

# Масштабирование

Сухорослов Олег Викторович

Распределенные системы

Факультет компьютерных наук НИУ ВШЭ

31.10.2020

# Проблема



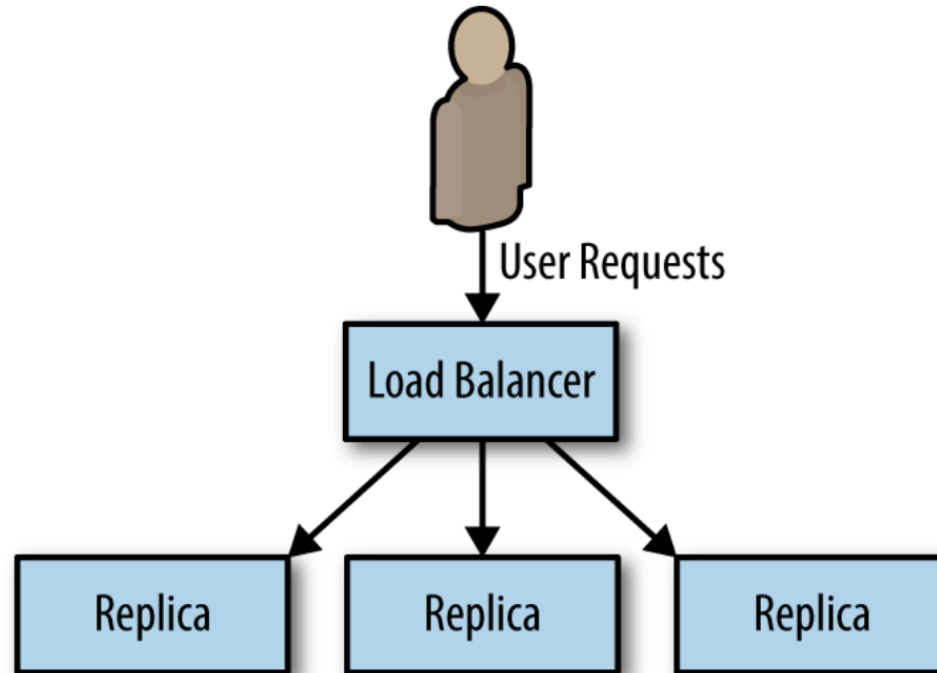
# Аспекты масштабируемости

- Число обрабатываемых запросов
- Объем данных
- Объем вычислений/данных на запрос

# Используемые техники

- Репликация
- Кэширование
- Шардинг
- Параллельная обработка

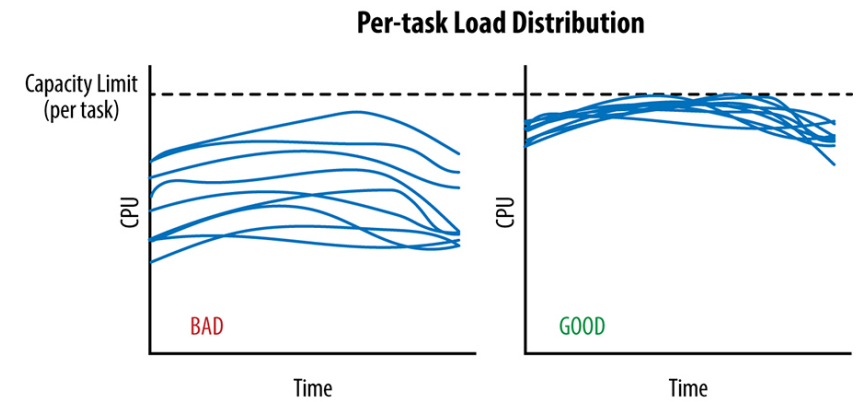
# Репликация (stateless сервис)



# Load Balancer: варианты реализации

- Layer 4 (connection/session)
  - Оперирует данными (пакетами) на транспортном (TCP, UDP) уровне
- Layer 7 (application)
  - Распределяет запросы на прикладном уровне (HTTP) на основе их содержимого
- Балансировка с помощью DNS
- Hardware vs Software
- Load Balancer vs Proxy

# Балансировка нагрузки

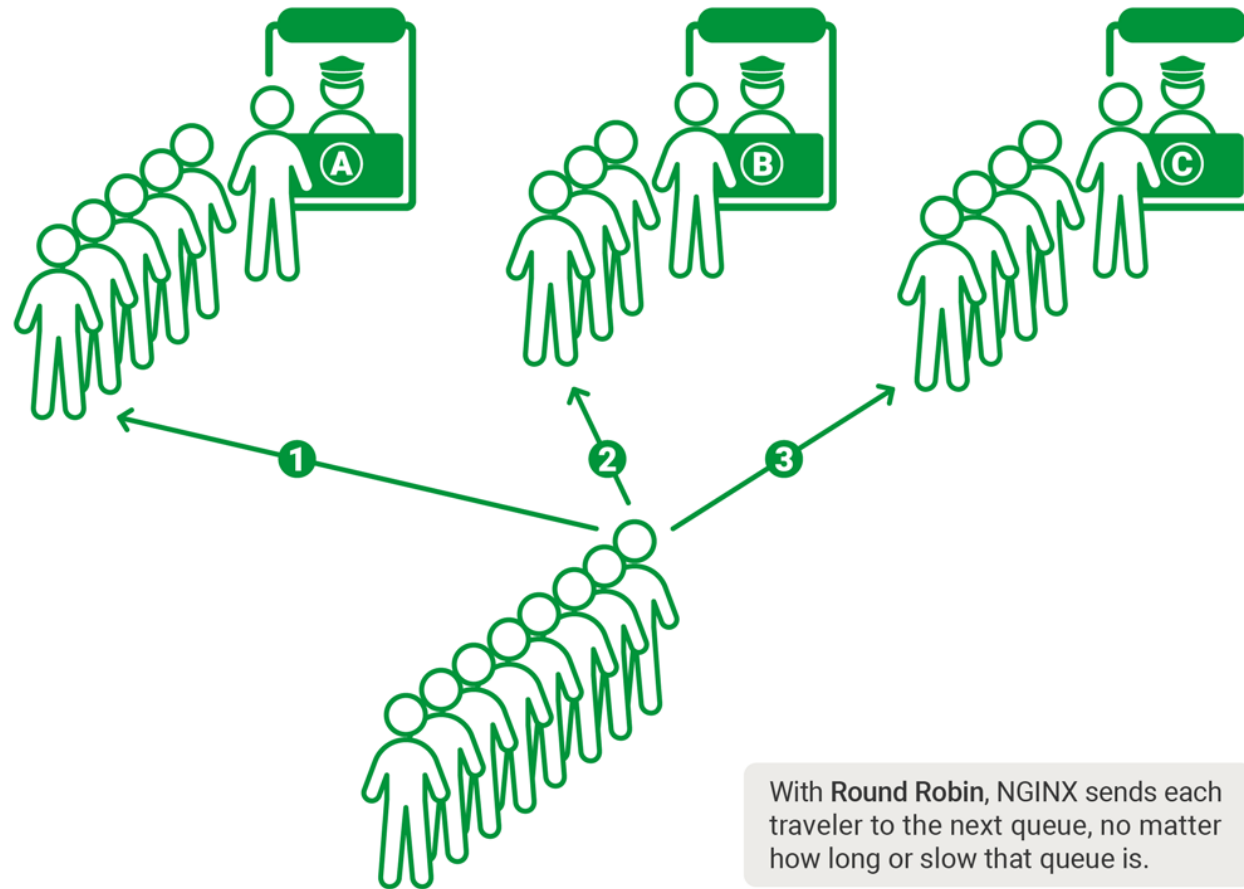


# Алгоритмы

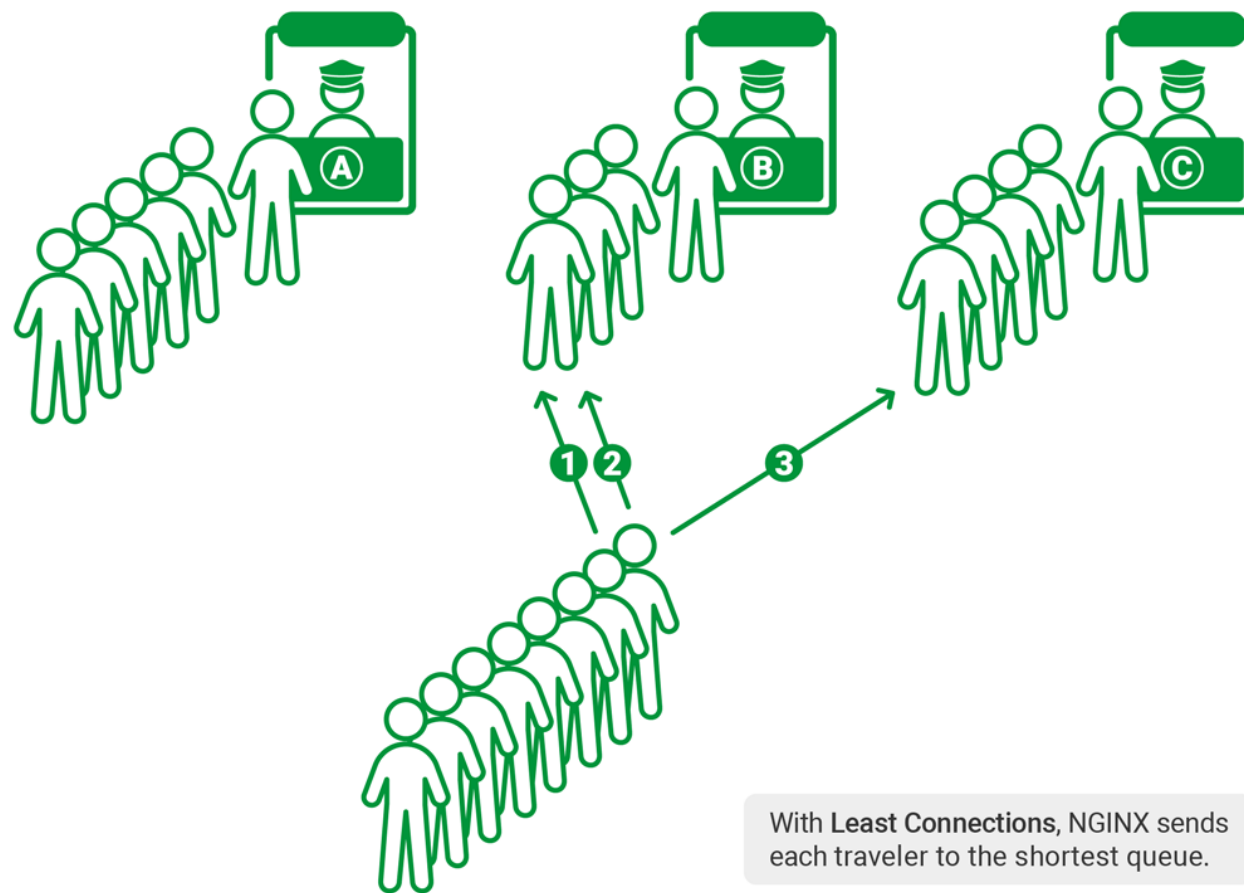
- Random
- Round Robin
- Least Connections
- Least Time
- Power of Two Choices



