BOINC: The Year in Review

David Anderson

31 Aug 2010
Credit: goals

**Device neutrality**: a job should get the same credit no matter what device processes

**Project neutrality**: a computer should get the same credit/day regardless of what project(s) it runs

(easy to show that these can’t both be achieved)
1st credit system

- CPU time x CPU benchmark
  - not device neutral
- Replication and credit averaging
  - granted credit depends on partner
2\textsuperscript{nd} credit system

- “Actual FLOPS”-based
- APIs for reporting FLOP counts
- SETI@home publishes average credit/CPU sec, other projects scale to match

Problems:
- most apps can’t count FLOPs
- doesn’t address GPUs
- no device neutrality
- doesn’t prevent cheating w/ single replication
Philosophy of 3rd system

- Credits is based on peak FLOP count (PFC)
  - \[ PFC(J) = \#\text{CPUs} \times \text{CPU benchmark} \]
    + \[ \#\text{GPUs} \times \text{GPU rated FLOPS} \]
  - Reflects “opportunity cost”, not actual work
- Normalize in 2 ways
Statistics

- Maintain mean, variance of $PFC(J) / WU.fpops_{est}$ for each:
  - app
  - app version
  - (host, app version)
Normalize to most efficient app version

mean PFC

CPU/Win
CPU/Mac
CPU/Linux
GPU/Win

Note: this provides device neutrality at the expense of project neutrality
Host normalization

• Scale PFC for version V, host H by $\frac{V.pfc_{avg}}{H.pfc_{avg}}$

• Provides cheat-resistance even with single replication

• but need to prevent cherry-picking: don’t use host normalization unless host has returned N consecutive valid results
GPU-only projects

• On a project with both CPU and GPU versions, version normalization provides a measure of relative efficiency CPU vs. GPU

• Projects with only GPU apps don’t have this

• Solution: such projects scale by the weighted averages of projects that do
Experience

• New system tested in SETI@home
• Works, but need to double credit (redefine Cobblestone)
• No project customization
Job runtime estimation

• Old system:
  \[ R(\text{est}) = \frac{\text{WU.fops\_est}}{\text{CPU benchmark}} \]

• Maintain and scale by a project-wide “duration correction factor”

• Problems:
  • bad if multiple versions
  • scientists shouldn’t think about FLOPS
  • doesn’t work for GPUs
New system

- Maintain mean, variance of normalized elapsed time for each (host, app version)
- Predicted runtime = mean * WU.fops_est (per-app-version duration correction factor)
Other per-(host, app version) items

- Daily quota (for host punishment)
- Consecutive valid results: replaces error rate for
  - “reliable” mechanism
  - cherry-picking prevention
Notice system

- How does the BOINC client software communicate with volunteers? Currently: the Messages Tab. Problems:
  - Requires user to look
  - Non-prescriptive techno-babble
  - Only bad news
  - Only text
  - Non-translatable
Notices architecture

- Multiple “notice” sources
  - from client
  - from schedulers
  - RSS feeds from projects
    - project news
    - private messages
    - friend requests
    - messages in subscribed threads
    - ...
Notice delivery

- System tray popup
- Notices tab
GPU support

- Exclusive apps
- Show GPU projects in attach wizard
- Snooze/suspend/resume GPU
  - app_plan(): specify GPU RAM requirements
    - use in scheduling; boinc_temporary_exit()
- Sample CUDA/OpenCL apps
- Support Fermi GPUs
Multithread app support

- boinc_init_parallel()
  - suspend/resume multiple threads
- show projects in attach wizard
Other goodies

- GUI RPC as HTTP
  - enable GUIs based on web technologies

- Web: project news as a message board
  - easier to post
  - users can discuss

- Preferences
  - Transfer at most X MB every N days
  - suspend if non-BOINC CPU load exceeds X
More goodies

• Stuff for Intel PtP
  • web-based registration (manager finds cookie)
  • HTTP proxy autodetect

• Server logging
  • <debug_xxx> flags instead of -d 3
  • -d 4 means print DB queries
Upcoming

- Rewrite or replacement of Simple View
  - or entire Manager?
- VM app support
  - BOINC installer includes VirtualBox?
- Volpex
  - IPC for BOINC apps
  - virtual cluster
- Integration with Drupal
What we didn’t do

- Integrate remote job submission system from GPUGRID
- Accelerated batch completion
Adoption by scientists

- Single-scientist projects: a dead end
  - Barriers to entry are too high
  - Wrong marketing model
  - Doesn’t handle sporadic requirements
Adoption by scientists

- Most scientists outsource HPC decisions to IT people
- IT people fear and loathe volunteer computing

Napoleon: Volunteer computing just can’t handle the kinds of jobs that *real* scientists run.

Me: What precisely is different about these jobs?

Napoleon: THEY’RE JUST DIFFERENT, THAT’S ALL
A way forward

Distinguish:

- Project operation
  - operate servers
  - port apps, interface with scientists
- Marketing
  - branding/strategy
  - mass media, online, non-traditional
  - web development
  - make bundling deals with computer/OS vendors
Project == existing HPC provider

- Supercomputer centers
- National grids (Teragrid, OSG)
- Hubs
  - “Facebook + iPhone app store” for science area
  - e.g. Nanohub
  - HUBzero/BOINC integration proposal
A consortium of funding agencies and HPC providers

Unified brand, web site for scientific volunteer computing in U.S.; implemented using account manager mechanism

Volunteers choose research areas, not projects

Committee of consortium members allocates computing power among projects
- How to realize this?
- European/Asian counterparts?
Summary

- Volunteer computing has not approached its potential
- There are still many skeptics
- Let's keep working