Максимальное распараллеливание алгоритмов на основе концепции Q-детерминанта

В.Н. Алеева

Кафедра системного программирования
Южно-Уральский государственный университетет (НИУ)

Доклад на заседании научного семинара по информационным технологиям под руководством профессора Л.Б. Соколинского

Челябинск, 2015

- В настоящее время мною совместно с группой студентов реализуется проект, который мы назвали

  MPA (Maximum Parallelization of Algorithms).

- Целью проекта **MPA** является автоматизированное выполнение максимально быстрых реализаций алгоритмов на параллельных вычислительных системах (ВС).

Для достижения поставленной цели необходимо решить следующие задачи:

- разработать программную систему **QStudio** для анализа ресурса распараллеливания любого алгоритма, в том числе для нахождения максимально быстрой реализации и построения ее плана выполнения на параллельных ВС;
- разработать и внедрить технологию выполнения максимально быстрой реализации алгоритма на параллельных ВС.

В основе исследований, проводимых в рамках проекта **MPA**, лежит концепция Q-детерминанта, основополагающим понятием которой является представление алгоритма в форме Q-детерминанта.

Причина появления концепции Q-детерминанта:

- Концепция была предложена мною в 1985 году.
- В то время в связи с появлением параллельных ВС проводились работы по распараллеливанию программ с целью использования имеющегося программного обеспечения.
- Однако, если решать задачу максимального распараллеливания алгоритма, то использовать реализующую его программу не целесообразно, т.к. она может не содержать всех реализаций алгоритма, в частности, может быть потеряна максимально быстрая реализация, поэтому более правильный подход — распараллеливание самого алгоритма.
- В результате стали появляться работы по распараллеливанию конкретных численных алгоритмов, в том числе авторами таких работ были и известные учёные В.Н. Фаддеева, Д.К. Фаддеев, Г.И. Марчук и другие.

- В качестве примера можно привести публикации:
  Фаддеева, В.Н. Параллельные вычисления в линейной алгебре / В.Н. Фаддеева, Д.К. Фаддеев // Кибернетика. — 1977 . — № 6. — С. 28–40.
  Фаддеева, В.Н. Параллельные вычисления в линейной алгебре / В.Н. Фаддеева, Д.К. Фаддеев // Кибернетика. — 1982 . — № 3. — С. 18–31.
  Марчук, Г.И. Параллельные вычисления в сеточных методах решения задач математической физики: Препринт № 220 / Г.И. Марчук, В.П. Ильин — Новосибирск: ВЦ СО АН СССР, 1979. — 22 с.
- При этом универсального подхода к распараллеливанию алгоритмов не существовало.
  Концепция Q-детерминанта явилась решением этой задачи.

Кратко о концепции Q-детерминанта:

- Q-детерминант алгоритма описывает все возможные реализации алгоритма, в том числе позволяет выявить максимально быструю.
- Предложенный в концепции анализ максимально быстрой реализации алгоритма дает возможность получить оценки сложности: число процессоров и число тактов работы ВС, необходимых для ее выполнения.
- В результате можно исследовать ресурс распараллеливания любого алгоритма, сравнивать ресурсы распараллеливания алгоритмов, решающих одну и ту же задачу, и выбирать алгоритм с максимальных ресурсом распараллеливания.

Кратко о концепции Q-детерминанта:

- Концепция Q-детерминанта — один из результатов моей кандидатской диссертации.
- Концепцию можно было использовать для алгоритма, если он был представлен в форме Q-детерминанта, однако она не содержала доказательство, что любой алгоритм может быть представлен в форме Q-детерминанта.
- В 2010 году магистрант С.В. Игнатьев в ВКР магистра показал, что по блок-схеме любого алгоритма можно получить Q-детерминант.
- В 2013 году магистрант Д.И. Свирихин ввел понятие максимально эффективной реализации алгоритма — реализации, имеющей минимальное время выполнения при заданном ограничении на количество процессоров, и предложил алгоритм ее получения.
- Идеи, предложенные в перечисленных работах, используются в проекте **MPA**.

