EUROfusion

Meeting of HPC ACHs

ERO2.0 Code: Enhancing Plasma-Wall Interaction Modeling

“A Journey into Parallelization with CUDA and OpenACC”

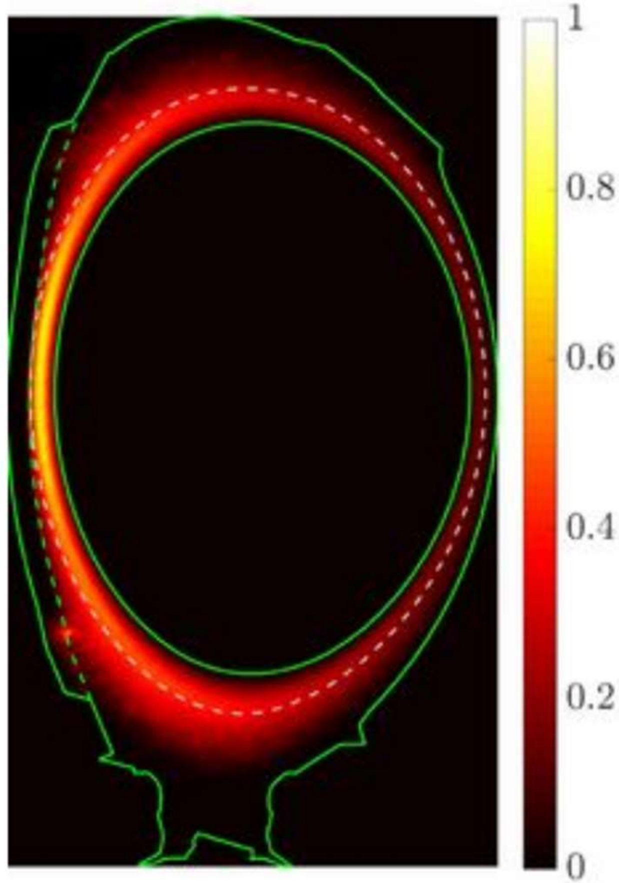
M. Augusto Maidana Silanes

HPC ACHs | 15-16 November 2023

This work has been carried out within the framework of the EUROfusion Consortium, funded by the European Union via the Euratom Research and Training Programme (Grant Agreement No 101052200 — EUROfusion). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Commission. Neither the European Union nor the European Commission can be held responsible for them.

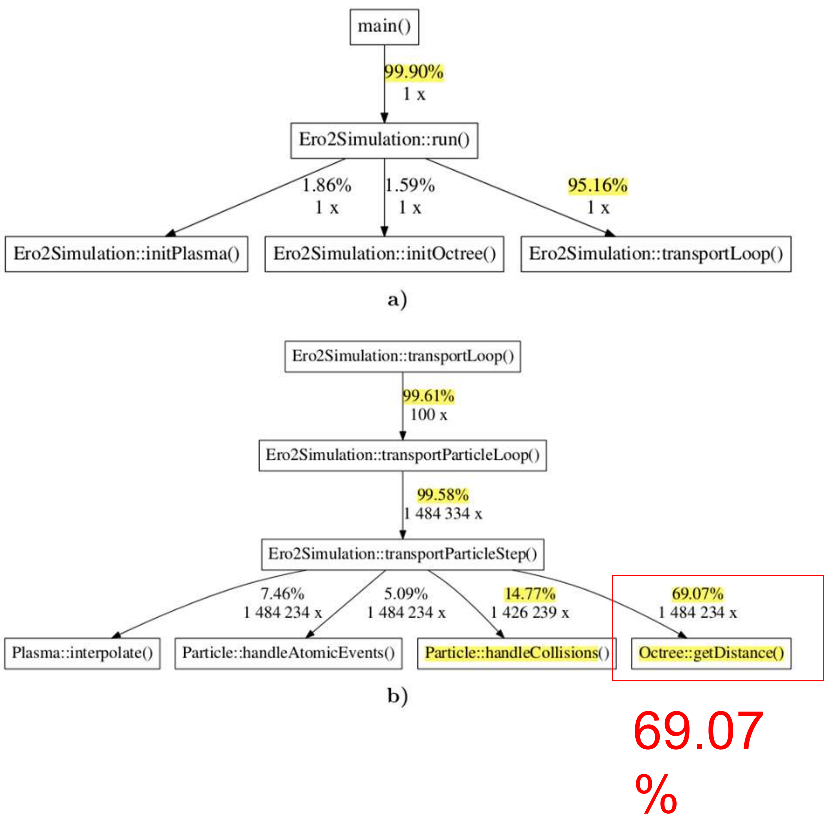
Introduction to ERO2.0

- Brief overview of **ERO2.0** model:
- **Plasma-wall interaction** and **global material migration** in fusion devices.
- Simulation of **3D trajectories** using **Monte-Carlo** test particles.
- Resolution of **3D gyro-orbits** without guiding-center approximation.
- Current parallelization using **MPI/OpenMP**.
- Goal: Porting to **GPU** for enhanced performance.

