УДК 519.6

ЭФФЕКТИВНОЕ ПРОГНОЗИРОВАНИЕ ВРЕМЕННЫХ ХАРАКТЕРИСТИК ВЫЧИСЛИТЕЛЬНЫХ ЗАДАНИЙ ПО РАСШИРЕННЫМ МЕТАДАННЫМ

С. А. Петунин
(ФГУП "ВНИИА им. Н. Л. Духова", г. Москва)

Проведено исследование применения статистического метода Random Forest, относящегося к группе машинного обучения, для прогнозирования временных показателей вычислительного задания: времени ожидания в очереди и времени выполнения. Все эксперименты проведены на данных большого объема менеджера ресурсов и планировщика заданий Slurm, накопленных им в журналах за различные календарные периоды. Программы обработки данных реализованы в виде скриптов в рамках статистической системы R. Принципиальной идеей работы явилось включение в структуру прогнозных моделей новых уникальных метаданных вычислительных заданий, не используемых ранее. Результаты исследований показали более чем удвоенную эффективность предложенных прогнозных моделей по сравнению с оценками пользователей и планировщика. Работа является первым шагом в проекте внедрения автоматических сервисов управления вычислительными заданиями на уровне метапланировщика в целях автоматизации запросов ресурсов, а также для более качественного оповещения пользователей.

Ключевые слова: прогнозирование, метаданные вычислительного задания, время выполнения, время ожидания, метод Random Forest.

Введение

Проведение регулярных массовых расчетов на высокопроизводительных вычислительных системах (ВВС) характеризуется высокой рабочей нагрузкой. Как следствие, это приводит к значительному времени нахождения в очередях запускаемых вычислительных заданий. Пользователи получают результаты расчетов позже ожидаемого времени, что негативно сказывается на сроках выполнения исследовательских и производственных проектов. Ситуацию не меняет даже постоянный рост производительности вводимых в эксплуатацию новых суперкомпьютеров, несмотря на доступные для заданий пользователей сотни или тысячи вычислительных узлов.

Наличие постоянных очередей в современных кластерах усугубляет известную проблему потери эффективности при попытках планировщика загружать все вычислительные узлы. Широкое вычислительное поле позволяет запускать множество многоузловых заданий. После этого в нем возникает дефрагментация с большой площадью свободных, но не распределенных вычислительных ресурсов в пространстве выполняемых заданий <*узлы, время*>. Их способно "запереть" одно приоритетное задание, ожидающее еще большего ресурса. Применение различных вариантов алгоритма *бэкфиллинга* — обратного заполнения очереди (приоритетного запуска малоресурсных заданий из хвоста очереди) — нельзя признать удовлетворительным для полного заполнения "дыр" и устранения дефрагментации. Проблема в том, что при постановке заданий в очередь пользователи дают приблизительную оценку времени их счета, которая передается планировщику. Малые оценочные значения помогают алгоритму быстрее запускать задания на счет. Однако статистика рабочей нагрузки почти всех суперкомпьютерных центров показывает сильное превышение запрошенного времени выполнения по сравнению с реальным [1]. Это объясняется нежеланием пользователя, чтобы его задание было принудительно снято по событию таймаута. Существенный разброс значений динамических характеристик метаданных

заданий в множестве их запусков также мешает пользователю выбрать подходящий прогноз продолжительности их выполнения.

Научная проблематика с предложениями по оптимизации выделения ресурсов была активно и полно обсуждена еще в начальный период параллельных вычислений [2]. Алгоритм планирования параллельных заданий на базе бэкфиллинга стал популярен, начиная со второй половины 1990-х гг., после его описания в сборнике статей под редакцией профессора Д. Фейтельсона [3]. В последние годы резко увеличилось количество публикаций в этой области в связи с привлечением современных интеллектуальных методов статистического машинного и глубокого обучения (ML — Machine Learning, DP — Deep Learning) для решения задачи повышения эффективности выделения ресурсов ВВС. Актуальность поиска новых методик частично связана с резким ростом количества узлов в современных кластерах петафлопсного класса. При этом потери от незаполнения свободных узлов, доходящие до нескольких процентов годовой недозагрузки, могут быть эквивалентны простою целого кластера средней производительности. В некоторых работах, касающихся проблемы улучшения оценок временных фаз вычислительного задания — времени ожидания и времени выполнения — с помощью интеллектуальных методов, получены значительные результаты [4—6].

Принципиальной идеей данной работы, расширяющей ранее известные методы, является использование дополнительных метаданных вычислительных заданий, обучаемых на статистических данных планировщика и менеджера ресурсов Slurm [7], для прогнозирования временных показателей рабочей нагрузки. Предложенные модели направлены на решение нескольких практических задач. Первая задача связана с оповещением пользователей в онлайн-режиме достоверными прогнозными данными по вычислительным заданиям. Вторая задача предполагает построение и внедрение полуавтоматической пилотной очереди, управляемой планировщиком, который формирует собственные "безопасные" значения времени выполнения запускаемых заданий, отталкиваясь от истории запусков приложений.

В данной статье:

1) на выборочных данных большого объема проведен анализ статистики планировщика;
2) предложены группы моделей метаданных с расширенной идентификацией приложений;
3) приведено описание экспериментов по определению качества моделей и их применимости.

Используемая в исследовании статистика получена из журналов Slurm при проведении массовых расчетов на одном из кластеров ВНИИА в течение трех различных календарных периодов: осень 2017 г. (3 месяца), весна—лето 2018 г. (3 месяца), лето—осень 2018 г. (4 месяца).

1. Анализ фаз обработки вычислительных заданий

1.1. Ошибки оценок пользователей. В структуре рабочей нагрузки вычислительных кластеров принципиальную роль играют две временные характеристики, составляющие фазы ожидания и выполнения заданий пользователей. Еще одна временная характеристика, которая используется в данном исследовании, задается пользователем в качестве запрашиваемой максимальной квоты времени выполнения задания. Для описания моделей, предлагаемых в следующем разделе, обозначим эти переменные соответственно *Twait*, *Trun* и *Tlimit*. Еще одну переменную, характеризующую долю "угадывания" пользователем длительности задания — отношение $Trun/Tlimit$, обозначим как *Kwall-time*. Малые значения этого коэффициента свидетельствуют о завышенных значениях *Tlimit*, которые при бэкфиллинге, как известно, должны негативно влиять на длительность *Twait*. Действительно, для объединенного набора данных из статистики трехмесячных периодов расчетов 2017—2018 гг. корреляграмма (рис. 1)* имеет большой угол наклона регрессионной линии и значение коэффициента корреляции Пирсона $r = 0{,}53$, что соответствует средней степени зависимости, так как попадает в интервал (0,3; 0,7). В качестве аргумента на графике выступает "площадь" задания, т. е. произведение заказанных пользователем узлов (*nodes*) и прогно-

*На рис. 1 и 3 шкалы графиков прологарифмированы, чтобы показать наглядную качественную картину распределений со значительной долей малых значений в данных. Количественные метки на осях соответствуют не реальным, а некоторым условным метрикам значений визуализируемых переменных. На других графиках также могут использоваться условные обозначения.