В настоящее время реализована и проходит опытную эксплуатацию следующая основная функциональность системы **Qstudio**:

- Построение Q-детерминанта алгоритма.
  При этом в качестве исходного представления алгоритма используется блок-схема. (И. Шарабура)
- Поиск на основе Q-детерминанта максимально быстрой реализации алгоритма и построение ее плана выполнения. (Д. Сулейманов)
- Поиск максимально эффективной реализации алгоритма на основе максимально быстрой реализации и построение ее плана выполнения. (Д. Сулейманов)

Планируется реализовать следующую функциональность системы **Qstudio**:

- Получение зависимости оценок сложности алгоритма от размерности задачи, для решения которой алгоритм предназначен.
- Сравнение зависимостей оценок сложности различных алгоритмов, решающих одну и ту же задачу.
- Получение зависимости количества тактов работы ВС при выполнении максимально эффективной реализации алгоритма от размерности задачи.
- Сравнение при разных ограничениях на количество процессоров ВС зависимостей количества тактов работы ВС при выполнении максимально эффективной реализации алгоритма.
- Поиск для любой задачи алгоритма, который обладает лучшим потенциалом распараллеливания, чем исходный, путём преобразования Q-детерминанта исходного алгоритма в Q-детерминант искомого алгоритма.

Список функциональностей по мере развития проекта может пополняться.

- Функциональность программной системы **QStudio** перечислена.
- О других аспектах системы **Qstudio** предлагается сделать отдельный доклад. (Д. Сулейманов)
- Далее будет изложена концепция Q-детерминанта.

План изложения концепции Q-детерминанта

- Выражение над множествами B и Q, уровни вложенности выражения, цепочка выражений,
  где B — множество переменных, используемых алгоритмом,
  Q — множество операций, используемых алгоритмом.
- Вычисление выражения.
- Q-терм: безусловный, безусловный логический, условный, условный бесконечный.
- Вычисление Q-терма.
- Q-детерминант алгоритма.
- Максимально быстрая реализация алгоритма.
- Выполнимая максимально быстрая реализация, пример невыполнимой.
- Условия выполнимости максимально быстрой реализации.
- Оценки сложности максимально быстрой реализации.
- Анализ ресурса распараллеливания некоторых алгоритмов с помощью концепции Q-детерминанта.

Множества B и Q

- Пусть $B = \{b_1, b_2, \dots\}$ — счетное множество переменных арифметического или логического типа.
- Q — конечное множество операций, например, следующих:
  - арифметические операции: $+$ (сложение), $-$ (вычитание или изменение знака), $\times$ (умножение), $/$ (деление), $|\ |$ (абсолютная величина числа);
  - операции сравнения: $<$ (меньше), $>$ (больше), $=$ (равно), $\neq$ (не равно), $\leqslant$ (меньше или равно), $\geqslant$ (больше или равно);
  - логические операции: $\neg$ (логическое отрицание), $\wedge$ (логическое умножение (конъюнкция)), $\vee$ (логическое сложение (дизъюнкция)).

  Q будем называть *множеством базовых операций.*
- Далее значение логического типа «истина» будем обозначать единицей, а «ложь» — нулем.

Выражение над множествами B и Q, уровни вложенности

- Константа и $b_i \in B$ являются выражениями. Их будем называть *выражениями нулевого уровня вложенности*.
- Выражение, заключённое в круглые скобки, является выражением. Заключение выражения в круглые скобки не меняет его уровня вложенности.
- Применяя одноместную базовую операцию к выражению $(i-1)$-го $(i \geqslant 1)$ уровня вложенности, получаем выражение i-го уровня вложенности, а исходное выражение называется его подвыражением $(i-1)$-го уровня вложенности. Саму базовую операцию будем называть в этом случае *операцией i-го уровня вложенности*.
- Применяя двухместную базовую операцию к выражениям соответственно i_1 и i_2 уровней вложенности, получаем выражение i-го уровня вложенности, где $i = \max(i_1, i_2)$. Исходные выражения называются его подвыражениями соответственно i_1 и i_2 уровней вложенности. Саму базовую операцию будем называть *операцией i-го уровня вложенности*.
- Других выражений нет.

