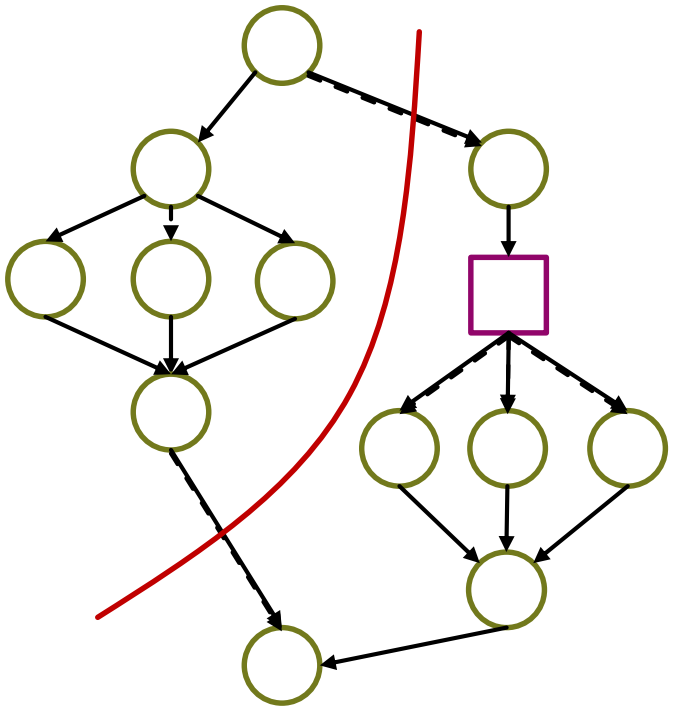
ASA: Enabling DSE

Off-/On-Chip buffer space and data movements

After we schedule a Canonical DAG we can analyze how data moves and where it can be stored (if needed)



- Pipelined communications (solid edges) happen within 2+ computational tasks in a spatial block
- Non-Pipelined/Buffered communications (dashed edges) occur:
  - Along edges that cross spatial blocks
  - Outgoing communications from buffer nodes

So far I considered non-pipelined data movements as off-chip accesses

But things are not black-and-white, and we could consider some of those communications to happen on-chip, resorting to on-chip memory space:

- It saves off-chip access (potentially reduce power consumption)
- It costs chip area (trade-off)

Finally, there is the buffer space that we need to avoid deadlocks

Off-/On-Chip buffer space and data movements

Is this (yet another) dimension that we want to explore?

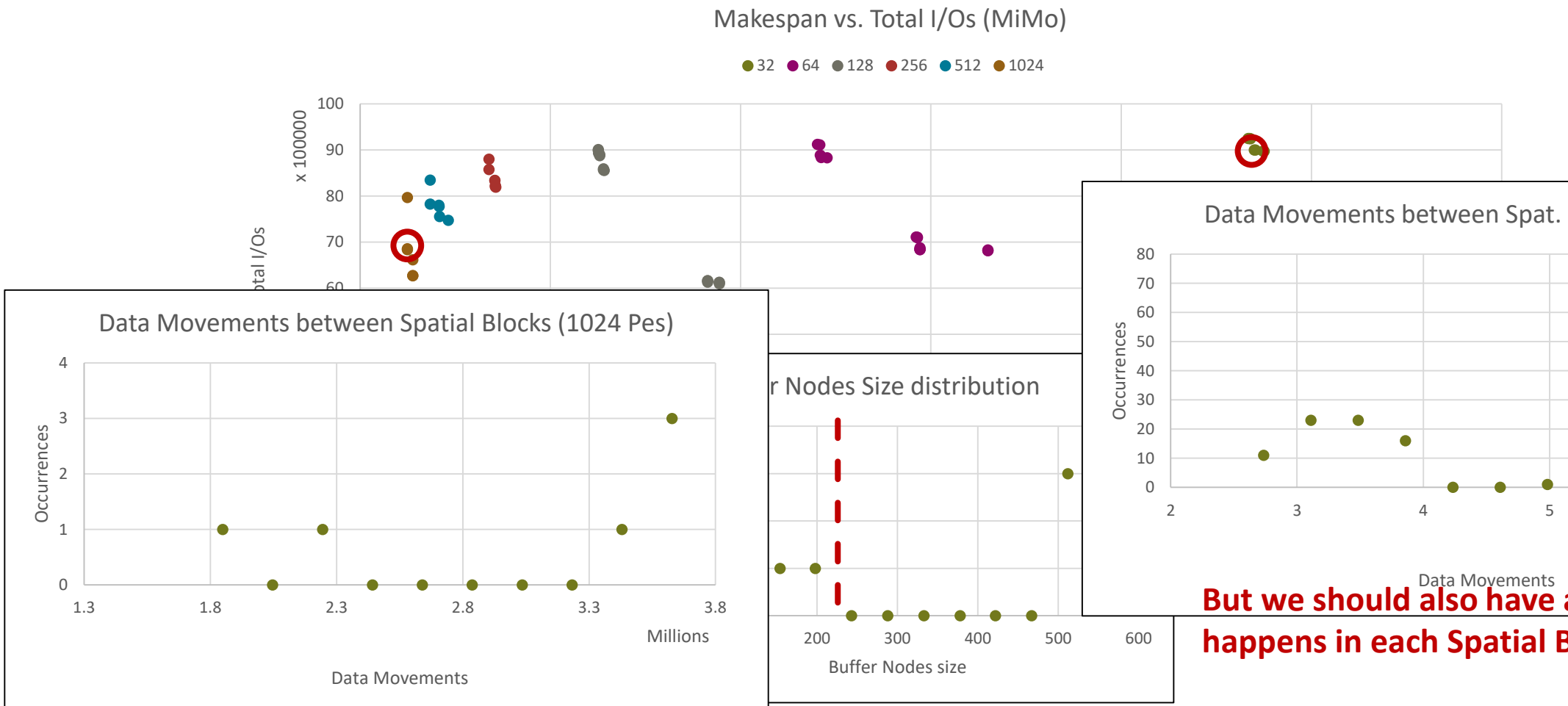
Difference between buffered communications due to buffer nodes and the ones due to spatial block crossing edges:

- Buffer nodes are used for data that must be replicated, read in a different order (RAM). They are implied by the computation, for example due to *accumulations, data reordering, data replication.*
  This means that, given a Canonical DAG, they will not change if we change the schedule or number of PEs)

- Frontier edges are used between streaming blocks (FIFO). They will change if we consider a different number of PEs, or, more in general, a different spatial partitioning

MiMo

Complete MiMo. Note that each point represent a different application expansion (and number of PEs)



But we should also have a look at what happens in each Spatial Block

Decide allocation of space for buffer nodes

Deciding whether to have on-chip or off-chip buffer space for buffer nodes is a form of Architecture Space Exploration (can be extended to inter-spatial block edges)

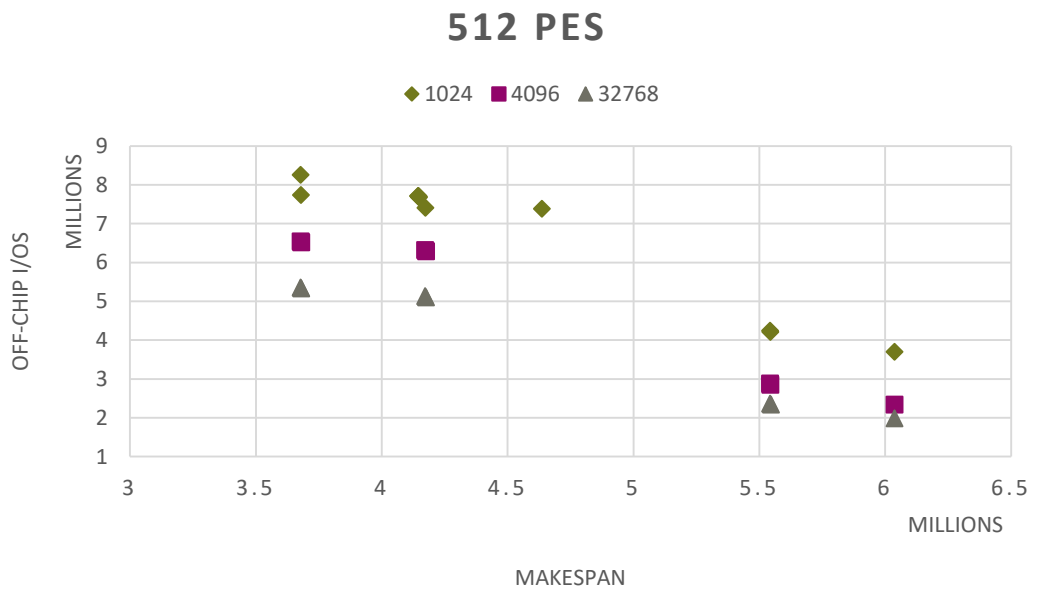
1. We generate the canonical DAGs out of the application, and we schedule them
2. For each (DAG, Schedule) we explore various thresholds for on-chip buffer space (e.g. 1KB, 2 KB, …)
   - For each of the spatial blocks we allow X data to be buffered on-chip, we decide which buffer node (if any) will stay in the on-chip memory. This is a another optimization problem: *"given a certain amount X of on-chip memory we want to allocate the buffer nodes that will give the highest reduction in off-chip I/O accesses"*
   - We count I/Os differentiating them between on-chip and off-chip accesses

Each of the considered thresholds gives a different solution, with the same makespan, but different on-chip/off-chip movements, hence different power and area

