AGH University of Science and Technology

ElasticSearch - NoSQL DataBase

Marcin Kopacz

Department of Automatics

November 13, 2012

ElasticSearch - NoSQL DataBase

Bazą danych

Bazą danych nazywamy zbiór danych o określonej strukturze, zapisany na zewnętrznym nośniku pamięciowym komputera, mogący zaspokoić potrzeby wielu użytkowników korzystających z niego w sposób selektywny w dogodnym dla siebie czasie (pierwsza definicja sformułowana w 1962-1963).

ElasticSearch - NoSQL DataBase

Początki

Wraz z powstawaniem i rozwojem technologi informatycznych pojawiła się potrzeba przechowywania i zarządzania olbrzymimi ilościami danych. Pojawiły się wtedy dwa podstawowe modele danych: sieciowy i hierarchiczny. Były ona niedoskonałe i ich głównymi wadami były nadmierność danych, słaba integralność i silna zależność od fizycznej implementacji. W 1970 E. F. Codd zaproponował relacyjny model danych, który był pozbawiony wad ówczesnych modeli.

Relacyjne Bazy Danych - zalety

- Olbrzymia i niegasnąca popularność na rynku.
- Posiadają prostą budowę opierającą się na relacjach i powiązaniach pomiędzy nimi, co jest naturalnym sposobem odbierania danych przez człowieka.
- Operacje na rekordach są bardzo proste.
- Są oparte na teorii relacji co daje solidny i łatwy zobrazowania fundament teoretyczny.
- Występuje w nich bardzo duża niezależność fizyczna i logiczna danych.

Relacyjne Bazy Danych - wady

- W procesie modelowania tracimy informację o tym, że w świecie rzeczywistym istnieją dwa identyczne obiekty.
- „Sztuczny” język opisu rzeczywistości – są nim dwuwymiarowe tabelki.
- Trudności w przechowywaniu informacji zmiennych w czasie.
- Trudności w przechowywaniu informacji niepełnej,
- Rozbudowane systemy posiadają tabele z setkami pól, ale często tylko część z nich jest istota w danym rekordzie czy zapytaniu.
- Proces normalizacji bazy prowadzi do tworzenia wielu relacji, co powoduje, że struktura danych przypomina drzewo. Pociąga to za sobą ogromne konsekwencje w kwestii wydajności (niestety głównie negatywne).

Skalowanie

Skalowanie

Skalowalność staje się jednym z ważniejszych zagadnień jeśli wcześniej projektowy system zaczyna mieć problemy z wydajnością (gwałtownie wzrasta ilość użytkowników i danych). Jest to powszechne zjawisko dotykające wiele obecnych systemów.

Istnieją dwa podejścia do tego zagadnienia:

- skalowanie poziome (ang. scalling out)
- skalowanie pionowe (ang. scalling up)

Skalowanie pionowe

Skalowanie pionowe polega na dodawaniu do serwera lepszego procesora (szybszego, z większą liczbą rdzeni), zwiększeniu pamięci operacyjnej, wprowadzeniu szybszych kanałów dyskowych (IDE SATA SCSI), itd.

Ten typ rozbudowy, poza kosztami poniesionymi na jego zakup i wdrożenie nie wiąże się z żadnymi dodatkowymi nakładami, a jednocześnie od razu pozwala cieszyć się lepszą wydajnością.

Skalowanie poziome

Polega na dodawaniu kolejnych serwerów. Większość SZBD umożliwia stosowanie takiego rozwiązania, wiąże się to jednak z koniecznością zakupu dodatkowych licencji oraz serwerów.

Jest również trudniejsze we wdrażaniu i utrzymaniu, a co najważniejsze dodawanie kolejnych serwerów nie zawsze przynosi oczekiwane rezultaty.

Skalowanie poziome - replikacji

Najpopularniejsze podejścia do poziomego skalowania to replikacja w oparciu o schemat master-slave. Istnieje jeden węzeł master, który odpowiada za obsługę operacji modyfikacji danych oraz kilka węzłów slave, które komunikują się z węzłem głównym aktualizując posiadane w replikach dane. Ich zadaniem jest wykonywanie operacji odczytu.

Drugim sposobem replikacji jest podział bazy na części (and. sharding). Polega to na wyłączeniu grup tabel używanych razem w zapytaniach i przenoszeniu ich na osobne serwery. Wymaga to zmian w logice aplikacji i nie zawsze jest możliwe do zastosowania.

