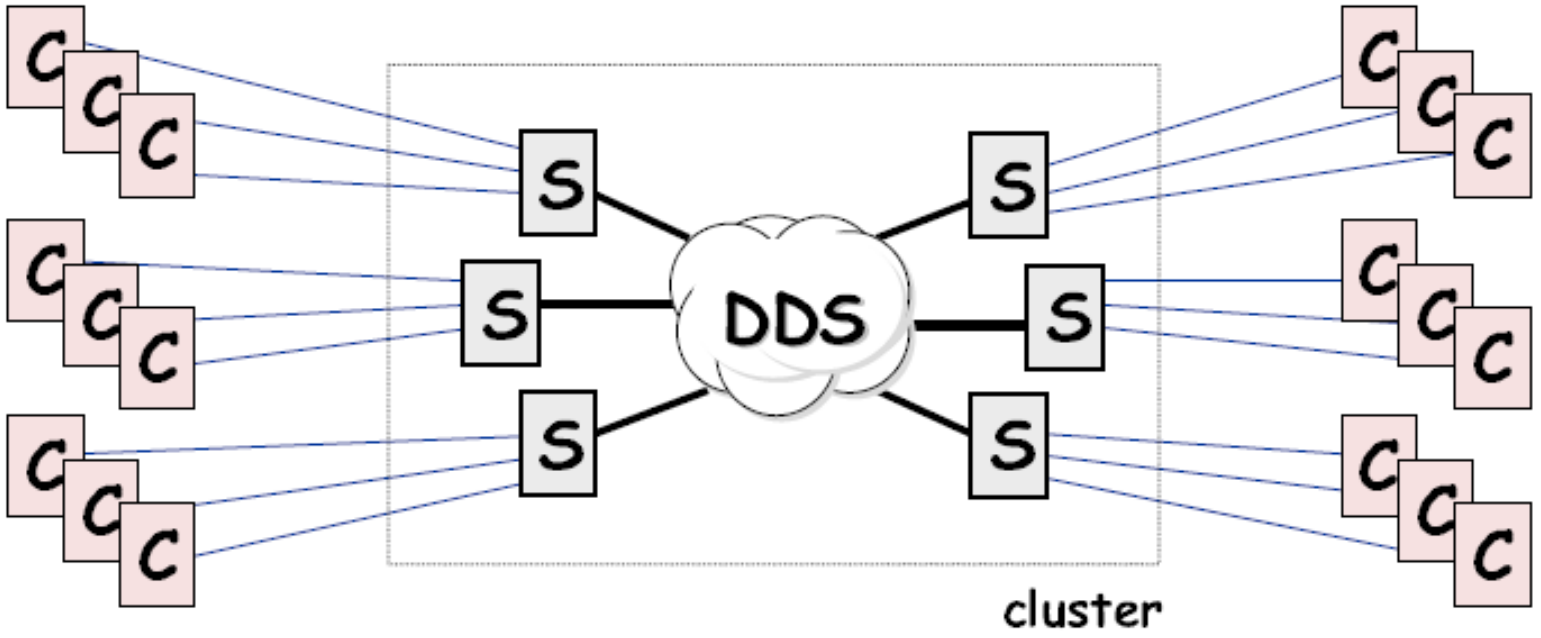
Infrastructure Technologies for Large-Scale Service-Oriented Systems

