Умножение плотных матриц на неоднородных высокопараллельных вычислительных системах (анализ коммуникационной нагрузки)

А.Климов

Аннотация

При распараллеливании задач на системе из большого числа процессоров важное значение имеет коммуникационная нагрузка на сеть и способность сети справляться с этой нагрузкой. На примере задачи умножения плотных матриц изучаются требования, предъявляемые к коммуникационному обеспечению вычислительной системы. В качестве эталона системы рассматривается проект суперкомпьютера Merrimac [1]. Ее развитая многоуровневая коммуникационная сеть, обеспечивает хорошие пропускные способности на разных уровнях локальности. Обосновывается тезис о важности снижения коммуникационных затрат за счет выбора оптимального распределения вычислений по процессорным элементам. Демонстрируется возможность выбора оптимального распределения, не зависящего от параметров сети.

1. Введение

При анализе эффективности параллельных программ обычно рассуждают о параллелизме выполнения операций. Дополнительно иногда принимают в расчет задержки на передачу данных между процессорами, считая, что они для всех пар процессоров одинаковы, или по крайней мере ограничены некоторой константой (см. например, [6]). И лишь очень редко встречаются оценки на основе учета конфликтов в коммуникационной сети, или, говоря иначе – учета ограничений по ее пропускной способности.

В данной работе исследуется влияние пропускной способности коммуникационной среды на производительность параллельной вычислительной системы на примере задачи умножения матриц.

Обычно в качестве меры пропускной способности используют понятие бисекционной пропускной способности. Мы увидим, что этот подход недостаточен, поскольку в нем не принимается в расчет сложная многоуровневая организация сети. Фактически учитываются только свойства сети самого верхнего уровня организации системы.

Нас интересуют очень большие системы, состоящие из многих тысяч процессоров. Они с неизбежностью будут иметь иерархическое строение, связанное с технологией их разработки и производства. Например, если в одном кристалле (чипе) могут находится десятки и сотни вычислительных устройств, то взаимодействие между устройствами одного кристалла будет скорее всего обеспечено гораздо лучше, чем взаимодействие между устройствами из разных кристаллов

В качестве модельной вычислительной системы мы будем рассматривать проект системы Merrimac, разработанный в Станфордском университете [1]. И хотя он так и не был воплощен в железо, его идейный потенциал далеко не исчерпан. Его можно считать образцом хорошо сбалансированной высокопроизводительной вычислительной системы. Для нас важным является его коммуникационное обеспечение, которое «заточено» на эффективную передачу интенсивного потока

мелких сообщений (до нескольких байтов). В разделе 1 даются основные существенные для нас сведения об этой системе. Мы используем ее лишь в качестве примера для иллюстрации нашего подхода к оценке производительности, но с равным успехом его можно применить и для других систем.

Затем, в разделе 2, проводится анализ алгоритма умножения плотных матриц с точки зрения возникающей при его выполнения коммуникационной нагрузки. При этом мы не ориентируемся ни на какую конкретную модель вычислений или язык программирования, и не используем никакого конкретного программного кода. Будем исходить просто из постановки математической задачи и математического же представления о вычислении как о некоем вычислительном графе, где в вершинах находятся скалярные операции типа сложения или умножения, а по ребрам передаются данные от результатов одних операций к аргументам других. Мы вводим понятие коммуникационной сложности алгоритма как наименьшей коммуникационной нагрузки на сеть, соединяющую К процессорных элементов, среди которых равномерно распределяется вычислительных граф. Для умножения матриц порядка N дается оценка коммуникационной сложности снизу в виде формулы $O\ N^2\sqrt[3]{K}$.

Наши рассуждения применимы к любой реализации математической схемы алгоритма и любым уровням организации системы (FPU, чип, плата и т.п.). Это и делается в разделе 3 для различных уровней системы Merrimac. Показывается, что производительность существенно зависит от распределения вычислений по элементам вычислительной системы. Наконец, в разделе 4 описывается конструктивно метод распределения, обеспечивающий минимум коммуникационной нагрузки на всех уровнях системы.