Примеры выражений:

- $b_1 \times (b_2 + b_3)/b_4$ — уровень вложенности 3;
- $((b_1 + b_2) \leqslant |b_3 \times b_4|) \vee \neg b_5$ — уровень вложенности 4;
- $(b_1 \geqslant b_3) \wedge ((b_2 - b_4) \neq 0) \wedge (b_5 = 0)$ — уровень вложенности 4.

Цепочка выражений

- Выражение, образованное в результате применения к n выражениям $n-1$ раз одной из ассоциативных операций, называется *цепочкой выражений длины* n.
- Примеры цепочек:

1) $b_1 + b_2 + b_3 + b_4$;
2) $(b_1 + b_2) \times (b_4 - b_3) \times (b_2 + b_4)$;
3) $b_1 \vee b_2 \vee (b_3 \geqslant b_5) \vee (b_2 \wedge \neg b_4)$.

Вычисление выражения

- *Интерпретацией переменной* $b_i \in B$ будем называть присваивание переменной b_i конкретного значения.
- Если задана интерпретация всех переменных, входящих в выражение, то будем говорить, что задана *интерпретация выражения*.
- Если задана некоторая интерпретация выражения, то его можно вычислить.

При вычислении выражения необходимо соблюдать следующий порядок выполнения базовых операций:

1) круглые скобки определяют порядок выполнения операций;
2) если круглые скобки не определяют полностью порядок выполнения операций, то операции выполняются в соответствии с их приоритетами;
3) при выполнении операций с одинаковым приоритетом, за исключением операций, образующих цепочки, выражение просматривается слева направо и предпочтение отдаётся операциям, стоящим левее;
4) операции цепочки могут выполняться в произвольном порядке, но при определении уровня вложенности выражения и его подвыражений порядок выполнения операций цепочки задаётся по схеме fan-in;
5) если при вычислении выражения обнаруживается, что необходимо выполнить операцию деления на ноль, то будем считать, что значение выражения не определено.

Определение Q-терма

- Пусть N — множество параметров размерности проблемы, которую решает алгоритм.
- Тогда 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.
- Пусть $N = \{n_1, \ldots, n_k\}$. Тогда 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 $\bar{N}$ будем обозначать $\{\bar{N}\}$.
- Если $N = \varnothing$, то любое выражение над B и Q будем называть *безусловным Q-термом*.

  Таким образом, при $N = \varnothing$ понятие безусловного Q-терма и выражения над B и Q совпадают.
- Пусть V — множество, состоящее из 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*.

Определение Q-терма

- Примеры безусловных Q-термов:

1) $b_1 + b_2 \times b_3 - b_4 (N = \varnothing)$;
2) $|b_1 + b_2 + \cdots + b_n| (N = \{n\})$;
3) $(b_1 = 0) \wedge \cdots \wedge (b_{1+2n_1} = 0) \wedge ((b_2 \times \cdots \times b_{2+2n_2}) > 0)$ $(N = \{n_1, n_2\})$.

Определение Q-терма

- 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 $N \neq \varnothing$.
- Если $N = \varnothing$, то *уровнем вложенности безусловного Q-терма w* будем называть уровень вложенности выражения w и обозначать T^w.
- Если $N \neq \varnothing$, то *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 $w(\bar{N})$.

Определение Q-терма

- Пусть $N = \varnothing$. Тогда, если выражение w при любой интерпретации B принимает значение логического типа, то безусловный Q-терм w будем называть *безусловным логическим Q-термом*.
- Пусть $N \neq \varnothing$. Если при любом $\bar{N} \in \{\bar{N}\}$ и любой интерпретации переменных B $w(\bar{N})$ принимает значение логического типа, то w будем называть *безусловным логическим Q-термом*.
- Пусть $u_1, \ldots, u_\ell$ — безусловные логические Q-термы, $w_1, \ldots, w_\ell$ — безусловные Q-термы.
  Множество ℓ пар $(u_i, w_i)(i = 1, \ldots, \ell)$ будем обозначать $(\bar{u}, \bar{w}) = \{(u_i, w_i)\}_{i=1,\ldots,\ell}$ и называть *условным Q-термом длины ℓ*.

