Максимальное распараллеливание алгоритмов на основе концепции Q-детерминанта

Валентина Николаевна Алеева

Южно-Уральский государственный университет (НИУ)

Новосибирск, 2015

ВВЕДЕНИЕ

- В докладе рассматривается подход к распараллеливанию алгоритмов, основанный на концепции Q-детерминанта.
- Концепция Q-детерминанта позволяет исследовать ресурс распараллеливания алгоритма: получить все его реализации, в том числе максимально быструю, провести анализ параллельной сложности.
- Концепция Q-детерминанта ориентирована на модель вычислительной системы (ВС) с потенциально неограниченными ресурсами, однако полученные с ее помощью результаты могут являться основой для эффективного выполнения алгоритмов на реальных параллельных ВС.

ВВЕДЕНИЕ

- Концепция Q-детерминанта была разработана в 1985 году.

  Ее появление вызвано тем, что в то время появились работы по распараллеливанию конкретных численных алгоритмов, при этом универсального подхода к распараллеливанию алгоритмов не существовало.

  Предложенная концепция решила эту задачу.

- Результаты, полученные на начальном этапе исследований, не были опубликованы в издании, доступном широкому кругу специалистов.

- Параллельные ВС того времени не имели достаточного количества вычислительных узлов, чтобы можно было выполнять на них максимально быстрые реализации алгоритмов, поэтому применить полученные результаты на практике было невозможно.

- В последние годы исследования на основе концепции Q-детерминанта были продолжены. В докладе о них будет сказано.

Представление алгоритма в форме Q-детерминанта

Базовым понятием концепции Q-детерминанта является представление алгоритма в форме Q-детерминанта.
Пусть $\mathcal{A}$ — некоторый алгоритм для решения алгоритмической проблемы

$$\bar{y} = F(N, B),$$

где N — множество параметров размерности проблемы,
N удовлетворяет условиям: либо $N = \varnothing$, либо $N = \{n_1, \ldots, n_k\}$, где $k \geqslant 1$, а n_i $(1 \leqslant i \leqslant k)$ принимает любые целые положительные значения;
B — множество входных данных (счетное множество переменных любого типа);
$\bar{y} = (y_1, \ldots, y_m)$ — множество выходных данных, $y_i \notin B$ $(i = 1, \ldots, m)$,
m является либо вычислимой функцией параметров N при $N \neq \varnothing$, либо константой.
Если $N = \{n_1, \ldots, n_k\}$, то через $\bar{N}$ обозначим вектор $(\bar{n}_1, \ldots, \bar{n}_k)$, где $\bar{n}_i$ — некоторое заданное значение параметра n_i $(1 \leqslant i \leqslant k)$.
Множество всевозможных векторов $\bar{N}$ обозначим $\{\bar{N}\}$.

Представление алгоритма в форме Q-детерминанта

- Пусть Q — множество операций, используемых алгоритмом $\mathcal{A}$.
- Для операций из множества Q заданы приоритеты выполнения.
- Будем допускать, что *все* операции из Q *нульместны (константы), одноместны, или двуместны.*
- Примером множества Q является множество арифметических, логических операций и операций сравнения.
- Над множествами B и Q можно строить *выражения*, которые имеют *уровни вложенности*.
- Будем обозначать уровень вложенности выражения w через T^w.
- Более точно, под *выражениями* будем понимать *термы сигнатуры* Q в стандартном смысле математической логики.

Представление алгоритма в форме Q-детерминанта

Уровень вложенности выражения определяется индуктивно:

1) Константы и элементы B имеют *нулевой уровень вложенности*.
2) Круглые скобки используются для задания порядка выполнения операций. Если w — выражение, то $T^w = T^{(w)}$.
3) Если w — выражение, $T^w = i - 1$ $(i \geqslant 1)$ и $f \in Q$ — одноместная операция, то $T^{f(w)} = i$, а w называется *подвыражением* $(i-1)$*-го уровня вложенности выражения* $f(w)$.

   В этом случае $f \in Q$ называется *операцией* i*-го уровня вложенности*.
4) Если w и v — выражения, $T^w = i$, $T^v = j$ и $g \in Q$ — двухместная операция, то $T^{g(w,v)} = k$, где $k = \max(i, j) + 1$.

   Выражения w и v называются *подвыражениями* $g(w, u)$ соответственно i и j уровней вложенности, а операция g называется *операцией* k*-го уровня вложенности*.

Представление алгоритма в форме Q-детерминанта

- Выражение, образованное из n выражений путем применения $n-1$ раз одной из ассоциативных операций множества Q, будем называть *цепочкой* выражений длины n.
- При определении уровня вложенности выражения и его подвыражений порядок выполнения операций цепочки задается по схеме сдваивания.
- Если задана интерпретация всех переменных, входящих в выражение, то будем говорить, что задана *интерпретация выражения*.
- Если задана интерпретация выражения, то его можно вычислить.

Представление алгоритма в форме Q-детерминанта

Введем понятие Q-терма.

1) Если $N = \varnothing$, то любое выражение w над множествами B и Q будем называть *безусловным Q-термом*.
   Уровнем вложенности безусловного Q-терма w будем называть уровень вложенности выражения w.
   Пусть V — множество всех выражений над B и Q.
   Если $N \neq \varnothing$, то любое однозначное отображение $w : \{\bar{N}\} \to V$ будем также называть *безусловным Q-термом*.
   Уровнем вложенности безусловного Q-терма w будем называть функцию $T^w : \bar{N} \to T^{w(\bar{N})}$, где $T^{w(\bar{N})}$ — уровень вложенности выражения $w(\bar{N})$.

