УДК 004.457+004.94+004.382.2
DOI: 10.15827/0236-235X.140.631-643

Дата подачи статьи: 05.07.22
2022. Т. 35. № 4. С. 631–643

Имитационная модель системы пакетирования суперкомпьютерных заданий на базе симулятора Alea

А.В. Баранов [1], *к.т.н., доцент, ведущий научный сотрудник, antbar@mail.ru, abaranov@jscc.ru*
Д.С. Ляховец [1], *младший научный сотрудник, anetto@inbox.ru*

[1] *Межведомственный суперкомпьютерный центр РАН – филиал ФГУ ФНЦ НИИСИ РАН, г. Москва, 119334, Россия*

Современные системы управления заданиями суперкомпьютера представляют собой сложные программные комплексы с множеством различных алгоритмов планирования и их параметров, влияние которых на показатели эффективности системы невозможно точно предсказать или рассчитать. По этой причине для определения оптимальных параметров системы управления заданиями применяют имитационное моделирование.

Настоящая статья посвящена задаче построения имитационной модели системы управления суперкомпьютерными заданиями на базе известного симулятора Alea. В качестве объекта исследования выступил разработанный алгоритм планирования, на основе которого построена программная система пакетирования суперкомпьютерных заданий. Суть алгоритма состоит в объединении заданий с длительным временем инициализации в пакеты по типам заданий. Для каждого пакета инициализация производится однократно, после чего одно за другим выполняются задания пакета. За счет применения системы пакетирования удается сократить долю накладных расходов на инициализацию и повысить эффективность планирования заданий.

Алгоритм пакетирования реализован в составе симулятора Alea, исследование характеристик этого алгоритма для различных входных потоков заданий произведено путем сравнительного имитационного моделирования. В сравнении участвовали встроенные в Alea алгоритмы планирования FCFS и Backfill. Для моделирования сгенерировано несколько входных потоков заданий с различной интенсивностью. По результатам моделирования для этих потоков удалось определить минимальные пороги доли инициализации задания, начиная с которых система пакетирования заметным образом улучшает показатели эффективности планирования по сравнению с алгоритмами FCFS и Backfill.

Результаты исследования показали, что построенная имитационная модель может быть применена в качестве инструментального программного средства для сравнительного анализа различных алгоритмов планирования суперкомпьютерных заданий.

Ключевые слова: *высокопроизводительные вычисления, системы управления заданиями, имитационное моделирование, пакетирование заданий, Alea.*

Для производства расчетов пользователь суперкомпьютерного центра, как правило, оформляет вычислительное задание. Задание включает в себя параллельную программу, требования к объему и типу необходимых для выполнения программы ресурсов, а также входные данные. Задания разных пользователей формируют входной поток, поступающий в специальную программную систему – *систему управления заданиями* (СУЗ). Примерами СУЗ являются SLURM, Moab, PBS, LSF. В Межведомственном суперкомпьютерном центре РАН в качестве СУЗ более 20 лет применяется *система управления прохождением параллельных заданий* (СУППЗ) [1]. СУЗ ведет очередь заданий, выделяет и конфигурирует вычислительные ресурсы для заданий, контролирует их выполнение. Современные СУЗ – это достаточно сложные системы, обладающие множеством настраиваемых параметров.

Ведение очереди заданий в СУЗ базируется на задаче построения расписания. От качества построенного расписания зависят показатели эффективности СУЗ, такие как загрузка вычислителя, среднее время ожидания задания в очереди и другие. На эти показатели существенное влияние оказывают алгоритм планирования, настраиваемые параметры СУЗ и планировщика, а также характеристики входного потока заданий. СУЗ может быть дополнена влияющими на планирование подсистемами, например, подсистемой формирования пакетов заданий [2]. Заранее невозможно предсказать или рассчитать характер влияния конфигурации

СУЗ на показатели качества. Для исследования влияния параметров входного потока и конфигурации СУЗ на показатели эффективности наиболее перспективным способом является имитационное моделирование.

Существующие методы, способы и средства моделирования СУЗ подробно рассмотрены в работе [3]. Для проведения имитационного моделирования могут быть применены различные симуляторы, такие как MONARC [4], Alea [5], OptorSim [6], WorkflowSim [7], Batsim [8], SLURM simulator [9].

Наиболее точным способом моделирования является применение модели СУЗ с виртуальным вычислителем – специальной программной системой, позволяющей обрабатывать входной поток заданий в виртуальном суперкомпьютере без реальных вычислений [10]. Главным недостатком такой модели является длительное время проведения эксперимента, сопоставимое со временем реальных вычислений за исследуемый период. Моделирование на базе симулятора Alea [5] несколько уступает в точности, но позволяет существенно ускорить эксперимент, в течение нескольких минут обеспечивая моделирование недельного периода работы суперкомпьютера. В Alea есть возможность реализовать и исследовать собственный алгоритм планирования.

В настоящее время Alea находит широкое применение. Например, в [11] на Alea проведено моделирование обработки 96 тысяч заданий для исследования алгоритма планирования. В работе [12] предлагается использовать генетический алгоритм для оптимизации расписания и в дальнейшем планируется применение Alea для экспериментального исследования характеристик алгоритма. В [13] предлагается механизм динамического планирования на базе объединения алгоритмов Backfill и Metaheuristic Local Search Optimization. В [14] рассматривается алгоритм least average load variance-based, а моделирование выполняется в Alea. В работе [15] описано применение Alea для изучения влияния назначения вычислительных узлов заданию на пропускную способность системы.

Для построения имитационной модели в Alea необходимо реализовать алгоритм планирования, сформировать входной поток заданий, провести эксперимент в Alea и проанализировать полученные результаты. Имитационная модель была реализована в виде программной системы для сравнительного анализа метода формирования пакетов заданий [2] со встроенными в Alea алгоритмами планирования: FCFS (первый пришел – первый обслужен) и Backfill (алгоритм обратного заполнения). Следует отметить, что FCFS является простейшим алгоритмом планирования, однако в большинстве современных СУЗ наиболее широко применяется алгоритм Backfill.

Метод формирования пакетов заданий по типам

Жизненный цикл задания включает в себя подготовку (инициализацию) задания, непосредственно его обработку, завершение задания (в наиболее простом случае – только сохранение результатов) и освобождение вычислительных ресурсов. К инициализации задания отнесем все подготовительные действия, которые необходимо выполнить до начала обработки задания, включая подготовку вычислительных ресурсов.

Все этапы, кроме непосредственно обработки задания, являются накладными расходами, сокращение которых приводит к увеличению эффективности вычислений. Высокая доля времени инициализации является следствием либо длительного времени инициализации, либо малого времени обработки задания. Для заданий с малой долей инициализации накладные расходы, как правило, не оказывают негативного влияния на показатели эффективности СУЗ.

В статье [2] рассматриваются различные классы задач с большой долей инициализации. Такие задачи встречаются в областях фото- и видеообработки, анализа движения, расчета освещенности после добавления или удаления объектов, секвенирования генома, фармакологии, ядерной физики и биоинформатики. Заметные накладные расходы могут быть характерны для заданий, которым необходимы подготовка и разворачивание виртуальной платформы, в том числе с использованием контейнерной виртуализации.

