Исследование алгоритмов планирования параллельных задач для кластерных вычислительных систем с помощью симулятора*

П.Н. Полежаев

В данной работе описываются результаты экспериментального исследования различных алгоритмов планирования задач для вычислительного кластера, полученные с помощью программного симулятора вычислительного кластера и его управляющей системы. Для проведения исследования предложена модель вычислительной загрузки кластера, разработана его имитационная схема, а также построены критерии и метрики сравнения алгоритмов планирования.

1. Введение

В настоящее время благодаря дешевизне серийно выпускаемого аппаратного обеспечения и возможности его гибкого конфигурирования кластерные вычислительные системы получили большое распространение. Они используются для решения вычислительно емких задач в различных отраслях человеческой деятельности. Однако, даже не очень производительные кластерные системы весьма дороги, и, чтобы окупить затраты на их покупку, необходимо максимально рационально использовать доступные вычислительные мощности. Это может быть сделано за счет применения эффективных алгоритмов планирования параллельных задач, поступающих в управляющую систему вычислительного кластера.

Алгоритмы планирования, используемые в существующих управляющих системах, обладают целым рядом недостатков. Прежде всего, они не в состоянии обеспечить высокую вычислительную загрузку узлов кластера. Часто возникает ситуация, когда есть свободные вычислительные узлы, в то время как в очереди ожидает своего исполнения достаточно большое количество задач. Более того, они не всегда рассматривают случай, когда вычислительные узлы имеют неоднородный состав, т.е. когда они отличаются объемами доступной оперативной и дисковой памяти, производительностью процессоров. Также не во всех управляющих системах учитывается возможность наличия локальной загрузки узлов, которая характерна для кластеров рабочих станций.

Современные исследования [1–4], рассматривающие различные списочные алгоритмы планирования, не достаточно полные, т.к. не затрагивают всех основных существующих алгоритмов или рассматривают достаточно ограниченное количество анализируемых метрик. Чаще всего это зависимости среднего времени ожидания в очереди и использования процессорных ресурсов от загрузки системы потоком работ. В источниках практически не затрагиваются не менее важные метрики сбалансированности, гарантированности и честности, а также не достаточно полно изучается поведение алгоритмов планирования на кластерах рабочих станций, на кластерах со значительной степенью гетерогенности аппаратной платформы.

Данная работа призвана попытаться частично исправить сложившуюся ситуацию. В нашем исследовании проводится достаточно полный сравнительный анализ всех основных списочных алгоритмов планирования с помощью широкой системы критериев и метрик. Она охватывает все аспекты эффективности составляемых расписаний запуска задач: производительность, используемость памяти узлов, сбалансированность их загрузки, гарантированность обслуживания задач, а также честность алгоритма планирования по отношению к задачам.

В настоящем исследовании рассматриваются четыре сценария: гомогенный кластер при наличии или отсутствии локальной загрузки узлов и гетерогенный кластер при наличии или отсутствии локальной загрузки вычислительных узлов.

* 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).

Для целей данной работы разработана модель вычислительной загрузки кластера, построена его имитационная схема, которая была реализована в виде программного симулятора работы вычислительного кластера. Он использовался для количественной оценки эффективности работы алгоритмов планирования на реалистичных рабочих загрузках кластера потоком параллельных задач и имитации локальной загрузки вычислительных узлов.

Использование симулятора по сравнению с реальным кластером дает финансовую выгоду, уменьшает дополнительные трудозатраты (например, нет необходимости обеспечивать отказоустойчивость, безопасность или составлять набор реальных задач для эксперимента), снижает время исследования, а также обеспечивает гибкую (возможность изменения аппаратной конфигурации) и контролируемую (отсутствие аппаратных и программных сбоев) среду вычислений. Кроме того, симулятор обеспечивает большую реалистичность результатов по сравнению с использованием метода аналитического анализа моделей, для которого характерно наличие значительного числа упрощений и предположений.

2. Имитационная схема и модель кластера

На рисунке 1 приведена имитационная схема управляющей системы кластера, включающая два источника: $И_1$ формирует поток параллельных задач, отправляемых пользователями в управляющую систему вычислительного кластера, $И_2$ генерирует локальную загрузку узлов. Не ограниченный по длине, накопитель $Н_1$ представляет собой очередь и используется для хранения заявок на выполнения задач, поступающих от источника $И_1$. Канал $К_0$ представляет собой выделенный управляющий узел, извлекающий из $Н_1$ подходящую задачу и распределяющий ее, в соответствии с заложенным алгоритмом, на требуемое количество доступных вычислительных узлов – подмножество свободных каналов $К_1$, $К_2$, …, $К_m$.

