A POWER CHARACTERIZATION AND MANAGEMENT OF GPU GRAPH TRAVERSAL

ADAM MCLAUGHLIN[*], INDRANI PAUL[†], JOSEPH GREATHOUSE[†], SRILATHA MANNE[†], AND SUDHAKAR YALAMANCHILI[*]
[*]GEORGIA INSTITUTE OF TECHNOLOGY
[†]AMD RESEARCH

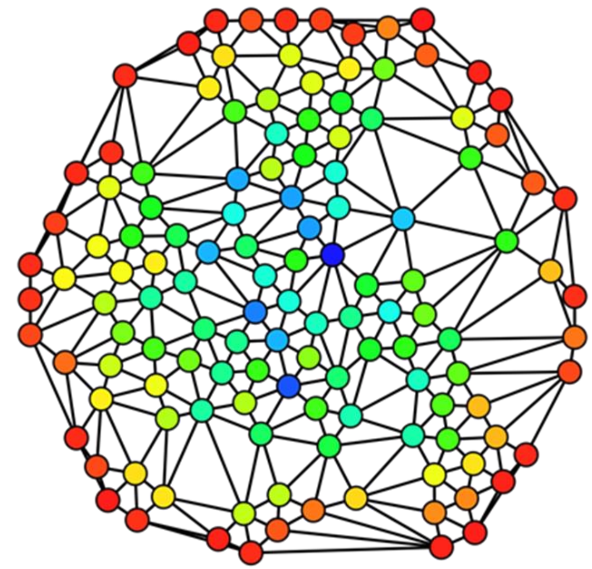
MOTIVATION

- Future machines may not be able to run at full power
  - Dark Silicon
  - Current SoCs prevent damaging hotspots and maintain thermal limits
  - Expensive
    - Installations consume tens of Megawatts
- Practical applications are constrained by power or thermal limitations
- The HPC community does not want to sacrifice performance for power
- All of the Top 10 machines from the Green 500 leverage GPUs
- It's critical to develop power management techniques for emergent irregular applications on GPUs

THE GREEN 500™

GRAPH 500

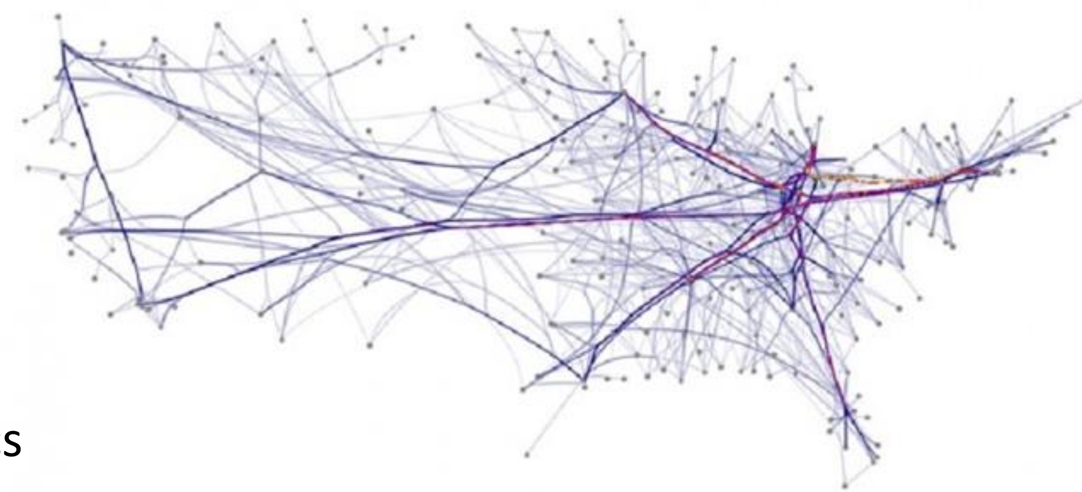
GRAPH ALGORITHMS



- Irregular Applications
  - Typically memory bound
  - Inconsistent memory access patterns
  - Characteristics unknown at compile time
  - Interesting data sets are massive
- Graph structures – Not a one size fits all problem
  - Scale-free
  - Small world
  - Road networks
  - Meshes

