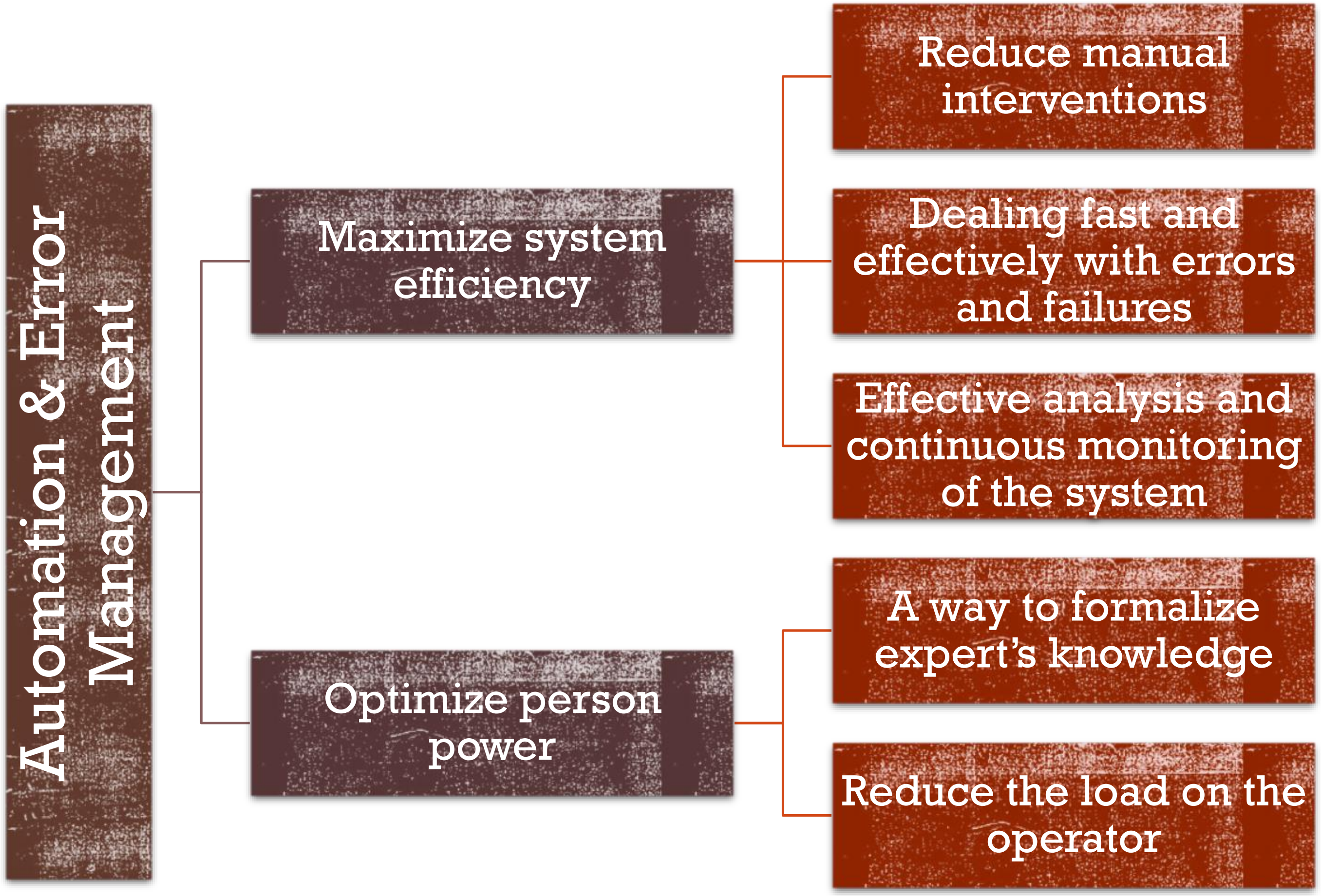
THE CENTRAL HINT AND INFORMATION PROCESSOR OF THE ATLAS TRIGGER AND DATA ACQUISITION SYSTEM

Giuseppe Avolio - CERN

OUTLINE

- **Automation and error management in the ATLAS Trigger and Data Acquisition (TDAQ) system**
  - Why?
- **Introducing a Complex Event Processing engine**
  - ESPER from EsperTech
- **The Central Hint and Information Processor (CHIP)**
  - Data sources and collection
  - Interaction with the Run Control
  - Performance
- **Conclusions and outlook**

WHY?