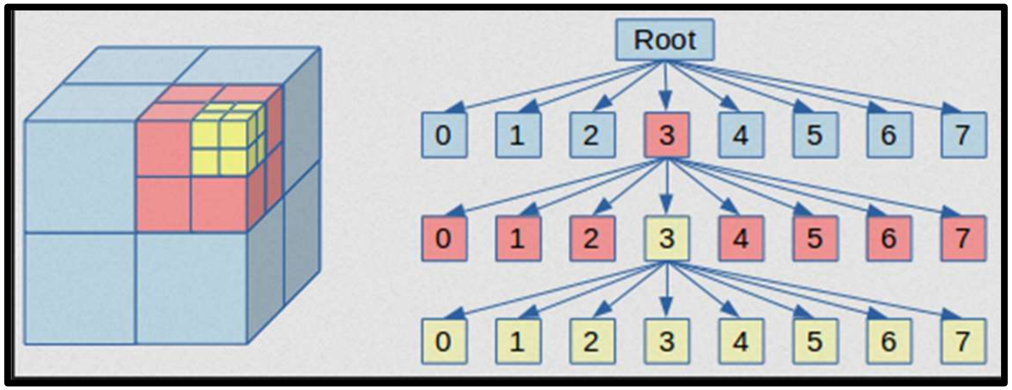


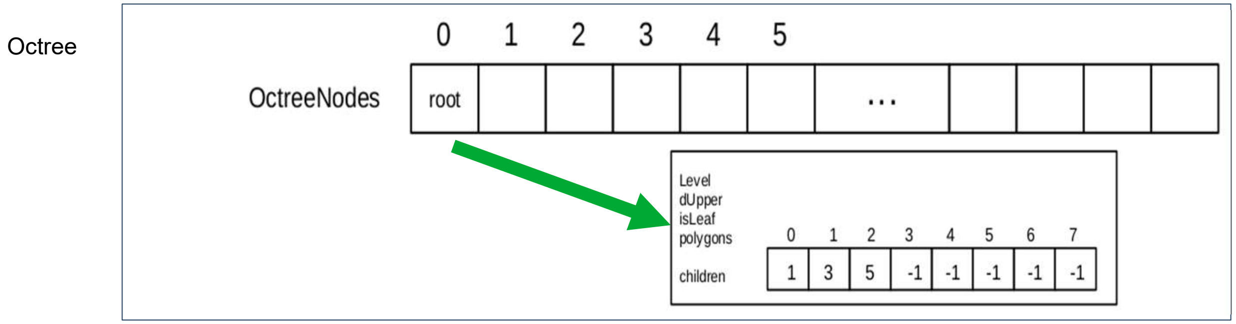
Barcelona Supercomputing Center
Centro Nacional de Supercomputación

- Challenges in **GPU implementation**:
  - **Hierarchical** and **recursive** nature of octree construction.
- Preparatory tasks for efficient **GPU execution**:
  - Translate recursive tree structure to a "**flattened**" **octree**.
  - Convert recursive octree search to an **iterative process**.
- Parallelization of queries using **CUDA**.



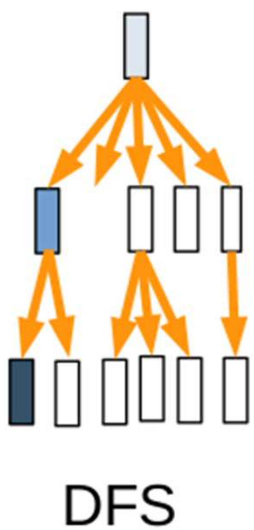
- Visual representation of the "**flattened**" octree structure.
- Explanation of the linear structure for efficient GPU traversal.



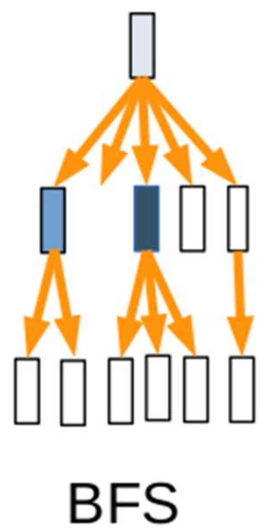


Octree Traversal Techniques

- Overview of tree traversal techniques:
  - Depth First Search (DFS) vs. Breadth First Search (BFS).
- Focus on current DFS and transition to BFS for GPU parallelization.



