Оглавление

Благодарности

Автор выражает искреннюю признательность коллегам за помощь в подготовке книги. Прежде всего хотелось бы поблагодарить Клуб топ-менеджеров «Клуб директоров ИТ — 4 СИО.Ру», редакционную коллегию учебников по цифровой трансформации 4CIO и 4CDTO[4], персонально **Сергея Кирюшина**, **Феликса Карасёва**, **Евгения Борисова**, **Дмитрия Северова**, **Алексея Кравченко** за полученный опыт разработки глав учебника 4CIO/4CDTO, мотивацию к созданию собственной книги, конструктивную критику и отзывы.

Автор благодарит Рената Лашина, исполнительного директора АРПП «Отечественный софт», и Игоря Дорофеева, президента Ассоциации участников отрасли ЦОД, за высокую оценку издания и отзывы.

Автор благодарит профильных экспертов за активное участие в обсуждении книги, тщательный анализ материала, критику, ценные замечания и конструктивные предложения:

- **Антона Жбанкова**, архитектора ЦОД и инфраструктуры высоконагруженных распределённых систем, @antonvirtual;
- **Александра Веретенникова**, доцента кафедры КВМиКН УрФУ, эксперта в области алгоритмов и структур данных, @veretennikovab;
- **Ростислава Глузмана**, архитектора распределённых высоконагруженных систем, @Rostislav_G;
- **Сергея Шелудько**, архитектора распределённых высоконагруженных программных систем управления;
- **Станислава Осипова**, архитектора операционных систем, @stas_os;
- **Игоря Гальцева**, эксперта в области создания и интеграции систем массового обслуживания, @galtsevia;
- **Михаила Малышева**, эксперта в области ИИ и машинного обучения. Михаил ведёт канал «Технозаметки Малышева», @tsingular[5].

Автор выражает благодарность опытному старшему товарищу, автору книг[6] по цифровизации АО «Концерн Росэнергоатом», а именно «Центры обработки данных», «Цифровой двойник», «Цифровая трансформация», «Искусственный интеллект», «Робототехника» и «Микроэлектроника», **Александру Прохорову** за комментарии, которые позволили сделать книгу более качественной.

[4] Учебник 4CDTO, #A4.
[5] Технозаметки Малышева, #A5.
[6] Digital Atom — книги, #A6.



Об авторе

Вадим Подольный с 2000 г. работает в области разработки программного обеспечения промышленных распределённых высоконагруженных систем управления. За 25 лет Вадим принял участие во многих крупных инфраструктурных проектах, связанных с обеспечением технологической независимости ключевых отраслей промышленности. Будучи главным конструктором, он руководил разработкой российской программной платформы АСУ ТП АЭС класса DCS/SCADA (ПО распределённых технологий автоматизации лицензированное, ПОРТАЛ) в ГК «Росатом». В настоящее время на базе этой разработки создано и успешно введено в эксплуатацию множество систем верхнего уровня АСУ ТП на современных энергоблоках АЭС с реакторами ВВЭР-1000, ВВЭР-1200 и БН-800. Вадим был главным конструктором и руководителем разработки операционной системы специального назначения «Заря», разрабатываемой в ГК «Ростех». Как эксперт Вадим участвовал в разработке решений в области распределённых автоматизированных систем для государственных корпораций и ряда коммерческих структур. Сейчас Вадим руководит несколькими проектами в области разработки промышленных распределённых высоконагруженных систем управления, обработки и хранения данных. Он является руководителем комитета промышленной автоматизации Ассоциации разработчиков программных продуктов (АРПП «Отечественный софт»), членом совета Клуба топ-менеджеров 4CIO, членом программного комитета и постоянным спикером многих профильных конференций. Вадим ведёт авторский курс «Архитектура высоконагруженных систем» в Университете Иннополис[7].

Вадим окончил факультет технической физики («Ф») Национального исследовательского ядерного университета МИФИ (Московский инженерно-физический институт) по специальности «Ядерные реакторы и энергетические установки», аспирантуру Всероссийского научно-исследовательского института по эксплуатации АЭС (ВНИИАЭС), получил дополнительное профессиональное образование в области автоматизированных систем управления технологическими процессами, разработки программного обеспечения и индустриальной кибербезопасности.

Связаться с Вадимом: *vadim@podolny.ru (email), @vpp01(tg)*[8]

[7] Иннополис, #A7.
[8] Podolny.ru, #A8.

О чём эта книга

Идея создания книги появилась после долгого обсуждения с техническими заказчиками подходов к разработке высоконагруженных распределённых систем. Всегда возникал один и тот же вопрос: «Как будем делать эту систему?» — а за ним один и тот же ответ: «Да вот так и будем!» Заказчик уходил думать, пытаясь «сколхозить» решение собственными силами, а когда в очередной раз не получалось, возвращался и говорил: «Вот ещё требования добавились, как делать-то будем?» Да всё так же! И это могло повторяться снова и снова достаточно долго. Бывало так: приходишь, а там уже другой человек, и всё начинается сначала. Мне это надоело, и я решил, что вместо объяснений буду вручать эту книгу. Конечно, это шутка, но в каждой шутке есть доля правды.

