

# (THE LANDSCAPE OF) PARALLEL GRAPH PROCESSING: A VIEW FROM HOLLAND

Ana Lucia Varbanescu, University of Amsterdam, The Netherlands  
[a.l.varbanescu@uva.nl](mailto:a.l.varbanescu@uva.nl)

With data, work, and (some) slides from a whole team:  
Merijn Verstraaten, Ate Penders, Yong Guo, Alexandru Iosup, and Dick Epema.

What to do when your graphs get out of control ?

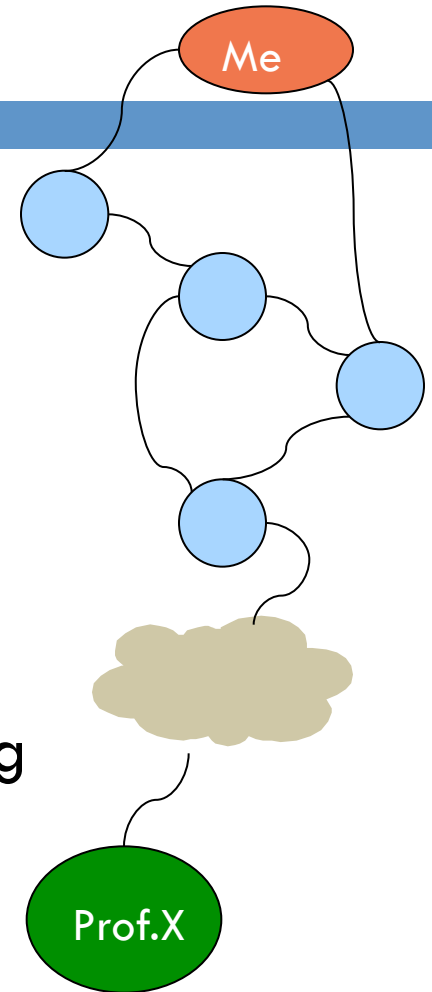
# In February 2015\*, LinkedIn...

- ... has exceeded 347 Million users ...
  - ... 56% male, 44% women ...
  - ... in over 200 countries ...
  - ... more than 70% outside of US.
- 
- Most users are “motivated” and “responsible”
  - Using a picture increases a user’s profile by 11x
  - CEOs have 900+ connections, on average



# Classical analytics

- ❑ Statistics
  - ▣ “How many connections do I have?”
- ❑ Traversing
  - ▣ “How can I reach Prof. X?”
- ❑ Querying
  - ▣ “Find all professionals in Graph Processing around San Jose.”
- ❑ Mining
  - ▣ “Find the most influential CS researcher in Amsterdam.”



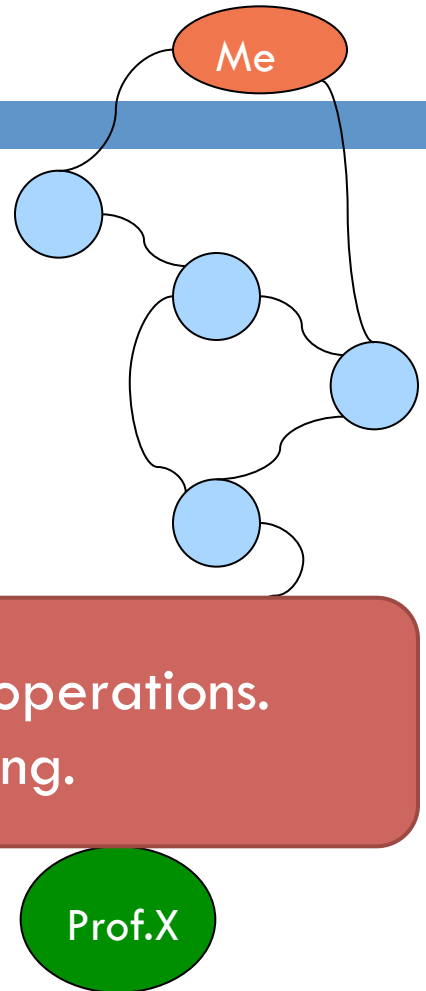
# Classical analytics

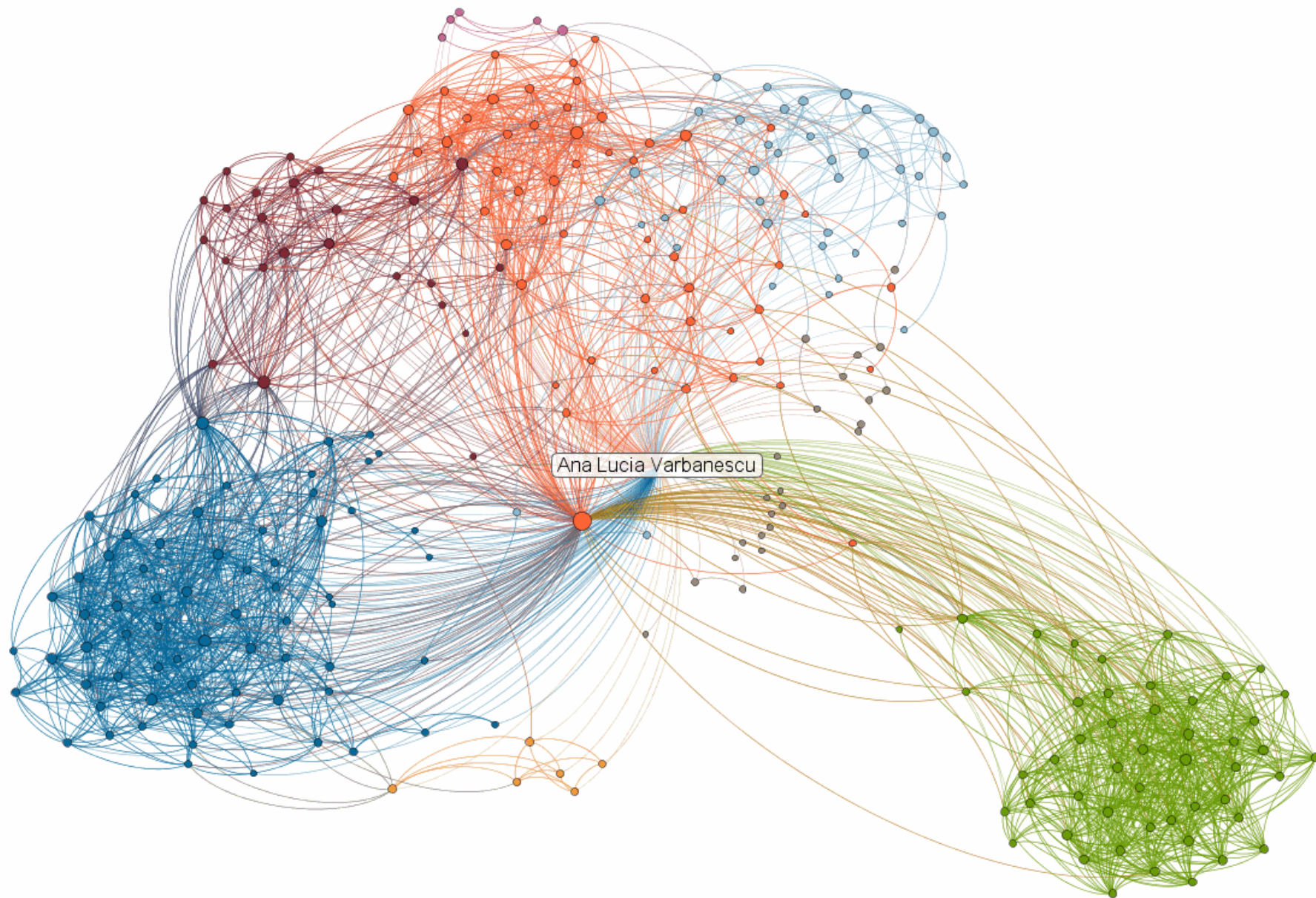
- Statistics
  - ▣ “How many connections do I have?”
- Traversing
  - ▣ “How can I reach Prof. X?”

