

БОСЕНКО ТИМУР МУРТАЗОВИЧ

# Распределенные системы

Институт цифрового образования

Москва, 2021г.

# Цель дисциплины

- Изучение принципов и технологий построения распределенных систем.
- Знакомство с различными классами распределенных систем и приложений.
- Получение практических навыков разработки распределенных приложений.

# Задачи дисциплины

- формирование практических навыков по основам распределенных систем информационных ресурсов;
- реализация требований, установленных в квалификационной характеристике в области распределенных систем информационных ресурсов.

# Планируемые результаты обучения

- Студент знает проблематику распределенных вычислений, области применения и виды распределенных систем.
- Студент знает принципы построения распределенных систем, модели программирования и методы решения типовых задач.
- Студент знает особенности различных классов распределенных систем и используемые при их реализации подходы.
- Студент владеет базовыми механизмами и технологиями разработки распределенных приложений.
- Студент умеет применять полученные знания на практике и обосновывать выбор метода решения поставленной задачи.



# Содержание учебной дисциплины

- Введение в распределенные вычисления.
- Сетевое взаимодействие, стек TCP/IP и сокет.
- Обмен сообщениями между процессами, протоколы и форматы данных.
- Промежуточное ПО, удаленные вызовы процедур и методов.
- Протокол HTTP, архитектурный стиль REST и веб-сервисы.
- Непрямое взаимодействие между процессами (очереди, publish-subscribe, мультикаст, общая память).
- Базовые механизмы (именование, поиск, обнаружение отказов, распространение информации).
- Обеспечение безопасности в распределенных системах.
- Распределенные системы хранения данных, репликация и согласованность данных.
- Консенсус и координация распределенных процессов.
- Децентрализованные распределенные системы, одноранговые сети, блокчейн.
- Параллельные вычисления, распределенные вычислительные системы и приложения.
- Распределенная обработка данных, MapReduce, модель dataflow.
- Облачные вычисления.
- Сервис-ориентированная архитектура, микросервисы.
- Современные практики и инструменты разработки распределенных систем и приложений.

# Оценка

- **ДЗ** - средняя оценка за домашние задания
- **ПР** - средняя оценка за проверочные работы(тестирование)

**Итоговая оценка =  $20 * \text{round}(0.7 * \text{ДЗ} + 0.3 * \text{ПР})$**



# **Формы контроля**

<b>100-балльная система оценки</b>	<b>Традиционная четырехбалльная система оценки</b>
<b>85 – 100 баллов</b>	оценка «отлично»/«зачтено»
<b>70 – 84 баллов</b>	оценка «хорошо»/«зачтено»
<b>50 – 69 баллов</b>	оценка «удовлетворительно»/«зачтено»
<b>менее 50 баллов</b>	Оценка «неудовлетворительно»/«незачтено»



# Литература

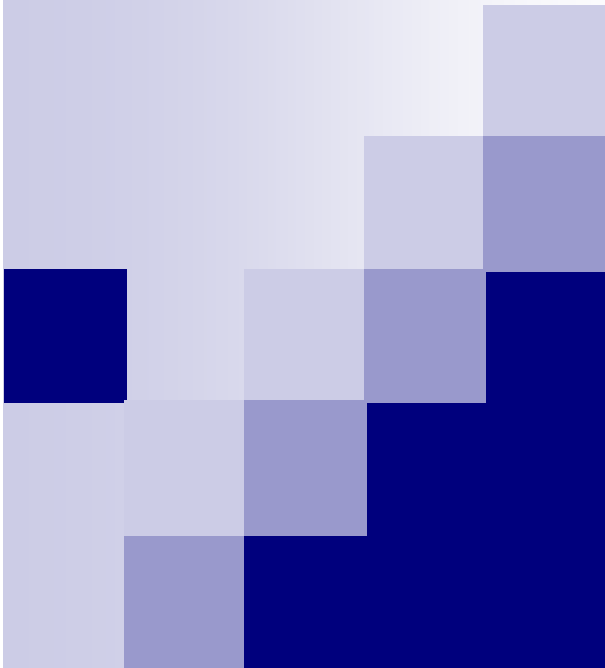
- Бабичев, Сергей Леонидович. Распределенные системы [Электронный ресурс] : учеб. пособие для вузов / С. Л. Бабичев, К. А. Коньков. – М. : Юрайт, 2020. – Добавлено: 17.08.2020. – Режим доступа: ЭБС Юрайт по паролю. - URL: <https://urait.ru/book/raspredelennye-sistemy-457005>
- Распределенные системы : учебное пособие для студентов, обучающихся по направлению подготовки 38.03.05 Бизнес-информатика / [авт.-сост. А.В. Демина, О.Н. Алексенцева]. – Саратов : Саратовский социально-экономический институт (филиал) РЭУ им. Г.В. Плеханова, 2018. – 108 с.
- Coulouris, G. F. (2012). Distributed Systems : Concepts and Design, Fifth Edition (Vol. Fifth edition, International edition). Harlow: Pearson Education. Retrieved from <http://search.ebscohost.com/login.aspx?direct=true&site=eds-live&db=edsebk&AN=1418002>
- Tanenbaum, A. S., & Steen, M. van. (2014). Distributed Systems: Pearson New International Edition : Principles and Paradigms (Vol. 2nd ed). Harlow, Essex: Pearson. Retrieved from <http://search.ebscohost.com/login.aspx?direct=true&site=eds-live&db=edsebk&AN=1418515>



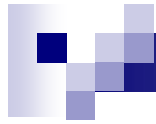


# Дополнительная литература

- Burns, B. (2018). Designing Distributed Systems : Patterns and Paradigms for Scalable, Reliable Services (Vol. First edition). Sebastopol, CA: O'Reilly Media. Retrieved from <http://search.ebscohost.com/login.aspx?direct=true&site=eds-live&db=edsebk&AN=1713745>
- Kleppmann, M. (2017). Designing Data-Intensive Applications : The Big Ideas Behind Reliable, Scalable, and Maintainable Systems. Sebastopol, CA: O'Reilly Media. Retrieved from <http://search.ebscohost.com/login.aspx?direct=true&site=eds-live&db=edsebk&AN=1487643>
- Petrov, A., & O'Reilly for Higher Education (Firm). (2019). Database Internals : A Deep Dive Into How Distributed Data Systems Work (Vol. First edition). Sebastopol, CA: O'Reilly Media. Retrieved from <http://search.ebscohost.com/login.aspx?direct=true&site=eds-live&db=edsebk&AN=2250514>



# Лекция 1. Распределенные системы



# Понятие распределенной системы

Не существует общепринятого и в то же время строгого определения распределенной системы.

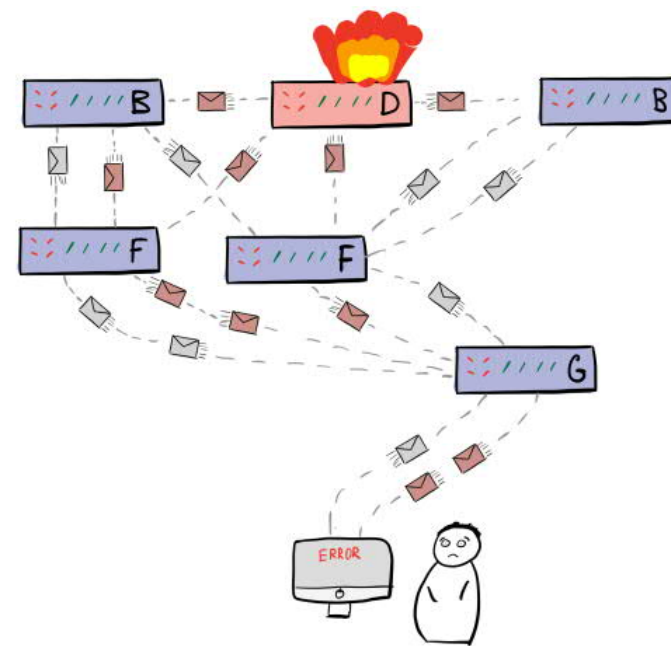
Э. Таненбаум

# Понятие распределенной системы

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)

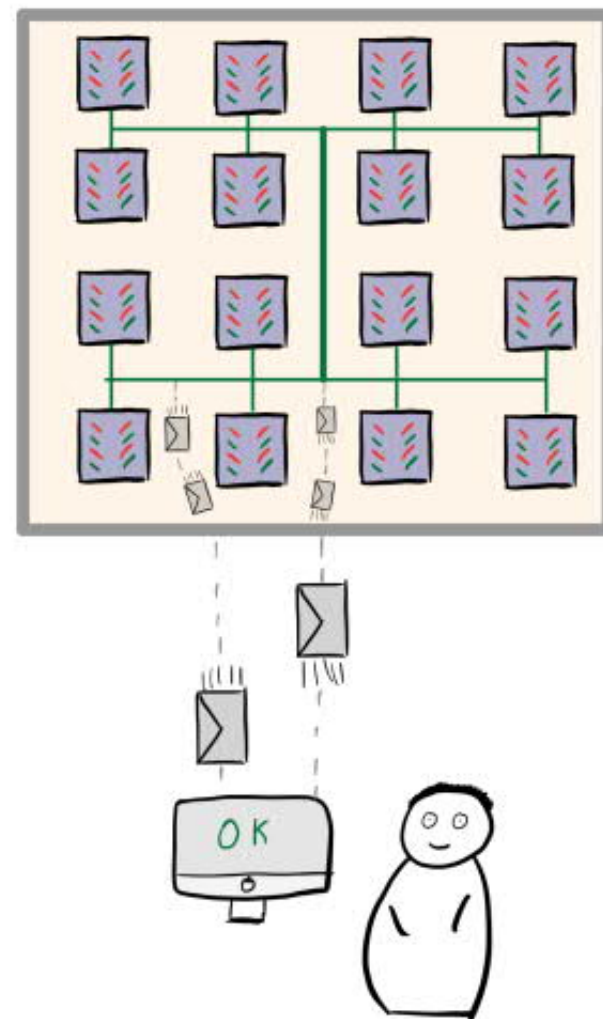
<https://www.microsoft.com/en-us/research/publication/distribution/>



