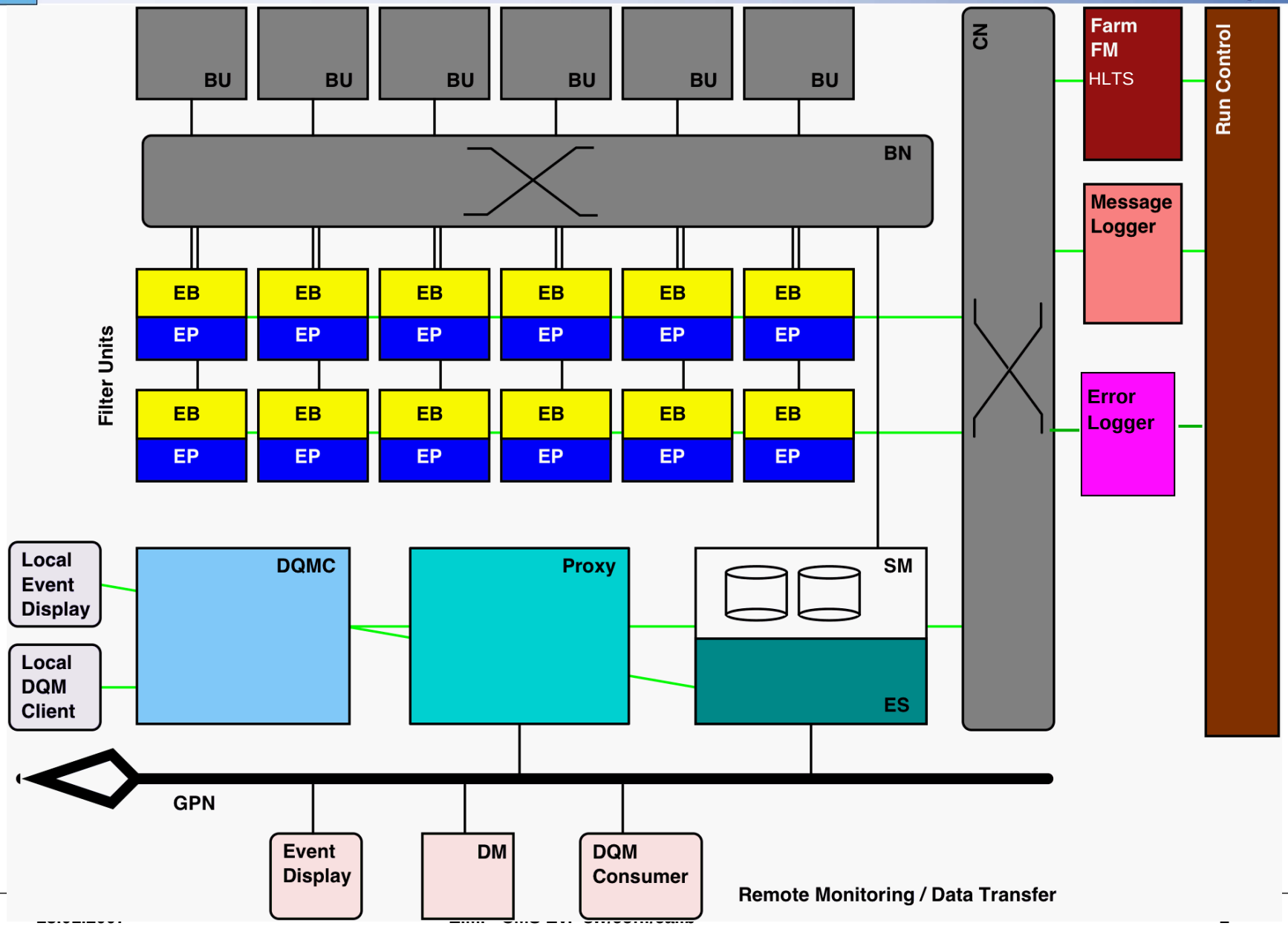
CMS Filter Farm

sw deployment and configuration

http://cern.ch/cmsevf
cms-daq-eventfilter@cern.ch

01/31/2006

E. Meschi – CERN PH/CMD


BU
BU
BU
BU
BU
BU
BN
CN
Farm
FM
HLTS
Run Control
Message
Logger
Error
Logger
EB
EP
Filter Units
Local
Event
Display
Local
DQM
Client
DQMC
Proxy
SM
ES
GPN
Event
Display
DM
DQM
Consumer
Remote Monitoring / Data Transfer