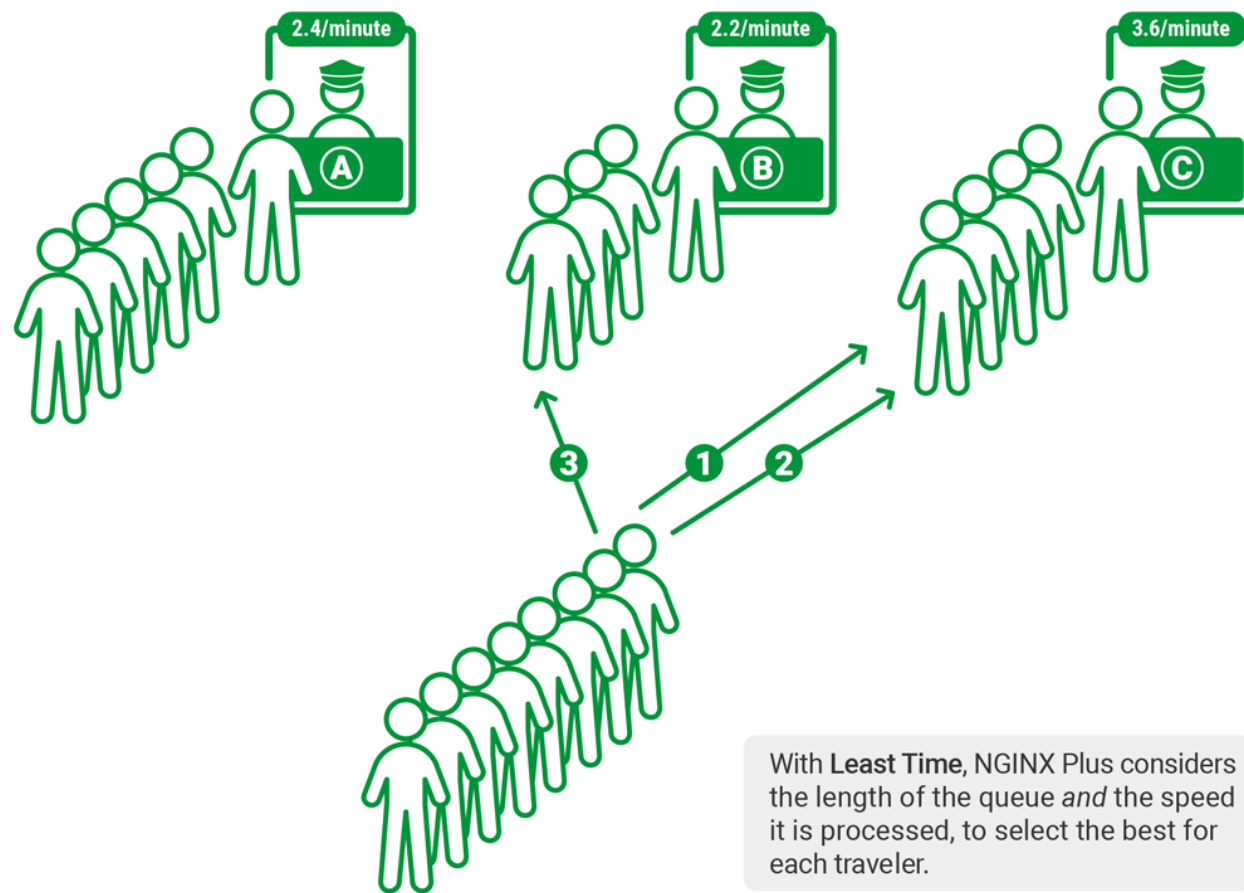
# Round Robin



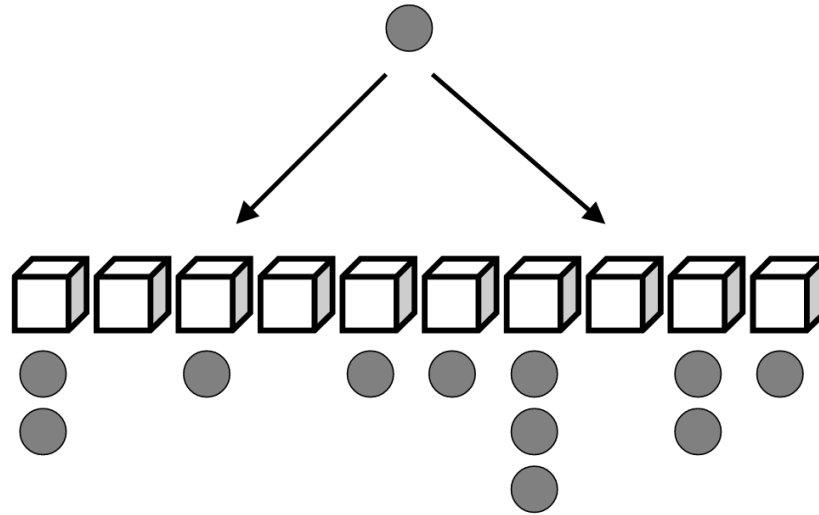
# Least Connections



# Least Time

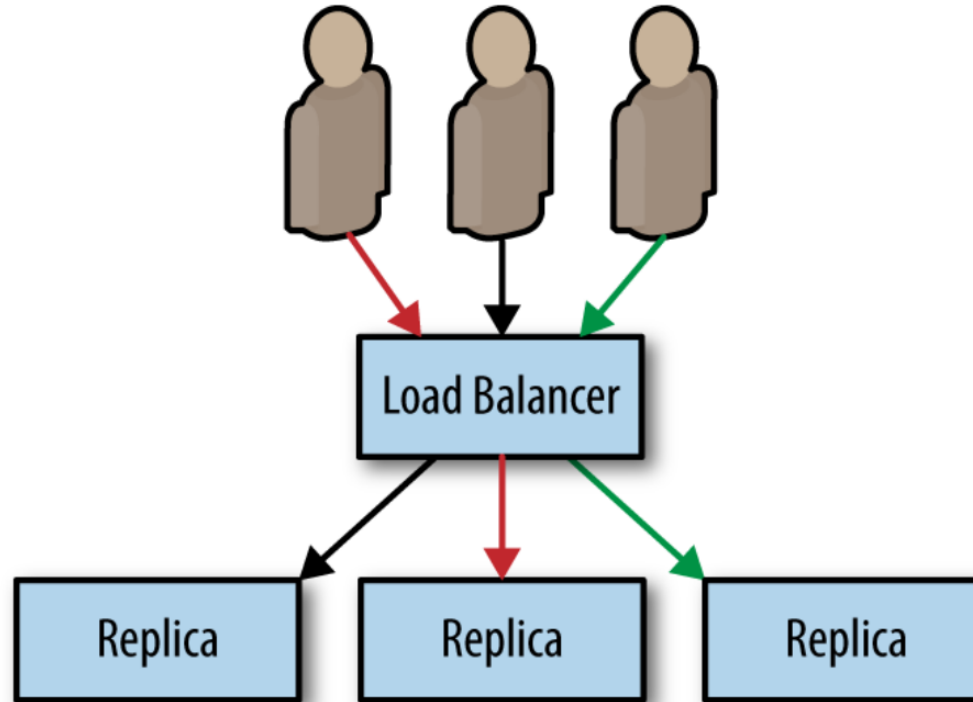


# Power of Two Choices

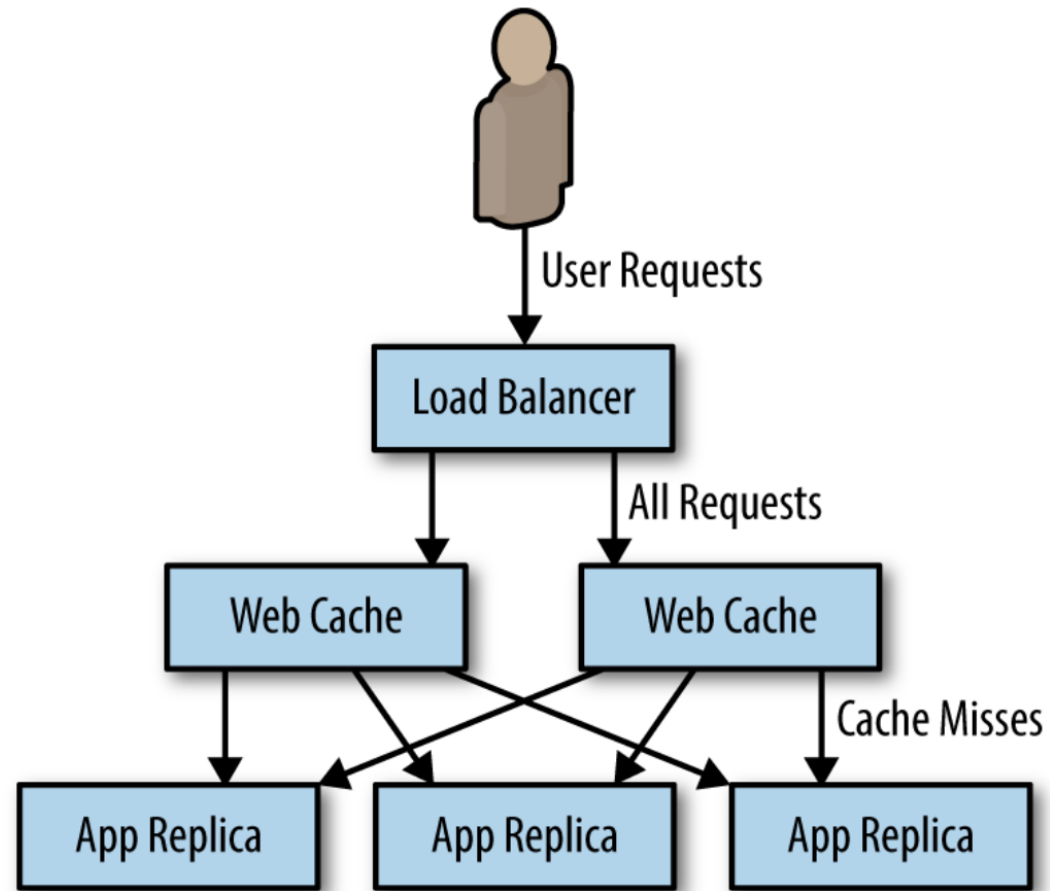


- Максимальная нагрузка:
  - Random:  $O\left(\frac{\log n}{\log \log n}\right)$
  - Two random choices:  $O(\log \log n)$
- Mitzenmacher M. The Power of Two Choices in Randomized Load Balancing (1996)

# Отслеживание клиентских сессий



# Кэширование



# Варианты реализации кэша

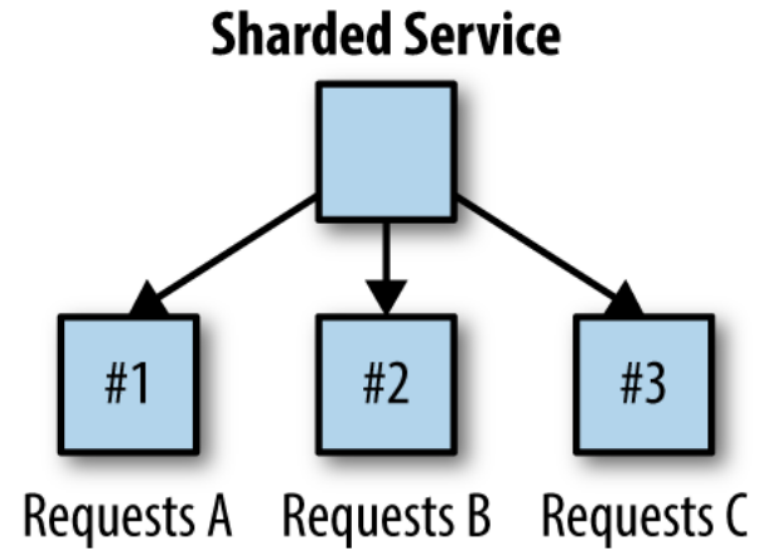