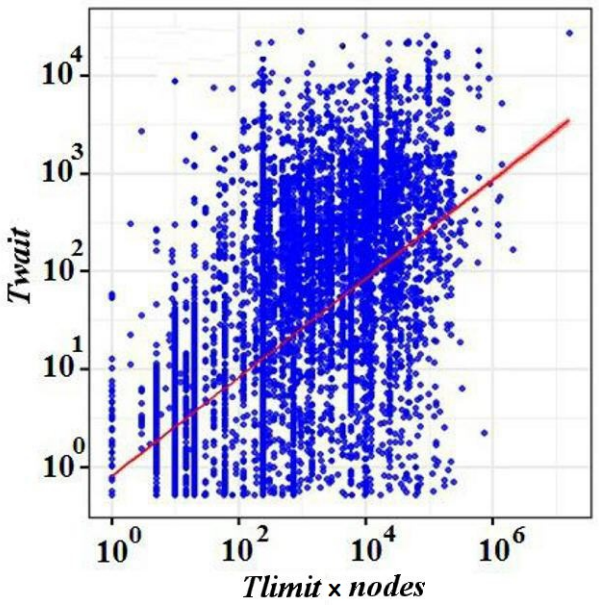

Рис. 1. Влияние запрашиваемых ресурсов на время ожидания, $r = 0{,}53$

за времени выполнения. Это подтверждает важность задачи оптимизации переменной *Tlimit*.

Особенно важна точность прогноза пользователя по *Tlimit* для коротких по времени запусков заданий, так как именно они помогают заполнять небольшие пустоты в вычислительном пространстве из распределенных и свободных ресурсов. Однако для этих заданий коэффициент *Kwall-time*, как правило, имеет очень низкие значения. Это демонстрирует рис. 2, где анализируется тот же набор данных, что и на рис. 1. Ордината (функция распределения CDF) показывает процент накопления упорядоченных значений коэффициента. Данные разбиты на два класса по порогу процессорного времени:

- длинные задания — размером больше 15 узло-минут;
- короткие задания — размером не больше 15 узло-минут.

Граничное значение в этой укрупненной метрике соответствует процессорному времени $15 \times 16 \times \times 60 = 14\,400$ процессорных секунд, так как данная статистика собрана для 16-ядерных узлов.

На рис. 2 наблюдается "ступенька" у кумулятивной функции распределения для немалого процента длинных заданий с *Kwall-time*, равным 1 (правые верхние части графиков). К сожалению, эта статистика свидетельствует не о проницательности пользователей, а, скорее, наоборот, — нежелании заботиться о точной оценке времени своих расчетов. Все эти задания сняты системой по событию таймаута. Длительный итерационный процесс моделирования по ряду методик позволяет пользователю получать и периодически сбрасывать на диск результаты приемлемой точности задолго до снятия задания системой. Таймаут является перестраховочным способом прекращения выполнения итераций счета.

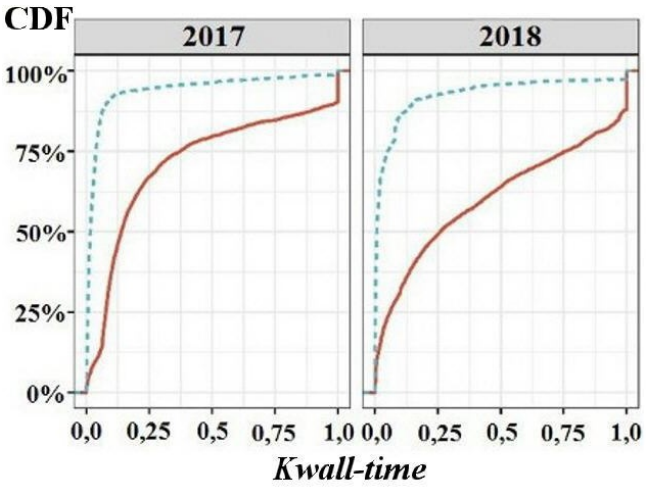

Рис. 2. Распределение времени выполнения заданий: — длинные задания; - - - — короткие задания

Тот факт, что к указанному классу относятся более длительные задания, чем те, которые сняты по нормальному коду завершения, иллюстрирует рис. 3. Диаграмма демонстрирует диапазон распределения значений *Tlimit* с высоким уровнем медианы для группы заданий с кодом таймаута.

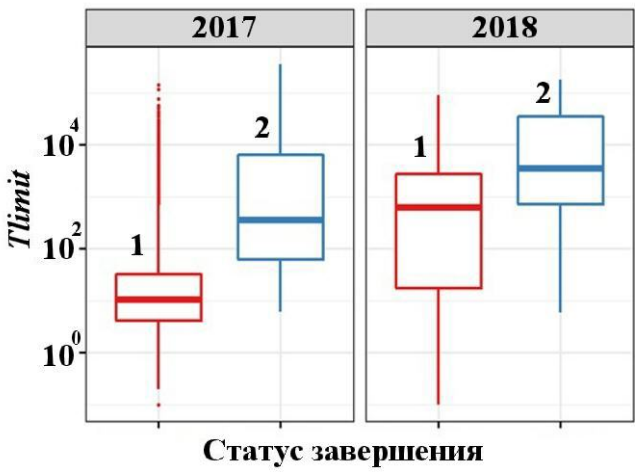

Рис. 3. Диаграмма диапазонов значений *Tlimit* по годам для двух классов заданий: 1 — с нормальным кодом завершения; 2 — завершенных по таймауту

1.2. Ошибки оценок планировщика. Задача корректировки значений *Tlimit* связана с прогнозированием времени нахождения в очереди *Twait*. Пользователь должен знать предполагаемый момент получения доступа к вычислительному ресурсу. Планировщик Slurm выдает эту информацию. Однако его оценка достаточно грубая, так как базируется на простой функции от значений *Tlimit* тех заданий, которые находятся в фазе ожидания в конкретном вычислительном разделе. Завышенные значения пользователей, как правило, приводят к пессимистическому прогнозу планировщика относительно времен старта заданий. Это подтверждают полученные при мониторинге трасс заданий кластера ВНИИА выборки, которые дополнены значениями прогноза времени запуска. Например, на такой выборке из 8 тыс. заданий (рис. 4) демонстрируется следующая статистика:

- не определено планировщиком 54 % заданий;
- находится в зоне переоценки 93 % от определенных;
- средняя ошибка на одно задание — 8 часов;
- суммарная ошибка — 42 условных суток.