Представление алгоритма в форме Q-детерминанта

2) Пусть $N = \varnothing$ и задан безусловный Q-терм w. Тогда, если выражение w над множествами B и Q при любой интерпретации B принимает значение логического типа, то безусловный Q-терм w будем называть *безусловным логическим Q-термом*.
Пусть $N \neq \varnothing$ и задан безусловный Q-терм $w : \{\bar{N}\} \to V$.
Если при любом $\bar{N} \in \{\bar{N}\}$ и любой интерпретации переменных B выражение $w(\bar{N})$ принимает значение логического типа, то w будем называть *безусловным логическим Q-термом*.
3) Пусть $u_1, \ldots, u_l$ — безусловные логические Q-термы, $w_1, \ldots, w_l$ — безусловные Q-термы.
Множество l пар (u_i, w_i), где $i = 1, \ldots, l$, будем обозначать $(\bar{u}, \bar{w}) = \{(u_i, w_i)\}_{i=1,\ldots,l}$ и называть *условным Q-термом длины l*.
4) Счетное множество пар безусловных Q-термов $(\bar{u}, \bar{w}) = \{(u_i, w_i)\}_{i=1,2,\ldots}$ будем называть *условным бесконечным Q-термом*, если $\{(u_i, w_i)\}_{i=1,\ldots,l}$ является условным Q-термом для любого $l < \infty$.
5) Если не имеет значения, является ли Q-терм безусловным, условным или условным бесконечным, то будем называть его *Q-термом*.

Представление алгоритма в форме Q-детерминанта

- *Под вычислением безусловного Q-терма w при интерпретации B* будем понимать вычисление выражения w, если $N = \varnothing$, и вычисление выражения $w(\bar{N})$ при некотором $\bar{N} \in \{\bar{N}\}$, если $N \neq \varnothing$.
- Опишем *вычисление при заданной интерпретации B условного Q-терма* $(\bar{u}, \bar{w}) = \{(u_i, w_i)\}_{i=1,\ldots,l}$.
  Если $N = \varnothing$, то следует вычислить выражения u_i, w_i $(i = 1, \ldots, l)$.
  Если существуют выражения u_{i_0}, w_{i_0} $(i_0 \leqslant l)$ такие, что u_{i_0} принимает значение `true`, а значение w_{i_0} определено, то будем считать, что $(\bar{u}, \bar{w})$ принимает значение w_{i_0}. В противном случае считаем, что значение $(\bar{u}, \bar{w})$ при данной интерпретации B не определено.
  Если $N \neq \varnothing$, то зададим $\bar{N} \in \{\bar{N}\}$.
  Получим выражения $u_i(\bar{N}), w_i(\bar{N})$ $(i = 1, \ldots, l)$ и вычислим их.
  Если существуют выражения $u_{i_0}(\bar{N}), w_{i_0}(\bar{N})$ $(i_0 \leqslant l)$ такие, что $u_{i_0}(\bar{N})$ принимает значение `true`, а значение $w_{i_0}(\bar{N})$ определено, то будем считать, что $(\bar{u}, \bar{w})$ принимает значение $w_{i_0}(\bar{N})$. В противном случае считаем, что значение $(\bar{u}, \bar{w})$ для данного $\bar{N}$ и данной интерпретации B не определено.

Представление алгоритма в форме Q-детерминанта

Опишем *вычисление при заданной интерпретации B условного бесконечного Q-терма* $(\bar{u}, \bar{w}) = \{(u_i, w_i)\}_{i=1,2,\dots}$.
Если $N = \varnothing$, то для вычисления условного бесконечного Q-терма $(\bar{u}, \bar{w}) = \{(u_i, w_i)\}_{i=1,2,\dots}$ необходимо, вычисляя выражения u_i, w_i $(i = 1, 2, \dots)$, найти u_{i_0}, w_{i_0} такие, что u_{i_0} принимает значение `true`, а значение w_{i_0} определено.
В качестве значения $(\bar{u}, \bar{w})$ нужно взять w_{i_0}.
Если установлено, что выражений u_{i_0}, w_{i_0} не существует, то значение $(\bar{u}, \bar{w})$ при данной интерпретации B не определено.
Аналогично можно определить вычисление бесконечного условного Q-терма в случае, когда $N \neq \varnothing$.

Представление алгоритма в форме Q-детерминанта

Предположим, что I_1, I_2, I_3 — подмножества множества $I = (1, \ldots, m)$, удовлетворяющие условиям:

1) $I_1 \cup I_2 \cup I_3 = I$;
2) $I_i \cap I_j = \varnothing$ $(i \neq j; i, j = 1, 2, 3)$;
3) Одно или два из множеств I_i $(i = 1, 2, 3)$ могут быть пустыми.

Рассмотрим такое множество Q-термов $\{f_i\}_{i \in I}$, что:

1) f_{i_1} $(i_1 \in I_1)$ — безусловный Q-терм, $f_{i_1} = w^{i_1}$;
2) f_{i_2} $(i_2 \in I_2)$ — условный Q-терм, $f_{i_2} = \left\{ \left(u_j^{i_2}, w_j^{i_2} \right) \right\}_{j=1,\ldots,l_{i_2}}$,

   l_{i_2} является либо вычислимой функцией параметров N при $N \neq \varnothing$, либо константой;
3) f_{i_3} $(i_3 \in I_3)$ — условный бесконечный Q-терм, $f_{i_3} = \left\{ \left(u_j^{i_3}, w_j^{i_3} \right) \right\}_{j=1,2,\ldots}$.

