# МОДЕЛЬ УПРАВЛЕНИЯ ПРОЦЕССАМИ В ОБЩЕЙ СЕМАНТИЧЕСКОЙ ПАМЯТИ ИНТЕЛЛЕКТУАЛЬНЫХ СИСТЕМ

Зотов Н. В.
Кафедра интеллектуальных информационных технологий,
Белорусский государственный университет информатики и радиоэлектороники
Минск, Республика Беларусь
E-mail: n.zotov@bsuir.by

*В данной работе исследуется задача распределенной обработки информации в общей семантической памяти. Работа представляет новую модель управления процессами, отвечая на растущие требования по обработке информации в многопоточных средах и открытых вычислительных системах.*

## Введение

Ранее в работах [1, 2], посвящённых описанию программной платформы для интеллектуальных систем, разрабатываемых по принципам Технологии OSTIS (программной платформе ostis-систем), была рассмотрена программная реализация общей семантической памяти (sc-памяти), а также подробно описана реализация её программного интерфейса. Цель текущей работы заключается в решении задачи распределённой обработки информации в общей семантической памяти, функционирующей в многопоточной среде. Актуальность работы обусловлена увеличением объемов обрабатываемой информации, ростом требований к скорости обработки информации и недостаточной производительностью современных открытых вычислительных систем. Новизна работы заключается в предлагаемой модели управления процессами в общей семантической памяти.

## I. Модель sc-памяти в платформе OSTIS-систем

*Программную платформу ostis-систем* можно рассматривать как систему управления графовыми базами данных (СУБД) [1]. Однако она имеет ряд особенностей по сравнению с современными СУБД [1]:

- обрабатываемые графовые конструкции являются конструкциями SC-кода (sc-конструкциями),
- в её основе заложены событийно-ориентированная обработка информации и основанный на ней многоагентный подход.

Под текущей *программной реализацией sc-памяти* понимается компонент программной платформы ostis-систем, осуществляющий хранение sc-конструкций и доступ к ним через соответствующий программный интерфейс [3]. В общем случае *sc-память* выполняет следующие задачи:

- хранение *sc-конструкций*,
- хранение внешних по отношению к *SC-коду* информационных конструкций (файлов ostis-системы),
- доступ (создание, чтение и удаление) к sc-конструкциям, реализуемый через соответствующий *программный интерфейс*,
- управление процессами (потоками).

В рамках данной *программной реализации ostis-платформы sc-память* представлена в виде набора фиксированного размера N, состоящего из *сегментов* ($S = \langle s_1, s_2, ..., s_i, ..., s_n \rangle$), каждый из которых представляет собой набор фиксированного размера M, состоящей из *элементов sc-памяти* (или ячеек) ($s_i = \langle e_{i1}, e_{i2}, ..., e_{ij}, ..., e_{im} \rangle$, $e_{ij} \in E$). Каждый *элемент sc-памяти* $e_{ij}$ (в т.ч. *элемент sc-памяти, соответствующий sc-узлу* $n_k$) содержит синтаксический и семантический классы, уровни доступа соответствующего sc-элемента, выраженные в виде бинарных строк $t \in T$ и $r \in R$ соответственно, *sc-адреса элементов sc-памяти, соответствующих первым входящему и выходящему sc-коннекторам*, а также *число входящих sc-коннекторов в заданный sc-элемент* и *число выходящих sc-коннекторов из заданного sc-элемента*. При этом каждый *элемент sc-памяти, соответствующий sc-коннектору* $c_h$ дополнительно содержит *sc-адреса элементов sc-памяти, соответствующих начальному и конечному sc-элементам этого sc-коннектора*, *sc-адреса элементов sc-памяти, соответствующих предыдущим и следующим входящим и выходящим sc-коннекторам* ($E = N \cup C,\ n_k \in N,\ c_h \in C$).

*Программная реализация sc-памяти* включает также *программную реализацию файловой памяти*, которая обеспечивает хранение содержимого $L$ *файлов ostis-систем* $F$ ($F \subset N,\ N \times L \to F, F \times L \to F$), то есть *внешних информационных конструкций* (линейных текстов, изображений, видео и т.д.), а также выгрузку sc-памяти на дисковое пространство. В качестве структур данных в файловой памяти используются префиксные деревья и линейные списки.

*Программный интерфейс реализации sc-памяти* представляет собой набор методов, позволяющих работать с sc-памятью. Его модель можно определить по формуле 1.

$$PI = N^T \times C^{E\times E\times T} \times F^{N\times L}\times$$
$$\times\{E^{E\times\{E\cup T\}\times E}\cup\ E^{\{E\cup T\}\times T\times E}\cup E^{E\times T\times\{E\cup T\}}\}\times$$
$$\times\ T^E \times\ F^L \times L^F \times \{\top,\bot\}^E \qquad (1)$$

## II. Модель управления процессами в sc-памяти

В *реализации sc-памяти* логически выделяется *программная реализация подсистемы управления процессами в ней*. Она выполняет следующие задачи:

- создание, чтение и удаление процессов,
- синхронизацию процессов,
- распределение sc-памяти между процессами,
- управление событиями.

