УДК 004.451
DOI: 10.25559/SITITO.15.201901.59-71

Организация вычислительного процесса в суперкомпьютерах с динамически управляемыми разделами

С. В. Назаров*, А. Г. Барсуков
Московский научно-исследовательский телевизионный институт, г. Москва, Россия
* nazarov@mniti.ru

Аннотация

Во всем мире непрерывно возрастает интерес к высокопроизводительным вычислениям. Он обусловлен потребностями различных отраслей науки, образования, медицины, инженерной и другой практической деятельностью человечества. Данная статья посвящена вопросам организации вычислительного процесса при выполнении информационно-связанных задач в компьютерах с динамически управляемыми разделами. Даже при высокой производительности суперкомпьютеров, предназначенные для них задачи, могут занимать значительное время и выполняться, как правило, неоднократно. Программы, реализующие эти задачи, имеют сложную многомодульную структуру с информационными и управляющими связями между ними. Поэтому они реализуются в пакетном режиме и часто в ночное время. В пакеты могут входить различные задачи, требующие высокой вычислительной мощности и большой основной и внешней памяти. В связи с этим даже в рамках мощнейших суперкомпьютеров требуется оптимизация вычислительного процесса с целью рационального расходования вычислительных ресурсов при одновременном требовании выполнения определенного набора задач (пакета) в заданные временные ограничения. Организация вычислительного процесса должна учитывать особенности архитектуры компьютера и его операционной системы. В статье предлагается решение задачи оптимизации вычислительного процесса для серверов и мэйнфреймов IBM на процессорах Power9 и z14 применительно к операционным системам, которые используются на этих компьютерах. Решение предлагается последовательностью связанных этапов. Для решения используются методы сетевого планирования и управления, теория параллельных вычислительных процессов и расписаний. В начале представляется структура приложения и дается постановка задачи. Решение начинается с определения величины критического пути и резервов времени по отдельным вычислительным работам. Далее рассматриваются возможности минимизация длины критического пути. После чего ставится и решается задача распределения количества долей физических процессоров, выделяемых динамическим разделам компьютера, при условии выполнения приложения без увеличения длины критического пути. В заключение разрабатывается временная диаграмма вычислительного процесса с фиксацией итогового результата занятости физических процессоров компьютера и числа необходимых логических разделов.

Ключевые слова: вычислительный процесс, оптимизация, суперкомпьютер, динамический раздел, критический путь, минимизация вычислительных ресурсов.

Для цитирования: *Назаров С. В., Барсуков А. Г.* Организация вычислительного процесса в суперкомпьютерах с динамически управляемыми разделами // Современные информационные технологии и ИТ-образование. 2019. Т. 15, № 1. С. 59-71. DOI: 10.25559/SITITO.15.201901.59-71

Organization of the Computing Process in Supercomputers with Dynamically Managed Partitions

S. V. Nazarov*, **A. G. Barsukov**
Moscow Research TV Institute Joint Stock Company, Moscow, Russia
* nazarov@mniti.ru

Abstract

Worldwide interest in high-performance computing is constantly growing. It is prompted by the needs of various branches of science, education, medicine, engineering and other practical activities of mankind. This article is devoted to the organization of the computational process in the performance of information-related tasks in computers with dynamically controlled cases. Even with the high performance of supercomputers, the tasks assigned to them can take considerable time and can be performed, as a rule, repeatedly. The programs implementing these tasks have a complex multi-module structure with information and control links between them. Therefore, they are implemented in batch mode and often at night. The packages can include various tasks that require high computing power and large main and external memory. In this regard, even within the framework of the most powerful supercomputers, optimization of the computing process is required for the purpose of rational consumption of computing resources while requiring the performance of a certain set of tasks (package) in a given time limit. The organization of the computing process should take into account the features of the computer architecture and its operating system. The article offers a solution to the problem of optimizing the computing process for IBM servers and mainframes on power9 and z14 processors in relation to the operating systems that are used on these computers. The solution is proposed by a sequence of related steps. Methods of network planning and management, theory of parallel computing processes and schedules are used for the solution. In the beginning, the structure of the application is presented and the problem statement is given. The solution begins with determining the critical path and time reserves for individual computational work. The possibilities of minimizing the length of the critical path are further considered. After that, the task of distributing the number of shares of physical processors allocated to dynamic partitions of the computer is set and solved, provided that the application is run without increasing the length of the critical path. In conclusion, we develop a time diagram of the computational process with fixing the final result of the employment of physical processors of the computer and the number of necessary logical partitions.

Keywords: Computational Process, Optimization, Supercomputer, Dynamic Partition, Critical Path, Minimization of Computing Resources.

For citation: Nazarov S.V., Barsukov A.G. Organization of the Computing Process in Supercomputers with Dynamically Managed Partitions. *Sovremennye informacionnye tehnologii i IT-obrazovanie* = Modern Information Technolo-gies and IT-Education. 2019; 15(1): 59-71. DOI: 10.25559/SITITO.15.201901.59-71

1. Архитектуры суперкомпьютеров

На сегодняшний день суперкомпьютеры являются уникальными системами, создаваемыми «традиционными» лидерами компьютерного рынка, такими как IBM, Hewlett–Packard, NEC, Lenovo и другими, которые приобрели множество ранних компаний, вместе с их опытом и технологиями12. Компания Cray Inc. по-прежнему занимает достойное место в ряду производителей суперкомпьютерной техники3.

Существует две основные архитектуры современных суперкомпьютеров – системы с общей памятью и кластеры [1, 2]. Каждый подход не исключает другого и имеет свои достоинства и недостатки. Достоинство систем с общей памятью – универсальность модели параллельного ПО, не требующей какого-либо дополнительного кода, или значительных изменений кода. Плюс кластерных систем – отказоустойчивость и лучшая масштабируемость. Системы с общей памятью SMP (общая память для всех процессоров) и NUMA (у каждого процессора, кроме доступа к общей памяти, есть локальная память) плохо масштабируются при росте числа процессоров. Кластеры масштабируются плохо из-за возрастающей сложности сети при добавлении узлов, но это происходит, когда число процессоров измеряется сотнями и даже тысячами4 [3-5]. С давних времен сложилось так что Intel продвигает SMP системы на рынке, а AMD – NUMA5. В случае Intel-а связь между процессорами осуществляется на основе QPI (QuickPath Interconnect), соответственно для AMD – это HyperTransport.

Конец 1980-х и начало 1990-х годов охарактеризовались сменой магистрального направления развития суперкомпьютеров от векторно-конвейерной обработки данных к большому и сверхбольшому числу параллельно соединенных скалярных процессоров. Массивно-параллельные системы стали объединять в себе сотни и даже тысячи отдельных процессорных элементов, причем ими могли служить не только специально разработанные, но и общеизвестные и доступные в свободной продаже процессоры. Большинство массивно-параллельных компьютеров создавалось на основе мощных процессоров с архитектурой RISC (Reduced Instruction Set Computer), наподобие Power PC или PA-RISC (процессоры с внекристальной реализацией кэша) [6].

В настоящее время развивается технология построения больших и суперкомпьютеров на базе кластерных решений. По мнению многих специалистов, на смену отдельным независимым суперкомпьютерам должны прийти группы высокопроизводительных серверов, объединяемых в кластер. Удобство построения кластерных систем заключается в том, что можно гибко регулировать необходимую производительность, подключая к кластеру с помощью аппаратных и программных интерфейсов обычные серийные серверы до тех пор, пока не будет получен суперкомпьютер требуемой мощности. Кластеризация позволяет манипулировать группой серверов как одной системой, упрощая управление и повышая надежность. Со списком ведущих производителей серверов и кластерных систем можно познакомиться на сайте6.

Одной из ведущих компаний мира, имеющей, пожалуй, самую мощную в отрасли исследовательскую команду, способную решать многогранные внедренческие задачи практически любого уровня сложности, на протяжении более столетней деятельности является IBM. Вполне очевидно, что реальной физической базой кластерных систем высокой готовности от IBM может выступать практически любой сервер компании. Это как простые и недорогие стандартные машины из серии eServer xSeries на базе 32-разрядной архитектуры x86 (в том числе с использованием 64-разрядных расширений EM64T и AMD64) и 64-разрядной архитектуры Itanium, так и производительные серверы eServer iSeries и eServer pSeries на базе архитектуры POWER5 и мощные мэйнфреймы eServer zSeries [7]. Стоит отметить, что, держа в своих руках все нити жизненного цикла аппаратных платформ, IBM неуклонно проводит политику переноса инновационных технических и технологических решений с систем высшего уровня zSeries (где они в первую очередь появляются) на системы среднего - iSeries и pSeries, а затем и низшего уровня – xSeries, обеспечивая заказчикам единую архитектуру законченных решений и вместе с тем избавляя их от избыточных вложений в поддержание и развитие ИТ-инфраструктуры. Рассматривая возможности построения суперкомпьютеров и кластеров на процессорах и серверах производства IBM в первую очередь следует остановиться на новейших процессорах IBM – Power9 и Z14.

