УДК 004.451.42:.44:004.023

Алгоритмы планирования решения масштабируемых задач на распределённых вычислительных системах*)

С. Н. Мамойленко, А. В. Ефимов

Предложены алгоритмы оптимизации функционирования распределённых вычислительных систем при решении масштабируемых задач с учётом штрафов за задержку их решения и приоритетов выбора возможных конфигураций подсистем. Представлены результаты моделирования алгоритмов.

Ключевые слова: распределённые вычислительные системы, оптимизация функционирования.

1. Введение

Распределённые вычислительные системы (ВС) относятся к перспективным средствами обработки информации [1]. Основным функциональным элементом распределённой ВС является элементарная машина (ЭМ). Структура ЭМ допускает варьирование от процессорного ядра до конфигураций, включающих универсальные процессоры и специализированные ускорители (например, GPGPU). Количество ЭМ в распределённых ВС может изменяться в широких пределах: от десятка до сотен тысяч. Широкое распространение получили распределённые ВС, в которых в качестве функциональных элементов используются ЭВМ (в частности, обычные персональные компьютеры или специализированные серверы). Такие системы принято называть кластерными. В списке Top500 суперкомпьютеров кластерные системы составляют 84.8 % [2]. Следует отметить, что основы построения распределённых ВС (включая доказательство принципиальной возможности построения систем в условиях запаздывания сигналов) впервые были изложены в 1962 г. в Сибирском отделении АН СССР [3].

Эффективность функционирования ВС в немалой степени зависит от способа управления её ресурсами. Принято выделять три режима функционирования ВС: решение одной сложной задачи, обработка наборов задач и обслуживание потоков задач [1]. Два последних режима являются мультипрограммными, они основываются на распределении ресурсов ВС между несколькими одновременно решаемыми задачами. Повышение эффективности эксплуатации ресурсов распределённых ВС связано именно с использованием технологии параллельного мультипрограммирования [1, 4].

Одной из актуальных проблем организации функционирования ВС в мультипрограммных режимах является планирование решения пользовательских задач, представленных параллельными программами. Суть планирования заключается в формировании расписаний решения задач, то есть в определении для каждой задачи времени начала решения и необходимых ресурсов ВС. В общем случае проблема планирования является трудноразреши-

*)Работа выполнена в рамках междисциплинарного проекта № 113 СО РАН и при поддержке РФФИ (гранты № 08-07-00022, 09-07-0095) и Совета по грантам Президента РФ (ведущая научная школа НШ-5176.2010.9).

мой [5]. Поэтому перспективной считается разработка приближённых методов и эвристических алгоритмов управления ресурсами ВС.

Повысить эффективность ВС и снизить время нахождения задач в очереди возможно, если, в частности, каждая из них допускает решение на различных конфигурациях вычислительных ресурсов. В этом случае задачи называют *масштабируемыми* или пластичными (moldable) [4]. Исследования пользовательских запросов показывают, что свойством масштабируемости обладают более 80 % задач [6].

Описывая масштабируемую задачу, пользователь задает допустимые значения для её параметров. Для любой задачи, предназначенной для решения на распределённой ВС, задаются как минимум два параметра: ранг (т.е. количество ЭМ, необходимых для её решения) и требуемое (максимально допустимое) время решения. При этом какие-то значения параметров для задачи могут быть более предпочтительными по отношению к другим значениям. Предпочтения значений определяется природой решаемой задачи или иными соображениями пользователя. Кроме этого, могут задаваться и другие параметры (например, штраф за задержку решения задачи, дополнительные характеристики ресурсов ВС).

В данной работе предлагаются алгоритмы планирования решения масштабируемых задач на распределённых ВС (с учётом пожеланий пользователей на выбор значений параметров и штрафов за задержку решения задач). В основу алгоритмов положен метод формирования укрупнённых задач (пакетов) [7]. Планирование выполняется в два этапа: распределяются задачи по пакетам с целью минимизации общего времени их решения и определяется последовательность решения пакетов, минимизирующая суммарный штраф за задержку решения задач.

2. Обзор существующих подходов к организации функционирования распределённых ВС

Исследования проблемы организации функционирования распределённых ВС активно ведутся во многих странах. Обзор результатов, полученных в области планирования решения задач, можно найти, например, в [4, 7].

Организация мультипрограммных режимов функционирования распределённых ВС осуществляется средствами управления ресурсами, включающими: систему очередей (среду пакетной обработки), планировщик и среду решения пользовательских задач. Наибольшее распространение получили системы пакетной обработки заданий [8 – 11]: TORQUE, Altair PBS Pro, Oracle Grid Engine, IBM LoadLeveler, Microsoft HPC. Каждая из них имеет собственный планировщик и предоставляет интерфейс (API) для управления задачами с помощью программных систем сторонних разработчиков, например MAUI [12].

Для планирования решения задач чаще всего используются алгоритмы: First-Come-First-Served (FCFS), Shortest-Job-First (SJF), Longest-Job-First (LJF), а так же технологии Backfilling и Gang Scheduling, позволяющие увеличить эффективность решения задач с малым временим решения [13]. Существуют алгоритмы, учитывающие несколько критериев при планировании решения задач [14].

Системы пакетной обработки, используемые на практике, допускают наличие в очередях масштабируемых задач и позволяют пользователям указывать значения ранга в виде множества или диапазона. Планировщики обычно используют свойство масштабируемости, выбирая первое подходящее значение ранга. Известны алгоритмы выбора значений параметров задач путём их ранжирования [15], исходя из прогнозируемой загрузки ВС [16] или предполагаемого характера поступления задач [17].

В данной работе при планировании решения масштабируемых задач на ВС предлагается учитывать среднюю “удовлетворённость” пользователей от выбора значений параметров задач. “Удовлетворённость” пользователя оценивается отношением приоритета выбранных значений к максимальному приоритету значений параметров задачи.

