УДК 519.687

Гергель В.П.[1], Полежаев П.Н.[2]
[1]Нижегородский государственный университет им. Лобачевского
[2]Оренбургский государственный университет
E-mail: peter.polezhaev@mail.ru

ТЕОРЕТИЧЕСКИЕ ОСНОВЫ ЭКСПЕРИМЕНТАЛЬНОГО ИССЛЕДОВАНИЯ АЛГОРИТМОВ ПЛАНИРОВАНИЯ ЗАДАЧ ДЛЯ ВЫЧИСЛИТЕЛЬНОГО КЛАСТЕРА С ПОМОЩЬЮ СИМУЛЯТОРА

В данной работе представлены теоретические основы экспериментального исследования алгоритмов планирования задач для вычислительного кластера, выполняемого с помощью симулятора кластера и его управляющей системы. Разработаны модели вычислительного кластера, его управляющей системы и вычислительной загрузки, а также система критериев и метрик сравнения алгоритмов планирования.

Ключевые слова: симулятор вычислительного кластера и его управляющей системы, планирование параллельных задач, высокопроизводительные вычисления, имитационное моделирование.

Введение

Проблема эффективного планирования параллельных задач, поступающих в управляющую систему вычислительного кластера, является актуальной, т. к. современные алгоритмы планирования не обеспечивают необходимого качества составляемых расписаний запуска задач. Довольно часто возникает ситуация, когда вычислительные узлы простаивают, хотя в очереди есть ожидающие своего исполнения параллельные программы.

Также следует отметить, что большинство современных управляющих систем вычислительного кластера используют алгоритмы планирования, не учитывающие топологию при размещении процессов задач на вычислительных узлах, а также их многопроцессорность. Учет данных факторов позволит увеличить эффективность алгоритмов планирования.

В связи с этим необходимо исследование существующих алгоритмов планирования и разработка новых более эффективных вариантов.

В данной работе представлены теоретические основы экспериментального исследования алгоритмов планирования задач для вычислительного кластера, выполняемого с помощью симулятора кластера и его управляющей системы.

Модель вычислительного кластера

Произвольная кластерная вычислительная система может быть описана с помощью ориентированного графа:

$$G_T = (P, K, E, b, c, m, d, s), \qquad (1)$$

где $P = \{p_1, p_2, ..., p_n\}$ – множество вычислительных узлов, $K = \{k_1, k_2, ..., k_z\}$ – множество коммутаторов, E – множество направленных сетевых связей между ними, $b : E \to Z_+ \cup \{0\}$ – отображение, характеризующее пропускную способность каждой сетевой связи в байтах в секунду. Функции $c, m, d : P \to Z_+$ определяют для каждого вычислительного узла p_i соответственно количество его вычислительных ядер $c(p_i) = C_i$, объемы оперативной $m(p_i) = M_i$ и дисковой памяти $d(p_i) = D_i$ в килобайтах. Отображение $s : P \to R_+$ для узла p_i задает относительную производительность $s(p_i) = S_i$ каждого его вычислительного ядра, которая определяет, во сколько раз ядра данного узла работают быстрее вычислительных ядер самого непроизводительного узла кластера.

Также в вычислительном кластере имеется выделенный узел p_0, на котором работает его управляющая система. Он не входит во множество P, но связан со всеми вычислительными узлами кластера с помощью выделенной управляющей сети.

Значения C_i, M_i, D_i и S_i являются статическими параметрами i-го узла вычислительного кластера. К числу его динамических характеристик относятся величины $u_i(t)$, $m_i(t)$ и $d_i(t)$, которые соответственно определяют загруженность его вычислительных ядер, объем доступной оперативной и дисковой памяти в момент времени $t \in [0; +\infty)$. Имеют место следующие неравенства:

$$\begin{aligned} 0 \le u_i(t) \le 1, \\ 0 \le m_i(t) \le M_i, \\ 0 \le d_i(t) \le D_i. \end{aligned} \qquad (2)$$

