ИМИТАЦИОННОЕ МОДЕЛИРОВАНИЕ: ПОДХОДЫ К ПАРАЛЛЕЛЬНОЙ РЕАЛИЗАЦИИ

Д.Л. Аверичева, А.Б. Шворин, Л.К. Эйсымонт

Введение

В данной работе рассматривается несколько классических методов параллельного имитационного моделирования и проводится их сравнительный анализ. Кроме того, предложен так называемый *асинхронный* подход к моделированию, который, по мнению авторов, во многих случаях позволяет значительно увеличить степень параллелизма. Речь идет прежде всего о задаче моделирования аппаратуры, однако обсуждаемые подходы, вероятно, имеют более широкое применение.

Особое внимание будет уделено эффективности моделирования на современных вычислительных системах.

Особенности моделирования аппаратуры

Имитационное моделирование как дисциплина охватывает широкий круг задач. В данной работе рассматриваемый спектр задач сужен до моделирования цифровой аппаратуры, в том числе: процессоров, сетевого оборудования, вычислительных систем в целом.

В рамках поставленной задачи можно выявить и конкретизировать некоторые особенности, что позволяет упростить разработку и повысить эффективность симулятора — инструмента, с помощью которого осуществляется моделирование. Эти особенности перечислены ниже.

- Моделируемая система представляется в виде ориентированного графа — как набор объектов и связей между ними. Для удобства всегда можно считать систему иерархичной. То есть объект, если требуется, может быть представлен в виде подграфа. И наоборот, можно выделить некоторый подграф и объявить его объектом — узлом объемлющего графа.
- Граф статичен (не изменяется во времени).
- Поведение объекта задается лишь его внутренними свойствами и сообщениями, которые могут приходить только по его связям (ребрам графа). То есть объект может описываться локальными правилами, похожими на описание конечного автомата.
- Сравнительно мало объектов (как правило, тысячи — сотни тысяч), в отличие от, например, механики сплошных сред, где частиц гораздо больше.
- На связях могут быть задержки, причем, как правило, фиксированные. Кроме того, во многих случаях считается, что на всех связях задержки ненулевые. По крайней мере не может существовать цикла, в котором на всех связях задержки нулевые.
- Иногда граф распадается на несколько слабосвязанных компонент (между которыми связей сравнительно мало и/или на них большие задержки). Характерный пример — моделирование сети. Эту особенность полезно учитывать при параллельном моделировании.
- Как правило, вводится глобальное дискретное время. В качестве альтернативы может использоваться событийная модель, тогда синхронизация будет причинно-следственной (в этом случае множество событий частично упорядочено, хотя нельзя каждому событию приписать время его возникновения).

На практике в качестве модели часто используется описание электронной схемы на специализированных языках типа VHDL или Verilog. Современные САПРы имеют встроенный симулятор, что позволяет проводить моделирование схемы без ее реализации на чипе или в ПЛИС. Кроме того, запуск схемы в ПЛИС тоже можно рассматривать как в некотором роде моделирование, особенно когда часть компонентов эмулируется программно.

Немаловажным аспектом моделирования является эффективность — сколько реального времени займет симуляция одного модельного такта. Для повышения эффективности применяют различные приемы.

- Замена модели. Вместо подробных, «настоящих», описаний схем используются более грубые представления, которые содержат меньше информации о структуре моделируемой сущности и позволяют, например, лишь проверить функциональность или приблизительно оценить скорость работы схемы.
- Распараллеливание процесса симуляции. Этот довольно очевидный прием требует, однако, некоторых усилий для того, чтобы добиться реального ускорения.

Основные понятия

Симулятор — программа или программно-аппаратное средство, предназначенное для исполнения

заданного описания модели.

Элементарный акт передачи информации в модели между объектами будем называть *событием*. Жизненный цикл события таков: оно генерируется одним объектом в момент модельного времени t, передается по каналу связи и потребляется другим объектом в момент времени t+L, где L — фиксированная величина задержки на канале связи между объектами. Можно считать, что событие генерируется на каждом такте; если же на данном такте полезной информации не передается, то соответствующее событие назовем *пустым*.

