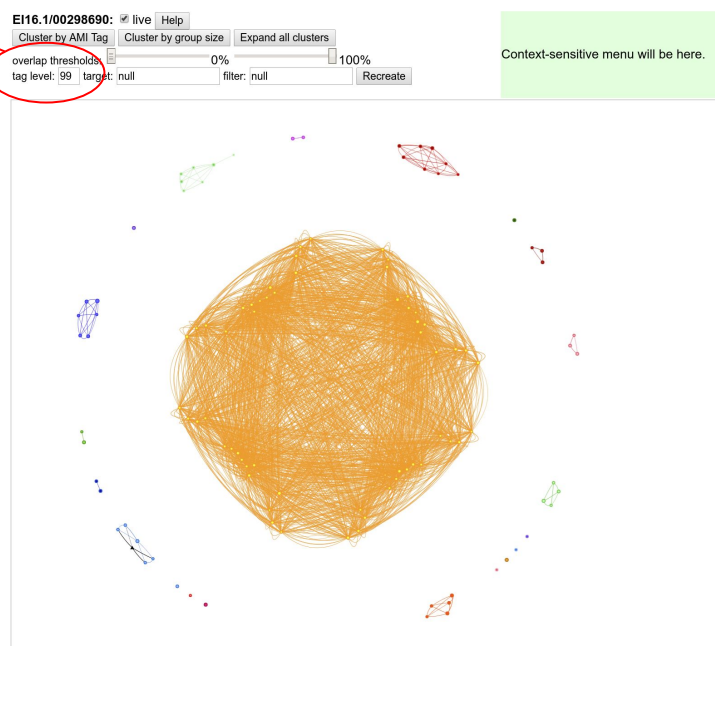


All Overlaps (different tag levels)

