UDON - A case for offloading to general purpose compute on CXL memory

Jon Hermes, Josh Minor, Minjun Wu, Adarsh Patil, Eric van Hensbergen

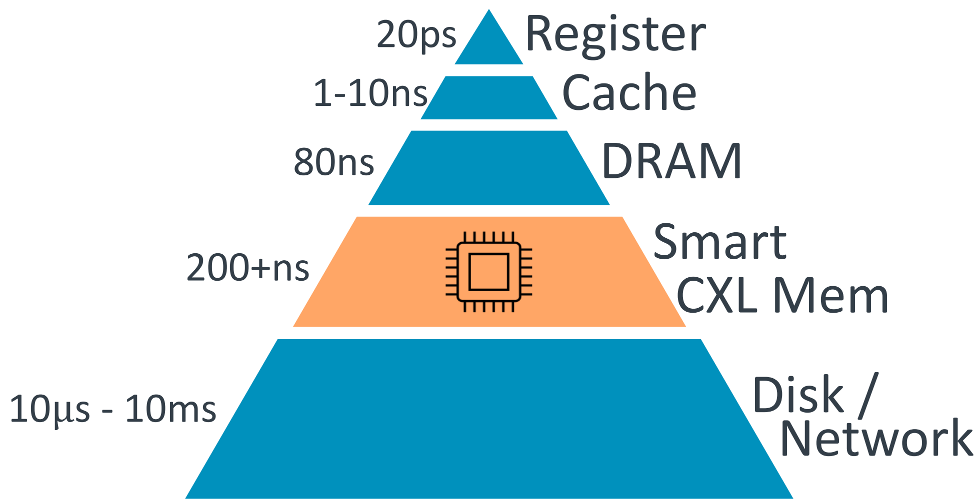
Arm Architecture and Technology Group

3rd Workshop on Heterogeneous Composable Disaggregated Systems (HCDS), April 2024

AI-generated image

Vision of CXL Future

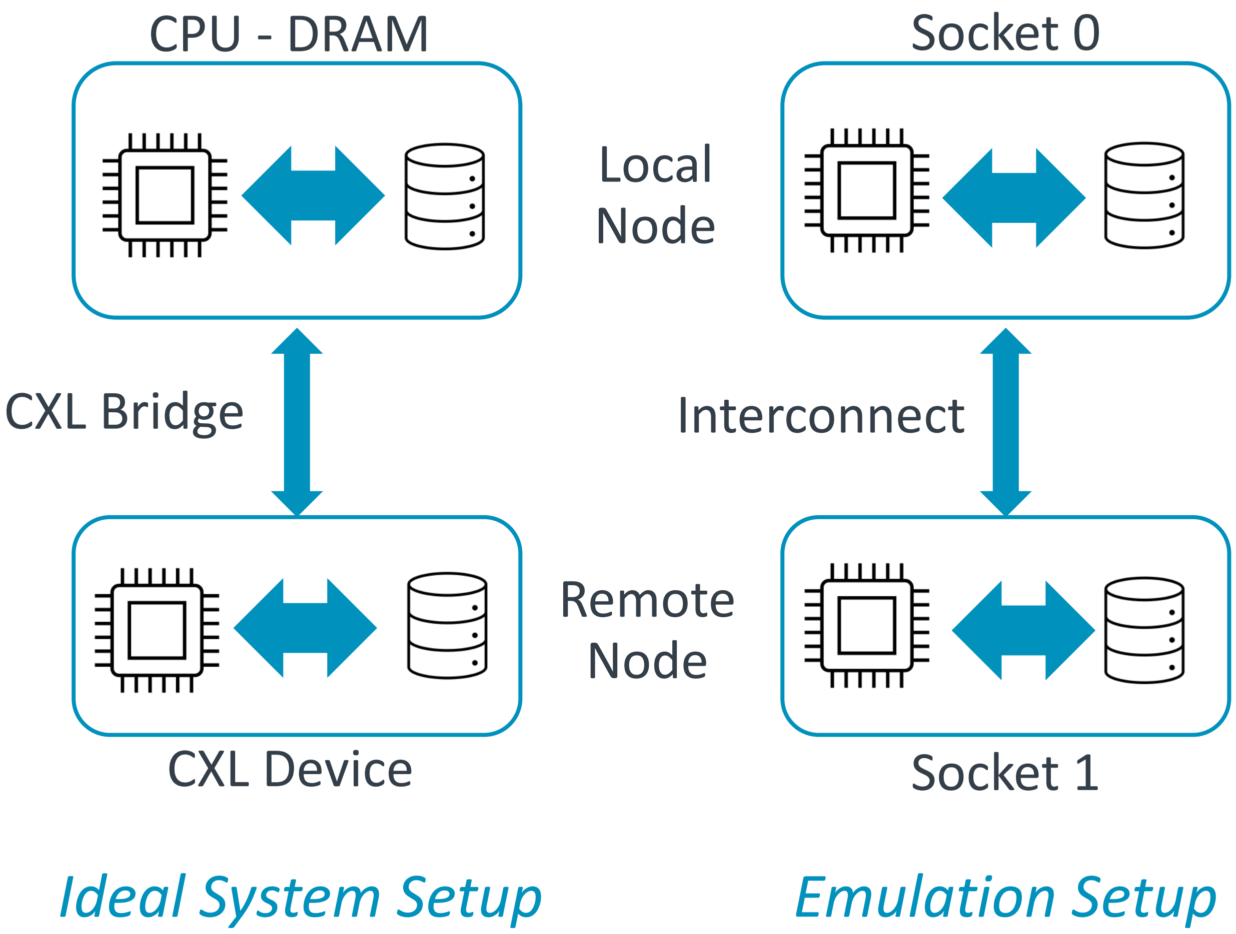
- We imagine a future where disagg memory is increasingly common
- **Memory is expensive**
  - Can be **50%** of server cost for Azure [1]
  - **40%** of rack cost for Meta [2]



