Progress Tracking Of Batch Farm Jobs: taskMaster Database

PAWEL AMBROZEWICZ, MARK M. ITO
September 18, 2002

Abstract

The ***taskMaster*** database, which is described in this document, was created to facilitate tracking the progress of jobs submitted to the batch farms. In principle, the `jobstat` command and `JASMine` status web pages are sufficient to track job progress, but using them is rather non-trivial, time consuming and must be done in a timely fashion. To alleviate the problem, a set of Perl scripts was developed to populate a MySQL database and provide the user with easy access to the progress information.

1 Introduction: Batch Farm Requests

1.1 Glossary

To reduce ambiguities to a minimum, the vital terms used in the text must have definite meaning. Necessary definitions have been formulated in the section below.

farm task: A set of farm requests, all having an identical value of the `JOBNAME` keyword in their `jsub` command files. Note that a single task can include multiple farm requests.

farm request: A single invocation of the `jsub` command. A farm request always takes as input a single job command file, as described at

http://cc.jlab.org/docs/scicomp/how-to/keywords.html.

Note that a single farm request can result in multiple farm jobs.

farm job: The actual executable entity that is run on a farm node. This is the object that is scheduled by the batch farm software and runs on a single, assigned node as job slots become available. It is the thing you see as an entry in the farm job queue listings.

silo request: A single invocation of the jput, jget or jcache commands. These request that a files or files be written to the silo, retrieved to an arbitrary location, or retrieved to a cache disk respectively. Note that a single silo request can result in multiple silo jobs.

silo job: A single-file operation of the silo, either read or write.

jcache farm request: A particular type of farm request. It has the following properties:

- It is submitted by the farm software, in response to a farm request that has silo files listed as values for its `INPUT_FILES` keyword.
- It is submitted to the jcache queue. The jcache queue is a queue in the batch farm system, like production and low_priority.
- It executes a single jcache command when it runs and then exits. The jcache request may or may not request multiple files. The files are cached to the farm stage cache disks.

In this note, if there is no ambiguity, a **farm job** will simply be referred to as a **job** and a **farm request** will be referred to as a **request**.

1.2 Sequence of Events

In the following it is assumed that the user has listed multiple files with the `INPUT_FILES` keyword of the `jsub` command file to create multiple jobs on the farm, each using one of these files. Execution of a job at the batch farm machines, managed by the Load Sharing Facility (LSF), proceeds through the following stages,

1. A job is submitted via a `jsub` command which takes as an argument a name of a command file (see *http://cc.jlab.org/docs/scicomp/how-to/keywords.html* for examples):
   `jsub filename`
   The farm software issues a farm request ID.

2. The input files in the `jsub` command file are put into groups of ten (the last group may contain fewer than ten files).

3. For each group, a jcache farm request is generated. When this farm job runs, a jcache silo request is issued to the silo system.

4. For each group, separate farm jobs are created, one for each input file in the group. All of the jobs in the group are put into the PEND state.

5. As each group's jcache silo request completes, the group's farm jobs are allowed to run if free job slots are available. If free slots are not available, the jobs continue to pend. Note that none of the jobs in the group are allowed to run until the jcache silo request corresponding to this group is complete; that is until all files in the group have been retrieved to the cache disk. Note further that there are never more than ten files requested in any one jcache silo request.

6. Eventually each farm job enters the "RUN" state. When completed successfully, the state is changed to "DONE". If it encountered software or hardware problems during its execution the job could be killed ("EXITED") or go into unknown state ("UNKWN").

2 taskMaster DB: Overview of the Model

The database itself consists of five tables,

- ***task:*** for storing information for all tasks,
- ***jsub:*** for storing information for submissions within a task,
- ***farmjob:*** for storing information for farm jobs within a submission,
- ***jcache:*** for storing information for `jcache` requests associated with a submission,
- ***silojob:*** for storing information for tape silo jobs within a `jcache` request.

where all the information, necessary to track down the task progress, is stored. Composition of each table and the way each field is being filled are shown in Figure 1 and Appendix A. In the present implementation the first execution of ***TMsubmit.pl*** results in an auto-incremented task entry in the ***task*** table with the name as given in the submission file under optional keyword `JOBNAME`. Repeated submissions with the same `JOBNAME` (task name) add new entries pertaining to the same task.

