Fredhopper®

ElasticSearch срещу Solr

Божидар Пенчев, Павел Пенчев

Месечна среща на BGJUG

Целта на тази презентация

Да представим две системи за пълнотекстово търсене – ElasticSearch и Solr, разпределената им архитектура и причините, поради които бихте избрали едната пред другата.

За нас

- Божидар Пенчев
- Павел Пенчев
- Fredhopper

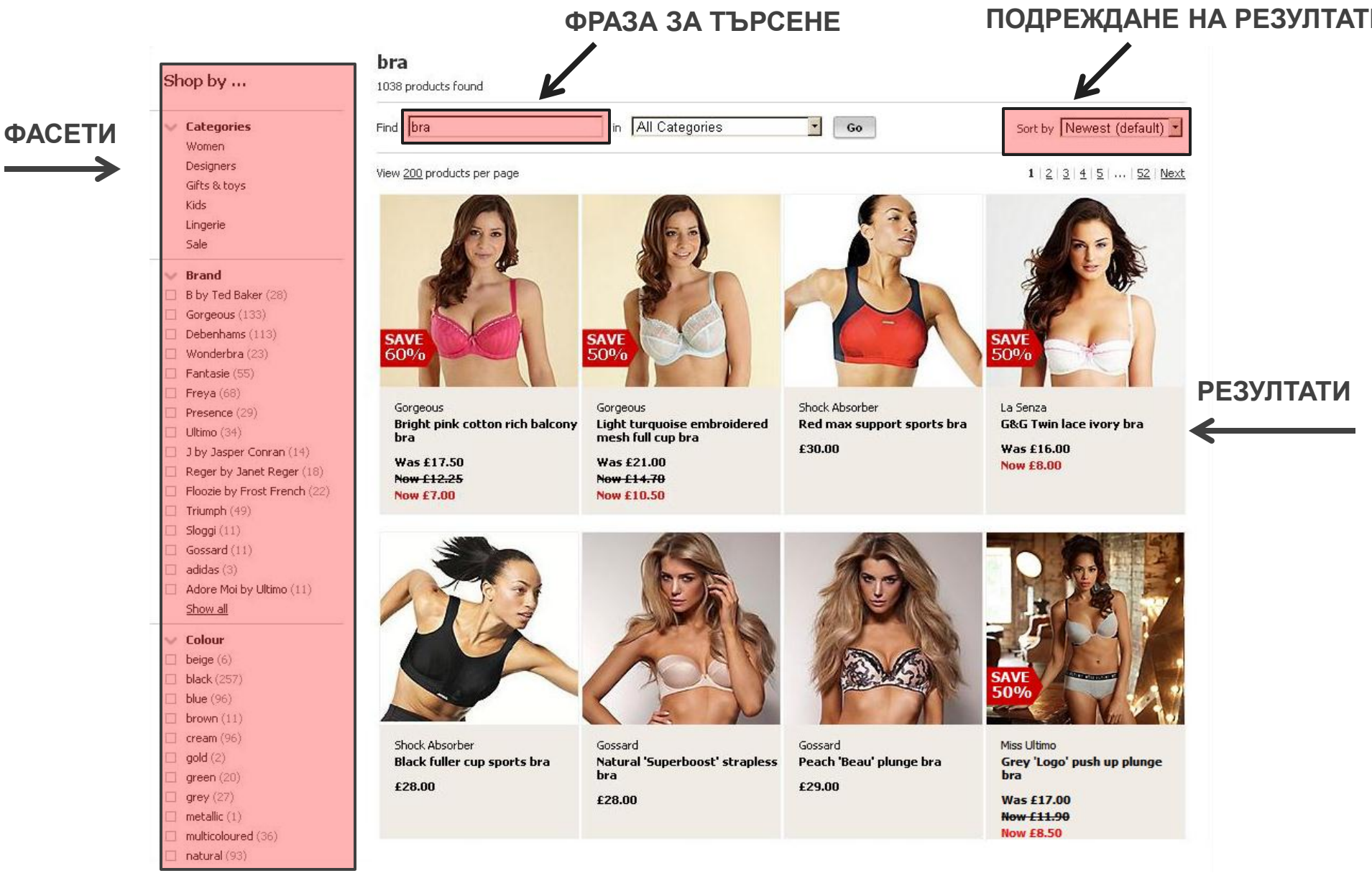
Пълнотекстово търсене

- Позволява ни намирането на документи съдържащи определена дума или фраза и подреждането им според тяхната значимост

- Ползваме го всеки ден - Google, Wikipedia, на мобилен телефон

- Пълнотекстовото търсене е повече от обикновена LIKE "%%" SQL команда. Необходими са:
  - лингвистичен анализ за съответния език
  - правила за определяне на значимост
  - корекция на правописа
  - фасети(изгледи)

Типично приложение на пълнотекстово търсене



План

- Изисквания
- Solr и ElasticSearch
- Lucene – в основата на всичко
- Архитектура
  - ElasticSearch
  - Solr
- Сравнение
- Заключение

Изисквания към системата за търсене

- Гъвкава документна структура
- HTTP интерфейс за индексиране и заявки, подръжка на фасети
- Висока производителност
- Възможност за мащабируемост над 100 милиона документа
- Устойчивост на грешки
- Четене веднага след запис: документът трябва да е наличен за четене веднага след индексиране
- Пълнотекстово търсене в почти реално време

Изборът

- Популярни системи за пълнотекстово търсене с отворен код
- Лицензирани под Apache лиценз
- Базирани на Apache Lucene
- Разполагат с добри Java клиентски API

Lucene

Lucene

- Lucene е библиотека за пълнотекстово търсене написана на Java
- Използва се за индексиране на текстови документи и търсене в създадения индекс
- Голяма набор от функционалност, стабилна, използвана в стотици проекти
- Проект с отворен код под шапката на Apache Software Foundation
- Активна общност, принос дават някои от най-големите компании в света
- Съществува от 2000 година, имплементации съвместими на индекс ниво съществуват на .NET, C++, Python и други

Lucene - концепция

- Lucene е библиотека, използва се програмно от вашия код
- Основната структура за данни в Lucene е обърнат текстов индекс
- Използва абстракция наречена Directory за съхранение на индекса
- Набор от инструменти за лингвистичен анализ на естествен език (stemming, filtering, decomposition) [1]

Lucene - IndexWriter

- Използва се за добавяне и изтриване на документи в индекса
- Съхранява промените в паметта до извикване на IndexWriter#commit
- Не може да има повече от един IndexWriter пишещ по индекса по едно и също време

Lucene - индексни сегменти

- С цел ефективна работа с голям обем данни индекса на Lucene е организиран като поредица от сегменти
- Всеки сегмент може да се разглежда като отделен индекс, непроменяем (immutable) до момента на изтриване
- Удобни за кеширане тъй като не се променят
- IndexWriter#commit добавя нови сегменти към индекса
- Сегментите се сливат в нови сегменти когато ползваемостта им спадне

Lucene – IndexReader

- Ползва се за търсене в индекса
- До версия 2.9 : IndexReader#open които виждат всички промени до последния IndexWriter#commit
- С цел да намали времето между индексирането на документ и наличността му за търсене Lucene 2.9 въвежда подръжка на търсене в почти реално време
- Версия 2.9 : IndexWriter#getReader ни дава IndexReader който "вижда" всички промени в IndexWriter до този момент дори без да извикваме IndexWriter#commit
- Сравнително тежък за създаване, не е добра идея да се създава при всяка заявка за търсене

