АЛГОРИТМ ЖИЗНЕННОГО ЦИКЛА ПРИЛОЖЕНИЯ В МНОГОСЛОЙНОЙ МОДЕЛИ ОРГАНИЗАЦИИ РАСПРЕДЕЛЕННЫХ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ НА ОСНОВЕ ГРИД

Белорусский государственный университет информатики и радиоэлектроники
г. Минск, Республика Беларусь

Царевич Д.Ю.

Бахтизин В.В. – канд. техн. наук, доцент

Рассматривается алгоритм жизненного цикла приложения в многослойной модели [1] организации распределенных вычислительных систем (РВС) на основе грид. Проведен анализ алгоритма в контексте решения некоторых проблем РВС [2][3].

На рисунке 1 представлена схема алгоритма жизненного цикла приложения, определяющая последовательность операций, происходящих в многослойной модели организации РВС на основе грид при исполнении абстрактного пользовательского распределенного приложения:

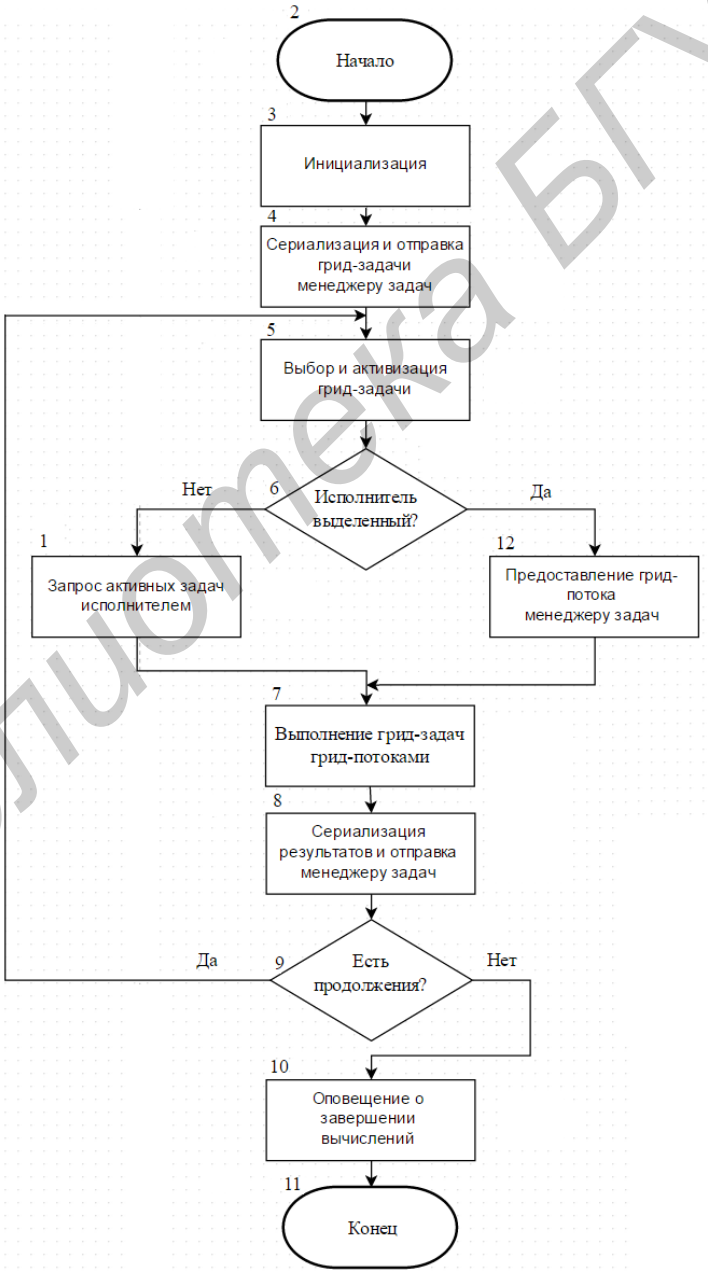

Рис. 1 – Схема алгоритма жизненного цикла приложения

Исходными данными для алгоритма жизненного цикла приложения является определяемая разработчиком грид-задача, описывающая последовательность действий для решения конкретной вычислительной задачи в многослойной модели организации РВС на основе грид.

В блоке 1 невыделенный исполнитель самостоятельно запрашивает у менеджера задач активные гридзадачи для выполнения до тех пор, пока таковые имеются.

В блоке 3 на узле пользователя происходит инициализация приложения, в частности инициализируется множество грид-задач приложения, где общее количество распределенных грид-задач может быть как строго заданным на этапе разработки приложения, так и динамически меняться на этапе исполнения с помощью обратной связи и/или набора эвристик, а также заполняется граф модулей-зависимостей приложения. В этом же блоке приложение подписывается на события изменения статуса у исходных грид-задач.

