Модель визуализации и отладки на графовом представлении Cloud-Sisal программ

Д.С. Гордеев¹

¹ Федеральное государственное бюджетное учреждение науки Институт систем информатики им. А.П. Ершова Сибирского отделения Российской академии наук, проспект Академика Лаврентьева, 6, г. Новосибирск, 630090, Российская Федерация

Аннотация

В работе описывается решение задач визуализации графов внутреннего представления Cloud Sisal программ, визуализации процесса вычислений и отладки Cloud Sisal программ. Приведены формальные определения графа с портами и графовой модели с портами и атрибутами. Описана модель визуализации графовой модели с портами и атрибутами с помощью статических изображений в формате векторной графики SVG. Описана модель отображения изменений графовой модели с портами и атрибутами с помощью анимаций, поддерживаемых форматом векторной графики SVG. Соединение графических анимаций, отображающие изменения в визуальных стилях и изменений в атрибутах графовой модели с портами реализовано с помощью безопасных сетей Петри. Описано моделирование вычислений, соответствующих функциям заданной Cloud Sisal программы с помощью иерархических сетей Петри, где переходы соответствуют функциям, а места аргументам и параметрам соответствующих функций. Также описаны модификации иерархических сетей Петри, обеспечивающие реализацию функциональности точек останова и редактирования аргументов или результатов функций при активированных точках останова в целях отладки с помощью добавления дополнительных мест и переходов. Таким образом для полученных сетей Петри рассматривается возможность изменения разметки мест в процессе функционирования.

Keywords¹

Cloud Sisal, Sisal, внутреннее представление, визуализация графовых моделей, динамический граф, сети Петри.

Visualization and debugging on internal representation graph of Cloud-Sisal programs

D.S. Gordeev¹

¹ A.P. Ershov Institute of Informatics Systems, Siberian Branch of the Russian Academy of Sciences, 6, Acad. Lavrentjev pr., Novosibirsk, 630090, Russia

Abstract

This paper describes the solution of the tasks of visualizing the graphs of the internal representation of Cloud Sisal programs, visualizing the process of computing and debugging Cloud Sisal programs. The formal definitions of the graph with ports and the graph model with ports and attributes are shown. A model of the visualization of the graph model with ports and attributes is described using static images in the SVG format. A model of displaying changes in the graph model with ports and attributes using animations

ГрафиКон 2021: 31-я Международная конференция по компьютерной графике и машинному зрению, 27-30 сентября 2021 г., Нижний Новгород, Россия
EMAIL: gds@iis.nsk.su (Д.С. Гордеев)
ORCID: 0000-0003-1623-9553 (Д.С. Гордеев)

supported by the SVG vector graphics format is described. The connection of graphic animations displaying changes in visual styles and changes in the attributes of the graph model with ports is implemented using Petri nets. It describes the modeling of calculations corresponding to the functions of a given Cloud Sisal program using the hierarchical Petri nets, where the transitions correspond to the functions, and the places of arguments and the parameters of the corresponding functions. Also described modifications of hierarchical Petri nets, ensuring the functionality of the breakpoints and editing the arguments or the results of functions at activated breakpoints for debugging purposes by adding additional places and transitions. Thus, for the obtained Petri nets, the possibility of changing markup of places in the process of functioning is considered.

Keywords

Cloud Sisal, Sisal, internal representation, graph model visualization, dynamic graph, Petri nets.

1. Введение

В рамках развития проекта облачной системы параллельного программирования CPPS [1], целью которой является предоставление возможности решения вычислительно-ёмких задач с помощью доступных по сети супервычислителей, интегрированных с системой CPPS, широкому кругу пользователей без прямого доступа к вычислителям большой мощности, но имеющих выход в Интернет, где пользователи создают и отлаживают программы на языке Cloud Sisal [2] с помощью вебприложения, а решение вычислительных задач осуществляется на поддерживаемом системой супервычислителя после предварительной адаптации с помощью настраиваемого компилятора из комплекта системы CPPS, возникает задача поддержки процесса программирования и отладки Cloud Sisal программы на локальной системе пользователя перед запуском на супервычислителе.

