A Framework for Efficient Inconsistency Detection in a Grid and Internet-Scale Distributed Environment

Yijun Lu and Hong Jiang
Department of Computer Science and Engineering, University of Nebraska-Lincoln
{yijlu, jiang}@cse.unl.edu

Inconsistency avoidance vs. Inconsistency detection

- Inconsistnecy avoidance predefines consistency control protocols and enforces the system to obey the protocols.

- Inconsistency detection detects inconsistency in a *timely* manner when it occurs instead of avoiding it in the first place.

Why inconsistency avoidance is not suitable for an Internet-scale environment?

- Protocols considered: strong consistency protocols and optimistic protocols.

- For strong consistency control protocols:
  (1) It can be very costly to maintain because of the memebership maintenance and strict protocol enforcement cost.
  (2) Frequent packet loss in such a large scale network makes it difficult, if not impossible, to maintain a perfect protocol.

- For optimistic consistency control protocols:
  (1) They relieve the costly maintenance and strict enforcement cost associated with strong consistency protocols.
  (2) But, they are still *predefined*. It can be either overkill or insufficient in an environment where many applications are deployed.

Advantages of an efficient inconsistency detection framework

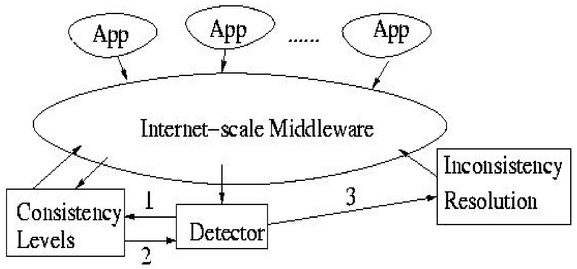
- It removes the costly membership management requirement that is used to enforce a consistency control protocol in the first place.

- By ensuring that the potential inconsistent bahavior be detected in a timely manner, it can control the inconsistency degrees.

- After an inconsistency is detected, the system can respond based on the applications' semantics.

A Framework for Efficient Inconsistency Detection

(1) Framework overview



- Multiple applications share data and service through the support of an Internet-scale middleware.

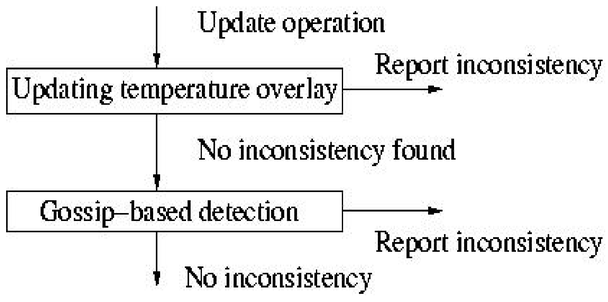
- The inconsistencies among nodes are detected by the detector.

- Upon detection, the detector consults with the inconsistency level monitor (step 1 and step 2) before reaction is initiated.

- Based on the applications' semantics, if the inconsistency is tolerable, the detector does not react.

- Otherwise, the detector informs the inconsistency resolution module to resolve the inconsistency (step 3).

(2) Timely inconsistency detection



- Basic idea: build an overlay on top of the underlying network based on nodes' updating history.

- The top layer is based on nodes' updating history, or updating temperature and is called "temperature overlay".

- The bottom layer is a backup and uses gossip-based inconsistency detection.

- The bottom layer is only triggered when the top layer fails to detect the inconsistencies.

(3) Design issues of the two-layer inconsistency detection module

- Meauring the updating patterns (by tracking updating frequency for a certain file)

- Learning the updating patterns by RanSub protocol.

- Interest-group based temperature collection/distribution to save network bandwidth

- Caching (for both temperature and routing information) to imporve the efficiency; garbage collection to remove stale states.

(4) Inconsistency resolution

- Determine whether to resolve the inconsistency based on applications' semantics.

- Two ways for the system to learn applications' semantics.
  (a) Users specify their preferences.
  (b) System passively monitor the users bahavior.

- Two ways to actually resolve the inconsistency
  (a) Middleware resolve it automatically.
  (b) Flag "inconsistency" and ask for human intervention.

Evaluation I: Probability that the top layer fails.

- Workload used in anlysis: each node changes its interested file with rate *r*.
- One rule: when a user becomes interested in a file, it reportes this interest one epoch (a term from RanSub, basically, it is a round of information collection/distribution) ahead.

- For two writers, *A* and *B*, the top layer fails in the following two cases:
  (1) One of them is not in the temperature overlay.
  (2) Both of them are not in the temperature overlay.
- Probability of failure under a typical configuration are:
  (1) 18.9%; (2) 0.04%

- Conclusions:
  (1) The probability of failure is generally small.
  (2) The probability of failure is very close to 0 if the target file becomes hot and attract many simutaneous writers.

Evaluation II: Maintenance cost

- We run the two-layer inconsistency detection module for 800 seconds. By the end of the simulation, we collect the number of messages received by each node and the result is listed below.

Maintenance cost

Max	Mean	Median
4680	51.9	26

- Conclusion:
  While the Max looks high, it is accumulated over 800 seconds. Even if the size of a message is 1KB, the network bandwidth cost is still only 6KB/s for the root (for RanSub tree), which bears the Max number.
  Thus the maintenance cost will not overwhelm the root.

Discussion

- Application background
- Is timely detection good enough? If not, what are other related research issues?

Future work

- Bound the detection delay
- Explore possible inconsistency resolution schemes

References

[1] P.T. Eugster, R. Grerraoui, S.B. Handurukande, et. al. Lightweight Probabilistic Broadcast, DSN 2001
[2] D. Dullmann, W. Hoschek, J. Jaen-Martinez, et. al. Models for Replica Synchronization and Consistency in Data Grid, HPDC 2001.
[3] D. Kostic, A. Rodriguez, J. Albrecht, et. al. Using Random Subset to Build Scalable Network Services, 4th USENIX Symposiou on Internet Technologies and Systems (2003)