# Понятие распределенной системы

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

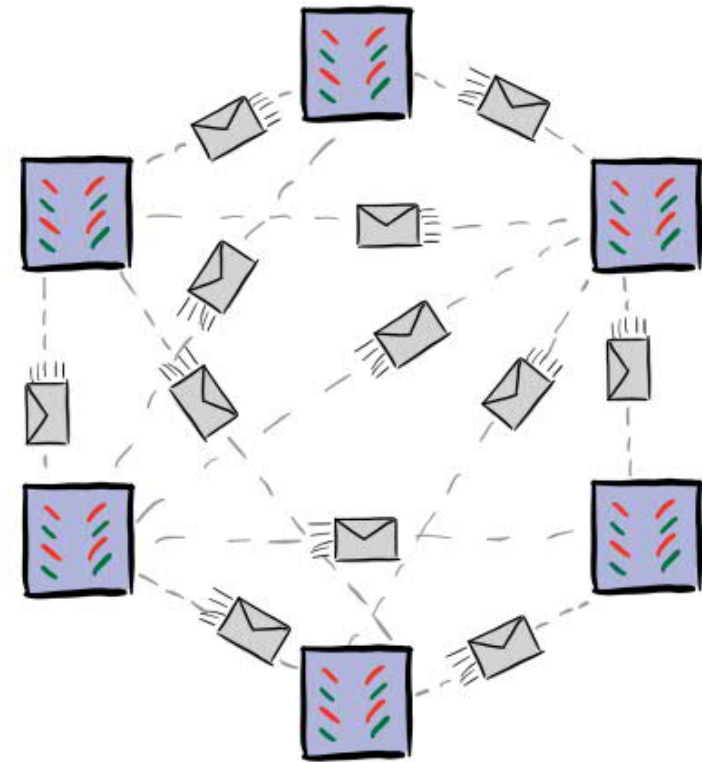
van Steen, Tanenbaum. Distributed Principles and Paradigms.



# Понятие распределенной системы

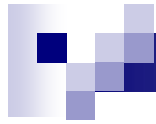
We define a distributed system as one in which hardware or software components located at networked computers communicate and coordinate their actions only by passing messages

Coulouris et al. Distributed Systems: Concepts and Design.



# Понятие распределенной системы

- Под **системой** понимается множество элементов и связей между ними.
- Обозначим  $V$  – множество элементов системы. Тогда бинарное отношение  $R_2 \subseteq V \times V$  задает наличие попарных связей между элементами. Если для некоторых элементов  $x \in V$  и  $y \in V$  пара  $(x, y) \in R_2$ , то в системе существует связь **от  $x$  к  $y$** . Если  $(x, y) \notin R_2$ , то такой связи нет.
- Порядок элементов в паре важен, так как связи могут быть направленными, несимметричными.



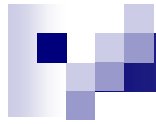
# Понятие распределенной системы

- В системе могут быть не только попарные связи, но и связи троек элементов. Такие связи описываются тернарными отношениями  $R_3 \subseteq V \times V \times V = V^3$ .
- Например, связь «ребенок и его родители» — связь трех элементов.



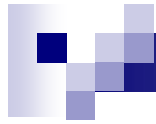
# Понятие распределенной системы

- С каждым отношением связан определенный смысл, выражаемый высказыванием, например, « $x$  следует за  $y$ », « $x$  посылает сигнал  $y$ », « $x$  – ребенок  $y$  и  $z$ » и т.д.



# Понятие распределенной системы

- Иначе говоря, вместо отношений (или вместе с отношениями) удобно рассматривать соответствующие **предикаты**  $P_2, P_3, P_4, \dots, P_n$ .
- В дополнение к перечисленным рассматривают и **предикаты**  $P_1$ , которые можно интерпретировать как **выражение свойств элементов множества  $V$** .

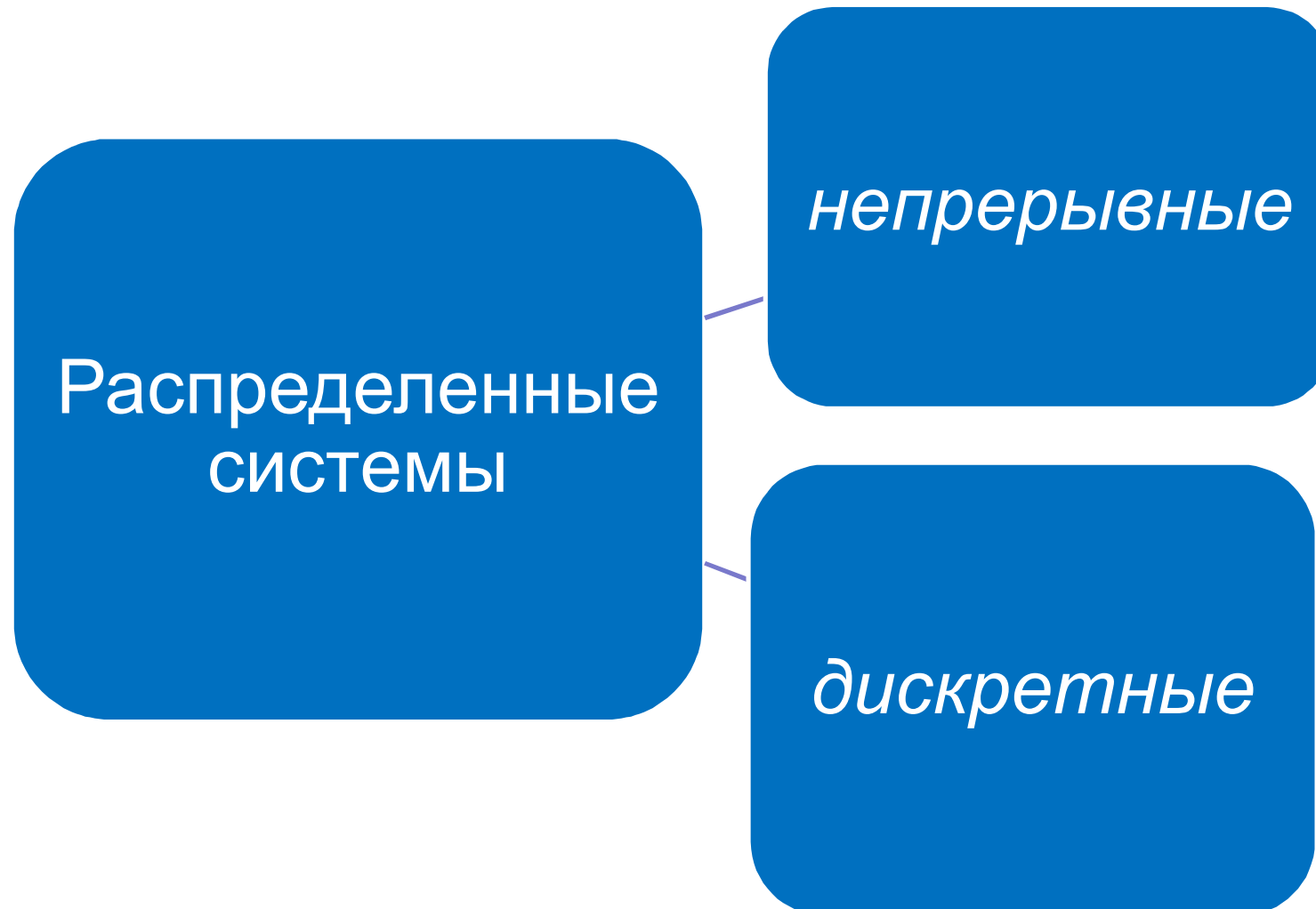


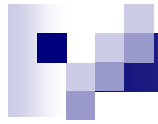
# Понятие распределенной системы

■ *Распределенными системами* будем называть такие системы, для которых **предикаты** местоположения элементов или групп элементов играют существенную роль с точки зрения функционирования системы, а, следовательно, и с точки зрения анализа и синтеза системы.



# Понятие распределенной системы

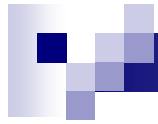




# Понятие распределенной системы

*Непрерывные*

характеризуются  
бесконечным количеством  
элементов, а также тем, что  
в любой малой окрестности  
любого элемента (в смысле  
предикатов  
местоположения)  
находится, по крайней  
мере, еще один элемент.



# Понятие распределенной системы

Дискретные

элементы системы четко «очерчены», отделены друг от друга. Между двумя соседними элементами других элементов нет.

# Примеры распределенных систем.

## Газопроводы России

Единая система газоснабжения (ЕСГ) России – крупнейшая в мире система транспортировки газа



В состав ЕСГ входят:

Россия занимает 1-е место в мире

## ■ Сеть газопроводов.

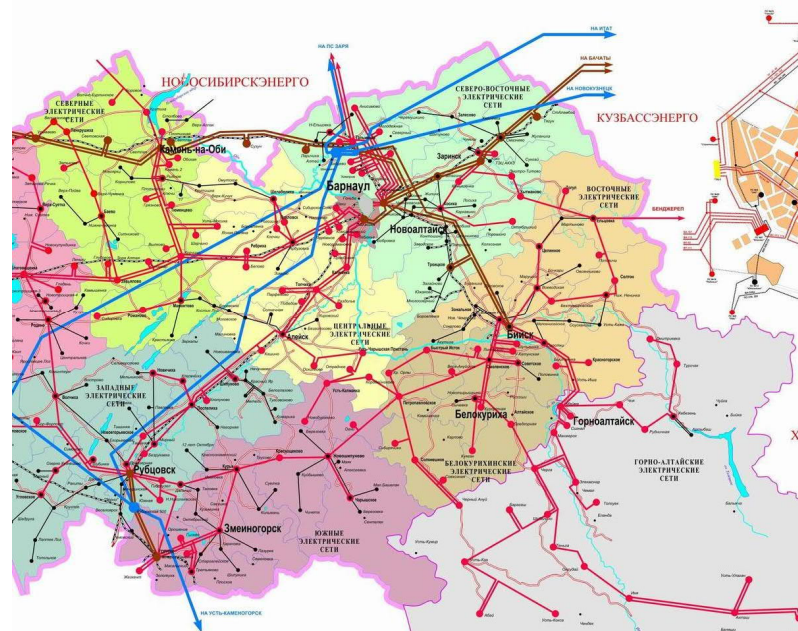
■ Сеть газопроводов состоит из газоперекачивающих станций, газораспределительных узлов, трубопроводов. Трубопроводы соединяют между собой станции и узлы. Трубопроводы проложены также к источникам (месторождениям) газа и к потребителям газа.

■ Основное отношение в системе – это соединение (направленное) элементов: соединение трубопровода и газоперекачивающей станции, соединение трубопровода и газораспределительного узла.

