Математическое моделирование

Моделирование распределенных СУБД с помощью теории массового обслуживания

А.В. Соловьев[I,II], Д.С. Кожевникова[II]

[I] Федеральный исследовательский центр «Информатика и управление» Российской академии наук, г. Москва, Россия
[II] Образовательное частное учреждение высшего образования «Православный Свято-Тихоновский гуманитарный университет», г. Москва, Россия

Аннотация. В статье выполнен анализ возможности моделирования современных распределенных СУБД с помощью многосерверных моделей теории массового обслуживания. Показана актуальность решаемой задачи. Обсуждена возможность достижения строгой согласованности при одновременной распределенности и доступности данных. Рассмотрен вопрос возможности применения математического аппарата теории массового обслуживания для моделирования распределенных СУБД, показаны модели очередей для многосерверных систем массового обслуживания. Разработана программная реализация для моделирования распределенных СУБД с помощью моделей теории массового обслуживания. Кратко представлены направления дальнейших исследований.

Ключевые слова: *системы управления базами данных, NewSQL СУБД, теория массового обслуживания, многосерверные модели, моделирование очередей, M/M/c.*
DOI: 10.14357/20790279260206 **EDN:** MFWSFO

Введение

Стремительная цифровизация экономики и общества приводит к все более и более жестким требованиям к своевременности предоставления информации, доступности информации, в том числе и в территориально-распределенных информационных системах. Требования высокой степени доступности, в свою очередь, приводят к спросу на самую актуальную информацию для принятия решений в условиях быстро меняющихся данных в информационных системах, обеспечивающих конкретные решения по цифровизации экономики и общества в целом.

Ответом на столь жесткие требования к информации стал класс NewSQL СУБД, начавший активно формироваться примерно 10 лет назад, и в настоящее время активно развивающийся.

Основные заявленные отличительные черты NewSQL СУБД кратко можно представить так:
- распределенное хранение данных на тысячах узлов системы хранения;
- поддержка реляционных схем хранения данных и языка SQL;
- высокая масштабируемость и надежность хранения;
- строгая согласованность выполняемых транзакций в объеме требований к традиционным реляционным СУБД: atomicity, consistency, isolation, durability (ACID), обеспечивающих при распределенном хранении доступность самой актуальной информации на всех узлах распределенной БД.

Краткий обзор современных решений NewSQL СУБД см., например, [1, 2].

Последняя отличительная черта особенно важна для темы данного исследования, на ней следует остановиться отдельно в контексте моделирования подобных систем.

Дело в том, что класс NewSQL СУБД очевидно востребован и получает широкое распространение, с одной стороны (см. так же краткие обзоры автора в [1, 2]). Существует определенное количество опубликованных тестов, практически подтверждающих строгую согласованность при одновременной распределенности и доступности данных. С другой стороны, существует теорема Брюера, также известная, как CAP-теорема (см., например, [3, 4]), в которой опровергается возможность одновременного достижения распределенности, строгой согласованности и доступности данных.

Попробуем разобраться, существует ли аппарат моделирования подобных распределенных систем, позволяющий подтвердить заявленные характеристики NewSQL, а, следовательно, снизить риски при широком применении подобного класса СУБД в цифровых решениях.

Поэтому поставленная в заголовке статьи задача моделирования распределенных СУБД является крайне актуальной в условиях стремительной цифровизации экономики и общества при одновременном усилении требований распределенности данных в условиях территориально-распределенной экономики, доступности и согласованности данных во всех узлах распределенной цифровой системы.

В рамках решения общей задачи моделирования распределенных СУБД, в данной статье обсуждена возможность достижения строгой согласованности при одновременной доступности и распределенности данных. Рассмотрен вопрос возможности применения математического аппарата теории массового обслуживания для моделирования СУБД, рассмотрены модели очередей для многосерверных систем массового обслуживания, например, популярные модели M/M/c и др.

Теперь перейдем к рассмотрению вопроса о возможности достижения строгой согласованности при одновременной доступности данных в распределенной информационной системе вообще, и в распределенной СУБД в частности.

1. Возможность достижения строгой согласованности при одновременной доступности и распределенности данных

Вопрос о возможности достижения строгой согласованности при одновременной доступности и распределенности данных отнюдь не праздный. Дело в том, что примерно 15 лет назад была сформулирована теорема Брюера, более известная как CAP-теорема, которая гласит, что невозможно одновременное достижение согласованности (consistency, C) данных, их доступности (availability, A) и устойчивости к распределению (partition tolerance, P) по узлам распределенной СУБД (см., подробнее [3, 4]). Формально теорему CAP можно сформулировать так:

$$(C \wedge A) \vee (C \wedge P) \vee (A \wedge P) = 1 \neq (C \wedge A \wedge P) \quad (1)$$

Таким образом, положения данной теоремы прямо противоречат заявленным характеристикам NewSQL СУБД, которые можно формализовать следующим утверждением:

$$(C \wedge A \wedge P) = 1 \quad (2)$$

Сформулированная чуть позже теорема PACELC не только подтверждает теорему CAP, но и является, по сути, ее расширением. Что также противоречит основным отличительным чертам NewSQL СУБД. Правда, теорема PACELC несколько смягчает строгие утверждения теоремы CAP и допускает возможность справедливости утверждения (2) при определенных условиях (см., подробнее [5]).

