Использование Map/Reduce для создания замкнутых OLAP-кубов

Кудрявцев Юрий, mail@ykud.com

Московская секция ACM Sigmod,
30 октября 2008

Ключевые моменты

1. Определение OLAP
2. Теория решеток и OLAP.
   - Некоторые определения теории решеток
   - Представление OLAP-кубов решетками
3. Map Reduce для OLAP
   - Map Reduce
   - Предложенный алгоритм

OLAP (OnLine Analytical Processing) – инструмент интерактивного анализа данных пользователем. Задачи OLAP-средств:

- Предоставление больших объемов данных пользователю для анализа
- Большая скорость работы (время обработки запроса не должно превышать 3 секунд)
- Удобная модель представления данных (многомерные кубы). Интуитивно понятные интерфейсы работы (roll-up, drill-down)

История появления термина OLAP

Термин введен в статье Кодда “Providing OLAP for End-User Analysis” отосланной в IEEE Computer. В статье сформулированы 12 признаков OLAP-системы.
Последние 4 страницы статьи посвящены Essbase – проверка соответствия OLAP критериям.
Жена Кодда в это время работает в Arbor Software (разработчик Essbase). Arbor Software спонсировало написание статьи.
Журнал Computer после публикации официально изымает статью Кодда из своих архивов

FASMI критерий

Nigel Pendse – http://www.olapreport.com
FASMI

- FAST
- Analysis
- Shared
- Multidimensional
- Information

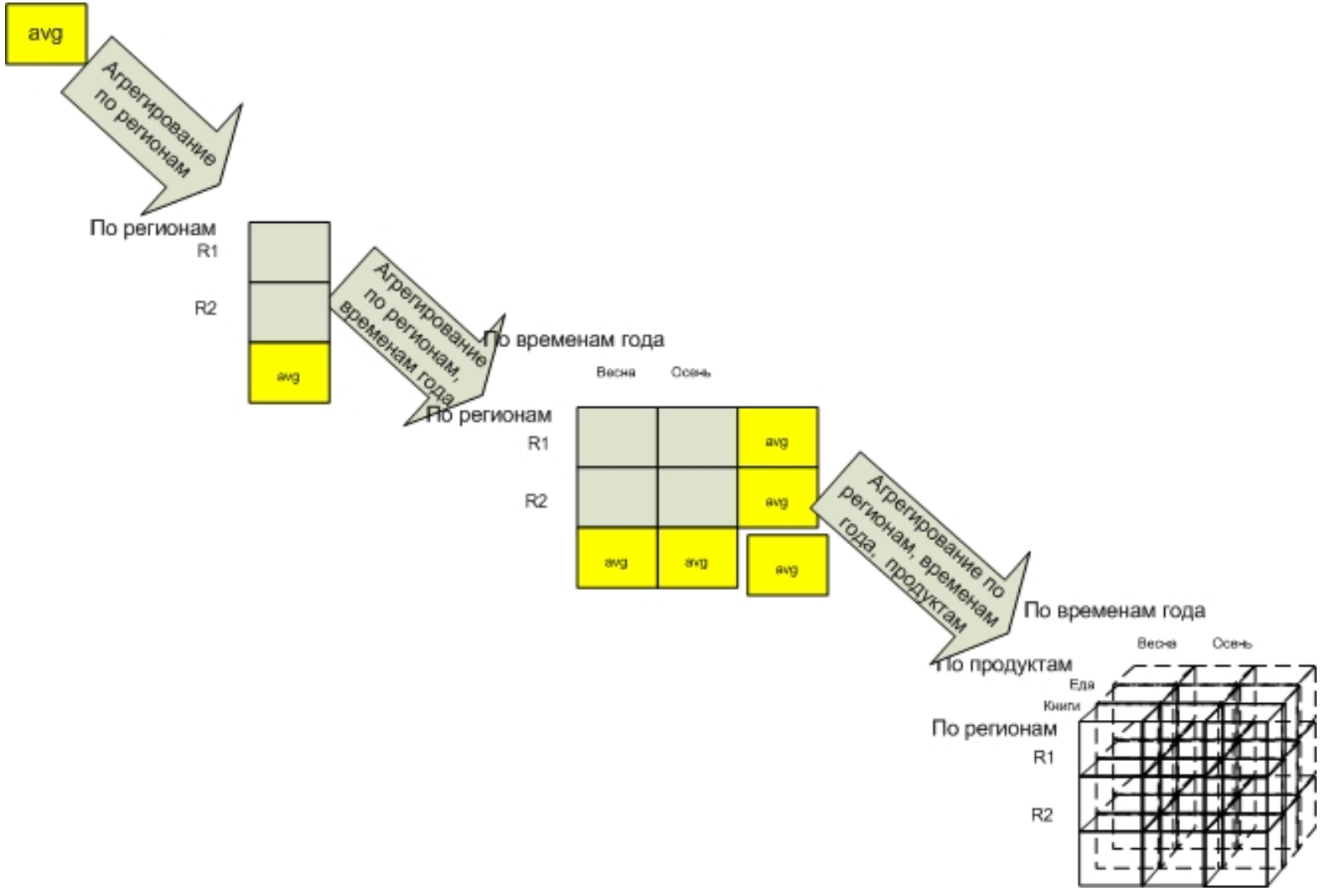
Система, удовлетворяющая вышеуказанным критериям – OLAP-система.

Основные понятия OLAP

- Измерение. В SQL обычно аргументы запроса с group by.
- Иерархия в измерении
- Мера
- Агрегирующие функции – агрегирующие функции group by запросов.
- Куб – Cube by, group by по всем комбинациям измерений.

При формировании куба возникают ячейки, которых не было в исходных данных (за счет выбора координат) – это разреженность, и масса дублирующих друг-друга агрегатов – избыточность.

Пример куба


avg
Агрегирование по регионам
По регионам
R1
R2
avg
Агрегирование по регионам, временам года
По временам года
Весна
Осень
По регионам
R1
R2
avg
avg
avg
avg
avg
Агрегирование по регионам, временам года, продуктам
По временам года
Весна
Осень
По продуктам
Еда
Книги
По регионам
R1
R2

Примеры OLAP-задач

Простые задачи:
Нарастающий итог продаж (за квартал, running period).
Window by запросы в SQL.
Кросс-таблица – pivot операторы в SQL 2005, Oracle 11g.

```
P:asc exec distinct p from t
exec P!(p!v) P by k:k from t
```

Сложные задачи Break-back, back-solving (обратное вычисление уравнений в многомерном пространстве)
Имитационное, сценарное моделирование

Как хранить OLAP кубы

Форматы хранения:

- ROLAP – все данные куба (фактическая таблица + агрегаты) хранятся в реляционной таблице
- MOLAP – и фактические данные, и агрегаты хранятся в многомерной бд
- HOLAP – фактические данные хранятся в реляционной таблице, агрегаты – в виде многомерных структур (многомерной БД)

Проблемы ROLAP

Проблемы:

- Хранение агрегатов (материализация) или вычисление на лету. Задача выбора оптимального набора материализованных представлений NP-полна.
- Моделирование измерений и вычислений
- Схемы хранения «снежинка» и «звезда» (Кимбалл и Инмон)

Расширения языка SQL:

- Grouping Set
- Rollup
- Cube
- Window By
- Oracle model by

Реляционные СУБД все еще плохо подходят для аналитических задач

Простой вопрос: посчитать в SQL тренд?

В Excel это просто $= Trend(X, Y)$

В Cognos $@Trend$

trend в Oracle Express.

И просто в Oracle

```
SELECT REG,X,
SUM(
CASE
WHEN ((SELECT REGR_SXX(Y,X) FROM data_t t1 WHERE T.REG
    = t1.REG) = 0) THEN (Y)
ELSE X*(SELECT REGR_SXY(Y,X)/REGR_SXX(Y,X) FROM data_t
    t1 WHERE T.REG = t1.REG)+(SELECT REGR_AVGY(Y,X) -
    REGR_SXY(Y,X)/REGR_SXX(Y,X)*REGR_AVGX(Y,X) FROM
    data_t t2 WHERE T.REG=t2.REG )
END
)
FROM data_t T
GROUP BY reg,X
```

Мы хотим хранить кубы в специализированной схеме, позволяющей оптимально по скорости отвечать на запросы – это MOLAP хранение.
Как выбрать подобную схему?

