

Введение

Сухорослов Олег Викторович

Распределенные системы

Факультет компьютерных наук НИУ ВШЭ

05.09.2020

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, возникающих при построении распределенных систем, и методов их решения
- Знакомство с различными классами распределенных систем и особенностями их реализации
- Изучение технологий и получение практических навыков разработки распределенных систем
- Более глубокое погружение в теорию будет на 4 курсе (ТОРС)
- Также дополнительный материал будет изучаться на НИС-е

Содержание

- Взаимодействие между процессами
- Обнаружение отказов
- Именование и поиск
- Масштабирование
- Параллельная обработка
- Репликация данных
- Порядок событий
- Распределенные алгоритмы
- Координация и консенсус
- Безопасность

Организация

- Лекции
- Семинары
- Проверочные
- Домашние задания
- Репозиторий
- Литература
- Канал и чат в Telegram

Оценка

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

Вопросы?

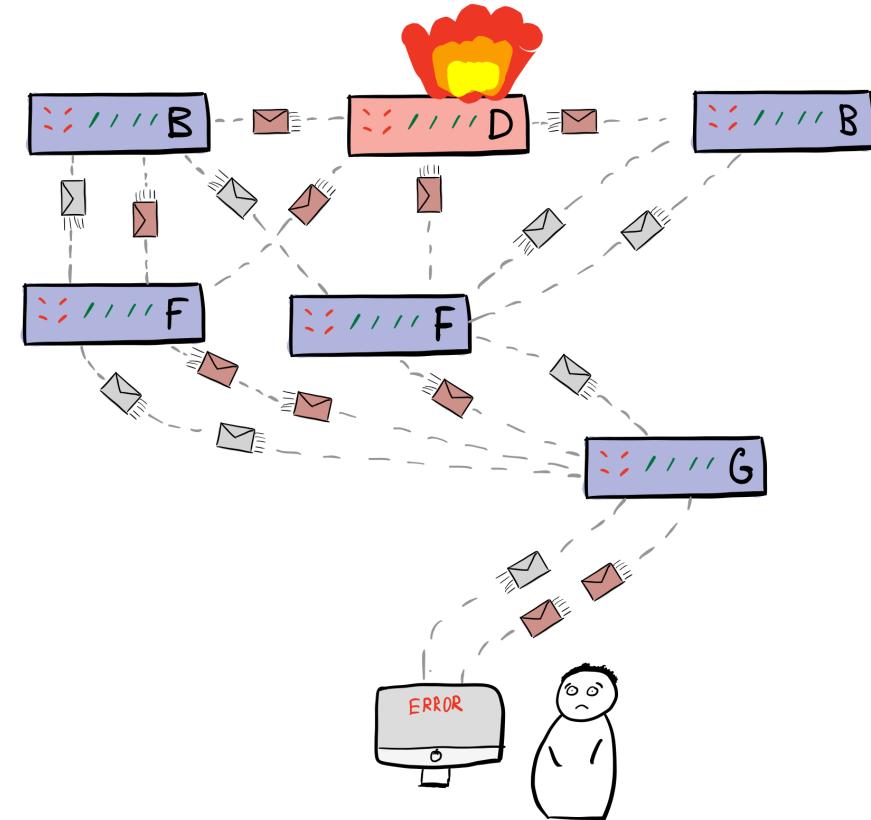
Распределенная система

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)

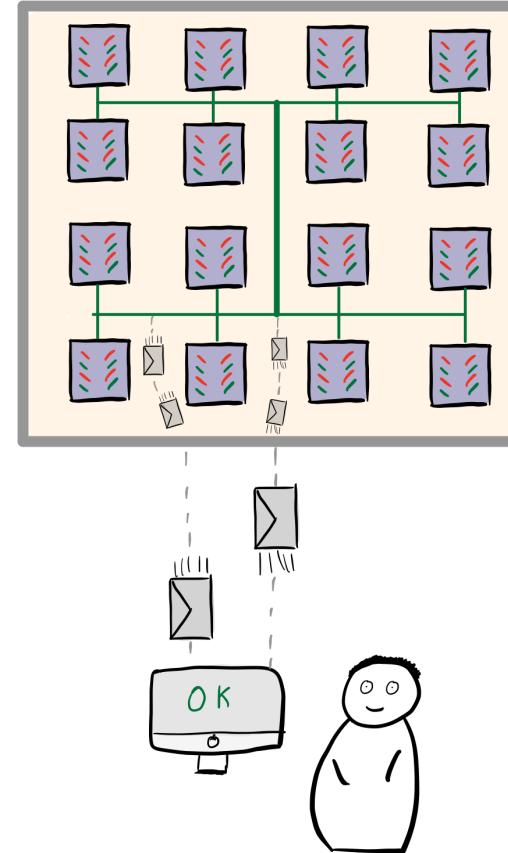
<https://www.microsoft.com/en-us/research/publication/distribution/>



