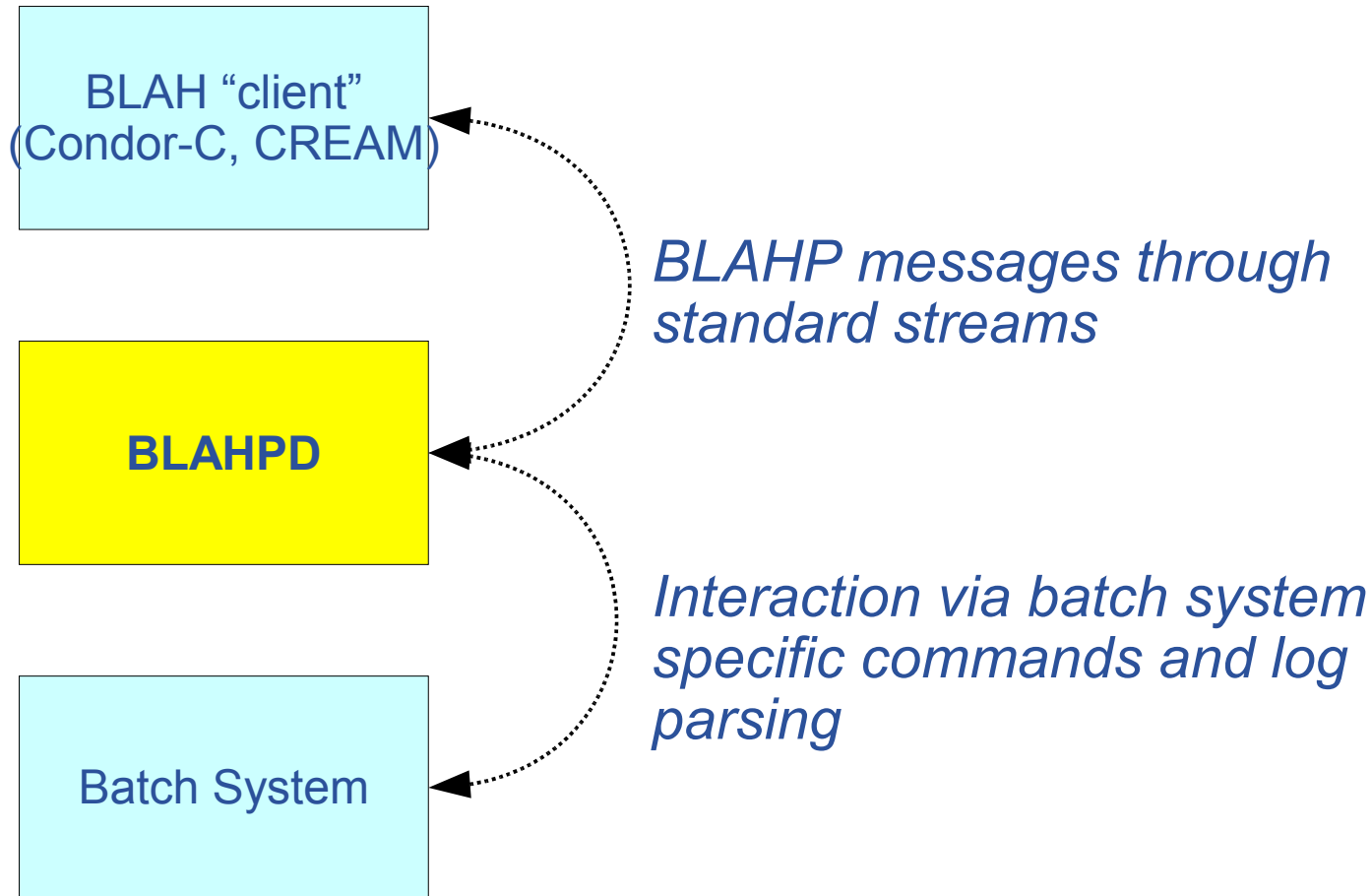


BLAHPD: the interface to batch systems in EGEE

*David Rebatto (david.rebatto@mi.infn.it)
I.N.F.N. - Milano*

- **The protocol**
 - The BLAHP (Batch Local ASCII Helper Protocol) provides a set of plain ASCII commands used by Condor-C (and CREAM) to manage jobs on the batch systems.
- **The daemon**
 - BLAHPD implements the helper daemon responsible for converting BLAHP commands into batch system actions, interpreting their results and reporting them in BLAHP format.



- **Batch system interactions**
 - Not part of the daemon's core.
 - Performed via external scripts with a common syntax among different batch systems.
 - Three scripts for every batch system:
 - **xxx_submit**
 - **xxx_status**
 - **xxx_cancel**
- **The effort for batch system abstraction is concentrated in these scripts.**
 - Adding support for new batch systems requires “only” to write a new set of scripts.
 - No need to go through the C code of the daemon itself.

