Модели параллельных вычислений. Обзор

17 января 2017 г.

1 Введение

Первые модели вычислений были математическими и использовали понятие "глобального состояния"для определения "шага вычисления". Каждый шаг вычисления шел от одного глобального состояния вычислений до следующего.

Подход "глобальности состояний"был продолжен в теории автоматов для конечных автоматов и машин со стеком, в том числе их недетерминированных версий.

Первыми параллельными программами были обработчики прерываний. Если в процессе работы компьютеру необходимо принять информацию от внешних устройств, то выполнение текущей программы "прерывается"и запускается специальный код, называемый обработчиком прерывания, который помещает информацию в буфер, откуда она может быть впоследствии считана.

В начале 1960-х гг. прерывания стали использовать для имитации одновременного выполнения нескольких программ на одном процессоре. Наличие параллелизма с общей памятью привело к проблеме его управления. Для ее решения Эдсгер Дейкстра разработал семафоры, а позднее, в период между 1971 и 1973 г., Чарльз Хоар и Пер Хансен для решения этой проблемы разработали мониторы. Однако ни одно из этих решений не создавало в языках программирования конструкций, которые предоставляли бы доступ к совместным ресурсам. Инкапсуляцию сделали позже Хьюитт и Аткинсон, построив параллельно-последовательный преобразователь.

2 Лямбда-исчисление

Лямбда - исчисление Алонзо Черча можно рассматривать как самый первый язык программирования обмена сообщениями.

Семантика лямбда-исчислений выражается при помощи подстановок переменных, в которых значения параметров заменяются в теле вызываемых лямбда-выражений. Модель подстановок непригодна для параллелизма, поскольку она не обеспечивает возможность совместного использования ресурсов. Под влиянием лямбда-исчислений интерпретатор языка программирования LISP использует структуры данных, в которых значения параметров не подлежат замене в теле запускаемых лямбда-выражений. Это обеспечивает совместное использование эффектов обновления общих структур данных, но не обеспечивает параллелизм.

3 Сети Петри

До появления модели акторов сети Петри широко использовались для моделирования недетерминированных вычислений.

Сеть Петри представляет собой двудольный ориентированный граф, состоящий из вершин двух типов - позиций и переходов, соединенных между собой дугами. Вершины одного типа не могут быть соединены непосредственно. В позициях могут размещаться метки (маркеры), способные перемещаться по сети.

Событием называют срабатывание перехода, при котором метки из входных позиций этого перехода перемещаются в выходные позиции. События происходят мгновенно, либо разновременно, при выполнении некоторых условий.

Недостаток этой модели в том, что сети Петри моделируют управление потоком, а не сам поток данных. Кроме того, элементарный шаг вычислений в сети Петри представляет собой переход, при котором признаки одновременности исчезают на входах перехода и появляются на его выходах. Несмотря на эти недостатки, сети Петри остаются популярной моделью при моделировании параллелизма.

4 Модель акторов

Модель акторов сформировалась на базе предшествующих моделей вычислений в 1973 г. Модель акторов - это математическая модель параллельных вычислений, которая трактует понятие "актор"как универсальный примитив параллельного численного расчета: в ответ на сообщения, которые он получает, актор может принимать локальные решения, создавать новых акторов, посылать свои сообщения, а также устанавливать, как следует реагировать на последующие сообщения.

Актор является вычислительной сущностью, которая в ответ на полученное сообщение может одновременно:

- отправить конечное число сообщений другим акторам;
- создать конечное число новых акторов;
- выбрать тип поведения, которое будет использоваться для следующего сообщения в свой адрес.

Может существовать произвольная последовательность вышеописанных действий, и все они могут выполняться параллельно.

Развязка отправителя и посланных сообщений стала фундаментальным достижением модели акторов, обеспечившая асинхронную связь и управление структурами как прототип передачи сообщений.

Получатели сообщений идентифицируются по адресу, который иногда называют "почтовым". Таким образом, актор может взаимодействовать только с теми акторами, адреса которых он имеет. Он может извлечь адреса из полученных сообщений или знать их заранее.

Модель акторов характеризуется параллелизмом вычислений внутри одного актора и между ними, динамическим созданием акторов, включением адресов акторов в сообщения, а также взаимодействием только через прямой асинхронный обмен сообщениями без каких-либо ограничений на порядок прибытия сообщений.

