6.033 Spring 2018

Lecture #16

- **Atomicity via Write-ahead logging**

goal: build reliable systems from unreliable components

the abstraction that makes that easier is

transactions, which provide **atomicity** and **isolation**, while not hindering **performance**

atomicity $\longrightarrow$ **shadow copies** (simple, poor performance)

isolation $\longrightarrow$ **?**

eventually, we also want transaction-based systems to be **distributed**: to run across multiple machines

goal: build reliable systems from unreliable components

the abstraction that makes that easier is

transactions, which provide **atomicity** and **isolation**, while not hindering **performance**

$$\begin{tikzcd}\textbf{atomicity} \arrow[r] & \text{shadow copies (simple, poor performance) or logs (better performance, a bit more complex)}\end{tikzcd}$$

$$\begin{tikzcd}\textbf{isolation} \arrow[r] & \textbf{?}\end{tikzcd}$$

eventually, we also want transaction-based systems to be **distributed**: to run across multiple machines

```
transfer(bankfile, account_a, account_b, amount):
  bank = read_accounts(bankfile)
  bank[account_a] = bank[account_a] – amount
  bank[account_b] = bank[account_b] + amount
  write_accounts(tmp_bankfile)
  rename(tmp_bankfile, bankfile)
```

using shadow copies to abort on error

```
transfer(bankfile, account_a, account_b, amount):
  bank = read_accounts(bankfile)
  bank[account_a] = bank[account_a] – amount
  bank[account_b] = bank[account_b] + amount
  if bank[account_a] < 0:
    print "Not enough funds"
  else:
    write_accounts("tmp_bankfile")
    rename(tmp_bankfile, bankfile)
```

with transaction syntax

```
transfer(account_a, account_b, amount):
  begin
  write(account_a, read(account_a) - amount)
  write(account_b, read(account_b) + amount)
  if read(account_a) < 0: // not enough funds
    abort
  else:
    commit
```

```
begin     // T1
A = 100
B = 50
commit    // A=100; B=50

begin     // T2
A = A-20
B = B+20
commit    // A=80; B=70

begin     // T3
A = A+30
crash! 💥
```

