Транзакционная память
STM
HTM

Калишенко Е.Л.
НИУ ВШЭ 2019

Проблемы locking ||

- Нет связи ресурс-примитив
- Сложный контроль времени жизни
- Инверсия приоритетов и т.п
- Сложность алгоритмов на CAS
- Проблемы с атомарностью операций над несколькими структурами данных:
  например, переложить данные из одной hash-таблицы в другую так, чтобы клиент не видел остутствия элемента

Транзакционность

- Организация транзакции
- Проверка на конфликты
- Применение или отмена

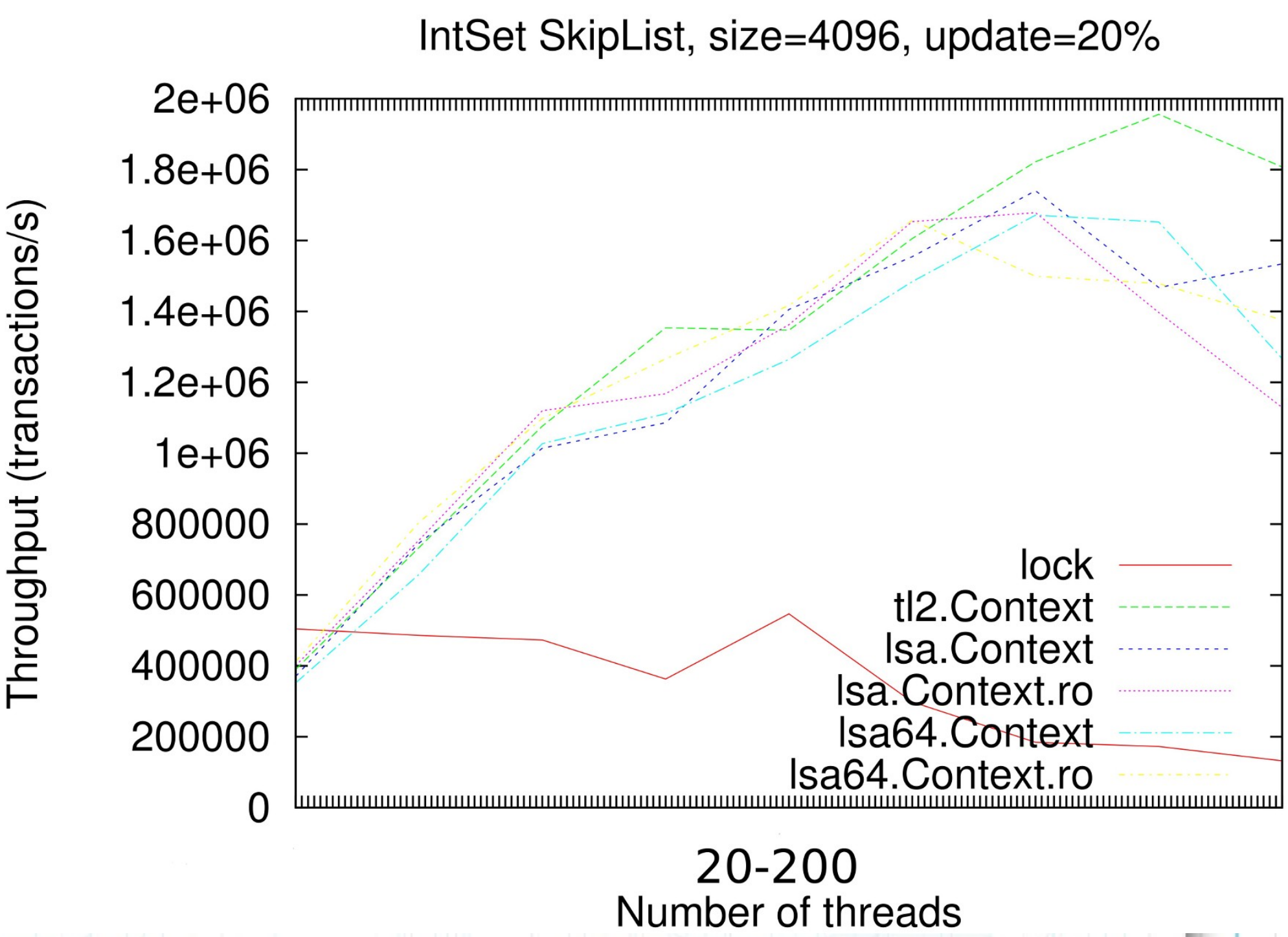
Пример очереди

```
1   public class TransactionalQueue<T> {
2     private Node head;
3     private Node tail;
4     public TransactionalQueue() {
5       Node sentinel = new Node(null);
6       head = sentinel;
7       tail = sentinel;
8     }
9     public void enq(T item) {
10      atomic {
11        Node node = new Node(item);
12        node.next = tail;
13        tail = node;
14      }
15    }
```

