Having Your Cake and Eating it Too: Combining Strong and Eventual Consistency

Konrad Siek and Paweł T. Wojciechowski
Poznań University of Technology
{konrad.siek,pawel.t.wojciechowski}@cs.put.edu.pl

13 IV 2014

http://dsg.cs.put.poznan.pl
Software Transactional Memory

def thread:
    lock_a.acquire()
    lock_b.acquire()
    a = b
    lock_a.release()
    b = b + 1
    lock_b.release()
def thread:
    lock_a.acquire()
    lock_b.acquire()
    a = b
    lock_a.release()
    b = b + 1
    lock_b.release()

def thread:
    transaction.start()
    a = b
    b = b + 1
    transaction.commit()
def thread:
    lock_a.acquire()
    lock_b.acquire()
    a = b
    lock_a.release()
    b = b + 1
    lock_b.release()

def thread:
    transaction.start()
    a = b
    b = b + 1
    transaction.commit()

Advantages:

- ease of use on top
- efficient concurrency control under the hood
Transaction Abstraction

Transaction:
\[ T_i \left[ \, op_1, \, op_2, \, \ldots, \, op_n \, \right] \]
where \( op = \{ \, r(x)v, \, w(x)v, \, \ldots \, \} \)
and \( x \) is some shared object

Commitment:
\[ \{ x = 1 \} \quad T_i \left[ \, w(x)2 \, \right] \quad \{ x = 2 \} \]

Rollback:
\[ \{ x = 1 \} \quad T_i \left[ \, w(x)2, \, \right] \quad \{ x = 1 \} \]
\[ \{ x = 1 \} \quad T_i \left[ \, w(x)2, \, \right] \quad \rightarrow \quad T'_i \left[ \, w(x)2 \, \right] \quad \{ x = 2 \} \]
Distributed Transactional Memory

Replicated TM

T₁
Distributed Transactional Memory

Distributed Transactions
Distributed Transactional Memory

Distributed Transactions

T₁

x
y
z

T₂

a
b
c

Distributed Transactions
Supremum Versioning Algorithm

Pessimistic approach

\[ \{x = 1, y = 1\} \quad \begin{array}{c}
T_1 \quad \left[ \begin{array}{c}
\{x = 3, y = 2\}
\end{array}\right]
\end{array}
\]

\[ \begin{array}{c}
\left[ \begin{array}{c}
r(x)1, w(x)2
\end{array}\right]
\end{array}
\]

\[ \begin{array}{c}
\left[ \begin{array}{c}
r(x)2, w(x)3
\end{array}\right]
\end{array}
\]
Supremum Versioning Algorithm

Pessimistic approach

\[ \{ x = 1, y = 1 \} \quad T_1 \left[ r(x)_1, w(x)_2 \right] \]
\[ | \quad T_2 \left[ r(x)_2, w(x)_3 \right] \quad \{ x = 3, y = 2 \} \]

- Defer execution to prevent conflicts
Supremum Versioning Algorithm

Pessimistic approach

\[
\begin{align*}
\{ x = 1, y = 1 \} & \quad T_1 \left[ r(x)1, w(x)2 \right] \\
\mid T_2 & \left[ r(x)2, w(x)3 \right] \quad \{ x = 3, y = 2 \}
\end{align*}
\]

- Defer execution to prevent conflicts (tolerate high contention)
Pessimistic approach

\[
\begin{align*}
\{x = 1, y = 1\} & \quad T_1 \left[ r(x)1, w(x)2 \right] \\
| & \quad T_2 \left[ r(x)2, w(x)3 \right] \quad \{x = 3, y = 2\}
\end{align*}
\]

- Defer execution to prevent conflicts (tolerate high contention)
- Avoid (most) forced aborts
Supremum Versioning Algorithm