Предположим, что алгоритм $\mathcal{A}$ состоит в том, что для определения y_i $(i \in I)$ требуется вычислить Q-терм f_i.
Тогда множество Q-термов f_i $(i \in I)$ будем называть *Q-детерминантом алгоритма $\mathcal{A}$*, а представление алгоритма в виде $y_i = f_i$ $(i \in I)$ представлением в форме Q-детерминанта.

Максимально быстрая реализация алгоритма

- Реализацией алгоритма $\mathcal{A}$, представленного в форме Q-детерминанта $y_i = f_i$ $(i \in I)$, будем называть вычисление Q-термов f_i $(i \in I)$.
- Если реализация такова, что две или более операций выполняются одновременно, то такую реализацию будем называть параллельной.
- Если реализация алгоритма $\mathcal{A}$ производится с помощью ВС, то будем говорить, что алгоритм выполняется на ВС или алгоритм реализуется на ВС.

Максимально быстрая реализация алгоритма

Для дальнейшего изложения предположим, что ВС имеет следующую модель:

1) процессоров достаточно для реализации алгоритма;
2) процессоры идентичны, выполняют все операции из множества Q;
3) время выполнения процессорами операций одинаково, это время будем называть *тактом;*
4) оперативная память достаточна для реализации алгоритма;
5) время обращения к памяти (выборки и записи информации) учитывается при оценке времени выполнения операции.

Здесь под процессором следует понимать любой вычислитель ВС. Условие 3) не является обязательным, оно лишь позволяет не загромождать дальнейшее изложение деталями, связанными с индивидуальными длительностями выполнения каждой из операций, мешающими пониманию сути исследования.

Максимально быстрая реализация алгоритма

Опишем некоторую реализацию алгоритма $\mathcal{A}$, представленного в форме Q-детерминанта.
Пусть $N = \varnothing$. Зададим интерпретацию переменных B.
Выражения

$$W = \left\{ w^{i_1}(i_1 \in I_1);\, u_j^{i_2}, w_j^{i_2}(i_2 \in I_2, j = 1, \ldots, I_{i_2});\right.$$
$$\left. u_j^{i_3}, w_j^{i_3}(i_3 \in I_3, j = 1, 2, \ldots) \right\}$$

будем вычислять одновременно (параллельно).
Будем говорить, что *операция готова к выполнению*, если определены значения ее операндов.
При вычислении каждого из выражений W на каждом такте будем выполнять все готовые к выполнению операции.
Если готовы к выполнению несколько операций цепочки, то они выполняются по схеме сдваивания.

Максимально быстрая реализация алгоритма

- Если для некоторой пары выражений (u_j^i, w_j^i) $(i \in I_2 \cup I_3, j = 1, 2, \dots)$ при вычислении установлено, что значение одного из выражений не определено, то вычисление другого выражения прекращается.
- Если для некоторого выражения u_j^i $(i \in I_2 \cup I_3, j = 1, 2, \dots)$ получено значение `false`, то вычисление соответствующего ему выражения w_j^i прекращается.
- Если для некоторой пары выражений $(u_{j_0}^i, w_{j_0}^i)$ $(i \in I_2 \cup I_3)$ в результате вычисления установлено, что их значения определены и $u_{j_0}^i$ принимает значение `true`, то вычисление выражений u_j^i, w_j^i $(i \in I_2 \cup I_3, j = 1, 2, \dots; j \neq j_0)$ прекращается.

Максимально быстрая реализация алгоритма

Пусть $N \neq \varnothing$. Зададим $\bar{N} \in \{\bar{N}\}$ и интерпретацию переменных B. Получаем множество выражений

$$W(\bar{N}) = \Big\{ w^{i_1}(\bar{N})(i_1 \in I_1); u_j^{i_2}(\bar{N}), w_j^{i_2}(\bar{N})(i_2 \in I_2, j = 1, \ldots, I_{i_2}); \\ u_j^{i_3}(\bar{N}), w_j^{i_3}(\bar{N})(i_3 \in I_3, j = 1, 2, \ldots) \Big\}.$$

Выражения $W(\bar{N})$ вычисляются по аналогии с W.
Описанную реализацию алгоритма $\mathcal{A}$ будем называть *максимально быстрой*.
Если максимально быстрая реализация является параллельной, то будем называть ее *максимально параллельной*.

Выполнимая реализация алгоритма

Будем говорить, что реализация алгоритма $\mathcal{A}$ *выполнима*, если при выполнении ее на ВС на каждом такте работы ВС требуется конечное число процессоров.

Существуют алгоритмы, для которых максимально быстрая реализация не является выполнимой.

Пример. Вычислить сумму ряда

$$\sum_{m=1}^{\infty}(-1)^m\frac{1}{m} \text{ с заданной точностью } \varepsilon < 1.$$

Q-детерминант алгоритма для вычисления суммы ряда S состоит из одного условного бесконечного Q-терма.

Выпишем представление алгоритма в форме Q-детерминанта

$$S = \left\{\left(\tfrac{1}{2} < \varepsilon, -1\right), \left(\tfrac{1}{3} < \varepsilon, -1 + \tfrac{1}{2}\right), \ldots \right.$$
$$\left. \ldots, \left(\tfrac{1}{m} < \varepsilon, -1 + \tfrac{1}{2} - \cdots + (-1)^{m-1}\tfrac{1}{m-1}\right), \ldots\right\}.$$

Для выполнения максимально быстрой реализации на первом такте требуется бесконечное число процессоров, так как к выполнению готово счетное множество операций деления.