Определение 2

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

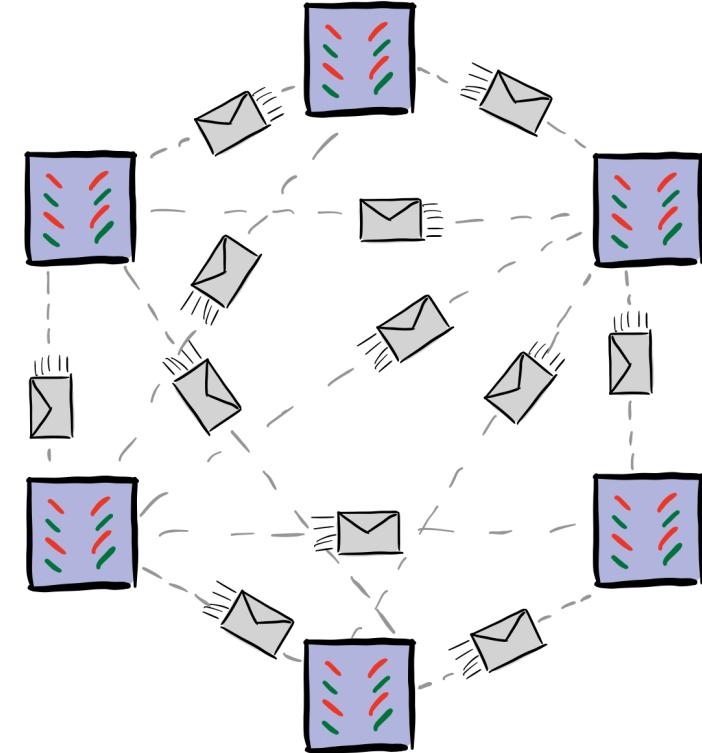
van Steen, Tanenbaum. Distributed Systems: Principles and Paradigms.



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 (1)

- 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

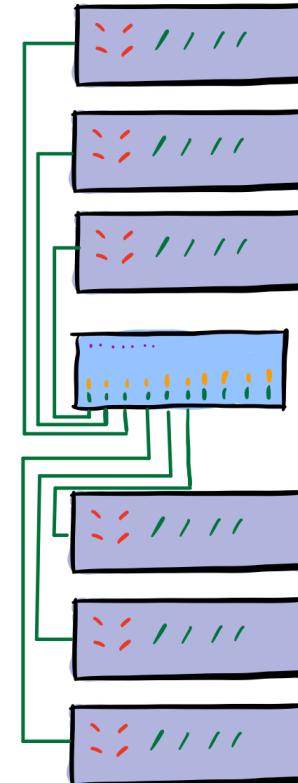
Определение 4 (2)

- A ***component*** can be a process or any piece of hardware required to run a process, support communications between processes, store data, etc.
- 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

