Анализ структуры коммуникационных взаимодействий процессов для решения задачи мэппинга параллельных программ

К.П. Бобрик[1], Н.Н. Попова[1]

МГУ им. М.В. Ломоносова[1]

В статье рассматривается новый взгляд на классическую задачу размещения процессов параллельных приложений по процессорам многопроцессорной вычислительной системы. Впервые предлагается метод анализа структуры коммуникационных взаимодействий, на основе которого могут быть построены алгоритмы мэппинга с привязкой к конкретной топологии. Основой предлагаемого метода являются данные об обмене информацией между процессами в форме матрицы коммуникационных взаимодействий. Множество процессов симулируется множеством точек на плоскости, которые эволюционируют под действием сил притяжения и взаимного отталкивания. Предлагается использовать найденные конфигурации распределения точек для размещения процессов по вычислительным узлам системы.

Ключевые слова: мэппинг, суперкомпьютеры, параллельные вычисления.

1. Введение

Современное развитие суперкомпьютерных комплексов неразрывно связано с ростом числа вычислительных узлов: вычислительные системы из первой десятки списка Top500 [1] суперкомпьютеров включают от сотен тысяч до миллионов ядер. При этом количество прямых связей между процессорами остается относительно постоянным, до нескольких десятков. Действительно, при росте количества процессоров практически невозможно обеспечить полносвязность систем: количество контактов каждого процессора возрастает в таком случае линейно, а общее количество связей в системе – квадратично [2].

Время выполнения параллельных приложений в том числе зависит от времени обмена данными между процессами. При росте количества процессоров системы увеличивается диаметр системы – максимальное расстояние между любой парой процессоров, что может привести к росту времени коммуникации.

Различают два вида пересылки сообщений: операции все-ко-всем и операции точка-точка. Очевидно, что операции все-ко-всем плохо масштабируются с ростом диаметра системы, а затраты времени на выполнение операций точка-точка можно оптимизировать, например, путем расположения наиболее коммуницирующих процессов на наиболее близких вычислительных узлах. К счастью, в большинстве практически используемых параллельных приложений преобладают операции точка-точка, при этом каждый процесс обменивается данными с относительно небольшим количеством других процессов [3]. Также исследователями рассматриваются механизмы замены операций все-ко-всем операциями точка-точка [4].

Предметом данной статьи является задача размещения наиболее коммуницирующих процессов на как можно более близких узлах вычислительной системы (задача мэппинга). Поиск оптимального размещения (мэппинга), вообще говоря, является NP-сложной задачей [2].

Большинство исследований в этой области сосредоточены на поиске эвристических алгоритмов поиска лучшего размещения. Например, в [5] достигается увеличение производительности параллельного приложения рекурсивного разбиения на архитектуре с топологией трехмерного тора на 25% относительно стандартного мэппинга, предлагаемого системой, путем реализации различных жадных эвристик.

Основными используемыми методами решения задачи мэппинга являются методы, основанные на бисекции графов [5], жадные алгоритмы [5], включая применение жадных

эвристик схожих задач (в частности, различные модификации поиска локального «оптимума», предложенные, например, в [6]), использование жадных эвристик встраивания графов и подграфов [3]. Во многих статьях используется простейший метод попарных перестановок (например, [7]). При этом большинство работ в данной области сконцентрированы на алгоритмах, заранее осведомленных о топологии конечной архитектуры и принимающих на каждой итерации построения решения, зависящие от топологии. Широкий обзор исследованных методов можно найти в [10].

В данной статье предлагается новый подход к построению мэппинга: предлагается строить мэппинг исходя из предварительного анализа структуры коммуникационных взаимодействий параллельного приложения на основе его коммуникационной матрицы. Важно, что результат такого анализа может быть использован в различных алгоритмах непосредственного мэппинга. Более детально рекомендации по использованию результатов анализа приведены в разделах 3, 7 и 8 настоящей статьи.

