Nowe Algorytmy i Własności do Pesymistycznego Sterowania Współbieżnością w Rozproszonej Pamięci Transakcyjnej

Konrad Siek

Poznań University of Technology

konrad.siek@cs.put.edu.pl

13 I 2015

Distributed Systems Group

Pamięć Transakcyjna

```
a_lock.acquire()
b_lock.acquire()
a = a + b
a_lock.release()
b = b + 1
b_lock.release()
```

Pamięć Transakcyjna

```
a_lock.acquire()
b_lock.acquire()
a = a + b
a_lock.release()
b = b + 1
b_lock.release()
```

```
transaction {
    a = a + b
    b = b + 1
}
```

Herlihy, Moss. *Transactional Memory: Architectural Support for Lock-free Data Structures*. ISCA'93.

Optymistyczne Sterowanie Współbieżnością

```
transaction {
    x = x + 1
}
```

Optymistyczne Sterowanie Współbieżnością

```
transaction {
    x = x + 1
}
```

$\{x = 1\}$

Optymistyczne Sterowanie Współbieżnością

```
transaction {
    x = x + 1
}
```

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

Optymistyczne Sterowanie Współbieżnością

```
transaction {
    x = x + 1
}
```

$$\{x = 1\} \quad \begin{array}{ll} T_1 & [\![\ r(x)1, \quad w(x)2\ ]\!] \\ T_2 & [\![\ r(x)1, \end{array}$$

Optymistyczne Sterowanie Współbieżnością

```
transaction {
    x = x + 1
}
```

$$\{x = 1\} \quad T_1 \; [\![\; r(x)1, \quad w(x)2 \;]\!]$$
$$T_2 \quad [\![\; r(x)1, \quad w(x)2$$

Optymistyczne Sterowanie Współbieżnością

```
transaction {
    x = x + 1
}
```

$$\{x = 1\} \quad T_1 \; [\![\; r(x)1, \;\; w(x)2 \;]\!]$$
$$T_2 \quad [\![\; r(x)1, \;\; w(x)2 \;\; \hookleftarrow$$

Optymistyczne Sterowanie Współbieżnością

```
transaction {
    x = x + 1
}
```

$$\{x = 1\} \quad T_1 \; [\![\; r(x)1, \quad w(x)2 \;]\!]$$
$$T_2 \quad [\![\; r(x)1, \quad w(x)2 \; \circlearrowleft \; \rightarrow T_2' \; [\![\; r(x)2, w(x)3 \;]\!] \quad \{x = 3\}$$

Optymistyczne Sterowanie Współbieżnością

```
transaction {
    x = x + 1
}
```

$$\{x = 1\} \quad T_1 \; [\![\; r(x)1, \;\; w(x)2 \;]\!]$$
$$T_2 \quad [\![\; r(x)1, \;\; w(x)2 \;\; \hookleftarrow \; \rightarrow T_2' \; [\![\; r(x)2, w(x)3 \;]\!] \quad \{x = 3\}$$

Konflikt: transakcja czyta lub modyfikuje zmienną, która jest jednocześnie modyfikowana przez inną transakcje.

Poprawność Wykonania

Szeregowalność (*serializability*)

Papadimitrou. *The Serializability of Concurrent Database Updates*. Journal of the ACM, 1979.

Poprawność Wykonania

Szeregowalność (*serializability*)

Papadimitrou. *The Serializability of Concurrent Database Updates*. Journal of the ACM, 1979.

Historia H:

$$\{x = 1\} \quad \begin{array}{ll} T_1 & [\![\ r(x)1, \ \ w(x)2\ ]\!] \\ T_2 & [\![\ r(x)1, \ \ w(x)2\ ]\!] \end{array} \quad \circlearrowleft\!\!\rightarrow \ T_2' \ [\![\ r(x)2, w(x)3\ ]\!] \quad \{x = 3\}$$

Poprawność Wykonania

Szeregowalność (*serializability*)

Papadimitrou. *The Serializability of Concurrent Database Updates*. Journal of the ACM, 1979.

Historia H:

$$\{x = 1\} \quad T_1 \ [\![\ r(x)1, \ \ w(x)2 \ ]\!] \\ T_2 \ \ [\![\ r(x)1, \ \ w(x)2 \ \circlearrowleft\!\!\rightarrow T_2' \ [\![\ r(x)2, w(x)3 \ ]\!] \quad \{x = 3\}$$

Historia szeregowa S ekwiwalentna do H:

$$\{x = 1\} \quad T_1 \ [\![\ r(x)1, w(x)2 \ ]\!] \quad T_2' \ [\![\ r(x)2, w(x)3 \ ]\!] \quad \{x = 3\}$$

Bezpieczeństwo Pamięci Transakcyjnych

```
transaction {
    x = 2
    y = 4
}
```

```
// precondition: y > x
transaction {
    local_array[y - x]
}
```

