HAUTE ECOLE EN HAINAUT

UMONS Université de Mons

Improving performances of an embedded RDBMS with a hybrid CPU/GPU processing engine

Samuel Cremer[1,2], Michel Bagein[1], Saïd Mahmoudi[1], Pierre Manneback[1]

[1]UMONS, University of Mons
Computer Science Department, University of Mons, Rue de Houdain 9, 7000, Mons, Belgium

[2]HEH, Haute Ecole en Hainaut
Computer Engineering Department, Haute Ecole en Hainaut, Av. Maistriau 8A, 7000, Mons, Belgium

samuel.cremer@heh.be
michel.bagein@umons.ac.be
said.mahmoudi@umons.ac.be
pierre.manneback@umons.ac.be

Context (1)

- **Exponential growth of data volumes**
  - Big Data and NoSQL
  - Not only data centers -> end-user applications
- **RDBMS -> still essential**
- **Embedded RDBMS (SQLite, MySQL embedded, SQL Server Compact)**
  - Targets: personal computers, embedded devices and servers
  - Used as: storage system
local cache system -> In-Memory DB
  - Does not take advantage of current hardware specificities

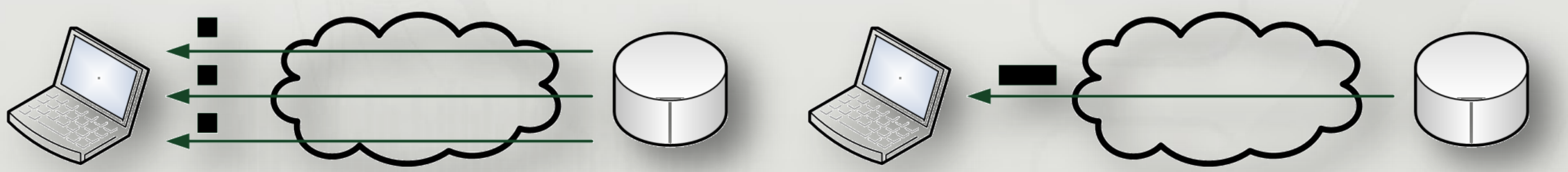
Context (2)

- **Idea:**

improving the performances of SQLite

with a hybrid implementation over multicore CPU and GPU (CuDB)

- **Benefits of faster Embedded RDBMS :**
  - Better latencies
  - Better energy efficiency
  - Processing of larger data volumes



Why using a GPU ?

- **GPUs -> widely available**

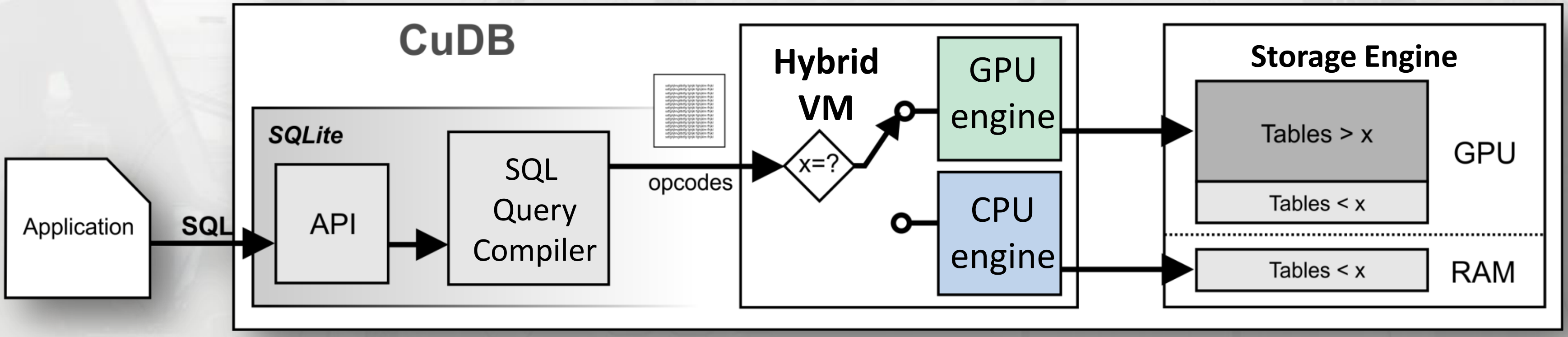
- **GPUs are SIMT architectures (Single Instruction, Multiple Threads)**
  - Fast for processing a same instruction on different data
  - SQL -> processing a same query on different rows