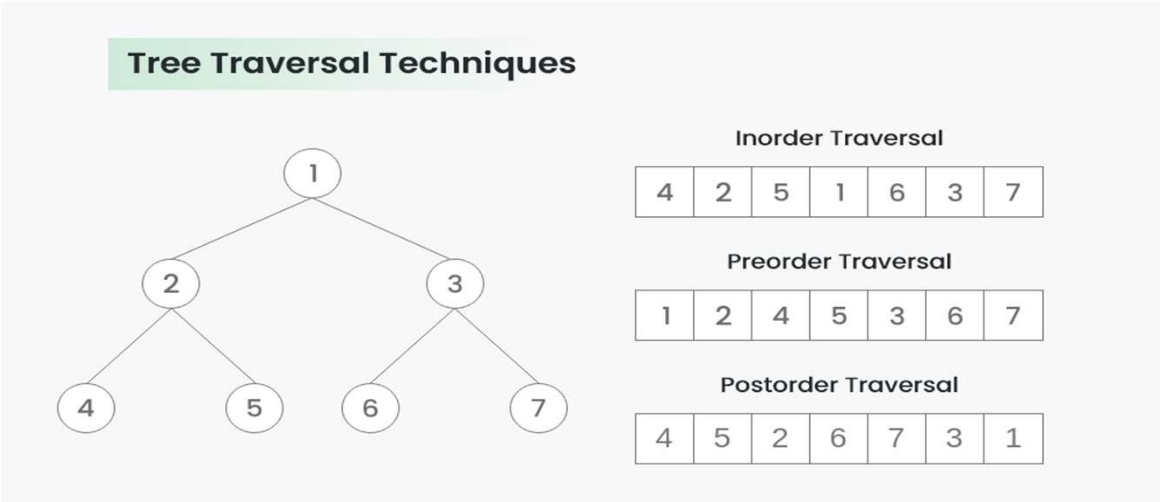
DFS algorithm starts at the root node (selecting some arbitrary node as the root node in the case of a graph) and explores as far as possible along each branch before backtracking.



Level Order Traversal (BFS) technique is defined as a method to traverse a Tree such that all nodes present in the same level are traversed completely before traversing the next level.

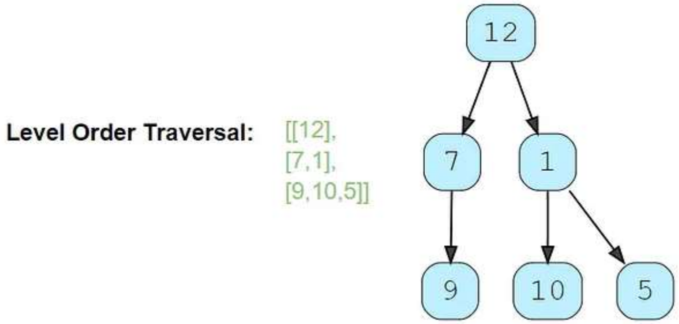
Depth First Search (DFS)

- Inorder Traversal:
  - Algorithm and uses in binary search trees.
  - Time Complexity: O(N).
  - Auxiliary Space: Considering the size of the stack for function calls then O(1) otherwise O(h) where h is the height of the tree.
- Preorder Traversal:
  - Algorithm and applications in creating a copy of the tree.
  - Time Complexity: O(N).
  - Auxiliary Space: Considering the size of the stack for function calls then O(1) otherwise O(h).
- Postorder Traversal:
  - Algorithm and applications in tree deletion.



Breadth First Search (BFS)

- Introduction to Level Order Traversal/BFS.
- Explanation of efficient queue-based approach.



1. Start by pushing the root node to the queue.
2. Keep iterating until the queue is empty.
3. In each iteration, first count the elements in the queue (let's call it levelSize). We will have these many nodes in the current level.
4. Next, remove levelSize nodes from the queue and push their values in an array to represent the current level.
5. After removing each node from the queue, insert both of its children(if exist) into the queue.
6. If the queue is not empty, repeat from step 3 for the next level.

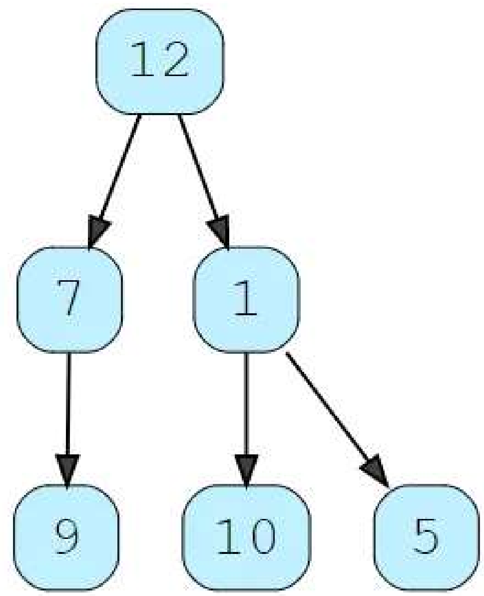
Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Breadth First Search (BFS)

Let’s take the example mentioned above to visually represent our algorithm:

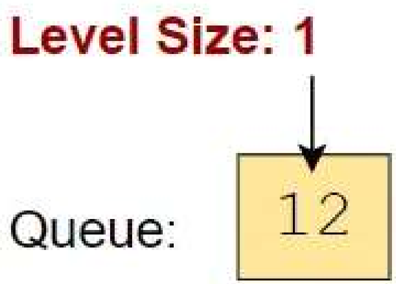
Level Size: 1

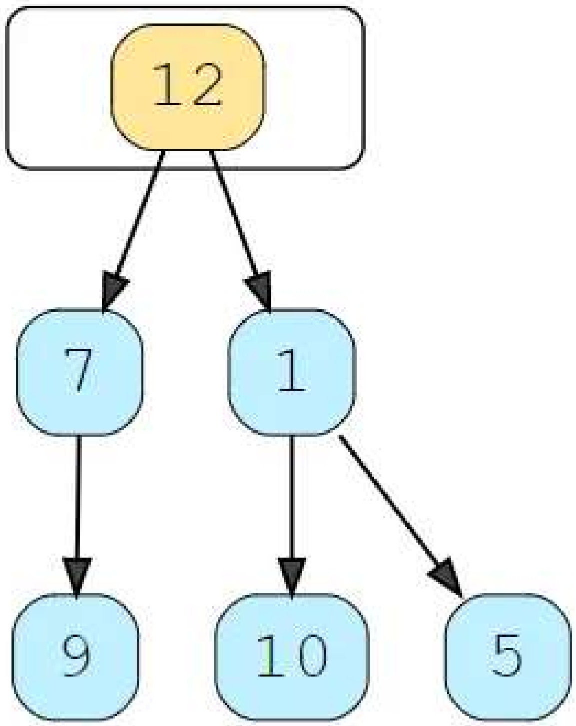
Queue: | 12 |



Start by pushing the root to the queue

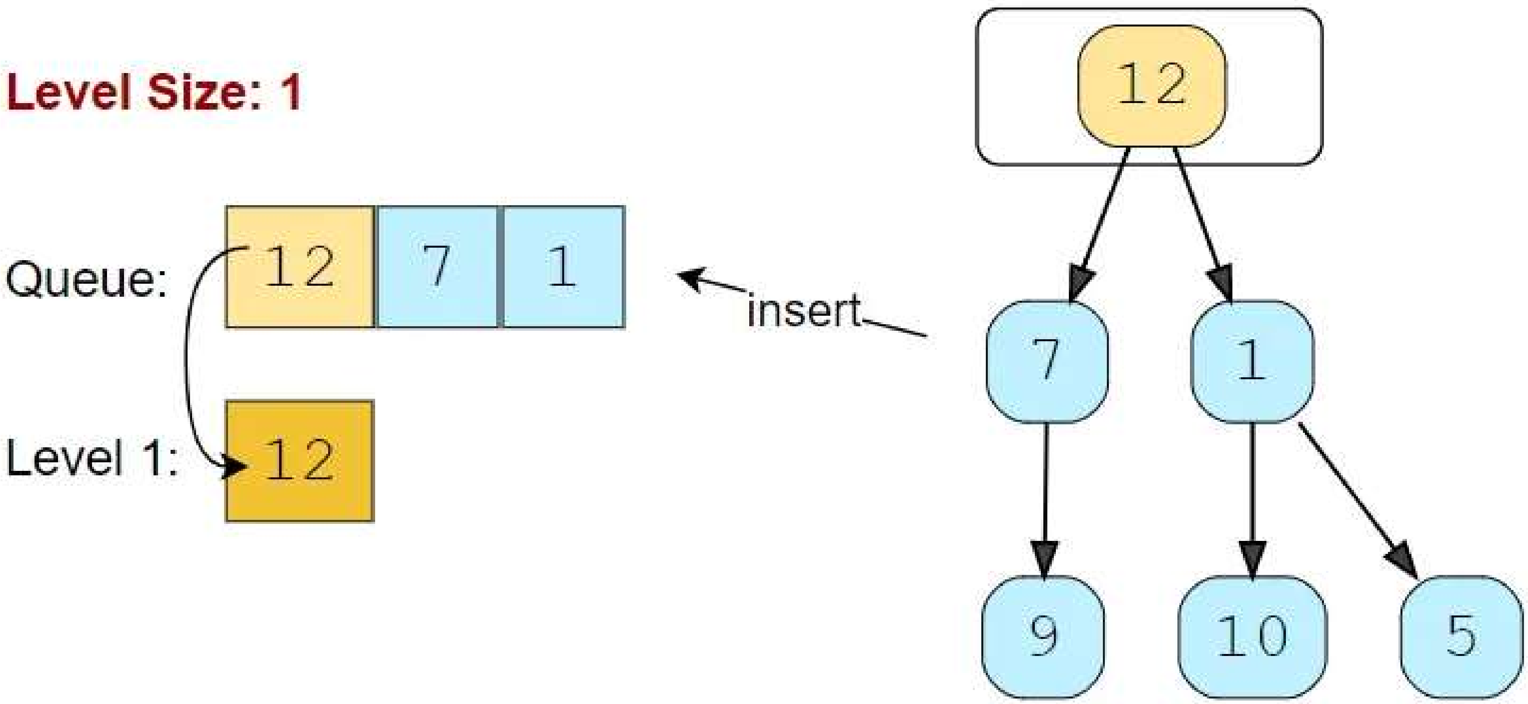
Breadth First Search (BFS)





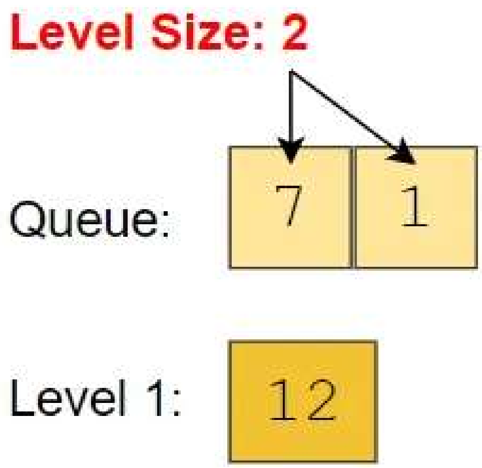
Count the elements of the queue (levelSize = 1), they all will be in the first level. Since the levelSize is “1” there will be one element in the first level.

