Data Processing on Modern Hardware

Jana Giceva

Exam preparation

Exam info

- Written exam
- Date: 5th August 11:00 – 12:30

- 90min → 90 points
- You need to pass the exam, to reclaim the bonus points from homework and project

- Exam is structured in two parts (theory and practice):
  - Theory (~ 1/3rd of the points)
    - multiple choice questions
    - short answer questions
  - Practice (~ 2/3rd of the points)
    - up to 4 problems related to homework assignments
    - practical (writing code/sketching) and explaining results

Sample theory questions

- **What does pipelining mean in modern CPUs?**
  - ❑ Page eviction for new allocations.
  - ❑ Overlapped instruction execution.
  - ❑ Cache hierarchy for loads.

- **What are non-temporal streaming stores meant for?**
  - ❑ Queuing writes until memory is attached again.
  - ❑ Putting data in all cache levels before write goes to memory.
  - ❑ Avoiding the cache for writes to memory.

- **Which one is *not* a type of cache miss?**
  - ❑ Compulsory miss.
  - ❑ Coherency miss.
  - ❑ Capacity miss.
  - ❑ Conflict miss.

Sample theory questions

- **Name** one advantage and one disadvantage for **Fully Associative** and **Direct-Mapped** caches.
  Explain why **Set-associative caches** are a compromise.

- Your program just accessed a **virtual address,** but the data is not in memory.
  Briefly explain how the **page fault** is handled by the system.

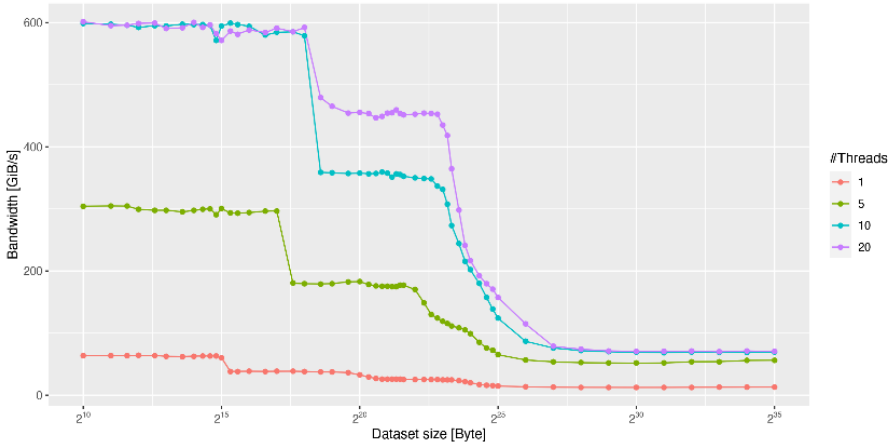
- **Name** one advantage and one disadvantage each for **column stores** and **row stores.**
  Which one is better for **OLAP workloads**. **Explain why.**

Practical questions

- May start with a short warm-up of theory questions.
- You may be given a code snippet and asked to optimize it with the technique in focus.
- You may be given alternatives to discuss.
- You may be asked to discuss the obtained results from experiments, explain the observed behavior and match it to the content of the lecture.

Cache-awareness

- The plot shows the **bandwidth** of **scan** reading a dataset. The x-axis shows the size of the **dataset** (from 2^10 to 2^30).



- **Questions:**
  - Why is the performance of the scan not constant?
  - What do you see and what is the cause?
  - Can you use it to infer properties of the machine? What are they?
  - Why do we observe the difference between the thread counts?
  - Is this a random read or sequential? How would it look like otherwise?