- **Compared to CPUs, GPUs have overall better :**

	GeForce GTX 1080 (~800€)	**Xeon E5-1660 v4 (~1000€)**
Number of cores:	2560	8 (16 threads)
Computing power:	~9000 Gflops	~800 GFlops
Memory bandwidth:	~300 GB/s	~80 GB/s
Energy efficiency:	50 GFlops/W	6 GFlops/W

- **Offloads the CPU**

CuDB: Internal Architecture



Hybrid VM chooses to execute processing, either on CPU cores or GPU cores according to the data volume they have to process.

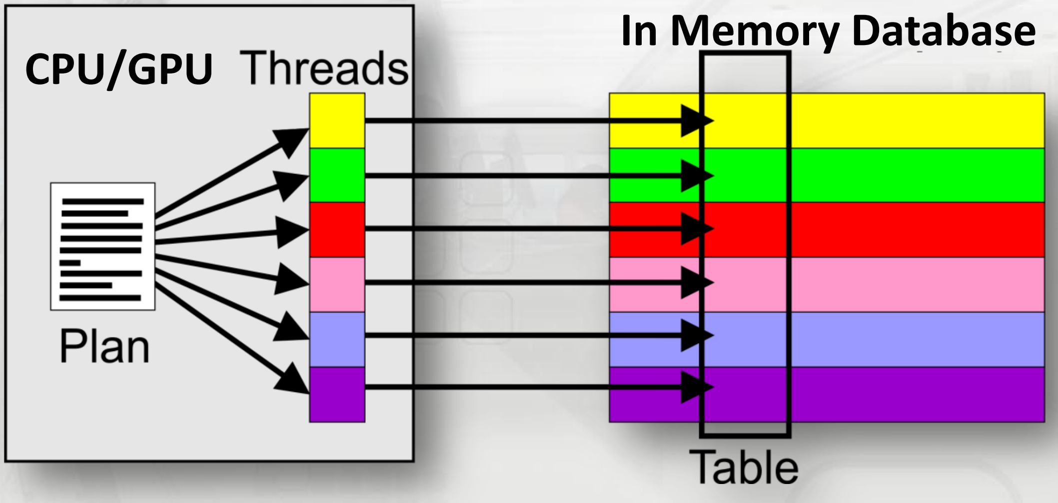
x = size of the biggest accessed table (threshold = ~1000 records)

GPU engine uses CUDA threads / CPU engine uses POSIX threads

Entire database is in GPU global memory: “In-GPUMem DB”

CuDB: Specificities

SELECT queries are boosted by S(QP)MD paradigm (Single Query Plan, Multiple Data)



Insertions are processed asynchronously by the CPU

Multiple storage engines : - **Affinity** (row order and dynamic typing)

