BOSS: the CMS interface for job summission, monitoring and bookkeeping

W. Bacchi , P. Capiluppi, G. Codispoti, C. Grandi
INFN - Bologna, Italy
D.Colling, B.McEvoy, S.Wakefield, Y.J. Zhang
Imperial College London, UK

BOSS: Batch Object Submission System

- A tool for batch job submission, real time monitoring and bookkeeping
  - interface to local and grid schedulers
  - retrieval of user defined info from process STDOUT
  - store job-specific logging and bookkeeping info in a relational DB
  - provide real time monitor
- Optimized for use in distributed environment
  - Glite ready

Interfacing batch schedulers

- BOSS allows transparent use of any local or distributed scheduler (LSF, PBS, Condor, gLite, ...)
  - allow to perform standard operations: submit, query, kill, output retrieval
  - plug in system:
    - Script interface: administrator site can modify existing scripts or add its own
  - Sample plug-in's for many schedulers provided
    - Particular effort developing glite scripts

CMS requirements

- PB of data produced by the online farm and MC simulations
  - Data stored in a distributed environment
  - Access to data in the site where they are stored
  - Multiple processing over the same dataset
    - Chains of processing to be done over datasets (e.g.: sim-digi-reco, analysis processes etc.)
    - complex jobs to handle
  - A lot of homogeneous processes to be run simultaneously over several datasets

BOSS job concept

- A BOSS job is a single elaboration unit
  - Can be made of multiple processing steps (user executables)
  - allows complex workflows: executables chaining
  - Chaining tool may be external
- Multiple identical jobs can be grouped in Tasks:
  - Logical grouping of jobs
  - compact description of multiple homogeneous jobs using iterators
  - Multiple iterators allowed
  - XML description

User defined information

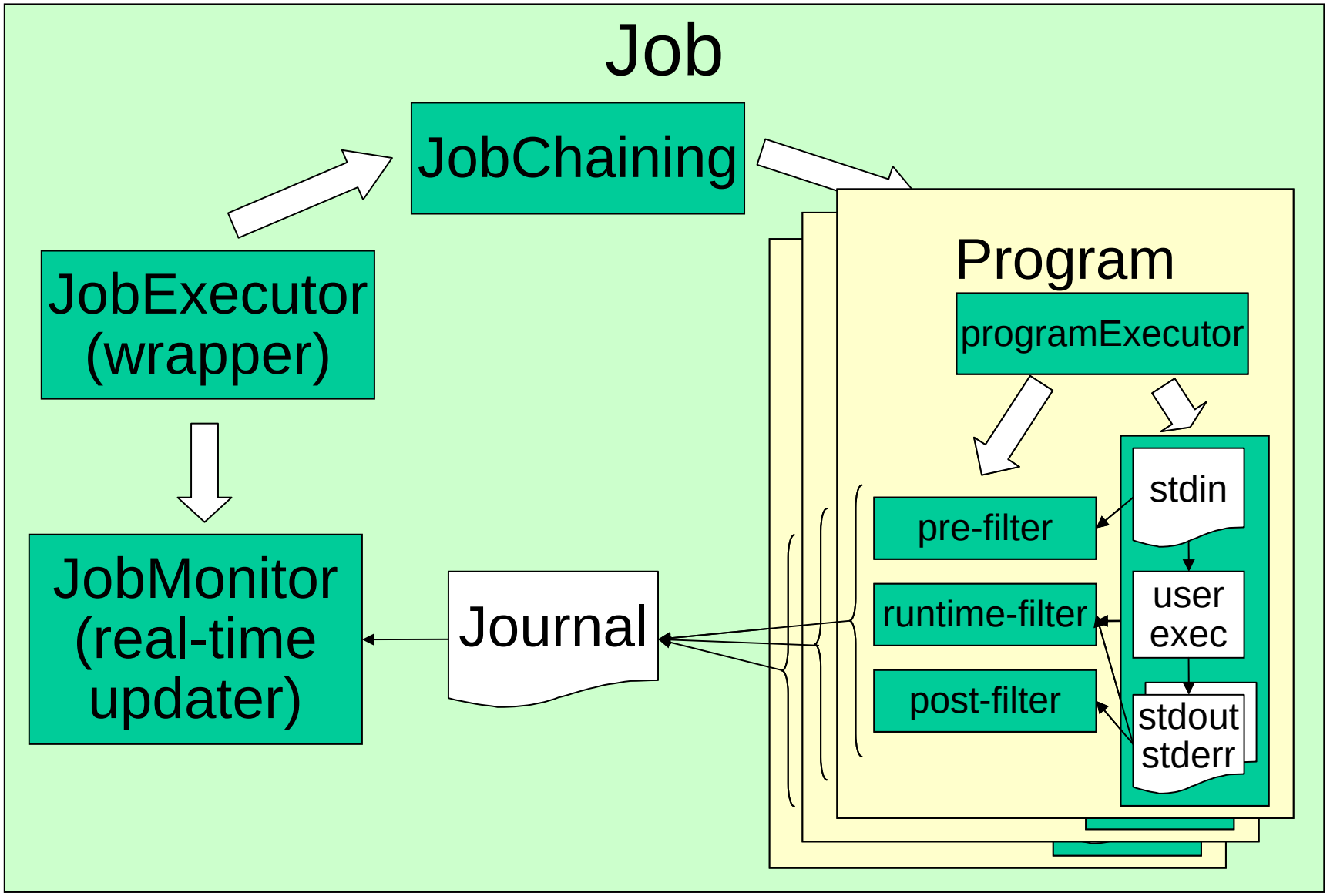
- A program type can be defined for a given elaboration:
  - Schema for the information to be monitored
    - A new table is created in the BOSS database with a defined structure
  - Algorithms to retrieve the information from the job
    - The program filters are stored in the database
- Defined user filters:
  - Perform stage in/out actions
  - retrieve wanted info from STDOUT
- One or more program types can be specified for a program
- Jobs standardization (analysis, MC,...)

