

1. Введение

Сухорослов Олег Викторович

07.09.2026

План

- О курсе
- Распределенные системы
 - Определение
 - Области применения и примеры
 - Требования и свойства
 - Отличия и особенности
 - Типовые задачи

О курсе

Вводный курс по распределенным системам (РС)

- базовые понятия и модели РС
- разновидности РС, их особенности, предъявляемые требования
- принципы реализации РС, типовые задачи и способы их решения
- важные теоретические результаты и алгоритмы
- технологии и практические навыки разработки РС

Темы

Модуль 1

- Взаимодействие между процессами, надежная передача и RPC
- Протокол HTTP, REST и веб-сервисы
- Групповые взаимодействия, рассылка и распространение информации
- Непрямое взаимодействие, очередь сообщений, publish-subscribe
- Отказы, их обнаружение, group membership
- Именованное пространство имен и поиск

Модуль 2

- Масштабирование, балансировка нагрузки и шардинг
- Репликация данных
- Время, часы и порядок событий
- Консенсус и связанные задачи
- Параллельная обработка
- Безопасность в распределенных системах
- Устойчивость к произвольным отказам

Организация

- Занятия
 - Онлайн, 1-2 общих семинара на неделе
- Домашние задания
 - 10 заданий со сроком 1-2 недели, защита решений
- Экзамены
 - (К) Коллоквиум в середине курса
 - (Э) Итоговый в конце курса
- Оценка: $0.6 \text{ ДЗ} + 0.2 \text{ К} + 0.2 \text{ Э}$
 - Возможен автомат, если $\text{ДЗ} \geq 8$ и $\text{К} \geq 8$
- Репозиторий: <https://distsys.ru/course/hse-2026>

Распределенная система

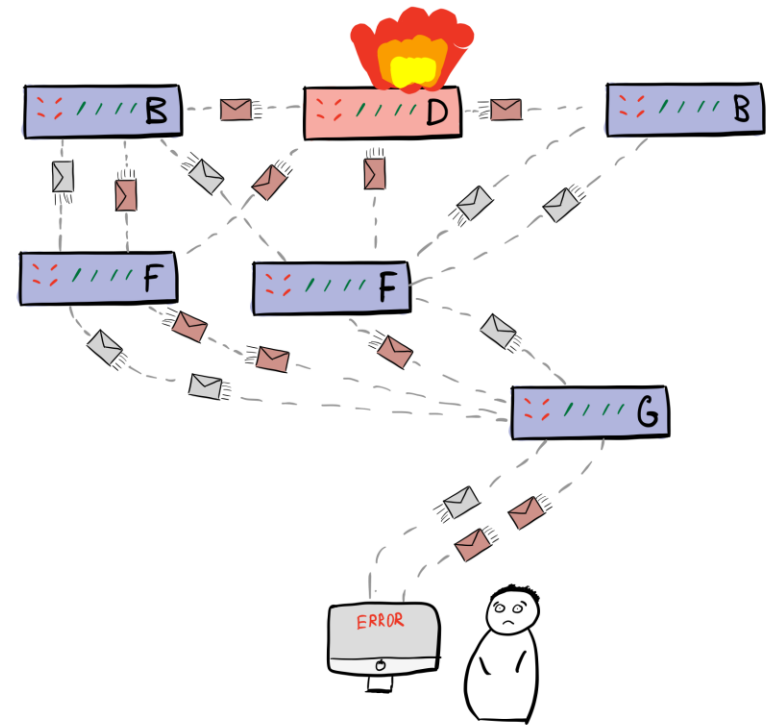
Что это?

Определение 1

“A distributed system is one in which the **failure of a computer you didn't even know existed** can render your own computer unusable.”

Leslie Lamport (1987)

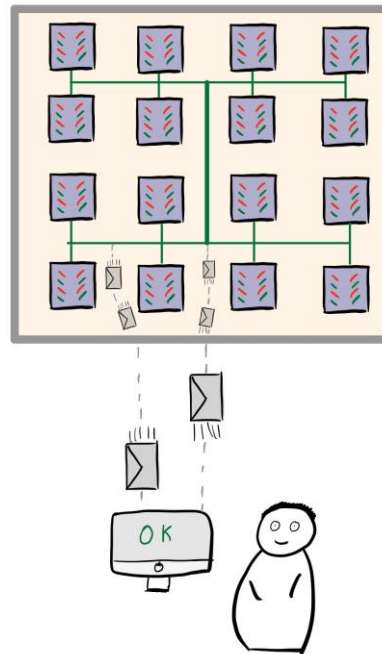
<https://www.microsoft.com/en-us/research/publication/distribution/>



Определение 2

“A distributed system is a **collection of autonomous computing elements** that appears to its users as a **single coherent system**.”

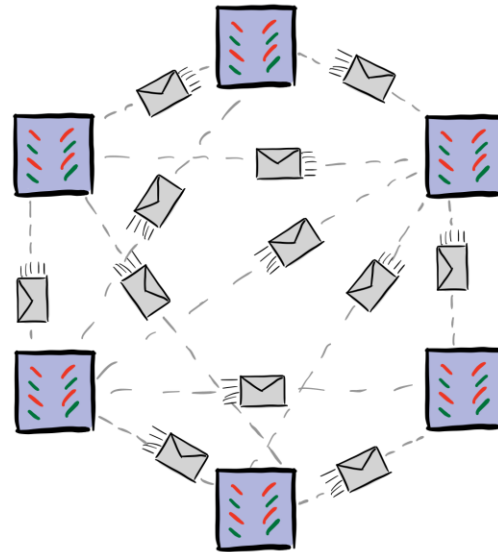
van Steen, Tanenbaum. Distributed Systems: Principles and Paradigms



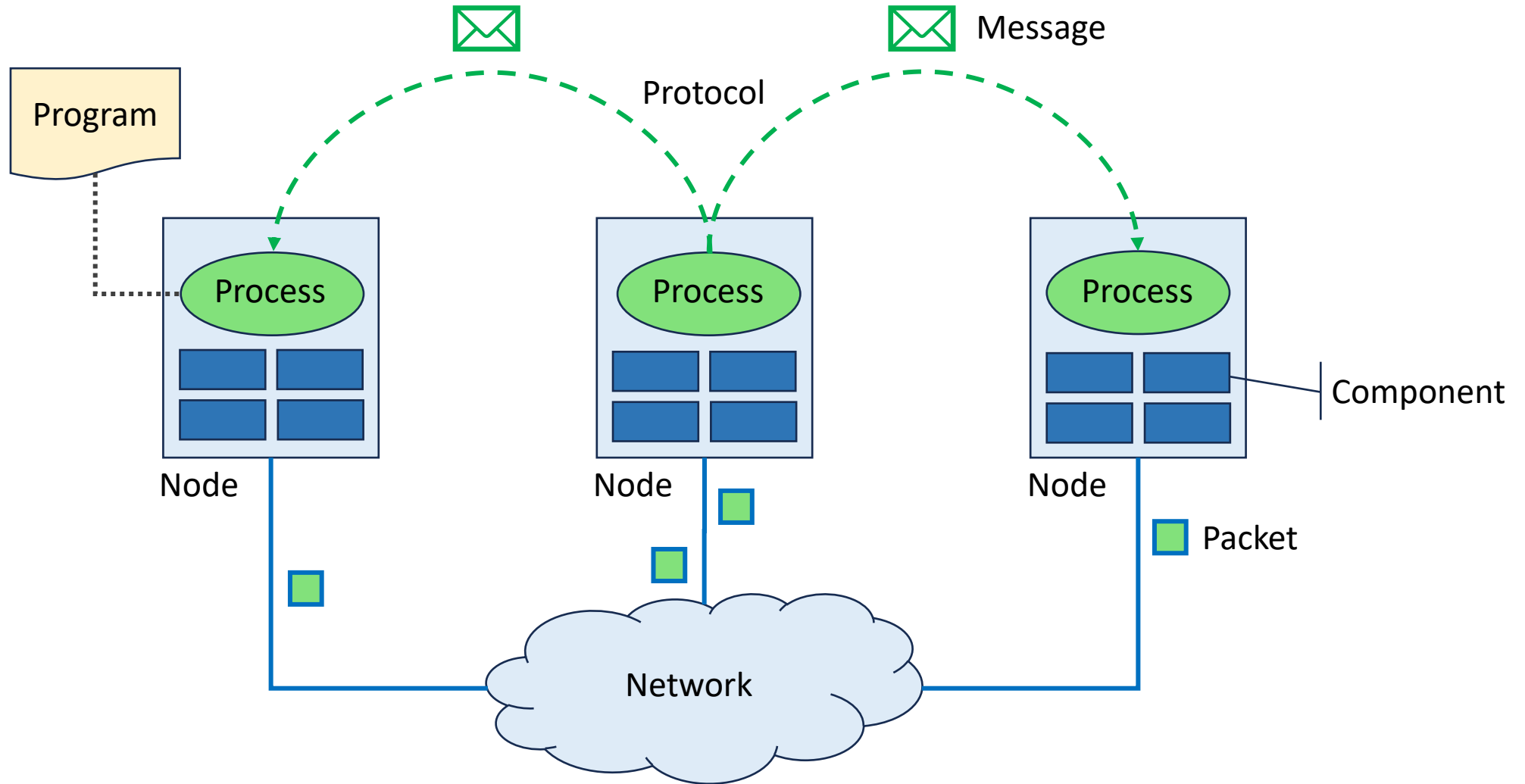
Определение 3

“We define a distributed system as one in which **hardware or software components** located at **networked computers** communicate and coordinate their actions **only by passing messages.**”

Coulouris et al. Distributed Systems: Concepts and Design.