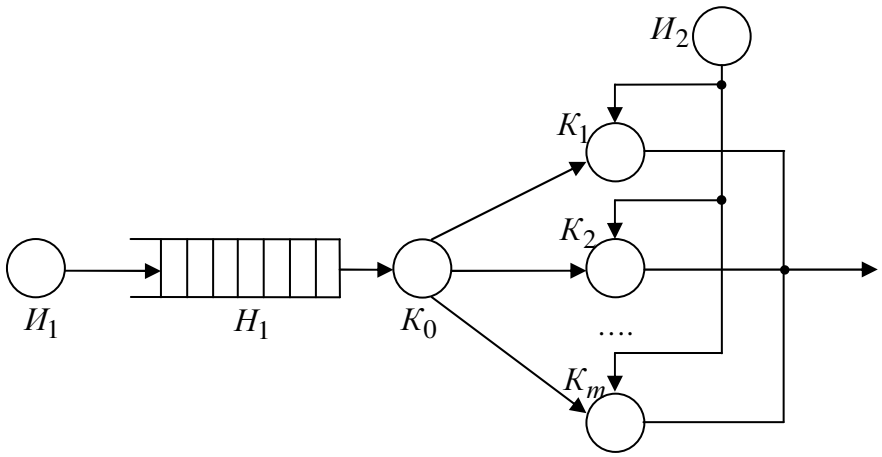

Рис. 1. Имитационная модель СПО кластера

Каждый вычислительный узел K_i имеет M_i Кбайт оперативной памяти и D_i Кбайт дисковой памяти за вычетом операционной системы и предустановленных программ, он также характеризуется относительной вычислительной скоростью S_i, которая показывает во сколько раз данный узел быстрее самого медленного в кластере. Кроме того, узел K_i описывается следующими динамическими параметрами: $m_i(t)$, $d_i(t)$ и $u_i(t)$ - соответственно значения доступной оперативной памяти, дисковой памяти и загруженности процессора i-го узла в момент времени $t \in [0;\infty)$.

В рамках данной модели не учитывается топология вычислительного кластера, многопроцессорность вычислительных узлов, а коммуникационные задержки, возникающие при передаче данных между процессами, исполняющейся параллельной программы, учитываются неявно и закладываются в ее время выполнения.

Данная имитационная схема и модель легли в основу симулятора вычислительного кластера и его управляющей системы, используемого для исследования алгоритмов планирования задач.

3. Модель вычислительной загрузки кластера

Вычислительная загрузка кластера формируется из потока задач, отправляемых пользователями в его управляющую систему, и локальной вычислительной загрузки узлов, формируемой локальными приложениями, выполняемыми на узлах.

Пусть $J = (J_1,...,J_n)$ - поток (последовательность) задач, упорядоченная по возрастанию времени их прибытия $a_j \in [0;\infty)$. Данный поток включает в себя все задачи, поступающие от пользователей в управляющую систему вычислительного кластера. Отдельно заметим, что алгоритм планирования, используемый в управляющей системе вычислительного кластера, знает только о тех задачах, которые попали в очередь до текущего момента времени.

Пользователь для задачи J_j указывает следующие требования к ресурсам кластера: n_j - количество узлов, mr_j - объем оперативной и dr_j - объем дисковой памяти в килобайтах на каждом узле. Также каждая задача характеризуется, выполняемой пользователем, оценкой времени выполнения $\tilde{p}_j$, осуществляемой, при условии, что все n_j узлов будут иметь единичные вычислительные скорости $S_i = 1$. Пусть N_j - множество узлов, выделенных планировщиком j-й задаче, тогда оценка времени ее выполнения с учетом различной скорости выделенных узлов будет равна $p_j = \dfrac{\tilde{p}_j}{\min\limits_{i \in N_j} S_i}$.

Кроме описанных выше характеристик, каждая задача также имеет параметр $\tilde{\tau}_j \in (0; p_j]$ – ее реальное время выполнения на узлах единичной скорости. Аналогично определяется реальное время выполнения задачи с учетом различных скоростей назначенных ей планировщиком узлов: $\tau_j = \dfrac{\tilde{\tau}_j}{\min\limits_{i \in N_j} S_i}$.

При планировании с использованием приоритетов для j-й задачи приоритет w_j либо задается пользователем, либо определяется непосредственно алгоритмом планирования.

Симуляция работы вычислительного кластера и его управляющей системы предполагает генерацию потока задач и локальной загрузки вычислительных узлов. Параметры задач и локальной вычислительной загрузки представляют собой случайные величины, имеющие определенные законы распределений, подбираемые на основе анализа реальных трасс, собираемых на кластерных вычислительных системах.

Анализ существующих моделей потоков задач [5–9] показывает, что они рассматривают небольшое количество параметров (чаще все только a_j, n_j, t_j и p_j) и требуют усовершенствования за счет учета требований к оперативной и дисковой памяти вычислительных узлов.

Далее коротко опишем используемые в настоящем исследовании законы распределений для каждого параметра, характеризующего задачу.

Требуемое количество процессоров n_j. Все задачи потока можно сгруппировать в три класса [9]: последовательные задачи, 2^k-задачи (параллельные задачи с n_j равным степени числа 2) и остальные. Генерация значений n_j (см. рисунок 2) осуществляется следующим образом. С вероятностью q_1 задача будет последовательной ($n_j = 1$), иначе для параллельной задачи выбирается согласно заданному закону распределения число x, являющееся логарифмом n_j. С вероятностью q_2 оно будет округлено до целого (в этом случае получим 2^k-задачу),

иначе получим параллельную задачу с произвольным числом требуемых процессоров. Для симуляции использовались следующие значения данных параметров [9]: $q_1 = 0.2$ и $q_2 = 0.325$. За основу для генерации значения n_j было взято двухэтапное равномерное распределение с параметрами [9]: $a = 1$ (наименьшее возможное значение), $b = \log_2 m$ (максимальное возможное значение), $c = \log_2 m - 1.5$ и $p = 0.3$.