Automation & Error Management

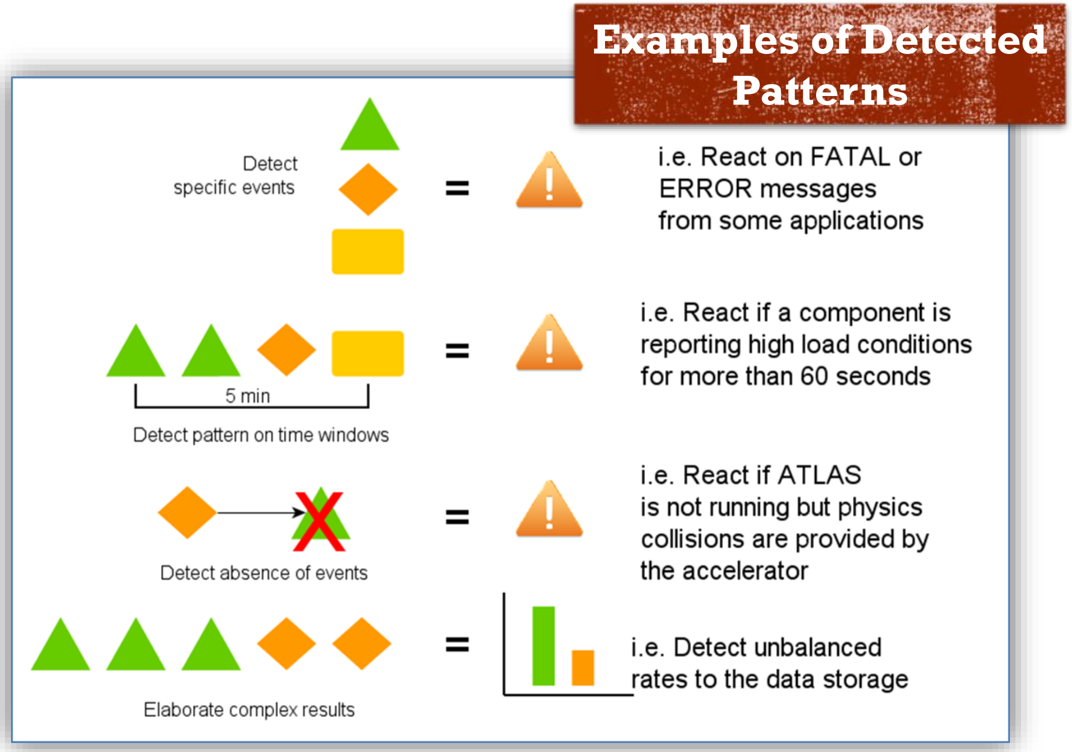
- Maximize system efficiency
  - Reduce manual interventions
  - Dealing fast and effectively with errors and failures
  - Effective analysis and continuous monitoring of the system
- Optimize person power
  - A way to formalize expert's knowledge
  - Reduce the load on the operator

COMPLEX EVENT PROCESSING

- **A set of technologies to process events and discover complex patterns among streams of events**
  - Used in financial analysis, wireless sensor networks, business process management
- **A cross between Data Base Management System and Rule Engines**
- **Main characteristics**
  - Continuous stream processing
  - Support for time/size windows, aggregation and grouping events
  - SQL-like languages (*streaming-SQL*) often used to query data streams
    - Augmented with constructs to express event relationships (time, cause and aggregation)
    - Streams replacing tables in a continuous evaluation model

A CEP ENGINE - ESPER

- **Open source, Java based**
  - Events as Java beans, XML documents, classes or key-value pairs
- **Support for advanced stream analysis**
  - Correlation, aggregation, sliding windows, temporal patterns
- **Knowledge base expressed in the Event Processing Language (EPL)**
  - Rich SQL-like language to express complex queries
- **Natively high-configurable multi-threaded architecture**
  - Inbound and outbound thread pools, timers
- **Support for historical data**
  - Full control over time!
- **Built-in advanced metrics**



Examples of Detected Patterns

i.e. React on FATAL or ERROR messages from some applications

i.e. React if a component is reporting high load conditions for more than 60 seconds