3. Постановка задачи

Имеются распределённая ВС, состоящая из N элементарных машин, и набор $I = \{I_i\}$ независимых задач, $i = \overline{1,L}$. Каждая задача характеризуется вектором $P_i = \{p_i^k\}$ и величиной штрафа c_i за задержку её решения в единицу времени, $c_i > 0$. Элементы вектора $p_i^k = (r_i^k, t_i^k, w_i^k)$ задают для задачи I_i допустимые значения ранга r_i и времени решения t_i и определяют приоритет w_i^k выбора этих значений, $0 < r_i^k \le N$, $t_i^k > 0, w_i^k > 0, k = \overline{1, \mathrm{q_i}}$, $q_i = |P_i|$. “Удовлетворённость” пользователя выбором для задачи значений с приоритетом w_i^k оценивается как $w_i^k / \max\limits_k w_i^k$. Допускается существование в векторе P_i нескольких элементов с одинаковым приоритетом. Считается, что в наборе присутствуют задачи с различным количеством q_i допустимых значений параметров и возможно взаимодействие ЭМ с любой другой машиной ВС; зависимости величин t_i^k, r_i^k, w_i^k, c_i друг от друга отсутствуют.

Необходимо для каждой задачи I_i набора выбрать: k_i – номер элемента вектора P_i, и s_i – время начала решения задачи, а также выделить подсистему $J_i = \{j \in E_1^N\}$ машин ВС, где $0 < k_i \le q_i$, $s_i \ge 0$, j–номер ЭМ, $E_1^N = \{1,2, \dots, N\}$, $|J_i| = r_i^{k_i}$. В результате будет сформирован вектор $S = \langle (k_i, s_i, J_i) \rangle$, $i = \overline{1,L}$, называемый *расписанием решения задач* на ВС. Характеристиками расписания являются: время $T(S)$ решения всех задач набора и суммарный штраф $F(S)$ за задержку их решения.

Требуется найти расписание S^* решения задач на вычислительной системе такое, чтобы:

$$T(S^*) = \min_{S \in \Omega} T(S), \qquad T(S) = \max_{i=\overline{1,L}} \{s_i + t_i^{k_i}\} \tag{1}$$

$$F(S^*) = \min_{S \in \Omega} F(S), \qquad F(S) = \sum_{i=1}^{L} s_i c_i \tag{2}$$

при ограничениях:

$$\forall i \in \{1,2, \dots L\}, |J_i| = r_i^{k_i} \text{ и } \forall j_1 \in J_i, \forall j_2 \in J_i \backslash \{j_1\} \Rightarrow j_1 \ne j_2 \tag{3}$$

$$\forall t \ge 0, \bigcap_{i \in \Xi(t)} J_i = \emptyset, \sum_{i \in \Xi(t)} r_i^{k_i} \le N, \tag{4}$$

$$L^{-1} \sum_{i=1}^{L} \frac{w_i^{k_i}}{\max\limits_k w_i^k} \ge e, \tag{5}$$

где Ω – область допустимых расписаний S, $\Xi(t) = \{i \in \{1,2, \dots, N\} | s_i \le t \le s_i + t_i^{k_i}\}$ – множество номеров задач, решаемых в момент времени t; e – минимально допустимая средняя “удовлетворённость” пользователей. Ограничение (4) определяет, что каждой задаче с учётом выбранных значений параметров должно быть выделено столько ЭМ, сколько требуется для её решения. Ограничение (5) гарантирует, что в каждый момент времени задачи решаются на непересекающихся подсистемах ЭМ и их суммарный ранг не превосходит количества машин ВС.

Ясно, что задача (1) – (5) относится [18] к целевому программированию (точнее, к многокритериальной оптимизации).

4. Алгоритмы формирования укрупнённых задач

На первом этапе планирования задачи набора разбиваются на подмножества [7] таким образом, чтобы выполнялись ограничения (3) – (5) и достигался минимум функции (1).

Сформировать разбиение возможно, например, с использованием алгоритмов прямоугольной ортогональной упаковки без поворотов и пересечений [19]. При этом задача кодируется прямоугольным объектом с высотой, равной рангу задачи, и шириной, соответствующей запрашиваемому времени. В данной работе для упаковки использован алгоритм FFDH [20]. Очевидно, что в случае разбиения масштабируемых задач процедура упаковки усложняется выбором одного из допустимых размеров прямоугольника.

Формально постановка задачи первого этапа приобретает следующий вид. Решением является тройка $S_1 = (K, M, \Im)$, где $K = \langle k_i \rangle$ – вектор выбранных для задач значений k_i, $\Im = \{\Im_m\}$ – множество укрупнённых задач $\Im_m = \{I_i \in I\}, m = \overline{1, M}$, $\mathrm{M} = |\Im|$– количество укрупнённых задач. Требуется определить решение: S_1^* такое, чтобы

$$T(S_1^*) = \min_{S_1 \in \Omega_1} T(S_1), \qquad T(S_1) = \sum_{m=1}^{M} T_m, \quad T_m = \max_{i \in \Phi(m)} t_i^{k_i} \tag{6}$$

при ограничениях:

$$\bigcup_{m=1}^{M} \Im_m = \Im, \ \bigcap_{m=1}^{M} \Im_m = \emptyset, \tag{7}$$

$$R_m \le N, \qquad R_m = \sum_{i \in \Phi(m)} r_i^{k_i}, \tag{8}$$

где Ω_1 – область допустимых решений S_1, R_m – суммарный ранг задач каждого подмножества $\Im_m$, $\Phi(m) = \{i \in \{1,2,\dots,L\} | I_i \in \Im_m\}$ – множество номеров задач, помещённых в m-ое подмножество.

При выборе для задач значений k_j необходимо контролировать выполнение ограничения (5). Представим набор задач I в следующем виде: $I^0 \cup I^1 \cup I^2$, где I^0 – подмножество задач набора с одним элементом вектора P_i; I^1 – подмножество задач набора, у которых все элементы вектора P_i имеют одинаковый приоритет; I^2 – остальные задачи набора; $I^0 \cap I^1 \cap I^2 = \emptyset$. Тогда ограничение (5) можно представить в следующем виде:

$$L^{-1}\left(\sum_{I_i \in I^0} \frac{w_i^{k_i}}{\max\limits_k w_i^k} + \sum_{I_i \in I^1} \frac{w_i^{k_i}}{\max\limits_k w_i^k} + \sum_{I_i \in I^2} \frac{w_i^{k_i}}{\max\limits_k w_i^k}\right) \ge e. \tag{10}$$

Очевидно, что для задач из подмножества I^0 выбирать значения параметров нет необходимости и первое слагаемое выражения (10) всегда равно $|I^0|$. Второе слагаемое выражения (10) также равно $|I^1|$ при любом выборе значений параметров для задач из этого подмножества. На выполнение ограничения (5) влияет лишь выбор значений параметров для задач из подмножества I^2.

