РОССИЙСКАЯ АКАДЕМИЯ НАУК
ВЫЧИСЛИТЕЛЬНЫЙ ЦЕНТР

СООБЩЕНИЯ ПО ПРИКЛАДНОЙ МАТЕМАТИКЕ

И.К. КУПАЛОВ-ЯРОПОЛК, Ю.Е. МАЛАШЕНКО,
И.А. НАЗАРОВА, А.Ф. РОНЖИН

МОДЕЛИ И ПРОГРАММЫ ДЛЯ СИСТЕМЫ УПРАВЛЕНИЯ РЕСУРСОЕМКИМИ ВЫЧИСЛЕНИЯМИ

ВЫЧИСЛИТЕЛЬНЫЙ ЦЕНТР РАН
МОСКВА 2013

УДК 519.85

Ответственный редактор
член-корр. РАН, доктор физ.-матем. наук
Ю.А. Флеров

Представлена общая концептуальная схема построения и функционирования интегрированного комплекса алгоритмов и программ CORSAR. CORSAR предназначен для диспетчеризации ресурсоемких вычислений и распределения ресурсов в гетерогенной специализированной системе. В состав комплекса входит набор оптимизационных моделей, описывающих выполнение разнородных вычислительных работ, допускающих распараллеливание по данным. Построены динамические стратегии, позволяющие решать наименее ресурсоемкие задачи в первую очередь и завершать каждое задание до наступления предписанного директивного срока окончания. Предложенные процедуры реализованы в рамках многоуровневой системы управления, инвариантной к типам вычислительных ресурсов и обеспечивающей их эффективное использование при эксплуатации, масштабируемость при модернизации и развитии.

Ключевые слова: распределенные вычисления, математическое моделирование, принцип гарантированного результата.

Рецензенты: А.В. Лотов,
В.Н. Суриков

Научное издание

Введение

В современной практике для выполнения ресурсоемких заданий используют либо системы распределенных вычислений, такие как грид [1], либо высокопроизводительные вычислительные системы (суперкомпьютеры). По сути и те, и другие являются гетерогенными. Первые представляют собой объединение с помощью глобальных сетей разнообразных типов вычислительных ресурсов от персональных компьютеров до суперкомпьютеров, вторые — вследствие непрерывной модернизации оборудования и использования, одновременно с новым, вполне работоспособного старого.

Объединение вычислительных ресурсов в гриды характерно для зарубежного опыта. Идея грида возникла под влиянием того факта, что в мире существует и продолжает наращиваться огромный парк слабо используемых, недостаточно загруженных вычислительных мощностей, что, естественно, наводит на мысль повысить их отдачу путем объединения в систему коллективного доступа. Грид представляет собой конгломерат идей, методов и технологий, которые позволяют многим пользователям с их разнообразными запросами использовать вычислительные ресурсы разных типов, распределенные на большом географическом пространстве [2].

Подходы к диспетчеризации ресурсов в гриде также существенно отличаются от управления ресурсами единичного компьютера. Планировщики обработки заданий рассматривают грид как большой пул ресурсов, к которым они имеют исключительный доступ. Фактически управление выполнением отдельного задания осуществляется независимо; нагрузка, приоритеты, варианты расписания, порождаемые другими, во внимание не принимаются. Отсутствие координирующего механизма приводит к возникновению узких мест, равно как и к недоиспользованию части ресурсов. В свою очередь конечные

потребители посылают заявки в LRMS, не имея сведений о времени отклика или сервисных возможностях. Иногда задачи находятся в очереди часами, что ведет к потере качества обслуживания.

Существующие системно-центрические (system-centric) подходы к планированию [3] не учитывают требования потребителей, заявки в гриде размещаются в соответствии с системными параметрами, которые касаются загрузки ресурсов или пропускной способности системы. Планировщики уделяют внимание либо минимизации времени отклика (response time — сумма времени ожидания и времени фактического выполнения задания), либо максимизации общей загрузки ресурсов. В такой ситуации целью планирования оказывается улучшение состояния системы в целом, но не требования, предъявляемые отдельными потребителями. Для того чтобы минимизировать время обслуживания, необходимо использовать стратегии планирования, которые, с одной стороны, учитывают интересы пользователей и приоритеты задач, с другой — информацию о диспетчеризации и состоянии ресурсов.

В [4, 5] предложена динамическая имитационная модель, которая опирается на подробное алгоритмическое описание процесса коллективного взаимодействия удаленных ресурсов и пользователей через телекоммуникационную сеть. Проблема сводится к задаче оптимального управления марковскими последовательностями, для ее решения используются оригинальные адаптивные алгоритмы.

В нашей стране интенсивно развивается проектирование и создание суперкомпьютеров. Внедрение и использование последних преследует различные цели — от обучения студентов до решения узковедомственных задач. Многие специализированные вычислительные системы (СВС) ориентированы на работу с определенными видами заданий и имеют уникальную

совокупность составных частей: общее число и типы процессоров, архитектуру, программное обеспечение, базы данных и пр. Однако основная схема построения СВС обычно остается стандартной: каждая из них состоит из управляющих подсистем, на которых разворачивается ядро системы управления, и множества непосредственно предназначенных для решения задач вычислительных узлов. Последние нередко дополнительно оснащаются специальными устройствами (далее ускорителями), позволяющими многократно увеличить быстродействие вычислительных алгоритмов. В качестве элементной базы таких ускорителей согласно [6] могут использоваться заказные СБИС (ASIC), БМК (eASIC), системы-на-кристалле (SoC), ПЛИС (FPGA), графические сопроцессоры GPGPU. Скорость обработки одного и того же задания различными типами устройств может отличаться в несколько раз. Некоторые ускорители предназначены для работы только с конкретными алгоритмами, поэтому с их помощью эффективно решаются только определенные классы задач.

Далее рассматривается проблема управления гетерогенной высокопроизводительной СВС, выполняющей ресурсоемкие задания, относящиеся к специальному классу, процедуры обработки которых можно определить как случайный перебор с неизвестным исходом [7]. Решение каждой такой задачи осуществляется переборными алгоритмами и сводится к поиску фрагмента с наперед заданными свойствами — уникального фрагмента, — в большом массиве исходных данных. Последний состоит из отдельных, неделимых, одинаковых по размеру, содержательно значимых фрагментов. Заявки поступают в систему в произвольные моменты времени, исходный массив для каждой из них становятся известен сразу после появления задания в СВС. Задача считается решенной, как только в предъявленном наборе данных удается выявить уникальный

фрагмент. В противном случае необходимо убедиться, что таковой в массиве отсутствует, и тем самым выполнить задание. Кроме того, обработка каждой из поступивших в СВС заявок должна быть завершена в максимально сжатые сроки.

Задачи бывают разных видов, перебор соответствующего массива исходных данных для каждой из них требует большого, точно не известного, объема вычислений. В общем случае можно указать только верхнюю оценку для величины вычислительных затрат, необходимых для выполнения конкретного задания; имеет ли решение соответствующая содержательная задача — неизвестно. Таким образом, при планировании работ возникает неопределенность, связанная с моментами поступления заявок, с составом и длительностью обработки заданий, а также с потенциальной возможностью получить решение.

Описанный выше специальный класс задач, в [8] определен как citu-задания, citu-задачи; от английского термина: computation - intensive - task - under - uncertainty. Следует выделить целый ряд существенных особенностей, присущих изучаемому классу: все citu-задачи выполняются в режиме реального времени, допускают распараллеливание по данным и являются ресурсоемкими; их решения одинаково важны и должны быть получены — каждое в отдельности и все в совокупности — как можно быстрее.

В отечественной практике существуют примеры успешных реализаций систем управления ресурсоемкими вычислениями, например, программный комплекс «Пирамида» [9, 10]. Последний имеет иерархическую древовидную архитектуру и обеспечивает высокую надежность вычислений за счет дублирования. К недостаткам первых версий «Пирамиды» можно отнести необходимость отдельного конфигурирования и разворачивания комплекса под каждое вычислительное задание. Конкретное задание получает некоторый статичный набор вычис-

лительных ресурсов, в рамках которого выполняется. Динамическое перераспределение ресурсов между заданиями в процессе обработки невозможно. При некоторых разрешенных параметрах конфигурации запуска, дублирование может составлять половину от общего количества работ, тем самым в два раза снижая эффективность вычислений. В последних версиях комплекса указанные недостатки частично исправлены, однако концепция "распределения машинного времени" сохранена.

Современные СВС могут иметь в своем составе тысячи узлов, оснащенных десятками вычислительных ядер в универсальной части и специализированными ускорителями различных типов. Для больших гетерогенных СВС оказывается неэффективным и сложно реализуемым единое централизованное планирование работы отдельного вычислительного элемента. Практика показывает, что массово-параллельные системы диспетчеризации, построенные с использованием архитектуры, подобной интерфейсу mpi [11-13], становятся малопригодными для крупных СВС.

Использование комплекса CORSAR позволяет избежать потери производительности при увеличении числа и мощности вычислительных ресурсов, наблюдается практически линейный рост кривой "эффективность — быстродействие". Перейдем к описанию интегрированного комплекса CORSAR, ориентированного на многократное выполнение различных видов citu-заданий.

1. CORSAR — комплекс моделей и программ для решения ресурсоемких вычислительных задач

1.1. Общие положения.

При проектировании и создании комплекса CORSAR использована новая концептуальная схема планирования и управления обработкой citu-заданий. В рамках предлагаемого подхода вводится понятие вычислительной *работы*, которая выполняется переборными алгоритмами и измеряется в числе специализированных элементарных вычислительных операций (СЭВ-операций). Каждая СЭВ-операция состоит в просмотре одного фрагмента данных определенного вида и проверке его уникальности.

В комплексе CORSAR, согласно концептуальной схеме, планирование *работ* осуществляется на основании результатов, полученных с помощью математических моделей. Весь процесс управления СВС разбивается контрольными точками на отдельные этапы — плановые периоды или операционные окна, а распараллеливание и распределение *работ* происходит с учетом текущей производительности ресурсов.

Для планирования и оценки времени пребывания в системе каждого конкретного citu-задания вводится понятие директивного срока окончания. Предполагается, что при поступлении заявки-задания в первую очередь определяется возможное время ее завершения в наихудшем случае, но с учетом объективных показателей загруженности СВС и имеющихся требований. Если предварительная оценка — срок, к которому citu-задание может быть завершено, устраивает пользователя, то оно принимается для обработки и ему присваивается директивный срок окончания, совпадающий с прогнозируемым. Таким образом, директивный срок окончания (ДСО) — это установленный администратором и согласованный с пользователем

момент календарного времени, до наступления которого citu-задача должна быть решена. Планирование *работ* осуществляется исходя из предположения, что события будут развиваться по наихудшему сценарию. В дальнейшем ДСО используются как ограничения при формировании диспетчерских стратегий совместного выполнения всех заданий, находящихся в СВС.

Поиск решений citu-задач ведется переборными алгоритмами. При этом массив исходных данных любого задания допускает разбиение на отдельные независимые части, которые не требуют синхронизации в процессе обработки. Декомпозиция по исходным данным происходит сразу на входе в СВС, что значительно упрощает написание прикладных программ и повышает их надежность, поскольку процесс распараллеливания полностью контролируется системой планирования CORSAR. Снятие требования синхронизации позволяет применить методы оптимизации при планировании, организовать гибкое динамическое управление и повысить эффективность использования вычислительных ресурсов.

Для повышения результативности работы СВС, на этапе развертывания системы CORSAR, для каждого из имеющихся алгоритмов и соответствующих ему типов вычислительных ресурсов составляется таблица производительности. Данные из таблицы анализируются системой управления на этапе планирования и используются для определения объемов текущих *работ*, а также при построении прогноза длительности решения каждой задачи. Наличие и постоянное обновление такой информации позволяет системе управления предоставлять каждому заданию наиболее подходящие ресурсы среди всех доступных на любом шаге планирования. Предполагается, что в случае необходимости для некоторых *работ* можно использовать ресурсы с меньшей производительностью, но, для увеличения общей эффективности СВС, последние следует свое-

временно освобождать при появлении задания, которое может быть обработано с их помощью быстрее.

1.2. Трехуровневая организация комплекса.

Архитектура программного комплекса CORSAR предполагает разбиение процесса диспетчеризации СВС на три уровня, что позволяет последовательно и в полной мере реализовать предлагаемую концепцию управления *работами*. Рассмотрим подробнее состав и функции различных уровней диспетчеризации. Важнейшей частью верхнего (первого) уровня комплекса CORSAR является программный компонент планирования и анализа (ПК-план). В ПК-план подготавливается список заданий и определяется размер подмассивов фрагментов данных, которые в дальнейшем составят пакет текущих *работ*. В контрольный момент пакет поступает в промежуточный буфер данных и начинается его обработка СВС. По завершении выполнения текущих *работ* принимается очередной "план выполнения" , а затем вся процедура повторяется с учетом изменения состава задач и состояния СВС.

ПК-план представляет собой программную реализацию ряда математических моделей, в которых обработка заданий в СВС определяется системой квазидинамических уравнений и неравенств, записанных в дискретном времени. Разработанные математические модели реализованы в виде загружаемых ядер, которые находятся в библиотеке комплекса CORSAR и используются ПК-план. В ПК-план строятся гарантированные оценки [14], и формируется оптимизационная модель [15], учитывающая всю текущую информацию (длительность пребывания задания в системе, объем обработки данных, и др.). В процессе решения оптимизационной задачи определяются неулучшаемые оценки длительности пребывания каждого конкрет-

ного задания в СВС и необходимые параметры управления системой. В результате реализуется динамическая стратегия планирования и правило выполнения пакета заданий, при котором в первую очередь завершаются наименее ресурсоемкие задания, требующие небольших объемов *работы*. Указанное правило выполнения текущих *работ* далее будет обозначаться аббревиатурой SWAP от английского short - work - ahead - performance.

На верхнем уровне происходит формирование пакета текущих работ (ТР-пакета), а именно: определяется список заданий, для каждого из которых назначается число фрагментов данных, предназначенных для обработки в ближайшем операционном окне. При планировании учитывается общая загрузка СВС и состояние (работоспособность, производительность) всех вычислительных ресурсов. Основная цель управления — всесторонний учет интересов пользователей: *все и каждое задание должны быть завершены в предписанные сроки, а задачи, требующие небольших ресурсных затрат должны быть решены в первую очередь* (соблюдение правила SWAP).

Предполагается, что длительность пребывания задания в СВС: во-первых, должна напрямую зависеть от *работы*, фактически необходимой для его завершения, хотя объем последней априори неизвестен; во-вторых, не может превышать предписанного ДСО.

На первом уровне осуществляется:

— анализ, учет и хранение данных всех поступивших и выполняемых заданий;

— контроль за состоянием вычислительных ресурсов;

— планирование и формирование ТР-пакета;

— загрузка ТР-пакета в буфер текущих работ (ТР-буфер), который управляется со второго уровня CORSAR;

— пересылка подзаданий из ТР-буфера по запросам для

выполнения группам вычислительных узлов.