# Примеры распределенных систем.

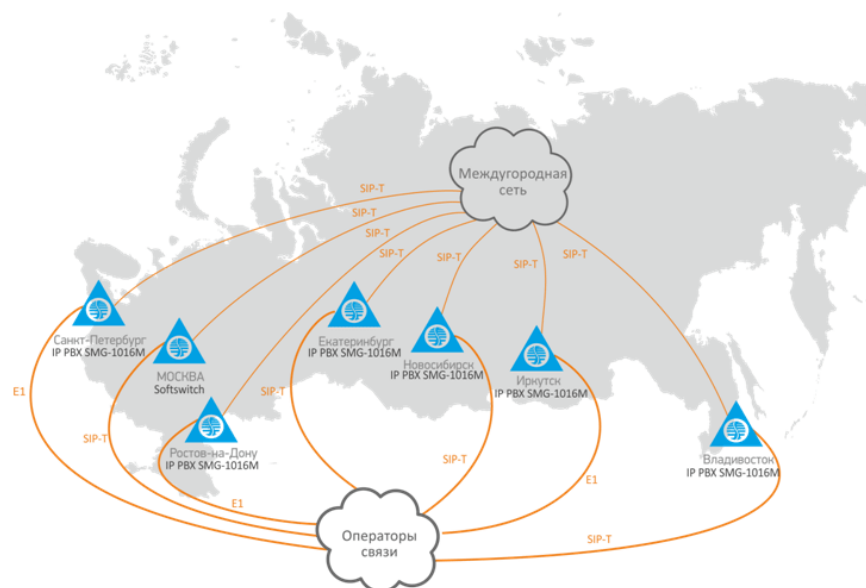
## ■ Электросети.

■ Электросеть включает линии электропередач различного напряжения и трансформаторные подстанции. Электросеть соединена с «генерирующими объектами» - ГЭС, теплоэлектростанциями, АЭС, и с потребителями. Трансформаторные подстанции, генерирующие объекты, потребители имеют географические координаты. Линии электропередач характеризуются плоскими кривыми





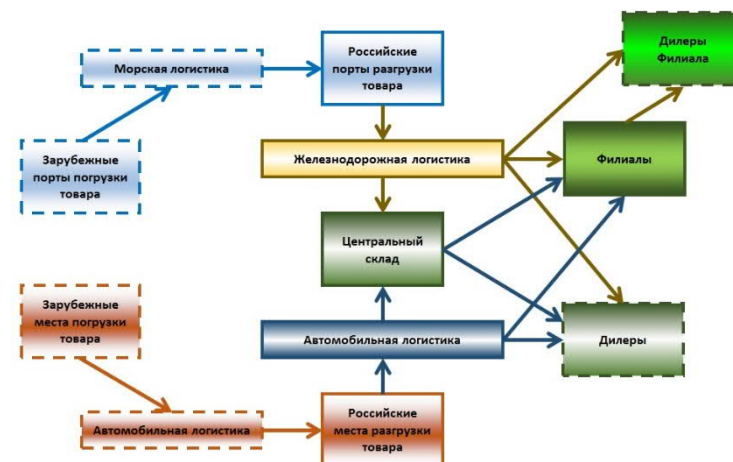
# Примеры распределенных систем.



## ■ Сети связи.

- Каждый узел сети характеризуется скоростью обработки информации. В сетях связи – это скорость коммутации (переключения) каналов связи. В компьютерных сетях в узле сети может находиться сервер, который выполняет определенное обслуживание по поступающим запросам. Время этого обслуживания – дополнительная характеристика узла сети.
- Каналы связи, соединяющие узлы, различаются своей пропускной способностью – скоростью передачи информации и иными техническими характеристиками (оптоволоконные, кабельные, спутниковые каналы, радиоканалы и др.).

# Примеры распределенных систем.



## ■ Логистические системы.

- Система перевозки грузов включает транспортные средства (грузовики, поезда, самолеты, корабли), транспортные пути (автомобильные, железные дороги, морские и речные пути и др.), склады, порты, станции, погрузочно-разгрузочные механизмы и проч.
- При транспортировке конкретного груза от поставщика к получателю логистическая цепочка может включать довольно большое количество элементов системы, в том числе многочисленные перегрузки с одного вида транспорта на другой.

# Примеры распределенных систем.

## ■ Банковская система.

■ Банковская система распределена по всему Земному шару. Но у всех банков есть общая задача – обслужить клиента, где бы он ни был. Клиент должен иметь возможность получить свои деньги даже там, где о его банке никогда не слышали. Или сделать банковский перевод в далекую страну. Для осуществления этих операций банки связаны между собой системой договоров и поддерживающими эту систему техническими средствами.



# Примеры распределенных систем.

## ■ Корпорации.

- Крупные фирмы (и даже не очень крупные) имеют офисы и производства, рассредоточенные в пределах города, страны и даже всей планеты. Может быть, это отсутствие достаточных помещений в одном месте или другие подобные факторы.
- Но есть и совершенно объективные причины создания удаленных подразделений (отделений) корпораций. Прежде всего, это приближение к рынку сбыта продукции.





# Сосредоточенные и распределенные системы

- Существуют (или могут существовать) системы, решающие одинаковые задачи, системы, функционально эквивалентные, но конструктивно различные. Обозначим две такие системы  $S_d$  и  $S_{sa}$  (от английских терминов **distributed** и **stand-alone**).

# Сосредоточенные и распределенные системы

- Множество элементов в каждой из систем, обычно, можно разделить на два подмножества,

- $V(S_d) = U_d \cup W_d$ ,

- $V(S_{sa}) = U_{sa} \cup W_{sa}$ .

- Множества, обозначенные буквой  $U$ , состоят из сосредоточенных элементов, занимающих относительно небольшой объем пространства и **реализующих некоторую функцию преобразования.**

# Сосредоточенные и распределенные системы

- $V(S_d) = U_d \cup W_d$ ,

- $V(S_{sa}) = U_{sa} \cup W_{sa}$ .

- Множества, обозначенные буквой **W**, состоят из элементов, связывающих некоторые сосредоточенные элементы между собой.

- Их основная задача **не преобразование, а передача** чего-либо в системе от одного элемента к другому



# ПРИМЕР РАСПРЕДЕЛЕННОЙ СИСТЕМЫ

- $V(S_d) = U_d \cup W_d$ ,

- **Например**, сосредоточенными элементами могут быть **два компьютера**, осуществляющих вычисления по некоторым алгоритмам. **Связывающим элементом может быть кабель**, один конец которого подсоединен к первому компьютеру, а второй конец — ко второму компьютеру.

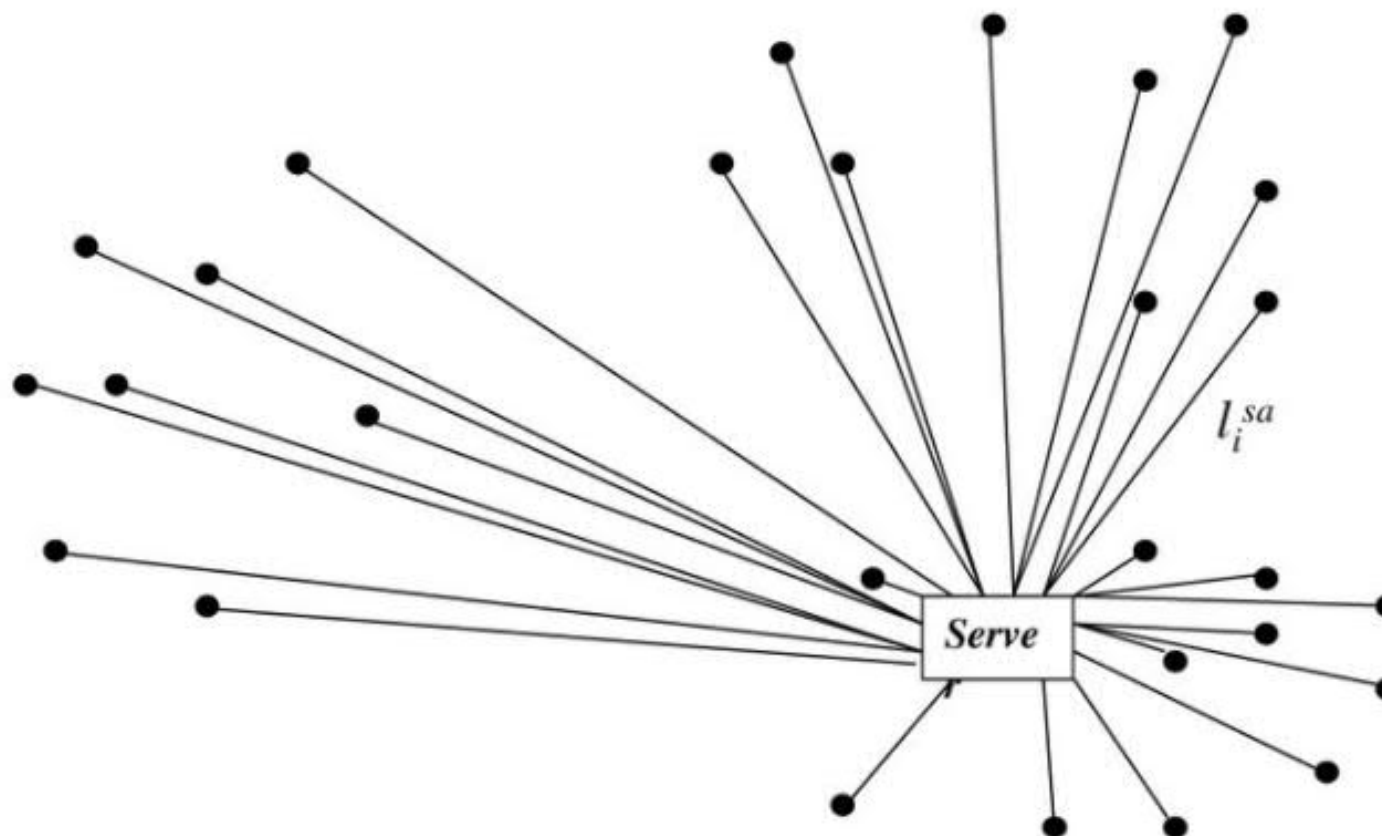


# ПРИМЕР РАСПРЕДЕЛЕННОЙ СИСТЕМЫ

- $V(S_d) = U_d \cup W_d$ ,
- Часто элементы из множества  $W$  имеют определенную «протяженность» в пространстве, например, кабель или витая пара имеют параметр «длина». Величина этого параметра существенным образом влияет на функционирование системы.
- Таким образом, элементы из множества  $W_d$  могут быть весьма разнообразными, с большим количеством характеристик.