В процессе моделирования симулятору необходимо передавать события между объектами. Это делается посредством *сообщений* — передач данных в рамках системы, на которой симулятор реализован. Например, в случае реализации симулятора на MPI таковыми будут являться MPI-сообщения; если же реализация сделана на языке Charm++ [3], то роль сообщений выполняют вызовы entry-методов. В общем случае сообщение может нести несколько событий, а также передавать дополнительную служебную информацию.

Узел — часть моделирующей системы, в рамках которой вычисления происходят последовательно. Таковым может выступать (в зависимости от моделирующей системы): процесс, тред (нить) или другая вычислительная единица без явного параллелизма.

Общие принципы параллельного моделирования

Без ограничения общности можно считать, что каждый объект отображен на отдельный моделирующий узел. Это всегда можно сделать, руководствуясь принципом иерархичности моделируемой системы. Не будем пока рассматривать динамическую балансировку объектов между моделирующими узлами, а, напротив, будем считать, что отображение объектов на узлы статично. Также оставим в стороне вопрос оптимального распределения объектов между узлами.

Таким образом, имеем на каждом узле один объект, у которого зафиксирован набор соседей, связи с которыми имеют заданные задержки. Основная идея асинхронного метода состоит в том, чтобы использовать эти модельные задержки для оптимизации параллельного моделирования.

Если требуется лишь проверка функциональности модели, то достаточно построить причинно-следственные связи между событиями. Однако, если в число целей входит количественная оценка эффективности моделируемой системы, то требуется ввести глобальное дискретное модельное время, выраженное в тактах. Будем рассматривать именно этот вариант.

Поскольку состояние объекта вычисляется последовательно, можно считать что в каждый момент реального времени объекту приписано состояние, соответствующее некоторому моменту модельного времени t. Вопрос о том, можно ли продолжить вычисления состояния в следующий момент времени t+1 и далее, зависит лишь от того, получил ли данный объект достаточно информации о событиях своих соседей.

Более формально это выражается в виде правила, которое можно назвать *принципом консистентности* модели.

Пусть t_i — (наибольший) момент модельного времени, такой что по i-му входу известны **все** события на интервале $[0, t_i]$. Тогда разрешается «продвинуть» состояние объекта до момента t’, если и только если $t' \leq \min t_i$.

