электронный журнал
МОЛОДЕЖНЫЙ НАУЧНО-ТЕХНИЧЕСКИЙ ВЕСТНИК
Издатель ФГБОУ ВПО "МГТУ им. Н.Э. Баумана". Эл No. ФС77-51038.

УДК 004.052.4

Подходы к верификации протоколов когерентности памяти

Буренков В.С., *аспирант,*
Россия, 105005, г. Москва, МГТУ им. Н.Э. Баумана
кафедры «Компьютерные системы и сети»

Научный руководитель: Иванов С.Р., к.т.н., доцент
Россия, 105005, г. Москва, МГТУ им. Н.Э. Баумана
v.suzev@bmstu.ru

1. Проблема когерентности и протоколы когерентности

Мультипроцессорная система с разделяемой памятью состоит из двух или более независимых процессоров, каждый из которых выполняет либо часть большой программы, либо независимую программу. Все процессоры обращаются к командам и данным, хранящимся в общей основной памяти. Для сокращения задержки на доступ к памяти каждый процессор снабжается локальной кэш-памятью.

Высокие уровни производительности устройств с мультипроцессорной архитектурой и разделяемой памятью обусловили строгую тенденцию к использованию именно этих архитектур в современных промышленно выпускаемых многоядерных процессорах.

В свою очередь, оснащение каждого процессора (ядра) локальной кэш-памятью приводит к проблеме обеспечения согласованности блоков кэш-памяти различных процессоров, или когерентности.

В решении проблемы когерентности выделяются два подхода: программный и аппаратный. Ввиду преимуществ в производительности, аппаратные механизмы, называемые *протоколами когерентности* кэш-памяти, нашли наиболее широкое применение.

Постоянно растущий уровень сложности мультипроцессорных систем с разделяемой памятью находит свое отражение и в замысловатости протоколов когерентности этих систем. Это, в свою очередь, определяет высокую трудность задачи проверки корректности протоколов, в то время как ошибки при их проектировании делаются часто.

2. Методы верификации протоколов когерентности

При верификации протоколов когерентности важными являются два аспекта:

- правильность разработки самих протоколов,
- корректность аппаратной *реализации* этих протоколов, то есть RTL-описания микропроцессора.

Существующие методы верификации протоколов когерентности памяти и их реализаций можно разделить на две группы:

- формальные методы;
- методы, основанные на моделировании.

2.1. Формальные методы

2.1.1. Анализ достижимости и model checking

В ходе формальной верификации протокола проверяется соответствие абстрактной модели протокола (*конечного автомата*) его спецификации, то есть набору свойств, которым должен отвечать протокол. Формальные методы нацелены на анализ всех достижимых состояний верифицируемой модели.

Одним из наиболее простых методов верификации протоколов когерентности является анализ достижимости состояний (*reachability analysis*), который полностью исследует пространство глобальных состояний, являющихся композицией состояний всех компонентов системы. Состояния, в которых ожидаемые свойства корректности протокола не удовлетворены, классифицируются как ошибочные; в противном случае состояния являются допустимыми. Если хотя бы одно из ошибочных состояний достижимо, протокол является некорректным [1, 2].

Анализ достижимости основан на алгоритме поиска методом полного перебора пространства глобальных состояний. Такая стратегия подвержена проблеме «взрыва числа состояний» и не применима для сложных систем [3].

Для верификации протоколов когерентности памяти может быть применен метод model checking [4], связанный с формальной проверкой выполнения на модели реализации свойств поведения, специфицированных на языке формальной логики. В качестве такой логики используется одна из темпоральных логик – логик, в которых значение логических формул зависит от момента времени, в котором вычисляются значения этих формул.

Модель верифицируемой системы, соответствующая методу model checking (структура Крипке), представляет из себя вариант конечного автомата. Данный метод может быть полностью автоматизирован, его использование предполагает удобный способ выражения требований к протоколам когерентности на языке формальной логики. Помимо этого, существует множество алгоритмов, позволяющих бороться с проблемой «взрыва числа состояний» при исследовании модели методом model checking.

http://sntbul.bmstu.ru/doc/603343.html

В [5, 6] сообщается, что существующие средства формальной верификации, основанные на алгоритмах model checking (Murphi, Spin), применяются для верификации промышленных протоколов когерентности памяти, и хорошо себя зарекомендовали. Однако при этом имеется ограничение на число узлов процессорной системы, отраженных в модели. В [5, 7] этот параметр ограничивается значением 4. Более того, современные протоколы когерентности настолько сложны, что даже верификация систем с тремя процессорами зачастую невозможна [8]. Таким образом, верификация протоколов когерентности при любых значениях числа процессоров (так называемая *параметризованная верификация*) только с помощью метода model checking невозможна. К необходимости методов верификации систем с большим числом процессоров (8, 16 и так далее) приводит постоянно растущая сложность микропроцессоров.