Рис. 4 показывает значительно большую площадь под линией отклонения (ошибок) прогнозных значений *Tpred* от реальных *Tstart* в положительной зоне графика (зона переоценки).

Несмотря на попытки планировщика периодически корректировать прогнозное время при присутствии задания в очереди, это не всегда улучшает суммарное отклонение. По метрике средней абсолютной ошибки предстартовый прогноз для взятого набора данных только увеличил ошибку начального прогноза с 42 до 60 условных суток. На рис. 5 (см. также цветную вкладку) представлена трасса трех заданий, находившихся в состояниях ожидания и выполнения в течение 2 сут. Пунктирными цветными линиями обозначены фазы ожидания, сплошной черной — фазы выполнения. Цветные точки на вертикали показывают два прогнозных времени планировщика (*Tpred*) — первое и последнее, перед запуском, а черные точки соответствуют определенному системой мониторинга (1 раз в 5 мин) времени перехода задания в состояние выполнения (*Tactual*). Заметим, что у задания 35 513 начальный прогноз оказался лучше предстартового. У задания 35 521 предстартовый прогноз оказался точным, но в течение основного времени ожидания планировщик ошибался на 2 сут.

Описанный анализ журнальных файлов менеджера и планировщика заданий Slurm показал и подтвердил важность уточнения оценок двух временных характеристик вычислительных заданий: времени выполнения и времени ожидания.

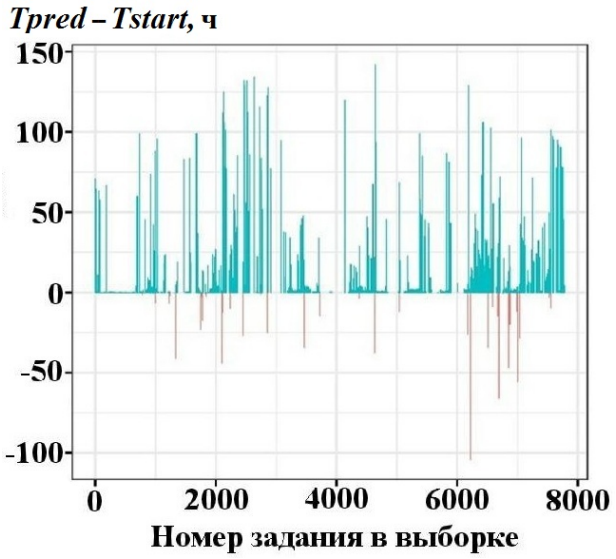

Рис. 4. Оценка Slurm времени запуска

2. Модели на основе расширения метаданных заданий

Рассмотрим две группы из четырех моделей метаданных M0, M1, M2, M3 для обучения па-

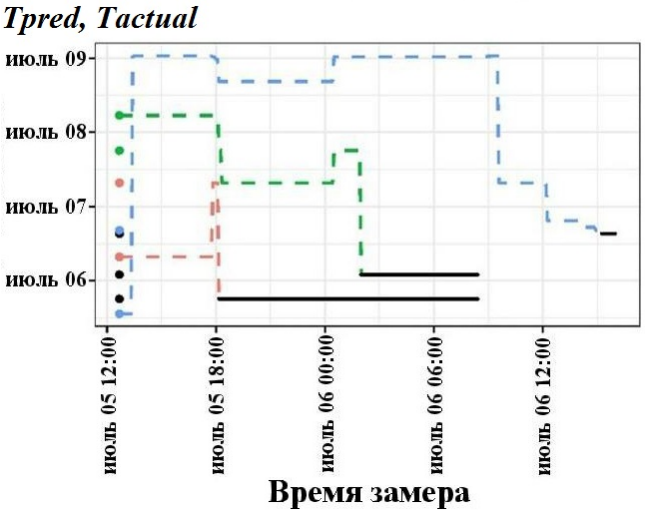

Рис. 5. Неэффективность корректировки Slurm прогноза времени запуска: •, - - - — задание 35 513; •, - - - — задание 35 520; •, - - - — задание 35 521

раметров-откликов при прогнозировании времен выполнения и ожидания заданий (табл. 1). Переменная отклика задает тип модели. Для непрерывных величин *Trun* и *Twait* прогнозная модель относится к типу регрессионной. Для двух других откликов, которые являются категориальными переменными, модель становится классификационной. Наборы предикторов для двух групп моделей совпадают, за исключением небольших отличий. Внутри группы каждая следующая модель содержит в себе предикторы предыдущей, расширяя их набор новыми метаданными.

Заметим, что только малая часть предикторов в явном виде содержится в типовых журналах Slurm. Модели 2 и 3 содержат метаданные, предложенные автором для увеличения количества предикторов (в табл. 1 они выделены серым фоном), которые формируются дополнительно, но также на основе статистики планировщика. Значения календарных параметров, которые, как ожидается, должны улучшать качество моделей, вычисляются по дате постановки задания в очередь раздела. Существенная часть предикторов идентифицирует приложение, по которому проводится расчет. Например, кортеж <*project, program, area, task, param*> является паспортом задания. С его помощью пользователь сохраняет в базе данных Slurm важную информацию о выполняемом приложении. Кортеж задается в атрибуте *comment* пакетного вызова. Если раньше эти дополнительные поля служили для анализа статистики запусков заданий [8], то в данной работе их предлагается использовать для

Таблица 1

Отклики и предикторы моделей из метаданных заданий

Параметр	Характеристика	Модель *Trun*				Модель *Twait*			
		M0R	M1R	M2R	M3R	M0W	M1W	M2W	M3W
	Отклики								
Trun	t выполнения	+	+	+	+				
Twait	t ожидания					+	+	+	+
class Trun	класс t выполнения	+	+	+	+				
class Twait	класс t ожидания					+	+	+	+
	Предикторы								
Tlimit	запрос t выполнения	+	+	+	+	+	+	+	+
Twait	t ожидания		+	+	+				
nodes	кол-во узлов	+	+	+	+	+	+	+	+
part	раздел		+	+	+		+	+	+
reserv	резервация		+	+	+		+	+	+
nuser	пользователь		+	+	+		+	+	+
nbat	файл запуска		+	+	+		+	+	+
nworkdir	рабочая директория			+	+			+	+
project	проект			+	+			+	+
program	код приложения			+	+			+	+
area	область моделирования			+	+			+	+
task	задача			+	+			+	+
param	расчетная модель			+	+			+	+
mon	месяц				+				+
hod	час дня				+				+
dow	день недели				+				+
pod	утро/день/вечер				+				+
nalloc	занятые узлы				+				+
nque	длина очереди				+				+
*Twait*1	t ожидания (Slurm)								+

улучшения качества моделей. Некоторые поля категориального типа, например имя пользователя, меняются на порядковые номера значений категориальной переменной, так как большое количество категорий ухудшает качество прогнозных моделей.

