УДК 004.75: 004.414.23

В.В. КАЗИМИР[*], П.Г. БИВОЙНО[**], О.А. ПРЕЛАЯ[**], Т.А. ГУЗА[**]

МЕТОДЫ ПЛАНИРОВАНИЯ ПОТОКОВ ЗАДАЧ В GRID-СРЕДЕ

[*]Черниговский государственный институт экономики и управления, Чернигов, Украина
[**]Черниговский национальный технологический университет, Чернигов, Украина

Анотація. *Розглянуто особливості використання grid-середовища для вирішення різних типів обчислювальних завдань. Представлені результати дослідження існуючих методів планування завдань. Запропоновано алгоритм оптимізації виконання потоків завдань в grid-середовищі на базі методу динамічного програмування. Представлені результати експериментального дослідження ефективності алгоритмів планування на базі комплексу імітаційних моделей.*
Ключові слова: *grid, поток задач, планирование, оптимизация, качество обслуживания.*

Аннотация. *Рассмотрены особенности использования grid-среды для решения разных типов вычислительных задач. Представлены результаты исследования существующих методов планирования задач. Предложен алгоритм оптимизации выполнения потоков задач в grid-среде на базе метода динамического программирования. Представлены результаты экспериментального исследования эффективности алгоритмов планирования на базе комплекса имитационных моделей.*
Ключевые слова: *grid, потік завдань, планування, оптимізація, якість обслуговування.*

Abstract. *The features of grid-environment usage for different computational problems solving are described in the paper. The results of the traditional grid scheduling methods are presented. An algorithm for workflow scheduling problems optimization in grid-environment based on the dynamic programming approach has been proposed. Experimental results on scheduling algorithms efficiency based on complex of simulation models are given.*
Keywords: *grid, workflow, scheduling, optimization, quality of service.*

1. Введение

Активно развивающимся направлением в области grid-вычислений является разработка технологий использования grid-инфраструктуры для решения вычислительных задач большой размерности, представленных в виде потока работ (workflow) – последовательно-параллельных подзадач с определенной схемой синхронизации вычислений. Наличие параллельных блоков обусловливает возможность одновременного использования нескольких ресурсов grid-сети. При этом важным является минимизация затрат на передачу данных между вычислительными блоками и предоставление требуемого пользователем уровня качества обслуживания (QoS). Планирование потоков работ является NP-полной задачей в общем виде [1].

Целью работы является исследование методов планирования потоков работ в grid-среде и оценка их эффективности на базе комплекса имитационных моделей в соответствии с критериями, которые сформулированы в работе.

2. Формализация задачи оптимизации выполнения потока работ в grid-среде

Одним из факторов, влияющих на производительность grid-сети, является эффективность планирования. Учитывая гетерогенность ресурсов grid-сети, а также структурные особенности решаемых задач, под эффективностью планирования следует понимать следующие факторы:

1) равномерная загруженность всех узлов вычислительной сети;
2) минимальное время простоя задач в очереди на выполнение;

3) минимальное время выполнения задач на выделенном наборе ресурсов, в том числе время, требуемое для пересылки данных между вычислительными блоками.

Авторами предложена классификация задач, решаемых в grid-среде, согласно критерию структурных особенностей задачи [2].

Эффективность выполнения задачи, представленной единым вычислительным блоком или набором последовательных, зависит от эффективности ее программной реализации, стратегии планирования низкоуровневых брокеров grid и локального планировщика.

В случае, если задача представлена в виде набора однотипных задач с разными входными данными, планирование сводится к оптимизации декомпозиции задачи с учетом текущих параметров доступной grid-инфраструктуры.

Наличие параллельных блоков задачи, представленной в виде потока работ, позволяет одновременно использовать несколько распределенных ресурсов для более эффективного решения задачи. При планировании потоков задач должны учитываться расходы на пересылку данных между ресурсами в соответствии с пропускной способностью сети. Расходы на пересылку данных могут быть устранены путем кластеризации нескольких блоков потока работ для решения на одном ресурсе. Существует понятие линейной и нелинейной кластеризации [1], когда группируются последовательные или параллельные блоки соответственно. Задача оптимизации сводится к нахождению оптимального решения между распараллеливанием и кластеризацией.

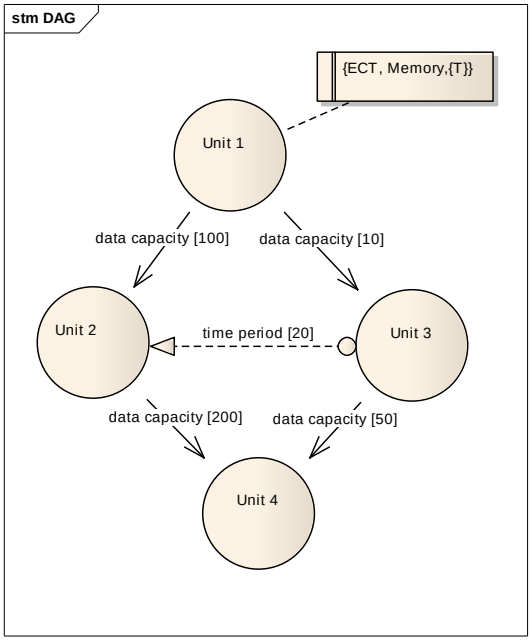

Рис. 1. Пример структуры задачи типа поток работ

Задача типа поток работ может быть представлена в виде направленного ациклического графа (DAG) [1], узлы (блоки) которого представляют вычислительные подзадачи, а дуги – зависимости между ними (рис. 1). Для эффективного планирования должны быть определены следующие параметры блока:

$$\{ECT, Memory, \{T\}\},$$

где ECT – прогнозируемое время вычислений, Memory – требования к памяти, {T} – множество связей с другими вычислительными блоками (связь однонаправленная).