Частично-упорядоченным множеством называется множество, на котором определено бинарное отношение $\leq$, удовлетворяющее для всех x, y, z следующим правилам:

1. $x \leq x$ (рефлексивность)
2. если $x \leq y$ и $y \leq x$,то $x = y$ (антисимметричность)
3. если $x \leq y$ и $y \leq z$, то $x \leq z$ (транизитивность)

Пусть $H \subseteq P$ и $a \in P$. Тогда a называется верхней границей подмножества H, если $h \leq a$ для всех $h \in H$. Верхняя граница a подмножества H называется его верхней гранью или супремумом, если $a \leq b$ для любой верхней границы b подмножества H. Будем писать $a = supH$ или $a = \vee H$.
Понятие нижней границы и нижней грани, или инфинума, вводятся аналогично; инфинум обозначается $inf H$ или $\wedge H$.
Чум $(A, \leq)$ называется решеткой, если $\forall x, y \in L$ существуют $sup(x, y)$ и $inf(x, y)$.
чум $(A, \leq)$ есть решетка, если sup H и inf H существуют для любого непустого конечного подмножества H множества A.

Определения. Пересечение и объединение

Мы будем использовать обозначения

$$a \wedge b = inf\{a, b\}$$

и

$$a \vee b = sup\{a, b\}$$

и называть $\wedge$ – пересечением, а $\vee$ – объединением. В решетках они называются бинарными операциями, которые являются отображениями $A^2 \longrightarrow A$.
Операции $\wedge$ и $\vee$ обладают следующими свойствами.

1. Идемпотентность $a \wedge a = a$, $a \vee a = a$.
2. Коммутативность $a \wedge b = b \wedge a$ и $a \vee b = b \vee a$
3. Ассоциативность $(a \wedge b) \wedge c = a \wedge (b \wedge c)$ и $(a \vee b) \vee c = a \vee (b \vee c)$

Легко показать справедливость тождества поглощения:
$a \wedge (a \vee b) = a, a \vee (a \wedge b) = a$

Для решетки $(A, \geq)$ оператор $C : A \longrightarrow A$ называется оператором замыкания если:

- $x \leq C(x), x \in A$ (экстенсивность)
- $x \leq y \Rightarrow C(x) \leq C(y), x, y \in A$ (монотонность)
- $C(C(x)) = C(x), x \in A$ (идемпотентность)

Пусть r отношение базы данных над схемой R. Атрибуты R разделяются на 2 группы:

1. D - множество измерений.$(d_1, d_2, \ldots, d_{|D|}), d_i \in D[i]$ – точка в многомерном пространстве.
2. M - множество мер. Для каждой $(d_1, d_2, \ldots, d_{|D|})$ - точки, существует $(m_1, m_2, \ldots, m_{|M|})$ - множество значений мер в этой точке.

Space(r) $= \{\times_{A \in D}(r[A] \cup ALL) \cup \{0, 0, \ldots, 0\}\}$, где $\times$ - декартово произведение, $\{0, 0, \ldots, 0\}$ - нулевой кортеж.
$\forall s \in Space(r), s$ – многомерный кортеж.
Отношение порядка $\geq_g$ $u, v \in Space(r)$

$$u \geq_g v = \begin{cases} \forall A \in D, v[A] \subseteq u[A] \\ \text{в противном случае } v = \{0, 0, \ldots, 0\} \end{cases}$$

Если $u, v \in Space(r), u \geq_g v$ тогда u обобщает v в $Space(r)$.

Операторы, создающие новые кортежи: сумма $(+)$ и произведение $(\bullet)$.
Сумма двух кортежей – наименьший кортеж, обобщающий оба операнда.
$u, v \in Space(r)$

$$t = u + v \Leftrightarrow \forall A \in D, t[A] = \begin{cases} u[A], \text{if } u[A] = v[A] \\ ALL \text{ , в противном случае} \end{cases}$$

Произведение двух кортежей – наибольший кортеж, уточняющий оба операнда.
Пусть $\forall A \in D,\ z[A] = u[A] \wedge v[A]$. Тогда $u, v \in Space(r)$

$$t = u \bullet v \Leftrightarrow \begin{cases} t = z \text{ если } \neg\exists A \in D | z[A] = \{0\} \\ \{0, 0, \ldots, 0\} \text{ , в противном случае} \end{cases}$$

Пусть r – отношение базы данных над $D \cup M$. Чум $CL(r) = \{Space(r), \geq_g\}$ – решетка, в которой пересечение и объединение вводятся следующим образом:

$$\forall T \subseteq CL(r),\ \wedge T = +_{t \subseteq T} t$$

$$\forall T \subseteq CL(r),\ \vee T = \bullet_{t \subseteq T} t$$

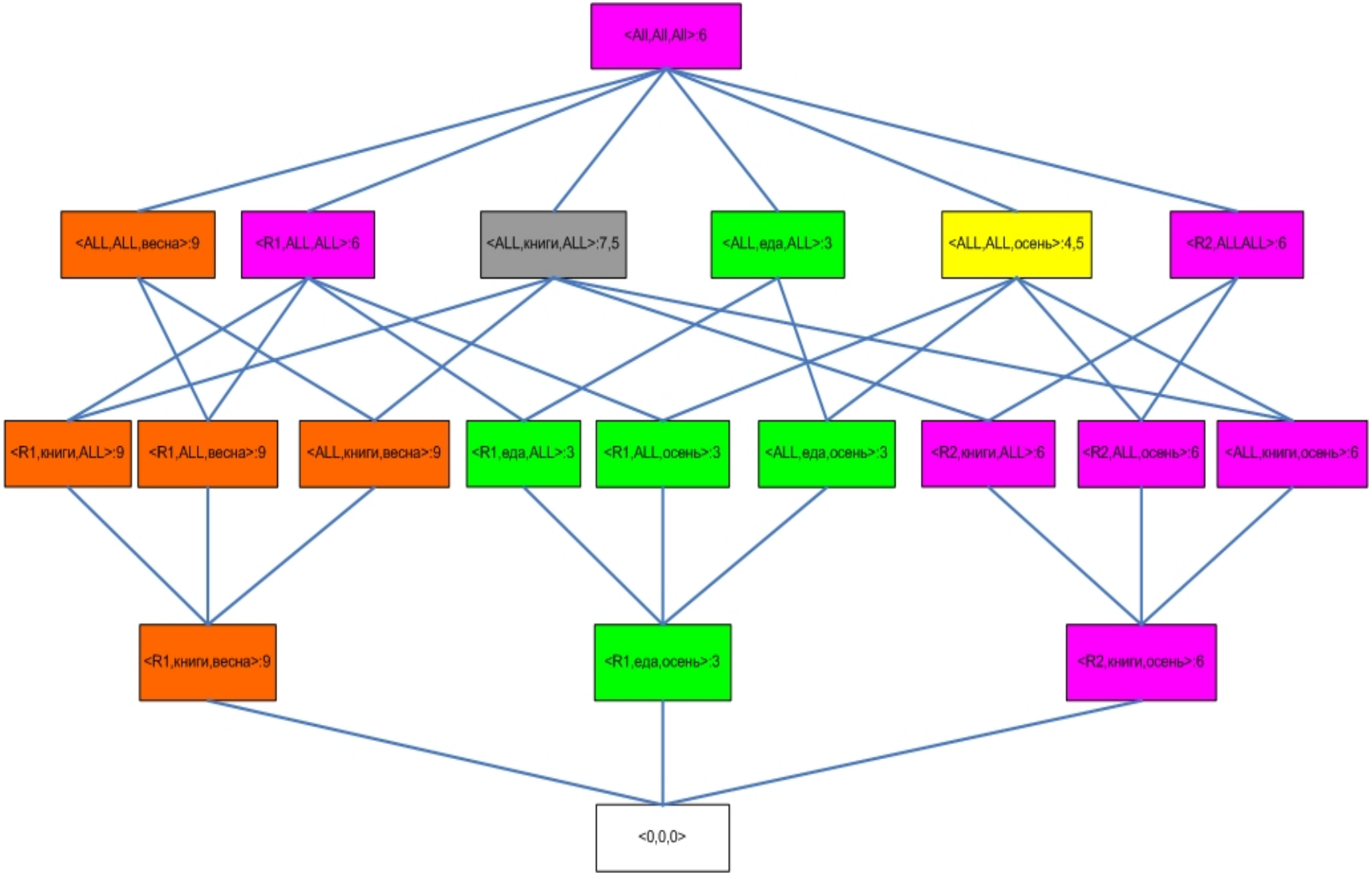
$\{CL(r), \geq_g, \wedge, \vee\}$ – решетка куба.