Эта книга не претендует на звание универсального свода знаний о высоконагруженной обработке данных или создании систем реального времени. Она лишь отражает некоторый опыт в этой области. Мой опыт в основном касается создания распределённых систем управления промышленными критическими информационными системами. В таких системах присутствуют сотни тысяч источников изменений данных и их потребителей. Сценарии управления зависят от характера и интенсивности этих изменений. Возможно, экспертам в области корпоративных информационных систем используемая терминология покажется несколько непривычной, но она появилась из-за сильно отличающихся от ИТ и отчасти завышенных требований к промышленным системам.

Возникает вопрос: как обуздать этот хаос? Нужен *ordo ab chao* (порядок из хаоса). А что, если из хаоса достаточно создать порядок, но не полный? Как вы обычно просите своего ребёнка, чтобы привёл в порядок свою комнату? Каковы метрики требуемого результата? С какого момента беспорядок можно считать в большей степени порядком, чем беспорядком? Так происходит и с данными в высоконагруженных системах. Чтобы с ними можно было работать, данные должны быть целостными. А какими должны быть метрики целостности и согласованности данных, позволяющие однозначно сказать, готовы данные к обработке или нет? Ведь если управлять, например, опасным производством, основываясь на неконсистентных (устаревших) данных, может случиться авария.

Книга даёт представление о том, как проектируются высоконагруженные распределённые системы управления; она адресована всем, кто хочет разобраться, как устроены и создаются такие системы.

Предисловие к 3-му изданию, дополненному

Первые два издания получили признание широкого круга читателей и собрали множество положительных отзывов и конструктивных замечаний, которые учтены в настоящей редакции. Третье издание существенно дополнено и переработано.

Книга обрела большую цельность — с точки зрения как структуры, так и логики изложения. Добавлен значительный объём нового материала, подкреплённый многочисленными примерами. Большинство глав пересмотрены с учётом экспертной критики.

Изменения отражают профессиональный и преподавательский опыт автора, полученный в Университете Иннополис и в рамках других образовательных программ. Этот опыт позволил оптимизировать стиль изложения, лучше структурировать основные акценты и включить новые важные темы.

Особое внимание уделено последовательному сравнению архитектурных подходов на примерах двух принципиально разных областей: систем промышленной автоматизации и систем хранения данных. Эти домены часто предъявляют противоположные требования к репликации данных, предсказуемости задержек и масштабированию. Такой контрастный анализ позволяет глубже раскрыть универсальные закономерности и архитектурные компромиссы, определяющие современный подход к проектированию высоконагруженных распределённых систем.

Обращение к читателю Евгения Борисова

Когда я читал предыдущую редакцию, она уже казалась фундаментальной и системной работой. Третье издание выросло почти вдвое: добавлены новые главы, в том числе про архитектурные шаблоны распределённых систем управления и современные приёмы их реализации. Автор проделал огромную работу по обновлению и переосмыслению содержания, фактически создав новую книгу!

Вадим Подольный — признанный специалист в своей области. Мне хорошо знаком стиль и подход автора: до этого мы вместе работали над «Учебником CDTO», для которого он подготовил главы об Интернете вещей и АСУ ТП. А я был заместителем главного редактора и соавтором ряда глав. Этот опыт позволил увидеть его системный подход, внимательность к деталям, точность формулировок и умение объяснять сложные технологические темы простым языком. Его опыт основан на более чем двадцати пяти годах в разработке промышленных распределённых систем управления и обработки данных. Сегодня Вадим руководит технологическими проектами, возглавляет комитет промышленной автоматизации АРПП «Отечественный софт», входит в экспертный совет 4CIO и преподаёт в Университете Иннополис. Это уровень, задающий высокую планку технической точности во всей книге.

Автор выстраивает повествование от терминов и фундаментальных понятий к архитектурным принципам и далее к шаблонам и практической инженерии. Он словно строит дом, возводя каркас от определений через проблематику, ограничения, решения и принятые архитектурные практики к приёмам их реализации. Это делает чтение системным и не даёт потеряться в сложности описываемой темы. Отличительные черты автора, которые нашли своё отражение и в этой книге, — это системность и структурированность.

Мы живём в эпоху, когда начинают активно применяться цифровые двойники, возрастает количество и качество ИИ-систем, edge- и fog-архитектуры становятся нормой, корпорации строят внутренние платформы и, соответственно, растут нагрузки и актуализируются требования к распределённому хранению и вычислениям. С точки зрения B2B-продуктов, венчурных проектов или оценки технологий в крупных организациях грамотная инженерия распределённых систем становится важным фактором устойчивости и стоимости продукта. Книга автора охватывает всю эту область комплексно и современно.

Книга читается легко, на одном дыхании. Она определённо станет своего рода энциклопедией высоконагруженных и распределённых систем, системным фундаментом, на который можно наслаивать новые знания и практические подходы!

