Consistency Metric for Distributed Datastores

Kyrylo Rukkas, Galyna Zholtkevych

V.N. Karazin Kharkiv National University, Ukraine

International Conference Taapsd – 2017

Kyiv, 4 – 8 Dec 2017

Agenda

- Motivation:
  - need to make consistent datastores
  - consistency problem of a distributed datastorage
  - need to present consistency state as stochastic value.
- Background:
  - eventual consistency definition;
  - consistency model expanded with new definitions;
- Formula to define inconsistency state. Example
- Graphics for inconsistency stochastic value
- Consistency convergence proposition. Proof
- Graphics for consistency convergence value
- Conclusions and future work

Motivation

- Consistency is not a new problem in datastores, especially if they are huge
- At this point in large DS it become more usable to present consistency state as stoschastic value.

Background

ACID vs BASE strategy

- ACID (Atomic, Consistent, Isolated, Durable) mean that once a transaction is complete, its data is consistent and stable on disk
- BASE (Basic Availability, Soft-State, Eventual Consistency) values availability, but does not offer guaranteed consistency of replicated data at write time

Eventual consistency definition

Consistency is called eventual if no updates take place for a long time and all replicas will gradually become consistent

Evolved model for consistency

Our model is a tuple (N, L, ∂, D, r), where
N – a finite set of nodes;
L – a finite set of links;
$\partial : L \to 2^N$ – a mapping that associates each link with two nodes;
D – a finite set of stored data units;
$r : D \to 2^N$ – a mapping that associates each data unit d with a subset of nodes that store its replica. N_d – a finite set of nodes that are having given dataunit d;
$l(N_d)$ – a number of nodes in datastore that are having given dataunit;
n_c – a number of nodes in a subset of N_d where all nodes have the same replica.

Formula and example

Consistency formula derivation

$$p_c = \frac{n_c}{l(N_d)} \cdot \frac{n_c - 1}{l(N_d) - 1} \tag{1}$$

$$l(d) = 1 - \sum_{k=1}^{K} \frac{N_k(N_k - 1)}{N(N-1)}. \tag{2}$$

Example

$l(5) = 0$

$l(4+1) = \frac{2}{5}$

$l(3+1+1) = \frac{7}{10}$

$l(2+1+1+1) = \frac{9}{10}$

$l(1+1+1+1+1) = 1$

$l(3+2) = \frac{3}{5}$

$l(2+2+1) = \frac{4}{5}$

Demonstration of experiments

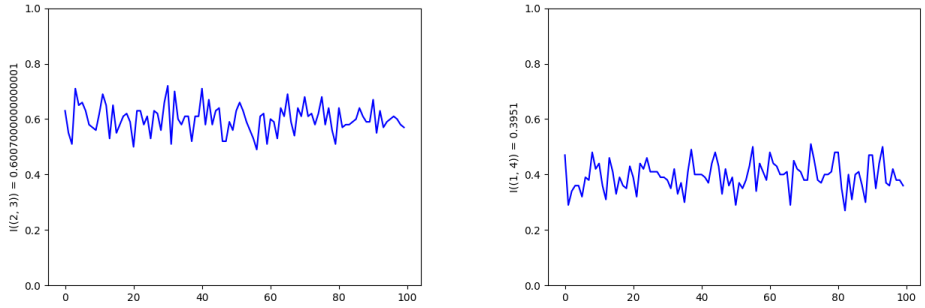


Figure: The datastore has two consistent partitions

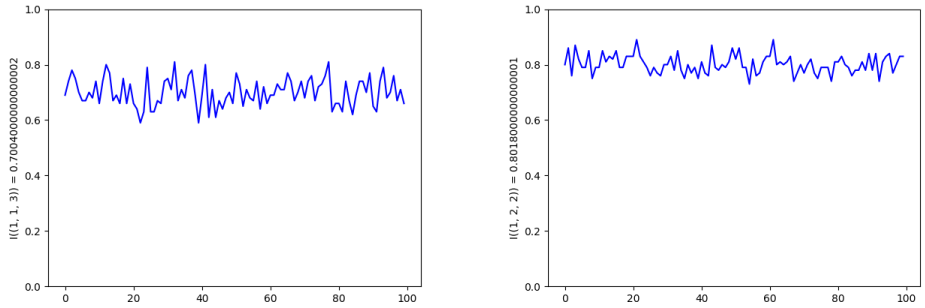


Figure: The datastore has three consistent partitions

Proposition about consistency convergence

Let we have a distributed datastore where all links are available and reliable (network partitions do not happen in a datastore and nodes are stable and respond in approximately equal time); the interval between writing operations is t_w. Then t_w is less than the diameter of network graph ensures eventual consistency of the datastore.

