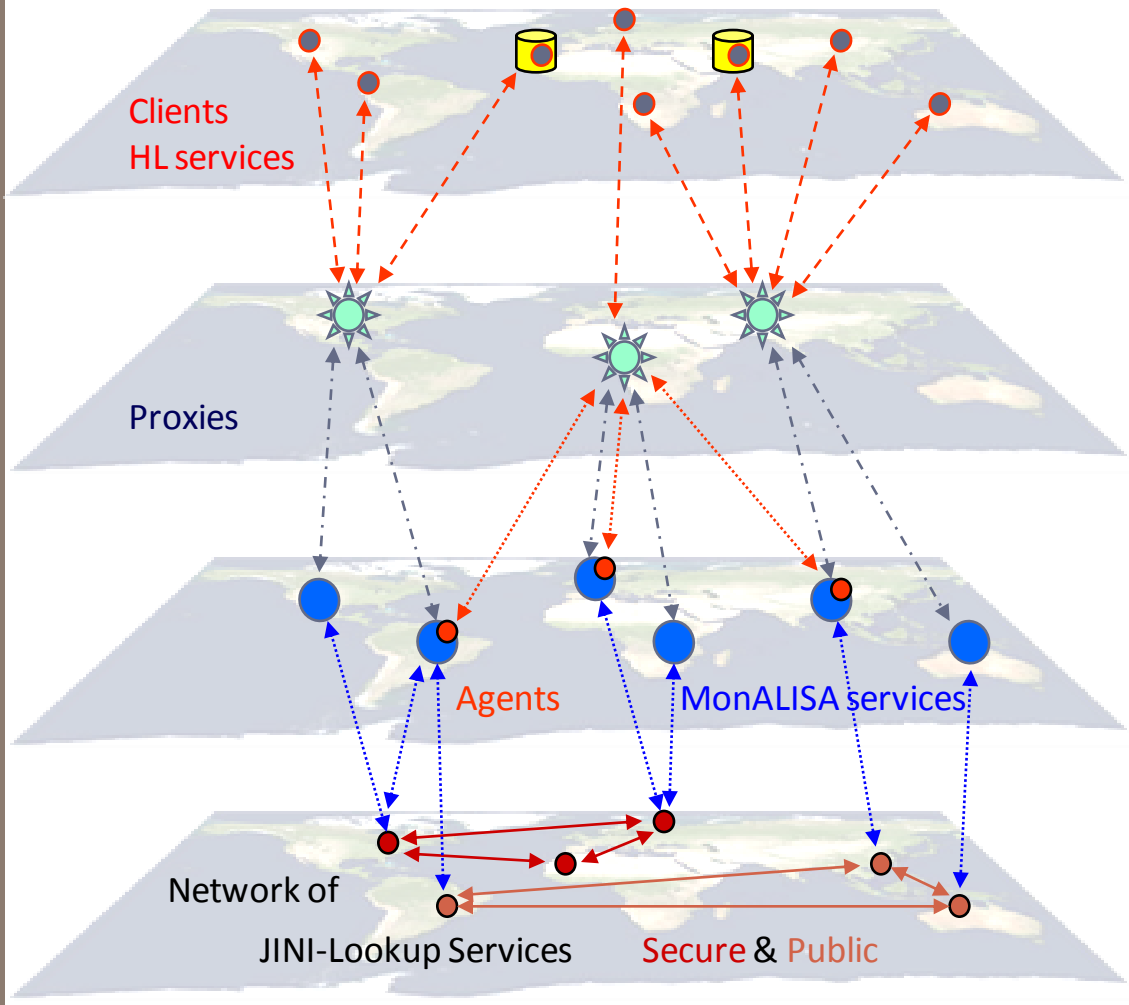
MONALISA MONITORING AND CONTROL

Costin Grigoras <costin.grigoras@cern.ch>

OUTLINE



Data consumers

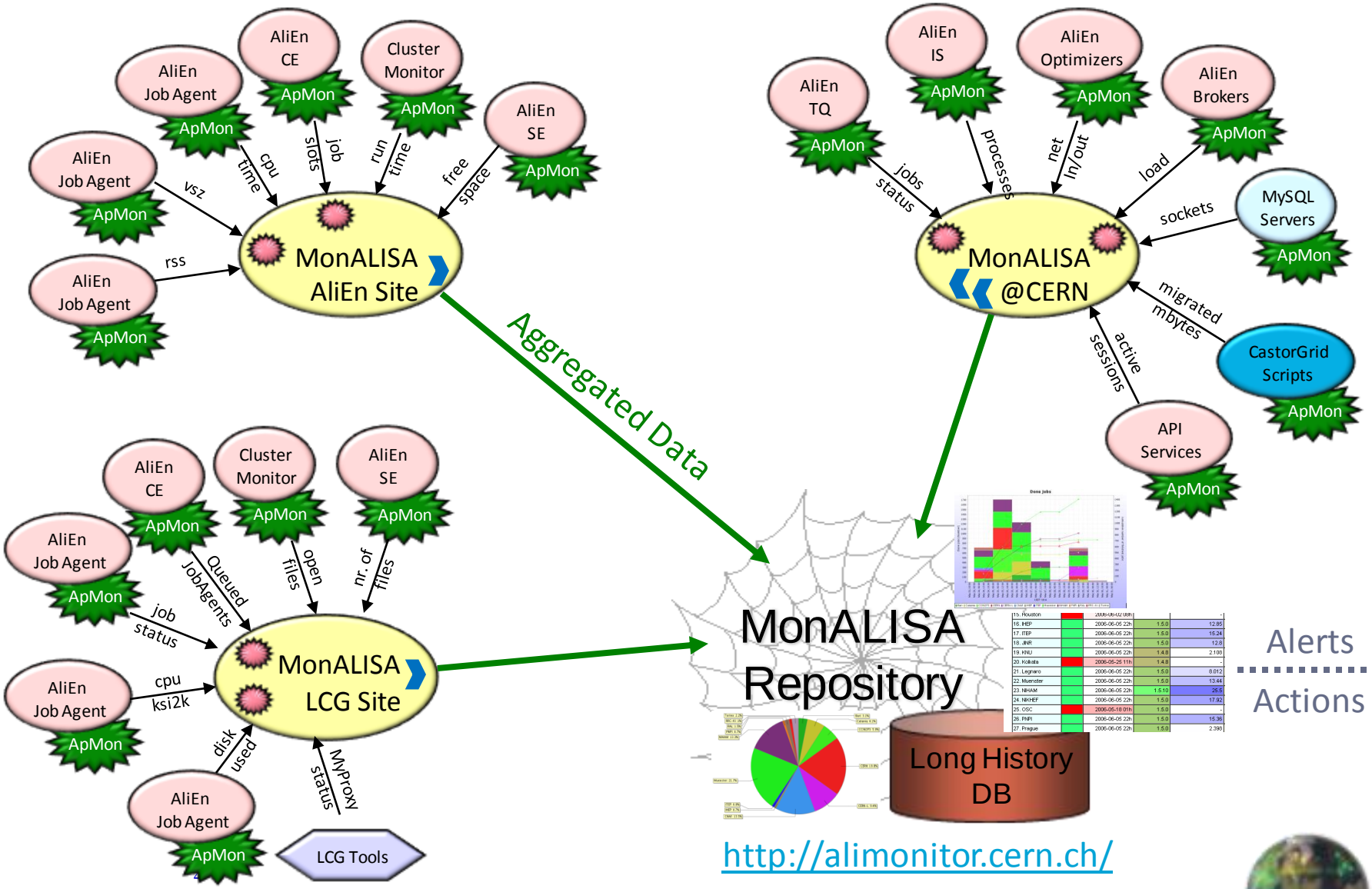
Multiplexing layer
Helps firewalled endpoints connect

