СТАТЬИ

УДК 004.42
DOI 10.17513/snt.40812

МЕТОД СНИЖЕНИЯ КОНТЕКСТНОГО ПЕРЕПОЛНЕНИЯ В АДАПТИВНЫХ МУЛЬТИАГЕНТНЫХ КОНВЕЙЕРАХ НА ОСНОВЕ СУММАРИЗАЦИИ КОНТЕКСТА НА ГИПЕРРЁБРАХ

Алпатов А. Н. ORCID ID 0000-0001-8624-1662

Федеральное государственное бюджетное образовательное учреждение высшего образования «МИРЭА – Российский технологический университет», Москва, Российская Федерация, e-mail: aleksej01-91@mail.ru

Рост сложности распределённых конвейеров, а также развитие мультиагентных адаптивных самоорганизующихся распределённых конвейеров обработки данных и расширение спектра используемых протоколов взаимодействия порождают проблему контекстного переполнения, при котором объём передаваемых между агентами данных превышает практические ограничения контекстного окна и пропускной способности. Целью исследования является разработка метода снижения контекстного переполнения в таких конвейерах, обеспечивающего ограниченность объёма локального контекста на каждом узле в условиях динамически перестраиваемой топологии и гетерогенности протоколов взаимодействия. В качестве методологической основы использованы принципы формализации распределённых систем, теория гиперграфов, а также методы математического моделирования динамических систем. В работе адаптивный мультиагентный конвейер описан как временной гиперграф, а накопление контекста представлено двухуровневой моделью состояния узла. В работе представлен метод адаптивной суммаризации контекста, реализуемый на уровне гиперрёбер временного гиперграфа в виде промежуточного программного слоя. Ключевым элементом данного подхода является перенос логики редукции на уровень гиперребра, что обеспечивает детерминированную обработку контекста на промежуточном слое и позволяет реализовать совмещённый реактивный и проактивный режимы сжатия контекста. Сформулированы и доказаны утверждения об устойчивости и ограниченности локального контекста при срабатывании детектора, что определяет условия предотвращения переполнения. Сделан вывод о том, что предложенный подход обеспечивает ограниченность объёма контекста, предотвращает накопление ошибок в циклических взаимодействиях и позволяет повысить масштабируемость распределённых конвейеров без модификации протоколов интеграции агентов.

Ключевые слова: мультиагентные системы, адаптивные вычислительные конвейеры, самоорганизующиеся распределённые системы, контекстное переполнение, временной гиперграф

REDUCING CONTEXT OVERFLOW IN ADAPTIVE MULTI-AGENT PIPELINES VIA HYPEREDGE-BASED CONTEXT SUMMARIZATION

Alpatov A. N. ORCID ID 0000-0001-8624-1662

Federal State Budget Educational Institution of Higher Education «MIREA – Russian Technological University», Moscow, Russian Federation, e-mail: aleksej01-91@mail.ru

The increasing complexity of distributed pipelines, along with the development of multi-agent adaptive self-organizing distributed data processing pipelines and the expansion of the range of interaction protocols, leads to the problem of context overflow, in which the volume of data exchanged between agents exceeds the practical limits of context windows and bandwidth capacity. The objective of this study is to develop a method for reducing context overflow in such pipelines, ensuring bounded local context at each node under conditions of dynamically reconfigurable topology and heterogeneous interaction protocols. The methodological foundation of this work is based on principles of distributed systems formalization, hypergraph theory, and mathematical modeling of dynamic systems. In this work, the adaptive multi-agent pipeline is described as a temporal hypergraph, while context accumulation is represented by a two-level node state model. This paper presents a method for adaptive context summarization implemented at the level of hyperedges of a temporal hypergraph in the form of an intermediate software layer. A key element of the proposed approach is the relocation of reduction logic to the hyperedge level, which enables deterministic context processing within the intermediate layer and supports the combined use of reactive and proactive context compression modes. Formal statements on the stability and boundedness of local context under detector activation are formulated and proven, defining the conditions for overflow prevention. It is concluded that the proposed approach ensures bounded context size, prevents error accumulation in cyclic interactions, and improves the scalability of distributed pipelines without requiring modifications to agent integration protocols.

Keywords: multi-agent systems, adaptive computational pipelines, self-organizing distributed systems, context overflow, temporal hypergraph

Введение

Многие современные вычислительные конвейеры обработки данных в настоящее время вышли за рамки традиционных ETL (англ. Extract, Transform, Load) конвейеров, которые строились на жёсткой фиксации логики и схем взаимодействия внутри конвейера и являлись, фактически, стандартом промышленной аналитики. Это связано с тем, что традиционные ETL-конвейеры представляют собой статичные ориентированные ациклические графы задач и по своей природе не являются очень гибкими, что проявляется в случаях, когда любое изменение бизнес-требований требует переписывания кода, пересмотра схем взаимодействия и повторного развёртывания в продакшен-среде, а это требует значительных финансовых и временных расходов. В результате им на смену приходят более адаптивные мультиагентные самоорганизующиеся распределённые конвейеры, в которых вся логика обработки данных формируется на лету в режиме реального времени, исходя из контекста конкретной задачи, требуемых условий выполнения и других ограничений и условий. В основе таких реализаций конвейеров находятся автономные агенты, которые умеют самостоятельно разбирать неструктурированные запросы на естественном языке, поступающие на вход конвейера, а также самостоятельно координироваться друг с другом при решении сложных многоэтапных задач, что приводит к фактическому перестроению топологии конвейера в реальном времени. Иными словами, конвейеры переходят от жёсткой заранее запрограммированной последовательности шагов к самоорганизующимся, основанным на намерениях (англ. Intent-driven design) сценариям и целеориентированной архитектуре.

