Санкт-Петербургский политехнический университет Петра Великого
Институт компьютерных наук и кибербезопасности
Высшая школа технологий искусственного интеллекта
направление
Математическое обеспечение и
администрирование информационных систем

Исследовательская работа №2

по дисциплине «Архитектура суперкомпьютеров»
по теме «Протоколы кеш когерентности и политики замещения: Xerox Dragon LRU»

Студенты группы 5130203/20002: В.А. Миронов
Д.В. Середа
И.В. Жданова
Д.А. Шульжик
Ч.Х. Лонг
И.Н. Дюро

Руководитель: _____ М. В. Чуватов

6 апрель 2024 г

Санкт-Петербург
2024

Содержание

1 Gragon (Xerox) protocol

1.1 Общая информация

Протокол Dragon — это протокол, основанный на обновлении, что означает, что при записи в кэш обновленные данные синхронизируются с другими ядрами, которым принадлежит эта строка кэша. Распространение записи выполняется путем прямого обновления всех кэшированных значений на нескольких процессорах. Протоколы, основанные на обновлении, такие как протокол Dragon, работают эффективно, когда за записью в блок кэша следует несколько операций чтения, выполняемых другими процессорами, поскольку обновленный блок кэша легко доступен во всех кэшах, связанных со всеми процессорами

Протокол Gragon (Xerox) имеет четыре состояния: D, SD, VE, SC.
Состояния:

- Invalid (I) - начальное состояние
- Dirty (D) - эксклюзивный, и данные не согласуются с памятью
- Vilid-Exclisive (VE) - эксклюзивные, и данные согласуются с памятью
- Shared-Dirty (SD) - данные существуют в нескольких кешах одновременно, и текущий процессор является последним, который модифицировал блок. Процессор несет ответственность за обновление основной памяти при удалении блока кэша.
- Shared-Clean (SC) - данные существуют в нескольких кэшах одновременно, и они не являются последними, записывающими данные в кэш.

Как видите, D и VE являются эксклюзивными, а если имеется несколько кэшей с одной и той же строкой кэша, то имеется несколько Sc и один Sm.

В определенный момент времени любые два кеша могут иметь следующие состояния для конкретного блока: Shared-Dirty (SD) и Shared-Clean (SC), Shared-Clean (SC) и Shared-Clean (SC). Протокол отличается тем, что:

- не имеет состояния I (Invalid) (за исключением начальной инициализации)
- является write-broadcasting прокотол (то есть не write-through потокол)
- использует общую линию (shared line)
- благодаря write-broadcasting протокол избегает состояния I

1.2 Процессорные операции

Рассмотрим четыре процессорных операции: попадание чтения (hit read), промах чтения (miss read), попадания записи (hit write), промах записи (miss write).

Попадание чтения (hit read) (Происходит, когда процессор завершает успешное чтение определенного блока кэша, помещенного в его кэше): Данные считываются из кеша.

Промах чтения (miss read) (Происходит, когда процессору не удается прочитать блок кэша из своего кэша и ему необходимо получить блок либо из памяти, либо из другого кэша.): При возникновении промаха чтения он проверяет на шине, есть ли уже в каком-либо кэше данные этой строки кэша, если нет, то считывает из памяти и переходит в состояние Valid-Exclusive (VE); если он уже находится в другом кеше, он считывает из другого кеша, кэш перемещается в состояние Shared-Clean (SC).

Попадание записи (hit write) (Происходит, когда процессор завершает успешную запись в определенный блок кэша, помещенный в его кэш. Благодаря этому процессор будет последним, обновившим блок кэша): Когда происходит попадание записи, если состояние Shared-Dirty (SD), то необходимо уведомить другие кэши о необходимости обновления данных (то есть другие кеши с состоянием Shared-Clean (SC)); если состояние Shared-Clean (SC), то необходимо уведомить другие кэши о необходимости обновления данных и позволить кэшу в состоянии Shared-Dirty перейти в состояние Shared-Clean (SC).

Промах записи (miss write) (Происходит, когда процессору не удается выполнить запись в блок кэша из своего кэша, и ему необходимо извлечь блок из памяти или другого кэша, а затем записать в него. Это снова делает процессор последним, обновившим блок кэша): Когда происходит промах записи, также проверяется состояние другого кэша, и если он является первым, к которому осуществляется доступ, он считывается из памяти, обновляется и перемещается в Dirty (D); если к нему обращаются не впервый раз, он переходит в состояние Shared-Dirty (SD), а другой кеш, который был в состоянии Shared-Ditry (SD), переходит в общее чистое состояние Shared-Clean (SC).