Data gathering services

Registration and discovery

Fully Distributed System with no Single Point of Failure

Monitoring follows the general AliEn deployment layout: one service per site collects and aggregates site-local monitoring information



The **Service** is the data collecting entity that runs on each VoBox to gather and aggregate local data

- Many available modules that listen for / poll data
  - Local host monitoring (CPU, memory, network traffic , processes and sockets in each state, LM sensors, APC UPSs), log files tailing
  - SNMP generic & specific modules;
  - Condor, PBS, LSF and SGE (accounting & host monitoring), Ganglia
  - Ping, tracepath, traceroute, pathload, xrootd
  - Ciena, Optical switches (TL1); Netflow/Sflow (Force10)
  - Calling external applications/scripts that output the values as text
  - XDR-formatted UDP messages (ApMon)
- In-memory buffer for recent data
  - Can also store persistently in a local database (not used in ALICE)
- Data aggregation filters
  - Creating high-level views like cluster-wide total traffic IN/OUT, number of processes in each state …
  - Derived data available to clients like the original stream
- Subscriber mechanism
  - Clients can ask for past data and/or subscribe to arbitrary cuts in the monitoring data stream and they are notified in real time of new data

- Lightweight library of APIs
  - C, C++, Java, Perl, Python
- Send any app-specific information to ML Service(s)
  - UDP/8884 (open XDR binary format)
- Flexible configuration
  - hardcoded in the app
  - configuration file or URL
    - Dynamic options reload while the app is running
- Very high throughput (50 KHz of parameters to a single service)
- ROOT wrapper as TMonaLisaWriter

- Background application monitoring
  - 10 parameters / PID
  - Used CPU & wall time, % of the machine CPU
  - Partition stats, size of workdir, open files
  - Memory usage (resident, virtual and %), page faults
- Background system monitoring
  - 70-80 parameters / host
  - Load, CPU, memory & swap usage
  - Network interfaces (in/out/IPs/errs)
  - Sockets in each state, processes in each state
  - Disk IO, swap IO

Site services collect data from all local components

- AliEn services, proxies' status, critical local directories on the VoBox
- Xrootd storage nodes (*ALICE::<SITE>::<SE>_xrootd_* ; XrdStatus*)
  - Machine parameters (CPU, load, memory, sockets, processes, network traffic, disk and swap IO)
  - Xrootd internal parameters and per transfer details
  - Storage space as seen by xrootd
- Job agents (*<SITE>_JobAgent*)
  - CPU and memory usage, payload job ID, status
- Jobs (*ALICE::<SITE>::<CE>_Jobs*)
  - Full machine parameters (same as above)
  - CPU and memory usage of the process tree itself (+ Si2K normalized values)
  - Owner and masterjob / subjob IDs
  - Payload status code (STARTING, RUNNING, SAVING)
- Available bandwidth, traceroute/tracepath between services

Values are aggregated in various categories (**_Summary*)

- Job summaries
  - Number of jobs in each state (absolute and rates)
  - min/max/avg/total resource usage (absolute and rates)
  - Per cluster and per user
  - Top memory consumers
- Job agent overview
  - Number of JAs in each state
  - Average TTL
  - Queued JAs (that don't report yet active monitoring data)
  - Histograms of number of executed jobs per JA
- Aggregated Xrootd network traffic
  - per IP class, target site name, LAN/WAN, absolute total
- Cluster worker nodes' summary status
- Xrootd and PROOF cluster summary

- You can recycle the VoBox ML instance for your own purposes
  - Sending extra data to it from local services or adding modules to it
  - For example cluster monitoring with http://monalisa.cern.ch/MLSensor/
  - Or an ApMon-based host or application sensor
- Start a separate, independent service in the "*alice*" group
  - In case you want to deploy custom filters / alarms
- Or start your own monitoring group
  - For several sites / redundant monitoring
  - Services and clients can belong/connect to several groups
- Any number of consumers can subscribe to any cut in the data stream
  - But try to find the minimal expression that gives the data you want (>100KHz of monitoring data for 8M+ parameters)

