How To Bring Together Fault Tolerance and Data Consistency To Enable Grid Data Sharing

Background

- Tools for data sharing:
  - Examples:
    - GridFTP
    - Internet Backplane Protocol (IBP)
    - Stork
  - All require explicit management by the user
- Explicit management of data locations limits efficient use of Grids
- Solution: A data sharing service for grid computing that provides transparent access to data

Data Sharing Service

- User accesses data via a global identifier
- The service handles data localization and transfer
- The service permits the programmer to give some input on how it should work
- It transparently uses replication strategies and consistency protocols to ensure data availability and consistency
- The service supports events such as storage resources joining and leaving or unexpectedly failing

Approach

- A hybrid system inspired by
  - Distributed Shared Memory (DSM) systems for:
    - transparent access to data
    - consistency management
  - P2P systems for:
    - scalability
    - volatility tolerance

Approach

- Uses replication for both consistency and fault tolerance
- Integrated design – Use same set of data replicas for both consistency and fault tolerance
  - The software layer is unnecessarily complex
- Decoupled design – addressing consistency and fault tolerance separately
  - Leverage existing consistency protocols
  - Cleaner design
- This approach uses a decoupled design

A DSM Approach

- DSM: assume stable entities
- Grid: no guarantees, replicate to avoid failure

- DSM: home node that never fails
- Grid: entity that acts as a home node and reacts to failure

- DSM: little need for fault tolerance since entities are stable
- Grid: Use replication, group membership, atomic multicast and consensus to create a layer of fault tolerance

- DSM: consistency protocol built on top of the home node
- Grid: create 'glue-layers' through which the consistency protocol interacts with the fault-tolerant blocks

Model

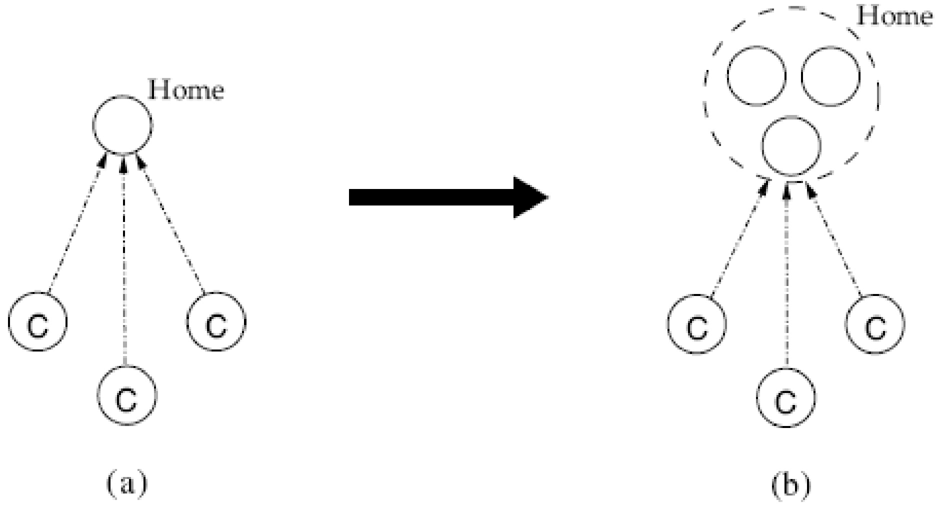

Figure 3. Building a fault-tolerant consistency protocol.

Failure Detection

- Failure model
  - Assume nodes may crash
  - Assume fair-lossy communication where some but not all messages may be duplicated or lost
  - Assume communication delays
  - Assume computation times have upper bounds but they are unknown.
- The algorithm assumes an asynchronous timing model, using a failure detection mechanism
- Such a service is in charge of providing a list of nodes suspected to have failed

Group Membership

- Ability to manage a set of nodes belonging to a group
- Nodes may join or leave the group
- Group nodes are required to store the current composition of the group (member list)
- Protocol ensures a degree of consistency by synchronizing members' view of the group
- Between two view synchronizations, the same set of messages should be delivered to all nodes in the group

Atomic Multicast

- Ensures that messages are received in the same order by all members of the group
- Members of the group agree on an order for message delivery and this agreement is reached using standard consensus protocols.
- Therefore, with our replication scheme all replicas are updated simultaneously

Consensus Protocol

- Allows group nodes to agree on a common value
- Each node proposes some value and the protocol ensures that:
  - eventually all nodes that do not fail decide on a value
  - that the value has been proposed by some node and
  - the decided value is the same for all unfaulty nodes
- In our case the decision is about the order in which messages are delivered to group members