Bezpieczeństwo Pamięci Transakcyjnych

```
transaction {
    x = 2
    y = 4
}
```

```
// precondition: y > x
transaction {
    local_array[y - x]
}
```

$$\{x = 0, y = 1\} \quad \begin{array}{ll} T_1 & [\![\ w(x)2, \qquad w(y)4\ ]\!] \\ T_2 & [\![\ r(y)1, r(x)2\ \circlearrowleft\!\!\rightarrow\ T_2'\ [\![\ r(y)4, r(x)2\ ]\!] \end{array}$$

Bezpieczeństwo Pamięci Transakcyjnych

```
transaction {
    x = 2
    y = 4
}
```

```
// precondition: y > x
transaction {
    local_array[y - x]
}
```

$$\{x = 0, y = 1\} \quad \begin{array}{ll} T_1 & [\![\ w(x)2, \qquad w(y)4\ ]\!] \\ T_2 & [\![\ r(y)1, r(x)2\ \circlearrowleft\!\!\rightarrow\ T_2'\ [\![\ r(y)4, r(x)2\ ]\!] \end{array}$$

Nieprzejrzystość (*opacity*):

> Historia musi być szeregowalna + żadna transakcja nie może odczytać wartości zapisanej przez niezatwierdzone transakcje.

Guerraoui and Kapałka. *Principles of Transactional Memory*. Morgan & Claypool, 2010.

Problemy: Rywalizacja

Duża Rywalizacja (*contention*): wiele transakcji konkuruje o niewielką pulę zasobów.

Problemy: Rywalizacja

Duża Rywalizacja (*contention*): wiele transakcji konkuruje o niewielką pulę zasobów.

$T_1 \ [\![\ r(x)1, w(x)2 \ ]\!]$
$T_2 \ \ [\![\ r(x)1, w(x)2$

Problemy: Rywalizacja

Duża Rywalizacja (*contention*): wiele transakcji konkuruje o niewielką pulę zasobów.

$T_1 \ [\![\ r(x)1, w(x)2 \ ]\!]$
$T_2 \ [\![\ r(x)1, w(x)2 \ \leftsquigarrow \ \rightarrow T_2' \ [\![\ r(x)2, w(x)3 \ ]\!]$

Problemy: Rywalizacja

Duża Rywalizacja (*contention*): wiele transakcji konkuruje o niewielką pulę zasobów.

T_1 $[\![r(x)1, w(x)2]\!]$
T_2 $[\![r(x)1, w(x)2 \ \circlearrowleft \ \rightarrow T_2' [\![r(x)2, w(x)3]\!]$
T_3 $[\![r(x)1, w(x)2$

Problemy: Rywalizacja

Duża Rywalizacja (*contention*): wiele transakcji konkuruje o niewielką pulę zasobów.

T_1 $[\![r(x)1, w(x)2]\!]$
T_2 $[\![r(x)1, w(x)2 \ \leftsquigarrow \ \rightarrow T_2' [\![r(x)2, w(x)3]\!]$
T_3 $[\![r(x)1, w(x)2 \ \leftsquigarrow \ \rightarrow T_3' [\![r(x)2, w(x)3$

Problemy: Rywalizacja

Duża Rywalizacja (*contention*): wiele transakcji konkuruje o niewielką pulę zasobów.

$T_1 \; [\![\; r(x)1, w(x)2 \;]\!]$
$T_2 \; [\![\; r(x)1, w(x)2 \; \circlearrowleft \; \rightarrow T_2' \; [\![\; r(x)2, w(x)3 \;]\!]$
$T_3 \; [\![\; r(x)1, w(x)2 \; \circlearrowleft \; \rightarrow T_3' \; [\![\; r(x)2, w(x)3 \; \circlearrowleft \; \rightarrow T_3'' \; [\![\; r(x)3, w(x)4 \;]\!]$

Problemy: Operacje Niewycofywalne

$T_i [\![\ ..., ir, ... \ ]\!]$

- nie operuje na danych współdzielonych
- wykonanie ma widoczne konsekwencje
- konsekwencje są niewycofywalne

Problemy: Operacje Niewycofywalne

$T_i [\![\ldots, ir, \ldots]\!]$

- nie operuje na danych współdzielonych
- wykonanie ma widoczne konsekwencje
- konsekwencje są niewycofywalne

```
transaction {
    x = x + 1
    fire_rocket()
}
```

Problemy: Operacje Niewycofywalne

$T_i [\![\ldots, ir, \ldots]\!]$

- nie operuje na danych współdzielonych
- wykonanie ma widoczne konsekwencje
- konsekwencje są niewycofywalne

```
transaction {
    x = x + 1
    local_lock.acquire()
    // ...
    local_lock.release()
}
```

Problemy: Operacje Niewycofywalne

$T_i [\![\ldots, ir, \ldots]\!]$

- nie operuje na danych współdzielonych
- wykonanie ma widoczne konsekwencje
- konsekwencje są niewycofywalne

```
transaction {
    x = x + 1
    local_lock.acquire()
    // ...
    local_lock.release()
}
```

$T_2 \; [\![\; r(x)1, w(x)2, ir \;\; \hookleftarrow \ldots \;\; [\![\; r(x)2, w(x)2, ir$