Lucene - заключение

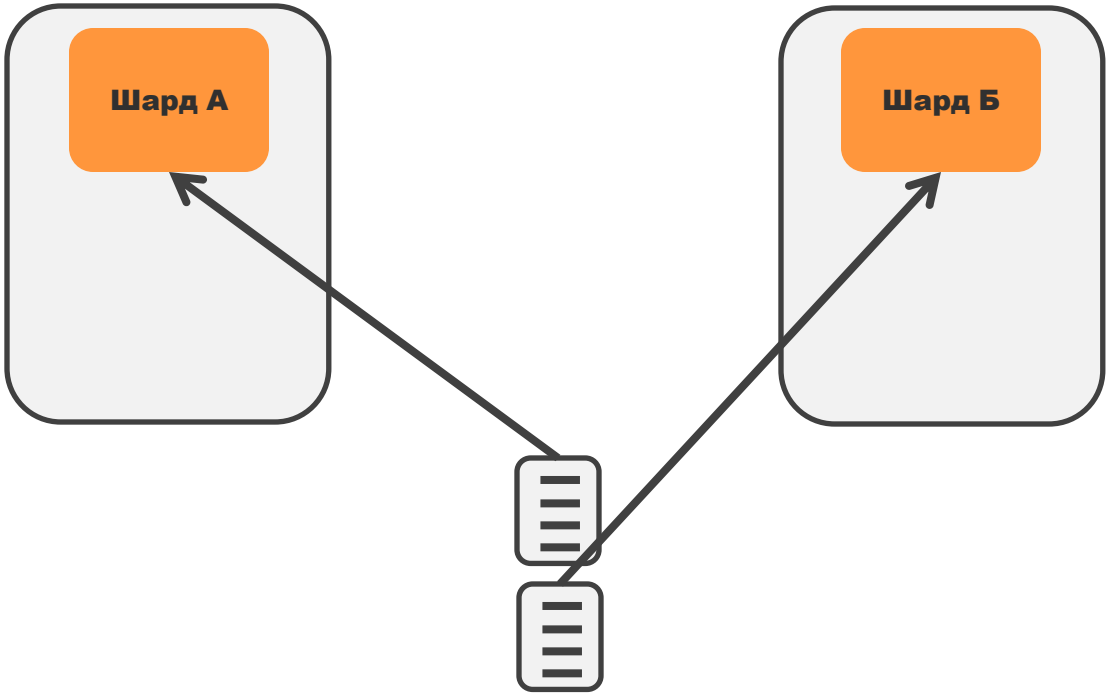
- Покрива напълно изискванията ни за чисто пълнотекстово търсене
- Особености в API-то които бихме предпочели да са изолирани от нас като потребители (например само един IndexWriter)
- Lucene **не е система** за пълнотекстово търсене, но е чудесна основа за такава

elasticsearch.

ElasticSearch - основи

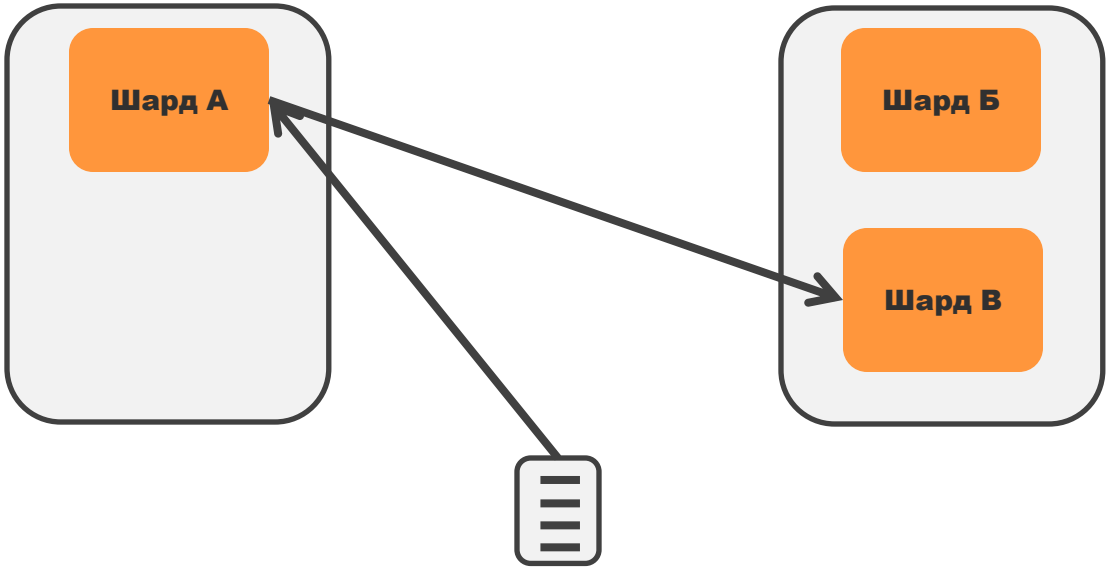
- Разпределена система за търсене базирана на Lucene
- Сравнително нов проект (създаден през 2009), наследник/заместник на Compass framework
- Документите, заявките, отговорите и конфугурацията са в JSON формат
- Търсене в почти реално време ползвайки възможностите на Lucene 2.9+
- Използва се от StumbleUpon, RedHat, Mozilla, Sony и др.

ElasticSearch – разпределена архитектура



- С цел работа с голям обем данни индекса се разбива на парчета (шардове), които се разпределят върху машини свързани в клъстер
- Всеки шард е отделен Lucene индекс. Документите се разпределят между шардовете - един документ се индексира в един шард

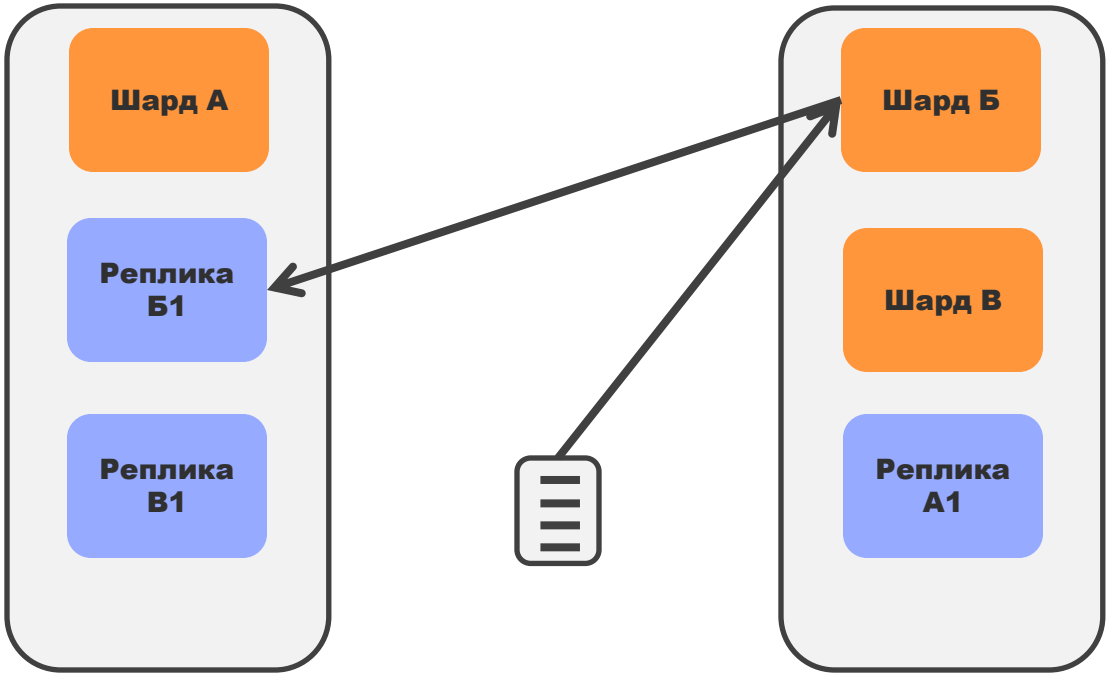
ElasticSearch - индексиране



- При индексиране документите се поставят във съответния шард според идентификатора им. Една машина в клъстера може да съдържа повече от един шард