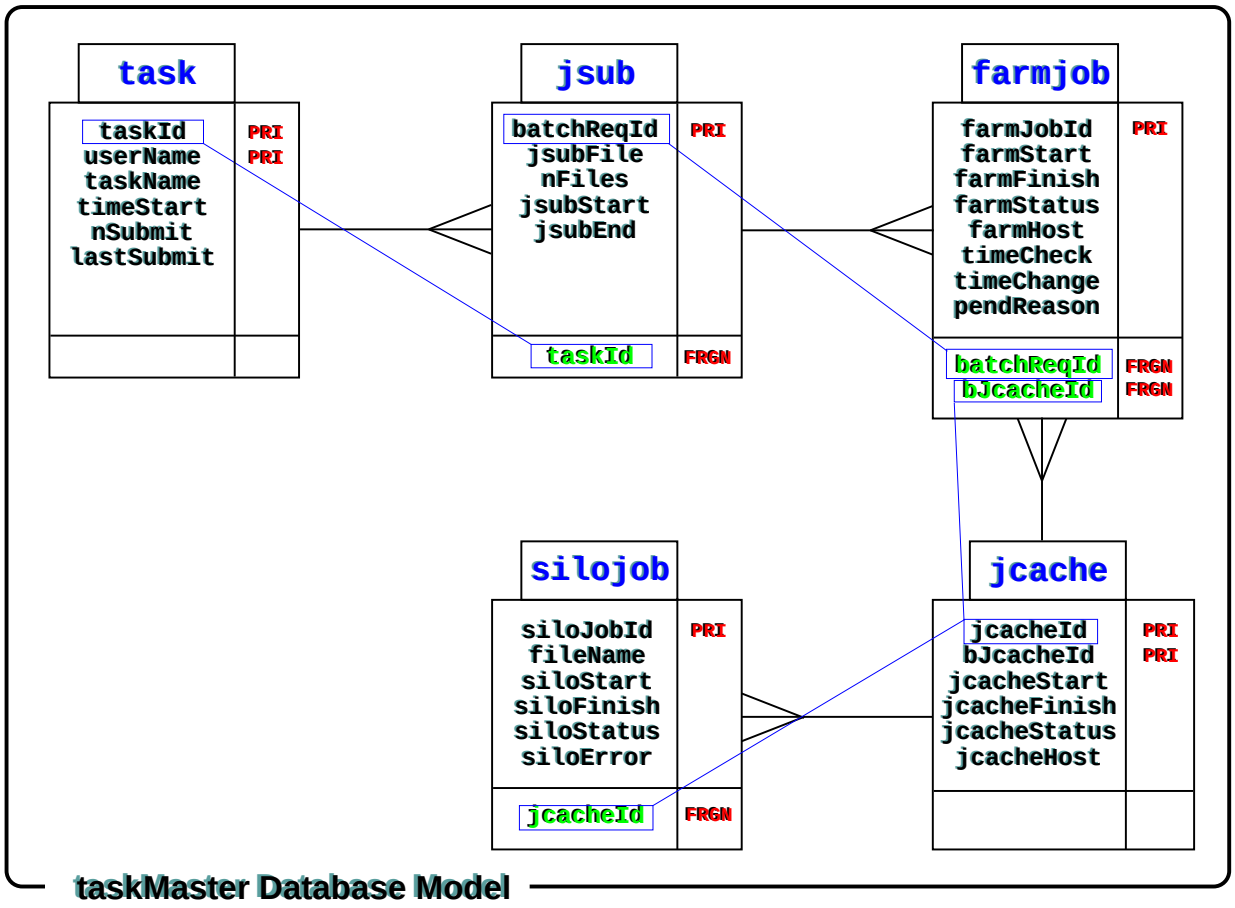


Figure 1: ***taskMaster Database:*** *This diagram explains the way the taskMaster database tables are related. "Crow feet" correspond to one-to-many relationships between the tables.*

3 The Scripts

The taskMaster (TM) Perl library, developed to ease the tracking process, consists of four scripts,

1. ***TMsubmit.pl*** - submits a request to the batch farm and waits until all the input files are queued. In this way, information sufficient for a successful execution of the update script, ***TMupdate.pl***, is obtained. ***TMsubmit.pl*** performs initial population of the database tables pertaining to a given farm request.

2. ***TMupdate.pl*** - updates old or inserts new information as it becomes available; timely execution is necessary.

3. ***TMdump.pl*** - dumps, in a concise form, essential information (request/job ID's, status, start/finish times, hosts) on standard output.

4. ***TMdelete.pl*** - deletes a task from the taskMaster database.

Invoking each script without any arguments prints out the usage on the standard output.