problem: after crash, A=110, but T3 never committed

we need a way to revert to A's previous committed value

```
    +--------+--------+--------+--------+--------+--------+--------+
TID |  T1    |  T1    |  T1    |  T2    |  T2    |  T2    |  T3    |
    | UPDATE | UPDATE | COMMIT | UPDATE | UPDATE | COMMIT | UPDATE |
OLD | A=0    | B=0    |        | A=100  | B=50   |        | A=80   |
NEW | A=100  | B=50   |        | A=80   | B=70   |        | A=110  |
    +--------+--------+--------+--------+--------+--------+--------+
```

```
begin     // T1
A = 100
B = 50
commit    // A=100; B=50

begin     // T2
A = A-20
B = B+20
commit    // A=80; B=70

begin     // T3
A = A+30
```

```
    +--------+--------+--------+--------+--------+--------+--------+
TID |  T1    |  T1    |   T1   |  T2    |  T2    |   T2   |  T3    |
    | UPDATE | UPDATE | COMMIT | UPDATE | UPDATE | COMMIT | UPDATE |
OLD | A=0    | B=0    |        | A=100  | B=50   |        | A=80   |
NEW | A=100  | B=50   |        | A=80   | B=70   |        | A=110  |
    +--------+--------+--------+--------+--------+--------+--------+
```

```
read(log, var):
  commits = {}
  // scan backwards
  for record r in log[len(log) - 1] .. log[0]:
    // keep track of commits
    if r.type == commit:
      commits.add(r.tid)
    // find var's last committed value
    if r.type == update and
       r.tid in commits and
       r.var == var:
        return r.new_value
    +--------+--------+--------+
TID |   T1   |   T1   |   T1   |
    | UPDATE | UPDATE | COMMIT |
OLD | A=0    | B=0    |        |
NEW | A=100  | B=50   |        |
    +--------+--------+--------+
```

```
begin      // T2
A = A-20
```

```
read(log, var):
  commits = {}
  // scan backwards
  for record r in log[len(log) - 1] .. log[0]:
    // keep track of commits
    if r.type == commit:
      commits.add(r.tid)
    // find var's last committed value
    if r.type == update and
       r.tid in commits and
       r.var == var:
        return r.new_value
```

```
commits = {}
```

```
    +---------+---------+---------+
TID |   T1    |   T1    |   T1    |
    | UPDATE  | UPDATE  | COMMIT  |
OLD | A=0     | B=0     |         |
NEW | A=100   | B=50    |         |
    +---------+---------+---------+
```

```
begin    // T2
A = A-20
```

```
commits = {T1}
```

```
read(log, var):
  commits = {}
  // scan backwards
  for record r in log[len(log) - 1] .. log[0]:
    // keep track of commits
    if r.type == commit:
      commits.add(r.tid)
    // find var's last committed value
    if r.type == update and
       r.tid in commits and
       r.var == var:
        return r.new_value
    +--------+--------+--------+--------+
TID |   T1   |   T1   |   T1   |   T2   |
    | UPDATE | UPDATE | COMMIT | UPDATE |
OLD | A=0    | B=0    |        | A=100  |
NEW | A=100  | B=50   |        | A=80   |
    +--------+--------+--------+--------+
```

```
begin     // T2
A = A-20
```

```
read(log, var):
  commits = {}
  // scan backwards
  for record r in log[len(log) - 1] .. log[0]:
    // keep track of commits
    if r.type == commit:
      commits.add(r.tid)
    // find var's last committed value
    if r.type == update and
       r.tid in commits and
       r.var == var:
        return r.new_value
    +--------+--------+--------+--------+
TID |   T1   |   T1   |   T1   |   T2   |
    | UPDATE | UPDATE | COMMIT | UPDATE |
OLD | A=0    | B=0    |        | A=100  |
NEW | A=100  | B=50   |        | A=80   |
    +--------+--------+--------+--------+
```

```
begin     // T2
A = A-20
A = A-30
```

```
read(log, var):
  commits = {}
  // scan backwards
  for record r in log[len(log) - 1] .. log[0]:
    // keep track of commits
    if r.type == commit:
      commits.add(r.tid)
    // find var's last committed value
    if r.type == update and
       r.tid in commits and
       r.var == var:
        return r.new_value
    +--------+--------+--------+--------+
TID |   T1   |   T1   |   T1   |   T2   |
    | UPDATE | UPDATE | COMMIT | UPDATE |
OLD | A=0    | B=0    |        | A=100  |
NEW | A=100  | B=50   |        | A=80   |
    +--------+--------+--------+--------+
```

```
begin     // T2
A = A-20
A = A-30
```

```
read(log, var):
  commits = {}
  // scan backwards
  for record r in log[len(log) - 1] .. log[0]:
    // keep track of commits
    if r.type == commit:
      commits.add(r.tid)
    // find var's last committed value
    if r.type == update and
       (r.tid in commits or r.tid == current_tid) and
       r.var == var:
        return r.new_value
```

TID	T1	T1	T1	T2	T2	T2	T3
	UPDATE	UPDATE	COMMIT	UPDATE	UPDATE	COMMIT	UPDATE
OLD	A=0	B=0		A=100	B=50		A=80
NEW	A=100	B=50		A=80	B=70		A=110

```
begin     // T1
A = 100
B = 50
commit

begin     // T2
A = A-20
B = B+20
commit

begin     // T3
A = A+30
crash! 💥
```

after a crash, the log is still correct; uncommitted updates will not be read

```
    +--------+--------+--------+--------+--------+--------+--------+
TID |   T1   |   T1   |   T1   |   T2   |   T2   |   T2   |   T3   |
    | UPDATE | UPDATE | COMMIT | UPDATE | UPDATE | COMMIT | UPDATE |
OLD | A=0    | B=0    |        | A=100  | B=50   |        | A=80   |
NEW | A=100  | B=50   |        | A=80   | B=70   |        | A=110  |
    +--------+--------+--------+--------+--------+--------+--------+
```

performance?

problem: reads are slow

```
    +--------+--------+--------+--------+--------+--------+--------+
TID |  T1    |  T1    |   T1   |  T2    |  T2    |   T2   |  T3    |
    | UPDATE | UPDATE | COMMIT | UPDATE | UPDATE | COMMIT | UPDATE |
OLD | A=0    | B=0    |        | A=100  | B=50   |        | A=80   |
NEW | A=100  | B=50   |        | A=80   | B=70   |        | A=110  |
    +--------+--------+--------+--------+--------+--------+--------+
```

cell storage (on disk)