Евгений Валерьевич Борисов,
директор по развитию Фонда развития интернет-инициатив[9]*,*
главный редактор книги «Продуктовый подход.
Как создавать продукты, которые зарабатывают»,
заместитель главного редактора книги «Учебник для CDTO»,
член грантового комитета РФРИТ,
Экспертного совета фонда Сколково
Комитета по развитию рынка инноваций Мосбиржи,
совета клуба топ-менеджеров «Клуб директоров ИТ — 4 СИО.Ру»

[9] ФРИИ, #A9.

Приветственное слово Рената Лашина

Уважаемые читатели!

Современные тенденции развития программного обеспечения неразрывно связаны с необходимостью создания эффективной программной архитектуры. Наиболее чувствительной задачей, важной для экономики Российской Федерации, является разработка программных продуктов, направленных на решение задач построения распределённых систем управления объектами критической информационной инфраструктуры, такими как крупные промышленные предприятия, банки, сервисы массового обслуживания клиентов. В современных условиях необходимо уметь разрабатывать такие системы силами отечественных разработчиков. Именно перед российскими разработчиками стоит задача создания программных решений для обеспечения технологической независимости страны. Опыт, накопленный автором в рамках решения указанных задач, консолидирует теоретические основы разработки распределённых высоконагруженных систем управления, которые будут полезны коллегам и начинающим разработчикам.

Ренат Леонидович Лашин,
исполнительный директор
Ассоциации разработчиков программных продуктов
(АРПП) «Отечественный софт»

Приветственное слово Игоря Дорофеева

Уважаемые читатели!

Объём общемировых данных ежегодно увеличивается на 30%. Однако за этой формальной цифрой скрываются важные качественные изменения.

Во-первых, растёт зависимость человечества от данных, их доступности и скорости обработки. Мы уже воспринимаем постоянное использование цифровых сервисов как нечто само собой разумеющееся. На вопрос, какой центр обработки данных (ЦОД) можно считать хорошим, есть элегантный ответ: тот, о котором вы не знаете или забыли. Обычно мы вспоминаем о ЦОДах только тогда, когда какой-то сервис — будь то платёжный процессинг, заказ такси или что-то иное — становится недоступен, вызывая заметные неудобства.

Во-вторых, из-за роста объёма «сырых» данных, которые генерируются относительно легко, повышается сложность их обработки и структурирования. Вопросы эффективности, оптимизации и защищённости становятся перманентными. Таким образом, развитие информационных технологий как отрасли и информационных систем как практики остаётся крайне динамичным. Даже архитектура ЦОДов, несмотря на их основательность как объекта недвижимости, вынуждена адаптироваться к этим изменениям, следуя за эволюцией ИТ-систем в силу своего обеспечивающего характера.

Уверен, эти тезисы подтверждают актуальность тематики данной книги. Вектор развития принципов построения информационных систем смещается в сторону подходов, ориентированных на обработку критически важных данных в режиме реального времени.

Отдельно хочу выделить ещё один важный, но неочевидный вопрос. Сегодня во многих передовых отраслях знания формируются практически «с колёс». К сожалению, львиная доля публичной информации — это фейки, мифы и манипуляции. Существует острая проблема систематизации знаний и создания качественной основы для дальнейшего развития. Частично она была решена в этой монографии, за что я выражаю автору большую благодарность.

Игорь Викторович Дорофеев,
президент Ассоциации участников отрасли ЦОД

Приветственное слово Дмитрия Комиссарова

Уважаемые читатели!

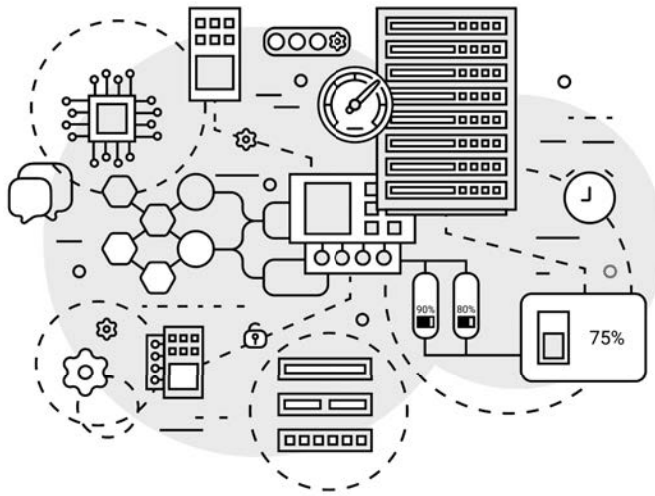
Неоднократно слышал от топ-менеджеров крупнейших ИТ-компаний, что десятикратный рост аудитории почти всегда является приговором старой архитектуре и приводит к радикальной смене стека. Вадим — один из немногих на российском рынке практиков, кто проходил через такие трансформации неоднократно, раз за разом добиваясь результата.

