Оценка характеристик алгоритма Backfill при управлении потоком задач на вычислительном кластере*

В. В. Мазалов, Н. Н. Никитина
Институт прикладных математических исследований Карельского научного центра РАН, Петрозаводск, Россия
e-mail: vmazalov@krc.karelia.ru, nikitina@krc.karelia.ru

Исследуются характеристики алгоритма планирования Backfill при управлении потоком задач на вычислительном кластере. Принятие решения о применении процедуры Backfill осуществляется с использованием аналитического выражения вероятности ошибки. Рассматриваемыми параметрами эффективности алгоритма являются среднее время ожидания задачи в очереди и количество ошибок при применении процедуры Backfill. Приводятся выводы о зависимости эффективности алгоритма планирования от характеристик потока задач.

Ключевые слова: алгоритм Backfill, вычислительный кластер, расписание задач.

Введение

Для решения ресурсоёмких многопроцессорных вычислительных задач в настоящее время широко применяются вычислительные кластеры. Вычислительный кластер представляет собой множество вычислительных узлов, объединённых высокоскоростными каналами связи в единую программно-аппаратную систему. С точки зрения пользователя кластер является единым вычислительным ресурсом [1, 2].

Для эффективного использования мощностей кластера необходимо управлять очередью заданий, т. е. определять порядок запуска задач и распределять между ними вычислительные ресурсы. Данные функции выполняет планировщик задач, являющийся важной составляющей вычислительного кластера. Планировщик оперирует такими характеристиками задач как приоритет, ожидаемое время выполнения, требуемое количество процессоров, объём оперативной памяти и дискового пространства и др.

Существует множество алгоритмов планирования, направленных на оптимизацию использования вычислительных ресурсов по разным параметрам с учётом различных характеристик задач, которые могут быть как предопределёнными, так и недетерминированными. Вопросам повышения производительности существующих алгоритмов планирования, сравнения и создания новых алгоритмов посвящён ряд работ (см., например, [3 – 5] и др.). В [4] приводится описание и сравнение алгоритмов планирования, учитывающих способность задач к изменению объёма требуемых ресурсов и времени выполнения задач, в [6] сформулирована задача распределения ресурсов распределённой многопроцессорной вычислительной системы как задача многокритериальной

*Работа выполнена при финансовой поддержке Отделения математических наук РАН, РФФИ (грант № 10-01-00089-а) и Программы стратегического развития ПетрГУ.

оптимизации и предложен ряд алгоритмов планирования. В целом наиболее распространенными базовыми алгоритмами планирования являются FCFS (First Come First Served), RR (Round Robin), SJF (Shortest Jobs First), LJF (Longest Jobs First). На их основе разрабатываются более сложные алгоритмы. В частности, широко используемым и одним из наиболее эффективных по ряду критериев является алгоритм Backfill. В работе [7] впервые была рассмотрена возможность предсказания распределения времени выполнения задач вместо точечных оценок для алгоритма планирования Backfill.

Цель настоящего исследования — получение оценок характеристик алгоритма Backfill при управлении потоком задач на вычислительном кластере. Для оценки времени выполнения предлагается применять параметры распределения, полученные анализом истории выполнения задач. Принятие решения при управлении очередью задач осуществляется с использованием аналитического выражения вероятности ошибки.

1. Алгоритм Backfill

Одним из простейших алгоритмов планирования является FCFS (First Come First Served). При его реализации задачи запускаются в том же порядке, в котором они были поставлены в очередь, как только освобождается достаточное количество ресурсов кластера. Для оптимизации алгоритма FCFS (а также других алгоритмов планирования) может быть использована техника Backfill. Базовый вариант алгоритма Backfill был впервые описан в работе [8] в рамках системы планирования задач на суперкомпьютере IBM SP1 Аргоннской Национальной лаборатории (США).

В ходе работы алгоритма Backfill обрабатывается очередь задач, ожидающих запуска на выполнение. Задачи в очереди упорядочены по времени их регистрации в системе. При каждой регистрации и каждом завершении задачи происходит выполнение следующих шагов алгоритма:

1) при наличии необходимого количества свободных процессоров для запуска первой в очереди задачи эта задача удаляется из очереди и запускается на выполнение;

2) если для запуска первой в очереди задачи свободных процессоров не хватает, то вычисляется момент времени, когда их станет достаточно, и производится резервирование для данной задачи;

3) продолжается движение по очереди с запуском на выполнение задач, не нарушающих резервирование первой в очереди задачи.

