Grafiikkasuorittimen käyttö keskusmuistitietokannoissa

Matti Nauha

Helsinki 9.3.2012

HELSINGIN YLIOPISTO
Tietojenkäsittelytieteen laitos

HELSINGIN YLIOPISTO – HELSINGFORS UNIVERSITET – UNIVERSITY OF HELSINKI

Tiedekunta – Fakultet –Faculty	Laitos – Institution – Department
Matemaattis-luonnontieteellinen tiedekunta	Tietojenkäsittelytieteen laitos

Tekijä – Författare – Author

Matti Nauha

Työn nimi – Arbetets titel – Title

Grafiikkasuorittimen käyttö keskusmuistitietokannoissa

Oppiaine – Läroämne – Subject

Tietojenkäsittelytiede

Työn laji – Arbetets art – Level	Aika – Datum – Month and year	Sivumäärä – Sidoantal – Number of pages
	9.3.2012	10 sivua

Tiivistelmä – Referat – Abstract

Avainsanat – Nyckelord – Keywords

Keskusmuistitietokanta, CUDA, GPGPU, Tietokanta, SQL

Säilytyspaikka – Förvaringställe – Where deposited

Muita tietoja – Övriga uppgifter – Additional information

Sisältö

1 Johdanto

Näytönohjaimesta (GPU – graphics processing unit) on tullut olennainen osa nykyaikaista tietojenkäsittelyä. Viime vuosina näytönohjaimien suorituskyky ja ominaisuudet ovat kehittyneet merkittävästi. Moderni näytönohjain ei ole ainoastaan tehokas grafiikkasuoritin, vaan se on myös rinnakkaislaskennassa erittäin tehokas ohjelmoitava suoritin, joka peittoaa niin aritmeettisten operaatioiden suoritusmäärissä, kuin muistikaistanleveydessäkin perinteiset prosessorit. Näytönohjaimien suorituskyvyn kasvaessa ja niiden ohjelmoinnin helpottuessa on lisännyt nopeasti mielenkiintoa näytönohjaimien laskentatehon valjastamiseksi yleiseen laskentaan [OHD08] [BS10].

Tietokannat ovat nykypäivänä osa lähes jokaista tietojärjestelmää, ja niiden laskennalliset vaatimukset asettavat suuria haasteita. Yksi mielenkiintoa herättänyt ratkaisu onkin näytönohjaimien suorituskyvyn hyödyntäminen tietokantojen laskennassa.

Tässä seminaarityössä esitellään näytönohjaimien tarjoamia mahdollisuuksia tietokantojen suorituksen nopeuttamiseksi. Luvussa kaksi käsitellään aihetta hieman yleisemmin, eli perehdytään siihen, miten näytönohjaimet toimivat, ja miten niillä voidaan suorittaa yleistä laskentaa. Luvussa kolme esitellään Nvidian CUDA teknologia, joka on toiminut suurena vaikuttimena näytönohjaimille tarkoitettujen sovellusten ohjelmoinnin helpottamisessa. Luvussa neljä esitellään SQLite tietokanta, joka toimii hyvänä esimerkkinä yksinkertaisesti toteutetusta ja yleisesti käytetystä tietokannasta, ja jota käytetään myös tässä seminaarityössä esitellyissä tutkimuksissa. Luvussa viisi esitellään kolme keskeistä tutkimusta, joissa ehdotetaan ja tarkastellaan erilaisia vaihtoehtoja näytönohjaimien hyödyntämiseksi tietokantojen laskennassa.

2 GPGPU

GPGPU eli General-purpose computing on graphics processing units on tekniikka, jossa näytönohjain suorittaa laskentaa, joka normaalisti kuuluisi prosessorille (CPU – central processing unit). [WGPU] Tyypillisesti näytönohjaimella suoritetaan ainoastaan grafiikkaan liittyvää laskentaa ja näin ollen ne ovat suunniteltuja suorittamaan hyvin

juuri tietynlaisia laskennallisia tehtäviä, joissa [OHD08]:

- Laskennalliset vaatimukset ovat suuria: Näytönohjaimien pitää kyetä käsittelemään miljardeja pikseleitä sekunnissa. Monimutkaiset reaaliajassa ajetut sovellukset vaativat valtavaa laskentatehoa.
- Rinnakkaisuus on olennaista
- Kerralla suoritettavan laskennan määrä on tärkeämpää, kuin yksittäisen operaation nopea suoritus.