Объясним идею вложенности моделей.

Модель 0 (M0R, M0W) является тривиальной и базируется только на предикторе *Tlimit*.

Модель 1 (M1R, M1W) опирается на типовые предикторы, с помощью которых обычно проводятся исследования рабочей нагрузки параллельных заданий (формат swf — standart workload format) [9]. Эта статистика, собранная в некоторых зарубежных суперкомпьютерных центрах, доступна через Интернет [10]. К сожалению, swf-архивы содержат информацию о рабочей нагрузке суперкомпьютеров прошлого века, когда ее структура была принципиально другой. Поэтому валидацию методик и моделей на основании данных этих архивов нельзя считать обоснованной. Модель 1 можно рассматривать как базовый вариант.

Модель 2 (M2R, M2W) принципиально расширяет метаданные заданий, характеризуя специфику запускаемых приложений. Именно дополнительные параметры должны повышать точность модели. Применение расширенной идентификации заданий — важная концепция данной работы. Таким образом осуществляется переход от парадигмы обезличенного вычислительного задания, запускаемого конкретным пользователем, к понятию вычислительного *приложения* как группы расчетов, которое обладает устойчивыми ресурсными характеристиками.

Модель 3 (M3R, M3W) дополняет предикторы календарными признаками, а в случае моделирования времени ожидания — еще и состоянием раздела (количество распределенных узлов и длина очереди). Важной идеей является разбивка календарного времени на несколько категориальных значений: час и период суток, день недели и т. д., что должно помогать при классификационном разбиении.

Моделирование будем проводить методом машинного обучения Random Forest (RF — *случайный лес*) [11], который является одним из немногих универсальных алгоритмов и хорошо зарекомендовал себя на разнородных наборах с числовыми и категориальными данными. Алгоритм устойчив к выбросам значений. Кроме того, он имеет высокую скорость обучения и способность обрабатывать большие объемы данных. Все эти качества подходят для анализа суперкомпьютерной рабочей нагрузки. Подбор оптимального метода из "большой тройки" эффективных ML-методик — опорных векторов, решающих деревьев или градиентного бустинга, которые могут показывать близкие результаты по качеству прогнозов в зависимости от конкретного набора данных, целью не являлся.

3. Условия проведения экспериментов

3.1. Исходные данные и программное обеспечение. Для проведения обучения была отобрана генеральная выборка периодов осень—зима 2017 г. и весна—лето—осень 2018 г. статистики вычислительных заданий кластера средней производительности с высокой рабочей нагрузкой. При сборе данных принципиально не использовались специальные мониторинговые системы, чтобы показать самодостаточность информации журналов, собираемых Slurm. Основные данные были выгружены командой `sacct`. Дополнительные данные для моделей получены при периодическом мониторинге состояния узлов кластера и очереди с помощью команд `squeue` и `sinfo`, запускаемых через сервис `cron`. Из генеральной выборки сформированы две группы данных с количеством 5, 10 и 50 тыс. записей, чтобы тестировать поведение моделей при изменении размера выборки. Обучающая выборка согласно стандартной методике содержала 70 % всех наблюдений, контрольная тестовая — 30 %.

Программные скрипты реализованы в рамках использования инструментов статистической экосистемы на языке R [13]. Набор скриптов включен в состав разработанной автором в 2014—2015 гг. интерактивной системы анализа "Антик" [14]. В проведенном исследовании использовалась версия метода RF из пакета randomForest, доступная на CRAN-репозитории системы R. Так как RF относится к типу машинного обучения, базирующемуся на ансамблях решающих деревьев, в методике необходимо задавать параметр количества деревьев в ансамбле. Было выбрано значение 100, которое обычно используется в случае исследований, не связанных с масштабированием этого параметра, и которое в рассматриваемом случае обеспечило порог, после которого ошибки моделей уже не уменьшаются.

3.2. Метрики оценки качества. Для диагностики изменения качества моделей при варьировании наборов метаданных заданий будем применять наиболее рекомендуемые метрики для регрессионных и классифицирующих моделей:

1. Корень из среднеквадратической ошибки

$$\mathrm{RMSE} = \frac{1}{n}\sqrt{\sum_{i=1}^{n}\left(t_i^a - t_i^{pred}\right)^2},$$

где n — количество заданий в выборке; t_i^a– актуальное значение временного отклика; t_i^{pred} — предсказанное моделью значение.

2. Средняя абсолютная ошибка

$$\mathrm{MAE} = \frac{1}{n}\sum_{i=1}^{n}\left|\left|t_i^a - t_i^{pred}\right|\right|.$$

3. Точность (*accuracy*) — процент (доля) прогнозных значений отклика, классифицированных правильно. Другие, более информативные критерии эффективности классификаторов (полнота, специфичность и др.) [12] в работе не используются, так как применяются к классу бинарных классификаторов или к каждому классу в отдельности.
4. Важность предикторов. Метод RF позволяет оценивать роль предикторов в обеспечении качества классификатора, что информативно при отборе различных метаданных и сокращении размерности модели.

Регрессионные метрики целесообразно применять и для оценки классификаторов, когда номера классов не кодируются как значения категориальной переменной, так как они упорядочены по возрастанию числовых значений откликов, относящихся к смежным классам.

4. Исследование моделей

4.1. Модель Trun. Сначала проанализируем поведение моделей M0R, M1R, M2R, M3R на регрессионной схеме прогнозирования значения непрерывного времени выполнения заданий. Так как абсолютные значения оценочных метрик при секундном масштабе отклика *Trun* являются большими числами и малоинформативны, предлагается рассматривать отношение прогнозных значений пользователя *Tlimit* к прогнозным значениям модели. Это отношение k показывает, во сколько раз модель точнее человеческого предсказания. В табл. 2 наблюдаются похожие приросты улучшения качества прогноза для разных размеров выборок. Метрика RMSE считается наиболее важным критерием для оценки прогнозирования значений непрерывных переменных. Она дает более чем трехкратное улучшение на последней модели. Из табл. 2 также видна монотонность изменения значений. Дополнительно функции распределения на рис. 6 иллюстрируют существенную разницу близости с реальным временем модельной оценки по сравнению с оценкой *Tlimit.* В визуализируемом наборе отфильтрованы малые времена, на которых все графики визуально близки, а также несколько больших "выбросов".

