УДК 004.75

Системный анализ и формализация задачи координации распределённых систем в условиях потокового анализа данных: общие понятия и проблемы

А.Г. Краснобородкин[1]✉, Е.С. Петрова[2]

[1]Воронежский институт высоких технологий, Воронеж, Россия
[2]Воронежский государственный технический университет, Воронеж, Россия

В статье рассматриваются проблемы координации и согласованности распределённых систем потоковой обработки данных. Проведён анализ существующих архитектур и алгоритмов консенсуса, а также формализована модель распределённой системы как динамического графа. Определено понятие асимптотического консенсуса и описаны критерии его достижения в условиях неопределённости, задержек и отказов узлов. Представлены направления дальнейших исследований по созданию адаптивных алгоритмов координации для потоковых вычислительных систем.

Ключевые слова: распределённые системы, потоковый анализ данных, консенсус, координация, динамические графы, асимптотический консенсус.

System Analysis and Formalization of the Task of Coordinating Distributed Systems in the Context of Streaming Data Analysis: General Concepts and Problems

A.G. Krasnoborodkin[1]✉, E.S. Petrova[2]

[1]Voronezh Institute of High Technologies, Voronezh, Russia
[2]Voronezh State Technical University, Voronezh, Russia

The article discusses the problems of coordination and consistency of distributed streaming data processing systems. The analysis of existing consensus architectures and algorithms is carried out, and the model of a distributed system as a dynamic graph is formalized. The concept of an asymptotic consensus is defined and the criteria for achieving it under conditions of uncertainty, delays and node failures are described. The directions of further research on the creation of adaptive coordination algorithms for streaming computing systems are presented.

Keywords: distributed systems, streaming data analysis, consensus, coordination, dynamic graphs, asymptotic consensus.

Введение

Современные системы потокового анализа данных используются для обработки больших объёмов информации в реальном времени, характерных для IoT, киберфизических систем и телекоммуникационных сетей. Одной из ключевых проблем является обеспечение координации вычислительных узлов в условиях асинхронности, ограниченной пропускной способности каналов и отказов. Традиционные централизованные подходы не обеспечивают требуемую масштабируемость и надёжность, поэтому возникает необходимость разработки распределённых алгоритмов, основанных на принципах консенсуса.

Классификация распределённых систем в контексте потокового анализа данных

Для эффективной обработки данных, генерируемых информационными системами, требуется не просто хранение, но и непрерывный анализ с минимальными задержками. В этом вопросе на передний план выходят распределённые системы потоковой обработки данных (stream processing systems) – программные платформы, способные обрабатывать непрерывные потоки событий на кластере из множества узлов.

Однако не все такие системы устроены одинаково. Их архитектура, модель обработки, уровень согласованности и отказоустойчивости могут сильно различаться. Поэтому актуальной становится классификация распределённых систем потокового анализа данных по ключевым признакам, что позволяет выбирать наиболее подходящее решение в зависимости от задачи – будь то мониторинг SIEM, анализ поведения пользователей или управление IoT-устройствами или распознавание речи [1].

Существует множество способов классификации распределённых систем в контексте потокового анализа данных: по архитектуре (централизованные и децентрализованные), по модели обработки (микропакетные и непрерывные), по уровню согласованности (строгая, eventual, causal) и другие [2].

По архитектуре: централизованные системы используют единый координирующий компонент (контроллер, мастер-узел), который управляет распределением задач, отслеживает состояние исполнителей и обеспечивает согласованность. Это, например, Apache Kafka Streams (с контроллером брокера) или Apache Flink (JobManager). Основным недостатком любых централизованных систем является наличие единой точки отказа.

Децентрализованные системы строятся без выделенного лидера. Все узлы равноправны, а согласование достигается с помощью алгоритмов консенсуса.

По модели обработки: микропакетная (micro-batching) модель разбивает поток на пакеты, которые обрабатываются последовательно (например, Apache Spark Streaming). Данные собираются за короткие интервалы (например, 100 мс), затем пакет проходит через конвейер трансформаций. В таких системах достаточно просто управлять состоянием, однако высокая степень задержки не позволяет использовать микропакетные модели.

