TARTU ÜLIKOOL
MATEMAATIKA-INFORMAATIKATEADUSKOND
ARVUTITEADUSE INSTITUUT
INFORMAATIKA ERIALA

Risto Saar

Privaatsust säilitava arvutussüsteemi Sharemind andmebaasikiht

Bakalaureusetöö (6 EAP)

Juhendaja: Dan Bogdanov, MSc

Autor: …… "….." jaanuar 2011
Juhendaja: …… "….." jaanuar 2011

Lubada kaitsmisele
Professor ..... ……..……. "….." jaanuar 2011

TARTU 2011

Sisukord

Sissejuhatus

Infosüsteem on süsteem teabe säilitamiseks ja töötlemiseks. Üheks tähtsaks infosüsteemi omaduseks on teabe kättesaamise kiirus. See omakorda sõltub andmebaasimootorist, mida infosüsteemi juures kasutatakse. Eriti tähtsaks muutub see siis, kui olemasolev andmebaasimootor on süsteemi kõige aeglasem osa ehk kui see on pudelikael.

Sharemind on privaatsust säilitav andmetöötlussüsteem, mis oskab privaatselt arvutada ning millel on privaatne andmebaas [1]. Sharemindi kasutamist reaalsetes rakendustes takistab selle vähene andmebaasisüsteemide toetus ning nõrk andmebaasikihi jõudlus.

Käesolev töö uurib selle andmetöötlussüsteemi andmebaasikihi puudusi ning pakub alternatiive, mis tõstavad süsteemi jõudlust ning mis lisavad tuge levinud andmebaasimootoritele.

Autori ülesandeks on täiendada hetkel kasutuses olev andmebaasisüsteemi, jättes vana alles alternatiivse valikuna. Seejärel tuleb uus andmekiht realiseerida ning Sharemindiga integreerida. Viimaks on vaja võrrelda uue ning vana andmebaasisüsteemi jõudlustulemusi ning mõõta, kui palju on uus süsteem vanast parem.

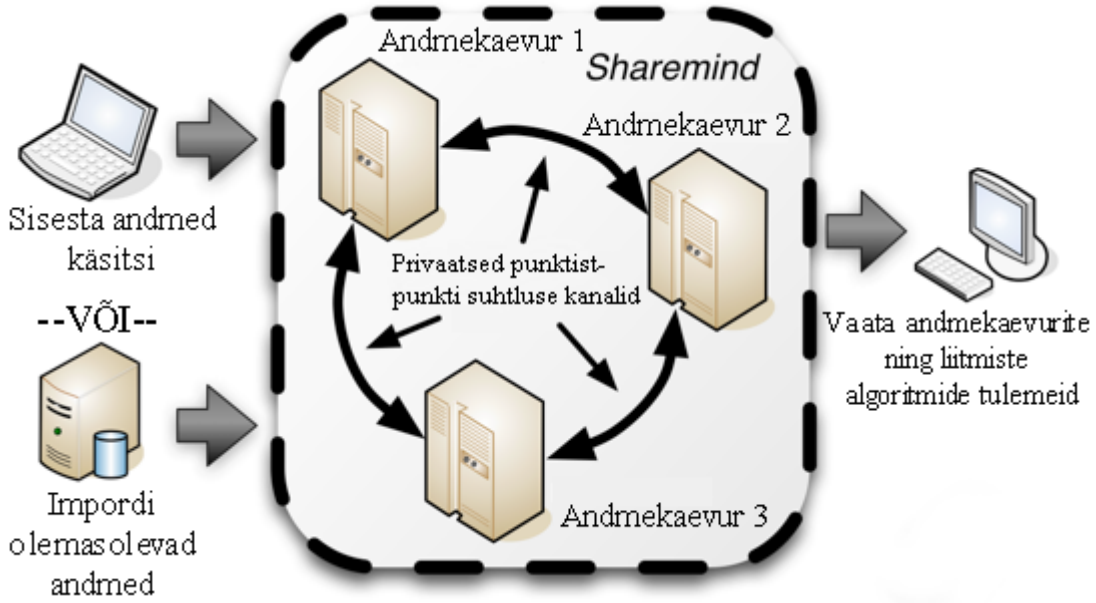
1. Sharemindi kirjeldus

1.1 Sharemindi funktsionaalsus

Sharemind on andmetöötlussüsteem, mis võimaldab sisendandmete töötlemist ilma nende privaatsust ohustamata [2]. Teisisõnu, see töötleb andmeid, ilma et ta neid näeks.

Süsteem põhineb krüptograafilistel alustel. Andmete privaatsuse säilitamiseks kasutatakse ühissalastust (*secret sharing*). Turvalised ühisarvutused (*secure multi-party computation)* lubavad meil ühissalastatud andmeid töödelda.

Töötav süsteem koosneb kolmest andmekaevurist, mis salvestavad ning töötlevad andmeid ning kontrollerrakendustest, mis võimaldavad andmeid sisestada ning andmekaevurite tööd koordineerida [Joonis 1.1].



Joonis 1.1 Sharemindi virtuaalmasin

Andmed sisestatakse käsitsi või imporditakse olemasolevatest andmestikest. Kontrollrakenduste abil kogutakse andmed kaevurite andmebaasidesse. Kui andmed on kogutud, toimub nende töötlemine ning lõpptulemuse avalikustamine. Vahepealsed tulemused jäävad salajaseks.