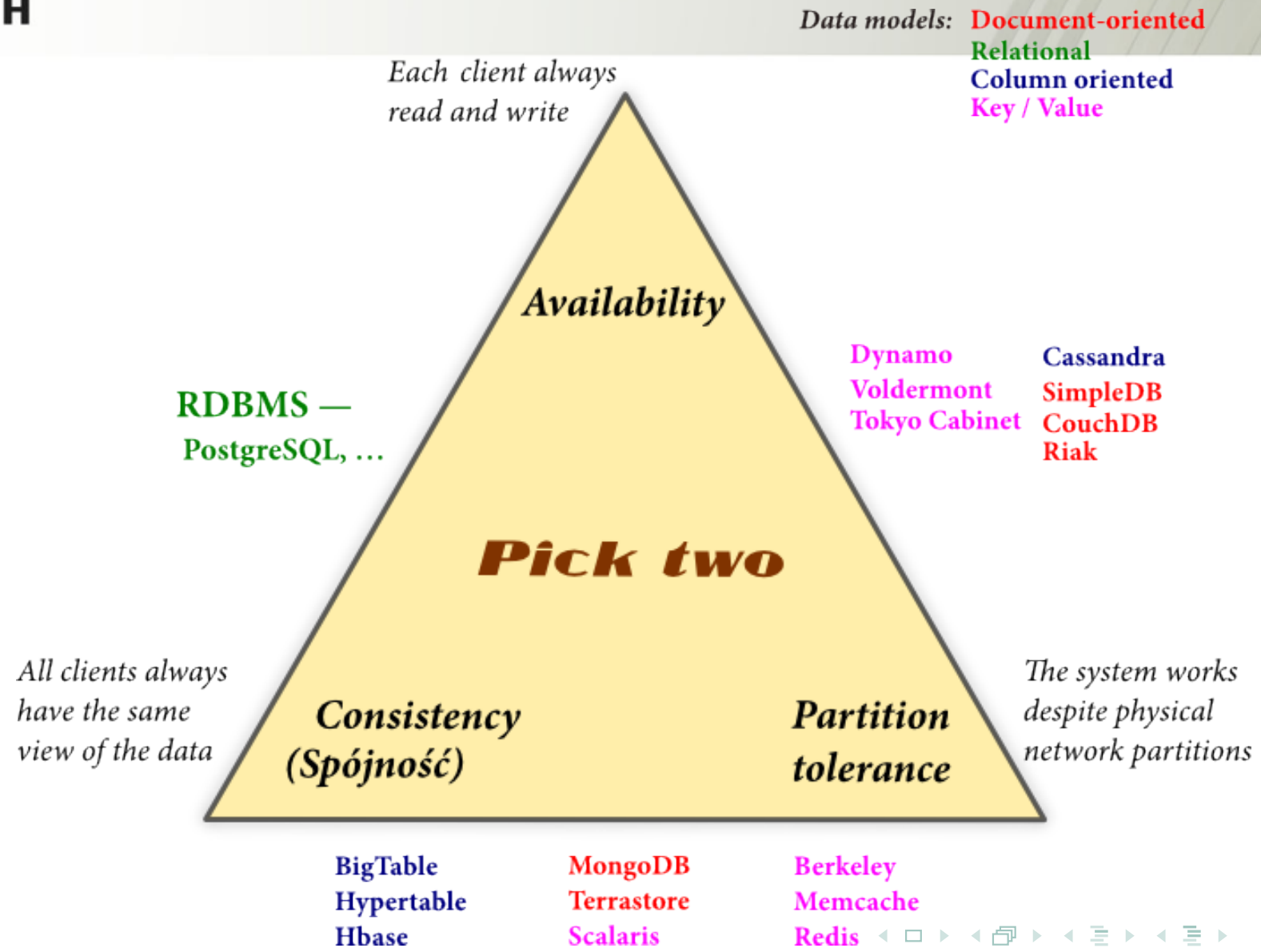
Twierdzenie CAP

Wysunięte przez Erica Brewera w 2000 roku podczas Symposium on Principles of Distributed Computing.