A	110
B	70

```
read(var):
  return cell_read(var)


write(var, value):
  log.append(current_tid, update, var, read(var), value)
  cell_write(var, value)
```

```
    +--------+--------+--------+--------+--------+--------+--------+
TID |  T1    |  T1    |  T1    |  T2    |  T2    |  T2    |  T3    |
    | UPDATE | UPDATE | COMMIT | UPDATE | UPDATE | COMMIT | UPDATE |
OLD | A=0    | B=0    |        | A=100  | B=50   |        | A=80   |
NEW | A=100  | B=50   |        | A=80   | B=70   |        | A=110  |
    +--------+--------+--------+--------+--------+--------+--------+
```

cell storage (on disk)

A	110

B	70

```
recover(log):
  commits = {}
  for record r in log[len(log)-1] .. log[0]:
    if r.type == commit:
      commits.add(r.tid)
    if r.type == update and r.tid not in commits:
      cell_write(r.var, r.old_val) // undo
```

TID	T1	T1	T1	T2	T2	T2	T3
	UPDATE	UPDATE	COMMIT	UPDATE	UPDATE	COMMIT	UPDATE
OLD	A=0	B=0		A=100	B=50		A=80
NEW	A=100	B=50		A=80	B=70		A=110

cell storage (on disk)

A	110

B	70

commits = {}

```
recover(log):
  commits = {}
  for record r in log[len(log)-1] .. log[0]:
    if r.type == commit:
      commits.add(r.tid)
    if r.type == update and r.tid not in commits:
      cell_write(r.var, r.old_val) // undo
    +--------+--------+--------+--------+--------+--------+--------+
TID |   T1   |   T1   |   T1   |   T2   |   T2   |   T2   |   T3   |
    | UPDATE | UPDATE | COMMIT | UPDATE | UPDATE | COMMIT | UPDATE |
OLD | A=0    | B=0    |        | A=100  | B=50   |        | A=80   |
NEW | A=100  | B=50   |        | A=80   | B=70   |        | A=110  |
    +--------+--------+--------+--------+--------+--------+--------+
```

cell storage (on disk) $\boxed{\text{A 110}}$ $\boxed{\text{B 70}}$

commits = {}

```
recover(log):
  commits = {}
  for record r in log[len(log)-1] .. log[0]:
    if r.type == commit:
      commits.add(r.tid)
    if r.type == update and r.tid not in commits:
      cell_write(r.var, r.old_val) // undo
    +--------+--------+--------+--------+--------+--------+--------+
TID |  T1    |  T1    |   T1   |  T2    |  T2    |   T2   |  T3    |
    | UPDATE | UPDATE | COMMIT | UPDATE | UPDATE | COMMIT | UPDATE |
OLD | A=0    | B=0    |        | A=100  | B=50   |        | A=80   |
NEW | A=100  | B=50   |        | A=80   | B=70   |        | A=110  |
    +--------+--------+--------+--------+--------+--------+--------+
```

cell storage (on disk)

A	80
B	70

commits = {}

```
recover(log):
  commits = {}
  for record r in log[len(log)-1] .. log[0]:
    if r.type == commit:
      commits.add(r.tid)
    if r.type == update and r.tid not in commits:
      cell_write(r.var, r.old_val) // undo
    +--------+--------+--------+--------+--------+--------+--------+
TID |  T1    |  T1    |  T1    |  T2    |  T2    |  T2    |  T3    |
    | UPDATE | UPDATE | COMMIT | UPDATE | UPDATE | COMMIT | UPDATE |
OLD | A=0    | B=0    |        | A=100  | B=50   |        | A=80   |
NEW | A=100  | B=50   |        | A=80   | B=70   |        | A=110  |
    +--------+--------+--------+--------+--------+--------+--------+
```

cell storage (on disk) $\boxed{\texttt{A 80}}$ $\boxed{\texttt{B 70}}$

commits = {}

```
recover(log):
  commits = {}
  for record r in log[len(log)-1] .. log[0]:
    if r.type == commit:
      commits.add(r.tid)
    if r.type == update and r.tid not in commits:
      cell_write(r.var, r.old_val) // undo
```