- Клъстерът подържа вътрешно рутиране на заявките за индексиране, т.е. заявки за индексиране могат да се пращат към всяка машина

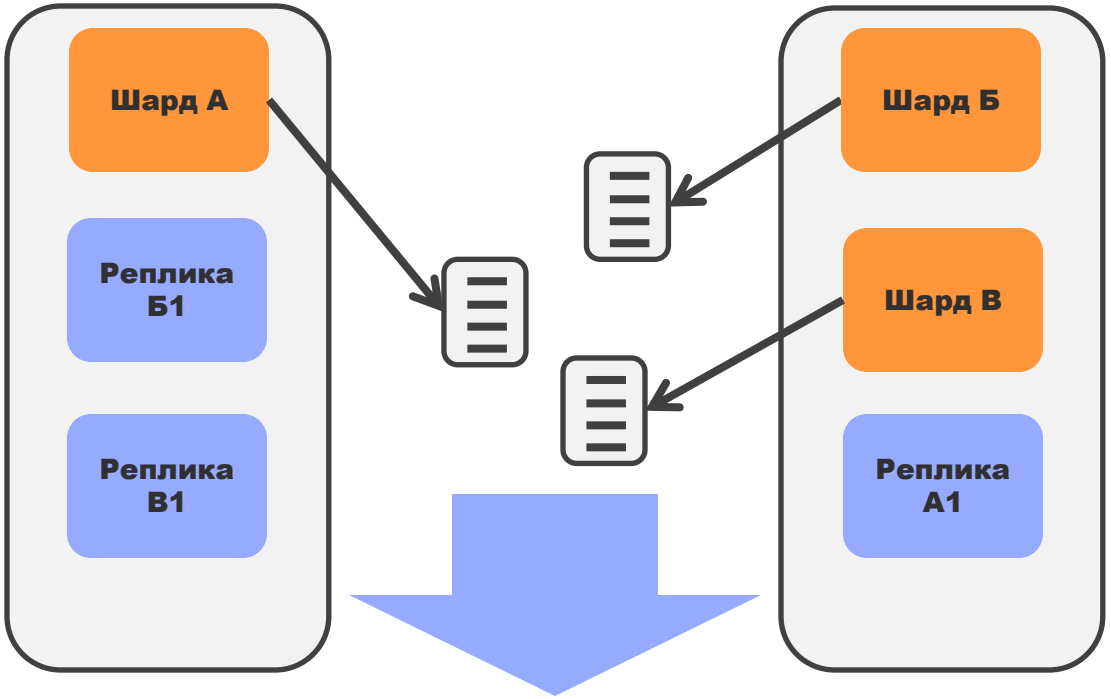
ElasticSearch - реплики



- С цел устойчивост на грешки всеки шард може да има реплики(копия). Повече реплики на повече машини ни дават по-голяма устойчивост

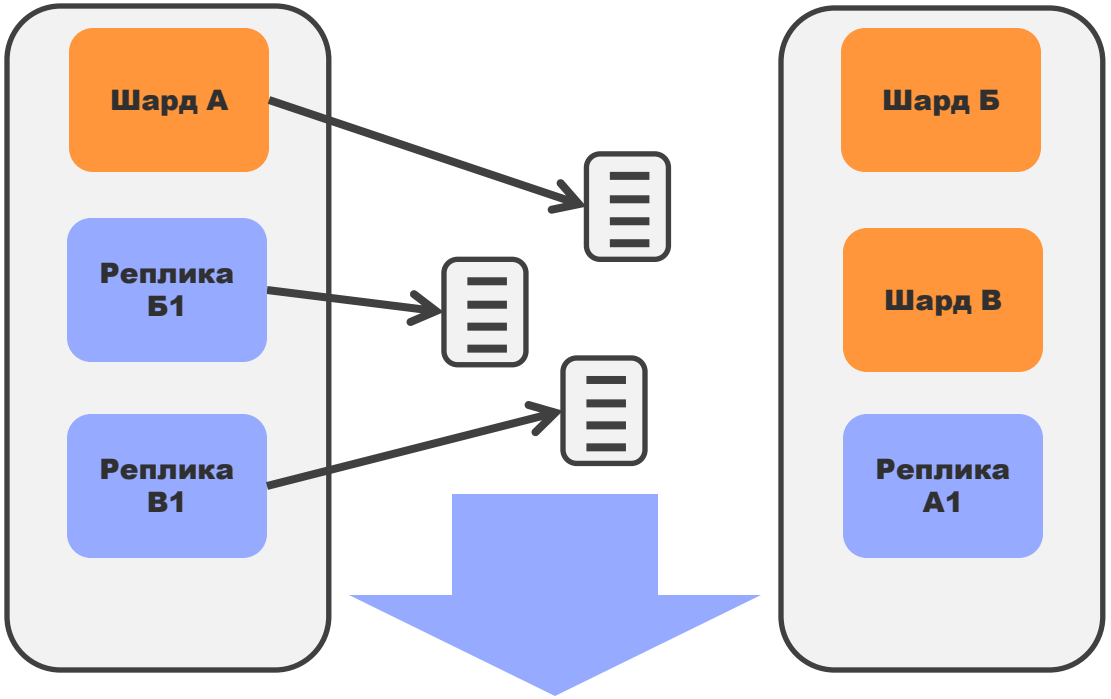
- При индексиране документът се поставя във съответния шард и във всички негови реплики - push replication

ElasticSearch - търсене



- Заявката за търсене автоматично се разпраща до всеки шард, резултатите се обединяват и връщат (ключови думи: Scatter/Gather, MapReduce)

