

Генерация ограничений для простых тестовых шаблонов (LRU)

Евгений Корныхин
Московский Государственный Университет

I. ТЕСТОВЫЕ ШАБЛОНЫ

некие общие слова по поводу тестирования, тестовых шаблонов и применения ограничений. Тегсеты, регионы, rfn. Какую задачу решает эта статья, почему она не решалась, пока этой статьи не было.

II. ПРОСТЫЕ ТЕСТОВЫЕ ШАБЛОНЫ

Инструкции с известным регионом это инструкции, регион которых можно определить без решения задачи на ограничения. Например, такой инструкцией является инструкция с [l1Hit, tlbHit], если она является самой первой в тестовом шаблоне.

Простой тестовый шаблон (структурное определение) это тестовый шаблон, в котором от первой инструкции с неизвестным регионом следует $w + 1$ инструкций с неизвестным регионом (включая эту) (возможно, с включениями инструкций с известным регионом), а затем следуют только инструкции с кэш-попаданиями. w ассоциативность кэш-памяти. Тестовый шаблон, не удовлетворяющий этому определению, не является простым тестовым шаблоном.

Простой тестовый шаблон (функциональное определение) это тестовый шаблон, сложность процедуры поиска тестовых данных для которого не зависит от размеров кэш-памяти. Эта сложность есть $O(n^2)$, где n длина тестового шаблона. Процедура поиска тестовых данных для непростого тестового шаблона приводит к необходимости рассмотрения 2^R систем ограничений, где R битовый размер сета физического адреса. Иными словами, простые тестовые шаблоны это такие шаблоны, для которых построение тестовых данных проводится без большого перебора. На самом деле исследователи отмечают [1], что тестовых шаблонов размера порядка w достаточно для проведения тестирования.

Пример простого шаблона:

```
[l1Hit, tlbHit] ts1
[l1Miss, tlbHit] ts2
[l1Miss, tlbMiss] ts3
```

Длина простого тестового шаблона не ограничена сверху. Любой тестовый шаблон, длина которого не превосходит $w + 1$, является простым. Пример непростого тестового шаблона длины $w + 2$ (для $w = 3$):

```
[l1Miss, tlbHit] ts1
```

```
[l1Miss, tlbHit] ts2
[l1Miss, tlbHit] ts3
[l1Miss, tlbHit] ts4
[l1Miss, tlbHit] ts5
```

В случае $ts_5 = ts'_4 = ts_1$ для выполнения тестовых ситуаций должно быть выполнено свойство $\{ts'_1, ts'_2, ts'_3\} = L \cap R(ts_1)$, но $R(ts_1)$ вычислить на основе только лишь шаблона невозможно, поэтому для возможности вычислить нужную часть кэш-памяти приходится перебирать все возможные значения $R(ts_1)$ (т.е. по сути всё содержимое кэш-памяти).

III. УРОВНИ МОДИФИКАЦИИ СОСТОЯНИЯ КЭШ-ПАМЯТИ И TLB

В зависимости от сложности модификации TLB и кэш-памяти (в общем случае, эта модификация осуществляется совместно, т.е. одна инструкция может изменить некоторым образом вместе TLB и кэш-память) типы инициализирующих программ можно разделить по следующим уровням (имеется отношение вложения):

- *нулевой уровень инициализации* тестовый шаблон поддерживает нулевой уровень инициализации, если для него существует инициализирующая программа, которая не меняет TLB и кэш-память (можно менять значения регистров и ячеек оперативной памяти, если это необходимо);
- *первый уровень инициализации* тестовый шаблон поддерживает первый уровень инициализации, если для него существует инициализирующая программа, которая меняет JointTLB, но не меняет MicroTLB и кэш-память; такое изменение можно сделать последовательностью инструкций TLBWI;
- *второй уровень инициализации* тестовый шаблон поддерживает второй уровень инициализации, если для него существует инициализирующая программа, которая меняет JointTLB и MicroTLB, но не меняет кэш-память; в дополнение к инициализации первого уровня здесь придется обратиться в Mapped Uncached область виртуально памяти, чтобы заполнить и выстроить в нужном порядке содержимое MicroTLB (количество таких инструкций равно ассоциативности MicroTLB);
- *третий уровень инициализации* тестовый шаблон поддерживает третий уровень инициализации, если для него существует инициализирующая

программа, которая меняет TLB и кэш-память; это изменение можно осуществить последовательностью инструкций, работающих с памятью.