Существует два класса методов параметризованной верификации протоколов когерентности:

- методы, основанные на *model checking*, которые нацелены на максимальную автоматизацию;
- методы, основанные на *доказательстве теорем (theorem proving)*, которые нацелены на масштабируемость.

2.1.2. Методы, основанные на model checking

Один из подходов, основанных на model checking, заключается в том, что строится абстракция верифицируемой системы, сохраняющая интересующие нас свойства исходной модели, которая затем проверяется методом model checking. Построение абстрактных моделей требует нахождения компромисса между двумя конфликтующими целями:

- получение абстрактных моделей небольшого размера, которые могут быть верифицированы методом model checking.
- получение точных абстрактных моделей.

Чем меньше модель, тем больше поведений она допускает. Это может привести к появлению ложных контрпримеров, не обнаруживаемых в исходной модели. Выходов из этой ситуации, по крайней мере, два: построение точных абстрактных моделей, анализ контрпримера на ложность и модификация абстрактной модели на основании полученной информации (*counterexample-guided abstraction refinement*).

На практике зачастую недостаточно построения лишь одной абстрактной модели: для достижения результата необходимо получить несколько абстрактных моделей различной точности. Поскольку вопрос, удовлетворяет ли (возможно, бесконечная)

система некоторому темпоральному свойству, в общем случае неразрешим [9], нет гарантий завершения цикла детализации абстрактных моделей. По этой причине техника абстракции должна быть ориентирована на некоторый конкретный класс систем. Еще не предложено общего механизма абстракции для всех случаев.

Выделим следующие подходы, основанные на model checking.

Нахождение такого k, что верификация системы $P(k)$ достаточна для гарантии корректности параметризованной системы $P(N)$ для любого N. На основании работ в этом [10, 11] и смежном направлениях [12, 13] можно сделать вывод, что получаемые значения k являются слишком большими. Это делает метод неприменимым на практике. Так, в [11] было найдено $k = 7$ для протокола когерентности, основанного на справочнике.

Применение абстракции counter abstraction и ее обобщения environment abstraction ([14, 15, 16]) путем выделения набора идентичных процессов в классы эквивалентности на основе предикатов, которым они удовлетворяют, и принятия во внимание только одного представителя каждого класса. Абстрактные модели, полученные с помощью этих методов, являются очень детальными и, следовательно, будут слишком большими в случае сложных протоколов. Однако роль пользователя в этих методах относительно небольшая. Так, в [15] предлагается подход, при котором пользователь только описывает протокол на предлагаемом входном языке. Далее специальное средство на основе environment abstraction извлекает из этого описания соответствующую абстрактную модель, представляет ее на языке SMV и подает на вход инструментального средства SMV, проверяющего выполнение свойств на абстрактной модели.

2.1.3. Методы, основанные на доказательстве теорем

Методы, основанные на доказательстве теорем, требуют от пользователя постоянного вмешательства в ход доказательства, с целью продвижения последнего.

Парк [17] применил средство автоматизированного доказательства теорем общего назначения PVS [18] для параметризованной верификации протокола когерентности FLASH [19]. Данный процесс очень сложен, так как требует от пользователя индуктивных инвариантов.

В [5] предлагается метод, базирующийся на комбинации model checking и доказательства теорем, в частности, основанный на идеях из [20]. [21] формализует описание этого метода, а также доказывает его корректность.

Метод [5], получивший название CMP (по фамилиям авторов – Chou, Mannava, Park), принимает на вход описание протокола для n процессоров, оставляет описание двух или трех процессоров, а описание остальных заменяет процессором Other с высокой

http://sntbul.bmstu.ru/doc/603343.html

степенью недетерминизма. Впоследствии, путем нахождения ряда инвариантов, поведение Other ограничивается до тех пор, пока не станет возможным с помощью model checking проверить выполнение рассматриваемого свойства. Инварианты обычно представляют из себя выражения, сформулированные в терминах переменных модели протокола. Таким образом, метод CMP основан на композиционных рассуждениях и состоит из двух основных шагов: получения абстрактной модели (abstraction) и ее модификации (strengthening), которые итеративно применяются к описанию протокола.