В статье описана постановка исходной задачи мэппинга (раздел 2), показана важность и актуальность анализа структуры коммуникационных взаимодействий (раздел 3) для решения такой задачи, предложен метод анализа структуры коммуникационных взаимодействий (раздел 4), предложена тестовая реализация метода и продемонстрированы результаты работы метода (разделы 5-6), затем предложено практическое применение метода для решения задачи мэппинга (раздел 7). Краткое резюме результатов и рекомендации к дальнейшим исследованиям приведены в разделах 8 и 9.

2. Постановка задачи мэппинга

Рассмотрим параллельное приложение, запускающее некоторое количество процессов в ходе своего выполнения $S=\{s_1, s_2, \dots s_{N_s}\}$. Допустим, для выполнения этого приложения выделено N_q однотипных процессоров вычислительной системы $Q=\{q_1, q_2, \dots q_{N_q}\}$. Будем предполагать, что каждый процесс должен запускаться на отдельном процессоре, то есть должно быть задано однозначное соответствие $q_i = q(s_i)$.

В процессе выполнения приложения каждый процесс s_i может обмениваться информацией с другими процессами. При этом происходит пересылка сообщений некоторой длины $f_{i,j} = f(s_i, s_j)$. Будем предполагать, что $f_{i,j}$ - это общее количество пересылаемой информации, как от процесса s_i в s_j, так и наоборот, от s_j к s_i. Симметричная матрица коммуникаций $F=\{f_{i,j}\}$ определяется параллельным приложением и не зависит от размещения процессов по процессорам. Матрица $F=\{f_{i,j}\}$ может быть соотнесена с произвольным периодом времени, а не обязательно с полным временем выполнения параллельного приложения. В этом случае матрица $F=\{f_{i,j}\}$ представляет общее количество информации, пересылаемое за некоторое время.

Пересылка любого пакета информации задействует различные ресурсы вычислительной системы, что приводит к понятию стоимости пересылки. Будем предполагать, что стоимость пересылки единицы информации $E=\{e_{i,j}\}$ определяется лишь взаимным расположением процессоров. Она не зависит от самих процессов, от решаемой в вычислительной системе задачи и числа запущенных ею процессов, а только от архитектуры вычислительной системы. В простейшем случае, для однородных систем, стоимость пересылки может быть выбрана как расстояние между процессорами в какой-либо метрике. В идеале, время, затраченное на пересылку сообщения между s_i и s_j должно быть пропорционально длине сообщения и стоимости пересылки $e_{i,j}$. Общая стоимость коммуникаций каждого процесса s_i для размещения $q_i = q(s_i)$ равна $E_i = \sum_{j, i \neq j} e_{q(s_i)q(s_j)} f_{i,j}$.

Задача мэппинга состоит в выборе такого размещения $q_i = q(s_i)$, чтобы общая стоимость всех пересылок информации при выполнении задачи была минимальной.

$$E = \left(\sum_{i=1,\dots N_s} E_i \right) \xrightarrow[q(S)]{} \min .$$

Функцию E в дальнейшем будем называть функцией качества размещения (мэппинга).

3. Построение мэппинга на основе анализа структуры коммуникационных взаимодействий

Интуитивно, основной идеей построения оптимального мэппинга для коммуникаций точка-точка является расположение наиболее коммуницирующих между собой процессов на наиболее близких друг к другу процессорах вычислительной системы. Различными исследователями проведены эксперименты [5], доказывающие результативность применения данной идеи, в частности, простых жадных алгоритмов, на ней основанных. Под результативностью при этом понимается практически важный результат, а именно снижение времени выполнения параллельных приложений при изменении мэппинга со стандартного на найденный. При этом рассмотренные ранее алгоритмы и эвристики по своему построению предполагают использование значительно урезанной информации, содержащейся в коммуникационной матрице, и использование знаний о топологии на каждом итерационном шаге.