Выполнимая реализация алгоритма

Приведем условия выполнимости максимально быстрой реализации алгоритма.
Пусть алгоритм $\mathcal{A}$ представлен в форме Q-детерминанта, $N = \varnothing$ и выполнены условия:
либо $I_3 = \varnothing$;
либо $I_3 \neq \varnothing$ и для любых i, r таких, что $i \in I_3, r = 1, 2, \dots,$ для выражений u_j^i, w_j^i $(j = 1, 2, \dots)$ множество всех различных выражений и подвыражений уровня вложенности r конечно.
Тогда максимально быстрая реализация алгоритма $\mathcal{A}$ выполнима.
Аналогичное утверждение справедливо для случая $N \neq \varnothing$.

Оценка сложности максимально быстрой реализации алгоритма

Для алгоритма $\mathcal{A}$, представленного в форме Q-детерминанта, введем характеристики:

1) $D_{\mathcal{A}}$ — число тактов работы ВС, требующихся при выполнении максимально быстрой реализации;
2) $P_{\mathcal{A}}$ — число процессоров ВС, требующихся при выполнении максимально быстрой реализации.

Оценка сложности максимально быстрой реализации алгоритма

Дадим верхние оценки для $D_{\mathcal{A}}$ и $P_{\mathcal{A}}$ в случае, когда $I_3 = \varnothing$.
Пусть $N = \varnothing$. Введем обозначение $T = \max\limits_{w \in W} T^w$,где

$$W = \{w^{i_1}(i_1 \in I_1); u_j^{i_2}, w_j^{i_2}(i_2 \in I_2, j = 1, \ldots, I_{i_2})\}.$$

Тогда $D_{\mathcal{A}} \leqslant T$, $P_{\mathcal{A}} \leqslant \max\limits_{1 \leqslant r \leqslant T} P_{\mathcal{A}}^r$, где $P_{\mathcal{A}}^r$ — число различных выражений и подвыражений уровня вложенности r множества W.
Пусть $N \neq \varnothing$. Введем обозначение $T(\bar{N}) = \max\limits_{w(\bar{N}) \in W(\bar{N})} T^w(\bar{N})$, где

$$W(\bar{N}) = \{w^{i_1}(\bar{N})(i_1 \in I_1); u_j^{i_2}(\bar{N}), w_j^{i_2}(\bar{N})(i_2 \in I_2, j = 1, \ldots, I_{i_2})\}.$$

Тогда $D_{\mathcal{A}} \leqslant T(\bar{N})$, $P_{\mathcal{A}} \leqslant \max\limits_{1 \leqslant r \leqslant T(\bar{N})} P_{\mathcal{A}}^r(\bar{N})$, где $P_{\mathcal{A}}^r(\bar{N})$ — число различных выражений и подвыражений уровня вложенности r множества $W(\bar{N})$.
Как видим, при $I_3 = \varnothing$ верхние оценки для $D_{\mathcal{A}}$ и $P_{\mathcal{A}}$ не зависят от интерпретации B.

Оценка сложности максимально быстрой реализации алгоритма

Рассмотрим случай, когда $I_3 \neq \varnothing$.
Предположим, что при любой интерпретации B и любом $\bar{N} \in \{\bar{N}\}$ (если $N \neq \varnothing$) значение Q-термов $f_i (i \in I_3)$ определено.
Кроме того предположим, что Q-детерминант удовлетворяет условиям выполнимости максимально быстрой реализации.
Пусть $N = \varnothing$. Тогда для заданной интерпретации B для $i \in I_3$ существует пара выражений $(u^i_{j_i}, w^i_{j_i})$, для которой $u^i_{j_i}$ принимает значение `true`, а значение $w^i_{j_i}$ определено.
Заметим, что j_i зависит от интерпретации B.

Оценка сложности максимально быстрой реализации алгоритма

Введем обозначение $\widetilde{T} = \max\limits_{w \in \widetilde{W}} T^w$, где

$$\widetilde{W} = \left\{ w^{i_1}(i_1 \in I_1); u_j^{i_2}, w_j^{i_2}(i_2 \in I_2, j = 1, \ldots, l_{i_2}); u_{j_{i_3}}^{i_3}, w_{j_{i_3}}^{i_3}(i_3 \in I_3) \right\}.$$

Тогда

$$D_{\mathcal{A}} \leqslant \widetilde{T}, \quad P_{\mathcal{A}} \leqslant \max_{1 \leqslant r \leqslant \widetilde{T}} P_{\mathcal{A}}^r,$$

где $P_{\mathcal{A}}^r$ — число различных выражений и подвыражений уровня вложенности r множества
$W = \{w^{i_1}(i_1 \in I_1); u_j^{i_2}, w_j^{i_2}(i_2 \in I_2, j = 1, \ldots, l_{i_2}); u_j^{i_3}, w_j^{i_3}(i_3 \in I_3, j = 1, 2, \ldots)\}$.
Таким образом, в рассмотренном случае оценки для $D_{\mathcal{A}}$ и $P_{\mathcal{A}}$ зависят от интерпретации B.
Аналогичные оценки справедливы для случая $N \neq \varnothing$.

