Планирование задач для вычислительного кластера с учетом сети и многопроцессорности узлов*

П.Н. Полежаев

Оренбургский государственный университет

В данной работе приводятся результаты исследования предложенных и существующих алгоритмов планирования задач для вычислительного кластера с учетом сети и многопроцессорности узлов. Исследование проводилось с помощью разработанного симулятора вычислительного кластера и его управляющей системы, в основу которого легла предложенная модель вычислительной системы, модель управляющей системы, имитационная схема работы кластера, а также модель загрузки системы потоком задач.

1. Введение

Алгоритмы планирования задач, используемые в существующих управляющих системах вычислительных кластеров, демонстрируют неплохую эффективность работы. Тем не менее, имеется возможность дальнейшего увеличения их производительности за счет учета топологии вычислительной системы и многопроцессорности вычислительных узлов.

Действительно, если алгоритм планирования будет назначать процессы параллельной задачи на топологически близкие вычислительные ядра, то это приведет к снижению времени ее исполнения в силу сокращения коммуникационных задержек при передаче данных между ее процессами. Что, в свою очередь, увеличивает производительность всей вычислительной системы.

Учет многопроцессорности вычислительных узлов при планировании также положительно сказывается на показателях работы кластерной системы, т.к. время выполнения коммуникационных операций между процессами параллельной программы, исполняющимися на соседних процессорах (или ядрах) одного узла, гораздо меньше, чем между процессами, исполняющимися на разных узлах.

Кроме того, при назначении параллельных программ на свободные вычислительные ядра необходимо учитывать сетевую конкуренцию между процессами одновременно исполняющихся задач. Ее снижение приводит к уменьшению времени выполнения сетевых коммуникаций, а это ведет к росту производительности кластерной системы.

Целью настоящего исследования является разработка эффективных алгоритмов планирования задач для высокопроизводительных кластерных систем, учитывающих топологию, сетевую конкуренцию и многопроцессорность вычислительных узлов.

2. Модель вычислительного кластера

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

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

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 соответ-

* Исследования выполнены при поддержке Министерства образования и науки Российской Федерации в рамках реализации ФЦП «Научные и научно-педагогические кадры инновационной России» на 2009-2013 гг. (государственные контракты №14.740.11.0287, №14.740.11.0689).

ственно 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 $t \in [0;+\infty)$.

Пусть $\mathrm{X}_i = \{\chi_{i,1}, \chi_{i,2}, ..., \chi_{i,C_i}\}$ – множество вычислительных ядер узла p_i, $\mathrm{X} = \bigcup_{i=1}^{n} \mathrm{X}_i$ – множество вычислительных ядер всех узлов кластера, тогда обозначим в качестве $id(\chi, t)$ номер задачи, исполняющейся на ядре $\chi \in \mathrm{X}$ 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:

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

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

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

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.

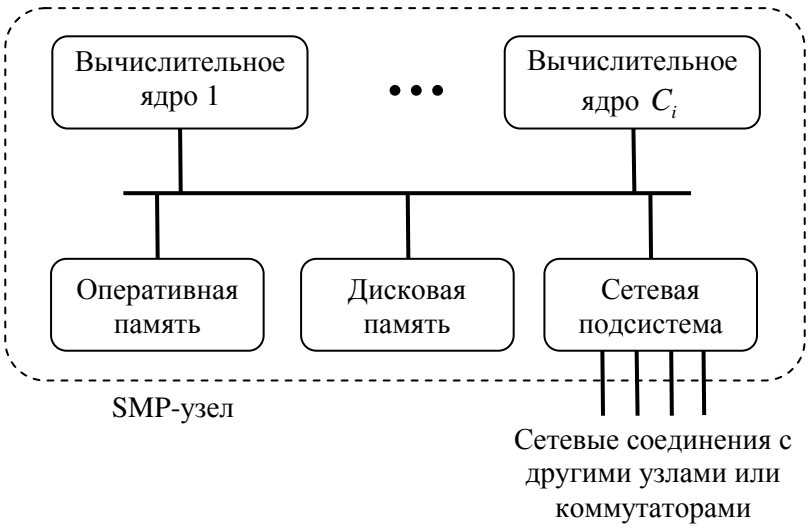

Рис. 1. Структура SMP-узла

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.