На практике для интеграции агентов между собой и с внешними инструментами используются специализированные протоколы, в частности Model Context Protocol (англ. MCP) [1; 2]. Подходы к межагентному взаимодействию подробно рассматриваются в работах [3; 4], где описываются A2A (англ. Agent-to-Agent) интерфейсы. В качестве инструментальной реализации также часто применяются фреймворки агентной коммуникации, например LangGraph [5]. Подобные решения резко снижают сложность реализации, что упрощает создание сложных мультиагентных систем. Например, распространённый в разработке протокол MCP за счёт стандартизации интерфейса превращает разрозненные API, базы данных, сторонние сервисы и службы в унифицированные контекстные серверы. Согласно текущей спецификации [2], технически это реализуется за счёт создания легковесного процесса, который может представлять собой как обычный исполняемый бинарный файл, так и легковесный контейнер, на стороне источника данных, который экспонирует возможности источника через три унифицированных примитива протокола MCP: «Инструменты» (англ. Tools), «Ресурсы» (англ. Resources) и «Инструкции» (англ. Prompts), описывающих, соответственно, исполняемые функции, ссылки на содержимое (через URI) и параметризованные шаблоны инструкций. Вызовы инструментов осуществляются через JSON-RPC, а описания передаются агенту в формате JSON Schema [2].

Такая схема интеграции приводит к тому, что любой агент в вычислительном конвейере подключается через единую унифицированную среду. Однако такое упрощение интеграции может приводить к резкому росту объёма контекстной информации, необходимой для корректной обработки потоков данных, тогда как архитектурные ограничения систем остаются в значительной мере статичными. Данная проблема усугубляется также гетерогенностью современных вычислительных конвейеров, многие из них одновременно обрабатывают потоки событий, пакетные задания, а также запросы реального времени, каждый из которых предъявляет принципиально разные требования к управлению контекстом.

Здесь важно отметить, что данная проблема проявляется по-разному и на разных уровнях в зависимости от уровня архитектуры вычислительного конвейера. Так, в вычислительных конвейерах на основе NLP-моделей контекстное переполнение проявляется как некоторая деградация семантической согласованности и характеризуется тем, что по мере роста контекстного окна точность рассуждений агента снижается нелинейно, так как модель показывает свои лучшие результаты, когда релевантная информация находится в самом начале или конце длинного запроса к модели [6]. А в рамках современных потоковых распределённых систем, реализованных, например, с использованием методологии Prompt2DAG [7] или иных подходов, данная проблема принимает уже форму переполнения буферов операторов и деградации пропускной способности, то есть той ситуации, когда скорость поступления данных превышает скорость их потребления [8]. В случае же реализации гетерогенных конвейеров механизмы деградации часто действуют одновременно и взаимно усилива-

ют друг друга, что явно проявляется при росте числа подключённых MCP-серверов, это увеличивает объём JSON описаний инструментов в контексте каждого агента, тогда как накопление промежуточных результатов JSON-RPC вызовов дополнительно насыщает контекстное окно.

В связи с этим возникает необходимость разработки специализированного метода, который позволял бы эффективно управлять контекстом в условиях высокой динамики и самоорганизации, сохраняя при этом адаптивность и масштабируемость распределённой системы.

Цель исследования – разработка метода снижения контекстного переполнения в адаптивных мультиагентных самоорганизующихся распределённых конвейерах, обеспечивающего ограниченность объёма локального контекста на каждом узле в условиях динамически перестраиваемой топологии и гетерогенности протоколов взаимодействия. В основе предлагаемого подхода лежит размещение логики адаптивной суммаризации непосредственно на уровне гиперрёбер временного гиперграфа.

Материалы и методы исследования

Исследование выполнено в форме теоретико-прикладного анализа архитектур адаптивных мультиагентных самоорганизующихся распределённых вычислительных конвейеров и механизмов управления контекстным состоянием в условиях динамически изменяемой топологии. В качестве основного материала использованы формализованные модели вычислительных конвейеров, представленных в виде временных гиперграфов, а также описания протоколов взаимодействия агентов (MCP, A2A), спецификации форматов обмена данными (JSON-RPC, JSON Schema) и структур контекстных состояний агентов. Дополнительно рассмотрены архитектурные паттерны (централизованные, децентрализованные и гибридные конвейеры), а также экспериментальные схемы распространения и аккумуляции контекста в гиперграфовых моделях.

Методологической основой исследования послужили принципы формализации распределённых систем, теории графов и гиперграфов, а также методы математического моделирования динамических систем. В качестве метода решения поставленной задачи использован разработанный подход адаптивной прогнозирующей суммаризации контекста, реализуемый на уровне гиперрёбер временного гиперграфа. Архитектурная реализация предложенного решения рассматривается в рамках промежуточного слоя, интегрируемого в вычислительный конвейер. В качестве технологической основы предполагается использование распределённых сервисных архитектур с поддержкой REST/gRPC взаимодействия, контейнеризации и масштабируемых key-value хранилищ для поддержки таблицы состояний гиперрёбер.

Результаты исследования и их обсуждение

В отличие от традиционного подхода, при котором описанные выше формы контекстного переполнения рассматриваются независимо, в данной работе оно определяется как состояние конвейера, при котором объём накопленного контекстного состояния превышает допустимый предел его результативной обработки, что сопровождается монотонным ухудшением метрик производительности и/или семантической согласованности. Такая фиксация позволяет перейти от рассмотрения отдельных механизмов деградации к совокупному анализу архитектурных особенностей адаптивных мультиагентных самоорганизующих распределённых конвейеров. В связи с этим целесообразно рассмотреть разнообразие подходов к построению таких конвейеров, выделить их ключевые структурные и функциональные характеристики.

Как было отмечено выше, на практике выделяют большое количество подходов для реализации мультиагентных адаптивных самоорганизующихся распределённых конвейеров. В данной работе выделяются три основных класса архитектур, различающихся по степени децентрализации управления и механизмам формирования топологии.

Так, одним из распространённых архитектурных паттернов является конвейер с центральным оркестратором [9], в котором центральный агент-оркестратор управляет набором специализированных агентов-воркеров. На рисунке 1 показана реализация данного архитектурного паттерна.

При таком подходе топология конвейера формируется динамически, но под контролем единого центра принятия решений. Несмотря на широкое распространение такого рода решения, оно может приводить к потенциальному возникновения узкого места в архитектуре и повышать риск его контекстного переполнения при масштабировании числа агентов. Это связано с тем, что оркестратор в адаптивном конвейере получает результаты от всех агентов и передаёт их в следующую итерацию, и уже после нескольких итераций его контекст заполняется сырыми выводами агентов, цепочками рассуждений, историей вызовов инструментов.

