

ОПТИМИЗАЦИЯ ФРАГМЕНТАЦИИ ПРИ ВЫДЕЛЕНИИ РЕСУРСОВ ДЛЯ ВЫСОКОПРОИЗВОДИТЕЛЬНЫХ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ С СЕТЬЮ АНГАРА*

© 2018 А.В. Мукосей, А.С. Семенов

АО «НИЦЭВТ»

(117587 Москва, Варшавское шоссе 125с15)

E-mail: mukosey@nicevt.ru, semenov@nicevt.ru

Поступила в редакцию: 04.06.2018

В данной работе рассматривается высокоскоростная вычислительная сеть с топологией многомерный тор. Работа посвящена оптимизации фрагментации, возникающей в результате последовательного выделения вычислительных узлов в многоузловой системе при заданном требовании о том, что сетевой трафик разных пользовательских заданий не должен пересекаться. В данной работе на основе идей из задачи о многомерной упаковке контейнера предложен метод поиска узлов с оценкой фрагментированности системы. Для такой оценки введено понятие прямоугольников максимального размера, которые возможно вписать в систему после размещения очередного пользовательского задания. Каждое множество узлов, подходящее для размещения задания, оценивается предложенной функцией, учитывающей размер и количество найденных прямоугольников максимального размера. Исследование разработанного метода проводилось с помощью симулятора работы вычислительной системы. Рассмотрен набор различных вычислительных систем с трехмерными и четырехмерными топологиями, размер минимальной системы — 32 вычислительных узла, максимальной — 144 узла. Для каждой системы задана синтетическая очередь заданий, параметры которой приближены к реально возможной. В качестве критерия качества метода выбора узлов рассматривается средняя утилизация ресурсов вычислительной системы и среднее время ожидания заданий в очереди. Исследование показало, что увеличение утилизации ресурсов для предложенного метода выбора узлов составило в среднем 11% по сравнению с базовым методом, а среднее значение времени нахождения задания в очереди сокращено на 45,3%.

Ключевые слова: Коммуникационная сеть Ангара, многомерный тор, правило порядка направлений, фрагментация вычислительной системы, выбор узлов

ОБРАЗЕЦ ЦИТИРОВАНИЯ

Мукосей А.В., Семенов А.С. Оптимизация фрагментации при выделении ресурсов для высокопроизводительных вычислительных систем с сетью Ангара // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2018. Т. X, № Y. С. Z1–Z2. DOI: 10.14529/cmseXXXXXX.

Введение

Фрагментация, возникающая в результате последовательного выделения вычислительных узлов в многоузловой системе играет критическую роль в эффективности использования суперкомпьютера. Особенное значение эта проблема имеет для сетей с топологией типа тор. На данный момент существует две стратегии выделения ресурсов, используемых в суперкомпьютерах с тороидальной сетью [1]. В суперкомпьютерах IBM Blue Gene/P, Blue Gene/Q предоставление ресурсов основано на разделении системы на партии, по которым размещаются задачи пользователей [2, 3]. Такая стратегия может порождать фрагмента-

*Статья рекомендована к публикации программным комитетом Международной научной конференции «Параллельные вычислительные технологии (ПаВТ) 2018»

цию, возникающую в результате выделения большего числа узлов, чем требовалось или невозможности выделить доступный набор узлов из разных партий.

Вторая стратегия используется в серии суперкомпьютеров Cray XT/XE, где расположение выделенных узлов [4] не зависит от топологии. Такой способ выделения ресурсов может привести к деградации производительности ввиду наличия конкурирующего трафика. В статье [1] алгоритм оптимизации упаковки заданий использует линейную нумерацию узлов, однако это неудобно для сети Ангара.

В АО «НИЦЭВТ» разработана высокоскоростная коммуникационная сеть Ангара [5, 6] с топологией «многомерный тор». В маршрутизаторе сети реализована бездедлоковая, адаптивная маршрутизация, основанная на правилах «пузырька» (Bubble flow control, [7]) и «порядка направлений» (Direction ordered routing, DOR, [8, 9]) с использованием битов направлений [9]. Благодаря алгоритму First Step/Last Step «нестандартного первого и последнего шага» [9] аппаратно поддерживается обход отказавших узлов или линков. Эффективность этого метода по поддержанию связности в сети с отказами была показана в статье [10].

Для сети Ангара необходимо разработать алгоритм оптимизации фрагментации при условии отсутствия пересечения трафика различных задач.

Такая задача носит название упаковки контейнера и является NP -полной задачей. Существуют различные способы решения такой задачи: авторы статьи [11] предлагают решать такую задачу муравьиным алгоритмом — полиномиальный алгоритм метаввристической оптимизации для нахождения приближённых решений, который основан на использовании модели поведения муравьёв, ищущих пути от колонии к источнику питания. Авторы статьи [12] разработали различные алгоритмы упаковки с использованием генетического алгоритма.

