Симулятор системы управления суперкомпьютерными заданиями с внешним интерфейсом управления

Д. С. Ляховец[1], А. В. Баранов[2], А. Ю. Кудрин[3]

[1]МСЦ РАН – филиал ФГУ ФНЦ НИИСИ РАН, НИЦ «Курчатовский институт», Москва, Россия, anetto@inbox.ru;

[2] МСЦ РАН – филиал ФГУ ФНЦ НИИСИ РАН, НИЦ «Курчатовский институт», Москва, Россия, abaranov@jscc.ru;

[3] МСЦ РАН – филиал ФГУ ФНЦ НИИСИ РАН, Москва, Россия, akudrin2002@gmail.com

Аннотация. Одним из востребованных инструментов исследования поведения систем управления суперкомпьютерными заданиями (СУЗ), как сложных систем коллективного пользования, является имитационное моделирование при помощи симуляторов. В статье сформулированы требования к симулятору высокопроизводительной вычислительной системы, включающей в свой состав территориально распределенные суперкомпьютеры. Соответствие выдвинутым требованиям может быть обеспечено за счет реализации внешнего интерфейса управления симулятором. В статье представлен анализ характеристик современных симуляторов СУЗ с точки зрения выдвинутых требований, предложена архитектура симулятора СУЗ с внешним симулятором управления. Рассмотрены первые результаты применения симулятора Elytra, реализующего предложенную архитектуру.

Ключевые слова: высокопроизводительные вычисления, имитационное моделирование, планирование заданий, система управления заданиями

1. Введение

Современные системы высокопроизводительных вычислений представляют собой сложные технологические комплексы, в состав которых может входить множество территориально распределенных вычислительных установок (суперкомпьютеров). Для проведения расчетов в такой вычислительной системе пользователь формирует так называемое задание – информационный объект, включающий расчетную программу, требования к ресурсам и исходные данные. Расчетная программа реализует прикладной параллельный алгоритм, который может быть выполнен на некотором множестве вычислительных узлов (процессорных ядер) суперкомпьютера. Минимальный набор требований к ресурсам определяется числом необходимых заданию узлов (процессорных ядер) и временем выполнения расчетной программы.

Сформированные пользователями задания образуют входной поток, поступающий в специализированную программную систему управления заданиями (СУЗ). СУЗ ведет одну или несколько очередей заданий и выполняет функцию их планирования, т.е. составления расписания запусков заданий на узлах суперкомпьютера. Современные СУЗ являются сложными многопараметрическими системами, поведение которых существенно зависит от значений конфигурационных параметров и характеристик входных потоков заданий, причем эту зависимость практически невозможно определить аналитически. В случае, когда СУЗ управляет заданиями в территориально распределенной системе (ТРС) высокопроизводительных вычислений, сложность системы только возрастает: в составе СУЗ, например, может появиться несколько уровней иерархии управления [1]. В этой связи для исследования алгоритмов планирования СУЗ и определения их оптимальных параметров широко применяется имитационное моделирование [2].

Имитационное моделирование подразумевает создание модели или симулятора СУЗ, повторяющих характеристики реальной системы. Моделирование позволяет исследовать поведение СУЗ без вмешательства в работу отдельного суперкомпьютера или ТРС в целом.

В соответствии с [1] в ТРС выделяют два уровня иерархии управления заданиями. На верхнем уровне ведется глобальная очередь, задания из которой в соответствии с некоторым алгоритмом планирования распределяются на нижний уровень иерархии, состоящий из очередей т.н. локальных СУЗ. Каждая локальная СУЗ управляет одной вычислительной установкой (суперкомпьютером) из состава ТРС и может принимать как задания из глобальной очереди,

так и непосредственно от пользователей своего суперкомпьютера. В результате образуются несколько входных потоков заданий: один глобальный и множество локальных.

Имитационная модель ТРС в этом случае естественным образом составляется из совокупности симуляторов глобальной и локальных СУЗ. При этом для модели ТРС необходимо обеспечить выполнение ряда принципов:

− реализовать работу с динамическим входным потоком заданий, то есть позволять добавлять и удалять задания из очередей СУЗ во время работы симуляторов;

− обеспечить доступ к очереди локальной СУЗ во время моделирования;

− обеспечить синхронизацию модельного времени между всеми экземплярами симуляторов СУЗ, участвующих в моделировании ТРС.

Указанные принципы реализуются в виде некоторого протокола, который будем называть внешним интерфейсом управления симулятора СУЗ. Внешний интерфейс управления позволит реализовать имитационную модель ТРС на основе симуляторов СУЗ.

