УДК 004.94

Агентная архитектура распределенной дискретно-событийной системы имитационного моделирования OpenGPSS.

Томашевский В.Н.

Диденко Д.Г.

1. Введение. Метод имитационного моделирования приобретает все большую популярность при проектировании различных систем. Однако, несмотря на рост производительности современных компьютеров, их мощности не хватает для моделирования задач, связанных с самолетостроением, автомобилестроением, логистикой, сборочным производством и т.п., когда имитационные прогоны моделей могут длиться часами. Одним из вариантов решения этой проблемы является использование параллельного и распределенного дискретно-событийного имитационного моделирования.

Существует много реализаций распределенных систем имитационного моделирования, например, SPEEDES [1] и PARASOL [2], и даже есть стандарт в этой области – HLA (High Level Architecture). Однако многие разработчики имитационных моделей отмечают сложность реализации этого стандарта, а также невозможность его использования для языков декларативного типа, таких как GPSS [3] из-за объектно-ориентированной направленности HLA. В тоже время язык GPSS широко используется в мире, недаром одна из последних его версий названа GPSS World [4]. Поэтому актуальной является проблема создания распределенной системы [5], с автоматическим распределением GPSS-программ.

Предлагается создание распределенной системы имитационного моделирования OpenGPSS [6] с использованием технологии взаимодействующих агентов и многосессионной клиент-серверной СУБД Oracle, позволяющей одновременно работать нескольким пользователям (сессиям). В такой системе пользователь сможет задавать задание системе моделирования через WEB-интерфейс, а потом получать результат подобно тому, как это реализовано в системе WebGPSS [7].

Одна из главных проблем в проектировании распределенных системах связана со сложностью синхронизации модельного времени, сущностей и т.д. Уже существуют десятки алгоритмов синхронизации модельного времени и их модификации. Более подробное описание алгоритмов можно найти в работах [8, 9]. Математическая формализация алгоритмов синхронизации времени для распределенного имитационного моделирования и сравнение их производительности приведено в работе [10].

2. Архитектурные особенности системы OpenGPSS

Предлагается создание системы OpenGPSS с использованием кластера моделирования (SC), состоящего из серверов моделирования (SS), предназначенных для хранения моделей, сущностей и проведения распределенных имитационных экспериментов (рис. 1.).

