MASARYKOVA UNIVERZITA
FAKULTA INFORMATIKY

Fulltextová indexace a relevance ranking archivu národního webu

DIPLOMOVÁ PRÁCE

Petr Šimkovič

Brno, podzim 2008

Prohlášení

Prohlašuji, že tato diplomová práce je mým původním autorským dílem, které jsem vypracoval samostatně. Všechny zdroje, prameny a literaturu, které jsem při vypracování používal nebo z nich čerpal, v práci řádně cituji s uvedením úplného odkazu na příslušný zdroj.

Vedoucí práce: Ing. Petr Žabička

Klíčová slova

Nutch, NutchWAX, WERA, fulltextový index, Hadoop, Page ranknig, SRU/W

Shrnutí

Práce ve zkratce mapuje současný stav archivace řešené projektem WebArchiv.cz. Zabývá se tím, co vlastně takový fulltextový index představuje. Co by měl splňovat a jak vypadaly počátky fulltextové indexace v projektu.

Pak se již plně soustředí na popis použitých nástrojů, jejich konfigurací a popisuje i průběh jejich práce. Zabývá se i možností vytváření indexu pomocí paralelního a distribuovaného zpracování na více počítačích současně. Na konci práce je soustředěna pozornost na page ranking.

Obsah

Kapitola 1

Úvod

Internet je v dnešní době prvním místem, kam se lidé uchylují, pokud potřebují získat nebo vyhledat určité informace. O tom, že je zde možné nalézt skutečně skoro vše, dnes již také nikdo nepochybuje. Většina světových, velkých vyhledávačů jim v tomto s radostí pomáhá.

Za nevýhodou těchto vyhledávacích strojů můžeme považovat skutečnost, že ve své databázi obsahují dokumenty jen z poslední doby. Možná lépe řečeno poslední „verze“ těchto dokumentů. Nikdo proto již nemůže zjistit, jak určité stránky v minulosti vypadaly nebo dokonce, jaké informace na nich uživatel mohl objevit a získat je tak ke své potřebě.

Proto se v posledních letech objevila iniciativa o zachování těchto informací. Té se především chopily státní či národní knihovny, různé univerzity a nesmíme zapomenout i na organizaci Internet Archive. Nejen že tyto informace uchovávají, ale nabízí je právě i jako výsledek dotazů svých vyhledávačů.

Kapitola 2

Současný stav

V současné době je celá archivace webu rozdělena do několika etap a oblastí. Pro bližší pochopení shrnuji a uvádím jen krátký přehled.

1. První fázi archivace představuje výběr webů a domén pro sklizení. Zde je i zahrnuto podepsaní smluv s partnery, jejichž obsah webu bude volně dostupný pro celou veřejnost.

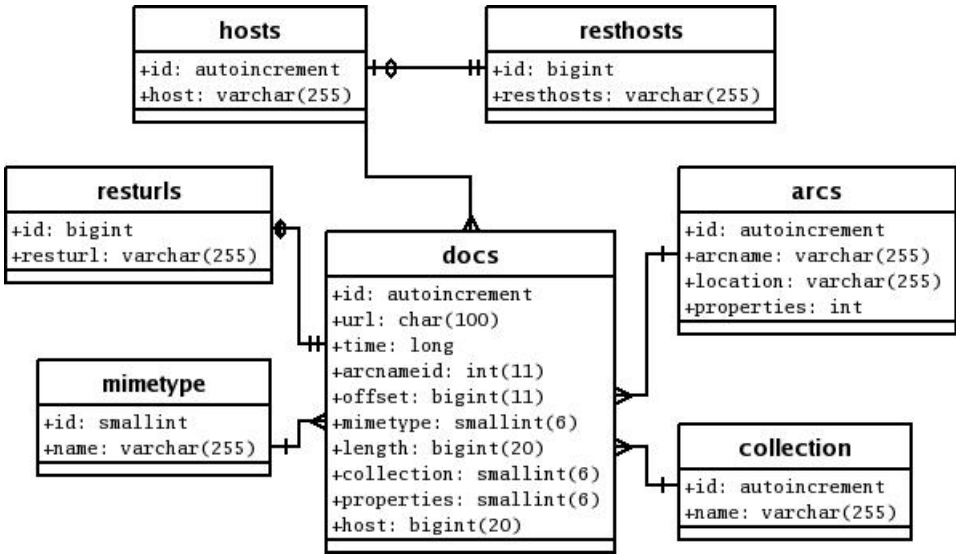
2. Druhou fázi představuje „sklízení“ samotného webu na základě tzv. semínek. Celou činnost zajišťuje jediný nástroj – Heritrix[1]. Výsledkem jeho práce jsou tzv. ARC soubory[2].