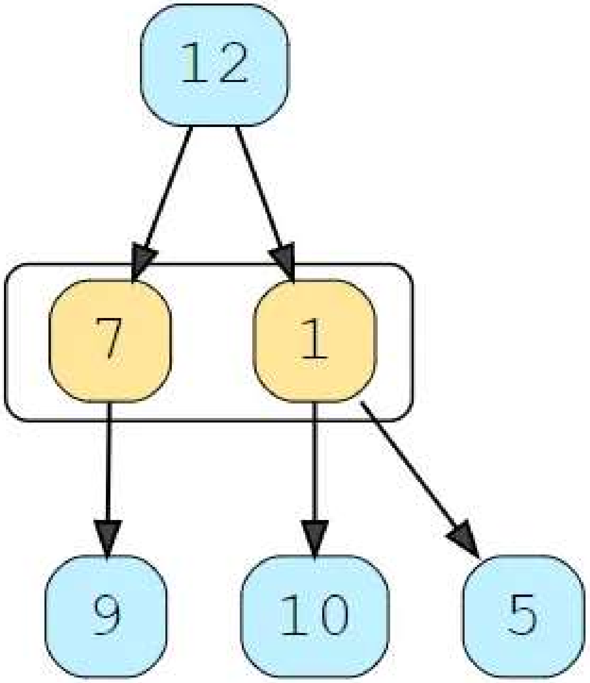
Breadth First Search (BFS)



Move "one" element to the the output array representing the first level and push its children to the queue.

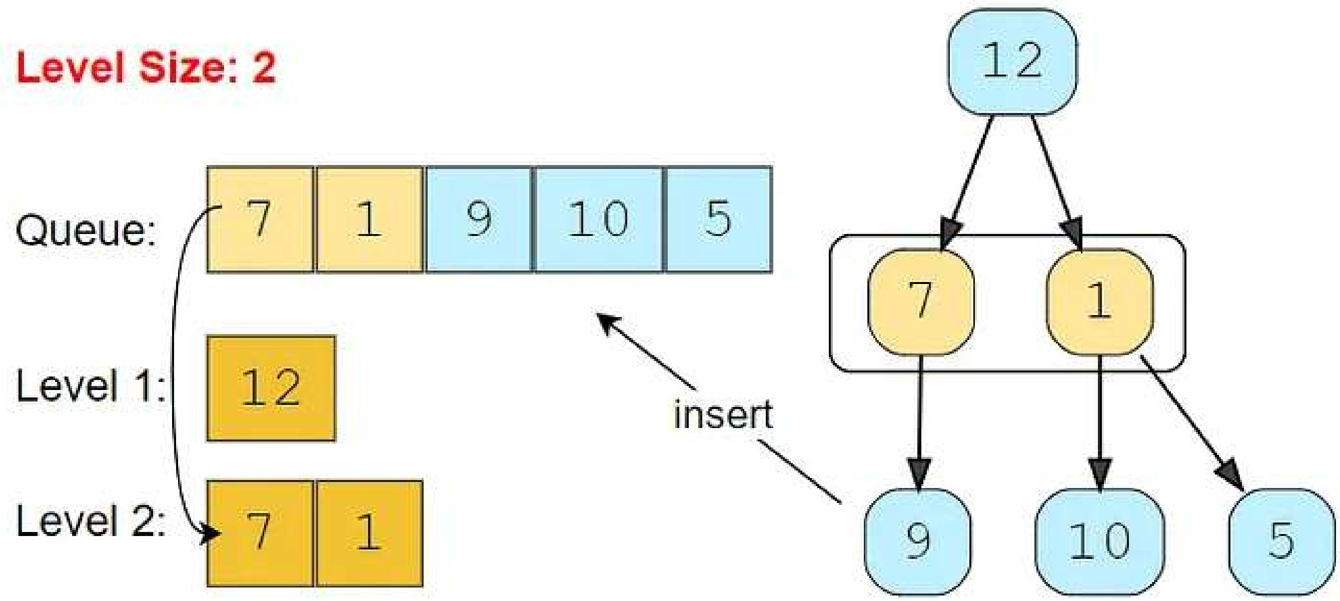
Breadth First Search (BFS)





