Автоматический перезапуск программных агентов

Лопаткин Р.Ю., Иващенко В.А.

Институт прикладной физики НАН Украины, Сумы, Украина

rlopatkin@gmail.com, va.ivashchenko@gmail.com

Аннотация. *В работе рассказывается как рашатеся проблема "горячей"перезагрузки программных агентов в мультиагентной системе. Решение этой проблемы позволит упростить процесс разработки и тестирования распределенных мультиагентных систем. Основная идея исследования состоит в том, чтобы делать автоматическое пересоздание агентов с новой логикой поведения без перезапуска всей платформы или ее составляющих, а также при возможности сохранять состояние агента. Также в данной работе рассмотрена техническая реализация перезагрузки агентов на языке программирования Java.*

Ключевые слова

Software agent, multiagent system, hot redeploy, hight performance computing.

1 Введение

В последнее время продолжается стремительное развитие мультиагентных систем [1, 2]. Успех этой технологии обусловлен тем, что мультиагентная архитектура позволяет упростить разработку распределенных систем предоставляя естественный способ синхронизации данных в системе путем отправки сообщений между агентами и обработки этих сообщений в порядке очереди. Добавлять новый функционал в агентную систему легче по сравнению с монолитными системами. Так, например, агент в вычислительной системе как правило представляет собой изолированную сущность, не связанную с другими агентами, которая взаимодействует с ними путем обмена сообщениями. Также мультиагентная архитектура позволяет создавать системы устойчивые к сбоям в работе их составляющих.

Процесс разработки и тестирования агентных распределенных систем [3] усложняется тем, что система состоит из множества агенов, логику каждого из которых в процессе разработки необходимо изменять. Причем агенты могут быть запущены на различных машинах системы (далее будем называть их нодами). Это желательно делать автоматически без необходимости перезаписывать вручную код агентов и затем заново запускать их на всех нодах системы, иначе разработка превратится в рутинный процесс. Более того, очень часто при разработке распределенных систем изменения касаются не всех классов компонента, а только некоторой связной между собой части, именно которая должна быть обновлена. Так, например, при разработке мультиагентных систем изменения чаще всего касаются логики поведения самих агентов и редко самой агентной платформы, которая обеспечивает жизнедеятельность и взаимодействие агентов. При этом система может находиться в состоянии, которое достигается путем длительного взаимодействия агентов и это состояние желательно сохранить после обновления логики агентов. Иначе дополнительно будет потрачено время на переход системы из начального в текущее состояние. Сохранить состояние системы можно только путем сохранения состояния агентов, так как состояние системы задается как набор состояний агентов.

В идеале при разработке мультиагентной платформы платформа не должна меняться вообще, а изменения должны касаться только агентов этой платформы. Поэтому нами решалась задача перезапуска агентов системы без перезапуска самой системы на всех нодах распределенной мультиагентной системы. Под перезапуском агента имеется ввиду обновление кода агента. Задача усложняется тем, что необходимо обновить агентов не на одной машинее, а сразу на всех машинах системы и при этом сохранить состояния всех агентов и не допустить краха системы. Еще одна сложность, которая возникает при решении этой задачи состоит в том, что после того как разработчик системы сохранил код агента, этот код необходимо перекомпилировать, но класс агента может иметь сложные зависимости от сторонних библиотек, своих форм, которые он предоставляет пользователю и других вспомагательных классов.

2 Литературный обзор

Перед тем, как начать изложение основных разделов статьи, стоит сформулировать постановку задачи, Задача “горячего перезапуска” компонентов системы не нова и существуют различные методы перезапуска в зависимости от архитектуры системы. В частности эта задача актуальна при разработке веб-приложений с использованием компилируемых не динамических языков программирования. В отличие от использования интерпретируемых языков программирования, в таком случае необходимо перезаписать часть или весь исполняемый код приложения после внесения изменений, чтобы получить возможность протестировать код с изменениями. Но при этом желательно сохранять состояние приложения – как минимум необходимо сохранить авторизацию пользователя, данные и текущую страницу на которой в текущий момент находится разработчик. Аналогичная проблема возникает при разработке мультиагентных систем – при внесении изменений в код агента, его необходимо обновить, при этом сохранив состояние. Но задача усложняется тем, что мультиагентная система – распределенная система и как правило запущена на множестве машин (нодов), на каждом из которых могут быть запущены экземпляры класса агента, который необходимо обновить. Фреймверк Akka [4] поддерживает перезагрузку агентов, но в нем перезагрузка реализована как восстановление агента после сбоя. обзор работ других авторов связанных с даной тематикой. Также стоит показать актуальность результатов, которые будут изложены в статье: почему они важны для решения научной или технической задачи.

