

# Введение

Олег Сухорослов

Распределенные системы

Факультет компьютерных наук НИУ ВШЭ

05.09.2022

# План

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

# Цели курса

- Вводный курс по распределенным системам
  - Изучение типовых задач и методов их решения
  - Знакомство с различными классами систем и их особенностями
  - Изучение технологий и получение практических навыков
- 
- Дополнительный материал будет изучаться на НИСе
  - Более глубокое погружение на последующих курсах специализации

# Содержание

- Взаимодействие между процессами
- Протокол HTTP и веб-сервисы
- Групповые взаимодействия и рассылка
- Непрямое взаимодействие
- Обнаружение отказов
- Именование и поиск
- Масштабирование
- Паралельная обработка
- Репликация данных и согласованность
- Время, часы и порядок событий
- Консенсус и связанные задачи
- Безопасность
- Устойчивость к произвольным отказам

# Организация

- Занятия
- Домашние задания
- Проверочные
- Телеграм, Canvas, репозиторий
- Описание выложено на вики ФКН, прочтайте его

# Оценка

- ДЗ - средняя оценка за домашние задания
- ПР - средняя оценка за проверочные работы
- Итоговая оценка =  $\text{round}( 0.7 * \text{ДЗ} + 0.3 * \text{ПР} )$
- Экзамена нет

# Вопросы?

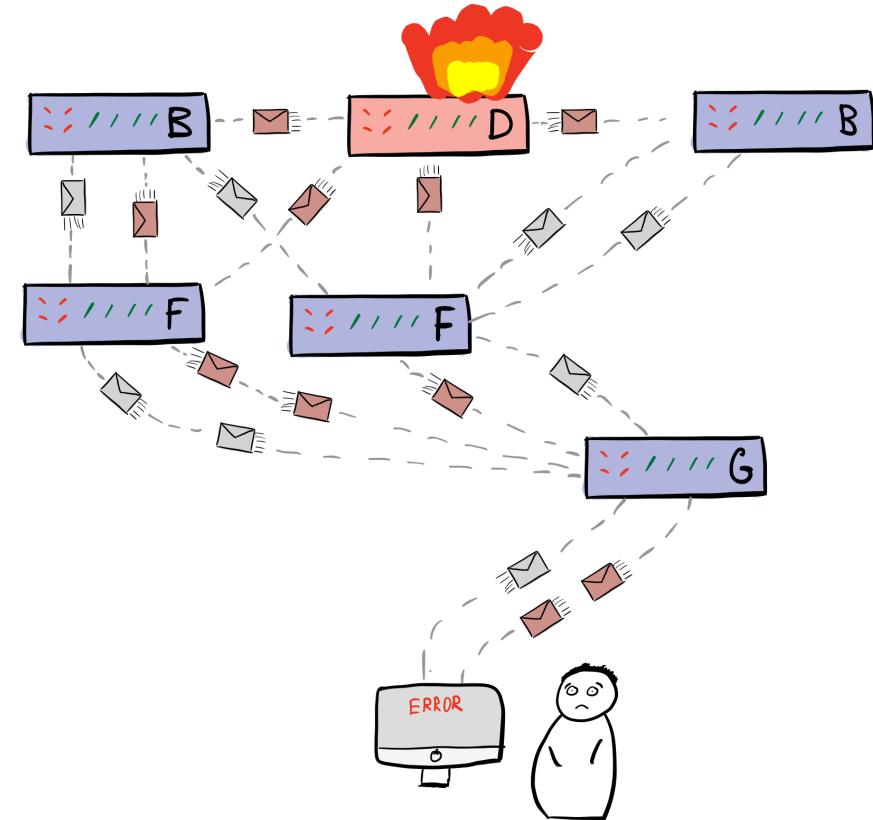
# Распределенная система

# 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)

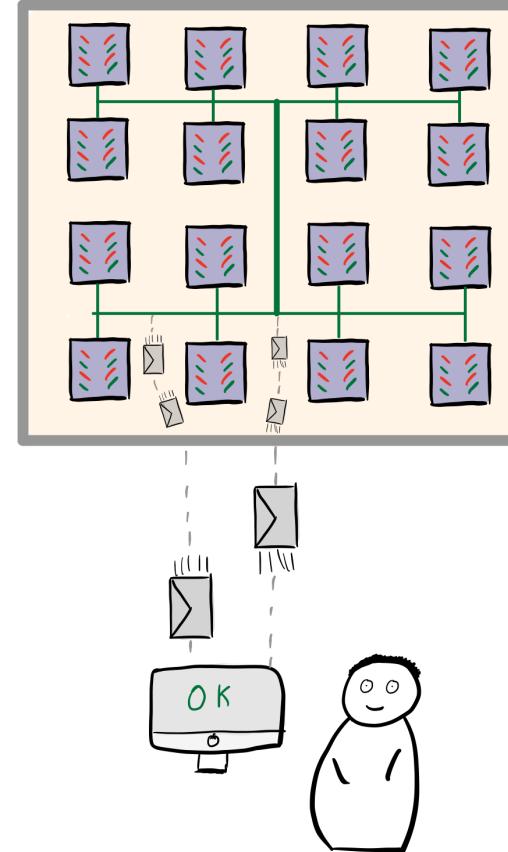
<https://www.microsoft.com/en-us/research/publication/distribution/>