2. Сравнительный анализ существующих симуляторов

Можно выделить два класса симуляторов: симуляторы распределённых систем и симуляторы локальных СУЗ. Симуляторы распределённых систем можно адаптировать для моделирования ТРС, но в таких симуляторах на уровне локальной СУЗ реализованы собственные алгоритмы планирования, что может негативно повлиять на точность моделирования.

Перспективным является построение симулятора ТРС на основе симуляторов СУЗ, для которых проведено исследование точности воспроизведения моделируемой СУЗ.

Сфокусируем внимание на двух важных аспекта работы симулятора. Во-первых, проанализируем, соответствуют ли существующие симуляторы выдвинутым требованиям. Во-вторых, рассмотрим, как разработчики того или иного симулятора валидировали свой продукт, то есть сравнивали его с реальной системой или другими симуляторами.

2.1. Симуляторы распределённых систем

Нами проанализированы следующие симуляторы распределённых систем:

– Optorsim [3];
– GSSIM [4];
– WorkflowSim [5];
– GroudSim [6].

Симулятор Optorsim разработан для моделирования грид-систем с упором на задания с интенсивными процессами обмена между вычислительными узлами. Центральной настройкой симулятора является матрица пропускной способности между вычислительными узлами.

В литературе не удалось найти сведений о валидации Optorsim. К существенным ограничениям Optorsim следует отнести работу только со статическим входным потоком, а также отсутствие возможности доступа к локальной очереди заданий во время симуляции и синхронизации модельного времени.

Программная платформа GSSIM создана в 2011 году и в значительной мере похожа на модель, пригодную для моделирования ТРС. В ней возможно реализовать как собственный алгоритм планирования уровня ТРС (реализация глобальной очереди, broker scheduling plug-in в терминологии авторов), так и алгоритм планирования уровня СУЗ (scheduling plug-in).

Как и в случае с Optorsim, сведений о валидации GSSIM нет, а сам симулятор GSSIM поддерживает работу только со статическим входным потоком. Судя по описанию GSSIM, у алгоритма планирования уровня ТРС есть доступ к локальной очереди заданий, реализована синхронизация модельного времени. При этом веб-сайт GSSIM не работает, и мы не смогли найти исходные тексты какой-либо версии кода.

Симулятор WorkflowSim расширяет функционал платформы CloudSim, предназначенной для имитации обработки потока заданий в облачных средах. WorkflowSim предлагается использовать для исследования планирования заданий в распределённых системах с учётом накладных расходов, связанных с отказами компонентов вычислительной системы. Симулятор является моделью ТРС и не основан на валидированном симуляторе СУЗ. При этом алгоритм планирования уровня ТРС необходимо выбрать из списка доступных, и, судя по описанию, нельзя заменить на самостоятельно реализованный. В модели используется событийное моделирование, всё взаимодействие внутри неё происходит через очереди сообщений. Это позволяет организовать единое модельное время для всей системы. В статье [5] нет сведений о возможности доступа к локальной очереди заданий СУЗ. Динамический входной поток заданий не поддерживается. Для валидации симулятора авторы на наборе из 10 422 заданий проанализировали точность, как отношение предсказанного моделью общего времени работы к реальному времени работы из журнала СУЗ. В 2021 году на официальной странице симулятора появилось предупреждение, что симулятор более не поддерживается.

Симулятор GroudSim является инструментом для имитации научных процессов и сосредоточен на моделировании и анализе алгоритмов планирования в грид-системах. Для синхронизации модельного времени авторы использовали механизм дискретных событий, что позволяет интегрировать этот симулятор в модель ТРС. Симулятор не предоставляет возможности работы с локальной очередью извне модели, а также возможность работы с динамическим входным потоком заданий. Для валидации авторы провели эксперименты по сравнению симулятора с GridSim по общему времени моделирования на нескольких наборах данных.

2.2. Симуляторы СУЗ

Нами проанализированы следующие симуляторы СУЗ:

– Batsim [7];

– Performance Prediction Toolkit (PPT) [8];

– Alea [9].

Симулятор Batsim создан для исследования различных планировщиков, и во главу угла авторами поставлена валидация самого симулятора с СУЗ OAR [10]. Авторы оформили алгоритм планирования из OAR как плагин для Batsim и провели серию экспериментов на 9 сгенерированных входных потоках, обработанных на реальной системе и в симуляторе Batsim. Авторы визуально сравнивали диаграмму Гантта расписания запусков заданий, анализировали следующие графики сравнения работы симулятора и реальной системы для каждого задания:

− разница времени выполнения заданий;

− разница времени поступления задания в очередь;

− разница времени полной обработки задания (turnaround time) с момента поступления задания в очередь до завершения выполнения.