Сегодня, когда стоимость ресурсов растёт, а обычная память может подорожать в пять раз, вопрос «покупать ещё железо или оптимизировать архитектуру» становится вопросом выживания бизнеса.

Но главное даже не в железе. В эпоху «вайбкодинга», когда нейросети берут на себя рутину, рынок джунов и мидлов ждёт серьёзная встряска. В этих условиях ценность архитектора, способного работать собственной головой, а не только подсказками ИИ, возрастает многократно. Код может сгенерировать машина, но спроектировать жизнеспособную, устойчивую систему может только глубокий человеческий интеллект.

В книге Вадима собраны актуальные, проверенные временем и практикой современные подходы для решения самых масштабных задач. Это инвестиция в ваше архитектурное мышление. Читайте — сэкономленное время и ресурсы бесценны.

Дмитрий Владимирович Комиссаров,
член правления
Ассоциации разработчиков программных продуктов
(АРПП) «Отечественный софт»,
основатель компаний «РОСА» («РОСА Линукс»),
«Новые Облачные Технологии» («МойОфис»)



Глава 1. Введение

Что такое высоконагруженная система (ВНС)? Очевидно, ВНС — это система, которая обрабатывает высокие нагрузки. А что такое высокая нагрузка? Как обрабатывает? На чём обрабатывает? Насколько эффективно обрабатывает? На все эти вопросы есть ответы. Высокая нагрузка — состояние, при котором система приближается к пределу своих возможностей. Любая ВНС должна обрабатывать данные эффективно, то есть задействуя все доступные ресурсы аппаратного обеспечения: вычислительную мощность всех ядер процессоров, оперативную память, сетевые устройства.

Недопустимо, чтобы из-за низкой эффективности (производительности) сетевых карт процессоры оставались загруженными не полностью. Такая ситуация означает, что аппаратное обеспечение выбрано неэффективно с точки зрения решения конкретной задачи обработки данных. Задачи обработки данных бывают самыми разными, их можно классифицировать.

Для каждого класса задач обработки существует своя эффективная (оптимальная) аппаратная конфигурация. С экономической точки зрения, аппаратную конфигурацию рекомендуется подбирать таким образом, чтобы при максимальной загрузке все аппаратные компоненты использовались

с максимальной эффективностью, а отдельные узлы и компоненты не простаивали.

Если попытаться точно и всеобъемлюще сформулировать определение ВНС, то звучать оно будет следующим образом. ВНС — это система, предназначенная для обработки большого объёма запросов, данных или пользователей с минимальными задержками, высокой доступностью и стабильной производительностью даже при пиковых нагрузках. Она должна эффективно масштабироваться, обеспечивать отказоустойчивость и сохранять целостность данных при интенсивной работе.

Современные задачи требуют серьёзных вычислительных ресурсов. Какой бы ни была конфигурация оборудования самого мощного сервера, который только можно собрать, одного такого устройства никогда не будет достаточно, чтобы решить задачу обработки высоких нагрузок, обусловленных наличием множества источников и потребителей данных. В ВНС обработку данных принято осуществлять на многочисленных объединённых в сеть устройствах, распределяя нагрузки между ними.

Оптимальное разделение нагрузки между связанными в единую сеть распределёнными устройствами — это сложная нетривиальная задача, которая напрямую определяет конечную стоимость аппаратного обеспечения. Она является ключевым показателем эффективности бизнеса — в особенности если таких устройств тысячи, десятки тысяч и более.

Существуют задачи, которые можно разделить (распределить) между совокупностью узлов, например обслуживание нагрузки на сайты в центрах обработки данных (ЦОД). За клиентом закрепляют определённый набор серверов (узлов), участвующих в его обслуживании. В рамках такого подхода в современных системах принято и рекомендовано использовать кластеры (разд. 9.1) или гиперконвергентные решения. Такие решения включают объединение ресурсов вычислений[10], хранения и сетевой инфраструктуры на одном уровне и поддерживают горизонтальное масштабирование (разд. 9.3.2). При этом вся работа, связанная с надёжностью, выполняется за счёт средств виртуализации и связанных средств резервирования.

В таких случаях обслуживание клиента осуществляется на закреплённом за ним узле без особого взаимодействия с окружающими узлами. Естественно, средства виртуализации «съедают» часть ресурса оборудования, однако при этом они обеспечивают высокую бизнес-эффективность.

[10] CPU, GPU-серверы.

Существуют задачи обработки данных, в которых одни изменения влекут за собой созависимые каскадные изменения. В задачах автоматизированных систем управления технологическими процессами (АСУ ТП) изменения одних переменных влияют на другие переменные и требуют мгновенной реакции (отклика) — например, перерасчёта переменных. В таких задачах не всегда можно использовать гиперконвергентные подходы, поэтому рекомендуется применять принцип «быть ближе к железу», исключив все лишние прослойки, приводящие к потере производительности и эффективности использования аппаратных ресурсов.