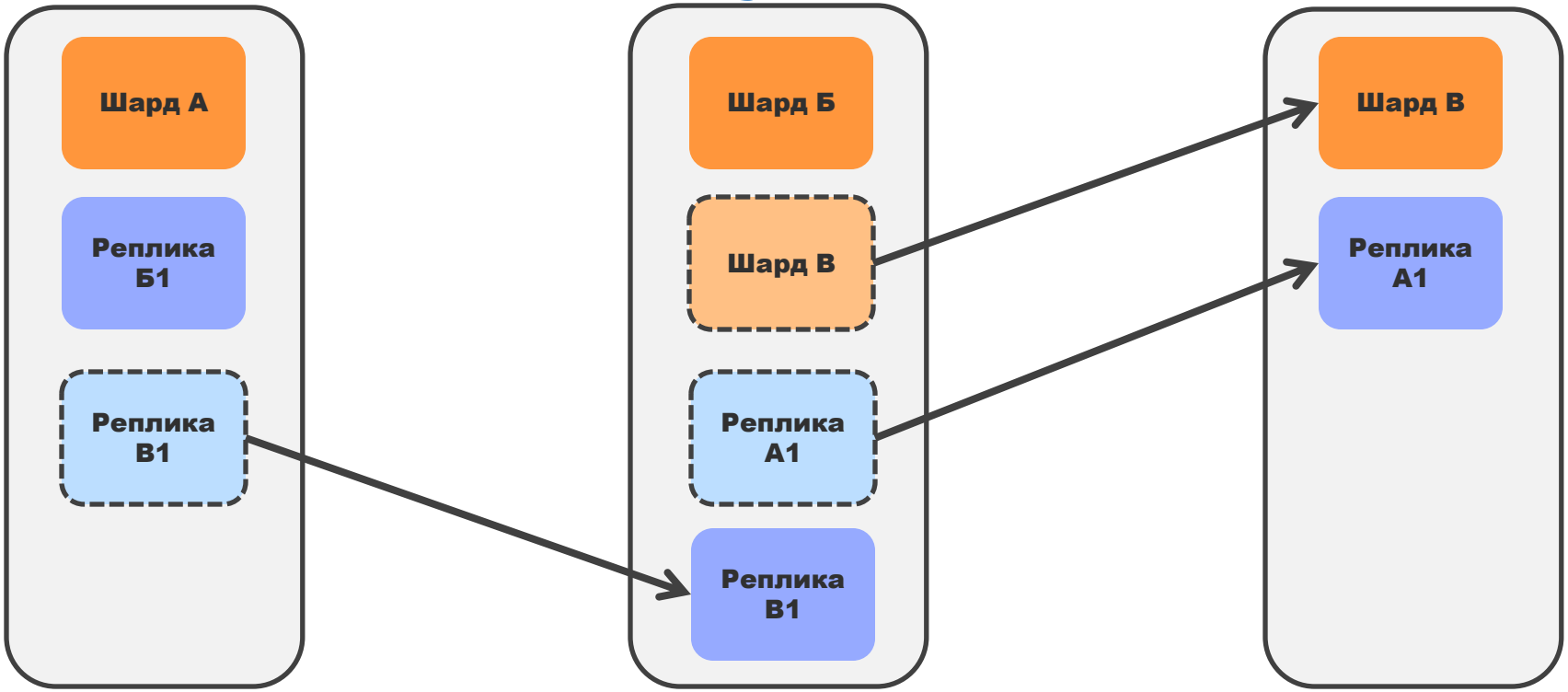
ElasticSearch - търсене



- Заявката за търсене автоматично се разпраща до всеки шард, резултатите се обединяват и връщат (ключови думи: Scatter/Gather, MapReduce)

- Репликите също се ползват за търсене пестейки мрежови ресурси

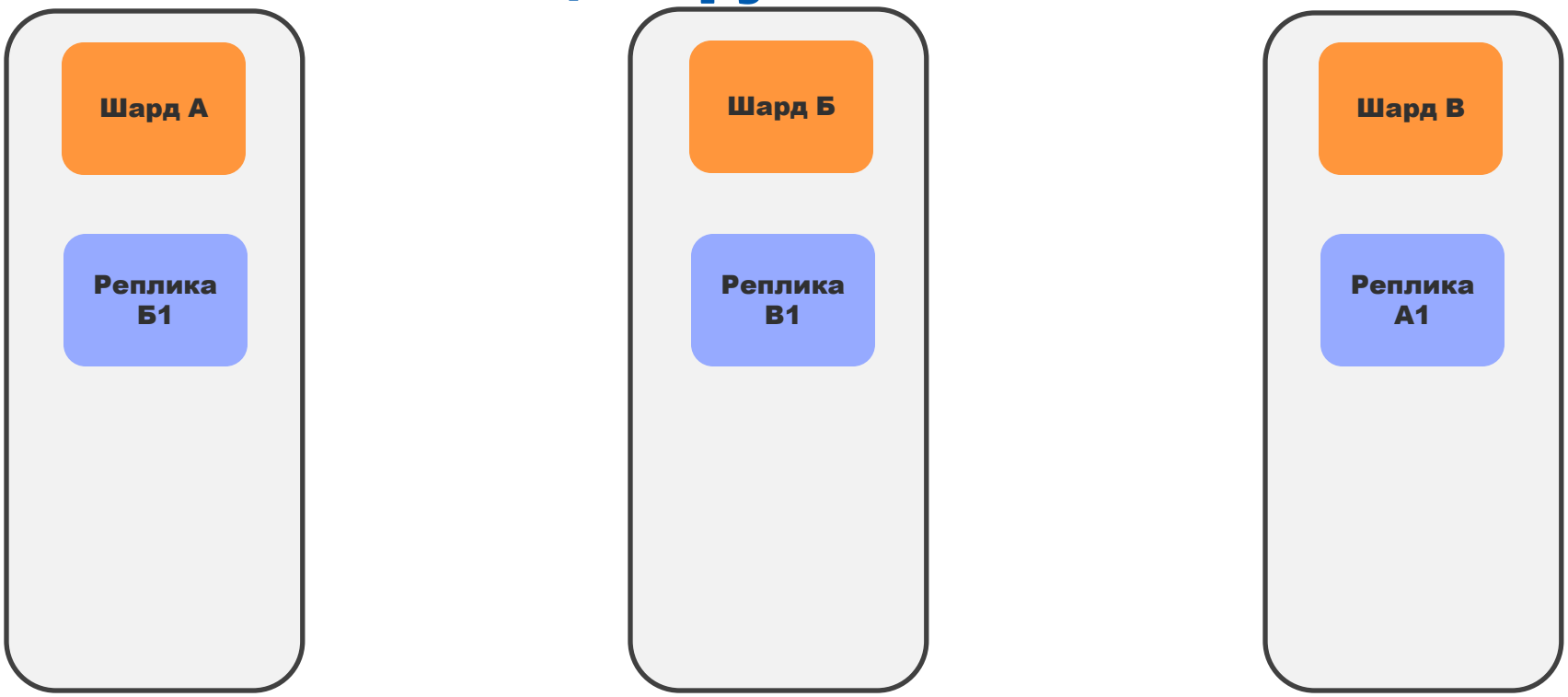
ElasticSearch - мащабируемост



- При добавяне на нова машина в клъстера части от индекса и/или копия може да се преместят автоматично върху нея