С помощью данной модели можно описать вычислительную систему произвольной топологии, в частности, в рамках данной работы она используется для задания вычислительных кластеров топологии толстого дерева, двумерного тора и звезды с однородными и неоднородными узлами.

3. Модель управляющей системы вычислительного кластера

В рамках настоящего исследования рассматривается классическая архитектура управляющей системы вычислительного кластера (УСВК), предполагающая наличие следующих основных компонент (см. рисунок 2):

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.).

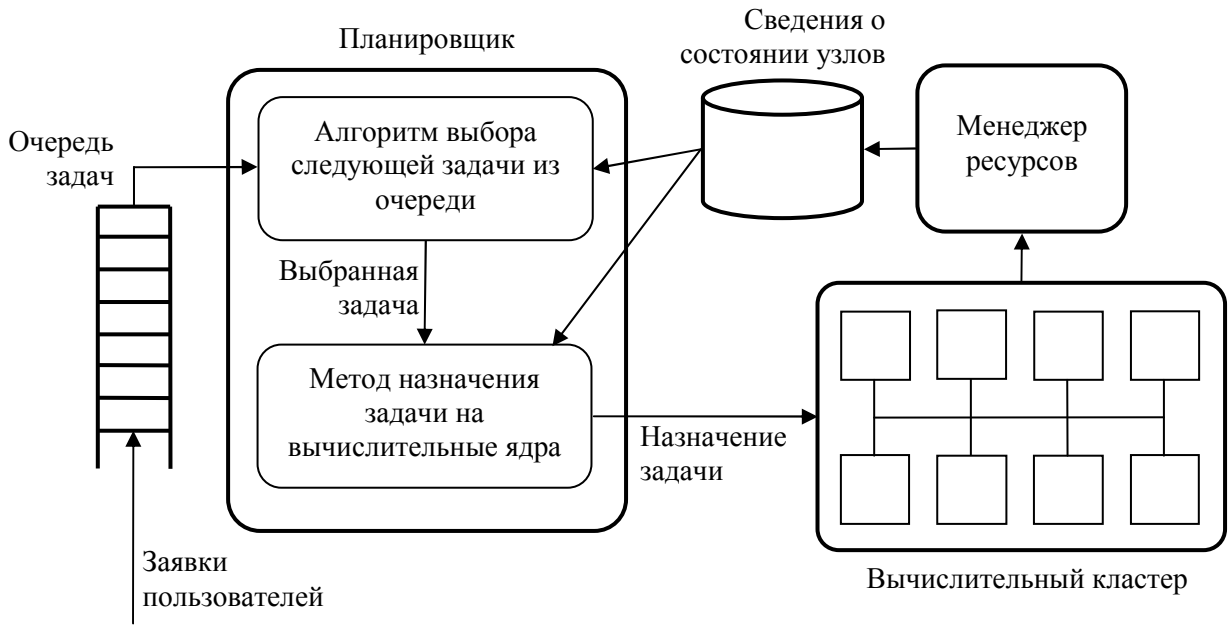

Рис. 2. Структура управляющей системы вычислительного кластера

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

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 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.

На рисунке 3 представлена обобщенная схема работы всех рассматриваемых в настоящем исследовании алгоритмов планирования.