Pessimistic approach

\[
\begin{align*}
\{x = 1, y = 1\} & \quad T_1 \left[ r(x)1, w(x)2 \right] \\
\mid & \quad T_2 \left[ \right. \\
& \quad \left. r(x)2, w(x)3 \right] \quad \{x = 3, y = 2\}
\end{align*}
\]

- Defer execution to prevent conflicts (tolerate high contention)
- Avoid (most) forced aborts (safe irrevocable operations)
Supremum Versioning Algorithm

Pessimistic approach

\[
\{x = 1, y = 1\} \quad T_1 \quad [ \quad r(x)1, w(x)2 \quad ]
\]

\[
| T_2 \quad [ \quad r(x)2, w(x)3 \quad ] \quad \{x = 3, y = 2\}
\]

- Defer execution to prevent conflicts (tolerate high contention)
- Avoid (most) forced aborts (safe irrevocable operations)

Early release on last use

\[
\{x = 1, y = 1\} \quad T_1 \quad [ \quad r(x)1, w(x)2, r(y)1, w(y)2 \quad ]
\]

\[
| T_2 \quad [ \quad r(x)2, w(x)3 \quad ] \quad \{x = 3, y = 2\}
\]
Supremum Versioning Algorithm

Pessimistic approach

\[ \{x = 1, y = 1\} \quad T_1 \left[ r(x)1, w(x)2 \right] \]
\[ | T_2 \left[ \quad \rightarrow r(x)2, w(x)3 \right] \quad \{x = 3, y = 2\} \]

- Defer execution to prevent conflicts (tolerate high contention)
- Avoid (most) forced aborts (safe irrevocable operations)

Early release on last use

\[ \{x = 1, y = 1\} \quad T_1 \left[ r(x)1, w(x)2, r(y)1, w(y)2 \right] \]
\[ | T_2 \left[ \quad \rightarrow r(x)2, w(x)3 \right] \quad \{x = 3, y = 2\} \]
Supremum Versioning Algorithm

Pessimistic approach

\[ \{x = 1, y = 1\} \quad T_1 \quad [r(x)1, w(x)2] \]

\[ \mid T_2 \quad [\quad \quad \quad r(x)2, w(x)3\quad] \quad \{x = 3, y = 2\} \]

- Defer execution to prevent conflicts (tolerate high contention)
- Avoid (most) forced aborts (safe irrevocable operations)

Early release on last use

\[ \{x = 1, y = 1\} \quad T_1 \quad [r(x)1, w(x)2, r(y)1, w(y)2] \]

\[ \mid T_2 \quad [\quad \quad \quad r(x)2, w(x)3\quad] \quad \{x = 3, y = 2\} \]

Completely distributed (no leader, dispatcher, etc.)
Supremum Versioning Algorithm

Pessimistic approach

\[ \{x = 1, y = 1\} \quad T_1 \quad [ r(x)1, w(x)2 ] \]
\[ \quad | \quad T_2 \quad [ \quad r(x)2, w(x)3 ] \quad \{x = 3, y = 2\} \]

- Defer execution to prevent conflicts (tolerate high contention)
- Avoid (most) forced aborts (safe irrevocable operations)

Early release on last use

\[ \{x = 1, y = 1\} \quad T_1 \quad [ r(x)1, w(x)2, r(y)1, w(y)2 ] \]
\[ \quad | \quad T_2 \quad [ \quad r(x)2, w(x)3 ] \quad \{x = 3, y = 2\} \]

Completely distributed (no leader, dispatcher, etc.)