В качестве дополнительной иллюстрации подхода, в разделе 5 даются аналогичные оценки для другой гипотетической системы – HTMT. Обосновывается тезис, что при оценивании и сравнении производительности на основе теста по умножению плотных матриц важен такой показатель, как минимальный размер матриц, при котором все еще достигается называемая проектная производительность.

2. Структура системы Merrimac

Система Merrimac с проектной производительностью 2 Пфлопс имеет иерархическую структуру, в которой насчитывается 5 уровней (6 вместе с уровнем всей системы). На первом уровне находится функциональное арифметическое устройство с плавающей точкой (FPU), способное вычислять одно сложение и одно умножения на каждом такте при частоте 1 ГГц. У каждого FPU имеется своя небольшая регистровая память (по 64 регистра длиной 64 бита на операнды и результат). FPU объединяются в кластеры по 4. В одном чипе находятся 16 таких кластеров. Далее чипы (nodes) размещаются по 16 штук на плате. 32 платы объединяются в стойку. 16 или 32 стойки образуют систему.

На каждом уровне объединения используется коммуникационная среда, обеспечивающая взаимодействие между объединяемыми компонентами, а также канал обмена между ними и вышестоящей средой. В таблице 1 приведены параметры коммуникационной среды на каждом уровне структурной иерархии. Для компонентов каждого уровня указаны:

- число компонентов предыдущего уровня, из которых состоит данный компонент,
- число FPU, содержащихся в данном компоненте,

- пиковая производительность (млрд. операций в секунду) данной единицы,
- ширина (пропускная способность) каналов связи данной единицы с ее внешней средой (Гбайт в секунду),
- удельная пропускная способность, исчисленная как отношение пропускной способности к объему данной единицы (здесь – к числу содержащихся в ней узлов).

Таблица 1. Параметры коммуникационной среды системы Merrimac

Компонент уровня	Число компонентов предыдущего уровня	Число минимальных элементов (FPU)	Производительность (ГФлоп/с)	Пропускная способность канала взаимодействия данного компонента с его внешней средой (Гбайт/с)	Пропускная способность сети, объединяющей компоненты данного уровня, в пересчете на 1 узел (Гбайт/с)
FPU	-	1	2	8	512 (внутрикластерный оммутатор)
Кластер	4 FPU	4	8	4	64 (межкластерный коммутатор)
Узел (чип)	16 кластеров	64	128	20	20
Плата	16 узлов	1024	2048	80	5
Стойка	32 плат	32K	64000	1280	2.5
Система	16 стоек[1]	512K	1000000		-

На некоторых уровнях (FPU, кластер, узел) имеется память (регистровая, стримовая, кэш + DRAM). Каждое FPU использует свою локальную регистровую память объемом 192 64-разрядных слов. Каждый кластер через внутрикластерный коммутатор пользуется своим банком стримовой памяти емкостью 8Кслов. Все кластеры чипа через межкластерный коммутатор используют общий кэш, выходящий на внешние блоки DRAM. Память каждого вида приписана тому структурному уровню, на котором к ней имеется непосредственный доступ, а доступ из других этажей нагружает соответствующую коммуникационную среду. Здесь нас будет интересовать только пропускная способность коммуникационных сред разного уровня как основной лимитирующий фактор.

Из Таблицы 1 видно, что на разных уровнях имеются разные пропускные способности в пересчете на одну единицу (узел или ФАУ), причем с ростом уровня эти пропускные способности убывают. По-видимому, это связано с возрастанием стоимости связей по мере повышения уровня в связи с увеличением их протяженности.

Есть два пути решения этой проблемы. Можно стремиться обеспечивать равную пропускную способность на всех уровнях, и тогда программисты будут практически избавлены от необходимости думать об оптимальном распределении вычислений и данных. Проблема будет оставаться только в некотором различии задержек, но оно

[1] Для удобства сопоставления с другими системами используется модель половинного размера с проектной пиковой производительностью ровно в 1 ПФлопс.

вполне может скрываться избыточным параллелизмом задач (при наличии достаточного места для сохранения состояний ожидающих процессов). Но цена этого пути может оказаться высокой, и основная доля стоимости системы будет приходиться на коммуникационное оборудование самого верхнего уровня.

