Update on SixDesk Scripts

R. De Maria, M. Giovannozzi, E. Mcintosh, A. Mereghetti, I. Zacharov
Outline

- SixDesk;
- BOINC;
- Motivations for update;
- Updates currently implemented;
- Conclusions and outlook;
run environment of SixTrack for DA studies:

- Scripts for pre-/post-processing: from MadX .mask files to final plots;
- Structured user space, containing scripts and templates;
- Collection of standard .mask files;

Main parameter scan:
- Amplitude/angle in x-y plane;
- $Q_x/Q_y$;
- WISE errors (magnetic errors);

Studies imply massive tracking campaigns:
- **High CPU demands**: 100k / 1M turns through LHC / HL-LHC / FCC lattices;
- **Low I/O per simulation**:
  - input files generally small;
  - output basically limited to final coordinates of particles in phase space;
- **Massive number of jobs**: a single job allows to obtain results on few points in the phase/parameter space;

→ These characteristics make DA studies suitable for being run not only in LSF but also in BOINC.
**BOINC**: Berkley Open Infrastructure for Network Computing

System which uses the idle time of private computers to make CPUs available to scientists.

Computing nodes are typically not those located in a lab/university, but private home’s/offices’ desktops!

Work flow:
1. Submission: a user submit a set of WUs to the BOINC server, which distributes them (with the exe) to volunteers’ PCs;
2. Computing: volunteers PCs perform the calculations; at the end, they give back results to the BOINC server;
3. Returning results: the BOINC server performs a validation (math stability) and gives results back to user;

![Diagram of BOINC system](image-url)
The BOINC server takes responsibility for the flow of WUs:

- User’s side: upload/download to/from a shared AFS volume (spooldir);
- Volunteer’s side: upload/download area;
- Internal DB, for keeping track of Wus (e.g. submitted/done/re-sent);
The BOINC manager on the volunteer’s pc takes responsibility of:

- Download new WUs / upload results;
- Scheduling tasks and allocating necessary resources (CPU, RAM, HD, …);

Many configurations available to volunteer: choice of projects, number of CPUs, when to run, pause/resume, …

…CERN desktops can be configured as volunteers’ PCs!
Motivations for Update

- **Main drivers (BE-ABP):**
  - Possibility to run *collimation* studies on BOINC:
    - Relevant for eLens cleaning simulations;
    - Requirement: online aperture check (work in progress);
  - Extend parameter space (e.g. dp/p, chroma/emittance, collimator scans, ...);

- **Contingency (IT infrastructure):**
  - AFS work volume giving problems on a *daily* basis;
  - AFS phasing out and EOS rising (“FS” behind CERNbox);
  - HTCondor coming along (production in 18 months);

- **At the same time:**
  - Decouple user space from scripts and templates;
    - When scripts are updated, the users do not have to go through all their workspaces to update all local versions of scripts!
  - Centralize scripts, to ensure that all users receive updates (and use them!);
  - (a bit of) code refactoring (enhance maintainability);

Since AFS volume gives problems and IT infrastructure will undergo relevant changes in the near future, *present set-up must work!*
Issues with AFS work.boinc Volume: Work Flow

For every WU:
- `ls setacl -dir $spooldir/<studyname>`
- Gunzip/gzip MadX files (WISE);
- * .zip (all input files);
- * .desc (boinc-specific settings);

**AFS spool dir**
/afs/cern.ch/work/b/boinc/boinc/<studyname>/
  /_ work/
  /_ results/
  /_ owner

**User’s HOME dir**

Retrieval of results: run_six
`find $spooldir/<studyname>/results ~name …`

**BOINC server**

cron job:
- `find $spooldir -maxdepth 2 -type d -name “work”`
- For every WU found, call create_work (actual submission to volunteers PCs…)

- Bitwise validator, for math stability (local disk);
- Assimilator: `cp` results back to AFS spooldir;

Dispatching results
Example of Load of BOINC Server

(~1h refresh rate);

**Initial guess**: problems are at the submission level:
- Expiration of AFS tokens → scripts hanging;
- Too high load of meta-data (e.g. recursive setacl commands);

**Monitoring** of recent activity: problems on the work.boinc volume can also start independently of submission…

Mitigation actions taken at **every step** involving the AFS work.boinc volume, i.e. at job submission / results retrieval, and on user / boinc server side!
Improving the Situation: Changes at Submission

For every WU:
- `fs setacl` –dir $spooldir/<studynamename>
- Gunzip/gzip MadX files (WISE);
- *.zip (all input files);
- *.desc (boinc-specific settings);

Submission of WUs:
- `fs setacl` limited only to setting up of spooldir (and not at every WU);
- Gunzip/gzip of MadX files done only ones, i.e. before/after a WISE configuration is used to generate all dependent WUs;
- Instead of a ‘find’ looking for new WUs, submit through a file list:
  1. generate the list of WUs to be submitted and save it to a .txt file;
  2. the cronjob (on the BOINC server) parses the .txt file and submit the listed WUs;

A.F.S spool
/afs/cern.ch/work/b/boinc/<studynamename>/
  |_ work/
  |_ results/
  |_ owner

cron job:
- `find` $spooldir –maxdepth 2 –type d –name “work”
- For every WU found, call create_work
Improving the Situation: Changes at Returning Data

Returning results:

✅ Assimilator: introduce a trash-bin mechanism (collab. I. Zacharov):
  - Avoid orphan results;
  - Avoid multiple trials to deliver results to an AFS dir with too many files (AFS load);

❌ Retrieval of results from spooldir: simplify find instruction;

❌ Use a zip mechanism:
  - Save results locally on BOINC server;
  - zip them according to study;
  - periodically dispatch zipped files to spooldir;

✅ Cron job:
  - find $spooldir/<studyname>/results –name …

(Bitmap validator (local disk);

Assimilator cp results back to AFS spooldir;
Updating Scripts

SixDesk (GitHub repo):

| __ sixjobs/ |
| __ sixutils/ |
| __ utilities/ |
| __ boinc_software/ |

User space + original scripts (3 copies!!)
Original scripts + other exes
Updated scripts (branch: isolateScripts)
Useful BOINC software (branch: add_boinc_soft)

utilities/
| __ awk/ |
| __ bash/ |
| __ bats/ |
| __ exes/ |
| __ fortran/ |
| __ gnuplot/ |
| __ perl/ |
| __ python/ |
| __ sed/ |
| __ templates/ |
| __ tex/ |

- All existing scripts and templates have been collected in a folder separate from the user space (sixjobs/);
- Scripts are collected based on the coding language;
- ksh and sh scripts all moved to bash;
- any (bash) script called by the user on the terminal line automatically detects the full path where it is located and headed it to any sub-script;
- No removal of ‘old’ stuff, not to break compatibilities at this stage;
Updating Scripts (II)

Two scripts already reworked quite deeply:
• mad6t.sh: generation of fort.2/.8/.16;
• run_six.sh: generation of fort.3 and submission;

Main features:
• All in bash;
• Actions / input parameters are treated via getopt (no options triggers help);
• Instead of doing everything, the user triggers actions:
  • mad6t.sh: generation of input files vs consistency check;
  • run_six.sh: generation of input files vs consistency check vs actual submission;

reasons:
• Reduce running time of a single action, to decrease probability of AFS token expiration;
• All the code related to a given step is collected in only one script, segmented in functions (e.g. original run_six and called scripts);
• General refactoring of code:
  • Improve readability;
  • Reduce number of lines;

Started with these ones since initial guess: problems are at the submission level:
• Expiration of AFS tokens → scripts hanging;
• Too high load of meta-data (e.g. recursive setacl commands);

mad6t.sh (329, -33%): run_mad6t (38) + dorun_mad6t (197) + check_mad6t (253);
runt_six.sh (1624, -25%): run_six (1350) + dot_boinc (269) + dotbsub (229) + uploadWorkunit (168) + prepare_fort3 (220);
Updating Scripts (III)

A common space where to keep the scripts (actually, git clones) has been created in:

/afs/cern.ch/project/sixtrack/SixDesk_utilities/

- _dev/_
- _pro/_
- _old/_

- Updated scripts (ongoing tests)
- Empty for the time being

Some preliminary (until changes go in release) documentation is available on the SixDesk twiki:
## Updating Scripts (IV)

**mad6t.sh –s**
- Submission to LSF;
- 60 WISE seeds: 3m for submission, 15m LSF jobs;

Next test: 120 kWUs…

<table>
<thead>
<tr>
<th></th>
<th>60seeds x 4amplis x 5angles</th>
<th>60seeds x 4amplis x 50angles</th>
</tr>
</thead>
<tbody>
<tr>
<td>run_six.sh –g</td>
<td>Generation of input (+check)</td>
<td>15m</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1h</td>
</tr>
<tr>
<td>run_six.sh –c</td>
<td>Check of input</td>
<td>1.5m</td>
</tr>
<tr>
<td></td>
<td></td>
<td>x10</td>
</tr>
<tr>
<td>run_six.sh –s</td>
<td>Actual submission (+check beforehand)</td>
<td>3m 40s</td>
</tr>
<tr>
<td></td>
<td></td>
<td>x12</td>
</tr>
</tbody>
</table>

**acrontab_retrieve_boinc_results.sh**
- *bash* script used as acrontab job (AFS);
- Based on simple example by E. McIntosh *acrontab.entry*;
- Retrieves results *in series* (avoid AFS overload of work.boinc volume);
- Enhanced *user interface*, i.e. user edits a list of studies;
- Meant to be run periodically (even if no studies are run), to avoid WUs rejected due to AFS limits;
Conclusions and Outlook

SixDesk scripts are being updated:

- **Short term** changes are meant to maintain the present set-up reliably working, mitigating issues with AFS work.boinc volume;
  - Limit impact of setacl at submission level;
  - Split submission steps to avoid long processes (expiration of AFS token);
  - Handle orphan results;
  - Improve run_results;
  - Implement submission to BOINC by file list;
  - Implement zipping of results and automatic dispatching back results;

- **Longer term** changes are meant to:
  - Include use of SixDB (R. De Maria);
  - Run collimation studies on BOINC;
  - Extend scanning parameters – to be noted:
    - Crossing angle (by D. Pellegrini) to be imported in updated scripts;
    - FCC tunes to be tested (J. Barranco) and then imported in updated scripts;

- Changes are also required to accommodate future modifications to the IT infrastructure:
  - HTCondor as queue system (collaboration with L. Field);
  - EOS as possible system where spooldirs are located (chosen by user);
  - In parallel: code refactoring (e.g. BNL flag, sussix, ...) and documentation;