# Определение 2

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

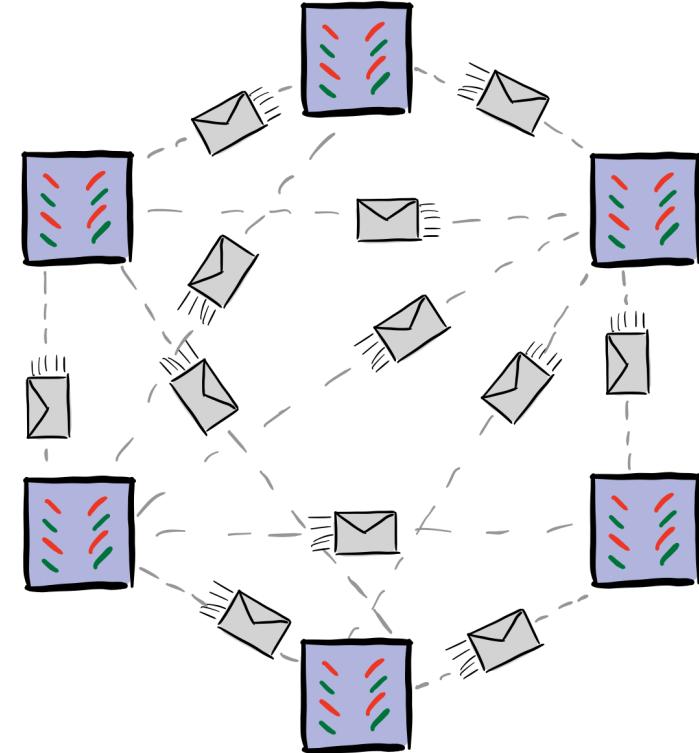
van Steen, Tanenbaum. Distributed Systems: Principles and Paradigms.



# 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)

- A *program* is ...
- A *process* is ...
- A *message* is ...
- A *packet* is ...
- A *protocol* is ...
- A *node* is ...
- A *network* is ...
- A *component* is ...

# 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.

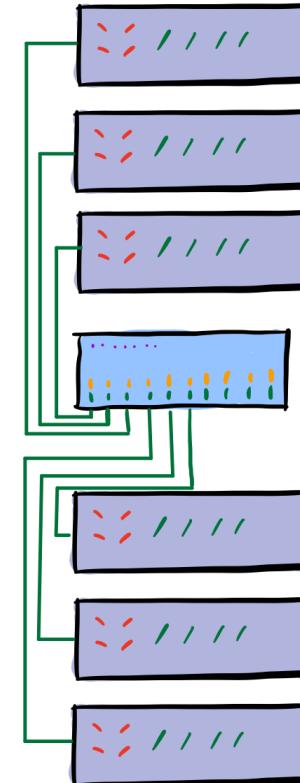
## Определение 4

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

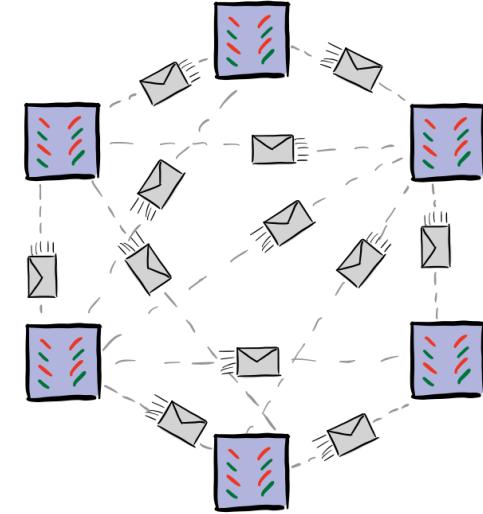
# Распределенная система

- **С аппаратной точки зрения:** совокупность автономных узлов, связанных сетью
  - Функционируют независимо, нет привычных разделяемых ресурсов (часы, память)
  - Могут быть географически分散, иметь отличающиеся характеристики
  - Подвержены (частичным) отказам, как и сеть между ними



# Распределенная система

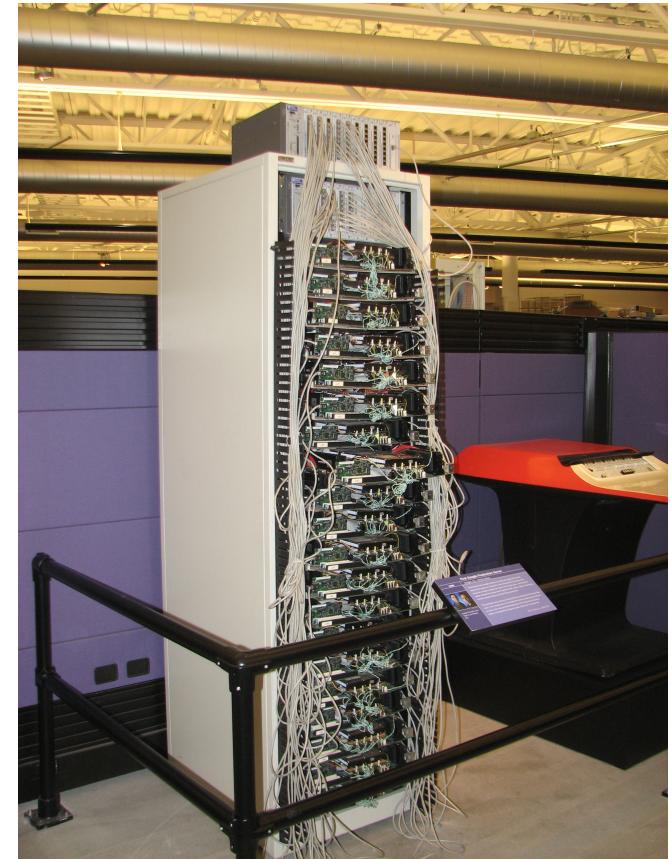
- **С программной точки зрения:**  
совокупность независимых процессов,  
взаимодействующих посредством  
передачи сообщений
  - Процессы выполняются на различных узлах
  - Каждый процесс имеет собственное состояние
  - Процессы не имеют прямого доступа к состояниям других процессов
  - Сообщения могут теряться,  
переупорядочиваться и дублироваться