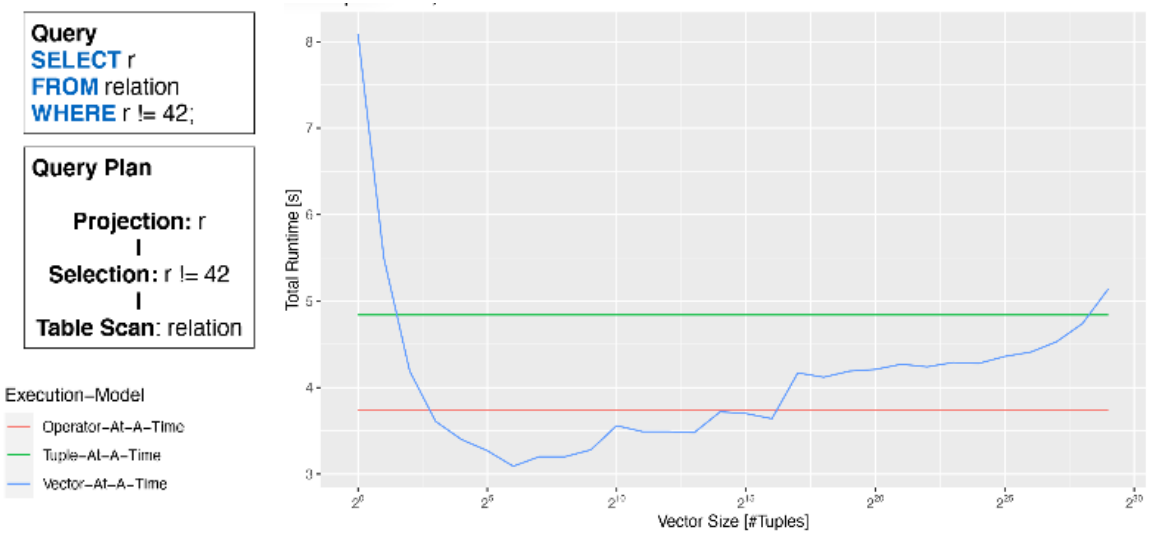
Execution Models

- The plot shows the runtime of a query for different execution models.
- We vary the vector size from 1 to 2^30 tuples for the vector-at-a-time model.

Query
SELECT r
FROM relation
WHERE r != 42;

Query Plan

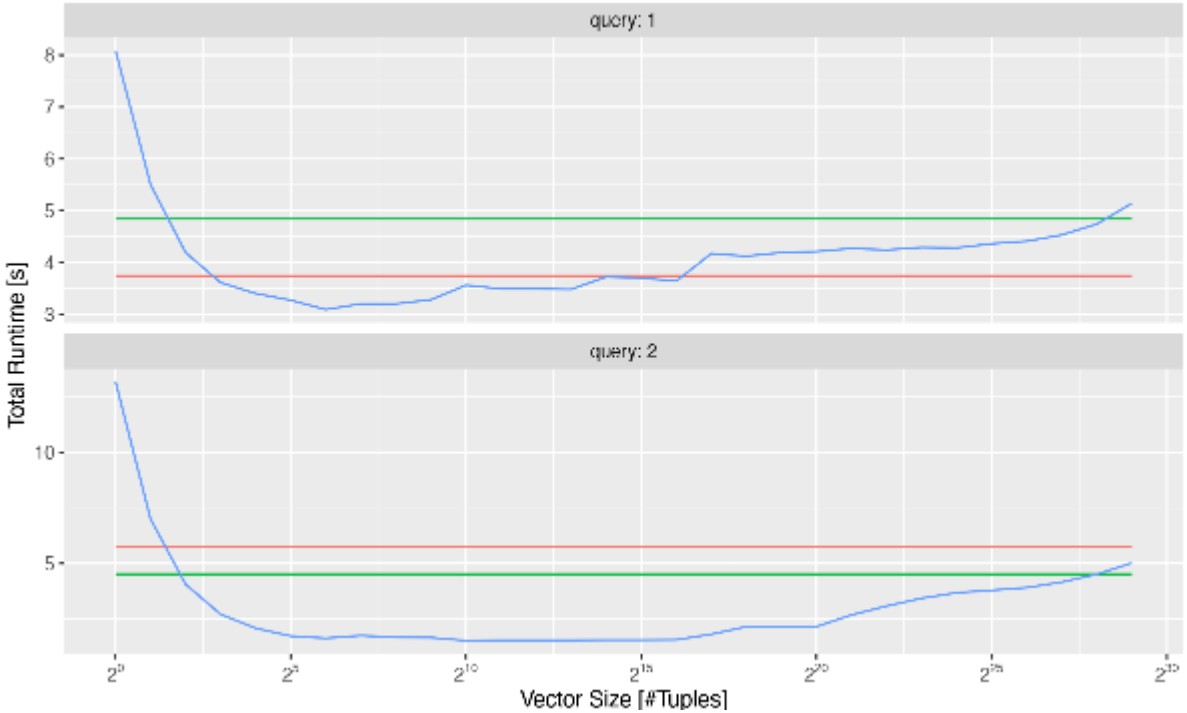
Projection: r
|
Selection: r != 42
|
Table Scan: relation



- **Questions:**
  - Describe the characteristics of the different execution models.
  - Explain the curve of the vector-at-a-time model in relation to the vector size.
  - Why is the operator-at-a-time faster than tuple-at-a-time for queries like this?
  - For which queries do you expect the reverse behavior?