No textbook algorithms exist for some of these operations.  
If they exist, they probably need changing.

around San Jose.

- Mining
  - ▣ “Find the most influential CS researcher in Amsterdam.”







# Your network is so large...

Sorry, but your network is too large to be computed, we are working to increase the limit, stay tuned!

# Large Scale Graph Processing

- Graph processing is (very) **data-intensive**
  - ▣ 10x larger graph => 100x or 1000x slower processing
- Graph processing becomes (more) **compute-intensive**
  - ▣ More complex queries => ?x slower processing
- Graph processing is (very) **dataset-dependent**
  - ▣ Unfriendly graphs => ?x slower processing

High performance enables *larger graphs* and support for *more complex analytics*.

# More performance? Many-cores!



Works for Top500!  
Virtually all machines are multi-cores,  
more than 10% are accelerated



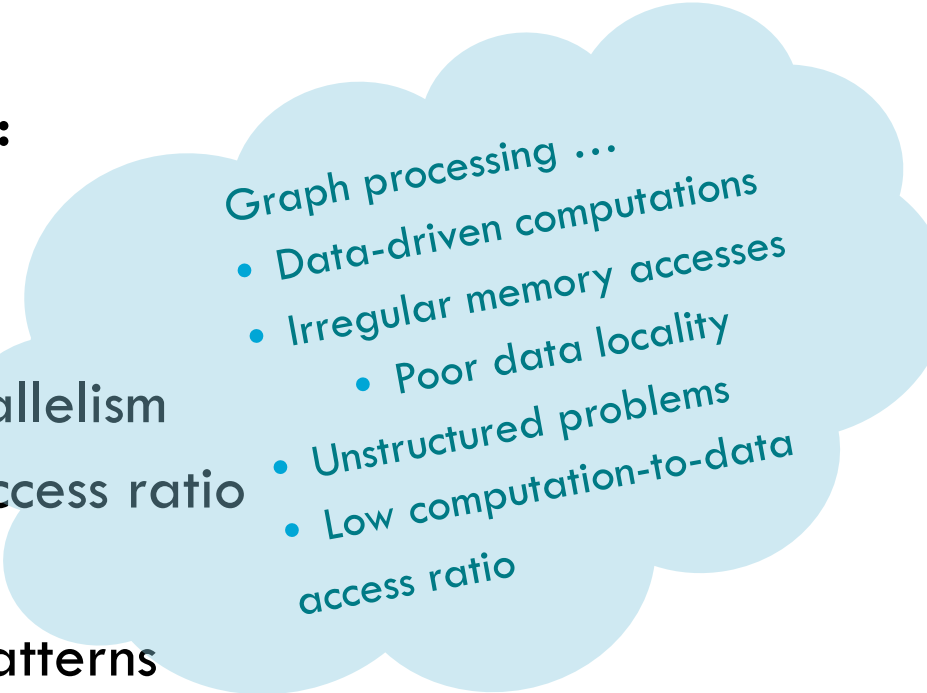


# Graph500 $\neq$ Top500 !

	Machine	Site	Nodes	Cores	Scale	GTEPS
1	DOE/NNSA/LLNL Sequoia (IBM - BlueGene/Q)	Lawrence Livermore National Laboratory	98304	1572864	41	<b>23751</b>
2	K computer (Fujitsu - Custom supercomputer)	RIKEN Advanced Institute for Computational Science (AICS)	82944	663552	40	<b>19585</b>
3	DOE/SC/Argonne National Laboratory Mira (IBM - BlueGene/Q)	Argonne National Laboratory	49152	786432	40	<b>14982</b>
4	JUQUEEN (IBM - BlueGene/Q)	Forschungszentrum Juelich (FZJ)	16384	262144	38	<b>5848</b>
5	Fermi (IBM - BlueGene/Q)	CINECA	8192	131072	37	<b>2567</b>
6	Tianhe-2 (MilkyWay-2)	NUDT, Changsha, China	8192	196608	36	<b>2061</b>
7	Turing (IBM - BlueGene/Q)	<div>Number 1 in Top500</div>	4096	65536	36	<b>1427</b>
8	Blue Joule (IBM - BlueGene/Q)		4096	65536	36	<b>1427</b>
9	DIRAC (IBM - BlueGene/Q)		4096	65536	36	<b>1427</b>
10	Zumbrota (IBM - BlueGene/Q)		4096	65536	36	<b>1427</b>

# A clash?

- Many-cores have emerged to improve performance by using massive parallelism.
- Performance gain in theory:  
N cores  $\Rightarrow$  N times faster
- For this, we need:
  - ▣ massive (multi-layered) parallelism
  - ▣ high computation-to-data access ratio
  - ▣ high data locality
  - ▣ structured, regular access patterns

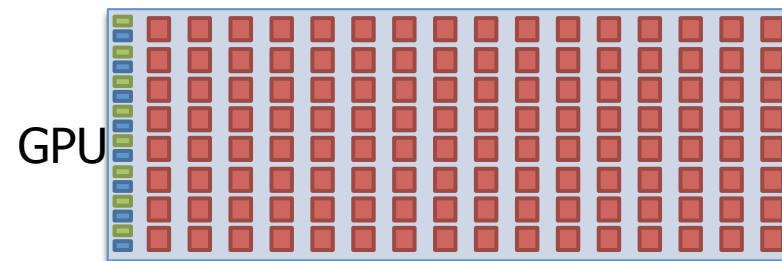
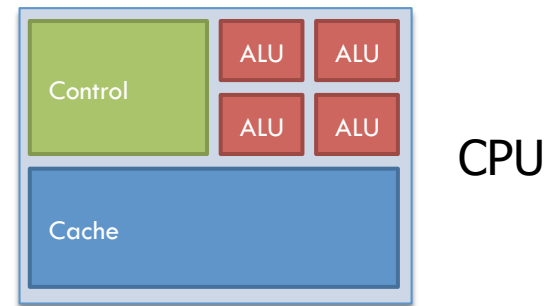


Graph processing ...

