



Distributed Semantic Reconciliation of Replicated Data

Vidal Martins, Esther Pacitti, Patrick Valduriez
*Atlas group, INRIA and LINA,
University of Nantes*



Motivation

- Context: collaborative applications on P2P systems
 - Professional communities
- Need: multi-master replication
 - High-availability and high-performance
- Main problem: assure replica consistency
- Solutions in distributed systems
 - Synchronous replication: does not scale-up
 - Lazy master replication: blocks when master fails; bad performance due to unpredictable network delays
 - **Optimistic replication: enables asynchronous collaboration; requires divergence reconciliation**



Optimistic Replication

- Conflict Resolution Approaches
 - Ordering actions based on timestamps
 - Sequence of update arrivals at a primary site
 - **Semantic reconciliation**
 - The most flexible, because the user may specify the reconciliation criteria
 - Domain-specific engines: Ramsey & Csirmaz [2001]
 - General-purpose centralized engine: IceCube [2003]



Outline

- Reconciliation in IceCube
- Distributed Semantic Reconciliation
- Performance Evaluation
- Problems in P2P
- Conclusion



Reconciliation in IceCube (1)

- Replica: collection of objects
 - Relational table, XML document
- Action: application-specific operation
 - $R(K, A, B)$: R is a relational table, where K is the key attribute
 - Actions
 - a_1^1 : update R set $A=a_1$ where $K=k_1$
 - a_2^1 : update R set $A=a_2$ where $K=k_1$
 - a_3^1 : update R set $B=b_1$ where $K=k_1$, $\text{Parcel}(a_3^1, a_3^2)$
 - a_3^2 : update R set $A=a_3$ where $K=k_2$, $\text{Parcel}(a_3^2, a_3^1)$
- Constraint: application invariant
 - User-defined constraint: $\text{Parcel}(a_3^2, a_3^1)$
 - System-defined constraint: $\text{MutuallyExclusive}(a_1^1, a_2^1)$
- Cluster: set of actions related by constraints
 - $C_1 = \{a_1^1, a_2^1\}$, $C_2 = \{a_3^1, a_3^2\}$
- Schedule: set of ordered actions
 - $S = \{a_1^1, a_3^1, a_3^2\}$



Reconciliation in IceCube (2)

- Tentative updating of replicas
 - Each node performs tentative updates on its local replica and records them in its log
- Periodic scheduling by a central node
 - Merge all local logs
 - Resolve conflicting updates based on application semantics (2 sequential steps: produce clusters; order clusters)
 - Produce a global schedule
- Updating of replicas
 - Each node applies the global schedule to its local replica
- Yields eventual consistency
 - If nodes stop submitting new actions, all replicas eventually reach the same state



Outline

- Reconciliation in IceCube
- **Distributed Semantic Reconciliation**
- Performance Evaluation
- Problems in P2P
- Conclusion



Distributed Semantic Reconciliation (1)

- Assumptions
 - Small world (group and time locality)
 - Full multi-master replication



Distributed Semantic Reconciliation (2)

- Local action production
 - Each node performs local actions to update replicas respecting user-defined constraints
- Action storage
 - Local actions and associated constraints are stored in a distributed hash table (DHT)
- Distributed Reconciliation
 - Nodes retrieve actions and constraints from DHT and produce a global schedule
 - Distributed conflict resolution based on application semantics
 - The global schedule is executed at each node
- Yields eventual consistency
 - Idem IceCube



Distributed Semantic Reconciliation (3)

- Definitions

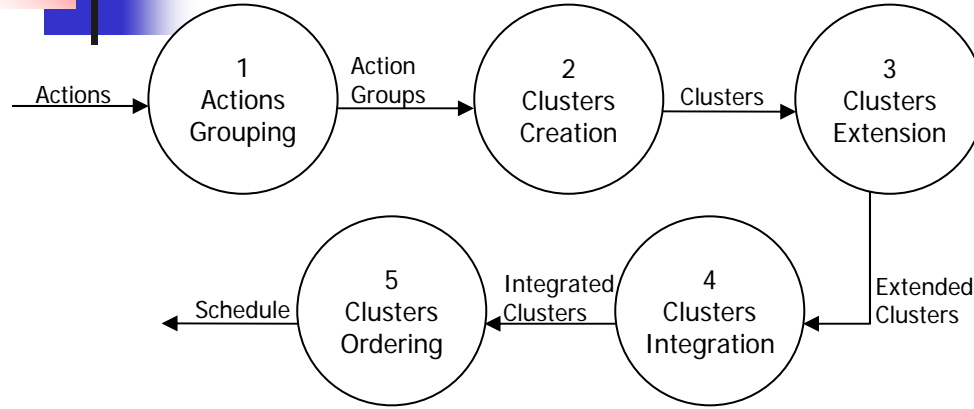
- Reconciliation object: object that holds data consumed by the reconciliation algorithm
- Object identifier: enables the storage and retrieval of reconciliation objects on DHT
- Reconciliation objects
 - Action log $R(L_R)$: stores actions coming from any node that tries to update the replica R
 - Action groups of $R(G_R)$: actions *suspect* of establishing semantic conflicts are gathered in action groups
 - Clusters set (CS): stores all clusters produced during reconciliation
 - Action summary (AS): captures semantic dependencies among actions and action memberships (AM)
 - Schedule (S): a set of ordered actions



