IV Nacionalna studentska konferencija
Kragujevac, 23 maj, 2013
Tematska radionica "Unapređenje studentskih praksi na univerzitetima u Srbiji"

TEPMUS JP 510985-2010

PARALELIZACIJA LUCENE INDEKSIRANJA

PARALLELIZATION OF LUCENE INDEXING

student: Milica Surla, Prirodno-matematički fakultet, Novi Sad, surmil19@yahoo.com

preduzeće/supervizor: Schneider Electric DMS, d.o.o., Novi Sad

mentor prakse: Kosa Nenadić, kosa.nenadic@schneider-electric-dms.com

ZADATAK: Realizacija zadatka u web okruženju (Tomcat,Eclipse) korišćenjem Java programskog jezika i Apache Lucene biblioteke. U sklopu zadatka potrebno je upoznati se sa višenitnim programiranjem u Java programskom jeziku i mehanizmom paralelnog upisa u Lucene indeks. Potrebno je postojeću implementaciju Lucene indeksiranja prilagoditi kako bi se iskoristila mogućnost paralelizacije koju nudi Java programski jezik i Lucene biblioteka. Takođe, potrebno je prilagoditi postojeću implementaciju Lucene pretraživanja kako bi korisnik prilikom pretrage video ažurirane podatke u indeksu.

1. UVOD

Lucene je moćna Java biblioteka za pretraživanje koja omogućava lako dodavanje pretrage bilo kojoj aplikaciji. U poslednjih nekoliko godina Lucene je postao izuzetno popularan i sada je najrasprostranjenija biblioteka za pronalaženje informacija. Jedan od ključnih faktora za popularnost Lucena je njegova jednostavnost, ali ne dozvolite da vas to zavara: pod haubom ima sofiticirane, najsavremenije tehnike pronalaženja informacija. Shodno tome, nije potrebno temeljno znanje o tome kako se informacije indeksiraju i pronalaze u cilju da počne da se koristiti. Lucene se koristi u iznenadjujuće različitim i rastućem broju mesta: NetFix, Digg, MySpace, FedEx, Apple, EclipseIDE ..

Da bi se brzo pretražile velike količine teksta, treba prvo da se indeksira tekst i pretvori u format koji omogućava da se brzo izvrši pretraga, eleminišući spor sekvencijalni proces skeniranja. Ovaj proces konverzije se zove indeksiranje, a njegov izlaz se naziva indeks. Indeks je specijalno dizajnirana struktura podataka, obično smeštena na fajl sistem kao skup indeksiranih datoteka. Indeksiranje se zapravo sastoji od niza različitih logičkih koraka: dobijanje sadržaja, izgradnja dokumenata, analiziranje dokumenata i indeksiranje dokumenata. Lucene, kao osnovna biblioteka za pretraživanje, ne pruža nikakve funkcionalnosti kao podršku za dobijanje sadržaja.

Pre nego što se tekst indeksira mora prvo da se izvrši analiza teksta, proces koji razdvaja tekstualne podatke u tokene, i obavi niz opcionih operacija na njima, što uključuje uklanjanje akcenata, uklanjanje interpukcije, uklanjanje često korišćenih reči (na primer a, an, the u engleskom jeziku), svođenje reči na osnovni oblik, normalizacija (pretvaranje ispisa teksta u ispis malim slovima, engl. lowercasing), itd. Lucene omogućava širok spektar ugrađenih analizatora koji vam omogućavaju finu kontrolu nad ovim procesom. Nakon toga Lucene skladišti sadržaj u indeks. Lokaciju Lucene indeksa predstavlja apstraktna klasa Directory. Lucene naravno obezbeđuje sve što je potrebno za ovaj korak.

Proces analize zahteva dokument koji sadrži polja koja treba da se indeksiraju. Dokument je Lucene-ova jedinica indeksiranja i pretraživanja. To je zapravo samo kontejner koji sadrži jedno ili više polja, koja zauzvrat sadrže "pravi" sadržaj. Svako polje ima svoje ime za identifikaciju, tekstualne ili binarne vrednosti, kao i niz opcija koje opisuju šta Lucene treba da uradi sa vrednostima polja kada se dokument doda u indeks. Lucene obezbeđuje API za kreiranje dokumenta i polja.

IndexWriter je centralna komponenta procesa indeksiranja. Ova klasa kreira novi indeks ili otvora postojeći, a zatim dodaje, uklanja ili ažurira dokumente u indeks. IndexWriter mora negde da smesti indeks i za to služi apstraktna klasa Directory.