На втором уровне реализуется управление обработкой ТР-пакета, находящегося в ТР-буфере. Подзадания из ТР-буфера распределяются между группами вычислительных узлов. Программа-диспетчер второго уровня, асинхронно и независимо, запрашивает у ПК-план, а затем распределяет подзадания из полученного ТР-пакета между всеми работоспособными узлами подконтрольной группы. На данном уровне основной целью диспетчеризации является эффективное использование ресурсов и завершение *работ* из ТР-буфера в течение планового периода или быстрее. Для повышения результативности СВС, все *работы* назначаются к исполнению вычислительными ресурсами того типа, на которых достигается максимальная производительность для данного вида задач. При таком распределении минимизируется время выполнения *работ*, находящихся в ТР-буфере. В результате задача быстродействия и повышения эффективности решаются комплексно и одновременно.

На третьем (нижнем) уровне *работа* выполняется: производится просмотр и обработка фрагментов данных на вычислительном узле. Длительность *работы* не планируется, а момент окончания и начало нового цикла только фиксируются. Фактически распараллеливание происходит на данном уровне при назначении и исполнении подзаданий на выделенных ресурсах. Здесь же осуществляется контроль за функционированием ресурсов и запуск соответствующих алгоритмических процедур. Анализируется весь ход выполнения *работы*, происходит отбор результатов счета.

На нижнем уровне располагаются специализированные программные компоненты — ресурсные агенты, каждому из которых соответствует конкретный тип ресурса в рамках одного фиксированного вычислительного узла. Ресурсный агент — это программная реализация стандартного интерфейса доступа к

соответствующей ресурсной единице, которую далее будем называть единичным вычислительным модулем (ЕВ-модуль). Например, для специализированных ускорителей в качестве такой единицы может рассматриваться каждая микросхема (ПЛИС, кристалл), поскольку она является наименьшим физическим объектом, допускающим автономное управление своим множеством состояний. Введение и постоянное обновление ресурсных агентов в комплексе CORSAR позволяет управлять имеющимися и вновь подключенными вычислительными ресурсами в определенном смысле независимо от физико-технической природы последних.

На третьем уровне:

— подзадания загружаются и обрабатываются на выделенных ЕВ-модулях;

— контролируется процесс выполнения *работ*: в случае обнаружения уникального фрагмента решение передается на вышестоящий уровень;

— отслеживается состояние вычислительных ресурсов, осуществляется "перепрошивка" ЕВ-модулей, а также их "перезапуск" в случае сбоя;

— фиксируется время завершения *работ*.

Общая схема комплекса CORSAR изображена на рис.1.

1.3. Состав программных компонент комплекса.

К программным компонентам верхнего (первого) уровня относятся:

а) центральная база данных заданий (БД-заданий), которая содержит постоянно обновляющуюся информацию:

— о всех поступивших заявках-заданиях;

— о типах и состояниях всех вычислительных ресурсов СВС;

— о привязке видов задач к типам вычислительных ресурсов и текущие данные о производительности элементов СВС;

б) служба регистрации и анализа задач, которая обеспечивает:

— прием поступающих заданий;

— распознавание видов задач;

— проверку корректности формата поступившего задания и оценку размера возможного объема *работ*;

— внесение полученной информации в БД-заданий;

в) ПК-планирования и анализа (ПК-план, "планировщик") где, в соответствии с актуальной политикой диспетчеризации, определяется:

— очередное операционное окно - плановый период;

— список заданий и размеры поднаборов данных для ТР-пакета, после чего производится загрузка последнего в ТР-буфер;

г) набор ("библиотека") динамически подгружаемых ядер для ПК-план, состав которого:

— определяет различные политики управления выбором подзаданий для обработки в текущем операционном окне;

— используется в ПК-план при планировании распределения вычислительных ресурсов;

д) центральный ПК-диспетчер обработки, в котором:

— происходит прием и обслуживание запросов ПК-диспетчеров групп вычислительных узлов;

— осуществляется контроль выполнения подзаданий, находящихся на обработке во всех группах;

— синхронизируется процесс выполнения *работ* различными группами вычислительных узлов;

— передаются команды администратора CORSAR.

Второй (средний) уровень управления группой вычислительных узлов содержит один основной компонент — ПК-дис-

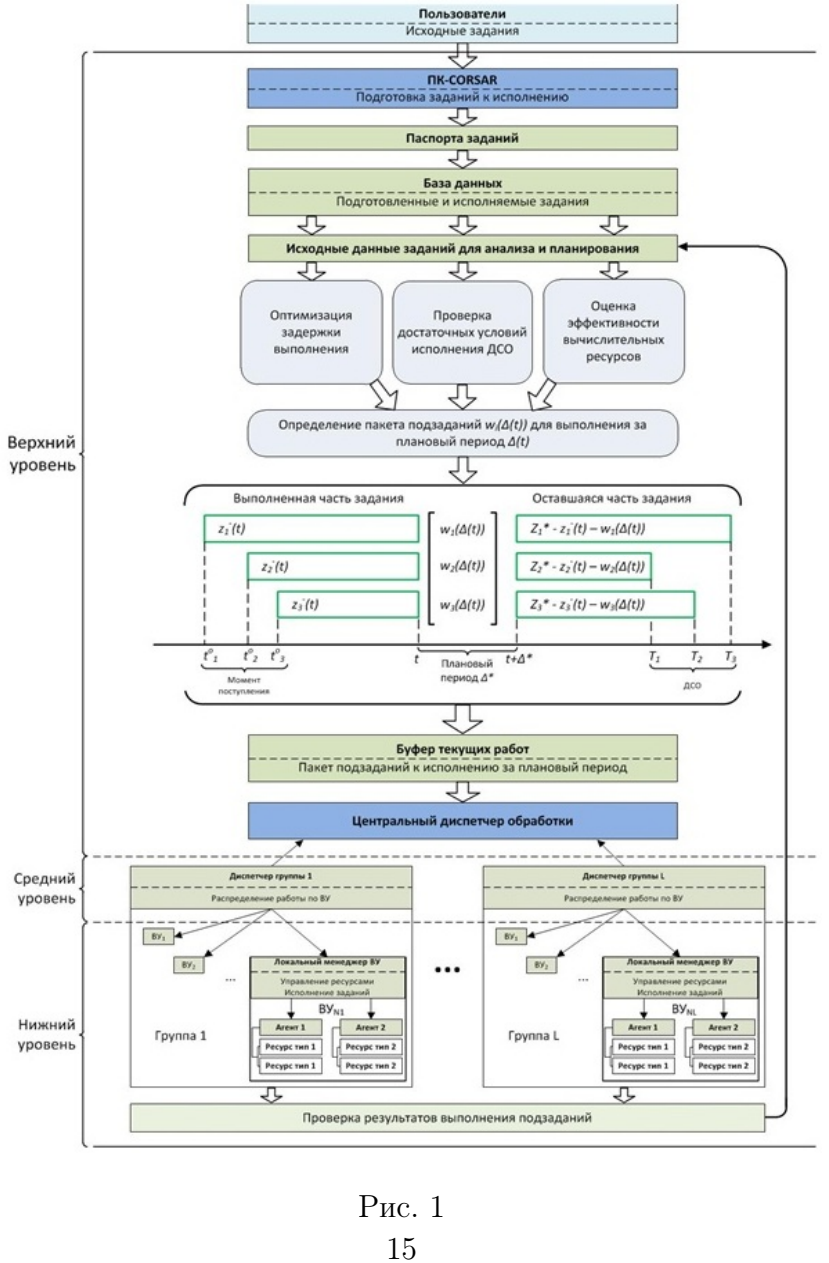
Пользователи
Исходные задания
ПК-CORSAR
Подготовка заданий к исполнению
Паспорта заданий
База данных
Подготовленные и исполняемые задания
Исходные данные заданий для анализа и планирования
Оптимизация задержки выполнения
Проверка достаточных условий исполнения ДСО
Оценка эффективности вычислительных ресурсов
Определение пакета подзаданий $w_i(\Delta(t))$ для выполнения за плановый период $\Delta(t)$
Верхний уровень
Выполненная часть задания
Оставшаяся часть задания
$z_1^-(t)$
$z_2^-(t)$
$z_3^-(t)$
$w_1(\Delta(t))$
$w_2(\Delta(t))$
$w_3(\Delta(t))$
$Z_1^* - z_1^-(t) - w_1(\Delta(t))$
$Z_2^* - z_2^-(t) - w_2(\Delta(t))$
$Z_3^* - z_3^-(t) - w_3(\Delta(t))$
t^0_1 t^0_2 t^0_3
Момент поступления
t
Плановый период Δ^*
$t+\Delta^*$
T_1 T_2 T_3
ДСО
Буфер текущих работ
Пакет подзаданий к исполнению за плановый период
Центральный диспетчер обработки
Средний уровень
Диспетчер группы 1
Распределение работы по ВУ
Диспетчер группы L
Распределение работы по ВУ
$ВУ_1$
$ВУ_2$
Локальный менеджер ВУ
Управление ресурсами Исполнение заданий
$ВУ_{N1}$
$ВУ_{NL}$
Агент 1
Агент 2
Ресурс тип 1
Ресурс тип 2
Нижний уровень
Группа 1
Группа L
Проверка результатов выполнения подзаданий

Рис. 1

петчер группы, который:

— следит за состоянием вычислительных ресурсов всех узлов группы;

— запрашивает у центрального диспетчера и получает подзадания для групп узлов;

— управляет распределением *работ* на все вычислительные узлы группы и контролирует их выполнение.

Локальный ПК-менеджер вычислительного узла и ресурсный агент образуют компоненты третьего (нижнего) уровня управления отдельным вычислительным узлом. Локальный ПК-менеджер:

— контролирует состояние всех ресурсов вычислительного узла (В-узла);

— принимает от диспетчера группы *работы*, обеспечивает их выполнение на данном узле и возвращает результаты диспетчеру группы;

— выполняет на В-узле команды администратора комплекса CORSAR ;

— представляет информацию о состоянии В-узла;

— при необходимости осуществляет отключение и перезапуск В-узла.

ПК-ресурсный агент:

— постоянно контролирует состояние и осуществляет непосредственное управление всеми экземплярами соответствующего типа ресурсов на В-узле;

— представляет стандартный интерфейс для доступа к определенному типу вычислительных ресурсов, например, CPU, специализированным ускорителям на базе FPGA, GPGPU, eASIC, ASIC.

Для каждого типа ресурсов вычислительного узла выполняется один экземпляр соответствующего ПК-ресурсного агента, обеспечивающего контроль состояний и доступ ко всем

устройствам данного типа.

В комплексе CORSAR реализованы автоматизированные *рабочие места* администратора и операторов. Соответствующий интерфейс позволяет осуществлять:

— настройку параметров функционирования СВС;

— просмотр информации о текущем состоянии разнотипных вычислительных ресурсов;

— контроль и управление процессами обработки заданий;

1.4. Общая схема функционирования комплекса.

Рассмотрим функционирование комплекса CORSAR и организацию взаимодействия программных компонент разных уровней.

Заявки-задания поступают в СВС после загрузки программами службы регистрации CORSAR. В последней имеются специальные, динамически подключаемые, библиотеки распознавания видов задач и проверки корректности формата загружаемых заданий. Задачи, вид которых определить на удалось, и задания, содержащие некорректные данные, на обработку не допускаются, и автоматически возвращаются на входное устройство для анализа оператором СВС. Заявки, успешно прошедшие предварительную проверку, помещаются в БД-заданий и становятся доступны ПК-план системы управления.

Политики диспетчеризации определены в сменных ядрах, которые оформлены в виде динамически подгружаемых библиотек, и реализуют определенный программный интерфейс в составе ПК-план. В настоящее время реализованы ядра, позволяющие уменьшить производственные потери и повысить эффективность СВС, кроме того поддерживается дисциплина диспетчеризации, при которой в первую очередь обслуживаются задания, требующие наименьших затрат ресурсного времени.

На практике соблюдение правила SWAP позволяет завершать наиболее ресурсоемкие задания последними, тем самым уменьшая время ожидания для остальных. Изменение политики диспетчеризации может инициировать администратор CORSAR.

На очередном цикле работы ПК-план составляет ТР-пакет, т.е. определяет какие *работы* должны быть выполнены в заданном операционном окне. Формирование ТР-пакета происходит в соответствии с актуальной политикой диспетчеризации, при этом учитывается доступность вычислительных ресурсов каждого типа, а также их текущее состояние. Сформированный ТР-пакет помещается в ТР-буфер.

Просмотром фрагментов данных, входящих в ТР-пакет, управляет центральный диспетчер обработки. Данный программный компонент принимает запросы от ПК-дипетчеров групп узлов и подготавливает для них подпакеты текущих работ (ТР-подпакеты). В запросах диспетчеров групп узлов указываются временной интервал для выполнения *работы*, количество и тип вычислительных ресурсов, планируемых для использования. Для различных типов ресурсов ПК-диспетчеры групп узлов формируют отдельные запросы. При обработке такого запроса центральный диспетчер подбирает ТР-подпакет на основе "жадного"алгоритма: наборы фрагментов выделяются в порядке убывания скорости их обработки на данном типе ресурсов. В ответ на любой из запросов ПК-диспетчеров групп формируется один ТР-подпакет, содержащий множество подналоров фрагментов различных заданий, размеры которых могут быть произвольными. Набор фрагментов подбирается таким образом, чтобы объем ТР-подпакета, сформированного по текущему запросу, максимально соответствовал запрошенному, но не превышал его.

Текущая информация о скорости обработки заданий разных видов на различных типах ресурсов (производительность)

хранится и постоянно обновляется в центральной базе данных. После передачи ТР-подпакета группе В-узлов центральный диспетчер устанавливает контрольные интервалы. По истечении отпущенного интервала времени посылается запрос диспетчеру соответствующей группы узлов о состоянии выделенного поднабора текущих *работ*. В случае превышения предписанных сроков окончания *работ*, в зависимости от текущей политики диспетчеризации:

— может быть установлен новый срок завершения;

— подзадание для данной группы узлов отменяется, а данные из ТР-подпакета возвращаются в ТР-буфер, а затем перераспределяются заново по новым запросам ПК-диспетчеров групп узлов.

Если одна из подзадач решена, т.е. искомый уникальный фрагмент найден в данном ТР-пакете, то центральный диспетчер:

— удаляет все связанные с данной задачей фрагменты из ТР-буфера;

— дает команду о прекращении обработки соответствующих подзаданий исполняющим их диспетчерам групп, которые участвовали в решении этой задачи;

— помечает в БД-заданий задачу как выполненную и сообщает в ПК-план об успешном завершении задания.

В процессе работы центральный диспетчер посылает контрольный сигнал в ПК-план, если общий текущий объем работ в ТР-буфере уменьшается до некоторого критического значения. Описанная ситуация может возникнуть в момент решения некоторой задачи, набор фрагментов которой занимал значительный объем ТР-буфера.

Кроме того, в ПК-центрального диспетчера формируется отдельная очередь из запросов на вычислительную *работу* для определенных типов ресурсов, для которых задания отсутству-

ют и сформировать пакет работ невозможно. "Отложенные" запросы обслуживаются центральным диспетчером при появлении подходящих заданий.

ПК-диспетчер группы узлов обеспечивает централизованное управление локальной группой В-узлов, а именно:

— учитывает количество работоспособных и текущее состояние вычислительных ресурсов каждого типа;

— формирует и передает центральному диспетчеру запросы на получение *работ* для всех типов ресурсов подчиненной группы;

— при получении ТР-подпакета производит распределение поднаборов фрагментов между ПК-менеджерами вычислительных узлов;

— обеспечивает выполнение команд и предоставление информации по запросам от операторов CORSAR.