Например, метод рекурсивной бисекции графов при каждом следующем разбиении не использует информацию о межкластерной коммуникации, отброшенной на прошлом шаге, т.е. в общем случае на каждой итерации отбрасывается половина исходной информации. Научным сообществом уже была отмечена возможность получения качественно плохих результатов таким алгоритмом [9]. Жадные алгоритмы, предложенные в [3, 5] (заметим, что аналогичные подходы являются распространенными и используется во многих других работах), рассматривают всех соседей уже размещенных процессов для выбора следующего. Коммуникация между соседями при этом используется исключительно для целей приоритезации выбора следующего. И тот, и другой алгоритмы, очевидно, не учитывают общую структуру взаимодействий. В случае блочной-диагональной матрицы коммуникаций (для точности будем считать, что размер блока не является степенью 2) оба вышеуказанных метода не дадут понимания о такой блочности и лишь частично учтут эту структурную особенность в построенном мэппинге.

В 2009 г. исследователями из Berkley было проведены исследования по поиску и классификации наиболее часто встречающихся классов параллельных приложений [8]. Среди наиболее часто встречающихся коммуникационных структур, в частности, отмечены классы приложений Structured Grids и FFT. Коммуникационные матрицы данных классов приложений имеют блочно-диагональный вид. Более того, для этих классов задач такая структура взаимодействий сохраняется при увеличении количества процессов, задействованных при выполнении приложения.

Таким образом, приобретает актуальность разработка метода анализа структуры взаимодействий параллельных приложений. Понимание структуры взаимодействий параллельных приложений может обеспечить построение более результативного мэппинга для параллельных приложений. Дополнительно, при априорных знаниях о сохранении структуры коммуникационных взаимодействий при увеличении числа задействованных процессов открывается возможность построения хорошо масштабируемых алгоритмов мэппинга. Это особенно важно с учетом отмеченного ранее [3] ухудшения качества работы жадных алгоритмов при увеличении размерности задачи.

Подводя итог описанному выше, предлагается произвести декомпозицию исходной задачи мэппинга на две задачи:

- Задача 1: анализ структуры взаимодействий (не зависит от топологии вычислительной системы)
- Задача 2: построение мэппинга (зависит только от топологии и ответа Задачи 1, не зависит от приложения как такового).

Важно отметить, что в Задаче 1 предполагается анализ общей структуры взаимодействий, а не только возможных разбиений процессов на множества, при котором, как показано выше, может наблюдаться потеря информации, критически важной для построения эффективного мэппинга.

Дополнительно, такой подход потенциально позволит построить мэппинги параллельного приложения на различные топологии на основе однократного запуска метода анализа структуры коммуникационных взаимодействий.

Ниже предложен метод анализа структуры коммуникационных взаимодействий путем симуляции гравитационных взаимодействий.

4. Гравитационный метод анализа коммуникационных взаимодействий

4.1 Описание метода

Каждый процесс представляется точкой на плоскости. Для каждой точки-процесса задаются законы движения, способствующие сближению более коммуницирующих между собой процессов друг с другом. Если предполагать, что при построении мэппинга соседство точек-процессов на плоскости будет приводить к размещению процессов в близких процессорах вычислительной системы, такие законы движения интуитивно должны обеспечивать минимизацию функции качества решения задачи размещения (мэппинга).

Зададим два типа взаимодействий между точками-процессами:

- Притяжение точек-процессов друг к другу. Сила притяжения действует на любом расстоянии и пропорциональна интенсивности коммуникации $f_{i,j}$.
- Отталкивание точек-процессов друг от друга. Сила отталкивания тем выше, чем меньше расстояние между точками-процессами. Для больших расстояний сила отталкивания незначительна, либо вообще не действует.

Дадим формальное описание предлагаемого метода.

Если два процесса i и j коммуницируют друг с другом, т.е. $f_{i,j} > 0$, то будем предполагать, что они притягиваются друг к другу с силой, пропорциональной интенсивности коммуникации.