Определение 4 (базовые понятия)



Определение 4 (базовые понятия)

- A **program** is the code you write
- A **process** is what you get when you run it
- A **message** is used to communicate between processes
- A **packet** is a fragment of a message that might travel on a wire
- A **protocol** is a formal description of message formats and the rules that processes must follow in order to exchange those messages
- A **node** is a computer where a process is running
- A **network** is the infrastructure that links nodes, and consists of routers which are connected by communication links
- A **component** can be a process or any piece of hardware required to run a process, support communications between processes, store data, etc.

Определение 4

“A distributed system is an **application** that executes a collection of **protocols** to **coordinate the actions** of multiple **processes** on a **network**, such that all **components** **cooperate together** to perform a single or small set of related **tasks**.”

[Introduction to Distributed System Design](#), Google

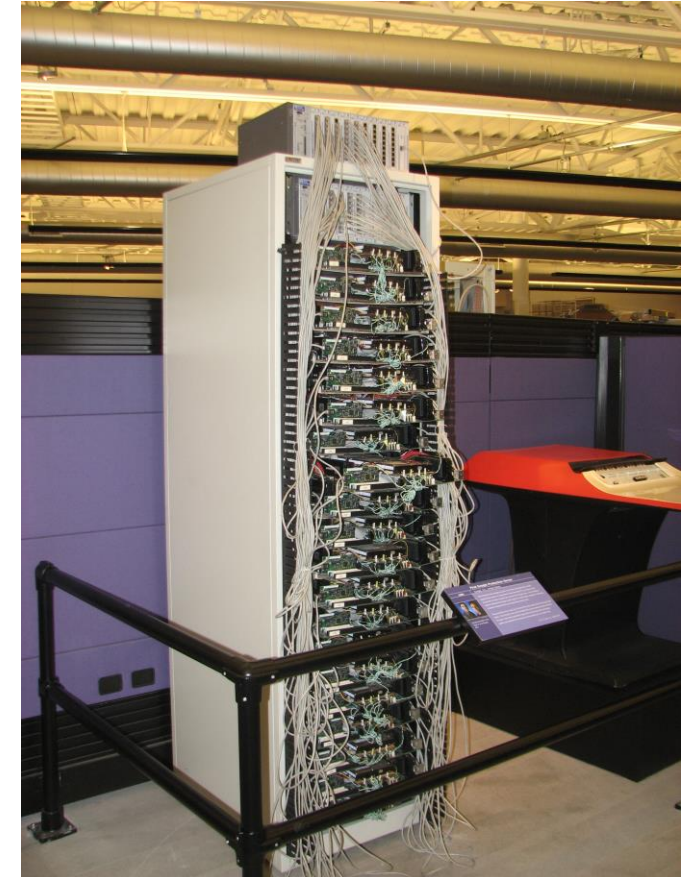
Распределенная система

- Аппаратный уровень: набор автономных узлов, связанных сетью
 - Функционируют независимо, нет привычных разделяемых ресурсов (часы, память)
 - Могут быть географически распределены, иметь отличающиеся характеристики
- Программный уровень: набор выполняющихся на узлах процессов, взаимодействующих с помощью сообщений
 - Каждый процесс имеет собственное локальное состояние
 - Процессы не имеют прямого доступа к состояниям других процессов
 - Передача сообщения требует времени и может завершиться неудачей
- Часть системы может отказать, пока другая продолжает работать
 - Отказы сети, узлов, процессов...

Зачем нужны такие системы?

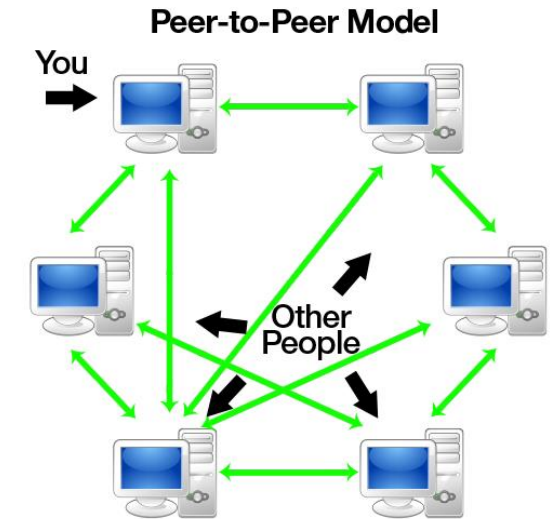
Применение распределенных систем

- Увеличение производительности
 - Решение сложных вычислительных задач
 - Обслуживание большого количества клиентов
 - Хранение и обработка больших объемов данных
- Доступность и отказоустойчивость
 - Устойчивость к отказам за счет избыточности



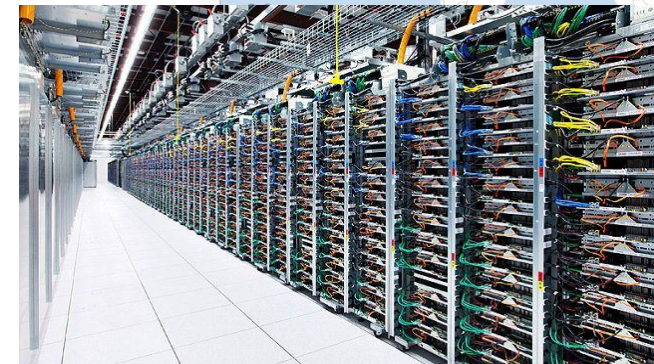
Применение распределенных систем

- Совместное использование ресурсов
 - Клиент-сервер, peer-to-peer, вычислительный кластер
 - Поддерживать единую систему дешевле, чем множество независимых
- Коммуникация и координация
 - Пользователи и узлы географически распределены
- Уменьшение задержки при обслуживании географически распределенных пользователей
 - Размещение данных как можно ближе к пользователям



Современные системы

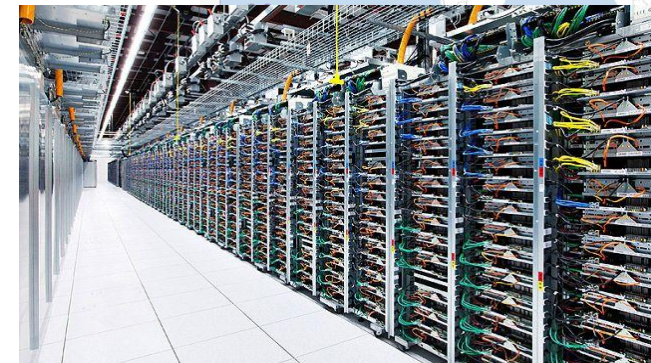
- Email, обмен сообщениями
- Интернет-банк
- Веб-поиск
- Онлайн-редактор документов
- Социальная сеть
- Хранилище данных
- Облако
- Сеть доставки контента, онлайн-кинотеатр
- Файлообменная сеть, криптовалюта
- Доступ к ИИ-моделям, их обучение



Современные системы

- Высокопроизводительные
- Высоконагруженные
- Высокодоступные
- Масштабируемые
- Децентрализованные
- Гибкие