System rozproszony nie może jednocześnie zapewniać:

- spójności (ang. consistency) - otrzymywanie identycznych odpowiedzi na zapytania kierowane w tym samym kwancie czasu do różnych węzłów,
- dostępności (ang. availability) - nieuszkodzony węzeł systemu musi niezwłocznie wysłać odpowiedź na każde zapytanie, jakie otrzyma,
- odporności na podziały (ang. partition tolerance) — utrata wiadomości w przesyle nie może skutkować utraceniem spójności lub dostępności.

Twierdzenie CAP cd.



Twierdzenie CAP cd.

Relacyjne bazy danych zapewniają spójność i dostępność, czego kosztem są problemy ze skalowalnością na wielu maszynach. Obecne systemy przetwarzają olbrzymie ilości informacji, dlatego dalsza skalowalność jest niezbędna.

Bazy danych NoSQL pozwalają rozwiązać problem skalowalności jednak za cenę utraty części spójności bądź dostępności.

ElasticSearch - NoSQL DataBase

Co to jest „NoSQL”?

Termin „NoSQL” jest rozwinięciem “Not only SQL i oznacza szeroką gamę systemów zarządzania bazami danych, identyfikowanych poprzez brak przynależności do systemów opartych o model relacyjny.

Termin ten został wyprowadzony przez Carlo Strozzi w 1998, a w 2009 roku Eric Evans użył tego terminu w kontekście „the emergence of a growing number of non-relational, distributed data stores”.

ElasticSearch - NoSQL DataBase

„NoSQL” DataBases

- Nie są zbudowane na tablicach (relacjach).
- Generalnie, nie używają języka SQL (structured query language) do manipulacji danymi.
- Zwykle nie używają operacji złączeń.
- Umożliwiają łatwe skalowanie w poziomie.
- Najczęściej wysoce zoptymalizowanie w kierunku operacji pobierania i dołączania.
- Często oferują małą funkcjonalność związaną z przechowywaniem danych (np. poprzez klucz-wartość)

ACID

Transakcje w relacyjnych bazach posiadają szereg właściwości określonych jako ACID:

- atomowość - każda transakcja jest jedną jednostką przetwarzania, co oznacza, że wykonuje się w całości albo nie wykonuje się w ogóle.
- spójność - żadne wykonanie transakcji nie jest w stanie naruszyć zasad integralności systemu.
- izolacja - współbieżnie wykonujące się transakcje nie widzą wprowadzonych przez inne transakcje zmian, aż do zakończenia ich wykonywania.
- trwałość - gwarancja, że zmiany wprowadzone przez zakończoną sukcesem transakcje są na stałe zapisane w systemie i dostępne.

