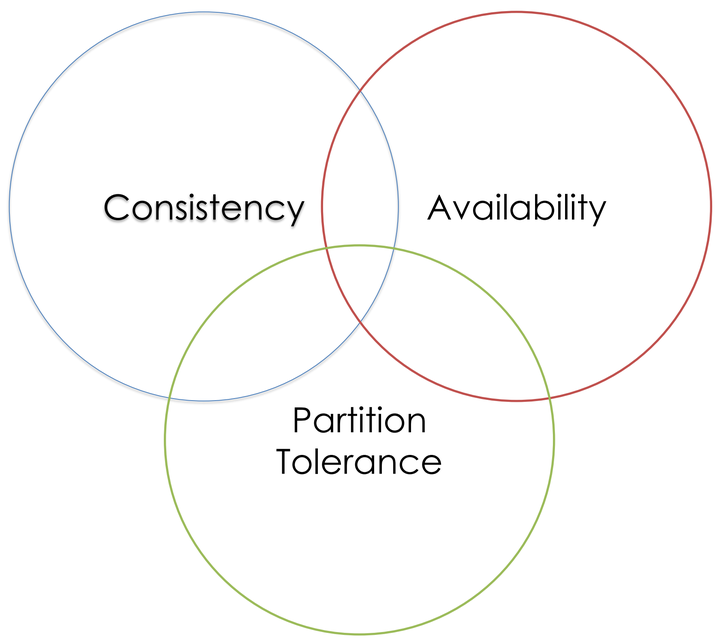
Evaluating Performance of Anti-Entropy Algorithms

Kenny Gao and Sriram Mohan

Introduction

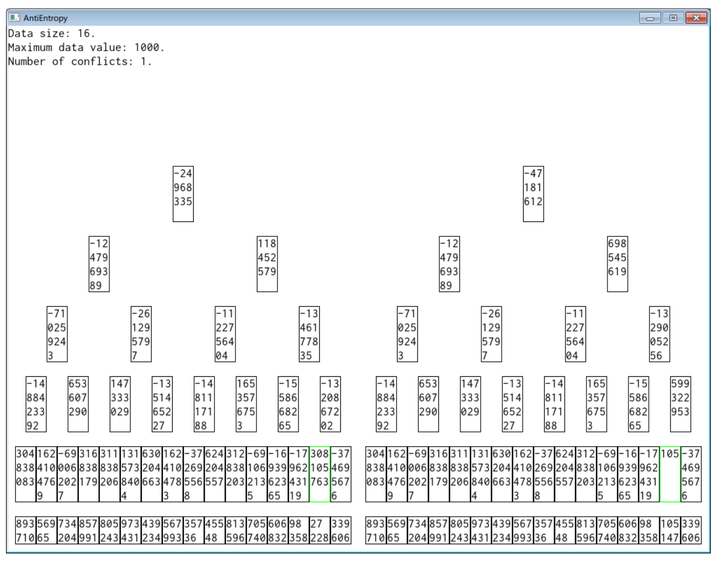
Brewer's CAP Theorem



Pick two.

Visualization

Algorithm visualization as a pedagogical tool



Visualization of Merkle Tree-Based Algorithm

Evaluating Performance

What performance can be:

- Network usage
- Speed (time)
- Robustness/resistance to failure

What to measure:

- Pairwise data comparison (naive)
- Naive algorithm with single hash of all data
- Naive algorithm with multiple layers of hashing
- Standard Merkle tree algorithm (full hash tree)
- Variations: non-binary trees, lazy hash computation

