

BOINC

BOINC: A System for Public-Resource
Computing and Storage

David P. Anderson

Serge Koren

CMSC714

November 22, 2005



Outline

- ➔ Introduction
- ➔ Contrast to Grid Computing
- ➔ BOINC Goals
- ➔ BOINC Project/Server
- ➔ BOINC Client
- ➔ Conclusion



Introduction

- Many personal PCs (petaFLOPs)
- Many large storage devices on personal PCs (exabytes)
- Can be exploited for appropriate programs
 - Low data to compute ratio
 - Independently parallel
 - Server/worker model



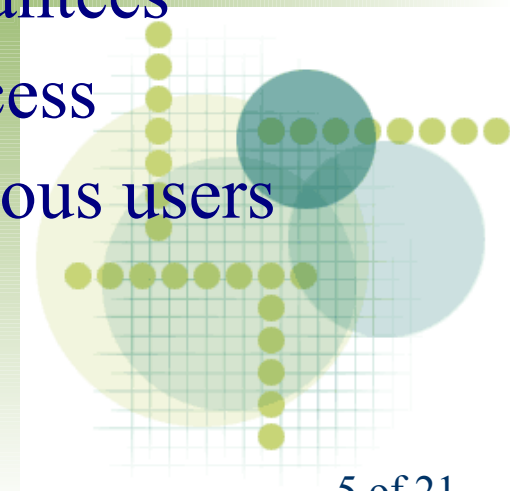
Introduction (cont)

- Computers connected with lower connectivity (DSL, cable, telephone)
- Participants volunteer to participate (rewards)
- Risk of malicious participants
 - Forged results
 - Forged reward



Contrasting Approaches

- Grid computing
 - Designed for centrally managed expensive equipment
 - Targets high performance, high bandwidth, full-time connectivity machines
 - Strict performance/availability guarantees
 - Need for resource discovery and access
 - Less concern over authorized malicious users (hard to become authorized user)



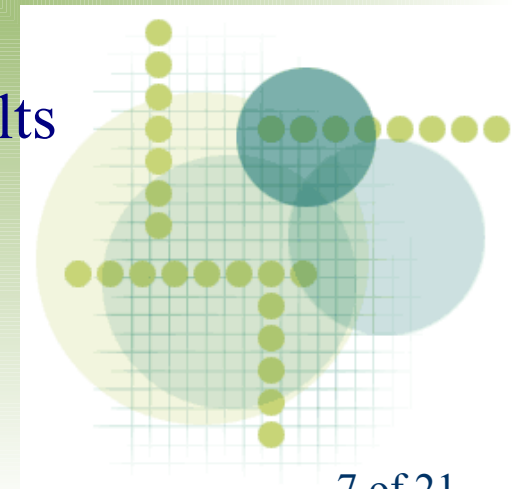
BOINC Goals

- Reduce the barriers of entry to public-resource computing
- Share resources among autonomous projects
 - participants can be part of multiple projects
- Support diverse applications
- Reward participants
 - Cheat-resistant “credit”



BOINC Architecture

- BOINC Project
 - Identified by master URL
 - Includes multiple applications
 - DB containing info on applications/participants/work/results/teams
 - Task Servers
 - issues work to clients and accepts results
 - Data Servers
 - accept file uploads