  => Applications may define their own program type

BOSS wrappers

- A Wrapping system is needed to manage the execution of several processes on the WN
- BOSS Wrapping system is made of:
  - a wrapper of the user job (jobExecutor):
    - access to local runtime infos
    - starts chaining of programs
    - starts real time monitor
  - a chainer: allow complex programs workflow
    - Linear chaining provided by default
    - External tools may be also used
  - a program wrapper (programExecutor)
    - access to local runtime info's for the single program
    - starts pre-runtime-post filters, allows access to specific info's

Job wrapper


Job
JobChaining
JobExecutor
(wrapper)
JobMonitor
(real-time
updater)
Journal
Program
programExecutor
pre-filter
runtime-filter
post-filter
stdin
user
exec
stdout
stderr

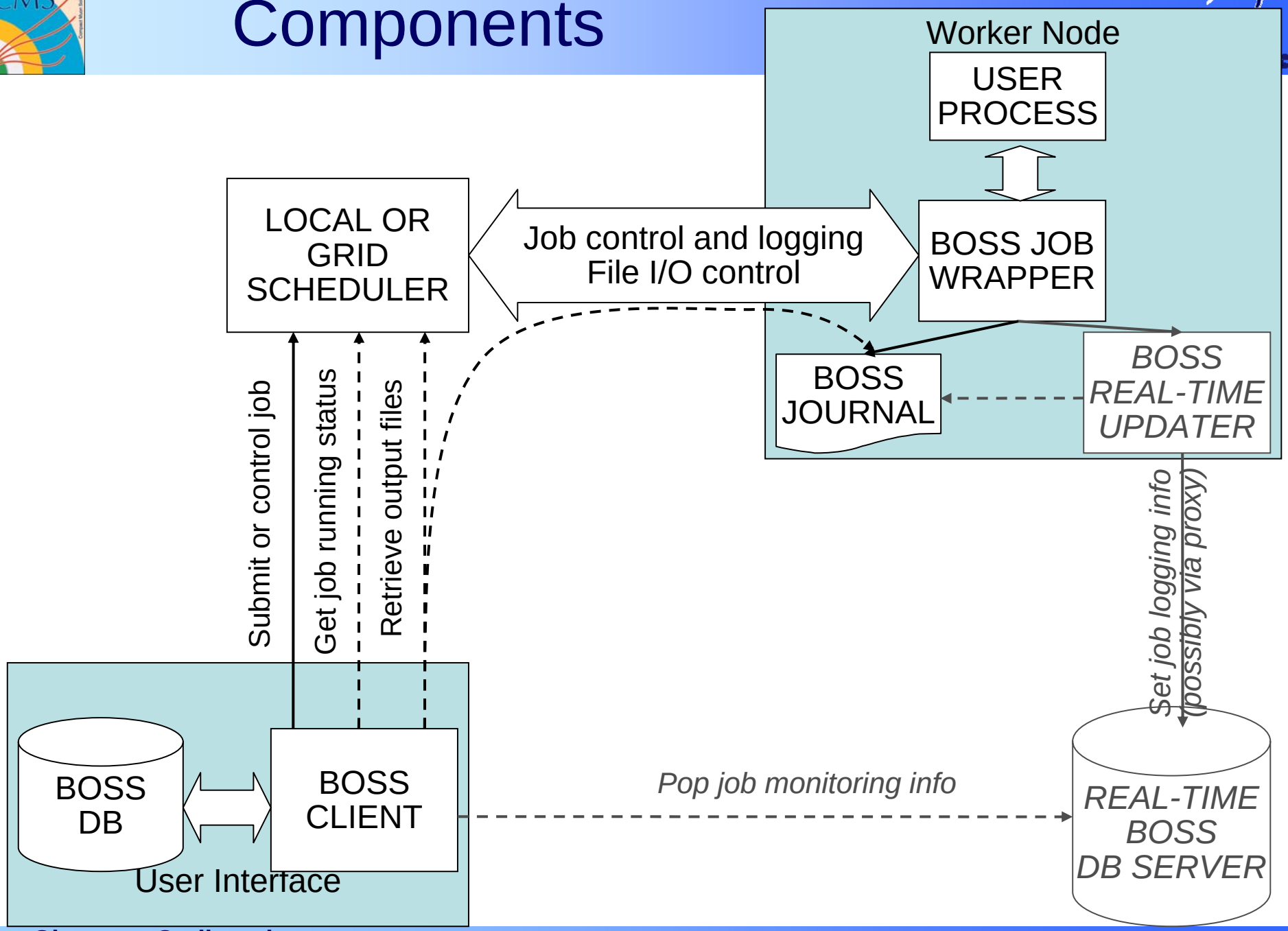
Logging and Monitoring

- Logging provides long term storage of information
  - allowed using a relational DB
  - allowed personal db implementation in SQLite on local disk