Во всех запросах-требованиях на получение заданий определенного вида ПК-диспетчер группы узлов указывает общее количество работоспособных ресурсов (ЕВ-модулей) соответствующего типа, а также величину используемого рабочего интервала (цикла). На один конкретный запрос ПК-диспетчер группы получает объем *работы*, который может быть выполнен на всех ресурсах соответствующего типа за один рабочий цикл. Такая диспетчеризация позволяет сократить количество обращений к центральному диспетчеру и снизить нагрузку на системы управления верхнего уровня. Запросы-требования для разных типов ресурсов формируются в ПК-диспетчер независимо друг от друга. В ПК-диспетчер группы не происходит формирование нового запроса, пока не будет выделена *работа* по предыдущему обращению для загрузки ресурса соответствующего типа.

При получении ТР-подпакета ПК-диспетчер группы постепенно распределяет *работы* между В-узлами своей группы.

На каждый единичный ресурс (ЕВ-модуль) загружается порция фрагментов данных, которая может быть обработана за один рабочий цикл. В комплексе CORSAR длительности рабочих интервалов ПК-диспетчера группы и вычислительного узла являются параметрами управления, и могут изменяться администратором. Указанные величины характеризуют максимальный объем вычислений, который может быть потерян и потребуется повторная обработка соответствующих фрагментов данных. Прерывание вычислений и потеря *работы* могут произойти при выходе из строя элемента системы управления среднего или нижнего уровня. Отказ элемента может наступить в результате программного и/или аппаратного сбоя.

ПК-диспетчер группы контролирует процесс исполнения заданий на всех подчиненных ему вычислительных узлах и накапливает промежуточные результаты до завершения полной обработки всего ТР-пакета. При выходе из строя узлов или отдельных ресурсов диспетчер перераспределяет задания внутри группы. ПК-диспетчер группы связывается с центральным диспетчером и отказывается от выполнения ТР-подпакета, если завершить работу невозможно, например, при выходе из строя всех ресурсов какого-то типа.

В случае, когда для какой-либо из задач искомый уникальный фрагмент обнаружен, ПК-диспетчер группы:

— помечает у себя все оставшиеся фрагменты как просмотренные;

— передает найденное решение центральному диспетчеру;

— перераспределяет оставшуюся часть незавершенной *работы* внутри группы.

По завершении группой узлов определенного объема *работ*, соответствующих конкретному типу ресурсов, диспетчер группы, не дожидаясь полного выполнения всего ТР-пакета, формирует новый запрос-требование. Такая стратегия позволяет:

получать новые порции *работ* не завершив полностью предыдущие; избегать простоев или неэффективного использования вычислительных ресурсов.

Центральный диспетчер постоянно контролирует работоспособность ПК-диспетчеров групп узлов. В случае выхода из строя любого из них, в автоматическом режиме назначается новый управляющий узел, на котором запускается программный компонент диспетчера для данной группы.

ПК-локальный менеджер осуществляет непосредственное управление вычислениями на отдельном В-узле, постоянно следит за состоянием всех ЕВ-модулей узла и контролирует их функционирование. Управление специализированными ускорителями вычислений осуществляется через соответствующих ресурсных агентов. В случае необходимости предпринимаются попытки восстановления работоспособности ускорителей, для чего у ПК-агента инициируется механизм восстановления обслуживаемого ресурса. Локальный менеджер В-узла принимает от ПК-диспетчера группы команды на выполнение определенного алгоритма с указанным диапазоном входных параметров и типов используемого ресурса. Менеджер подбирает конкретный ЕВ-модуль в узле и на нем запускает алгоритм, функционирование которого в дальнейшем контролируется. При выборе конкретных ЕВ-модулей действия ПК-локального менеджера направлены на то, чтобы уменьшить количество необходимых переинициализаций ресурса (например, смены "прошивки"ПЛИС). При любом сбое ПК-менеджер пытается перезапустить соответствующий алгоритм, например, на другом подконтрольном ему ЕВ-модуле.

ПК-диспетчер группы извещается о невозможности обработки данного подзадания, если все попытки терпят неудачу.

Здесь важно отметить, что объем *работы* (размер подзаданий) не превышает числа СЭВ-операций, которые могут быть

выполнены данной группой узлов за один рабочий цикл. Другими словами, суммарный размер фрагментов данных для всех заданий, переданных на исполнение, не превышает числа вычислительных операций, которые могут быть выполнены данной группой узлов за один рабочий интервал. Процесс просмотра каждого фрагмента документируется и сохраняется в базе данных. В случае экстренного (аварийного) отключения и последующего возобновления работы СВС не возникает необходимости повторного выполнения вычислений для обработанных ранее фрагментов, что уменьшает потери и повышает эффективность СВС в целом.

Перейдем к описанию математических моделей, которые используются для оперативного планирования работ и реализации различных диспетчерских политик.

2. Общее описание математических моделей

2.1. Основные обозначения и предположения.

В блок планирования и анализа комплекса CORSAR заложен целый ряд моделей и программно-реализованных методов оптимизации. Далее при математическом описании будем говорить об абстрактной гетерогенной высокопроизводительной СВС, в которой в условиях неопределенности выполняются разнородные *работы*.

В рамках предположений, формирующих модель, считается, что указанная СВС состоит из центрального управляющего устройства (ЦУП-устройства) и набора независимых исполняющих единичных вычислительных модулей (ЕВ-модулей) различных конструктивных типов. Под ЦУП-устройством будем понимать управляющую структуру СВС верхнего уровня. ЕВ-модули подчиняются непосредственно соответствующему локальному менеджеру вычислительного узла.

В комплексе CORSAR вся исходная и текущая информация о citu-заданиях, как вновь поступивших, так и находящихся в обработке, помещается и далее хранится в БД-заданий. В математических моделях динамика обработки массивов исходных данных описывается системой неравенств и конечно-разностных уравнений в дискретном времени. Шаг по времени называется плановым периодом или операционным окном.

В контрольной точке принятия решения t по команде администратора CORSAR в ПК-план анализируются текущие данные о выполнении всех заданий, находящихся в СВС. Здесь t — начальный момент очередного планового периода. Далее администратор (специалист и/или алгоритмическая процедура) выбирает число единиц календарного времени $\Delta(t)$ — длительность планового периода, в ПК-план формируется ТР-пакет, который состоит из подмассивов (поднаборов) неделимых фрагментов данных тех подзаданий, которые будут обрабатываться

в СВС в течение $\Delta(t)$. ПК-план помещает ТР-пакет в специальный ТР-буфер и, начиная с момента t, СВС выполняет плановую *работу* на всех работоспособных ЕВ-модулях.

В рамках рассматриваемых моделей считается, что citu-задания поступают в СВС в произвольные моменты времени без каких-либо особых признаков, отношений предпочтения или предписанного порядка обслуживания.

Далее в тексте верхний индекс любой переменной будет всегда относиться к типу вычислительного модуля, а нижний (один или два) в зависимости от рассматриваемого случая — отвечать номеру и виду задачи соответственно. Для того, чтобы избежать путаницы, второй нижний идентификатор, соответствующий виду задачи, помещен в скобки. При модельном описании СВС и комплекса CORSAR используются следующие обозначения:

H — ЦУП-устройство, в котором происходит анализ поступающих заданий, определяется порядок их выполнения, а также осуществляется контроль процесса обработки;

E — набор (множество) ЕВ-модулей различных типов.

Пусть M — общее число типов ЕВ-модулей, из которых состоит рассматриваемая СВС;

e^m — отдельный ЕВ-модуль (устройство) m-го типа;

E^m — множество e^m ЕВ-модулей m-го типа, $m = \overline{1, M}$, следовательно, множество **E** всех ЕВ-модулей в СВС можно записать в виде объединения

$$\mathbf{E} = E^1 \bigcup E^2 \bigcup ... \bigcup E^M.$$

Под специализированной элементарной вычислительной операцией (СЭВ-операция) будем понимать просмотр отдельного неделимого содержательно значимого фрагмента данных определенного вида и проверку его уникальности. Считается, что в комплексе CORSAR любой ЕВ-модуль может обрабатывать

задания по крайней мере одного вида; производительность конкретного ЕВ-модуля при выполнении citu-заданий разных видов неодинакова и может меняться (перегрев оборудования, изменение тактовой частоты или загрузка другого программного обеспечения); с течением времени ЕВ-модули могут выходить из строя.

В зависимости от контекста через t обозначим либо текущий календарный момент времени, либо контрольную точку принятия решения. Таким образом, t не является переменной в традиционном смысле, это обозначение определенных моментов времени в модельном описании процесса.

Пусть $p_k^m(t)$ — производительность ЕВ-модуля m-го типа при обработке задания k-го вида, т.е. модуль e^m может выполнять в единицу времени $p_k^m(t)$ СЭВ-операций k-го вида, начиная с момента t;

$R^m(t)$ — общее число работоспособных ЕВ-модулей m-го типа на момент времени t.

В СВС предусмотрена возможность одновременной обработки различных фрагментов разнородных заданий на всех работоспособных в данный момент ЕВ-модулях. Отдельное задание разбивается на подзадания — наборы фрагментов, которые выполняются самостоятельно или в составе некоторого пакета. Для получения решения каждой citu-задачи достаточно найти хотя бы один уникальный фрагмент.

Для формального описания заданий будут использоваться следующие обозначения:

z_n — задание-задача с собственным номером n;

K — общее число различных видов заданий, которые могут обрабатываться в данной СВС;

$z_{n(k)}$ — задание с собственным идентификационным номером n, для которого явно указан вид k, $k = \overline{1, K}$. Данное обозначение введено для удобства прочтения формул, описываю-

щих процесс обработки заданий определенного вида.

Для множества $\mathcal{Z}(t)$ — заданий z_n, находящихся в СВС в момент t, обозначим через

$\mathcal{N}(t)$ множество номеров (индексов) заданий z_n из $\mathcal{Z}(t)$;

$N(t)$ — общее число заданий z_n из $\mathcal{Z}(t)$. Таким образом $|\mathcal{Z}(t)| = |\mathcal{N}(t)| = N(t)$;

$\mathcal{Z}_k(t)$ — множество заданий k-го вида;

$\mathcal{N}_k(t)$ — множество номеров (индексов) заданий k-го вида;

$N_k(t)$ — общее число заданий k-го вида.

Тогда

$$\mathcal{Z}(t) = \bigcup_{k=1}^{K} \mathcal{Z}_k(t),\ \mathcal{N}(t) = \bigcup_{k=1}^{K} \mathcal{N}_k(t),\ N(t) = \sum_{k=1}^{K} N_k(t);$$

t_n^0 — календарный момент поступления задания z_n в СВС;

$T_n(t) = (t - t_n^0)$ — длительность промежутка времени, в течение которого задание z_n находится в СВС при условии, что на момент t оно еще не выполнено до конца, т.е. $T_n(t)$ — число единиц времени, которое прошло от поступления z_n в СВС до текущего момента t;

t_n^+ — момент завершения задания z_n и/или его удаления из СВС;

$T_n^+ = (t_n^+ - t_n^0)$ — длительность промежутка времени, в течение которого задание z_n обрабатывалось и находилось в СВС, т.е. T_n^+ — длительность пребывания z_n в системе.

В моделях предполагается, что для каждого задания z_n в момент поступления t_n^0 становится известна нормативная величина $\mathbf{Z}_n$ — общее число фрагментов данных задания z_n, которые необходимо будет обработать, если среди них нет уникального. Заметим, если задача имеет решение, то $\mathbf{Z}_n$ является верхней оценкой для объема *работы*, которую необходимо будет проделать, чтобы найти уникальный фрагмент.

Обозначим через $z^{-}_{n(k)}(t)$ число СЭВ-операций k-го вида, которые были завершены для задания z_n от его поступления в СВС до момента времени t, или в рамках содержательного модельного описания — число неделимых фрагментов, которые уже были просмотрены для задания z_n к моменту t; $z^{+}_{n(k)}(t)$ — число фрагментов, которые остались необработанными для z_n к моменту t. Тогда

$$z^{+}_{n(k)}(t) = \mathbf{Z}_n - z^{-}_{n(k)}(t), \quad n \in \mathcal{N}(t).$$

Как уже указывалось ранее, подзадания ТР-пакета представляют собой различающиеся по объему поднаборы фрагментов данных для заданий $z_n, n \in \mathcal{N}(t)$. Размер подзадания для $z_{n(k)}$ обозначим через $w_{n(k)}(\Delta(t))$. Таким образом, $w_{n(k)}(\Delta(t))$ — число неделимых фрагментов в поднаборе, взятом из массива исходных данных задания $z_{n(k)}$ в момент t для просмотра в течение $\Delta(t)$, или *работа* — число СЭВ-операций k-го вида, которые планируется выполнить в операционном окне $\Delta(t)$ для $z_{n(k)}$. Размеры отобранных поднаборов данных (*работы*) $w_{n(k)}(\Delta(t))$ являются управлениями или управляющими параметрами, которые определяются в ПК-план с учетом текущей диспетчерской политики.

Поскольку в момент t для z_n нельзя взять для обработки данных больше, чем осталось, управления $w_{n(k)}(\Delta(t))$ на интервале $\Delta(t)$ должны удовлетворять ограничениям

$$0 \le w_{n(k)}(\Delta(t)) \le z^{+}_{n(k)}(t), \;\; n \in \mathcal{N}(t), \tag{1}$$

Обозначим через $W_k(\Delta(t))$ — суммарное число фрагментов k-го вида, которые планируется включить в ТР-пакет. Тогда

$$W(\Delta(t)) = \sum_{k=1}^{K} W_k(\Delta(t)) = \sum_{k=1}^{K} \sum_{n \in \mathcal{N}_k(t)} w_{n(k)}(\Delta(t)) =$$

$$= \sum_{n \in \mathcal{N}(t)} w_n(\Delta(t)) \qquad (2)$$

— общий объем ТР-пакета, для просмотра в операционном окне $\Delta(t)$, т.е. объем *работы* на период $\Delta(t)$.

Введем переменные $r_k^m(t)$ — общее число ЕВ-модулей m-го типа, которые начинают обрабатывать подзадания k-го вида в момент времени t. $r_k^m(t)$ также являются управлениями и определяются исходя из текущих установок диспетчерских политик.

Предположим, что ни один из поднаборов $w_{n(k)}(\Delta(t))$ не содержит уникального фрагмента, но все поднаборы планируется выполнить за период $\Delta(t)$. Тогда в момент t для просмотра всех данных для подзаданий k-го вида необходимо выделить $r_k^m(t)$ ЕВ-модулей m-го типа,

$$W_k(\Delta(t)) = \sum_{m=1}^{M} p_k^m(t) r_k^m(t) \Delta(t), \;\; k = \overline{1, K}, \qquad (3)$$

которые за период $\Delta(t)$ должны проделать определенную *работу*: осуществить $W_k(\Delta(t))$ СЭВ-операций k-го вида. Поскольку число ЕВ-модулей, распределяемых для выполнения заданий, не может превышать общего числа работоспособных, то значения $r_k^m(t)$ должны удовлетворять следующим ограничениям:

$$\left.\begin{array}{l} \sum\limits_{k=1}^{K} r_k^m(t) \le R^m(t), \quad m = \overline{1, M}, \\ r_k^m(t) \ge 0, \quad k = \overline{1, K}, \quad m = \overline{1, M}. \end{array}\right\} \qquad (4)$$

2.2. Производительность и эффективность СВС.

