Move your code, not your data

Michael Giardino, Siddharth Gupta, Lukas Humbel, René Müller, Anirban Nag

The view of the Memory Wall from the core

Example modern server processor

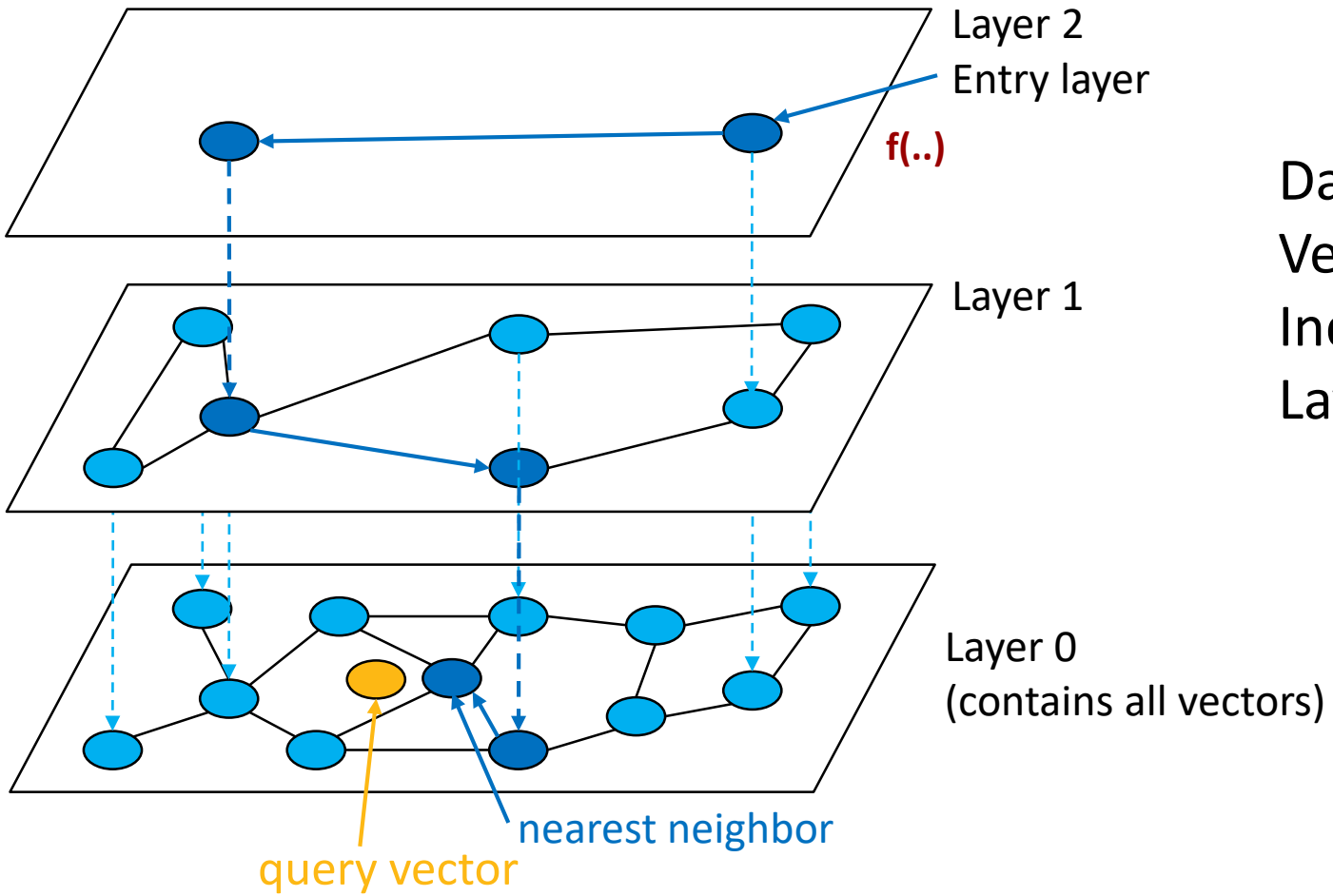
- 3 GHz
- 6-wide execution
- 320 instruction reorder buffer

Memory type	Cycles	Time @ 3 GHz	Instructions (6-wide)
L2 (hit)	14	4.6 ns	84
L3 (hit)	50	16.6 ns	300
DRAM	~180	80 ns	1,080
CXL memory	~750	250 ns	4,500

So how do we solve this?

- Bigger reorder buffers to find more parallelism (512 in Sierra Forest)
- Find more threads to run in parallel using SMT (2, 4, and 8 way)
- Attempt to prefetch data close to the core (smarter multilevel prefetchers)
- Bigger caches to hold this prefetched data
- Identify and move hot pages closer to data