  - logging database updated from journal file retrieved at end of job and, optionally at runtime from information in RT server
- Monitoring provides real time access to logging info
  - using an intermediate Real Time Server
  - RT-clients registered to the BOSS client as plug-in's
  - may use different servers and technologies : R-GMA, Clarens, MonaLisa etc…
  - allow different RT mechanisms for each job

Real-time system

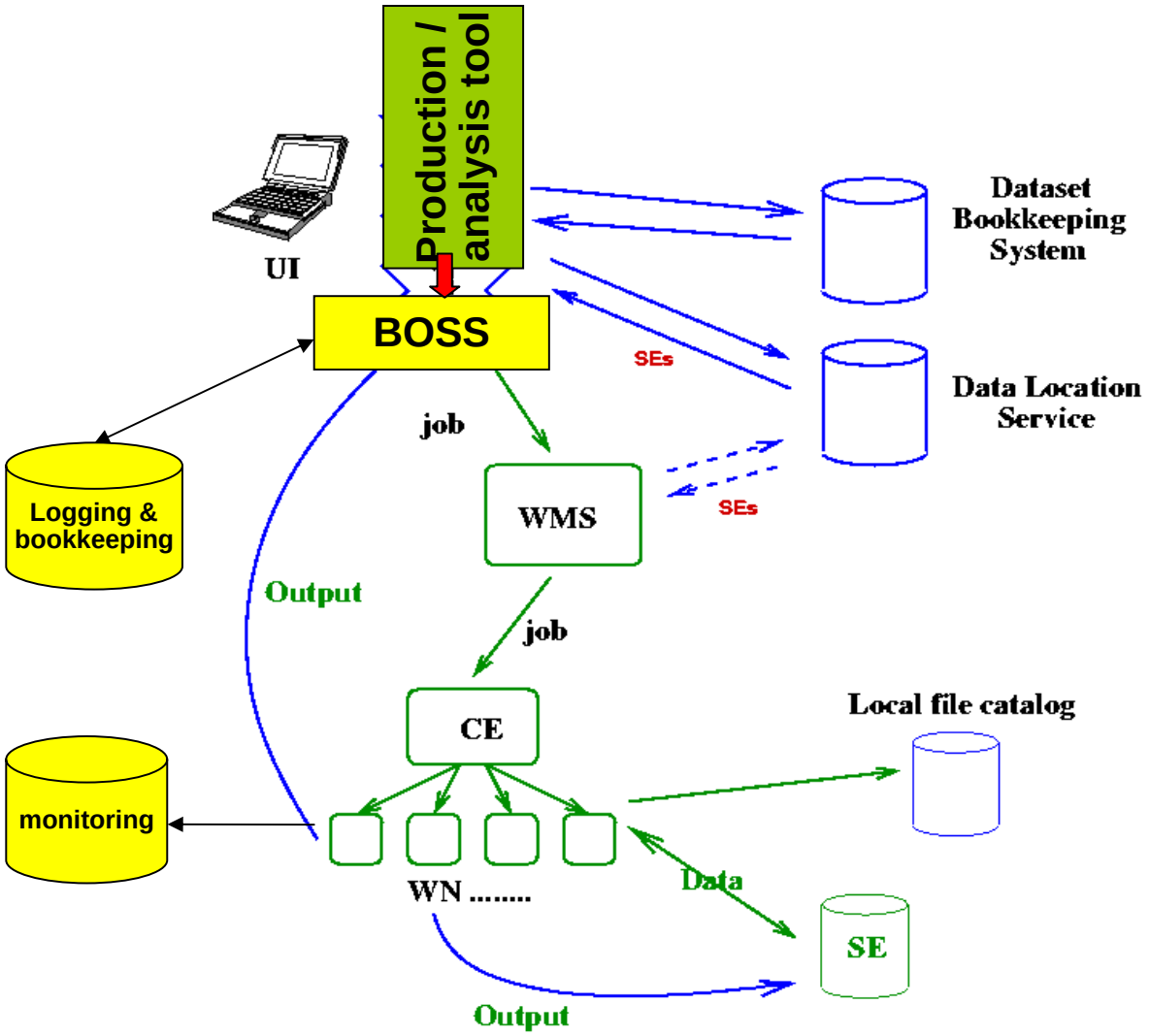
- Two RT-clients
  - the real-time updater that runs on the execution host
    - inserts or updates information about the running job
  - a plug-in used by the boss client on the user interface
    - fetches information about selected jobs and deletes it afterwards
- One server
  - the real-time DB server
    - stores temporarily job information while the job is running
    - shared by many users
    - simple structure
      - identification of BOSS-client and of user
      - identification of destination table/variable
      - value of parameter and time-stamp
    - final L&B doesn't rely on it