== Распределенные

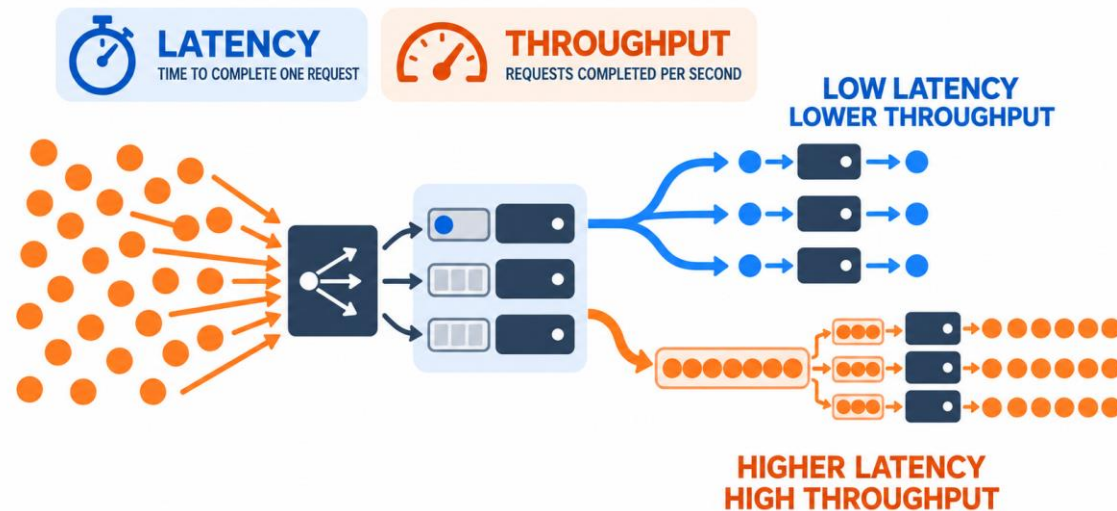
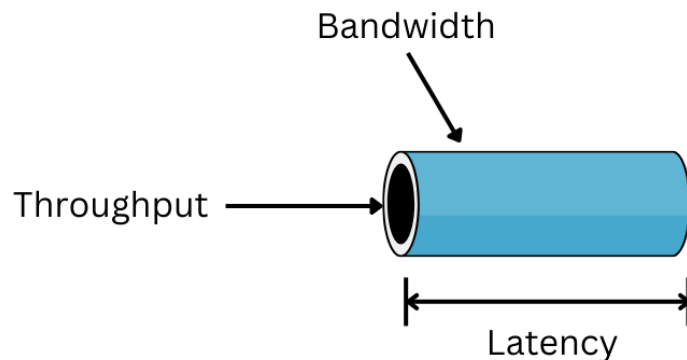


Нефункциональные требования

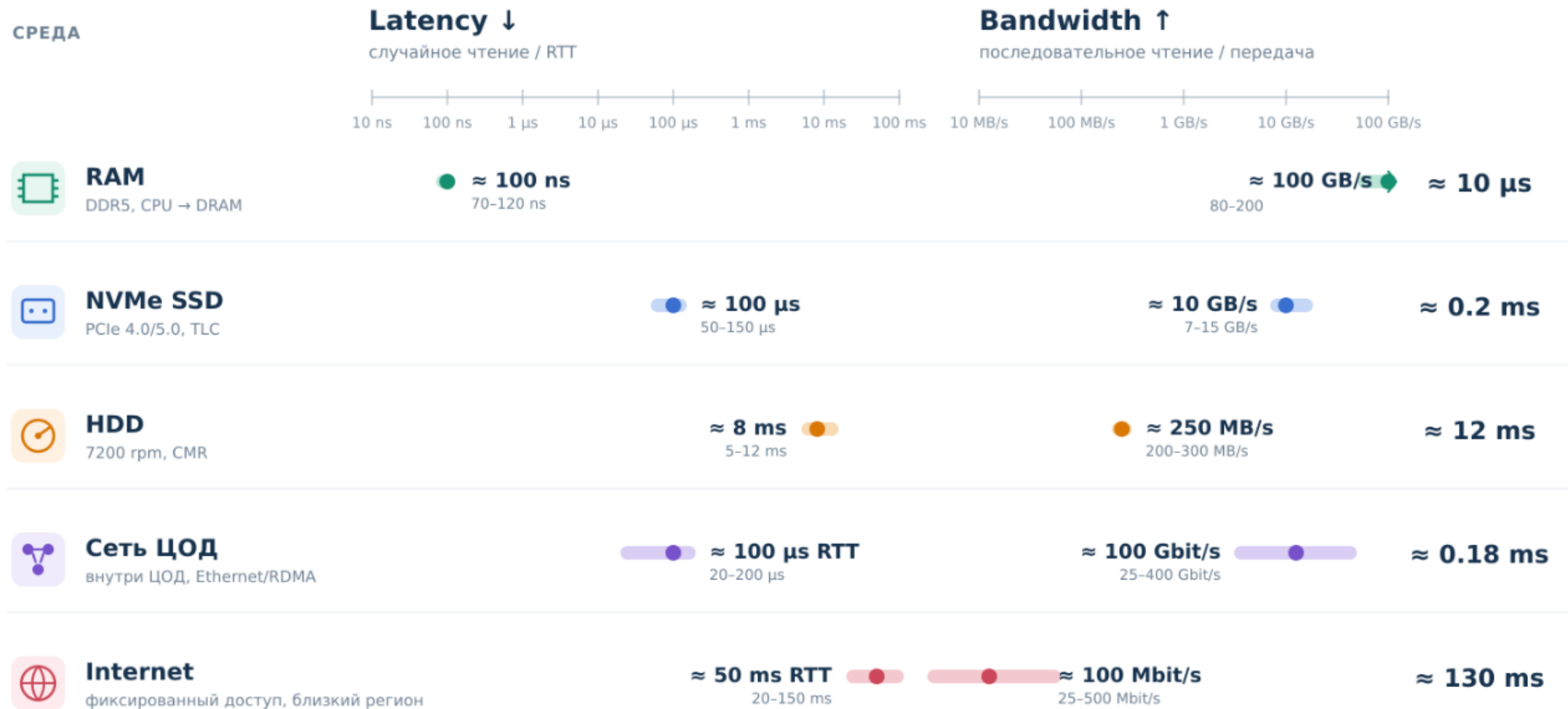
- Производительность
- Масштабируемость
- Надежность
- Доступность
- Отказоустойчивость
- Безопасность
- Согласованность
- Удобство сопровождения

Производительность (Performance)

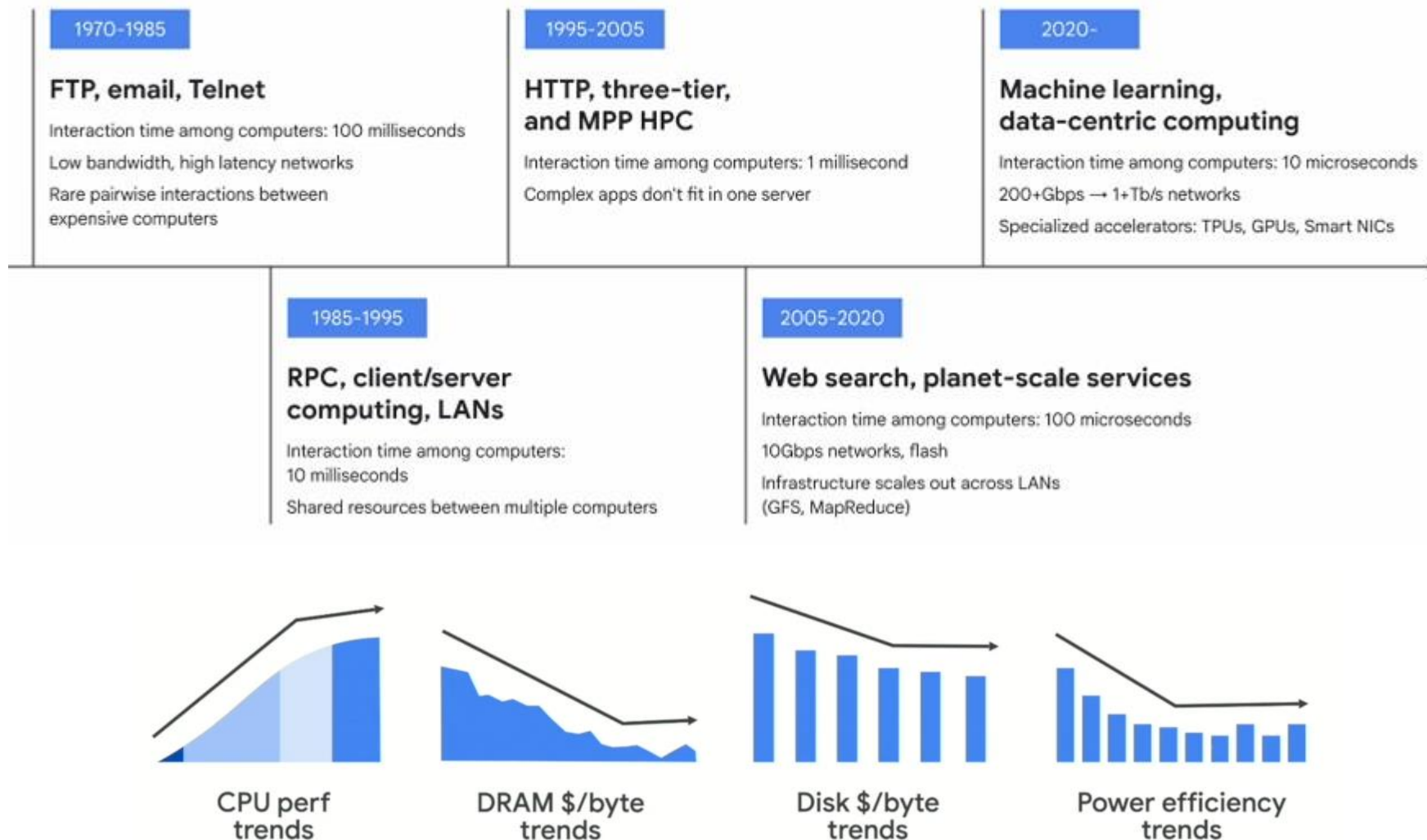
- Показатели производительности
 - **Задержка (latency)**: время обработки запроса, время ожидания ответа
 - **Пропускная способность (throughput)**: обрабатываемые запросов/данные в секунду
- У отдельных компонентов системы могут быть свои показатели
- Разные показатели могут конфликтовать друг с другом