Утверждение 1. *Если* $|I^0| + |I^1| \ge eL$ *или* $\frac{\min\limits_{I_i \in I^2} \min\limits_k w_i^k}{\max\limits_{I_i \in I^2} \max\limits_k w_i^k} \ge \frac{eL - |I^0| - |I^1|}{|I^2|}$, *то ограничение (5) выполняется при любом выборе значений параметров для задач подмножества* I^2.

Доказательство. В силу того, что первое и второе слагаемые выражения (10) всегда равны соответственно$|I^0|$ и $|I^1|$, то выражение можно преобразовать следующим образом: $\sum_{i \in I^2} \frac{w_i^{k_i}}{\max\limits_k w_i^k} \ge eL - |I^0| - |I^1|$. Очевидно, что это неравенство всегда верно, если $|I^0| + |I^1| \ge eL$.

Далее. Рассмотрим самый худший вариант. Пусть каждая задача подмножества I^2 имеет элементы $p_i^{k_1}$ и $p_i^{k_2}$ соответственно с минимально и максимально возможным (для данного набора задач) приоритетом, и элемент $p_i^{k_1}$ выбирается для решения. Тогда выражение (10) можно записать в следующем виде: $|I^2| \frac{\min\limits_{I_i \in I^2} \min\limits_{k} w_i^k}{\max\limits_{I_i \in I^2} \max\limits_{k} w_i^k} \geq eL - |I^0| - |I^1|$. Утверждение доказано.

Если утверждение 1 не выполняется, то используем следующую процедуру. Считаем, что для задач уже были выбраны значения параметров (либо удовлетворяющие ограничению (5), либо с наивысшим приоритетом).

Шаг 1. Для задач подмножества I^1 значения k_i выбираются произвольно.

Шаг 2. Вычисляются значения средней “удовлетворённости” пользователей при решении задач набора согласно текущему выбору параметров.

Шаг 3. Произвольно выбирается задача из подмножества I^2 и новое значение k_i для неё.

Шаг 4. Если изменение приоритета значений параметров задачи не приводит к нарушению условия (5), то фиксируется k_i и осуществляется переход к шагу 2; в противном случае значение k_i для выбранной задачи не меняется.

Шаг 5. Процедура продолжается до тех пор, пока не будет перебраны все задачи подмножества I^2.

4.2. Алгоритм на основе метода цепей Монте-Карло

Алгоритм на основе метода цепей Монте-Карло [21] – итерационный. На каждой итерации случайным образом выбираются значения k_i так, чтобы удовлетворять ограничению (5), и формируются укрупнённые задачи алгоритмом FFDH. Итерации повторяются до тех пор, пока на заданном количестве итераций не будет найдено разбиение задач набора с меньшим значением функции (6).

4.3. Генетический алгоритм

Генетический алгоритм предусматривает последовательность жизненных циклов популяции. Каждый цикл состоит из следующих операций: выбора пар особей, их размножения, мутации и селекции наиболее приспособленных особей (формирования новой популяции). Процесс повторяется до тех пор, пока на протяжении заданного количества популяций не будет появляться особь с лучшим значением функции приспособленности. Итоговое разбиение задач набора формируется из особи, имеющей наилучшее значение функции приспособленности.

В терминах генетических алгоритмов [22] будем считать:

- *ген – укрупнённая задача (пакет);*
- *особь* или *хромосома* – допустимое разбиение задач набора при фиксированных значениях k_i;
- *популяция – несколько особей с различными значениями* k_i*;*
- *функция приспособленности особи* – значение функции (6) для соответствующего разбиения задач набора.

Размер популяции задаётся параметром алгоритма и остаётся постоянным в процессе всей эволюции.

Начальная популяция формируются следующим образом. Первая особь выбирает для каждой задачи k_i такое, при котором значения параметров имеют наивысший приоритет w_i^k, вторая – минимум запрашиваемых ресурсов $r_i t_i$ и третья – максимум “удельной ди” $w_i^k/(r_i^k t_i^k)$. Остальные особи выбирают k_i произвольно, так, чтобы удовлетворять ограничению (5). Далее каждая особь формирует укрупнённые задачи, используя алгоритм

FFDH. В итоге начальную популяцию составляют те особи, для которых выполняются ограничения (3) – (5).

На каждом жизненном цикле из популяции выбирается несколько пар. Количество пар равняется половине от количества особей в популяции. Для формирования пары из популяции случайным образом выбирается одна особь. Парой для неё выбирается особь, наиболее удалённая от неё по значению функции приспособленности и не состоящая в других парах.

Далее над каждой выбранной парой с заданной вероятностью либо выполняется оператор кроссинговера, либо производится мутация родительских особей.

В качестве оператора кроссинговера предлагается алгоритм, в основу которого положен метод перетасовки генов [23]:

Шаг 1. Гены родительских особей помещаются в общий контейнер и сортируются по критерию раскроя ($(R_m T_m)^{-1} \sum_{i \in \Phi(m)} r_i^{k_i} t_i^{k_i}$).

Шаг 2. Если в контейнере имеются одинаковые гены (с полностью одинаковыми задачами), то из двух генов удаляется тот, которому соответствует меньшее значение раскроя.

Шаг 3. Полученная последовательность генов просматривается до тех пор, пока не встретиться ген, в котором присутствует хотя бы одна задача, входящая в ранее просмотренные гены. Если такой ген не найден (т.е. просмотрен весь контейнер), то алгоритм завершается.

Шаг 4. Оставшиеся гены контейнера расформировываются. Из получившихся задач удаляются повторяющиеся или присутствующие в не расформированных генах.

Шаг 5. Из оставшихся задач по алгоритму FFDH создаются новые гены и помещаются обратно в контейнер.

В результате из генов контейнера получается новая особь, унаследовавшая от своих родителей лучшие гены.

Мутация с заданной вероятностью (задается параметром алгоритма) выполняется над одной или двумя родительскими особями. Оператор мутации с равной вероятностью либо случайным образом изменяет значения параметров задач, гарантируя выполнение ограничения (5), либо выбирает, для заданной доли задач особи, значения параметров, имеющие максимальный приоритет.

