Синхронная Символьная Иерархия
Дезагрегация оперативной памяти

Автор : Балыбердин А.Л.

Новосибирск, 2022 г.

В настоящее время есть несколько направлений в развитии интерфейсов оперативной памяти вычислительных систем. Для оптимизации низкоуровневых интерфейсов это OMI (Open Memory Interface) и прочее, которые позволяют решать проблему нехватки электрических контактов для подключения внешней оперативной памяти. Появились первые полностью оптические варианты меж-чипового интерфейса.
https://ayarlabs.com/supernova/
https://3dnews.ru/assets/external/downloads/2020/12/04/1026993/demophot.mp4
Есть множество технологий позволяющих в той или иной степени дезагрегировать оперативную память (RDMA). Появились предпосылки к появлению процессоров, не нуждающихся в традиционной материнской плате для своей работы. Для таких процессоров есть возможность реализовать крайне плотную упаковку чипов и соответственно минимизировать задержки передачи данных между взаимодействующими элементами.
Но пока нет самого главного: Сети, универсальной и лишенной недостатков современных пакетных сетей связи.
Сеть ССИ является оптимальным решением для таких систем: крайне большие и гарантированные скорости передачи, малые стабильные задержки, отсутствие взаимного влияния виртуальных каналов, не требуется больших, да и вообще отдельных коммутаторов.

Как это может выглядеть реализация будущего суперкомпьютера:

1. Пластиковая подложка с большим числом впаянных в пластик оптических волокон, которые реализуют сеть соединений между компонентами вычислительной системы.
2. В подложке есть «вырезы» с оптическим интерфейсом для установки модулей процессора и любых других (память, внешние интерфейсы и прочее). Расстояния связи совсем небольшие и потому особых требований к точности реализации нет.
3. Процессор оформлен в виде модуля небольшой толщины (2-5мм) из двух пластин, хорошо проводящих тепло и электрический ток, между которыми находится кремниевая подложка с микросхемой процессора и прочими чиплетами.
4. На торцах такого модуля располагаются выводы оптического интерфейса, которые через оптический интерфейс соединяются с оптическим волокном подложки.
5. Верхняя и нижняя пластина представляют собой электроды для подачи питания на процессор. Расстояние между процессорами рассчитываются из требований по отводу выделяемого при работе тепла к радиаторам, одновременно являющимися и шинами питания. Сечение таких шин достаточно большое и позволяет передавать питающие токи в сотни кило ампер.
6. Размеры подложки могут быть практически любыми и эту подложку можно располагать самыми разными способами между последовательностями пластин радиатора (питания). Полученную подложку можно банально резать «ножницами» для получения требуемых размеров и формы, те подложка может стать универсальным продуктом с высокими тиражами.
7. При стандартизации оптического интерфейса, появится стандарт на оптико-электрический модуль, с возможностью универсальной установки в данную подложку. Если модуль расположен рядом с краем подложки, то есть возможность устанавливать любые внешние интерфейсы, разъем модуля выступает за пределы радиаторов.
8. Часть оптических интерфейсов можно использовать для связи «слоев» такого пирога напрямую, что позволит создавать сети с многомерной архитектурой.
9. Можно сделать вырезы без оптического интерфейса и использовать их передаче тепла от радиатора охлаждающей жидкости.

Такой DataFlow суперкомпьютер кубической формы со стороной в 10 метров, теоретически может иметь производительность 10E20 Flops и максимальной задержкой передачи данных менее 1 мкс (если удастся отвести от него один гигаватт тепла). Для соседних элементов задержки передачи составят первые наносекунды, что сопоставимо с задержками передачи данных в пределах кристалла. Это позволит решить проблему «стягивания» всей электроники процессора на один кристалл, которая из-за проблем с теплоотводом зашла в тупик и породила такое явление как «темный кремний».

Алгоритмы совместного доступа к разделяемому ресурсу на примере распределенной общей памяти (DSM)

Для понимания дальнейшего текста необходимо прочитать:

- *ССИ_Синхронный поток_Эффект проскальзывания (1)*
- *ССИ_Синхронный поток_Фрагментация потока (2)*
- *ССИ_Коммутатор синхронных потоков (3)*
- *ССИ_Управление списком виртуальных потоков (4)*
- *ССИ_Асинхронная передача данных (5)*
- *https://parallel.ru/krukov/lec6.html*

Учитывая все вышесказанное можно представить, как будет выглядеть модуль оперативной памяти. Такие модули (кремниевая подложка с чиплетами) должны иметь оптический интерфейс, электрический интерфейс не позволит передать весь трафик (100Т и более) за пределы подложки. В состав модуля входит контроллер виртуальной памяти и множество многослойных микросхем памяти типа НВМ3. Еще раз повторюсь никакого электрического интерфейса за пределы подложки.
Модуль процессора может вообще не содержать значимых объемов оперативной памяти и оптимальной конфигурацией будет использование современной КЭШ памяти третьего уровня для организации системы виртуальной памяти, хранения страниц виртуальной памяти (примерно, как в архитектуре X86). Для заполнения и вытеснения страниц виртуальной памяти использовать внешние модули оперативной памяти.

