УДК 621.391

Д-р техн. наук. С.В. Листровой,
Аспирант М.С. Курцев

ПОДХОД К ОРГАНИЗАЦИИ ПЛАНИРОВАНИЯ РАСПРЕДИЛЕНИЕМ РЕСУРСОВ В СИСТЕМАХ УПРАВЛЕНИЯ ЖЕЛЕЗНОДОРОЖНЫМ ТРАНСПОРТОМ

Ключевые слова: *распределенные вычислительные системы, планирование, задания, ресурсы.*

Введение

В настоящее время во всех отраслях экономики, в том числе и на транспорте, происходят кардинальные изменения в области информационных технологий и систем. Так, в соответствии с Транспортной стратегией Украины до 2020 года, основными направлениями ее реализации являются обеспечение доступности и качества транспортных услуг за счет создания комплексных информационных систем управления, контроля и идентификации грузов и контейнеров. Также на железно-дорожном транспорте идут процессы модернизации существующих информационных систем и их интеграции в единую автоматизированную систему управления грузовыми перевозками Украинских железных дорог. Эти процессы должны основываться на единой архитектуре построения систем, учитывающей особен-ности современных и перспективных информационных технологий, таких как Grid технологии и технологии облачных вычисле-ний. Одной из основных проблем реализации данных технологий является вопрос организации планирования распределения ресурсов при выполнении заданий возни-кающих в распределенной системе управления. Поэтому представляется актуальным разработка общих универсальных подходов к решению данной задачи.

Постановка и формализация задачи. Как показано в работах [1,2] в общем виде данная задача в произвольной распре-деленной вычислительной среде может быть описана в виде кортежа:

$$Schedule = (T, R, ComLinkThroughout, MappingEvent, F(Map), SchedMethod, ResChar, TaskChar, StratSchedul(Focus), StratExec, ObjectFunc, Mode, NoLevel),$$

где T – множество заданий глобального потока;

R – множество ресурсов;

$ComLinkThroughout$ – множество коммуни-кационных каналов связи и их пропускной способности между заданиями и ресурсами;

$MappingEvent$ – время отображения заданий на ресурсы;

$F(Match)$ – функция отображения заданий на ресурсы;

$ResChar$ – множество характеристик ресурсов R;

$TaskChar$ – множество характеристик заданий T;

$StratSchedul(Focus)$ – множество стратегий планирования заданий;

$StratExec$ – множество стратегий выполнения заданий на ресурсах;

$ObjectFunc$ – множество целевых функций;

$Mode$ – множество различных режимов планирования;

$NoLevel$ – уровень планирования.

Множество T определяет множество глобального потока заданий, поступивших на обработку в распределенную вычислительную систему (РВС): $T = (T_1, T_2, T_3, ..., T_m)$.

Множество R определяет множество всех ресурсов РВС: $R = (R_1, R_2, R_3, ..., R_n)$.

Время отображения $MappingEvent$ заданий на ресурсы определяется минимальным количеством заданий для планирования, определяемым количеством доступных ресурсов, и/или минимальным временем для процедуры мониторинга состояния ресурсов и заданий системы.

Функция отображения F заданий T на ресурсы R системы представляет матрицу соответствия:

$$F(Matching): T \times R \times \times ComLinkThroughout \rightarrow R^{+},$$

где *Matching* – матрица соответствия планируемых заданий T ресурсам R РВС с учетом пропускной способности множества коммуникационных каналов связи *ComLinkThroughout* между спланированными заданиями и ресурсами РВС.

Множество методов планирования *SchedMethod* определяет методы планирования заданий T на ресурсы R. Множество стратегий планирования заданий $StratSchedul(Focus)$, определяет стратегии планирования, описываемые кортежем:

$$StratSchedul(Focus) = (SystemOrient, UserOrient),$$

где *SystemOrient* – системно-ориентирова-нная стратегия планирования, ориентиро-ванная (сфокусированная) на повышение производительности и пропускной способности РВС (уровень метапланировщика потоков заданий);

UserOrient – стратегия планирования заданий, ориентированная (сфокусированная) на пользователей (уровень приложений (локального менеджера ресурсов или локального планировщика)). Множество стратегий *StratExec* выполнения заданий T на выделенных для них из множества ресурсов R, описываемых кортежем:

$$StratExec = (Deadline, Budget),$$

где *Deadline* – директивные сроки выполнения заданий;

Budget – ограничения на бюджет (стоимость) выполнения заданий.