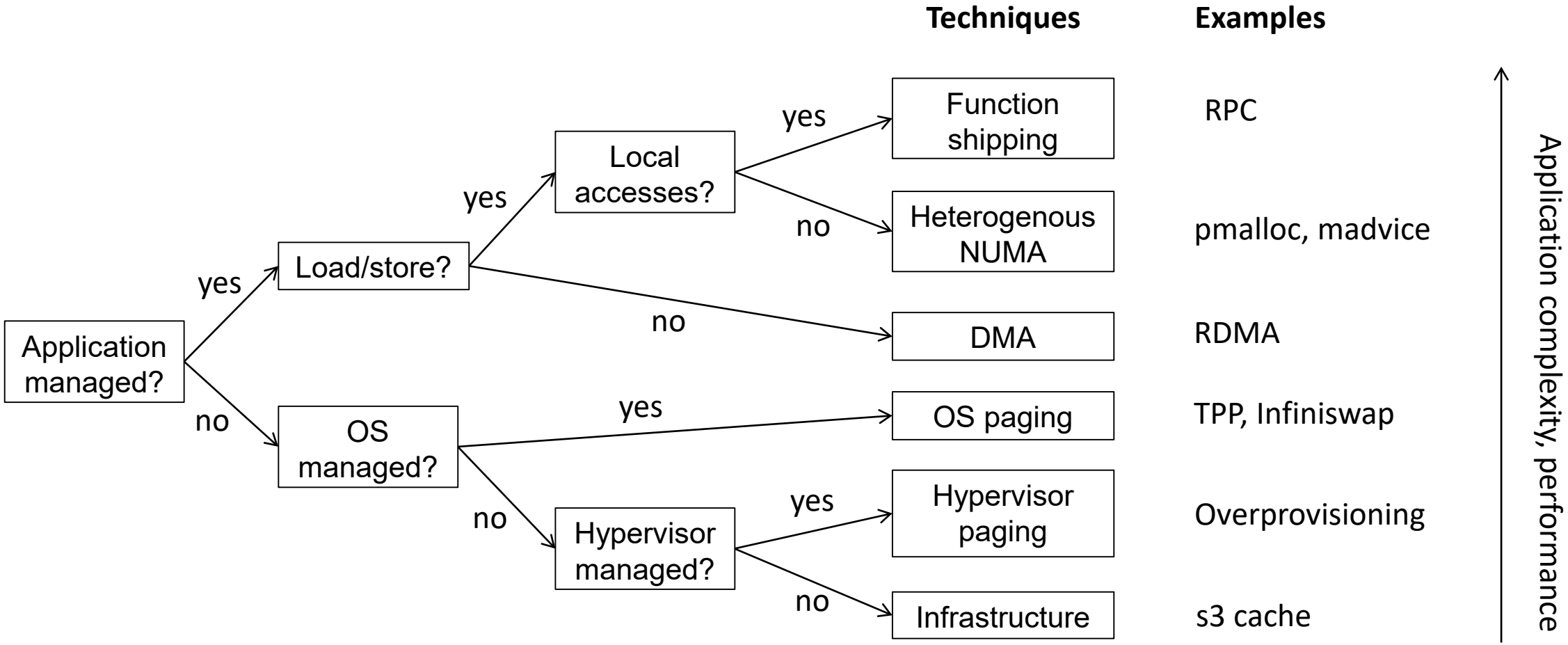
Vector Search Example



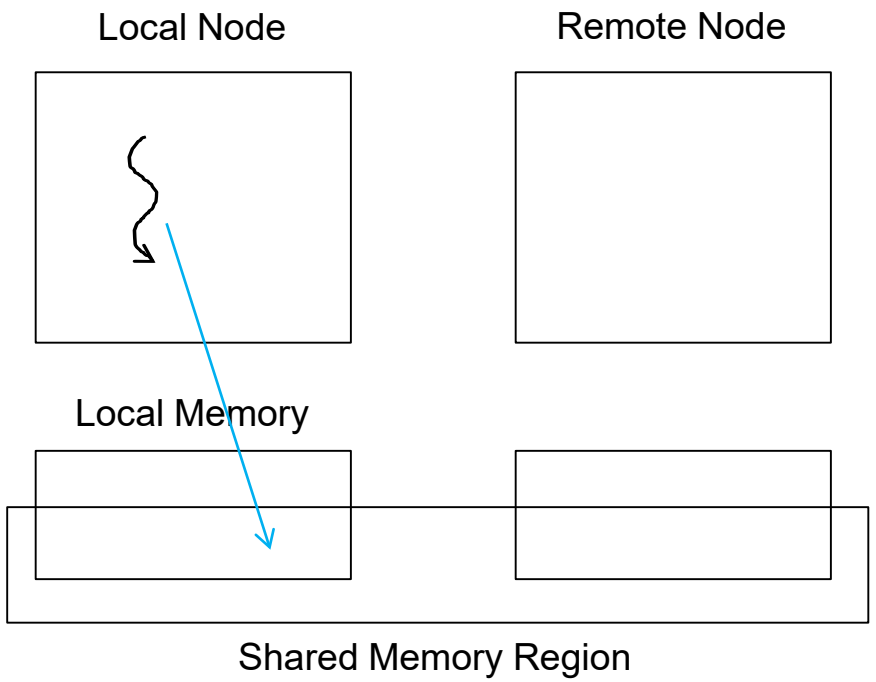
Data structure sizes for 1B vector BigANN
Vector size: 128 B
Index: **122 GiB**
Layer0: **256 GiB**

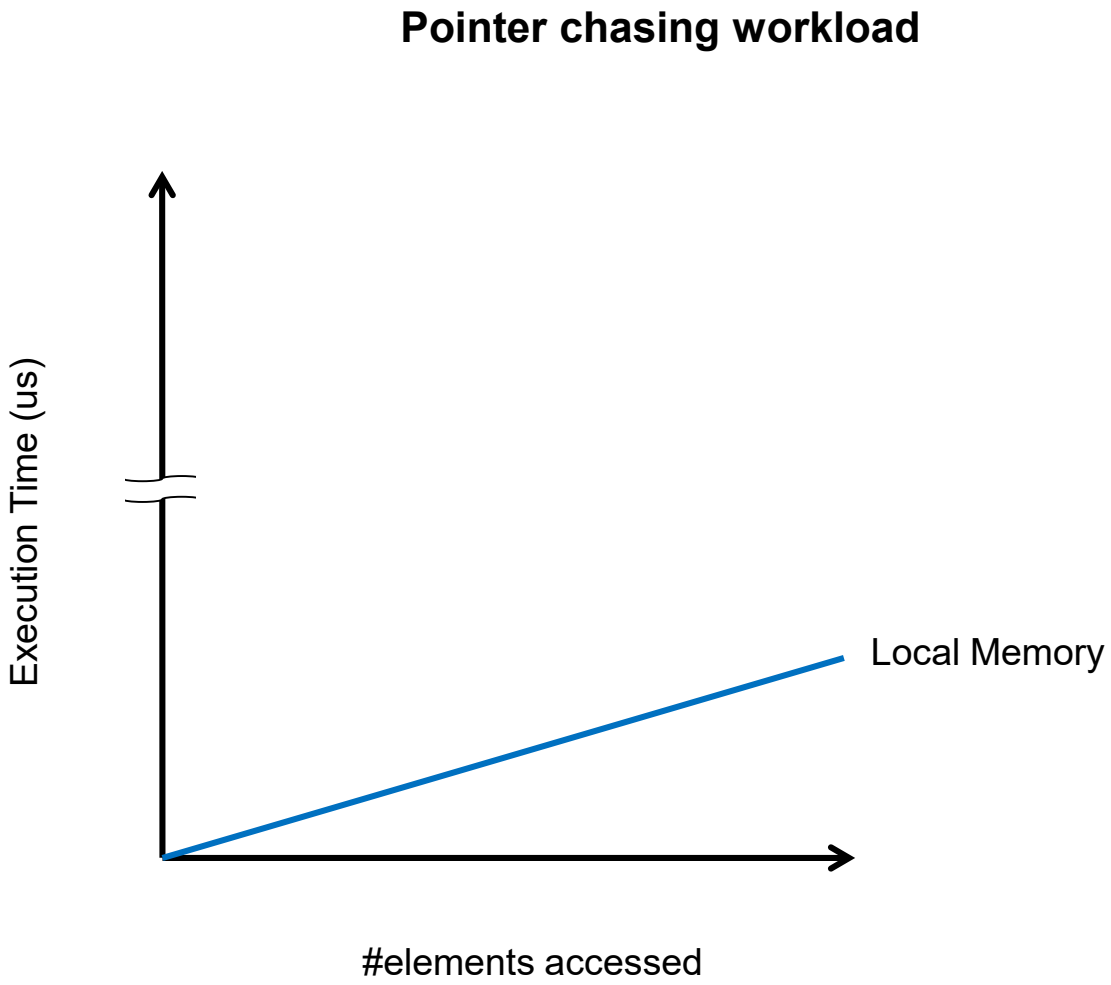
Configuration	Median Latency
All in local memory	1.0x
Index local, vectors far	1.75x
All data far	2.36x