1.1. Немного о системах реального времени

ВНС, обеспечивающую выполнение проектных характеристик, таких как выполнение заданного числа операций в пределах заданного интервала времени, можно считать системой реального времени (СРВ[11]). Любую ВНС можно проектировать как СРВ или хотя бы как систему псевдореального времени. Как минимум, в ВНС стоит заложить некоторые присущие СРВ полезные архитектурные свойства.

СРВ совсем не обязательно должна обрабатывать высокие нагрузки, однако при решении современных задач большинство таких систем являются высоконагруженными. Например, умный датчик (устройство IIoT[12]) вполне может работать в режиме реального времени (РВ), но при этом не быть ВНС. Технологически такие компоненты обычно являются частью более крупных распределённых ВНС, о которых идёт речь в этой книге.

К СРВ относятся системы, в которых все компоненты должны работать максимально быстро и слаженно. Каналы связи должны обеспечивать высокую пропускную способность. Физические соединения и кабели должны быть произведены из лучших, эффективных по скорости прохождения импульсов и, следовательно, дорогостоящих материалов. В таких системах необходимо использовать быстродействующие микропроцессоры. В таких системах должно быть обеспечено многократное резервирование компонентов. Использование такого рода ресурсов оправдано при решении ряда задач — например, таких как управление органами регулирования ядерного реактора и остальными системами управления атомной электростанцией, органами управления летательными аппаратами, системами управления вооружением, медицинскими системами. Такие системы принято относить к СРВ,

[11] Real Time System, #01-01 [w].
[12] Industrial Internet of Things, #01-02 [w].


а в некоторых случаях — к системам жёсткого РВ, к компонентам которых предъявляют особенно высокие требования.

Для СРВ важнейшими понятиями являются задержка[13], представляющая собой время отклика, и дедлайн, представляющий собой предельную длительность выполнения операции. Поэтому СРВ и некоторые ВНС обычно называются низколатентными или ультранизколатентными (ULL[14]) системами (разд. 7.1.2).

В области знаний о СРВ и в их описаниях часто возникают путаница с терминами и споры об их использовании. СРВ классифицируют по степени строгости требований к величине задержек и дедлайнов. Выделяют следующие разновидности СРВ:

- жёсткого РВ (Hard Real-Time, HRT);
- умеренно жёсткого РВ (Firm Real-Time, FRT);
- мягкого РВ (Soft Real-Time, SRT);
- близкого к РВ (Near Real-Time, NRT);
- псевдо-РВ (Pseudo-Real-Time, PRT).

Во многих системах допустимо ослабление требований к режимам РВ. Для более точной классификации таких систем введено понятие джиттера[15]. Джиттер, определяющий разброс значений времени отклика, может увеличиваться в связи с различными обстоятельствами, например проблемами в сети, пиковыми запросами или иными перегрузками (разд. 7.1.4). Выделяют несколько методик вычисления джиттера:

- абсолютный — максимальное отклонение от заданного значения;
- пиковый (P2P[16] jitter) — отклонение между максимальной и минимальной задержкой;
- средний (average jitter) — среднее отклонение в пределах интервала;
- среднеквадратичный (RMS[17] jitter) — в пределах интервала.

В системах, работающих в режиме HRT, недопустимо нарушение дедлайнов, которое гарантированно приведёт к отказам (разд. 7.2). В таких системах джиттер должен быть минимальным и предсказуемым. Обычно, когда заявляют реализацию HRT, подразумевают аппаратное исполнение или исполнение в виде аппаратно-программного комплекса (АПК), в котором

[13] Latency, #01-03. ▸
[14] Low latency, ultra-low latency, #01-04.
[15] Jitter, #01-05 [w].
[16] Peak To Peak Jitter.
[17] Root Mean Square Jitter.



ответственность за реализацию режима РВ в основном возложена на железо. Такой режим РВ актуален для критических систем, таких как авионика, системы противоаварийной защиты (ПАЗ) АСУ ТП ядерных реакторов, электроэнергетических установок, исследовательских и медицинских устройств.

В системах с FRT допускаются редкие нарушения дедлайнов, что не приводит к отказу. Результат вычислений, полученный с нарушением дедлайнов, может считаться бесполезным. Джиттер в таких системах должен быть предельно низким. Этот режим РВ актуален для АСУ ТП нормальных режимов эксплуатации большинства критических систем, а также финансовых сервисов.

В системах с SRT допустимы нарушение дедлайнов и умеренно контролируемый джиттер. Такой режим РВ актуален для систем оперативного управления, видеозвонков, онлайн-игр.

В системах с NRT строгие требования к дедлайнам отсутствуют, допустимы значительные отклонения задержки. В таком режиме функционируют системы, нетребовательные к оперативной реакции, — например, телеметрические сервисы, системы бизнес-аналитики, допускающие значительные задержки получения информации.