- **Boost** (column order and static typing) **<- fastest engine**

Experimental results: Hardware

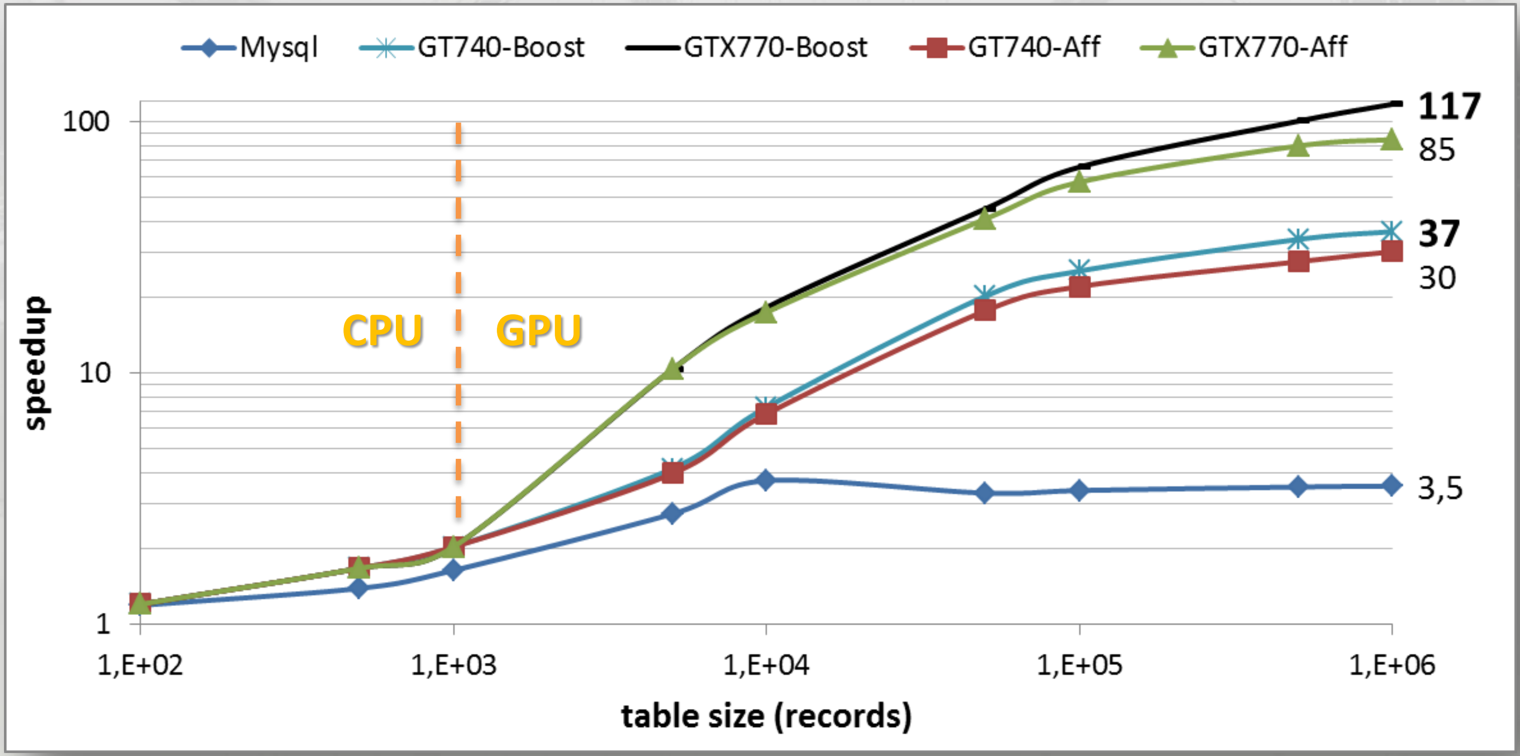
	Intel Core i7 2600K	GeForce GT740 GDDR5	GeForce GTX 770
Cores	4 (8 Threads)	**384**	**1536**
Frequency	**3.4 – 3.8 GHz**	~1 GHz	~1 GHz
Memory Bandwidth	**21,4 GB/s**	**80 GB/s**	**224 GB/s**
Computing Power (SP)	217 GFlops	762 GFlops	3.213 GFlops
TDP	95 W	64 W	230 W

- **Only focused on extraction queries** -> execution time of prepared statements

- **CuDB compared to:**
  - SQLite with an In-Memory database
  - MySQL 5.7 with MEMORY tables