Дуги графа характеризуются параметром объема передаваемых между узлами данных. Связь между блоками Unit2 и Unit3 определяет необходимость периодической синхронизации данных между параллельно выполняющимися блоками.

Отдельно следует выделить комбинированный тип задач, когда задаче присущи характеристики нескольких типов, и частный случай задачи типа поток работ, когда задача декомпозируется на полностью независимые параллельные подзадачи. В последнем случае процесс адаптации к grid значительно упрощается, поскольку синхронизация между блоками вычислений не требуется.

Для эффективного решения в grid-среде на структуру задачи типа поток работ накладываются следующие ограничения.

1. Отсутствие циклов и ветвлений. Указанные ограничения определяются формой представления структуры задачи в виде ациклического направленного графа. Однако структура задачи, имеющей циклическую связь, может быть преобразована в DAG

посредством добавления дополнительного уровня. Ветвления могут обрабатываться на уровне метапланировщика посредством динамического подхода к планированию.

2. Высокий уровень гранулярности задачи. Размерность вычислений должна быть значительно больше по отношению к размерности пересылаемых данных [1]. В [3] гранулярность задачи определяется следующим образом:

$$g = \min_{x=1:\nu}\{\tau_x / \max_j\{c_{x,j}\}\}, \quad (1)$$

где τ_x – вычислительная сложность узла n_x;

$c_{x,j}$ – размерность пересылаемых данных между узлами n_x и n_j;

ν – количество вычислительных узлов задачи.

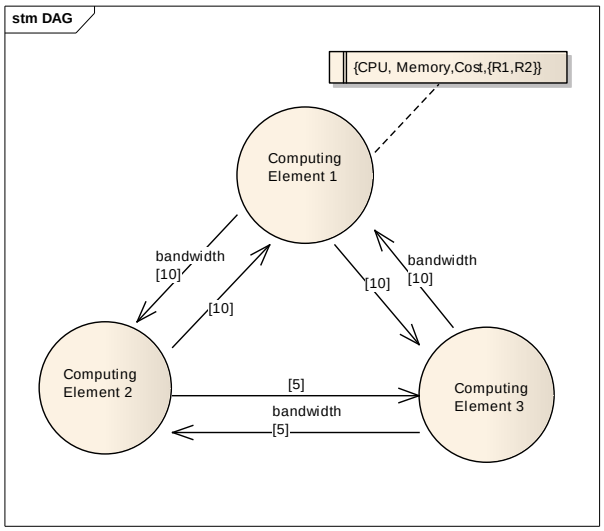

Рис. 2. Пример структуры grid-сети

Структура grid-сети может быть представлена в виде полного направленного графа, вершины которого – ресурсы, а веса дуг задают пропускную способность вычислительной сети (рис. 2). Каждый узел структуры grid-сети характеризуется следующими обязательными параметрами:

{CPU, Memory, Cost, {R1, R2}},

где CPU – вычислительная мощность, Memory – характеристики памяти, Cost – стоимость использования, R1 – пропускная способность сети на получение и R2 – на передачу данных соответственно.

Задача оптимизации заключается в нахождении оптимального варианта размещения потока работ на доступном множестве ресурсов.

В [4] представлена классификация задач планирования согласно следующим критериям:

$$\{\alpha \mid \beta \mid \gamma \mid \delta\}, \quad \beta = \{\beta_1 \mid \beta_2 \mid \beta_3\}, \quad (2)$$

где α – определяет характеристики распределенной среды (гомогенная/гетерогенная);

β – характеристики задачи и наличие ограничений в структуре задачи;

β_1 – наличие зависимостей между вычислительными блоками задачи;

β_2 – однородность/разнородность вычислительных блоков задачи;

β_3 – наличие ограничений по времени начала вычислений блока задачи вне зависимости от результата выполнения связанных с ним блоков;

γ – определяет критерий оптимизации и вид целевой функции;

δ – определяет функцию затрат на взаимодействие между ресурсами распределенной среды при выполнении задачи.

Согласно приведенной классификации, задача планирования в grid-среде определяется следующим образом:

$$\{R \mid PREC, \varnothing,\ r_i \mid C_{\max} \mid JP\}, \quad (3)$$

где R – определяет гетерогенность ресурсов распределенной среды; время выполнения вычислительного блока задачи является функцией мощности узла распределенной среды;

$PREC$ – наличие зависимостей между вычислительными блоками задачи;

$\varnothing$ – блоки задачи имеют разную вычислительную сложность;

r_i – имеются ограничения по времени начала вычислений блока задачи;

$C_{\max}$ – целью задачи планирования является минимизация времени выполнения задачи;

JP – затраты на взаимодействие между узлами распределенной среды определяются как параметрами пропускной способности сети, так и характеристиками задачи (объемы пересылаемых данных между вычислительными блоками).