Непрерывная модель обрабатывает каждое событие сразу после его поступления. Так работают Apache Flink, Apache Storm, ksqlDB. Каждое сообщение проходит через операторы без ожидания других. Несмотря на сложность управления, именно непрерывные модели идеально подходят для обработки в реальном времени.

В контексте потокового анализа, где важно быстро обнаружить аномалию (например, DDoS-атаку), предпочтение отдаётся непрерывной модели.

По уровню согласованности: распределённые системы сталкиваются с дилеммой CAP (Consistency, Availability, Partition tolerance): в распределённой системе невозможно гарантировать все три свойства одновременно, реализуемы лишь два. В потоковом анализе особенно важен выбор уровня согласованности (consistency model).

Строгая согласованность гарантирует, что все узлы видят одни и те же данные в каждый момент времени. Достигается с помощью механизмов блокировок и консенсуса (Paxos/Raft). Используется в системах, где критична точность, например, при блокировке IP-адреса в SIEM.

Eventual consistency допускает временное несовпадение данных, но гарантирует, что со временем все реплики сойдутся. Характерна для систем с высокой нагрузкой, где приоритет – доступность. Подходит для аналитики, где допустимы небольшие расхождения.

Causal consistency сохраняет причинно-следственные связи между событиями: если событие A повлияло на B, то все узлы увидят их в правильном порядке. Это золотая середина для многих систем потоковой обработки.

Выбор модели влияет на производительность и корректность выводов. Например, в системе обнаружения вторжений потеря порядка событий может привести к ложным срабатываниям.

Проблемы координации и согласованности в распределённых системах

Эффективность распределённых систем напрямую зависит от способности обеспечивать согласованность состояния между узлами. Однако в реальных условиях существует множество проблем, не допускающих существование идеальных моделей: коммуникационные задержки, отказы узлов, византийские сбои и явление «расщепления мозга» (split-brain) [3].

Коммуникационные задержки

В реальных сетях нет гарантии немедленной доставки сообщений. Задержки возникают из-за очередей в маршрутизаторах, перегрузки канала, географического удаления узлов, колебаний качества связи и других препятствий, что приводит к асинхронности и сбоям в сети.

Для решения такого рода проблемы можно использовать принудительные тайм-ауты, позволяющие нивелировать влияние задержек на принятие решений. Однако в асинхронных сетях верхняя граница задержки отсутствует, что делает тайм-ауты бесполезными.

Отказы узлов

Узлы могут выходить из строя по разным причинам: сбой ПО, перезагрузка, падение питания. Такие отказы называются crash failures – узел перестаёт отвечать, но до этого вёл себя корректно. Это влечёт за собой серьёзные последствия, вплоть до потери данных.

Главная проблема состоит в том, что обеспечить отказоустойчивость всех узлов системы на практике невозможно, и в то же время существует теорема FLP, которая утверждает, что в асинхронной распределённой системе нельзя гарантировать наступление консенсуса, если существует возможность выхода из строя хотя бы одного узла.

Византийский сбой

Наиболее сложный тип сбоя – византийский отказ, когда узел ведёт себя произвольно: отправляет противоречивые сообщения, имитирует работу, передаёт ложные данные. Причина – ошибки ПО, аппаратные дефекты или злонамеренные действия.

Для противодействия f византийским узлам (то есть для достижения консенсуса) в системе должно быть минимум 3f+1 узлов.

На практике такая проблема встречается редко, однако в случае её возникновения существует шанс сбоя процессов без выхода системы из строя, что и делает византийский отказ опасным и трудно вычисляемым.

Split-Brain

Split-brain – ситуация, при которой сеть делится на две или более изолированные части, и каждая часть считает себя «истинной» копией системы. В этом случае как минимум два узла могут считать себя главными, что может привести к двойной записи, потери данных или дублированию операций.

Для решения этой проблемы используется, например, кворумное управление, то есть система работает только при наличии большинства узлов.