3 Основные разделы

В каждый момент времени агент характеризуется своей логикой и состоянием. Состояние агента можно выразить в виде вектора

$$\mathbf{S} =< s_1, s_2, ..., s_n > \quad (1)$$

где $s_1, \ldots, s_n$ – свойства агента. При изменении кода агента, у него могут добавляться или удаляться свойства. Если свойство s_i было удалено, то это никак не мешает создать агента с новой логикой и сохранением его состояния (естественно без этого свойства): $\mathbf{S}' =< s_1, s_2, ..., s_{i-1}, s_{i+1}, ..., s_n >$. Действительно, если свойство удалено, то оно не является частью состояния агента, а также от него не зависят другие свойства, иначе - будет ошибка компиляции агента. Сложнее случай когда агенту было добавлено новое свойство. Тогда возможны следующие варианты:

1. новое состояние связано с остальными и оно зависит от других состояний или другие состояния зависят от него.
2. это состояние не связано с остальными состояниями.

В связи с этим были разработаны различные политики редеплоя.

Существуют следующие основные политики редеплоя:

1. Агент всегда создается в своем начальном состоянии.
2. Агент создается в своем начальном состоянии только если были добавлены новые свойства агента, иначе - сохраняется состояние.
3. Агент всегда создается с сохранением своего состояния, даже если были добавлены новые свойства.
4. Агент никогда не пересоздается, даже если пришло обновление с обновленным кодом агента.

3.1 Алгоритм перезапуска агентов системы

В общем алгоритм перезапуска всех агентов системы состоит из глобальной и локальной части. Отличие глобальной части алгоритма перезапуска от локальной состоит в том, что в первой участвуют агенты специального типа, которые ведут переговоры с целью распространения обновления по всем нодам системы, а во второй те же агенты выполняют действия по обновлению без какого-либо обмена сообщениями с другими агентами. Рассмотрим глобальную часть алгоритма обновления агентов системы, смотртире рисунок 1.

Согласно алгоритму, изображенному в виде UML диаграммы на рисунке 1, в какой-то момент времени один из специального типа агентов (Updater Agent) обнаруживает обновление. Сразу же этот агент распространяет обновление по системе при помощи обычнго или лавинообразного алгоритма [6]. После того, как обновление будет распространено по всем нодам системы, агент, который изначально получил обновление дает команду всем остальным

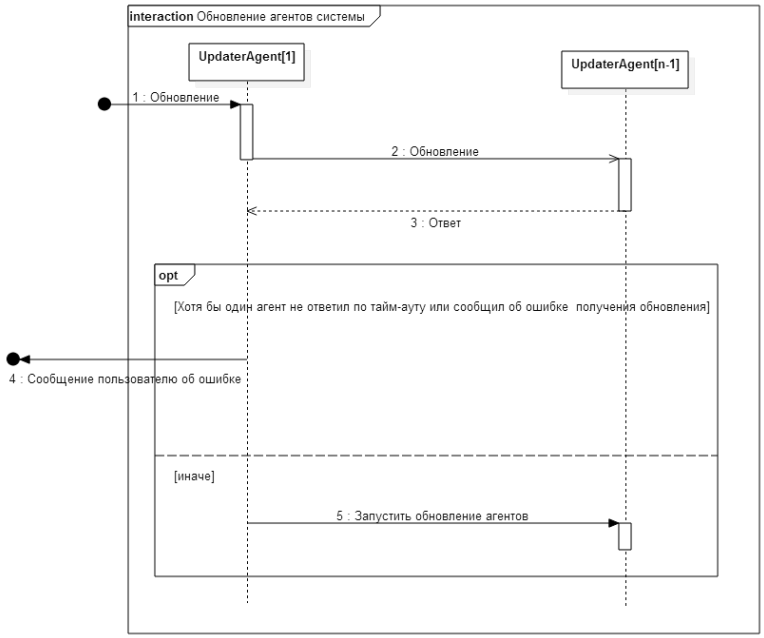

Рис. 1. Глобальный алгоритм обновления агентов системы.

UpdaterAgent-ам пересоздать агентов, для которых пришло обновление и ждет от них ответа. Если хотя бы один ответ содержит сообщение об ошибке или не был получен до истечения тайм-аута, то обновление считается не удачным и об этом показывается соответствующее сообщение пользователю. Иначе – UpdaterAgent снова посылает остальным UpdaterAgent-ам агентам команду, на этот раз – для запуска новых экземпляров агентов. В случае, если агент перезапускается с сохранением своего предидущего состояния, то локальный алгоритм перезапуска агента следующий:

1. “Заморозить” агента.
2. Создать новый экземпляр такого же вида агента.
3. Копировать состояние из старого агента в нового.
4. Удалить старый экземпляр агента из системы.