Näytönohjaimet ovat perinteisiin prosessoreihin nähden hyvin tehokkaita rinnakkaislaskennassa [HL09].

2.1 Näytönohjaimen arkkitehtuuri

Näytönohjaimilla on erityinen arkkitehtuuri, jonka keskiössä on suuri määrä pieniä rinnakkaisia laskentayksiköitä. Näytönohjaimet ovat aina olleet suorittimia, jotka tarjoavat runsaasti laskentatehoa. Viime aikojen tärkeimpänä suuntauksena on ollut tuon laskentatehon tuominen ohjelmoijien oluttuville. Näytönohjaimet ovat kehittyneet yhteen asiaan erikoistuneista suorittimista täysimittaisiksi ohjelmoitaviksi rinnakkaissuorittimiksi, jotka taipuvat monenlaisiin tehtäviin. [OHD08]

2.2 Stream prosessointi

Näytönohjaimen laskennassa *stream* on joukko tietueita, joille halutaan suorittaa samankaltaisia laskutoimituksia. *Kernelit* ovat funktioita, joita sovelletaan jokaiseen streamin elementtiin. Näytönohjain prosessoi kaikki elementit itsenäisinä osina, joten niillä ei voi olla staattista jaettua tietoa. Jokainen elementti luetaan, sille suoritetaan operaatioita ja lopuksi kirjoitetaan tulos.

Ihanteellisessa GPGPU sovelluksessa käytetään suuria tietojoukkoja (data set), paljon rinnakkaisuutta ja pidetään tietoelementtien välinen riippuvuus minimissään [WGPU].

Yleisin stream syötteen muoto GPGPU laskennassa on kaksiulotteinen matriisi, mikä sopii hyvin luonnollisesti näytönohjaimille. Monenlainen laskenta, kuten matriisialgebra, kuvien prosessointi ja fysiikkasimulaatio käyttävät luonnostaan kaksiulotteisia matriiseja.

3 CUDA

Yksi olennainen ongelma näytönohjaimien hyödyntämisessä yleisessä laskennassa on ollut se, että vaikka kehitettävällä sovelluksella ei olisi mitään tekemistä grafiikan kanssa, on se pitänyt kuitenkin ohjelmoida grafiikkaohjelmointirajapintojen (graphics API) avulla. Tämän vuoksi grafiikkaohjelmointia on pidetty vaikeana ja monimutkaisena [HL09].

CUDA (*Compute Unified Device Architecture*) on NVIDIA:n kehittämä rinnakkaislaskenta-arkkitehtuuri. CUDA tarjoaa C-kielen kaltaisen ohjelmointirajapinnan, jonka avulla ohjelmistokehittäjät pääsevät hyödyntämään näytönohjaimien tarjoamaa laskentatehoa sovelluksissaan [CUDA] [WCUDA]. Näin näytönohjainta päästään käyttämään laskentaan tuttuun tyyliin ja näytönohjainta hyödyntävän sovelluksen ohjelmoiminen on hyvin samankaltaista, kuin perinteiselle prosessorille tarkoitetun sovelluksenkin ohjelmointi [BS10].

CUDA toimii kaikissa NVIDIA:n näytönohjaimissa G8x sarjasta eteenpäin.

CUDA:n ohella kilpailevia rajapintoja ovat OpenCL ja DirectCompute. Nämä rajapinnat eivät vaadi minkään tietyn valmistajan näytönohjainta toimiakseen.

4 SQLITE

SQLite on sulautettu relaatiotietokantajärjestelmä, joka on toteutettu pienenä (n. 275kt) C-kirjastona. Toisin kuin monet muut tietokannat, SQLite ei toimi erillisenä prosessina, vaan sulautettuna sitä käyttävään sovellukseen. Näin ollen SQLite ei myöskään käytä erillistä palvelinprosessia. SQLite:ssa koko tietokanta tauluineen ja näkymineen on yhdessä tiedostossa. Tietokanta voidaan pitää kokonaan tietokoneen muistissa, tai tallentaa kiintolevylle yhteen tiedostoon, joka lukitaan transaktioiden ajaksi. Muistiin tallennettu tietokanta häviää heti, kun tietokantayhteys suljetaan [BS10] [LY11].