Two important clients

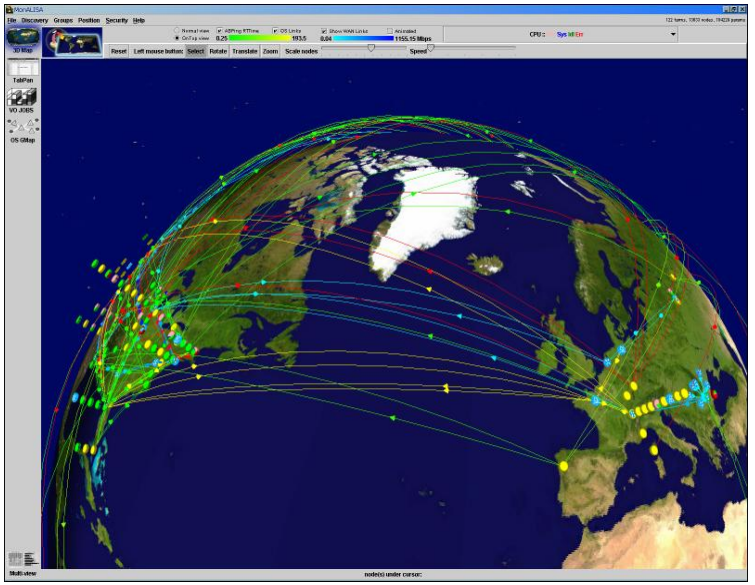
- GUI client

Interactive exploring of all the parameters

Can plot history or real-time values

Customizable history query interval

Subscribes to those particular series and updates the plots in real time

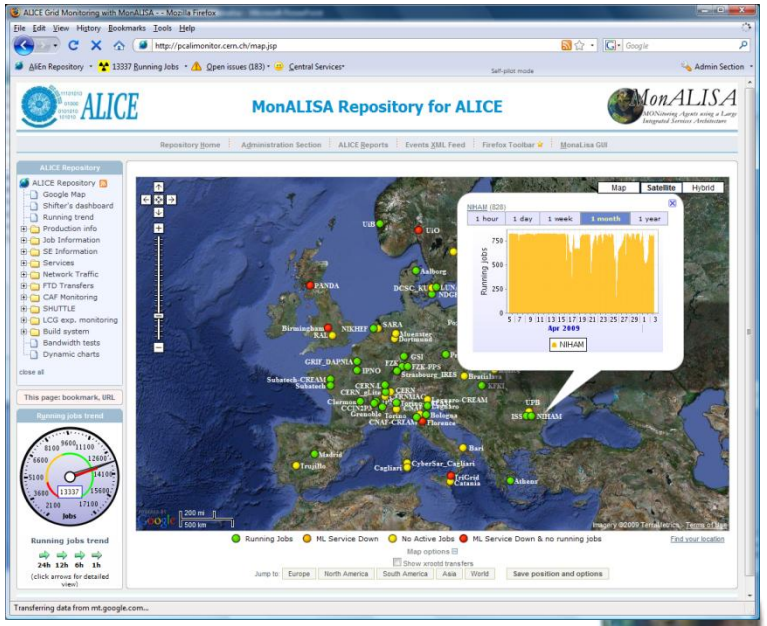


- Storage client (aka Repository)

Subscribes to a set of parameters and stores them in database structures suitable for long-term archival

Is usually complemented by a web interface presenting these values

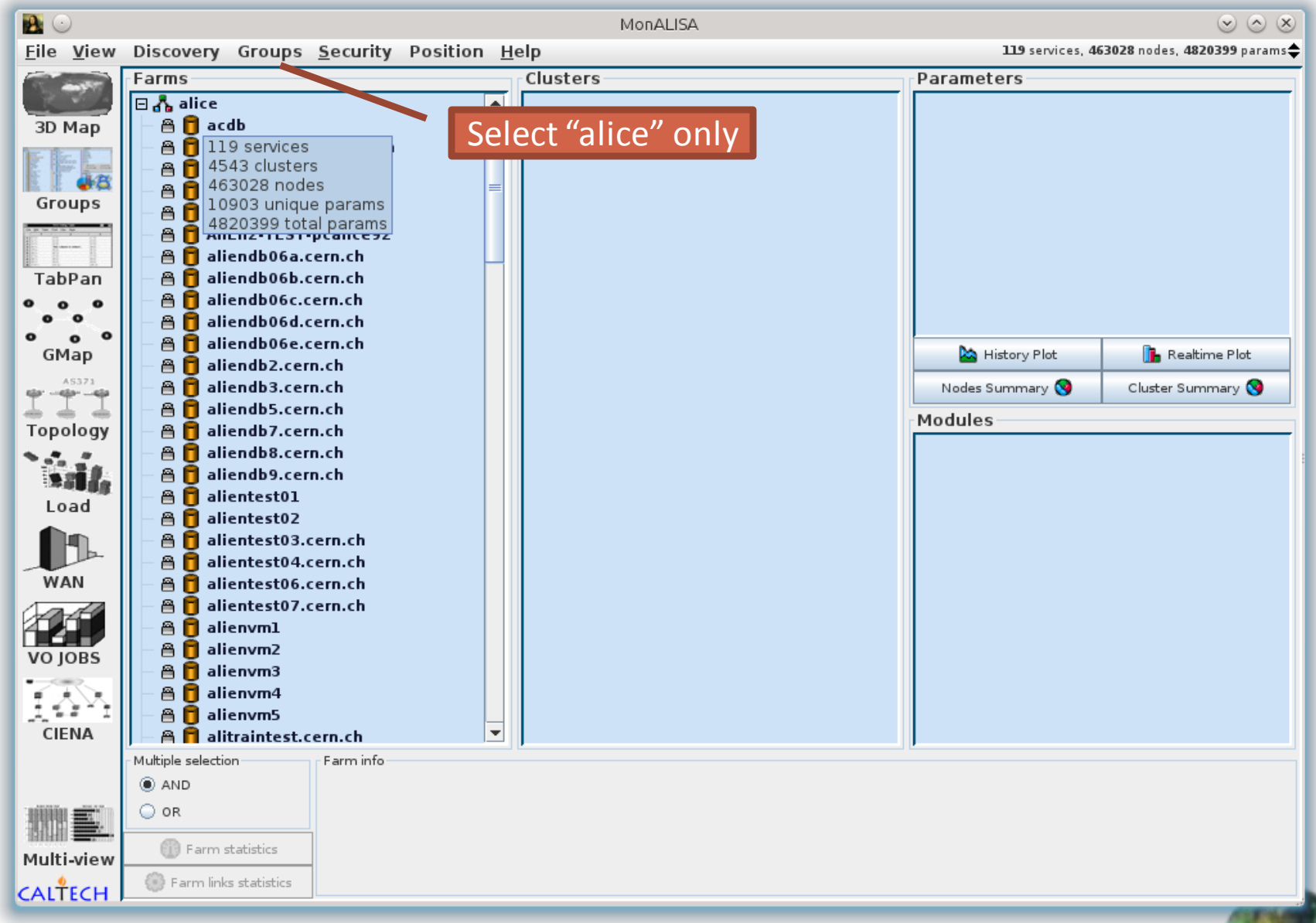
Can also be embedded in another controlling application