TID	T1	T1	T1	T2	T2	T2	T3
	UPDATE	UPDATE	COMMIT	UPDATE	UPDATE	COMMIT	UPDATE
OLD	A=0	B=0		A=100	B=50		A=80
NEW	A=100	B=50		A=80	B=70		A=110

cell storage (on disk) A 80 B 70

commits = {T2}

```
recover(log):
  commits = {}
  for record r in log[len(log)-1] .. log[0]:
    if r.type == commit:
      commits.add(r.tid)
    if r.type == update and r.tid not in commits:
      cell_write(r.var, r.old_val) // undo
    +--------+--------+--------+--------+--------+--------+--------+
TID |   T1   |   T1   |   T1   |   T2   |   T2   |   T2   |   T3   |
    | UPDATE | UPDATE | COMMIT | UPDATE | UPDATE | COMMIT | UPDATE |
OLD | A=0    | B=0    |        | A=100  | B=50   |        | A=80   |
NEW | A=100  | B=50   |        | A=80   | B=70   |        | A=110  |
    +--------+--------+--------+--------+--------+--------+--------+
```

cell storage (on disk)

A	80
B	70

commits = {T2}

```
recover(log):
  commits = {}
  for record r in log[len(log)-1] .. log[0]:
    if r.type == commit:
      commits.add(r.tid)
    if r.type == update and r.tid not in commits:
      cell_write(r.var, r.old_val) // undo
```

TID	T1	T1	T1	T2	T2	T2	T3
	UPDATE	UPDATE	COMMIT	UPDATE	UPDATE	COMMIT	UPDATE
OLD	A=0	B=0		A=100	B=50		A=80
NEW	A=100	B=50		A=80	B=70		A=110

cell storage (on disk)

A 80	B 70

commits = {T2}

```
recover(log):
  commits = {}
  for record r in log[len(log)-1] .. log[0]:
    if r.type == commit:
      commits.add(r.tid)
    if r.type == update and r.tid not in commits:
      cell_write(r.var, r.old_val) // undo
    +--------+--------+--------+--------+--------+--------+--------+
TID |   T1   |   T1   |   T1   |   T2   |   T2   |   T2   |   T3   |
    | UPDATE | UPDATE | COMMIT | UPDATE | UPDATE | COMMIT | UPDATE |
OLD | A=0    | B=0    |        | A=100  | B=50   |        | A=80   |
NEW | A=100  | B=50   |        | A=80   | B=70   |        | A=110  |
    +--------+--------+--------+--------+--------+--------+--------+
```

cell storage (on disk) `A 80` `B 70`

commits = {T2}

```
recover(log):
  commits = {}
  for record r in log[len(log)-1] .. log[0]:
    if r.type == commit:
      commits.add(r.tid)
    if r.type == update and r.tid not in commits:
      cell_write(r.var, r.old_val) // undo
    +--------+--------+--------+--------+--------+--------+--------+
TID |  T1    |  T1    |  T1    |  T2    |  T2    |  T2    |  T3    |
    | UPDATE | UPDATE | COMMIT | UPDATE | UPDATE | COMMIT | UPDATE |
OLD | A=0    | B=0    |        | A=100  | B=50   |        | A=80   |
NEW | A=100  | B=50   |        | A=80   | B=70   |        | A=110  |
    +--------+--------+--------+--------+--------+--------+--------+
```

cell storage (on disk)   A 80   B 70

commits = {T2, T1}

```
recover(log):
  commits = {}
  for record r in log[len(log)-1] .. log[0]:
    if r.type == commit:
      commits.add(r.tid)
    if r.type == update and r.tid not in commits:
      cell_write(r.var, r.old_val) // undo
```

TID	T1	T1	T1	T2	T2	T2	T3
	UPDATE	UPDATE	COMMIT	UPDATE	UPDATE	COMMIT	UPDATE
OLD	A=0	B=0		A=100	B=50		A=80
NEW	A=100	B=50		A=80	B=70		A=110

cell storage (on disk)