В результате от каждой пары создаются одна или две изменённые особи “потомков”, которые могут не выжить, если для них по каким-либо причинам не выполняются ограничения (3) – (5). В новую популяцию попадают родительские и “выжившие потомки”, имеющие наилучшие показатели функции приспособленности.

5. Выбор последовательности решения укрупнённых задач

На втором этапе определяется последовательность решения укрупнённых задач (пакетов), позволяющая получить минимум функции (2). Состав укрупнённых задач не изменяется.

Пусть $S_2 = m_1, \dots, m_l, \dots, m_M$ – некоторая последовательность решения пакетов, где m_l – номер пакета, который будет решаться в m-ю очередь. Штраф за решение задач при выбранной последовательности решения пакетов определяется как

$$F(S_2) = \sum_{l=2}^{M} C_{m_l} \sum_{r=1}^{l-1} T_{m_r}, \qquad (12)$$

где $C_{m_l} = \sum_{i \in \Phi(m_l)} c_i$ – штраф за задержку решения задач, входящих в m-й пакет.

Требуется найти такую последовательность S_2^*, чтобы достичь минимума функции (12).

Запишем функцию (12) в следующем виде

$$F(S_2) = C * T - f(S_2), \qquad (13)$$

где

$$C = \sum_{l=1}^{M} C_{m_l}, T = \sum_{l=1}^{M} T_{m_l}, \tag{14}$$

$$f(S_2) = \sum_{l=1}^{M} T_{m_l} \sum_{r=1}^{l} C_{m_r}. \tag{15}$$

Очевидно, что минимум (13) достигается при максимуме (15).

Утверждение 2. *Для того чтобы последовательность решения пакетов задач S_2 обеспечивала минимум функции (13), необходимо и достаточно, чтобы выполнялись неравенства*

$$\frac{T_{m_1}}{C_{m_1}} \le \frac{T_{m_2}}{C_{m_2}} \le \cdots \le \frac{T_{m_l}}{C_{m_l}} \le \cdots \le \frac{T_{m_M}}{C_{m_M}} \tag{16}$$

Необходимость. Докажем, что если при последовательности решения пакетов S_2 функция (15) принимает максимальное значение, то выполняются неравенства (16). Для этого рассмотрим последовательность

$$S_2' = m_1, \dots, m_{l_1}, \dots, m_{l_2}, \dots, m_M, \tag{17}$$

полученную из S_2 путём перестановки членов m_{l_1} и m_{l_2}.

По определению,

$$f(S_2) - f(S_2') = \sum_{l=1}^{M} T_{m_l} \sum_{r=1}^{l} C_{m_l} - \sum_{l=1}^{l_1-1} T_{m_l} \sum_{r=1}^{l} C_{m_l} - \sum_{l=l_2+1}^{M} T_{m_l} \sum_{r=1}^{l} C_{m_l} -$$

$$-T_{m_{l_1}} \left(\sum_{r=1}^{l_1-1} C_{m_r} + C_{m_{l_1}} \right) - T_{m_{l_1+1}} \left(\sum_{r=1}^{l_1-1} C_{m_r} + C_{m_{l_1}} + C_{m_{l_1+1}} \right) - \cdots -$$

$$-T_{m_{l_2}} \left(\sum_{r=1}^{l_1-1} C_{m_r} + C_{m_{l_1}} + C_{m_{l_1+1}} + \cdots + C_{m_{l_2-1}} + C_{m_{l_2}} \right) =$$

$$= \left(-T_{m_{l_1}} + T_{m_{l_2}} \right) \left(C_{m_{l_1+1}} + \cdots + C_{m_{l_2}} \right) + (C_{m_{l_1}} + C_{m_{l_2}})(T_{m_{l_1+1}} + \cdots + T_{m_{l_2}}) \ge 0.$$

Следовательно,

$$T_{m_{l_1}} \le T_{m_{l_2}} + \left(C_{m_{l_1}} + C_{m_{l_2}} \right) \frac{T_{m_{l_1+1}} + \cdots + T_{m_{l_2}}}{C_{m_{l_1+1}} + \cdots + C_{m_{l_2}}}, \quad l_1 < l_2. \tag{18}$$

Неравенство (12) является необходимым условием для того, чтобы пакет $\Im_{m_{l_1}} \in \Im$ решался раньше $\Im_{m_{l_2}} \in \Im$. При $l_2 = l_1 + 1$ условие (18) принимает следующий вид:

$$\frac{T_{m_{l_1}}}{C_{m_{l_1}}} \le \frac{T_{m_{l_2}}}{C_{m_{l_2}}}. \tag{19}$$

Учитывая (19), методом математической индукции легко доказать, что для любых двух пакетов $\Im_{m_{l_1}}$ и $\Im_{m_{l_2}}$ последовательности S_2, обеспечивающей максимальное значение функции (15), выполняется соотношение

$$\frac{T_{m_{l_1}}}{C_{m_{l_1}}} \le \frac{T_{m_{l_2}}}{C_{m_{l_2}}}, \quad l_1 < l_2 \tag{20}$$

Необходимость доказана.

Достаточность. Докажем, что если для последовательности S_2 выполняется (16), то значение функции (15) будет максимальным. Предположим противное. Пусть максимальное значение функции (15) соответствует некоторой последовательности

$$S_2'' = m_1, \dots, m_{l_3}, m_{l_3+1}, \dots, m_M, \tag{14}$$

для которой существует такое l_3, что

$$\frac{T_{m_{l_3+1}}}{C_{m_{l_3+1}}} \le \frac{T_{m_{l_3}}}{C_{m_{l_3}}}. \tag{15}$$

Формируем S_2''' из S_2'' и меняем в ней местами элементы m_{l_3} и m_{l_3+1}. Получим

$$f(S_2''') - f(S_2'') =$$

$$= \sum_{l=1}^{M} T_{m_l} \sum_{r=1}^{q} C_{m_r} - \sum_{l=1}^{l_3-1} T_{m_l} \sum_{r=1}^{l} C_{m_r} - T_{m_{l_3+1}} \left(\sum_{r=1}^{l_3-1} C_{m_r} + C_{m_{l_3+1}} \right) -$$

$$- T_{m_{l_3}} \left(\sum_{r=1}^{l_3-1} C_{m_r} + C_{m_{l_3+1}} + C_{m_l} \right) - \sum_{l=l_3+2}^{M} T_{m_l} \sum_{r=1}^{l} C_{m_r} = T_{m_{l_3+1}} C_{m_{l_3}} - T_{m_{l_3}} C_{m_{l_3+1}} < 0.$$

Итак, получено противоречие с предположением о соответствии максимального значения функции (15) последовательности S_2'', что и доказывает достаточность условия утверждения. Утверждение доказано.

