© 2012 г. А.П. БЕЛЬТЮКОВ, д-р физ.-мат. наук
(Удмуртский государственный университет, г. Ижевск),

БЕЗТРАНЗАКЦИОННЫЕ БАЗЫ ДАННЫХ ДЛЯ РАСПРЕДЕЛЁННЫХ ВЫЧИСЛЕНИЙ

Предлагается альтернативный подход к организации данных в распределённых вычислительных системах. При этом подходе исключается какое-либо согласование действий в разных узлах. Любая часть системы может "забежать вперёд" или "отстать" как угодно и при этом не возникнет никаких задержек других частей. Как только все действия будут выполнены, результирующие данные будут доступны для потребителя немедленно.

TRANSACTION-FREE DAT ABASES F OR DISTRIBUTED CO MPUTATIONS / A.P. Beltiukov, (Udmurt State University, Universitetskaya, 1, Izhevsk 426004, Russia, E-mail: belt@udsu.ru). An alternative approach to a data structure in the distributed computing systems is offered. At this approach any coordination of actions in different nodes is excluded. Any part of system can "run forward" or "lag behind" somehow and there will be no delays of other parts. As soon as all actions will be executed, resultant data will be available to the consumer immediately.

1. Введение

При организации вычислений с данными, распределёнными по обширной вычислительной сети, часто возникают проблемы синхронизации действий, выполняемых в нескольких узлах сети согласованно. При этом данные, изменяющиеся в разных узлах, не могут быть использованы другими узлами до тех пор, пока их согласованное изменение не будет произведено. Это приводит к понятию распределённой транзакции. Поддержка распределённых транзакций может вести к существенному замедлению работы сети и дополнительным накладным расходам.

В настоящей работе предлагается альтернативный подход к организации данных в распределённых вычислительных системах. При этом подходе вообще исключается какое-либо согласование действий в разных узлах. Любая часть системы может "забежать вперёд" или "отстать" как угодно и при этом не возникнет никаких задержек других частей. Как только все действия будут выполнены, результирующие данные будут доступны для потребителя немедленно.

Этот подход позволяет осуществлять инкрементную работу сети, при которой любое незначительное изменение какого-либо входного информационного массива приводит к автоматическому распространению изменений на все зависимые массивы.

Для решения поставленной задачи предлагается следующий математический подход. Во-первых, все обрабатываемые данные представляются в виде функций, отображающих некоторые конечные множества ключей в элементы абелевых групп. На пространствах ключей могут быть заданы линейные порядки. Вместо абелевых групп иногда

можно брать коммутативные моноиды. Считается, что за пределами областей определения рассматриваемые функции равны нулям соответствующих абелевых групп.

Рассматриваемые исходные абелевы группы можно считать простыми (начальными) типами данных. На этих типах данных определены некоторые гомоморфизмы - встроенные операции. Операции могут быть определены и на парах типов данных - гомоморфизмы, заданные на соответствующих прямых произведениях абелевых групп. Задаются и встроенные операции на ключах и кортежах ключей - функции, отображающие эти ключи и кортежи также в элементы некоторых абелевых групп. Пространства ключей - также исходные типы данных. Упомянутые два класса типов могут пересекаться. Роль логического типа играет абелева группа с двумя элементами. Роль условного оператора – тривиальное умножение на элемент этой группы: умножение на ноль даёт ноль, при умножении на единицу сохраняется исходный элемент. Разные ветви условия просто складываются.

Вводятся операции на полученной алгебре хранимых функций, аналогичные операциям реляционной алгебры. Таким образом, получается некоторый новый язык программирования. Можно доказать, что полученный язык при некоторых естественных условиях универсален.

При реализации системы разные хранимые функции и даже разные части одной и той же функции могут размещаться в разных узлах вычислительной сети. Задачи для такой системы, написанные на упомянутом языке, программирования компилируются в задания, рассылаемые узлам системы для выполнения. Имеется механизм, позволяющий оценить сложность функционирования системы в зависимости от задачи, записанной в исходной программе.

Работа развивает идеи автора из статьи [1].

2. Модель данных

Предлагаемая модель данных содержит два класса типов данных: типы *ключей* и типы *значений.* Считаем, что все данные могут быть закодированы двоичными словами. Типы ключей могут быть линейно упорядочены. Типы значений являются абелевыми группами с определёнными на них операциями сложения. Можно обобщить рассмотрение, допустив в качестве типов значений коммутативные моноиды. Предполагается, что на типах данных заданы некоторые исходные эффективно вычислимые операции. Виды этих операций перечислены ниже.