Для времени полной обработки задания в дополнение к графику для каждого заданий был построен статистический график с плотностью вероятности значения времени полной обработки.

Batsim работает только со статическим входным потоком. В Batsim не реализованы доступ к локальной очереди заданий во время симуляции и синхронизация модельного времени.

Симулятор PPT является параллельным дискретно-событийным симулятором, написанным на Python и предназначенным для быстрого анализа и прогнозирования производительности научных приложений в суперкомпьютерах. В PPT реализованы несколько алгоритмов планирования, такие как First Come First Served, Shortest Job First и другие. Алгоритм Backfill не реализован и отмечен в направлениях дальнейших исследований. Симулятор PPT работает только со статическим входным потоком заданий. Модельное время и доступ к локальной очереди заданий не реализованы. Для валидации авторы провели серию экспериментов по сравнению с симулятором PYSS (Python Scheduler Simulator), для которого нет опубликованных статей и информации о его валидации. При обработке результатов эксперимента авторы визуально сравнивают графики загрузки вычислительных ресурсов и числа запущенных заданий для алгоритма First Come First Served.

Одним из широко используемых решений для моделирования СУЗ в настоящее время является симулятор Alea. В работе [11] с помощью симулятора Alea исследуется влияние скорректированного заказанного времени (soft walltime) на характеристики системы. Позднее в работе [12] тот же авторский коллектив анализирует результаты внедрения исследованного ранее механизма soft walltime с предсказанным временем на грид-системе MetaCentrum в Чехии. Направлением дальнейших работ в [12] указано более детальное исследование других вариантов предсказания времени на симуляторе. Другие исследователи на базе Alea изучают системы пакетирования заданий [13]. В работе [14] представлено расширение Alea, позволяющее оценивать характеристики предоперационного планирования исследования пациентов с использованием ультразвука для лечения онкологических заболеваний.

На вход симулятора подаётся журнал работы реального суперкомпьютера в специализированном формате Standard Workflow Format с характеристиками заданий. Центральной частью симулятора является планировщик, в котором реализован внешний алгоритм планирования заданий. Алгоритм планирования можно выбрать из набора встроенных (FCFS, Shortest Job First, EASY backfilling, Conservative backfilling и другие) или реализовать самостоятельно, соблюдая предлагаемую спецификацию. Планировщик определяет время запуска каждого задания на виртуальном вычислителе, параметры которого также задаются пользователем. В результате работы Alea формирует выходной файл с построенным расписанием и предоставляет различные графики, такие как построенное расписание, средняя загрузка вычислительных ресурсов, число заданий в очереди и на вычислителе и другие. По нашему опыту, для входных потоков с большим числом заданий (порядка нескольких тысяч) графики Alea перестают корректно строиться.

Разработчики Alea не проводили валидацию своего симулятора, так как предлагают собственные реализации алгоритмов планирования, не связанные с планировщиками СУЗ. В работе

[15] мы проводили сравнение точности моделирования симулятора Alea, симулятора отечественной Системы управления прохождением параллельных заданий (СУППЗ) [16] с виртуальным вычислителем и реального журнала работы суперкомпьютера.

Работа с динамическим входным потоком (dynamic workload) заявлена в статье [9], но в инструкции пользователя самого проекта Alea этот функционал отсутствует, как и примеры его использования в конфигурационном файле. В статье под динамическим входным потоком подразумевается возможность увеличения или уменьшения интервала времени до поступления следующего задания в зависимости от производительности планировщика, что не вполне удовлетворяет описанному нами принципу работы с динамическим входным потоком заданий. В Alea не реализованы доступ к локальной очереди заданий во время симуляции и синхронизация модельного времени.

Результаты сравнительного анализа представлены в таблице 1. На его основе авторами было принято решение создать собственную модель СУЗ с внешним интерфейсом управления.

