DOI: 10.18137/RNU.V9187.24.01.P.86 УДК 004.272.2

А.И. Захаров, Г.А. Брякалов, Т.С. Швец, И.С. Петров, С.М. Петренко

ТЕНДЕНЦИИ РАЗВИТИЯ ПАРАЛЛЕЛИЗМА В СРЕДСТВАХ ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ

Аннотация. Статья посвящена рассмотрению основ параллелизма в плане его классического определения и тенденций развития применительно к средствам вычислительной техники. Детально освещены вопросы, связанные с особенностями и направлениями развития параллелизма в архитектуре компьютерных средств, в частности, в средствах суперкомпьютерной техники, а также в языках и средствах программирования. Материал статьи охватывает широкий круг вопросов и может быть полезен для специалистов, интересующихся вопросами параллелизма в плане его практического применения.

Ключевые слова: вычислительная техника, параллелизм в компьютерной архитектуре, языки и средства параллельного программирования, суперкомпьютерная техника.

A.I. Zakharov, G.A. Bryakalov, T.S. Shvets, I.S. Petrov, S.M. Petrenko

TRENDS IN PARALLELISM DEVELOPMENT IN COMPUTING ENGINEERING

Abstract. The article focuses on the consideration of the foundations of parallelism in terms of its classical definition and development trends in relation to computer technology. The issues related to the features and directions of parallelism development in the architecture of computer tools and, in particular, in supercomputing equipment, as well as in languages and programming tools are highlighted in more detail. The article covers a wide range of issues and can be useful for specialists interested in parallelism in terms of its practical application.

Keywords: computing engineering, parallelism in computer architecture, languages and means of parallel programming, supercomputing equipment.

Введение

Параллелизм применительно к вычислительным средствам – это свойство систем, при котором несколько вычислений выполняются одновременно, и при этом, возможно, взаимодействуют друг с другом, что позволяет достигнуть повышения производительности за счет одновременной работы всех элементов структуры, осуществляющих заданное решение. Вместе с тем сама идея параллелизма появились давно. Изначально она внедрялась в самых передовых однопроцессорных компьютерах своего времени. Затем после должной отработки технологии и удешевления производства стала внедряться в компьютеры среднего класса и, наконец, сейчас в полном объеме применяется в рабочих станциях и персональных компьютерах. Анализируя мнения специалистов и данные специальной литературы, можно в общем понятии параллелизма выделить и рассмотреть следующие его составные части:

- параллелизм в архитектуре средств компьютерной техники;
- параллелизм в программировании;
- параллелизм в языках программирования;
- параллелизм в средствах суперкомпьютерной техники.

Захаров Анатолий Иванович
кандидат технических наук, профессор, доцент кафедры математического и программного обеспечения, Военно-космическая академия имени А.Ф. Можайского, Санкт-Петербург. Сфера научных интересов: информационные технологии, организация параллельного вычислительного процесса, параллельное программирование. Автор более 120 опубликованных научных работ. SPIN-код: 7421-0234, AuthorID: 798392.
Электронный адрес: vka_kaf27_1@mil.ru

Брякалов Геннадий Алексеевич
кандидат технических наук, доцент, доцент кафедры математического и программного обеспечения, Военно-космическая академия имени А.Ф. Можайского, Санкт-Петербург. Сфера научных интересов: информационные технологии, информатика, организация вычислительного процесса, программирование. Автор более 120 опубликованных научных работ. SPIN-код: 7548-0940, AuthorID: 798908.
Электронный адрес: vka_kaf27_2@mil.ru

Швец Татьяна Сергеевна
адъюнкт кафедры математического и программного обеспечения, Военно-космическая академия имени А.Ф. Можайского, Санкт-Петербург. Сфера научных интересов: информационные технологии, параллельное программирование, искусственный интеллект. Автор трех опубликованных научных работ.
Электронный адрес: vka_kaf27_3@mil.ru

Петров Иван Сергеевич
адъюнкт кафедры математического и программного обеспечения, Военно-космическая академия имени А.Ф. Можайского, Санкт-Петербург. Сфера научных интересов: информационные технологии, программирование.
Электронный адрес: vka_kaf27_4@mil.ru

Петренко Савелий Максимович
курсант 5-го курса, Военно-космическая академия имени А.Ф. Можайского, Санкт-Петербург. Сфера научных интересов: информационные технологии, программирование.
Электронный адрес: vka_kaf27_5@mil.ru

Параллелизм в архитектуре средств компьютерной техники

О развитии параллелизма, как считают многие специалисты, принято говорить, начиная с истории развития современных суперкомпьютеров. Однако впервые термин «суперкомпьютер» был использован в начале 60-х годов, когда группа специалистов Иллинойского университета (США) под руководством доктора Д. Слотника предложила идею реализации первой в мире параллельной вычислительной системы. Проект, получивший название SOLOMON, базировался на принципе векторной обработки данных, который был сформулирован Дж. фон Нейманом в начале 50-х годов. Дэниел Слотник (1931–1985) – математик и компьютерный архитектор. После окончания в 1952 году Колумбийского университета до 1954 года принимал участие в работе над IAS-машиной в команде Джона фон Неймана в Институте перспективных исследований. В своих работах он впервые выдвинул идеи использования параллелизма в численных вычислениях. Позднее он

был главным архитектором суперкомпьютера ILLIAC-IV, и под его руководством работали такие известные изобретатели, как Джордж Майкл и Сидней Фернбачу [1].

