ОБСУЖДЕНИЕ

Перспективная стратегия «множество запросов на один узел кластера» с репликацией БД по отдельным узлам ориентирована на работу с консервативными базами данных умеренных объемов, не превышающих объем оперативной памяти узла. Как показано в докладе, переход к GPU-кластерам с использованием регулярного плана обработки запросов и адаптацией инструментальных СУБД к такой платформе в этом случае неконкурентоспособен.Целесообразна разработка специализированных СУБД с оптимизацией запросов, ориентированной на использование графических ускорителей [12].

При использовании обычных вычислительных кластеров, организация обработки запросов к консервативным БД объемами до нескольких ТВ по регулярному плану – пока что объективная закономерность. Это обусловлено необходимостью хеширования таких БД на множестве узлов кластера.Их репликация по узлам приведет к чрезмерным задержками по съему данных с внешних носителей и возможным трудностям размещения обрабатываемых отношений в оперативной памяти узла. Столь же обосновано и динамическое сегментирование промежуточных и временных отношений. Следствием указанных обстоятельств оказывается «квазипараболическая» зависимость времени обработки ПТ от числа узлов (для вычислительных кластеров более характерно приближение к гиперболе). При этом максимум производительности на грани масштабируемости оказывается достаточно низким. Такова в данном случае плата за работу с базами данных больших объемов.Переход к мультикластеризации при ограничении числа узлов в монокластерах-компонентах значительно улучшает положение, приближая указанную зависимость к гиперболе.

В работе показано, что серьезное повышение эффективности при работе с базами данных сравнительно небольших объемов должно иметь место при переходе на платформу GPU-кластера при условии хранения распределенной БД в сжатом виде в оперативной памяти IO-узлов и организации обработки в JOIN-узлах по схеме «запрос на ядро». В IO-узлах GPU-акселерация используется для реализации операции *«select».* Реализация по такому принципу GPU-СУБД объемами до нескольких ТВ требует построения асимметричных кластеров со сравнительно малым числом IO-узлов, достаточно большим числом JOIN-узлов и скоростной сетью передачи данных. Но имеется ограничительное условие: *суммарный объем промежуточных отношений для любого запроса действующего потока не должен превышать объема оперативной памяти узла*.

Представляет интерес вопрос: насколько все же оправдано использование в принятой натурной модели GPU-узла для реализации *«select – project»*-обработки, и нельзя ли в данном случае применить схему «запрос на ядро» с хранением несжатой БД в оперативной памяти узла? Поскольку по условию ее размеры не позволяют загрузку данных объемом >100GB, придется БД хешировать по двум узлам, так что в блоке JOJN останется 4 узла. Но это не повлияет на значение $(t_{\Sigma join})' = 1{,}61$час для ПТ. Время параллельного функционирования блока IO определится величиной $t_{\Sigma\sigma,\pi}$ для самого «тяжелого» на ПТ запроса. Согласно табл.3, таковым вновь является запрос №9. Для него $t_{\Sigma\sigma,\pi} = 11{,}355\text{сек}\cdot 100 = 1135{,}5\text{сек} \approx 0{,}32\text{час}$, что сравнимо с полученным для одного GPU-узла значением 0,18час.

Отсюда следует: максимум, что может дать GPU-акселерация для натурной модели при действии непрерывного потока запросов, случайно формируемых на множестве запросов ПТ,это – повышение производительности примерно на 25% за счет ввода в действие пятого узла IOIN. Так что *наш ответ на поставленный в заголовке доклада вопрос скорее отрицательный, нежели положительный.* Экспериментальное подтверждение справедливости такого вывода связывается с разработкой и исследованием натурной модели СУБД *Clusterix-G*.

Как бы то ни было, но *последний рассмотренный вариант представляет принципиальное изменение методологии Clusterix*. Суть этого изменения состоит в отказе от распределенной обработки запроса на уровне JOIN и от динамического сегментирования промежуточных и временных отношений, что существенно облегчает разработку СУБД в целом и повышает ее эффективность. Но нельзя забывать о последнем сформулированном условии. Гарантировать его выполнение в общем случае невозможно (пример запроса №4). *Снятие этого условия означает возврат к методологии Clusterix-M*.

Выполненный анализ в целом имел целью обсуждение GPU-принципов высокоэффективного построения консервативных СУБД объемом ≥ 100GB довольно скромными средствами, доступными в настоящее время широкому кругу научно-образовательных организаций.

Использование перспективных,но довольно дорогих технологий с объемами оперативной памяти до 2TB и значительным числом ядер (не менее 18) в узле делает подход [22] недосягаемым по эффективности другими известными методами построения большеобъемных консервативных СУБД кластерного типа.

Во всяком случае, экспериментальных данных, опровергающих это утверждение, пока не имеется.