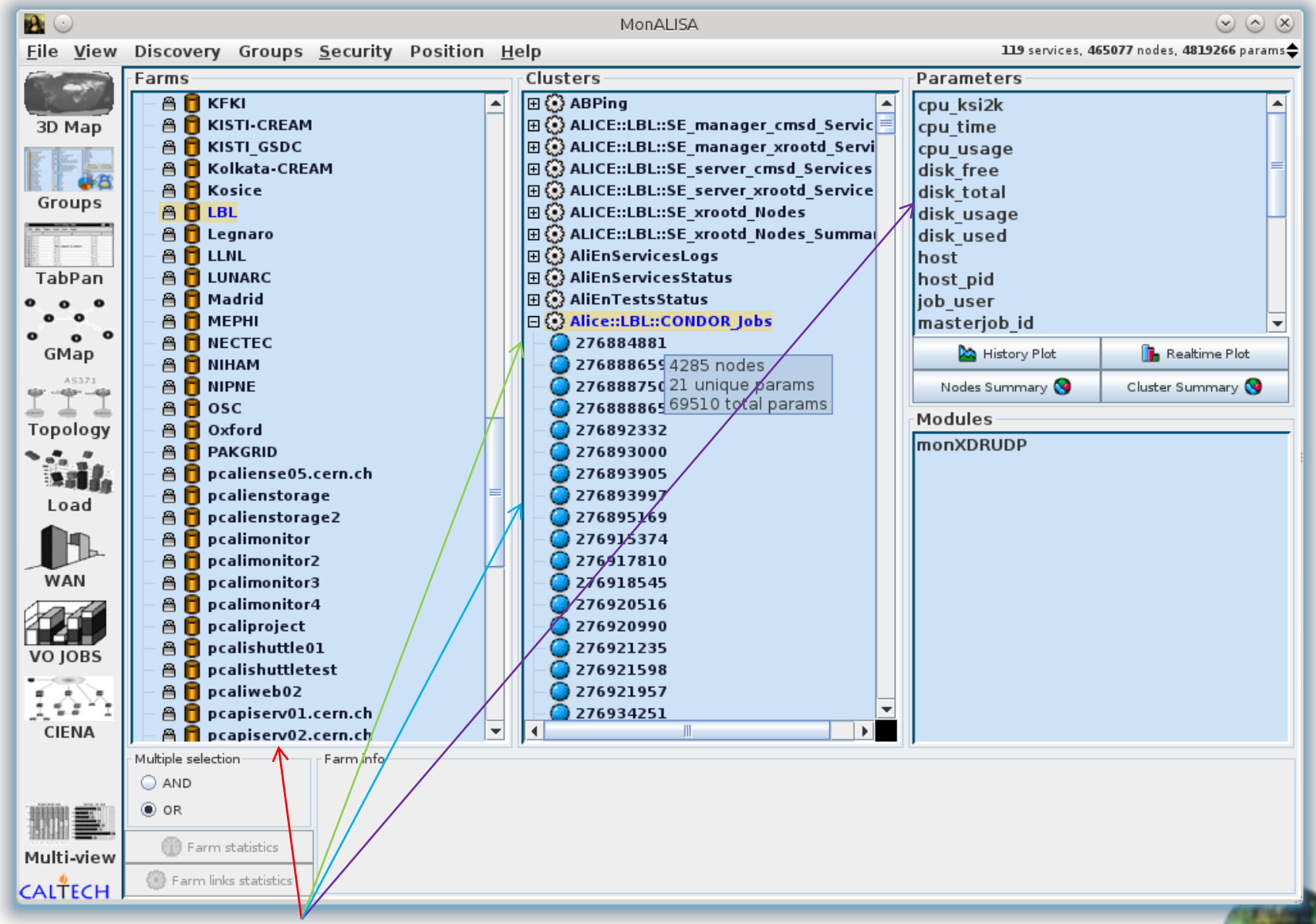
- WebServices & REST clients

Limited functionality: they lack the subscription mechanism

http://monalisa.cern.ch/monalisa__Interactive_Clients__MonALISA_client.html