Практическим подтверждением теоремы CAP является класс СУБД, известный под обозначением NoSQL (Not only SQL). В представителях этого класса СУБД доступность и устойчивость к распределению данных достигнута за счет отсутствия их строгой согласованности, т.е. согласованности данных непосредственно после изменения на одном из узлов распределенной СУБД с копиями этих же данных на других узлах СУБД. В данном классе СУБД для достижения согласованности действует принцип eventual consistency, или «согласованность в конечном счете», согласно которому данные могут быть какое-то неопределенное время рассогласованы, но в конечном итоге по истечении неопределенного времени приходят в согласованное состояние. Но пока в распределенной СУБД состояние eventual consistency не достигнуто, любой узел распределенной СУБД при выполнении запроса может опираться на не актуальные данные, что влечет за собой риски принятия неверных решений при использовании СУБД NoSQL для цифровизации территориально-распределенного предприятия или экономики большой страны в целом. Тем не менее, появившийся примерно 20 лет назад класс СУБД NoSQL стал естественным ответом на вызов времени, связанный с необходимостью обеспечивать одновременную работу

TPC-C Top Performance Results - Clustered

Version 5 Results As of 13-Mar-2026 at 1:45 PM [GMT]

Note 1: The TPC believes it is not valid to compare prices or price/performance of results in different currencies.

All Active Results | Active Clustered Results | Active Non-Clustered Results | Currency: All | Include Historical Results

Rank	Company	System	Performance (tpmC)	Price/tpmC	Watts/KtpmC	System Availability	Database	Operating System	TP Monitor	Date Submitted
1	Alibaba Cloud	Alibaba Cloud PolarDB Limitless	2,055,076,649	.80 CNY	NR	01/27/25	Alibaba Cloud PolarDB for MySQL 8.0.2 Enterprise Edition	Alibaba Group Enterprise Linux Server 7.2 (Paladin)	Nginx 1.25.3.1	01/27/25
2	Tencent Cloud	Tencent Database Dedicated Cluster (1650 Nodes)	814,854,791	1.27 CNY	NR	06/18/23	Tencent TDSQL v10.3 Enterprise Pro Edition with Partitioning and Physical Replication	Tencent tlinux 2.2	Nginx 1.16.1	03/24/23

Рис. 1. Рейтинг теста TPC-C для Alibaba Cloud PolarDB, опубликованные на сайт tpc.org

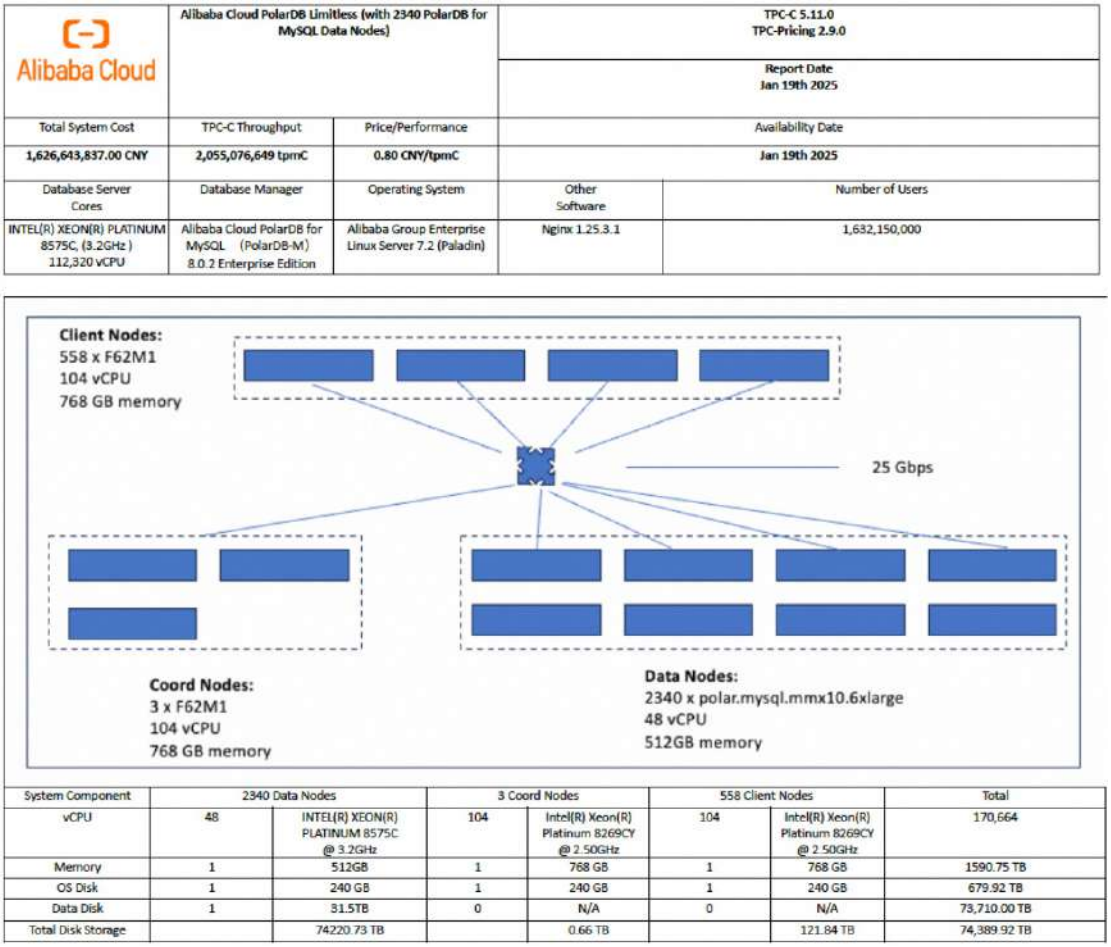
в распределенных системах хранения данных, а также с необходимостью поддерживать различные типы данных, не относящиеся только к реляционным таблицам.