Перейдем к классификационной схеме прогнозирования. Ее целесообразно использовать в случае автоматической безопасной замены прогноза пользователя на порядковый номер класса. Прогнозировать значение непрерывного времени выполнения не имеет большого практического смысла, важнее определить границы подходящего класса и выбрать его верхнее значение для замены прогноза, предлагаемого пользователем.

Шесть классов для отклика *Trun* в рамках конкретной выборки сформируем следующим образом:

Таблица 2

Улучшение качества регрессионных моделей при расширении метаданных (2018 г.)

Размер выборки	Тип коэффициента	Модель метаданных				
		Tlimit	M0R	M1R	M2R	M3R
5 тыс.	k(RMSE)	1	2,48	2,73	3,09	3,27
	k(MAE)	1	2,87	3,05	3,93	4,11
10 тыс.	k(RMSE)	1	2,60	2,86	3,14	3,23
	k(MAE)	1	3,00	3,16	3,97	4,22
50 тыс.	k(RMSE)	1	2,97	3,49	3,61	3,75
	k(MAE)	1	3,42	5,09	5,55	5,78

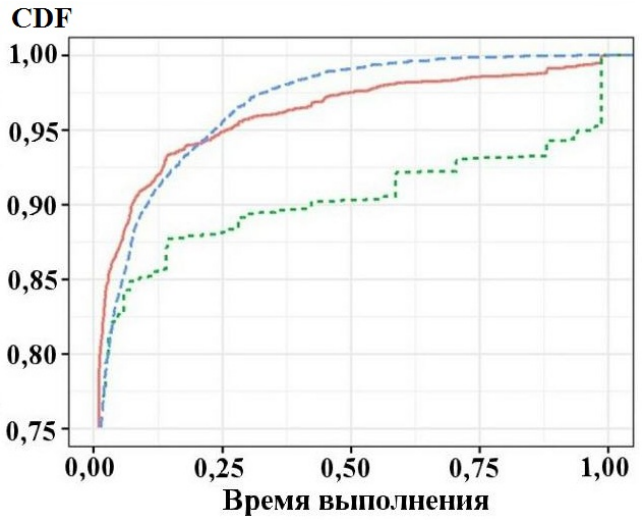

Рис. 6. Сравнение близости распределений прогнозов M3R (- - -) и *Tlimit* (···) с реальным временем (——)

1) разобъем шкалу времени на два сегмента: задания, выполнявшиеся менее и более часа;
2) внутри каждого сегмента проведем группировку значений *Trun* в три кластера по методу k-means [13].

Подобное динамическое разбиение границ кластеров для определения классов отклика по сравнению с фиксированно заданными границами позволяет учитывать частотность и структуру распределения параметра $Trun$. Наличие тестовых методических заданий с коротким временем выполнения, а также производственных методик типа Bag-of-Tasks (BoT), когда запускается большое количество коротких заданий (в частности, в области молекулярно-динамического моделирования), обосновывает специальное выделение первого сегмента. В табл. 3 приведены данные о качестве классификатора, построенного по прежним метрикам с добавлением метрики точности попадания в требуемый класс. При вычислении формул ошибок используются абсолютные (числовые) порядковые значения классов. В связи с тем, что категории классов упорядочены по возрастанию значений переменной, регрессионные метрики можно использовать как классификационные. Чисто классификационная метрика *accuracy* показывает долю правильно определенных классов. В модели 3 для различных выборок ее значение составляет чуть менее 90 %. Здесь также наблюдается монотонность результатов обучения по всем критериям.

Приведенные в таблице результаты соответствуют случайным, но типовым "усредненным" по структуре выборкам. Большое количество проведенных экспериментов показало зависимость качества оценок от процента BoT-приложений в структуре выборок. При исключении этой группы запусков из анализа прогноз ухудшается. И наоборот, он улучшается, когда задания данной группы составляют большинство. Итоговый разброс значений метрики *accuracy* определен в интервале между 70 и 96 %.

4.2. Модель Twait. В подразд. 1.2 был приведен пример ошибок планировщика Slurm в оценке времени ожидания заданий в очереди. Для получения этой информации пришлось запускать процедуру мониторинга состояния очереди. В набор таких заданий попадали те, которые ожидали в очередях более 3 мин. Для сформированного набора из 8 тыс. заданий получены оценки прогнозирования значений *Twait* плани-

Таблица 3

Оценка качества прогноза для *Trun*-модели с классами

Размер выборки	Метрика	Модель метаданных			
		M0R	M1R	M2R	M3R
5 тыс.	RMSE	0,871	0,699	0,698	0,677
	MAE	0,312	0,212	0,207	0,200
	accuracy	0,836	0,879	0,886	0,887
10 тыс.	RMSE	0,726	0,524	0,523	0,511
	MAE	0,361	0,175	0,172	0,157
	accuracy	0,716	0,866	0,870	0,884
50 тыс.	RMSE	0,678	0,589	0,564	0,551
	MAE	0,288	0,209	0,194	0,187
	accuracy	0,780	0,845	0,854	0,859

ровщиком и с использованием моделей по регрессионной схеме (табл. 4). В этом примере модели метода RF в несколько раз превосходят по качеству оценку, сделанную планировщиком. Дополнительно функции распределения на рис. 7 иллюстрируют существенную разницу близости с реальным временем модельной оценки по сравнению с оценкой Slurm.

Как и для *Trun*-моделей, в табл. 5 приведен пример улучшения качества прогноза для классификационных моделей на выборке размером в 10 тыс. записей периода весна—лето 2018 г.

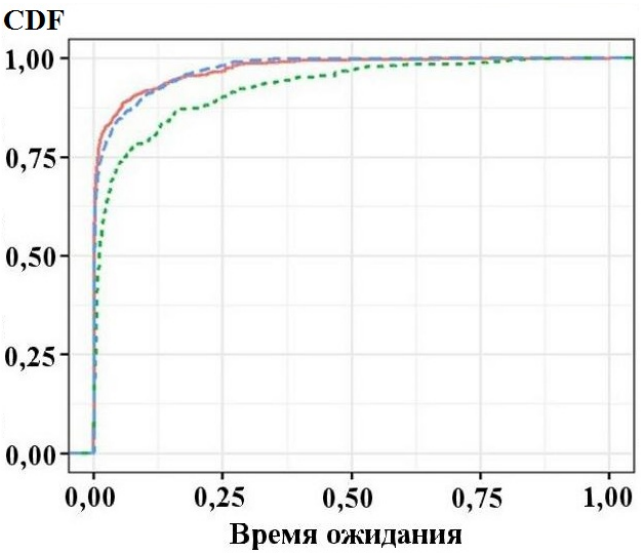

Рис. 7. Сравнение близости распределений прогнозов M3W (- - -) и Slurm (···) с реальным временем (——)

Монотонность у *Twait*-моделей при расширении метаданных сохраняется, при этом она еще более выражена по сравнению с *Trun*-моделями (рис. 8). Ошибки уменьшаются быстрее, точность растет соответственно.