# ПРИМЕР РАСПРЕДЕЛЕННОЙ СИСТЕМЫ



Сосредоточенная система

# ПРИМЕР СОСРЕДОТОЧЕННОЙ СИСТЕМЫ

- $V(S_{sa}) = U_{sa} \cup W_{sa} .$

- Напротив, в сосредоточенных системах элементы из множества  $W_{sa}$  описываются просто, а часто вообще исключаются из рассмотрения как несущественные для анализа свойств системы.

- Таким элементом, например, в компьютере является набор проводников, соединяющих процессор с микросхемами памяти.

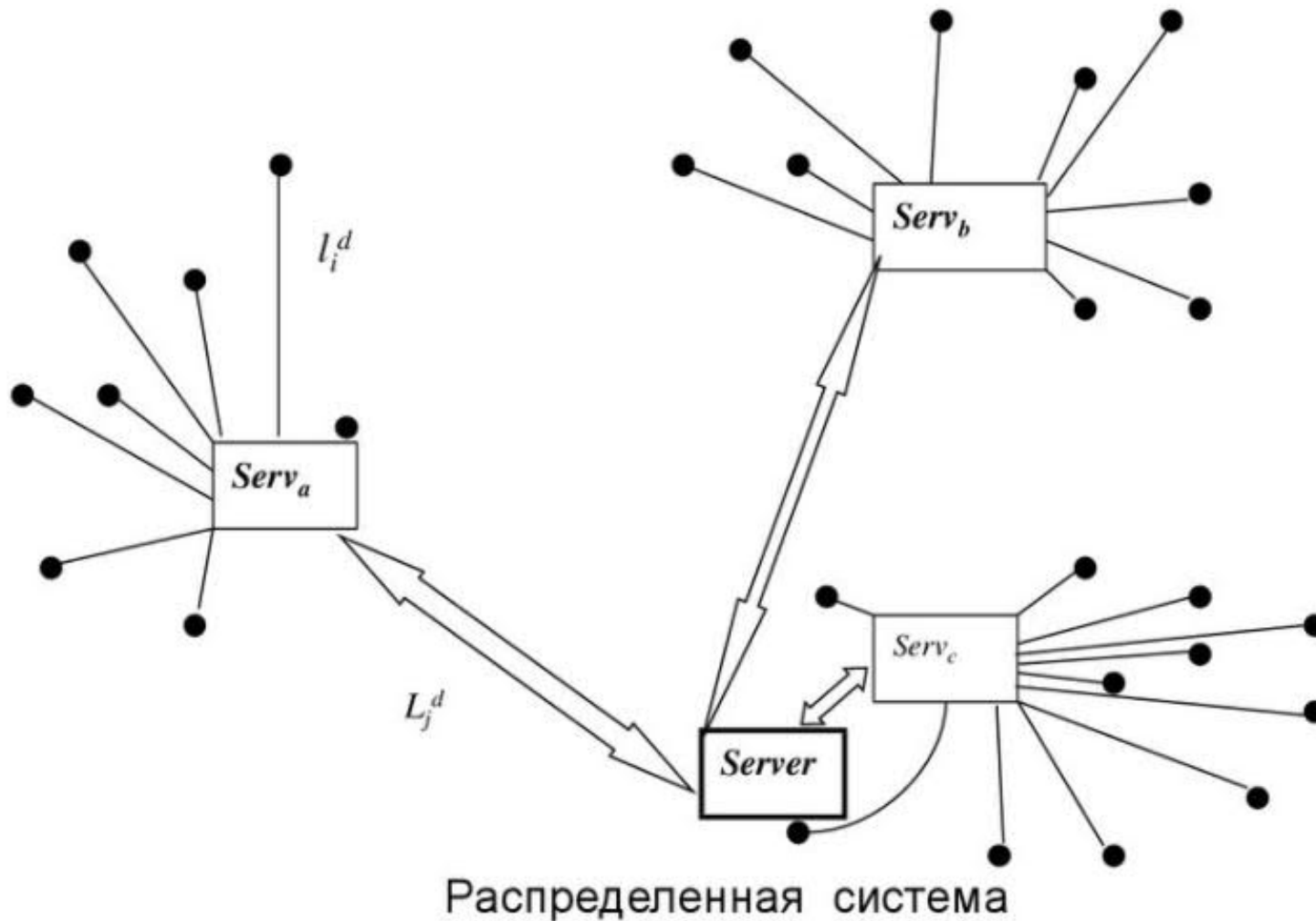
# Сосредоточенные и распределенные системы

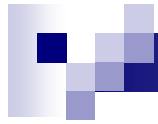
■ Множества сосредоточенных элементов  $U_{sa}$  и  $U_d$  сосредоточенной и распределенной систем также могут отличаться.

■ **В сосредоточенной системе**  
функционирование элементов из  $U_{sa}$   
инвариантно их местоположению.

■ **В распределенной системе**  
функционирование элементов из  $U_d$  в общем  
случае зависит от их местоположения.

# Сосредоточенные и распределенные системы





# Определение

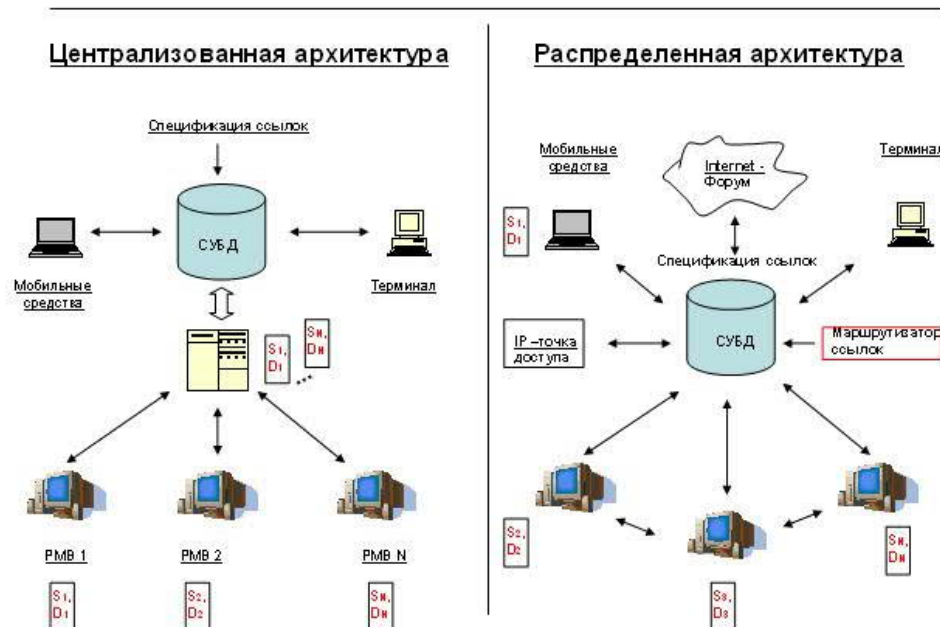
- A **program** is the code you write
- A **process** is what you get when you run it
- A **message** is used to communicate between processes
- A **packet** is a fragment of a message that might travel on a wire
- A **protocol** is a formal description of message formats and the rules that processes 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

# 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



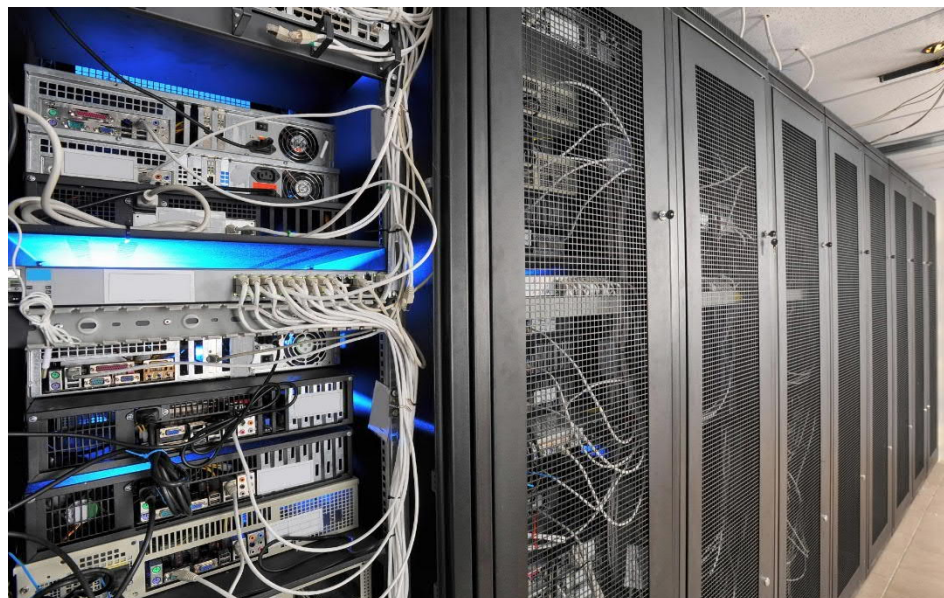
# Применение распределенных систем

## 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

## Совместное использование ресурсов

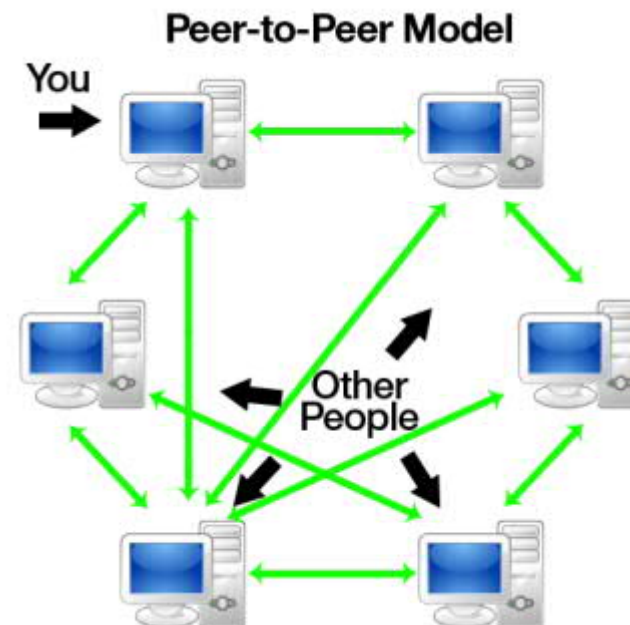
- ✓ Клиент-сервер, peer-to-peer, вычислительный кластер
- ✓ Поддерживать единую систему дешевле, чем множество независимых