Здесь кеш в состоянии Shared-Dirty (SD) отвечает за запись данных в память при выгрузке.

1.3 Транзакции шины

Существует две шинные транзакции: Bus Read (в двух вариантах), Write Broadcasting.

Bus Read (Чтение с кеша). Если кеш в состоянии Dirty (D) или Shared-Dirty (SD) данные отправляются в запрашивающий кеш. Кеш остается (устанавливается) в состояние SD. В противном случае кеш остается

в состоянии Shared-Clean (SC).

Bus Read (Чтение в кеш). Если состояние Dirty (D) или Shared-Dirty (SD), то данные получаются от запрашиваемого кеша.Кеш устанавливается в Shared-Clean (SC).

Write Broadcasting. Обновление шины. Происходит, когда процессор изменяет блок кэша, а другим процессорам требуется обновление соответствующих блоков кэша. Это уникально для протоколов обновления. Обновление шины занимает меньше времени по сравнению с операцией Flush, поскольку запись в кэш выполняется быстрее, чем в память. Еще один момент, на который следует обратить внимание: кэш не может обновить свою локальную копию блока кэша, а затем запросить шину отправить обновление шины. Если это произойдет, то возможно, что два кеша независимо обновят свою локальную копию, а затем запросят шину. Тогда они увидят две записи одновременно, что не будет соответствовать последовательной согласованности.

Flash это происходит, когда процессор помещает на шину весь блок кэша. Это необходимо для отражения изменений, внесенных процессором в кэшированный блок в основной памяти.

Общая строка (shared line) необходима для указания того, доступен ли определенный блок кэша в нескольких кэшах. Это необходимо, поскольку один из кэшей может удалить блок без необходимости обновления других блоков. Общая линия помогает уменьшить количество транзакций памяти и шины в некоторых случаях, когда блок доступен только в одном кэше и, следовательно, обновление шины не требуется

1.4 Операции

В протоколе Dragon (Xerox) существует 4 операции:

- Write Allocate
- Intervention - от D-SD (но не от VE)
- Write-broadcasting
- Copy-Back: Замена состояния Dirty (D) на Shared-Dirty (SD).

