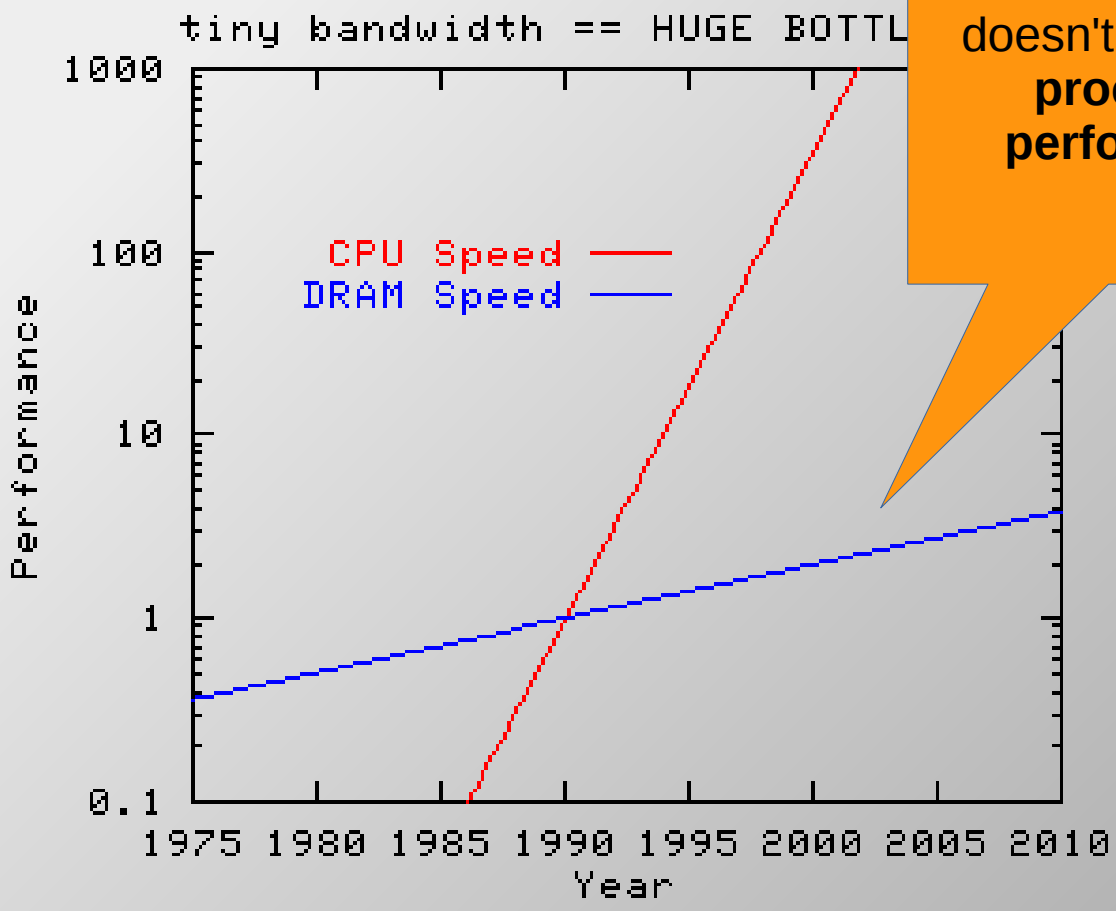
Fast Iterative Graph Computation with Block Updates

Xie, *et al.*
(Proceedings of the Very Large Database Endowment, 2013)

