InterBase vs Firebird?

kdv, www.ibase.ru, 15.02.2019

На статью Embarcadero "Comparing InterBase and Firebird" меня «натолкнули» на sql.ru. Так бы я ее никогда не прочитал. Посмотрев, заинтересовался графиками. Уж очень странно они выглядели. Поскольку я очень давно работаю и с InterBase и с Firebird, разумеется, производил разные тесты. Например, тест скорости бэкапа, рестора, и так далее.
В некоторых InterBase был быстрее Firebird, в других – наоборот, но вот прямо ужасно сильных перекосов по скорости как-то не замечал.
А тут – 5кратная разница по скорости select count, и так далее.
Но, начнем по порядку. В обсуждаемой статье вначале идет матрица по отличиям в функциональности. Конечно, разработчики InterBase сосредоточены на «бизнес-функциях», а разработчики Firebird – на пожеланиях пользователей. Понятно, что в статье Embarcadero эти самые бизнес-функции будут выпячены. Но, для сравнения можно посмотреть наш перечень различий:
https://www.ibase.ru/ib_fb_difference/
Большинство пунктов – отличия в SQL, но разве не это главное для разработчиков приложений?
А журнал в InterBase, и тому подобное – что-ж, я рапортовал о нескольких багах в XE7, которые не просто баги, а шоу-стопперы. Баги эти не исправлены в последнем Update 6 (и до сих пор отмечены как неисправленные – IBP-34, IBP-37), а версию InterBase 2017 я на эти баги не проверял.

Дальше – сравнение скорости. Первым идет график многопользовательской работы в тесте TPC-C. Я не очень люблю многопользовательские тесты, если вдруг этот тест повторю, то дополню статью результатами, но сам тест выполнен с журналом и Forced Writes OFF на InterBase, и с Forced Writes ON на Firebird. А это, извините, нечестная игра. Журнал работает совсем не так как Forced Writes ON, поэтому даже если бы при FW ON оба сервера показывали одинаковые результаты, режим с журналом и FW OFF будет однозначно быстрее.

Ладно. Переходим к следующему сравнению – select count по трём таблицам. Тут меня задело. Если в тесте выше даже изображенной разнице по скорости можно было бы поверить, т.к. это запись, то как можно было получить 5 и более раз медленнее простое чтение на Firebird?

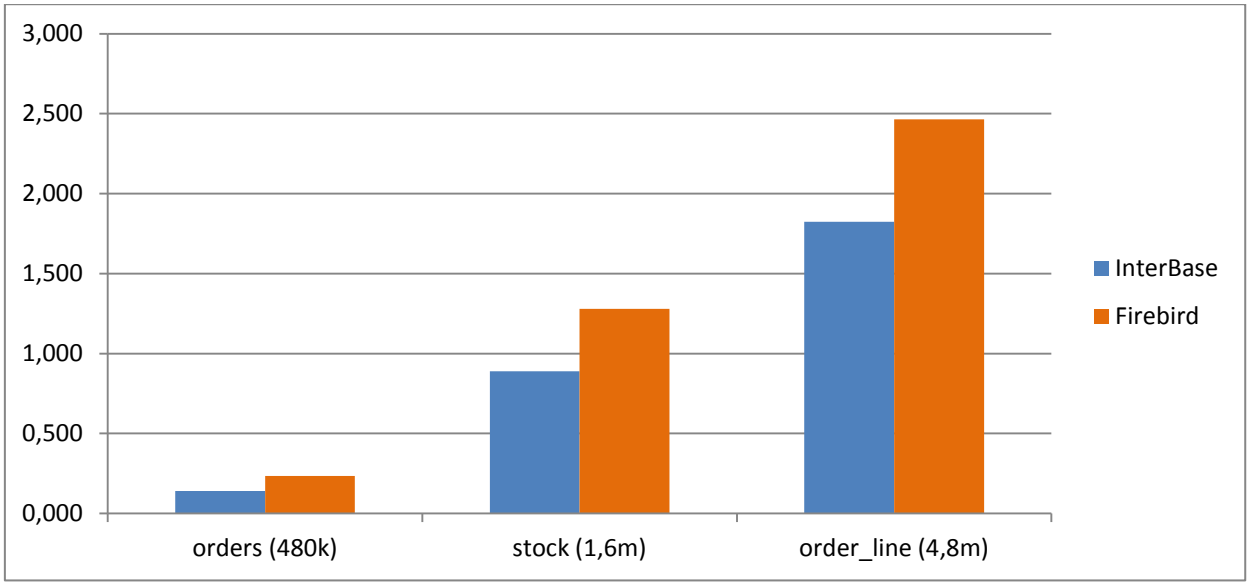
И я засел за тест. Создал базу TPC-C с 300 warehouses, чтобы база получилась около 30 гигабайт –у меня на компьютере сейчас 16 гиг памяти, поэтому нужна база, которая однозначно не влезла бы в файловый кэш операционной системы (т.е. при 16 гиг ОЗУ гарантированно больше 8 гиг).
На 30 гигабайтной базе данных сразу получил крайне странные результаты, не в пользу InterBase. Решил их понимание отложить на потом, и создал базу с 16 warehouses, которая и использовалась в статье Embarcadero – по такой базе select count(*) from orders выдает 480к, stock – 1,6 млн, order_line – 4802711, то есть, всё как в статье.