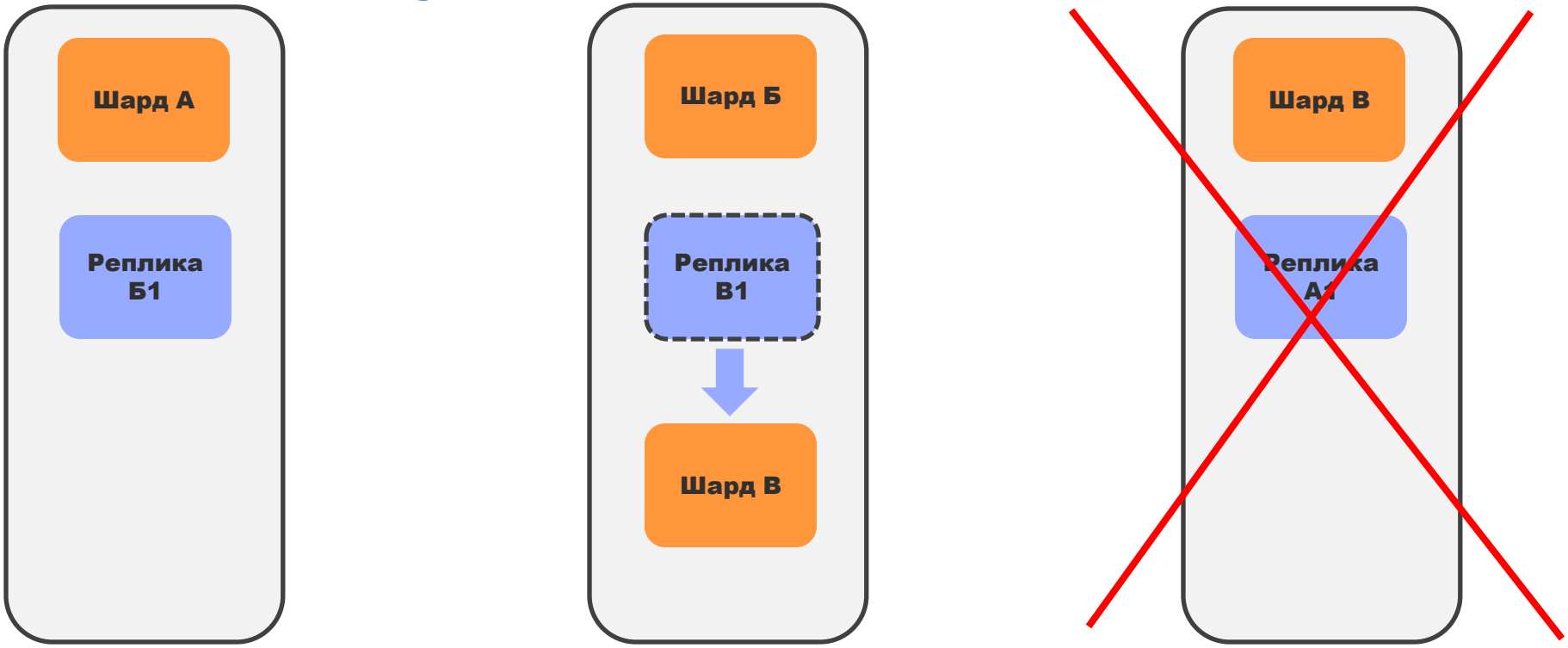
- Работата на клъстера не се прекъсва по време на преместването, използва се история на транзакциите (transaction log)

ElasticSearch - мащабируемост



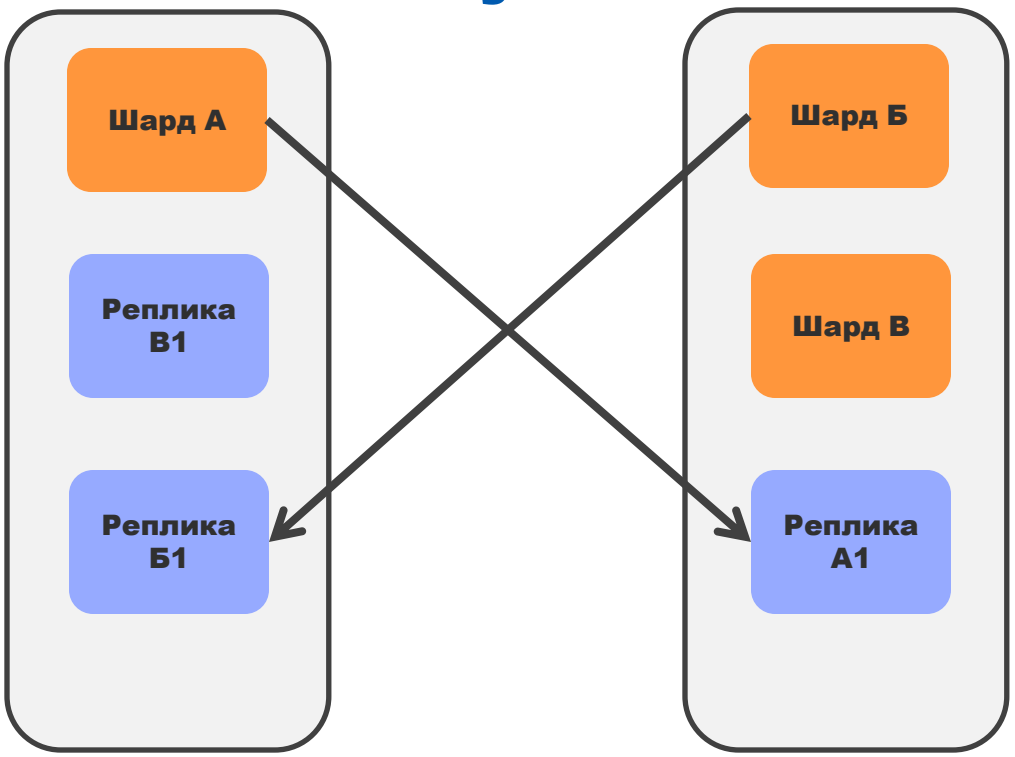
- При добавяне на нова машина в клъстера части от индекса и/или копия може да се преместят автоматично върху нея
- Работата на клъстера не се прекъсва по време на преместването, използва се история на транзакциите (transaction log)

ElasticSearch – устойчивост на грешки



- При отпадане на машина от клъстера репликите от другите машини заемат местата на загубените шардове автоматично

ElasticSearch – устойчивост на грешки



- При отпадане на машина от клъстера репликите от другите машини заемат местата на загубените шардове автоматично

- Клъстера автоматично балансира шардовете и репликите, създавайки реплики при нужда

ElasticSearch – концепция за клъстер

- За да подържа всички функционалности описани досега ElasticSearch обединява машини в клъстер
- Автоматично откриване на машините в клъстера посредством multicast/unicast
- Една от машините в клъстера бива избрана за master и операциите на ниво клъстер(създаване на индекс, ребалансиране) се контролират от нея
- При загуба на master останалите машини в клъстера избират нов master

ElasticSearch – заключение

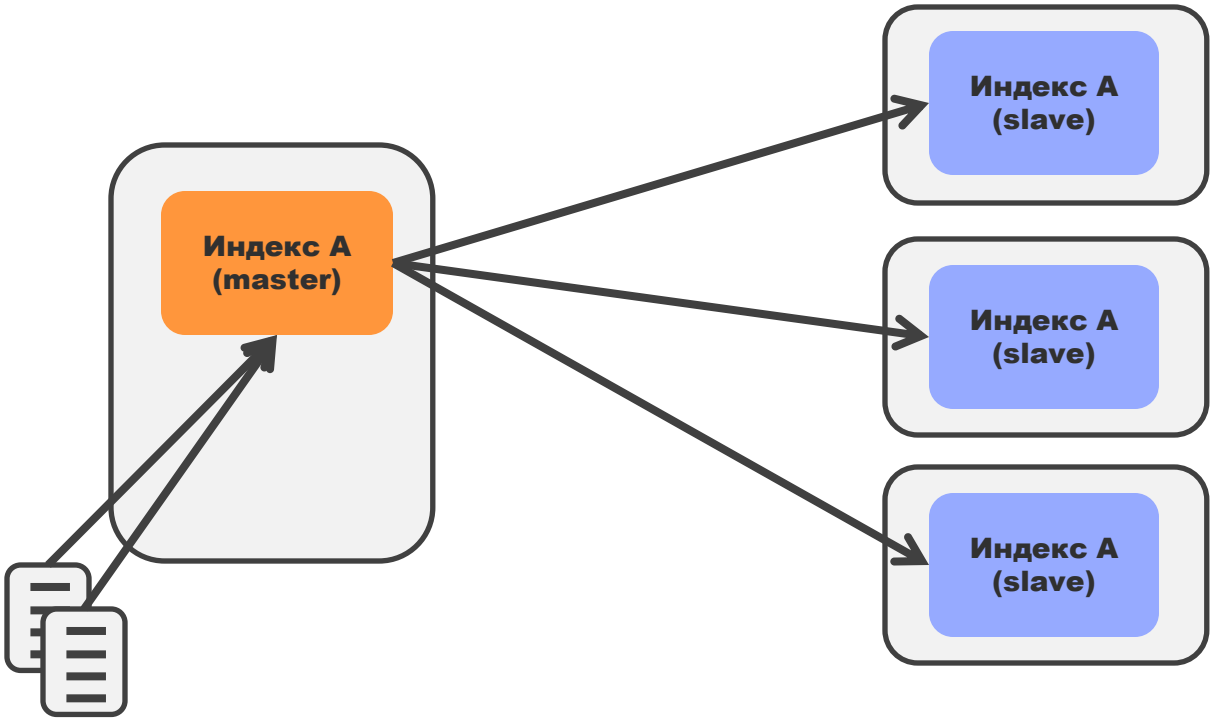
- Проект започнат “на чисто”, с ясна цел разпределено пълнотекстово търсене
- Архитектура изградена според правилата за разпределени системи
- Младостта му носи рискове от дефекти