tag levels to be considered as equivalent







All Overlaps of AODs

show all overlaps within a tag

show all overlaps between tags

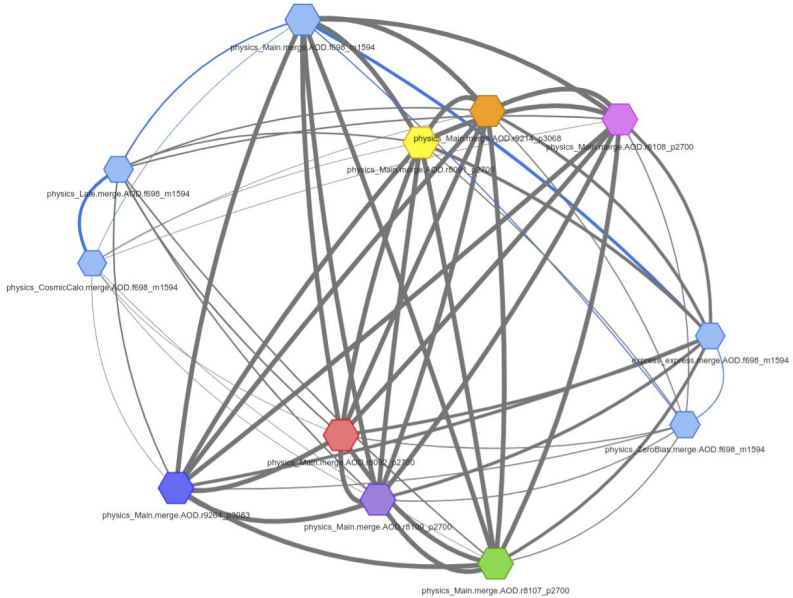
EI16.1/00299184: live Help

Cluster by AMI Tag | Cluster by group size | Expand all clusters

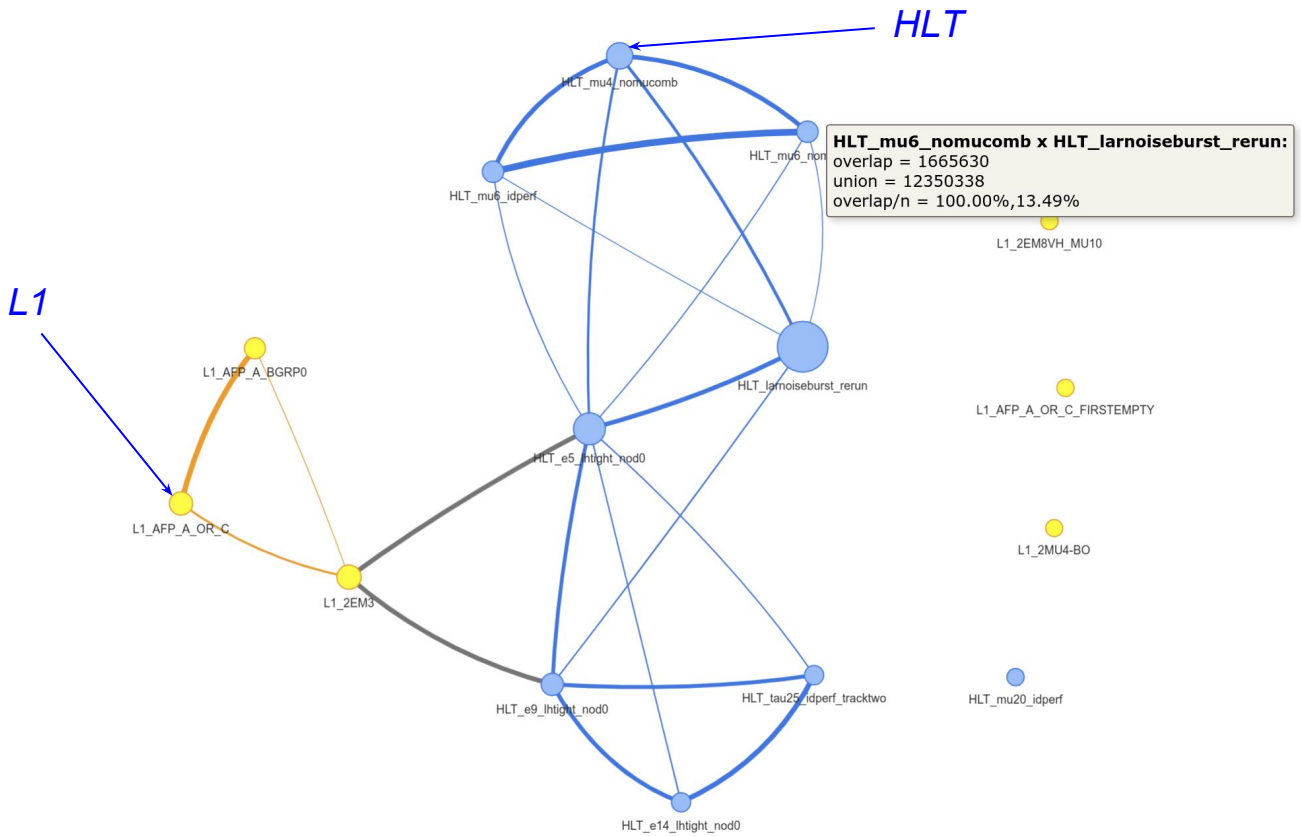
overlap thresholds: 0% 0%

tag level: 1 target: filter: .AOD. Recreate

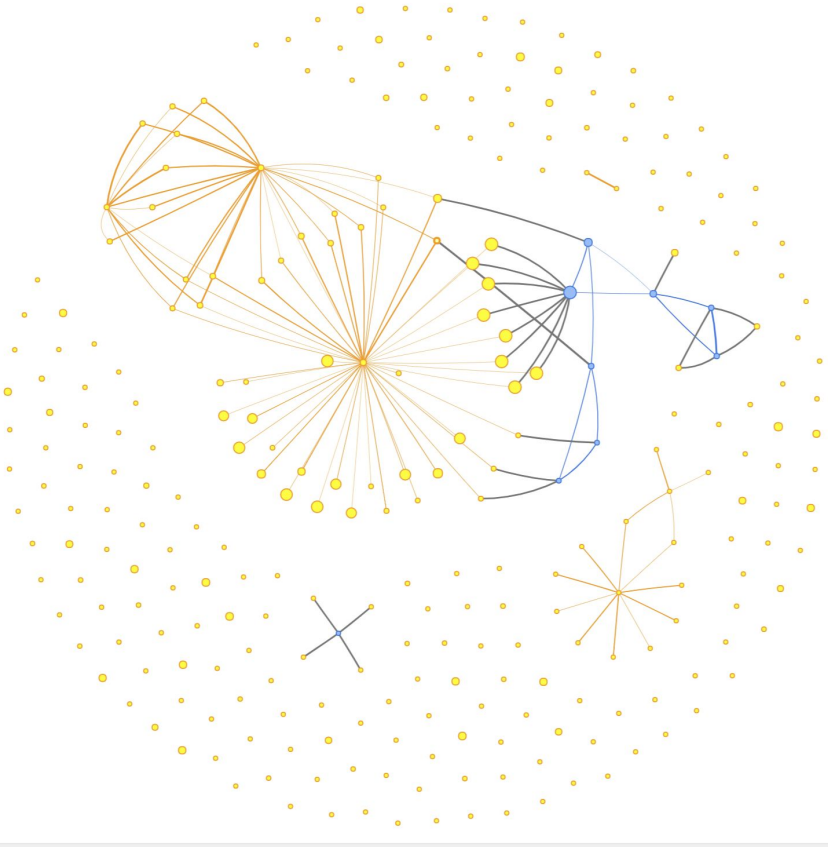
Context-sensitive menu will be here.



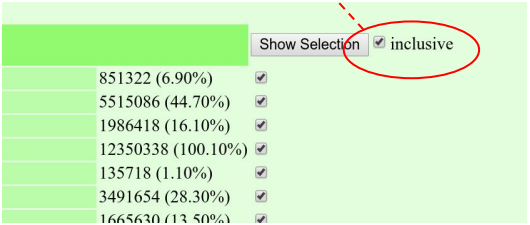
Trigger Overlaps



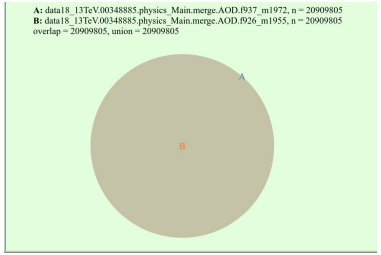
Trigger Overlaps (inclusive)



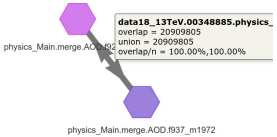
shows all triggers,
but only overlaps to selected ones

