Реализация алгоритма распределенной блокировки на основе БД Cassandra

Денис Мухаметьянов

9 июня 2015

Google Chubby

- Подробно описанная технология
- Реализация закрыта

Apache ZooKeeper

- Открытый исходный код
- Нет готового клиента для C#

Нужно собственное решение

Google Chubby

- Подробно описанная технология
- Реализация закрыта

Apache ZooKeeper

- Открытый исходный код
- Нет готового клиента для C#

Нужно собственное решение

Google Chubby

- Подробно описанная технология
- Реализация закрыта

Apache ZooKeeper

- Открытый исходный код
- Нет готового клиента для C#

Нужно собственное решение

Google Chubby

- Подробно описанная технология
- Реализация закрыта

Apache ZooKeeper

- Открытый исходный код
- Нет готового клиента для C#

Нужно собственное решение

Google Chubby

- Подробно описанная технология
- Реализация закрыта

Apache ZooKeeper

- Открытый исходный код
- Нет готового клиента для C#

Нужно собственное решение

Google Chubby

- Подробно описанная технология
- Реализация закрыта

Apache ZooKeeper

- Открытый исходный код
- Нет готового клиента для C#

Нужно собственное решение

Google Chubby

- Подробно описанная технология
- Реализация закрыта

Apache ZooKeeper

- Открытый исходный код
- Нет готового клиента для C#

Нужно собственное решение

Старый алгоритм

- Использует особенности хранения разреженной таблицы в Cassandra
- Корректно решает поставленную задачу

Старый алгоритм

- Использует особенности хранения разреженной таблицы в Cassandra
- Корректно решает поставленную задачу

Старый алгоритм

- Использует особенности хранения разреженной таблицы в Cassandra
- Корректно решает поставленную задачу

Недостатки

- Неравномерное распределение блокировки между потоками
- Большое время захвата блокировки при конкурентности

Недостатки

- Неравномерное распределение блокировки между потоками
- Большое время захвата блокировки при конкурентности

Недостатки

- Неравномерное распределение блокировки между потоками
- Большое время захвата блокировки при конкурентности

Новый алгоритм

- Использует особенности хранения разреженной таблицы в Cassandra
- Выстраивает потоки в очередь в строке и распределяет блокировку между потоками в порядке этой очереди
- Корректно решает поставленную задачу

Новый алгоритм

- Использует особенности хранения разреженной таблицы в Cassandra
- Выстраивает потоки в очередь в строке и распределяет блокировку между потоками в порядке этой очереди
- Корректно решает поставленную задачу

Новый алгоритм

- Использует особенности хранения разреженной таблицы в Cassandra
- Выстраивает потоки в очередь в строке и распределяет блокировку между потоками в порядке этой очереди
- Корректно решает поставленную задачу

Новый алгоритм

- Использует особенности хранения разреженной таблицы в Cassandra
- Выстраивает потоки в очередь в строке и распределяет блокировку между потоками в порядке этой очереди
- Корректно решает поставленную задачу

- Алгоритм реализован на C#
- Проведено модульное и нагрузочное тестирование
- Проведено сравнение производительности со старым алгоритмом

- Алгоритм реализован на C#
- Проведено модульное и нагрузочное тестирование
- Проведено сравнение производительности со старым алгоритмом

- Алгоритм реализован на C#
- Проведено модульное и нагрузочное тестирование
- Проведено сравнение производительности со старым алгоритмом