- Data-driven computations
- Irregular memory accesses
  - Poor data locality
- Unstructured problems
- Low computation-to-data access ratio

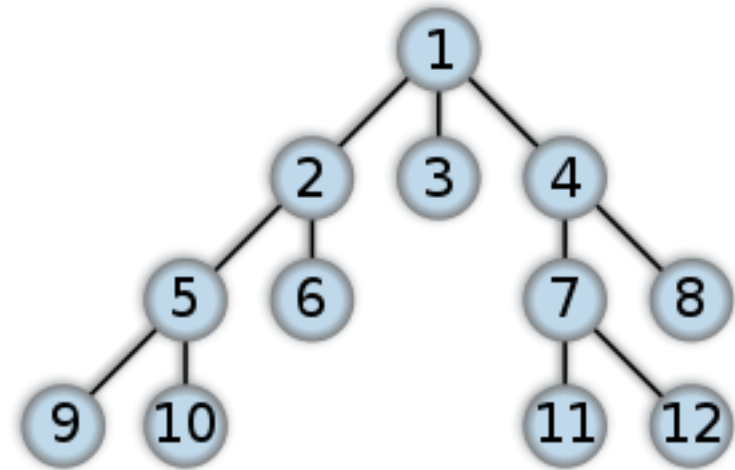
# Experiment 1: CPU and/or GPU \*

- Question:
  - ▣ Which multi-/many-core architectures are suitable for graph processing?
- Setup:
  - ▣ Three parallelized algorithms
  - ▣ Use different graphs
  - ▣ Use different hardware



# Algorithms: BFS→APSP→BC

- Graph traversal (Breadth First Search, BFS)
  - ▣ Traverses all vertices “in levels”
- All-Pairs Shortest Paths (APSP)
  - ▣ Repeat BFS for each vertex
- Betweenness Centrality (BC)
  - ▣ APSP once to determine paths
  - ▣ Bottom-up BFS to count paths
- Implementation in OpenCL\*
  - ▣ Same algorithm
  - ▣ CPU- and GPU-specific **tuning** applied



# Data sets & devices

	Abbreviation	Vertices	Edges	Diameter	Avg. Degree
Wikipedia Talk Network	WT	2,394,385	5,021,410	9	2,10
California Road Network	CR	1,965,206	5,533,214	850	2,81
Rodinia Graph 1M	1M	1,000,000	6,000,000	36	6,00
Stanford Web Graph	SW	281,903	2,312,497	740	8,20
EU Email Communication Network	EU	265,214	420,045	13	1,58
Star	ST	100,000	99,999	1	0,99
Chain	CH	100,000	99,999	99,999	1,00
Epinions Social Network	ES	75,879	508,837	13	6,70
Rodinia Graph 64K	64K	64,000	393,216	28	6,14
Wikipedia Vote Network	VW	7,115	103,689	7	14,57
Rodinia Graph 4K	4K	4000	25,356	19	6,38