1.2 Andmebaasikiht

Sharemindi igal andmekaevuril on kohalik andmebaas. Andmebaasikiht põhines töö koostamise alguses andmebaasisüsteemil SQLite. Puudusid võimalused teiste andmebaasikihtide kasutamiseks. Andmebaasikihti kasutatakse nii avalike kui ühissalastatud andmete salvestamiseks. Andmebaasimoodul pakub Sharemindi rakendustele järgnevaid teenuseid:

Tegevused andmemudeliga:

1) Andmebaasi avamine (vajadusel loomine)

2) Tabeli loomine

3) Tabeli kustutamine

4) Tabeli metainfo lugemine

5) Tühja veeru lisamine tabelile

6) Ridade arvu tagastamine

7) Veergude arvu tagastamine

Tegevused andmetega:

1) Andmerea tabelisse lisamine

2) Mitme andmerea korraga tabelisse lisamine

3) Andmeveeru lugemine

4) Andmerea lugemine

5) Andmete tagastamine etteantud rea ja veeru järgi

Eraldi metaandmete tabelid sisaldavad infot selle kohta, millised andmed on ühissalastatud ja millised mitte.

1.3 Puudused

1.3.1 Jõudlus

Hetkel on kasutatava andmebaasimootori jõudlus madal. Jõudluse testimiseks sooritame järgnevad jõudlustestid. Esiteks testime andmebaasi kirjutamise jõudlust. Testimiseks kasutame Sharemindi rakendust nimega *TransactionDataImporter.*

TransactionDataImporter on tööriist, mis impordib transaktsiooniandmeid tekstifailist Sharemindi andmebaasidesse. Failis peab igal real olema transaktsioon, kus andmed on täisarvu kujul ning eraldatud komaga. Neid andmeid saab edaspidi kasutada Sharemindi sees privaatse andmekaevanduse jaoks. Sooritasime testid nelja andmehulga peal: mushroom.dat, chess.dat, connect.dat ning pumsb.dat [3]. Iga andmehulga korral mõõtsime aega, mis kulus selle täielikuks importimiseks Sharemindi andmebaasi.

Tabel	**Kulunud aeg**
mushroom	*56,78s*
chess	*23,46s*
connect	-
pumsb	-

Tabel 1. andmete impordi jõudlus

Connect ja pumsb andmestikke ei õnnestunud importida, sest neis oli liiga palju veerge ning SQLite'il põhinev andmekiht ei võimaldanud neid importida. Andmebaasist lugemise jõudluse testimiseks kasutame reaalset andmekaevandusalgoritmi, mis leiab transaktsioonidest tihtiesinevaid alamhulki. Sharemindi platvormil on realiseeritud mtmeid tihtiesinevate alamhulkade leidmise algoritme. Meie kasutasime Roman Jagomägise kohandatud Apriori algoritmi [4].

Andmeanalüüsi ülesanded põhinevad suurema osa juhtudest tihtiesinevate mustrite leidmises andmebaasis. Sellisteks mustriteks võivad olla assotsiatsioonireeglid, korrelatsioonid, klastrid ning paljud muud. Assotsiatsioonireeglite kaevandamine on üks populaarsemaid probleeme. Algne motivatsioon assotsiatsioonireeglite otsimise jaoks tulenes vajadusest analüüsida nö kaupluse transaktsiooni andmeid. Eesmärk on uurida

kliendi käitumist ostetud kaupade suhtes ning leida, mis kaupu ostetakse tihedamini koos.

Täpsemalt, tihtiesinevate alamhulkade leidmise probleemi saab esitada järgmiselt [4]. Olgu A= (a_1,...,a_m) nimekiri tunnustest, mis esinevad transaktsioonis ehk kõik kaubad, mis on kaupluses saadaval. Sellisel juhul, iga transaktsioon on alamhulk A-st. Privaatses esituses moodustab transaktsioonide nimekiri T_1,...,T_n transaktsioonilise andmebaasi $D^{n \times m}$ nii, et D[i,j] = 1 kui $a_j \in T_i$,vastasel juhul D[i,j] = 0. Oluline on tähele panna, et andmete privaatne esitus on oluliselt mahukam kui avalik. Avalikus esituses ei pea ruumi kulutama puuduvate esemete salvestamisele. Samas, privaatse esituse puhul tuleb ka puuduvate väärtuste kohta ühissalastatud väärtuseid hoida [4].

	a_1	a_2	...	a_m
T_1	1	1	...	0
T_1	0	1	...	1
T_1	1	0	...	0
...	...	...	...	...
T_n	0	1	...	1

Joonis 2. Transaktsiooniline andmebaas [4]

Käivitasime Apriori algoritmi kahe andmestiku peal, mida importida õnnestus. Kasutasime parameetreid, mis vähendasid muule töötlusele kuluvat aega ning rõhutasid andmebaasikihi nõrka jõudlust.

	Andmekiht	**Algoritm kokku**
Tabel	**Kulunud aeg (ms)**	**Kulunud aeg (ms)**
mushroom	3763	25018
chess	537	460537

Tabel 2. Andmebaasist lugemise jõudlus

1.3.2 Andmebaasimootorite vähene tugi

Hetkel kasutab Sharemind SQLite andmebaasimootorit andmete töötlemiseks ning seda ei saa vahetada. See on piiranguks Sharemindi rakendamisele, sest ettevõtetes on kasutusel võimsamad süsteemid nagu MySQL, PostgreSQL, Oracle ja Microsoft SQL Server. Kui ettevõttel on olemas sissetöötatud andmebaasisüsteem koos varunduspoliitikaga, siis on hea, kui Sharemind oskab seda kasutada.

1.3.3 Nõrk tugi andmekaevandusrakendustele

Sharemindi üks eesmärke on privaatsust säilitav andmekaevandus. Paraku on seal tihti tegemist väga suurte andmestikega (näiteks transaktsiooniandmed ja geeniandmed). Seega oleks hea, kui Sharemindil oleks ka võimalus teha väga suuri andmebaase (sadu tuhandeid veerge ja ridu). Tüüpiliste relatsiooniliste andmebaasidega ei ole see nii efektiivne kui NoSQL andmebaasidega nagu Google BigTable või MongoDB [5]. Tulles tagasi meie sooritatud testide juurde, on näha, et *connect* ning *pumsb* andmestike korral ei olnud võimalik andmeid edukalt importida, seda just seetõttu, et SQLite jaoks oli nende andmestike veergude arv liiga suur.