Review by Matthew Huxtable
R212: 13th November 2014

The problem

The problem

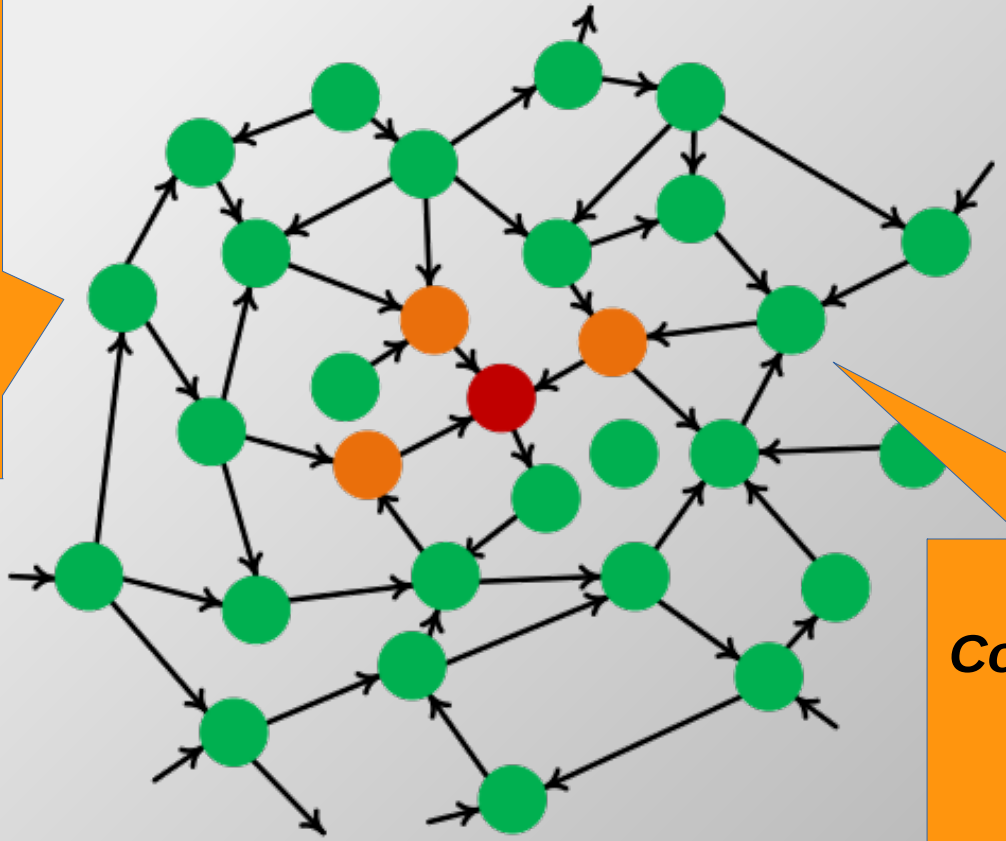


Memory access bandwidth doesn't scale with **processor performance**

Image reproduced from STREAM project website
http://www.cs.virginia.edu/stream/ref.html

The problem

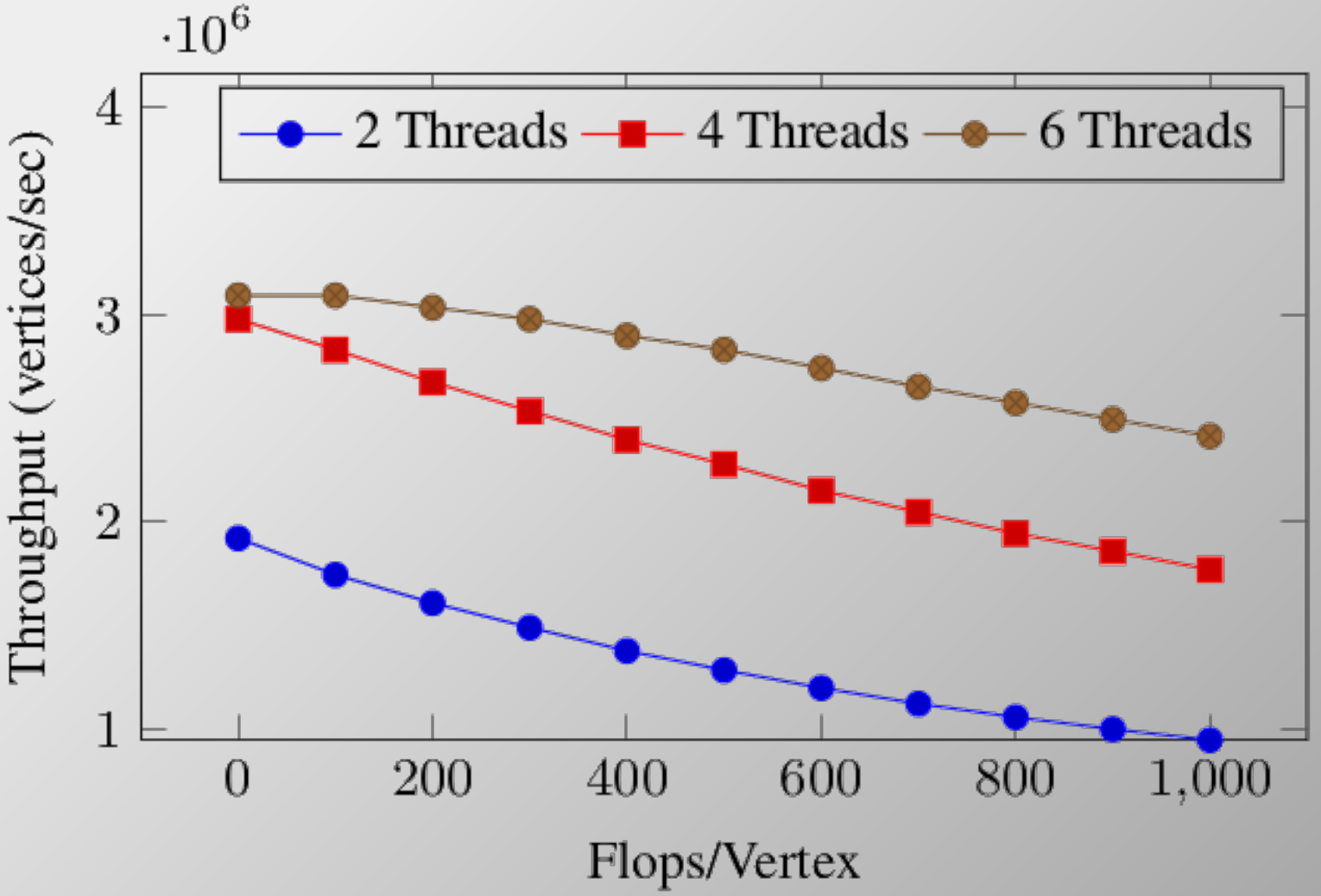
Vertex-centric computation performance poor



Computationally light **algorithms** suffer the most

(i.e. the common ones: PageRank shortest paths, SCC, etc.)