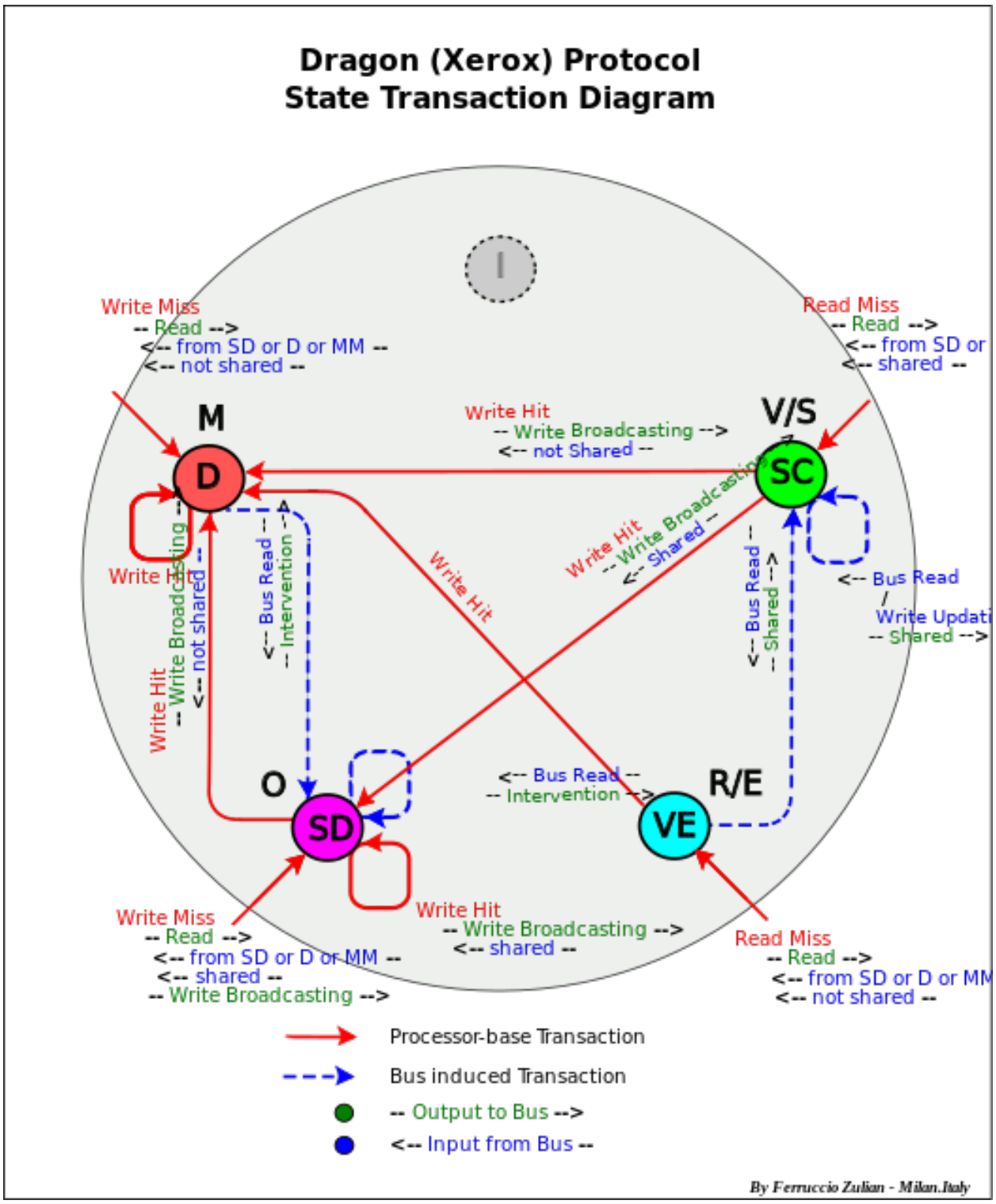

Рис 1: Диаграмма транзакций состояния протокола Dragon (Xerox)