Approach and Implementation

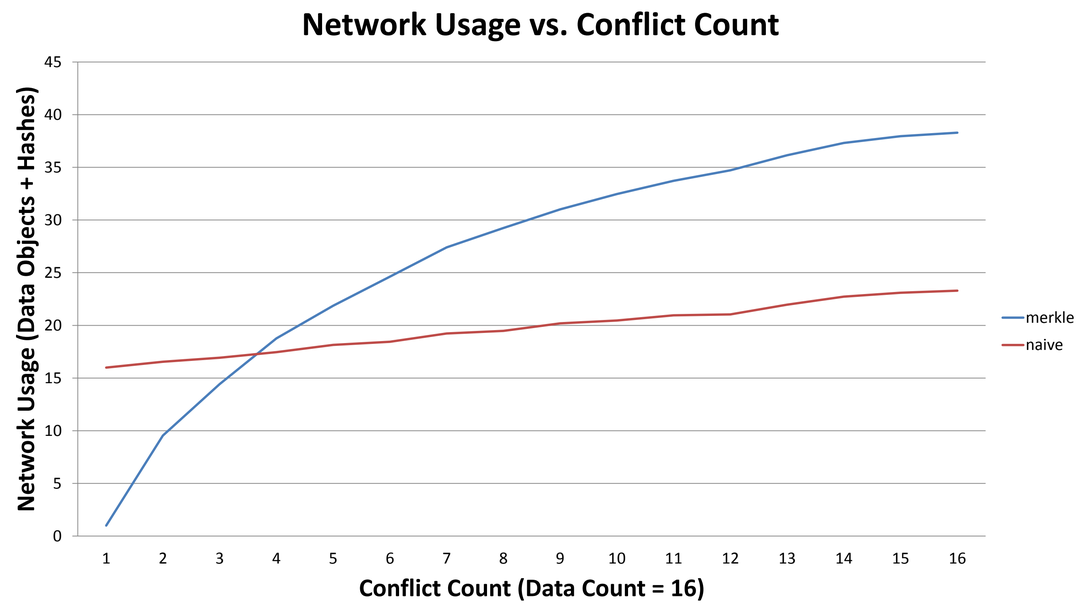
Given the following variables:

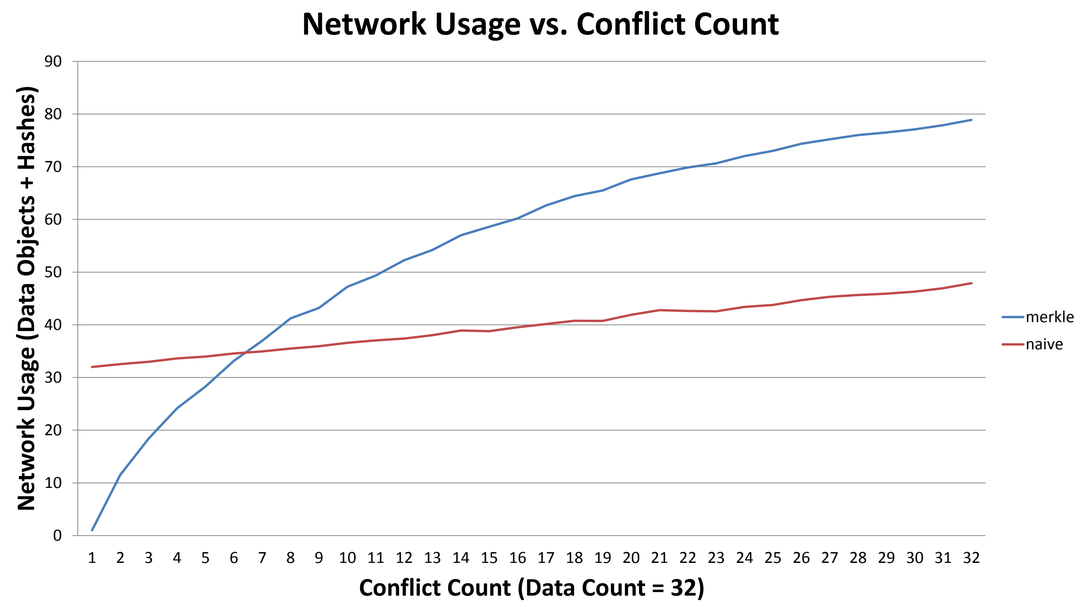
- # of data items stored on replicas
- # of conflicts between replicas

Measure each algorithm's performance in terms of network usage.

Naive and standard Merkle tree algorithms implemented in Python and repeatedly simulated on randomly-generated data for each configuration of variables.

Results





Each data point is the mean value of 100 simulations of an algorithm for some configuration of variables.

Discussion and Further Work

Results seem to scale with data count.

merkle outperforms *naive* when conflict count is low, and vice versa.

naive grows linearly with respect to conflict count, whereas *merkle* grows logarithmically (approximately).

Further work:

- Implement remaining algorithms
- Extend performance evaluation to speed (quantitative)
- Extend performance evaluation to robustness (qualitative)

Conclusions

The standard Merkle tree algorithm is superior to the naive algorithm in terms of network usage provided that the conflict count is below the threshold of roughly 22% of the data count. Above the threshold, the naive algorithm is superior.