# Зачем нужны распределенные системы?

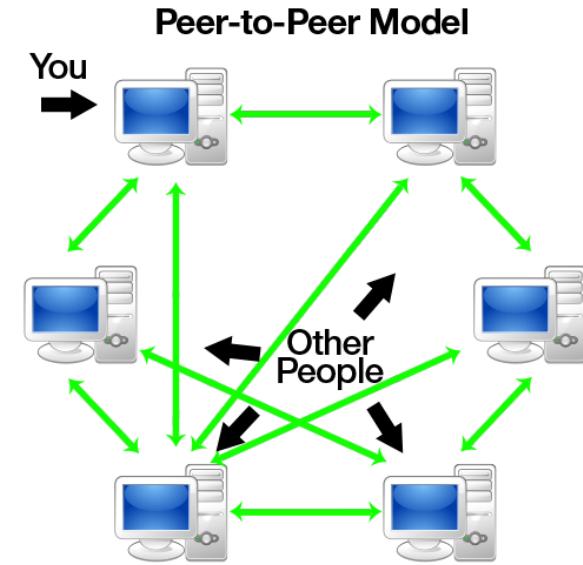
# Применение распределенных систем

- Увеличение производительности
  - Решение больших задач (HPC)
  - Хранение и обработка больших объемов данных (Big Data)
  - Обслуживание большого количества клиентов (Web/Cloud Services)
- Отказоустойчивость и доступность
  - Устойчивость к частичным отказам за счет избыточности



# Применение распределенных систем (2)

- 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 (распределенные) системы

- Email, обмен сообщениями
- Интернет-банк
- Веб-поиск
- Онлайн-редактор документов
- Социальная сеть
- Хранилище данных
- Грид-система
- Облако
- Сеть доставки контента, онлайн-кинотеатр
- Файлообменная сеть, криптовалюта



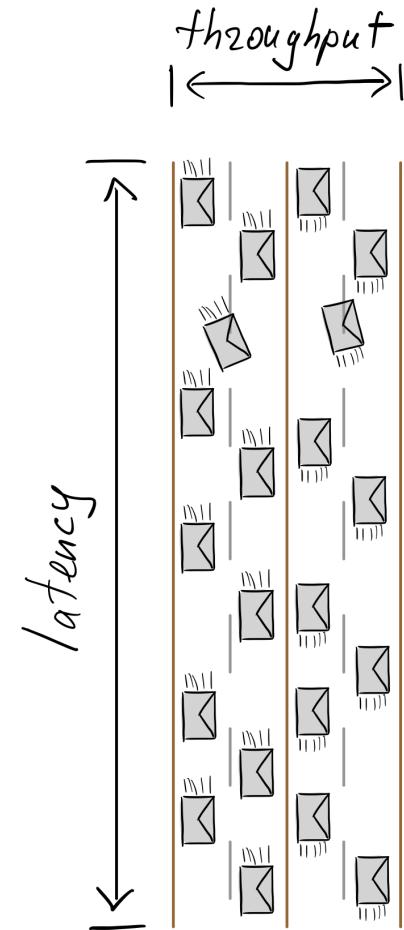
# Нефункциональные требования

Базовые свойства, которыми должна обладать система:

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

# Производительность (Performance)

- Основные показатели
  - Задержка, время обработки запроса, время ожидания ответа
  - Пропускная способность, число обрабатываемых запросов/данных в секунду
  - Качество обслуживания, битрейт, доля пропущенных кадров потокового видео
- Могут конфликтовать друг с другом
  - В разных системах может отдаваться приоритет разным показателям