В данной статье предлагается алгоритм выбора узлов для суперкомпьютеров с топологией «многомерный тор» с учетом расположения заданий в топологии в коммуникационной сети. Статья организована следующим образом. В разделе 1 приводятся необходимые формальные определения и постановка задачи. В разделе 2 описаны разработанные алгоритмы. В разделе 3 проведено исследование построенных алгоритмов.

В результате, разработанный алгоритм на рассмотренных системах и предложенных очередях в среднем увеличил утилизацию ресурсов на 11%, а значение времени нахождения заданий в очереди сократил на 57% по сравнению с первоначальным вариантом алгоритма, использовавшимся для выбора узлов.

1. Определения и постановка задачи

В данном разделе приводятся формальные определения, которые в дальнейшем будут использоваться в статье.

Рассмотрим вычислительную систему, узлы которой объединены в тороидальную топологию. Размерности тора обозначим (d_1, d_2, \dots, d_n) , а множество всех узлов вычислительной системы обозначим $N = \{u | u = (u_1, \dots, u_n), \forall i u_i \in \mathbb{Z}_{d_i}\}$, а общее число узлов — $|N|$. Расстояние на множестве N определим следующим образом: $L(u, v) = \sum_{i=1}^n |u_i - v_i|, \forall u, v \in N$.

Состояние системы S можно описать множествами узлов, доступных и недоступных для выделения, обозначим эти множества N_{free} и N_{locked} , соответственно.

Будем называть *маршрутизируемым* множеством узлов в коммуникационной сети Ангара такое множество, что для каждого узла этого множества существует сетевой маршрут

в любой другой узел множества, удовлетворяющий правилам маршрутизации сети Ангара, а также весь сетевой трафик узлов множества не выходит за его пределы.

Будем называть *заданием* W — число узлов W_{nodes} , запрашиваемое пользователем в момент времени W_{start} на время W_{time} , а *ресурсами для задания* — маршрутизируемое множество узлов, размер которого не меньше, чем W_{nodes} . *Потоком заданий* назовем множество различных заданий W .

Ранее в работе [13] авторами статьи решалась проблема поиска маршрутизируемого множества заданного размера в коммуникационной сети Ангара с учетом топологии и маршрутизации. Обозначим алгоритм, решающий эту проблему как $Find_Systems(W, S)$. На вход этому алгоритму подается состояние системы S и задание W с требуемым числом вычислительных узлов W_{nodes} . Результатом работы алгоритма является набор вариантов ресурсов для задания. Необходимо заметить, что особенностью алгоритма является то, что все ресурсы для задания представляют собой многомерные прямоугольники.

Под *утилизацией* ресурсов U вычислительного кластера будем понимать среднее значение утилизации по всем вычислительным узлам:

$$U = \frac{\sum_{i=1}^{|N|} U_i}{|N|}, U_i = \frac{T_i}{T},$$

где U_i — утилизация i -го вычислительного узла, T — время работы вычислительного кластера, T_i — полезное время работы i -го вычислительного узла.

Обозначим *значение времени нахождения задания в очереди относительно запрошенного времени* как $T_{delay}^i = \frac{Q^i}{W_{time}^i}$, где W_{time}^i — запрошенное время для задания W^i , Q^i — время ожидания задания W^i в очереди. За *среднее значение времени нахождения задания в очереди относительно запрошенного времени задания* примем $T_{mid} = \frac{\sum_{i=1}^k T_{delay}^i}{k} = \frac{1}{k} \sum_{i=1}^k \frac{Q^i}{W_{time}^i}$, где k — число различных заданий в потоке.

За оценку качества решения для потока пользовательских заданий возьмем утилизацию ресурсов вычислительного кластера и среднее значение времени нахождения задания в очереди относительно запрошенного времени. Эти характеристики используются по аналогии с работой [15].

Во введенных обозначениях проблема, которую решает данная статья, будет формулироваться следующим образом. Для заданного вычислительного кластера и последовательности заданий $Q = W^1, \dots, W^k$ требуется найти ресурсы для всех заданий из последовательности, которые будут максимизировать утилизацию вычислительного кластера и минимизировать среднее значение времени нахождения задания в очереди относительно запрошенного времени.