Следует отметить, что предложенная классификация не учитывает многокритериальность целевой функции. Однако для коммерческой grid-среды, помимо времени выполнения задачи, важным является учет конкурирующего критерия – стоимости вычислений. Задача планирования потока работ в grid-среде является NP-полной задачей в общем случае [1].

На уровне промежуточного программного обеспечения (ППО) grid не предоставляет полноценной поддержки grid – задач разного типа. Например, ARC Nordugrid (http://www.nordugrid.org/) и gLite (http://glite.cern.ch/), которые включены в качестве одних из основных поставщиков ППО grid в EMI (http://www.eu-emi.eu/) и наиболее широко используются в украинской национальной grid-инфраструктуре, используют следующие форматы спецификации grid-задач: JSDL [5], xRSL [6] и JDL [7]. Среди указанных только JDL-формат вводит понятие типа задания (Job, DAG и Collection), но имеет определенные ограничения – не предоставляется возможность определения периодической синхронизации между блоками и объемами пересылаемых данных. Форматы JSDL и xRSL предоставляют только средства определения параметров отдельных задач, жизненный цикл потока работ, также, как и связи между отдельными задачами, не поддерживаются.

Брокеры ППО grid реализуют упрощенные стратегии планирования и не позволяют выполнять планирование потоков работ с учетом особенностей задачи и параметров QoS. Например, брокер ARC Nordugrid реализует следующие политики выбора доступных вычислительных ресурсов: RandomBroker – случайный выбор ресурса; BenchmarkBroker – согласно параметру производительности ресурса; FastestQueueBroker – согласно параметру размера очереди задач; DataBroker – согласно наличию в кэше вычислительного ресурса данных, необходимых для выполнения вычислений [6]. Следовательно, механизмы планирования сложных grid-задач должны быть реализованы и расположены поверх уровня ППО grid.

3. Структура метапланировщика системы управления потоками задач

Важной составляющей использования grid-среды является обеспечение требуемого пользователем уровня качества обслуживания (QoS).

Помимо поиска оптимального размещения потока задач на доступном множестве вычислительных ресурсов, важными аспектами работы метапланировщика являются: а) стратегия обработки входного потока задач с учетом их приоритетов; б) выбор алгоритма планирования согласно структурным особенностям задачи; в) контроль динамики выполнения задач; г) учет динамичности grid-сети, а также уровня качества обслуживания ресурсов grid-сети.

Приоритет задачи должен учитывать:

1) время постановки задачи в очередь;

2) запрашиваемый пользователем уровень QoS.

Возможность предварительного резервирования ресурсов [8] для grid-сайтов, которые поддерживают данную политику, должна обеспечиваться на уровне метапланировщика.

Существуют решения относительно порядка выборки, обработки и планирования множества задач, представленных в виде потока работ (MultipleWorkflow Scheduling) [9]. В работе [10] представлена структура координатора входного потока задач с учетом требуемого времени выполнения задачи, однако решение не является полным.

Под динамичностью grid-сети понимаются состояние и загруженность ресурсов сети, изменение локальных политик распределения ресурсов. Уровень качества обслуживания ресурсов grid-сети определяется следующими параметрами:

– общие данные (пропускная способность сети, вычислительная мощность узлов, объем дискового пространства и т.д.);

– неявные данные про уровень качества обслуживания (загруженность и частота отказа узлов, время простоя задач в очереди, соревнование за определенный ресурс на узле между независимыми задачами).

Ниже рассматриваются подходы к размещению задачи типа поток работ на доступном множестве гетерогенных grid-ресурсов без учета стратегии выбора задачи из входной очереди, а также динамики ее выполнения.

4. Методы планирования потока задач в grid-среде

В работах [1, 11, 12] приведены классификация и результаты исследований эффективности и сложности алгоритмов. Ниже представлена классификация алгоритмов планирования в grid-среде согласно обозначенным критериям.

1. Этап полного планирования задачи.

Планирование всех вычислительных блоков задачи может выполняться как до начала выполнения задачи (статический подход), так и в процессе выполнения задачи (динамический подход). Статический подход подразумевает наличие информации о текущем состоянии ресурсов grid-сети и последовательности выполнения блоков задачи до начала вычислений. Динамический подход позволяет учитывать динамичность ресурсов grid-сети, а также обрабатывать ветвления в структуре задачи. Однако это значительно усложняет процесс планирования.

Предлагается использование гибридного подхода к планированию, который заключается в применении статических методов для получения первичного распределения с последующим динамическим регулированием первичного распределения с учетом динамики выполнения задачи и состояния ресурсов сети. Подобная схема реализуется на уровне метапланировщика посредством периодического опроса состояния ресурсов сети, контроля выполнения блоков задачи и перепланирования задачи при необходимости.

2. Оптимальное / субоптимальное решение.

Для решения задачи планирования потока работ применяются следующие подходы: поиск решений в пространстве состояний, математическое программирование и эвристические подходы [1].

Методы поиска в пространстве состояний делятся на информированные и неинформированные. Неинформированные методы или методы “слепого” поиска (полный перебор, поиск в глубину, поиск в ширину, упорядоченный перебор и др.) не используют дополнительной информации о конкретной задаче. Они последовательно просматривают все состояния до тех пор, пока не будет найдено решение. Различия между методами неинформированного поиска сводятся к порядку просмотра состояний.