Чтобы всё было честно, я одни и те же запросы прогонял в разных режимах – с кэшем 2048, 20480, 100000 страниц, и для каждого размера кэша в режиме direct io, т.е. без файлового кэша операционной системы. В итоге получилось по 6 (шесть) измерений для каждого запроса, причем для двух баз данных – первая с 16 warehouses, 1.4 гиг, вторая 300 warehouses, 27 гиг.

Сравнивал я только SuperServer, т.к. у InterBase нет ничего другого, а у Firebird на чтение в однопользовательском режиме архитектуры SuperServer, SuperClassic и Classic выдают почти одинаковые результаты (см. тест order vs sort, страница 10).

В тесте участвуют только **InterBase 2017 update 3** и **Firebird 3.0.4**.
Все запросы выполнялись по 2-3-4 раза для заполнения кэша СУБД и файлового кэша ОС, результат брался самый последний (самый быстрый).

Итак, первый результат с включенным файловым кэшем, и кэшем IB/FB по умолчанию (2048 страниц), цветом столбцы помечены так же, как и в статье Embarcadero. Меньше – лучше.



Сравниваем с данными из статьи Embarcadero.

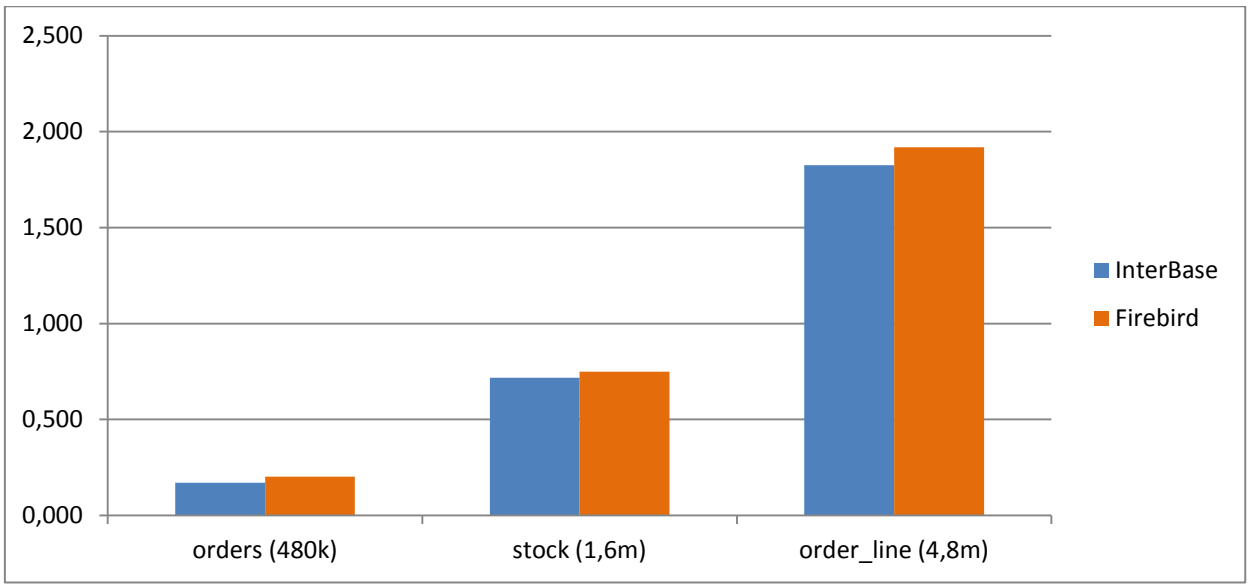
Да, Firebird здесь медленнее, на order_line почти на 30%. Но напомню, что конфиг у обоих по умолчанию, что редко бывает на промышленных серверах.