3. Nyní máme archivované dokumenty připraveny ve formě již zmíněných ARC souborů. Ty se dále zpracovávají, kdy se metadata jednotlivých dokumentů ukládají (jsou indexována) do MySQL databáze ARCRepos. Její schéma je ukázáno na obrázku 2.1. Zde bych se zmínil o jedné vlastnosti atributu *properties* (typu smallint(6)) tabulky *docs*, který bude mít pro nás v následujících kapitolách význam. Pokud má tento atribut nastaven 5. bit na hodnotu 1 (tj. operace ′| 16′), znamená to, že dokument je označeno jako nevalidní. V dalším textu pro nás bude ještě důležitá tabulka *arcs* uchovávající informace o jednotlivých ARC souborech.

4. Jakmile je hotova indexace dokumentů je možná další práce s těmito informacemi. Jako příklad uvádím např. již zmíněnou validaci nebo označení dokumentů, které jsou veřejně přístupné.

5. Za poslední fázi lze považovat zpřístupnění dokumentů. Tento účel zajišťují aplikace jako je Wayback umožňující vyhledávání dokumentů na základě URL a času. Možná je výměna metadat pomocí protokolu

1. http://crawler.archive.org/
2. http://www.archive.org/web/researcher/ArcFileFormat.php



Obrázek 2.1: Datový model databáze ARCRepos

OAI-PMH[3] a další. Detailnější popis celého workflow archivace a zpřístupnění lze najít na stránkách WebArchivu[4] nebo v diplomové práci Lukáše Matějky[5]. V dalším textu se budeme zabývat procesem full-textové indexace, která ještě v celém workflow chybí.

3. http://www.openarchives.org/
4. http://www.webarchiv.cz
5. http://www.webarchiv.cz/files/dokumenty/ostatni/thesis.pdf

Kapitola 3

Krátce o fulltextu

Pokud mluvíme o fulltextovém vyhledávání, musíme si uvědomit, že jde o vyhledávání dokumentů na základě porovnání zadané fráze se všemi ostatními slovy v daném dokumentu. K tomuto jsou nezbytné algoritmy umožňující vytvořit vyčerpávající statistiku jednotlivých výskytů slov/pojmů z dokumentů a tuto statistiku uložit (zaindexovat). Zde záleží na jednotlivých implementacích, a tak jsou datové struktury těchto indexů různé, chceme-li proprietární.

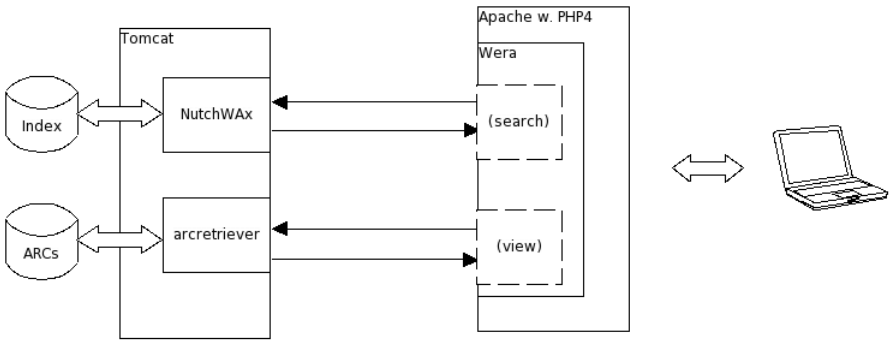
Narozdíl od běžných vyhledávání (např. vyhledávání v názvech, hledání v metadatech, ...) hraje ve fulltextovém vyhledávání velkou roli tzv. relevance výsledků. Znamená to sledování počtu výskytů slov v dokumentu. V sofistikovanějších indexech např. to, zda se dané slovo vyskytuje v nadpisu, titulku atd. Běžné je i skloňování nebo nebrání v potaz diakritiku či velikost písmen. Vedle těchto parametrů (existují samozřejmě i jiné), kdy se berou ohledy jen v rámci jednoho dokumentu, existují i fulltexty zabývající se vztahy i mezi jednotlivými dokumenty. V prostředí webu např. počet odkazů na stránku z jiných stránek.

Kapitola 4

První pokusy, WERA

Za jako první pokusy práce s fulltextem ve WebArchivu lze označit práci s aplikací WERA. Pro úplný přehled se stručně zmíním o její charakteristice. Tato webová aplikace umožňuje fulltextové vyhledávání a zobrazování dokumentů. K navigaci mezi různými verzemi téhož dokumentu (z hlediska času) používá časovou osu. Ta je dnes také zahrnuta i do Waybacku. Narozdíl od něj je ale část této aplikace napsána v jazyku PHP, což nepovažuji (soudě dle mých dosavadních zkušeností) za ideální řešení. K hlavnímu důvodu, proč tuto aplikaci nepoužívat je ale jiný důvod, a to její úplné zastavení vývoje.

Schéma celé aplikace je vidět na obrázku 4.1. Jak již bylo zmíněno WERA



Obrázek 4.1: WERA

je aplikace napsaná v jazyku PHP. Aplikaci je možné , jak z obrázku patrné, umístit do webového serveru Apache. Pro vyhledávání je nutné vytvořit index pomocí nástroje NutchWAX. O něm bude řeč později. Jako přístupová bod k dokumentům používá arcretriever (javovská aplikace). Ten získá me-

tainformace a zobrazí příslušný dokument. Tento těžkopádný mechanismus je právě způsoben implementací v PHP, které neumí pracovat s ARC soubory.

