1 Introduction

Today, Dr. Soma continued a lecture of Quorum Systems. Topics that would be covered were Load and Availability, Resilience and Failure Probability, Optimal Resilience Quorum Systems, Classical Quorum-Based Emulations, Static Quorum Membership, Asymmetric Read/Write Quorum Systems (AQS), and Generalized ABD Algorithm for Client Side.

2 Load and Availability

When we talked about the Quorum Systems. One thing we should consider is the term load and availability (fault tolerant), that is, a protocol using a quorum system needs to access at least one quorum. With this situation, the best case that can occur is the case that a process accessing a quorum will access all nodes in some quorum \( Q \). For the term ‘load’, it is measuring the minimal access probability of the most busy node. Load is reverse variation to quality of the system.

Given strategy \( \sigma \), load on node \( s_i \) is the probability that \( s_i \) will belong to a quorum selected according to \( \sigma \).

The load of \( \sigma \) on a quorum system is defined on maximum load on any individual node.

The load on quorum system is the minimum load of all strategies.

**Definition 1** Load, let \( Q \) be quorum system over \( S \), then,

1. Load induced by \( \sigma \) on \( s_i \in S \)
   
   \[ l_\sigma(s_i, QS) = \sum_{Q \ni s_i} \sigma_{QS}(Q) \]

2. Load induced by \( \sigma \) on \( QS \)
   
   \[ L_\sigma(QS) = \max_{s_i \in S} l_\sigma(s_i, QS) \]

3. Load of \( QS \)
   
   \[ L(QS) = \min_\sigma L_\sigma(QS) \]
3 Resilience and Failure Probability

Definition 2 The resilience $R(QS)$ of a quorum system $QS$ is the maximal integer $t$ such that despite the failure of any $t$ processes, there is a quorum $Q \in QS$ such that no process $p \in Q$ fails. For instance, if there is a singleton quorum, then, resilience is equal to 0.

We noted that if $R(QS) < m(QS)$, clearly, the failure of all nodes in any single quorum implies at least one node failure in every quorum. Thus, the resilience is bounded by the minimum quorum cardinality $m(QS)$.

4 Optimal Resilient Quorum Systems

No quorum system can have resiliency $> \lfloor \frac{n-1}{2} \rfloor$. That is, it is exactly resilience of majorities coteries.

Definition 3 Quorum Systems $QS$ is $t$-resilient if and only if $R(QS) \geq t$.

Definition 4 Quorum System $QS$ is resilient to a set system $\mathcal{F}$ if and only if $\forall F \in \mathcal{F}, \exists Q \in QS$ such that $Q \cap F = \emptyset$.

5 Failure Probability

Global Failure Probability is defined assuming independent process failures in uniform probabilistic failure model.

$F_p(QS)$ is probability that no quorum in $QS$ will have all non-faulthy nodes, where $p$ is probability of failure of each node in $S$.

Definition 5  

$$F_p(QS) = \prod_{Q \in QS} F_p(Q)$$

$F_p(Q)$ is probability that some node in $Q$ fails.

<table>
<thead>
<tr>
<th>QS</th>
<th>$\mathcal{L}(QS)$</th>
<th>$R(QS)$</th>
<th>$F_p(QS)$</th>
<th>$F_p(QS)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>singleton</td>
<td>1</td>
<td>0</td>
<td>$p$ ($p &gt; \frac{1}{2}$)</td>
<td></td>
</tr>
<tr>
<td>majority</td>
<td>$\lfloor \frac{n+1}{2} \rfloor$</td>
<td>$\lfloor \frac{n+1}{2} \rfloor$</td>
<td>$e^{-\Omega(n)}$</td>
<td></td>
</tr>
<tr>
<td>grid</td>
<td>$O(\frac{1}{\sqrt{n}})$</td>
<td>$O(\sqrt{n})$</td>
<td>$1^k$</td>
<td></td>
</tr>
<tr>
<td>B-grid</td>
<td>$O(\frac{1}{\sqrt{n}})$</td>
<td>$O(\sqrt{n})$</td>
<td>$O(e^{-\frac{n}{2}})$</td>
<td></td>
</tr>
</tbody>
</table>

The left table represents the value of $\mathcal{L}(QS)$, $R(QS)$, and $F_p(QS)$ in different types of Quorum System, which consist of singleton, majority, grid, and B-grid.
If $p < \frac{1}{2}$, majority has the best availability but poor load ($QS$ are considered to have good failure probability if $F_p(QS)$ tends to 0 for large $n$, assuming $p < \frac{1}{2}$).