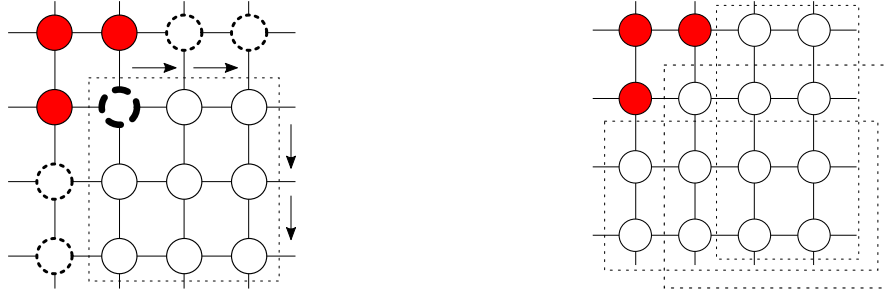
2. Алгоритм выбора узлов

Для решения поставленной проблемы в статье предложен алгоритм выбора узлов, основанный на методах, предложенных в работе [12], посвященной трехмерной упаковке контейнера. Идея алгоритма выбора узлов заключается в расположении задания таким образом, чтобы максимизировать оставшееся пространство в многомерном торе.

2.1. Алгоритм построения прямоугольников максимального размера

Назовем *прямоугольником максимально возможного размера MSS* (*MaxSpaceSize*) многомерный прямоугольник, состоящий только из узлов N_{free} , который нельзя расширить ни в одну из его сторон. Расширить прямоугольник может быть невозможно по двум причинам — либо по соответствующему измерению тора достигнуто максимальное количество узлов в кольце (расширять некуда), либо сторона прямоугольника граничит с узлом из множества N_{locked} . Множество различных прямоугольников *MSS* характеризуют меру фрагментированности системы.

Алгоритм поиска различных прямоугольников *MSS* (*Find_MSSs(S)*) реализован следующим образом. Из множества N_{free} выбирается узел $u_1 \in N_{free}$. Выбранный узел последовательно расширяется во все стороны, пока это возможно. Полученное множество узлов обозначим MSS_1 . На следующем этапе выбирается узел $u_2 \in N_{free} \setminus MSS_1$ и аналогичным образом строится множество MSS_2 . Алгоритм продолжается до тех пор, пока множество $N_{free} \setminus \bigcup_{iter=1}^{Iters} MSS_{iter}$ не пусто, где *Iters* — число итераций алгоритма. Псевдокод алгоритма представлен на рис. 2. Обозначим множество различных MSS_{iter} , как *MSSs*. Важно отметить, что каждый прямоугольник строится независимо от остальных прямоугольников, в предположении доступности всех изначально свободных узлов N_{free} .



(а) Построение прямоугольника максимально возможного размера (б) Все прямоугольники максимально возможного размера, построенные алгоритмом

Рис. 1. Выделение прямоугольников максимального размера

Иллюстрация работы алгоритма приведена на рис. 1а и рис. 1б, на которых в двумерной решетке узлы множества N_{locked} закрашены, а свободные узлы N_{free} — нет. Жирным контуром на рис. 1а выделен узел, из которого поочередным расширением построен двумерный прямоугольник, который обозначен пунктирной линией. Узлы, выделенные полужирным пунктиром, соответствуют множеству узлов N_{free} , не входящих в построенный прямоугольник. Из этих узлов будут строиться последующие прямоугольники. Все построенные прямоугольники *MSS* показаны на рис. 1б.

2.2. Оценка состояния вычислительной системы на основе прямоугольников максимального размера

Для оценки состояния вычислительной системы предложена функция φ , которая тем больше, чем большее число прямоугольников максимального размера имеется в системе:

$$\varphi(S) = N * MSS_{max}^{nodes} + |MSS_{max}|,$$

```

Input:
S -- массив, характеризующий состояние системы
S[u], может принимать следующие значения: 0 - free, 1 - locked, 2 - discovered
Output:
MSSs -- массив прямоугольников максимального размера
Find_MSS(S)
{
    Iter = 1
    dirs -- массив доступных направлений в торе, например, +x,-x,+y,-y...
    MSSs -- результирующий массив, изначально пуст
    for u in S {
        if S[u] == free {
            MSSs[Iter].push_back(u)
            for dir in dirs {
                MSSs[Iter].extend(dir)
            }
            for v in MSSs[Iter] {
                S[v] = discovered
            }
            Iter++
        }
    }
    return MSSs
}

```

Рис. 2. Псевдокод алгоритма Find_MSS поиска прямоугольников максимального размера

где MSS_{max} — множество прямоугольников максимального размера, имеющих наибольшее число узлов MSS_{max}^{nodes} , $|MSS_{max}|$ — число таких прямоугольников, S — текущее состояние системы.