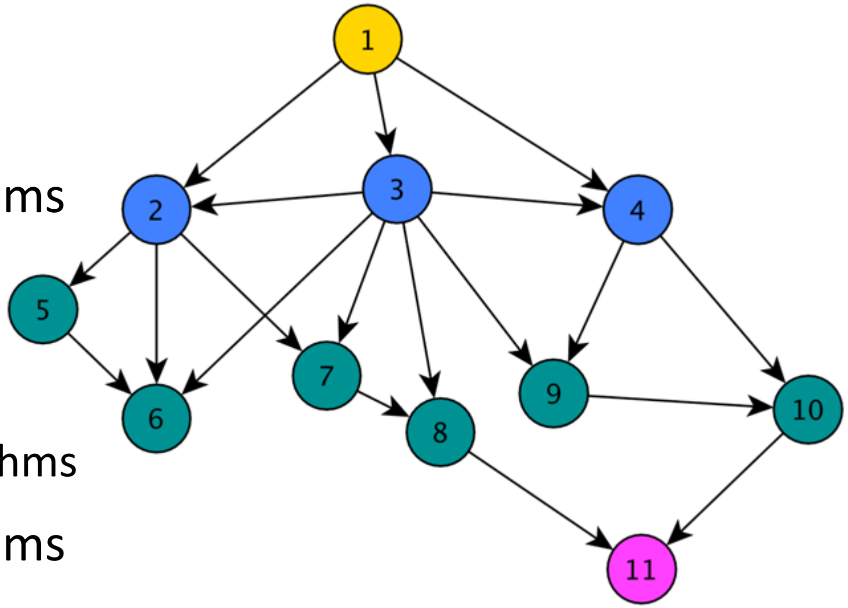
APPLICATIONS OF GRAPH ALGORITHMS

- Machine Learning
- Compiler Optimization
  - Register allocation
  - Points-to Analysis
- Social Network Analysis
- Computational Biology
- Computational Fluid Dynamics
- Urban Planning
- Path finding



BREADTH-FIRST SEARCH

- Choose a source node s to start from
- Explore neighbors of s
  - Explore neighbors of neighbors, and so on
- Building block to more complicated problems
  - Betweenness Centrality
  - All-pairs Shortest Paths
  - Strongly Connected Components
  - “Bricks and Mortar” of classical graph algorithms
- Especially useful for parallel graph algorithms
  - Depth-First Search is P-Complete