Задачи, требующие небольшого числа процессоров или процессорного времени, могут запускаться на выполнение вне очереди при наличии необходимого количества доступных ресурсов. При этом их запуск не должен задерживать выполнение предыдущих по очереди задач (рис. 1). Таким образом удаётся задействовать вычислительные ресурсы, простаивающие во время выполнения первоочередных задач.

Ожидаемое время выполнения задачи в базовом алгоритме выражено точечной оценкой, что позволяет повысить производительность алгоритма FCFS при условии, что оценка является достаточно точной [9]. Однако в связи со значительным количеством факторов, влияющих на ход вычислительного процесса, получить точную оценку времени выполнения, как правило, сложно. В частности, выполнение задач может быть прервано из-за разного рода ошибок, пользователи могут отменять задачи или запрашивать значительно большее время, чем необходимо в действительности и др. В работе [7] точечная оценка заменяется оценкой распределения времени выполнения задач и приводятся результаты экспериментов, показавшие повышение производительности

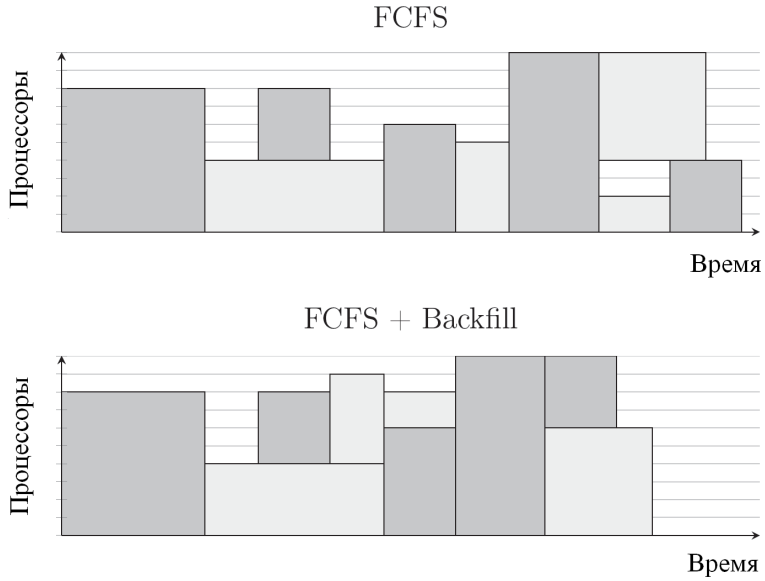

Рис. 1. Управление очередью заданий с использованием алгоритма Backfill

базового алгоритма Backfill. При этом условие принятия решения о запуске задачи, осуществляемого на шаге 2 и 3, заменяется условием: "вероятность задержки момента запуска первой задачи из очереди (вероятность ошибки) меньше заданного порога τ". Вероятность ошибки вычисляется при помощи методов динамического программирования каждый раз, когда необходимо принять решение о внеочередном запуске задачи.

В настоящей работе представлены математическая модель задачи и аналитическое выражение для оценки вероятности ошибки, предложенной в [7], в предположении, что распределения моментов завершения задач имеют экспоненциальный вид и количество освобождаемых в эти моменты процессоров подчиняется пуассоновскому закону. Предположение экспоненциальности позволяет аналитически найти основные характеристики процесса вычислений для применения алгоритма Backfill. Заметим, что это не всегда согласуется с реальной ситуацией. Однако такой поток задач даёт возможность предварительно оценить эффективность алгоритма благодаря возможности проведения экспериментов с разными значениями параметров модели, видами и значениями параметров распределений моделируемых случайных величин. В работе [10] приводятся результаты исследования, показавшие, что даже весьма упрощённые модели потоков задач, как правило, позволяют получить те же выводы об относительной эффективности алгоритмов планирования, что и более адекватные реальным системам модели, воссозданные из истории функционирования используемых вычислительных систем.