Исходя из утверждения 2, последовательность S_2^* формируется таким образом, чтобы выполнялись неравенства (16). Таким образом, на втором этапе используется алгоритм сортировки (упорядочивания) пакетов согласно неравенствам (16).

6. Формирование итогового расписания

Итоговое расписание $S^* = \langle (k_i, s_i, J_i) \rangle$ формируется исходя из S_1^* и S_2^* следующим образом. Для k_i используются значения из S_1^*. Время начала решения s_i определяется началом решения пакета, в который она была помещена. Время s_r^* начала решения пакета определяется найденной последовательностью: $s_r^* = \sum_{l=1}^{r-1} T_{m_l}, s_1^* = 0$. Итак, $\forall m \in \{1,2, \dots, M\}$, $\forall i \in \Phi(m), s_i = s_m^*$. Машины ВС распределяются между задачами каждого пакета в соответствии с требованиями.

7. Моделирование и результаты

Моделирование и численные эксперименты осуществлялись с использованием ресурсов пространственно-распределённой мультикластерной вычислительной системы Центра параллельных вычислительных технология ГОУ ВПО ”Сибирский государственный университет телекоммуникаций и информатики” [24]. Предложенные алгоритмы реализованы с использованием языка программирования C++. Компиляция программ осуществлялась GNU\GCC с указанием максимально возможной степени оптимизации (–03). Наборы задач генерировались для систем с количеством ЭМ от 2^1 до 2^{20} на основе модели рабочей загрузки, предложенной в работе [10]. Рассматривались наборы с количеством задач 10, 100, 1000 и 10000. Приоритеты значениям параметров задач задавались путём их простой нумерации.

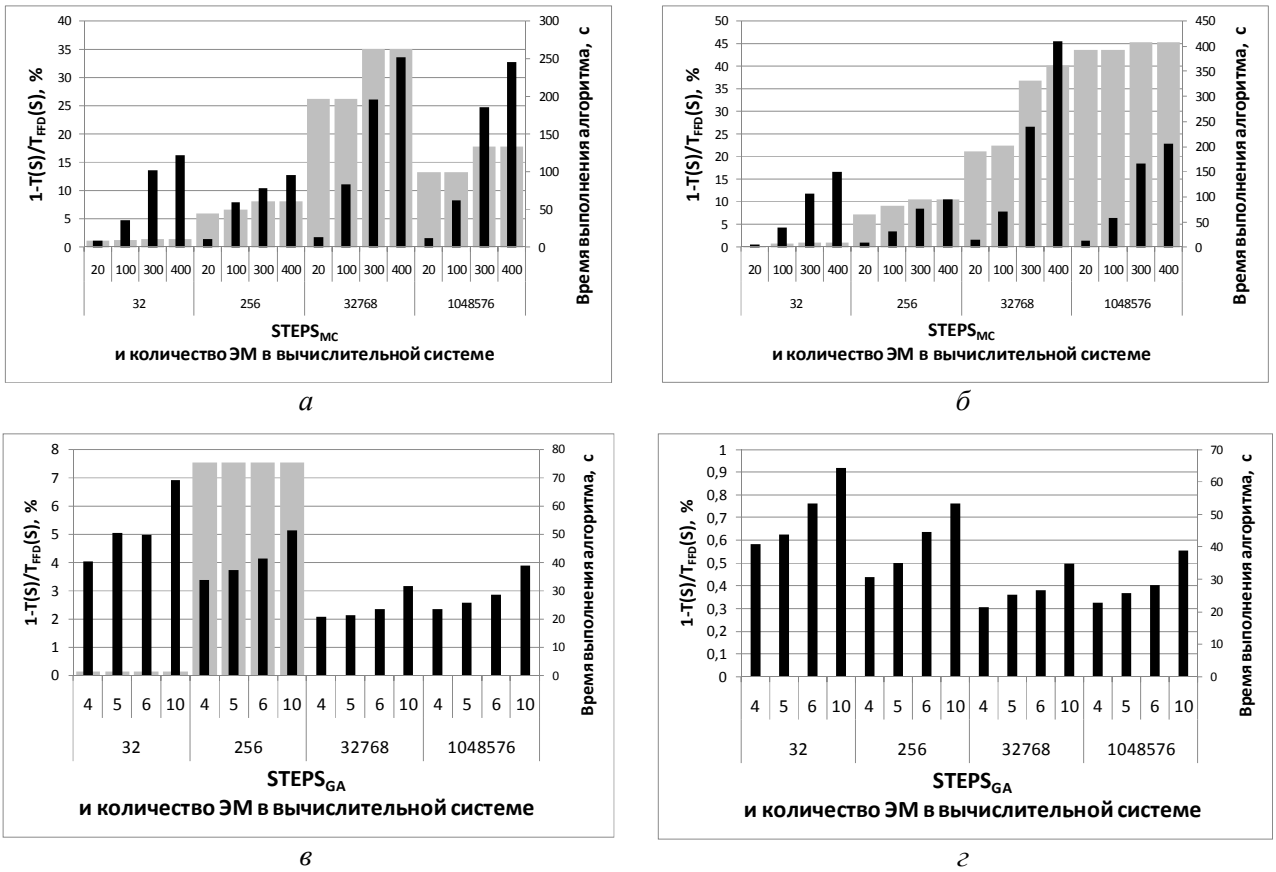
Процедура моделирования алгоритмов состояла из двух этапов: анализ влияния параметров алгоритмов на качество получаемого решения и сравнение алгоритмов между собой. Под качеством получаемого решения понимаем: полученное значение функции *T(S)*, средняя удовлетворённость пользователей и уровень загрузки вычислительных ресурсов (коэффициент раскроя) при решении задач по сформированному расписанию.

7.1. Анализ влияния параметров алгоритмов на качество формируемых расписаний решения задач

Для алгоритма Монте-Карло на качество решения влияет один параметр: количество итераций (обозначим его как $STEPS_{MC}$), на которых не получается расписание с лучшим значением функции *T(S)*. Генетический алгоритм требует задания трёх параметров: количества жизненных циклов популяций без улучшения потомства (обозначим как $STEPS_{GA}$), размера популяции (обозначим как PS_{GA}), вероятности выполнения кроссинговера и мутации (обозначим как P_{GA}). Кроме этого, оба алгоритма зависят от минимально-допустимой средней удовлетворённости пользователей (обозначим как *e*).