1.4 Ülesandepüstitus

Käesoleva töö eesmärgiks on luua andmebaasikihi paindlik, abstraktne lahendus. Lisaks SQLite'ile lisatakse ka mõne muu andmebaasimootori tugi ning selle käigus parandatakse jõudlust. Kuna Sharemind vajab ka väga suuri andmebaase ning mille haldamisega tavalised relatsioonilised andmebaasid hakkama ei saa, siis lisatakse ka NoSQL andmebaasi tugi.

2.Tehnoloogiate ülevaade

2.1 Andmebaaside kasutamine C++ keeles

Sharemind on kirjutatud C++ keeles, seega huvitavad meid C++ teegid ning tehnoloogiad. Kui programm soovib kasutada mõnda andmebaasimootorit, siis on selleks tavaliselt vaja vastavat teeki. Vaatleme kolme levinumat kategooriat:

1) Andmebaasimootorite oma teegid
2) Integreeritud andmebaasid
3) Andmebaasiühendusi abstraheerivad teegid

Andmebaasimootorite oma teekide ning integreeritud andmebaaside vahe on selles, et esimene ühendab eraldi rakenduses asuva andmebaasi serveri ning teise puhul on andmebaasi kood teegi sees. Andmebaasiühendusi abstraheerivate teekide puhul on mugav vahetada andmebaasisüsteem ilma rakendust oluliselt muutmata. Järgnevalt esitame lühiülevaate iga kategooria mõnest populaarsemast esindajast.

2.2 Andmebaasimootorite oma teegid

2.2.1 libmysql

MySQL Connector/C, tuntud kui ka libmysql, või MySQL Native C API, on eraldiseisev API, mis on loodud C-keele baasil ning on teek, mida saab kasutada C rakendustes, et ühenduda MySQL andmebaasiserveriga. Libmysql on teegi nimi, mille tagab MySQL Connector/C [6]. Sellest on olemas ka C++ variant.

Koodinäide andmebaasi ühendamisest C++ teegi abil [7]:

```
sql::mysql::MySQL_Driver *driver; //draiveri instants
sql::Connection *con; //ühenduse instants

//draiveri instantsi väärtustamine
driver = sql::mysql::MySQL_Driver::Instance();
//ühenduse loomine
con = driver->connect("tcp://127.0.0.1:3306", "user", "password");
//ühenduse kustutamine
delete con;
```

2.2.2 OCCI

Oracle C++ Call Interface (OCCI) on suure jõudlusega ning põhjalik API, pääsemiseks Oracle andmebaaside ligi. OCCI on loodud standardse C++ baasil ning järgib objekt-orienteeritud paradigmat. See on disainitud arendajate produktiivsuse ning koodi kvaliteedi parandamiseks Oracle andmebaasirakendusi arendades.

OCCIt, mida tutvustati esmakordselt Oracle9*i*-s, kasutatakse edukalt klient/server rakendustes ja keerulistes mudelitarkvarades [8].

SQL ja PL/SQL Tugi

OCCIs on tagatud täielik SQL tugi sealhulgas päringutele nagu DDL, DML, SELECT. Samuti on võimalik parameetritega käivitada PL/SQL'is salvestatud alamprogramme. Lisaks on olemas tugi kõikidele Oracle andmetüüpidele nagu näiteks Number, Date, Timestamp, Interval.

Väljavõte andmebaasi ühendamisest ning kirjete leidmisest tabelis *example* [9].

```
...
environment = oracle::occi::Environment::createEnvironment
(oracle::occi::Environment::DEFAULT);
//ühenduse loomine
con = environment->createConnection("gldbuser", "gldbuser",
"MYDATABSE");
//päringu koostamine
stmt = con->createStatement("select * from example");
//päringu käivitamine
res = stmt->executeQuery();
//tulemuste väljastamine ekraanile
while (res->next())
std::cout<<res->getInt(1)<<"  "<<res->getString(2)<<std::endl;
stmt->closeResultSet(res);
con->terminateStatement(stmt);
//ühenduse katkestamine
environment->terminateConnection(con); ...
```

2.3 Integreeritud andmebaasid

Terminit integreeritud andmebaas kasutatakse kahe erineva andmebaasi ülesehituse kirjeldamiseks:

1) ühendus mitme andmebaasi vahel

2) andmebaas, mis on ehitatud mõne rakenduse või tööriista sisse

Kuigi esmase kirjelduse põhjal võib arvata, et need kaks ülesehitust on väga erinevad, siis see nii ei ole. Tegelikult on nad oma ülesehituselt üsna sarnased [10].

2.3.1 SQLite

SQLite on tarkvarateek, mis implementeerib autonoomse, serverita, konfigureerimiseta, transaktsioonilise SQL andmebaasimootori. SQLite on üks kõige laialdasemalt paigaldatav SQL andmebaasimootor maailmas [11]. SQLite lähtekood on avalik [11].

Koodinäide andmebaasi ühendamisest ning tabeli loomisest [12].

```
#include <sqlite3.h>
sqlite3* db;
char* db_err;
int main() {
      //ühenduse loomine
      sqlite3_open("my_db.sql3", &db);
      //päringu käivitamine
      sqlite3_exec(db, "create table 'helloworld' (id integer);",
      NULL, 0,&db_err)
      // ühenduse katkestamine
      sqlite3_close(db);
}
```

2.3.2 Berkeley DB

Berkeley DB võimaldab programmeerijatel kasutada andmebaasi, muretsemata, kuidas andmed on säilitatud ning kuidas väärtusi kätte saada. Andmete kättesaamine on Berkley DB-d kasutades tihti kiirem kui tekstifailidest. Aega säästetakse salvestamisviisiga, mis kiirendab spetsiifilise võtmepaari asukoha leidmist.

Andmete loomine, muutmine ja kustutamine on Berkeley DB-d kasutades üsna lihtne; Kui andmebaas on kord skriptiga seotud, siis saab muutujaid kasutada ning manipuleerida nagu tavaliselt [13].