Эта метрика была добавлена в алгоритм $Find_Systems(W, S)$ поиска маршрутизируемого множества заданного размера. Для каждого найденного маршрутизируемого множества оценивается значение функции $\varphi(S')$, где S' — состояние вычислительной системы S после выделения узлов. Для увеличения утилизации вычислительного кластера требуется выбирать решения с наибольшим значением функции φ . Модифицированный алгоритм $Find_Systems(W, S)$, в котором возможные варианты систем отсортированы с учетом значения функции φ , в дальнейшем будем обозначать $Find_Systems_{MSS}(W, S)$

2.3. Первоначальный алгоритм выбора узлов для кластеров с сетью Ангара

Алгоритм, который изначально работал на кластерах с сетью Ангара, устроен следующим образом. Для всего кластера строится таблица маршрутизации [13]. Для требуемого числа узлов W_{nodes} и допустимого числа транзитных узлов $N_{transit}$ строятся всевозможные разложения чисел $W_{nodes}, W_{nodes} + 1, \dots, W_{nodes} + N_{transit}$ на n множителей, таких что $1 \leq p_i \leq d_i, \forall i \in 1..n$, где p_i — множитель разложения. Все такие разложения обозначим D . Эти разложения описывают всевозможные размеры прямоугольников, подходящих под решение задачи W . Средним диаметром прямоугольника, соответствующего разложению

$D_j \in D$, назовем среднее арифметическое всех расстояний между узлами прямоугольника:

$$\frac{\sum_{u,v \in D_j, u \neq v} L(u,v)}{|D_j|}.$$

Следующий этап выбора узлов — поиск множества узлов вычислительного кластера, которое можно покрыть одним из найденных прямоугольников таким образом, чтобы в покрытии присутствовали только узлы из множества N_{free} , то есть доступные для выделения. Поиск покрытия начинается с разложений с наименьшим средним диаметром. При первом найденном решении алгоритм заканчивает свою работу.

3. Экспериментальное исследование

3.1. Симулятор вычислительного кластера

Для оценки утилизации ресурсов вычислительного кластера разработан симулятор очереди задач (заданий) и модель состояния кластера. Заметим, что в данной работе не рассматривалась возможность обгона заданий, то есть предоставление ресурсов заданию, имеющему более позднее время W_{time} , если заданию с ранним временем W_{time} не были выделены ресурсы. На вход симулятору подается поток пользовательских задач $Q = W^1, \dots, W^k$. На выходе выдается полное время работы всего кластера T , время работы каждого узла T_i и время предоставления ресурсов для каждого задания. Используя эти данные, можно вычислить утилизацию ресурсов вычислительного кластера U и среднее значение времени нахождения задания в очереди T_{mid} .

Введем некоторые формальные определения, необходимые для описания работы симулятора. *Очередью* симулятора назовем набор заданий из потока, для которых не выделялись ресурсы и время их запуска T_{start} меньше текущего симулируемого времени t . *Временем занятости узла u* системы S , назовем время, на которое узел u был выделен для некоторого задания W . В начальный момент времени $t = 0$ время занятости всех узлов равно 0. Операцией *выделения набора узлов* на время T_{alloc} назовем увеличение времени занятости для этих узлов на время T_{alloc} . *Временем изменения системы T_S* назовем время, через которое освободится хотя бы один из выделенных узлов. *Временем изменения очереди T_{queue}* назовем время, через которое хотя бы одно задание перейдет из потока заданий в очередь симулятора. Тогда *временем ожидания симулятора T_{sleep}* назовем минимальное время до изменения состояния симулятора: $T_{sleep} = \min(T_S, T_{queue})$.

Алгоритм работы симулятора устроен следующим образом. Если очередь симулятора не пуста, симулятор выполняет процедуру поиска маршрутизируемого множества для каждого задания из очереди по очереди. Если удалось найти решение, то симулятор выделяет найденные ресурсы на необходимое время, а также удаляет это задание из очереди. Если решение не было найдено, время симулятора сдвигается на время ожидания T_{sleep} , а время занятости каждого занятого узла u системы S уменьшается на T_{sleep} . Если очередь заданий пуста, а все узлы перешли в состояние свободных, то симулятор завершает свою работу.

3.2. Результаты исследования

Исследование разработанного алгоритма проводилось на симуляторе для вычислительных систем [16], представленных в таблице 1.

Поток заданий для каждой из систем характеризуется вероятностью появления задания для каждого числа узлов от 1 до максимального. Распределение вероятностей таких вероятностей представлено на рис. 3. На рис. 3б представлено реальное распределение пользова-

Таблица 1. Моделируемые системы

Количество узлов вычислительной системы	Топология 3х-мерный тор	Топология 4х-мерный тор
32	4x4x2	4x2x2x2
36	4x3x3	3x3x2x2
64	4x4x4	4x4x2x2
96	6x4x4	4x4x3x2
144	8x6x3	4x4x3x3

тельских задач (заданий) по количеству узлов, полученное с гибридного суперкомпьютера Desmos на базе сети Ангара, имеющую топологию 4х-мерный тор 4x2x2x2 [18]. В дальнейшем системы с данной очередью будем пометать символом s — 4x2x2x2s и 4x4x2s. Остальные распределения долей заданий по количеству узлов — синтетические, основанные на предположении о том, что чаще всего встречаются заданий с требуемым числом узлов, равным степеням двойки.