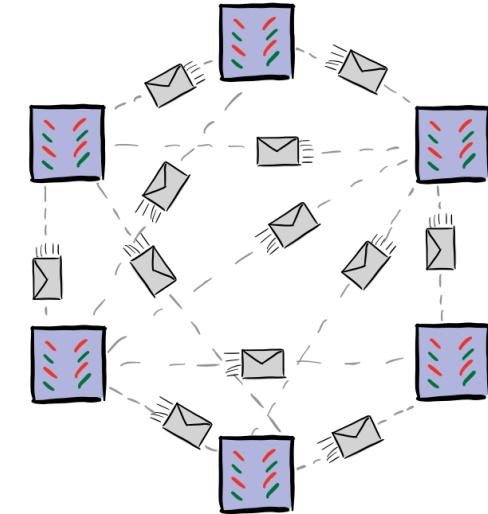
Распределенная система

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



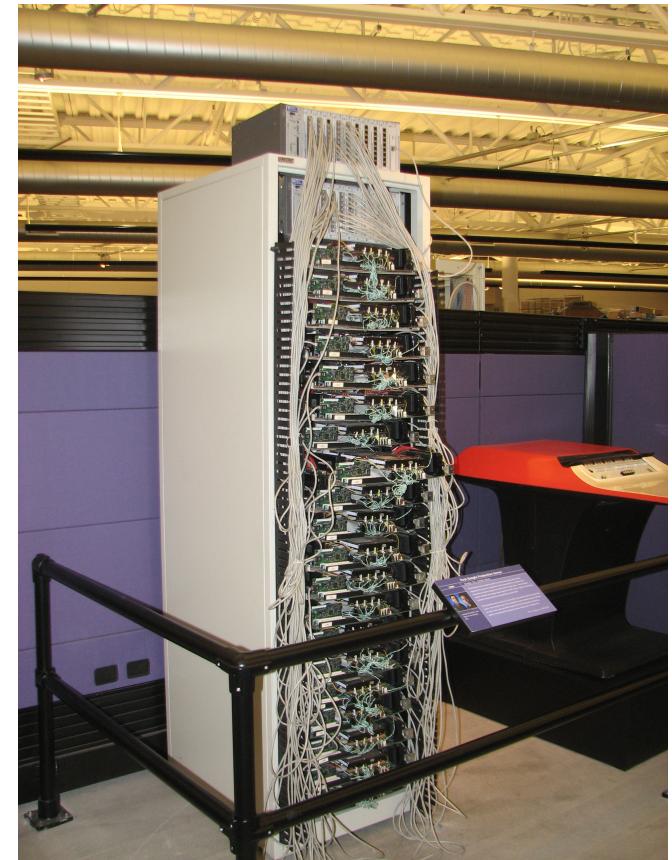
Распределенная система

- 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



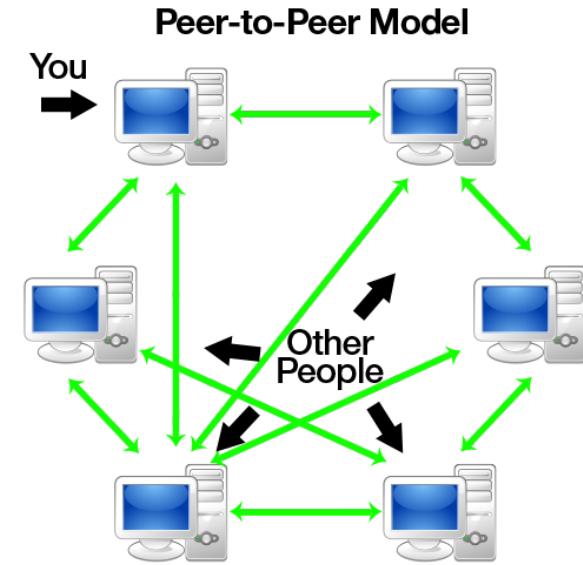
Применение распределенных систем

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



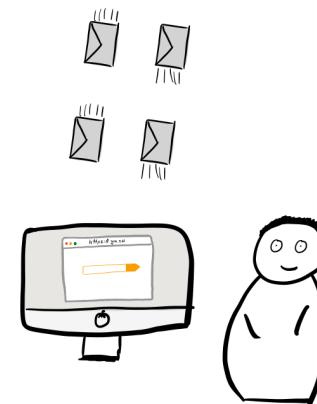
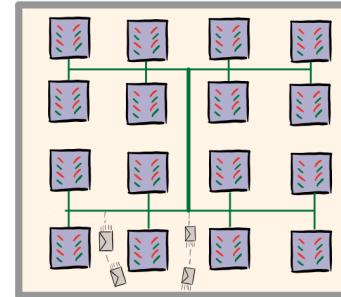
Применение распределенных систем (2)

- 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



Примеры приложений

- Вычисления и обработка данных
- Хранение и доступ к данным
- Интеграция приложений
- Массовые (высоконагруженные) сервисы
- Повсеместные вычисления



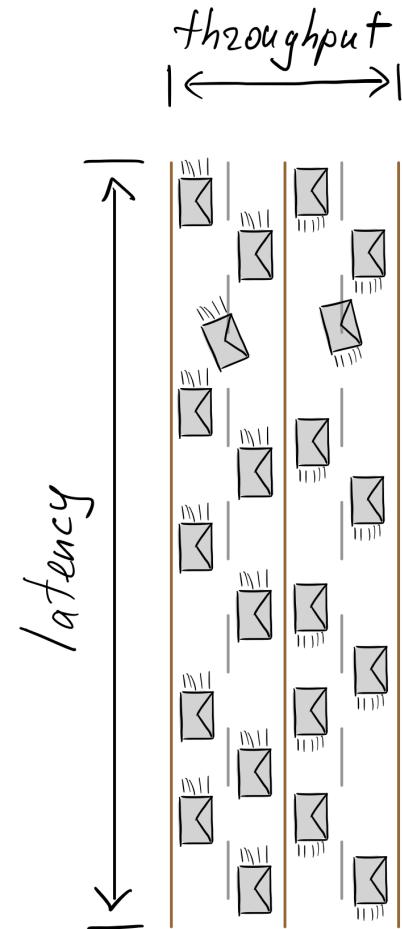
Нефункциональные требования

Базовые свойства, которыми должна обладать система:

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

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



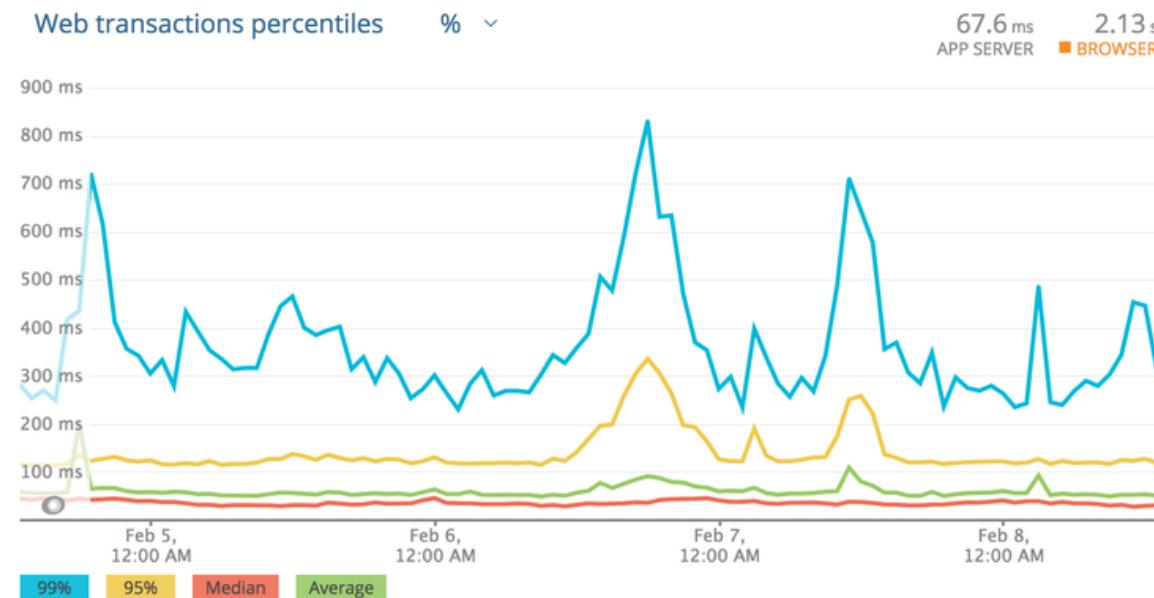
Как оценить производительность?

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

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

Как измерить производительность?

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



Масштабируемость (Scalability)

Способность системы "расти" 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

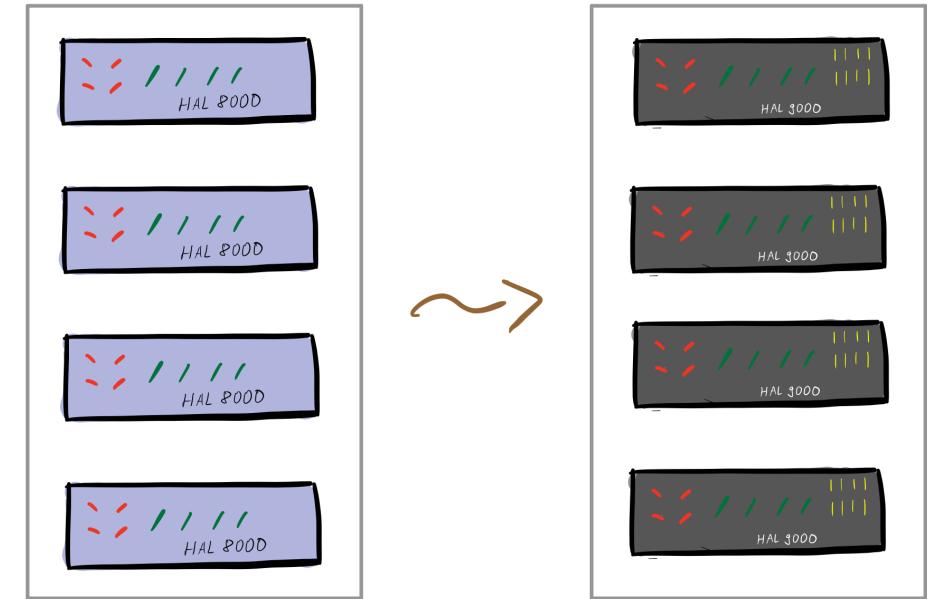


Нагрузочная масштабируемость

- Способность системы увеличивать свою производительность при увеличении нагрузки путем замены существующих или добавления новых 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)