Сложность построения и разрешения ограничений для тестовых данных монотонно увеличивается с каждым следующим уровнем инициализации. Поэтому следует сначала проверять данный тестовый шаблон на наличие у него инициализации нулевого уровня, затем первого, и т.д.

IV. ПОСТРОЕНИЕ ОГРАНИЧЕНИЙ ДЛЯ НУЛЕВОГО УРОВНЯ ИНИЦИАЛИЗАЦИИ

A. Алгоритм построения ограничений

Для каждой инструкции $l1Hit(x)$ составляется ограничение $x \in L \cap [PFN] \cup \{x_1, \dots, x_n\}$, где x_1, \dots, x_n тэгсеты всех предыдущих кэш-промахов.

Для каждой инструкции $l1Miss(x)$ составляется дизъюнкция из следующих элементов:

- $x \notin L \cap [PFN] \wedge x \notin \{x_1, \dots, x_n\}$, где x_1, \dots, x_n тэгсеты всех предыдущих кэш-промахов;
- $x \notin \{x_1, \dots, x_n\} \wedge LRU(x, x_1, \dots, x_n)$, где x_1, \dots, x_n тэгсеты всех предыдущих кэш-промахов; если $L \cap [PFN] = \emptyset$, это ограничение не входит в дизъюнкцию.

Ограничение $LRU(x, x_1, \dots, x_n)$ тождественно ложно при $L \cap [PFN] = \emptyset$. В остальных случаях для каждого элемента $\lambda \in L \cap [PFN]$ составляется дизъюнкция, каждый элемент которой соответствует инструкции с кэш-промахом x_i ($i \in \{w - k + 2, \dots, n\}$, где k — индекс λ в сети, если первым стоит самый старый элемент сета; если $n \leq w - k + 1$, то дизъюнкция LRU пустая). Конъюнкт, соответствующий x_i , включает дизъюнкцию, составленную выбором без повторов $w - k + 1$ инструкций (блок) среди первых $i - 1$ инструкций. Каждому такому выбору соответствует конъюнкция следующих ограничений:

- $x = \lambda$;
- для каждого $l1Miss(y)$, не входящего в блок, ограничение $R(y) \neq R(\lambda)$;
- для каждого $l1Miss(y)$, входящего в блок, ограничение $R(y) = R(\lambda)$;
- для каждого $l1Hit(y)$, входящего в блок, ограничение $y \in \{\lambda_{k-1}, \dots, \lambda_1\} \setminus \{y_1, \dots, y_m\}$, где y_1, \dots, y_m предыдущие инструкции с $l1Hit$, входящие в блок;
- для каждого $l1Hit(y)$, расположенного до первой инструкции блока, ограничение $y \notin \{\lambda_k \dots \lambda_1\}$;
- для каждого $l1Hit(y)$, расположенного внутри блока, но не входящего в него, ограничение $y \notin \{\lambda_k, \dots, \lambda_1\} \setminus \{y_1, \dots, y_m\}$, где y_1, \dots, y_m предыдущие инструкции с $l1Hit$, входящие в блок;
- для каждого $l1Hit(y)$, расположенного после последней инструкции блока, ограничение не нужно.

B. Полнота и корректность алгоритма

Как представляется состояние кэш-памяти.

Как выражаются в операциях над множествами кэш-попадания и кэш-промахи. Двойственность.

Упрощение этих операций в простых шаблонах.

Алгоритм полный (т.е. не упускает никаких случаев) и корректный (т.е. не может построить систему ограничений, дающую результат, несоответствующий шаблону), т.к. все ограничения однозначно соответствуют определению тестовых ситуаций в простых тестовых шаблонах.

V. РАСШИРЕНИЕ ПРОСТЫХ ТЕСТОВЫХ ШАБЛОНОВ

подбор регионов последних кэш-промахов за счет перебора случаев равенства предыдущих вытесняемых тэгсетов, для которых известен регион.

СПИСОК ЛИТЕРАТУРЫ

- [1] статья Саши Камкина в сборнике ИСП