Будем называть задания с одинаковой процедурой инициализации однотипными заданиями. Однотипные задания одного пользователя могут быть объединены в один пакет (метазадание). Для метазадания можно произвести однократную инициализацию, после чего одно за другим выполнить задания пакета, снизив тем самым долю накладных расходов. Будем рассматривать линейно масштабируемые задания с постоянным временем инициализации заданий одного типа [16]. Для таких заданий время

инициализации зависит только от типа задания и не зависит от количества выделяемых вычислительных ресурсов.

Для объединения заданий в группы известны различные способы [17–23]. В отличие от предлагаемого в настоящей статье метода *формирования пакетов заданий по типам* (ФПЗТ) в указанных работах при объединении заданий в группы не учитываются их типы. Предложенный в работе [2] метод ФПЗТ может быть применен для произвольной СУЗ. Практически этот метод был реализован в виде системы пакетирования заданий для СУППЗ. Для исследования характеристик алгоритма, реализующего метод ФПЗТ, в работе [10] применялась модель СУЗ с виртуальным вычислителем, которая из-за длительного процесса моделирования не позволила за приемлемое время провести необходимое число экспериментов. Существенно ускорить моделирование удалось за счет применения симулятора Alea и построения на его базе имитационной модели системы пакетирования заданий.

Имитационная модель СУЗ на базе Alea

Построенная на базе Alea имитационная модель СУЗ представлена на рисунке 1. Изменяемыми параметрами модели являются характеристики входного потока заданий, алгоритм планирования, настройки типов заданий и виртуального вычислителя моделируемого суперкомпьютера. Генератор модельного потока заданий формирует входной поток заданий, который подается на вход Alea. Результатом работы Alea является выходной поток, в котором для каждого задания в процессе моделирования определяется время старта и завершения. Анализатор выходного потока заданий обрабатывает выходной поток и вычисляет показатели эффективности планирования.

В качестве алгоритма планирования в предлагаемой модели может быть использован любой из трех вариантов: встроенные в Alea FCFS и Backfill, а также алгоритм Packet, реализующий метод ФПЗТ. Алгоритм Packet ведет отдельную очередь для заданий каждого типа, формирует пакеты заданий одного типа на основе динамически определяемых весов каждой очереди с учетом размера очереди и длительности нахождения в очереди самого старого задания и определяет требуемое число вычислительных узлов для каждого пакета.

Входной поток заданий, передаваемый в симулятор, представляет собой текстовый файл в формате SWF (Standard Workload Format), в котором каждая строка файла соответствует заданию с 25 параметрами [24, 25]. Были использованы следующие основные параметры задания: идентификатор, время поступления в систему, время выполнения, заказанное число процессоров (вычислительных узлов), номер очереди (тип задания).

Для хранения типа задания было решено использовать существующее поле «номер очереди» (queue number, столбец 15 в SWF), поскольку метод ФПЗТ фактически предусматривает ведение отдельной очереди для каждого типа задания.

Из имеющихся настроек виртуального вычислителя в Alea было задействовано только число вычислительных узлов, сконфигурированных как однопроцессорные.

Схема реализации в Alea собственного алгоритма планирования

Входное задание из SWF-файла преобразуется в экземпляр класса GridletInfo (рис. 2), содержащий различную информацию (например,

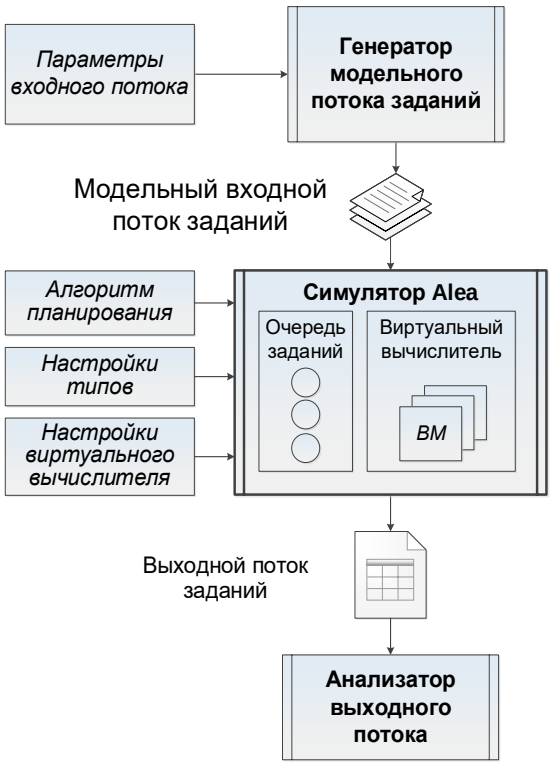

Рис. 1. Имитационная модель СУЗ на основе Alea

Fig. 1. Job management system simulation model based on the Alea simulator

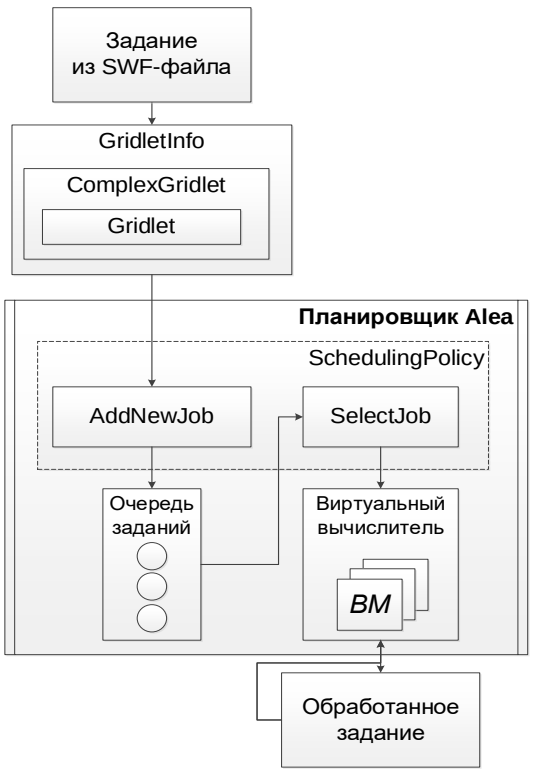

Рис. 2. Обработка задания в симуляторе Alea

Fig. 2. Job processing in the Alea simulator

идентификатор задания, его статус, идентификатор ресурса). В GridletInfo содержится информация о задании в экземпляре класса ComplexGridlet (например, время поступления в очередь и заказанное число процессоров). Класс ComplexGridlet, в свою очередь, расширяет базовый класс Gridlet из проекта gridsim, на котором основан Alea [26]. При необходимости добавления новой информации в сущность задания разработчики Alea рекомендуют расширять класс GridletInfo.

Алгоритмы планирования в симуляторе Alea представляются классами, реализующими интерфейс SchedulingPolicy с двумя методами – AddNewJob и SelectJob. Метод AddNewJob с параметром GridletInfo вызывается в момент поступления нового задания в очередь. Метод SelectJob без параметров используется для выбора задания из очереди на выполнение.