How to decide these thresholds must be understood: there could be some constraints due to Area. For the moment being the user can define them

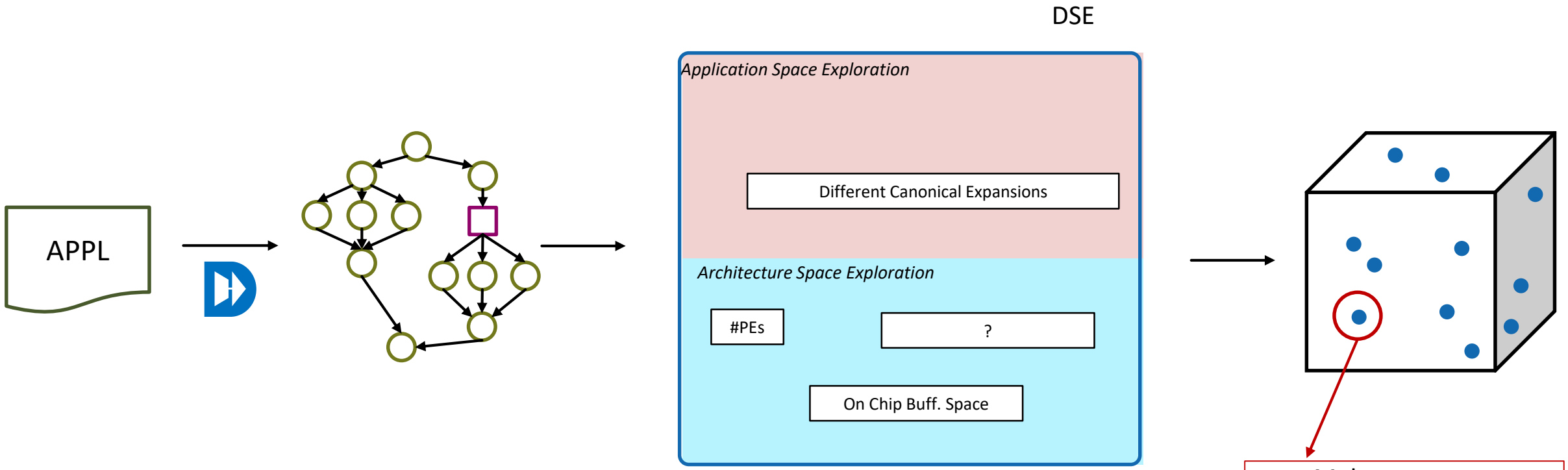
MiMo

Considering 3 possible on-chip buffer sizes (1 KB, 4KB, 32KB), we can generate 243 different configurations per each considered number of PEs. The following results are the pareto frontier points with NPEs= 512



These are just the pareto points. And how we determine pareto frontier depends on how we evaluate each configuration

Optimization Problem



We need some PPA model to evaluate each solution

Single- vs Multi-Objective Criteria

For Power-Area basic models at the moment being, any idea on how to improve them?

$$Power_D = \alpha\, A_{Off-chip} + \beta A_{On-Chip} \quad \text{where } \alpha > \beta$$

$$Power_S = \gamma NPEs$$

$$Perf = makespan$$

$$Area = \delta OnChipM + \epsilon NPEs$$

It should be easy to *plug* other models

Once we have these models we can pick pareto frontier points

$$\min_{x \in X} \{Power_D(x), Power_S(x), Perf(x), Area(x)\}$$

Ideally we would let the user define constraints for the set feasible solutions (e.g., all solutions that use less than W power)

Things to do (not in chro

Ideally: a DaCe program
In practice: NumPy Python Code that can be compiled to DaCe (e.g., DaCe does not support Collections and Recursion)

Use cases: be able to represent ap

- 5G/ use case: be more compliant with orig. appl — Needs your support
- ML: something more interesting than Lenet: Encoder. — DONE

Explore Application representations: be able to "compile" the application to Canonical DAGs

- Conveniently represent iterative algorithms — DONE
- Support considered use cases — DONE

Explore Architectures: be able to "consider" different macro-architectures

- We can change the number of PEs, this will affect the scheduling of the Canonical DAG — DONE
- Changing the PEs (still under the homogenous PE assumption) supporting only certain type of operations — Ongoing
- On-Chip Buffer Space Exploration — Ongoing

Space Exploration Goals and Optimization:

- Goals: optimize/minimize performance/power/area: we need way of estimating these
  - Performance is given by the scheduling makespan — Ongoing
  - Area: # of PEs, but also on-chip buffer space (e.g. for deadlock prevention) — Then needs your support
  - Power: directly proportional to the off-chip memory accesses

Documentation — Ongoing

Artifacts and documentation

Scheduling Framework: https://github.com/spcl/streamingsched

DSE Framework (depends on Scheduling Framework and DaCe): https://github.com/spcl/ASA

Discuss MIMO code, how to release it publicly