

УДК 004.94

**АГЕНТНАЯ АРХИТЕКТУРА РАСПРЕДЕЛЕННОЙ  
ДИСКРЕТНО-СОБЫТИЙНОЙ СИСТЕМЫ  
ИМИТАЦИОННОГО МОДЕЛИРОВАНИЯ OpenGPSS**

**В.Н. ТОМАШЕВСКИЙ, Д.Г. ДИДЕНКО**

Рассматриваются вопросы применения программных агентов для проектирования распределенной дискретно-событийной системы имитационного моделирования OpenGPSS, построенной на основе серверов СУБД Oracle с применением оптимистического алгоритма Джефферсона синхронизации модельного времени. Исследуются проблемы обеспечения отказоустойчивости системы моделирования путем репликации и распределения нагрузки методом переконфигурации системы в случае отказа одного из серверов.

**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). 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 то же время язык GPSS широко используется в мире, недаром одна из последних его версий названа GPSS World [4]. Поэтому актуальна 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 давать 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 систем 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].

### 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).

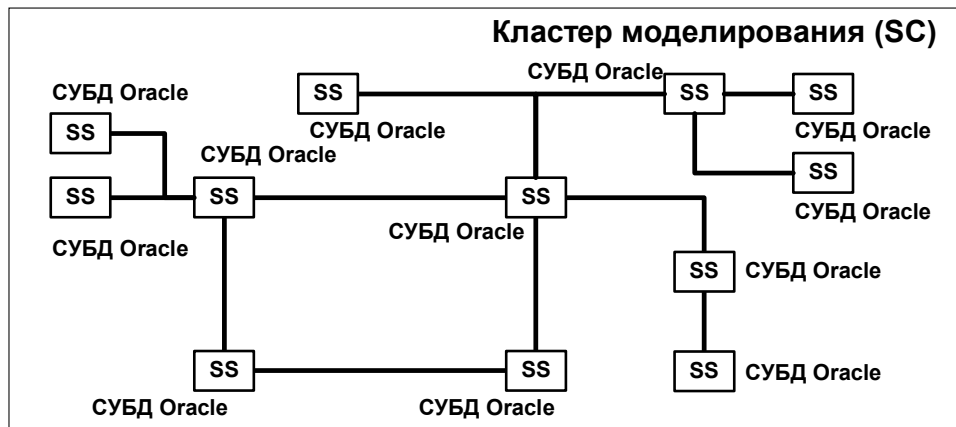


Рис. 1. Структура системы OpenGPSS

Дадим некоторые определения, необходимые для описания архитектуры системы OpenGPSS. Под *сущностями* будем понимать множество объектов (блоков) языка GPSS [3]: одноканальные (Facility) и 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] со 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, путем переноса соответствующего задания в списке заданий.

- **Активность** (pro-activeness) — способность ставить цели и инициативно действовать для их достижения. 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 приемо-передачи данных.

Система OpenGPSS реализует 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].

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 обеспечивает большую производительность, хотя и увеличивает сложность реализации системы. Для упрощения работы пользователей реализован WEB-интерфейс в Интернете. Обеспечение отказоустойчивости достигается с помощью механизмов репликации и других способов защиты данных.

## АГЕНТНАЯ АРХИТЕКТУРА СЕРВЕРА МОДЕЛИРОВАНИЯ

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

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

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

$$\mathbf{V}_{iz} = \begin{cases} 1, & \text{если существует связь между } i\text{-м и } z\text{-м сервером} \\ 0 & \text{в противном случае,} \end{cases}$$

$$i = \overline{1, n}, z = \overline{1, n}.$$

Пусть  $S_{ij}$  —  $j$ -е задание пользователя на  $i$ -м сервере,  $i = \overline{1, n}$ ,  $j = \overline{1, k_i}$ , где  $k_i$  — количество заданий на  $i$ -м сервере. Обозначим  $F_i$  — устаревшие

задания пользователя  $i$ -го сервера. Множество модельных времен  $i$ -го сервера определим как  $T_i = \{t_{i1}, t_{i2}, \dots, t_{ij}, \dots, t_{ik_i}\}$ , а результаты выполнения  $j$ -го задания пользователя на  $i$ -м сервере обозначим  $R_{ij}$ .