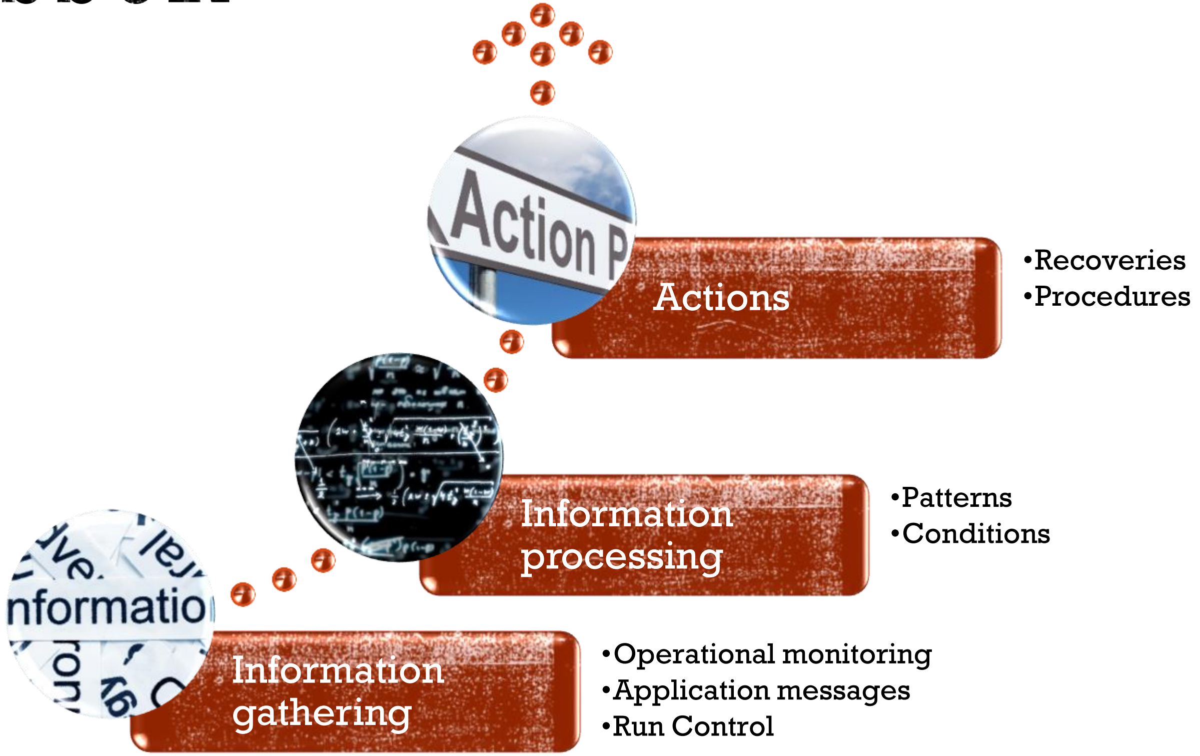
i.e. React if ATLAS is not running but physics collisions are provided by the accelerator

i.e. Detect unbalanced rates to the data storage

ESPER is available at http://www.espertech.com/esper/

CHIP: THE CENTRAL HINT AND INFORMATION PROCESSOR

- **CHIP is an "*intelligent*" application having a global view of the TDAQ system**
  - Supervises the ATLAS data taking
  - Takes operational decisions
  - Handles abnormal conditions
  - Automates complex procedures
  - Performs advanced recoveries
- **CHIP embeds the ESPER engine**

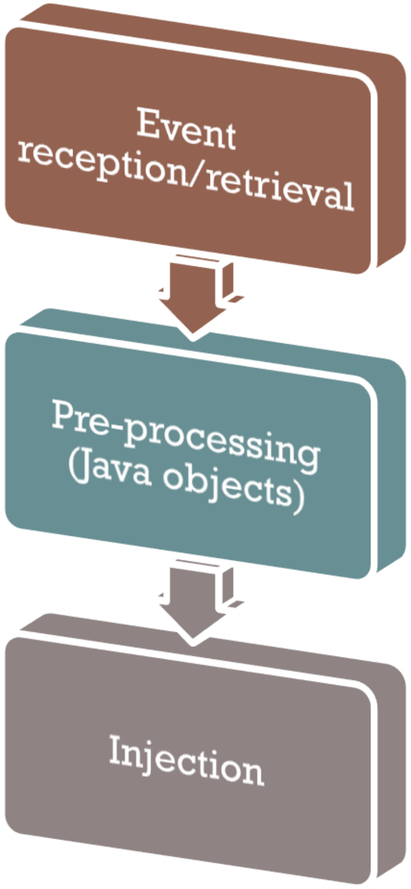


TDAQ system largely deterministic → Possible to identify "signatures" and react properly

CHIP DATA SOURCES

Typical information sources

Run Control	Process status
	Executed Commands
	Finite State Machine (FSM) status
Application Messages	Different severities
	Reporting anomalies
	Can trigger on-demand actions
Operational Data	LHC status
	Detector working parameters
	Run conditions and parameters
System Configuration	Enabled resources and detectors
	Application parameters
	Application dependencies