Apache
Solr

Solr – основи

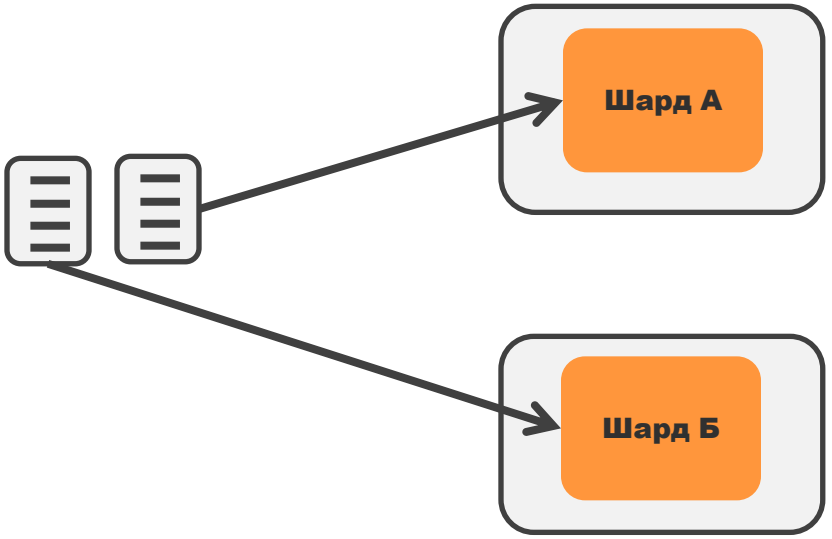
- Първонално създаден от CNET Networks
- От 2006 става Apache проект
- Представлява “сървъризация” на Lucene
- Използва се в много от най-големите сайтове в света – CNet, Netflix, Digg и други [2]

Solr – разпределена архитектура



- С цел обслужване на повече заявки индекса се репликира(копира) върху машини свързане в master/slave конфигурация
- Индексът е ограничен по размер от физическите ресурси на една машина
- Самостоятелно тази архитектура не е подходяща за мащабируемост над 100 милиона документа

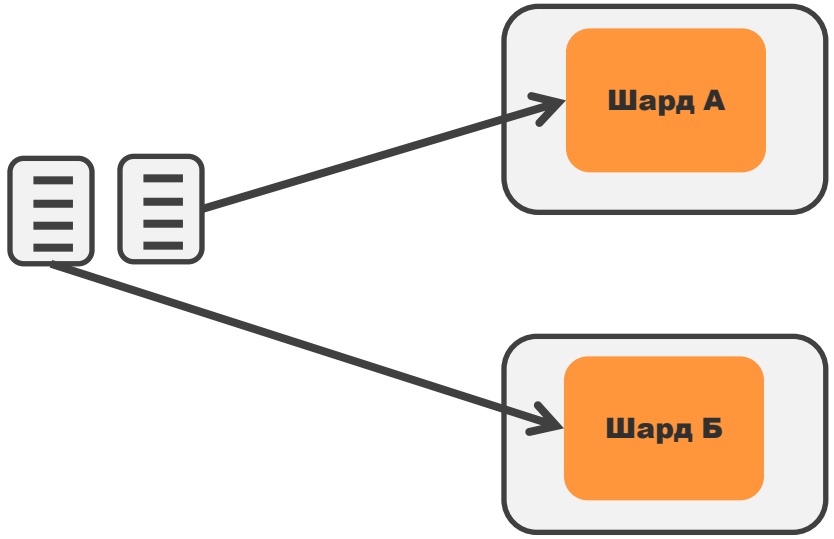
Solr – разпределена архитектура, шардове



- С цел работа с голям обем данни индекса се разбива на парчета(шардове) които се рапределят върху отделни машини **несвързани** помежду си

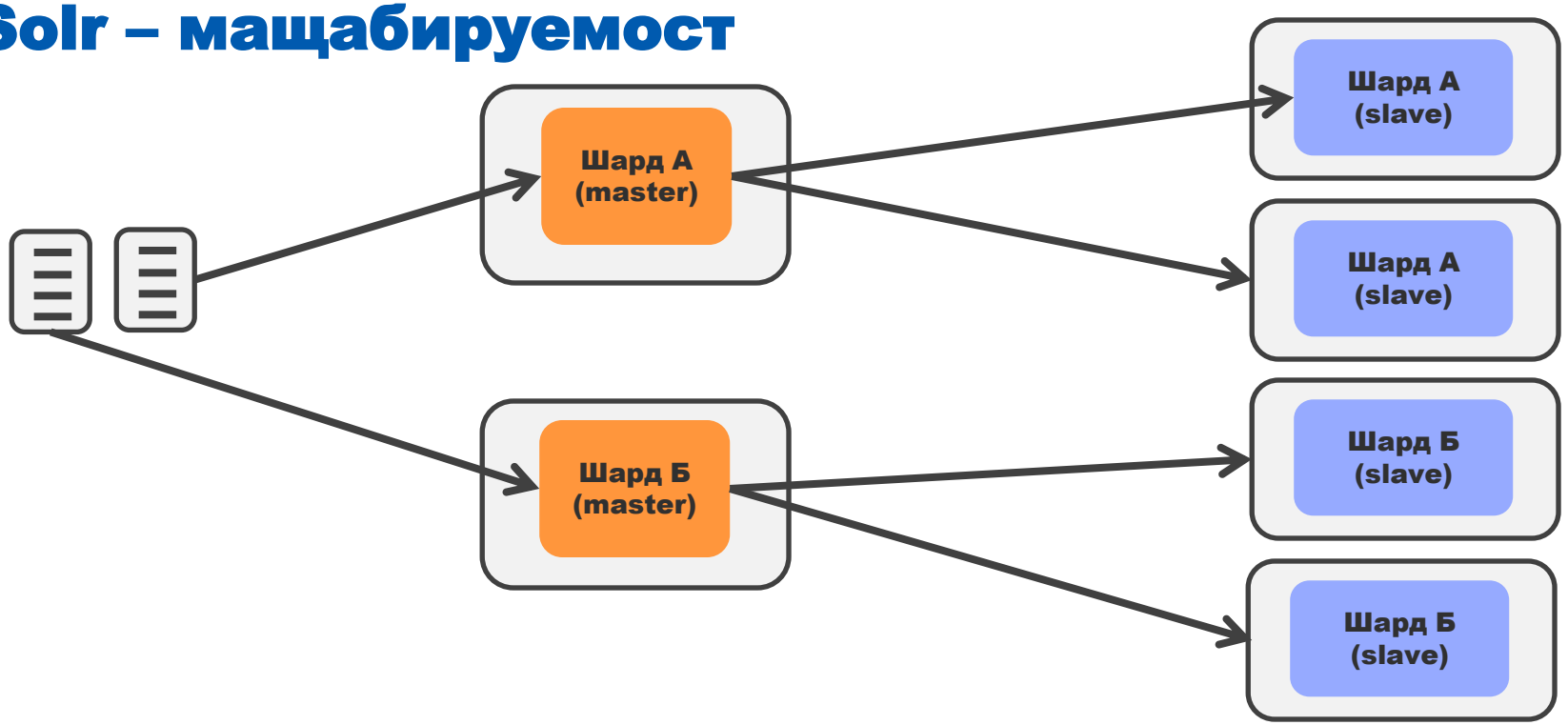
- Всеки шард е отделен Lucene индекс. Документите се разпределят между шардовете - един документ се индексира в един шард