Kapitola 5

NutchWAX

Při popisu aplikace jsem se zmínil o nástroji NutchWAX, „Nutch (W)eb (A)rchive e(X)tensions“. Jde o pouhou nadstavbu modulárního systému Nutch, softwaru vyvinutého pro stahování a zpracování velkého množství stránek/dokumentů. NutchWAX umí fulltextově indexovat dokumenty uložené v již zmíněném ARC formátu, ale i nově stále více používajícím WARC formátu.

5.1 Konfigurace

Než začneme se samotnou fulltextovou indexací, musíme nejprve správně nastavit a zkonfigurovat celý její průběh. K tomu slouží konfigurační soubor `nutch-site.xml`. Tímto souborem překryjeme defaultní nastavení ze souboru `nutch-default.xml`. Jak je vidět konfigurace se provádí pomocí XML souboru. Jednotlivé položky (properties) mají následující strukturu

```
<property>
  <name>jméno položky konfigurace</name>
  <value>hodnota této položky</value>
</property>
```

Mezi nejdůležitější nastavení patří:

- Zahrnutí pluginů do průběhu indexace pomocí property *plugin.includes*. Zde mimojiné nastavujeme, že chceme indexovat všechny následující typy dokumentů: textové, html, pdf, Microsoft office dokumenty (jednotlivě pak Excel, Word, Powerpoint) a Open Office dokumenty. Naopak odebírám zpracování javascriptových souborů.
- Specifikace polí, které se mají zahrnout do indexu se provádí pomocí property `nutchwax.filter.index`. Zde můžeme přidat např. pro-

perties z již zmíněné tabulky docs. O implementačních detailech pak později.

- Je vhodné také vypnout volbu detekci mimetypu pomocí property `mime.type.magic`. Důvody pro to jsou dva: jednak ARC/WARC soubory již v sobě mají tuto informaci zahrnutou (nástrojem Heritrix, který tyto souboru vytváří) a druhak také, protože implementace Nutche načte dokument do pole bytů a pak jej převadí na datový typ String, jehož hodnota je porovnávána s mimetype databází. Toto celé může snadno vést k nedostatku paměti, zejména pro audio či video soubory. V neposlední řadě mimetype máme již také uložen v naší databázi ARCRepos.
- Jako o poslední položce, potřebné však jen pokud chceme dále i v idexu vyhledávat, bych se zmínil o property `searcher.dir`. Zde se nastavuje cesta k našemu indexu.

5.2 Indexace

Protože již ale dokumenty máme (již jsou sklizené nástrojem Heritrix), bude nám stačit, když se budeme dále zabývat jen samotnou indexací. Ta se skládá z následujících kroků.

1. **import** – Původní nástroj Nutch potřebuje pro import dokumentů uvést cestu k ARC souborům. NutchWAX raději volí způsob manifest souboru. Tento soubor obsahuje jednotlivé cesty a ARC soubory, které se mají zaindexovat. V implementaci pro WebArchiv je tato varianta, však nahrazena. Import dokumentů znamená postupné zpracování všech souborů uvedených v manifest souboru. Výsledek je pak uložen v adresáři segments. Vše se provádí následujícím příkazem

   ```
   $NUTCH_HOME/bin/nutchwax import manifest segments
   ```

   Při importu NutchWAX přidává do svého indexu potřebná metadata, které Nutch sám o sobě do indexu nepřidává. Vše je zřejmé z následujícího kódu:

   ```
   ...
   contentMetadata.set( NutchWax.CONTENT_TYPE_KEY,
   meta.getMimetype());
   contentMetadata.set( NutchWax.ARCNAME_KEY,
   ```

```
meta.getArcFile().getName() );
contentMetadata.set( NutchWax.COLLECTION_KEY,
collectionName);
contentMetadata.set( NutchWax.DATE_KEY,
meta.getDate());
...
```

2. **updatedb** – Nyní můžeme přistoupit k další činnosti.
   ```
   $NUTCH_HOME/bin/nutch updatedb crawldb \
   -dir segments
   ```
   Zde se zpracovávájí výsledeky importu (adresář segments) a výsledek se ukládá do adresáře crawldb. Ten nyní obsahuje všechny záznamy z naimportovaných dokumentů nutné pro další zpracování. Adresář segments obshauje jen surová data.

3. **invertlinks** – Před samotnou indexací ještě musíme provést inverzi všech linků. Příkaz
   ```
   $NUTCH_HOME/bin/nutch invertlinks linkdb \
   -dir segments
   ```
   Je to důležité pro tzv. kotvy v textu (anchors) hrající roli např. až v dále zmiňovaném pagerankingu.

4. **index** – Poslední je samotná indexace,
   ```
   $NUTCH_HOME/bin/nutch index indexes crawldb linkdb \
   segments/*
   ```
   která vytvoří index v adresáři indexes. Dobrou vlastností NutchWAXe je, že se zpracovávají jen nově vzniklé segmenty a např. proces updatedb provadí „update“ jen z těchto adresářů.

