BOINC
The Year in Review

David P. Anderson
Space Sciences Lab
U.C. Berkeley

12 Sept 2008

State of the paradigm: the good, the bad, and the ugly

- PetaFLOPS barrier broken
  - Folding@home: Sept 07
  - BOINC: Jan 08
- Apps for Cell, GPU
- Mobile computing
  - BOINCoid
  - S@h on ARM

More good stuff

- New organizational models
- Interesting research
  - VM-based apps
  - Communicating apps, P2P data distribution
  - simulators
  - availability studies
  - master/worker programming
- Active community
  - support, project help, testing

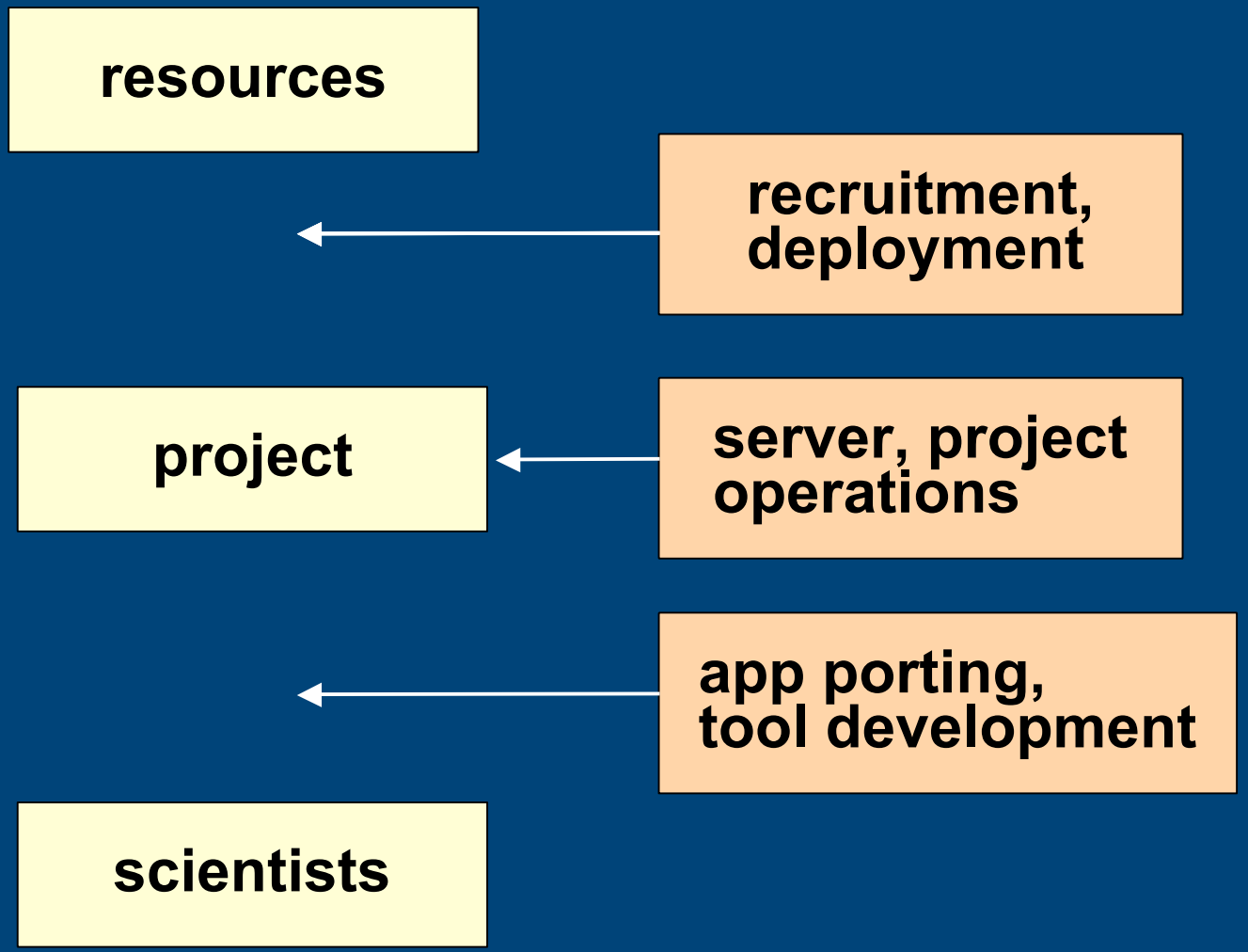
The bad

- Volunteer population is stagnant
- Few new projects
- No significant PR
- Folding@home/SETI@home monopoly
- Berkeley@home not funded
- Volunteer computing still unknown to
  - HPC world
  - scientific computing world
  - general public

The ugly

- “After 17 years, volunteer computing has yet to make a single scientific discovery.”