- Memory BW bound app performance loss may follow ratio of single socket memory performance to socket-socket/socket-device
- **Improvements to per-socket performance exacerbates this problem (mem wall)**
- **Our idea: It should be possible to mitigate performance loss of CXL backed memory by dispatching targeted compute tasks to the memory pool**
- Question: Do we need specialized In-Mem compute, or will general purpose CPU suffice?

Modeling CXL without CXL Hardware

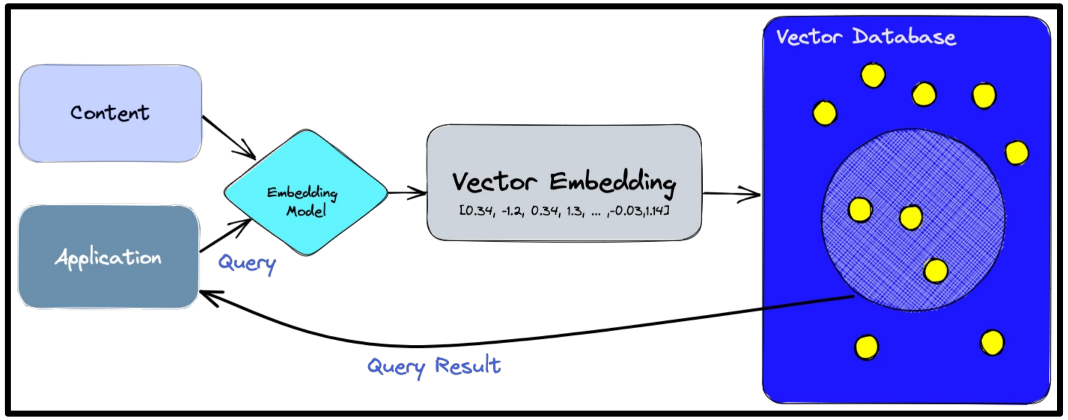
- Using the same strategy from Pond paper: forcing cross-socket NUMA access to emulate CXL link
- Measured: +100ns memory access delay, 32GB/s cross-socket bandwidth
- Real CXL hardware will be **no better** in terms of latency or bandwidth, so we expect all our findings to be a "best case" scenario



Workload Analysis: VectorDB

Vector Databases

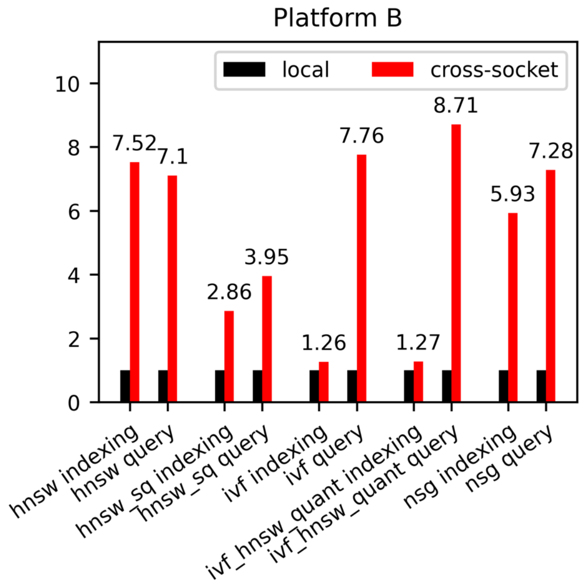
- What are vector databases?
  - Vector databases store and maintain **embedding vectors** from structured/unstructured data (i.e. text or images)
  - The distance of 2 embedding vectors in the vector space implies their semantic similarity
  - Traditional distance calculation is expensive. Vector databases use **vector indexing** to pre-calculate the distances to enable faster retrieval at query time

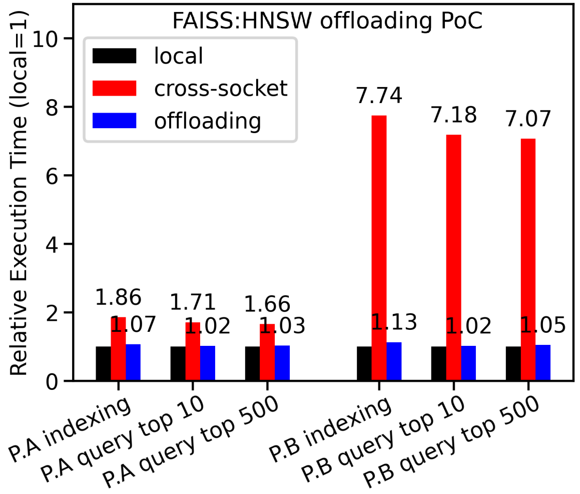


- VectorDB codebase
  - FAISS (Facebook AI Similarity Search): library by Meta, integrated into many VectorDB products (Milvus etc.)
  - Key kernels:
    - **1. Indexing**: different algorithms, pre-compute to enable fast search
    - **2. Query**: irregular accesses and BW pressure