Разработчики IBM выпустили RISC-процессор Power9, претендующий на роль ведущего обработчика больших данных [8]. Следует заметить, что SPARC64 и архитектура Power – единственные сегодня действующие представители RISC-архитектуры высокой производительности. Power9 производят по 14-нанометровой технологии Fin-FET (транзисторы с трехмерными затворами), а процессор содержит 8 млрд транзисторов на площади 695 кв. мм7), что чуть больше, чем в Power8. Структура конвейеров разделяется на фронтальный компонент (внешний, до начала выполнения команд отвечающий, например, за диспетчирование) и компонент блоков выполнения (EU). Усовершенствованы были оба компонента. Процессорное ядро Power9 типа SMT4 обеспечивает аппаратную поддержку четырех программных нитей (Simultaneous MultiThreading 4). Но Power9 может базироваться и на ядрах других типов – SMT8. Общее число процессорных ядер SMT4 в микросхеме Power9 равно 24, а в варианте с SMT8 – 12. SMT-ядра поддерживают внеочередное выполнение команд; SMT4

[1] Серверы (мировой рынок) [Электронный ресурс] // TAdviser. 2019. URL: http://www.tadviser.ru/index.php (дата обращения: 17.01.2019).

[2] Greene T. 10 of the world's fastest supercomputers. The cream of the Top500 supercomputer list [Электронный ресурс] // Network World. 2018. URL: https://www.networkworld.com/article/3236875/embargo-10-of-the-worlds-fastest-supercomputers.html#slide1 (дата обращения: 17.01.2019).

[3] Компьютерные системы компании Cray INC. в списке Top500 [Электронный ресурс]. URL: http://skif.pereslavl.ru/~csa/kurs_1/Company/Cray/cray.htm (дата обращения: 17.01.2019).

[4] Ведущие производители высокопроизводительных компьютеров [Электронный ресурс] // PARALLEL.RU. URL: https://parallel.ru/computers/vendors.html (дата обращения: 17.01.2019).

[5] Архитектура современных суперкомпьютеров [Электронный ресурс] // OpenMP.ru. 2008. URL: http://openmp.ru/2008/09/12/arxitektura-sovremennyx-superkompyuterov/ (дата обращения: 17.01.2019).

[6] Ведущие производители высокопроизводительных компьютеров [Электронный ресурс] // PARALLEL.RU. URL: https://parallel.ru/computers/vendors.html (дата обращения: 17.01.2019).

[7] POWER9 - Microarchitectures - IBM [Электронный ресурс]. URL: https://en.wikichip.org/wiki/ibm/microarchitectures/power9 (дата обращения: 17.01.2019).

обеспечивает возможность завершения до 128 команд на каждом такте, а SMT8 – до 256 команд.

Процессор Power9 с 24 SMT4-ядрами предназначен для универсального применения в качестве платформы для ОС Linux, 12-ядерный Power9 с SMT8 ориентируется на работу систем с виртуализацией через PowerVM для Linux и ОС IBM i и AIX. В версии ядра Linux 4.12 уже имеется расширенная поддержка Power9. Процессор Power9 вместе с Nvidia V100 (Volta) будет применяться в вычислительных узлах суперкомпьютеров Summit и Sierra в знаменитых Окриджской и Ливерморской национальных лабораториях США, что уже говорит само за себя8. Данные по производительности в тестах систем на Power9 пока не опубликованы, однако, например, схожая по архитектуре конфигурация на базе гетерогенных вычислительных узлов с Power8 и с 4 GPU Nvidia P100 на 64 узлах демонстрирует ускорение 95% от линейного910 [9]. Summit из американской Окриджской национальной лаборатории согласно спецификациям обеспечивает производительность в 200 PFLOPS двойной точности (FP64) и 3,3 EFLOPS смешанной точности, включая INT811.

Объявленный в июле прошлого года последним мэйнфреймом IBM является z14, который был запущен в 2015 году1213. Для z14 IBM продолжала ориентироваться на три основных вектора: увеличение производительности на потоке, пропускная способность системы и повышение эффективности. Одним из основных факторов достижения этих целей является технология процесса производства микропроцессоров. Как и в случае с POWER9, IBM использует уникальный уникальный 14-нм SOI FinFET (14HP) GlobalFoundries со встроенной DRAM для извлечения большей плотности и снижения мощности14. Система построена на 10-ядерных процессорах при частоте 5,2 ГГц, ядра поддерживают Simultaneous Multithreading (SMT), но в отличие от настольных CPU каждое ядро может работать не с двумя, а с четырьмя потоками одновременно. Десятиядерный процессор может обрабатывать 40 потоков. z14 поддерживает повышенную эффективность виртуализации ядер Linux и более высокую пропускную способность встроенных процессоров z Integrated Information Processor (zIIP). Последний представляет собой специализированный процессор, работающий асинхронно с основным процессором мейнфрейма, который предназначен для повышения эффективности использования вычислительных ресурсов. Благодаря многопоточности производительность z14 стала до 25 % выше при использовании IFL (Integrated Facility for Linux) или zIIP1. Теперь многопоточная обработка поддерживает специализированные процессоры ввода-вывода под названием System Assist Processor (SAP).

Поскольку кластерная архитектура суперкомпьютеров в настоящее время является преобладающей, а основными ее компонентами (вычислительными узлами), судя по публикациям, во многих случаях являются серверы и мэйнфреймы IBM на процессорах Power9 и z14, вопросы оптимизация вычислительного процесса при выполнении информационно-связанных задач в суперкомпьютерах будем рассматривать применительно к операционным системам, которые используются на этих компьютерах.

2. Операционные системы серверных и кластерных систем IBM

В качестве базовых ОС в каждой серверной серии IBM используются как свои системы, так и ОС сторонних производителей – z/OS и варианты Linux для zSeries; i5/OS, Linux и AIX 5L – для iSeries; AIX 5L и Linux – для pSeries; Microsoft Windows Server, Novell Netware и Linux – для xSeries, оптимизированные для построения кластеров высокой готовности. Необходимо особо отметить, что IBM как многопрофильная технологическая компания, выпускающая все без исключения виды аппаратных ресурсов, применяемых для построения кластерных систем высокой готовности, ориентируется на собственные изделия, лишь изредка предлагая список совместимого оборудования от сторонних поставщиков. Система кластеризации Parallel Sysplex для z/OS позволяет объединять в рамках одного решения до 32 узлов, обеспечивая при этом разделение ресурсов и параллельное исполнение задач. Используемые в ней технические решения гарантируют практически линейный рост производительности при добавлении в кластер новых серверов [10].

Кроме того, расширение Parallel Sysplex – Geographically Dispersed Parallel Sysplex (GDPS) [11, 12] с помощью технологии HyperSwap позволяет строить территориально разнесенные, катастрофоустойчивые системы. В качестве промежуточного кластеризующего ПО на системах iSeries предлагается использовать продукты, поставляемые технологическими партнерами IBM. Основное рекомендуемое решение - MIMIX High Availability Cluster от фирмы Lakeview Technology. Модульность и широкие возможности мониторинга и автоматизации процессов репликации данных и системных объектов, переключения нагрузки между производственной и резервными системами позволяют создавать решения, оптимальные с точки зрения требований бизнес-пользователей.

Система кластеризации High Availability Cluster Multi-Processing версии 5.2, используемая на серверах pSeries с установленной AIX 5L, позволяет организовать на базе ПО Oracle Application Server 10g как системы с разделением ресурсов (для чего используется технология Oracle Real Application Cluster), так и системы активного дублирования без разделения ресурсов – Cold Failover Cluster, использующие один из узлов в режиме го-

[8] Исторический этап: Как изменились принципы построения крупнейших суперкомпьютеров [Электронный ресурс]. 2017. URL: http://www.tadviser.ru/index.php/Статья: Суперкомпьютеры_для_искусственного_интеллекта_(AI_supercomputers) (дата обращения: 17.01.2019).

[9] Щербаков Д. Беглый обзор IBM Power Systems [Электронный ресурс]. URL: https://habr.com/post/116948/ (дата обращения: 17.01.2019).

[10] Feldman M. IBM Pushes Envelope in Deep Learning Scalability [Электронный ресурс] // TOP500 Supercomputer Sites. 2017. URL: https://www.top500.org/news/ibm-pushes-envelope-in-deep-learning-scalability/ (дата обращения: 17.01.2019).

[11] Тадтаев Г. В США создали самый мощный суперкомпьютер в мире [Электронный ресурс] // РБК. 2018. URL: https://www.rbc.ru/technology_and_media/09/06/2018/5b1b6b379a79475afecfc620 (дата обращения: 17.01.2019).

[12] Шиллинг А. IBM z14: 10 ядер на частоте 5,2 ГГц и 30 Мбайт кэша на ядро [Электронный ресурс] // Hardwareluxx Russia. 2017. URL: https://www.hardwareluxx.ru/index.php/news/hardware/prozessoren/42585-ibm-z14.html (дата обращения: 17.01.2019).

[13] Schor D. ISSCC 2018: The IBM z14 Microprocessor And System Control Design [Электронный ресурс]. URL: https://fuse.wikichip.org/news/941/isscc-2018-the-ibm-z14-microprocessor-and-system-control-design (дата обращения: 17.01.2019).