Taxonomy of far memory performance mitigation techniques

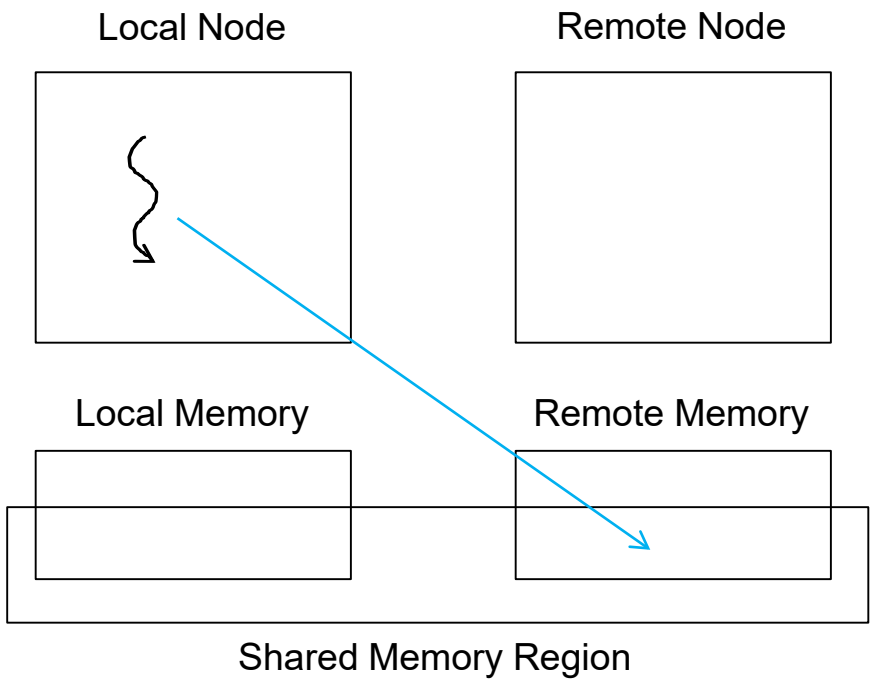


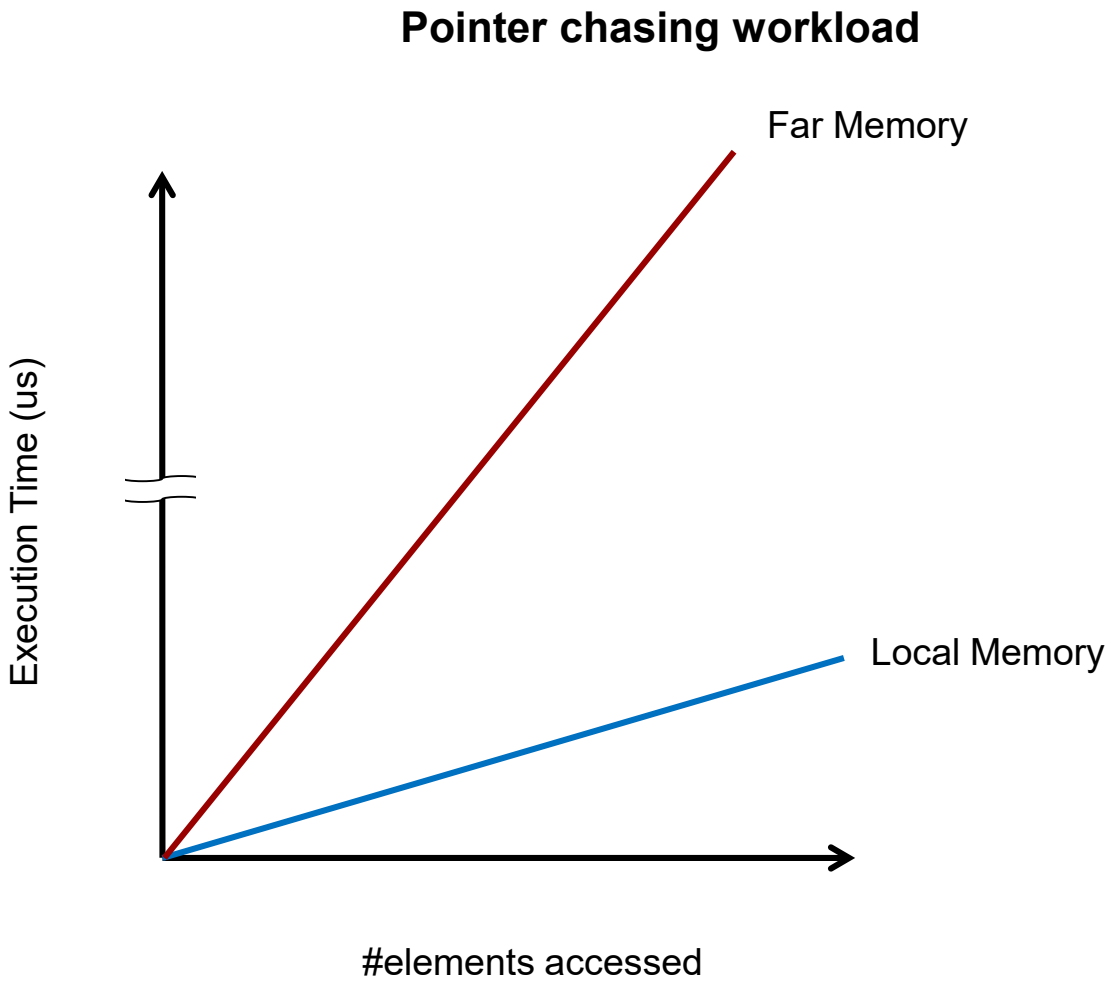
Overheads of different far memory access techniques