Распределённая система как динамический граф

Распределённую систему можно описать как ориентированный взвешенный граф:

$$G(t) = (V(t), E(t), W(t)), \quad (1)$$

где $V(t) = \{v_1, v_2, ..., v_N\}$ – множество узлов, $E(t)$ – множество связей, а $W(t) = [w_{ij}(t)]$ – матрица весов, отражающая интенсивность обмена данными.

Каждый узел имеет локальное состояние $x_i(t)$, изменяющееся по правилу:

$$x_i(t+1) = w_{ij}(t) \cdot x_i(t) + \sum_{j \in N_i(t)} w_{ij}(t) \cdot x_j(t), \quad (2)$$

где $N_i(t)$ – множество соседних узлов, с которыми узел v_i обменивается данными в момент времени t.

Состояние консенсуса достигается, если значения всех узлов асимптотически сходятся к одному и тому же значению:

$$\lim_{t \to \infty} |x_i(t) - x_j(t)| = 0, \forall i, j \in V(t). \quad (3)$$

Если при этом существует предел

$$x^* = \lim_{t \to \infty} x_i(t), \quad (4)$$

то говорят, что система достигла асимптотического консенсуса.

Задача координации как задача оптимизации

В распределенных системах, где различные узлы одновременно обрабатывают поток данных, каждый из них принимает собственные локальные решения. *Координация* – это процесс установления единого согласованного состояния системы на каждом отдельном узле.

Каждый узел i хочет минимизировать свою локальную функцию ошибки $f_i(x_i)$ По сути проблема координации может быть представлена как задача минимизации суммарного функционала:

$$\min_{x_1, x_2, \ldots, x_n} \sum_{i=1}^{n} f_i(x_i), \; x_j = x_j, \forall i, j. \quad (5)$$

Таким образом, система стремится к глобальному оптимальному состоянию, сохраняя согласованность локальных решений. В условиях потоковой обработки такая задача решается итерационно, с учётом поступления новых данных и динамики топологии.

Стоит отметить, что *асимптотический консенсус* – это ослабленная форма консенсуса, при которой узлы постепенно сближаются к одному и тому же значению. Полного совпадения может не быть в каждый момент времени (в отличие от строгого консенсуса), но с течением времени разница между значениями на узлах асимптотически стремится к нулю (3).

Для того чтобы система достигала асимптотического консенсуса, должны выполняться следующие ключевые свойства:

1. *Сходимость.*

Узлы постепенно сближают свои значения. Это достигается за счёт итеративного обмена информацией и коррекции собственного состояния на основе среднего (или взвешенного среднего) значений соседей [4].

2. *Согласованность.*

В пределе все узлы приходят к одному и тому же значению:

$$\lim_{t\to\infty} x_i(t) = \lim_{t\to\infty} x_j(t),\ \forall i, j. \tag{6}$$

Это требует, чтобы граф связности был связным и достаточно регулярным (например, сильно связный в каждый момент или по времени).

3. *Устойчивость к начальным условиям.*

Итоговое значение зависит только от начальных значений узлов и их весов, а не от порядка сообщений или задержек.

Общая схема работы сети относительно стремления достигнуть консенсус представлена на рисунке.