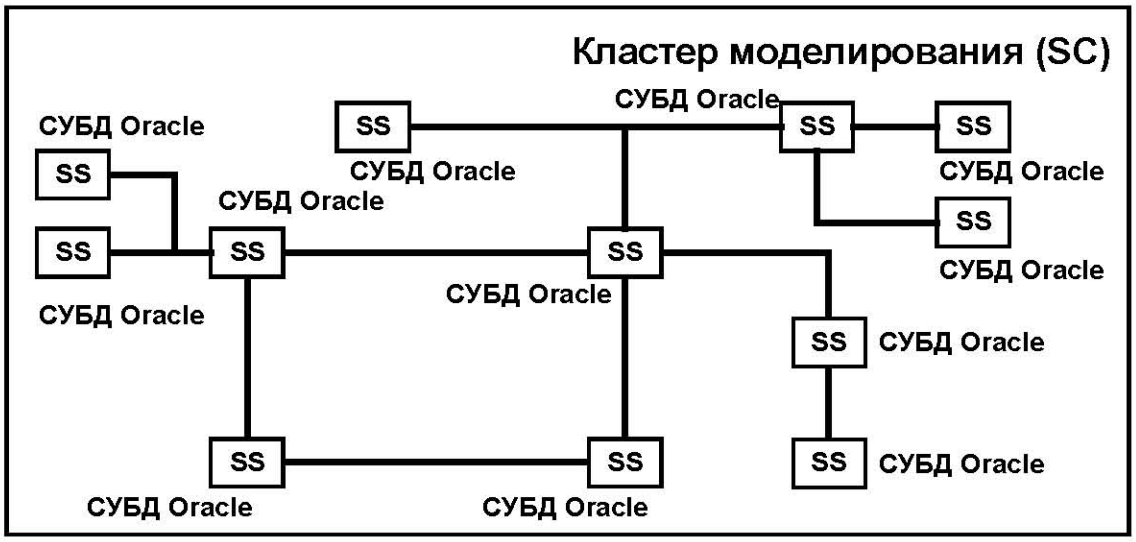

Рис. 1. Структура системы OpenGPSS

Дадим некоторые определения, необходимые для описания архитектуры системы OpenGPSS. Под *сущностями* будем понимать множество объектов (блоков) языка GPSS [3]: одноканальные устройства (Facility), многоканальные устройства (Storage), очереди (Queue), функции (Function), логические переключатели (Logic Switch), матрицы (Matrix), сохраняемые величины (Save Value), таблицы (Table), списки пользователя (User Chain), а также, дополнительно, генераторы (Generate).

Понятие *сообщение* в системе OpenGPSS аналогично понятию транзакта в языке GPSS и используется для изменения состояния сущностей.

Под *агентом* в системе OpenGPSS будем понимать программную систему [11], обладающую следующими свойствами:

- **Автономность** – способность работать без внешнего управляющего воздействия. В реализации OpenGPSS агентами выступают задания (job) [12], которые выполняются сервером. Автономность понимается как асинхронный запуск агента в отдельной сессии согласно расписанию выполнения заданий.
- **Реактивность** – возможность воспринимать среду, реагировать на ее изменения. Время запуска задания может изменяться в зависимости от состояния сервера. Например, в некоторых случаях возможен неотложный запуск таких агентов как AgRep, AgSnc, AgGbr, AgSpl, описанных ниже, путём переноса соответствующего задания в списке заданий.

- **Активность** (pro-activeness) – способность ставить цели и инициативно действовать для достижения поставленной цели. Активность зависит от типа агента. Цели и задачи агентов описаны ниже.
- **Коммуникативность** – способность взаимодействовать с другими агентами (или пользователями). Каждый агент обладает интерфейсами, доступными другим агентам и пользователям. Кроме того, существуют интерфейсы сервера, с помощью которых агенты могут общаться друг с другом. Примером может быть использование каналов (pipe) [10] агентом приёмо-передачи данных.

Система OpenGPSS использует технологию взаимодействующих агентов, базируется на кластере и поддерживает динамическую переконфигурацию кластера в случае отказа или при подключении нового сервера. Информация о сетевой топологии кластера используется только на системном уровне, что обеспечивает прозрачность имитации для пользователей. В GPSS-программе нет директив распараллеливания, свойственных языкам параллельного программирования, например, таким как HPF и OpenMP Fortran [5].

Система OpenGPSS работает в гетерогенной сети, на различных архитектурах компьютеров с различными операционными системами. Для синхронизации модельного времени используется оптимистический подход [9], который, в отличие от консервативного подхода, обеспечивает большую производительность, хотя и увеличивает сложность реализации системы. Для упрощения работы пользователей реализован WEB-интерфейс для работы через Интернет. Обеспечение отказоустойчивости достигается с помощью механизмов репликации и других способов защиты данных.

2. Агентная архитектура сервера моделирования

Рассмотрим более подробно архитектуру сервера моделирования SS (рис. 2), состоящую из взаимодействующих агентов: агент имитационного моделирования AgSim, агент репликации модели AgRep; агент разделения нагрузки AgSpl; агент синхронизации AgSnc; агент приёмо-передачи сообщений AgTrf; агент производительности сервера AgPwr; агент взаимодействия с пользователем AgUsr; агент сборки мусора AgGbr.