Мы хотим каким-то образом сократить объем данных, которые требуется хранить.
Допустим, мы сгруппируем ячейки куба по значению меры в ячейке.

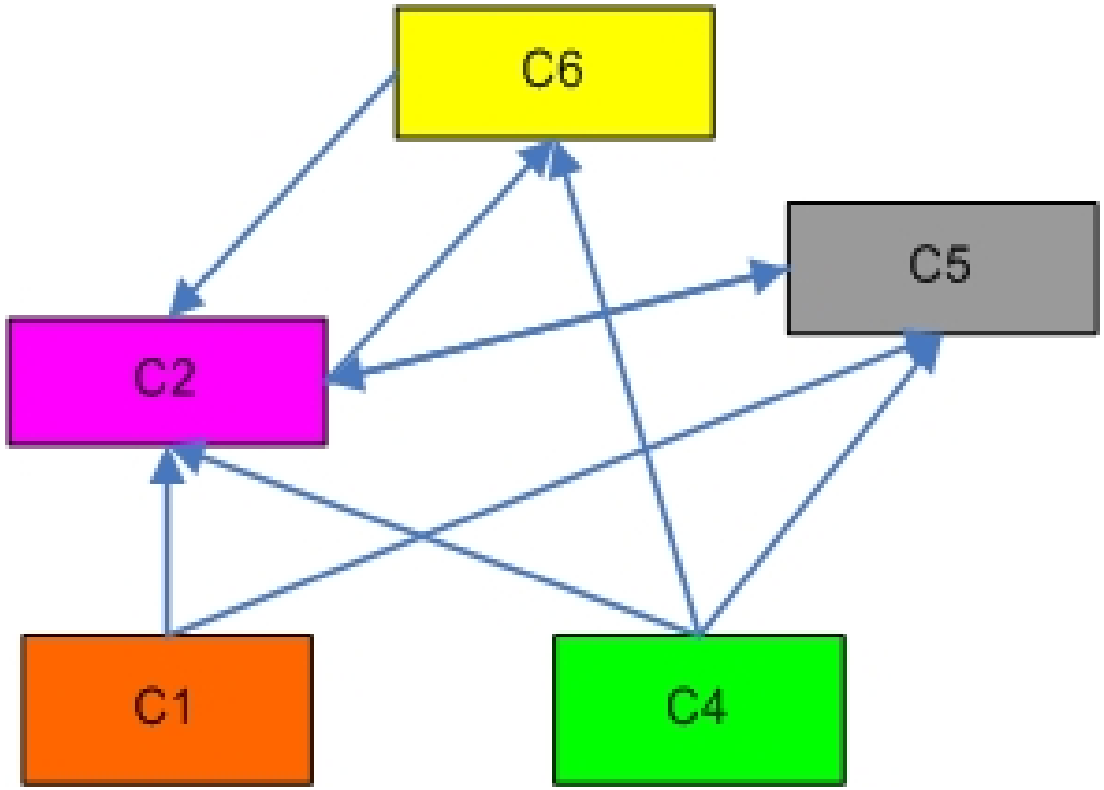
Пример

Регион	Продукт	Время года	Продажи
R_1	книги	весна	9
R_1	книги	осень	3
R_2	еда	осень	6

Группировка по значению меры в ячейке



Замкнутые классы по значению меры в ячейке



Не сохраняются отношения специализации и агрегирования – подобное представление нельзя использовать.

Отношение покрытия для решетки куба

Кортеж $c \in CL(r)$ покрывает базовый (фактический) кортеж $t \in r$, если $c \geq_g t$.

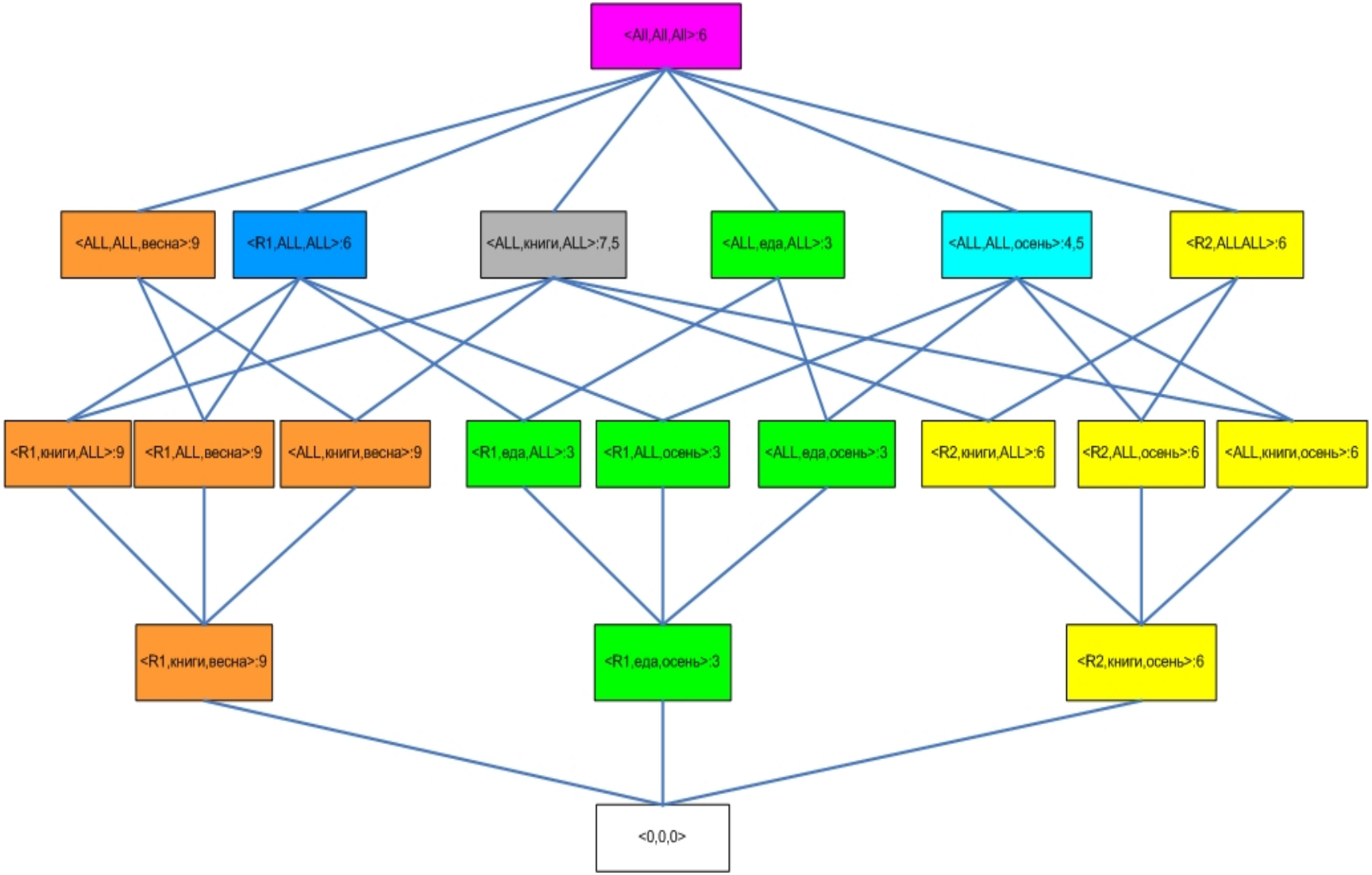
Отношение эквивалентности по покрытию

Определим отношение эквивалентности $\equiv_{cov}$ как транзитивное и рефлексивное замыкание R, где:

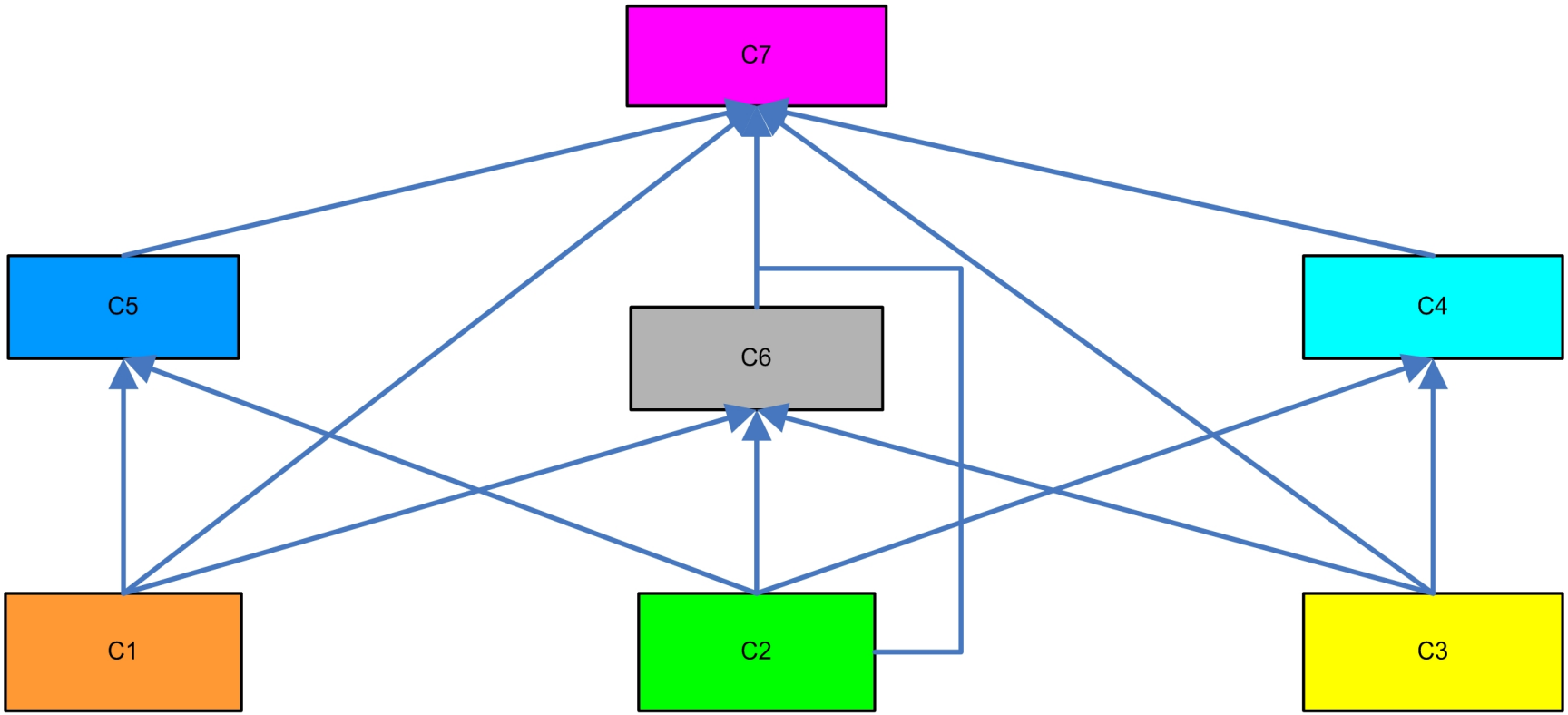
$R: \quad a, b \in CL(r)$

$$R(a,b) \Longleftrightarrow \exists T \in CL(r), \forall t \in T \begin{cases} t \in r \\ a \geq_g t \text{ и } b \geq_g t \\ \neg \exists t' \notin T : t' \in r \\ a \geq_g t' \text{ или } b \geq_g t' \end{cases}$$

Группировка по покрытию

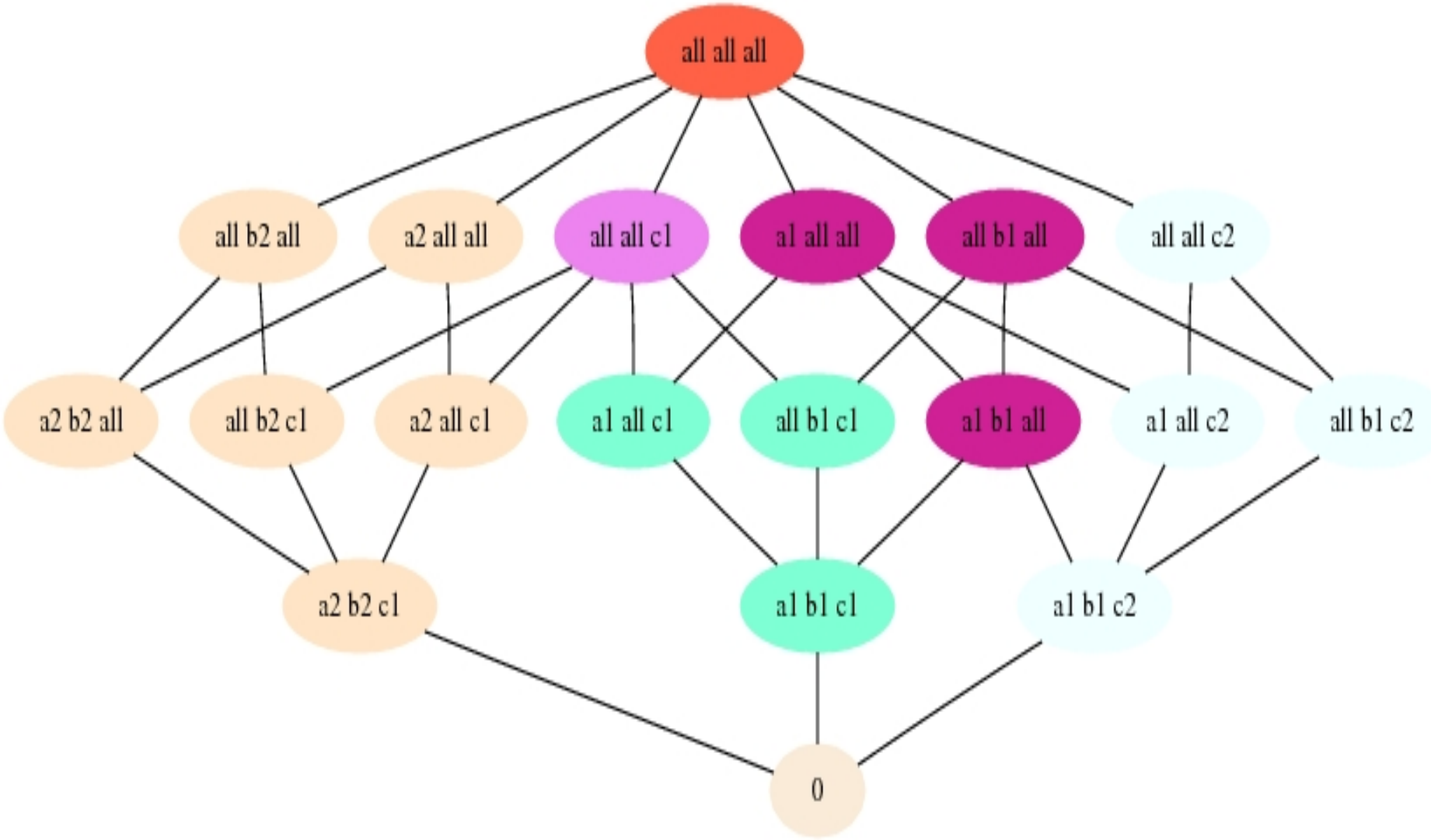


Замкнутые классы по покрытию



Сохраняются отношения специализации и агрегирования.