Distributed Semantic Reconciliation (4)

- Steps of DSR algorithm
 1. Group actions that update common objects
 - Produces groups of potentially conflicting actions
 2. Create clusters based on action conflicts
 - Produces clusters of conflicting actions
 3. Extend clusters based on user-defined constraints
 - Produces extended clusters
 4. Integrate extended clusters
 - Produces integrated clusters
 5. Order integrated clusters' actions
 - Produces a schedule

Distributed Semantic Reconciliation (5)



a_1^1 : update T set $A=a1$ where $K=k1$
 a_2^1 : update T set $A=a2$ where $K=k1$
 a_3^1 : update T set $B=b1$ where $K=k1$, Parcel(a_3^1, a_3^2)
 a_3^2 : update T set $A=a3$ where $K=k2$, Parcel(a_3^2, a_3^1)

| Step | Node | Node Input | Node Output | Data Stored in DHT |
|------|----------------|--|--|---|
| 1 | n_1 n_2 | $\{a_1^1, a_2^1\}$ $\{a_3^1, a_3^2\}$ | $G_1 = \{a_1^1, a_2^1\}$ $G_2 = \{a_3^1, a_3^2\}$ | $G_T = \{G_1, G_2\}$ $G_I = \{a_1^1, a_2^1, a_3^1\}, G_2 = \{a_3^2\}$ |
| 2 | n_1 n_2 | $G_1 = \{a_1^1, a_2^1, a_3^1\}$ $G_2 = \{a_3^2\}$ | $C_1 = \{a_1^1, a_2^1\}, C_2 = \{a_3^1\}$ $AM_1 = \{(a_1^1, C_1), (a_2^1, C_1), (a_3^1, C_2)\}$ $C_3 = \{a_3^2\}$ $AM_2 = \{(a_3^2, C_3)\}$ | $CS = \{C_1, C_2\} \cup \{C_3\} = \{C_1, C_2, C_3\}$ $AM = AM_1 \cup AM_2$ |
| 3 | n_1 n_2 | $C_1 = \{a_1^1, a_2^1\}$ $C_2 = \{a_3^1\}, C_3 = \{a_3^2\}$ | $C_2^+ = \{a_3^1, a_3^2\}, C_3^+ = \{a_3^2, a_3^1\}$ $AM_3 = \{(a_3^2, C_2), (a_3^1, C_3)\}$ | $CS = \{C_1, C_2^+, C_3^+\}$ $AM = AM \cup AM_3$ |
| 4 | n_1 n_2 | $\{a_3^1, a_3^2\}$ $\{a_1^1, a_2^1\}$ | $C_4^{++} = C_2^+ \cup C_3^+ = \{a_3^1, a_3^2\}$ | $CS = CS \cup C_4^{++} = \{C_1, C_2^+, C_3^+, C_4^{++}\}$ $(C_4^{++} = C_2^+ \cup C_3^+) \rightarrow (CS = \{C_1, C_4^{++}\})$ |
| 5 | n_1 n_2 | $C_1 = \{a_1^1, a_2^1\}$ $C_4^{++} = \{a_3^1, a_3^2\}$ | $S_1 = \{a_1^1\}$ $S_4 = \{a_3^1, a_3^2\}$ | $S = S_1 \cup S_4$ $S = \{a_1^1, a_3^1, a_3^2\}$ |