При трансляции программы на языке Cloud Sisal создаётся граф внутреннего представления IR входной программы, который представим в виде иерархического ориентированного графа [3]. При отладке программы полезно иметь визуальное изображение процесса исполнения для понимания движения данных по графу, а также об отображении функций на вычислительные узлы. При этом возникает задача отображения графа, и свойств вершин и дуг этого графа с помощью графических примитивов, таких как многоугольники, ломаные линии, цветовые стили, и визуального отображения изменений атрибутов вершин или дуг. Данная задача является нетривиальной, так как основным пользователем системы является человек, и ожидания пользователя должны совпадать с наблюдением за отображением изменений, чтобы возникло корректное представление о том, как происходило вычисление для заданной программе. Для удобства пользователя часто применяется плавное отображение изменений с помощью анимаций, например, с помощью SVG-анимаций в системе VisuAlgo [4].

Результатом визуализации динамический граф, и для его визуализации в основном применяются два семейства техник визуализации: автономная (offline) и интерактивная (online) [5]. В первом случае строится последовательность изображений графа, при визуализации которой между двумя соседними изображениями используются техники интерполяции для получения промежуточного изображения, и набор таких промежуточных изображений описывает трансформацию между двумя соседними автономными изображениями графа. Во втором случае используется динамическое вычисление очередного изображения по текущему состоянию графовой модели и следующему событию, которое следует графически отобразить. Для этого способа требуется выделение множества событий для отображения [6].

В данной работе обсуждается реализация модели визуализации динамических графов для решение задачи отображения процесса вычислений и отладки программ на языке Cloud Sisal с помощью сетей Петри [7] в рамках проекта развития облачной системы параллельного программирования CPPS.

2. Внутреннее представление

Текст Cloud Sisal программы транслируется внутреннее представление IR, которое описывается как иерархический ориентированный граф с портами. Формально такой граф описывается следующим образом. Пусть задан граф,

$$G = (V, P, F_{p,i}^v, F_{p,o}^v, E) \quad (1)$$

где V это непустое конечное множество вершин, P это непустое конечное упорядоченное множество портов, представимое в виде объединения двух непересекающихся множеств P_i и P_o . $F_{p,i}^v$ это функция $F_{p,i}^v: P_i \rightarrow V$, $F_{p,o}^v$ это функция $F_{p,o}^v: P_o \rightarrow V$, E это подмножество декартового произведения $(P_o \times P_i)$.

Множество P_i называется множеством входных портов, и с помощью функции $F_{p,i}^v$ позволяет говорить об упорядоченном множестве входных портов для каждой вершины из графа G . Аналогично множество P_o называется множеством выходных портов, и с помощью функции $F_{p,o}^v$ позволяет говорить об упорядоченном множестве выходных портов для каждой вершины из графа G .

В случае если функции $F_{p,i}^v$ и $F_{p,o}^v$ являются тождественными, то каждая вершина имеет ровно один входной порт и ровно один выходной порт, и граф G эквивалентен классическому определению графа $G' = (V', E')$, где E' есть подмножество декартового произведения $(V' \times V')$.

Для вершины v из V и ребра (p_o, p_i) из E будем говорить о инцидентности, если и только если $F_{p,i}^v(p_i) = v$ или $F_{p,o}^v(p_o) = v$. Для вершин v_1 и v_2 из V будем говорить о смежности вершин, если и только если существует ребро (p_o, p_i) из E такое, что $(F_{p,o}^v(p_o) = v_1$ и $F_{p,i}^v(p_i) = v_2)$ или $(F_{p,o}^v(p_o) = v_2$ и $F_{p,i}^v(p_i) = v_1)$.

Каждая вершина графа G внутреннего представления IR соответствует функции Cloud Sisal программы, по которой построен граф внутреннего представления. Входные порты P_i вершины v отвечают аргументам соответствующей функции, а выходные порты P_o вершины v отвечают результатам вычислений соответствующей функции. Дуга e в графе G , соединяющая выходной порт p_o^1 и p_i^2 , обозначает передачу данных из выходного порта p_o^1 во входной порт p_i^2 .

3. Графовая модель

При визуализации вершины графов обычно представляются геометрическими фигурами, такими как прямоугольники, окружности или более сложные многоугольники. Дуги часто представлены в виде ломаных или гладких кривых. Порты, в зависимости от применения изображения графа, могут отображаться с помощью кругов или более мелких квадратов, относительно вершины. Причём обычно порты располагаются на границах многоугольников, представляющих вершины. При записи решения задачи с помощью языка программирования часто функции вызывают другие функции. На графе внутреннего представления это отражается наличием вложенных графов, причём вложенные графы могут быть как связными, так и несвязными. В таких случаях принято считать, что вершина имеет атрибут, значением которого является вложенный граф, а саму вершину принято называть составной. Графы по отношению существования вложенного графа образуют иерархию графов внутреннего представления.