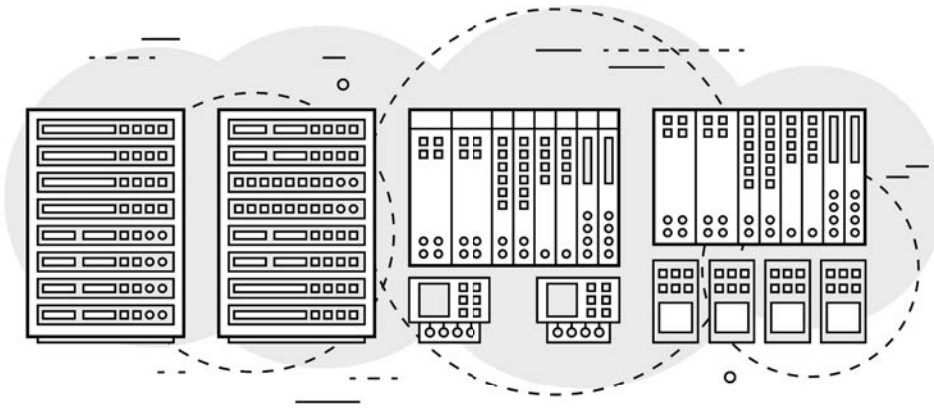
В системах с PRT не задаются требования к дедлайнам, допустимы высокие джиттеры. В таком режиме функционируют системы, не требующие интерактивного обновления данных, — например, системы бизнес-аналитики, отчётности.

Чем выше требования к режимам РВ, тем больше ответственность при проектировании, верификации и валидации решений. Чем ниже требования к режиму РВ, тем разработка таких систем проще и дешевле. Иногда лучше отказаться от требований к режиму РВ, даже если таковые подразумеваются. На практике такие системы создавать проще, чем системы с соответствующими требованиями. Обычно проектируют некую систему, оценивают её возможности, а затем система масштабируется и оптимизируется в узких местах. Такой подход является менее ресурсоёмким, чем создание СРВ с высокими требованиями к режимам РВ.

Системы без требований РВ — это системы, в которые не закладывают требования к режимам работы РВ. Тем не менее такие системы могут быть эффективно и качественно разработаны, они могут обладать исключительными характеристиками производительности. Однако если в такую систему заранее не заложить требования режимов работы РВ, то при высоких нагрузках и перегрузках такая система с высокой вероятностью поведёт себя

непредсказуемо. Не всегда будет сразу понятно, что с ней не так, где именно возникли проблемы.

Самая большая проблема для принимающих решения лиц возникает в тот момент, когда появляется путаница в терминах. Разработчики навязывают систему, критичную к быстродействию, но реальная задача позволяет обойтись и не очень критичной. В таком случае возникает перерасход на разработку, закупку оборудования, пусконаладку, начинаются проблемы, связанные с обслуживанием серьёзной, критичной к быстродействию системы. А такая дорогостоящая и «капризная» система нужна далеко не всегда.



Глава 2. Аппаратная архитектура

Аппаратная инфраструктура является важнейшей составляющей высоконагруженных распределённых систем. В этой книге не будут подробно детализированы, но будут отмечены некоторые моменты, которые в итоге влияют на надёжность, эффективность и стоимость инфраструктуры целевой системы. При проектировании аппаратной инфраструктуры для распределённых систем следует учитывать несколько факторов, в том числе:

- выбор типа оборудования;
- выбор места размещения оборудования;
- оценку конфигурации при нагрузке;
- оценку зависимости от производителей и технологий;
- планирование масштабирования.

2.1. Выбор типа оборудования

При проектировании рассматриваемых систем всегда возникает вопрос выбора типа используемого оборудования: использовать меньше серверов, но крутых, мощных и дорогих с резервированием компонентов, или больше, но дешёвых и слабых без резервирования компонентов? На этот вопрос можно ответить, рассмотрев задачу как с точки зрения аппаратного обеспечения (АО), так и в совокупности с программным обеспечением (ПО), потому что именно оно влияет на итоговую эффективность работы оборудования.

При выборе между «круто и дорого» и «завалить кучей дешёвого железа» существует оптимальный алгоритм действий: необходимо тщательно оценить технические требования, географию, условия размещения оборудования, доступность технического персонала на местном рынке труда и т. д. В абсолютном большинстве случаев этот выбор является надуманным. Если пройти все этапы проектирования системы, то либо многообразие вариантов сократится до одного безальтернативного, после чего останется только провести тендер на поставку, либо существенно сократится число вариантов выбора оборудования.

2.2. Место размещения оборудования

Место размещения оборудования почти всегда обусловлено правильно сформулированными бизнес- и техническими требованиями при проектировании. География распределённых кластеров напрямую зависит от показателей RPO/RTO (разд. 7.2.2) и требований по защите от катастроф (разд. 7.2.3). Выбор между коммерческим ЦОД и локальным размещением на площадке предприятия вытекает из требований к удалённости, безопасности, защищённости и резервированию каналов связи.

