6.033 Spring 2016
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** → **shadow copies** (simple, poor performance)

**isolation** → ?

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**

| atomicity | shadow copies (simple, poor performance) or **logs** (better performance, a bit more complex) |
| isolation | ? |

eventually, we also want transaction-based systems to be **distributed**: to run across multiple machines
using shadow copies to abort on error

```python
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)
```
using transactions to abort on error

```python
transfer(account_a, account_b, amount):
    begin
    write(a, read(a) - amount)
    write(b, read(b) + amount)
    if read(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

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

we need a way to revert to A’s previous committed value
<table>
<thead>
<tr>
<th>TID</th>
<th>T1</th>
<th>T1</th>
<th>T1</th>
<th>T2</th>
<th>T2</th>
<th>T2</th>
<th>T3</th>
</tr>
</thead>
<tbody>
<tr>
<td>OLD</td>
<td>A=0</td>
<td>B=0</td>
<td>A=100</td>
<td>B=50</td>
<td>A=80</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NEW</td>
<td>A=100</td>
<td>B=50</td>
<td>COMMIT</td>
<td></td>
<td>COMMIT</td>
<td></td>
<td>A=110</td>
</tr>
</tbody>
</table>

```
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
```
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
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
performance?

**problem:** reads are slow
### TID
<table>
<thead>
<tr>
<th>T1</th>
<th>T1</th>
<th>T1</th>
<th>T2</th>
<th>T2</th>
<th>T2</th>
<th>T3</th>
</tr>
</thead>
<tbody>
<tr>
<td>OLD</td>
<td>A=0</td>
<td>B=0</td>
<td>A=100</td>
<td>B=50</td>
<td>A=80</td>
<td>A=110</td>
</tr>
<tr>
<td>NEW</td>
<td>A=100</td>
<td>B=50</td>
<td>COMMIT</td>
<td>A=80</td>
<td>B=70</td>
<td>COMMIT</td>
</tr>
</tbody>
</table>

#### read(var):
return cell_read(var)

#### write(var, value):
log.append(current_tid, update, var, read(var), value)
cell_write(var, value)
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
cell storage

|   | 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
<table>
<thead>
<tr>
<th>TID</th>
<th>OLD</th>
<th>NEW</th>
<th>TID</th>
<th>OLD</th>
<th>NEW</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A=0</td>
<td>A=100</td>
<td>T1</td>
<td>B=0</td>
<td>B=50</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>COMMIT</td>
<td></td>
</tr>
<tr>
<td></td>
<td>A=100</td>
<td></td>
<td>T2</td>
<td>B=50</td>
<td>B=70</td>
</tr>
<tr>
<td></td>
<td></td>
<td>COMMIT</td>
<td>T2</td>
<td>COMMIT</td>
<td></td>
</tr>
<tr>
<td></td>
<td>A=80</td>
<td>A=110</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

cell storage  

```
recovers(log):
    commits = {}
    for record r in log[log.len()-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
```
### recover(log):

```python
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
```
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
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
<table>
<thead>
<tr>
<th>TID</th>
<th>OLD</th>
<th>NEW</th>
<th>cell storage</th>
<th>commits</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1</td>
<td>A=0</td>
<td>A=100</td>
<td>A 80</td>
<td>{}</td>
</tr>
<tr>
<td>T1</td>
<td>B=0</td>
<td>B=50</td>
<td>B 70</td>
<td></td>
</tr>
<tr>
<td>T1</td>
<td>COMMIT</td>
<td>COMMIT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T2</td>
<td>A=100</td>
<td>A=80</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T2</td>
<td>B=50</td>
<td>B=70</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T3</td>
<td>A=80</td>
<td>A=110</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

recover(log):

```python
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
```
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
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
<table>
<thead>
<tr>
<th>TID</th>
<th>OLD</th>
<th>NEW</th>
<th>commits</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1</td>
<td>A=0</td>
<td>A=100</td>
<td>T2</td>
</tr>
<tr>
<td>T1</td>
<td>B=0</td>
<td>B=50</td>
<td>T2</td>
</tr>
<tr>
<td>T1</td>
<td></td>
<td>COMMIT</td>
<td>T2</td>
</tr>
<tr>
<td>T2</td>
<td>A=100</td>
<td>A=80</td>
<td>T2 COMMIT</td>
</tr>
<tr>
<td>T2</td>
<td>B=50</td>
<td>B=70</td>
<td>T2 COMMIT</td>
</tr>
<tr>
<td>T2</td>
<td></td>
<td>A=110</td>
<td></td>
</tr>
</tbody>
</table>

recover(log):