[14] Там же.

рячего резерва на случай сбоя в основном узле. В проектах создания кластерных систем высокой готовности на базе серверов стандартной архитектуры из семейства xSeries IBM полагается на бизнес-партнеров и их программные разработки - в системе Windows предлагается использовать сервисы Microsoft Cluster Services, для Netware применяются Novell Cluster Services, а для Linux используется пакет LifeKeeper компании SteelEye и собственный продукт IBM – Tivoli System Automation, работающий в среде разных версий Linux не только на xSeries, но и на всех аппаратных платформах IBM. Этот продукт позволяет объединять механизмы высокой доступности, уже имеющиеся в программных комплексах более высокого уровня, а также организовывать кластерные системы активного дублирования для тех продуктов, в которых такой поддержки нет.

Современные технологии виртуализации, используемые в серверах и мэйнфреймах IBM, начинают свое развитие от разработки в IBM гипервизора, который был использован для работы виртуальных машин в операционной системе VM/370. В более ранних системах IBM для реализации мультипрограммирования разработчики использовали технологию разделов. В дальнейшем эта технология была использована в виртуальных средах (виртуальных серверах) на базе гипервизора. Корпорация IBM существенно расширила технологию разделов, разработав так называемые динамические LPAR (DLPAR), позволяющие перемещать ресурсы между разделами без перезагрузки системы или самих разделов [13-15]. Во многих публикациях, например, [16, 17] предлагаются методы распараллеливания алгоритмов и программ, методы организации оптимальных параллельных вычислительных процессов управления и информационного обслуживания, методы диспетчирования и синхронизации. Обсуждается применение методов параллельного программирования при разработке GRID-технологий.

Сервер может содержать несколько изолированных друг от друга виртуальных серверов, работающих одновременно, и разделяющих ресурсы. Каждый логический (виртуальный сервер) выполняется в логическом разделе LPAR (Logical Partition Access Resources), в который устанавливается гостевая операционная система. Максимальное количество LPAR на сервере зависит от количества процессоров и памяти: на один процессор физического сервера можно создать максимум 10 LPAR, выделив на каждый LPAR минимальную долю в 0,1 часть ресурсов процессора и минимум по 256 Мб оперативной памяти. Теоретическое ограничение тоже есть: например, для Power 795 модели можно создать максимум 254 LPAR15. Таким образом можно оптимизировать нагрузку на серверы, поделив процессорный пул между виртуальными серверами, например, какому-то LPAR дать 0,5 процессора, а более нагруженному – 1,2.

Существует большое количество операционных систем совместимых с LPARs, включая z/OS, z/VM, z/VSE, z/TPF, AIX, Linux и i/OS. На универсальных компьютерах типа IBM, LPAR управляет гипервизор PR/SM (Processor Resource/System Manager) [16]. Универсальные компьютеры типа IBM работают исключительно в режиме LPAR, даже когда только один раздел на машине. Всеми LPAR управляет гипервизор. Для управления LPAR применяются следующие гипервизоры: для zSeries - PR/SM (Processor Resource/System Manager); для IBM Power Systems - Power Hypervisor (PowerVM). Для взаимодействия между LPAR внутри одного физического сервера используется виртуальная сеть – HiperSocket. PR/SM (Processor Resource/System Manager) представляет собой гипервизор 1-го типа (монитор виртуальных машин), который позволяет использовать несколько логических разделов для разделения физических ресурсов, таких как процессоры, каналы ввода-вывода и прямых запоминающих устройств доступа (DASD). Гипервизор работает на основе компонента CP VM/XA непосредственно на уровне машины, и выделяет системные ресурсы через LPAR'ы, чтобы совместно использовать физические ресурсы. Это стандартная функция на Системе z IBM, Системе p и i машин [16].

3. Формализация и решение задачи планирования вычислительного процесса

3.1. Представление структуры приложения и постановка задачи

Будем считать, что выполнение приложения производится на суперкомпьютере IBM типа Power или z. Известна структура приложения, состоящего из некоторого множества информационно-связанных частей (задач) с известным (ожидаемым) временем выполнения каждой задачи. Предполагаем, что каждая задача выполняется отельным логическим (виртуальным сервером) в логическом разделе LPAR, в который устанавливается гостевая операционная система. Считаем, что это время определяется элементарным вычислителем (процессором) многопроцессорного компьютера. Взаимодействие LPAR внутри физического сервера обеспечивается общей оперативной памятью или использованием виртуальной сети – HiperSocket [17]. Структуру подлежащего выполнению приложения удобно представить взвешенным ориентированным графом, как это показано на рис. 1, который будем далее использовать в качестве иллюстративного примера решения поставленной задачи (общность предлагаемого решения задачи от этого не изменится):

$G = \{\langle s_i, s_j \rangle, t_{ij} | j > i, i = 0,1,\ldots, M-2, j = 1,2,\ldots, M-1\}$, где s_i, s_j – номера событий (вершины) в графе, t_{ij} – время решения задачи (вес соответствующей дуги), M – количество задач в пакете G. Считаем, что временные характеристики приложения известны по результатам его разработки и определены из условия выполнения каждой задачи приложения на одном процессоре (в одном LPAR) суперкомпьютера, выбранного для реализации этого приложения.

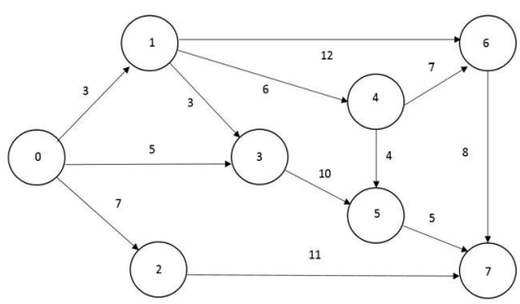


Рис. 1. Структура приложения
Fig. 1. Application struct

[15] Щербаков Д. Беглый обзор IBM Power Systems [Электронный ресурс]. URL: https://habr.com/post/116948/ (дата обращения: 17.01.2019).

Требуется разработать план $P_{opt}(G)$ реализации заданного набора задач (ЗНЗ), обеспечивающий выполнение этих задач за время $T(G)$, не превышающее требуемого директивного значения T_d при условии минимизации ресурсов $Rs(G)$, занятых в процессе выполнения ЗНЗ. Таким образом, требуется найти такой план

$$P_{opt}(G) \in \{P(G) | T(G) \le T_d\}, \quad (1)$$

при котором обеспечивается

$$minRs(G) = Rs(G) | P_{opt}(G). \quad (2)$$

Здесь правая часть выражения (1) означает множество возможных планов организации вычислительного процесса, которые обеспечивают завершение ЗНЗ за требуемое время. Оптимальным из этого множества будет план, для которого выполняется условие (2). Правая часть выражения (2) означает, что оптимальный план реализуется с минимальными затратами ресурсов системы.

Заметим, что решению задачи в постановке (1) – (2) или несколько иной форме посвящено достаточно много публикаций. Например, в [18] дана схема планирования работы параллельного суперкомпьютера, в фундаментальной монографии [19] даются методы статического планирования и балансировки нагрузки в параллельных и распределенных системах. Здесь содержится обзор и подробное обсуждение широкого спектра тем: от теоретических основ до практических методов планирования и распределения нагрузки. Значительное количество публикаций посвящено распараллеливанию алгоритмов и программ (например, [20]) а также методам параллельных вычислений при решении ряда задач оптимизации, методам организации оптимальных параллельных вычислительных процессов управления и информационного обслуживания, методам диспетчирования и синхронизации, применением методов параллельного программирования при разработке GRID-технологий [21]. В отличие от подобных публикаций, целью настоящей статьи является организация вычислительного процесса в суперкомпьютерах с динамически управляемыми разделами. При этом задача в постановке (1) – (2) заключается в минимизации требуемого числа разделов (т.е. минимизация требуемых ресурсов) при условии выполнения ЗНЗ в требуемое директивное время.

Для дальнейшей формализации задачи будем использовать понятия сетевого планирования и управления. В нашем случае работы представляются дугами графа $\langle s_i, s_j \rangle$, или просто (i, j), причем для любой дуги $j > i$. Обмен информацией между задачами инициируется событиями, например, событие s_1 инициирует завершение работы $\langle s_0, s_1 \rangle$, длительностью t_{01}, возможность запуска вычислений $\langle s_1, s_4 \rangle, \langle s_1, s_6 \rangle$, и $\langle s_1, s_{31} \rangle$. Организация вычислительного процесса при выполнении данного приложения сводится к определению временных параметров сетевого графика и к его оптимизации по длительности выполнения всего комплекса задач и затратам вычислительных ресурсов в соответствии с заданными требованиями. Следует иметь ввиду, что, как правило, на суперкомпьютере выполняется в пакетном режиме несколько различных приложений, и вычислительные ресурсы для реализации каждого из них ограничены.