Анализ ресурса распараллеливания некоторых алгоритмов

Рассмотрим алгоритм $\mathcal{A}$ для вычисления скалярного произведения векторов $\bar{a}^1 = (a_1^1, \ldots, a_n^1)$ и $\bar{a}^2 = (a_1^2, \ldots, a_n^2)$

$$(\bar{a}^1, \bar{a}^2) = \sum_{i=1}^{n} a_i^1 a_i^2.$$

Алгоритм представлен в форме Q-детерминанта.
Он состоит из одного безусловного Q-терма, уровень вложенности которого $\lceil \log n \rceil + 1$.
Здесь и далее $\log c$ — двоичный логарифм c, $\lceil x \rceil$ — ближайшее сверху целое к числу x.
Для данного алгоритма $\mathcal{A}$
$D_{\mathcal{A}} = \lceil \log n \rceil + 1$, $P_{\mathcal{A}} = n$, $N = \{n\}$, $B = \{a_1^1, a_1^2, \ldots, a_n^1, a_n^2\}$.

Анализ ресурса распараллеливания некоторых алгоритмов

Проведем анализ ресурса распараллеливания метода Гаусса–Жордана для решения систем линейных уравнений $A\bar{x} = \bar{b}$.
Пусть $A = [a_{ij}]$ — матрица размерности $n \times n$ с определителем $\neq 0$,
$\bar{x} = (x_1, \ldots, x_n)^T$, $\bar{b} = (a_{1,n+1}, \ldots, a_{n,n+1})^T$ — векторы–столбцы,
$\bar{A} = [a_{ij}]$ — расширенная матрица системы.
Метод Гаусса–Жордана состоит из n шагов.
Шаг 1.
В качестве ведущего элемента выберем a_{1j_1} с условием, что $a_{1j} = 0$ при $j < j_1 \leqslant n$, $a_{1j_1} \neq 0$.
Получим расширенную матрицу $\bar{A}^{j_1} = [a_{ij}^{j_1}]$, элементы которой вычисляются по формулам

$$a_{1j}^{j_1} = \frac{a_{1j}}{a_{1j_1}}, \; a_{ij}^{j_1} = a_{ij} - \frac{a_{1j}}{a_{1i_1}} a_{ij_1} (i = 2, \ldots, n; j = 1, \ldots, n+1).$$

Уровень вложенности выражений, являющихся правыми частями равенств, не превосходит 3.

Анализ ресурса распараллеливания некоторых алгоритмов

Метод Гаусса-Жордана

Шаг k $(2 \leqslant k \leqslant n)$.

После $(k-1)$-го шага получили расширенную матрицу $\bar{A}^{j_1 \dots j_{k-1}}$.

В качестве ведущего элемента выберем $a_{kj_k}^{j_1 \dots j_{k-1}}$ с условием, что $a_{kj}^{j_1 \dots j_{k-1}} = 0$ при $j < j_k \leqslant n$, $a_{kj_k}^{j_1 \dots j_{k-1}} \neq 0$.

Получим матрицу $\bar{A}^{j_1 \dots j_k} = [a_{ij}^{j_1 \dots j_k}]_{i=1,\dots,n; j=1,\dots,n+1}$, элементы которой вычисляются по формулам

$$a_{kj}^{j_1 \dots j_k} = \frac{a_{kj}^{j_1 \dots j_{k-1}}}{a_{kj_k}^{j_1 \dots j_{k-1}}}, \quad a_{ij}^{j_1 \dots j_k} = a_{ij}^{j_1 \dots j_{k-1}} - \frac{a_{kj}^{j_1 \dots j_{k-1}}}{a_{kj_k}^{j_1 \dots j_{k-1}}} a_{ij_k}^{j_1 \dots j_{k-1}}$$

$$(i = 1, \dots, n; i \neq k; j = 1, \dots, n+1).$$

Уровень вложенности выражений, являющихся правыми частями равенств, не превосходит $3k$.

Анализ ресурса распараллеливания некоторых алгоритмов

Метод Гаусса-Жордана

После шага n получаем систему уравнений $A^{j_1\dots j_n}\bar{x} = \bar{b}^{j_1\dots j_n}$, где

$$A^{j_1\dots j_n} = [a_{ij}^{j_1\dots j_n}]_{i=1,\dots,n;j=1,\dots,n},\ \bar{b}^{j_1\dots,j_n} = (a_{1,n+1}^{j_1\dots j_n},\dots,a_{n,n+1}^{j_1\dots j_n})^T.$$

Матрица $A^{j_1\dots j_n}$ такова, что
$a_{kj}^{j_1\dots j_n} = 0 (j = 1,\dots,n; j \neq j_k)$, $a_{kj_k}^{j_1\dots j_n} = 1$, $a_{k,n+1}^{j_1\dots j_n} (k = 1,\dots,n-1)$ являются выражениями уровня вложенности $3n$, а выражение $a_{n,n+1}^{j_1\dots j_n}$ имеет уровень вложенности $3n-2$.
Таким образом, $x_{j_k} = a_{k,n+1}^{j_1\dots j_n} (k = 1,\dots,n)$, при этом максимальное число уровней вложенности выражений для вычисления x_{j_k} равно $3n$.