SQLite tukee suurinta osaa SQL-kielen SQL-92-standardista. Standardista puuttuvia osia ovat esimerkiksi viiteavaimet, joten SQLite ei valvo ACID-mallin yhdenmukaisuussääntöä, mutta muut osat ACIDista on toteutettu. SQLite käyttää epätavallista tietotyyppijärjestelmää: kun useimmissa tietokannoissa tyyppi on sarakekohtainen, SQLitessä tyyppi on arvokohtainen; SQLiteä voikin kuvata

dynaamisesti tyypitetyksi tietokannaksi staattisesti tyypitetyn sijaan. SQLiten tietokannan sarakeleveyksiä ei myöskään tarvitse määrittää kiinteästi jolloin tietokanta varaa muistia vain sen verran kuin sen sisällä oleva data edellyttää [BS10].

SQLite-kirjasto on public domain -ohjelmisto, joten sitä voidaan muokata ja levittää vapaasti ja se voidaan linkittää kaikkiin ohjelmistoihin ilman erillistä lupaa.

5 Tietokantaoperaatioiden nopeuttaminen näytönohjaimen avulla

Näytönohjaimien arkkitehtuuri asettaa omat rajoituksensa keskusmuistitietokantojen toteutukselle. Yksi merkittävä rajoittava tekijä on näytönohjaimen muisti. Tiedon siirto keskusmuistista näytönohjaimelle on hidasta verrattuna muuhun laskentaan, ja näin ollen selvä pullonkaula. Ratkaisuna olisi tietokannan tallentaminen kokonaisuudessaan näytönohjaimen muistiin, mutta suurimmillaankin näytönohjaimien muistit jäävät vielä liian pieniki verrattuna tavalliseen keskumuistiin [BS10].

Muistiin liittyvien rajoitusten lisäksi näytönohjaimella voidaan ajaa ainoastaan yhtä sovellusta kerrallaan, ja se luottaa keskussuorittimeen

Nykypäivänä tietokannat ovat suuryritysten työjuhtia, ja tiedon etsintä tietokannasta on jokapäiväinen laskennallinen haaste. Datamäärien kasvaessa tietokantojen toimittajat kuten Microsoft, Oracle ja IBM etsivät ratkaisua skaalautuvuuteen näytönohjaimien tarjoamasta suorituskyvystä [BS10].

Tietokantojen toiminnan nopeuttamiseksi näytönohjaimien avulla on ehdotettu monenlaisia eri lähestymistapoja, joista tässä luvussa esitellään muutama keskeinen aiheeseen liittyvää tutkimus.

5.1 SQLiten nopeuttaminen näytönohjaimella

P. Bakkum ja K. Skadron esittelevät tutkimuksessaan Accelerating SQL Database Operations on a GPU with CUDA menetelmän, jossa he toteuttavat SQLite virtuaalikoneen ja osan SQL-komentoja CUDA:lla. Tällä tavoin he vähentävät huomattavasti työmäärää, joka muutoin tarvittaisiin näytönohjaimen hyödyntämiseen tietokantaoperaatioiden nopeuttamiseksi, sillä tietokantaohjelmoijan ei tarvitse itse käyttää uudenlaisia ohjelmointikieliä, kuten CUDA:aa tai muuttaa ohjelmaansa

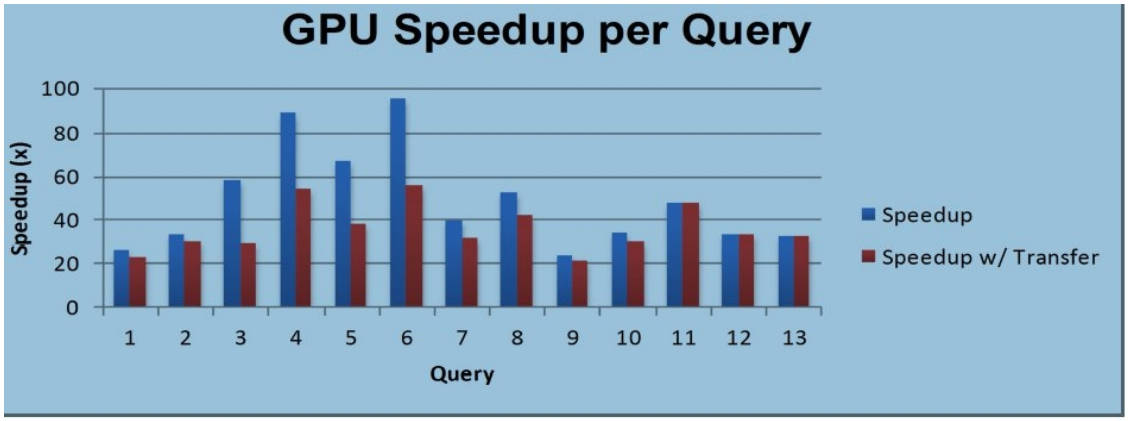
käyttämään jotakin non-SQL kirjastoa. SQL on suhteellisen yksinkertainen ja laajalti käytetty, mikä tekee siitä hyvän rajapinnan näytönohjaimen hyödyntämiseen [BS10].