В процессе реализации метода SelectJob для алгоритма ФПЗТ была обнаружена проблема отслеживания числа заданий в Alea. Планировщик Alea отслеживает число поступивших заданий и завершает работу только после обработки всех заданий из входного потока. Особенностью алгоритма пакетирования является преобразование большего числа заданий в меньшее число пакетов, в результате чего планирование заданий никогда не завершалось. Чтобы избежать модификации ядра Alea, было принято решение трактовать пакет как набор из всех заданий этого пакета, где у первого задания задаются характеристики самого пакета (требуемое число процессоров и время обработки), а остальные задания пакета преобразуются в псевдозадания с ресурсными требованиями в один процессор и 1 секунду обработки. В результате число заданий в выходном потоке совпадает с числом заданий во входном потоке, при этом псевдозадания не рассматриваются при расчете характеристик выходного потока заданий.

Формирование входного потока заданий

Для анализа метода ФПЗТ в разных условиях был использован генератор Lublin& Feitelson [27], который создает поток заданий, статистически схожий с реальным потоком заданий суперкомпьютера. Генерируемыми параметрами задания являются время поступления, время выполнения, количество процессоров. Каждый из параметров моделируется случайной величиной. В [27] определены вид и параметры распределения случайных величин, проведен статистический анализ журналов работы нескольких суперкомпьютеров за длительные периоды времени. Тип задания не рассматривался, поэтому было принято решение определить его случайной величиной, равномерно распределенной от 1 до 8.

Для проведения экспериментов было сформировано множество входных потоков по 5 000 заданий каждый. В сгенерированном с помощью Lublin&Feitelson потоке обнаружились две существенные проблемы, затрудняющие моделирование и анализ его результатов: широкий диапазон времени поступления заданий в очередь и неоднородность сгенерированного потока. Рассмотрим подробнее каждую из проблем и предлагаемые способы их решения.

Для 5 000 заданий формировался входной поток, где последнее задание поступает в очередь через 30 суток после начала эксперимента. Время поступления последнего задания существенно отличалось в разных генерациях и составляло от 28 до 37 суток от начала эксперимента. При этом неочевидно, как можно изменить параметры входного потока для повышения интенсивности поступления заданий в очередь. При генерации использовалась сложная схема учета пиков поступления зада-

ния во время работы суперкомпьютера. В качестве вычислителя использовалась виртуальная суперкомпьютерная система из 500 узлов.

При запуске на 500 узлах сгенерированный с помощью Lublin&Feitelson входной поток оказался недостаточно интенсивным (рис. 3). Очередь заданий почти всегда держалась около нуля с редкими пиками. Общее время эксперимента составило 30 дней.

Загрузка вычислителя во время эксперимента представлена на рисунке 4. Большую часть времени вычислитель простаивал. Средняя загрузка за время эксперимента составила 20,8 % для алгоритма Backfill.

Было принято решение кратно уменьшить интервалы времени между заданиями, чтобы последнее поступило через 100 часов (приблизительно 4 дня) после начала эксперимента. Для этого определялся коэффициент масштабирования путем деления времени поступления последнего задания на число секунд в 100 часах, и на этот коэффициент разделялось каждое время поступления задания в очередь. Если последнее задание поступает через 30 суток после начала эксперимента и планируется ограничить эксперимент 4 сутками, то нужно уменьшить время поступления всех заданий в очередь в 30/4 = 7,5 раза.

После масштабирования на 100 часов число заданий в очереди почти всегда ненулевое, при этом очередь не нарастает бесконечно, как показано на рисунке 5.

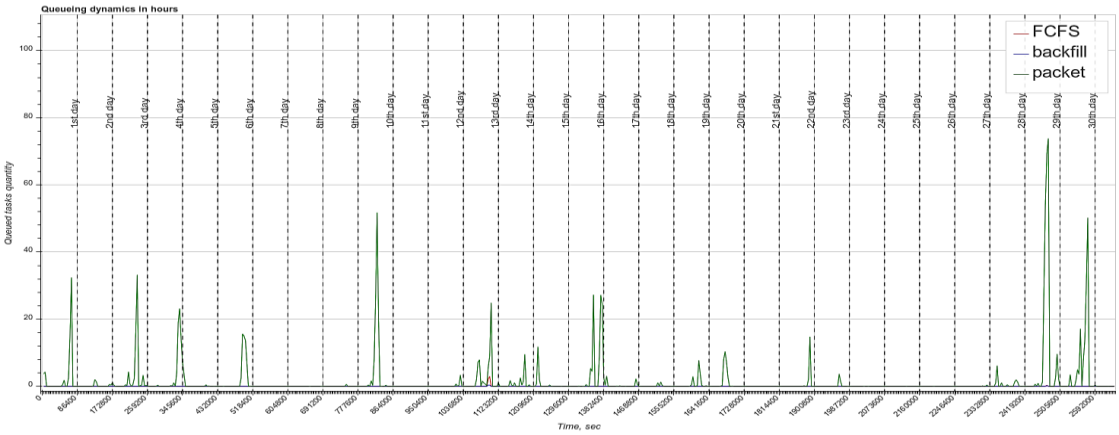

Рис. 3. Число заданий в очереди для 30 дней эксперимента с потоком, сгенерированным с помощью Lublin&Feitelson

Fig. 3. Queue length for 30 days of Lublin&Feitelson workload experiment

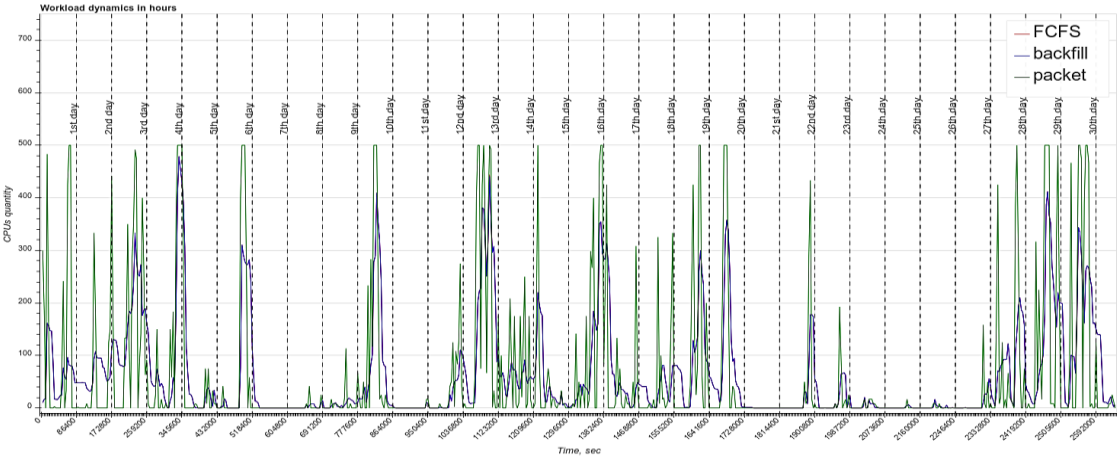

Рис. 4. Загрузка вычислителя для 30 дней эксперимента с потоком, сгенерированным с помощью Lublin&Feitelson

Fig. 4. Supercomputer utilization for 30 days of Lublin&Feitelson workload experiment

Масштабирование входного потока позволяет обеспечить одинаковую длительность каждого эксперимента вне зависимости от входного потока и получить входной поток с приемлемой загрузкой виртуального вычислителя (рис. 6).

