Sharing Memory with semi-Byzantine Clients and Faulty Storage Servers

Hagit Attiya
Technion CS

Amir Bar Or
Technion CS & HP Labs Haifa

Storage Systems

- storage servers
  - disks
  - active disks
  - object stores ...
- clients
- complete network
  - asynchronous
  - reliable

Our Contributions

- Fault-tolerant simulations of a single-writer multi-reader regular register in storage systems, tolerating
  - Semi-Byzantine failures of clients, resulting in erroneous write operations
  - Fail-stop failures of storage servers
    - requires a majority of nonfaulty servers
  - Byzantine failures of storage servers
    - requires that two-thirds of the servers are nonfaulty
- Use the simulations to improve the fault-tolerance of the Paxos algorithm [Lamport]

Talk Outline

- Model
- Simulations
- Application:
  - The Paxos Algorithm
- Conclusion
Client Failures

- Well debugged applications do not suffer malicious failures

Still... corrupted data may be written to shared storage:
- A client fails during an update operation, leaving a corrupted disk block
- Failures of network switches and storage hardware cause loss / re-ordering of messages from the client to the storage servers

Semi-Byzantine Clients

- Memory objects are protected by access control lists
  - Only specified client may access the object
- Clients fail by stopping or executing some memory access operations incorrectly

Modeling Semi-Byzantine Clients

- Use the faulty shared memory model [Afek, Greenberg, Merritt and Taubenfeld]
- A faulty register experiences erroneous writes
  - A sequence of write / read operations (to the same register) is f-faulty if inserting f arbitrary write operations makes it legal
  - An algorithm is f-reliable if it is correct in any f-faulty execution

Talk Outline

- Introduction
- Simulations
- Application: The Paxos Algorithm
- Conclusion
Handling Semi-Byzantine Clients

Algorithm 1
An f-reliable simulation of a SWMR register with semi-Byzantine clients

• An f-reliable SWMR atomic register [Afek et al.]
• Using 20f+8 atomic SWMR registers

Handling Fail-Stop Storage Servers

Algorithm 2
A simulation of a SWMR register with semi-Byzantine clients and fail-stop servers
  – assume a majority of storage servers do not fail

• Shared-memory simulation [Attiya, Bar-Noy and Dolev]
• Message complexity 2n*(20f+8)
  – Message aggregation reduces complexity to 2n.
• Time complexity 20f+8

Simulation with Byzantine Storage Servers

Simulation Properties

WRITE(x)
send STORE(x) to all servers
loop
receive ack from server S
until ack is received from n-t servers
READ()
loop
send READ to all servers
loop
answer[S] = receive VAL(x) from server S
until received from n-t servers
until (one value appears n-t times in answer[])

• A read operation returns the value of the latest preceding or overlapping write operation
  – Safety is always guaranteed
• A read operation terminates if it overlaps a finite number of write operations
  – May block if overlaps an infinite number of writes
• Requires n > 3t
  – t is the number of faulty storage servers.
Handling Byzantine Storage Servers

Algorithm 3
A simulation of SWMR regular register with n>3t Byzantine storage servers

Algorithm 4
A simulation of SWMR regular register with semi-Byzantine clients and n>3t Byzantine storage servers

Talk Outline

- Introduction
- Simulations
- Application: The Paxos Algorithm
- Conclusion

Shared-Memory Paxos

- A weak consensus algorithm using SWMR regular registers.  
  [Gafni and Lamport]

- Phases of the Synod consensus algorithm
  - Agreement is always guaranteed

- Terminates when a single process executes a phase
  - Use unreliable leader election algorithm
  - When the system is stable, only one process executes the algorithm

Fail-Stop Storage Servers and Semi-Byzantine Clients

- Use Algorithm 2
- Tolerates f erroneous writes executed by the client in a single phase
- Masks invalid values written by clients

- Message and time complexity is 20f+8 times the complexity of Paxos.
- Self-checking allows to use only 28 registers (f=1)
  - client stops writing after the first faulty operation
Byzantine Storage Servers and Semi-Byzantine Clients

- Use Algorithm 4
- Paxos attempts to terminate only when a single process executes the algorithm
  - There are no overlapping writes
  - Reads terminate

Talk Outline

- Introduction
- Simulations
- Application: The Paxos Algorithm
- Conclusion

Other Simulations

[Martin, Alvisi and Dahlin]

- Clients fail only by stopping
  - Can be combined with Algorithm 1 to handle semi-Byzantine clients
- Assume $n > 3t$
- May block when there are overlapping writes
- Storage servers push updates to clients
  - Maintain data about each client

Other Simulations

[Chockler, Keidar and Malkhi]

- Clients fail only by stopping
  - Can be combined with Algorithm 1 to handle semi-Byzantine clients
- Assume $n > 3t$
- Give a terminating simulation
  - Write take two rounds
  - Reads take $t+1$ rounds
- Show this is optimal
- Fairly complex
  - But termination is not always necessary!
Byzantine Paxos

[Castro and Liskov]
- Tolerates Byzantine clients
- Does not tolerate faulty storage servers
- Specially-tailored and not modular

What Else?
- Find other applications
- Improve on the 28f+1 bound
- More practical simulation?

THANK YOU!