The Costs and Limits of Availability for Replicated Services

Haifeng Yu, Amin Vahdat
[SOSP 2001]

Presented by: Fahad Arshad, Ignacio Laguna

Dependable Computing Systems Lab
School of Electrical and Computer Engineering
Purdue University

INTRODUCTION

• Explores benefits of dynamically trading consistency for availability using a continuous consistency model.

• Continuous Consistency Model
  – Applications specify a max deviation from strong consistency on a per-replica basis.
  – Optimistic consistency models leave this deviation unbounded.
  – Continuous consistency model exposes a tradeoff between consistency and availability that can be dynamically varied based on changing network and service characteristics.
INTRODUCTION

• Goal
  – Enable services to tune their system availability as their workload changes and as network reliability changes

• Main contributions
  – Evaluate availability as a function of consistency level, consistency protocol, and failure characteristic.
  – Maximizing availability requires as strong a consistency level as possible during times of full connectivity. This is required to build up a large "cushion" for the times when failures prevent communication.
  – Demonstrate that simple optimizations to existing consistency protocols result in significant availability improvements.
  – Provide tight upper bounds on the availability of services.

BACKGROUND

• Motivation
  – 0.1% availability improvement gives 8 more hours of uptime corresponding to approximately $1 million in additional revenue for every $1 billion in annual revenue conducted online.

• Study TACT consistency model [Yu, Vahdat OSDI 2000]
  – TACT gradually reduces the amount of required synchronous communication among replicas in moving from strong to optimistic consistency.
  – The model allows replicas to locally buffer a max number of writes before requiring remote communication.
  – At any replica, updates can be in either TENTATIVE or COMMITED state.
Consistency Metrics

• Three per replica consistency metrics
  
  – *Numerical error* (NE) is the max weight of writes not seen by a replica.
  
  – *Order error* (OE) is the max weight of writes that have not established their commit order at the local replica.
  
  – *Staleness* (STL) is the max amount of time before a replica is guaranteed to observe a write accepted by a remote replica.
  
• Strong consistency is defined as NE=0, OE=0, STL=0

• Optimistic consistency is defined as NE=∞, OE=∞, STL=∞

• Consider a TACT application as a replicated airline reservation system.

TACT application Example

– NE corresponds to the maximum number of system-wide reservations that have not been propagated to the local replica.

– OE corresponds to the maximum number of tentative reservations in a replica’s local view i.e. have not established the final COMMIT.

– NE bounds the max rate of conflicting reservations, OE bounds the rate of false negatives, staleness guarantees the max elapsed time before a reservation is seen system-wide.

(Assume Serialization Order = W1 W2 W3 W4)
Assumptions and Methodology

- Replica failures are modeled as singleton network partitions.
- Failures are symmetric. (Approximation to protocols like TCP)
- They do not assume reachability among hosts to be transitive.
- CPU processing time and network delay is negligible compared to the duration of time where network connectivity does not change.
- TACT supports the notion of application-specific consistency units (or conits) that determine the granularity over which consistency is enforced.

Definitions

- Availability is defined over submitted accesses from the client to the network service.
  - A failed access if the request cannot reach any replica because of network failures.
  - A rejected access if it is received by some replica but its acceptance would violate some consistency requirement.
  - An accepted access otherwise.
- $\text{Avail}_{\text{client}} = \frac{\text{accepted accesses}}{\text{submitted accesses}}$.
- $\text{Avail}_{\text{network}} = \text{the } \% \text{ of accesses that can reach a replica}$
- $\text{Avail}_{\text{service}} = \text{the } \% \text{ of accesses reaching replicas that are actually accepted}$.
- $\text{Avail}_{\text{client}} = \text{Avail}_{\text{network}} \times \text{Avail}_{\text{service}}$
**Workload & Faultload Approach**

- Investigate $Avail_{service}$ using a workload & faultload approach.

- A workload is a trace of time stamped accesses.

- A faultload is a trace of timestamped fault events.

- A fault event is either a failure that divides an existing network component into two components or a recovery that merges two existing components.

**Deriving Approach to Availability Upper Bounds**

- At a higher level, any consistency maintenance protocol must answer a number of questions to achieve a target level of consistency among replicas.
  - The protocol must determine which writes to accept/reject from clients
  - The protocol must determine when and where to propagate writes.
  - The protocol must decide the serialization order i.e. which writes to commit and in what order.

- Divide these questions into two disjoint sets:
  - $Q_{offline}$ is the set of questions with optimal answers that can be pre-determined offline.
  - $Q_{online}$ contains all remaining questions whose optimal answers depend on consistency level, workload, and faultload.

- For proving upper bounds on $Avail_{service}$, it is necessary to search for the optimal answers to these questions.