Event Filter

- Hw
  - ~ 4000 CPU/cores
  - ~ 4TB RAM
  - ~ 100 TB local disks
  - ~ 250 TB networked storage
  - ~ 6000 processes
- Sw (HLT only)
  - ~ 1000 packages
  - ~ 1/2 GB distribution
- Det
  - ~ 10^7 channels
  - ~ 100 MB calibration constants (HLT only)

HLT Sw Deployment

- HLT sw
  - Same code base as general offline reconstruction
  - Subset of packages
  - Subset of external
- Distribution
  - Cut from general release
  - Source distribution RPMs for development/testing/validation
  - Binary distribution RPMs
  - Externals from their distributions (lcg etc.)
- Deployment
  - Binaries (shared library pkgs + externals)
  - No reliance on source tree and/or additional data
  - Pkg = fully bound shared library: no reliance on loading order or pre-loading
  - Several versions must coexist on the system

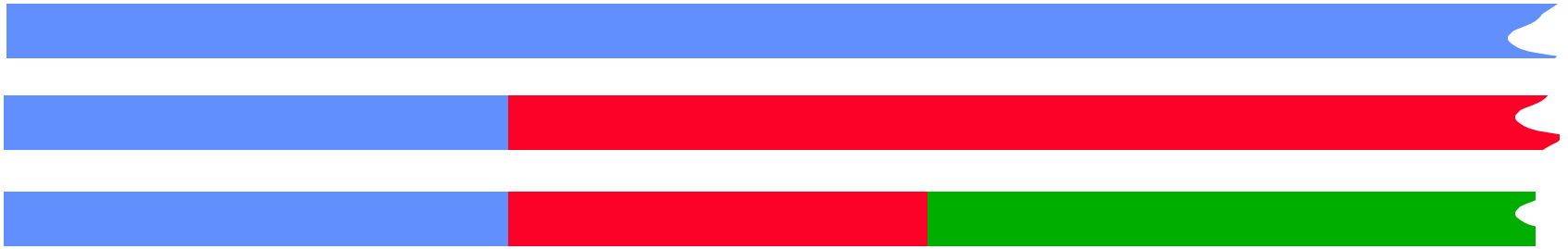
Sw Deployment Details

- RPM installation managed as part of DAQ farm sw management (see)
- All EvF sw components configured and controlled as part of the DAQ system
- Special services:
  - Calibration/conditions distribution and cache
  - Configuration DB
  - Data Management