Таблица 1. Результаты сравнительного анализа симуляторов

Симулятор	Динамический входной поток	Получение состояние очереди	Синхронизация модельного времени	Год обновления (статья/код)
Alea	Заявлено, но не работает	Не реализовано	Не реализовано	2020/2023
Batsim	Не реализовано	Не реализовано	Не реализовано	2017/2024
Optorsim	Не реализовано	Не реализовано	Не реализовано	2016/2015
PPT	Не реализовано	Не реализовано	Не реализовано	2017/2021
GSSIM	Не реализовано	Реализовано	Реализовано	2011/2014
WorkflowSim	Не реализовано	Не реализовано	Реализовано	2012/2015
GroudSim	Не реализовано	Не реализовано	Реализовано	2011/не доступен

3. Симулятор Elytra

Разработанный нами симулятор получил название Elytra. При его создании мы использовали опыт, полученный при использовании симулятора Alea. Например, в работе [17] в симуляторе Alea был реализован алгоритм пакетирования заданий. За время длительной эксплуатации симулятора Alea нами был накоплен набор инструментов для обработки выходного потока заданий Alea, позволяющих рассчитать требуемые характеристики и визуализировать различные графики.

В качестве входного потока Alea использует файл формата SWF [18], в котором для каждого задания указываются следующие основные характеристики:

- идентификатор задания;
- время поступления задания в очередь;
- число процессорных ядер, необходимых для выполнения задания;
- заказанное время выполнения задания;
- реальное время выполнения задания, которое меньше или равно заказанному времени выполнения;
- идентификатор пользователя.

Все эти характеристики доступны алгоритму планирования, но при этом их использование не обязательно. Так, встроенные в Alea алгоритмы планирования игнорируют заказанное время выполнения задания, а ориентируются только на реальное время.

Основным результатом моделирования являются рассчитанное для каждого задания время ожидания в очереди и, соответственно, время запуска задания на вычислителе.

Архитектура симулятора Elytra представлена на рисунке 1. Модуль ожидания заданий входного потока позволяет реализовать динамический входной поток заданий, то есть добавление и удаление заданий из очереди модели СУЗ во время работы симулятора. Для удобства модуль поддерживает два режима работы: со статическим входным потоком заданий, при котором все задания поступают в начальный момент времени, и с динамическим входным потоком, при котором задания поступают в моменты времени, определённые внешним источником. Модуль доступа к очереди заданий реализует возможности

просмотра текущей загрузки симулятора локальной СУЗ и добавления задания в очередь во время моделирования.

Модуль управления модельным временем позволяет синхронизировать модельное время с внешним миром. С помощью библиотеки SimPy на языке программирования Python нами был реализован способ моделирования, основанный на событийной обработке заданий.

В Elytra нами были реализованы два алгоритма: обычная очередь (FCFS) и алгоритм обратного заполнения (backfill).

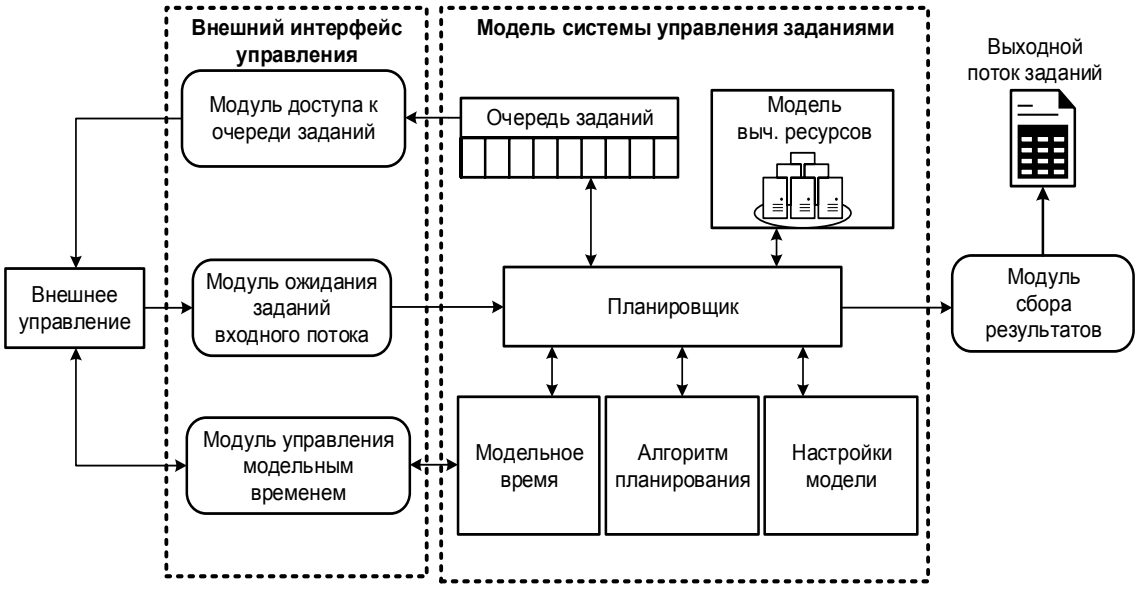

Рис. 1. Архитектура симулятора Elytra

Архитектура симулятора Elytra позволяет реализовать модель TPC с единым модельным временем, с доступом к очереди заданий каждого из симуляторов локальной СУЗ и обменом заданиями между симуляторами за счёт возможности добавления или удаления заданий в процессе моделирования.