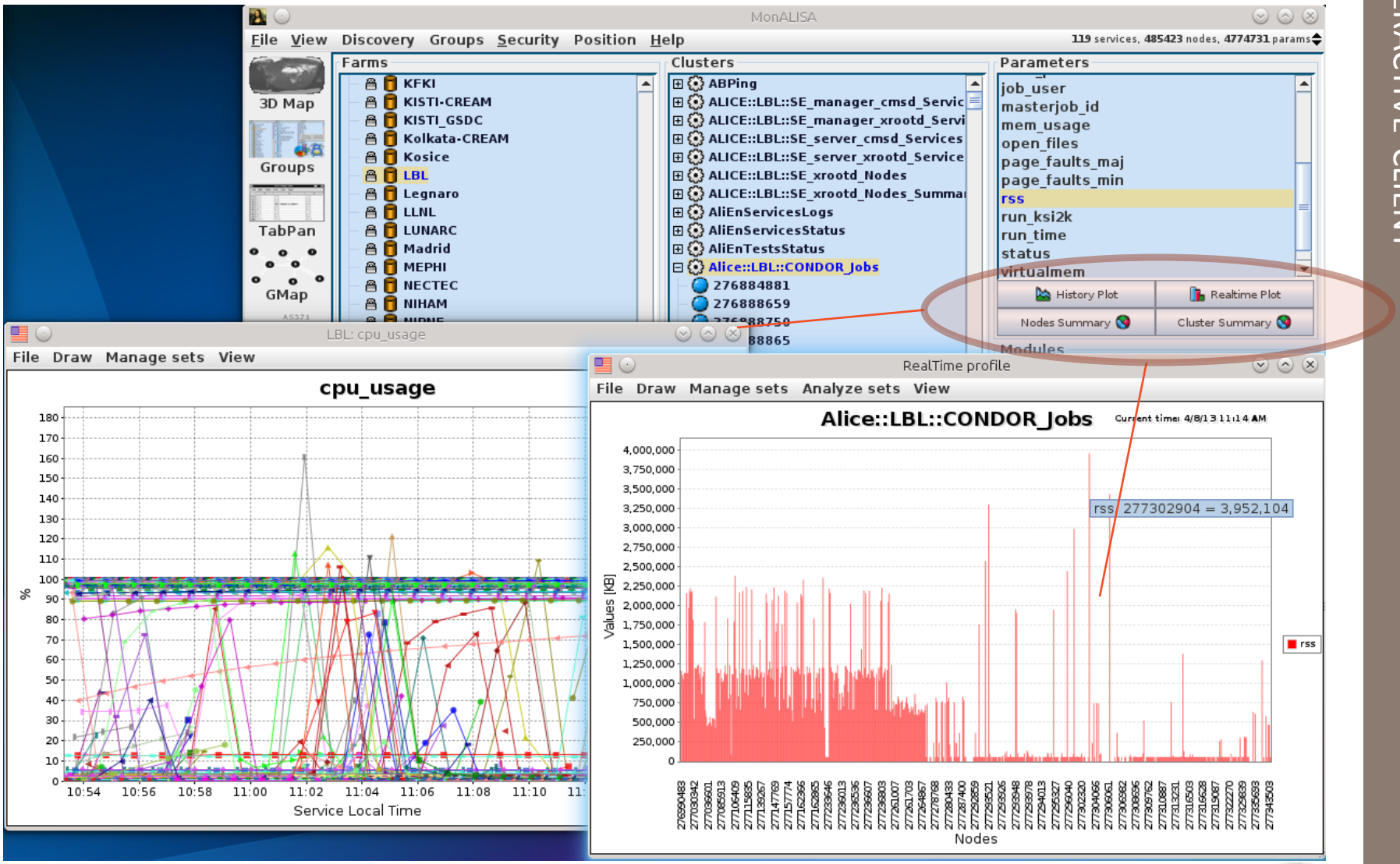


Browsing parameters

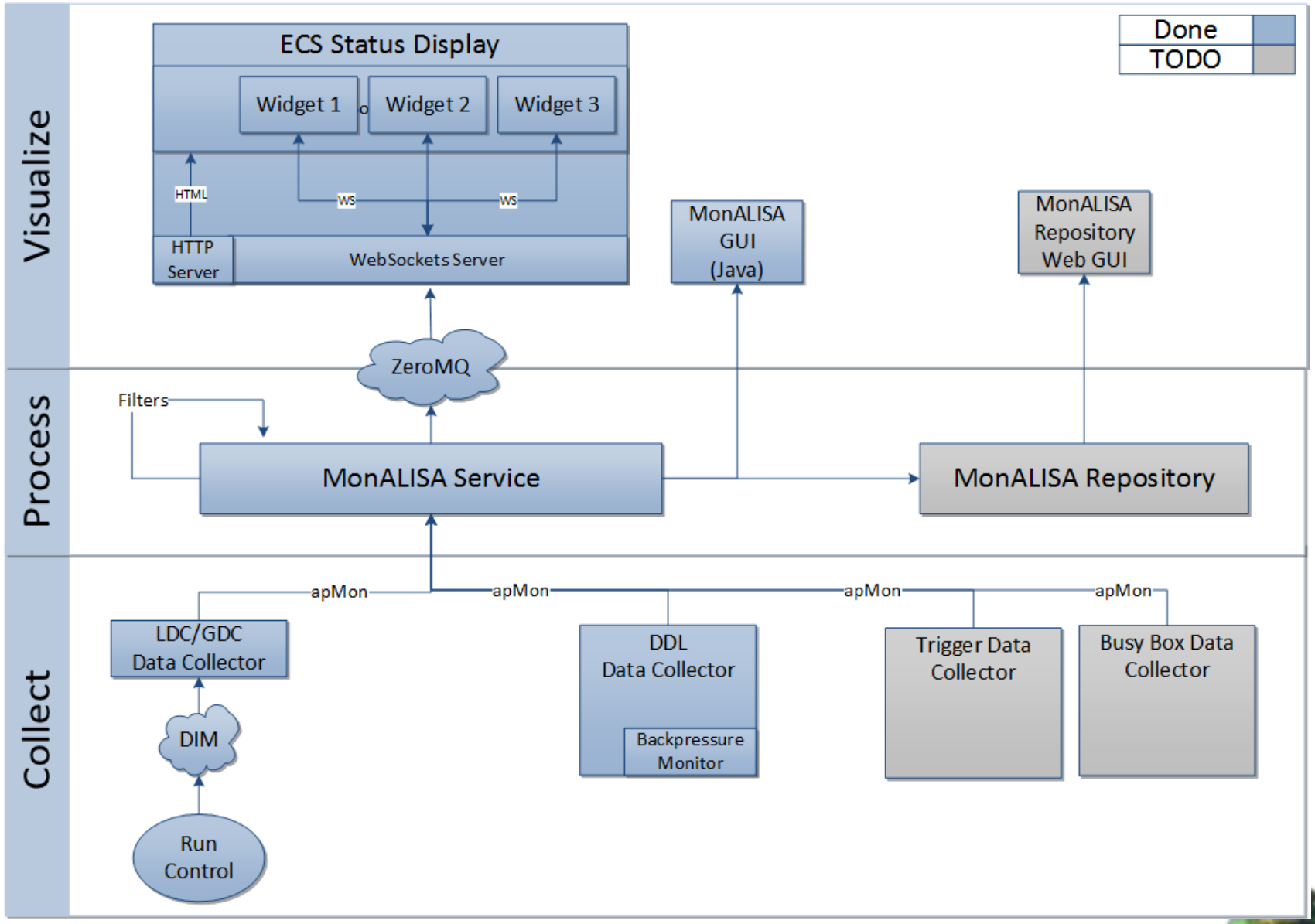


4 hierarchical levels of parameters (Farm, Cluster, Node, Function)