Execution models cont.

- Why do we observe a different behavior for the two queries? Which model does each of the lines follow?



Query 1 - Query Plan

Projection: r
|
Selection: r != 42
|
Table Scan: relation

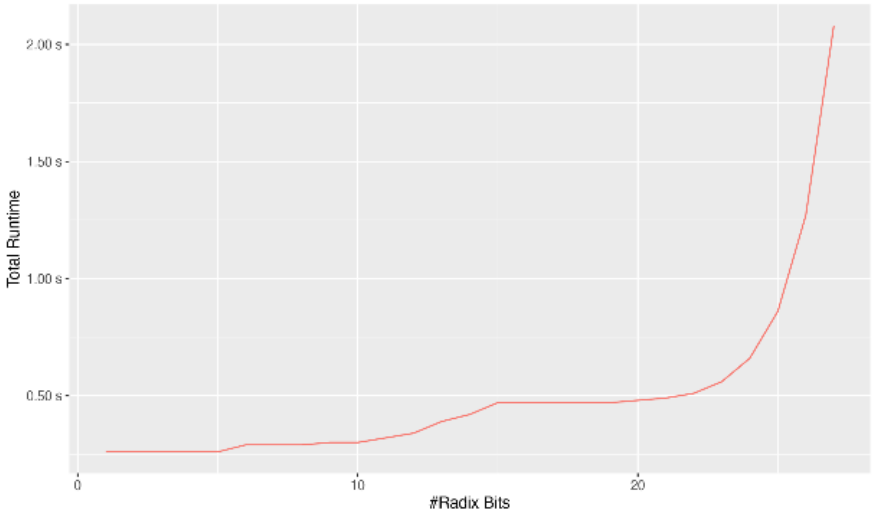
Query 2 - Query Plan

Aggregation: sum(r)
|
Selection: r != 43
|
Selection: r != 25
|
Selection: r != 12
|
Table Scan: relation

Radix partitioning

- The plot shows the **total runtime** of a **naïve radix partitioning.**
- We vary the number of radix bits (x-axis)

Machine
- 10 Cores (20 Threads)
  - L1d: 10x 2^{15} B
  - L1i: 10x 2^{15} B
  - L2: 10x 2^{20} B
  - L3: $2^{23.3}$ B
- RAM: 64 GB
- Pagesize: 4 KB
- L1 TLB: 64 Entries
- L2 TLB: 1536 Entries



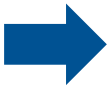
- **Questions**
  - What causes the step-wise decrease in performance?
  - What is the idea of **two-pass partitioning** and how can it help?
  - How can we further optimize performance?
  - Are there cases when radix partitioning is not beneficial? Elaborate.

SIMD

- In this task we look at the code for the following SQL-query and its C++ code.
- **Optimize** the code by:
  - making it **branch-free**
  - Transferring it to **SIMD (AVX-512)**
    - Describe the necessary steps to execute it using SIMD (load, apply mask, etc.)