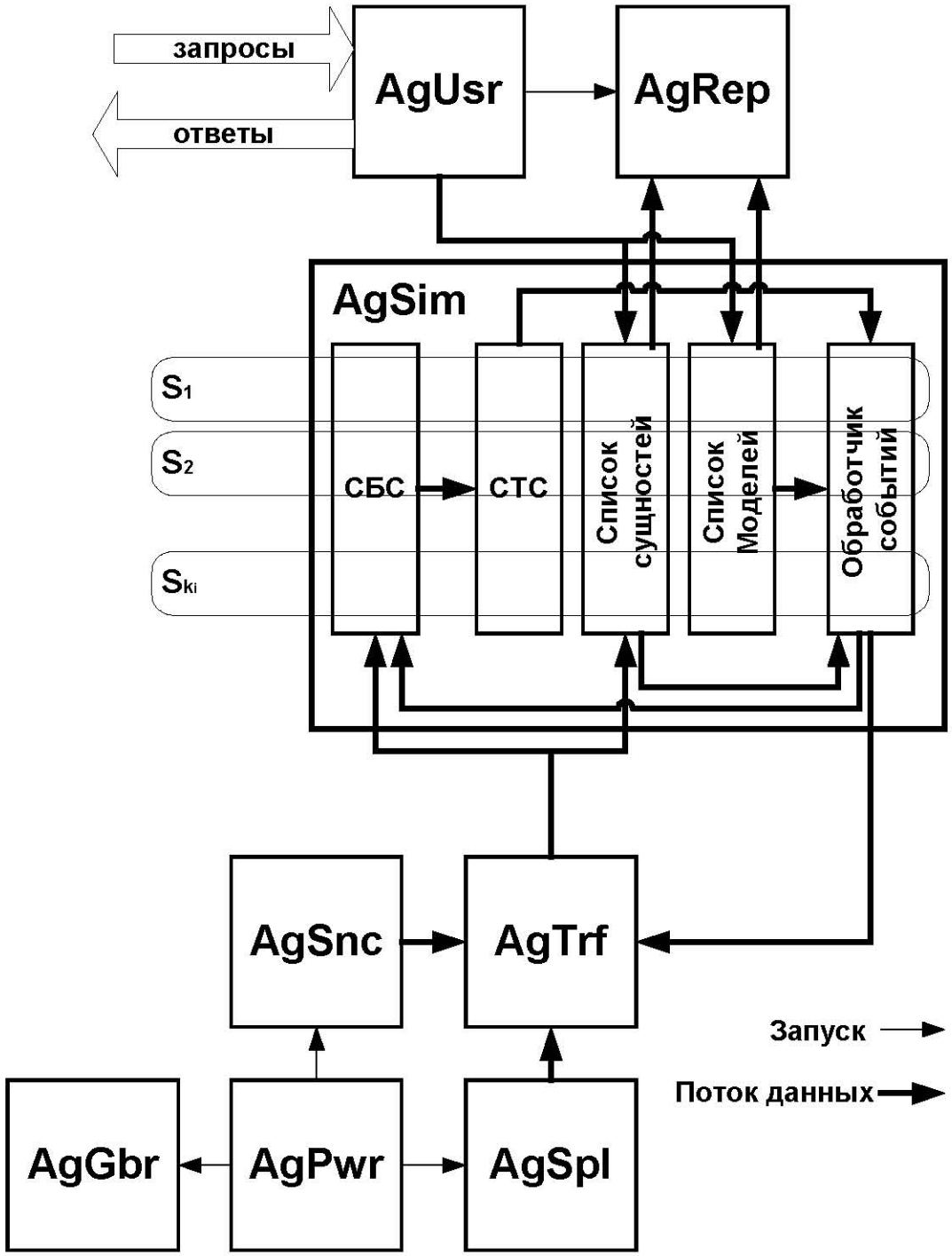

Рис. 2. Агентная архитектура сервера моделирования

Пусть $A_i = \{A_i^{AgSim}, A_i^{AgRep}, A_i^{AgSpl}, A_i^{AgSnc}, A_i^{AgTrf}, A_i^{AgPwr}, A_i^{AgUsr}, A_i^{AgGbr}\}$ – множество активных агентов i-го сервера, $i = \overline{1,n}$, где n – количество серверов имитационного моделирования в кластере.

