УДК 004.75, 519.172

ОБЗОР ТЕХНОЛОГИЙ РАСПРЕДЕЛЕННЫХ ВЫЧИСЛЕНИЙ

А.В. Шамакина

В данной работе представлен обзор технологий распределенных вычислений с точки зрения организации планирования заданий в грид и облачных средах. Рассмотрена архитектура платформы UNICORE, ориентированной на обеспечение прозрачного безопасного доступа к ресурсам распределенной вычислительной среды. Приведена общая классификация алгоритмов планирования в распределенных вычислительных средах, а также классификация алгоритмов планирования для заданий, имеющих зависимости между задачами.

Ключевые слова: распределенные вычисления, проблемно-ориентированные распределенные вычислительные среды, грид, UNICORE, алгоритмы планирования ресурсов, алгоритмы кластеризации, ориентированный ациклический граф, потоки работ.

Введение

Развитие технологий распределенных вычислений в конце 1990-х годов позволило объединить географически-распределенные по всему миру гетерогенные ресурсы. Появились технические возможности для решения масштабных задач в области науки, техники и коммерции на территориально-распределенных ресурсах, принадлежащих разным владельцам. Исследования данной тематики привело к возникновению концепции грид вычислений (grid computing) [1, 2–4], и затем — к новой концепции облачных вычислений (cloud computing) [5–8]. Для раскрытия всех потенциальных возможностей использования распределенных вычислительных ресурсов принципиально важно наличие результативных и эффективных алгоритмов планирования, используемых менеджерами ресурсов.

Управление ресурсами в традиционных гомогенных многопроцессорных системах (вычислительных кластерах) — хорошо изученный и проработанный вопрос. Существует большое количество менеджеров ресурсов для подобных систем [9]. Менеджеры ресурсов включаются в пакетные планировщики, в инструментарий для управления очередями заданий, в операционные системы. Эти менеджеры являются локальными, имеют полный контроль над ресурсами и реализуют механизмы и политики для эффективного использования данных изолированных ресурсов. Алгоритмы планирования для изолированных гомогенных многопроцессорных систем не могут также хорошо работать в распределенных вычислительных средах [4].

Главной задачей, которую решают технологии распределенных вычислений, является обеспечение доступа к глобально распределенным ресурсам с помощью специального инструментария. Сложность управления глобальными ресурсами заключается в том, что запуск, выполнение работы и доступ к необходимым данным могут производиться на различных компьютерах. Глобальные распределенные вычислительные сети формируются из автономных ресурсов, конфигурация которых динамически изменяется. Кроме того, распределенные ресурсы могут принадлежать различным административным доменам, поэтому возникает проблема их администрирования, заключающаяся в согласовании различных политик. Еще одной немаловажной проблемой является гетерогенность ресурсов. Ранние работы [6, 10–12] в области управления ресурсами в распределенных вычислительных средах, фокусирующиеся на гетерогенности ресурсов, привели к созданию

стандартных протоколов управления ресурсами и механизмов описания требований заданий к ресурсам. Однако практика показала, что эффективные методы и алгоритмы планирования для однородных изолированных многопроцессорных систем плохо адаптируются для распределенных гетерогенных систем [13]. Управление ресурсами в неоднородных распределенных вычислительных средах требует принципиально новых моделей вычислений и управления ресурсами.

В настоящее время перспективным является направление, связанное с применением распределенных вычислительных технологий для решения ресурсоемких научных задач в разных предметных областях: медицине, инженерном проектировании, нанотехнологиях, прогнозировании климата и др. Вычислительное задание в подобных предметных областях во многих случаях имеют потоковую структуру и могут быть описаны с помощью модели потока работ (workflow) [14], в соответствии с которой задание представляется в виде ориентированного ациклического графа, узлами которого являются задачи, являющиеся составными частями задания, а дуги соответствуют потокам данных, передаваемых между отдельными задачами. При этом набор задач, из которых строятся задания, является конечным и предопределенным. Проблемно-ориентированная специфика потоков работ в подобных сложных приложениях выражается в том, что в подавляющем большинстве случаев, еще до выполнения задания, для каждой задачи могут быть получены оценки таких качественных характеристик, как время выполнения задачи на одном процессорном ядре, пределы масштабируемости и объем генерируемых данных. Использование подобных знаний о специфике задач в конкретной проблемно-ориентированной области может существенно улучшить эффективность методов управления вычислительными ресурсами. В настоящее время известно несколько программных систем, ориентированных на управление сложными приложениями с потоковой структурой в распределенных вычислительных средах. В качестве примера можно перечислить такие инструменты как Condor DAGMan [15], CoG [16], Pegasus [17], GridFlow [18] и ASKALON [19]. Существуют также алгоритмы планирования, использующие знания о проблемно-ориентированной специфике задач, составляющих вычислительное задание, такие как алгоритм Кима и Брауна [20], алгоритм DSC [21]. Однако данный класс алгоритмов применим для задач, выполняющихся на одном процессорном ядре некоторой многопроцессорной системы. В соответствие с этим актуальной является задача разработки методов и алгоритмов управления ресурсами в проблемно-ориентированных распределенных вычислительных средах, учитывающих специфику предметной области, масштабируемость отдельных задач в задании и использующих возможность параллельного выполнения независимых задач.

Статья организована следующим образом. В разделе 1 рассматриваются основные технологии распределенных вычислений: грид-вычисления и облачные вычисления. Раздел 2 посвящен обзору алгоритмов планирования ресурсов в распределенных вычислительных средах. В заключении изложены выводы, основанные на данном обзоре.

1. Технологии распределенных вычислений

1.1. Грид-вычисления

Вычислительный грид является программно-аппаратной инфраструктурой, которая обеспечивает надежный и прозрачный доступ к высокопроизводительным вычислительным ресурсам [4]. Грид представляет общую среду для развертывания инфраструктуры,

ориентированной на сервисы, поддерживающей создание и совместное использование ресурсов распределенных организаций. Под ресурсами понимаются аппаратное обеспечение, инструментарий, программное обеспечение и данные, а также сервисы, подключенные посредством промежуточного слоя программного обеспечения и обеспечивающие безопасность, мониторинг, управление ресурсами и др.

При рассмотрении проблемы планирования в грид-средах для повышения уровня абстракции часто игнорируют такие компоненты инфраструктуры, как аутентификация, авторизация, обнаружение ресурсов и контроль доступа. В работе [22] приводится следующее адаптированное определение: «Грид — это тип параллельных и распределенных систем, которые обеспечивают совместное использование, выбор и динамическую агрегацию географически-распределенных автономных и гетерогенных ресурсов в зависимости от их доступности, характеристик, производительности, стоимости и требований качества обслуживания, предъявляемых пользователями».

С точки зрения функциональности можно выделить логическую архитектуру подсистемы планирования заданий в грид. В работе [13] предложена общая архитектура данной подсистемы. Процесс планирования в распределенных вычислительных сетях может быть представлен тремя этапами: 1) обнаружение ресурсов и их фильтрация, 2) поиск подходящих ресурсов и планирование в соответствии с определенными целями, 3) выполнение задания [23]. На рис. 1 представлена модель системы планирования в распределенных вычислительных средах, в которой функциональные компоненты связывают два типа потоков данных: прерывистая линия определяет поток ресурсов/поток информации о приложении, прямая линия — поток заданий/поток команд планирования заданий.

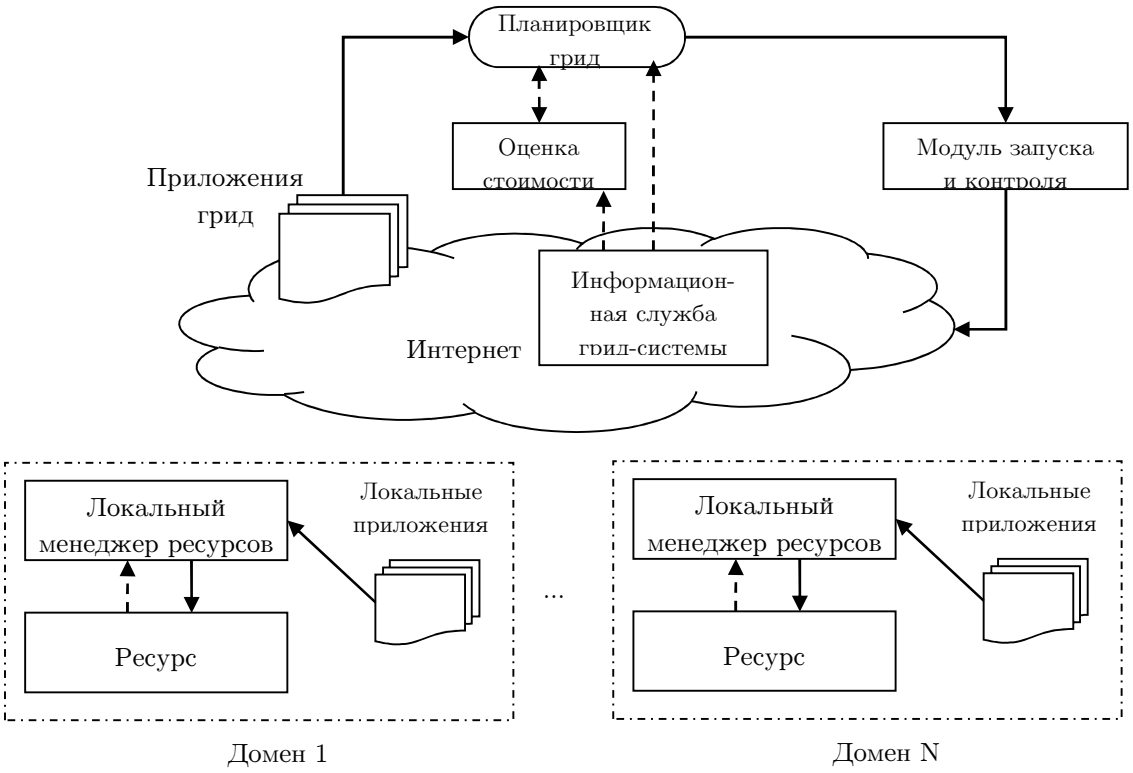

Рис. 1. Логическая архитектура системы планирования в грид

Основная работа *планировщика грид* (GS — Grid Scheduler) заключается в том, что он принимает заявки на выполнение некоторых приложений пользователей, производит поиск подходящих ресурсов в соответствии с полученными от *информационного сервиса*

грид-среды (GIS — Grid Information Service) данными и создает отображение приложений на ресурсы, основанное на целевых функциях и предсказании характеристик ресурсов. В отличие от планировщиков традиционных параллельных и распределенных систем, грид-планировщики обычно не контролируют грид-ресурсы напрямую, а работают как брокеры или агенты [24]. Грид-планировщики не всегда находятся в одном домене с ресурсами, которые им доступны. На рис. 1 показан только один грид-планировщик, однако в действительности может быть развернуто несколько подобных планировщиков для формирования различных структур системы планирования (централизованной, иерархической и децентрализованной [25]) для решения проблем производительности и масштабируемости. Хотя планировщик уровня грид (иначе его называют метапланировщиком [26]) не является необходимым компонентом в инфраструктуре грид-среды (он не включен в Globus Toolkit [27]), но он имеет решающее значение для использования в грид-средах, которые имеют тенденцию быстро расширяться, добавляя в свой состав все новые и новые ресурсы от суперкомпьютеров до настольных компьютеров.

Информация о состоянии имеющихся ресурсов необходима грид-планировщику для составления приемлемого расписания работ, особенно в условиях гетерогенного и динамичного характера грид. Расписание часто представляется в виде диаграммы Ганта, горизонтальная ось которой представляет собой время, вертикальная ось — ресурсы в грид. На каждом из ресурсов прямоугольниками обозначаются задачи, ширина прямоугольника показывает длительность задачи. Роль информационного сервиса грид-среды заключается в предоставлении информации о состоянии имеющихся ресурсов планировщикам грид. Информационный сервис ответственен за сбор и предсказание информации о состоянии ресурса. Информационный сервис грид-среды также может отвечать на запросы, предоставляя информацию о ресурсе, или передавать информацию подписчикам. Примерами информационных сервисов в грид являются Globus Monitoring и Discovery System (MDS) [28].

Для составления приемлемого расписания помимо информации о ресурсе необходимо знать свойства приложений (приблизительное количество инструкций, требования к памяти и хранению, зависимости подзадач в задании) и производительность ресурсов для различных видов приложений. Информация о свойствах приложения может быть получена с помощью *профилирования* (AP — Application profiling), а измерение производительности ресурса для данного типа задания — с помощью *компоненты тестирования* (AB — Analogical Benchmarking) [29, 30]. На основе информации, полученной при профилировании приложений, и информации от компоненты тестирования, а также используемой модели производительности [4], производится оценка стоимости планирования узлов-кандидатов на выполнение приложения, из которых планировщик выбирает те, которые оптимизируют целевые функции.

Модуль запуска и контроля (LM — Launching and Monitoring) формирует окончательное расписание, предоставляет приложениям соответствующие ресурсы, поставляет входные данные и исполняемые файлы, если это необходимо, и выполняет мониторинг исполнения приложений. Модуль запуска и контроля иногда называют *компановщиком* [31]. Примером модуля запуска и контроля является Globus GRAM (Grid Resource Allocation and Management) [32].

Локальный менеджер ресурсов (LRM — Local Resource Manager) несет основную ответственность за составление локального расписания внутри домена, где присутствуют

задания не только от внешних пользователей грид, но и выполняются задания локальных пользователей домена, а также предоставляет отчетную информацию для информационного сервиса грид-среды. Внутри домена могут работать сразу несколько локальных планировщиков каждый со своей локальной политикой управления ресурсами. В качестве примеров подобных локальных планировщиков можно привести OpenPBS [33] и Condor [34]. LRM собирает информацию о локальных ресурсах с помощью таких инструментальных средств, как Network Weather Service [35], Hawkeye [34] и Ganglia [36], и формирует отчет с информацией о состоянии ресурсов для информационного сервиса.

1.2. Облачные вычисления

Концепция предоставления вычислительных ресурсов, названная облачными вычислениями (cloud computing), сформировалась в 2008 г. В 2011 г. Национальный институт стандартов и технологий США (The National Institute of Standards and Technology, NIST) опубликовал 16-е, окончательное определение данного понятия. Согласно NIST, *облачные вычисления* — это модель обеспечения удобного повсеместного сетевого доступа по требованию к совместно используемому пулу конфигурируемых вычислительных ресурсов, которые можно быстро предоставить и внедрить с минимумом административных усилий или взаимодействия с сервис-провайдером [37].

NIST зафиксированы следующие пять обязательных характеристик облачных вычислений:

- самообслуживание по требованию;
- универсальный доступ по сети;
- объединение ресурсов;
- эластичность;
- учет потребления.