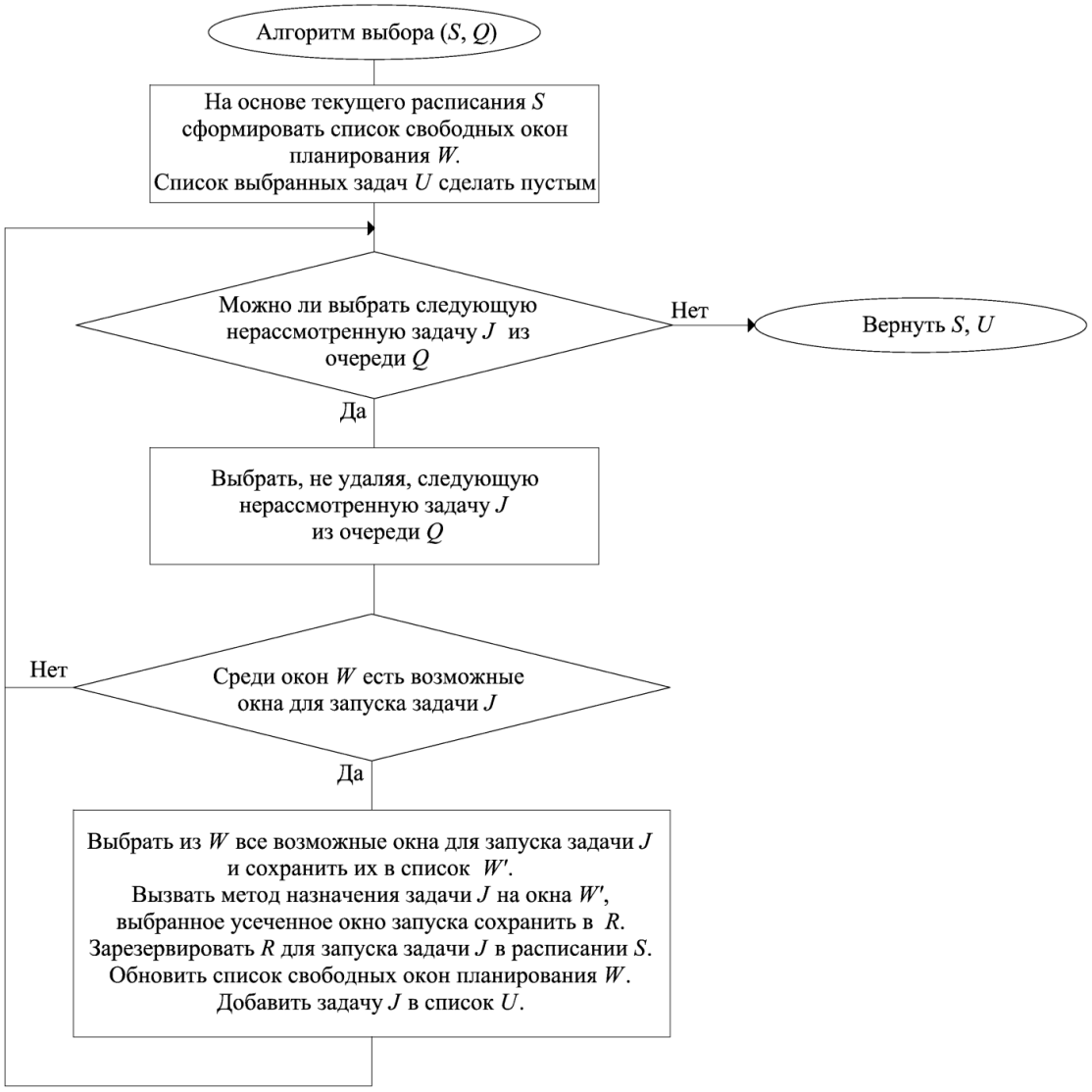

Рис. 3. Обобщенная блок-схема работы алгоритма планирования

При каждом запуске цикла планирования алгоритм выбора следующей задачи из очереди на основе текущего расписания $Schedule$ и очереди ожидающих задач Q динамически формирует на момент времени t список окон планирования WL. Затем выполняется цикл, на каждой итерации которого по некоторому принципу из Q выбирается очередная задача J, для которой на основе списка окон планирования WL составляется список подходящих окон WL', который затем передается методу назначения. В процессе своей работы последний формирует для J окно запуска R, на основе которого обновляются WL и $Schedule$. Итогом работы алгоритма является обновленное расписание $Schedule$ и список выбранных задач U.