Dynamic views



- Base for the http://alimonitor.cern.ch/ service
- Single package including
  - Headless client
  - Apache Tomcat
  - PostgreSQL database
- Web interface includes examples of dynamic views
  - History charts
  - Real-time bar plots
  - Pie, spider, histograms
  - Status tables
    - Most views generated with one *.properties* file
- With this foundation you can build a custom repository
  - Dynamic pages (JSP or servlets)
  - Other plots with the included JFreeChart library
  - Data aggregation filters
  - Alarms, actions

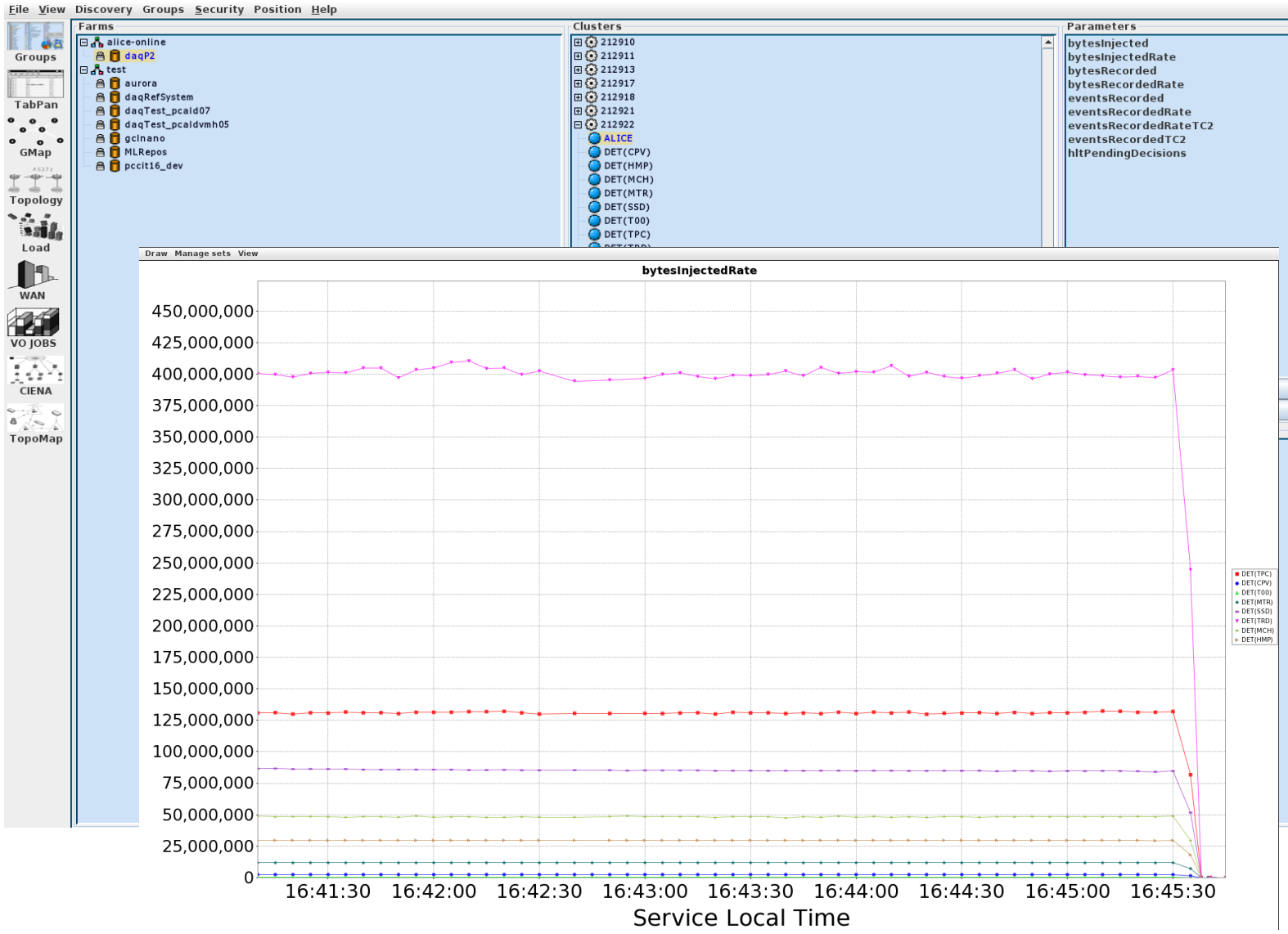

Visualize
Process
Collect
ECS Status Display
Widget 1
Widget 2
Widget 3
HTML
WS
WS
HTTP Server
WebSockets Server
Done
TODO
MonALISA GUI (Java)
MonALISA Repository Web GUI
ZeroMQ
Filters
MonALISA Service
MonALISA Repository
apMon
apMon
apMon
apMon
LDC/GDC Data Collector
DIM
Run Control
DDL Data Collector
Backpressure Monitor
Trigger Data Collector
Busy Box Data Collector

Continuously updating web interface, real time data

Run N°	Beam	Partition	Run type	HLT	Rec	Duration	Events
212370	-	PHYSICS_2	TECHNICAL	A	N	01:03:28	3.3M
212371	-	PHYSICS_1	PHYSICS	A	Y	00:22:00	105k