Solr – индексиране



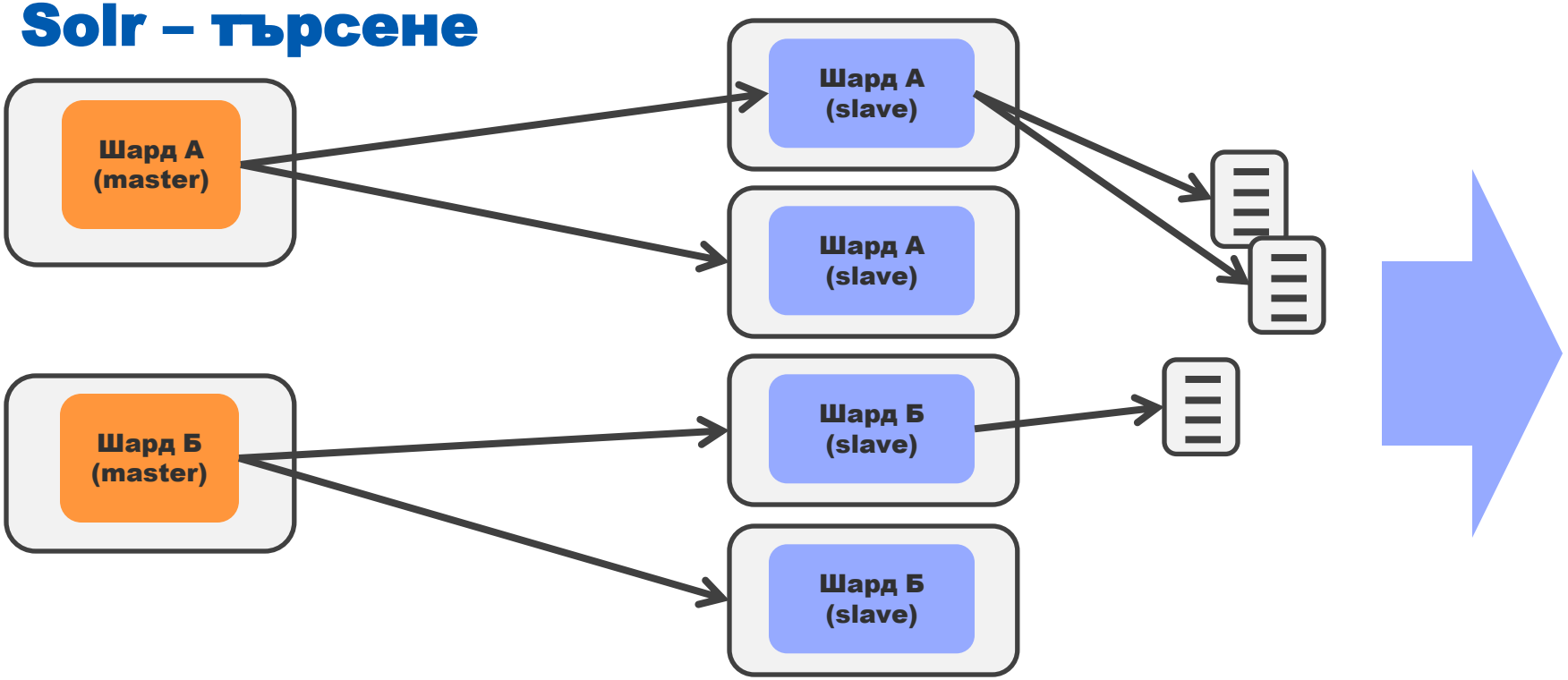
- При индексиране документите се поставят във съответния шард според идентификатора им. Една машина може да съдържа само един шард
- Машините не са свързани в клъстер и разпределението на документите не е автоматично

Solr – мащабируемост



- С цел обслужване на повече заявки всеки шард се репликира(копира) върху машини свързане в master/slave конфигурация
- Slave машините проверяват за нови данни в master машините и изтеглят промените – pull replication
- Прехвърлянето на данни е на ниво Lucene сегменти