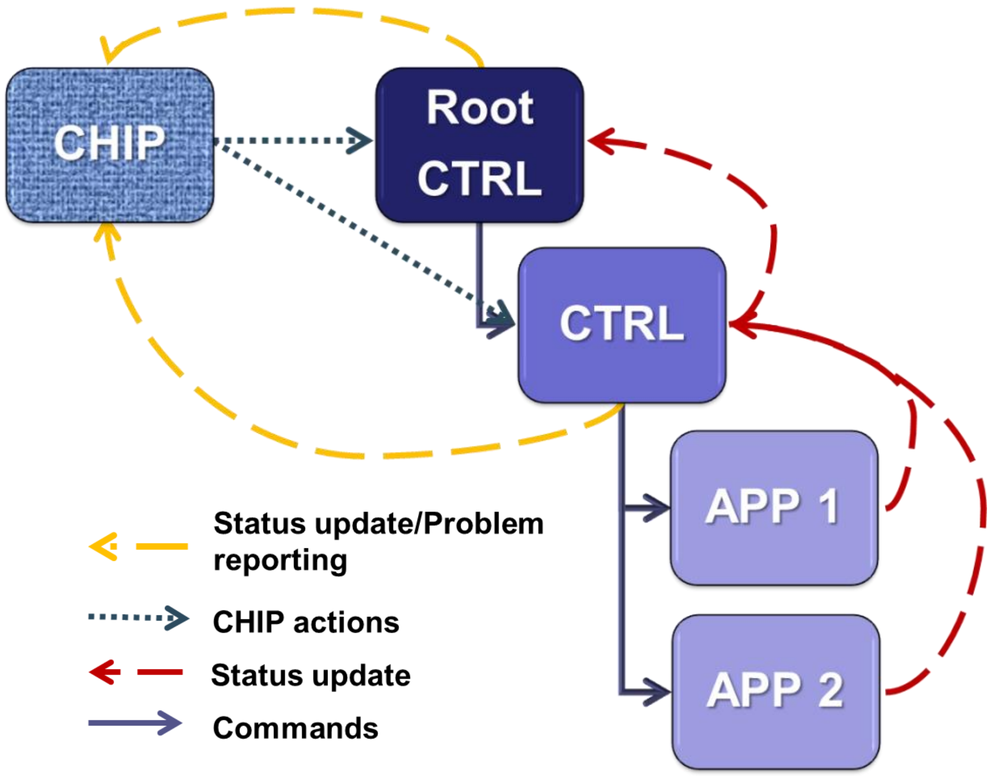
Event reception/retrieval

Pre-processing (Java objects)

Injection

CHIP AND THE RUN CONTROL

- **Applications in the Run Control (RC) are organized as a hierarchical tree**
  - *Controllers* are responsible of *leaf applications*
  - More than 100 *controllers*
- **CHIP is the "brain" of the RC system…**
  - Continuous monitoring of the state of *all* the applications (about 30000)
  - Detection and proper handling of any misbehaving application
- **…that, in its turn, is CHIP's "right hand"**
  - Execution of commands
- **Mission critical**
  - Status of the RC essential for safe data taking



CHIP: RECOVERIES AND AUTOMATION

- **Large knowledge base**
  - More than 300 EPL statements
    - 28 different contexts (*i.e.*, EPL modules)
  - Actions organized in 26 different categories
    - Each *action executor* being fully parametric
- **More than 50 different types of events (streams) concurrently handled by CEP engine**

Partial list

Core

- Run Control error management
- Dealing with application failures

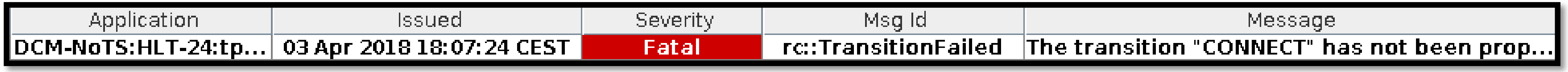
Recovery

- (Re)synchronization of detectors
- Removal and re-insertion of busy channels
- Full reconfiguration of detectors

Automation

- Autopilot: automatic cycling through the Run Control FSM states
- Setting ATLAS reference clock
- Moving to physics mode with stable beams
- Detector specific procedures