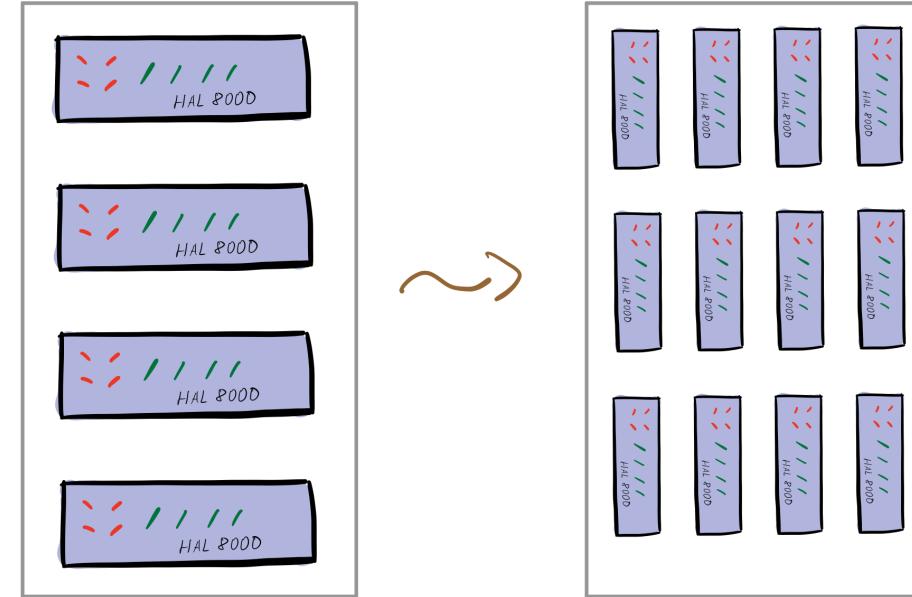
Вертикальное масштабирование

Замена существующих аппаратных средств на более мощные, обладающие лучшими характеристиками



Горизонтальное масштабирование

Наращивание аппаратных средств
путем добавления в систему
новых узлов

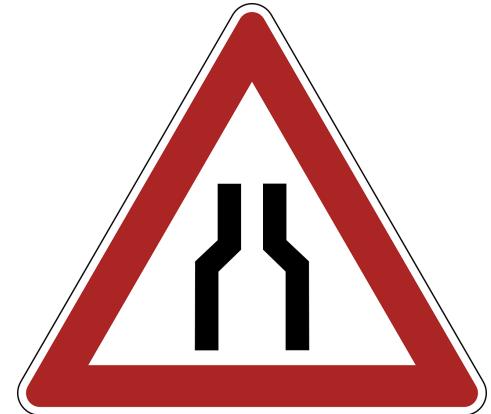


Пример

- Система из 4 серверов легко справляется с нагрузкой 10K RPS, при этом в 99% время ответа < 70 ms
 - Может ли система обрабатывать 100K RPS? 1M?
 - Сколько понадобится для этого серверов?
 - Как это повлияет на время ответа?
 - Что произойдет при увеличении объема хранимых данных?

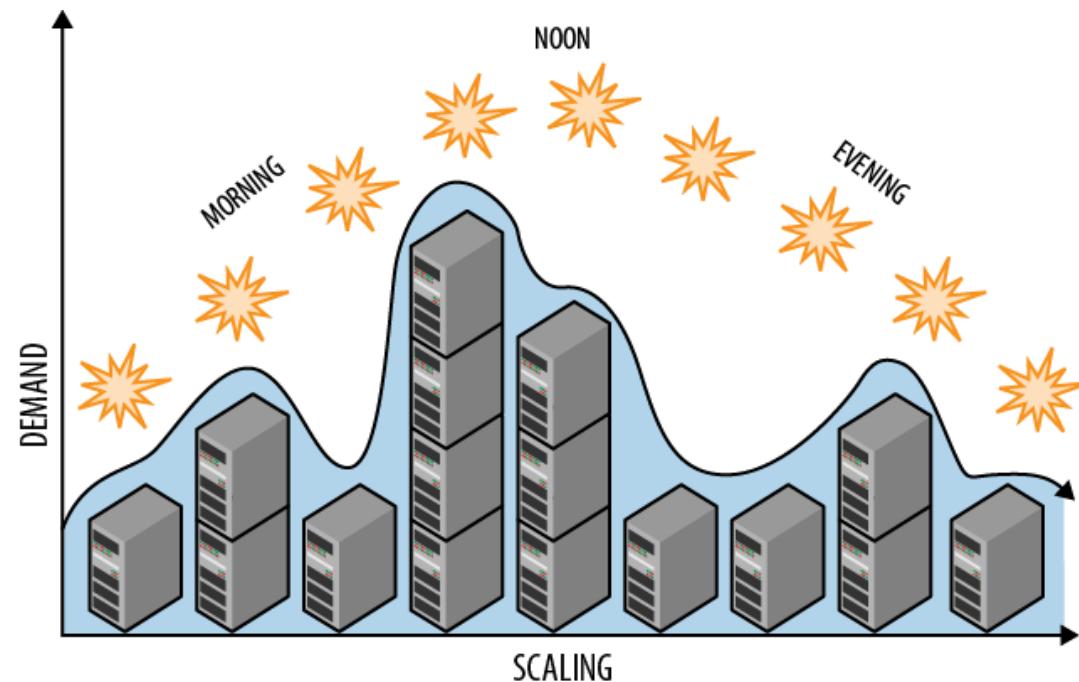
Узкие места

- 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 свои основные характеристики (производительность, удобство использования) при территориальном разнесении ее компонентов
- Отличия глобальных сетей от локальных
 - Заметное увеличение задержки и снижение пропускной способности
 - Связь менее надежна и стабильна, нет поддержки рассылки
- Плохо масштабируемые подходы
 - Клиент, вызывающий сервер, блокируется до получения ответа
 - Использование рассылки для обнаружения и координации процессов
- Пример: доставка контента (CDN)