Определение Q-терма

Опишем вычисление условного Q-терма $(\bar{u}, \bar{w})$ при заданной интерпретации B:

- Если $N = \varnothing$, то следует вычислить выражения $u_i, w_i (i = 1, \ldots, \ell)$.
- Если существуют выражения $u_{i_j}, w_{i_j} (j = 1, \ldots, s; s \leqslant \ell)$ такие, что значение u_{i_j} равно единице, а значение w_{i_j} определено, то будем считать, что $(\bar{u}, \bar{w})$ принимает значения w_{i_j}.
- В противном случае считаем, что значение $(\bar{u}, \bar{w})$ при данной интерпретации B не определено.

Определение Q-терма

- Если $N \neq \varnothing$, то зададим $\bar{N} \in \{\bar{N}\}$.
- Получим выражения $u_i(\bar{N}), w_i(\bar{N})(i = 1, \ldots, \ell)$ и вычислим их.
- Если существуют выражения $u_{i_j}(\bar{N}), w_{i_j}(\bar{N})(j = 1, \ldots, s; s \leqslant \ell)$ такие, что значение $u_{i_j}(\bar{N})$ равно единице, а значение $w_{i_j}(\bar{N})$ определено, то будем считать, что $(\bar{u}, \bar{w})$ принимает значения $w_{i_j}(\bar{N})$.
- В противном случае считаем, что значение $(\bar{u}, \bar{w})$ для данного $\bar{N}$ и данной интерпретации B не определено.

Определение Q-терма

- Счётное множество пар безусловных Q-термов $(\bar{u}, \bar{w}) = \{(u_i, w_i)\}_{i=1,2,\ldots}$ будем называть *условным бесконечным Q-термом*, если для любого $\ell < \infty$ $\{(u_i, w_i)\}_{i=1,\ldots,\ell}$ является условным Q-термом.

Определение Q-терма

Опишем вычисление условного бесконечного Q-терма $(\bar{u}, \bar{w})$ при заданной интерпретации B:

- Предположим, что $N = \varnothing$. Тогда, чтобы вычислить условный бесконечный Q-терм $(\bar{u}, \bar{w}) = \{(u_i, w_i)\}_{i=1,2,\dots}$, необходимо, вычисляя выражения $u_i, w_i (i = 1, 2, \dots)$, найти u_{i_0}, w_{i_0} такие, что значение u_{i_0} равно единице, 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 $N \neq \varnothing$.
- Если не имеет значения, является ли Q-терм безусловным, условным или условным бесконечным, то будем называть его *Q-термом*.

Определение Q-детерминанта алгоритма

- Пусть $\mathcal{A}$ — некоторый алгоритм для решения алгоритмической проблемы $\bar{y} = F(N, B)$,
  где N — множество параметров размерности проблемы,
  B — множество входных данных,
  $\bar{y} = (y_1, \ldots, y_m)$ — множество выходных данных, при этом $y_i \notin B (i = 1, \ldots, m)$,
  m является либо вычислимой функцией параметров N при $N \neq \varnothing$, либо константой.

Определение Q-детерминанта алгоритма

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, $f_{i_2} = \left\{ \left(u_j^{i_2}, w_j^{i_2} \right) \right\}_{j=1,\ldots,\ell_{i_2}}$, 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, $f_{i_3} = \left\{ \left(u_j^{i_3}, w_j^{i_3} \right) \right\}_{j=1,2,\ldots}$.

Определение Q-детерминанта алгоритма

- 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.

Примеры представления алгоритмов в форме Q-детерминанта будут приведены ниже.