Пусть $X_i = \{\chi_{i,1}, \chi_{i,2}, ..., \chi_{i,C_i}\}$ – множество вычислительных ядер узла p_i, $X = \bigcup_{i=1}^{n} X_i$ – множество вычислительных ядер всех узлов кластера, тогда обозначим в качестве $id(\chi, t)$ номер задачи, исполняющейся на ядре $\chi \in X$ в момент времени t. Если же ядро χ в момент времени t было свободно, то $id(\chi, t) = 0$. Значения динамических параметров i-го узла в момент времени t могут быть вычислены по следующим формулам:

$$u_i(t) = \frac{\left|\{id(\chi,t) > 0 : \chi \in X_i\}\right|}{C_i},$$

$$m_i(t) = M_i - \sum_{\substack{\chi \in X_i, \\ j = id(\chi,t) > 0}} m_j, \qquad (3)$$

$$d_i(t) = D_i - \sum_{\substack{\chi \in X_i, \\ j = d(\chi,t) > 0}} d_j,$$

где m_j и d_j – соответственно объемы оперативной и дисковой памяти, необходимой каждому процессу j-й параллельной задачи. Данные формулы выведены, исходя из предположений, что мы не рассматриваем алгоритмы планирования с разделением времени и вычислительные узлы не имеют локальную загрузку, характерную для кластеров рабочих станций.

Структура типичного вычислительного SMP-узла изображена на рисунке 1. Вычислительные ядра могут относиться как к отдельным процессорам (по одному ядру на каждом), так и к нескольким многоядерным процессорам. С целью упрощения модели они рассматриваются как единое множество X_i. Процессы, выполняющиеся на вычислительных ядрах, могут напрямую считывать и записывать информацию в оперативную и дисковую память данного узла, взаимодействуя с ними через системную шину или коммутатор. При необходимости передачи сообщения другому вычислительному узлу задействуется сетевая подсистема, осуществляющая разбиение сообщения на пакеты и передачу их по высокопроизводительной коммуникационной сети. Сетевая подсистема каждого узла также содержит коммутатор, выполняющий функции маршрутизации собственных и транзитных пакетов.

Обозначим в качестве $N = P \cup K$ множество всех сетевых устройств системы. Сетевые связи множества $E \subseteq N \times N$ вместе с N образуют высокопроизводительную коммуникационную сеть, по которой процессы параллельных программ обмениваются сообщениями. Передача данных между управляющим узлом p_0 и вычислительными узлами совершается по отдельной управляющей сети и не мешает информационному обмену между процессами исполняющихся параллельных программ. Поэтому в рамках данной модели управляющая сеть не учитывается явно.

Исследуемые топологии современных кластерных вычислительных систем:

1. Толстые деревья. Рассматриваются блокируемые и неблокируемые варианты с двухуровневой организацией коммутаторов. Примеры известных кластеров, имеющих данную топологию: «СКИФ К-1000», «СКИФ МГУ», «IBM Roadrunner».

2. Торы. Исследуются варианты двумерных и трехмерных торов, а также k-арных n-кубов. Примеры: «СКИФ Аврора», «СКИФ К-500», «IBM BlueGene/P».

3. Звезды. Рассматриваются варианты с одним центральным коммутатором, к которому подключены все вычислительные узлы. Примеры: «Infinity», кластер ОГУ.

4. k-арные n-деревья. Представляют собой неблокируемые сети, состоящие из n уровней по k^{n-1} коммутаторов вида $k \times k$ в каждом.

Все рассматриваемые топологии могут быть представлены с помощью орграфа G_T.

Настоящая модель вычислительной системы предполагает использование принципа коммутации пакетов при передаче данных. Пусть g задает стандартный фиксированный размер одного пакета, а $h < g$ – размер его заголовочной части, тогда для передачи сообщения размера q потребуется $p = \left\lceil \frac{q}{g-h} \right\rceil$ пакетов.

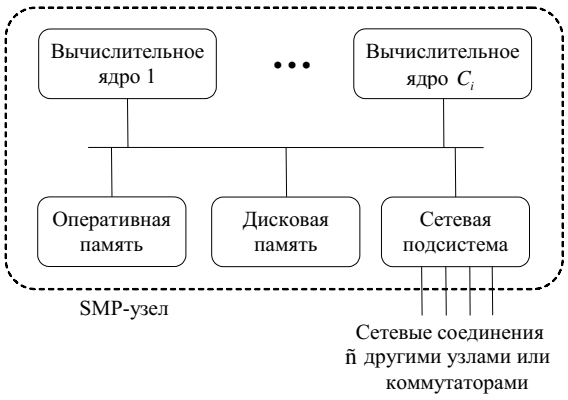

Рисунок 1. Структура SMP-узла

Время передачи сообщения от сетевого устройства n_{i_0} к n_{i_R} вдоль пути $\alpha = (n_{i_0}, n_{i_1}, ..., n_{i_R})$ может быть вычислено по формуле:

$$T_{перед}(\alpha, p) = \sum_{k=0}^{R-1} \frac{g}{b((n_{i_k}, n_{i_{k+1}}))} + \frac{g(p-1)}{\min\limits_{k=0,R-1} b((n_{i_k}, n_{i_{k+1}}))} + l_{марш} p(R+1), \quad (4)$$

