Self Tuning Data-stores for Geo-distributed Cloud Applications

Shankaranarayanan P N, Ashiwan Sivakumar, Sanjay Rao, Mohit Tawarmalani

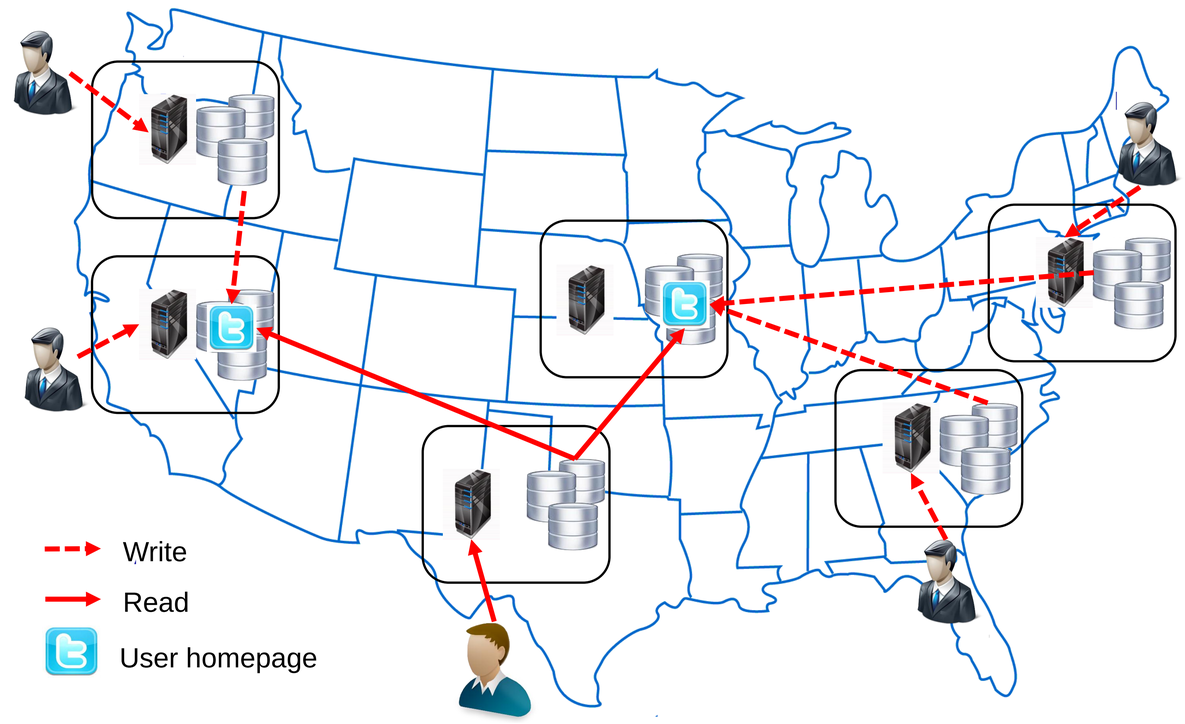
Purdue University

Interactive Online Applications

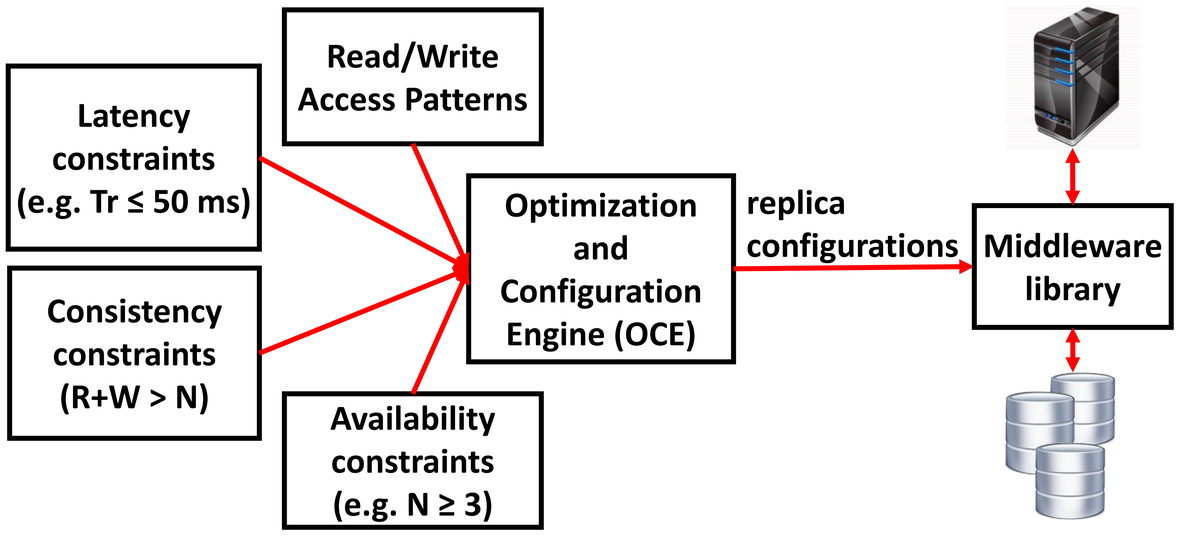
- Latency, availability and consistency are ALL critical
- Solutions:
  - ✓ Latency → higher replication (e.g., CDNs)
  - ✓ Availability → geo-redundancy (e.g., geo-distributed DBs)
  - ✓ Consistency → quorum protocols (e.g., Dynamo, Cassandra)

Challenges

- Can I achieve all the three for my application?
- Will my application meet client's SLA constraints?
- How do I optimally balance the contrasting requirements?
- Is there a feasible solution for a given set of constraints?
- Can this complex decision making be automated?