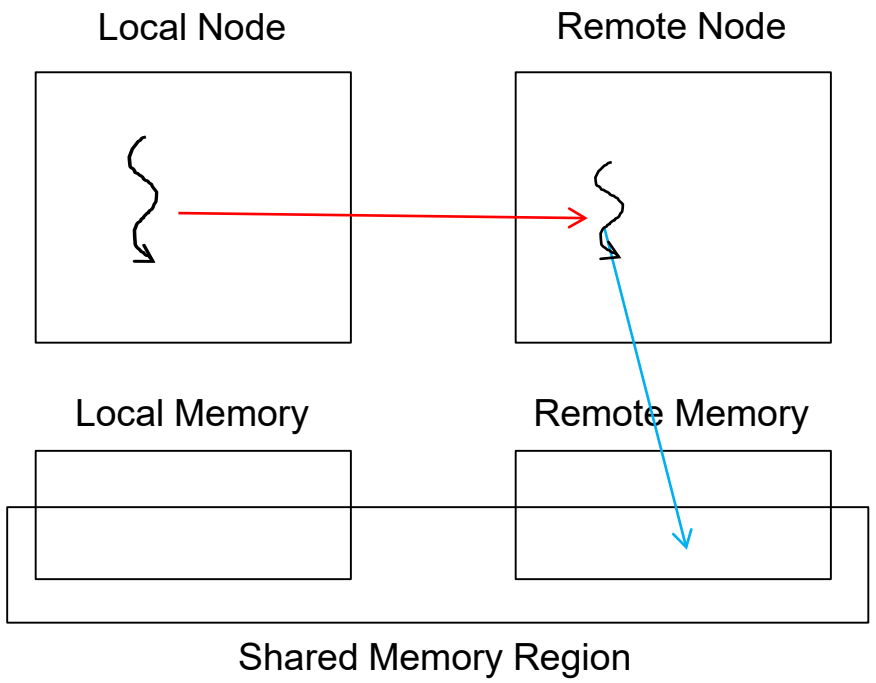


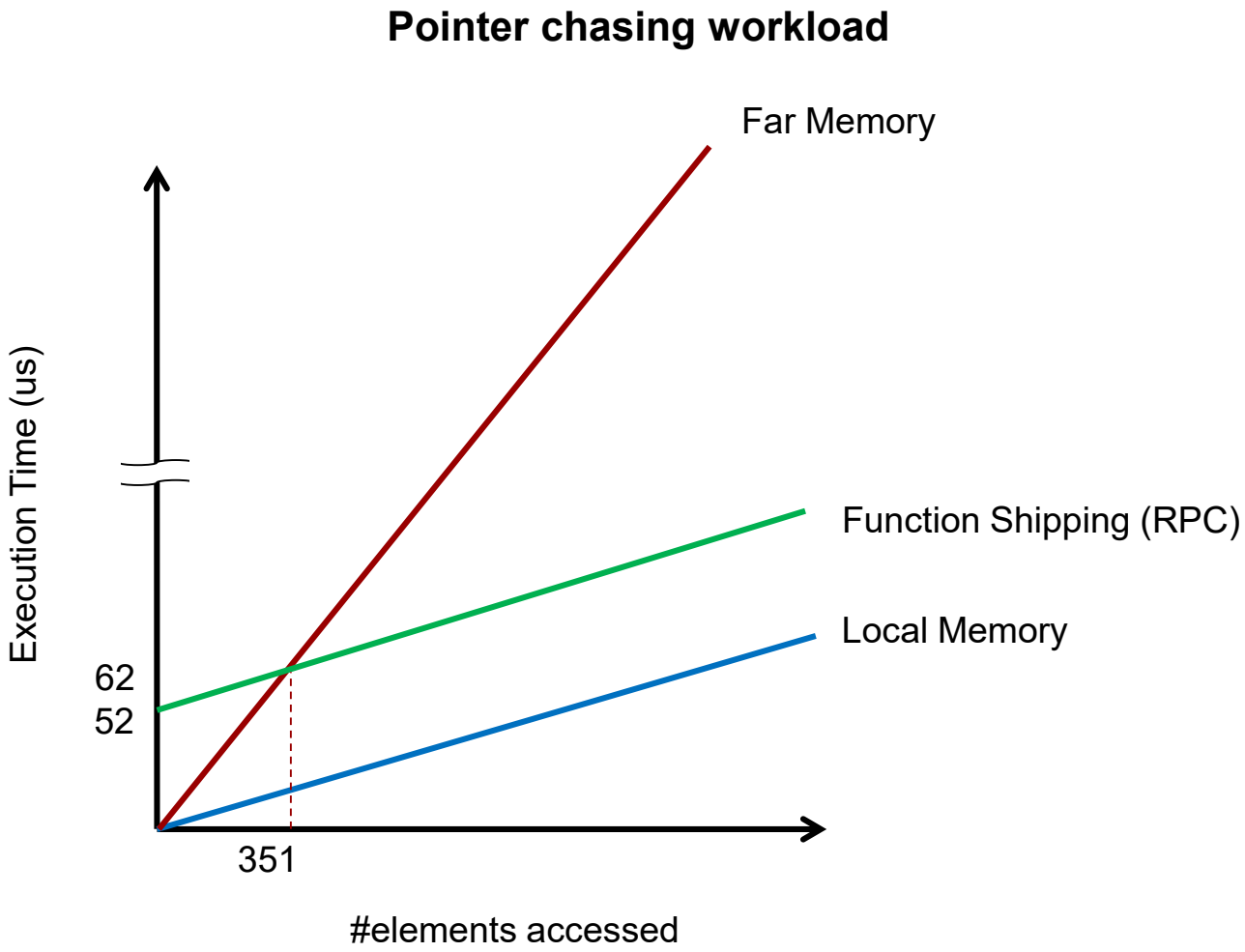
Overheads of different far memory access techniques