Рассмотрим проблему неоднородности сгенерированного потока. Введем понятие расчетной загрузки U_{calc}, определяемой как $U_{calc} = \frac{\sum_{i=1}^{n} e_i \times p_i}{D \times P}$, где e_i – время обработки задания i; p_i – требуемое число узлов для задания i; $n = 5\,000$ – число заданий; $D = 4$ – длительность эксперимента до поступления в очередь последнего задания; P – число узлов виртуального суперкомпьютера.

Для 20 сгенерированных при помощи Lublin&Feitelson модельных потоков расчетная загрузка находилась в диапазоне от 0,47 до 1,08. Такой разброс осложняет применение сгенерированных потоков. Для потока с загрузкой в 0,47 моделируемая СУЗ будет функционировать в недогруженном режиме с простаиванием значительного объема вычислительных ресурсов, для потока с загрузкой 1,08 – в перегруженном режиме с бесконечно растущей очередью. Интерес представляют значения загрузки в диапазоне 0,85–0,95, которые являются распространенными для высоконагруженных СУЗ.

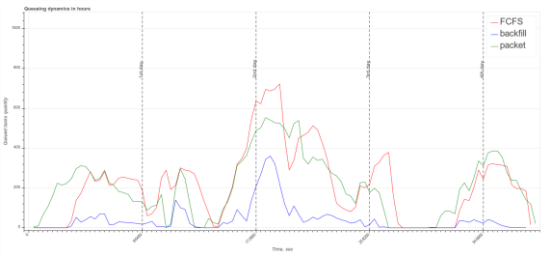

Рис. 5. Число заданий в очереди для 4 дней эксперимента с масштабированным на 100 часов потоком

Fig. 5. The number of jobs in the queue for 4 days of the experiment with the workload scaled by 100 hours

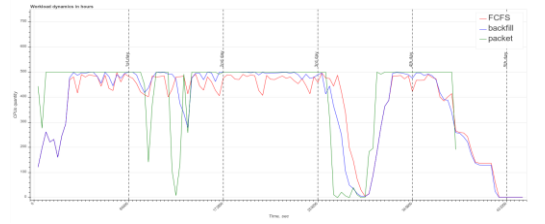

Рис. 6. Загрузка вычислителя для 4 дней эксперимента с масштабированным на 100 часов потоком

Fig. 6. Supercomputer utilization for 4 days of the experiment with the workload scaled by 100 hours

Для исследования модели в условиях более однородного входного потока генератор Lublin&Feitelson был модифицирован. Интервалы между поступлениями заданий в очередь смоделированы распределением Пуассона с интенсивностью $\lambda = 0{,}0142$, которая подбиралась эмпирически для обеспечения требуемой расчетной загрузки U_{calc}. Время выполнения заданий в соответствии с [27] было задано гамма-распределением с параметрами $\alpha = 4{,}2$ и $\beta = 0{,}94$. При этом для формирования интенсивного потока небольших заданий отбрасывались задания со временем выполнения менее e^5 (148 сек.) и более e^9 (2 980 сек. = 50 мин.). В результате при помощи модифицированного генератора сгенерировано 20 модельных потоков, для которых расчетная загрузка находилась в диапазоне от 0,8 до 0,95, средняя загрузка составила 0,86 со стандартным отклонением 0,04.

Рассмотрим характеристики сгенерированных входных потоков заданий. Первая группа потоков была сформирована с помощью генератора Lublin&Feitelson. Названия потоков этой группы начинаются с префикса orig. Вторая группа сформирована с помощью модифицированного генератора. Названия потоков этой группы начинаются с префикса mod. Расчетная загрузка в названии потока указана в суффиксе. В каждую группу вошло по три потока с расчетной загрузкой в 85 % (orig-85 и mod-85), 90 % (orig-90 и mod-90) и 95 % (orig-95 и mod-95).

Для каждого задания входных потоков рассчитывалась доля времени инициализации S, вычисляемая по формуле $S_i = \frac{s_i}{s_i + e_i}$, где s_i – время инициализации задания i; e_i – время обработки задания i.

Средняя по входному потоку доля времени инициализации S_{avg} рассчитывалась как $S_{avg} = \frac{\sum_{i=1}^{n} s_i}{\sum_{i=1}^{n} s_i + \sum_{i=1}^{n} e_i}$, где $n = 5\,000$ – число заданий во входном потоке.

Характеристики сгенерированных входных потоков приведены в таблице 1.

Для примера рассмотрим поток orig-90, имеющий очень большой разброс между поступлениями заданий в очередь. На диаграмме (рис. 7) представлено число заданий, для которых время до поступления следующего задания попадает в указанный интервал. Так, для 3 309 заданий время поступления следующего задания находится в интервале от 0 до 14 сек. У 236 заданий время до поступления следующего задания превышает 200 сек., при этом среднее значение для этих заданий 1 084 сек., а максимальная разница составляет 23 351 сек., то есть более 6 часов. Таким образом, поток заданий orig-90 неравномерен с точки зрения поступления заданий в очередь: 3 309 заданий (66 % всего потока) поступают почти одновременно (от 0 до 14 сек. между заданиями), при этом значительное число заданий (около 5 %) поступают в очередь с большой задержкой, измеряющейся в часах.

Для потока mod-90 распределение поступления заданий в очередь более равномерное (рис. 8). У 289 заданий время до поступления следующего задания превышает 200 сек., при этом среднее значение для этих заданий 271 сек., а максимальная разница составляет 560 сек., то есть менее 10 мин. Таким образом, поток заданий mod-90 более равномерен с точки зрения поступления заданий в очередь: 978 заданий (20 % всего потока) поступают почти одновременно (от 0 до 14 сек. между заданиями), для 719 заданий задержка составляет от 14 до 29 сек. и т.д. Максимальное время между заданиями не превышает 10 мин.

Время выполнения заданий из потока orig-90 также имеет большой разброс: 4 064 задания (80 % всего потока) выполняются менее 5 мин. (рис. 9). У 520 заданий время выполнения превышает 2 200 сек., при этом среднее значение для этих заданий 19 621 сек., а максимальное время выполнения составляет 130 093 сек., то есть более 36 часов. Поток orig-90 неоднороден по времени выполнения заданий, наблюдается разброс времени выполнения от секунд до суток.

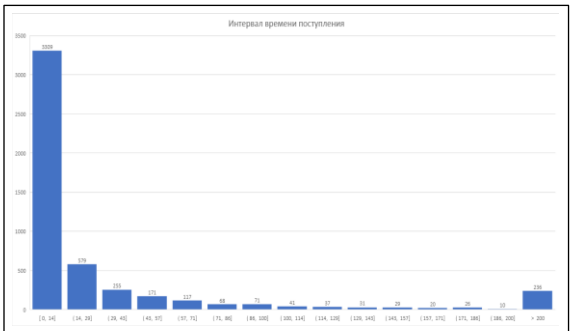

Рис. 7. Интервалы времени поступления следующего задания для потока orig-90 (задания с интервалом свыше 300 сек. добавлены к последнему столбцу)

Fig. 7. Intervals of the next job submit for the orig-90 workload (jobs with an interval greater than 300 seconds are added to the last column)

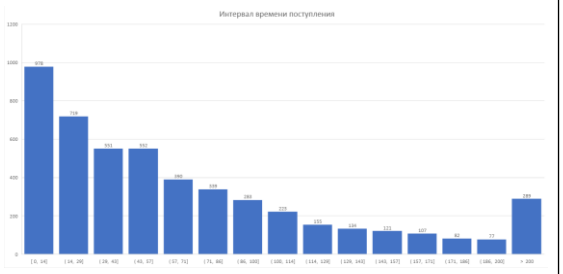

Рис. 8. Интервалы поступления следующего задания для потока mod-90

Fig. 8. Intervals of the next job submit for the mod-90 workload

Для потока mod-90 наблюдается более равномерное распределение времени выполнения заданий (рис. 10). Есть как небольшие задания (1 816, то есть 36 % всего потока заданий до 5 мин.), так и задания крупнее. У 223 заданий время выполнения превышает 2 200 сек., при этом среднее значение для этих заданий 2 571 сек., а максимальное время выполнения составляет 2 974 сек., то есть чуть менее 50 мин.