Множество целевых функций *ObjectFunc*, характеризующих работу РВС, описываемых кортежем:

$$ObjectFunc = (Utilization, LoadBalance, Time(Cost), Makespan, Penalty),$$

где $Utilization(NoResources, ResUtiliz)$ – количество задействованных ресурсов *NoResources* и величина коэффициента использования *ResUtiliz* ресурсов R системы;

LoadBalance – балансировка загрузки ресурсов R;

$Time(Cost)$ – время (стоимость) выполнения заданий глобальной очереди;

Makespan - максимальное время завершения выполнения заданий глобальной очереди T на ресурсах R;

Penalty – величина штрафов за превышение допустимого времени выполнения или директивного срока выполнения заданий.

Множество режимов планирования выполнения заданий *Mode* в РВС, описываемых кортежем: $Mode = (Batch, Online)$, *Batch* – пакетный режим выполнения заданий;

Online – режим планирования и выполнения заданий непосредственно после их поступления в РВС на обработку.

Множество уровней планирования выполнения заданий *NoLevel*, описываемых кортежем:

$$NoLevel = (Global, Local),$$

где *Global* – глобальный уровень планирования заданий – уровень метапланировщика потоков заданий;

Local – локальный уровень планирования заданий – уровень локальных планировщиков (планировщиков систем управления локальными ресурсами) пакетов локальных заданий.

Множество *ResChar* определяет множество характеристик ресурсов R системы, описываемых кортежем:

$$ResChar = (NoCores, NoProcessors, MinDisk, MinMemory, OS, MinCPUFreq, MaxCPUFreq, MinCPUPower, MaxPUPower, Architecture, ComLinkThroughout\ (R_j)),$$

где *NoCores* – количество процессорных ядер ресурса;

NoProcessors – количество процессоров ресурса;

Disk - объем дисковой памяти ресурса;

Memory - объем оперативной памяти ресурса;

OS - тип операционной системы, используемой на ресурсе;

MinCPUFreq - минимальная частота процес-сора ресурса;

MaxCPUFreq - максимальная частота процессора ресурса;

MinCPUPower - минимальное энергопотребление процессором;

MaxPUPower - максимальное энергопотребление процессором;

Architecture - тип архитектуры ресурса; *ComLinkThroughout* (R_j) - пропускная способность коммуникационных каналов связи к ресурсу R_j.

Множество *TaskChar* определяет множество характеристик заданий T системы, описываемых кортежем:

$$TaskChar = (TaskResource\ Requirements\ (TaskType,\ NoCores, minDisk, minMemory, Software,\ Hardware, OS), walltime, deadline, budget),$$

где *TaskResource Requirements* - ресурсные требования задания;

TaskType - тип заданий – параллельное (Parallel), последовательное (Serial);

NoCores - требуемое для выполнения количество ядер;

minDisk - минимальный объем дисковой памяти;

minMemory - минимальный объем оперативной памяти;

Software - требуемое программное обеспечение;

Hardware - требуемое аппаратное обеспечение;

OS - требуемая операционная система; *walltime* - ожидаемое время выполнения заданий на ресурсе;

deadline - директивный срок выполнения задания;

budget - ограничение (бюджет) стоимости выполнения задания, определяется по формуле:

$$cost(walltime) + penalty(deadline) \le budget.$$

Общая концепция создания управления диспетчеризацией в Grid. В системе оптимального планирования распределением ресурсов в Grid регионального уровня, с точки зрения повышения эффективности и гибкости использования ресурсов предлагается иметь четырех уровневую, иерархическую структуру управления и планирования сочетающую в себе централизованное и децентрализованное управление. Первый уровень управления должен обеспечить координацию перераспределения ресурсов и заданий между группами регионов глобальной сети второй уровень управления должен обеспечить требуемое качество обслуживания динамически меняющихся виртуальных сообществ в сети в рамках региона, предоставляя им возможности полностью использовать все ресурсы сети [3,4,5]. Третий уровень независимо от первого уровня, на основе общего пула заданий осуществляет перераспределение заданий между кластерами и осуществляя обмен заданиями между собой. И четвертый уровень осуществляет планирование выполнения заданий внутри кластера. Для этого в сети целесообразно иметь центральный пункт управления, основными функциями которого, является обслуживание общего пула заданий, и координации работы диспетчеров второго уровня, практически не вмешиваясь в процесс их планирования, но предоставляя диспетчерам второго уровня необходимую информацию для процесса планирования. Диспетчеры второго уровня сами выбирают задания из общего пула и могут пересылать их друг другу и отправлять непосредственно на решение в определенный кластер.