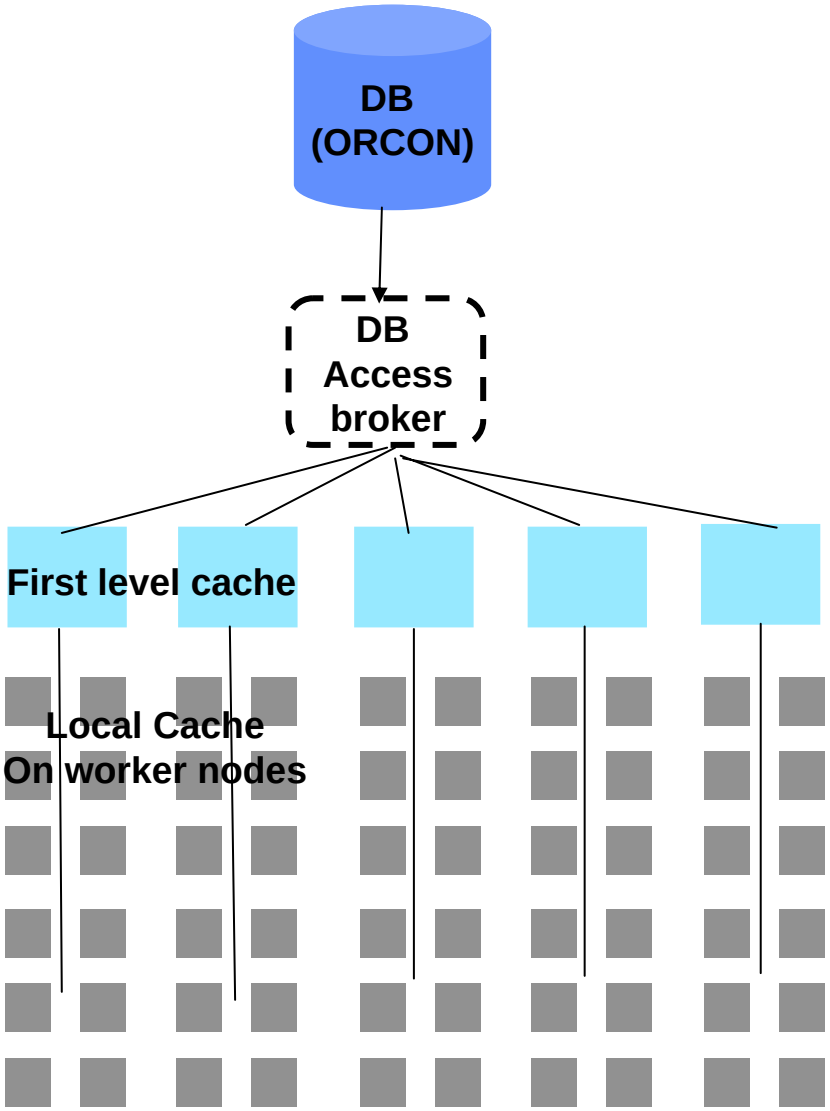
CondDB access

- Estimated O(100) MB calibration data required at “cold start”
  - Most of these data will change only rarely
  - Correct set known in advance and **assumed not to change within run**

  → Preloaded local cache
- CondDB access patterns
  - POOL-ORA (object-relational access, and LCG-AA product) for CondDB access from reconstruction code
  - Using Interval Of Validity (IOV)
  - “Intrinsic” IOV = use set with current IOV extending in the future