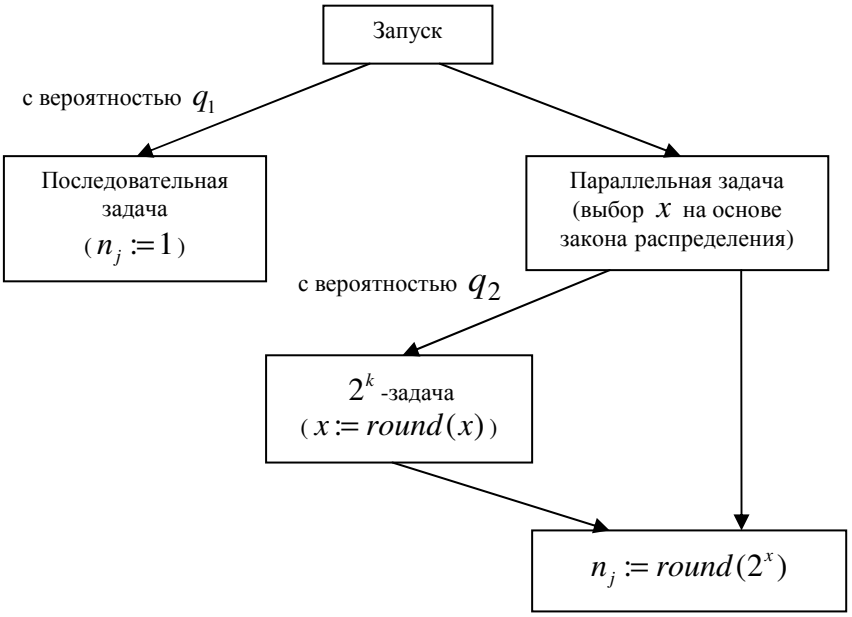

Рис. 2. Алгоритм генерации требуемого количества процессоров

Реальное время выполнения задачи τ_j. В данной работе используется подход, описанный в [9]. Логарифм реального времени выполнения задачи имеет гипер-гамма распределение, параметр p которого линейно зависит от n_j. При исследовании использовались следующие усредненные значение параметров распределения: $\alpha_1 = 6.57$, $\beta_1 = 0.823$, $\alpha_2 = 639.1$, $\beta_2 = 0.0156$, а линейная зависимость описывалась с помощью формулы $p = -0.003 n_j + 0.6986$.

Оценка пользователя времени выполнения задачи $\tilde{p}_j$. В данной модели будет использован подход, описанный в работе [3], согласно которому $\tilde{p}_j$ – равномерно распределенная величина в интервале $[\tau_j; f\tau_j]$, где $f \in [1;4]$ – константный показатель, отражающий неточность оценок пользователей. Для исследования было принято значение $f = 2$.

Время поступления a_j задачи в управляющую систему. Времена прибытия задач имеют распределение Пуассона [2, 5, 9], следовательно, интервалы времени между прибытиями подчиняются экспоненциальному закону распределения. В качестве параметра интенсивности может было взято значение $\lambda = \frac{1}{15}$ [7].

Требования к оперативной и дисковой памяти узлов. Данный способ генерации предложен нами и опирается на равномерное распределение. Пусть $Q_1 < Q_2 < ... < Q_k$ – отсортированный по возрастанию список значений объемов оперативной памяти на узлах, в котором исключены повторяющиеся значения, $W_1 < W_2 < ... < W_s$ – аналогичный список для объемов дисковой памяти узлов. Пусть g_{ij} - количество узлов кластера, у которых $M_l \geq Q_i$ и $D_l \geq W_j$ ($i = \overline{1,k}$ и $j = \overline{1,s}$). Тогда алгоритм генерации требований l-й работы к оперативной и дисковой памяти узлов будет следующим:

1. Если $n_l < \left\lceil \frac{m}{2} \right\rceil$, то $n_l := \left\lceil \frac{m}{2} \right\rceil$.

2. $I := \{\}$ – множество пар индексов вначале пусто.

3. Для каждого $i = \overline{1,k}$ найти такое наибольшее j, чтобы $g_{ij} \geq n_j$. Если для данного i такое j найти нельзя, то переходим к следующему значению для i, иначе сохраняем пару найденных индексов: $I := I \cup \{(i, j)\}$.

4. Выбрать случайно и равновероятно упорядоченную пару индексов (i, j) из I.

5. Выбрать произвольное значение mr_l (равномерное распределение) из интервала $[mr_{\min}; Q_i \cdot \alpha_m]$, где $mr_{\min}$ – минимальный объем требуемой оперативной памяти, $\alpha_m \in [0;1]$ – доля оперативной памяти узла, которая может быть занята работой.

6. Выбрать произвольное значение dr_l (равномерное распределение) из интервала $[dr_{\min}; W_j \cdot \alpha_d]$, где $dr_{\min}$ – минимальный объем требуемой дисковой памяти, $\alpha_d \in [0;1]$ – доля дисковой памяти узла, которая может быть занята работой.

Шаг 1 необходим для того, чтобы гарантировать, что любая задача может быть потенциальна запущена хотя бы на половине узлов кластера. Для симуляции использовались следующие значения параметров: $\alpha_m = \alpha_d = 0.70$ и $mr_{\min} = md_{\min} = 1024$ Кб.

Приоритет задачи w_j. Для генерации приоритета w_j задачи можно выбрать различные распределения. В данной модели будет использоваться равномерное распределение в интервале $[1;100]$.

Локальная вычислительная загрузка узлов (рабочих станций) формируется за счет программ, запускаемых их локальными пользователями. Непосредственное моделирование запуска программ, их диспетчеризации планировщиком локальной ОС узла чрезмерно бы усложнило разрабатываемую модель, при этом нет необходимости в такой точности получаемых результатов. Поэтому нами был предложен описанный далее подход.

Пусть $\psi_{mi}(t)$, $\psi_{di}(t)$ и $\psi_{ui}(t)$ – величины локальной загруженности соответственно оперативной памяти, дисковой памяти и процессора i-го узла в момент времени t. Причем, для любого узла i в любой момент t времени данные значения лежат в диапазоне от 0 до 1.

