The Bulk Service Version 2 and WLCG TAPE API support

Albert L. Rossi (FNAL)

HELMHOLTZ RESEARCH FOR GRAND CHALLENGES

The dCache Bulk Service (version 2)

The Bulk service was

- introduced in 6.2 as a prototype;
- intended to add processing of multiple targets to the nascent Tape API making it similar to SRM;
- discovered to have a number of issues suggesting it would not scale as written.

The dCache Bulk Service (version 2)

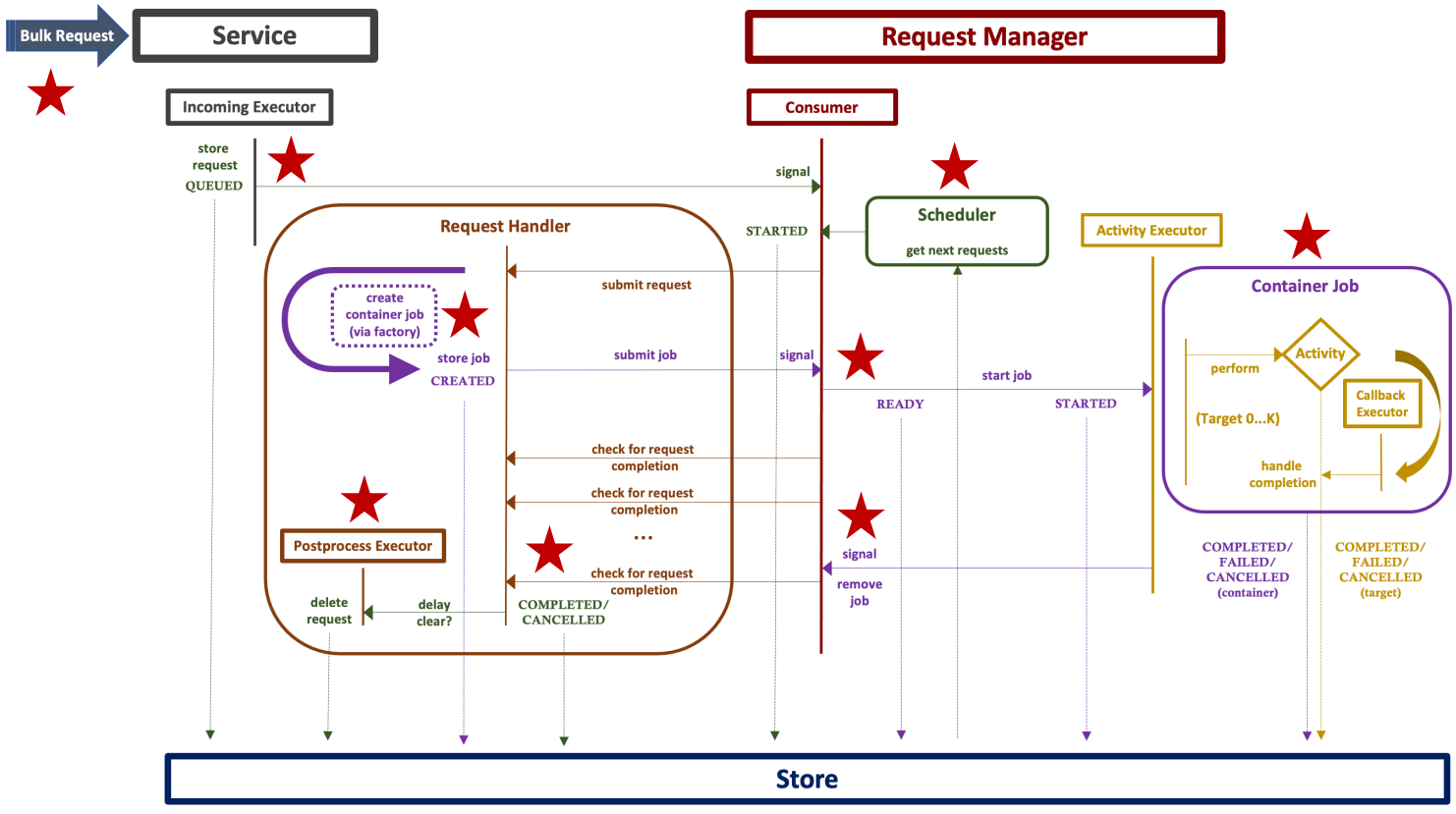
The redesign aims were:

- **better horizontal scaling**;
- **more reliable persistence of requests and idempotent restarts**;
- **more efficient processing of low latency requests like deletion**.