```python
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
```

cell storage

A 80  B 70

commits = {T2}
cell storage

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
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

cell storage

\[
\begin{array}{c|c|c|c|c|c|c|c}
\hline
TID & T1 & T1 & T1 & T2 & T2 & T2 & T3 \\
\hline
OLD & A=0 & B=0 & & A=100 & B=50 & & & A=80 \\
NEW & A=100 & B=50 & COMMIT & A=80 & B=70 & COMMIT & A=110 \\
\hline
\end{array}
\]

\[
commits = \{T2\}\]
<table>
<thead>
<tr>
<th>TID</th>
<th>OLD</th>
<th>NEW</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A=0</td>
<td>A=100</td>
</tr>
<tr>
<td></td>
<td>B=0</td>
<td>B=50</td>
</tr>
<tr>
<td>T1</td>
<td>COMMIT</td>
<td>COMMIT</td>
</tr>
<tr>
<td>T1</td>
<td>A=100</td>
<td>A=80</td>
</tr>
<tr>
<td>T1</td>
<td>B=50</td>
<td>B=70</td>
</tr>
<tr>
<td>T2</td>
<td>COMMIT</td>
<td>COMMIT</td>
</tr>
<tr>
<td>T2</td>
<td>A=100</td>
<td>A=80</td>
</tr>
<tr>
<td>T2</td>
<td>B=50</td>
<td>B=70</td>
</tr>
<tr>
<td>T3</td>
<td>A=80</td>
<td>A=110</td>
</tr>
</tbody>
</table>

```
cell storage

A 80 B 70
```

```
commit = {T2}
```

```python
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
```
<table>
<thead>
<tr>
<th>TID</th>
<th>OLD</th>
<th>NEW</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A=0</td>
<td>A=100</td>
</tr>
<tr>
<td></td>
<td>B=0</td>
<td>B=50</td>
</tr>
<tr>
<td></td>
<td>T1</td>
<td>COMMIT</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T2</td>
<td>A=100</td>
<td>A=80</td>
</tr>
<tr>
<td></td>
<td>B=50</td>
<td>B=70</td>
</tr>
<tr>
<td></td>
<td>COMMIT</td>
<td>COMMIT</td>
</tr>
<tr>
<td>T3</td>
<td>A=80</td>
<td>A=110</td>
</tr>
</tbody>
</table>

```
cell storage
```

```
T2
```

```
commits = {T2}
```

```
recover(log):

commit logs = {}
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
```
recover(log):

```python
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
```

```
commits = {T2}
```

```
cell storage:

A  80  B  70
```

+-------+------+--------+-------+------+--------+-------+
| TID   | T1   | T1     | T1    | T2   | T2     | T2    |
| OLD   | A=0  | B=0    | A=100 | B=50 | A=80   | B=70  |
| NEW   | A=100| B=50   | COMMIT| A=100| B=80   | COMMIT|
+-------+------+--------+-------+------+--------+-------+
<table>
<thead>
<tr>
<th>TID</th>
<th>T1</th>
<th>T1</th>
<th>T1</th>
<th>T2</th>
<th>T2</th>
<th>T2</th>
<th>T3</th>
</tr>
</thead>
<tbody>
<tr>
<td>OLD</td>
<td>A=0</td>
<td>B=0</td>
<td>A=100</td>
<td>B=50</td>
<td>A=80</td>
<td>B=70</td>
<td>A=80</td>
</tr>
<tr>
<td>NEW</td>
<td>A=100</td>
<td>B=50</td>
<td>COMMIT</td>
<td>A=80</td>
<td>COMMIT</td>
<td>A=110</td>
<td></td>
</tr>
</tbody>
</table>

recover(log):