Величины $\psi_{mi}(t)$, $\psi_{di}(t)$ и $\psi_{ui}(t)$ имеют гамма распределения с соответствующими параметрами (α_m, β_m), (α_d, β_d) и (α_u, β_u).

Пусть ΔT_m, ΔT_d и ΔT_u – случайные величины, характеризующие интервалы времени, через которые происходит изменение соответствующих величин $\psi_{mi}(t)$, $\psi_{di}(t)$ и $\psi_{ui}(t)$. Они имеют экспоненциальный закон распределения с соответствующими параметрами λ_{m}, λ_{d} и λ_{u} (интенсивностями). Причем, логично предположить, что значения локальной загрузки процессора меняются интенсивнее значения доступности оперативной памяти, а оно, в свою очередь, интенсивнее значения доступности жесткого диска.

Значение загрузки процессора, а также доступной оперативной и дисковой памяти i-го узла в момент времени t определяется по формулам (1)-(3):

$$u_i(t) = \begin{cases} 1, & \text{если на } P_i \text{ выполняется } j\text{-ая работа в момент } t, \\ \psi_{ui}(t), & \text{иначе,} \end{cases} \quad (1)$$

$$m_i(t) = \begin{cases} M_i - (M_i - mr_j)\psi_{mi}(t) - mr_j, & \text{если на } P_i \text{ выполн. } j\text{-ая работа ,} \\ M_i - M_i\psi_{mi}(t), & \text{иначе,} \end{cases} \quad (2)$$

$$d_i(t) = \begin{cases} D_i - (D_i - dr_j)\psi_{di}(t) - dr_j, & \text{если на } P_i \text{ выполн. } j \text{ - ая работа}, \\ D_i - D_i\psi_{di}(t), & \text{иначе.} \end{cases} \quad (3)$$

Для целей симуляции были выбраны следующие значения параметров: $\alpha_m = 0.2$, $\beta_m = 0.3$, $\alpha_d = 0.2$, $\beta_d = 0.2$, $\alpha_u = 0.5$, $\beta_u = 0.5$, $\lambda_{\text{u}} = 0.08$, $\lambda_{\text{m}} = 0.04$ и $\lambda_{\text{d}} = 0.02$.

В данном разделе была описана модель вычислительной загрузки кластера, используемая в процессе симуляции работы вычислительного кластера и его управляющей системы.

4. Исследуемые алгоритмы планирования задач

Составление оптимальных расписаний запуска работ для кластерной вычислительной системы относится в общем случае к классу NP-полных задач [10–13], требующих экспоненциального времени решения, что неприемлемо для планировщика, т.к. он должен инициировать цикл планирования каждый раз при наступлении в системе некоторого события – поступления в очередь новой задачи, завершении исполняющейся задачи или изменении локальной загрузки вычислительного узла. В связи с этим для планирования задач на кластере используются различные эвристические алгоритмы планирования, строящие субоптимальные расписания.

В данной работе исследуются наиболее известные списочные эвристические алгоритмы планирования задач [1–3, 13, 14], ориентированные на разделение пространства вычислительных ресурсов. Среди прочих алгоритмов, оставшихся за рамками исследования, можно отметить алгоритм планирования групп задач (gang scheduling) [6], основанный на принципе разделения времени, когда на одних и тех же узлах поочередно выполняются различные наборы задач.

В рамках данного исследования в структуре алгоритма планирования выделяются следующие части: алгоритм выбора задачи для назначения на узлы, метод назначения задачи на узлы и критерий доступности узлов для планирования.

Алгоритм выбора задачи для назначения – алгоритм, который на основе имеющейся информации о доступных узлах кластера и ожидающих исполнения задачах, выбирает из очереди очередную задачу для назначения ей ресурсов.

Метод назначения определяет наиболее подходящие узлы из доступных для выбранной задачи. Варианты методов назначения хорошо освещены в источниках [10–14].

Критерий доступности узла представляет собой логическое выражение, завязанное на статических и динамических параметрах узла. Узел доступен тогда и только тогда, когда соответствующее логическое выражение принимает истинное значение. Критерий доступности узла особенно актуален для кластера рабочих станций. В рамках данного исследования при наличии локальной загрузки вычислительных узлов применялся критерий, представляющий собой требование, чтобы свертка загруженности оперативной памяти, дисковой памяти и процессора была не больше некоторого заданного порогового значения.

В данной работе рассматриваются следующие эвристические алгоритмы выбора задач для назначения: First Come First Served (FCFS, первым пришел – первым обслужен), First Come First Served Scan (FCFS Scan, первым пришел – первым вышел со сканированием очереди), Priority Queue (PQ, очередь с приоритетами), Priority Queue Scan (PQ Scan, очередь с приоритетами со сканированием), Shortest Job First (SJF, кратчайшая задача первая), Shortest Job First Scan (SJF Scan, кратчайшая задача первая со сканированием), Longest Job First (LJF, длиннейшая задача первая), Longest Job First Scan (LJF Scan, длиннейшая задача первая со сканированием), Most Processors First Served (MPFS, задача с наибольшим количеством процессоров первая), Most Processors First Served Scan (MPFS Scan, задача с наибольшим количеством процессоров первая со сканированием), Least Processors First Served (LPFS, задача с наименьшим количеством процессоров первая), Smallest Work Job First (SWJF, задача с наименьшей работой первая), Smallest Work Job First Scan (SWJF Scan, задача с наименьшей работой первая со сканированием), Largest Work Job First (LWJF, задача с наибольшей работой первая), Largest Work Job First Scan (LWJF Scan, задача с наибольшей работой первая со сканированием), Random First Served (RFS,

случайная задача обслуживается первой), агрессивный вариант алгоритма Backfill (обратного заполнения). Их описание можно найти в литературе [1–4, 10–14].

