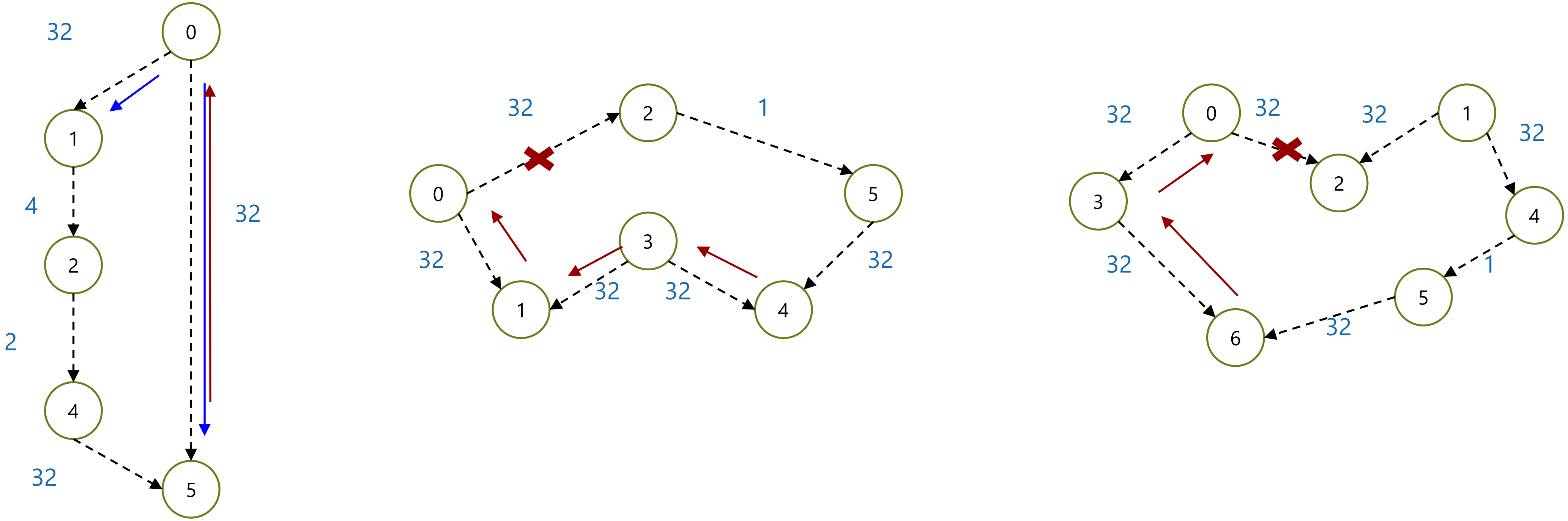
ASA: Scheduling

Buffer Space

What situations can trigger a deadlock?



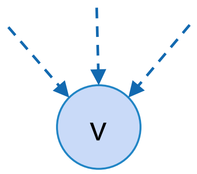
We have to look at **all the undirected cycles** inside a streaming component

We know that the streaming interval would allow us to not deadlock, but we want to avoid **bubbles** as well.

First Output

We indicate with FO (first-output) time, the time at which a node produces its first output element. If the node is not a barrier node:

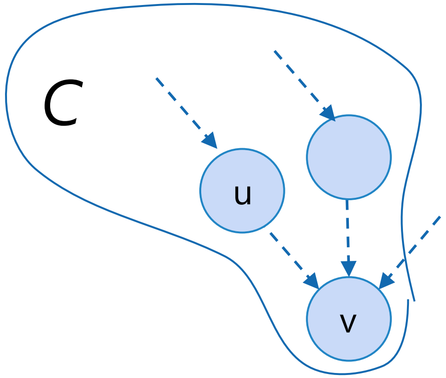
$$FO(v) = \max_{(u,v)\in E(G)} FO(u) + \begin{cases} \left(\frac{1}{R(v)} - 1\right) S^{+}(u) + 1 & \text{if } R(v) < 1 \\ 1 & \text{else.} \end{cases}$$



Otherwise:

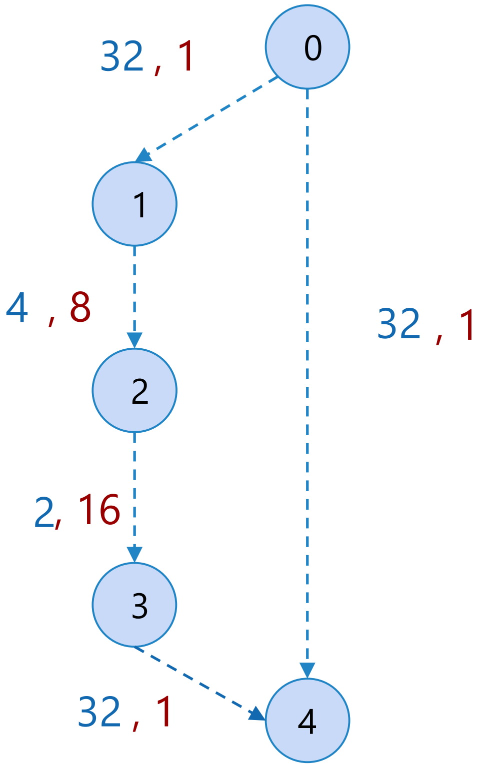
$$FO(v) = \max_{(u,v)\in E(G)} LO(u) + 1$$