Model

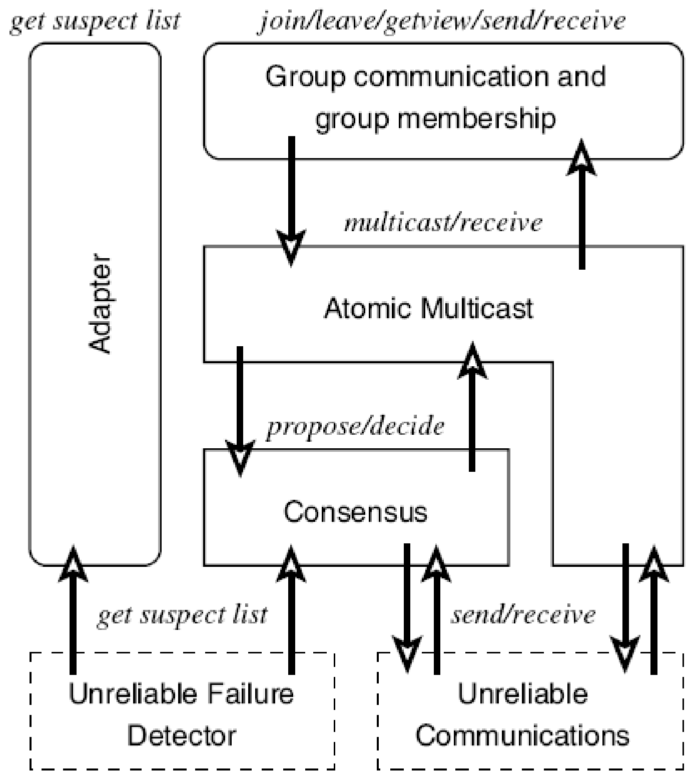
get suspect list
join/leave/getview/send/receive
Group communication and group membership
Adapter
multicast/receive
Atomic Multicast
propose/decide
Consensus
get suspect list
send/receive
Unreliable Failure Detector
Unreliable Communications

Figure 1. An architecture for group communication and group membership protocols.

Group Self-Organization

- Handles the composition of the group of nodes that together act as the home node
- Enforces a group membership policy
- Parameterizes the quality of service of failure detector
- Removes nodes that appear to have failed
- Removes nodes that provide notification of future disconnection
- Adds new members to the group, to maintain home node availability

Dynamic Consistency Protocol Configuration

- Defines how to instantiate a consistency protocol on nodes that have been added to the home group
- Initialize the state of the newcomer so that it is consistent with the state of other members of the group
- The new node must take into account the configuration messages generated by other members of the group at the level of this layer, before reacting to external messages addressed to the group

Model

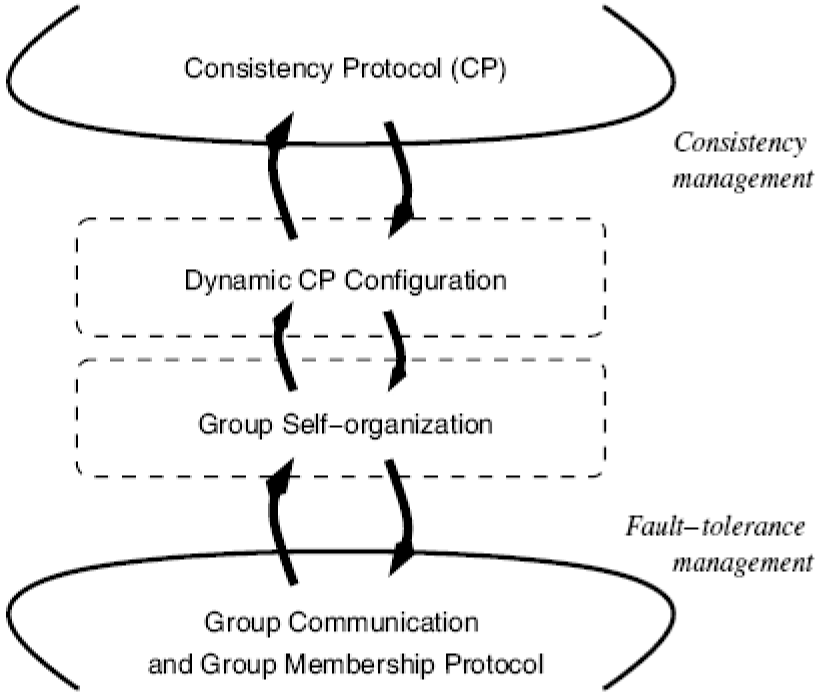
Consistency Protocol (CP)
Consistency management
Dynamic CP Configuration
Group Self–organization
Fault–tolerance management
Group Communication and Group Membership Protocol

Figure 2. Decoupled architecture for managing consistency and fault tolerance.