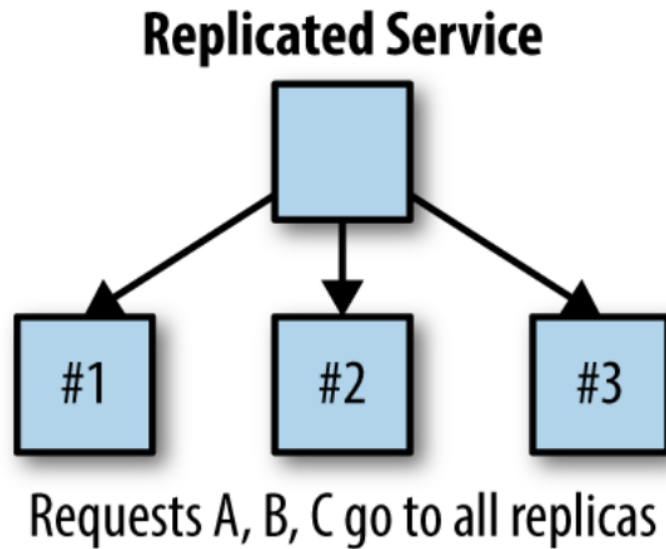
- Кэш на стороне клиента
- Промежуточные кэширующие прокси-сервера
- Caching HTTP reverse proxy
  - Varnish, Nginx
  - Дополнительные функции: rate limiting, защита от DoS-атак, SSL termination
- Отдельные сервисы
  - memcached, Redis

# Политики вытеснения

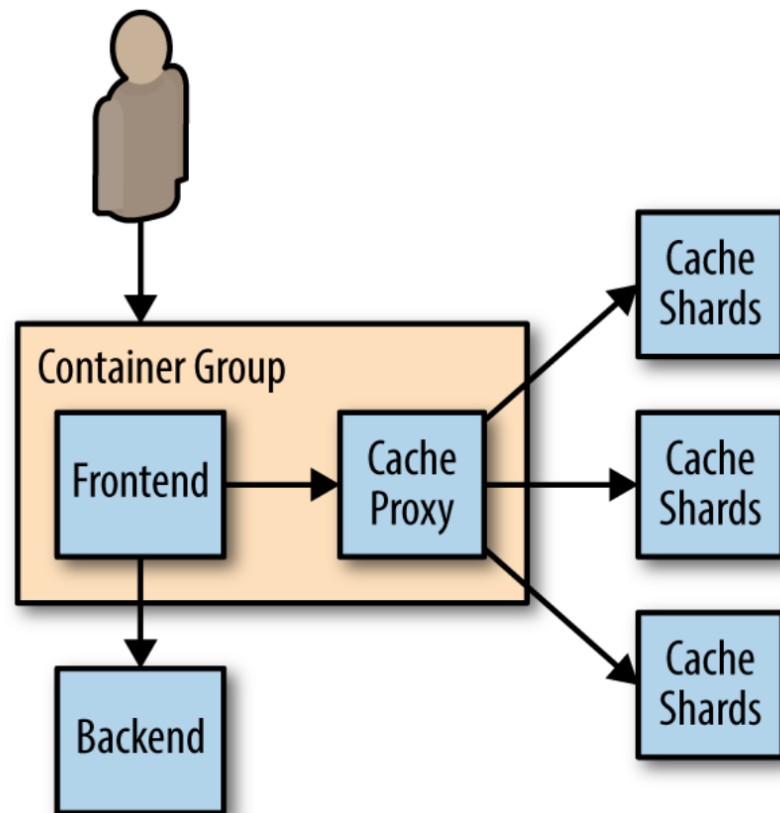
- Least Recently Used
- Least Frequently Used
- Least Frequently Recently Used



# Шардинг (stateful сервис)



# Шардинг кэша

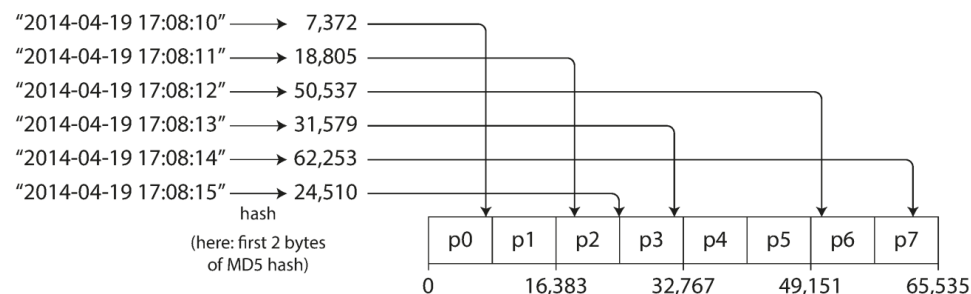
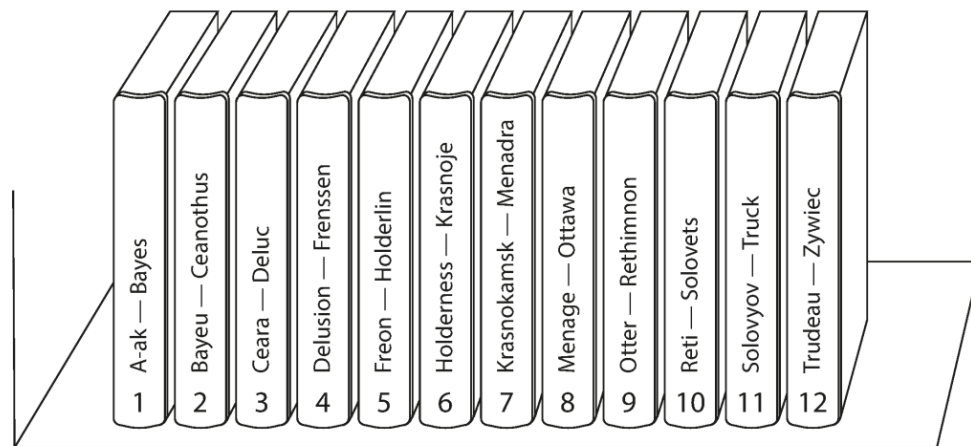


