МЕТОД УЛУЧШЕНИЯ МЕТРИКИ ЦИКЛОМАТИЧЕСКОЙ СЛОЖНОСТИ ДЛЯ ОЦЕНКИ ВЫСОКОТОЧНЫХ РАСПРЕДЕЛЕННЫХ ВЫЧИСЛЕНИЙ

METHOD FOR IMPROVEMENT METRICS FOR EVALUATION CYCLOMATIC COMPLEXITY OF HIGH-PRECISION DISTRIBUTED COMPUTING

A. Babichev

Annotation

The article describes the methodological aspects of the complexity evaluation of large–block cloud computing with enhanced precision. We propose a method for improving the complexity evaluation metrics, which use control flow graph of the program, for designing of optimal performance computing systems that operate in the cloud. We derive formulas for evaluation of modified cyclomatic complexity and execution time of programs operating in sequential and parallel modes.

Keywords: complexity evaluation metrics, parallel computing; cloud computing; precise computations; precision–trust arithmetics.

Бабичев Антон Михайлович
Аспирант, ФГБОУ ВО "Тамбовский государственный технический университет", Тамбов

Аннотация

В статье рассмотрены методологические аспекты оценки сложности крупноблочных облачных вычислений с повышенной точностью. Предлагается метод улучшения метрик оценки сложности, использующих для этого граф потока управления программ, для построения оптимальных по производительности вычислительных систем, функционирующих в облачной среде. Выводятся формулы для оценки модифицированной цикломатической сложности и времени выполнения программ работающих в последовательном и параллельном режимах.

Ключевые слова:

Метрики оценки сложности, параллельные вычисления, облачные вычисления, высокоточные вычисления.

В настоящее время индустрия программного обеспечения (ПО) постоянно развивается. Появляется множество новых инструментов для написания ПО, существует большое разнообразие программных платформ, что непосредственно влияет на сложность и многообразие программ. Создается большое количество программ, в том числе и для научных целей, используемых для вычислений с повышенной точностью [1].

Существует множество областей науки и производства, требующих очень высокоточных вычислений. Малейшие неточности в таких отраслях могут привести к крайне негативным последствиям и, зачастую, экономическим потерям и даже вреду человеческому здоровью. Поэтому в таких вычислениях крайне важно следить за точностью получаемых результатов.

При использовании компьютерных расчетов распространено такое явление, как накопление ошибки вычислений. Как правило, оно происходит при циклическом вычислении каких–либо значений, которые основываются на предыдущих. Если такое значение вычисляется с помощью числового типа, содержащего недостаточное количество знаков после запятой, то программа округляет последнее число, что приводит к небольшой, но погрешности. Но когда это значение передается далее по циклу, то погрешности при вычислении следующих значений начинают накладываться друг на друга, и ошибка начинает заметно увеличиваться. Это легко заметить, если сравнить эти вычисления с их теоретическими идеальными значениями (**рис. 1**).

Для компенсации этого недостатка существуют различные способы но, в основном, они связаны с модернизацией способа округления при использовании чисел с небольшим количеством знаков после запятой, как, например, случайное округление или чередующееся округление.

Однако для высокоточных вычислений такие методы не подходят, так как округление остается неточным методом и в лучшем случае можно только немного уменьшить накопление ошибки. В таких ситуациях необходимо использовать типы чисел, которые позволяют производить операции с числами с очень большим числом знаков после запятой, что позволит либо вообще избавиться от ошибки округления, либо свести ее к минимуму.