## Коммуникация и координация

- ✓ Пользователи и узлы географически распределены

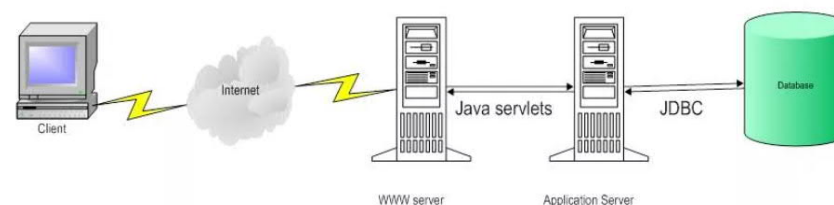
## Уменьшение задержки при обслуживании географически распределенных пользователей

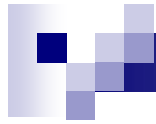
- ✓ Размещение данных как можно ближе к пользователям



# 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

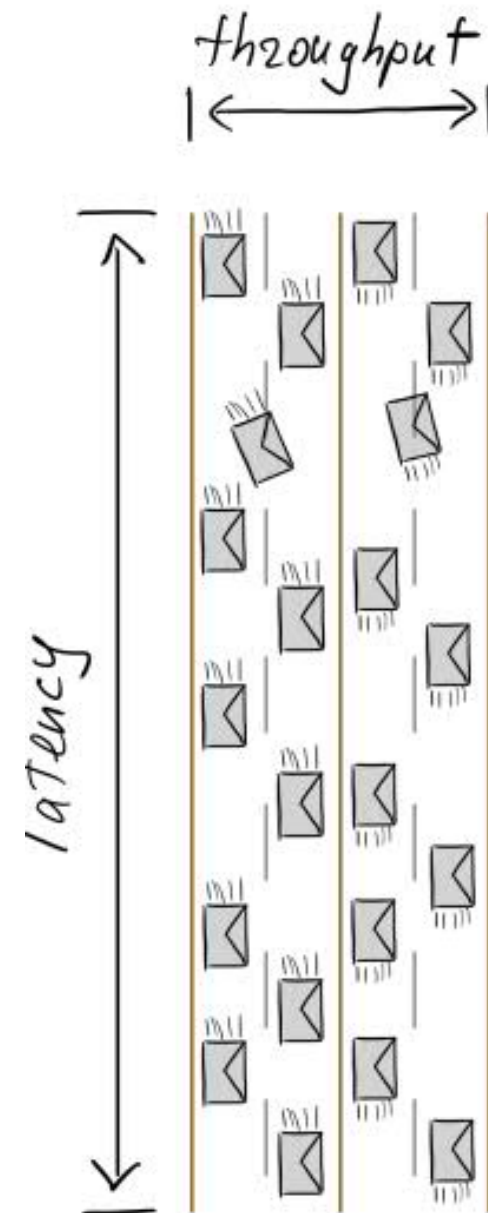
# Производительность (Performance)

## ➤ Основные показатели

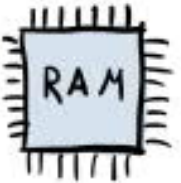
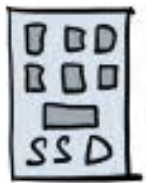



- Задержка, время обработки запроса, время ожидания ответа
- Пропускная способность, число обрабатываемых запросов/данных в секунду
- Качество обслуживания, битрейт, доля пропущенных кадров потокового видео

## ➤ Могут конфликтовать друг с другом

- В разных системах может отдаваться приоритет разным показателям



# Как оценить производительность?

Type	 RAM	 SSD	 HDD	 Data center	
Latency	100 ns	10 $\mu$ s	1 ms	10 ms	100 ms
Throughput	100 GB/s	1 GB/s	100 MB/s	1 GB/s	10 MB/s

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

# Как оценить производительность?

coggle

made for free at coggle.it



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



# Основные типы тестирования и вопросы, которые они решают

№	Вид тестирования	Вид тестирования по английский	Вопрос на который отвечает тестирование
1	Нагрузочное тестирование	Load Testing	Достаточно ли быстро работает система?
2	Тестирование стабильности	Stability Testing	Достаточно ли надежно работает система на долгом интервале времени?
3	Тестирование отказоустойчивости	Failover Testing	Сможет ли система переместиться сама на другой сервер в случае сбоя основного сервера?
4	Тестирование восстановления	Recovery Testing	Как быстро восстановится система?
5	Стрессовое тестирование	Stress Testing	Что произойдет при незапланированной нагрузке?
6	Тестирование объемов	Volume Testing	Как будет работать система, если объем базы данных увеличится в 100 раз?
7	Тестирование масштабируемости	Scalability Testing	Как будет увеличиться нагрузка на компоненты системы при увеличении числа пользователей?
8	Тестирование потенциальных возможностей	Capacity Testing]	Какое количество пользователей может работать?
9	Конфигурационное тестирование	Configuration Testing	Как заставить систему работать быстрее?
10	Тестирование сравнения	Compare Testing	Какое оборудование и ПО выбрать?

# 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



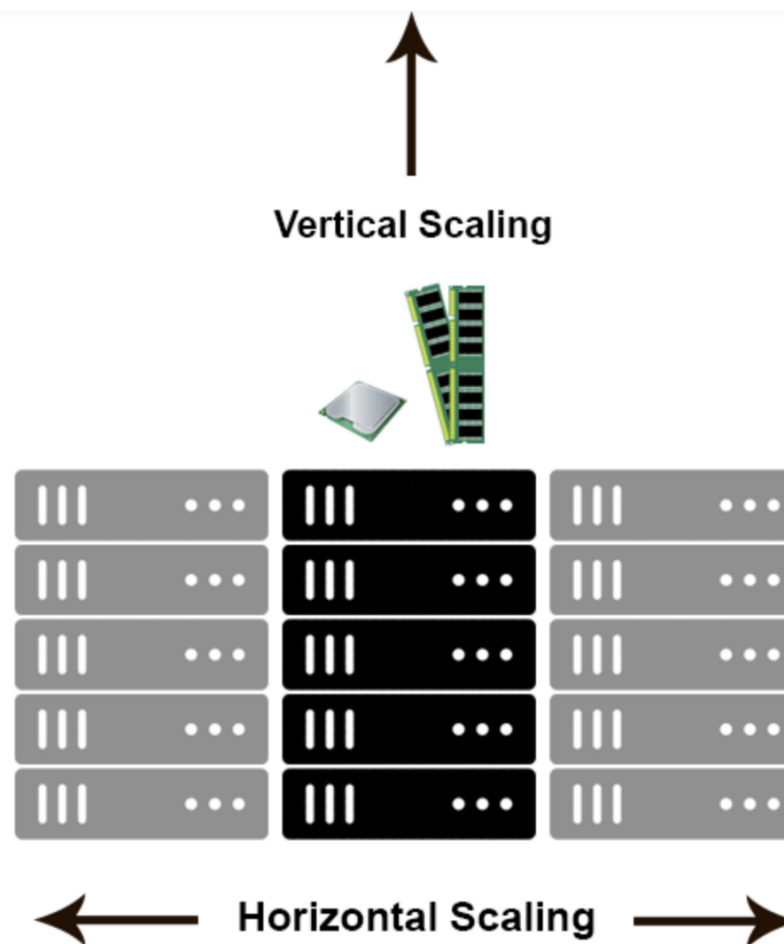


# Нагрузочная масштабируемость

- **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

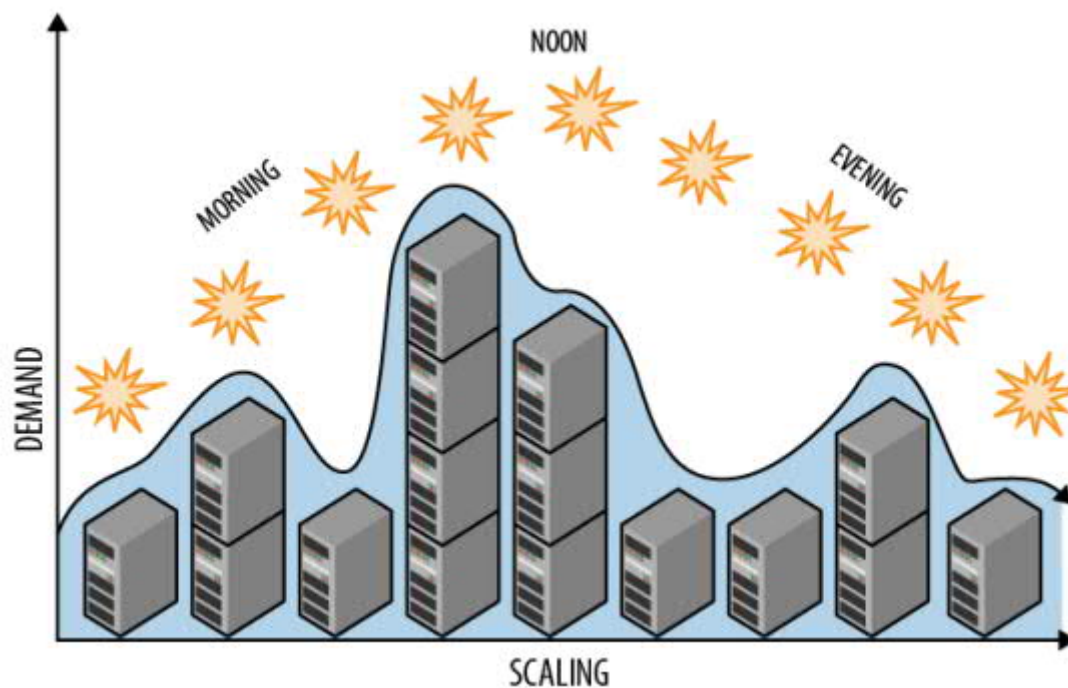


# Горизонтальное масштабирование

- **Наращивание аппаратных средств путем добавления в систему новых узлов**



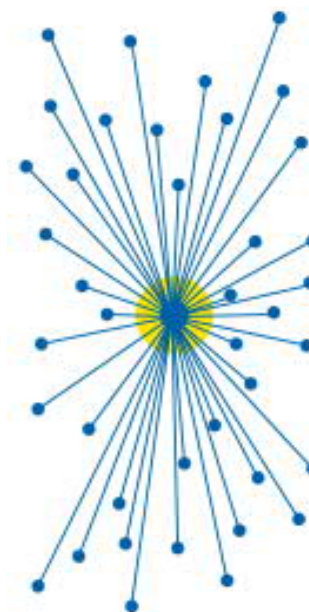
# Эластичность



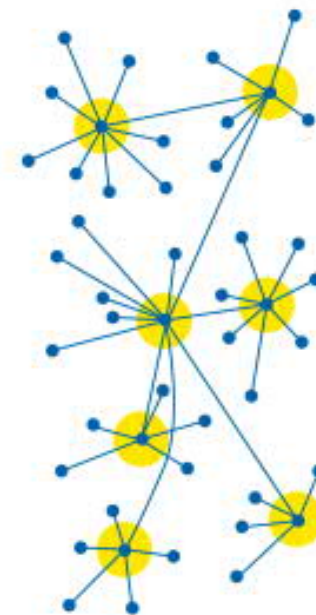
- **Автоматическое масштабирование ресурсов под текущую нагрузку**