Во-первых, смущает общая разница времени выполнения самого долгого запроса. У Embarcadero это даже для InterBase 3 секунды для order_line, а у меня – 1.8 секунды. У них такие медленные диски? (мои базы лежат на обычном hdd sata 3).

Во-вторых, никакой 5кратной, и даже 2кратной разницы не наблюдается.

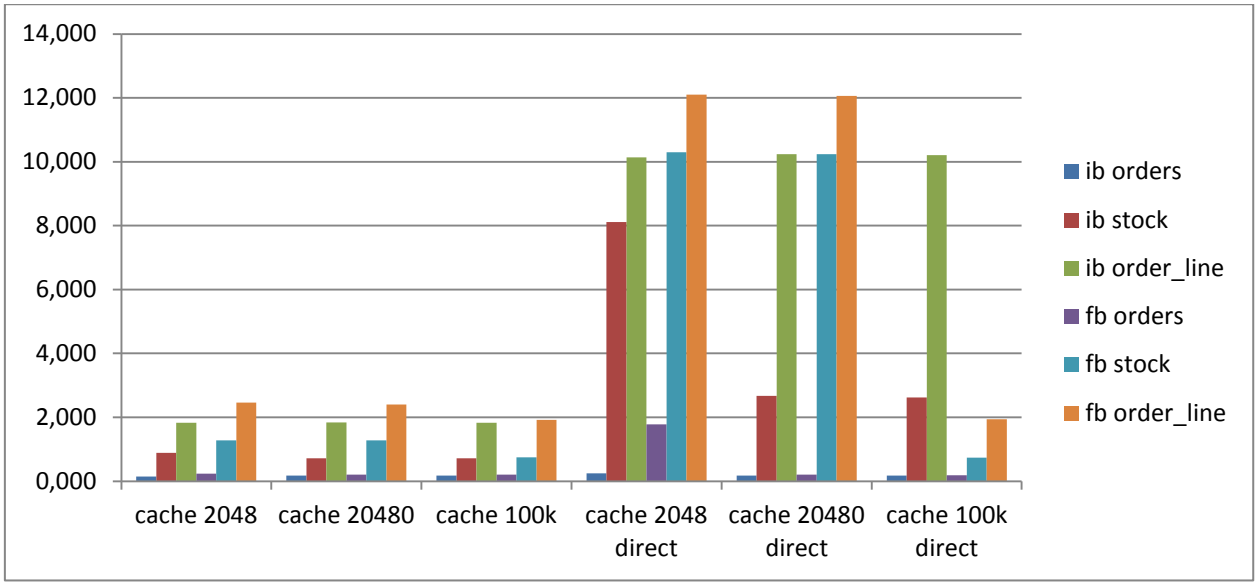
В третьих, полученные мной результаты для IB и FB по пропорциям совпадают с данными IB в статье Embarcadero. А значит, у них для Firebird данные получены «как-то не так».

Давайте теперь посмотрим на те же запросы, только с кэшем 100к страниц:



Разница между IB и FB почти пропала, на самом долгом запросе составляет не более 5%.
То есть, при нормальных настройках в однопользовательском режиме по чтению IB и FB одинаковы.

Теперь посмотрим объединенный график – кэши 2048, 20480 и 100000 страниц, и вариант с выключенным файловым кэшем.



В левой части 3 группы столбцов – с файловым кэшем. Что интересно, на кэше в 100к результаты стали одинаковы на всех запросах и у Interbase, и у Firebird.
С выключенным файловым кэшем результаты отличаются.
Напомню, что файловый кэш для баз данных отключается

- у InterBase – для конкретной базы данных, gfix –write direct dbfilename
- у Firebird – FileSystemCacheThreshold = 0 или меньше чем DefaultDbCachePages, в firebird.conf для всех баз данных, или в databases.conf для конкретной БД

Итак, с кэшем 2048 страниц и без файлового кэша у InterBase с таблицей stock случился конкретный просад – select count выполнялся аж 8 секунд. С большим кэшем стало получше, 2,6 секунд.