- **Transfer times required to send query-plans and results were considered**

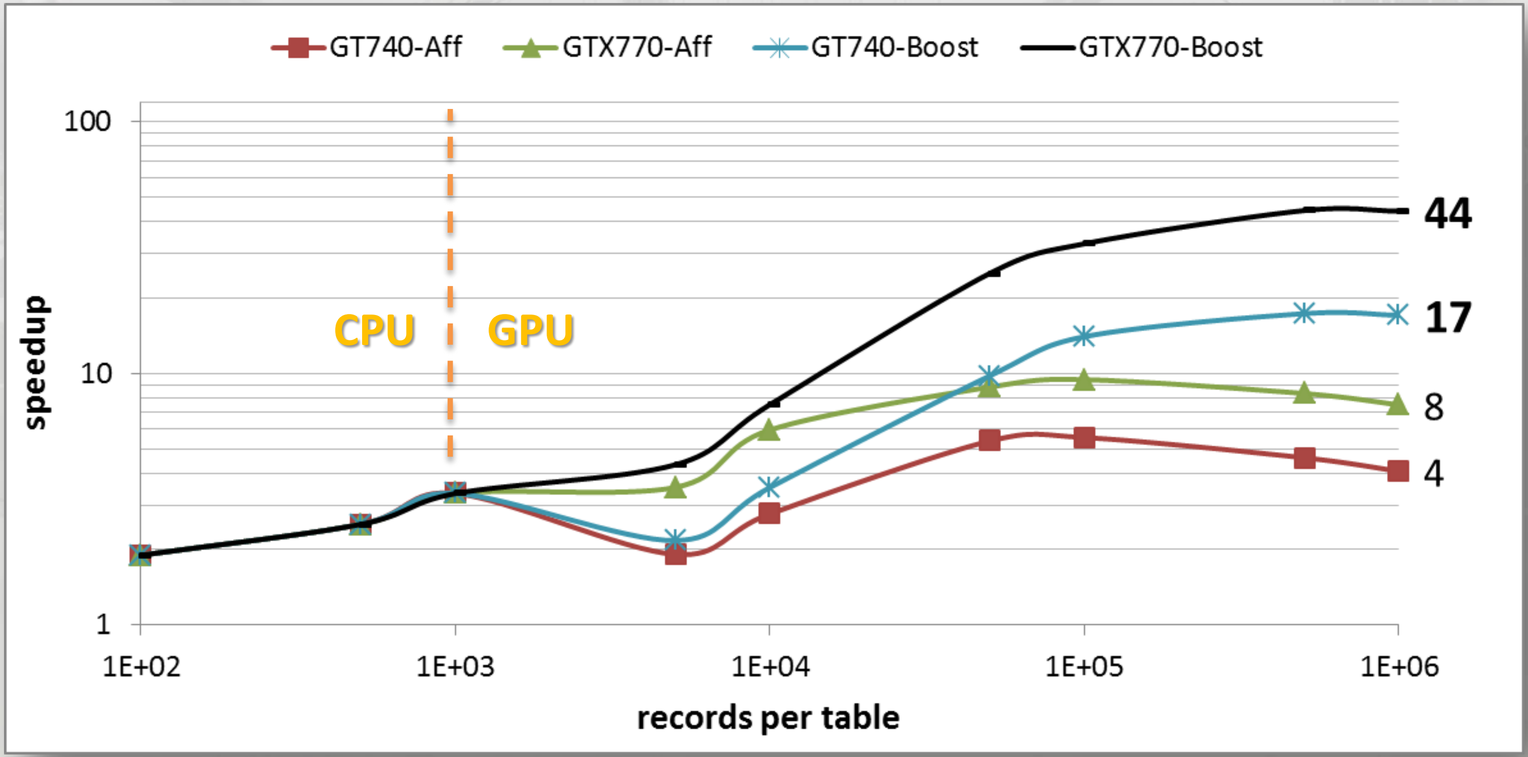
Experimental results: SELECT WHERE Queries



Average speedups with SELECT WHERE Queries

Peak speedup of 411x with: SELECT * WHERE col LIKE '%susbstring%'