В рамках данного исследования рассматриваются следующие методы назначения задач на доступные вычислительные узлы кластера: First Fit (FF, первый подходящий) – выбираются первые попавшиеся подходящие узлы из первого подходящего окна расписания; Best Fit (BF, наилучший подходящий) – выбираются узлы из такого окна расписания, которое при закрытии его задачей даст наименьшую оставшуюся площадь; Fastest Node First (FNF, самый быстрый узел первым) – выбираются подходящие узлы подходящего окна, на которых данная задача может быть выполнена максимально быстро; Least Utilized Node First (LUNF, наименее загруженный узел первым) – выбираются наименее загруженные подходящие узлы подходящего окна; Random First (RF, случайный узел первым) – случайным образом (равновероятно) выбираются подходящие узлы случайно выбранного подходящего окна.

5. Система критериев и метрик сравнения алгоритмов планирования задач

С целью охвата всех аспектов эффективности составляемых алгоритмами планирования расписаний запуска задач была построена, описываемая в данном разделе, система критериев и метрик их сравнения.

Анализ алгоритмов планирования по определенному критерию представляет собой сравнение для различных алгоритмов зависимостей метрик этого критерия от величины системной загрузки и выбор лучшего варианта для того или иного случая. Сравнительный анализ удобнее всего представлять визуально с помощью графиков зависимостей метрик от системной загрузки L, определяемой 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.

Критерии сравнения алгоритмов планирования и метрики, к ним относящиеся:

1. Производительность расписаний. Включает следующие метрики: средняя загруженность процессора узла кластера $U_{проц.}$, потеря производительности кластера CL, максимальное время завершения задачи $C_{\max}$, среднее время ожидания задачи в очереди $\bar{t}_{ож.}$ и среднее ограниченное замедление задачи $\bar{s}_{огр.}$. Метрики $U_{проц.}$, CL и $C_{\max}$ характеризуют непосредственно производительность расписания, поэтому они более приоритетны для исследования, чем $\bar{t}_{ож.}$ 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. Их, например, можно отдать под локальные приложения.

3. Сбалансированность загрузки узлов. Включает метрики Du, Dm и Dd, обозначающие соответственно дисперсии средние загруженности процессора, оперативной и дисковой памяти вычислительного узла. Демонстрирует честность алгоритма планирования по отношению к узлам.

4. Гарантированность обслуживания задач. Включает метрики максимального времени ожидания задачи в очереди $t_{ож.\max}$ и максимальное ограниченное замедление задачи $s_{огр.\max}$. 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.

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

6. Описание технологии экспериментального исследования алгоритмов планирования задач с помощью симулятора вычислительного кластера и его управляющей системы

Опишем технологию экспериментального исследования алгоритмов планирования с помощью симулятора кластера и его управляющей системы, которая используется в данной работе.

Исследование состоит из нескольких сценариев, каждый из которых определяет вид используемого кластера. Например, они могут отличаться однородность аппаратной конфигурации, наличием локальной загрузки узлов. Каждый сценарий предполагает сбор значений метрик по каждой из $N_{тр.}$ трасс, которые затем усредняются для каждого алгоритма планирования. Трасса состоит из фиксированного числа задач n, параметры которых формируются согласно модели вычислительной загрузки.

Предполагается, что в первую очередь алгоритмы планирования (сочетание алгоритма выбора задачи, метода назначения и критерия доступности узлов) анализируются по критерию производительности.

Исследование с помощью симулятора кластера для одного сценария включает следующие шаги:

1. Выбрать и фиксировать критерий доступности узлов.
2. Выполнить симуляцию по всем трассам сценария для всех планировщиков. Получить усредненные значения метрик для каждого планировщика и каждого значения системной загрузки $L \in \{10\%, 20\%, ..., 100\%\}$.
3. Выбрать очередную метрику критерия производительности расписания ($U_{проц.}$, CL, $C_{\max}$, $\bar{t}_{ож.}$ или $\bar{s}_{огр.}$).
4. Для данной метрики и для каждого алгоритма выбора задач проанализировать его всевозможные варианты сочетания с методами назначения задач на узлы и в каждом случае выбрать лучший вариант. Удобно изображать в одной системе координат графики зависимости метрики от системной загрузки для всевозможных сочетаний алгоритма выбора с методами назначения.
5. Сопоставить графики лучших вариантов по данной метрике в одной системе координат. Сделать выводы о наиболее эффективных по данной метрике сочетаниях.
6. Если есть нерассмотренная метрика, то перейти на шаг 3.
7. Проанализировать наиболее эффективные по метрикам критерия производительности сочетания и на предмет использования одинаковых методов назначения и алгоритмов выбора задач и, учитывая, что метрики $U_{проц.}$, CL и $C_{\max}$ более приоритетны, выбрать лучшие алгоритмы планирования.
8. Для лучших алгоритмов планирования произвести сравнительный анализ по метрикам остальных вторичных критериев. При построении их графиков по некоторой вторичной метрике возможно рассмотрение также графиков сочетаний, которые являются лучшими среди комбинаций алгоритма выбора задач с методами назначения на узлы по этой метрике. Это позволит выявить сочетания, которые не входят в число лучших алгоритмов, но являются лучшими по отдельным метрикам.