Рис. 1. Мультиагентный адаптивный конвейер с центральным оркестратором
Источник: составлено автором

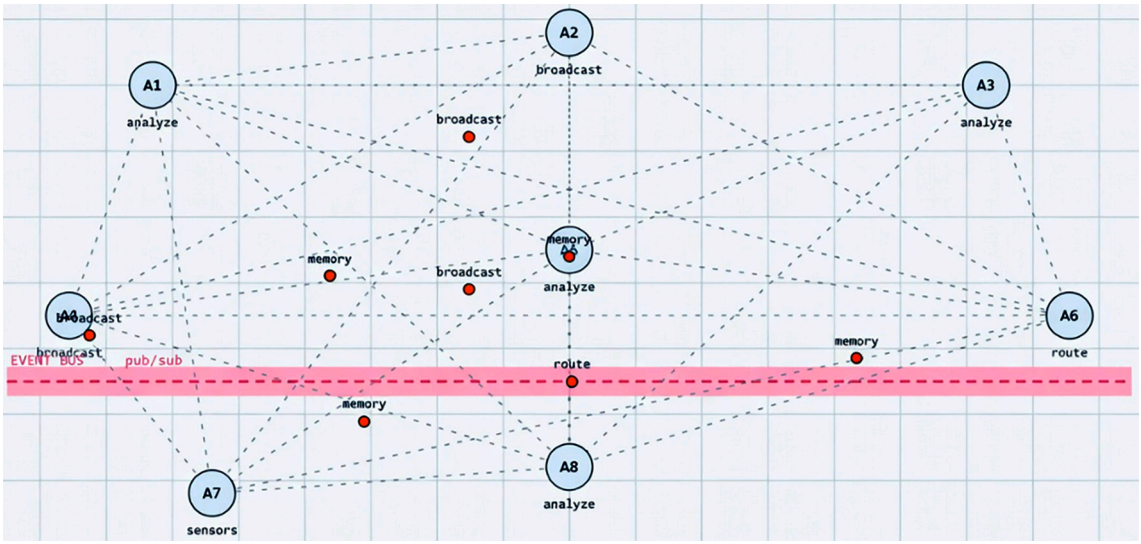

Рис. 2. Пример децентрализованной роевой архитектуры вычислительного конвейера
Источник: составлено автором

В самоорганизующихся системах агенты могут порождать подагентов динамически, а в случае если агенты представлены языковыми моделями, то записывать в общую шину памяти промежуточные схемы рассуждений. В результате объём контекста оркестратора растёт по оценке

$$O(n \times d \times i), \quad (1)$$

где n – число агентов,
d – глубина рассуждений,
i – число итераций.

Децентрализованные роевые архитектуры [10] исключают единый управляющий центр. Агенты взаимодействуют напрямую через протоколы интеграции, обмениваясь сообщениями по принципу публикации/подписки или через общую шину событий, без единого центра управления. Здесь топология вычислительного конвейера возникает и меняется в процессе самоорганизации, за счёт локальных взаимодействий между агентами. На рисунке 2 показан пример такой архитектуры из восьми равно-

правных агентов, координирующих действия (analyze, validate, broadcast, memory) через pub/sub-шину.

Гибридные адаптивные конвейеры сочетают элементы обеих моделей. Здесь высокоуровневое планирование осуществляется централизованно, тогда как исполнение и координация на уровне доменов делегируются автономным подагентам. Такая реализация показана на рисунке 3.

Для единого анализа всех трёх выделенных классов введём математическую модель адаптивного конвейера. Традиционное моделирование таких систем как статических направленных ациклических графов, имеет ряд существенных ограничений при описании самоорганизующихся систем. Обычные рёбра в направленных ациклических графах (англ. Directed Acyclic Graph, DAG) отражают только бинарные взаимодействия (REST/HTTP API вызовы, gRPC, Point-to-Point и т. д.) от источника к приёмнику, без возможности простым способом моделировать многостороннюю координацию, проявляющуюся в виде агрегации результатов от нескольких MCP-серверов, консенсуса нескольких агентов или широковещательной рассылки. Кроме того, динамика топологии в таких моделях задаётся внешним параметром, что затрудняет совместный анализ структурных изменений и накопления контекстного состояния. В рамках данной работы для устранения данной проблемы предлагается описать адаптивный мультиагентный самоорганизующийся конвейер как временной гиперграф.

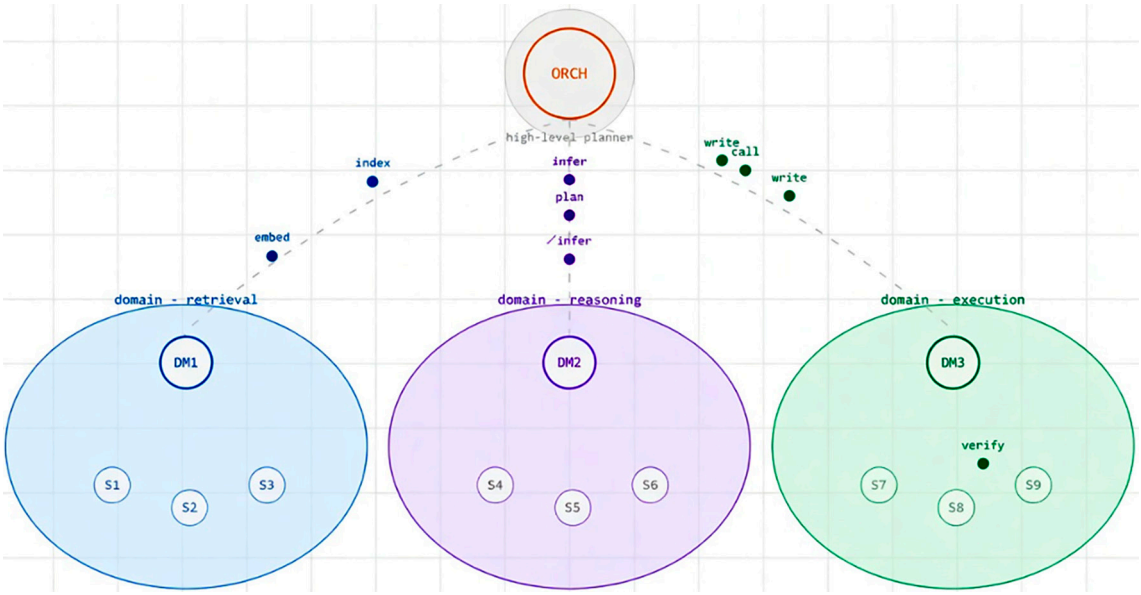

Рис. 3. Пример гибридной архитектуры вычислительного конвейера
Источник: составлено автором

Формальная модель временного гиперграфа

Пусть в момент времени t адаптивный мультиагентный самоорганизующийся конвейер описывается временным гиперграфом с динамическим множеством узлов

$$H(t)=(V(t), E(t), T), \qquad (2)$$

где V(t) – конечное, но изменяемое во времени множество узлов; E(t) – множество активных гиперрёбер в момент времени t.