4.3. Важность влияния предикторов. Метод RF позволяет учитывать важность вклада переменных-предикторов в эффективность модели. Для оценки используются две специальные метрики. Первая определяет результаты перестановки значений в предикторе, т. е. уменьшения точности модели. Вторая метрика определяет результат уменьшения показателя разнородности Джини (энтропии). На рис. 9 показан пример ранжирования предикторов M3W по этим двум критериям для самой большой многомесячной выборки в 50 тыс. заданий.

Как и предполагалось, в прогнозе времени ожидания, кроме запрашиваемых ресурсов (*Tlimit, nodes*), корреляция с которыми показана на рис. 1, важную роль по первому критерию играют календарные параметры (*dow, hod, mon* и др.). Также среди наиболее важных параметров фигурируют данные паспорта заданий (*project, area, program* и др.). В верхней части списка также присутствуют введенные в модель 3 данные о состоянии загруженности кластера (*nalloc, nque*). Этот пример подтверждает правильность идеи включения расширенных метаданных в состав моделей.

Еще одно предложение заключается в применении прогноза Slurm в дополнение к ал-

Таблица 4

Улучшение качества прогноза Slurm с использованим *Twait*-моделей

Размер выборки	Тип коэффициента	Модель метаданных				
		Slurm	M0W	M1W	M2W	M3W
8 тыс.	k(RMSE)	1	1,83	2,39	2,42	2,86
	k(MAE)	1	2,48	2,87	2,99	4,02

Таблица 5

Оценки качества прогноза для *Twait*-моделей с классами

Размер выборки	Метрика	Модель метаданных			
		M0W	M1W	M2W	M3W
10 тыс.	RMSE	0,877	0,767	0,692	0,494
	MAE	0,436	0,387	0,314	0,146
	accuracy	0,674	0,689	0,746	0,889

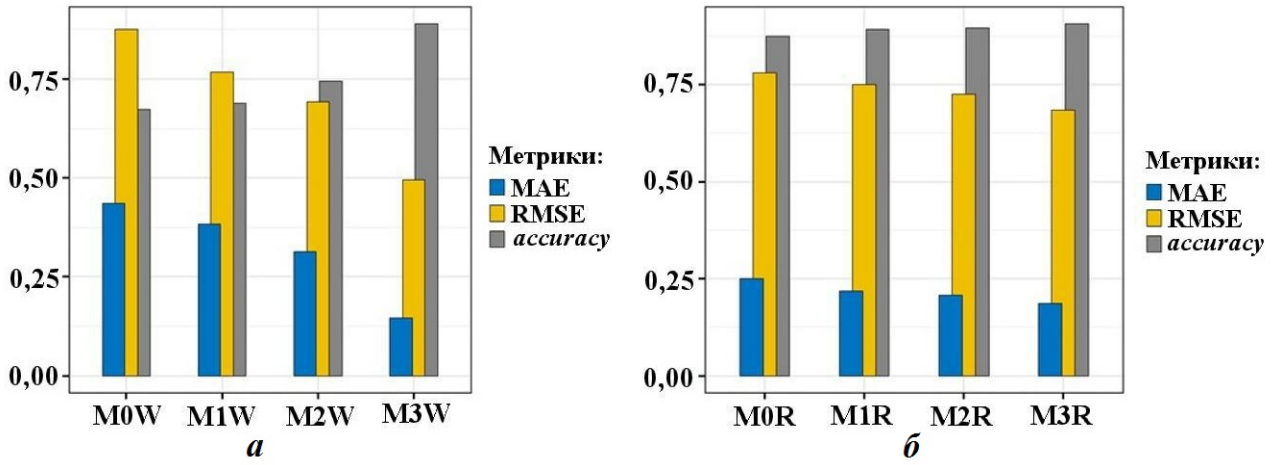

Рис. 8. Сравнение монотонности обучения *Twait*- (*а*) и *Trun*-моделей (*б*) (выборка в 10 тыс. записей, 2018 г.)

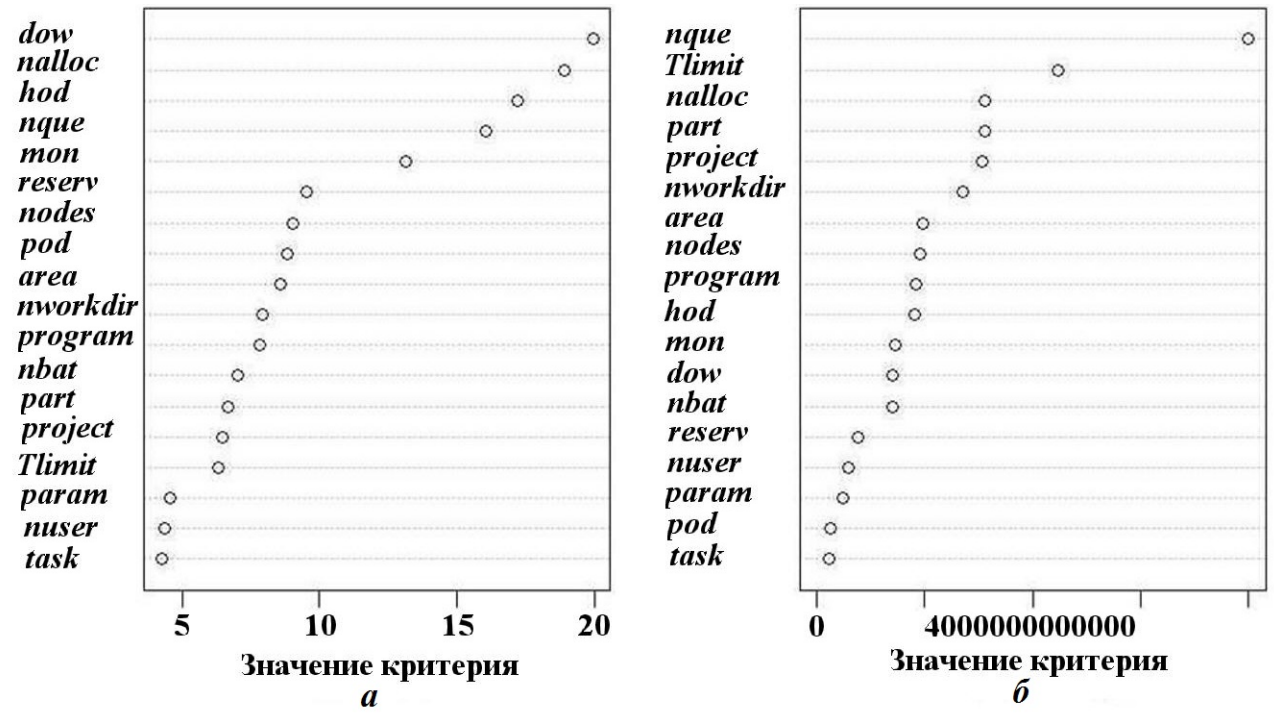

Рис. 9. Важность предикторов для модели M3W: *а* — критерий точности; *б* — критерий Джини

горитму RF. Несмотря на неточность оценки планировщика, в роли предиктора модели (см. последний параметр табл. 1) он получает первое место в рейтинге важности по обоим критериям, существенно повышая качество прогнозирования. Для набора данных в 8 тыс. заданий это улучшение составило 6 % для RMSE и 12 % для MAE.