The problem
(as seen in practice)



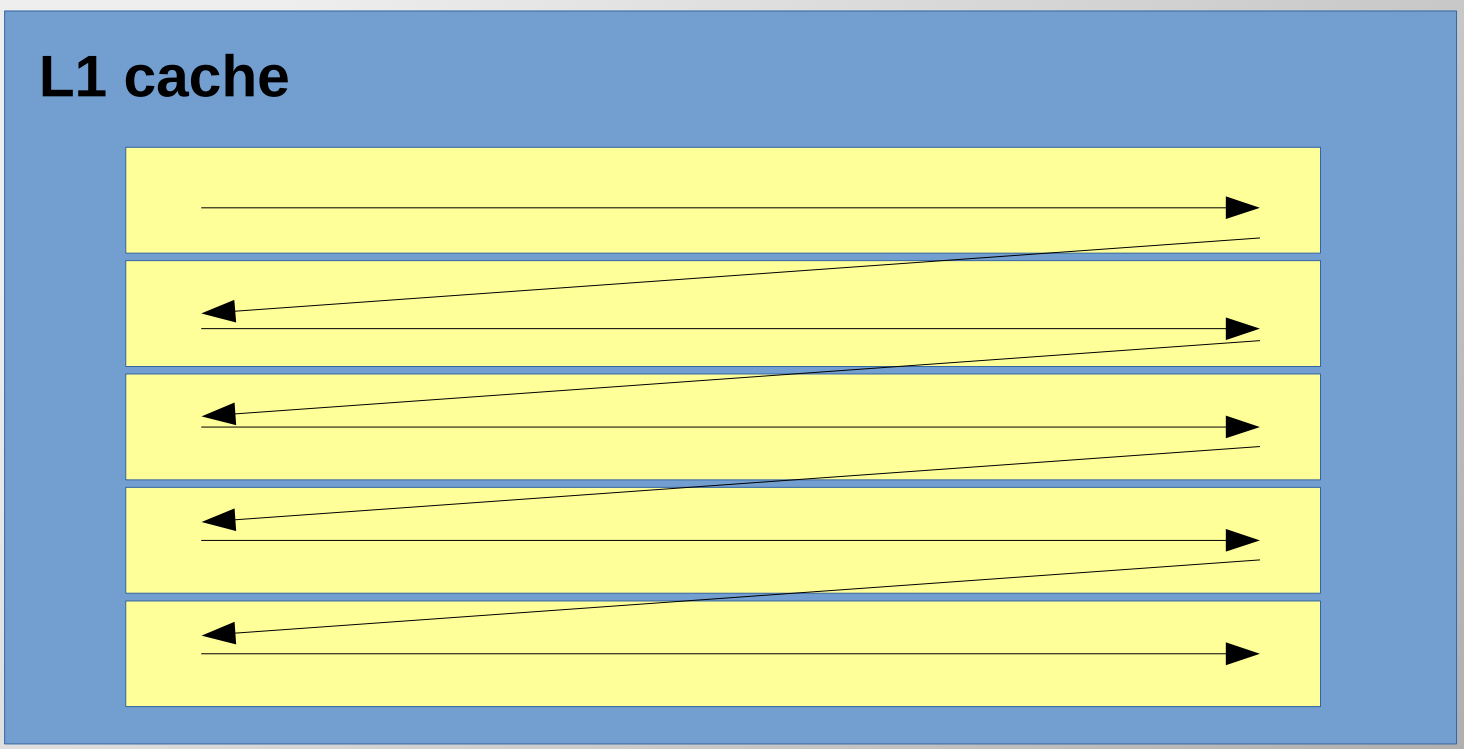
The novel computation model

The novel computation model

(using cache blocking)