Organizational models



The single-scientist model

- Dead end?

Meta-projects

- IBM World Community Grid
  - solicitation of applications
  - partner organizations
  - corporate partication
- Campus-level (example)
  - 1,000 instructional PCs
  - 5,000 faculty/staff
  - 30,000 students
  - 400,000 alumni
- Center-level
  - Lattice: U. Maryland Center for Bioinformatics
  - LHC@home: CERN

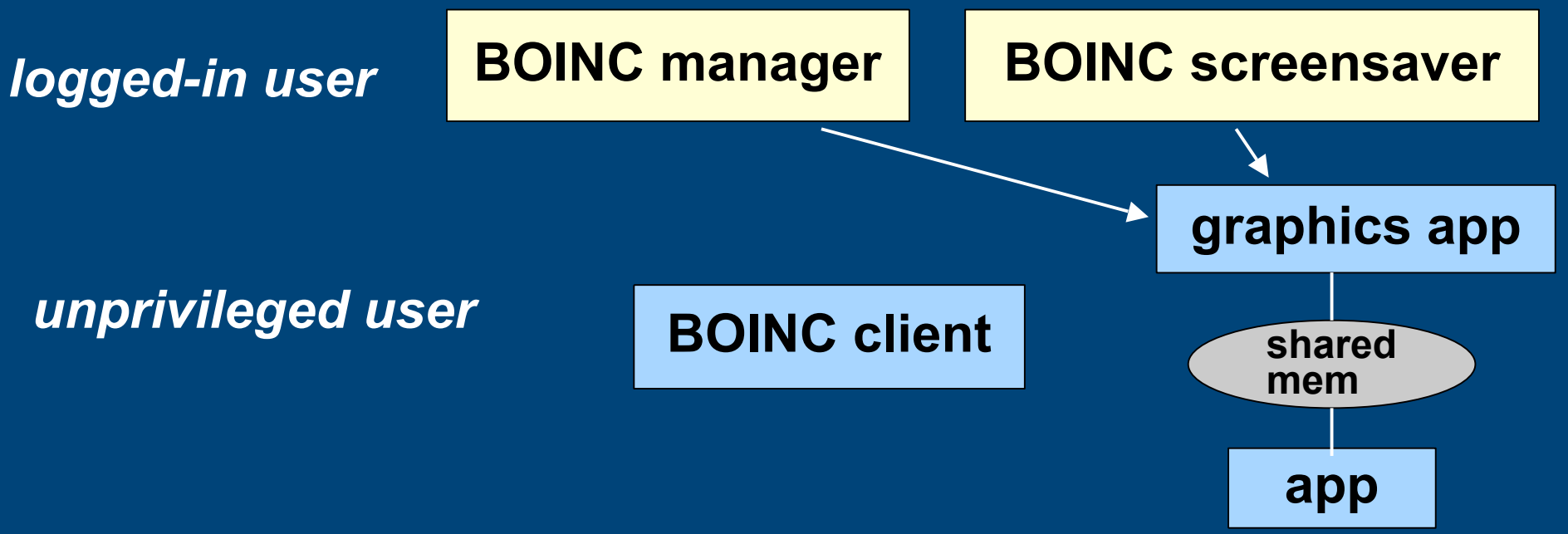
Other models

- MindModeling.org
  - ACT-R community (~20 universities)
- Ibercivis
  - consortium of universities and labs
  - separate projects but unified resources
- EDGeS (SZTAKI)
  - hierarchical desktop grids
  - EGEE@home?
- Almere Grid
  - community computing

Software progress report: client

- Account-based sandboxing on Windows
- Weak account keys
  - for mass deployment on insecure computers
- # CPUs preference is now a fraction
  - prepare for 80-core CPUs

New graphics architecture



- Graphics crashes don’t affect app
- Graphics work with account-based sandboxing

BOINC Manager

- RPCs in separate thread
  - GUI doesn’t block waiting for RPCs
- Combined grid/regular views
- Multiple selection
  - e.g. abort 10 jobs at once

Server scheduling policy (old)

