How to enable accounting for jobs submitted with BOINC

- It should be straightforward to create APEL accounting records from BOINC accounting information (maybe just summaries?) and send them off to APEL.
- It would probably involve an agent asking BOINC for accounting info and then creating the APEL records every so often.
- The APEL EGI documentation does describe how it works.
  - For example for the usage record files: https://wiki.egi.eu/wiki/APEL/MessageFormat
  - the ssmsend command which sends them off: https://wiki.egi.eu/wiki/APEL/SSM2AddingFiles
- It is also possible to adapt APEL package to add a new parser parsing BOINC job logs on worker nodes.
  - An APEL client specific for BOINC job accounting can be setup, if necessary
  - Since BOINC job logs are on the worker nodes, the parser needs to be run on each worker node

An adaptation of APEL parser to handle BOINC job logs

An example BOINC job log line

To begin with, here is a log line from the BOINC job log file on a worker node;

```
1581252044 ue 5362.095746 ct 19643.120000 fe 43200000000000 nm VKdLDmsflKwnsSi4apGgGQJmABFKDmABFK
```

which was written by the following boinc code

```
    fprintf(f, "%.0f ue %f ct %f fe %.0f nm %s et %f es %d\n",
        gstate.now, estimated_runtime_uncorrected(), final_cpu_time,
        wup->rsc_fpops_est, name, final_elapsed_time,
        exit_status
    );
```

The information that we can use from the above log are the fields 0, 4, 8,and 10 which will be set as `endTime`, `cpuTime`, `jobName` and `elapsedTime`, respectively.

Records to be filled by the parser

There are two types of parses in APEL. One is to parse job logs and fill EventRecord, the other is to parse accounting logs (*blah logs*) and fill BlahdRecord. `apelparser` does insert or replace collected records to corresponding DB tables. Here are contents of the records to be filled by the parsers.

EventRecord	BlahdRecord
Site	***Site***
MachineName	CE
Infrastructure	GlobalUserName
JobName	***LrmsId***
LocalUserID	GlobalJobId
LocalUserGroup	VO
CpuDuration	VOGroup
WallDuration	VORole
StartTime	FQAN
StopTime	TimeStamp
MemoryReal	ValidFrom
MemoryVirtual	ValidUntil

Processors	Processed
NodeCount	

The fields in ***bold italic*** fonts are mandatory according to APEL/MessageFormat.

Since BOINC job log contains very limited information just enough to fill the mandatory fields, many other non-mandory but essential fields must be filled with some kinds of conventions.

Mandatory fields for site accounting

The following mandatory fields essential for site accounting can be filled with values from BOINC job logs.

Field	Value
Site	site name
StartTime	`endTime - elapsedTime`
StopTime	`endTime`
CpuDuration	`cpuTime`
WallDuration	`cpuTime`

Some fields also can be set easily with values from job logs, and other fields can be filled with arbitrary values as described in following sections.

A note about `WallDuration`, `Processors` and `NodeCount`

According to "Definitiopn of the Compute Accounting Record", `WallDuration` is the ***elapsed time*** regardless of number of cores, processors, etc. But due to the nature of the BOINC jobs, which run with high nice value, it's not easy to calculate proper estimation of how much system was dedicated to BOINC jobs; certainly, `cpuTime * nCores` would be an overestimation. An easy way is to use `cpuTime` as `WallDuration` and set `Processors` to 1 or None.

The job runs on a single node, NodeCount can be set to 1 too.

Field	Value
WallDuration	`cpuTime`
Processors	1
NodeCount	1

Other essential non-mandatory fields that can be filled with data from job logs

The following fields can be filled with the values on the right column.

Field	Value
LocalUserId	local user name for boinc jobs (eg, *boinc*)
TimeStamp	`endTime`
ValidFrom	`valid_from(endTime)`
ValidUntil	`valid_until(endTime)`
Processed	`Parser.UNPROCESSED`

A set of conventions for the other fields

The remaining feilds except for `JobName` can't be determined by boic job logs itself. Thus they must be set with arbitrary values. Here is a set of conventions used in this adaptation.

1. JobName, LrmsId and GlobalJobId

A boinc job name (`jobName`) is already a global id so it can be used as `GlobalJobId` as it is. It can also be used as `LocalJobId` (which is `JobName`) but it would be useful to add woker node information to `LocalJobId`. But `JobName` is `VARCHAR(60)` while `jobName` is 56 chars so it needs to reduce it to combine worker node with it to build `LocalJobId`.

A simple method is to concatenate worker node name and truncated `jobName`. If necessary, `endTime` can be added too to ensure uniqueness of LocalJobId. For example,

```
LocalJobId = JobName = shortHostName + '.' + endTime + '.' + jobName[:N]
```

Note that `LrmsId` and `JobName` must be the same for the same job so the same naming convention must be applied to `LrmsId`.