Также для определения стилей визуального отображения геометрических фигур требуется хранить информацию об атрибутах вершин и их значениях. Далее будем рассматривать графовую модель G' , расширяющую определение графа G (1) атрибутами. Пусть

$$G' = (V, P, F_{p,i}^v, F_{p,o}^v, E, A_v, A_p, A_e) \quad (2)$$

где $A_v: V \rightarrow \{(n_v, u_{n_v})\}$, где n_v это уникальное имя атрибута, а u_{n_v} это значение атрибута с именем n_v . Аналогично $A_p: P \rightarrow \{(n_p, u_{n_p})\}$ и $A_e: E \rightarrow \{(n_e, u_{n_e})\}$. Для вершин, например,

именем атрибута может быть “форма” со значениями “прямоугольник” или “окружность”, “цвет границы” со значениями цвета соответственно, “цвет фона” также со значениями цвета или “подграф” со значением графа, удовлетворяющего определению (2).

4. Визуальное представление

В каждом программном проекте существует подсистема вывода информации. В рамках данной работы подсистема вывода является графическим интерфейсом, основанным на технологии SVG², которая поддерживается большинством современных браузеров. Обычно для изображения элементов графов используются различные геометрические фигуры. Для описания двумерной векторной графики существует язык SVG, который также является расширением языка XML³. Язык SVG поддерживает такие фигуры как эллипсы, прямоугольники и более сложные фигуры, задаваемые с помощью последовательностей точек.

Рисунок 1: Пример IR-графа внутреннего представления программы, содержащей цикл.

Данный язык поддерживается основными браузерами, что делает его подходящим для представления графов при визуализации в контексте реализации облачного сервиса. Будем строить изображение графов по следующим правилам. Вершины будем представлять с помощью прямоугольников с границей. Порты будем представлять с помощью точек с координатами, заданными относительно содержащих их вершин. Порты вершин будем располагать только на верхних или нижних гранях прямоугольников, изображающих вершины. Координаты вершин также задаются относительно содержащих их вершин-графов. Дуги будем представлять с помощью ломаных линий или кривых Безье. Последние поддерживаются форматом SVG естественным образом. Кроме того, следует отметить, что при использовании SVG изображений в браузерах, изображения естественным образом включаются в DOM⁴ модель документа. Данное обстоятельство позволяет использовать стандартные методы для обеспечения интерактивного взаимодействия с построенным изображением. Например, это позволяет с помощью стандартных браузерных событий реализовать сворачивание и

² Scalable Vector Graphics. <https://www.w3.org/TR/SVG2/>

³ Extensible Markup Language. <https://www.w3.org/TR/xml/>

⁴ Document Object Model. <https://www.w3.org/DOM/DOMTR/>

разворачивание вершин графа, что облегчает навигацию по изображению. Также естественное включение в DOM позволяет использовать существующие средства для поиска вершин в графе по заданным значениям атрибутов, например, по имени функции [8]. На Рис. 1 представлен пример изображения внутреннего представления IR программы языка Cloud Sisal с помощью технологии SVG [9].

5. Визуализация изменений модели

При отладке программы полезно иметь визуальное отображение исполнения для представления движения данных по графу и того, на каких узлах вычислителя осуществляются вычисления тех или иных функций. При этом возникает задача отображения изменений значений атрибутов элементов графа, которые отображаются в изменение визуальных стилей графических примитивов, соответствующих вершинам, портам или дугам графовой модели. Методы визуализации процесса вычислений зависят от решаемой задачи и её предметной области. В данном случае это предметная область графов потока данных, где результатом визуализации в том или ином виде является последовательность изображений состояния графовой модели со всеми графически отображаемыми атрибутами. Рассмотрим последовательность изображений состояния графовой модели как последовательность событий, где каждое событие соответствует изменению одного атрибута для одного элемента графа. Так, например, событие заполнения входного порта значением, соответствует изменению значения атрибута «значение» с «пусто» на «существует», или с 0 на 1.

Рисунок 2: Сеть Петри, моделирующая процесс изменения значения атрибут элемента графа.