3.1 Initialization: TMsubmit.pl

Initialization of a task in the taskMaster database is handled by Perl script ***TMsubmit.pl***. It takes as an argument the name of the `jsub` command file and does the following,

(*) submits the task: it issues `jsub` command using a command file provided by the user. The command file must have a value for the `JOBNAME` keyword. This value will be used as the name of the task. If this task does not exist already, one will be created.

(*) captures `jsub` output: the output of `jsub` is piped through the script; this output contains batch request ID which uniquely identifies the submission.

(*) issues `jobstat` command: polls the batch farm system in order to verify whether all the jobs were scheduled; this allows to extract batch farm `jcache` ID's, which is vital for the tracking purpose,

(*) populates taskMaster DB tables: it again issues `jobstat` and parses its output to extract the information necessary to describe the task and inserts appropriate pieces of information into the taskMaster tables. It tries to match batch farm `jcache` ID with tape silo `jcache` ID, if the request to the silo has already been made. The whole idea of this project hinges on this identification which is being made by matching farm start time and silo submit time given a user name, type of tape silo request (jcache) and a host name designated to handle the request. In case the link cannot be made the farm `jcache` ID is inserted into the ***farmjob*** table while silo `jcache` ID is assigned a zero value.

The following is the usage message from ***TMsubmit.pl***:

```
TMsubmit.pl:  Depending on the arguments specified on the command line
              it is capable of submitting a job to the batch farms via
              issuing the "jsub myfile" command, populating the taskMaster
              database and retaining in a separate file ("filename")
              the request ID issued by LSF software upon submitting a job
              to the batch farms.

USAGE:        TMsubmit.pl jsubfile=<myfile> [reqfile=<filename>]

TEMPLATE:     TMsubmit.pl jsubfile=
TEMPLATE:     TMsubmit.pl jsubfile= reqfile=
```

3.2 Information updates: TMupdate.pl

Any subsequent update is done by running ***TMupdate.pl***. This needs to be performed frequently as the information available through `jobstat` command is not stored in any database for long. It becomes unavailable after 1 hour unless the number of jobs in the queue is greater than 1000. In that case it is removed even faster. Therefore timely execution of the script is desired. ***TMupdate.pl*** performs the following,

(*) **Updates** the existing task database information by comparing it with `jobstat` output and/or with information obtained from the tape silo database (***JobQueue*** - on `mssdb1`); especially important is identifying the correct silo `jcache` ID's for which corresponding ***farmjob*** entries are zeros, which is done the same way as in ***TMsubmit.pl***

(*) **Inserts** any task information not available at the time ***TMsubmit.pl*** was run. This is usually information from the tape silo database.

The following is the usage message from ***TMupdate.pl***:

```
TMupdate.pl:  Updates progress info for the task submitted to batch farms
              via issuing "jsub jsubfile" command.

USAGE:        TMupdate.pl jsubfile=<myfile>

TEMPLATE:     TMupdate.pl jsubfile=
```

3.3 Checking the progress

3.3.1 Standard Output Print-Out: TMdump.pl

To view the essential pieces of information one can execute ***TMdump.pl***. This script prints on the standard output job ID's on the farm, jcache and silo levels as well as start/finish times, job hosts and job states. An example of such a printout is shown in Figure 2. In this example a task consists four separate submissions: three single-input-file submissions and

one multiple-input-file submission. Each submission in turn is displayed in four sections. The first corresponds to the farm request generated by this submission and shows its beginning and ending times. The second section displays a line for each farm job generated from the farm request. The third section connects the silo `jcache` requests with their respective parent jcache farm requests. The fourth section shows the progress of the individual silo jobs.

The following is the usage message from ***TMdump.pl***:

```
TMdump.pl:    Dumps database info for the task submitted to batch farms
              via issuing "jsub jsubfile" command.

USAGE:        TMdump.pl jsubfile=<myfile>

TEMPLATE:     TMdump.pl jsubfile=
```

3.3.2 Task Database Query Page: tabledump.pl

An alternative way of accessing query results is invoking a web interface ***tabledump.pl***. Figures 3 and 4 show consecutive dynamically generated webpages with the results. The URL location of the script is

`http://clasweb.jlab.org/cgi-bin/farm/tabledump.pl`.