A 80	B 70

commits = {T2, T1}

```
recover(log):
  commits = {}
  for record r in log[len(log)-1] .. log[0]:
    if r.type == commit:
      commits.add(r.tid)
    if r.type == update and r.tid not in commits:
      cell_write(r.var, r.old_val) // undo
```

TID	T1	T1	T1	T2	T2	T2	T3
	UPDATE	UPDATE	COMMIT	UPDATE	UPDATE	COMMIT	UPDATE
OLD	A=0	B=0		A=100	B=50		A=80
NEW	A=100	B=50		A=80	B=70		A=110

cell storage (on disk)

A 80	B 70

commits = {T2, T1}

```
recover(log):
  commits = {}
  for record r in log[len(log)-1] .. log[0]:
    if r.type == commit:
      commits.add(r.tid)
    if r.type == update and r.tid not in commits:
      cell_write(r.var, r.old_val) // undo
    +--------+--------+--------+--------+--------+--------+--------+
TID |  T1    |  T1    |   T1   |  T2    |  T2    |   T2   |  T3    |
    | UPDATE | UPDATE | COMMIT | UPDATE | UPDATE | COMMIT | UPDATE |
OLD | A=0    | B=0    |        | A=100  | B=50   |        | A=80   |
NEW | A=100  | B=50   |        | A=80   | B=70   |        | A=110  |
    +--------+--------+--------+--------+--------+--------+--------+
```

cell storage (on disk) `A 80` `B 70`

```
read(var):
  return cell_read(var)


write(var, value):
  log.append(current_tid, update, var, read(var), value)
  cell_write(var, value)
```

```
    +--------+--------+--------+--------+--------+--------+--------+
TID |   T1   |   T1   |   T1   |   T2   |   T2   |   T2   |   T3   |
    | UPDATE | UPDATE | COMMIT | UPDATE | UPDATE | COMMIT | UPDATE |
OLD | A=0    | B=0    |        | A=100  | B=50   |        | A=80   |
NEW | A=100  | B=50   |        | A=80   | B=70   |        | A=110  |
    +--------+--------+--------+--------+--------+--------+--------+
```

```
cell storage    +-------+   +-------+
 (on disk)      | A 110 |   | B  70 |
                +-------+   +-------+
```

performance?

problem: read performance is now great, but writes got (a little bit) slower and recovery got (a lot) slower

```
     +--------+--------+--------+--------+--------+--------+--------+
 TID |   T1   |   T1   |   T1   |   T2   |   T2   |   T2   |   T3   |
     | UPDATE | UPDATE | COMMIT | UPDATE | UPDATE | COMMIT | UPDATE |
 OLD | A=0    | B=0    |        | A=100  | B=50   |        | A=80   |
 NEW | A=100  | B=50   |        | A=80   | B=70   |        | A=110  |
     +--------+--------+--------+--------+--------+--------+--------+
```

cell storage (on disk) $\boxed{\texttt{A 110}}$ $\boxed{\texttt{B 70}}$ cache (memory) A 110 B 70

```
read(var):
  if var in cache:
    return cache[var]
  else:
    // may evict others from cache to cell storage
    cache[var] = cell_read(var)
    return cache[var]

write(var, value):
  log.append(current_tid, update, var, read(var), value)
  cache[var] = value

flush(): // called "occasionally"
  cell_write(var, cache[var]) for each var
```

```
    +--------+--------+--------+--------+--------+--------+--------+
TID |   T1   |   T1   |   T1   |   T2   |   T2   |   T2   |   T3   |
    | UPDATE | UPDATE | COMMIT | UPDATE | UPDATE | COMMIT | UPDATE |
OLD | A=0    | B=0    |        | A=100  | B=50   |        | A=80   |
NEW | A=100  | B=50   |        | A=80   | B=70   |        | A=110  |
    +--------+--------+--------+--------+--------+--------+--------+
```

```
cell storage   +-------+  +-------+     cache   .......  .......
 (on disk)     | A 100 |  | B  50 |  (memory)   :A 110:  :B  70:
               +-------+  +-------+             .......  .......
```