Ideally, we want to check all nodes in an undirected cycle that have at least two predecessor (from that cycle), and evaluate the difference in the FOs



$$B(u,v) = \frac{\max_{(t,v)\in C} FO(t) - FO(u)}{S^{+}(v)}$$

Example



v	FO(v)
0	1
1	9
2	18
3	19

$$FO(v) = \max_{(u,v)\in E(G)} FO(u) + \begin{cases} \left(\frac{1}{R(v)} - 1\right) S^{+}(u) + 1 & \text{if } R(v) < 1 \\ 1 & \text{else.} \end{cases}$$

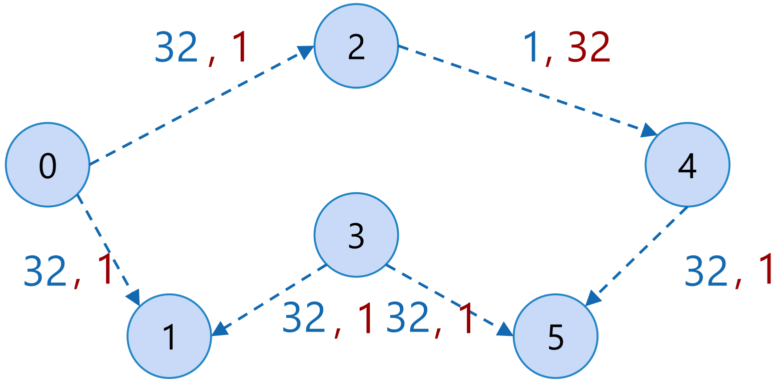
$$B(u,v) = \frac{\max_{(t,v)\in C} FO(t) - FO(u)}{S^{+}(v)}$$

In this case, the edge (0,4) must have buffer space B=18. **Note** that a buffer space=16 would have been sufficient to avoid deadlock, but not to avoid bubbles.

Example

$$FO(v) = \max_{(u,v)\in E(G)} FO(u) + \begin{cases} \left(\frac{1}{R(v)} - 1\right) S^{+}(u) + 1 & \text{if } R(v) < 1 \\ 1 & \text{else.} \end{cases}$$

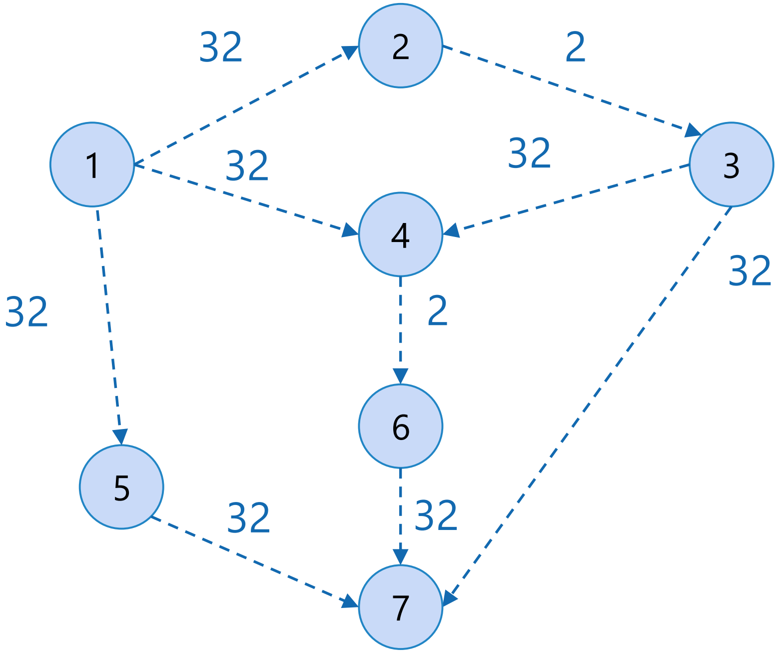
$$B(u,v) = \frac{\max_{(t,v)\in C} FO(t) - FO(u)}{S^{+}(v)}$$



v	*FO(v)*
0	1
1	2
2	33
3	1
4	33
5	34

The nodes with more than one input edge are 1 and 5. Buffer space for (4,5) is 32

