ГЛАВА 4

ОБЩИЕ ПРИНЦИПЫ И ОСОБЕННОСТИ ОРГАНИЗАЦИИ СИСТЕМЫ ПЛАНИРОВАНИЯ В РСОД

ПРЕДИСЛОВИЕ

Технологические достижения последних лет позволяют создавать мощные вычислительные системы, в которых можно обработать одновременно большой поток разнотипных задач. В таких системах, построенных на высокопроизводительных элементах, эффективность обработки высокораспараллеленных задач во многом зависит от организации вычислений, то есть от того, как и каким образом построено управление вычислениями и решены задачи распределения работ и ресурсов системы. При этом задача планирования распределения заявок (заданий, процессов...) на ресурсы параллельной вычислительной системы (ПВС) в многопроцессорных, многомашинных комплексах и сетях (локальных или глобальных) является одной из самых труднорешаемых задач.

4.1. ОБЪЕКТЫ ПЛАНИРОВАНИЯ В ПВС

В общем виде вычислительную систему и вычислительный узел (как объект планирования) можно рассматривать с различной степенью детализации как совокупность ресурсов. Кроме этого, объектами планирования вычислений или *"заданий"* на ресурсы могут быть также и программы, процедуры, параллельные участки программ, отдельные блоки команд, команды и отдельные макро — или микрооперации. Таким образом, объектом системы планирования *ПВС* может быть либо множество ресурсов ВС, либо множество заданий, поступающих на вход ВС, либо множество отношений задание-ресурс. Объект планирования может меняться в зависимости от целей задачи планирования: либо это загруженность оборудования, либо эффективность обслуживания поступающих на вход ВС заданий, либо и то и другое.

Расширение понятия "*ресурса*" в системах параллельной обработки информации и функциональных возможностей вычислительного узла до возможностей вычислительной системы требует изменения подходов к организации и планированию вычислительного процесса.

Процесс планирования распределения ресурсов и задач, в целом, является отображением трех макрофакторов: *система* (архитектура, структура и характеристики), *поток заданий* (требования и структура) и *время* (в системах реального времени или при ограниченных ресурсах системы) в расписание выполнения множества заданий — *потока заданий* в пространственно (*система)* — временных (*время*) координатах. Кроме этого, каждый из выше обозначенных макро факторов рекурсивно представляет собой совокупность мини — и

микрофакторов при детализации расписания в пространстве ресурсов ВС и времени выполнения этого расписания на каждом ресурсе в отдельности. Данная задача, с учетом всех требований и параметров ВС и/или входного потока заданий, является критически трудной. Она представляет собой совокупность нескольких труднорешаемых задач, принадлежащих к классу *NP-полных* [11]. При разработке стратегии решения данной задачи применяют следующие группы критериев оптимизации:

1. *Критерии оптимизации относящиеся к вычислительной системе*: процесс вычисления необходимо планировать так, чтобы обработка потока задач была оптимальна по эффективности использования ресурсов (процессоров, памяти, каналов связи), времени нахождения задач в системе, времени завершения задач, времени, затрачиваемого на выполнение пересылок, и т.д.
2. *Критерии оптимизации самого процесса планирования*: качество планирования, время решения задачи планирования или динамичность планирования в зависимости от того, что в данной ВС является критичным.

4.1.1. Роль процесса планирования и организации вычислений в распределенных системах обработки данных

Анализ тенденций развития распределенных вычислительных систем позволяет определить некоторые особенности не только их структурной организации, но и изменение подходов к организации вычислительного процесса в рамках как самой вычислительной системы, так и в отдельном вычислительном узле. Как уже упоминалось, если рассматривать с различной степенью детализации вычислительную систему и вычислительный узел как совокупность неоднородных ресурсов, то объекты планирования или заявки могут быть представлены как программы, процедуры, параллельные участки программ, отдельные блоки команд, команды и отдельные макро — или микрооперации. Сложность решения задач организации и планирования вычислительного процесса в такой системе определяется, с одной стороны, характеристиками вычислительной системы, в которой необходимо организовать вычислительный процесс, с другой — описанием входного потока заданий в вычислительную систему. Сложность описания входного потока работ и среды, где они должны быть выполнены, определяет сложность решения задач эффективной организации и планирования вычислительного процесса в вычислительной системе.

Каждой вычислительной системой управляет операционная система (ОС), в функции которой входят: управление процессорами, памятью, внешними устройствами, данными, инициализацией программ и их связей по управлению и данным, обработка прерываний, синхронизация процессов и их доступа к ресурсам. В общем виде место системы планирования в организации вычислительного процесса как интерфейса между входным потоком заданий и ресурсов ВС представлено на рис.4.1.

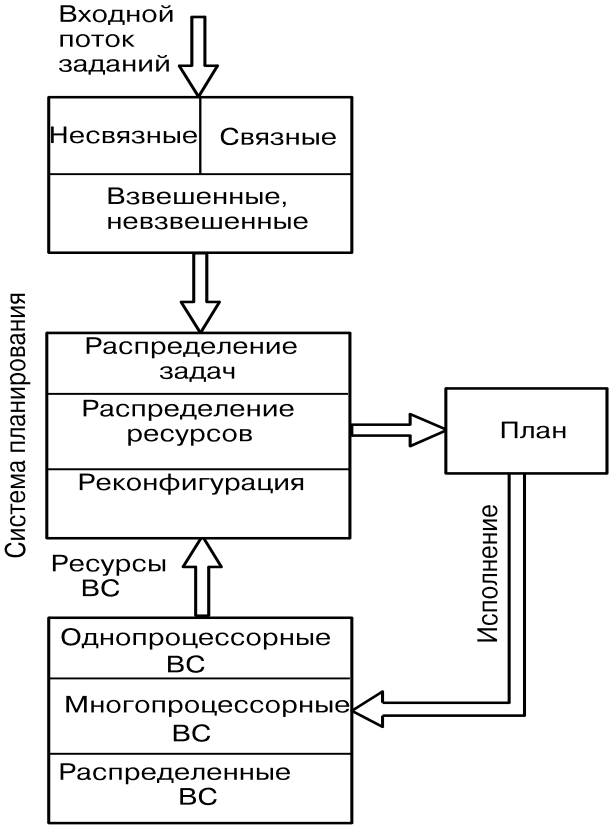

Рис. 4.1

Для изучения вопросов организации, диспетчеризации и планирования вычислительного процесса в системах массового распараллеливания и распределенных вычислительных системах целесообразно рассмотреть влияние архитектурных особенностей этих систем на решение упомянутых задач в общем виде без уточнения способов их реализации на конкретных моделях ВС. Эффективность организации вычислительного процесса в параллельных системах обработки информации зависит от:

* способа соединения исполнительных устройств;
* возможности обмена информацией между исполнительными устройствами;
* принятого способа обмена информацией между исполнительными устройствами;

* используемого способа хранения данных, команд и доступа к ним;
* организации доступа и метода управления памятью в ВС;
* принятого способа управления вычислительной системой в целом и каждым отдельным вычислительным элементом в частности.

Для детального выделения действий, выполняемых при планировании, рассмотрим схему прохождения работ через ВС. При этом будем отслеживать прохождение задания с момента его ввода до полного завершения (вывода), фиксируя моменты изменения формы представления задания в ВС (Mi) и интервалы времени преобразования задания из одного состояния в другое (рис 4.2.)

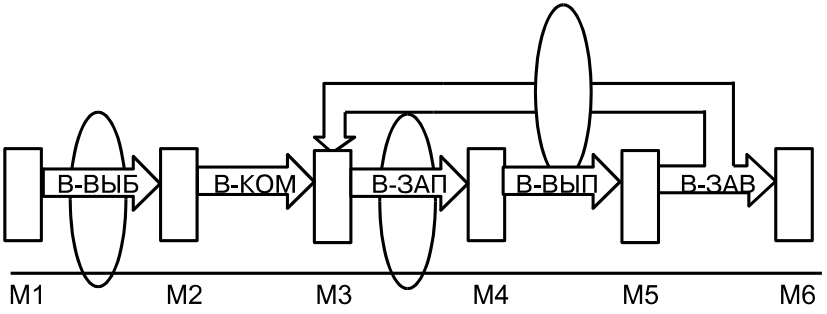

Рис. 4.2

Где:

- M1 — момент запроса: момент фиксации системой заявки/заявок на обслуживание;
- В-ВЫБ — время выборки: время предварительной обработки входного потока заявок;
- M2 — момент выборки: момент принятия решения об активизации заявок или отсрочке их выполнения;
- В-КОМ — время компиляции: время подготовки исходного описания задания-заявки и преобразования его в форму, необходимую для выполнения в параллельной системе;
- M3 — момент компоновки: момент, когда компиляция закончена, и входные задания переводятся в ранг "процессов", для которых ВС должна выделить ресурсы;
- В-ЗАП — время запуска: время, в течение которого выполняются действия, требуемые для запуска (инициирования) готовых процессов, — выделение ресурсов и перевод процессов в активное состояние;
- M4 — момент запуска: момент инициирования (активизации) задачи-процесса после распределения ресурсов (устройств, данных, памяти), который выполняется во время В-ЗАП.
- В-ВЫП — время выполнения: период времени, когда задания в системе

активны, то есть выполняются.

- М5 — момент завершения: момент фиксации завершения задания (естественного или аварийного);
- В-ЗАВ — время завершения: время, в течение которого диспетчер ОС определяет, выполнено ли задание или его нужно продолжать выполнять. В последнем случае он ставит задание обратно в очередь на выполнение;
- М6 — момент выхода: момент, когда ОС считает задание полностью выполненным или устанавливает, что система не имеет возможности его дальнейшего выполнения, тогда оно удаляется из системы;
- Эллипс означает выполнение действий планировщика системы.

Решение задач планирования захвата ресурсов задачами — процессами выполняется в самых ключевых точках прохождения процессов через вычислительную систему. В процессе планирования можно условно выделить несколько уровней, каждый из которых выполняется своим планировщиком соответствующего типа.

4.1.2. Общая модель процесса планирования

Ресурсы, предоставляемые компьютерной системой, обычно подразделяются на две категории: физические ресурсы и логические ресурсы. В распределенной системе все ресурсы распределяются еще и в пространстве — территориально. И физические, и логические ресурсы должны быть управляемыми. Задачи управления ресурсами можно определить следующим образом:

* возможность планирования выполнения работ, их идентификации и организации обеспечения ресурсами: при этом уточняются места размещения ресурсов, доступность ресурсов и их стоимость;
* возможность управления использованием ресурсов и доступа к ним, согласования распределения ресурсов, оптимизации их использования и взаимодействия;
* возможность проверки доступности ресурсов для управления или контроля правильности их функционирования; и, если ресурсы не доступны, своевременность формирования соответствующих сигналов, указывающих на отказ в доступности ресурсов.

Под ресурсом в ВС в настоящее время подразумевают не только техническое обеспечение ВС, но и время работы процессора, состав программного обеспечения, наличие данных в узле. При таком понятии ресурса, вычислительные системы, традиционно считавшиеся однородными по структуре (архитектуре), становятся *неоднородными* с точки зрения системы планирования

Классическая трехуровневая схема [2, 24, 26] планирования вычислительного процесса (рис. 4.3) приемлема в ВС с ограниченными ресурсами и предусматривает использование в полной мере механизма вынужденного прерывания для реализации переключений между различными работами. Для организации вычислительного

процесса используются режимы мультипрограммирования, квантования, разделения времени и различные дисциплины обслуживания очередей.

1	Предварительное входное планирование исходного потока заявок, претендующих на захват ресурсов вычислительной системы	Статическое или динамическое планирование
2	Планирование потока процессов, претендующих на захват времени процессора/процессоров вычислительной системы	Динамическое планирование
3	Планирование очереди процессов на выделение времени процессора/процессоров вычислительной системы	

Рис. 4.3. Классическая трехуровневая модель планирования

Следует отметить, что данная схема планирования отражает временную сущность планирования, т.е. планирование во времени. Поступающие на вход вычислительной системы задачи (заявки) проходят несколько стадий планирования, а точнее отбора — определения очередности решения задач в ВС, и только на последней стадии в соответствии с принятой в ВС дисциплиной обслуживания, им предоставляется время процессора.

При таком подходе усложнен механизм синхронизации процессов в системе и повышена вероятность "дедлоков". Трехуровневая модель системы планирования характерна для однопроцессорных систем, работающих в однопрограммном или многопрограммном режимах. При этом планирование выполняется, в основном, по *временной* координате, и время процессора является разделяемым ресурсом. При появлении многопроцессорных систем, предназначенных для обработки множества несвязных и/или взаимосвязанных процессов в вычислительной среде, планирование должно выполняться в пространственно—временных координатах с привлечением аппарата теории массового обслуживания, очередей и комбинаторной оптимизации. Следует отметить, что появление ВС, в которых количество используемых ресурсов может меняться в зависимости от требований входного потока задач, повлияло на способы решения задач управления и диспетчеризации. При организации вычислительного процесса для несвязных процессов в таких системах можно исключить временную координату и решать задачу планирования только в пространстве, что несколько облегчает ее решение, т.к. заявкам (процессам) система выделяет столько вычислительных ресурсов, сколько им необходимо.

Исходя из этой схемы планирования, можно сформулировать общие требования, предъявляемые к системе планирования:

- обеспечение принципа равного доступа — дисциплина планирования равного доступа характеризуется тем, что всем процессам одинаково выделяется ресурс и нет процесса, находящегося в режиме бесконечного откладывания;

- обеспечение максимальной производительности ВС — дисциплина планирования обеспечивает обслуживание как можно большего количества процессов в единицу времени;
- максимальное количество интерактивных пользователей получило приемлемые времена ответов;
- предсказуемость — любое задание будет выполнено за приблизительно то же количество времени и с приблизительно теми же затратами, несмотря на загрузку системы;
- минимизация затрат — следует отметить, что это не всегда рассматривается как одно из самых важных требований. Затраты иногда рассматриваются как ненужное требование. Но минимизация затрат системных ресурсов на решение задач планирования может значительно улучшить общую работу системы;
- использование балансного принципа планирования — механизмы планирования, сохраняющие ресурсы системы занятыми;
- достижение баланса между ответом и применением, — лучшая гарантия хорошего времени ответов, имея возможность эффективного использования ресурсов. Цена, заплаченная за эту стратегию, превышает плохое использование ресурса. В системах реального времени быстрые ответы существенны, а использование ресурсов имеет меньшую значимость. В других типах систем экономические соображения часто вынуждают делать эффективным использование ресурса;
- избежание бесконечного откладывания — во многих случаях бесконечное откладывание может быть хуже, чем тупик. Избежание бесконечного откладывания лучше всего реализуется динамическим изменением приоритета, т.е. чем дольше процесс ожидает ресурс, тем выше его приоритет. В итоге, приоритет становится настолько высоким, что процессу будет выделен ресурс;
- вынужденные приоритеты — в средах, в которых процессам присваиваются приоритеты, механизм планирования предпочтителен для процессов с высоким приоритетом;
- снижение эффективности под большими загрузками — механизм планирования не разрушается под весом большой системной загрузки. Он будет предотвращать чрезмерную загрузку, не позволяя создавать новые процессы, когда загрузка велика.