Koodinäide andmebaasi avamisest [14]:

```
Db db(NULL, 0);                // Db objekti instantsi loomine
u_int32_t oFlags = DB_CREATE; // Ava lipud
try { // Ava andmebaas
    db.open(NULL,                 // Transaktsiooni viit
            "my_db.db",           // Andmebaasi faili nimi
            NULL,              // Valikuline loogilise andmebaasi nimi
            DB_BTREE,             // Andmebaasi ühendusviis
            oFlags,               // Ava lipud
            0);                  // faili olek (kasutades vaikeväärtusi)
} catch(DbException &e) {    // Veakäsitlus siia
} catch(std::exception &e) {// Veakäsitlus siia }
```

2.3.3 Tokyo Cabinet

Tokyo Cabinet on teek andmebaasi haldamise meetoditest. Andmebaas ise on lihtne andmefail, mis koosneb kirjetest. Kirje omakorda koosneb võtmest ning sellele vastavast väärtusest. Võtme ning temale vastavat väärtust saab hoida nii binaarsete andmetena kui ka sõnena. Andmetabeli ja andmetüübi kontsept puudub täielikult. Kirjed paiknevad organiseerituna paisktabelis, B+ puus või kindla pikkusega määratud massiivis.

Tokyo Cabinet on arendatud GDBMi (GNU Database Manager) ning QDBMi (Quick DataBase Manager) järglasena. Põhiomadused, mille poolest Tokyo Cabinet on parem kui GDBM ja QDBM, on järgnevad:

1) andmebaasifaili suurus on väiksem.

2) töötlemiskiirus on suurem.

3) suurem jõudlus mitmelõimelises keskkonnas.

4) lihtsustatud liides.

5) andmebaasifail ei muutu mitteloetavaks isegi katastroofilistes olukordades.

6) toetab 64-bitist arhitektuuri: Saab kasutada väga suurt mäluruumi ning luua suuri andmebaasifaile.

Tokyo Cabinet on kirjutatud C-keeles. Liides toetab järgnevaid keeli: C, Perl, Ruby, Java ning Lua. Tokyo Cabineti saab kasutada platvormidel, mis järgivad C99 ning POSIX standardit. Tokyo Cabinet on vabavara, mis on litsenseeritud GNU Lesser General Public License alusel [15].

2.4 Andmebaasiühendusi abstraheerivad teegid

2.4.1 SQLAPI++

SQLAPI++ on C++ teek pääsemaks ligi mitmetele SQL andmebaasidele (Oracle, Microsoft SQL Server, DB2, Sybase, Informix, InterBase, SQLBase, MySQL, PostgreSQL, ODBC ja SQLite). See kasutab sihtandmebaaside haldussüsteemide APIt nii et rakendused, mida luuakse SQLAPI++ teegi abil töötavad kiirelt ning efektiivselt. Samuti pakub see kerget kasutajaliidest, mis lubab arendajatel ligipääsu andmebaasispetsiifilistele valikutele. Teek on hästi kirjutatud, lihtsasti kasutatav ning stabiilne.

SQLAPI++ kasutamiseks peab ostma iga arendaja jaoks "Site" või "Site +" litsentsi, mille tingimused võib leida SQLAPI++ kodulehelt [16].

Koodinäide andmebaasi ühendamisest [17]:

```
// andmebaasiühenduse loomine
// selles näites kasutame Oracle andmebaasi,
// kuid see võib olla ka Sybase, Informix, DB2
// SQLServer, InterBase, SQLBase ning ODBC
con.Connect(
    "test",     // andmebaasi nimi
    "tester",   // kasutajanimi
    "tester",   // parool
    SA_Oracle_Client);
    printf("We are connected!\n");
// ühenduse katkestamine on valikuline
// automaatne ühenduse katkestamine tehakse destruktoris
con.Disconnect();
printf("We are disconnected!\n");
```

2.4.2 ODBC

ODBC (Open Database Connectivity) pakub standardse liidese andmebaasihaldamise süsteemidega töötamiseks. ODBC loojate eesmärk oli teha see sõltumatuks programmeerimiskeeltest, andmebaasi- ning operatsioonisüsteemidest. [18]

Implementatsioon

ODBC implementatsioonid töötavad paljudes operatsioonisüsteemides, kaasa arvatud Microsoft Windowsis, Unixis, Linuxis, OS/2's,OS/400's, IBM i5/OS'is ning Mac OS Xis. Eksisteerib sadu ODBC draivereid DBMS'idele: Oracle, DB2, Microsoft SQL Server, Sybase, Pervasive SQL, IBM Lotus Domino, MySQL, PostgreSQL, OpenLink Virtuoso ning töölaua andmebaaside tooted nagu FileMaker ning Microsoft Access. Samuti leidub draivereid ka CSV failidele.

Microsofti ODBC

Microsoft väljastas esimese ODBC versiooni teegina Microsoft Windowsile. Alates 2006. aastast tarnib Microsoft omaenda ODBC Windowsi koosseisus. Microsoft isegi lisab ODBC draivereid, mis tagab piiratud andmebaasisarnaseid päringuvahendeid Exceli töölehtedele spreadsheet'idele ning andmefailidele.

iODBC

iODBC (Independent Open DataBase Connectivity) pakub vaba lähtekoodiga platvormisõltumatut implementatsiooni nii ODBC kui ka X/Open spetsifikatsioonidele, mida kasutatakse tavaliselt teistel platvormidel kui Microsoft Windows. OpenLink Software hooldab ning toetab iODBC projekti ning levitab oma tarkvara LGPL või/ja BSD litsentsi alusel. Apple implementeeris iODBC Mac OS X'ile ning Darwinile, alustades Darwin 6.0 ning Mac OS X v10.2'st.