4. Экспериментальное исследование симулятора Elytra

Разработанный симулятор после формирования выходного потока заданий рассчитывает ряд характеристик, таких как среднее время ожидания заданий в очереди, среднюю загрузку вычислительных ресурсов и среднюю длину очереди. Для каждой из этих характеристик симулятор формирует график её изменения по времени. Кроме того, симулятор формирует HTML-страницу с планом запуска заданий.

На рисунке 2 представлен пример плана запуска для 10 заданий. По вертикали указаны вычислительные ресурсы, по горизонтали – время от начала эксперимента. Цветные прямоугольники соответствуют заданию, и по клику доступна информация об идентификаторе задания, временах поступления в очередь, старта, завершения, реальном времени выполнения задания и заказанных вычислительных ресурсах. Внутри прямоугольника указывается номер задания. В настройках возможно отключать отображение номеров заданий, что актуально для плана запуска с большим числом заданий.

Для проверки работоспособности и характеристик симулятора Elytra был сгенерирован входной поток из 5000 заданий средней интенсивности на основе подхода, рассмотренного в статье [19]. В этом входном потоке задания поступают в течение 100 часов (4-х с небольшим дней). Моделировалось планирование заданий с помощью реализованных алгоритмов FCFS и Backfill.

На рисунке 3 представлено расписание запусков заданий для исследуемого входного потока заданий. Видно, что в первый день после начала эксперимента в плане запуска множество свободных мест (окон). Это так называемый недогруженный режим работы, во время которого заданий недостаточно для заполнения всех вычислительных ресурсов. В начале 5-го дня задания во входном потоке заканчиваются, после чего в плане запуска снова начинают образовываться окна, связанные с недостатком заданий для заполнения всех ресурсов.