Kostas Magoutis
magoutis@cse.uoi.gr
http://www.cse.uoi.gr/~magoutis

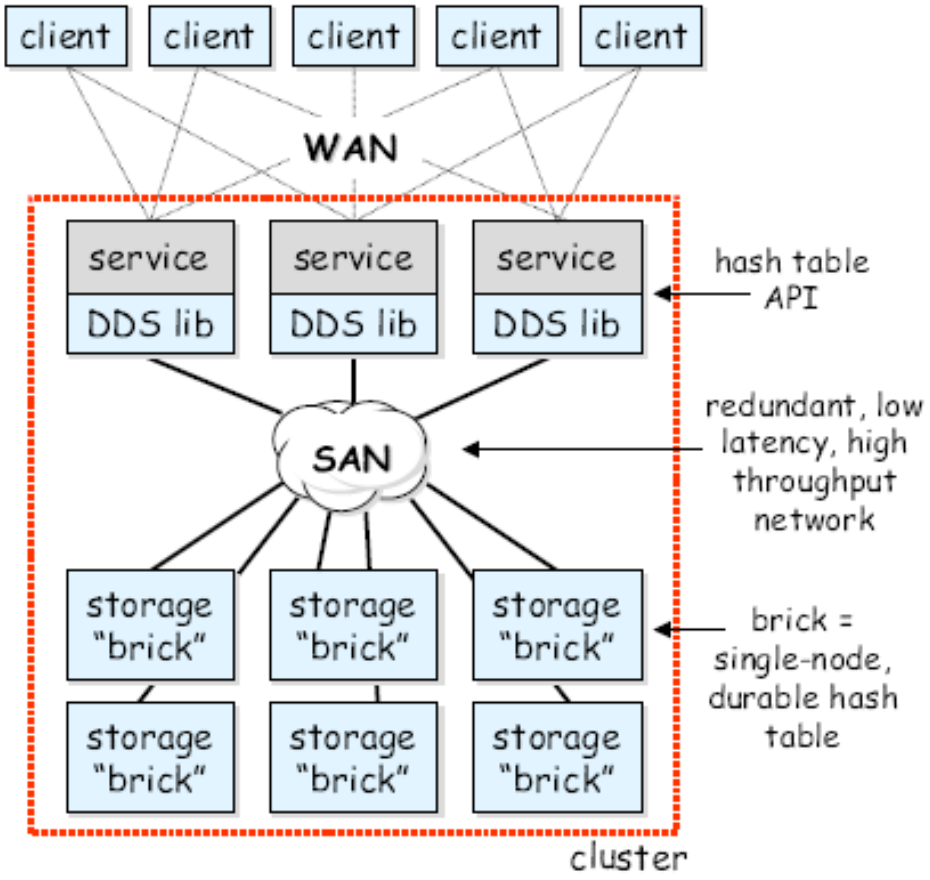
Motivation

- Provide support for Internet services
- Design for clusters of workstations connected by high-performance network
- Improve over existing systems
  - Parallel databases do not scale (heavyweight mechanisms)
  - Scalable file systems offer a too low-level abstraction

High-level view of a DDS



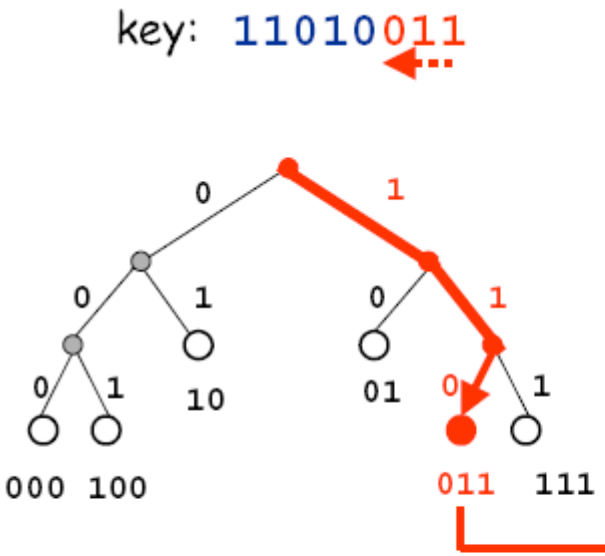
Distributed hash-table architecture



Assumptions

- Unreachable nodes must have crashed
  - Network partitions do not happen
  - Network redundancy makes this a realistic assumption
- Software components are fail-stop
  - On deviation from correct behavior, terminate
- Software failures are independent
- Some degree of synchrony
  - Task execution takes bounded amount of time
  - Messages delivered in bounded amount of time

Metadata: DP and RG maps



RGname	RG membership list
000	dds1.cs, dds2.cs
100	dds3.cs, dds4.cs
10	dds5.cs
01	dds3.cs, dds4.cs
011	dds5.cs, dds6.cs
111	dds7.cs