# Как оценить производительность?

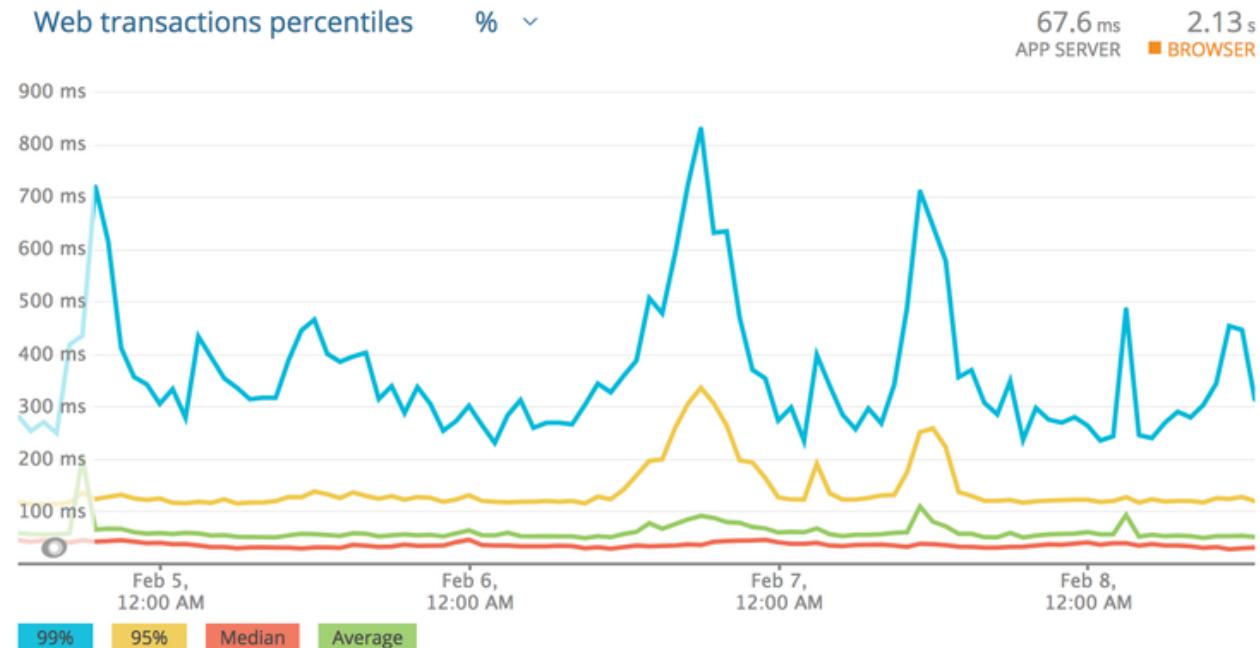
Type	RAM	SSD	HDD	Data center	Internet
Latency	100 ns	10 μs	1 ms	1 ms	100 ms
Throughput	100 GB/s	1 GB/s	100 MB/s	1 GB/s	10 MB/s

$$1\text{ s} = 10^3 \text{ ms} = 10^6 \mu\text{s} = 10^9 \text{ ns}$$

Latency Numbers Every Programmer Should Know

# Как измерить производительность?

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



Источник: <https://blog.runscope.com/posts/phil-sturgeon-taking-a-timeout-from-poor-performance>

# Масштабируемость (Scalability)

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

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



# Нагрузочная масштабируемость

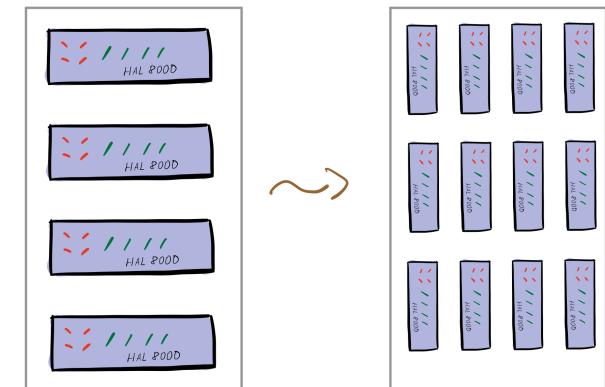
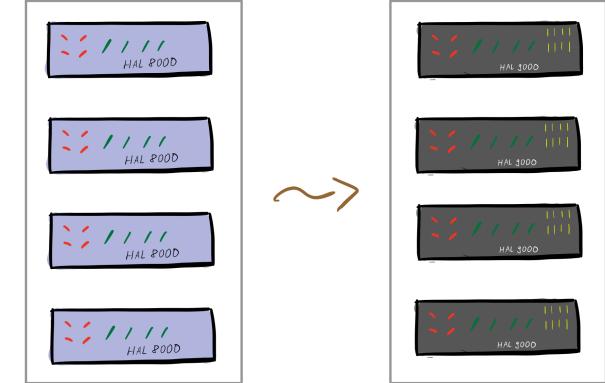
Способность системы увеличивать свою производительность при увеличении нагрузки путем замены или добавления аппаратных средств

Параметры, описывающие нагрузку

- Число запросов в секунду
- Число активных пользователей
- Соотношение операций чтения и записи

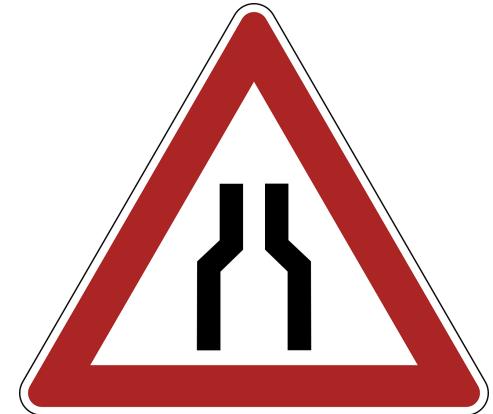
Подходы

- вертикальное масштабирование (scale up)
- горизонтальное масштабирование (scale out)