Другой путь состоит в том, чтобы посредством надлежащего программирования обеспечивать значительное преобладание ближних передач данных над дальними. Для одних задач это проще, для других труднее. Зато будет заметно дешевле оборудование – в ущерб производительности некоторых «трудных» задач.

Возможность снижения удельного веса дальних передач для конкретной задачи может иметь свои пределы. Ниже мы проанализируем на предмет этих пределов задачу умножения матриц. Исходя из полученных оценок, будут выведены ограничения на производительность, связанные с передачами данных между компонентами системы.

3. Задача умножения матриц

Результатом умножение двух матриц **A** и **B** размеров, соответственно, $N_1 \times$ _ и $N_0 \times$ _ является матрица **C** размера $N_1 \times$ _, с элементами:

$$C_{ij} = \sum_{k=}^{N} \qquad (1)$$

Данную задачу (обычно при $N_0 =$ _ = _ =) часто используют для оценки потенциальных возможностей высокопроизводительных вычислительных систем (суперкомпьютеров). Это связано с тем, что в ней:

- число операций заметно больше чем объем вводимых данных (N^3 против N^2);
- граф связей достаточно тесный, и задача плохо разбивается на слабо связанные подзадачи;
- структура задачи достаточно проста для проведения анализа эффективности;
- у нее нет простых альтернативных заметно более эффективных решений.

Для формализации второго свойства введем понятие *коммуникационной сложности* задачи. Это функция $S\ N, K$ от размера задачи N и числа процессоров K. Ее значение показывает минимально возможный объем пересылаемой между процессорами информации при равномерном распределении вычислительной нагрузки. Можно показать (и ниже это будет сделано), что коммуникационная сложность умножения матриц размера $N \times$ _ равна $O\ N^2 \sqrt[3]{K}$.

Используя понятие коммуникационной сложности можно дать оценку снизу времени выполнения задачи при ее распределенном выполнении на системе из K элементов. В роли элементов могут выступать любые структурные единицы (плата, процессор, …). Соответственно, коммуникационную сложность, вычисленную исходя из общего числа этих единиц следует сопоставить с общей пропускной способностью

При умножении двух матриц согласно формуле (1) необходимо выполнить $N_0 N_1 N_2$ скалярных умножений и почти столько же сложений. Несмотря на то, что существуют теоретически более эффективные (в смысле числа умножений) решения [4], мы будем исходить из прямого вычисления данной формулы. Будем допускать только вариации, связанные с ассоциативностью и коммутативностью сложений.

Пусть имеется K процессорных модулей, соединенных коммуникационной сетью. Множество операций скалярного умножения, выполняемых в алгоритме,

представляет собой параллелепипед размера $N_0 \times \_ \times \_$. Он должен быть как-то разбит на части, распределяемые среди K процессорных модулей.

Не вдаваясь в доказательство, примем достаточно правдоподобное допущение, что оптимальное (в смысле экономии пересылок) распределение устроено следующим образом. Каждое измерение параллелепипеда разбивается, соответственно, на K_0, K_1 и K_2 равных частей, где $K_0 K_1 K_2 =$. В каждом процессоре исполняется один из блоков, ограниченный плоскостями, проведенными через точки деления (см. рис 1).