$\left|\overrightarrow{F_{i,j}^{att}}\right| = c_{att} * f_{i,j}$, где c_{att} – коэффициент притяжения.

Эта сила действует на любом расстоянии. c_{att} является постоянным коэффициентом, одинаковым для всех пар процессов.

Силы отталкивания, напротив, действуют на некотором ограниченном расстоянии.

$\left|\overrightarrow{F_{i,j}^{p}}\right| \xrightarrow{dist(i,j)\gg 1} 0.$

Но сила отталкивания увеличивается неограниченно по мере сближения двух точек.

$\left|\overrightarrow{F_{i,j}^{p}}\right| \xrightarrow{dist(i,j)\to 0} \infty$

Конкретная форма функциональной зависимости здесь не так важна. Например, она может быть выбрана обратной квадрату расстояния между точками-процессами на плоскости. Важно только ее стремление к бесконечности, когда точки сильно сближаются друг с другом.

Динамика движения множества точек-процессов задается в общем случае следующим уравнением:

$$\vec{r}_i(t+h) = \vec{r}_i(t) + h\sum_j \left(\overrightarrow{F_{i,j}^{att}} + \overrightarrow{F_{i,j}^{p}}\right)$$

Здесь $\vec{r}_i(t)$ является вектором положения точки-процесса на плоскости в момент времени t.

Начальные координаты точек-процессов предлагается выбирать случайным образом, либо, если известна какая-либо априорная информация, экспертным путем.

Иногда для наглядности эволюции положения множества точек-процессов можно добавить силы, которые препятствуют дрейфу множества от выбранного центра. Например, при большом расстоянии до центра добавить малую силу, направленную в центр. Но использование подобных сил не является обязательным.

4.2 Итерационный алгоритм симуляции

Итерационный алгоритм симуляции движения точек-процессов представлен ниже.

1. Выбрать координаты $\vec{r_s}$ каждой точки-процесса s случайным образом.
2. **Пока не выполнен критерий останова, для каждой итерации** i:
   a. Для каждой точки-процесса s рассчитать результирующую сил притяжения $\overrightarrow{F_s^{atr}}$ и отталкивания $\overrightarrow{F_s^p}$, действующих на нее в результате взаимодействия с другими точками-процессами
   b. Изменить координаты по направлению результирующей силы

$$\vec{r_s}(i+1) = \vec{r_s}(i) + \overrightarrow{F_s^{atr}} + \overrightarrow{F_s^p}$$

3. **Конец цикла**
4. **Результат ← координаты положений точек-процессов** $\{\vec{r}_s\}$ **(конфигурация точек-процессов).**

5. Реализация метода

Тестовые реализации предлагаемого метода произведены в Visual Studio 2010 и системе Matlab с использованием матричных операций для каждой итерации. Вычислительные эксперименты проводились для общего числа процессов до 2048 с эмпирически выбранным числом итераций.

Также реализован критерий останова, основанный на анализе среднего положения процессов за определенное количество итераций, а именно:

- Вычислить для каждого процесса $S_n(\vec{r_s})$ и $S_m(\vec{r_s})$ среднее положение процесса s за n и m последних положений.
- Если $\sum_s |S_n(\vec{r_s}) - S_m(\vec{r_s})| < \varepsilon$, то работа алгоритма прекращается.

6. Результаты применения метода к модельной задаче

Проиллюстрируем работу предложенного метода для выделения коммуницирующих групп из множества точек-процессов.

Рассмотрим следующий простой модельный пример. Пусть запущено 20 процессов. Первые десять процессов коммуницируют друг с другом и не коммуницируют с другими процессами. Аналогично последние десять процессов коммуницируют между собой и не коммуницируют с первыми десятью. Интенсивности всех коммуникаций предполагаются равными между собой.

Нетрудно заметить, что процессы разделяются на две группы коммутации. Продемонстрируем работу алгоритма на этой примере и разделим процессы на две группы.