В результате моделирования установлено, что в данных условиях:

1. Для любого минимально допустимой средней "удовлетворённости" пользователей *e* оптимальным значением для $STEPS_{MC}$ является 100, а для $STEPS_{GA}$ – 5. При дальнейшем увеличении значении $STEPS_{MC}$ и $STEPS_{GA}$ время работы алгоритмов растёт, а качество получаемого решения практически не изменяется (рис. 1). $T_{FFD}(S)$ – это время решения задач по расписанию, для формирования которого задачам назначены значения параметров с максимальным приоритетом и однократно выполнен алгоритм FFDH.



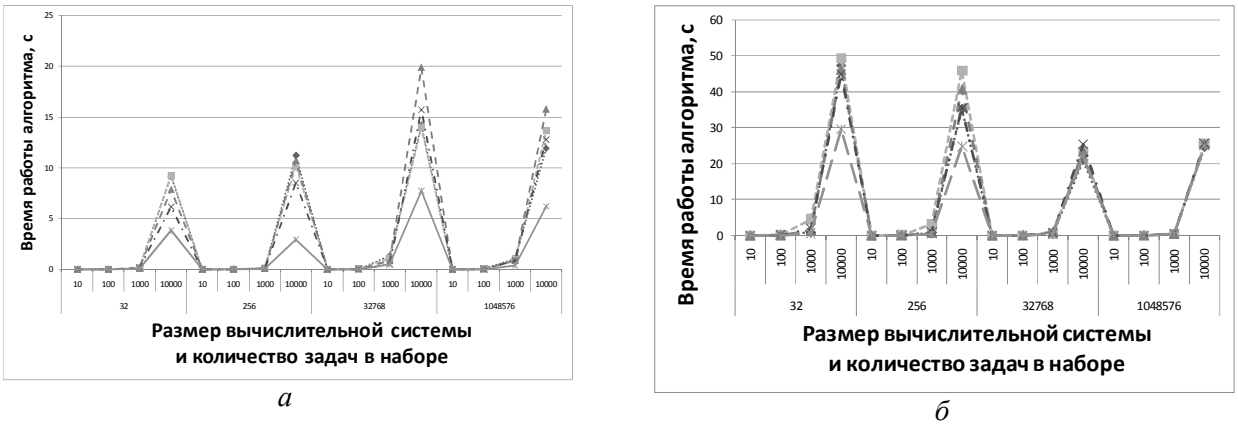
а *б*

в *г*

Рис.1. Влияние параметров $STEPS_{MC}$ и $STEPS_{GA}$ соответственно на качество работы алгоритма Монте-Карло (*а* и *б)* и генетического алгоритма (*в* и *г)*,
(L = 10000, PS_{GA} = 32, P_{GA} = 0.8, *а* и *в* – *e* = 0.50, *г* и *д* – *e* = 0.95),
■ – улучшение значения функции *T(S)*, %, ■ – время работы алгоритма, сек

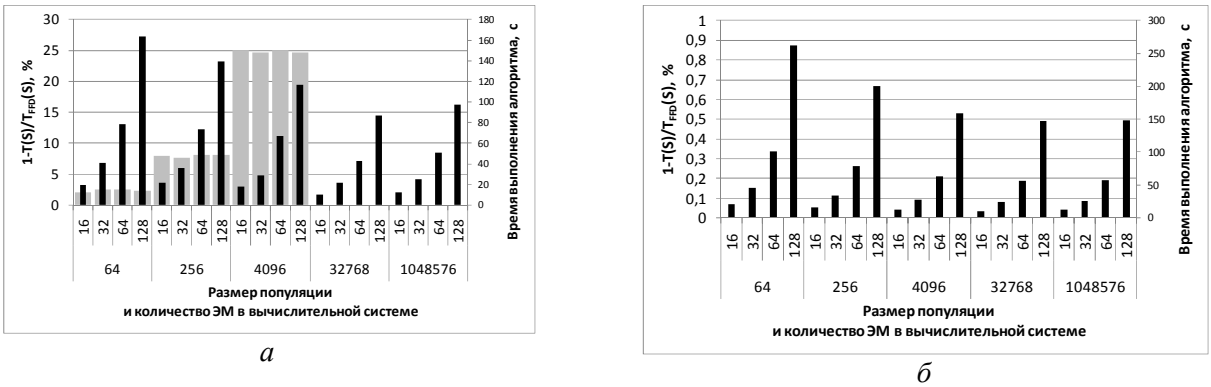
2. С увеличением значения параметра *e* уменьшается время работы алгоритмов. Это связано с тем, что количество вариантов значений параметров с увеличением *e* сокращается (рис. 2).

3. Наилучшим размером популяции PS_{GA} является 32 особи для любого значения минимально допустимой средней ”удовлетворённости” пользователей (рис. 3).
4. Вероятность выполнения операторов кроссинговера и мутации не влияет на качество получаемого решения (рис. 4).



а *б*

Рис.2. Влияние e на время работы алгоритмов, *а* – Монте-Карло, *б* – генетический алгоритм, L = 10000, $STEPS_{MC}$ = 100, $STEPS_{GA}$ = 5, PS_{GA} = 32, P_{GA} = 0.8),
e = – 0.50, – 0.75, – 0.90, – 0.95, – 1.00.