6 Classical Quorum-Based Emulations

Definition 6 A storage (or consensus) emulation is based on a quorum system if the emulation satisfies:

1. Safety property in all executions
2. Liveness Property in all executions in which at least one quorum $Q \in QS$ is available.

7 Static Quorum Membership

ABD atomic storage emulation (Attiya, Bar-Noy, and Dolev) has a quorum-based variant in which we would consider.

There are two distinct finite set of processors, the set of client $C$ and a set of servers $S$. Each client has unique id $c \in \mathbb{N}$, $s_1, s_2, s_3, ..., s_n$ are servers. Any client or server may fail by crashing.

8 Asymmetric Read/Write Quorum System $AQS$

We defined $RQ \subseteq AQS$, $WQ \subseteq AQS$. $ABD$ is based on $AQS$. It is required to satisfy atomicity in all executions. Wait-freedom is achieved as long as there is at least one read quorum and at least one write quorum with correct server, i.e.,

$$\exists Q_R \in RQ, \exists Q_W \in WQ.$$  

$$\forall t, Q_R \subseteq Alive(S), Q_W \subseteq Alive(S)$$

Both read and write proceed in two phases.

1. In the first phase of write (Timestamp synchronized phase), a client (writer) contacts a read quorum in order to determine the highest timestamp $maxts$ used prior to current write.

2. After incrementing timestamp, $maxts$ locally, it propagates the written value along with timestamp and its own id to a write quorum.

9 Generalized ABD Algorithm for Client
Algorithm 1 Generalized ABD (Client)

Require: \( \text{opCnt} \leftarrow 0, \text{maxts} \leftarrow 0, \text{maxcid} \leftarrow 0, \text{maxvalue} \leftarrow \bot \)

\[ \text{inv}_c(\text{ABD}, \text{write}(v)) \]
\[ \text{optCnt} \leftarrow \text{opCnt} + 1 \]
\[ \langle \text{phase1} \rangle \text{ (timestamp synchronized phase)} \]
\[ \text{send}(\text{read}, \text{opCnt}) \text{ to all servers} \]
\[ \text{WHEN receive}(\text{readAck}, \text{val}_i, <\text{ts}_i, \text{cid}_i>, \text{opCnt}) \text{ from all } s_i \text{ from some } Q \in \text{RQ} \]
\[ \text{maxts} \leftarrow \text{maximum } \text{ts}_i \text{ received} \]
\[ \text{END WHEN} \]
\[ \langle \text{phase2} \rangle \text{ (write phase)} \]
\[ \text{send}(\text{write}, v, <\text{maxts} + 1, c>, \text{opCnt}) \text{ to all server } S \]
\[ \text{WHEN receive}(\text{writeAck}, \text{opCnt}) \text{ from all } s_i \text{ from some } Q \in \text{WQ} \]
\[ \text{return resp}_c(\text{ABD}, \text{ack}) \]
\[ \text{END WHEN} \]
\[ \text{end} \]

\[ \text{inv}_v(\text{ABD}, \text{read}()) \]
\[ \text{opCnt} \leftarrow \text{opCnt} + 1 \]
\[ \langle \text{phase1} \rangle \]
\[ \text{send}(\text{read}, \text{opCnt}) \text{ to all servers} \]
\[ \text{WHEN receive}(\text{readAck}, \text{val}_i, <\text{ts}_i, \text{cid}_i>, \text{opCnt}) \text{ from all } s_i \text{ in some } Q \in \text{RQ} \]
\[ \text{maxts} \leftarrow \text{maximum } \text{ts}_i \text{ in received messages} \]
\[ \text{maxcid} \leftarrow \text{maximum } \text{cid}_i \text{ in received messages with maxts} \]
\[ \text{maxval} \leftarrow \text{val}_i \text{ in received message with } <\text{maxts}, \text{maxcid}> \]
\[ \text{END WHEN} \]
\[ \langle \text{phase2} \rangle \]
\[ \text{send}(\text{write}, \text{maxval}, <\text{maxts}, \text{maxcid}>, \text{opCnt}) \]
\[ \text{WHEN receive}(\text{writeAck}, \text{opCnt}) \text{ from all } s_i \text{ in some } Q \in \text{WQ} \]
\[ \text{return resp}_v(\text{ABD}, \text{maxval}) \]
\[ \text{END WHEN} \]
\[ \text{end} \]