Информированные методы поиска (эвристические методы) пользуются дополнительной информацией о конкретной задаче, что позволяет сократить перебор путём исключения заведомо бесперспективных вариантов. Такой подход ускоряет работу алгоритма по сравнению с полным перебором.

Одним из эвристических алгоритмов является генетический алгоритм решения задач оптимизации путём случайного подбора, комбинирования и вариации искомых

параметров с использованием механизмов, аналогичных естественному отбору в природе [13].

Методы поиска в пространстве состояний и методы математического программирования могут давать оптимальные решения, однако в общем случае характеризуются высокой вычислительной сложностью алгоритма. С использованием эвристических подходов можно получить эффективные решения за полиномиальное время, однако в общем случае подобные подходы не позволяют получить оптимальные решения, поскольку средняя, худшая и лучшая производительность этих алгоритмов неизвестна [1].

Среди эвристических выделяют следующие подходы: индивидуальное планирование задач, планирование списков задач, кластеризация и дублирующее планирование [11].

Индивидуальное планирование (Min-Min, Min-Max, Sufferage [12]) может применяться только для последовательно выполняющихся задач, представленных единым вычислительным блоком.

Планирование на основе списков подразумевает определение приоритетов вычислительных блоков задачи и запуска для выполнения согласно установленным приоритетам. Может выполняться в следующих режимах: пакетном, когда все задачи считаются независимыми, и в режиме с учетом зависимостей. Последний подход позволяет учитывать затраты на передачу данных между узлами и уменьшить суммарное время выполнения потока работ, но приводит к увеличению сложности алгоритма. Примерами алгоритмов, основанных на планировании на основе списков с учетом зависимостей, являются Heterogeneous Earliest-Finish-Time (HEFT), Fast Critical Path (FCP) [12].

Согласно алгоритму HEFT, приоритет вычислительного блока определяется следующим образом:

$$rank(n_i) = \overline{\omega_i} + \max_{n_j \in succ(n_i)} (\overline{c_{i,j}} + rank(n_j)), \qquad (4)$$

где $\overline{\omega_i}$ – среднее время выполнения задачи на всех доступных ресурсах;

$\overline{c_{i,j}}$ – усредненная оценка затрат на пересылку данных между подзадачами n_i и n_j для всех пар ресурсов;

$succ(n_i)$ – множество всех задач, которые зависят от задачи n_i.

Таким образом, приоритет задачи зависит от приоритета всех зависимых от нее задач. Назначение задач на ресурсы происходит следующим образом: задача с наибольшим приоритетом при условии выполнения всех задач, от которых она зависит, назначается на вычислительный ресурс, обеспечивающий наименьшее время выполнения задачи. HEFT является одним из наиболее распространенных эвристических алгоритмов планирования, однако для задач со сложной структурой не гарантирует нахождение оптимального решения.

Кластерный (DSC, CASS-II [14]) и дублирующий (TDS, TANH [15]) подходы нацелены на уменьшение временных затрат на пересылку данных между узлами путем размещения задач, которые требуют обмена большого количества данных на одном ресурсе или дублирования блоков соответственно [12]. Недостатками данных подходов является сложность учета разнородности подзадач, а также отсутствие возможности одновременного использования нескольких ресурсов grid-сети для выполнения параллельных блоков задачи.

3. Критерий оптимизации.

Согласно критерию оптимизации, выделяют два направления в алгоритмах планирования задач типа поток работ: нацеленные на наилучшую производительность (минимизация времени выполнения без учета других факторов) и основанные на минимизации экономических затрат. Важным аспектом использования коммерческой grid-среды является необходимость оптимизации взаимоисключающих характеристик – стоимости ресурса и времени выполнения с учетом коэффициентов значимости каждой из характеристик.

Большинство эвристических алгоритмов планирования ориентированы на улучшение одного из критериев. На сегодняшний день единственным алгоритмом планирования потока задач, решающим многокритериальную задачу оптимизации, является алгоритм LOSS/GAIN [11]. Согласно алгоритму, задаются две функции: LOSS, которая дает некоторый результат планирования, такой, что общее время работы увеличивается, в то же время, уменьшая стоимость; GAIN – функция, которая дает результат планировки со временем выполнения меньшим, чем текущее, но с более высокой стоимостью. Работа алгоритма продолжается до тех пор, пока не будет достигнуто выполнение первого ограничения (минимальное время выполнения) с наиболее благоприятным значением второго (стоимость).

Многие из существующих алгоритмов планирования накладывают ряд ограничений относительно структуры задачи, структуры grid-сети и критериев оптимизации.

5. Применение метода динамического программирования для задачи планирования потоков работ в grid-среде

Метод динамического программирования является одним из методов математического программирования и применим к задачам с оптимальной подструктурой. Оптимальная подструктура задачи предполагает, что оптимальное решение составляющих ее подзадач меньшего размера может быть использовано для решения исходной задачи [16].