VectorDB (faiss) HNSW kernel offloading

- HNSW: Hierarchical Navigable Small Worlds
  - One indexing algorithm used commonly in VectorDB, uses layers to reduce the neighbor search space
  - **Both indexing (write) and query (read) are memory sensitive**
- Proof of Concept offloading
  - Two processes running separately:
    - host runs the main app
    - device runs an offloading service
- **Results**: For specific kernels, offloading PoC demonstrates huge performance benefit in near memory processing
  - Up to 7x improvement in latency
  - Limited overhead (under 10%)

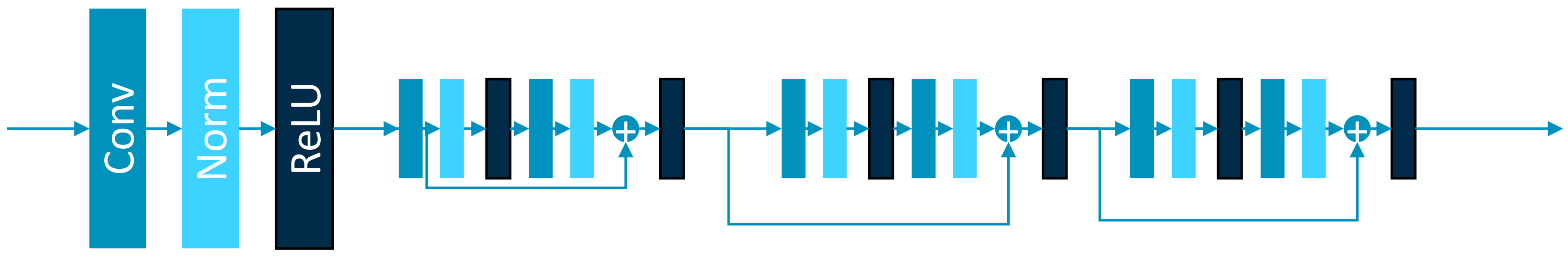




Platform B	Indexing	Query top 10	Query top 500
Saving	6.87x	7.04x	6.75x
Overhead	3.76%	5.84%	8.22%