Experimental results: SELECT JOIN Queries

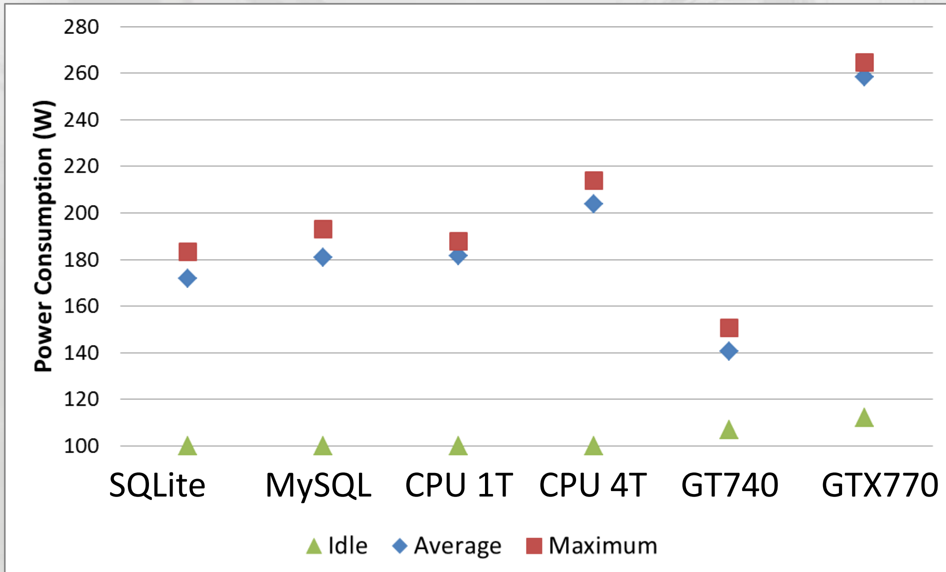


Average speedups with SELECT JOIN Queries

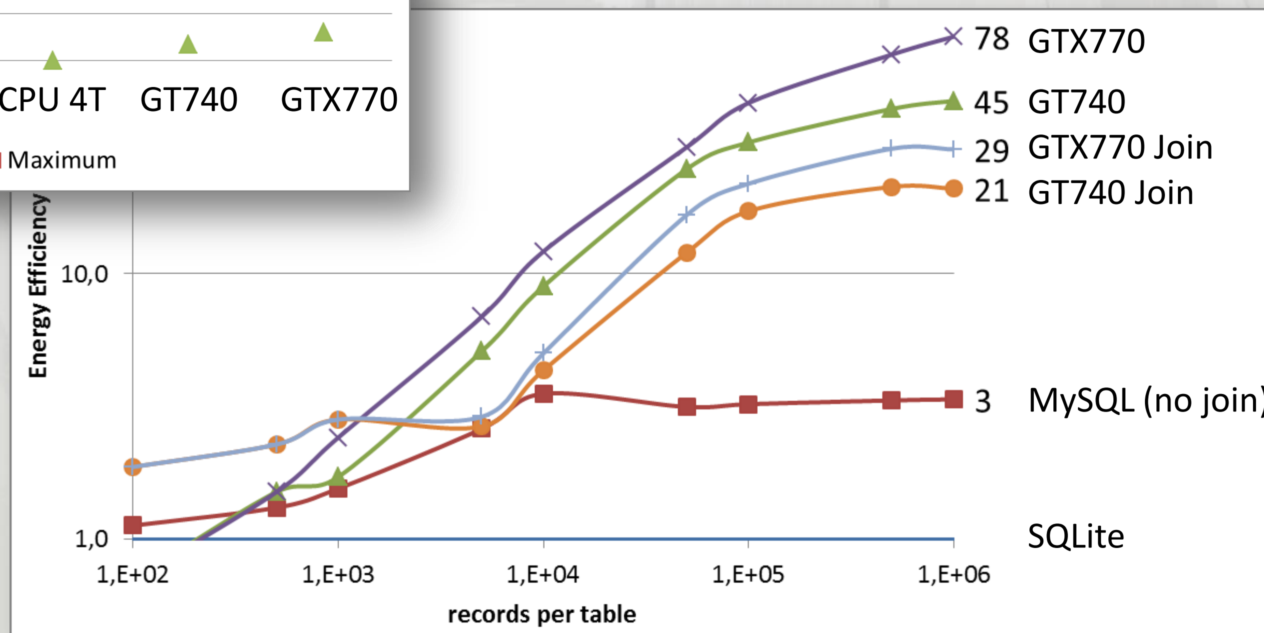
SQLite and CuDB build transient indexes, MySQL does not

Peak speedup of 66x with self-join queries

Experimental results: Energy Efficiency



CPU (X)T = CPU engine of CuDB with (X) threads



Conclusion and Future Works

- **Great speedups for full table scans**
- **Better energy efficiency**
- **We plan to:**
  - overcome the limitations of the GPU memory capacity
  - add full indexation mechanisms
  - improve SQL support -> TPC-H and SSB

Thank You !

Any questions ?