Этот алгоритм является локальным потому, что работает в рамках каждого отдельного нода. Следует отметить, что на четвертом шаге новые экземпляры агентов созданы но еще не запущены. Рассмотрим более подробно по каждый из перечисленных пунктов. Во-первых, “Заморозить” агента означает приостановить выполнение агента без его удаления. Технически это достигается, напрмер, путем приостановки потока агента. Создание нового экземпляра агента такого же вида подразумевает создание экземпляра объекта этого агента, но уже с нового скомпилированного кода. Этот алгоритм необходимо запускать одновременно на всех нодах для всех агентов системы, чтобы не было конфликтов взаимодействия старых версий агентов с новыми.

3.2 Техническая реализация

Каждое поле (свойство) агента, которое есть частью его состояния и соответсвенно должно быть устойчивым к перезапуску агента необходимо помечать специально аннотацией @StateProperty. Задать политику редеплоя агента можно с помощьюь аннотации @Redeploy, в которой как параметр задается политика редеплоя. Например, @Redeploy(RedeployPolicy.IMMUTABLE) задает четвертый тип политики редеплоя согласно которому агент никогда не пересоздается, даже если пришло обновление с обновленным кодом. Для упрощения “редеплоя” нами был разработан плагин для среды NetBeans. Этот плагин добавляет кнопку на панель Build среды NetBeans, которая осуществляет перезапуск агентов системы, смотрите рисунок 2. Для того, чтобы произвести перезапуск, данный плагин использует Apache Ant Build Tool [7]. “Инструкция” для пересборки загружается из xml файла.

По нажатии на кнопку “Redeploy agents” выполняется перезапуск, который включает в себя последовательность действий описанную ниже. Все начинается с очистки папки со сборкой от предидущей версии файлов. Затем код

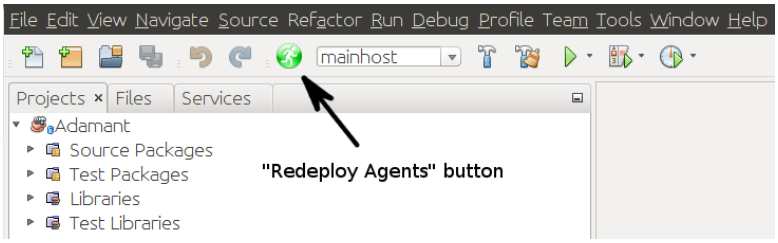

Рис. 2. Кнопка в среде разработки NetBeans для перезапуска агентов системы.

агентов, а также необходимых для них компонентов, который находится в отдельном пэкэдже (org.sumy.iap.agents) перекомпилируется. Затем скомпилированный код упаковывается в jar архив, после чего архив копируется в специальную папку проекта, которую постоянно мониторит специальный агент, UpdaterAgent. Когда этот агент обнаруживает новый архив, он рассылает его другим UpdaterAgent-ам на все машины системы. После успешнно перезапуска агентов будет показано сообщение об этом, иначе — сообщение об ошибке.

Стоит отметить, что создание подобных расширений возможно и для других интегрированых сред разработки. После того как архив распространен по всей системе, UpdaterAgent-ам необходимо договориться об одновременном обновлении кода на своих машинах, особенно важно стартовать всех “обновленных агентов” системы одновременно.

4 Заключение

Реализован редеплой агентов на лету. Разработан общий алгоритм “гарячего” обновления агентов в распределенной системе, что особенно важно при отработке логики их работы. Для перекомпиляции кода агентов использовалась библиотека Ant с открытым исходным кодом.

Список литературы

[1] Deloach, S.A., Wood, M.F., Sparkman, C.H.: Multiagent Systems Engineering. Int. Journal of Software Engineering and Knowledge Engineering 11(3), 231–258 (2001).

[2] C. Bernon, V. Chevrier, V. Hilaire, et al., “Applications of Self-Organising Multi-Agent Systems: An Initial Framework for Comparison,” Informatica, No. 30, 73–82 (2006).

[3] Cu D. Nguyen. Testing in Multi-Agent Systems / Cu D. Nguyen, Anna Perini, Carole Bernon, Juan Pavón, John Thangarajah // Agent-Oriented Software Engineering X. Lecture Notes in Computer Science Volume 6038, 2011, pp 180-190.

[4] Actors - Akka Documentation [Электронный ресурс], режим доступа: http://doc.akka.io/docs/akka/snapshot/java/untyped-actors.html - Заголовок с экрана.

[5] NetBeans Platform Plugin Quick Start [Электронный ресурс], режим доступа: https://platform.netbeans.org/tutorials/nbm-google.html - Заголовок с экрана.

[6] Лопаткин Р. Ю., Иващенко В. А. Распространение информации в распределенной вычислительной сети / междунар. конф. PDCS-2013, г. Харьков (International Conference "Parallel and Distributed Computing Systems") PDCS 2013 (Ukraine, Kharkiv, March 13-14, 2013) с. 206 - 208

[7] Apache Ant [Электронный ресурс], режим доступа: http://ant.apache.org/ - Заголовок с экрана.