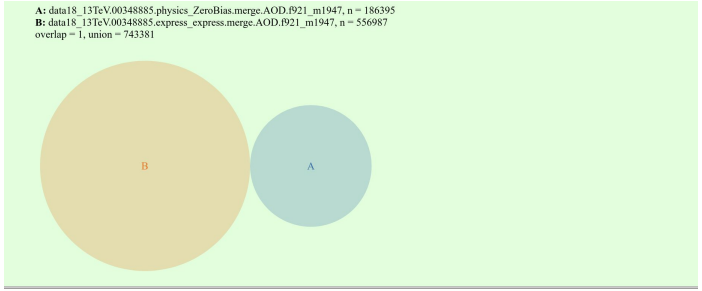


Venn Diagrams

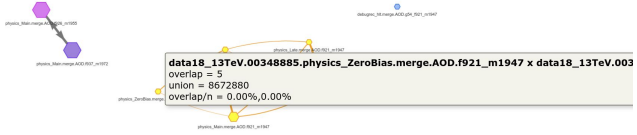


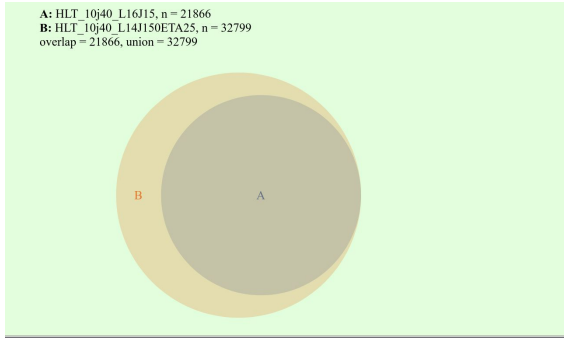
data18_13TeV.00348885.physics_Main.merge.AOD.f937_m1972 x data18_13TeV.00348
Venn Diagram



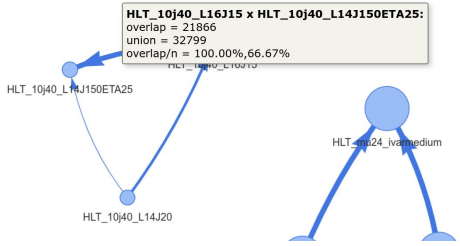


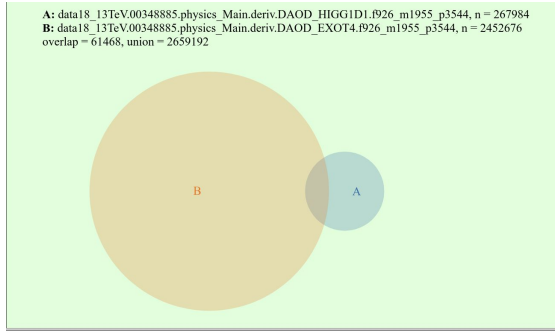
data18_13TeV.00348885.physics_ZeroBias.merge.AOD.f921_m1947 x data18_13TeV.00348885.express_express.merge.AOD.f921_m1947
Venn Diagram



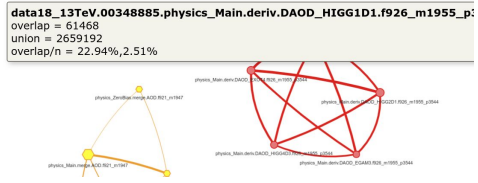


HLT_10j40_L16J15 x HLT_10j40_L14J150ETA25
Venn Diagram

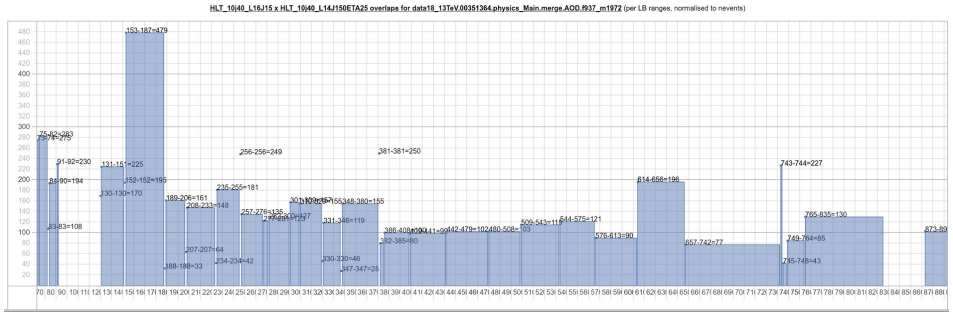




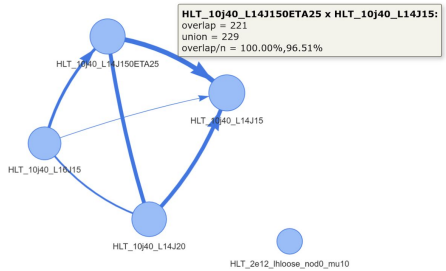
data18_13TeV.00348885.physics_Main.deriv.DAOD_HIGG1D1.f926_m1955_p3544 x data18_13TeV
Venn Diagram