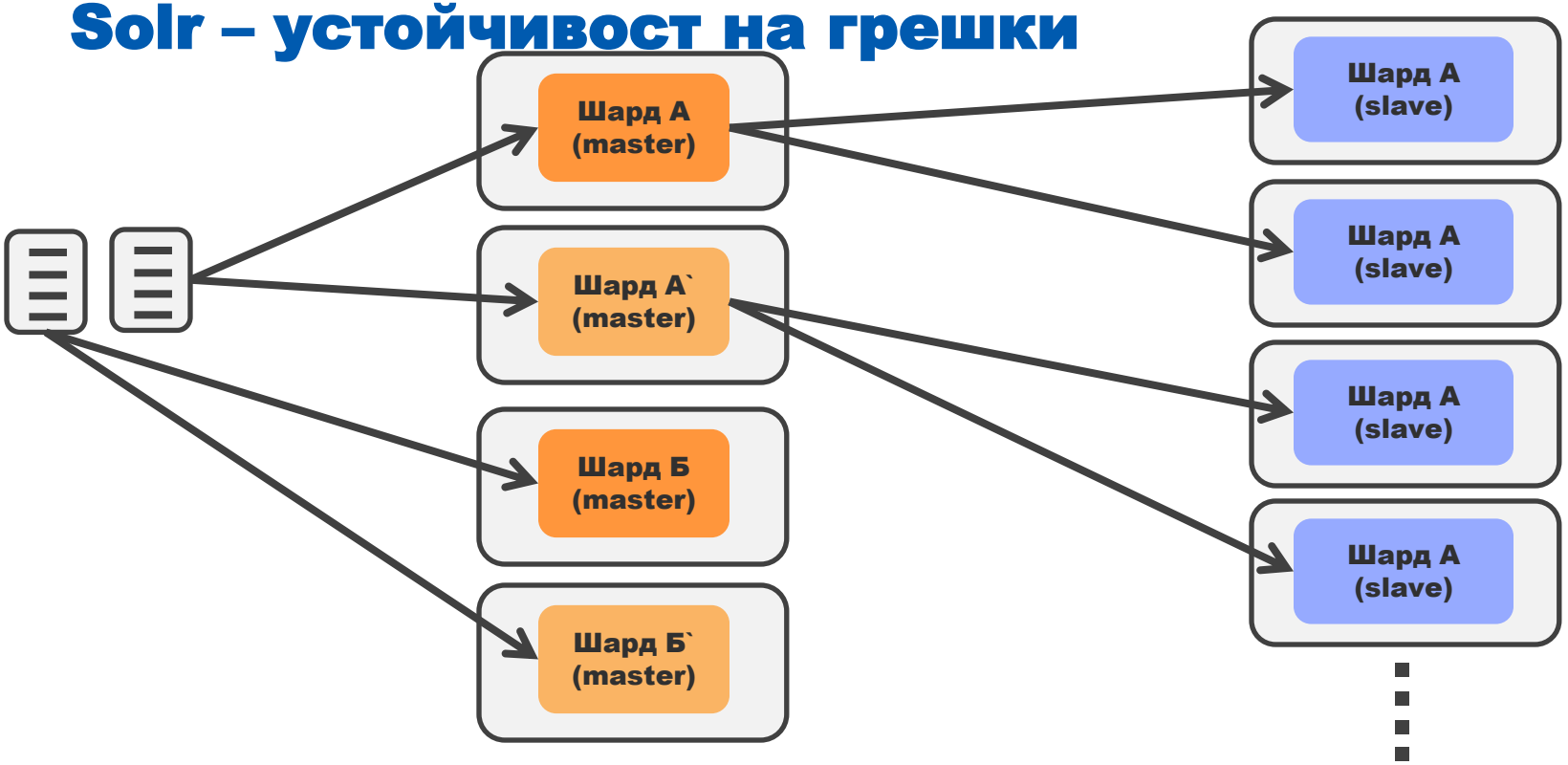
Solr – търсене



- При търсене изрично се указва до кои slave шардове да се разпрати заявката. Резултатите се обединяват и връщат (ключови думи : Scatter/Gather, MapReduce)

- Не всички видове заявки, които Solr подържа, могат да работят режим на шардове + master/slave конфигурация

Solr – устойчивост на грешки



- За да подсигурим данните индексираме един документ в повече от един master шард
- Slave машините се разполагат зад load balancer
- При отпадане на машина е необходима ръчна намеса или наличие на допълнителен слой от приложението грижещ се за възстановяването ú

Solr - SolrCloud

- За да подобри работата в разпределена среда Solr развива надстройката SolrCloud
- Интеграция на ZooKeeper за подръжка на клъстер от Solr машини
- Очаква се да е налично в Solr 4.0
- Начален стадий на разработка, концепция доста сходна с ElasticSearch

Solr – заключение

- Стабилна система с много години работа в реални условия
- Архитектурно принадлежи към по-старото поколение разпределени системи

Очи в очи - ElasticSearch срещу Solr

Изисквания	ES	Solr
Гъвкава документна структура	✓	✓
HTTP интерфейс	✓	✓
Висока производителност	✓	✓
Мащабируемост над 100м	✓	✓
Устойчивост на грешки	✓	✓*
Четене веднага след запис	✓	
Търсене в почти реално време	✓	

Гъвкава документна структура, ElasticSearch

- Документ в ElasticSearch се представя чрез JSON и се индексира през REST интерфейса

```
curl -XPUT http://localhost:9200/myindex/message/1234 -d '{
  "name": "Иван Иванов",
  "date": "2011-10-19",
  "message": "Индексират ме",
  "age": 30
 }'
```

- Не е необходимо да се указва типа на полетата предварително, разчита се на типизацията на JSON

- Типизация на полетата може да се зададе изрично при създаване на индекса, включително и wildcard правила

Гъвкава документна структура, Solr

- Документ в Solr се представя чрез XML и се индексира през REST интерфейса

```
curl http://localhost:8983/solr/update -H "Content-Type:
text/xml" -d
'<add>
        <doc>
          <field name="id">1234</field>
          <field name="name">Иван Иванов</field>
          <field name="date">2011-10-19T00:00:00Z</field>
          <field name="message">Индексират ме</field>
          <field name="age">30</field>
        </doc>
</add>'
```

- Не е необходимо да се указва типа на полетата предварително, но задължително трябва да има поне правила за динамично типизиране на полетата

HTTP заявки и фасети, ElasticSearch

- Заявките за търсене също са в JSON формат

```
curl -XGET http://localhost:9200/myindex/message/_search -d '{
        "query" : {
                "term" : { "name": "иван" }
        },
        "facets" : {
                "age_facet" : {
                        "range" : {
                                "field" : "age",
                                "ranges" : [
                                        { "from" : 0, "to" : 20 },
                                        { "from" : 20, "to" : 50 },
                                        { "from" : 50, "to" : 120 }
                                ]
                        }
                }
        }
}'
```

- Подръжка на различни типове фасети, по-горе виждаме range фасет

HTTP заявки и фасети, Solr

- Заявките към Solr се подават като HTTP параметри

```
curl -XGET
http://localhost:8983/solr/select?q=name:иван&facet=true&facet.field=age&facet.quer
y=age:[0+TO+20]&facet.query=age:[21+TO+50]&facet.query=age:[51+TO+120]
```

- Подръжка на различни типове фасети, по-горе виждаме range фасет

Висока производителност

- В нашите тестове и двете системи обслужваха над 200 заявки за търсене в секунда
- И двете системи индексираха над 400 документа в секунда
- И при индексирането и при търсенето Solr беше по-бърз от ElasticSearch

Мащабируемост, ElasticSearch

- Архитектурата на ElasticSearch определя лесната му мащабируемост чрез добавяне на нови машини в клъстера

- Броят шардове не може да се променя след създаване на индекса, при нарастване извън първоначално планираното трябва да се създаде нов индекс и да се преместят данните

Мащабируемост, Solr

- Разбиването на индекса на шардове в Solr постига мащабируемостта
- Нуждае се от допълнителна функционалност, която да разпределя документите по шардовете
- Предимство е, че лесно може да се добавят нови шардове без нужда от преиндексиране

Устойчивост на грешки, ElasticSearch

- Репликите гарантират устойчивостта на грешки. По-голям брой реплики забавя индексирането, но забързва търсенето. Броят реплики може да се променя динамично

- SplitBrain проблем и настройката "minimum_master_nodes“

- Забавяне на избора на master в клъстера и настройката "ping_timeout"

Устойчивост на грешки, Solr

- Паралелните master-slave конфигурации за един и същи шард (индекс) решават проблема

- Все пак е необходим допълнителен слой за поддържане на напълно автоматична устойчивост на грешки

Четене веднага след запис, ElasticSearch

- ElasticSearch прави документа наличен за директно четене веднага след индексиране (ключова дума: Realtime GET)

- Позволява ползването на ElasticSearch като основен слой за данни

Четене веднага след запис, Solr

- В Solr документа все още не е наличен за четене веднага след индексиране

- Очаква се тази функционалност да се появи в Solr 4.0

Търсене в почти реално време, ElasticSearch

- ElasticSearch е изграден изцяло върху Lucene 2.9+, търсенето в почти реално време се подържа от самото начало на проекта

- Нов Lucene IndexReader се отваря периодично (1 секунда по подразбиране)

Търсене в почти реално време, Solr

- Официалната версия Solr все още не поддържа търсене в почти реално време и новите индексирани документи не са налични преди commit на IndexWriter-а и отваряне на нов IndexReader

- Също се очаква да се поддържа в Solr 4.0

Заключение

- С оглед на изискванията, които поставихме по-рано, изборът ни пада върху ElasticSearch

- Фактори
  - Лесна мащабируемост
  - Устойчивост на грешки без допълнителна работа от наша страна
  - Подръжка на директно четене веднага след индексиране

Благодарим ви!

Въпроси?

Връзки

- [1] Bulstem - http://people.ischool.berkeley.edu/~nakov//bulstem/index.html
- [2] Public Websites using Solr - http://wiki.apache.org/solr/PublicServers
- ElasticSearch - http://www.elasticsearch.org
- Apache Solr - http://lucene.apache.org/solr
- SolrCloud and NRT Search - http://blog.sematext.com/2011/09/14/solr-digest-spring-summer-2011-part-2-solr-cloud-and-near-real-time-search/
- Scaling Lucene and Solr - http://www.lucidimagination.com/content/scaling-lucene-and-solr