Существует три основных модели обслуживания облачных вычислений:

- программное обеспечение как услуга (SaaS — Software-as-a-Service),
- платформа как услуга (PaaS — Platform-as-a-Service),
- инфраструктура как услуга (IaaS — Infrastructure-as-a-Service),

а также появляются дополнительные модели:

- аппаратное обеспечение как услуга (HaaS — Hardware as a Service),
- безопасность как сервис (SECaaS — Security as a Service),
- данные как услуга (DaaS — Data as a Service).

Модель обслуживания SaaS предоставляется возможность использования прикладного программного обеспечения провайдера, работающего в облачной инфраструктуре и доступного из различных клиентских устройств или посредством тонкого клиента, например, из браузера (например, почта Gmail) или интерфейса программы.

Модель обслуживания PaaS предоставляет потребителю возможность использования облачной инфраструктуры для размещения базового программного обеспечения для последующего размещения на нем новых или существующих приложений. В состав платформ входят инструментальные средства создания, тестирования и выполнения прикладного программного обеспечения, предоставляемые облачным провайдером. Примерами подобных платформ являются Google App Engine [8] и Windows Azure [5].

Модель обслуживания IaaS предоставляет возможность использования облачной инфраструктуры для самостоятельного управления ресурсами обработки, хранения, сетей и

другими фундаментальными вычислительными ресурсами. Примером облаков, предоставляющих инфраструктуру как услугу, является Nimbus [6].

Эталонная архитектура облачных вычислений NIST содержит пять главных действующих субъектов – *актеров (actors)*. Каждый актер выступает в некоторой *роли (role)* и выполняет *действия (activities)* и *функции (functions)*. Эталонная архитектура представляется в виде последовательности диаграмм с увеличивающимся уровнем детализации. Виды актеров облачных вычислений представлены в таблице.

Таблица

Виды узлов логического плана решения задач

Актер	**Определение**
Облачный потребитель Cloud Consumer	Лицо или организация, поддерживающая бизнес-отношения и использующая услуги *Облачных провайдеров.*
Облачный провайдер Cloud Provider	Лицо, организация или сущность, отвечающая за доступность облачной услуги для *Облачных потребителей.*
Облачный аудитор Cloud Auditor	Участник, который может выполнять независимую оценку облачных услуг, обслуживания информационных систем, производительности и безопасности реализации облака.
Облачный брокер Cloud Broker	Сущность, управляющая использованием, производительностью и предоставлением облачных услуг, а также устанавливающая отношения между *Облачными провайдерами* и *Облачными потребителями.*
Облачный оператор связи Cloud Carrier	Посредник, предоставляющий услуги подключения и транспорт (услуги связи) для доставки облачных услуг от *Облачных провайдеров* к *облачным потребителям.*

Обобщенная облачная среда содержит три концептуальных уровня.

– *Уровень сервиса (Service Layer)* определяет базовые сервисы, предоставляемые облачным провайдером.
– *Уровень абстракции и контроля ресурсов (Resource Abstraction and Control Level)* назначает/предоставляет элементы программного обеспечения, такие как гипервизор, виртуальные хранилища данных и поддерживающие программные компоненты, используемые для реализации облачной инфраструктуры, поверх которой может быть определен/установлен облачный сервис. Кроме того, данный уровень назначает/предоставляет ассоциированные функциональные модули, которые управляют абстрагированными ресурсами для обеспечения эффективного, безопасного и надежного использования.
– *Уровень физических ресурсов (Physical Resource Level)* включает все физические ресурсы: компьютерное оборудование и инженерную инфраструктуру.

На рис. 2 представлена концептуальная диаграмма эталонной архитектуры облачных вычислений согласно NIST.

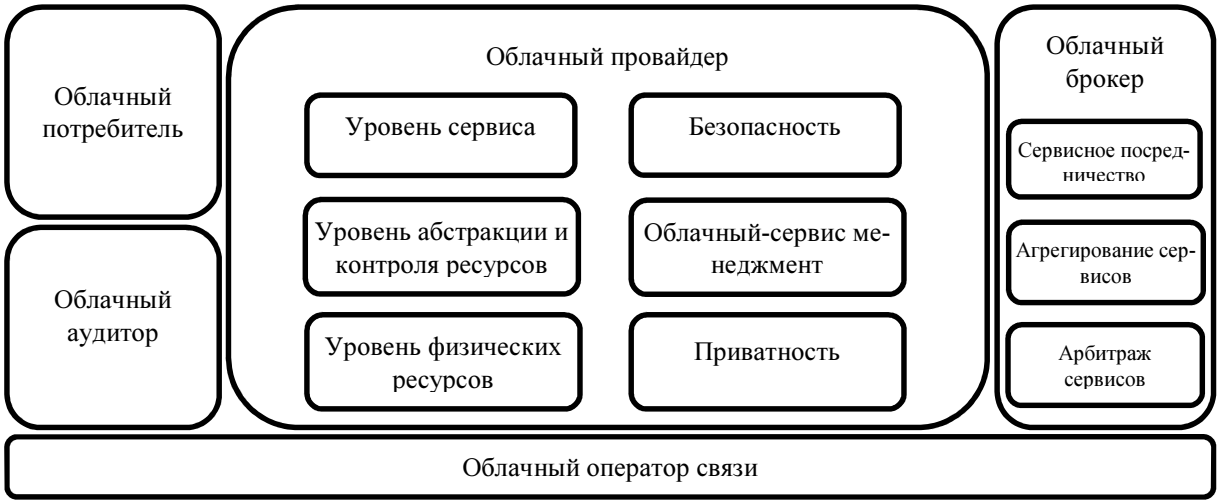

Рис. 2. Концептуальная диаграмма эталонной архитектуры облачных вычислений

1.3. Платформа UNICORE

Проект UNICORE (Uniform Interface to Computing Resources — единый интерфейс к вычислительным ресурсам) появился в 1997 году, и к настоящему моменту представляет собой комплексное решение, ориентированное на обеспечение прозрачного безопасного доступа к ресурсам распределенной вычислительной среды [38].

Архитектура UNICORE 6 [39] формируется из пользовательского, сервисного и системного слоев (рис. 3).

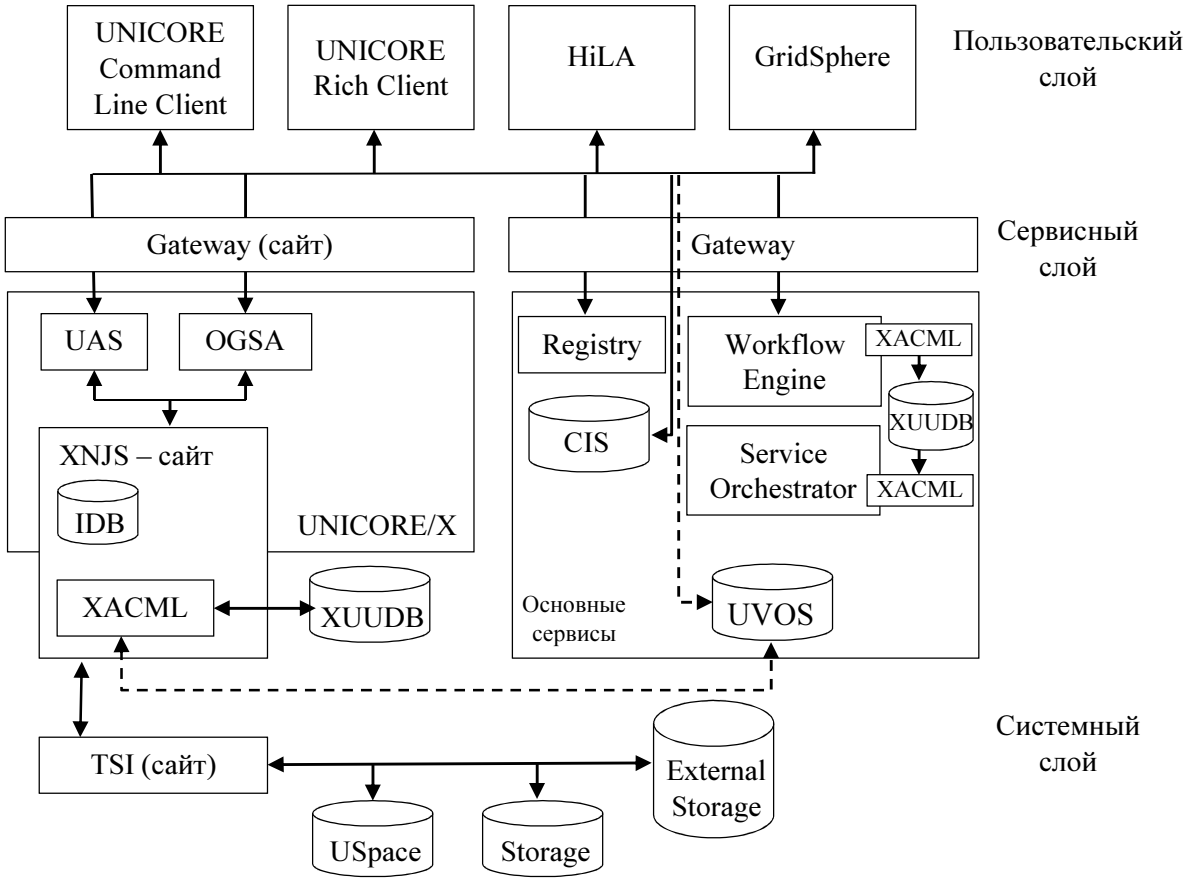

Рис. 3. Архитектура UNICORE 6

Верхним слоем в архитектуре является *пользовательский слой*. В нем располагаются различные клиенты, обеспечивающие взаимодействие пользователей с распределенной вычислительной средой:

На пользовательском уровне доступ организуется с использованием графического интерфейса и интерфейса командой строки. Задания могут быть запущены на любой из платформ UNICORE в распределенной вычислительной среде. Пользователь может осуществлять мониторинг и управление запущенными заданиями, используя часть интерфейса, называемую *монитором заданий*. Сейчас UNICORE позволяет использовать различные клиенты по требованию: Command Line Client (UCC), Eclipse-based UNICORE Rich Client (URC), High Level API for Grid Applications (HiLA) и Portal Clients (например, GridSphere).

Средний слой или слой сервисов включает все сервисы и компоненты сервис-ориентированной архитектуры UNICORE, основанные на WS-RF 1.2 и SOAP, а также на стандартах WS-I.

Компонент *Gateway* является точкой входа на сайт UNICORE и выполняет аутентификацию всех входящих запросов.

Сервер *UNICORE/X* является главным компонентом сайта UNICORE 6. Посредством WSRF сервер UNICORE/X обеспечивает доступ к ресурсам хранилища, сервису передачи файлов и сервисам выполнения заданиями и мониторинга.

Набор собственных интерфейсов веб-сервисов называется *UNICORE Atomic Services (UAS)*. UAS предоставляет набор базовых услуг для сервисов более высокого уровня, клиентов и пользователей.

Компонент, отвечающий за механизм выполнения заданий в UNICORE, называется *XNJS*. Он обеспечивает ресурсы хранения, сервисы передачи файла и сервисы управления заданием.

База данных *IDB (Incarnation Data Base)* используется для отображения описания абстрактного задания на языке JSDL (Job submission description language) в описание конкретного задания для определенного ресурса. Информация о доступных приложениях и характеристиках ресурсов должна быть определена в базе данных IDB.

Компонент *XUUDB* представляет собой веб-сервис и базу данных для отображения сертификатов X.509 на логины фактических пользователей и их роли. Управление доступом основывается на политике XACML.

Компонент *UNICORE VO Service (UVOS)* используется в качестве альтернативы XUUDB, а также для авторизации пользователей с помощью стандарта SAML.

Компонент *Registry* предназначен для регистрации сервисов, доступных в UNICORE. Единая сервисная регистрация необходима для построения распределенной инфраструктуры UNICORE и управления ей.

Common Information Service (CIS) является информационным сервисом UNICORE. CIS собирает статическую и динамическую информацию от всех компонентов XNJS, которые с ним связаны.

Компонент *Workflow Engine* обеспечивает выполнение потока работ. Поток работ создается/запускается с помощью графического интерфейса UNICORE Rich Client или из командной строки Command Line Client.

Компонент *Service Orchestrator* отвечает за выполнение отдельных задач в потоке работ, обработку выполнения задач и мониторинг в распределенной вычислительной

среде. В Service Orchestrator реализованы различные стратегии планирования ресурсов. Существует возможность подключения пользовательских стратегий.

В основании архитектуры платформы UNICORE находится *системный слой*. Компонент *Target System Interface (TSI)* обеспечивает взаимодействие между UNICORE и отдельным ресурсом распределенной вычислительной сети. TSI обеспечивает трансляцию команд, поступающих из распределенной вычислительной среды, в команды для локальной системы.

Целевая система (Target System, TS) — это совокупность программного и аппаратного обеспечения, доступного в распределенной вычислительной среде. Описание приложений и аппаратных ресурсов хранится в файле simpleidb директории TSI.

USpace представляет собой директорию для хранения заданий пользователей. Существует отдельные директории для каждого задания, в которых XNJS и TSI хранят исходные данные, стандартный вывод и стандартный поток ошибок. Для передачи данных между сайтами, например, для передачи данных с/на внешние хранилища, используется протокол GridFTP.

External Storage служит для передачи данных между сайтами. Для работы с внешними хранилищами компонент External Storage использует протокол GridFTP.

UNICORE 6 позволяет использовать *потоки работ* и поддерживает следующие возможности:

- представление задания в виде ориентированного графа;
- использование переменных в потоках работ;
- использование циклов и управляющих конструкций:
  - while, for-each, if-else;
- использование в потоках работ следующих условий:
  - код выхода;
  - существование файла;
  - размер файла.

Основным достоинством использования платформы UNICORE 6 для разработки распределенных вычислительных систем является наличие большого количества различных клиентов, обеспечивающих взаимодействие пользователя с ресурсами распределенной вычислительной сети, а также развитых средств обеспечения безопасности при разработке приложений в распределенных вычислительных средах.

2. Обзор алгоритмов планирования ресурсами в распределенных вычислительных средах

2.1. Общая классификация алгоритмов планирования ресурсами

Существующие алгоритмы планирования ресурсов в распределенных вычислительных средах могут быть классифицированы по разным признакам: с точки зрения архитектуры компонентов, участвующих в планировании; используемых политик; целевых функций; моделей приложений; ограничений качества обслуживания (QoS — Quality of Service); стратегий, применяемых для ресурсов с динамическим поведением и др. [9].

На рис. 4 представлена иерархическая классификация алгоритмов планирования общего назначения для параллельных и распределенных вычислительных систем [40].