Средство рисования замкнутых решеток кубов

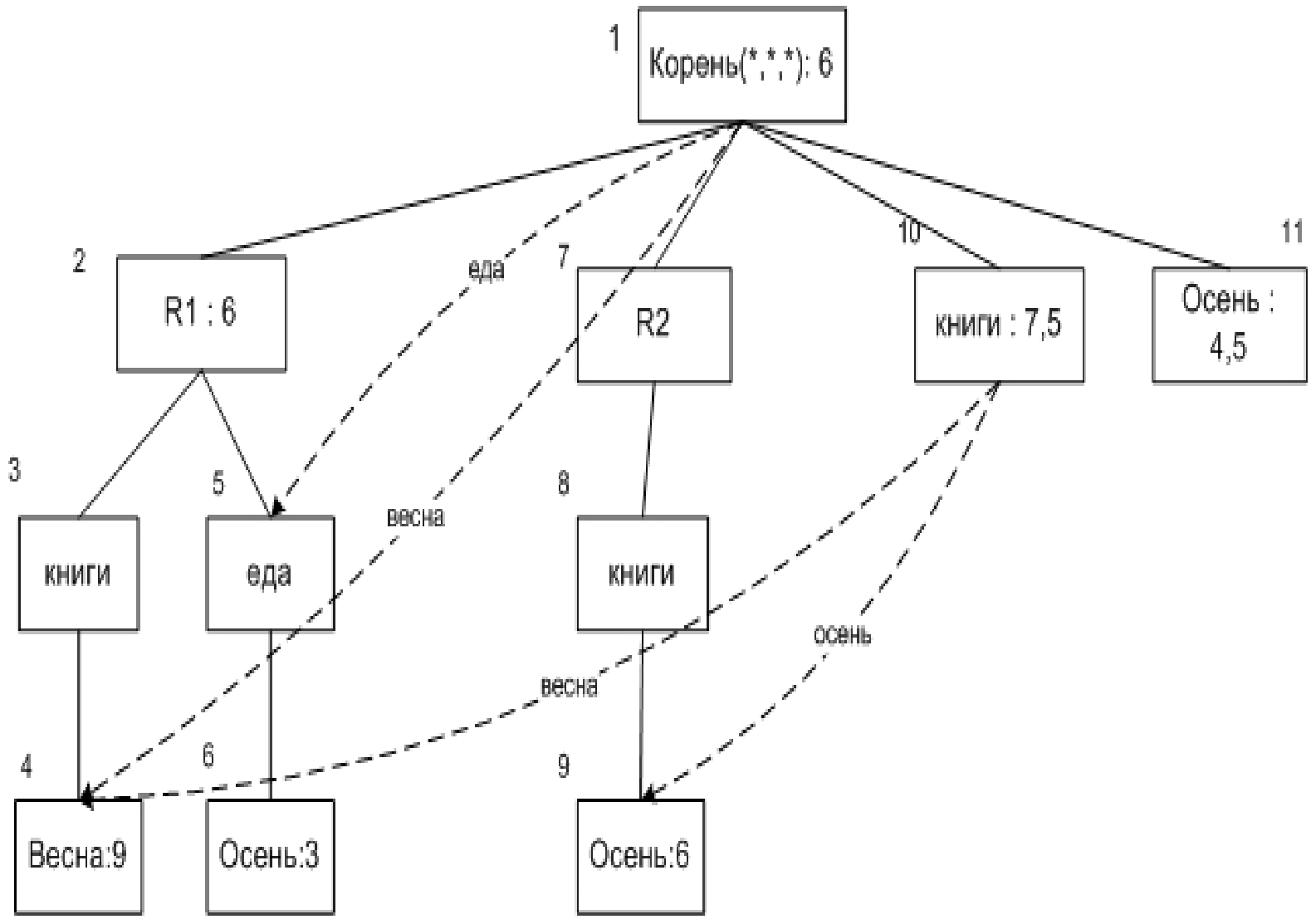


http://ykud.ru/draw_closed_cube

Оптимальность разбиения куба на классы эквивалентности по покрытию

Минимальным (с точки зрения количества классов эквивалентности) представлением куба является замкнутая решетка куба, но она не сохраняет roll-up/drill-down отношения.
Можно показать, что замкнутая решетка куба и решетка куба по введенному отношению покрытия – эквивалентны (см статью "Математическая модель OLAP-кубов", список литературы)

Схема хранения куба



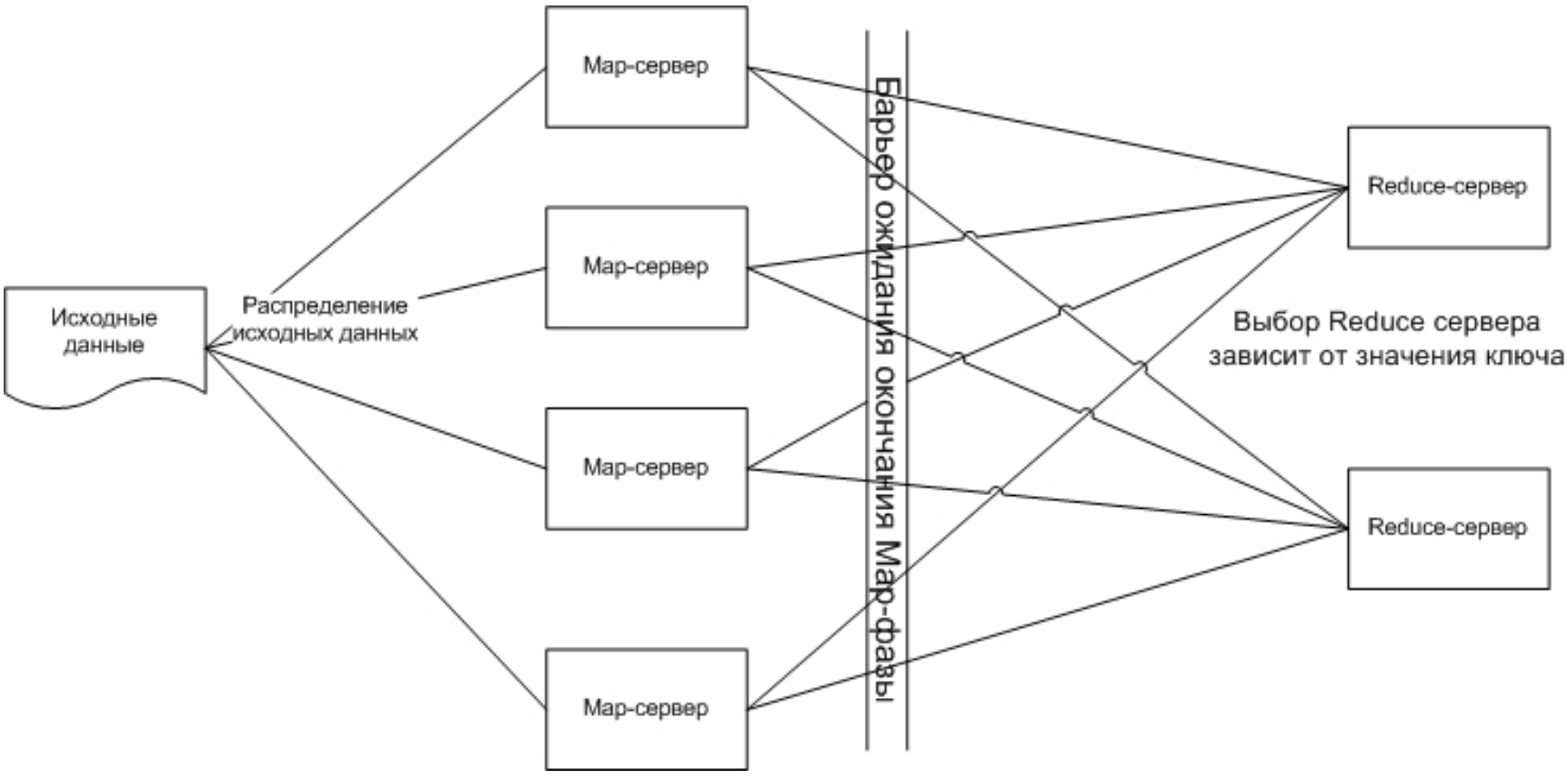
См описание MOLAP алгоритмов (список литературы)

Что такое Map Reduce

Map/Reduce – парадигма параллельных вычислений, предложенная Google. Алгоритм в такой платформе состоит из 2х фаз – map и reduce (эквивалентно map и fold в функциональных ЯП)

- Map-процессы запускаются над подмножествами исходных данных и выполняются абсолютно независимо друг от друга.
- Reduce-процессы обрабатывают результаты map-фазы, разбивая их по значениям ключей на непересекающиеся блоки, что также позволяет выполнять их независимо.
- Таким образом, каждая из фаз может обрабатываться на любом количестве серверов параллельно.