Při indexování velkého množství dat můžeme narazit na řadu problémů. Jako první bych se zmínil o problému, v jazyce Java řečeno vyjímce, „Too many open files“. Jde o problém, kdy při indexaci je používáno velké množství souborů. Z některých souborů se čte, do některých souborů se zapisuje. V Unixovém světě můžeme narazit na následující radu – zvýšení limitu pro otevřené soubory pomocí přidání následujícího řádku do souboru `/etc/security/limits.conf`.

```
@nutch     hard    nofile  32768
```

Toto nám zajistí zvýšení příslušného limitu pro všechny uživatele ve skupině „nutch“. Pokud pracujeme jako uživatel s root právy, což se nedoporučuje, můžeme použít následujícího triku,

```
(ulimit -n 32768; $NUTCH_HOME/bin/...)
```

který nám zajistí zvýšení limitu pro daný proces.

Další problém může nastat se systémovými prostředky a časem potřebným pro celý průběh indexace. K tomu nám může pomoci Hadoop, kterým se zabývá další kapitola.

Kapitola 6

Hadoop

Jedná se o softwarovou platformu sloužící aplikacím, které zpracovávají velké množství dat. V jistém slova smyslu a do jisté míry jej lze považovat za distribuovaný filesystem. Mezi jeho přednosti patří např. schopnost zpracovávat až petabyty dat. Umí distribuovat data a řídit cluster, kdy jsou data a práce rozdělována mezi dostupné stanice. Tímto je pak snadné paralelně zpracovávat jakákoliv data. V neposlední řadě se umí vypořádat s různými problémy jako výpadek stanice nebo selhání zpracování některé ze stanic.

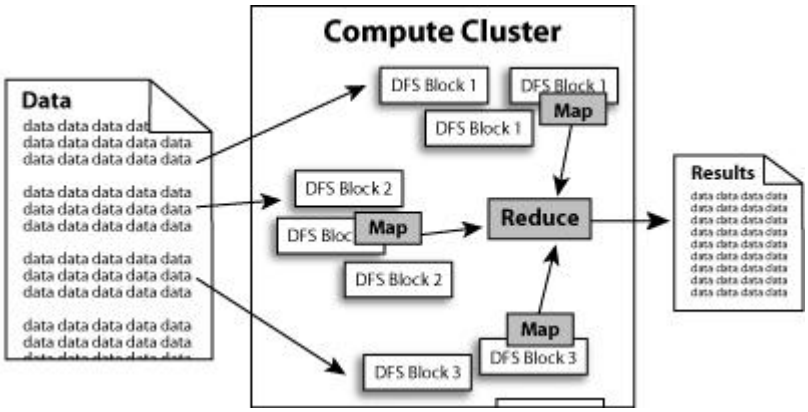
6.1 Struktura

Celý cluster se skládá ze dvou druhů stanic:

- Master
  - namenode – řídí a kontroluje samotný filesystem, dále řídí a reguluje přístup k souborům pro jednotlivé operace klientů (open, close, rename)
  - jobtracker – plánuje a předává práci stanicím vykonávající zpracování dat (tasktrackers)
- Slave
  - datanode – stojí mezi namenode a jdením tasktrackerem, jehož požadavky obsluhuje
  - tasktracker – vykonává zpracování dat

6.2 Zpracování dat

Veškeré zpracování je rozděleno do dílčích subzpracování, tzv. map. Tyto úlohy vykonávají již zmíněné tasktrakery. Tímto je tak zajištěné paralelní zpracování a rozdělení pracé mezi více stanic. Výsledky jednotlivých „map" úloh se předávají tzv. „reduce" úloze, která dílčí výsledky předává jako celek namenode.



Obrázek 6.1: Výpočetní cluster

6.3 NutchWAX a Hadoop

Jak už bylo několikrát zmíněno celý proces indexace může trvat velice dlouho. Rovněž lze očekávat vysoké nároky na hardwarové prostředky. Proto vývojáři nástroje Nutch i NutchWAX využívaji ve svých implementacích Hadoop. Celá indexace je tak rozdělena do několika dílčích úloh.

Zde bych se ještě rád zmínil o módech, ve kterých Hadoop může pracovat.

1. **Standalone Operation** – Defaultní mód, ve spojení s NutchWAXem však totožný jako by byl použit jen tento nástroj a tedy indexace by probíhala jen na jedné pracovní stanici. Proto se tímto módem nebudeme dále zabývat. Je vhodná spíše jen vývoj a debugging.