Query

```
SELECT sum(r)
FROM relation
WHERE r < 42;
```

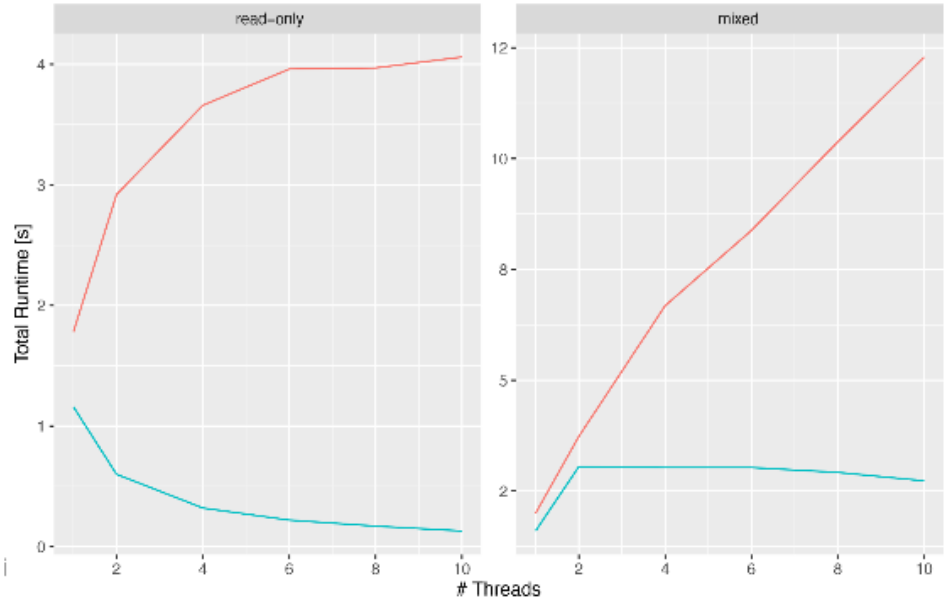


```
int64_t sum(int64_t *relation, unsigned relSize, int64_t predicate) {
  int64_t sum = 0;
  for (auto i = 0; i < relSize; i++)
    if (relation[i] < predicate)
      sum += relation[i];

  return sum;
}
```

Synchronization

- The **list** contains values from the range of 0 to 63, and we apply different **synchronization mechanisms:** red is **coarse RW,** blue is **Opt. Lock Coupling.**
- We measure the **total runtime** for two **workloads**: (1) **read-only:** contains, (2) **mixed**: contains, insert, remove.

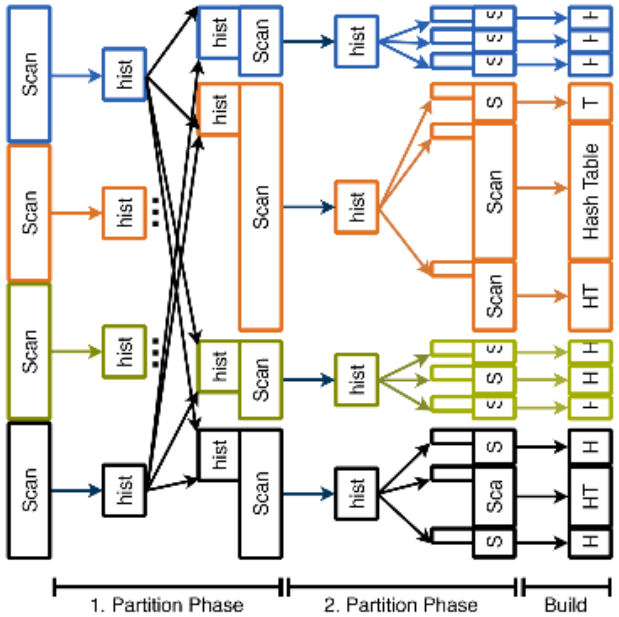


- **Questions:**
  - Describe the locking mechanisms shown in the plot. Which types of locks do they use?
  - What are their advantages and disadvantages?
  - Why does Coarse RW slow down linearly with higher core count on mixed?
  - Optimistic lock coupling scales very good for read-only. Why not for the mixed workloads?

Thread-based Parallelism

- **Task:** build side of **parallel radix-join** with **multi-pass partitioning.**
- We apply **thread-based parallelism.** Four hardware threads, each denoted with different color.
- Relation is **statically split** into 4 chunks, **one for each thread.**

- **Questions:**
  - We applied static work partitioning here. Which problem can this cause and how can we solve it?
  - Describe how task-based parallelism would work here?
  - Where would you add barriers?
  - What do you need to change to make the task-based parallel implementation NUMA-aware?



Chiplets

- What are chiplets?
- One of the hidden effects of chiplets is the partitioned LLC cache within a NUMA region. Explain what that means.
- Set-up: a student initiates 8 threads on the machine in case a) all of them run on chiplet 0, and in case b) they are distributed across all 8 chiplets. The threads work together on a joint dataset (e.g., populate data in a hash-table). The student measures the achieved bandwidth when varying the dataset size.
- Task: draw the bandwidth graph for both set-ups. How do you explain the behavior with respect to the chiplet architecture design of this machine.

How does it look like in an exam form?