У Firebird наоборот – при кэше меньше чем количество считываемых запросами страниц запросы выполняются долго, но как только всё помещается в кэш СУБД, результат становится идентичным InterBase с включенным файловым кэшем. У InterBase это не так. Такое впечатление, что кэш СУБД сработал только для таблицы stock. А для таблицы order_line изменения кэша не повлияли никак.

Теперь переходим **к базе размером 27 гигабайт**.

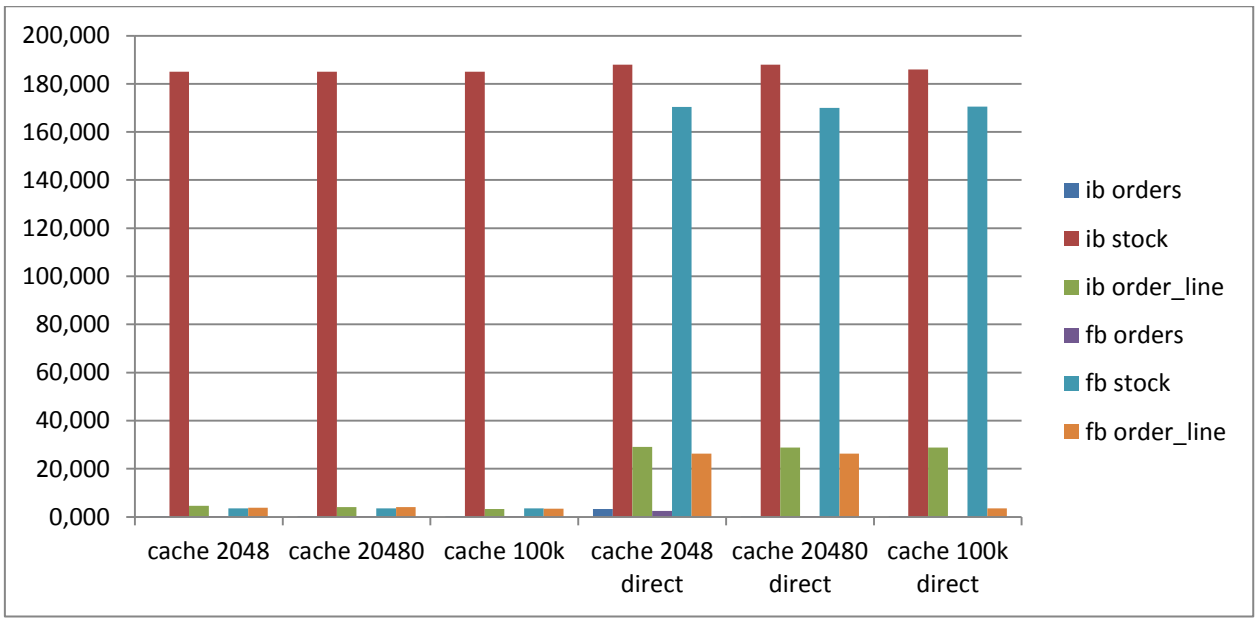
Чтобы выбирать то же самое количество записей, к запросу пришлось добавить условие where, например,
select count(*) from orders o
where o.o_w_id < 17

В этом случае запрос вместо PLAN NATURAL использует выборку по индексу, т.е. PLAN TABLE INDEX INDEXNAME.
Напомню, происходит это следующим образом – вначале идет скан индекса для поиска нужного диапазона ключей, затем из полученных номеров записей формируется битовая маска, и далее осуществляется проход по записям и подсчет их количества.
Особенность этого метода, в отличие от table order index – повторного обращения к страницам данных не происходит.

Сразу приведу общий график

У InterBase с запросом по таблице stock творится что-то неладное. При любом размере кэша, и с включенным или выключенным кэшем файловой системы результат абсолютно одинаков - ~185 секунд. Даже у Firebird при выключенном файловом кэше (только) результат получше – 170 секунд.
Также видно что и с order_line без файлового кэша у InterBase не все хорошо. Впрочем, это было видно и на предыдущем графике базы в 1.4 гигабайта.