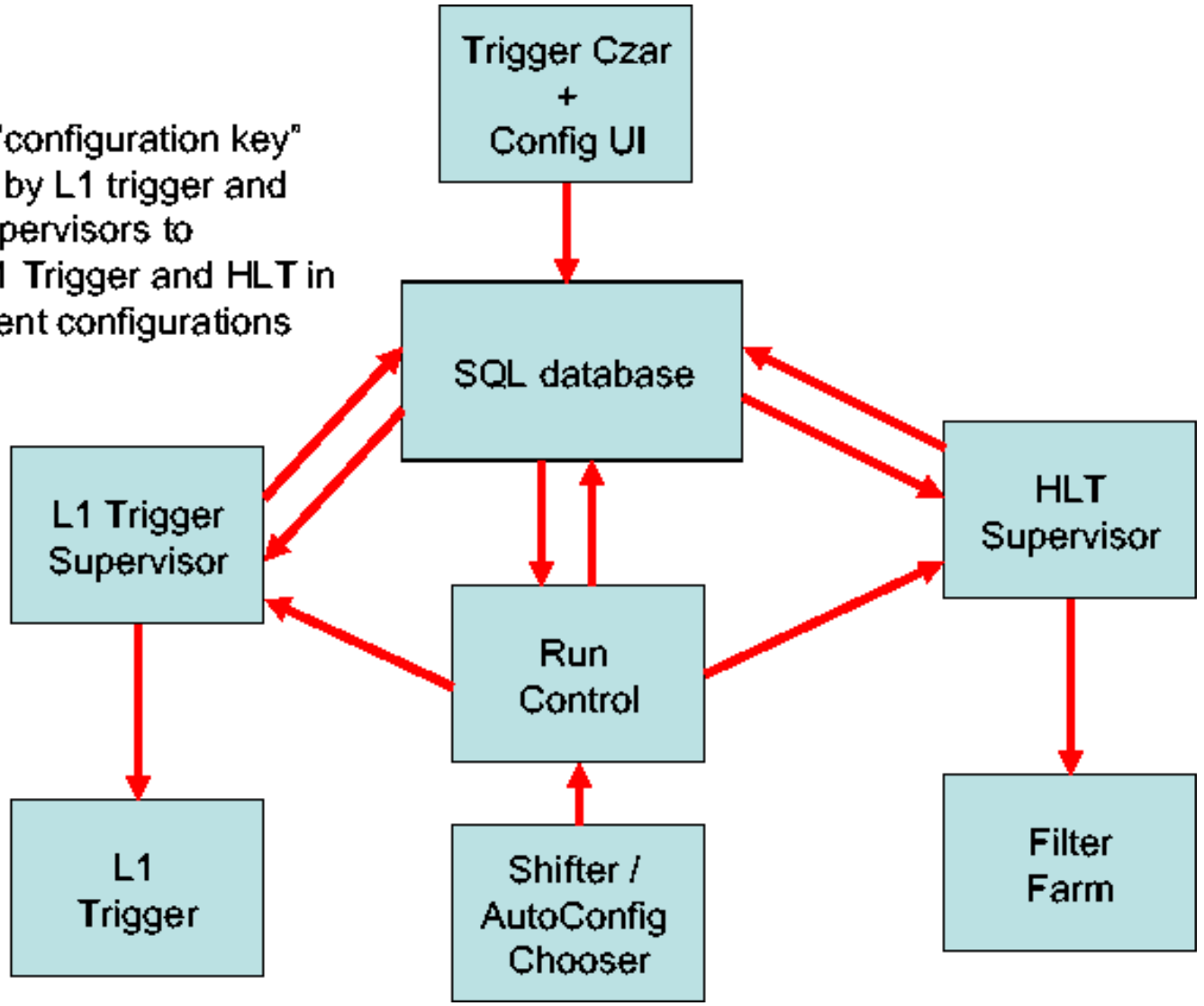
Condition Data Distribution



- Two-level cache to avoid network bottlenecks
- (Optional) access broker for better DB connection management
- Mechanism for cache invalidation
  - Always query IOV metadata
- Cache preload
  - Forced in between runs
- Implementation
  - E.g. with FronTier/Squid

HLT Configuration

Single "configuration key" utilized by L1 trigger and HLT supervisors to keep L1 Trigger and HLT in consistent configurations