На верхнем уровне иерархической классификации выделяют *статические и динамические алгоритмы* планирования ресурсов. Статическое планирование и расчет стоимостной оценки вычислений осуществляется до начала выполнения задания, когда информация относительно всех ресурсов в распределенных вычислительных средах и всех задачах задания уже доступна [41–43]. Одно из главных преимуществ статической модели — это простота реализации планировщика. Однако, стоимостная оценка, основанная на статической информации, не адаптивна к ситуациям, когда один из вычислительных узлов выходит из строя, становится изолированным для системы из-за сетевого отказа или из-за высокой загрузки на систему время отклика становится более длительным, чем ожидалось. Для решения проблемы используют вспомогательные механизмы, такие как механизм перепланирования [31].

Динамическое планирование обычно применяется, когда трудно оценить вычислительную стоимость приложений, поступающих на выполнение динамически в режиме online [44–47]. Примером использования динамического планирования является управление очередью заданий в системах метавычислений Condor [48] и Legion [49]. Динамическое планирование задач включает в себя два важных компонента: оценка состояния системы и принятие решения о связывании задачи из очереди с выбранным ресурсом [50]. Для сохранения оптимального состояния вычислительной системы используется балансировка загрузки всех ее ресурсов. Преимущество динамической балансировки загрузки над статической состоит в том, что система не обязана знать о поведении приложения во время его выполнения до его запуска. Особенно это подход полезен в системе, где основной целью является максимизация утилизации ресурса, а не минимизация времени выполнения отдельных заданий [51].

Динамические алгоритмы планирования, описанные в работах [26] и [52], рассматривают случай резервирования ресурса, которое популярно в распределенных вычислениях. Резервирование ресурсов используется для получения некоторой степени уверенности в производительности ресурса. Алгоритмы в этих двух работах стремятся минимизировать время исполнения входящих заданий, которые состоят из набора задач.

В динамических сценариях планирования ответственность за принятие глобальных решений планирования ресурсов может лежать на одном централизованном планировщике или нескольких распределенных планировщиках. *Централизованная стратегия* имеет преимущество, заключающееся в простоте реализации, но она плохо масштабируется, не отказоустойчива и часто становится узким местом для производительности системы. Например, в работе [53] предлагается централизованный метапланировщик, который использует алгоритм обратного заполнения (Backfill) для планирования параллельных заданий на сложных гетерогенных сайтах. Аналогично, в работе [54] представлен полностью *децентрализованный, динамический и инициируемый отправителем алгоритм* планирования и балансировки загрузки для распределенных вычислительных сред. Главное свойство этого алгоритма заключается в том, что он использует интеллектуальную стратегию поиска узлов-партнеров, на которые могут быть перенесены задачи.

В том случае, когда вся информация относительно состояния ресурсов и заданий известна, *оптимальная привязка* заданий к ресурсам может быть сделана на основании некоторой целевой функции, такой как минимизация времени выполнения заданий и максимальная утилизация ресурсов.

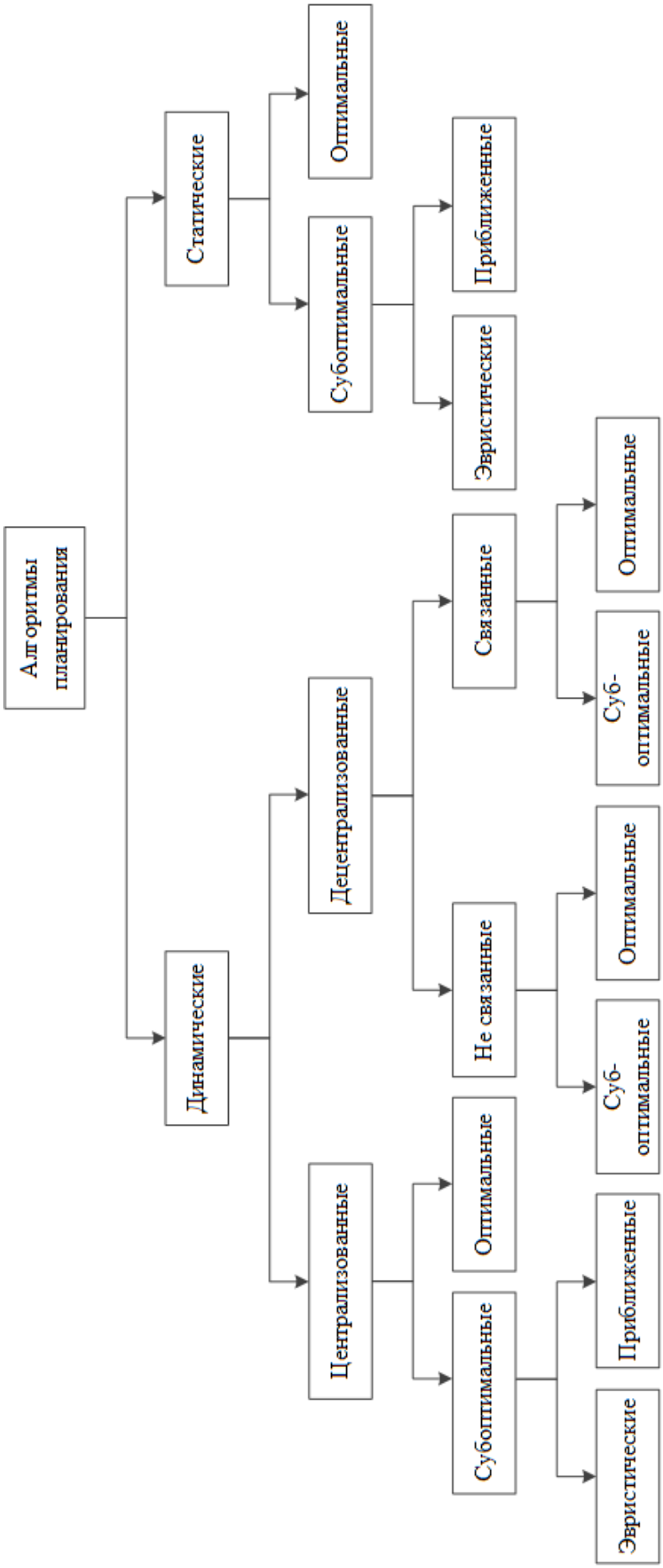

Рис. 4. Иерархическая классификация алгоритмов планирования

Однако, доказать оптимальность алгоритма или сделать некоторые разумные предположения об оптимальности представляется невозможным из-за того, что общая задача планирования является NP-полной [55]. Текущие исследования пытаются найти *субоптимальные решения*, которые могут быть далее разделены на следующие основные категории. *Приближенные алгоритмы* используют формальные вычислительные модели, вместо поиска всего пространства решений и выбора из него оптимального решения. Данные алгоритмы осуществляют поиск приемлемого решения, близкого к оптимальному. В случае, когда существует метрика пригодная для оценки решения, этот метод может использоваться для уменьшения времени, потраченного на поиск приемлемого расписания.

Эвристики — представляют собой класс алгоритмов, которые делают наиболее реалистичные предположения об априорном знании относительно характеристик загрузки системы и выполнения. Оценка этого вида решения обычно основана на экспериментах в реальном мире или на моделировании. Наиболее популярными в настоящее время являются экономические подходы [56–62] и эвристики, основанные на природных явлениях: генетический алгоритм (GA — Genetic Algorithm) [41, 63–65], моделируемый отжиг (SA — Simulated Annealing) [41, 62, 66,], запрещенный поиск (TS — Tabu Search) [41] и комбинированная эвристика [67].

Распределенные алгоритмы планирования можно разделить на *связанные или несвязанные*, в зависимости от того, как работают узлы, использующиеся при планировании заданий, совместно или независимо (несовместно). В несовместном случае локальные планировщики действуют как автономные сущности и принимают решения, с учетом их собственных целевых функций. В совместном случае каждый планировщик в распределенной вычислительной среде несет ответственность за выполнение его собственной части задачи планирования, но при этом все планировщики работают с одной общей целью в масштабе всей системы [68].

2.2. Классификация на основе зависимости задач в задании

При рассмотрении отношений между задачами в задании обычно используют следующую дихотомию: есть ли между ними зависимость или же задачи независимы [9]. Зависимость означает, что у задач есть приоритеты, то есть, задача не может запуститься до того, как были выполнены все ее родителя. Зависимость оказывает решающее влияние на разработку алгоритмов планирования. На рис. 5 представлена классификация алгоритмов планирования, основанная на наличии/отсутствии связей между задачами.

Главная *стратегия планирования независимых задач* заключается в назначении независимых задач на ресурсы в зависимости от их загрузки с целью обеспечения высокой пропускной способности вычислительной системы. Примерами статических алгоритмов с оценкой производительности являются: алгоритм минимального времени выполнения (MET — Minimum Execution Time), алгоритм минимального времени завершения (MCT — Minimum Completion Time), эвристики Min-Min [69, 70] и Max-min, Suffrage [12] и XSuffrage [42]. Данные алгоритмы обычно используются для планирования заданий, которые состоят из множества независимых задач с большими модулями и интенсивными вычислениями. Muthuvelu и др. [47] предлагают динамический алгоритм планирования сгруппированных задач, который позволяет уменьшить среднее значение издержек при планировании и запуске задания и увеличить использование ресурса.

В гетерогенных средах на производительность вышеупомянутых алгоритмов также влияет уровень неоднородности задач и ресурсов. Исследование в работе [12] показало,

что не существует ни одного алгоритма, который бы имел преимущества во всех аспектах. Для достижения максимальной высокой производительности в вычислительном грид планировщик должен уметь адаптироваться под неоднородные приложения/ресурсы.

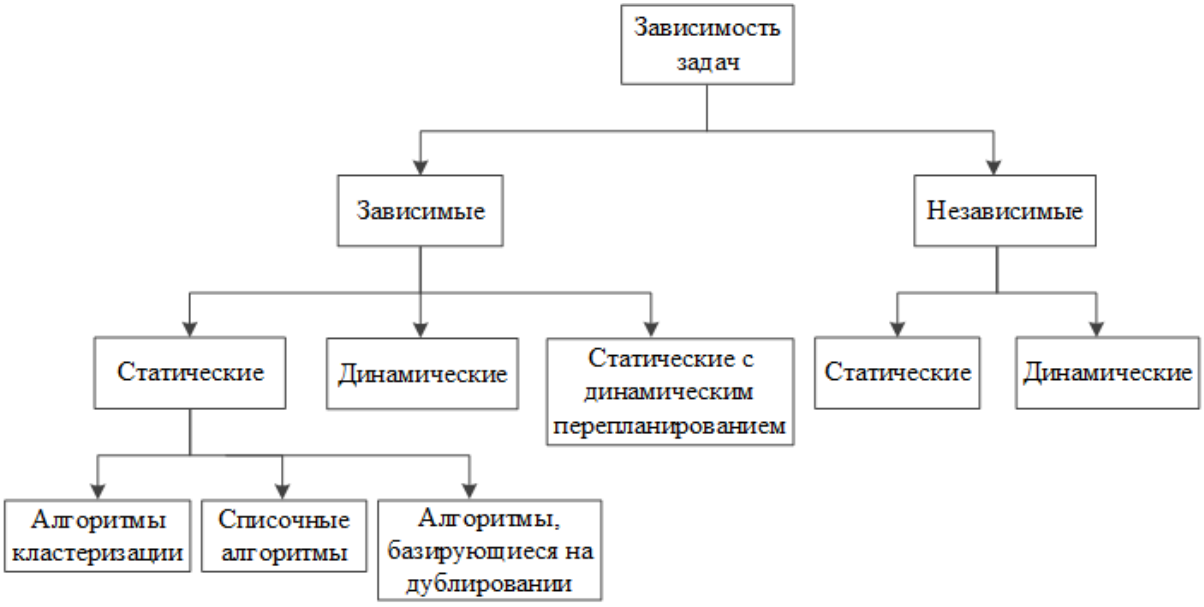

Рис. 5. Классификация зависимых и независимых задач в алгоритмах планирования ресурсами

Алгоритмы, представленные выше, предсказывают производительность для отображения задачи на ресурс. В [71] и [72] предложены два алгоритма, которые не используют оценку производительности, но основываются на идее дублирования, которое выполнимо в распределенной вычислительной среде, где вычислительные ресурсы присутствуют в достаточном количестве, но непостоянны.

В случае *планирования задач, имеющих зависимости*, задание обычно представляется в виде ориентированного ациклического графа, в котором каждая вершина представляет собой задачу, ориентированное ребро обозначает порядок приоритета между двумя вершинами. В некоторых случаях к вершинам и ребрам могут быть добавлены веса, показывающие вычислительную стоимость и коммуникационную стоимость соответственно.

Важнейшей проблемой при планировании заданий с потоковой структурой является нахождение компромисса между использованием максимального параллелизма задач в задании и минимизации коммуникационных задержек. Для решения данной проблемы (также известной как проблема максимина [55]) в гетерогенных вычислительных системах ранее были предложены три вида эвристических алгоритмов: эвристики списка [73, 74]; алгоритмы, базирующиеся на дублировании; алгоритмы кластеризации (рис. 5).

Планирование списком — это класс эвристик планирования, в котором задачам присваиваются приоритеты, задачи помещаются в список, упорядоченный по мере уменьшения величины приоритета. Решение о выборе задачи из списка для ее выполнения осуществляется на основе приоритета. Сначала осуществляется привязка к ресурсам задач с высоким приоритетом [55]. Классическими примерами эвристик списка являются HEFT (Heterogeneous Earliest-Finish-Time) [75] и FCP (Fast Critical Path) [73].

Критической проблемой в эвристике списка для DAG является вычисление ранга узла. Собственные значения, используемые для принятия решения об упорядочивании, могут быть усредненными значениями (как в оригинальном алгоритме HEFT в [75]), средним значением [11], худшем значением, оптимальным значением и так далее. Zhao и др. [76] показали, что различные варианты могут влиять на производительность эвристики списка, которые существенны для HEFT (время исполнения может изменяться на 47,2 % для определенного графа). Sakellariou и др. [77] предложил гибридный алгоритм, который менее чувствителен к разным подходам для ранжирования узлов.

Предыдущие алгоритмы планирования для ориентированного ациклического графа были рассмотрены применительно к использованию в распределенных вычислительных средах. Списочный алгоритм планирования, предложенный в работе [43], подобен алгоритму HEFT, но модифицирует его для вычисления уровня задачи в графе и учитывает входящую коммуникационную стоимость его родительских задач. В работе [78] Ma и др. предлагают новый списочный алгоритм для распределенных вычислительных сред под названием «расширенный динамический предельный путь» (xDCP — Extended Dynamic Critical Path), который представляет собой модификацию алгоритма DCP для однородной среды.

Альтернативным способом сокращения времени исполнения является копирование задачи на различные ресурсы. Основная идея дублирования состоит в том, что планирование использует время простоя ресурсов для копирования предшествующей задачи. Это позволяет избежать передачи результатов от предшествующей задачи последующей, таким образом, уменьшается коммуникационная стоимость. Дублирование также может решить проблему максимина.