Рис.: Моменты взятия блокировок старым алгоритмом

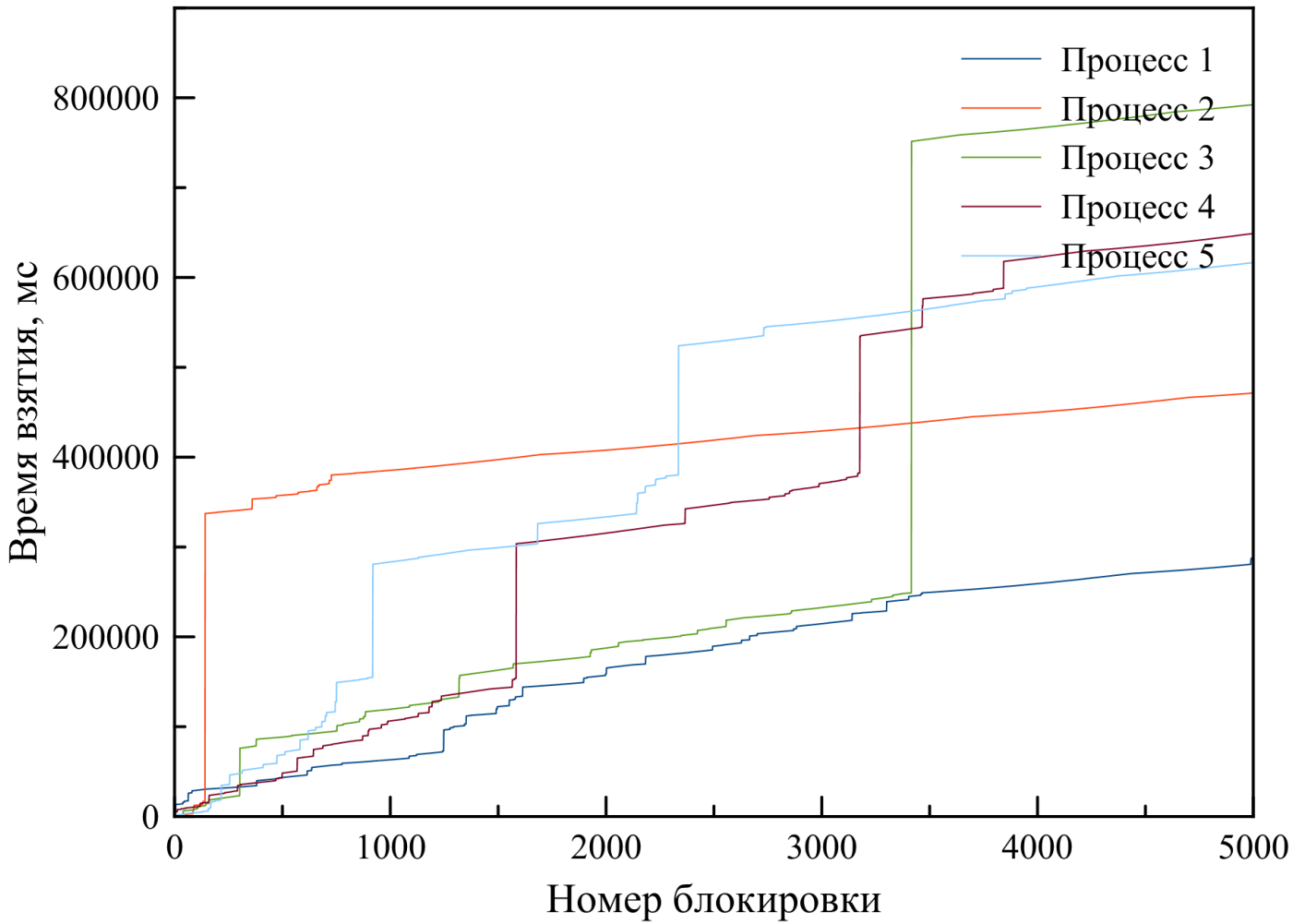


Рис.: Моменты взятия блокировок новым алгоритмом

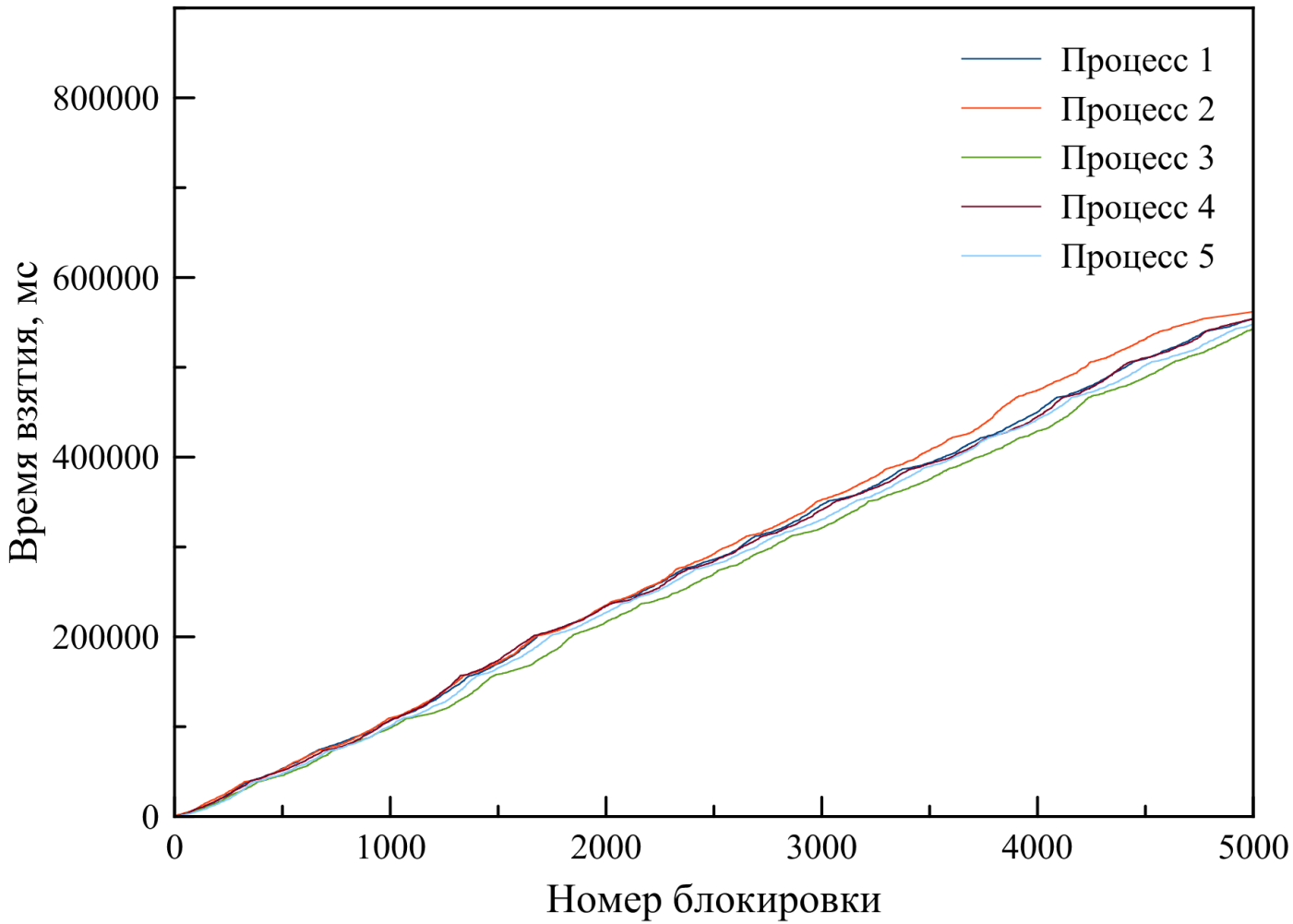