Components


Worker Node
USER PROCESS
LOCAL OR GRID SCHEDULER
Job control and logging
File I/O control
BOSS JOB WRAPPER
BOSS JOURNAL
BOSS REAL-TIME UPDATER
Submit or control job
Get job running status
Retrieve output files
Set job logging info (possibly via proxy)
BOSS DB
BOSS CLIENT
User Interface
Pop job monitoring info
REAL-TIME BOSS DB SERVER

BOSS in CMS computing

- Used in CMS MC production for 4 years
- Prototype CMS distributed analysis system (GROSS) based on BOSS and later new analysis system using BOSS
- BOSS v4 with new architecture and many new features



Glite Bulk submission

- BOSS modified to profit from glite bulk submission
  - Chains grouped for submission to allow creation of an unique input sandbox with common files
  - Actual implementation of a bulk submission delegated to the scheduler submit script
- Submission scripts implemented to efficiently use jdl job types:
  - Normal, for single submission
  - Parametric, most compact jdl, iteration over the boss job id: still to investigate if limitations can arise from the possibility to iterate over an unique parameter
  - Collection, more general bulk submission possibilities: a single file keeps all the job jdl's, shared input sandbox allowed
- Tested version 1.4.1, planned 3.0.0 as soon as it will be ready on the pre-production system (PPS)

Summarizing

- Transparent use of any batch system
- Provide persistent storage of the logging infos
- Logging specific info
- Real Time Monitoring
- Glite optimization
- Sandbox packing for efficient use in distributed systems
- XML task description
- Command Line Interface
- Integrability in an experiment framework through API (C++ & Python)

Status

- First version released with a full set of basic functionalities
  - MySQL and SQLite back-ends for local DB
  - MySQL real-time DB – full working RT monitoring
  - XML task description at declaration time, nested iterators allowed
  - Full task description in the database
  - Glite Bulk Submission
  - Basic executable linear chaining, default solution
  - plug-in system for chainer implemented but we need to better understand how to handle external chainers (mainly to configure them allowing the use of the program wrapper)
  - MonaLisa monitoring allowed via APMon
  - plug-in for many schedulers: local submission, lsf, edg, glite; we are experiencing also some effort for condorG with v3.6 via end user support to allow use within OSG

Future plans

- Allowing the use of chainer plugins, mainly external programs (e.g. SHREEK)
- Implement more backend possibilities (i.e. ORACLE)
- Implement more RT monitoring solutions (i.e. R-GMA, Clarens, web services, but also mysql query encryption to avoid firewall problems)
- Finalize API, increase query possibilities
- Use external standard libraries (mainly from BOOST)
- Look at writing wrapper in scripting language i.e Perl/Python