## □ Devices

Intel(R) Xeon(R) CPU E5620 @ 2.40GHz

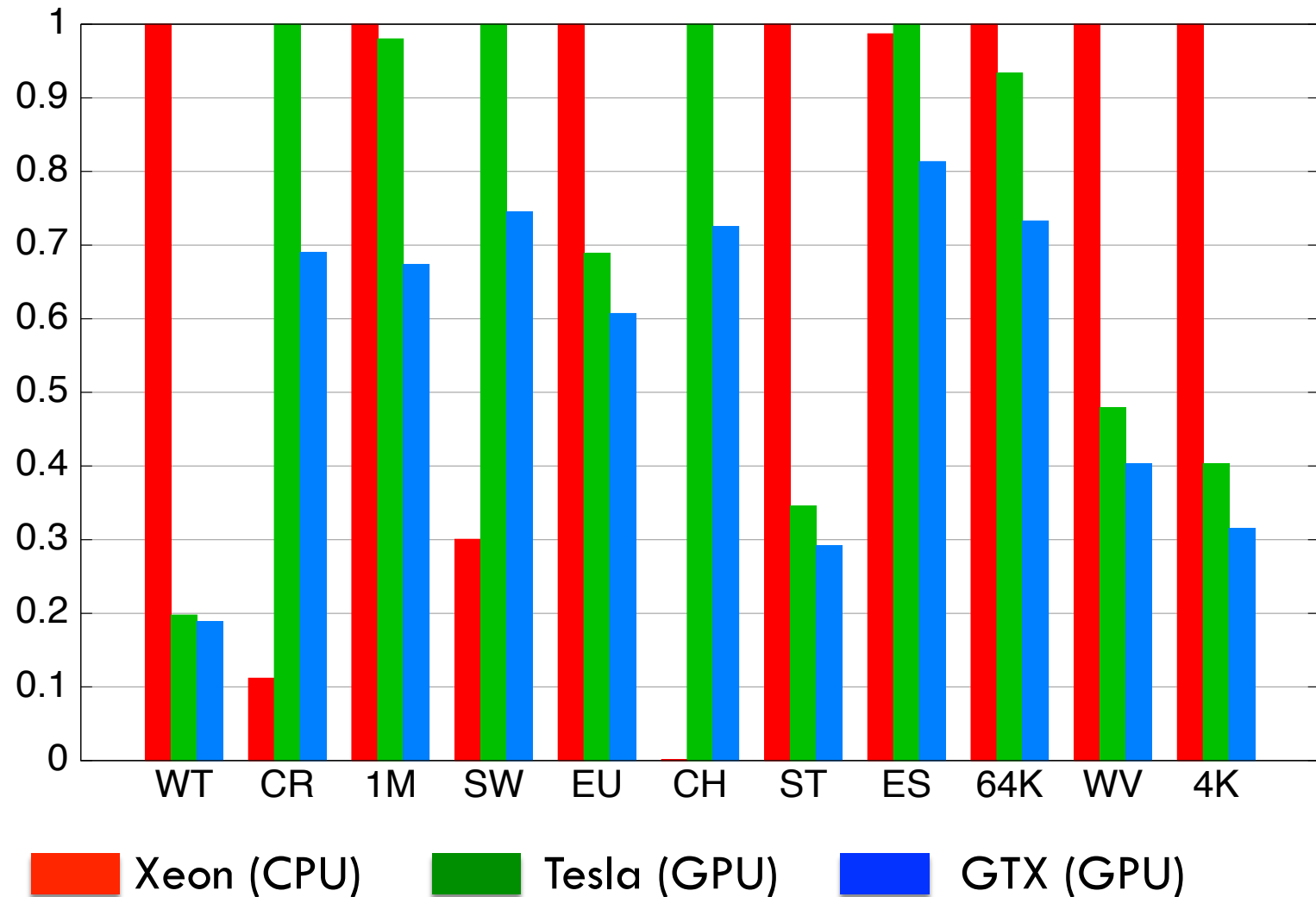
GeForce GTX 480

Tesla C2050 / C2070

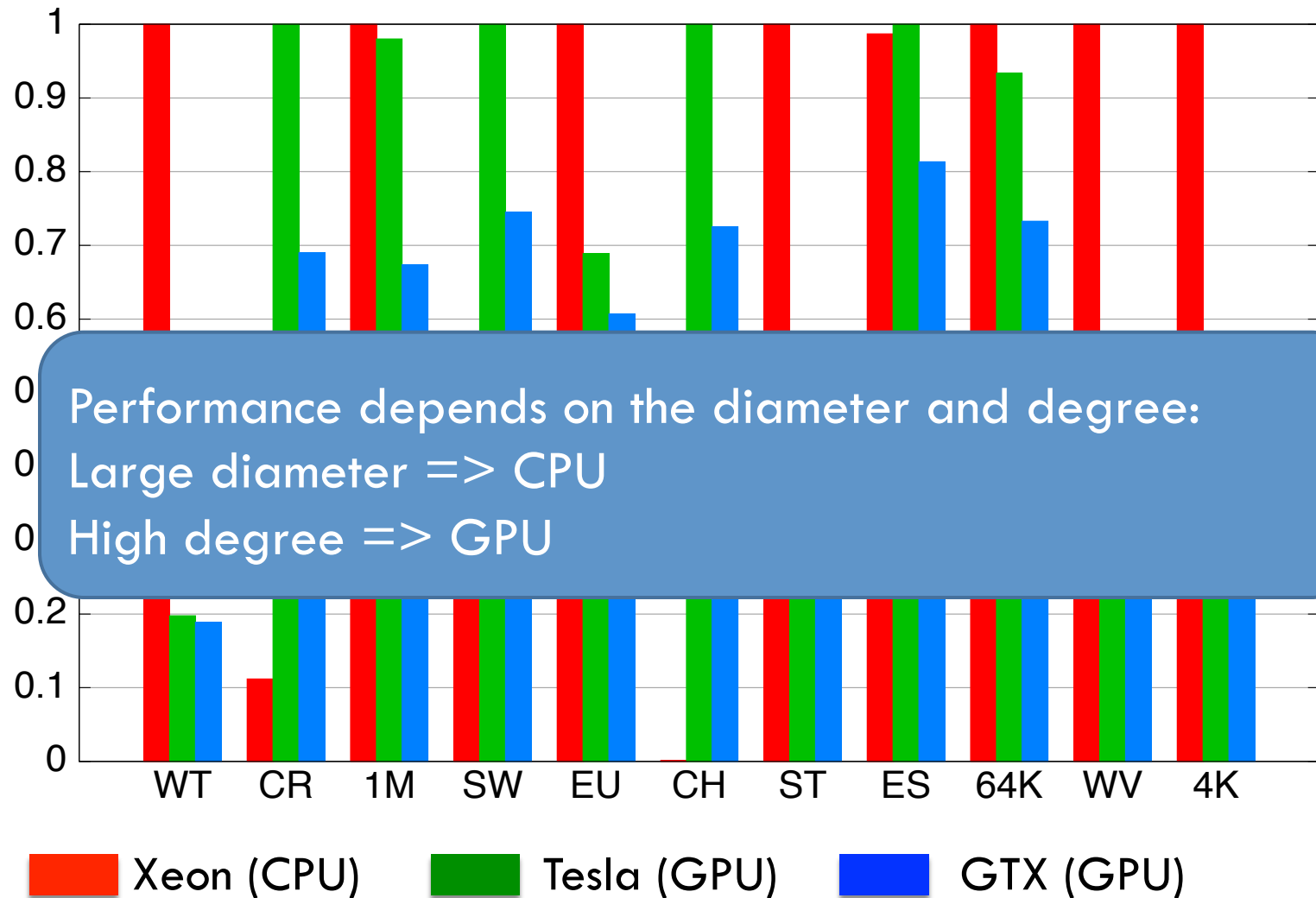




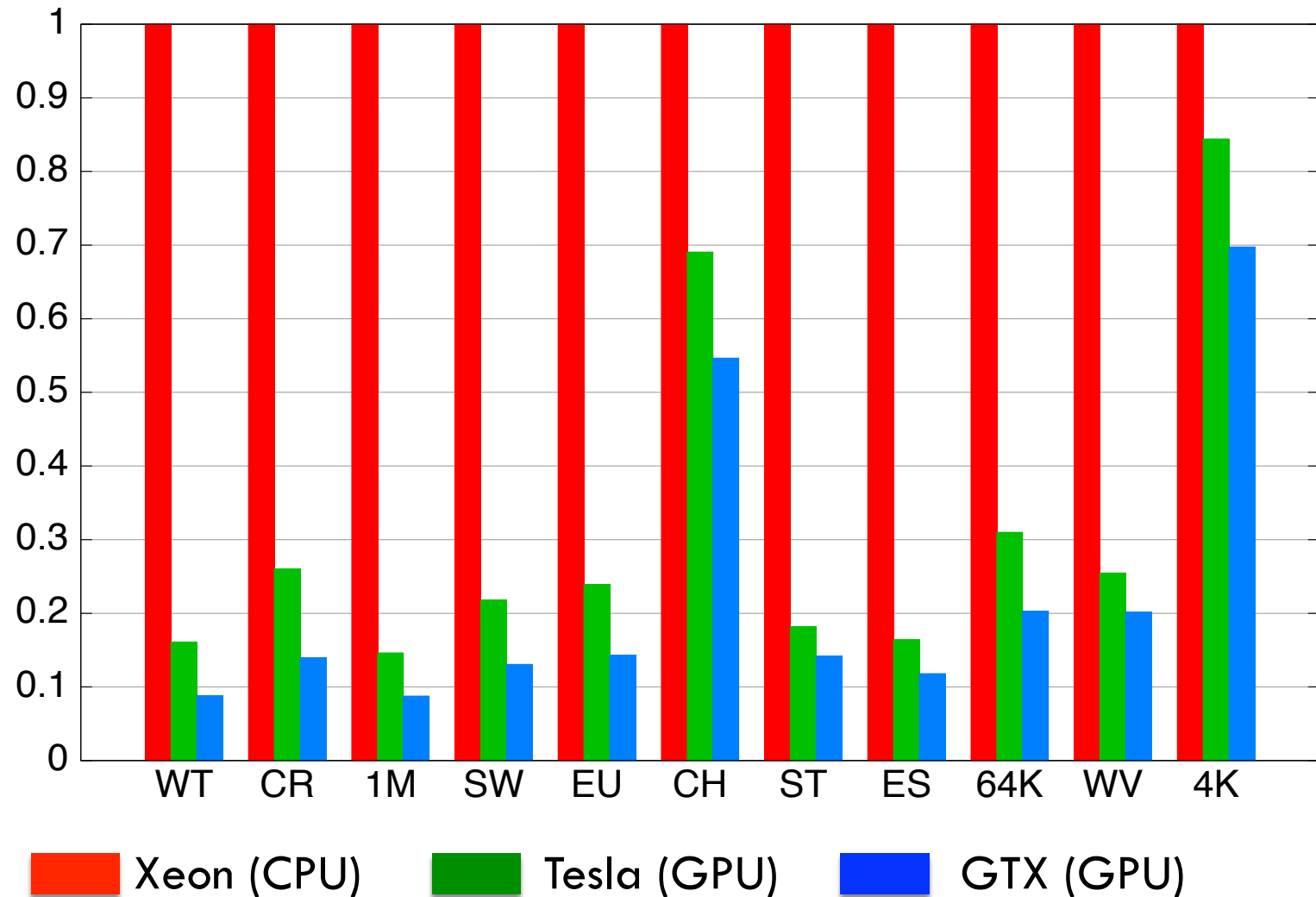
# BFS – normalized



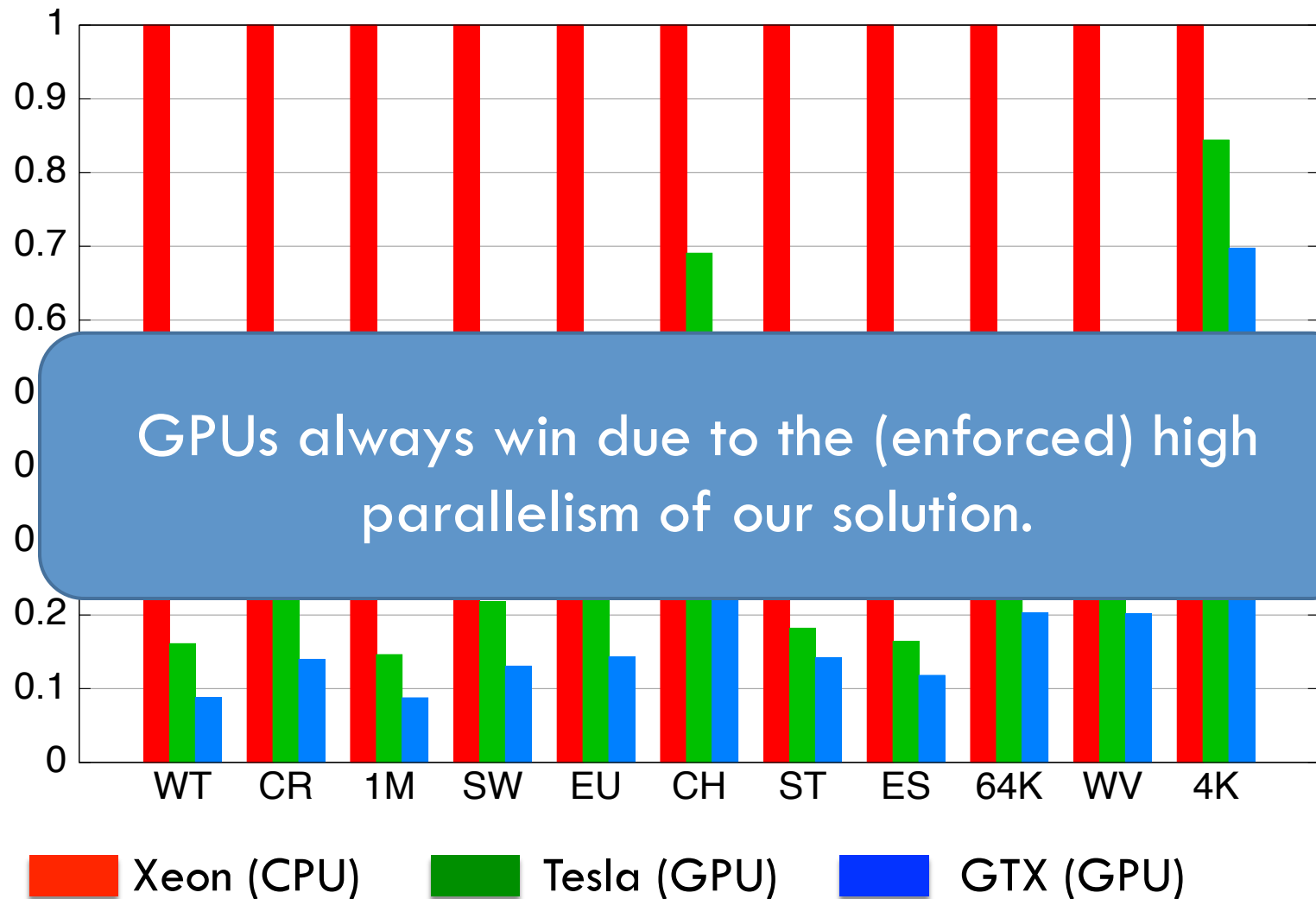
# BFS – normalized



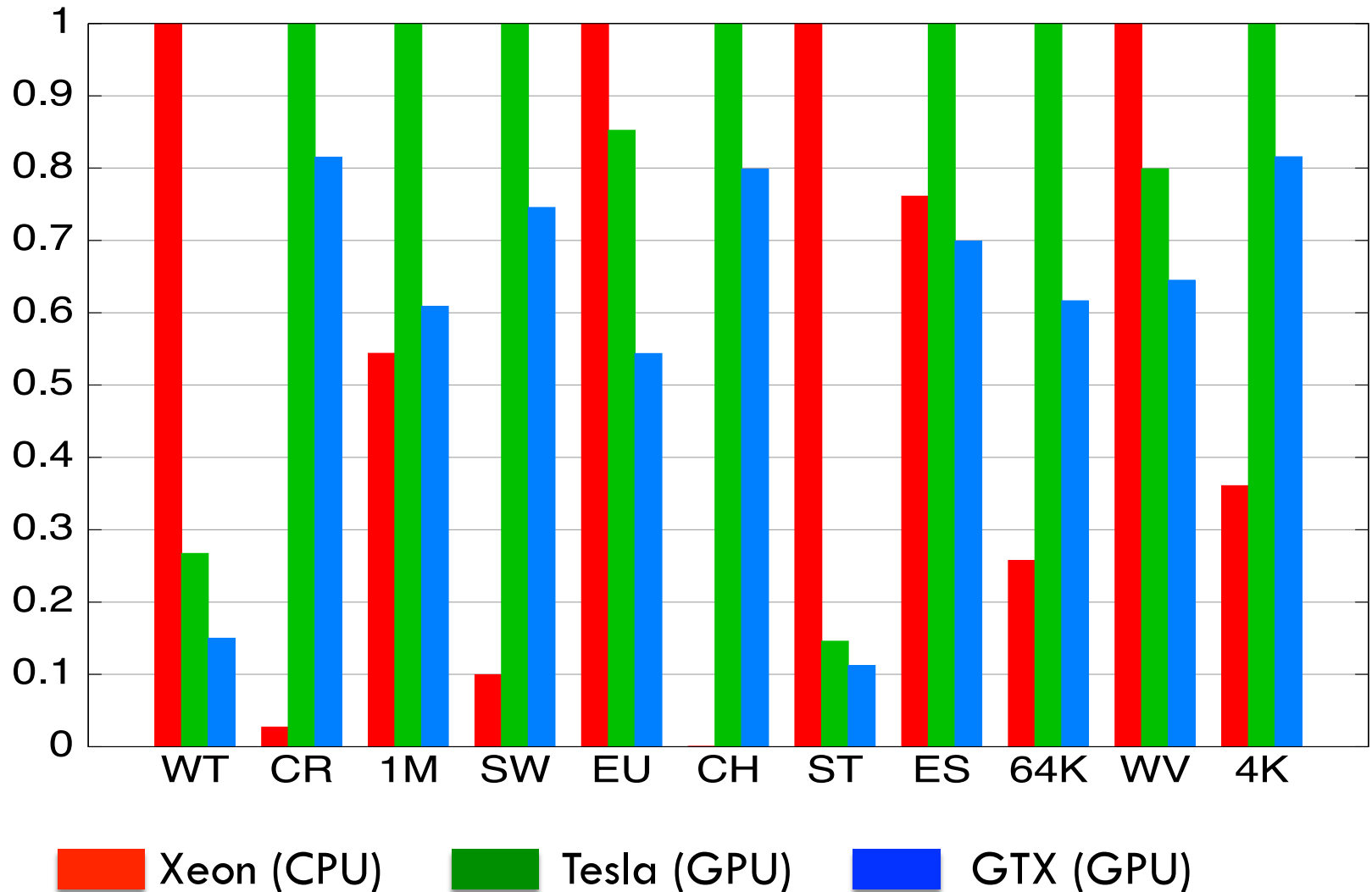
