Viewstamped replication

10/20/17
A note on assignment 2

Your tests need to pass deterministically

Use SyncMap, err on using too many (correctness > performance here)

You don’t need maps of maps (bad design in general)

Due tonight!!
MIDTERM

Next Friday 10/27 at 10am or 11am, you choose (90 minutes)

Covers all material up to and including today’s class
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 nodes

Can tolerate f = 1 node failing at once
### Client 136

**Request**
- **op:** $x = 18$
- **cid:** 136
- **request num:** 0

<table>
<thead>
<tr>
<th>Status</th>
<th>Replica</th>
<th>View</th>
<th>Op</th>
<th>Commit</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>normal</td>
<td>0</td>
<td>0</td>
<td>-1</td>
</tr>
</tbody>
</table>

<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>0</td>
<td>-1</td>
</tr>
</tbody>
</table>

<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>0</td>
<td>-1</td>
</tr>
</tbody>
</table>

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

Prepare
view: 0
op: 1
commit: -1

A:

B:

C:

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

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

<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>1</td>
<td>1</td>
</tr>
</tbody>
</table>

A

B

C

\(<0, 1> x = 18 \)
Primary informs backups that op 1 is committed during the next Prepare.

---

**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></td>
<td>normal</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

\[<0, 1> \times 18 \checkmark\]

---

**Table B**

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

\[<0, 1> \times 18\]

---

**Table C**

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

\[<0, 1> \times 18\]
Client 136

Request
op: x += 3
cid: 136
request num: 1

A

status normal
replica 0
view 0
op 1
commit 1

B

status normal
replica 1
view 0
op 1
commit -1

C

status normal
replica 2
view 0
op 1
commit -1

<view, op> committed

<0, 1> x = 18

<0, 1> x = 18

<0, 1> x = 18
A

status  replica  view  op  commit
normal  0       0     2   1

<x, y>
<x, y> x = 18

B

status  replica  view  op  commit
normal  1       0     1  -1

<<view, op>
committed

C

status  replica  view  op  commit
normal  2       0     1  -1

<x, 1> x = 18
A status: normal
  replica: 0
  view: 0
  op: 2
  commit: 1

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

B PrepareOK
  view: 0
  op: 2
  replica: 1

B status: normal
  replica: 1
  view: 0
  op: 2
  commit: 1

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

C PrepareOK
  view: 0
  op: 2
  replica: 2

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

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

*committed*
### Client 136

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

### Status and Replica Table

<table>
<thead>
<tr>
<th>A</th>
<th>status</th>
<th>normal</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>replica</td>
<td>view</td>
</tr>
<tr>
<td></td>
<td></td>
<td>op</td>
</tr>
<tr>
<td></td>
<td></td>
<td>commit</td>
</tr>
<tr>
<td>---</td>
<td>--------</td>
<td>--------</td>
</tr>
<tr>
<td>&lt;0, 1&gt;</td>
<td>x = 18</td>
<td>✓</td>
</tr>
<tr>
<td>&lt;0, 2&gt;</td>
<td>x += 3</td>
<td>✓</td>
</tr>
</tbody>
</table>

### View and Commit Data

- **<view, op> committed**
- Client 136
  - view: 0
  - request num: 1
  - result: x = 21

### Replicas and Commit Data

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

<table>
<thead>
<tr>
<th>C</th>
<th>status</th>
<th>normal</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>replica</td>
<td>view</td>
</tr>
<tr>
<td></td>
<td></td>
<td>op</td>
</tr>
<tr>
<td></td>
<td></td>
<td>commit</td>
</tr>
<tr>
<td>---</td>
<td>--------</td>
<td>--------</td>
</tr>
<tr>
<td>&lt;0, 1&gt;</td>
<td>x = 18</td>
<td>✓</td>
</tr>
<tr>
<td>&lt;0, 2&gt;</td>
<td>x += 3</td>
<td>✓</td>
</tr>
</tbody>
</table>
What if the next Prepare never comes?

Primary times out and sends a Commit message to each backup

<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>0</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>normal</td>
<td>1</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>normal</td>
<td>2</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Commit view: 0
commit: 2
</view, op>
committed

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

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

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

<view, op>
<table>
<thead>
<tr>
<th>Replica</th>
<th>View</th>
<th>OP</th>
<th>Commit</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>status</td>
<td>normal</td>
<td></td>
<td></td>
</tr>
<tr>
<td>replica</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>view</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>op</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>commit</td>
<td>2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

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

\[
\text{<view, op> committed}
\]

<table>
<thead>
<tr>
<th>Replica</th>
<th>View</th>
<th>OP</th>
<th>Commit</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>status</td>
<td>normal</td>
<td></td>
<td></td>
</tr>
<tr>
<td>replica</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>view</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>op</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>commit</td>
<td>2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

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

<table>
<thead>
<tr>
<th>Replica</th>
<th>View</th>
<th>OP</th>
<th>Commit</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>status</td>
<td>normal</td>
<td></td>
<td></td>
</tr>
<tr>
<td>replica</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>view</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>op</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>commit</td>
<td>2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

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

\[
\text{<view, op> committed}
\]
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

- $x = 1$
- Failed
Overlapping quorums

Write quorum contains $f + 1$ nodes

Client

$\text{x} = 1$

FAILLED

$\text{x} = ?$
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

\[ x = 1 \]

\[ x = ? \]
Overlapping quorums

Write quorum contains \( f + 1 \) nodes

Read quorum contains \( f + 1 \) nodes
Non-overlapping quorums?