Workload Analysis: ML Inference

Typical Machine Learning Workload



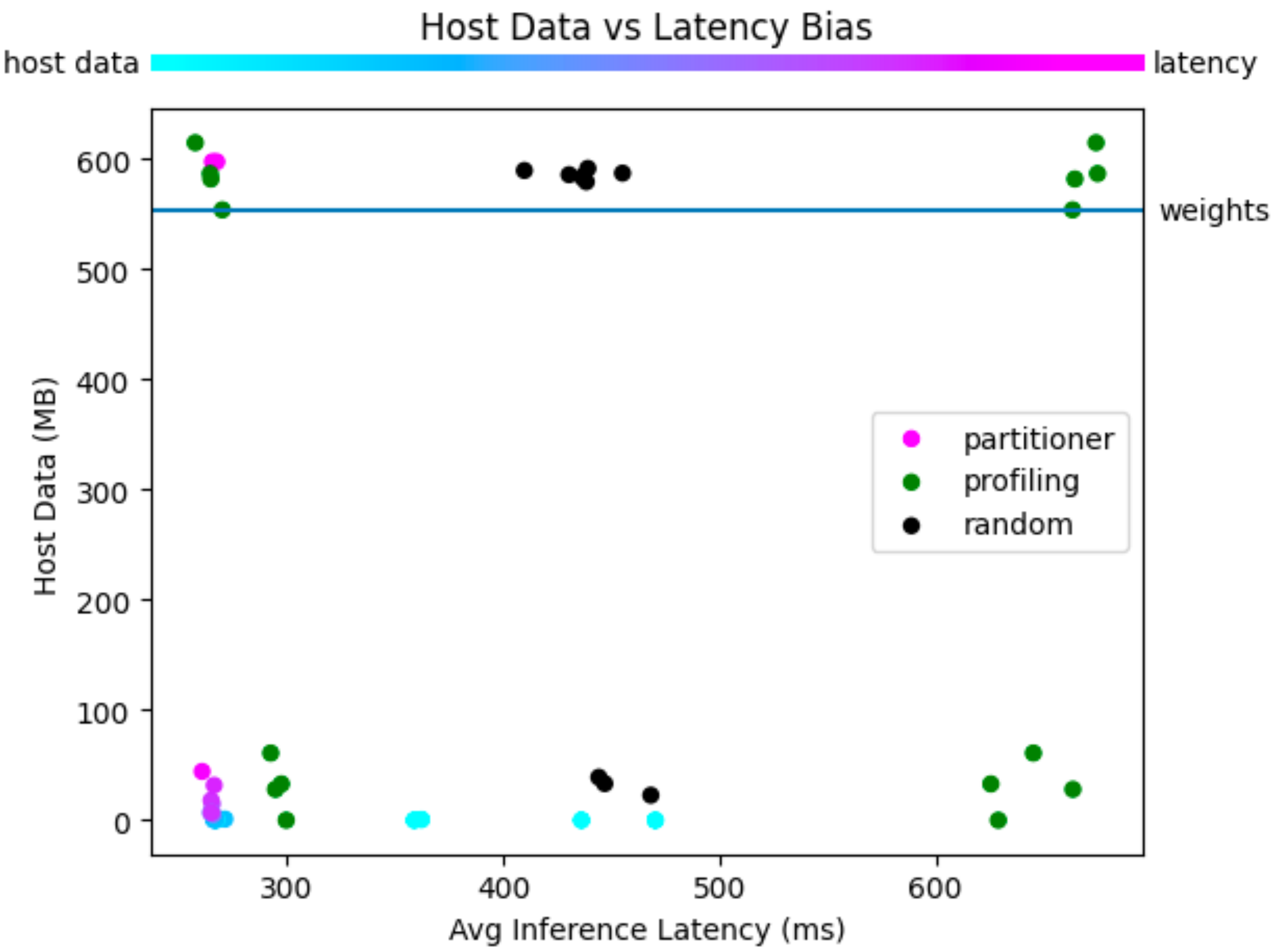
A simplified CNN model ResNet

- Model consists many operations structured in DAG (Directed Acyclic Graph)
  - Convolution (Conv), Normalization (Norm), Activation (ReLU)
- Different models use different sets of operators
- Different kernels have very different characteristics

Creating a Memory-Compute Placement Algorithm

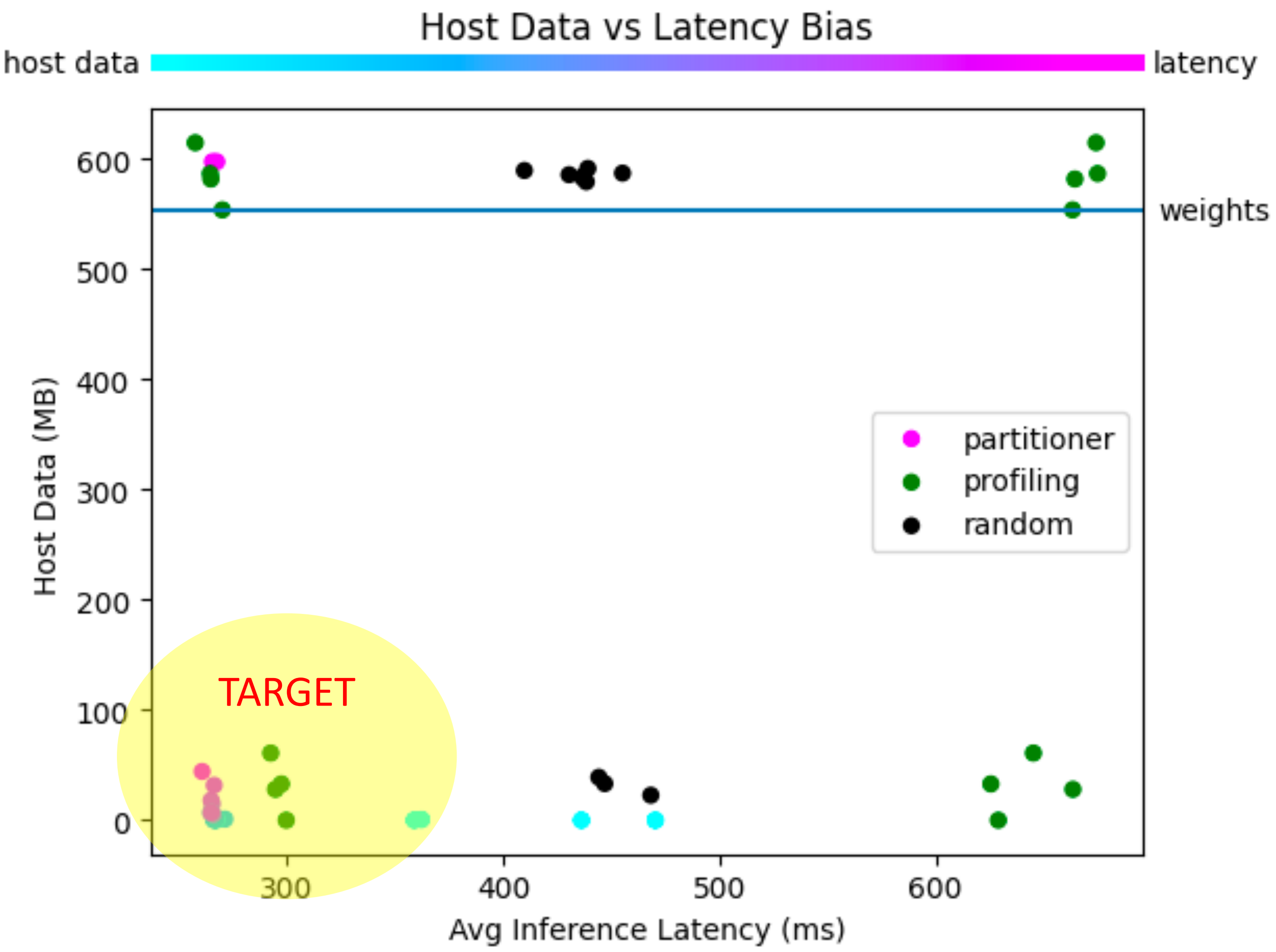
- CXL Inference Serving - there exists a multi-objective function:
  - **Maximize** the “far” CXL pool memory allocation
  - **Minimize** the total run-time of the ML model (referred to as “latency”)
  - **Weighted sum cost function** w/ weights [0-1] selected to prioritize host data placement or latency
- **Offline Memory-Compute Placement Algorithm** takes the cost function bias as input, and assign for each layer LOCAL/REMOTE:
  - Weights
  - Intermediate Tensors
  - Compute
- To emulate “lesser” cores on CXL Mem Pool side, drop clock freq on socket 2 from 3GHz to 2GHz