Анализ ресурса распараллеливания некоторых алгоритмов

Метод Гаусса-Жордана

В зависимости от значений элементов матрицы A возможны $n!$ способов выбора $(j_1, \ldots, j_n)$, то есть $n!$ перестановок элементов $(1, \ldots, n)$. Занумеруем все перестановки от 1 до $n!$. Введем обозначения

$$L_{j_1} = \bigwedge_{j=1}^{j_1-1} (a_{1j} = 0), \quad L_{j_l} = \bigwedge_{\substack{j=1 \\ j \notin \{j_1, \ldots, j_{l-1}\}}}^{j_l-1} (a_{ij}^{j_1 \ldots j_{l-1}} = 0).$$

Пусть i — номер перестановки $(j_1, \ldots, j_n)$. Тогда

$$w_i^{j_l} = a_{l,n+1}^{j_1 \ldots j_n} (l = 1, \ldots, n), u_i = L_{j_1} \wedge (a_{1j_1} \neq 0) \wedge \left(\bigwedge_{l=2}^{n} \left(L_{j_l} \wedge \left(a_{lj_l}^{j_1 \ldots j_{l-1}} \neq 0 \right) \right) \right)$$

можно рассматривать, как Q-термы при условии, что $N = \{n\}$, $B = \{a_{ij}\}(i = 1, \ldots, n; j = 1, \ldots, n+1)$.

Анализ ресурса распараллеливания некоторых алгоритмов

Метод Гаусса-Жордана

Q-детерминант описанного алгоритма состоит из n условных Q-термов, а представление в форме Q-детерминанта имеет вид:

$$x_j = \{(u_1, w_1^j), \ldots, (u_{n!}, w_{n!}^j)\}(j = 1, \ldots, n).$$

Оценим сложность алгоритма.

Пусть i_0 — номер перестановки $(n, n-1, \ldots, 1)$.

Уровень вложенности каждого Q-терма $u_i (i = 1, \ldots, n!; i \neq i_0)$ не превосходит уровня вложенности Q-терма u_{i_0}.

Можно доказать, что $T^{u_{i_0}} = 3n - 1$.

Как было доказано, $T^{w_i^j} \leqslant 3n (i = 1, \ldots, n!, j = 1, \ldots, n)$, причем для некоторых Q-термов w_i^j имеет место равенство.

Таким образом, для описанного алгоритма $\mathcal{A}$ имеем $D_{\mathcal{A}} = 3n$.

Если ведущие элементы выбирать по условию

$$|a_{1j_1}| = \max_{1 \leqslant j \leqslant n} |a_{1j}|, \ \left|a_{lj_l}^{j_1 \ldots j_{l-1}}\right| = \max_{\substack{1 \leqslant j \leqslant n \\ j \notin \{j_1, \ldots, j_{l-1}\}}} |a_{lj}^{j_1 \ldots j_{l-1}}|,$$

то также $D_{\mathcal{A}} = 3n$.

Анализ ресурса распараллеливания некоторых алгоритмов

Метод Гаусса-Жордана

Оценим число процессоров, необходимых для реализации алгоритма $\mathcal{A}$. Заметим, что при любом n выражение $u_i (i = 1, \ldots, n!)$ можно представить в виде $u_i = u_i^1 \wedge u_i^2 \wedge \cdots \wedge u_i^{s_i}$, где u_i^s $(s = 1, \ldots, s_i)$ — некоторые выражения, $s_i \leqslant j_1 + j_2 + \cdots + j_n, (j_1, \ldots, j_n)$ — перестановка с номером i.

При выполнении максимально быстрой реализации метода Гаусса-Жордана будем предполагать, что если при вычислении u_i для некоторого s $(1 \leqslant s \leqslant s_i)$ u_i^s принимает значение `false`, то вычисление u_i, w_i^j $(j = 1, \ldots, n)$ прекращается.

При такой организации вычислений для выполнения максимально быстрой реализации самое большое число процессоров требуется на первом такте работы ВС, оно составляет $n^2 + 2n - 1$.

Если же вычислять u_i $(i = 1, \ldots, n!)$ до конца, то требуется процессоров не менее $2(n-1)n!$, то есть $P_{\mathcal{A}} \geqslant 2(n-1)n!$.

Анализ ресурса распараллеливания некоторых алгоритмов

Решение системы сеточных уравнений методом Якоби

Предположим дана пятиточечная СЛАУ

$$u_{ij} = \frac{1}{e_{ij}}(f_{ij}+a_{ij}u_{i-1,j}+b_{i,j-1}u_{i,j-1}+c_{ij}u_{i+1,j}+d_{ij}u_{i,j+1})(1 \leqslant i \leqslant L, 1 \leqslant j \leqslant M),$$

где u_{ij} — значения сеточной функции, $a_{ij}, b_{ij}, c_{ij}, d_{ij}, e_{ij}, f_{ij}$ — константы.
Рассмотрим ее решение методом простой итерации

$$u_{ij}^{n} = \frac{1}{e_{ij}}(f_{ij} + a_{ij}u_{i-1,j}^{n-1} + b_{i,j-1}u_{i,j-1}^{n-1} + c_{ij}u_{i+1,j}^{n-1} + d_{ij}u_{i,j+1}^{n-1})(n = 1, 2, \dots),$$

где u_{ij}^{0} — это начальные приближенные значения сеточной функции, u_{ij}^{n} $(n \geqslant 1)$ — приближенные значения сеточной функции, полученные на n-м шаге итерации.
Выразив значения сеточной функции $(n-1)$-го шага итерации через значения $(n-2)$-го шага, можно представить u_{ij}^{n} через

$$u_{kl}^{n-2}(|k-i| + |l-j| \leqslant 2, 1 \leqslant k \leqslant L, 1 \leqslant l \leqslant M).$$

Продолжив этот процесс, можно выразить u_{ij}^{n} через

$$u_{kl}^{0}(|k-i| + |l-j| \leqslant n, 1 \leqslant k \leqslant L, 1 \leqslant l \leqslant M).$$

Анализ ресурса распараллеливания некоторых алгоритмов

Решение системы сеточных уравнений методом Якоби

Условием окончания итерационного процесса является

$$||u^n - u^{n-1}|| < \varepsilon,$$

или

$$v^n = \bigwedge_{1 \leqslant i \leqslant L; 1 \leqslant j \leqslant M} (|u_{ij}^n - u_{ij}^{n-1}| < \varepsilon)$$

принимает значение `true`.