Only internals of the Bulk service proper were affected.

The original REST API for submitting requests remains unmodified (though we also changed it to accept attributes expressed in any of the three forms—camelCase, snake_case, kebab-case).

Bulk Request Handling (Submit)



1. Service receives request message.
2. Service stores unprocessed request, signals manager.
3. Consumer retrieves next requests from scheduler.
4. Request submitted to handler, which creates and stores container job.
5. Job submitted to manager, which launches it using the activity's thread executor.
6. Job processes request targets by:
   a) expanding directories (if indicated);
   b) pre-storing targets (if indicated);
   c) performing activity;
   d) processing completion as callback;
   e) storing and/or updating.
7. Consumer removes completed job from queue.
8. Consumer checks and updates state of requests. If clear is indicated, deletes request.
9. Delayed clear is processed (if indicated).

WLCG Tape REST API

Common interface to

- manage disk residency of tape-stored files;
- observe the progress of files as they are written to tape.

Implementation-agnostic for tape-backed storage systems in WLCG (dCache, StoRM or EOS+CTA).

Document:

https://docs.google.com/document/d/1Zx_H5dRkQRfju3xIYZ2WgjKoOvmLtsafP2pKGpHqcfY

dCache Implementation

REST resource endpoints:

a) General Bulk requests

- https://example.org:3880/api/v1/bulk-requests

b) WLCG TAPE

- https://example.org:3880/api/v1/stage
- https://example.org:3880/api/v1/release
- https://example.org:3880/api/v1/archiveinfo

All are supported by the Bulk service with the exception of *archiveinfo*, which is implemented in the Frontend service itself.

RESTful Resource Features

BULK REQUESTS options	WLCG TAPE
cancelOnFailure	FALSE
Cancel request preemptively on first failure.	
clearOnFailure	FALSE
Remove request from storage if any targets failed.	
clearOnSuccess	FALSE
Remove request from storage if all targets succeeded.	
delayClear	UNDEF
Wait in seconds before clearing (one of the clear options	
must be true for this to have effect). Defaults to 0.	
expandDirectories	NONE
NONE = do not expand directories; TARGETS = shallow	
expansion of directories (only take action on its targets	
with no further expansion); ALL = full recursion.	
Recursion is done depth-first. Defaults to NONE.	
prestore	TRUE
Store all targets first before performing the activity on	
them. (This applies to recursive as well as non-recursive,	
and usually results in significantly lower throughput.)	

RESTful API: Bulk vs WLCG

		BULK	WLCG TAPE
POST*	**Pin**	api/v1/bulk-requests	N/A
		{"activity":"PIN","arguments":{"lifetime":"1","lifetimeUnit": "DAYS","id":"<id>"},"target":["path",...]}	
POST*	**Stage to disk**	api/v1/bulk-requests	api/v1/stage
		{"activity":"PIN","arguments":{"lifetime":"1","lifetimeUnit": "DAYS","id":"<id>"},"target":["path",...]}	{"files": ["diskLifetime":"P1D","path",...]} (*diskLifetime* is ISO 8601)
POST*	**Unpin**	api/v1/bulk-requests	N/A
		{"activity":"UNPIN","arguments":{"id":"<request_id>"}, "target": ["path",...]}	
POST	**Release**	api/v1/bulk-requests	api/v1/release/<request_id>
		{"activity":"UNPIN","arguments":{"id":"<request_id>"}, "target": ["path",...]}	{"paths":["path",...]}
POST*	**Change file QoS**	api/v1/bulk-requests	N/A
		{"activity":"UPDATE_QOS","arguments":{"targetQos": "<new qos>"},"target":["path",...]}	
POST*	**Delete**	api/v1/bulk-requests	N/A
		{"activity":"DELETE","arguments":{"skipDirs":"<true/false>"}, "target":["path",...]}	

(*) The request id is returned with the response.

RESTful API: Bulk vs WLCG

To clarify:

- Both api/v1/bulk-requests with "activity":"PIN" and api/v1/stage result in the submission of the same kind of request to the bulk service; similarly for api/v1/bulk-requests with "activity":"UNPIN" and api/v1/release, though there are some differences in the JSON content description required.
- UNPIN accepts a request for arbitrary files (unpinning all user pins) as well as a specific request id; RELEASE requires the request id.