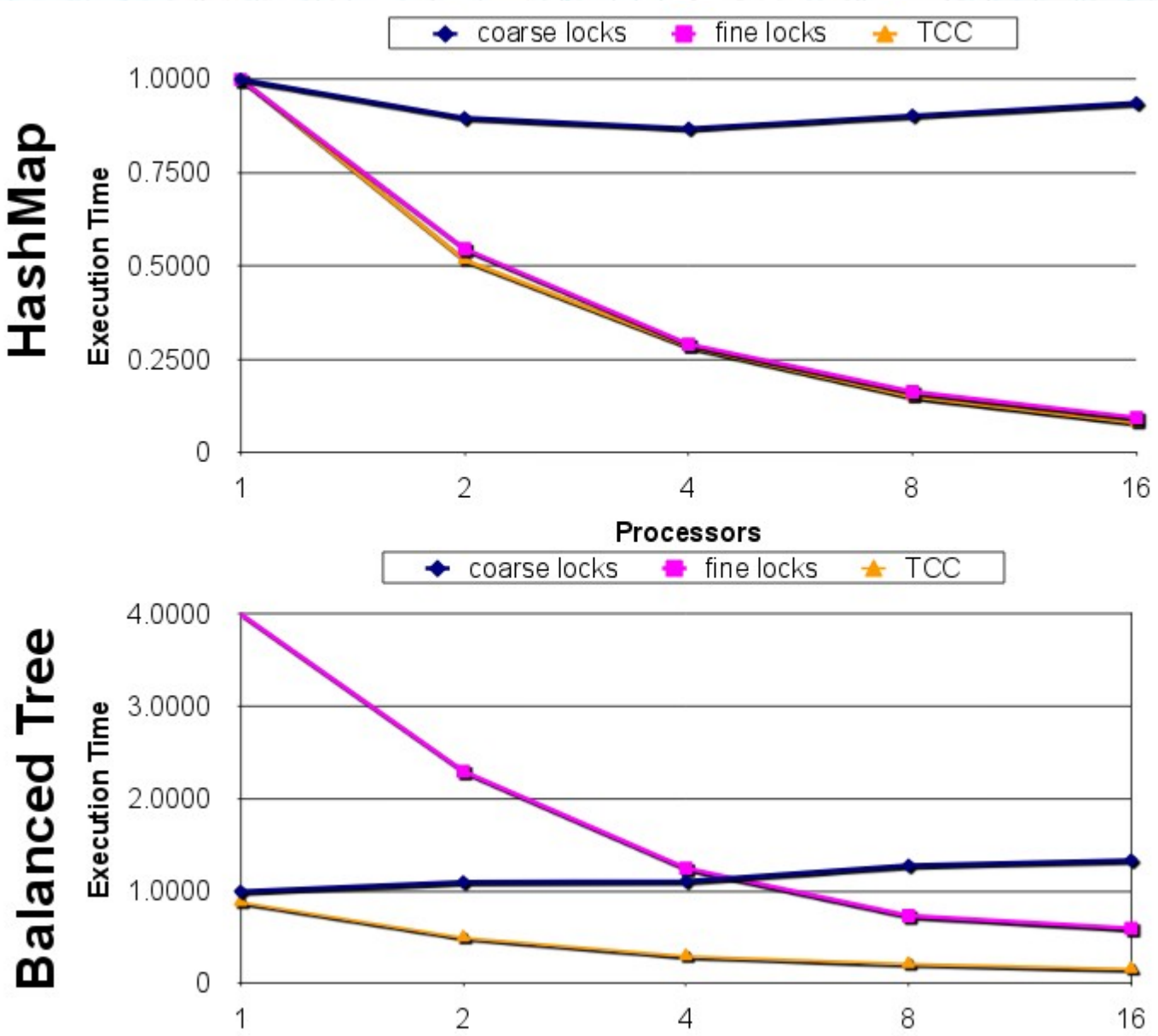
Принцип

```
int s;
int __declspec (tm_callable) f(int);

void foo(void)
{
      int i;
      for (i=0;i<10;i++)
      {
        atomic {
              s += f(i);
              if (s > 1000) __tm_abort;
        }
      }
}
```

```
int s;
int f(int);
static _ITM_srcLocation start_outer_loc = {…};
static _ITM_srcLocation commit_outer_loc = {…};
static _ITM_srcLocation abort_loc = {…};

void foo(void)
{
_ITM_transaction * td = _ITM_getTransaction();
 for (i=0;i<10;i++) {
    int doWhat = _ITM_beginTransaction (td,pr_instrumentedCode |
                                        &start_outer_loc);
    if (doWhat & a_restoreLiveVariables) {
    /* Compiler restores live local variables that aren't instrumented */
    }
    if (doWhat & a_abortTransaction) goto txn1_abort_label;
    if ((doWhat & a_saveLiveVariables)) {
    /* Compiler saves live local variables that won't be
     * instrumented.
     */
    }
    int sval = (int) _ITM_RU4 (td, (uint32 *)&s);
    sval += f_@TXN(i);     // Calls the instrumented "f"
    _ITM_WaRU4 (td, (uint32 *)&s, (unit32_t)sval);
    if (sval > 1000)
          _ITM_abortTransaction(td,userAbort,&abort_loc);
    _ITM_commitTransaction(td,commit_outer_loc);
    txn1_abort_label:
  }
  return;
}
```