Многие из этих требований конфликтуют с другими, делая при этом планирование сложной задачей.

4.1.3. Дисциплины обслуживания заявок

Эффективность работы вычислительной системы зависит не только от собственной эффективности алгоритмов обработки информации и технических характеристик ВС, но и от принятых в системе правил выполнения работ, приема и обработки запросов пользователей.

Эффективность методов обслуживания определяется возможностью задержки или потери заявки до обработки, а также временем нахождения заявки в системе. В зависимости от типа системы управления и диспетчеризации, задержка заявок может учитываться по общему среднему времени задержки или по допустимому времени ожидания.

Во время изучения дисциплин обслуживания заявок предполагается, что процессы ввода и обслуживания являются независимыми. Заявка, которая поступает в систему, начинает обслуживаться немедленно, если в этот момент ресурс для ее обслуживания свободен.

Если ресурс занят обслуживанием предыдущих заявок, тогда в зависимости от типа заявки, которая поступила, и принятого в системе правила (дисциплины) обслуживания, только что поступившая заявка может ожидать свою очередь или прервать заявку, которая выполняется. В случае прерывания заявки предполагается, что она возвращается в очередь, где она будет ожидать продолжения прерванного обслуживания. Длительность пребывания каждой заявки в ВС составляется из времени ожидания заявки и времени обслуживания машиной.

Основные дисциплины обслуживания представлены на рис. 4.5.

* *Линейные дисциплины обслуживания*

В случае дисциплин без приоритетов заявки различных типов не имеют сопровождающих приоритетов для обслуживания и считаются при входе в систему равноприоритетными. Эти правила соблюдаются, когда заявки для обслуживания выбираются согласно:

- порядку поступления (первой в получении обслуживания будет заявка, которая пришла первой FIFO — First Input First Output) (рис.4.6);
- порядку, инверсному порядку поступления (первой получает обслуживание заявка, которая поступила последней LIFO — Last Input First Output) (рис. 4.7);
- заявки для обслуживания выбираются из очереди случайно.

Эти три дисциплины без приоритетов характеризуются одинаковым средним временем ожидания для всех заявок. Однако дисциплина FIFO минимизирует дисперсию ожидания, поэтому FIFO является более предпочтительной дисциплиной.

* *Циклические дисциплины обслуживания*

Из циклических дисциплин обслуживания в системах с разделенным временем наибольшее практическое применение имеют:

- циклический алгоритм обслуживания с одной очередью (RR) (Round Robin) (рис. 4.8);
- многоприоритетный циклический алгоритм обслуживания (FBn) (Foregraund—Backgraund) (рис. 4.9);

- многоприоритетный циклический алгоритм обслуживания (алгоритм Корбато) (рис. 4.10);
- смешанный алгоритм.

В случае циклического алгоритма планирования (RR) заявки получают обслуживание центральным процессором согласно их поступлению и в течение определенного кванта времени. Если обслуживание завершается в течение этого кванта, тогда обслуженная заявка покидает ресурс, и на обслуживание поступает заявка, которая следует за ней в очереди; иначе недообслуженная заявка размещается в конце очереди и должна ожидать следующий квант для получения окончательного обслуживания.

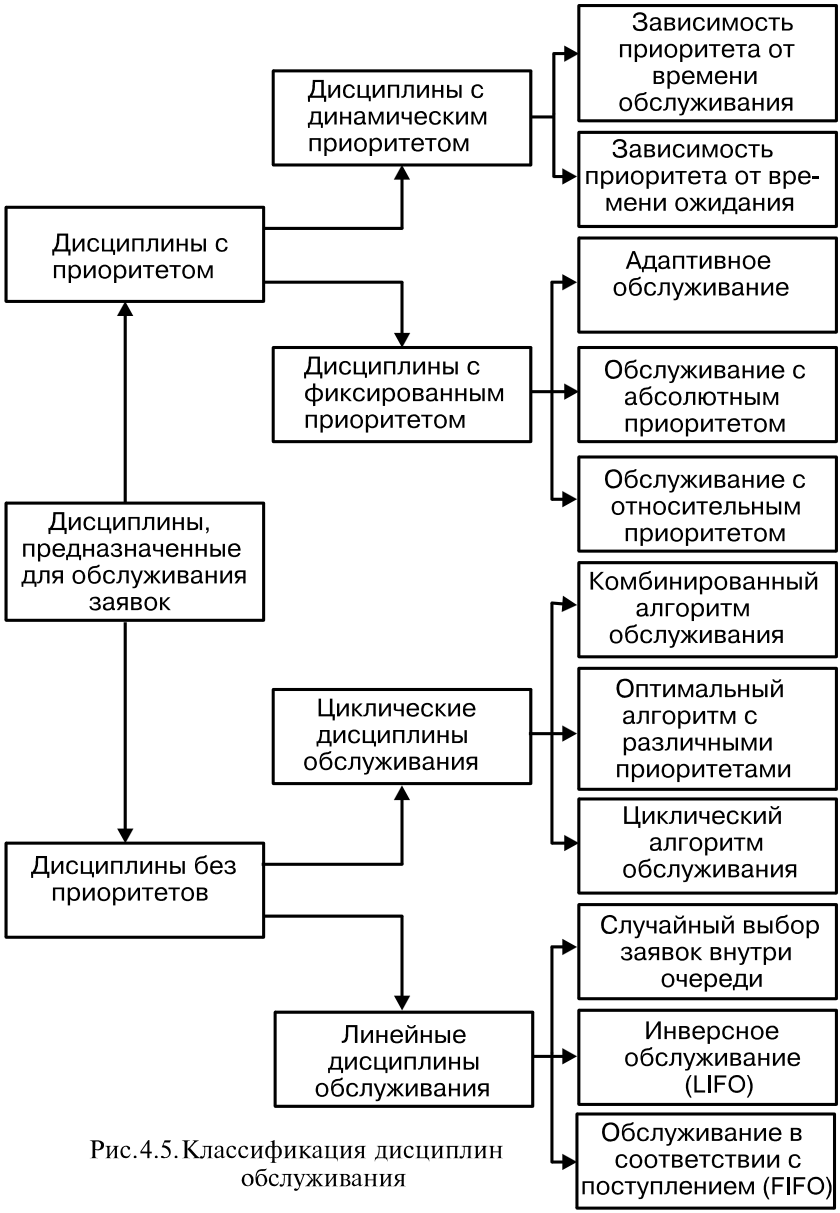

Рис.4.5. Классификация дисциплин обслуживания

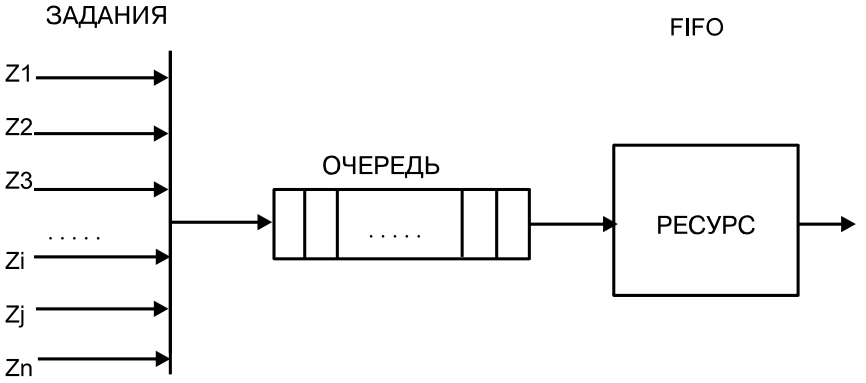

Рис. 4.6

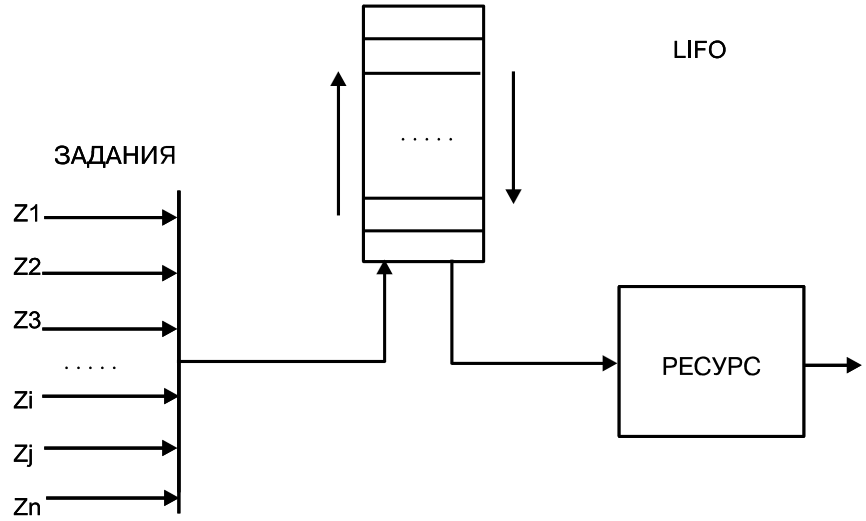

Рис. 4.7

В этом режиме важным является выбор величины кванта времени T. Если T велико, алгоритм приближается к алгоритму FIFO. По мере того, как T уменьшается, сокращается время обслуживания коротких заявок, но когда величина кванта мала, время переключения процессора на другую заявку будет больше, чем время обслуживания. Поэтому необходимо выбирать T ни очень большим, ни очень малым. Для систем, в которых предвидится большое число пользователей, уменьшение числа заявок может вызывать значительную задержку в сатисфакции

заявок. С этим можно бороться двумя методами. Первый из них состоит в выборе постоянной величины времени цикла, то есть времени ожидания для обслуживания заявки пользователя с момента ее поступления. Величина T зависит от количества заявок в очереди; чем большим будет количество заявок в очереди, тем меньшим будет величина T.

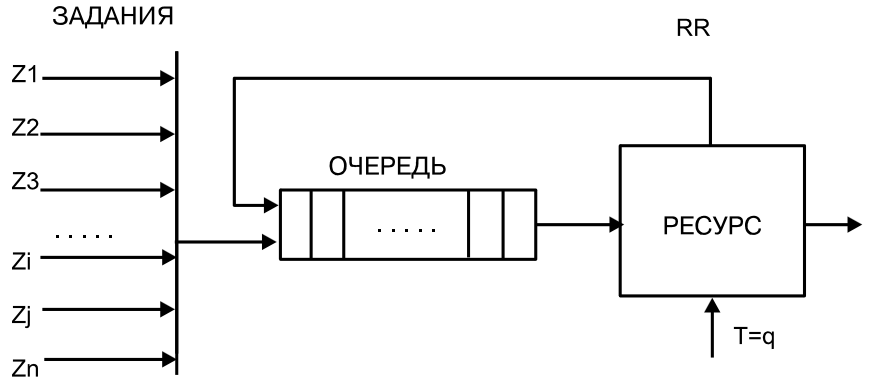

Рис. 4.8

Второй используется, когда мы не желаем значительного уменьшения времени T. Для этого мы определяем минимальную величину T, и, если новые заявки требуют меньший квант, чем установленный лимит, они не помещаются в очередь до тех пор, пока какая-нибудь из заявок в очереди не будет полностью обслужена.

Многоприоритетный циклический алгоритм (рис. 4.9), в отличие от предыдущего (RR), использует N очередей. Для всех очередей устанавливаются приоритеты, первая имеет наибольший приоритет. Внутри очереди заявки равноприоритетны.

Новая заявка помещается в очередь с наибольшим приоритетом. Если при завершении кванта обслуживание заявки ресурсом не завершилось, тогда она перемещается в конец очереди с меньшим приоритетом, чем предыдущая. Заявки в последней очереди обрабатываются без прерывания обслуживания.

Этот алгоритм является упрощенным случаем алгоритма Ф. Корбато (рис. 4.10).

В соответствии с этим алгоритмом, программа, которая состоит из Wp слов, включается в очередь с номером:

$$n = \left\lceil \log_2 \left(\left[\frac{Wp}{Wq} \right] + 1 \right) \right\rceil$$

Где:

Wq — количество перемещенных слов из внешней памяти в основную память, и наоборот, в течение времени q, символ [ ] означает целую часть числа.

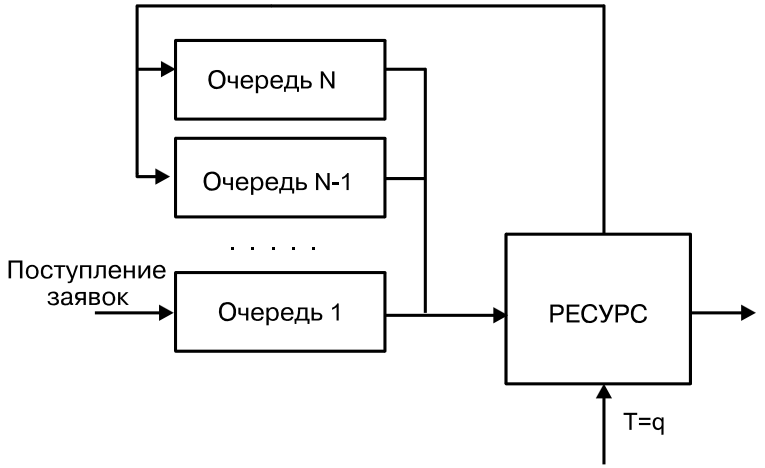

Рис. 4.9. FBn

Заявке поступившей на обслуживание из очереди, с номером N назначается квант обслуживания длиной $2^n * q$, и, в случае, когда она недообслуживается, она поступает в очередь n + 1. Если в течение времени обслуживания программы в очереди с номером N появляется программа в очереди N′<N, тогда ее обслуживание прерывается, она возвращается в начало очереди с номером N, начинается обслуживание программы в очереди N′.

В циклических алгоритмах принимается во внимание польза "коротких" программ, что отчетливо влияет на уменьшение величины обмена информацией между основной и вспомогательной памятью.