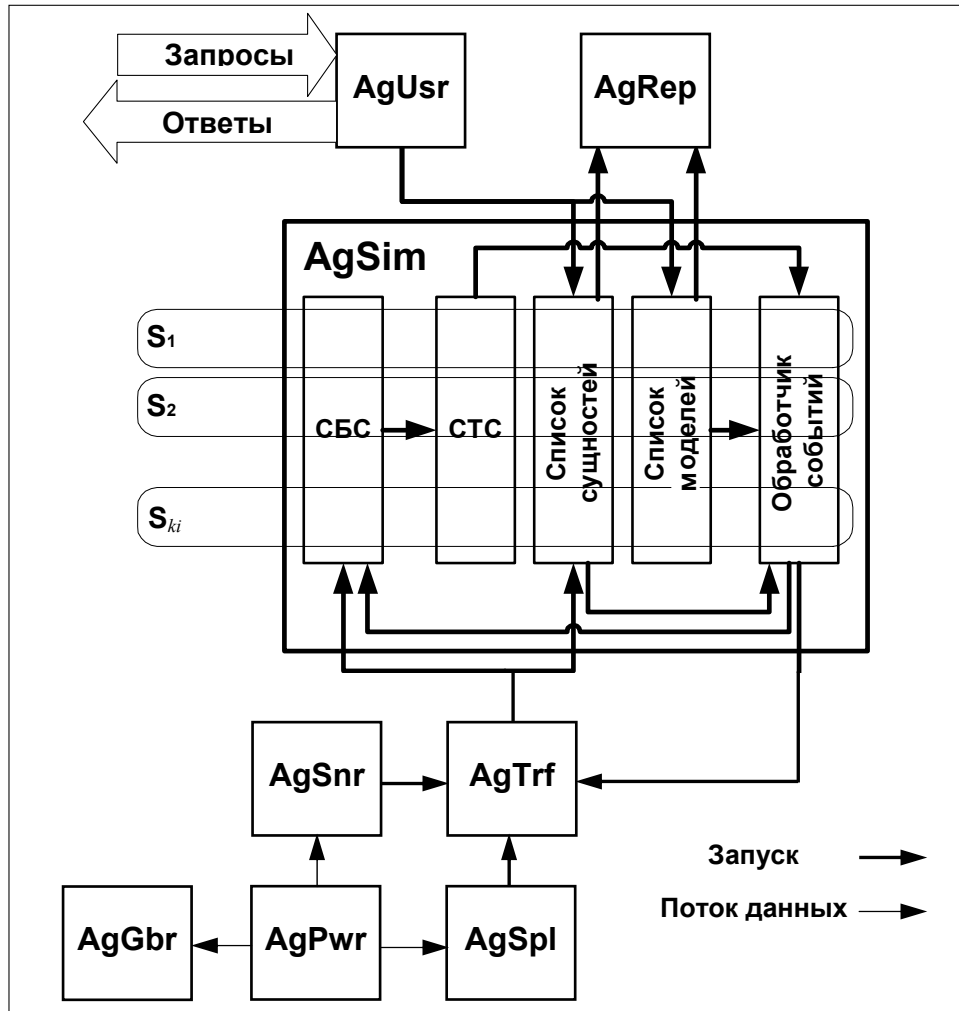


Рис. 2. Агентная архитектура сервера моделирования

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

Обозначим  $D_{ij}$  очереди исходящих сообщений для  $j$ -го задания на  $i$ -м сервере, а  $Q_{ij}$  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 кластера моделирования —  $P = \bigcup_{i=1}^n P_i(A_i, S_i, F_i, T_i, R_i, M_i, Q_i, D_i)$ .