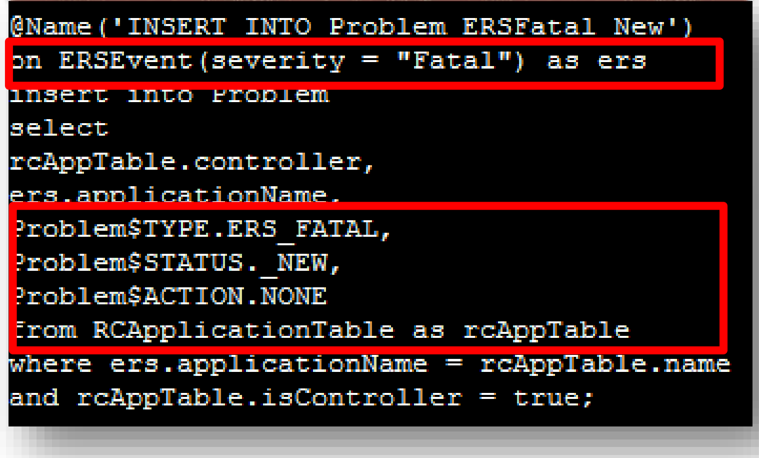
CHIP & EPL: AN EXAMPLE



Application	Issued	Severity	Msg Id	Message
DCM-NoTS:HLT-24:tp...	03 Apr 2018 18:07:24 CEST	Fatal	rc::TransitionFailed	The transition "CONNECT" has not been prop...

Injection into the ESPER engine as an ERSEvent

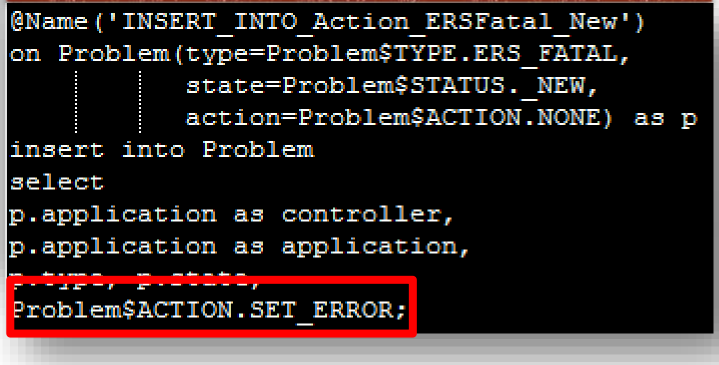
1 - Detection
Application sending a FATAL error



```
@Name('INSERT INTO Problem ERSFatal New')
on ERSEvent(severity = "Fatal") as ers
insert into Problem
select
rcAppTable.controller,
ers.applicationName,
Problem$TYPE.ERS_FATAL,
Problem$STATUS._NEW,
Problem$ACTION.NONE
from RCApplicationTable as rcAppTable
where ers.applicationName = rcAppTable.name
and rcAppTable.isController = true;
```

***ERS** = **E**rror **R**eporting **S**ystem*

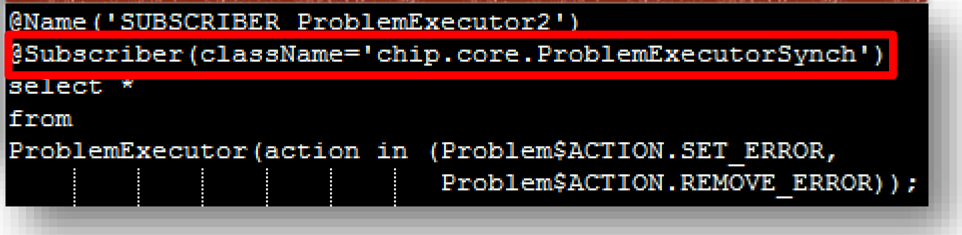
2 - Decision - Set error state



```
@Name('INSERT_INTO_Action_ERSFatal_New')
on Problem(type=Problem$TYPE.ERS_FATAL,
           state=Problem$STATUS._NEW,
           action=Problem$ACTION.NONE) as p
insert into Problem
select
p.application as controller,
p.application as application,
p.type, p.state,
Problem$ACTION.SET_ERROR;
```

