Cycle Sharing Systems

Jagadeesh Dyaberi
Dependable Computing Systems Lab
Purdue University
Outline

- Introduction
- Design of Program
- Security
- Communication
- Architecture
- Implementation
- Conclusion
Introduction

- What is cycle-sharing?

Sharing CPU resources across a network so that all machines function as one large supercomputer.

*Cycle sharing model across the Internet*
Introduction

Why Internet?

- 93 million connected hosts
- Rapid growth in computing power
- Revolutionary expansion of mobile devices like cellular phones and PDA’s.
Introduction

Examples of Cycle Sharing Systems

- SETI @ Home
- Mersenne Prime Search
- Distributed.net
- XtremWeb
Introduction

- **Global Computing Issues**
  - **Scalability**
    - Scale to hundreds of thousands of nodes
  - **Heterogeneity**
    - Across hardware, OS and basic software
  - **Availability**
    - Owner sets limits on resource
  - **Fault Tolerance**
    - Must accept frequent faults while maintaining performance
  - **Security**
    - Protection from malicious or erroneous manipulations
  - **Dynamicity**
    - Varying configuration, communication latency
Two Ways of using XtremWeb

1. Worker
   - Volunteer Machine
   - Registers with the administration server
   - Contributes when idle

2. Collaborator
   - Setup their own global distributed application
   - Works for the main XtremWeb
     - Idle
     - Has no work to send to his community of PC’s.
XtremWeb

Diagram showing the network setup for XtremWeb.

Legend:
- Worker
- ISPs
- Personal Devices
- Collaborators LAN
- Private LAN
- Results Collector
- Application dedicated

Steps:
1. Workers LAN
2. Personal Devices
3. Private LAN
4. Results Collector
5. Application dedicated
6. Collaborators LAN
7. ISP Server
8. Personal Devices
XtremWeb

Security and Native Code Execution

- Native code execution -> High Performance
- Ad-hoc verification process
  i. Only trusted institutions can propose code
  ii. Code is tested on dedicated workers
  iii. Code is encrypted before downloading to workers
  iv. Code download uses a private-public key to secure transaction
  v. Worker sends back a checksum of code to server which verifies it
**XtremWeb**

- **Communication between Worker and Server**
  - Three protocol layers
    1. First level (connection) – Enable connection between entities behind firewall or a proxy
    2. Second level (transport) – Responsible for reliable and secure transport
    3. Third level (protocol) – RPC API
**Communication between Worker and Server**

- **Worker**
  - worker Identity
    - user login
    - worker Desc.
  - worker Desc.
    - runtime Desc.
    - binaries cache
  - id Task
  - result Desc.
    - id_task
    - task status
      - (completed, aborted)
    - Output file (results)

- **Server**
  - host register
  - comm. Vector
    - Protocol Desc.
  - Work Request
  - task Desc.
    - id_task
    - input file, binaries
    - @ results collector
  - Work Alive
  - task Status
    - continue, abort
  - result Request
  - result Store
  - result Desc.
    - id_task
    - task status
      - (completed, aborted)
    - Output file (results)
Worker Architecture

**Implementation**

- **Java**
  - Easily portable
  - Not related to performance but to security and portability
- Calls to specific OS functionalities - C
Server Architecture

- Pool of Applications
- Pool of Jobs
- Accounting Modules
Server Architecture

- **Scheduling**
  - **Dispatcher**
    1. Selects tasks from the task pool and forwards to the scheduler.
    2. Priority is set by minimum ratios of tasks running per application.
  - **Scheduler**
    1. Tasks are scheduled in a FIFO scheme.
    2. Determines task that may be run by a worker.
Implementation

- Java, PHP and Perl
- MySQL database
- Volunteers and Administrators interact though a Web interface
Fault Tolerance

- Uses Fault-Tolerant MPI implementation.
  - Programmer may save results periodically in case of an entire restart
  - MPI hides faults from programmer by using a fully automatic fault detection and recovery.
- Pessimistic logging principle to tolerate $N$ concurrent faults. ($N = \text{total number of MPI processes}$)
Fault Tolerance

MPICH-V1

- Channel Memories
- Dispatcher
- Computing Nodes
- Checkpoint servers

MPICH-V2

- Checkpoint Scheduler
- Event Loggers
- Dispatcher
- Computing Nodes + Communication Daemons
- Checkpoint servers
Implementation

- Projects implemented
  - AIRES
  - Protein Folding by Mutations
Conclusion

- Global Computing systems are concerned with ease of deployment.
- Fault tolerance is critical since there are external threats.
- Threats come from application, data and the computing nodes.
References