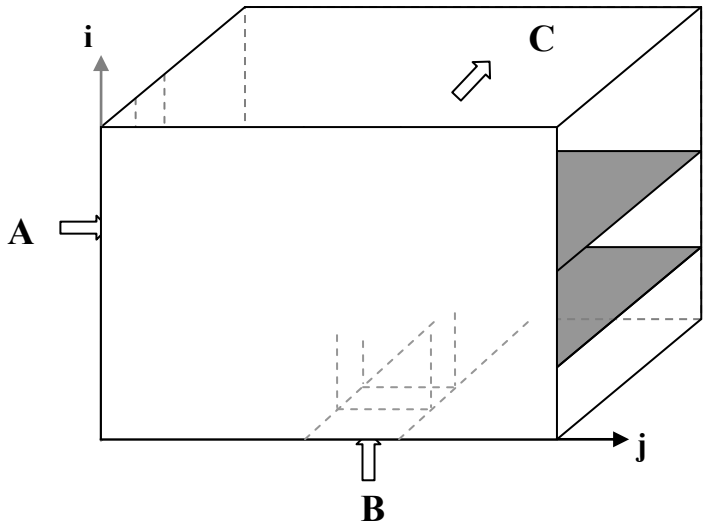

Рис. 1. Пространство узлов вычислительного графа с операцией умножения. Внутри выделена область, выполняемая в отдельном процессорном элементе. Элементы исходной матрицы $\boldsymbol{A}_{ik}$ распространяются вдоль направления **j**, матрицы $\boldsymbol{B}_{kj}$ – вдоль направления **i**. Результат $\boldsymbol{C}_{ij}$ снимается вдоль направления **k**.

В модуле номер $\langle k_0, k_1, k_2 \rangle$ (где k_l - номер части соответствующего направления) блок номер $\langle k_1, k_0 \rangle$ матрицы **A** умножается на блок номер $\langle k_0, k_2 \rangle$ матрицы **B**. Результатом является слагаемое для блока номер $\langle k_1 \cdot$ матрицы **C**. В каждый модуль приходят два входных блока, и выходит один выходной. Их размеры – $d_1 \times$, $d_0 \times$ и $d_1 \times$ соответственно, где $d_l =$. Общая нагрузка (в словах) на один процессорный модуль составляет:

$$S_{PROC} = \quad + \quad + \quad = \quad + \quad + \qquad (2)$$

И тогда общая коммуникационная сложность

$$S_{TOTAL} = \quad = \quad + \quad + \quad = \quad \left(\qquad \right) \qquad (3)$$

Если считать размеры всех матриц и общее число процессоров K заданными, то в скобках будет стоять сумма трех чисел с фиксированным произведением. Ее минимум будет достигаться при равных слагаемых, то есть все блоки должны быть как можно ближе к квадратным. В этом случае коммуникационная сложность задачи составит:

$$S_{OPT} = \sqrt[3]{\frac{K}{N_0 N_1 N_2}} \tag{4}$$

В частности, при подстановке $N_0 =$ = = получаем

$$S_{TOTAL} = + + \tag{5}$$

$$S_{OPT} = \tag{6}$$

Мы пока не приняли в расчет операции межблочного сложения и соответствующие передачи данных. Но в этом нет необходимости, поскольку сложения не меняют количества передаваемой информации. Если требуется выдать сумму K чисел, хранящихся в K разных процессорах, то, как ни крути, придется передать через коммутатор не менее K чисел.

Полученные оценки позволяют оценить снизу время выполнения задачи, если известна общая пропускная способность сети BW или удельная пропускная способность $BW_{PROC} =$:

$$T_{COMM} \geq = \cdot \tag{7}$$

Интересно сопоставить коммуникационную сложность с нагрузкой сети при далеком от оптимального распределении вычислительных операций по процессорам. В частности, если использовать равномерно-случайное разбрасывание, то величина трафика составит:

$$S_{RAN} = \frac{K}{K} - \tag{8}$$

При больших K сомножителем $\frac{K-}{K}$ можно пренебречь. Получаем

$$\frac{S_{RAN}}{S_{OPT}} \approx \sqrt[3]{K} \tag{9}$$

При $K \approx$ формула (9) утверждает, что случайное разбрасывание будет также оптимальным. Не удивительно, поскольку в этом случае каждая операция назначается на свой отдельный процессор, а стоимости пересылок между разными процессорами считаем не зависящими от номеров процессоров.

Если K , то коммуникационная нагрузка в оптимальном разбиении будет в $\sqrt[3]{\frac{N^3}{K}}$ раз лучше, чем в случайном. Соответственным будет и выигрыш, если сеть загружена на пределе.