Вот график с развернутыми на 90 градусов данными



Сначала я заметил существенную разницу в интенсивности работы с диском при выполнении запроса к stock. У InterBase любой повторный запрос читал базу данных со скоростью 80 мегабайт в секунду, при этом образовывая очередь диска равную 4. У Firebird скорость чтения БД была на уровне 20 мегабайт в секунду, с очередью диска чуть выше 1 (1,14).
Тут надо вспомнить, что в InterBase XE7 Update 2 появился новый параметр конфига PREDICTIVE_IO_PAGES (которого так и нет в дефолтном ibconfig). Это так называемый «префетч», 0 – выключено, от 1 до 64 включено, по умолчанию 64.
Я попробовал включать и выключать этот параметр (с рестартом ibserver, разумеется), но толку от этого не было никакого.
Такое впечатление, что при старте запроса InterBase начинает сканировать всю таблицу в 4 потока, каждый со скоростью 20мб сек (в результате наблюдаем 80мб сек), но поскольку результат возвращает только один поток, длится все это столько же, сколько у Firebird. Чем занимаются дополнительные три потока – совершенно непонятно.
Да, при включенном файловом кэша по RAMMAP видно, что InterBase «загрузил» в файловый кэш ОС около 7 гигабайт базы данных. Таблица stock в этой БД занимает 9.7 гигабайт. То есть, вместо чтения 5% размера таблицы читается почти вся таблица. Зачем? И почему этого не происходит с таблицей order_line?
У Firebird со stock таких проблем нет, разве что при выключенном файловом кэше.
Что интересно – при любом размере кэша запрос к stock всегда показывает
Reads from disk to cache = 132 130
Writes from cache to disk = 1
Fetches from cache = 1 731 947

То есть, кэш СУБД никак в данном случае не помогает. Для базы в 1.5 гиг кэш в 100к страниц вполне вмещал в себя читаемые данные, там их было не более 34 тысяч страниц (для любого запроса). Поэтому повторное выполнение запроса приводит к обращениям только к кэшу, и запрос выполняется намного быстрее.
А вот если количество читаемых страниц больше кэша – увы. При данном типе запроса, как уже было сказано выше, страницы читаются последовательно (сначала индекса потом данных), и к ним нет повторных обращений. Поэтому при любом размере кэша меньше читаемого количества страниц они всегда будут снова и снова считываться с диска.

Например, у Firebird страницы запроса к order_line «влезли» в кэш 100к, поэтому даже при выключенном файловом кэше он выполнился за 3,5 секунды. У InterBase чуда почему-то не произошло, запрос к order_line всегда выполнялся 29 секунд (без файлового кэша ОС).

Я даже подумал, что в странностях со stock виновата какая-то внутренняя структура, например warehouses перемешаны, или еще что. Но заливка базы данных производилась абсолютно одинаковым образом, и заполняемость страниц stock, а также общий размер, почти идентичны (93%, и 9773,19 мб у IB и 9773,25 мб у FB).
Firebird при запросе к stock «закачивает» в файловый кэш ОС кусок базы размером 2.4 гигабайта (132130 страниц по 16к это как раз 2.1 гигабайта). Отсюда можно сделать вывод, что да, данные одинаковых warehouses размазаны по страницам данных, но опять же, почему InterBase при таком же запросе грузит в файловый кэш аж 7 гигабайт базы данных?

В общем, перехожу к суммированию всех странностей и не странностей.

Итого

1. При умолчательных конфигах производительность InterBase и Firebird по чтению данных в мелкой базе (1.5 гиг) отличается не более 30% (Firebird медленнее). В статье данные кривые, такое впечатление, что для Firebird приведены данные по чтению без использования файлового кэша.
2. Если кэш суперсервера настроить, то разница в производительности не более 5% (Firebird чуть-чуть отстает)
3. На большой базе данных у Interbase видно в 4 раза большую интенсивность работы с диском при почти одинаковых результатах с Firebird.
4. InterBase сильнее «потребляет» файловый кэш ОС.
5. У InterBase на данных запросах predictive io никак себя не проявляет.
6. На мелких базах у InterBase кэш работает, на больших – почему-то нет (см. ib cache 100k для order_line)
7. У InterBase из-за predictive io (наверное) невозможно понять, сколько реально страниц запрос читает (или не читает) с диска. Например, для запроса к stock на большой БД reads from disk to cache вместо 132130 показывает 1306. Что это за значение – вообще непонятно (и так – с любыми запросами к любой базе). То есть, если вы захотите настроить кэш, то делать это придется «на глаз».
8. У InterBase на большой базе и злосчастной stock нет разницы между выключенным и включенным файловым кэшем. Это, конечно, катастрофа. У Firebird первый («холодный») старт запроса к stock занимает даже меньше времени (2мин 50 сек вместо 3 мин 5 сек).