RESTful API: Bulk vs WLCG

		BULK	WLCG TAPE
PATCH • POST	**Cancel**	api/v1/bulk-requests/<request_id>	api/v1/stage/<request_id>/cancel
		{"action":"CANCEL","paths":["<path>",...]}	{"paths":["<path>",...]}
		If "paths" is undefined, the entire request is cancelled.	*"paths" is required.*
DELETE	**Clear request**	api/v1/bulk-requests/<request_id>	api/v1/stage/<request_id>
		Will fail if request is running.	*Will cancel the entire request first if running.*
GET	**Requests of user**	api/v1/bulk-requests	N/A
		?status=<QUEUED|STARTED|COMPLETED|CANCELLED>[,...]>	
GET	**Request info**	api/v1/bulk-requests/<request_id>	api/v1/stage/<request_id>
		?offset=<next_seq_no>	*Automatically appends by paging to last offset.*
POST	**File locality info**	N/A	api/v1/archiveinfo
			{"paths":["<path>,...]}

NB: the paths given to *archiveinfo* need not belong to a single (or any) request.

RESTful API: Bulk vs WLCG

NB: The JSON returned for the GET differs somewhat:

Bulk Request


```
{
 "nextSeqNo" : 6933633,
 "id" : "0f8f4102-47f1-4b0a-aeda-e40280c2a769",
 "arrivedAt" : 1652712101542,
 "startedAt" : 1652712101551,
 "lastModified" : 1652712369571,
 "status" : "COMPLETED",
 "targets" : [ {
   "target" :
"/pnfs/fs/usr/fermilab/users/arossi/volatile/000/data-
20220509125616521189761113768885",
   "state" : "COMPLETED",
   "submittedAt" : 1652712101670,
   "startedAt" : 1652712101670,
   "finishedAt" : 1652712101672,
   "seqNo" : 6923634
 }, {
 ...
]}
```


WLCG Stage Request


```
{
 "id" : "0f8f4102-47f1-4b0a-aeda-e40280c2a769",
 "createdAt" : 1652712101542,
 "startedAt" : 1652712101551,
 "completedAt" : 1652712369571,
 "files" : [ {
   "path" : "/pnfs/fs/usr/fermilab/users/arossi/volatile/000/data-
20220509125616521189761113768885",
   "finishedAt" : 1652712101672,
   "startedAt" : 1652712101670,
   "state" : "COMPLETED"
 }, {
 ...
]}
```

RESTful API: WLCG

The JSON returned for archiveinfo:


```
[
{"path":"/pnfs/fs/usr/fermilab/users/arossi/tape/517/data-202205091742165213617613760 1559","locality":"TAPE"},
{"path":"/pnfs/fs/usr/fermilab/users/arossi/tape/510/data-202205121931165240186132582 7866","locality":"TAPE"},
{"path":"/pnfs/fs/usr/fermilab/users/arossi/tape/512/data-202205130025165241952663708 1982","locality":"TAPE"},
{"path":"/pnfs/fs/usr/fermilab/users/arossi/tape/512/data-202205122137165240944825148 7828","locality":"TAPE"},
{"path":"/pnfs/fs/usr/fermilab/users/arossi/tape/476/data-202205101136165220057534118 9653","locality":"DISK_AND_TAPE"},
{"path":"/pnfs/fs/usr/fermilab/users/arossi/tape/498/data-202205090937165210702376306 5121","locality":"TAPE"},
{"path":"/pnfs/fs/usr/fermilab/users/arossi/tape/466/data-202205091519165212759527063 8001","locality":"DISK_AND_TAPE"},
 ...
]
```

RESTful API: Swagger

https://example.org:3880/api/v1

bulk-requests

GET	`/bulk-requests/{id}`	Get the status information for an individual bulk request.
DELETE	`/bulk-requests/{id}`	Clear all resources pertaining to the given bulk request id.
PATCH	`/bulk-requests/{id}`	Take some action on a bulk request.
GET	`/bulk-requests`	Get the status of bulk operations submitted by the user.
POST	`/bulk-requests`	Submit a bulk request.

archiveinfo

POST	`/archiveinfo`	Return the file locality information for a list of file paths.

release

POST	`/release/{id}`	RELEASE files associated with a STAGE request.

stage

POST	`/stage/{id}/cancel`	Cancel a STAGE request.
POST	`/stage`	Submit a STAGE request.
GET	`/stage/{id}`	Get the status information for an individual stage request.
DELETE	`/stage/{id}`	Clear all resources pertaining to the given stage request id.