suppose we flushed the cache after T1 committed,
but have not flushed it since then

```
    +--------+--------+--------+--------+--------+--------+--------+
TID |   T1   |   T1   |   T1   |   T2   |   T2   |   T2   |   T3   |
    | UPDATE | UPDATE | COMMIT | UPDATE | UPDATE | COMMIT | UPDATE |
OLD | A=0    | B=0    |        | A=100  | B=50   |        | A=80   |
NEW | A=100  | B=50   |        | A=80   | B=70   |        | A=110  |
    +--------+--------+--------+--------+--------+--------+--------+
```

```
cell storage   +-------+  +-------+     cache    ..........  ..........
 (on disk)     | A 100 |  | B  50 |   (memory)   :        :  :        :
               +-------+  +-------+              ..........  ..........
```

```
recover(log):
  commits = {}
  for record r in log[len(log)-1] .. log[0]:
    if r.type == commit:
      commits.add(r.tid)
    if r.type == update and r.tid not in commits:
      cell_write(r.var, r.old_val) // undo
    +--------+--------+--------+--------+--------+--------+--------+
TID |   T1   |   T1   |   T1   |   T2   |   T2   |   T2   |   T3   |
    | UPDATE | UPDATE | COMMIT | UPDATE | UPDATE | COMMIT | UPDATE |
OLD | A=0    | B=0    |        | A=100  | B=50   |        | A=80   |
NEW | A=100  | B=50   |        | A=80   | B=70   |        | A=110  |
    +--------+--------+--------+--------+--------+--------+--------+
```

```
cell storage   | A 80 |   | B  50 |     cache     :........:  :........:
 (on disk)                              (memory)   :........:  :........:
```

```
recover(log):
  commits = {}
  for record r in log[len(log)-1] .. log[0]:
    if r.type == commit:
      commits.add(r.tid)
    if r.type == update and r.tid not in commits:
      cell_write(r.var, r.old_val) // undo
    +--------+--------+--------+--------+--------+--------+--------+
TID |   T1   |   T1   |   T1   |   T2   |   T2   |   T2   |   T3   |
    | UPDATE | UPDATE | COMMIT | UPDATE | UPDATE | COMMIT | UPDATE |
OLD | A=0    | B=0    |        | A=100  | B=50   |        | A=80   |
NEW | A=100  | B=50   |        | A=80   | B=70   |        | A=110  |
    +--------+--------+--------+--------+--------+--------+--------+
```

cell storage (on disk) A 80 B 50 **cache (memory)**

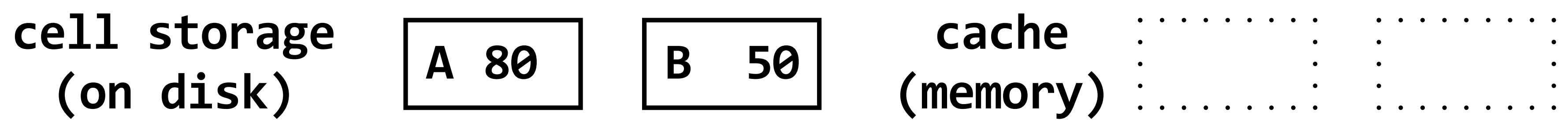
```
recover(log):
  commits = {}
  for record r in log[len(log)-1] .. log[0]:
    if r.type == commit:
      commits.add(r.tid)
    if r.type == update and r.tid not in commits:
      cell_write(r.var, r.old_val) // undo
    +--------+--------+--------+--------+--------+--------+--------+
TID |  T1    |  T1    |   T1   |  T2    |  T2    |   T2   |  T3    |
    | UPDATE | UPDATE | COMMIT | UPDATE | UPDATE | COMMIT | UPDATE |
OLD | A=0    | B=0    |        | A=100  | B=50   |        | A=80   |
NEW | A=100  | B=50   |        | A=80   | B=70   |        | A=110  |
    +--------+--------+--------+--------+--------+--------+--------+
```

```
cell storage    +------+  +------+    cache   ..........  ..........
 (on disk)      | A 80 |  | B 50 |  (memory)  :        :  :        :
                +------+  +------+            ..........  ..........
```

```
recover(log):
  commits = {}
  for record r in log[len(log)-1] .. log[0]:
    if r.type == commit:
      commits.add(r.tid)
    if r.type == update and r.tid not in commits:
      cell_write(r.var, r.old_val) // undo
```