Pretraživanje informacija se odnosi na proces traženja dokumenata, informacija u dokumentima ili metapodatke o dokumentima. Lucene može da indeksira i pretražuje sve podatke koji mogu da se izdvoje iz teksta. Lucene ne brine o izvoru podataka, njegovom formatu ili čak njegovom jeziku, dokle god može da se tekst izvuče iz njega. To znači da može da indeksira i pretražuje veb stranice na udaljenim serverima, dokumente sačuvane u lokalnim sistemima, jednostavne tekstualne datoteke, Microsoft Word dokumenta, XML ili HTML ili PDF fajlove, ili bilo koji drugi format iz kojih mogu da se izvuku tekstualni podaci.

IndexSearcher je centralna klasa koja se koristi za pretragu dokumenata u indeksu. Ona ima nekoliko metoda za pretragu, od kojih su neke impelmentirane u apstraktnoj nadređenoj klasi Searcher. Najjednostavniji metod uzima Query objekat i neki integer parameter N i vraća TopDocs objekte. TopDocs klasa je jednostvan kontejner koji pokazuje na prvih N rangiranih rezultata pretrage (dokumenti koji odgovaraju datom upitu).

Query objekti mogu biti veoma jednostavni ili veoma složeni. Lucene dolazi sa velikim brojem konkretnih Query podklasa (TermQuery, BooleanQuery, NumericRangeQuery, itd.). Lucene pruža i moćnu klasu QueryParser, za obradu korisničkog teksta u query objekat koristeći zajedničku sintaksu pretrage. Upit može da sadrži logičke operacije, phrase upite, džoker uslove itd. Ako aplikacija ima više kontrole nad pretragom ili dodatnih zanimljivih ograničenja, mora se implementirati logika koja prevodi ovo u ekvivalentni upit.

Nakon toga se vrši pretraga i preuzimaju se dokumenata koja odgovaraju upitu, sortirana u traženom redosledu. Term je osnovna jedinica za pretraživanje. Slično Field objektu, sastoji se od par string elemenata: ime polja i tekstualna vrednost tog polja. Term objekti su takođe uključeni u proces indeksiranja.

Kada se nova dokumenta dodaju u indeks ili brišu iz indeksa oni se prvo smeštaju u memoriju umesto da se odmah upišu u indeks i to se radi zbog performansi. Ove se promene periodično upisuju u indeks direktorijum kao novi segment.

2. OPIS REALIZOVANIH AKTIVNOSTI

Višenitno programiranje je simultano izvršavanje više niti koji mogu deliti neke zajedničke podatke. Java platforma je iz temelja dizajnirana da podrži višenitno programiranje sa osnovnom podrškom konkurentnosti u Java programskom jeziku i bibliotekama Java klasa. Kada se pravilno koriste, niti mogu da smanje troškove razvoja i održavanja i da poboljšaju performanse složenih aplikacija. Višenitno programiranje je izuzetno pogodno u praksi. Na primer, internet čitač bi trebalo da bude u mogućnosti da simultano dovuče veći broj slika. Nadalje, veb server bi trebalo da bude u mogućnosti da opsluži konkurentne zahteve. I sam programski jezik Java koristi niti da u pozadini vrši sakupljanje odbačenog memorisjkog prostora i tako programera poštedi muke da vodi računa o upravljanju memorijom.

Lucene je pažljivo dizajniran da dobro funkcioniše sa više niti. Lucene je thread safe: podržano je deljenje IndexSearcher-a, IndexReader-a, IndexWriter-a, itd. preko više niti. Lucene je takođe thread friendly: sinhronizovani kod je sveden na minimum tako da niti mogu da u potpunosti iskoristite konkurentnost hardvera.

Za paralelizaciju Lucene indeksiranja korišćena je klasa ThradPoolExecutor. ThreadPoolExecutor obično obezbeđuje poboljšane performanse prilikom izvršavanja velikog broja asinhronih zadataka i obezbeđuje sredstva za ograničavanje i upravljanje resursima, uključujući niti korišćene pri izvršavanju kolekcije zadataka.

2.1 PARALELIZACIJA INDEKSIRANJA

Za paralelizaciju Lucene indeksiranja korišćena je klasa ThreadPoolExecutor. Prilikom instanciranja klase se navodi koliko niti će se koristiti i veličina queue. Treba izbegavati da veličina thread pool-a bude „prevelika“ ili „premala“. Ako je thread pool prevelik, onda se niti takmiče za oskudne resurse procesora i memorije, što rezultira u većoj upotrebi memorije i mogućoj iscrpljenosti resursa. Ako je thread pool suviše mali, propustnost trpi jer procesori se ne koriste uprkos dostupnosti rada. Da bi se pronašla odgovarajuća veličina treba testirati na različite vrednosti da bi pronašli idealan broj niti za aplikaciju, ali dobro pravilo za broj niti je jedan plus broj CPU jezgara na računaru, a veličina queue da bude četiti puta veća od broja niti. Upotrebljen je LinkedBlockingQueue koji sortira elemente na FIFO (first-in-first-out) način i ima veću propusnost od ArrayBlockingQueue, ali manje predvidljive performanse u većini konkurentnih aplikacija.