Алгоритм вводит понятие уровня в структуре задачи типа поток работ, который определяется множеством подзадачи, которые могут выполняться одновременно на определенном этапе выполнения задачи. Например, в структуре задачи, представленной на рис. 1, могут быть выделены следующие уровни: 1) {Unit 1}; 2) {Unit 2, Unit 3}; 3) {Unit 4}.

Оптимальное решение задачи содержит оптимальные решения на каждом уровне, а, значит, задача обладает свойством оптимальности [16].

Целевая функция алгоритма может определяться несколькими параметрами, имеющими определенные веса.

Блок-схема работы алгоритма представлена на рис. 3.

Как следует из блок-схемы алгоритма, на каждом уровне для каждого варианта размещения предполагается сохранение оптимального решения с учетом стоимости размещения и стоимости взаимодействия с блоками предыдущего уровня. Неэффективные решения для каждого размещения отбрасываются и далее не рассматриваются. На последнем шаге алгоритма оптимальное размещение определяется посредством возврата снизу вверх по уровням. При этом рекомендуется сохранять все полученные планы размещений, отсортированные по значению целевой функции, которые могут быть использованы при динамическом перепланировании задачи в случае необходимости.

При наличии связи «через уровень» для какого-либо блока задачи алгоритм предполагает добавление «фиктивного» блока нулевой вычислительной сложности.

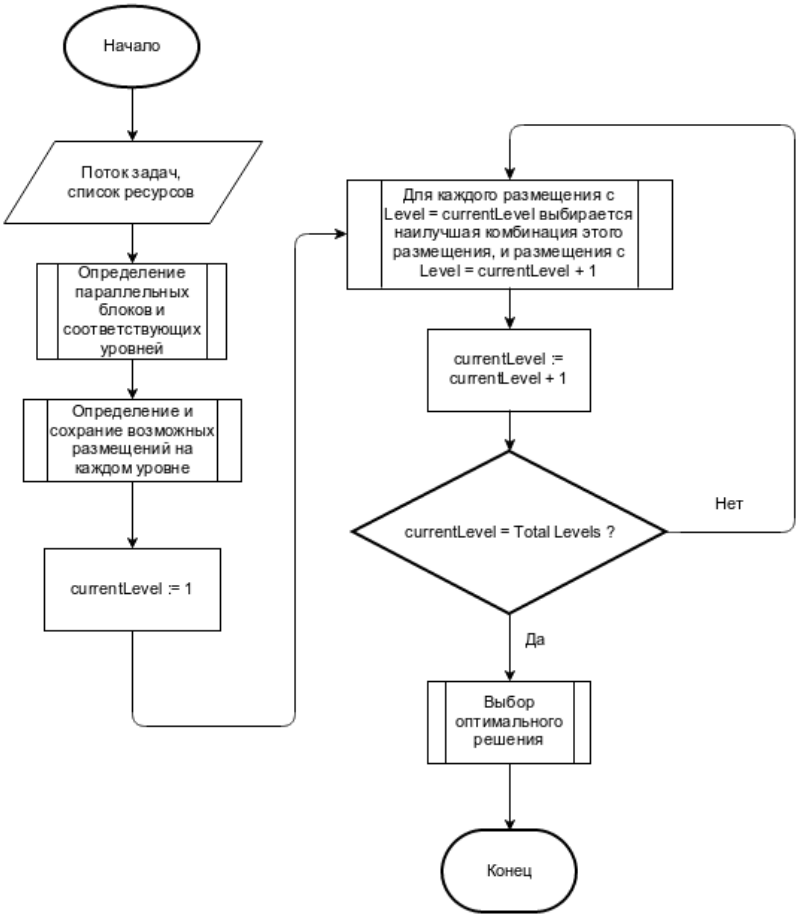

Рис. 3. Блок-схема алгоритма планирования на базе метода динамического программирования

6. Оценка эффективности алгоритмов планирования на базе имитационной модели процесса выполнения задач в grid

Для исследования свойств алгоритмов планирования был расширен инструментарий GridSim [17] посредством добавления новых сущностей, необходимых для моделирования процессов планирования и выполнения потоков работ в grid-среде. Диаграммы классов реализованных авторами модулей приведены на рис. 4–5.

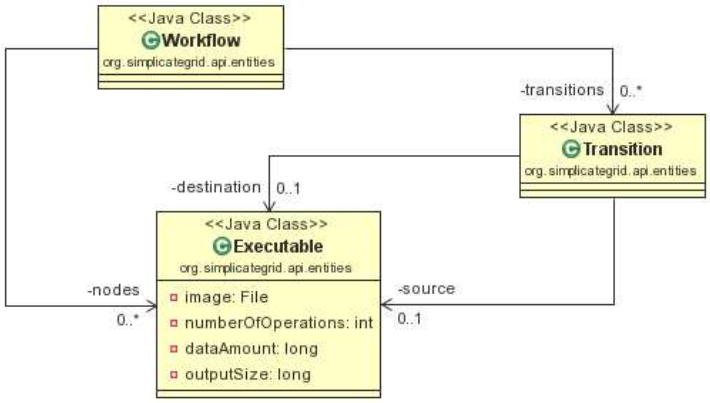

Рис. 4. Диаграмма классов модуля представления потока задач

Поток задач представляется сущностью Workflow, которая является хранилищем списка узлов, а также направленных ребер, связывающих эти узлы. Поток задач может состоять как минимум из одного узла (Executable) и без единого перехода. Workflow обязательно должен быть ориентированным ациклическим графом. Граф может иметь несколько точек входа и выхода.