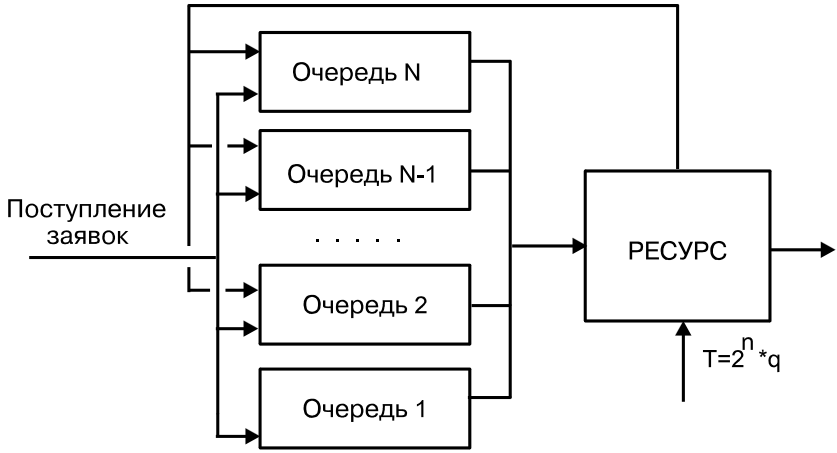

Рис. 4.10. Корбото

- *Смешанный алгоритм обслуживания*

Смешанный алгоритм обслуживания представляет комбинацию простого и многоприоритетного алгоритмов. Этот алгоритм используется в системе разделенного времени ЭВМ AN/FSO-32 фирмы IBM и был разработан как результат экспериментальных исследований.

Из программ, которые находятся в системе, организованы 3 очереди. Программы, которые введены в систему, размещаются в первой очереди, где они имеют право на три цикла обслуживания, каждый по 600 мс, в соответствии с циклическим алгоритмом обслуживания. Программы, которые требуют больше времени обслуживания, перемещаются во вторую очередь, где каждая из них получает обслуживание в течение цикла из шести квантов, каждый из которых 600 мс. Если обслуживание еще не закончено, они перемещаются в третью очередь.

В этой очереди программа согласно циклическому алгоритму получает обслуживание до конца и может получить до 20 квантов по 600 мс каждый раз.

После окончания каждого кванта обслуживание в очередях 2 и 3 может прерываться, если в течение этого времени появляется программа в очереди 1. Последовательность в обслуживании очередей устанавливается в соответствии с циклическим многоприоритетным алгоритмом.

- *Дисциплины приоритетного обслуживания*

Реальные свойства функционирования вычислительных систем в различных задачах могут отличаться своей относительной важностью и длительностью исполнения. Эти обстоятельства заставляют разрабатывать алгоритмы различных дисциплин обслуживания, которые присваивают приоритет задачам большей важности до поступления их в систему. В соответствии с принципами назначения приоритетов и их обработки, различают дисциплины с фиксированными и динамическими приоритетами.

Более общие — это дисциплины с относительно фиксированными приоритетами для различных потоков заявок. Особенностью, которая отличает эту дисциплину, является то, что появление заявки с более высоким приоритетом не вызывает прерывания обслуживания заявок с меньшим приоритетом.

В алгоритмах с относительными приоритетами обслуживание каждой новой заявки может начаться, как только прекратится обслуживание предыдущей заявки, не смотря на то, что последняя имеет более низкий приоритет. Как результат указанного ограничения, время ожидания в очереди для заявок с большим приоритетом может оказаться очень большим.

Уменьшения времени ожидания заявок с большим приоритетом можно достичь введением, так называемых, абсолютных приоритетов обслуживания. Обслуживание заявок с низким приоритетом прерывается каждый раз, когда в системе появляется заявка с более высоким приоритетом. Заявки с одинаковым приоритетом получают обслуживание в соответствии с порядком их появления в системе.

Время ожидания в очереди заявок с низким приоритетом, при наличии прерывания, зависит также от метода восстановления прерванного обслуживания.

Дисциплины бывают с восстановлением кванта обслуживания и с потерей прерванного кванта обслуживания. В случае использования алгоритмов обслуживания с абсолютными приоритетами возникает проблема определения целесообразности прерывания заявок с низкими приоритетами заявками с высокими абсолютными приоритетами. Чтобы оценить эффективность использования абсолютных приоритетов необходимо определить общие потери, с учетом системы штрафов за ожидание заявок каждого приоритета. Система должна оценить потери времени на прерывание и восстановление заявки с низким приоритетами время, необходимое на ее дообслуживание.

Упомянутые трудности обуславливают необходимость искать некоторую промежуточную дисциплину среди абсолютных и относительных приоритетов — адаптивное обслуживание. В случае этой дисциплины обслуживания определяется целесообразность прерывания обслуживания заявок с низким приоритетом, когда появляются заявки с высоким приоритетом. Если заявка получила необходимое обслуживание почти полностью, тогда оказывается целесообразным не прерывать ее и завершить обслуживание.

Критерий необходимости прекращения обслуживания заявок с низким приоритетом может быть представлен следующим отношением:

$$\Delta T_I$$

$$\Delta T_{JI} < \frac{}{a_I} a_J$$

Где: ΔT_{JI} — время необходимое для прекращения обслуживания заявки j в момент появления заявки с более высоким приоритетом i; a_J, a_I — штраф для каждой единицы времени ожидания заявок i, j; T_I — время обслуживания заявки i.

- *Дисциплины с динамическими приоритетами*

Дисциплины обслуживания с фиксированными приоритетами предполагают неизменность приоритетов заявок в течение времени ожидания и их обработки системой.

Так как относительная важность заявок может изменяться в зависимости от времени ожидания обслуживания и изменения времени обслуживания, появляется необходимость иметь такие дисциплины с приоритетами в обслуживании, где приоритет заявок зависит от реального времени обслуживания или от продолжительности ожидания заявки в памяти машины.

В описании алгоритма обслуживания в ряде случаев целесообразно использовать сочетание нескольких дисциплин обслуживания, которые приводят к рациональному компромиссу среди достоинств и недостатков для создания каждой из них .

Проанализируем изменение времени ожидания в очереди в зависимости от использования различных дисциплин обслуживания заявок. Сопоставляя время ожидания заявок различных приоритетов, можно показать, что введение относительных приоритетов имеет как результат уменьшение времени ожидания заявок с высоким приоритетом и увеличение времени ожидания заявок с низким приоритетом по сравнению с обслуживанием без приоритетов (FIFO). Это представлено на рис. 4.11, где кривая ОР соответствует дисциплине с относительными приоритетами.

Результат использования приоритетных дисциплин обслуживания представлен на рис. 4.12, где ОР является кривой относительного приоритета; РА — кривая абсолютного приоритета. Присваивая заявке абсолютный приоритет, мы получаем уменьшение времени ожидания заявок с высокими приоритетами, но одновременно увеличение времени ожидания заявок с низкими приоритетами.

В некоторых системах необходимо осуществлять строгие ограничения во времени ожидания некоторых заявок, фактически необходимо присваивать указанным заявкам абсолютные приоритеты. Как результат, времена ожидания таких заявок, имеющих низкие приоритеты, могут оказаться высокими. Для осуществления ограничений всех типов заявок можно вместе с абсолютными приоритетами присваивать некоторым заявкам относительные приоритеты, а остальные обслуживать без приоритетов. Такая дисциплина обслуживания называется смешанная. Например, в систему поступает М типов заявок. Если заявки типа 0-М имеют безприоритетное обслуживание (FIFO), среднее время ожидания может оказаться недопустимо большое (рис. 4.11., 4.12.).

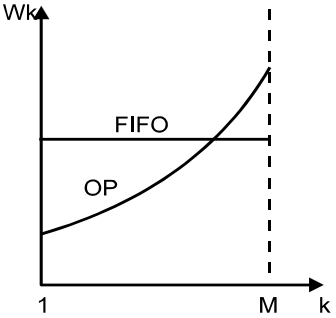

Рис. 4.11

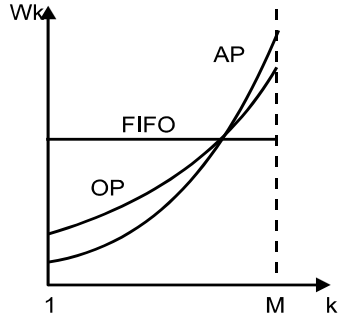

Рис. 4.12

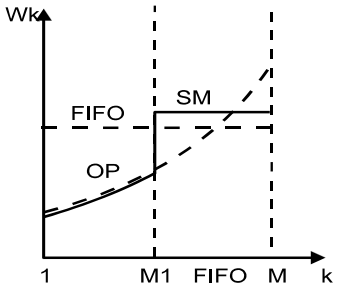

Рис. 4.13

Если заявки получают обслуживание с относительными приоритетами (кривая OP), тогда для заявок типа 1,..,M1 ситуация улучшается, но время ожидания заявок с низкими приоритетами превысит допустимые значения.

Если заявки типа 1,..,M1 получают обслуживание дисциплиной с относительным приоритетом и заявки M1+1..M получают обслуживание без приоритета (FIFO), тогда среднее время ожидания, соответствующее кривой SP, может соответствовать данным ограничениям (рис. 4.13).

Другие случаи использования смешанных дисциплин обслуживания показаны на рис. 4.14 и 4.15.

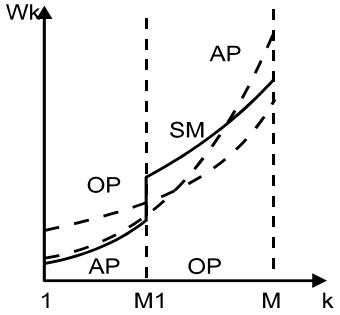

Рис. 4.14

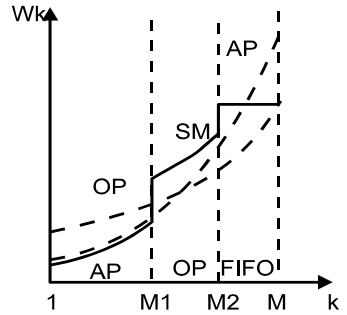

Рис. 4.15

4.1.4. Схема планирования в параллельных ВС

Интенсивное внедрение распределенных систем и систем массового распараллеливания требует изменения подходов ко всей системе организации и планирования вычислительного процесса. Существуют две составляющие в стратегии планирования работ в распределенной системе обработки информации:

- распределение рабочей нагрузки среди компьютеров (узлов) распределенной системы посредством перемещения и удаленного выполнения процессов;
- локальное планирование, которое определяет распределение времени локального процессора между локальными процессами в каждом узле.

Задача распределения рабочей нагрузки узлов распределенной ВС во времени может быть сформулирована как выбор***;*** перемещаемого ***процесса***, ***момента*** перемещения процесса и ***места*** назначения перемещаемого процесса, обеспечивающих повышение общей эффективности распределенной системы. Для оценки эффективности системы чаще всего используются следующие показатели: пропускная способность распределенной системы, время нахождения задания в системе (время ответа), длина очереди и время ожидания в очереди к ресурсу системы.

Классическая трехуровневая модель планирования вычислительного процесса (рис. 4.16 и 4.17) приемлема в ВС с небольшим числом процессоров и состоит из 3 типов планировщиков:

1. планировщик высокого уровня (ПВ);
2. промежуточный планировщик (ПП);
3. планировщик низкого уровня (ПН).

ПВ предназначен для *предварительного планирования* входного потока заявок, которые поступают в систему и претендуют на захват ресурсов вычислительной системы. Его иногда называют планировщиком доступа, т.к. он определяет, каким заданиям можно предоставить доступ к системе. Из всех этих заданий ПВ создает очередь О1 (рис. 4.17.) по некоторым критериям: относительные приоритеты, сроки

запуска и завершения, время выполнения, интенсивность ввода/вывода данных, потребность памяти. Назначение ПТ и БУФ будут обьяснены ниже.

ПП определяет, каким задачам будет разрешено конкурировать за захват ресурсов ВС. В результате работы ПП формируется очередь О2, в которой находятся работы (задания), которые ВС приняла для обработки и выделила основные ресурсы, кроме времени процессора. Кроме этого, планировщик промежуточного уровня, реализуя функции, связанные с эффективностью работы ВС, может временно приостанавливать и активизировать (или возобновлять) процессы для достижения бесперебойной работы ВС. Таким образом, планировщик промежуточного уровня действует как интерфейс между доступом заданий в систему и распределением ресурсов по этим заданиям.

ПН выполняет следующие функции:

- определяет какому процессу, готовому к выполнению, будет назначен процессор, который становится доступным, т.е. ПН фактически назначает процессор этому процессу;
- обеспечивает *прием* незавершенных ── прерванных заданий из О3;
- выполняет *распределение ресурсов* без конфликтов;
- обеспечивает *загрузку заданий* на соответствующие ресурсы по разработанному расписанию для выполнения.

Только после этого начинается процесс выполнения заданий.

ПН *делит* задания на два типа. Задания, требующие продолжения, отправляются в очередь О3, где они обрабатываются ПН при следующем цикле планирования, если они могут выполняться; или в БУФ, если задания приостанавливаются из-за невозможности дальнейшего продолжения (отсутствие ресурсов). Полностью обработанные задания удаляются из системы.

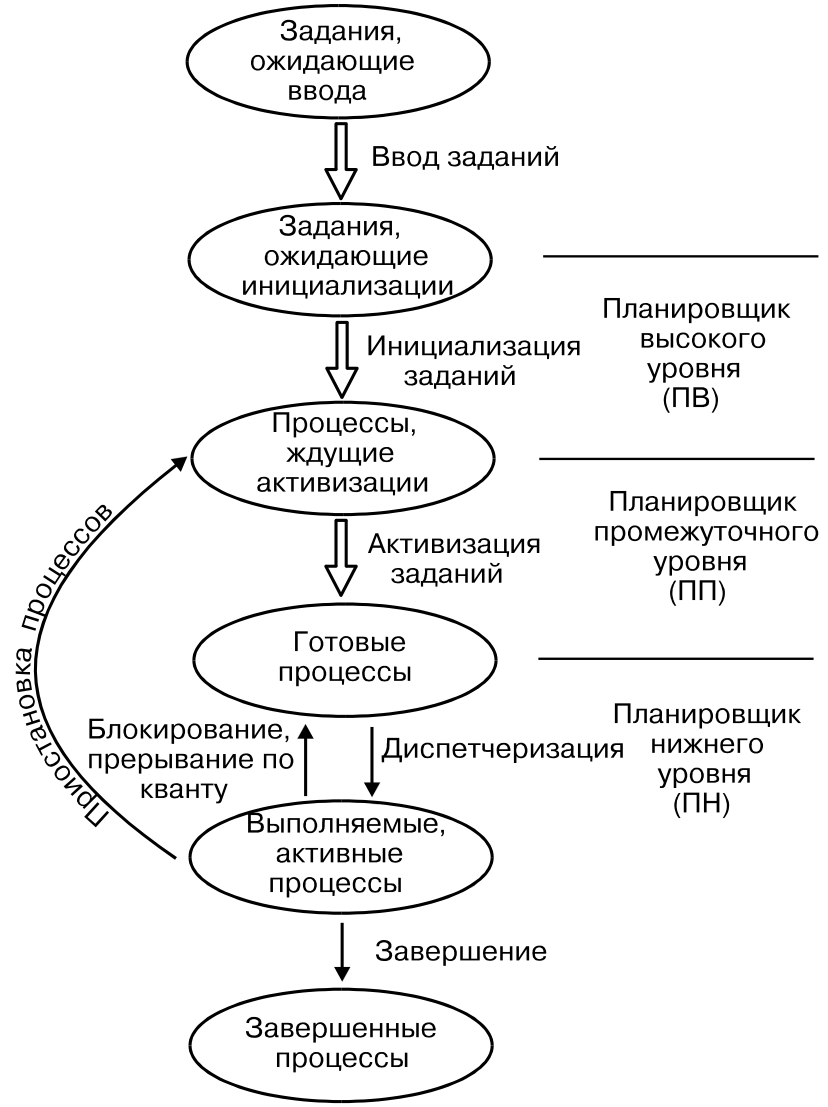

Рис.4.16. Уровни планирования классической модели планирования

ПН непосредственно выполняет управление ресурсами системы и работает в

динамическом режиме.