Partitioning Results – VGG_16 Platform B



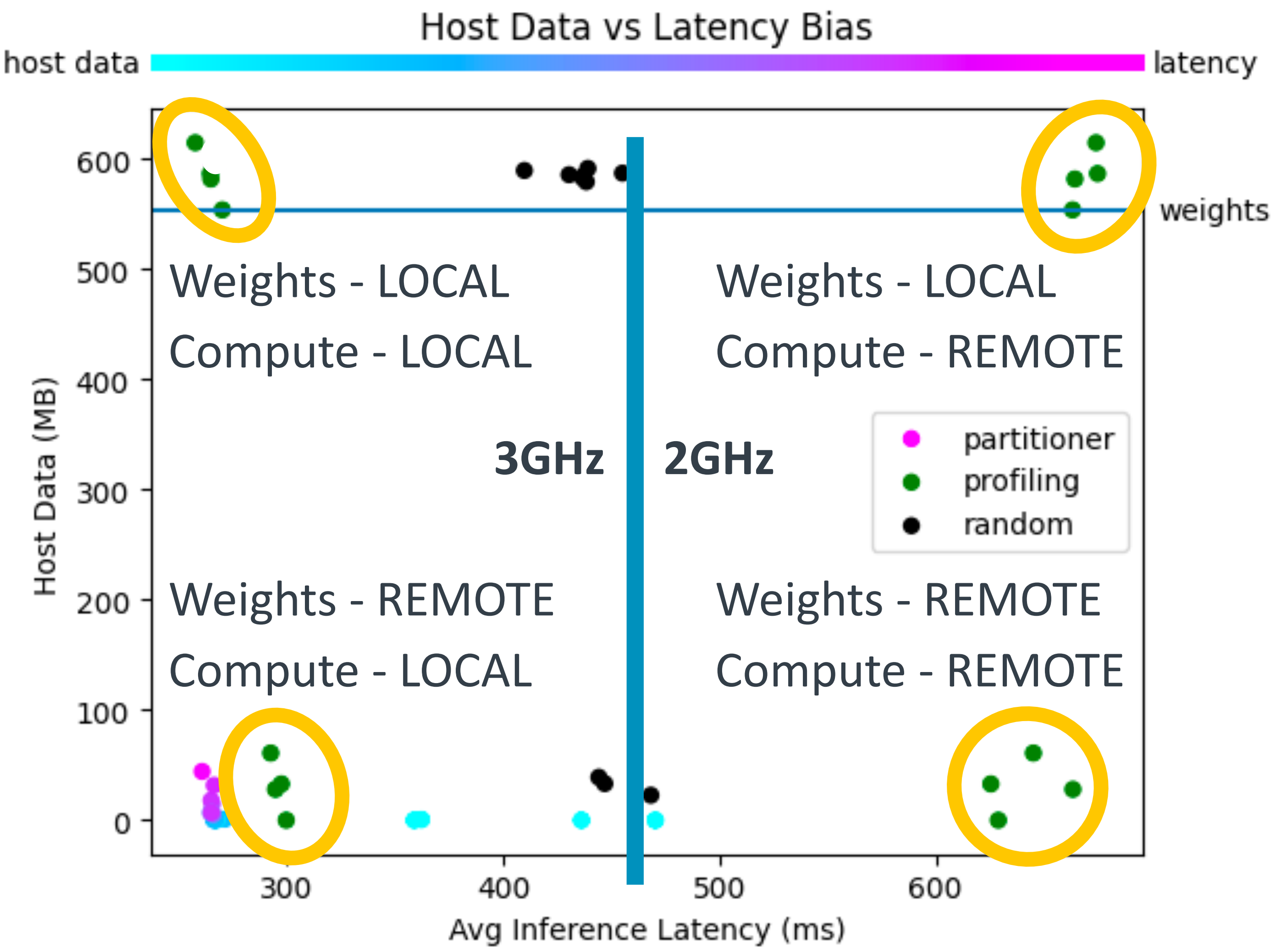
Partitioning Results – VGG_16 Platform B

- Target of optimization is to find **pareto frontier** of solutions
- Measure of efficacy:
  - Given fixed amount of data placed on host, no solution should improve on latency that we didn't find
  - Vice versa for fixed amount of latency



