Toward Fast Eventual Consistency with Performance Guarantees

Feng Yan[1], Alma Riska [2], Evgenia Smirni[1]

[1] College of William & Mary, Williamsburg, VA, USA, {fyan, esmirni}@cs.wm.edu

[2] EMC Corporation, Cambridge, MA, USA, alma.riska@emc.com

EMC² where information lives

Abstract

Systems have adopted the notion of *eventual consistency, which* means that the targeted redundancy of data in the system is reached *asynchronously,* i.e., outside of the critical path of user traffic, so that performance of user traffic is impacted minimally. Here, we propose a scheduling framework that makes decisions about when to schedule the asynchronous tasks associated with new or updated data such that they are completed as soon as possible without violating user traffic quality targets. At the heart of the framework lies a learning methodology that extracts the characteristics of idle periods and infers the average amount of work to be filled during periods of idleness so that asynchronous tasks are completed transparently to the user.

Eventual Consistency

- In distributed systems, data is distributed across multiple nodes and geographic locations.
- Eventual consistency [1]: data is updated asynchronously.
- Inconsistency window: time between Active and the last Inactive node acknowledgement.
  - Important because it reflects the reliability of system.
  - Active node: the node that receives the new data.
  - Inactive node: the nodes that would receive replicas asynchronously.
- Buffering is required at the Inactive node to protect performance.

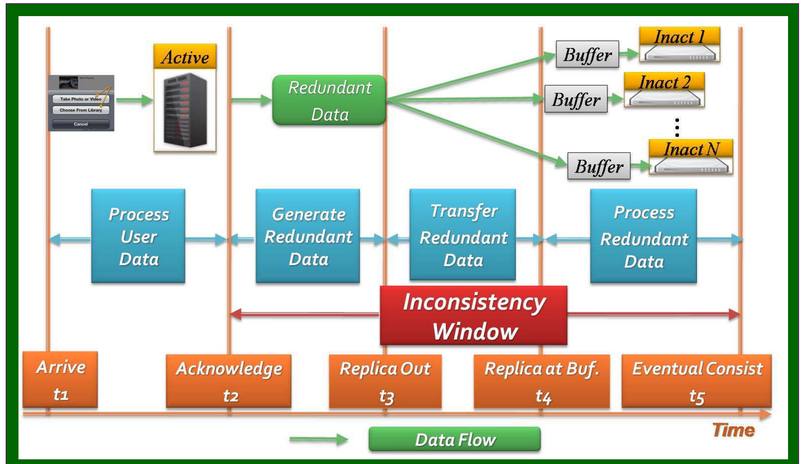

Fig. 1. Schematic View of the system with asynchronous replication and eventual consistency.

Scheduling Framework

1. The framework in [2] computes *(I, T)* tuple from *CDH* of idle intervals:
   - ***I***: idle wait time before scheduling.
   - ***T***: the time to serve background jobs.
   - ***CDH****:* Cumulative Distribution Histogram of idle intervals.
2. The scheduling in [3] schedules asynchronous updates:
   - ***D***: the user-provided average relative performance degradation target.
   - ***RT***: the average IO request response time without asynchronous updates.
   - ***W***: extra IO request wait time due to asynchronous updates.
   - B_W: the workload defined average replication work amount target.
   - B_{BG}: the average replication work under *(I, T)* scheduling.
   - Performance guarantees: $D \geq W / RT$.
   - No backlog guarantees: $B_{BG} \geq B_W$.

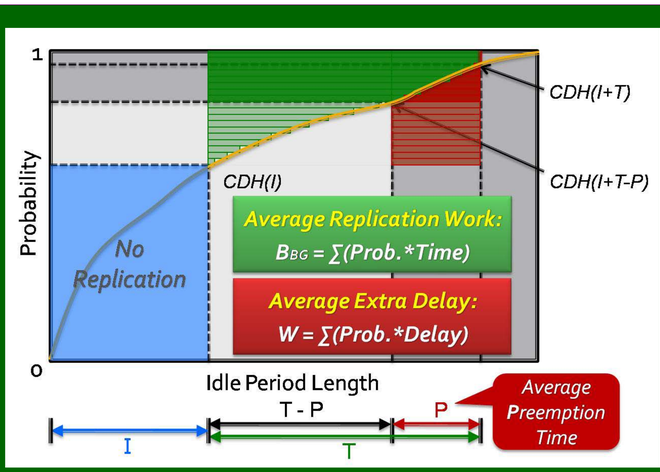

Fig. 2. Use CDH to estimate the average extra delay and average replication work finished.