```python
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
```
<table>
<thead>
<tr>
<th>TID</th>
<th>T1</th>
<th>T1</th>
<th>T1</th>
<th>T2</th>
<th>T2</th>
<th>T2</th>
<th>T3</th>
</tr>
</thead>
<tbody>
<tr>
<td>OLD</td>
<td>A=0</td>
<td>B=0</td>
<td></td>
<td>A=100</td>
<td>B=50</td>
<td></td>
<td>A=80</td>
</tr>
<tr>
<td>NEW</td>
<td>A=100</td>
<td>B=50</td>
<td>COMMIT</td>
<td>A=80</td>
<td>B=70</td>
<td>COMMIT</td>
<td>A=110</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>cell storage</th>
<th>A</th>
<th>80</th>
<th>B</th>
<th>70</th>
</tr>
</thead>
</table>

commits = \{T2\}

recover(log):

```python
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
```
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
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
**recover(log):**

```python
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
```

**cell storage**

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>80</td>
<td>B</td>
<td>70</td>
</tr>
</tbody>
</table>

**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

    cell storage
    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
<table>
<thead>
<tr>
<th>TID</th>
<th>T1</th>
<th>T1</th>
<th>T1</th>
<th>T2</th>
<th>T2</th>
<th>T2</th>
<th>T3</th>
</tr>
</thead>
<tbody>
<tr>
<td>OLD</td>
<td>A=0</td>
<td>B=0</td>
<td>COMMIT</td>
<td>A=100</td>
<td>B=50</td>
<td>COMMIT</td>
<td>A=80</td>
</tr>
<tr>
<td>NEW</td>
<td>A=100</td>
<td>B=50</td>
<td>COMMIT</td>
<td>A=80</td>
<td>B=70</td>
<td>COMMIT</td>
<td>A=110</td>
</tr>
</tbody>
</table>

```
cell storage

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>80</td>
<td>70</td>
</tr>
</tbody>
</table>

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
```
cell storage | A 80 | B 70 |

commit = \{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
### cell storage

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>80</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>70</td>
<td></td>
</tr>
</tbody>
</table>

### commits

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
```
cell storage

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

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
## Recovering Transactions

The recovery process involves the following steps:

1. **Initialize** the `commits` set to an empty set.

2. **Process Log Records**:
   - For each record `r` in the log, starting from the last record to the first:
     - **Commit Records**:
       - If `r` is a commit record, add its transaction ID to the `commits` set.
     - **Update Records**:
       - If `r` is an update record and its transaction ID is not already in the `commits` set, perform the following:
         - Write the old value to the cell.
         - **Undo** changes.

### Example Log

<table>
<thead>
<tr>
<th>TID</th>
<th>OLD</th>
<th>NEW</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1</td>
<td>A=0</td>
<td>A=100</td>
</tr>
<tr>
<td></td>
<td>B=0</td>
<td>B=50</td>
</tr>
<tr>
<td></td>
<td></td>
<td>COMMIT</td>
</tr>
<tr>
<td>T2</td>
<td>A=100</td>
<td>A=80</td>
</tr>
<tr>
<td></td>
<td>B=50</td>
<td>B=70</td>
</tr>
<tr>
<td></td>
<td></td>
<td>COMMIT</td>
</tr>
<tr>
<td>T3</td>
<td>A=80</td>
<td>A=110</td>
</tr>
</tbody>
</table>

### Example Cell Storage

- **Cell Storage**:
  - A: 80
  - B: 70

### Recover Function

```python
def recover(log):
    commits = {}
    for record r in log[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
```

### Commits Set

- The set of transactions that were committed:
  - `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

cell storage

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
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
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
cell storage

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
<table>
<thead>
<tr>
<th>TID</th>
<th>OLD</th>
<th>NEW</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A=0</td>
<td>A=100</td>
</tr>
<tr>
<td></td>
<td>B=0</td>
<td>B=50</td>
</tr>
<tr>
<td>T1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T2</td>
<td>A=100</td>
<td>A=100</td>
</tr>
<tr>
<td>T2</td>
<td>B=50</td>
<td>B=70</td>
</tr>
<tr>
<td>T2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T3</td>
<td>A=80</td>
<td>A=110</td>
</tr>
</tbody>
</table>

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

commit storage: 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
```
cell storage  

A  110  B  70  cache  

A  110  B  70

**read(var):**

```python
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):**

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

```python
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
```
• **(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.