где $l_{марш}$ – среднее время маршрутизации одного пакета, выполняемой каждым сетевым устройством. Заметим, что данная формула не учитывает возможные задержки, связанные с сетевой конкуренцией пакетов.

Также данная модель вычислительной системы предполагает использование алгоритмов статической и динамической маршрутизации, характерных для конкретных топологий вычислительной системы, в том числе алгоритмы кратчайших коммуникационных путей и покоординатной маршрутизации.

Перечислим основные возможности описанной модели вычислительной системы:

1. Формирование вычислительных кластеров однородных или гетерогенных по составу вычислительных узлов и строению коммуникационной сети.

2. Задание статических (узлы связаны друг с другом непосредственно) и динамических (применяются коммутаторы) сетей.

3. Описание полудуплексных и полнодуплексных сетевых соединений, а при добавлении в орграф G_T гиперребер – и шинных связей.

4. Формирование топологий большинства современных кластерных вычислительных систем.

В рамках данной работы прежде всего будут рассматриваться вычислительные кластеры с однородной коммуникационной сетью и полнодуплексными связями.

Модель управляющей системы вычислительного кластера

Управляющая система вычислительного кластера (УСВК) – программный пакет, выполняющий функции управления прохождением потока параллельных задач через кластерную систему. Типичная структура УСВК приведена на рисунке 2.

Основные компоненты УСВК [1]:

1. Очередь представляет собой накопитель задач пользователей, отправленных на запуск.

2. Планировщик на основе информации о задачах в очереди и сведений о состоянии узлов принимает решение относительно выбора очередной задачи для ее назначения на свободные вычислительные ядра узлов, удовлетворяющих требованиям.

3. Менеджер ресурсов собирает информацию о состоянии вычислительных узлов (их доступность, загруженность и пр.).

4. Агенты на вычислительных узлах выполняют три основные функции: запуск задач на вычислительных ядрах, контроль их выполнения и сбор информации о доступности ресурсов.

Пользователь взаимодействует с УСВК через специальную программу, с помощью которой он может поместить задачу в очередь, просмотреть состояние очереди, узнать статус своей задачи и т. п.

Основным процессом УСВК является диспетчеризация – автоматическая обработка множества заданий. Она включает: планирование (составление расписания для задач), доставку необходимых файлов на исполнительные узлы, мониторинг процесса выполнения и сбор его результатов.

Планировщик в процессе своей работы составляет расписание запуска параллельных задач из очереди в соответствии с заложенным в него онлайновым алгоритмом планирования. В его структуре обычно выделяют: алгоритм выбора следующей задачи из очереди и метод ее

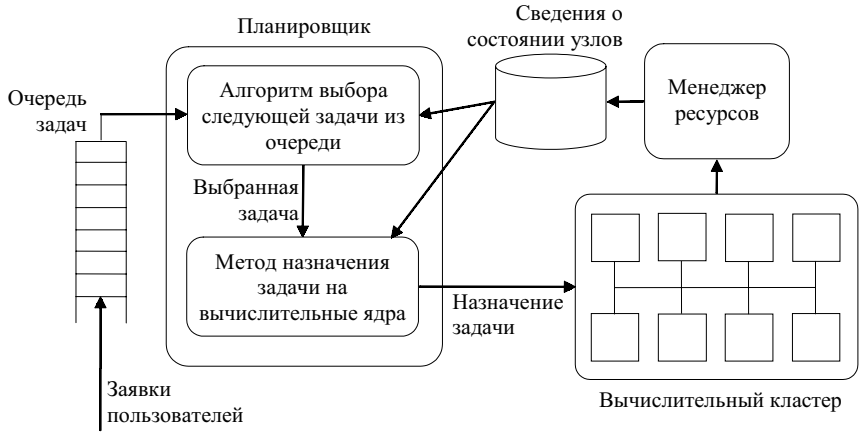

Рисунок 2. Структура управляющей системы вычислительного кластера

назначения на свободные ядра подходящих по ресурсам вычислительных узлов.