Введем матрицу $\mathbf{B}_{iz}$ размерности $n \times n$, которая показывает есть ли связь между i-м и z-м сервером:

$$\mathbf{B}_{iz} = \begin{Bmatrix} 1, & \textit{если существует связь между } i-\textit{м и } z-\textit{м сервером}; \\ 0, & \textit{в противномслучае}; \end{Bmatrix}, \; i = \overline{1,n}, \; z = \overline{1,n}.$$

Пусть S_{ij} – *j*-е задание пользователя на *i*-ом сервере, $i=\overline{1,n}$, $j=\overline{1,k_i}$, где k_i - количество заданий на *i*-ом сервере. Обозначим через F_i – устаревшие задания пользователя *i*-го сервера. Множество модельных времен *i*-го сервера определим как $T_i=\{t_{i1},t_{i2},..,t_{ij},..,t_{ik_i}\}$, а результаты выполнения *j*-го задания пользователя на *i*-м сервере обозначим R_{ij}.

Пусть $m_{ijz}=(m_{ijz1},m_{ijz2})$ – маркеры начала и конца обработки блока, и $M_{ij}=\bigcup_z(m_{ijz1},m_{ijz2})$ – разделение всего GPSS-сценария на непересекающиеся части для разных серверов, при этом берутся такие *i* и *z*, для которых $\mathbf{B}_{iz}=1$, причем $M_i=\bigcup_{j=1}^{k_i}M_{ij}$.

Обозначим через D_{ij} – очереди исходящих сообщений для *j*-го задания на *i*-м сервере, а через Q_{ij} очереди входящих сообщений на *i*-м сервере, причем $Q_i=\bigcup_{j=1}^{k_i}Q_{ij}$ и $D_i=\bigcup_{j=1}^{k_i}D_{ij}$.

Состояние *i*-го сервера опишем как $P_i(A_i,S_i,F_i,T_i,R_i,M_i,Q_i,D_i)$, а состояние кластера моделирования – $P=\bigcup_n^{i=1}P_i(A_i,S_i,F_i,T_i,R_i,M_i,Q_i,D_i)$.

Определим агент имитационного моделирования AgSim как незамкнутую систему дискретно-событийного имитационного моделирования, состоящую из списка будущих событий (СБС), списка текущих событий (СТС), списка моделей, списка сущностей и обработчика событий. Списки логически разделяются между пользователями (сессиями) Oracle. Работа этого агента описывается оператором

$$A_{ij}^{AgSim}:P_i(\{t_{i1},t_{i2},..,t_{ij},..,t_{ik_i}\})\to P_i(\{t_{i1},t_{i2},..,t_{ij}^{`},..,t_{ik_i}\}),\text{ где } t_{ij}<=t_{ij}^{`}. \quad (1)$$

После завершение моделирования *j*-го задания, агент имитационного моделирования сохраняет результаты в списке результатов R_i:

$$A_{ij}^{AgSim}:P_i(A_i,R_i)\to P_i(A_i\setminus\{A_{ij}^{AgSim}\},R_i\cup R_{ij}).$$

Агент взаимодействия с пользователем AgUsr отвечает за взаимодействие системы с пользователями. Он транслирует текст GPSS-программы [13] во внутренний формат системы (строки модели, сущности), а далее заносит в список заданий (job) сервера новую задачу проведения имитации с помощью оператора

$$A_i^{AgUsr}:P_i(A_i,S_i,T_i)\to P_i(A_i\cup\{A_{ik_i+1}^{AgSim}\},S_i\cup\{s_{ik_i+1}\},T_i\cup\{t_{ik_i+1}\})$$

Любое изменение структуры GPSS-модели (перегрузки блоков) приводит к “устареванию” старой задачи и созданию на ее основе новой.