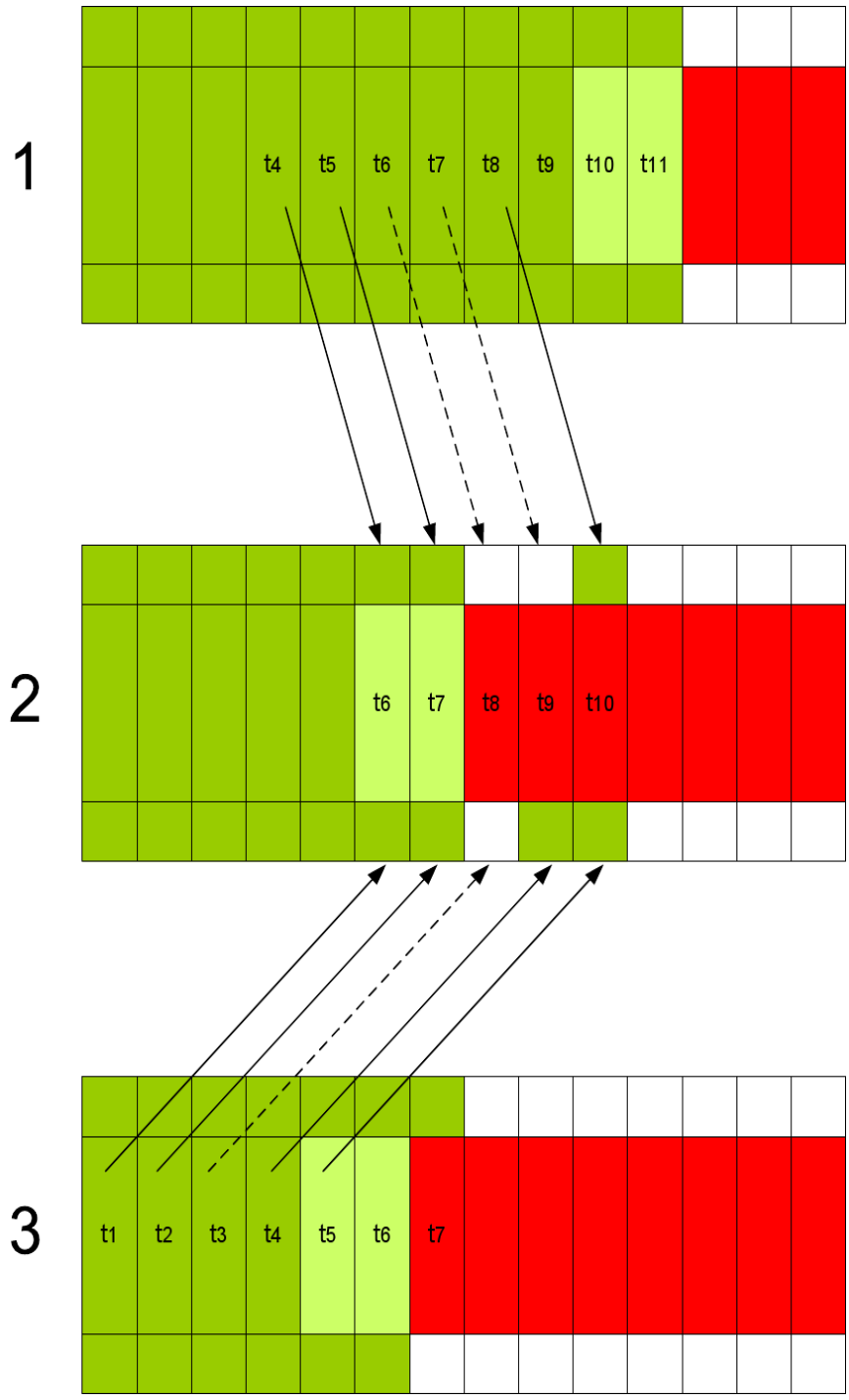

Рис. 1

На иллюстрации схематично представлена идея принципа консистентности. Здесь изображены состояния трех объектов, развернутые в модельном времени по горизонтали. Зеленым обозначены вычисленные состояния, красным — еще не вычисленные, а светло-зеленым — те, что, согласно принципу консистентности, можно вычислять. Стрелки показывают передачу событий между объектами, причем моделирующая система не обязана доставлять события в порядке отправления.

Основные подходы к параллельному моделированию

По способу синхронизации различные методы симуляции можно разделить на несколько видов.

Потактовая барьерная синхронизация заключается в том, что каждый объект вычисляет новое состояние для следующего такта, рассылает свои сообщения и дожидается получения сообщений от соседей, а потом ждет на барьере, пока остальные объекты сделают то же самое. Принцип консистентности сохраняется, поскольку задержки на всех каналах составляют не менее одного такта. Барьерная синхронизация необходима, чтобы все посланные сообщения достигли своих адресатов.

Симуляция с *пустыми сообщениями* является некоторой оптимизацией потактовой барьерной синхронизации. Каждый объект посылает, возможно пустые, сообщения всем без исключения соседям. Объект может приступить к вычислению следующего состояния, как только получит все сообщения (поскольку известно, сколько их должно быть). В результате посылаются «лишние» пустые сообщения, но зато нет глобального барьера.

Спекулятивная симуляция (с откатами) позволяет иногда нарушать принцип консистентности, однако умеет обнаруживать ситуации, когда такое поведение приводит к проблеме, и исправлять возникающую ошибку. Сообщения несут информацию только о нетривиальных событиях. Если сообщений нет, то объект предполагает, что и событий нет, и (спекулятивно) продолжает продвигать свое состояние. Если вдруг приходит запоздавшее сообщение, то приходится откатывать состояние на несколько тактов назад и, кроме того, рассылать соседям специальные сообщения, вынуждающие их сделать то же самое. Примером реализации спекулятивной симуляции может служить библиотека POSE, созданная на языке Charm++ в Иллинойсском университете [1].

Симуляция *с окном* [2] является обобщением потактовой барьерной синхронизации. Если известно, что на всех каналах между объектами задержка не меньше L, то можно выполнить $w \le L$ тактов без синхронизации, после чего требуется глобальный барьер. Здесь окном называют интервал, в течение которого нет синхронизаций, w — размер окна. Преимущество по сравнению с обычной потактовой синхронизацией состоит в том, что барьеры происходят в w раз реже.

Асинхронная симуляция позволяет наиболее гибко действовать в рамках принципа консистентности. Данный подход подробно рассмотрен ниже.