Однако, стремительная цифровизация, как было сказано выше диктует более строгие требования к строгой согласованности данных, не отменяя требований к их доступности и распределенности, что привело к созданию NewSQL СУБД.

Рассмотрим вопрос, как в настоящее время подтверждается строгая согласованность данных в NewSQL СУБД при их доступности и распределенности.

2. Подтверждение строгой согласованности данных в NewSQL СУБД

Согласно данных открытой печати, единственным подтверждением на сегодняшний момент являются тесты, которые проводят разработчики крупных решений NewSQL СУБД. В настоящий момент наибольший интерес представляют 2 опу-



Alibaba Cloud

Alibaba Cloud PolarDB Limitless (with 2340 PolarDB for MySQL Data Nodes)			TPC-C 5.11.0 TPC-Pricing 2.9.0	
			Report Date Jan 19th 2025	
Total System Cost	TPC-C Throughput	Price/Performance	Availability Date	
1,626,643,837.00 CNY	2,055,076,649 tpmC	0.80 CNY/tpmC	Jan 19th 2025	
Database Server Cores	Database Manager	Operating System	Other Software	Number of Users
INTEL(R) XEON(R) PLATINUM 8575C, (3.2GHz) 112,320 vCPU	Alibaba Cloud PolarDB for MySQL (PolarDB-M) 8.0.2 Enterprise Edition	Alibaba Group Enterprise Linux Server 7.2 (Paladin)	Nginx 1.25.3.1	1,632,150,000

System Component	2340 Data Nodes		3 Coord Nodes		558 Client Nodes		Total
vCPU	48	INTEL(R) XEON(R) PLATINUM 8575C @ 3.2GHz	104	Intel(R) Xeon(R) Platinum 8269CY @ 2.50GHz	104	Intel(R) Xeon(R) Platinum 8269CY @ 2.50GHz	170,664
Memory	1	512GB	1	768 GB	1	768 GB	1590.75 TB
OS Disk	1	240 GB	1	240 GB	1	240 GB	679.92 TB
Data Disk	1	31.5TB	0	N/A	0	N/A	73,710.00 TB
Total Disk Storage		74220.73 TB		0.66 TB		121.84 TB	74,389.92 TB

Рис. 2. Состав кластера по данным сайта tpc.org

бликованных результата теста TPC-C, которые провели разработчики YugabyteDB (Yugabyte Inc., США) и PolarDB (Alibaba Cloud, КНР). Тест TPC-C – это стандартный тест для OLTP (Online transaction processing) БД для подтверждения строгой согласованности данных в результате проводимых в БД транзакций. Тест, в частности, показывает процент успешных транзакций, среднее время транзакций, распределение по типам запросов и др. Проанализируем кратко результаты этих тестов.

Начнем обзор с анализа результатов теста TPC-C для Alibaba Cloud PolarDB (КНР), т.к. данный тест является подтвержденным и лучшим на сегодняшний момент среди всех тестов TPC-C, опубликованных на сайте tpc.org (см. рис. 1 и [6]).

Рассмотрим детали теста, т.е. состав распределенного кластера СУБД, количество машин, размер БД, скорости выполнения, задержки, возникающие при обеспечении строгой согласованности транзакций.

На сайте tpc.org, кроме рейтинга теста также опубликован отчет о тесте, содержащий сведения о компьютерном оборудовании, использовавшемся для проведения теста (см. Рис.2). Согласно представленному отчету, для теста использовалась БД, распределенная по 2340 узлам кластера (виртуальные машины, каждая из которых – отдельный узел СУБД). При этом, физических процессоров в кластере 1170. Использовались также 3 узла-координатора, 558 клиентских машин для имитации работы пользователей. Подробнее характеристики кластера см. рис. 2 и [6].

В такой конфигурации результаты теста оказались очень хорошими. Почти 99% транзакций окончилось успешно (см. рис. 3), средние задержки при согласовании данных не превышают 170 мс (см. рис. 4).

С одной стороны, можно утверждать, что разработчики NewSQL СУБД добились значимых успехов при разработке распределенных СУБД со строгой согласованностью при том, что время длительность разработки подобных решений не превышает 10 лет. В то же время, заявленные результаты, подтверждающие строгую согласованность данных, приводятся не для сотен тысяч узлов, а для десятков, сотен, максимум – 2340 (и то речь идет о виртуальных машинах). Кроме того, в стандартном тесте используется ограниченный набор таблиц (7 шт.) схемы БД, набор запросов также сильно ограничен, что делает ситуацию довольно искуственной, а результаты тестов сложно переносимыми на любую распределенную схема БД.

Подробнее результаты теста и отчет см. [6].

Теперь рассмотрим результаты теста СУБД YugabyteDB, созданной по другую сторону Тихого океана (США).

Несмотря на то, что тест проведен разработчиками СУБД и не подтвержден tpc.org, он представляет безусловный интерес (см. [7]). В первую очередь тем, что для теста использовалось стандартное оборудование платформы облачных вычислений компании Amazon (AWS). Тест, согласно опубликованной информации проводился в одной зоне доступности (временной зоне) на 63 серверах (3 из них – узлы координаторы и 60 – узлы БД, на которых размещены 100000 БД (складов данных – data warehouses). Для имитации работы пользователей использовалось 40 клиентских компьютеров.

В такой конфигурации результат теста оказался очень хорошим: почти 99,8% успешных транзакций (см. рис. 5) при очень низких задержках транзакций (си. рис.5, 6).