а *б*

Рис. 3. Влияние размера популяции на качество работы генетического алгоритма (L = 10000, $STEPS_{GA}$ = 5, P_{GA} = 0.8, *а* – e = 0.50 и *б* – e = 0.95),
– улучшение значения функции $T(S)$, %, – время работы алгоритма, сек

7.2. Сравнение эффективности алгоритма Монте-Карло и генетического алгоритма

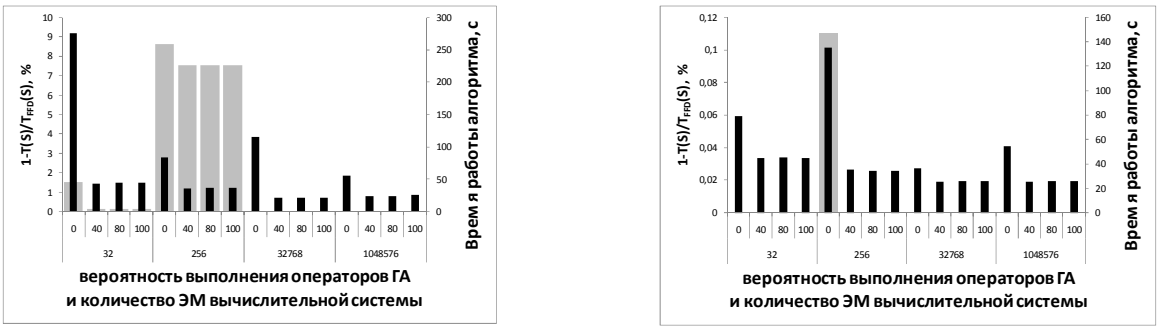
Алгоритмы сравнивались между собой по следующим критериям:

- $K_1=1-\frac{T_{MC}}{T_{GA}}$, где T_{MC} и T_{GA} – соответственно, время работы программы, реализующей алгоритм Монте-Карло и генетический алгоритм;
- $K_2=1-\frac{T_{MC}(S)}{T_{GA}(S)}$, где $T_{MC}(S)$ и $T_{GA}(S)$ – время решения задач по расписанию сформированному, соответственно, алгоритмом Монте-Карло и генетическим алгоритмом;
- $K_3=1-\frac{F_{GA}(S)}{F_{MC}(S)}$, где $F_{MC}(S)$ и $F_{GA}(S)$ – итоговая “удовлетворённость” пользователей при решении задач по расписанию, сформированному, соответственно, алгоритмом Монте-Карло и генетическим алгоритмом;

- $K_4=1-\frac{U_{GA}(S)}{U_{MC}(S)}$, где $U_{MC}(S)$ и $U_{GA}(S)$ – степень использования ресурсов ВС при решении задач по расписанию сформированному, соответственно, алгоритмом Монте-Карло и генетическим алгоритмом. $U(S)=\frac{\sum_i r_i^{k_i} t_i^{k_i}}{T(S)N}$ – коэффициент раскроя.

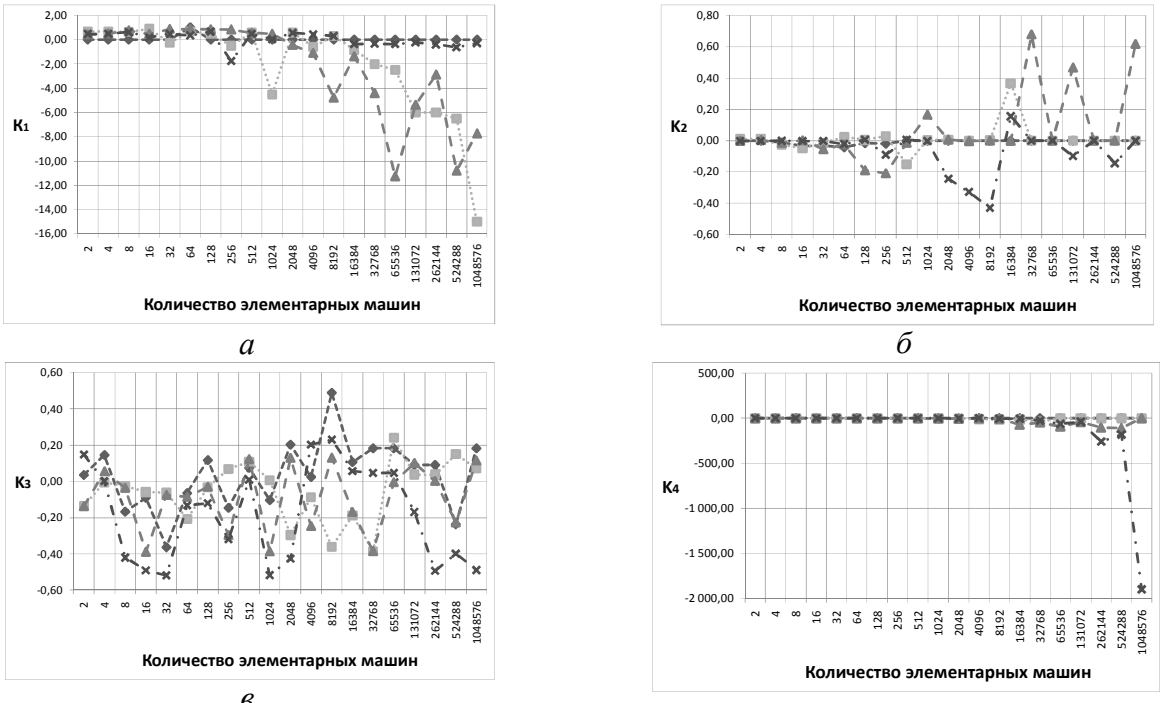
Очевидно, что чем больше значение критерия, тем лучше результаты алгоритма Монте-Карло, и наоборот. Абсолютное значение соответствующего критерия определяет, во сколько раз результаты одного алгоритма лучше по сравнению с другим. Некоторые результаты моделирования представлены на рис. 5.

В результате моделирования можно сделать вывод, что в данных условиях алгоритмы Монте-Карло и генетический алгоритм имеют примерно одинаковую эффективность.



а *б*

Рис. 4. Влияние вероятности выполнения операторов кроссинговера и мутации на качество работы генетического алгоритма (L = 10000, PS_{GA} = 32, *а* – *e* = 0.5, *б* – *e* = 0.95)
■ – улучшение значения функции *T(S)*, %, ■ – время работы алгоритма, сек



а *б*

в *г*

Рис. 5. Сравнение алгоритмов планирования решения задач
а – по критерию K_1, *б* – по критерию K_2, *в* – по критерию K_3, *г* – по критерию K_4,
e = 0.5, $STEPS_{MC}$ = 100, $STEPS_{GA}$=5, P_{GA}=0.8, PS_{GA}=32,
L – 10 100 1000 10000 .

8. Заключение

Разработанные алгоритмы планирования решения масштабируемых задач на ВС с учётом штрафов за задержку их решения и приоритетов выбора возможных конфигураций подсистем лягут в основу планировщика ресурсов пространственно-распределённой мультикластерной вычислительной системы Центра параллельных вычислительных технологий ГОУ ВПО ”СибГУТИ”.