Итоговое расписание после запуска всех задач из очереди Q может быть представлено в виде множества упорядоченных пар $Schedule = \{(J_j, W_j)\}_{j=\overline{1,m}}$, где W_j – окно запуска работы J_j.

4. Исследуемые алгоритмы планирования задач на вычислительном кластере

В рамках проводимого исследования были рассмотрены алгоритмы планирования задач, представляющие собой сочетания алгоритма выбора следующей задачи из очереди Most Processors First Served Scan (MPFS Scan) или алгоритма обратного заполнения Backfill с методами ее назначения на вычислительные ядра, учитывающими топологию вычислительной системы [1, 2]: для топологии толстого дерева – Sorting Nodes by Performance (SNP), Fat Tree Sorting Commutators by Performance (FTSCP), Fat Tree Sorting Commutators by Cores (FTSC); для топологии двумерного тора – модифицированные варианты алгоритмов Minimizing message-passing Contention 1x1 (MC1x1) и Minimizing message-passing Contention 1x1 Incremental (MC1x1+Inc); для топологии звезды – Sorting Nodes by Performance (SNP) и Sorting Nodes by Speed (SNS).

Также рассматривались сочетания алгоритмов MPFS Scan и Backfill с методами назначения, не принимающими топологию во внимание [2, 3]: First Fit (FF), Best Fit (BF), Fastest Node Fist (FNF) и Random First (RF).

В ходе исследования были предложены собственные алгоритмы планирования, представляющие собой сочетания алгоритма MPFS Scan или Backfill с разработанными методами назначения Summed Distance Minimization (SDM) и Maximum Distance Minimization (MDM). Данные методы учитывают топологию вычислительной системы, сетевую конкуренцию между одновременно исполняющимися задачами и многопроцессорность вычислительных узлов.

Пусть методу назначения передается выбранная из очереди задача J_j, требующая для своего исполнения n_j вычислительных ядер. Далее опишем принцип работы алгоритмов SDM и MDM.

Метод назначения SDM для каждого допустимого окна планирования $W \in WL'$ перебирает все сетевые устройства d кластера (коммутаторы и узлы) и определяет для каждого из них n_j близжайших вычислительных ядер окна W, подходящих по конфигурации для задачи J_j. Эти вычислительные ядра формируют возможное окно запуска $R_{W,d}$ задачи J_j. Для назначения в качестве результата R выбирается окно $R_{W,d}$ с минимальным суммарным попарным расстоянием, если таких несколько, то выбирается окно с большей суммарной производительностью вычислительных ядер.

Данный метод является обобщением алгоритма Manhattan Median для случая произвольных топологий. За счет компактного размещения процессов параллельной задачи алгоритм SDM пытается минимизировать суммарное попарное расстояние между выделенными ей вычислительными ядрами, это позволяет снизить сетевую конкуренцию между одновременно выполняющимися в системе задачами и уменьшить степень распределения процессов параллельных задач по вычислительной системе.

Метод назначения MDM для каждого допустимого окна планирования $W \in WL'$ перебирает все сетевые устройства d кластера и для каждого из них запускает алгоритм поиска в ширину. В процессе его работы формируется множество достигнутых вычислительных ядер узлов окна W, которые подходят по конфигурации для задачи J_j, до тех пор, пока не будет получено n_j ядер. При этом среди ядер, находящихся на одинаковом расстоянии от начального сетевого устройства в первую очередь выбираются те, которые имеют большую скорость. Результатом работы поиска в ширину является сформированное возможное окно запуска $R_{W,d}$. Для назначения задаче J_j в качестве результата R выбирается окно $R_{W,d}$ с минимальным максимальным расстоянием от первоначального сетевого устройства d.