Example



v	$FO(v)$
1	1
2	17
3	18
4	34
5	2
6	35

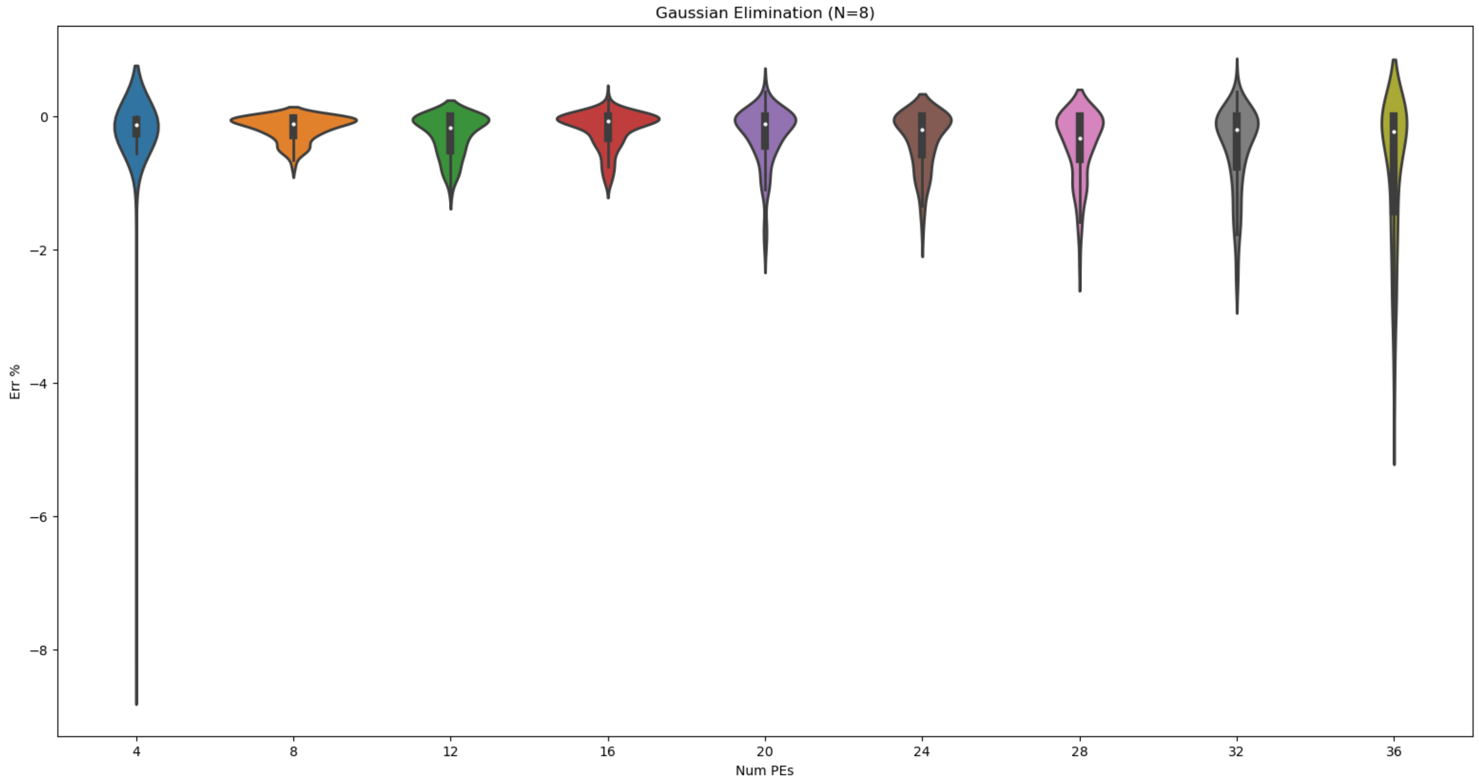
(u,v)	B
(1,4)	17
(5,7)	33
(3,7)	17

- Do we need to look at all possible cycles?