IntSet SkipList, size=4096, update=20%
Throughput (transactions/s)
2e+06
1.8e+06
1.6e+06
1.4e+06
1.2e+06
1e+06
800000
600000
400000
200000
0
lock
tl2.Context
tsa.Context
tsa.Context.ro
tsa64.Context
tsa64.Context.ro
20-200
Number of threads

Locks and transactions



Преимущества

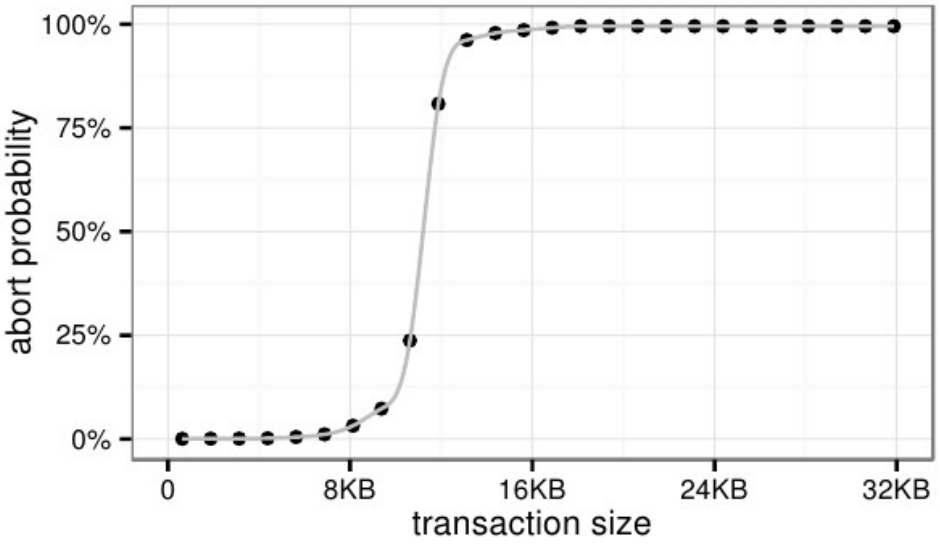
- Отсутствие блокировок в коде
- Возможность контроля: откат, повтор...
- Лучшая утилизация ресурсов*
- Выше уровень абстракции (что защищаем, а не как защищаем)