Brief proof

1. n_1, n_2 are outermost each from other
2. Time slots t taken from n_1 to n_2 is the shortest path from n_1 to n_2
3. From definition of the diameter $T_c = diameter(G)$

Kyrylo Rukkas, Galyna Zholtkevych

Demonstration of experiments

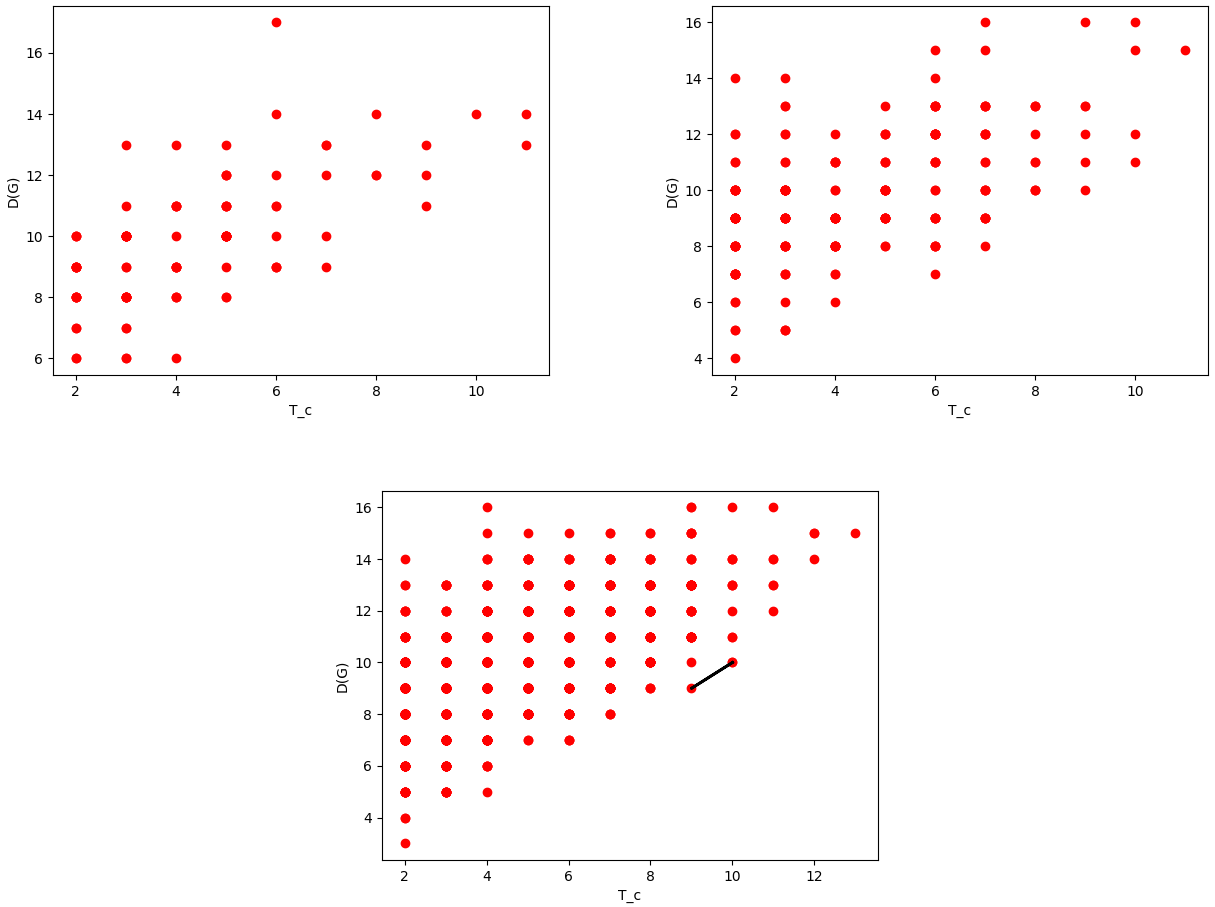
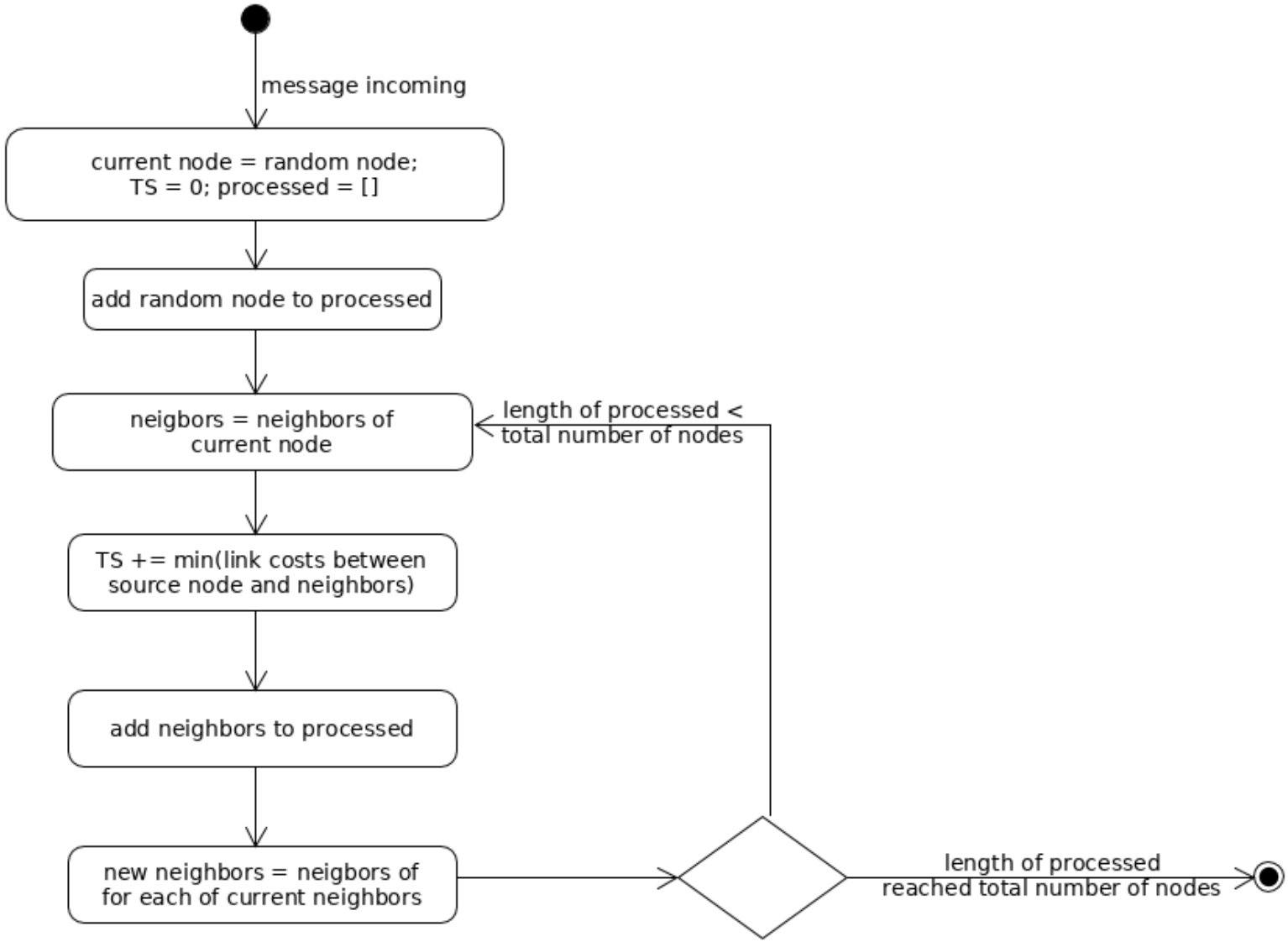


Figure: Graphics for consistency convergence time on weighted graph

Kyrylo Rukkas, Galyna Zholtkevych

Activity diagram of simulation process



Conclusions

- the sufficient condition for distributed datastore can be choosing network topology where frequency of write operations will be no greater than diameter of the graph;
- if frequency of write operations is grater than diameter of the graph, it may be useful to evaluate current inconsistency state of the graph;
- if inconsistency state is close to 0 enough for current requirements, it means that there are inconsistent nodes that are far enough to not conflict with new replicas. The next source can be chosen from consistent list and far enough for inconsistent ones;
- if still more strict consistency needs to be satisfied and inconsitency state is close enough to 0, the node closest to the source node where previous write operation occurred can be chosen for writing;
- if replica's history is not important for requirements, it may be possible to solve the problem fixing conflicts.

Future work

- complicate a datastore introducing different availability for each node;
- complicate a datastore introducing various network partitions for a datastore;
- provide an algorithm that will allow to make a decision what is the best approach for a given datastore such that the eventually consistency is satisfied or an approach when inconsistency will not impact on datastore requirements;
- theory for finding the most actual replica and time complexity for this algorithm
- consider all the proposed above metrics in various kinds of real topologies.