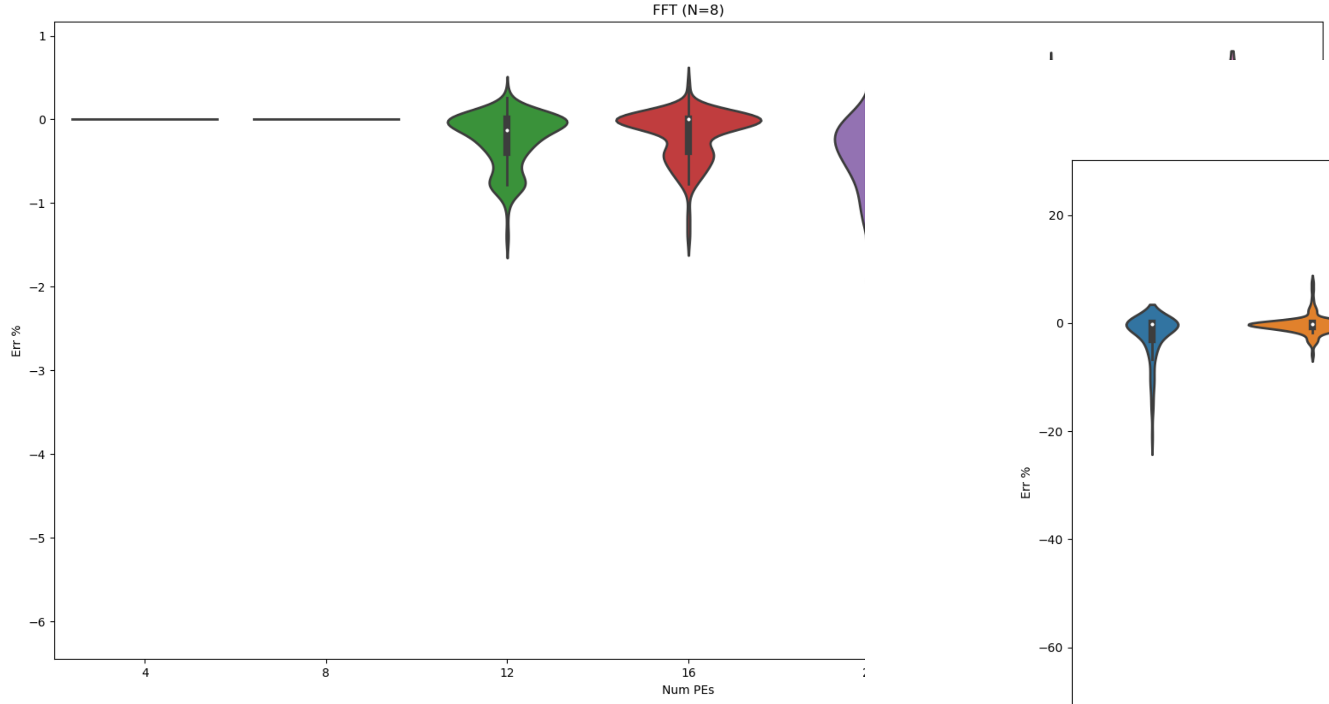
Evaluation

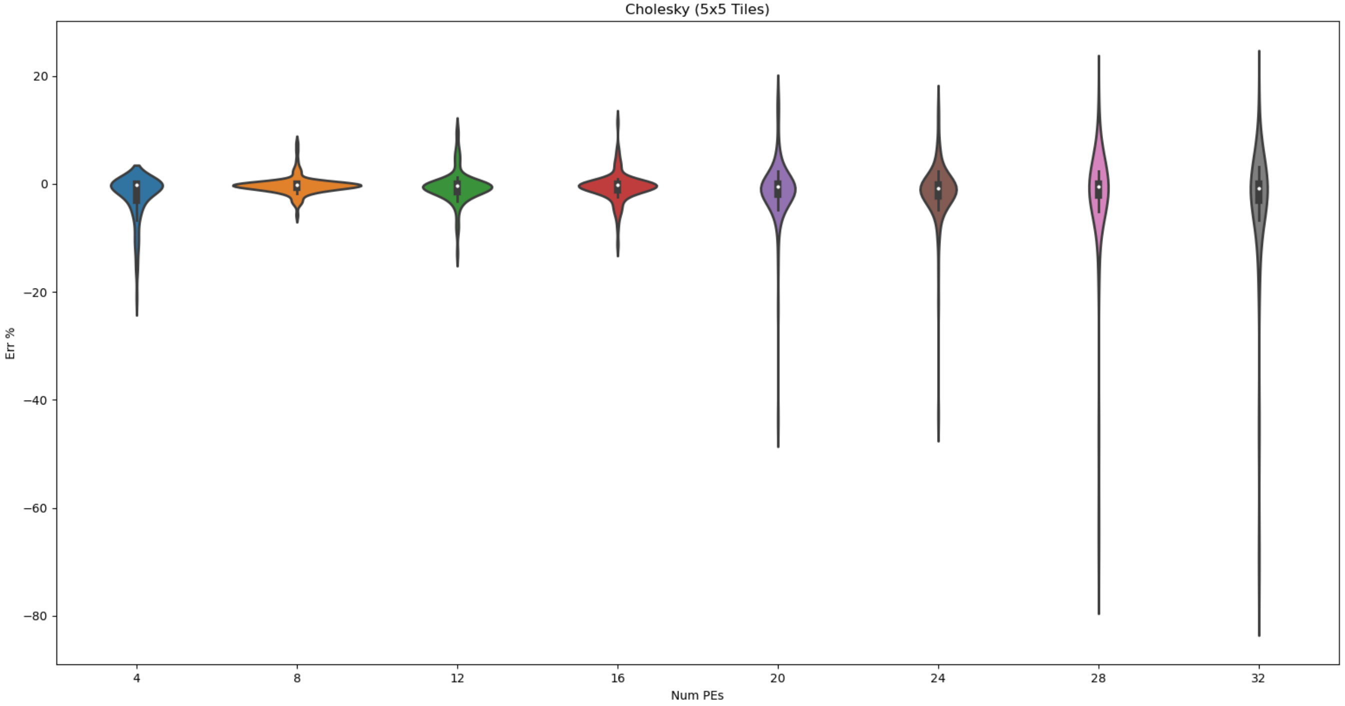
We implemented the buffer space detection in our proof-of-concept scheduling and simulations. Tested with random graphs. We want to:

- Verify that no DAG deadlocks
- Check how far are the scheduling and simulation makespan



Evaluation



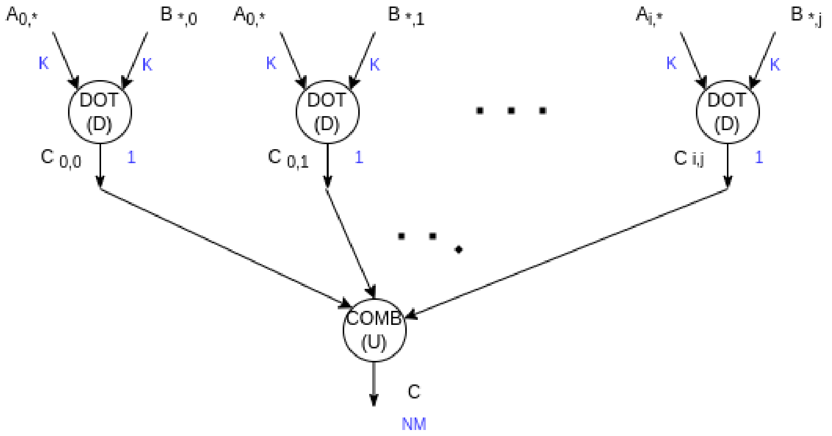


Next steps

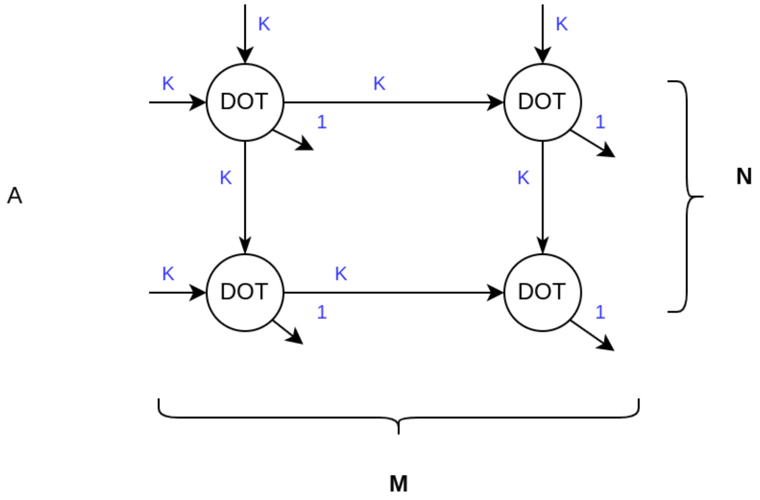
- Currently, we are looking at all undirected cycles, computed naively -> we need to improve on this
- Understand outliers