Проектирование обычно осуществляется сверху вниз, путём декомпозиции от общего к частному. Важно своевременно учитывать возможности самого нижнего уровня, чтобы компенсировать его недостатки на верхнем. При отсутствии подходящих аппаратных решений и/или площадок для размещения оборудования можно снизить требования к нижнему уровню (площадке) за счёт использования эффективной программной архитектуры и реализации общего уровня защищённости на более высоких уровнях. Важно помнить, что требования архитектуры целевой системы — это способ реализации основной бизнес-задачи, а не истина в последней инстанции. Всегда существует несколько способов реализации таких требований.

2.3. Оценка конфигурации оборудования при нагрузке

Необходима определённая аккуратность в оценке комплектации оборудования. Например, при максимальной нагрузке на сетевые устройства может оказаться, что процессоры серверов остаются недостаточно загруженными. В таких случаях необходимо оптимизировать соответствующие аппаратные ресурсы, увеличить мощность сетевого оборудования или выбрать менее мощные процессоры, при этом даже немного сэкономив.

Требования к конфигурации оборудования должны исходить из следующих целевых показателей:

- производительность и загрузка при установленной нагрузке;
- производительность и загрузка при пиковой нагрузке;
- уровень резервирования;
- размеры доменов единичного и двойного отказа;
- стандартизация конфигураций;
- доступность оборудования и конфигурации (комплектующих) на рынке, планы по поддержке производителем (вендором);
- доступность квалифицированных кадров на локальном рынке труда.

Зачастую имеет смысл разместить достаточно мощное оборудование с большим запасом по производительности и резервированию на удалённых площадках для максимизации времени безостановочной работы без обслуживания при возникшем сбое. При наличии надёжной логистики и большого склада в крупном центре уровень резервирования запасных частей, инструментов и принадлежностей (ЗИП) можно снизить, ориентируясь на статистику отказов и обеспечивая своевременное пополнение.

2.4. Зависимость от вендора или технологии

Необходимо оценить риски зависимости от одного вендора или конкретной технологии. Всегда лучше рассчитывать на то, что и вендора, и технологию в какой-то момент придётся заменить. Или сам вендор может выбрать другую технологию для своего будущего оборудования, и ваши системы станут несовместимы. Например, бывает так, что программная инфраструктура использует особую аппаратную технологию. В этой связи можно вспомнить классический пример того, как компания DEC перестала поддерживать серверы AlphaServer с архитектурой RISC в начале 2000-х гг. Такие решения продолжают использовать и сейчас на множестве предприятий, и у многих из них не осталось резервов (ЗИП). Системы такого рода продолжают эксплуатировать, потому что разработчики ПО, занимавшиеся ныне устаревшей архитектурой, уже на пенсии или недоступны, поэтому портировать системы на современную архитектуру некому. Модернизировать такие системы можно только целиком. Представьте себе, например, что это АЭС с несколькими энергоблоками. Такие ситуации встречаются весьма часто во многих отраслях и странах мира. Возникает вопрос организации технической и гарантийной поддержки для большого разнообразия оборудования и решений на основе продукции одного вендора.

Крупные сервисные провайдеры и интеграторы самостоятельно осуществляют сборку оптимальных конфигураций оборудования для конкретных инфраструктурных решений, что довольно эффективно. Например, Google, Amazon и Яндекс самостоятельно разрабатывают серверные и инфраструктурные решения, контролируя их архитектуру, производство и жизненный цикл.

2.5. Планирование масштабирования

Необходимо заранее планировать сценарий масштабирования ресурсов (разд. 9.3). Следует планировать возможность увеличения мощности серверов за счёт добавления в них оперативной памяти, сетевых карт, процессорных и GPU-модулей в случае модульных серверов. Кроме того, следует планировать увеличение числа серверов. Бывают ситуации, когда программная инфраструктура рассчитана на предельное число узлов и просто не может работать при большем их числе. О таких ограничениях лучше знать заранее. Может случиться так, что условия лицензии на дополнительные узлы не устроят потребителя. Кроме того, на этапе проектирования стоит рассмотреть возможность применения решений для разгрузки[18] CPU[19].

На этапе проектирования стоит рассмотреть возможность применения специализированных аппаратных ускорителей для решения специализированных задач. Среди них следует отметить:

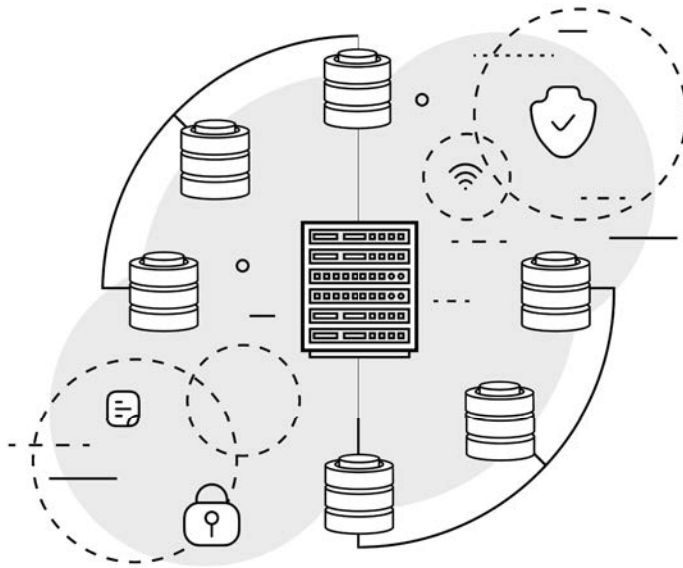
- DPU[20]: для разгрузки CPU от задач инфраструктуры — виртуализации, обработки сетевого стека (SmartNIC), управления СХД и безопасности;
- GPU[21]: для задач, требующих массового параллелизма, — искусственный интеллект, высокопроизводительные вычисления (HPC), машинное обучение, графика;
- NPU[22]: для аппаратного ускорения инференса нейросетей при низком энергопотреблении;
- LPU[23]: для специализированных задач машинного обучения;