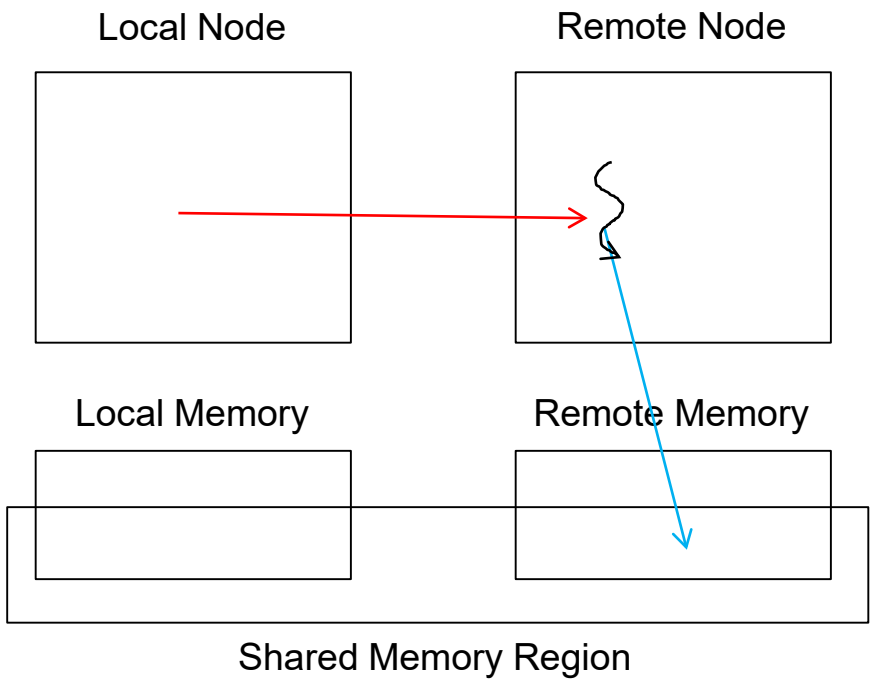


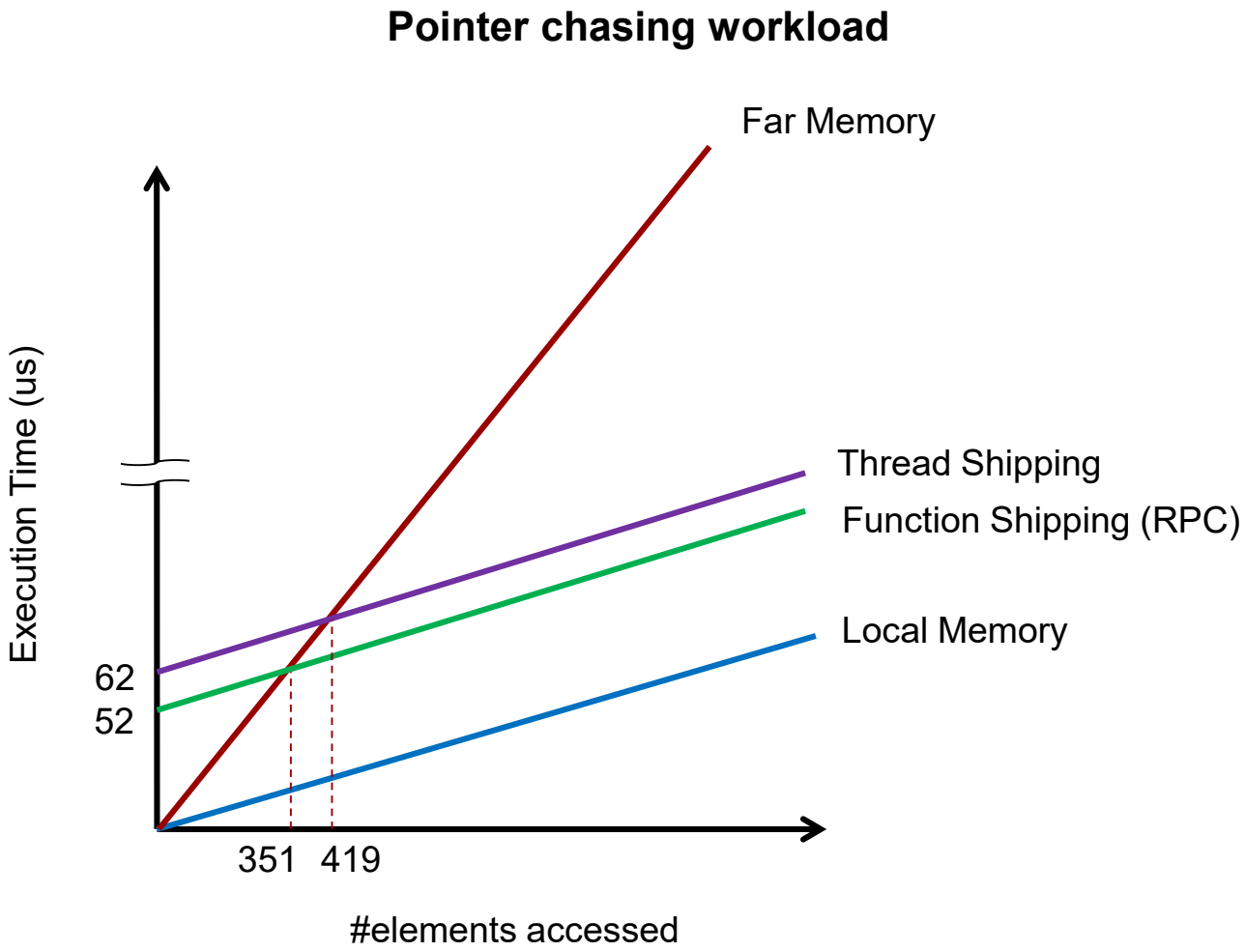
Overheads of different far memory access techniques