Алгоритмы, базирующиеся на дублировании, отличаются стратегиями выбора задач для дублирования. Первоначально, алгоритмы этой группы применялись для неограниченного числа идентичных процессоров, таких как многопроцессорные системы с распределенной памятью. Также они имеют более высокую сложность, чем алгоритмы, осаждавшиеся выше. Например, Darbha и др. [79] представляют алгоритм под названием «алгоритм планирования, основанный на дублировании задач» (TDS — Task Duplication-based Scheduling Algorithm) для машин с распределенной памятью, имеющий сложность $O(v^2)$ для гомогенных сред.

Для применения дублирования в гетерогенных средах был предложен новый алгоритм под названием «алгоритм планирования, основанный на дублировании задач, для гетерогенных систем» (TANH — Task duplication-based scheduling Algorithm for Network of Heterogeneous systems), который представлен в работах [80] и [81]. Дублирование уже получило некоторое признание [71, 72], но на сегодняшний день алгоритмы, основанные на дублировании, в распределенных вычислительных средах имеют дело только с независимыми заданиями.

Применение *алгоритмов кластеризации* в параллельных и распределенных системах — это эффективный способ уменьшить коммуникационную задержку в ориентированном ациклическом графе. Основная идея данных алгоритмов заключается в кластеризации взаимосвязанных задач в маркированные группы для дальнейшего их присвоения некоторой группе ресурсов. Примерами алгоритмов кластеризации являются DSC (Dominant Sequence Clustering) [21], CASS-II [83]. Экспериментальные исследования алгоритмов кластеризации приводятся в работах [83, 84].

Рассмотрим более подробно следующие алгоритмы кластеризации: алгоритм KB/L, алгоритм Саркара, алгоритм DSC и алгоритм POS.

Алгоритм KB/L

В работах [20, 85] рассматривается алгоритм линейной кластеризации Кима и Брауна, также называемый алгоритмом KB/L. Алгоритм KB/L предназначен для кластеризации графа задания и предполагает, что количество вычислительных узлов неограниченно.

Первоначально все дуги графа задания маркируются как «не рассмотренные». На первом шаге выполнения алгоритма KB/L происходит поиск критического пути графа задания, который включает только «нерассмотренные» дуги, с помощью стоимостной функции веса (1). Все вершины найденного критического пути объединяются в один кластер, коммуникационные стоимости дуг обнуляются. На втором шаге дуги, инцидентные вершинам критического пути, маркируются как «рассмотренные». Описанные выше шаги алгоритма KB/L повторятся до тех пор, пока все дуги не будут рассмотрены.

Ким в работе [20] использует стоимостную функцию

$$Cost\ function = w_1 * \sum \tau_i + (1 - w_1)\Big(w_2 * \sum c_{ij} + (1 - w_2) * \sum c_{ij}^{adj}\Big) \qquad (1)$$

для определения длины критического пути графа задания, где w_1 и w_2 — нормализующие множители, $\sum \tau_i$ — сумма вычислительных стоимостей всех вершин критического пути, c_{ij}^{adj} — коммуникационная стоимость дуг между вершиной критического пути и всеми его смежными вершинами, не входящими в критический путь. Ким не приводит систематического способа для определения нормализующих множителей. Положим, $w_1 = \frac{1}{2}$ и $w_2 = \frac{1}{2}$ для стоимостной функции, уменьшающей длину критического пути. Целевой функцией алгоритма KB/L является минимизация параллельного времени графа.

Рассмотрим алгоритм KB/L на примере графа задания на рис. 6а. Результат кластеризации с помощью алгоритма KB/L с параллельным временем 12,5 приведен на рис. 6б.

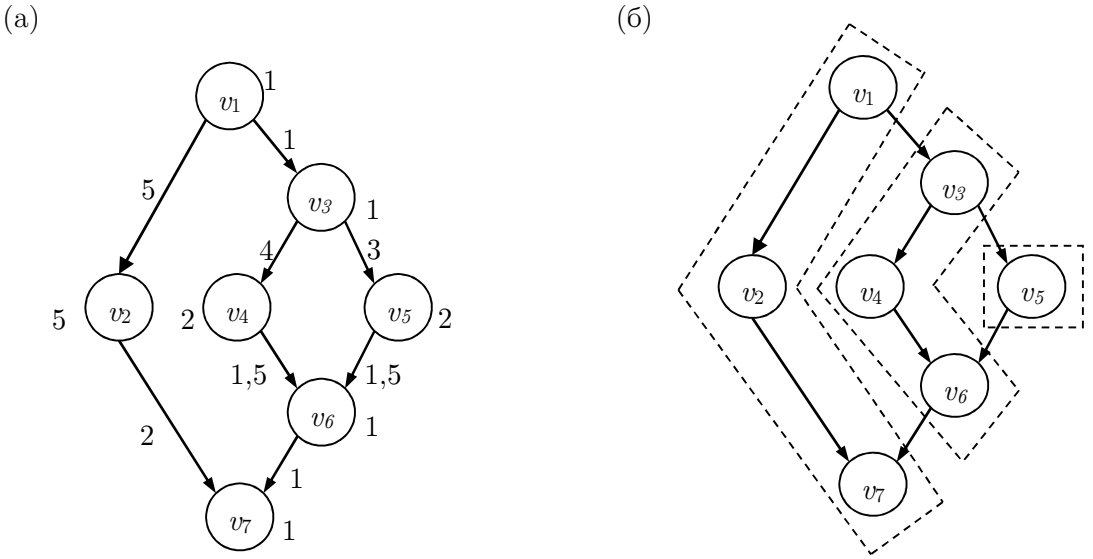

Рис. 6. (а) Граф с весами и разметкой;
(б) кластеризация графа с помощью алгоритма KB/L

На первом шаге критический путь состоит из вершин v_1, v_2 и v_7. Объединяем данные вершины в один кластер и исключаем их из дальнейшего рассмотрения. Оставшийся граф содержит вершины v_3, v_4, v_5, v_6. Новый критический путь состоит из вершин v_3, v_4, v_6. Данные вершины также группируются в один кластер. Окончательно, имеем три кластера $W_0 = \{v_1, v_2, v_7\}$, $W_1 = \{v_3, v_4, v_6\}$ и $W_2 = \{v_5\}$.

Алгоритм KB/L имеет сложность $O\big(v(v+e)\big)$, где v — число вершин, e — число дуг графа.

Алгоритм Саркара

В работе [86] рассматривается алгоритм кластеризации Саркара (Sarkar). Суть алгоритма может быть описана следующим образом. Все дуги графа задания сортируются в порядке убывания их коммуникационных стоимостей. Начиная с дуги с большей коммуникационной стоимостью, производится процесс их обнуления. Коммуникационная стоимость дуги обнуляется только в том случае, если параллельное время не увеличится на следующем шаге. Алгоритм Саркара завершается, когда рассмотрены все дуги графа задания. Целевой функцией алгоритма также является минимизация параллельного времени графа.

Рассмотрим алгоритм Саркара на примере графа задания на рис. 6а. Результат кластеризации с помощью алгоритма Саркара с параллельным временем равным 10 приведен на рис. 7.

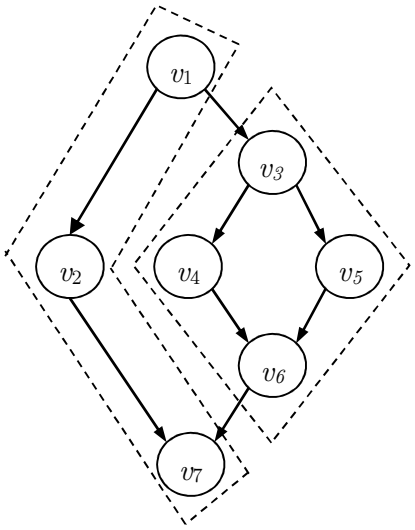

Рис. 7. Нелинейная кластеризация графа с помощью алгоритма Саркара

На первом шаге алгоритма Саркара выполним сортировку коммуникационных стоимостей дуг графа задания по убыванию. Получим следующий список дуг $\{(v_1, v_2), (v_3, v_4), (v_3, v_5), (v_2, v_7), (v_4, v_6), (v_5, v_6), (v_1, v_3), (v_6, v_7)\}$. В начальный момент времени каждая вершина графа задания находится в отдельном кластере. Параллельное время равно 14. На втором шаге обнуляем две первые дуги из списка (v_1, v_2) и (v_3, v_4). При этом параллельное время уменьшится до 12,5. На третьем шаге обнуляем коммуникационную стоимость дуги (v_3, v_5) и т.д. Окончательно, имеем два кластера $W_0 = \{v_1, v_2, v_7\}$ и $W_1 = \{v_3, v_4, v_5, v_6\}$ и параллельное время равное 10.

Алгоритм Саркара имеет сложность $O\big(e(v+e)\big)$, где v — число вершин графа, e — число дуг графа.

Алгоритм DSC

Алгоритм кластеризации доминирующей последовательности DSC (Dominant Sequence Clustering) [21] относится к эвристикам кластеризации. Главная идея алгоритма DSC состоит в выполнении последовательности шагов по обнулению коммуникационных стоимостей дуг графа задания с целью сокращения длины доминирующей последовательности, т.е. минимизации параллельного времени. Здесь *доминирующая последовательность* — это критический путь распланированного графа (рис. 8в). Заметим, что обычно под критическим путем графа понимают путь наибольшей длины, включающий вычислительные стоимости его вершин и ненулевые коммуникационные стоимости дуг (рис. 8а).

В начальный момент времени до выполнения алгоритма DSC каждая вершина содержится в отдельном кластере, все дуги графа задания помечаются как «нерассмотренные». После рассмотрения некоторой дуги на предмет возможности ее обнуления, дуга обозначается как «рассмотренная», а вершина, из которой исходит дуга, помечается как «распланированная». Распланированные вершины образуют множество S, не распланированные вершины составляют множество U. Вершина называется «свободной», если все ее предшествующие вершины распланированы.

На первом шаге алгоритма DSC из дуг, принадлежащих доминирующей последовательности графа задания, выбирается первая нерассмотренная дуга. Поиск дуги в доминирующей последовательности производится сверху вниз. На втором шаге дуга обнуляется, а ее вершины объединяются в один кластер, если параллельное время не увеличивается. Порядок выполнения задач в кластере определяется наибольшим значением $bot\_level(n_x, i)$, которое рассчитывается как сумма всех коммуникационных стоимостей дуг и вычислительных стоимостей вершин между вершиной n_x и нижней вершиной графа в доминирующей последовательности. Алгоритм DSC завершается, когда все дуги рассмотрены.

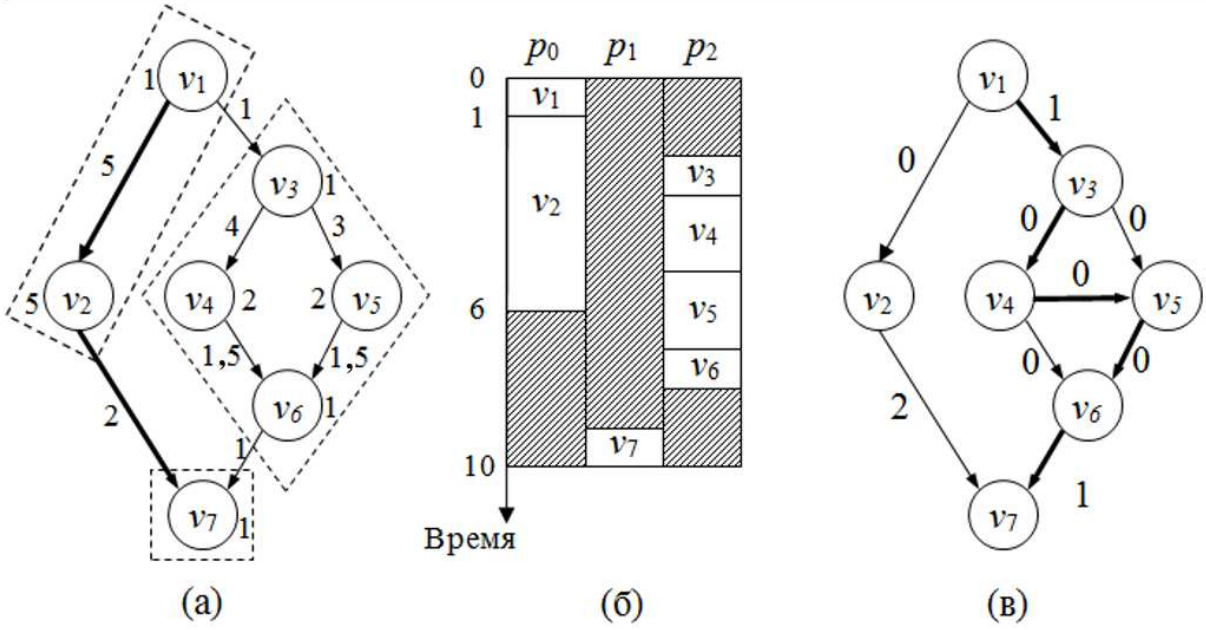

Рис. 8. (а) Кластеризованный граф и его критический путь; (б) диаграмма Ганта; (в) распланированный граф и его доминирующая последовательность

Рассмотрим алгоритм DSC на примере графа задания на рис. 6а. Результат кластеризации с помощью алгоритма DSC с параллельным временем равным 10.5 приведен на рис. 9.

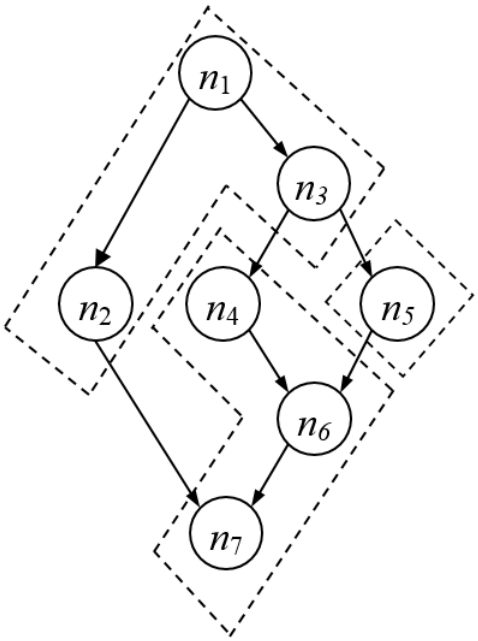

Рис. 9. Нелинейная кластеризация графа с помощью алгоритма DSC

На первом шаге алгоритма DSC определим доминирующую последовательность $DS_1 = \langle v_1, v_2, v_7 \rangle$ и параллельное время $PT_1 = 14$. Рассмотрим первую дугу из доминирующей последовательности — (v_1, v_2). Обнуление коммуникационной стоимости дуги уменьшит параллельное время до 13,5, поэтому произведем объединение вершин v_1, v_2 в один кластер.

Находим новую доминирующую последовательность $DS_2 = \langle v_1, v_3, v_4, v_6, v_7 \rangle$ и параллельное время $PT_1 = 13{,}5$. Рассмотрим первую дугу (v_1, v_3) из этой доминирующей последовательности. Ее обнуление уменьшит критический путь до 12,5. Следовательно, обнуляем данную дугу.