Цикл планирования представляет собой единичный акт планирования, в течение которого происходит выбор ожидающих задач и назначение им ресурсов в соответствии с алгоритмом. Он запускается каждый раз при наступлении одного из следующих событий: поступление в очередь новой задачи, завершение выполняющейся программы или запуск зарезервированной задачи.

Модель вычислительной загрузки кластера

Вычислительная загрузка кластера формируется потоком задач $J = (J_1,...,J_r)$, помещаемых пользователями в очередь его управляющей системы. Каждая задача представляет собой параллельную неинтерактивную программу, способную работать в пакетном режиме. Ее процессы запускаются планировщиком одновременно на всех выделенных вычислительных ядрах, во время работы они обмениваются сообщениями между собой. Ресурсы, выделенные задаче, освобождаются при завершении всех ее процессов.

Пользователь для задачи J_j указывает следующие требования к ресурсам кластера: количество необходимых вычислительных ядер n_j, объем оперативной mr_j и объем дисковой памяти dr_j в килобайтах, необходимой для исполнения каждого процесса на узлах, оценку $\tilde{t}_j$ в секундах времени выполнения задачи на узлах с единичной относительной производительностью. Следующая формула позволяет вычислить t_j – оценку времени выполнения задачи с учетом производительности выделенных ей планировщиком вычислительных узлов:

$$t_j = \frac{\tilde{t}_j}{\min\limits_{i \in I_j} S_i}, \qquad (5)$$

где I_j – множество номеров узлов, ядра которых были отданы задаче.

Кроме описанных выше характеристик каждая задача также имеет следующие параметры, необходимые для целей симуляции: a_j – время ее поступления в очередь, $\tau_j \in (0; \tilde{t}_j]$ – время, затрачиваемое задачей только на вычисления (без сетевых коммуникаций) при условии единичной относительной производительности всех назначенных ей вычислительных узлов.

В рамках данной модели предполагается, что пользователь, делая оценку времени выполнения задачи, возможно, допускает ошибку, переоценивая ее по сравнению с реальным временем выполнения. Противоположный случай ошибок не рассматривается, т. к. управляющая система будет принудительно завершать задачи, превысившие лимит выделенного им времени.

Структура программы каждой параллельной задачи J_j может быть представлена следующей SPMD-моделью:

Process *u*:
for $i = 1$ to q_j do
{
Communication(*i,j*);
Computation(*i,j*);
}

Выполнение каждого процесса параллельной задачи представляет собой чередование фаз коммуникации и вычислений.

Коммуникационная часть каждой итерации может быть описана с помощью коммуникационного паттерна – ориентированного взвешенного графа следующего вида:

$$CP_{ij} = (\Pi_j, p_{ij}), \qquad (6)$$

где $\Pi_j = \{\pi_1, \pi_2, ..., \pi_{n_j}\}$ – множество процессов задачи, $p_{ij} : \Pi_j \times \Pi_j \to Z_+ \cup \{0\}$ – функция, определяющая вес каждой дуги, равный размеру в пакетах передаваемого сообщения. $p_{ij}(u, v)$ равно количеству пакетов сообщения, передаваемого от процесса u к процессу v на i-й итерации SPMD.

Пусть $pred_{ij}(u) = \{v \in \Pi_j : p_{ij}(v, u) > 0\}$ – множество предшественников процесса u в орграфе (6), $succ_{ij}(u) = \{v \in \Pi_j : p_{ij}(u, v) > 0\}$ – множество его последователей. Выполнение фазы Communication (*i,j*) для каждого процесса u заключается в том, что сначала происходит неблокируемая рассылка сообщений всем процессам множества $succ_{ij}(u)$, а затем выполняется блокируемый прием всех сообщений от процессов $pred_{ij}(u)$. Заметим, что остальные случаи исключены из рассмотрения, т. к. сочетания блокируемых рассылок с неблокируемыми или блокируемыми приемами сообщений могут приводить к взаимоблокировкам, а случай неблокируемых рассылок и неблокируемых приемов не представляется интересным.