Каждое гиперребро $e=(S_e,T_e,[\tau_{start},\tau_{end}])$ определяется непустыми непересекающимися подмножествами источников $S_e \subset V(t)$ и приёмников $T_e \subset V(t)$ ($S_e \cap T_e=\varnothing$), а также временным интервалом существования источника; $T:E \to R_+ \times R_+$ – функция, сопоставляющая каждому гиперребру интервал его активности.

Представленная выше формализация с использованием гиперграфа даёт строгое и адекватное описание, а также отражает ключевые особенности самоорганизующихся конвейеров. Так, например, атомарные вызовы инструментов через протокол MCP в такой модели интерпретируются как гиперрёбра с единичной кардинальностью множеств $e = (\{v_{agent}\},\{v_{mcp}\},[\tau,\tau])$, то есть гиперребро с $|S_e| = |T_e| = 1$ структурно эквивалентно стандартной дуге ориентированного графа.

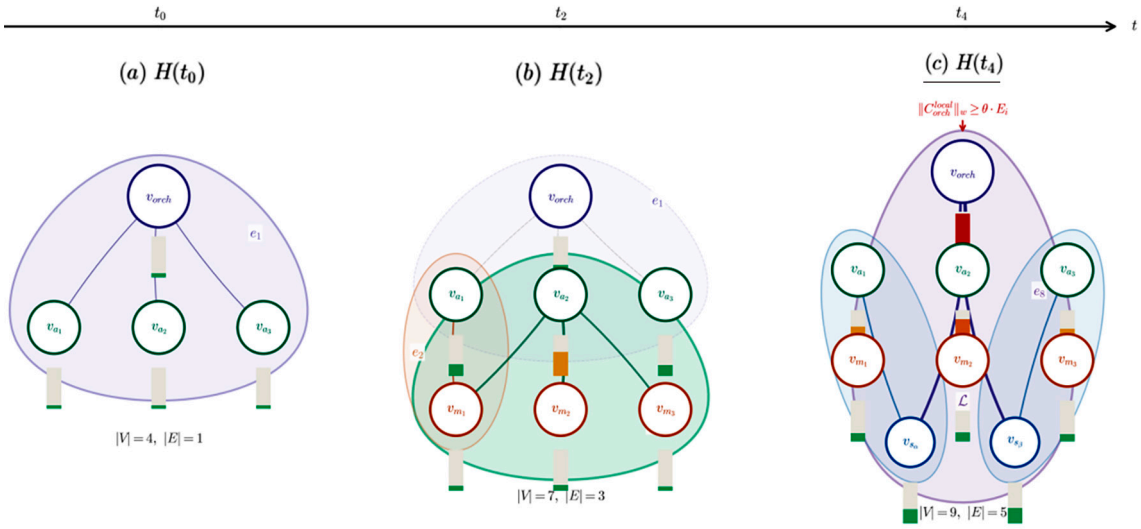

Рис. 4. Пример эволюции временного гиперграфа H(t)=(V(t), E(t), T) адаптивного самоорганизующегося распределённого конвейера
Источник: составлено автором

Переход к более сложным паттернам, например асинхронной событийной рассылке, многосторонней агрегации, агентному консенсусу, реализуется параметрически, через варьирование интервала активности и оператора агрегации, без введения новых классов рёбер. В отличие от DAG, гиперграф непосредственно кодирует отношение инцидентности множества источников приёмнику на уровне самого ребра. Динамическое же перестроение топологии вычислительного конвейера определяется как появление/исчезновение гиперрёбер и узлов в V(t) и E(t). На рисунке 4 показана данная динамика.

Гиперграф задаёт топологию и формальные пути распространения контекста, однако для анализа переполнения требуется также описать механизм аккумуляции и эволюции самого контекстного состояния. Для этого в рамках данной статьи вводится двухуровневая модель состояния каждого узла $v_i \in V(t)$ разделяющая долгосрочную историю вычислений и активное рабочее окружение. Полное сохраняемое состояние $C_i^{full}(t)$ аккумулирует архивированные чекпойнты, долгосрочную память и артефакты, вынесенные во внешние хранилища, например в объектном хранилище S3. Локальный контекст $C_i^{local}(t)$ представляет собой оперативную часть состояния, непосредственно загруженную в контекстное окно модели или буфер оператора. Именно динамика локального контекста является центральным объектом изучения, поскольку его рост вдоль гиперрёбер напрямую приводит к переполнению. Для количественного анализа используется взвешенный объём контекста, определяемый формулой

$$\left\|C_i^{local}(t)\right\|_w = \sum_{j=1}^{d} w_{ij} \cdot \left\|c_{i,j}^{local}(t)\right\|_1 . \quad (3)$$

Веса $w_{ij} \geq 0$ калибруются эмпирически с помощью профилировщика путём измерения зависимости задержки инференса (или буферизации) от объёма контекста каждого типа. Подобный подход полностью согласуется с методологиями оценки стоимости внимания в NLP-моделях, где квадратичная сложность механизма самовнимания (англ. self-attention) оправдывает повышенные веса для текстовых и эмбеддинг-компонентов [11]. Далее динамика контекста на v_i узле может быть описана уравнением эволюции контекстного состояния на гиперграфе

$$C_i^{local}(t_{k+1}) = C_i^{local}(t_k) + \sum_{e \in E_i(t_k)} A_e(t_k) A\left(\{C_j^{local}(t_k)\}_{j \in S_e}\right) - B_i(t_k) p_i(t_k) + \delta_i(t_k), \quad (4)$$

где $A(\cdot)$ – это агрегирующая функция гиперребра;

$A_e(t_k)$ – матрица трансформации контекста при передаче по гиперребру;

$B_i(t_k)p_i(t_k)$ – объём контекста, удаляемого в результате обработки (реализуется через через механизмы TTL (англ. Time to Live) или выноса в многоуровневые хранилища);

$\delta_i(t_k)$ – дополнительные накладные расходы протоколов (служебные заголовки JSON-RPC, метаданные трассировки, дескрипторы согласования и т. д.).