3.2. Решение задачи

3.2.1. Определение величины критического пути и резервов времени по отдельным вычислительным работам

Критический путь T_{kr} – это полный путь, определяющий длительность всего комплекса вычислительных работ и имеющий наибольшую продолжительность. Для решения поставленной задачи необходимо найти длину критического пути и возможные резервы времени при выполнении отдельных вычислительных работ. Определение длины критического пути можно проводить различными способами. Наиболее удобным способом расчёта сетевого графика при небольшом количестве работ является расчёт непосредственно на сети графика и заключается в нахождении критического пути и определении резервов времени для работ, которые не располагаются на этом пути.

При расчетах сетевых моделей применяют следующие наименования ее параметров [22, 23]:

1. Продолжительность работы (t_{ij}) (здесь i и j – номера соответственно начального и конечного событий.
2. Раннее начало работы $t_{ij}^{p.н}$ – характеризуется выполнением всех предшествующих работ и определяется продолжительностью максимального пути от исходного события всей модели до начального события рассматриваемой работы.
3. Раннее окончание работы $t_{ij}^{p.o}$ – определяется суммой раннего начала и продолжительности рассматриваемой работы.
4. Позднее начало работы $t_{ij}^{п.н}$ – определяется разностью позднего окончания и продолжительности рассматриваемой работы.
5. Позднее окончание работы $t_{ij}^{п.о}$ – определяется разностью продолжительности критического пути и максимальной продолжительности пути от завершающего события всей модели до конечного события рассматриваемой работы.
6. Общий резерв времени работы R_{ij} – характеризуется возможностью роста продолжительности работы без увеличения продолжительности критического пути и определяется как разность между поздним и ранним окончанием рассматриваемой работы. Общий резерв работы принадлежит не только первой работе, но и всем последующим работам данного пути. В случае использования на одной из работ общего резерва критический путь не изменит своей продолжительности, но все последующие работы окажутся критическими и лишатся резерва.
7. Частный резерв времени работы r_{ij} – характеризуется возможностью увеличения продолжительности работы без изменения раннего начала последующей работы и определяется разностью между ранним началом последующей работы и ранним окончанием рассматриваемой работы. Частный резерв имеет место, когда одним событием заканчивается не менее двух работ. Отличие частного резерва от общего заключается в том, что частный резерв может быть использован только на рассматриваемой или предшествующих работах и не может быть использован на последующих.
8. Полным резервом некоторого пути в сетевой модели R называют разность между продолжительностью критического пути модели и продолжительностью рассматриваемого пути.

Результаты расчета сетевой модели выполнения пакета задач, представленного на рис. 1, приведены в табл.1. Работы, не имеющие резерва времени и выделенные жирным шрифтом, образуют критический путь T_{kr} = 24.

Таблица 1. Таблица расчетов параметров сетевого графика
Table 1. Calculation table of network parameters

Работа (ij)	Продолжительность работы t_{ij}	Ранее начало работы $t_{ij}^{р.н}$	Ранее окончание работы $t_{ij}^{р.о}$	Позднее начало работы $t_{ij}^{п.н}$	Позднее окончание работы $t_{ij}^{п.о}$	Общий резерв работы R_{ij}	Частный резерв работы r_{ij}
(01)	3,00	0,00	3,00	0,00	3,00	**0,00**	0,00
(02)	7,00	0,00	7,00	6,00	13,00	6,00	0,00
(03)	5,00	0,00	5,00	4,00	9,00	4,00	0,00
(13)	3,00	3,00	6,00	6,00	9,00	3,00	0,00
(14)	6,00	3,00	9,00	3,00	9,00	**0,00**	0,00
(16)	12,00	3,00	15,00	4,00	16,00	1,00	0,00
(27)	11,00	7,00	18,00	13,00	24,00	6,00	0,00
(35)	10,00	6,00	16,00	9,00	19,00	3,00	3,00
(45)	4,00	9,00	13,00	15,00	19,00	6,00	0,00
(46)	7,00	9,00	16,00	9,00	16,00	**0,00**	0,00
(57)	5,00	16,00	21,00	19,00	24,00	3,00	3,00
(67)	8,00	16,00	24,00	16,00	24,00	**0,00**	0,00

Таблица 2. Исходные данные для решения задачи поиска критического пути
Table 2. Source data for solving the problem of finding a critical path

	Переменные	x_{01}	x_{02}	x_{03}	x_{13}	x_{14}	x_{16}	x_{27}	x_{35}	x_{45}	x_{46}	x_{57}	x_{67}	Огранич. функция	Ограничение
	Значения переменных	1	1	1	1	1	1	1	1	1	1	1	1		
Номера узлов	0	1	1	1										3	1
	1	-1			1	1	1							2	0
	2		-1					1						0	0
	3			-1	-1				1					-1	0
	4					-1				1	1			1	0
	5								-1	-1		1		-1	0
	6						-1				-1		1	-1	0
	t_{ij}	3	7	5	3	6	12	11	10	4	7	5	8		

3.2.2. Минимизация длины критического пути

Имеющийся резерв времени по работам, не лежащим на критическом пути, свидетельствует о том, что имеется возможность изъятия ресурсов, выделенных на некоторые работы, с целью добавления ресурсов на работы, лежащие на критическом пути. Это приведет к уменьшению длины критического пути, т.е. к сокращению всего цикла выполняемых вычислений. В нашем примере суммарный резерв времени составляет значение

$$R = \sum_{i=0}^{M-2}\sum_{j=1}^{M-1} R_{ij} = 32.$$

Передача ресурса от какой-либо работы означает увеличение ее выполнения на величину изъятого ресурса, добавление ресурса к другой работе, лежащей на критическом пути, означает уменьшение ее выполнения на величину добавленного ресурса. Например, изъятие половины резерва, т.е. трех единиц ресурса от работы (02) означает ее удлинение до 10 единиц времени. Это означает, что для ее выполнения нужен будет виртуальный процессор со следующим значением доли физического процессора

$$V_{02} = \frac{t_{02}}{t_{02} + 0,5 * R_{02}} = \frac{7}{7 + 0,5 * 6} = 0,7.$$

Изъятый ресурс можно добавить с целью сокращения критического пути, например, к работе (67). При этом время выполнения этой работы уменьшится с 8 до 5 единиц времени, но для выполнения этой работы потребуется виртуальный процессор со следующим значением доли физического процессора

$$V_{67} = \frac{t_{67}}{t_{67} - 0,5 * R_{02}} = \frac{8}{8 - 0,5 * 6} = 1,6.$$

Для решения задачи минимизации критического пути необходимо построить модель линейного программирования с целью определения значений T_{kr} для различных вариантов перераспределения ресурсов. Удобно это сделать в электронных таблицах, например, Excel. Воспользуемся для этого формализацией задачи поиска критического пути, предложенной в [24]. Исходные данные удобно представить в форме таблицы (табл. 2).

Задача поиска критического пути на основе табл. 2 заключается в определении значения функции

$$T_{kr} = \sum_{i=0}^{i=6}\sum_{j=1}^{j=7} t_{ij} \cdot x_{ij} \to max \quad (1)$$

при следующих ограничениях:

$$1 \cdot x_{01} + 1 \cdot x_{02} + 1 \cdot x_{03} = 1; \quad (2)$$

$$-1 \cdot x_{01} + 1 \cdot x_{13} + 1 \cdot x_{14} + 1 \cdot x_{16} = 0 \quad (3)$$

$$-1 \cdot x_{02} + 1 \cdot x_{27} = 0; \quad (4)$$

$$-1 \cdot x_{03} - 1 \cdot x_{13} + 1 \cdot x_{35} = 0; \quad (5)$$

$$-1 \cdot x_{14} + 1 \cdot x_{45} + 1 \cdot x_{46} = 0; \quad (6)$$

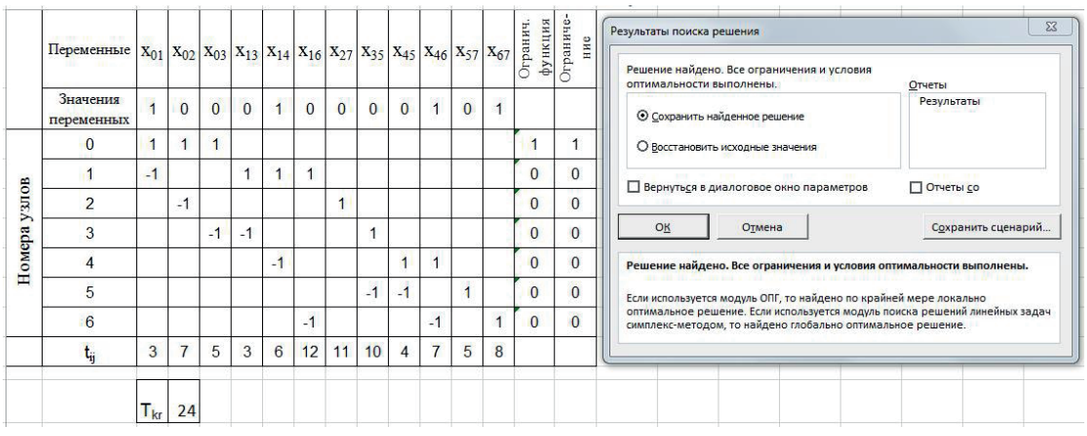
$$-1 \cdot x_{35} - 1 \cdot x_{45} + 1 \cdot x_{57} = 0; \quad (7)$$

$$-1 \cdot x_{16} - 1 \cdot x_{46} + 1 \cdot x_{67} = 0; \quad (8)$$

$$\forall x_{ij} \in \{0,1\} \mid i = 0,1,\ldots 6; j = 1,2,\ldots 7. \quad (9)$$