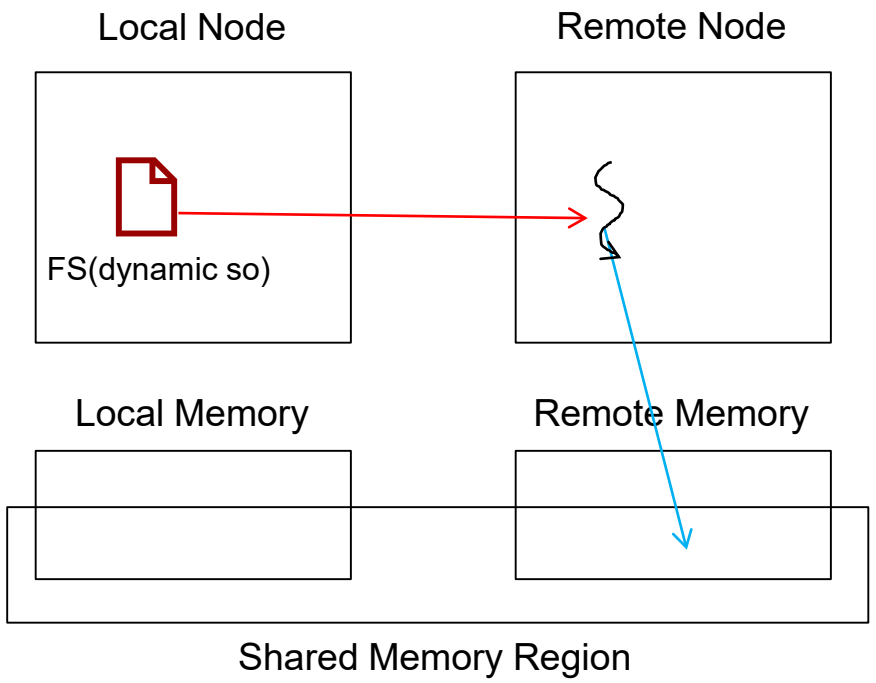


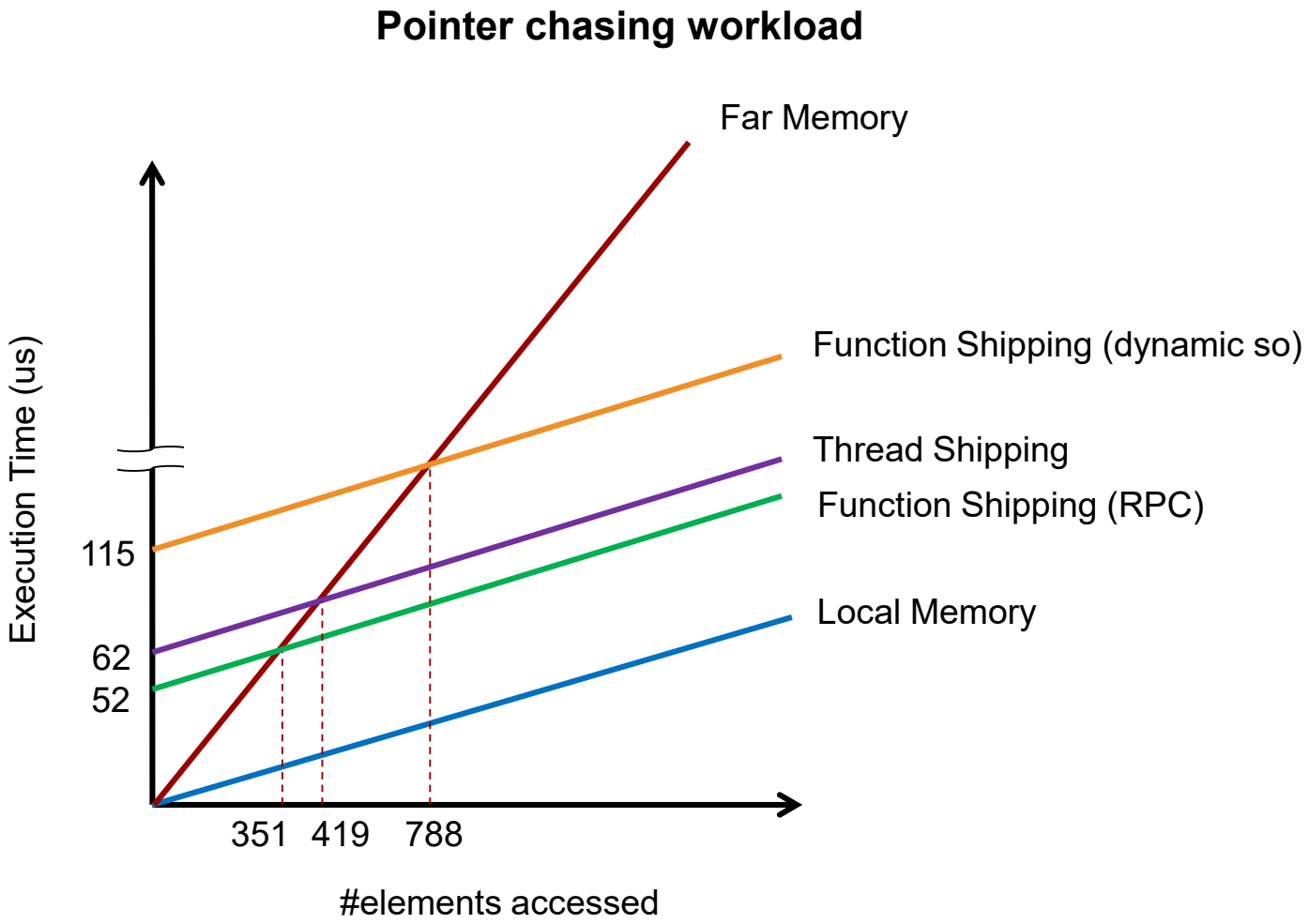
Overheads of different far memory access techniques