Tutkimus osoittaa useaan otteeseen geneerisen rajapinnan tiedon prosessointiin näytönohjaimella olevan tehokas, ja vahvistaa näyttöä siitä, että tietokantaoperaatioiden nopeuttaminen siirtämällä kyselyitä näytönohjaimelle toimii tehokkaasti. Tutkimuksessa keskitytään ainoastaan tiedon lukemiseen, eli SELECT-kyselyin, ja niitäkin tutkitaan vain rajallisesti. Vaikka ainoastaan osa kaikista SQL-kyselyistä toteutettiin, tulokset ovat lupaavia ja on syytä uskoa, että myös kaikkien SELECT-kyselyiden toteuttaminen johtaisi samankaltaiseen lopputulokseen. Tutkimuksen tulokset osoittavat, että tietokantojen toteuttaminen näytönohjaimien raudalle vaikuttaisi olevan tuottoisa aihe tulevaisuuden tutkimukselle ja kaupalliselle kehittelylle [BS10].

Table 1: Performance Data by Query Type

Queries	Speedup	Speedup w/ Transfer	CPU time (s)	GPU time (s)	Transfer Time (s)	Rows Returned
Int	42.11	28.89	2.3843	0.0566	0.0259148	1950104.4
Float	59.16	43.68	3.5273	0.0596	0.0211238	1951015.8
Aggregation	36.22	36.19	1.0569	0.0292	0.0000237	1
All	50.85	36.20	2.2737	0.0447	0.0180920	1500431.08

Kuva 1: Taulukko kuvaa prosessorin ja näytönohjaimen eroa kyselyiden suorituksessa.



Kuva 2: Näytönohjaimen tuoma suorituskyvyn parannus kolmessatoista eri kyselyssä.

SQLiten suorittaminen näytönohjaimella saatiin aikaan toteuttamalla SQLite:n virtuaalikone CUDA kernelinä. Kyselyt, jotka suoritettiin näytönohjaimella olivat keskimäärin 35 kertaa nopeampia verrattuna prosessorilla suoritettuihin kyselyihin. Jokaisen kyselyn erityispiirteet, haettavan tiedon tyyppi ja tulosjoukon suuruus vaikuttivat suuresti prosessorilla tehtyjen ja näytönohjaimella tehtyjen kyselyjen

nopeuksien eroon. Tästä huolimatta kolmessatoista eri kyselyssä näytönohjaimella kiihdytetyt kyselyt olivat vähintään kaksikymmentä kertaa nopeampia.

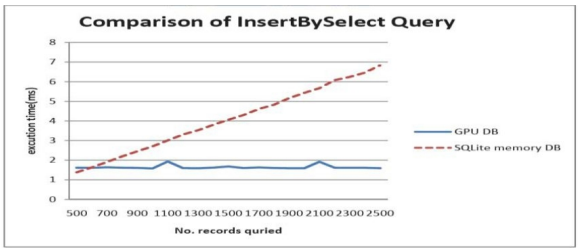
5.2 CUDADB

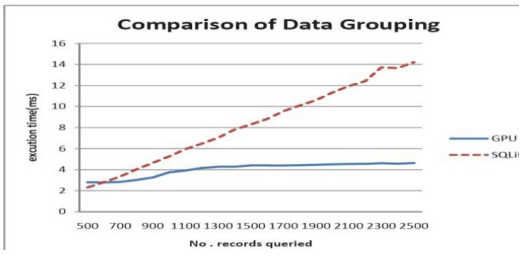
Toinen hieman edellä esitetyn kaltaisen lähestymistavan ovat esittäneet Chia-Feng Lin ja Shyan-Ming Yuan. Tutkimuksessaan he luovat CUDA:aa käyttäen oman keskusmuistitietokantatoteutuksen, jota he nimittävät CUDADB:ksi, ja vertaavat tämän suoritusta perinteisen peräkkäisesti suoritettavan SQLite tietokannan suoritukseen [LY11].