При удалении *j*-го задания, $1 \le j \le k_i$ этот агент переносит задание из списка активных заданий S_i в список устаревших заданий F_i и очищает все списки *i*-го сервера от *j*-го задания с помощью оператора

$$A_i^{AgUsr} : P_i(A_i, S_i, F_i, T_i) \to P_i(A_i \setminus \{A_{ij}^{AgSim}\}, S_i \setminus \{s_{ij}\}, F_i \cup \{s_{ij}\}, T_i \setminus \{t_{ij}\} \mid \forall i, z, \mathbf{B}_{iz} = 1).$$

Агент синхронизации AgSnc определяет окончание моделирования, когда входящие списки агентов приёмо-передачи сообщений для j-того задания, списки СБС и СТС пусты, и проводит распределенное подтверждение – синхронизирует сущности на серверах. После этого, по запросу пользователя, агент AgUsr генерирует стандартный отчет с результатами моделирования.

Агент репликации модели AgRep копирует GPSS-модель во внутреннем формате (модель, сущности) для данного задания с *i*-го сервера на *z*-й

$$A_i^{Ag\,\mathrm{Re}\,p} : P_z(S_z, F_z, T_z, R_z) \to$$
$$P_z(S_z \cup (S_i \setminus S_z), F_z \cup (F_i \setminus F_z), T_z \cup \{t_{ij} \mid \forall j, s_{ij} \in (S_i \setminus S_z)\}, R_z \cup (R_i \setminus R_z)), \forall i, z : \mathbf{B}_{iz} = 1.$$

На каждом сервере хранится своя копия задания, и далее при балансировке нагрузки передаются только маркеры начала и конца обработки. Задача этого агента состоит в том, чтобы реплицировать модель на как можно большее количество серверов в кластере.

Для исключения ситуации остановки репликации в случае отказа сервера-инициатора при добавлении или обновлении моделей, а также для случая подключения нового сервера SS в кластер моделирования SC, агент репликации периодически запускается и добавляет и (или) обновляет модели своего сервера.

В общем случае в кластере одновременно могут работать множество таких агентов – но только по одному на каждом сервере, при этом каждый копирует (обновляет) модели только со своего сервера. Возникновение конфликтов, когда два агента пытаются реплицировать одну и ту же модель, исключаются использованием алгоритма двухфазного подтверждения 2PC, описанного в работе [14]. Когда агент-инициатор блокирует заданную сессию для всех агентов репликации на всех серверах, и далее останавливает обработку данной сессии, обновляет модель на всех серверах, и только потом подтверждает и (или) отменяет изменения.

Агент приёмо-передачи сообщений AgTrf обеспечивает обмен сообщениями между серверами и поддерживает целостность системы относительно модельного времени

$$A_{ij}^{AgTrf} : P_z(Q_z) \to P_z(Q_z \cup \{d_{ijz} \mid \forall z, \mathbf{B}_{iz} = 1, z \in M_{ij}\}).$$

Оптимистическая синхронизация, используемая в системе, основывается на обработке событий, не смотря на то, что позже могут возникнуть более ранние события, определении неправильного порядка событий и использовании отката в случае “неправильного” порядка событий.

Неправильный порядок событий возникает тогда, когда приходят сообщения, намеченные на более раннее время, чем текущее модельное время, в результате чего происходят откаты. Откат можно разделить на две части: восстановление состояния задания сервера SS и удаление недействительных сообщений из кластера SC.

Для реализации оптимистической синхронизации используются очереди исходящих сообщений D и очереди входящих сообщений Q.

В качестве алгоритма синхронизации используется модифицированный алгоритм виртуального времени (TimeWrap) Джефферсона [8] с временным окном. Основой этого алгоритма является понятие антисообщения и поглощение сообщений, с помощью которых решается проблема определения неправильного порядка сообщений и частичного отката (откат недействительных сообщений). При этом проблема восстановления состояния j-го задания на i-ом сервере при откате решается средствами Oracle.

При приёме сообщения агент передает его далее в СБС для указанного в сообщении задания, если сообщение является транзактом, или передает его в список сущностей, если сообщение изменяет состояние сущности.