# APSP - normalized



# APSP - normalized

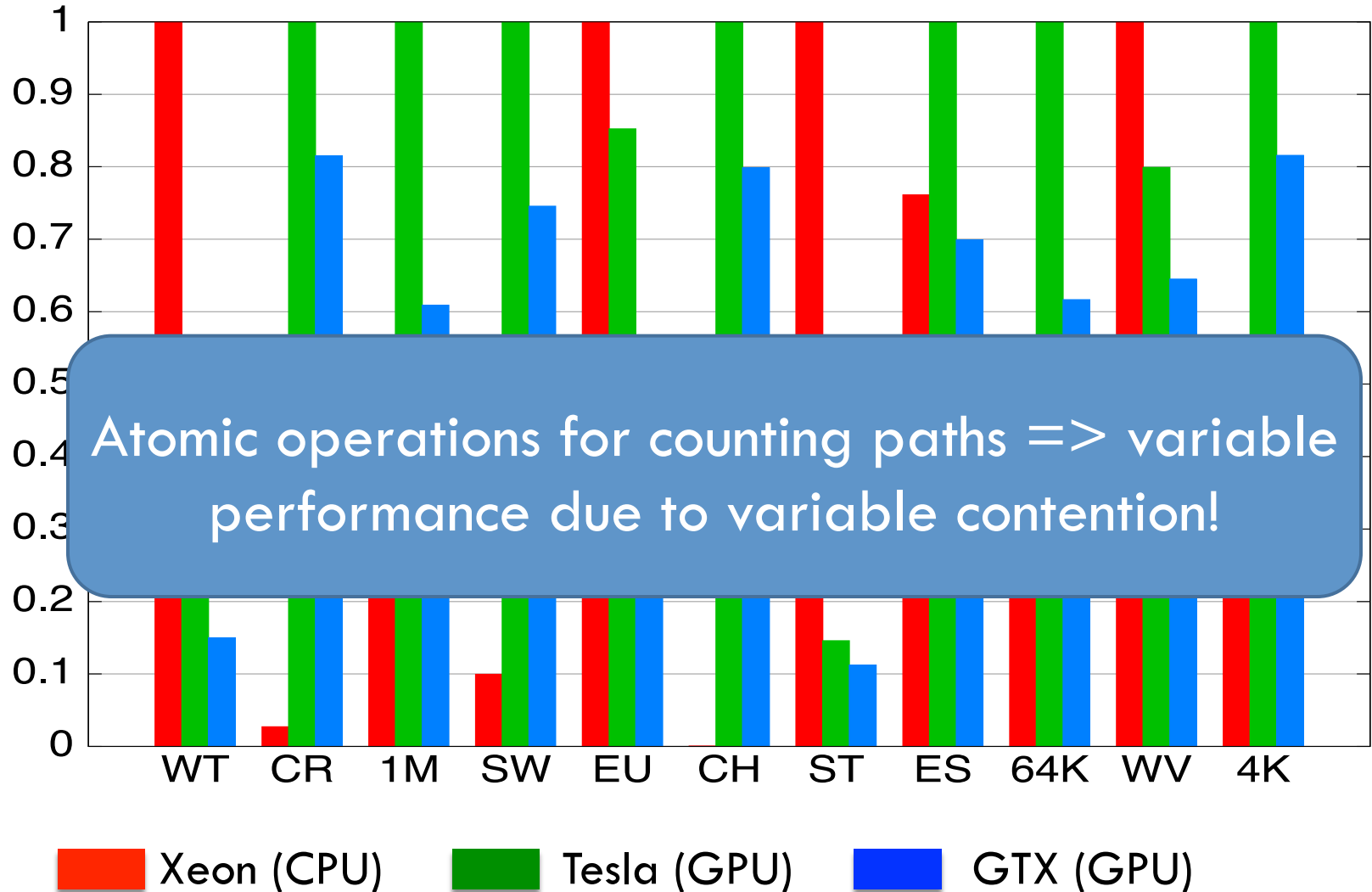


# BC - normalized





# BC - normalized



# Lessons learned

- ❑ Increased algorithm complexity may increase parallelism
- ❑ Dataset properties and data representation may increase parallelism
- ❑ Synchronization can be a hidden bottleneck
  - ▣ E.g.: BC mixes compute with synchronization
- ❑ We have no clear understanding of graph “sizes”
  - ▣ # vertices or # edges? Diameter? Other properties?
- ❑ Graphs seem to be CPU or GPU friendly
  - ▣ Heterogeneous processing?

# Experiment 2: BFS traversals

## □ Question:

- Is there a **best** BFS algorithm?

  - On GPUs ?

  - Overall ?