Ориентиры производительности



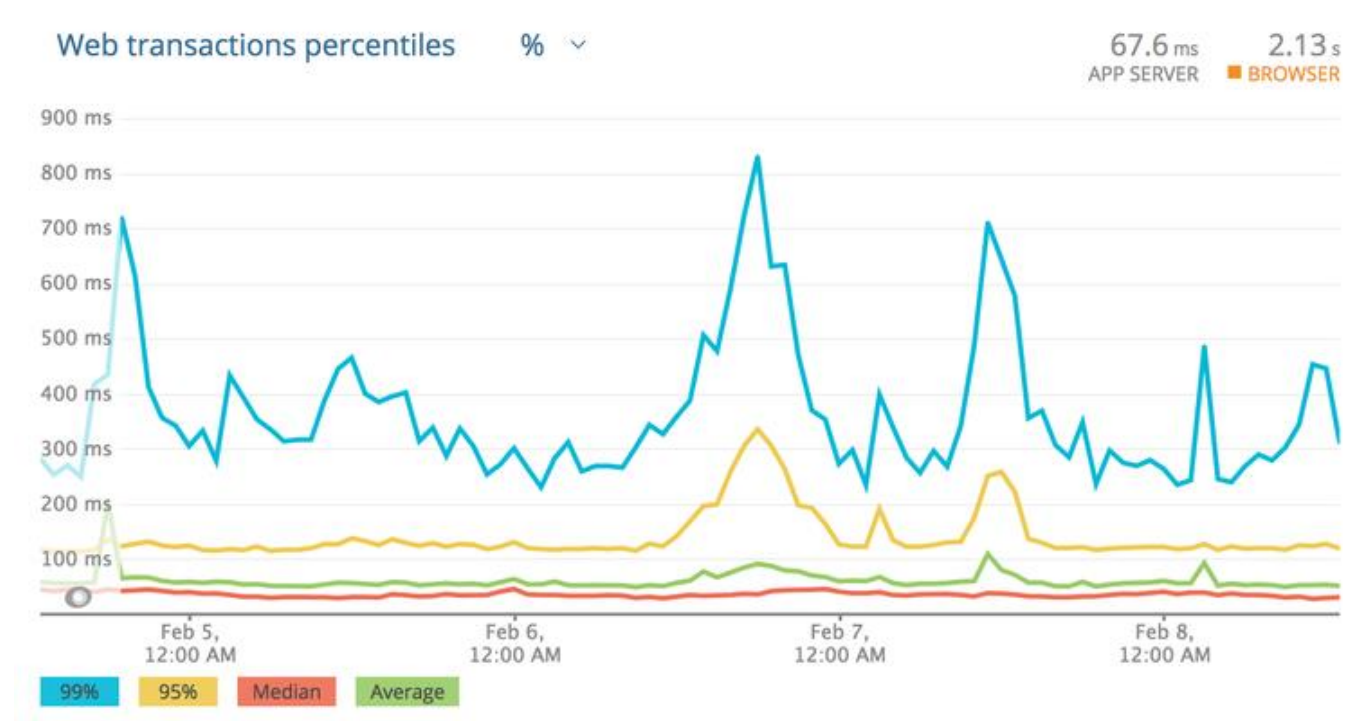
Five epochs of distributed computing



<https://cloud.google.com/blog/topics/systems/the-fifth-epoch-of-distributed-computing>
<https://www.nextplatform.com/2025/08/21/google-is-already-using-the-future-ai-network-you-might-get-in-2028/>

Измерение производительности

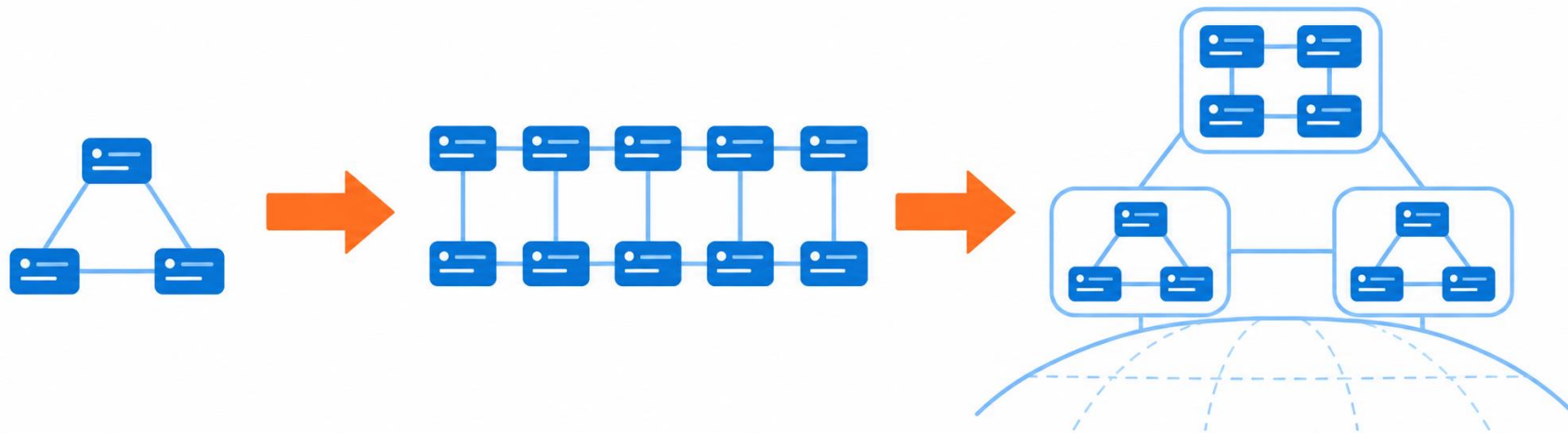
- Средних значений недостаточно, важны также перцентили
- Производительность должна лежать в допустимом интервале



Масштабируемость (Scalability)

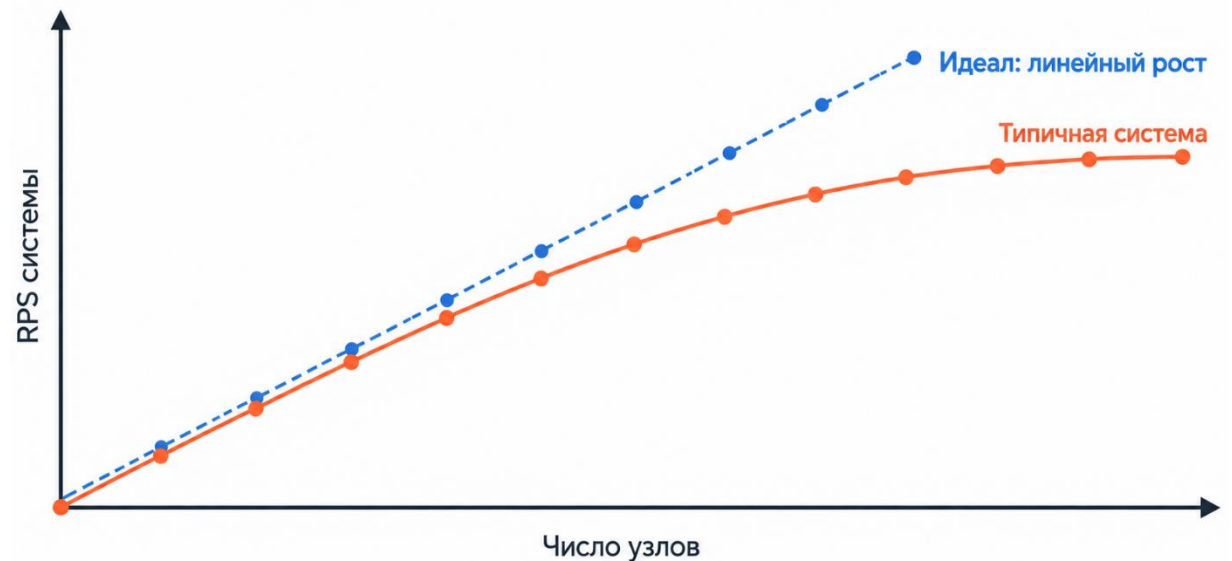
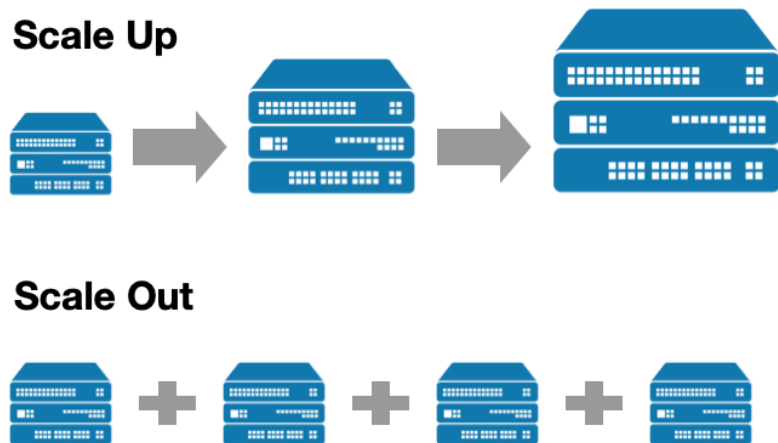
Способность системы "расти" в некотором измерении без потери производительности и других характеристик, а также без необходимости изменять реализацию