Strong consistency
def deposit:
    account.deposit(sum)
def deposit:
    account.deposit(sum)
def withdraw:
    account.withdraw(sum)
def deposit:
    account.deposit(sum)

def withdraw:
    account.withdraw(sum)

def balance:
    print account.getBalance()
def deposit:
    account.deposit(sum)

def withdraw:
    account.withdraw(sum)

def balance:
    print account.getBalance()

def transfer:
    account1.withdraw(sum)
    account2.deposit(sum)
def deposit:
    account.deposit(sum)

def withdraw:
    account.withdraw(sum)

def balance:
    print account.getBalance()

def transfer:
    account1.withdraw(sum)
    account2.deposit(sum)

def audit:
    for a in accounts:
        sum += a.getBalance()
    value = bank.getCapital()
    bank.setCapital(sum)
    print "Accumulated capital", sum - value
def deposit:
    transaction.start()
    account.deposit(sum)
    transaction.commit()

def withdraw:
    transaction.start()
    account.withdraw(sum)
    transaction.commit()

def balance:
    transaction.start()
    print account.getBalance()
    transaction.commit()

def transfer:
    transaction.start()
    account1.withdraw(sum)
    account2.deposit(sum)
    transaction.commit()

def audit:
    transaction.start()
    for a in accounts:
        sum += a.getBalance()
    value = bank.getCapital()
    bank.setCapital(sum)
    print "Accumulated capital", sum - value
    transaction.commit()
def deposit:
    transaction.start()
    account.deposit(sum)
    transaction.commit()

def withdraw:
    transaction.start()
    account.withdraw(sum)
    transaction.commit()

def balance:
    transaction.start()
    print account.getBalance()
    transaction.commit()

def transfer:
    transaction.start()
    account1.withdraw(sum)
    account2.deposit(sum)
    transaction.commit()

def audit:
    transaction.start()
    for a in accounts:
        sum += a.getBalance()
    value = bank.getCapital()
    bank.setCapital(sum)
    print "Accumulated capital", sum - value
    transaction.commit()

weaken consistency → improve efficiency
Eventually Consistent Extension

x
y
z

x
y
z

a
b
c

a
b
c

T₁

T₂
Eventually Consistent Extension

\[
\begin{align*}
\mathcal{T}_1 & \quad \mathcal{T}_2 \\
\begin{array}{c}
x \\
y \\
z \\
\end{array} & \quad \begin{array}{c}
x \\
y \\
z \\
\end{array} & \quad \begin{array}{c}
a \\
b \\
c \\
\end{array} & \quad \begin{array}{c}
a \\
b \\
c \\
\end{array} \\
\begin{array}{c}
x_1 \\
y_1 \\
\end{array} & \quad \begin{array}{c}
a_1 \\
b_1 \\
\end{array}
\end{align*}
\]
What we require from weak transactions

- do not wait for variables
What we require from weak transactions

- do not wait for variables
- do not block other transactions
What we require from weak transactions

- do not wait for variables
- do not block other transactions
- internal consistency
What we require from weak transactions

- do not wait for variables
- do not block other transactions
- internal consistency
- do not disturb consistent transactions
What we require from weak transactions

- do not wait for variables
- do not block other transactions
- internal consistency
- do not disturb consistent transactions
- converge
Transaction Modes

Transaction $T_1$

$T_1[ r(x)v_c, w(x)u_c ]$
Transaction Modes

Transaction $T_1$

$$T_1 \left[ r(x)v_c, w(x)u_c \right]$$

Consistent mode

$$T_1^c \left[ r(x)v_c, w(x)u_c \right]$$
Transaction Modes

Transaction $T_1$

\[ T_1 \left[ r(x)v_c, w(x)u_c \right] \]

**Consistent mode**

\[ T_1^c \left[ r(x)v_c, w(x)u_c \right] \]

**Eventually consistent mode**

\[ T_1^{ec} \left[ r(x)v_{ec}, w(x)u_{ec} \right] \]
Transaction Modes

Transaction $T_1$

\[ T_1 \langle r(x)v_c, w(x)u_c \rangle \]

Consistent mode

\[ T_1^c \langle r(x)v_c, w(x)u_c \rangle \]

Eventually consistent mode

\[ T_1^{ec} \langle r(x)v_{ec}, w(x)u_{ec} \rangle \]