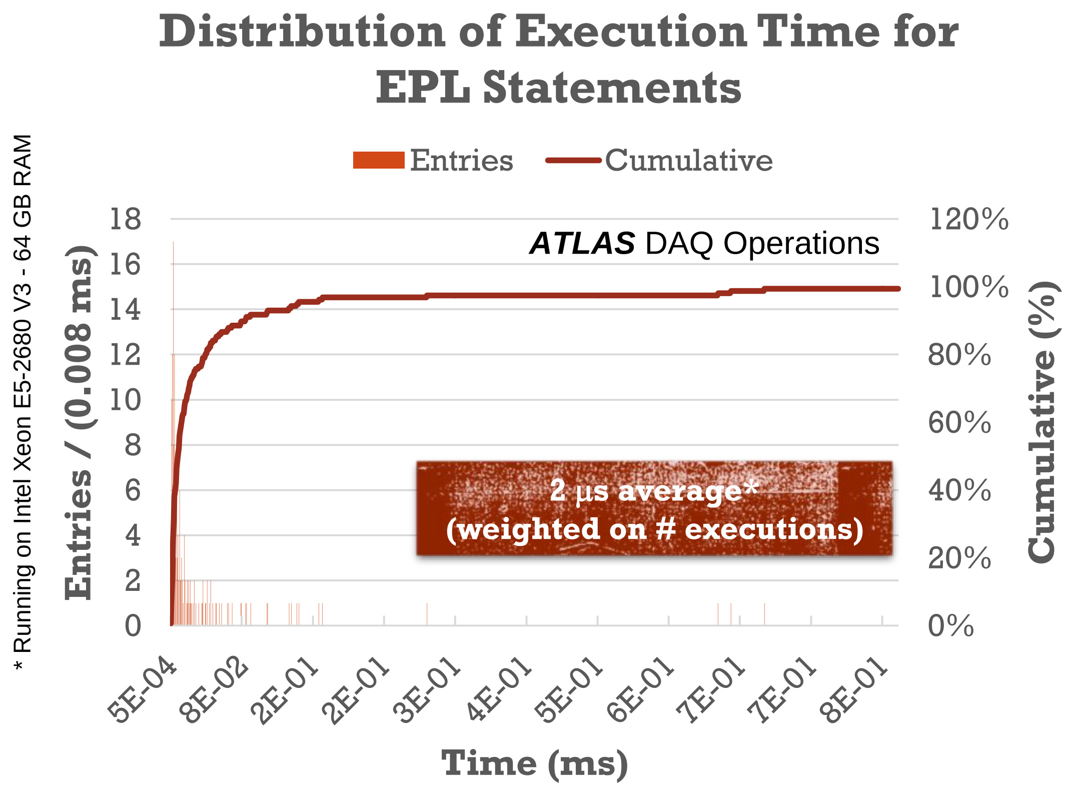
The executor is a Java object

3 - Execution - Call executor to send command



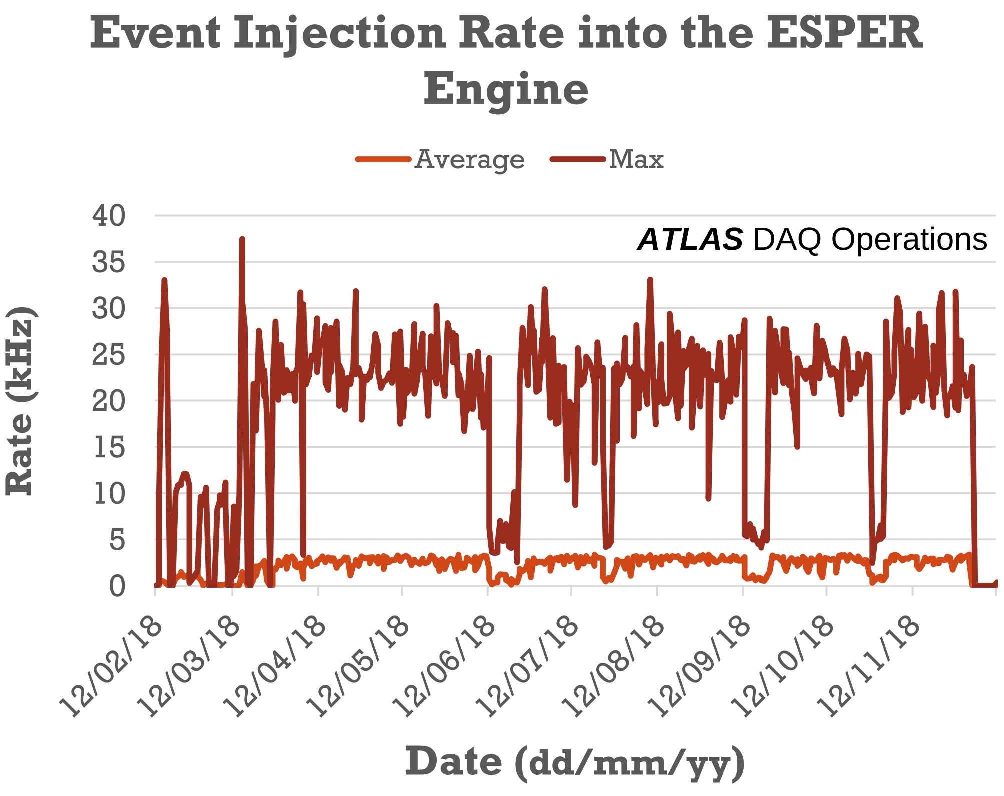
```
@Name('SUBSCRIBER ProblemExecutor2')
@Subscriber(className='chip.core.ProblemExecutorSynch')
select *
from
ProblemExecutor(action in (Problem$ACTION.SET_ERROR,
                           Problem$ACTION.REMOVE_ERROR));
```

