Citizen Cyberscience

Using the Internet, involve the global public in current scientific research

- Volunteer computing: use people’s PCs
  - 50 projects
  - 500,000 participants
- Volunteer thinking: use people’s brains (intelligence, knowledge, cognition)
  - 10 projects
  - 50,000 participants
Stardust@home

• The Stardust mission
  − 1999-2006
  − collect cometary, interstellar dust

• Where’s the dust?
  − aerogel collector
  − 100K “focus movies”

• Stardust@home
  − http://stardustathome.ssl.berkeley.edu/
  − 23,000 volunteers
  − 43M viewings
  − 14 verified tracks, 50 candidates found
Lessons learned from Stardust@home

- Volunteers will do lots of monotonous tasks
- You don’t need to play games
- Motivators:
  - community, competition
  - keep volunteers informed
- Possible to achieve quantifiably high accuracy
  - calibration jobs
FossilFinder

- Collect photos of Middle Awash (Ethiopia)
- Look for hominid and other fossils
Fold it!: a protein-folding game
What else?

• Visual pattern recognition
  – Stardust@home, Clickworkers
  – GalaxyZoo
  – Africa satellite photos: UNOSAT
  – text/handwriting recognition
  – image tagging

• Real-world knowledge
  – http://openmind.org

• 3D spatial manipulation
  – Fold it!

• Language
  – Distributed proofreading: http://pgdp.net
Bossa: open-source middleware for distributed thinking

BOINC

volunteer computing

Bossa

volunteer thinking

Bolt

teaching, training

BOINC Basics

accounts, groups, credit, communication
What does “middleware” mean?

- Volunteer identity
- Job queuing
- Job assignment
- Maybe other stuff
  - social features (Bossa)
  - accounting features (Mechanical Turk)
Bossa design philosophy

- Projects have widely varying requirements
- Bossa provides mechanisms
- Applications define policies (using PHP)
Bossa abstractions

![Diagram showing the relationship between Project, Application, Job, User, and Instance.]
Job and result representation

- Job parameters (filenames etc.) are stored in an “opaque” PHP data structure
- Similar for job results (instances)
- Callback function to display a job:
  ```php
  job_show($job, $inst)
  ```

- Types of jobs
  - single web page
  - sequence of pages
  - offline app
Job distribution policy

- Project A: few jobs, lots of volunteers
  - do all jobs once, then twice, ...
- Project B: infinite jobs, finite volunteers
  - do first job $N$ times, then 2$^{nd}$ job, ...
- Bossa: each job has “priority”
  - adjusted by callback functions
User assessment

• How to assess?
  - training course
  - calibration jobs
    • not feasible for some applications
  - correctness as determined by replication

• Representation
  - scalar
  - sensitivity and discrimination
  - many dimensions

• Bossa mechanisms
  - opaque data for users
  - calibration jobs
  - Bolt course prerequisite
Replication policy

• Examples:
  - Fixed replication (N instances)
  - Adaptive replication (achieve target accuracy)

• Bossa mechanism:
  - job_finished() can get list of instances and users, decide whether more instances are needed, and set priority
Use of “experts”

• Alternatives:
  - Experts do the same job, but better
    • use them to resolve difficult jobs, or verify detection of rare features)
  - Experts do a different job
    • e.g., use them to classify or estimate age of fossils)

• Bossa mechanism:
  - Users can be assigned “level” (0, 1, ...)
  - Each job has a priority P(i) for each level i

• Example:
  - use level 1 users to resolve jobs where level 0 users haven’t reached consensus
  - job_finished(): if ambiguous, set P(1) to 2
Examples

- Basic
- Fixed replication
- Calibration tasks and adaptive replication
- Image annotation
Conclusion

- Bossa: a lightweight framework for volunteer thinking projects
- Plug in your own:
  - job representation and display
  - job distribution and replication policy
  - volunteer assessment
- Future directions
  - thinking/computing workflows
  - group jobs
    - teams as organizations
  - jobs as online multiplayer games