Пусть $P_k(t)$ — максимальное число СЭВ-операций k-го вида, — *работа*, которую можно выполнить в единицу времени на СВС, начиная с момента t на всех работоспособных ЕВ-модулях,

$$P_k(t) = \sum_{m=1}^{M} p_k^m(t) \cdot R^m(t), \quad k = \overline{1, K}.$$

При диспетчеризации СВС, где на разнотипном оборудовании с различной скоростью выполняются разнородные СЭВ-операции, возникает проблема эффективного использования имеющихся ресурсов. В реальной ситуации значения $p_k^m(t)$ для разных k и m могут сильно отличаться друг от друга, поскольку представляют собой число СЭВ-операций разных видов, выполняемых в единицу времени на разнотипных ЕВ-модулях. В результате при формировании пакетов подзаданий, т.е. выборе значений вектора $\mathbf{w}^0(t) =< w_1^0(\Delta(t)), \ldots, w_{N(t)}^0(\Delta(t)) >$, в некоторые моменты может возникать недогрузка ЕВ-модулей определенных типов. С формальной точки зрения ограничение (4) например, для l-го типа ЕВ-модулей

$$\sum_{k=1}^{K} r_k^l(t) \le R^l(t), 1 \le l \le M,$$

при конкретных значениях $r^0(t)$ выполняется как строгое неравенство, что ниже будет трактоваться как неэффективное использование ЕВ-модулей соответствующего l-го типа.

Для анализа суммарно возможной производительности СВС в рамках моделей решается следующая оптимизационная задача: при заданных $\mathcal{Z}(t), p(t), \Delta(t), R(t)$ найти

$$\varphi^*(t) = \max_{\varphi, w, r, W} \varphi(t), \tag{5}$$

при ограничениях: на суммарное число фрагментов всех видов:

$$\varphi(t) \le \sum_{n=1}^{N(t)} w_{n(k)}(\Delta(t)), \tag{6}$$

$$0 \le w_{n(k)}(\Delta(t)) \le z^{+}_{n(k)}(t), \quad n \in \mathcal{N}(t), \tag{7}$$

на число фрагментов k-го вида, которые могут быть обработаны за время $\Delta(t)$:

$$\sum_{n \in \mathcal{N}_k(t)} w_{n(k)}(\Delta(t)) = W_k(\Delta(t)) =$$

$$= \sum_{m=1}^{M} p^m_k(t) \cdot r^m_k(t) \cdot \Delta(t), \quad k = \overline{1, K}, \tag{8}$$

на число использованных ЕВ-модулей m-го типа, выполняющих обработку фрагментов всех видов:

$$\left.\begin{array}{l} \sum\limits_{k=1}^{K} r^m_k(t) \le R^m(t), \quad m = \overline{1, M}, \\ r^m_k(t) \ge 0, \quad k = \overline{1, K}, \quad m = \overline{1, M}. \end{array}\right\} \tag{9}$$

Содержательный смысл задачи (5) — (9) состоит в следующем: при заданных ограничениях (6) — (9) для всех z_n за время $\Delta(t)$ требуется просмотреть максимально возможное суммарное число фрагментов независимо от их вида. Таким образом, для исходной группы заданий $\mathcal{Z}(t)$ решение задачи (5) при ограничениях (6) — (9) позволяет максимизировать суммарное число обработанных фрагментов данных на отрезке планирования $\Delta(t)$. В данном случае в ПК-план в каждой контрольной точке t в первую очередь решается задача (5) — (9) и вычисляется значение $\varphi^*(t)$.

Задача (5) — (9) может быть решена стандартными методами линейного программирования [16, 17]. При любых заданных $\Delta(t) > 0, R(t) > 0, z_n^+(t) > 0$ ограничения (6) — (9) непротиворечивы. Если в момент t для всех $m = \overline{1, M}$ значение $R^m(t) = 0$, т.е. работоспособные модули отсутствуют, то $w_n(\Delta(t)) \equiv r_k^m(t) \equiv 0$ для всех n, k, m.

Обозначим через $\rho(t)$ интегральную производительность СВС. Под $\rho(t)$ будем понимать суммарную производительность СВС на интервале $\Delta(t)$, т.е. общее число СЭВ-операций, которые могут быть выполнены на всех действующих ЕВ-модулях, начиная с момента t на отрезке планирования $\Delta(t)$:

$$\rho(t) = \sum_{m=1}^{M} \sum_{k=1}^{K} p_k^m(t) r_k^m(t) \Delta(t).$$

В качестве максимальной интегральной производительности СВС $P^*(t)$ на интервале $\Delta(t)$ выберем скалярное решение следующей задачи линейного программирования: найти

$$P^*(t) = \max_{\rho, r} \rho(t),$$

при ограничениях на производительность СВС:

$$\left.\begin{array}{l} \rho(t) = \sum\limits_{m=1}^{M} \sum\limits_{k=1}^{K} p_k^m(t) r_k^m(t) \Delta(t), \\ \sum\limits_{k=1}^{K} r_k^m(t) \le R^m(t), \quad m = \overline{1, M}; \\ r_k^m(t) \ge 0, \quad k = \overline{1, K}, \quad m = \overline{1, M}; \end{array}\right\} \tag{10}$$

Значение $P^*(t)$ показывает, какое максимальное число СЭВ-операций суммарно для всех видов заданий может быть выполнено СВС в операционное окно $\Delta(t)$, начиная с момента

t. Пусть на интервале $\Delta(t)$ предполагается произвести $\varphi^*(t)$ СЭВ-операций всех видов. Тогда отношение

$$\gamma(t) = \frac{\varphi^*(t)}{P^*(t)}$$

назовем показателем плановой эффективности использования СВС на интервале $\Delta(t)$ для данного набора заданий $\mathcal{Z}(t)$. Максимальное значение $\varphi^*(t)$, и соответственно, $\gamma(t)$, существенно зависит от состава и объема заданий, находящихся на обработке в момент времени t. Сравнивая множества ограничений (6)—(9) и (10), можно утверждать , что для любого набора заданий $\mathcal{Z}(t)$ выполняются неравенства

$$P^*(t) \geq \varphi^*(t), 0 \leq \gamma(t) \leq 1.$$

Кроме того, при диспетчеризации следует учитывать другие ограничения и требования пользователей, которым должны подчиняться управления $r_k^m(t), w_{n(k)}(\Delta(t))$.

2.3. Директивный срок окончания.

Для управления в реальном времени гетерогенными СВС необходимы и разрабатываются интерактивные сценарии диалога администратор — пользователь в декларативной, частично вербальной, частично технологической, постановке: задание — требования — возможности — сроки. В комплексе CORSAR предполагается, что при поступлении заявки в СВС определяется возможное время ее завершения с учетом объективных показателей загруженности и состояния ресурсов, а также имеющихся требований. Если предварительная оценка времени завершения устраивает пользователя, то задание принимается для обработки, и ему присваивается директивный срок окончания, совпадающий с прогнозируемым. Таким образом, дирек-

тивный срок окончания (ДСО) — это установленный администратором и согласованный с пользователем момент календарного времени, до наступления которого задача должна быть решена. Дальнейшее планирование осуществляется исходя из предположения, что реализуется наихудший случай: ни одна из задач не имеет решения. В условиях объективной неопределенности ДСО используется как ограничение при формировании диспетчерских правил (политики) совместного выполнения всех заданий, находящихся в СВС [18].

Заметим, что понятие ДСО, введенное выше, несколько разнится от классического dead-line [19], поскольку устанавливается администратором, а не пользователем системы, хотя и по договоренности с последним. Однако принятое допущение больше соответствует реальной ситуации. Действительно, в этом случае:

— заявка, которая не может быть выполнена по объективным причинам, будет сразу отозвана, что позволит решить другие задачи;

— у администратора появляется возможность более гибко подходить к формированию пакета текущих *работ*, и минимизировать возможные производственные потери, т.е. повышается эффективность использования СВС.

В рамках моделей в ПК-план для каждого вновь поступившего citu-задания вычисляется гарантированная оценка времени завершения. Фактически решается задача *быстродействия*: определяется время выполнения всех заданий в наихудшем случае. Обозначим через d_n ДСО для z_n.

Пусть в момент t в СВС обрабатывается пакет заданий $\mathcal{Z}(t)$ и поступает новая задача $\hat{z}$, например, $\hat{k}$-го вида. В ПК-план последней присваивается порядковый номер $\hat{n}(\hat{k})$. Кроме того, определяется величина $\mathbf{Z}_{\hat{n}(\hat{k})}$ — общее число фрагментов данных, которые необходимо будет просмотреть, при условии, что

среди нет уникального.

Для каждого $z_n, n \in \mathcal{N}(t)$ и каждого m определим переменную $\nabla_n^m(t)$ — число единиц времени (единичных временных интервалов), в течение которых задание z_n планируется выполнять на всех работоспособных ЕВ-модулях m-го типа, $m = \overline{1, M}$, после момента t. При определении ДСО для вновь поступивших заданий рассматривается наихудший случай: ни одна из вновь полученных и ни одна из находящихся в обработке задач не имеют решения. Другими словами, для завершения всех имеющихся заданий необходимо будет просмотреть все оставшиеся фрагменты данных. Следовательно для заданий, обработка которых уже начата, будет необходимо просмотреть

$$\mathbf{Z}_{n(k)} - z^-_{n(k)}(t) = \sum_{m=1}^{M} p_k^m(t) R^m(t) \nabla^m_{n(k)}(t), n \in \mathcal{N}_k(t), n = \overline{1, K},$$

фрагментов, а для вновь поступившего задания $\hat{z}$ —

$$\mathbf{Z}_{\hat{n}(\hat{k})} = \sum_{m=1}^{M} p_{\hat{k}}^m(t) R^m(t) \nabla^m_{\hat{n}(\hat{k})}(t),$$

$$\nabla^m_{n(k)}(t) \geq 0, m = \overline{1, M}, n \in \mathcal{N}(t),$$

$$\nabla^m_{\hat{n}(\hat{k})}(t) \geq 0, m = \overline{1, M}.$$

Введем переменную $\nabla^m(t)$:

$$\nabla^m(t) = \sum_{k=1}^{K} \sum_{n \in \mathcal{N}_k(t)} \nabla^m_{n(k)}(t) + \nabla^m_{\hat{n}(\hat{k})}(t) =$$

$$= \sum_{n \in \mathcal{N}(t)} \nabla^m_{n(k)}(t) + \nabla^m_{\hat{n}(\hat{k})}(t), m = \overline{1, M}.$$

Значение $\nabla^m(t)$ показывает, сколько единиц времени после момента t потребуется ЕВ-модулям m-го типа для выполнения всех *работ*, имеющихся в СВС, при условии, что необходимо просмотреть все данные всех заданий из $\mathcal{Z}(t)$, если среди них нет уникального.

В ПК-план в качестве оценки ДСО для вновь поступившего задания $z_{\hat{n}(\hat{k})}$ используется решение следующей задачи *быстродействия*:

в момент t при заданных $\mathbf{Z}_n, \mathbf{Z}_{\hat{n}(\hat{k})}, z^-_{n(k)}(t), \mathcal{N}(t), R^m(t), d_n$, найти

$$\delta^* = \min_{\delta,\nabla} \delta$$

при условиях:

что ДСО для вновь поступившего задания будет не меньше, чем у находящихся в обработке:

$$\delta \geq d_j, j \in \mathcal{N}(t),$$

что ДСО не меньше гарантированного времени завершения всех заданий на всех ЕВ-модулях:

$$\delta \geq t + \sum_{n \in \mathcal{N}(t)} \nabla^m_{n(k)}(t) + \nabla^m_{\hat{n}(\hat{k})}(t), m = \overline{1, M},$$

что для выполнения всех заданий необходимо будет просмотреть все имеющиеся в момент t данные

$$\mathbf{Z}_{n(k)} - z^-_{n(k)}(t) = \sum_{m=1}^{M} p^m_k(t) R^m(t) \nabla^m_{n(k)}(t), n \in \mathcal{N}_k(t), n = \overline{1, K},$$

$$\nabla^m_{n(k)}(t) \geq 0, m = \overline{1, M}, n \in \mathcal{N}(t), k = \overline{1, K},$$

$$\mathbf{Z}_{\hat{n}(\hat{k})} = \sum_{m=1}^{M} p^m_{\hat{k}}(t) R^m(t) \nabla^m_{\hat{n}(\hat{k})}(t),$$

$$\nabla^m_{\hat{n}(\hat{k})}(t) \geq 0, m = \overline{1, M}.$$

Полученное значение δ^* является гарантированной оценкой ДСО для вновь поступившей задачи. Администратор сообщает его пользователю, который предоставил данную задачу. Если последнего не устраивают прогнозируемые сроки завершения, то он отказывается от обработки и отзывает свою заявку. В противном случае — задание $z_{\hat{n}(\hat{k})}$ помещается в общий пакет $\mathcal{Z}(t)$, с условием, что его обработка должна быть завершена до момента $d_{\hat{n}(\hat{k})} = \delta^*$.

Рассмотрим проблему диспетчеризации citu-заданий с фиксированными ДСО. В рамках моделей ПК-план предполагается, что в случае превышения ДСО произведенные вычислительные затраты записываются в производственные потери, которые администратору предписывается свести к минимуму. На первый взгляд, описанный выше способ назначения ДСО для всех заданий, принятых на обработку, гарантирует их окончание в срок. Однако в реальности в СВС могут происходить сбои, изменение приоритетов в обслуживании заявок, и др., что неизбежно приведет к нарушению ДСО. В ПК-план в момент t анализируются достаточные условия завершения заданий в срок и вычисляются гарантированные оценки возможных превышений ДСО в наихудшем случае.

В соответствии с принятыми ранее обозначениями d_j — ДСО задачи z_j. Каждому $d_j,\ j \in \mathcal{N}(t)$ ставится в соответствие множество (список) номеров $\mathcal{N}(t, d_j)$ всех заданий z_i из $\mathcal{Z}(t)$ таких, что каждое z_i требуется завершить не позднее d_j, т. е.

$$\mathcal{N}(t, d_j) = \{i \mid d_i \leq d_j,\ i \in \mathcal{N}(t)\}, j \in \mathcal{N}(t).$$

Для получения гарантированных оценок рассмотрим наихудший случай: пусть ни одна из задач $z_n,\ n \in \mathcal{N}(t)$, не имеет решения. Для фиксированного момента времени $(t + \Delta(t))$ введем дополнительный параметр управления $\nabla^m_{n(k)}(t + \Delta(t))$ —

число единиц времени, в течение которых задание z_n может обрабатываться на всех работоспособных ЕВ-модулях m-го типа, $m = \overline{1, M}$, начиная с $(t + \Delta(t))$. Таким образом $\nabla^m_{n(k)}(t + \Delta(t))$ измеряется в единицах времени, которые используются в модельном описании.