Данный метод является аналогом алгоритма MC1x1 для произвольных топологий. За счет компактного размещения процессов параллельной задачи алгоритм MDM пытается минимизировать максимальное расстояние от выбранного центрального сетевого устройства до назначенных задаче вычислительных ядер.

5. Симулятор вычислительного кластера и его управляющей системы

Для экспериментального сравнительного исследования эффективности работы предложенных методов назначения с существующими вариантами использовался созданный в рамках настоящей работы симулятор вычислительного кластера и его управляющей системы TopSimity. Он был реализован на языке программирования C++ с использованием стандартной библиотеки шаблонов STL. Получено свидетельство о его регистрации в ФГУ ФИПС [4].

Имитационная схема работы симулятора представлена на рисунке 4. Источник I генерирует поток параллельных задач, отправляемых пользователями в очередь Q управляющего узла p_0. Канал S представляет собой планировщик, который в соответствии с заложенным в него алгоритмом осуществляет извлечение задач из Q и назначение их на свободные вычислительные ядра подходящих по конфигурации узлов.

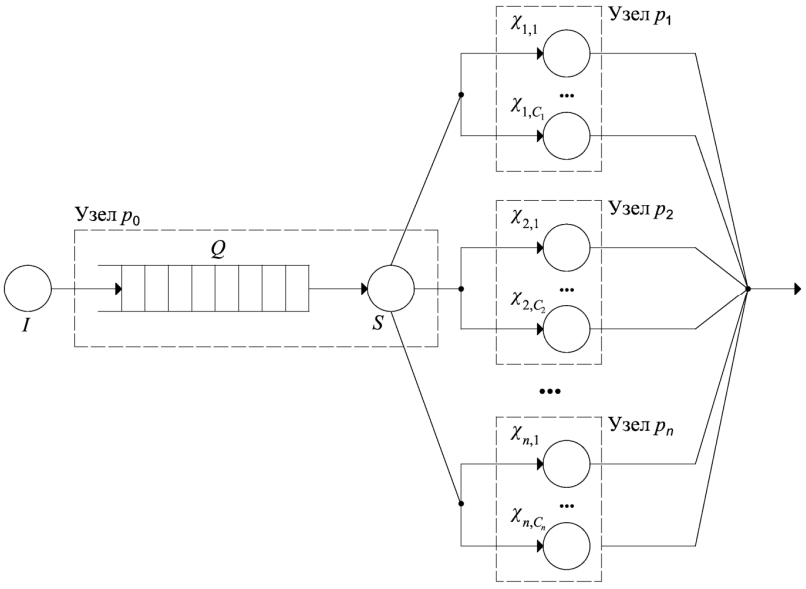

Рис. 4. Имитационная схема управляющей системы вычислительного кластера

Схема работы вычислительного узла p_i приведена на рисунке 5 p_i соединен с другими узлами и коммутаторами вычислительной сети с помощью r_i дуплексных связей. Все входящие пакеты, а также пакеты сообщений, генерируемых процессами, выполняющимися на

локальных вычислительных ядрах, сначала поступают в очередь $Q_{inp.i}$, а затем маршрутизируются каналом обслуживания R_i. Если соответствующий пакет предназначен для локального вычислительного ядра, то он передается ему непосредственно, иначе – помещается в одну из очередей $Q_{out.i,1}, Q_{out.i,2}, \ldots, Q_{out.i,r_i}$, соответствующую выбранной алгоритмом маршрутизации исходящей связи. При выполнении маршрутизации каналом R_i моделируется временная задержка величиной $l_{routing}$. Имитационная схема коммутатора представляет собой урезанный вариант имитационной схемы узла. Отличие только в том, что коммутатор не может выполнять вычисления, и поэтому у него отсутствуют вычислительные ядра.

На рисунке 5 приведена схема работы дуплексной связи $L_{ij} = \{E_{ij}, E_{ji}\}$, соединяющей сетевые устройства $d_i \in D$ и $d_j \in D$. Каналы обслуживания E_{ij} и E_{ji} добавляют к времени передачи пакета задержку величиной $b(E_{ij}) = b(E_{ji})$.