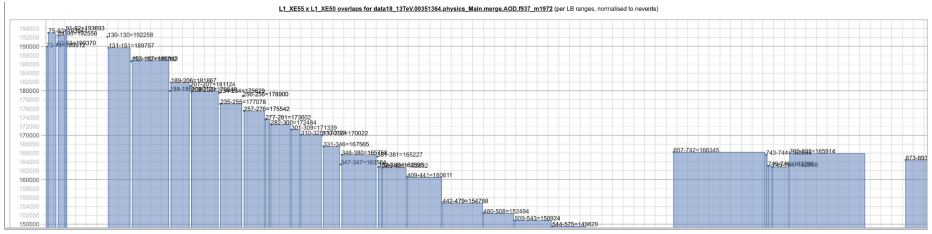


Trigger Overlaps for LB Ranges

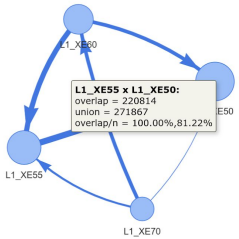


HLT_10j40_L16J15 x HLT_10j40_L14J150ETA25
Venn Diagram - LB Graph

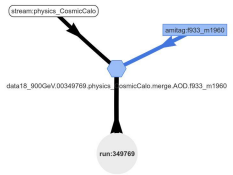


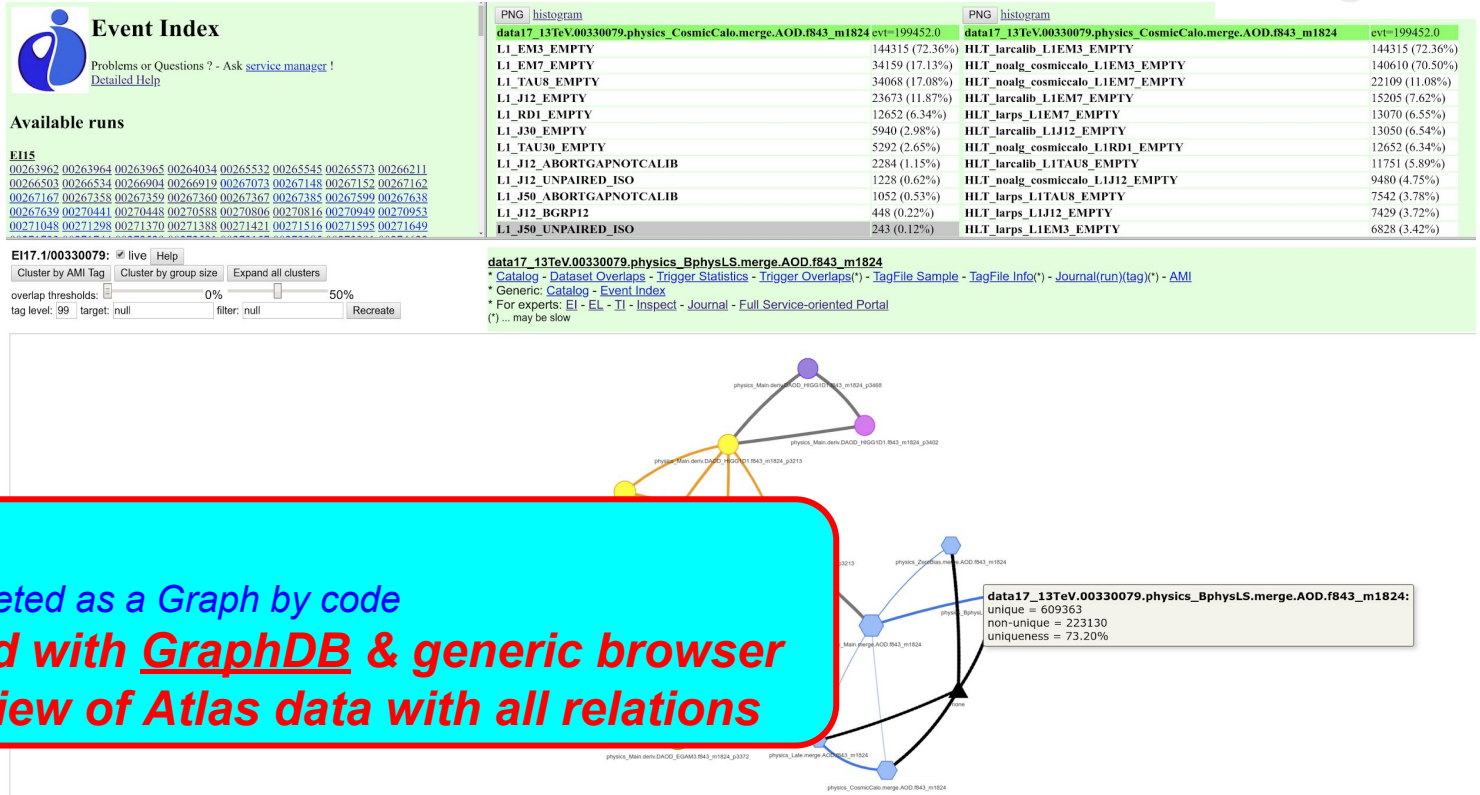


L1_XE55 x L1_XE50
Venn Diagram - LB Graph(*)



Existing Relational WS





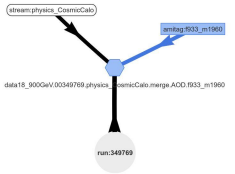
- *In production*
- *Simple data interpreted as a Graph by code*
- ***To be replaced with GraphDB & generic browser***
- ***Aim: Global View of Atlas data with all relations***

Data-oriented Event Index WS

https://atlas-event-index.cern.ch/EIHadoop

Graph Databases

Standards & Choices

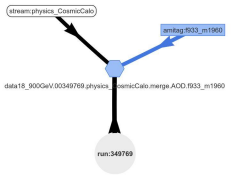


Functional syntax with additional navigational semantics !