В качестве недостакой можно сказать об ограниченном количестве узлов БД (60), что существенно ниже, чем у PolarDB. По сути речь идет о большом множестве БД, размещенных на небольшом количестве физических узлов. Также к недостаткм можно отнести, ту же ограниченность теста TPC-C для проверки строгой согласованности в распределенной СУБД.

Подробнее результаты теста и отчет см. [7].

Что можно сказать в качестве выводов по данному разделу.

New Order	
Percentage of Home order-line	99.048%
Percentage of remote order-line	0.952%
Percentage of Rolled Back Transactions	1.004%

Рис. 3. Успешность выполнения транзакций по данным сайта tpc.org

Response Time (ms)	Min Latency	Avg Latency	P90 Latency	Max Latency
New-order	101.030	138.450	163.300	59,999.900
Payment	100.657	137.156	168.100	59,999.900
Order-status	100.800	166.628	190.200	59,999.900
Stock-level	101.185	151.218	175.500	59,999.900
Delivery	100.090	103.037	101.500	11,804.000
Menu	100.071	103.272	101.400	11,813.200
Delivery_defferred	2.408	33.936	61.000	59,999.900

Рис. 4. Задержки при выполнении транзакций по данным сайта tpc.org

Результаты тестов подтверждают несомненны успехи, позволяющие утверждать, что разработчики NewSQL СУБД добились значимых успехов при разработке распределенных решений со строгой согласованностью данных. В то же время, заявленные результаты тестов приводятся не для сотен тысяч узлов БД и петабайт данных объемов БД, работоспособность на которых со строгой согласованностью утверждает разработчик СУБД.

Результаты приведенных выше тестов показывают возможность строгой согласованности для NewSQL СУБД, но только в пределах одного кластера без учета особенностей функционирования геораспределенной системы, включая необходимость согласования данных в разных временных зонах, возможную недоступность узлов (групп узлов) БД, что может теоретически приводить к рассогласованности данных на разных узлах.

Еще одним существенным минусом рассмотренных тестов является их ограниченность. Так, например, в открытой печати доступны только результаты тестов TPC-C, но нет результатов других тестов, таких как TPC-E (более сложный, чем TPC-C, тест для проверки строгой согласованности транзакций в OLTP СУБД), TPC-H (тест для проверки согласованности транзакций для аналитических запросов в OLAP СУБД), TPCx-BB (стандартный тест для «больших данных»), что позволило бы более полно представить картину функционирования распределенной СУБД со строгой согласованностью.

Results	Efficiency	99.78%
	Max theoretical tpmC	1,286,000
	Observed tpmC	1,283,254.06
Throughput and latency	Throughput	630K operations / sec
	Client side latency (new order transactions only)	~54 ms
Connections and data set	# of logical client connections	1 million
	# of physical DB connections	6000
	Data size (compressed)	11.5 TB (~35TB replicated)
	Replication factor	3
Benchmark spec	Number of warehouses	100,000
	Benchmark runtime	2 hours
	Cluster size	63 instances of c5d.12xlarge (48 vCPUs, 96 GB RAM)

Рис. 5. Параметры и результаты теста TPC-C по данным yugabyte.com

Operation Type	Throughput (ops/sec)	Avg server-side latency
SELECT	210K	3ms
UPDATE	320K	1.5ms
INSERT	90K	7ms
DELETE	10K	3ms

Рис. 6. Средние задержки транзакций со стороны сервера по данным yugabyte.com

Наличие опубликованных тестов TPC-C, строго говоря, не доказывает, что распределенная СУБД сможет эффективно работать на сотнях тысячах узлов СУБД, обеспечивая строгую согласованность при одновременной доступности и распределенности данных. Можно оптимизировать работу СУБД для успешного выполнения конкретных требований популярного теста, но при подтверждении правильности функционирования распределенной системы необходимо полноценное моделирование такой системы в различных режимах и условиях.

Моделирование функционирования СУБД необходимо, в первую очередь, для проверки возможности устойчивого функционирования такой СУБД, равномерной загрузки узлов БД и процессоров для обработки данных.

В этой связи, если рассматривать каждый элемент распределенной системы – сервер, канал передачи данных, хранилище данных, - как канал обслуживания, а порции данных для синхронизации, запросы к данным – как заявки на обслуживание, то логично рассмотреть модели теории массового обслуживания в качестве базовых для построения систем моделирования распределенных систем.

В следующем разделе статьи постараемся ответить на вопрос возможности применения математических моделей теории массового обслуживания для моделирования распределенной информационной системы.

3. Краткий анализ применимости теории массового обслуживания для моделирования распределенных систем

В современной научной литературе существует не очень много работ, посвященных моделированию распределенных систем хранения. Тем не менее в последние 2-3 года появились работы, в которых доказывается применимость математического аппарата теории массового обслуживания для распределенных информационных систем.

В работе [8] с помощью теории массового обслуживания моделируется время ожидания транзакций в очереди в блокчейн-системах. Целью моделирования является определение ключевых показателей эффективности операций в блокчейн-системе: время поступления, время обслуживания и время ожидания в очереди. В [8] доказывается достоверность и точность результатов выполненного моделирования.

В работе [9] доказывается применимость теории массового обслуживания для моделирования и «туманных» вычислительных систем. Концепция туманных вычислений предполагает вычисления в распределенной вычислительной среде и работу с информацией как в локальной, так и в глобальной сети, обеспечивая промежуточное положение между облачными дата-центрами, конечными устройствами и другими элементами инфраструктуры данных. В работе доказывается, что для интернета вещей также могут быть успешно смоделированы и применены «туманные» вычисления. Модели массового обслуживания могут помочь в такому моделировании, позволяя правильно рассчитать параметры функционирования подобных систем.