Схема Map Reduce



Пример Map-Reduce, Алгоритм подсчета слов в тексте

map – разбивает блок строк на слова, для каждого слова генерирует пару (слово, 1)
reduce – для каждого слова, складывает 1

Ключевые понятия map-reduce

- Максимально локальная обработка данных – сетевая передача самый медленный этап
- Map-фаза не должна увеличивать объем данных
- Combiner функции – локально агрегируют данные, чтобы уменьшить объем передаваемых map-результатов. Для примера с подсчетом слов – такая же как и reduce
- Distributed Cache – буфер, доступный всем map-задачам чтобы позволить сокращать объем результатов
- Возможно запускать собственные внешние функции на других ЯП (streaming)

Простой вариант map-reduce алгоритма расчета замыкнутых ячеек

- map – для каждой фактической ячейки порождаем все агргегаты N-го уровня с 1 для подсчета support. Для (R1, еда, весна), 1го уровня получаем ((*, еда, весна),1), ((R1, *, весна),1), ((R1, еда, *),1)
- reduce – считаем support (количество покрываемых ячеек) для каждого агрегата и агрегаты с support = N являются замкнутыми ячейками.
- Считаем map - reduce для агрегатов максимального уровня. Те, у которых support = 1 – замкнутые классы.
- Аналогично для максимального -1.

Почему такого подхода недостаточно

В map-фазе порождается экспоненциально растущий объем данных – скорость работы алгоритма очень низкая.
Можно генерировать множество локальных замкнутых кубов и использовать map/reduce для агрегирования результатов запросов. Вся обработка данных происходит на локальном сервере, по сети передаются лишь результаты запросов.
При этом локальные кубы можно так же строить с помощью map/reduce – используем многопроцессорные/многоядерные сервера.

Техническая платформа

- Многосерверный map/reduce – Apache Hadoop
  - Самая популярная открытая map reduce платформа
  - Используется в кластерах Yahoo, самый большой кластер 20 тысяч процессоров
  - Готовые образа для Amazon EC2 – cloud computing платформы аренды серверов
- Многопроцессорный map/reduce – Phoenix, разработка Stanford University.
  - Оптимизирована для работы с данными в ОП
  - Тесты показывают, что она не сильно уступает параллельному программированию на pthreads
  - Distributed Cache – просто область оперативной памяти

Схема создания кубов

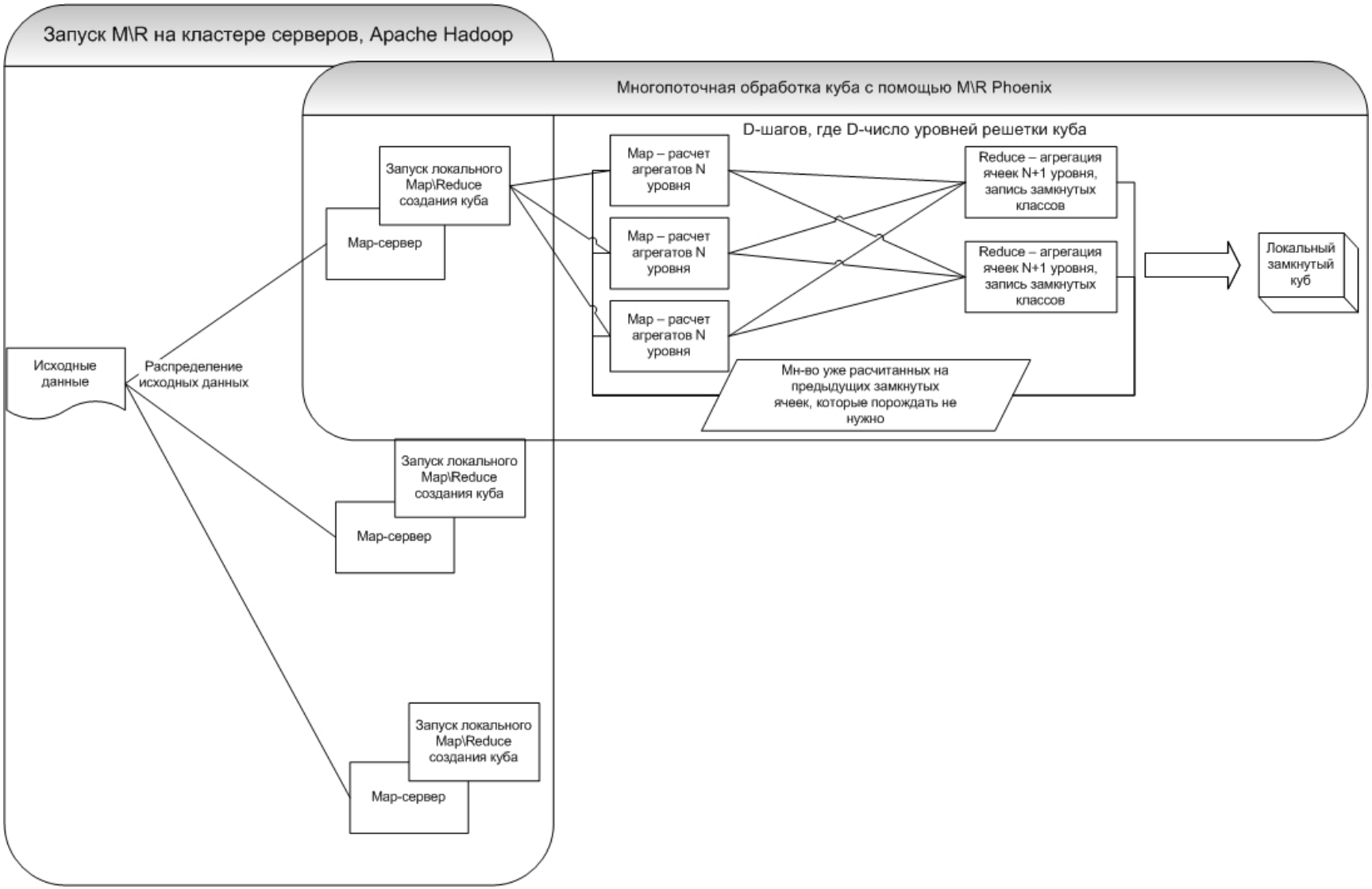
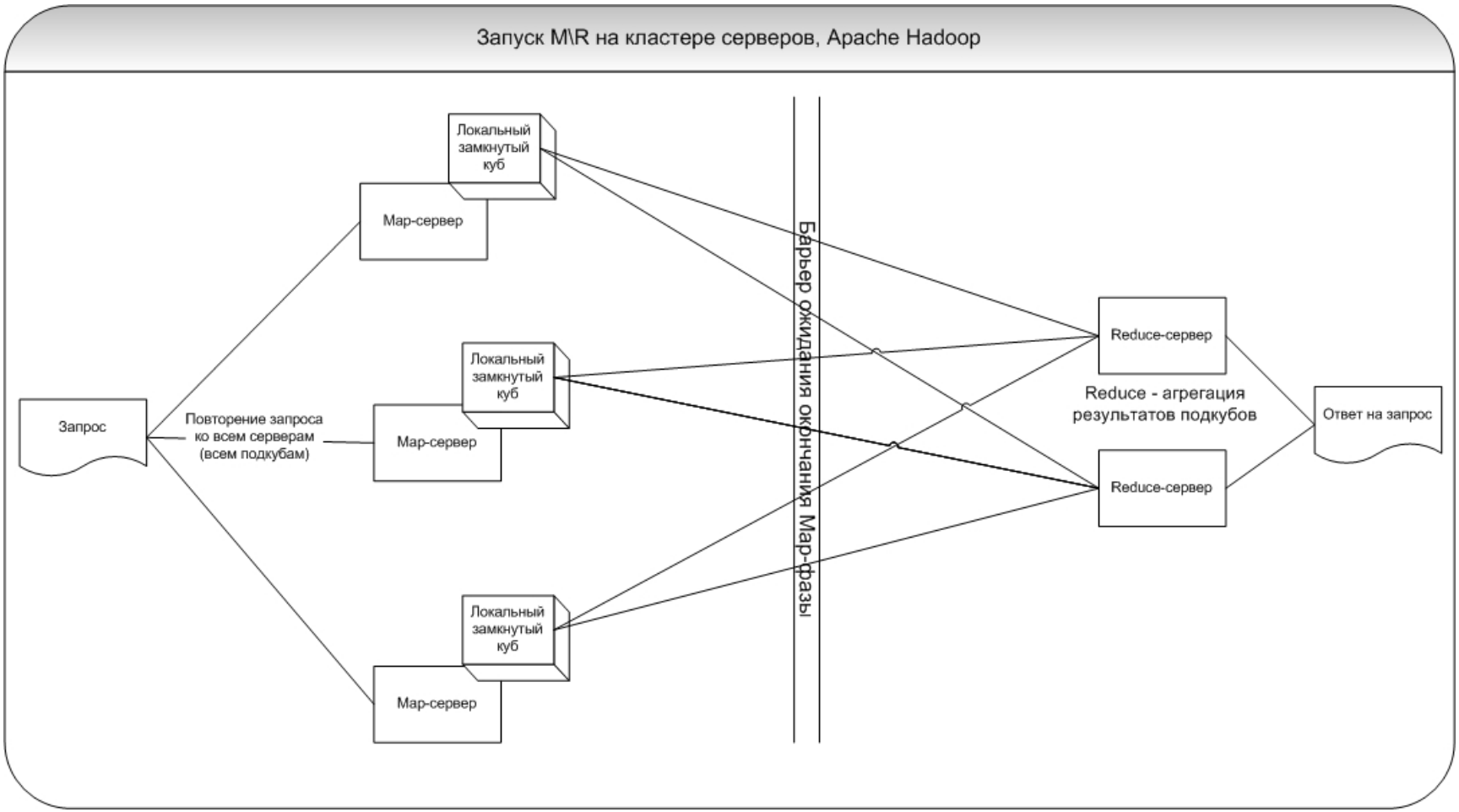