Bulk/QoS

Refactored Resilience service, called QoS, available in 8.1.

Bulk version 2 (8.2), will communicate with this new service.

See QoS service documentation (*The Book*) for its design and features.

Bulk QoS updates (UPDATE_QOS) try to maximize throughput to the Pin Manager in order to take advantage of future tape recall optimization.

(Similarly for Bulk STAGE/PIN requests, without the extra hop through an intermediate service like QoS.)

Bulk/QoS Configuration

Documentation:

- See properties files for the various limits which can be adjusted for both.
- *The Book* is also being updated (QoS is done, Bulk will go in with 8.2).

Both services can be run out of the box.

NOTE

There is no requirement to run QoS, especially if you have not been using the Resilience service. Bulk and QoS are independent, though Bulk uses the latter to request QoS transitions.

Example Bulk/QoS Configuration

Bulk Service

Before startup (first time):

createdb -U <user> bulk

Layout:

[bulkDomain]
dcache.java.memory.heap=4096m
[bulkDomain/bulk]
bulk.allowed-directory-expansion=ALL

see next slide

QoS Services

Before startup (first time):

createdb -U <user> qos

Layout:

[qosEDomain]
[qosEDomain/qos-engine]

[qosVDomain]
dcache.java.memory.heap=8384m
[qosVDomain/qos-verifier]

[qosSDomain]
[qosSDomain/qos-scanner]

[qosADomain]
dcache.java.memory.heap=4096m
[qosADomain/qos-adjuster]

Bulk: Some Properties to Note

```
# ---- Global setting for directory expansion.
#
#       NONE:       do not allow any processing of targets inside a directory (depth 0)
#       TARGETS:    allow only the immediate children of directories to be processed (depth 1)
#       ALL:        allow full recursive processing of directories (depth N)
#
(one-of?NONE|TARGETS|ALL)bulk.allowed-directory-expansion=TARGETS

# ---- Prevents users from monopolizing the service.
#       The limit is in terms of the number of submitted requests
#       which have not yet completed (but not necessarily cleared).
#
bulk.limits.max-requests-per-user=10

# ---- The maximum number of unexpanded targets which can appear in the bulk request list.
#       when request expansion is set to NONE.
#
bulk.limits.max.targets-per-flat-request=100000

# ---- The maximum number of unexpanded targets which can appear in the bulk request list.
#       when request expansion is set to TARGETS.
#
bulk.limits.max.targets-per-shallow-request=1000

# ---- The maximum number of unexpanded targets which can appear in the bulk request list.
#       when request expansion is set to ALL.
#
bulk.limits.max.targets-per-recursive-request=10
```

All of these can be controlled by an admin command:

```
[fndcatemp2] (bulk@bulkDomain) admin > request policy
Maximum concurrent (active) requests     :        110
Maximum requests per user                :         10
Maximum expansion depth                  :        ALL
Maximum flat targets                     :     100000
Maximum shallow targets                  :       1000
Maximum recursive targets                :         10
[
[fndcatemp2] (bulk@bulkDomain) admin > \h request policy
NAME
       request policy -- Change service policy limits.

SYNOPSIS
       request policy [-maxAllowedDepth=none|targets|all]
       [-maxConcurrentRequests=integer] [-maxFlatTargets=integer]
       [-maxRecursiveTargets=integer] [-maxRequestsPerUser=integer]
       [-maxShallowTargets=integer]

DESCRIPTION
       Allows modification of policy configuration without requiring
       domain restart.

OPTIONS
         -maxAllowedDepth=none|targets|all
              Maximum level of directory recursion allowed. NONE = no
              directory expansion; TARGETS = process only the children
              of directory targets; ALL = full recursion.  (Requests
              with expandDirectories set to a stronger value than
              allowed will be immediately rejected.)
         -maxConcurrentRequests=integer
              Maximum number of requests that can be processed at a
              time.
         -maxFlatTargets=integer
              Maximum number of targets a request can contain if the
              expandDirectories attribute of the request = NONE.
         -maxRecursiveTargets=integer
              Maximum number of targets a request can contain if the
              expandDirectories attribute of the request = ALL.
         -maxRequestsPerUser=integer
              Maximum number of queued or active requests a user may
              have at a given time. Requests which have completed but
              have not been cleared/deleted are not counted.
         -maxShallowTargets=integer
              Maximum number of targets a request can contain if the
              expandDirectories attribute of the request = TARGETS.
```

REST API Demo

Thank you for your attention.

Questions? ...