# Принцип разбиения на шарды

- $Shard = ShardingFunction(Req)$
- Требования
  - Детерминированность
  - Равномерность
  - Устойчивость к изменениям числа узлов (rebalancing)

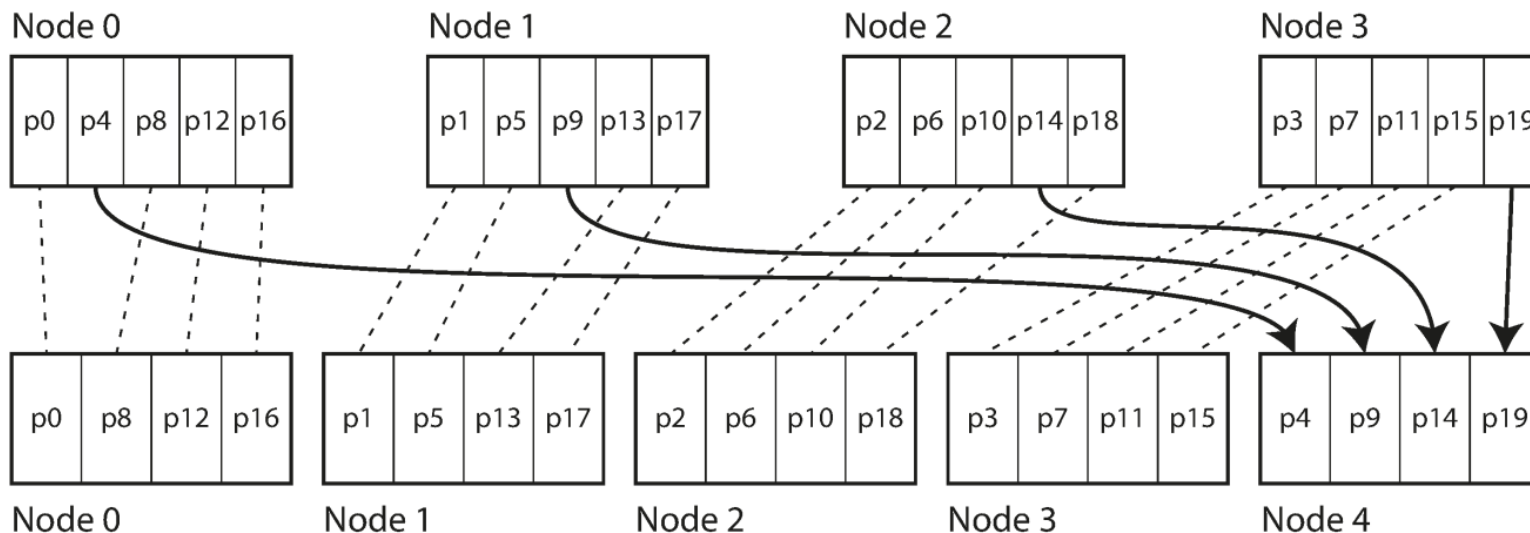
# Подходы

- Вертикальный шардинг
- Горизонтальный шардинг
  - (Выбор ключа)
  - Интервалы ключей
  - Хэширование ключей
  - $\text{hash}(k) \bmod N$
  - Согласованное хеширование
- Число шардов
  - Фиксированное
  - Пропорционально числу узлов
  - Пропорционально размеру данных

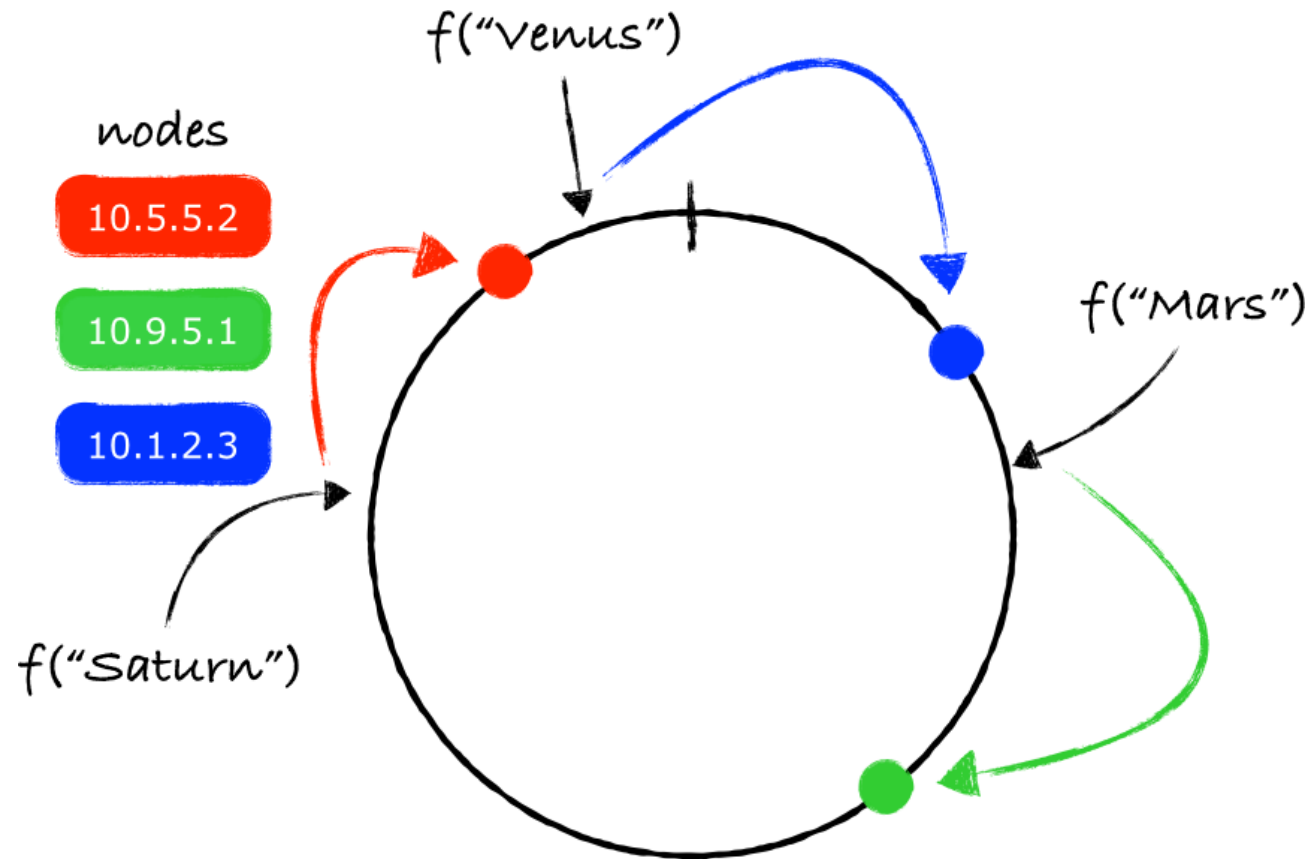


# Фиксированное число шардов

Before rebalancing (4 nodes in cluster)