Programmeerijad on portinud iODBC mitmetele teistele operatsioonisüsteemidele nagu Mac OS 9, Linux (x86, x86-64, IA-64, Alpha, MIPS, ja ARM), Solaris (SPARC ja x86), AIX, HP-UX (PA-RISC ja Itanium), Compaq Tru64, Digital UNIX, Dynix, Generic UNIX 5.4, FreeBSD, DG-UX, OpenVMS ja AmigaOS.

IBM i5/OS

IBM i5/OS'is, IBM'i DB2 implementatsioon toetab ODBC'd. Vahendaja tagab ODBC drivereid Windowsile ning JDBC'le osana serveri ja kliendi pakkides. IBM'il on samuti teenus DB2 Connect, mida kasutatakse tagamaks ODBC funktsionaalsus Z/OS'ile ning Universal Database DB2 Environments'ile.

UnixODBC

UnixODBC on avatud lähtekoodiga projekt, mis implementeerib ODBC liidese. Lähtekoodi levitatakse GNU GPL litsentsi alusel ning saab kasutada mitmetel erinevatel operatsioonisüsteemidel nagu Unix, Linux, Mac OS X, IBM OS/2 ja Microsoft Interix.

Projekti eesmärkideks on:

- Tagada arendajatele tööriistad, et oleks võimalik tuua üle Microsoft Windowsi ODBC rakendusi teistele platvormidele minimaalse koodimuutmisega.
- Ühtse ning andmebaasisüsteemide tootjatest sõltumatu programmeerimisliidese loomine
- Tagada ODBC draiverikirjutajatele tööriistad, et tuua üle draivereid platvormidele, mis ei kasuta Microsoft Windowsit.
- Tagada kasutajale hulk graafilise kasutajaliidese ning käsurea tööriistu andmebaasi haldamiseks.
- Hoida suhteid nii tasuta tarkvara kommuuniga kui ka kommertsandmebaaside vahendajatega, et tagada nende omavahelist koostöövõime.

2.5 Tehtud valikud

Jõudluse parandamiseks võetakse kasutusele Tokyo Cabinet. Andmebaaside toe laiendamiseks võetakse kasutusele ODBC. Võib ka oodata, et hästi konfigureeritud ODBC-liidesega andmebaasid pakuvat paremat jõudlust kui SQLite.

Arvestades, et Tokyo Cabinetil on vähem piiranguid tabeli suurusele, siis lahendab Tokyo Cabineti kasutuselevõtt ka andmekaevandusrakenduste probleemi. Tokyo Cabineti eelis teiste sarnaste süsteemide ees on sobiv litsents, platvormide tugi ning skaleeruvus.

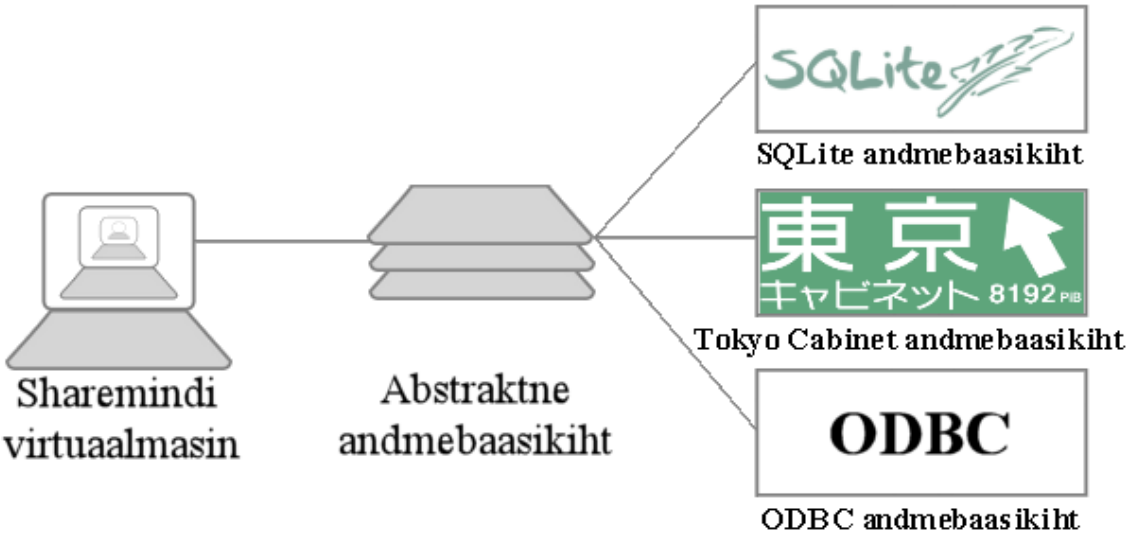
3. Andmebaasikihi abstraktseks muutmine

3.1 Muudatused arhitektuuris

Sharemindil oli varem arvutusi tegev virtuaalmasin seotud otse andmebaasikihiga (Joonis 3.1).

Joonis 3.1 Sharemindi arhitektuur enne muutmist

See tähendas seda, et puudusid alternatiivsed võimalused andmebaasimootori vahetamiseks ning kasutamine oli limiteeritud ainult SQLite andmebaasimootorile. Kuna on vaja lisada tugi uutele andmebaasidele, samal ajal säilitades SQLite toe, siis on vaja arhitektuurilist muudatust. Selleks luuakse abstraktne liides andmebaasikihtidele. C++ mõistes tähendab see abstraktse klassi loomist, mida teised andmebaasikihid pärivad (Joonis 3.2).



Joonis 3.2 Sharemindi arhitektuur peale abstraktse liidese loomist

3.2 Muudatused konfigureerimises

Varem oli igal Sharemindi andmekaevuri serveril konfiguratsiooniplokk, mis konfigureeris SQLite'i andmebaasisüsteemi. Uues süsteemis on aga võimalik valida, millist andmebaasisüsteemi kasutatakse. Selleks tuleb vastava andmekaevuri konfiguratsioonifailis muuta andmebaasimootorit.