# 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

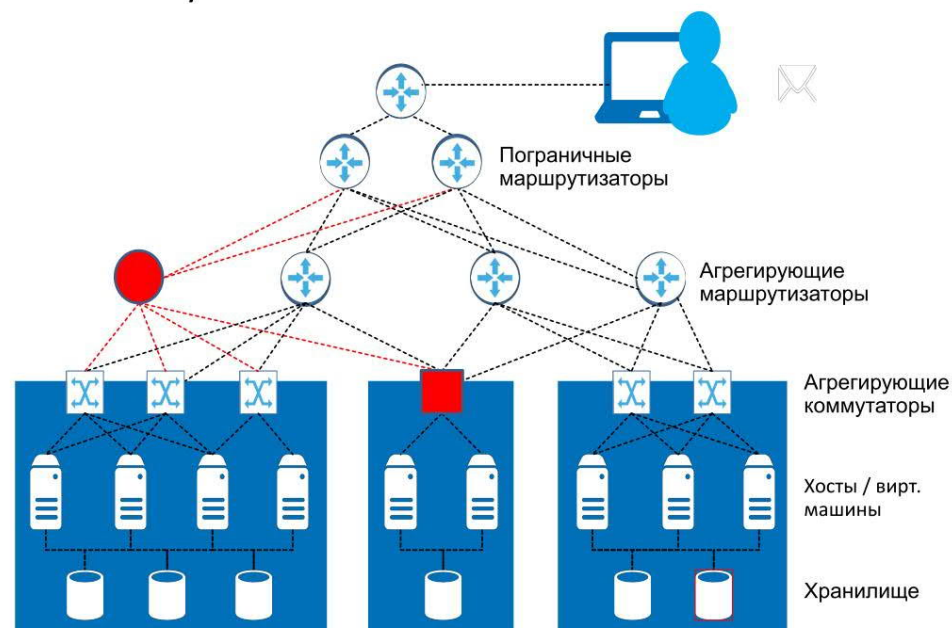


Decentralized

# Отказоустойчивость (Fault-Tolerance)

- Способность системы продолжать функционировать корректно в присутствии отказов компонентов

Отказоустойчивость и Fault Domain



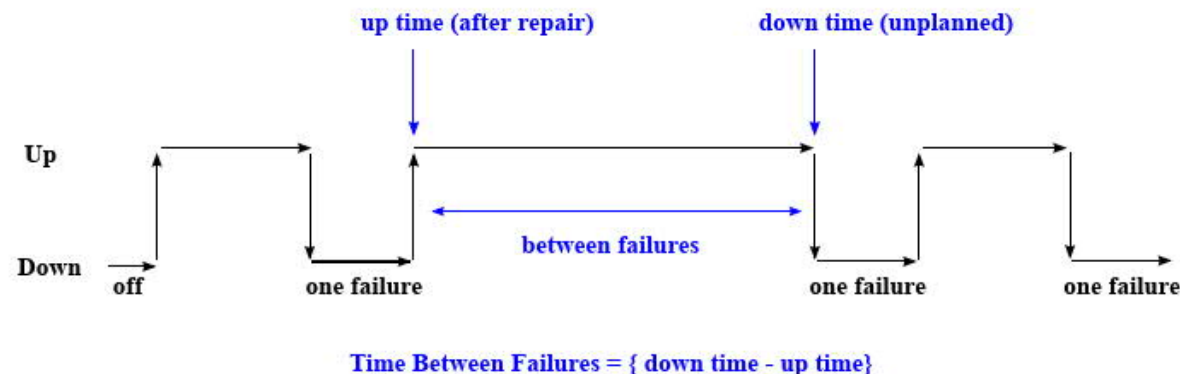
# Отказоустойчивость (Fault-Tolerance)

## ➤ Типичные отказы Data center

Problem Type	Example causes or explanations
Change	Device deployment, UPS maintenance
Incident	Network Connection
Network Connection	OSPF convergence, UDLD errors, Cabling, Carrier signaling/timing issues
Software	IOS hotfixes, BIOS upgrade
Hardware	Power supply/fan, Replacement of line card/chassis/optical adapter
Configuration	VPN tunneling, Primary-backup failover, IP/MPLS routing

# 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)





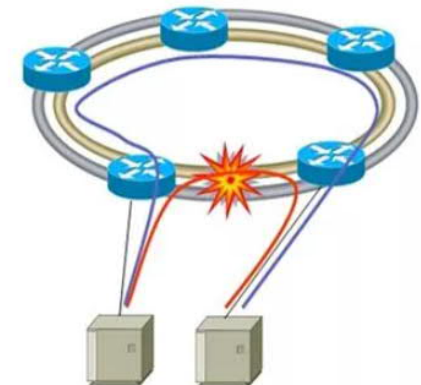
# Доступность (Availability)

- Система доступна, когда пользователи могут взаимодействовать с системой, получать требуемые сервисы, корректные ответы.
- 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

## Доступность (Д)

Обеспечение доступа к информации, когда это необходимо.

- Резервирование
- SLA: 24/7, 99.9%



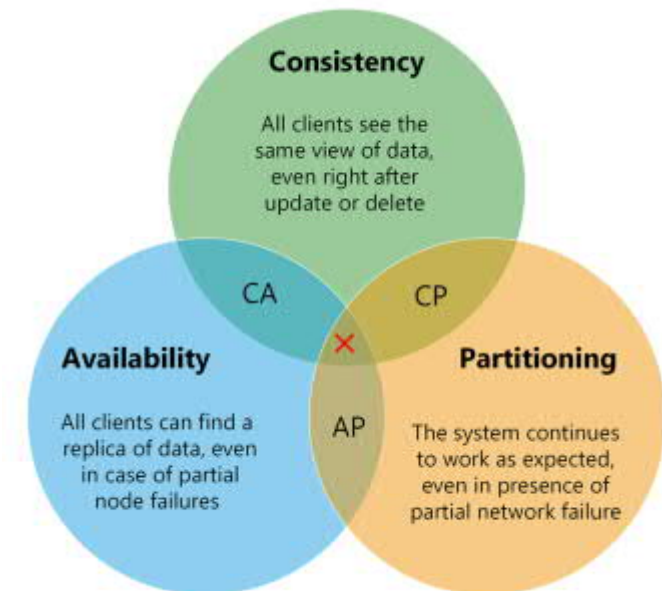


# Безопасность

- 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 K всегда возвращается последнее записанное по этому ключу значение
  - как обеспечить если в системе хранится несколько копий (реплик) данных?
- Компромисс между согласованностью, доступностью и масштабируемостью



# Реализация распределенных систем

## Особенности

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

# Распределенные задачи и алгоритмы

- Распределенная система порождает **распределенную задачу**, поскольку исходные данные для задачи возникают в различных точках пространства. Также в различных точках требуются те или иные результаты решения задачи. Зачастую, задачу можно разбить на совокупность подзадач, некоторые из них могут быть сосредоточенными: все исходные данные возникают в одной точке пространства, и там же требуются результаты решения подзадачи.
- Распределенная задача решается **распределенным алгоритмом**, собирающим исходные данные из разных точек и посылающим результаты расчетов в разные точки.

# Распределенные задачи и алгоритмы

- Один из видов распределенных алгоритмов – **протоколы**.  
Протокол характеризуется тем, что имеется как минимум две стороны, разделенные каналом связи. На каждой из сторон выполняется локальный (сосредоточенный) алгоритм  $A_k$ .
- Локальный алгоритм  $A_1$  выполняется до некоторого момента времени, когда для продолжения работы ему требуются данные от другого локального алгоритма  $A_2$ . Он посылает через линию связи запрос на данные локальному алгоритму  $A_2$ . Алгоритм  $A_2$  отвечает, пересылая сообщение по линии связи. После этого локальный алгоритм  $A_1$  продолжает свою работу.
- Протоколы, обычно, играют техническую роль и служат для установления режимов приема/передачи данных между удаленными объектами. В вычислительном отношении локальные алгоритмы – части протокола – не являются сложными.

# Распределенные задачи и алгоритмы

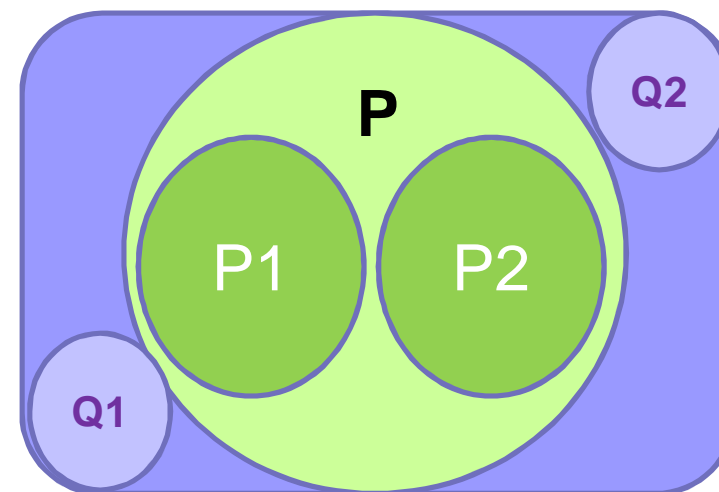
## *Криптографические протоколы*

- Особый вид распределенных алгоритмов – ***криптографические протоколы***. Они предназначены для доказательства одной или обеими сторонами, что они именно те, за кого себя выдают, для обмена конфиденциальной информацией и других подобных целей.
- Рассмотрим один из простейших криптографических протоколов – распределенных алгоритмов.