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

$$A_{ij}^{\text{AgSim}} : P_i(\{t_{i1}, t_{i2}, \dots, t_{ij}, \dots, t_{ik_i}\}) \rightarrow P_i(\{t_{i1}, t_{i2}, \dots, t'_{ij}, \dots, t_{ik_i}\}), \quad (1)$$

где  $t_{ij} \leq t'_{ij}$ .

После завершения моделирования  $j$ -го задания агент имитационного моделирования сохраняет результаты в списке результатов  $R_i$ .

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

Агент взаимодействия с пользователем AgUsg 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 сервера новую задачу проведения имитации с помощью оператора

$$A_i^{\text{AgUsg}} : P_i(A_i, S_i, T_i) \rightarrow P_i(A_i \cup \{A_{ik_i+1}^{\text{AgSim}}\}, S_i \cup \{s_{ik_i+1}\}, T_i \cup \{t_{ik_i+1}\}).$$

Любое изменение структуры GPSS-модели (перегрузка блоков) приводит к «устареванию» старой задачи и созданию на ее основе новой.

При удалении  $j$ -го задания,  $1 \leq j \leq k_i$ , 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

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

Агент синхронизации AgSnc определяет окончание моделирования, когда входящие списки агентов приема-передачи сообщений для  $j$ -го задания, списки СБС и СТС пусты, и проводит распределенное подтверждение — синхронизирует сущности на серверах. После этого по запросу пользователя агент AgUsg 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.

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

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 может 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, исключается использованием алгоритма двухфазного подтверждения 2PC, описанного в работе [14]. Когда агент-инициатор блокирует заданную сессию для всех агентов репликации на всех серверах и останавливает обработку данной сессии, то он обновляет модель на всех серверах, и только потом подтверждает и (или) отменяет изменения.

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

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

Оптимистическая синхронизация, используемая в системе, основывается на обработке событий (несмотря на то, что позже могут возникнуть более ранние события), определении неправильного порядка событий и использовании отката в этом случае.

Неправильный порядок событий возникает тогда, когда приходят сообщения, намеченные на более раннее, 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  $D$  и входящих  $Q$  сообщений.

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  $i$ -м сервере при откате решается средствами Oracle.

При приеме 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.

Анτισообщение — специальное сообщение, которое используется для отмены недействительного сообщения. Оно определяется как обычное сообщение с «перевернутым» первым разрядом, и для него нет необходимости

передавать всю информацию (например, про транзакт), а достаточно передать лишь идентификатор (уникальный 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, то оба сообщения удаляются из нее. 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, то 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 раннее сообщение (отстающее). Откат предусматривает восстановление состояния  $j$ -го задания  $i$ -го сервера SS (состояние значений переменных) до состояния раннего сообщения и отмену недействительных сообщений путем посылки соответствующих антисообщений.

От количества контрольных точек существенно 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 времен всех заданий, агент AgTrf удаляет из входящих и исходящих списков устаревшие события, модельное время которых меньше  $t_{ij}^{GVT}$  [8].

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

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

При обработке транзактов агентом имитационного моделирования AgSim возможны два режима передачи сообщений: локальный и глобальный, которые зависят от маркеров начала и 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 приемо-передачи сообщений (2), который рассылает этот транзакт следующему серверу из списка  $M_{ij}$ .

$$A_{ij}^{AgSim} : P_i(\{t_{i1}, t_{i2}, \dots, t_{ij}, \dots, t_{ik}\}, D_i) \rightarrow$$

$$\rightarrow P_i(\{t_{i1}, t_{i2}, \dots, t'_{ij}, \dots, t_{ik}\}, D_i \cup \{d_{ij} \mid m_{ijz} \in M_{ij}, \mathbf{B}_{iz} = 1\}), \quad (2)$$

где  $t_{ij} \leq t'_{ij}$ .

Если при обработке происходит изменение состояний сущностей, то всегда генерируется сообщение об этом и далее передается агенту AgTrf, который, снова используя  $M_{ij}$ , передает сообщения нужным серверам.