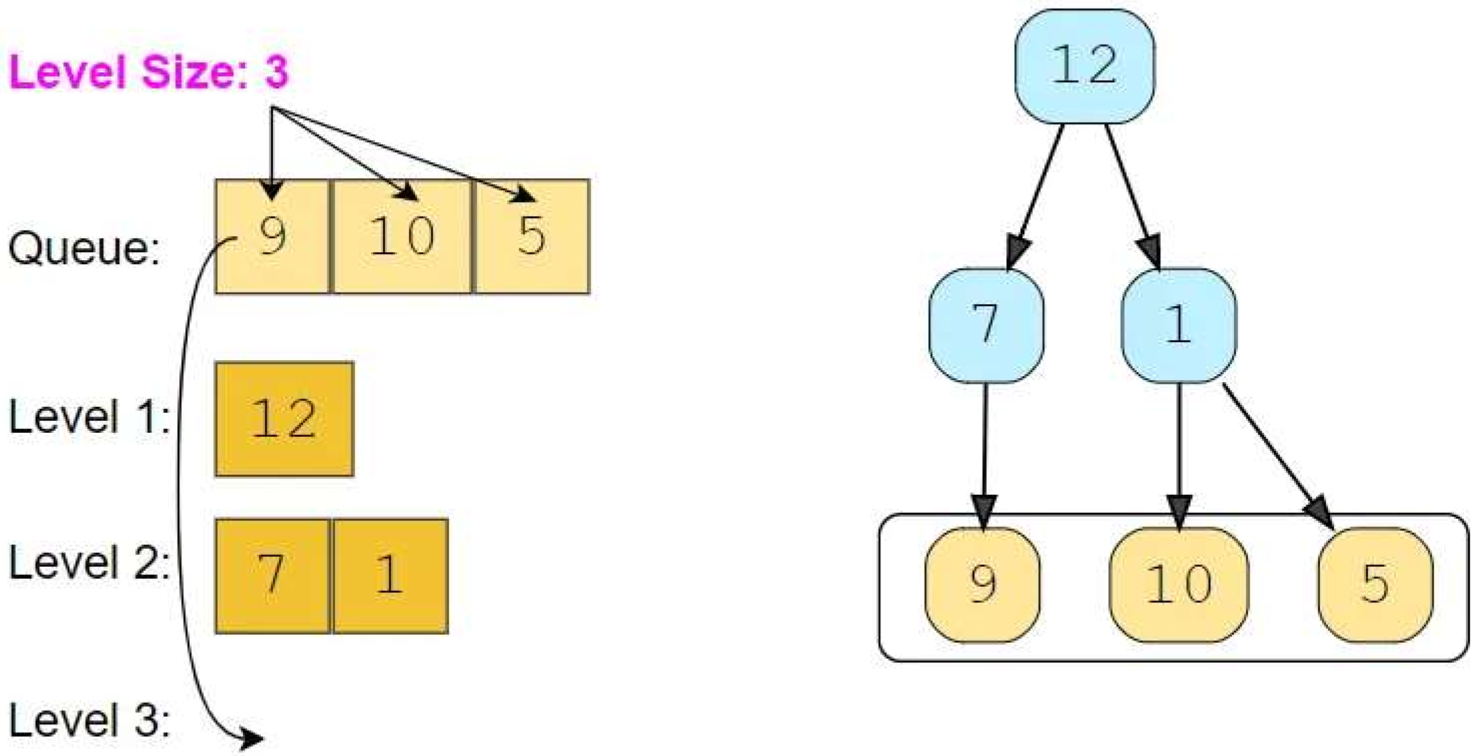
Count the elements of the queue (levelSize = 2), they all will be in the second level. Since the levelSize is "2" there will be two elements in the second level.

Breadth First Search (BFS)



Move “two” elements to the the output array representing the second level and push their children to the queue in the same order.

