ISSN 2079-3316 ПРОГРАММНЫЕ СИСТЕМЫ: ТЕОРИЯ И ПРИЛОЖЕНИЯ №1(32), 2017, с. 193–208

УДК 004.457+004.031.2+004.382.2

А. В. Баранов, Д. С. Ляховец

Влияние пакетирования на эффективность планирования параллельных заданий

Аннотация. В статье рассматривается разработанная авторами система пакетирования параллельных заданий, позволяющая объединять однотипные задания с длительным временем инициализации в пакеты. Длительная инициализация влечёт за собой снижение эффективности использования вычислительных ресурсов и планирования параллельных заданий.

В статье приводятся результаты экспериментов по исследованию влияния пакетирования на такие показатели эффективности планирования параллельных заданий, как полная и полезная загрузка вычислительных ресурсов.

Ключевые слова и фразы: пакетирование заданий, система пакетной обработки, СУППЗ, симулятор, эффективность планирования.

Введение

В настоящей работе под системой высокопроизводительных вычислений будем понимать вычислительную установку типовой кластерной архитектуры, состоящую из нескольких вычислительных модулей, объединенных одной или несколькими высокоскоростными сетями. Вычислительный модуль (ВМ) такой системы представляет собой самостоятельный компьютер, оснащенный, как правило, несколькими центральными процессорами (двумя или более), собственными оперативной и дисковой памятями. Очень часто вычислительные модули имеют в своем составе сопроцессоры — ускорители на базе графических процессоров, ПЛИС или многоядерных мультитредовых решений (Intel Xeon Phi).

Управление вычислительными ресурсами и пользовательскими заданиями в современных системах высокопроизводительных вычислений осуществляет специальное программное обеспечение — система

Работа выполнена при поддержке гранта РФФИ №16-07-01098..

управления заданиями (СУЗ). СУЗ обеспечивает коллективный доступ пользователей к супер-ЭВМ, принимает входной поток различных заданий от разных пользователей, планирует очереди заданий, выделяет необходимые для выполнения задания вычислительные ресурсы и освобождает их после завершения задания.

Жизненный цикл задания включает следующие этапы:

- нахождение задания в очереди;
- инициализация задания;
- обработка задания;
- завершение обработки и сохранение результатов.

Выделим несколько категорий заданий, снижающих эффективность использования вычислительных ресурсов из-за длительного времени инициализации:

(*1*) Для краткосрочного задания требуется инициализация высокопроизводительного ускорителя (ГПУ или ПЛИС), время инициализации ускорителя сравнимо со временем обработки задания [**1**].
(*2*) Для задания необходима специфическая виртуальная программная платформа, развёртывание такой платформы перед стартом задания может занять существенное время [**2**, **3**].
(*3*) Для задания необходимо до начала обработки загрузить на вычислительные узлы значительный объём входных данных.

Задания перечисленных категорий объединяет длительное относительно остальных этапов обработки время инициализации, которое может включать в себя время перепрограммирования ПЛИС, компиляции программы на ГПУ, запуска виртуальных машин, подготовки контейнеров, копирования данных.

Обратим внимание на тот факт, что биллинговая подсистема СУЗ относит инициализацию к полезной загрузке, завышая реальную утилизацию ресурсов. Более того, при инициализации могут быть задействованы ЦПУ, оперативная память и сеть, так что даже интеллектуальная система мониторинга не способна достоверно отличить инициализацию задания от процесса обработки. В то же время потери времени на инициализацию равнозначны потери эффективности использования ресурсов.

Идея пакетирования состоит в том, что во входном потоке можно выделить задания, для инициализации которых необходимо совершить одинаковые действия. Будем называть задания с одной и той же

процедурой инициализации заданиями одинакового типа. Доля затраченного на инициализацию времени может быть уменьшена за счёт однократной инициализации перед выполнением нескольких заданий одного типа, объединяемых в пакет (метазадание).