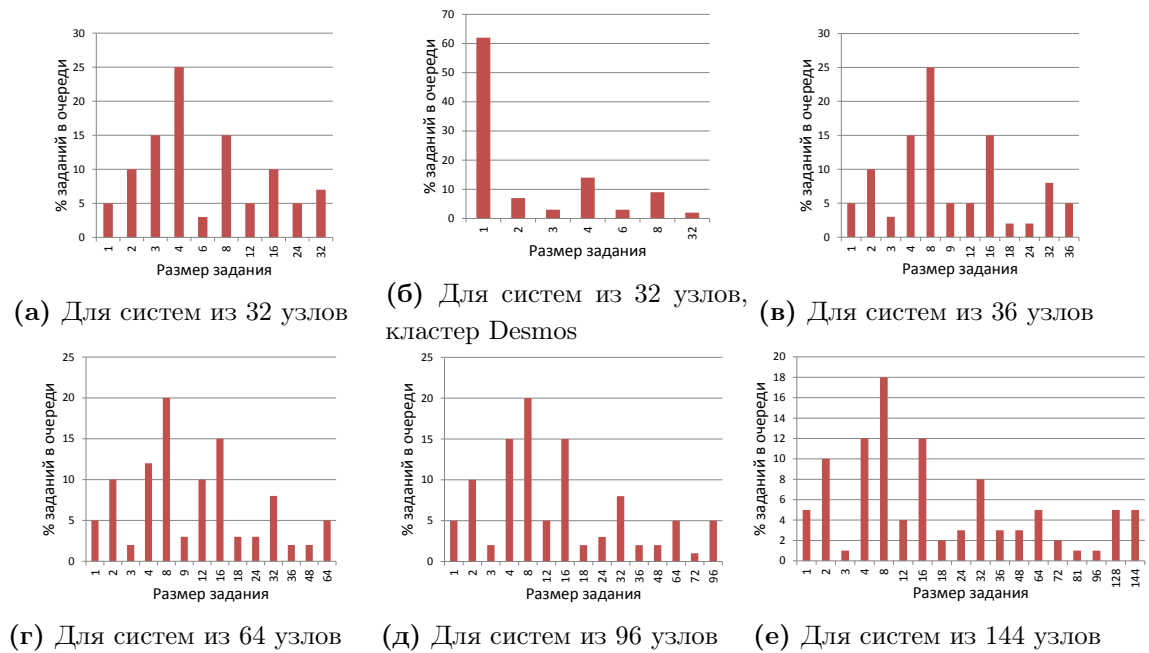
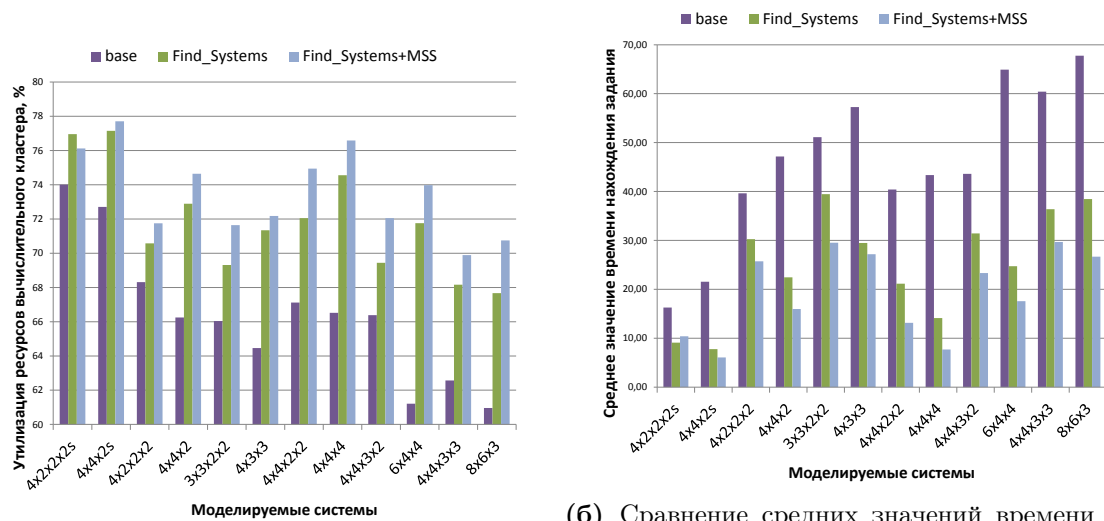


Рис. 3. Распределение заданий для систем с разным числом узлов

Вероятности для остальных чисел узлов равны 0. Задания равномерно случайно размещаются на временной шкале в диапазоне $[0; 40500]$ секунд вне зависимости от числа требуемых узлов. Время продолжительности заданий равномерно распределено по диапазону $[50; 100]$.