В общеупотребительный лексикон термин «суперкомпьютер» вошел благодаря распространенности компьютерных систем Сеймура Крея, таких как Cray-1, Cray-2. Однако первые суперкомпьютеры Крея, например CDC-6600, созданный в 1963 году, имел только один центральный процессор. Следующий в истории суперкомпьютер CDC-8600 проектировался для использования четырех процессоров с общей памятью, однако CDC-8600 так никогда и не был выпущен, а его разработка была прекращена в 1972 году. Лишь в 1983-м удалось создать работающий суперкомпьютер (CRAY X-MP), в котором использовались два центральных процессора с общей памятью. Сеймур Крей с середины 60-х годов до 1996 года разрабатывал вычислительные машины, которые становились основными вычислительными средствами правительственных, промышленных и академических научно-технических проектов США [2]. В 1994-м можно было свободно приобрести компьютер с двумя процессорами, когда компания ASUS выпустила свою первую материнскую плату с двумя сокетами (разъемами для установки процессоров).

Первым многоядерным процессором массового использования стал POWER-4, выпущенный фирмой IBM в 2001 году, однако широкое распространение многоядерная архитектура получила в 2005-м, когда компании AMD и INTEL выпустили свои первые двухъядерные процессоры. Сегодня параллелизм в архитектуре компьютеров приобрел массовый характер. Все современные микропроцессоры используют тот или иной вид параллельной обработки. В настоящее время большинство выпускаемых микропроцессоров являются многоядерными. Для полной реализации потенциала многоядерной системы программисту необходимо использовать специальные методы параллельного программирования, которые становятся всё более востребованными в промышленном программировании [3; 4].

Программирование задач для таких вычислительных установок получило название параллельного программирования, а соответствующие языки называются языками параллельного программирования.

Под *параллельным программированием* будем понимать создание объектов – параллельных программ, каждая из которых есть организованная совокупность модулей с возможностями одновременного выполнения и взаимодействия.

Параллелизм в программировании

Первые попытки одновременного выполнения задач на компьютере относятся к 1961 году. Именно тогда Том Килберн и Дэвид Ховарт смоделировали параллельное выполнение нескольких программ на компьютере Atlas. Этот метод программирования на основе прерываний стал известен как мультипрограммирование [5].

По мере того как многопрограммные операционные системы продолжали развиваться, стали проявляться слабые стороны этой техники. Выяснилось, что взаимоблокировка задачи может произойти в любое время. Единственная ошибка программирования могла вызвать неожиданное, недетерминированное поведение, причем тестирование этих систем было очень сложным, а иногда невозможным.

Растущая ненадежность многопрограммных операционных систем побудила ученых-компьютерщиков углубиться в параллельное программирование, сформулировать и установить недостающие правила.

Первые подробные работы по концептуальным основам параллельного программирования появились во второй половине 60-х годов. Одним из самых влиятельных участни-

ков этого процесса был Эдсгер Дейкстра, который определил и решил проблему взаимного исключения. Он представил концепцию семафоров для управления доступом к общим ресурсам в параллельных системах [6].

В этой же области проводились исследования Тони Хоура и Пера Бринча Хансена, чьи работы была неоценимы в решении различных вопросов параллельного программирования [7; 8].

Исследователи по всему миру начали воплощать сформулированные идеи в практических рекомендациях, пригодных для создания экспериментальных языков. Основываясь на своих результатах, авторы дополнительно усовершенствовали упомянутые принципы взаимного исключения. Результат этого итеративного процесса лежит в основе современного параллельного программирования.

Параллелизм в языках программирования

Следующим логическим шагом в развитии параллелизма стало создание языков программирования или расширение уже существующих с помощью общей теории параллельного программирования. Средства описания вычислительного процесса, заложенные в большинстве языков программирования, носят, как правило, последовательный характер, однако в связи с созданием и эксплуатацией многопроцессорных систем и многомашинных комплексов начала воплощаться в жизнь идея описания алгоритмов в последовательно-параллельной форме, что позволило явно указывать в программе элементы, допускающие их параллельное выполнение.

Добавление новых правил в язык программирования – сложная задача. Новые правила, то есть их синтаксис и семантика – их значения в контексте, должны быть точно определены, чтобы они были понятны программисту. В конце концов, именно они используют новые возможности языка. Если синтаксис или семантика не ясны, вся концепция неэффективна и в худшем случае бесполезна.