В качестве алгоритма работы симулятора используется алгоритм моделирования по принципу особых состояний (событий).

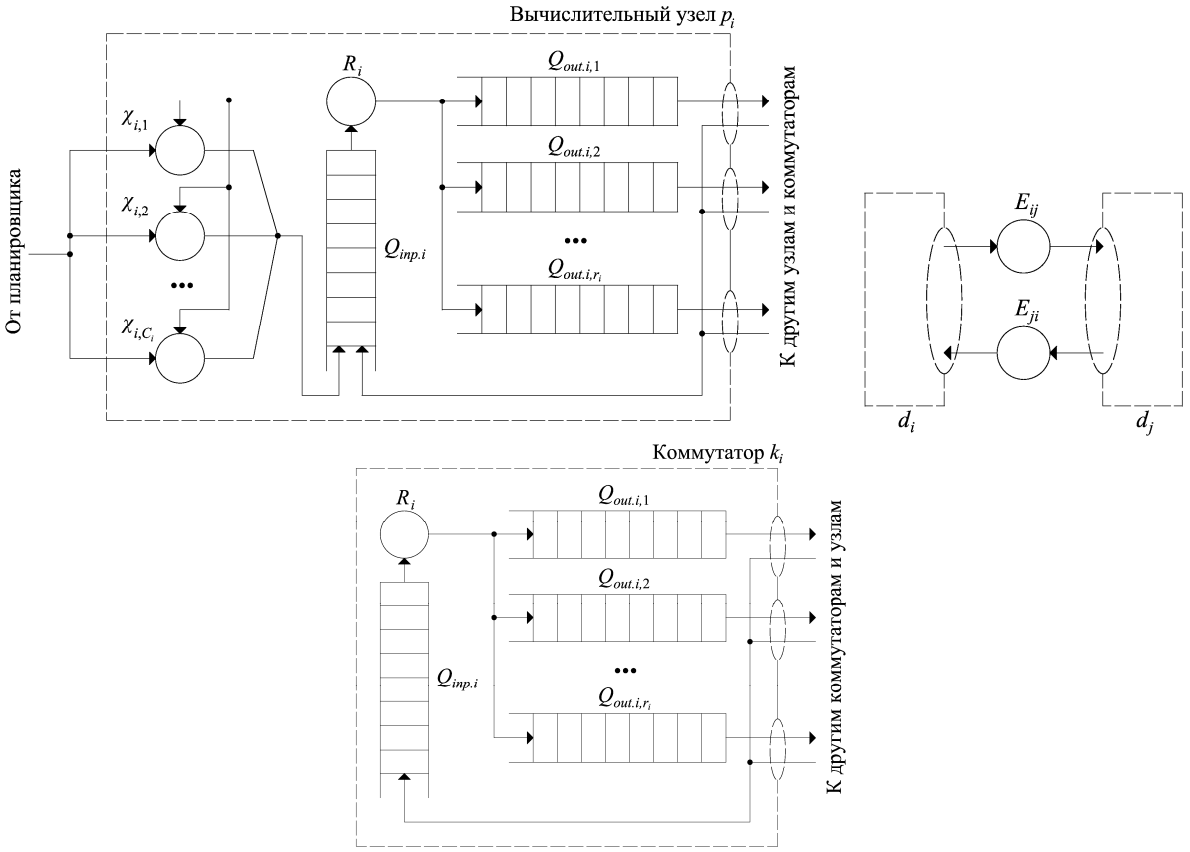

Рис. 5. Имитационная схема узла p_i, коммутатора k_i и дуплексной сетевой связи $L_{ij} = \{E_{ij}, E_{ji}\}$

Вычислительная загрузка кластера моделируется 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 па-

мяти 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:

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

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:

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

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 способа 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 [5–7]:

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 рамках проводимого исследования применял коммуникационный паттерн All-to-all, т.к. он обеспечивает максимальную загрузку вычислительной сети и сетевую конкуренцию. В дальнейшем планируется рассмотреть другие возможные варианты.

