SETI@home: An Experiment in Public-Resource Computing

Teng Long

Class Presentation for CMSC818K

April. 14th, 2011

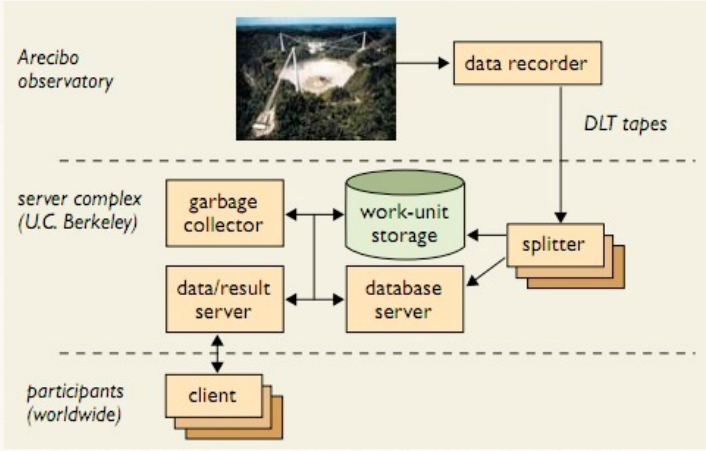
Department of Computer Science

University of Maryland

Background

System Structure

- A Server-Client System



Features

- Highly Parallelizable:
  - It does not require communication with each other;
- Computational-Intensive task:
  - 350 KB download;
  - 1KB upload;
  - 10-hour computation(at that time);
- Error Toleration:
  - An error just affect the goal slightly, not totally.♪

♪

Technical Details

- Redundant computing;
- Task Assignment:
  - Delete work unit if RECEIVE enough results(bottleneck);
  - Delete work unit if SENT enough copies.
- Continuous computing on a frequently turned-off machine;
  - Periodical state saving;

Non-Technical Details

- GUI is important
  - So the user knows what they are doing;
- Statistics is also important
  - So the users are encouraged by their own contribution.

Designing a Runtime System for Volunteer Computing(BOINC)

Teng Long

Class Presentation for CMSC818K

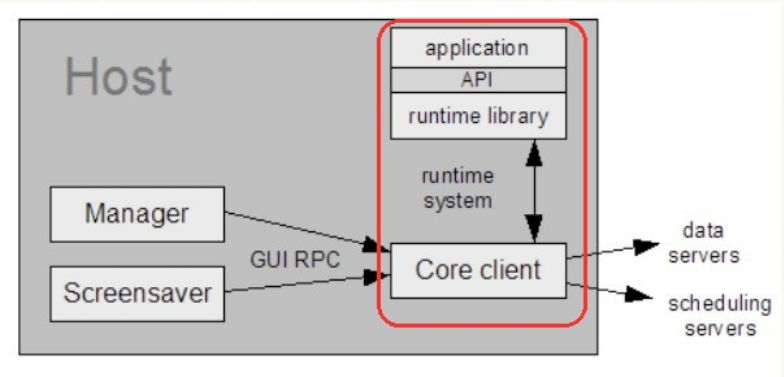
April. 14th, 2011

Department of Computer Science
University of Maryland

Design Goals

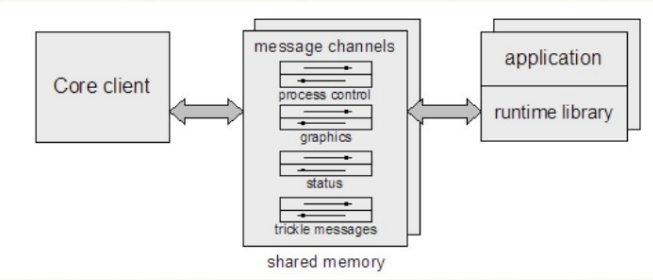
- As a middleware system:
  - Handle widely differing applications and tasks;
- To the users:
  - No significant system slowdown;
  - Local preference specification;
  - Incentives to participants;
  - Simple to install, maintenance and use.

