Методы и средства параллельной обработки знаний

Кондратович А. А.
Кафедра интеллектуальных информационных технологий
Белорусский государственный университет информатики и радиоэлектроники
Минск, Республика Беларусь
e-mail: andrew.kondratovich@gmail.com

***Аннотация*—В работе рассматриваются модели обеспечения параллельных вычислений в интеллектуальных системах с практической и формальной точек зрения.**

Ключевые слова: параллельные вычисления; интеллектуальные системы; модель акторов; многоагентные системы.

I. Введение

Идея распараллеливания вычислений основана на том, что большинство задач может быть разделено на набор меньших задач, которые могут быть решены одновременно.

Описывать логику работы для параллельных систем сложнее, чем для последовательных, так как конкуренция за ресурсы представляет новый класс потенциальных ошибок в программном обеспечении, среди которых состояние гонки является самой распространённой. Взаимодействие и синхронизация представляют большой барьер для получения высокой производительности параллельных систем.

Несмотря на повсеместное распространение и эксклюзивную поддержку на многих платформах, модель параллелизма, основанная на потоках операционной системы, имеет ряд принципиальных недостатков.

Во-первых, доступен лишь самый низкий уровень абстракции для общения между параллельно выполняющимися задачами - разделяемое состояние. В такой модели дополнительно к работе со данными каждая задача должна прилагать усилия по координации этой работы с другими задачами. Главная проблема здесь в том, что программисту необходимо очень детально описать логику работы и координации, что в свою очередь создает семантический разрыв между человеческим и компьютерным представлениями программы. Этот разрыв снижает эффективность выражения мыслей, ухудшает читаемость кода и в итоге приводит к ошибкам.

Во-вторых, в операционных системах слишком велики издержки на запуск потока, что делает непрактичным или невозможным разбиение программы на большое количества параллельно исполняющихся элементов. Но значительное количество программ (веб-приложения, обработчики систем реального времени, интеллектуальные системы и т.п.), наоборот, нуждаются в одновременном запуске как можно большего числа исполнителей. Поэтому, и здесь мы наблюдаем семантический разрыв, который заставляет программиста вручную поддерживать отображение логических потоков на потоки физические.

II. Модель акторов

Можно заметить, что в силу своей низкоуровневости модель потоков не слишком хорошо подходит для программирования человеком - отсюда вытекают проблемы, с которыми обычно сталкиваются разработчики при написании параллельных программ. Один из выходов из этой ситуации - использование высокоуровневых моделей, которые позволяют человеческое понимание программы более просто выразить в программном коде.

Одной из таких моделей является модель акторов[1]. В 1973 году Карл Хьюитт с коллегами выпустили работу "A Universal Modular Actor Formalism for Artificial Intelligence". В этой работе описана математическая модель параллельных вычислений, которая трактует понятие «актор» как универсальный примитив параллельного численного расчёта: в ответ на сообщения, которые он получает, актор может принимать решения, создавать новых акторов, посылать свои сообщения, а также устанавливать, как следует реагировать на последующие сообщения. Она использовалась как основа для понимания исчисления процессов и как теоретическая база для ряда практических реализаций параллельных систем.

В этой архитектуре вычисления реализуются набором акторов, каждый из которых:

- Имеет идентификатор, по которому он может быть опознан и адресован другими акторами.
- Умеет общаться с другими акторами путем посылки и получения сообщений, причем для набора отправленных сообщений гарантируется только сам факт их доставки адресатам, но не порядок их получения.
- Реализует свое поведение в реакциях на поступающие сообщения.
- Имеет недоступное для внешнего мира состояние, которое может влиять на его поведение.
- В ответ на некоторое сообщение может выполнить произвольную комбинацию следующих действий: а) изменить свое состояние, б) изменить логику обработки последующих сообщений, в) послать одно или несколько асинхронных сообщений, г) создать одного или нескольких новых акторов, д) завершить свою работу.

Как можно видеть, модель акторов очень похожа на модель объектов в объектно-ориентированном программировании[2]: а) программа представляется в

виде набора однородных взаимодействущих сущностей, б) состояние каждой сущности инкапсулировано внутри и доступно только теми способами, которые разрешит владелец, в) в процессе выполнения программы поведение сущностей может меняться.

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

Этот подход обладает большим потенциалом для решения практических задач. С точки зрения прикладного программирования, он построен на следующих базовых принципах:

— Наличие легковесных процессов, не привязанных к потокам ОС.