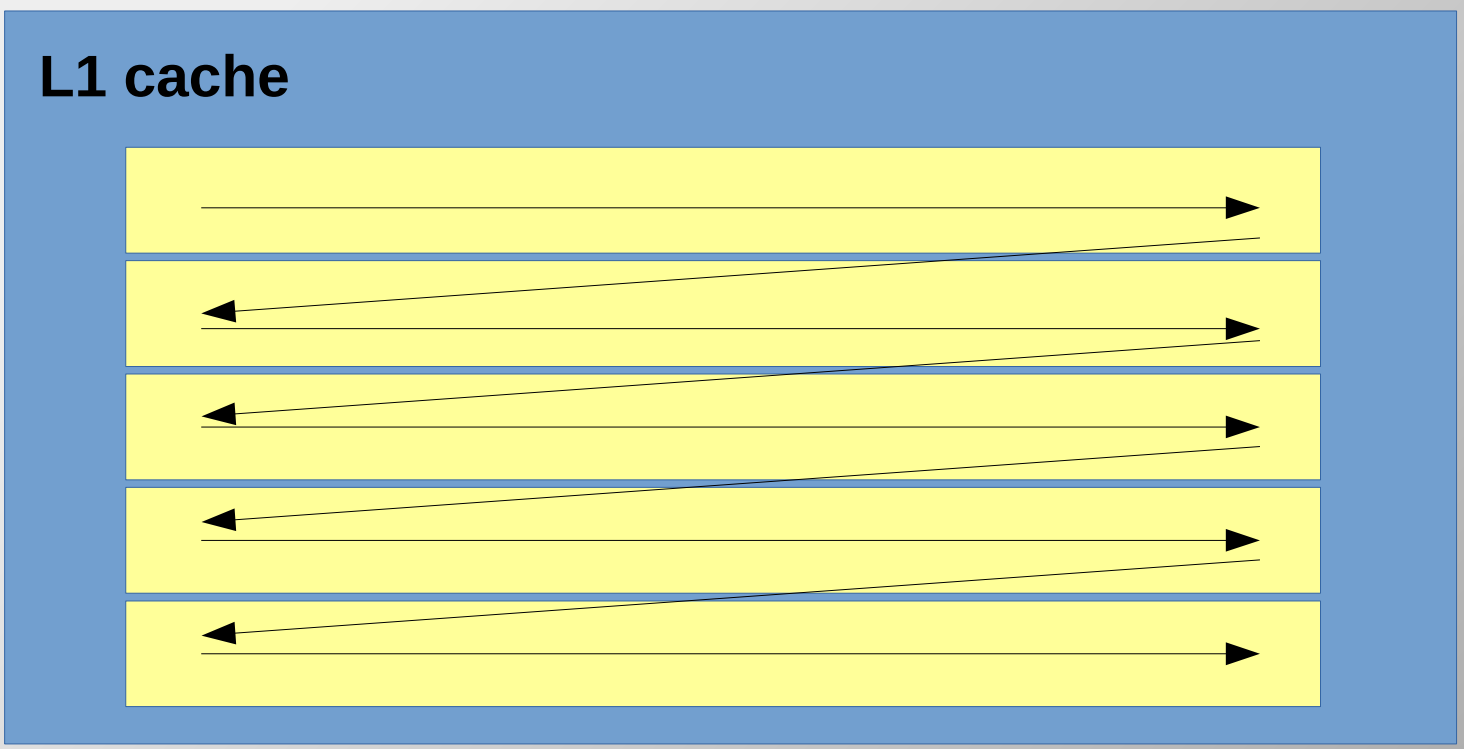
The novel computation model

- Process updates in cache line granularity



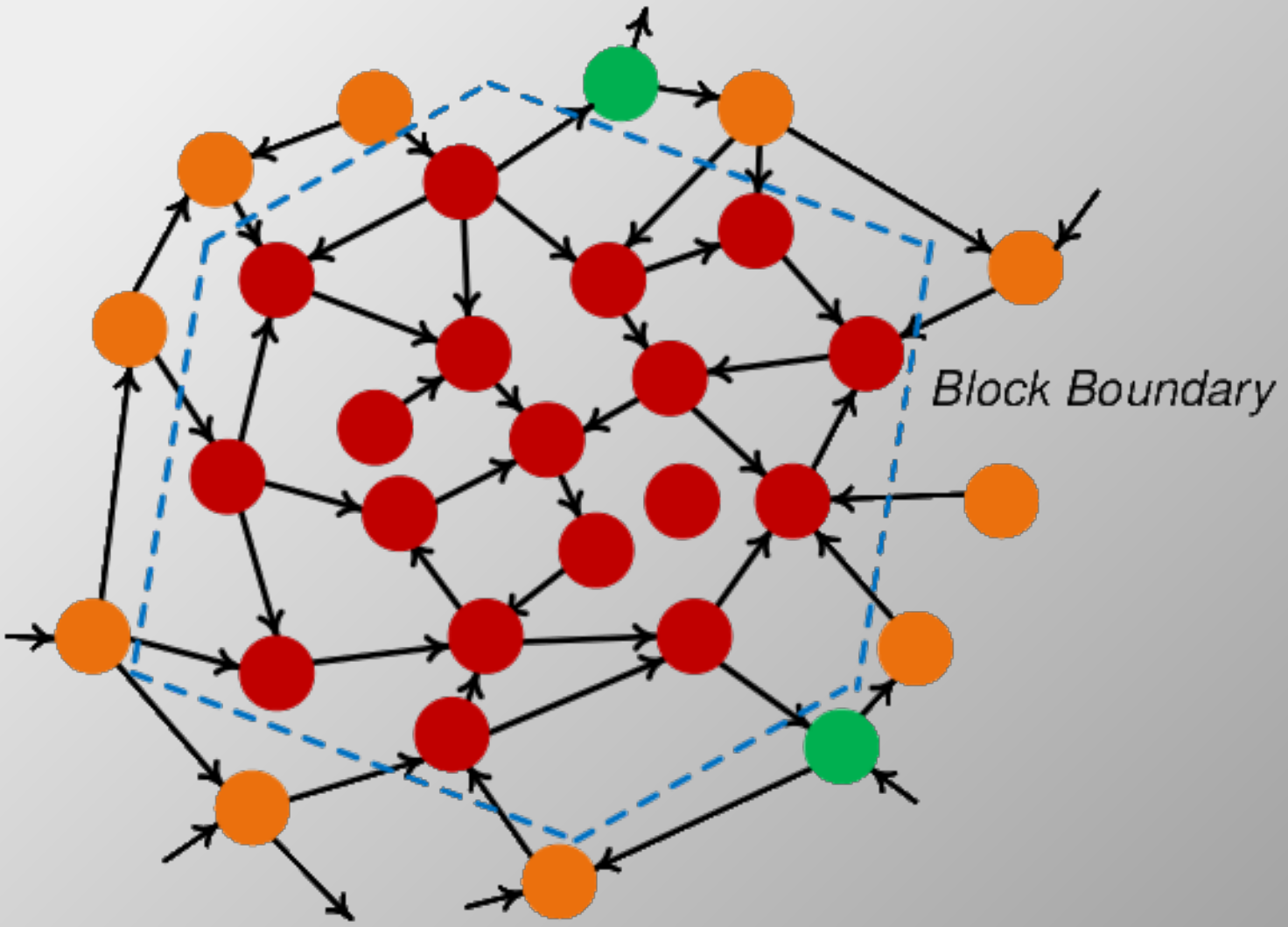
The novel computation model