- De-facto standard language/api: **Gremlin**
  - Gremlin is a functional, data-flow language to **traverse a property graph.** Every Gremlin traversal is composed of a sequence of (potentially nested) steps. A step performs an atomic operation on the data stream. Every step is either a *map*-step (transforming the objects in the stream), a *filter*-step (removing objects from the stream), or a *sideEffect*-step (computing statistics about the stream).
  - Gremlin supports **transactional** & **non-transactional** processing in **declarative** or **imperative** manner.
  - Gremlin can be expressed in all languages supporting function composition & nesting.
  - Supported languages: **Java**, **Groovy**, Scala, Python, Ruby, Go, …
- Commonly used framework: **TinkerPop**
- Leading implementation: **JanusGraph**
  - Supported storage backends: Cassandra, **HBase**, Google Cloud, Oracle BerkeleyDB
  - Supported graph data analytics: Spark, Giraph, Hadoop
  - Supported searches: Elastic Search, Solr, Lucene
  - Other candidate: **Neo4j**, the same Gremlin interface, used by Atlas Geometry DB
- Chosen visualisation: **visj.org**

GraphDB Schema for EI

realisation



Customize the interactions with the **graph**.

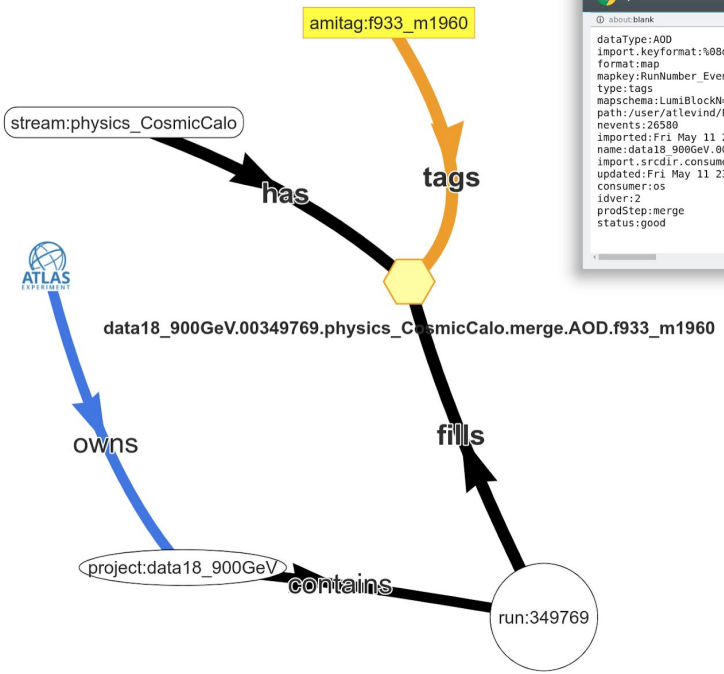
Cluster by group type | Cluster by group size | Expand all clusters

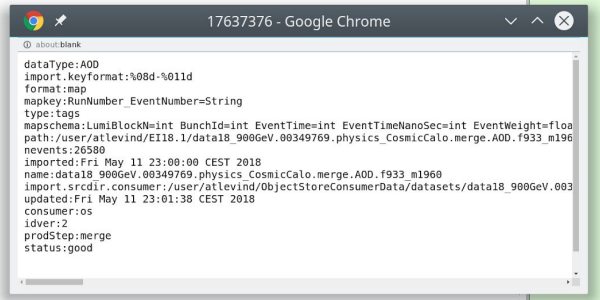
live | expand children | expand parents | remove old

data18_900GeV.00349769.physics_CosmicCalo.merge.AOD.f933_m1960 Remove Describe

Catalog - Sample - Info -

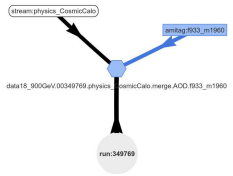
Expanding data18_900GeV.00349769.physics_CosmicCalo.merge.AOD.f933_m1960 # Showing 0 new elements # Showing 0 new elements # Showing 3 new elements # Showing 3 new elements #





--- commands output ---

GraphDB Schema for EI



overlap
overlapping_events
overlapB
overlapC

dataset
prodStep
dataType
status
nevents
unique_events
…
triggerStat: url
triggerOverlap: url
…
name()
...

has

stream
name

tags

amitag
version

keeps

event
number
LumiBlockN
…
XtrigChainsXXX: Set[String]
guid: Set[String]
...

fills

run
number

contains

project
name

owns

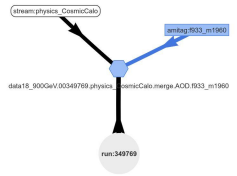
experiment
name

- Should decide what is a feature and what is a relation
- Arrows have only logical meaning, they can be navigated equally from both sides
- Relations and features have defined multiplicities (not shown here) and types (int, string, date,...)
- Defined entities (and combinations of them) can be indexed for fast search
- Other features and relations can be freely added to any entity (vertex or edge)