Эта классическая схема планирования приемлема в ВС с небольшим числом процессоров или ресурсов.

Интенсивное внедрение распределенных систем и систем массового распараллеливания требует изменения и дополнения системы организации и планирования вычислительного процесса, например:

- в функции ПВ добавляется структурный анализ взаимосвязи входного потока заявок;
- требуется новый тип планировщика (назовем его ПТ — планировщик транспортный, рис. 4.17) для распараллеливания заданий, синхронизации процессов по данным — обеспечения поступления требуемых данных и поддержки связей между вычислительными узлами при реализации связи по данным;
- обработанные задания (или их распараллеленные модули) транслятором ОС и ПТ ставятся в новую очередь — буфер БУФ;
- к ПП добавляется функция *адаптирования*, при выполнении которой задания распределяются соответственно особенностям данной системы (например: специальные схемы для систем гиперкуб, транспьютерных систем или дополнительная схема для неоднородной среды).
- к ПН добавляется функция балансирования в случае реконфигурации (отказа некоторых элементов) системы.

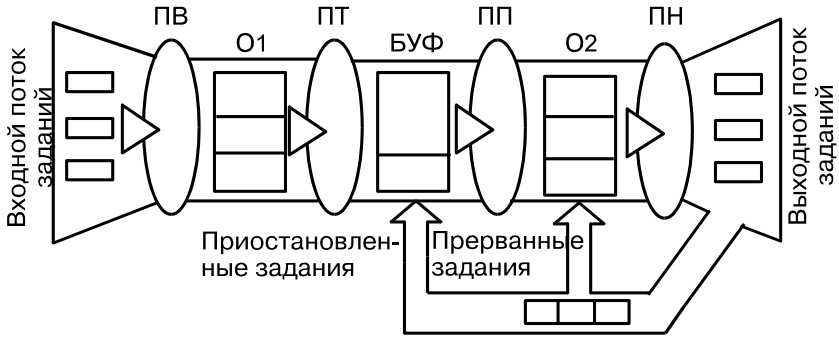

Рис. 4.17. Схема прохождения заданий через ВС

Полную систему планирования в этом случае можно представить в виде семиуровневой модели (табл. 4.1), где каждый уровень часто рассматривают как отдельную задачу. Порядок выполнения уровней может быть изменен в зависимости от особенности системы и целей задачи планирования.

Табл. 4.1

ПВ	1	Предварительное входное планирование исходного потока заявок, претендующих на захват ресурсов вычислительной системы	Задача Ввода
	2	Структурный анализ взаимосвязи входного потока заявок по ресурсам и определение общих ресурсов	Задача анализа
ПТ	3	Структурный анализ заявок и определение возможности распараллеливания каждой работы	Задача распараллеливания
ПП	4	Адаптирование распределения работ соответственно особенностям вычислительной системы	Задача адаптирования
	5	Составление расписания выполнения взаимосвязанных процедур: оптимизация плана по времени решения, количеству используемых ресурсов и количеству пересылок	Задача оптимизации
	6	Планирование потока процессов, претендующих на захват времени процессора/процессоров вычислительной системы	Задача распределения
ПН	7	Выделение времени процессоров ВС активизированным процессам, перераспределение (реконфигурация) работ в вычислительной среде (отказ оборудования)	Задача распределения и перераспределения

4.1.5. Общая постановка задачи планирования

Задачи распределения заданий (работ, процессов) на ресурсы в системах массового распараллеливания (СМР — однородных системах) или в распределенных системах (РС — неоднородных системах) имеют следующую общую постановку:

- *Вычислительная система* задана графом системы (Machine configuration graph) $G_R=(E_R,U_R,WE_R,WU_R)$, где $E_R=\{X_i \mid i=1,2,..,n\}$ являются узлами системы (ресурсами: процессорами, машинами...), $WE_R=g(Xi)$ являются весами узлов Xi, $U_R=\{(Xi,Xj), Xi\in E_R \mid Xj\in E_R\}$ являются ребрами (связями между узлами системы) и есть отображение Г от E_R, т.е. $U_R=\Gamma(E_R)$; $WU_R=f(U_R)$ являются весами ребер (например, задают время для передачи данных из одного узла в другой) (рис. 4.18).

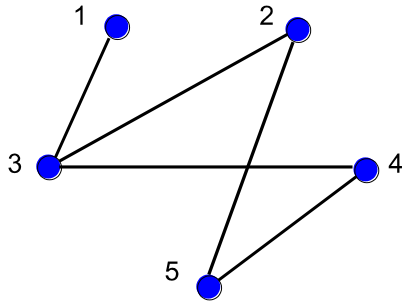

Рис 4.18. Граф системы

- *Поток заданий*, которые нужно распределить на заданной системе задается графом данных (заданий) (Dataflow Graph) $G_J=(E_J,U_J,WE_J,WU_J)$, где $E_J=\{Y_i \mid i=1,2,..,m\}$ являются заданиями (работами, процессами), $WE_J=h(Yi)$ являются весами заданий, $U_J=\{(Yi,Yj) \mid Yi \in E_J, Yj \in E_J\}$ являются дугами потока (связями между заданиями) и есть отображение Γ^* от E_J т.е. $U_J=\Gamma^*(E_J)$; где $WU_J=k(U_J)$ являются весами дуг (например, требование передачи данных из одного задания в другое). Граф заданий обычно является ориентированным и ациклическим. В том случае, если $U_J=\varnothing$, то задания, определяемые E_J, несвязаны (рис. 4.19).

- *Требованием задачи планирования* работ по ресурсам или распределения ресурсов по заданиям является нахождение множества $A=\{(Xp,Yq) \mid Xp \in E_R, Yq \in E_J\}$ такого, чтобы A удовлетворяло некоторым целевым функциям $\alpha=\{\alpha_1, \alpha_2,.., \alpha_s\}$ и также условиям оптимизации $\beta=\{\beta_1, \beta_2,.., \beta_t\}$, где βi является критерием оптимизации (по количеству используемых для выполнения заданий процессоров, по времени или количеству пересылок данных, по использованию памяти, по времени выполнения заданий, по занятию каналов связи и т.д.). Таким образом, A является отображением трех макрофакторов: *система* (архитектура, структура и характеристики), *поток заданий* (требования и структура) и *времени.*

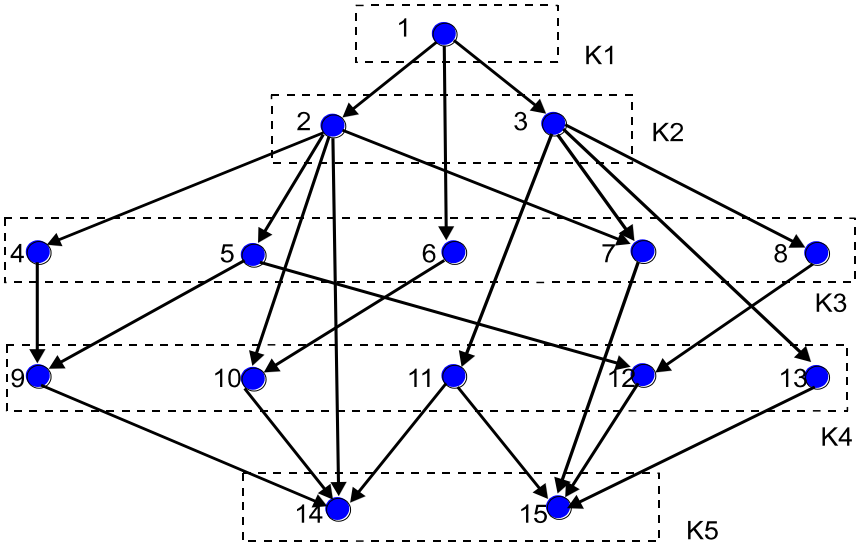

Рис 4.19. Граф заданий

4.2. КЛАССИФИКАЦИЯ И ХАРАКТЕРИСТИКА АЛГОРИТМОВ РЕШЕНИЯ ЗАДАЧИ ПЛАНИРОВАНИЯ В ПВС

За последние годы опубликовано много алгоритмов решения задач планирования и диспетчеризации [59, 103, 104, 134, 154]. Но, как отмечают многие авторы [25-155], процесс планирования многообразен и очень сложен. Несмотря на многообразие используемых терминов при описании методов, методик, подходов и алгоритмов решения задачи планирования—диспетчеризации (graph partitioning, grain packing, task allocation, task assignment, task scheduling, list scheduling, clustering), все они относятся к решению задачи планирования (составления расписания) захвата процессами ресурсов вычислительной системы. Полная задача планирования содержит сразу несколько подзадач, большинство из которых являются NP-полными или NP-сложными. Анализ известных алгоритмов [59, 103, 134, 135, 154] показал, что предлагая очередной "Алгоритм планирования", большинство авторов на самом деле решают только частные случаи полной задачи планирования [103, 135, 154]. Кроме этого, практически все алгоритмы имеют существенные ограничения по области применения и предназначены для решения задач планирования заданий (работ) в вычислительных системах с конкретными архитектурами, то есть являются узко специализированными. Следовательно, методы решения поставленной задачи, целевые функции и используемые техники

планирования также очень разнообразные и в значительной степени специализированы. Поэтому и классификация их усложнена, так как нет единой методики классификации алгоритмов планирования.

Существуют традиционные методы и критерии классификации алгоритмов планирования: по особенностям построения ВС (схожа с классификацией ВС) [59], по использованным методам планирования [103], по критериям оптимизации [134], по характеру исходной информации [154] и т.д. Из всего разнообразия видов классификации алгоритмов планирования процессов в ВС можно выделить следующие основные виды.

Это классификация по:

1. Динамичности решения задач планирования.
2. Особенностям вычислительных систем (однородности, организации памяти, степени параллелизма).
3. Характеру входного потока заданий.
4. Используемым критериям оптимизаций.
5. Дисциплинам обслуживания потока заданий.
6. Используемой технике планирования
7. Организации процесса планирования.

4.2.1. Классификация по динамичности решения задач планирования

Это наиболее распространенная схема. Ее использовали с первого момента, когда была сформулирована задача планирования, но обобщение этого метода было впервые сделано в работе Casavant и Kuhl [59], где была представлена общая постановка задачи планирования, а также классификация методов ее решения в распределенных системах обработки информации. В соответствии с этой классификацией все планировщики делятся на *статические* и *динамические* в зависимости от того, ***когда*** получены и выдаются для исполнения расписания и другие результаты решения задачи планирования, ***когда*** и на ***каком*** оборудовании эти результаты получены.

Сравнение двух моделей систем планирования — трехуровневой и семиуровневой показывает, что новые уровни планирования во второй модели предназначены для выполнения операционной системой действий, связанных с планированием загрузки множества ресурсов (необязательно однородных) ВС в пространственной координате и обеспечением их эффективного использования, увеличения общей пропускной способности ВС и уменьшения времени нахождения в системе каждой работы. Перераспределение в этой модели решаемых на некоторых уровнях задач в область статического планирования, т.е. решение их до начала вычислительного процесса, приводит модель планирования к традиционной схеме. В последнее время появились возможности решения некоторых задач, традиционно считавшихся статическими, в динамическом режиме. Это обусловлено развитием распределенных систем обработки информации и управлением ими распределенными операционными системами с совершенно новыми свойствами по

управлению, планированию и диспетчеризации с использованием новых математических методов.

Все задачи планирования и диспетчеризации, решаемые на разных стадиях организации и управления вычислительным процессом в вычислительной системе, можно разделить на следующие категории:

* *динамическое* планирование;
* *статическое* планирование;
* *балансовое* планирование;
* Планирование в *реальном* времени.

Под *динамическим* планированием понимается решение задачи распределения работ (составление расписания) и процессов в пространственных или пространственно—временных координатах во время выполнения вычислений и на том же оборудовании.

При динамическом планировании основным требованием, предъявляемым к планировщикам, является минимизация времени решения самой задачи планирования, т.к. эти затраты являются непроизводительными расходами машинного времени и ресурсов ВС, влияющими на эффективность использования ВС. Указанные временные ограничения определяют, в основном, те принципы и алгоритмы, которые положены в основу работы динамических планировщиков. При динамическом планировании используются простые алгоритмы приближенного решения или эвристические алгоритмы, не дающие гарантии получения оптимального решения. Динамические алгоритмы, как правило, имеют линейную временную сложность.

К задачам *статического* планирования относятся задачи, решаемые до выполнения самого вычислительного процесса. При этом возможно их выполнение на другом оборудовании и в другое время. Как правило, статический планировщик имеет полную информацию о совокупности вычислительных работ, ресурсов, их взаимосвязи, взаимодействии, а также количественных и качественных характеристиках. В этом смысле задачи статического планирования относятся к детерминированным задачам планирования. Сам процесс статического планирования разделяется на две фазы: подготовки информации и собственно составления расписания (планирования) и выполнения этого расписания на конкретном оборудовании (распределения). Первая фаза связана с выполнением процедур распараллеливания поступившей на вход ВС исполняемой программы [8] и ее представления в форме, удобной для параллельного исполнения в ВС. Следует отметить, что выполнение данной фазы может осуществляться и на стадии подготовки вычислительного процесса, при выборе параллельного алгоритма решения задачи, представления программы в виде взаимосвязанных модулей, процедур. Выбор параллельного алгоритма решения задачи может существенно повлиять на время реализации приложения в параллельной вычислительной системе. На этой фазе составляется расписание выполнения заданий без указания точного места его выполнения (похоже на составленное расписание занятий в ВУЗЕ без указания номера аудиторий). При выполнении второй фазы система

планирования производит адресное назначение вычислительного узла—процессора для каждого задания — процесса с учетом архитектурных особенностей ВС. Под статическим планированием большинство авторов [2, 3, 26] подразумевают составление расписания загрузки процессоров однородной или неоднородной вычислительной системы взаимосвязанными работами, описываемыми ориентированным, ациклическим графом.