Рис. 2. Пример расписания запусков 10 заданий

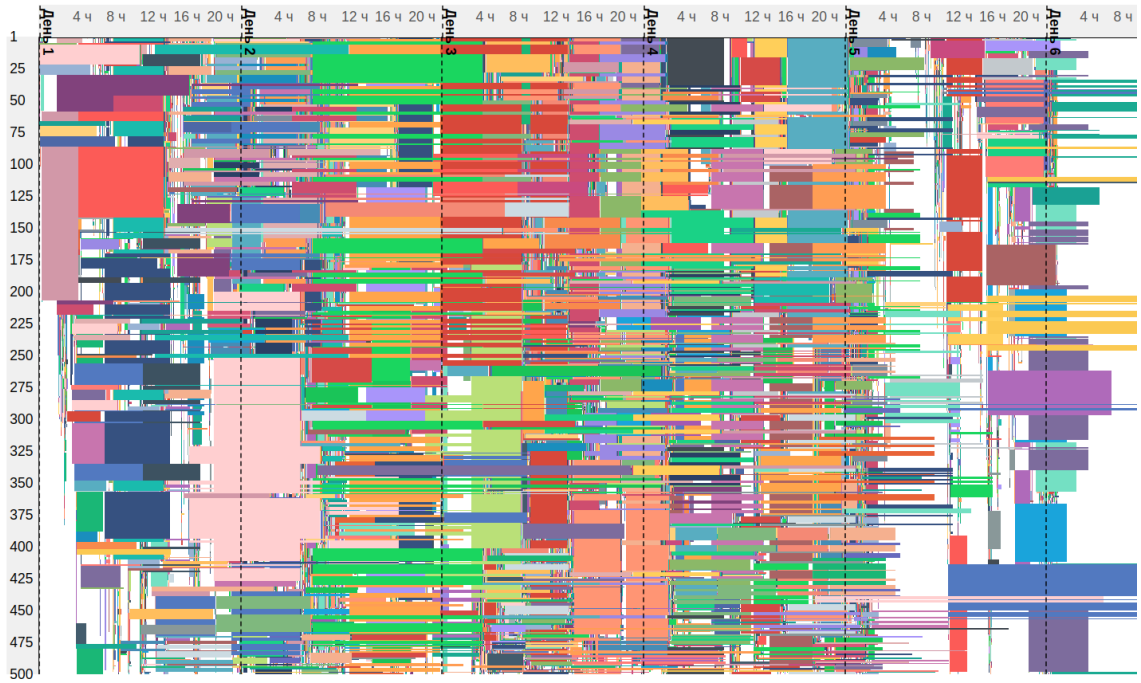

Рис. 3. Расписание запусков заданий для входного потока из 5000 заданий

На рисунке 4 представлен график загрузки вычислительных ресурсов во время эксперимента. В первый день заданий недостаточно для полной загрузки вычислителя. Дни со 2-го по 4-й соответствуют устоявшемуся режиму работы симулятора, когда вычислительные ресурсы интенсивно используются для обработки потока заданий. Видно, что алгоритм backfill обеспечивает бóльшую загрузку вычислителя.

На рисунке 5 представлен график числа заданий в очереди. В 1-й день очереди практически нет, далее очередь некоторое время нарастает. В начале 5-го дня, когда входной поток завершается, очередь быстро пустеет. Видно, что для алгоритма backfill очередь заданий меньше, чем для алгоритма FCFS.

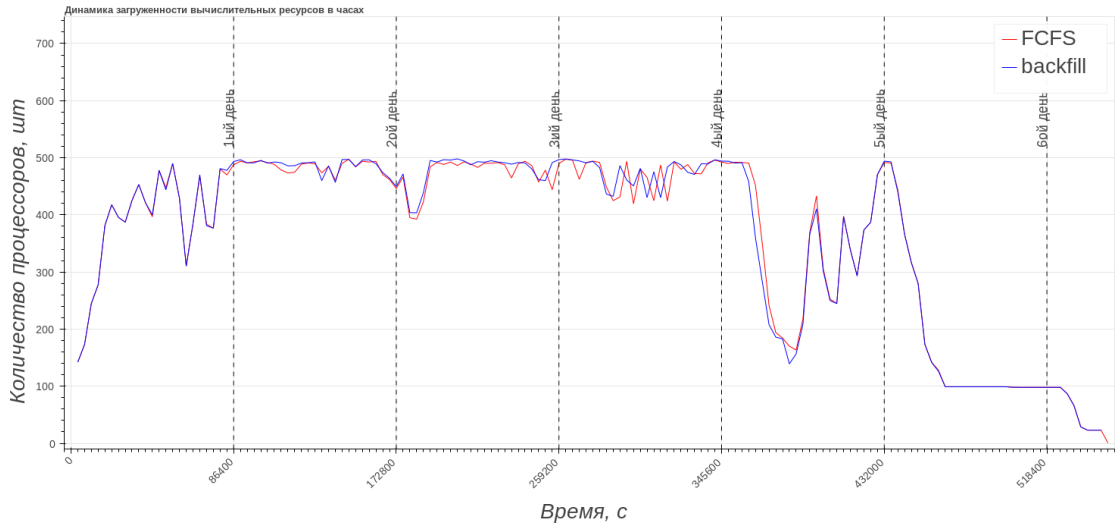

Рис. 4. График загрузки вычислительных ресурсов в Elytra для двух алгоритмов FCFS и backfill