Антисообщение – специальное сообщение, которое используется для отмены недействительного сообщения. Оно определяется как обычное сообщение с “перевернутым” первым разрядом, и для него нет необходимости передачи всей информации (например, про транзакт), а достаточно передавать лишь идентификатор сообщения (является уникальным для всего кластера SC) с отрицательным флагом.

Для каждого полученного сообщения в кластере может существовать одно и только одно антисообщение. Когда в одной очереди находятся и сообщение и антисообщение – оба сообщения удаляются из неё. Для отмены недействительных сообщений достаточно послать соответствующее антисообщение, которое на сервере не обрабатывается, а хранится во входящей очереди агента AgTrf.

Полученные агентом антисообщения, для которых нет соответствующего им сообщения в СБС, хранятся во входящей очереди в качестве “обогнавших” зависимые сообщения. Если в СБС есть зависимое сообщение, тогда происходит поглощение двух сообщений. Если же зависимое сообщение уже обработано, то все обработанные сообщения, следующие за антисообщением, объявляются недействительными, и делается откат до модельного времени антисообщения.

Механизм отката активизируется, когда агент AgTrf на входе получает более ранее сообщение – отстающее сообщение. Откат предусматривает восстановление состояния j-го задания i-го сервера SS (состояние значений переменных) на состояние до раннего сообщения и отмену недействительных сообщений путём посылки соответствующих антисообщений.

От количества контрольных точек сильно зависит производительность сервера, что является параметром настройки “глубины отката”: от крайнего случая, когда точка сохранения происходит после каждого сообщения, до оптимистического выполнения серии сообщений.

При откате локальное время переводится назад и устанавливается равным модельному времени отстающего сообщения, путем восстановления состояния узла моделирования на этот момент из очередей агента AgTrf.

Когда агент AgSnc присылает список глобальных виртуальных модельных времён всех заданий, агент AgTrf удаляет из входящих и исходящих списков устаревшие события, модельное время которых меньше t_{ij}^{GVT} [8]:

$$A_{ij}^{AgTrf} : P_i(Q_i, D_i) \to P_i(Q_i \setminus \{q_{ij} \mid t_{q_{ij}} < t_{ij}^{GVT}\}, D_i \setminus \{d_{ij} \mid t_{d_{ij}} < t_{ij}^{GVT}\}).$$

Получив сообщение на отправку, агент создаёт из него антисообщение, и далее отсылает только сообщение, а антисообщение остаётся в выходящей очереди.

При обработке транзактов агентом имитационного моделирования AgSim возможны два режима передачи сообщений: локальный и глобальный, которые зависят от маркера начала обработки и маркера конца обработки транзактов. В первом случае происходит попадание события в локальный кэш событий: транзакт не нужно передавать на другой сервер и он передается в СБС (1). Во втором, если вышли за маркер конца обработки, сообщение пересылается агенту приёмо-передачи сообщений (2), который рассылает этот транзакт следующему серверу из списка M_{ij}.

$$A_{ij}^{AgSim} : P_i(\{t_{i1}, t_{i2}, .., t_{ij}, .., t_{ik}\}, D_i) \to P_i(\{t_{i1}, t_{i2}, .., t_{ij}^{`}, .., t_{ik}\}, D_i \cup \{d_{ij} \mid m_{ijz} \in M_{ij}, \mathbf{B}_{iz} = 1\}) \quad (2),$$

где $t_{ij} <= t_{ij}^{`}$.

Если при обработке происходит изменение состояний сущностей, то всегда генерируется сообщение об изменении состояния сущности и далее передаётся агенту AgTrf, который опять используя M_{ij} передаёт сообщения нужным серверам.