После того как виртуальные сообщества сформированы, и сообщества предъявили в центральный пункт управления свои задания, он им предлагает за определенную цену возможности подсоединения дополнительных ресурсов сети, которые в сети остаются, не задействованы. При этом предполагается, что все желающие предоставлять свои ресурсы оповещают об этом центральный пункт управления, который берет на себя функции посредника по предоставлению дополнительных ресурсов диспетчерам второго уровня.

Посредническая деятельность центрального пункта управления позволит разгрузить диспетчеров второго уровня от процесса анализа и обработки состояния всех ресурсов сети, и полноценно заниматься только планированием распределения ресурсов через общий пул заданий в сети. На меж-региональном уровне центральные пункты управления каждого региона взаимо-действуют между собой, образуя распределенный центральный пункт управления глобальной сети соответствующий четвертому уровню управления создающему потоки информации о состоянии ресурсов в регионах глобальной сети [6,7,8]. Рассмотрим принципы построения Grid систем, в которых распределение заданий осуществляется в гетерогенной вычислительной среде, и они в общем случае должны решать такие задачи как: определение функционального назначения каждой компоненты архитектуры Grid; определение общих принципов взаимодействия компонентов Grid; создание математического и программного обеспечения гарантирующего эффективное и надежное функционирование Grid систем.

Будем рассматривать Grid систему как гетерогенную среду, в которой каждый ресурс R_i характеризуется вектором характеристик $(\alpha_1^{R_i}, \alpha_2^{R_i}, ..., \alpha_q^{R_i})$, где, каждое задание $З_i$ тоже будем характеризовать характеристиками $(\alpha_1^{З_i}, \alpha_2^{З_i}, ..., \alpha_q^{З_i})$. Задание $З_i$ может быть выполнено на ресурсе R_i, если выполняются неравенства

$$\alpha_1^{R_i} \le \alpha_1^{З_i}; \alpha_2^{R_i} \le \alpha_2^{З_i}; ..., \alpha_q^{R_i} \le \alpha_q^{З_i}.$$

Распределение заданий рассмотрим на основе принципа раздельного распределения задач см. рис.1, но при этом распределение с помощью диспетчера осуществляться будет не статично [9,10], как это делается в известных раздельных схемах распределения задач, а динамично и непрерывно на основе следующей процедуры *D*.