2. MachineName and CE

`EventRecord.MachineName` and `BlahdRecord.CE` become `MachineName` and `SubmitHost` in job messages to be sent to APEL server, respectively. `MachineName` seems not being used anywhere so it can be named arbitrarily.

`SumitHost` is used in grouping jobs to normalize their cpu and wall times with given spec values. Even though it's possible to use one of existing submit host names for `CE`, it would be better to define a new submit host name for BOINC jobs. Its spec type and spec value can be configured in `client.cfg` file.

A simple solution is to use the name of the APEL client node publishing BOINC accounting messages to APEL server. For example,

```
boinc.lcg.trumf.ca
```

3. Infrastructure

According to APEL/MessageFormat wiki, it is `<accounting client>-<CE type>-<batch system type>`. CE and batch system types for BOINC jobs are not well-defined so we may assign arbitrary type names, for example,

```
APEL-BOINC-BOINC
```

Configuration of the above fields

Values of the above fields can be configured in config files; **`boinc-acc.cfg`**, `parser.cfg` and `client.cfg`.

- <boinc-acc.cfg>

```
[blah]
# name to be used in <client.cfg>
#
#   [spec_updater]
#   manual_specX=<ce>,<spec_type>,<spec_value>
#
ce =

# submitter of the jobs
#  note : setting 'dn' to 'null' or 'none' as described in <APEL/MessageFormat>
#         raises the following error:
#
#dn=null  -->  Error loading records: (1048, "Column 'name' cannot be null")
#
dn = <host_dn>
```

```
fqan = /atlas/Role=NULL/Capability=NULL

[batch]
local_user_id = boinc
#local_user_group =
```

- <parser.cfg>

```
[site_info]
site_name = <site>
lrms_server = <boinc_ce_name>

[blah]
dir = /var/log/apel/accounting
filename_prefix = boinc_blahp.log

[batch]
type = BOINC
dir = /var/log/apel/accounting
filename_prefix = boinc_jobs_logs.txt
```

- <client.cfg>

```
[spec_updater]
site_name = <site>
manual_spec1 = <boinc_ce_name>,<spec_type>,<spec_level>
```

To make it simple, <boinc-acc.cfg/blah/ce> is used for both <parser.cfg/site_info/lrms_server> and <client.cfg/spec_updater/manual_spec1>.

Feilds left undefined

The following non-mandatory fields are left undefined in this adaptation

- EventRecord
  - LocalUserGroup
  - MamoryReal
  - MemoryVirtual
- BlahdRecord
  - GlobalUserName

An implementation of the above adaptation

An implementation of the above adaptation can be found at here.

Example configuration files, boinc-acc.cfg, parser-boinc.cfg and client-boinf.cfg, can be found under conf dir.

Once all the parsed log results are loaded to DB on the APEL client, **apelclient** will combine them to create job records and send relevant accounting messages to the APEL server. It is ***recommended*** to send ***summary messages*** instead of *individual job messages*.

Example messages generated by `apelclient` with the above configuration

An individual job message

```
APEL-individual-job-message: v0.3
Site: TRIUMF-LCG2
SubmitHost: boinc.lcg.triumf.ca
MachineName: boinc.lcg.triumf.ca
Queue: None
LocalJobId: wns0010.077NDmViIGwnsSi4apGg
LocalUserId: boinc
GlobalUserName: <host_dn>
FQAN: /atlas/Role=NULL/Capability=NULL
VO: atlas
VOGroup: /atlas
VORole: Role=NULL
WallDuration: 23341
CpuDuration: 23341
Processors: 1
NodeCount: 1
StartTime: 1580216440
EndTime: 1580224397
InfrastructureDescription: APEL-BOINC-BOINC
InfrastructureType: grid
MemoryReal: None
MemoryVirtual: None
ServiceLevelType: HEPSPEC
ServiceLevel: 21.69
%%
```

A summary message

```
APEL-summary-job-message: v0.2
Site: TRIUMF-LCG2
Month: 1
Year: 2020
GlobalUserName: <host_dn>
VO: atlas
VOGroup: /atlas
VORole: Role=NULL
SubmitHost: boinc.lcg.triumf.ca
InfrastructureType: grid
ServiceLevelType: HEPSPEC
ServiceLevel: 21.690
NodeCount: 1
Processors: 1
EarliestEndTime: 1578565351
LatestEndTime: 1580508691
WallDuration: 4021949
CpuDuration: 4021949
NumberOfJobs: 176
%%
```

Note that the above record is only for a single worker node that a new parser was tested.

Standalone application

It is possible to write a standalone version independent of APEL package.

-- JuliaAndreeva - 2018-07-05

This topic: LCG > AccountingForBoinc
Topic revision: r5 - 2020-03-31 - unknown