Выражение v^n также определяется через u_{kl}^0 $(1 \leqslant k \leqslant L, 1 \leqslant l \leqslant M)$.

Выражения v^n и u_{ij}^n $(1 \leqslant i \leqslant L, 1 \leqslant j \leqslant M, n \geqslant 1)$ являются Q-термами, при этом $N = \{L, M\}$, $B = \{u_{kl}^0\}(k = 1, \ldots, L; l = 1, \ldots, M)$.

Q-детерминант метода простой итерации состоит из LM условных бесконечных Q-термов, а представление в форме Q-детерминанта имеет вид

$$u_{ij} = \{(v^1, u_{ij}^1), (v^2, u_{ij}^2), \ldots, (v^n, u_{ij}^n), \ldots\},$$

где $1 \leqslant i \leqslant L, 1 \leqslant j \leqslant M, n = 1, 2, \ldots$.

Анализ ресурса распараллеливания некоторых алгоритмов

Решение системы сеточных уравнений методом Якоби

Q-детерминант удовлетворяет условию выполнимости, поэтому максимально быстрая реализация алгоритма выполнима.
На каждом такте работы ВС занято процессоров не более

$$LM\left\lceil 5+\frac{1}{8}\left(1+\frac{1}{32}+\cdots+\frac{1}{32^k}\right)\right\rceil, \text{где } k=\left\lceil \log_{32}\frac{LM}{8}\right\rceil=\left\lceil\frac{\log(LM)-3}{5}\right\rceil,$$

поэтому

$$P_{\mathcal{A}}\leqslant LM\left(5+\frac{4(32^{k+1}-1)}{31\cdot 32^{k+1}}\right)<6LM.$$

Как только для некоторого n_0 значение v^{n_0} будет равно `true`, выполнение максимально быстрой реализации необходимо закончить и в качестве u_{ij} взять вычисленные значения Q-термов $u_{ij}^{n_0}$, при этом

$$D_{\mathcal{A}}=5n_0+3+\lceil\log LM\rceil.$$

Автоматизированный анализ ресурса распараллеливания алгоритмов

- Алгоритм может быть не представлен в форме Q-детерминанта, поэтому получение Q-детерминанта алгоритма и его максимально быстрой реализации может являться отдельной непростой задачей.
- В 2010 г. С.В. Игнатьев показал, как по блок-схеме алгоритма получить Q-детерминант (ВКР магистра http://omega.sp.susu.ac.ru/publications/masterthesis/10-Ignatyev.pdf). Это позволяет реализовать получение Q-детерминанта программными средствами.
- При выполнении максимально быстрой реализации алгоритма на реальной вычислительной системе необходимо учесть ограничение на количество процессоров (вычислительных узлов).
- В связи с этим в 2012 г. Д.И. Свирихиным было введено и исследовано понятие *максимально эффективной реализации*, которая имеет минимальное время выполнения при заданном ограничении на количество процессоров, обеспечивая при этом максимально возможную загрузку процессоров (ВКР магистра http://omega.sp.susu.ac.ru/publications/masterthesis/13-Svirikhin.pdf).

Автоматизированный анализ ресурса распараллеливания алгоритмов

В настоящее время в Южно-Уральском государственном университете разрабатывается программная система QStudio для анализа ресурса распараллеливания алгоритмов.
Первая версия системы QStudio реализует следующие функции:

1) построение Q-детерминанта алгоритма по блок-схеме;
2) поиск на основе Q-детерминанта максимально быстрой реализации алгоритма и построение ее плана выполнения;
3) поиск максимально эффективной реализации алгоритма на основе максимально быстрой реализации и построение ее плана выполнения.

План выполнения представляет собой направленный уровневый граф, где нулевой уровень содержит входные данные алгоритма, а каждый ненулевой уровень соответствует такту работы ВС и содержит все операции алгоритма, выполняющиеся на данном такте параллельно.
На рисунке показан граф плана выполнения максимально быстрой реализации алгоритма скалярного произведения векторов размерности 9. Граф получен с помощью системы Qstudio.