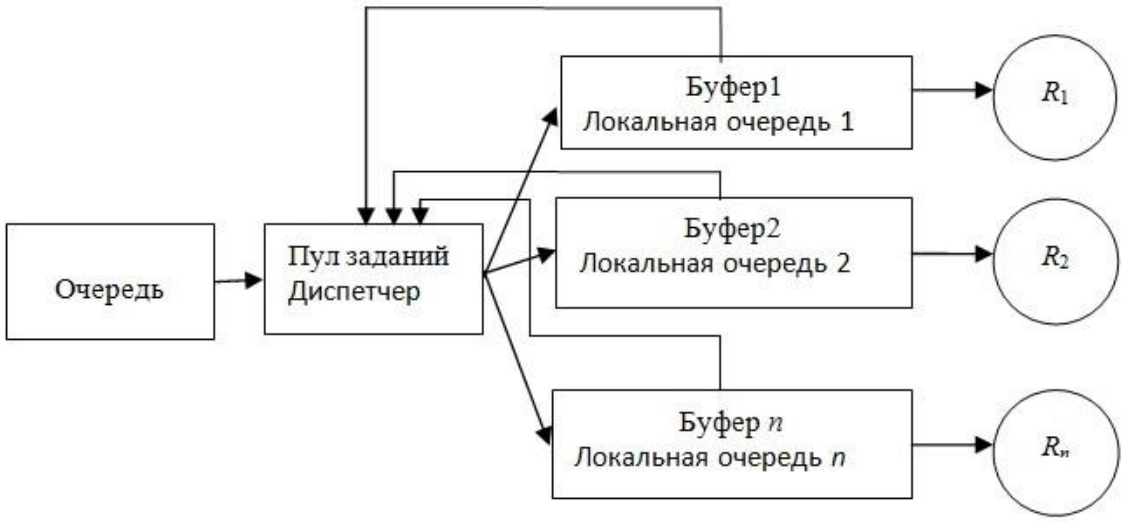

Рис. 1 - Раздельное распределение задач в гетерогенной среде

Процедура D

На основе сведений о задачах и свободных ресурсах, в диспетчере создается таблица соответствия, столбцам которой соответствуют ресурсы {Ri}, строкам задачи из сформированной очереди заданий. На пересечении строки и столбца будем ставить единицу, если из сравнения и следует, что данная задача может быть реализована на имеющемся свободном ресурсе и ноль в противном случае. В дальнейшем при планировании предлагается использовать одну из двух следующих процедур. Первая основана на решении задачи о наименьшем покрытии (ЗНП),

находим минимальное число ресурсов, на котором сформированная очередь заданий может быть выполнена, и отправляем задания на решение в выделенные ресурсы. Далее очередь пополняется новыми заданиями, и процедура повторяется, те задания, для которых мы в таблице имеем нулевые строки, оправляются обратно в очередь и делается сообщение администратору о невозможности выполнения данного задания не на одном ресурсе; вторая основана на использовании. Математической моделью данной процедуры является задача линейного булевого программирования $L_t = \sum_{j=1}^{n} x_j(t_k) \to \min$ при ограничениях:

$$\sum_{j=1}^{n} \beta_{ji} x_j(t_k) \geq 1; i = \overline{(1,m)}$$

$\beta_{ji} \in \{0,1\}$ xj (tk) {0,1},

где m – количество заданий, подлежащих планированию;

n – количество ресурсов системы, доступных и свободных на момент планирования tk [T0, Tn].

Планирование осуществляется на интервале времени [T0, Tn], где T0 – время начала планирования; Tn – время завершения планирования заданий. Задачу можно рассматривать как задачу определения минимального числа столбцов в булевой матрице *B*, покрывающего все строки данной матрицы, элементы которой в контексте решения задачи планирования интерпретируются следующим образом: столбцам соответствуют доступные и свободные на момент планирования ресурсы распределенной вычислительной системы, а строкам – задания, подлежащие планированию, которые должны быть решены на этих ресурсах. Особенностью здесь является то что расстановка единиц в матрице *B* динамически изменяется.

Вторая процедура базируется на методе групповой выборки – это такой метод, при реализации которого из очереди заданий обслуживается несколько заданий одновременно. Выбираются задания, которые требуют для реализации ресурсы разных типов, и чтобы сумма их приоритетов была максимальной. В случае наличия равнозначных заданий выбирают более «старые». Поэтому необходимо выбрать из очереди как можно большее количество заданий, которые используют различные типы ресурсов, и сумма приоритетов выбранных заданий должна быть максимальна. Причем стремление к максимуму суммы приоритетов выбранных заданий является главным критерием при выборе заданий из очереди. Будем полагать, что задание, требующее ресурс определенного типа может быть выполнено на любом ресурсе данного типа. В принципе для реализации задания Зk может потребоваться определенный набор ресурсов. Пусть $\{\vec{X}\}$ – множество всех вариантов выбора заданий из очереди, $\vec{X}$ – один из вариантов выбора заданий. Причем $\vec{X} = \{x_1, x_2, ..., x_k, ..., x_p\}$, где $k = \overline{1..p}$, p – количество заданий в очереди, xk – булева переменная равная 1 если задание Зk выбрано в этом варианте и 0 если нет. Ck – приоритет задания Зk. Для описания суммы приоритетов выбранных заданий используем функционал: $F = \sum_{k=1}^{p} C_k x_k \to \max$. Пусть Akg – булева переменная равная 1 если Зk использует ресурс Rg и 0 если нет. Bg – количество ресурсов данного типа Rg. Тогда исходя из условия, что в любой момент времени любой ресурс может быть использована для выполнения задания получаем M ограничений вида: $\sum_{k=1}^{p} A_{kg} x_k \leq B_g$, $g = \overline{1..M}$. Следовательно, нам необходимо найти такую выборку $\vec{X}$ из множества $\{\vec{X}\}$ для которой функционал примет максимальное значение, при выполнении всех ограничений. Мы получили задачу линейного программирования с булевыми переменным. Разрешение очереди запросов, при такой формализации, происходит поэтапно.