BOINC Server

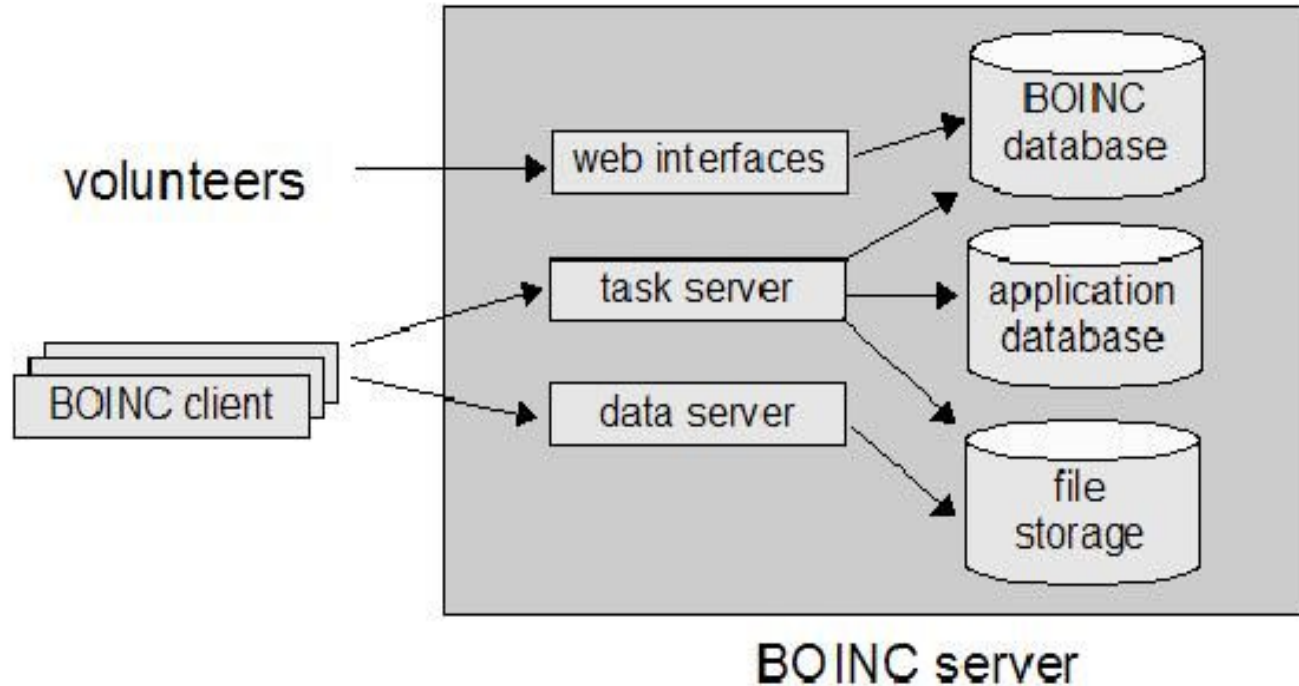


Figure 1: A BOINC server consists of several components, sharing several forms of storage.

How it Works

- Application versions
 - Support different client machines
- Workunits (version independent)
 - Contains input data, command-line args
 - Lists compute/memory/storage requirements and deadline
- Results
 - Result of a specific workunit
 - Consists of output files



Task Server

- Work generator – generates workunits
- Scheduler – determines appropriate workunits for a client
- Feeder – caches database information
- Transitioner – manages state of workunit
- Validator – examines results and selects canonical
- Assimilator – assimilates validated results