BASE

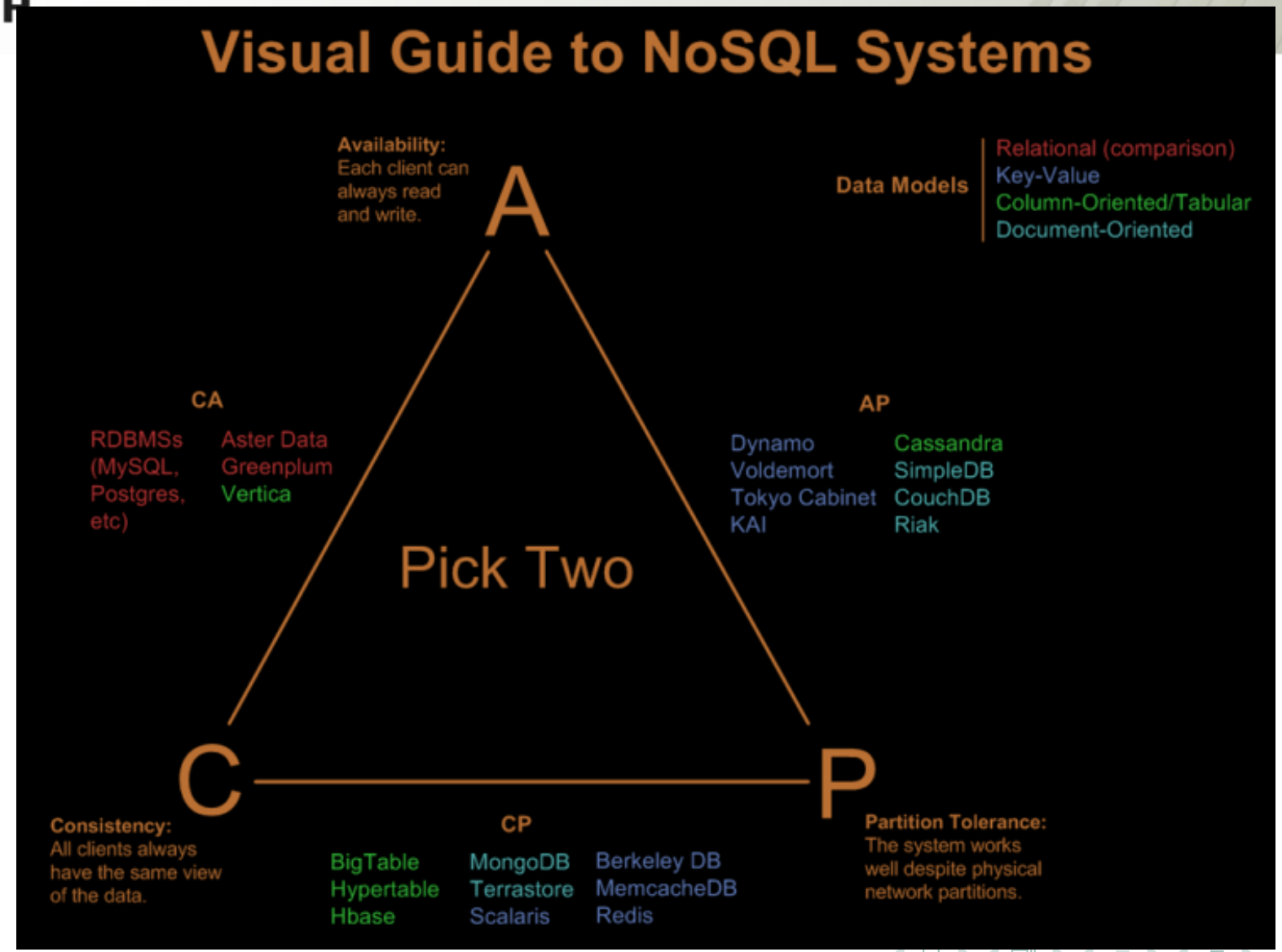
Podejście w całkowitej opozycji do wcześniejszego to BASE, które opiera się na :

- podstawowa dostępność - system pozostaje dostępny, ale nie musi koniecznie zapewniać dostępności wszystkich elementów w każdej chwili czasu.
- niestały stan - wprowadzone informacje, mogą one ulec automatycznej zmianie lub usunięciu, gdy nie zostaną w ustalonym czasie odświeżone przez użytkownika, innymi słowy informacje mogą stracić ważność.
- ostateczna spójność - dane w bazie nie zawsze są spójne. Zakładamy jednak, że po odpowiednio długim czasie bez nowych zamian wszystkie repliki w węzłach będą identyczne.

Podział względem struktury danych:

- Bazy dokumentowe (ang. document stores) (CouchDB, MongoDB, OrientDB, ElasticSearch, Terrastore, SimpleDB)
- Bazy grafowe (ang. graph stores) (Triplestores: Virtuoso)
- Bazy klucz-wartość (ang. key-value stores) (Dynamo, Cassandra, Voldemort, Riak, Redis, Membase, Membrain)
- Bazy obiektowe (ang. object databases) (Loxim)
- Bazy kolumnowe (ang. tabular stores) (BigTable, Mnesia, HyperTable, ScaleDB)
- Przestrzenie krotek (ang. tuple stores) (Jini, Apache River - JavaSpaces)

ElasticSearch - NoSQL DataBase



ElasticSearch - NoSQL DataBase

ElasticSearch - NoSQL DataBase

- Jest to rozproszony, open source'owy silnik wyszukiwania oparty na bibliotece Apache Lucene.
- Został napisany od podstaw przez Shay Banon'a jako "skalowalne rozwiązanie do wyszukiwań" dla Projektu Compose.
- Pierwsza wersja została wydana w lutym 2010 roku.
- W pełni wykorzystuje wzorzec architektury REST (ang. Representational State Transfer).
- Jest napisany w języku Java i rozpowszechniany zgodnie warunkami licencji Apache.