2. Математическая модель

Пусть в настоящий момент времени свободно c_0 процессоров вычислительного кластера, в очереди находится N задач J_1, J_2, ..., J_N, упорядоченных по времени их регистрации и для запуска первой задачи в очереди J_1 требуется c_q свободных процессоров и $c_q > c_0$, т. е. свободных процессоров для её запуска в настоящий момент недостаточно. Согласно алгоритму Backfill необходимо принять решение о запуске некоторой задачи в очереди J_i, $1 < i \leqslant N$, для запуска которой свободных процессоров достаточно: $c_i \leqslant c_0$. Пусть

t_e — ожидаемое время выполнения задачи J_i. Если в течение интервала времени длиной t_e освободится число процессоров, достаточное для запуска задачи J_1, то немедленный запуск J_i может задержать запуск J_1. Согласно алгоритму решение запустить задачу J_i принимается, если вероятность этого события меньше заданного порога τ. Необходимо найти аналитическое выражение вероятности задержки запуска задачи J_1 в случае принятия решения о немедленном запуске задачи J_i.

Число свободных процессоров в дискретный момент времени t_i, $i \geqslant 0$, описывается случайным процессом

$$S(t_i) = S(t_{i-1}) + \xi_i,$$

где ξ_i — число процессоров, освобождённых в момент i-го скачка, $S(0) = S_0 = 0$. Предполагается, что величины ξ_1, ξ_2, ... распределены экспоненциально с параметром μ, а моменты скачков (завершения задач) — экспоненциально с параметром λ.

Таким образом, необходимо найти аналитическое выражение

$$P(\exists t_i : c_q \leqslant S(t_i) < c,\ c = c_q + c_i) \tag{1}$$

вероятности существования момента скачка, в который число свободных процессоров окажется достаточным для запуска первой задачи очереди J_1, но недостаточным для одновременной работы J_1 и рассматриваемой задачи J_i (рис. 2). В предположении, что число скачков N на интервале $[0; t_e]$ имеет пуассоновское распределение с параметром λt_e, вероятность существования момента скачка, в который число свободных процессоров окажется достаточным для запуска J_1, но недостаточным для одновременной работы J_1 и J_i, может быть выражена следующим образом:

$$P(\exists t_i : c_q \leqslant S(t_i) < c) = \sum_{k=1}^{\infty} P(N = k) \sum_{n=1}^{k} P(S_{n-1} < c_q, c_q \leqslant S_n < c) =$$

$$= \sum_{k=1}^{\infty} \frac{(\lambda t_e)^k}{k!} e^{-\lambda t_e} \sum_{n=1}^{k} P(S_{n-1} < c_q, c_q \leqslant S_n < c).$$

Вероятность $P(S_{n-1} < c_q, c_q \leqslant S_n < c)$ того, что после $(n-1)$-го скачка свободных процессоров недостаточно для запуска J_1, а после n-го скачка их станет достаточно для

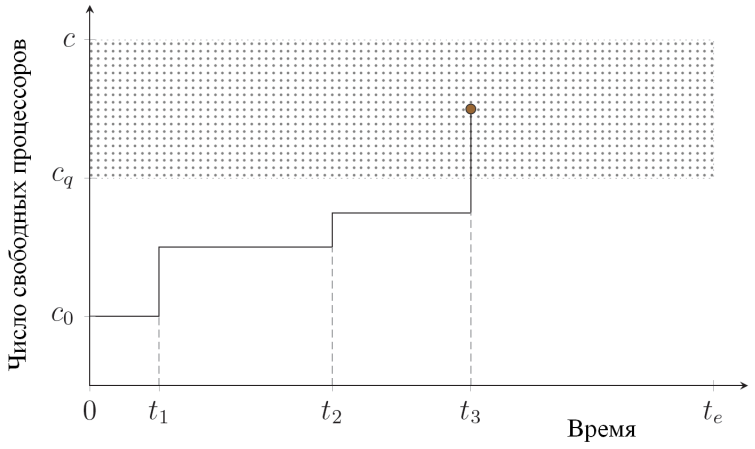

Рис. 2. Задержка запуска первой задачи в очереди

запуска J_1, но недостаточно для одновременной работы J_1 и J_i, может быть вычислена рекурсивно.