В исследовании [10] представлен инновационный подход к оптимизации распределения задач в облачных вычислениях с помощью теории массового обслуживания. В частности, моделируется планирование задач в компьютерных системах несколькими серверами и ограниченной пропускной способностью. Результаты моделирования говорят о возможности оптимизации процесса планирования и сокращении времени ожидания и длины очереди. Доказывается эффективность предложенного метода моделирования в сравнении с ранее существующими.

В работе [11] представлена аналитическая модель марковской цепи для архитектуры 5G. Предложенная модель учитывает обработку огромного количества входящих, исходящих и обрабатываемых данных в сети 5G. Доказывается эффективность разработанной авторами исследования [11] модели. В целом можно сказать, что работа [11] также косвенно доказывает применимость теории моделей массового обслуживания для моделирования распределенных систем.

Работа [12] посвящена в целом доказательству применимости теории массового обслуживания для задачи моделирования производительности широкого класса компьютерных систем.

Интересным исследованием является работа [13], в которой доказывается, что при использовании искусственного интеллекта и нейросетей возможно повышение эффективности моделирования управления очередями и планирования задач для облачных вычислений. В работе [13] показано, что среднее время ожидания в очереди сокращается на 30%, длина очереди – на 25%. Доказывается достоверность полученных результатов моделирования с 95% доверительной вероятностью. В работе показано, что объединение AI с методами моделирования систем массового обслуживания помогает совершенствовать процесс моделирования.

Хотя в представленных работах не говорится о NewSQL СУБД, тем не менее, на основании проведенного краткого обзора можно считать доказан-

ным применимость моделей массового обслуживания для моделирования распределенных систем хранения и обработки информации, к которым, без сомнения относятся и NewSQL СУБД.

Теперь рассмотрим конкретные модели теории массового обслуживания и проанализируем их применимость к задаче моделирования распределенных СУБД.

4. Моделирования распределенных СУБД с помощью моделей теории массового обслуживания

Так как распределенные СУБД зависят от компьютерного оборудования и возможностей управления ресурсами этого оборудования, логично обратиться к так называемым моделям массового обслуживания в операционных системах (см., например, [12, 14, 15]). Такие модели используются для оценки эффективности алгоритмов управления ресурсами в операционных системах (ОС), в том числе оценки задержек обработки запросов на обслуживание в ОС, пропускной способности, производительности и т.д.

Для начала рассмотрим основные термины, использующиеся в данной теории.

Запрос – это задача, требующая обслуживания в ОС. Под запросом в СУБД может пониматься любой запрос на выборку данных, запись или удаление данных, а также запрос на синхронизацию данных между узлами БД для обеспечения строгой согласованности.

Очередь – это множество запросов, ожидающих обслуживания.

Сервер – это любой канал обслуживания, обрабатывающий запросы. Под сервером в СУБД может пониматься как узел БД (собственно сервер), так и оперативная память, канал передачи данных вычислительной сети, процессор, сопроцессор и т.д. Тогда каждый сервер обладает своими параметрами, объемом, скоростью обработки/передачи данных и другими ограничениями и может рассматриваться как независимый канал обслуживания с точки зрения теории массового обслуживания.

С этой точки зрения, распределенные СУБД следует рассматривать либо как многосерверные системы массового обслуживания, что близко к реальности, при этом под сервером будем понимать любой канал обслуживания из перечисленных в предыдущем абзаце. Либо как набор взаимосвязанных односерверных систем массового обслуживания, что проще для моделирования конкретного узла распределенной СУБД, но не соответствует реальной картине, т.к. сложно будет строить взаимосвязи между серверами.

Обычно второй вариант используется для моделирования розничных магазинов, транспортных систем и т.д. из-за относительной независимости серверов друг от друга и возможностью пренебречь их взаимовлиянием. Тогда как первый вариант используется для распределенных информационных систем и нейронных сетей, которые распределяют нагрузку между несколькими серверами, следовательно, такие модели учитывают взаимовлияние каналов обслуживания (серверов, нейронов и т.д., см. [12, 14, 15]).

Для описания моделей многосерверных систем используются обозначения Кендалла, которые представляют собой трёхсимвольный код, обозначающий: процесс поступления запросов, распределение обслуживания и количество серверов.

Например, обозначение для модели M/M/1 означает систему, в которой поступление запросов рассматривается как марковский процесс с пуассоновским (случайным) распределением времени поступления запросов (M), экспоненциальным распределением времени обслуживания заявок (M) и одним сервером (1). M/M/1 часто используется в качестве отправной точки для понимания работы более сложных систем массового обслуживания, в том числе многосерверных, поэтому нужно подробнее описать оцениваемые с помощью этой системы показатели эффективности.

Пусть частота поступления запросов: λ, частота обслуживания запросов сервером: μ.

Тогда, согласно формуле Эрланга-С, вероятность того, что сервер не занят обработкой запросов: $P_0 = (\lambda/(\lambda+\mu))$. Среднее количество запросов в системе: $L = (\lambda/\mu)/(1 - P_0)$. Среднее время обработки запроса: $W = 1/(\mu - \lambda)$, здесь и далее при условии, что $\mu > \lambda$.

Вероятность того, что в системе будет одновременной n запросов для обработки (рассчитывается с помощью распределения Пуассона): $P(n)= (\lambda^n/n!)\, e^{-\lambda}$ (1).