Схема ответов на запросы



Алгоритм расчета замыканий. Ключевые идеи

- Чередуем шаги сверху и снизу (генерируем ячейки максимального и минимального уровня), стараясь максимально сузить пространство потенциальных ячеек.
- Используем Distributed Cache для хранения промежуточных результатов – в результате работы в нем оказываются все замкнутые классы

Listing 1: Close cube computation

```
1 compute_closed_cube(data d,distributed cache dc, number
      of dimensions D):
2 for i in [1..D/2]:
3 { if map(d, D–i, dc) <> 0 {reduce(d, D–i, dc)}
4 else cube is computed
5 if map(d, i, dc) <> 0 {reduce(d, i, dc)}
6 else cube is computed}
7 Замкнутые классы трансформируются в QC–Tree для
      обработки запросов.
```

Если на map шаге не сгенерировано данных – все возможные замыкания уже построены и куб записан в Distributed Cache.

Listing 2: Map function

```
1 map(data d, level L, distributed cache dc):
2 for each cell c in d
3 {
4 c_aggr = generate_aggregate_cells(c, L)
5 for each aggr_cell in c_aggr:
6 {if for each ubound in dc[classes_upper_bounds]
7 (aggrc * ubound == {0,0,...,0}) then output (aggrc,1)}
8 }
```

Шаги 6-7 определяют входит ли ячейка в уже расчитанный замкнутый класс и позволяют ее в таком случае не порождать.

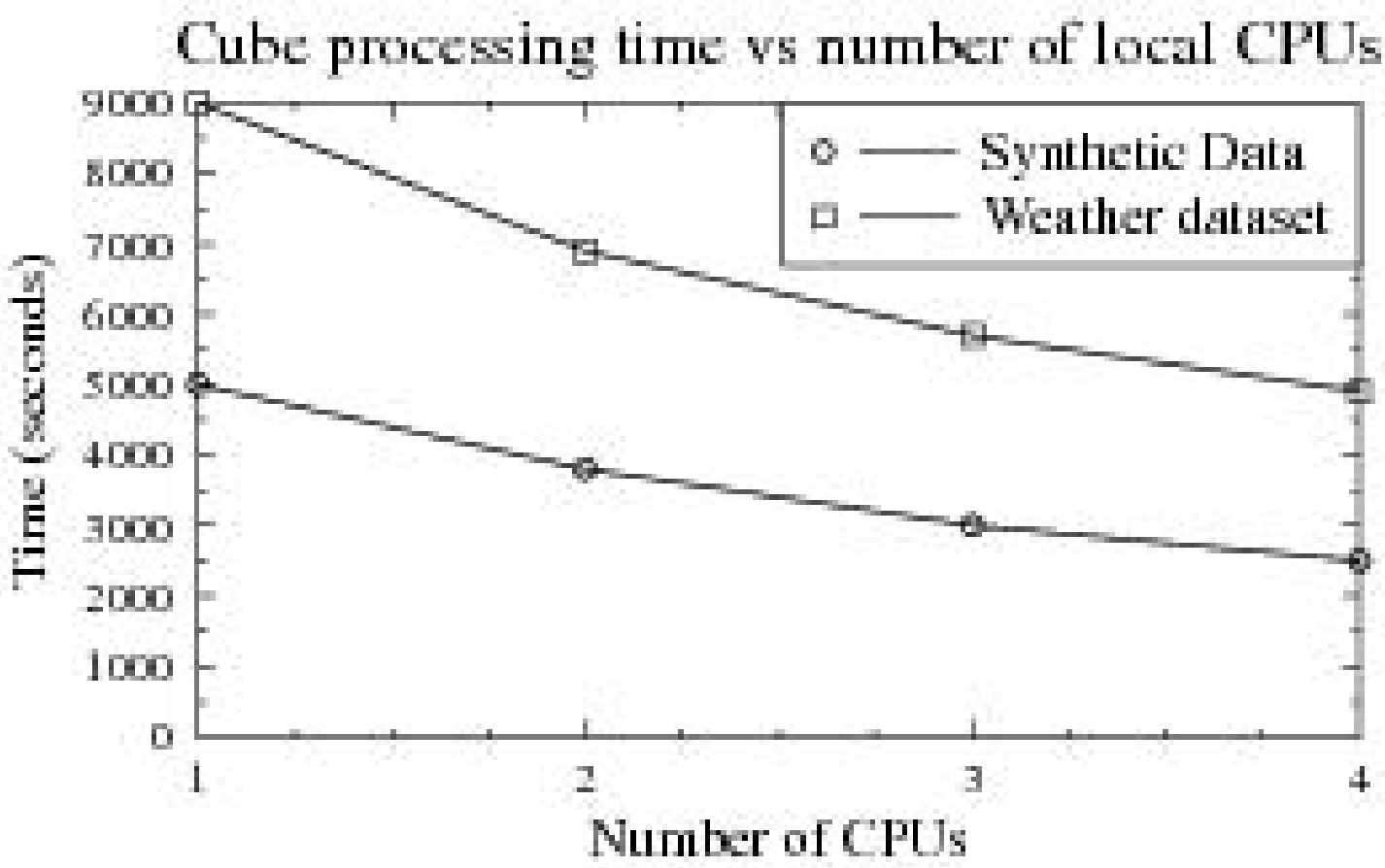
Алгоритм расчета замыканий. Reduce функция