Агрегирующая функция гиперребра принимает множество локальных состояний всех источников (отправителей) в гиперребре e, $\{C_j^{local}(t_k)\}_{j\in S_e}$ и возвращает одно агрегированное значение, которое представляет результат взаимодействия внутри данного гиперребра в момент времени t. В случае отсутствия механизма контроля контекста, то есть когда $B_i(t) \approx 0$ уравнение (4) гарантирует монотонный рост нормы

$$\left\|C_i^{local}(t)\right\|_w,$$

особенно ярко проявляющийся при динамическом увеличении множества узлов V(t).

Учитывая, что формы контекстного переполнения в данной работе рассматриваются в едином контуре, фиксируется, что контекстное переполнение формализуется через две функции потерь, которые отражают деградацию производительности L_p и семантической согласованности L_s. Деградация производительности определяется через отклонение текущей глубины очереди от целевой пропускной способности узла, в соответствии с формулой

$$L_P^{stream}(t_k) = \alpha_P \cdot \max\left(0, \frac{Q_i(t_k)}{Q_i^{max}} - \theta_P\right)^{\gamma},$$

$$\gamma \geq 2, \qquad (5)$$

где $Q_i(t_k) \propto \left\|c_i^{data}(t_k)\right\|_l$ – текущая глубина очереди оператора;

Q_i^{max} – максимальная допустимая глубина очереди;

θ – пороговое значение устойчивости (backpressure) [12, с. 12–28].

Параметр степени полинома в функции деградации производительности выбран больше 2, что обеспечивает сверхлинейный рост штрафа при превышении порога, а также является стандартным подходом в методах штрафов и барьеров [13].

Деградация же семантической согласованности описывает эффект затерянности в середине (англ. lost-in-the-middle) [6], что может быть описано формулой

$$L_S(t_k) = 1 - \frac{\sum_{p\in R} u(p, N(t_k))}{\sum_{q=1}^{N(t_k)} u(q, N(t_k))},$$

где $u(p, N) = e^{-\lambda \cdot p} + e^{-\lambda \cdot (N-p)}$ – U-образный профиль эффективного внимания с пиками у границ контекста;

R – позиции релевантных артефактов в контексте;

$N(t_k) = \left\|c_i^{agent}(t_k)\right\|_l$ – текущий размер контекста агента;

$\lambda > 0$ – параметр экспоненциального спада внимания.

При $N(t_k) \to N_{max}$ распределение внимания становится практически равномерным и $L_s(t_k) \to 1$, иными словами, релевантная информация почти полностью теряется при значениях, близких к 1.

На основе введённых выше метрик формулируется строгое условие перехода системы в критическое состояние. Состояние конвейера считается переполненным на уровне узла $v_i \in V(t)$ в момент времени t_k, если выполняется неравенство

$$\left\|C_i(t)\right\|_w \geq \theta_i \cdot E_i(t),$$

где $E_i(t)$ представляет собой динамическую эффективную ёмкость обработки. Пороговый коэффициент θ_i рекомендуется брать больше 0.8, так как такой порог обеспечивает раннее обнаружение проблемы деградации семантической согласованности, когда система ещё работоспособна, но уже входит в режим плавной деградации, характеризуемый строго положительными производными функций потерь

$$\frac{dL_P}{dt} > 0 \text{ и } \frac{dL_S}{dt} > 0,$$

обеспечивая тем самым запас устойчивости узла до достижения его эффективной ёмкости $E_i(t)$.

В гетерогенных конвейерах гиперрёбра нередко формируют не изолированные потоки, а циклические зависимости, при которых результаты одного узла рекурсивно порождают контекстную нагрузку на другие узлы того же цикла. В гибридных и роевых архитектурах вычислительного конвейера такие циклы могут активироваться асинхронно и взаимно усиливать накопление контекста. Рассмотрим упрощённую модель взаимодействия между потоковым узлом и агентным узлом, связанными гиперребром. Пусть в гиперграфе существует замкнутый цикл

$$L = (x_1 \to x_2 \to \cdots \to x_m \to x_1),$$

где каждое ребро соответствует локальному приращению контекстного состояния узла. Если выполняется условие

$$\prod_{(x\to y)\in L} \frac{\partial y}{\partial x} > 1,$$

то за каждый полный обход цикла совокупный объём контекста возрастает, и система

переходит в режим лавинообразного переполнения. Таким образом, задача снижения контекстного переполнения сводится к построению такого механизма регулирования динамики временного гиперграфа H(t), который обеспечивает минимизацию взвешенной суммы потерь $w_p L_p + w_s L_s$. Решение этой задачи представлено в следующем разделе.

Метод адаптивной прогнозирующей суммаризации контекста на гиперрёбрах

Из формальной модели следуют два принципиально разных режима деградации конвейера. В первом сценарии $\left\|C_i^{local}(t)\right\|_w$ растёт монотонно вследствие, как отмечалось ранее, $B_i(t) \approx 0$. Во втором сценарии система входит в режим самоусиливающегося каскада, при котором задержки потокового слоя порождают рекурсивные вызовы агента, и в этом случае пороговое сравнение нормы срабатывает с запозданием. Предлагаемый метод покрывает оба сценария единым программным механизмом, встроенным в оператор передачи $A_e(t_k)$. Ключевым архитектурным решением предлагаемого подхода является размещение логики редукции контекста непосредственно на уровне гиперребра, то есть до записи данных в контекстное окно узла-приёмника. Такой подход соответствует принципу раннего связывания ресурсных ограничений, применяемому в планировщиках операционных систем и менеджерах памяти современных систем [14, с. 200–230, 354–380].