Таблица 1

Характеристики сгенерированных входных потоков

Table 1

Characteristics of generated workloads

Характеристика	orig-85	orig-90	orig-95	mod-85	mod-90	mod-95
Генератор потока	Lublin&Feitelson			Модифицированный авторами статьи		
Неоднородность потока	Высокая			Средняя		
Расчетная загрузка, %	85	90	95	85	90	95

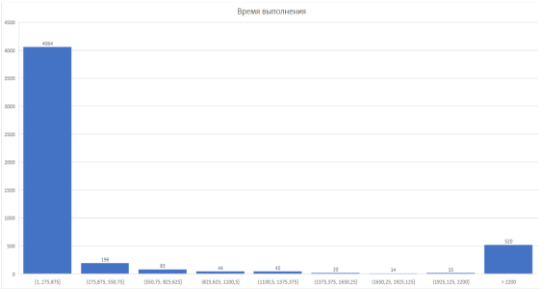

Рис. 9. Время выполнения задания для потока orig-90 (задания со временем выполнения свыше 2 500 сек. добавлены к последнему столбцу)

Fig. 9. Job runtime for the orig-90 workload (runtimes greater than 2 500 seconds are added to the last column)

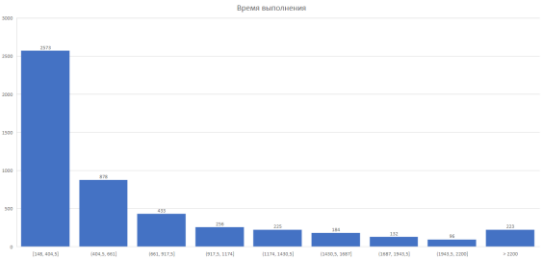

Рис. 10. Время выполнения задания для потока mod-90

Fig. 10. Job runtime for the mod-90 workload

Анализатор выходного потока

С помощью представленной имитационной модели была проведена серия из 96 экспериментов, для каждого из 6 сформированных входных потоков по 16 экспериментов с долей инициализации от 0 до 10 % с шагом 1 % и от 10 до 50 % с шагом 10 %. Рассмотрим два эксперимента для потока orig-90 – с 5 % и 10 % средней доли инициализации. На рисунке 11 представлено число заданий в очереди. Очевидно, что FCFS значительно проигрывает двум другим алгоритмам. Видно, что для первых двух дней очередь для Packet больше, а впоследствии ниже.

С ростом средней доли инициализации пакетирование почти всюду начинает показывать меньшую очередь, чем другие алгоритмы (рис. 12).

Разработанная программная система позволяет получать различные данные. Результаты экспериментов для разных долей инициализации с потоками оригинального генератора представлены в таблице 2. Цветом выделены первые показатели Packet, которые превзошли аналогичные показатели Backfill в эксперименте. Так, для orig-85 не наблюдается улучшения на представленных данных, для 10 % доли инициализации Packet уже показывает аналогичные с Backfill показатели эффективности, а при дальнейшем увеличении доли инициализации Packet превосходит Backfill. Для orig-90 медианное время ожидания для Packet лучше, чем для Backfill, начиная с доли инициализации в 6 %, а среднее время ожидания и средняя длина очереди улучшаются при доле выше 8 %. Для orig-95 показатели эффективности Packet превышают аналогичные для Backfill начиная с 6 % доли инициализации.

Результаты экспериментов для разных долей инициализации с потоками модифицированного генератора представлены в таблице 3. Для mod-85 улучшение характеристик от применения Packet наблюдается при 7 % доли инициализации. Для mod-90 медианное время ожидания для Packet лучше, чем для Backfill, начиная с доли инициализации в 2 %, а среднее время ожидания и средняя длина очереди улуч-

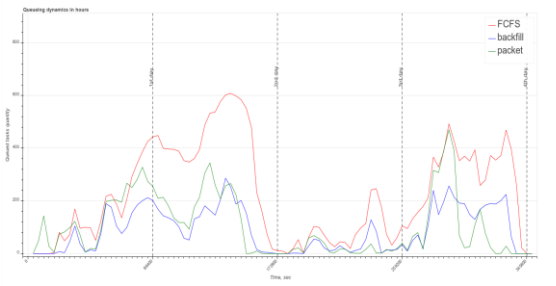

Рис. 11. Число заданий в очереди для потока orig-90 и 5 % средней доли инициализации

Fig. 11. The number of jobs in the queue for the orig-90 workload and 5 % average initialization share

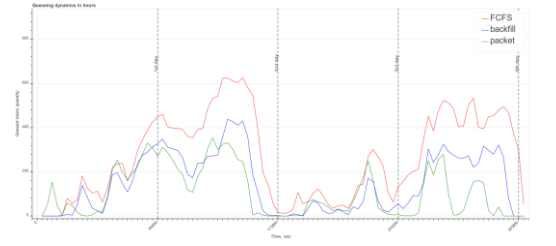

Рис. 12. Число заданий в очереди для потока orig-90 и 10 % средней доли инициализации

Fig. 12. The number of jobs in the queue for the orig-90 workload and 10 % average initialization share

Таблица 2

Сравнение трех потоков orig разной интенсивности для долей инициализации от 5 до 10 %

Table 2

Comparison of three orig workloads of different intensity for initialization shares from 5 to 10 %