all other updates were committed; **B**'s value won't ever be changed

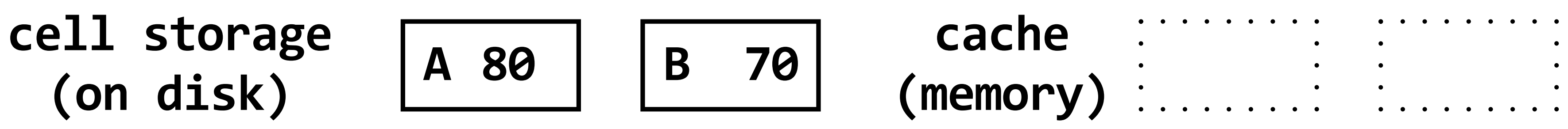
```
    +--------+--------+--------+--------+--------+--------+--------+
TID |   T1   |   T1   |   T1   |   T2   |   T2   |   T2   |   T3   |
    | UPDATE | UPDATE | COMMIT | UPDATE | UPDATE | COMMIT | UPDATE |
OLD | A=0    | B=0    |        | A=100  | B=50   |        | A=80   |
NEW | A=100  | B=50   |        | A=80   | B=70   |        | A=110  |
    +--------+--------+--------+--------+--------+--------+--------+
```



```
recover(log):
  commits = {}
  for record r in log[len(log)-1] .. log[0]:
    if r.type == commit:
      commits.add(r.tid)
    if r.type == update and r.tid not in commits:
      cell_write(r.var, r.old_val) // undo
  for record r in log[0] .. log[len(log)-1]:
    if r.type == update and r.tid in commits:
      cell_write(r.var, r.new_value) // redo
    +--------+--------+--------+--------+--------+--------+--------+
TID |  T1    |  T1    |   T1   |  T2    |  T2    |   T2   |  T3    |
    | UPDATE | UPDATE | COMMIT | UPDATE | UPDATE | COMMIT | UPDATE |
OLD | A=0    | B=0    |        | A=100  | B=50   |        | A=80   |
NEW | A=100  | B=50   |        | A=80   | B=70   |        | A=110  |
    +--------+--------+--------+--------+--------+--------+--------+
```

cell storage (on disk) **A 80** **B 70** **cache (memory)**

```
recover(log):
  commits = {}
  for record r in log[len(log)-1] .. log[0]:
    if r.type == commit:
      commits.add(r.tid)
    if r.type == update and r.tid not in commits:
      cell_write(r.var, r.old_val) // undo
  for record r in log[0] .. log[len(log)-1]:
    if r.type == update and r.tid in commits:
      cell_write(r.var, r.new_value) // redo
    +--------+--------+--------+--------+--------+--------+--------+
TID |   T1   |   T1   |   T1   |   T2   |   T2   |   T2   |   T3   |
    | UPDATE | UPDATE | COMMIT | UPDATE | UPDATE | COMMIT | UPDATE |
OLD | A=0    | B=0    |        | A=100  | B=50   |        | A=80   |
NEW | A=100  | B=50   |        | A=80   | B=70   |        | A=110  |
    +--------+--------+--------+--------+--------+--------+--------+
```



performance?

problem: recovery is still slow

```
     +--------+--------+--------+--------+--------+--------+--------+
TID  |  T1    |  T1    |   T1   |  T2    |  T2    |   T2   |  T3    |
     | UPDATE | UPDATE | COMMIT | UPDATE | UPDATE | COMMIT | UPDATE |
OLD  | A=0    | B=0    |        | A=100  | B=50   |        | A=80   |
NEW  | A=100  | B=50   |        | A=80   | B=70   |        | A=110  |
     +--------+--------+--------+--------+--------+--------+--------+
```

cell storage (on disk) $\boxed{\texttt{A 80}}$ $\boxed{\texttt{B 70}}$ cache (memory)

performance?

solution: write checkpoints and truncate the log

- **(Write-ahead) logs** provide **atomicity** with better performance than shadow copies. The primary benefit is making small appends for each update, rather than copying and entire file over for every change.

- **Cell storage** is used with the log to improve read-performance, and **caches** and **truncation** can be used to improve write- and recovery-performance.