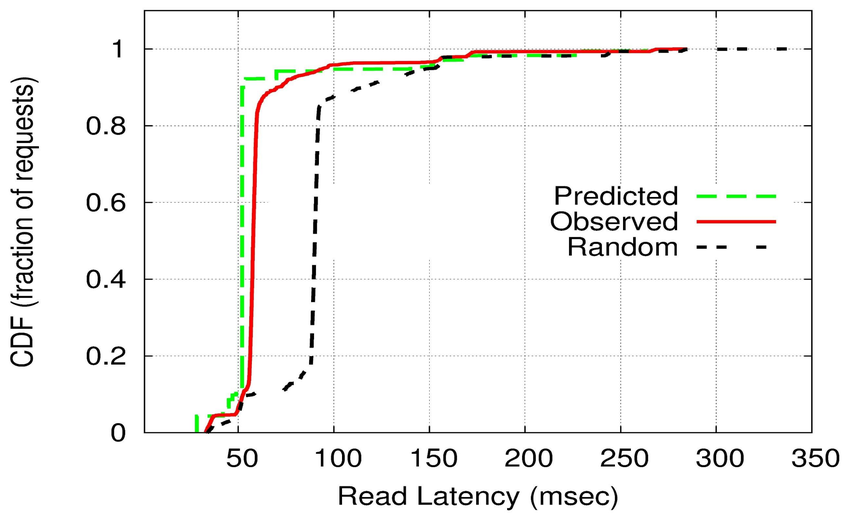


System Design



Additional inputs

- ✓ Percentiles of requests that needs to be optimized
- ✓ Priority of reads and writes



Experimental evaluation of our models on EC2 test-bed

- Evaluation on a Cassandra cluster with 27 nodes (DCs) using real world application traces - Twitter, Wikipedia
- Location of DCs similar to AWS edge locations
- Inter DC delays measured using planet-lab nodes and emulated using dummynet
- Failure of DCs emulated by shutting down nodes in the Cassandra cluster

Observations

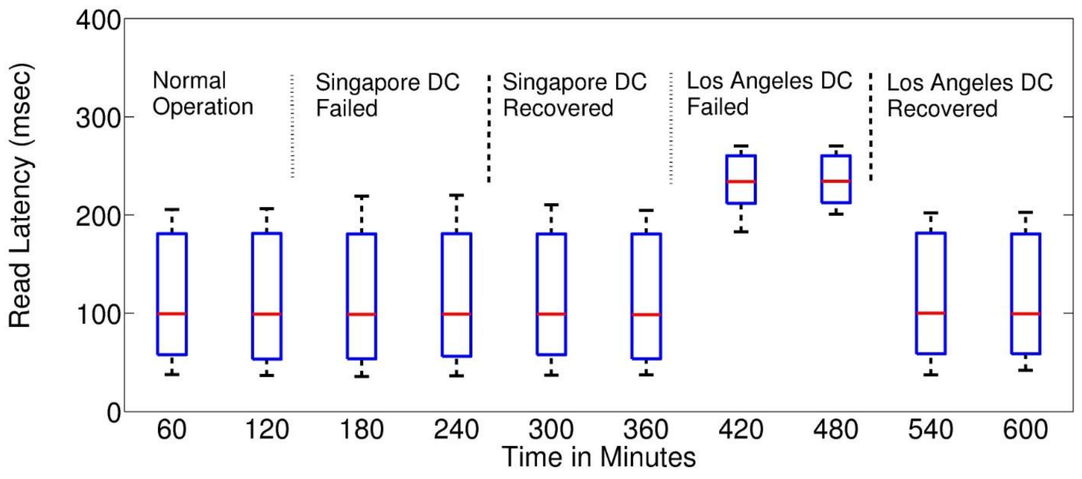
- Models predictions are very close to reality
- Models perform significantly better than naïve, off-the-shelf configurations (e.g. random partitioning)
- Failure of different DCs can result in different performance

Our approach

- Develop models that capture the relationship between Latency, Availability and Consistency
- Build optimization framework using the models

Model goals

- Automatically determine the replication configuration parameters for "buckets" of data items
  - ✓ number (N) and location of replicas
  - ✓ read/write quorum size (R,W)
- Honor application availability and consistency constraints
- Leverage workload characteristics to minimize latency
  - Geographical distribution of accesses
  - Asymmetry between the reads and writes
  - Relative priority between the reads and writes



Conclusions

Our optimization models

- evaluate limits on achievable performance given application constraints and a given workload
- show the importance of choosing different replication strategies across different buckets
- highlight the significant benefits of optimizing for the optimal latency percentiles
- show that explicit modeling of performance under failure is critical for good performance
- are embarrassingly parallel, flexible and easily extensible – replica configuration can be automated at scale

PURDUE UNIVERSITY