Обозначим $CP_j = \{CP_{1j}, CP_{2j}, ..., CP_{q_j j}\}$ – множество всех коммуникационных паттернов задачи J_j. Можно выделить два основных спосо-

ба его задания: случайная генерация и моделирование реальных программ, например эталонных тестов, реализации быстрого преобразования Фурье, алгоритма параллельного умножения матриц и др.

Типичные коммуникационные паттерны, встречаемые в научной литературе [2, 3]:

1. Random. Каждый процесс пересылает сообщение одному другому случайно выбранному процессу.

2. Pairs. Процессы случайно разбиваются на пары и обмениваются сообщениями между собой.

3. Ring. Процессы объединяются в кольцо. Каждый процесс посылает сообщения следующему и принимает данные от предыдущего.

4. One-to-all. Один случайно выбранный процесс рассылает сообщения всем остальным.

5. All-to-all. Каждый процесс отправляет сообщения остальным процессам.

В проводимом исследовании будут отдельно рассматриваться случаи использования каждого типичного коммуникационного паттерна.

Заметим, что в рамках настоящей модели рассматриваются только коммуникации вида «точка - точка». Моделировать коллективные операции достаточно сложно, т. к. способ их выполнения в виде совокупности точечных операций зависит от нескольких факторов: используемых алгоритмов, характеристик вычислительной системы и передаваемых сообщений. В частности, коллективные операции MPI поразному реализованы в библиотеках разных производителей, а также в разных версиях библиотеки одного производителя. В будущем планируется расширение данной модели за счет учета коллективных операций.

Пусть функция $\gamma_j : \Pi_j \to P$ позволяет определить для каждого процесса задачи J_j вычислительный узел, на ядро которого он был назначен планировщиком. Для простоты положим, что каждый процесс тратит одинаковое время на выполнение вычислительной части каждой итерации SPMD-модели задачи. Тогда время выполнения вычислительной фазы Computation(*i,j*) процессом u можно вычислить по формуле:

$$T_{\text{вычисл.ij}}(u) = \frac{\tau_j}{q_j s(\gamma(u))}. \qquad (7)$$

Пусть $T_{\text{комм.ij}}(u)$ – время выполнения коммуникационной фазы Communication(*i,j*) процессом u задачи J_j, тогда реальное время выполнения данной задачи T_j может быть вычислено по формуле:

$$T_j = \max_{u \in \Pi_j} \sum_{i=1}^{q_j} (T_{комм.ij}(u) + T_{вычисл.ij}(u)) =$$

$$= \max_{u \in \Pi_j} \{ \sum_{i=1}^{q_j} T_{комм.ij}(u) + \frac{\tau_j}{s(\gamma(u))} \}. \qquad (8)$$

Величина $T_{\text{комм.ij}}(u)$ зависит от сетевой конкуренции и от физического расположения процессов на вычислительных узлах, ее значение может быть вычислено в процессе симуляции работы вычислительного кластера и его управляющей системы. Заметим, что в рамках данной модели мы в силу незначительности пренебрегаем временем, которое тратится на разбиение сообщения на пакеты и его сборку из них.

Симуляция работы вычислительного кластера и его управляющей системы предполагает формирование потока задач. Все описанные выше количественные параметры генерируются на основе определенных законов распределений случайных величин, подобранных в результате анализа реальных трасс, собираемых на кластерных вычислительных системах (см. таблицу 1). Более подробную информацию можно найти в статье [4].

Система критериев и метрик сравнения алгоритмов планирования

С целью учета всех аспектов эффективности работы алгоритмов планирования задач для

Таблица 1. Законы распределений параметров модели

Параметр	Используемый закон распределения
$\log n_j$	двухэтапное равномерное распределение с выделением классов последовательных и 2^k-задач
$\log_{\ j}$	гипергамма-распределение, параметр p которого линейно зависит от n_j
$\tilde{t}_j$	равномерное распределение на отрезке $[\tilde{}_j; 2\tilde{}_j]$
$a_{j\,1}\ \ a_j$	экспоненциальное распределение
mr_j, dr_j	равномерное распределение
$p_{ij}(u, v)$	экспоненциальное распределение для всех дуг, существующих в CP_{ij}

кластерных вычислительных систем была построена система критериев и метрик их сравнения. Она включает следующие критерии:

1. Производительность расписаний. Содержит метрики: средняя загруженность вычислительных ядер узлов кластера $U_{ядер}$, потеря производительности кластера CL, максимальное время завершения задачи C_{max}, среднее время ожидания задач в очереди $\bar{t}_{ожс}$ и среднее ограниченное замедление задач $\bar{s}_{огр}$. Метрики $U_{ядер}$, CL и C_{max} характеризуют непосредственно производительность расписания, поэтому они более приоритетны для исследования, чем $\bar{t}_{ожс}$ и $\bar{s}_{огр}$, которые имеют косвенный характер.

