А.Е.ЛАГУТИН[1], С.А.ХОССЕЙН[2]

ПОТОКОБЕЗОПАСНЫЕ СТРУКТУРЫ ДАННЫХ С ОСЛАБЛЕННОЙ СЕМАНТИКОЙ ВЫПОЛНЕНИЯ ОПЕРАЦИЙ

[1]*Учреждение образования «Белорусская государственная академия связи», г. Минск, Республика Беларусь*
[2]*Учреждение образования «Белорусский государственный университет информатики и радиоэлектроники», г. Минск, Республика Беларусь*

Многоядерные вычислительные системы сейчас представляют особенный интерес, так как разработчики процессоров стали уделять большее внимание количеству физических и логических ядер, нежели частоте одного ядра. Таким образом возникает проблема одновременного (параллельного) доступа к общим данным. Традиционные структуры данных: массивы, списки, деревья и другие, не приспособлены для работы в многопоточной системе. Для работы с общей структурой в разных потоках возникает необходимость в синхронизации доступа к данным, с использованием блокировок, что является серьезной проблемой при распараллеливании программ,

как с точки зрения времени выполнения, так и с точки зрения консистентности данных. Синхронизация доступа, к данным структуры, необходима, так как разные потоки могут одновременной обращаться к одной и той же структуре и соответственно изменять либо читать данные из нее. Параллельное выполнение этих операций невозможно, из-за возникающей гонки данных (data race), когда операция со структурой начинается до того, как данные были записаны или изменены.

Структуры с ослабленной семантикой доступа решают поставленную проблему следующим образом: создается количество структур большее или равное количеству потоков, при изменении или чтении данных выбирается случайный поток, который в данный момент не заблокирован, и выполняется действие.

В статье с описанием SprayList [1], предлагается улучшение структуры SkipList [2]. В отличие от SkipList, SprayList предполагает не линейный поиск сверху вниз и из начала в конец, переходя по одной ссылке связанного списка, а изначальное перемещение случайным образом по вертикальным уровням и далее уже поиск элемента списка со случайным смещением по горизонтальному уровню. Если поиск не дал результатов или элемент оказался заблокированным другим потоком, алгоритм начинается с начала. После нахождения нужного элемента операции со списком выполняются также, как и в SkipList.

В работе [3] был разработан алгоритм операции удаления элемента с минимальным ключом из очереди с приоритетами и выдачи его в качестве результата функции. Суть данного подхода в том, что на каждый поток необходимо иметь больше, чем одну очередь. Операция вставка элемента осуществляется в случайную незаблокированную очередь. Операция удаление элемента с минимальным ключом осуществляется следующим образом: выбираются две случайные очереди, у них находятся элементы с минимальными ключами и сравниваются между собой, после нахождения минимального, этот элемент удаляется из очереди с наименьшим из найденных ключом. Как можно было заметить этот элемент не всегда будет минимальным из вставленных, однако принимается, что он довольно близок к минимальному и данной погрешностью можно пренебречь.

В статье [4] была разработана k-LSM (Log-Structured Merge tree) структура, входящая в подкласс структур с ослабленной семантикой выполнения операций. В качестве базовой структуры используется LSM дерево. Оно представляет собой указатели на сортированные массивы, называемые блоками, где каждый блок находится на определенном уровне L дерева и может содержать N элементов ($2^{L-1} < N \leq 2^{L}$). Структура k-LSM состоит из двух других: общей k-LSM и распределенной LSM очередей с приоритетом. Общая k-LSM очередь представляет собой массив указателей на очереди с приоритетами, сортированные по уровню. Все потоки могут обращаться к данной структуре по единому указателю. У общей k-LSM структуры существует два узких места: а) операция вставки элемента, однако обычно используется массовая вставка, таким образом сокращается количество обращений к данной структуре, но возрастает средний размер блока и уменьшается количество операций слияния, б) операция удаления минимального элемента, также вызывает блокировку, однако в данной статье предполагается, использование ослабленной семантики доступа, таким образом из общей k-LSM структуры удаляется не элемент с самым минимальным ключом, а минимальный из k+1 случайно выбранных ключей. Распределенная LSM разрабатывалась на основе идеи work-stealing [5], каждый поток имеет собственную очередь приоритетов и работает исключительно с ней, но, если очередь пуста, а требуется операция отличительная от операции вставки, начинается попытка доступа к чужим очередям с приоритетами и, если они не заблокированы, выполняется операция с ними. Таким образом, объединив общую k-LSM и распределенную LSM структуры, была получена k-LSM структура. При операции вставки, поток сохраняет элемент в собственной распределенной LSM структуре, если размер данной структуры превышает заданный, то данная распределенная LSM осуществляет слияние с общей k-LSM структурой. При операции удаления используется поиск наименьшего ключа в собственной и общей LSM структурах и оба элемента удаляются, если данные структуры пусты, то осуществляется поиск среди структур у других потоков.

ЛИТЕРАТУРА

1. Д. Алистер [D. Alistarh], Дж. Копински [J. Kopinsky], Дж. Ли [J. Li], Н. Шэвит [N. Shavit] The SprayList: A Scalable Relaxed Priority Queue [Электронный ресурс] // URL: http://www.mit.edu/~jerryzli/SprayList-CR.pdf

2. В. Пуг [W. Pugh] Skip Lists: A Probabilistic Alternative to Balanced Trees [Электронный ресурс] //URL:http://epaperpress.com/sortsearch/download/skiplist.pdf

3. Х. Рихани [H. Rihani], П. Сандерс [P. Sanders], Р. Дементьев [R. Dementiev] MultiQueues: Simpler, Faster, and Better Relaxed Concurrent Priority Queues [Электронный ресурс] //URL: https://arxiv.org/pdf/1411.1209.pdf

4. М. Виммер [M. Wimmer], Дж. Грабер [J. Gruber], Дж. Л. Траф [J. L. Traf], Ф. Тсигас [P. Tsigas] The Lockfree k-LSM Relaxed Priority Queue [Электронный ресурс] // URL: https://arxiv.org/pdf/1411.1209.pdf

5. Р. Блюмо [R. Blumofe], Ч. Лейзерсон [C. Leiserson] Scheduling Multithreaded Computations by Work Stealing [Электронный ресурс] // URL: http://supertech.csail.mit.edu/papers/steal.pdf