Representational State Transfer (REST)

REST - jest to styl architektury oprogramowania dla systemów rozproszonych takich jak sieć WWW. Stał on się dominującym modelem dla Web serwisów. Jest efektem wielu lat doświadczeń przy projektowaniu protokołu HTTP.

Model ten opisuje charakterystyki i ograniczenia makro-interakcji pomiędzy czterema elementami sieci, mianowicie: bazowymi serwerami, portami, serwerami proxy i klientami, bez zakładania ograniczeń na poszczególnych uczestników. REST zasadniczo reguluje prawidłowe zachowanie tych uczestników (skalowanie interakcji, powszechność interfejsu, niezależność rozmieszczenia elementów, elementy pośredniczące w celu zmniejszenia opóźnień, egzekwowanie bezpieczeństwa i hermetyzacji starszych systemów).

ElasticSearch - NoSQL DataBase

ElasticSearch - indeksowanie

ElasticSearch jest w stanie osiągnąć bardzo szybką odpowiedź ponieważ zamiast wyszukiwania bezpośrednio poprzez tekst używa wyszukiwania po indeksach. Indeksowanie odbywa się przez indeks inwersyjny. Jego działanie można zobrazować do pobierania stron w książce związanych z pewnym kluczem poprzez skanowanie indeksu na końcu książki zawierającego podane słowo klucz.
Do tworzenia tych indeksów wykorzystywane są bezpośrednio rozwiązania zawarte w bibliotece Apache Lucene

ElasticSearch - reprezentacja danych

Dane w ElasticSearch reprezentowane są wyłącznie w postaci JSON (ang. JavaScript Object Notation). Również większość konfiguracji jest zapisana w tym formacie (część jest w YAML-u). Poniżej przykład opisania książki w tym języku.

```
{
    "book" : {
        "isbn" : "0812504321"
        "name" : "Call of the Wild",
        "author" : {
            "first_name" : "Jack",
            "last_name" : "London"
        },
        "pages" : 128,
        "tag" : ["fiction", "children"]
    }
}
```

ElasticSearch - główne elementy

- Cluster
- Node
- Shard and Replica
- Index
- Dokument

ElasticSearch - struktura danych

W ElasticSearch'u jednostką wyszukiwania i indeksowania jest dokument. Indeks składa się z jednego lub więcej dokumentów, a dokument może składać się z jednego lub więcej pól. W terminologii bazodanowej, dokument odpowiada wierszu tabeli, a pole odpowiada kolumnie tabeli. Pole tworzy para klucz-wartość.

Dzięki temu rozwiązaniu ElasticSearch nie ma sztywnego schematu, przynajmniej w teorii. W praktyce chociaż nie jest wymagane aby określić schemat przed indeksowaniem dokumentu, to jednak niezbędne jest dodanie deklaracji mapowania jeżeli wymagamy czegokolwiek ponad najbardziej podstawowe pola i operacje.

ElasticSearch - struktura danych

Mapowanie opisuje:

- jakie pola występują,
- które pola powinny być używane jako klucze unikalne i podstawowe,
- które pola są wymagane,
- jak indeksować i przeszukiwać każde pole,

ElasticSearch - reprezentacja danych

W ElasticSearch, indeks może przechowywać dokumenty różnych "typów mapowania". Można skojarzyć wiele definicji mapowania dla każdego typu mapowania. Typ mapowania jest sposób rozdzielania dokumentów w indeksie w logiczne grupy.

ElasticSearch - NoSQL DataBase

Moduły

ElasticSearch składa się z kilkunastu modułów, z których każdy odpowiada za wydzieloną funkcjonalność:

- Discovery (wykrywanie na bieżąco nodów w clustrze)
- Gateway (utrwalanie zmian w metadanych clustra)
- HTTP (udostępnienie ElasticSearch'owego API poprzez HTTP)
- Transport (wewnętrzna komunikacja pomiędzy nodami)
- Network (udostępnianie ustawień sieciowych pomiędzy nodami)

ElasticSearch - NoSQL DataBase

- Indices (kontrola globalnie zarządzanych ustawień wszystkich indeksów)
- Cluster (zarządzanie alokacją shardów dla nodów)
- Scripting (obsługa wykonywania skryptów)
- Thread Pool (zarządzanie pulami wątków i ich konsumpcją pamięci)
- Node (kontrola zaawansowanych ustawień nodów)
- Plugins, JMX, memcached, Thrift