Для исследования разработанного алгоритма поиск ресурсов для заданий производился четырьмя различными способами: методом *Find_Systems* без применения разработанной метрики (*Find_Systems*); методом *Find_Systems_{MSS}* с применением разработанной метрики (*Find_Systems+MSS*); методом, который изначально функционировал на кластерах с сетью Ангара (*base*).

Диаграмма сравнения утилизации для различных методов и систем представлена на рис. 4а. Сравнение средних значений времени нахождения задания в очереди относительно запрошенного времени (T_{mid}) для различных методов и систем представлено на рис. 4б.



(а) Сравнение утилизации вычислительного кластера для различных системы и методов
(б) Сравнение средних значений времени нахождения задания в очереди относительно запрошенного времени (T_{mid}) для различных методов и систем

Рис. 4. Результаты моделирования

Разработанный алгоритм *Find_Systems* из статьи [13] сократил в среднем значение времени нахождения заданий в очереди на 45,3%, а утилизация ресурсов кластера в среднем увеличилась на 8,4% относительно базового метода *base*.

Добавление в *Find_Systems* учета фрагментированности вычислительной системы позволило еще сократить значение времени нахождения заданий в очереди до 57% относительно метода *base*, что на 11,7% больше чем у метода *Find_Systems*. Утилизация ресурсов при этом увеличивается до 11% относительно метода *base*, что на 2,6% больше чем у метода *Find_Systems*.

Заключение

В данной работе представлен метод выбора вычислительных узлов для потока пользовательских заданий, сокращающий фрагментацию вычислительной системы в сети Ангара с топологией многомерный тор. Метод основан на алгоритме выделения ресурсов для задания таким образом, чтобы максимизировать оставшееся пространство в многомерном торе. Это достигается построением прямоугольников максимального размера, которые возможно вписать в систему после размещения очередного пользовательского задания. Каждое множество узлов, подходящее для размещения задания, оценивается предложенной функцией, учитывающей размер и количество найденных прямоугольников максимального размера.

Проведено экспериментальное исследование разработанного алгоритма для различных вычислительных систем с топологией многомерный тор с общим числом узлов 32, 36, 64, 96 и 144, при этом рассмотрены 3х-мерные и 4х-мерные конфигурации топологий. Эксперимент проводился при помощи разработанного симулятора вычислительной системы. Для каждой исследуемой вычислительной системы была создана синтетическая очередь заданий, параметры которой приближены к реальным.

Средняя утилизация ресурсов для разработанного алгоритма выбора узлов увеличилась на 11%, а значение времени нахождения заданий в очереди сократилось на 57% по сравнению с первоначальным вариантом алгоритма.

В дальнейшем необходимо провести исследование разработанного алгоритма для систем большого размера, а также детально исследовать и провести возможную оптимизацию времени работы алгоритма. Также возможно провести исследование о влиянии перестановки задач в очереди для улучшения характеристик использования вычислительной системы.

Литература

1. Lan Z., Tang W., Wang J., Yang X., Zhou Z., Zheng X. Balancing Job Performance with System Performance via Locality-aware Scheduling on Torus-connected Systems // 2014 IEEE International Conference on Cluster Computing (CLUSTER). 2014. P. 140–148. DOI: 10.1109/CLUSTER.2014.6968751.
2. IBM Redbooks Publication: IBM System Blue Gene Solution: Blue Gene/Q System Administration. 2013. 282 p.
3. Tang W., Lan Z., Desai N., Buettner D., Yu Y. Reducing Fragmentation on Torus-Connected Supercomputers // In Proceedings of the 2011 IEEE International Parallel Distributed Processing Symposium (IPDPS'11). IEEE Computer Society, Washington, DC, USA. 2011. P. 828–839 DOI: 10.1109/IPDPS.2011.82.
4. Cray Document: Managing System Software for Cray XE and Cray XT Systems. 2010.
5. Агарков А.А., Исмаилов Т.Ф., Макагон Д.В., Семенов А.С., Симонов А.С. Результаты оценочного тестирования отечественной высокоскоростной коммуникационной сети Ангара // Суперкомпьютерные дни в России: Труды международной конференции (26–27 сентября 2016 г., г. Москва). М.: Изд-во МГУ, 2016. С. 626–639.
6. Симонов А.С., Макагон Д.В., Жабин И.А., Щербак А.Н., Сыромятников Е.Л., Поляков Д.А. Первое поколение высокоскоростной коммуникационной сети «Ангара» // Научные технологии. 2014. Т. 15. №1. С. 21–28.
7. Puente V., Beivide R., Gregorio J.A., Pallezo J.M., Duato J., Izu C. Adaptive Bubble Router: a Design to Improve Performance in Torus Networks // Proceedings of the International Conference Parallel Processing (ICPP). 1999. P. 58–67. DOI: 10.1109/ICPP.1999.797388.
8. Adiga N.R., Blumrich M., Chen D. Blue Gene/L Torus Interconnection Network // IBM Journal of Research and Development. 2005. Vol. 49. No. 2. P. 265–276. DOI: 10.1147/rd.492.0265.
9. Scott S.L. The Cray T3E Network: Adaptive Routing in a High Performance 3D Torus. 1996.
10. Пожилов И.А., Семенов А.С., Макагон Д.В. Алгоритм определения связности сети с топологией «многомерный тор» с отказами для детерминированной маршрутизации // Программная инженерия. 2015. № 3. С. 13–19.
11. Кагиров Р.Р. Многомерная задача о рюкзаке: новые методы решения // Вестник Сибирского государственного аэрокосмического университета им. академика МФ Решетнева. 2007. №3.
12. Gonçalves J.F., Resende M.G.C. A Parallel Multi-population Biased Random-key Genetic Algorithm for a Container Loading Problem // Computers & Operations Research. February 2012. Vol. 39. No. 2. P. 179–190. DOI: 10.1016/j.cor.2011.03.009.