На третьем шаге алгоритма доминирующая последовательность не именилась $DS_3 = DS_2$. Рассмотрим дугу (v_3, v_4). Ее обнуление приведет к увеличению параллельного времени до 13,5. Следовательно, вершины v_3 и v_4 остаются в разных кластерах.

Аналогично, рассматриваем дуги (v_4, v_6), (v_3, v_5), (v_5, v_6) и (v_6, v_7) соответственно. Окончательно, имеем три кластера $M_0 = \{v_1, v_2, v_3\}$, $M_1 = \{v_4, v_6, v_7\}$ и $M_2 = \{v_5\}$ параллельное время равное 10,5.

Алгоритм DSC в общем случае имеет сложность $O\big((e + v)\log v\big)$, где v — число вершин графа, e — число дуг графа.

Алгоритм POS

Алгоритм планирования POS (Problem-Oriented Scheduling) [87, 88] для распределенных кластерных вычислительных сред в отличие от алгоритма DSC позволяет планировать запуск одной задачи на нескольких процессорных ядрах с учетом ограничений по масштабируемости данной задачи. Вычислительное задание, представляющее собой поток работ, моделируется в виде нагруженного, помеченного ориентированного графа задания. Каждая задача-вершина помечается парой натуральных чисел, первое из которых задает

время выполнения задачи на одном процессорном ядре, а второе — максимальное количество ядер, до которого задача демонстрирует ускорение близкое к линейному. Веса дуг задают объем данных, передаваемый между задачами. Предложенный алгоритм планирования ресурсов предусматривает построение последовательности конфигураций графа задания с целью минимизации критического пути в данном графе. В начальной конфигурации граф задания разбивается в каноническую ярусно-параллельную форму (ЯПФ). Задаются начальные кластеризация задач по вычислительным кластерам и расписание их выполнения. На следующем шаге происходит переход от начальной конфигурации графа к новой конфигурации с помощью перемещения одной из задач по ярусам ЯПФ и ее присоединения к определенному кластеру. Таким образом, изменяются кластеризация и расписание. Переход к новой конфигурации происходит только в том случае, когда сложность критического пути не увеличивается. Процесс построения последовательности конфигураций завершается, когда все вершины зафиксированы. Данный алгоритм реализован в брокере ресурсов [89–91] системы DiVTB [92–98] на платформе UNICORE на языке программирования Java.

Заключение

В статье были рассмотрены основные технологии распределенных вычислений: грид-вычисления и облачные вычисления. Дан обзор алгоритмов планирования ресурсов в распределенных вычислительных средах. На сегодняшний день предложено большое количество алгоритмов планирования, ориентированных на использование в распределенных вычислительных средах. Некоторые из этих алгоритмов производят планирование с учетом потоков работ в сложных приложениях. Однако часто алгоритмы не учитывают проблемно-ориентированную специфику, что может существенно повысить эффективность планирования. В связи с этим, перспективным является направления, связанное с разработкой алгоритмов планирования ресурсами в проблемно-ориентированных средах, которые позволяют создать эффективную и действенную систему планирования ресурсов.

Работа выполнялась при поддержке Российского фонда фундаментальных исследований (проект № 14-07-31159мол_а «Мой первый грант»).

Литература

1. Buyya, R. Economic Models for Resource Management and Scheduling in Grid Computing / R. Buyya, D. Abramson et al. // Journal of Concurrency and Computation: Practice and Experience. — 2002. — Vol. 14, — Issue 13–15. — P. 1507–1542.
2. Foster, I. A Quality of Service Architecture That Combines Resource Reservation and Application Adaptation / I. Foster, A. Roy, V. Sander // Proceedings 8th Int. Workshop on Quality of Service (Pittsburgh, PA, USA, June 5–7, 2000). — Carnegie Mellon University, 2000. — P. 181–188.
3. Foster, I. The Anatomy of the Grid: Enabling Scalable Virtual Organizations / I. Foster, C. Kesselman, S. Tuecke // International Journal of Supercomputer Applications and High Performance Computing. — 2001. — Vol. 15, — No 3. — P. 200–222.
4. Foster, I. The Grid. Blueprint for a new computing infrastructure / I. Foster, C. Kesselman — San Francisco: Morgan Kaufman, 1999. — 677 p.

5. Jennings, R. Cloud Computing with the Windows Azure Platform / R. Jennings, — Indianapolis, Indiana: Wiley Publishing, Inc., 2009. — 360 p.
6. Marshall, P. T. Improving Utilization of Infrastructure Clouds / P. Marshall, K. Keahey, T. Freeman // Cluster, Cloud and Grid Computing (CCGrid 2011): Proceedings of the IEEE/ACM International Symposium (Newport Beach, CA, USA. May 23–26, 2011). — IEEE Computer Society, 2011. — P. 205–214.
7. NIST Special Publication 800-145. A NIST Definition of Cloud Computing. URL: http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf (дата обращения: 01.02.2014).
8. Sanderson, D. Programming Google App Engine: Build and Run Scalable Web Apps on Google's Infrastructure / D. Sanderson, — O'Reilly Media, 2009. — 400 p.
9. Dong, F. Scheduling algorithms for grid computing: State of the art and open problems. Technical Report No. 2006-504 / F .Dong, S.G. Akl, — Queen's University, Canada, 2006. — P. 55.
10. Berman, F. Application-Level Scheduling on Distributed Heterogeneous Networks / F. Berman, R. Wolski et al. // Supercomputing: Proceedings of the ACM/IEEE conference (Pittsburgh, Pennsylvania USA, May 25–28, 1996). — IEEE Computer Society, 1996. — P. 39–39.
11. Iverson, M. Dynamic, Competitive Scheduling of Multiple DAGs in a Distributed Heterogeneous Environment / M. Iverson, F. Ozguner // Proceedings of Seventh Heterogeneous Computing Workshop (Orlando, Florida USA, March 30, 1998). — IEEE Computer Society, 1998. — P. 70–78.
12. Maheswaran, M. Dynamic Matching and Scheduling of a Class of Independent Tasks onto Heterogeneous Computing Systems / M. Maheswaran, S. Ali et al. // Journal of Parallel and Distributed Computing. — 1999. — Vol. 59, — No. 2. — P. 107–131.
13. Zhu, Y. A Survey on Grid Scheduling Systems. — Technical Report SJTU_CS_TR_200309001. — Department of Computer Science and Engineering, Shanghai Jiao. — 2013. / Y. Zhu, L.M. Ni. URL: http://www.cs.sjtu.edu.cn/~yzhu/reports/SJTU_CS_TR_200309001.pdf (дата обращения: 01.08.2014).
14. Belhajjame, K. A flexible workflow model for process-oriented applications / K. Belhajjame, G. Vargas-Solar, C. Collet // Web Information Systems Engineering (WISE'01): Proceedings of the Second International Conference (Kyoto, Japan, December 3–6, 2001). — IEEE Computer Society. — Vol. 1, — No. 1. — P. 72–80.
15. Condor. URL: http://www.cs.wisc.edu/condor (дата обращения: 04.06.2014).
16. Laszewski, G. CoG Kits: A Bridge between Commodity Distributed Computing and High-Performance Grids / G. Laszewski, I. Foster et al. // Proceedings of the ACM Java Grande 2000 Conference (San Francisco, CA, USA, June 3–5, 2000). — ACM Press. — P. 97–106.
17. Deelman, E. Pegasus: Mapping Scientific Workflows onto the Grid / E. Deelman, J. Blythe et al. // AcrossGrids Conference (AxGrids 2004): Proceedings of the Second European Conference (Nicosia, Cyprus, January 28–30, 2004). — Springer. — P. 11–26.
18. Cao, J. GridFlow: Workflow Management for Grid Computing / J. Cao, S.A. Jarvis et al. // Cluster Computing and the Grid (CCGrid'03): Proceedings of the 3rd International Symposium (Tokyo, Japan, May 12–15, 2003). — IEEE Computer Society, 2003. — P. 198–205.

19. Wieczorek, M. Scheduling of Scientific Workflows in the ASKALON Grid Environment / M. Wieczorek, R. Prodan, T. Fahringer // ACM SIGMOD Record. — 2005. — Vol. 34, — No. 3. — P. 56–62.
20. Kim, S.J. A general approach to multiprocessor scheduling. — Report TR-88-04. — Department of Computer Science, University of Texas at Austin. — 1988. / S.J. Kim. URL: ftp://ftp.cs.utexas.edu/pub/techreports/tr88-04.pdf (дата обращения: 01.08.2014).
21. Yang, T. DSC: Scheduling Parallel Tasks on an Unbounded Number of Processors / T. Yang, A. Gerasoulis // IEEE Transactions on Parallel and Distributed Systems. — 1994. — Vol. 5, — No. 9. — P. 951–967.
22. Baker, M. Grids and Grid Technologies for Wide-area Distributed Computing / M. Baker, R. Buyya, D. Laforenza // Journal of Software-Practice & Experience. — 2002. — Vol. 32, — No. 15. — P. 1437–1466.
23. Schopf, J. Ten Actions When SuperScheduling, document of Scheduling Working Group, Global Grid Forum. / J. Schopf. URL: http://www.ggf.org/documents/GFD.4.pdf (дата обращения: 12.06.2014).
24. Berman, F. Adaptive Computing on the Grid Using AppLeS / F. Berman, R. Wolski et al. // IEEE Transactions on Parallel and Distributed Systems. — 2003. — Vol. 14, — No. 4. — P. 369–382.
25. Hamscher, V. Evaluation of Job-Scheduling Strategies for Grid Computing / V. Hamscher, U. Schwiegelshohn et al. // GRID 2000: Proceedings of the First IEEE/ACM International Workshop (Bangalore, India, December 17, 2000). — IEEE Computer Society, 2000. — P. 191–202.
26. Mateescu, G. Quality of Service on the Grid via Metascheduling with Resource Co-Scheduling and Co-Reservation / G. Mateescu // International Journal of High Performance Computing Applications. — 2003. — Vol. 17, — No. 3. — P. 209–218.
27. The Globus Toolkit. URL: http://www.globus.org (дата обращения: 04.01.2013).
28. Czajkowski, K. Grid Information Services for Distributed Resource Sharing / K. Czajkowski, S. Fitzgerald et al. // High-Performance Distributed Computing (HPDC-10): Proceedings the 10th IEEE International Symposium (San Francisco, California, USA, August 7–9, 2001). — IEEE Computer Society, 2001. — P. 181–194.
29. Khokhar, A.A. Heterogeneous Computing: Challenges and Opportunities / A.A. Khokhar, V.K. Prasanna et al. // IEEE Computer. — 1993. — Vol. 26, — No. 6. — P. 18–27.
30. Siegel, H.J. Software Support for Heterogeneous Computing / H.J. Siegel, H.G. Dietz, J.K. Antonio // ACM Computing Surveys. — 1996. — Vol. 28, — No. 1. — P. 237–239.
31. Cooper, K. New Grid Scheduling and Rescheduling Methods in the GrADS Project / K. Cooper, A. Dasgupta et al. // International Parallel and Distributed Processing Symposium (IPDPS'04): Proceedings of the 18th International Symposium (Santa Fe, New Mexico USA, April 26–30, 2004). — IEEE Computer Society, 2004. — P. 199–206.
32. Czajkowski, K. A Resource Management Architecture for Metacomputing Systems / K. Czajkowski, I. Foster et al. // Job Scheduling Strategies for Parallel Processing (JSSPP 1998): Proceedings of the 4th Workshop (Orlando, Florida USA, March 30, 1998). — Springer, Lecture Notes in Computer Science, 1998. — Vol. 1459. — P. 62–82.
33. OpenPBS. URL: http://www.openpbs.org (дата обращения: 14.12.2013).
34. Condor. URL: http://www.cs.wisc.edu/condor (дата обращения: 04.12.2013).

35. Wolski, R. The Network Weather Service: A Distributed Resource Performance Forecasting Service for Metacomputing / R. Wolski, N.T. Spring, J. Hayes // Future Generation Computing Systems. — 1999. — Vol. 15, — No. 5–6. — P. 757–768.
36. Sacerdoti, F.D. Wide area cluster monitoring with Ganglia / F.D. Sacerdoti, M.J. Katz et al. // Cluster Computing (CLUSTER 2003): Proceedings of IEEE International Conference (Hong Kong, December 1–4, 2003). — IEEE Computer Society, 2003. — P. 289-298.
37. NIST Special Publication 800-145. A NIST Definition of Cloud Computing. URL: http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf (дата обращения: 01.08.2014).
38. Streit, A. UNICORE: Getting to the heart of Grid technologies / A. Streit // eStrategies. — 2009. — Vol. 3. — P. 8–9.
39. Streit, A. UNICORE - What lies beneath Grid functionality? / A. Streit // eStrategies. — 2008. — Vol. 7. — P. 38–39.
40. Casavant, T. A Taxonomy of Scheduling in General-purpose Distributed Computing Systems / T. Casavant, J. Kuhl // IEEE Transactions on Software Engineering. — 1988. — Vol. 14, — No. 2. — P. 141–154.
41. Braun, R. A Comparison of Eleven Static Heuristics for Mapping a Class of Independent Tasks onto Heterogeneous Distributed Computing Systems / R. Braun, H. Siegel et al. // Parallel and Distributed Computing. — 2001. — Vol. 61, — No. 6, — P. 810–837.
42. Casanova, H. Heuristics for Scheduling Parameter Sweep Applications in Grid Environments / H. Casanova, A. Legrand et al. // Heterogeneous Computing Workshop (HCW'00): Proceedings of the 9th Workshop (Cancun, Mexico, May 1, 2000). — IEEE Computer Society, 2000. — P. 349–363.
43. You, S.Y. Task Scheduling Algorithm in GRID Considering Heterogeneous Environment / S.Y. You, H.Y. Kim et al. // Parallel and Distributed Processing Techniques and Applications (PDPTA '04): Proceedings of the International Conference (Nevada, USA, June 21–24, 2004). — CSREA Press, 2004. — Vol. 1. — P. 240–245.
44. Kurowski, K. Improving Grid Level Throughput Using Job Migration And Rescheduling / K. Kurowski, B. Ludwiczak et al. // Scientific Programming. — 2004. — Vol. 12, — No. 4. — P. 263–273.
45. Takefusa, A. A Study of Deadline Scheduling for Client-Server Systems on the Computational Grid / A. Takefusa, S. Matsuoka et al. // High Performance Distributed Computing (HPDC-10): Proceedings of the 10th IEEE International Symposium (San Francisco, California, USA, August 7–9, 2001). — IEEE Computer Society, 2001. — P. 406–415.
46. Chen, H. Distributed Dynamic Scheduling of Composite Tasks on Grid Computing Systems / H. Chen, M. Maheswaran // International Parallel and Distributed Processing Symposium (IPDPS 2002): Proceedings of the 16th International Symposium (Fort Lauderdale, FL, USA, April 15-19, 2002). — IEEE Computer Society, 2002. — P. 88–97.
47. Muthuvelu, N. A Dynamic Job Grouping-Based Scheduling for Deploying Applications with Fine-Grained Tasks on Global Grids / N. Muthuvelu, J. Liu et al. // Grid Computing and e-Research (AusGrid 2005): Proceedings of the 3rd Australasian Workshop (Newcastle, NSW, Australia, January 30 – February 4, 2005). — Australian Computer Society, 2005. — P. 41–48.