Task Server

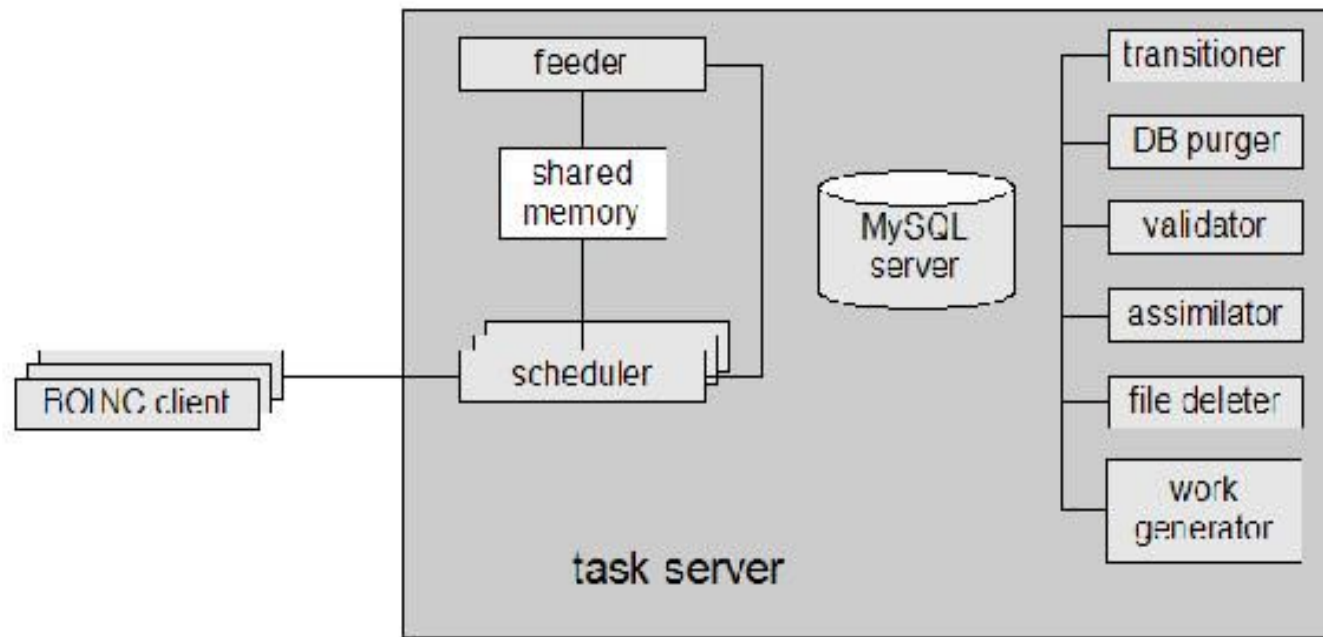
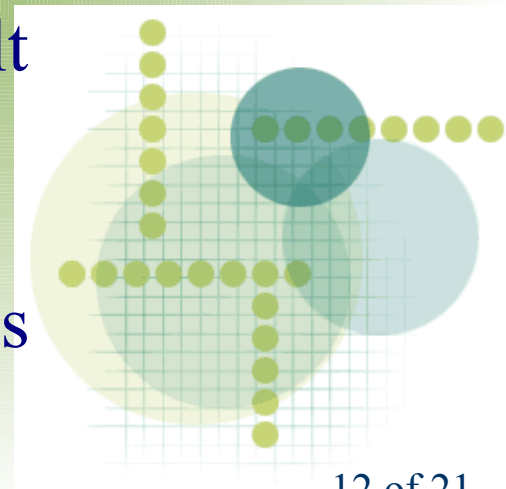


Figure 2: The components of a BOINC task server

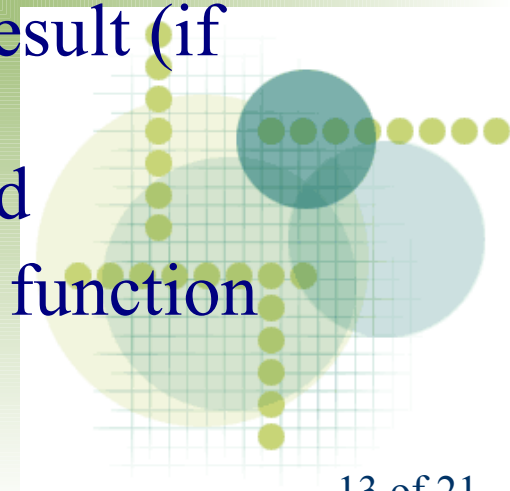
Security Issues (Server)

- Result falsification
 - redundant computing
- Credit falsification
 - credit verification
- DoS attacks on server
 - limited file upload sizes/signed result description
- Theft of project files
 - BOINC does not encrypt project files