Показатель эффективности	Доля инициализации, %	orig-85			orig-90			orig-95		
		FCFS	Backfill	Packet	FCFS	Backfill	Packet	FCFS	Backfill	Packet
Среднее время ожидания задания в очереди, сек.	5	11043	3488	11095	15923	6215	7141	25367	8923	11845
	6	11257	3949	7284	16582	7397	10555	25898	10245	**9673**
	7	11484	4411	6962	17314	8215	9308	26426	11722	9908
	8	11714	4981	6611	18167	8735	**6501**	26866	12554	6675
	9	11929	5254	5742	19234	10449	7649	27480	13694	9874
	10	12150	5761	**6224**	20017	11151	5462	28066	14643	13445
Медиана времени ожидания задания в очереди, сек.	5	9768	712	7681	10667	4465	3206	26301	6570	6643
	6	10113	941	2870	11446	5164	**5155**	27338	6528	**5664**
	7	10540	1309	2706	12134	5659	3852	282890	7772	4966
	8	10905	1717	3033	12960	6453	**3234**	29268	9388	3366
	9	11305	1693	2108	14021	7364	3681	29907	9885	5782
	10	11749	2177	3007	14655	7785	2376	30531	11121	8579
Полная утилизация, %	5	80,2	82,2	79,6	91,5	92,3	83,1	87,2	91,3	86,8
	6	80,6	82,6	73,3	91,9	92,9	84,8	87,4	91,7	85,3
	7	81,0	83,1	75,7	92,3	93,4	87,0	87,6	92,0	82,6
	8	81,5	83,6	81,1	92,6	93,8	81,3	87,8	92,3	80,2
	9	81,9	84,0	78,6	92,8	94,4	85,9	87,9	92,5	83,3
	10	82,4	84,5	79,8	93,0	94,9	78,5	88,1	92,7	88,0
Средняя длина очереди	5	159	51	158	230	90	103	351	128	168
	6	162	58	112	240	107	152	358	147	**145**
	7	164,4	65,9	99,7	250,4	118,9	134,1	365	167	140
	8	168	74	94	261	126	**94**	373	179	97
	9	170,8	78,6	81,3	272,5	151,2	110,2	382	194	140,5
	10	174	86	**89**	284	161	79	390	207	190

шаются при доле выше 5 %. Для mod-95 медианное время ожидания для Packet лучше, чем для Backfill, начиная с доли инициализации в 2 %, а среднее время ожидания и средняя длина очереди улучшаются при доле выше 3 %.

Заключение

Для сравнительного анализа влияния различных алгоритмов планирования на показатели эффективности СУЗ суперкомпьютера на базе симулятора Alea построена имитационная модель СУЗ. В рамках модели реализован алгоритм планирования Packet, основанный на методе формирования пакетов суперкомпьютерных заданий по типам. Модель включает в свой состав генератор входного потока заданий Lublin&Feitelson, симулятор Alea со встроенными алгоритмами Backfill и FCFS и реализованным алгоритмом Packet, а также анализатор выходного потока заданий.

Для генератора Lublin&Feitelson выявлены недостатки, существенно затрудняющие моделирование: широкий диапазон времен поступления заданий в очередь и высокая неоднородность сгенерированного потока заданий. Для устранения недостатков генератор Lublin& Feitelson был модифицирован. В модифицированном генераторе проблема широкого диапазона решена масштабированием времени поступления заданий в очередь. Неоднородность входного потока заданий преодолена путем моделирования времени поступления задания в очередь с помощью распределения Пуассона и отбрасыванием заданий со слишком большим или слишком малым временем выполнения. Для исследования предложенного алгоритма Packet сгенерированы входные потоки

Таблица 3

Сравнение трех потоков mod разной интенсивности для долей инициализации от 2 до 7 %

Table 3

Comparison of three mod workloads of different intensity for initialization shares from 2 to 7 %

Показатель эффективности	Доля инициализации, %	mod-85			mod-90			mod-95		
		FCFS	Backfill	Packet	FCFS	Backfill	Packet	FCFS	Backfill	Packet
Среднее время ожидания задания в очереди, сек.	2	5300	957	1250	20759	1664	2060	24723	2578	2652
	3	5961	1042	1211	21724	1928	2168	25528	2921	**2843**
	4	7176	1189	1437	22886	2234	2344	26861	3514	3061
	5	8122	1280	1486	23988	2657	**2610**	28220	3968	3388
	6	9039	1454	1755	25261	3122	2837	29414	4856	3724
	7	10046	1565	**1438**	26351	3703	2258	30631	6056	3227
Медиана времени ожидания задания в очереди, сек.	2	4025	545	540	22750	1079	**1074**	22851	1874	**1562**
	3	4622	639	572	23623	1249	1141	23759	2190	**1790**
	4	6008	789	736	24269	1543	1348	24997	2799	2083
	5	7161	846	825	25371	1968	**1533**	26147	3167	2390
	6	8158	1002	975	26685	2395	1717	27294	3934	2608
	7	9232	1107	**855**	27456	3070	1512	28405	4854	2304
Полная утилизация, %	2	82,4	85,8	92,6	80,1	90,6	95,1	81,9	94,3	97,0
	3	82,5	86,6	94,4	80,2	91,2	96,4	82,1	94,9	97,9
	4	82,7	87,5	96,1	80,3	92,0	97,4	82,1	95,7	98,7
	5	82,9	88,3	97,2	80,5	92,9	98,5	82,2	96,1	99,1
	6	83,0	89,0	98,0	80,6	93,6	99,1	82,3	96,5	99,4
	7	83,1	89,7	98,4	80,5	94,3	99,0	82,3	96,5	99,4
Средняя длина очереди	2	78	14	18	293	24	30	364	37	38
	3	88	15	17	306	28	32	378	42	**41**
	4	105	17	21	324	32	33	397	51	44
	5	119	18	21	339	38	**37**	414	57	49
	6	133	21	25	356	45	40	430	70	54
	7	147	23	**21**	373	53	32	448	87	47

заданий с разной интенсивностью и разной долей времени инициализации заданий. С применением сформированных входных потоков заданий проведены эксперименты, позволившие определить минимальную долю инициализации заданий, при которой применение алгоритма Packet заметно повышает показатели эффективности СУЗ.

Таким образом, предложенная имитационная модель СУЗ на базе симулятора Alea является инструментальным программным средством, которое позволяет исследователю оценить эффект от применения произвольного алгоритма планирования для произвольного сгенерированного входного потока заданий.

Работа выполнена в МСЦ РАН в рамках государственного задания по теме FNEF-2022-0016.

Литература

1. Savin G.I., Shabanov B.M., Telegin P.N. et al. Joint supercomputer center of the Russian academy of sciences: Present and future. Lobachevskii J. of Mathematics, 2019, vol. 40, no. 11, pp. 1853–1862. DOI: 10.1134/S1995080219110271.
2. Lyakhovets D.S., Baranov A.V. Group based job scheduling to increase the high-performance computing efficiency. Lobachevskii J. of Mathematics, 2020, vol. 41, no. 12, pp. 2558–2565. DOI: 10.1134/S1995080220120264.
3. Баранов А.В., Ляховец Д.С. Методы и средства моделирования системы управления суперкомпьютерными заданиями // Программные продукты и системы. 2019. Т. 32. № 4. С. 581–594. DOI: 10.15827/0236-235X.128.581-594.
4. Legrand I.C., Newman H.B. The MONARC toolset for simulating large network-distributed processing systems. Proc. Winter Simulation Conf., 2000, vol. 2, pp. 1794–1801. DOI: 10.1109/WSC.2000.899171.
5. Klusáček D., Soysal M., Suter F. Alea – complex job scheduling simulator. Parallel processing and applied mathematics. In: PPAM, 2020, pp. 217–229. DOI: 10.1007/978-3-030-43222-5_19.

6. Bell W.H., Cameron D.G., Millar F.P., Capozza L., Stockinger K., Zini F. Optorsim: A grid simulator for studying dynamic data replication strategies. The Int. J. of High Performance Computing Applications, 2003, vol. 17, no. 4, pp. 403–416. DOI: 10.1177/10943420030174005.

7. Chen W., Deelman E. WorkflowSim: A toolkit for simulating scientific workflows in distributed environments. Proc. IEEE VIII Int. Conf. on E-Science, 2012, pp. 1–8. DOI: 10.1109/eScience.2012.6404430.