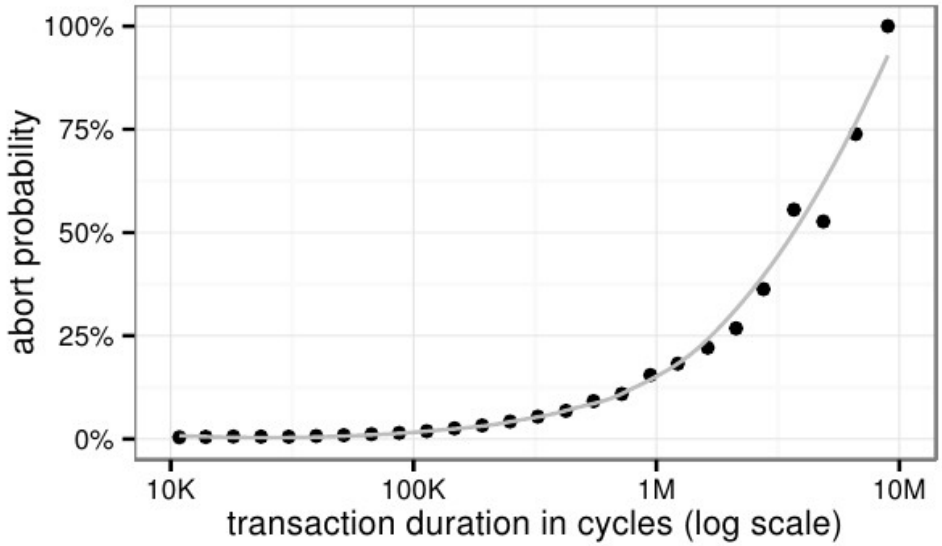
HTM

- Уже реализованы алгоритмы поддержки когерентности кэшей *(MESI)*
- Достаточно сыграть на битах транзакционности в линейках кэша

Ограничения HTM на кэше

- Транзакция ограничена размером кэша
- Время транзакции может быть ограничено квантом планирования





C++ TM

- 2008: сформирована группа для проработки части стандарта по TM
- 2009: 1.0 версия Draft
- 2011: 1.1 версия Draft
- 2016: реализация libitm.so в GCC 6.1

Пример TSE GCC 4.7

```
int contains(int value)
{
  int result;
  node_t *prev, *next;
  __transaction_atomic {
    prev = set->head;
    next = prev->next;
    while (next->val < val) {
      prev = next;
      next = prev->next;
    }
    result = (next->val == val);
  }
  return result;
}
```

Пример TSE GCC 6.1 (-fgnu-tm)

```
int f()
{
    static int i = 0;
    synchronized { // begin synchronized block
        std::cout << i << " -> ";
        ++i;       // each call to f() obtains a
unique value of i
        std::cout << i << '\n';
        return i; // end synchronized block
    }
}
```

Коды ошибок

```
if (_xbegin() == _XBEGIN_STARTED) {
  speculative code
} else if (status & _XABORT_EXPLICIT) {
  aborted by user code
} else if (status & _XABORT_CONFLICT) {
  read-write conflict
} else if (status & _XABORT_CAPACITY) {
  cache overflow
} else {
  …
}
```

Ошибки HTM

- 2014: Haswell TSX Bug (вышел 2013)
- 2016: Skylake TSX Bug

HSW136. Software Using Intel® TSX May Result in Unpredictable System BehaviorProblem:Under a complex set of internal timing conditions and system events, software using the Intel® TSX instructions may result in unpredictable system behavior.

SKL043. Detecting an Intel® PT Stopped or Error Condition Within an Intel® TSX Region May Result in a System Hang