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 по-разному 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:

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

Пусть $T_{комм.ij}(u)$ – 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:

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

Величина $T_{комм.ij}(u)$ 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.

Таблица 1. Законы распределений параметров модели

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

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 статьях [3, 5].

Данная модель вычислительной загрузки позволяет генерировать типичные потоки задач, поступающие в очереди управляющих систем большинства современных вычислительных кластеров.

6. Результаты экспериментального исследования

Исследование проводилось для двух групп сценариев: вычислительный кластер с однородными узлами и однородной высокопроизводительной сетью и вычислительный кластер с неоднородными узлами и однородной сетью.

В первом случае моделировался кластер из 12 узлов, каждый из которых имеет 2 вычислительных ядра, 2 Гб оперативной памяти, 40 Гб локальной дисковой памяти, единичную относительную вычислительную скорость. Во втором – использовалась конфигурация из 12 узлов с суммарным количеством вычислительных ядер 24, разными объемами оперативной и дисковой памяти, разной относительной вычислительной скоростью.

Каждая группа сценариев рассматривалась для трех топологий: толстое дерево (два 6-портовых корневых коммутатора и три 8-портовых листовых коммутатора (4 восходящих портов и 4 узловых портов на каждом)), звезда (один 12-портовый коммутатор), двумерный тор размера 3 x 4 узла.

Оценка эффективности работы алгоритмов планирования осуществлялась с помощью системы количественных критериев и метрик, которая охватывает все аспекты состояния вычислительной системы: производительность, сбалансированность загруженности вычислительной системы, используемость памяти вычислительных узлов, гарантированность обслуживания задач, честность по отношению к задачам, характер распределения процессов параллельных задач по вычислительной системе и сетевую конкуренции между ними.

В частности критерий производительности включает следующие метрики: средняя загруженность вычислительных ядер узлов кластера U_{cores}, потеря производительности кластера CL, максимальное время завершения задачи C_{max}, среднее время ожидания задач в очереди $\overline{T}_{wait}$ и среднее ограниченное замедление задач $\overline{s}_{lim}$.

Для получения достоверных результатов симуляция проводилась в каждом сценарии 100 раз для различных потоков из 500 случайно сгенерированных задач. Все вычисляемые значения метрик усреднялись по всем потокам.

Сравнение алгоритмов планирования осуществлялось путем построения графиков зависимости метрик от величины системной загрузки L. Например, на рисунке 6 изображены графики зависимости метрик производительности U_{cores} и $C_{\max}$ от L для сценария кластера топологии толстого дерева с неоднородными вычислительными узлами.

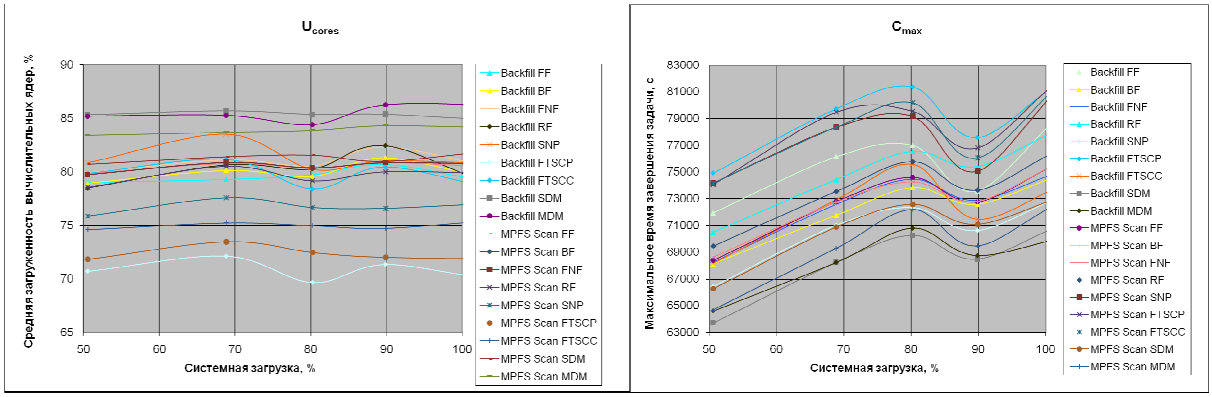

Рис. 6. Графики зависимости значений метрик производительности U_{cores} и $C_{\max}$ от величины системной загрузки L