4. Анализ производительности суперкомпьютера Merrimac на задаче умножения матриц

При изучении эффективности параллельной системы обычно рассматривают два аспекта: загрузку вычислительного оборудования и загрузку коммуникационных каналов. Если предположить, что параллелизм задачи более чем достаточен для того, чтобы нагрузить на 100% все арифметические устройства, то производительность может сдерживаться либо со стороны FPU, либо со стороны коммуникационной сети. Зная количество K и производительность PR вычислительных компонентов, а также общее число операций в задаче N_{OP}, легко рассчитать предельное вычислительное время $T_{COMP} =$. С другой стороны, зная пропускную способность межкомпонентного обмена BW (на компонент) и

нагрузку S, можно рассчитать предельное коммуникационное время $T_{COMM} = \frac{S}{K \cdot}$. Далее, предполагая возможность совмещения обменов с вычислениями, имеем следующую оценку снизу для времени выполнения алгоритма:

$$T = \quad _{OMP}, T_{COMM} \tag{10}$$

Система Merrimac имеет несколько уровней иерархии, и для разных уровней будут разные значения T_{COMM}. Вычислим оценки для задачи умножения матриц размера 1024x1024. На каждом уровне иерархии будет свое оптимальное разбиение параллелепипеда скалярных операций по компонентам. Для удобства будем выбирать его так, чтобы все K_l были степенями 2. Тогда условием оптимальности будет не равенство всех K_l, а то, что они должны отличаться друг от друга не более чем в два раза.

В Таблице 2 приведены данные по значениям K, PR и BW для компонентов разных уровней и вычисленные значения T_{COMM} для случая $N_0 = \_ = \_ = \quad =$. Для вычислительного времени имеем

$$T_{COMP} = \frac{2N^3}{1 PFlops} = \tag{11}$$

Таблица 2. Расчет коммуникационного времени для $N =$ **,** $T_{COMP} =$

Уровень	Число элементов, K	Пропускная способность, BW, Гслов/с	Наилучшее разбиение, $K_0 + \_ + \_$	Коммуникационная нагрузка на сеть, $S = \quad + \quad +$ Мслов	$T_{COMM} = \frac{S}{K \cdot} =$ мксек
FPU	512K	1	64+64+128=256	256	0.5
Кластер	128K	0.5	32+64+64=160	160	2.5
Узел	8K	2.5	16+16+32=64	64	3.2
Плата	512	10	8+8+8=24	24	4.8
Стойка	16	160	2+2+4=8	8	3.2
Система	1				

Для каждого уровня получилось свое время T_{COMM}. Наибольшее из них (4.8мксек) и будет определять нижнюю оценку. Поскольку она получилась больше, чем T_{COMP}, приходится признать, что на данной задаче производительность $1ПФлопс$ недостижима. В лучшем случае будет $1ПФлопс \frac{T_{COMM}}{T_{COMP}} = \quad лопс$.

Можно улучшить показатель производительности, если увеличить размер матриц. Дело в том, что число операций растет пропорционально N^3, а нагрузка на коммуникационные сети - пропорционально N^2. При любом N наихудшее коммуникационное время будет на уровне взаимодействия между платами. В Таблице 3 показаны времена T_{COMM} и T_{COMP}, а также максимально возможную производительность для различных N.

Таблица 3. Времена вычислений и коммуникаций для различных N

N	T_{COMP} , *мкс*	T_{COMM} , *мкс*	*ПФлопс*
256	0.03	0.3	0.10
512	0.25	1.2	0.21
1024	2	4.8	0.42
2048	16	19.2	0.84
4096	128	76.8	1.0
8192	1024	307.2	1.0

Таким образом, на данной системе в силу ограничений ее коммуникационной сети предельная производительность в 1ПФлопс достижима на задаче умножения матриц размера не ниже 4096x4096. Сохранение производительности при понижении размера матриц вдвое требует соответствующего удвоения пропускной способности коммуникационных сред, в первую очередь тех, которые работают на пределе. Основным ограничителем в данном случае является сеть между платами. Было бы разумно увеличить ее пропускную способность хотя бы в 1.5 раза. Тогда пропускные способности трех верхних уровней были бы более сбалансированными.