- Process updates in cache line granularity



Keep the vertex-centric programming abstraction. (How?)

In practice

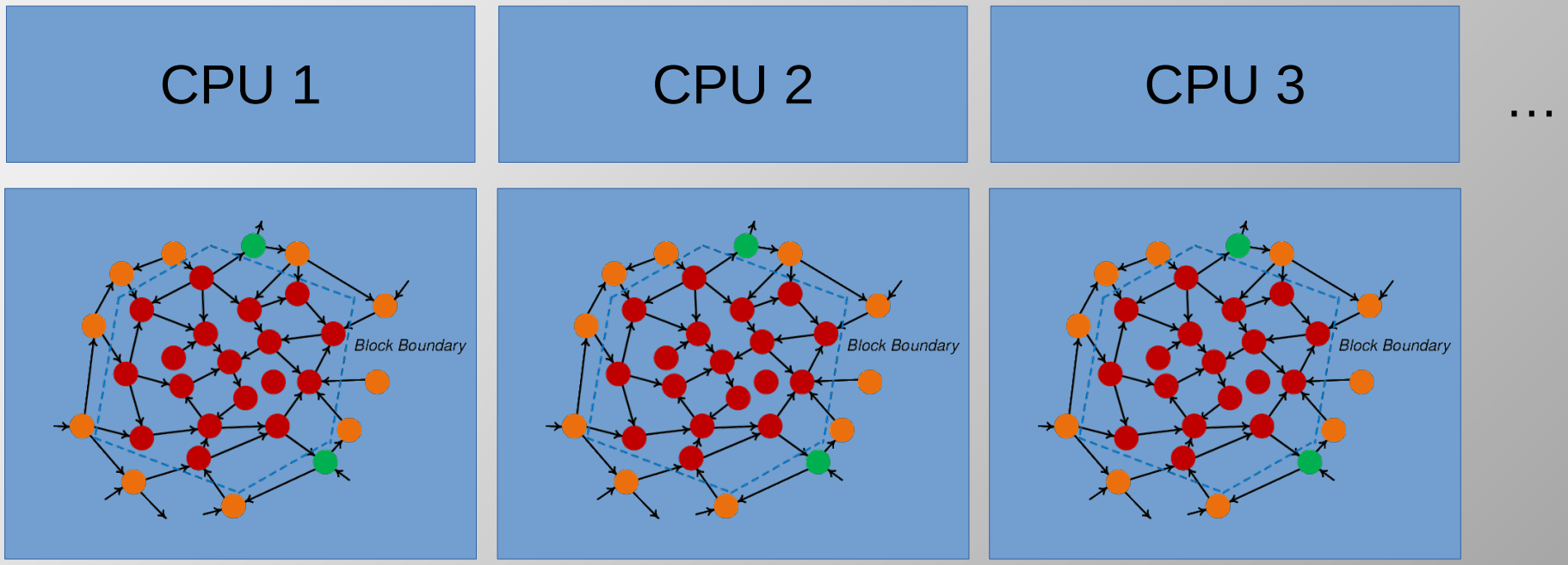


In practice

- Dual layer scheduler (Eager, Prior)