Но расчет программы, использующей числа очень высокой точности, может занять несколько суток или даже недель и месяцев. И перед тем, как начинать эти расчеты, важно иметь оценить их ресурсоемкость и время–затраты. Такая оценка описывает целесообразность и возможность исполнения программы на одном компью–

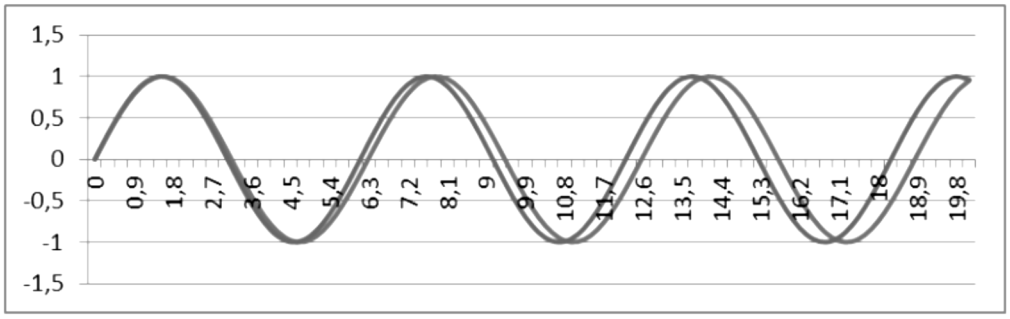

Рисунок 1. Разница между идеальным графиком и графиком с накопленной ошибкой вычисления.

тере, так как часто время получения результата не вписывается в требуемые сроки. В таком случае используется распараллеливание программы и расчет ее в распределенном режиме на нескольких машинах или в облаке. Такой режим работы программы также должен быть подвергнут оценке, потому что работа в распределенном режиме отличается от расчета на одной машине, работает по другому алгоритму и занимает другое количество времени.

Не всегда возможно использование традиционных методов распараллеливания, которые подходят для структурно простых задач. Для этого требуются новые методы облачного распараллеливания, для которых нужны новые критерии оценки вычислительной сложности. Оценка вычислительной сложности должна давать оптимальный результат для выбора распараллеливания крупноблочных вычислений, чтобы оптимально использовать вычислительное оборудование и время для решения таких задач.

Для измерения характеристик и критериев качества используются метрики измерения сложности программ. Как правило, метрики сложности дают информацию рекомендательного характера и используются для изучения сложности разрабатываемого программного обеспечения (ПО), оценки трудозатрат и объема работ, необходимых для реализации проекта, и дают представление о возможностях улучшения программного кода или оптимизации усилий, затрачиваемых разработчиком на создание программы.

Метрики оценки сложности делятся на 3 основных группы:

1. Метрики размера программ, или количественные метрики;
2. Метрики сложности потока управления программы;
3. Метрики сложности потока данных программ.

При этом метрики из одной группы могут использоваться одновременно и в связке с метриками из других групп, для получения более точной оценки определенных характеристик или самого ПО в целом.

В данной работе мы используем метрики оценки сложности программы, использующих для этого граф потока управления программ (ГПУ), в частности, метрику цикломатической сложности, предложенную Томасом МакКейбом в 1976 году [2]. Метрика сложности, предложенная Маккейбом, называется цикломатической сложностью программы и может быть вычислена по формуле:

$$V(G)=E-N+2P\ , \qquad (1)$$

где E – число дуг в графе потока управления программы; N – число вершин; P – число компонент связности графа.

Однако, оценка вычислительной сложности с использованием этой метрики может быть неполной, так как она дает достаточно субъективную оценку программы, и ориентирована больше на оценку стоимости разработки и стилевых особенностей программиста, без учета сложности формул, топологии связей и прочих нюансов, которые в действительности оказывают сильное влияние на окончательную вычислительную сложность ПО.

Для оценки сложности и времязатрат необходимо уметь оценивать самый базовый элемент программы – формулы уравнений [3]. Для этого мы будем использовать двоичные деревья формул. Представим уравнения из системы с помощью бинарного дерева с использованием бинарных операций. Возьмем для примера линейное уравнение вида $ax+by+c=0$. Для оценки сложности берется левая часть уравнения и представляется в виде бинарного дерева (**рис. 2**).

