The 9th Annual BOINC Workshop

25-27 September 2013

INRIA, Grenoble, France

http://boinc.berkeley.edu/trac/wiki/WorkShop13

The BOINC Community

Projects

UC Berkeley developers (2.5)

Computer scientists

PC volunteers (240,000)

Other volunteers:
testing
translation
support

Workshop goals

- Learn what everyone else is doing
- Form collaborations
- Get ideas
- Steer BOINC development

Hackfest (Thu/Fri)

- Goal: get something done
  - design and/or implement software
  - improve docs
  - learn and use a new feature
- Bring your ideas

The state of volunteer computing

- Volunteership: stagnant
  - 240K people (down from 290K)
  - 350K computers
- Science projects: stagnant
- Computer Science research: a little
- Let's keep trying anyway

Requests to projects

- Do public outreach
  - Notices (with pictures)
  - Automated reminder emails
  - News emails
  - Message boards
  - Mass media
- Use current server code
  - Avoid code divergence

To developers/researchers

- Talk with me before starting anything
  - especially if it’s of general utility
- Let me know if you need data

What's new in BOINC?

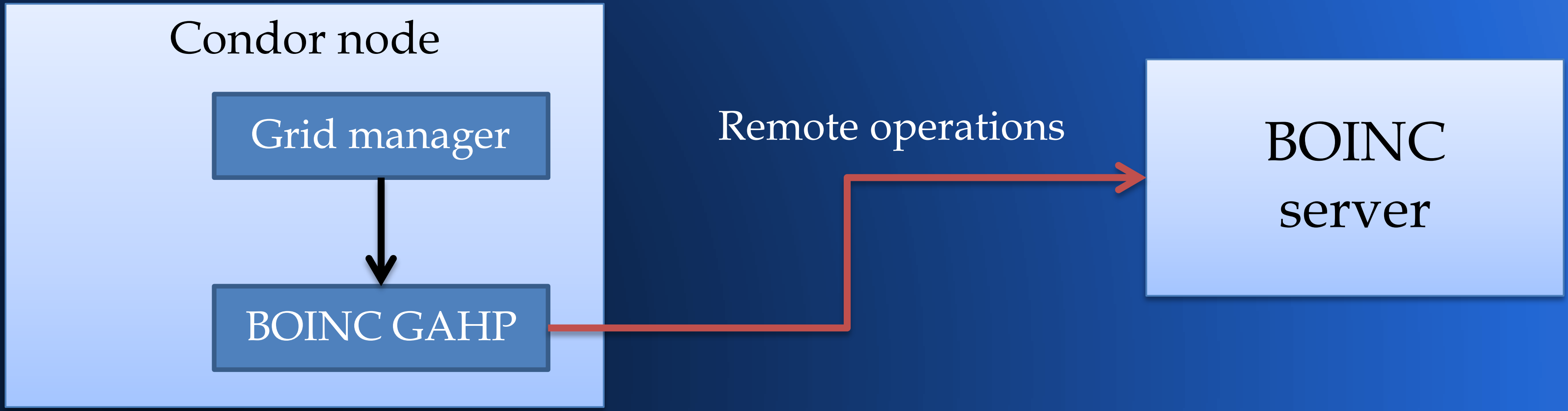
- Funding
- Integration projects
- Remote job and file management
- Android
- Scheduler
- GPU and multicore apps
- Client
- Plans

Funding

- Current NSF grant runs another 18 months
- Not clear if current model will continue
- Collaborations are important for future funding
- Projects may need to help fund BOINC directly