CHIP – PERFORMANCES

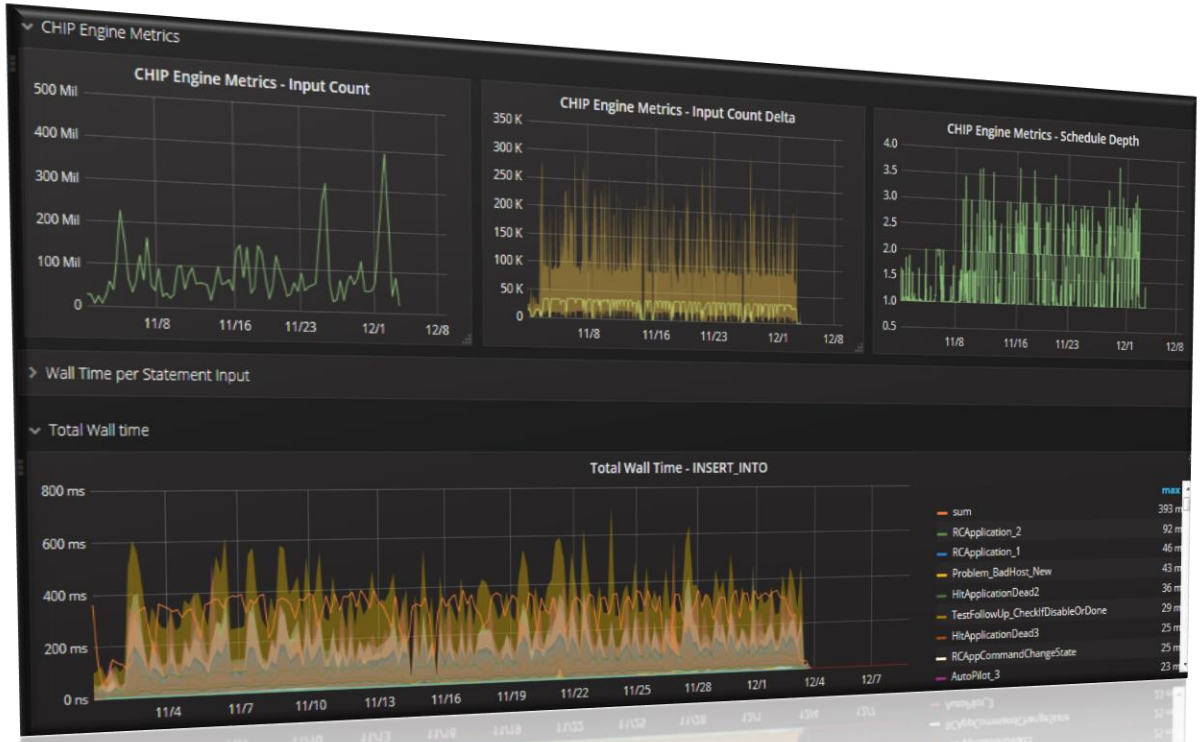


2 µs average*
(weighted on # executions)

* Running on Intel Xeon E5-2680 V3 - 64 GB RAM



CHIP: METRICS ANALYSIS



Grafana is available at https://grafana.com/

- **Exploiting ESPER's built-in metrics**
  - Detailed information for every single rule in the knowledge base
    - Wall and CPU execution times, number of times a statement is evaluated
- **Real-time and historical data**
  - Data pushed to a time series database and made available via *Grafana* dashboards
  - Fundamental feature to assess the state of CHIP and identify possible issues
- **Metrics reporting can be enabled/disabled at runtime**