Сущность Executable позволяет описать подзадачу потока задач и хранит информацию о её вычислительной

сложности в абстрактных единицах и размерности пересылаемых данных (подразумевается суммарный размер входных и выходных данных) в байтах.

Абстракция Allocation описывает множество узлов, которые могут выполняться одновременно, и их размещение на ресурсах. Предполагается, что распределение ресурсов полностью контролируется на уровне метапланировщика.

Для задания контракта алгоритма планирования был создан интерфейс SchedulingAlgorithm, который принимает в качестве параметров список доступных для планирования ресурсов и поток задач, а возвращает список объектов Allocation, отображающий последовательность выполнения узлов данного потока задач.

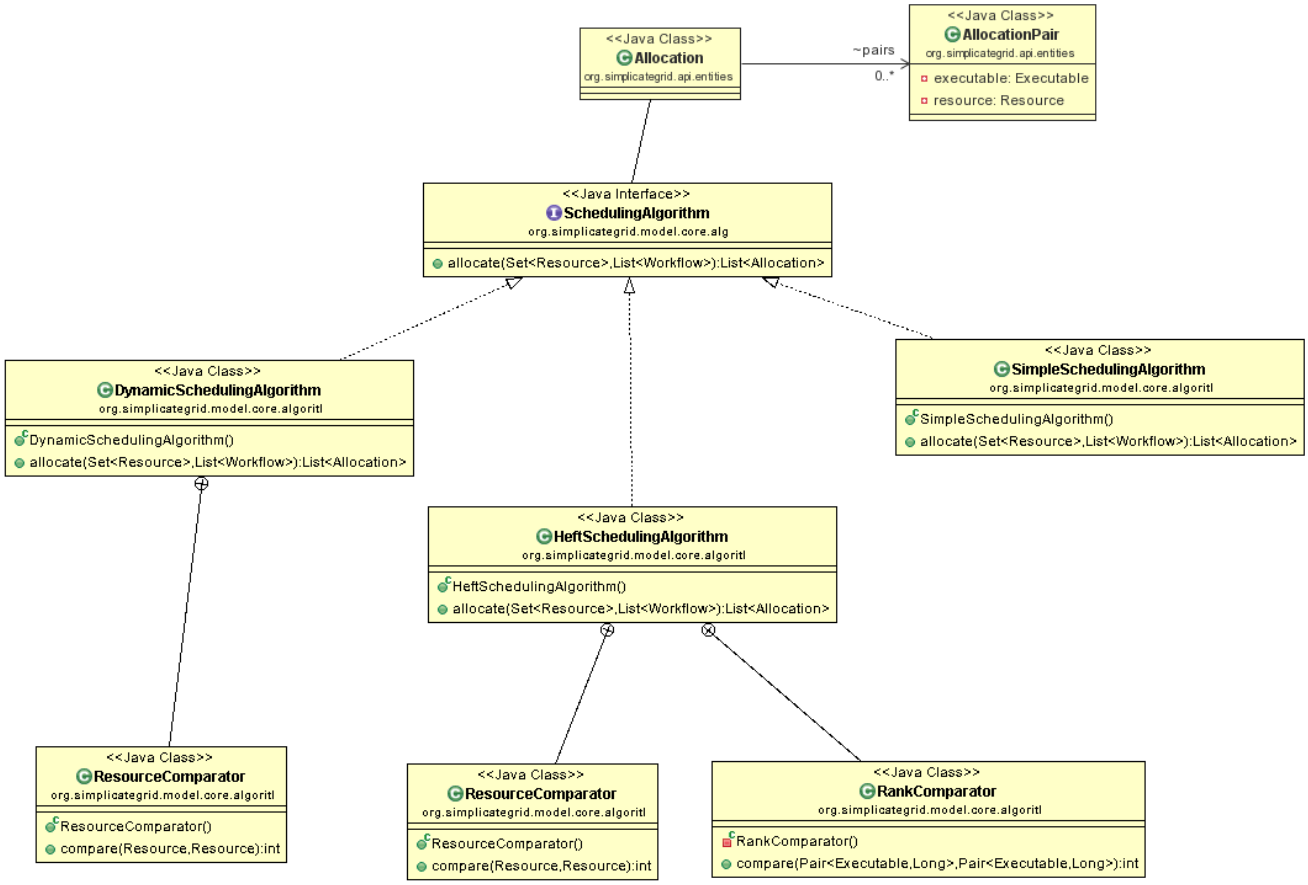

Рис. 5. Диаграмма классов модуля метапланировщика

Эксперименты проводились для случайных сгенерированных задач типа поток работ разной сложности. Пример сгенерированного случайным образом графа изображен на рис. 6.

Очевидно, что потоки реальных задач конкретных прикладных областей имеют меньшее количество вершин и связей. Однако использование более сложных потоков задач позволяет выявить узкие места в исследуемых алгоритмах.

Модель распределенной grid-среды для проведения экспериментов представлена четырьмя вычислительными узлами со следующими характеристиками: CE1 = {10 mips, 100, 0, {100, 50}}; CE2 = {25 mips, 100, 0, {100, 20}}; CE3 = {35 mips, 100, 0, {80, 40}}; CE4 = {47 mips, 100, 0, {100, 30}}.

