Christian Dominic Angus

6197873

1. Why is it difficult to parallelize “particle system” that we discussed in the lectures.

Processors create more particles than others therefore there is an imbalance in the system, because of this some processors in the system operate more than processors that do not have as many particles. That is one of the reasons why parallelizing “particle systems” is difficult.

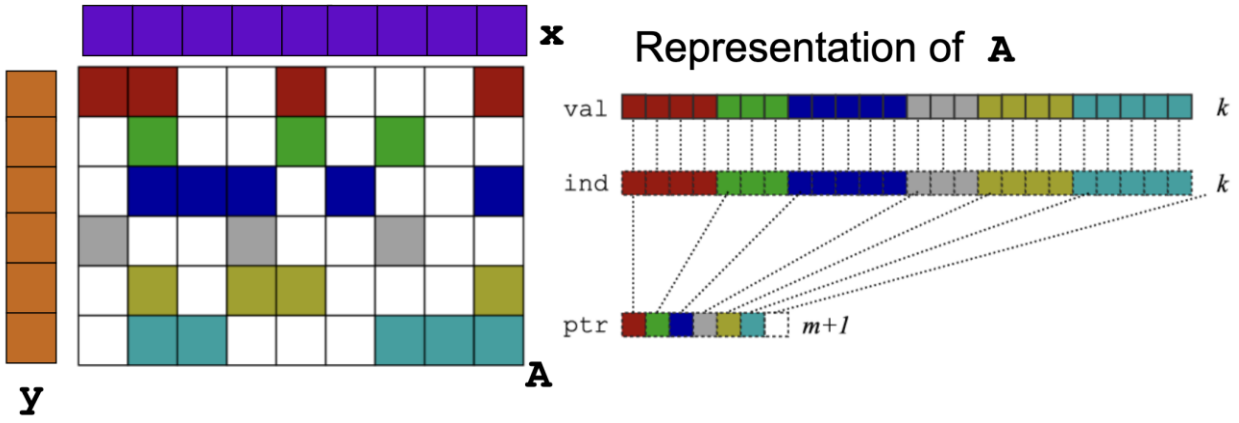
2. What is meant by low surface-to-volume ratio in simulating large particle systems? Why is it important for scalability?

Low surface-to-volume means there is a minimum to small number of edges attached to each system vertices. It is important for scalability because the more data and vertices there are in the system it is important that there are multiple pathways available for the data to traverse for it to complete the task. If it were to run into a pathway issue, then it is bottlenecked by the number of edges it can traverse, therefore the performance of the system is slower compared to ones with high surface-to-volume ratio.

3. What is Compressed Sparse Row (CSR) Format? Explain and illustrate with an example. Why is it a good idea to represent Matrix in this format?

It is a data structure for a sparse graph represented as a matrix by three 1D arrays, which only store arithmetic on nonzero entries. Using CSR is more space efficient than using a 2D array since it does not store any nonzero entries and represents data in a compact form.

In this illustration, you can see the space comparison between using 2D array and CSR:



4. What is meant by a race condition? Explain and illustrate with an example on a shared-memory system. Do race conditions operate in the same way on distributed-memory architectures, and shared-memory architectures? Discuss.

A race condition or data race happens when there are two processors or two threads that try to access the same variable and at last one does a write. The accesses are concurrent, meaning they are not synchronized, so it could happen simultaneously.

```
static int s = 0;
```

Thread 1

```
for i = 0, n/2-1
    s = s + f(A[i])
```

Thread 2

```
for i = n/2, n-1
    s = s + f(A[i])
```

Take the illustration above for example, if both threads were to access variable “s” at the same time the thread that finishes after will return an incorrect value because by the time the thread that finishes first would have changed the value of s.

Assume that A[] = [3, 5] and f(x) = x^2, depending on which thread is performed first the value of s could return as 9, 25 or 34.

Race conditions on distributed-memory architectures and shared-memory architecture do not operate in the same way. Distributed-memory and shared-memory architectures do not have race conditions since memories on each device is independent from each other and that data accessed from one node are not usually dependent of data from another.

Citation

Saeed, Fahad. "Lecture 3 (Part 1) PowerPoint (PPT)." Canvas, 2020. PowerPoint.

Saeed, Fahad. "Lecture 4 PowerPoint (PPT)." Canvas, 2020. PowerPoint.

Saeed, Fahad. "Lecture 5 PowerPoint (PPT)." Canvas, 2020. PowerPoint.