- Возможные измерения: число узлов, пользователей, запросов, организаций, территория
- Разновидности: нагрузочная, географическая, административная



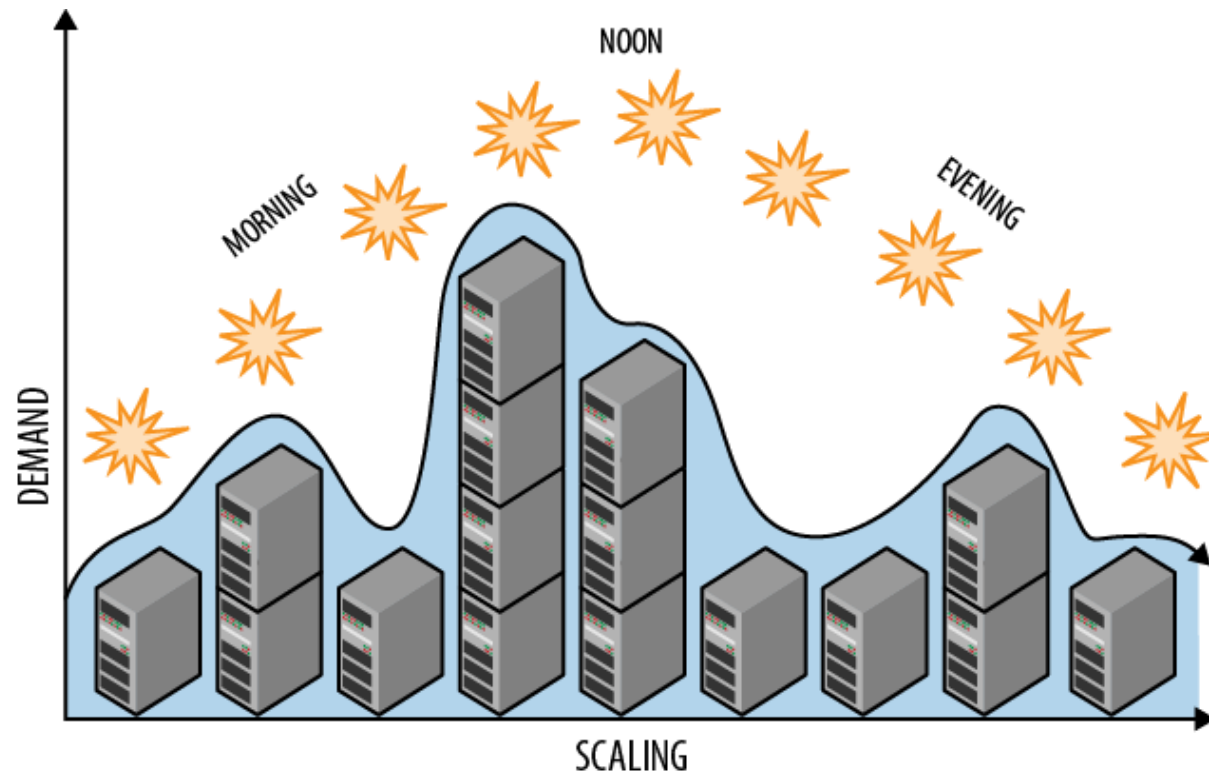
Нагрузочная масштабируемость

- Способность системы увеличивать свою производительность при увеличении нагрузки путем замены или добавления аппаратных средств
- Подходы
 - вертикальное масштабирование (scale up)
 - горизонтальное масштабирование (scale out)



Эластичность

Автоматическое масштабирование ресурсов под текущую нагрузку



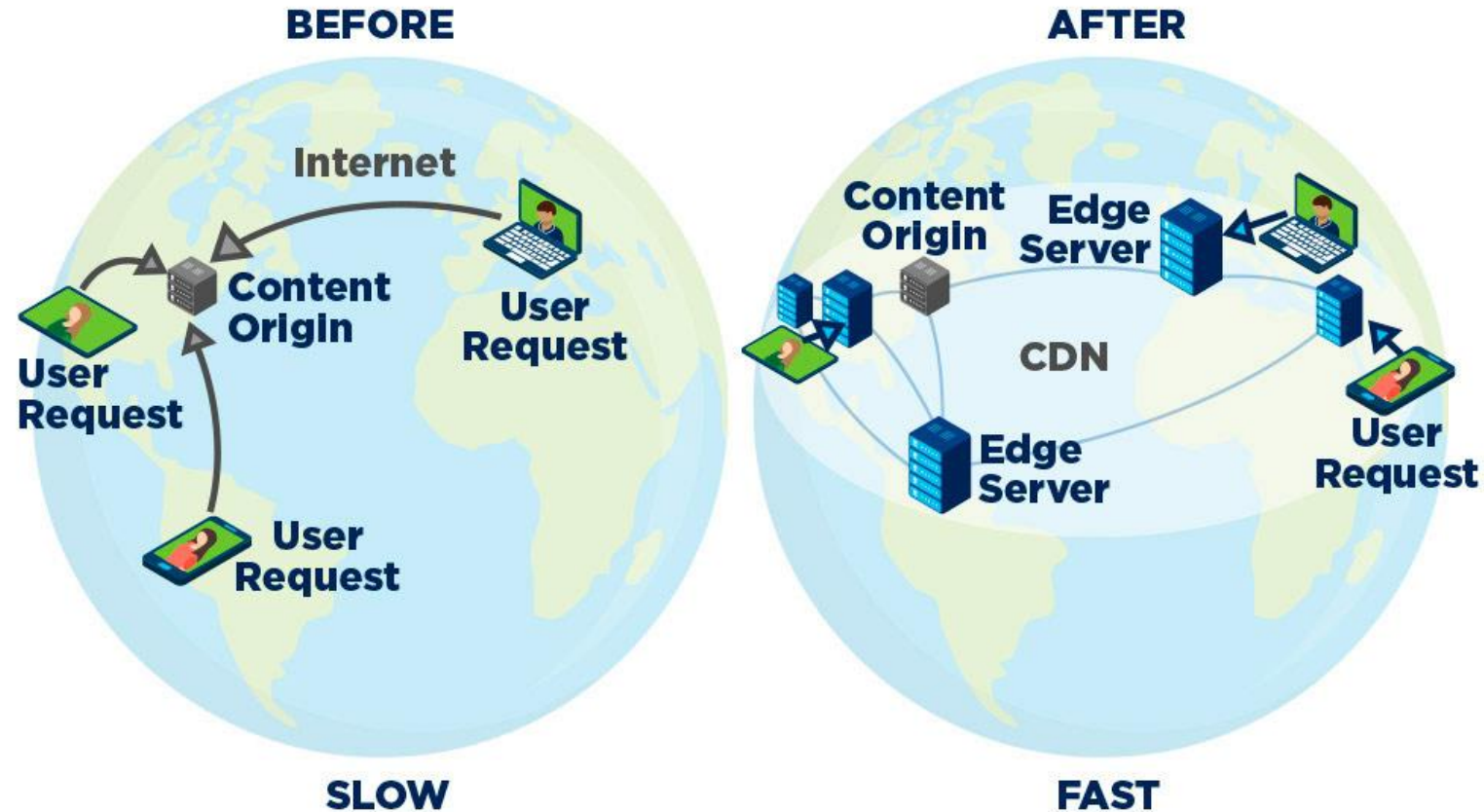
Географическая масштабируемость

Способность системы сохранять требуемые характеристики (например, производительность) при территориальном разнесении ее компонентов