Асинхронный подход

Можно заметить, что все перечисленные выше подходы, за исключением спекулятивного, подчиняются принципу консистентности модели. Особенность асинхронного подхода в том, чтобы вводить как можно меньше ограничений сверх этого правила. (Можно сказать, что асинхронный метод в некотором смысле является обобщением всех перечисленных выше неспекулятивных методов.)

А именно, в процессе моделирования будем продвигать состояние объекта до тех пор, пока соблюдается правило консистентности. При этом не обязательно рассылать соседям события сразу при их возникновении; вместо этого можно аккумулировать несколько событий и посылать их одним сообщением. Важно, что сообщение может нести информацию о непроизошедших, то есть пустых событиях. Точнее, каждое сообщение описывает все события из целого, явно заданного, интервала в модельном времени. Неформально выражаясь, структура передаваемого сообщения выглядит так:

«Я, объект A, уведомляю объект B, что по каналу связи A→B в течение интервала модельного времени [1000, 1099] были переданы следующие события: e_1 на такте 1010, e_2 на такте 1025. В остальные моменты передавались только пустые сообщения.»

В результате, с учетом задержке на канале, объект B, получив данное сообщение, узнает сразу обо всех событиях, которые должны произойти в интервале [1000+L, 1099+L]. После чего объект B, вероятно, сможет продвинуть свое состояние дальше в модельном времени, не нарушая правило консистентности. Благодаря задержкам на каналах между объектами, один объект может достаточно далеко обогнать другой в модельном времени (точнее, не более чем на величину задержки канала), что позволяет эффективней скрывать коммуникационные задержки моделирующей системы.

Существуют две экстремальные стратегии отправки сообщений. Первая, «кумулятивная», заключается в том, чтобы отправлять сообщения как можно реже, стремясь накопить побольше информации о событиях. В предельном варианте — не отправлять сообщения до тех пор, пока возможно продвижение собственного состояния. И, как только это станет невозможным без нарушения правила консистентности, разослать всем соседям по одному (большому) сообщению.

Данная стратегия позволит снизить нагрузку на сеть моделирующей машины, но может привести к так называемому «голоданию», при котором соседние объекты будут вынуждены простаивать в ожидании информации о событиях.

Вторая стратегия, которую условно можно назвать «безотлагательной», выражается в том, чтобы как можно раньше отправлять сообщения, предотвращая простой соседних узлов. Как только будет закончено вычисление текущего состояния, станут известны все исходящие на данном такте события. Их можно сразу отсылать. Недостатком данной стратегии будет порождение большого количества мелких сообщений, что в результате даст бóльшую нагрузку на сеть, чем малое количество кумулятивных сообщений.

Ясно, что оптимальная стратегия лежит где-то между двумя этими экстремальными. И интенсивность, с которой следует посылать сообщения, зависит как от свойств модели, так и от моделирующей системы. Вероятно, в каждом конкретном случае следует экспериментально подбирать оптимальную интенсивность отправки сообщений.

Заключение

Существует целый класс задач имитационного моделирования, содержащий в себе, возможно,

недооцененный запас внутреннего параллелизма. Этот класс характеризуется относительно слабой связностью модели, то есть сравнительно большими задержками на передачу информации между компонентами модели. Предлагаемая методика — асинхронный подход — ставит своей целью использовать эту особенность модели, чтобы скрыть задержки моделирующей системе и, в конечном итоге, существенно повысить эффективность моделирования.

ЛИТЕРАТУРА:

1. Terry L. Wilmarth, Gengbin Zheng, Eric J. Bohm, Yogesh Mehta, Nilesh Choudhury, Praveen Jagadishprasad, Laxmikant V. Kalé. Performance Prediction using Simulation of Large-scale Interconnection Networks in POSE // University of Illinois at Urbana-Champaign, http://charm.cs.illinois.edu/newPapers/05-03/paper.ps
2. А.А. Корж, Д.В. Макагон, А.Б. Шворин Использование многоядерных процессоров на задачах дискретного моделирования // Материалы Всероссийской научной конференции "Научный сервис в сети ИНТЕРНЕТ", Новороссийск, 22-27 сентября 2008 г., Изд-во Московского Университета, с. 244-246
3. The Charm++ Programming Language Manual // University of Illinois at Urbana-Champaign, http://charm.cs.uiuc.edu/manuals/pdf/charm++.pdf