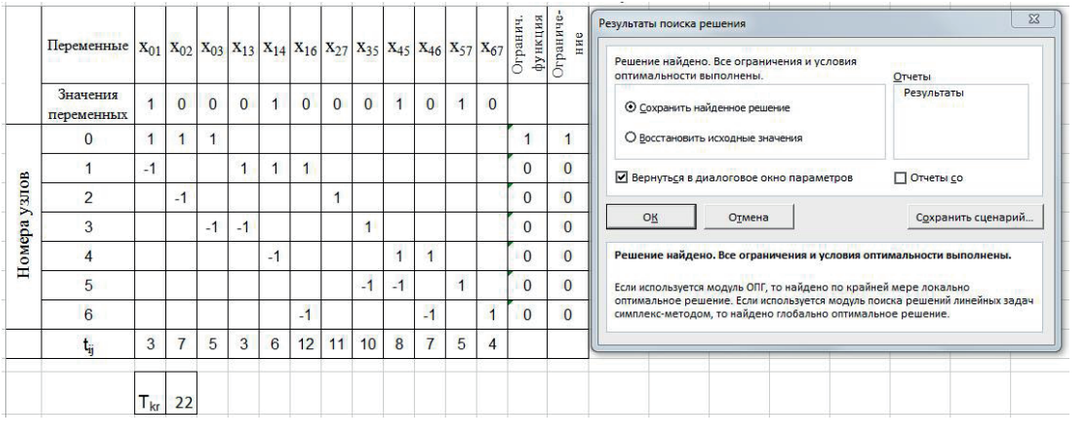
Решение задачи (1) – (9) для исходного графа сетевой модели показано на рис. 2.

Определим теперь, как изменится величина критического пути, если 4 единицы резерва изъять у работы (45) и передать их для выполнения работе (67). При этом задача (45) будет выполняться 8 единиц времени, а задача (67) – 4 единицы времени. Решение задачи в этом случае показано на рис. 3 и дает значение критического пути, равное 22. Если рассчитанное значение критического пути окажется больше директивного, можно рассмотреть другие варианты использования резервов других работ, пока не будет полученное требуемое значение критического пути. Если этого не удается сделать за счет резервов в выполнении работ, необходимо увеличить количество ресурсов для их выполнения (в рассматриваемой задаче мы исходили из предположения, что каждой задаче выделяется один процессор).



	Переменные	x_{01}	x_{02}	x_{03}	x_{13}	x_{14}	x_{16}	x_{27}	x_{35}	x_{45}	x_{46}	x_{57}	x_{67}	Огранич. функция	Ограниче-ние
	Значения переменных	1	0	0	0	1	0	0	0	0	1	0	1		
Номера узлов	0	1	1	1										1	1
	1	-1			1	1	1							0	0
	2		-1					1						0	0
	3			-1	-1				1					0	0
	4					-1				1	1			0	0
	5								-1	-1		1		0	0
	6						-1				-1		1	0	0
	t_{ij}	3	7	5	3	6	12	11	10	4	7	5	8		
	T_{kr}	24													

Рис. 2. Критический путь исходного графа сетевой модели
Fig. 2. The critical path of the original graph of the network model



	Переменные	x_{01}	x_{02}	x_{03}	x_{13}	x_{14}	x_{16}	x_{27}	x_{35}	x_{45}	x_{46}	x_{57}	x_{67}	Огранич. функция	Ограниче-ние
	Значения переменных	1	0	0	0	1	0	0	0	1	0	1	0		
Номера узлов	0	1	1	1										1	1
	1	-1			1	1	1							0	0
	2		-1					1						0	0
	3			-1	-1				1					0	0
	4					-1				1	1			0	0
	5								-1	-1		1		0	0
	6						-1				-1		1	0	0
	t_{ij}	3	7	5	3	6	12	11	10	8	7	5	4		
	T_{kr}	22													

Рис. 3. Вариант расчета критического пути сетевой модели выполнения графа задач при использовании резерва задачи (45)
Fig. 3. Variant of calculating the critical path of the network model of performing the task graph when using the task reserve (45)

3.2.3. Минимизация количества ресурсов выполнения пакета без увеличения длины критического пути

Будем считать, определенный выше критический путь, равный 24 единицам времени, соответствует директивному времени решения заданного пакета задач, т.е. $T_{kr} = T_d$. В этом случае становится актуальной задача минимизации ресурсов (в нашем случае количества процессоров), необходимых для реализации всех задач при условии отсутствия превышения длины критического пути. Другими словами, длительности выполнения всех работ пакета нужно изменить так, чтобы длина любого пути в графе была в идеале равна длине критического пути. Практически все вычислительные работы будут увеличены с учетом возможных резервов времени для их выполнения.

Для формализации задачи введем дополнительные обозначения:

t_{ij}^x – время выполнения работы (ij) после минимизации количества выделяемых ресурсов;

T_i^x – время свершения событий в графе работ (события отождествляются с вершинами графа);

N – исходное число процессоров, которое позволяет выполнить все работы по принципу: один процессор на одну работу;

N_m – минимальное количество процессоров, которое будет получено в результате решения оптимизационной задачи (пока еще без учета возможности их параллельной работы). Значение минимального количества процессоров определяется как сумма долей физических процессоров в виртуальных единичных процессорах логических разделах LPAR, т.е. $N_m = \sum_{i=0}^{i=6}\sum_{j=1}^{j=7} d_{ij}$,

где d_{ij} – доля физического процессора соответствующего виртуального процессора LPAR, необходимая для выполнения работы (ij) после минимизации количества выделяемых ресурсов, $d_{ij} = 1 - \frac{\left(t_{ij}^x - \bar{t}_{ij}\right)}{t_{ij}^x}$.

В качестве целевой функции задачи выбираем время занятости процессоров (полное машинное время) на решение пакета задач – его нужно минимизировать за счет использования имеющихся резервов времени на выполнение отдельных задач. Таким образом, целевая функция имеет следующий вид:

$$\sum_{i=0}^{i=6}\sum_{j=1}^{j=7} t_{ij} - \sum_{i=0}^{i=6}\sum_{j=1}^{j=7}(t_{ij}^x - t_{ij}) \to min \qquad (10)$$

Первое слагаемое этой функции представляет собой полные затраты машинного времени на решение всего пакета задач. Второе – экономию машинного времени при условии того, что можно увеличить время решения некоторых задач за счет имеющегося резерва времени на их выполнение (здесь $(t_{ij}^x - t_{ij}) \geq 0$). Рассмотрим ограничения, которые должны учитываться при решении этой задачи.

Операция	Продолжительность работы t_{ij}	R_{ij}	$t_{ij} + R_{ij}$	t^x_{ij}	$t^x_{ij} - t_{ij}$	Исходное число процессоров N	Минимальное число процессоров Nm	Путь	Сумма работ по пути
(01)	3	0	3	3	0	1	1,00	0-2-7	24
(02)	7	6	13	13	6	1	0,54	0-3-5-7	24
(03)	5	4	9	9	4	1	0,56	0-1-6-7	24
(13)	3	3	6	6	3	1	0,50	0-1-3-5-7	24
(14)	6	0	6	6	0	1	1,00	0-1-4-5-7	24
(16)	12	1	13	13	1	1	0,92	0-1-4-6-7	24
(27)	11	6	17	11	0	1	1,00		
(35)	10	3	13	10	0	1	1,00		
(45)	4	6	10	10	6	1	0,40		
(46)	7	0	7	7	0	1	1,00		
(57)	5	3	8	5	0	1	1,00		
(67)	8	0	8	8	0	1	1,00		
	81			101	20	12	9,92		
	Fцел	61							

Рис. 4. Представление задачи (10) – (12) в электронных таблицах
Fig. 4. Representation of task (10) - (12) in spreadsheets