Outline

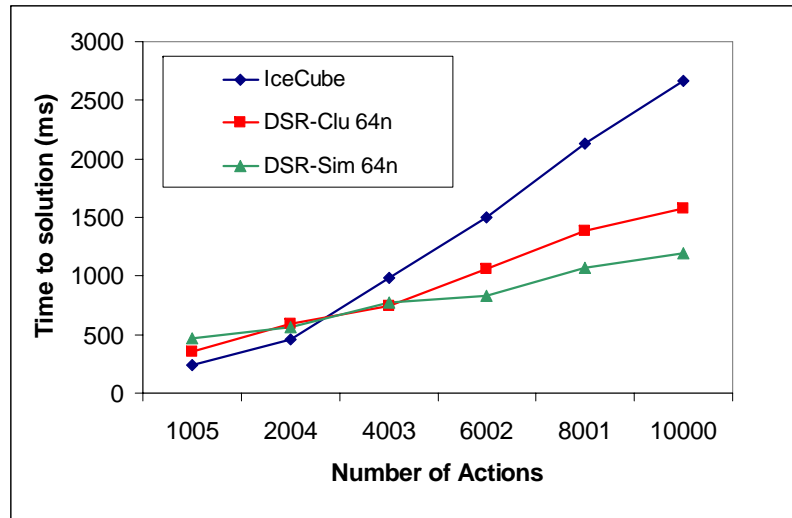
- Reconciliation in IceCube
- Distributed Semantic Reconciliation
- Performance Evaluation
- Problems in P2P
- Conclusion



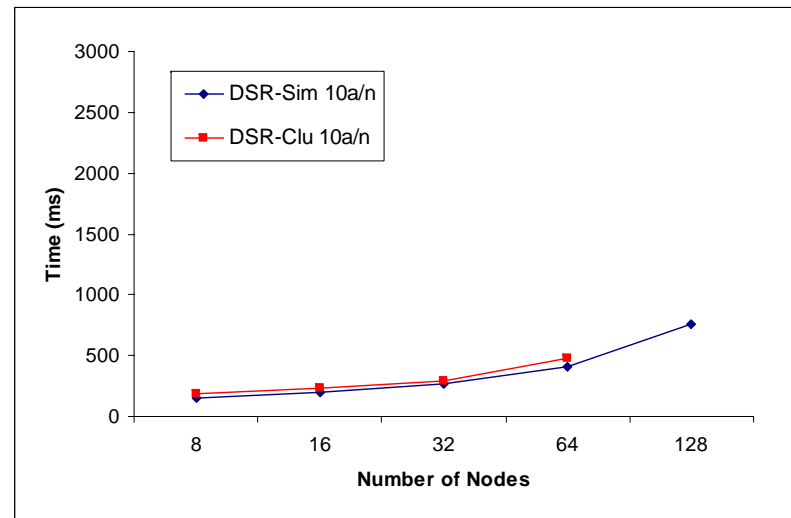
Performance Evaluation (1)

- Agenda application
- Application setup as IceCube
- Prototype
 - Linux and Java
 - Cluster: 64 nodes, 1 Gbps, Intel Pentium 2.4 GHz
- Simulator
 - Windows XP, Java and SimJava (a process based discrete event simulation package)
 - Intel Pentium 2.6 GHz, 1 Gb of RAM
- DHT Chord

Performance Evaluation (2)



Response time vs. number of actions



Scale-up with variable load



Outline

- Reconciliation in IceCube
- Distributed Semantic Reconciliation
- Performance Evaluation
- Problems in P2P
- Conclusion



Problems in P2P (1)