Можно выделить два подхода к проектированию языков параллельного программирования (далее – ЯПП). При так называемом *параллельно-последовательном подходе* ЯПП должен иметь средства для явного указания параллельных ветвей и порядка следования участков параллельности. При выполнении программы управление переходит от одного участка параллельности к другому, каждый раз ветвясь на требуемое число независимых управлений. Переход между участками программы осуществляется только тогда, когда все независимые управления достигнут конца своих ветвей [3; 9; 10].

При *асинхронном подходе* параллельные ветви явно не задаются, и лишь в некоторые моменты времени при выполнении программы выясняется возможность (готовность) выполнения отдельного фрагмента задачи независимо от других фрагментов.

При проектировании ЯПП работа ведется как в направлении создания дополнительных средств в уже существующих последовательных языках, так и в плане создания ЯПП на совершенно новых принципах.

При эксплуатации первых многопроцессорных вычислительных систем (далее – ВС) для повышения эффективности их работы возникала необходимость в параллельных алгоритмах, следовательно, и в языках параллельного программирования, имеющих специальные средства для описания параллельных процессов. Поэтому ЯПП в первую очередь должны предоставлять программистам средства для описания явного и обнаружения скрытого параллелизма. К тому же сегодня уже разработаны параллельные алгоритмы во многих областях обработки информации.

Среди ЯПП можно выделить две группы языков исходя из средств задания взаимодействий между параллельными процессами:

1) взаимодействия фрагментов (процессов) через доступ к общим переменным;

2) взаимодействия посредством передачи межпроцессорных сообщений.

Для решения проблемы диспетчеризации, то есть распределения задач по процессорам (машинам), необходимо разрабатывать методы распознавания и выделения независимых фрагментов программ.

Для формального исследования общих свойств процесса выполнения последовательно-параллельных программ (далее – ПП-программы) необходим математический аппарат описания процесса выполнения программ. В качестве математического аппарата описания программ широко используются схемы программ.

Основные результаты по теории схем программ появились в литературе в 1960-е годы.

Схемы В.В. Мартынюка были введены в 1961 году. Эти схемы оказались удобной моделью для описания ряда алгоритмов построения некоторых множеств вершин схем и их декомпозиций на подсхемах [11].

Схемы С.С. Лаврова были введены в литературу в 1961 году в связи с задачей экономии памяти.

Академик А.П. Ершов рассмотрел вариант схем, в которых информационные связи между входами и выходами операторов задаются не с помощью имен переменных, а явно – в виде дополнительного информационного графа схемы [10; 12].

В ряде работ [13; 14] отражена возможность описания параллельно-последовательных процессов выполнения программ на языке графов.

На языке графов можно выделить самые известные на современном этапе развития схемы алгоритмов, а именно:

- параллельно-последовательный граф (далее – ПП-граф);
- ярусно-параллельный граф;
- граф матричной формы;
- список дуг графа.

Моделирование и расчет характеристик параллельных программ проводятся с целью анализа свойств параллелизма алгоритмов программ, преобразования алгоритмов к параллельным формам и последующего планирования параллельных вычислительных процессов.

Методика выполнения этих мероприятий включает несколько этапов:

- разбиение задачи (алгоритма) на подзадачи (подпрограммы, операции);
- составление списка операций с указанием длины (времени) каждой операции;
- построение графа алгоритма и приведение его к ярусно-параллельной форме (далее – ЯПФ);
- построение матрицы смежности графа по ЯПФ алгоритма для ввода модели программы в ЭВМ;
- расчет операционных характеристик модели программы;
- расчет метрик параллельных вычислений.

Моделирование параллельной программы (далее – ПП) представляет собой процесс создания формального ее представления в виде модели. Наибольшее распространение получили графовые и матричные формы моделей ПП [15].

Модель ПП включает в себя следующие элементы: операции (задачи), связи между операциями, время выполнения операций.

Графовые формы модели ПП основываются на представлении ПП в виде ориентированного взвешенного графа. В качестве моделей ПП используются обычно ориентированные графы с конечным числом вершин и дуг, с которыми сопоставляются соответствующие параметры [16].

Например, для ПП-графа используются следующие основные характеристики:

- ширина *B*;
- число вершин *W*;
- число ветвей *A*;
- длины ветвей l_{ij} (что соответствует времени выполнения каждой ветви-операции).

Наряду с приведенными выше основными характеристиками (ширина, высота и длина ветвей ПП-графа), вводятся производные от них количественные характеристики:

- коэффициент параллельности (*KP*);
- коэффициент ускорения (*KU*);
- ускорение (*US*) [16].

Ярусно-параллельной формой графа называется такое его представление, при котором вершины распределены по уровням (ярусам), причем на каждом ярусе находятся задачи, имеющие непосредственных предшественников на более высоких ярусах и преемников (потомков) – на более низких. Задачи, лежащие на одном ярусе, оказываются независимыми между собой [15].