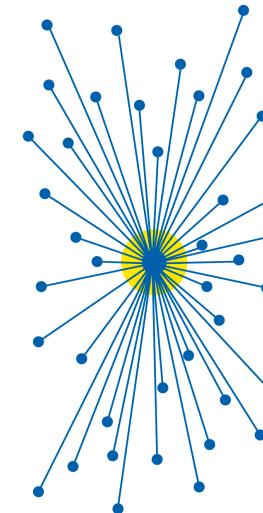
Времена ping

		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	

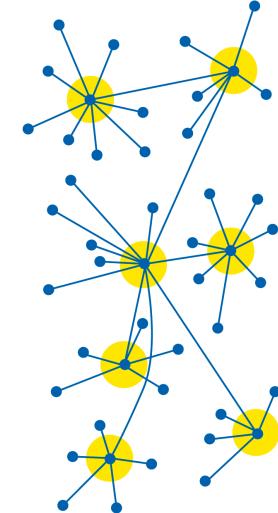
<https://wondernetwork.com/pings>

Административная масштабируемость

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



Centralized



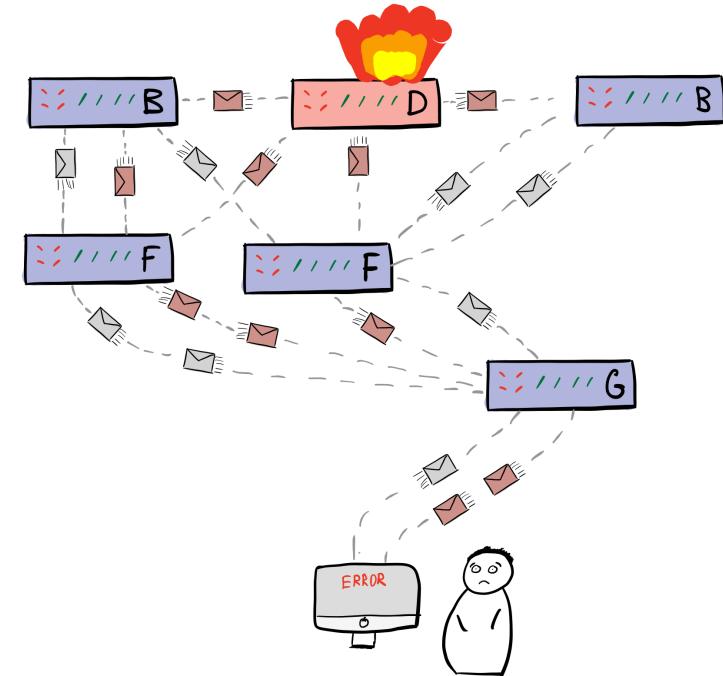
Decentralized

Отказоустойчивость (Fault-Tolerance)

Способность системы продолжать функционировать корректно в присутствии отказов компонентов

When you design distributed systems, you have to say, "Failure happens all the time." So when you design, you design for failure. It is your number one concern.

Ken Arnold. Designing Distributed Systems, 2002.



Пример

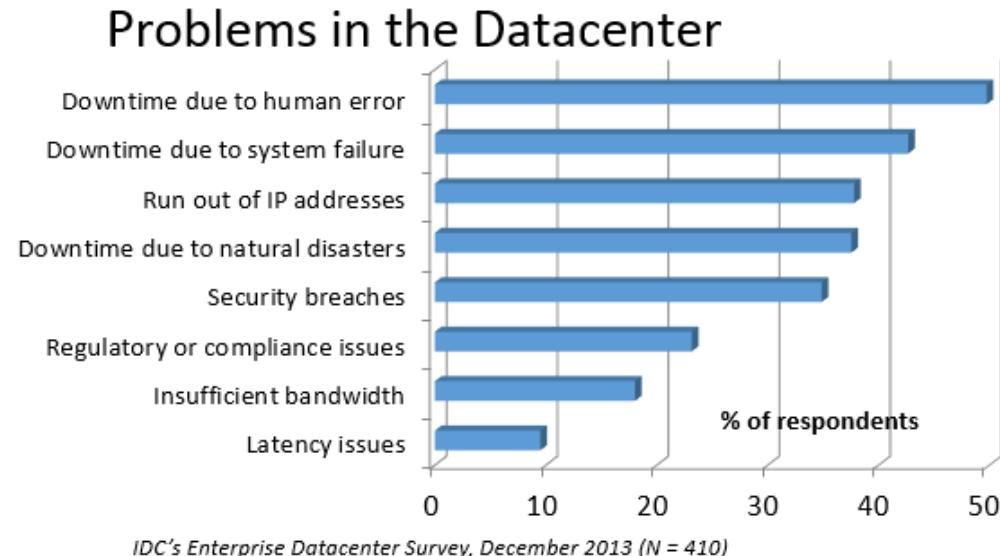
- Процесс А отправил запрос процессу В, но не получил ответа
- Что это значит?
 - Запрос потерялся и не дошел до В
 - Запрос дошел до В, но В пока не успел его обработать
 - Запрос дошел до В, но В упал, не успев обработать его
 - Запрос дошел до В, но В его просто проигнорировал
 - Запрос дошел до В 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