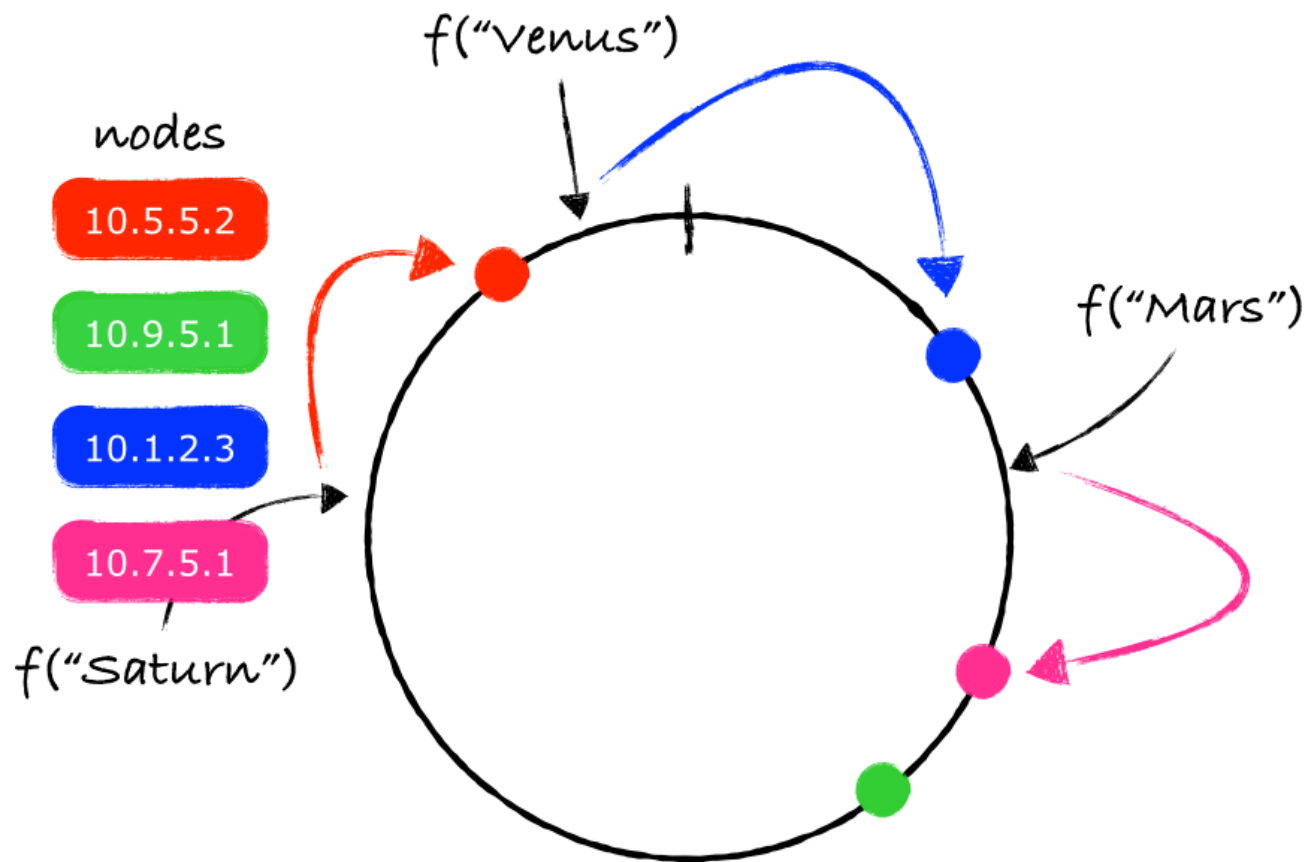


# Согласованное хеширование

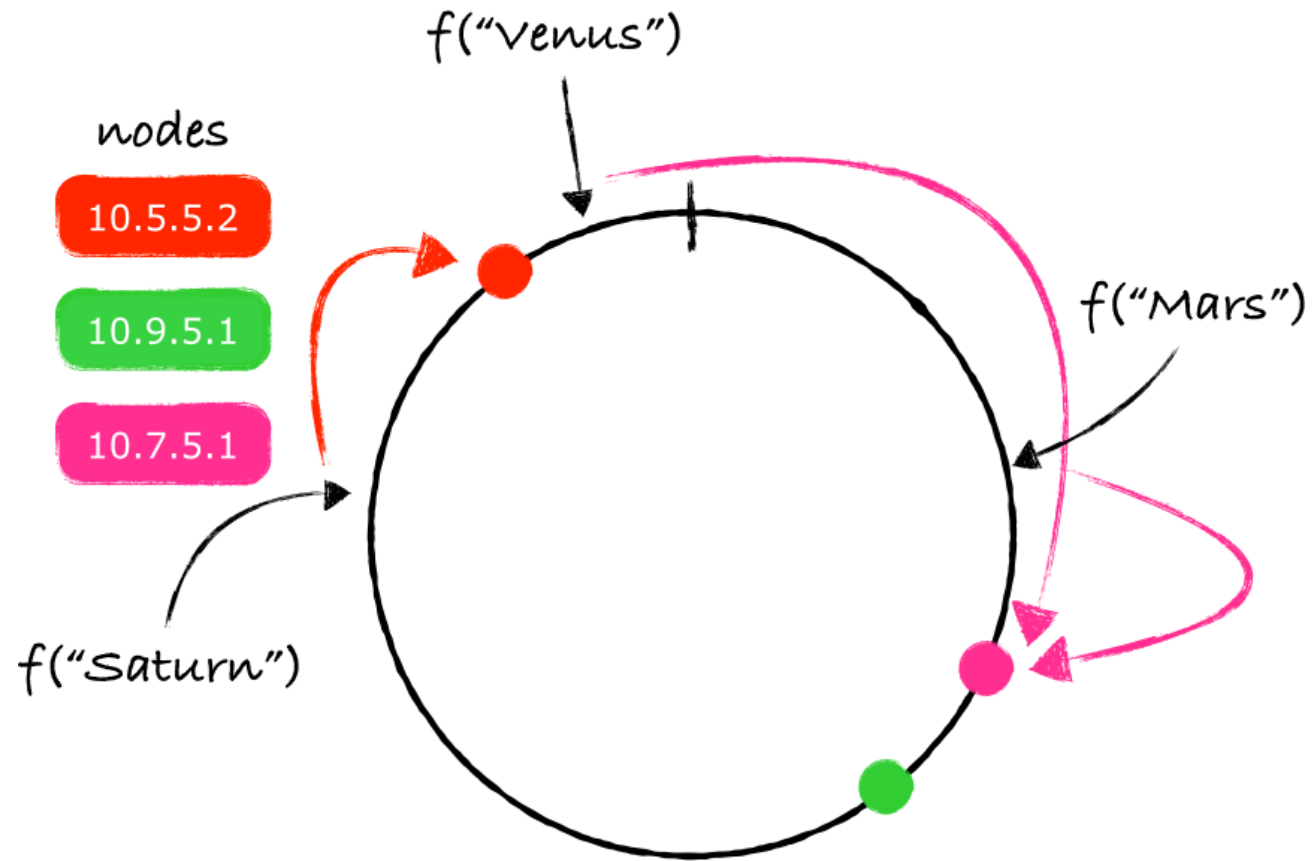


Karger D. et al. Consistent hashing and random trees: Distributed caching protocols... (1997)