Модель акторов может быть использована для моделирования параллельных систем, в которых есть распараллеливание вычислений:

– электронная почта (e-mail), в которой клиенты моделируются как акторы, а адреса электронной почты — как адреса акторов;

– Web-сервисы с конечными точками SOAP, которые моделируются как адреса акторов;

– объекты с семафорами (например, в Java и C), которые смоделированы как параллельно-последовательный преобразователь, при условии что их реализация такова, что сообщения могут приходить постоянно (возможно, они хранятся во внутренней очереди). Параллельно-последовательный преобразователь является важным видом актора, характеризующийся тем, что он постоянно доступен для прихода новых сообщений. Каждое сообщение, отправленное на параллельно-последовательный преобразователь, гарантированно будет получено;

– нотация тестирования и управления тестами (как TTCN-2, так и TTCN-3), где актором является тест компонента: либо параллельной тест компонента (PTC), либо главный тест компонента (MTC). Тесты компонентов могут отправлять и получать сообщения на/от удаленных партнеров (равноправные тесты компонентов или тест интерфейса системы), причем последний идентифицируется по его адресу. Каждый тест компонента имеет дерево поведения, связанное с ним. Тесты компонентов запускаются параллельно и могут быть динамически созданы родительскими тестами компонентов. Встроенные языковые конструкции позволяют определить действия, которые необходимо выполнить, когда сообщение получено из внутренней очереди сообщений, а также отправить сообщения другим равноправным субъектам или создать новые тесты компонентов.

В отличие от предшествующих ей моделей, основанных на комбинировании последовательных процессов, модель акторов была сразу разработана как параллельная модель, последовательность в которой представляет собой особый случай, вытекающий из одновременных вычислений.

Основным новшеством модели акторов было введение понятия поведения, определенное как математическая функция, выражающая действия актора, когда он обрабатывает сообщения, включая определение нового поведения на обработку следующего поступившего сообщения. Поведение обеспечивает функционирование математической модели параллелизма.

В настоящее время рост производительности микросхем происходит за счет использования методов локального и глобального массового параллелизма. Локальный параллелизм используется в новых микросхемах для 64-разрядных многоядерных микропроцессоров, в мультичиповых модулях и высокопроизводительных системах связи. Глобальный параллелизм в настоящее время используеися в новом оборудовании для проводной и беспроводной широкополосной пакетной коммутации сообщений (Wi-Fi и Ultra Wideband). Модель акторов применяется также в клиентах облачных вычислений.

5 Свойства модели акторов

Модель акторов обладает следующими свойствами:

– Неограниченный индетерминизм (называемый также неограниченным недетерминизмом) является свойством параллельных вычислений, при котором величина задержки исполнения запроса может стать неограниченной в результате использования запросами общих вычислительных ресурсов или ресурсов памяти, в то же время гарантируется, что запрос в конечном итоге будет обслужен. Неограниченный индетерминизм является характерной чертой модели акторов, в которой используется математическая модель Билла Клингера, основанная на теории доменов. В модели акторов не существует глобального состояния.

– Изменяемая топология. Естественным развитием модели акторов была возможность передачи адресов в сообщениях, что привело к тому, что модель акторов может имеет переменную топологию матрицы связей.

– В модели акторов отсутствует требование о том, что сообщения должны прибывать в том порядке, в котором они отправлены. Если необходимо упорядочить входящие сообщения, то это можно сделать явно при помощи очереди FIFO, которая обеспечивает такую функциональность. Отсутствие гарантий порядка доставки сообщений позволяет системе коммутации пакетов буферизовать пакеты, использовать несколько путей их отправки, повторно пересылать поврежденные пакеты и использовать другие методы оптимизации.

– Локальность модели акторов означает, что при обработке сообщения актор может отправлять его только по тем адресам, которые он получил из этого сообщения, по адресам, которые он уже имел до получения, и по адресам, которые он создал при его обработке, а также то, что не может одновременно произойти несколько изменений адресов. В этом отношении модель акторов отличается от некоторых других моделей параллелизма, например от сетей Петри, в которых реализации одновременно могут быть удалены из нескольких позиций и размещены по другим адресам.

– Миграцией в модели акторов называется способность актора изменить свое местоположение.

– Безопасность акторов может быть обеспечена одним из следующих способов:

  – с использованием аппаратного управления, к которому акторы подключены физически;

  – через специальное оборудование, как, например, в Лисп-машине и т.д.;

  – через виртуальную машину, как, например, виртуальную машину Java, общеязыковую исполняющую среду и т.д.;

  – через ОС, как, например, в системах с параметрической защитой;

  – с использованием электронной цифровой подписи и/или шифрования для акторов и их адресов.

– Синтез адресов акторов. Существует возможность синтеза адреса актора. В некоторых случаях СБ может запрещать синтез адресов. Поскольку адрес актора - это просто битовая строка, то, очевидно, его можно синтезировать, хотя, если битовая строка достаточно длинная, подобрать адрес актора довольно трудно или даже невозможно. SOAP в качестве адреса конечной

точки использует URL, по которому актор расположен. Так как URL является строкой символов, то, очевидно, ее можно синтезировать, хотя если применить шифрование, то подобрать строку практически невозможно.

6 Нейронные сети

Искусственные нейронные сети (ИНС) - это математические модели, построенные по принципу организации и функционирования биологических нейронных сетей нервных клеток живого организма, представляющих собой систему соединенных и взаимодействующих между собой объединенных в достаточно большую сеть с управляемым взаимодействием простых процессоров (искусственных нейронов), которые могут получать и принимать сигналы.