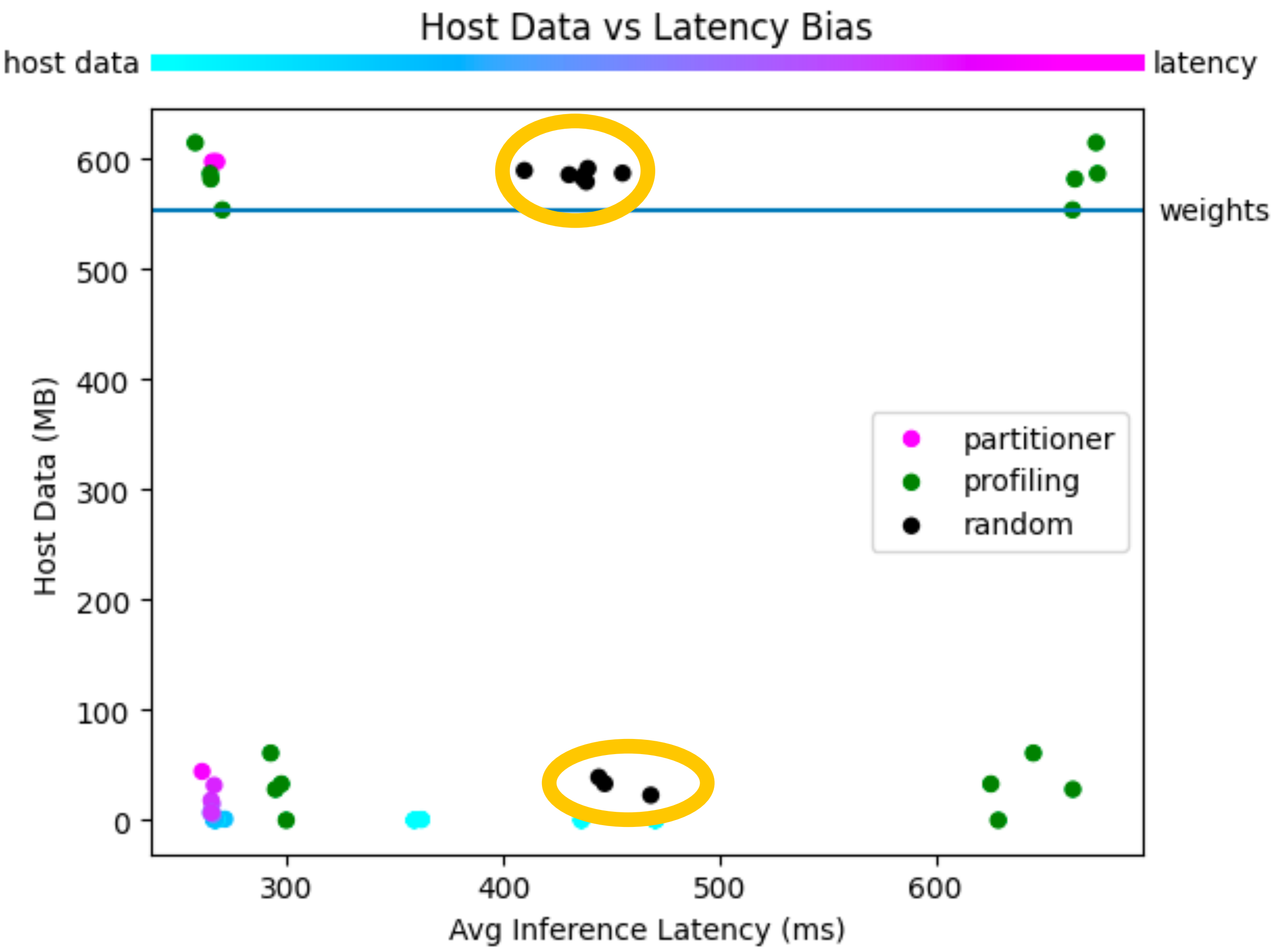
Partitioning Results – VGG_16 Platform B

- Points in **GREEN** captured during profiling step to generate perf lookup table used in partitioning strategy
- All intermediate tensors placements generated, then for each run w/ compute **all** LOCAL/REMOTE
- No migration -> no migration overhead
- Symmetry across X axis, as twin points differ by compute at 3GHz/2GHz