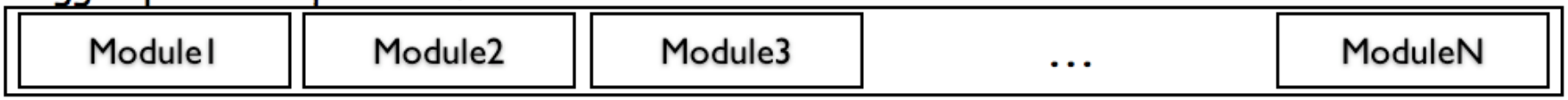
HLT ConfDB

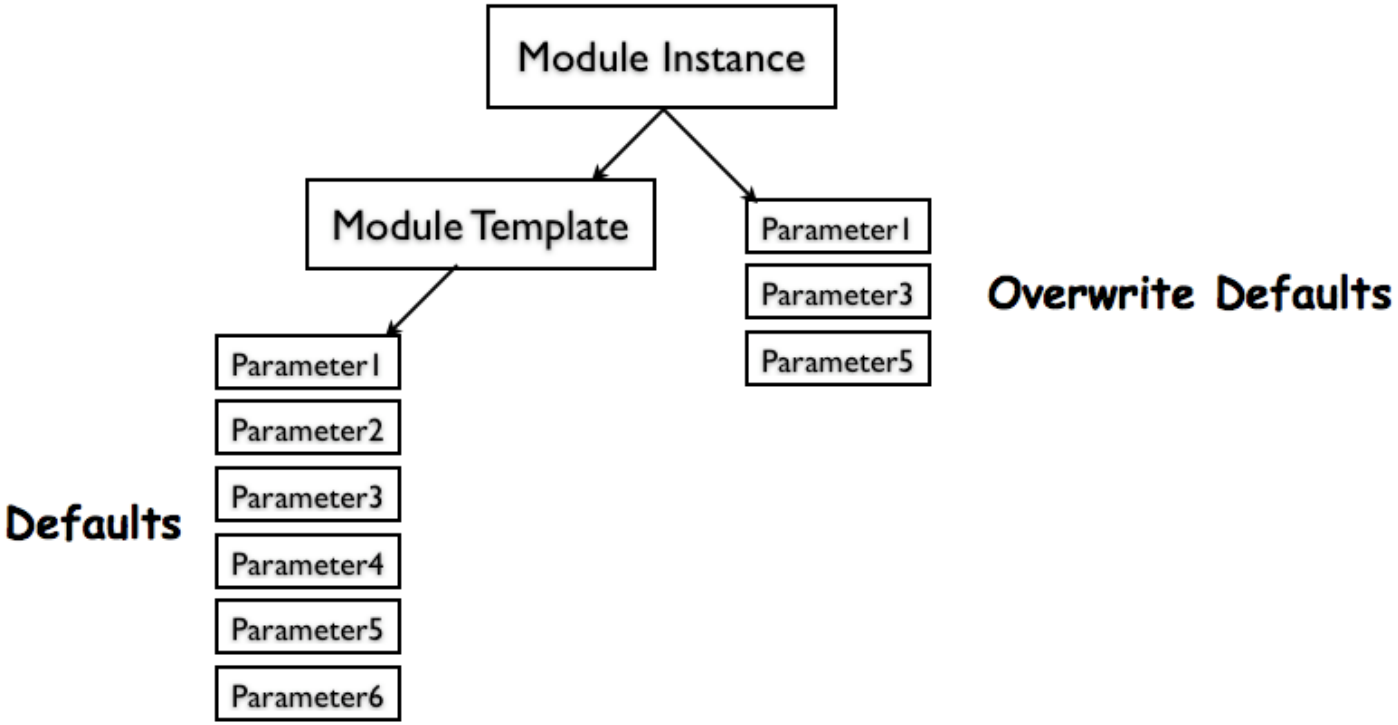
- Abstract components and rules of configuration grammar in a grammar-independent database schema.
- Configuration building blocks, services and modules, can change with each new software release.
  - The database schema needs to be able to cope with the history of these changes while minimizing the duplication of information:
    - Component templates have versions
    - One version of a component can be associated with multiple software releases
    - A trigger list created with one software version can be migrated to another
  - The Database should enforce data integrity where feasible (referential integrity, data types, etc.)
- Component templates are filled automatically from code alone
- Generation of configuration data is delegated to external components serving the “grammar of the day”