5. Достижимость оценок

Нагрузки *S* (см. Табл.2), вычисленные в предыдущем разделе, являются нижними оценками. Они получены из рассмотрения некоторого класса отображений вычислительных операций на аппаратную среду. Выведенные общие формулы были независимо применены к различным структурным уровням, причем оценки, получаемые для каждого уровня, реализуются на своем отображении. Возникает вопрос, а достижимы ли все эти оценки одновременно, то есть возможно ли такое отображение вычислительных операций на совокупность из всех вычислительных устройств, при котором достигаются все полученные оценки.

Ответ утвердительный. Действительно, существует отображение, которое обеспечивает оптимальность распределения на всех уровнях. Чтобы его описать, введем единую нумерацию всех арифметических устройств (FPU). Полный номер FPU на рассмотренной конфигурации системы кодируется 19-разрядным двоичным числом, составленным из полей, кодирующих номер FPU в кластере, номер кластера в узле, номер узла в плате и т.д. расположенных справа налево (рис 2).

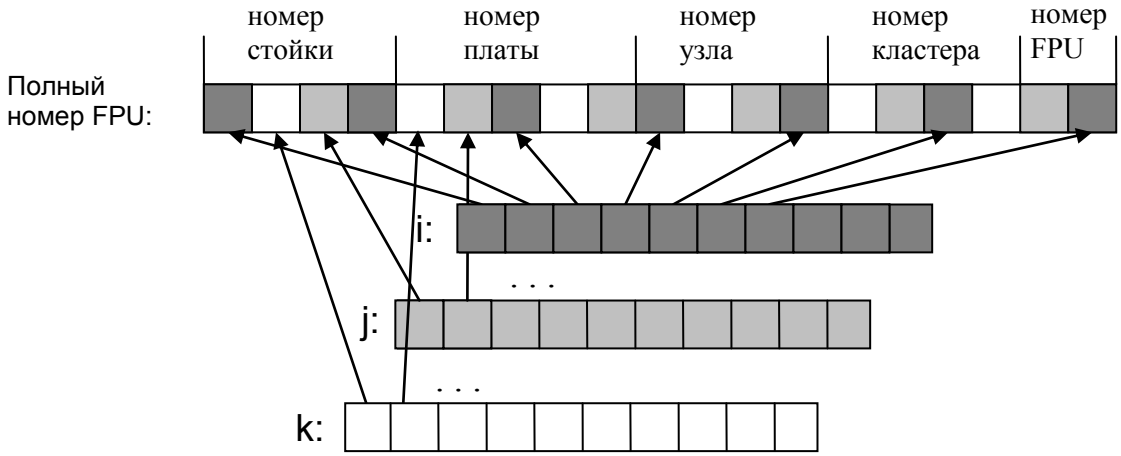

Рис.2. Отображение двоичных разрядов индексов i,j,k в двоичные разряды полного номера FPU.

Операции умножения, подлежащие отображению в FPU кодируются тройкой индексов: <i,k,j>, где <i,k> – индексы элемента первой матрицы, <k,j> – индексы элемента второй матрицы. Каждый индекс имеет десять двоичных разрядов (для N=1024). Для получения полного номера FPU по индексам <i,k,j> двоичные разряды последних скрещиваются как показано на рисунке.

Рассмотрим какой-либо уровень системы, например уровень, где компонентами являются платы. Полный номер платы состоит из номера стойки и номера платы внутри стойки и содержит 9 двоичных разрядов. Среди этих разрядов на каждый из трех индексов приходится ровно по три разряда. Это разряды старшие. Следовательно, на конкретную плату попадают тройки индексов, у которых в старших трех битах всех индексов определенные общие значения. Отсюда видно, что блок операций конкретной платы имеет размеры, равные 1/8 от размеров полного параллелепипеда (в данном случае – куба). Поэтому для плат $K_0 =$ $=$ $=$, что в точности совпадает с оптимальными параметрами, указанными в Таблице 2. Легко проверить, что для остальных уровней тройки $\langle K_0, K_1, K_2 \rangle$ также соответствуют указанным в Таблице 2. Таким образом, построенное отображение действительно является оптимальным одновременно для всех уровней системы.