Построение формулы в виде дерева дает нам возможность оценить сложность каждой отдельно взятой операции в формуле. Так как в уравнения математических моделей могут входить различные по точности переменные, операции с этими переменными будут занимать

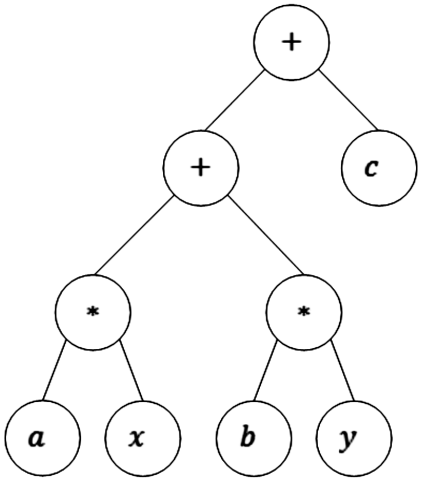

Рисунок 2. Бинарное дерево формулы $ax+by+c$.

различное время вычисления и, соответственно, у них будет различная сложность. Несмотря на то, что в итоге все приводится к единой точности, во время вычисления она может постоянно меняться, поэтому нам важно исследовать каждую из операций по отдельности, для чего мы анализируем построенное бинарное дерево [4].

Мы рассматриваем дерево снизу вверх, начиная с самых последних элементов, так как именно они соответствуют переменным формулы. С их помощью мы можем определить сложность операции, соответствующей узлу, из которого исходят эти переменные, основываясь на точности этих переменных.

Умение находить сложность формул дает нам возможность оценить общую сложность программы с очень высокой точностью, в сравнении с базовой метрикой цикломатической сложности. По сути, мы разбиваем программу на минимальные части, каждая из которых является строчкой исходного кода, содержащей различные команды или математические операции, и эти части объединяются в узлы графа потока управления. После этого мы можем оценить сложность каждого отдельного узла ГПУ на основе его составляющих.

Формула оценки сложности выражения исходного кода S_v в узле ГПУ выглядит следующим образом:

$$S_v = \sum_{n}^{i=1} S_i^o \qquad (2)$$

Где

S_i^o – сложность i –ой операции, n – общее кол–во операций в выражении.

Все оцениваемые выражения составляют узел ГПУ, и каждый такой узел можно разложить на множество деревьев формул.

Соответственно, сложность узла S_u определяется совокупностью оценок всех выражений, которые входят в него:

$$S_u = \sum_{m}^{i=1} S_i^v \qquad (3)$$

где

m – общее количество оцениваемых выражений в узле.

Общая сложность операций в ГПУ определяется как сумма сложностей каждого узла графа:

$$S_p = \sum_{N}^{i=1} S_i^u \qquad (4)$$

где

N– количество всех узлов ГПУ. Модифицированная цикломатическая сложность программы вычисляется на основе сложности ее ГПУ и общей сложности операций. Для этого находится десятичный логарифм от S_p, умноженный на цикломатическую сложность программы, найденную по формуле (1), – данная характеристика, обозначает порядок сложности программы:

$$V' = \log_{10}(Sp) \times V(G). \qquad (5)$$

С помощью данной характеристики можно даже без расчета времени определить, за какое ориентировочное время может выполниться программа, но такая оценка зависит от конкретного вычислительного оборудования, поэтому использовать ее представляется возможным только после серии тестовых запусков программ различной сложности на конкретном оборудовании.

Эта оценка времени является приблизительной и просто дополняет модифицированную оценку сложности (хотя иногда может быть достаточно и только ее), а время выполнения программы для получения точного представления можно рассчитывать с помощью полиномов, описывающих среднее время вычисления различных операций, а также производительности исследуемого вычислительного оборудования.

Нахождение сложности формул – первый, базовый этап определения сложности программы, осуществляющей вычисления с повышенной точностью. Следующий, промежуточный этап – нахождение модифицированной цикломатической сложности, на основе уже известной нам метрики. Но, как сказано выше, двух этих оценок может быть мало, когда речь идет об очень ресурсоемких и долгих вычислениях, для которых разумно использовать распределенные вычисления, что позволяет получить результаты гораздо быстрее.