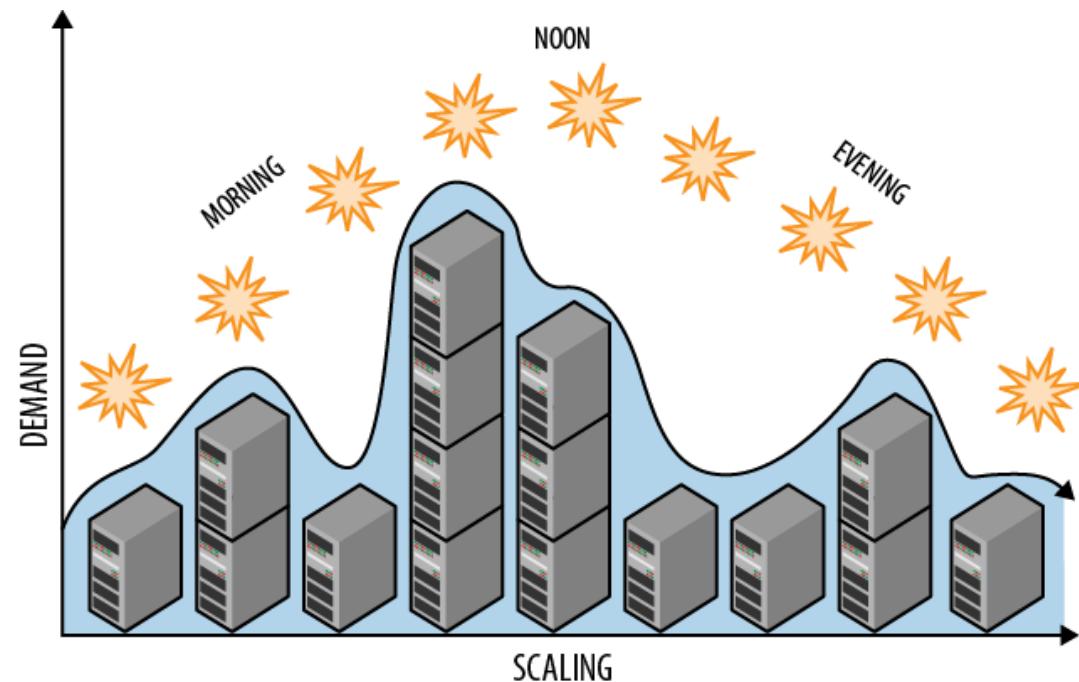
# Узкие места

- Централизованные компоненты
  - Вычислительный сервис (CPU-bound)
  - Поисковый сервис (I/O-bound)
  - Видео по запросу (сетевой канал)
- Накладные расходы на взаимодействия
- Пропускная способность сети
- Системное и промежуточное ПО



Чем больше система, тем сложнее ее масштабировать

# Эластичность



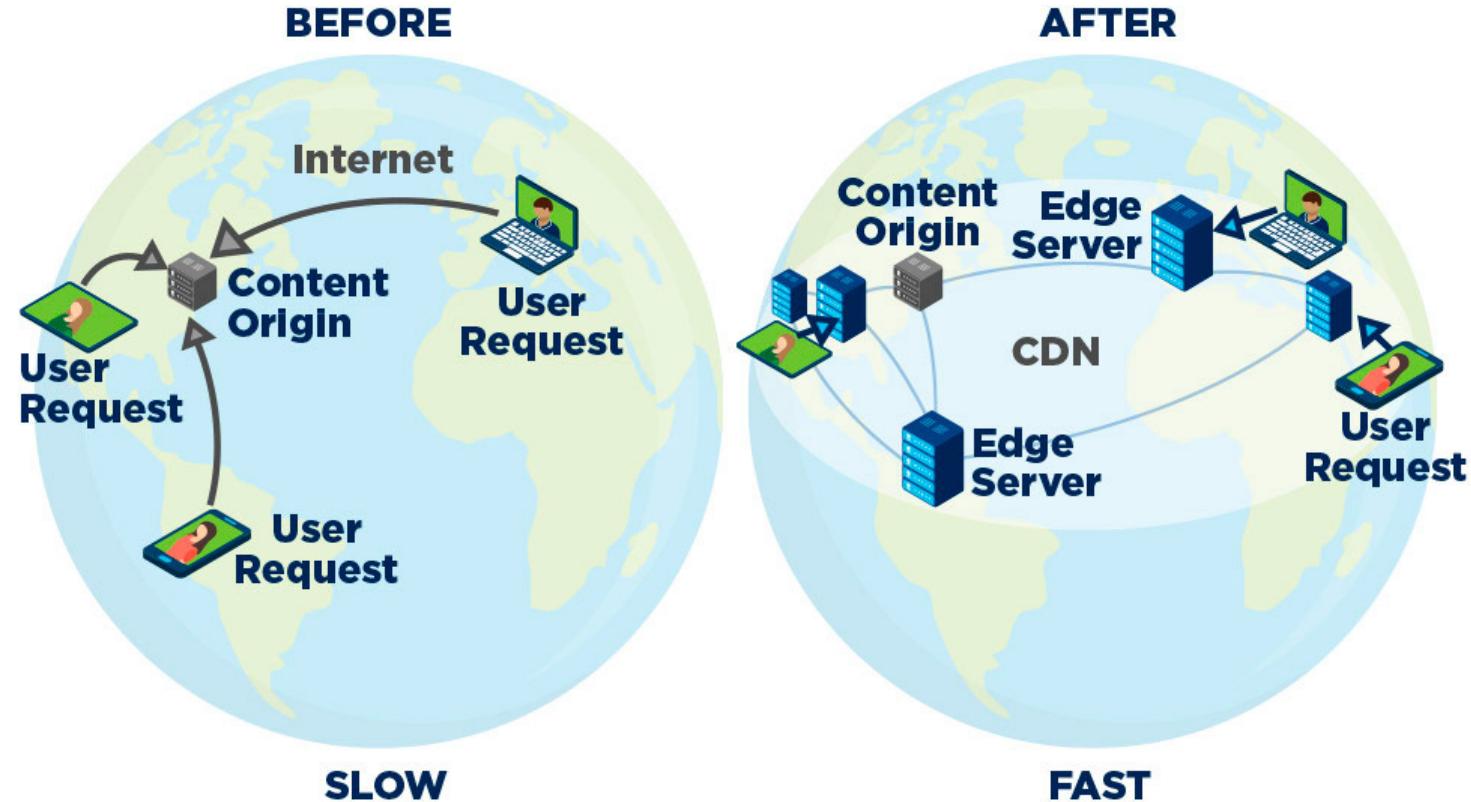
Автоматическое масштабирование ресурсов под текущую нагрузку