На Рис. 2 представлена сеть Петри, моделирующая изменение некоторого атрибута со значения X на значение Y . Место с пометкой X соответствует значению X некоторого атрибута, например, атрибута заполненности порта значением. Место с пометкой Y соответствует значению Y для того же самого атрибута того же порта. Место с пометкой $X|Y$ соответствует промежуточному состоянию, когда значение изменяется от X к Y или от Y к X . Опишем процесс функционирования данной сети, когда места d_i и b_i могут получать фишки в процессе функционирования сети. Будем считать, что место d_1 получает фишку, когда получена команда изменить значение атрибута с X на Y и при этом значение атрибута равно X . В этом случае срабатывает переход d_5 , место $X|Y$ получает фишку, обозначая промежуточное состояние, и место d_2 получает фишку, обозначая сигнал для запуска анимации на SVG изображении. Например, это может быть анимация изменения цвета порта со значения «чёрный» на значение «белый», которые обозначают отсутствие и наличие значения, соответственно. Когда анимация изменения завершается, место d_3 получает фишку, разрешая сработать переходу d_6 . Место Y получает фишку, обозначая завершение изменения значения атрибута порта. Место d_4 получает фишку, что предназначено для соединения с местом d_1 другой аналогичной сети, соответствующей изменению состояния инцидентной дуги. Аналогичным образом сеть может сработать в направлении от b_4 к b_1 для изменения значения порта с «заполненного» состояния

на «пустое», если потребуется произвести изменения значений в обратном направлении. Кроме того, если рассмотреть входной порт и инцидентную дугу и пары значений «пусто»-»заполнено» для порта, и «ожидание»-»передача значения», то место d_4 сети, относящейся к порту, может быть соединено с местом d_1 аналогичной сети, относящейся к дуге, и таким образом анимация заполнения порта значением будет синхронизирована с анимацией передачи значения по дуге. Продолжая аналогичные построения можно построить композицию сетей, представленных на Рис. 2, для всего графа программы Cloud Sisal, позволяя тем самым отображать распространение значений по дугам от порта к порту и от вершине к вершине в виде SVG-анимаций. Значения для мест X и Y определяются значениями из соответствующего трассировочного события. На Рис. 3. представлен результат работы композиции таких сетей.

Рисунок 3: Пример отображения локальных данных при активированной точке останова.

В сравнении с Рис. 1 можно видеть, как распространение значений по дугам отображается утолщением линий, изображающей дугу. Заполненные порты имеют более светлый оттенок при изображении, тогда как пустые порты отображаются тёмными точками или кругами. Для вершин с помощью дополнительных линий отображается промежуточное состояние между «ожидает начала вычислений» и «вычисления завершены». Например, на Рис. 3 для вершины “Mul” (умножение), содержащейся в вершине тела цикла («LoopBody»), видно, что утолщённые линии покрыли прямоугольник, изображающий вершину, что отражает завершенность вычислений на данной вершине. Кроме того, на Рис. 3 видны дополнительные линии разных цветов, изображающие распространение значений по графу программы. Этот приём визуализации показывает, что вычисления соответствующего элемента произведены в разных вычислительных потоках среды исполнения, где один и тот же цвет означает вычисления на одном и том же вычислительном потоке. Информация об идентификаторе вычислительного потока предоставляется в рамках трассировочного события.

6. Отладка

Рассмотрим некоторую функцию F , принимающую два аргумента, и возвращающую три результата. Например, это может быть функция одновременно вычисляющая минимум,

максимум и среднее значение от двух целых чисел. Пусть IR граф для этой функции состоит из одной вершины, имеющей два входных и три выходных порта. Для такого IR графа построим сеть Петри, состоящую из одного перехода, помеченного функцией F , двух входных мест и трёх выходных, причём будем считать, что входные и выходные места перехода упорядочены соответственно входным и выходным портам вершины графа IR. На Рис. 4 элементы данной сети выделены чёрным цветом. При срабатывании перехода, помеченного функцией F , выходные места заполняются результирующими значениями функции F для значений, заданных во входных местах. Если IR граф построен так, что существуют дуги, передающие значения из одной вершины в другую, то для каждой такой дуги в сеть Петри добавляем две дуги и переход, для передачи значения из места, соответствующего порту-источнику, в место, соответствующее порту-приёмнику. Таким образом достигается моделирование вычислений исходной Cloud Sisal программы через сети Петри. Для решения задачи визуализации распространения значений по графу программы во время вычислений модифицируем полученную сеть Петри следующим образом.

Для входных и выходных мест перехода добавим по одному переходу, при срабатывании которых места-источники продолжают содержать значения-аргументы. Также добавленные переходы снабдим входное место, гарантирующее срабатывание перехода ровно один раз. На Рис. 4 добавленные элементы помечены серым цветом.