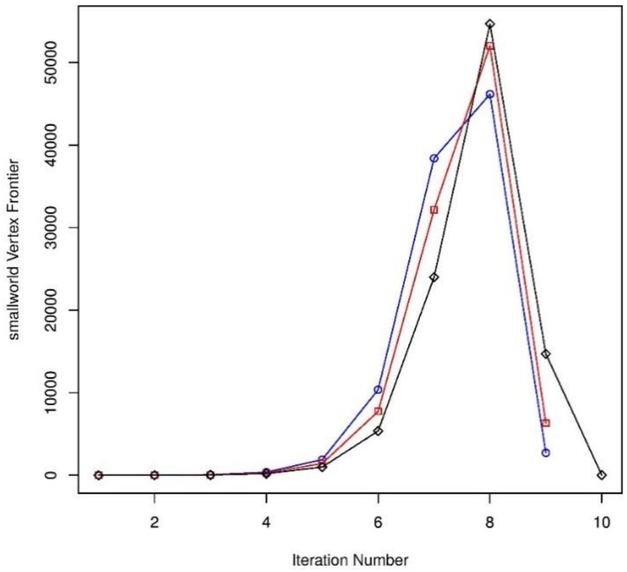


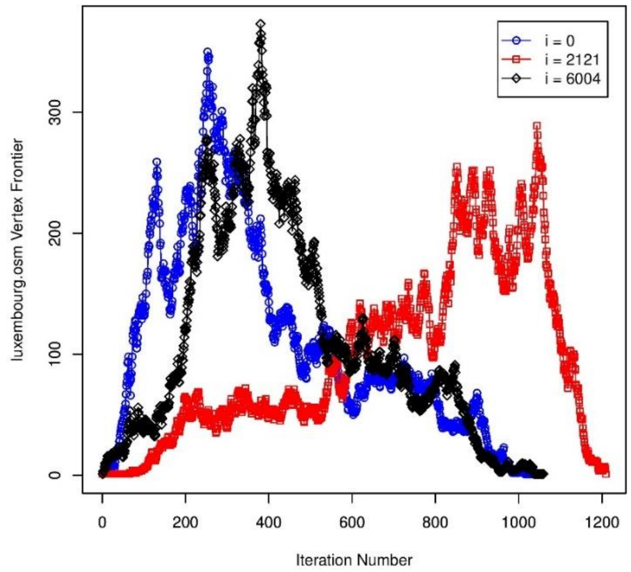
RECENT WORK ON BFS

- SHOC Benchmark Suite
  - Quadratic [Harish and Narayanan HiPC ‘07]
    - Naïvely assign a thread to every vertex on every iteration
    - Lots of unnecessary memory fetches and branch overhead
  - Linear with atomics [Luo, Wong, and Hwu DAC ’10]
    - Asymptotically Optimal $O(m+n)$ work
      - For graphs with n vertices and m edges
    - Fastest publicly available OpenCL implementation
    - Used for the experiments in this paper
- Linear with prefix sums [Merrill, Garland, and Grimshaw PPoPP ‘12]
  - Fastest GPU implementation
- Direction-Optimizing [Beamer, Asanović, and Patterson SC’12]

CHANGE IN PARALLELISM OVER TIME

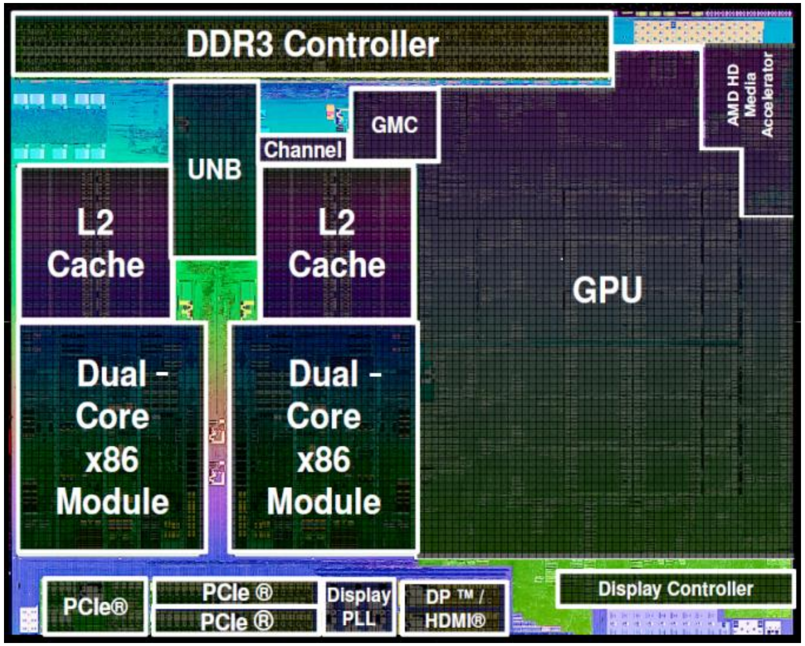
- Two trends
  - Few BFS iterations that process many nodes each
    - Scale-free, small world
  - Many BFS iterations that process few nodes each
    - Road networks, sparse meshes





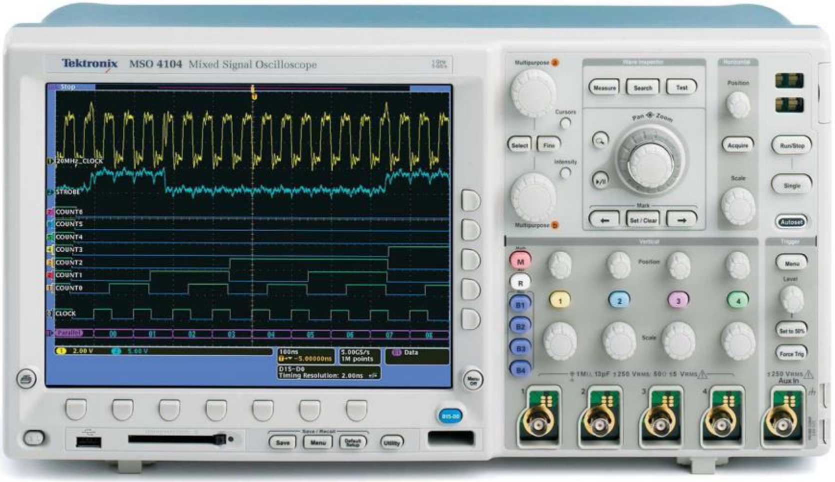
EXPERIMENTAL SETUP

- How do we leverage this information to manage power?
  - Two "knobs" of control
    - DVFS state
    - Number of active Compute Units (CUs)
- A10-5800K Trinity APU
  - 384 Radeon Cores
    - 6 SIMD Units
    - 16 Lanes with 4-way VLIW
  - 3 DVFS States
    - High: 800 MHz, 1.275V
    - Medium: 633 MHz, 1.2V
    - Low: 304 MHz, 0.9375V
  - 18 Manageable Power States
    - Up to 6 Active SIMDs (Compute Units)
    - 3 DVFS States