2. Используемость оперативной и дисковой памяти. Включает метрики $\overline{m}$ и $\overline{d}$, определяющие соответственно средние загруженности оперативной и дисковой памяти вычислительных узлов. Позволяет выявить процент неиспользованных ресурсов.

3. Сбалансированность загрузки узлов. Определяет метрики Du, Dm и Dd, обозначающие соответственно дисперсии загруженности ядер, оперативной и дисковой памяти вычислительных узлов. Демонстрирует честность алгоритма планирования по отношению к узлам.

4. Гарантированность обслуживания задач. Включает метрики максимального времени ожидания задач в очереди $t_{ож\ max}$ и максимального ограниченного замедления задач $s_{огр\ max}$. Данный критерий имеет особую важность в случае, если кластер используется в рамках системы реального времени.

5. Честность по отношению к задачам. Содержит метрику $Dt_{ож}$ – дисперсию времени ожидания задач в очереди. Позволяет оценить степень равноправности задач с точки зрения алгоритма планирования.

6. Коммуникационный критерий. Включает следующие метрики: среднее суммарное расстояние между вычислительными ядрами, назначенными процессам параллельных задач, средняя дисперсность задач $\overline{JD}$ и среднее время блокировки сообщений $\overline{MB}$. Данный критерий позволяет оценить сетевую конкуренцию и степень фрагментации параллельных задач.

При исследовании различных алгоритмов планирования нас прежде всего интересует производительность составляемых ими расписаний. Поэтому при анализе алгоритмов в первую очередь будет рассматриваться первый критерий системы, остальные – имеют вторичный характер.

Анализ алгоритмов планирования по определенному критерию представляет собой сравнение для различных алгоритмов графиков зависимостей метрик этого критерия от величины системной загрузки L и выбор лучшего варианта для того или иного случая. Системная загрузка определяется как отношение суммарного объема вычислительной работы всех задач к объему работы, который кластер может потенциально выполнить (при полной загруженности) за время до появления в управляющей системе последней задачи.

Исследуемые алгоритмы планирования

К числу алгоритмов планирования, которые будут экспериментально исследованы с помощью симулятора вычислительного кластера и его управляющей системы, относятся сочетания алгоритмов выбора задачи из очереди Most Processors First Served Scan и Backfill со следующими методами назначения, учитывающими топологию вычислительной системы [2, 3, 5–7]:

– Paging, Multiple Buddy System, Adaptive None-Contiguous Allocation, Greedy Available Busy List, Minimizing Contention, Minimizing Contention Incremental, Manhattan Median, NEP (для топологий торов);

– Contiguous Allocation, Quasi-Contiguous Allocation (k-арные n-деревья);

– алгоритмы сортировки узлов (толстые деревья, звезды);

– модифицированный вариант алгоритма Minimizing Contention (для произвольных топологий).

А также со следующими методами назначения, не принимающими ее во внимание [4]: First Fit, Best Fit, Fastest Node Fist, Random First.

Заключение

В данной работе приведены модели вычислительного кластера, его управляющей системы и вычислительной загрузки, сформирована система критериев и метрик сравнения алгоритмов планирования. На их основе будет разработан симулятор вычислительного кластера и управляющей системы, учитывающий его топологию и многопроцессорность узлов. Также определен набор алгоритмов планирования, которые будут исследованы с помощью симулятора.