Агент разделения нагрузки AgSpl распределяет нагрузку среди серверов в кластере в зависимости от структуры GPSS-модели (статический анализ) и нагрузки серверов,

полученной от агента AgPwr (динамический анализ). Работа агента описывается следующим оператором

$$A_{ij}^{AgSpl} : P_i(M_i) \to P_i(M_i \cup M_{ij}\}).$$

Результатом работы агента является передача серверам (а именно обработчикам событий каждого сервера) маркеров начала обработки и маркеров конца обработки. Агентом формируется список M_{ij}, в котором хранится информация какие сообщения переслать каким серверам. Этот список агент AgSpl передает агенту приёмо-передачи данных для дальнейшего хранения и использования для передачи сообщений.

На агент разделения нагрузки также возложено выполнение переконфигурации в случае отказа одного из серверов. Отказ определяется агентом AgSnc на основе списка M_{ij}, который хранится у агента AgTrf. После этого на множестве зависимых серверов происходит откат до контрольной точки. Далее агент AgSpl заново разделяет нагрузку на новое множество серверов (исключается отказавший сервер) и моделирование возобновляется. При добавлении нового сервера SS в кластер SC переконфигурация не происходит до очередного запуска агента разделения нагрузки.

Основными задачами агента синхронизации AgSnc являются глобальное управление выполнением имитаций и поддержание непротиворечивости моделей в точках сохранения.

В системе OpenGPSS используется оптимистический подход синхронизации, и агент AgSnc используется как механизм глобального контроля. Агент вычисляет глобальное виртуальное модельное время t_{ij}^{GVT} для каждого задания сервера (минимум модельного времени всех необработанных или частично-обработанных сообщений или антисообщений.) Все сообщения и антисообщения, которые хранятся в СБС, входящих и исходящих очередях сообщений агента AgTrf с модельным временем меньшим, чем глобальное время t_{ij}^{GVT}, могут быть удалены из соответствующих списков для всего кластера.

Глобальность агента AgSnc приводит к достаточности одного такого агента на весь кластер SC. Используя алгоритм голосования “забияки”, описанный в работе [14], определяется единственный координатор для сервера с наименьшей загрузкой (или с уже запущенным таким агентом), которая получается от агента AgPwr. На этом сервере-координаторе и запускается (продолжает работу) агент синхронизации.

Далее агент вычисляет глобальное время t_{ij}^{GVT} каждого задания для всего сервера SC и по очереди опрашивает все агенты AgTrf из кластера. Пересылается полученные

модельные времена всем агентам AgTrf, которые “безопасно” уменьшают очереди входящих и исходящих сообщений и СБС, удаляя устаревшие сообщения.

Непротиворечивость моделей в точках сохранения необходима для обеспечения отказоустойчивости модели (в случае отказа одного из серверов информация оказывается продублированной на остальных.) Работа агента защищена от сбоев – используется распределенная транзакция с двухфазным подтверждением 2PC. Первая фаза начинается перед определением агентом глобального виртуального модельного времени t_{ij}^{GVT}. В контрольных точках все сущности и все списки на серверах одинаковы, чем и достигается непротиворечивость имитации. Обновление данных о сущностях происходят путем посылки сообщений с модельным временем t_{ij}^{GVT} о состояниях сущностей агентам AgTrf нужных серверов, на которых, в случае опережения времени t_{ij}^{GVT}, происходят откаты. При успешном проведенной фазы обновления, выполняется фаза подтверждения транзакции и на каждом сервере выполняется фиксация транзакции (commit).

СУБД Oracle поддерживает систему контрольных точек [15], которые используются для восстановления состояния заданий серверов и управляются с помощью вызова команд:

SAVEPOINT POINT1 – установка контрольной точки POINT1;

ROLLBACK TO POINT1 – возврат состояния к установленной точке POINT1.

В случае неудачного обновления производится откат до успешной предыдущей контрольной точки.

На основе полученной от агента AgPwr информации вычисляется время следующего запуска, и, наконец, агент AgSnc переходит в режим ожидания.

