Viewstamped Replication

10/19/18
MIDTERM

Next Wednesday 10/24 at 7 - 9pm in CS 104

Covers all material up to and including Monday’s lecture
Viewstamped Replication

A way to implement replicated state machines

Goal: strong consistency across replicas

Similar to Paxos and RAFT, but less popular
Viewstamped Replication
Normal operation
\[2f + 1 = 3 \text{ nodes}\]

*Can tolerate \( f = 1 \) node failing at once*
Client 136

Request

op: x = 18
cid: 136
request num: 0

A

status: normal
replica: 0
view: 0
op: 0
commit: -1

B

status: normal
replica: 1
view: 0
op: 0
commit: -1

C

status: normal
replica: 2
view: 0
op: 0
commit: -1
A
status: normal
replica: 0
view: 0
op: 1
commit: -1

<0, 1> x = 18

B
status: normal
replica: 1
view: 0
op: 0
commit: -1

<empty>

C
status: normal
replica: 2
view: 0
op: 0
commit: -1

<empty>

Prepare
view: 0
op: 1
commit: -1
<Request>

<view, op>
Primary only needs to wait for $f = 1$ replies before committing.
Client 136

Reply
view: 0
request num: 0
result: x = 18

A

status  normal
replica 0
view 0
op 1
commit 1

<0, 1> x = 18
✓ committed

B

status  normal
replica 1
view 0
op 1
commit -1

<0, 1> x = 18

C

status  normal
replica 2
view 0
op 1
commit -1

<0, 1> x = 18

<view, op>
Primary informs backups that op 1 is committed during the next Prepare.
<table>
<thead>
<tr>
<th>Status</th>
<th>Replica</th>
<th>View</th>
<th>Op</th>
<th>Commit</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>normal</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>B</td>
<td>normal</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>C</td>
<td>normal</td>
<td>2</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
A

<table>
<thead>
<tr>
<th>status</th>
<th>replica</th>
<th>view</th>
<th>op</th>
<th>commit</th>
</tr>
</thead>
<tbody>
<tr>
<td>normal</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

<0, 1> x = 18
<0, 2> x += 3

B

<table>
<thead>
<tr>
<th>status</th>
<th>replica</th>
<th>view</th>
<th>op</th>
<th>commit</th>
</tr>
</thead>
<tbody>
<tr>
<td>normal</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>-1</td>
</tr>
</tbody>
</table>

<0, 1> x = 18

Prepare

view: 0
op: 2
commit: 1

<Request>

C

<table>
<thead>
<tr>
<th>status</th>
<th>replica</th>
<th>view</th>
<th>op</th>
<th>commit</th>
</tr>
</thead>
<tbody>
<tr>
<td>normal</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>-1</td>
</tr>
</tbody>
</table>

<0, 1> x = 18

<view, op> committed
A status normal
replica 0
view 0
op 2
commit 1

<0, 1> x = 18 ✓
<0, 2> x += 3

B
status normal
replica 1
view 0
op 2
commit 1

<0, 1> x = 18 ✓
<0, 2> x += 3

C
status normal
replica 2
view 0
op 2
commit 1

<0, 1> x = 18 ✓
<0, 2> x += 3

PrepareOK
view: 0
op: 2
replica: 1

PrepareOK
view: 0
op: 2
replica: 2

committed
Client 136

**Reply**
- view: 0
- request num: 1
- result: x = 21

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th></th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>status</td>
<td>normal</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>replica</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>view</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>op</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>commit</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

A: \(<0, 1> x = 18\) \(\checkmark\)
B: \(<0, 1> x = 18\) \(\checkmark\)
C: \(<0, 1> x = 18\) \(\checkmark\)

\(<0, 2> x += 3\) \(\checkmark\) committed

\(<view, op>\)
What if the next Prepare never comes?

Primary times out and sends a Commit message to each backup

<table>
<thead>
<tr>
<th></th>
<th>status</th>
<th>replica</th>
<th>view</th>
<th>op</th>
<th>commit</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>normal</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td></td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>C</td>
<td></td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

Commit

- view: 0
- commit: 2

---

<view, op>

committed

<0, 1> x = 18
<0, 2> x += 3
A status replica view op commit
normal 0 0 2 2
<0, 1> x = 18 ✓
<0, 2> x += 3 ✓
<view, op> committed

B status replica view op commit
normal 1 0 2 2
<0, 1> x = 18 ✓
<0, 2> x += 3 ✓

C status replica view op commit
normal 2 0 2 2
<0, 1> x = 18 ✓
<0, 2> x += 3 ✓
Why is waiting for $f$ nodes enough?