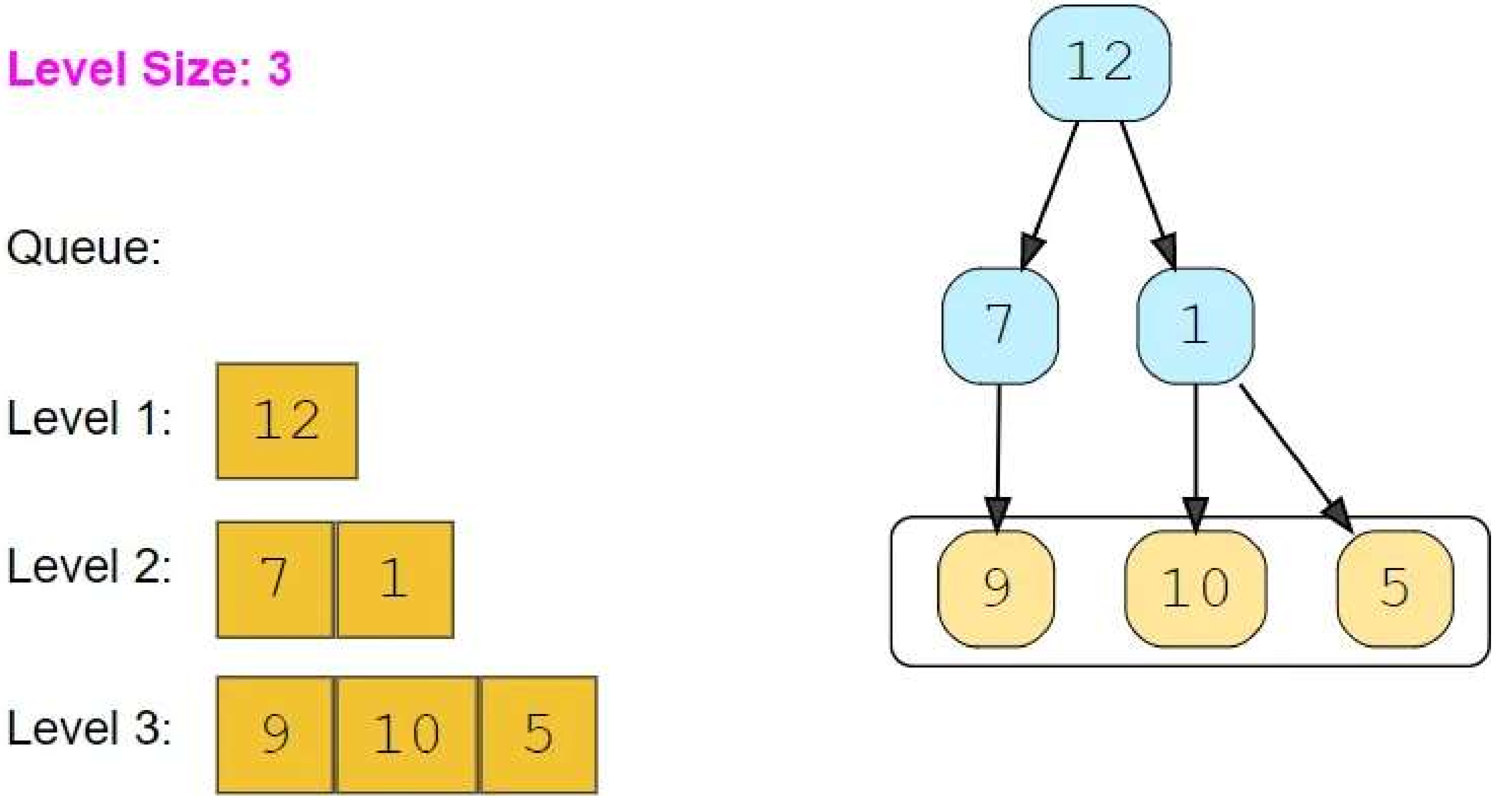
Breadth First Search (BFS)



Count the elements of the queue (levelSize = 3), they all will be in the third level. Since the levelSize is “3” there will be three elements in the third level.

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

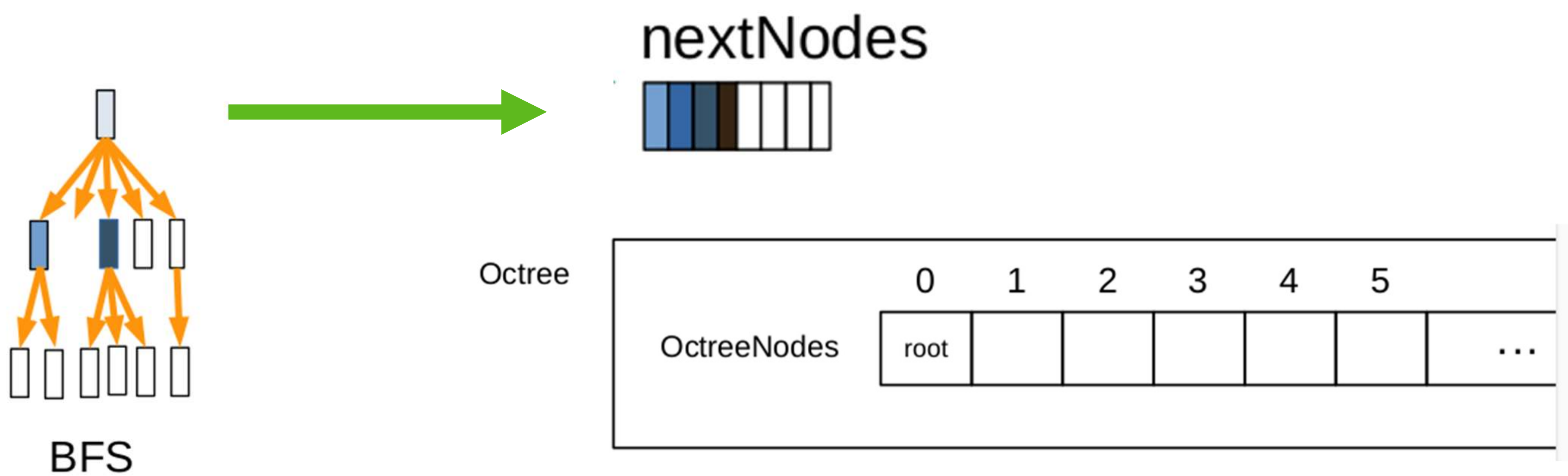
Breadth First Search (BFS)



Move “three” elements to the the output array representing third level.

Octree BFS for GPU

- Application of BFS to octree traversal on GPU.
- Two-array approach for efficient processing.
- Node exploration and pushing child nodes for further processing.