# 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

# Content Delivery Network (CDN)

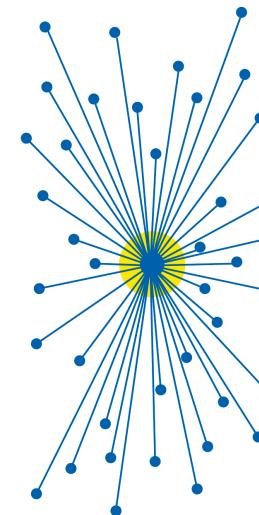


Источник: <https://www.limelight.com/resources/white-paper/5-things-multi-cdn-strategy/>

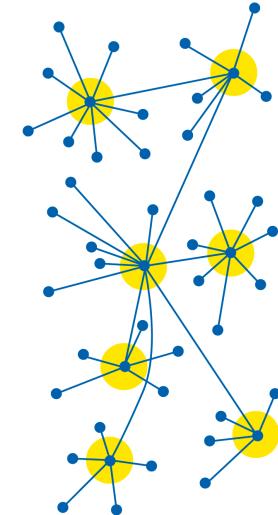
# Административная масштабируемость

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

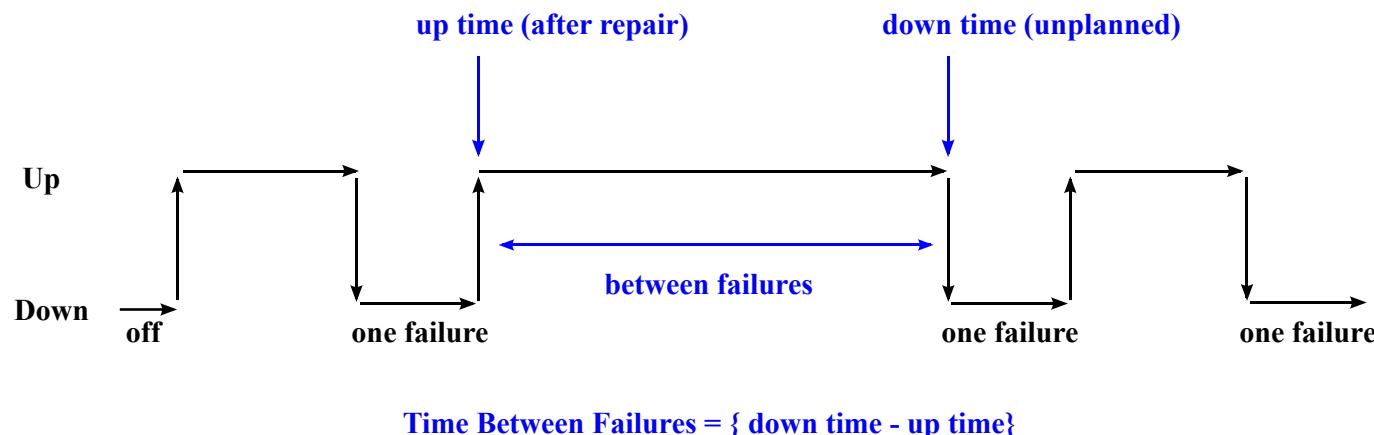


Decentralized

# Надежность (Reliability)

Способность системы сохранять работоспособное состояние (не отказывать) в течение некоторого промежутка времени

- Характеризуется с помощью средней продолжительности работы между отказами (mean time between failures, MTBF)



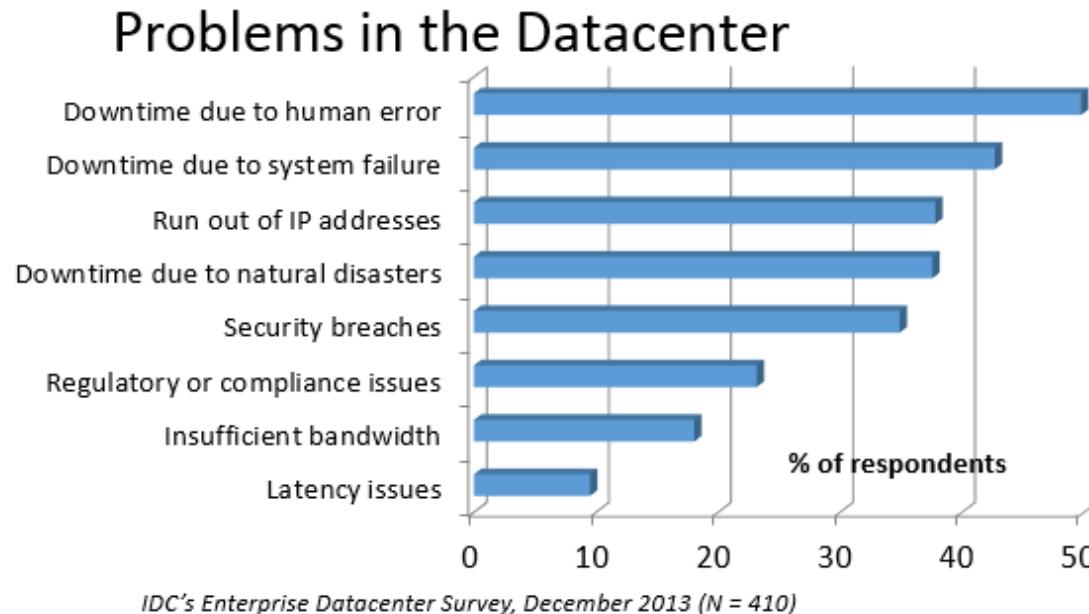
Источник: [https://en.wikipedia.org/wiki/Mean\\_time\\_between\\_failures](https://en.wikipedia.org/wiki/Mean_time_between_failures)

# Типичные отказы за год (Jeff Dean, Google)

- ~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.