Определение максимально быстрой реализации алгоритма, представленного в форме Q-детерминанта

- Реализацией алгоритма $\mathcal{A}$, представленного в форме Q-детерминанта $y_i = f_i(i = 1, \ldots, m)$, будем называть вычисление Q-термов $f_i(i = 1, \ldots, m)$.
- 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 реализация алгоритма выполняется на ВС или алгоритм реализуется на ВС.

Определение максимально быстрой реализации алгоритма, представленного в форме Q-детерминанта

- При дальнейшем изложении будем предполагать, что ВС имеет идеальную модель:

1) число процессоров не ограничено;
2) процессоры идентичны, выполняют все базовые операции;
3) время выполнения процессорами базовых операций одинаково, это время будем называть тактом;
4) оперативная память не ограничена;
5) время выборки информации из памяти несущественно по сравнению со временем ее обработки на процессоре.

Определение максимально быстрой реализации алгоритма, представленного в форме Q-детерминанта

- Опишем некоторую реализацию алгоритма $\mathcal{A}$, представленного в форме Q-детерминанта:

1) Рассмотрим сначала случай, когда $N = \varnothing$.

   Зададим интерпретацию переменных B.

   Выражения

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

   будем вычислять одновременно (параллельно).

   Будем говорить, что базовая 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 fan-in.

Определение максимально быстрой реализации алгоритма, представленного в форме Q-детерминанта

- Если для некоторого выражения $u_j^i (i \in I_2 \cup I_3, j = 1, 2, \dots)$ получено значение ноль, то вычисление соответствующего ему выражения w_j^i прекращается.
- 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 некоторой пары выражений $(u_{j_0}^i, w_{j_0}^i)(i \in I_3)$ в результате вычисления установлено, что их значения определены и значение $u_{j_0}^i$ равно единице, то вычисление выражений $u_j^i, w_j^i (i \in I_3, j = 1, 2, \dots; j \neq j_0)$ прекращается.
- Если в W встречаются одинаковые выражения и подвыражения, то достаточно вычислить одно из них, то есть выполнять вычисления без дублирования.

Определение максимально быстрой реализации алгоритма, представленного в форме Q-детерминанта

2) Рассмотрим случай, когда $N \neq \varnothing$.
Зададим $\bar{N} \in \{\bar{N}\}$ и интерпретацию переменных B.
Получаем множество выражений

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

Выражения $W(\bar{N})$ вычисляются по аналогии с W.

Определение максимально быстрой реализации алгоритма, представленного в форме Q-детерминанта

- Описанную реализацию алгоритма $\mathcal{A}$ будем называть максимально быстрой.
- Если максимально быстрая реализация является параллельной, то будем называть ее максимально параллельной.
- Будем говорить, что реализация алгоритма $\mathcal{A}$ выполнима, если при выполнении ее на ВС на каждом такте работы ВС требуется конечное число процессоров.

Определение максимально быстрой реализации алгоритма, представленного в форме Q-детерминанта

- 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.

Условия выполнимости максимально быстрой реализации

- 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 $i \in I_3, r = 1, 2, \ldots,$ для выражений $u_j^i, w_j^i (j = 1, 2, \ldots)$ 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

- Для алгоритма $\mathcal{A}$, представленного в форме Q-детерминанта, введём следующие 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

- 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

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

Тогда $D_{\mathcal{A}} \leqslant T$, $P_{\mathcal{A}} \leqslant \max\limits_{1 \leqslant r \leqslant T} P_A^r$, где $P_{\mathcal{A}}^r$ — число различных выражений и подвыражений уровня вложенности r для 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

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

Тогда $D_{\mathcal{A}} \leqslant T(\bar{N})$, $P_{\mathcal{A}} \leqslant \max\limits_{1 \leqslant r \leqslant T(\bar{N})} P_A^r(\bar{N})$, где $P_{\mathcal{A}}^r(\bar{N})$ — число различных выражений и подвыражений уровня вложенности r для 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

- 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.

Оценка сложности максимально быстрой реализации

Пусть $N = \varnothing$.