Интуитивное представление подсказывает, что пакетирование должно положительно повлиять на эффективность. Однако при планировании в современных СУЗ, таких как СУППЗ [4], SLURM, Moab, используются достаточно сложные алгоритмы (например, обратное заполнение), что не позволяет однозначно спрогнозировать влияние пакетирования на эффективность планирования заданий. Например, в работе [5] рассмотрен квазипланировщик, поставляющий в очередь СУППЗ специальным образом подобранные задания. Подбор заданий осуществлялся с учётом особенностей работы алгоритма обратного заполнения, и использование квазипланировщика, по интуитивным предположениям авторов, должно было положительно повлиять на загрузку вычислителя. Однако, результаты проведённых экспериментов [5] не позволили выявить сколь-либо существенного влияния квазипланировщика на показатели эффективности СУППЗ.

Целью настоящей работы является экспериментальное исследование влияния механизма пакетирования на показатели эффективности СУЗ, в качестве которой в экспериментах выступала СУППЗ. Статья посвящена начальному этапу работы, на котором рассматривается несколько упрощённая модель входного потока заданий, каждое из которых является идеально масштабируемым. Для идеально масштабируемых заданий время их выполнения уменьшается прямо пропорционально объёму вычислительных ресурсов, выделенных СУЗ для задания. Например, если время обработки задания на 1 ВМ составляет 30 мин., тогда на 2 ВМ составит 15 мин., на 30 ВМ — 1 мин.

Время инициализации для идеально масштабируемых заданий не масштабируется и является константой для задания. Так, если время инициализации составляет 10 мин., время обработки задания на 1 ВМ — 20 мин. (суммарное время выполнения 30 мин.), то на 2 ВМ время выполнения составит $10 + (20/2) = 20$ мин., на 30 ВМ составит $10 + (20/30) = 10,67$ мин. и т.д.

1. Показатели эффективности планирования заданий

Пусть изменения в системе (добавление заданий, изменение количества доступных вычислительных модулей и др.) происходят в моменты времени t_i. Разобьём всё время работы системы на промежутки времени $\Delta t_i = t_{i+1} - t_i$.

Под модуль-часом будем понимать вычислительный ресурс, эквивалентный 1 вычислительному модулю, доступному для обработки задания в течение 1 часа. 1 модуль-час может означать, что 1 ВМ работал 1 час или 2 ВМ работали по 0,5 часов. Аналогично вводятся модуль-секунды и модуль-минуты.

Пусть n_i — число работоспособных вычислительных модулей в i-й момент времени, m_i — число занятых выполнением заданий вычислительных модулей в i-й момент времени. К выполнению задания относится инициализация и обработка задания. Пусть также u_i — число занятых обработкой заданий вычислительных модулей в i-ый момент времени. Занятые инициализацией задания модули не учитываются в u_i.

Определим как S_{all} число модуль-часов, доступных для выполнения заданий за определённый временной период:

$$S_{all}(t) = \sum_{i=0}^{t} n_i \Delta t_i.$$

Определим как S_{work} количество модуль-часов, использованных для выполнения заданий за тот же период:

$$S_{work}(t) = \sum_{i=0}^{t} m_i \Delta t_i, \qquad \text{где } m_i \in (0, n_i),$$

а как S_{useful} количество модуль-часов, использованных для обработки заданий за тот же период (без учёта инициализирующихся модулей):

$$S_{useful(t)} = \sum_{i=0}^{t} u_i \Delta t_i, \qquad \text{где } u_i \in (0, n_i).$$

Пусть R_i — заказанное время выполнения i-го задания, Q_i — время ожидания i-го задания в очереди с момента постановки в очередь до начала его выполнения, E_i — время обработки i-го задания.

Приведём ряд известных показателей эффективности.