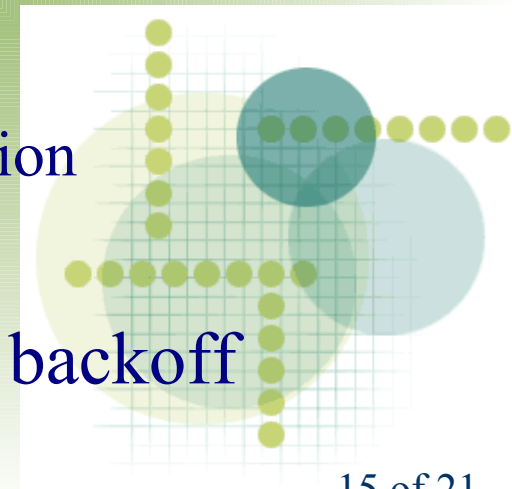
Redundant Computing

- Redundancy
 - N copies of each workunit created
 - Two instances never sent to same participant
 - If an instance times out, another one is sent
- Validation
 - Each result compared to canonical result (if available)
 - Otherwise a quorum must be reached
 - Validated using application specific function



Credit and Failure

- Credit
 - claimed credit – CPU time * BOINC benchmark
 - can include other application-specific credit
 - granted credit
 - actual credit based on average or min of workunit instances
 - trickle messages
 - allow credit for intermediate computation
- Failure
 - all communications use exponential backoff



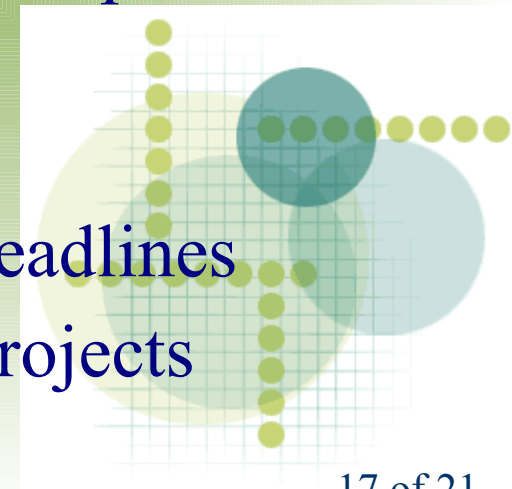
BOINC Client

- Used by participants
- Shared for all projects
- Constructed from several components
 - Core client
 - Client GUI
 - API for applications to get CPU access
 - Screensaver



BOINC Client Features

- Participant preferences
 - when computation is allowed
 - limits of CPU time/bandwidth
- Anonymous platforms
 - participants compile source code and report results as anonymous
- Local scheduler
 - satisfy requirements and workunit deadlines
 - maintain variety between multiple projects