Данная последовательность шагов исследования выполняется для каждого сценария, затем выявляются общие закономерности и делаются выводы. Заметим, что нас в первую очередь будут интересовать значения метрик, когда $L \geq 50\%$, т.к. при меньших значениях L интенсивность потока работ будет низкой и в этих условиях нельзя говорить о преимуществе того или иного алгоритма планирования.

7. Результаты экспериментального исследования алгоритмов планирования задач

Сценарии исследования алгоритмов планирования представлены в таблице 1. Они отличаются однородностью аппаратной конфигурации кластера и наличием локальной загрузки узлов.

В данной таблице также отражены составляющие части алгоритма планирования, которые имеет смысл варьировать для конкретного сценария. При наличии локальной загрузки вычислительных узлов применялся критерий в виде свертки Convolution of Utilizations (CU), а при отсутствии – Local Utilization Free (LUF).

Таблица 1. Сценарии исследования алгоритмов планирования

№ сцен.	Характеристики сценария		Алгоритмы планирования		
	Гомогенный кластер	Локальная загрузка узлов	Алгоритмы выбора работ для назначения	Методы назначения работ на узлы	Критерий доступности узлов
1	+	-	все	FF, RF	LUF
2	+	+	все	FF, LUNF, RF	CU
3	-	-	все	FF, FNF, LUNF, RF	LUF
4	-	+	все	все	CU

Для симуляции за основу была выбрана аппаратная конфигурация кластера Оренбургского государственного университета. Для формирования гетерогенного кластера она была видоизменена. В первом и втором сценарии кластер состоит из 14 узлов, каждый из которых обладает 1Гб оперативной памяти и 40Гб дисковой, все узлы имеют единичную относительную вычислительную скорость. В двух оставшихся сценариях применяется гетерогенный кластер, аппаратная конфигурация которого приведена в таблице 2.

Таблица 2. Аппаратная конфигурация гетерогенного кластера

Объем оперативной памяти, Мб	Объем дисковой памяти, Гб	Относительная вычислительная скорость	Количество узлов
1024	40	1	2
1024	40	1	2
512	40	1	2
512	20	3	2
2048	40	1	2
2048	40	4	2
256	20	4	2

Во втором и четвертом сценарии критерий доступности узлов имеет вид:

$$(M_i - m_i(t))/M_i \cdot 0.3 + (D_i - d_i(t))/M_i \cdot 0.1 + u_i(t) \cdot 0.6 \le 0.5. \qquad (4)$$

Коэффициенты выбраны исходя из соображений балансировки локальной загрузки узла и загрузки от управляющей системы.

В каждом сценарии исследуется 100 различных трасс по 500 задач в каждой.

Для простоты изложения договоримся название алгоритма планирования формировать из сокращений имен алгоритма выбора задачи и метода назначения. Например, алгоритм Most Processors First Served с методом Random First будет именоваться MPFS RF.

Подробно рассмотрим результаты исследования алгоритмов планирования задач на примере критерия производительности для первого сценария – гомогенного кластера без локальной загрузки вычислительных узлов. Согласно таблице 1 были составлены сочетания каждого алгоритма выбора задач с возможными методами назначения First Fit и Random Node First при фиксированном критерии доступности узлов Local Utilization Free. На основе результатов симуляции для каждой такой пары сочетаний в одной системе координат строились графики зависимостей конкретной метрики производительности от L, выбирался лучший из них вариант.

Графики зависимости метрик критерия производительности для всех лучших вариантов сочетаний приведены на рисунке 3.