ElasticSearch - instalacja

- Pobieramy i rozpakujemy paczkę oficjalnej dystrybucji ElasticSearch'a.
- Uruchamiamy z konsoli "bin/elasticsearch -f" dla unix'a lub "bin/elasticsearch.bat" dla systemu windows.
- W oddzielnej konsoli testujemy działanie za pomocą komendy "curl -X GET http:/localhost:9200/". Jeśli instalacja zakończyła się sukcesem otrzymamy informacje w stylu:

{ "ok" : true, "status" : 200, "name" : "Tommy Lightning",
"version" : { "number" : "0.19.11", "snapshot_build" : false },
"tagline" : "You Know, for Search" }

ElasticSearch - Tworzenie node'a

```
Microsoft Windows [Wersja 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. Wszelkie prawa zastrzeżone.

e:\elasticsearch-0.19.11\bin>elasticsearch.bat
[2012-11-05 16:49:18,114][WARN ][bootstrap                ] jvm uses the client vm, make sure to run `java` with the ser
ver vm for best performance by adding `-server` to the command line
[2012-11-05 16:49:18,124][INFO ][node                     ] [Living Colossus] {0.19.11}[5968]: initializing ...
[2012-11-05 16:49:18,135][INFO ][plugins                  ] [Living Colossus] loaded [], sites []
[2012-11-05 16:50:17,304][INFO ][node                     ] [Living Colossus] {0.19.11}[5968]: initialized
[2012-11-05 16:50:17,305][INFO ][node                     ] [Living Colossus] {0.19.11}[5968]: starting ...
[2012-11-05 16:50:17,766][INFO ][transport                ] [Living Colossus] bound_address {inet[/0:0:0:0:0:0:0:0:9300]
}, publish_address {inet[/192.168.192.42:9300]}
[2012-11-05 16:50:21,102][INFO ][cluster.service          ] [Living Colossus] new_master [Living Colossus][ypeX8KFXTke5A
l5Lv0xg6Q][inet[/192.168.192.42:9300]], reason: zen-disco-join (elected_as_master)
[2012-11-05 16:50:21,164][INFO ][discovery                ] [Living Colossus] skryty/ypeX8KFXTke5Al5Lv0xg6Q
[2012-11-05 16:50:21,419][INFO ][http                     ] [Living Colossus] bound_address {inet[/0:0:0:0:0:0:0:0:9200]
}, publish_address {inet[/192.168.192.42:9200]}
[2012-11-05 16:50:21,420][INFO ][node                     ] [Living Colossus] {0.19.11}[5968]: started
[2012-11-05 16:50:22,118][INFO ][gateway                  ] [Living Colossus] recovered [1] indices into cluster_state
```

Wstawienie elementu do indeksu

```
<-- operacja wstawienia elementu
curl -XPUT 'http://localhost:9200/twitter/tweet/4' -d '{
    "tweet" : {
        "user" : "kimchy2",
        "post_date" : "2009-11-15T14:12:12",
        "message" : "trying out Elastic Search new :)"
    }
}'

<-- wynik operacji
{"ok":true,"_index":"twitter","_type":"tweet","_id":"4","_version":1}

<-- przed dodaniem (update) możemy sprawdzić wersje
curl -XPUT 'http://localhost:9200/twitter/tweet/1?version=12' (...)

<-- wymuszenie operacji tworzenia
curl -XPUT 'http://localhost:9200/twitter/tweet/1?op_type=create' (...)
curl -XPUT 'http://localhost:9200/twitter/tweet/1/_create' (...)
```

Operacja dodania POST

```
<-- przed updatem możemy sprawdzić wersje
curl -XPUT 'http://localhost:9200/twitter/tweet/1?version=12' (...)

<-- wymuszenie operacji tworzenia
curl -XPUT 'http://localhost:9200/twitter/tweet/1?op_type=create' (...)
curl -XPUT 'http://localhost:9200/twitter/tweet/1/_create' (...)

<-- automatyczne uzyskanie indeksu poprzez operacje POST
curl -XPOST 'http://localhost:9200/twitter/tweet/' (...)

{
    "ok" : true,
    "_index" : "twitter",
    "_type" : "tweet",
    "_id" : "6a8ca01c-7896-48e9-81cc-9f7066lfcb32",
    "_version" : 1
}

<--wymusznie odpowiedniego indeksowania
curl -XPOST 'http://localhost:9200/twitter/tweet?routing=skryty' (...)
```