Максимальное число вершин, расположенных на одном ярусе ЯПФ-графа, называется его шириной. Отметим, что ширина ЯПФ графа определяет максимально необходимое количество вычислителей для достижения максимального ускорения при параллельной обработке алгоритма.

Матричное представление ПП заключается в построении матриц связности составляющих ее операций. Каждый элемент матрицы учитывает связь между соответствующими операциями [15].

По форме схемы ПП-графа или ЯПФ-графа можно рассчитать и оценить основные метрики параллельных вычислений:

Ускорение параллельной обработки

$$S = T_1 / T_n,$$

где T_1 – время последовательного выполнения операций (суммарное время выполнения всех операций алгоритма; T_1 – время параллельного выполнения операций (минимально возможное время, определяемое величиной критического пути графа); n – количество операций (ветвей) в графе-схеме.

Эффективность распараллеливания операций (E)

$$E = S / n.$$

Стоимость параллельных вычислений (C)

$$C = n \times T_n.$$

Следующий этап развития параллелизма начался с перехода от схем графов к описанию их с помощью языков программирования.

Исторически предложение по организации языка высокого уровня для параллельных вычислений впервые сделал К. Айверсон в 1962 году (A Program Language) в проекте языка APL [17]. В этом языке был предложен механизм реализации многомерных векторов, приспособленный к распараллеливанию их в последующей обработке. В результате любое определение функции может быть стандартно распространено на произвольные структу-

ры данных. APL обладает набором базовых средств обработки векторов и манипулирования паспортами структур данных как данными.

Среди языков функционального программирования заметное место занимают языки параллельного программирования, например, язык функционального программирования Sisal (аббревиатура от Streams and Iterations in a Single Assignment Language) [10; 18].

Функциональные языки способствуют разработке корректных параллельных программ. Функциональные программы свободны от побочных эффектов и ошибок, зависящих от реального времени. Это существенно снижает сложность отладки. Функциональные языки уменьшают нагрузку на программиста при программировании, а также в них проще анализировать информационные потоки и схемы управления.

Широкое распространение получили языки, разработанные с учетом требований параллельного программирования, такие как T++, Linda, Chapel, X10, ЯГСПП и др.

Язык ***T++*** является входным языком Т-системы – системы параллельного программирования с открытой архитектурой, поддерживающей автоматическое динамическое распараллеливание программ. Т-система разработана в ИПС РАН (информационно-поисковая система РАН) и в настоящее время развивается совместно в ИПС РАН и МГУ. Синтаксически язык T++ приближен к языку C++.

Язык ***Linda*** работает с параллельной программой, состоящей из множества параллельных процессов. Каждый процесс – это обычная последовательная программа. Все процессы имеют доступ к общей памяти, единицей хранения в которой является *кортеж.* Отсюда происходит и специальное название для общей памяти – *пространство кортежей*. Каждый кортеж – это упорядоченная последовательность значений. Все процессы работают с пространством кортежей по принципу *поместить кортеж, забрать, скопировать* [3].

В основе языка ***Chapel*** лежит параллелизм задач. Параллелизм задач – это форма распараллеливания машинного кода между множеством процессоров в параллельных средах с помощью процессов или потоков. Во время запуска программы создается пул потоков, каждый из которых ожидает задачу из очереди задач. В контексте Chapel задачи – это единицы вычисления, которые могут быть выполнены параллельно с остальными задачами [9].

Язык программирования ***X10***, созданный корпорацией IBM, первыми авторами которого стали Кемаль Эбсиоглу, Вияй Сарасват и Вивек Сакар, был разработан с учетом требований параллельного программирования. По своей сути это «расширенное подмножество» языка программирования Java, обладающее особой дополнительной поддержкой массивов и процессов. X10 использует модель разделяемого глобального адресного пространства [19].

Язык ***ЯГСПП*** графсхемного потокового параллельного программирования ориентирован на модульное потоковое программирование задачи. Был разработан в 2004 году учеными НИУ МЭИ Д.В. Котляровым, В.П. Кутеповым и М.А. Осиповым. Он также может эффективно применяться для программного моделирования распределенных систем, систем массового обслуживания, информационных связей между компонентами, которые структурированы и управляются потоками данных, передаваемых по этим связям.

Потребовалось около 20 лет исследований и экспериментов, чтобы создать надежную поддержку параллелизма в языках программирования.

Параллелизм в средствах суперкомпьютерной техники

Дальнейшие тенденции параллелизма – переход к мультипроцессорным и мультикомпьютерным ВС, в том числе к кластерным ВС и суперЭВМ. Все они относятся к ВС с массовым параллелизмом, программируемой структурой и отказоустойчивой организацией вычислений и характеризуются следующими параметрами.

1. ***Многоядерность*** (Multicore) – явление, когда несколько ядер находятся в корпусе одного процессора. Вычисления могут выполняться на нескольких ядрах одного чипа с вытесняющим разделением времени потоков на одном процессоре либо на физически отдельных процессорах.