Рисунок 4: Пример сети Петри для функции принимающей два аргумента и возвращающей три результата.

Данные переходы позволяют применить функции с побочными эффектами для сохранения трассировочных данных, при этом не нагружая основной переход. Если для моделирования функции F достаточно одного перехода, то возможно добавить функции для сохранения трассировочных данных. Однако, если функция F является составной и моделируется составным IR графом, и следовательно более сложной сетью Петри, то удобнее добавить отдельные переходы для сбора трассировочных данных в общем случае.

Таким образом для проведения отладки Cloud Sisal программ можно использовать трассировочные данные. Однако часто для отладки используется механизм точек останова, когда есть возможность приостановить вычисления, прочитать данных из локальных переменных, изменить значения, и продолжить вычисления с изменёнными значениями. Для достижения этого добавим к трассировочным переходам ингибиторную дугу и место. На Рис. 4 добавленные элементы выделены зелёным цветом. Если в процессе работы сети Петри в место точки останова добавить метку, и при этом трассировочных переход ещё не срабатывал, то процесс работы сети остановится до удаления метки из места точки останова. Это справедливо как для перехода входных мест, так и для перехода выходных мест. При проведении отладки это позволяет приостановить вычисления всей программы как перед началом вычисления функции F , так и после. При активированной точке останова вместо локальных переменных для чтения доступны значения из мест сети Петри, а для изменения доступны значения только для мест, где значения уже были заданы. На Рис. 5 представлен пример для точек останова, активированных для оператора умножения (справа) и для оператора сложения (слева).

Рисунок 5: Пример отображения локальных данных при активированной точке останова.

7. Благодарности

Исследование выполнено за счет гранта Российского научного фонда (проект 18-11-00118).

8. Заключение

Целью данной работы является описание решения задачи визуализации и отладки Cloud Sisal программ с помощью моделирования процесса вычислений через сети Петри. Предложенная модель описания Cloud Sisal программ позволяет производить вычисление по заданной программе, сбор трассировочных данных, осуществлять отладку с помощью точек останова, а также строить визуальное представление процесса вычислений, как процесса распространения значений по иерархической графовой модели программы. Распространение значений по вершинам, портам и дугам графа отражается на визуальном отображении анимацией графических стилей элементов графа. Вершины графа представлены геометрическими фигурами, а дуги ломаными или гладкими кривыми с помощью технологии SVG, а анимация графических стилей вершин, портов и дуг с помощью SVG-анимаций, что делает возможным и удобным визуализацию вычислений Cloud Sisal программ, отладку программ с точками останова, чтением и изменением значений в портах при включенных точках останова в рамках интернет среды.

9. Литература

- [1] В. Н. Касьянов, Е. В. Касьянова, Методы и система облачного параллельного программирования, материалы XIV Международной Азиатской школы-семинара «Проблемы оптимизации сложных систем», Алматы, 2018, сс. 298-307.

- [2] В. Н. Касьянов, Е. В. Касьянова, Язык программирования Cloud Sisal, препринт 181, ИСИ СО РАН, Новосибирск, 2018.
- [3] В. Н. Касьянов, В. А. Евстигнеев, Графы в программировании: обработка, визуализация и применение, БХВ-Петербург, Санкт-Петербург, 2003.
- [4] S. Halim, VisuAlgo-Visualising Data Structures and Algorithms Throug Animation, volume 9 of Olympiads in Informatics, 2015, pp. 243-245.
- [5] Д. С. Гордеев, Обзор техник визуализации алгоритмов на графах, Научная визуализация (2018), Том 10, 18-48, doi:10.26583/sv.10.1.02.
- [6] C. Demetrescu, I. Finocchi, J. T. Stasko, Specifying Algorithm Visualizations: Interesting Events or State Mapping?, in: Diehl S. (eds) Software Visualization, volume 2269 of Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, 2002, pp. 16-30, doi:10.1007/3-540-45875-1_2.
- [7] В. Е. Котов, Сети Петри, Наука, 1984.
- [8] M. Bostock, V. Ogievetsky, J. Heer, D3: Data-Driven Documents, IEEE Transactions on Visualization and Computer Graphics, 2011.
- [9] Д. С. Гордеев, Визуализация внутреннего представления программ на языке Cloud SISAL, Научная визуализация (2016), Том 8, 98–106.