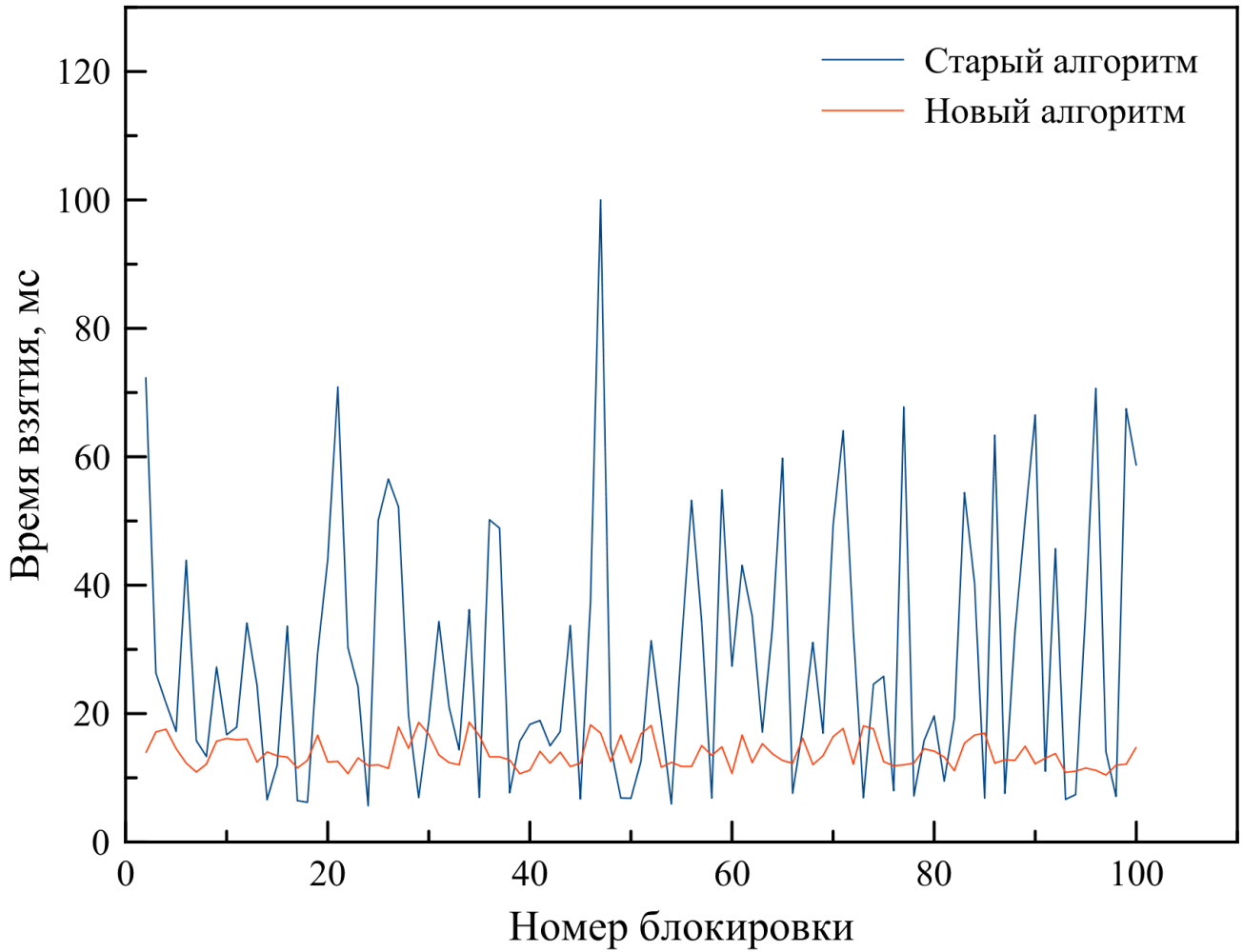


Рис.: Время взятия блокировки



Планы на будущее

- Промышленное использование
- Снижение среднего времени ожидания блокировки

Планы на будущее

- Промышленное использование
- Снижение среднего времени ожидания блокировки

Планы на будущее

- Промышленное использование
- Снижение среднего времени ожидания блокировки