Вероятность того, что в начальный момент времени свободных процессоров недостаточно для запуска J_1, а после первого скачка их будет достаточно для запуска J_1, но недостаточно для одновременной работы J_1 и J_i

$$P(S_0 < c_q, c_q \leqslant S_1 < c) = P(c_q \leqslant S_1 < c) = P(c_q \leqslant \xi_1 < c) =$$
$$= F_{\xi_1}(c) - F_{\xi_1}(c_q) = 1 - e^{-\mu c} - 1 + e^{-\mu c_q} = e^{-\mu c_q} - e^{-\mu c}.$$

Вероятность того, что после первого скачка свободных процессоров недостаточно для запуска J_1, а после второго скачка их станет достаточно для запуска J_1, но недостаточно для одновременной работы J_1 и J_i

$$P(S_1 < c_q, c_q \leqslant S_2 < c) = P(\xi_1 < c_q, c_q \leqslant \xi_1 + \xi_2 < c) =$$
$$= \int_0^{c_q} f_{\xi_1}(x_1) \int_{c_q - x_1}^{c - x_1} f_{\xi_2}(x_2)\, dx_2\, dx_1 = \int_0^{c_q} \mu e^{-\mu x_1} \int_{c_q - x_1}^{c - x_1} \mu e^{-\mu x_2}\, dx_2\, dx_1 =$$
$$= \mu \left(e^{-\mu c_q} - e^{-\mu c}\right) \int_0^{c_q} dx_1 = \mu \left(e^{-\mu c_q} - e^{-\mu c}\right) c_q.$$

Вероятность того, что после второго скачка свободных процессоров недостаточно для запуска J_1, а после третьего скачка их будет достаточно для запуска J_1, но недостаточно для одновременной работы J_1 и J_i

$$P(S_2 < c_q, c_q \leqslant S_3 < c) = P(\xi_1 < c_q, \xi_1 + \xi_2 < c_q, c_q \leqslant \xi_1 + \xi_2 + \xi_3 < c) =$$
$$= \int_0^{c_q} f_{\xi_1}(x_1) \int_0^{c_q - x_1} f_{\xi_2}(x_2) \int_{c_q - x_1 - x_2}^{c - x_1 - x_2} f_{\xi_3}(x_3) dx_3 dx_2 dx_1 = \frac{\mu^2 c_q^2}{2} \left(e^{-\mu c_q} - e^{-\mu c}\right).$$

Продолжая те же рассуждения, получим

$$P(S_{n-1} < c_q, c_q \leqslant S_n < c) = \frac{(\mu c_q)^{n-1}}{(n-1)!} \left(e^{-\mu c_q} - e^{-\mu c}\right). \tag{2}$$

Следовательно, аналитическое выражение вероятности (1) имеет следующий вид:

$$P(\exists t_i : c_q \leqslant S(t_i) < c) =$$
$$= \sum_{k=1}^{\infty} \frac{(\lambda t_e)^k}{k!} e^{-\lambda t_e} \sum_{n=1}^{k} \frac{(\mu c_q)^{n-1}}{(n-1)!} \left(e^{-\mu c_q} - e^{-\mu c}\right) =$$
$$= e^{-\lambda t_e} \left(e^{-\mu c_q} - e^{-\mu c}\right) \sum_{n=1}^{\infty} \frac{(\mu c_q)^{n-1}}{(n-1)!} \sum_{k=n}^{\infty} \frac{(\lambda t_e)^k}{k!}. \tag{3}$$

В следующем разделе представлены результаты экспериментов, проведённых на имитационной модели в целях исследования характеристик алгоритма Backfill с использованием полученного выражения вероятности ошибки.

Следует заметить: эксперименты на реальных данных показывают, что распределение необходимых задачам ресурсов хорошо приближается усечённым распределением Парето [11]. В этом случае выражение для соответствующей вероятности будет иметь более громоздкий вид, хотя принципиальных изменений в работе алгоритма не произойдет. Ниже представлены результаты экспериментов, проведённых на потоках из 1000 задач, смоделированных соответственно с экспоненциальным распределением и усечённым распределением Парето при одинаковых значениях параметров:

Вид распределений в модели	ΔT_w	Доля применений Backfill	Доля ошибок
Экспоненциальное	0.27	0.16	0.02
Усечённое Парето	0.23	0.14	0.03

Видно, что характеристики алгоритма для обеих моделей оказались приблизительно одинаковыми.

