Моя работа посвящена построению, анализу и оптимизации динамического потока данных в однородной распределенной среде. В этой работе были поставленны следующие цели:

- создание формальной модели потоков данных,
- анализ неопределенностей в схемах потоков данных,
- оценка ресурсов необходимых для запуска потока данных в распределенных средах.

Вначале рассмотрим понятия потоков данных. Поток данных является концепцией представления вычеслительных процессов. Среди характерных особенностей потоков данных можно выделить представление в виде наборов независимых вычислителей (акторов, блоков) несущих определенный алгоритм обработки и связей между ними. Связи представляют из себя каналы передачи данных. Также обычно модели потоков данных допускаю вложенные подпотоки и содержат некие стратегии запуска. Сама концепция потоков данных популярна в научных кругах и часто используется для задач с большими объемами данных и сложными распределенными системами благодаря понятности и простоте создания алгоритмов и изоляции от технических деталей распределенной системы и особенностей вычислений в ней. Рассмотрим способы представления потоков данных. Среди часто используемых можно выделить такие как:

- декларативные языки программирования (концепция которых очень близка к концепции потоков данных),
- графы (самый популярный способ представления из-за наглядной графической нотации),
- сети Петри (которые являются подклассом графов, особенно распространены для представления родственной концепции потоков работ),
- сети пороцессов Кана (используются для описания алгоритмов потоковой обработки).

В нашей модели используется представление в виде графа. Весь поток данных представляется в виде четверки:

- B — множество блоков,
- множество E задает связи или кналы передачи данных,
- множества I и O описывают собственные входные и выходные порты потока данных подобно блоку.

Блок в нашей модели также представляется в виде кортежа:

- наборы I и O входных и выходных портов блока,
- FA — конечный автомат Мили.

В реальных системах построения и запуска потоков данных каждый блок несет детерминированный алгоритм обработки данных, однако во время статического анализа данные недоступны, поэтому алгоритм блока в нашей модели описывается автоматом Мили, который показывает лишь качественное поведение блока — он содержит набор характерных состояний, ребра автомата показывают по какому набору портов блоку требуются данные и что блок выдаст в результате работы. Опять же из-за недоступности конкретных данных, автомат Мили несколько отличается от классического определения, в частности он может быть недетерминированным и часто не полным. Входной и выходной алфавиты соответственно множество подмножеств входных и выходных портов.

На слайде Вы можете видеть графическую нотацию блоков автомата Мили и всего потока данных. На первом слайде нарисован блок отвечающего за цикл и его автомат Мили. Поясним принцип автомата Мили этого блока. Состояние initial соответствует отсутствию внутреннего состояния алгоритма блока. Состояние non_trivial соответствует промежуточному состоянию — блок находится в процессе итерации. Переход из initial в initial отвечает за пустой список для преобразования. Переход из начального состояния в состояние итерации соответствует началу работы, то есть первой итерации. При этом блок поглащает сам список и выдает его первый элемент. Далее в состоянии итерации блок ожидает преобразованное значение и соответственно может выйти из цикла, излучив результат, либо продолжить итерацию в зависимости от длины остаточного списка.

На следующем слайде показан иерархический поток данных. Для удобства введены два фиктивных блока — source и stock, которые соответствуют входным и выходным портам потока данных. Внутри рамочки выделен подпоток данных.

Теперь о работе самого потка данных. В процессе работы потока данных, любой блок может находиться в одном из трех состояний:

- свободен — означает, что блоку не хватает данных для начала работы;
- работает,
- заблокирован.

В нашей модели для простоты все связи могут хранить не более одного набора данных, поэтому завершивший свою работу блок переходит в состояние заблокирован, если он не может излучить результат своей работы. Он продолжает находиться в этом состоянии ровно до тех пор, пока не может полностью излучить данные. На слайде можно видеть схему работы каждого блока. Ребра соответствуют соответствующим условиям переходов.

Алгоритм работы всего потока данных можно видет на слайде. В начале работы система управления опрашивает каждый блок, чтобы узнать необходимые наборы данных для запуска. Далее данные передаются необходимым блокам и осуществляется проверка на возможность перевода блока из состояния заблокирован. Так система управления разрешает состояния блокировки, далее ожидает окончания работы какого-либо блока, затем процедура повторяется. Поток данных завершается, если все блоки свободны и данные остались только на выходных портах потока данных.