- **Why scripts?**

- In principle, the three job management commands could be implemented in any language (e.g. in C using the batch system API)
- We didn't use APIs because:
 - using API would add more dependencies in the gLite building process (BLAHPD is a gLite component);
 - when upgrading from one version of a batch system to the next one, CLI commands should be at least as consistent as the API, if not more.
- Anybody willing to interface BLAHPD to a new batch system can implement the command in the preferred way, provided that the syntax and the semantics are respected.
 - Note that there are some issues to be addressed when creating the commands (either scripts or binaries), see “Issues” slides.

- **xxx_submit:**

- Syntax

```
xxx_submit -c <command> [-i <stdin>] [-o <stdout>] [-e <stderr>] [-v
<environment>] [-s <yes|no>] [-w <workdir>] [-q <queue>] [-n
<mpi_nodes>] [-r <yes|no> -p <poll_interval>
-l <min_proxy_lifetime>]
```

-c	the command to execute
-i, -o, -e	the standard streams
-v	the job environment, specified as a semicolon separated list of key=value pairs
-s	yes if the executable specified with -c has to be staged on WN
-w	initial workdir (on the submitting machine)
-q	the queue where to submit the job
-n	number of nodes to reserve for an MPI job
-r	yes for proxy renewal support
-p	polling interval for proxy check
-l	minimum proxy lifetime under which the job is killed

- Returns:

- the jobId of the submitted job and exitcode = 0 if successful;
 - an error message and exitcode >0 in case of error.

- **xxx_status**

- Syntax:

- `xxx_status <jobId> [-w]`

- w if specified, the command should return the name of the executing host

- Returns:

- a classad describing the job status and exitcode 0 if successful:

- [
 BatchJobId="xxx";
 JobStatus=<s>;
 WorkerNode=<wn_hostname>;
 ExitCode=n;
]
 <s>: 1=queued; 2=running; 3=deleted; 4=finished; 5=hold;
 WorkerNode: only if -w is specified;
 ExitCode: only if job is in status 4 (finished);

- an error string and exit code >0 in case of error.

- **xxx_cancel**
 - Syntax:
xxx_cancel <jobId>
 - Returns:
 - exitcode = 0 if successful, >0 otherwise

- **Job submission**
 - When a successful submission command returns a job ID, the job has to be already accepted by the server (i.e. there are no possibility that the job gets lost).
- **Job status**
 - The batch system must be able to report the status of already finished jobs (at least for a given amount of time), and to distinguish between done, failed and cancelled jobs.
- **Job cancellation**
 - The cancel command should return a success only if it was really able to cancel the job.
- **Job modification**
 - There must be a way to dispatch files to running jobs.
- **More assumptions...**
 - ...if some planned feature (e.g. two-phase commit) are to be implemented (see slide 12).

- **Some of the previous assumptions are not satisfied**
 - PBS can “lose” jobs in internal queues during submission.
 - When a job finish (either successfully or not) *qstat* does not report about it anymore.
 - Moreover, a high rate of *qstat* and *bhist* commands quickly overload the server.
- **Proposal**
 - Reconstruct job status from events in the batch system log file.
- **Problems**
 - The CE and the head node can be separated machines.
 - Parsing the log file for every status request quickly kill the server.
- **Solution**
 - We developed a “status cache server” running on head node, monitoring the batch system log file in real time and caching the status of all BLAHP managed jobs.

- **Proxy renewal**
 - The proxy renewal service can make a fresh proxy available on the CE, but how can we move it on the job's work directory?
 - We did not find a way to deliver files to a job - once it has begun to run - through the batch system itself.
- ***Solution***
 - We developed a “proxy receiver”, a server started by the job wrapper on a WN, and listening for incoming connections from the CE.
 - When a client connects, the server answers with the jobId; if it matches, the client sends the new proxy over a GSI encrypted connection.
 - The server acts also as “watchdog”, killing the job if a fresh proxy is not received within n minutes before proxy expires.

- **Two-phase commit in submission**
 - To be sure that no “lost” jobs run on the system
 - E.g., jobs could be submitted in *held* status, and released only when the BLAHP client (e.g. Condor-C gridmanager, CREAM) confirm that it received and saved the jobID
 - Being able to submit “held” jobs is another assumption on the batch system (currently satisfied by both PBS and LSF).
- **Job requirements**
 - For a better usage of the farms, the job's requirements used for the matchmaking should be used also to steer the local submission.
 - *Open issue*: how do we express such requirements for different batch systems (and for different configurations of the same batch system)?
- **Job output inspection at runtime**
 - Still to be discussed, “proof of concept” ready.