При этом, если имеются ограничения на количество запросов, например, вследствие ограничения памяти сервера, то оценивается также вероятность отказа $P_R = P(n > N)$, равную вероятности ожидания клиентов в очереди при $n > N$, или $P_R = P(n=N+1) + P(n=N+2) + ...$ (см. формулу (1) для каждого слагаемого). Это соотношение также означает, что если длина очереди ограничена, а скорость поступления запросов превышает скорость обслуживания, то, в конечном итоге, это приведёт к высокому проценту отказов и запросам придётся долго ждать, прежде чем их обслужат.

Вариантом модели M/M/1 является односерверная модель M/D/1. D обозначает детерминированное время обслуживания, т.е., что запросы всегда обслуживаются в течение одного и того же времени.

Для данной модели оцениваются следующие показатели эффективности.

Коэффициент использования сервера: $U = (\lambda/(\lambda+\mu))$. Вероятность того, что сервер не занят обработкой запросов (рассчитывается по формуле Эрланга-B): $P_0 = (\lambda/\mu)/(1+(\lambda/\mu))$.

Среднее количество запросов в системе: $L = \lambda \times U/(1 - U)$. Среднее время обработки запроса: $W = 1/(\mu - \lambda)$.

M/D/1 – это частный случай более общей модели G/D/1, в которой время поступления запросов является общим и не обязательно соответствует марковскому процессу.

Еще одним важным расширением модели M/M/1 является модель M/G/1, в которой время обслуживания запроса имеет общую функцию распределения. M/G/1 также широко используется, как и M/M/1, например, для задачи моделирования производительности жесткого диска.

В этом случае для оценки показателей эффективности используются более сложные соотношения, так, например, для оценки загруженности/простоя сервера P_0 используется преобразование Лапласа функции плотности вероятности времени обслуживания s.

Для оценки остальных показателей используются следующие соотношения. Среднее количество запросов в системе: $L = (\lambda/\mu) + ((\lambda/\mu)^2 + \lambda^2 \times D(s))/(2 \times (1 - (\lambda/\mu)))$, где $D(s)$ – дисперсия распределения времени обслуживания s.

Среднее время обработки запроса: $W = ((\lambda/\mu) + \lambda \times \mu \times D(s))/(2 \times (\mu - \lambda)) + 1/\mu$.

Тем самым получается, что для реализации такой модели необходимо знание о функции распределения времени обслуживания, что не всегда возможно.

Рассмотрев односерверные модели, перейдем к рассмотрению типовых многосерверных моделей. Рассмотрение начнем с популярной модели массового обслуживания Эрланга-C – M/M/c. Модель описывает систему, в которой есть ***c*** серверов, множество запросов (очередь) обслуживается этими серверами одновременно.

Эта модель применяется для моделирования ситуаций, в которых обслуживание осуществляется несколькими серверами. Показатели производительности этой системы аналогичны показателям M/M/1. Однако показатели рассчитать сложнее, поскольку на поведение очереди влияет количество серверов c.

В M/M/c запросы поступают в систему независимо и случайно с постоянной интенсивностью по времени, время обслуживания каждого запроса также случайно и экспоненциально распределено.

Пусть частота поступления запросов: λ, частота обслуживания запросов на каждом сервере: μ. Тогда показатели эффективности такой системы в простейшем случае можно оценить следующими соотношениями.

Коэффициент использования серверов:

$$U = \lambda / (c \times \mu).$$

Вероятность того, что многосерверная система в целом не занята обработкой запросов (рассчитывается по формуле Эрланга-C):

$$P_0 = U^c / (c! \times (1 - U)) =$$
$$= (\lambda /(c \times \mu))^c / (c! \times (1 - (\lambda /(c \times \mu)))),$$
где $c! = c \times (c-1) \times (c-2) \times ... \times 1$.

Среднее количество запросов в системе: $L = (\lambda / (c \times (\mu - \lambda)))$.

Среднее время, которое запрос проводит в системе: $W = 1 / (c \times (\mu - \lambda))$.

Расширением или более общим случаем моделей M/M/c и M/G/1 является модель M/G/c, в которой для обработки запросов существует c серверов, а время обслуживания запроса имеет общую функцию распределения. Открытой проблемой является то, что в настоящее время показатели эффективности для модели M/G/c до конца не определены.

Таким образом, можно сделать вывод, что для моделирования распределенной СУБД из рассмотренных моделей наиболее подходящей является M/M/c.

5. Вопросы практического применения

Обосновав необходимость моделирования распределенных СУБД, применимость моделей теории массового обслуживания для подобного моделирования, а также рассмотрев имеющиеся популярные модели, была выполнена программная реализация модели M/M/c.

Данная реализация выполнена в рамках выпускной квалификационной работы, опробована на тестовых данных. Программа имеет потенциал развития, а именно возможность реализации более сложных моделей теории массового обслуживания, а также расширение моделей за счет задания ограничений на каждый сервер.

Заключение

В статье выполнен анализ возможности моделирования современных распределенных СУБД

с помощью многосерверных моделей теории массового обслуживания. В статье показана актуальность решаемой задачи вследствие распространения распределенных СУБД со строгой согласованностью класса NewSQL.

Обсуждена возможность достижения строгой согласованности при одновременной распределенности и доступности данных, в том числе, подтверждение строгой согласованности с помощью тестов, достоинства и недостатки тестов.

Рассмотрен вопрос возможности применения математического аппарата теории массового обслуживания для моделирования распределенных СУБД, рассмотрены модели очередей для многосерверных систем массового обслуживания, в том числе, популярные модели M/M/c и др. Разработана программная реализация для моделирования распределенных СУБД с помощью моделей теории массового обслуживания.