Такая модель потоков данных является довльно гибкой и удобной для описания вычислений, однако она допускает неопределенности в своей работе. Некоторые неопределенности связаны с заменой алгоритмов блоков на автоматы Мили. Так в реальных запусках такие неопределенности отсутствуют, так как данные однозначно определяют поведение потока в этом случае. Такие неопределенности будем называть неопределенности первого рода.

Однако сама схема может так же допускать неопределенности, такие как неколько способов поглощения одного и того же набора данных, состояния гонок, также к неопределенностям отнесем возможность некорректного завершения потока данных. Такие неопределенности будем называть неопределенностями второго рода и будем считать их наличие ошибкой построения схемы, так как при их наличии в зависимости от обстоятельств запуска поведение и соответственно результат всего потока может различаться от запуска к запуску. Поэтому поток данных содержащий такие ошибки нуждается в исправлении, а его анализ лишен смысла.

Корректным потоком данных будем называть поток, схема которого не допускает неопределенностей второго рода.

Для начала введем понятие траектории запуска потока данных. В зависимости от времен затраченных блоками для работы и вариантов разрешения неопределенностей первого рода, поток данных будет вести себя различным образом. Траектория — последовательность состояний потока данных во все моменты модельного времени при определенных временах работы блоков и способе разрешения неопределенностей. Каждая траектория пораждает так называемый граф причинности или граф задач. Таким образом все возможные траектории разбиваются на классы эквивалентности. При корректном потоке данных, каждый класс определяется лишь способом разрешения неопределенностей и содержит траектории со всевозможными наборами времен работы блоков.

На этом факте основан предлагаемый волновой алгоритм. В работе представлены два варианта этого алгоритма: простой для случая корректных ациклических потоков данных и алгоритм для общего случая корректных потоков данных. На первом слайде приведена словестная формулировка простого алгоритма. Сам алгоритм представляет из себя моделирование работы потока данных, учитывающий все неопределенности. Алгоритм представлен в виде рекурсивной функции, на вход которой подается в том числе и частичный граф причинности, который алгоритм достраивает в зависимости от варианта разрешения неопределенностей. В результате, начальный запуск функции вернет множество всех возможным графов причинности.

Волновой алгоритм для общего случая отличается способом разрешения циклов — для того строится внутренний автомат потока данных. Алгоритм также представляется в виде рекусивной функции, которая проверяет, наличие в промежуточном внутреннем автомате вершины соответствующей обрабатываемому состоянию потоку данных.

Изначально рассматривалась возможность введения в автомате Мили распределений вероятностей переходов и времен их работы. Однако пример, часто использующийся на практике, изображенный на слайде показывает, что данный подход приводит к плохим результатам. Для этого достаточно рассмотреть произвольное распределение на переходах автомата Мили блока цикла. Формула для мат. ожидания длины цикла, которое определяет

время работы всего потока данных, приведена на слайде. Однако на практике типичный диапазон длин может охватыват несколько порядков, что говорит о бессмысленности рассмотрения подобных оценок.

Поэтому было принято решение отказаться от введения распределений вероятностей и следовать подходу, напоминающему минимаксный метод оценивания. Для анализа ресурсов требуемых потоком данных, требуется рассмотрение всех графов причинности, которые и описывают вариант поведения потока данных. Для простоты будем считать, что поток данных выполняется в гомогенной распределенной среде, а один блок для работы требует оин вычислитель. Заметим, что алгоритм использующийся для анализа позволяет отказаться от этих предположений. Для оценки ресурсов будем рассматривать так называемые срезы графа причинности и эволюции графа причинности — множество назависимых задач и их последовательные связанные цепочки. Каждой возможной эволюции графа причинности поставим в соответствие максимальную мощность среза. Эта величина будет определять максимальную загруженность. Соотвественно для графа причинности можно получить минимум и максимум этих оценок.

Данные оценки можно получить с помощью алгоритма HEFT.