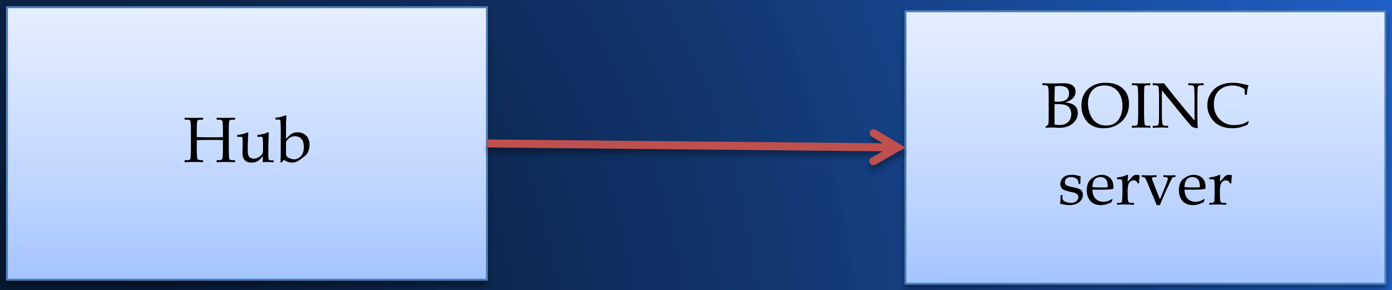
Integration projects

- HTCondor (U. of Wisconsin)
  - Goal: BOINC-based back end for Open Science Grid or any Condor pool



Integration projects

- HUBzero (Purdue U.)
  - Goal: BOINC-based back end for science portals such as nanoHUB



Integration projects

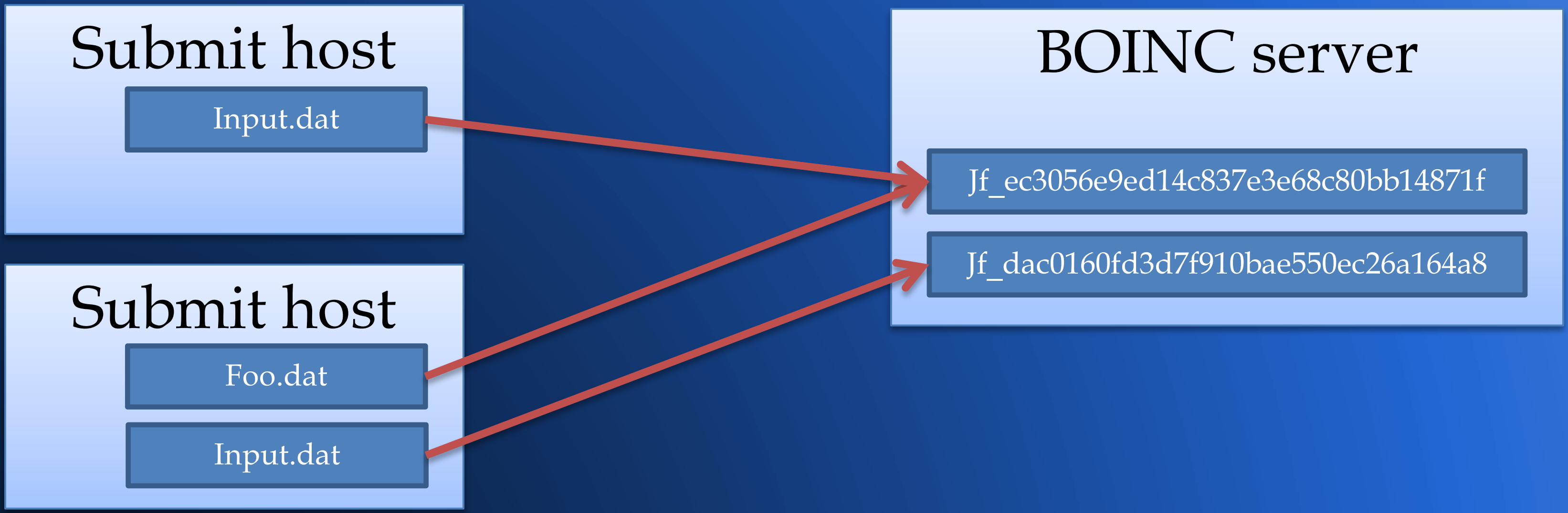
- Texas Advanced Computing Center (TACC)
  - Android/iOS app
  - They supply
    - Interfaces, visualization, support for scientists
    - Storage
    - BOINC server