- Problem: the set of possible answers is exponential.
Deriving Approach to Availability Upper Bounds

• A key challenge is to make the search of the set of possible answers tractable by proving that certain types of answers will always result in better availability than others.

• Using the pre-determined optimal answers to $Q_{\text{offline}}$, an abstract dominating algorithm is constructed.

• By def dominating algo makes strictly better decisions then other protocols for $Q_{\text{offline}}$.

Deriving Approach to Availability Upper Bounds

• Dominating algorithm does not specify the answer of any question from $Q_{\text{online}}$, rather it takes some inputs that specify these answers.

• For a given workload and faultload, we say that a consistency protocol $P_1$ dominates protocol $P_2$, if i) $P_1$ achieves the same or higher level of availability as $P_2$ and ii) $P_1$ maintains the same or higher level of consistency as $P_2$.

• The upper bound is the availability achieved by the protocol $P$ that dominates all protocols.
Availability Upper Bound as a Function of Order Error

- Order error bounding protocol needs to answer three kinds of questions:
  - questions regarding write propagation
  - questions regarding write acceptance
  - questions regarding write commitment.
- To commit a write, a replica must see all preceding writes in the *serialization order*
- *Serialization order* is the global total write order that an OE bounding protocol tries to maintain across all replicas
- Consider a system with two replicas that are partitioned from each other
  - Suppose *replica*₁ receives \( W_1 \) then \( W_2 \), while *replica*₂ receives \( W_3 \) then \( W_4 \). A serialization order here can be any permutation of the four writes.
  - If the serialization order is \( S = W_1W_2W_3W_4 \), then a replica can only commit \( W_3 \) after it sees and commits \( W_1 \) and \( W_2 \).

Availability Upper Bound as a Function of Order Error

- If *replica*₂ propagates \( W_3 \) to *replica*₁ before *replica*₁ accepts \( W_2 \), then *replica*₁ cannot commit \( W_3 \) and its order error is increased.
  - Therefore aggressive write propagation always reduces NE, Staleness
  - In certain cases it can actually increase the OE.
- Optimizing approach for aggressive write propagation.
  - Remote writes seen by a replica are not immediately applied to data store and thus do not count towards OE.
  - Apply them only when they can be committed. (Increases NE)
  - Local writes are always applied to the data store immediately.
Availability Upper Bound as a Function of Order Error

- To optimize search on serialization orders, a small set of serialization orders that are strictly "better" than all other serialization orders is found.

- $S$ dominates $S'$ if $S$ allows the commitment of any write that could be committed using $S'$. From previous example
  - Serialization order $S = W1W2W3W4$ dominates $S' = W2W1W3W4$
  - *This is because whenever $W2$ can be committed using $S'$, the replica must have already seen $W1$ (which is accepted before $W2$), and thus can also commit $W2$ in $S$.*

Availability Upper Bound as a Function of Order Error

- Using this definition of "domination", the prove that $S$ dominates $S'$ outlined .
  - *ALL* All possible serialization orders.
  - *CAUSAL* Serialization orders compatible with causal order.
  - *CLUSTER* Serialization orders where writes accepted by the same partition during a particular interval cluster together.

- From previous example of 2 replicas
  - *ALL* contains all possible permutations of the four writes
  - *CAUSAL* contains the six orderings where W1 precedes W2 and W3 precedes W4.
  - *CLUSTER* only contains W1W2W3W4 and W3W4W1W2

- They prove that CAUSAL dominates ALL and CLUSTER dominates CAUSAL, so the upper bound becomes tractable by restricting our search scope to CLUSTER.
Availability Upper Bound as a Function of Order Error

• To prove CAUSAL dominates ALL, only need to show that if write W1 causally precedes write W2, then it is always "better" to place W1 before W2 in the serialization order.

• The proof of CLUSTER dominating CAUSAL is intricate.

• For each serialization order enumerated, they derive the upper bound by solving a linear programming problem.

IMPLEMENTATION

• Sample Faultloads
  – Collect a sample of Internet connectivity with average measurement intervals of 3 seconds on each path. (previous intervals 10 min)
  – Measure interconnectivity among 8 sites.
  – Total duration of the trace is 6 days with over 12 million samples.
  – Faultload has an average failure time on all paths of 0.046%.
  – Focus on the first day of this trace (SAMPLED1). Failure rate=0.17%
  – Use a simple event-driven simulator (Internet topology generator) to obtain diverse faultloads based on a sample Internet-like topology.
  – 24 backbone routers in the sample topology (vary replica 1 to 24)
  – Use exponential distributions for both failure duration and failure inter-arrival time. Vary parameters of distributions to have simulated faultloads
IMPLEMENTATION