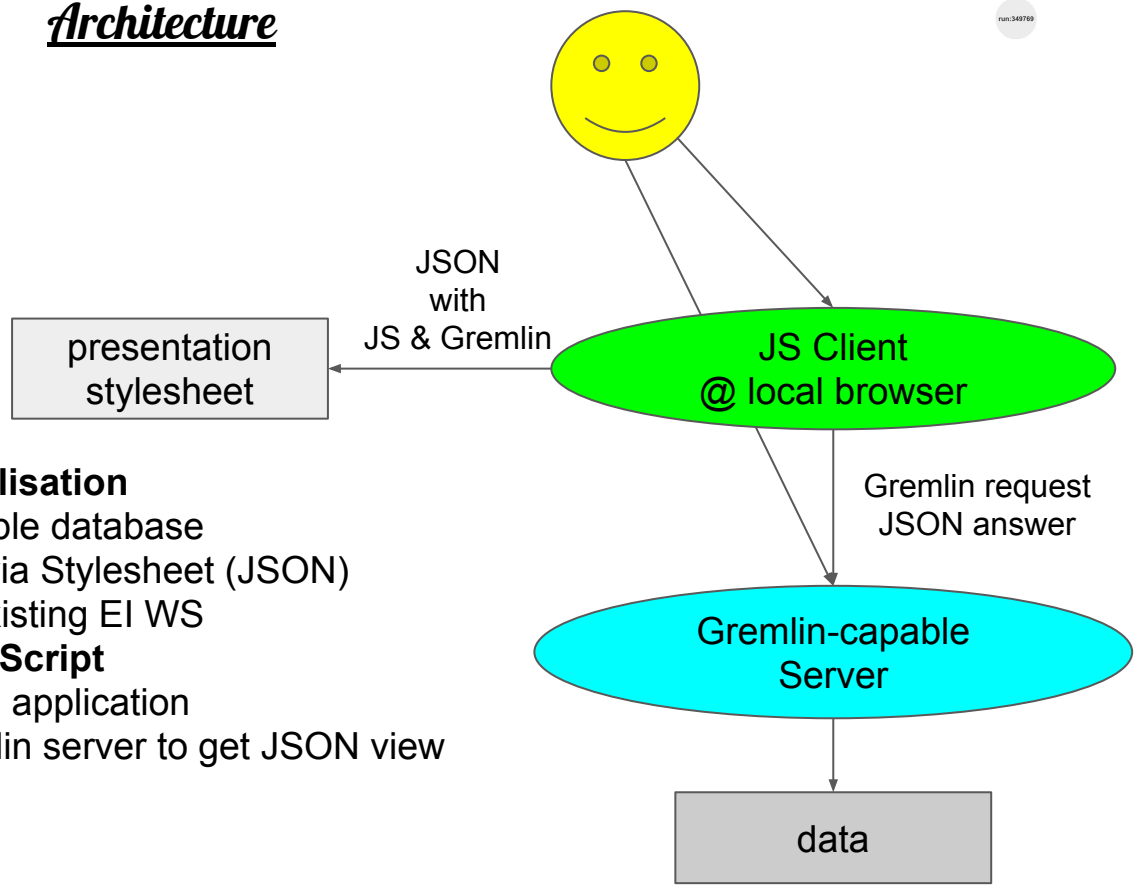
https://atlas-event-index.cern.ch/GraphDB-doc/Gremlin/schema.gremlin

GraphDB for EI

Architecture

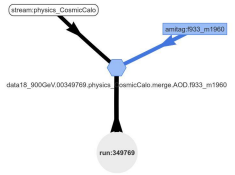


- Using standard Graph DB
  - Importing data from EI
- **Generic Web Service graphical visualisation**
  - Can display any Gremlin-compatible database
  - Visualisation can be customised via Stylesheet (JSON)
    - To give the same L&F as existing EI WS
  - Implemented **completely in JavaScript**
    - So doesn't need server-side application
    - Connects to standard Gremlin server to get JSON view of data



GraphDB for EI

Presentation Stylesheet



how to present "dataset" vertex

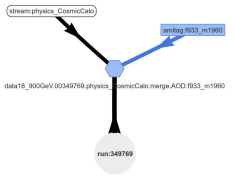
how to show it
(can contain Gremlin or JS code)

```
"dataset": {
      graphics: {
        label:{gremlin:"sideEffect(values('prodStep').store('4')).sideEffect(values('dataType').....values().join().toString()"},
        title:"dataType",
        subtitle:{gremlin:"values('nevents').join().toString().concat(' events')"},
        group:{gremlin:"in().hasLabel('amitag').values('version')"},
        shape:{js:"if(title=='dataset:AOD') {shape = 'hexagon';} else {shape = 'dot';}"},
        value:{gremlin:"values('nevents').join().toString()"}
        },
      actions: [
        {name:"Catalog", url:"https://atlas-event-index.cern.ch/EIHadoop/CatalogView.jsp?query=dataset:"},
        {name:"Sample" , url:"https://atlas-event-index.cern.ch/EIHadoop/InspectView.jsp?view=txt&action=dump&climit=1&limit=10&query=dataset:"},
        {name:"Info" ,   url:"https://atlas-event-index.cern.ch/EIHadoop/InspectView.jsp?view=txt&action=info&climit=1&limit=10&query=dataset:"}
        ]
      },
```