Remote input file management

- Issues
  - Naming/immutability
  - Efficiency
  - Garbage collection
- User file sandbox (web-based) used by CAS

Content-based file management

- Server file names based on MD5


Submit host
Input.dat
Submit host
Foo.dat
Input.dat
BOINC server
Jf_ec3056e9ed14c837e3e68c80bb14871f
Jf_dac0160fd3d7f910bae550ec26a164a8

- DB table for file/batch association; garbage collection
- Web RPCs to query lists of files, upload files

Remote job submission

- Web RPCs
  - Batch: estimate, create, query, abort, retire
  - Batch expire time
  - Job: query, abort
  - App: get templates
- Input file modes
  - Local, local-staged, semilocal, remote, inline
- C++, PHP bindings

Output retrieval

- Web RPCs to
  - Get specific output files
  - Get zip of job’s outputs
  - Get zip of batch’s outputs

BOINC on Android

- New GUI
- Battery-related issues
- Device naming
- Released July 22
  - Google Play Store, Amazon App Store
  - ~30K active devices

Job size matching

- Problem: 1000X speed difference GPU vs Android
- An app can have jobs of N “size classes”
- “size_census.php”: computes quantiles of effective speed for each app
- Scheduler tries to send jobs of size class *i* to devices in quantile *i*
- “size regulator” makes sure jobs of all size classes are available to send

New score-based scheduler

```
for each resource type (starting w/ GPUs)
    scan job array starting at random point
    make list of jobs with app version for resource
    assign score (include job-size term)
    sort list
    for each job in list
        do quick checks
        lock array entry, do slow checks
        send job
        if request satisfied, break
```

BOINC client

- New work-fetch, job scheduling
  - Handle GPU exclusions
- “App config” mechanism
  - User can set device usage parameters, limit # of concurrent jobs per app
- Maintain/report current, previous uptime
- Maintain list of completed jobs
- Sub-second CPU throttling

GPU and multicore apps

- Support Intel GPUs
- Support OpenCL CPU apps
  - Detect, advertise multiple OpenCL libraries
- Develop OpenCL example app
- Detect GPUs in a separate process
  - Mac notebooks: allow system to use low-power GPU

BOINC runtime system

- Replace heartbeat with PID check
  - Not on Win2K: PID reuse
- Support apps that are in a critical section most of the time (e.g. GPU apps)

Volunteer storage

- Finished data archival system
  - Store large files for long periods
  - Multi-level erasure coding
- Developed simulator for testing, performance study

Software engineering

- Finished SVN → git migration
- Automated translation process
  - build_po → Pootle → commit → deploy
- Code hardening
  - strcpy() → strlcpy()
  - MAXPATHLEN

Didn’t start

- OpenID/OpenAuth support
- Remodel computing preferences
- BOINC in app stores (Windows, Apple)

Planned

- Automated build/test using Jenkins
  - Server code release management
- Accelerated batch completion
- Apple iOS client

My wish list: new GPU design

- Current: all GPUs of a given vendor are equivalent
  - Scheduler requests ask for NVIDIA jobs, not jobs for a specific NVIDIA GPU
  - This doesn’t work well for machines with heterogeneous GPUS
  - Work-arounds (GPU exclusions) cause problems
- Proposed: treat each GPU as a separate resource

My wish list: fully embrace latency-oriented scheduling

- Types of workload
  - Throughput-oriented
  - Small/fast batches
  - Large/slow batches
- Suppose a project has all three?
  - Goal: client requests and processes short jobs even if fast jobs are in progress
  - Requires complete redesign of scheduling policies

Science@home

- The “project ecosystem” hasn’t materialized
  - Creating a project is too difficult, too risky
  - Volunteers tend to be passive
  - Marketing and PR: too many brands
- Umbrella projects: good, but not enough

Science@home

- Single “brand” for volunteer computing
- Register for science areas rather than projects
- Facebook/Google login
- Use account-manager architecture
- How to allocate computing power?
  - Involve the HPC, scientific funding communities