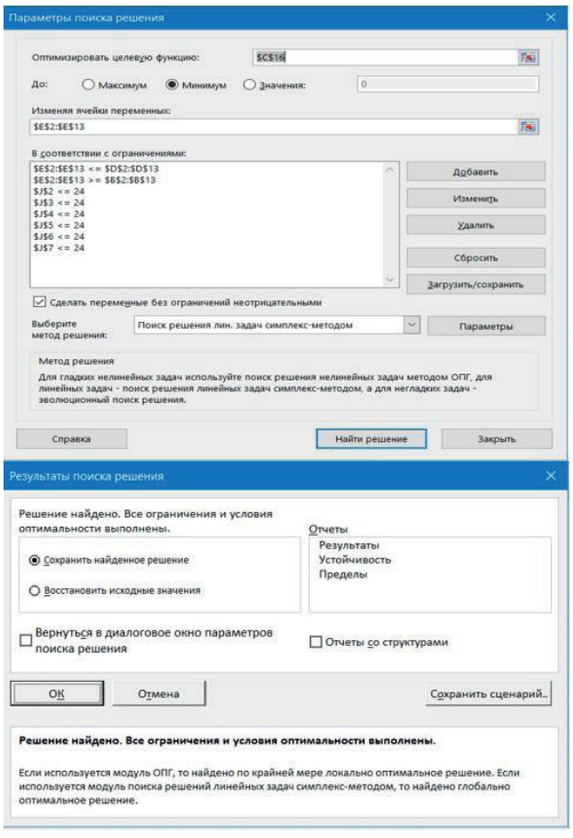


Рис. 5. Решение задачи (10) – (12) в электронных таблицах
Fig. 5. Solution of the problem (10) - (12) in spreadsheets

Первый вид ограничений связан с принятым условием использования для решения задачи только имеющихся резервов для выполнения задач пакета. Отсюда следует система ограничений для продолжительности выполнения работ следующего вида: $t_{ij} + R_{ij} \geq t^x_{ij} \geq t_{ij}$. (11)
Второй вид ограничений должен обеспечить такое изменение длительности выполнения всех работ пакета, чтобы длина любого пути в графе была в идеале равна длине критического пути. В рассматриваемом примере таких путей шесть (перечислим их последовательностями вершин графа): P_1 : 0 – 2 – 7; P_2 : 0 – 3 – 5 – 7; P_3 : 0 – 1 – 6 – 7; P_4 : 0 – 1 – 3 – 5 – 7; P_5 : 0 – 1 – 4 – 5 – 7; P_6 : 0 – 1 – 4 – 6 – 7. Ограничения на длину этих путей имеют следующий вид: $L_i(P_i) \leq T_{kr} \,|\, 1 = 1,2,\ldots,M-1$ (12)
Решение задачи (10) – (13) показано на рис. 4 и 5.
Как видно из результата решения задачи минимизации использования ресурсов, число процессоров для реализации пакета задач без увеличения длины критического пути можно сократить с 12 до 10. Однако структура пакета свидетельствует о возможной параллельности работы этих процессоров при выполнении задач пакета.

3.2.4. Возможности организации мультипроцессорного выполнения пакета задач, представленного сетевой моделью

Потенциальную параллельность выполнения заданного набора задач можно определить, преобразовав граф задач в ярусно-параллельную форму [20, 25]. Ярусно-параллельная форма графа (ЯПФ) – деление вершин ориентированного ациклического графа на перенумерованные подмножества V_j такие, что, если дуга идет от вершины $v_l \in V_j$ к вершине $v_m \in V_k$,то обязательно $j < k$.
Каждое из множеств V_j называется ярусом ЯПФ, *j* – его номером, количество вершин $|V_j|$ в ярусе – его шириной. Количество ярусов в ЯПФ называется её высотой, а максимальная ширина её ярусов – шириной ЯПФ. Для ЯПФ графа алгоритма важным является тот факт, что операции, которым соответствуют вершины одного яруса, не зависят друг от друга (не находятся в отношении связи), и поэтому заведомо существует параллельная реализация алгоритма, в которой они могут быть выполнены параллельно на разных устройствах вычислительной системы. Поэтому ЯПФ графа алгоритма может быть использована для подготовки такой параллельной реализации алгоритма.
Для получения ЯПФ пакета задач, представленного в форме сетевой модели, как показано на рис. 1, его предварительно необходимо преобразовать, заменив дуги графа (работы) вершинами. В преобразованном графе (рис. 6) номера вершин соответствуют работам, длительность которых пересчитана в соответствии с решением задачи (10 – 12) по рис.4.

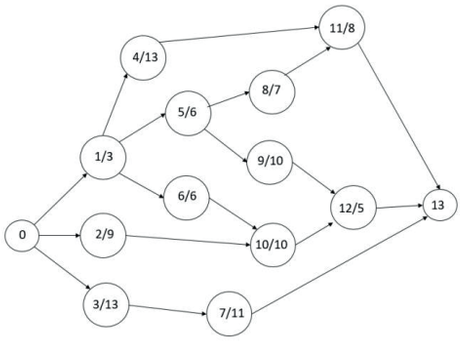


Рис. 6. Преобразованный граф пакета задач приложения
Fig. 6. Transformed application task package graph

Получить ЯПФ можно, построив матрицу смежности графа.
Матрица смежности – это квадратная матрица размерностью $(M+1)\times(M+1)$, (где M – число вершин графа), однозначно представляющая его структуру. Обозначим ее как $A = |a_{ij}|$, где каждый элемент матрицы определяется следующим образом: $a_{ij} = 1$, если есть дуга (i, j), $a_{ij} = 0$, если нет дуги (i, j). В нашем примере матрица смежности будет иметь следующий вид, приведенный на рис. 7.

номер вершины	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	0	1	1	1	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	1	1	1	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	1	0	0	0
3	0	0	0	0	0	0	0	1	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	1	0	0
5	0	0	0	0	0	0	0	0	1	1	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	1	0	0	0
7	0	0	0	0	0	0	0	0	0	0	0	0	0	1
8	0	0	0	0	0	0	0	0	0	0	0	1	0	0
9	0	0	0	0	0	0	0	0	0	0	0	0	1	0
10	0	0	0	0	0	0	0	0	0	0	0	0	1	0
11	0	0	0	0	0	0	0	0	0	0	0	0	0	1
12	0	0	0	0	0	0	0	0	0	0	0	0	0	1
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Рис. 7. Матрица смежности
Fig. 7. The adjacency matrix

Алгоритм распределения модулей системы по уровням:
1. Находим в матрице нулевые строки. В нашем случае это только одна строка с номером 13.
2. Вершина с этим номером образует нулевой (низший) уровень ЯПФ.
3. Вычеркиваем столбцы с номерами найденных вершин. В нашем случае – столбец 13.
4. Находим в матрице нулевые строки (7, 11, 12). Это вершины 1-го уровня.
5. Вычеркиваем столбцы с номерами 7, 11, 12.
6. Находим в матрице нулевые строки (3, 4, 8, 9 и 10). Это вершины 2-го уровня.
7. Вычеркиваем столбцы с номерами найденных вершин.
8. Находим в матрице нулевые строки (2, 5, 6). Это вершины 3-го уровня.
9. Вычеркиваем столбцы с номерами 5, 6.
10. Вершина с номером 1 образует 4-й уровень.

ЯПФ графа задач пакета, поученная на основе матрицы смежности представлена на рис. 8.
Построенный граф задач в ЯПФ далеко не прямоугольный, что неудобно для организации параллельного вычислительного процесса. Визуально из рис. 8 понятно, что граф можно легко перестроить, не меняя связей между вершинами, например, можно переместить вершины 2 и 3 на 4-й уровень, а вершину 4 – на 3-й. После таких преобразований граф примет вид, показанный на рис. 9. Здесь у каждой вершины курсивом дано число долей физического процессора, которое необходимо разделу LPAR выполнения задачи в соответствии с решением, минимизирующим количество ресурса процессоров для выполнения всего пакета задач (рис. 4).

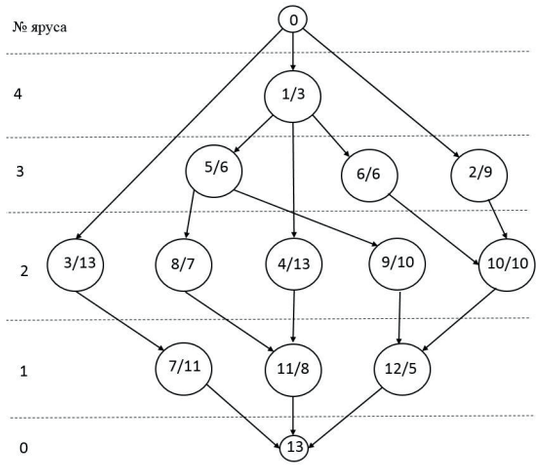


Рис. 8. ЯПФ графа пакета задач приложения
Fig. 8. Multilevel structure of the application task package graph

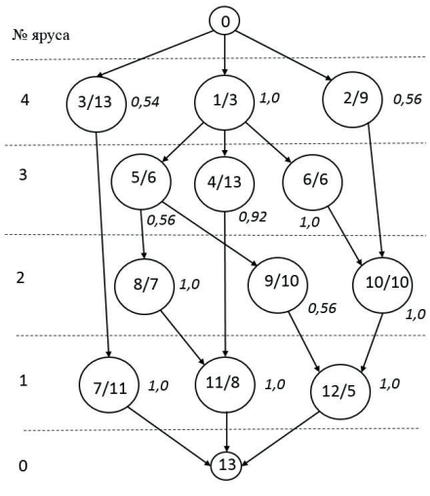


Рис. 9. ЯПФ пакета задач приложения
Fig. 9. Multilevel structure of an application task package

Рис. 10. Диаграмма выполнения вычислительных работ пакета
Fig. 10. Diagram of the computational package

По ЯПФ (рис. 9) легко построить план реализации вычислительного процесса во времени. На рис. 10 сверху показана шкала времени (равная длине критического пути), под которой обозначены временные промежутки реализации задач пакета. В нижней части диаграммы приведены сведения о необходимом количестве логических разделов LPAR и количестве долей физических процессорных модулей (здесь необходимо округление с учетом дискретности выделения 0,1), образующих виртуальные процессоры разделов. Вертикальными стрелками показаны передачи данных между разделами. Конкретное представление найденного плана выполнения вычислительных работ по реализации ЗНЗ $P_{opt}(G)$ зависит от особенностей диспетчера вычислительной системы и в статье не рассматривается.