Агент производительности сервера AgPwr периодически оценивает загруженность процессора и размеры СБС, СТС, входящей и выходящей очередей агента приёма-передачи сообщений. При размерах списков больше пороговых происходит неотложный запуск агента AgSpl с использованием оператора $A_i^{AgPwr} : P_i(A_i) \to P_i(A_i \cup \{A_i^{AgSpl}\})$, агента AgSnc с использованием оператора $A_i^{AgPwr} : P_i(A_i) \to P_i(A_i \cup \{A_i^{AgSnc}\})$ и (или) агента AgGbr с использованием оператора $A_i^{AgPwr} : P_i(A_i) \to P_i(A_i \cup \{A_i^{AgGbr}\})$.

Данные о загруженности процессора сохраняются в Oracle с помощью системного пакета dbms_application_info [12] и доступны другим агентам и заданиям.

Агент сборки мусора AgGbr предназначен для удаления неиспользуемых моделей. Когда агент AgPwr определяет небольшую загрузку сервера, запускается агент AgGbr, который удаляет завершившиеся программы и их статистические отчеты на каждом

сервере после заданного времени ожидания. Работа агента описывается оператором $A_i^{AgGbr} : P_i(S_i, F_i, T_i) \to P_i(S_i \setminus F_i, F_i, T_i \setminus \{t_{ij} \mid \forall j, s_{ij} \in F_i\})$.

4. Заключение. В статье предложена агентная архитектура распределенной дискретно-событийной системы имитационного моделирования, построенная на основе СУБД Oracle. Использование агентов в таких системах повышает масштабируемость, отказоустойчивость и надежность, при незначительном снижении производительности из-за избыточности данных и поддержания целостности имитаций. Перспективным является реализация WEB-интерфейса пользователя для доступа к системе OpenGPSS через Интернет. Для этого зарегистрирован домен http://www.simulation.kiev.ua.

Список ссылок

1. SPEEDES. http://www.speedes.com

2. Mascarenhas E., Knop F., Vernon R. ParaSol: A multithreaded system for parallel simulation based on mobile threads. Winter Simulation Conference 1995.

3. Шрайбер Томас Дж. Моделирование с использованием GPSS. – М.: Машиностроение, 1980. – 593 с.

4. Томашевский В.Н., Жданова Е.Г. Имитационное моделирование в среде GPSS. – М.: Бестселлер, 2003. – 416 с.

5. Воеводин В. В., Воеводин В. В. Параллельные вычисления. Научное издание. – СПб.: BHV.

6. Киевский Центр Имитационного Моделирования. http://www.simulation.kiev.ua

7. WebGPSS. http://www.webgpss.com

8. Richard M. Fujimoto. Parallel And Distributed Simulation Systems. Wiley, 2000.

9. Bagrodia R., Meyer R., Takai M. and others. Parsec: A Parallel Simulation Environment for Complex Systems. /Computer, October 1998, – P. 77-85.

10. Вознесенская Т. В. Математическая модель алгоритмов синхронизации времени для распределенного имитационного моделирования. // Материалы V Всероссийской научно-технической конференции "Повышение эффективности методов и средств обработки информации " (12 мая - 15 мая 1997, Тамбов), стр. 193.

11. Wooldridge M., Jennings R. Intelligent Agents: Theory and Practice. Knowledge Engineering Review, Jan 1995.

12. Кайт Том. Oracle для профессионалов. Книга 2. Расширение возможностей и защита: Пер. с англ. – М.: ДиасофтЮП, 2003. – 848 с.

13. Ахо Альфред, Сети Рави, Ульман Джеффри. Компиляторы: принципы,технологии, инструменты".: Пер. с англ. – М.: Издательский дом "Вильямс", 2001. – 768 с.

14. Таненбаум Э., Стеен ван М. Распределенные системы. Принципы и парадигмы. – СПб.: Питер, 2003. – 877 с.

15. Кайт Том. Oracle для профессионалов. Книга 1. Архитектура и основные особенности: Пер. с англ. – М.: ДиасофтЮП, 2003. – 672 с.