POWER MEASUREMENTS

- Measure GPU power directly
  - Receive estimates from power management firmware
  - Sample power every millisecond
- Overhead of changing DVFS state ~ microseconds
- Analyze power configurations offline
  - Limitations in changing power states during execution
- Throughput Baseline
  - Low Frequency
  - 4 Active CUs
- Latency Baseline
  - Medium Frequency
  - 2 Active CUs



DISTINGUISHING POWER AND ENERGY

- Our goal is to maximize performance in a power-constrained environment
- Our goal is NOT to minimize energy
  - "Race to idle" is not a valid solution

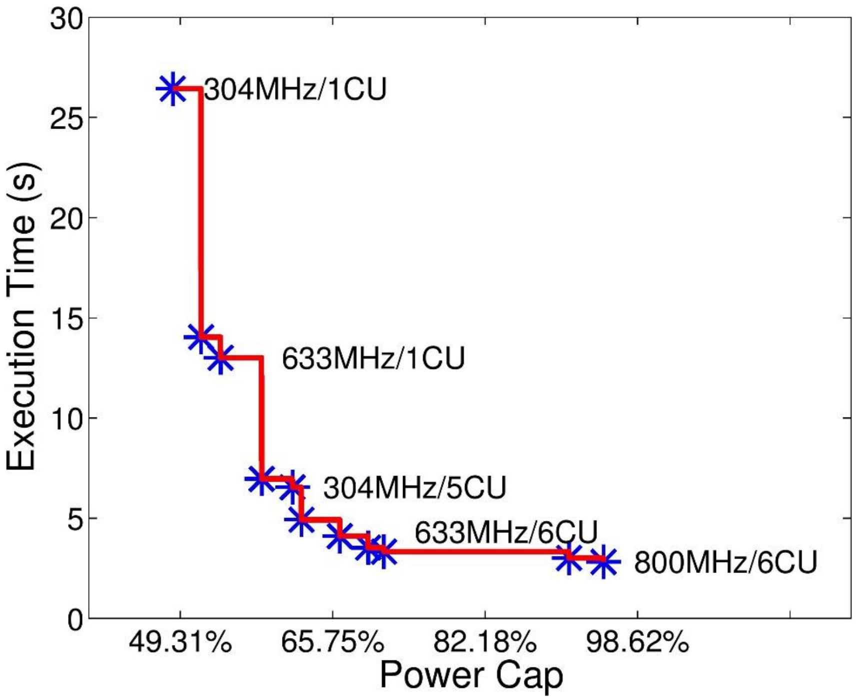
BENCHMARK GRAPHS

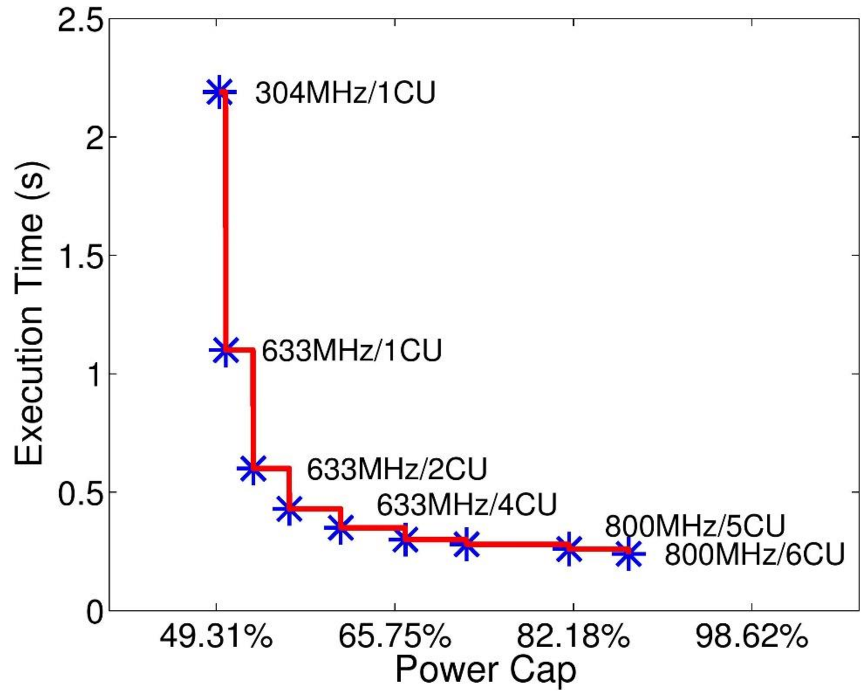
Name	Vertices	Edges	Significance
coPapersCiteseer	434,102	16,036,720	Social Network
delaunay_n23	8,388,608	25,165,784	Random Triangulation
asia.osm	11,950,757	12,711,603	Street Network
ldoor	952,203	22,785,136	Sparse Matrix
af_shell10	1,508,065	25,582,130	Sheet Metal Forming
kkt_power	2,063,494	6,482,320	Nonlinear Optimization
rgg_n_2_22_s0	4,194,304	30,359,198	Random Geometric Graph
G3_circuit	1,585,478	3,037,674	AMD Circuit Simulation
hugebubbles_00020	21,198,119	31,790,179	2D Dynamic Simulations
in-2004	1,382,908	13,591,473	Web Crawl
packing_500x100x100-b050	2,145,852	17,488,243	Fluid Mechanics