4. Обсуждение результатов предложенного подхода к организация вычислительного процесса в компьютерах с динамически управляемыми разделами

Рассмотрим на нашем примере результаты использования предложенной последовательности задач оптимизация вычислительного процесса при выполнении информационно-связанных задач в суперкомпьютерах типа IBM типа Power или z. Цель разработанной методики – обеспечить решение заданного пакета задач с ограничением на время выполнения пакета при минимизации занятых ресурсов. Применительно к компьютерам указано типа и операционным системам, используемым на этих компьютерах, итогом решения задачи является определение числа логических разделов LPAR (по сути – виртуальных машин) и выделение каждому разделу виртуального процессора необходимой доли физического процессора, которую можно назначать с дискретностью 0,1.

Первым шагом является определение длины критического пути, которое задает минимально возможное время реализации задач пакета и возможных резервов времени в выполнении отдельных задач пакета. Исходными данными для решения на этом этапе являются результаты выполнения каждой задачи (после ее отладки) на одном процессоре компьютера (см. п. 3.2.1). Если длина критического пути превышает желаемое время реализации пакета, задачам, лежащим на критическом пути, выделяется большая вычислительная мощность за счет задач, имеющих резерв времени выполнения (см. п. 3.2.2). В нашем примере принято решение оставить первоначально определенный критический путь, равный 24 единицам процессорного времени. Следующий шаг – минимизация количества ресурсов выполнения пакета без увеличения длины критического пути (см. п. 3.2.3). В результате его выполнения учтены резервы времени, имеющиеся у работ, не лежащих на критическом пути, и суммарное количество процессоров, необходимое для выполнения всех работ в расчете на их раздельное выполнение в однопрограммном режиме сокращено с 12 до 9,92 (см. рис. 4). Наконец последний шаг – учет возможности организации мультипроцессорного выполнения пакета задач, представленного сетевой моделью (см. п. 3.2.4). Исходный граф пакета задач представляется в ЯПФ и определяется динамика изменения количества LPAR (3 – 5 – 5 – 5 – 4) и числа долей физических процессоров (2,1 – 3,1 – 3,5 – 3,6 – 3,0) при реализации пакта задач в мультипроцессорном режиме и временная диаграмма выполнения вычислительного процесса (см. рис. 10).

Заключение

Серверные компьютеры (мэйнфреймы) и кластерные системы на базе процессоров высокой производительности типа Power9 и z14 обеспечивают гибкое создание и управление динамическими разделами (виртуальными машинами) при широкой номенклатуре гипервизоров и гостевых операционных систем. Это позволяет эффективно использовать подобные системы для решения сложных задач, требующих больших вычислительных мощностей. Несмотря на высокую вычислительную мощность таких систем одновременное решение сложных пакетов информационно-связанных задач для большого количества пользователей требует такой организации вычислительного процесса, при которой каждый пакет выполняется с требуемыми временными ограничениями при минимально необходимых вычислительных ресурсах. На практике это означает предварительную подготовку информации для диспетчера кластерной системы. Для этого необходимо подготовить план организации вычислительного процесса по критерию минимизации требуемых вычислительных ресурсов с учетом ограничений по директивному времени реализации каждого пакета задач. Учитывая многократный характер реализации пакетов задач научного характера, обработки экспериментов, моделирования и т. п., затраты на такую подготовку будут незначительными. Как показывает рассмотренный в данной статье пример, планирование вычислительного процесса можно провести, используя методы сетевого планирования и управления и методы распараллеливания вычислительного процесса в мультипроцессорных системах. Предложенный подход можно использовать в кластерных системах на базе менее мощных серверах стандартной архитектуры из семейства xSeries и eServer, а также в кластерах, использующих однопроцессорные вычислители без организации динамических разделов.

Список использованных источников

[1] *Андреев А.Н., Воеводин В.В., Жуматий С.А.* Кластеры и суперкомпьютеры -*Dongarra J., Sterling T., Simon H., Strohmaier E.* High-performance computing: clusters, constellations, MPPs, and future directions // Computing in Science & Engineering. 2005. Vol. 7, Issue 2. Pp. 51-59. DOI: 10.1109/MCSE.2005.34

[2] *McLay R., Schulz K.W., Barth W.L., Minyard T.* Best practices for the deployment and management of production HPC clusters // State of the Practice Reports (SC '11). ACM, New York, NY, USA, 2011. Article 9, 11 p. DOI: 10.1145/2063348.2063360

[3] Архитектуры и топологии многопроцессорных вычислительных систем / А. В. Богданов [и др.]. М.: ИНТУИТ. РУ, 2012. 176 с.

[4] *Воеводин В.В., Воеводин Вл.В.* Параллельные вычисления. СПб.: БХВ-Петербург, 2002. 608 с.

[5] *Левин В.К.* Тенденции развития суперкомпьютеров // Computational nanotechnology. 2014. № 1. С. 35-38. URL: https://elibrary.ru/item.asp?id=23786741 (дата обращения: 17.01.2019).

[6] *Sinharoy B., Kalla R.N., Tendler J.M., Eickemeyer R.J., Joyner J.B.* POWER5 system microarchitecture // IBM Journal of Research and Development. 2005. Vol. 49, Issue 4.5. Pp. 505-521. DOI: 10.1147/rd.494.0505

[7] *Кузьминский М.* Пополнение в семействе Power // Открытые системы. СУБД. 2014. № 7. С. 16-18. URL: https://www.osp.ru/os/2014/07/13042909/ (дата обращения: 17.01.2019).

[8] *Sadasivam S.K., Thompto B.W., Kalla R., Starke W.J.* IBM Power9 Processor Architecture // IEEE Micro. 2017. Vol. 37, Issue 2. P.p. 40-51. DOI: 10.1109/MM.2017.40

[9] IBM z/OS Parallel Sysplex Operational Scenarios / F. Kyne [et al.]. IBM, International Technical Support Organization, USA, 2009. 532 p. URL: http://www.redbooks.ibm.com/redbooks/pdfs/sg242079.pdf (дата обращения: 17.01.2019).

[10] Geographically dispersed parallel sysplex support for peer-to-peer VTS // z/OS DFSMS Object Access Method Planning, Installation, and Storage Administration Guide for Tape Libraries. V2R1.0. IBM Corporation, USA, 2013. Pp. 22-23. URL: https://www.ibm.com/support/knowledgecenter/SSLTBW_2.1.0/com.ibm.zos.v2r1.idao300/o3035.htm (дата обращения: 17.01.2019).

[11] *Varrette S., Bouvry P., Cartiaux H., Georgatos F.* Management of an academic HPC cluster: The UL experience // Proceedings of the 2014 International Conference on High Performance Computing & Simulation (HPCS). Bologna, 2014. Pp. 959-967. DOI: 10.1109/HPCSim.2014.6903792

[12] *Creasy R.J.* The Origin of the VM/370 Time-Sharing System // IBM Journal of Research and Development. 1981. Vol. 25, Issue 5. Pp. 483-490. DOI: 10.1147/rd.255.0483

[13] *Александров А.* Спирали аппаратной виртуализации // Открытые системы. СУБД. 2007. № 3. С. 12-18. URL: https://www.osp.ru/os/archive/2007/03/ (дата обращения: 17.01.2019).

[14] *Борзенко А.* Динамические разделы в серверах IBM eServer pSeries // BYTE Россия. 2004. № 7(71). URL: https://www.bytemag.ru/articles/detail.php?ID=8579 (дата обращения: 17.01.2019).

[15] zEnterprise System. Processor Resource/Systems Manager Planning Guide. IBM Corporation, USA, 2011. 236 p. URL: https://www-01.ibm.com/support/docview.wss?uid=isg202e537c11be0929c8525776100663570&aid=1 (дата обращения: 17.01.2019).

[16] IBM HiperSockets Implementation Guide. An IBM Redbooks publication / M. Ebbers [et al.]. IBM, International Technical Support Organization, USA, 2014. 182 p. URL: http://www.redbooks.ibm.com/redbooks/pdfs/sg246816.pdf (дата обращения: 17.01.2019).

[17] *Polychronopoulos C.D., Kuck D.J.* Guided Self-Scheduling: A Practical Scheduling Scheme for Parallel Supercomputers // IEEE Transactions on Computers. 1987. Vol. C-36, Issue 12. Pp. 1425-1439. DOI: 10.1109/TC.1987.5009495

[18] Scheduling and Load Balancing in Parallel and Distributed Systems / B.A. Shirazi, A.R. Hurson, K.M. Kavi (eds.). IEEE Computer Society Press, Los Alamitos, CA, USA, 1995. 520 p.