3.4 Termination: TMdelete.pl

To terminate a task and delete the corresponding entries from the taskMaster database ***TMdelete.pl*** has to be executed. This completes the task.[1]

The following is the usage message from ***TMdelete.pl***:

```
TMdelete.pl:    Deletes database info for the task submitted to batch farms
                via issuing "jsub jsubfile" command.

 USAGE:         TMdelete.pl jsubfile=<myfile>

 TEMPLATE:      TMdelete.pl jsubfile=
```

A Description of the Database Tables

This section is intended to provide the reader with a detailed description of the way each database field is filled. Each field is identified by its tag called **key**. To populate the database tables unambiguously some of the fields must be unique - the corresponding keys are called primary. Also, to model the relations between the tables for each one-to-many mapping the

[1] As of this writing this function has not been fully implemented. Please check with the authors if you experience difficulty clearing out your old tasks.

		jcache Id						
batchId	farm Id	farm	silo	silo Id	status	start	finish	node
662385						06-13 09:01:35	06-13 09:32:01	
662385	987293	666078			EXIT	06-13 09:12:43	06-13 09:32:01	farml104
662385		666078	431107		Done	06-13 09:04:57	06-13 09:12:15	farml13
662385			431107	1	Failed	06-13 09:04:57		farml13
666728						06-19 06:50:05	06-19 07:19:47	
666728	993661	670449			DONE	06-19 07:17:22	06-19 07:19:47	farml105
666728		670449	437601		Done	06-19 07:02:15	06-19 07:16:16	farml13
666728			437601	1	Done	06-19 07:11:04	06-19 07:16:16	farml13
666729						06-19 06:51:06	06-19 08:31:13	
666729	993663	670450			EXIT		06-19 08:31:13	None
666729		670450	437602		Done	06-19 07:03:29	06-19 09:07:44	farml13
666729			437602	1	Done	06-19 09:04:23	06-19 09:07:44	farml13
676029						06-26 12:36:22		
676029	7506	679765			RUN	06-26 13:31:23		farml39
676029	7505	679765			RUN	06-26 13:31:23		farml168
676029	7507	679765			RUN	06-26 13:31:23		farml47
676029	7508	679765			RUN	06-26 13:31:23		farml181
676029	7509	679765			RUN	06-26 13:31:23		farml41
676029	7510	679765			RUN	06-26 13:31:24		farml51
676029	7511	679765			RUN	06-26 13:31:24		farml182
676029	7512	679765			RUN	06-26 13:31:24		farml152
676029	7513	679765			RUN	06-26 13:31:24		farml44
676029	7514	679765			RUN	06-26 13:31:24		farml156
676029	7516	679766			PEND			None
676029	7517	679766			PEND			None
676029		679765	448854		Done	06-26 12:37:13	06-26 13:30:47	farml13
676029			448854	1	Done	06-26 12:41:34	06-26 13:07:11	farml13
676029			448854	2	Done	06-26 12:40:38	06-26 13:08:09	farml13
676029			448854	3	Done	06-26 12:43:42	06-26 13:09:18	farml13
676029			448854	4	Done	06-26 12:46:00	06-26 13:13:29	farml13
676029			448854	5	Done	06-26 12:50:18	06-26 13:05:18	farml13
676029			448854	6	Done	06-26 12:46:02	06-26 13:13:52	farml13
676029			448854	7	Done	06-26 12:55:30	06-26 12:55:31	farml13
676029			448854	8	Done	06-26 13:00:13	06-26 13:10:41	farml13
676029			448854	9	Done	06-26 13:05:04	06-26 13:29:11	farml13
676029			448854	10	Done	06-26 13:09:41	06-26 13:30:47	farml13
676029		679766	448855		Active	06-26 12:46:30		farml12
676029			448855	1	Done	06-26 12:49:12	06-26 13:04:37	farml12
676029			448855	2	Running	06-26 13:15:11		farml12

Figure 2: Output of ***TMdump.pl***.