		Barcelona ✖	Paris ✖	Tokyo ✖	Toronto ✖	Washington ✖
Amsterdam ✖		🟢 32.261ms	🟢 10.548ms	🔴 247.928ms	🟡 92.134ms	🟢 83.94ms
Auckland ✖		🔴 260.975ms	🔴 275.071ms	🟡 198.422ms	🟡 187.308ms	🟡 214.29ms
Copenhagen ✖		🟢 39.605ms	🟢 23.809ms	🟡 233.871ms	🟡 107.62ms	🟡 103.205ms
Dallas ✖		🟡 133.646ms	🟡 113.981ms	🟡 146.888ms	🟢 45.877ms	🟢 37.964ms
Frankfurt ✖		🟢 24.933ms	🟢 10.728ms	🟡 221.853ms	🟡 94.884ms	🟡 97.392ms
London ✖		🟢 29.842ms	🟢 8.224ms	🟡 218.565ms	🟡 92.374ms	🟢 78ms
Los Angeles ✖		🟡 157.134ms	🟡 144.735ms	🟡 114.896ms	🟢 78.137ms	🟢 63.333ms
Moscow ✖		🟢 67.861ms	🟢 50.043ms	🔴 278.057ms	🟡 132.542ms	🟡 137.487ms
New York ✖		🟡 106.043ms	🟢 73.134ms	🟡 176.005ms	🟢 21.74ms	🟢 8.432ms
Paris ✖		🟢 22.62ms	—	🟡 234.953ms	🟡 91.149ms	🟢 82.149ms
Stockholm ✖		🟢 54.971ms	🟢 32.158ms	🔴 246.37ms	🟡 112.309ms	🟡 108.417ms
Tokyo ✖		🔴 279.284ms	🟡 234.991ms	—	🟡 178.581ms	🟡 170.332ms

<https://wondernetwork.com/pings>

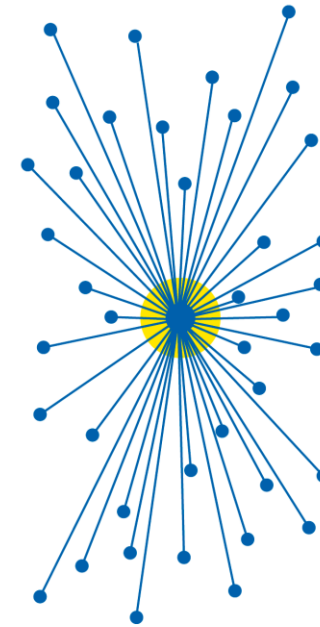
Content Delivery Network (CDN)



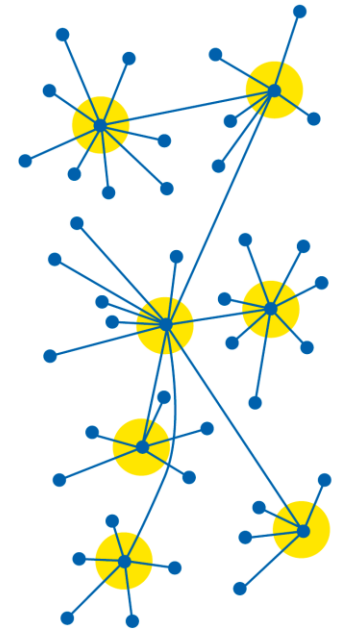
<https://www.limelight.com/resources/white-paper/5-things-multi-cdn-strategy/>

Административная масштабируемость

- Возможность системы функционировать на базе произвольного количества независимых владельцев, обслуживающих части системы и предоставляющих ресурсы в рамках системы
- Примеры: peer-to-peer, файлообменные сети, Биткойн, IPFS, грид-инфраструктуры



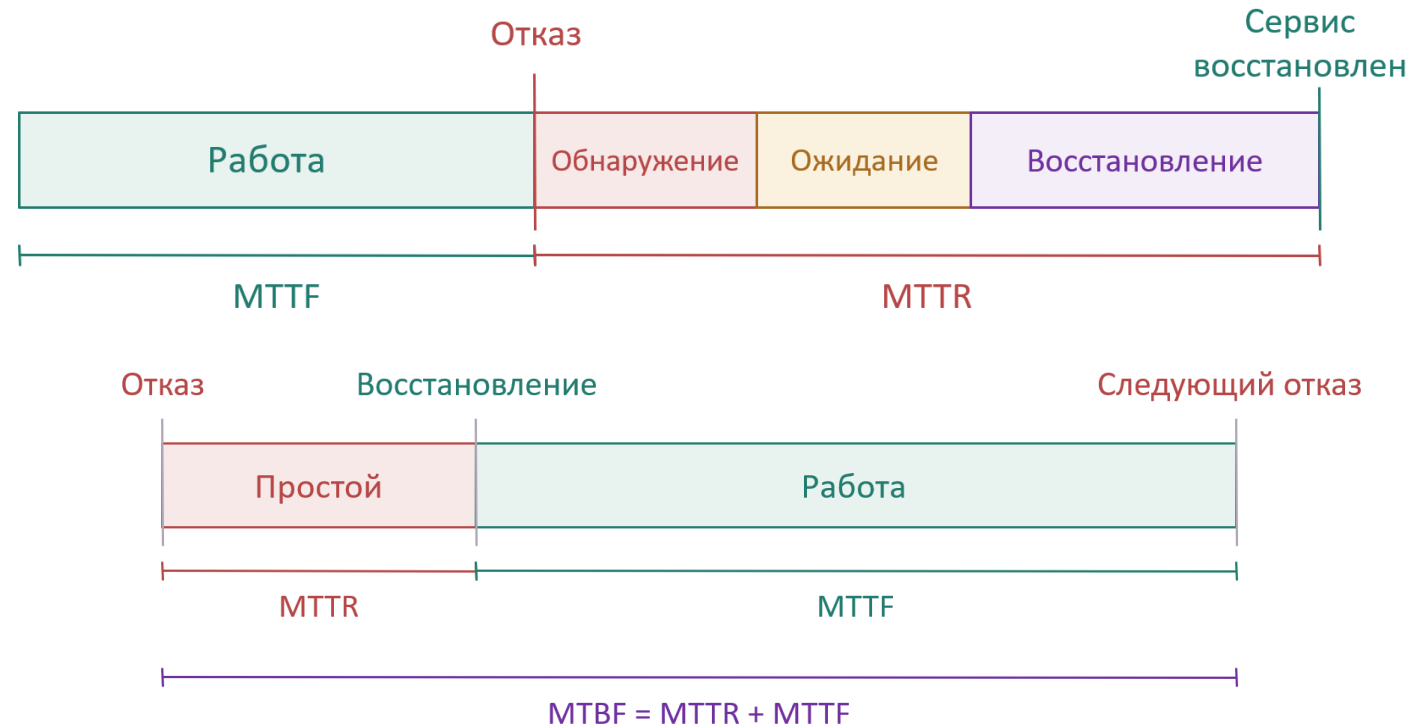
Centralized



Decentralized

Надежность (Reliability)

- Способность системы сохранять работоспособное состояние (не отказывать) в течение заданного интервала при заданных условиях
 - Компоненты системы могут иметь разную надежность
 - Отказ компонента может не приводить к отказу всей системы