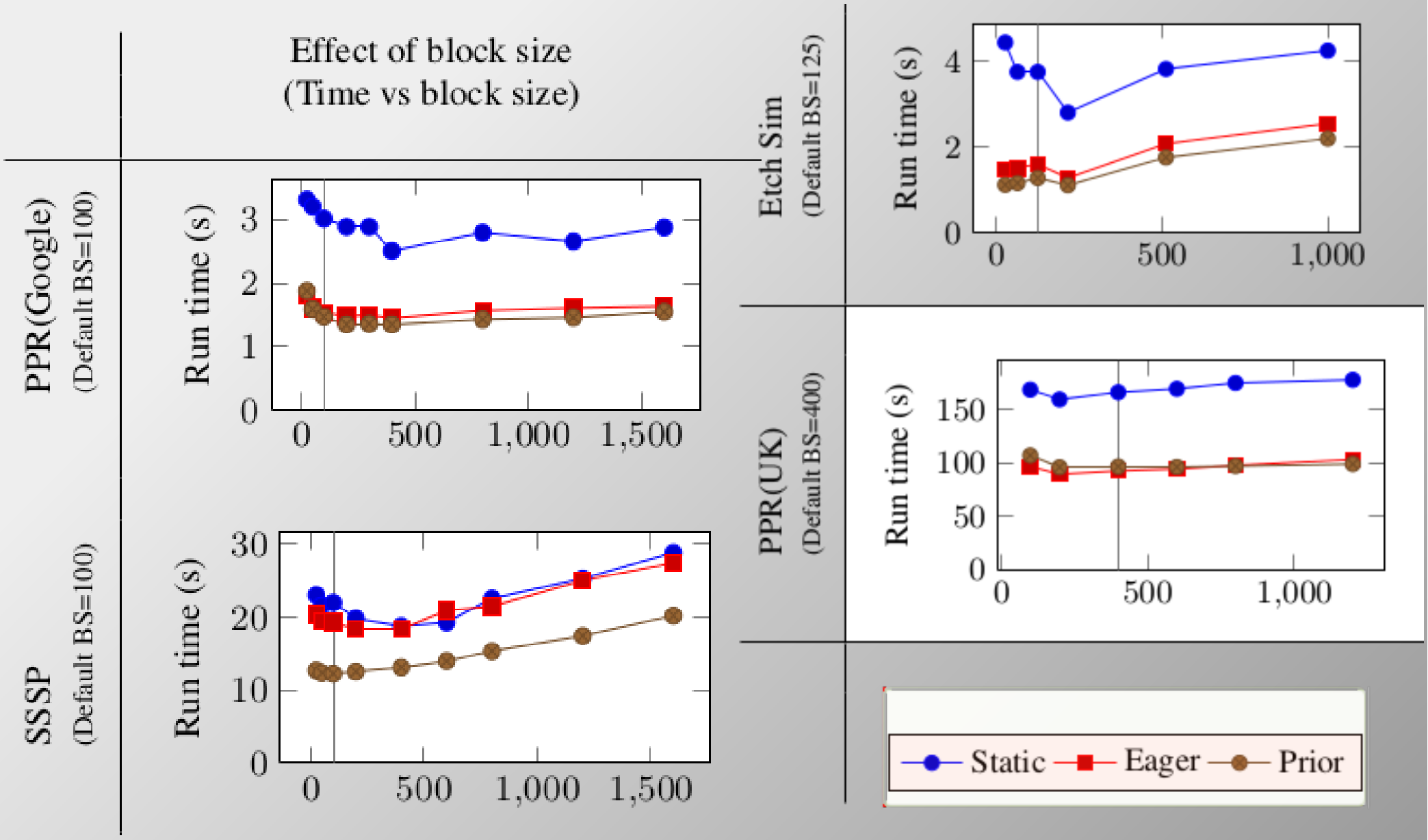
In practice

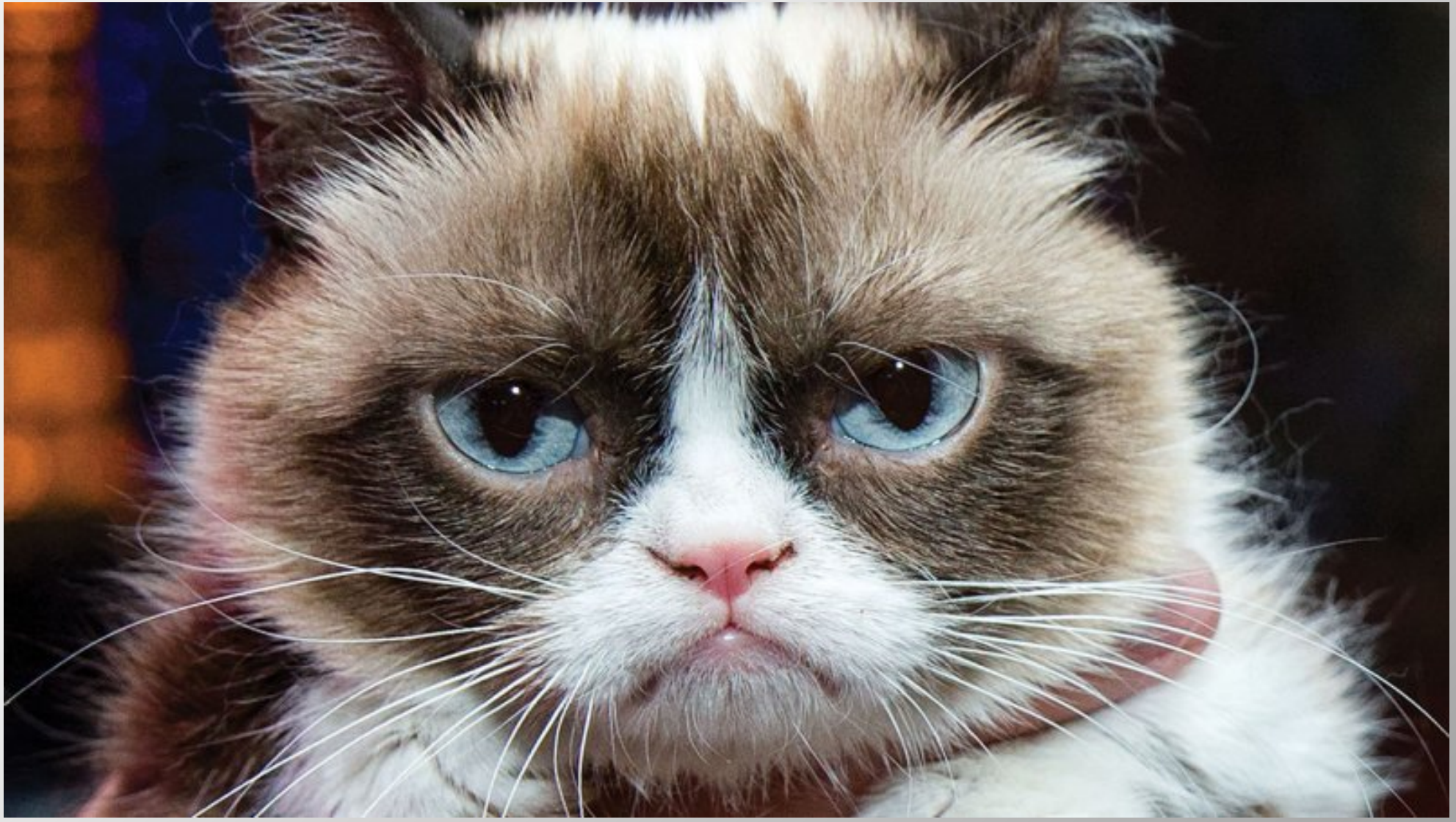
- Dual layer scheduler (Eager, Prior)
- Multiversion concurrency control