HLT Table: Abstractions

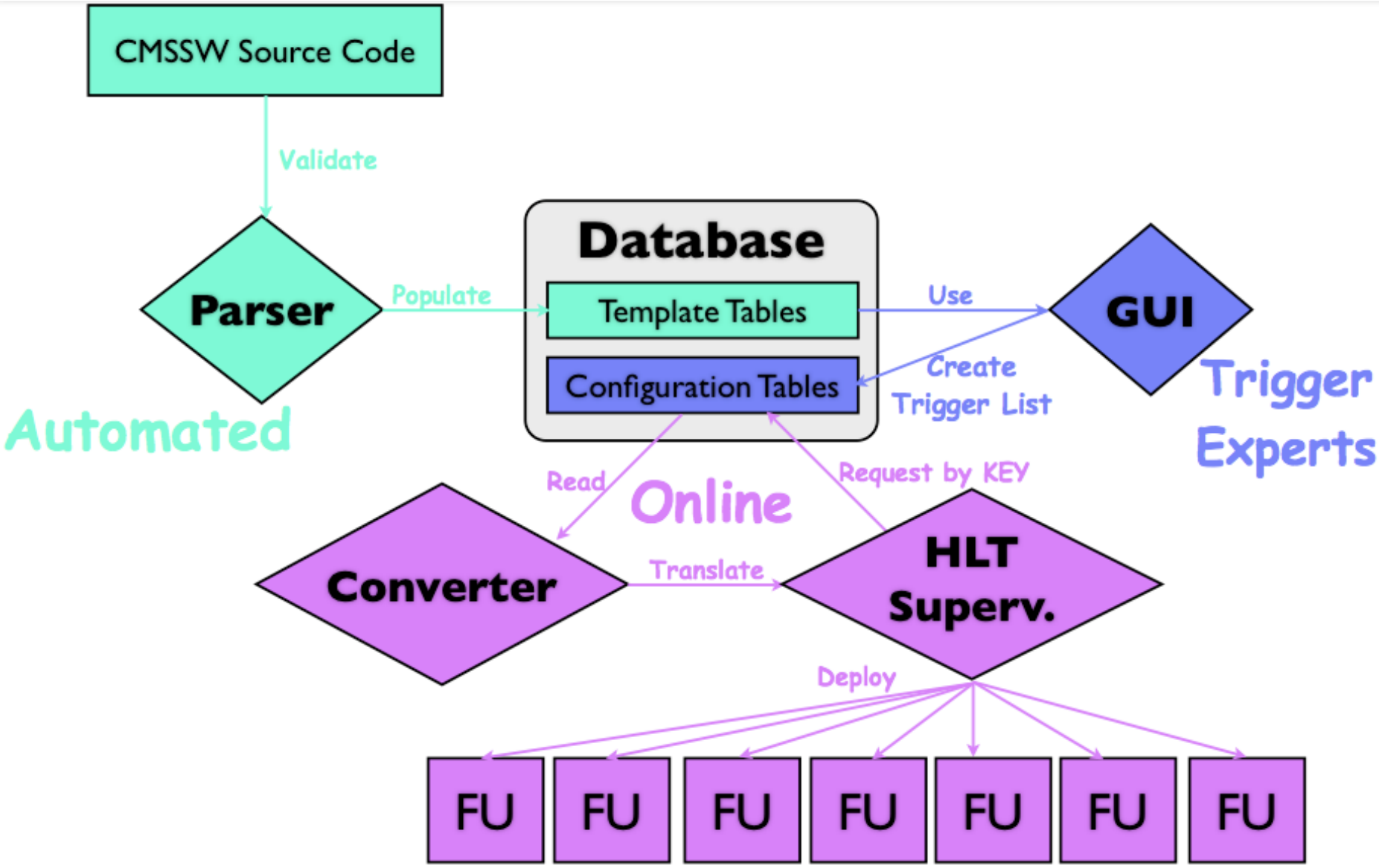
trigger list = sequence of trigger paths

trigger path = sequence of module instances





HLT Configuration Management



Database Schema: Components

Associate each parameterizable component with a superId which is unique among all component types!

superId = Parameters.parentId

SuperIds
superId: INT

ModuleTemplateSuperAssoc
superId: INT tmoduleId: INT

ModuleSuperAssoc
superId: INT moduleId: INT

ServiceSuperAssoc
superId: INT serviceId: INT

ServiceTemplateSuperAssoc
superId: INT tserviceId: INT

ModuleTemplates
tmoduleId: INT moduleTypeId: INT name:VARCHAR

Modules
moduleId: INT tmoduleId: INT configID: INT name:VARCHAR

Services
serviceId: INT tserviceId: INT configId: INT name:VARCHAR

ServiceTemplates
tserviceId: INT serviceTypeId: INT name:VARCHAR

ModuleTypes
moduleTypeId: INT moduleType:VARCHAR

Producer, Analyzer, Filter, ...