Pesymistyczne vs Optymistyczne Pamięci Transakcyjne

Podejście optymistyczne:

$$T_1 \; [\![\; r(x)1, w(x)2 \;]\!]$$
$$T_2 \; [\![\quad r(x)1, \quad w(x)2 \quad \circlearrowleft \quad \rightarrow \quad T_2' \; [\![\; r(x)2, w(x)3 \;]\!]$$

Podejście pesymistyczne

$$T_1 \; [\![\; r(x)1, w(x)2 \;]\!]$$
$$T_2 \; [\![\qquad\qquad \searrow r(x)2, w(x)3 \;]\!]$$

Pesymistyczne vs Optymistyczne Pamięci Transakcyjne

Podejście optymistyczne:

$$T_1 \; [\![\; r(x)1, w(x)2 \;]\!]$$
$$T_2 \; [\![\quad r(x)1, \quad w(x)2 \quad \hookleftarrow \quad \rightarrow \quad T_2' \; [\![\; r(x)2, w(x)3 \;]\!]$$

Podejście pesymistyczne

$$T_1 \; [\![\; r(x)1, w(x)2 \;]\!]$$
$$T_2 \; [\![\qquad\qquad \searrow r(x)2, w(x)3 \;]\!]$$

- Zachowanie abstrakcji transakcji
- Tolerowanie wysokiej rywalizacji
- Bezpieczna obsługa operacji nieodwołalnych

Pesymistyczne vs Optymistyczne Pamięci Transakcyjne

Podejście optymistyczne:

$$T_1 \; [\![\; r(x)1, w(x)2 \;]\!]$$
$$T_2 \; [\![\quad r(x)1, \quad w(x)2 \; \circlearrowleft \quad \rightarrow \quad T_2' \; [\![\; r(x)2, w(x)3 \;]\!]$$

Podejście pesymistyczne

$$T_1 \; [\![\; r(x)1, w(x)2 \;]\!]$$
$$T_2 \; [\![\qquad\qquad \searrow r(x)2, w(x)3 \;]\!]$$

- Zachowanie abstrakcji transakcji
- Tolerowanie wysokiej rywalizacji
- Bezpieczna obsługa operacji nieodwołalnych

Matveev, Shavit. Towards a Fully Pessimistic STM Model. TRANSACT'12.

Afek, Matveev, Shavit. Pessimistic Software Lock-Elision. DISC'12.

Supremum Versioning Algorithm

SVA w skrócie:

T_i przy starcie dostaje następny wolny numer wersji (numer wersji) dla każdego z zasobów x, y, z

Supremum Versioning Algorithm

SVA w skrócie:

T_i przy starcie dostaje następny wolny numer wersji (numer wersji) dla każdego z zasobów x, y, z

T_i ma dostęp do x jak tylko numer wersji T_i dla x jest równy licznikowi wersji zmiennej x, w przeciwnym wypadku T_i czeka

Supremum Versioning Algorithm

SVA w skrócie:

T_i przy starcie dostaje następny wolny numer wersji (numer wersji) dla każdego z zasobów x, y, z

T_i ma dostęp do x jak tylko numer wersji T_i dla x jest równy licznikowi wersji zmiennej x, w przeciwnym wypadku T_i czeka

T_i przy zatwierdzeniu x, y, z licznik wersji każdej ze zmiennych jest powiększony o 1 (transakcja z następnym numerem wersji dostaje dostęp do x, y, z)

Supremum Versioning Algorithm

SVA w skrócie:

T_i przy starcie dostaje następny wolny numer wersji (numer wersji) dla każdego z zasobów x, y, z

T_i ma dostęp do x jak tylko numer wersji T_i dla x jest równy licznikowi wersji zmiennej x, w przeciwnym wypadku T_i czeka

T_i przy zatwierdzeniu x, y, z licznik wersji każdej ze zmiennych jest powiększony o 1 (transakcja z następnym numerem wersji dostaje dostęp do x, y, z)

Kiedy T_i używa zmiennej x ostatni raz (wg *supremum*) licznik wersji zmiennej x jest powiększony o 1

Wojciechowski. *Isolation-only Transactions by Typing and Versioning.* PPDP'05.

Supremum Versioning Algorithm

SVA w skrócie:

- T_i przy starcie dostaje następny wolny numer wersji (numer wersji) dla każdego z zasobów x, y, z
- T_i ma dostęp do x jak tylko numer wersji T_i dla x jest równy licznikowi wersji zmiennej x, w przeciwnym wypadku T_i czeka
- T_i przy zatwierdzeniu x, y, z licznik wersji każdej ze zmiennych jest powiększony o 1 (transakcja z następnym numerem wersji dostaje dostęp do x, y, z)
- Kiedy T_i używa zmiennej x ostatni raz (wg *supremum*) licznik wersji zmiennej x jest powiększony o 1

Wojciechowski. *Isolation-only Transactions by Typing and Versioning.* PPDP'05.

Wymagana wiedzy *a priori* o *supremach*: max liczbie dostępów do zmiennych w transakcji

Siek, Wojciechowski. *A Formal Design of a Tool for Static Analysis of Upper Bounds on Object Calls.* FMICS'12.