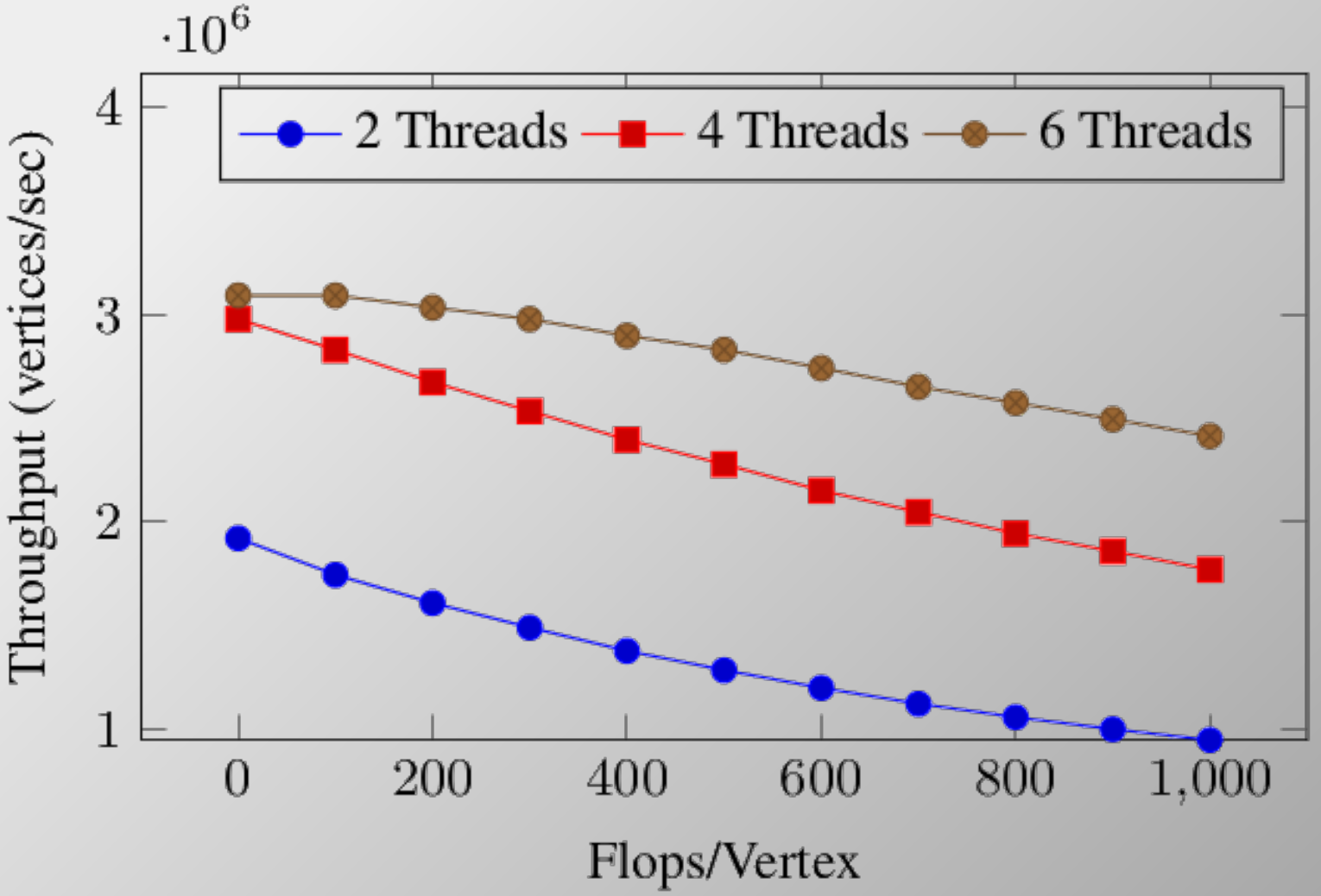
The evaluation

Scheduling policies

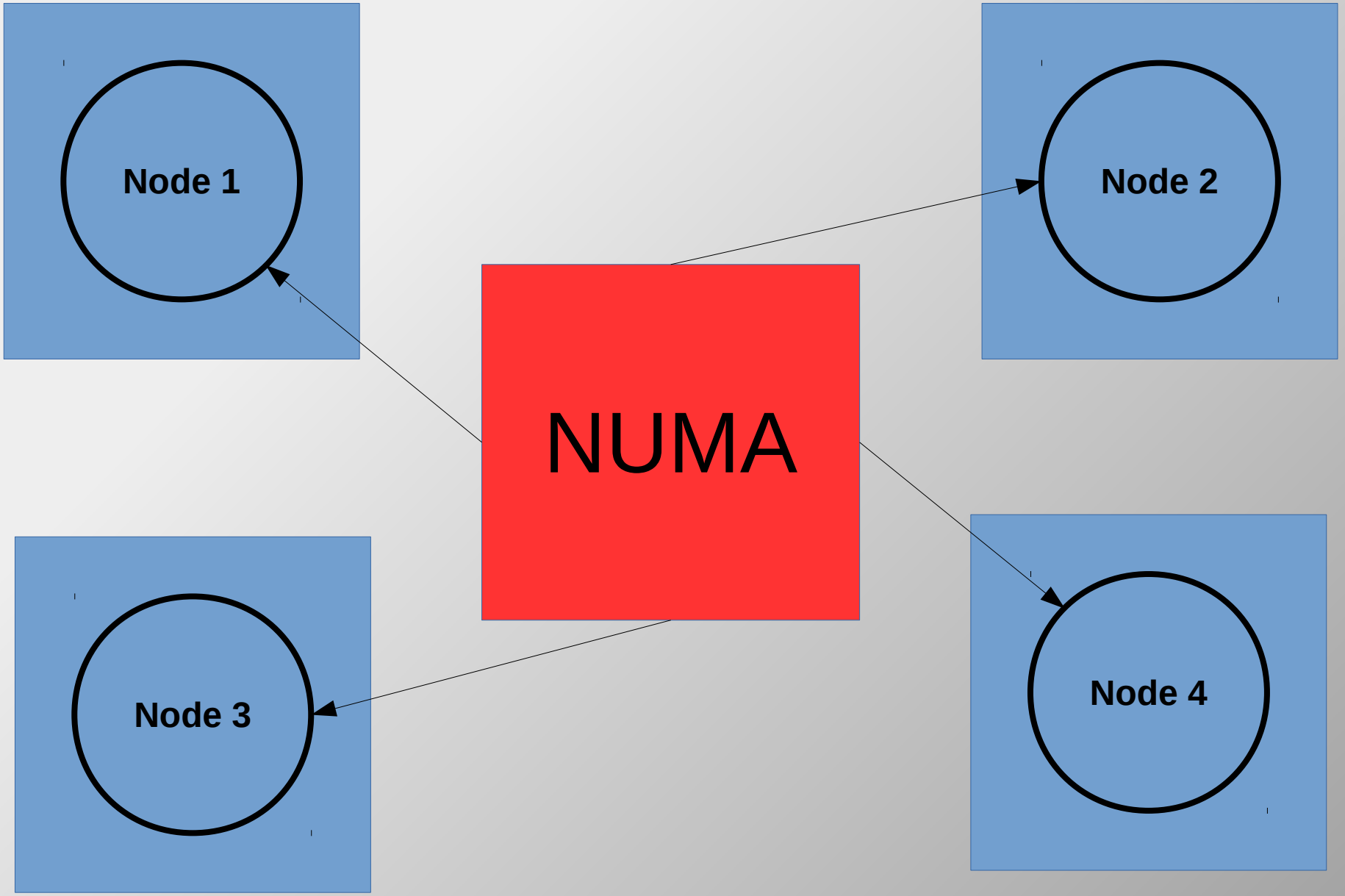




~~The problem~~
What problem?



How do we pick vertices to form blocks?


Node 1
Node 2
NUMA
Node 3
Node 4

Conclusions

- Block-parallel graph-centric framework
- Dynamically scheduled blocks containing >1 vertex per block
- Better cache interaction over alternatives → faster programs (dubious?)
- Useful in common cases:
  Dijkstra, SCC, PageRank

Conclusions

- Block-parallel graph-centric framework
- Dynamically scheduled blocks containing >1 vertex per block
- Better cache interaction over alternatives → faster programs (dubious?)
- Useful in common cases:
  Dijkstra, SCC, PageRank
- Verdict on paper...