Step 1: lookup key in DP map to find RGname

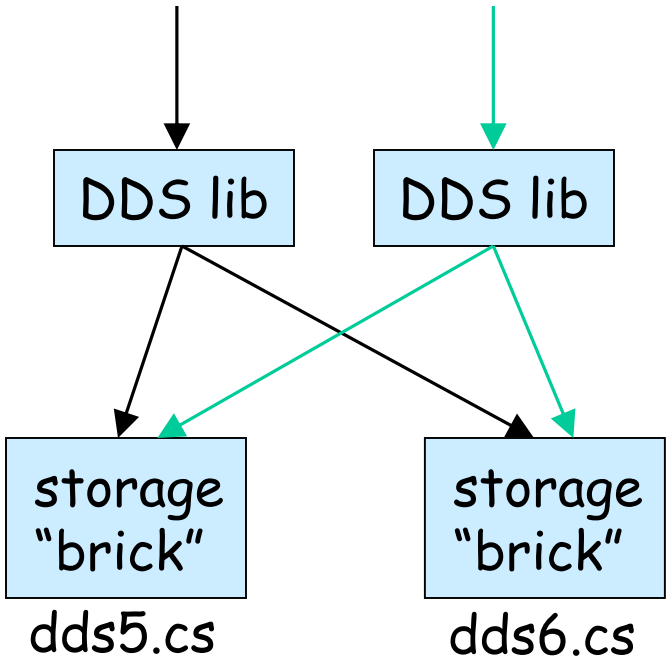
Step 2: lookup RGname in RG map to find list of replicas

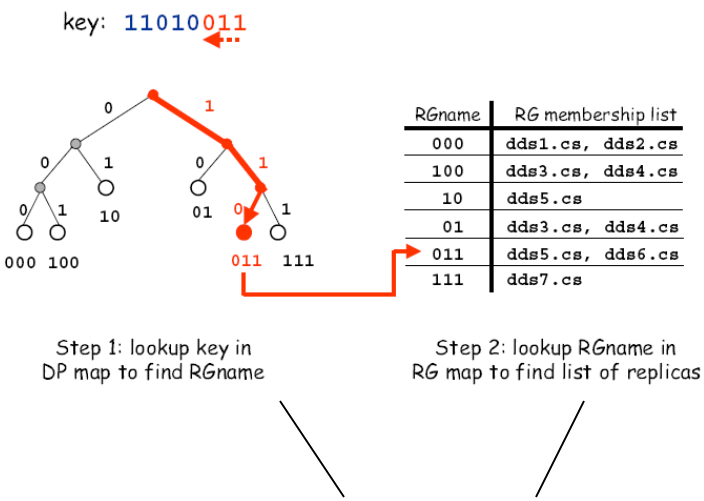
Metadata consistency

- DP and RG replicated across all DDS libs and bricks
- Lazy updates
  - Send metadata updates but do not wait until all in sync
  - They do not use Paxos
- Each data update checks on metadata consistency
  - Using 32-bit fingerprints
  - All participating nodes must agree on metadata maps
  - Any inconsistency fails the I/O, triggers a “repair” action

Failure- and reorganization-free path

put (ht, 11010011, byte[])





Replicated across all DDS libs and bricks

put operations must:

1. Happen on all replicas or none -> use atomic commit
2. Be in the same order -> use locks

Two-phase atomic commit

- Coordinator sends prepare msg to Participants
  - Participants respond commit/abort
  - A Participant may decide to abort if they cannot obtain lock
- If Coordinator fails
  - Participants time out
  - They contact each other to figure outcome (commit/abort)
- If Participant fails
  - Coordinator times out, excludes participant
- Replicas consistent in cache, not necessarily on disk
  - Data can be lost if entire RG goes down

Metadata updates

- Node goes down (failure)
  - RGs must be updated
- Node rejoins (recovery)
  - RGs must be updated
  - Must get fresh copies of replicas
- A partition is split (reorganization)
  - DP and RG must be updated
  - Nodes must get copies of new replicas
- Two partitions are merged (reorganization)
  - DP and RG must be updated
  - Nodes must get copies of new replicas

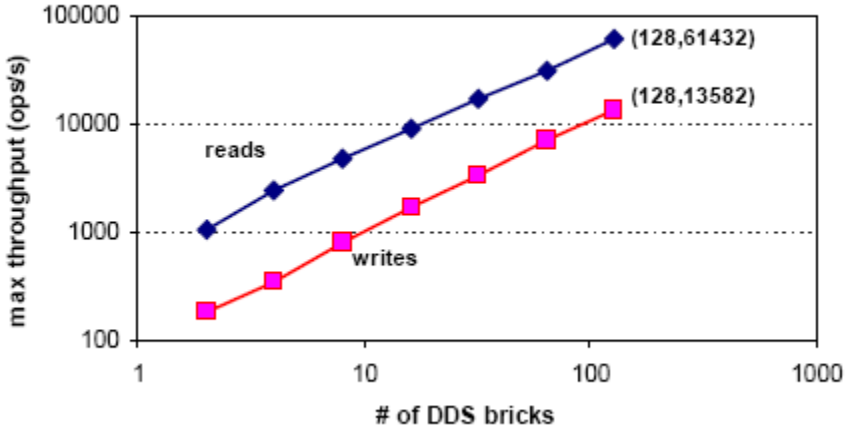
Recovery (node rejoins)