Литература

1. Хорошевский В.Г. Архитектура вычислительных систем. – М.: МГТУ им. Н.Э. Баумана, 2008. – 520 с.
2. 34-я редакция (ноябрь 2009 года) списка 500 суперкомпьютеров мира. URL: http://www.top500.org/lists/2010/06 (дата обращения: 25.06.2010).
3. Евреинов Э.В., Косарев Ю.Г. О возможности построения вычислительных систем высокой производительности. Новосибирск: Изд-во СО АН СССР, 1962.
4. Dror G. Feitelson, Larry Rudolph, Uwe Schwiegelshohn, Kenneth C. Sevcik, Kenneth C. Parkson Wong. Theory and practice in parallel job scheduling // Job Scheduling Strategies for Parallel Processing, Volume 1291, 1997, pp. 1–34, ISBN: 978-3-540-63574-1.
5. Бруно Дж. Л., Грэхем Р.Л., Коглер В.Г., Коффман Э.Г. мл., Сети Р., Ульман Дж.Д., Штиглиц К., Теория расписаний и вычислительные машины // Под ред. Б.А. Головкина, пер. с англ. В.М. Амочкина, М.: Изд-во «Наука», 1984, 336 С.
6. W. Cirne and F. Berman, “A model for moldable supercomputer jobs”. 15th Intl. Parallel & Distributed Processing Symp., Apr. 2001 URL: http://www.lsd.dsc.ufpb.br/papers/moldability-model.pdf (дата обращения: 12.04.2010).
7. Евреинов Э.В., Хорошевский В.Г. Однородные вычислительные системы. – Новосибирск: Наука, 1978. – 319 с.
8. PBS Works – Enabling On-Demand Computing. URL: http://www.openpbs.org (дата обращения: 25.06.2010).
9. Grid Computing | Oracle Grid Engine | Software | Sun Microsystems. URL: http://www.sun.com/software/sge (дата обращения: 25.06.2010).
10. IBM Redbooks | Workload Management with LoadLeveler. URL: http://www.redbooks.ibm.com/abstracts/sg246038.html (дата обращения: 25.06.2010)
11. Windows HPC Server 2008 | Microsoft Supercomputing | Supercomputers. http://www.microsoft.com/hpc/ (дата обращения: 25.06.2010).
12. Cluster resources :: Products – Maui Cluster Scheduler URL: http://www.clusterresources.com/pages/products/maui-cluster-scheduler.php (дата обращения: 25.06.2010).
13. E. Shmueli and D. G. Feitelson, “Backfilling with lookahead to optimize the packing of parallel jobs”. J. Parallel & Distributed Comput. 65(9), pp. 1090–1107, Sep. 2005.
14. W. Cirne, C. Grande and F. Berman, “When the herd is smart aggregate behavior in the selection of job request”, IEEE Transactions in Parallel and Distributed Systems, 2003, vol. 14, pp. 181–192.
15. L. Barsanti and A. Sodan “Adaptive job scheduling via predictive job resource allocation”, Lecture Notes in Computer Science, 2007, vol. 4376, pp. 115–140.
16. Седельников М.С. Алгоритмы распределения набора задач с переменными параметрами по машинам вычислительной системы // Автометрия. – 2006. – Т. 42. – № 1. – С. 68–76.

17. Pierre-François Dutot, Lionel Eyraud, Grégory Mounié, Denis Trystram. Bi-criteria algorithm for scheduling jobs on cluster platforms // Proceedings of the sixteenth annual ACM symposium on Parallelism in algorithms and architectures, Barcelona, Spain, 2004, Pages: 125 – 132, ISBN:1-58113-840-7
18. Таха, Х. Введение в исследование операций : 6-е изд. / Таха Хэмди А., пер. с англ. В.И. Тюпти, А.А. Минько. – М.: Вильямс, 2001. – 911 с.
19. Survey on two-dimensional packing. URL: http://www.csc.liv.ac.uk/~epa/survey.pdf (дата обращения: 25.06.2010).
20. E.G. Coffman Jr and M.R. Garey and D.S. Johnson and R.E. Tarjan. Performance bounds for level-oriented two-dimensional packing algorithms. SIAM Journal on Computing, 9:808–826, 1980.
21. Ермаков С. М. Методы Монте-Карло и смежные вопросы. М.: Наука, 1971г.
22. Гладков Л.А., Курейчик В.В., Курейчик В.М. Генетические алгоритмы / Под ред. В.М. Курейчика. – 2-е изд., испр. и доп. –М.: ФИЗМАТЛИТ, 2006. – 320 с.ISBN 5-9221-0510-8
23. Philipp Rohlfshagen, John A. Bullinaria. A genetic algorithm with exon shuffling crossover for hard bin packing problems // Proceedings of the 9th annual conference on Genetic and evolutionary computation.–ACM NewYork, NY, USA, 2007.–pp.1365 – 1371
24. Ресурсы Центра параллельных вычислительных технологий ГОУ ВПО «СибГУТИ». URL: http://cpct.sibsutis.ru. (дата обращения: 25.06.2010).

Статья поступила в редакцию 30.05.2010

Мамойленко Сергей Николаевич

Доцент, к.т.н., заместитель заведующего Кафедрой вычислительных систем Сибирского государственного университета телекоммуникаций и информатики. Область научных исследований – параллельное мультипрограммирование, методы организации функционирования вычислительных систем в моно- и мультипрограммных режимах.

Тел. (383) 269-82-75, e-mail: msn@sibsutis.ru, msn@isp.nsc.ru.

Ефимов Александр Владимирович

Аспирант Кафедры вычислительных систем Сибирского государственного университета телекоммуникаций и информатики. Область научных исследований – параллельное мультипрограммирование, методы организации функционирования вычислительных систем в моно- и мультипрограммных режимах

Тел. (383) 269-82-75, e-mail: efimov@cpct.sibsutis.ru

Algorithms of solution planning of moldable jobs on distributed computer systems

S. N. Mamoilenko, A.V. Efimov

The algorithms of functioning optimization of distributed computer systems at scheduling moldable jobs with allowance for the penalty for delay of its decision and priority of possible choice of subsystems configurations are proposed. Algorithms modeling results are presented.

Keywords: geographically-distributed computer systems, functioning optimization.