Consistency Protocol

- The consistency protocol implements exactly the same distributed algorithm as its initial, non-fault tolerant version

- It assumes that the home is always available, but this property is now achieved transparently

- The consistency protocol and replication based fault tolerance mechanism are thus clearly decoupled

Hierarchical Implementation

- Improve efficiency by minimizing inter-cluster communication

- Implement a two-level hierarchy of home nodes
  - Local Data Group (LDG): Cluster-level home nodes to serve accesses from the local cluster
  - Global Data Group (GDG): A group of nodes whose members are the LDGs

- The GDG and LDGs have the self-organizing properties

- They maintain some user-specified replication degree by dynamically adding new members when necessary

Model

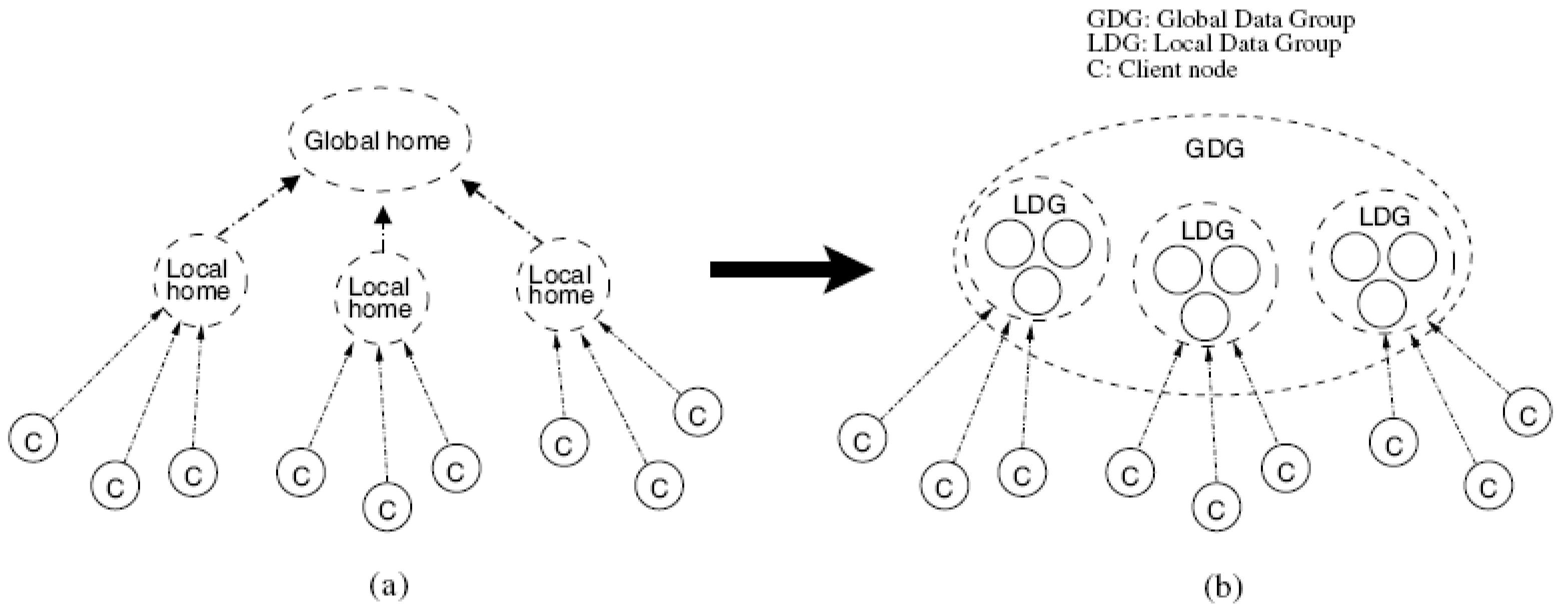

Figure 4. Building a hierarchical, fault-tolerant consistency protocol.

Limitations

- If a client fails before exiting the critical section, other clients cannot ignore this failure because they risk using inconsistent data
- False failure detection due to timeout or temporary overload
- Cluster level failure: if a physical cluster is not available the local data group fails

How To Bring Together Fault Tolerance and Data Consistency To Enable Grid Data Sharing