# Добавление узла

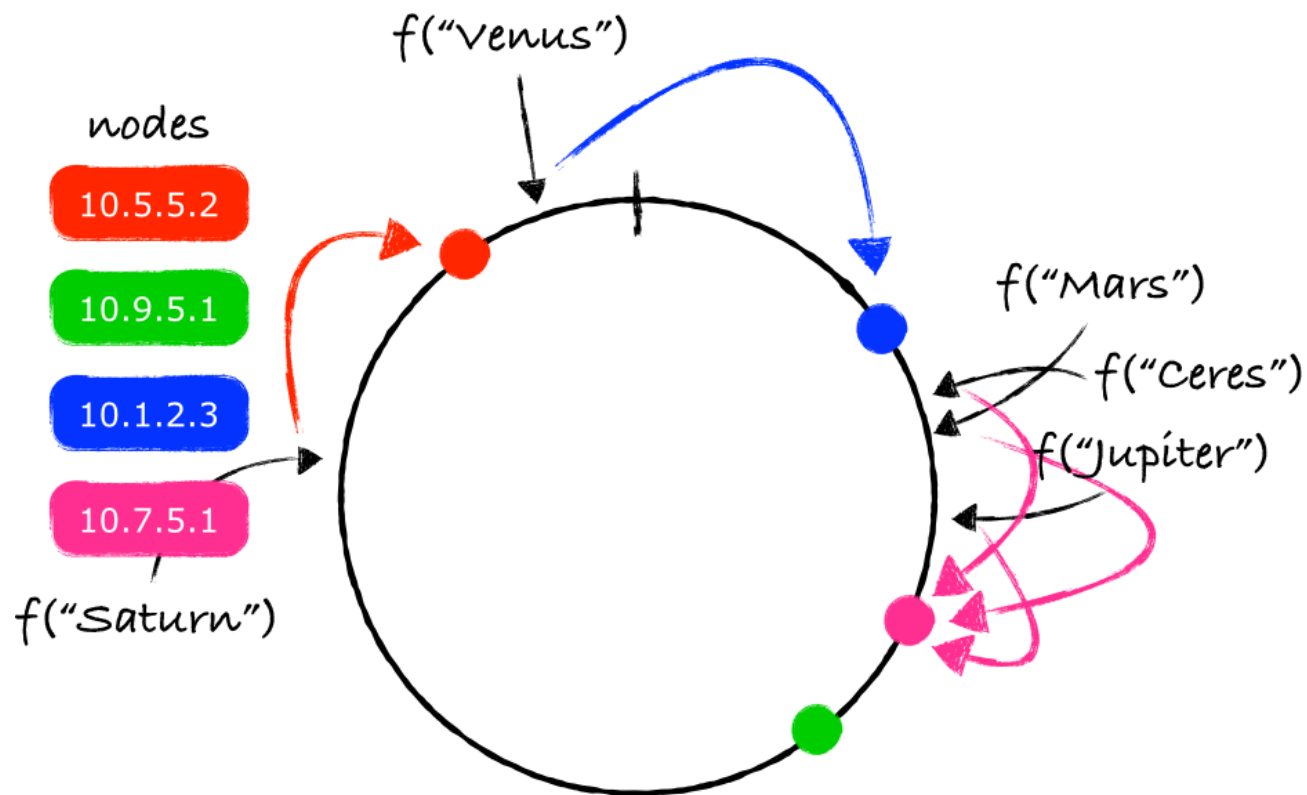


# Удаление узла

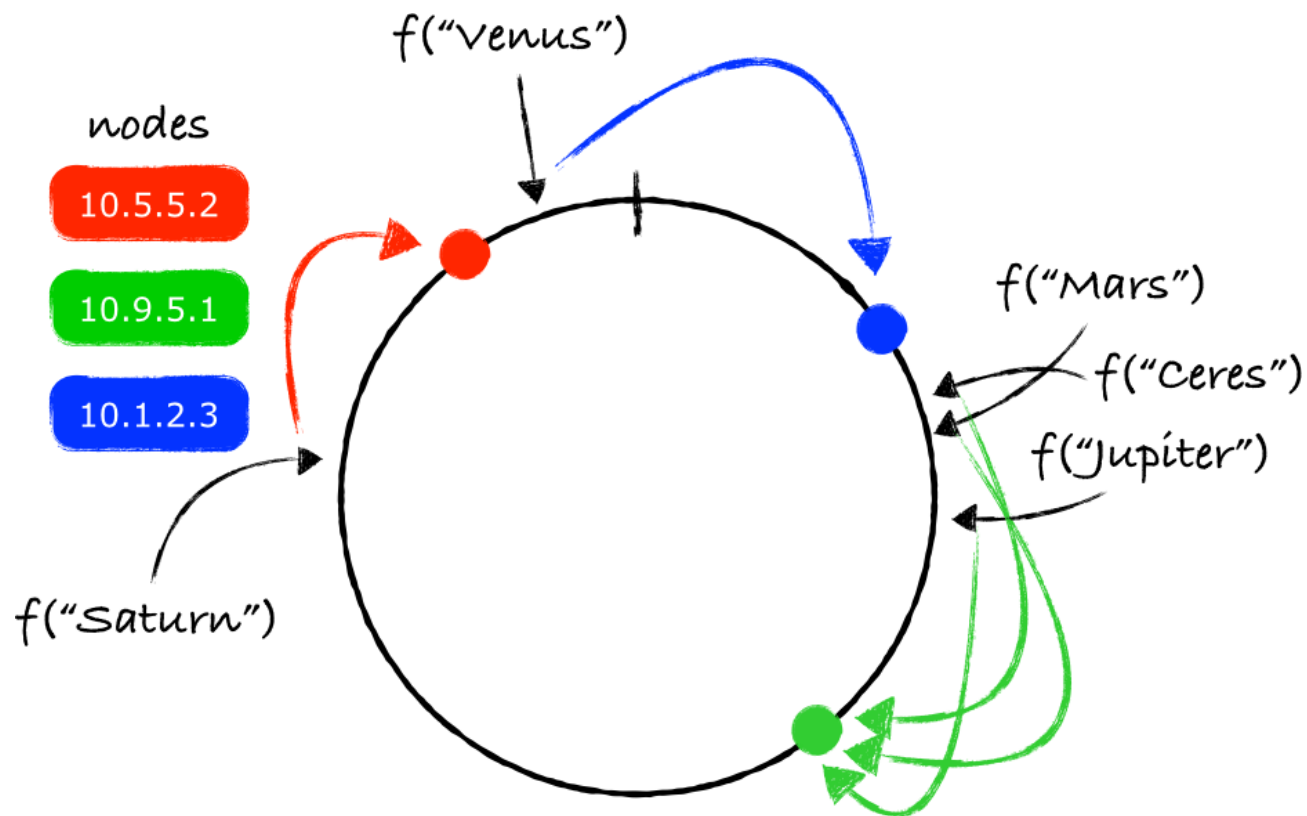




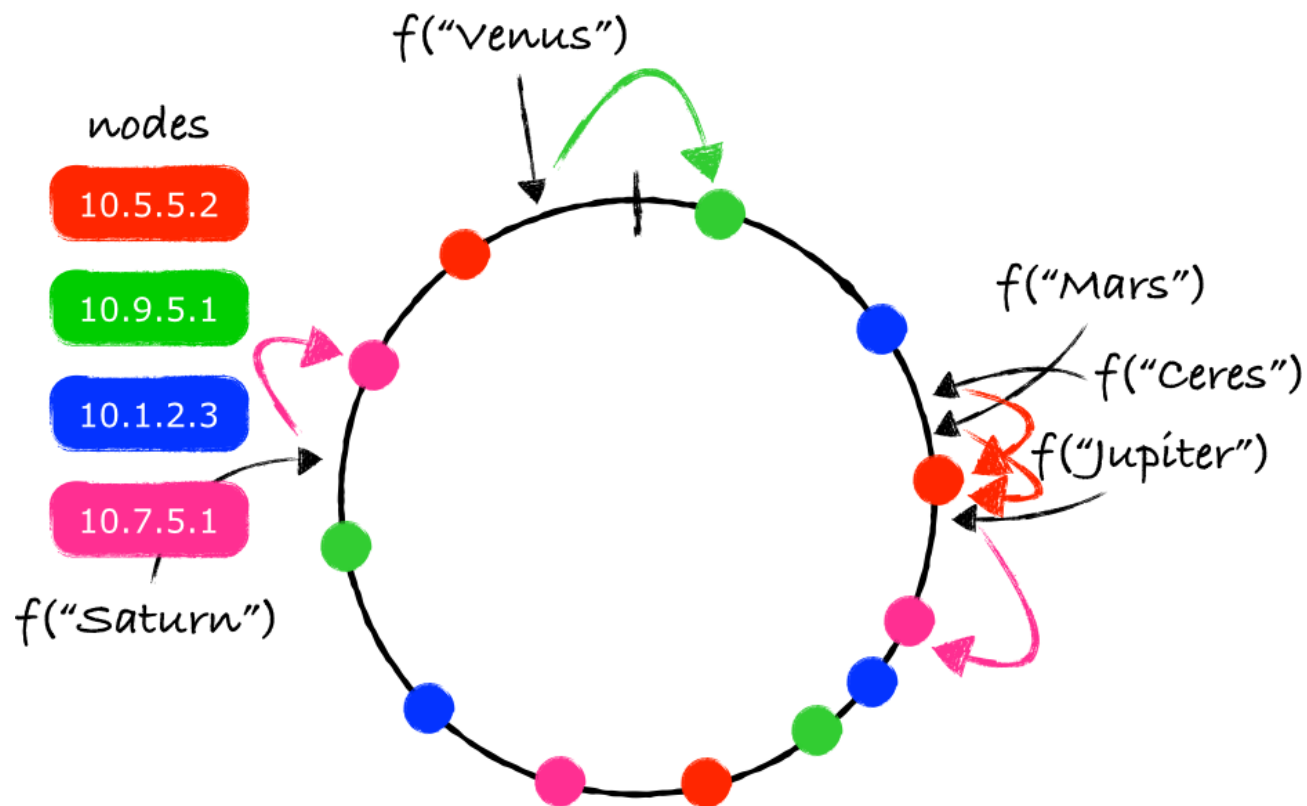
# Проблема 1



# Проблема 2

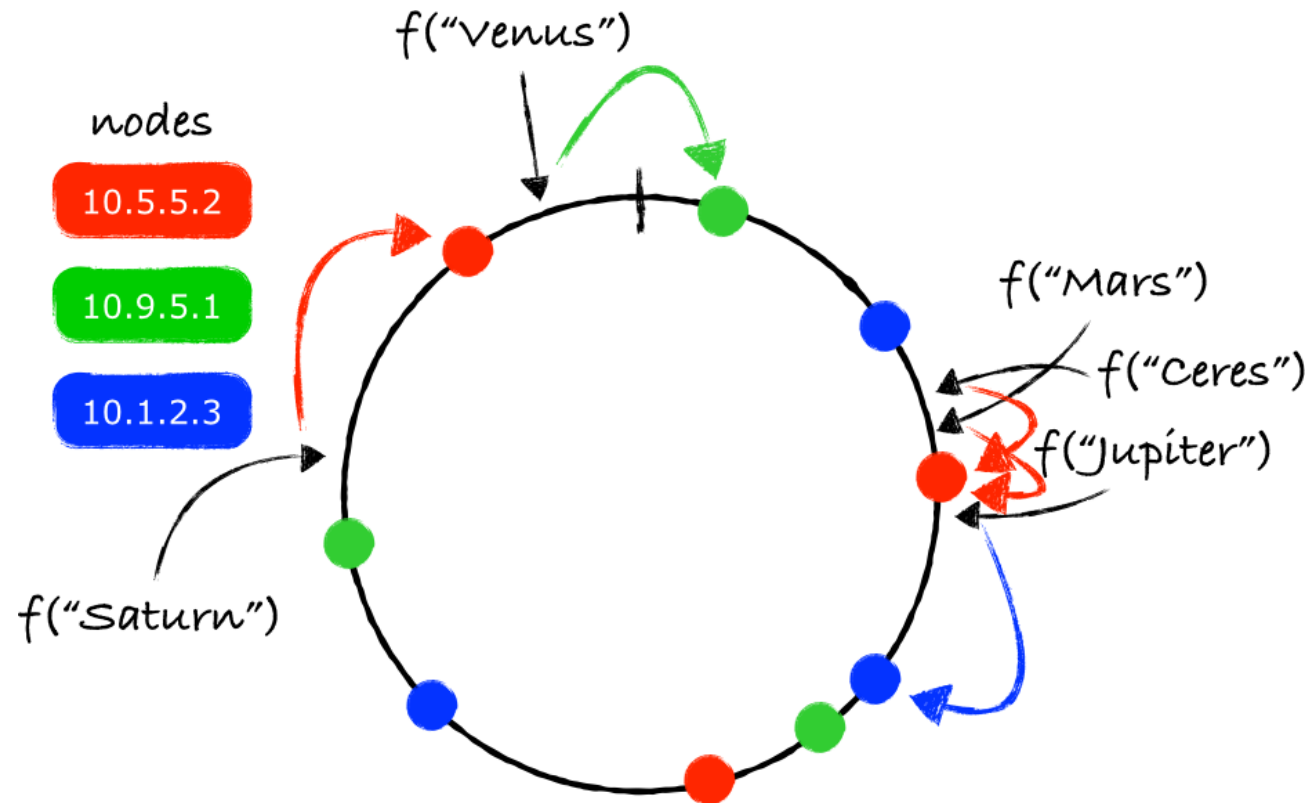


# Виртуальные узлы



Детерминированный алгоритм: New token allocation algorithm in Cassandra 3.0

# Удаление узла



# Rendezvous Hashing

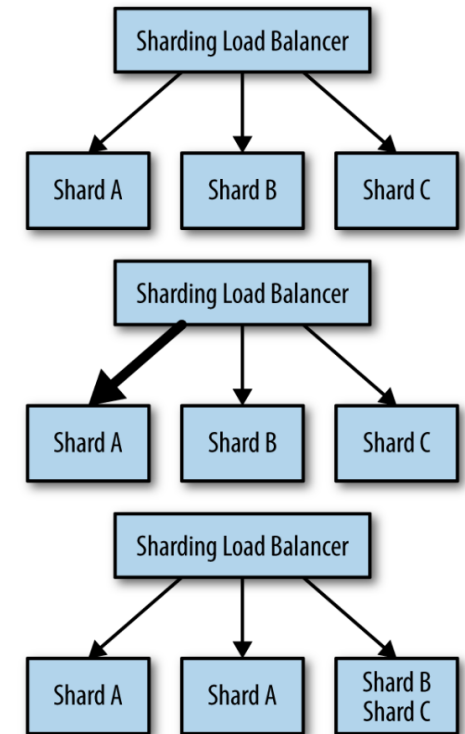
- Другое название Highest Random Weight (HRW)
  - Вес сервера для ключа вычисляется как  $hash(key, server)$
  - Выбирается сервер с максимальным весом
- Thaler D. G., Ravishankar C. V. Using name-based mappings to increase hit rates (1998)