Väljavõte konfiguratsioonifailist:

```
[VirtualMachine]
# Supported options are: SQLITE, ODBC, TOKYOCABINET
# See the engine-specific configuration options
DatabaseEngine=TOKYOCABINET
```

Igale andmebaasikihile tekib ka oma enda spetsiifiline konfiguratsiooniväärtuste plokk, mida töödeldakse siis, kui see konkreetne andmekiht valitud on.

3.3 Muudatused Sharemindi ehitussüsteemis

Kuna kõigis süsteemides ei tarvitse ODBC ja Tokyo Cabineti teegid saadavalt olla, siis on Sharemindi ehitussüsteemi täiendatud. Sharemindi ehitamisel on kaks põhisammu - konfigureerimine ja ehitamine. Ehitussüsteem põhineb programmil GNU Autotools, mis võimaldab ehitussüsteemil automaatselt tuvastada ja üles leida süsteemis paigaldatud teegid.

Tokyo Cabineti toe lisamiseks on vaja ./configure skriptile lisada võti --with-tokyo-cabinet=PATH, kus PATH on Tokyo Cabineti päiste ja teegi asukoht süsteemis. Sarnaselt, ODBC jaoks on võti --with-odbc=PATH. Kui neid võtmeid skripti välja kutsudes ei lisata, siis ehitatakse Sharemind ainult SQLite andmebaasi toega.

3.4 Muudatused süsteemi käivitamises

Süsteemi käivitamisel loeb andmekaevur konfiguratsioonifailides oleva konfiguratsiooni ning sõltuvalt loetud väärtusest proovib luua uue instantsi vastavast andmebaasikihist. Kui selgub, et konkreetne Sharemindi versioon sellist andmekihti ei toeta (sest see on tundmatu või siis pole Sharemind selle toega kompileeritud), siis antakse veateade ja töö peatub.

Pärast andmebaasikihi loomist antakse sellele üle konfiguratsioon ning see käivitatakse.

4. Tokyo Cabineti andmebaasikiht

4.1 Eesmärgid

Tokyo Cabinet võimaldab Sharemindil teha rohkemate veergudega andmebaase, kuna veergude arv pole nii limiteeritud kui SQLite andmebaasi puhul. Samuti on failivormingust lähtuvalt oodata ka jõudluse kasvu.

4.2 Konfigureerimine

Selleks, et konfigureerida Tokyo Cabineti andmebaasi, tuleb konfiguratsioonifailis muuta vastavat konfiguratsiooniplokki. Tokyo Cabineti konfiguratsiooniplokis on hetkel ainult üks muutuja: *DatabaseFolder*. See määrab ära kausta, kuhu andmebaas luuakse.

Väljavõte konfiguratsioonifailist:

```
...
[TokyoCabinet]
DatabaseFolder=miner1/tokyocabinet
...
```

4.3 Tööpõhimõtted

Tokyo Cabineti andmebaasi ülesehitus erineb mõnevõrra tavaliste andmebaaside ülesehitusest. Andmebaase hoitakse kaustades, ehk iga kaust tähistab ühte andmebaasi. Tabeleid hoitakse faili kujul. Teisisõnu, üks fail tähistab ühte tabelit. Igal tabelil on olemas ka talle vastav metaandmete tabel. See tähendab, et ühe tabeli jaoks luuakse alati kaks faili. Andmete lisamine tabelisse käib järgmise põhimõtte alusel: kõigepealt arvutatakse sisestava väärtuse suurus baitides. Seejärel pannakse see väärtus tähtmärkide massiivi. Lõpuks lisatakse tähtmärkide massiiv tabelisse vastava meetodiga.

Väljavõte koodist:

```
...
// arvutatakse väärtuse suurus baitides
int iSize = sizeof(values[i]);
// luuakse tähtmärkide massiiv baitide suurusega
unsigned char input[iSize];
// kopeeritakse väärtus baitide massiivi
memcpy (input, &values[i], iSize);
// baitide massiiv lisatakse tabelisse
tcmapput2(cols, vecS[i].c_str() , (const char*)input);
...
```

4.4 Realisatsioon

Tokyo Cabinet pakub väga palju erinevaid liideseid andmebaasi loomiseks ja haldamiseks. Meie kasutasime peamiselt *Table Database* liidest, mis lubas meil jäljendada tabeli-tüüpi andmebaase, kus on olemas read ning veerud. Igal kirjel on endiselt olemas primaarvõti, kuid meil lubatakse oma tahte järgi luua veergudele indekseid ning isegi nende peal päringuid koostada.

5. ODBC andmebaasikiht

5.1 Eesmärgid

ODBC võimaldab ühendada rohkemate erinevate andmebaasimootoritega ning seega tõsta Sharemindi vastuvõetavust ettevõtetes, kus on kasutusel konkreetsed suured andmebaasisüsteemid. Andmebaasikihi omadused sõltuvad ka reaalsest andmebaasist, mis ODBC abil ühendatakse.

5.2 Konfigureerimine

Selleks, et konfigureerida ODBC andmebaasi, tuleb sarnaselt Tokyo Cabinetile konfiguratsioonifailis muuta vastavat konfiguratsiooniplokki. ODBC konfiguratsiooniplokis on neli muutujat: *hostname* – arvuti nimi, kuhu ühendatakse, *datasource* – andmeallika nimi, kuhu ühendatakse, *username* – kasutajanimi, *password* – salasõna.

Väljavõte konfiguratsioonifailist:

```
...
[ODBCDB]
Hostname=localhost
DataSource=SimpleDataTestDB.miner1
Username=root
Password=sql

...
```

Selleks, et andmeallikasse ühendada, peab süsteemis vastav ODBC andmeallikas süsteemis konfigureeritud olema.

5.3 Tööpõhimõtted

Selleks, et andmekaevurid saaks kasutada ODBC andmebaasi, tuleb vastava süsteemi ODBC halduri alt defineerida vastavad andmeallikad ning valida draiverid, mille andmebaase tahetakse kasutada. Näiteks MySQL, PostgreSQL või CSV. Andmeallika nimeks tuleb kirjutada andmebaasi nimed, mida andmekaevurid oma rakendustes kasutavad.