48. Wright, D. Cheap Cycles from the Desktop to the Dedicated Cluster: Combining Opportunistic and Dedicated Scheduling with Condor / D. Wright URL: http://www.linuxclustersinstitute.org/conferences/archive/2001/PDF/wright_wisc.pdf (дата обращения: 04.08.2014).
49. Chapin, S.J. The Legion Resource Management System / S.J. Chapin, D. Katramatos et al. // Job Scheduling Strategies for Parallel Processing (JSSPP '99): Proceedings of the 5th Workshop (San Juan, Puerto Rico, April 16, 1999). — Springer, Lecture Notes in Computer Science, 1999. — Vol. 1659. — P. 162–178.
50. Rotithor, H.G. Taxonomy of Dynamic Task Scheduling Schemes in Distributed Computing Systems / H.G. Rotithor // Computer and Digital Techniques. — 1994. — Vol. 141, — No. 1. — P. 1–10.
51. James, H.A. Scheduling in Metacomputing Systems. — Ph.D. Thesis. — The Department Of Computer Science, University of Adelaide, Australia. — 1999. / H.A. James URL: http://digital.library.adelaide.edu.au/dspace/bitstream/2440/19450/1/09phj274.pdf (дата обращения: 04.08.2014).
52. Aggarwal, A.K. An Adaptive Generalized Scheduler for Grid Applications / A.K. Aggarwal, R.D. Kent // High Performance Computing Systems and Applications (HPCS'05): Proceedings of the 19th Annual International Symposium (Guelph, Ontario Canada, May 15–18, 2005). — IEEE Computer Society, 2005. — P. 15–18.
53. Sabin, G. Scheduling of Parallel Jobs in a Heterogeneous Multi-Site Environment / G. Sabin, R. Kettimuthu et al. // Job Scheduling Strategies for Parallel Processing (JSSPP'03): Proceedings of the 9th International Workshop (Seattle, WA, USA, June 24, 2003). — Springer, Lecture Notes in Computer Science, 2003. — Vol. 2862. — P. 87–104.
54. Arora, M. A Decentralized Scheduling and Load Balancing Algorithm for Heterogeneous Grid Environments / M. Arora, S.K. Das, R. Biswas // International Conference on Parallel Processing Workshops (ICPPW'02): Proceedings of International Conference (Vancouver, British Columbia Canada, August 20-23, 2002). — IEEE Computer Society, 2002. — P. 499–505.
55. El-Rewini, H. Task Scheduling in Parallel and Distributed Systems / H. El-Rewini, T. Lewis, H. Ali — Prentice Hall, 1994. — 290 p.
56. Buyya, R. The Grid Economy / R. Buyya, D. Abramson, S. Venugopal // Proceedings of the IEEE. — 2005. — Vol. 93, — No. 3. — P. 698–714.
57. Buyya, R. An Evaluation of Economy-based Resource Trading and Scheduling on Computational Power Grids for Parameter Sweep Applications / R. Buyya, J. Giddy, D. Abramson // Active Middleware Services (AMS 2000): Proceedings of the 2nd International Workshop (Pittsburgh, USA, August 1, 2000). — Kluwer Academic Press, 2000. — Vol. 583. — P. 221–230.
58. Yu, J. QoS-based Scheduling of Workflow Applications on Service Grids / J. Yu, R. Buyya, C.K. Tham // e-Science and Grid Computing (e-Science'05): Proceedings of the 1st IEEE International Conference (Melbourne, Australia, December 5–8, 2005). — IEEE Computer Society, 2005. — P. 1–9.
59. Venugopal, S. A Deadline and Budget Constrained Scheduling Algorithm for eScience Applications on Data Grids / S. Venugopal, R. Buyya // International Conference on Algorithms and Architectures for Parallel Processing (ICA3PP-2005): Proceedings of 6th

International Conference (Melbourne, Australia, October 2-3, 2005). — Springer, Lecture Notes in Computer Science, 2005. — P. 60–72.

60. Ernemann, C. Economic Scheduling in Grid Computing / C. Ernemann, V. Hamscher, R. Yahyapour // Job Scheduling Strategies for Parallel Processing (JSSPP'02): Proceedings of 8th Workshop (Edinburgh, Scotland, UK, July 24, 2002). — Springer, Lecture Notes in Computer Science, 2002. — P. 128–152.
61. Zhu, Y. Incentive-based P2P Scheduling in Grid Computing / Y. Zhu, L. Xiao et al. // Grid and Cooperative Computing (GCC'04): Proceedings of the 3rd International Conference (Wuhan, China, October 21-24, 2004). — Springer, Lecture Notes in Computer Science, 2004. — P. 209–216.
62. Young, L. Scheduling Architecture and Algorithms within the ICENI Grid Middleware / L. Young, S. McGough et al. // Proceedings of UK e-Science All Hands Meeting (Nottingham, UK, September 2003). — IOP Publishing Ltd., 2003. — P. 5–12.
63. Kim, S. A Genetic Algorithm Based Approach for Scheduling Decomposable Data Grid Applications / S. Kim, J.B. Weissman // International Conference on Parallel Processing (ICPP'04): Proceedings of the International Conference (Montreal, Quebec Canada, August 15–18, 2004). — IEEE Computer Society, 2004. — P. 406–413.
64. Song, S. Security-Driven Heuristics and A Fast Genetic Algorithm for Trusted Grid Job Scheduling / S. Song, Y. Kwok, K. Hwang // International Parallel and Distributed Processing Symposium (IPDPS'05): Proceedings of 19th IEEE International Symposium (Denver, Colorado USA, April 25–29, 2005). — IEEE Computer Society, 2005. — P. 65–74.
65. Spooner, D.P. Localised Workload Management using Performance Prediction and QoS Contracts / D.P. Spooner, J. Cao et al. // UK Performance Engineering Workshop (UKPEW 2002): Proceedings of the 18th Annual Workshop (Glasgow, UK, July 10–11, 2002). — University of Glasgow, 2002. — P. 69–80.
66. Liu, Y. Survey on Grid Scheduling. — Department of Computer Science, University of Iowa. — 2004.
67. Abraham, A. Nature's Heuristics for Scheduling Jobs on Computational Grids / A. Abraham, R. Buyya and B. Nath // Advanced Computing and Communications (ADCOM 2000): Proceedings of 8th IEEE International Conference (Cochin, India, December 14–16, 2000). — IEEE Computer Society, 2000. — P. 45–52.
68. Shan, H. Scheduling in Heterogeneous Grid Environments: The Effects of Data Migration / H. Shan, L. Oliker et al. // Advanced Computing and Communication (ADCOM 2004): Proceedings of the 12th IEEE International Conference (Ahmedabad Gujarat, India, December 15–18, 2004). — IEEE Computer Society, 2004. — P. 1–8.
69. He, X. A QoS Guided Min-Min Heuristic for Grid Task Scheduling / X. He, X. Sun, G. Laszewski // Computer Science and Technology: Special Issue on Grid Computing. — 2003. — Vol. 18, — No. 4. — P. 442–451.
70. Wu, M. Segmented Min-Min: A Static Mapping Algorithm for Meta-Tasks on Heterogeneous Computing Systems / M. Wu, W. Shu, H. Zhang // Heterogeneous Computing Workshop (HCW'00): Proceedings of the 9th Workshop (Cancun, Mexico, May 1, 2000). — IEEE Computer Society, 2000. — P. 375–385.
71. Subramani, V. Distributed Job Scheduling on Computational Grids using Multiple Simultaneous Requests / V. Subramani, R. Kettimuthu et al. // High Performance Distributed

Computing (HPDC 2002): Proceedings of 11th IEEE Symposium (Edinburgh, Scotland, July 23–26, 2002). — IEEE Computer Society, 2002. — P. 359–366.

72. Silva, D.P. Trading Cycles for Information: Using Replication to Schedule Bag-of-Tasks Applications on Computational Grids / D.P. Silva, W. Cirne, F.V. Brasileiro // Euro-Par 2003: Proceedings of the 9th International Conference (Klagenfurt, Austria, August 26–29, 2003). — Springer, Lecture Notes in Computer Science, 2003. — P. 169–180.
73. Radulescu, A. On the Complexity of List Scheduling Algorithms for Distributed Memory Systems / A. Radulescu, A.J.C. Gemund // Supercomputing (SC'99): Proceedings of 13th International Conference (Portland, Oregon, USA, November 13–19, 1999). — IEEE Computer Society, 1999. — P. 68–75.
74. Sakellariou, R. A Low-cost Rescheduling Policy for Efficient Mapping of Workflows on Grid Systems / R. Sakellariou, H. Zhao // Scientific Programming. — 2004. — Vol. 12, — No. 4. — P. 253–262.
75. Topcuoglu, H. Performance-Effective and Low-Complexity Task Scheduling for Heterogeneous Computing / H. Topcuoglu, S. Hariri, M.Y. Wu // IEEE Transactions on Parallel and Distributed Systems. — 2002. — Vol. 13, — No. 3. — P. 260–274.
76. Zhao, H. An Experimental Investigation into the Rank Function of the Heterogeneous Earliest Finish Time Scheduling Algorithm / H. Zhao, R. Sakellariou // Euro-Par 2003: Proceedings of the 9th International Conference (Klagenfurt, Austria, August 26–29, 2003). — Springer, Lecture Notes in Computer Science, 2003. — P. 189–194.
77. Sakellariou, R. A Hybrid Heuristic for DAG Scheduling on Heterogeneous Systems / R. Sakellariou, H. Zhao // International Parallel and Distributed Processing Symposium (IPDPS'04): Proceedings of the 18th International Symposium (Santa Fe, New Mexico USA, April 26–30, 2004). — IEEE Computer Society, 2004. — P. 111–123.
78. Ma, T. Critical-Path and Priority based Algorithms for Scheduling Workflows with Parameter Sweep Tasks on Global Grids / T. Ma, R. Buyya // Computer Architecture and High Performance Computing (SBAC-PAD 2005): Proceedings of the 17th International Symposium (Rio de Janeiro, Brazil, October 24–27, 2005). — IEEE Computer Society, 2005. — P. 251–258.
79. Darbha, S. Optimal Scheduling Algorithm for Distributed Memory Machines / S. Darbha, D.P. Agrawal // IEEE Transactions on Parallel and Distributed Systems. — 1998. — Vol. 9, — No. 1. — P. 87–95.
80. Ranaweera, S. A Task Duplication Based Scheduling Algorithm for Heterogeneous Systems / S. Ranaweera, D.P. Agrawal // International Parallel and Distributed Processing Symposium (IPDPS'00): Proceedings of 14th International Symposium (Cancun, Mexico, May 1–5, 2000). — IEEE Computer Society, 2005. — P. 445–450.
81. Bajaj, R. Improving Scheduling of Tasks in A Heterogeneous Environment / R. Bajaj, D.P. Agrawal // IEEE Transactions on Parallel and Distributed Systems. — 2004. — Vol. 15, — No. 2. — P. 107–118.
82. Gerasoulis, A. A comparison of clustering heuristics for scheduling directed acyclic graphs on multiprocessors / A. Gerasoulis, T. Yang // Parallel and Distributed Computing. — 1992. — Vol. 16, — No. 4. — P. 276–291.
83. Liou, J. An Efficient Task Clustering Heuristic for Scheduling DAGs on Multiprocessors / J. Liou, M.A. Palis // Parallel and Distributed Processing (SPDP 1996): Proceedings of

the Eighth IEEE Symposium (New Orleans, Louisiana, USA, October 23–26, 1996). — IEEE Computer Society, 1996. — P. 152–156.

84. Liou, J. A Comparison of General Approaches to Multiprocessor Scheduling / J. Liou, M.A. Palis // International Parallel Processing Symposium (IPPS '97): Proceedings the 11th International Symposium (Geneva, Switzerland, April 1–5, 1997). — IEEE Computer Society, 1996. — P. 152–156.
85. Kim, S.J. A general approach to mapping of parallel computation upon multiprocessor architectures / S.J. Kim, J.C. Browne // International Conference on Parallel Processing, (ICPP'88): Proceedings of the International Conference (The Pennsylvania State University, University Park, PA, USA, August 1988). — Pennsylvania State University Press., 1988. — Vol. 3. — P. 1-8.
86. Sarkar, V. Partitioning and Scheduling Parallel Programs for Execution on Multiprocessors / V. Sarkar — Cambridge, MA: The MIT Press, 1989. — P. 215.
87. Шамакина, А.В. Формальная модель задания в распределенных вычислительных средах / А.В. Шамакина, Л.Б. Соколинский // Параллельные вычислительные технологии (ПаВТ'2014): Труды международной научной конференции (Ростов-на-Дону, 1–3 апреля 2014 г.). — Челябинск: Издательский центр ЮУрГУ, 2014. — С. 343–354.
88. Радченко, Г.И. Модели и методы профилирования и оценки времени выполнения потоков работ в суперкомпьютерных системах / Г.И. Радченко, Л.Б. Соколинский, А.В. Шамакина // Вычислительные методы и программирование: Новые вычислительные технологии. — 2013. — Т. 14. — Вып 4. — С. 96–103.
89. Шамакина. А.В. Брокер ресурсов для поддержки проблемно-ориентированных сред / А.В. Шамакина // Вестник ЮУрГУ. Серия "Вычислительная математика и информатика". — 2012. — № 46(305). — Вып. 1. — С. 88–98.
90. Шамакина. А.В. Организация брокера ресурсов в системе CAEBeans / А.В. Шамакина // Вестник Южно-Уральского государственного университета. Серия "Математическое моделирование и программирование". — 2008. — № 27(127). — Вып. 2. — С. 110–116.
91. Шамакина. А.В. CAEBeans Broker: брокер ресурсов системы CAEBeans / А.В. Шамакина // Вестник ЮУрГУ. Серия "Математическое моделирование и программирование". — 2010. — № 16(192). — Вып. 5. — С. 107–115.
92. Савченко, Д.И. DiVTB Server: среда выполнения виртуальных экспериментов / Д.И. Савченко, Г.И. Радченко // Параллельные вычислительные технологии (ПаВТ'2013): Труды международной научной конференции (Челябинск, 1–5 апреля 2013 г.). — Челябинск: Издательский центр ЮУрГУ, 2013. — С. 532–539.
93. Радченко, Г.И. Распределенные виртуальные испытательные стенды: использование систем инженерного проектирования и анализа в распределенных вычислительных средах / Г.И. Радченко // Вестник Южно-Уральского государственного университета. Серия: Математическое моделирование и программирование. — 2011. — № 37. — С. 108–121.
94. Радченко, Г.И. Технология построения проблемно-ориентированных иерархических оболочек над инженерными пакетами в грид-средах / Г.И. Радченко // Системы управления и информационные технологии. — 2008. — № 4(34). — С. 57–61.
95. Радченко, Г.И. Технология построения виртуальных испытательных стендов в распределенных вычислительных средах / Г.И. Радченко, Л.Б. Соколинский // Научно-

технический вестник информационных технологий, механики и оптики. — 2008. — № 54. — С. 134–139.
96. Радченко, Г.И. Методы организации грид-оболочек системного слоя в технологии CAEBeans / Г.И. Радченко // Вестник Южно-Уральского государственного университета. Серия: Математическое моделирование и программирование. — 2008. — № 15. — С. 69–80.
97. Радченко, Г.И. Грид-система CAEBeans: интеграция ресурсов инженерных пакетов в распределенные вычислительные среды / Г.И. Радченко // Вестник Нижегородского университета им. Н.И. Лобачевского. — 2009. — № 6-1. — С. 192–202.
98. Savchenko, D. Mjolnirr: private PaaS as distributed computing evolution / D. Savchenko, G. Radchenko // MIPRO 2014: Proceedings of the 37th International Convention (Opatija, Croatia, May 26–30, 2014). — IEEE Computer Society, 2014. — P. 386–391.