В sc-памяти, обычно, используются потоки процессов. Поток может возникнуть в результате инициированного по событию sc-агента или в ответ на запрос к sc-памяти через сеть, выполняющих операции чтения и записи в ней.

В предыдущих версиях программной платформы ostis-систем для обеспечения согласованного доступа к данным в разделяемой памяти (sc-памяти) применялись мьютексы и атомарные операции. В версии 0.9.0 для согласования доступа к структурам данным в sc-памяти были реализованы мониторы [3], а также методы для захвата и освобождения ресурсов для потоков-читателей $P_r$ (далее – читателей) и потоков-писателей $P_w$ (далее – писателей) ($P = P_r \cup P_w$).

Монитор $m \in M$ состоит из: счётчика активных читателей, флага, показывающего, активен ли в данный момент какой-либо писатель, очереди читателей и писателей, а также мьютекса, используемого для синхронизации доступа к этим элементам монитора. Очередь читателей и писателей представляет собой последовательность запросов на захват определенного ресурса. Каждый запрос включает в себя уникальный идентификатор потока, тип потока (читатель или писатель) и условную переменную, позволяющую обмениваться сообщениями между процессами (потоками). Эта очередь гарантирует, что ни один поток не останется «голодным». Важно отметить, что захват ресурса методом монитора для чтения одним потоком не блокирует захват этого же ресурса методом монитора для чтения другим потоком.

Распределение sc-памяти писателям осуществляется посегментно с использованием специализированной хеш-таблицы $T_{ps}$, которая позволяет определить, занят ли данный незаполненный сегмент sc-памяти другим писателем ($s : S \to S_{ne},\ T_{ps} \subseteq P_w \times S_{ne}$). Незаполненный сегмент sc-памяти может представлять собой как сегмент с невыделенными ячейками, так и сегмент с освобожденными ячейками. При распределении sc-памяти сначала происходит поиск незаполненных сегментов, которые не используются другими писателями. Если такие сегменты не найдены, то выделяются новые сегменты. Если в sc-памяти нет доступного места для новых сегментов, писатели могут использовать сегменты из списка занятых незаполненных сегментов. Для обеспечения согласованного доступа к сегментам для чтения каждый сегмент содержит уникальный монитор ($s_m : S \times M \to S_m$).

Также кроме сегментов мониторы временно назначаются элементам sc-памяти и регестрируемым событиям в ней. Мониторы элементов sc-памяти хранятся в специализированной хеш-таблице $T_{em}$. С помощью данных мониторов синхронизируется доступ к информации о sc-элементе, содержащейся в элементе sc-памяти ($T_{em} \subset A \times M$). События $v \in V$, их типы $t_v \in T_v$, sc-агенты, подписанные на них $A_v \subset A$, а также мониторы этих событий хранятся в единой хеш-таблице $T_v$. С помощью данных мониторов синхронизируются подписка и отписка на события через единую хеш-таблицу, а также инициирование самих sc-агентов ($v_i = \langle t_v^i, A_v^i, m_v^i\rangle, t_v^i \in T_v, A_v^i \subset A_v, m_v^i \in M, v_m : V \times M \to V_m$).

## III. Заключение

Предложенная модель управления общей семантической памятью позволяет эффективно отслеживать и синхронизировать параллельный доступ к данным. Реализация этой модели демонстрирует значительное увеличение (на 2-3 порядка) пропускной способности параллельного выполнения задач в сравнении с предыдущими версиями платформы. Однако для обеспечения (причинной, последовательной) консистентности процессов и их операций кроме уровня данных необходимо управлять уровнем знаний [4].


1. Zotov, N. Software platform for next-generation intelligent computer systems = Программная платформа для интеллектуальных компьютерных систем нового поколения / N. Zotov // Open Semantic Technologies for Intelligent Systems (OSTIS-2022) : сборник научных трудов / Белорусский государственный университет информатики и радиоэлектроники ; редкол.: В. В. Голенков [и др.]. – Минск, 2022. – Вып. 6. – С. 297–326.
2. Zotov, N. Implementation of Information Retrieval Subsystem in the Software Platform of ostis-systems = Реализация информационно-поисковой подсистемы в программной платформе ostis-систем / N. Zotov // Open Semantic Technologies for Intelligent Systems (OSTIS) : сборник научных трудов / Белорусский государственный университет информатики и радиоэлектроники ; редкол.: В. В. Голенков [и др.]. – Минск, 2023. – Вып. 7. – С. 77–94.
3. Cole M. I. Algorithmic skeletons: structured management of parallel computation. – London : Pitman, 1989.
4. Miret L. P. Consistency models in modern distributed systems. an approach to eventual consistency //Master. MA thesis. Universitat Politecnica de Valencia, Spain. – 2014.