Calib	Bsy	Bck	Name	RUN	TC 1	TC 2	TC 3	TC 4	TC 5	TC 6	TC 7	TC 8
	-		-	105k	104k	44k	100	-	-	-	-	-
-	-	-	ACO	✓	✓	✓	-	-	-	-	-	-
PEDESTAL...	-	-	CPV	-	-	-	-	-	-	-	-	-
-	-	-	EMC	-	-	-	-	-	-	-	-	-
-	-	-	FMD	-	-	-	-	-	-	-	-	-
-	-	-	HMP	✓	✓	✓	-	-	-	-	-	-
-	-	-	MTR	-	-	-	-	-	-	-	-	-
-	-	-	MCH	-	-	-	-	-	-	-	-	-
-	-	-	PHS	-	-	-	-	-	-	-	-	-
-	-	-	PMD	-	-	-	-	-	-	-	-	-
-	-	-	SDD	✓	✓	✓	-	-	-	-	-	-
-	-	-	SPD	✓	✓	✓	-	-	-	-	-	-
-	-	-	SSD	✓	✓	✓	-	-	-	-	-	-
-	-	-	T00	-	-	-	-	-	-	-	-	-
-	-	-	TOF	✓	✓	✓	-	-	-	-	-	-
-	-	-	TPC	✓	✓	✓	✓	-	-	-	-	-
-	-	-	TRD	✓	-	✓	-	-	-	-	-	-
-	-	-	TRI	✓	✓	✓	✓	-	-	-	-	-
-	-	-	TST	-	-	-	-	-	-	-	-	-
-	-	-	V00	-	-	-	-	-	-	-	-	-
-	-	-	ZDC	-	-	-	-	-	-	-	-	-