8. Dutot P.F., Mercier M., Poquet M., Richard O. Batsim: a realistic language-independent resources and jobs management systems simulator. In: Job Scheduling Strategies for Parallel Processing, 2015, pp. 178–197. DOI: 10.1007/978-3-319-61756-5_10.

9. Simakov N.A., Innus M.D., Jones M.D. et al. A slurm simulator: Implementation and parametric analysis. In: Lecture Notes in Computer Science, 2017, vol. 10724, pp. 197–217. DOI: 10.1007/978-3-319-72971-8_10.

10. Baranov A.V., Lyakhovets D.S. Accuracy comparison of various supercomputer job management systems. Lobachevskii J. of Mathematics, 2021, vol. 42, no. 11, pp. 2510–2519. DOI: 10.1134/S1995080 22111007X.

11. Agung M., Watanabe Y., Weber H., Egawa R., Takizawa H. Preemptive parallel job scheduling for heterogeneous systems supporting urgent computing. IEEE Access, 2021, vol. 9, pp. 17557–17571. DOI: 10.1109/ACCESS.2021.3053162.

12. Jaros M., Jaros J. Performance-cost optimization of moldable scientific workflows. In: Job Scheduling Strategies for Parallel Processing, 2021, vol. 12985, pp. 149–167. DOI: 10.1007/978-3-030-88224-2_8.

13. Dakkak O., Fazea Y., Awang Nor S., Arif S. Towards accommodating deadline driven jobs on high performance computing platforms in grid computing environment. J. of Computational Science, 2021, vol. 54, art. 101439. DOI: 10.1016/j.jocs.2021.101439.

14. Hijab M., Damodaram A. Efficient online-task scheduling in distributed computing environments using least average load variance algorithm. Proc. Int. Conf. on Advances in Computer Engineering and Communication Systems. LAIS, 2021, vol. 20, pp. 463–471. DOI: 10.1007/978-981-15-9293-5_42.

15. Matsui Y., Watashiba Y., Date S., Yoshikawa T., Shimojo S. Job scheduling simulator for assisting the mapping configuration between queue and computing nodes. Proc. AINA, 2019, vol. 926, pp. 1024–1033. DOI: 10.1007/978-3-030-15032-7_86.

16. Cirne W., Berman F. A model for moldable supercomputer jobs. Proc. XV IPDPS, 2001, p. 8. DOI: 10.1109/IPDPS.2001.925004.

17. Byun C., Arcand W., Bestor D., Bergeron B. et al. Node-based job scheduling for large scale simulations of short running jobs. Proc. IEEE HPEC, 2021, pp. 1–7. DOI: 10.1109/HPEC49654.2021.9622870.

18. Ade G.S., Amdani S.Y. Survey on dynamic group job scheduling in grid computing. IJETR, 2016, vol. 5, no. 1, pp. 63–66.

19. Latchoumy P., Khader P.S. Grouping based scheduling with resource failure handling in computational grid. JATIT, 2014, vol. 63, no. 3, pp. 605–614.

20. Muthuvelu N., Vecchiola C., Chai I., Eswaran C., Buyya R. Task granularity policies for deploying bag-of-task applications on global grids. Future Generation Computer Systems, 2013, vol. 29, no. 1, pp. 170–181. DOI: 10.1016/j.future.2012.03.022.

21. Sandeep K., Sukhpreet K. Efficient load balancing grouping based job scheduling algorithm in grid computing. IJETTCS, 2013, vol. 2, no. 4, pp. 138–144.

22. Pinky R. Grouping based job scheduling algorithm using priority queue and hybrid algorithm in grid computing. Int. J. of Grid Computing & Applications, 2012, vol. 3., no. 4, pp. 55–65. DOI: 10.5121/ijgca. 2012.3405.

23. Belabid J., Aqil S., Allali K. Solving permutation flow shop scheduling problem with sequence-independent setup time. J. of Applied Mathematics, 2020, pp. 1–11. DOI: 10.1155/2020/7132469.

24. Corbalan J., D'Amico M. Modular workload format: Extending SWF for modular systems. Proc. JSSPP. Lecture Notes in Computer Science, 2021, vol. 12985, pp. 43–55. DOI: 10.1007/978-3-030-88224-2_3.

25. The Standard Workload Format. URL: https://www.cs.huji.ac.il/labs/parallel/workload/swf.html (дата обращения: 21.06.2022).

26. Buyya R., Murshed M. GridSim: A toolkit for the modeling and simulation of distributed resource management and scheduling for Grid computing. Concurrency and Computation: Practice and Experience, 2002, vol. 14, no. 13-15, pp. 1175–1220. DOI: 10.1002/cpe.710.

27. Lublin U., Feitelson D.G. The workload on parallel supercomputers: modeling the characteristics of rigid jobs. J. of Parallel and Distributed Computing, 2003, vol. 63, no. 11, pp. 1105–1122. DOI: 10.1016/ S0743-7315(03)00108-4.

Software & Systems
DOI: 10.15827/0236-235X.140.631-643

Received 05.07.22
2022, vol. 35, no. 4, pp. 631–643

Modelling a supercomputer job bundling system based on the Alea simulator

A.V. Baranov [1], *Ph.D. (Engineering), Lead Researcher, antbar@mail.ru, abaranov@jscc.ru*
D.S. Lyakhovets [1], *Junior Researcher, anetto@inbox.ru*

[1] *Joint Supercomputer Center of the RAS – branch of Federal State Institution*
Scientific Research Institute for System Analysis of the RAS, Moscow, 119334, Russian Federation

Abstract. Modern supercomputer job management systems (JMS) are complex software using many different scheduling algorithms with various parameters. We cannot predict or calculate the impact of changing these parameters on JMS quality metrics. For this reason, researchers use simulation modelling to determine the optimal JMS parameters.

This article discusses the problem of developing a supercomputer job management system model based on the well-known Alea simulator. The object of study is our scheduling algorithm used for developing the supercomputer job bundling system. The algorithm bundles jobs with a long initialization time into groups (packets) according to job types. Initialization is performed once for each group, and then the jobs of the group are executed one after the other. By using a bundling system, it is possible to reduce the initialization overhead and increase the job scheduling efficiency. We implemented the bundling algorithm as a part of the Alea simulator. We have done comparative simulation of implemented algorithm for various workloads. The comparison involved the FCFS and Backfill scheduling algorithms built into Alea. Several workloads with different intensities were generated for the simulation. The minimum job initialization share thresholds for these workloads were determined based on the simulation results. The bundling system noticeably improves the scheduling efficiency compared to the FCFS and Backfill algorithms starting from these thresholds.

The study results showed that the developed simulation model could be used as a software tool for a comparative analysis of various algorithms for supercomputer job scheduling.

Keywords: high performance computing, job management system, supercomputer job scheduling, simulation, simulation model, job bundling, Alea.