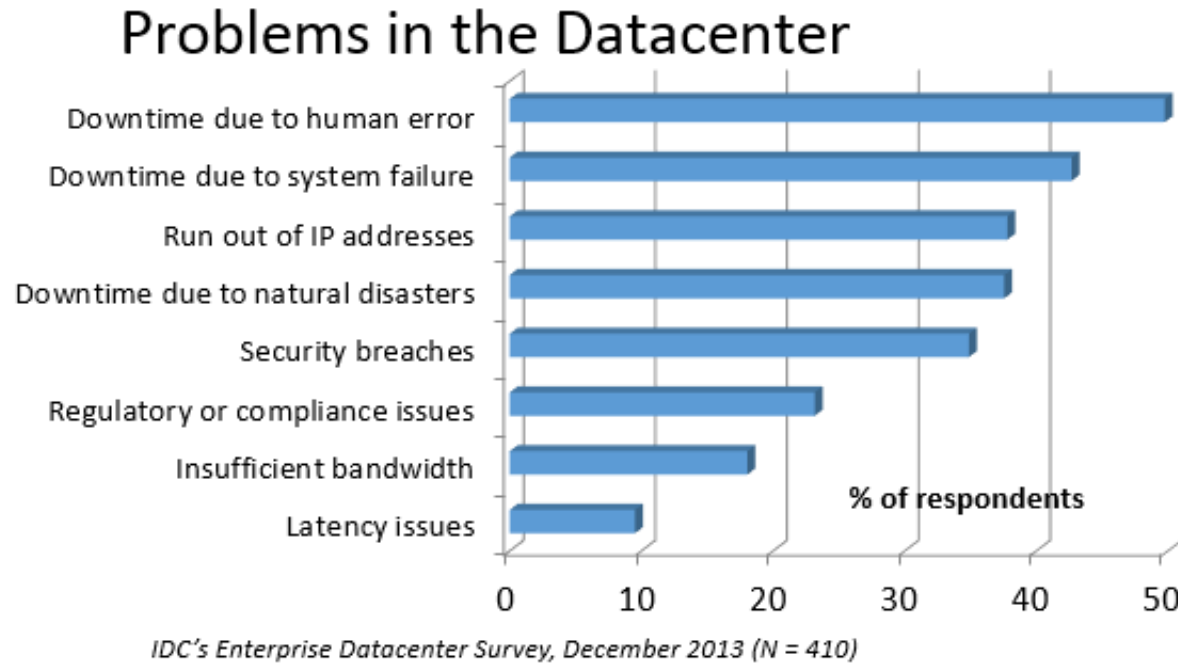
Какие отказы бывают в РС?

Typical first year for a new cluster:

- ~0.5 **overheating** (power down most machines in <5 mins, ~1-2 days to recover)
- ~1 **PDU failure** (~500-1000 machines suddenly disappear, ~6 hours to come back)
- ~1 **rack-move** (plenty of warning, ~500-1000 machines powered down, ~6 hours)
- ~1 **network rewiring** (rolling ~5% of machines down over 2-day span)
- ~20 **rack failures** (40-80 machines instantly disappear, 1-6 hours to get back)
- ~5 **racks go wonky** (40-80 machines see 50% packetloss)
- ~8 **network maintenances** (4 might cause ~30-minute random connectivity losses)
- ~12 **router reloads** (takes out DNS and external vips for a couple minutes)
- ~3 **router failures** (have to immediately pull traffic for an hour)
- ~dozens of minor **30-second blips for dns**
- ~1000 **individual machine failures**
- ~thousands of **hard drive failures**
- slow disks, bad memory, misconfigured machines, flaky machines, etc.**

Long distance links: **wild dogs, sharks, dead horses, drunken hunters, etc.**

Причины отказов



- [What can we learn from four years of data center hardware failures? \(slides\)](#)
- [The Network is Reliable: An informal survey of real-world communications failures](#)
- [Reading postmortems](#) + [collection of postmortems](#)

Доступность (Availability)

Система доступна, когда пользователи могут взаимодействовать с системой, получать требуемые сервисы, корректные ответы и т.д.

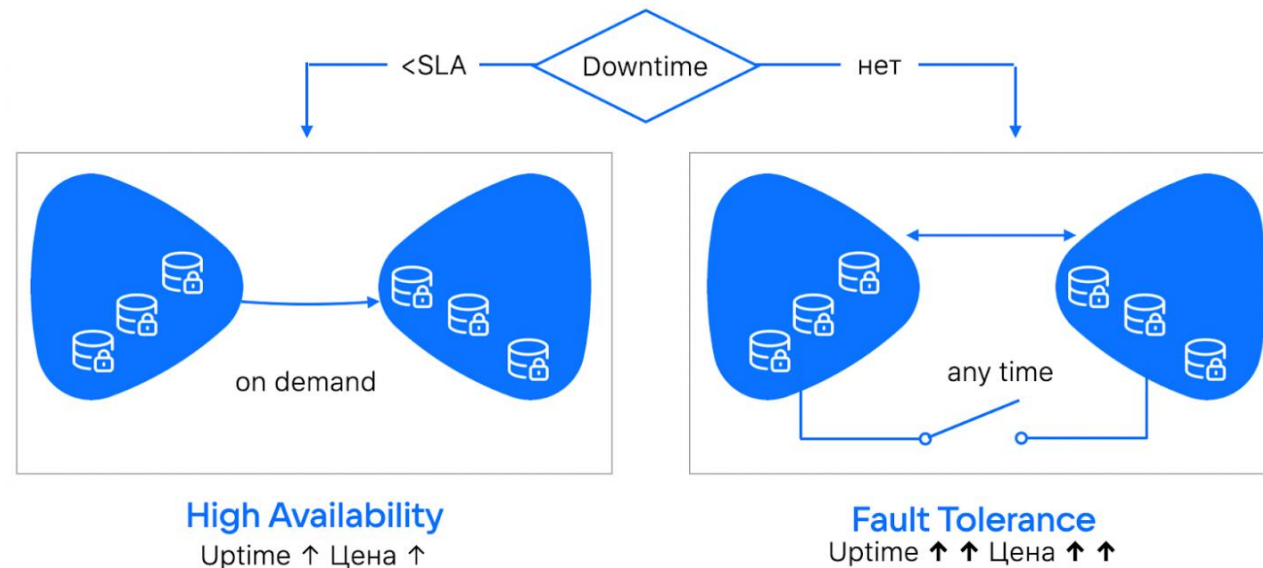
- Доступность часто измеряется как процент времени, когда система доступна
- Причины недоступности: отказы, ошибки, обновление ПО, технические работы...
- Время восстановления после отказов (mean time to repair, MTTR)
- $\text{Availability} = (1 - \text{MTTR}/\text{MTBF}) * 100\%$

Availability %	Downtime/year	Downtime/month	Downtime/week	Downtime/day
90.0% (one 9)	36.53 days	73.05 hours	16.8 hours	2.4 hours
99.0% (two 9s)	3.65 days	7.31 hours	1.68 hours	14.4 minutes
99.9% (three 9s)	8.77 hours	43.83 minutes	10.08 minutes	1.44 minutes
99.99% (four 9s)	52.6 minutes	4.38 minutes	1.01 minutes	8.64 seconds
99.999% (five 9s)	5.25 minutes	26.3 seconds	6.05 seconds	864 ms
99.9999% (six 9s)	31.56 seconds	2.63 seconds	604.8 ms	86.4 ms

Отказоустойчивость (Fault-Tolerance)

Способность системы продолжать нормально функционировать после отказа одного или нескольких её компонентов

- Подразумевает 100% доступность, обходится дороже высокой доступности
- Имеет определенные пределы (например, отказ менее половины узлов)



Обеспечение отказоустойчивости

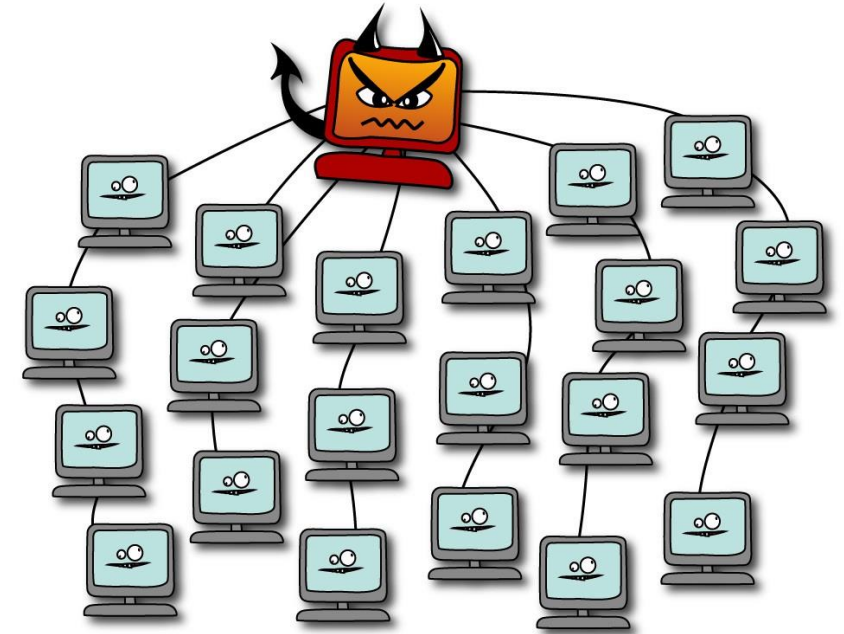
- Исключение единых точек отказа (single points of failure)
- Избыточность на аппаратном уровне, репликация состояния
- Обнаружение и обработка отказов на программном уровне
- Восстановление отказавших компонентов
- Прогнозирование и предотвращение отказов