2. **Pseudo-Distributed Operation** – Pokud bychom zvolili tento způsob indexace, indexace by probíhala pomocí výpočetního clusteru popsaného výše. Všechny role, zpracování všech úloh by však zastávala jediná stanice, na níž bychom indexaci spustili. Pro úplnost uvádím ukázku konfigurace pro tento mód. Samotný popis konfigurace bude popsán však níže.

```
<configuration>
  <property>
    <name>fs.default.name</name>
    <value>localhost:9000</value>
  </property>
  <property>
    <name>mapred.job.tracker</name>
    <value>localhost:9001</value>
  </property>
  <property>
    <name>dfs.replication</name>
    <value>1</value>
  </property>
</configuration>
```

3. **Fully-Distributed Operation** – Tento mód zajišťuje, že veškerá práce bude rozdělována mezi jednotlivé stanice, jak je popsáno výše. Spojení mezi stanicemi se provádí pomocí protokolu SSH a nepředpokládá se žádné zadávání hesla či potřeba přihlašovacího certifikátu. Je nutné počítat i s tím, že komunikace mezi stanicemi běží na vyhrazených portech a musí se s ní při sestavování bezpečnostní politiky počítat. Stanice, které chceme aby se do zpracování zapojily, musíme uvést v konfiguračním souboru `slaves`. Každá stanice musí být uvedena na jednom řádku.

   Malá poznámka pro pozdější pochopení se týká importu dokumentů. Jak jsem již zmiňoval pro import potřebujeme tzv. manifest soubor s uvedenými ARC soubory pro indexaci. Hadoop pro takový každý soubor vytváří samostatnou úlohu (map task). Znamená to tedy, že v této fázi je práce rozdělována způsobem – jeden soubor zpracovává jedna stanice (tasktracker) v jedné úloze.

   Pokud bychom chtěli Hadoop využívat v tomto módu musíme nejprve vytvořit jeho filesystem. Ten se vytáří příkazem

```
$NUTCH_HOME\bin\hadoop namenode -format
```

Rovněž se změní i příkazy pro samotnou indexaci, a to následně,

```
$NUTCH_HOME/bin/hadoop jar \
nutch-1.0-dev.jar třída_pro_operaci
```

kde třída pro příslušnou zastupuje
„org.archive.nutchwax.Importer“ pro import,
„org.apache.nutch.crawl.CrawlDb“ pro updatedb,
„org.apache.nutch.crawl.LinkDb“ pro invertlinks a konečně
„org.apache.nutch.indexer.Indexer“ pro indexaci.

6.4 Konfigurace

Veškeré nastavení Hadoopu se provádí pomocí souboru `hadoop-site.xml`. Podobně jako u Nutche je defaultní nastavení předdefinováno v souboru `hadoop-default.xml`. Celá konfigurace je velice důležitá a obsahuje velké množství variant nastavení. Význam jednotlivých položek je možné najít na webových stránkách Hadoopu sekci dokumentace. Zde uvádím jen přehled nejdůležitějších položek k nastavení.

- hadoop.tmp.dir – adresář pro dočasné soubory (nutno počítat s tím, že při indexaci je produkovány soubory s velmi vysokou velikostí)
- fs.default.name – představuje nodename (např. hdfs://localhost:9000/)
- mapred.job.tracker – adresa jobtrackeru (např. localhost:9001)
- dfs.namenode.logging.level – logovací úroveň (např. INFO)
- dfs.name.dir – adresář s umístěním filesystému pro namenode
- mapred.child.java.opts – „java options“ pro tasktracker
- hadoop.logfile.size – maximální velikost logovacího souboru
- hadoop.logfile.count – maximální počet posledních logovacích souborů k uchování
- dfs.block.size – velikost bloku pro nové soubory
- mapred.job.tracker.handler.count – počet vláken pro tasktracker (doporučuje udávat velikost rovnající se asi 4% počtu tasktracker stanic)

- mapred.output.compress – true/false hodnota pro zapnutí/vypnutí komprese výsledků jednotlivých úloh

Kapitola 7

Customizace

Při customizaci NucthWAXe bylo nejdůležitější zachovat možnost jeho spolupráce se systémem Hadoop. Rovněž tak zachovat možnost, aby se import dokumentů mohl provádět v jeho plně distribuovaném módu. Proto byla navrhnuta a naimplentována knihovna pro import dokumentů, zachovávající všechny předchozí připomínky. Zůstává i zapisování velkého množství informací do logovacího souboru. To může být jen omezeno, snížením logovací úrovně (např. z úrovně INFO na úroveň WARN, nebo pokud chceme jen logovat chybové hlášky a stavy, nastavíme úroveň na ERROR).

Proces importu je pak následující. Z manifest souboru se čtou jména ARC souborů uvedené již cesty, kde se nachází. Je možné také zadávat jen id souborů získané z databáze. V implementaci zůstává, že jeden celý soubor představuje jedno dříve zpracování tasktrackerem. Je tak silně omezena možnost, že by vyskytla, již zmíněná, vyjímka toomanyopenfiles. Pro každý řádek je v databázi nalezen příslušný ARC soubor. Pokud není, do logu se zapíše hláška o nenalezení souboru a Hadoop pokračuje (distribujuje další soubory) v práci. Pokud soubor je nalezen jsou vyhledány všechny jeho dokumenty (s ohledem na validnost, viz atribut properties).