Агент разделения нагрузки AgSpl распределяет ее среди серверов в кластере в зависимости от структуры GPSS-модели (статистический анализ) и нагрузки серверов, полученной от агента AgPwr (динамический анализ). Работа агента описывается следующим оператором:

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

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

На агента 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 переконфигурации не происходит до очередного запуска агента разделения нагрузки.

Основные задачи агента синхронизации AgSnc — глобальное управление выполнением имитаций и поддержание непротиворечивости моделей в точках сохранения.

В системе OpenGPSS применяется оптимистический подход синхронизации, и агент AgSnc используется как механизм глобального контроля. Он вычисляет глобальное виртуальное модельное время  $t_{ij}^{\text{GVT}}$  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  $t_{ij}^{\text{GVT}}$ , 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 всех агентов AgTrf из кластера. Полученные модельные времена пересылаются 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, так как 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 происходит 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 проведении фазы обновления выполняется фаза подтверждения транзакции и на каждом сервере — 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 приема-передачи сообщений. При размерах списков больше пороговых происходит неотложный запуск агента AgSpl с использованием оператора  $A_i^{AgPwr} : P_i(A_i) \rightarrow P_i(A_i \cup \{A_i^{AgSpl}\})$ , AgSnc с использованием  $A_i^{AgPwr} : P_i(A_i) \rightarrow P_i(A_i \cup \{A_i^{AgSnc}\})$  и (или) AgGbr с оператором  $A_i^{AgPwr} : P_i(A_i) \rightarrow P_i(A_i \cup \{A_i^{AgGbr}\})$ .

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 AgGbr, который удаляет завершившиеся программы и их статистические отчеты на каждом сервере после заданного време-

ни ожидания. Работа агента описывается оператором  $A_i^{\text{AgGbr}} : P_i(S_i, F_i, T_i) \rightarrow P_i(S_i \setminus F_i, F_i, T_i \setminus \{t_{ij} \mid \forall j, s_{ij} \in F_i\})$ .

## ЗАКЛЮЧЕНИЕ

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>.

## ЛИТЕРАТУРА

1. *SPEEDES*. <http://www.speedes.com>.
2. *Mascarenhas E., Knop F., Vernon R.* ParaSol: A multithreaded system for parallel simulation based on mobile threads // Proceeding of the 27-th conference on Winter simulation — Arlington. Virginia, 1995. — P. 690–697.
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 Вл.В.* Параллельные вычисления. — СПб.: BHV, 2002. — 608 с.
6. *Киевский центр имитационного моделирования*. <http://www.simulation.kiev.ua>.
7. *WebGPSS*. <http://www.webgpss.com>.
8. *Fujimoto Richard M.* Parallel And Distributed Simulation Systems. — New York: Wiley Interscience, 2000. — 320 p.
9. *Parsec: A Parallel Simulation Environment for Complex Systems* / R. Bagrodia, R. Meyer, M. Takai et al. // Computer. — October. — 1998. — P. 77–85.
10. *Вознесенская Т.В.* Математическая модель алгоритмов синхронизации времени для распределенного имитационного моделирования // Материалы V Всерос. научн.-техн. конф. «Повышение эффективности методов и средств обработки информации», Тамбов, 12 – 15 мая 1997 г. — С. 193.
11. *Wooldridge M., Jennings R.* Intelligent Agents: Theory and Practice // Knowledge Engineering Review. — 1995. — 10 (2). — P. 115–152.
12. *Кайт Т.* Oracle для профессионалов. Кн. 2. Расширение возможностей и защита / Пер. с англ. — М.: ДиасофтЮП, 2003. — 848 с.
13. *Ахо А., Сети Р., Ульман Дж.* 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 Т.* Oracle для профессионалов. Кн. 1. Архитектура и основные особенности / Пер. с англ. — М.: ДиасофтЮП, 2003. — 672 с.

Поступила 16.03.2006