Особенности отказов в РС

- Процесс А отправил запрос процессу В, но не получил ответа
- Что это значит?
 - Запрос потерялся и не дошел до В
 - Запрос дошел до В, но В пока не успел его обработать
 - Запрос дошел до В, но В упал, не успев обработать его
 - Запрос дошел до В, но В его просто проигнорировал
 - Запрос дошел до В и был обработан, но ответ пока не дошел до А
 - Запрос дошел до В и был обработан, но ответ потерялся при доставке
- Нельзя отличить отказ сети от отказа узла или процесса
- [Стажёр Вася и его истории об идиоматичности API](#)

Безопасность

- Предотвращение возможных угроз
 - Утечка, фальсификация, вандализм...
- Защита от атак
 - Подслушивание, подмена, повтор, DDoS...
- Базовые требования
 - Конфиденциальность
 - Целостность
 - Аутентификация
 - Невозможность отказа
 - Авторизация



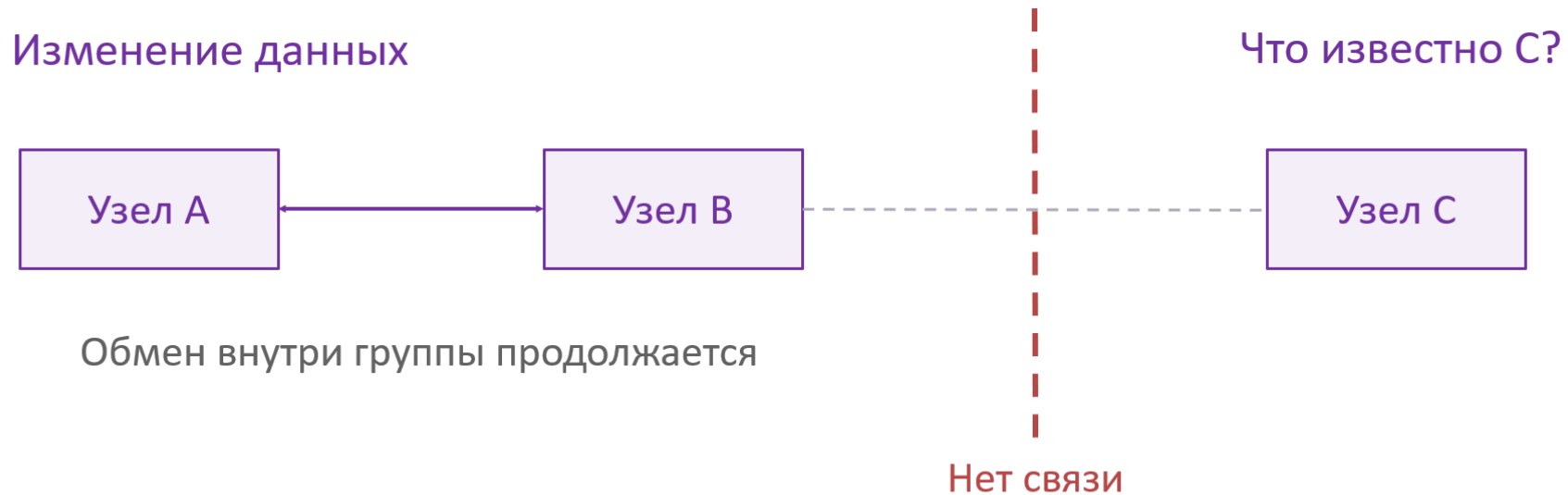
Согласованность (Consistency)

- Модель согласованности определяет гарантии, которые система предоставляет клиентам на операции с хранимыми в ней данными
 - Пример: при чтении всегда возвращается последнее записанное значение
- Как обеспечить если в системе есть несколько реплик (копий) данных и одновременно работающих клиентов?



Разделение сети

- Узлы работают, но часть системы не может связаться с другой
- Как обеспечить доступность и согласованность?



Продолжать работу с неполной информацией или ждать восстановления связи?

Удобство сопровождения (Maintainability)

- Есть ли удобный мониторинг системы и подробное логирование?
- Насколько быстро можно диагностировать и устранить проблему?
- Можно ли выполнять обновления системы без downtime?
- Можно ли отключить часть машин и продолжать работать?
- Насколько быстро система восстанавливается после полной остановки?
- Насколько легко можно проводить расширение системы?
- Насколько легко понять как устроена и работает система?
- Можно ли адаптировать систему под меняющиеся требования?

Реализация распределенных систем

“Distributed systems need radically different software than centralized systems do.” (A. Tanenbaum)

Отличия от классической модели

~~Классический~~ Распределенный алгоритм

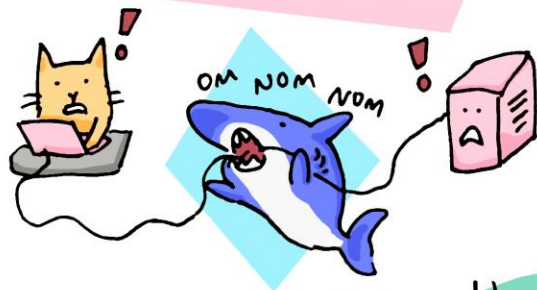
~~Последовательное~~ Параллельное исполнение (concurrency)

~~Полная~~ Частичная информация (нет глобальных часов)

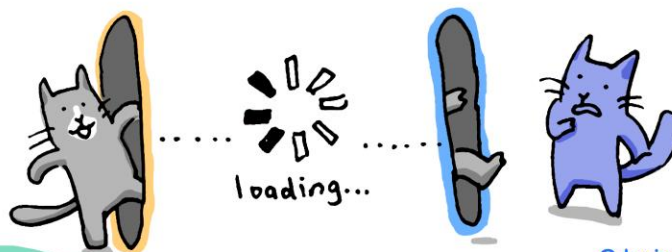
~~Отсутствуют~~ Присутствуют независимые отказы частей системы

Сложность ~~$C(\#Op)$~~ $C(\#Op, \#Comm)$

① The network is reliable



② Latency is ZERO



③ Bandwidth is infinite



⑧ The network is homogeneous



the 8 Fallacies of Distributed Computing

Originally formulated by L. Peter Deutsch & Colleagues at Sun Microsystems in 1994; #8 added in 1997 by James Gosling

④ The network is secure



⑦ Transport costs \$0



⑥ There is only one administrator



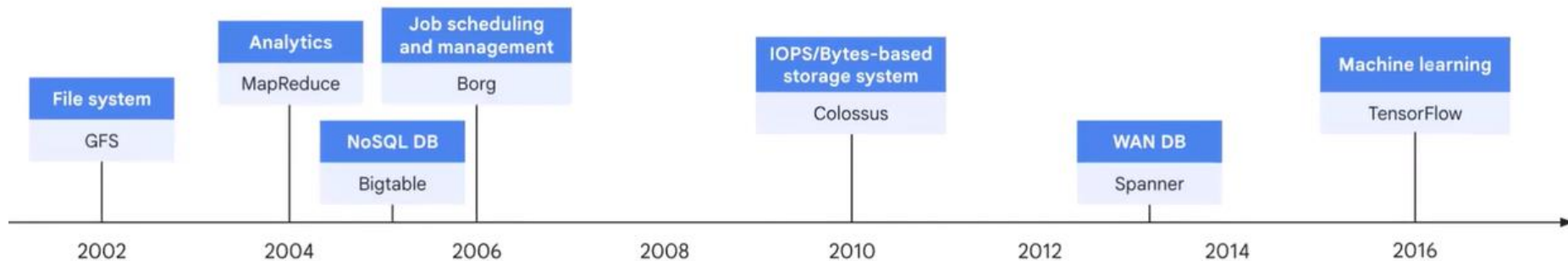
⑤ Topology doesn't change



<https://deniseyu.io/art>

Distributed Computing

- Раздел компьютерных наук, изучающий распределенные системы
 - Теоретические модели РС, типовые задачи, распределенные алгоритмы
- Применение распределенных систем для решения трудоемких вычислительных задач
 - Разновидность параллельных вычислений
- Опыт разработки распределенных систем в индустрии



Типовые задачи

- Взаимодействие между процессами (в паре или группе)
- Обнаружение отказов и учёт участников
- Именованное, поиск и распространение информации
- Масштабирование, балансировка нагрузки
- Репликация данных, обеспечение согласованности
- Упорядочивание событий, обнаружение конфликтов
- Координация процессов (взаимное исключение, выборы лидера, консенсус...)
- Организация параллельной обработки запросов и данных
- Обеспечение безопасности и устойчивости к произвольным отказам

Материалы

- Distributed Systems (глава 1)
- Designing Data-Intensive Applications (глава 1)
- Why Do We Need Distributed Systems?
- Why Are Distributed Systems So Hard?
- Fallacies of Distributed Computing Explained
- Lessons and Advice from Building Large Distributed Systems
- Coming of Age in the Fifth Epoch of Distributed Computing

