ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ

УДК 519.687

ИССЛЕДОВАНИЕ АЛГОРИТМОВ ПЛАНИРОВАНИЯ ПАРАЛЛЕЛЬНЫХ ЗАДАЧ ДЛЯ КЛАСТЕРНЫХ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ С ПОМОЩЬЮ СИМУЛЯТОРА

В.П. Гергель [1], ***П.Н. Полежаев*** [2]

[1] Нижегородский госуниверситет им. Н.И. Лобачевского
[2] Оренбургский госуниверситет

peter.polezhaev@mail.ru

Поступила в редакцию 28.03.2010

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.

Ключевые слова: алгоритмы планирования, кластерные вычислительные системы, симулятор кластера, имитационное моделирование.

Введение

Алгоритмы планирования, используемые в существующих управляющих системах, обладают целым рядом недостатков – они не в состоянии обеспечить высокую вычислительную загрузку узлов, не всегда учитывают возможную гетерогенность кластера, а также возможность локальной загрузки узлов, характерную для кластеров рабочих станций.

Современные исследования [1–4], рассматривающие различные списочные алгоритмы планирования, недостаточно полные, т.к. не затрагивают всех основных существующих алгоритмов или рассматривают достаточно ограниченное количество анализируемых метрик. Недостаточно полно изучается поведение алгоритмов планирования на кластерах рабочих станций, на кластерах со значительной степенью гетерогенности аппаратной платформы.

Данная работа призвана частично исправить сложившуюся ситуацию. В нашем исследовании проводится достаточно полный сравнительный анализ всех основных списочных алгоритмов планирования с помощью широкой системы критериев и метрик. Рассматриваются четыре сценария: гомогенный кластер при наличии или отсутствии локальной загрузки узлов и гетерогенный кластер при наличии или отсутствии локальной загрузки вычислительных узлов.

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.

Имитационная схема и модель кластера

На рис. 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 выполнение 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 алго-

ритмом на требуемое количество доступных вычислительных узлов – подмножество свободных каналов K_1, K_2, …, K_m.

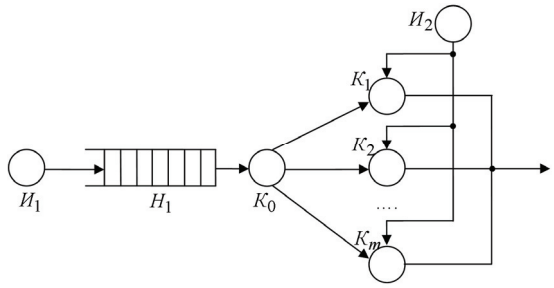

Рис. 1. Имитационная схема кластера

Каждый вычислительный узел K_i имеет M_i Кбайт оперативной памяти и D_i Кбайт дисковой памяти, характеризуется относительной вычислительной скоростью S_i. 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 $t \in [0;\infty)$.

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

Модель вычислительной загрузки кластера

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

Пусть $J = (J_1,...,J_n)$ – последовательность (поток) поступающих в очередь задач, упорядоченная по возрастанию времени их прибытия $a_j \in [0;\infty)$.

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 $\widetilde{p}_j$, 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 $p_j = \dfrac{\widetilde{p}_j}{\min\limits_{i \in N_j} S_i}$.

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 τ_j с учетом различных скоростей назначенных ей узлов.

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 всего только a_j, n_j, τ_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 параллельную задачу.

Требуемое количество процессоров n_j. Генерация значений n_j осуществляется следующим образом [9]. 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 $p = 0.3$.

Реальное время выполнения задачи $\tilde{\tau}_j$. Логарифм реального времени выполнения задачи имеет гипер-гамма-распределение, параметр p которого линейно зависит от n_j [9]. При исследовании использовались следующие усредненные значения параметров распределения: $\alpha_1 = 6.57$, $\beta_1 = 0.823$, $\alpha_2 = 639.1$, $\beta_2 = 0.0156$, $p = -0.003n_j + 0.6986$.

Оценка пользователя времени выполнения задачи $\tilde{p}_j$ имеет равномерное распределение на интервале $[\tilde{\tau}_j; f\tilde{\tau}_j]$, где $f \in [1;4]$ – константный показатель, отражающий неточность оценок пользователей [3]. Для исследования было принято значение $f = 2$.

Интервалы времени между поступлением задач в очередь $a_{j+1} - a_j$ имеют экспоненциальное распределение [2, 5, 9] с интенсивностью $\lambda = \frac{1}{15}$.