Client Overview



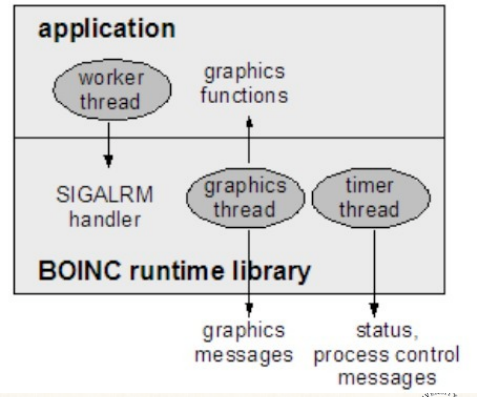
Share-Memory Message Passing

- Eight channels, four for each direction, with specific functions;
- No message queuing;
- All messages are XML.



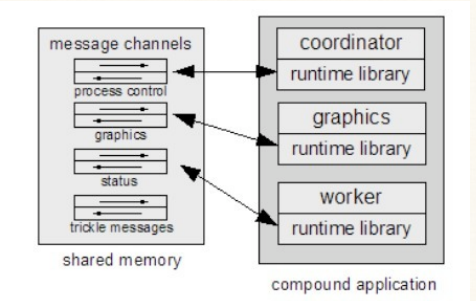
Thread Structure in Unix

- The threads are for main computation and EXTRA activities;
- Three threads:
  - Worker thread:
    - main computation;
    - resource usage monitoring;
    - suspension simulation.
  - Graphics thread:
    - GUI related task;
  - Timer thread



Applications

- Single & Compound Applications
  - BOINC allows multiple worker programs;
  - Coordinator usually needed for compound applications.
- Worker programs use shared-memory space to communicate with each other.



Task management

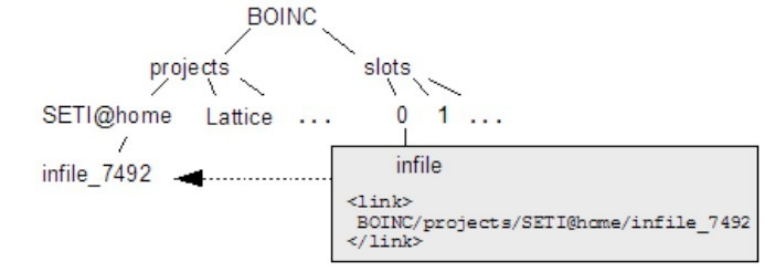
- Task control
  - Suspend, resume, quit, abort;
  - Implemented by sending messages to the process control channel.
- Orphaned and duplicated process
- Reliable termination checking
  - Use *waitpid()* in Unix;
  - Use *finished file* if there is no system support.

Status and Credit Reporting

- CPU time and memory usage
  - The worker thread gets the information and pass it to the core client via *"status"* message channel
- Task completion estimation(calculated every sec.)
  - For scheduling purpose;
  - Again, to give users the information;
  - Implemented by calling *boinc_fraction_done()* every sec.
- Credit tracing and reporting
  - Estimate CPU time cost by default;
  - Feedback to the users.

Directory Structure and File Access

- BOINC must run tasks in separate directories;
- *Slots* are created for active tasks, and *link files* are used.
- If two concurrent tasks use a (read-only) file, there should be only one copy of it.



Checkpointing

- BOINC EXPECTS applications to do checkpoint/restart;
  - BOINC applications typically have *checkpointable* status.
  - The runtime system supports frequent checkpointing;
- The core client is informed of check points.

Graphics

- Graphical need is (again) for users;
- BOINC provides an API to support graphics rendering
  - *app_graphics_render();*
- Hardware acceleration is hopped, not required.
- BOINC limits the fraction of CPU time used by graphics thread.

Failure Information Collection

- Some failures occur only in specific context;
- To collect information:
  - *stderr* is stored as a file and sent to the project server;
  - In case of crashes, stack trace is written to *stderr*;
  - In case of task aborted, stack trace is written to *stderr*.
- The server maintains the database of failure information♪

Long-running applications

- Connection with server during tasks is needed.
- Trickle messages:
  - Trickle-up: send to server its local statistics;
  - Trickle-down: get termination message.
- BOINC allows file uploading.

Non-CPU-intensive applications

- Examples:
  - Study of network structure and performance;
  - Study of the dynamics of computer usage;
  - Applications that provide a network service.
- Solution:
  - The core client still always runs;
  - The BOINC API supplies functions that suspend and resume **all** BOINC activities.

Questions?

Teng Long

Class Presentation for CMSC818K

April. 14th, 2011

Department of Computer Science
University of Maryland