# Другие подходы

- Lamping J., Veach E. A fast, minimal memory, consistent hash algorithm (2014)
- Appleton B., O'Reilly M. Multi-probe consistent hashing (2015)
- Eisenbud D. et al. Maglev: A fast and reliable software network load balancer (2016)
- Mirrokni V. et al. Consistent Hashing with Bounded Loads (2016)

# Балансировка нагрузки при шардинге

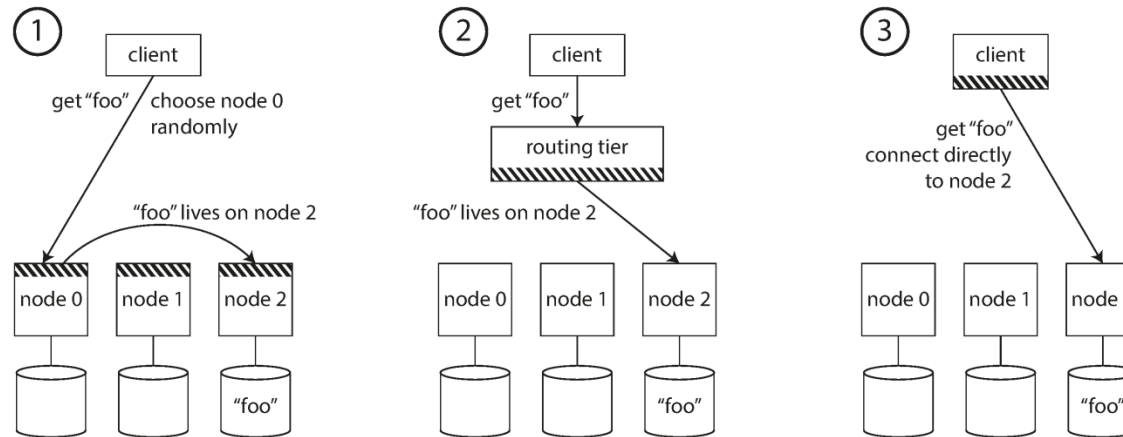
- Выбор "хорошего" принципа разбиения на шарды
  - Не гарантирует отсутствие "горячих" шардов
- Микро-шарды
  - Число шардов >> машин
- Переконфигурация шардов
  - Разбиение и слияние шардов
- Выборочная репликация шардов
  - Изменение числа реплик в зависимости от нагрузки



# Маршрутизация запросов

Как клиент определит, на какой узел надо отправить запрос?

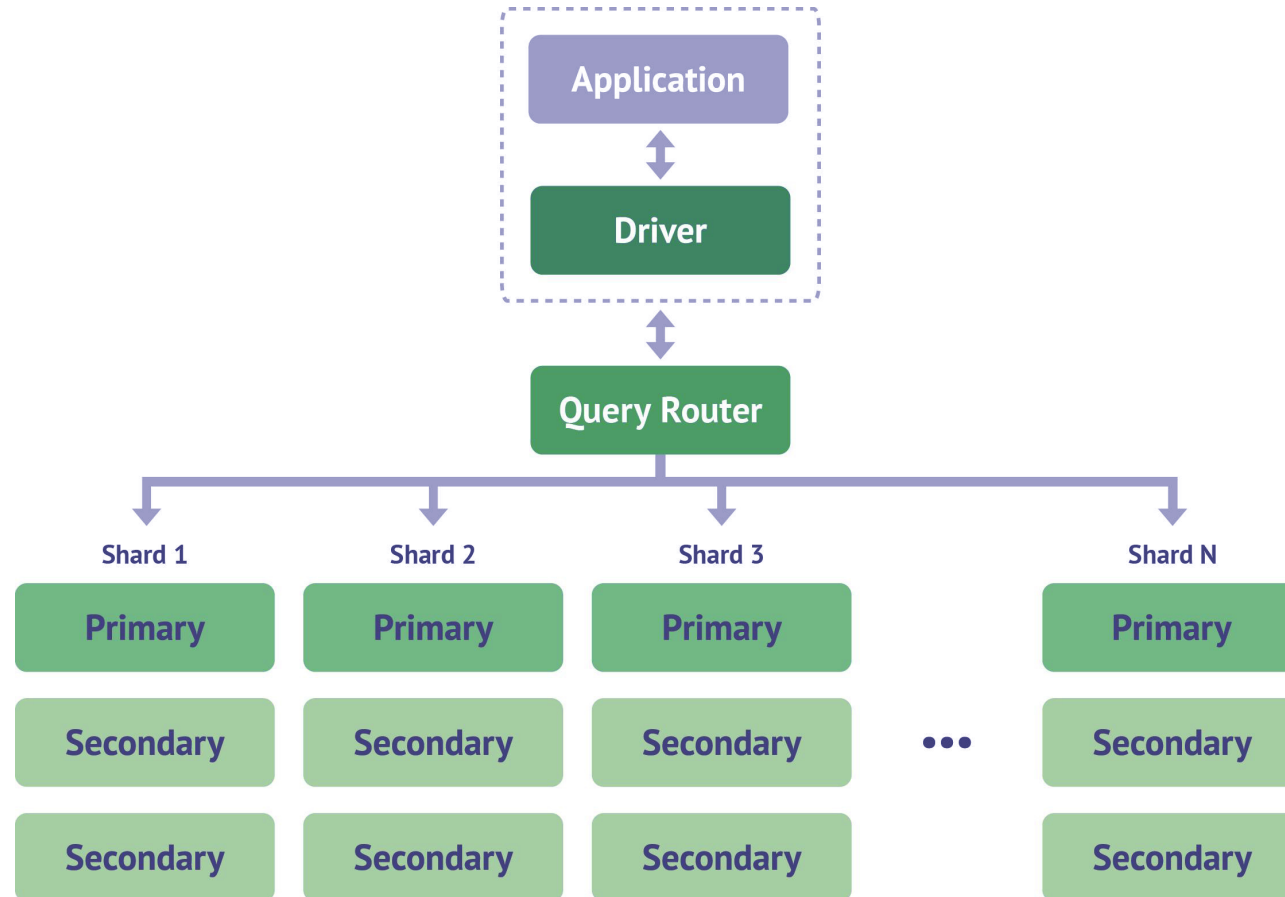
- Обратиться к любому узлу, а тот перенаправит запрос при необходимости
- Использовать отдельный промежуточный слой, знающий о шардах
- Хранить информацию (шард, узел) на клиенте, возможно частично (DHT)