Наиболее оптимальным типом виртуальной распределенной памяти (мое мнение) является память с последовательной консистентностью. Такой тип памяти не гарантирует мгновенную одинаковость данных для всех копий разделяемого ресурса, но гарантирует что последовательность изменений будет строго одинаковой для всех клиентов. С учетом достаточно больших скоростей сети ССИ, состояние при котором происходит обновление будет очень небольшим и практически никак не зависит от числа копий. Время обновления зависит только от выделенной для конкретного разделяемого ресурса скорости и физического расстояния от исходных данных до копии.
Поскольку требуется соблюдение строгой последовательности обновлений, то должен быть модуль, который решает кто и в какой последовательности будет получать доступ к разделяемому ресурсу. Наиболее оптимальным будет назначить таковым, контроллер виртуальной памяти к которому присоединена разделяемая физическая память.
Кроме предоставления доступа, арбитража, рассылки обновлений, контроллер виртуальной памяти может предоставлять еще много различных «сервисов». Такими «сервисами» могут быть предоставление данных в формате и последовательности наиболее удобной для последующей обработки, возможно даже одновременно в разных вариантах для разных потребителей. Различных вариантов блокировок, приоритетов, предварительных подготовительных работ и предоставления данных к нужному времени и многое другое. Не будет преувеличением называть контроллер виртуальной памяти универсальным процессором управления и подготовки потоков обрабатываемых данных.

Число копий данных может быть очень большим и необходим механизм позволяющий осуществлять гарантированный доступ без излишнего резервирования ресурсов сети, но и с гарантией заранее заданной пропускной способности (гарантировать определенный минимум производительности). ССИ для этого предоставляет два уникальных механизма: возможность создавать канал передачи данных в виде дерева и асинхронный тип канала. Запросы на изменения «оригинала» данных от владельцев копий могут поступать достаточно асинхронно и потому их лучше передавать с использованием большого числа асинхронных каналов с некоторой гарантированной синхронной скоростью. Запросы на обновление данных в копиях, происходят достаточно часто (каждый запрос на изменение оригинала порождает запрос на изменение копии для всех владельцев копий данных), соответственно оптимальным будет один синхронный канал с достаточной производительностью. Для обеспечения сбалансированности системы, скорость синхронного канала должна быть пропорциональна сумме гарантированных скоростей асинхронных каналов. В таком случае контроллеру виртуальной памяти не потребуется буфер для хранения запросов из-за занятости канала обновления данных.

Примерный алгоритма работы (опускаем подготовительные операции):

- Если процессор уже содержит локальную копию данных и ему требуется изменить конкретное значение. Будет создан запрос на изменение данных, установлена метка в локальной памяти о неактуальности содержимого данной ячейки для локального процессора, далее процессор продолжит свою работу, не дожидаясь завершения работы по актуализации данных (алгоритм КЭШ с прямой записью).
- Через некоторое время запрос будет получен модулем виртуальной памяти, хранящем «оригинал», обработан и в синхронный виртуальный канал будет отправлена информация для обновления данных во всех локальных копиях.
- Виртуальный канал рассылки обновлений строится как дерево, корнем которого является владелец «оригинала» данных. При поступлении данных в промежуточный коммутатор, они могут быть отправлены сразу в несколько выходных физических каналов нескольким коммутаторам одновременно (нет необходимости строить канал, последовательно обходящий всех владельцев копий или обращаться к ним последовательно). Такой подход к отправке уведомлений минимизирует задержку передачи, которая никак не зависит от числа копий.
- После получения от контроллера виртуальной памяти обновленных данных, происходит изменение содержимого локальной памяти и сброс сигнала о неактуальности конкретного слова (для владельца, который породил это изменение)
- Процессор, владеющий локальной копией (если он ее перед этим не изменял) свободно читает содержимое ячеек памяти, никак не синхронизируя с процессом их обновления. Успел прочитать не обновленные данные считаем, что чтение случилось раньше, чем другой процессор их изменил. Даже если по абсолютному времени данные уже были обновлены, но не доставлены конкретному владельцу копии.
- Процессор, владеющий копией (если он недавно обновил данные и имеется метка о неактуальности) может вести себя по-разному. Он может проверить значение метки и выполнить какое-то действие в соответствии со значением метки. Может просто попытаться прочитать значение ячейки памяти и быть «остановленным» до момента пока не будет сброшена метка о неактуальности (не получено именно его обновление от контроллера виртуальной памяти — своеобразная синхронизация вычислительного процесса относительно последовательности изменений).
- Можно и вообще принимать во внимание только локальную последовательность изменений данных.
- Если запрос на изменение потерян (ошибка передачи), то данное событие может быть детектировано в течении первых микросекунд по отсутствию данных обновления на

свой запрос изменения (таймаут времени неактуальности ячейки памяти)

Данный тип алгоритма достаточно прост в реализации и будет хорошо работать для высокоскоростной ССИ. Скорость в 100Т почти на три порядка превышают производительность локальной динамической памяти, время «открытия» новой страницы динамической памяти сопоставимо с временем распространения электромагнитной волны на расстояние в 10 метров, что делает распределенную виртуальную память сопоставимой по скорости с локальной динамической памятью и по факту лишает смысла нахождение динамической памяти рядом с процессором. Конкретные протоколы взаимодействия процессора и модуля оперативной памяти могут быть сильно разными, но все их объединяют и в значительной степени определяют сервисы сети ССИ.