Näiteks MySQL korral on ODBC andmebaasi ühendamiseks vaja alla laadida MySQL Connector kodulehelt [19]. Peale selle paigaldamist on ODBC halduri alt võimalik lisada andmeallikas vastava MySQL draiveriga. Teiste andmebaasimootorite jaoks tuleb analoogiliselt nende kodulehelt laadida alla ODBC draiver ning see paigaldada.

Sharemindi puhul on tegemist SQL baasidega, kus andmebaasitabel on Sharemindi tabel ning kus kehtivad SQL päringud. Tabeleid koostatakse andmetüübiga *unsigned int*. Metaandmete jaoks on eraldi tabel, nii nagu ka Tokyo Cabineti ning SQLite'i puhul.

5.4 Realisatsioon

Erinevaid ODBC draivereid on väga palju, kuid selles projektis kasutasime MySQL ning PostgreSQL andmebaasidraivereid. Andmebaasi väärtuste sisestamine, nende tagastamine ning kõik tabeliga seotud operatsioonid toimuvad läbi SQL päringute.

6. Tulemused

6.1 Jõudlus

Sooritame peatükis 1.3.1 tehtud jõudlustestid uuesti. Seekord kasutame andmebaasi Tokyo Cabinet. Esiteks testime andmete importi.

Tabel	**Kulunud aeg nüüd**	**Kulunud aeg varem**	**Paranemine**
mushroom	*5,28s*	*56,78s*	*~10,8 korda*
chess	*2,11s*	*23,46s*	*~11,1 korda*

Tabel 3. Andmete impordi jõudlus

Seejärel testime andmebaasist lugemise jõudlust.

	Andmekiht			**Algoritm kokku**		
Tabel	**Kulunud aeg nüüd (ms)**	**Kulunud aeg varem (ms)**	**Paranemine**	**Kulunud aeg nüüd (ms)**	**Kulunud aeg varem (ms)**	**Paranemine**
mushroom	1565	3763	~2,4 korda	25280	25018	Ei paranenud
chess	310	537	~1,7 korda	404009	460537	~1,1 korda

Tabel 4. Andmebaasist lugemise jõudlus

Vaadates tulemusi, võib öelda, et jõudluse parandamise eesmärk sai täidetud. Eriti suurt tõusu kiiruses on näha andmete importimisel. Erandina on näha, et tabeli mushroom korral algoritmi kiirus ei paranenud, kuid selle võib pigem kanda puhtalt algoritmist tingitud iseärasuste peale, kuna andmekihi lugemine paranes sellest hoolimata.

6.2 Andmebaasimootorite tugi

ODBC andmebaasimootor annab meile võimaluse kasutada ka muid andmebaase kui SQLite. Testisime ODBC andmebaasikihti MySQLiga. Tegime läbi andmete impordi, testimaks, et MySQL draiver ODBC puhul töötab ning lisasime võrdluse Tokyo Cabineti ning SQLite'iga.

Tabel	**Kulunud aeg**		
	MySQL	**SQLite**	**Tokyo Cabinet**
mushroom	*9,76s*	*56,78s*	*5,28s*

Tabel 5. Andmete impordi jõudlus ODBC andmebaaside korral

6.3 Andmekaevanduse tugi

Tokyo Cabineti toe lisamisega paranes ka tugi andmekaevandusele. Nimelt õnnestus importida ka need transaktsiooniandmed, mida varem SQLite' veerupiirangu tõttu importimata jäid.

Tabel	**Kulunud aeg**	**Transaktsioonide arv**
mushroom	*5,28s*	8124
chess	*2,11s*	3196
connect	41,72s	67557
pumsb	5m 35,51s	49046

Tabel 6. Suurte veergude arvuga andmete impordi jõudlus

Lisaks on uue andmebaasikihi testimise käigus edukalt tehtud tööd ka kuni 500 000 veerulise geeniandmete baasiga.

7. Kokkuvõte

Sharemind on andmetöötlussüsteem, mis töötleb andmeid, ilma et ta neid näeks. Süsteem põhineb krüptograafilisel alusel ning see koosneb kolmest andmekaevurist, mis salvestavad ning töötlevad andmeid ning kontrollerrakendusest, mis võimaldab andmeid sisestada ning mis kordineerib andmekaevurite tööd

Andmebaasikiht põhineb SQLite'il ning see on igas andmekaevuris. Andmebaasikihti kasutatakse andmete salvestamiseks. See võimaldab teha kõiki enimkasutatavaid andmebaasipäringuid.

Süsteemis kasutatava andmebaasimootori jõudlus on väike ning puudub teiste andmebaasimootorite tugi. Samuti puudub võimalus luua suuri andmebaase(näiteks geeniandmete andmebaas), kus veergude arv võib ulatuda poole miljonini või rohkem. Käesoleva töö eesmärgiks oli lisada mõne muu andmebaasimootori tugi, selle käigus parandada jõudlust ning tagada parem andmekaevanduse tugi.

Kuna Sharemind on kirjutatud C++ keeles, siis oli meil vaja otsida C++ teeke ja tehnoloogiaid. Valik tuli teha kolme kategooria vahel, mis jagunesid järgnevalt: Andmebaasimootorite oma teegid, integreeritud andmebaasid või andmebaasiühendusi abstraheerivad teegid. Integreeritud andmebaaside hulgast (Berkeley DB, Tokyo Cabinet) valisime Tokyo Cabineti oma väga heade jõudlusnäitajate, suurte andmebaaside loomise võimaluse ning litsentsi poolest. Lisaks jõudluse parandamisele oli meil soov suurendada tuge teistele andmebaasimootoritele. Selleks valisime andmebaasiühendusi abstraheerivate teekide (ODBC, SQLAPI++) hulgast ODBC oma lihtsuse ning sisseehitatud toe poolest mitmetel platvormidel.