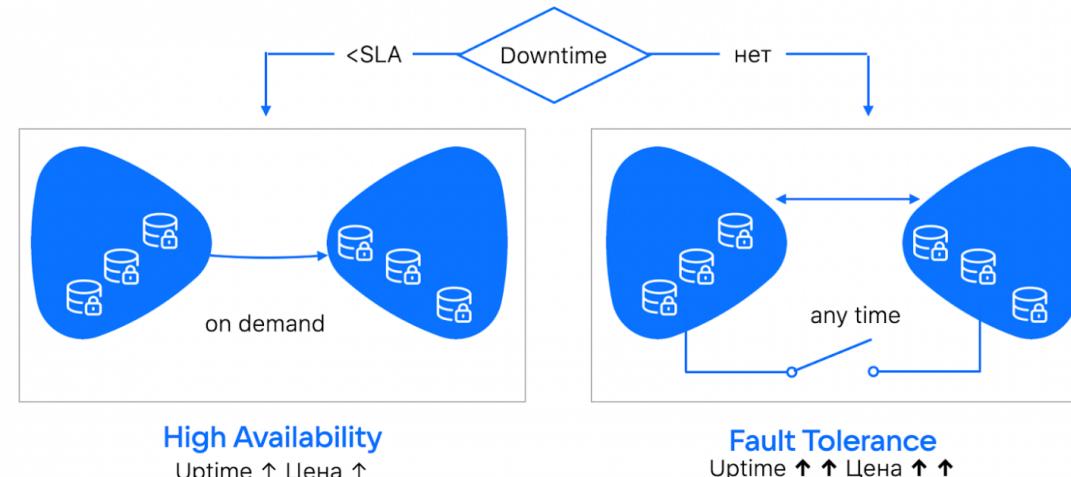
- Доступность часто измеряется как процент времени, когда система доступна
- Причины недоступности: отказы, ошибки, обновление ПО, технические работы...
- Время восстановления после отказов (mean time to repair, MTTR)
- Availability =  $(1 - \text{MTTR}/\text{MTBF}) * 100\%$

Availability	Downtime / Year	Downtime / Month	Downtime / Week	Downtime / Day
99.999%	5.256 Minutes	0.438 Minutes	0.101 Minutes	0.014 Minutes
99.995%	26.28 Minutes	2.19 Minutes	0.505 Minutes	0.072 Minutes
99.990%	52.56 Minutes	4.38 Minutes	1.011 Minutes	0.144 Minutes
99.950%	4.38 Hours	21.9 Minutes	5.054 Minutes	0.72 Minutes
99.900%	8.76 Hours	43.8 Minutes	10.108 Minutes	1.44 Minutes
99.500%	43.8 Hours	3.65 Hours	50.538 Minutes	7.2 Minutes
99.250%	65.7 Hours	5.475 Hours	75.808 Minutes	10.8 Minutes
99.000%	87.6 Hours	7.3 Hours	101.077 Minutes	14.4 Minutes

# 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

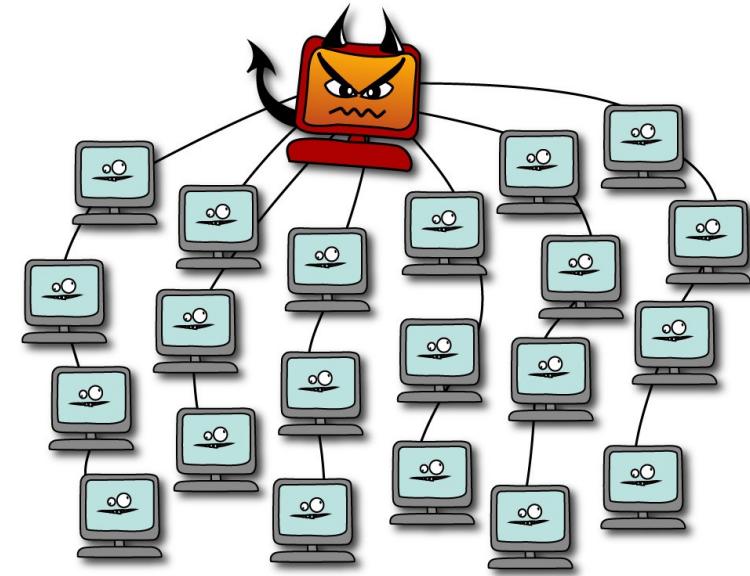
- Исключение единых точек отказа (single points of failure)
- Избыточность на аппаратном уровне, репликация состояния
- Обнаружение и обработка отказов на программном уровне
- Восстановление отказавших компонентов
- Прогнозирование и предотвращение отказов