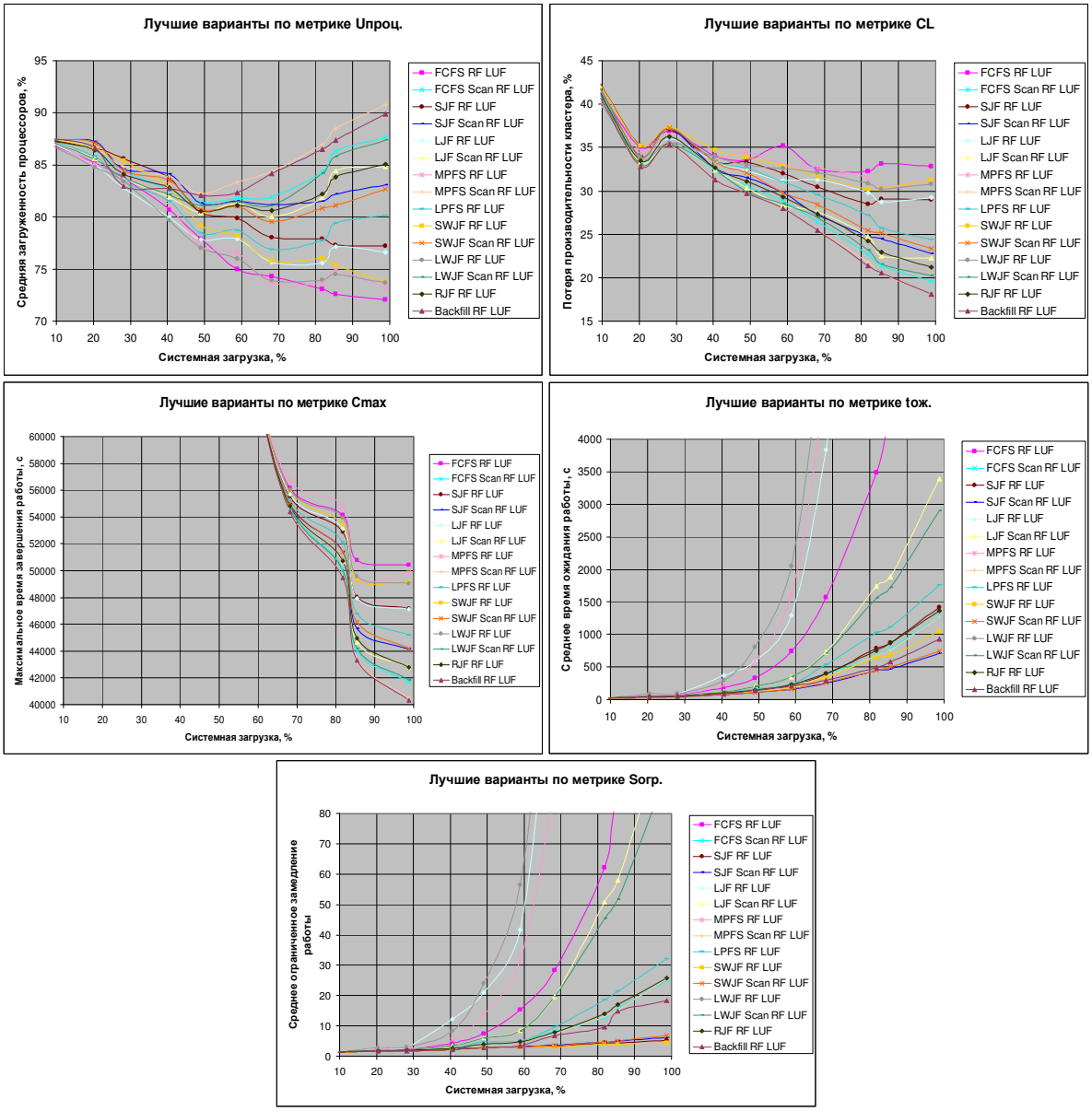

Рис. 3. Графики зависимости метрик критерия производительности от системной загрузки для различных алгоритмов планирования.

Согласно метрике средней загруженности процессора узла кластера $U_{проц.}$ лучшими сочетаниями являются комбинации алгоритмов выбора задач с методом назначения Random First. Наиболее эффективными алгоритмами планирования по данной метрике являются MPFS Scan RF и с незначительным отрывом – Backfill RF. Несколько хуже ведут себя алгоритмы FCFS Scan RF и LWJF Scan RF. Самые плохие результаты показывает алгоритм FCFS RF, что и следовало ожидать.

Метрика потери производительности кластера CL показывает, что Backfill RF и MPFS Scan RF являются наиболее эффективными алгоритмами, для них значения метрики CL практически неразличимы. Чуть хуже – FCFS Scan RF и LWJF Scan RF. Худшую потерю производительности показывает алгоритм FCFS RF.

Лучше всех минимизирует максимальное время завершения работы $C_{\max}$ алгоритмы Backfill RF и MPFS Scan RF, несколько хуже – FCFS Scan RF и LWJF Scan RF. Худшие результаты у FCFS RF и MPFS RF.

Наиболее эффективными по метрике среднего времени ожидания задачи в очереди $\bar{t}_{ож.}$ являются алгоритмы – SJF Scan RF и SWJF Scan RF, т.к. они обрабатывают большое число не-

больших задач быстрее, чем другие алгоритмы обрабатывают небольшое число больших и средних задач. Несколько хуже показывает результаты алгоритм Backfill RF. Результаты алгоритмов MPFS Scan и FCFS Scan практические не отличаются и находятся в середине. Хуже всех – LWJF RF.

Согласно метрике среднего ограниченного замедления задачи $\bar{s}_{огр.}$ очень близкие друг к другу наилучшие результаты демонстрируют алгоритмы SJF RF и SWJF RF, чуть хуже – SJF Scan RF и SWJF Scan RF. Далее идет алгоритм Backfill RF, несколько хуже показывают результаты MPFS Scan и FCFS Scan. Хуже всех – LWJF RF.

Значения всех метрик, кроме $U_{проц.}$, для сочетаний алгоритмов выбора задач с методами назначения First Fit и Random First совпадают. Это связано с тем, что кластер в данном сценарии имеет гомогенную структуру.

Результаты исследования по критерию производительности расписания показывают, что наиболее эффективными по метрикам $U_{проц.}$, CL и $C_{\max}$ являются алгоритмы Backfill RF и MPFS Scan RF, которые имеют очень близкие результаты. По метрикам $\bar{t}_{ож.}$ – SJF Scan RF и SWJF Scan RF, $\bar{s}_{огр.}$ – SJF RF и SWJF RF, в то же время данные алгоритмы демонстрируют плохие результаты по остальным метрикам. По двум последним метрикам Backfill RF и MPFS Scan RF также показывают неплохие результаты, поэтому они являются лучшими алгоритмами планирования по критерию производительности.

Аналогичным образом для первого сценария алгоритмы планирования задач были исследованы по оставшимся критериям.

Во всех четырех сценариях алгоритмы планирования, включающие алгоритмы выбора задач Backfill или MPFS Scan, являются лучшими по основному критерию производительности расписания, причем в первых двух (гомогенный кластер) они также являются лучшими по большинству вторичных критериев, а в оставшихся (гетерогенный кластер) – демонстрируют неплохие результаты.

Алгоритм с Backfill по основному критерию превосходит алгоритм с MPFS Scan во всех сценариях. При гомогенном кластере (первые два сценария) разница между ними небольшая, а при гетерогенном (последние два сценария) – является существенной. Также заметим, что в первых двух сценариях алгоритм MPFS Scan превосходит Backfill по большинству вторичных критериев, а в оставшихся сценариях наоборот. С другой стороны алгоритм Backfill по сравнению с MPFS Scan более сложен в реализации и требует указания пользователем оценок времени выполнения для задач.