A

x = 1

FAILED

B

C

X = ?

Uhhh...

Client
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

B
status normal
replica 1
view 0
op 2
commit 2

C
status normal
replica 2
view 0
op 2
commit 2

Client 25

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

A
status normal
replica 0
view 0
op 2
commit 2

B
status normal
replica 1
view 0
op 2
commit 2

C
status normal
replica 2
view 0
op 2
commit 2

<view, op> committed
Primary fails before sending Prepare to B

---

**A**

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

### Commit

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

---

**B**

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

### Normal

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

---

**C**

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

### Normal

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

---

**Prepare**

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

**Request**

Primary fails before sending Prepare to B
Logs are out of sync

<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>B</td>
<td>normal</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;0, 1&gt; x = 18 ✓</td>
<td></td>
<td>&lt;0, 2&gt; x += 3 ✓</td>
<td></td>
</tr>
</tbody>
</table>

<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>C</td>
<td>normal</td>
<td>2</td>
<td>0</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;0, 1&gt; x = 18 ✓</td>
<td></td>
<td>&lt;0, 2&gt; x += 3 ✓</td>
<td></td>
</tr>
</tbody>
</table>
C times out on hearing from the primary and starts view change

<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>B</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>x = 18</td>
<td>x += 3</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

C times out on hearing from the primary and starts view change

<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>C</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>x = 18</td>
<td>x += 3</td>
<td>y = 100</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(view, op) committed
Who is the new primary?
Go through the list of sorted IP addresses and find the next one (i.e. B)

B  status   normal
replica  1
view     0
op       2
commit   2

C  status   normal
replica  2
view     0
op       3
commit   2

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
Start view change:

Status = change
Increment local view
Send SVC to all nodes

StartViewChange
view: 1
replica: 2

C
status change
replica 2
view 1
op 3
commit 2

B
status replica view op commit
normal 1 0 2 2

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

<view, op> committed

C
status change
replica 2
view 1
op 3
commit 2

<0, 1> x = 18
<0, 2> x += 3
<0, 3> y = 100
Receive SVC where:

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

StartViewChange
view: 1
replica: 2

Receive SVC where:

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

\[
\text{SVC.view} > \text{local view} \{
\text{Status} = \text{view change}
\text{Advance local view}
\text{Send SVC to other nodes}
\}
\]
Receive f SVCs where:

\[
\text{SVC.view} == \text{local view} \begin{cases} 
\text{Send DVC to new primary} 
\end{cases}
\]
Receive $f$ SVCs where:

$$\text{SVC.view} == \text{local view} \{ \quad \text{Send DVC to new primary} \quad \}$$

---

**DoViewChange**
- **replica:** 2
- **view:** 1
- **op:** 3
- **commit:** 2

**<log>**

---

**B**
- **status:**
  - replica: 1
  - view: 1
  - op: 2
  - commit: 2
- **change**
  - $<0, 1> x = 18$
  - $<0, 2> x += 3$

---

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

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

- Become new primary
- Send StartView to others

Why do we send the log here?

---

**Status**

- **A**
  - View: 1
  - Replica: 1
  - Op: 2
  - Commit: 2
  - Log:
    - $<0, 1> \ x = 18$
    - $<0, 2> \ x += 3$
    - $<0, 3> \ y = 100$

- **B**
  - Status: Normal
    - Replica: 1
    - View: 1
    - Op: 3
    - Commit: 2

- **C**
  - Status: Change
    - Replica: 2
    - View: 1
    - Op: 3
    - Commit: 2
    - Log:
      - $<0, 1> \ x = 18$
      - $<0, 2> \ x += 3$
      - $<0, 3> \ y = 100$

**StartView**

- View: 1
- Replica: 1
- Op: 3
- Commit: 2
- Log:
  - $<0, 1> \ x = 18$
  - $<0, 2> \ x += 3$
  - $<0, 3> \ y = 100$
Notice $<0, 3>$ is uncommitted and from an old view...

Do we commit it?
Are uncommitted ops like <0, 3> guaranteed to survive into the new view?

What about committed ops? (e.g. <0, 1> and <0, 2>)
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?
Failure detection
Two kinds of failures

Server failures

Network partitions

These two are indistinguishable from a single machine!
Failure detection goals

**Completeness:** Each failure is detected

**Accuracy:** There is no mistaken detection

**Speed:** Time to first detection of a failure

**Scale:** Equal load on each node

… in terms of CPU and network bandwidth
Completeness, accuracy, speed, load?
Centralized detection

Gossip detection

If we’re running the view change protocol, what happens in each case?
What is gossip detection good for?

Certainly not viewstamped replication!

May cause *liveness* issues; primary cannot reach $f$ nodes

Dynamo uses gossip for membership and failure detection

More suitable for completely decentralized environments
Additional reading for viewstamped replication


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