2. ***Технологии с явным параллелизмом команд***. Архитектура всех современных 64-разрядных процессоров Intel построена на использовании технологии EPIC (Explicitly Parallel Instruction Computing), которая позволяет явно указывать на параллелизм команд. Параллелизм сегодня стал главным ресурсом наращивания вычислительной мощности компьютеров. Существенным препятствием к параллельному выполнению программ являются точки ветвления, в которых решается вопрос, по какому из нескольких возможных путей пойдет выполнение программы после этой точки. Чем более совершенным является механизм предсказания ветвлений, тем лучше может быть распараллелена программа. После того как программа достигнет точки ветвления, можно уже окончательно выбирать результаты, полученные по одной из возможных ветвей. Сущность EPIC заключается в том, что блок предсказания ветвлений выносится из аппаратной логики процессора в компилятор. Анализируя программу, компилятор сам определяет параллельные участки и дает процессору явные инструкции по их выполнению – отсюда и следует название архитектуры EPIC. При этом следует иметь в виду, что изменить устройство процессора в уже работающей вычислительной системе невозможно. Установленный процессор можно только заменить новым подходящим для данной системы процессором. Поэтому архитектура EPIC позволяет более эффективно модифицировать работающие вычислительные системы за счет установки более совершенных версий компилятора.

3. ***Многопоточность*** (Thread Level Parallelism – TLP) – имеет место тогда, когда в каждом ядре процессора выполняется одновременно несколько потоков, конкурирующих между собой. Например, в ядре Pentium 4 на разных стадиях выполнения может одновременно находиться до 126 микроопераций [20].

Главная задача в развитии вышеперечисленных направлений – существенное повышение производительности вычислительных систем. Без развития параллельных технологий решить задачу повышения производительности вычислительных систем в настоящее время не представляется возможным, поскольку современные технологии микроэлектроники подошли к технологическому барьеру, препятствующему дальнейшему существенному увеличению тактовой частоты работы процессора.

Развитие суперкомпьютеров обусловлено существованием особо важных задач, главным требованием к решению которых является скорость вычислений в реальном времени. Нужная скорость достигается ценой весьма сложной организации связей между синхронизированными на уровне аппаратуры процессорами. Активно используется локальная память процессоров, приспособленная к быстрым матричным вычислениям и однородной обработке информации [21].

При программировании целенаправленно выделяются конвейерные процессы, приспособленные к минимизации хранения промежуточных результатов. Разработаны специ-

альные архитектуры сверхдлинных команд с внутренним совмещением микроопераций. Архитектура со сверхдлинными командами (Very Long Instruction Word – VLIW) известна с начала 1980-х годов. Специфику языков параллельного программирования для высокопроизводительных вычислений на базе суперкомпьютеров и многоядерных архитектур можно рассмотреть на базе показателей эффективности параллельной программы [22].

Для оценки эффективности параллельной программы принято сравнивать показатели скорости исполнения этой программы при ее запуске на нескольких идентичных вычислительных системах, которые различаются только количеством центральных процессоров или ядер. На практике измерения проводятся на одной многопроцессорной (многоядерной) вычислительной системе, в которой искусственно ограничивается количество процессоров (ядер), задействованных в вычислениях. Это обычно достигается одним из следующих способов:

- установка аффинности процессоров (ядер);
- виртуализация процессоров (ядер);
- управление количеством нитей.

Установка аффинности – это указание операционной системе запускать указанный поток (процесс) на явно заданном процессоре (ядре).

Виртуализация процессоров (ядер) – это возможность «выделить» в создаваемой виртуальной машине не все присутствующие в системе процессоры (ядра), а только часть из них [23].

Наиболее важным при разработке параллельного алгоритма является деление на компьютеры с общей и распределенной памятью. Для компьютеров с общей памятью пользователю не нужно заботиться о распределении данных – достаточно предусмотреть лишь затраты на выбор необходимых данных из этой памяти.

При реализации параллельного алгоритма на компьютерах с распределенной памятью необходимо продумать рациональную с точки зрения потерь на обмен данными схему их размещения.

Если говорить о параллелизме в средствах суперкомпьютерной техники, то на современном этапе ее развития различаются следующие типы параллельных машин:

- машины по типу памяти (общая или распределенная);
- машины по типу управления, по классификации Флинна – машины типа SIMD (одна инструкция – множество данных) и типа MIMD (много инструкций – много данных).

В машине SIMD управляющий узел один, он отправляет инструкцию всем остальным процессорам. Каждый процессор имеет свой набор данных для работы. В машине MIMD каждый процессор имеет свой собственный управляющий узел и может выполнять разные инструкции с разными данными. SIMD-машины обычно используются для конкретных задач, требующих, как правило, не столько гибкости и универсальности вычислительной машины, сколько самой вычислительной производительности. MIMD-машины обладают более широким функциональным диапазоном. MIMD-машина с общей памятью – это, например, двуядерный процессор (портативный компьютер типа ноутбука).