ServiceTypes
serviceTypeId: INT serviceType:VARCHAR

Service, EDSource, ESSource,...

DatabaseSchema: Components II

- Several References to the same Module Instance can appear in different paths
- A component (version) can be associated with one or many software releases
- A configuration can be associated with (be valid for) several sofware releases

Configurations
configId: INT config: VARCHAR

SoftwareReleases
releaseId: INT cvstag: VARCHAR

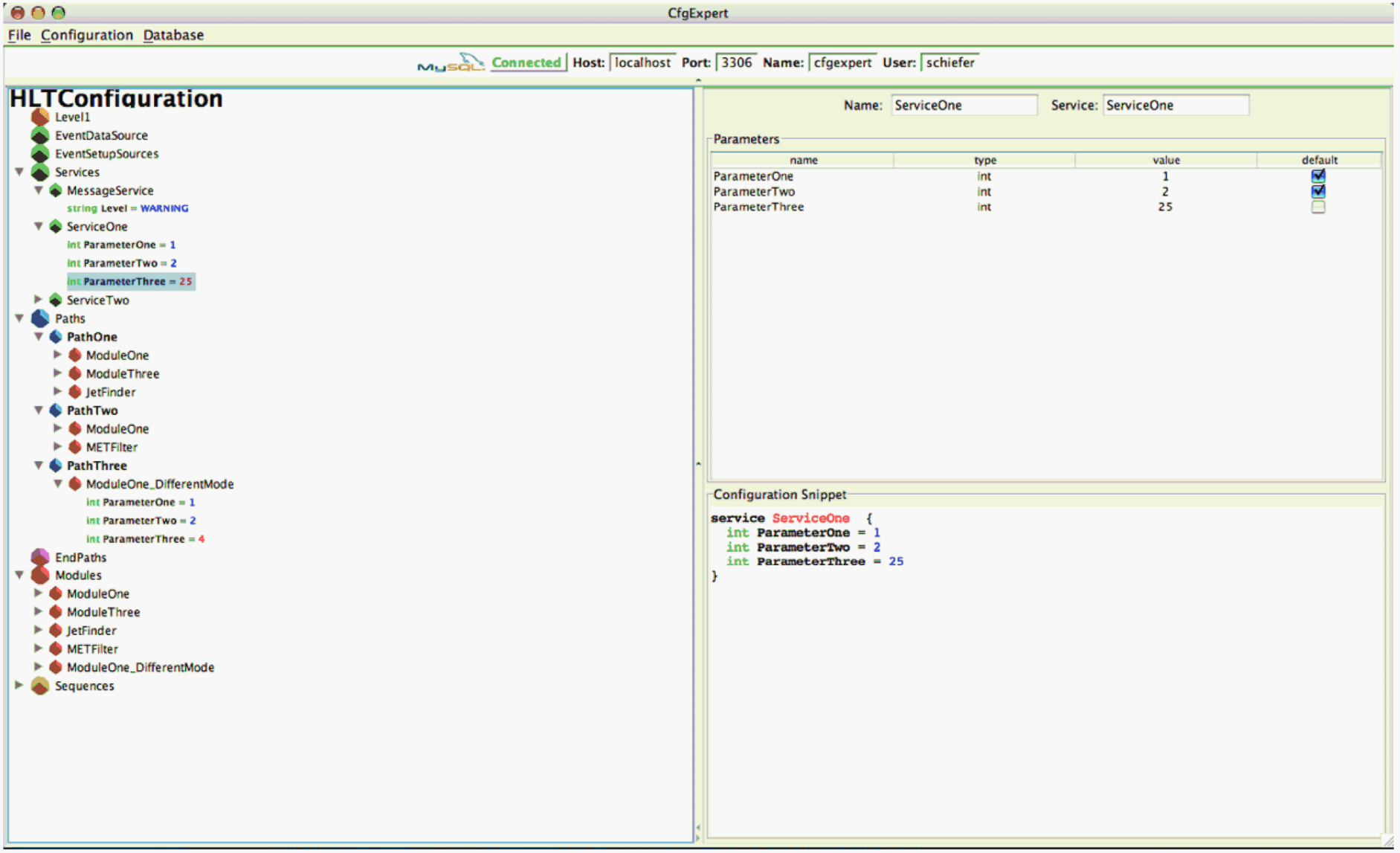
Paths
pathId: INT configId: INT name:VARCHAR sequenceNb: INT