Execute consistent and inconsistent modes simultaneously
What we require from weak transactions

- do not wait for variables
- do not block other transactions
- internal consistency
- do not disturb consistent transactions
- converge
What we require from weak transactions

- do not wait for variables
- do not block other transactions
- internal consistency
- do not disturb consistent transactions
- converge ✔️
Variable Modification Versions

\[ \{x = 1, y = 1\} \quad T_1 \left[ r(x)1, w(x)2, r(y)1, w(y)2 \right] \quad T_2 \left[ r(x)2, w(x)3 \right] \quad \{x = 3, y = 2\} \]
\[
\{ x = 1, y = 1 \} \quad T_1^{ec} \quad \left[ r^{0}(x), w^{1}(x), r^{0}(y), w^{1}(y) \right] \\
\left\downarrow \right. \\
\{ x = 3, y = 2 \}
\]
Variable Modification Versions

\[
\{ x = 1, y = 1 \} \quad T_1^{ec} \left[ r(x)^0, w(x)^1, r(y)^0, w(y)^1 \right] \\
| T_2 \left[ r(x)^1, w(x)^2, r(x)^1, w(x)^2 \right] \\
| T_3 \left[ r(x)^2, w(x)^3, r(x)^2, w(x)^3 \right] \\
| T_4 \left[ r(x)^3, w(x)^4, r(x)^3, w(x)^4 \right]
\]

Snapshot Read Consistency

\[
T_1 \left[ r(x)^0, w(x)^1, r(y)^0, w(y)^1, w(y)^2 \right] \\
| T_2 \left[ r(x)^1, w(x)^2 \right] \\
| T_3 \left[ r(x)^2, w(x)^3 \right] \\
| T_4 \left[ r(x)^3, w(x)^4 \right] \\
\]
Variable Modification Versions

\[ \{ x = 1, y = 1 \} \quad T_{1}^{ec} \left[ \begin{array}{c} r(x)_1, w(x)_2, r(y)_1, w(y)_2 \\ \end{array} \right] \]

\[ \quad \left| T_2 \right[ \quad \Rightarrow r(x)_2, w(x)_3 \right] \quad \{ x = 3, y = 2 \} \]

Snapshot Read Consitency

\[ \left[ \begin{array}{c} r(x)_1, w(x)_2, r(y)_1, w(y)_2, w(y)_3 \end{array} \right] \]

\[ \left| T_2 \right[ \quad \Rightarrow r(x)_2, w(x)_3 \right] \]

\[ \left| T_3 \right[ \quad \Rightarrow r(x)_3, w(x)_4, r(y)_3, w(y)_4 \right] \]

\[ \left| T_4 \right[ \quad \Rightarrow r(x)_4, w(x)_5 \right] \]

\[ \{ x, y \} \]
Variable Modification Versions

\[
\{x = 1, y = 1\} \quad T_1^{ec} \quad \left[ r_x^0, w_x^1, r_y^0, w_y^1 \right] \\
| \quad T_2 \quad \left[ r_x^1, w_x^2 \right] \\
\quad T_2 \left[ r_x^2, w_x^3 \right] \quad \{x = 3, y = 2\}
\]

Snapshot Read Consistency

\[
T_1 \quad \left[ r_x^0, w_x^1, r_y^0, w_y^1, w_y^2 \right] \\
| \quad T_2 \quad \left[ r_x^1, w_x^2 \right] \\
| \quad T_3 \quad \left[ r_x^3, w_x^4, r_y^3, w_y^4 \right] \\
| \quad T_4 \quad \left[ r_x^3, w_x^5 \right]
\]

\[
\{x, y\}, \{x, y\}
\]
Variable Modification Versions

\[
\{ x = 1, y = 1 \} \quad T_{1}^{ec} \quad [ \; r(0)x1, w(1)x2, r(0)y1, w(1)y2 \; ] \\
| \quad T_{2} \quad [ \quad \Rightarrow r(1)x2, w(2)x3 \quad ] \quad \{ x = 3, y = 2 \}
\]