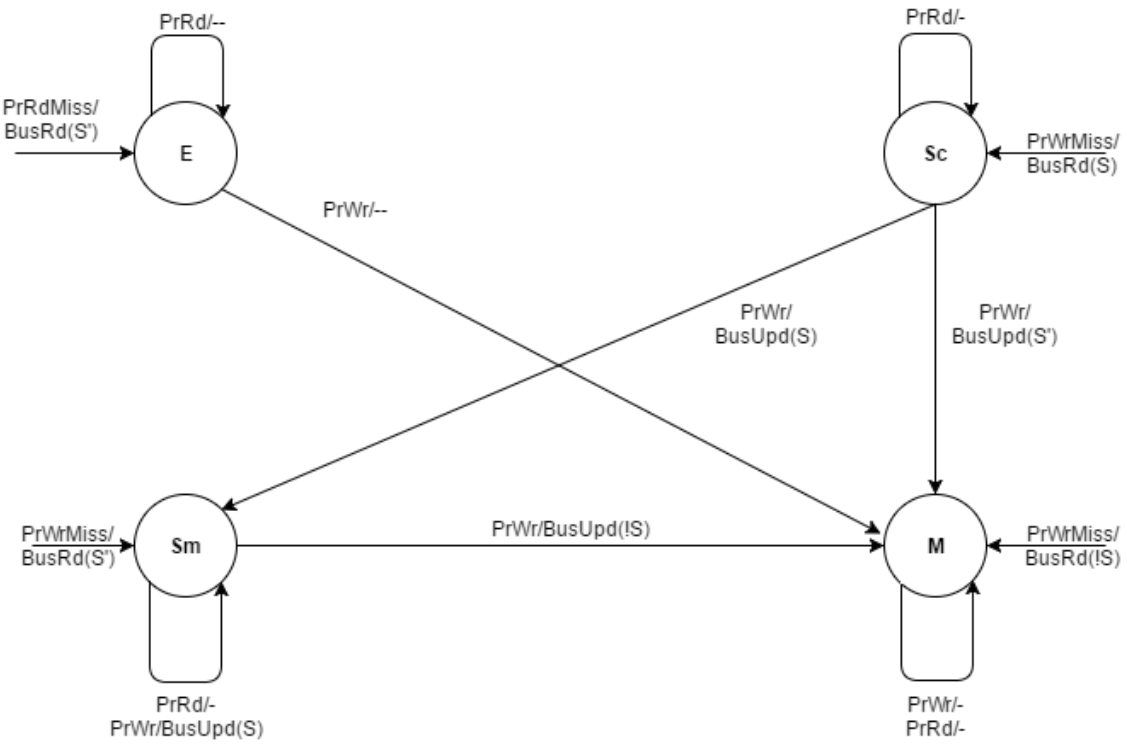

Рис 2: Транзакции, инициированные процессором

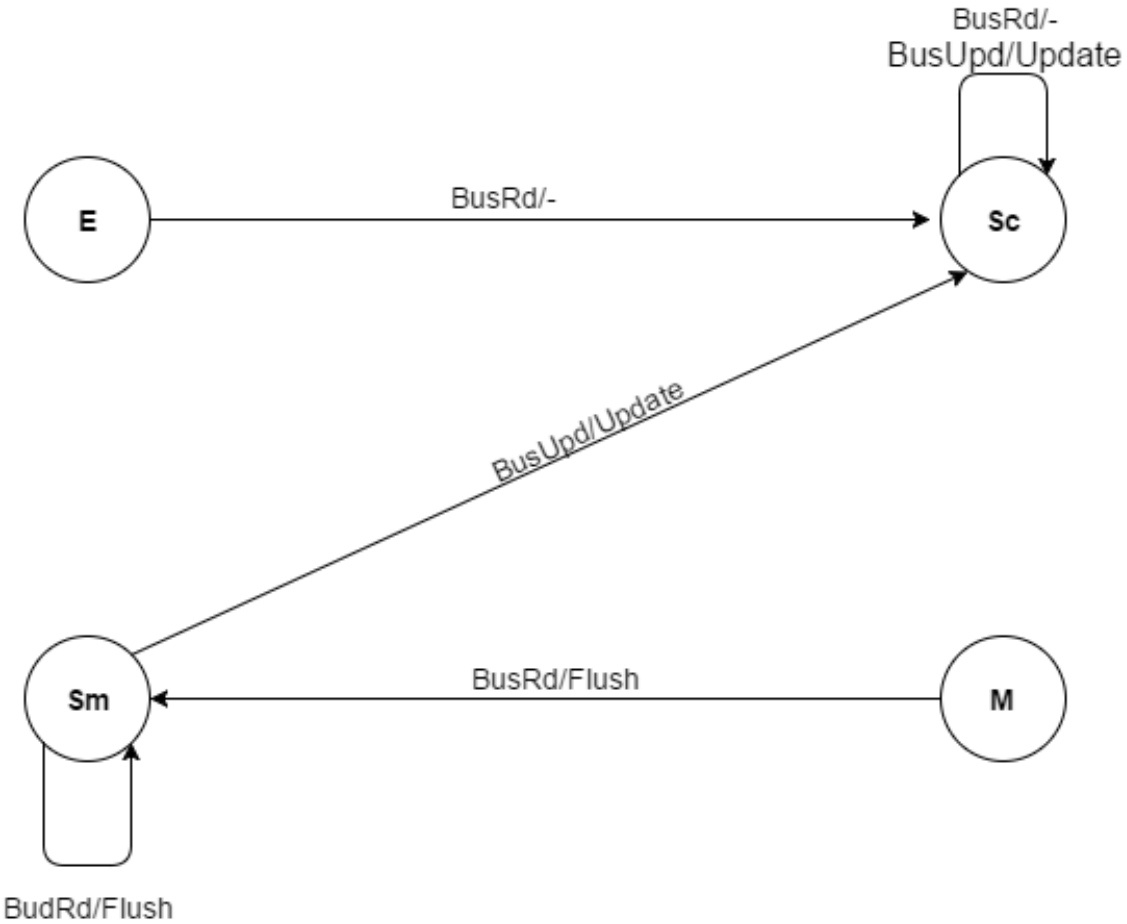

Рис 3: Транзакции, инициированные шиной

2 LRU

LRU (Least recently used) - политика вытеснения, которая в первую очередь вытесняет неиспользованные дольше всех данные.

Этот алгоритм требует отслеживания того, что и когда использовалось, что может оказаться довольно накладно, особенно если нужно проводить дополнительную проверку, чтобы в этом убедиться. Общая реализация этого метода требует сохранения «бита возраста» для строк кэша и за счет этого происходит отслеживание наименее использованных строк (то есть за счет сравнения таких битов). В подобной реализации, при каждом обращении к строке кэша меняется «возраст» всех остальных строк.

3 Литература

1. Алгоритмы кэширования. URL: https://ru.wikipedia.org/wiki/Алгоритмы_кэширования (Дата обращения: 14.04.2024)
2. Cache coherency protocols (examples). URL: https://en.m.wikipedia.org/wiki/Cache_coherency_protocols_(examples) (Дата обращения: 14.04.2024)
3. Cache coherence protocol analysis. URL: https://www.sobyte.net/post/2022-04/cache-coherence-protocol/ (Дата обращения: 14.04.2024)
4. Dragon protocol. URL: https://en.wikipedia.org/wiki/Dragon_protocol (Дата обращения 14.04.2024)