Тогда при заданной интерпретации B для любого $i \in I_3$ существует пара выражений $(u^i_{j_i}, w^i_{j_i})$, для которой значение $u^i_{j_i}$ равно единице, а значение $w^i_{j_i}$ определено.

Заметим, что j_i зависит от интерпретации B.

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

$$\widetilde{W} = \left\{ w^{i_1}(i_1 \in I_1);\, u^{i_2}_j, w^{i_2}_j (i_2 \in I_2, j = 1, \ldots, \ell_{i_2});\, u^{i_3}_{j_{i_3}}, w^{i_3}_{j_{i_3}} (i_3 \in I_3 \right\}.$$

Тогда

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

где $P^r_{\mathcal{A}}$ — число различных выражений и подвыражений уровня вложенности r для множества

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

Как видим, 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

- **Вычисление скалярного произведения векторов**

  Для векторов $\bar{a}^1 = (a_1^1, \ldots, a_n^1)$ и $\bar{a}^2 = (a_1^2, \ldots, a_n^2)$ вычислить их скалярное произведение

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

  В данном случае алгоритм представлен в форме Q-детерминанта.

  Q-детерминант 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 описанного 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

- **Решение системы линейных уравнений методом Гаусса-Жордана**

  Решить систему линейных уравнений $A\bar{x} = \bar{b}$,

  где $A = [a_{ij}]_{i,j=1,\ldots,n}$ — матрица размерности $n \times n$, $detA \neq 0$,
  $\bar{x} = (x_1, \ldots, x_n)^T$, $\bar{b} = (a_{1,n+1}, \ldots, a_{n,n+1})^T$.

  Пусть $\bar{A} = [a_{ij}]_{i=1,\ldots,n;j=1,\ldots,n+1}$ — расширенная матрица системы.

  Метод Гаусса-Жордана состоит из n шагов.

  Шаг 1. Найдем ведущий элемент a_{1j_1}, удовлетворяющий условию $a_{1j} = 0$ при $j < j_1 \leqslant n, a_{1j_1} \neq 0$.

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

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

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

Анализ ресурса распараллеливания некоторых алгоритмов

- **Решение системы линейных уравнений методом Гаусса-Жордана**

  Шаг $k(2 \leqslant k \leqslant n)$. Систему уравнений, полученную после $(k-1)$-го шага, обозначим $A^{j_1 \dots j_{k-1}} \bar{x} = \bar{b}^{j_1 \dots j_{k-1}}$, где

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

  Найдем ведущий элемент $a_{kj_k}^{j_1 \dots j_{k-1}}$, удовлетворяющий условию $a_{kj}^{j_1 \dots j_{k-1}} = 0$ 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

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

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

Анализ ресурса распараллеливания некоторых алгоритмов

- **Решение системы линейных уравнений методом Гаусса-Жордана**

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

- **Решение системы линейных уравнений методом Гаусса-Жордана**

  В зависимости от значений элементов матрицы $\mathcal{A}$ возможны $n!$ способов выбора $(j_1, \ldots, j_n)$, т.е. $n!$ перестановок элементов $(1, \ldots, n)$. Занумеруем все перестановки от 1 до $n!$.

  Введем обозначения

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

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

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

  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

- **Решение системы линейных уравнений методом 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 из Q-термов $u_i(i = 1, \ldots, n!; i \neq i_0)$ не превосходит уровня вложенности Q-терма u_{i_0}.
  Можно доказать, что $T^{u_0} = 3n - 1$.

Анализ ресурса распараллеливания некоторых алгоритмов

- **Решение системы линейных уравнений методом Гаусса-Жордана**

  Ранее было доказано, что

  $$T^{w_i^j} \leqslant 3n (i = 1, \ldots, n!, j = 1, \ldots, n),$$

  причем для некоторых Q-термов w_i^j имеет место равенство.

  Тогда для описанного алгоритма $\mathcal{A}$ метода Гаусса-Жордана

  $$D_{\mathcal{A}} = 3n.$$

  Если ведущие элементы выбирать по условию

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

  то также $D_{\mathcal{A}} = 3n$.