(1) Полная загрузка (или просто загрузка) вычислительных ресурсов L за определённый период времени или утилизация, которая может быть определена как

$$L(t) = \frac{S_{work}(t)}{S_{all}(t)}.$$

(2) Полезная загрузка вычислительных ресурсов U за определённый период времени, которая может быть определена как

$$U(t) = \frac{S_{useful}(t)}{S_{all}(t)}.$$

(3) Среднее время ожидания задания в очереди Q. Для k заданий этот показатель может быть рассчитан как

$$Q = \frac{\sum_{i=0}^{k} Q_i}{k}.$$

(4) Среднее приведённое время ожидания задания в очереди относительно заказанного времени его счёта K. Для k заданий формула будет иметь вид

$$K = \frac{\sum_{i=0}^{k} \frac{Q_i}{R_i}}{k}.$$

Среднее приведённое время адекватно отражает качество планирования заданий. Например, ожидание в очереди в течение 1 часа вполне приемлемо для задания с заказанным временем счёта в 24 часа, но тот же час ожидания для задания с заказанным временем счёта в 10 минут может являться показателем плохого качества обслуживания.

(5) Дисперсия приведённого времени ожидания D, которая характеризует разброс скорости обслуживания для различных заданий, то есть, фактически, «справедливость» обслуживания. Большая дисперсия свидетельствует о неравномерности качества обслуживания различных заданий при одном и том же среднем приведённом времени ожидания заданий в очереди.

(6) Среднее время полного обслуживания задания F. Для k заданий формула имеет вид:

$$F = \frac{\sum_{i=0}^{k} (Q_i + E_i)}{k}.$$

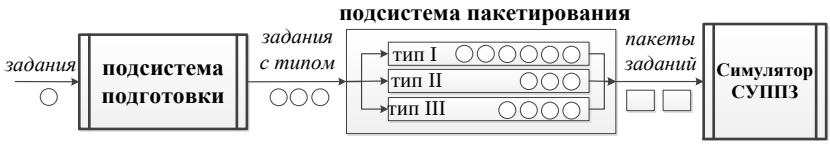

Рис. 1. Состав экспериментального стенда

Как уже упоминалось, настоящая статья описывает начальный этап экспериментальной работы, на котором авторы исследовали влияние пакетирования идеально масштабируемых заданий на показатели загрузки L и полезной загрузки U.

2. Состав экспериментального стенда и подсистема пакетирования заданий

Основу экспериментального стенда составил разработанный авторами симулятор СУППЗ, который позволяет эмулировать запуск параллельных заданий на обычном персональном компьютере. Симуляторы часто используются при исследовании различных СУЗ [6, 7] и позволяют эмулировать высокопроизводительные системы, состоящие из сотен и тысяч вычислительных модулей.

Для реализации возможности пакетирования авторами были разработаны и сопряжены с СУППЗ две дополнительные подсистемы (рис. 1). Подсистема подготовки определяет тип задания и, при необходимости, может менять формат представления задания. Подсистема пакетирования ведёт очереди по числу типов заданий, определяет очередь для формирования пакета заданий, формирует пакет заданий и передаёт сформированные пакеты заданий в СУППЗ для обработки. В момент формирования пакета заданий определяется число ВМ, выделенных для его обработки. Когда в СУППЗ есть доступные ресурсы, система пакетирования формирует из заданий одного типа пакет и запускает его на свободных ресурсах.

Заметим, что обе разработанных подсистемы универсальны и могут быть сопряжены с любой СУЗ.

Пусть в формировании пакета участвуют n идеально масштабируемых заданий с временами выполнения на 1 ВМ $e_1, e_2, \ldots, e_n$ и временем инициализации $t_{\text{иниц}}$.Тогда общее время выполнения на 1 ВМ пакета из n заданий составит некоторую величину $e^1_{\text{пакета}}$:

$$e^1_{\text{пакета}} = t_{\text{иниц}} + \sum_{i=1}^{n} e_i.$$

Соответственно, время $e^m_{\text{пакета}}$ выполнения пакета на m ВМ составит:

$$e^m_{\text{пакета}} = t_{\text{иниц}} + \sum_{i=1}^{n} \frac{e_i}{m}.$$

При формировании пакета система пакетирования должна принять два решения: какое число n заданий войдёт в очередной пакет, и какое число m вычислительных модулей следует выделить для обработки пакета. Главным параметром при принятии этих решений является т.н. порог целесообразности k, определяющий отношение времени обработки пакета заданий ко времени инициализации пакета. Другими словами, порог целесообразности k говорит о том, что время $e^m_{\text{пакета}}$ обработки пакета на m ВМ должно не менее чем в k раз превышать время инициализации пакета $t_{\text{иниц}}$:

$$e^m_{\text{пакета}} > k \cdot t_{\text{иниц}}.$$

Пусть в некоторый момент времени в системе присутствуют $n-1$ заданий определённого типа, наличествует m свободных ВМ, и на вход поступает следующее задание n того же типа. Подсистема пакетирования примет положительное решение о формировании пакета и его направлении на обработку на m ВМ, если выполняется условие

$$m < \frac{\sum_{i=1}^{n} e_i}{t_{\text{иниц}} \cdot k},$$

и в СУЗ наличествует $m_{\text{своб}} \geqslant m$ свободных ВМ. Если $m_{\text{своб}} < m$, то пакет заданий будет направлен на обработку на $m_{\text{своб}}$ ВМ.

Чем больше порог целесообразности, тем на меньших вычислительных ресурсах будет обрабатываться сформированный пакет заданий.

3. Структура и параметры тестового входного потока заданий для проведения экспериментов

Для проведения экспериментов необходимо генерировать тестовый входной поток заданий. Пусть $T = (T_1, \ldots, T_n)$ — упорядоченная по времени поступления в очередь последовательность заданий. Параметры задания T_i:

- время поступления задания в очередь t_i;
- тип задания (задания одного типа имеют одинаковые процедуры инициализации и могут объединяться в пакеты);
- время инициализации задания $t_{\text{иниц}}$;

- требуемое число вычислительных модулей для обработки задания m_i;
- время обработки задания e_i.

Выделим несколько особенностей тестового потока заданий. По примеру работы [8], в которой при исследовании различных алгоритмов планирования параметры входного потока заданий были построены на базе статистики работы кластера Оренбургского государственного университета, будем считать, что интервалы между временами t_i поступления заданий имеют распределение Пуассона с интенсивностью λ. Интенсивность λ в экспериментах подбиралась таким образом, чтобы обеспечить загрузку вычислительных ресурсов на уровне 95%–100% и постоянное наличие заданий в очереди. Меньшая загрузка нецелесообразна, поскольку ресурсы заведомо будут простаивать, большая загрузка приведёт к бесконечной очереди заданий.

Требуемое количество ВМ для задания m_i в экспериментах принималось равным единице, поскольку для идеально масштабируемых заданий при выполнении на одном ВМ достигается наименьшая доля времени инициализации $t_{\text{иниц}}$ относительно времени обработки задания e_i. Чем меньше доля времени инициализации, тем меньше выгода от применения пакетирования, поэтому в случае $m_i = 1$ подсистема пакетирования попадает в наиболее неблагоприятную для себя, и, тем самым, наиболее интересную для исследования ситуацию.

Время инициализации задания $t_{\text{иниц}}$ в проводимых авторами экспериментах было определено как параметр, и в зависимости от эксперимента составляло 10%, 25% и 100% от среднего времени выполнения задания. В свою очередь, выбор вида распределения времени выполнения заданий был основан на работе [9], в которой строится математическая модель вычислительной загрузки кластера на базе анализа статистики работы трёх вычислительных установок — Суперкомпьютерного центра в Сан-Диего (416 вычислительных модуля, данные по 113 515 заданиям за два года), Лос-Аламосской национальной лаборатории (1024 вычислительных модуля, 36 306 заданий за полгода), Королевского технологического института в Стокгольме (100 вычислительных модулей, 16 221 заданий за год). Для моделирования времени выполнения заданий e_i подходит гамма-распределение с параметрами α_e и β_e.