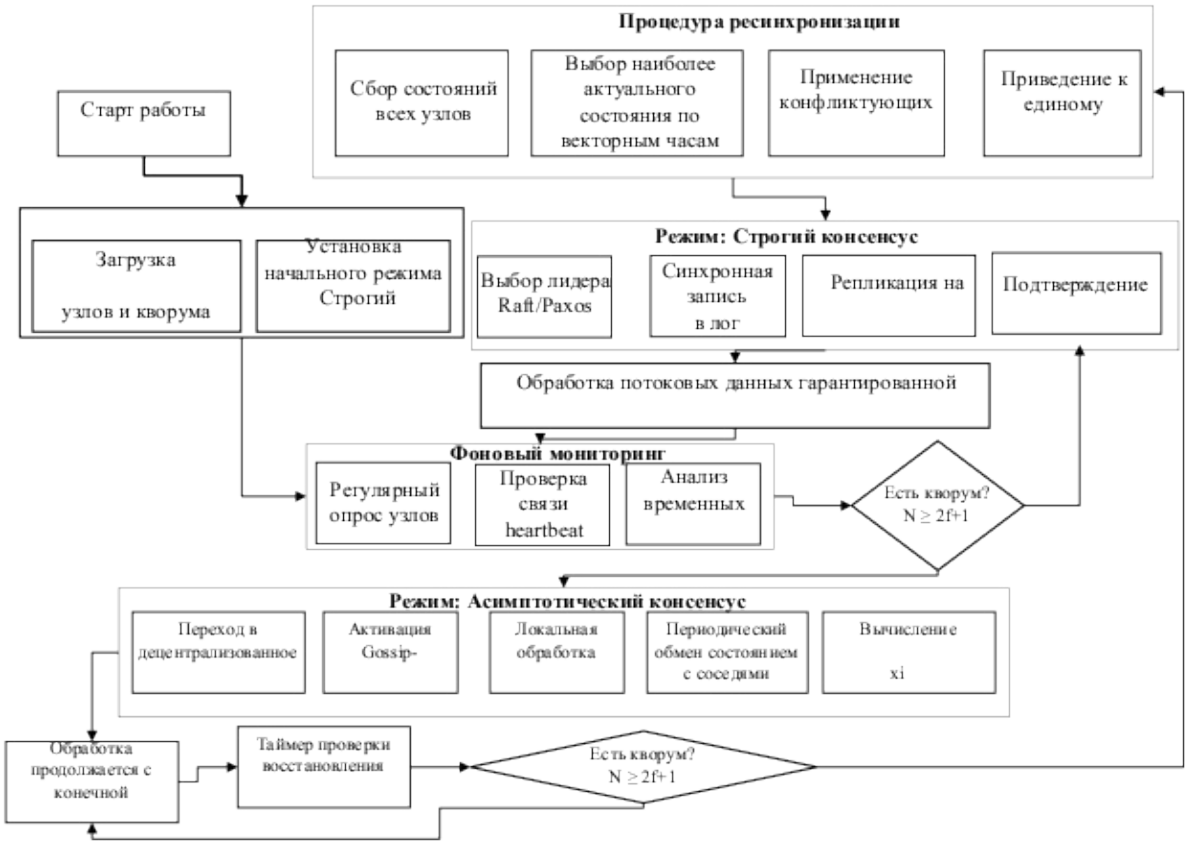

Рисунок. Схема достижения консенсуса

На схеме представлены следующие основные блоки

1. Инициализация. Система начинает работу с загрузки конфигурации и установки начального режима строгого консенсуса.

2. Фоновый мониторинг. Постоянно проверяет доступность узлов через: регулярный опрос; Heartbeat-сообщения; анализ временных меток.

3. Условие переключения. Проверка наличия кворума ($N \geq 2f + 1$ для отказоустойчивости).

4. Режим строгого консенсуса. Выбор лидера по Raft/Paxos; синхронная запись в лог; репликация на большинство узлов; подтверждение коммита.

5. Режим асимптотического консенсуса: децентрализованная работа; Gossip-протокол для обмена состояниями; локальная обработка с периодическим обмен; математическая модель усреднения состояний.

6. Процедура ресинхронизации. Сбор состояний после восстановления связи; разрешение конфликтов через векторные часы; использование CRDT (Conflict-free Replicated Data Types). Приведение системы к единому состоянию.

Такая детализированная схема показывает не только логику переключения, но и конкретные механизмы работы в каждом режиме, а также важный процесс восстановления согласованности после сбоя.

Анализ существующих алгоритмов консенсуса

Для любой распределённой системы критически важным является обеспечение консенсуса, и главной проблемой здесь является несовершенство узлов и связей между ними. Частые отказы, ошибки и задержки сети вынуждают применять различные алгоритмы, которые позволяют в таких жестких, но вполне естественных условиях, функционировать системе как единое целое. Классическими алгоритмами для достижения консенсуса принято считать Paxos, предложенный и описанный Лесли Лампортом ещё в 1998 году, и альтернативный ему алгоритм Raft, который, при обеспечении тех же свойств, более прост в понимании.