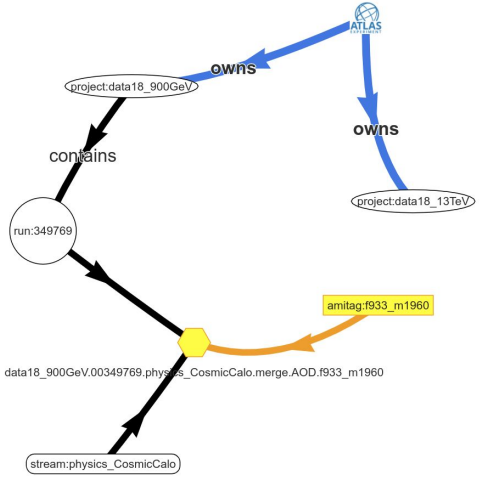
external actions

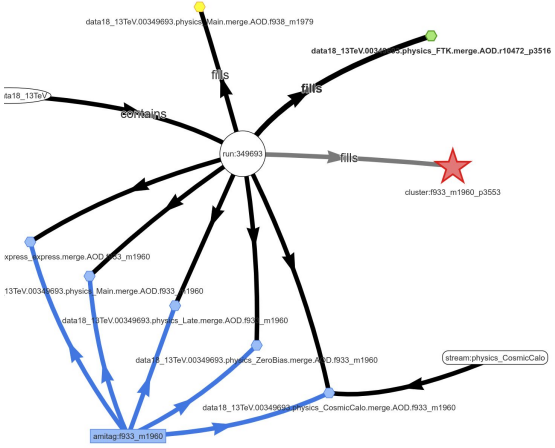
Graph Database for EI

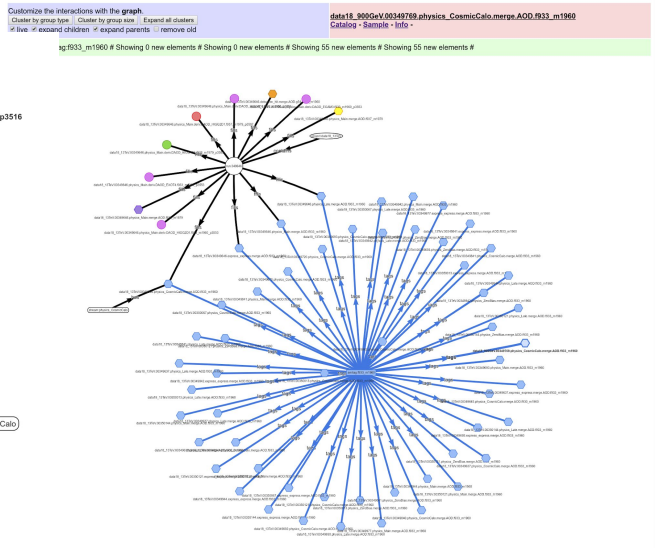
Status



- Using HBase backend
- Subset of data imported
- Most of the graphical part implemented
  - By standalone JS implementation



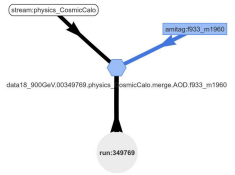




--- commands output ---

Graph Database for EI

Gremlin examples (Groovy style)

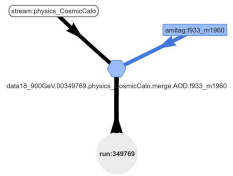


- Functional syntax
- Functional & **navigational** semantics

```
# add a vertex ‘experiment’ with the name ‘ATLAS’
g.addV('experiment').property('ename', 'ATLAS')
# add edges ‘owns’ from all vertices ‘project’ to vertex ‘experiment’ ‘ATLAS’
g.V().hasLabel('project').addE('owns').from(g.V().hasLabel('experiment').has('ename', 'ATLAS'))
# a function deriving a dataset name (which is not stored as such)
# from existing dataset relations by traversing the graph
def datasetName(d) {
  return d.sideEffect(values("prodStep").store("4"))
          .sideEffect(values("dataType").store("5")).in()
          .sideEffect(hasLabel("run").values("rname").store("2"))
          .sideEffect(hasLabel("amitag").values("version").store("6"))
          .sideEffect(hasLabel("stream").values("sname").store("3"))
          .sideEffect(hasLabel("run").in().hasLabel("project").values("pname").store("1"))
          .cap("1", "2", "3", "4", "5", "6").next().values().join().toString()
  }
```

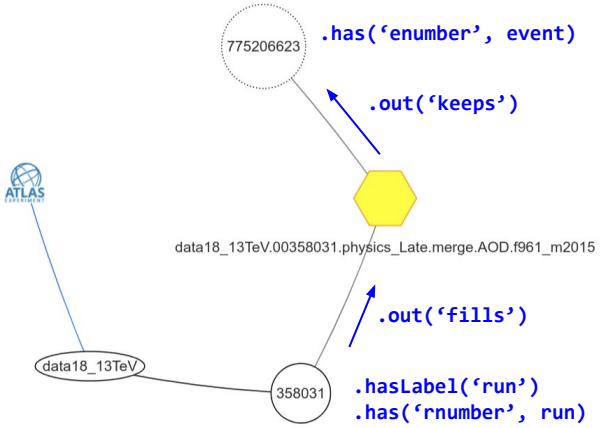
Graph Database for EI

Event Lookup (Groovy style)



- Both search and traversal steps
- Search steps can be boosted by indexes

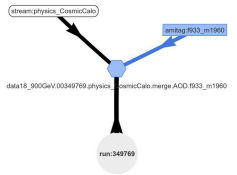
```
# Event-Lookup function (loaded inside the server)
def el(run, event, g) {
  e = g.V().hasLabel('run')          # all runs
           .has('rnumber', run)      # selected run
           .out('fills')             # all datasets filling that run
           .out('keeps')             # all events kept in that dataset
           .has('enumber', event)    # selected event
           .values('guid')           # its guid
  }
```




```
# CLI command
curl -XPOST -d '{"gremlin":"el(run, event, amitag)"}' http://ei-gremlin-server.cern.ch:8182
# or using standard gremlin client
gremlin << EOF
:remote connect tinkerpop.server $janusgraph_home/conf/remote.yaml
el(run, event, amitag)
EOF
```