- Multiple passes through shared-memory array
  - if “reliable” host, send retry jobs
  - if beta-test user, send beta-test jobs
  - send jobs already committed to host’s HR class
  - send other jobs
  - if no jobs yet, allow apps not selected by user
- Problems:
  - fixed ordering of scans
  - inefficient

Server scheduling policy (new)

- Score-based scheduling
  - Scan at least N, at most M jobs in array
  - send those with highest “score”
- Score function:
  - bonus for sending beta jobs to beta users
  - bonus for sending retry jobs to reliable hosts
  - bonus for send HR-committed jobs
  - score += expected GFLOPS
    - favor sending jobs with GPU apps to hosts with GPUs
  - score –= (job size – host speed)^2
    - favor sending large jobs to fast hosts

Other new server features

- Assigned jobs
  - to a particular host
  - to all hosts of a user or team
  - to all hosts
- Can limit jobs by download bandwidth
- Job-size matching
  - “census” computes host statistics
  - feeder maintains job statistics
- Support “optional” files in validator framework

Adaptive replication

- Goal: trusted results with 1+ε replication
- Policy:
  - maintain error rate E(h) of each host
  - if E(h) > X, replicate (e.g., 2-fold)
  - else replicate with probability E(h)/X
- How to use:
  - workunit.target_nresults = 1
  - app.target_nresults = 2
- Can this be gamed?
  - need to hide jobs in progress on web
- Should use “invalid rate” instead of error rate?

Teams

- Team message boards
- Team admins
- Team membership history
- Multi-criteria team search
  - keywords, country, type
- Team finding in signup process
- User search (for team recruitment)
  - filter by country, presence of team/profile
  - sort by join time, credit

Social network

- Friends
- Account page has social stuff in RH column
- Generalized notification mechanism
  - sources:
    - friend requests
    - new PMs
    - posts to subscribed threads
  - delivery:
    - email
    - batched email
    - web site
    - RSS feed
    - soon: BOINC Manager

Single-job submission

```
boinc_submit --infile X ... program
```

- Avoids:
  - template files
  - validator, assimilator
  - directory hierarchy
  - update_versions
- But you lose:
  - platform independence
  - validation
  - code signing

Support for multithread and coprocessor apps

- Multithread apps: use N threads, M<N cores
- Coprocessor apps
  - Cell: 6-8 SPEs
  - CUDA
- Or any combination
- Old:
  - Server: 1 app version per platform
  - Client: 1 core per job
- New:
  - Server: many app versions per platform
  - Client: a job can use X cores and N instances of each coprocessor type

Multithread/coprocessor (cont.)

- How to decide which app version to use?
  - app versions have “plan class” string
  - scheduler has project-supplied function

```
bool app_plan(SCHEDULER_REQUEST &sreq, char* plan_class, HOST_USAGE&);
```

  - returns:
    - whether host can run app
    - coprocessor usage
    - CPU usage (possibly fractional)
    - expected FLOPS
    - cmdline to pass to app
  - embodies knowledge about sublinear speedup, etc.
- Scheduler: call app_plan() for each version, use the one with highest expected FLOPS

Multithread/coprocessor (cont.)

- Client
  - coprocessor handling (currently just CUDA)
    - hardware check/report
    - scheduling (coprocessors not timesliced)
  - CPU scheduling
    - run enough apps to use at least N cores

Other stuff

- Documentation
  - moved user docs to MediaWiki
- BOINC project publication list
  - please maintain!
- Release management
  - Server stable branch
  - Client: distro-specific Linux packages
  - Server packages (SZTAKI)

Upcoming work

- Client
  - break up state file
  - New scheduling philosophies
    - fill unused RAM, CPU, network
    - heat limited computing
- API/runtime
  - replace heartbeat mechanism
  - replace brittle message-passing scheme
- Installer
  - deal with “standby after X minutes”
- Server
  - enforce file immutability
  - detect need to update DB

Projects:

- BOINC is here to serve you
  - if you want a feature, let us know
- Fold your changes into the trunk
- Keep server, API code up-to-date
- Develop apps for
  - GPU (CUDA)
  - Multicore (MCUDA)
  - BOINCoid?
- Use Bossa, Bolt

Projects:

- Maintain your web site
  - Science education and news
  - Have a personal presence
- Retain your volunteers
  - “reminder email”
  - newsletters
  - tell-a-friend link
- Generate publicity
  - use your university’s PR office
- Think about organizational models