Performance Evaluation

General Trace Description (Duration: 168 hours)

Microsoft SNIA IOTTA Repository

Trace	UTIL (%)	Average Arrival Rate	Average Service Rate	Average Response Time	Idle Mean (ms)	Idle C.V.	R/W ratio
usro	1.07	0.0012	0.1203	8.94	805.4	1.74	0.11
mdso	0.52	0.0007	0.1412	7.21	1404.2	1.93	0.03
tso	0.61	0.0008	0.1455	7.06	1150.2	1.74	0.04
webo	0.72	0.0010	0.1468	7.12	959.7	2.11	0.13

Fig. 3. General Trace Information. ms stands for millisecond.

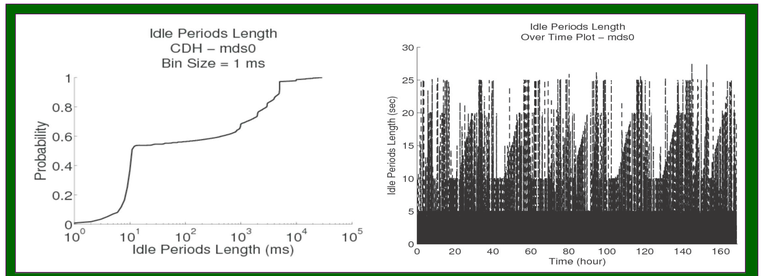

Fig. 4. CDH of idle period lengths (left) and the idle periods lengths overtime plots (right).

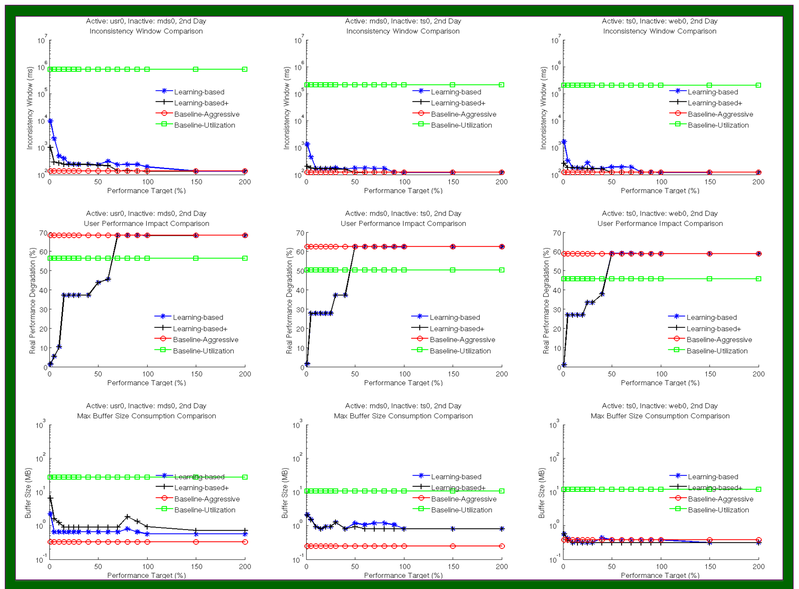

Fig. 5. Inconsistency Window (the first row), user performance impact (the second row), buffer consumption (the third row) comparison between different scheduling for various active-inactive pairs.

References

[1] W. Vogels, "Eventually consistent," *ACM Queue, vol. 6, no. 6, pp. 14–19, 2008.*

[2] N. Mi, A. Riska, X. Li, E. Smirni, and E. Riedel, "Restrained utilization of idleness for transparent scheduling of background tasks," in *Proceedings of the Eleventh International Joint Conference on Measurement and Modeling of Computer Systems, SIGMETRICS/Performance, 2009, pp. 205–216.*

[3] F. Yan, A. Riska and E. Smirni, "Toward Fast Eventual Consistency with Performance Guarantees", in *Proceedings of 9th ACM international conference on Autonomic computing, 2012 (to appear).*