Каждый этап состоит из нахождения оптимальной выборки $\vec{X}$, ее обслуживания и изменения функционала и ограничений с учетом изменений в очереди после обслуживания выборки. Эффективные методы реализации обеих процедур приведены в работах [1-3]. Динамическая процедура D осуществляет формированием локальных очередей, обеспечивая максимальную загрузку каждого ресурса. Поскольку любая Grid система потенциально является гетерогенной средой, рассмотрим возможности применения данной процедуры для планирования выполнения заданий в ней. В Grid системе целесообразно иметь два основных уровня это серверы анализа состояний ресурсов и заданий клиентов, которые должны осуществлять сертификацию заданий клиентов, непосредственно подключенных к серверу и сертификацию ресурсов, на которых возможно выполнить задания клиентов. Серверы данного уровня фактически создают поток задач Grid в сети. При этом сеть целесообразно разбить на зоны в пределах, которых, управляющие серверы могут осуществлять мониторинг свободных ресурсов группы кластеров {*Ki*} рис.2 подключенных к данному сертификационному серверу за время *T* не превышающее некоторое допустимое время *Tд*.

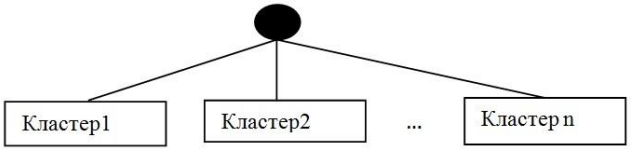

Рис. 2 - Кластеры зоны {K_i} которые могут быть опрошены за время $T \le T_д$.

Серверы обмениваются информацией о наличии свободных ресурсов в своих зонах рис. 3.

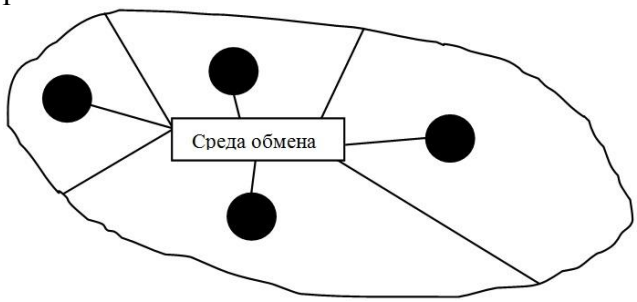

Рис.3 - Взаимодействие управляющих серверов.

Если в зоне сервера нет свободных ресурсов в кластерах, то задания передаются на решение в сертификационный центр с максимальным числом свободных ресурсов в кластерах его зоны {K_i} при этом приоритет этих заданий целесообразно повышать, для того чтобы избежать ситуации, когда некоторые задания могут очень долго оставаться не решенными.

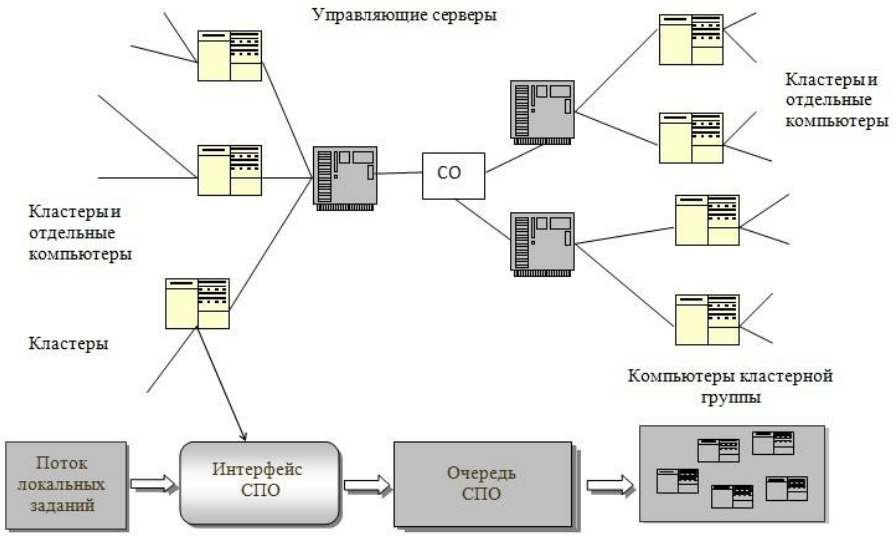

Рис.4 – Архитектура двухуровневой Grid

Очередь задач находится в буфере сервера сертификации, при этом каждая задача харак-