Требования к объему оперативной памяти mr_j *и к объему дисковой памяти* dr_j узлов генерировались на основе равномерного распределения, при этом обеспечивалось наличие в кластере необходимого числа процессоров n_j.

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 усложнило бы 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, имеющие гамма-распределения с соответствующими параметрами (α_m, β_m), (α_d, β_d) и (α_u, β_u). Обозначим 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 интенсивностями λ_m, λ_d и λ_u, причем $\lambda_u > \lambda_m > \lambda_d$.

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

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

$$m_i(t) = \begin{cases} M_i - (M_i - mr_j)\psi_{mi}(t) - mr_j, \\ \quad \text{если на } P_i \text{ выполняется } j \text{ - я работа}, \\ M_i - M_i\psi_{mi}(t), \quad \text{иначе,} \end{cases} \tag{2}$$

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

Для целей симуляции были выбраны следующие значения параметров: $\alpha_m = 0.2$, $\beta_m = 0.3$, $\alpha_d = 0.2$, $\beta_d = 0.2$, $\alpha_u = 0.5$, $\beta_u = 0.5$, $\lambda_u = 0.08$, $\lambda_m = 0.04$ и $\lambda_d = 0.02$.

Исследуемые алгоритмы планирования задач

В структуре алгоритма планирования можно выделить следующие части: алгоритм выбора задачи для назначения на узлы, метод назначения задачи на свободные вычислительные 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. Критерий доступности узла представляет собой логическое выражение, завязанное на статических и динамических параметрах узла, он особенно актуален для кластера рабочих станций.

В данной работе рассматриваются следующие эвристические списочные алгоритмы выбора задач для назначения [1–4; 10–14]: First Come First Served (FCFS, первым пришел – первым обслужен), First Come First Served Scan (FCFS Scan, первым пришел – первым вышел со сканированием очереди), 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).

В рамках данного исследования рассматриваются следующие методы назначения задач на доступные вычислительные узлы кластера [10–14]: First Fit (FF, первый подходящий), Best Fit (BF, наилучший подходящий), Fastest Node First (FNF, самый быстрый узел первый), Least Utilized Node First (LUNF, наименее загруженный узел первый), Random First (RF, случайный узел первый).

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. Она включает следующие критерии:

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 $\bar{m}$ и $\bar{d}$, определяющие соответственно средние загруженности оперативной и дисковой памяти вычислительных узлов. Позволяет выявить процент неиспользованных ресурсов.

3. Сбалансированность загрузки узлов. Включает метрики Du, Dm и Dd, обозначающие соответственно дисперсии средней 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 максимального ограниченного замедления 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.

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

Результаты экспериментального исследования алгоритмов планирования задач

Сценарии исследования алгоритмов планирования представлены в табл. 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

№ сценария	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

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

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

Во втором и четвертом сценариях критерий доступности узлов имеет вид:

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

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, генерируемых согласно модели вычислительной загрузки. Значения вычисляемых метрик усреднялись по всем трассам.

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 рис. 2.

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.