Шамакина Анастасия Валерьевна, старший преподаватель кафедры системного программирования, Южно-Уральский государственный университет (Челябинск, Российская Федерация), shamakinaav@susu.ru.

Поступила в редакцию 5 августа 2014 г.

Bulletin of the South Ural State University
Series "Computational Mathematics and Software Engineering"
2014, vol. 3, no. 3, pp. 51–85

SURVEY ON DISTRIBUTED COMPUTING TECHNOLOGIES

A.V. Shamakina, South Ural State University (Chelyabinsk, Russian Federation)

This paper presents an overview of distributed computing technologies from the perspective of the scheduling organization in grid and cloud environments. The architecture of the UNICORE platform, focused on ensuring a transparent secure access to resources of distributed computing environment, is considered. A general classification of scheduling algorithms in distributed computing environments and a classification of scheduling algorithms for independent tasks and jobs with dependencies between tasks are shown.

Keywords: distributed computing, problem-oriented distributed computing environments, Grid, UNICORE, scheduling algorithms, clustering algorithms, a directed acyclic graph, workflow.

References

1. Buyya R., Abramson D. et al. Economic Models for Resource Management and Scheduling in Grid Computing // Journal of Concurrency and Computation: Practice and Experience. 2002. Vol. 14, Issue 13–15. P. 1507–1542.
2. Foster I., Roy A., Sander V. A Quality of Service Architecture That Combines Resource Reservation and Application Adaptation // Proceedings 8th Int. Workshop on Quality

of Service, Pittsburgh, PA, USA, June 5–7, 2000. Carnegie Mellon University P. 181–188.

3. Foster I., Kesselman C., Tuecke S. The Anatomy of the Grid: Enabling Scalable Virtual Organizations // International Journal of Supercomputer Applications and High Performance Computing. 2001. Vol. 15, No 3. P. 200–222.
4. Foster I., Kesselman C. The Grid. Blueprint for a new computing infrastructure. San Francisco: Morgan Kaufman, 1999. 677 p.
5. Jennings R. Cloud Computing with the Windows Azure Platform. Indianapolis, Indiana: Wiley Publishing, Inc., 2009. 360 p.
6. Marshall, P., Keahey K., Freeman, T. Improving Utilization of Infra–structure Clouds // Proceedings of the IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, Newport Beach, CA, USA. May 23–26, 2011. IEEE Computer Society, 2011. P. 205–214.
7. NIST Special Publication 800–145. A NIST Definition of Cloud Computing. URL: http://csrc.nist.gov/publications/nistpubs/800–145/SP800–145.pdf (дата обращения: 01.02.2014).
8. Sanderson D. Programming Google App Engine: Build and Run Scalable Web Apps on Google`s Infrastructure. O'Reilly Media, 2009. 400 p.
9. Dong F, Akl S.G. Scheduling algorithms for grid computing: State of the art and open problems. Technical Report No. 2006–504, Queen's University, Canada, 2006. P. 55.
10. Berman F., Wolski R. et al. Application–Level Scheduling on Distributed Heterogeneous Networks. // Proceedings of the ACM/IEEE conference on Supercomputing, Pittsburgh, Pennsylvania USA, May 25–28, 1996. IEEE Computer Society, 1996. P. 39–39.
11. Iverson M., Ozguner F. Dynamic, Competitive Scheduling of Multiple DAGs in a Distributed Heterogeneous Environment // Proceedings of Seventh Heterogeneous Computing Workshop, Orlando, Florida USA, March 30, 1998. IEEE Computer Society, 1998. P. 70–78.
12. Maheswaran M., Ali S. et al. Dynamic Matching and Scheduling of a Class of Independent Tasks onto Heterogeneous Computing Systems // Journal of Parallel and Distributed Computing. 1999. Vol. 59, No. 2. P. 107–131.
13. Zhu Y. A Survey on Grid Scheduling Systems. Technical Report SJTU_CS_TR_200309001. Department of Computer Science and Engineering, Shanghai Jiao. 2013. URL: http://www.cs.sjtu.edu.cn/~yzhu/reports/SJTU_CS_TR_200309001.pdf (дата обращения: 01.08.2014).
14. Belhajjame K., Vargas–Solar G., Collet C. A flexible workflow model for process–oriented applications // Proceedings of the Second International Conference on Web Information Systems Engineering (WISE'01), Kyoto, Japan, December 3–6, 2001. IEEE Computer Society. Vol. 1, No. 1. P. 72–80.
15. Condor. URL: http://www.cs.wisc.edu/condor (дата обращения: 04.06.2014).
16. Laszewski G., Foster I. et al. CoG Kits: A Bridge between Commodity Distributed Computing and High–Performance Grids // Proceedings of the ACM Java Grande 2000 Conference, CA, USA, June 3–5, 2000. ACM Press. P. 97–106.
17. Deelman E., Blythe J. et al. Pegasus: Mapping Scientific Workflows onto the Grid // Proceedings of Grid Computing: Second European AcrossGrids Conference (AxGrids 2004), Nicosia, Cyprus, January 28–30, 2004. Springer. P. 11–26.

18. Cao J., Jarvis S. A. et al. GridFlow: Workflow Management for Grid Computing // Proceedings of the 3rd International Symposium on Cluster Computing and the Grid (CCGrid'03), Tokyo, Japan, May 12–15, 2003. IEEE Computer Society, 2003. P. 198–205.
19. Wieczorek M., Prodan R., Fahringer T. Scheduling of Scientific Workflows in the ASKALON Grid Environment // ACM SIGMOD Record. 2005. Vol. 34, No. 3. P. 56–62.
20. Kim S.J. A general approach to multiprocessor scheduling. Report TR-88-04. Department of Computer Science, University of Texas at Austin. 1988. URL: ftp://ftp.cs.utexas.edu/pub/techreports/tr88-04.pdf (дата обращения: 01.08.2014).
21. Yang T., Gerasoulis A. DSC: Scheduling Parallel Tasks on an Unbounded Number of Processors // IEEE Transactions on Parallel and Distributed Systems. 1994. Vol. 5, No. 9. P. 951–967.
22. Baker M., Buyya R., Laforenza D. Grids and Grid Technologies for Wide–area Distributed Computing // Journal of Software-Practice & Experience. 2002. Vol. 32, No. 15. P. 1437–1466.
23. Schopf J. Ten Actions When SuperScheduling, document of Scheduling Working Group, Global Grid Forum. URL: http://www.ggf.org/documents/GFD.4.pdf (дата обращения: 12.06.2014).
24. Berman F., Wolski R. et al. Adaptive Computing on the Grid Using AppLeS // IEEE Transactions on Parallel and Distributed Systems. 2003. Vol. 14, No. 4. P. 369–382.
25. Hamscher V., Schwiegelshohn U. et al. Evaluation of Job–Scheduling Strategies for Grid Computing // Proceedings of GRID 2000 GRID 2000, First IEEE/ACM International Workshop, Bangalore, India, December 17, 2000. IEEE Computer Society. P. 191–202.
26. Mateescu G. Quality of Service on the Grid via Metascheduling with Resource Co–Scheduling and Co–Reservation // International Journal of High Performance Computing Applications. 2003. Vol. 17, No. 3. P. 209–218.
27. The Globus Toolkit. URL: http://www.globus.org (дата обращения: 04.01.2013).
28. Czajkowski K., Fitzgerald S. et al. Grid Information Services for Distributed Resource Sharing // Proceedings the 10th IEEE International Symposium on High– Performance Distributed Computing (HPDC–10), San Francisco, California, USA, August 7–9, 2001. IEEE Computer Society, 2001. P. 181–194.
29. Khokhar A.A., Prasanna V.K. et al. Heterogeneous Computing: Challenges and Opportunities // IEEE Computer. 1993. Vol. 26, No. 6. P. 18–27.
30. Siegel H. J., Dietz H. G., Antonio J. K., Software Support for Heterogeneous Computing // ACM Computing Surveys. 1996. Vol. 28, No. 1. P. 237–239.
31. Cooper K., Dasgupta A. et al. New Grid Scheduling and Rescheduling Methods in the GrADS Project // Proceedings of the 18th International Parallel and Distributed Processing Symposium (IPDPS'04), Santa Fe, New Mexico USA, April 26–30, 2004. IEEE Computer Society, 2004. P. 199–206.
32. Czajkowski K., Foster I. et al. A Resource Management Architecture for Metacomputing Systems // Proceedings of the 4th Workshop on Job Scheduling Strategies for Parallel Processing, Orlando, Florida USA, March 30, 1998. Springer, Lecture Notes in Computer Science, 1998. Vol. 1459. P. 62–82.

33. OpenPBS. URL: http://www.openpbs.org (дата обращения: 14.12.2013).
34. Condor. URL: http://www.cs.wisc.edu/condor (дата обращения: 04.12.2013).
35. Wolski R., Spring N.T., Hayes J. The Network Weather Service: A Distributed Resource Performance Forecasting Service for Metacomputing // Future Generation Computing Systems. 1999. Vol. 15, No. 5–6. P. 757–768.
36. Sacerdoti F.D., Katz M.J. et al. Wide area cluster monitoring with Ganglia // Proceedings of IEEE International Conference on Cluster Computing, Hong Kong, December 1–4, 2003. IEEE Computer Society, 2003. P. 289–298.
37. NIST Special Publication 800–145. A NIST Definition of Cloud Computing. URL: http://csrc.nist.gov/publications/nistpubs/800–145/SP800–145.pdf (дата обращения: 01.02.2012).
38. A. Streit. UNICORE: Getting to the heart of Grid technologies // eStrategies. Vol. 3. 2009. P. 8–9.
39. A. Streit. UNICORE – What lies beneath Grid functionality? // eStrategies. Vol. 7. 2008. P. 38–39.
40. Casavant T., Kuhl J. A Taxonomy of Scheduling in General–purpose Distributed Computing Systems // IEEE Transactions on Software Engineering. 1988.Vol. 14, No. 2. P. 141–154.
41. Braun R., Siegel H. et al. A Comparison of Eleven Static Heuristics for Mapping a Class of Independent Tasks onto Heterogeneous Distributed Computing Systems // Journal of Parallel and Distributed Computing. 2001. Vol. 61, No. 6, P. 810–837.
42. Casanova H., Legrand A. et al. Heuristics for Scheduling Parameter Sweep Applications in Grid Environments // Proceedings of the 9th heterogeneous Computing Workshop (HCW'00), Cancun, Mexico, May 1, 2000. IEEE Computer Society, 2000. P. 349–363.
43. You S.Y., Kim H.Y. et al. Task Scheduling Algorithm in GRID Considering Heterogeneous Environment // Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA '04), Nevada, USA, June 21–24, 2004. CSREA Press, 2004. P. 240–245.
44. Kurowski K., Ludwiczak B. et al. Improving Grid Level Throughput Using Job Migration And Rescheduling // Scientific Programming. 2004. Vol. 12, No. 4. P. 263–273.
45. Takefusa A., Matsuoka S. et al. A Study of Deadline Scheduling for Client–Server Systems on the Computational Grid // Proceedings of the 10th IEEE International Symposium on High Performance Distributed Computing (HPDC–10'01), San Francisco, California USA, August 7–9, 2001. IEEE Computer Society, 2001. P. 406–415.
46. Chen H., Maheswaran M. Distributed Dynamic Scheduling of Composite Tasks on Grid Computing Systems // Proceedings of the 16th International Parallel and Distributed Processing Symposium (IPDPS 2002), Fort Lauderdale, Florida USA, April 15-19, 2002. IEEE Computer Society, 2002. P. 88–97.
47. Muthuvelu N., Liu J. et al. A Dynamic Job Grouping-Based Scheduling for Deploying Applications with Fine–Grained Tasks on Global Grids // Proceedings of the 3rd Australasian Workshop on Grid Computing and e–Research (AusGrid 2005), Newcastle, Australia, January 30 – February 4, 2005. Australian Computer Society, 2005. — P. 41–48.