Заметим, что алгоритмы планирования в первую очередь сравнивались по критерию производительности, остальные критерии носили вторичный характер.

Полученные результаты экспериментального исследования сведены в таблицу 2. Для каждого сценарии в зависимости от топологии системы и величины системной загрузки L может быть рекомендован свой алгоритм планирования, представляющий собой сочетания алгоритма выбора следующей задачи из очереди Backfill с предложенными методами ее назначения на вычислительные ядра MDM или SDM.

Таблица 2. Алгоритмы планирования, показавшие лучшие результаты в каждом сценарии

Группа сценариев	Топология	Наилучшие алгоритмы
Неоднородные вычислительные узлы и однородная сеть	Толстое дерево	Backfill SDM (при $L < 85$-93%), Backfill MDM (при $L \geq 85$-93%)
	Двумерный Тор	Backfill SDM (при $L < 67$-71%), Backfill MDM (при $L \geq 67$-71%)
	Звезда	Backfill SDM, Backfill MDM
Однородные вычислительные узлы и однородная сеть	Толстое дерево	Backfill SDM (при $L < 75$-79%), Backfill MDM (при $L \geq 75$-79%)
	Двумерный Тор	Backfill SDM (при $L < 71$-73%), Backfill MDM (при $L \geq 71$-73%)
	Звезда	Backfill SDM, Backfill MDM, Backfill SNP

7. Заключение

В рамках данной работы была предложена имитационная схема кластера, разработана модель вычислительной системы, модель его управляющей системы, модель вычислительной загрузки потоком задач. Все они легли в основу симулятора вычислительного кластера и его управляющей системы – основного инструмента настоящего исследования. Он позволяет проводить сравнительное экспериментальное исследование алгоритмов планирования с учетом сети и многопроцессорности вычислительных узлов.

Проведенное исследование показало эффективность работы алгоритмов планирования, представляющих собой сочетания алгоритма Backfill с предложенными методами назначения MDM и SDM.

В дальнейшем планируется реализация данных алгоритмов в рамках УСВК Torque и их апробация на реальных потоках разнообразных вычислительных задач, характерных для современных кластерных систем.

Литература

1. Топорков В.В. Модели распределенных вычислений. М.: ФИЗМАТЛИТ, 2004. – 320 с.
2. Гергель, В.П. Теоретические основы экспериментального исследования алгоритмов планирования задач для вычислительного кластера с помощью симулятора / В.П. Гергель, П.Н. Полежаев // Журнал "Вестник Оренбургского государственного университета", №9(115), 2010. - С. 114-120.
3. Полежаев П.Н. Исследование алгоритмов планирования параллельных задач для кластерных вычислительных систем с помощью симулятора // Параллельные вычислительные технологии (ПАВТ'2010): Труды международной конференции. – Челябинск: Изд. ЮУрГУ, 2010 г. стр. 287 – 298.
4. Симулятор высокопроизводительного кластера TopSimity с учетом топологии системы и многопроцессорности узлов / Полежаев П.Н. Свидетельство Федеральной службы по ин-

теллектуальной собственности, патентам и товарным знакам №2010617255 от 29 октября 2010 г.

5. Полежаев П.Н. Симулятор вычислительного кластера и его управляющей системы, используемый для исследования алгоритмов планирования задач. // Журнал «Вестник ЮУрГУ», №35(211). Серия «Математическое моделирование и программирование», вып. 6, 2010. стр. 79 – 90.
6. 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.
7. 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.