Zalety Wczesnego Zwalniania Zasobów

Wykonanie bez wczesnego zwalniania

T_1 $[\![r(x)1, w(x)2, r(y)1, w(y)2]\!]$

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

Wczesne zwalnianie przy ostatnim użyciu

T_1 $[\![r(x)1, w(x)2, r(y)1, w(y)2]\!]$

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

Zalety Wczesnego Zwalniania Zasobów

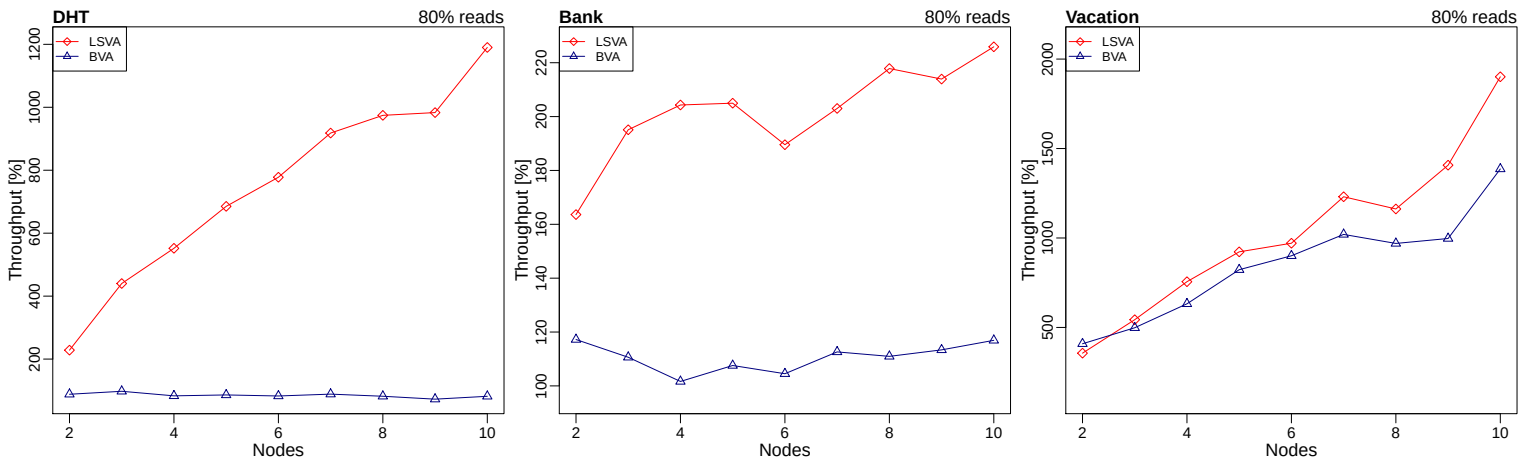
Wykonanie bez wczesnego zwalniania

$$T_1 \; [\![\; r(x)1, w(x)2, r(y)1, w(y)2 \;]\!]$$
$$T_2 \; [\![\qquad\qquad\qquad\qquad \searrow r(x)2, w(x)3 \;]\!]$$

Wczesne zwalnianie przy ostatnim użyciu

$$T_1 \; [\![\; r(x)1, \color{red}{w(x)2}, r(y)1, w(y)2 \;]\!]$$
$$T_2 \; [\![\qquad\qquad \searrow \color{red}{r(x)2}, w(x)3 \;]\!]$$

Poprawa wydajności:



Zalety Wczesnego Zwalniania Zasobów

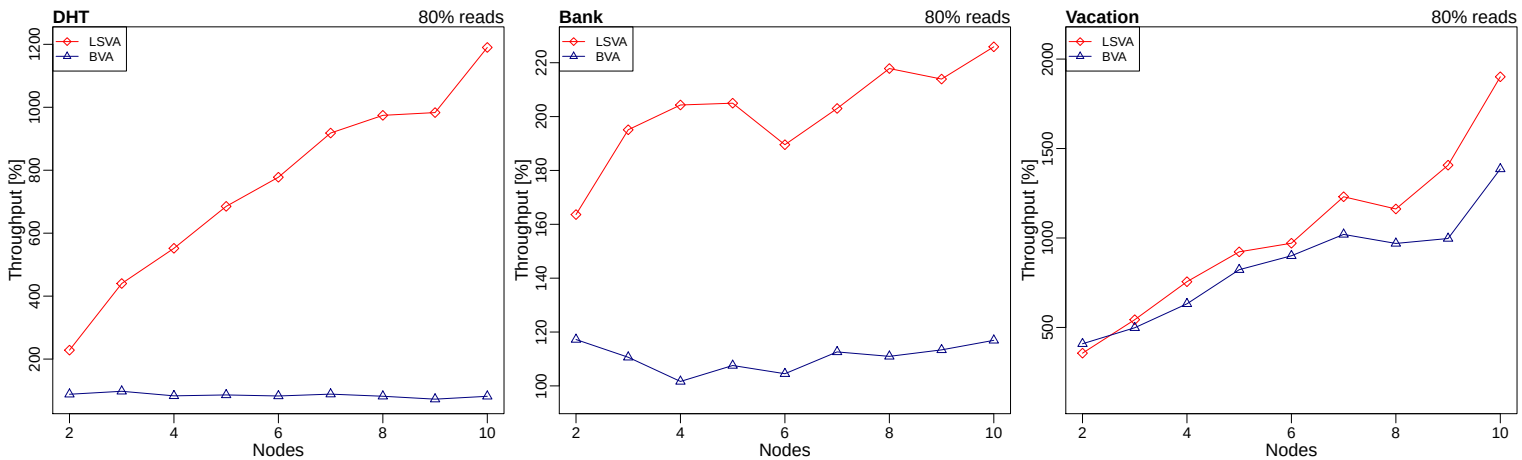
Wykonanie bez wczesnego zwalniania

$$T_1 \; [\![\; r(x)1, w(x)2, r(y)1, w(y)2 \;]\!]$$
$$T_2 \; [\![\qquad\qquad\qquad\qquad \searrow r(x)2, w(x)3 \;]\!]$$

Wczesne zwalnianie przy ostatnim użyciu

$$T_1 \; [\![\; r(x)1, w(x)2, r(y)1, w(y)2 \;]\!]$$
$$T_2 \; [\![\qquad\qquad \searrow r(x)2, w(x)3 \;]\!]$$

Poprawa wydajności:



Unika całkowicie wycofań. Spełnia *opacity*.

Siek, Wojciechowski. *Atomic RMI: a Distributed Transactional Memory Framework*. HLPP'14/IJPP.

Rollback

Konieczność możliwości programistycznego wycofania transakcji (*rollback*):

- Silniejsze (bardziej uniwersalne) narzędzie
- Niezbędne dla mechanizmów tolerowania awarii

Siek, Wojciechowski. Brief Announcement: Towards a Fully-Articulated Pessimistic Distributed Transactional Memory. SPAA'13.

Rollback

Konieczność możliwości programistycznego wycofania transakcji (*rollback*):

- Silniejsze (bardziej uniwersalne) narzędzie
- Niezbędne dla mechanizmów tolerowania awarii

Siek, Wojciechowski. Brief Announcement: Towards a Fully-Articulated Pessimistic Distributed Transactional Memory. SPAA'13.

Wprowadza wycofania i kaskady wycofań

$$T_1 \; [\![\; r(x)1, w(x)2, r(y)1, w(y)2, \; \circlearrowleft$$
$$T_2 \; [\![\qquad\qquad \searrow r(x)2, w(x)3 \qquad \searrow \circlearrowleft \; \ldots$$

Rollback

Konieczność możliwości programistycznego wycofania transakcji (*rollback*):

- Silniejsze (bardziej uniwersalne) narzędzie
- Niezbędne dla mechanizmów tolerowania awarii

Siek, Wojciechowski. Brief Announcement: Towards a Fully-Articulated Pessimistic Distributed Transactional Memory. SPAA'13.

Wprowadza wycofania i kaskady wycofań

$$T_1 \; [\![\; r(x)1, w(x)2, r(y)1, w(y)2, \; \circlearrowleft$$
$$T_2 \; [\![\qquad\qquad \searrow r(x)2, w(x)3 \qquad \searrow \circlearrowleft \ldots$$

Wprowadza niespójne widoki: nie spełnia *opacity*
Kompromis między bezpieczeństwem i elastycznością abstrakcji

Niespójne widoki

Możliwy niespójny widok (możliwe w SVA):

$$
\begin{array}{lll}
T_i & [\![\ \mathrm{w}(x)0, & \mathrm{w}(x)1 \quad \circlearrowleft \\
T_j & & [\![\ \searrow \mathrm{r}(x)0 \ \searrow \circlearrowleft \ T_j' [\![\ \mathrm{r}(x)1, \mathrm{w}(x)2\ ]\!]
\end{array}
$$

Niespójne widoki

Możliwy niespójny widok (możliwe w SVA):

$$
\begin{array}{lll}
T_i & [\![\ \mathrm{w}(x)0, & \mathrm{w}(x)1 \quad \circlearrowleft \\
T_j & & [\![\ \searrow \mathrm{r}(x)0 \ \searrow \circlearrowleft\ T_j' [\![\ \mathrm{r}(x)1, \mathrm{w}(x)2\ ]\!]
\end{array}
$$

Nadpisywanie (wykluczone przez SVA):

$$
\begin{array}{lll}
T_i & [\![\ \mathrm{w}(x)0, & \mathrm{w}(x)1\ ]\!] \\
T_j & & [\![\ \searrow \mathrm{r}(x)0 \ \searrow \circlearrowleft\ T_j'\ [\![\ \mathrm{r}(x)1, \mathrm{w}(x)2\ ]\!]
\end{array}
$$

Własności Bezpieczeństwa i Wczesne Zwalnianie

- Serializability
- Elastic Opacity
- Virtual World Consistency
- TMS1 & TMS2
- Opacity

Własności Bezpieczeństwa i Wczesne Zwalnianie

- Serializability
- Elastic Opacity
- Virtual World Consistency
- TMS1 & TMS2
- Opacity

Siek, Wojciechowski. Zen and the Art of Concurrency Control: An Exploration of TM Safety Property Space with Early Release in Mind. WTTM'14.