Математически это отображение можно выразить с помощью функции **zip3**(a,b,c), которая выдает целое число, вычисленное по следующей рекурсивной схеме:

zip3(0,0,0) = 0

zip3(2*a+d,2*b+e,2*c+f) = (((zip(a,b,c)*2+d)*2+e)*2+f

где **d**, **e** и **f** могут принимать значения только из $\{0,1\}$. Тогда отображение запишется в виде формулы:

$$N_{FPU}(i,k,j) = \qquad _{FPU} * zip(i,k,j) / N^3) \qquad (12)$$

В зависимости от языка программирования это отображение в той или иной форме должно быть частью дизайна параллельного алгоритма.

Заметим, что указанное отображение не зависит от числа уровней и особенностей каждого уровня, хотя зависит от общего количества элементарных вычислительных устройств K_{FPU}. От последней зависимости тоже можно

избавиться, если кодировать номер FPU не целым числом, а дробным, лежащим в диапазоне 0..1. Тогда формула для отображения упроститься до

$$N(i,k,j) = \qquad j)/N^3 \tag{13}$$

где «/» понимается как деление с вещественным результатом. Эта формула определяет оптимальное отображение при любом выборе элементарной единицы (FPU, узел, и т.п.). И есть основания полагать, что алгоритм с таким отображением является оптимальным для систем с любой коммуникационной структурой. Главное, чтобы нумерация аппаратных единиц соответствовала аппаратной связности, когда коммуникационно близкие единицы имеют близкие номера.

6. Аналогичные работы

Для сравнения упомянем работу [5], где на примере этой же задачи обсуждается производительность другой гипотетической системы – HTMT (Hybrid Technology Multi-Threaded architecture), основанной на криогенной технологии одноквантовой логики RSFQ. Для получения предельных производительности в 1.1 ПФлопс авторы используют умножение матриц с N=208000, выполняемое за 16.2 секунд. Как следует из наших рассуждений, такой тест не раскрывает достоинства системы в должной мере. Интересно было бы найти наименьший размер матриц, которые умножались бы с такой же производительностью.

В самой статье [5] размер матриц выбран как наибольший, при которой она помещается в памятях разных уровней. Реализован трехуровневый блочный алгоритм, где на среднем уровне применяется систолический алгоритм Кэннона на двумерной решетке процессорных элементов SPELL размера 64x64. На ней умножаются подматрицы размера 8000x8000 за 0.75мс, что равнозначно производительности с производительностью чуть более 1ПФлопс . Нас интересует возможность уменьшить размер матриц с сохранением петафлопсной производительности.

Рассмотрим умножение матриц размера 1024x1024. На каждый из 4096 SPELL приходится по блоку размера 16x16. Каждый SPELL имеет 5 FPU с циклом 15пс. Поэтому умножение блоков (при полной загрузке FPU) потребует время (обозначения взяты из [5])

$$M_S = \cdot \quad \times \frac{15}{5} = \tag{14}$$

Передача блока между соседними SPELL потребует время

$$D_S = \qquad \times \frac{1}{16./1\,слов/с} = \tag{15}$$

Поскольку $M_S >$ _, время пересылок может быть скрыто их совмещением с вычислениями (хотя это потребует дополнительных ухищрений), и потому общее время работы алгоритма составит $64M_S =$. Дополнительно следует учесть время загрузки и выгрузки результатов, которое, исходя из скорости работы внешней памяти SRAM, можно оценить в 0.3 мкс. Тем самым на этом размере производительность 1ПФлопс достигается, хотя и на пределе. А на размере 512x512 это будет уже не так, поскольку время пересылок уже будет на 20% превосходить время вычислений. Таким образом, минимальный (из выражающихся степенью двойки) размер матрицы, при умножении которых достижим (теоретически) петафлопсный уровень есть 1024.

Из этого мы можем также сделать вывод, что коммуникационное обеспечение системы HTMT в целом примерно в 3-4 раза лучше, чем у Merrimac (по крайней

мере для задач с коммуникационной сложностью подобной той, которая свойственна умножению матриц). Не так много, если учесть, что оно основано на гораздо более продвинутой технологии RSFQ.