При статическом планировании решаются две основные задачи [2,26]:

1. *Поиск минимального количества процессоров, необходимых для решения комплекса информационно — и по управлению взаимосвязанных задач за время, не превышающее заданное или критическое.*
2. *Поиск плана решения заданного комплекса информационно — и по управлению взаимосвязанных задач на заданном количестве процессоров за минимальное время.*

Обе задачи в дискретной математике относятся к классу NP-полных и называются "многопроцессорное расписание" [11, 26]. При снятии ограничений на количество процессоров, а также при двух процессорах, задача имеет точное решение за полиномиальное время [11]. Сложность задач планирования еще более увеличивается в случае неоднородной вычислительной среды, а также в том случае, если на систему планирования накладываются ограничения по количеству процессоров, времени выполнения и времени, затрачиваемого на обмен информацией между отдельными подзадачами. В большинстве работ, посвященных решению задач статического планирования, рассматриваются отдельные аспекты данной задачи и нет единой модели, охватывающей все аспекты ее решения.

Решение задачи *балансового* планирования характерно для распределенных систем обработки информации. Очень часто, при решении задач статического и динамического планирования одновременно, решается задача балансного планирования. Основными показателями эффективности управления распределенными системами являются: равномерность загрузки оборудования, пропускная способность системы, среднее время обслуживания. Эти показатели и определяют совокупность задач, решаемых при балансовом управлении и диспетчеризации процессов. При этом следует учитывать, что распределенная вычислительная система неоднородна как по своей структуре, так и по совокупности ресурсов, предоставляемых поступающим на вход системы процессам. При балансовом планировании следует выделить следующие основные задачи:

Начальное, базовое определение вычислительного узла, предназначенного для обслуживания поступивших на обработку процессов. Это задача определения адреса ресурса для выполнения заявки.

Перераспределение процессов при ухудшении макрохарактеристик эффективности системы в процессе обслуживания. Это задача динамической реконфигурации процессов — вычисление нового адреса выполнения процесса.

Выравнивание нагрузки или решение задачи распределения и перераспределения процессов с точки зрения оптимизации пропускной способности ВС.

Обеспечение восстановления и перераспределения процессов при выходе из строя ресурсов.

Все четыре задачи следует отнести к задачам динамического планирования и сложность их решения определяется принятой в системе стратегией централизованного или децентрализованного управления. При решении проблемы начально-базового распределения задач, как правило, учитывается длина очереди к каждому вычислительному узлу, и, исходя из этого, определяется адрес исполнительного элемента. Однако, длина очереди не может характеризовать время выполнения каждой заявки, а следовательно и характеризовать время обслуживания заявок. В некоторых алгоритмах учитывается длина программ в очереди, подчиненных активным процессам, косвенно влияющая на время их выполнения. Кроме этого следует учитывать различную производительность узлов ВС. Эти проблемы и обусловливают необходимость решения второй задачи *балансового* планирования, а именно, перераспределение процессов в ВС. Выполнение этой задачи требует от системы выполнения процедур, связанных с циркуляцией в системе информации о состоянии каждого узла, и вычисления обобщенных характеристик их функционирования. Выполнение второй задачи связано с необходимостью выравнивания нагрузки вычислительных узлов для улучшения характеристик всей системы в целом. Если поступившие задания распределены между узлами ВС неравномерно и управление ресурсами осуществляется каждым узлом независимо, то некоторые узлы могут оказаться перегруженными, в то время как другие недогруженными или простаивающими. При этом могут быть сорваны сроки выполнения работ. В этих условиях для улучшения функционирования вычислительной системы нагрузку желательно распределять между узлами равномерно, а управление ресурсами в каждом узле системы согласовывать с другими узлами.

Выравнивание нагрузки является попыткой улучшить эффективность функционирования распределенной вычислительной системы путем использования ресурсов всей системы для сглаживания периодов высокой нагрузки в отдельных узлах. Это осуществляется путем пересылки некоторой части рабочей нагрузки от перегруженного узла к другому, менее загруженному узлу.

Для выравнивания нагрузки должен быть решен целый ряд задач:

* выбор момента перемещения заданий пользователей с целью выравнивания рабочей нагрузки системы;
* выбор методов сравнения рабочей нагрузки различных узлов и определения порядка дозагрузки недогруженных узлов;
* выбор вычислительного процесса для перемещения;
* выбор наилучшего адресата для перемещаемого вычислительного процесса;
* выбор узла для включения в список недогруженных узлов;
* выбор момента принятия решения о выравнивании нагрузки;

* выбор параметров, учитываемых при решении вышеупомянутых задач;
* выбор стратегии управления в условиях недоступности данных;
* поиск компромисса между эффективностью системы и непроизводительными затратами на решение задач планирования;
* выбор стратегии прогнозирования перегрузки узлов ВС;
* выбор стратегии согласования узлами своих действий при принятии решений относительно выравнивания нагрузки;
* выбор централизованной или децентрализованной стратегии сбора и хранения данных, необходимых для принятия решения относительно выравнивания нагрузки.

Обеспечение решения задач, связанных с повышенными характеристиками надежности и живучести, требует включения в систему *балансового* планирования процедур создания контрольных точек для активных процессов, возможности восстановления потерянного процесса, определения места хранения контекста процесса в контрольной точке, определения момента времени потери процесса и его идентификации, знания пространственно—временных координат возобновления. При использовании в полной мере этого механизма для всех процессов, находящихся в ВС, возникает опасность значительного снижения эффективности работы системы в целом за счет дополнительных работ, выполняемых системой. Кроме этого, инфраструктура системы может быть перегружена обменами, обусловленными выполнением внутренних функций обеспечения *балансового* планирования.

Планирование в *реальном* времени характеризуется решением следующей задачи — определением плана решения совокупности задач с заданным временем исполнения и ограничениями по времени выхода задач из системы. От системы планирования требуется выполнение требований по минимизации суммарного времени отклонения реального выхода задач из системы (выполнения сроков решения) от исходных временных ограничений (системе планирования сообщается время поступления задачи в систему, время решения и крайнее время выхода заявки из системы) при полном соблюдении порядка следования работ. Задача относится к классу NP-полных.

В большинстве случаев разработчики систем планирования реального времени используют статические алгоритмы и заранее определяют максимальный список заданий, допустив наихудший случай для получения статической управляющей таблицы (плана). Этот план фиксируется и используется для безусловного исполнения в динамическом режиме со следующими допущениями:

- все временные ограничения остаются неизменными на время выполнения плана;
- все задачи вкладываются в свое критическое время.

В других случаях при помощи приемов статического планирования создается статический список приоритетов для использования во время диспетчеризации самих работ в динамическом режиме.

Если система реального времени работает только в динамическом режиме, то использование соглашений статического планирования (где все известно априори) недопустимо. В этом случае выбирается один из возможных алгоритмов составления расписания и тщательно анализируется на применимость в ожидаемом динамическом окружении.

Классическая теория планирования в реальном времени обычно пользуется такими критериями оценки системы планирования и получаемого плана (расписания), как минимизация суммарного времени выполнения плана решения задач, минимизация взвешенной суммы времен выполнения, минимизация длины управляющей таблицы, минимизация требуемого количества процессоров, минимизация максимального или суммарного запаздывания работ. Во многих случаях критический срок выхода из системы для каждой задачи в отдельности не принимается во внимание, а оценивается общая эффективность системы. Когда же критические времена выхода каждой заявки учитываются, то они обычно оцениваются вместе с другими критериями, например, управляющая таблица планирования должна быть минимальной длины при условии, что все задачи должны уложиться в критические сроки. Если же одна или несколько задач в критические сроки не укладываются, то такое решение является неприемлемым.

Параметр минимизации максимального запаздывания реального выхода заявки из системы по отношению к заданному может быть полезен во время разработки алгоритма планирования, когда можно добавлять ресурсы до тех пор, пока максимальное запаздывание не будет сведено до нуля. В этом случае все задачи укладываются в свои критические сроки. С другой стороны, минимизация максимального запаздывания не гарантирует того, что все задачи уложатся в свои критические времена и может случиться, что одна, несколько или даже все задачи не укладываются в критические сроки.

Вместо использования вышеупомянутых параметров многие системы реального времени в процессе работы ищут оптимальный алгоритм, определяемый следующим образом: оптимальным является тот алгоритм планирования, который не может обеспечить выполнение всех задач в критические сроки только в том случае, когда этого не может сделать ни один другой алгоритм.

Статический планировщик можно представить как компилятор, работа которого схематически изображена на рис. 4.20.

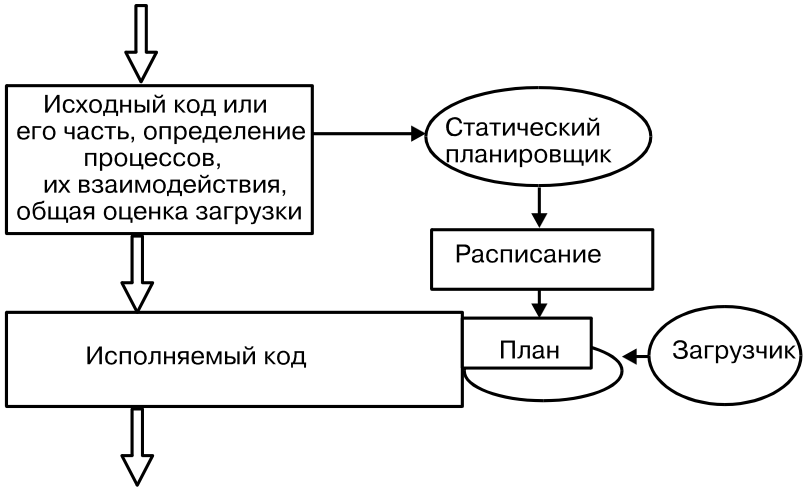

Рис. 4.20. Обшая схема статического планирования

Динамический Планировщик представляет собой часть ОС. Он получает нужную информацию о состоянии системы непосредственно перед выполнением процедур планирования (рис. 4.21.).

Граница между статическим и динамическим планированием не является строгой, и данная классификация не годится для алгоритмов, которые выполняют коррекцию расписания работ для исполнения его конкретными конфигурациями ресурсов, т.е. когда статические алгоритмы создают исполняемые расписания во время выполнения работ в ВС [24].

Stankovic [72] предложил новое определение *динамического планировщика* как процедуры, которая имеет полное *знание* о текущих заданиях, но у нее нет знаний о заданиях, которые возможно поступят. Он также ввел понятия *On-line и Off-Line части планировщика.* Off-Line часть выполняет действия до начала вычислительного процесса, результаты которых используются в On-Line части. В этой работе также утверждается, что один и тот же алгоритм планирования (возможно с некоторой модификацией) можно относить как к статическим, так и к динамическим и можно использовать в On-line или Off-Line частях. Несмотря на кажущуюся простоту классификации, этот подход не обладает универсальностью, так как в некоторых случаях нельзя определить к какому виду следует отнести алгоритм планирования: к статическому или динамическому.

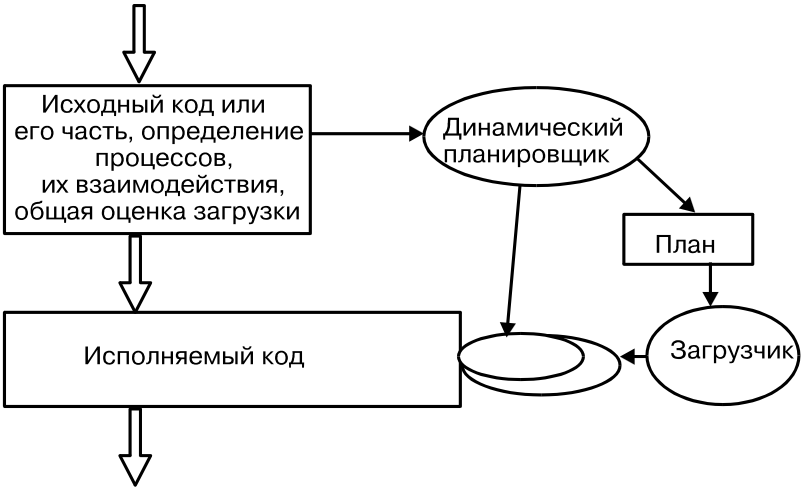

Рис. 4.21. Обшая схема динамического планирования

4.2.2. Классификация по особенностям систем

Эта классификация включает в себя сразу несколько видов классификационных признаков из-за того, что ВС можно характеризовать по несколькими различным критериям. Можно выделить и классифицировать алгоритмы планирования по трем основных характеристикам вычислительной системы:

- по степени *однородности* системы;
- по способу *организации* памяти;
- по *степени параллелизма* системы.

- *По степени однородности системы*

Имеются два существенно отличающихся вида алгоритмов:

1. Алгоритмы для *однородных* систем (АОС).
2. Алгоритмы для *неоднородных* систем (АНС).

АОС [78, 83, 95, 116, 147, 154] часто используют в системах массового распараллеливания (СМР), а АНС [51, 72, 80, 101, 142, 145, 155] ── в распределенных системах обработки информации (РСОД или РС). Различие между ними показано в табл. 4.2. Следует отметить, что для СМР имеются много специализированных и точных методов и алгоритмов распределения ресурсов, а для РС практически отсутствуют методы точного решения за приемлемое время.

- *По организации памяти системы*

В общем случае планирование делится на централизованное и распределенное. Однако, при решении задач планирования, связь между процессами по управлению или по данным часто рассматривается как организация передач между блоками памяти в пространстве и/или во времени. Поэтому алгоритмы планирования делятся на:

1. Алгоритмы для *систем с общей (разделяемой) памятью (АСОП);*
2. Алгоритмы для *систем с распределенной памятью (АСРП)*.

Основным отличием между АСОП [73, 113, 116, 140] и АСРП [124, 150, 153] является значительное различие по времени доступа к информации и сложности организации и реализации обменов информацией между отдельными вычислительными узлами при выполнении работ, имеющих связь по данным. Поэтому критерии оптимизации при решении задач планирования для этих систем также являются разными. Если для АСОП критерием качества получаемого расписания является достижение максимального использования ресурсов, то для АСРП ── это минимизация затрат времени на пересылки, хотя общей целью обоих видов алгоритмов планирования является минимизация времени выполнения исполняемых работ.