4.4. Влияние параметров при обучении. Алгоритм RF не обладает множеством параметров для его настройки на конкретный набор данных в процессе обучения модели. Принципиальным параметром является количество деревьев в ансамбле — параметр *ntree*. Проведенные эксперименты показали, что во всех случаях ошибка перестает уменьшаться после *ntree* >100. При этом для моделей *Twait* резкое уменьшение ошибки происходит уже в интервале [20,40] (рис. 10), а для моделей *Trun* порогом является *ntree* = 50. Следовательно, при встраивании алгоритма в систему планирования заданий скрипт построения модели будет выполняться быстрее, если для параметра будет выбрано уменьшенное значение. Модель M3W демонстрирует более чем двукратное улучшение.

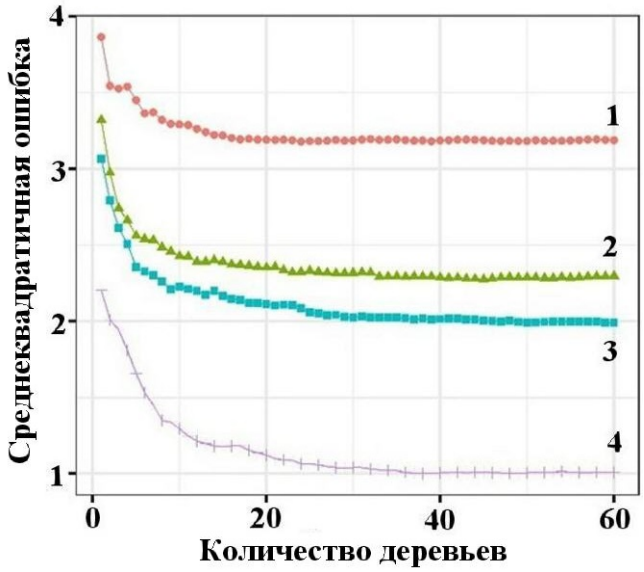

Рис. 10. Зависимость ошибки от количества деревьев в ансамбле для *Twait*-моделей: 1 — M0W; 2 — M1W; 3 — M2W; 4 — M3W

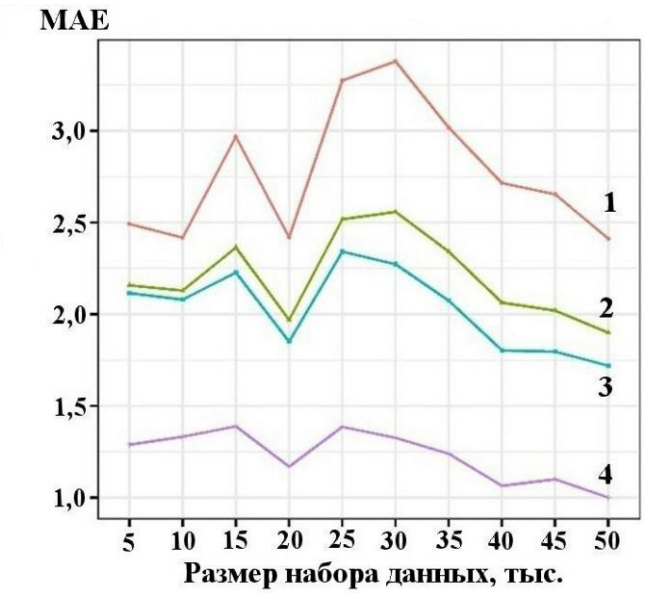

Рис. 11. Зависимость ошибки от размера выборки для *Twait*-моделей: 1 — M0W; 2 — M1W; 3 — M2W; 4 — M3W

Другим параметром, влияющим на качество оценки прогнозирования, является размер выборки. Применим следующую схему масштабирования. Интервал периода рабочей нагрузки будем увеличивать при движении назад от текущего последнего времени замера в генеральной выборке 2018 г. с шагом в 5 тыс. заданий. Рис. 11 иллюстрирует увеличение прогнозной ошибки для всех моделей при размере выборки в интервале 20—30 тыс. заданий. Однако с дальнейшим увеличением размера возникает монотонное улучшение качества регрессионного прогнозирования как для простых, так и многопредикторных наборов метаданных. Таким образом, при реальном выполнении прогнозного скрипта в системе оповещения пользователей для него необходимо выбирать оптимальные размеры набора данных, полученные при предварительном исследовании влияния масштабирования.

5. Две задачи внедрения результатов исследования

Полученные выше результаты позволяют перейти к практической фазе включения моделей в окружение планировщика. На первом этапе предполагается решение двух задач, сформулированных во Введении.

Первый проект предполагает разработку команды-утилиты для пользователей, стремящихся получать более достоверную информацию об обработке своих заданий, находящихся в очередях вычислительного кластера. Интерфейс команды прост: на входе задается *JobID* — идентификатор задания в стадии ожидания, на выходе пользователь получает две временные оценки — предполагаемое календарное время запуска задания и прогнозное значение длительности его выполнения после запуска.

Схема реализации взаимодействия Slurm и блока прогнозирования следующая:

1) при постановке любого задания в очередь планировщик совместно со стандартным парсингом опций пакетных команд `sbatch` выполняет расширенные функции по формированию 20 метаданных (см. табл. 1) запускаемого задания и сохранению этой записи во внутренней базе данных. Часть кода подобного расширения уже включена в текущую версию планировщика для обеспечения синтаксического анализа полей паспорта задания;
2) R-скрипт построения моделей M3W и M3R запускается через механизм `cron` по выбранному расписанию: раз в час или реже, чтобы актуализировать модели с учетом последних завершенных заданий. Глубина данных статистики является параметром скрипта;

3) по запросу пользователя из базы планировщика выбирается вектор метаданных запрашиваемого задания, загружается ранее построенная текущая модель и осуществляется прогнозирование двух единичных откликов, которые визуализируются в консольном окне пользователя.