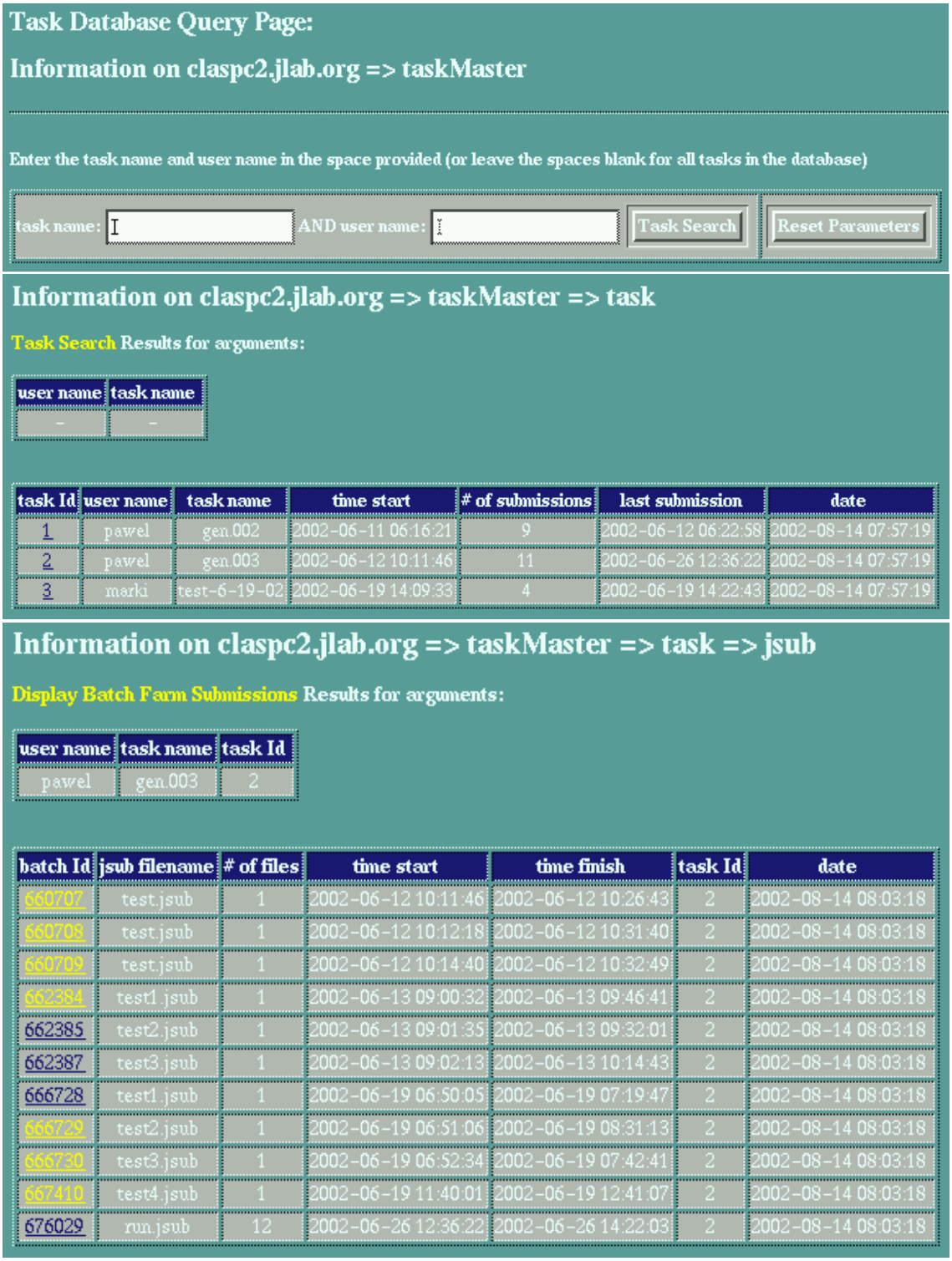
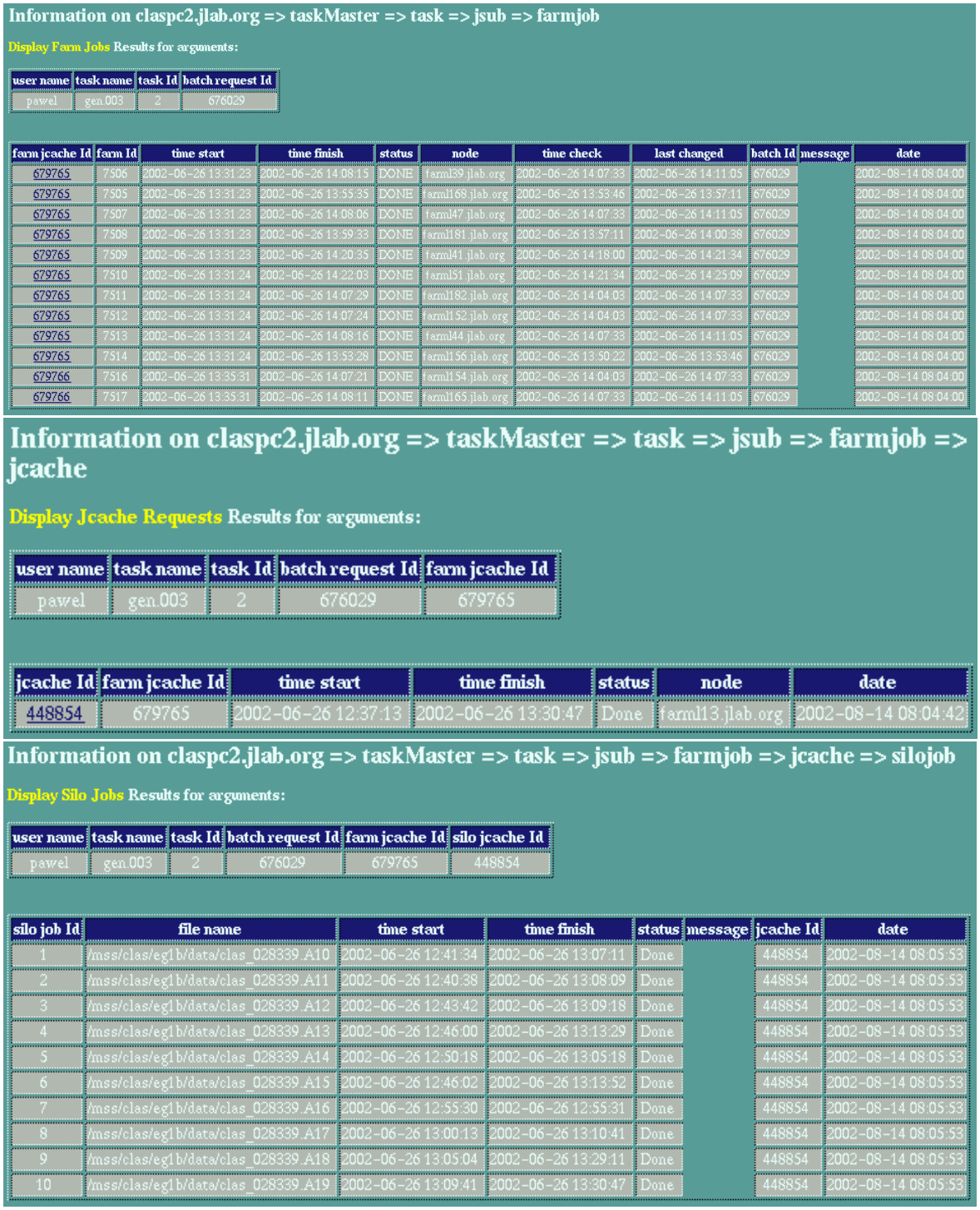