12.05.2010

Список использованной литературы:
1. Топорков В.В. Модели распределенных вычислений. М.: ФИЗМАТЛИТ, 2004. – 320 с.
2. Moore S.Q., Lionel M.N. The Effects of Network Contention on Processor Allocation Strategies // Proceedings of the 10th International Parallel Processing Symposium. – Washington, DC: EEE Computer Society, 1996. – P. 268-273.
3. Bani-Mohammad S., Ould-Khaoua M., Abaneh, I. An efficient processor allocation strategy that maintains a high degree of contiguity among processors in 2D mesh connected multicomputers // Proceedings of ACS/IEEE International Conference on Computer Systems and Applications, AICCSA. – 2007. – P. 934-941.
4. Полежаев П.Н. Исследование алгоритмов планирования параллельных задач для кластерных вычислительных систем с помощью симулятора // Параллельные вычислительные технологии (ПАВТ'2010): Труды международной конференции. – Челябинск: ЮУрГУ, 2010 г. – С. 287-298.
5. Bender M.A., Bunde D.P., Demaine E.D. Communication-Aware Processor Allocation for Supercomputers // Lecture Notes in Computer Science. V. 3608/2005. – Berlin: Springer, 2005. – P. 169-181.
6. Cheng C., Mohapatra P. Improving Performance of Mesh-connected Multicomputers by Reducing Fragmentation // Journal of Parallel and Distributed Computing. V. 52(1), 1998. – P. 40-68.
7. Pascual1 J.A., Navaridas J., Miguel-Alonso J. Effects of Topology-Aware Allocation Policies on Scheduling Performance // Lecture Notes in Computer Science. V. 5798/2009. – Berlin: Springer, 2009. – P. 138-156.

Исследования выполнены при поддержке Федерального агентства по образованию в рамках реализации ФЦП «Научные и научно-педагогические кадры инновационной России» на 2009–2013 гг. (государственный контракт №П2039).

Сведения об авторах:
Гергель Виктор Павлович, декан факультета вычислительной математики и кибернетики Нижегородского государственного университета им. Лобачевского, доктор технических наук, профессор
603950, г. Нижний Новгород, пр-т Гагарина, 23, тел. (831) 4654859, e-mail: gergel@unn.ru

Полежаев Петр Николаевич, аспирант кафедры математического обеспечения информационных систем математического факультета Оренбургского государственного университета, 460018, г. Оренбург, пр-т Победы, 13, тел. (3532) 372534, e-mail: peter.polezhaev@mail.ru

Gergel V.P., Polezhaev P.N.
The theoretical bases of experimental study of algorithms of planning tasks for the computational cluster with help of simulator
This work represented the theoretical bases of an experimental study of the algorithms of planning tasks for the computational cluster, carried out with help of the simulator of cluster and its manager of system. The authors developed the models of computational cluster, its manager of system and computational load, and also the system of criteria and certificates of the comparison of the algorithms of planning.
The key words: the simulator of computational cluster and its manager of system, planning parallel tasks, highly productive calculations, imitation simulation.

Bibliography:
1. Toporkov V.V. Models of distributed computation. M.: FIZMATLIT, 2004. – 320 c.
2. Moore S.Q., Lionel M.N. The Effects of Network Contention on Processor Al-location Strategies // Proceedings of the 10th International Parallel Processing Symposium. – Washington, DC: EEE Computer Society, 1996. – P. 268-273.
3. Bani-Mohammad S., Ould-Khaoua M., Abaneh, I. An efficient processor allo-cation strategy that maintains a high degree of contiguity among processors in 2D mesh connected multicomputers / / Proceedings of ACS / IEEE International Conference on Computer Systems and Applications, AICCSA. – 2007. – P. 934-941.
4. Polezhaev P.N. Study of parallel task scheduling algorithm for cluster computing systems using a simulator / Parallel Computing Technologies (PAVT'2010): Proceedings of the International Conference. – Chelyabinsk: South Ural State University, 2010 – S. 287-298.
5. Bender M.A., Bunde D.P., Demaine E.D. Communication-Aware Processor Allocation for Supercomputers // Lecture Notes in Computer Science. V. 3608/2005. – Berlin: Springer, 2005. – P. 169-181.
6. Cheng C., Mohapatra P. Improving Performance of Mesh-connected Multi-computers by Reducing Fragmentation // Journal of Parallel and Distributed Computing. V. 1952 (1), 1998. – P. 40-68.
7. Pascual1 J.A., Navaridas J., Miguel-Alonso J. Effects of Topology-Aware Al-location Policies on Scheduling Performance // Lecture Notes in Computer Science. V. 5798/2009. – Berlin: Springer, 2009. – P. 138-156.