Эффективность алгоритма планирования при проведении экспериментов определялась: 1) значением целевой функции; 2) вычислительной сложностью алгоритма.

В целях упрощения значение целевой функции определялось временем выполнения задачи без учета экономических затрат.

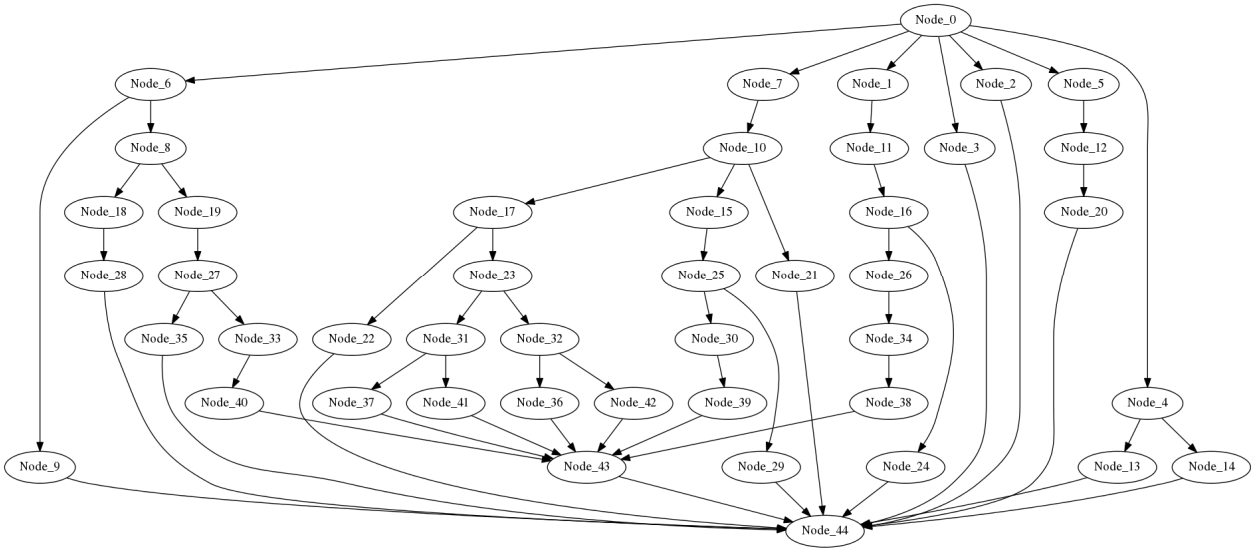

Рис. 6. Пример структуры задачи

В табл. 1 представлены результаты проведенных экспериментов для задач разной сложности и следующих алгоритмов планирования: 1 – эвристический алгоритм HEFT, 2 – алгоритм планирования на базе метода динамического программирования, 3 – случайный выбор ресурса для выполнения вычислительного блока задачи, готового к выполнению без учета стоимости размещения.

Таблица 1. Результаты экспериментов

Количество узлов	Количество связей	Количество уровней	Время планирования, мс	Время выполнения, с	Алгоритм	Отношение количество узлов / количество связей
10	13	5	3	518,4	1	0,769231
25	34	7	9	6859,91	1	0,735294
50	72	9	30	7241,28	1	0,694444
10	13	5	5	408	2	0,769231
25	34	7	12	5833,44	2	0,735294
50	72	9	43	6167,76	2	0,694444
10	13	5	1	576	3	0,769231
25	34	7	5	7873,92	3	0,735294
50	72	9	20	8203,2	3	0,694444

На рис. 7–8 представлены результаты оценки зависимости критериев эффективности алгоритмов планирования от сложности задачи.

Предложенный авторами алгоритм показал более высокую эффективность относительно критерия времени выполнения потока работ. Время планирования с применением метода динамического программирования выше, чем для других алгоритмов, однако при этом несравнимо меньше времени выполнения задачи в grid-среде, что оправдывает целесообразность предложенного решения.