Using the GUI for browsing the data

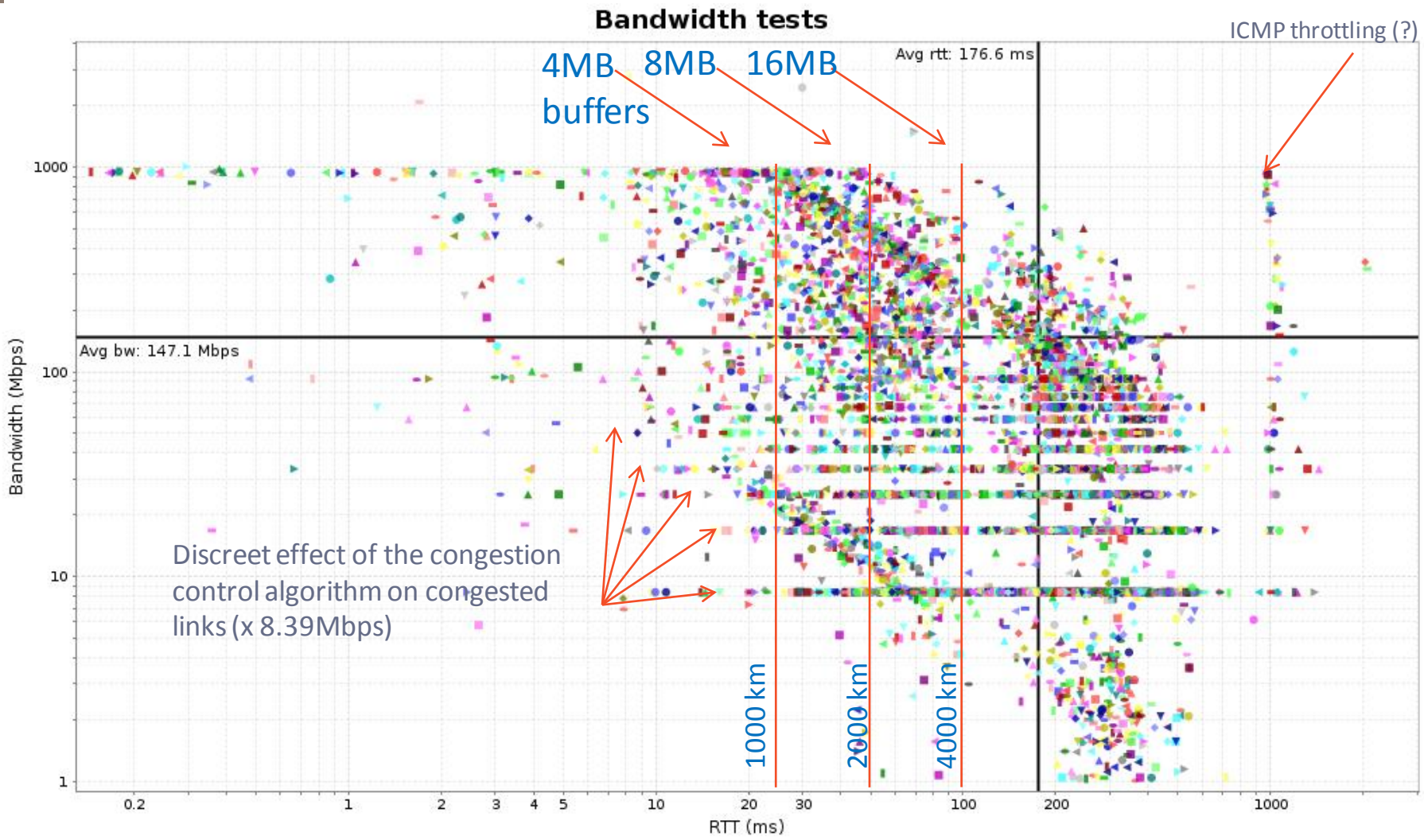


Base parameters for the discovery algorithm

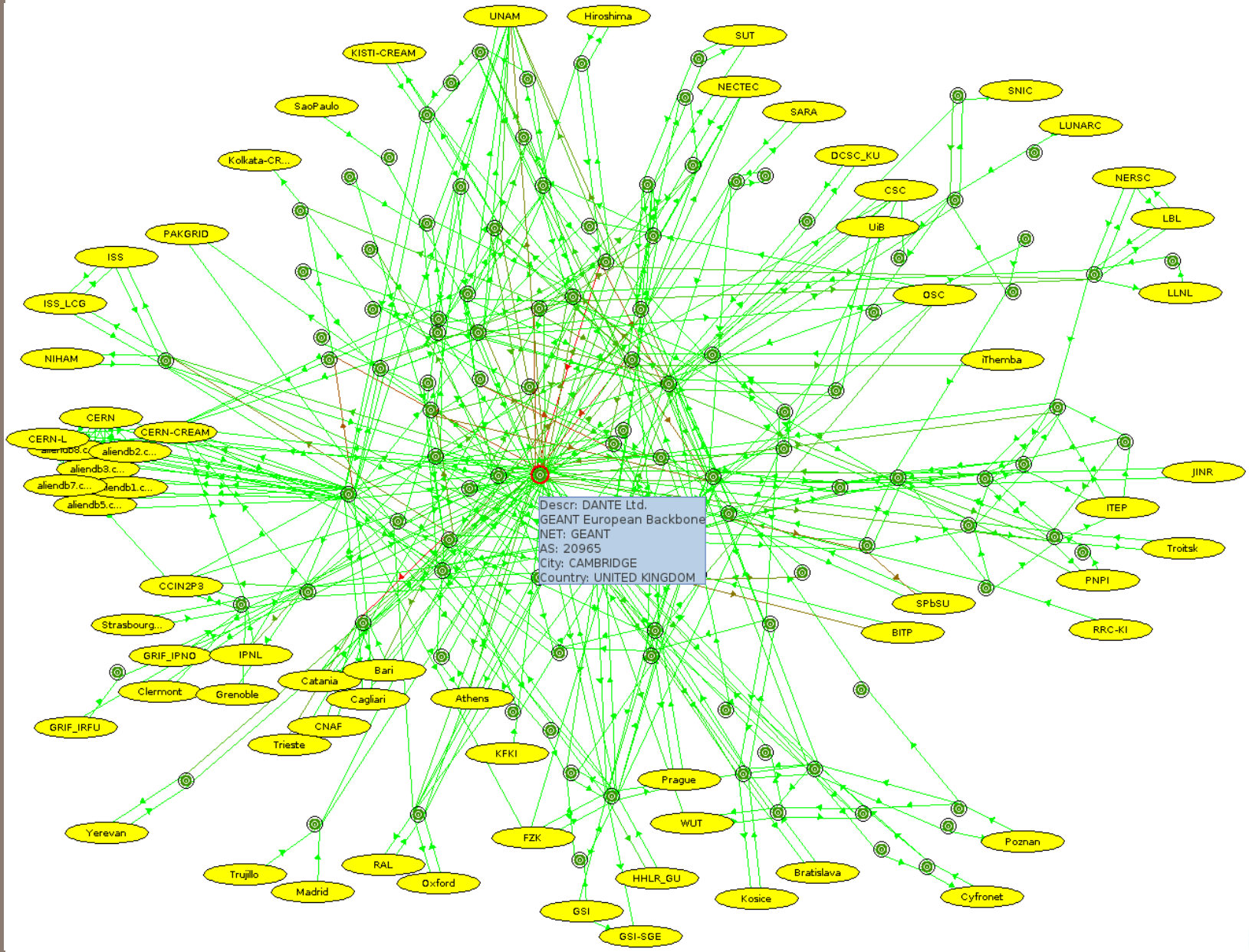
- Network topology
  - traceroute / tracepath between pairs of VoBox services
  - 1 stream available bandwidth measurements
- SE functional tests
  - Performed centrally every 2h, targeting the declared redirector
    - *add/get/rm* suite using the entire AliEn stack
    - Or just *get* if the storage is full
  - The dynamically discovered xrootd data servers are tested individually, with a simplified suite
  - Monitor discrepancies between declared volume and total space currently seen by the redirector
  - Above issues can be seen here
  - Plus many other related tests, like insufficiently large TCP buffer sizes

Online version of this plot: http://alimonitor.cern.ch/speed/chart1.jsp

- Recommended TCP buffer sizes: at least 8 if not 16MB



traceroute / tracepath aggregated at AS level



- Closest working replicas are used for both reading and writing
  - Sorting the SEs by the network distance to the client making the request
  - Combining network topology data with the geographical location
  - Leaving as last resort the SEs that fail the respective functional test
  - Weighted with their free space and recent reliability

- Writing is slightly randomized for more ‘democratic’ data distribution

Distance between any to IP addresses

- *distance*(IP, IP)
  - 0 Same C-class network
  - Common domain name
  - Same AS
  - Same country (+ f (RTT between the respective AS-es if known))
  - If distance between the AS-es is known, use it
  - Same continent
  - 1 Far, far away

- *distance*(IP, Set<IP>): Client's public IP to all known IPs for the storage (storage nodes, redirectors, VoBoxes near it…)

- Free space modifies the distance with

  f (ln(free space / 50TB))

- Storage-reported space usage has priority over the catalogue view on the space

- Recent history of *add,* resp. *get* contribute with

  *75% * last day success ratio +*

  *25% * last week success ratio*

- To all these a per-SE knob allows tuning to particular situations

  Isolated SEs that need to attract more data

  Avoiding SEs to be upgraded / decommissioned

- Data transfers
  - Still relying on Andreas' *xrd3cp*
  - Falling back to the plain *xrdcp* in/out
- Data deletion
  - AliEn should queue all physical deletes
  - In practice "dark" data creeps in
  - `*xrd ls*` is veeery slow, resync with catalogue in O(months)

```
Removed 6496568 files (63.05 TB), kept 14437736 files (509.3
TB), 49371 directories from ALICE::LBL::SE, took 89d 15:13
… ALICE::CERN::EOS, took 38d 16:08
```

  - Still cannot `*ls*` dCache SEs (tokens are not passed by the `*xrd*` 3.x cmd)
- SE incidents
  - Full or partial decommissioning
  - New hardware
  - Lost files
  - Handled on a case-by-case basis

Authen and SE keys expiring

- Current central services certificate expires Apr 25
- Is it enough to generate a new public key from the existing private one ?
- How to deploy it without affecting the running system ?
  - In sync on all SEs ?
  - A cron job watching an URL and acting on change ?
- When would be a good time to do this operation ?
  - We have 2 more months to plan, deploy and execute it
- A good opportunity to also upgrade Xrootd ☺

Questions ?

2Q||2Q ?