6.1 Иллюстрация работы метода

В качестве начального расположения выберем расположение точек-процессов на окружности с центром в начале координат. Точки отстоят друг от друга на постоянный угол, как показано на рис. 1. Зеленым отрезком показывается направление движения точки. Исходя из проведенных экспериментов, такое начальное расположение является самым «неудачным» и ведет к наиболее долгой сходимости метода.

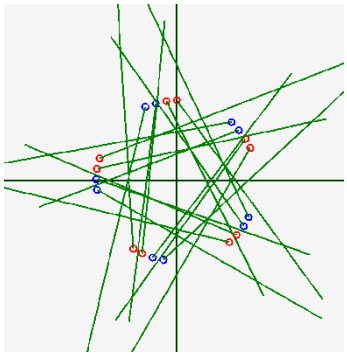
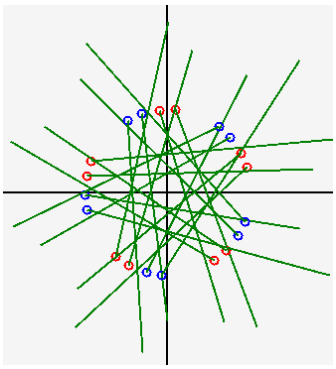
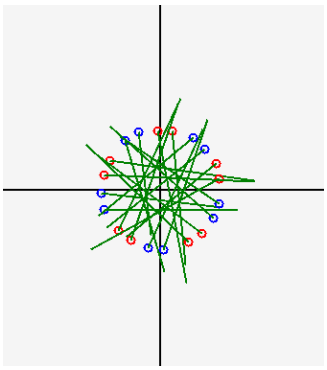

Рис. 1. Начальное положение **Рис. 2.** Движение к центру **Рис. 3.** Локальное равновесие

На рисунках 1-3 показаны начальные итерации метода. Поскольку элементы матрицы коммуникаций неотрицательны, то точки-процессы притягиваются друг к другу и приближаются к центру насколько позволяют силы взаимного отталкивания. Скорости движения сначала достаточно большие, но по мере приближения к центру они снижаются. Ситуация близка к локальному равновесию, когда точкам больше не удается приблизиться к центру из-за сил отталкивания, как показано на рис. 3.

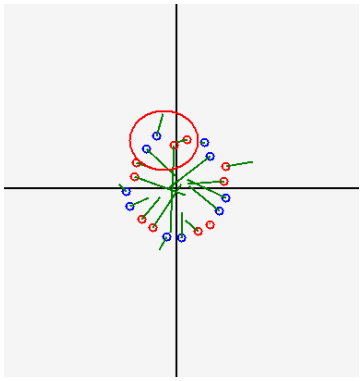
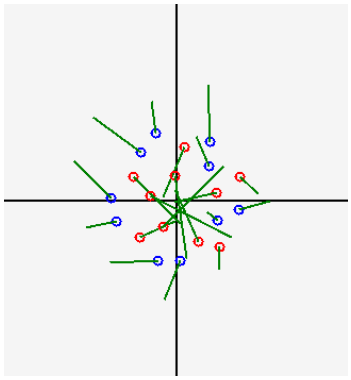
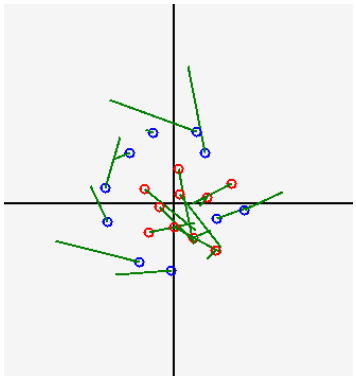

Рис. 4. Начало расхождений **Рис. 5.** Начало перестроений **Рис. 6.** Начало «кластеризации»

На рис. 4-6 показан процесс перестроения точек-процессов после выхода одной из точек-процессов из круга. Происходит изменение конфигурации множества точек-процессов. Скорости увеличиваются. На рис. 5. первоначальный порядок преобразуется к конфигурации, отражающей структуру взаимодействия. А на рис. 6 выделяется «красный» кластер.