Last-use opacity (nieprzejrzystość do ostatniego użycia)

Składowe nieprzejrzystości:

- Szeregowalność (*serializability*)
- Porządek czasu rzeczywistego (*real-time order*)
- Spójność (*consistency*)

Last-use opacity (nieprzejrzystość do ostatniego użycia)

Składowe nieprzejrzystości:

- Szeregowalność (*serializability*)
- Porządek czasu rzeczywistego (*real-time order*)
- Spójność (*consistency*)

Składowe **nieprzejrzystości do ostatniego użycia**:

- Szeregowalność (*serializability*)
- Porządek czasu rzeczywistego (*real-time order*)
- Spójność do ostatniego użycia (*last-use consistency*)
- Odtwarzalność (*recoverability*)

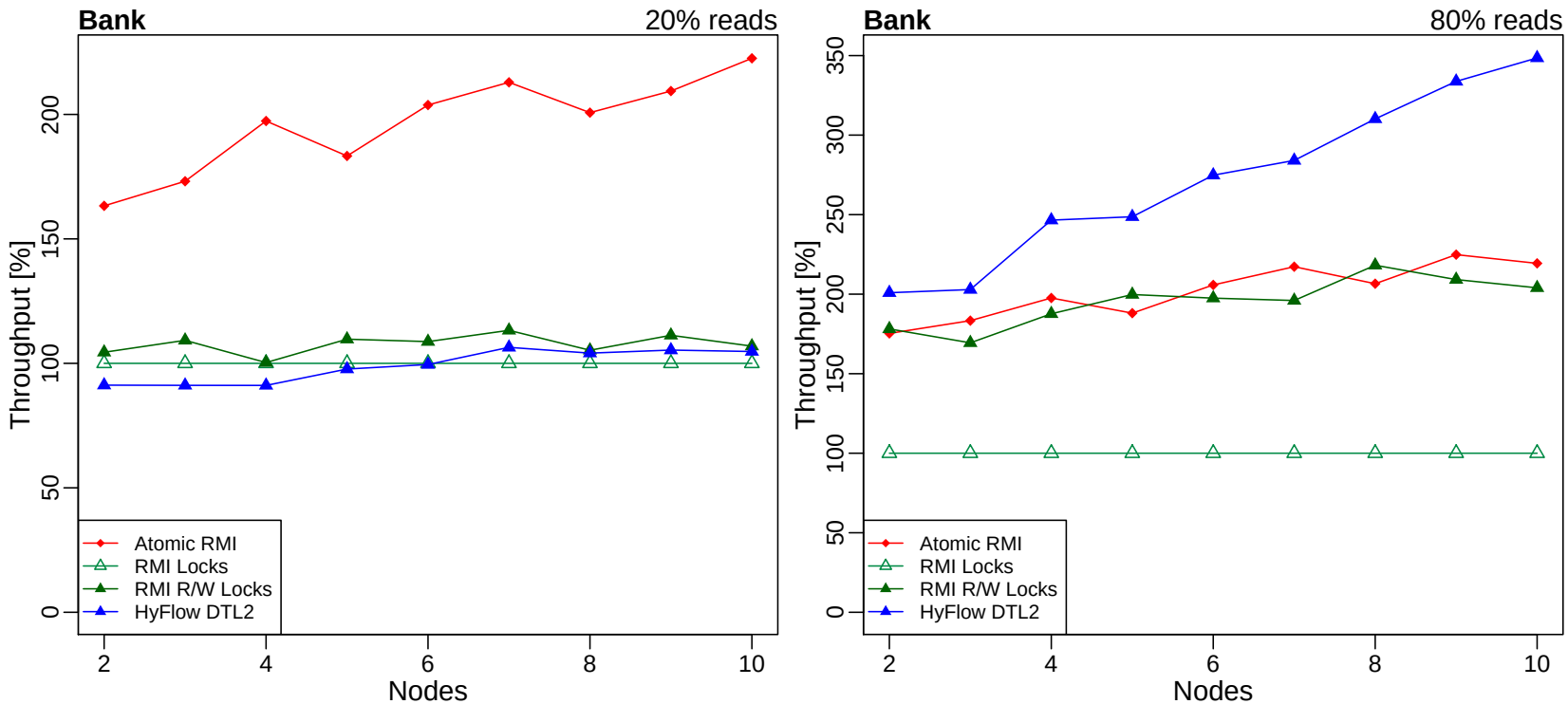
Siek, Wojciechowski. Relaxing Opacity in Pessimistic Transactional Memory. DISC'14.

Odczyty w SVA

Zdalne wywołanie metod (RMI) *vs* rozproszone bazy danych:
SVA nie rozróżnia operacji odczytu i zapisu.