Nyní máme všechny dokumenty pro daný ARC soubor načteny z databáze (respektivě jejich metadata) a můžeme pokračovat v práci. Je nutné poznamenat, že skoro všechny další činnosti se provádí v závislosti na tom, jak máme indexaci nakonfigurovanou. Nejprve se kontroluje status kód (viz HTPP protokol) získaný při stahování dokumentu Heritrixem a pokračuje se, jestli je v povoleném rozsahu. Podobný průběh má pak test validnosti URL. Pokud je vše v pořádku, můžeme samotná data dokumentu načíst z ARC souboru a postupně zapisovat všechny potřebné informace do indexu. Rovněž se do něj ukládá i jméno kolekce, do níž dokument patří.

7.1 Poznámky k vlastní indexaci

Zde bych rád ještě uvedl pár poznámek a možných problémů při celém procesu. Jako první bych připomněl, že při naší indexaci se intenzivně pracuje s databází. Proto bych raději volil cestu, kdy bychom celou databázi „naklonovali“ na jiný stroj a všechny dotazy prováděli zde. Je to možné z důvodu, že indexace nemění v databázi žádná data. Rovněž pro urychlení dotazů bych nejprve vytvořil na některých atributech, kde zatím nejsou, indexy, které by dotazy více urychlily.

Pro získání všech id ARC souborů pro danou kolekci a daný rok nějaké sklizně či kolekce můžeme využít tento SQL dotaz:
`select distinct(arcnameid) from docs where collection=XX and time between 20YY0000000000 AND 20YY1300000000.` XX reprezentuje id potřebné kolekce a YY daný rok.

Abychom se vyhnuli vyjímce tomanyopenfiles není dobré mít uvedeno v manifest souboru příliš mnoho ARC souborů. Vyjímka se objevila už při počtu něco málo nad 200. Proto jsem pro každý rok vytvořil samostatný manifest soubor a ten Unixovou utilitkou „split“ rozdělil do více dalších souborů, které jsou postupně za sebou zpracovávány.

Každý ARC soubor má zhruba velikost 100MB. Četnými pokusy je vyzkoušeno, že pokud takový soubor importujeme do HDFS (Hadoop filesystem) budeme v něm potřebovat přibližně násobek 1.5 jeho velikosti (cca 150MB). Dále je dobré řídit se radou Internet Archive (IA), která říká, že pokud bychom chtěli plně využívat služeb Hadoopu, je dobré mít nejméně 5 pracovních stanic. Jednu jako master a zbylé využít jako slaves. Je nutné si i uvědomit že defaultní replikační faktor má hodnotu 3. To znamená, že všechna data uložená v HDFS jsou uložena na třech stanicích. Pokud bychom tedy měli méně jak 5 stanic, replikace dat by zaměstnávala téměř všechny ostatní uzly, které by jinak mohli zpracovávat ARC soubory či jiným způsobem se mohli podílet indexaci. Dále IA doporučuje používat stanice se stejnou hardwarovou konfigurací. Jako minimální konfiguraci doporučuje: dual-core x86-64 (64-bit) procesor a 4GB RAM s 2GB jako swap.

Kapitola 8

Page ranking

Pokud již máme fulltextový index vytvořený a vyhledávání je také funkční, budeme jistě chtít výsledky vyhledání libovolného dotazu nějakým způsobem třídit. To lze zajistit pomocí vhodně zvoleného page rankingu. Ve své podstatě jde o algoritmus, který na základě parametrů (např. výskyt daného slova v textu, počet odkazů na dokument, atd.) přiřazuje každému výsledku číselnou hodnotu. Podle těchto hodnot jsou pak výsledky setříděny a prezentovány uživateli. Můžeme na něj také pohlížet jako na „skóre“ či „míru důležitosti“ dokumentu.

V implementaci NutchWAXe používám page ranking, který lze parametrizovat následujícími hodnotami. Ty se nastavují v konfiguračním souboru „nutch-site.xml“. Samotný ranking je počítán až procesu vyhledávání, což umožňuje změnu těchto parametrů, aniž bychom museli vytvářet nový index. V závorkách jsou vždy uvedeny defaultní hodnoty. Pro to aby vše fungovalo je nutné mít v položce `plugin.includes` téhož konfiguračního souboru mít zahrnut plugin index basic, anchor a nutchwax.

1. query.url.boost – (4.0f) zesiluje výskyt hledaného výrazu při jeho nalezení v URL
2. query.anchor.boost – (2.0f) zesílení pomocí tzv. kotev
3. query.title.boost – (1.5f) zesílení na základě výskytu v nadpisu
4. query.host.boost – (2.0f) zesílení na základě výskytu v URL hosta
5. query.phrase.boost – (1.0f) zesílení na základě výskytu v textu

Dále je vhodné nahradit plugin scoring-opic pluginem scoring-nutchwax. Důležité je pak během indexace, přesněji po fázi invertlinks, provést příkaz

```
$NUCTH_HOME/bin/nutchwax pagerank pagerank.txt linkdb
```

Soubor `pagerank.txt` pak bude představovat jakýsi seznam počtů inlinků (odkazů z dokumentů na jiné dokumenty). Např.

```
702448 http://www.elbalero.gob.mx/
703517 http://www.mexicoenlinea.gob.mx/
764195 http://www.brasil.gov.br
```

Z ukázky je možné vidět, že na URL http://www.brasil.gov.br existuje přesně 764 195 inlinků a je to tedy nejvíce odkazované URL. Zesílení je pak dáno následující rovnicí

$$boost = log10(pocetinlinku) + 1$$

Pokud by boost mělo hodnotu 1, znamená to, že dokument nebude nijak zvýhodněn před ostatními dokumenty na základě inlinků. Přičtení hodnoty 1 zajistí, že boost bude mít nejmenší hodnotu rovnající se právě této jedničce.