Figure 3: ***Task Database Query Pages:*** *(Top) Task Database Query Page. (Center) The results page. The query performed corresponded to leaving blank task and user name spaces. (Bottom) The results page for a selected task Id.*



Information on claspc2.jlab.org => taskMaster => task => jsub => farmjob

Display Farm Jobs Results for arguments:

user name	task name	task Id	batch request Id
pawel	gen.003	2	676029

farm jcache Id	farm Id	time start	time finish	status	node	time check	last changed	batch Id	message	date
679765	7506	2002-06-26 13:31:23	2002-06-26 14:08:15	DONE	farm139.jlab.org	2002-06-26 14:07:33	2002-06-26 14:11:05	676029		2002-08-14 08:04:00
679765	7505	2002-06-26 13:31:23	2002-06-26 13:53:35	DONE	farm169.jlab.org	2002-06-26 13:53:46	2002-06-26 13:57:11	676029		2002-08-14 08:04:00
679765	7507	2002-06-26 13:31:23	2002-06-26 14:08:06	DONE	farm47.jlab.org	2002-06-26 14:07:33	2002-06-26 14:11:05	676029		2002-08-14 08:04:00
679765	7508	2002-06-26 13:31:23	2002-06-26 13:59:33	DONE	farm181.jlab.org	2002-06-26 13:57:11	2002-06-26 14:00:38	676029		2002-08-14 08:04:00
679765	7509	2002-06-26 13:31:23	2002-06-26 14:20:35	DONE	farm141.jlab.org	2002-06-26 14:18:00	2002-06-26 14:21:34	676029		2002-08-14 08:04:00
679765	7510	2002-06-26 13:31:24	2002-06-26 14:22:03	DONE	farm151.jlab.org	2002-06-26 14:21:34	2002-06-26 14:25:09	676029		2002-08-14 08:04:00
679765	7511	2002-06-26 13:31:24	2002-06-26 14:07:29	DONE	farm182.jlab.org	2002-06-26 14:04:03	2002-06-26 14:07:33	676029		2002-08-14 08:04:00
679765	7512	2002-06-26 13:31:24	2002-06-26 14:07:24	DONE	farm152.jlab.org	2002-06-26 14:04:03	2002-06-26 14:07:33	676029		2002-08-14 08:04:00
679765	7513	2002-06-26 13:31:24	2002-06-26 14:08:16	DONE	farm44.jlab.org	2002-06-26 14:07:33	2002-06-26 14:11:05	676029		2002-08-14 08:04:00
679765	7514	2002-06-26 13:31:24	2002-06-26 13:53:28	DONE	farm156.jlab.org	2002-06-26 13:50:22	2002-06-26 13:53:46	676029		2002-08-14 08:04:00
679766	7516	2002-06-26 13:35:31	2002-06-26 14:07:21	DONE	farm154.jlab.org	2002-06-26 14:04:03	2002-06-26 14:07:33	676029		2002-08-14 08:04:00
679766	7517	2002-06-26 13:35:31	2002-06-26 14:08:11	DONE	farm165.jlab.org	2002-06-26 14:07:33	2002-06-26 14:11:05	676029		2002-08-14 08:04:00