Симметричные мультипроцессоры (Symmetric Multiprocessors – SMP) состоят из совокупности процессоров, имеющих разделяемую общую память с единым адресным пространством и функционирующих под управлением одной операционной системы. Недостаток SMP в том, что число процессоров, имеющих доступ к общей памяти, нельзя сделать бо́льшим.

Кластеры образуются из вычислительных модулей любого из рассмотренных выше типов, объединенных системой связи или посредством разделяемой внешней памяти. Могут использоваться как специализированные, так и универсальные сетевые технологии. Это направление, по существу, является комбинацией предыдущих двух.

Заключение

Поводя итоги вышесказанному, можно отметить следующее.

1. Понятие физической параллельности применимо, когда для выполнения параллельных модулей используются несколько процессоров. Если параллельные модули выполняются на одном процессоре, применяется термин «логическая параллельность». Логическая параллельность – базовая модель для любого вида параллельности [24].

2. Параллельное выполнение может осуществляться на уровне подпрограмм, модулей или на уровне операторов.

3. Параллельные языки должны обеспечивать две основные возможности: взаимно исключающий доступ к совместно используемым структурам данных и взаимодействие задач.

4. Мониторы – это абстракции данных, обеспечивающие естественный способ взаимно исключающего доступа к данным, совместно используемым несколькими задачами. В параллельном программировании монитор – это конструкция синхронизации, которая позволяет потокам иметь как взаимное исключение, так и возможность «ожидания» [25].

5. При передаче сообщений основными параллельными модулями являются задачи, взаимодействующие между собой с помощью механизма взаимодействия, представляющего собой синхронную передачу сообщений.

6. Параллельные вычисления – это способ организации компьютерных вычислений, при котором программы разрабатываются как набор взаимодействующих вычислительных процессов, работающих параллельно (одновременно). Термин охватывает совокупность вопросов параллелизма в программировании, а также создание эффективно действующих аппаратных реализаций.

7. В некоторых параллельных системах программирования передача данных между компонентами скрыта от программиста, тогда как в других она должна указываться явно.

8. Теория параллельных вычислений составляет раздел прикладной теории алгоритмов.

Литература

1. *Schneck P.B.* Dedication. Daniel L. Slotnick, 1931 to 1985 // The Journal of Su-percomputing. 1987. Vol. 1. Pp. 5–6. DOI: https://doi.org/10.1007/BF00138601

2. *Murray C.J.* The Supermen: The Story of Seymour Cray and the Technical Wizards Behind the Supercomputer. Wiley, 1997. 232 p. ISBN 9780471048855.

3. *Воеводин В.В.* Параллельная обработка данных : Курс лекций // Parallel.ru. Лаборатория параллельных информационных технологий НИВЦ МГУ. 2000. URL: https://parallel.ru/vvv/ (дата обращения: 23.01.2023).

4. *Гафаров Ф.М., Галимянов А.Ф.* Параллельные вычисления: учеб. пособие. Казань : Изд-во Казан. ун-та, 2018. 149 с.

5. *Kilburn T., Howarth D.J., Payne R.B., Sumner F.H.* The Manchester University Atlas Operating System. Part I: Internal Organization // The Computer Journal. 1961. Vol. 4. Issue 3. Pp. 222–225. DOI: https://doi.org/10.1093/comjnl/4.3.222