Snapshot Read Consistency

\[
T_{1} \quad [ \; r(0)x1, w(1)x2, r(0)y1, w(1)y2, w(2)y3 \; ] \\
| \quad T_{2} \quad [ \quad \Rightarrow r(1)x2, w(2)x3 \quad ] \\
| \quad T_{3} \quad [ \quad \Rightarrow r(2)x3, w(3)x4, r(2)y3, w(3)y4 \quad ] \\
| \quad T_{4} \quad [ \quad \Rightarrow r(3)x4, w(4)x5 \quad ]
\]

\[
\{ 1, 2 \}, \{ 2, 2 \}, \{ 3, 3 \}, \{ 4, 3 \}
\]
Variable Modification Versions

\[
\{x = 1, y = 1\} \quad T_1^{ec} \quad [\quad r(x)1, w(x)2, r(y)1, w(y)2 \quad ]
\]

<table>
<thead>
<tr>
<th>\hspace{1cm}</th>
<th>\hspace{1cm}</th>
</tr>
</thead>
<tbody>
<tr>
<td>\hspace{1cm}</td>
<td>\hspace{1cm}</td>
</tr>
<tr>
<td>\hspace{1cm}</td>
<td>\hspace{1cm}</td>
</tr>
<tr>
<td>\hspace{1cm}</td>
<td>\hspace{1cm}</td>
</tr>
<tr>
<td>\hspace{1cm}</td>
<td>\hspace{1cm}</td>
</tr>
</tbody>
</table>

Snapshot Read Consistency

\[
T_1 \quad [\quad r(x)1, w(x)2, r(y)1, w(y)2, w(y)3 \quad ]
\]

<table>
<thead>
<tr>
<th>\hspace{1cm}</th>
<th>\hspace{1cm}</th>
</tr>
</thead>
<tbody>
<tr>
<td>\hspace{1cm}</td>
<td>\hspace{1cm}</td>
</tr>
<tr>
<td>\hspace{1cm}</td>
<td>\hspace{1cm}</td>
</tr>
<tr>
<td>\hspace{1cm}</td>
<td>\hspace{1cm}</td>
</tr>
<tr>
<td>\hspace{1cm}</td>
<td>\hspace{1cm}</td>
</tr>
</tbody>
</table>

\[
T_2 \quad [\quad r(x)2, \ w(x)3 \quad ]
\]

\[
T_3 \quad [\quad r(x)3, \ w(x)4, \ r(y)3, \ w(y)4 \quad ]
\]

\[
T_4 \quad [\quad r(x)4, \ w(x)5 \quad ]
\]

\[
\{1, 2\}, \ \{x, y\}, \ \{2, 2\}, \ \{3, 3\}, \ \{4, 3\}, \ \{3, 2\}, \ \{x, y\}
\]
Variable Modification Versions

\[
\{x = 1, y = 1\} \quad T^{ec}_1 \left[ \begin{array}{c} r(x)1, w(x)2, r(y)1, w(y)2 \\ r(x)2, w(x)3 \end{array} \right] \quad \{x = 3, y = 2\}
\]

Snapshot Read Consistency

\[
\begin{align*}
T_1 & \left[ \begin{array}{c} r(x)1, w(x)2, r(y)1, w(y)2, w(y)3 \\ r(x)2, w(x)3 \end{array} \right] \\
T_2 & \left[ \begin{array}{c} r(x)2, w(x)3 \end{array} \right] \\
T_3 & \left[ \begin{array}{c} r(x)3, w(x)4, r(y)3, w(y)4 \end{array} \right] \\
T_4 & \left[ \begin{array}{c} r(x)4, w(x)5 \end{array} \right]
\end{align*}
\]

\[
\{x, y\}, \{x, y\}, \{x, y\}, \{x, y\}, \{x, y\}, \{x, y\}
\]
What we require from weak transactions