2.2 PARALELIZACIJA PRETRAŽIVANJA

IndexSearcher vidi samo one podatke u indeksu u vreme kada je otvoren. Jednom kada se dese promene u indeksu IndexSearcher mora ponovo da se otvori da bi video te promene. Nažalost, to može biti skupa operacija, koja troši CPU i I/O resurse. Ipak, za neke aplikacije smanjenje index-to-search kašnjenja vredi tog troška što znači da će se često ponovno otvarati pretraživač.

Niti čine ponovno otvaranje pretraživača izazovnim, jer ne može da se zatvori stari pretraživač dok god vrši pretraživanje, uključujući i iteracija kroz pogotke koje je search metod vratio. Pored toga, možda će biti potrebno da se zadrži stari pretraživač dovoljno dugo dok se sve sesije pretrage (orginalna pretraga plus sve prateće aktivnosti) ne završe ili ne isteknu. Ako se iznenada između stranica izvrši promena u novi pretrazivač onda može da se desi da korisnik vidi duplirane rezultate ili da propusti neke rezultate. Ova neočekivana ponašanja mogu da smanje poverenje korisnika. To može da se spreči slanjem novih stranica za prethodnu pretragu natrag orginalnom pretraživaču kada je to moguće.

Postojeća implementacija je izmenjena da koristi klasu SearcherManager. SearcherManager je korisna klasa koja obezbeđuje da se instanca IndexSearcher-a bezbedno deli među nitima, dok se periodično ponovo otvara. SearcherManager obezbeđuje da se searcher zatvori samo kada su sve niti završile sa njegovim korišćenjem.

SearcherManager je korisna klasa koja krije nezgodne detalje ponovnog otvaranja pretraživača u prisustvu više niti. On je u stanju da otvori searcher-a (reader-a) ili iz Directory instance, u slučaju kada nema direktan pristup IndexWriter-u koji pravi promene, ili iz IndexWriter-a. Ako ima direktan pristup IndexWriter-u koji pravi izmene u indeksu, onda je najbolje da se koristi konstruktor koji prihvata IndexWriter i na ovaj način dobiće se brži učinak ponovnog otvaranja. U suprotnom, treba da se koristi konstruktor koji uzima Directory instancu koji će direktno otvoriti IndexSearcher. Kad god je potreban pretraživač treba imati na umu da klasa neće sama uraditi ponovno otvaranje. Umesto toga, mora da se pozove metod maybeRefresh dovoljno često u skladu sa potrebama aplikacije. Na primer, dobar trenutak da se pozove ovaj metod je nakon napravljenih izmena sa IndexWriter-om. Ako je prosleđen Directory u SearcherManager, mora prvo da se potvrde promene iz IndexWriter-a pre nego što se pozove maybeRefresh.

3. OSTVARENI REZULTATI

Nakon završetka implementacije, cilj je bio da se testira koliko je vremena potrebno da se izvrši indeksiranje. Vršena su testiranja na kompjuteru sa intel dual core procesorom i na kompjuteru sa intel core i3 procesorom. Na osnovu rezultata dobijenih testiranjem sa velikom šemom, došlo se do zaključka da se indeksiranje korišćenjem ThreadPoolExecutor-a za 60 sekundi brže izvršava nego indeksiranje korišćenjem jedne niti. Takođe, došlo se do zaključka da se indeksiranje na kompjuteru sa intel core i3 procesorom za 12 sekundi sporije izvršava od indeksiranja na kompjuteru sa dual core procesorom. Kod testiranja sa malom šemom primeti se mala razlika u rezultatima.

4. ZAKLJUČAK

Izrada informatičkog projekta je ovim testiranjem i završetkom implementacije uokvirena, međutim, potrebno je nastaviti testiranje na računaru sa više jezgara, kao i sa više niti nego što je korišćeno pri ovom testiranju.

LITERATURA

[1] Lucene in Action, Second Editon - Michael McCandless, Erik Hatcher, Otis Gospodnetić

[2] Programski jezik Java – Mirjana Ivanović, Mihal Bađonski, Zoran Budimac, Dragoslav Pešović

[3] Java Concurrency in Practice - Brian Goetz, Tim Peierls, Joshua Bloch , Joseph Bowbeer, David Holmes, Doug Lea

[4] http://docs.oracle.com/javase/tutorial/

[5] http://docs.oracle.com/javase/6/docs/api/java/util/concurrent/ThreadPoolExecutor.html

[6] http://www.lucenetutorial.com/lucene-in-5-minutes.html

[7] http://lucene.apache.org/core/4_0_0/core/org/apache/lucene/search/SearcherManager.html

[8] http://lucene.apache.org/core/4_0_0/core/org/apache/lucene/search/IndexSearcher.html