3. Эксперименты

Для исследования характеристик алгоритма Backfill реализована имитационная модель управления потоком задач на вычислительном кластере. В качестве входных параметров этой модели были приняты следующие величины, полученные статистическим анализом истории выполнения задач на вычислительном кластере Карельского научного центра РАН [12]: $\mu = 0.10493$ — параметр распределения числа процессоров, запрашиваемого задачами; $\lambda_a = 0.00944$ — параметр распределения интервалов времени между приходами задач; $\alpha = 0.0048$ — параметр распределения времени выполнения задач.

Моделирование потока задач производилось на вычислительном кластере с 64 процессорами. Один и тот же поток из 1000 задач, смоделированный с указанными входными параметрами, сначала обрабатывался с использованием дисциплины обслуживания FCFS, затем — процедуры Backfill. Было подсчитано среднее время ожидания задачи в очереди, количество применений Backfill и количество ошибок (задержек первоочередных задач) в этом случае. Затем значения входных параметров изменялись и эксперименты повторялись для исследования влияния входных параметров на производительность алгоритма Backfill.

На рис. 3, 4 приведены результаты серии экспериментов. Использованы следующие обозначения: T_{wait}^{FCFS}, $T_{wait}^{\text{Backfill}}$ — среднее время ожидания задачи в очереди соответственно при дисциплине обслуживания FCFS и с применением процедуры Backfill; $\Delta T_{wait} = \dfrac{T_{wait}^{\text{FCFS}} - T_{wait}^{\text{Backfill}}}{T_{wait}^{\text{FCFS}}}$ — относительное изменение среднего времени ожидания задачи в очереди; $\bar{t}_e$ — среднее время выполнения задачи; $\tilde{p}$ — средняя доля процессоров кластера, занимаемых задачей.

Эксперименты показали, что в целом использование алгоритма планирования Backfill с принятием решения на основе вероятности ошибки по сравнению с алгоритмом планирования FCFS позволяет снизить среднее время ожидания задач в очереди (см. рис. 3). Выигрыш в τ имеет прямую зависимость как от среднего количества процессоров $\tilde{p}$, требуемых задачей, так и от среднего времени выполнения задачи $\bar{t}_e$. Однако с увеличением этих параметров возрастает и количество ошибок. На рис. 4 представлены зависимости долей успешных и ошибочных применений Backfill среди общего количества задач от соответствующих характеристик вычислительного процесса. При

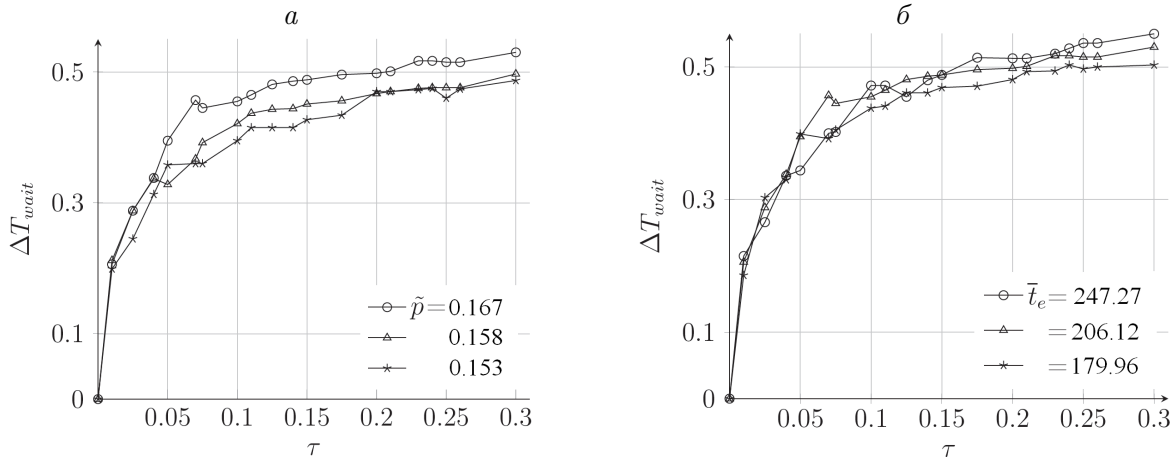

Рис. 3. Изменение среднего времени ожидания в зависимости от порогового значения τ; *а* — $\overline{t}_e = 206.12$ мин, *б* — $\tilde{p} = 0.17$; доля ошибок не более 4 %