Pobieranie elementu z indeksu

```
<-- operacja pobierania

curl -XGET 'http://localhost:9200/twitter/tweet/1'

<-- z kontrolą wersji
curl -XGET 'http://localhost:9200/twitter/tweet/1?version=12'

<-- konkretne pola
curl -XGET 'http://localhost:9200/twitter/tweet/1?fields=user, post_date'

<-- z kontrolą adresu
curl -XGET 'http://localhost:9200/twitter/tweet/1?routing=skryty'


<-- wynik przykładowego pobierania

{"_index":"twitter","_type":"tweet","_id":"1","_version":5,"exists":true, "_source" : {
    "tweet" : {
        "user" : "kimchy2",
        "post_date" : "2009-11-15T14:12:12",
        "message" : "trying out Elastic Search new :)"
    }
}}
```

Usuwanie elementów z indeksu

```
<-- operacja usuwania

curl -XDELETE 'http://localhost:9200/twitter/tweet/1'

<-- z kontrolą wersji
curl -XDELETE 'http://localhost:9200/twitter/tweet/1?version=12'

<-- z kontrolą adresu
curl -XDELETE 'http://localhost:9200/twitter/tweet/1?routing=skryty'

<-- wynik usuwania
{"ok":true,"found":true,"_index":"twitter","_type":"tweet","_id":"1","_version":6}
```

Ustawienia i mapowanie

```
<-- pobranie ustawień
curl -XGET 'http://localhost:9200/twitter/_settings'

<-- pobranie mapowania
curl -XGET 'http://localhost:9200/twitter/tweet/_mapping'

<-- wprowadzenie mapowania
curl -XPOST localhost:9200/test -d '{
"settings" : {
    "number_of_shards" : 1
},
"mappings" : {
    "type1" : {
        "_source" : { "enabled" : false },
        "properties" : {
            "field1" : { "type" : "string", "index" : "not_analyzed" }
        }
    }
}
}'
```

Wyszukiwanie

```
<-- operacja wyszukiwania
curl 'http://localhost:9200/twitter/tweet/_search?q=user:kimchy2&pretty=true'

{
  "took" : 842,
  "timed_out" : false,
  "_shards" : {
    "total" : 5,
    "successful" : 5,
    "failed" : 0
  },
  "hits" : {
    "total" : 1,
    "max_score" : 0.30685282,
    "hits" : [ {
      "_index" : "twitter",
      "_type" : "tweet",
      "_id" : "4",
      "_score" : 0.30685282, "_source" : {
    "tweet" : {
        "user" : "kimchy2",
        "post_date" : "2009-11-15T14:12:12",
        "message" : "trying out Elastic Search new :)"
    }
}
    } ]
  }
}
```

Wyszukiwanie cd.

```
<-- wyszukanie z negacją
curl 'http://localhost:9200/blog/post/_search?q=-title:search&pretty=true'

<-- suma warunków
curl 'http://localhost:9200/blog/post/_search?q=+title:search%20-title:distributed&pretty=t

<-- z zakresem
curl -XGET 'http://localhost:9200/blog/_search?pretty=true' -d '
{
    "query" : {
        "range" : {
            "postDate" : { "from" : "2011-12-10", "to" : "2011-12-12" }
        }
    }
}'
```