Инварианты играют важную роль в различных подходах к верификации, но нахождение подходящих инвариантов – очень сложная проблема.

В [22] представлен подход на основе потоков сообщений (message flows), призванный сократить объем ручной работы при нахождении инвариантов. Потоки сообщений – последовательности сообщений, пересылаемые между процессорами во время работы протокола.

В [8] далее обобщается метод [22], расширяя понятие потоков так, чтобы они позволяли охватывать сложные сценарии, наблюдаемые в протоколах промышленного масштаба. Это позволило авторам провести параметризованную верификацию протокола LCP (Larrabee coherence protocol), разработанного в Intel и имеющего около 50 различных типов сообщений.

2.2. Методы, основанные на моделировании

Формальные методы обычно требуют описания протоколов (устройств, реализующих протоколы) на входных языках инструментов верификации. Поскольку протоколы когерентности реализуются множеством нетривиальных устройств, очень сложно непосредственно применить формальные методы к верификации этих реализаций.

Методы, основанные на моделировании, позволяют напрямую иметь дело с RTL-моделями микропроцессоров, а потому широко распространены. Они заключаются в генерации тестовых воздействий и их подаче на RTL-модель. При этом возникает вопрос, как достичь высокого тестового покрытия при минимальных избыточностях в тестах. Наиболее часто применяемые методы случайной генерации тестов не могут дать ответ на этот вопрос: случайная природа тестовых последовательностей не дает гарантии полного покрытия, тем более за приемлемое время. Несмотря на то, что данные методы позволяют найти большое количество ошибок, сложные ошибки, связанные с передачей множества сообщений между частями верифицируемой системы, к тому же и в некотором необычном порядке, могут оказаться нераскрытыми.

Тесты можно получить на основании результатов верификации формальной модели протокола. В [23] приводится пример такой комбинации формальных и

неформальных методов: на основании данных, получаемых при формальной верификации, строятся воздействия, являющиеся входными воздействиями для системы, моделирующей соответствующую Verilog-реализацию.

В [24] предлагается метод направленной генерации тестов, позволяющий при моделировании достичь полного покрытия состояний и переходов протокола. Сообщается, что при покрытии переходов устраняются избыточности (то есть лишние повторы одних и тех же переходов), присущие методу, основанному на поиске в ширину.

В [24] проиллюстрировано применение метода к широко известным протоколам SI, MSI, MESI, MOSI, MOESI и сделано заключение о том, что метод может быть эффективно приложен к сложным протоколам когерентности памяти, работающим в системах с большим количеством ядер (в качестве экспериментальных результатов приведены сведения верификации систем с числом ядер до 16), а также пригоден для верификации будущих поколений многоядерных архитектур. Однако в современных микропроцессорных системах применяются протоколы, в которых переходы между основными состояниями (такими, как M, O, S, I) не являются атомарными: существуют переходные состояния, описывающие кэш-блоки, находящиеся в процессе транзакции. Особенный интерес представляют одновременные обращения нескольких ядер к одному кэш-блоку. Это обстоятельство и его анализ в статье не отражены, поэтому неясно, можно ли использовать предлагаемый метод для верификации реализаций протоколов когерентности памяти, применяемых в промышленных микропроцессорах.

3. Заключение

В статье рассмотрены различные подходы к верификации протоколов когерентности. Наиболее перспективный из них – метод model checking – подвержен проблеме комбинаторного «взрыва числа состояний», а потому может быть применен лишь к системам с ограниченным числом процессорных ядер. При этом model checking очень удобен, так как позволяет автоматически осуществлять проверку выполнения свойств на модели.

Ключ к решению проблемы виден в комбинации метода model checking с другими формальными методами, в частности, основанными на доказательстве теорем, а также применении абстракций.

Абстракции позволяют получить модель меньшего размера, которая сохраняет интересующие нас свойства исходной модели и может быть проверена методом model checking. При этом существуют методы абстракции, применяемые к описанию системы

еще до того, как построена ее модель в виде графа. Это позволяет избежать построения модели, которая является слишком большой, чтобы поместиться в память.