[19] *Карпов В.Е.* Введение в распараллеливание алгоритмов и программ // Компьютерные исследования и моделирование. 2010. Т. 2, № 3 С. 231-272. URL: https://elibrary.ru/item.asp?id=15621762 (дата обращения: 17.01.2019).

[20] *Барский А.Б.* Параллельное программирование. М.: Интуит, 2016. 345 с.

[21] *Голенко Д.И.* Статистические методы сетевого планирования и управления. М.: Наука, 1968. 400 с.

[22] *Таха Х.А.* Введение в исследование операций. М.: Издательский дом «Вильямс», 2005. 912 с.

[23] *Вагнер Г.* Основы исследования операций. Том 1. М.: Изд. «МИР», 1972. 336 с.

[24] *Коваленко Д.А.* Методы и средства автоматического синтеза параллельных программ на базе теории структурных функциональных моделей // Известия Томского политехнического университета. 2008. Т. 312, № 5. С. 39-44. URL: https://elibrary.ru/item.asp?id=11170114 (дата обращения: 17.01.2019).

Поступила 17.01.2019; принята к публикации 20.02.2019; опубликована онлайн 19.04.2019.

Об авторах:

Назаров Станислав Викторович, профессор, главный специалист, Московский научно-исследовательский телевизионный институт (105094, Россия, г. Москва, ул. Гольяновская, д. 7а, стр. 1); действительный член Международной академии информатизации, доктор технических наук, ORCID: http://orcid.org/0000-0002-7789-1484, nazarov@mniti.ru

Барсуков Алексей Григорьевич, заместитель генерального директора, Московский научно-исследовательский телевизионный институт (105094, Россия, г. Москва, ул. Гольяновская, д. 7а, стр. 1); профессор Академии военных наук, действительный член Международной академии безопасности; Заслуженный изобретатель Российской Федерации, кандидат технических наук, ORCID: http://orcid.org/0000-0002-8787-4987, mniti@mniti.ru

Все авторы прочитали и одобрили окончательный вариант рукописи.

References

[1] Andreev A.N., Voevodin V.V., Zhumatiy S.A. Clusters and supercomputers -Dongarra J., Sterling T., Simon H., Strohmaier E. High-performance computing: clusters, constellations, MPPs, and future directions. *Computing in Science & Engineering*. 2005; 7(2):51-59. (In Eng.) DOI: 10.1109/MCSE.2005.34

[2] McLay R., Schulz K.W., Barth W.L., Minyard T. Best practices for the deployment and management of production HPC clusters. State of the Practice Reports (SC'11). ACM, New York, NY, USA, 2011. Article 9, 11 pp. (In Eng.) DOI: 10.1145/2063348.2063360

[3] Bogdanov A.V., Korkhov V.V., Mareev V.V., Stankova E.N. Architecture and topology of multiprocessor computing systems. M.: Internet University of Information Technology. 2012. 176 pp. (In Russ.)

[4] Voevodin V.V., Voevodin Vl.B. Parallel Computing. SPb.: BHV-Petersburg, 2002. 608 pp. (In Russ.)

[5] Levin V.K. Supercomputer Trends. *Computational nanotechnology*. 2014; 1:35-38. Available at: https://elibrary.ru/item.asp?id=23786741 (accessed 17.01.2019). (In Russ.)

[6] Sinharoy B., Kalla R.N., Tendler J.M., Eickemeyer R.J., Joyner J.B. POWER5 system microarchitecture. *IBM Journal of Research and Development*. 2005; 49(4.5):505-521. (In Eng.) DOI: 10.1147/rd.494.0505

[7] Kuzminskiy M. Addition to the Power Family. *Open Systems. DBMS*. 2014; 7:16-18. Available at: https://www.osp.ru/os/2014/07/13042909/ (accessed 17.01.2019). (In Russ.)

[8] Sadasivam S.K., Thompto B.W., Kalla R., Starke W.J. IBM Power9 Processor Architecture. *IEEE Micro*. 2017; 37(2):40-51. (In Eng.) DOI: 10.1109/MM.2017.40

[9] Kyne F. et al. IBM z/OS Parallel Sysplex Operational Scenarios. IBM, International Technical Support Organization, USA, 2009. 532 pp. Available at: http://www.redbooks.ibm.com/redbooks/pdfs/sg242079.pdf (accessed 17.01.2019). (In Eng.)

[10] Geographically dispersed parallel sysplex support for peer-to-peer VTS. z/OS DFSMS Object Access Method Planning, Installation, and Storage Administration Guide for Tape Libraries. V2R1.0. IBM Corporation, USA, 2013, pp. 22-23. Available at: https://www.ibm.com/support/knowledgecenter/SSLTBW_2.1.0/com.ibm.zos.v2r1.idao300/o3035.htm (accessed 17.01.2019). (In Eng.)

[11] Varrette S., Bouvry P., Cartiaux H., Georgatos F. Management of an academic HPC cluster: The UL experience. Proceedings of the 2014 International Conference on High Performance Computing & Simulation (HPCS). Bologna, 2014, pp. 959-967. (In Eng.) DOI: 10.1109/HPCSim.2014.6903792

[12] Creasy R.J. The Origin of the VM/370 Time-Sharing System. *IBM Journal of Research and Development*. 198; 25(5):483-490. (In Eng.) DOI: 10.1147/rd.255.0483

[13] Alexandrov A. Spirals of hardware virtualization. *Open Systems.DBMS*. 2007; 3:12-18. Available at: https://www.osp.ru/os/archive/2007/03/ (accessed 17.01.2019). (In Russ.)

[14] Borzenko A. Dynamic partitions in IBM eServer pSeries servers. *BYTE Russia*. 2004; 7(71). Available at: https://www.bytemag.ru/articles/detail.php?ID=8579 (accessed 17.01.2019). (In Russ.)

[15] zEnterprise System. Processor Resource/Systems Manager Planning Guide. IBM Corporation, USA, 2011. 236 pp. Available at: https://www-01.ibm.com/support/docview.wss?uid=isg202e537c11be0929c8525776100663570&aid=1 (accessed 17.01.2019). (In Eng.)

[16] Ebbers M. et al. IBM HiperSockets Implementation Guide. An IBM Redbooks publication. IBM, International Technical Support Organization, USA, 2014. 182 pp. Available at: http://www.redbooks.ibm.com/redbooks/pdfs/sg246816.pdf (accessed 17.01.2019). (In Eng.)

[17] Polychronopoulos C.D., Kuck D.J. Guided Self-Scheduling: A Practical Scheduling Scheme for Parallel Supercomputers. *IEEE Transactions on Computers*. 1987; C-36(12):1425-1439. (In Eng.) DOI: 10.1109/TC.1987.5009495

[18] Shirazi B.A., Hurson A.R., Kavi K.M. Scheduling and Load Balancing in Parallel and Distributed Systems. IEEE Computer Society Press, Los Alamitos, CA, USA, 1995. 520 pp. (In Eng.)

[19] Karpov V.E. Introduction to the parallelization of algorithms and programs. *Computer Research and Modeling*. 2010; 2(3):231-272. Available at: https://elibrary.ru/item.asp?id=15621762 (accessed 17.01.2019). (In Russ.)

[20] Barsky A.B. Parallel Programming. M.: Intuit, 2016. 345 pp. (In Russ.)

[21] Golenko D.I. Statistical Methods of Network Planning and Management. Nauka, Moscow, 1968, 400 pp. (In Russ.)

[22] Taha H.A. Operations Research: An Introduction. Prentice Hall, 2006. 838 pp. (In Eng.)

[23] Wagner H.M. Principles of Operations Research With Applications to Managerial Decisions. Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1969. 937 pp. (In Eng.)

[24] Kovalenko D.A. Methods and means of automatic synthesis of parallel programs on the basis of the theory of structural functional models. *Bulletin of the Tomsk Polytechnic University*. 2008; 312(5):39-44. Available at: https://elibrary.ru/item.asp?id=11170114 (accessed 17.01.2019). (In Russ.)

Submitted 17.01.2019; revised 20.02.2019;
published online 19.04.2019.

About the authors:

Stanislav V. Nazarov, Professor, Chief Specialist, Moscow Research TV Institute Joint Stock Company (7a, build. 1 Golyanovskaya St., Moscow 105094, Russia); Member of the International Academy of Informatization, Dr.Sci. (Engineering), ORCID: http://orcid.org/0000-0002-7789-1484, nazarov@mniti.ru

Alexey G. Barsukov, Deputy CEO, Moscow Research TV Institute Joint Stock Company (7a, build. 1 Golyanovskaya St., Moscow 105094, Russia); Professor of Academy of Military Sciences; Full member of the International Academy of Safety, Honored inventor of the Russian Federation, Ph.D. (Engineering), ORCID: http://orcid.org/0000-0002-8787-4987, mniti@mniti.ru

All authors have read and approved the final manuscript.