Прежде всего, на ключах можно задавать исходные вычислимые функции и предикаты. В частности, на типе ключа может быть задан некоторый стандартный линейный порядок. На значениях могут быть заданы *внешние* (и *внутренние*) произведения, обладающие дистрибутивностями относительно сложений:

$$(x+y) \circ z = x \circ z + y \circ z, \quad x \circ (y+z) = x \circ y + x \circ z, \; 0 \circ y = x \circ 0 = 0,$$

где $\circ$ – операция внешнего умножения, + – операции сложений в различных группах, 0 – нули абелевых групп. В дальнейшем обозначения операций умножения (при их единственности для данных типов значений) будут опускаться. Могут быть заданы также операции-гомомомфизмы, отображающие абелевы группы друг в друга и в себя. Можно обобщить понятие внешнего произведения и на многоместные операции. Ассоциативность умножений не требуется. Не требуется и наличие единиц. Роль логического типа играет абелева группа с двумя элементами: 0 и 1. Предикаты считаются функциями со значениями этого типа. Конъюнкция – умножение, другая исходная логическая операция – сложение (по модулю 2).

Хранимыми данными (структурами данных) являются функции, заданные на кортежах ключей и принимающие *значения* из абелевых групп. При этом функция должна иметь конечную область ненулевых значений. На самом деле, только эти ненулевые значения и хранятся. Хранимая структура имеет тип, определяемый числом и типами ключей, а также типом значений.

3. Определения вычисляемых структур

В предлагаемой модели баз данных будут различаться *входные* структуры, значения которых поступают из «внешнего мира» и *вычисляемые* структуры, значения которых зависят от входных структур и вычисляются в соответствии с заданными вычислительными формулами. Некоторые из вычисляемых структур могут оказаться выходными, поставляющими значения во «внешний мир».

Перечислим виды вычислительных формул. На самом деле, при создании практического языка программирования эти формулы могут не задаваться в указанном ниже виде, а порождаться из других более удобных для рассматриваемой предметной области конструкций. Все формулы задают значения функций (структур данных) для указанных далее аргументов. Используются следующие обозначения:

f, g, h – хранимые структуры данных (функции),

y – ключ,

$\boldsymbol{x}$, $\boldsymbol{y}$, $\boldsymbol{z}$ – кортежи ключей,

d, e – функции и предикаты на ключах (как исходные, так и построенные из исходных с помощью выражений), функция d выдаёт *значение*, e выдаёт ключ, хранимые структуры также можно считать функциями на ключах,

K – тип ключей,

$\mathrm{S}[y{:}K]g(\boldsymbol{x},y)$ – сумма всех (ненулевых) значений $g(\boldsymbol{x},y)$ (в абелевой группе) по всем значениям ключа y из множества K,

m – гомоморфизм абелевых групп.

Предлагается использовать для определений структур следующие формулы:

$f(\boldsymbol{x}) = \mathrm{S}[y{:}K]g(\boldsymbol{x},y)$,
$f(\boldsymbol{x}) = d(\boldsymbol{x})\ g(\boldsymbol{x})$ (если $e(\boldsymbol{x})$ – логическое значение, это – условный оператор),
$f(\boldsymbol{x}) = g(\boldsymbol{y})h(\boldsymbol{z})$ (кортеж $\boldsymbol{x}$ должен быть объединением кортежей $\boldsymbol{y}$ и $\boldsymbol{z}$).
$f(\boldsymbol{x},e(\boldsymbol{x})) = g(\boldsymbol{x})$,
$f(\boldsymbol{x},h(\boldsymbol{x})) = g(\boldsymbol{x})$,
$f(\boldsymbol{x}) = m(g(\boldsymbol{x}))$.

Вычислимые структуры могут определяться рекурсивно: в итоге значение одной структуры может зависеть от её же значений, но при других ключах. Естественно, некорректное определение может привести к зацикливанию.

Заметим, что предлагаемая модель вычислений не накладывает принципиальных ограничений на алгоритмическую полноту: любое сложное циклическое вычисление может быть разбито на шаги, и номера этих шагов могут быть объявлены дополнительными ключами. Для получения окончательного результата достаточно умножить процесс на условие окончания и просуммировать его по времени.

4. Функционирование системы

Считаем, что в начальном состоянии все структуры системы находятся в нулевом состоянии (тождественные нули). В процессе работы в систему поступает сообщение об *изменении* одной из входных структур (приращение функции в одной точке). Оно приводит к пересчёту значений структур зависящих от этой входной структуры. В свою очередь, изменения, вызванные этим пересчётом, передаются дальше для пересчёта значений следующих структур и т.д.

Для реализации этого процесса в распределённой вычислительной системе можно считать, что каждая структура образует отдельный узел сети (или систему узлов с разными областями ключей). Во все узлы, от которых может зависеть данная структура, разосланы соответствующие запросы о рассылке информации о приращениях структур в этих узлах. При обработке произведений опрашиваются все связанные значения в структуре другого сомножителя. Скорость обработки запросов при этом процессе значения не имеет. Если после внесения группы изменений во входную структуру процесс стабилизируется, то он придёт к одному и тому же результату, независимо от скорости и порядка протекания частных взаимодействий. Это гарантируется тем, что в качестве значений используются элементы абелевых групп.

СПИСОК ЛИТЕРАТУРЫ

1. *Бельтюков А.П.* Машинно-независимое описание класса запросов, поддерживаемых в реальное время // Математические методы построения и анализа алгоритмов, Ленинград, 1990, с. 16-25