Список литературы

1. Pong F., Dubois M. A New Approach for the Verification of Cache Coherence Protocols // IEEE Transactions on Parallel and Distributed Systems. 1995. Vol. 6, Issue 8, pp. 773-787.
2. Буренков В.С. Метод перебора состояний для верификации протоколов когерентности памяти. // 54-я научная конференция МФТИ, 2011.
3. Pong F., Dubois M. Verification Techniques for Cache Coherence Protocols // ACM Computing Surveys. 1997. Vol. 29, Issue 1, pp. 82-126.
4. Clarke E., Grumberg O., Peled D. Model Checking. MIT Press, 1999. 314 pp.
5. Chou C., Mannava P., Park S. A Simple Method for Parameterized Verification of Cache Coherence Protocols // Formal Methods in Computer-Aided Design. 2004. Vol. 3312, pp. 382-398.
6. Harrison J. Formal Methods at Intel – An Overview // Second NASA Formal Methods Symposium, 2010.
7. Буренков В.С. Инструмент верификации протокола когерентности памяти // Молодежный научно-технический вестник. 2013. № 1.
8. O'Leary J., Talupur M., Tuttle M. Protocol Verification Using Flows: An Industrial Experience // Formal Methods in Computer-Aided Design. 2009. pp. 172-179.
9. Apt K., Kozen D. Limits for Automatic Program Verification of Finite-State Concurrent Systems // Information Processing Letters. 1986. Vol. 22, Issue 6, pp. 307-309.
10. Emerson A., Kahlon V. Reducing Model Checking of the Many to the Few // Automated Deduction. 2000. Vol. 1831. pp. 236-254.
11. Emerson A., Kahlon V. Model Checking Large-Scale and Parameterized Resource Allocation Systems // Tools and Algorithms for the Construction and Analysis of Systems. 2002. Vol. 2280. pp. 251-265.
12. Emerson A., Namjoshi K. Reasoning about Rings. // Proceedings of the 22nd ACM SIGPLAN-SIGACT symposium on Principles of programming languages. 1995. pp. 85-94.
13. Clarke E., Talupur M., Touili T., Veith H. Verification by Network Decomposition // CONCUR. 2004. Vol. 3170. pp. 276-291.
14. Clarke E., Talupur M., Veith H. Environment Abstraction for Parameterized Verification // Verification, Model Checking, and Abstract Interpretation. 2006. Vol. 3855. pp. 126-141.
15. Talupur M. Abstraction Techniques for Parameterized Verification // PhD Thesis. 2006.

16. Clarke E., Talupur M., Veith H. Proving Prolemy Right: The Environment Abstraction Framework for Model Checking Concurrent Systems // Proceedings of the Theory and practice of software, 14th international conference on Tools and algorithms for the construction and analysis of systems. 2008. pp. 33-47.
17. Park S., Dill D. Verification of FLASH Cache Coherence Protocol by Aggregation of Distributed Transactions // Proceedings of the eighth annual ACM symposium on parallel algorithms and architectures. 1996. pp. 288-296.
18. Owre S., Rushby J., Shankar N. PVS: A prototype verification system // Automated Deduction. 1992. Vol. 607. pp. 748-752.
19. Kuskin J., Ofelt D., Heinrich M., Heinlein J., Simoni R., Gharachorloo K., Chapin J., Nakahira D., Baxter J., Horowitz M., Gupta A., Rosenblum M., Hennessy J. The Stanford FLASH multiprocessor // ISCA Proceedings of the 21st annual international symposium on Computer architecture. 1994. pp. 302-313.
20. McMillan K. Parameterized Verification of the FLASH Cache Coherence Protocol by Compositional Model Checking // Proceedings of the 11th IFIP WG 10.5 Advanced Research Working Conference on Correct Hardware Design and Verification Methods. 2001. pp. 179-195.
21. Krstic S. Parameterized System Verification with Guard Strengthening and Parameter Abstraction. 2005.
22. Talupur M., Tuttle M. Going with the Flow: Parameterized Verification Using Message Flows // Formal Methods in Computer-Aided Design. 2008. pp. 1-8.
23. Abts D., Scott S., Lilja D. So Many States, So Little Time: Verifying Memory Coherence in the Cray X1 // Proceedings of the 17th International Symposium on Parallel and Distributed Processing. 2003.
24. Qin X. Mishra P. Automated Generation of Directed Tests for Transition Coverage in Cache Coherence Protocols // Design, Automation & Test in Europe Conference & Exhibition. 2012. pp. 3-8.