В целях оптимизации времени проведения экспериментов, а также с учётом того, что исследуется зависимость показателей эффективности от доли времени инициализации заданий (а не от абсолютных

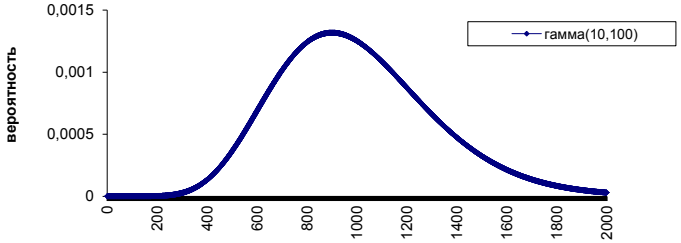

Рис. 2. Распределение времени обработки заданий

Таблица 1. Влияние доли инициализации на загрузку СУППЗ

Доля времени инициализации задания	Полная загрузка	Полезная загрузка
10%	0,971	0,862
25%	1,000	0,752
100%	1,000	0,481

значений времени), среднее время выполнения заданий было выбрано в 1000 секунд (заведомо больше времени планирования, не превышающем 1 с), при этом большинство заданий выполняются за время от 500 до 1500 секунд (рис. 2). Такой поток заданий может быть описан параметрами гамма-распределения $\alpha_e = 10$ и $\beta_e = 100$.

В зависимости от вида эксперимента, входной поток заданий поступал либо в очередь СУППЗ, либо в очередь подсистемы пакетирования.

4. Условия проведения и результаты экспериментов

Для проведения экспериментов однократно был сгенерирован тестовый поток из 7000 заданий, который затем использовался в качестве входного потока во всех экспериментах. Расчёт загрузки вычислителя осуществлялся без учёта начального и конечного отрезков времени проведения эксперимента, когда заданий для полной загрузки вычислителя либо ещё недостаточно, либо уже недостаточно.

В первой серии экспериментов исследовалось влияние доли времени инициализации относительно среднего времени обработки задания. Среднее время обработки задания во всех экспериментах составляет

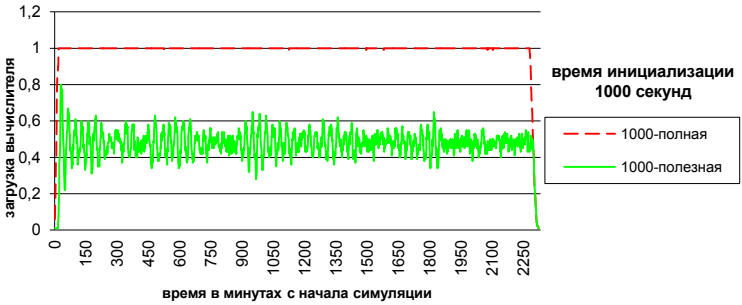

Рис. 3. Сравнение полной и полезной загрузки для СУППЗ без использования подсистемы пакетирования

1000 секунд. Доля времени инициализации для различных экспериментов составляет 10%, 25% и 100%. Таблица 1 содержит результаты эксперимента, во время которого задания поступали напрямую на вход СУППЗ без использования подсистемы пакетирования.

Из таблицы 1 видно, что в ходе проведения всех экспериментов полная загрузка составляет 100% или почти 100%. Это следствие интенсивного потока заданий, требующих для своего выполнения только один ВМ. Планировщик СУППЗ способен плотно, без пропусков сформировать план запуска заданий.

С полезной загрузкой ситуация иная. Для наименьшей доли времени инициализации полезная загрузка составляет уже 86%. С ростом доли времени инициализации падает полезная загрузка, вплоть до менее 50% при равном среднем времени обработки и времени инициализации.

Рассмотрим характерную зависимость загрузки вычислителя (полной и полезной на одном графике) от времени проведения экспериментов (рис. 3) для доли времени инициализации 100%. Полная загрузка составляет 100% и совпадает с горизонтальной линией. Полезная загрузка же колеблется около 50%.

Далее будем рассматривать только полезную загрузку.

Сравним СУППЗ с системой пакетирования на одном и том же входном потоке заданий. Для системы пакетирования будем указывать число различных типов заданий во входном потоке (1, 2 или 8) и порог целесообразности формирования пакетов (1, 2, 10, 100).