В каждом сценарии лучшие результаты в сочетании с алгоритмом выбора демонстрирует свой метод назначения задач на узлы: гомогенный кластер без локальной загрузки узлов – Random First, гомогенный кластер с локальной загрузкой – Least Utilized Node First, гетерогенный кластер при наличии или отсутствии локальной загрузки – Fastest Node First.

По критерию используемости оперативной и дисковой памяти наиболее эффективными являются сочетания, использующие алгоритмы выбора Backfill или MPFS Scan.

Исследование алгоритмов планирования по критерию сбалансированности показывает следующие результаты: лучшие алгоритмы выбора по метрике Du – SJF Scan (первый сценарий при L не больше 50-70%), MPFS Scan (первый при L не меньше 50-70% и второй), Backfill (второй при L не больше 50-70%), FCFS Scan (второй при L не больше 50-70% и четвертый), FCFS (третий и четвертый); по метрикам Dm и Dd – SJF Scan (первый сценарий при L не больше 50-70%), MPFS Scan (первый при L не меньше 50-70%), Backfill (второй), FCFS Scan (второй), SJF (второй), FCFS (третий и четвертый).

Наилучшую гарантированность демонстрируют алгоритмы SWJF и SJF, которые могут быть использованы для построения систем реального времени.

Наилучшую честность по отношению к задачам обеспечивают следующие алгоритмы выбора (метрика $Dt_{ож.}$): MPFS Scan и Backfill в случае первого и второго сценария (гомогенный кластер); FCFS Scan – четвертый сценарий при L не больше 50-60%; FCFS –третий сценарий при L не больше 50-60%; SJF Scan – четвертый и третий сценарии при L не меньше 50-60%.

8. Заключение

В рамках данной работы была предложена имитационная схема кластера, разработана модель его вычислительной загрузки, а также построена система критериев и метрик сравнения различных алгоритмов планирования. Все они легли в основу симулятора вычислительного кластера и его управляющей системы – основного инструмента настоящего исследования.

Лучшим является алгоритм планирования, использующий Backfill, несколько хуже показывает себя MPFS Scan. Однако, Backfill более сложен в реализации и требует наличия достаточно точных оценок времени выполнения задач. Также для каждого сценария может быть рекомендован свой метод назначения задач и критерий доступности узлов для планирования.

Данное исследование будет продолжено в будущем за счет учета топологии вычислительного кластера, коммуникационных задержек при передаче данных, а также многопроцессорности вычислительных узлов.

Литература

1. Jones W.M., Pang L.W. Beowulf Mini-grid Scheduling. [Электронный ресурс] http://www.parl.clemson.edu/beosim.
2. Aida K., Kasahara H., Narita S. Job Scheduling Scheme for Pure Space Sharing among Rigid Jobs. //Lecture Notes In Computer Science, Proceedings of the Workshop on Job Scheduling Strategies for Parallel Processing, Vol. 1459. – London: Springer-Verlag, 1998. – p. 98–121.
3. Feitelson G. Utilization and Predictability in Scheduling the IBM SP2 with Backfilling. //12th International Parallel Processing Symposium / 9th Symposium on Parallel and Distributed Processing. – Orlando: Springer, 1998. – p. 542–546.
4. Feitelson G., Rudolph L., Metrics and Benchmarking for Parallel Job Scheduling. //Job Scheduling Strategies for Parallel Processing. – Orlando: Springer, 1998. – p. 1–24.
5. Feitelson G. Workload Modeling for Computer Systems Performance Evaluation. [Электронный ресурс] http://www.cs.huji.ac.il/~feit/wlmod/
6. Feitelson G. Packing Schemes for Gang Scheduling. //Lecture Notes In Computer Science, Proceedings of the Workshop on Job Scheduling Strategies for Parallel Processing, Vol. 1162. – London: Springer-Verlag, 1996. – p. 89 - 110.
7. Downey A.B. A parallel workload model and its implications for processor allocation. //Cluster Computing, Volume 1 , Issue 1. – Hingham: Kluwer Academic Publishers, 1997. – p. 133 - 145.
8. Jann J. Modeling of Workload in MPPs. //Lecture Notes In Computer Science, Vol. 1291. – London: Springer-Verlag, 1997. – p. 95 - 116.
9. Lublin U. Feitelson G. The workload on parallel supercomputers: modeling the characteristics of rigid job. //Journal of Parallel and Distributed Computing archive, Volume 63 , Issue 11. – Orlando: Academic Press, 2003. – p. 542-546.
10. Blazewicz J., Ecker K., Pesch E., Schmidt G., Weglarz J. Handbook on Scheduling. From Theory to Applications. – Berlin: Springer, 2007. – 647 p.
11. Leung J. Y. Handbook of Scheduling. Algorithms, Models and Performance Analysis. – Boca Raton: CRC Press, 2004. – 622 p.
12. Коффман Э.Г. Теория расписаний и вычислительные машины. – М.: Наука, 1984. – 336 с.
13. Топорков В.В. Модели распределенных вычислений. – М.: ФИЗМАТЛИТ, 2004. – 320 с.
14. В.Н. Коваленко, Е.И. Коваленко, Д.А. Корягин, Д.А. Семячкин. Управление параллельными заданиями в гриде с неотчуждаемыми ресурсами. [Электронный ресурс] http://www.keldysh.ru/papers/2007/source/prep2007_63.doc