6. *Дейкстра Э.* Дисциплина программирования / Пер. с англ. И.Х. Зусман и др.; под ред. Э.З. Любимского. М. : Мир, 1978. 320 с.
7. *Hoare C.A.R.* Parallel Programming: An Axiomatic Approach // Bauer F.L., et al. Language Hierarchies and Interfaces. Lecture Notes in Computer Science. Vol. 46. Springer, Berlin, Heidelberg, 1976. Pp. 11–42. DOI: https://doi.org/10.1007/3-540-07994-7_47
8. *Hansen P.B.* Distributed Processes: A Concurrent Programming Concept // Hansen P.B. (Ed) The Origin of Concurrent Programming. New York, NY : Springer, 1978. Pp. 444–463. DOI: https://doi.org/10.1007/978-1-4757-3472-0_17
9. *Гергель В.П.* Современные языки и технологии параллельного программирования: Учебник для вузов. М. : Изд. МГУ, 2012. ISBN 978-5-211-06380-8. EDN QMXKOJ.
10. *Ершов А.П.* Избранные труды / Сибирское отделение РАН; Институт систем информатики. Новосибирск : Наука, 1994, 416 с. ISBN 5-02-030344-5. EDN YJPWWH.
11. *Касьянов В.Н.* Оптимизирующие преобразования программ. М.: Наука, 1988. 334 с. ISBN 5020137782.
12. *Евреинов Э.В., Косарев Ю.Г.* Однородные универсальные вычислительные системы высокой производительности. Новосибирск : Наука, 1966. 308 с.
13. *Корниенко Н.М.* Комбинаторные алгоритмы на классе графов // Известия Национальной академии наук Беларуси. Серия физико-технических наук. 1984. Вып. 3. С. 109–111.
14. *Свами М., Тхуласираман К.* Графы, сети и алгоритмы / Пер. с англ. М.В. Горбатовой и др. М. : Мир, 1984. 456 с.
15. *Басыров А.Г.* Вычислительные системы: практикум / Сост.: А.Г. Басыров, А.С. Дудкин, И.В. Захаров, А.С. Швецов, А.О. Шушаков. СПб. : ВКА им. А.Ф. Можайского, 2016. 118 с.
16. *Захаров А.И., Пореченский М.А., Чмыхова Я.В.* Имитационная модель исследования влияния распараллеливания информационных процессов на рост производительности многоядерных вычислительных систем // Сборник алгоритмов и программ типовых задач. СПб. : ВКА имени А.Ф. Можайского. 2017. Вып. 34. С. 173–181.
17. *Iverson K.E.* A personal view of APL // IBM Systems Journal. 1991. Vol. 30. No. 4. Pp. 582–593. DOI: 10.1147/sj.304.0582
18. *Касьянова Е.В., Касьянова Е.В.* Язык программирования Cloud Sisal / Сибирское отделение РАН. Новосибирск : Изд. Института систем информатики имени А.П. Ершов., 2018. 232 с. EDN SZQQEW.
19. *Biever C.* Chip revolution poses problems for programmers // New Scientist. 2007. No. 2594. URL: https://www.newscientist.com/article/mg19325946-000-chip-revolution-poses-problems-for-programmers/ (дата обращения: 23.01.2023).
20. *Кузьминский М.* Athlon: от микропроцессоров к материнским платам // Открытые системы. 2000. №1-2. С. 8–13. URL: https://www.osp.ru/os/2000/01-02/178169?ysclid=ltywxu1ktf691003635 (дата обращения: 23.01.2023).
21. *Golub G.H., Van Loan C.L.F.* Matrix Computations. 4th edition. Baltimor : Johns Hopkins University Press, 2013. 458 p. ISBN: 978-1-4214-0794-4.
22. *Соснин В.В., Балакшин П.В., Шилко Д.С., Пушкарев Д.А., Мишенёв А.В., Кустарев П.В., Тропченко А.А.* Введение в параллельные вычисления : Учебно-метод. пособие. СПб. : Университет ИТМО, 2023. 128 с. URL: https://books.ifmo.ru/file/pdf/3230.pdf (дата обращения: 23.01.2023).
23. *Соснин В.В., Балакшин П.В.* Введение в параллельные вычисления. СПб. : Университет ИТМО, 2015. 51 с. URL: https://books.ifmo.ru/file/pdf/1900.pdf (дата обращения: 23.01.2023).

24. *Clark K.L., Gregory St.* Parlog Parallel Programming in Logic // ACM Transactions on Programming Languages and Systems. 1985. Vol. 8. No. 1. Pp. 1–49. DOI: https://doi.org/10.1145/5001.5390
25. *Hansen P.B.* The Architecture of Concurrent Programs. Prentice-Hall, 1977. 317 p. ISBN 0130446289.