В блоке 4 исполняемое приложение сериализует и отправляет решаемую грид-задачу в менеджер задач, после чего менеджер задач помещает ее в очередь планировщика задач, а состояние приложения и грид-задачи сохраняется в базе данных.

В блоке 5 менеджер задач извлекает грид-задачу из очереди планировщика задач, и устанавливает ее статус как “активна”.

В блоке 6 выполняется определение менеджером типа исполнителя – выделенный или не выделенный. Если исполнитель не выделенный, то происходит переход в блок 1. Иначе происходит переход в блок 12.

В блоке 7 происходит выполнение грид-задач грид-потоками в рамках доменов приложений, где выделяется один домен приложения на приложение. Если домена приложения нет, то он создается через запрос, десериализацию и динамическую загрузку модулей-зависимостей приложения из заполненного в блоке 3 графа модулей-зависимостей приложения.

В блоке 8 результаты выполнения грид-задачи сериализуются и возвращаются в менеджер задач, статус грид-задачи устанавливается в “завершена”, а приложение оповещается через подписку на событие изменения статуса в блоке 3 о изменении статуса.

В блоке 9 выполняется определение, есть ли продолжение грид-задачи (т.е. зависимая грид-задача, заданная разработчиком и использующая результаты основной). Если задача-продолжение есть, то происходит переход к блоку 5 и обработка продолжения происходит таким же образом, как и основной задачи. Иначе происходит переход к блоку 10.

В блоке 10 вызывается определённая на этапе создания приложения функция-делегат, выполняющая необходимые действия после окончания вычислений (вывод результатов, отправка результатов по почте и т.д.).

В блоке 12 выделенный исполнитель непосредственно предоставляет экземпляру менеджера задач грид-поток для выполнения грид-задачи.

Кратко проанализируем рассматриваемый алгоритм в контексте решения некоторых проблем организации распределенных вычислений [2][3]:

1. Обеспечение гетерогенности компонент РВС. Является характеристикой многоуровневой модели организации РВС на основе грид, поэтому анализ этой характеристики не входит в рамки статьи.

2. Отказоустойчивость. Определяется механизмами избыточности окружения и вычислений – грид-задачи с одинаковыми исходными данными одновременно исполняются несколькими грид-потоками, консенсус о корректности полученного решения определяется голосованием. Цена обеспечения отказоустойчивости – рост стоимости полученного решения и частичное падение производительности. Уязвимость предлагаемого алгоритма – наличие централизованного менеджера задач, в случае отказа которого РВС выходит из строя. В качестве компенсации уязвимости следует рассмотреть резервирование менеджеров задач.

3. Масштабируемость. По горизонтали обеспечивается увеличением количества выделенных / не выделенных исполнителей (ограничена степенью параллелизма алгоритма и характеристиками окружения), по вертикали обеспечивается за счет замены исполнителей, менеджера задач и сетей коммуникаций между ними на обладающие более высокими характеристиками (ограничена уровнем развития технологий).

4. Открытость интерфейсов и расширяемость РВС. Не рассматривается в статье, т.к. является характеристикой выбранных организационного и технологического подходов к реализации РВС.

5. Безопасность. Соответствие результатов исходным данным обеспечивается за счет избыточности вычислений и механизма консенсуса. Конфиденциальность исходных данных и получаемых результатов обеспечивается за счет шифрования при коммуникации и механизмов контроля доступа к исполнителям и менеджерам задач. Защита исполнителей от грид-задач обеспечивается механизмом песочницы исполнения.

Таким образом, алгоритм жизненного цикла приложения в многослойной модели организации РВС на основе грид позволяет реализовать РВС с относительно высокими характеристиками отказоустойчивости (при резервировании менеджеров задач), масштабируемости и безопасности, но за счет частичного снижения производительности РВС и увеличения стоимости как РВС, так и выполнения расчетов на ней.

Список использованных источников:

1. Царевич, Д.Ю. Многослойная модель организации распределенных вычислительных систем на основе грид / Д.Ю. Царевич, В.В. Бахтизин // Альманах мировой науки: Научный журнал. – М.: «АР-Консалт», 2016. – Ч. 9-1 (12). – с. 44-45.
2. Таненбаум, Э. Распределенные системы. Принципы и парадигмы / Э. Таненбаум, М. Стеен. – Санкт-Петербург: Питер, 2003. – 877 с.
3. Coulouris, G. Distributed Systems: Concepts and Design (5th Edition). Coulouris G. [and others]. – Boston: Addison-Wesley, 2011. – 1008 p.