Information on claspc2.jlab.org => taskMaster => task => jsub => farmjob => jcache

Display Jcache Requests Results for arguments:

user name	task name	task Id	batch request Id	farm jcache Id
pawel	gen.003	2	676029	679765

jcache Id	farm jcache Id	time start	time finish	status	node	date
448854	679765	2002-06-26 12:37:13	2002-06-26 13:30:47	Done	farm113.jlab.org	2002-08-14 08:04:42

Information on claspc2.jlab.org => taskMaster => task => jsub => farmjob => jcache => silojob

Display Silo Jobs Results for arguments:

user name	task name	task Id	batch request Id	farm jcache Id	silo jcache Id
pawel	gen.003	2	676029	679765	448854

silo job Id	file name	time start	time finish	status	message	jcache Id	date
1	/mss/clas/eg1b/data/clas_028339.A10	2002-06-26 12:41:34	2002-06-26 13:07:11	Done		448854	2002-08-14 08:05:53
2	/mss/clas/eg1b/data/clas_028339.A11	2002-06-26 12:40:38	2002-06-26 13:08:09	Done		448854	2002-08-14 08:05:53
3	/mss/clas/eg1b/data/clas_028339.A12	2002-06-26 12:43:42	2002-06-26 13:09:18	Done		448854	2002-08-14 08:05:53
4	/mss/clas/eg1b/data/clas_028339.A13	2002-06-26 12:46:00	2002-06-26 13:13:29	Done		448854	2002-08-14 08:05:53
5	/mss/clas/eg1b/data/clas_028339.A14	2002-06-26 12:50:18	2002-06-26 13:05:18	Done		448854	2002-08-14 08:05:53
6	/mss/clas/eg1b/data/clas_028339.A15	2002-06-26 12:46:02	2002-06-26 13:13:52	Done		448854	2002-08-14 08:05:53
7	/mss/clas/eg1b/data/clas_028339.A16	2002-06-26 12:55:30	2002-06-26 12:55:31	Done		448854	2002-08-14 08:05:53
8	/mss/clas/eg1b/data/clas_028339.A17	2002-06-26 13:00:13	2002-06-26 13:10:41	Done		448854	2002-08-14 08:05:53
9	/mss/clas/eg1b/data/clas_028339.A18	2002-06-26 13:05:04	2002-06-26 13:29:11	Done		448854	2002-08-14 08:05:53
10	/mss/clas/eg1b/data/clas_028339.A19	2002-06-26 13:09:41	2002-06-26 13:30:47	Done		448854	2002-08-14 08:05:53

Figure 4: ***Task Database Query Pages:*** *(Top) The farm jobs page for previously selected task ID and for a given submission (identified by batch farm request ID). (Center) The jcache request page associated with a given farm job. (Bottom) The tape silo page for a selected jcache ID.*

primary key on the "one" side of the relationship is placed into the table on the "many" side - the corresponding keys are called foreign. In what follows the primary keys are color-coded red while the foreign keys are color-coded green. All ordinary keys are in black.