Different implementation, different representation

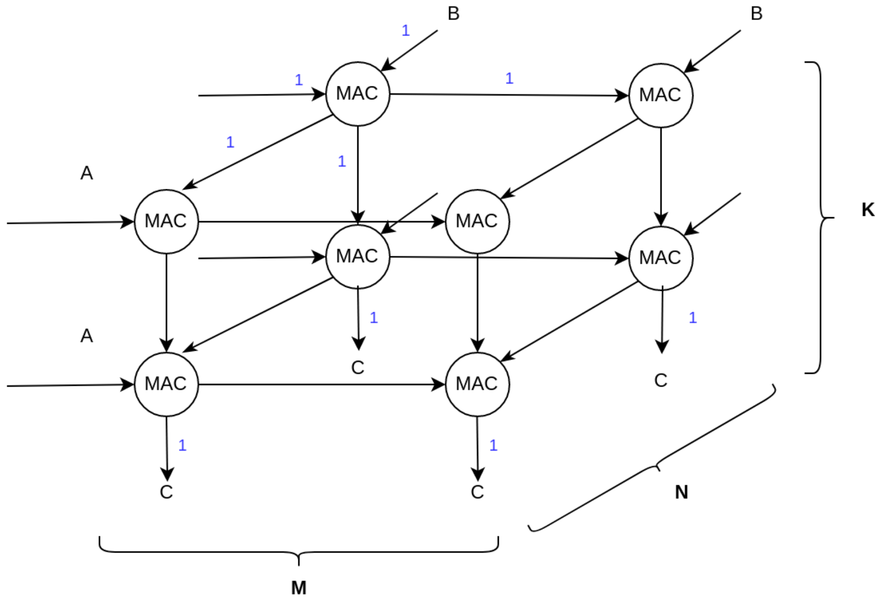
Consider the case of a MMM C=AB, where A is an NxK matrix, B is KxM and C is NxM



DOT Product



Systolic DOT

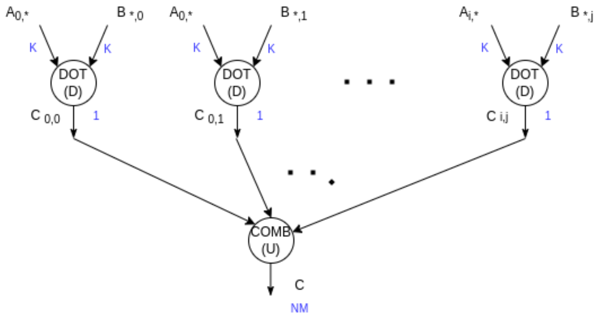


Systolic MAC

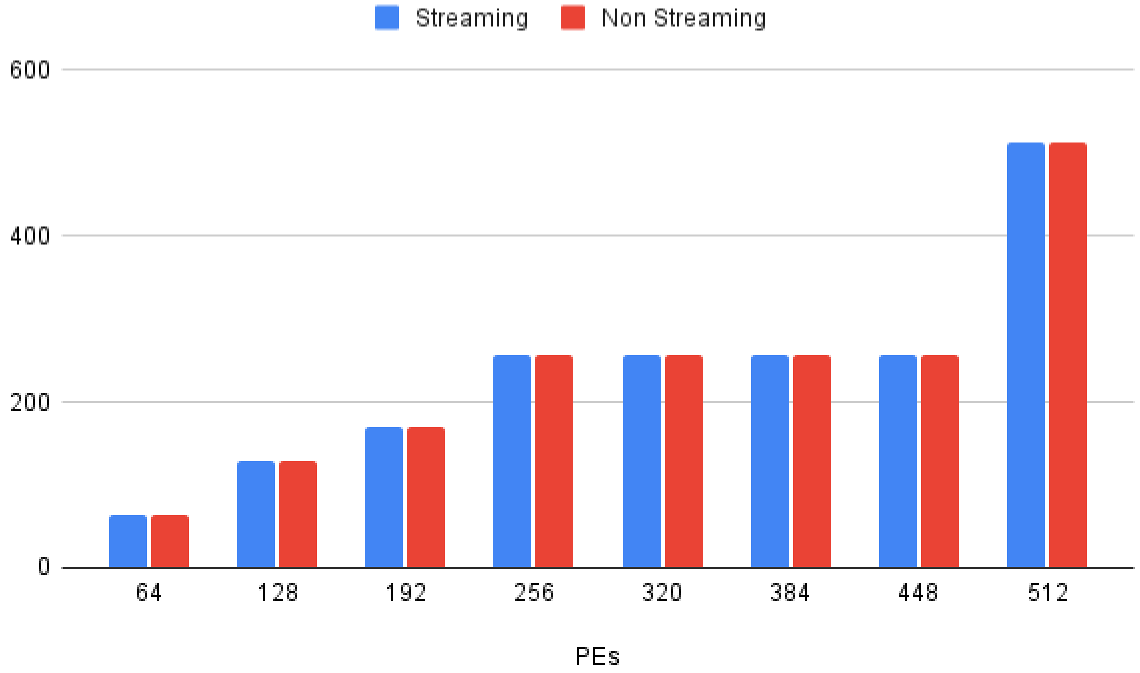
We can have others (for example with outer products, taking into account tilings, …)