Вычислительные системы, основанные на искусственных нейронных сетях, обладают рядом качеств, которые отсутствуют в машинах с архитектурой фон Неймана (но присущи мозгу человека):

- массовый параллелизм;
- распределенное представление информации и вычисления;
- способность к обучению и обобщению, а именно:
  - адаптивность;
  - контекстно-зависимая обработка информации, устойчивость к ошибкам.

ИНС позволяют моделировать искусственный интеллект с помощью компьютерных алгоритмов. Нейронные сети используются в задачах адаптивного управления для робототехники, а также при распределении нагрузки между параллельно работающими устройствами. Отличие этих методов от всех вышеописанных — в алгоритмах обучения. Нейронные сети не программируются в привычном смысле этого слова, они обучаются. Обучение заключается в нахождении коэффициентов связей между нейронами, то есть в обобщении зависимостей между входными и выходными данными. Это значит, что в случае успешного обучения сеть сможет выдать верный результат на основании данных, которые отсутствовали в обучающей выборке, а также неполных и/или "зашумленных частично искаженных данных. Тип сети для конкретной постановки задачи выбирают с учетом имеющихся для обучения данных. Для обучения с учителем требуется наличие для каждого элемента выборки "экспертной"оценки. Иногда получение такой оценки для большого массива данных просто невозможно. В этих случаях естественным выбором является сеть, обучающаяся без учителя, например самоорганизующаяся карта Кохонена или нейронная сеть Хопфилда. При решении других задач, таких как прогнозирование временных рядов, экспертная оценка уже содержится в исходных данных и может быть выделена при их обработке. В этом случае можно использовать многослойный перцептрон или сеть Ворда.

С математической точки зрения обучение нейронных сетей - это многопараметрическая задача нелинейной оптимизации.

После выбора общей структуры ИНС нужно экспериментально подобрать параметры сети. Для сетей, подобных перцептрону, это будет число слоев, число блоков в скрытых слоях (для сетей Ворда), наличие или отсутствие обходных соединений, передаточные функции нейронов. При выборе количества слоев и

нейронов в них следует исходить из того, что способности сети к обобщению тем выше, чем больше суммарное число связей между нейронами. С другой стороны, число связей ограничено сверху количеством записей в обучающих данных.

После выбора топологии сети необходимо подобрать параметры обучения нейронной сети. Этот этап особенно важен для сетей, обучающихся с учителем. От правильного выбора параметров ИНС зависит скорость сходимости процесса обучения к правильным ответам.

В процессе обучения сеть в определенном порядке просматривает обучающую выборку. Порядок просмотра может быть последовательным, случайным и т.д. Некоторые сети, обучающиеся без учителя, например сети Хопфилда, просматривают выборку только один раз. Другие, например сети Кохонена, а также сети, обучающиеся с учителем, просматривают выборку множество раз, при этом один полный проход по выборке называется эпохой обучения. При обучении с учителем набор исходных данных делят на две части - собственно обучающую выборку и тестовые данные; принцип разделения может быть произвольным. В процессе обучения могут проявиться различные проблемы, такие как паралич или попадание сети в локальный минимум поверхности ошибок.

Даже в случае успешного обучения сеть не всегда обучается именно тому, чего от нее хотели. Известен случай, когда сеть обучалась распознаванию изображений танков по фотографиям, однако позднее выяснилось, что все танки были сфотографированы на одном и том же фоне. В результате сеть "научилась"распознавать этот тип ландшафта, вместо того чтобы "научиться"распознавать танки. Таким образом, сеть "понимает"не то, что от нее требовалось, а то, что проще всего обобщить.

7 BOINC

BOINC (англ. Berkeley Open Infrastructure for Network Computing) — открытая программная платформа (университета Беркли для GRID вычислений) — некоммерческое межплатформенное ПО для организации распределённых вычислений. Используется для организации добровольных вычислений.

BOINC — программный комплекс для быстрой организации распределённых вычислений. Состоит из серверной и клиентской частей. Первоначально разрабатывался для крупнейшего проекта добровольных вычислений — SETI@home, но впоследствии разработчики из Калифорнийского университета в Беркли сделали платформу доступной для сторонних проектов. На сегодняшний день BOINC является универсальной платформой для проектов в области математики, молекулярной биологии, медицины, астрофизики и климатологии. BOINC даёт исследователям возможность задействовать огромные вычислительные мощности персональных компьютеров со всего мира.

BOINC разработан командой во главе с Дэвидом Андерсоном (David Pope Anderson), возглавляющим также SETI@home, из Space Sciences Laboratory Калифорнийского университета в Беркли. На 26 апреля 2013 года BOINC представляет собой распределённую сеть из более чем 1 200 000 активных компьютеров (хостов) со средней производительностью всей сети около 8,5 петафлопс.

Платформа работает на различных операционных системах, включая Microsoft Windows и варианты юниксоподобных GNU/Linux, CentOS/RHEL, FreeBSD, NetBSD, OpenBSD, Solaris, Mac OS X и Android. BOINC распространяется под

лицензией GNU Lesser General Public License, как свободное программное обеспечение с открытым исходным кодом.