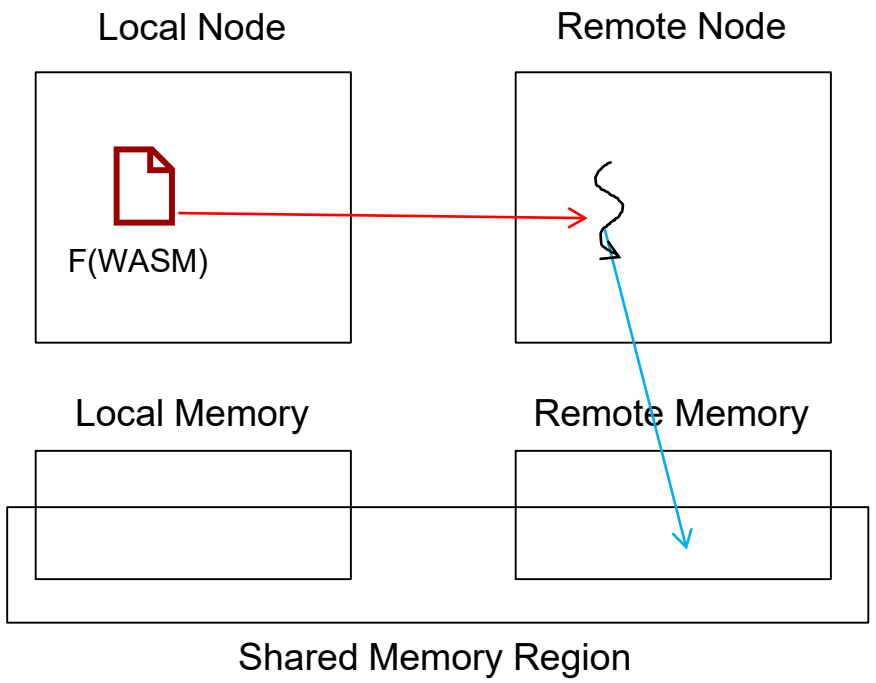


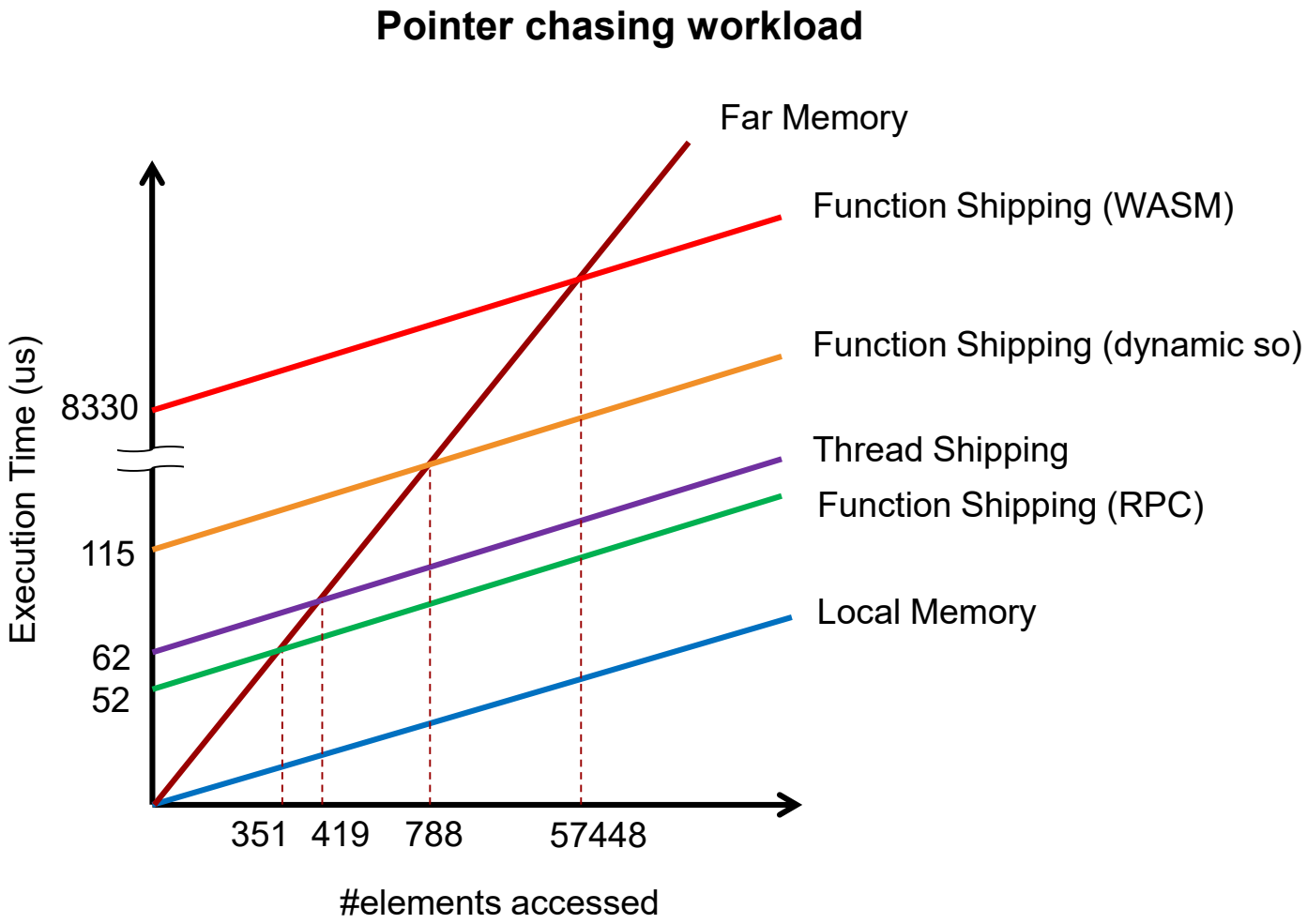
Overheads of different far memory access techniques