Scheduling

Let’s consider N=64, M=8, K=64 (first MMM of the MIMO channel estimation)



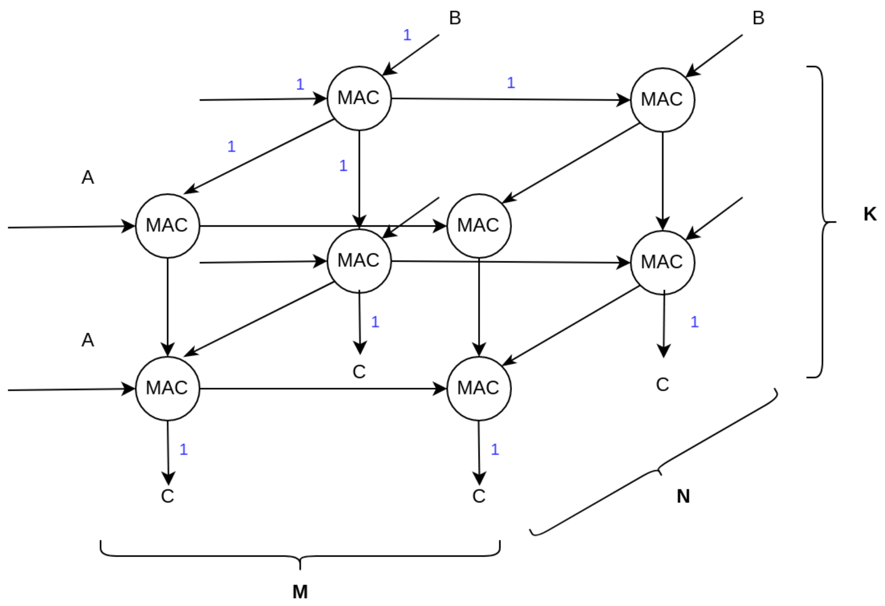
DOT Product



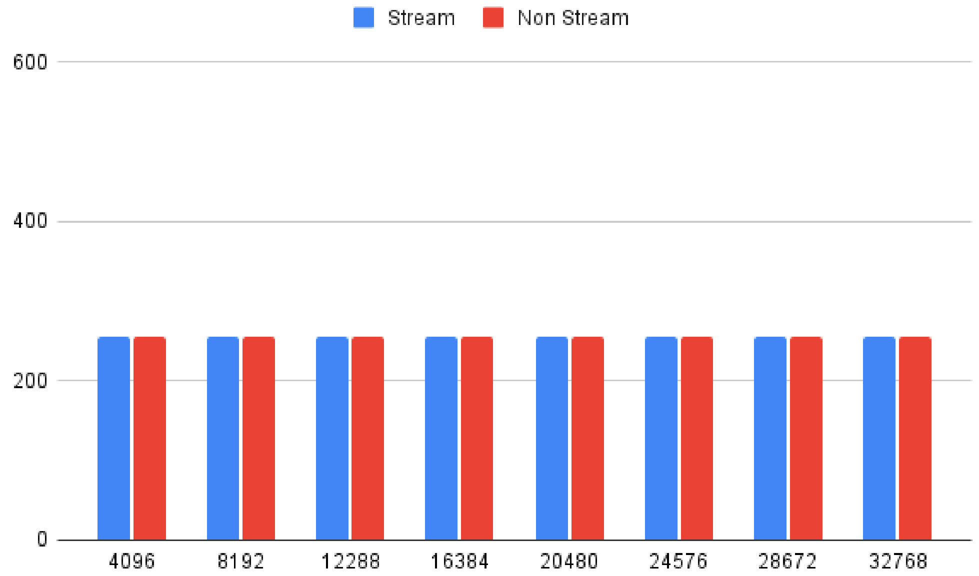
These results make sense:

- All the dot products are independent, hence the perfect scaling
- There is no edge to stream (hence no differences)
- (Here we are assuming infinite memory bandwidth)