теризуется вектором показателей $(\alpha_1^{3_i}, \alpha_2^{3_i}, ..., \alpha_q^{3_i})$, в качестве которых может объем памяти требуемый для решения задачи, требования к операционной системе ресурса, договорной ценой решения и т.д. Каждый кластер $\{K_i\}$ предоставляет сведения на сертификационный сервер своей зоны данные о наличии свободных ресурсов и их характеристики в том же формате, что и для задач $(\alpha_1^{R_i}, \alpha_2^{R_i}, ..., \alpha_q^{R_i})$. Из буфера сервера формируется пул заданий, который требуется отправить на выполнение в кластеры $\{K_i\}$. На основе сведений о задачах и свободных ресурсах кластеров, в базе данных сертификационного сервера данной зоны создается таблица, столбцам которой соответствуют кластеры зоны $\{K_i\}$, строкам задачи из сформированного пула заданий. На пересечении строки и столбца будем ставить единицу, если из сравнения $(\alpha_1^{3_i}, \alpha_2^{3_i}, ..., \alpha_q^{3_i})$ и $(\alpha_1^{R_i}, \alpha_2^{R_i}, ..., \alpha_q^{R_i})$ следует, что данная задача может быть реализована на имеющемся свободном ресурсе и ноль в противном случае. Далее решая задачу используя одну из предложенных процедур, находим минимальное число кластеров, на котором сформированный пул заданий может быть выполнен, и отправляем его на решения в выделенные кластеры. Далее из буфера выбирается следующий пул заданий, и процедура повторяется, те задания, для которых мы в таблице соответствуют нулевые строки отправляются для решения в соседнюю зону. При формировании пула заданий из буфера можно в первую очередь выбирать те задания, которые прошли через большее число центров управления. Второй уровень образуют множества кластеров $\{K_i\}$ в каждой зоне. Таким образом, двухуровневая система имеет вид рис.4. На первом уровне управляющие серверы центров обмениваются информацией о наличии свободных ресурсов по среде обмена (СО) информацией и если в зоне сервера нет свободных ресурсов, то он отправляет задание на решение на управляющий сервер в зоне которого имеется наибольшее число свободных ресурсов. На втором уровне возникают два конкурирующих потока это поток заданий Grid и поток локальных заданий.

Обобщенная модель верхнего уровня распределения ресурсов в Grid системе на основе процедуры D.

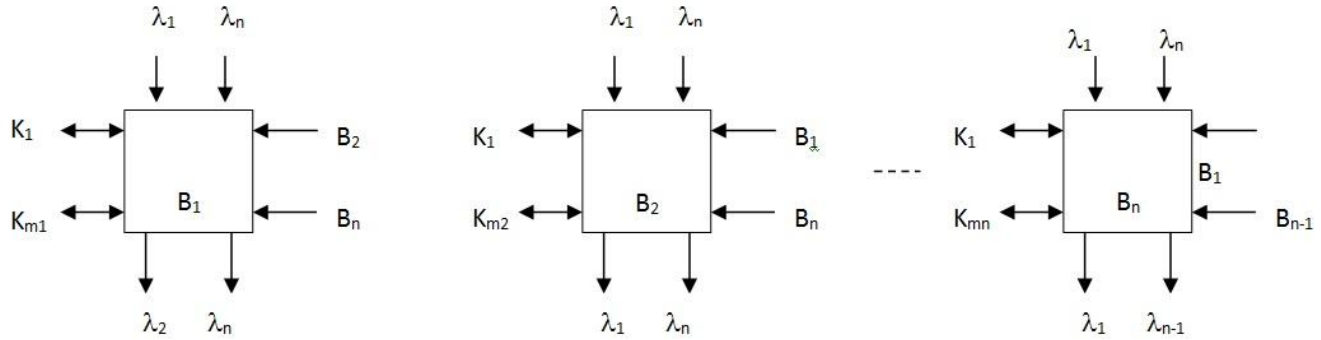

Рис.5 - Модель взаимодействия брокеров верхнего уровня.