В этих случаях мы предлагаем использовать сети Петри для описания процессов, происходящих в программах, работающих в параллельном режиме, и оценка поведения таких сетей в совокупности с двумя предыдущими завершает общую оценку программы [5]. Оценка

сложности параллельных вычислительных потоков должна учитывать характер их выполнения на ЭВМ: если эти потоки обрабатываются в параллельном режиме, оценка сложности должна быть согласована с критической линией, сложность которой максимальна. На **рис. 3** приводится иллюстрация к вышесказанному [6].

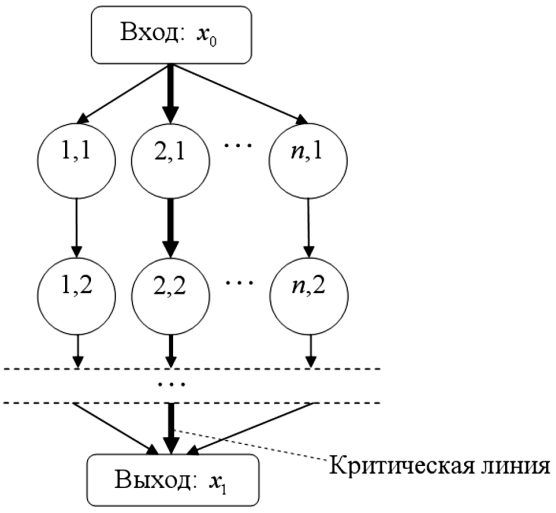

Рисунок 3. Параллельные вычислительные потоки.

Для определения модифицированной цикломатической сложности параллельной программы нам нужно найти сумму максимальных сложностей всех параллельных друг другу потоков:

$$V_p^{'} = \log_{10}(\sum_{n}^{i=1} \max(S_1^p, S_1^p, \ldots, S_j^p)) \times V_S(G), \quad (6)$$

где n – количество максимальных по сложности потоков в программе, j – количество потоков в одной группе параллельных вычислений, $V_S(G)$ – сквозная цикломатическая сложность, основанная на ГПУ, построенному по проходящим через сеть Петри критическим линиям.

В данной работе были выведены формулы для оценки модифицированной цикломатической сложности (5) и времени выполнения программ, работающих в последовательном режиме. С помощью использования сетей Петри был предложен метод распараллеливания и выведены формулы оценки модифицированной цикломатической сложности (6) и времени выполнения программ, работающих в параллельном режиме. Таким образом, предлагается методика улучшения метрик, использующих граф потока управления для более точного определения сложности программ.

Данная работа выполняется в рамках проекта #1346 из реестра государственных заданий высшим учебным заведениям и научным организациям в сфере научной деятельности.

ЛИТЕРАТУРА

1. Толстых С. С., Подольский В. Е., Оценка сложности крупноблочных облачных вычислений, использующих арифметику повышенной точности. Труды ИСП РАН, том 26, вып. 5, 2014 г., с. 29–64.
2. McCabe T.J., A complexity measure // IEEE Transactions on Software Engineering, vol. SE–2, no. 4, 1976. – с. 308–320.
3. Толстых С.С., Подольский В.Е., Бабичев А.М., Толстых С.Г. Вычислительная сложность решения систем линейных алгебраических уравнений: этап эксперимента, Вестник научных конференций. 2015. № 1–7 (1). с. 42–51.
4. Альфс Берзтисс. Глава 3. Теория графов. 3.6. Деревья // Структуры данных – М.: Статистика, 1974. – 131 с.
5. Федотов И. Е., Некоторые приёмы параллельного программирования. Учебное пособие. М.: Изд–во МГИРЭА(ТУ), 2008. – 188 с.
6. Толстых С.С., Подольский В.Е., Бабичев А.М., Толстых С.Г. Структурно–параметрическая минимизация орграфа облачной вычислительной системы, Вестник научных конференций. 2015. № 1–7 (1). с. 51–59.