При реализации второго проекта схема несколько меняется: шаг 3 схемы запускается сразу после шага 1 самим планировщиком для прогнозирования по модели M3R и замены пользовательского значения *Tlimit* на значение вычисленного отклика. Таким образом, второй расширенный проект предусматривает для нового высокопроизводительного кластера ВНИИА внедрение экспериментальной пилотной очереди, управляемой планировщиком, который формирует собственные оптимальные "безопасные" значения времени выполнения запускаемых заданий, отталкиваясь от истории запуска приложений на основе прогнозных моделей. Наличие нового кластера для проекта вызвано также технической проблемой отладки алгоритмов интеграции. К сожалению, безболезненная интеграция Slurm и блока моделирования требует наличия испытательных стендов и практически невозможна на работающем высоко загруженном кластере, так как затрагивает самый важный и чувствительный элемент управления кластером — систему управления вычислительными заданиями.

Выводы

Проведенное исследование продемонстрировало способность методик машинного обучения к эффективному прогнозированию важнейших показателей рабочей нагрузки, связанных с временными характеристиками обработки вычислительных заданий. Получены следующие результаты:

1. На значительной (по времени и объему данных) статистике показана существенная неточность прогнозирования в оценках пользователей и планировщика заданий.
2. Предложены новые группы метаданных вычислительных заданий для построения прогнозов их временных характеристик на базе метода машинного обучения RF.
3. Показана эффективность предложенных моделей: качество оценок для любых выборок данных по сравнению с оценками пользователей и планировщика Slurm лучше в несколько раз.
4. В процессе обучения выявлены зависимости качества оценок от расширения типов предикторов, параметров самой методики и структуры рабочей нагрузки (в частности, установлено существенное улучшение прогноза при наличии большого процента заданий типа BoT).
5. Коды разработанных скриптов включены в состав системы статистического анализа "Антик" [14] (разработка ВНИИА).

Осуществляемое в настоящий момент внедрение описанных интеллектуальных методов для управления вычислительными заданиями кластеров ВНИИА позволит собрать материал для анализа повышения эффективности загрузки их вычислительных ресурсов. Предполагается провести сравнение результативности работы планировщика в штатном и прогнозном режимах на реальных данных массовых расчетов и опубликовать результаты этого сравнения.

Список литературы

1. *Muhalem A, Feitelson.* Utilization, predictability, workloads, and user runtime estimates in scheduling the IBM SP2 with backfilling // IEEE Trans. Parallel Distrib. Syst. 2001. Vol. 12(6). P. 529—543.
2. *Smith W., Foster I., Taylor V.* Predicting application run times using historical information // Job Scheduling Strategies for Parallel Processing. Lect. Notes Comp. Sci. 1998. P. 122—142.
3. *Skovira J., Chan W., Zhou H., Lifka D.* The EASY — LoadLeveler API project // Ibid. 1996. Vol. 1162. P. 41—47.
4. *Park J.-W., Kim E.* Runtime prediction of parallel applications with workload-aware clustering // J. Supercomputing. 2017. Vol. 73. P. 4635—4651.
5. *Soysal M., Bergho M., Streit A.* Analysis of job metadata for enhanced wall time prediction // Job Scheduling Strategies for Parallel Processing. Lect. Notes Comp. Sci. 2018. Vol. 11332. P. 1—14.

6. *Klusacek D., Chlumsky V.* Evaluating the impact of soft walltimes on job scheduling performance // Job Scheduling Strategies for Parallel Processing. Lect. Notes Comp. Sci. 2018. Vol. 11332. P. 15—38.

7. *Yoo A. B., Jette M. A., Grondona M.* SLURM: Simple Linux utility for resource management // Job Scheduling Strategies for Parallel Processing. Lect. Notes Comp. Sci. 2003. Vol. 2862. P. 44—60.

8. *Петунин С. А.* Методика начального анализа рабочей нагрузки вычислительных кластеров // Тр. 4-й Межд. науч. конф. "Информационные технологии и системы (ИТиС)". Челябинск: Челябинский гос. университет, 2015. С. 129—130.
Petunin S. A. Metodika nachalnogo analiza rabochey nagruzki vychislitelnykh klasterov // Tr. 4-oy Mezhd. Nauch. Konf. "Informatsionnye tekhnologii i sistemy (ITiS)". Chelyabinsk: Chelyabinskiy gos. universitet, 2015. S. 129—130.

9. *Feitelson D. G.* Parallel Workloads Archive. http://www.cs.huji.ac.il/labs/parallel/workload.

10. *Chapin S. J., Cirne W., Feitelson D. G., Jones J. P., Leutenegger S. T., Schwiegelshohn U., Smith W., Talby D.* Benchmarks and standards for the evaluation of parallel job schedulers // Job Scheduling Strategies for Parallel Processing. Lect. Notes Comp. Sci. 1999. Vol. 1659. P. 67—90.

11. *Груздев А. В.* Прогнозное моделирование в IBM SPSS Statistics, R и Python: метод деревьев решений и случайный лес. М.: ДМК Пресс, 2018.
Gruzdev A. V. Prognoznoe modelirovanie v IBM SPSS Statistics, R i Python: metod derevev resheniy i sluchayny les. M.: DMK Press, 2018.

12. *Джеймс Г., Уиттон Д.* Введение в статистическое обучение с примерами на языке R. М: ДМК Пресс, 2016.
Dzheyms G., Uitton D. Vvedenie v statisticheskoe obuchenie s primerami na yazyke R. M.: DMK Press, 2016.

13. Официальный сайт R-проекта.
http://www.r-project.org.
Ofitsialny sayt R-proekta.
http://www.r-project.org.

14. *Petunin S. A., Vorontsov A. G.* The use of the historical job data scheduler to analyze high performance computing applications // Proc. 19th Workshop on Computer Science and Information Technologies CSIT'2017. Germany, Baden-Baden, 2017. P. 164—167.

Статья поступила в редакцию 17.12.18.

EFFICIENT JOB TIMING DATA PREDICTION BASED ON EXTENDED METADATA / S. A. Petunin (FSUE "N. L. Dukhov VNIIA", Moscow).

Application of the Random Forest statistical method from the machine learning group for predicting the job timing data (queuing time and runtime) was studied. All the experiments were done with a large body of data logged by the Slurm workload manager and job scheduler during different calendar periods. The data processing programs were implemented as scripts within the R statistical system. The key idea of the work was to incorporate new unique job metadata, which have never been employed before, into the structure of the predictive models. The results of the studies demonstrated that the proposed predictive models were twice as efficient as the estimates of users and the job scheduler. This work is the first step in the implementation of automatic job management services at the level of the metadata-based job scheduler to provide automatic processing of resource requests and more efficient notification of users.

Keywords: prediction, job metadata, runtime, latency, Random Forest method.