Предлагаемый метод структурно состоит из трёх программных модулей, взаимодействующих через разделяемую структуру данных, и представляет собой таблицу состояния гиперрёбер гиперграфа. Первый модуль представляет собой двухканальный детектор, основной задачей которого, реализуемой в рамках первого канала, является расчёт взвешенной нормы $\left\|C_i^{local}(t)\right\|_w$ и сравнения её с установленным порогом. Второй канал обходит список активных гиперрёбер из таблицы состояний гиперрёбер и инкрементально обновляет произведение

$$\Gamma_i(t_k) = \prod_{(x \to y) \in L} \left| \frac{\partial y}{\partial x} \right|$$

и формирует сигнал бинарного флага

$$\delta_i(t_k) = 1\left[\left\|C_i\right\|_w \geq \theta_i \cdot E_i(t_k)\right] \vee 1\left[\Gamma_i(t_k) > 1\right].$$

В случае если значение флага становится равным 1, это приводит к активации компонента, представляющего собой классификатор данных.

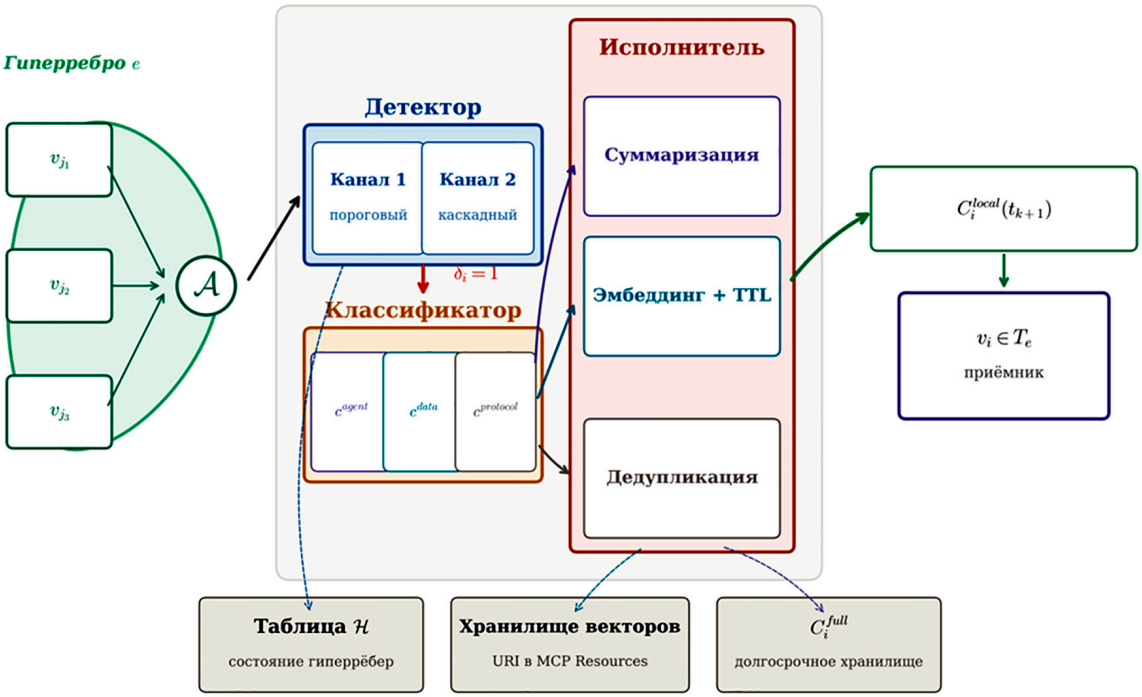

Рис. 5. Структурная схема двухканального детектора и тип-зависимого исполнителя редукции контекста на гиперребре
Источник: составлено автором

Данный компонент выполняет однократный обход локального контекста и разбивает его на три непересекающихся компонента по тегам типа «агентный контекст» (c_i^{agent}), «потоковые данные» (c_i^{date}) и «накладные расходы протокола» ($c_i^{protocol}$). При этом реализация классификатора не требует реализации сложного семантического анализа, так как теги типа проставляются детерминированно на основе источника записи в контекстное окно.

Далее модуль исполнения применяет соответствующую стратегию к каждому определённому ранее компоненту, вызывая соответствующую подпрограмму. Например, для типа c_i^{agent} вызывается подпрограмма суммаризации, которая генерирует краткое семантическое резюме цепочки рассуждений. Сам же оригинальный контекст сериализуется и помещается в полное сохраняемое состояние $C_i^{full}(t)$ для возможного последующего восстановления при необходимости. После исполнения всех трёх подпрограмм локальный контекст узла-приёмника обновляется, а в таблицу состояния гиперрёбер записываются новые значения нормы и коэффициентов для следующего такта. На рисунке 5 показана архитектура данного промежуточного слоя.

Реализация разработанного метода и его анализ

Описанный выше метод реализуется в виде самостоятельного программного компонента, представляющего собой промежуточный программный слой. Такая архитектурная реализация упрощает процесс встраивания в инфраструктуру мультиагентного конвейера без модификации протоколов взаимодействия агентов, что может быть полезно для уже существующих реализаций. На практике данный слой разворачивается как лёгкий вспомогательный процесс, размещаемый рядом с каждым узлом вычислительного конвейера (паттерн sidecar [15]). Он перехватывает передаваемые данные до их поступления в узел-приёмник и выполняет необходимую обработку, связанную с редукцией контекста. Таблица состояния гиперрёбер, через которую взаимодействуют три описанных выше модуля, реализуется в виде разделяемой структуры данных типа key-value хранилища с поддержкой атомарных операций чтения и обновления, что позволяет избежать повторных вычислений. В качестве идентификатора записи используется хеш, объединяющий множества источников и приёмников гиперребра, а также идентификатор временного окна.