Java API

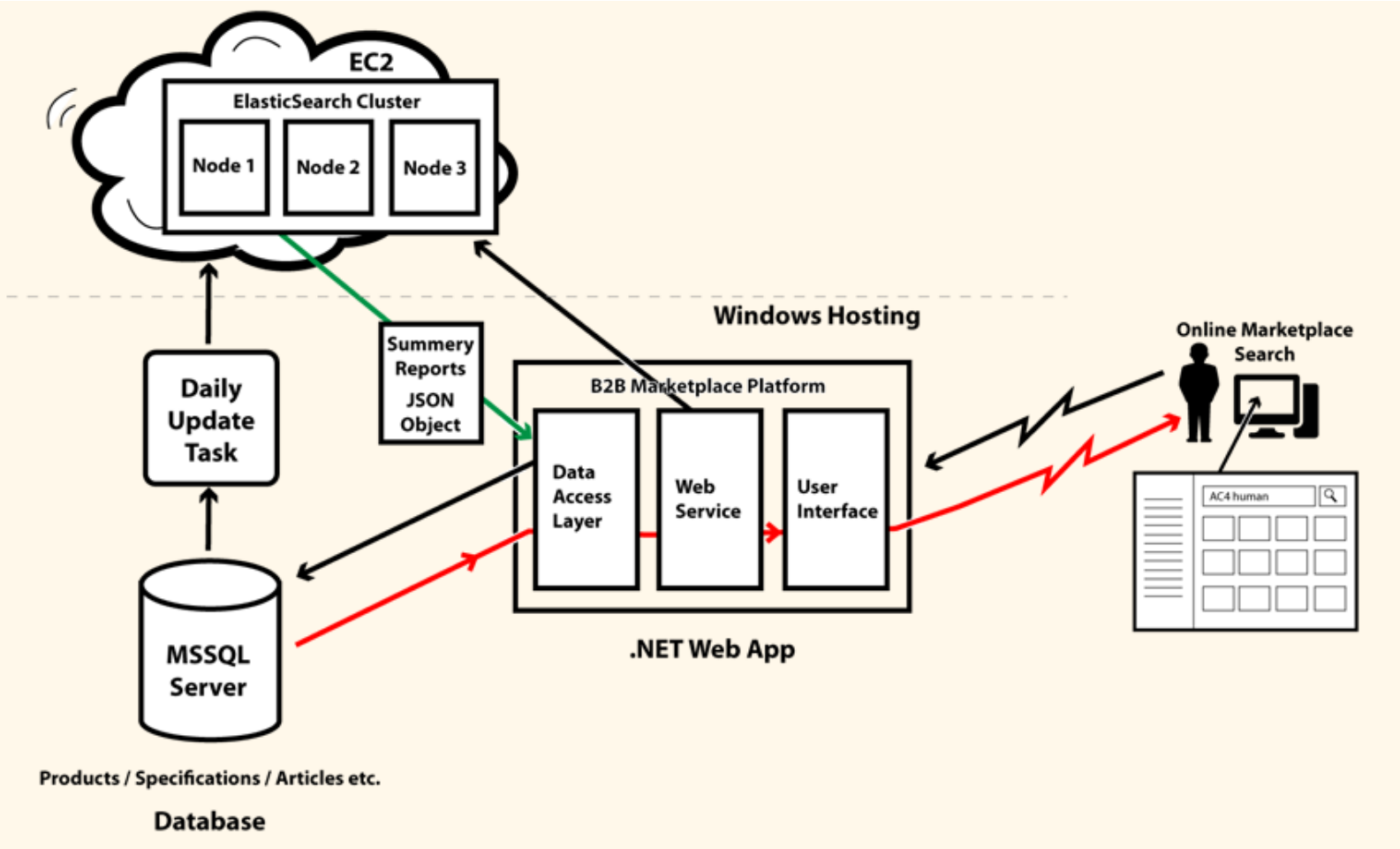
```
import static org.elasticsearch.node.NodeBuilder.*;

Node node = nodeBuilder().node();
Client client = node.client();
BulkRequestBuilder bulkRequest = client.prepareBulk();

// either use client#prepare, or use Requests# to directly build index/delete requests
bulkRequest.add(client.prepareIndex("twitter", "tweet", "1")
        .setSource(jsonBuilder()
                    .startObject()
                        .field("user", "kimchy")
                        .field("postDate", new Date())
                        .field("message", "trying out Elastic Search")
                    .endObject()
                  )
        );

node.close();
```

SQL-ElasticSearch hybrydy



Korporacje wykorzystujące ElasticSearch:

- Mozilla Foundation
- StumbleUpon
- Sony Computer Entertainment
- InSTEDD
- IGN
- Sonian Inc.
- Comarch
- inne ...

Podsumowanie:

- Bazy NoSQL przeżywają rozkwit ze względu na:
  - łatwą skalowalność,
  - wysoką wydajność,
  - elastyczny model danych,
  - nowość.
- Wybór SZBD powinien zależeć od konkretnych zastosowań.
- Użycie nierelacyjnych baz danych jest nadużywane i często nie odpowiada im efektywnym obszarom zastosowań.

ElasticSearch - NoSQL DataBase

Dziękuje za uwagę