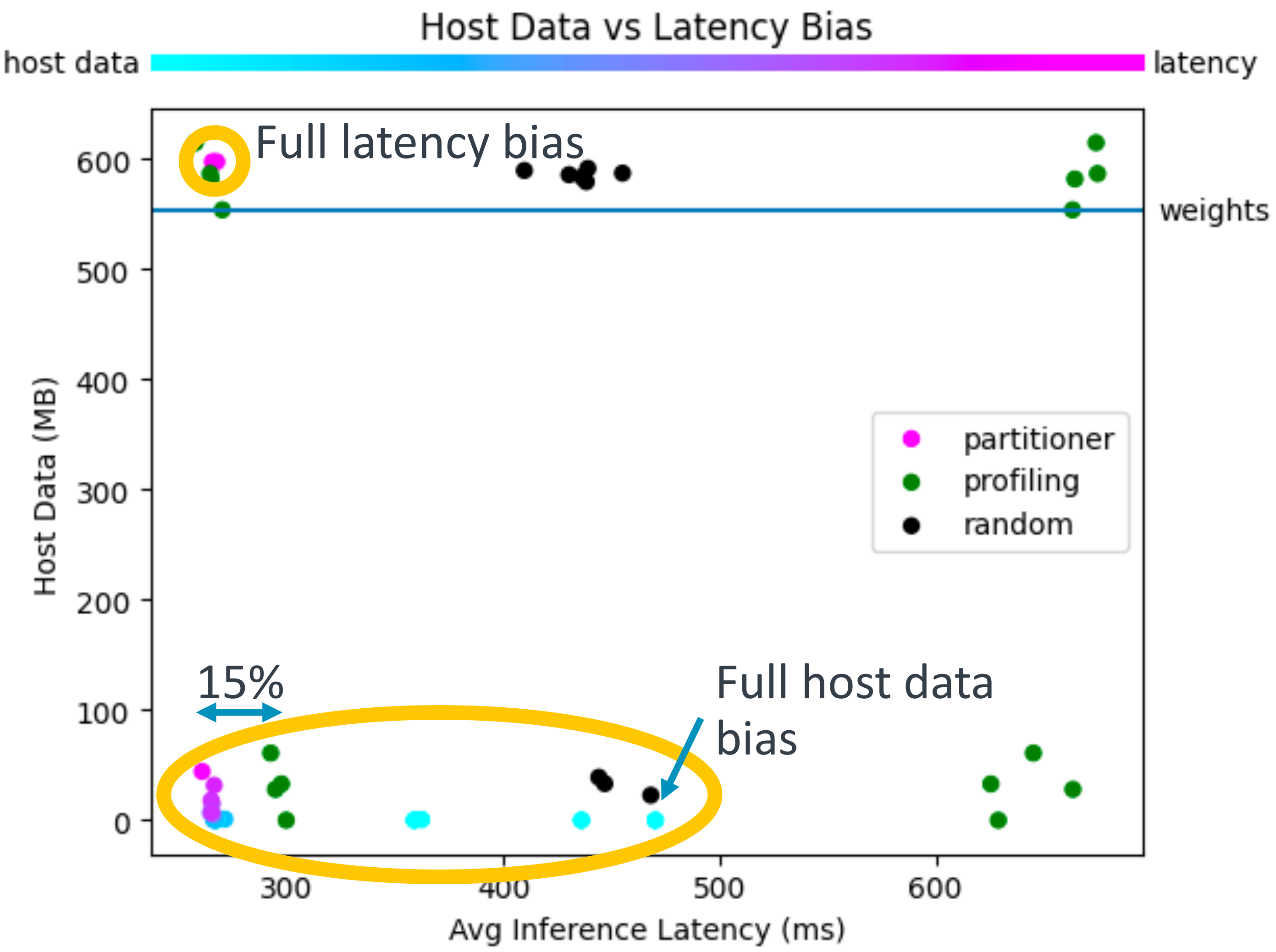
Partitioning Results – VGG_16 Platform B

- Points in **BLACK** randomly generated run configurations
  - Weight placement LOCAL/REMOTE
  - For each op random:
    - Result Tensor LOCAL/REMOTE
    - Compute LOCAL/REMOTE
- Suffers overhead of compute migration
- Provides baseline for algorithmic improvement



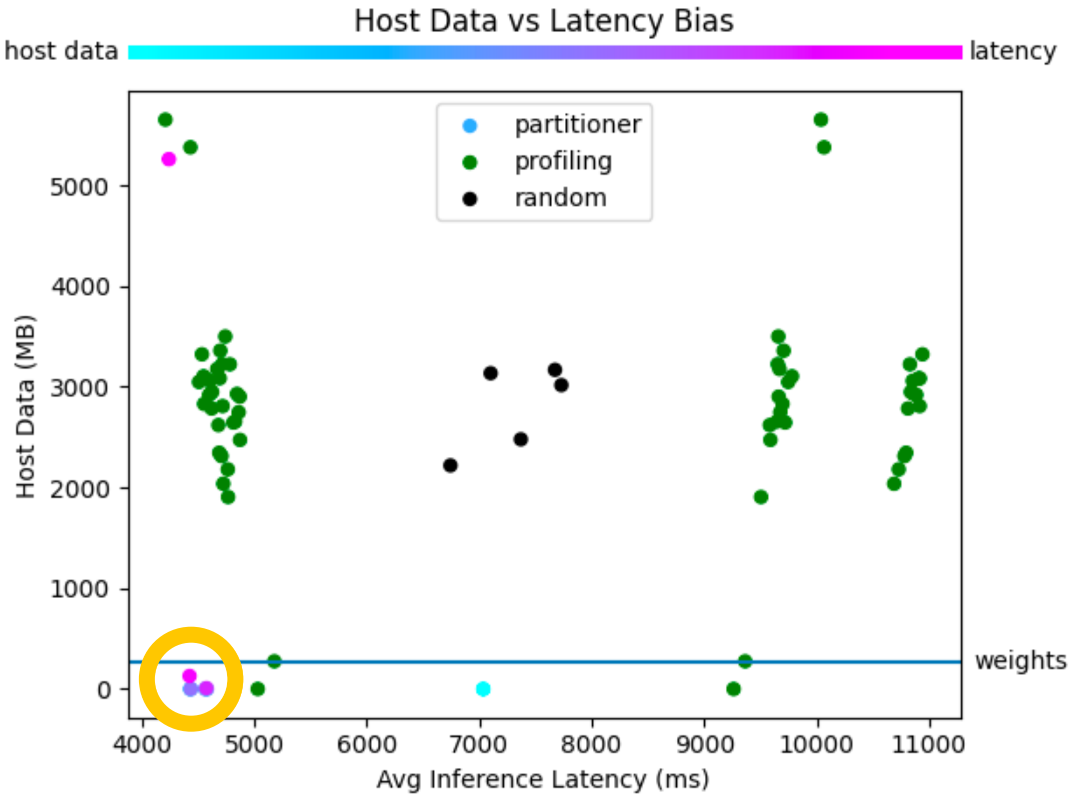
Partitioning Results – VGG_16 Platform B