Алгоритм 1 Адаптивная суммаризация контекста на гиперрёбрах

Вход: $H(t_k) = (V(t_k), E(t_k), T)$, таблица состояний $\mathcal{H}$, пороги $\{\theta_i\}$, веса $\{w_{ij}\}$
Выход: $\{C_i^{local}(t_{k+1})\}$, обновлённая таблица $\mathcal{H}$

for all $v_i \in V(t_k)$ параллельно **do**
  $norm_i \leftarrow \sum_j w_{ij} \, \|c_{i,j}^{local}(t_k)\|_1$
  $\Gamma_i \leftarrow \prod_{(x \to y) \in L_i,\ \mathcal{H}[e].changed = true} \left| \frac{\partial y}{\partial x} \right|$
  if $norm_i < \theta_i E_i(t_k)$ **and** $\Gamma_i \leq 1$ **then**
    continue
  end if
  $(C_{agent}, C_{data}, C_{proto}) \leftarrow split(C_i^{local}(t_k))$
  $C_{agent}^{new} \leftarrow Summarize_{LLM}(C_{agent})$
  $C_{data}^{new} \leftarrow CompressData(C_{data})$
  $C_{proto}^{new} \leftarrow Deduplicate(C_{proto}, D(t_k))$
  $C_i^{local}(t_{k+1}) \leftarrow C_{agent}^{new} \cup C_{data}^{new} \cup C_{proto}^{new}$
  UpdateState$(\mathcal{H}, v_i, C_i^{local}(t_{k+1}), \Gamma_i, t_{k+1})$
end for

Инвариант: $\|C_i^{local}(t_{k+1})\|_w < \theta_i \cdot E_i(t_{k+1})$ для всех $v_i \in V(t_k)$

Рис. 6. Псевдокод алгоритма адаптивной суммаризации контекста на гиперграфах
Источник: составлено автором

Реализация компонента двухканального детектора осуществлена в виде отдельного рабочего потока, выполняющего периодический опрос таблицы состояний с настраиваемым интервалом дискретизации. Классификатор данных реализуется как детерминированный компонент, не использующий никаких моделей машинного обучения, что принципиально отличает его от типовых решений на основе семантической классификации. Обработка контекста осуществляется дифференцированно в зависимости от его типа. Например, агентный контекст подвергается суммаризации с помощью облегчённой языковой модели с последующей сериализацией исходных данных во внешнее хранилище, а потоковые данные уже обрабатываются по принципу скользящего окна. В свою очередь, активация редукции определяется двухканальным детектором, который реализует как реактивный, так и прогнозирующий подход.

Алгоритм работы одного такта системы представлен ниже в виде псевдокода, что показано на рисунке 6.

Для формального обоснования метода покажем, что предложенный программный компонент гарантирует инвариант ограниченности локального контекста и сохраняет вычислительную сложность на уровне, не зависящем от числа потенциальных циклов в гиперграфе. Для этого можно воспользоваться теорией липшицевых отображений, применённой не к непрерывным динамическим системам [16, с. 25–40], а, в рамках данной работы, к дискретным операторам над контекстом.

Допущение A1 (ограниченность приращения контекста за такт). Существует константа Δ_{max}, такая, что для любого t_k выполняется

$$\left\| \Sigma_e \; A_e(t_k)\, A(C_j^{local}(t_k)_{(j\in S_e)}) + \delta_i(t_k) \right\|_w \le \; \Delta_{max}. \qquad (7)$$

Данное допущение связано с тем, что на практике число одновременно активных гиперрёбер, нормы матриц трансформации ограничено топологией конвейера (программной составляющей), а также используемыми форматами сообщений (например, размер заголовков JSON-RPC ограничен сверху по конструкции, а практические лимиты на объём передаваемых данных определяются конфигурацией сервера, в том числе значениями по умолчанию для распространённых реализаций [17, с. 35–50]).

Допущение A2 (нижняя оценка ёмкостной характеристики). Существует константа $E_{min} > 0$ такая, что $E_i(t) \ge E_i^{min}$ для всех $t \ge t_0$ и всех $v_i \in V(t)$. Это допущение соответствует физическим ограничениям инфраструктуры развёртывания вычислительного конвейера, таким как, например, размер контекстного окна модели узла конвейера, без привязки к конкретной реализации.

Утверждение 1 (свойства оператора редукции). Реализуемый программный модуль определяет оператор редукции R, состоящий из трёх независимых компонентов, обрабатывающих агентный контекст, потоковые данные и накладные расходы от используемого протокола интеграции. Оператор R обладает следующими свойствами:

a) ограниченность нормы контекста, когда для любого входного контекста C локальный контекст на каждом узле/агенте никогда не превышает определённый безопасный порог, то есть выполняется

$$\left\| R(C) \right\|_w \le \beta_{eff}, \;\; \beta_{eff} = L_{sum} \cdot w_{i_{agent}} + W_{size} \cdot w_{i_{data}} + TT \cdot v_{max} \cdot w_{i_{protocol}}. \qquad (8)$$

b) нерасширяющее отображение, что означает, что для любых двух локальных пар контекстов C_1 и C_2, попадающих в область активации детектора, выполняется

$$\left\| R(C_1) - R(C_2) \right\|_w \le L_R \cdot \left\| C_1 - C_2 \right\|_w,$$

$$L_R \in [0,1).$$

Доказательство. Свойство (а) следует из детерминированных ограничений классификатора, который разделяет контекст на три типа компонентов. Компонент (как подпрограмма) суммаризации ограничивает агентный контекст, скользящее окно ограничивает потоковые данные, а механизм TTL удаляет устаревшие метаданные. Свойство (б) обеспечивается уже конструкцией подпрограмм, так как компоненты обработки потоковых данных и протокольных расходов являются линейными, с константой Липшица, не превышающей 1, а компонент суммаризации проектируется с ограничением вариации выхода, что гарантирует, что константа Липшица также не превышает 1. Общая же константа Липшица всего реализуемого модуля определяется как максимум констант отдельных подпрограмм, что с учётом вышеизложенного гарантирует сходимость и устойчивость процесса суммаризации/редукции.

Утверждение 2 (ограниченность локального контекста). Допущения A1 и A2 совместно определяют границы задачи регуляции контекста. Программный компонент редукции функционирует в пределах допустимого рабочего диапазона, не допуская, чтобы накопленный объём контекста приближался

к пороговому значению быстрее, чем оператор R успевает его уменьшить. Пусть двухканальный детектор активирует редукцию с периодом Δt, удовлетворяющим условию

$$\Delta t \le \frac{\theta_i E_i^{min} - \beta_{eff}}{\Delta_{max}}, \theta_i E_i^{min} > \beta_{eff}, \quad (9)$$

где Δ_{max} – максимальное приращение нормы контекста за один такт;

E_i^{min} – нижняя оценка эффективной ёмкости.