В рассматриваемом наихудшем случае задача z_n не имеет решения, поэтому для завершения задания необходимо выполнить следующую *работу*: перебрать все данные

$$z^+_{n(k)}(t+\Delta(t)) = \mathbf{Z}_{n(k)} - z^-_{n(k)}(t) - w_{n(k)}(\Delta(t)), \qquad (11)$$

оставшиеся необработанными для z_n к моменту $(t+\Delta(t))$. Здесь управление $w_{n(k)}(\Delta(t))$ — число фрагментов данных для z_n, которые в момент t планируется просмотреть за время $\Delta(t)$.

Таким образом, для оставшихся фрагментов в момент $(t + \Delta(t))$ и $n \in \mathcal{N}_k(t), k = \overline{1, K}$, должно выполняться

$$z^+_{n(k)}(t+\Delta(t)) = \sum_{m=1}^{M} p^m_k(t) R^m(t) \nabla^m_{n(k)}(t+\Delta(t)), \qquad (12)$$

и условие неотрицательности

$$\nabla^m_{n(k)}(t + \Delta(t)) \geq 0, m = \overline{1, M}, n \in \mathcal{N}(t).$$

В момент $(t + \Delta(t))$ рассмотрим переменные

$$\nabla^m(t + \Delta(t), d_j) = \sum_{k=1}^{K} \sum_{n \in \mathcal{N}(t,d_j) \bigcap \mathcal{N}_k(t)} \nabla^m_{n(k)}(t + \Delta(t)) =$$

$$= \sum_{n \in \mathcal{N}(t,d_j)} \nabla^m_{n(k)}(t+\Delta(t)), \;\; m = \overline{1, M}, \;\; j \in \mathcal{N}(t). \qquad (13)$$

Значение $\nabla^m(t + \Delta(t), d_j)$ показывает, сколько единиц времени после момента $(t + \Delta(t))$ потребуется ЕВ-модулям m-го типа

для выполнения всех заданий $z_n,\ n \in \mathcal{N}(t, d_j)$, которые должны быть завершены до наступления d_j. В случае, когда ни одна из задач не имеет решения, величина $\nabla^m(t + \Delta(t), d_j)$ — точная верхняя оценка временных затрат на обработку всех $z_n,\ n \in \mathcal{N}(t, d_j)$, с помощью работоспособных ЕВ-модулей m-го типа, $m = \overline{1, M}$. В момент t достаточные условия соблюдения всех ДСО в наихудшем случае можно записать в виде

$$t + \Delta(t) + \nabla^m(t + \Delta(t), d_j) \le d_j,\ j \in \mathcal{N}(t),\ m = \overline{1, M}.$$

Для получения гарантированных оценок относительных величин возможного превышения ДСО для z_n, введем переменные:

$$\omega^m(t + \Delta(t), d_j) =$$

$$= \frac{t + \Delta(t) + \nabla^m(t + \Delta(t), d_j) - t_j^0}{d_j - t_j^0},\ j \in \mathcal{N}(t),\ m = \overline{1, M}, (14)$$

где t_j^0 — время поступления z_j в СВС. Конкретное значение $\omega^m(t + \Delta(t), d_j)$ характеризует возможность соблюдения ДСО следующим образом:

1) если существуют управления $w_n(\Delta(t)), \nabla^m_{n(k)}(t + \Delta(t))$ и значение $\nabla^m(t + \Delta(t), d_j)$, удовлетворяющие (11)-(14), при которых $\omega^m(t + \Delta(t), d_j) \le 1$ для всех $j \in \mathcal{N}(t), m = \overline{1, M}$, то все задания можно гарантированно завершить до наступления их ДСО даже в наихудшем случае;

2) если же при любых управлениях $w_n(\Delta(t)), \nabla^m_{n(k)}(t+\Delta(t))$ и значениях $\nabla^m(t+\Delta(t), d_j)$ в (11)-(14), хотя бы для одного сочетания m и j величина $\omega^m(t + \Delta(t), d_j) > 1$, то может возникнуть ситуация, при которой хотя бы одно задание в наихудшем случае завершится после предписанного срока.

На самом деле максимальная величина $\omega^m(t + \Delta(t), d_j)$ является своего рода индикатором, который указывает, располагает ли СВС на момент t необходимой мощностью для того,

чтобы выполнить необходимую работу — обработать все имеющиеся разнородные задания до наступления соответствующих ДСО. Для проверки достаточных условий выполнения ДСО в момент t и получения гарантированных оценок максимально возможного их превышения в наихудшем случае введем параметр управления $\omega(t)$:

$$\omega(t) \le 1 - \omega^m(t + \Delta(t), d_j), \quad m = \overline{1, M}, j \in \mathcal{N}(t).$$

Величина $\omega(t)$ характеризует максимальное относительное значение возможного превышения ДСО, т.е. $\omega(t)$ в момент t оценивает каждое управление $w_{n(k)}(\Delta(t))$.

Рассмотрим возможный способ формирования ТР-пакета, при котором в процессе выполнения заданий требуется добиться высокой эффективности использования СВС и минимизировать производственные потери, в данном случае за счет соблюдения ДСО. В рамках моделей ПК-план для анализа эффективности и получения гарантированных оценок превышения ДСО в момент t решается следующая задача оптимизации:

при заданных $\mathbf{Z}_n, z^-_{n(k)}(t), N(t), R^m(t), d_j, \mathcal{N}(t), P^*(\Delta(t))$,

найти

$$\Phi^* = \max_{\gamma,\omega,\nabla,w,r,W} (c_1\gamma(t) + c_2\omega(t) + c_3\omega^\Sigma(t)) \tag{15}$$

при ограничениях:

на эффективность использования разнотипных ЕВ-модулей в СВС:

$$0 \le \gamma(t)P^*(\Delta(t)) = \sum_{k=1}^{K}\sum_{m=1}^{M} p_k^m(t) r_k^m(t)\Delta(t); \tag{16}$$

на суммарную величину отклонения от заданных ДСО:

$$\left.\begin{array}{l} \omega^\Sigma(t) \le \sum\limits_{m=1}^{M}\sum\limits_{j\in\mathcal{N}(t)} (1 - \omega^m(t + \Delta(t), d_j)), \\ \omega^\Sigma(t) \le 0; \end{array}\right\} \tag{17}$$

на максимальную величину отклонения от заданных ДСО

$$\left.\begin{array}{c}\omega(t) \le 1 - \omega^m(t + \Delta(t), d_j), \\ j \in \mathcal{N}(t),\ m = \overline{1, M}; \\ \omega(t) \le 0;\end{array}\right\} \tag{18}$$

на балансовые соотношения выполнения заданий после момента $t + \Delta(t)$

$$\left.\begin{array}{c}\mathbf{Z}_n - z_n^-(t) - w_n(\Delta(t)) = \sum\limits_{m=1}^{M} p_k^m(t) R^m(t) \nabla_{n(k)}^m(t + \Delta(t)), \\ n \in \mathcal{N}_k(t),\quad k = \overline{1, K}, \\ \nabla_{n(k)}^m(t + \Delta(t)) \ge 0,\ n \in \mathcal{N}(t),\ m = \overline{1, M};\end{array}\right\} \tag{19}$$

$$\left.\begin{array}{c}\nabla^m(t + \Delta(t), d_j) = \sum\limits_{n \in \mathcal{N}(t, d_j)} \nabla_{n(k)}^m(t + \Delta(t)), \\ m = \overline{1, M}, j \in \mathcal{N}(t);\end{array}\right\} \tag{20}$$

$$\left.\begin{array}{c}\omega^m(t + \Delta(t), d_j) = \dfrac{t + \Delta(t) + \nabla^m(t + \Delta(t), d_j) - t_j^0}{d_j - t_j^0}, \\ j \in \mathcal{N}(t),\ m = \overline{1, M};\end{array}\right\} \tag{21}$$

на управления в операционном окне $\Delta(t)$:

а) на суммарное число СЭВ-операций k-го вида, планируемых для ТР-пакета в момент t

$$\left.\begin{array}{c}\sum\limits_{n \in \mathcal{N}_k(t)} w_n(\Delta(t)) = W_k(\Delta(t)),\ k = \overline{1, K}, \\ 0 \le w_n(\Delta(t)) \le \mathbf{Z}_n - z_n^-(t),\ n \in \mathcal{N}(t);\end{array}\right\} \tag{22}$$

б) на число ЕВ-модулей, выделяемых для обработки фрагментов для подзаданий k-го вида в момент t

$$\left.\begin{array}{l}W_k(\Delta(t)) \le \sum\limits_{m=1}^{M} p_k^m(t) r_k^m(t) \Delta(t),\ k = \overline{1, K}; \\ r_k^m(t) \ge 0,\quad k = \overline{1, K},\quad m = \overline{1, M}; \\ \sum\limits_{k=1}^{K} r_k^m(t) \le R^m(t),\ m = \overline{1, M}.\end{array}\right\} \tag{23}$$

Условия (16) — (23) совместны, поэтому задача (15) может быть решена стандартными методами линейного программирования [16]. Поскольку переменные $\gamma(t)$, $\omega(t)$ и $\omega^{\Sigma}(t)$ являются относительными, безразмерными и изменяются в ограниченном диапазоне, то с помощью выбора значений c_1, c_2, c_3 можно учитывать текущие приоритеты при выполнении заданий.

Предположим, в момент времени t задача (15) решается при некоторых значениях $c_1 = \hat{c}_1, c_2 = \hat{c}_2, c_3 = \hat{c}_3$, таких, что $\hat{c}_1 \ll \hat{c}_2 = \hat{c}_3$. Например, $\hat{c}_2 = \hat{c}_3 = 10^6, \hat{c}_1 = 1$. Пусть $\hat{\omega}^0(t)$ ее оптимальное решение. Тогда:

если $\hat{\omega}^0(t) = 0$, то существует гарантированная стратегия управления выполнением заданий из $\mathcal{Z}(t)$, при которой все задания могут быть завершены до наступления своих ДСО, даже если ни одна задача $z_n,\ n \in \mathcal{N}(t)$, не имеет решения (наихудший случай);

если $\hat{\omega}^0(t) < 0$, то найденный вектор управлений $\hat{\mathbf{w}}^0(t)$, т.е. состав ТР-пакет на период $\Delta(t)$, является наилучшим решением, минимизирующим возможное относительное превышение некоторых ДСО на величину не более $|\hat{\omega}^0(t)|$. В последнем случае можно взять вновь полученный вектор управлений $\hat{\mathbf{w}}^0(t)$, сформировать соответствующий ТР-пакет и приступить к его выполнению. Действительно, для некоторых $z_n, n \in \mathcal{N}(t)$ существует опасность не уложиться в ДСО, и поэтому все ресурсы должны быть направлены на выполнение таких заданий. Список задач и число фрагментов, которые необходимо срочно обработать, содержатся в векторе $\hat{\mathbf{w}}^0(t)$, т.е. состав аварийного ТР-пакет уже готов, поскольку получен в ходе решения (15) при $c_1 = 1, c_2 = c_3 = 10^6$.

Далее можно взять найденные значения $\hat{\omega}^0(t), \hat{\omega}^{\Sigma}(t)$, зафиксировать их значения, а именно, положить в ограничениях (17) $\omega^{\Sigma}(t) = \hat{\omega}^{\Sigma}(t)$, а в (18) — $\omega(t) = \hat{\omega}^0(t)$; исключить их из функционала, приравняв c_2, c_3 нулю, т.е. $c_2 = c_3 = 0$. После

этого найдем максимум (15) при $c_1 = 10^6$, и в результате получим максимальную эффективность СВС при жестко заданных ограничениях на выполнение ДСО.

2.4. Рабочее место администратора.

В комплексе CORSAR отображение текущей информации и предварительное планирование осуществляется программным компонентом рабочее место администратора (ПК-рабочее-место) В процессе обработки потока заявок производится его декомпозиция по видам задач, агрегирование и определение объема требований для выполнения соответствующих *работ.* А именно, в момент времени t каждому $j \in \mathcal{N}_k(t)$ ставится в соответствие множество (список) номеров $\mathcal{N}_k(t, d_j)$ всех заданий z_i из $\mathcal{Z}_k(t)$ таких, что $d_i \le d_j$, т.е.

$$\mathcal{N}_k(t, d_j) = \{i \mid d_i \le d_j, i \in \mathcal{N}_k(t)\}, j \in \mathcal{N}_k(t), k = \overline{1, K}.$$

Другими словами, у каждого задания $z_{n(k)}, n \in \mathcal{N}_k(t, d_j)$ предписанный ДСО наступает раньше, чем d_j.

В ПК-рабочее-место в момент t при заданном $\Delta(t)$ для всех $n \in \mathcal{N}(t)$ вычисляются *требования на работу*

$$W_k(\Delta(t), d_j) = \frac{\Delta(t)}{d_j - t} \sum_{i \in \mathcal{N}_k(t, d_j)} z_i^+(t), j \in \mathcal{N}_k(t), k = \overline{1, K}.$$

Величина $W_k(\Delta(t), d_j)$ определяет необходимое число СЭВ-операций k-вида, которое требуется выполнить на интервале $\Delta(t)$, чтобы группа задач с номерами $i \in \mathcal{N}_k(t, d_j)$, была завершена до момента d_j, даже если ни одна из них не имеет решения, т.е. в наихудшем случае.

Агрегированные запросы-требования на выполнение *работ* k-го вида на интервале $\Delta(t)$ определяются максимальными

значениями

$$W_k^0(\Delta(t), d_{j^0}^0) = \max_{j \in \mathcal{N}_k(t)} \{W_k(\Delta(t), d_j)\}, k = \overline{1, K},$$

где j^0 — номер из списка задач $\mathcal{N}_k(t)$, требующих максимальной *работы* на $\Delta(t)$; $d_{j^0}^0$ — ДСО задания z_{j^0}.

Величина $W_k^0(\Delta(t), d_{j^0}^0)$ показывает, какое максимальное число СЭВ-операций k-го вида необходимо выполнить на интервале $\Delta(t)$, чтобы в наихудшем случае гарантированно завершить все задания $z_n, n \in \mathcal{N}_k(t)$ в предписанные сроки. Полученные значения $W_k^0(\Delta(t), d_{j^0}^0)$ отображаются на мониторе в виде изображенных на рис. 2 гистограмм.

Кроме того, с помощью оптимизационной модели из библиотеки ПК-план производится предварительная агрегированная оценка возможности удовлетворить *требования* $W_k^0(\Delta(t), d_{j^0}^0)$, т.е. выполнимость запрашиваемой *работы*. Для этого при заданных $W_k^0(\Delta(t), d_{j^0}^0), R^m(t), p^m(t), \Delta(t)$ решается задача:

$$\alpha^* = \max_{\alpha, W, r} \alpha,$$

$$\alpha W_k^0(\Delta(t), d_{j^0}^0) \le W_k, \ k = \overline{1, K},$$

$$W_k \le \sum_{m=1}^{M} p_k^m(t) r_k^m(t) \Delta(t), \ k = \overline{1, K},$$

$$\sum_{k=1}^{K} r_k^m(t) \le R^m(t), \ m = \overline{1, M},$$

$$0 \le r_k^m(t), \ k = \overline{1, K}, \ m = \overline{1, M}.$$

Оптимальное решение данной задачи обозначим через $< \alpha^*, W_1^*, W_2^*, ..., W_K^* >$.