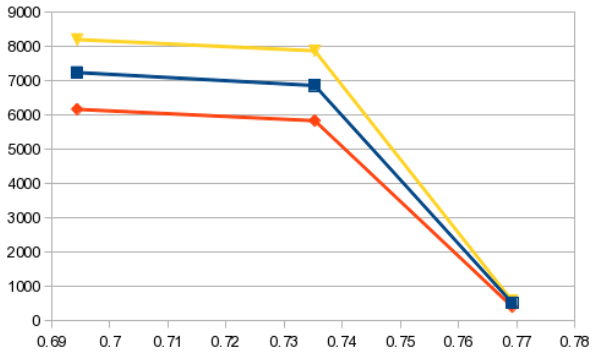

Рис. 7. Зависимость времени выполнения задачи от сложности задачи

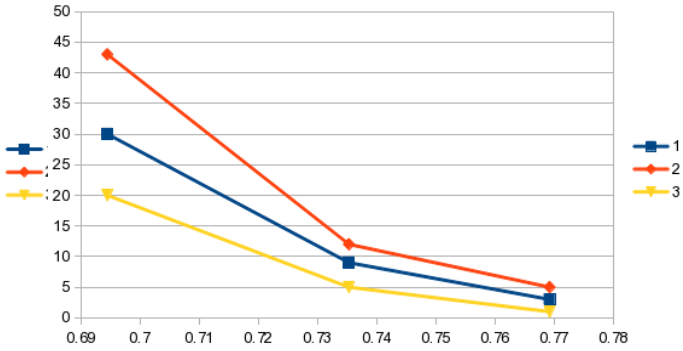

Рис. 8. Зависимость времени планирования от сложности задачи

7. Выводы

Рассмотрены особенности выполнения разных типов вычислительных задач в grid-среде. Представлены результаты разработки имитационной модели планирования задач типа потока работ на базе инструментария GridSim. Приведены результаты оценки эффективности предложенного авторами алгоритма планирования на базе метода динамического программирования и эвристического алгоритма HEFT. Эксперименты проводились на базе комплекса имитационных моделей.

Разработанная модель планирования потоков работ в grid-среде может быть интегрирована с модулем прогнозирования и планирования системы управления потоками работ. Перспектива дальнейших исследований заключается в формировании общей стратегии работы метапланировщика по обработке входного потока задач разных типов и проведении соответствующих экспериментов.

СПИСОК ЛИТЕРАТУРЫ

1. Forti A. DAG Scheduling for grid computing systems / A. Forti // Ph.D. Thesis, University of Udine, Department of Mathematics and Computer Science. – Italy, 2005 – 2006. – P. 43 – 46, 52 – 55.
2. Kazymyr V. Grid workflow design and management system / V. Kazymyr, O. Prila, V. Rudyi // International Journal “Information Technologies & Knowledge”. – 2013. – Vol. 7, N 3. – P. 241 – 255.
3. Gerasoulis A. A comparison of clustering heuristics for scheduling directed acyclic graphs on multiprocessors / A. Gerasoulis, T. Yang // Journal of Parallel and Distributed Computing. – 1992. – N 16. – P. 276 – 291.
4. Tompkins M.F. Optimization techniques for task allocation and scheduling in distributed multi-agent operations / M.F Tompkins // Diss. Massachusetts Institute of Technology. – 2003. – P. 20 – 23.
5. Job Submission Description Language (JSDL) Specification, Version 1.0, GFD-R.136. – 2008. – P. 5 – 10.
6. Extended Resource Specification Language, Reference Manual for ARC versions 0.8 and above, Nordugrid-Manual-4. – 2013. – P. 13 – 28.
7. Job description language attributes specification for the gLite Workload Management System, WMS-JDL.doc. – 2011. – P. 7 – 10, 38 – 40.
8. Sulistio A. A Grid Simulation Infrastructure Supporting Advance Reservation / A. Sulistio, R. Buyya // Proceedings of the 16th International Conference on Parallel and Distributed Computing and Systems (PDCS 2004), ACTA Press, Anaheim, California. – Cambridge, Boston, USA, 2004. – November 9–11. – P. 4 – 7.
9. A. Hirales-Carbajal. Multiple Workflow Scheduling Strategies with User Run Time Estimates on a Grid / A. Hirales-Carbajal, A. Tchernykh, R. Yahyapour [et al.] // Journal of Grid Computing. – 2012. –Vol. 10, Is. 2. – P. 325 – 346.
10. Melnyk A. Multiple DAGs Scheduling with Deadline Driven Coordinator in Grid / A. Melnyk // Second International Conference “Cluster Computing”. – Lviv, Ukraine, 2013. – June 3–5. – P. 127 – 130.

11. Yu J. Workflow Scheduling Algorithms for Grid Computing, Metaheuristics for Scheduling in Distributed Computing Environments / Yu J., Buyya R., Ramamohanarao K.; Xhafa F., Abraham A. (Ed.). – Berlin, Germany: Springer, 2008. – P. 111 – 149.
12. Fangpeng D. Scheduling Algorithms for Grid Computing: State of the Art and Open Problems / D. Fangpeng, Akl.G. Selim // School of Computing, Queen's University Kingston, Ontario, Technical Report. – 2006. – N 504. – P. 7 – 32.
13. Zomaya A.Y. Genetic Scheduling for Parallel Processor Systems: Comparative Studies and Performance Issues / A.Y. Zomaya, C. Ward, B. Macey // IEEE Transactions on Parallel and Distributed Systems. – 1999. – Vol. 10, N 8. – P. 795 – 812.
14. Liou J. CASS: an efficient task management system for distributed memory architectures / J. Liou, M.A. Palis // International Symposium on Parallel Architectures, Algorithms and Networks (ISPAN '97). - Taipei, Taiwan, 1997. – P 289 – 295.
15. 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, N 2. – P. 107 – 118.
16. Introduction to algorithms, third ed. / Thomas H.C., Leiserson C.E., Ronald L.R. [et al.]. – [3 ed.]. – Cambridge, Massachusetts London, England: The MIT Press, 2009. – P. 357 – 414.
17. Buyya R. GridSim: A Toolkit for the Modeling and Simulation of Distributed Resource Management and Scheduling for Grid Computing / R. Buyya, M. Manzur // The Journal of Concurrency and Computation: Practice and Experience (CCPE). – 2002. – Vol. 14, Is. 13–15. – P. 1179 – 1219.

Стаття надійшла до редакції 12.11.2013