Таблица 2. Зависимость полезной загрузки от времени инициализации задания при 8 типах заданий и пороге целесообразности 1

Доля времени инициализации задания	Полезная загрузка	
	Без пакетирования	С пакетированием
10%	0,862	0,989
25%	0,752	0,968
100%	0,481	0,894

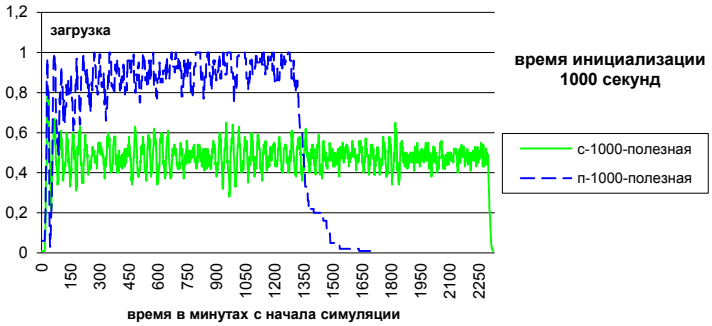

Рис. 4. Сравнение полезной загрузки для СУППЗ без использования и с использованием подсистемы пакетирования

Сравним СУППЗ без использования подсистемы пакетирования и с её применением для 8 типов заданий и порогом целесообразности, равным 1 (таблица 2).

Пакетирование позволяет сохранить полезную загрузку вычислителя на высоком уровне. Сравним график полезной загрузки для СУППЗ и пакетирования при доле времени инициализации 100% (рис. 4). Около 50% колеблется полезная загрузка для СУППЗ. Для пакетирования видно, что повышение полезной загрузки в начале вычислений позволяет раньше завершить обработку заданий. После 1350 минуты также видна особенность пакетирования — комплектование заданий в пакеты на часть свободных ресурсов, в результате чего часть вычислительных ресурсов простаивает. Но этот момент наступает после завершения поступления заданий в очередь, когда система начинает работать в недогруженном режиме.

Таблица 3. Зависимость полезной нагрузки от числа типов заданий при пороге целесообразности 1 и доле времени инициализации 100%

Эксперимент	Полезная загрузка	Выигрыш от пакетирования, %
Без пакетирования	0,481	0
Пакетирование:		
с 8 типами заданий	0,894	86
с 2 типами заданий	0,874	82
с 1 типом заданий	0,853	77

Таблица 4. Зависимость полезной нагрузки от величины порога целесообразности при одном типе заданий и доле времени инициализации 100%

Эксперимент	Полезная загрузка	Выигрыш от пакетирования, %
Без пакетирования	0,481	0
Пакетирование:		
порог 1	0,853	77
порог 2	0,843	75
порог 10	0,855	78
порог 100	0,858	78

Следующая часть экспериментов была посвящена исследованию зависимости полезной загрузки от числа типов заданий. Предполагалось, что с ростом числа типов заданий загрузка должна уменьшаться. Однако, результаты (таблица 3) не позволили выявить сколь-либо существенного влияния числа типов заданий на величину полезной загрузки. Таблица 3 содержит результаты только для доли времени инициализации 100%, однако аналогичные данные были получены и для 10% и 25%.

В ходе экспериментов не было выявлено влияния значения порога целесообразности на полезную загрузку вычислителя (таблица 4). Для доли времени инициализации 10%, 25% и 100% (для краткости приведены данные только для 100%) времени инициализации и одного типа задания был задан различный порог целесообразности (1, 2, 10, 100).

По собранным статистическим данным, при доле инициализации в 10% повышение полезной загрузки в результате пакетирования составляет от 5% до 15%, при доле инициализации в 25% — от 20 до 30%, при доле в 100% — от 75% до 90%.

Заключение

Авторами работы была разработана подсистема пакетирования заданий, которая может быть сопряжена с любой системой управления заданиями (СУЗ). Разработанная подсистема, по мнению авторов, может быть применена в универсальных высокопроизводительных системах коллективного пользования при обработке заданий, представленных в виде набора виртуальных машин или контейнеров. В этом случае старт задания будет предваряться достаточно длительной процедурой развёртывания виртуальных машин или контейнеров из заранее подготовленных образов, хранимых в специально выделенной области файловой системы. Предполагается, что пользователь, желающий развернуть для своего задания определённую виртуальную программную платформу, будет указывать при постановке задания в очередь конкретный размещенный в хранилище подготовленный образ, фактически определяя этим тип задания для подсистемы пакетирования.

