Краткое описание работы массивов JUDY

Doug Baskins, doug@sourcejudy.com

16 октября 2001 г., изменено в июле 2002

Разработчика алгоритма Judy часто спрашивают, почему этот алгоритм такой быстрый. Здесь предпринята попытка кратко ответить на этот вопрос. Более полное описание пакета приведено в документе Judy Shop Manual.

Исходный код Judy был открыт в июне 2002 г. и доступен по ссылке http://sourceforge.net/projects/judy[1].

Дерево Judy в общем случае быстрее и требует меньше памяти, нежели такие варианты как бинарные деревья (AVL), b-деревья и skip-списки. При использовании конфигурации Judy Scalable Hashing пакет обычно быстрее других методов хэширования.

Термины пространство (expanse), численность (population) и плотность (density) нечасто применяются в литературе о поиске по дереву, поэтому ниже даны краткие определения.

Пространство
Диапазон возможных ключей (например, 256 - 511).
Численность
Число ключей в пространстве (например, для 260, 300, 499, 500 это будет 4).
Плотность
Описывает степень разреженности пространства ключей (density = population/expanse). Плотность 1,0 означает, что все возможные ключи установлены или действительный в данном пространстве.
Термины узел (node) и ветвь (branch) в этом документе взаимозаменяемы.

Термины ключ (key) и индекс (index) взаимозаменяемы в документе. Дерево Judy рассматривается как неограниченный массив Judy на уровне интерфейса API. Пространства массивов JudyL и Judy1 ограничены размером слова (32 или 64 бита), применяемого в качестве индекса. Массив JudySL ограничен лишь размером строки ключа, которая может быть сохранена в машине.

Заполнение строки кэша CPU - это дополнительное время, требуемое для чтения ссылки из ОЗУ (RAM) если слово не найдено в кэше. В современных компьютерах это время составляет 50 - 2000 машинных команд[2]. Поэтому такого заполнения следует избегать, если ту же работу можно выполнить менее чем за 50 команд.

Некоторые причины превосходства Judy по сравнению с двоичными деревьями, b-tree и skip-списками указаны ниже.

- Judy редко отдаёт предпочтение простоте по отношению к скорости и пространству (алгоритм Judy прост лишь на уровне API).
- Judy по возможности избегает заполнения строк кэша (основной критерий устройства Judy).
- Для b-tree требуется поиск в кажом узле (ветви), что ведет к заполнению большого числа строк кэша.
- Двоичные деревья имеют значительно больше уровней (примерно в 8 раз), что увеличивает число заполнений строк кэша.
- Skip-списки приблизительно эквивалентны дереву степени 4, что увеличивает число заполнений строк кэша.
- Основанное на пространстве цифровое дерево (Judy является его вариантом) не требует балансировки при расширении.
- Часть ключа (8 битов) служит для деления пространства на субдеревья, а в этих субдеревьях требуется лишь оставшаяся часть ключа , что сокращает размер.

Основным недостатком простых деревьев является неэффективное использование памяти, особенно при больших значениях степени втевления узлов N. Дерево Judy решает эту проблему. Фактически, это дерево превосходит по эффективности использования памяти почти все другие структуры (включая простые связные списки). Исключением может являться плотно заполненный линейный массив (array[]).

С точки зрения скорости Judy представляет собой дерево степени 256 (trie) в соответствии с определением D. Knuth (том 3). Число 256 для степени дерева является «магическим» по ряду причин. Основная причина заключается в том, что байт (минимальный адресуемый элемент памяти) имеет размер 8 битов. Высокая степень также ведет к снижению числа заполнений строк кэша на доступ.

Интересно отметить, что в ранней версии Judy использовалось расширение ветвей (иногда это называют level-compressed trie). Возможности расширения возникают главным образом на верхних уровнях дерева. Поскольку дерево представляет собой иерархию, верхние ветви с большой вероятностью присутствуют в кэше, поэтому их расширение не снижает существенно число заполнений строк кэша. Позднее расширение ветвей было исключено из Judy. Однако пакет Judy был настроен на использование наименьшего возможного числа инструкций, когда доступ вероятно будет к кэшу.

Наличие кэше CPU в современных машинах изменило многие алгоритмы. Для использования преимуществ кэша важно применять его как можно чаще. В дереве Judy наличие кэша снижает число заполнений строк кэша в 1-3 раза (или более) на поиск.

В пространстве 2^{32} (или 256^4) требуется не более 4 заполнений строк кэша для наихудшего случая доступа к плотно заполненному дереву степени 256. В пространстве 2^{64} (или 256^8) требуется 8 заполнений в наихудшем случае. На практике Judy обычно делает это гораздо эффективней. Одной из причин этого является то, что «плотность» ключей

[1]Этот вариант кода не поддерживает некоторые современные процессоры. Более современный вариант доступен по ссылке. *Прим. перев.*
[2]Современные машины часто используют конвейерную запись в ОЗУ и это не вносит дополнительной задержки в Judy.