References

1. Schneck P.B. (1987) Dedication. Daniel L. Slotnick, 1931 to 1985. *The Journal of Su-percomputing*. Vol. 1. Pp. 5–6. DOI: https://doi.org/10.1007/BF00138601
2. Murray C.J. (1997) *The Supermen: The Story of Seymour Cray and the Technical Wizards Behind the Supercomputer*. Wiley. 232 p. ISBN 9780471048855.
3. Voevodin V.V. (2000) Parallel data processing: Course of lectures. *Parallel.ru. Parallel Information Technology Laboratory of Lomonosov MSU research computing centre*. URL: https://parallel.ru/vvv/ (accessed 23.01.2023). (In Russian).
4. Gafarov F.M., GalimyanovA.F. (2018) *Parallel'nye vychisleniya* [Parallel computing] : Training manual. Kazan : Kazan State University Publ. 149 p. (In Russian).
5. Kilburn T., Howarth D.J., Payne R.B., Sumner F.H. (1961) The Manchester University Atlas Operating System. Part I: Internal Organization. *The Computer Journal*. Vol. 4. Issue 3. Pp. 222–225. DOI: https://doi.org/10.1093/comjnl/4.3.222
6. Dijkstra E.W. (1976) *A discipline of programming*. Prentice-Hall. 217 p. ISBN 013215871X. (Russian edition: transl. by I.Kh. Zusman, ed. by E.Z. Lyubimskiy, Moscow : Mir Publ., 1978, 320 p.).
7. Hoare C.A.R. (1976) Parallel Programming: An Axiomatic Approach. In: Bauer F.L., et al. *Language Hierarchies and Interfaces. Lecture Notes in Computer Science*. Vol. 46. Springer, Berlin, Heidelberg. Pp. 11–42. DOI: https://doi.org/10.1007/3-540-07994-7_47
8. Hansen P.B. (1978). Distributed Processes: A Concurrent Programming Concept. In: Hansen P.B. (Ed) *The Origin of Concurrent Programming*. New York, NY : Springer. Pp. 444–463. DOI: https://doi.org/10.1007/978-1-4757-3472-0_17
9. Gergel' V.P. (2016) *Sovremennye yazyki i tekhnologii parallel'nogo programmirovaniya* [Modern languages and parallel programming technologies] : Textbook for universities. Moscow : Moscow State University Publ. ISBN 978-5-211-06380-8. (In Russian).
10. Ershov A.P. (1994) *Izbrannye Trudy* [Selescted Works]. Novosibirsk : Nauka Publ. 416 p. ISBN 5-02-030344-5. (In Russian).
11. Kasyanov V.N. (1988) *Optimiziruyushie preobrazovaniya program* [Programme optimizing transformation]. Moscow : Nauka Publ. 334 p. ISBN 5020137782. (In Russian).
12. Evreinov E.V., Kosarev Yu.G. *Odnorodnye universal'nye vychislitel'nye sistemy vysokoi proizvoditel'nosti* [Homogeneous high-performance computing systems]. Novosibirsk : Nauka Publ. 1966, 308 p. (In Russian).
13. Kornienko N.M. (1984) Combinatorial Algorithms on a Graph Class. *Proceedings of the National Academy of Sciences of Belarus. Physical-technical series*. No. 3. Pp. 109–111. (In Russian).
14. Swami M., Thulasiraman K. (1981) *Graphs, Networks and Algorithms*. Wiley. 592 p. ISBN 0471035033. (Russian edition: transl. by M.V. Gorbatova, Moscow : Mir Publ., 1984. 456 p.).
15. Basyrov A.G., Dudkin A.S., Zakharov I.V., Shvetsov A.S., Shushakov A.O. (2016) *Vychislitel'nye sistemy : praktikum* [Computer Systems : Workshop]. St. Petersburg : A. Mozhaysky Military Space Academy Publ. 118 p. (In Russian).
16. Zakharov A.I., Porechenskiy M.A., Chmykhova Ya.V. (2017) Simulation model of research of influence of parallelization of information processes on growth of performance of multicore computing sys-

tems. In: *Sbornik algoritmov i programm typovykh zadach* [A collection of algorithms and applications]. St. Petersburg : A. Mozhaysky Military Space Academy Publ. Issue 34. Pp. 173–181. (In Russian).
17. Iverson K.E. (1991) A personal view of APL. *IBM Systems Journal*. Vol. 30. No. 4. Pp. 582–593. DOI: 10.1147/sj.304.0582
18. Kasyanov V.N., Kasyanova E.B. (2018) *Yazyk programmirovaniya Cloud Sisal* [Programming Language Cloud Sisal]. Siberian Division of the Russian Academy of Sciences. Novosibirsk : A.P. Ershov Institute of Informatics Systems. 232 p. (In Russian).
19. Biever C. (2007) Chip revolution poses problems for programmers. *New Scientist*. Issue 2594. URL: https://www.newscientist.com/article/mg19325946-000-chip-revolution-poses-problems-for-programmers/ (accessed 23.01.2023).
20. Kuzminskyi M. (2000) Athlon: From microprocessors to motherboards. *Otkrytye sistemy*. No. 1-2. Pp. 8–13. URL: https://www.osp.ru/os/2000/01-02/178169?ysclid=ltywxu1ktf691003635 (accessed 23.01.2023). (In Russian).
21. *Golub G.H., Van Loan C.L.F.* Matrix Computations. 4th edition. Baltimor : Johns Hopkins University Press, 2013. 458 p. ISBN: 978-1-4214-0794-4.
22. Sosnin V.V., Balakshin P.V., Shilko D.S., Pushkarev D.A., Mishenev A.V., Kustarev P.V., Tropchenko A.A. (2023) *Vvedenie v parallel'nyevychisleniya* [Introduction to Parallel Computing] : Educational-methodical manual. St. Petersburg : ITMO University Publ. 128 p. URL: https://books.ifmo.ru/file/pdf/1900.pdf (accessed 23.01.2023). (In Russian).
23. Sosnin V.V., Balakshin P.V. (2015) *Vvedenie v parallel'nyevychisleniya* [Introduction to Parallel Computing]. St. Petersburg : ITMO University Publ. 51 p. URL: https://books.ifmo.ru/file/pdf/1900.pdf (accessed 23.01.2023). (In Russian).
24. Clark K.L., Gregory St. (1985). Parlog Parallel Programming in Logic. *ACM Transactions on Programming Languages and Systems*. Vol. 8. No. 1. Pp. 1–49. DOI: https://doi.org/10.1145/5001.5390
25. Hansen P.B. (1977) *The Architecture of Concurrent Programs*. Prentice-Hall. 317 p. ISBN 0130446289.