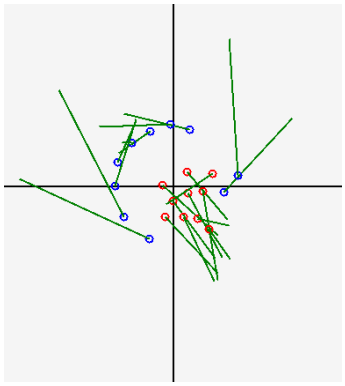
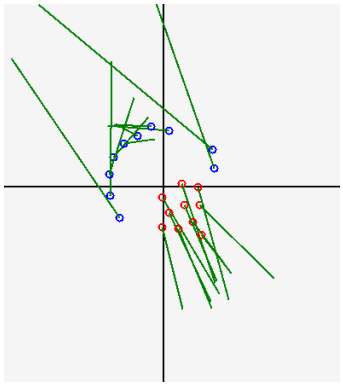
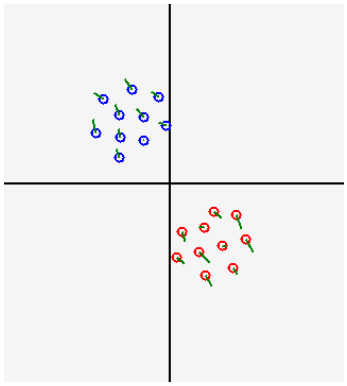

Рис. 7. Формирование первой группы **Рис. 8.** Расхождение групп **Рис. 9.** Окончательная конфигурация

На рис. 7-9 показано окончание работы метода.

На рис. 7 «красные» точки-процессы образовали единую группу в центре и начали дрейфовать вниз вправо. Синие начала тяготеть к периферии. На рис. 8 под действием сил отталкивания две группы начали удаляться друг от друга. Окончательное положение на рис. 9

характеризуется почти полным обнулением скоростей и ясно выраженными кластерами коммуницирующих процессов.

6.2 Зависимость результата от начального положения точек-процессов

Метод является градиентным, и поэтому результат работы алгоритма и его сходимость к локальному оптимуму, вообще говоря, зависит от начального положения точек. Тем не менее, структура получающихся предложенным методом расположений точек-процессов на плоскости практически не зависела от начального положения точек-процессов в проведенных вычислительных экспериментах.

7. Практическое применение метода

Полученное в результате работы алгоритма расположение точек-процессов показывает внутреннюю структуру их коммуникаций. Наиболее коммуницирующие точки-процессы оказываются поблизости.

После нахождения структуры расположения множества точек-процессов на плоскости, найденные закономерности, группы и взаимное расположение точек-процессов можно использовать для построения эффективных алгоритмов мэппинга. Например, использовать евклидово расстояние между точками-процессами как меру близости процессов в жадных алгоритмах (в частности, для жадных алгоритмов, описанных в [5]). Полученные группы точек-процессов могут использоваться для сжатия мощности множества процессов до мощности множества вычислительных узлов вместо алгоритмов бисекции графов (метод бисекции описан в [3,7]). Наглядность метода позволяет исследователю использовать результаты при построении эвристик для конкретного типа коммуникационных взаимодействий и выбирать мэппинг экспертно на основе полученной дополнительной информации о структуре коммуникационных взаимодействий. Метод можно также рассматривать как аналог кластеризации.

Важно отметить, что в используемом методе никак не используется информация о топологии целевой вычислительной системы. В результате работы метода происходит поиск расположения точек-процессов на плоскости, отвечающего структуре коммуникационных взаимодействий исходного параллельного приложения. При этом происходит выделение групп наиболее коммуницирующих между собой точек-процессов в самом общем виде, выявление закономерностей не сразу очевидных из вида коммуникационной матрицы.