# Распределенные задачи и алгоритмы

## Криптографические протоколы

- Предположим, что в разных точках пространства находятся два объекта,  $O_1$  и  $O_2$ , каждый из которых решает свою часть ( $P_1$  и  $P_2$ ) общей задачи  $P$ . Вместе с тем, кроме стремления решить общую задачу, объекты имеют и свои частные задачи,  $Q_1$  и  $Q_2$ .
- Таким образом, объект  $O_1$  решает одновременно задачи  $P_1$  и  $Q_1$ , а объект  $O_2$  — задачи  $P_2$  и  $Q_2$ . И, если задачи  $P_1$  и  $P_2$  совместимы и взаимно дополняют друг друга, то задачи  $Q_1$  и  $Q_2$  — противоречат друг другу.
- Следовательно, объекты  $O_1$  и  $O_2$  сотрудничают, но, одновременно, и конкурируют. А можно сказать и, что объекты конкурируют, но вынуждены сотрудничать. Смотря как расставить акценты, какая из задач,  $P_i$  или  $Q_i$ , важнее для объекта  $O_i$ .





# Распределенные задачи и алгоритмы

## *Криптографические протоколы*

- В нашей задаче объекты  $O_1$  и  $O_2$  должны прийти к общему компромиссному решению в интересах решения задачи  $P$ .
- Предположим, что имеется два равноценных (с точки зрения задачи  $P$ ) решения. Одно из них также устраивает объект  $O_1$ , а другое – устраивает объект  $O_2$ . Тогда объекты  $O_1$  и  $O_2$  решают бросить жребий, однако, бросить жребий и «увидеть» результат, находясь на большом расстоянии – весьма затруднительно.
- Процедуру мог бы выполнить объект  $O_1$  и сообщить результат объекту  $O_2$ . Но объект  $O_2$  не вполне доверяет объекту  $O_1$ .
- Выход состоит в следующем:

# Распределенные задачи и алгоритмы

## *Криптографические протоколы*

- Обозначим одно из решений числом  $0$ , другое – числом  $1$ . Каждый из объектов независимо от другого должен назвать какое-нибудь целое число  $n_i$  в пределах, например, от  $0$  до  $99$ . Затем объекты обмениваются числами  $n_i$  и вычисляют результат  $(n_1 + n_2) \pmod{2}$ . Это и будет решение, т.е. число  $0$  или  $1$ .
- Абсолютно одновременно переслать друг другу числа  $n_i$  объекты не могут. Кто-то пришлет свое число первым и окажется в невыигрышном положении. Например,  $O_1$  переслал свое число  $n_1$  объекту  $O_2$ , заинтересованному в том, чтобы выиграло решение «1». Тогда  $O_2$ , зная  $n_1$ , решает уравнение  $(n_1 + n_2) \pmod{2} = 1$  относительно переменной  $n_2$ , и посылает  $O_1$  найденное значение  $n_2$ . Решение уравнения практически не требует времени: получив от  $O_1$  четное число,  $O_2$  должен ответить нечетным, и наоборот.

# Распределенные задачи и алгоритмы

## *Криптографические протоколы*

- Эта несправедливость должна быть устранена. Ясно, что какая-то из сторон обмена сообщениями первой пришлет свое число.
- Необходимо сделать так, чтобы у второй стороны не хватило времени на «подбор ответа».
- Выход:
  - Объекты  $O_1$  и  $O_2$  могут договориться, что вся процедура «бросания жребия» на расстоянии должна завершиться за несколько минут. Если, при этом, оба объекта знают, что **подбор ответа** требует нескольких часов работы суперЭВМ, то они могут быть спокойны за то, что решение не «подстроено» другой стороной.
  - Для обеспечения таких временных параметров в вычислениях должна использоваться функция  $f(x)$ , значение которой  $y = f(x)$  при известном аргументе  $x$  вычислить можно относительно быстро. Но вот решить уравнение  $f(y) = x$ , т.е. отыскать неизвестное  $x$  при известном значении  $y$  быстро нельзя. Более того, желательно, чтобы не было известно никаких математических методов для решения этого уравнения, кроме полного перебора или подобного ему по сложности.

# Распределенные задачи и алгоритмы

## *Криптографические протоколы*

- Распределенный алгоритм бросания жребия состоит из шагов:
  - 1. Объект  $O_2$  выбирает случайным образом число  $x$  из большого интервала  $[0, q - 1]$ . Вычисляет  $y = f(x)$ .
  - 2. Объект  $O_2$  пересылает число  $y$  объекту  $O_1$ . Объект  $O_1$  не сможет восстановить число  $x$ .
  - 3. Объект  $O_1$  выбирает случайным образом число  $z$  из большого интервала  $[0, q - 1]$ , и случайный бит  $w$ . Эти действия могут выполняться одновременно с п.1.
  - 4. Объект  $O_1$  вычисляет  $s = h(z, y, w)$ . Число  $z$  здесь необходимо для «маскировки» бита  $w$ , а число  $y$  – для «проверки» объектом  $O_2$  правильности действий объекта  $O_1$ .
  - 5. Объект  $O_1$  пересылает число  $s$  объекту  $O_2$ . Бит  $w$  отправляется объекту  $O_2$ , но он «запрятан» в числе  $s$ . Объект  $O_2$  не сможет восстановить этот бит. Объект  $O_2$  не сможет восстановить и число  $z$ .
  - 6. Объект  $O_2$  выбирает случайный бит  $c$  и отправляет его объекту  $O_1$  открытой пересылкой.
  - 7. Объект  $O_1$  пересылает число  $z$  и бит  $w$  объекту  $O_2$ . Открытая пересылка. Объект  $O_1$  уже может определить результат бросания жребия:  $c + w \pmod{2}$ .
  - 8. Объект  $O_2$  вычисляет  $t = h(z, y, w)$ . Здесь  $z$  и  $w$  только что получены от  $O_1$ , а  $y$  было вычислено в п.1.
  - 9. Объект  $O_2$  сравнивает  $t$  и  $s$ , ранее полученное от объекта  $O_1$ .
  - 10. Если  $t = s$ , то объект  $O_2$  вычисляет результат бросания жребия:  $c + w \pmod{2}$ .

# Распределенные задачи и алгоритмы

## *Криптографические протоколы (дополнительные условия)*

- Целые числа, с которыми приходится оперировать, должны иметь в десятичной записи не менее 150-200 цифр или не менее 512 бит в двоичной записи. Такие числа называют «длинными».
- Пусть  $f(x)$  и  $h(z, y, w)$  – две таких трудно обрабатываемых функции.
  - В функции  $h(z, y, w)$  первые два аргумента – длинные числа, а третий – битовый.
  - Функции  $f$  и  $h$  известны объектам  $O_1$  и  $O_2$ , и они владеют алгоритмами быстрого вычисления значений этих функций при заданных значениях аргументов, но не умеют быстро обращать эти функции, т.е. решать уравнения.

# Распределенные задачи и алгоритмы

- Функции  $f$  и  $h$  могут быть различными. В частности, используются функции:

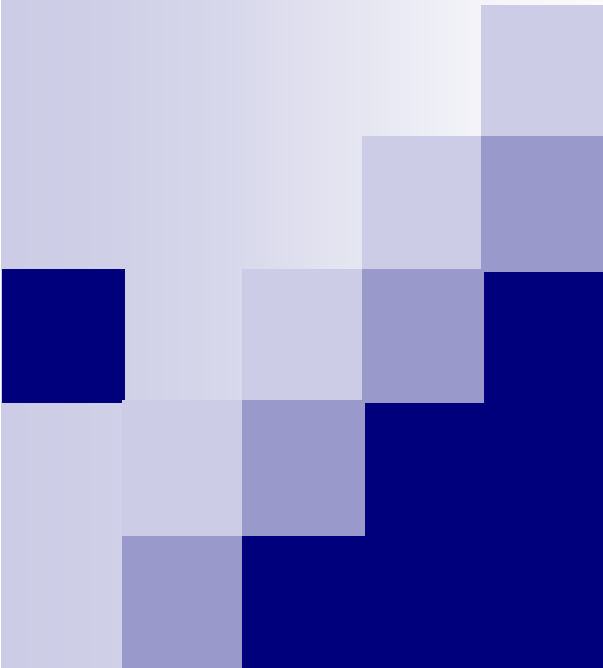
$$f(x) = g^x \pmod{p} \text{ и } h(z, y, w) = y^w g^z \pmod{p}.$$

- Здесь «секретные» значения  $x, w, z$  находятся в показателях степеней, и для того, чтобы найти их, требуется решить задачу дискретного логарифмирования, для которой эффективный алгоритм, существенно лучший полного перебора, неизвестен.
- Константы  $p, q, g$  должны быть известны тому и другому объектам. Число  $p$  – длинное простое число. Число  $q$  – также длинное простое число, являющееся делителем числа  $p - 1$ . Число  $g < p$ , не равное 1, удовлетворяет условию

$$g^q \neq 1 \pmod{p}.$$

# Типовые задачи (продолжение)

- Взаимодействие между процессами (в паре или группе)
- Организация обработки запросов на стороне сервера
- Обнаружение отказов и учёт участников
- Именованное, 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



# Надежность и безопасность распределенных систем

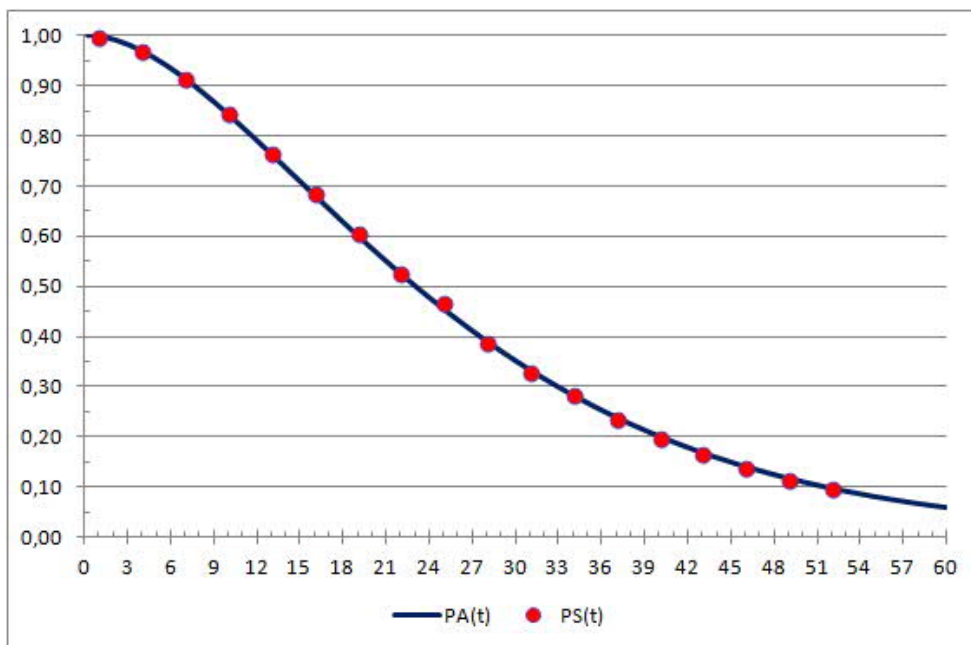


# Надежность и безопасность распределенных систем

- Под *надежностью* понимается в соответствии с ГОСТ 27.002-89 *свойство системы сохранять во времени в установленных пределах значения всех параметров, характеризующих способность выполнять требуемые функции в заданных режимах и условиях применения, технического обслуживания и транспортирования.*
  - Многие системы не являются абсолютно надежными, т.е. свойство надежности системы проявляется на *конечном интервале времени эксплуатации системы*, по истечении которого происходит *отказ* в работе. Длительность интервала безотказной работы зависит от очень большого числа факторов, предсказать которые нереально, поэтому, отказ обычно считают *случайным событием*.