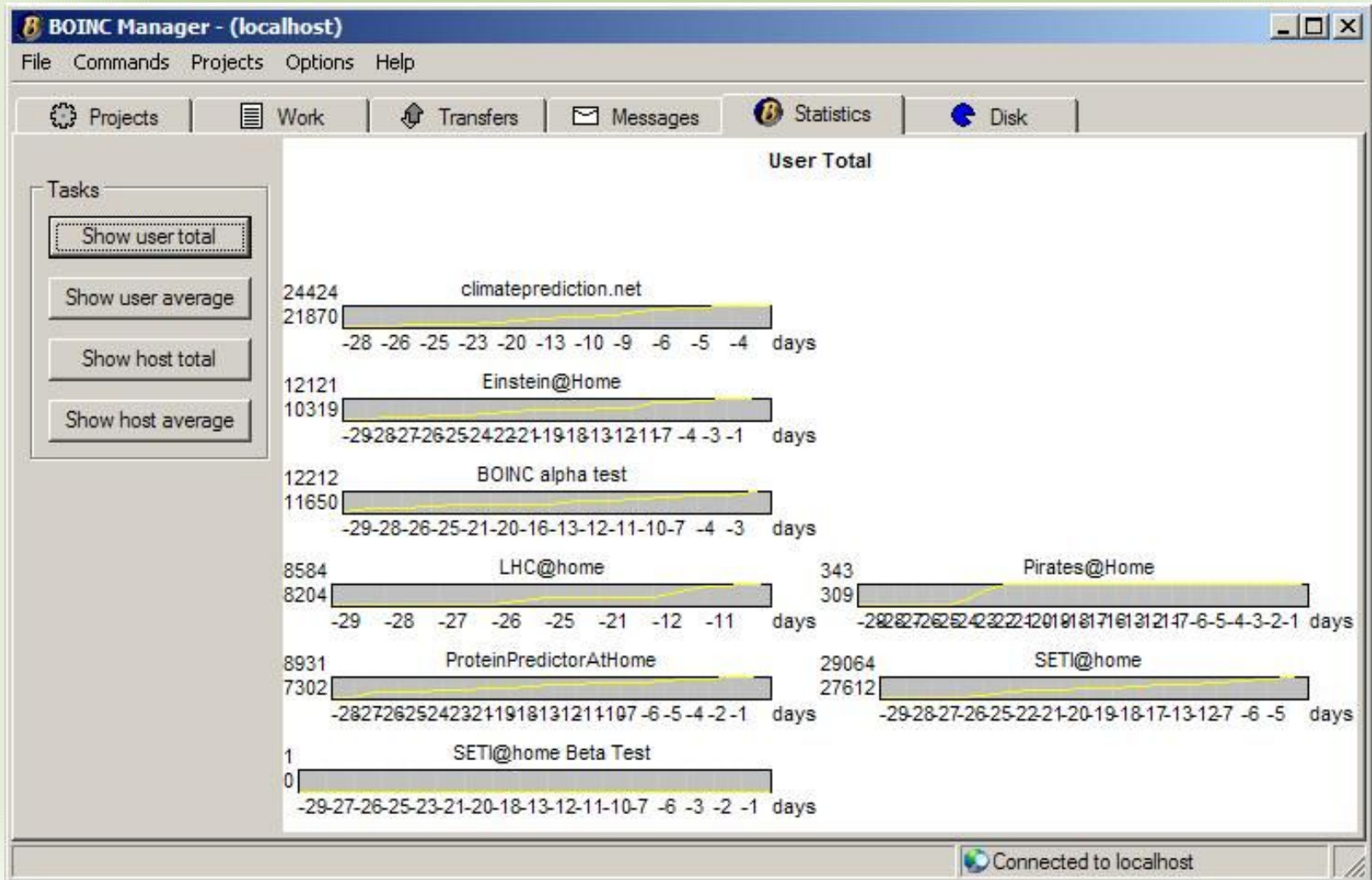


Security Issues (Participant)

- Malicious executable distribution
 - code/executable signed by server
- Theft of account information from server
 - server must be secured
- Theft of account information over network
 - no protection
- Intentional/Accidental abuse of machine
 - no sandboxing, projects should be well tested

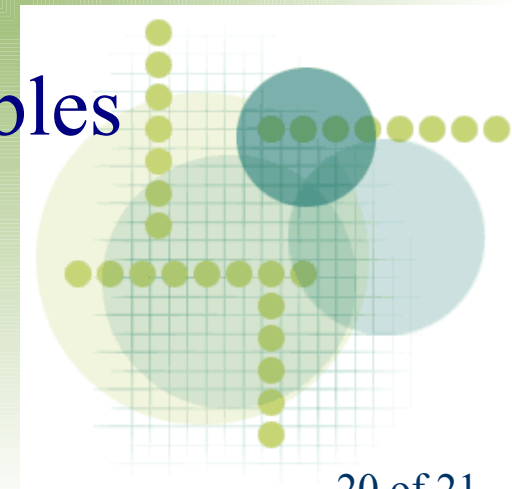


BOINC Client



Conclusion

- Many examples of projects
- DoS on server still possible by malicious client
- Centralized DB for project server is bottleneck (MySQL)
- No sandboxing of project executables



References

- D. P. Anderson. BOINC: A System for Public-Resource Computing and Storage. 5th IEEE/ACM International Workshop on Grid Computing, November, 2004.
- D. P. Anderson, E. Korpela, R. Walton. High-Performance Task Distribution for Volunteer Computing. To appear in First IEEE International Conference on e-Science and Grid Technologies, December 2005.
- BOINC Homepage. <http://boinc.berkeley.edu/>