Tento soubor je pak využit při indexaci a je nutné jej předat jako parametr

```
$NUTCH_HOME/bin/nutch index \
          -Dnutchwax.scoringfilter.pagerank.ranks=pagerank.txt \
          indexes \
          linkdb \
          crawldb \
          segments/*
```

Kapitola 9

SRU/SRW

V současné době probíhají testy s SRU rozhraním pro vyhledávání nad full-textovým indexem a jeho integrací s *Jednotnou integrační branou*. Mezi další pokus patří zpřístupnění indexu pomocí standardního SRU rozhraní. Tato iniciativa byla zahájena Státní knihovnou v Berlíně, respektive Slavistik-Portalem. Jedná se o experiment ve vyhledávání nad více Lucene indexy (tj. indexy vytvořené pomocí nástroje Nutch, resp. NutchWAX). Do experimentu jsou nyní zapojeny knihovny např. z Polska nebo Ruska.

9.1 SRU

Zkratka SRU znamená *Search/Retrieval via URL*, tj. vyhledání a obdržení dokumentů je umožněno na základě dotazu skrz URL. Veškeré parametry pro vyhledávaní (např. operation, version, query, startRecord, . . .) jsou obsaženy v URL. Jde tedy o běžnou GET metodu protokolu HTTP. Výsledek dotazu je ve formátu XML, který se pomocí XLST transformuje do HTML (to už je ale záležitostí příslušného softwaru/implementace webového klienta).

9.2 SRW

Kromě SRU je pro vyhledávání možné použít dnes stále populárnějších webových služeb, SRW – *Search/Retrieve via Web Service*. Tedy vyhledávání a obdržení dokumentů je zajištěno, jak bylo již zmíněno, prostřednictvím webových služeb.

9.3 CQL

Pro veškeré vyhledávání se používá „Contextual Query Language“. Do verze specifikace 1.1 se jednalo o „Common Query Language“. Je to formální jazyk pro reprezentování dotazů nad informačními systémy jako jsou webové indexy, bibliografické katalogy nebo různé datové kolekce. Velký důraz je kladen na to, aby schéma a „vzhled“ dotazu byl co nejvíce intuitivní a čitelný, a tedy snadný i na zapamatování. Do dotazů se přidávají i takové klauzule, které zajišťují např. třídění dle daného pole v indexu.

Příklad SRU dotazu je uveden níže (v současné době funkční[1])

```
.../SRWLucene/search/LuceneDemoDB?
query=local.collection='serials'
+and+local.content='brod'&version=1.1
&operation=searchRetrieve
```

Pokud bychom dotaz chtěli vyzkoušet, dostaneme metadata dokumentů v kolekci serials, které ve svém textu obsahují slovo brod. Dále jsme použili specifikaci verze 1.1 a operaci „searchRetrieve“. Podobný dotaz pomocí SRW by mohl vypadat následovně

```
<SOAP:Envelope
xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP:Body>
    <SRW:searchRetrieveRequest
xmlns:SRW="http://www.loc.gov/zing/srw/">
      <SRW:version>1.1</SRW:version>
      <SRW:query>
(local.collection = "serials" and
local.content = "brod")</SRW:query>
      <SRW:startRecord>1</SRW:startRecord>
      <SRW:maximumRecords>10</SRW:maximumRecords>
      <SRW:recordSchema>
http://www.loc.gov/mods/
</SRW:recordsSchema>
    </SRW:searchRetreiveRequest>
  </SOAP:Body>
```

1. http://raptor.webarchiv.cz:8080/SRWLucene/search/LuceneDemoDB?query=local.collection='serials'+and+local.content='brod'&version=1.1&operation=searchRetrieve

```
</SOAP:Envelope>
```

Jako poznámku bych rád zmínil možnosti zúžení výsledků pomocí času

```
<SRW:query>
... and local.date >= ”20040101”
and  local.date < ”20050101” ...
</SRW:query>
```

a možnosti použití boolean operátorů (v našem případě operátoru and).

9.4 Další specifikace a implementace

SRU/W si jako svůj vzor vzalo protokol Z39.50 (klient-server protokol pro vyhledávání a obdržení informací ze vzdálených počítačových systemů/databází). Ve své podstatě SRU/W není tak komplexní, a proto jeho implementace je značně ulehčena (tedy i rychlejší, levnější). Je použito jen běžných webových protokolů a nástrojů jako WSDL, SOAP, HTTP a XML. Podobně jako SRU používá pro dotazy CQL jazyk.