# Надежность и безопасность распределенных систем

- Надежность принято характеризовать **вероятностью отказа** в работе (или **вероятностью безотказной работы**) в течение определенного отрезка времени.
- Другой характеристикой надежности системы является **среднее время наработки на отказ**.



Функция **надежности** тестовой МАС  $PA(t)$  и экспериментальные значения вероятности безотказной работы  $PS(t)$

# Надежность и безопасность распределенных систем

- Под **безопасностью** понимается состояние защищенности системы от потенциально и реально существующих угроз, или отсутствие таких угроз. Система находится в состоянии безопасности, если действие внешних и внутренних факторов не приводит к ухудшению или невозможности ее функционирования.
- Угрозы могут быть различного рода, в том числе угроза физического разрушения.
- В контексте нашей темы интересны угрозы **информационные**. К ним относятся угрозы получения системой
  - недостоверной входной информации,
  - искажения внутрисистемной информации,
  - утечка информации о функционировании системы.



# Надежность и безопасность распределенных систем

- **Информационная безопасность** — состояние защищенности информационной среды общества, обеспечивающее ее формирование, использование и развитие в интересах граждан, организаций, государства.
- В качестве стандартной модели безопасности часто приводят модель CIA:
  - **конфиденциальность** информации – confidentiality (обязательное для выполнения лицом, получившим доступ к определенной информации, требование не передавать такую информацию третьим лицам без согласия ее владельца);
  - **целостность** (integrity) - гарантия существования информации в непротиворечивом виде;
  - **доступность** (availability) - возможность получение информации авторизованным пользователем в нужное для него время.

# Надежность и безопасность распределенных систем

- Выделяют и другие категории безопасности:
  - **аутентичность** — возможность установления автора информации;
  - **апеллируемость** — возможность доказать что автором является именно заявленный человек, и не никто другой.
- Все физические элементы любой системы являются потенциально ненадежными и уязвимыми с точки зрения безопасности.

# Надежность и безопасность распределенных систем

- Ненадежность элементов системы, осуществляющих переработку информации, может заключаться
  - в полном отказе от переработки,
  - в изменении функции (стабильном получении неверных результатов),
  - в сбоях (периодическом возникновении ошибок).
  - в полном прекращении передачи, в одностороннем прекращении передачи (для двунаправленных каналов),
  - в возникновении случайных ошибок при передаче (помех).

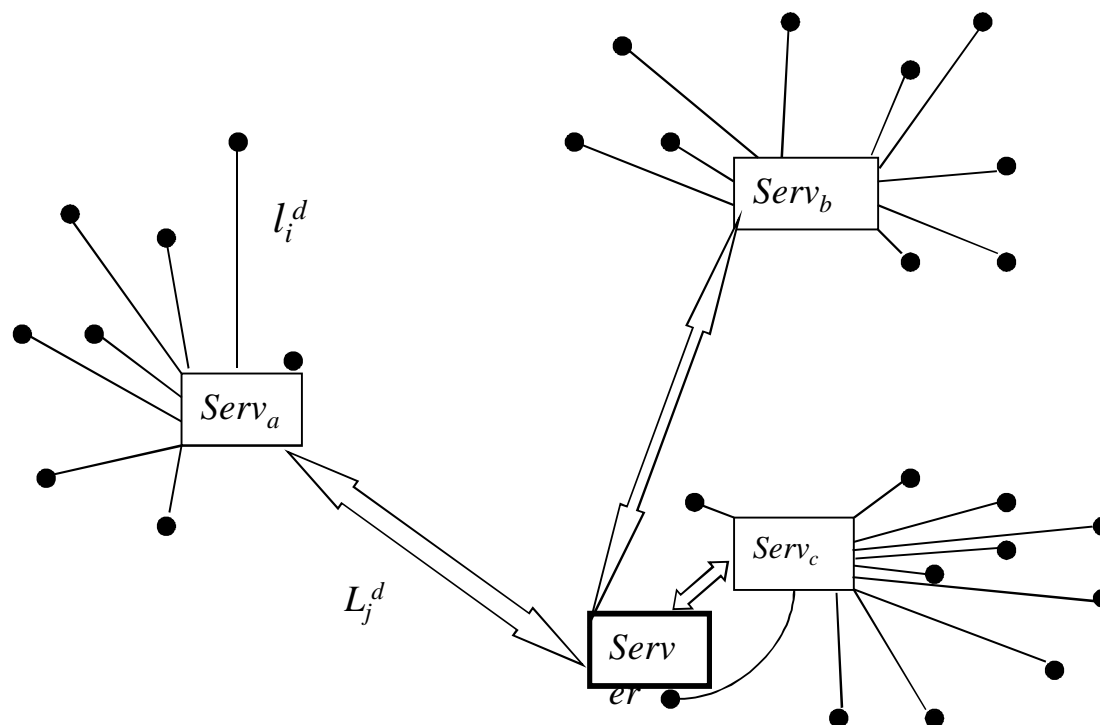
# Надежность и безопасность распределенных систем

- Таким образом, проблемы надежности и безопасности во многом родственны. Они связаны с вмешательством в функционирование системы. Различие заключается в том, что ненадежность определяется физическими, природными факторами и не связана с чьими-то целями.
- Небезопасность определяется, в основном, «человеческим фактором» - наличием злоумышленников и/или беспечных сотрудников. Но одна из проблем безопасности – утечка информации при несанкционированном доступе – не имеет аналога среди проблем надежности.

# Надежность и безопасность распределенных систем

- В распределенной системе количество элементов больше, чем в сосредоточенной:  $S_d$  включает дополнительные серверы  $Serv_a$ ,  $Serv_b$ ,  $Serv_c$  и дополнительные элементы (линии связи)  $L_j$ . Количество линий связи  $l_i$  объектов с серверами в сосредоточенной и распределенной системах одинаково – оно определяется количеством объектов.

• Каждый фактор с точки зрения надежности, если его рассматривать изолированно, играет положительную или отрицательную роль.





# Надежность и безопасность распределенных систем

## ■ *Негативно влияет на надежность:*

- увеличение количества ненадежных элементов в системе при прочих равных условиях играет отрицательную роль. Под прочими равными условиями здесь понимается
  - неизменность архитектуры (соединений и распределения функций) системы,
  - неизменность параметров элементов и проч.
- Если же архитектуру изменить, например, использовать дополнительные элементы для дублирования (резервирования), то надежность, напротив, повышается.
- Увеличение длины линий связи: Обычно с увеличением длины линии увеличивается
  - количество помех,
  - стоимость передачи,
  - возможности злоумышленников по съему информации или по ее искажению.

# Методики построения алгоритмов для обеспечения надежности DS

Все эти факторы надо учитывать при разработке **методики построения алгоритмов**:

- ☐ Она должна быть рассчитана на возможные сбои или рассогласования в работе узлов системы (при программировании в сосредоточенных системах возможность сбоя обычно не учитывается).
- В настоящее время существует два подхода к разработке распределенных алгоритмов:
  - ☐ построение отказоустойчивых алгоритмов и
  - ☐ построение стабилизирующих алгоритмов.
- В **устойчивых** алгоритмах каждый шаг каждого процесса предпринимается с достаточной осторожностью, чтобы гарантировать, что, несмотря на сбои, правильные процессы выполняют только правильные шаги.
- В **стабилизирующих** алгоритмах правильные процессы могут быть подвержены сбоям, но алгоритм в целом гарантирует исправление ошибок.

# Устойчивые алгоритмы

- Разработаны так, чтобы учитывать возможность сбоев в некоторых процессах (относительно небольшом их количестве) и гарантировать при этом правильность выполнения тех процессов, в которых не произошло сбоев.
- Эти алгоритмы используют такие стратегии как голосование, вследствие чего процесс воспримет только такую информацию извне, о получении которой объявит достаточно много других процессов. Однако процесс никогда не должен ждать получения информации от всех процессов, потому что может возникнуть тупик, если при выполнении какого-либо процесса произойдет сбой.
- Устойчивые алгоритмы защищают систему от отказов ограниченного числа узлов. Остающиеся работоспособными узлы поддерживают правильное (хотя возможно менее эффективное) поведение во время восстановления и реконфигурации системы.
- Следовательно, устойчивые алгоритмы должны использоваться, когда невозможно временное прерывание работы.

# Стабилизирующие алгоритмы

- Предлагают защиту против *временных сбоев*, то есть, временного аномального поведения компонент системы. Эти сбои могут происходить в больших частях распределенной системы, когда физические условия временно достигают критических значений, стимулируя ошибочное поведение памяти и процессоров.
- Примером может быть система управления космической станцией, когда станция подвергается сильному космическому излучению, а также системы, в которых на многие компоненты одновременно воздействуют неблагоприятные природные условия. Когда воздействие этих условий исчезает, процессы восстанавливают свою работоспособность и функционируют на основе программ. Однако из-за их временного аномального поведения глобальное результирующее состояние системы может быть непредвиденным. Свойство стабилизации гарантирует сходимость к требуемому поведению.

# 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