Табл. 4.2.
СРАВНИТЕЛЬНЫЙ АНАЛИЗ ЗАДАЧИ ПЛАНИРОВАНИЯ В СМР И В РС

Системы / Характер.	СМР (Однородные)	РС (Неоднородные)
Системы (Граф)	Однотипные, регулярные	Общего вида
Задания (Граф)	Связные	Связные, несвязные
Критерии оптимизации	Оптимизация загрузки оборудования, минимизация передач данных, времени исполнения	Оптимизация загрузки оборудования, балансирование, минимизация передач данных, времени исполнения
Специализация алгоритмов	Специализированные	Общего вида
Возможность реконфигурации	Аппаратная	Аппаратная, программная

Преимущество типа планирования	Статическое	Статическое, динамическое
Техника планирования	Специализированная	Универсальная

Понятие РС обычно связывается с ВС, удаленные независимые узлы которых характеризуются ограничениями на скорость передачи информации по каналам связи. В последнее время применение новых технологий при организации межузловых передач информации по каналам, таким как FDDI и HiPPI, существенно ускорили скорость передачи данных по каналам связи и позволили существенно уменьшить время взаимодействия процессов. Эти достижения позволяют прогнозировать возможность сближения подходов, разработанных для решения задач планирования для РС с АСРП и СМР с АСОП.

- *По степени параллелизма.*

В зависимости от отношения числа заданий, подлежащих распределению, к числу ресурсов вычислительной системы, различают три вида планирования:

1. Крупно-зернистое планирование (Coarse Grain Scheduling);
2. Средне-зернистое планирование (Medium Grain Scheduling);
3. Мелко-зернистое планирование (Fine Grain Scheduling).

Следует отметить, что это деление условное и невозможно точно определить границу между отдельными видами.

При *крупно-зернистом* планировании число заданий и ресурсов является небольшим и измеряется единицами. Планирование выполняется с использованием "таблицы назначений". Методы, используемые для планирования, являются приближенными, например, планирование по спискам (List Scheduling). Крупнозернистое планирование применяется при планировании решения крупных задач в распределенных ВС.

При *средне-зернистом* планировании имеется много заданий и много идентичных ресурсов (например, в СМР (MPS)). Планирование этого типа часто выполняется без учета ресурсных ограничений (то есть ограничения на количество процессоров) и имеет 2 направления:

- сначала выполняют планирование без учета ресурсного ограничения, затем адаптируют полученное расписание на имеющиеся оборудование — процессоры [75, 77];
- планируют без учета ресурсных требований вообще или когда можно доказать, что полученное решение достаточно эффективное [78].

При *мелко-зернистом* планировании число заданий значительно превышает число ресурсов. Ресурсы являются неоднородными, но могут делиться на группы (кластеры), где каждая группа имеет небольшое число идентичных ресурсов. Это

направление начало развиваться 30 лет назад и часто используется в конвейерных системах и супер-системах (CDC 6600, VLIW). Для этого вида планирования невозможно не считаться с ресурсным ограничением. В литературе встречаются алгоритмы для решения этой проблемы приближенными методами [132], такими как метод трассировки (Trace Scheduling), конвейерный метод планирования.

Планирование методом трассировки (Trace Scheduling) является основой для разработки алгоритмов планирования по спискам для основных блоков одной программы (basic blocks) [79]. Конвейерный метод планирования применяется для простых циклов (simple loops) с целью достижения максимального использования ресурсов и минимизации суммарного времени выполнения заданий [133]. Расписание для каждого цикла характеризуется интервалом старта, который равен периоду времени между стартами 2-х следующих друг за другом итераций этого цикла. Этот интервал имеет 2 границы — ограничения. Верхняя граница задается временем последовательного выполнения всех итераций. Нижняя вычисляется без ограничений на количество ресурсов. Большинство алгоритмов этого вида, принимая метод планирования по спискам, предполагает, что загрузка каждого ресурса возможна в интервале старта. Потом, с заданным допушением ошибок, система планирования ищет оптимальный вариант методом двоичного исключения [110]. Применение мелко-зернистого планирования характерно для систем массового распараллеливания (многопроцессорных систем).

4.2.3. Классификация по характеру связности заданий

В зависимости от степени связности заданий имеются два вида алгоритмов планирования:

1. Алгоритмы для связных заданий (АЗС).
2. Алгоритмы для несвязных заданий (АНЗ).

Они отличаются степенью связности, детализацией описания планируемых процессов и оборудования, адресностью планирования. По этим критериям задачу планирования рассматривают в 2-х главных направлениях [155]:

1. Задача — распределение (загрузка) (Task Allocation). Вычислительные процессы имеют *слабые* требования по предшествованию и отображаются несвязными графами. Это направление часто связывается с задачами балансирования и минимизации пересылок. При решении этих задач основой является определение — какой процесс (задание) будет выполняться на каком процессоре (ресурсе), а не определение порядка их выполнения [155]. При этом, в основном, решаются задачи пространственного распределения процессов (заданий).
2. Задача — планирование (Task Scheduling). Вычислительные процессы имеют сильные требования по предшествованию и отображаются связными ациклическими графами (DAG). При этом, к решению задач первого направления добавляется также определение порядка выполнения заданий. На этом уровне часто решаются задачи минимизации суммарного времени

выполнения полного DAG на выделенных или имеющихся ресурсах.

Задачи по обоим направлениям, в общем случае, являются NP-полными или NP-сложными даже для двухпроцессорной системы [11] и имеют экспоненциальную временную сложность.

Известно три алгоритма точного решения задачи составления расписания для трех частных случаев данной задачи. Все они основаны на теории планирования очередей и не учитывают время, затрачиваемое на обмен информацией:

1. Задача минимизации времени выполнения графа задач, заданного в виде дерева для мульти — процессорной системы [61].
2. Задача минимизации времени выполнения графа задач любого вида для двух — процессорной системы [61].
3. Задача минимизации времени выполнения интервально- упорядоченного графа для мульти-процессорной системы (interval-oder) [123].

Для более общих случаев известны только эвристические алгоритмы, которые дают субоптимальные результаты.

Имеются следующие основные подходы, использующие понятие "критический путь":

Планирование по спискам (*List Scheduling*). Это классический метод с разными вариантами реализации [47, 71, 112, 151], который часто называют однопроходным или одношаговым. Планирование по спискам обычно используют для планирования в синхронных системах. Его суть состоит в следующем: задана очередь L, состоящая из вершин — заданий графа G. Каждому заданию из L присвоен приоритет. Список расписания S для L можно найти следующими процедурами:

- элементы расписания S для процессоров определяются, начиная с некоторого временного такта i так, чтобы при сканировании очереди L первое готовое задание с наивысшим приоритетом (которое еще не было запланировано), было выбрано для выполнения в этом i-том такте;
- если нет ни одного подходящего задания для свободных процессоров на данный i-й такт, то поиск продолжается в следующем i+1 такте.

Имеются некоторые варианты планирования по спискам. Они различаются по методам выбора задания из L :

- HLF — задача с наивысшим приоритетом — первая (Highest Level First);
- CP — обработка задач по критическому пути (Critical Path);
- LP — самые отдаленные задачи первые (Longest Path);
- LPT — задачи, требующие большего времени обслуживания, обслуживаются первыми (Longest Processing Time).

2. *Кластерное планирование (Clustered Scheduling).* В чистом виде применяется только для систем, где нет ограничений на выделяемые ресурсы или в масштабируемых вычислительных системах. Задания из исходного графа делятся на группы (кластеры). Этот процесс называется кластеризацией, и его

решение в общем виде имеет экспоненциальную временную сложность. Задания одной группы (кластера) выполняются на отдельном процессоре [155];

3. Планирование по *смешанному* методу (Multistage). Это комбинация предыдущих подходов. Сначала используют второй подход, с предположением, что число процессоров неограниченно и сеть является полносвязной. Потом, на следующих шагах, используя выделенные кластеры, используют первый подход для планирования заданий на n процессорах [87, 137, 151].

Моделирование этих алгоритмов [68, 88, 100] показало, что результаты, полученные смешанным подходом, существенно лучше по сравнению с использованием метода планирования по спискам. Но эти преимущества становятся незначительными, когда число заданий значительно превышает число процессоров.

Рассмотрим более подробно алгоритмы кластерного планирования.

Модель: исходный граф DAG задается в виде G=(V,E,C,W), где V — множество заданий — вершин n_i , его размер v=|V|, E — множество связей (дуг), $e_{ij}=(n_i,n_j)$ и их количество e=|E|, C — множество весов связей, W — множество времен выполнения заданий. Значения w_i в W есть время выполнения задания n_i в V.

Все задания получают все нужные данные перед выполнением. Задания выполняются непрерывно до их завершения. После выполнения очередного задания результаты *немедленно* передаются на все следующие. Граф заданий является статическим. Система является однородная и полносвязная, состоящая из p процессоров. Целью планирования является нахождение квазиоптимального решения по суммарному времени выполнения без учета времени, затрачиваемого на обмен информацией.

Можно выделить две основные стратегии решения:

- "Планирование *вперед*";
- "Планирование *назад*".

1. "Планирование вперед". Сначала все задания в исходном графе назначают на различные процессоры. Затем, используя один из методов, обьединяют (кластеризуют) задания в группы.. Наиболее известными "кластерными" алгоритмами являются *Edge —zeroing Based Clustering Algorithms* (ДЗКА) [155]. В этой группе алгоритмов в кластеры группируются только связные задания и предполагается, что нет ограничений на количество процессоров и, следовательно, нет необходимости решения задачи "балансирования". Т.е. цель кластеризации — уменьшение количества занятых процессоров.

Имеются 2 вида ДЗКА:

- без учета количества процессоров (ДЗКАбО).
- с учетом количества процессоров (ДЗКАсО).

Из группы ДЗКАбО: алгоритм Efe [71] был одним из первых разработанных алгоритмов с использованием метода планирования по группам (АПГ). Kim, Browne [100] первыми предложили линейный АПГ с временной сложностью O(v(e+v)). Новую технику клайстеризации, названную "собирание" (Clan), использовали в [117]. А наилучшую временную сложность имеет алгоритм Yang—Gerasoulis [152], который является развитием алгоритма

Gerasoulis—Venugopal [86] и алгоритма *Yang—Gerasoulis* [151] с временной сложностью *O(elog(e)) O((e+v)logv)* соответственно

Из группы ДЗКАсО: алгоритмы класса АПГ с ограничением на число процессоров (АПГсО) предложили Wu-Gajski [149] и Sakar [137]. Временная сложность алгоритмов равна $O(v^2 logV)$ и O(v(e+v)).

Они выполняются с использованием следующих предпосылок:

- выполнять обнуление дуг или кластеризацию, начиная с неограниченного числа процессоров;
- обьединение (кластеризация) вершин графа выполняется с минимальным увеличением критического времени при адаптировании числа групп (кластеров) к числу имеющих процессоров.
- планирование по очереди является основной процедурой планирования.

Недостатки ДЗКА*:*

- неопределенное число кластеров;
- нет возможности балансировать загрузку;
- сложность реализации из-за применения сразу несколько техник: группирование, объединение, планирование по спискам;
- в случае, если исходный граф большой, а число процессоров маленькое, данный метод не отличается от ПО.

2. "Планирование назад". Сначала предполагают, что все задания назначены и выполняются на одном процессоре. Если имеется еще один процессор, некоторые задания будут перемещаться на дополнительный процессор для параллельного выполнения, и так продолжается до тех пор, пока количество процессоров не станет равным заданному или время решения минимальным.

В алгоритме Kernighan-Lin [113] используется эвристический невозвратный метод без дублирования задания на разных процессорах.

Типичными *несвязными* заданиями являются итерации параллельного цикла (IPL—Iterations in Parallel Loop). Например, параллельные операции в цикле DO во многих языках программирования. Параллельные циклы (ПЦ) могут быть однородными (когда время выполнения каждого задания — итерации одинаковое) или неоднородными (время выполнения разное). В неоднородных циклах возможен случай, когда одно задание завершается намного позднее, чем другие задания цикла (процессор, на котором выполняется последнее задание называется *критическим Pc*). Поэтому, в этом случае балансирование является ключевой задачей улучшения эффективности вычисления [113].

Общая постановка задачи планирования в данном случае — распределить (спланировать) N заданий параллельного цикла L на P процессорах.

Имеются следующие основные подходы планирования при решении задачи балансирования:

1 Статический (Static Scheduling Schemes). Задания назначают на ресурсы во время компиляции, поэтому уменьшаются временные затраты для из

динамического планирования во время выполнения. Но при этом не решается задача балансирования работы процессоров. Примерами этого метода являются алгоритм: Static Chunk (SC) [111] и алгоритм Round Robin (RR) [111].

2 Динамический (Self Scheduling Schemes). Задания назначаются во время их выполнения, поэтому непроизводительные временные затраты увеличены [120], т.к. планирование выполняется в процессе решения основной задачи. Но при этом имеется возможность решения задачи балансирования работы процессоров. Динамическая модель, как правило, предусматривает глобальный список заданий. Свободный процессор сам выбирает и загружает следующее задание и выполняет его. Это означает, что процессор сам планирует себе работу, как планировщик. Этот процесс иногда называют "самостоятельное планирование" (Self—Scheduling).

Примерами этого метода являются: алгоритм Pure Self—Scheduling (PSS) [121], алгоритм Chunk Self—Scheduling (CSS) [111], алгоритм Guided Self—Scheduling (GSS)[129], алгоритм Trapezoid Self—Scheduling (TSS) [147], алгоритм Factoring [95], алгоритм Affinity Scheduling [116], SAFE Self—Scheduling Algorithm (SSS) [113].

4.2.4. Классификация по критериям оптимизации

Учитывая критерии оптимизации вычислительного процесса, системы планирования и диспетчеризации можно классифицировать по:

- минимальному времени выполнения заданий;
- минимальному количеству обменов информацией;
- минимальному суммарному времени передачи данных;
- максимальному использованию ресурсов;
- максимальной степени параллелизма;
- максимальному балансированию загрузки.

По критериям оптимизации процесса планирования:

- качеству планирования (точное решение);
- времени, затрачиваемом на планирование;
- минимальным ресурсным затратам для выполнения планирования.

4.2.5. Классификация алгоритмов по технике планирования

Техника планирования, используемая при составлении плана, имеет два направления со следующими подходами:

- по технике моделирования;
- по технике решения.

4.2.5.1. Классификация по технике моделирования

Эта классификация имеет философскую основу. Она заключается в том, что каждый алгоритм должен иметь 3 следующих составляющих:

1. Модель загрузки — МЗ (Load Model). Это модель (описание) всех компонент ВС, участвующих в процессе загрузки, которая включает в себя описание и информацию о системе. Описание загрузки должно содержать определение объектов загрузки (ОЗ) (load units), то есть задания и ресурсы, отношение между ними, их требования для процесса вычисления и обслуживания;
2. Модель действий — МД (Action Model). Эта модель определяет нужные действия в соответствующих ситуациях и с соответствующими компонентами системы. Она касается также процесса распределения и сбора информации. Она отражает физические действия процесса планирования и является источником разработки метода решения;
3. Модель решения — МР (Solution Model). Эта модель отражает цель и стратегию планирования, приблизительные знания про вычислительный контекст решения задачи планирования в данной системе.