— Возможность асинхронной передачи сообщений между процессами.

— Отсутствие разделяемых данных между процессами и, следовательно, невозможность гонок и взаимных блокировок.

Параллельные программы, разработанные с использованием акторов, удобны в написании и сопровождении, а также принципиально не содержат проблем с гонками и взаимными блокировками, которые характерны для традиционной методологии.

III. МНОГОАГЕНТНЫЕ СИСТЕМЫ

В классической теории искусственного интеллекта решение какой-либо задачи сводится к созданию некоторой одной интеллектуальной системы, называемой агентом, которая, имея в своем распоряжении все необходимые знания, способности и вычислительные ресурсы, способна решить некоторую глобальную проблему.

В теории многоагентных систем[3] за основу берется противоположный принцип. Считается, что один агент владеет всего лишь частичным представлением о глобальной проблеме, а значит, он может решить лишь некоторую часть общей задачи. В связи с этим для решения сложной задачи необходимо создать некоторое множество агентов и организовать между ними эффективное взаимодействие, что позволит построить единую многоагентную систему. В многоагентных системах весь спектр задач по определенным правилам распределяется между всеми агентами, каждый из которых считается членом организации или группы. Распределение заданий означает присвоение каждому агенту некоторой роли, сложность которой определяется исходя из возможностей агента.

Для организации процесса распределения задачи в многоагентных системах создается либо система распределенного решения проблемы либо децентрализованный искусственный интеллект. В первом варианте процесс декомпозиции глобальной задачи и обратный процесс композиции найденных решений происходит под управлением некоторого единого «центра». При этом многоагентная система проектируется строго сверху вниз, исходя из ролей определенных для агентов и результатов разбиения глобальной задачи на подзадачи. В случае использования децентрализованного искусственного интеллекта распределение заданий происходит в процессе взаимодействия агентов и носит больше спонтанный характер.

В многоагентной системе агенты имеют несколько важных характеристик:

— Автономность: агенты, хотя бы частично, независимы

— Ограниченность представления: ни у одного из агентов нет представления о всей системе, или система слишком сложна, чтобы знание о ней имело практическое применение для агента.

— Децентрализация: нет агентов, управляющих всей системой.

Технология многоагентных систем, хотя и насчитывает уже более чем десятилетнюю историю своего активного развития, находится в настоящее время еще в стадии становления. Ведутся активные исследования в области теоретических основ формализации основных понятий и компонент систем, в особенности в области формализации ментальных понятий. Основные достижения в этой части пока не очень ориентируются на аспекты практической реализации и пока далеки от практики. В частности, при формализации ментальных понятий полностью игнорируются все разработанные в искусственном интеллекте подходы для работы с плохо структурируемыми понятиями, не вполне определенными понятиями, методы, которые базируются на вероятность и нечеткость. Представляется, что это обширное, новое и чистое поле деятельности для соответствующих специалистов.

Технология мультиагентных систем не является просто объединением различных результатов в области искусственного интеллекта. Интеграция, которая приводит к парадигме многоагентных систем, привносит ряд принципиально новых свойств и возможностей в информационные технологии и по существу представляет собой качественно новый, более высокий уровень ее развития, тот уровень, который позволяет прогнозировать ее ведущее положение в ближайшие десятилетия. Специалистам в области искусственного интеллекта здесь принадлежит ведущая роль.

IV. ЗАКЛЮЧЕНИЕ

В работе рассмотрены подходы, используемые в разработке интеллектуальных систем, дано описание и обоснование выбора. Используя эти подходы, можно добиться гибкости и расширяемости информационных и интеллектуальных систем.

[1] Actor model [Электронный ресурс]. Минск, 2012. – Режим доступа: http://en.wikipedia.org/wiki/Actor_model. – Дата доступа: 29.09.2012

[2] Hewitt. C., Bishop P., Steiger R. – A Universal Modular Actor Formalism for Artificial Intelligence [Электронный ресурс]. Минск, 2012. – Режим доступа: http://ijcai.org/Past%20Proceedings/IJCAI-73/PDF/027B.pdf. – Дата доступа: 03.10.2012

[3] Multi-agent System [Электронный ресурс]. Минск, 2012. – Режим доступа: http://en.wikipedia.org/wiki/Multi-agent_system. – Дата доступа: 05.10.2012