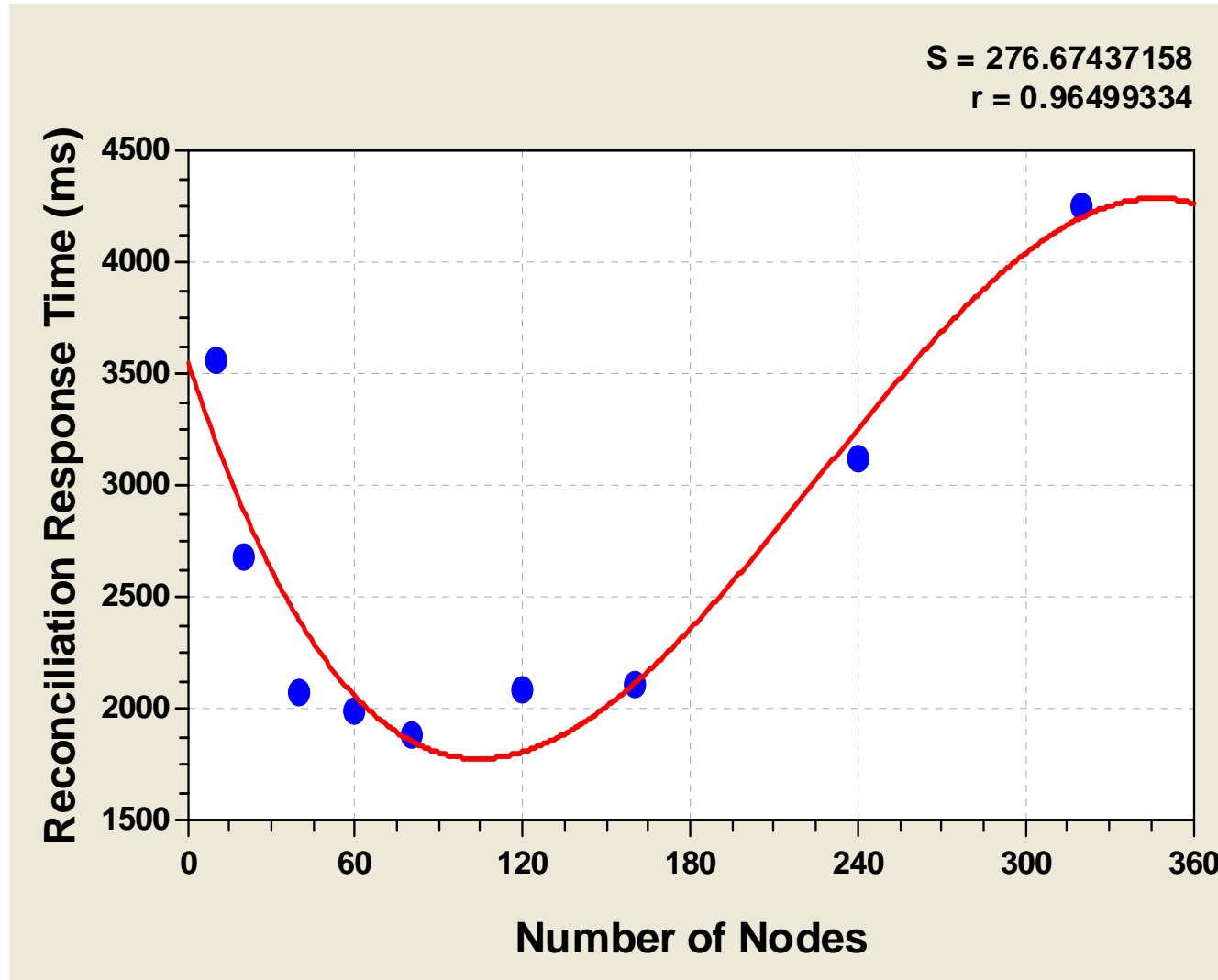
- No centralized information
 - Need for network communication during reconciliation
 - How to optimize network communication during reconciliation?
 - Distributed Hash Table (DHT)
 - Action partitioning based on the number of reconcilers (minimal number of messages)



Problems in P2P (2)

- Large number of peers
 - Network communication may dominate reconciliation time
 - How many peers should proceed as reconcilers?
 - Sampling through simulation
 - Regression analysis
 - Computation of minimal point
 - Which peers should proceed as reconcilers?
 - Cost-based peer allocation
 - Cost is a function of latency time
 - A distributed group management service manages cost information

Problems in P2P (2)





Problems in P2P (3)

- **Autonomy of peers**
 - Peers may join and leave at any time
 - How to assure reconciliation liveness?
 - Peers do not leave during reconciliation (spontaneously)
 - Peers try to run several rounds
 - How to assure reconciliation objects availability?
 - Single master replication over a small set of peers
 - Group communication support to assure consistency



Problems in P2P (4)

- Autonomy of peers
 - How to assure eventual consistency for disconnected peers?
 - A history of schedules is stored in DHT
 - $H = (S_1, S_2, \dots, S_i)$
 - Peers know the identifier of the last schedule locally applied (S_{last})
 - Whenever a peer p reconnects
 - p applies locally all schedules of H that succeed S_{last}
 - Actions locally produced by p while p was disconnected are stored in the involved action logs (DHT) for later reconciliation



Conclusion

- Distributed Semantic Reconciliation
 - Optimistic multi-master replication
 - Based on semantic reconciliation
 - Yields eventual consistency
 - Distributed data management (DHT)
 - Parallel processing
 - Evaluated using a simulator and a prototype
 - Good performance and scale up
- Future work
 - Cost-based allocation of reconciler nodes
 - Dealing with joins and leaves