- Points in **BLUE** - **PINK** spectrum captured based on user selected host-data or latency bias
- Sweep across host-data/latency bias evenly 0 -> 1
- 15% latency improvement vs no compute offload
- **Takeaway**: Intelligent mem partitioning **and** compute offload -> run models with **nearly all memory remote** with minimal latency penalty
- **Takeaway**: Compute offload to slow cores recovers lost latency

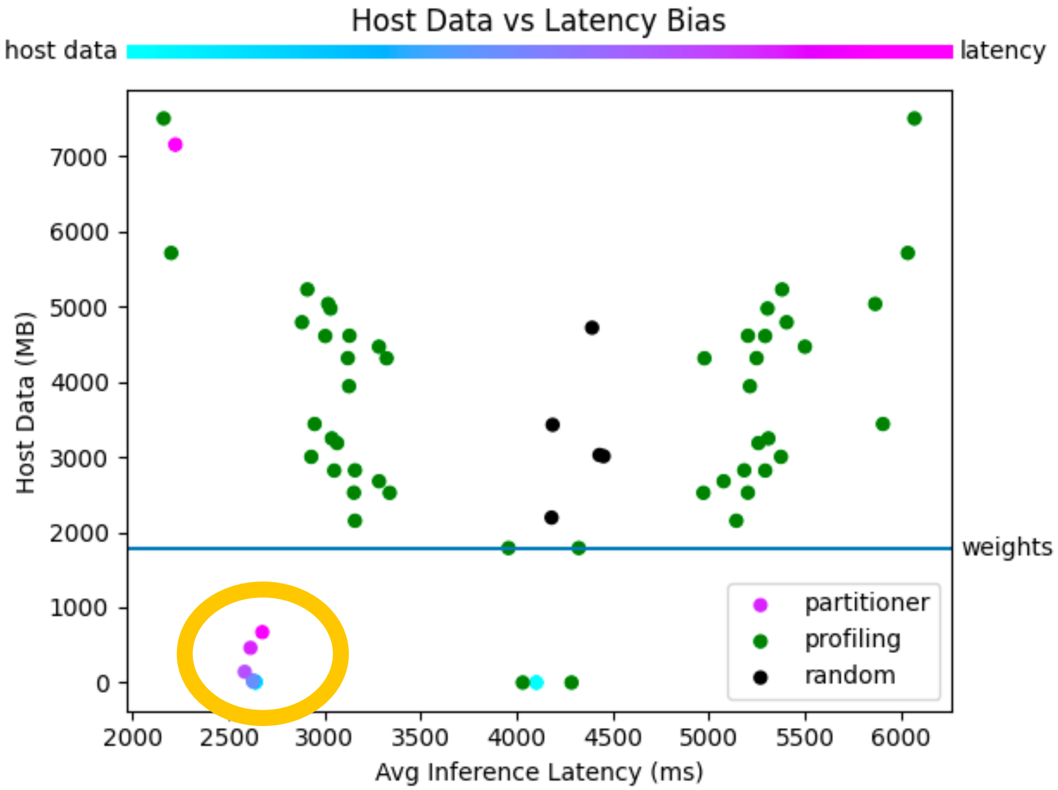


Two other models, same result:

Yolo v3



Stable Diffusion



Results hold across models, and improvement scales with memory sensitivity. The best use of far memory in almost all cases needs to move both compute and data.

Conclusion and Takeaways

- **Data Placement** and **Compute Placement** are both important
  - The most efficient use of far memory *requires* compute offload; not just data placement
- Identifying and offloading memory-sensitive parts of applications using **Near-Memory Compute** helps mitigate the latency and bandwidth limitations inherent in these types of devices
  - In some cases, it can nearly recoup all lost performance
- **Challenges** of course exist to support adoption of function-level compute offload:
  - Software must be easily broken down into tasks and profiled for memory sensitivity
  - Host and CXL devices must share addressing if not be fully coherent for efficient offload
  - CXL devices need to include dedicated Near Memory Compute resources
- **Automation** of function-level profiling and offloading a good direction for future research

Thank You
Danke
Gracias
Grazie
谢谢
ありがとう
Asante
Merci
감사합니다
धन्यवाद
Kiitos
شكرًا
ধন্যবাদ
תודה
ధన్యవాదములు