Автоматизированный анализ ресурса распараллеливания алгоритмов

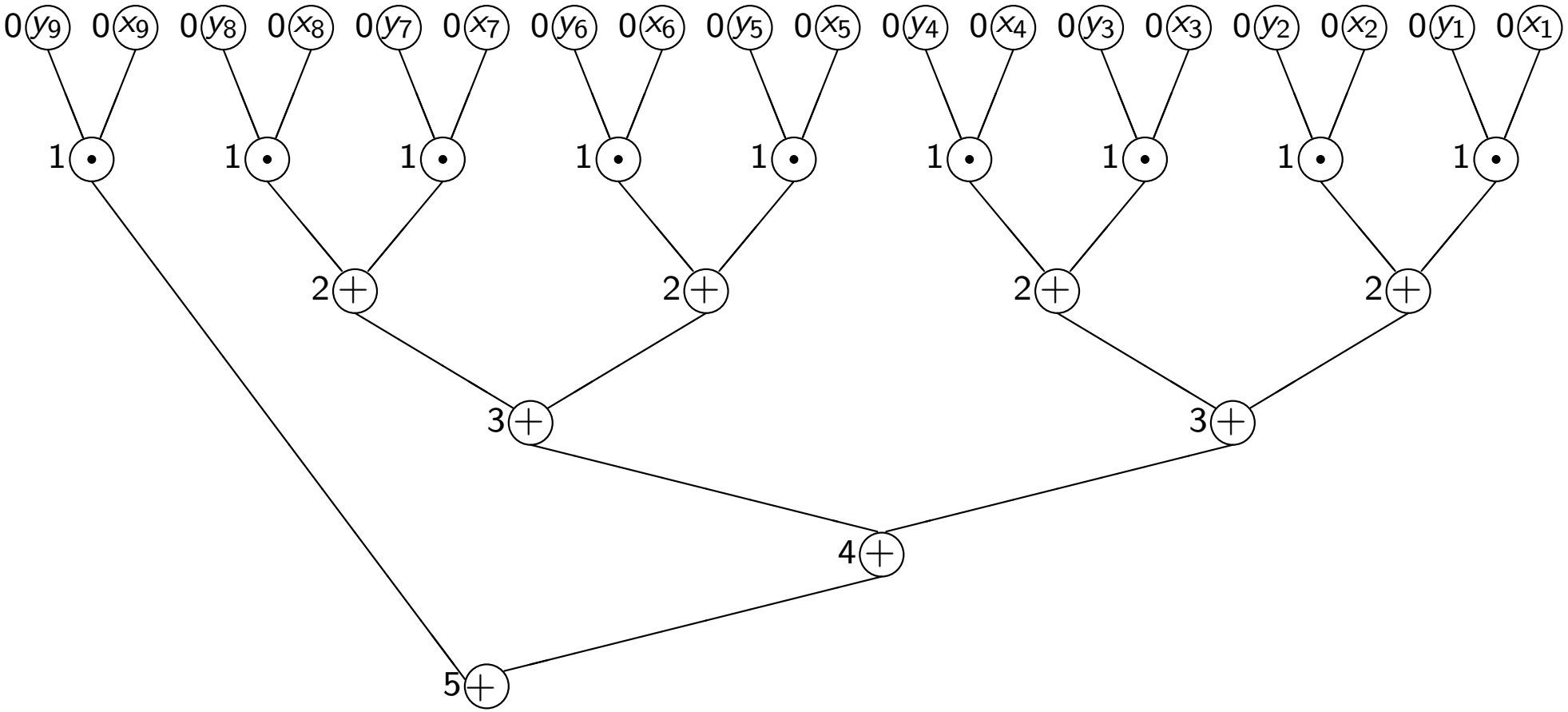

Рис.: Граф плана выполнения максимально быстрой реализации алгоритма скалярного произведения векторов размерности 9

Автоматизированный анализ ресурса распараллеливания алгоритмов

- В дальнейшем планируется для плана выполнения генерировать программный код.
- Подробно описание первой версии системы QStudio изложено в ВКР Д.Э. Сулейманова (http://omega.sp.susu.ac.ru/publications/masterthesis/15-Suleymanov.pdf).
- Если известен Q-детерминант алгоритма, то его структура полностью открыта для исследования, поэтому с помощью программной системы QStudio можно автоматизировать анализ всевозможных свойств алгоритмов.

Автоматизированный анализ ресурса распараллеливания алгоритмов

- Проведенные эксперименты показали, что система *QStudio*, используя блок-схему, находит Q-детерминант метода Гаусса-Жордана и строит план выполнения его максимально быстрой реализации:
  при размерности матрицы **1000** примерно за **2 секунды**,
  при размерности матрицы **1000000** (реально таких матриц не существует, но расчет дает представление о быстродействии) примерно за **18 секунд**.
- Эксперименты проводились на ПК, имеющем характеристики: процессор Intel Core i5 3.2Ггц, RAM 8Гб.

Автоматизированный анализ ресурса распараллеливания алгоритмов

- В последнее время аспирантом ЮУрГУ Д.Э. Сулеймановым с целью расширения функциональных возможностей QStudio разрабатывается новый метод получения Q-детерминанта алгоритма на основе абстрактного синтаксического дерева исходного кода программы, реализующей алгоритм.
- Работа, в которой изложены полученные в настоящее время результаты данного исследования, подана на конференцию ПаВТ'2016.

Заключение

- Q-детерминант делает алгоритм прозрачным с точки зрения структуры и реализации, он показывает все его реализации, в том числе максимально быструю.
- Однако максимально быстрая реализация может плохо согласовываться с архитектурой реальной ВС.

  В результате время ее выполнения на реальной ВС может оказаться больше, чем какой-либо другой реализации.
- Для того, чтобы на конкретной ВС конкретный алгоритм можно было выполнить за минимально возможное для данной ВС и данного алгоритма время, нужно, используя Q-детерминант алгоритма, решить задачу оптимального наложения структуры алгоритма на архитектуру ВС, то есть из всех реализаций алгоритма найти ту, которая лучше всего согласуется с архитектурой конкретной ВС.
- Таким образом, прозрачность алгоритмов с точки зрения структуры и реализации, обеспечиваемая с помощью концепции Q-детерминанта, позволит выполнять алгоритмы на реальных ВС быстрее, чем мы их выполняем в условиях ее отсутствия.

Спасибо за внимание!