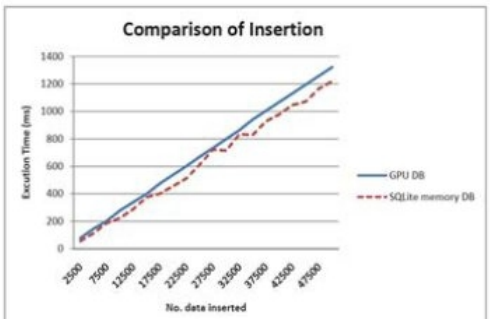
Linin ja Yuanin näkemyksen mukaan CUDA:n käyttö on paras vaihtoehto kehittäjille, kun halutaan toteuttaa keskusmuistitietokanta näytönohjaimelle. CUDA:n käyttämisessäkin on kaikesta huolimatta omat haasteensa, sillä arkkitehtuurin tarjoamia kirjastoja ei voida käyttää suoraviivaisesti esimerkiksi tietotaulujen järjestykseen tai vertailuun. Lin ja Yan muokkaavat omat versionsa valmiina tarjotuista algoritmeista, jotta ne sopisivat paremmin heidän tietokantatoteutukseensa [LY11].

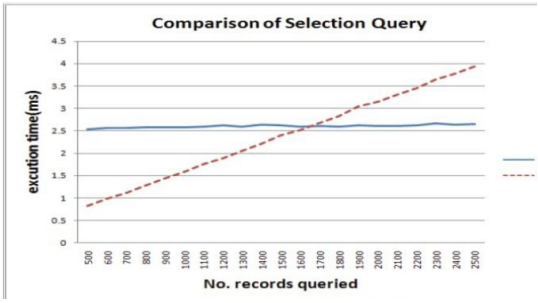
Perinteiset tietokantajärjestelmät, kuten esimerkiksi SQLite, käyttävät yleensä tiedon järjestämiseen puurakenteita, kuten B-puita, jotka ovat optimoituja peräkkäiseen laskentaan, eivätkä sovellu kovin hyvin rinnakkaislaskentaan nojautuviin arkkitehtuureihin. Niinpä CUDADB:ssä ovat käytössä kaksiulotteiset taulukot, jotka mahdollistavat jatkuvan muistiin pääsyn ja auttavat hyödyntämään koko näytönohjaimen laskentatehoa.

Tutkimuksen suorituskykumittauksesessa verrattiin CUDADB ja SQLite keskusmuistitietokantojen suoritusaikoja toisiinsa. Testidatana käytettiin virallista SQLiten tarjoamaa testidataa ja mittaukset tehtiin laitteistolla, joka koostui Intelin 4-ydin prosessorista, GeForce 9800 GT (512 Mt) näytönohjaimesta ja 2 Gt DDR-800 keskusmuistista.









Lopputulemana todetaan, että nykyaikaiset näytönohjaimet todellakin tarjoavat merkkitävää parannusta suorituskykyyn suuria datamääriä käsiteltäessä. Testitulosten perustella käsiteltävän tiedon määrä vaikuttaa huomattavasti vähäisemmin CUDADB:n suoritusnopeuteen, kuin perinteisen SQLite tietokannan toimintaan.

5.3 GDB

Bingsheng He et al. esittelevät tutkimuksessaan *Relational Query Coprocessing on Graphics Processors* näytönohjaimella tehtävään laskentaan perustuvan tietokannan GDB:n, jonka tarkoituksena on parantaa relaatiokyselyiden yleistä suorituskykyä. GDB:n toteutuksessa lähdetään luomalla CUDA:lla joukko näytönohjaimelle optimoituja primitiivejä, yleisiä operaatioita (esimerkiksi map, reduce ja split), joita käytetään rakennuspalikoina toteutettaessa yleisiä relaatiokyselyihin liittyviä operaattoreita [HL09].

Tutkimuksessa esitellään prosessointitekniikoita, jotka ottavat huomioon sekä laskennallisen suorituskyvyn, että näytönohjaimen ja prosessorin välisen tiedonsiirron kustannukset. Tällä tavoin jokainen operaattori kyselyssä voi hyödyntää sille sopivinta suoritinta, jolloin saavutetaan optimi suoritus. GDB:n avulla on siis tarkoitus mitata sekä prosessorin ja näytönohjaimen suorituksien erotusta, myös niiden yhteistyön nopeutta.