1. **task:** stores task information and in present implementation consists of six fields,

   - **taskId** - Unique numerical identifier for all tasks, past and present.
   - **userName** - the corresponding field is filled by importing the appropriate environmental variable into ***TMsubmit.pl***; this prevents the tasks with the same task name but for different users to be merged together under one **taskId**,
   - **taskName** - the value of the field is taken from the `JOBNAME` keyword field in the `jsub` command file by parsing the content of this file,
   - **timeStart** - the value of this field is obtained by using Perl function ***time*** which is fed through another Perl function, ***localtime***, and the result is converted to one of the MySQL time-stamp formats (`YYYY-MM-DD hh:mm:ss`); this time refers to the time of the first submission of the task.
   - **nSubmit** - a key whose field is incremented with each submission pertaining to the task,
   - **lastSubmit** - this field value corresponds to the time of the most recent submission and is obtained the same way as described in the **timeStart** case,

2. **jsub:** holds the information pertaining to `jsub` submissions in six fields,

   - **batchReqId** - the ID number assigned to the request by LSF software; this field value is obtained by piping the output of `jsub` command through the ***TMsubmit.pl*** script,
   - **jsubFile** - the field which contains the name of the `jsub` command file which is passed to the submitting script as an argument,
   - **nFiles** - the value of the field is filled by parsing the content of the `jsub` command file from ***TMsubmit.pl***,
   - **jsubStart** - a key whose field contains the time at which the request was submitted to the batch farm.
   - **jsubEnd** - a field containing the time of termination of the last single input file job as reported by `jobstat` run with `-a -l -u user` options.
   - **taskId** - this field ties a given submission with an appropriate task,

3. **farmjob:** the information contained in this table pertains to single input file farm jobs and is mainly retrieved from the `jobstat` command output and stored in ten fields,

   - **farmJobId** - this field stores the ID number of a job as reported by the LSF software; it is retrieved by parsing the output of the `jobstat -a -l -u user` command.
   - **farmStart** - a field filled with the time at which the transition from "PEND" state to "RUN" occurs; obtained from the output of the `jobstat` command.

- **farmFinish** - a field holding the time the job was finished, meaning its state was changed from "RUN" to "DONE" or "EXITED"; again, obtained from the output of the `jobstat` command.
- **farmStatus** - this field contains current state of the job; this one is also obtained from the output of the `jobstat` command.
- **farmHost** - a key whose field contains the name of the node executing the job; also obtained from the output of the `jobstat` command.
- **timeCheck** - in this field the time of the most recent running of the ***TMupdate.pl*** is recorded.
- **timeChange** - the time of any change in the status of any job pertaining to a given submission is inserted/updated in this field.
- **pendReason** - the message on why the job is pending as reported by LSF software is stored in the field; again, obtained from the output of the `jobstat` command.
- **batchReqId** - this field value links a given job with the submission that generated it,
- **bJcacheId** - the numbers contained in this field provide the vital information that links `jcache` requests at batch farm level with `jcache` requests at tape silo level.

4. **jcache:** the information contained in this table corresponds to `jcache` requests generated by a given submission and, as it is implemented, it is stored in six fields and is linked with the appropriate submission by making **bJcacheId-jcacheId** identification,

   - **jcacheId** - a field whose value is obtained by performing a MySQL query on the tape silo JobQueue database, searching for a silo request to match the farm jcache job. Identification is done on the value of the **userName**, type of the tape silo request (which, by definition, is `jcache`), the node handling the request and the requirement that the start time at the farm be approximately the same as submission time at the silo,
   - **bJcacheId** - this field provides the necessary identification with the batch farm jcache job,
   - **jcacheStart** - the start time of a `jcache` request as found by MySQL query of JobQueue database ,
   - **jcacheFinish** - the end time of a given `jcache` request as found by MySQL query of JobQueue database,
   - **jcacheStatus** - the status of the `jcache` request as found by MySQL query of JobQueue database,
   - **jcacheHost** - the name of the node performing the `jcache` request,

5. **silojob:** stores the information on the progress of each silo job; ID's are obtained by querying the JobQueue database,

   - **siloJobId** - this field contains the ID number of a given job,

- **fileName** - the name for the file that should be copied from silo to a local disk; this name should match the appropriate name in the `jsub` command file (this could provide the final check whether the silo `jcache` requests were identified correctly),
- **siloStart** - the start time of a silo job as returned by JobQueue MySQL query,
- **siloFinish** - the end time of a silo job as returned by JobQueue MySQL query,
- **siloStatus** - the state of a silo job as returned by JobQueue MySQL query,
- **siloError** - this field contains the error message in case of failure of the silo job,
- **jcacheId** - the value of this field ties the silo job to the appropriate `jcache` request.

B To-Do List

The issues that, sooner or later, need to be addressed,

- implement `SINGLE_JOB` and `MULTI_JOBS` requirements in the ***TMsubmit.pl*** script,
- ***TMupdate.pl*** script should optionally be called with a task name as an argument rather than a jsub command file name,
- include the input file names in the ***farmjob*** table.

`$Id: taskMaster.tex,v 1.13 2002/10/02 13:45:17 pawel Exp $`