Анализ ресурса распараллеливания некоторых алгоритмов

- **Решение системы линейных уравнений методом Гаусса-Жордана**
  Оценим число процессоров, необходимых для реализации алгоритма Гаусса-Жордана.
  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.

Анализ ресурса распараллеливания некоторых алгоритмов

- **Решение системы линейных уравнений методом Гаусса-Жордана**

  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 ноль, 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 $2(n-1)n!$, т.е. $P_{\mathcal{A}} \geqslant 2(n-1)n!$.

Анализ ресурса распараллеливания некоторых алгоритмов

- **Решение системы сеточных уравнений**

  Уравнение Пуассона можно аппроксимировать системой сеточных уравнений

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

$$(1 \leqslant i \leqslant L, 1 \leqslant j \leqslant M),$$

  где u_{ij} — значения сеточной функции, $a_{ij}, b_{ij}, c_{ij}, d_{ij}, e_{ij}, f_{ij}$ — константы.

  Рассмотрим решение системы сеточных уравнений 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} — начальные приближенные значения сеточной функции, $u_{ij}^{n}(n \geqslant 1)$ — приближенные значения сеточной функции, полученные на n-м шаге итерации.

Анализ ресурса распараллеливания некоторых алгоритмов

- **Решение системы сеточных уравнений**

  Выразив значения сеточной функции $(n-1)$-го шага итерации через значения $(n-2)$-го шага, можно представить u_{ij}^n через

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

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

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

  Условием окончания итерационного процесса является $|u_{ij}^n - u_{ij}^{n-1}| < \varepsilon (1 \leqslant i \leqslant L, 1 \leqslant j \leqslant M)$, т.е. значение

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

  равно единице.

  v^n также выражается через $u_{k\ell}^0 (1 \leqslant k \leqslant L, 1 \leqslant \ell \leqslant M)$.

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

Анализ ресурса распараллеливания некоторых алгоритмов

- **Решение системы сеточных уравнений**

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

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

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

  Q-детерминант удовлетворяет условию выполнимости, поэтому максимально быстрая реализация алгоритма выполнима.

  На каждом такте работы ВС занято процессоров не более

  $$LM \left\lceil 5 + \frac{1}{8}(1 + \frac{1}{32} + \dots + \frac{1}{32^k}) \right\rceil,$$

  где

  $$k = \left\lceil \log_{32} \frac{LM}{8} \right\rceil = \left\lceil \frac{\log(LM) - 3}{5} \right\rceil,$$

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

Анализ ресурса распараллеливания некоторых алгоритмов

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

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

Заключение

- Q-детерминант алгоритма содержит все его реализации, в том числе максимально быструю, он делает алгоритм прозрачным с точки зрения его структуры и его реализации.
- Вместе с тем очевидно, что структура максимально быстрой реализации алгоритма может плохо согласовываться с архитектурой реальной ВС, модель которой далека от идеальной.
- По этой причине время выполнения на реальной ВС максимально быстрой реализации алгоритма может оказаться больше, чем какой-либо другой его реализации, причем для разных ВС это могут быть разные реализации.
- Таким образом, если ставить задачу выполнения алгоритма на ВС за минимально возможное время, нужно найти компромисс между тем, что требует алгоритм, и тем, что может предоставить ВС для его реализации.

Заключение

- В связи с этим для того, чтобы на конкретной ВС любой алгоритм можно было выполнить за минимально возможное время, нужно из всех реализаций алгоритма найти ту, которая лучше всего согласуется с архитектурой конкретной ВС, т.е. решить задачу оптимального наложения структуры алгоритма на архитектуру конкретной ВС.
- Придет время и появится архитектура ВС, согласующаяся со структурой максимально быстрых реализаций алгоритмов.
- По моему мнению разработка компанией IBM совместно с группой разработчиков из Университета Корнелла нейроморфического процессора TrueNorth является движением в этом направлении.