48. Wright D. Cheap Cycles from the Desktop to the Dedicated Cluster: Combining Opportunistic and Dedicated Scheduling with Condor. URL: http://www.linuxclustersinstitute.org/conferences/archive/2001/PDF/wright_wisc.pdf (дата обращения: 04.08.2014).
49. Chapin S. J., Katramatos D. et al. The Legion Resource Management System // Proceedings of the 5th Workshop on Job Scheduling Strategies for Parallel Processing (JSSPP '99), San Juan, Puerto Rico, April 16, 1999. Springer, Lecture Notes in Computer Science, 1999. Vol. 1659. P. 162–178.
50. Rotithor H. G. Taxonomy of Dynamic Task Scheduling Schemes in Distributed Computing Systems // Proceedings on Computer and Digital Techniques, January 1994. Vol. 141, No. 1. P. 1–10.
51. James H. A. Scheduling in Metacomputing Systems, Ph.D. Thesis. The Department Of Computer Science, University of Adelaide, Australia. 1999. URL: http://digital.library.adelaide.edu.au/dspace/bitstream/2440/19450/1/09phj274.pdf (дата обращения: 04.08.2014).
52. Aggarwal A. K., Kent R. D. An Adaptive Generalized Scheduler for Grid Applications // Proceedings of the 19th Annual International Symposium on High Performance Computing Systems and Applications (HPCS'05). Guelph, Ontario Canada, May 15–18, 2005). IEEE Computer Society, 2005. P. 15–18.
53. Sabin G., Kettimuthu R. et al. Scheduling of Parallel Jobs in a Heterogeneous Multi–Site Environment // Proceedings of the 9th International Workshop on Job Scheduling Strategies for Parallel Processing (JSSPP'03), Seattle, WA, USA, June 24, 2003). Springer, Lecture Notes in Computer Science, 2003. Vol. 2862. P. 87–104.
54. Arora M., Das S.K., Biswas R. A Decentralized Scheduling and Load Balancing Algorithm for Heterogeneous Grid Environments // Proceedings of International Conference on Parallel Processing Workshops (ICPPW'02). Vancouver, British Columbia Canada, August 20-23, 2002). IEEE Computer Society, 2002. P. 499–505.
55. El-Rewini H., Lewis T., Ali H. Task Scheduling in Parallel and Distributed Systems. Prentice Hall, 1994. 290 p.
56. Buyya R., Abramson D., Venugopal S. The Grid Economy // Proceedings of the IEEE. 2005. Vol. 93, No. 3. P. 698–714.
57. Buyya R., Giddy J., Abramson D. An Evaluation of Economy–based Resource Trading and Scheduling on Computational Power Grids for Parameter Sweep Applications // Proceedings of the 2nd International Workshop on Active Middleware Services (AMS 2000), Pittsburgh, USA, August 1, 2000. Kluwer Academic Press, 2000. Vol. 583. P. 221–230.
58. Yu J., Buyya R., Tham C.K. QoS–based Scheduling of Workflow Applications on Service Grids // Proceedings of the 1st IEEE International Conference on e–Science and Grid Computing (e–Science'05), Melbourne, Australia, December 5–8, 2005. IEEE Computer Society, 2005. P. 1–9.
59. Venugopal S., Buyya R. A Deadline and Budget Constrained Scheduling Algorithm for eScience Applications on Data Grids // Proceedings of 6th International Conference on Algorithms and Architectures for Parallel Processing (ICA3PP–2005), Melbourne, Australia, October 2-3, 2005. Springer, Lecture Notes in Computer Science, 2005. P. 60–72.

60. Ernemann C., Hamscher V., Yahyapour R. Economic Scheduling in Grid Computing // Proceedings of 8th Workshop on Job Scheduling Strategies for Parallel Processing, in conjunction with HPDC/GGF 5, Edinburgh, Scotland, UK, July 24, 2002. Springer, Lecture Notes in Computer Science, 2002. P. 128–152.
61. Zhu Y., Xiao L. et al. Incentive–based P2P Scheduling in Grid Computing // Proceedings of the 3rd International Conference on Grid and Cooperative Computing (GCC2004), Wuhan, China, October 21-24, 2004. Springer, Lecture Notes in Computer Science, 2004. P. 209–216.
62. Young L., McGough S. et al. Scheduling Architecture and Algorithms within the ICENI Grid Middleware // Proceedings of UK e–Science All Hands Meeting, Nottingham, UK, September 2003). IOP Publishing Ltd., 2003. P. 5–12.
63. Kim S., Weissman J.B. A Genetic Algorithm Based Approach for Scheduling Decomposable Data Grid Applications // Proceedings of the 2004 International Conference on Parallel Processing (ICPP'04), Montreal, Quebec Canada, August 15–18, 2004). IEEE Computer Society, 2004. P. 406–413.
64. Song S., Kwok Y., Hwang K. Security–Driven Heuristics and A Fast Genetic Algorithm for Trusted Grid Job Scheduling // Proceedings of 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05), Denver, Colorado USA, April 25–29, 2005. IEEE Computer Society, 2005. P. 65–74.
65. Spooner D.P., Cao J. et al. Localised Workload Management using Performance Prediction and QoS Contracts // UK Performance Engineering Workshop (UKPEW 2002): Proceedings of the 18th Annual Workshop, Glasgow, UK, July 10–11, 2002. University of Glasgow, 2002. P. 69–80.
66. Liu Y. Survey on Grid Scheduling. Department of Computer Science, University of Iowa. 2004.
67. Abraham A., Buyya R. and Nath B. Nature's Heuristics for Scheduling Jobs on Computational Grids // Proceedings of 8th IEEE International Conference on Advanced Computing and Communications (ADCOM 2000), Cochin, India, December 14–16, 2000. IEEE Computer Society, 2000. P. 45–52.
68. Shan H., Oliker L. et al. Scheduling in Heterogeneous Grid Environments: The Effects of Data Migration // Proceedings of ADCOM2004: International Conference on Advanced Computing and Communication, Ahmedabad Gujarat, India, December 15–18, 2004. IEEE Computer Society, 2004. P. 1–8.
69. He X., Sun X., Laszewski G. A QoS Guided Min–Min Heuristic for Grid Task Scheduling // Journal of Computer Science and Technology: Special Issue on Grid Computing. 2003. Vol. 18, No. 4. P. 442–451.
70. Wu M., Shu W., Zhang H. Segmented Min–Min: A Static Mapping Algorithm for Meta–Tasks on Heterogeneous Computing Systems // Proceedings of the 9th Heterogeneous Computing Workshop (HCW'00), Cancun, Mexico, May 1, 2000. IEEE Computer Society, 2000. P. 375–385.
71. Subramani V., Kettimuthu R. et al. Distributed Job Scheduling on Computational Grids using Multiple Simultaneous Requests // Proceedings of 11th IEEE Symposium on High Performance Distributed Computing (HPDC 2002), Edinburgh, Scotland, July 23–26, 2002. IEEE Computer Society, 2002. P. 359–366.

72. Silva D.P., Cirne W., Brasileiro F.V. Trading Cycles for Information: Using Replication to Schedule Bag-of-Tasks Applications on Computational Grids // Proceedings of Euro–Par 2003, Klagenfurt, Austria, August 26–29, 2003. Springer, Lecture Notes in Computer Science, 2003. P. 169–180.
73. Radulescu A., Gemund A.J.C. On the Complexity of List Scheduling Algorithms for Distributed Memory Systems // Portland, Oregon, USA, November 13–19, 1999). IEEE Computer Society, 1999. P. 68–75.
74. Sakellariou R., Zhao H. A Low-cost Rescheduling Policy for Efficient Mapping of Workflows on Grid Systems // Scientific Programming. 2004. Vol. 12, No. 4. P. 253–262.
75. Topcuoglu H., Hariri S., Wu M.Y. Performance–Effective and Low–Complexity Task Scheduling for Heterogeneous Computing // IEEE Transactions on Parallel and Distributed Systems. 2002. Vol. 13, No. 3. P. 260–274.
76. Zhao H., Sakellariou R. An Experimental Investigation into the Rank Function of the Heterogeneous Earliest Finish Time Scheduling Algorithm // Proceedings of Euro–Par 2003, Klagenfurt, Austria, August 26–29, 2003. Springer, Lecture Notes in Computer Science, 2003. P. 189–194.
77. Sakellariou R., Zhao H. A Hybrid Heuristic for DAG Scheduling on Heterogeneous Systems // Proceedings of 18th International Parallel and Distributed Processing Symposium (IPDPS'04), Santa Fe, New Mexico USA, April 26–30, 2004. IEEE Computer Society, 2004. P. 111–123.
78. Ma T., Buyya R. Critical–Path and Priority based Algorithms for Scheduling Workflows with Parameter Sweep Tasks on Global Grids // Proceedings of the 17th International Symposium on Computer Architecture and High Performance Computing, Rio de Janeiro, Brazil, October 24–27, 2005. IEEE Computer Society, 2005. P. 251–258.
79. Darbha S., Agrawal D.P. Optimal Scheduling Algorithm for Distributed Memory Machines // IEEE Transactions on Parallel and Distributed Systems. January 1998. Vol. 9, No. 1. P. 87–95.
80. Ranaweera S., Agrawal D.P. A Task Duplication Based Scheduling Algorithm for Heterogeneous Systems // Proceedings of 14th International Parallel and Distributed Processing Symposium (IPDPS'00), Cancun, Mexico, May 1–5, 2000. IEEE Computer Society, 2005. P. 445–450.
81. Bajaj R., Agrawal D. P. Improving Scheduling of Tasks in A Heterogeneous Environment // IEEE Transactions on Parallel and Distributed Systems. 2004. Vol. 15, No. 2. P. 107–118.
82. Gerasoulis A., Yang T. A comparison of clustering heuristics for scheduling directed acyclic graphs on multiprocessors // Journal of Parallel and Distributed Computing, 1992. Vol. 16, No. 4. P. 276–291.
83. Liou J., Palis M.A. An Efficient Task Clustering Heuristic for Scheduling DAGs on Multiprocessors // Proceedings of the Eighth IEEE Symposium on Parallel and Distributed Processing (SPDP 1996), New Orleans, Louisiana, USA, October 23–26, 1996. IEEE Computer Society, 1996. P. 152–156.
84. Liou J., Palis M.A. A Comparison of General Approaches to Multiprocessor Scheduling // Proceedings the 11th International Parallel Processing Symposium (IPPS '97), Geneva, Switzerland, April 1–5, 1997. IEEE Computer Society, 1996. P. 152–156.

85. Kim S.J., Browne J.C. A general approach to mapping of parallel computation upon multiprocessor architectures // Proceedings of the International Conference on Parallel Processing (ICPP'88), The Pennsylvania State University, University Park, PA, USA, August 1988). Pennsylvania State University Press., 1988. Vol. 3. P. 1–8.
86. Sarkar V. Partitioning and Scheduling Parallel Programs for Execution on Multiprocessors. Cambridge, MA: The MIT Press, 1989. P. 215.
87. Shamakina A.V., Sokolinsky L.B. Formal'naja model' zadanija v raspredelennyh vychislitel'nyh sredah [A mathematical model of a job in distributed computing environments]. Parallelnye vychislitelnye tekhnologii (PaVT'2014): Trudy mezhdunarodnoj nauchnoj konferentsii (Rostov-na-Donu, 1–3 aprelya 2014) [Parallel Computational Technologies (PCT'2010): Proceedings of the International Scientific Conference (Rostov-on-Don, Russia, April 1–3, 2014)]. Chelyabinsk, Publishing of the South Ural State University, 2014. P. 343–354.
88. Radchenko G.I., Sokolinsky L.B., Shamakina A.V. Modeli i metody profilirovanija i ocenki vremeni vypolnenija potokov rabot v superkomp'juternyh sistemah [Models and methods of profiling and evaluation of workflow runtime in supercomputing systems] // Vychislitel'nye metody i programmirovanie: Novye vychislitel'nye tehnologii [Numerical Methods and Programming: New computing technologies]. 2013. Vol. 14, No. 4. P. 96–103.
89. Shamakina A.V. Broker resursov dlja podderzhki problemno-orientirovannyh sred [Brokering service for supporting problem-oriented grid environments] // Vestnik JUUrGU. Serija "Vychislitel'naja matematika i informatika" [Bulletin of the South Ural State University. Series "Computational Mathematics and Software Engineering"]. 2012. Vol. 46(305). No. 1. P. 88–98.
90. Shamakina A.V. Organizacija brokera resursov v sisteme CAEBeans [Organization of a resource broker in the CAEBeans system] // Vestnik JUzhno-Ural'skogo gosudarstvennogo universiteta. Serija "Matematicheskoe modelirovanie i programmirovanie" [Bulletin of the South Ural State University. The series "Mathematical Modelling, Programming and Computer Software"]. 2008. Vol. 27(127). No. 2. P. 110–116.
91. Shamakina A.V. CAEBeans Broker: broker resursov sistemy CAEBeans [CAEBeans Broker: a resource broker of the CAEBeans system] // Vestnik JUzhno-Ural'skogo gosudarstvennogo universiteta. Serija "Matematicheskoe modelirovanie i programmirovanie" [Bulletin of the South Ural State University. The series "Mathematical Modelling, Programming and Computer Software"]. 2010. Vol 16(192). No. 5. P. 107–115.
92. Savchenko D.I., Radchenko G.I. DiVTB Server: sreda vypolnenija virtual'nyh jeksperimentov [DiVTB Server: a runtime environment for virtual experiments]. Parallelnye vychislitelnye tekhnologii (PaVT'2013): Trudy mezhdunarodnoj nauchnoj konferentsii (Cheljabinsk, 1–5 aprelya 2013) [Parallel Computational Technologies (PCT'2010): Proceedings of the International Scientific Conference (Chelyabinsk, Russia, April 1–5, 2013)]. Chelyabinsk, Publishing of the South Ural State University, 2013. P. 532–539.
93. Radchenko G.I. Raspredelennye virtual'nye ispytatel'nye stendy: ispol'zovanie sistem inzhenernogo proektirovanija i analiza v raspredelennyh vychislitel'nyh sredah [Distributed virtual test beds: the use of CAE-systems in distributed computing environments] // Vestnik JUzhno-Ural'skogo gosudarstvennogo universiteta. Serija "Matematicheskoe modelirovanie i programmirovanie" [Bulletin of the South Ural State University. The

series "Mathematical Modelling, Programming and Computer Software"]. 2011. Vol. 37. P. 108–121.

94. Radchenko G.I. Tehnologija postroenija problemno-orientirovannyh ierarhicheskih obolochek nad inzhenernymi paketami v grid-sredah [Technology of building problem-oriented hierarchical shells over engineering packages in Grid environments] // Sistemy upravlenija i informacionnye tehnologii [Management systems and information technologies]. 2008. Vol. 4(34). P. 57–61.
95. Radchenko G.I., Sokolinsky L.B. Tehnologija postroenija virtual'nyh ispytatel'nyh stendov v raspredelennyh vychislitel'nyh sredah [Technology of building virtual test beds in distributed computing environments] // Nauchno-Tehnicheskii Vestnik Informatsionnykh Tekhnologii, Mekhaniki i Optiki [Scientific and Technical Journal of Information Technologies, Mechanics and Optics]. 2008. Vol. 54. P. 134–139.
96. Radchenko G.I. Metody organizacii grid-obolochek sistemnogo sloja v tehnologii CAEBeans [Methods of organizing grid shells of a system layer in the CAEBeans technology] // Vestnik JUzhno-Ural'skogo gosudarstvennogo universiteta. Serija "Matematicheskoe modelirovanie i programmirovanie" [Bulletin of the South Ural State University. The series "Mathematical Modelling, Programming and Computer Software"]. 2008. Vol. 15. P. 69–80.
97. Radchenko G.I. Grid-sistema CAEBeans: integracija resursov inzhenernyh paketov v raspredelennye vychislitel'nye sredy [The CAEBeans grid system: an integration of resources of CAE-package in distributed computing environments] // Vestnik Nizhegorodskogo universiteta im. N.I. Lobachevskogo [Bulletin of the Nizhny Novgorod University. N.I. Lobachevsky]. 2009. Vol. 6-1. P. 192–202.
98. Savchenko D., Radchenko G. Mjolnirr: private PaaS as distributed computing evolution // MIPRO 2014: Proceedings of the 37th International Convention (Opatija, Croatia, May 26–30, 2014). IEEE Computer Society, 2014. P. 386–391.

Received 5 August 2014.