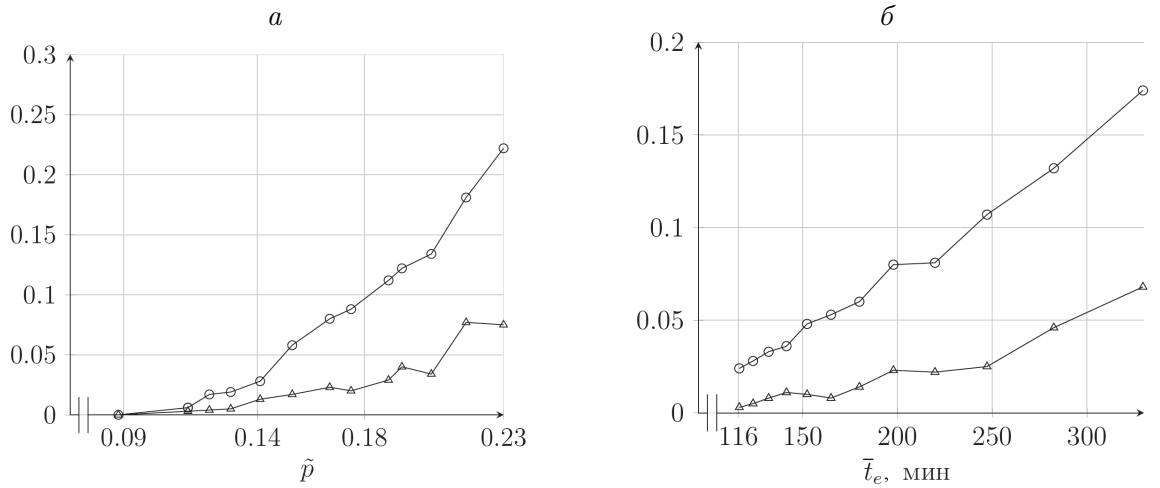

Рис. 4. Доля применений Backfill (○) и доля ошибок (△) в зависимости от среднего числа процессоров и среднего времени выполнения одной задачи; *а* — $\overline{t}_e = 206.12$ мин, *б* — $\tilde{p} = 0.17$; $\tau = 0.2$

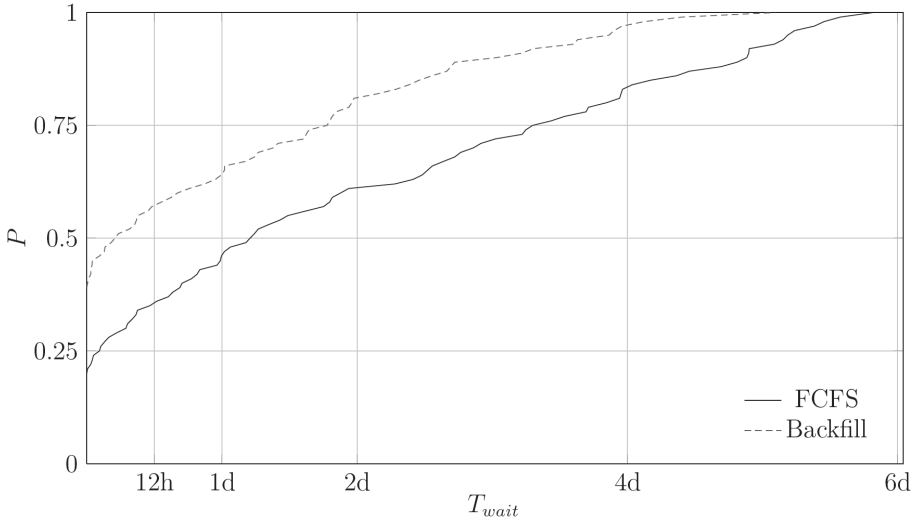

Рис. 5. Эмпирическая функция распределения времени ожидания задачи в очереди

использовании в имитационной модели значений параметров, полученных статистическим анализом истории выполнения задач на вычислительном кластере КарНЦ РАН, среднее время ожидания снижается более чем в 2 раза, при этом количество ошибок не превышает 4 % (см. рис. 3).