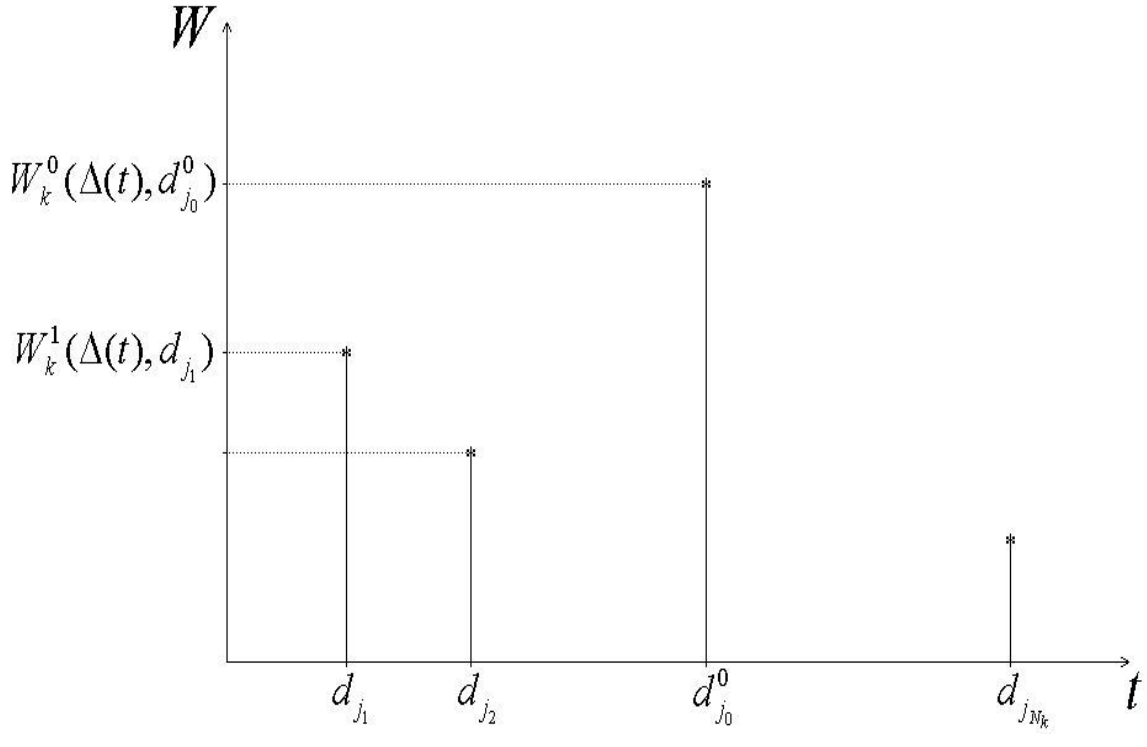

Рис. 2

Если $\alpha^* \geq 1$, то требуемая *работа* может быть выполнена в полном объеме, и существует $\hat{W}_k \leq W_k^*$, такое, что $\hat{W}_k = W_k^0(\Delta(t), d_{j^0}^0)$.

Если $\alpha^* < 1$, то для некоторых $\hat{k}$ имеет место неравенство $W_k^* < W_k^0(\Delta(t), d_{j^0}^0)$, и выполнить запрашиваемую работу за $\Delta(t)$ невозможно. Полученные предварительные результаты также отображаются на экране компьютера в графическом виде, аналогичном рис.2.

2.5. Показатель относительной задержки выполнения задания.

С содержательной точки зрения, в рассматриваемом классе citu-задачи в определенном смысле персонифицированы: все они *равнозначны*, а пользователи, предоставляющие задачи для обработки, *равноправны* в рамках правил распределения вычислительных ресурсов и очередности обработки, поэтому стратегии управления СВС не должны быть дискриминирующими по отношению к той или иной группе заявок. Здесь и далее под *недискриминирующим* правилом выполнения равнозначных заданий равноправных пользователей понимается стратегия, при которой ни один из партнеров не может улучшить временные характеристики выполнения своей заявки, не ухудшив соответствующие показатели своих коллег. Предполагается, что пользователи не являются антагонистами и стремятся к достижению некоторой общей корпоративной цели.

Монопольным режимом решения задачи z_n будем называть такой способ организации работы СВС, при котором на всех работоспособных ЕВ-модулях одновременно просматриваются фрагменты данных только одного задания z_n. Суммарное время, затраченное СВС на обработку всего исходного массива данных для z_n в монопольном режиме, назовем *абсолютным*

временем выполнения задания и обозначим его через τ_n.

Пусть верно одно из предположений: в поднаборе $w_n(\Delta(t))$ находится уникальный фрагмент для z_n или обработка поднабора $w_n(\Delta(t))$ завершает выполнение z_n. Тогда значение

$$z^-_{n(k)}(t+\Delta(t)) = z^-_{n(k)}(t) + w_{n(k)}(\Delta(t)), n \in \mathcal{N}_k(t), k = \overline{1,K},$$

является оценкой сверху для общего числа фрагментов, которые в действительности необходимо просмотреть для завершения задания z_n или для решения соответствующей ему задачи.

В п. 2.2 через $P_k(t)$ было обозначено максимальное число СЭВ-операций k-го вида, которые может выполнить СВС в единицу времени на всех работоспособных ЕВ-модулях, начиная с момента t. Другими словами, $P_k(t)$ — производительность СВС в монопольном режиме для любого задания z_n k-го вида. Следовательно, величина

$$\overline{\tau}_n(\Delta(t)) = \frac{z^-_n(t) + w_n(\Delta(t))}{P_k(t)}, n \in \mathcal{N}_k(t),\ k = \overline{1,K},$$

является оценкой сверху для τ_n на момент времени $t+\Delta(t)$, и

$$\overline{\tau}_n(\Delta(t)) \geq \tau_n, n \in \mathcal{N}(t).$$

В п. 2.1 время пребывания z_n в СВС на момент времени t было обозначено через $T_n(t)$, при этом

$$T_n(t+\Delta(t)) = T_n(t) + \Delta(t) = t + \Delta(t) - t^0_n,$$

напомним, что здесь t^0_n — момент поступления задания z_n в СВС. Если выполнение задания z_n будет завершено к моменту $t^+_n \leq t+\Delta(t)$, то длительность пребывания z_n в СВС составит $T^+_n = t^+_n - t^0_n$, и

$$T^+_n \leq T_n(t+\Delta(t)) = t + \Delta(t) - t^0_n.$$

Введем переменную

$$\chi_n(t+\Delta(t)) = \frac{\overline{\tau}_n(\Delta(t))}{T_n(t+\Delta(t))}, n \in N(t), \quad (24)$$

которую будем называть *показателем относительной задержки выполнения задания* z_n, если оно завершено на отрезке $\Delta(t)$. Величина $\chi_n(t+\Delta(t))$ является гарантированной оценкой той доли, которую составляет абсолютное время обработки z_n в монопольном режиме от фактического времени пребывания в СВС z_n при условии, что оно будет завершено к моменту $t+\Delta(t)$. При этом

$$\chi_n(t+\Delta(t)) \geq \frac{\tau_n}{T_n(t+\Delta(t))}, n \in N(t).$$

Обратную величину $æ_n(t+\Delta(t)) = (\chi_n(t+\Delta(t)))^{-1}$ назовем коэффициентом относительной задержки. Последний показывает, во сколько раз время фактического пребывания в системе оказалось больше абсолютного времени обработки в монопольном режиме.

В [20-23] анализировались различные диспетчерские стратегии обработки пакетов однородных citu-заданий с одинаковыми $\mathbf{Z}_n$. Исследования показали, что стратегии должны учитывать собственные характеристики каждой citu-задачи, в частности, время пребывания и число просмотренных фрагментов из соответствующих массивов. Формирование ТР-пакета на базе оценок абсолютного времени обработки позволяет реализовать правило SWAP, что на практике приводит к тому, что задачи с небольшим абсолютным временем обработки покидают СВС в первую очередь, а задания, не имеющие решения — завершаются последними. В комплексе CORSAR для соблюдения правила SWAP в каждой контрольной точке оптимальные значения управлений вычисляются с учетом показателей

относительной задержки. В ПК-план величины выполняемых подзаданий $w_n(\Delta(t))$ выбираются так, чтобы минимальная величина $\chi_n(t+\Delta(t))$ оказалась максимальной из возможных для всех заданий z_n. Формально ведется поиск

$$\theta^*(t) = \max_{w_n(\Delta(t))} \min_{1\le n\le N(t)} \chi_n(t+\Delta(t)). \qquad (25)$$

Выпишем значение $\chi_n(t+\Delta(t))$ для случая выполнения разнородных заданий в гетерогенной СВС. Если поднабор $w_n(\Delta(t))$ содержит уникальный фрагмент, или массив исходных данных z_n после проверки $w_n(\Delta(t))$ окажется исчерпан, то величина

$$\overline{\tau}_n((\Delta(t))) = \frac{z_n^-(t) + w_n(\Delta(t))}{P_k(t)}, n \in N_k(t), k = \overline{1,K},$$

является оценкой сверху для абсолютного времени выполнения задания z_n в монопольном режиме. С учетом предыдущего равенства определение (24) — показатель относительной задержки выполнения любого из разнородных заданий на момент времени $(t+\Delta(t))$ в гетерогенной СВС — можно записать в виде

$$\chi_n(t+\Delta(t)) = \frac{z_n^-(t) + w_n(\Delta(t))}{P_k(t)} \cdot \frac{1}{T_n(t)+\Delta(t)},$$

$$n \in N_k(t), k = \overline{1,K}.$$

При этом, фактически, завершенное задание в момент $t_n^+ \le t + \Delta(t)$ покинет СВС, и, следовательно, длительность пребывания z_n в СВС составит $T_n^+ = t_n^+ - t_n^0 \le T_n(t) + \Delta(t)$.

В сделанных предположениях поиск решения (25) можно свести к следующей задаче линейного программирования:

найти

$$\theta^*(t) = \max_{\theta,w,r,W} \theta(t) \qquad (26)$$

при ограничениях:

на показатель относительной задержки выполнения разнородных заданий:

$$\theta(t+\Delta(t)) \le \chi_n(t+\Delta(t)), n \in \mathcal{N}(t), \tag{27}$$

$$\chi_n(t+\Delta(t)) = \frac{z_n^-(t) + w_n(\Delta(t))}{P_k(t)} \cdot \frac{1}{T_n(t)+\Delta(t)}, \tag{28}$$

$$n \in \mathcal{N}_k(t), k = \overline{1,K};$$

на число фрагментов в поднаборе $w_n(\Delta(t))$:

$$\left.\begin{array}{l} \sum\limits_{n\in\mathcal{N}_k(t)} w_n(\Delta(t)) = W_k(\Delta(t)),\ k = \overline{1,K}, \\ 0 \le w_n(\Delta(t)) \le \mathbf{Z}_n - z_n^-(t),\ n \in \mathcal{N}(t); \end{array}\right\} \tag{29}$$

на число фрагментов k-го вида, которые могут быть обработаны за время $\Delta(t)$:

$$W_k(\Delta(t)) \le \sum_{m=1}^{M} p_k^m(t) r_k^m(t) \Delta(t),\ k = \overline{1,K}; \tag{30}$$

на число использованных ЕВ-модулей m-го типа, выполняющих обработку фрагментов всех видов, начиная с момента t:

$$\left.\begin{array}{l} \sum\limits_{k=1}^{K} r_k^m(t) \le R^m(t),\ m = \overline{1,M}; \\ 0 \le r_k^m(t), k = \overline{1,K}, m = \overline{1,M}; \end{array}\right\} \tag{31}$$

Обозначим оптимальное решение (26)-(31) через

$$< \theta^*(t), w_1^*(\Delta(t)), w_2^*(\Delta(t)), \ldots, w_{N(t)}^*(\Delta(t)) > .$$

Вектор значений

$$\mathbf{w}^*(t) =< w_1^*(\Delta(t)), w_2^*(\Delta(t)), \ldots, w_{N(t)}^*(\Delta(t)) >$$

будем называть оптимальным по критерию (25) распределением выполняемых подзаданий $w_n(\Delta(t))$. При этом для всех z_n, завершенных на интервале $\Delta(t)$, показатель относительной задержки выполнения — не меньше $\theta^*(t)$.

При указанном распределении для каждой citu-задачи выбирается такая доля фрагментов данных для просмотра, которая соответствует реальному, хотя и неизвестному в текущий момент абсолютному времени решения с учетом фактического пребывания в СВС. Любая попытка увеличить долю $w_n^*(\Delta(t))$ для любой заявки z_n приведет к уменьшению значения (25) для остальных пользователей. С формальной точки зрения предлагаемый способ выбора размеров подзаданий можно рассматривать как обобщение известного метода Round-Robin [24] разделения операционного времени центрального процессора.

2.6. Однородные задания.

Рассмотрим важную частную проблему: управление выполнением заданий одного вида на однотипном оборудовании. Анализ однородного случая позволяет получить соотношения, использование которых открывает широкие возможности для декомпозиции исходной проблемы и построения содержательных эвристик, реализованных в комплексе CORSAR.

Обозначим через $W^+(\Delta(t))$ — максимальное суммарное число СЭВ-операций, которое планируется выполнить на интервале планирования $\Delta(t)$, при этом

$$W^+(\Delta(t)) = R(t) \cdot p(t) \cdot \Delta(t).$$

Перепишем условия задачи (26)-(31) для получения оценок в однородном случае, положив, $K = 1, M = 1$, и, следовательно, $N(t) = N_k(t)$. Тогда возникает следующая задача линейного программирования:

найти

$$\theta^*(t) = \max_{\theta, w} \theta(t), \tag{32}$$

при ограничениях:

на $\theta(t)$ — минимальную величину показателя относительной задержки:

$$\left.\begin{array}{l} \theta(t) \le \chi_n(t + \Delta(t)), \\ \chi_n(t + \Delta(t)) = \dfrac{z_n^-(t) + w_n(\Delta(t))}{P(t) \cdot T_n(t + \Delta(t))}, n \in \mathcal{N}(t); \end{array}\right\} \tag{33}$$

на суммарное число фрагментов данных:

$$\left.\begin{array}{r} \sum\limits_{n \in \mathcal{N}(t)} w_n(\Delta(t)) \le W^+(\Delta(t)), \\ w_n(\Delta(t)) \ge 0, n \in \mathcal{N}(t). \end{array}\right\} \tag{34}$$

Условия (33) - (34) совместны, поэтому задача (32) может быть решена стандартными методами линейного программирования [16]. Оптимальное решение (32)-(34) обозначим через $< \theta^*(t), w_1^*(\Delta(t)), \ldots, w_{N(t)}^*(\Delta(t)) >$.

Пусть $\mathcal{N}^*(t)$ — множество номеров z_n таких, что $n \in \mathcal{N}^*(t)$ при условии $w_n^*(\Delta(t)) > 0, \theta^*(t) = \chi_n^*(t + \Delta(t))$;

$\mathcal{N}^0(t)$ — множество номеров z_n таких, что $n \in \mathcal{N}^0(t)$, при условии $w_n^*(\Delta(t)) = 0, \theta^*(t) \le \chi_n^*(t + \Delta(t))$. Из определения следует $\mathcal{N}(t) = \mathcal{N}^*(t) \bigcup \mathcal{N}^0(t)$.