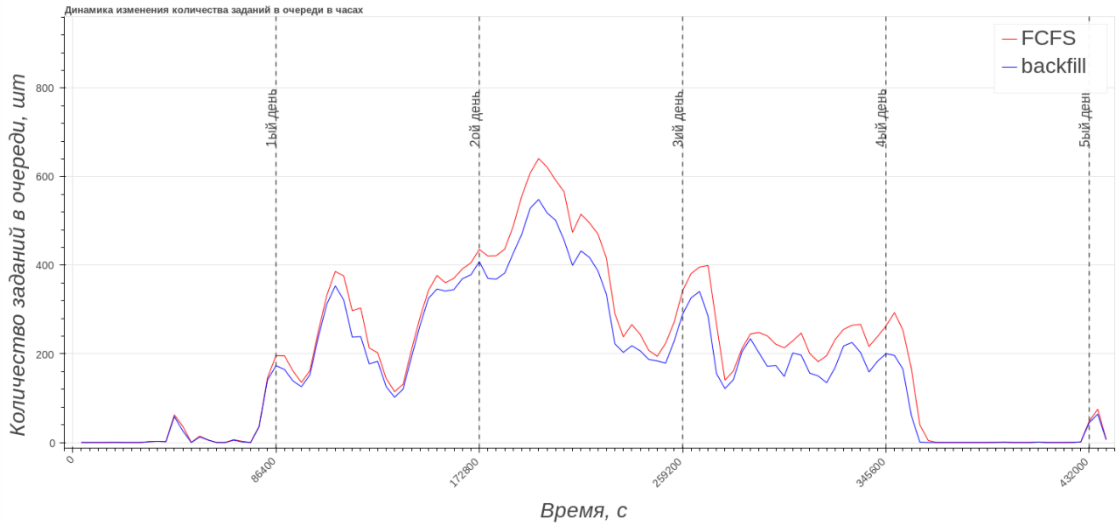

Рис. 5. График числа заданий в очереди Elytra для двух алгоритмов FCFS и backfill

В таблице 1 представлены некоторые из рассчитываемых симулятором характеристик заданий.

Таблица 1. Характеристики выходного потока заданий эксперимента с симулятором Elytra

Характеристика/ Алгоритм	**FCFS**	**Backfill**
Средняя загрузка вычислителя, %	95,2	96,0
Среднее время ожидания задания в очереди, часов	14,8	12,5
Среднее число заданий в очереди	196,1	165,1

Была проверена работоспособность внешнего интерфейса управления. Во время эксперимента симулятор Elytra реагирует на внешнее управление временем, обеспечивает доступ к состоянию очереди заданий, перепланирует задания при добавлении нового задания или удаления из очереди старого, но не запущенного задания. Мы исследовали вопрос снятия с выполнения уже запущенного задания, но такое изменение повлечёт за собой изменение реального времени обработки задания.

Работа симулятора по обработке входного потока проводится быстро, недели вычислительного эксперимента проводятся за секунды. В таблице 2 представлены результаты замера времени работы симулятора для входных потоков заданий от 1000 до 5000.

Таблица 2. Время проведения эксперимента в секундах с симулятором Elytra в зависимости от размера входного потока заданий

Число заданий	FCFS	Backfill
1000	0,8	1,4
2000	1,4	3,9
3000	2,1	8,3
4000	2,6	12,5
5000	3,1	18,7

5. Заключение

В качестве инструмента имитационного моделирования высокопроизводительной вычислительной системы, включающей в свой состав территориально распределенные вычислительный установки (суперкомпьютеры), должен использоваться программный симулятор, отвечающий ряду сформулированных в статье требований. В частности, симулятор должен поддерживать работу с динамическим входным потоком заданий, обеспечивать доступ к локальным очередям заданий и синхронизацию модельного времени между всеми моделируемыми суперкомпьютерами и внешним источником. Выполнение указанных требований возможно за счет реализации некоторого протокола, названного внешним интерфейсом управления симулятора системы управления суперкомпьютерными заданиями (СУЗ).

Анализ существующих средств моделирования – симуляторов СУЗ – показал, что среди них отсутствуют инструменты, удовлетворяющие выдвинутым требованиям. На основе опыта применения распространенного симулятора Alea была предложена архитектура нового симулятора системы управления заданиями, получившего название Elytra. В симуляторе Elytra реализован внешний интерфейс управления, удовлетворяющий сформулированным требованиям. Проведенные эксперименты продемонстрировали работоспособность Elytra, что позволяет сделать вывод о возможности построения на его основе адекватной модели территориально распределенной высокопроизводительной вычислительной системы.

Работа была выполнена в рамках государственного задания по теме FNEF-2024-0016.

Supercomputer Job Management System Simulator with External Control Interface

D. Lyakhovets, A. Baranov, A. Kudrin

Abstract. Simulators are popular tools for studying the supercomputer workload managers as complex multi-user systems. The paper formulates requirements for a simulator of a HPC system that includes geographically distributed supercomputers. Compliance with the stated requirements can be ensured by implementing an external simulator control interface. An analysis of the characteristics of modern HPC workload manager simulators is presented from the stated requirements point of view. The architecture of a workload manager simulator with external control interface is proposed. The first results of using the Elytra simulator, which implements the proposed architecture, are considered.

Keywords: high performance computing, simulating, job scheduling, workload manager

Литература

1. А.В. Баранов, А.И. Тихомиров. Методы и средства организации глобальной очереди заданий в территориально распределенной вычислительной системе. «Вестник ЮУрГУ. Серия: Вычислительная математика и информатика», Т. 6 (2017), № 4, 28-42.