7. Выводы

Показателем «распараллеливаемости» задачи является не только ширина ее вычислительного графа (число операций, которые могут выполняться параллельно), но и объемы возникающих информационных потоков. Пределы в реализации вычислений могут проистекать не только из ограничений по производительности арифметических устройств и латентности (задержек) при передаче данных, но и из пропускной способности коммуникационной сети. Важно уметь оценивать свойство алгоритмов оказывать давление на сеть и оптимизировать их с этой точки зрения.

При равной стоимости сетей разного уровня, пропускные способности сетей более высоких уровней обычно получаются меньше (в пересчете на один процессорный модуль). Поэтому при распределении вычислительного графа по процессорным модулям следует стремиться к тому, чтобы большая часть связей падала на сети более низких уровней. Если этого не делать, полагаясь, например, на случайное разбрасывание узлов графа по модулям, то большая часть коммуникационной нагрузки будет падать на сеть самого верхнего уровня (с самыми дальними связями), обладающей самой низкой удельной пропускной способностью, со всеми вытекающими из этого последствиями для производительности системы.

Обычно в качестве меры пропускной способности сети используют *бисекционную пропускную способность*. Из наших рассуждений вытекает, что этого недостаточно. Бисекционная пропускная способность характеризует пределы потока «самых дальних» передач. Но, если задача обладает локальностью, то близкие передачи преобладают над дальними, и, как следствие, могут стать более существенными пределы для потока на близкие расстояния. Так, при умножении матриц размера 1024x1024 на системе Merrimac, основным сдерживающим фактором является сеть между платами. А сеть между стойками заполнена на 2/3. Поэтому снижение бисекционная пропускной способности на 30% на нашей задаче пройдет незамеченным. То есть как показатель пропускной способности сети в целом она не достаточна. Необходимо обобщение. Например, можно определить k-секционную пропускную способность. Разбиваем систему на k примерно равных частей, выбираем произвольно одну из них и определяем максимальных поток из этого части в направлении остальных. Минимум этого потока по всем разбиениями и будет значением k-секционной пропускной способности. При $k =$ получается обычная бисекционная. И теперь пропускная способность сети выражается не одним числом, а некоторой функцией от k, где а аргумент k изменяется от 2 до N, где N – число минимальных функциональных вычислительных устройств.

Задачи должны программироваться и распределяться так, чтобы коммуникационная нагрузка на сеть была оптимальной. При этом желательно, чтобы не приходилось подстраиваться под особенности коммуникационной аппаратуры. Для задачи умножения матриц нам это удалось. Достижение этой цели можно сравнить с разработкой так называемых кэш-независимых (cash-oblivious) алгоритмов [3].

8. Благодарности

Благодарю В.Н.Балина – за привлечение меня к данной работе, Д.С.Северова за подчеркивание важности проблемы коммуникаций, В.А.Александрова – за побуждение к работе по написанию статьи, А.М.Степанова – за ценные замечания, А.С.Окунева – за моральную поддержку.

9. Ссылки

1. Dally,W. et al, Merrimac: Supercomputing with streams. http://merrimac.stanford.edu/publications/sc03_merrimac.pdf
2. Э.Танненбаум. Архитектура компьютера, пер.с англ, 4-е изд. СПб.: Питер, 2002.
3. M.Frigo, C.E.Leiserson, H.Prokop, and S.Ramachandran. Cache Oblivious Algorithms. In 40th Annual Symposium on Foundations of Computer Science. pp.285-298. New York, USA, Oct.1999.
4. Strassen,V., Gaussian Elimination is Not Optimal. *Numerische Matematik*, vol.13, pp 353-356, 1969.
5. Amaral,J.N. et al., An HTMT Performance Prediction Case Study: Implementing Cannon's Dense Matrix Multiply Algorithm, CAPSL Tech.Memo 26, February, 17, 1999, University of Delaware, USA.
6. Wilkinson,B., Allen,M., Parallel Programming, 2d edition, Pearson Prentice Hall, 2005.