Scheduling

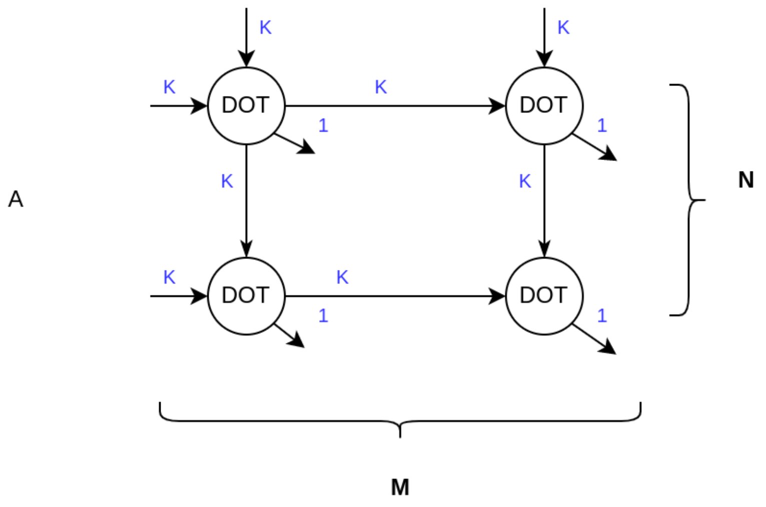


Systolic MAC

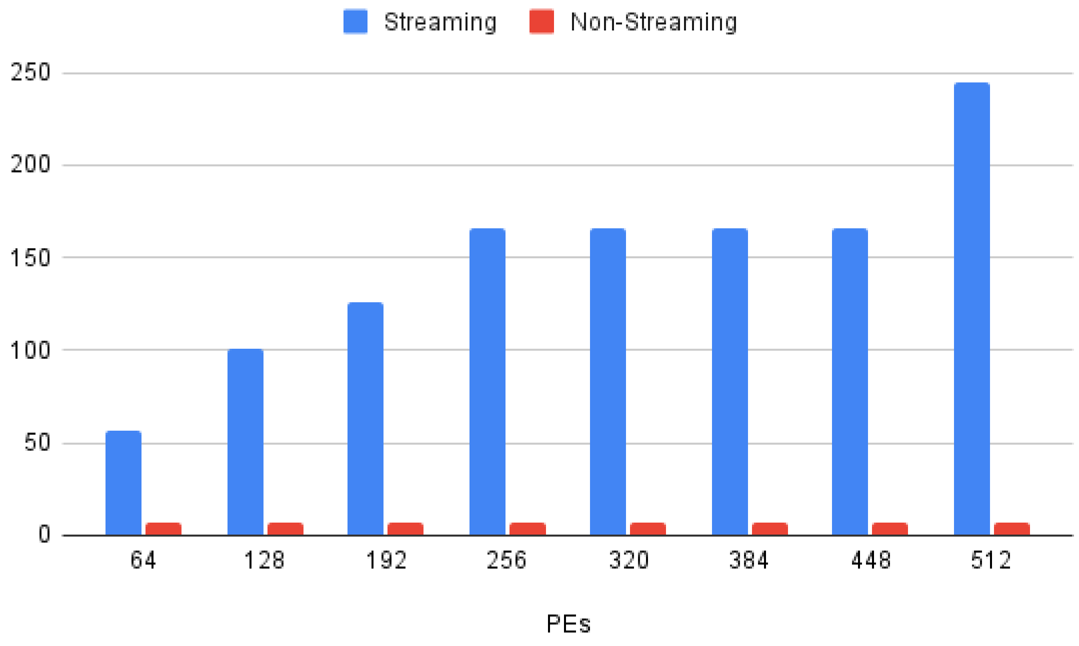


- Each task reads produce a single element
- Even if we stream, we don’t see any benefit from this
- We have the maximum parallelism in the middle of the computation (data propragates)

Scheduling



Systolic DOT

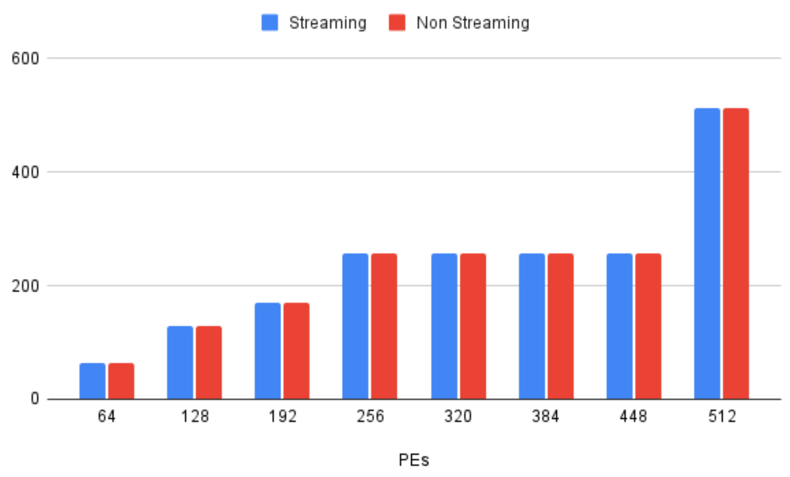


- Each task reads produce K elements
- We can see the benefits of streaming
- For non streaming, the maximum number of parallel dot product depends on the diagonal size

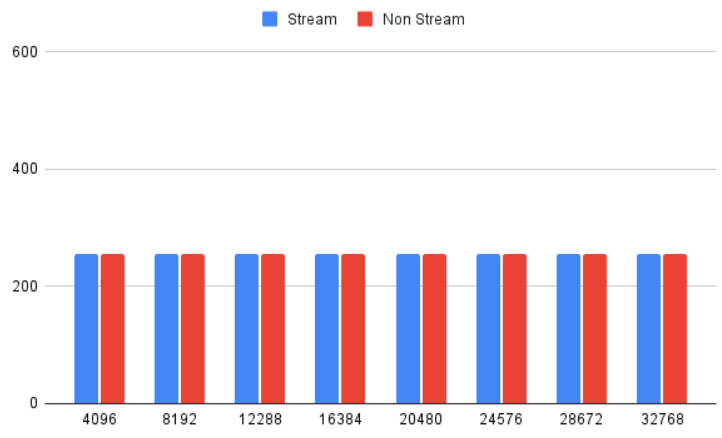
Considerations

We have seen three different implementation of MMM:

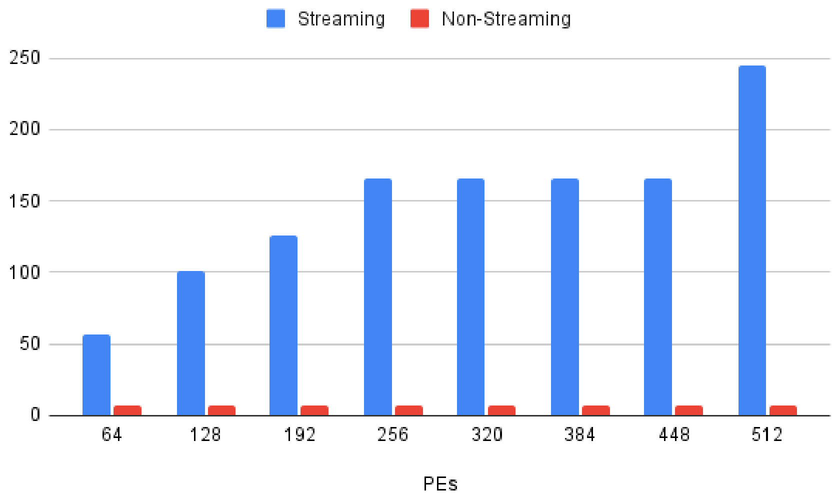
- We can add others (anything that in your opinion is worth investigating?)
- Each of them works with different granularity
- Different MMM size -> different scheduling results
- It is difficult to compare them: we should think how we want to do this



DOT Products

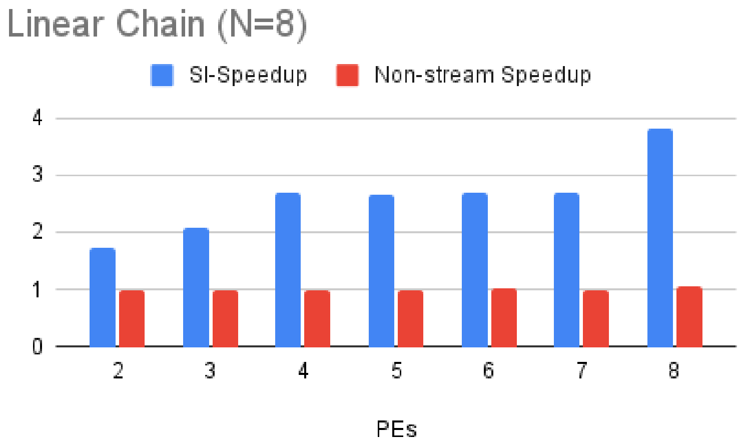


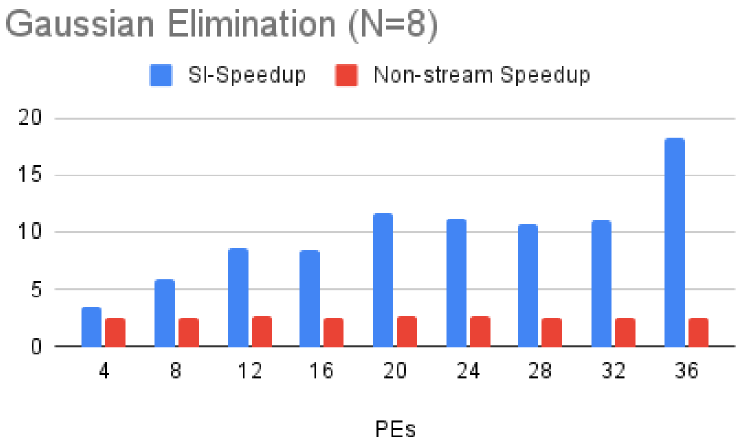
Systolic MAC

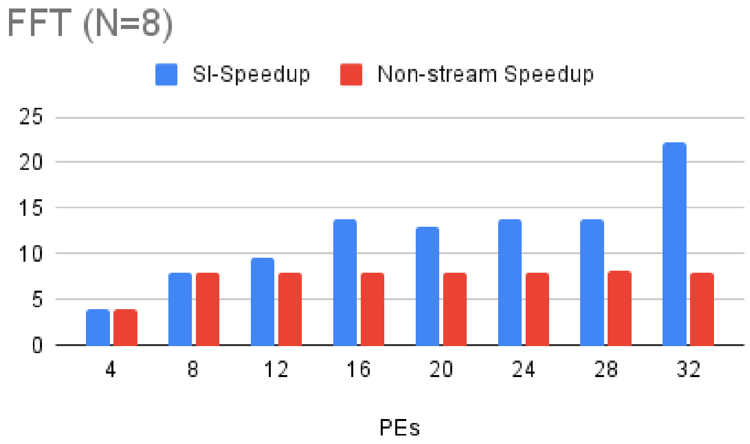


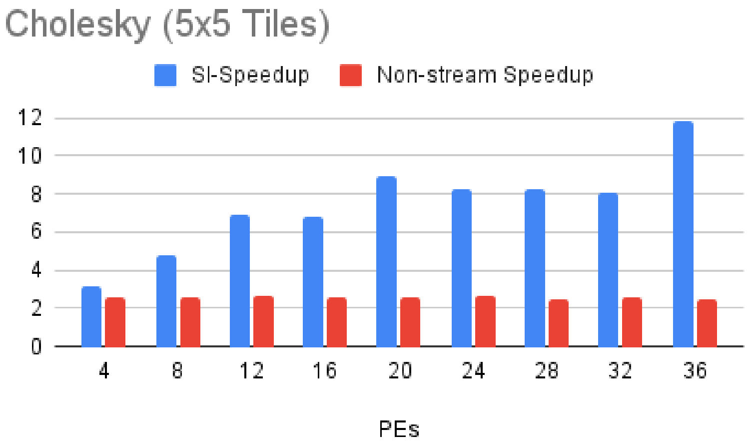
Systolic DOT

Heuristic results, some clarifications









- Fluctuating speedup may be due to wrong choices. We run other experiments with minor changes. Things improves (but not dramatically)
- Large jump at the end can be explained