Technical University of Munich

Tutorial 5: Data layouts, performance and roofline

Informatik elective: GPU Computing

Pratik Nayak

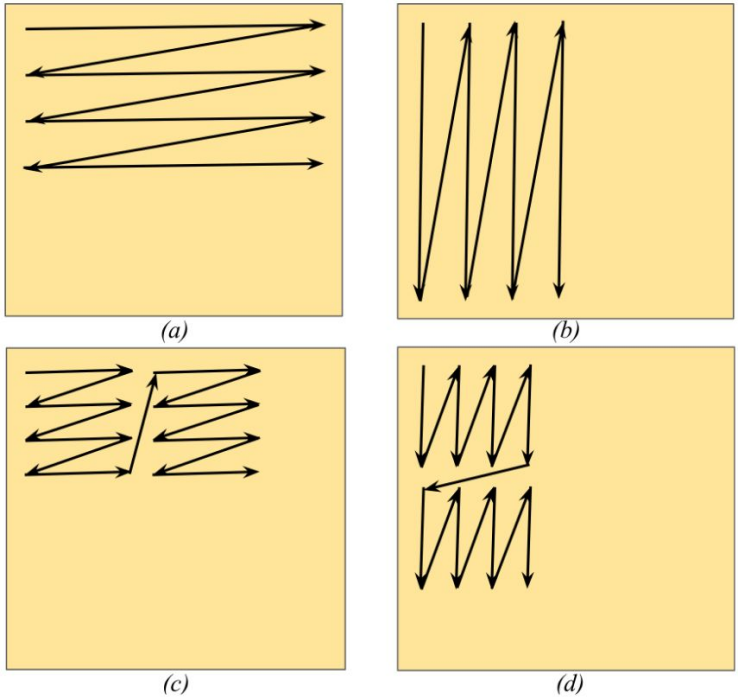
Data layouts

- Data layout defines how your data is laid out in memory.
- Structured data layouts enable efficient access and minimize cache misses.
- Some common data layouts:

(a) Row major: Elements of one row are stored consecutively

(b) Column major: Elements of one column are stored consecutively

(c) and (d) Tiled: Store tiles of size (t x t) consecutively



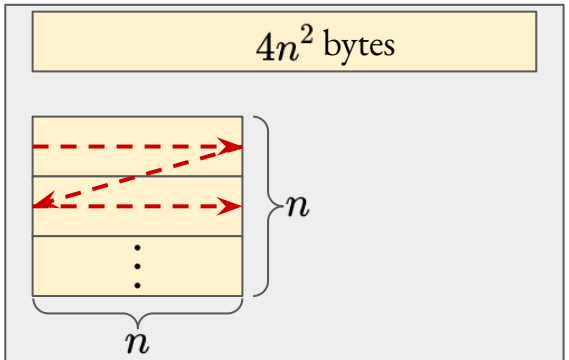
Row and column-major examples

- Consider a matrix, $M \in \mathbb{R}^{n \times n}$:

Row major :

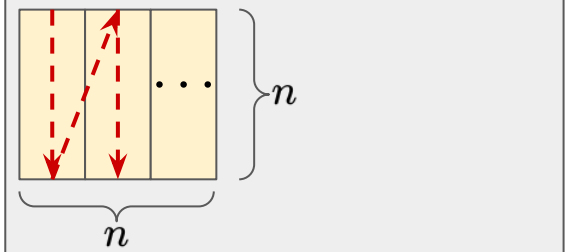
```
std::vector<float> mat(n*n)

for(each row){
  for(each col){
        mat(row, col) = val;
  }
}
```



Column major :

```
for(each col){
  for(each row){
        mat(row, col) = val;
  }
}
```



Contiguous and non-contiguous data-layouts

- Consider a matrix, $M \in \mathbb{R}^{n \times n}$

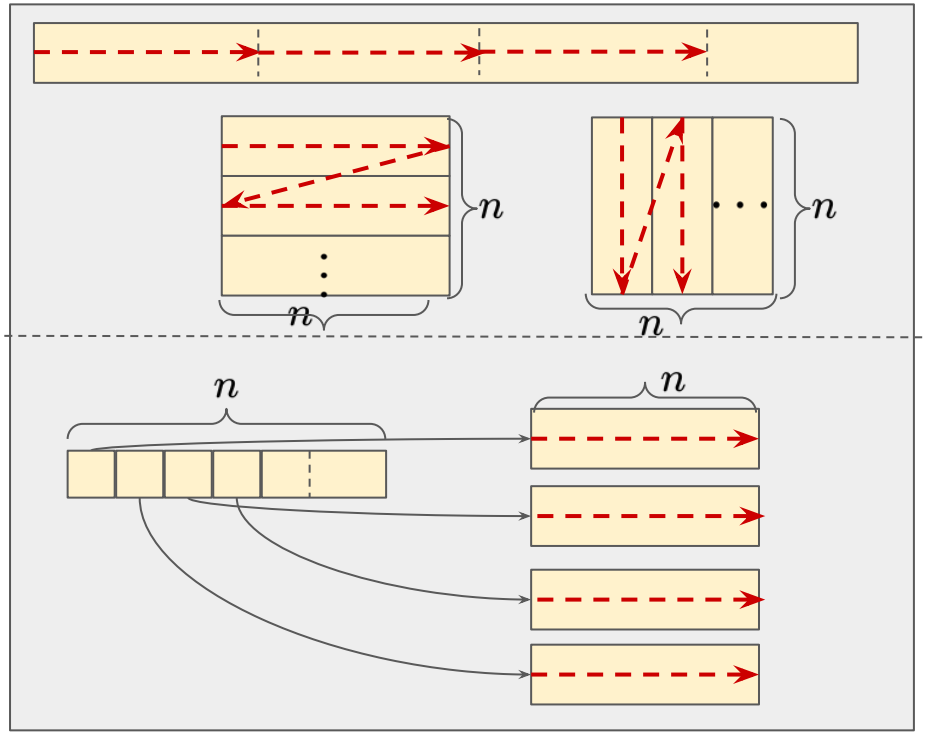
Contiguous :

```
vector<float> mat(n*n)
```

Non-contiguous :

```
vector<vector<float>> mat(n,vector<float>(n))
```

Note the indirection in access!



Measuring performance (best practices)

- Compilation flags are appropriate: For example with all the required optimization flags and no debug flags.
- Eliminate noise by repeating measurement a few times and averaging the overall time.
- Ensure operation is complete (synchronize explicitly if necessary).
- Ensure clock used is steady and cannot be arbitrarily updated (system_clock v/s steady_clock).
- Ensure operation is actually being performed (not skipped due to predication, or due to some clause).
- Try to eliminate common sources of interference: other users, some unrelated I/O, system effects.

Highly recommended read: https://blogs.fau.de/hager/archives/category/fooling-the-masses

Roofline: Exercise 1 solutions

- See https://gitlab.lrz.de/2024ws-gpu-computing/exercises/-/tree/master/ex1 for a typical solution.
- Performance can always be improved, but the general objective of this exercise was to get familiar with calculating peak FLOP and BW of your machine, calculating Arithmetic intensity and implementing some basic operations and classifying them into Compute bound/Memory bound operations.