Selleks, et lisada meie valitud andmebaasimootorid, tuli Sharemindis teha arhitektuuriline muudatus. Tuli luua abstraktne liides andmebaasikihtidele, kuna varem oli arvutusi tegev virtuaalmasin seotud otse andmebaasikihiga. Lisaks sellele muutus Sharemindi konfigureerimine, ehitussüsteem ning käivitamine.

Uue andmebaasi kasutuselevõtuks tuleb muuta vastavas konfiguratsioonifailis muutujat, mis määrab, missugust andmebaasi hetkel kasutatakse. Samuti on olemas igale andmebaasile oma konfiguratsiooniplokk, kus saab muuta andmebaasile omaseid muutujaid.

Testimine vana ning uue süsteemi vahel näitas selget paranemist kiiruses. Tehtud testid näitasid, et Tokyo Cabineti andmebaasikihiga paranes baasi kirjutamise jõudlus ligikaudu 11 korda ning baasist lugemise jõudlus ligikaudu 2 korda. Samuti on tagatud parem tugi andmekaevandusele. Lisaks on ODBC andmebaasikihiga tagatud tugi teiste andmebaasimootorite jaoks. Seega, töö eesmärk sai täidetud.

8. Summary

Sharemind is a data processing system capable of performing computations on input data without compromising its privacy. Basically, it can process data without seeing it. The system is based on solid cryptographic foundations. Secret sharing is used to preserve the privacy of the data and secure multi-party computation allows us to work with the data.

A working system consists of three *miners* who store and process the data and *controller* which provides input and coordinates the miners' work.

Sharemind uses SQLite for its database layer and it cannot be changed. This is a major downside because other companies use more powerful and flexible systems like MySQL, PostgreSQL, Oracle and Microsoft SQL Server. It is also very slow and there is a limit for how much columns it can store. Because of that, the current database system is a bottleneck of the whole system.

The aim of this work was to improve Sharemind by implementing a better database system and add support to other database systems other than SQLite. There were many options to choose from but in the end the best choice was to use ODBC (Open Database Connectivity) and Tokyo Cabinet. We chose ODBC for its simplicity and built-in support on many systems other than Windows. Tokyo Cabinet itself is very efficient in terms of speed and space and it allows us to create tables which contain at least 500 000 columns.

In order to use our new database systems, we had to change Sharemind's architecture. Before our change the Sharemind's Virtual Machine was directly connected to the SQLite database. We had to create an abstract database layer which our other databases could inherit. In addition to that there were also changes in the Sharemind configuration, build-system and startup. There are now also configuration files where you can change the database which the system uses. Each database has also its own configuration block where you can assign values specific to the database.

Testing done on the old system and on the new system with Tokyo Cabinet show an improvement in speed roughly by 11 times when writing to the database and roughly 2 times when reading from the database. In addition to that, Sharemind is now not limited to using a single database engine. Therefore, we succeeded in our goal.

9. Viited

1. Privaatsust säilitav arvutussüsteem Sharemind
http://research.cyber.ee/sharemind/ Viimati külastatud 19.01.2011 20:06

2. Dan Bogdanov, Sven Laur, and Jan Willemson. Sharemind: A framework for fast privacy-preserving computations. ESORICS '08, lk 192–206, 2008. Springer.

3. Frequent Itemset Mining Implementations Repository
http://fimi.cs.helsinki.fi/data/ Viimati külastatud 19.01.2011 20:06

4. Roman Jagomägis. SecreC: a Privacy-Aware Programming Language with Applications in Data Mining. Magistritöö lk 31-32. Tartu 2010.

5. NoSQL andmebaasid
http://en.wikipedia.org/wiki/NoSQL Viimati külastatud 19.01.2011 20:06

6. Libmysql teek
http://dev.mysql.com/doc/refman/5.0/en/connector-c-faq.html Viimati külastatud 19.01.2011 20:06

7. Libmysql teegi koodinäide
http://forge.mysql.com/wiki/Connector_C%2B%2B#Getting_Started:_Usage_Examples Viimati külastatud 19.01.2011 20:06

8. Oracle C++ Call Interface
http://www.oracle.com/technology/tech/oci/occi/index.html Viimati külastatud 19.01.2011 20:06

9. Oracle C++ Call Interface koodinäide
http://www.tidytutorials.com/2009/08/oracle-c-occi-database-example.html Viimati külastatud 19.01.2011 20:06

10. Integreeritud andmebaasid
http://www.wisegeek.com/what-is-an-integrated-database.htm Viimati külastatud 19.01.2011 20:06

11. SQLite teek
http://www.sqlite.org/ Viimati külastatud 19.01.2011 20:06

12. SQLite koodinäide
http://www.coderetard.com/2008/12/09/sqlite-with-c-example-a-compact-serverless-database-alternative/ Viimati külastatud 19.01.2011 20:06

13. BerkeleyDB teek
http://www.theonestopwebsiteshop.com/web-design/database-type.htm Viimati külastatud 19.01.2011 20:06

14. BerkeleyDB koodinäide
http://download.oracle.com/docs/cd/E17076_02/html/gsg/CXX/databases.html#DBOpen Viimati külastatud 19.01.2011 20:06

15. Tokyo Cabineti teek
http://fallabs.com/tokyocabinet/ Viimati külastatud 19.01.2011 20:06

16. SQLAPI++ teek
http://www.sqlapi.com/Order/index.html Viimati külastatud 19.01.2011 20:06

17.SQLAPI++ koodinäide
http://www.sqlapi.com/Examples/step1.cpp Viimati külastatud 19.01.2011 20:06

18.ODBC teek
http://en.wikipedia.org/wiki/Open_Database_Connectivity Viimati külastatud 19.01.2011 20:06

19. ODBC MySQL Connector
http://dev.mysql.com/downloads/connector/odbc/3.51.html Viimati külastatud 19.01.2011 20:06

Lisa 1

Tarkvara lähtekood (asub CD-l)