• WAN Prototype Details
  – Prototype is written in Java based on RMI.
  – Run availability experiments using the bulletin board service.
  – To bound NE, each replica ensures that the error bound on other replicas is not violated. [Yu, Vahdat VLDB 2000]
  – To bound OE, the work implements 3 such protocols
    • Primary copy [Petersen SOSP 97]
    • Golding's algorithm [Golding Computing Systems 92 ]
    • Voting [Gifford SOSP 79]
  – Primary copy protocol
    • A write is committed when it reaches the primary replica.
    • Serialization order is the write order seen by the primary replica.
    • Replica that needs to reduce order error commits writes by first pushing its tentative writes to the primary and then pulling any other unseen updates from the primary.

– Golding's algorithm
  • Each write is assigned a logical timestamp that determines the serialization order
  • Each replica maintains a version vector to determine whether or not it has seen all writes with logical time less than \( t \).
  • To reduce OE, a replica pulls writes from other replicas to advance its version vector

– Voting
  • Voting protocol conducts a series of elections to determine a serialization order
  • During a round, each replica casts a vote for the first uncommitted write.
  • The write with the most votes wins and is committed next (in serialization order) across all replicas.
  • To reduce OE with voting, a replica first pushes writes to remote sites.
  • These sites then cast their votes and the results are pulled in subsequent sessions to allow write commitment.
IMPLEMENTATION

• Emulation Environment and Verification
  – Major evaluation done using a local area emulation environment.
  – Emulation accuracy is verified through live wide-area deployment.
  – To validate emulation results, the prototype system running the replicated bulletin board service is deployed on the 7 sites they use.
  – Run two separate 24 hour experiments at two different target consistency levels using Golding’s algorithm to bound order error.
  – In the first experiment, NE = 6 (recall that there are 7 replicas total) and leave OE unbounded,

• Table 2 summarizes the accuracy of the emulation

<table>
<thead>
<tr>
<th>Consistency</th>
<th># Writes (WAN)</th>
<th># Rejected (WAN)</th>
<th>Avail. (WAN)</th>
<th># Writes (emulation)</th>
<th># Rejected (emulation)</th>
<th>Avail. (emulation)</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>NE=6, OE=∞</td>
<td>120,703</td>
<td>1,099</td>
<td>98.6%</td>
<td>120,703</td>
<td>1,762</td>
<td>98.5%</td>
<td>96.3%</td>
</tr>
<tr>
<td>NE=∞, OE=1</td>
<td>60,489</td>
<td>293</td>
<td>99.6%</td>
<td>60,439</td>
<td>298</td>
<td>99.5%</td>
<td>98.3%</td>
</tr>
</tbody>
</table>

Table 2: Wide-area deployment and emulation verification results.
Results

- Data points are from repeated runs of the TACT software while varying:
  - NE, OE, Consistency protocols & Fault Loads.
  - Workload is a uniform update rate of one write per 10 seconds on each replica, resulting in 0.8 writes per second system-wide for the eight emulated replicas.

- For initial set of results, $\text{Avail}_{\text{service}}$ is used as the availability metric
  - Assume that replicas accept all reads and reject the writes that would violate global consistency requirements.
  - Service availability is re-defined to be the percentage of writes that are accepted by the replicas.

![Figure 2: Availability as function of numerical error (SIM1.00).](image1.png)

![Figure 3: Availability as function of order error.](image2.png)

![Figure 4: Availability as function of numerical error (SAMPLED1).](image3.png)

![Figure 5: Availability as function of order error (SAMPLED1).](image4.png)
Availability/Communication Tradeoffs

![Graph](image)

**Figure 6:** Availability as function of numerical error (\text{SIM5.00}).

![Graph](image)

**Figure 7:** Availability as function of order error (\text{SIM5.00}).

![Graph](image)

**Figure 8:** Availability with different average failure rate.
Availability/Communication Tradeoffs

Figure 9: Number of messages to maintain numerical error.

Availability/Communication Tradeoffs

Figure 10: Number of messages to maintain order error.
Availability/Communication Tradeoffs

- An interesting result is that achieving maximum $\text{Avail}_{\text{service}}$ with a relaxed consistency model can entail increased communication overhead.

- The communication costs of maintaining consistency can be reduced by waiting as long as possible to propagate writes.

- Results show that maximizing availability requires aggressive write propagation.
SUMMARY

• Evaluation shows that simple optimizations to existing consistency protocols can greatly improve the availability of replicated services.

• Staying as close to strong consistency as possible during times of good connectivity allows services to approach the tight upper bounds on availability derived.

• Voting and primary copy generally achieve the best availability.

• Additional replicas will not always improve service availability and can in fact reduce it.

» THANK YOU