Graph Database for EI

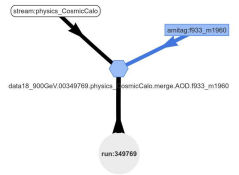
Event Lookup (Performance - 1)




```
gremlin> el(358031, 775206623, g).profile()
==>Traversal Metrics
Step                                                               Count  Traversers       Time (ms)    % Dur
=============================================================================================================
JanusGraphStep([],[~label.eq(event), enumber.eq...                     1           1         204.805    75.74
    \_condition=(~label = event AND enumber = 775206623)
    \_isFitted=true
    \_query=multiKSQ[1]@2147483647
    \_index=event:enumber:u
    \_orders=[]
    \_isOrdered=true
  optimization                                                                               4.614
  optimization                                                                             130.444
  backend-query                                                        1                     7.742
    \_query=event:enumber:u:multiKSQ[1]@2147483647
JanusGraphVertexStep(IN,[keeps],vertex)                                1           1          25.560     9.45
    \_condition=type[keeps]
    \_isFitted=true
    \_vertices=1
    \_query=org.janusgraph.diskstorage.keycolumnvalue.SliceQuery@b3a55b7f
    \_orders=[]
    \_isOrdered=true
  optimization                                                                              11.927
  backend-query                                                        1                     3.103
    \_query=org.janusgraph.diskstorage.keycolumnvalue.SliceQuery@b3a55b7f
```

Graph Database for EI

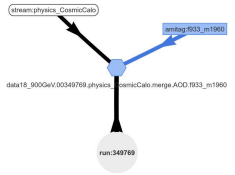
Event Lookup (Performance - 2)



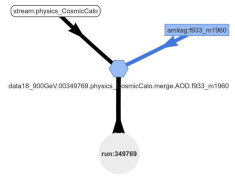
JanusGraphVertexStep(IN,[fills],vertex) _condition=type[fills] _isFitted=true _vertices=1 _query=org.janusgraph.diskstorage.keycolumnvalue.SliceQuery@b3a605c1 _orders=[] _isOrdered=true	1	1	10.388	3.84
optimization			7.661	
backend-query	1		1.442	
_query=org.janusgraph.diskstorage.keycolumnvalue.SliceQuery@b3a605c1				
HasStep([rnumber.eq(358031)])	1	1	13.129	4.86
SelectOneStep(last,e)	1	1	0.993	0.37
NoOpBarrierStep(2500)	1	1	0.159	0.06
JanusGraphPropertiesStep([guid],value) _condition=type[guid] _isFitted=true _vertices=1 _query=org.janusgraph.diskstorage.keycolumnvalue.SliceQuery@b11f98a7 _orders=[] _isOrdered=true	2	2	14.800	5.47
optimization			7.478	
NoOpBarrierStep(2500)	2	2	0.568	0.21
>TOTAL	-	-	270.406	-

75% of the time is spend by the entry point search, following graph traversal is very fast

Event Index Migration



- Export of existing data as Gremlin source
- Executing Gremlin source
  - Filling JanusGraph with data in tabular form
- Re-arranging JanusGraph data with Gremlin script
  - Creating graph
- Slow, but very flexible procedure
  - Useful for evaluation
- Next step:
  - Install JanusGraph together with EI Core to enable
    - Performance evaluation
    - Data migration
    - Code migration



Event Index Migration

Client

Apache/Tomcat

EI Core

Hadoop/HBase

Client

EI Core (GraphDB)

JanusGraph,...

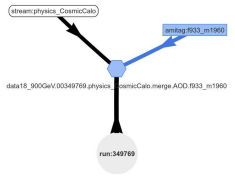
Hadoop/HBase

Cassandra,...

- Data (almost) without structure
- **Complex code** (error-prone)

- **Structured data**
- Interpreted by JanusGraph

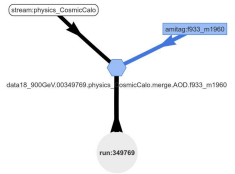
Problems
(not serious)



- Physical configuration of non-default set of tools
  - API ok
- Coexistence with Hadoop/HBase
  - Many overlapping libs with different versions
- Naive JavaScript implementation
  - Should use a framework
- JavaScript Gremlin client exists only for Node.js
  - Browser-based client being developed here
    - Not completely generic
- No good generic Gremlin GUI
  - Some very good special-purpose GUIs, or commercial ones
  - Our developpement already quite good
    - Generic, customised by powerful stylesheet
- Web Service (Gremlin Service) security
  - As always

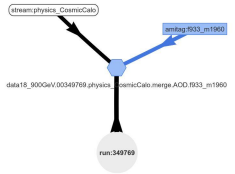
Added Values

wrt current implementation



- **Big part of the current Core absorbed in GraphDB structure**
  - In past, we had in fact implemented our own GraphDB
    - Common development pattern in HEP :-)
- Apache/Tomcat service not needed
  - JavaScript client connects directly do Gremlin server
- No special CLI API needed
  - Standard Gremlin functions used
    - Can provide command wrappers for backward compatibility
- Using standards
  - So components can be replaced
    - JanusGraph with Neo4J
    - Hadoop with Cassandra
- New interface with the same (and enhanced) functionality
  - Also for WS
- Same or better performance
  - As the internal DB structure is very similar + better code

Try It !



- Growing documentation: https://atlas-event-index.cern.ch/GraphDB-doc
- Web GUI (should work inside CERN): http://aiatlas016.cern.ch/GraphDB

A 'playground' has been installed @aiatlas016.
It contains the full Catalog, several datasets and DOverlap tables.
To play with it: connect to aiatlas016 as atlevind
(or checkout https://:@gitlab.cern.ch:8443/atlas-event-index/GraphDB.git).

```
$ cd work/DB/GraphDB/ant
$ source setup.sh
$ gremlin_local
```

And try some examples from
https://atlas-event-index.cern.ch/GraphDB-doc/Gremlin/examples.gremlin
https://atlas-event-index.cern.ch/GraphDB-doc/Gremlin/functions.gremlin
https://atlas-event-index.cern.ch/GraphDB-doc/Gremlin/tests.gremlin