## □ Setup:

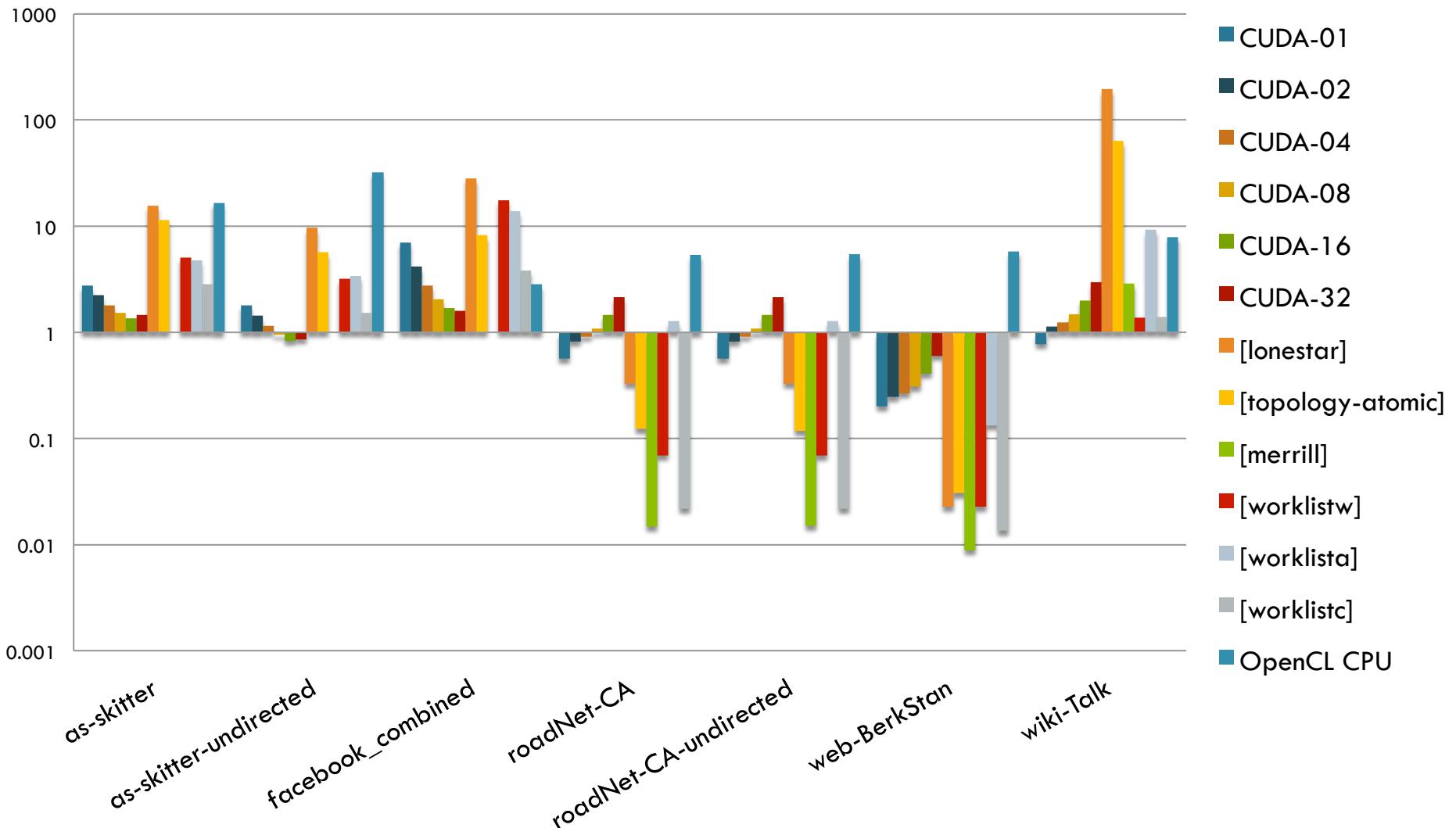
- Run multiple BFS implementations

  - Including the ones @ LonestarGPU

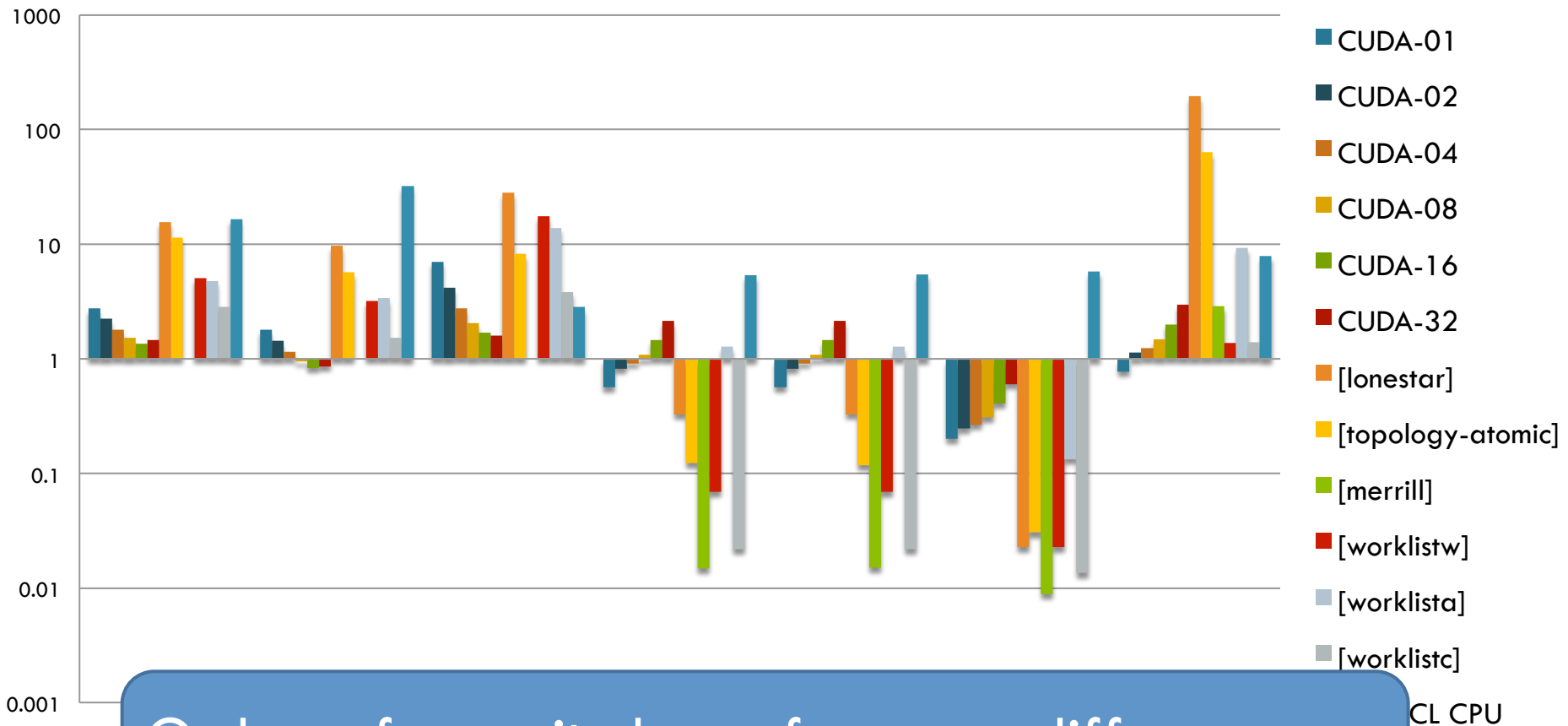
- Run on different graphs

  - 6 datasets

- Run on different hardware

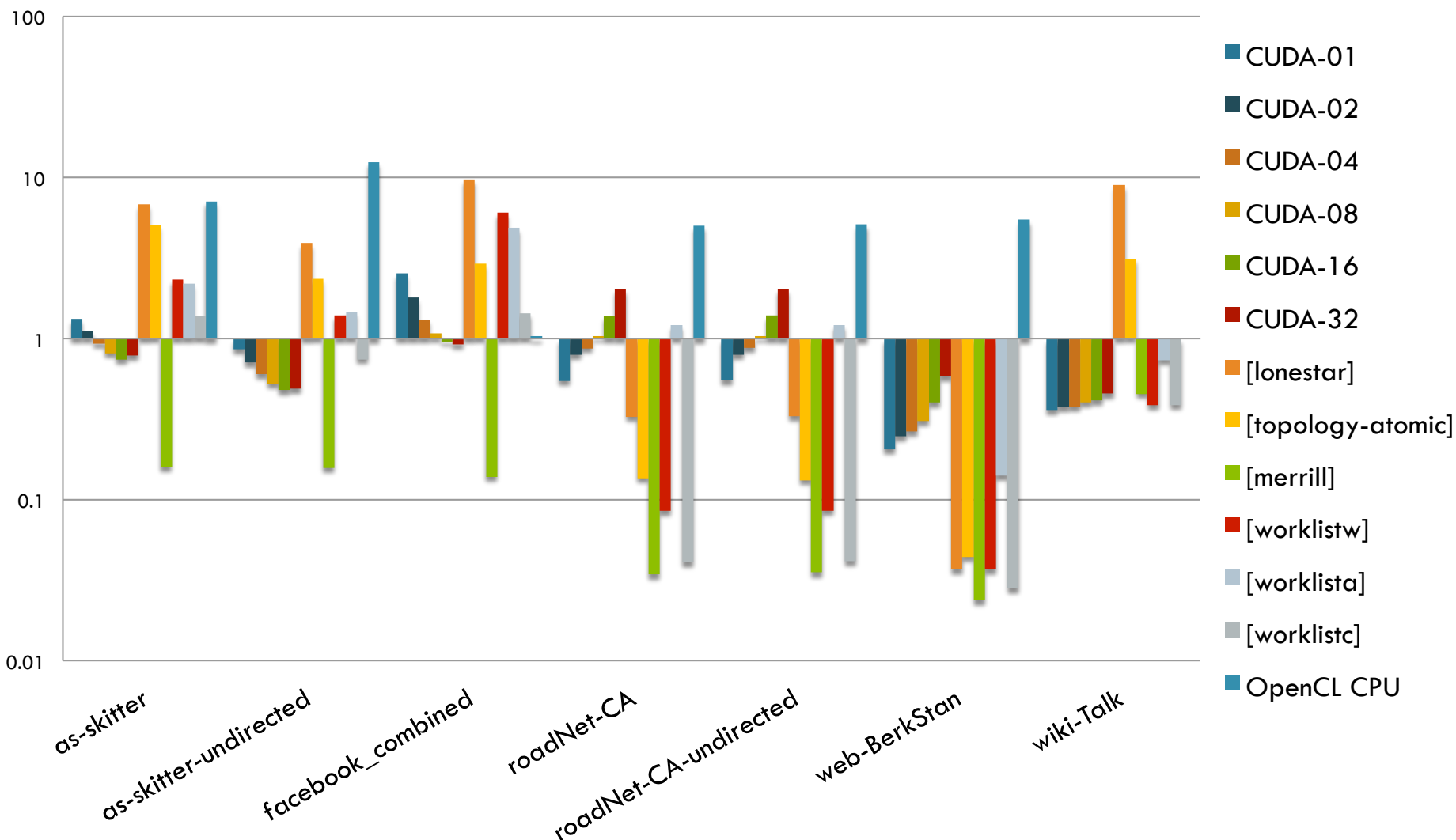


# Normalized on naïve GPU, kernel



Orders of magnitude performance difference.  
No clear winner.