В дальнейших исследованиях планируется расширение количества программных реализаций моделей для проведения моделирования распределенных СУБД для подтверждения возможности обеспечения строгой согласованности данных.

Безусловно, только вопросами обеспечения строгой согласованности, моделирование распределенных систем ограничиваться не может, важно моделировать процессы синхронизации данных, контроль устаревания данных, особенно, когда данные расположены в разных часовых поясах, процессов записи и чтения данных с контролем времени создания. Этим и другим вопросам моделирования распределенных СУБД будут также посвящены отдельные исследования, т.к. рассмотреть их в рамках данного исследования невозможно из-за ограниченного объема статьи.

Литература

1. Соловьев А.В. Организация хранения данных в распределенных СУБД со строгой согласованностью данных // Информационные технологии и вычислительные системы. 2025. №2. С.113-122. DOI: 10.14357/20718632250210. EDN: BFSUXU.
2. Solovyev A.V. Digital Twins and the Problem of Distributed Data Storage // IEEE Xplore (Institute of Electrical and Electronics Engineers), Proceedings - 2025 International Russian Smart Industry Conference, SmartIndustryCon 2025. 2025. P.77–82. DOI: 10.1109/SmartIndustryCon65166.2025.10985972.
3. Brewer E. A certain freedom: thoughts on the CAP theorem // PODC ‘10: Proceedings of the 29th ACM SIGACT-SIGOPS symposium on Principles of distributed computing. 2010. P.335. DOI: https://doi.org/10.1145/1835698.1835701.
4. Frank L., Pedersen R.U., Frank C.H., Larsson N.J. The CAP theorem versus databases with relaxed ACID properties // ICUIMC ‘14: Proceedings of the 8th International Con-ference on Ubiquitous Information Management and Communication. 2014. P.1-7. DOI: https://doi.org/10.1145/2557977.2557981.
5. Golab W. Proving PACELC // ACM SIGACT News. 2018. Vol.49. No 1. P.73-81. DOI: https://doi.org/10.1145/3197406.3197420.
6. TPC-C Top Performance Results – Clustered// tpc.org – URL: https://www.tpc.org/tpcc/results/tpcc_perf_results5.asp?resulttype=CLUSTER&version=5%&currencyID=0 (дата обращения: 03.03.2026).
7. Hemant B., Sonal A. TPC-C Benchmark: Scaling YugabyteDB to 100,000 Warehouses // yugabyte.com – URL: https://www.yugabyte.com/blog/tpc-c-benchmark-100000-warehouses-yugabytedb/ (дата обращения: 03.03.2026).
8. Kandpal M., Keshari N., Yadav A.S. et al. Modelling of blockchain based queuing theory implementing preemptive and non-preemptive algorithms // Int J Syst Assur Eng Manag. 2024. Vol.15. P.2554–2570. DOI: https://doi.org/10.1007/s13198-024-02276-0.
9. Mas L., Vilaplana J., Mateo J. et al. A queuing theory model for fog computing // J Super-comput. 2022. Vol. 78. P.11138–11155. DOI: https://doi.org/10.1007/s11227-022-04328-3.
10. Sinha A., Banerjee P., Roy S. et al. Improved Dynamic Johnson Sequencing Algorithm (DJS) in Cloud Computing Environment for Efficient Resource Scheduling for Distributed Overloading // J. Syst. Sci. Syst. Eng. 2024. Vol.33. P.391–424. DOI: https://doi.org/10.1007/s11518-024-5606-z.
11. Aliyu A.L., Diockou J. An Analytical Queuing Model Based on SDN for IoT Traffic in 5G // Barolli, L. (eds) Advanced Information Networking and Applications. AINA 2023. Lecture Notes in Networks and Systems. 2023. Vol.655. P. 435–445. DOI: https://doi.org/10.1007/978-3-031-28694-0_42.
12. Kounev S., Lange KD., von Kistowski J. Operational Analysis and Basic Queueing Models // Systems Benchmarking. 2025. P.149–184. DOI: https://doi.org/10.1007/978-3-031-85634-1_7.
13. Chaudhary H., Sharma G., Nishad D.K. et al. AI-enhanced modelling of queueing and scheduling systems in cloud computing // Discov Appl Sci. 2025. Vol.7. P.276. DOI: https://doi.org/10.1007/s42452-025-06755-2.

14. Queuing Models in Operating System // geeksforgeeks.org. – URL: https://www.geeksforgeeks.org/operating-systems/queuing-models-in-operating-system/ (дата публикации 09.04.2025).
15. Печинкин А.В., Разумчик Р.В. Системы массового обслуживания в дискретном времени. М.: Физматлит, Россия, 2018. 432 с.

Соловьев Александр Владимирович. Федеральный исследовательский центр «Информатика и управление» Российской академии наук, г. Москва, Россия. Главный научный сотрудник, доктор технических наук. Область научных интересов: системный анализ, системы управления базами данных, теория надежности, математическое моделирование, долговременное хранение электронных документов. E-mail: soloviev@isa.ru (Ответственный за переписку)

Кожевникова Дарья Сергеевна. Образовательное частное учреждение высшего образования «Православный Свято-Тихоновский гуманитарный университет», г. Москва, Россия. Студент бакалавриата. Область научных интересов: системы управления базами данных, теория массового обслуживания, математическое моделирование. E-mail: dashikozh@gmail.com

Modeling Distributed DBMSs Using Queuing Theory

A.V. Solovyev[I,II], D.S. Kozhevnikova[II]

