Improving Service Performance Through Object Replication in Middleware: A peer-to-peer Approach

Tianying Chang
Mustaque Ahamad
Georgia Institute of Technology

Presented by Wai Gen Yee,
Illinois Institute of Technology
Large Scale Distributed Services

- Popular
  - Real-time traffic information dissemination
  - Provided by central servers
  - Large number of distributed users

- Distributed object platform
  - Underlying middleware layer hide low level details and provide more support
  - Familiar programming interface, RPC
Problems

- Scalability and performance
  - Large number of users
    - Limited server processing power
    - Limited bandwidth
  - Large network delay
Caching

- Clients cache interested objects locally
  - Invoke locally to reduce network delay
  - Only for self
  - Good for high ratio of read/write
  - High consistency maintenance overhead
Replication

- Dedicated servers are distributed
  - Geographically distributed: closer to quests
  - Larger total processing power
  - Statically deployed servers are not flexible
    - Number of servers
    - Locations of servers
P2P System

- Large amount of resources available
  - Scalable: increases as the system size increases
- Peer nodes volunteer to contribute resources
  - Dynamically replicate service objects at volunteering peer nodes
  - Select the nearest service objects
Challenges

- How to decide which peer nodes to replicate a service?
A Peer-to-Peer Approach for Replication

- Peer nodes volunteer to contribute their resources to replicate the service for others
- Flexible and adaptive
  - Number of replicas
  - Locations of replicas
Consistency Maintenance

Server

Peer Replicas at Time I

RPC Calls
Goals

- Fairness
  - Important for P2P system
- Controllable number of replica
  - Consistency maintenance overhead
- Autonomous Decision
  - Central decision limits scalability
- Performance
  - Good placement of replicas
  - Small client-to-replica distance
Stable Replica Fraction

- Scalable
  - High enough to provide enough processing power
  - Small fraction of replicas: for consistency maintenance overhead
  - Tradeoff between processing power and consistency maintenance cost
- Maintain a replica fraction of $f$ in the system
Fairness

- Different definitions
  - The amount of resources contributed by each node is proportional to its demands
  - Each node contributes similar amount of time
- Each node tries to contribute similar amount of time
  - Each node contributes a fraction of $f$ of its participation time
Performance

- Small average client-to-replica distance

\[
\sum_{i=0}^{\mid U \mid} d(u_i, L(u_i)) \cdot r_i
\]

- Heuristic
  - Keep replicas far apart, prevent clustering of replicating nodes
  - The more requests generated, the more replicas should be provided
Autonomous Adaptability

- Centralized adaptability
  - Periodically collect information in global scope, compute and deploy the new placement
  - Limiting scalability, not desirable

- Autonomous adaptability
  - Each node makes decision based on self maintained information
  - Improved scalability if using local information
Three States of Each Peer

- **Ready**
  - Waiting for SOSes from its neighbors; ready to work.

- **Replicating**
  - Have a replica and can service requests; working.

- **Sleeping**
  - Not doing anything; resting.
Heartbeat

- Maintain connectivity and learn nearby replicas’ information
  - Each replicating node heartbeat its replicating status and its serving load to nearby nodes with fixed TTL
  - Therefore, each peer node only receive heartbeat message within the fixed TTL range
Heartbeat (cont’d)

- Distributed counting protocol to estimate the global replica fraction
  - Initialized counter as 1 if the node is replicating; otherwise 0
  - Shed and exchange the counter value among neighbor nodes through heartbeat
  - Eventually, each peer node has the correct estimate of the current replica fraction in the system.
Creation of Replica

- Each peer node makes replication decision periodically:
  - Calculate the creation probability $p_i$
  - $p_i$ is affected by the overload condition and resource contribution history
- If replication decision is yes:
  - Replicate the object and set the replicating time $\beta$
Replica Deletion

- Delete the replica when:
  1. When scheduled time slice $i$ runs out;
  2. When receives much fewer requests than other replicas in its vicinity
Performance Evaluation

Simulation setup

- 1000 nodes
- Nodes randomly located in 5 dimension space
  - End-to-end distance is calculated with the Euclidian distance
- Topology
- Target replica fraction $f=5\%$
- Each node generates requests following exponential distribution with $\lambda$
Replica Fraction
Fairness
Performance

- Greedy heuristic algorithm
  - Select best $k$ replicas, $k=fN$
- Random algorithm
  - Randomly select $k$ replicas
- AURA
  - Dynamically place a fraction of $f$ replicas
Thank you!

Contact:
Tianying Chang
(tychang@cc.gatech.edu)
Mustaque Ahamad
(mustaq@cc.gatech.edu)
Thank you guests, organizing committee and Konstanz U.

(Let’s drink!)