Значение $< \theta^*(t), w_1^*(\Delta(t)), \ldots, w_{N(t)}^*(\Delta(t)) >$ — оптимальное решение (32)-(34) — размеры подзаданий, планируемых для выполнения к моменту $t + \Delta(t)$, удовлетворяют равенствам

$$\theta^*(t) = \frac{z_n^-(t) + w_n^*(\Delta(t))}{P(t) \cdot T_n(t + \Delta(t))}, \quad n \in \mathcal{N}^*(t),$$

$$\sum_{n \in \mathcal{N}^*(t)} w_n^*(\Delta(t)) = W^+(\Delta(t)).$$

После преобразований получим:

$$\theta^*(t) = \frac{W^+(\Delta(t)) + \sum\limits_{n\in\mathcal{N}^*(t)} z_n^-(t)}{P(t)\cdot \sum\limits_{n\in\mathcal{N}^*(t)} T_n(t+\Delta(t)),} \tag{35}$$

$$w_n^*(\Delta(t)) =$$

$$= T_n(t+\Delta(t))\frac{W^+(\Delta(t)) + \sum\limits_{n\in\mathcal{N}^*(t)} z_n^-(t)}{\sum\limits_{n\in\mathcal{N}^*(t)} T_n(t+\Delta(t))} - z_n^-(t), n\in\mathcal{N}^*(t).$$

Вектор значений $\mathbf{w}^*(t) =< w_1^*(\Delta(t)), w_2^*(\Delta(t)), \ldots, w_{N(t)}^*(\Delta(t)) >$ назовем равноправным (недискриминирующим) по критерию (25) распределением выполняемых подзаданий $w_n(\Delta(t))$. При этом для всех z_n, завершенных на интервале $\Delta(t)$, показатель относительной задержки выполнения будет не меньше $\theta^*(t)$.

Указанное распределение предполагает, что все заявки-задания получают долю просмотренных фрагментов данных соответствующую их абсолютному времени решения с учетом фактического пребывания в СВС. При такой стратегии диспетчеризации принудительное увеличение доли $w_n^*(\Delta(t))$ для любой заявки z_n приведет к уменьшению оптимального значения (32) для остальных.

Предположим, что решена оптимизационная задача (32)-(34) и полученное множество подзаданий — ТР-пакет $\mathbf{w}^*(t) =< w_1^*(\Delta(t)), \ldots, w_{N(t)}^*(\Delta(t)) >$ — фрагменты данных выбранных заданий $z_n, n\in\mathcal{N}^*(t)$ помещаются в промежуточный ТР-буфер и в момент t начинают обрабатываться СВС.

Рассмотрим правило обработки пакета, при котором каждому подзаданию $w_n^*(\Delta(t))$ назначается некоторый набор ЕВ-модулей. Например, ПК-диспетчер формирует соответствующий список, где указывает какие ЕВ-модули будут выполнять

данное подзадание $w_n^*(\Delta(t))$ начиная с момента t. При этом, если некоторая подзадача $w_n^*(\Delta(t))$ решится до истечения срока $\Delta(t)$, то освободившийся ресурс — работоспособные ЕВ-модули — будут использованы ПК-диспетчером для просмотра данных других подзаданий $w_l^*(\Delta(t))$, которые еще не завершены. Описанную процедуру назовем параллельной обработкой ТР-пакета.

Пусть $r_n^*(t)$ — планируемое число ЕВ-модулей, которые в момент t должны начать решение подзадачи $w_n^*(\Delta(t))$, и завершат ее в момент $(t + \Delta(t))$, при условии, что ни один из $w_j^*(\Delta(t)), j \in \mathcal{N}(t)$ не содержит уникальный фрагмент. Поскольку, $p(t)$ — число фрагментов, которые просматриваются одним ЕВ-модулем за единицу времени, то

$$w_n^*(\Delta(t)) = r_n^*(t) \cdot p(t) \cdot \Delta(t), \quad n \in \mathcal{N}^*(t).$$

При этом

$$\sum_{n \in \mathcal{N}^*(t)} r_n^*(t) \le R(t),$$

$$r_n^*(t) = 0, n \in \mathcal{N}^0(t).$$

Из определения $W^+(\Delta(t))$ следует, что

$$W^+(\Delta(t)) = R(t) \cdot p(t) \cdot \Delta(t), \tag{36}$$

а из соотношений (35),(36) —

$$r_n^*(t) \cdot p(t) \cdot \Delta(t) = \theta^*(t) \cdot p(t) \cdot R(t) \cdot T_n(t + \Delta(t)) - z_n^-(t), \; n \in \mathcal{N}^*(t),$$

где

$$\theta^*(t) = \frac{\Delta(t)}{\sum\limits_{n \in \mathcal{N}^*(t)} T_n(t + \Delta(t))} + \frac{\sum\limits_{n \in \mathcal{N}^*(t)} z_n^-(t)}{R(t) \cdot p(t) \cdot \sum\limits_{n \in \mathcal{N}^*(t)} T_n(t + \Delta(t))}.$$

Вектор значений $\mathbf{r}^* =< r_1^*(t), r_2^*(t), \ldots, r_{N(t)}^*(t) >$ является оптимальным распределением вычислительных ресурсов $R(t)$ по критерию (25) при запуске параллельной обработки пакета в момент t. Если какие-то подзадачи $w_l^*(\Delta(t))$ будут решены на интервале $\Delta(t)$, то фактическое суммарное время выполнения данного ТР-пакета $\mathbf{w}^*(t)$ окажется меньше $\Delta(t)$, и очередной контрольный момент наступит раньше $(t + \Delta(t))$.

Предположим, как и ранее, что в ТР-буфере в момент t находится пакет выбранных подзаданий $\mathbf{w}^*(t) =< w_1^*(\Delta(t)), w_2^*(\Delta(t)), \ldots, w_{N(t)}^*(\Delta(t)) >$, который планируется выполнить в течение периода $\Delta(t)$. Правило обработки заданий из пакета будем называть *последовательным*, если подзадания $w_n^*(\Delta(t))$ выполняются одно за другим в монопольном режиме СВС, т.е. каждое сразу на всех работоспособных ЕВ-модулях. Как только очередная подзадача $w_i^*(\Delta(t))$ завершена, начинается просмотр следующего поднабора данных $w_j^*(\Delta(t))$.

Обозначим через Δ_n^* промежуток времени, которое необходимо для решения выбранной подзадачи $w_n^*(\Delta(t))$ в монопольном режиме, при условии,что само задание z_n не будет завершено за $\Delta(t)$. Для всех подзаданий ТР-пакета априори требуется выполнение равенства

$$\sum_{n \in \mathcal{N}^*(t)} \Delta_n^* = \Delta(t).$$

Поскольку $w_n^*(\Delta(t))$ — число фрагментов данных, которые будут обработаны в монопольном режиме на $R(t)$ работоспособных ЕВ-модулях за время Δ_n^*, если в них нет уникального, то должно выполняться равенство

$$w_n^*(\Delta(t)) = p(t) \cdot R(t) \cdot \Delta_n^*, \; n \in \mathcal{N}^*(t).$$

Из последнего равенства и (36) следует

$$\Delta_n^* = \frac{w_n^*(\Delta(t))}{p(t)\cdot R(t)} =$$

$$= \frac{1}{p(t)\cdot R(t)} \cdot \left[\theta^*(t)\cdot p(t)\cdot R(t)\cdot T_n(t+\Delta(t)) - z_n^-(t)\right],$$

где

$$\theta^*(t) = \frac{\Delta(t)}{\sum\limits_{n\in\mathcal{N}^*(t)} T_n(t+\Delta(t))} + \frac{1}{p(t)\cdot R(t)} \cdot \frac{\sum\limits_{n\in\mathcal{N}^*(t)} z_n^-(t)}{\sum\limits_{n\in\mathcal{N}^*(t)} T_n(t+\Delta(t))}.$$

Для соотношений, приведенных выше, справедливы равенства:

$$\frac{w_n^*(\Delta(t))}{W^+(t)} = \frac{r_n^*(t)}{R(t)} = \frac{\Delta_n^*}{\Delta(t)}, n \in \mathcal{N}^*(t).$$

Таким образом, априори для каждого задания $z_n, n \in N^*(t)$, доля обработанных фрагментов данных, задействованных вычислительных ресурсов, затраченного "процессорного" времени совпадают.

Для однородного случая рассмотрим параметрический метод планирования и формирования ТР-пакета, опирающийся на показатель относительной задержки выполнения заданий.

Предположим, что в СВС находится множество заданий $\mathcal{Z}(t)$, и для каждого z_n известно $z_n^-(t)$ — число фрагментов данных, обработанных для z_n в момент t. На основе текущей информации вычислим оценки показателей относительной задержки к моменту $t+\Delta(t)$:

$$\chi_n(t+\Delta(t)) = \frac{z_n^-(t)}{P(t)T_n(t+\Delta(t))}, n \in \mathcal{N}(t).$$

Обозначим:

$$x_* = \min_{n \in \mathcal{N}(t)} \{\chi_n(t + \Delta(t))\},$$

$$x^* = \max_{n \in \mathcal{N}(t)} \{\chi_n(t + \Delta(t))\},$$

и рассмотрим произвольное x, удовлетворяющее условию $x_* \leq x \leq x^* < 1$.

Пусть x — некоторое "запланированное" значение показателя относительной задержки, которое должно быть достигнуто на интервале $\Delta(t)$. Для каждого $x \in [x_*; x^*]$ введем подмножество номеров $\mathcal{N}(x,t)$ заданий z_i из $\mathcal{N}(t)$, для которых показатель относительной задержки ниже запланированного параметра, т.е. :

$$\mathcal{N}(x,t) = \{i \in \mathcal{N}(t) |\ \chi_i(t + \Delta(t)) \leq x\}.$$

Для каждого $j \in \mathcal{N}(x,t)$ обозначим через $w_j(x, \Delta(t))$ размер поднабора — число фрагментов данных, которые необходимо просмотреть на интервале $\Delta(t)$ для задания z_j, чтобы было достигнуто запланированное значение x для показателя относительной задержки. Следуя ранее приведенным рассуждениям, можно утверждать, что для любого $x \in [x_*; 1]$ и $z_j, j \in \mathcal{N}(x,t)$

$$w_j(x, \Delta(t)) = \max_w w,$$

при ограничениях

$$\frac{z_j^-(t) + w}{P(t)T_j(t + \Delta(t))} \leq x,$$

$$w < \mathbf{Z}_j - z_j^-(t).$$

Фактически, для любого $x \in [x_*; 1]$ и $j \in \mathcal{N}(x,t)$ значение $w_j(x, \Delta(t))$ определяется из условий:

$$w_j(x,\Delta(t)) = \begin{cases} x \cdot P(t) \cdot T_j(t+\Delta(t)) - z_j^-(t), \\ \qquad\qquad \text{если } x \cdot P(t) \cdot T_j(t+\Delta(t)) < \mathbf{Z}_j, \\ \mathbf{Z}_j - z_j^-(t), \\ \qquad\qquad \text{если } x \cdot P(t) \cdot T_j(t+\Delta(t)) \geq \mathbf{Z}_j. \end{cases}$$

Для любого $x \in [x_*; x^*]$ и всех заданий z_j из $\mathcal{N}(x,t)$ вычислим величину

$$W(x,\Delta(t)) = \sum_{j \in \mathcal{N}(x,t)} w_j(x,\Delta(t)).$$

При этом для $x \in [x^*; 1]$

$$W(x,\Delta(t)) = \sum_{j \in \mathcal{N}(t)} w_j(x,\Delta(t)).$$

Полученные при различных значениях $x \in [x_*; 1]$ значения $W(x,\Delta(t)), \chi(t+\Delta(t)), \ldots$ объединяются в таблицы. На рис. 3. приведен пример диаграммы $< W, x >$.

Составленные таблицы облегчают планирование и выбор стратегий среди различных диспетчерских политик. Действительно, если известна величина $W^0(x,\Delta(t))$ — допустимый общий объем ТР-пакета, то из таблиц соотношений $< W, x >$ можно определить список заданий и размер соответствующих поднаборов данных, которые должны быть обработаны в операционном окне $\Delta(t)$. Если же задается "плановое" значение параметра x^0, то из таблиц можно определить состав ТР-пакета и, что очень важно, $W^0(x^0,\Delta(t))$ — в данном случае — запрос-требование на общий объем работ, которые должны быть выполнены на интервале $\Delta(t)$ для достижения "планового показателя" x^0.

Диаграмма $< W, x >$ является агрегированным отображением состояния очереди заявок-запросов на выполнение *работ*

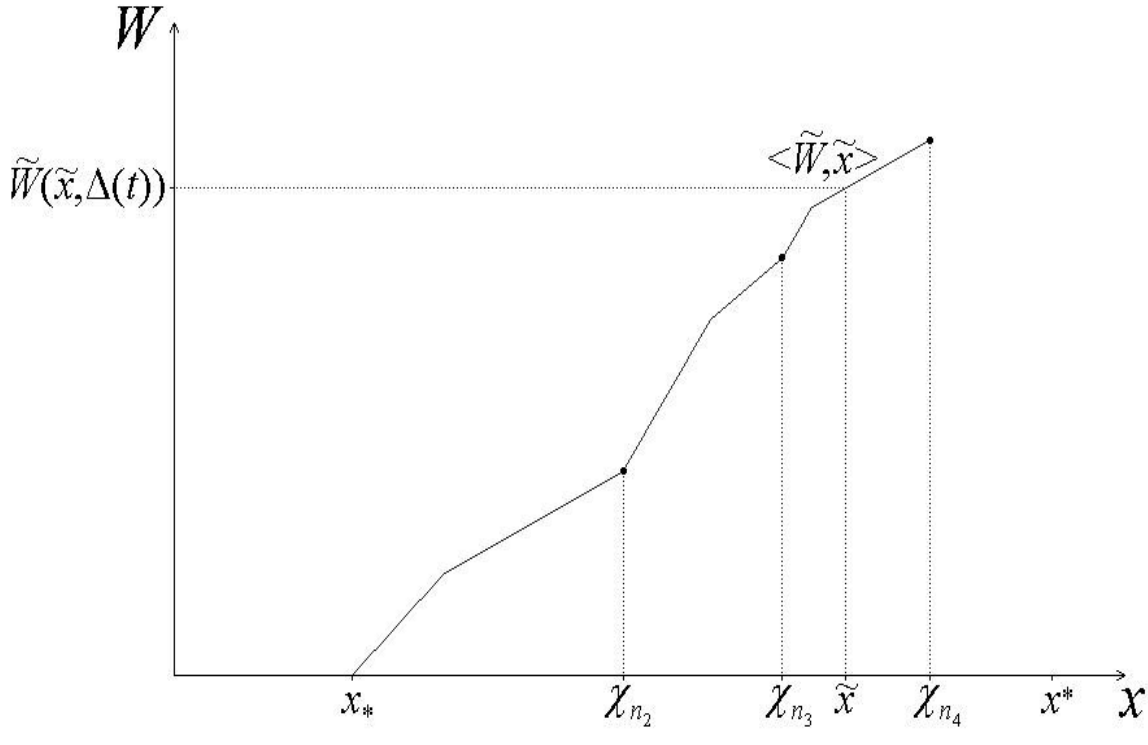
W
$\widetilde{W}(\widetilde{x},\Delta(t))$
$<\widetilde{W},\widetilde{x}>$
x_*
χ_{n_2}
χ_{n_3}
$\widetilde{x}$
χ_{n_4}
x^*
x

Рис. 3

и, соответственно, на предоставление вычислительных ресурсов. В системе CORSAR диаграмма оперативно обновляется и воспроизводится на мониторе администратора СВС.