Тогда после первой активации детектора для любого $t_k \ge t_0$ выполняется

$$\left\| C_i^{local}(t_k) \right\|_w \le \beta_{eff} + \Delta_{max} \cdot \Delta t. \quad (10)$$

При срабатывании детектора оператор R гарантирует $\|C\|_w \le \beta_{eff}$. Рассмотрим последовательность моментов активации t_k. После применения оператора R в момент t_k норма не превосходит β_{eff}. По индукции норма остаётся ограниченной указанной константой независимо от числа тактов. Это напрямую предотвращает переполнение контекстного окна и даёт гарантии того, что локальный контекст на каждом узле/агенте никогда не превышает определённый безопасный порог. Иными словами, при выборе параметров, удовлетворяющих $\beta_{eff} + \Delta_{max}\Delta t < \theta_i E_i^{min}$, норма локального контекста никогда не достигает порога переполнения, что делает переход узла в критическое состояние алгоритмически невозможным.

Заключение

Проведённое исследование позволило разработать метод снижения контекстного переполнения в мультиагентных самоорганизующихся конвейерах на основе адаптивной суммаризации контекста, реализуемый в виде промежуточного слоя на уровне гиперрёбер временного гиперграфа. Предложенная архитектура обеспечивает отделение слоя регулирования контекста от прикладной логики агентов, заменяя реактивную обработку переполнения ранним связыванием ресурсных ограничений. Показано, что использование такого слоя позволяет перейти от реактивной обработки переполнения к ранней регуляции ресурсных ограничений, при которой стратегия редукции выбирается с учётом типа передаваемого контекста и текущего состояния гиперребра. Решение ориентировано, прежде всего, на гетерогенные распределённые среды и применимо для повышения устойчивости интеллектуальных конвейеров без модификации протоколов интеграции, всё чаще применяемых в таких системах.

Список литературы

1. Hou X., Zhao Y., Wang S., Wang H. Model Context Protocol (MCP): Landscape, Security Threats, and Future Research Directions // ACM Transactions on Software Engineering and Methodology. 2026. DOI: 10.1145/3796519.

2. What is the Model Context Protocol (MCP)? URL: https://modelcontextprotocol.io/docs/getting-started/intro (дата обращения: 11.01.2026).

3. Liao C. C., Liao D., Gadiraju S. S. AgentMaster: A Multi-Agent Conversational Framework Using A2A and MCP Protocols for Multimodal Information Retrieval and Analysis // Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: System Demonstrations. 2025. P. 52–72. DOI: 10.18653/v1/2025.emnlp-demos.5.

4. Agent2Agent (A2A) Protocol Specification. URL: https://a2a-protocol.org/latest/ (дата обращения: 20.01.2026).

5. LangGraph. LangChain. URL: https://www.langchain.com/langgraph (дата обращения: 20.01.2026).

6. Liu N. F., Lin K., Hewitt J., Paranjape A., Bevilacqua M., Petroni F., Liang P. Lost in the Middle: How Language Models Use Long Contexts // Transactions of the Association for Computational Linguistics. 2024. Vol. 12. P. 157–173. DOI: 10.1162/tacl_a_00638.

7. Alidu A., Ciavotta M., De Paoli F. Prompt2DAG: A Modular Prompting Approach for Democratizing Data Pipeline Generation // 2025 IEEE International Conference on Software Services Engineering (SSE). 2025. P. 1–11. DOI:10.1109/SSE67621.2025.00010.

8. Matteussi K. J., dos Anjos J. C. S., Leithardt V. R. Q., Geyer C. F. R. Performance Evaluation Analysis of Spark Streaming Backpressure for Data-Intensive Pipelines // Sensors. 2022. Vol. 22. № 13. Article 4756. DOI: 10.3390/s22134756.

9. Nguyen T. C., Nguyen V. S., Nguyen N. H., Van D. C., Nguyen M. H., Nguyen T. D., Nguyen H. T. AutoMind: Automated Insight Discovery via Multi-Agent Navigation // IEEE Access. 2026. Vol. 14. P. 23936–23955. DOI: 10.1109/ACCESS.2026.3661202.

10. Joseph A. A., Nambiar G. S., Jayapandian N. Swarm Intelligence Decentralized Decision Making in Multi-Agent System // 2023 8th International Conference on Communication and Electronics Systems (ICCES). 2023. P. 1425–1430. DOI: 10.1109/ICCES57224.2023.10192625.

11. Vaswani A., Shazeer N., Parmar N., Uszkoreit J., Jones L., Gomez A. N., Kaiser Ł., Polosukhin I. Attention Is All You Need // Advances in Neural Information Processing Systems. 2017. Vol. 30. P. 5998–6008.

12. Davis A. L. Reactive Streams in Java: Concurrency with RxJava, Reactor, and Akka Streams. Berkeley, CA: Apress, 2019. 153 p. DOI: 10.1007/978-1-4842-4176-9. ISBN: 978-1-4842-4175-2.

13. Chen X., Xin L., Zhao M. Hidden Convexity in Queueing Models // SSRN Electronic Journal. 2025. DOI: 10.2139/ssrn.5709506.

14. Silberschatz A., Galvin P. B., Gagne G. Operating System Concepts. 10th ed. Hoboken, NJ: John Wiley & Sons, 2019. 896 p. ISBN: 978-1-119-45408-3.

15. Meadows C., Hounsinou S., Wood T., Bloom G. Sidecar-based path-aware security for microservices // Proceedings of the 28th ACM Symposium on Access Control Models and Technologies. 2023. P. 157–162. DOI: 10.1145/3589608.3594742.

16. Арнольд В. И. Обыкновенные дифференциальные уравнения. 3-е изд., стер. М.: МЦНМО. 2024. 344 с. ISBN: 978-5-4439-4548-4.

17. Sharma R. NGINX High Performance. Birmingham: Packt Publishing, 2015. 168 p. ISBN: 978-1-78528-183-9.

Конфликт интересов: Авторы заявляют об отсутствии конфликта интересов.

Conflict of interest: The authors declare that there is no conflict of interest.

Финансирование: Авторы заявляют об отсутствии внешнего финансирования.

Financing: The research was performed without external funding.