[18] Offloading, #02-01 [w].
[19] Central Processor Unit.
[20] Data Processing Unit, #02-02.
[21] Graphics Processing Unit, #02-03 [w].
[22] Neural Processing Unit, #02-04 [w].
[23] Language Processing Unit, #02-05.

- FPGA[24] (программируемая пользователем вентильная матрица): для задач, требующих детерминированного выполнения логики с минимальными и предсказуемыми задержками за счёт реализации логики непосредственно в аппаратуре без программного стека и джиттера планировщика ОС, широко применяется в телекоммуникационных и автоматизированных системах управления;
- ASIC[25] (интегральная схема специального назначения[26]): для решения узкоспециализированных задач с максимальной производительностью и энергоэффективностью.

Важно отметить, что термин NPU неоднозначен. В контексте сетевых устройств он исторически обозначал Network Processing Unit[27] — функции которого сегодня выполняют DPU и SmartNIC. В контексте 2026 г. NPU принято относить к Neural Processing Unit.

[24] Field-Programmable Gate Array, #02-06 [w].
[25] Application-Specific Integrated Circuit.
[26] ИССН, #02-07 [w].
[27] Network Processing Unit, #02-08 [w].



Глава 3. Принципы распределения данных

Современные информационные системы оперируют колоссальными объёмами данных. Эти данные генерируются в режиме реального времени миллионами распределённых источников. Такими источниками могут быть финансовые транзакции, телеметрия IoT-устройств, потоковое видео или контент, связанный с обслуживанием социальных сетей и мессенджеров. Обработка данных такого рода предъявляет принципиально новые требования к архитектуре вычислительных комплексов. Монолитные централизованные системы уже не справляются с обеспечением необходимой производительности, отказоустойчивости и масштабируемости.

На смену приходят распределённые системы — сложные программно-аппаратные комплексы, состоящие из множества независимых вычислительных узлов, скоординированно решающих общую задачу. Ключевым преимуществом таких систем является их способность к горизонтальному масштабированию. При росте нагрузки в кластер добавляются новые узлы, что позволяет практически линейно наращивать вычислительную мощность и пропускную способность.

Для обеспечения эффективности функционирования распределённых систем прежде всего необходимо определиться с принципами разделения данных между распределёнными узлами.

3.1. Данные и метаданные

Основная нагрузка в распределённых системах связана с обработкой большого объёма (потока) сообщений (данных, изменений, событий) из разнообразных внешних источников и внутренних составных компонентов системы (разд. 4.7). Принято выделять следующие типы событий:

- асинхронные — полностью непредсказуемые события, такие как оповещения, тревоги, команды управления;
- синхронные — предсказуемые события, случающиеся с определённой частотой, такие как сообщения, связанные с обменом данными и их синхронизацией (разд. 4.4.1);
- изохронные — разновидность синхронных событий, происходящих с гарантированной периодичностью и минимальным джиттером, например диагностические сообщения о состоянии системы (разд. 7.1.3).

Часто меняющиеся данные и возникающие события можно отнести к оперативным данным, или данным реального времени. Не все данные в системе являются оперативными. Часть данных необходима для описания оперативных данных, и они могут подвергаться изменениям крайне редко. Такие вспомогательные данные называются данными о данных или метаданными. Например, фотоизображения могут содержать дату, координаты, выдержку в качестве метаданных.

Метаданные могут содержать разную полезную, но не относящуюся к оперативному взаимодействию узлов информацию. Следовательно, метаданные можно отнести к неоперативным данным.

В распределённых системах следует тщательно разделять способы обмена оперативными и неоперативными данными. Объём неоперативных данных может быть весьма заметен, и если их включать в трафик обмена оперативными данными, они могут занять существенный его объём, а также значительно повлиять на нагрузку целевой системы.

Оптимизация баланса обмена данными является весьма непростой задачей. Изменение метаданных может повлиять на системное восприятие оперативных данных. Следовательно, все изменения метаданных следует оперативно доставить на требуемые узлы. При обмене метаданными следует учитывать их приоритет.

В целевой системе необходимо определить, какие изменения оперативных и неоперативных данных считать значимыми или бесполезными. Тогда управлять данными этой системы будет значительно проще.