2.7. Планирование работ в комплексе CORSAR.

В гетерогенной системе при формировании ТР-пакета в ПК-план последовательно, при различных значениях коэффициентов целевой функции, решается следующая М-задача:
при заданных $\mathbf{Z}_n, z_n^-(t), R^m(t), p_k^m(t), d_n, t_n^0, c_1 \geq 0, c_2 \geq 0, c_3 \geq 0$ найти

$$\phi^*(t) = \max_{w,\nabla,r,\omega,\theta,\varphi} [c_1\theta(t) + c_2\omega(t) + c_3\varphi(t)] \tag{37}$$

при ограничениях

— на показатель относительной задержки выполнения заданий на интервале $\Delta(t)$ и соблюдение ДСО:

$$0 \leq \theta(t) \leq \chi_n(t+\Delta(t)), n \in \mathcal{N}(t), \tag{38}$$

$$\left.\begin{array}{c} \chi_n(t+\Delta(t)) = \dfrac{z_n^-(t) + w_n(\Delta(t))}{P_k(t)} \cdot \dfrac{1}{T_n(t)+\Delta(t)} \\ n \in \mathcal{N}_k(t), k = \overline{1,K}; \end{array}\right\} \tag{39}$$

$$z_n^+(t+\Delta(t)) = \mathbf{Z}_{n(k)} - z_{n(k)}^-(t) - w_{n(k)}(\Delta(t)) \tag{40}$$

$$z_n^+(t+\Delta(t)) = \sum_{m=1}^{M} p_k^m(t)\cdot R^m(t)\cdot \nabla_{n(k)}^m(t+\Delta(t)), \tag{41}$$

$$n \in \mathcal{N}_k(t), k = \overline{1,K},$$

$$\nabla_{n(k)}^m(t+\Delta(t)) \geq 0, m = \overline{1,M}, n \in \mathcal{N}(t).$$

$$\nabla^m(t+\Delta(t), d_j) = \sum_{k=1}^{K} \sum_{n \in \mathcal{N}(t,d_j) \bigcap \mathcal{N}_k(t)} \nabla_{n(k)}^m(t+\Delta(t)) =$$

$$= \sum_{n\in\mathcal{N}(t,d_j)} \nabla^m_{n(k)}(t+\Delta(t)), \quad m=\overline{1,M}, \quad j\in\mathcal{N}(t). \qquad (42)$$

$$\left.\begin{array}{c} \omega^m(t+\Delta(t),d_j) = \dfrac{t+\Delta(t)+\nabla^m((t+\Delta(t)),d_j)-t^0_j}{d_j-t^0_j}, \\ j\in\mathcal{N}(t),\ m=\overline{1,M}; \end{array}\right\} \qquad (43)$$

$$\left.\begin{array}{c} \omega(t)\le 1-\omega^m(t+\Delta(t),d_j), \\ \omega(t)\le 0, \\ m=\overline{1,M}, j\in\mathcal{N}(t); \end{array}\right\} \qquad (44)$$

$$0\le w_n(\Delta(t))\le \mathbf{Z}_n - z^-_n(t), n\in\mathcal{N}(t), \qquad (45)$$

— на число ЕВ-модулей, планируемых для обработки фрагментов k-го вида, должны выполняться неравенства:

$$\left.\begin{array}{c} \sum\limits_{k=1}^{K} r^m_k(t)\le R^m(t), \quad m=\overline{1,M}, \\ r^m_k(t)\ge 0, \quad k=\overline{1,K}, \quad m=\overline{1,M}; \end{array}\right\} \qquad (46)$$

— на суммарное число фрагментов k-го вида, обработанных за время $\Delta(t)$:

$$\sum_{n\in\mathcal{N}_k(t)} w_{n(k)}(\Delta(t)) = \sum_{m=1}^{M} p^m_k(t)\cdot r^m_k(t)\cdot\Delta(t), \quad k=\overline{1,K}. \qquad (47)$$

$$0\le\varphi(t)=\sum_{n=1}^{N(t)} w_n(\Delta(t)). \qquad (48)$$

М-задача (37) с условиями (38)-(48) и различными, заранее заданными c_1, c_2, c_3 позволяет анализировать и выбирать управления в момент времени t.

Рассмотрим один из возможных вариантов. Разобьем формирование ТР-пакета на несколько взаимосвязанных этапов.

ЭТАП 1. В (37) положим $c_2 = \hat{c}_2 \gg c_1 = c_3 = \hat{c}$, например, $\hat{c}_2 = 10^6, \hat{c} = 1$, и найдем решение М-задачи (37) с ограничениями (38)-(48). Полученное максимальное значение $\omega(t)$ обозначим $\hat{\omega}^*(t)$. Величина $\hat{\omega}^*(t)$ показывает, на сколько могут быть превышены некоторые ДСО в наихудшем случае.

ЭТАП 2. В условиях (44) зафиксируем параметр $\omega(t)$, положив его равным $\hat{\omega}^*(t)$, т.е. $\omega(t) = \hat{\omega}^*(t)$, и исключим его из функционала, положив, $c_2 = 0$. Присвоим в (37) остальным коэффициентам значения $c_1 = 10^6, c_3 = 1$, вновь решим М-задачу. Из полученного вектора выберем только значение $\theta^*(t)$, которое показывает, что для управлений, удовлетворяющих условиям (38)-(48) показатели относительной задержки выполнения не меньше $\theta^*(t)$ для любого $z_n,\ n \in \mathcal{N}(t)$. Таким образом, на этапе 1 была получена оценка $\hat{\omega}^*(t)$ возможного превышения ДСО в наихудшем случае, которая зафиксирована при решении М-задачи на данном этапе. На втором этапе получено наилучшее гарантированное значение $\theta^*(t)$.

ЭТАП 3 (заключительный). В функционале (37) положим $c_1 = c_2 = 0, c_3 = 10^6$, в ограничениях (44) сохраняем фиксированное значение $\omega(t) = \hat{\omega}^*(t)$ а в (38) полагаем $\theta(t) = \theta^*(t)$, и вновь решаем М-задачу. На этом этапе на подмножестве всех допустимых управлений, ограниченных значениями $\hat{\omega}^*(t), \theta^*(t)$, максимизируется общий объем *работ*. В результате решения (37) при фиксированных $\hat{\omega}^*(t), \theta^*(t)$ строится вектор оптимального управления $\mathbf{w}^*(t)$, компоненты которого $< w_1^*(\Delta(t)), w_2^*(\Delta(t)), ..., w_{N(t)}^*(\Delta(t)) >$ определяют "поименный" и количественный состав ТР-пакета. Сформированный ТР-пакет загружается в ТР-буфер и в контрольный момент t начинает выполняться на всех работоспособных ЕВ-модулях .

Следует отметить, что можно рассматривать и другие способы использования М-задачи, отличные от описанного выше. Однако, предложенная лексикографическая схема с фор-

мально скалярным функционалом позволяет, с одной стороны, учесть все существенные ограничения, а с другой — обеспечивает максимальную эффективность СВС.

Заключение

С позиций математического моделирования проблема управления ресурсоемкими вычислениями в условиях неопределенности является многокритериальной. По сути от администратора СВС требуется достижение одновременно нескольких различных целей:

— эффективно использовать разнотипные вычислительные устройства и добиваться максимальной производительности СВС при выполнении разнородных заявок;

— завершать каждое задание до наступления назначенного (предписанного) срока;

— в условиях неопределенности распределять вычислительные ресурсы с учетом реальных, фактически необходимых затрат, требуемых для решения конкретной задачи.

Управление процессом обработки осуществляется на основе математических моделей, в рамках которых:

— прежде всего строятся гарантированные оценки для изначально неизвестных величин;

— на основании гарантированных оценок находится вектор управления, определяющий недискриминирующее распределение *работ* и вычислительных ресурсов в условиях неопределенности;

— для каждой задачи происходит проверка достаточных условий выполнения ДСО;

— повышается эффективность использования СВС.

На практике вследствие соблюдения правила SWAP в первую очередь решаются задачи с небольшим абсолютным временем выполнения, а при пакетной обработке все задания, не содержащие уникальных фрагментов, завершаются в последнюю очередь.

В случае возможного нарушения директивных сроков окончания осуществляется релейное переключение на обработку со-

ответствующих заданий.

Для реализации принципов, заложенных в основу математических моделей, был разработан и программно реализован комплекс CORSAR - трехуровневая система планирования и диспетчеризации ресурсоемких разнородных вычислений в гетерогенной высокопроизводительной СВС.

Комплекс CORSAR имеет ряд преимуществ:

— предварительная фильтрация поступающего потока заявок предотвращает появление некорректно составленных заданий, выполнение которых приведет к заведомо бесполезному расходованию вычислительных ресурсов, а использование подключаемых библиотек программных компонентов приводит в действие простой и гибкий механизм настройки входных фильтров;

— набор подгружаемых сменных ядер дает возможность администратору задействовать различные динамические стратегии диспетчеризации при выполнении работ;

— разбиение массива исходных данных на отдельные фрагменты позволяет:

а) осуществлять режим скользящего планирования в контрольных точках;

б) проводить распараллеливание и оптимизацию состава пакета *работ*;

в) решать наименее ресурсоемкие задачи в первую очередь;

г) завершать задания в срок;

д) минимизировать производственные потери;

— введение и реализация подсистемы ресурсных агентов для управления доступом к вычислительным ресурсам значительно упрощает переход на принципиально новые классы ускорителей, поскольку организация промежуточного уровня управления группой узлов обеспечивает масштабируемость СВС при модернизации и развитии;

— декомпозиция планирования и управления повышает эффективность использования вычислительных ресурсов, позволяет избежать падения относительной производительности при увеличении общей мощности СВС.

В заключение хочется отметить, что при принятии решений в условиях неопределенности CORSAR выступает носителем особого рода знания, воплощая в себе убежденность предвидения, всеми своими действиями свидетельствуя, что хаос поддается обузданию, равновесие интересов может быть достигнуто и разум восторжествует.

Литература

1. Foster I., Kesselman C., Nick J. et al. The physiology of the grid: An open grid services architecture for distributed systems integration.
http : //www.globus.org/research/papers.html

2. Foster I., Kesselman C., Tuecke S. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. // Int. J. of Supercomputer Applications. 2001. V. 15. N. 3. P. 200-222.

3. Gentzsh W. Sun grid engine: Towards creating a compute power grid // Proc. 1st IEEE/ACM Int. Symp. on Cluster Computing and the Grid. IEEE, 2002. P. 35-36.

4. Коновалов М.Г., Малашенко Ю.Е., Назарова И.А. Модели и методы управление заданиями в системах распределенных вычислительных ресурсов. М.: ВЦ РАН, 2009.
*http : //www.ccas.ru/depart/malashen/papper /konovalov_*2009*_preprint.pdf*

5. Коновалов М.Г., Малашенко Ю.Е., Назарова И.А. Управление заданиями в гетерогенных вычислительных системах // Изв. РАН. ТиСУ. 2011. № 2. С. 72–90.

6. Каляев И.А., Левин И.И. Реконфигурируемые мультиконвейерные вычислительные структуры для решения потоковых задач обработки информации и управления // Тр. междунар. научн.-практ. конф. "Суперкомпьютерные

технологии: разработка, программирование, применение". Ростов-на-Дону: Южный федеральный ун-т, 2010. Т. 1. С.100-102.

7. Ронжин А.В., Суриков В.Н. О математических проблемах применения высокопроизводительных вычислительных систем для решения задач случайного поиска // Тр. междунар. научн.-практ. конф. "Суперкомпьютерные технологии: разработка, программирование, применение". Ростов-на-Дону: Южный федеральный ун-т, 2010. Т. 2. С. 239-243.

8. Малашенко Ю.Е., Назарова И.А. Модель управления разнородными вычислительными заданиями на основе гарантированных оценок времени выполнения. // Изв. РАН. ТиСУ. 2012. № 4. С. 29-38.

9. Забродин А.В., Левин В.К., Корнеев В.В. Массово-параллельные системы МВС-100 и МВС-1000. // Сб. тр. научн. сессии МИФИ - 2000. М.: МИФИ, 2000. Т. 2. С. 194-195.

10. Баранов А.В., Киселев А.В., Корнеев В.В. и др. Программный комплекс "Пирамида"организации параллельных вычислений с распараллеливанием по данным // Тр. междунар. суперкомпьютерной конф. и конф. молодых ученых "Научный сервис в сети Интернет: Суперкомпьютерные центры и задачи". М.: МГУ, 2010. С. 299-302.

11. $http://www.parallel.ru/tech/tech_dev/mpi.html$

12. $http://www.opennet.ru/docs/RUS/MPI_intro$

13. *http : //ru.wikipedia.org/wiki/MPI*

14. Гермейер Ю.Б. Введение в теорию исследования операций. М.: Наука, 1971.

15. Сухарев А.Г., Тимохов А.В., Федоров В.В. Курс методов оптимизации. М.: Наука, 1986.

16. Данциг Дж. Б. Линейное программирование, его применения и обобщения. М.: Прогресс, 1966.

17. Гасс С. Линейное программирование (методы и проложения). М.: Гос. изд-во физ.-мат. лит., 1961.

18. Малашенко Ю.Е., Назарова И.А. Управление ресурсоемкими разнородными вычислительными заданиями с директивными сроками окончания. // Изв. РАН. ТиСУ. 2012. № 5. С. 3-11.

19. Stankovic J.A., Spuri M., Ramamritham K. et. al. Deadline Scheduling for Real-Time Systems, EDF and Related Algorithms. Boston: Kluwer, 1998.

20. Голосов П.С., Козлов М.В., Малашенко Ю.Е. и др. Модель системы управления специализированным вычислительным комплексом. М.: ВЦ РАН, 2010.
*http : //www.ccas.ru/depart/malashen/papper /golosov_*2010*_preprint.pdf*

21. Козлов М.В., Малашенко Ю.Е., Назарова И.А. Гарантированные оценки распределения вычислительных ресурсов в условиях неопределенности. М.: ВЦ РАН, 2011.

http : *//www.ccas.ru/depart/malashen/papper /kozlov_*2011*_preprint.pdf*

22. Козлов М.В., Малашенко Ю.Е., Назарова И.А. и др. Анализ режимов управления вычислительным комплексом в условиях неопределенности. М.: ВЦ РАН, 2011. *http* : *//www.ccas.ru/depart/malashen/papper /ronzhin_*2011*_preprint.pdf*

23. Голосов П.С., Козлов М.В., Малашенко Ю.Е.и др. Анализ управления заданиями специализированными вычислительными заданиями в условиях неопределенности // Изв. РАН. ТиСУ. 2011. № 6. С. 63-79.

24. Rasmussen R.V., Trick M.A. Round Robin Scheduling - A Survey // European Journal of Operational Research. 2008. V. 188. Iss. 3. P. 617-636.

СОДЕРЖАНИЕ