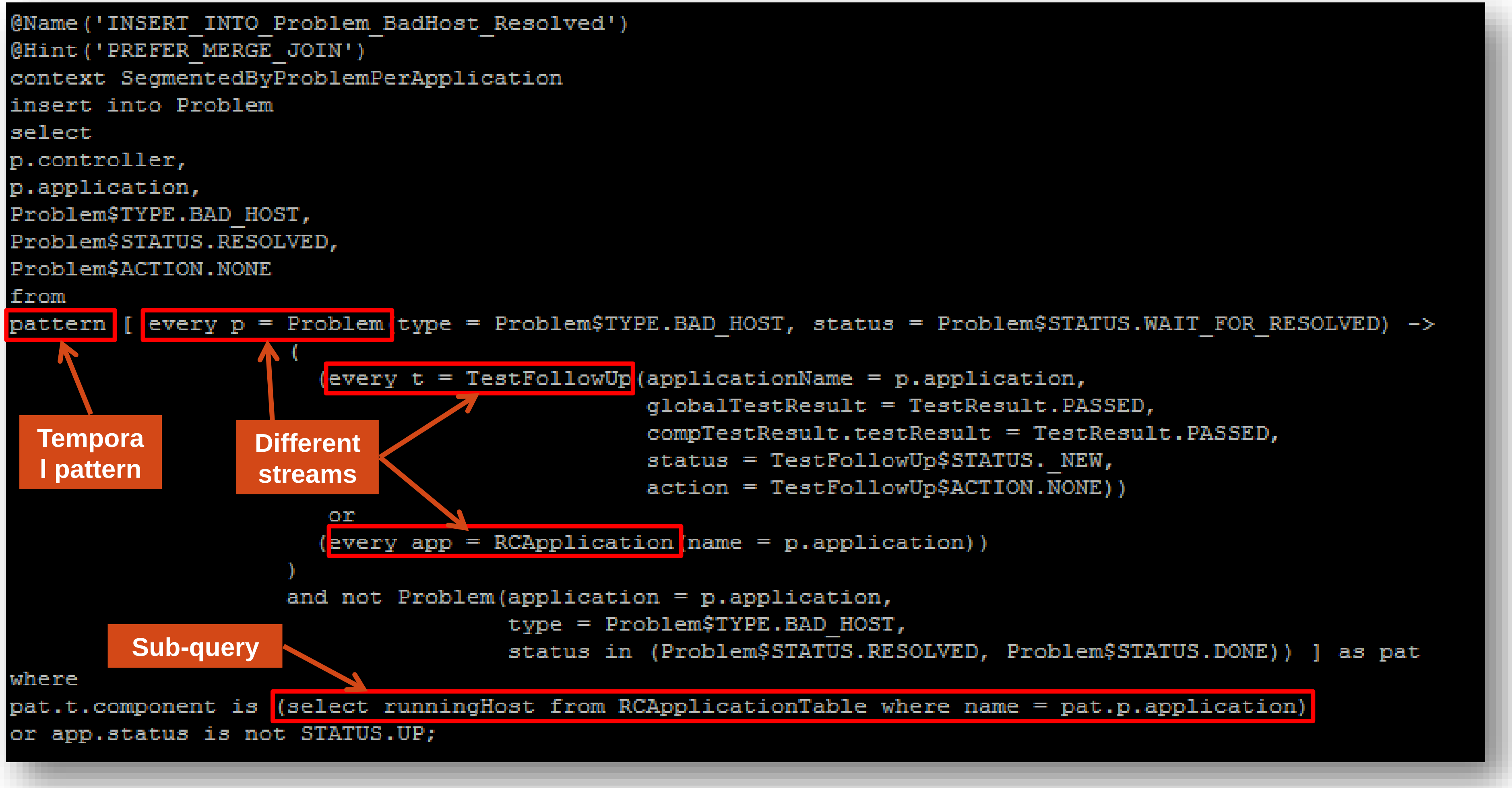
CONCLUSIONS & OUTLOOK

- **During LHC Run 2, the use of CHIP for automation and recovery proved to be a valuable asset**
  - Reduce probability of mistakes
  - Improve response time (computers are faster than humans…)
- **Extensive use of CHIP during every ATLAS run**
  - From simple actions (like restarting a failing application) to complex automation and recovery procedures
- **The introduction of a CEP engine has added flexibility and simplification to the continuous monitoring and supervision of the DAQ system**
  - Extensive anomaly detection
  - Complex patterns
  - Advanced configuration
- **Looking forward for more advanced automation and anomaly detection for LHC Run 3**

14

BACK-UP

CHIP & EPL: MORE COMPLEX EXAMPLE



```
@Name('INSERT_INTO_Problem_BadHost_Resolved')
@Hint('PREFER_MERGE_JOIN')
context SegmentedByProblemPerApplication
insert into Problem
select
p.controller,
p.application,
Problem$TYPE.BAD_HOST,
Problem$STATUS.RESOLVED,
Problem$ACTION.NONE
from
pattern [ every p = Problem(type = Problem$TYPE.BAD_HOST, status = Problem$STATUS.WAIT_FOR_RESOLVED) ->
                    (
                      (every t = TestFollowUp(applicationName = p.application,
                                              globalTestResult = TestResult.PASSED,
                                              compTestResult.testResult = TestResult.PASSED,
                                              status = TestFollowUp$STATUS._NEW,
                                              action = TestFollowUp$ACTION.NONE))
                       or
                      (every app = RCApplication(name = p.application))
                    )
                    and not Problem(application = p.application,
                                    type = Problem$TYPE.BAD_HOST,
                                    status in (Problem$STATUS.RESOLVED, Problem$STATUS.DONE)) ] as pat
where
pat.t.component is (select runningHost from RCApplicationTable where name = pat.p.application)
or app.status is not STATUS.UP;
```