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

Обеспечение отказоустойчивости

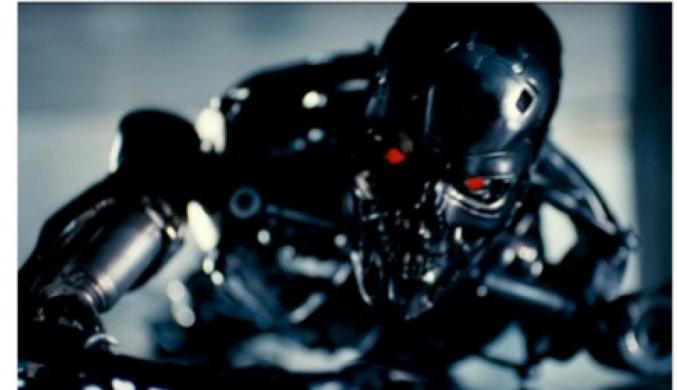
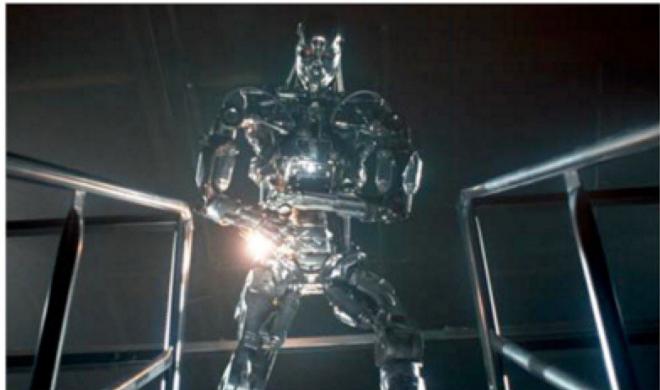
- Избыточность на аппаратном уровне
- Обнаружение и обработка отказов на программном уровне
- Прогнозирование и предотвращение отказов

Fault Tolerance vs Resilience

@ticofab
ticofab.io



“A fault-tolerant component is hit but keeps going, possibly with reduced functionality”

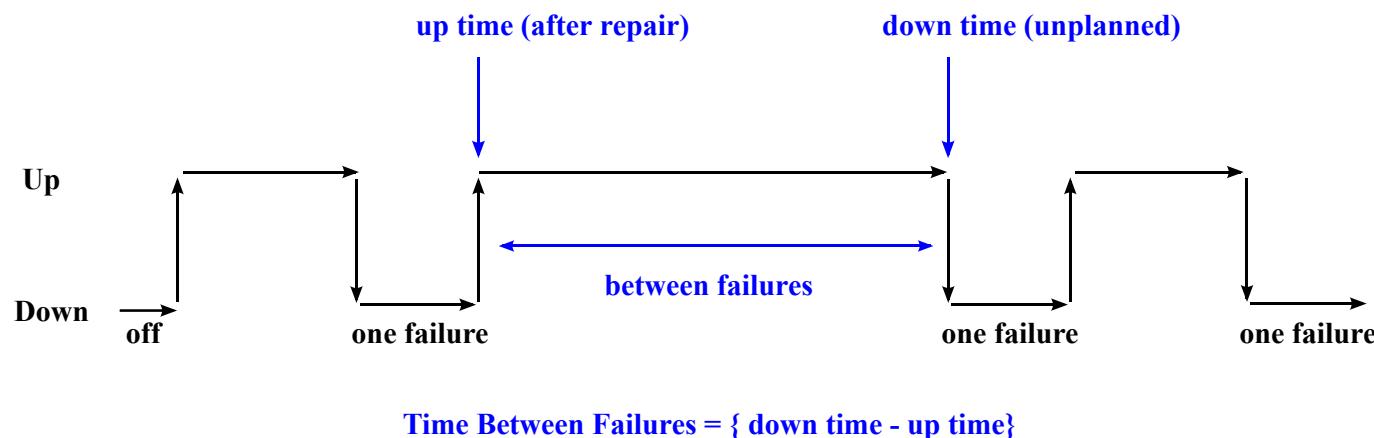


“Upon failure, a resilient component is able to jump back to a fully functional fresh state”



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)
- Сбой (fault) vs отказ (failure)



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 (например, 99% в год)
- Причины недоступности: отказы, ошибки, обновление ПО, технические работы...
- Важные факторы: времена перезапуска и восстановления после отказов