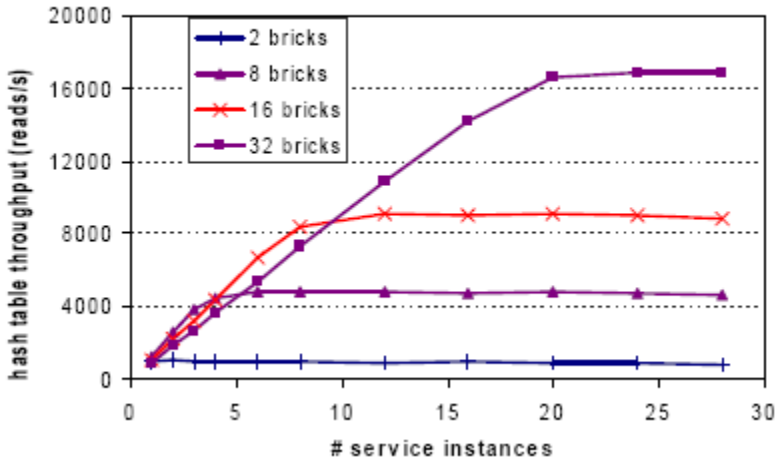
- Obtain lock on entire partition
  - Implemented as write lease on partition
- Copy entire partition (no log replays)
  - Partition is about 100MB, takes 1 sec in 1Gbps network
  - Tradeoff between recovery time and impact on throughput
- Hash table accesses fail during recovery
  - DDS lib, service, or WAN client, will retry

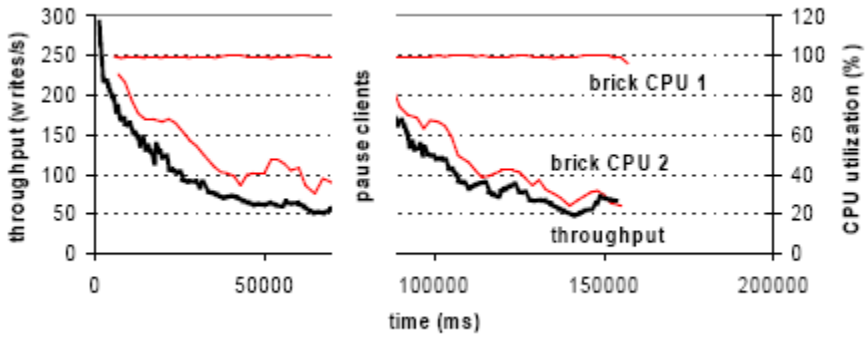
What is left out

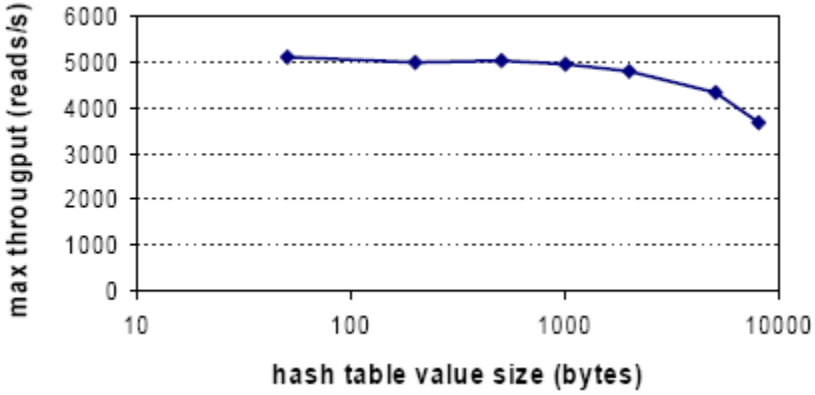
- Replica placement policy
  - How is a node selected to host a partition replica
- Replica migration policy
  - When does the system decide to migrate a replica
- Degree of replication

Performance









Availability and recovery