Как показано в разделе 3 настоящей статьи, предлагаемый метод является актуальным для распространенного класса параллельных приложений Structured Grids и FFT, может использоваться для построения масштабируемых алгоритмов мэппинга, а также в решении задачи мэппинга методом декомпозиции, таким образом, исключая необходимость повторного запуска при построении мэппинга для различных топологий вычислительных систем.

8. Заключение

Развитие современных вычислительных систем подразумевает рост числа узлов при сохранении числа связей на относительно небольшом уровне, в результате чего возрастает диаметр системы, что может привести к увеличению времени коммуникации между процессами параллельных приложений.

В этой связи актуальной становится задача мэппинга – размещения процессов параллельных программ на процессоры таким образом, чтобы снизить затраты на коммуникацию.

Доказано, что задача мэппинга является NP-полной. Исследования в этой области сосредоточены на поиске эвристических методов поиска наиболее оптимального мэппинга, в большинстве своем игнорирующие общую структуру взаимодействий параллельных приложений.

В разделе 3 показана актуальность разработки метода, позволяющего выявить структуру коммуникационных взаимодействий параллельных приложений, для дальнейшего развития

масштабируемых алгоритмов мэппинга, а также предложен подход к решению задачи мэппинга на основе решений следующих двух подзадач: анализа структуры взаимодействия (не зависит от топологии целевой вычислительной системы) и построения мэпппинга на основе полученных знаний о структуре (зависит только от топологии).

В данной статье впервые предлагается метод анализа структуры коммуникационных взаимодействий параллельных приложений. Анализ производится на основе их коммуникационных матриц.

Практическая важность подхода обуславливается тем, что найденные конфигурации точек-процессов описывают структуру коммуникационных взаимодействий и могут быть использованы, в частности, следующим образом:

- для определения близости процессов в жадных алгоритмах;
- для разбиения множества процессов на кластеры на основе полученной структуры: использование для бисекции, разбиения на p подграфов, построение алгоритмов мэппинга на основе кластеризации;
- для экспертного построения эвристик исходя из полученных знаний о структуре взаимодействий.

К дополнительным плюсам метода можно отнести:

- наглядность метода: исследователь может подбирать эвристики или выбирать из доступных методов исходя из понимания структуры взаимодействий;
- отсутствие привязки к топологии вычислительной системы, что обеспечивает возможность использования метода в двухфазном подходе к решению задачи мэппинга.

9. Направление дальнейших исследований

С нашей точки зрения, наиболее приоритетными направлениями дальнейших исследований являются:

- анализ и исследование посредством гравитационного метода структур взаимодействий процессов для различных классов приложений;
- построение новых эффективных алгоритмов мэппинга для различных типов архитектур, на основе получаемых предложенным гравитационным методом конфигураций точек-процессов;
- исследование эффективности модификаций существующих алгоритмов, построенных с использованием получаемых предложенным гравитационным методом конфигураций точек-процессов, для конкретных архитектур и приложений.

Работа выполнена при поддержке грантов РФФИ 14-07-00628 и 14-07-00654.

Литература

1. Top-500: The List. URL: http://www.top500.org/ (accessed: 14.07.2016).
2. Shahid H. Bokhari. On the Mapping Problem // IEEE Transactions on computers, vol. c-30, No. 3, 1981.
3. Toersten Hoefler, Marc Snir. Generic Topology Mapping Strategies for Large-scale Parallel Architectures // ICS'11, 2011.
4. D. Pekurovsky. P3DFFT – Highly scalable parallel 3D Fast Fourier Transforms library // Technical report, 2010.
5. C.D. Sudheer, A. Srinivasan. Optimization of the Hop-Byte Metric for Effective Topology Aware Mapping // IEEE, 2012.
6. Yong Li, Panos M. Pardalos, Mauricio G.C. Resende. A Greedy Randomized Adaptive Search Procedure for the Quadratic Assignment Problem // DIMACS Series in Discrete Mathematics and Theoretical Computer Science, 1991.