Samotný protokol umožňuje tři typy základních operací (parametr operation):

1. **explain** – získání informací o databázi umístěné na vyhledávacím serveru (její umístění, její popis, přístupové body nebo podporované vlastnosti)
2. **searchRetrive** – umožňuje vyhledávaní pomocí CQL jazyka (formát výsledků je možné získat pomocí předchozí operace explain, např. Doublin Core, MODS, Marc)
3. **scan** – slouží k získávaní termínů z indexu databáze pro procházení rejstříku

K zajištění k přístupu k našemu fulltextovému indexu byl použit nástroj „SRW FRED LOM Web Service“ (viz URL [2]) implementující SRW Search/Retrive standard. Jako možný další se jeví *OCLC Research SRW Server 2.0*, tento nástroj však není nějakou dobu dostupný a tedy není možné jej vyzkoušet (viz URL [3]). Naše aplikace běží pod javovským kontejnerem

2. http://fred.usq.edu.au/searchtoolkit.html
3. http://www.oclc.org/research/software/srw/default.htm

Apache Tomcat 5.5 a pro svou konfiguraci používá dvou konfiguračních souborů:

1. `SRWDatabase.props` – mezi nejdůležitější nastavení patří přiřazení xsl souborů k jednotlivým druhům operací (xsl soubory slouží pro XSLT transformaci výsledků operací do HTML výstupu), dalším důležitým nastavením tohoto soboru je namapování jednotlivých polí indexu na standardní pole. Např. používáme dva různé indexy, v jednom indexu je URL zaznamenáno v poli local.url a ve druhém indexu v poli lucene.url. Namapování potom znamená, že pro obě pole použijeme zástupné jméno, např. url. Umožní to, že jeden dotaz je možný klást nad více indexy a nemusíme měnit pro každý index název daného pole.

2. `SRWServer.props` – zde se nastavují jména „databází“ pro vyhledávání nad různými indexy a cesty k těmto indexům.

Jak je z ukázek vidět, naše „databáze“ (pojem databáze je zde značně zavádějící, koncovému uživateli se to spíše může jevit jako přístupový bod) se jmenuje *LuceneDemoDB*. Ke každé takové databázi se přiřazuje cesta k indexu a je tak tedy možné prohledávat více indexů pod různými URL. Za jako jeden možný případ použití lze považovat nemožnost inkrementálního indexování. Z webového rozhraní je patrné, na co vše je možné se dotazovat. Nutno podotknout, že tato implementace však vyhledávání ve všech polích neumožňuje. Používání zástupných znaků v dotazech rovněž neumožňuje. Jako pozitivum lze považovat její napojení na Slavistik-portal[4].

4. http://www.slavistik-portal.de/en/indexsearching.html

Literatura

[1] http://www.webarchiv.cz/

[2] http://www.archive.org/web/researcher/ArcFileFormat.php

[3] http://lucene.apache.org/nutch/

[4] http://archive-access.sourceforge.net/projects/nutch/

[5] http://hadoop.apache.org/core/

[6] http://www.loc.gov/standards/sru/

Příloha

Ukázka souboru `SRWDatabase.properties`

```
databaseInfo.title=SRU/W Test - WebArchiv
databaseInfo.description=An index of the metadata
databaseInfo.author=WebArchiv.cz
databaseInfo.contact=contact
qualifier.cql.serverChoice=fredlom.fullrecord
explainStyleSheet=/SRWLucene/explainResponse.xsl
scanStyleSheet=/SRWLucene/scanResponse.xsl
searchStyleSheet=/SRWLucene/searchRetrieveResponse.xsl
#searchStyleSheet=/SRWLucene/dublinCoreRecord.xsl
qualifier.lucene.content=content
qualifier.cql.serverChoice=content
```

V tomto souboru se definuje vzhled všech stránek (např. vzhled stránky se samotnými výsledky dotazů) pomocí souborů pro XSLT transformaci. Je zde možné definovat i jednotlivé jména kvalifikátorů (polí pro vyhledávání).

Ukázka souboru `SRWServer.properties`

```
default.database=DSpace
resultSetIdleTime=300
makeIndex.html=true
db.test.class=ORG.oclc.os.SRW.SRWTestDatabase
db.LuceneDemoDB.class=ORG.oclc.os.SRW.Lucene.
SRWLuceneDatabase
db.LuceneDemoDB.home=/mnt/pole3/nutch-index/segments/
tar2006-11-13-215909
#db.LuceneDemoDB.home=/home/xsimkov/crawl/indexes
db.LuceneDemoDB.configuration=SRWDatabase.props
```

Zde stojí za povšimnutí slova „LuceneDemoDB“. Jako první je definována třída zodpovědná za obsluhu všech požadavků, dále se definuje cesta k samotnému indexu. Jako poslední je nastavena cesta k prvnímu konfigu-

račnímu souboru. Tímto způsobem je umožněno v jedné SRU/W aplikaci vyhledávat pod různými URL (každé pro jiný index). V našem případě jde o URL .../SRWLucene/search/LuceneDemoDB?.... Není tak tedy problém nakonfigurovat další přístupový bod např. LuceneDemoDB2.