Acknowledgements. *The work has been done at the ISC RAS within the framework of the state assignment on the topic FNEF-2022-0016.*

References

1. Savin G.I., Shabanov B.M., Telegin P.N. et al. Joint supercomputer center of the Russian academy of sciences: Present and future. *Lobachevskii J. of Mathematics*, 2019, vol. 40, no. 11, pp. 1853–1862. DOI: 10.1134/S1995080219110271.
2. Lyakhovets D.S., Baranov A.V. Group based job scheduling to increase the high-performance computing efficiency. *Lobachevskii J. of Mathematics*, 2020, vol. 41, no. 12, pp. 2558–2565. DOI: 10.1134/S1995080220120264.
3. Baranov A.V., Lyakhovets D.S. Methods and tools for modeling supercomputer job management system. *Software & Systems*, 2019, vol. 32, no. 4, pp. 581–594. DOI: 10.15827/0236-235X.128.581-594 (in Russ.).
4. Legrand I.C., Newman H.B. The MONARC toolset for simulating large network-distributed processing systems. *Proc. Winter Simulation Conf.*, 2000, vol. 2, pp. 1794–1801. DOI: 10.1109/WSC.2000.899171.
5. Klusáček D., Soysal M., Suter F. Alea – complex job scheduling simulator. Parallel processing and applied mathematics. In: *PPAM*, 2020, pp. 217–229. DOI: 10.1007/978-3-030-43222-5_19.
6. Bell W.H., Cameron D.G., Millar F.P., Capozza L., Stockinger K., Zini F. Optorsim: A grid simulator for studying dynamic data replication strategies. *The Int. J. of High Performance Computing Applications*, 2003, vol. 17, no. 4, pp. 403–416. DOI: 10.1177/10943420030174005.
7. Chen W., Deelman E. WorkflowSim: A toolkit for simulating scientific workflows in distributed environments. *Proc. IEEE VIII Int. Conf. on E-Science*, 2012, pp. 1–8. DOI: 10.1109/eScience.2012.6404430.
8. Dutot P.F., Mercier M., Poquet M., Richard O. Batsim: a realistic language-independent resources and jobs management systems simulator. In: *Job Scheduling Strategies for Parallel Processing*, 2015, pp. 178–197. DOI: 10.1007/978-3-319-61756-5_10.
9. Simakov N.A., Innus M.D., Jones M.D. et al. A slurm simulator: Implementation and parametric analysis. In: *Lecture Notes in Computer Science*, 2017, vol. 10724, pp. 197–217. DOI: 10.1007/978-3-319-72971-8_10.

10. Baranov A.V., Lyakhovets D.S. Accuracy comparison of various supercomputer job management systems. *Lobachevskii J. of Mathematics*, 2021, vol. 42, no. 11, pp. 2510–2519. DOI: 10.1134/S1995080 22111007X.

11. Agung M., Watanabe Y., Weber H., Egawa R., Takizawa H. Preemptive parallel job scheduling for heterogeneous systems supporting urgent computing. *IEEE Access*, 2021, vol. 9, pp. 17557–17571. DOI: 10.1109/ACCESS.2021.3053162.

12. Jaros M., Jaros J. Performance-cost optimization of moldable scientific workflows. In: *Job Scheduling Strategies for Parallel Processing*, 2021, vol. 12985, pp. 149–167. DOI: 10.1007/978-3-030-88224-2_8.

13. Dakkak O., Fazea Y., Awang Nor S., Arif S. Towards accommodating deadline driven jobs on high performance computing platforms in grid computing environment. *J. of Computational Science*, 2021, vol. 54, art. 101439. DOI: 10.1016/j.jocs.2021.101439.

14. Hijab M., Damodaram A. Efficient online-task scheduling in distributed computing environments using least average load variance algorithm. *Proc. Int. Conf. on Advances in Computer Engineering and Communication Systems. LAIS*, 2021, vol. 20, pp. 463–471. DOI: 10.1007/978-981-15-9293-5_42.

15. Matsui Y., Watashiba Y., Date S., Yoshikawa T., Shimojo S. Job scheduling simulator for assisting the mapping configuration between queue and computing nodes. *Proc. AINA*, 2019, vol. 926, pp. 1024–1033. DOI: 10.1007/978-3-030-15032-7_86.

16. Cirne W., Berman F. A model for moldable supercomputer jobs. *Proc. XV IPDPS*, 2001, p. 8. DOI: 10.1109/IPDPS.2001.925004.

17. Byun C., Arcand W., Bestor D., Bergeron B. et al. Node-based job scheduling for large scale simulations of short running jobs. *Proc. IEEE HPEC*, 2021, pp. 1–7. DOI: 10.1109/HPEC49654.2021.9622870.

18. Ade G.S., Amdani S.Y. Survey on dynamic group job scheduling in grid computing. *IJETR*, 2016, vol. 5, no. 1, pp. 63–66.

19. Latchoumy P., Khader P.S. Grouping based scheduling with resource failure handling in computational grid. *JATIT*, 2014, vol. 63, no. 3, pp. 605–614.

20. Muthuvelu N., Vecchiola C., Chai I., Eswaran C., Buyya R. Task granularity policies for deploying bag-of-task applications on global grids. *Future Generation Computer Systems*, 2013, vol. 29, no. 1, pp. 170–181. DOI: 10.1016/j.future.2012.03.022.

21. Sandeep K., Sukhpreet K. Efficient load balancing grouping based job scheduling algorithm in grid computing. *IJETTCS*, 2013, vol. 2, no. 4, pp. 138–144.

22. Pinky R. Grouping based job scheduling algorithm using priority queue and hybrid algorithm in grid computing. *Int. J. of Grid Computing & Applications*, 2012, vol. 3., no. 4, pp. 55–65. DOI: 10.5121/ijgca.2012. 3405.

23. Belabid J., Aqil S., Allali K. Solving permutation flow shop scheduling problem with sequence-independent setup time. *J. of Applied Mathematics*, 2020, pp. 1–11. DOI: 10.1155/2020/7132469.

24. Corbalan J., D'Amico M. Modular workload format: Extending SWF for modular systems. *Proc. JSSPP. Lecture Notes in Computer Science*, 2021, vol. 12985, pp. 43–55. DOI: 10.1007/978-3-030-88224-2_3.

25. *The Standard Workload Format*. Available at: https://www.cs.huji.ac.il/labs/parallel/workload/swf.html (accessed June 21, 2022).

26. Buyya R., Murshed M. GridSim: A toolkit for the modeling and simulation of distributed resource management and scheduling for Grid computing. *Concurrency and Computation: Practice and Experience*, 2002, vol. 14, no. 13-15, pp. 1175–1220. DOI: 10.1002/cpe.710.

27. Lublin U., Feitelson D.G. The workload on parallel supercomputers: modeling the characteristics of rigid jobs. *J. of Parallel and Distributed Computing*, 2003, vol. 63, no. 11, pp. 1105–1122. DOI: 10.1016/S0743-7315(03)00108-4.

Для цитирования

Баранов А.В., Ляховец Д.С. Имитационная модель системы пакетирования суперкомпьютерных заданий на базе симулятора Alea // Программные продукты и системы. 2022. Т. 35. № 4. С. 631–643. DOI: 10.15827/0236-235X.140.631-643.

For citation

Baranov A.V., Lyakhovets D.S. Modelling a supercomputer job bundling system based on the Alea simulator. *Software & Systems*, 2022, vol. 35, no. 4, pp. 631–643 (in Russ.). DOI: 10.15827/0236-235X.140.631-643.