При использовании Paxos консенсус достигается путём создания условий, гарантирующих принятие единого решения всеми узлами сети [5]. Его суть заключается в распределении трёх ролей среди всех узлов: proposers (предлагающие), acceptors (принимающие) и learners (обучающиеся). Предлагающие распространяют ту или иную информацию, принимающие одобряют или не одобряют её, а обучающиеся сохраняют выбранное значение. Главным принципом работы Paxos является функционирование большинства узлов. В 2007 году был предложен Multi-Paxos, отличающийся от базового меньшей задержкой и большей производительностью за счёт выбора единого лидера для обработки всех консенсусов.

Алгоритм Raft характеризуется более явным выбором лидера, за счёт чего реализация становится в разы проще. Также меняются роли: существует выбранный лидер, управляющий всеми операциями, последователи, выполняющие команды лидера, и кандидаты, появляющиеся в случае потери связи с лидером.

Все эти алгоритмы обеспечивают строгую согласованность, но характеризуются высокой задержкой и плохой масштабируемостью. Gossip-протоколы обеспечивают вероятностный консенсус, что делает их более подходящими для потоковых систем. Они обеспечивают высокую доступность данных за счет децентрализованной архитектуры, при которой узлы обмениваются информацией с соседями, а не с центральной точкой. Это позволяет системе оставаться работоспособной даже при сбоях узлов или сетевых разделах. Кроме того, они ограничивают количество передаваемых сообщений и пропускную способность, что предотвращает перегрузку сети и снижает потребление ресурсов.

Дальнейшие исследования должны быть направлены на развитие гибридных подходов, где при нормальной работе используется строгий консенсус, а при потере связности система автоматически переключается в режим асимптотического консенсуса, позволяя узлам постепенно сближать состояния без блокировки обработки. Такой адаптивный подход сохраняет доступность и позволяет продолжать анализ данных даже в условиях частичных отказов. После восстановления сети система

должна корректно синхронизироваться, обеспечивая сходимость к единому состоянию без потерь. Таким образом, комбинирование двух моделей согласованности делает систему более гибкой и устойчивой к реальным условиям эксплуатации.

Заключение

В работе выполнен системный анализ проблем координации распределённых систем в условиях потокового анализа данных. Предложена формальная модель в виде динамического графа и определено состояние асимптотического консенсуса. Показано, что существующие алгоритмы требуют модификации для работы в асинхронных потоковых средах. Дальнейшие исследования направлены на разработку адаптивных и интеллектуальных механизмов координации.

СПИСОК ИСТОЧНИКОВ

1. Жаксылык К. Распределенная система потоковой обработки данных для задач распознавания речи / К. Жаксылык, В.А. Захарьев // BIG DATA и анализ высокого уровня: Сборник научных статей X Международной научно-практической конференции. – Минск: Белорусский государственный университет информатики и радиоэлектроники, 2024. – С. 358–370.

2. Таненбаум Э. Распределённые системы: принципы и парадигмы / Э. Таненбаум, М. ван Стеен; пер. с англ. В. Горбунков. – Москва: Питер, 2003. – 876 с.

3. Асимптотический консенсус в модели коллективного принятия решений на конкурентном рынке / Д.Г. Алгазина, Ю.Г. Алгазина, Н.Н. Шаховалов [и др.] // Известия Алтайского государственного университета. – 2025. – № 1 (141). – С. 75–80.

4. Ongaro D. In Search of an Understandable Consensus Algorithm / D. Ongaro, J. Ousterhout // Proceedings of USENIX ATC ’14: 2014 USENIX Annual Technical Conference. – USENIX Association, 2014. – P. 305–319.

5. Замечания о распределенных системах для начинающих // Хабр [Электронный ресурс]. – URL: https://habr.com/ru/companies/piter/articles/262807/ (дата обращения: 22.11.2025).

ИНФОРМАЦИЯ ОБ АВТОРАХ

Краснобородкин Александр Геннадьевич, аспирант, Воронежский институт высоких технологий, Воронеж, Россия.
e-mail: sashabayker36@mail.ru

Петрова Елена Сергеевна, старший преподаватель, Воронежский государственный технический университет, Воронеж, Россия.
e-mail: alexanderostapenkoias@gmail.com