В ходе экспериментального исследования на симуляторе СУППЗ выявлен существенный разрыв между полной и полезной загрузкой вычислительных ресурсов в случае обработки потока идеально масштабируемых заданий. С ростом доли времени, затрачиваемого на инициализацию вычислительных ресурсов, уменьшается полезная загрузка вычислителя.

Применение пакетирования позволяет существенно увеличить полезную загрузку вычислителя. При доле времени инициализации в 10% полезная загрузка за счёт пакетирования возросла с 86% до 99%, при доле в 100% — возросла с 48% до 89%. Влияния других параметров (числа типов заданий и величины порога целесообразности) на значение полезной загрузки выявлено не было. В связи с этим авторы планируют продолжить экспериментальные исследования с целью определения влияния параметров подсистемы пакетирования на другие рассмотренные в настоящей работе показатели эффективности планирования заданий. Проведение подобных экспериментов, а также использование в экспериментах более сложной модели входного потока заданий, включающей не только идеально масштабируемые задания, являются ближайшей перспективой работы.

Список литературы

[1] И. И. Левин, А. И. Дордопуло, И. А. Каляев, Ю. И. Доронченко, М. К. Раскладкин. «Современные и перспективные высокопроизводительные вычислительные системы с реконфигурируемой архитектурой», *Вестн. ЮУрГУ. Сер. Выч. матем. информ.*, **4**:3 (2015), с. 24–39. ↑ 194

[2] А. В. Баранов, Д. С. Николаев. «Использование контейнерной виртуализации в организации высокопроизводительных вычислений», *Программные системы: теория и приложения*, **7**:1(28) (2016), с. 117–134, URL: http://psta.psiras.ru/read/psta2016_1_117-134.pdf ↑ 194

[3] О. С. Аладышев, А. В. Баранов, Р. П. Ионин, Е. А. Киселёв, В. А. Орлов. «Сравнительный анализ вариантов развертывания программных платформ для высокопроизводительных вычислений», *Вестник УГАТУ*, **18**:3 (2014), с. 295–300, URL: http://journal.ugatu.ac.ru/index.php/vestnik/article/view/1101 ↑ 194

[4] СУППЗ — Система Управления Прохождением Параллельных Заданий (дата обращения 13.11.2016), URL: http://suppz.jscc.ru/ ↑ 195

[5] А. В. Баранов, Е. А. Киселёв, Д. С. Ляховец. «Квазипланировщик для использования простаивающих вычислительных модулей многопроцессорной вычислительной системы под управлением СУППЗ», *Вестн. ЮУрГУ. Сер. Выч. матем. информ.*, **3**:4 (2014), с. 75–84. ↑ 195

[6] П. Н. Полежаев. «Симулятор вычислительного кластера и его управляющей системы, используемый для исследования алгоритмов планирования задач», *Вестн. ЮУрГУ. Сер. Матем. моделирование и программирование*, 2010, №6, с. 79–90. ↑ 198

[7] SLURM Simulator (data accessed 13.11.2016), URL: http://www.bsc.es/marenostrum-supportservices/services/slurm-simulator ↑ 198

[8] В. П. Гергель, П. Н. Полежаев. «Исследование алгоритмов планирования параллельных задач для кластерных вычислительных систем с помощью симулятора», *Вестник ННГУ*, 2010, №5-1, с. 201–208. ↑ 200

[9] U. Lublin, G. Feitelson. “The Workload on Parallel Supercomputers: Modeling the Characteristics of Rigid Job”, *Journal of Parallel and Distributed Computing Archive*, **63**:11 (2003), pp. 542–546. ↑ 200

Рекомендовал к публикации *Программный комитет*
Пятого национального суперкомпьютерного форума ***НСКФ-2016***

Пример ссылки на эту публикацию:

А. В. Баранов, Д. С. Ляховец. «Влияние пакетирования на эффективность планирования параллельных заданий», *Программные системы: теория и приложения*, 2017, **8**:1(32), с. 193–208.
URL: http://psta.psiras.ru/read/psta2017_1_193-208.pdf

Об авторах:

Антон Викторович Баранов

Ведущий научный сотрудник МСЦ РАН, к.т.н., доцент. Области научных интересов: организация высокопроизводительных вычислений, планирование заданий и управление вычислительными ресурсами в суперкомпьютерах, технологии виртуализации и облачных вычислений

e-mail: antbar@mail.ru

Дмитрий Сергеевич Ляховец

Стажёр-исследователь МСЦ РАН. Области научных интересов: планирование заданий и управление вычислительными ресурсами в суперкомпьютерах, технологии симуляции при проведении экспериментов

e-mail: anetto@inbox.ru

Anton Baranov, Dmitriy Lyakhovets. *The influence of packaging on efficiency of parallel jobs scheduling.*

Abstract. The article considers the system of parallel jobs scheduling designed by the authors. The system allows combining into one package the same type of parallel jobs with the long initialization time. Long-time initialization results into decrease of efficiency of computing resources usage and parallel jobs scheduling.

The article presents the results of the experiments on the research of the influence of packaging on such scheduling efficiency metrics as full and useful load of computing resources. (*In Russian*).

Key words and phrases: jobs packaging system, batch mode, grouping-based scheduling, SUPPZ, simulator, scheduling efficiency.

References

[1] I. I. Levin, A. I. Dordopulo, I. A. Kalyayev, Yu. I. Doronchenko, M. K. Raskladkin. "Modern and Next-Generation High-Performance Computer Systems with Reconfigurable Architecture", *Vestn. YuUrGU. Ser. Vych. matem. inform.*, **4**:3 (2015), pp. 24–39 (in Russian).

[2] A. V. Baranov, D. S. Nikolayev. "The Use of Container Virtualization in the Organization of High-Performance Computing", *Programmnyye sistemy: teoriya i prilozheniya*, **7**:1(28) (2016), pp. 117–134 (in Russian), URL: http://psta.psiras.ru/read/psta2016_1_117-134.pdf

[3] O. S. Aladyshev, A. V. Baranov, R. P. Ionin, Ye. A. Kiselëv, V. A. Orlov. "Comparative Analysis of Variants of Deployment of Program Platforms for High Performance Computing", *Vestnik UGATU*, **18**:3 (2014), pp. 295–300 (in Russian), URL: http://journal.ugatu.ac.ru/index.php/vestnik/article/view/1101

[4] Parallel Job Management System (data obrashcheniya 13.11.2016) (in Russian), URL: http://suppz.jscc.ru/

[5] A. V. Baranov, Ye. A. Kiselëv, D. S. Lyakhovets. "The Quasi Scheduler for Utilization of Multiprocessing Computing System'S Idle Resources Under Control Of The Management System of the Parallel Jobs", *Vestn. YuUrGU. Ser. Vych. matem. inform.*, **3**:4 (2014), pp. 75–84 (in Russian).

[6] P. N. Polezhayev. "Simulator of Computer Cluster and its Management System Used for Research of Job Scheduling Algorithms", *Vestn. YuUrGU. Ser. Matem. modelirovaniye i programmirovaniye*, 2010, no.6, pp. 79–90 (in Russian).

[7] SLURM Simulator (data accessed 13.11.2016), URL: http://www.bsc.es/marenostrum-supportservices/services/slurm-simulator

[8] V. P. Gergel', P. N. Polezhayev. "The Study of Parallel Job Scheduling Algorithms for Cluster Computing Systems Usinag a Simulator", *Vestnik NNGU*, 2010, no.5-1, pp. 201–208 (in Russian).

[9] U. Lublin, G. Feitelson. "The Workload on Parallel Supercomputers: Modeling the Characteristics of Rigid Job", *Journal of Parallel and Distributed Computing Archive*, **63**:11 (2003), pp. 542–546.

Sample citation of this publication:

Anton Baranov, Dmitriy Lyakhovets. "The influence of packaging on efficiency of parallel jobs scheduling", *Program systems: Theory and applications*, 2017, **8**:1(32), pp. 193–208. (*In Russian*).

URL: http://psta.psiras.ru/read/psta2017_1_193-208.pdf