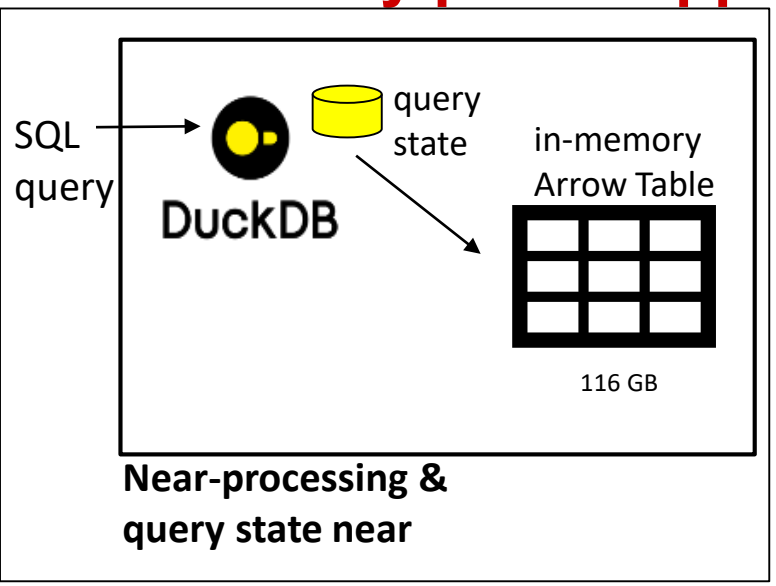


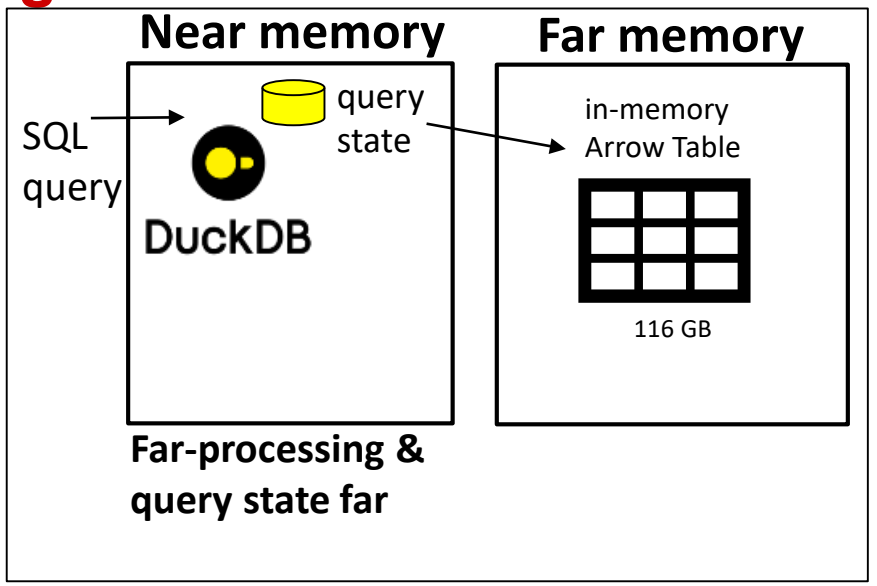
Overheads of different far memory access techniques

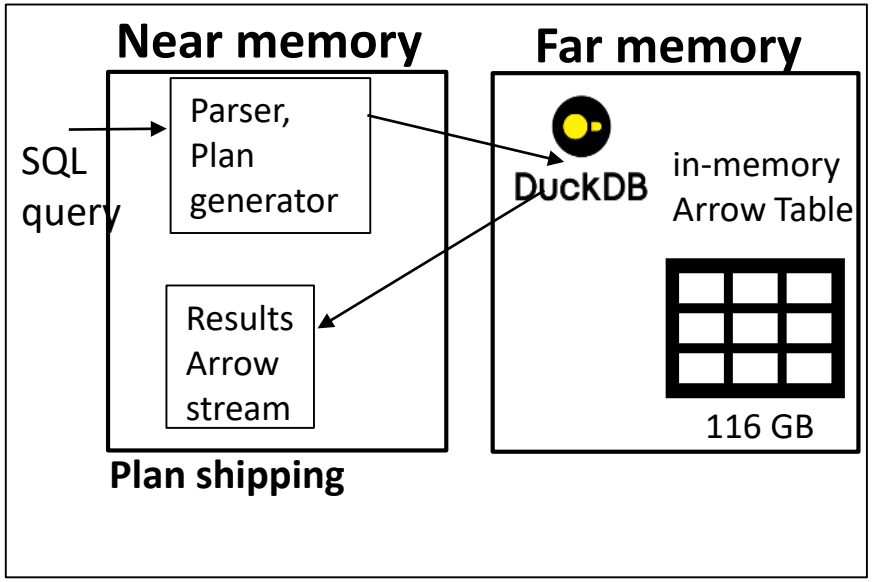




Query plan shipping







selecti vity	# groups	End-to-end time			Speedup over remote	
		near-processing	Plan shipping	far-processing	Local	plan shipping
4%	1	1.8s	1.8s	3.4s	**1.9x**	**1.9x**
4%	1,000	1.8s	1.8s	3.4s	1.9x	1.9x
4%	250,000	2.4s	2.8s	3.9s	1.6x	1.4x
4%	2,499,754	2.8s	14.9s	4.6s	1.6x	0.3x

Ongoing questions and Conclusions

Questions

- What is the minimum useful abstraction for code?
- Is there hardware that can help accelerate the migration of execution between cores/sockets/nodes/racks?
- What kind of mechanisms can we use to identify when it is appropriate to pay the overhead of shipping?

- What kind of software and system architectures make the most sense for far memory systems?

Conclusions

1. Memory is just too far away
2. We've spent a lot of time moving data to compute, but we need more emphasis on moving compute to data