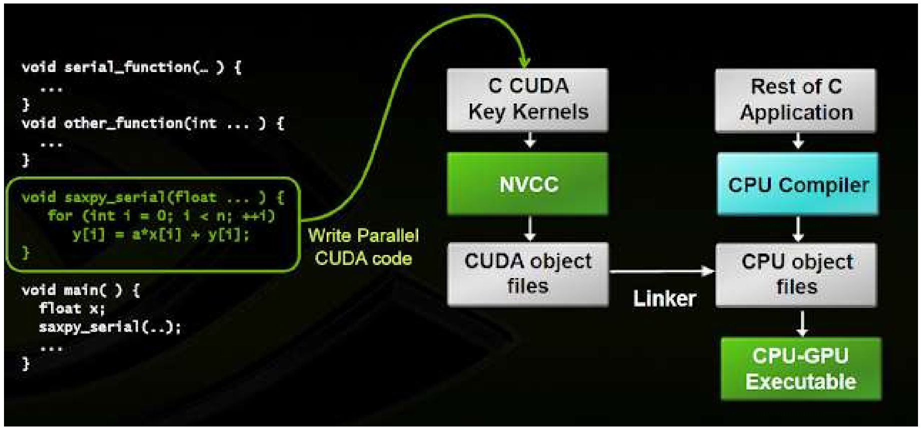
- Limitations of using std::vector in CUDA.
- Introduction of Thrust and its limitations in CUDA kernels.
- Solution: Using std::vector on CPU and manual conversion to arrays on GPU.

`#include <vector>`	
`std::vector<int> v {2, 4, 5};`	2 4 5
`v.push_back(6);`	2 4 5 6
`v.pop_back();`	2 4 5
`v[1] = 3;`	2 3 5
`v.resize(5, 0);`	2 3 5 0 0
`cout << v[2];`	prints 5
`for (int x : v)`	
`  cout << x << ' '`	prints 2 3 5 0 0

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

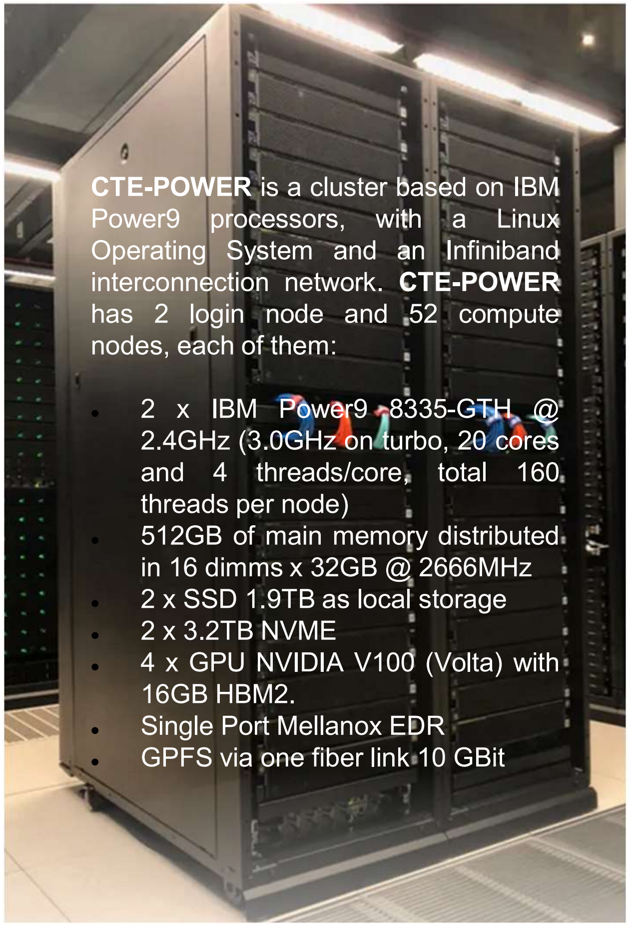
Key Points in CUDA Programming

- Overview of **CUDA** programming:
  - Memory management.
  - Kernel definition and launch.
  - Memory transfer.



Work Summary:

- Overview of key development steps:
  - Reading and searching for necessary **bibliography**.
  - Debugging and compiling on **Power9** with **openMP** and **MPI**.
  - Installation and configuration of **GPU** benchmarking tools.
  - Study and analysis of **ERO2.0** code.
  - Various development milestones, including **BFS** implementation.
- Recognition of the ERO2.0 **Gitlab** repository.



Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Next Steps:

- Outline of upcoming tasks:
  - Flatten the “polygon” vector.
  - Complete CUDA implementation and evaluate.
  - Assess the use of SoA for octree nodes.
  - Ongoing development to port **ERO2.0** code to GPU.

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Thank you

Any questions?

M. Augusto Maidana Silanes

Barcelona Supercomputing Center (BSC)

augusto.maidana@bsc.es

EUROfusion

Meeting of HPC ACHs

ERO2.0 Code: Enhancing Plasma-Wall Interaction Modeling

"A Journey into Parallelization with CUDA and OpenACC"

M. Augusto Maidana Silanes

HPC ACHs | 15-16 November 2023

This work has been carried out within the framework of the EUROfusion Consortium, funded by the European Union via the Euratom Research and Training Programme (Grant Agreement No 101052200 — EUROfusion). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Commission. Neither the European Union nor the European Commission can be held responsible for them.