# 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
- Пример из жизни

# Безопасность

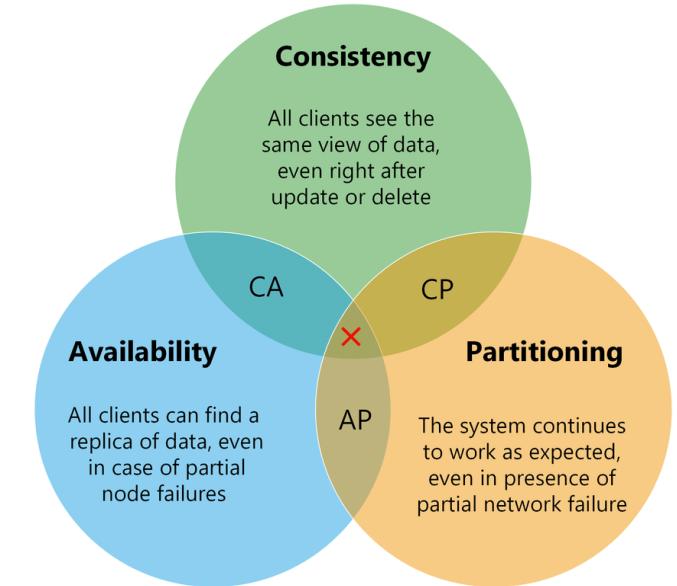
- Предотвращение возможных угроз
  - Утечка, фальсификация, вандализм...
- Защита от атак
  - Подслушивание, подмена, повтор, DDoS...
- Базовые требования
  - Конфиденциальность
  - Целостность
  - Аутентификация
  - Невозможность отказа
  - Авторизация



# Согласованность (Consistency)

Модель согласованности определяет гарантии, которые система дает клиентам на операции с хранимыми в ней данными

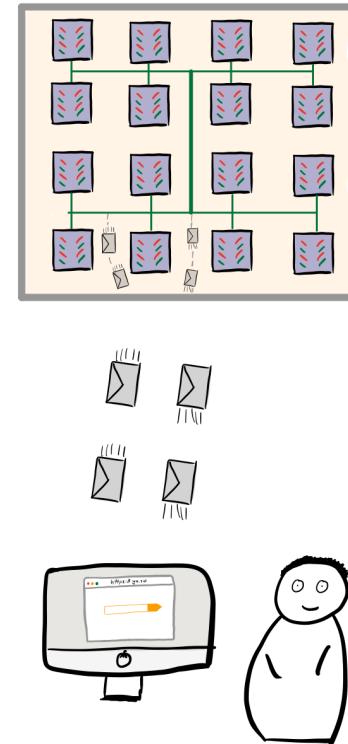
- Пример: при чтении записи из базы всегда возвращается последнее записанное значение
- Как обеспечить если в системе есть несколько реплик данных и одновременно работающих клиентов?
- Чем сильнее гарантии согласованности, тем сложнее и дороже их обеспечивать
- Компромисс между согласованностью и доступностью



# Прозрачность

Способность системы скрывать от пользователей и приложений свою распределенную природу, то есть делать "прозрачным" физическое распределение процессов и ресурсов

- Прозрачность доступа, местоположения, репликации, одновременного доступа, отказов, масштабирования ...
- Обеспечить полную прозрачность (иллюзию работы с локальной системой) крайне сложно
- Стоит ли скрывать распределенность системы?



# Открытость

Система реализует открытые спецификации интерфейсов, протоколов, форматов данных и т.д.

Открытая спецификация

- общедоступна, не принадлежит производителю
- не зависит от конкретных технических и программных средств или продуктов
- поддерживается открытым процессом, под контролем общественного мнения



Преимущества

- Переносимость прикладного ПО
- Поддержка нескольких независимых реализаций
- Способность взаимодействия с другими приложениями и системами
- Легкая миграция пользователей от системы к системе

# 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. Tannenbaum

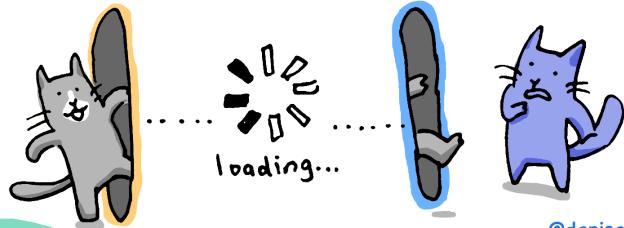
# Особенности

- Классический Распределенный алгоритм
- Последовательное Параллельное исполнение (concurrency)
- Полная Частичная информация (нет глобальных часов)
- Отсутствуют Присутствуют независимые отказы частей системы
- Сложность  $\in(\#Op) C(\#Op, \#Comm)$

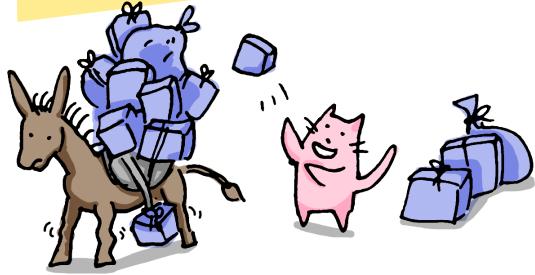
① The network is reliable



② Latency is ZERO



③ Bandwidth is infinite



⑧ The network is homogeneous



④ The network is secure



⑦ Transport costs \$0



⑥ There is only one administrator



⑤ Topology doesn't change



Источник: <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

# Типовые задачи

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

# Материалы к лекции

- van Steen M., Tanenbaum A.S. Distributed Systems: Principles and Paradigms. (глава 1)
- Kleppmann M. Designing Data-Intensive Applications. (глава 1)
- Yu D. Why Are Distributed Systems So Hard?
- Rotem-Gal-Oz A. Fallacies of Distributed Computing Explained
- Dean J. Designs, Lessons and Advice from Building Large Distributed Systems