Только после того, как эти модели определены, можно сказать о самой технике планирования, которую назовем процедурами решения (Decision making Procedure).

Между МЗ, МД, МР и процедурами решения существуют логические и физические связи. Цель планирования так же оказывает влияние на МД и на МР (а иногда на МЗ) (рис 4.23.).

МД описывает процесс загрузки модели, выбранной в МЗ. В МЗ определяются статические характеристики загрузки и стратегии для динамического описания системных состояний МР. Тогда МР выполняет только те действия, которые определены в МЗ. В общем случае можно выделить 3 разных подхода (тактики) для решения задачи планирования:

- решить данную проблему аналитически точно или приблизительно;
- решить упрошенную задачу данной проблемы;
- выбрать решение перебором нескольких вариантов на основе некоторых критериев.

В зависимости от выбора тактики решения, алгоритмы планирования могут иметь совершено различные формы. Цель планирования является важным критерием для классификации алгоритмов.

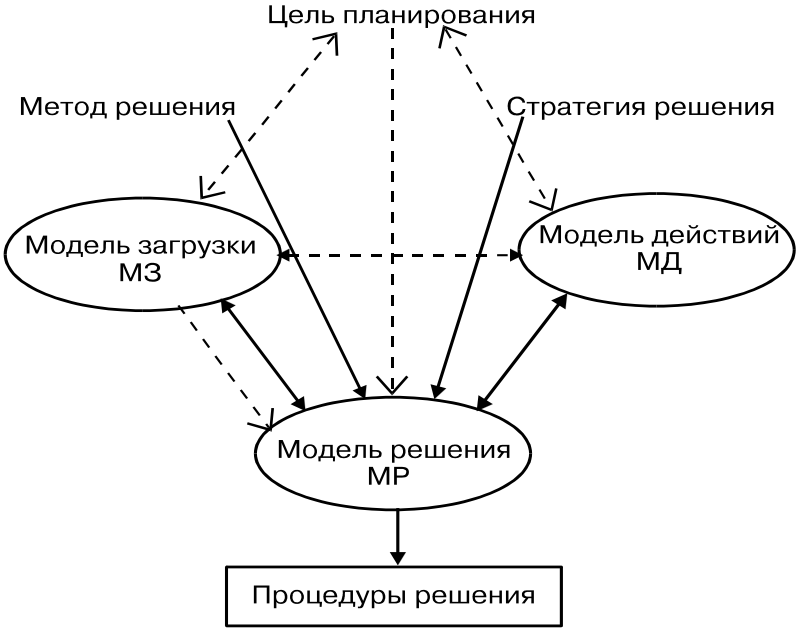

Рис. 4.23

Цель описывается в МР в двух видах:

1. Как функция затрат на решение задачи планирования (Cost Functions).
2. Как функция входных переменных для МР (Input Variablee).

Целями планирования могут быть:

1. Эффективное обслуживание заданий.

Достижение:

- малого времени ожидания;
- малого времени обслуживания.;
- малого времени реакции (ответа).

2. Предсказуемость времени ответа.
3. Выполнение обслуживания заданий за заданное время.
4. Высокая готовность системы.
5. Малое время доступа заданий к данным.(High affinity).
6. Высокая пропускная способность ВС.
7. Выполнение требований "балансирования":
   - выравнивание загрузки ресурсов системы;
   - адаптирование числа используемых ресурсов к числу имеющихся для обработки заданий.

8. Эффективное использование ресурсов:
   - малое время простоя ресурсов;
   - малое время использования ресурсов или маленький трафик (минимизация количества и времени обменов).

Имеются следующие виды алгоритмов:

1. Алгоритмы с меняющимися различными стратегиями.
2. Алгоритмы с одной стратегией, но с динамично—меняющимися параметрами.
3. Алгоритмы с одной статической стратегией.

В МЗ определяется:

- что является источником информации в процессе выполнения заданий? Какой тип информации?
- какой ресурс выполняет действия, определяемые процедурами решения?

Имеются два пути получения информации о состоянии системы для выполнения действий загрузки:

- от источника фиксирующего реальное состояние системы;
- от планировщика в виде расписания, которого система строго придерживается.

В зависимости от степени использования информации о системе имеются следующие типы алгоритмов планирования:

1. Статические (не используются вообще).
2. Динамические (используются).
3. Комбинированные (работают как статические алгоритмы с реакциями на некоторые изменения системы представляют собой комбинацию статических и динамических компонентов).

Разница времен доступов к данным в значительной степени влияет на распределение заданий и ресурсов. Для решения этой проблемы необходимо обеспечить согласование местонахождения данных, необходимых для выполнения процессов, и адреса вычислительного узла на котором будет выполняться процесс в соответствии с расписанием. Факторами, которыми определяются физическое согласование данных являются:

- физическое распределение данных и ресурсов, нужных для данного класса заданий;
- трудоемкость действий, выполняемых системой для выполнения задания;
- необходимое время для перемещения данных и заданий в вычислительный узел для выполнения.

В соответствии с вышесказанным имеются:

1. МЗ с описанием отношения между данными и заданиями.
2. МЗ без этого описания.

МД определяет:

- что является источником распределения?
- когда это происходит?
- где это происходит?
- сколько затрачено ресурсов?

С этих позиций можно выделить 2 вида процедур планирования:

1. планирование с перераспределением;
2. планирование без перераспределения.

В соответствии с формами реализации управления различают 3 вида:

- централизованное;
- децентрализованное;
- полудецентрализованное (некоторые узлы обслуживают узлы только своего кластера).

Здесь следует обратить внимание на физические и логические концепции управления. Обычно физически централизованное планирование и управление является также логически централизованным. Но не обязательно наоборот. Например, LINDA является логически централизованным диспетчером, но физически выполняет свои действия по способу децентрализованного распределения.

Инициатором действий планирования и управления могут быть:

- отправитель—инициатор;
- получатель—инициатор.

Но на практике функции инициатора часто выполняются на основе взаимодействия отправителя и получателя. На основании этого можно предложить более точную классификацию:

1. *централизованные алгоритмы* — отправитель-инициатор. Алгоритм SRLD (Static Random Load Distribution) [99];
2. *распределенные алгоритмы* — отправитель-инициатор или получатель-инициатор. Они также называются полными распределенными алгоритмами [113];
3. алгоритмы, где получатель-инициатор с центральным источником загрузки [85, 104].

Существенная разница между этими типами заключается в определении времени, когда происходит распределение. В первом типе распределение происходит сразу после поступления запроса на распределение, а во втором типе распределение происходит с задержкой. В первом типе все ресурсы должны быть в состоянии готовности к работе, пока еще есть задания которые не распределены. Во втором такого нет. Распределенные алгоритмы не обязательно могут использовать системную информацию. Алгоритмы получатель-инициатор, исходя из своих локальных интересов, логически обязательно должны быть распределенными, но они могут использовать и центральные услуги.

4.2.5.2. Классификация по технике решения

В зависимости от целевых функций и требований к результатам планирования, используются следующие методы:

1. Эвристические (Heuristic)[51, 72, 150].

2. Генетические (Genetic) [51, 71, 140].
3. Теории очередей (List Theory) [129, 147, 154, 155].
4. Теории графов (Graph Theory) [62, 74, 86].
5. Математического программирования (Math programming) [76, 80, 104].

Эта современная классификация является развитием идеи, которую впервые сформулировал Stanicovic [143], и связана с терминами On-Line и Off-Line режимов. Существенный вклад в ее развитие был сделан в работе [103]. В данной работе термины On-line и Off-Line представлены как 2 части одного алгоритма. Часть Off-Line разрабатывает некоторые решения, которые необязательно касаются конкретных назначений (расписания). Например: что является объектом планирования, как представляются состояния планировщика во время выполнения, какие правила будет использовать планировщик для конкретных ситуаций и т.д.

- *Общая модель планировщика*

Каждая вычислительная система в процессе функционирования имеет 2 состояния:

* *состояние развития,* характеризующее процесс развития системы.
* *состояние выполнения,* характеризующее процесс выполнения заданий в системе.

Их соединение реализует файл—исполнитель (рис. 4.24).

Процесс развития (ПР) системы включает в себя следующие действия: системный анализ состояния системы, проектирование (планирование) вычислительного процесса и активизация его выполнения. Процесс выполнения (ПВ) системы начинается со старта системы (электрическое включение) или просто с командой строки. Все активные процессы (задания, работы), которые будут выполняться в течении процесса выполнения, должны быть зафиксированы и присутствовать в процессе развития. Таким образом, совершено новые задания, которые не были признаны в ПР, не могут участвовать в ПВ.

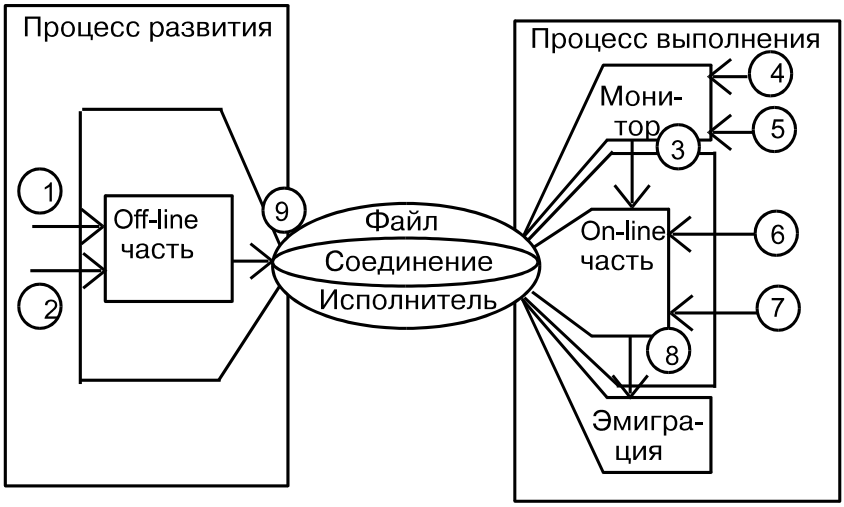

Рис.4.24. Процесс планирования в вычислительной системе

Где:

1 — исходные коды (задания, готовые для выполнения);
2 — информация о характеристиках ресурсов ВС;
3 — On-Line информация;
4 — информация об использовании ресурсов системы;
5 — сообщение о выходе ресурсов из строя;
6 — условия начала выполнения заданий;
7 — условия завершения выполнения заданий;
8 — команды для планирования и распределения.
9 — On-Line алгоритм.

Все результаты действий ПР системы являются источниками информации для Off-Line части планировщика и определяют состояние развития системы — СР. Текущее состояние системы во время ПВ является источником информации для On-Line части планировщика и определяет состояние выполнения — СВ. Планировщик всегда имеет Off-Line ввод и On-Line вывод, который работает динамически и связан с ПВ системы. Взаимодействие между состоянием развития (СР) и состоянием выполнения (СВ) системы также, как и между On-line и Off-Line частями планировщика является строго однонаправленным, так как они не повторяются во времени (любой шаг назад считается как начало нового планирования и выполнения).

В этом представлении файл—исполнитель играет роль соединителя или

прохода из Off-Line в On-line часть планировщика.
Таким образом, планировщик обязательно состоит из 3 основных частей:

- On-Line часть;
- соединитель;
- Off-Line часть.

- *Off-Line часть*

Эта часть не имеет жестких временных ограничений (по сравнению с On-Line частью). Ее задачей является подготовка информации для On-Line части в соответствии с особенностями системы.

Она решает следующие задачи:

- анализ исходного кода, системных требований и других данных о состоянии системы;
- проверку характеристик On-Line части на возможность выполнения плана;
- активизация протоколов поддержки коммуникаций к файл — исполнителю для активизации On-Line части.

Кроме выполнения функций "создавать On-Line часть" и "соединение", которые нужно всегда выполнять, остальные задачи являются необязательными.

Имеются 3 вида соединения:

1. Off-Line часть может действовать как предпроцессор или компилятор для On-line части. В этом случае Off-Line часть собирает данные от обьектов планирования и управления СР системы и преобразует их в удобный вид для On-line части. Потом присоединяет уже сам алгоритм для планирования (который будет работать как On-line часть) и результат передается файл — исполнителю.
2. Планировщик имеет несколько алгоритмов для планирования. Off-Line часть, используя информацию СР системы, выбирает, какой именно алгоритм будет использоваться On-line частью и подготавливает соответствующие данные. Потом передает все файл — исполнителю.
3. Все данные для On-Line алгоритма и сам он генерируется во время выполнения. Off-Line часть действует как диспетчер процессов и становится динамической.

- *Соединитель*

Соединитель On-Line и Off-Line частей всегда содержит On-Line алгоритм в явном или неявном виде. Он управляет процессами и принимает информацию во время выполнения. Поэтому его эффективность определяет эффективность общего процесса планирования.

- *On-Line часть*

Конечными действиями процесса планирования являются

пространственно—временные координаты выполнения процессов в ВС, указания на перепланирование процессов, разрешения на старт и завершение заданий. On-Line часть может быть реализована в простом виде и действовать как *алгоритм загрузки,* использующий ранее составленный план, или в сложном виде, как *адаптивный динамический планировщик,* принимающий решение самостоятельно в динамическом режиме. Другой деятельностью On-Line части является наблюдение за состоянием системы. В общем случае, она действует как реализатор решений Off-Line части, который статически использует данные, трансформированные Off-Line частью и динамические данные, поступающие от системы во время выполнения задания в On-Line части.

Временная сложность On-Line части, работающей в динамическом режиме, должна быть линейной. Большая временная сложность обычно является неприемлемой.

- ***Классификация алгоритмов планирования по способу реализации On-Line и Off-Line частей***

По размерам и соотношению On-Line и Off-Line частей и объема информации, передаваемой от Off-Line части в On-Line часть, все известные алгоритмы можно делить на следующие группы:

1. Алгоритмы классического статического планирования

Определение алгоритмов статического планирования было дано в предыдущем разделе. Эти планировщики имеют почти одинаковые On-Line части, которые имеют малые размеры, функции их ограничены и реализуют простые алгоритмы. Исходной информацией для On-Line части обычно является: адрес назначения процессов на процессоры, порядок их выполнения, момент времени старта. Иногда планировщик высшего уровня определяет для On-Line части параметры взаимодействия процессов в пространстве или во времени. On-Line часть диспетчирует процессы (задания) на детерминированные процессоры и управляет процессом их выполнения точно по плану, разработанному Off-Line частью.