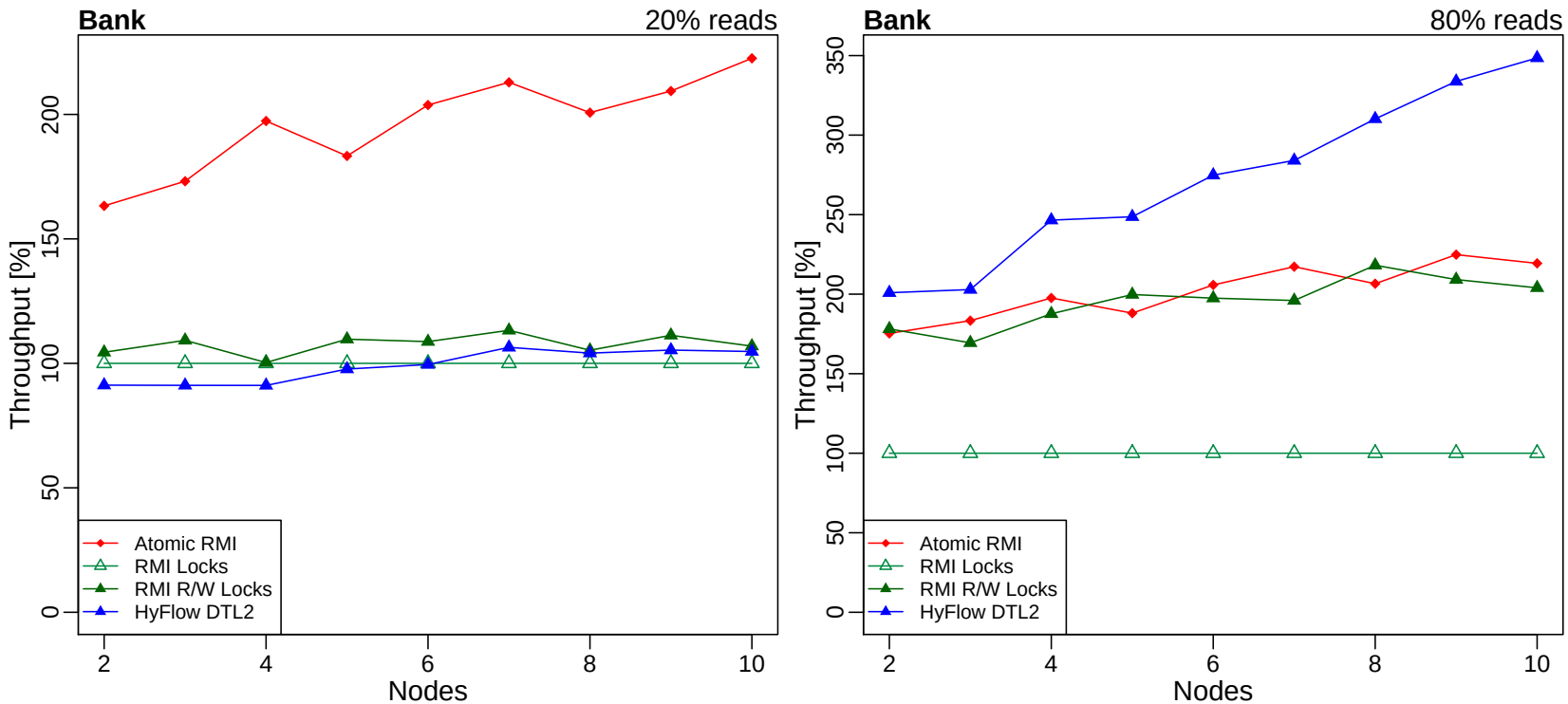
Odczyty w SVA

Zdalne wywołanie metod (RMI) *vs* rozproszone bazy danych:
SVA nie rozróżnia operacji odczytu i zapisu.



Odczyty w SVA

Zdalne wywołanie metod (RMI) *vs* rozproszone bazy danych:
SVA nie rozróżnia operacji odczytu i zapisu.



Optymalizacje SVA dozwalające dodatkowe przeploty.

Lipton. Reduction: A Method of Proving Properties of Parallel Programs. Comm. of the ACM. 1975.

OptSVA: Transakcje Aktualizujące

Transakcja wykonująca wyłącznie zapisy:

- Buforowanie zapisów
- Aktualizacja zmiennych opóźniona do zatwierdzenia
- Pobieranie numerów wersji opóźnione do zatwierdzenia

OptSVA: Zmienne Tylko-do-odczytu

Transakcja wykonująca wyłącznie odczyty na zmiennej:

- Buforowanie wartości zmiennej przy starcie transakcji
- Zwolnienie zmiennej po buforowaniu
- Odczyt wyłącznie z bufora

OptSVA: Buforowanie i Wczesne Zwalnianie

Transakcja wykonująca odczyty i zapisy na zmiennej:

- Buforowanie wartości zmiennej przy pierwszym zapisie
- Zwolnienie zmiennej po ostatnim zapisie
- Odczyt z bufora, jeśli dostępny

