Parallelism in Database Operations

Kalle Kärkkäinen

Introduction

- The problem with databases
  - During queries the database can waste at least half it's time STALLING
    - 20% of this is due to branch mispredictions, rest is mainly due to cache misses
  - There are clear signs that we can do better! (Ailamaki et al.)

Introduction

- Why do databases work so slow?
  - Queries are complex
    - Selection on volatile criteria
  - Data amounts are larger

But mainly: because of the way we code

Code? What are we doing wrong?

- It all boils down to the CPU
- Modern CPU's process instructions in steps
  - For example there might be 5 steps to process an instruction in a RISC processor
- Each of these steps takes a cycle
  - A CPU might have 1.4 billion cycles to spare each second

CPU

- Instruction level parallelism
  – Instruction pipelining
- Branching
  – IF … THEN … ELSE …
  – Which branch should the pipeline take?
  – Guesstimation based on past

What happens on a branch miss prediction?

- The pipeline is dropped, and new pipeline is created for the new direction
  - COSTLY
- How does this affect databases?
  - Since they rely on external data, and queries may use specific limits…
    - It becomes impossible to predict the correct branch

Avoid the problem

- Design algorithms to not use branches
- Use operations that don’t use branches

- In a nut shell : DON’T USE BRANCHES, if mispredictions cost too much.

Introduction

- Flynn's taxonomy

	Single data	Multiple data
Single instruction	SISD	SIMD
Multiple instruction	MISD	MIMD

Normal sequential style:

```
for( x in RECORDS)
   if( condition (x) )
        process1 (x);
   else
        process2 (x);
```

For all the records, we test a condition, and then process it according to the test result.

SIMD version

```
for( x in R ; step S)
    mask = SIMD_cond (R[x…S]);
    SIMD_process(mask, R[x…S]);
```

We process R in blocks of S, and we process every element without branches.

What is SIMD in GPU/CPU

- GPU
  - Process a dataset with a kernel
- CPU
  - Process contents of wide registers with this operation

GPU

- Data is loaded into GPU memory and processed with a kernel program
  - Kernel programs are compiled into GPU compatible applications during runtime or before hand (implementation specific)
- The data is processed with multiple cores in parallel

- Allows for much larger datasets to be processed

CPU

- CPU SIMD is about changing the small portion of the application to use a different algorithm
  - Process multiple items at same time

What are the gains?

- GPU: speed gains of x20 (Skadron&Bakkum)
- CPU: speed gains of x5 (Zhou&Ross)

- Hardware is very different in these tests, and thus we can not compare the numbers

Problems

- GPU: different design for large part of the program
  - Switching to kernels is not an easy change
- Memory copying issues
  - Data transferred to memory, results transferred back
- Specialized hardware
  - Not a general purpose solution
- Programming environments do not support everything that is expected (for example, branching, 32 bit integers and such)

Problems

- CPU: achievable gains are limited by register width