13. Мукосей А.В., Семенов А.С., Приближенный алгоритм выбора оптимального подмножества узлов в коммуникационной сети Ангара с отказами // Вычислительные методы и программирование. 2017. Т. 18. С. 53–64.
14. Аладышев О.С., Киселёв Е.А. Алгоритм эффективного размещения программ на ресурсах многопроцессорных вычислительных систем // Программные продукты и системы. 2012. №4. С. 18–25.
15. Баранов А.В., Киселёв Е.А., Ляховец Д.С. Квазипланировщик для использования простаивающих вычислительных модулей многопроцессорной вычислительной системы под управлением СУППЗ // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2014. Т. 3. №4. С. 75–84. DOI: 10.14529/cmse140405.
16. Полежаев П.Н. Симулятор вычислительного кластера и его управляющей системы, используемый для исследования алгоритмов планирования задач // Вестник ЮУрГУ. Серия: Математическое моделирование и программирование. 2010. № 6. С. 79–90.
17. Sharma D.D., Pradhan D.K. A Fast and Efficient Strategy for Submesh Allocation in Mesh-connected Parallel Computers // In Procs. of the 5th IEEE Symp. on Parallel and Distributed Processing. 1993. P. 682–689. DOI: 10.1109/SPDP.1993.395466.
18. Dlinnova E., Smirnov G., Stegailov V., Biryukov S., Kondratyuk N. Hybrid Supercomputer Desmos with Torus Angara Interconnect: Performance and Efficiency Optimisation // 12th International Conference, PCT 2018, Rostov-na-Donu, Russia, April 2–6, 2018.

Мукосей Анатолий Викторович, научный сотрудник, сектор управления разработки вычислительной техники, акционерное общество «Научно исследовательский центр электронной вычислительной техники» (Москва, Российская Федерация)

Семенов Александр Сергеевич, к.т.н, зам. начальника отдела архитектуры и программного обеспечения суперкомпьютеров, акционерное общество «Научно исследовательский центр электронной вычислительной техники» (Москва, Российская Федерация)

ALLOCATION OPTIMIZATION FOR REDUCING RESOURCE FRAGMENTATION IN ANGARA HIGH-SPEED INTERCONNECT

© 2018 A.V. Mukosey, A.S. Semenov

JSC «NICEVT»

117587 Moscow, Varshavskoye shosse 125c15

E-mail: mukosey@nicevt.ru, semenov@nicevt.ru

Received: 04.06.2018

This paper considers a high-speed interconnect with a multidimensional topology. The paper is devoted to the optimization of fragmentation resulting from sequential allocation of compute nodes in a supercomputer provided that network traffic from different user's tasks should not overlap. This paper proposes a method for searching nodes with an evaluation of the fragmentation of the system based on ideas from multidimensional container packaging problem. For such an evaluation, the concept of rectangles is introduced, which can be inscribed into the system after placing the next user task. Each set of nodes that is suitable for placing the task is evaluated by the proposed function taking into account the size and the number of found rectangles of maximum size. The

proposed method was evaluated using computer system model. A set of different computer systems with three-dimensional and four-dimensional topologies was considered. The minimum system size is 32 compute nodes and the maximum is 144. A synthetic queue of tasks is set for each system. The parameters of the synthetic queues are close to a real ones. The average utilization of the resources of the computer system and the average waiting time for the tasks in the queue is chosen as a method quality criterion. The study showed that the increase of the resources utilization for the proposed method averaged 11% compared to the base method, and the average time spent in queue is reduced by 45,3%.

Keywords: Angara interconnect, multidimensional torus, deterministic routing, direction ordered routing, fragmentation, allocation

FOR CITATION

Mukosey A.V., Semenov A.S. Allocation Optimization for Reducing Resource Fragmentation in Angara High-speed Interconnect. *Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering*. 2018. vol. X, no. Y. pp. Z1–Z2. (in Russian) DOI: 10.14529/cmseXXXXXX.

References

1. Lan Z., Tang W., Wang J., Yang X., Zhou Z., Zheng X. Balancing job Performance with System Performance via Locality-aware Scheduling on Torus-connected Systems. 2014 IEEE International Conference on Cluster Computing (CLUSTER). 2014. pp. 140–148. DOI: 10.1109/CLUSTER.2014.6968751.
2. IBM Redbooks Publication: IBM System Blue Gene Solution: Blue Gene/Q system administration. 2013. 282 p.
3. Tang W., Lan Z., Desai N., Buettner D., Yu Y. Reducing Fragmentation on Torus-Connected Supercomputers. In Proceedings of the 2011 IEEE International Parallel Distributed Processing Symposium (IPDPS'11). IEEE Computer Society, Washington, DC, USA. 2011. pp. 828–839 DOI: 10.1109/IPDPS.2011.82.
4. Cray Document: Managing System Software for Cray XE and Cray XT Systems. 2010.
5. Agarkov A.A., Ismagilov T.F., Makagon D.V. Performance Evaluation of the Angara Interconnect. in Proc. Int. Conf. on Russian Supercomputing Days, Moscow, Russia, September 26–27, 2016 (Mosk. Gos. Univ., Moscow, 2016), pp. 626–639.
6. Simonov A.S., Makagon D.V., Zhabin I.A., Shcherbak A.N., Syromyatnikov E.L., Polyakov D.A. The First Generation of Angara High-Speed Interconnect. *Science Technologies*. 2014. vol. 15. no. 1. pp. 21–28.
7. Puente V., Beivide R., Gregorio J.A., Prellezo J.M., Duato J., Izu C. Adaptive Bubble Router: a Design to Improve Performance in Torus Networks. Proceedings of the International Conference Parallel Processing (ICPP). 1999. pp. 58–67. DOI: 10.1109/ICPP.1999.797388.
8. Adiga N.R., Blumrich M., Chen D.. Blue Gene/L Torus Interconnection Network. *IBM Journal of Research and Development*. 2005. vol. 49. no. 2. pp. 265–276. DOI: 10.1147/rd.492.0265.
9. Scott S.L., et al. The Cray T3E Network: Adaptive Routing in a High. Performance 3D Torus. 1996.
10. Pozhilov I.A., Semenov A.S., Makagon D.V., Connectivity Problem Solution for Direction Ordered Deterministic Routing in nD Torus. *Software Engineering*. 2015. no. 3. pp. 13–19.

11. Kagiroy R.R. Multiple Knapsack Problem: New Solving Methods. Vestnik of the Reshetnev Siberian State Aerospace University. 2007. iss. 3. pp. 16–20 (in Russ.).
12. Gonçalves J. F., Resende M. G. C. A Parallel Multi-population Based Random-key Genetic Algorithm for a Container Loading Problem. Computers & Operations Research. February 2012. vol. 39. no. 2. pp. 179–190. DOI: 10.1016/j.cor.2011.03.009.
13. Mukosey A.V., Semenov A.S. An Approximate Algorithm for Choosing the Optimal Subset of Nodes in the Angara Interconnect with Failures. Numerical methods and Programming. 2017. vol. 18. pp. 53–64.
14. Aladyshev O.S., Kiselov E.A. An efficient application mapping algorithm for multiprocessor systems. Software & Systems. 2012. no. 4. pp. 18–25.
15. Baranov A.V., Kiselev E.A., Lyakhovets D.S. The quasi scheduler for utilization of multiprocessing computing system's idle resources under control of the management system of the parallel jobs. Vestnik Yuzhno-Ural'skogo Gosudarstvennogo Universiteta. Seriya «Vychislitelnaya Matematika i Informatika». 2014. vol. 3. no. 4. pp. 75–84. DOI: 10.14529/cmse140405.
16. Polezhaev P.N. Simulator of computer cluster and its management system used for research of job scheduling algorithms. Vestnik Yuzhno-Ural'skogo Gosudarstvennogo Universiteta. Seriya «Vychislitelnaya Matematika i Informatika». 2010. vol. 6. pp. 79–90.
17. Sharma D. D., Pradhan D. K. A fast and efficient strategy for submesh allocation in mesh-connected parallel computers. In Procs. of the 5th IEEE Symp. on Parallel and Distributed Processing. 1993. pp. 682–689. DOI: 10.1109/SPDP.1993.395466.
18. Dlinnova E., Smirnov G., Stegailov V., Biryukov S., Kondratyuk N. Hybrid Supercomputer Desmos with Torus Angara Interconnect: Performance and Efficiency Optimisation. 12th International Conference, PCT 2018, Rostov-na-Donu, Russia, April 2–6, 2018.