Table VII. Elapsed Time for Primitives and Joins (ms)

Primitive	CPU	GPU	Speedup
Map	109	4	27.3
Scatter	1312	104	12.6
Gather	1000	103	9.7
Prefix scan	141	14	10.1
Reduce	31	11	2.8
Filter	62	37	1.7
Split	813	125	6.5
Sort(qsort)	2313	945	2.4

Operators	CPU	GPU	Speedup
Selection	63	36.08	1.7
Projection	20	0.86	23.3
OrderBy	2500	1000	2.5
GroupBy	2323	945	2.5

Kuva 3: Taulukossa näkyvät näytönohjaimella ja keskussuorittemella suoritettujen primitiivien ja relaatio-operaatioiden nopeuserot.

Lopputuloksena tutkimuksessa todetaan, että näytönohjaimella suoritetut algoritmit ovat 2-27 kertaa nopeampia, kuin prosessorille optimoidut verrokkinsa. Lisäksi todetaan myös, että ehdotettu näytönohjaimen ja prosessorin yhteislaskenta toimii vähintään yhtä hyvin tai paremmin, kuin pelkästään näytönohjaimella tai prosessorilla suoritettu laskenta.

6 Yhteenveto

Näytönohjaimet ovat viimevuosien aikana kehittyneet yhden asian hoitavista suorittimista voimakkaiksi laskennallisiksi työvälineiksi perinteisten prosessorien rinnalle. Näytönohjaimilla on myös omat rajoitteensa: Käytössä oleva muisti on pieni verrattuna keskusmuisteihin, tiedon siirto näytönohjaimen muistilta keskusmuistiin on hidasta, niillä ei voi ajaa kuin yhtä sovellusta kerrallaan, ja ne luottavat keskussuorittimien hoitavan syötteet ja tulostukset. Näytönohjaimien laskentateho erityisesti rinnakkaislaskennassa on kuitenkin vertaansa vailla, ja kun sovellus osataan suunnitella siten, että rajoitukset otetaan huomioon, tarjoavat näytönohjaimet parhaimmillaan monikymmen kertaisia parannuksia sovellusten suorituskykyyn.

Perinteisesti näytönohjaimille tarkoitettujen sovellusten ohjelmoiminen on ollut hyvin vaikeaa. Uudet teknologiat kuten Nvidian CUDA ovat kuitenkin helpottaneet näytönohjainohjelmointia huomattavasti, mikä on vauhdittanut kiinnostusta näytönohjaimien käyttöön sovellusten nopeuttamiseksi. Yksi tehtävä, mihin

näytönohjainten laskenta soveltuu hyvin, ovat tietokantaoperaatiot. Tietokantojen nopeuttamisesta näytönohjainten avulla onkin tehty useita tutkimuksia, joissa yleensä toteutetaan tietokanta tai osia siitä CUDA:lla, ja verrataan sitten suoritusta johonkin perinteiseen peräkkäisesti suoritettavaan tietokantaan.

Vaikka tutkimukset ovatkin tiukasti rajattuja, vaikuttavat tulokset lupaavilta. Tutkimustulokset ovat osoittaneet, että näytönohjaimen käyttö tarjoaa usein moninkertaisesti nopeamman suorituksen, kuin perinteisen prosessorin käyttö. Vaikuttaa selvästi siltä, että tietokantojen nopeutus näytönohjaimien avulla on hyvä aihe sekä lisätutkimuksille, että kaupalliselle kehitykselle.

Lähteet

OHD08 Owens, Houston, Luebke, Green, Stone, Phillips, GPU computing, Proceedings of the IEEE, 2008

BS10 Bakkum, P., Skadron, K., Accelerating SQL Database Operations on a GPU with CUDA, 2010

LY11 Lin, Yuan, The Design and Evaluation of GPU based Memory Database, 2011

HL09 He, Lu, Yang, Fang, Govindaraju, Luo, Sander, Relational Query Coprocessing on Graphics Processors, ACM Transactions on Database Systems 2009

CUDA http://developer.nvidia.com/what-cuda [9.3.2012]

WCUDA http://en.wikipedia.org/wiki/CUDA[9.3.2012]

WGPU http://en.wikipedia.org/wiki/GPGPU [9.3.2012]