Формирование загрузки брокеров $\{B_i\}$ осуществляется на основе таблицы соответствия хранящейся в базе данных (БД) столбцам которой соответствуют типы свободных ресурсов и их количество, а строкам типы задач которые могут быть решены на данных типах ресурсов. Если число заданий в столбце таблицы соответствия превышает число ресурсов соответствующего типа, то мы оставляем в столбцах таблицы соответствия только те задания, которые имеют максимальным суммарный приоритет. Особенностью работы гетерогенных систем является, то что некоторые задания могут быть выполнены только на определенных типах ресурсов, а другие на нескольких типах ресурсов и при произволь-

ном размещении заданий в очереди по таблице соответствия может возникать конкуренция некоторых заданий к одному ресурсу, что может приводить к неравномерности загрузки ресурсов, и существенной задержке выполнения конкурирующих заданий. Задания желательно распределять в очереди на предыдущем шаге таким образом, чтобы подготовить на последующем шаге максимально большее число свободных ресурсов, для выполнения следующего подмножества заданий. Для этого предлагается процедура преобразования общей очереди в локальные на основе непрерывного реализации процедуры *D*. Т.е. на каждом шаге планирования локальных очередей определяется минимальное число брокеров, на которых можно выполнить все задания помещенные диспетчером в пул заданий и в очередь помещаются задания к группе брокеров, которые могут решить весь пул заданий. Причем сначала заполняется очередь к брокеру, на котором будет решаться большее число заданий и если очередь начинает превышать объем буфера, то перемещаем задания на следующий брокер данной группы, который может его выполнить. Если таких брокеров, в группе уже нет, то по таблице соответствия выбираем любой из брокеров, на котором задание данного типа может быть выполнено, т.е. реализуется процедура *D*. Если задание может быть реализовано только на том брокере, к которому локальная очередь превышает размеры буфера, то задание отправляется обратно в пул заданий. Формирование очередей к кластерам $\{K_i\}$ подключенным к брокеру B_i осуществляется точно также на основе непрерывной реализации процедуры *D*. Рассмотрим модель брокера верхнего уровня более подробно. Каждый брокер является одновременно и центром сертификации заданий поступающих в Grid систему, см. рис. 6.