редко бывает минимально возможнойй в большинстве подпространств. Для увеличения глубины дерева Judy требуется высокая плотность в сочетании с большой численностью. Объяснять это долго. Можно провести аналогию с кучей песка. Потребуется много песка для создания высокой кучи, особенно если под каждой песчинкой должно размещаться 256 других. В 64-битовом варианте Judy для создания 8 уровней может потребоваться больше памяти, нежели имеется на планете.

Judy эффективно приспосабливается к разным уровням плотности и численности. Поскольку структурой данных Judy служит дерево деревьев, каждое субдерево является статическим пространством, которое оптимизировано в соответствии с плотностью содержащихся ключей. Для поддержки такой гибкости в 32(64)-битовом Judy имеется приблизительно 25(85) первичных и примерно столько же вторичных структур данных. Здесь описаны некоторые из них для демонстрации связи (синергии) плотности и сжатия.

С точки зрения расхода памяти (размера) дерево Judy совместно использует (без дублирования) цифровой ключ в дереве. Эта форма сжатия ключа является естественным результатом применения цифрового дерева. Было бы очень неудобно делать на это деревьях, сбалансированных по численности (и, по-видимому, это не было сделано). Каждый указатель в дереве Judy ссылается на похожее меньшее субпространство, в то же время декодируя другие 8 битов ключа (в чистом цифровом дереве ключи не хранятся внутри и определяются по положению).

Рассмотрим верхнюю часть небольшого дерева Judy (JudyL) и нижнюю часть плотно заполненного дерева Judy1. Дерево Judy с численностью 0 является просто указателем NULL. Дерево JudyL с одним ключом является корневым указателем на объект из двух слов, содержащий ключ и связанное с ним значение. Дерево JudyL с численностью два ключа является объектом из 4 слов (2 значения и 2 сортированных ключа. Дерево с численностью 3 ключа является объектом из 8 слов с (число слов + 3) значений и тремя сортированными ключами.

Это можно продолжать до численности в 32 ключа, когда формируется фактическая древовидная структура со «сжатым» узлом степени 256, которая декодирует первый байт каждого ключа. Значение 32 было выбрано потому, что именно здесь структура дерева требует эквивалентного числа заполнений строк кэша. Все объекты ниже этого узла содержат ключи, которые сокращаются по меньшей мере на первый байт.

Имеется три типа узлов, два из которых связаны с 1 заполнением строки кэша при прохождении и один - с 2. На каждом пути вниз[1] по дереву при любой численности проходится не более 1 объекта с заполнением двух строк кэша. Это означает, что иногда может быть дополнительное заполнение одной строки кэша по сравнению с ожидаемым при прохождении чистого узла степени 256.

С другой стороны, густонаселенное дерево Judy1, в котором ключ декодирован на 1 байт и плотность подпространства ключей степени 256 выше 0,094 (25 ключей/пространство 256), битовая последовательность из 32 байтов (256 битов) формируется из имеющегося сортированного массива из 24 1-байтовых ключей (обработка значений опущена). В результате для ключа используется около 1,3 (32/25) байта памяти (по сравнению с 1,0). Отметим, что рост плотности (численности) на этом этапе не требует увеличения памяти для ключей. Например, при достижении плотности 0,5 (численность = 128 / пространство = 256) потребляется около 2 битов ((32/128)*8) на ключ с добавлением некоторых издержек (2,0 слова) для структуры дерева.

Отметим, что вставка или удаление ключа почти так же просты, как установки или сброс бита. Кроме того, расход памяти почти одинаков для 32- и 64-битовых деревьев Judy. При одинаковом размере ключей 32- и 64-битовые деревья Judy очень похожи по расходу памяти на ключ, глубине и производительности. Однако расход памяти в 64-битовом Judy выше, поскольку размер указателей и значений (JudyL) удваивается.

В короткой заметке невозможно описать такие детали структур данных как Root, JPM, указатели, линейные, битовые и несжатые узлы, области значений, непосредственные индексы, терминальные узлы (листья), менее сжатая форма, управления памятью, быстрый поиск листьев и подсчет деревьев. Однако базовые понятия здесь рассмотрены, а остальное можно узнать из документации.

Doug Baskins
doug@sourcejudy.com

Вольный перевод на русский язык

Николай Малых

nmalykh@protokols.ru

[1]Американские деревья растут корнем вверх. *Прим. перев.*