ConfigReleaseAssoc
configId: INT releaseId: INT

SuperIdReleaseAssoc
superId: INT releaseId: INT

PathModuleAssoc
pathId: INT moduleId: INT sequenceNb: INT

Trigger List Creation Tool



Parser overview

- Given a release, parser will search for relevant modules/services and their parameters in CMSSW
  - Search all packages by default
  - CLI option for user to supply a list of packages to use, or a list of packages to ignore
- Written in python, w. DB interface via MySQLdb module
  - DB schema proposed by Phillip S. in Dec. meeting + followup emails
- Separation of text-parsing and DB interface tools - allow for future extensions to include
  - Parsing of .cf* or python-style configuration files
  - Interface to other DB technologies (e.g. Oracle)

Text parser operation

1. Search for module/service declarations (via DEFINE_ANOTHER_FWK_MODULE, DEFINE_ANOTHER_FWK_SERVICE, etc.)
2. Find each module's base class (HLTFilter, EDAnalyzer, etc.)
3. Find constructor and ParameterSet (both tracked and untracked parameters)
4. Search .cfi files for default values of tracked parameters
5. Record the CVS tag of this package

- Currently handles most (all?) "special cases": typedefs, casts, vector parameters…
- Parsing source tree w/o DB queries/loads takes O(20 seconds) for CMSSW_1_2_0

ConfDB converter

- Converter is part of HLT supervisor
- Embedded in HLT FunctionManager
  - Part of (HLT) run control system
- Tasks:
  - Gets configuration key from top level run control
  - Reads configuration according to key from database
  - Converts info tree into format understandable by filter process
  - Gives data to HLT supervisor to be forwarded to farm nodes

Implementation

- Language: Java
  - Has to run inside tomcat FM environment
- Database access: JDBC
- Using the same library which is used by gui

Open Issues

- Availability, and use of HLTConfDB outside online
- What, if any, information is to be migrated to CondDB
- Interface(s) with L1ConfDB (see later)

backup

Database Schema: Parameters

- A parameter has a parent component, a name, a type, and a value
- For type-safety and storage efficiency, we propose to keep all parameters in one table, and all parameter values in a table associated with the parameter type

Parameters
paramId: INT parentId: INT name:VARCHAR typedId: INT

ParameterTypes
typeId: INT name:VARCHAR

IntParamValues
paramId: INT value: INT

DoubleParamValues
paramId: INT value: REAL

StringParamValues
paramId: INT value:VARCHAR

...

VecIntParamValues
paramId: INT sequenceNb: INT value: INT

VecDoubleParamValues
paramId: INT sequenceNb: INT value: REAL

VecStringParamValues
paramId: INT sequenceNb: INT value:VARCHAR

...

DB Interface

- When the parser is run, a new template will be loaded if:
  - The list of parameters has changed (parameters added, removed, or values changed)
  - The package has a new CVS tag
- Wrapper functions for several common DB uses:
  - Check if module template exists (with a given CVS tag)
  - Create new template
  - Retrieve list of parameters and compare to list from parser

Status

- Database access: nearly ready
- Formatter: nearly ready
  - Final table will be checked by parser based on grammar in CMSSW
- Missing:
  - Filled database
  - Key from RC
  - Distribution mechanism

Plans

- Integrate converter into HLTS
- Work on distribution mechanism
  - Based on xdaq relays
  - On farm nodes: xdaq process puts data into named pipe
  - Event processor reads configuration data from pipe instead of file
- Time schedule?