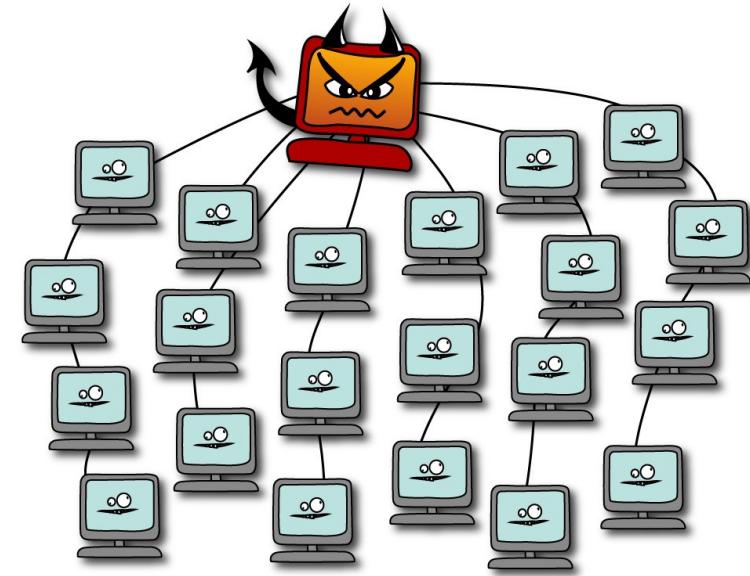


Удобство поддержки (Maintainability)

- 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 работу системы, используются ли хорошие абстракции?
- Можно ли адаптировать систему под меняющиеся требования?

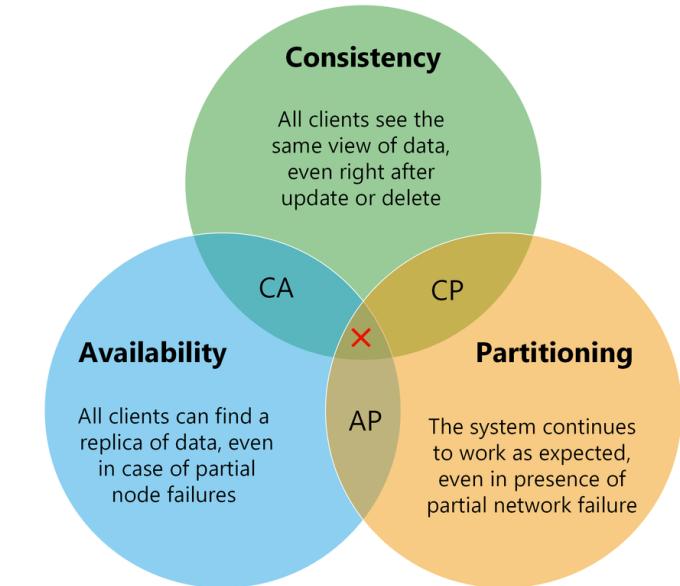
Безопасность

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



Согласованность данных (Consistency)

- Определенные гарантии при работе пользователя с хранимыми в системе данными
- Пример: при чтении значения по ключу К всегда возвращается последнее записанное по этому ключу значение
 - как обеспечить если в системе хранится несколько копий (реплик) данных?
- Компромисс между согласованностью, доступностью и масштабируемостью



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 или одиночной системой) крайне сложно
- Стоит ли скрывать распределенность системы от пользователей и разработчиков приложений?

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

Реализация распределенных систем

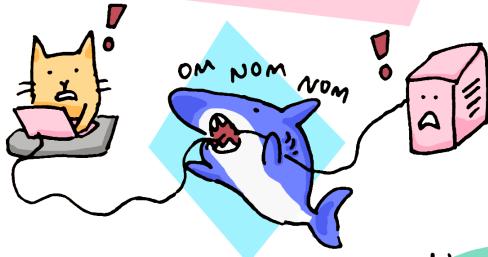
Distributed systems need radically different software than centralized systems do.

A. Tannenbaum

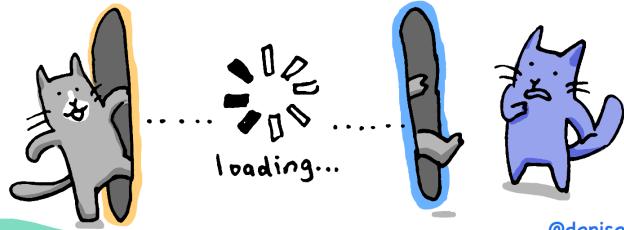
Особенности

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

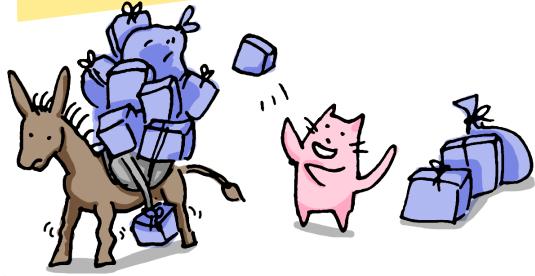
① The network is reliable



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

⑦ Transport costs \$0



⑥ There is only
one administrator



⑤ Topology doesn't
change



Source: <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)
- Организация обработки запросов на стороне сервера
- 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
- Координация процессов (взаимное исключение, выборы лидера, консенсус)
- Обеспечение безопасности

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