Op is guaranteed to have been executed on $f + 1$ nodes (majority)
Overlapping quorums

Write quorum contains $f + 1$ nodes
Overlapping quorums

Write quorum contains $f + 1$ nodes

$\text{FAILED}$
Overlapping quorums

Write quorum contains $f + 1$ nodes

$X = 1$

FAILED

$X = ?$

Client
Overlapping quorums

Write quorum contains \( f + 1 \) nodes

Read quorum contains \( f + 1 \) nodes

Client

\( x = 1 \)

\( x = ? \)
Overlapping quorums

Write quorum contains \( f + 1 \) nodes

Read quorum contains \( f + 1 \) nodes

\( x = 1 \)

\( x = ? \)
Overlapping quorums

Write quorum contains $f + 1$ nodes

Read quorum contains $f + 1$ nodes
Non-overlapping quorums?

The diagram shows three components labeled A, B, and C. Component A is connected to component B, which is labeled "FAILED." Component C is connected to component B. The client is connected to component C with a question mark "x = ?" Uhhh...
Viewstamped replication
View change
Client 25

Request
op: y = 100
cid: 25
request num: 0

A
status | normal
--- | ---
replica | 0
view | 0
op | 2
commit | 2

<x, y>
x = 18
<x, y> x += 3

(view, op)
committed

B
status | normal
--- | ---
replica | 1
view | 0
op | 2
commit | 2

<x, y>
x = 18
<x, y> x += 3

C
status | normal
--- | ---
replica | 2
view | 0
op | 2
commit | 2

<x, y>
x = 18
<x, y> x += 3

Client 25

Request
op: y = 100
cid: 25
request num: 0

A
status | normal
--- | ---
replica | 0
view | 0
op | 2
commit | 2

<x, y>
x = 18
<x, y> x += 3

(view, op)
committed

B
status | normal
--- | ---
replica | 1
view | 0
op | 2
commit | 2

<x, y>
x = 18
<x, y> x += 3

C
status | normal
--- | ---
replica | 2
view | 0
op | 2
commit | 2

<x, y>
x = 18
<x, y> x += 3
Primary fails before sending Prepare to B
Logs are out of sync

**B**

- status: 1
- replica: 1
- view: 0
- op: 2
- commit: 2

Operations:
- \(<0, 1> \quad x = 18\)
- \(<0, 2> \quad x += 3\)

**C**

- status: 2
- replica: 2
- view: 0
- op: 3
- commit: 2

Operations:
- \(<0, 1> \quad x = 18\)
- \(<0, 2> \quad x += 3\)
- \(<0, 3> \quad y = 100\)

The diagram illustrates a situation where logs are out of sync, indicated by the marked operations and the status of the replicas and views.
C times out on hearing from the primary and starts view change

B

<table>
<thead>
<tr>
<th>status</th>
<th>replica</th>
<th>view</th>
<th>op</th>
<th>commit</th>
</tr>
</thead>
<tbody>
<tr>
<td>status</td>
<td>normal</td>
<td>1</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>replica</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>view</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>op</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>commit</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

C

<table>
<thead>
<tr>
<th>status</th>
<th>normal</th>
<th>2</th>
<th>0</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>replica</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>view</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>op</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>commit</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<view, op> committed

<0, 1> x = 18
<0, 2> x += 3
<0, 3> y = 100
Who is the new primary?

Go through the list of sorted IP addresses and find the next one (i.e. B)
Start view change:

Status = change
Increment local view
Send SVC to all nodes

---

B

<table>
<thead>
<tr>
<th>status</th>
<th>normal</th>
<th>view</th>
<th>op</th>
<th>commit</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

<0, 1> x = 18
<0, 2> x += 3

C

<table>
<thead>
<tr>
<th>status</th>
<th>normal</th>
<th>view</th>
<th>op</th>
<th>commit</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>2</td>
<td>3</td>
<td>2</td>
</tr>
</tbody>
</table>

<0, 1> x = 18
<0, 2> x += 3
<0, 3> y = 100

---

<view, op>
✓ committed
Start view change:

Status = change
Increment local view
Send SVC to all nodes
Receive SVC where:

SVC.view > local view {
    Status = view change
    Advance local view
    Send SVC to other nodes
}

\[
\begin{align*}
\text{status} & \quad \text{normal} \\
\text{replica} & \quad 1 \\
\text{view} & \quad 0 \\
\text{op} & \quad 2 \\
\text{commit} & \quad 2
\end{align*}
\]

\[
\begin{align*}
\text{B} & \quad \text{<0, 1> x = 18} \\
& \quad \text{<0, 2> x += 3}
\end{align*}
\]

StartViewChange

view: 1
replica: 2

\[
\begin{align*}
\text{status} & \quad \text{change} \\
\text{replica} & \quad 2 \\
\text{view} & \quad 1 \\
\text{op} & \quad 3 \\
\text{commit} & \quad 2
\end{align*}
\]

\[
\begin{align*}
\text{C} & \quad \text{<0, 1> x = 18} \\
& \quad \text{<0, 2> x += 3} \\
& \quad \text{<0, 3> y = 100}
\end{align*}
\]

<view, op>

✓ committed
Receive SVC where:

SVC.view > local view {
    Status = view change
    Advance local view
    Send SVC to other nodes
}

StartViewChange
view: 1
replica: 1

C
status change
replica 2
view 1
op 3
commit 2

<view, op>
✓ committed
Receive $f$ SVCs where:

\[
\text{SVC.view} == \text{local view} \{
  \text{Send DVC to new primary}
\}
\]
Receive \( f \) SVCs where:

\[
\text{SVC.view == local view} \Rightarrow \text{Send DVC to new primary}
\]
Logs are no longer out of sync!

With more nodes, we may receive multiple different logs

Pick the one with highest view and op number

<table>
<thead>
<tr>
<th>B</th>
<th>status</th>
<th>change</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>replica</td>
<td>1</td>
<td>&lt;0, 1&gt; x = 18</td>
</tr>
<tr>
<td></td>
<td>view</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>op</td>
<td>3</td>
<td>&lt;0, 2&gt; x += 3</td>
</tr>
<tr>
<td></td>
<td>commit</td>
<td>2</td>
<td>&lt;0, 3&gt; y = 100</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>C</th>
<th>status</th>
<th>change</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>replica</td>
<td>2</td>
<td>&lt;0, 1&gt; x = 18</td>
</tr>
<tr>
<td></td>
<td>view</td>
<td>1</td>
<td>&lt;0, 2&gt; x += 3</td>
</tr>
<tr>
<td></td>
<td>op</td>
<td>3</td>
<td>&lt;0, 3&gt; y = 100</td>
</tr>
<tr>
<td></td>
<td>commit</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>
Receive f DVCs:

Become new primary
Send StartView to others

Why do we send the log here?

StartView

view: 1
replica: 1
op: 3
commit: 2
<log>

B

status  | normal:
replica  | 1
view     | 1
op       | 3
commit   | 2

C

status  | change:
replica  | 2
view     | 1
op       | 3
commit   | 2

<view, op>

Why do we send the log here?
Notice <0, 3> is uncommitted and from an old view...

Do we commit it?

B

status  | normal
replica | 1
view    | 1
op      | 3
commit  | 2

<0, 1> x = 18
<0, 2> x += 3
<0, 3> y = 100

PrepareOK
view: 0
op: 3
replica: 2

C

status  | normal
replica | 2
view    | 1
op      | 3
commit  | 2

<0, 1> x = 18
<0, 2> x += 3
<0, 3> y = 100
Are uncommitted ops like <0, 3> guaranteed to survive into the new view?

What about committed ops? (e.g. <0, 1> and <0, 2>)

B

<table>
<thead>
<tr>
<th>status</th>
<th>normal</th>
<th>replica</th>
<th>view</th>
<th>op</th>
<th>commit</th>
</tr>
</thead>
<tbody>
<tr>
<td>x = 18</td>
<td></td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>x += 3</td>
<td></td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

C

<table>
<thead>
<tr>
<th>status</th>
<th>normal</th>
<th>replica</th>
<th>view</th>
<th>op</th>
<th>commit</th>
</tr>
</thead>
<tbody>
<tr>
<td>x = 18</td>
<td></td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>x += 3</td>
<td></td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>y = 100</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Summary: view change in VR

New primary is pre-selected based on IP address (round-robin)

View change triggered by timeout, could be any node

Wait for $f$ SVC that matches our view number before sending DVC

Wait for $f$ DVC to start new view (primary)

- Why $f$ in both cases?

- Provided that at most $f$ servers fail, is liveness guaranteed?
Additional reading for viewstamped replication


https://blog.acolyer.org/2015/03/06/viewstamped-replication-revisited/