# Normalized on naïve GPU, full exec.



Adding the data transfer times narrows the performance gaps.

# Lessons learned

- Depending on the graph ...
  - ▣ Large variability in performance (fastest to slowest ratio)
  - ▣ The relative performance of BFS implementation varies.
    - Fastest on one graph CAN BE slowest on another graph.
- Data representation and data structures make a BIG difference
- A naive CPU implementation can be competitive with some of the GPU implementations.
  - ▣ On small graphs (GPUs are underutilized)
  - ▣ When data transfer is an issue (think BFS)

# More experiments

- Similar results
  - ▣ Different BC implementations (available)
  - ▣ Different PageRank implementations (available)
  - ▣ Different APSP implementations (in progress)
  
- Different results for community detection!
  - ▣ GPUs are much better (algorithms have much more parallelism)
  - ▣ Heterogeneous computing pays off ... for memory increase!

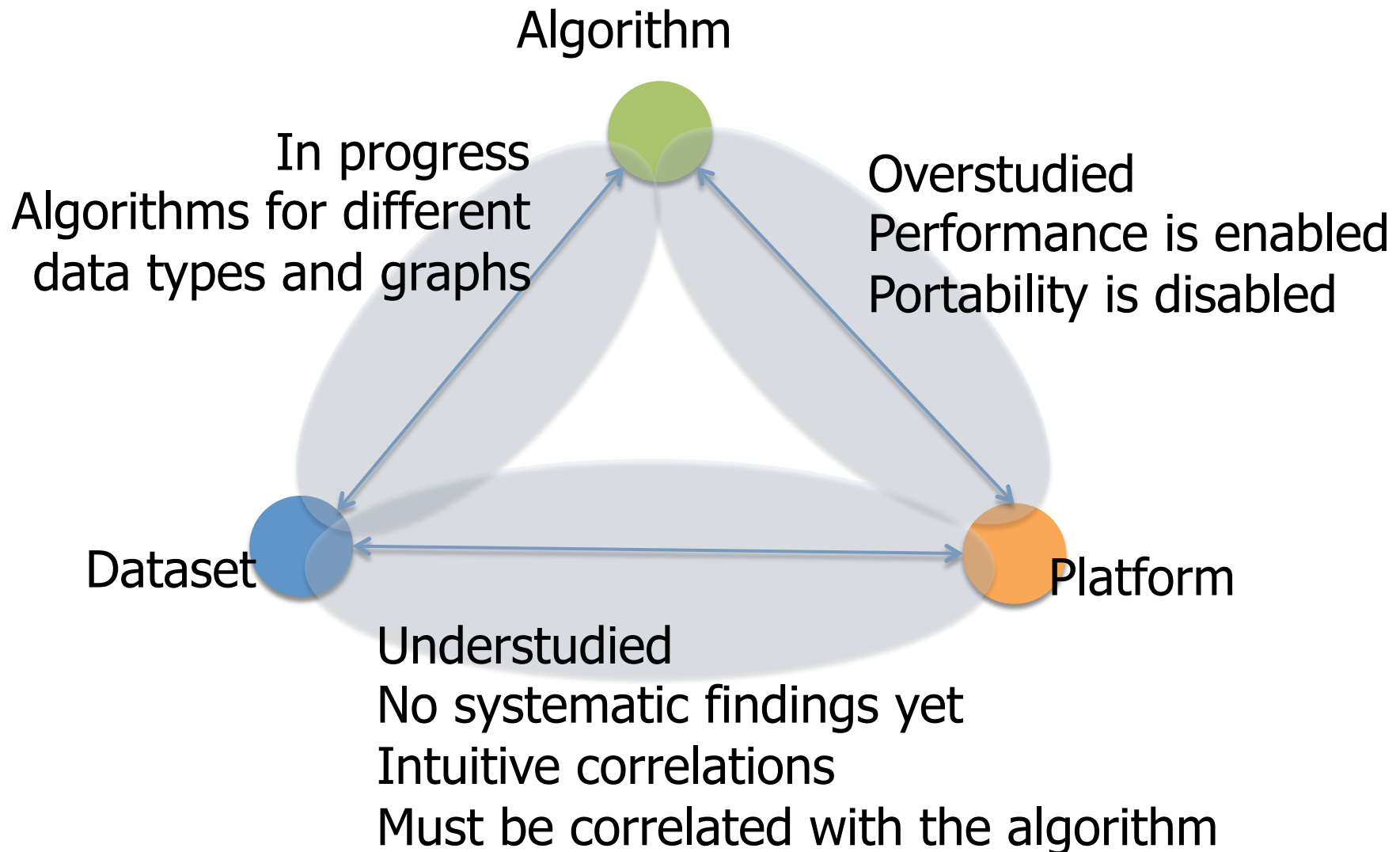
# Take home message



- Large scale graph processing IS high performance computing
  - ▣ Due to/for data scale \*and\* analysis complexity
- HPC hardware is useful for graph processing
  - ▣ yet performance is (for now) unpredictable
- Performance is dependent on all three “axes”
  - ▣  $\text{Performance} = f(\text{dataset, algorithm, hardware})$



# P-A-D triangle





# The landscape

# The landscape of modern graph processing

Performance

Today ... [1]

- Systems for graph processing
- Separate users from backends
- Think Giraph, Totem, Medusa, ....

Dedicated Systems

Custom

- Specify application
- Choose the hardware
- Implement & optimize
- Think Graph500

Generic

In other work of ours ... [2][3]

- Use existing large scale distributed systems
- Mapping is difficult
- Parallelism is “free”
- Think MapReduce

Development Effort

# The landscape of modern graph processing

- Graph processing is a hot HPC topic for both software and hardware developers
  - ▣ Challenges in scale and irregularity
- Existing graph processing systems : 80+
  - ▣ Survey in progress
- Choose which one to use?
  - ▣ Quick-Pick: choose a platform where (1) your graph fits and (2) you can program.
  - ▣ Systematic: meta-benchmarking, a.k.a., Graphalytics\*

# References

- [1] A.L.Varbanescu et. al – “Can Portability Improve Performance?” ICPE 2015
- [2] Y. Guo et. al, “An Empirical Performance Evaluation of GPU-Enabled Graph-Processing Systems”, CCGrid 2015
- [3] Y. Guo et. Al, “How Well do Graph-Processing Platforms Perform? An Empirical Performance Evaluation and Analysis”, IPDPS 2014