2. А.Г. Феоктистов, А.С. Корсуков, Ю.А. Дядькин. Инструментальные средства имитационного моделирования предметно-ориентированных распределенных вычислительных систем. «Системы управления, связи и безопасности», № 4 (2016), 30–60.

3. D. Cameron, R. Carvajal-Schiano, A. Millar, C. Nicholson, K. Stockinger, F. Zini. OptorSim: A simulation tool for scheduling and replica optimisation in data grids. “Computing in High Energy and Nuclear Physics”, 2010, 707-711.

4. S. Bąk, M. Krystek, K. Kurowski, A. Oleksiak, W. Piatek, J. Waglarz. GSSIM - A tool for distributed computing experiments. “Scientific Programming”, V. 19 (2017), 231-251.

5. W. Chen, E. Deelman. WorkflowSim: A toolkit for simulating scientific workflows in distributed environments. “2012 IEEE 8th International Conference on E-Science, e-Science 2012”, 2012, 1-8.

6. S. Ostermann, K. Plankensteiner, R. Prodan, T. Fahringer. GroudSim: An Event-Based Simulation Framework for Computational Grids and Clouds. “Euro-Par 2010 Parallel Processing Workshops. Euro-Par 2010. Lecture Notes in Computer Science”, V. 6586 (2011), 305–313.

7. P.-F. Dutot, M. Mercier, M. Poquet, O. Richard. Batsim: A Realistic Language-Independent Resources and Jobs Management Systems Simulator. “Job Scheduling Strategies for Parallel Processing. Lecture Notes in Computer Science”, V. 10353 (2017), 178-197.

8. M. Obaida, J. Liu. Simulation of HPC job scheduling and large-scale parallel workloads. “2017 Winter Simulation Conference (WSC)”, 2017, 920-931.

9. D. Klusáček, M. Soysal, F. Suter. Alea – Complex Job Scheduling Simulator. “Parallel Processing and Applied Mathematics. PPAM 2019. Lecture Notes in Computer Science”, V. 12044 (2020), 217-229.

10. N. Capit, G. Da Costa, Y. Georgiou, G. Huard, C. Martin, G. Mounié, P. Neyron, O. Richard. A batch scheduler with high level components. “CCGrid 2005. IEEE International Symposium on Cluster Computing and the Grid”, V. 2 (2005), 776-783.

11. D. Klusáček, M. Soysal. Walltime Prediction and Its Impact on Job Scheduling Performance and Predictability. “Job Scheduling Strategies for Parallel Processing. JSSPP 2020. Lecture Notes in Computer Science”, V. 12326 (2020), 127-144.

12. V. Chlumský, D. Klusáček. Improving Accuracy of Walltime Estimates in PBS Professional Using Soft Walltimes. “Job Scheduling Strategies for Parallel Processing. JSSPP 2022. Lecture Notes in Computer Science”, V. 13592 (2023), 192-210.

13. D. Lyakhovets, A. Baranov. Efficiency Thresholds of Group Based Job Scheduling in HPC Systems. “Lobachevskii Journal of Mathematics”, V. 43 (2023), 2863-2876.

14. M. Jaros, D. Klusáček, J. Jaros. Optimizing Biomedical Ultrasound Workflow Scheduling Using Cluster Simulations. “Job Scheduling Strategies for Parallel Processing. JSSPP 2020. Lecture Notes in Computer Science”, V. 12326 (2020), 68-84.

15. A. Baranov, D. Lyakhovets. Accuracy Comparison of Various Supercomputer Job Management System Models. “Accuracy Comparison of Various Supercomputer Job Management System Models”, V. 42 (2021), 2510–2519.

16. G. I. Savin, B. M. Shabanov, P. N. Telegin, and A. V. Baranov, “Joint Supercomputer center of the Russian Academy of Sciences: Present and future,” Lobachevskii J. Math. 40 (2019). 1853–1862.

17. А.В. Баранов, Д.С. Ляховец. Имитационная модель системы пакетирования суперкомпьютерных заданий на базе симулятора Alea. «Программные продукты и системы», №4 (2022), 631-643.

18. W. Cirne and F. Berman, “A model for moldable supercomputer jobs,” in Proceedings of the 15th International Parallel and Distributed Processing Symposium IPDPS 2001 (2001), p. 8.

19. D. Lyakhovets, A. Baranov, P. Telegin. Scale Ratio Tuning of Group Based Job Scheduling in HPC Systems. “Lobachevskii Journal of Mathematics”, V. 44 (2024), 5012-5026.