7. Ponnuswamy Sadayappan, Fikret Ercal. Cluster-Partitioning Approaches to Mapping Parallel Programs onto a Hybercube // Conference: Supercomputing, 1st International Conference, Athens, Greece, 1987.
8. Krste Asanovic, Rastislav Bodik, James Demmel, Tony Keav eny, Kurt Keutzer, John Kubiatowicz, Nelson Morgan, David Patterson, Koushik Sen, John Waw rzynek, David Wessel, Katherine Yelick. A view of Parallel Computing Landscape // Communications of the ACM, vol. 52, No. 10, 2009, p.56 – 67.
9. H.D. Simon and S.-H. Teng. How good is recursive bisection? // SIAM J. Sci. Comput., 18:1436-1445, 1997.
10. Torsten Hoefler, Emmanuel Jeannot, Guillaume Mercier. An Overview of Process Mapping Techniques and Algorithms in High-Performance Computing. Emmanuel Jeannot and Julius Zilinskas. High Performance Computing on Complex Environments, Wiley, pp.75-94, 2014.

Analysis of processes communication structure for better mapping of parallel applications

K. Bobrik[1], N. Popova[1]

Moscow State University[1]

We consider a new approach for the classical problem of parallel applications' processes placement to nodes of the multiprocessor computing system. The first time an algorithm which analyzes parallel processes communication structure is presented. Found communication structure can be used to recommend mapping for the multiprocessor computing system with given topology. An input for the proposed approach is data representing total size of messages sent between each two processes. A set of processes is analyzed as a system of points which evolve under the influence of gravity forces. The found distribution of the points reflects communication patterns of the underlying application and can be used for effective mapping heuristics.

Keywords: mapping, supercomputers, parallel applications

References

1. Top-500: The List. URL: http://www.top500.org/ (accessed: 14.07.2016).
2. Shahid H. Bokhari. On the Mapping Problem // IEEE Transactions on computers, vol. c-30, No. 3, 1981.
3. Toersten Hoefler, Marc Snir. Generic Topology Mapping Strategies for Large-scale Parallel Architectures // ICS'11, 2011.
4. D. Pekurovsky. P3DFFT – Highly scalable parallel 3D Fast Fourier Transforms library // Technical report, 2010.
5. C.D. Sudheer, A. Srinivasan. Optimization of the Hop-Byte Metric for Effective Topology Aware Mapping // IEEE, 2012.
6. Yong Li, Panos M. Pardalos, Mauricio G.C. Resende. A Greedy Randomized Adaptive Search Procedure for the Quadratic Assignment Problem // DIMACS Series in Discrete Mathematics and Theoretical Computer Science, 1991.
7. Ponnuswamy Sadayappan, Fikret Ercal. Cluster-Partitioning Approaches to Mapping Parallel Programs onto a Hybercube // Conference: Supercomputing, 1st International Conference, Athens, Greece, 1987.
8. Krste Asanovic, Rastislav Bodik, James Demmel, Tony Keav eny, Kurt Keutzer, John Kubiatowicz, Nelson Morgan, David Patterson, Koushik Sen, John Waw rzynek, David Wessel, Katherine Yelick. A view of Parallel Computing Landscape // Communications of the ACM, vol. 52, No. 10, 2009, p.56 – 67.
9. H.D. Simon and S.-H. Teng. How good is recursive bisection? // SIAM J. Sci. Comput., 18:1436-1445, 1997.
10. Torsten Hoefler, Emmanuel Jeannot, Guillaume Mercier. An Overview of Process Mapping Techniques and Algorithms in High-Performance Computing. Emmanuel Jeannot and Julius Zilinskas. High Performance Computing on Complex Environments, Wiley, pp.75-94, 2014.