[I] Federal Research Center “Computer Science and Control” of the Russian Academy of Sciences, Moscow, Russia

[II] Private educational institution of higher education Saint Tikhon’s Orthodox University for the Humanities, Moscow, Russia

Abstract. The article analyzes the possibility of modeling modern distributed DBMS using multi-server models of the theory of the queue of the service. The article shows the relevance of the problem solved due to the spread of distributed DBMS with strict consistency of the NewSQL class. The possibility of achieving strict consistency with simultaneous distribution and availability of data is discussed, including confirmation of strict consistency using tests, advantages and disadvantages of tests. The possibility of using the mathematical apparatus of the theory of mass service for modeling distributed DBMS is considered, queue models for multi-server mass service systems, including the popular M/M/c model, etc. are considered. A software implementation for modeling distributed DBMS using the models of the theory of mass service is developed. In further research, it is planned to expand the number of software implementations of models for conducting distributed DBMS modeling to confirm the possibility of ensuring strict data consistency.

Keywords: *database management systems, NewSQL DBMS, queuing theory, multi-server models, queue modeling, M/M/c.*

DOI: 10.14357/20790279260206 **EDN:** MFWSFO

References

1. Solovyev A.V. Organization of data storage in distributed databases with strict data consistency. *Journal of Information Technologies and Computation Systems*. 2025;(2):113-122. DOI: 10.14357/20718632250210. EDN: BFSUXU (In Russ).
2. Solovyev A.V. Digital Twins and the Problem of Distributed Data Storage. *IEEE Xplore (Institute of Electrical and Electronics Engineers), Proceedings - 2025 International Russian Smart Industry Conference, SmartIndustryCon 2025*. 2025;77–82. DOI: 10.1109/SmartIndustryCon65166.2025.10985972.
3. Brewer E. A certain freedom: thoughts on the CAP theorem. *PODC ‘10: Proceedings of the 29th ACM SIGACT-SIGOPS symposium on Principles of distributed computing*. 2010;335. DOI: https://doi.org/10.1145/1835698.1835701.
4. Frank L., Pedersen R.U., Frank C.H., Larsson N.J. The CAP theorem versus databases with relaxed ACID properties. *ICUIMC ‘14: Proceedings of the 8th International Con-ference on Ubiquitous Information Management and Communication*. 2014;1-7. DOI: https://doi.org/10.1145/2557977.2557981.
5. Golab W. Proving PACELC. *ACM SIGACT News*. 2018;49(1):73-81. DOI: https://doi.org/10.1145/3197406.3197420.
6. *TPC-C Top Performance Results – Clustered.* Available from: https://www.tpc.org/tpcc/results/tpcc_perf_results5.asp?resulttype=CLUSTER&version=5%&currencyID=0 [Accessed 03 March 2026].

7. Hemant B., Sonal A. *TPC-C Benchmark: Scaling YugabyteDB to 100,000 Warehouses.* Available from: https://www.yugabyte.com/blog/tpc-c-benchmark-100000-warehouses-yugabytedb/ [Accessed 03 March 2026].
8. Kandpal M., Keshari N., Yadav A.S. et al. Modelling of blockchain based queuing theory implementing preemptive and non-preemptive algorithms. *Int J Syst Assur Eng Manag*. 2024;(15):2554–2570. DOI: https://doi.org/10.1007/s13198-024-02276-0.
9. Mas L., Vilaplana J., Mateo J. et al. A queuing theory model for fog computing. *J Super-comput*. 2022;(78):11138–11155. DOI: https://doi.org/10.1007/s11227-022-04328-3.
10. Sinha A., Banerjee P., Roy S., et al. Improved Dynamic Johnson Sequencing Algorithm (DJS) in Cloud Computing Environment for Efficient Resource Scheduling for Distributed Overloading. *J. Syst. Sci. Syst. Eng*. 2024;(33):391–424. DOI: https://doi.org/10.1007/s11518-024-5606-z.
11. Aliyu A.L., Diockou J. An Analytical Queuing Model Based on SDN for IoT Traffic in 5G. *Advanced Information Networking and Applications. AINA 2023. Lecture Notes in Networks and Systems*. 2023;(655):435–445. DOI: https://doi.org/10.1007/978-3-031-28694-0_42.
12. Kounev S., Lange K.D., von Kistowski J. Operational Analysis and Basic Queueing Models. *Systems Benchmarking*. 2025:149–184. DOI: https://doi.org/10.1007/978-3-031-85634-1_7.
13. Chaudhary H., Sharma G., Nishad D.K., et al. AI-enhanced modelling of queueing and scheduling systems in cloud computing. *Discov Appl Sci*. 2025;(7):276. DOI: https://doi.org/10.1007/s42452-025-06755-2.
14. *Queuing Models in Operating System*. April 2025. Available from: https://www.geeksforgeeks.org/operating-systems/queuing-models-in-operating-system/ [Accessed 03 March 2026].
15. Pechenkin A.V., Razumchik R.V. *Discrete-time queueing systems*. Moscow, "Fizmatlit"; 2018. 432 p. (In Russ).

Solovyev Alexander V. Chief Researcher, Doctor of Technical Sciences. Federal Research Center "Computer Science and Control" of the Russian Academy of Sciences, 44/2 Vavilova str., Moscow, 119333, Russia. E-mail: soloviev@isa.ru

Kozhevnikova Darya S. Student of Bachelor Course. Private educational institution of higher education Saint Tikhon's Orthodox University for the Humanities. 23B, Novokuznetskaya str., Moscow, 115184, Russia. E-mail: dashikozh@gmail.com