//// = the knowledge of which partition is assigned to which node



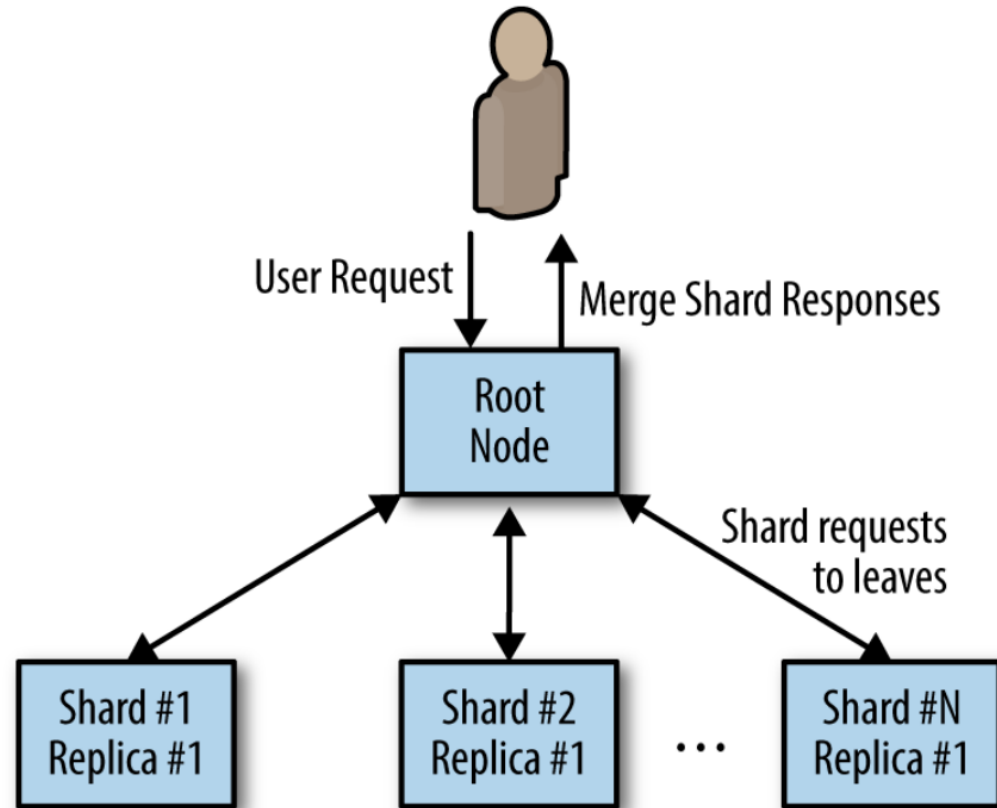
# Шардинг + репликация данных



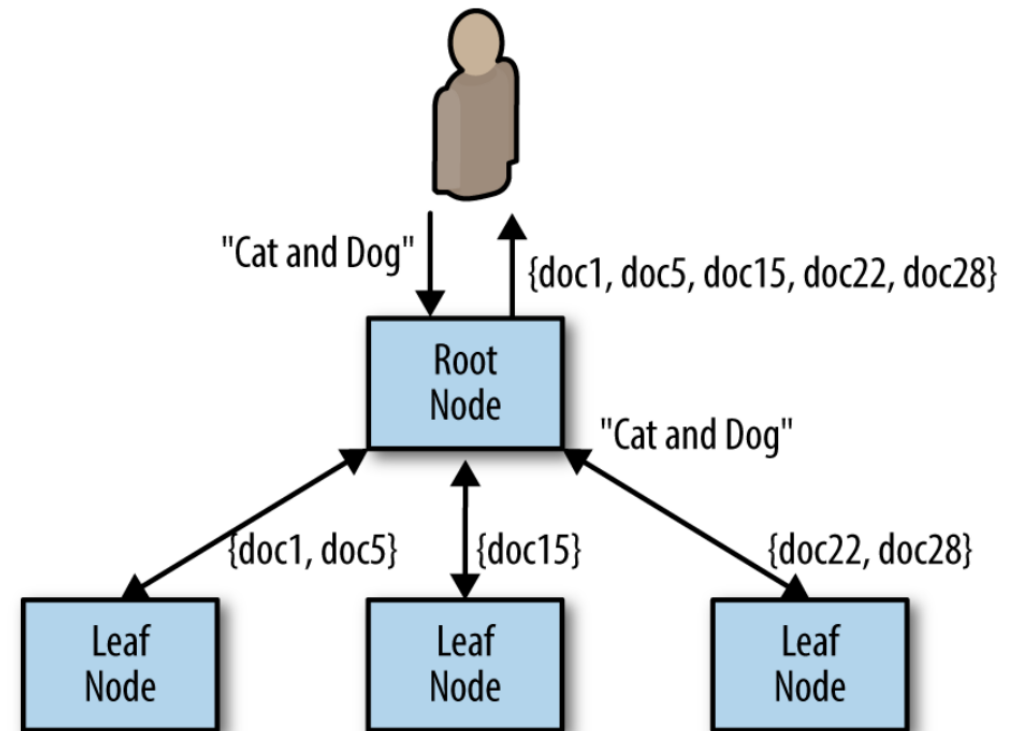
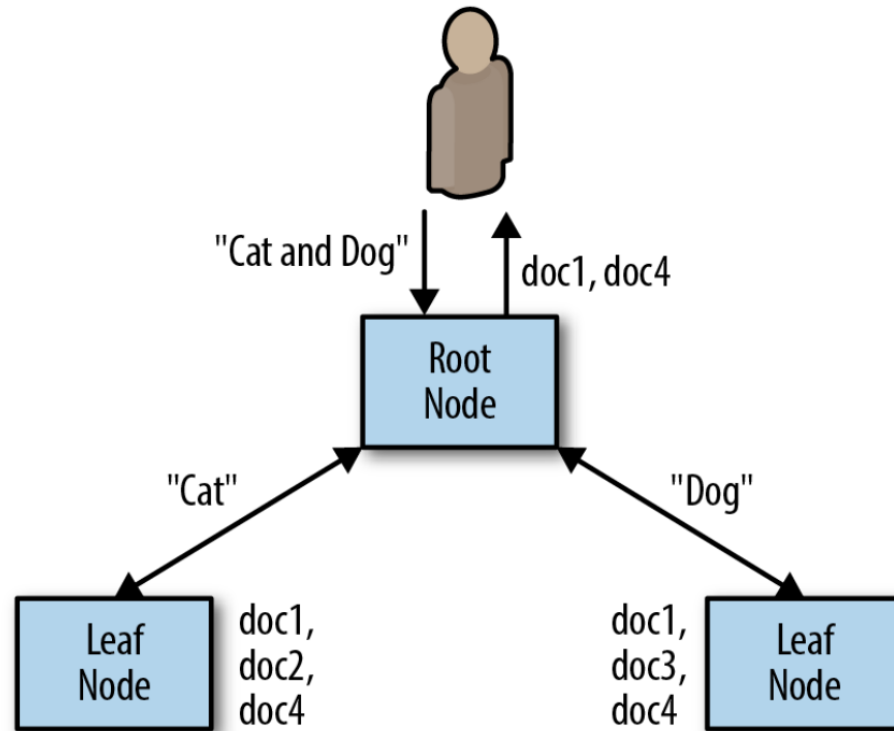
# Масштабирование тяжелых запросов

- Запрос требует проведения вычислений или чтения данных
- Объемы требуемых вычислений и данных растут
- Как обеспечить масштабируемость?

# Параллельная обработка (Scatter/Gather)

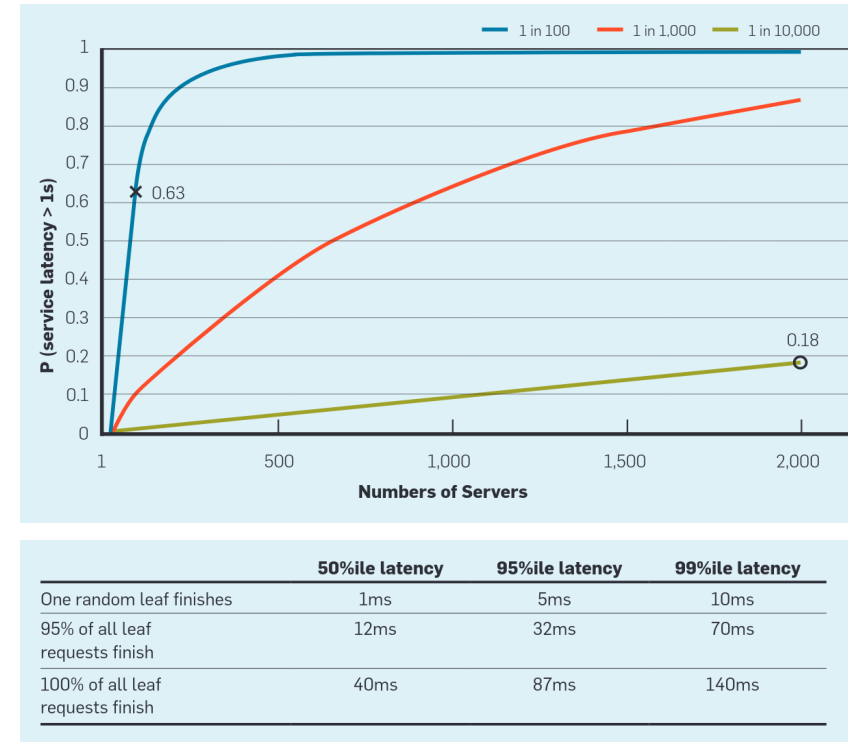


# Распределенный поиск



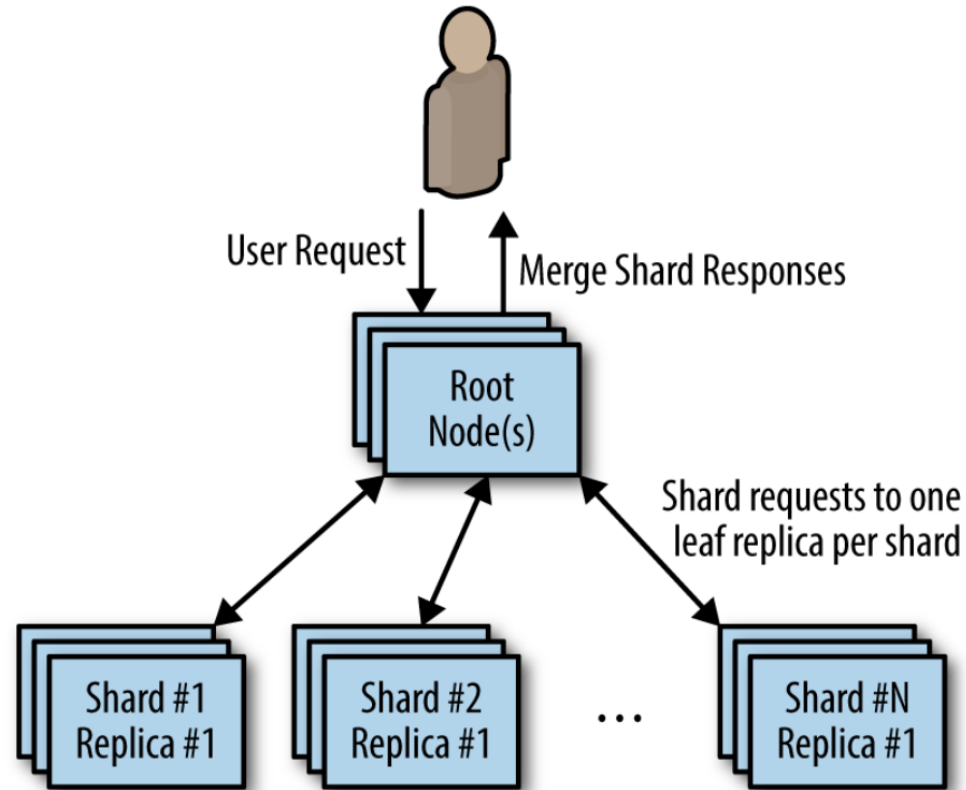
# Проблемы

- Выбор числа серверов
  - Фиксированные накладные расходы на обработку подзапросов
  - Дополнительные накладные расходы на распаралелливание
- Эффект tail latency amplification
  - Время обработки запроса определяется самым медленным сервером (straggler)
  - С ростом числа серверов риск задержки увеличивается



<https://research.google/pubs/pub40801/>

# Распределенный поиск (+репликация)



# Литература

- Burns B. Designing Distributed Systems: Patterns and Paradigms for Scalable, Reliable Services (главы 5-7)
- Site Reliability Engineering (глава 20)
- Kleppmann M. Designing Data-Intensive Applications (глава 6)
- Gryski D. Consistent Hashing: Algorithmic Tradeoffs

# Дополнительно

- Site Reliability Engineering (глава 19)
- Tarreau W. Test Driving "Power of Two Random Choices" Load Balancing
- McMullen T. Load Balancing is Impossible
- Gury S. Predictive Load Balancing: Unfair but Faster & more Robust
- Gessner K. How Etsy caches: hashing, Ketama, and cache smearing
- Аксёнов А. Теория шардирования
- Упомянутые статьи