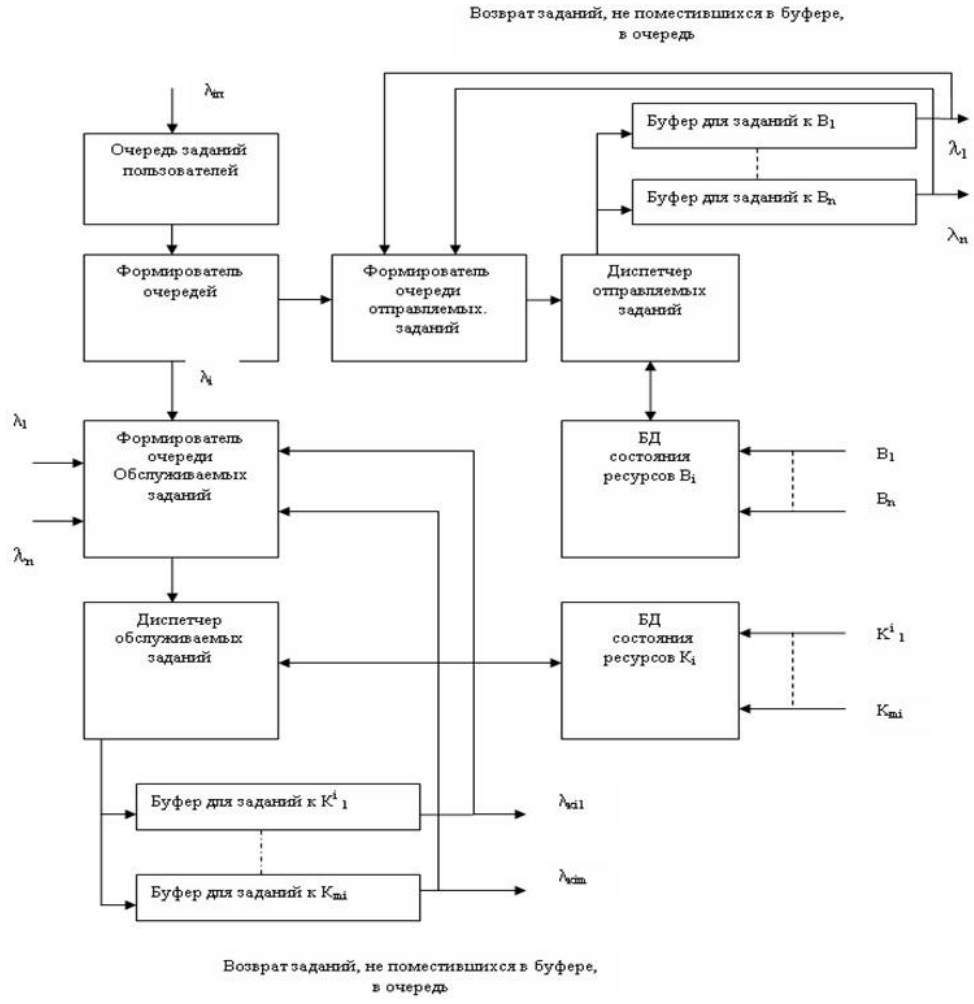

Рис.6 - Модель брокера B_i верхнего уровня.

Задания, поступившие в центр, разбиваются на две очереди: очередь заданий, которая будет решаться кластерами, подключенными к данному сертификационному центру, и оче-

редь заданий которые по характеристикам ресурсов и типов решаемых задач, не могут быть реализованы на ресурсах кластеров подключенных к данному сертифика-ционному центру. Формирование обоих очередей осуществляется однотипно и независимо на основе модели процедуры *D*.

Заключение

Таким образом, в работе предложена унифицированная процедура организации взаимодействия всех ресурсов распределенной системы позволяющая с единых позиций организовывать процесс планирования выполнения заданий в распределенных вычислительных средах.

Литература

1. Минухин С.В. Модели и методы решения задач планирования в распределенных вычислительных системах. / С.В. Минухин– Х.: Щедрая усадьба плюс, 2014. - 323 с.

2. Методы и модели планирования ресурсов в GRID-системах / [В.С.Пономаренко, С.В. Листровой, С.В. Минухин, С.В.Знахур]. – Х.: ИНЖЭК, 2008. – 407 с.

3. Листровой С.В., Подход и модель планирования распределения ресурсов в Grid / С.В.Листровой, С.В.Минухин // Международный научно-технический журнал «Проблемы управления и информатика» - 2012. - №5. - С. 65-82.

4. . Пападимитриу Х. Комбинаторная оптимизация Алгоритмы и сложность / Х. Пападимитриу, К. Стайглиц // М. Мир – 1985. - 509с.

5. Foster, C. Kesselman, S. Tuecke. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. International J. Supercomputer Applications, 15(3), 2001. http://www.globus.org/alliance/publications/papers/anatomy.pdf.

6. Методы и модели планирования ресурсов в GRID-системах. // монография[Пономаренко В.С., Листровой С.В., Минухин С.В., Знахур С.В.]. - Харьков: ИД «ИНЖЭК», 2008. - 408с.

7. Листровой С.В., Общий подход к решению задач оптимизации в распределенных вычислительных системах и теории построения интеллектуальных систем / С.В. Листровой, С.В Минухин – Международный научно-технический журнал «Проблемы управления и информатика» 2010. - №2. - С. 65-82.

8. S.V. Listrovoy S.V. Minukhin «General Approach to Solving Optimization Problems in Distributed Cjmputing Sysntems and Theory of Intelligence Systems Construction» // Journal of automation and information sciences Volume 42, Namber 3, 2010,P. 30-46

9. Listrovoy S.V., Solution method on the basis of rank approach for integer linear problems with boolean variables / S.V. Listrovoy D.Yu. Golubnichiy, E.S. Listrovaya // Engineering Simulation. – 1999. – vol.16. – P. 707–725.

10. Listrovoy S.V., Parallel algorithms of calculation process optimization for the boolean programming problems / S.V.Listrovoy, V.F. Tretjak, A.S. Listrovaya / Engineering Simulation. – 1999. – vol.16. – P. 569–579.

ВІДОМОСТІ ПРО АВТОРІВ

Лістровий Сергій Володимирович,
професор кафедри «Спеціалізовані комп'ютерні системи» Українського державного університету залізничного транспорту
Пл. Фейербаха, 7, Харків, Україна, 61050
Тел.:+38 050 935 50 42
E-mail: om1@yandex.ru

Курцев Максим Сергійович,
аспірант кафедри «Спеціалізовані комп'ютерні системи» Українського державного університету залі-зничного транспорту
Пл. Фейербаха, 7, Харків, Україна, 61050
Тел.: +38 050 302 99 12
E-mail: kurtsev_m@ukr.net