OptSVA

OptSVA spełnia *last-use opacity*

OptSVA

OptSVA spełnia *last-use opacity*

OptSVA jest optymalnym algorytmem spełniającym *last-use-opacity*

Transakcje Ostatecznie Spójne

Transakcja T_1

$$T_1 \; [\![\; r(x)v, w(x)u \;]\!]$$

Transakcje Ostatecznie Spójne

Transakcja T_1

$$T_1 \; [\![\; r(x)v, w(x)u \;]\!]$$

Tryb spójny

$$T_1^c \; [\![\; r(x)v, w(x)u \;]\!]$$

Tryb ostatecznie spójny (słaby)

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

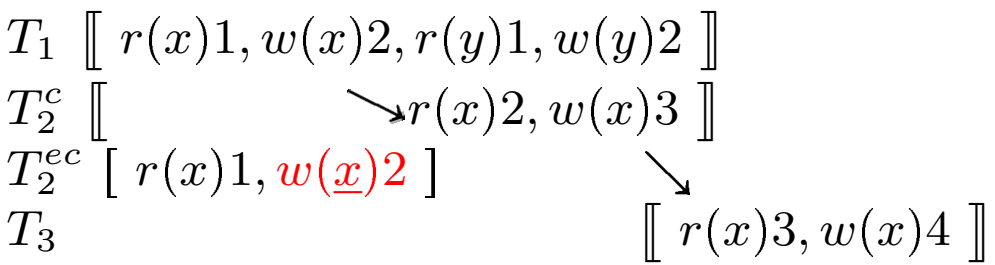
Tryby uruchamiane są jednocześnie → zbieżność (*convergence*)

Wojciechowski, Siek. *Having Your Cake and Eating it Too: Combining Strong and Eventual Consistency.* PaPEC'14.

Spójny odczyt w SVA w praktyce

Zbuforowana najnowsza spójna migawka → słabe transakcje nie czekają/nie blokują dostępu do danych współdzielonych

Zapisy w trybie słabym ukrywane przed innymi transakcjami.



Podsumowanie

- *SVA*: pesymistyczne sterowanie współbieżnością z wczesnym zwalnianiem zasobów
- *Last-use consistency*: własność spójności przy wczesnym zwalnianiu zasobów
- *Last-use opacity*: własność bezpieczeństwa przy wczesnym zwalnianiu zasobów
- *OptSVA*: optymalny algorytm spełniający last-use opacity
- Pamięć transakcyjna z ostateczną spójnością

Publikacje

Konrad Siek, Paweł T. Wojciechowski. *Atomic RMI: a Distributed Transactional Memory Framework*. International Journal of Parallel Processing, 2015 (TBR). **15 pt**.

Konrad Siek, Paweł T. Wojciechowski. *A Formal Design of a Tool for Static Analysis of Upper Bounds on Object Calls in Java*. In Proceedings of FMICS 2012: the 17th International Workshop on Formal Methods for Industrial Critical Systems (co-located with FM 2012). Lecture Notes in Computer Science, pages 192–206. August 2012. **13 pt**.

Konrad Siek, Paweł T. Wojciechowski. *Last-use Opacity: A Strong TM Safety Property with Early Release Support*. Theoretical Computer Science. **20 pt**. (Under Review)

Konrad Siek, Paweł T. Wojciechowski. *Transactions Scheduled While You Wait: Augmenting Transactional Memory with a Sorting Queue* . Parallel Computing. **35 pt**. (Under Review)

Publikacje Konferencyjne

Konrad Siek, Paweł T. Wojciechowski. *Atomic RMI: a Distributed Transactional Memory Framework*. In proceedings of HLPP 2014: the 7th International Symposium on High-level Parallel Programming and Applications. July 2014.

Konrad Siek, Paweł T. Wojciechowski. *Brief Announcement: Relaxing Opacity in Pessimistic Transactional Memory*. In Proceedings of DISC'14: the 28th International Symposium on Distributed Computing. October 2014.

Konrad Siek, Paweł T. Wojciechowski. *Zen and the Art of Concurrency Control: An Exploration of TM Safety Property Space with Early Release in Mind*. In Proceedings of WTTM'14: the 6th Workshop on the Theory of Transactional Memory. July 2014.

Paweł T. Wojciechowski, Konrad Siek. *Having Your Cake and Eating it Too: Combining Strong and Eventual Consistency.* In Proceedings of PaPEC 2014: the 1st Workshop on the Principles and Practice of Eventual Consistency. April 2014.

Konrad Siek, Paweł T. Wojciechowski. *Towards a Fully-Articulated Pessimistic Distributed Transactional Memory (Brief announcement).* In Proceedings of SPAA 2013: the 25th ACM Symposium on Parallelism in Algorithms and Architectures. July 2013.

Paweł T. Wojciechowski, Konrad Siek. *Transaction Concurrency Control via Dynamic Scheduling Based on Static Analysis (Extended Abstract).* In Proceedings of WTM 2012: Euro-TM Workshop on Transactional Memory (co-located with ACM SIGOPS EuroSys 2012). April 2012.

Konrad Siek, Paweł T. Wojciechowski. *Statically Computing Upper Bounds on Object Calls for Pessimistic Concurrency Control (Extended Abstract).* In Proceedings of EC^2 2010: Workshop on Exploiting Concurrency Efficiently and Correctly (co-located with CAV 2010). July 2010.

?