CREATM: Cost Reduction and Elision of Aborts in Transactional Memory

Rahul Shrivastava & JyothiKrishna V S

Presented by: Rahul Shrivastava & JyothiKrishna V S

IIT Madras

29/5/2015

Transactional Memory

- Abstraction for mutual exclusion.
- Simplifies programming.
- Internally uses fine grained locks.

Transactional Memory

- Abstraction for mutual exclusion.
- Simplifies programming.
- Internally uses fine grained locks.
- Commit: Effects visible globally.
- Abort: Actions are rollbacked.
- Aborts reduce concurrency.

Elision of False Aborts

- False Aborts: Different memory locations mapped to same lock.

T1	T2
TRANSACTION_BEGIN( )	TRANSACTION_BEGIN( )
1: SHARED_WRITE(A.x, 4);	2: SHARED_WRITE(A.y, 5);
TRANSACTION_END( )	TRANSACTION_END( )

Elision of False Aborts

- False Aborts: Different memory locations mapped to same lock.

T1	T2
TRANSACTION_BEGIN()	TRANSACTION_BEGIN()
1: SHARED_WRITE(A.x,4);	2: SHARED_WRITE(A.y,5);
TRANSACTION_END()	TRANSACTION_END()

- MHP + field sensitive alias analysis
- Let T_i be transaction and D_i be the access set

$\forall$ d $\in$ D_i:
  $\forall$ T_j :
    if $MHP(T_i, T_j)$ == TRUE then:
     if d $\notin$ D_j then:
      replace SHARED_WRITE(d) with normal write.

Reduction of CAS

```
SHARED_WRITE(queuePtr->elements, newElements);
SHARED_WRITE(queuePtr->pop, newCapacity - 1);
SHARED_WRITE(queuePtr->capacity, newCapacity);
```

- Need to perform 3 CAS when only one could be sufficient.

Reduction of CAS

```
SHARED_WRITE(queuePtr->elements, newElements);
SHARED_WRITE(queuePtr->pop, newCapacity - 1);
SHARED_WRITE(queuePtr->capacity, newCapacity);
```

- Need to perform 3 CAS when only one could be sufficient.
- Global lock acquire for all the three writes.

Multiple granularity of locks

T1

```
TRANSACTION_BEGIN()
 SHARED_WRITE(Ptr->a, n1);
 SHARED_WRITE(Ptr->b, n2);
 SHARED_WRITE(Ptr->c, n3);
TRANSACTION_END()
```

barrier()

T2

```
TRANSACTION_BEGIN()
SHARED_WRITE(Ptr->a, n4);
TRANSACTION_END()
```

T3

```
TRANSACTION_BEGIN()
SHARED_WRITE(Ptr->b, n5);
TRANSACTION_END()
```

Multiple granularity of locks

```
                        T1
                TRANSACTION_BEGIN()
                  SHARED_WRITE(Ptr->a, n1);
                  SHARED_WRITE(Ptr->b, n2);
                  SHARED_WRITE(Ptr->c, n3);
                TRANSACTION_END()
        ----------------barrier()----------------

          T2                                T3
TRANSACTION_BEGIN()               TRANSACTION_BEGIN()
SHARED_WRITE(Ptr->a, n4);         SHARED_WRITE(Ptr->b, n5);
TRANSACTION_END()                 TRANSACTION_END()
```

- T1 would perform better with a global lock.
- T2 and T3 would perform better with a word based lock

Multiple granularity of locks

```
                    T1
TRANSACTION_BEGIN()
  SHARED_WRITE(Ptr->a, n1);
  SHARED_WRITE(Ptr->b, n2);
  SHARED_WRITE(Ptr->c, n3);
TRANSACTION_END()
-------------barrier()-------------

        T2                                  T3
TRANSACTION_BEGIN()                 TRANSACTION_BEGIN()
SHARED_WRITE(Ptr->a, n4);           SHARED_WRITE(Ptr->b, n5);
TRANSACTION_END()                   TRANSACTION_END()
```

- T1 would perform better with a global lock.
- T2 and T3 would perform better with a word based lock
- We propose multiple granularity of lock - hybrid approach

STAMP Benchmark

Benchmark	Trans commit	% Trans Aborted	% False Abort	No of CAS	Exe Time(s)
bayes	1981	6.8	2.2		4.43
genome	2489220	0.2	0.5		1.06
intruder	23428133	28.2	8.3	300	7.94
kmeans	11098403	10.8	5.1		3.87
labyrinth	1040	9.1	0		8.15
ssca2	22362293	0.2*e-5	86.2		5.79
vacation	4194304	0.03	0.03		3.71
yada	2609140	436	10.7	6680097	29.23

Related Work

- Reconciling transactional conflicts with compilers help[S. Mannarswamy et al.]
- Lowering the overhead of nonblocking software transactional memory[V. J. Marathe et al.]

Conclusion and Future Work

- Number of false aborts are significant, could improve the performance if reduced.
- Multiple granularity of lock would save significant number of expensive CAS without inducing false aborts.
- Plan - To understand all the benchmarks and play with their parameters(number of threads and application specific).
- Identify more opportunites to reduce false aborts and number of CAS operations.

STAMP Benchmark

Benchmark	Trans commit	% Trans Aborted	% False Abort	No of CAS	Exe Time(s)
bayes	1981	6.8	2.2		4.43
genome	2489228	6.4	17.7		1.06
intruder	23428157	113.5	17.6	300	7.94
kmeans	8741300	89.8	9.7		3.87
labyrinth	1088	588.8	93.8		5.15
ssca2	22362293	0.5	87.4		5.79
vacation	4194304	0.42	33.3	6141188	2.14
yada	29032242	2315.9	6.9	6680097	29.23