Транзакционная память
STM
HTM

Калишенко Е.Л.
ПОМИ 2014

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

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    }
```

Преимущества

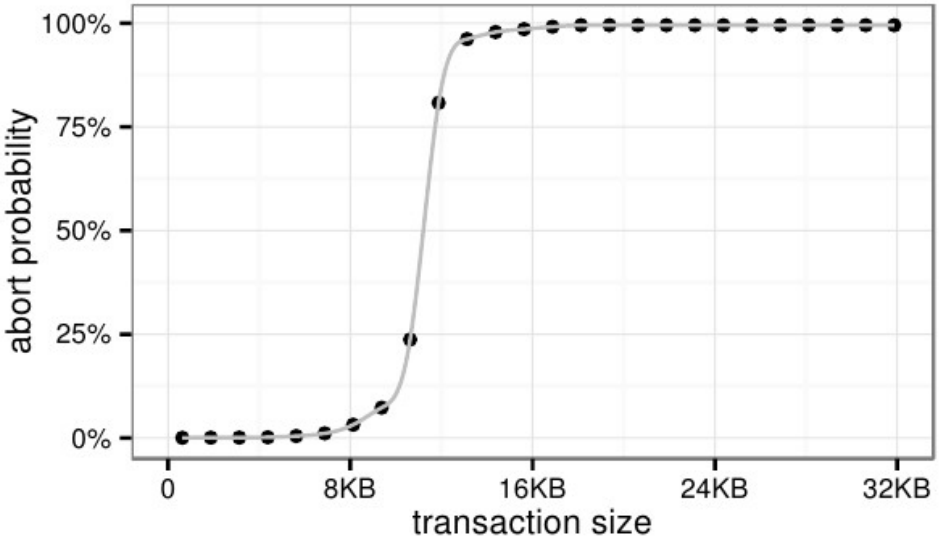
- Отсутствие блокировок в коде
- Возможность контроля: откат, повтор...
- Лучшая утилизация ресурсов*

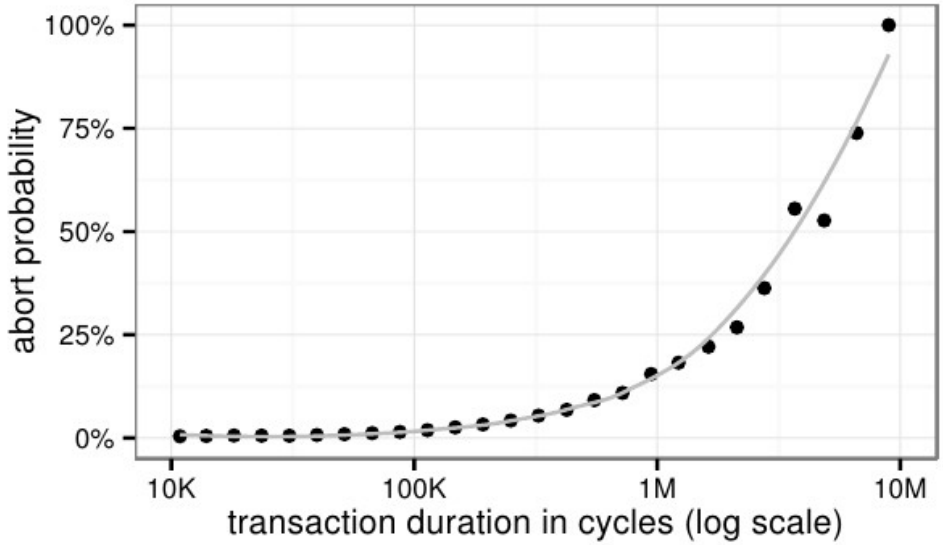
HTM

- 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





Пример Transactional Synchronization Extensions

- Расширение gcc 4.7:
  __transaction_atomic { c = a - b; }

```
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;
}
```