- do not wait for variables
- do not block other transactions
- internal consistency
- do not disturb consistent transactions
- converge ✔
What we require from weak transactions

- do not wait for variables
- do not block other transactions
- internal consistency ✓
- do not disturb consistent transactions
- converge ✓
Transactions:

- record the latest committed version of variable
Consistent Snapshot in Practice

Transactions:
- record the latest committed version of variable
- record the latest released version of variable (early release)
Transactions:
- record the latest committed version of variable
- record the latest released version of variable (early release)
- when releasing a variable early: record variables that were not released early
What we require from weak transactions

- do not wait for variables
- do not block other transactions
- internal consistency ✓
- do not disturb consistent transactions
- converge ✓
What we require from weak transactions

- do not wait for variables ✓
- do not block other transactions ✓
- internal consistency ✓
- do not disturb consistent transactions
- converge ✓
Write Bufferring

\[
\{ x = 1, y = 1 \} \ T_1 \ \left[ r(x)_1, w(x)_2, r(y)_1, w(y)_2, w(y)_3 \right] \ \{ x = 2, y = 3 \}
\]
Write Bufferring

\[ \{x = 1, y = 1\} T_1 [ \{r(x)^0, w(x)^2, r(y)^1, w(y)^2, w(y)^3\} \{x = 1, y = 1\} \{x = 2, y = 3\} \]
Write Bufferring

\[
\{ x = 1, y = 1 \} \ T_1 \ [ \ r(x)1, w(x)2, r(y)1, w(y)2, w(y)3 \] \ \{ x = 1, y = 1 \}
\{ x = 2, y = 3 \}
\]

Consistent mode either:
- applies the buffered writes
Write Bufferring

\{ x = 1, y = 1 \} T_1 \left[ r(x)_1, w(x)_2, r(y)_1, w(y)_2, w(y)_3 \right] \{ x = 1, y = 1 \}
{ x = 2, y = 3 \}

Consistent mode either:
- applies the buffered writes (if consistency condition allows)
Write Bufferring

\[
\{x = 1, y = 1\} \ T_1 \ [ \ r(x)1, w(x)2, r(y)1, w(y)2, w(y)3 \] \ \{x = 1, y = 1\} \\
\{x = 2, y = 3\}
\]

Consistent mode either:

- applies the buffered writes (if consistency condition allows)
- re-executes from scratch
What we require from weak transactions

- do not wait for variables ✓
- do not block other transactions ✓
- internal consistency ✓
- do not disturb consistent transactions
- converge ✓
What we require from weak transactions

- do not wait for variables ✓
- do not block other transactions ✓
- internal consistency ✓
- do not disturb consistent transactions ✓
- converge ✓
Eventually Consistent SVA

\[
\begin{align*}
\{x = 1, y = 1\} & \quad T_1 \left[ r(x)^0, w(x)^1, r(y)^1, w(y)^2 \right] \\
\left| T_2^c \right[ & \quad \rightarrow r(x)^1, w(\overline{x})^2 \right] \quad \{\overline{x} = 3, \overline{y} = 2\} \\
\left| T_2^{ec} \right[ & \quad r(x)^0, w(x)^2 \right]
\end{align*}
\]
Eventually Consistent SVA

\[
\{ x = 1, y = 1 \} \quad T_1 \quad \left[ \begin{array}{c}
T_2^c \left[ \begin{array}{c}
T_2^{ec} \left[ \begin{array}{c}
T_3 \quad \left[ \begin{array}{c}
\end{array} \right] \\
\end{array} \right] \\
\end{array} \right] \\
\end{array} \right]
\]
Summary

- eventual consistency extension for pessimistic distributed TM
- minimal extra cost
- eventually consistent transactions read consistent snapshots
- strongly consistent transactions are unaffected
Related Papers:

