УДК 004.272

А. А. Пазников, А. В. Табаков, М. С. Куприянов, А. Р. Лисс
Санкт-Петербургский государственный электротехнический университет «ЛЭТИ» им. В. И. Ульянова (Ленина)

Алгоритм выбора локальных окрестностей децентрализованных диспетчеров в пространственно-распределенных вычислительных системах

Исследуется влияние выбора локальных окрестностей децентрализованных диспетчеров пространственно-распределенных мультикластерных и Grid-систем на эффективность диспетчеризации. Построена математическая модель для оценки эффективности заданной структуры локальных окрестностей для систем децентрализованной диспетчеризации с возможностью миграции задач. Предложен нетрудоемкий эвристический алгоритм, позволяющий строить субоптимальные графы логических структур вычислительных систем. Приводятся результаты натурного моделирования диспетчеризации параллельных MPI-программ с использованием распространенных логических структур, которые демонстрируют влияние выбора локальных окрестностей на показатели эффективности обслуживания задач. Выполнен сравнительный анализ распространенных логических структур и графов, генерируемых созданным алгоритмом. Результаты численного моделирования показывают, что предлагаемый алгоритм позволяет в разы сократить стоимость обслуживания задач.

Децентрализованная диспетчеризация, мультикластерные системы, пространственно-распределенные вычислительные системы, Grid, локальная окрестность, логическая структура

Применение пространственно-распределенных вычислительных систем (ВС) является сегодня одной из основных тенденций в области высокопроизводительной обработки информации. Данные системы представимы в виде композиции территориально распределенных вычислительных кластеров или систем с массовым параллелизмом (подсистем). Взаимодействие между ними осуществляется через сети связи различного уровня [1]–[3]. К пространственно-распределенным относятся мультикластерные и Grid-системы. Такие системы являются большемасштабными (современные Grid-системы, такие как Worldwide LHC Computing Grid, World Community Grid, NorduGrid, Open Science Grid, Grid-5000, включают десятки и сотни подсистем) и разнородными по составу.

Каждая подсистема в составе мультикластерной или Grid-системы поддерживает выполнение параллельных задач с помощью системы управления ресурсами (СУР): SLURM, TORQUE, PBS Pro и др. СУР организует выделение подсистем процессорных ядер и выполнение пользовательских задач. В пространственно-распределенных ВС могут использоваться гетерогенные и несовместимые вычислительные средства; программная и аппаратная совместимость обеспечивается посредством промежуточного программного обеспечения (middleware), к которому относят средства авторизации пользователей (MyProxy, SimpleCA, Unity), средства передачи и синхронизации данных (GridFTP, RLS, UFTPD, Shared Registry) и средства взаимодействия с локальными СУР (GRAM, TSI, UNICORE/X).

Рассмотрим функционирование распределенной ВС в режиме обслуживания потока задач [1]. В систему в случайные моменты времени поступают задачи, представленные в виде параллельных программ. Каждая задача характеризуется рангом (числом параллельных ветвей – процессов, потоков), ожидаемым временем решения, требованиями по архитектуре, аппаратному и программному обеспечению, размерами и расположением входных и выходных данных и т. д.

В режиме обслуживания потока задач существует задача диспетчеризации. Для поступаю-

щих параллельных программ требуется выделить множество процессорных ядер с подсистем распределенной ВС с последующим их выполнением. Целью диспетчеризации является оптимизация заданных показателей эффективности (время обслуживания задачи, пропускная способность системы, энергопотребление и т. д.). При этом необходимо учитывать большемасштабность ВС, иерархическую структуру и динамический характер состава и загрузки ресурсов. Задача диспетчеризации возникает для задач, разработанных как в модели передачи сообщений [4]–[7], так и в моделях RMA [8], PGAS [9], [10], а также для многопоточных программ [11]–[13].

Для решения задачи диспетчеризации применяется централизованный и децентрализованный подходы. Основным недостатком централизованного подхода является то, что отказ диспетчера приводит к неработоспособности всей системы. Помимо этого, в случае использования централизованной системы в большемасштабных ВС возрастают временные затраты на поиск требуемых ресурсов.

На сегодняшний день среди программных пакетов диспетчеризации задач в пространственно-распределенных ВС в первую очередь следует назвать [14], [15]: GridWay, AppLeS, GrADS, Nimrod/G, Condor-G, WMS. Пакет GridWay [16] является наиболее распространенным и интенсивно развивающимся GRID-диспетчером. В GridWay при диспетчеризации преследуется цель минимизации времени обслуживания задач; поддерживается механизм миграции задач между подсистемами и реализован инструмент выполнения задач, представленных в виде ориентированных ациклических графов. В пакете AppLeS [17] диспетчеризация выполняется на уровне самой программы. Это требует от пользователей знания конфигурации системы, что снижает универсальность диспетчера. Пакет GrADS [18], как и GridWay, реализует миграцию задач и допускает указание зависимостей по данным между задачами. Nimrod/G [19] основан на экономических моделях, обеспечивающих равновесие между условными поставщиками (подсистемами) и потребителями (задачами) вычислительных ресурсов. Диспетчер Condor-G [20] максимизирует пропускную способность системы и поддерживает представление задач в виде ориентированных ациклических графов.

В отдельную группу следует выделить работы, связанные с созданием инструментария диспетчеризации задач, имеющих зависимости по данным и представленных в виде ориентированных ациклических графов (Grid Workflows). Существующие системы диспетчеризации таких задач (например, Pegasus [21], Taverna [22], Triana [23], ASKALON [24], ICENI [25], Kepler [26], UNICORE [27]) являются централизованными. Большинство этих диспетчеров ориентировано на применение в узкоспециализированной области. Кроме того, в них нет поддержки Grid-сервисов и промежуточного программного обеспечения пространственно-распределенных ВС, что затрудняет их применение в мультикластерных и Grid-системах.

В настоящее время получил широкое распространение децентрализованный подход к диспетчеризации [28]–[34], представленный такими диспетчерами, как GBroker [28]–[30], SmartGRID [31], DIRAC [32]. Программный пакет GBroker поддерживает семейство децентрализованных алгоритмов диспетчеризации, позволяющих учитывать динамическую загрузку ресурсов и каналов связи пространственно-распределенных ВС и обеспечивать их живучее функционирование. GBroker реализует миграцию и репликацию задач между подсистемами с целью минимизации времени их обслуживания.

В системах децентрализованной диспетчеризации в пространственно-распределенной ВС функционирует коллектив диспетчеров, которые координируют свои действия по выделению множества процессорных ядер для решения поступающих в систему задач. При этом каждый диспетчер может отправлять и получать запросы (о состоянии загруженности системы, о назначении задач) заданному количеству других диспетчеров, которые формируют его локальную окрестность. Композиция таких локальных окрестностей определяет логическую структуру ВС, которую можно представить в виде ориентированного графа. Авторы полагают, что от выбора локальных окрестностей (логической структуры ВС) существенно зависит время обслуживания параллельных задач.

Как известно, проблема поиска субоптимальных логических структур децентрализованных диспетчеров в настоящее время недостаточно исследована. В связи с этим данная задача и рассматривается в настоящей работе.

Алгоритм поиска субоптимальных локальных окрестностей диспетчеров. Математическая модель диспетчеризации. При выборе локальных окрестностей децентрализованных

диспетчеров могут быть использованы известные перспективные структуры, например, *ND*-торы, решетки, *N*-мерные гиперкубы, D_n-графы, $Л(N, v, g)$-графы [1]. Однако недостатком таких структур является их однородность. Вместе с тем известно, что в пространственно-распределенных ВС загрузка подсистем может существенно различаться и изменяться с течением времени. Поэтому неоднородные логические структуры локальных окрестностей могут быть более эффективны для пространственно-распределенных ВС, и поэтому актуальным является создание алгоритмов поиска таких структур.

Опишем формальную модель диспетчеризации в пространственно-распределенных ВС с возможностью миграции задач. Предположим, имеется пространственно-распределенная ВС, включающая H вычислительных подсистем, поддерживающих очередь параллельных задач, организованную диспетчером. Множество диспетчеров может быть представлено в виде ориентированного графа $G = (V, E)$, в котором вершинам соответствуют диспетчеры, а ребрам – логические связи между ними (рис. 1). Наличие дуги $(i, j) \in E$ означает возможность отправки задачи диспетчером i задач своей очереди диспетчеру j. Множество диспетчеров j, смежных с диспетчером i, образует его локальную окрестность $L(i)$ (т. е. диспетчеры, которым диспетчер i может отправлять задачи).

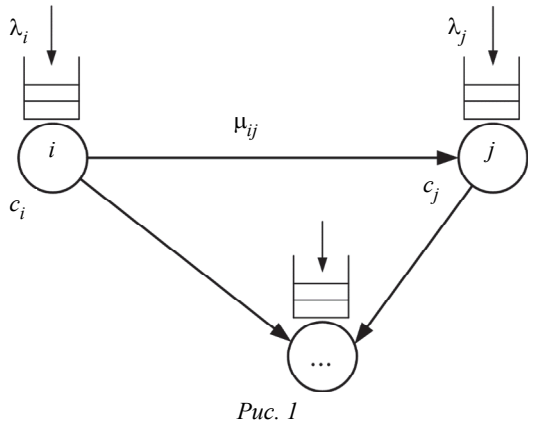


Рис. 1

Введем обозначения: c – стоимость использования подсистемы в единицу времени; λ – интенсивность потока поступления задач на подсистему i; μ – интенсивность потока миграции задач с подсистемы i на подсистему j. Пусть $x = \{0, 1\}$ – наличие дуги от подсистемы i к j: $x = 1$, если существует дуга $(i, j) \in E$, иначе $x = 0$. Значения x представлены в виде матрицы $X = \{x : i \in V, j \in C\}$.

Можно оценить количество задач, мигрировавших с подсистемы i на другие подсистемы: $\sum_{j=1...H} x_{ij}\mu_{ij}$. Время обслуживания мигрировавших задач складывается из времени t обслуживания задачи на первой назначенной подсистеме и штрафа kt за миграцию (k – коэффициент штрафа за миграцию). Тогда стоимость обслуживания задач, мигрировавших с подсистемы i: $\sum_{j=1...H} x_{ij}\mu_{ij}c_j(t_i + kt_i)$. Стоимость решения немигрировавших задач на подсистеме i:

$$\left(\lambda_i - \sum_{j=1...H} x_{ij}\mu_{ij}\right)c_i t_i .$$

Таким образом, суммарная стоимость обслуживания задач на подсистеме i:

$$\left(\lambda_i - \sum_{j=1...H} x_{ij}\mu_{ij}\right)c_i t_i + \sum_{j=1...H} x_{ij}\mu_{ij}c_j(t_i + kt_i).$$

Сформулируем задачу поиска оптимальных структур локальных окрестностей диспетчеров. Необходимо минимизировать функцию штрафа за обслуживание задач на пространственно-распределенной ВС:

$$F(X) = \sum_{i=1...H}\left(\left(\lambda_i - \sum_{j=1...H} x_{ij}\mu_{ij}\right)\times\right.$$

$$\left.\times c_i t_i + \sum_{j=1...H} x_{ij}\mu_{ij}c_j(t_i + kt_i)\right) \to \min_{(x_{ij})} \qquad (1)$$

при ограничениях

$$x_{ij} = \{0,1\} \qquad (2)$$

$$\sum_{j=1...H} x_{ij} \le l, \quad i = 1, 2, \ldots, H, \qquad (3)$$

где l – максимально допустимый размер локальной окрестности диспетчеров.

Задача (1)–(3) относится к целочисленной оптимизации. Целесообразным является разработка эвристических алгоритмов поиска субоптимальных локальных окрестностей диспетчеров.

Алгоритм формирования субоптимальных локальных окрестностей диспетчеров. На основе метода локального поиска предложен эвристический алгоритм DS_LN_LS (Decentralized Scheduling Local Neighborhood Local Search) формирования субоптимальных локальных окрестностей диспетчеров. На рис. 2 представлен псевдокод алгоритма.

Входные данные: λ – интенсивности потоков поступления задач на подсистему (для каждой подсистемы $i \in V$);
μ – интенсивности потоков миграции задачи с подсистемы i на подсистему j (для каждой пары $i, j \in V$);
c – стоимость использование подсистемы i в единицу времени;
t – среднее время обслуживания задачи на подсистеме i;
k – коэффициент штрафа за миграцию;
l – максимальный размер локальной окрестности диспетчера.

Выходные данные: матрица значений X: $x = 1$, если существует дуга $(i, j) \in E$, иначе $x = 0$.

```
1   x* ← InitSol(l)          // сформировать начальное субоптимальное решение
2   d ← d                    // начальная дистанция
3   while d > d do
4     for i ← 0 to i do
5       x ← GenSol(x, d, l)  // сгенерировать решение на расстоянии d от текущего
6       if F(x, λ, μ, c, t, k) < F(x*, λ, μ, c, t, k) then
7         x* ← x
8         break
9       end if
10    end for
11    d ← d / 2
12  end while
```

Рис. 2

На первом шаге (например, «жадным» алгоритмом) генерируется начальное решение (процедура InitSol()). Далее случайным образом генерируется новое решение на расстоянии d от текущего лучшего решения x^*. Расстояние между решениями – число различающихся значений x в новом и предыдущем решениях. На следующем этапе среди сгенерированного и лучшего решения определяется минимальное значение. Если целевая функция нового решения меньше лучшего на текущий момент, текущее решение принимается за лучшее. Если после i итераций не найдено очередное лучшее решение, расстояние d уменьшается вдвое. Алгоритм выполняется до тех пор, пока расстояние d между решениями больше минимального расстояния d.

Результаты моделирования. Анализ эффективности распространенных логических структур. Было выполнено натурное моделирование с целью анализа эффективности диспетчеризации при использовании перспективных логических структур ВС. Эксперименты проводились на пространственно-распределенной мультикластерной ВС, созданной Центром параллельных вычислительных технологий Сибирского государственного университета телекоммуникаций и информатики (ЦПВТ СибГУТИ) и лабораторией ВС Института физики полупроводников им. А. В. Ржанова (ИФП СО РАН). Число подсистем $H = 6$, количество процессорных ядер $N = 130$. Программное обеспечение: GNU/Linux, локальная СУР на подсистемах TORQUE 2.3.7, промежуточное программное обеспечение Globus Toolkit 5.0, пакет диспетчеризации GBroker (в том числе модули мониторинга DCSMon и NetMon) [27]–[29].

Применялись MPI-программы из пакета SPEC MPI 2007 тестов производительности распределенных ВС: WRF, POP2, LAMMPS, RAxML, Tachyon. Первый сегмент содержал входные данные. Генерировались простейшие потоки с различной интенсивностью λ поступления задач, которые псевдослучайно с равномерным распределением выбирались из тестового набора. Выходные данные также поступали на первую подсистему. Обозначим общее число задач M. Число процессов r (ранг) программы выбирался случайно из набора $\{1, 2, 4, 8\}$.

Введем обозначения: t – момент поступления программы $k \in \{1, 2, \ldots, M\}$, t'_k – момент начала выполнения программы k, t''_k – момент завершения выполнения программы k. Обозначим также τ – общий период обслуживания M задач потока.

Для оценки эффективности алгоритмов диспетчеризации использовались следующие показатели: пропускная способность B системы, среднее время T обслуживания задачи и среднее время W пребывания задачи в очереди:

$$B = \frac{M}{\tau},\ T = \frac{1}{M}\sum_{k=1}^{M}(t''_k - t_k),\ W = \frac{1}{M}\sum_{k=1}^{M}(t'_k - t_k).$$

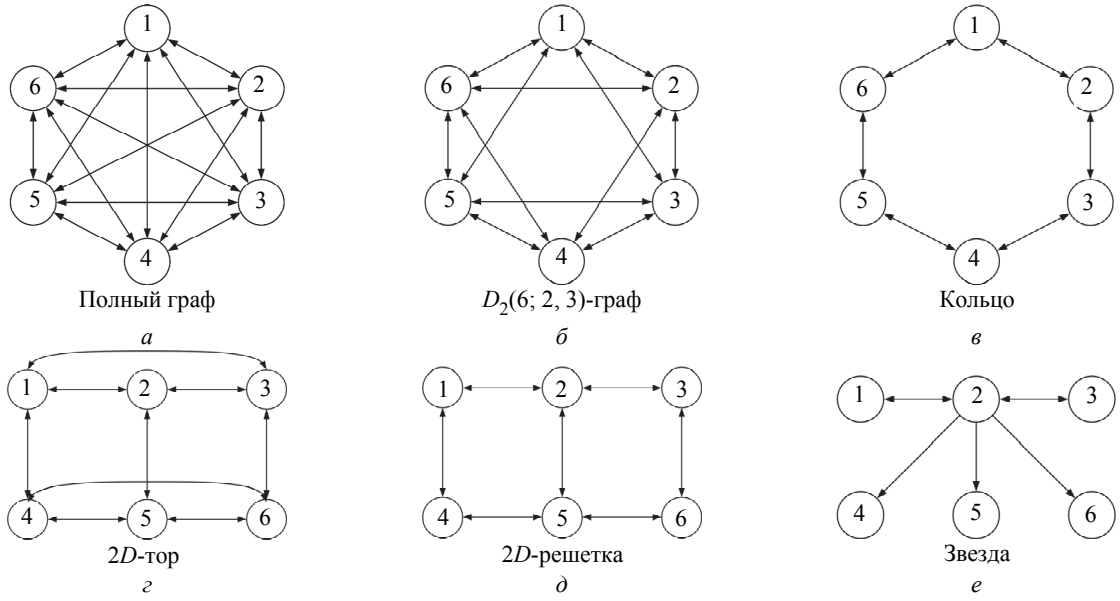


Рис. 3

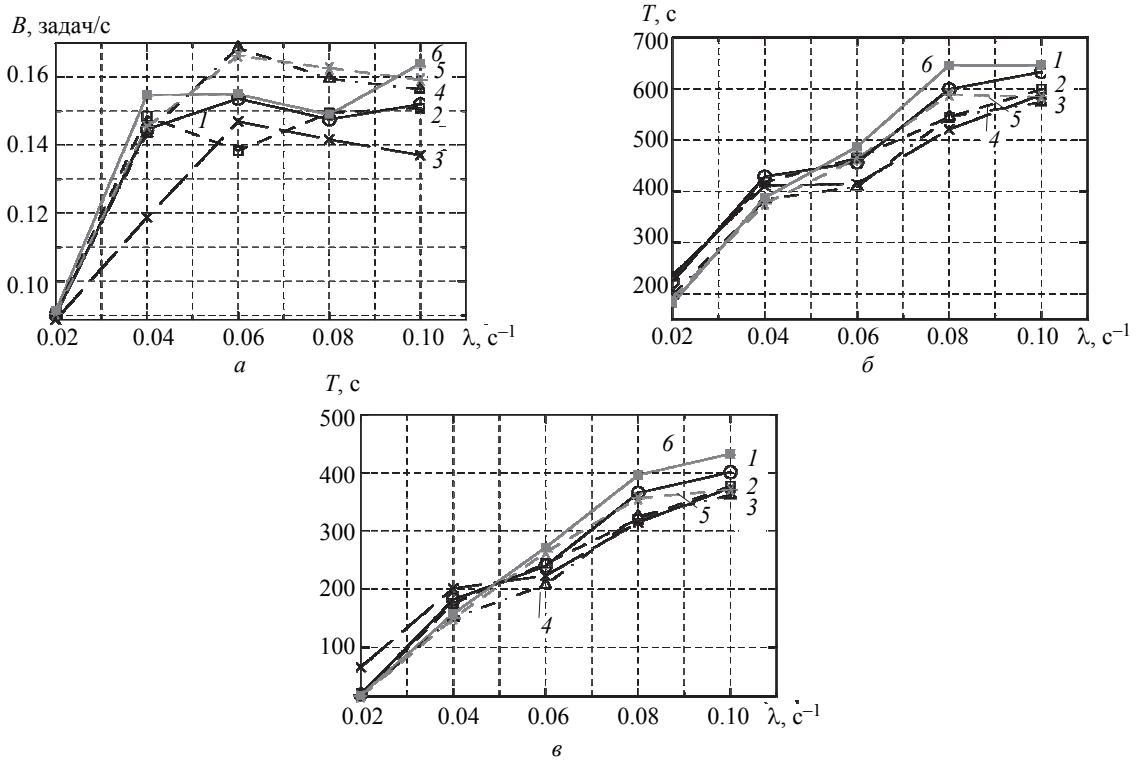


Рис. 4

Исследовалась эффективность диспетчеризации в зависимости от выбора логической топологии диспетчеров. Использовались топологии пространственно-распределенных ВС в виде полных графов, колец, решеток, звезд, 2*D*-торов и *D*-графов [1] (рис. 3). В проведенном моделировании количество задач $M = 50$. Программы одновременно поступали на все подсистемы. Использовался алгоритм диспетчеризации на основе миграции программ.

В результате проведенных экспериментов получены графики зависимостей показателей эффективности обслуживания потока задач от интенсивности потока задач [27]–[29] (рис. 4) (*1* – полный граф; *2* – звезда; *3* – кольцо; *4* – решетка; *5* – 2*D*-тор; *6* – *D*-граф). Сопоставимые значения показателей, кроме полного графа, были получены для топологий на основе 2*D*-тора, двумерной решетки и *D*-графа (для двух последних достигнуты максимальные значения). Таким образом,

построение ВС на основе неполносвязных графов не приводит к значительному снижению эффективности диспетчеризации.

Структуры, отличные от полносвязной, могут быть сформированы при отсутствии непосредственных линий связи между подсистемами, например, при отказе некоторых подсистем. В качестве логических топологий можно рекомендовать торы, решетки или *D*-графы, обладающие малым (средним) диаметром.

Моделирование алгоритма генерации субоптимальных локальных окрестностей. Моделирование созданного алгоритма DS_LN_LS и сравнение с известными перспективными структурами проводилось на подсистеме мультикластерной ВС ЦПВТ СибГУТИ и ИФП СО РАН. Процессор – Intel Xeon E5420 (2.5 ГГц), размер памяти 8 Гб.

Выбраны следующие параметры алгоритма DS_LN_LS:

– начальное расстояние между решениями $d = 100$;

– количество итераций для каждого расстояния $i = 100$;

– максимальный размер локальной окрестности $l = 10$;

– коэффициент штрафа за миграцию задачи $k = 0.2$;

– стоимости использования подсистемы $c \in U(0, 1)$ (c – случайная величина, равномерно распределенная в интервале (0, 1);

– интенсивности потоков задач $\lambda \in U(0, 100)$;

– интенсивности потоков миграции задач $\mu \in U(0, 20)$;

– среднее время решения задачи $\lambda \in U(0, 200)$.

На рис. 5 представлено сравнение стоимости (штрафа) диспетчеризации ($F(x)$, о. е.) для логических структур локальных окрестностей, полученных с помощью алгоритма DS_LN_LS и распространенных перспективных структур. На рис. 6 приведены графы для ВС из $H = 8, 16$ подсистем, полученные с помощью алгоритма DS_LN_LS, а также соответствующие графам значения целевой функции. Отметим, что стоимость диспетчеризации для полного графа (не приведена на рис. 5, *б–г*) значительно превосходит стоимость диспетчеризации для остальных структур. Это объясняется высокой интенсивностью миграции задач на системы локальной окрестности (в которую входят все подсистемы). В реальных системах, кроме того, нужно учитывать накладные расходы на получение информации о системах локальной окрестности и дополнительную нагрузку на коммуникационную сеть. Результаты для структур тор и *D*-граф различают-

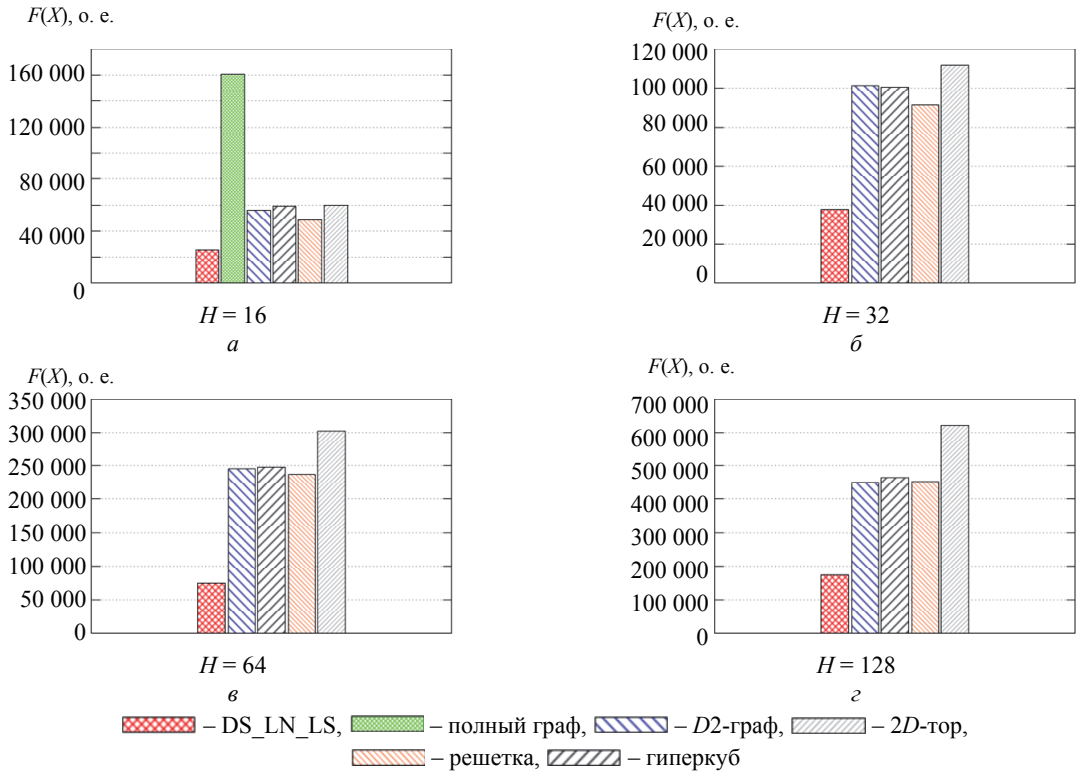
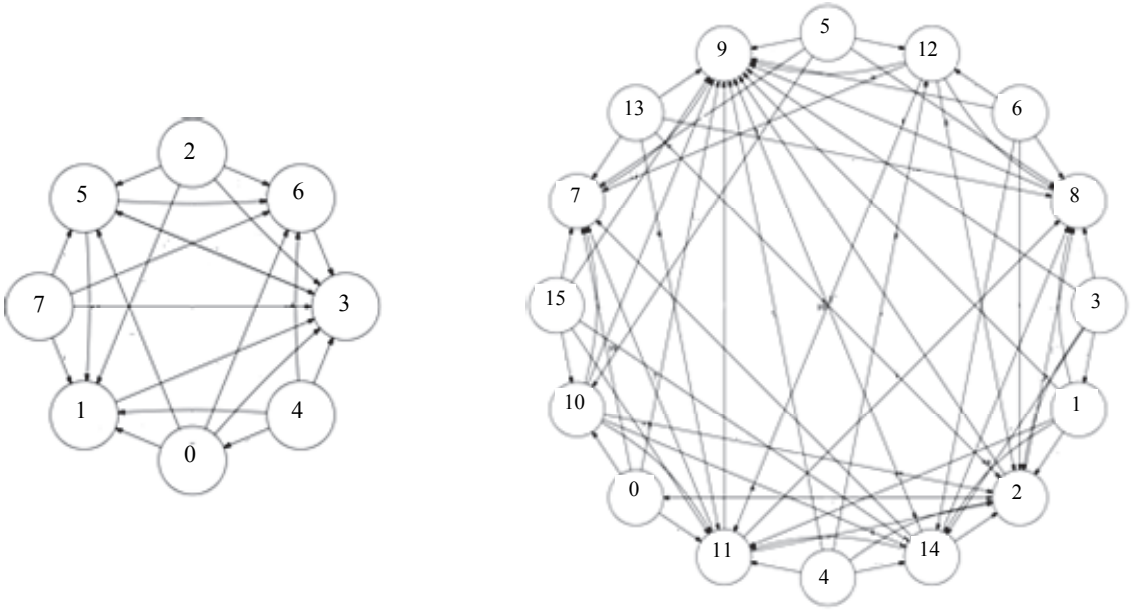


Рис. 5



$H = 8$, $F = 74\,466$
а

$H = 16$, $F = 25\,896$
б

Рис. 6

ся в пределах погрешности. Топологии распределенных ВС, полученные алгоритмом DS_LN_LS, позволяют сократить функцию штрафа в 2.5…3 раза. Это объясняется тем, что пространственно-распределенные ВС являются гетерогенными: интенсивности потоков поступления задач на подсистемы и миграции задач между подсистемами могут значительно варьироваться.

Основные перспективные структуры (гиперкубы, решетки, *N*-мерные торы, циркулянтные структуры и т. д.) относятся к однородным. Это не позволяют учитывать дисбаланс состава и загрузки подсистем. Данное обстоятельство ограничивает их применение при децентрализованной диспетчеризации. Алгоритм DS_LN_LS позволяет получать неоднородные структуры, которые обеспечивают минимизацию времени обслуживания задач и повышают пропускную способность системы. В процессе функционирования мультикластерной или Grid-системы периодически запускается процедура формирования локальных окрестностей диспетчеров на основе алгоритма DS_LN_LS. Данная процедура на основе данных о загрузке подсистем формирует субоптимальные локальные окрестности диспетчеров [3].

Время выполнения алгоритма DS_LN_LS (приведено на рис. 7) является приемлемым для большемасштабных систем, поскольку моделирование выполняется один раз перед настройкой логических структур администратором системы.

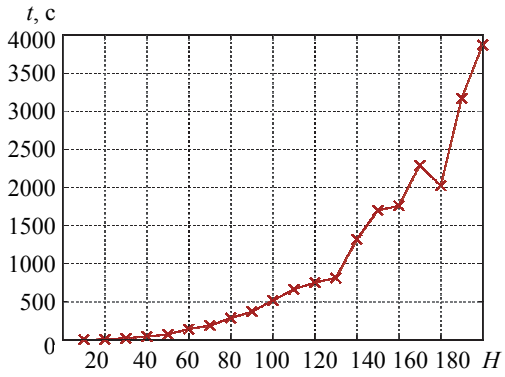


Рис. 7

В данной работе экспериментально показано, что эффективность децентрализованной диспетчеризации задач в пространственно-распределенных вычислительных системах зависит от выбора локальных окрестностей диспетчеров. К структурам, которые рекомендуется использовать для ВС с небольшим количеством подсистем, можно отнести полный граф, *ND*-тор, решетку и *D*-граф. При этом использование неполносвязных структур не приводит к значительному снижению показателей эффективности диспетчеризации. Такие структуры могут быть образованы при отсутствии прямых линий связи между отдельными подсистемами, например, в результате отказов некоторых диспетчеров или подсистем.

С увеличением количества подсистем в ВС полносвязные топологии становятся неэффективными, поскольку возрастают накладные расходы на миграцию задач и получение информации о

системах локальных окрестностей. Для генерации субоптимальных логических структур в большемасштабных системах была разработана математическая модель диспетчеризации задач в распределенных ВС и построен эвристический алгоритм формирования субоптимальных локальных окрестностей диспетчеров. Алгоритм позволяет сократить штраф при обслуживании потока параллельных задач в 2.5…3 раза по сравнению с известными перспективными структурами и может быть использован для повышения эффективности функционирования мультикластерных и Grid-систем.

Исследование выполнено при финансовой поддержке РФФИ в рамках научного проекта № 19-07-00784 и при поддержке Совета по грантам президента РФ для государственной поддержки молодых российских ученых (проект СП-4971.2018.5).

СПИСОК ЛИТЕРАТУРЫ

1. Хорошевский В. Г. Архитектура вычислительных систем. М.: Изд-во МГТУ им. Н. Э. Баумана, 2008. 520 с.

2. Масштабируемый инструментарий параллельного мультипрограммирования пространственно-распределенных вычислительных систем / В. Г. Хорошевский, М. Г. Курносов, С. Н. Мамойленко, К. В. Павский, А. В. Ефимов, А. А. Пазников, Е. Н. Перышкова // Вестн. СибГУТИ. 2011. № 4. С. 3–19.

3. Пазников А. А. Алгоритмы организации функционирования мультикластерных вычислительных систем с иерархической структурой: дис. … канд. техн. наук. Специальность 05.13.15 «Вычислительные машины, комплексы и компьютерные сети» / СибГУТИ. Новосибирск, 2013. 146 с.

4. Курносов М. Г., Пазников А. А. Эвристические алгоритмы отображения параллельных MPI-программ на мультикластерные вычислительные и GRID-системы // Вычислительные методы и программирование. 2013. № 14. С. 1–10.

5. Пазников А. А., Курносов М. Г., Куприянов М. С. Многоуровневые алгоритмы отображения параллельных MPI-программ на вычислительные кластеры // Проблемы информатики. 2015. № 1. С. 4–17.

6. Adaptive Barrier Algorithm in MPI Based on Analytical Evaluations for Communication Time in the LogP Model of Parallel Computation / V. Zharikov, A. Paznikov, K. Pavsky, V. Pavsky // Proc. of the Intern. Multi-conf. on Industrial Engineering and Modern Technologies (Far East Con-2018), Vladivostok, 2018. P. 1–5.

7. Жариков В. В., Пазников А. А. Адаптивный алгоритм барьерной синхронизации в стандарте MPI на основе модели параллельных вычислений LogP // Изв. СПбГЭТУ «ЛЭТИ». 2018. № 4. С. 26–32.

8. Paznikov A., Anenkov A. Implementation and Analysis of Distributed Relaxed Concurrent Queues in Remote Memory Access Model // Procedia Computer Science. 2019. Vol. 150. P. 654–662

9. Kulagin I., Paznikov A., Kurnosov M. Heuristic Algorithms for Optimizing Array Operations in Parallel PGAS-programs // Lecture Notes in Computer Science. 2015. Vol. 9251. P. 405–409.

10. Кулагин И. И., Пазников А. А., Курносов М. Г. Эвристические алгоритмы оптимизации информационных обменов в параллельных PGAS-программах // Вестн. СибГУТИ. 2014. № 3. С. 52–66.

11. Paznikov A., Shichkina Y. Algorithms for Optimization of Processor and Memory Affinity for Remote Core Locking Synchronization in Multithreaded Applications // Information. 2018. Vol. 9, № 1. P. 1–12.

12. Аненков А. Д., Пазников А. А. Алгоритмы оптимизации масштабируемого потокобезопасного пула на основе распределяющих деревьев для многоядерных вычислительных систем // Вестн. ТГУ. Управление, вычислительная техника и информатика. 2017. № 39. С. 73–84.

13. Tabakov A., Paznikov A. Algorithms for Optimization of Relaxed Concurrent Priority Queues in Multicore Systems // Proc. of the 2019 IEEE Conf. of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus), St. Petersburg, 2019. P. 360–365

14. Dong, F., Akl S. Scheduling Algorithms for Grid Computing: State of the Art and Open Problems: Tech. Rep. 2006-504. Ontario: School of Computing, Queen's University Kingston, 2006.

15. Xhafa F., Abraham A. Computational models and heuristic methods for Grid scheduling problems // Future generation computer systems. 2010. № 26 (4). P. 608–621.

16. Huedo E., Montero R. S., Llorente I. M. A framework for adaptive execution on grids // Software – Practice and Experience (SPE). 2004. Vol. 34. P. 631–651.

17. Berman F., Wolski R., Casanova H. Adaptive computing on the grid using AppLeS // IEEE Trans. on Parallel and Distributed Systems. 2003. Vol. 34. P. 369–382.

18. Berman F. The GrADS project: Software support for high-level grid application development // The Intern. J. of High Performance Computing Applications. 2001. № 15 (4). P. 327–344.

19. Buyya R., Abramson D., Giddy J. Nimrod/G: An architecture for a resource management and scheduling system in a global computational Grid // Proc. of the 4th Intern. Conf. on High Performance Computing, Beijing, 2000. P. 283–289.

20. Condor-G: A computation management agent for multi-institutional grids / J. Frey, T. Tannenbaum, M. Livny et al. // Cluster Computing. 2001. Vol. 5. P. 237–246.

21. Pegasus: A framework for mapping complex scientific workflows onto distributed systems / E. Deelman, G. Singh, M.-H. Su et al. // Scientific Programming. 2005. Vol. 13, № 3. P. 219–237.

22. Taverna: a tool for building and running workflows of services / D. Hull, K. Wolstencroft, R. Stevens et al. // Nucleic Acids Research. 2006. Vol. 34. P. 729–732.

23. Programming scientific and distributed workflow with Triana services: Research Articles / C. David, G. Gombas, A. Harrison et al. // Concurr. Comput.: Pract. Exp. 2006. Vol. 18, № 10. P. 1021–1037.

24. Wieczorek M., Prodan R., Fahringer T. Scheduling of scientific workflows in the askalon grid environment // ACM SIGMOD Record J. 2005. Vol. 34. P. 56–62.

25. Scheduling Architecture and Algorithms within the ICENI Grid Middleware / L. Young, S. Mcgough, S. Newhouse, J. Darlington. UK, Nottingham, 2003. P. 5–12.

26. Scientific workflow management and the Kepler system / B. Ludascher, I. Altintas, C. Berkley et al. // Concurr. Comput.: Pract. Exp. 2006. Vol. 18, № 10. P. 1039–1065.

27. Romberg M. The UNICORE Grid Infrastructure // Scientific Programming, Special Issue on Grid Computing. 2002. Vol. 10. P. 2002.

28. Kurnosov M. G., Paznikov A. A. Efficiency analysis of decentralized grid scheduling with job migration and replication // Proc. of ACM Intern. Conf. on Ubiquitous Information Management and Communication (IMCOM/ICUIMC), Kota Kinabalu, 2013. P. 1–7.

29. Курносов М. Г., Пазников А. А. Децентрализованные алгоритмы диспетчеризации пространственно-распределенных вычислительных систем // Вестн. ТГУ. Управление, вычислительная техника и информатика. 2012. № 1 (18). P. 133–143.

30. Курносов М. Г., Пазников А. А. Инструментарий децентрализованного обслуживания потоков параллельных MPI-задач в пространственно-распределенных мультикластерных вычислительных системах // Вестн. ТГУ. Управление, вычислительная техника и информатика. 2011. № 3 (16). С. 78–85.

31. SmartGRID: A Fully Decentralized Grid Scheduling Framework Supported by Swarm Intelligence / Y. Huang, A. Brocco, P. Kuonen et al. // Proc. of the 2008 7th Intern. Conf. on Grid and Cooperative Computing, Beijing, 2008. P. 160–168.

32. Evaluation of Meta-scheduler Architectures and Task Assignment Policies for High Throughput Computing: Tech. Rep. 5576 / E. Normale, S. Lyon, E. Caron et al. Lyon: Ecole Normale Superieure de Lyon, 2005. P. 1–17.

33. Decentralized Grid Scheduling with Evolutionary Fuzzy Systems / F. Alexander, C. Grimme, J. Lepping, A. Papaspyrou // Workshop on Job Scheduling Strategies for Parallel Processing. Rome, 2009. P. 16–36.

34. Exploring decentralized dynamic scheduling for grids and clouds using the community-aware scheduling algorithm / Y. Huang, N. Bessis, P. Norrington, P. Kuonen // Future Generation Computer Systems. 2013. № 29 (1). P. 402–415.

A. A. Paznikov, A. V. Tabakov, M. S. Kupriyanov, A. R. Liss
Saint Petersburg Electrotechnical University

ALGORITHM FOR CHOOSING LOCAL NEIGHBOURHOODS OF DECENTRALIZED SCHEDULERS IN GEOGRAPHICALLY DISTRIBUTED COMPUTER SYSTEMS

We study the influence of local neighborhood choice to the efficiency of decentralized scheduling in geographically distributed multicluster systems and computational grids. We constructed the mathematical model for evaluation of the efficiency of the logical structures. We also designed efficient algorithm, which builds suboptimal graphs of CS local structures. We give the results of full-scale experiments of scheduling of MPI-programs using common logical structures, which demonstrate the influence of the local neighborhood choice to the efficiency metrics. We also compare common structures with the structures, generated by means of designed algorithm. The results of numerical modeling show that the proposed algorithm significantly reduces the cost of job service.

Decentralized scheduling, multicluster systems, geographically distributed computer systems, Grid, local neighborhood, logical structure