В качестве критерия эффективности алгоритма также может быть рассмотрена эмпирическая функция распределения времени ожидания (рис. 5), которая позволяет сделать выводы об улучшении производительности системы в целом. В данном случае доля задач, время ожидания для которых уменьшилось благодаря использованию предложенного алгоритма, составила 100 %.

Заключение

Алгоритм планирования задач Backfill повышает эффективность базового алгоритма FCFS при условии наличия точных оценок времени выполнения задач. Однако на практике точные оценки не всегда могут быть получены. В [7] предлагается использовать оценку распределения времени выполнения задач и на этапе принятия решения вычислять вероятность ошибки методами динамического программирования. В настоящей работе представлено аналитическое выражение для оценки вероятности ошибки. Для исследования характеристик алгоритма Backfill с принятием решения на основе вероятности ошибки реализована имитационная модель управления потоком задач на вычислительном кластере. На первом этапе экспериментов для моделирования использованы параметры распределений, полученные статистическим анализом истории выполнения задач на вычислительном кластере КарНЦ РАН. Исследуемыми параметрами эффективности алгоритма планирования являются среднее время ожидания задачи в очереди и количество ошибок при применении процедуры Backfill. Сделаны выводы о зависимости эффективности алгоритма планирования от характеристик потока задач.

Список литературы

[1] Воеводин В.В., Жуматий С.А. Вычислительное дело и кластерные системы. М.: Изд-во Московского гос. ун-та, 2007. 150 с.

[2] Narayan A. High-performance Linux clustering. Part 1: Clustering fundamentals. IBM developerWorks, 2005. URL: http://www.ibm.com/developerworks/linux/library/l-cluster1

[3] Полежаев П.Н. Исследование алгоритмов планирования параллельных задач для кластерных вычислительных систем с помощью симулятора // Сб. тр. Междунар. науч. конф. “Параллельные вычислительные технологии-2010”. Уфа, 2010. С. 287–298.

[4] Новиков А.Б. Алгоритмы планирования масштабируемых заданий кластерной вычислительной системы // Молодой ученый. 2011. Т. 1, № 11. С. 74–79.

[5] Iqbal S., Gupta R., Fang Y.-C. Planning Considerations for Job Scheduling in HPC Clusters // Dell Power Solutions. 2005. P. 133–136.

[6] Мамойленко С.Н., Ефимов А.В. Алгоритмы планирования решения масштабируемых задач на распределённых вычислительных системах // Вестник ГОУ ВПО “СибГУТИ”. 2010. № 2. С. 66–78.

[7] Nissimov A., Feitelson D.G. Probabilistic Backfilling // Job Scheduling Strategies for Parallel Proc. LNCS. Springer-Verlag, 2007. Vol. 4942. P. 102–115.

[8] Lifka D. The ANL/IBM SP scheduling system // Job Scheduling Strategies for Parallel Proc. LNCS. Springer-Verlag, 1995. Vol. 949. P. 295–303.

[9] Tsafrir D., Etsion Y., Feitelson D.G. Backfilling using system-generated predictions rather than user runtime estimates // IEEE Trans. Parallel and Distributed Systems. 2007. Vol. 18, No. 6. P. 789–803.

[10] Lo V., Mache J., Windisch K. A Comparative study of real workload traces and synthetic workload models for parallel job scheduling // Job Scheduling Strategies for Parallel Proc. LNCS. Springer-Verlag, 1998. Vol. 1459. P. 25–46.

[11] Морозов Е.В., Румянцев А.С. Модели многосерверных систем для анализа вычислительного кластера // Тр. Карельского науч. центра РАН. 2011. № 5. С. 75–85.

[12] Центр высокопроизводительной обработки данных ЦКП КарНЦ РАН / Ин-т прикл. матем. исследований КарНЦ РАН. URL: http://cluster.krc.karelia.ru

Поступила в редакцию 15 марта 2012 г.,
с доработки — 17 апреля 2012 г.