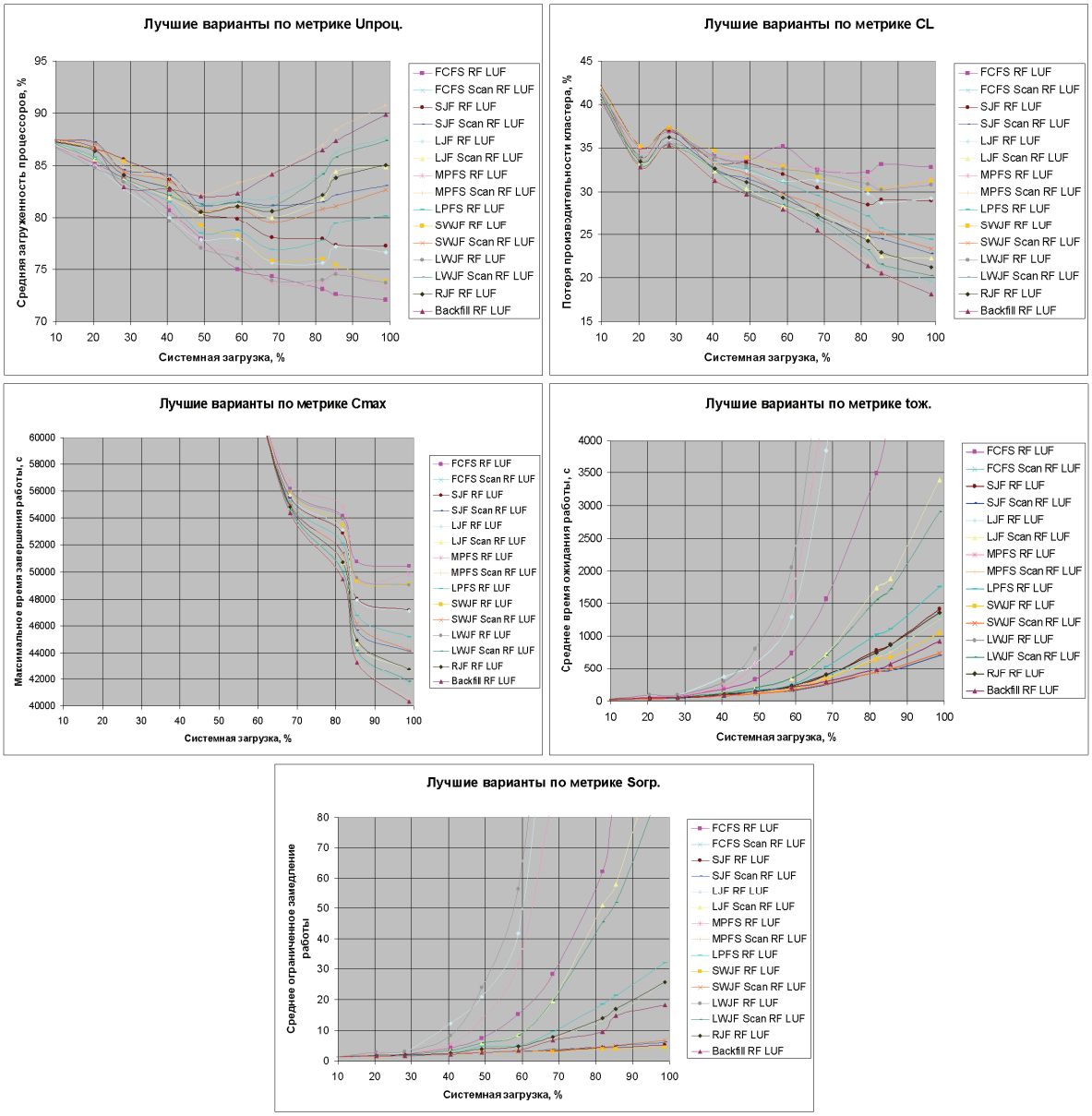

Рис. 2. Графики зависимости метрик критерия производительности от системной загрузки для различных алгоритмов планирования

Лучше всех минимизируют 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 небольших задач быстрее, чем другие алгоритмы обрабатывают небольшое число больших и средних задач. Несколько худшие результаты показывает алгоритм Backfill RF. Результаты алгоритмов MPFS Scan и FCFS Scan практически 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 худшие результаты показывают 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 вторичным 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 сценариев (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%.

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 за счет учета топологии вычислительного кластера, коммуникационных задержек при передаче данных, а также многопроцессорности 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).

Список литературы

1. Jones W.M., Pang L.W. Beowulf Mini-grid Scheduling [Электронный ресурс]. – Режим доступа: 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. V. 1459. L.: 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]. – Режим доступа: 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. V. 1162. L.: Springer-Verlag, 1996. P. 89–110.

7. Downey A.B. A Parallel Workload Model and its Implications for Processor Allocation // Cluster Computing. V. 1. I. 1. Hingham: Kluwer Academic Publishers, 1997. P. 133–145.

8. Jann J. Modeling of Workload in MPPs // Lecture Notes in Computer Science. V. 1291. L.: 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. 2003. V. 63. № 11. P. 542–546.

10. Blazewicz J., Ecker K., Pesch E. et al. Handbook on Scheduling. From Theory to Applications. B.: 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. Коваленко В.Н., Коваленко Е.И., Корягин Д.А., Семячкин Д.А. Управление параллельными заданиями в гриде с неотчуждаемыми ресурсами [Электронный ресурс]. – Режим доступа: http://www.keldysh.ru/papers/2007/source/prep2007_63.doc

THE STUDY OF PARALLEL JOB SCHEDULING ALGORITHMS FOR CLUSTER COMPUTING SYSTEMS USING A SIMULATOR

V.P. Gergel, P.N. Polezhaev

The results of the experimental study of various job scheduling algorithms for a computing cluster obtained by a program cluster simulator and its control system are described. The cluster workload management model and its imitation scheme have been elaborated. The criteria and metrics to compare scheduling algorithms have been worked out.

Keywords: scheduling algorithms, cluster computing systems, cluster simulator, simulation modeling.