Почти всю работу по планированию и диспетчеризации берет на себя Off-Line часть, которая должна найти оптимальное [59, 99, 102, 134], оптимизированное [138, 143] или приемлемое расписание. Эта задача в общем виде имеет экспоненциальную временную сложность. Точные решения за приемлемое время имеются только для некоторых особых случаев [7]. Для преодоления этой проблемы применяются разные методы: эвристические, генетические, вероятностные.

В некоторых случаях On-Line часть проверяет эффективность предлагаемого расписания на возможность его реализации на наличных ресурсах. Некоторые

алгоритмы способны определить подходящую структуру ресурсов для найденного расписания. В этом случае система является реконфигурируемой. Тогда эта конфигурация также передается в On-Line часть через файл—исполнитель.

2. Планирование с реконфигурацией

Алгоритмы этой группы похожи на предыдущие за исключением того, что они более динамичны по отношению к видам ресурсов и их характеристикам. В таких системах информация по расписанию является более сложной: система планирования может генерировать несколько расписаний для разных конфигураций систем с различными параметрами для On-Line части [105].

On-Line часть также является более сложной, чем при классическом статическом планировании. Она должна определить вид данной системы и, соответственно, выбрать и выполнить подходящий план из вариантов, разработанных в Off-Line части. Несмотря на это, Off-Line часть использует только один фиксированный алгоритм для On-Line части. Как и в классическом статическом планировании, Off-Line часть должна найти оптимальное или приемлемое расписание. В общем виде эта задача также является NP-полной.

3. Статическое планирование с приоритетами

В этих алгоритмах пытаются сжать все требования в один параметр для каждого процесса (задания) — приоритет. Этот параметр является единственным параметром, передаваемым из Off-Line часть в On-Line часть.

On-Line часть является достаточно простой. Принцип планирования осуществляется по приоритетам. В распределенных системах возможно возникновение некоторых проблем, когда выбирается задание с самым высшим приоритетом. Эти алгоритмы имеют полиномиальную временную сложность. Для упрощения этой процедуры, выбор процесса для исполнения выполняет освободившийся процессор. Другой проблемой является ресурсное распределение (балансирование нагрузки) [69] и динамическое изменение приоритетов в системах реального времени [139].

Off-Line часть может просто фиксировать приоритеты заданий из исходного кода заданий (что часто используют в коммерческих системах реального времени). Алгоритмы этого вида обычно являются линейными [138]. Так же возможен вариант динамического вычисления приоритетов или их изменения в соответствии с требованиями выполнения [113] или интересами ВС.

4. Планирование по параметрам

Этот вид планирования используется только для однопроцессорных систем. В качестве информации, передаваемой в On-Line часть, выступает календарь функции [88]. Эти функции используют времена старта и завершения текущих заданий как параметры для вычисления времени старта следующего задания, в то время как

порядок выполнения заданий не изменяется.

Off-Line часть должна подготовить этот календарь и проверить его приемлемость, в соответствии с исходными временными требованиями заданий. А On-Line часть использует календарь для вычисления времени стартов следующих заданий. Сложность алгоритмов On-Line части является линейной.

5. Динамическое планирование с приоритетами

Этот вид планирования имеет много общего со статическим планированием с приоритетами за исключением того, что некоторые параметры (требования) для вычисления приоритетов система получает только во время выполнения задания. Эти требования могут быть приняты планировщиком [48, 93] или с помощью ОС системы в виде признаков режимов работы системы [139].

В случае, когда приоритеты являются динамическими, они похожи на протоколы коммуникации внутри On-Line части. Эта часть тогда делится на 2 раздела: первый занимается текущими заданиями, которые требуют изменения их приоритетов; второй собирает характеристики системы и преобразует их в карты распределения и команды планирования. Таким образом, On-Line часть способна на принятие решения об изменение приоритетов и, соответственно, коррекцию предварительного расписания, полученного от Off-Line части. Но здесь есть одно замечание — для внесение изменения приоритетов требуются некоторые алгоритмы поддержки изменения приоритетов.

6. Планирование в системах с общей памятью

Отличительной особенностью планирования в системах с общей памятью от планирования в системах с распределенной памятью является незначительное время обмена информацией между процессами. В результате этого уменьшается объем информации, необходимой для On-Line части и, следовательно, объем информации, передаваемой в файл—исполнитель.

Off-Line часть проверяет требования заданий [128, 129] и определяет совокупность необходимых им ресурсов [118]. Иногда требования на ресурсы приводят к созданию нового процесса в Off-Line части и она может изменить исходное задание так, чтобы оно могло планировать свое выполнение самостоятельно [128].

On-Line часть распределяет задания среди процессоров в соотвествии с требованиями заданий к ресурсам, полученным от Off-Line части. Благодаря низким затратам на передачу данных в данной системе, планировщик может иметь несколько вариантов расписаний и, следовательно, имеет возможность адаптироваться к разным динамическим изменениям заданий и системы. Самой большой проблемой является разница числа готовых заданий и числа имеющихся процессоров.

7. Балансирование загрузки и планирование с адаптацией

Эти планировщики [69, 114, 131, 138] имеют маленькие Off-Line части и большие On-Line части. Они активно наблюдают за действиями системы и информация, полученная в результате этих наблюдений, используется для создания подходящего расписания [50]. Исходную информацию о системе и некоторые параметры заданий, например, ресурсные требования [114, 131], содержит сам On-Line алгоритм.

Этот вид планировщиков нельзя использовать в системах реального времени из-за того, что алгоритмы с точными, детерминированными временами планирования не будут правильно работать. Эти планировщики работают только с вероятностными моделями [138].

Off-Line частью является компилятор и соединитель, которые преобразуют задания в вид, приемлемый для On-Line части, а также снабжающие задания некоторыми статическими характеристиками для дальнейшего планирования и исполнения.

On-Line часть использует эти характеристики вместе с On-Line требованиями для создания карты распределения и команд планирования. Целью On-Line части часто является оптимизация некоторых характеристик системы. Почти все алгоритмы для оптимизации по этой схеме являются приблизительными из-за NP сложности задач.

8. Планирование с распределением загрузки

Эта группа является особым случаем динамического планирования, когда не используют никакой статической информации о заданиях [60, 74, 148].

Трансформированная информация содержит только On-Line алгоритм. Off-Line часть является минимальной по сравнению с предыдущими видами. On-Line часть берет на себя выполнение всех действий по наблюдению за состоянием системы, заданий и их выполнением.

Соотношение On-Line и Off-Line частей в алгоритмах группы 1-8 показано на рис. 4.25.

4.3. КЛАССИФИКАЦИЯ АЛГОРИТМОВ ПЛАНИРОВАНИЯ ПО КАРТАМ

Классификацию считают хорошей, если она обладает 2-мя следующими свойствами:

1. Способностью различить любые 2 разных алгоритма.
2. Схема классификации должна полностью покрыть все алгоритмы.

Обзор и анализ, выполненный в предыдущей части, показали, что каждая отдельная, выше рассмотренная классификация, не удовлетворяет первому свойству.

Поэтому предлагается классифицировать алгоритмы планирования по обобщенной схеме, названной "классификация по картам". Для лучшего различия алгоритмов и общего представления о процессе планирования предлагается использовать все рассмотренные классификации вместе, объединенные в единую систему, которая используется для анализа каждого алгоритма. Результат такого анализа может быть представлен в двух документах:

Процесс Планирования

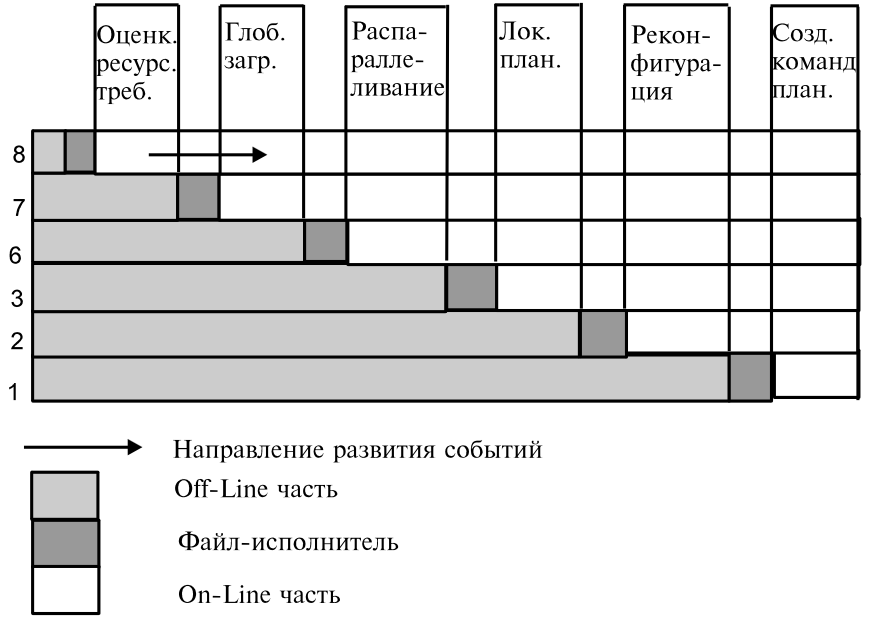

Рис. 4.25 Соотношение On-Line и Off-Line частей в алгоритмах 1-8

8 Алгоритм с распределением загрузки
7 Алгоритм балансирования
6 Алгоритм планирования в системе с общей памятью
3 Алгоритм статического планирования. с приоритетами
2 Алгоритм статической реконфигурации
1 Алгоритм классического статического планирования

1. Карта, где будет указано место нахождения данного алгоритма.
2. Доклад, который содержит информацию о сравнительных характеристиках данного алгоритма с аналогичными алгоритмами по отдельно взятому критерию классификации.

Описание этих действий представлено в рис. 4.26, где анализатор работает по схеме, описанной в таб. 4.3. Все алгоритмы анализируются по 7-и описанным в предыдущей части классификационным признакам.

Представленная таблица может быть использована так же для разработки нового алгоритма планирования. Для этого нужно выполнить следующие процедуры:

1. Определить характеристики ресурсов.
2. Определить характеристики заданий.
3. Определить требования к системе планирования.
4. Определить характер планировщика по его динамичности и временную сложность, которая требуется для алгоритма планирования.
5. По исходным требованиям (цели) задачи планирования определить критерии оптимизации.
6. Сформировать схему планирования с учетом.
7. Выбрать из алгоритмов, имеющихся в базе данных, алгоритм с соответствующими параметрами. В том случае, если не найден ни один алгоритм удовлетворяющий требованиям, разработать новый метод решения.
8. Для применения разработанного алгоритма определить, какие его части (шаги, действия..) являются Off-Line и какие On-Line так, чтобы получить согласование между моделируемой схемой планирования и реальной системой.

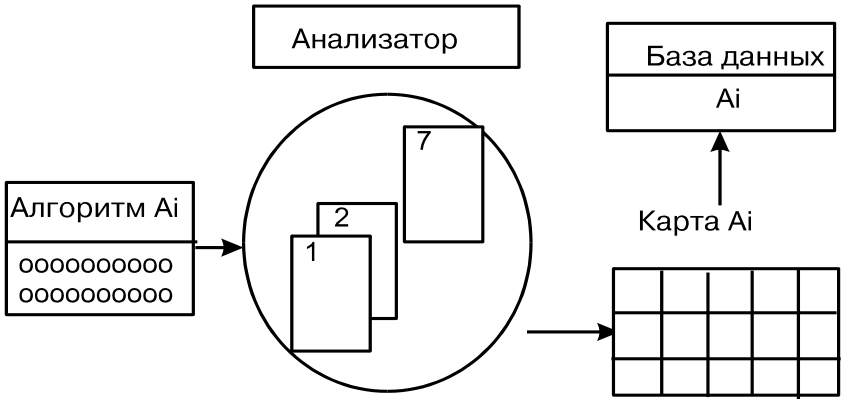

Рис. 4.26. Процесс формирования карт классификации

Для классификации нового алгоритма можно использовать Таблицу характеристики ТХ (табл. 4.4), где каждый из 7-ми уровней соответствует одной характеристике. Любой алгоритм описывается 7-ю признаками, где на каждом уровне он может характеризоваться только одной характеристикой.

Табл. 4.3

1	**Классификация по динамичн.**	Статические (АС)		
		Динамические (АД)		
2	***Классификация по особенностям систем***	***Однородности***	*Однородных* систем (АОС)	
			Неоднородных систем (АНС)	
		Организации памяти	Общей памятью (АСОП)	
			Распределенной памятью (АСРП)	
		Степени параллелизма	Крупно-зернистое планирование(АКЗ)	
			Средне-зернистое планирование(АСЗ)	
			Мелко-зернистое планирование(АМЗ)	
3	***Классификация по характеру заданий.***	**Связанные задания (АСЗ)**	по Спискам	(АСС)
			Кластерное планирование	Вперед(АСКВ)
				Назад(АСКН)
		Несвязанные задания(АНЗ)	Обычное	(АНО)
			Самостоятельное	(АНС)
	Классификация	**Вычислительного**	Время выполнения (ОВВ)	
			Время коммуникаций (ОВК)	

4	***по критериям оптимизаций***	**процесса**						Балансирование (ОВБ)			
		Процесса планирования						Качество планирование (ОПК)			
								Время планирования(ОПВ)			
5	***Классификация по дисциплинам обслуживания очередей***							Бесприоритетные(ДБ))			
								Приоритетные (ДП)			
6	***Классификация по технике решение***	Т1	Т2		Т3		Т4		Т5		Т6
7	***Классификация по организации On-Line, Off-Line***	О1	О2	О3		О4	О5	О6		О7	О8

Где:
Т1 — эвристические, Т2 — генетический, Т3 — линейного программирования, Т4 — теория очередей, Т5 — теория графов, Т6 — приоритетное обслуживание.
О1. Классическое статическое планирование.
О2. Планирование с реконфигурацией в соотвествии виды ресурсов.
О3. Статическое планирование с приоритетами.
О4. Планирование по параметрам.
О5. Динамическое планирование с приоритетами.
О6. Планирование в системах с общей памятью.
О7. Балансирование загрузки и адаптированное планирование.
О8. Планирование распределением загрузки.

Табл. 4.4. Объединенная табл. характеристик алгоритмов планирования

1	(АС)
	(АД)
2	(АОС)
	(АНС)
	(АСОП)
	(АСРП)
	(АКЗ)
	(АСЗ)
	(АМЗ)
3	(АСС)
	(АСКВ)
	(АСКН)
	(АНО)

	(АНС)							
4	(ОВВ)							
	(ОВК)							
	(ОВБ)							
	(ОПК)							
	(ОПВ)							
5	(ДБ) (ДП)							
6	Т1	Т2	Т3	Т4	Т5	Т6		
7	О1	О2	О3	О4	О5	О6	О7	О8