Listing 3: Reduce function

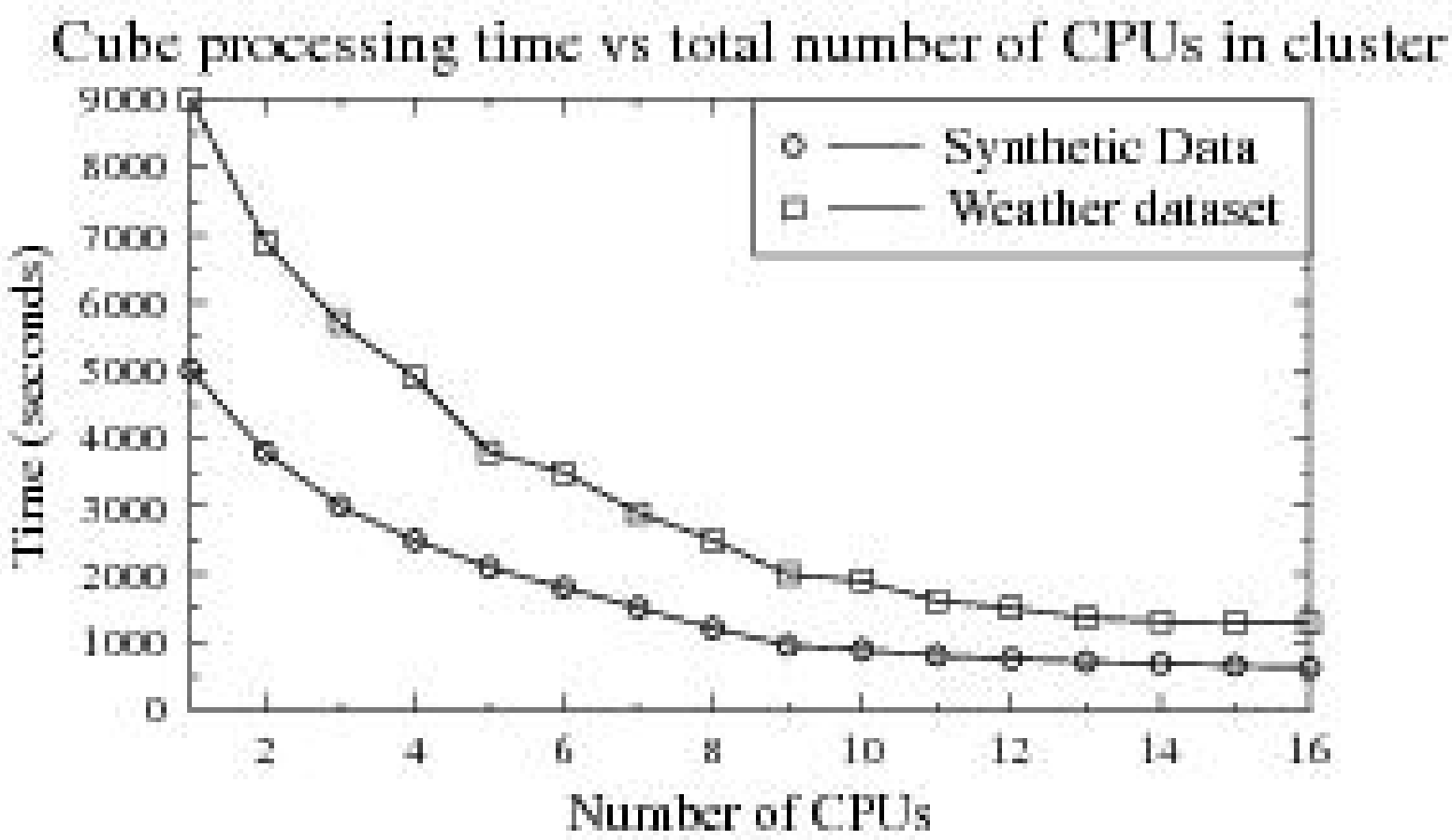
```
1  reduce(map_output, level L, distributed cache dc):
2  for each (cell,support) in map_output
3  {resulting_support = sum of map emited supports
4  if (resulting_support = (D - L + 1))
5  {if exists lbound,lbound_support in dc[closed_classes_lowerbounds]
6  (cell >= lbound) and (lbound_support = resulting_support)
7  {dc[closed_classes] = (cell,ubound)}
8  else
9  {dc[closed_classes_upperbounds]+= (cell,resulting_support)}
10 }
11 else if resulting_support > 1
12 {if exists ubound,ubound_support in dc[closed_classes_upperbounds]
13 (ubound >= cell) and (ubound_support = resulting_support)
14 {dc[closed_classes] = (cell,ubound)}
15 else
16 {dc[closed_classes_lowerbounds]+= (cell,resulting_support)}}
17 }
```

Во время вычисления ячеек мы записываем нижние и верхние грани, т.о. определяя замкнутые классы на идя сверху вниз (шаги 5-9) или снизу вверх (12-14)

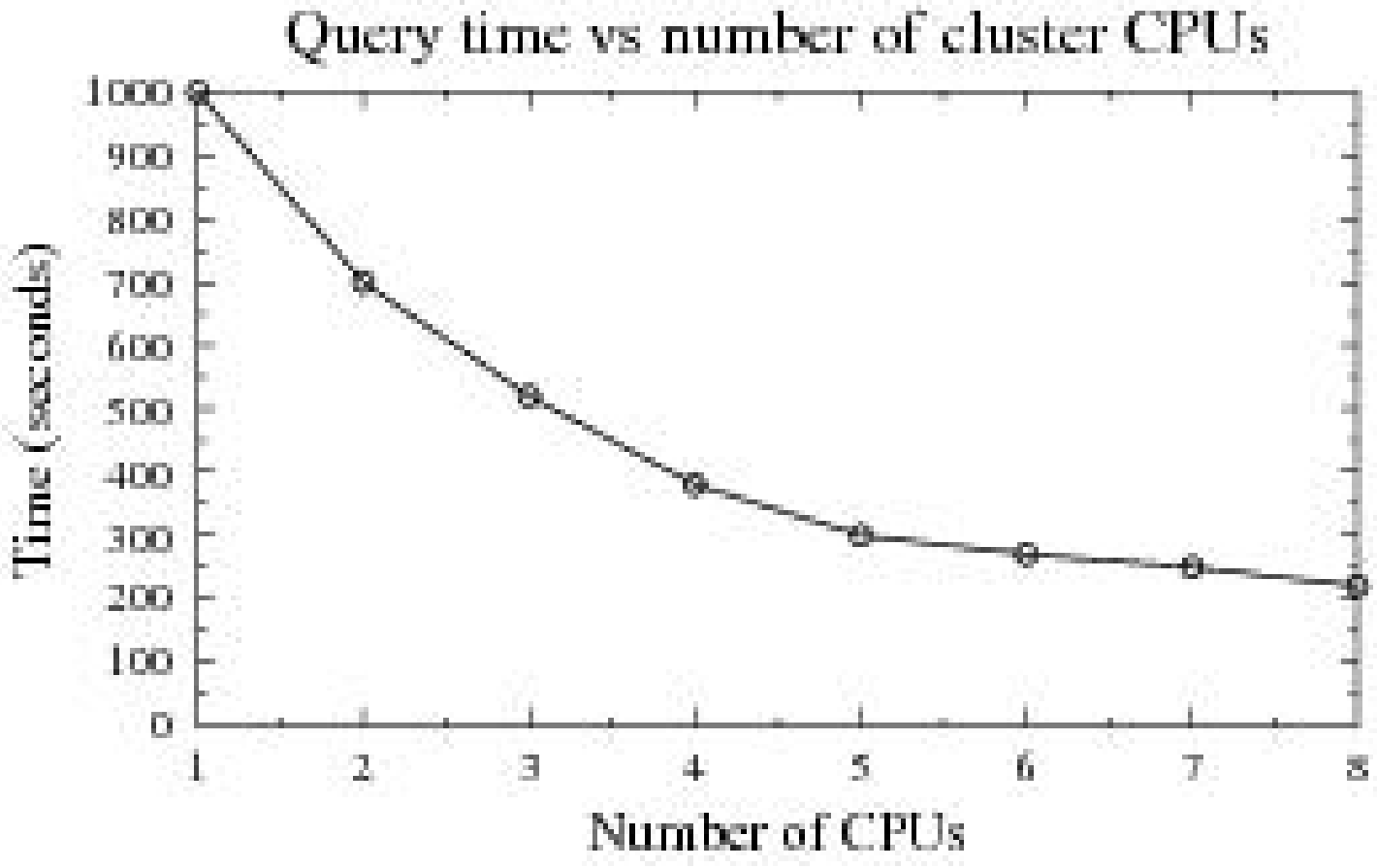
Результаты экспериментов. Ускорение на многопроцессорном сервере



Результаты экспериментов. Ускорение на кластере



Результаты экспериментов. Ответы на запросы



Преимущества подобного алгоритма

- Апробация map/reduce для OLAP
- Масштабирование вычислительно сложной задачм рассчета замыканий
- Эффективное использование оборудования – сервера подключаются по мере необходимости. Машино-час стоит менее 10 центоа

Недостатки подобного алгоритма

- Только дистрибутивные функции
- Относительно медленный отклик на запросы
- Сложная технологическая система

Направления дальнейших исследований

- Распределение данных для локальных кубов – определяет сложность ответов на запросы.
- Многоуровневые иерархии
- Частичная поддержка XML/A
- Стабильность работы)

Список литературы I


Codd E.F.
Providing OLAP for end-user analysys: An IT mandate.
IEEE Computer,1993


Thomsen Eric
OLAP Solutions: Building Multidimensional Information Systems. Second Edition.
Wiley, 2002


Rafanelli M.
Multidimensional Databases — Problems and Solutions.
Idea Grouping Publ., Hershey, London, Melbourne, Singapore, Beijing, 2003


Celko Joe.
Analytics and OLAP in SQL.
Morgan Kaufmann, 2006


Кудрявцев Ю.
Обзор алгоритмов MOLAP.
http://citforum.ru/consulting/BI/molap_overview/


Кузнецов С.Д., Кудрявцев Ю.
Математическая модель OLAP-кубов
http://ykud.ru/articles/olap_lattices/index.html

Спасибо!