STATIC ORACLE

- Given a graph and power cap, determine the best power state
  - Exhaustively run all settings
  - Pick the setting that has…
    - …the least execution time
    - …instantaneous power within the cap at all times
  - Refer to this setting as the *static oracle*
    - "Static" because the same power setting is used throughout the traversal

BEST CONFIGURATION VARIES WITH GRAPH INPUT

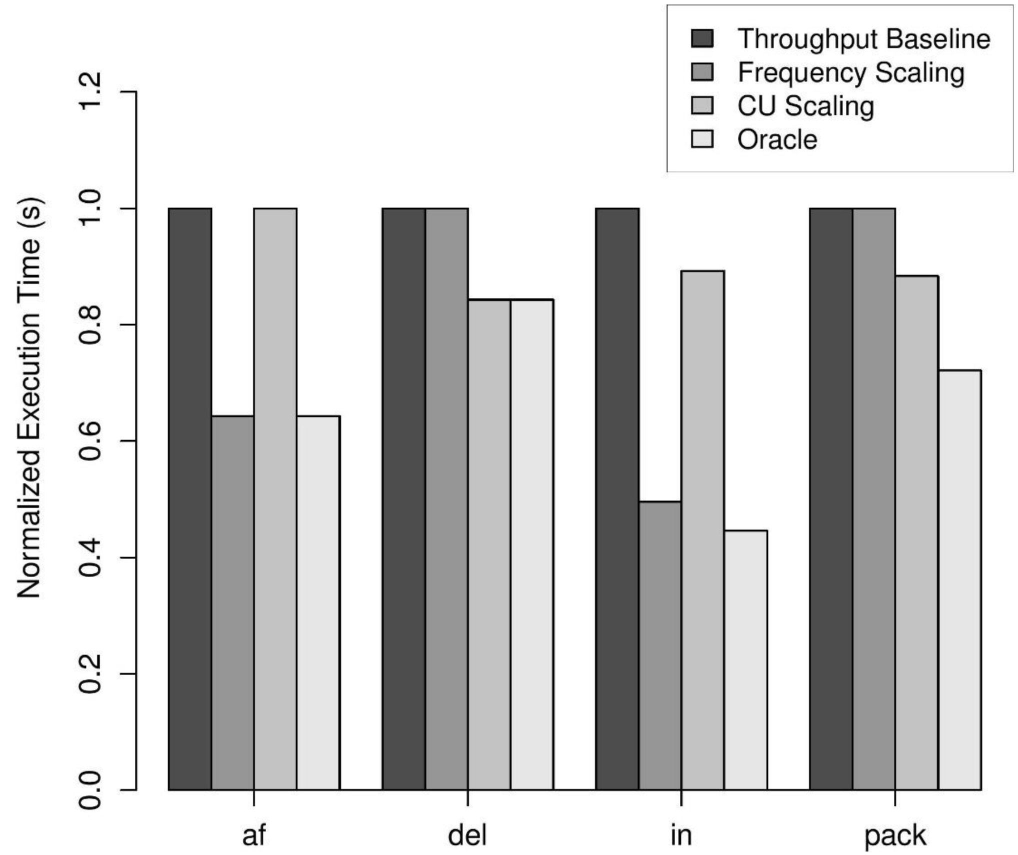




- Consider an 82.18% Power Cap
  - Left (delaunay_n23): Medium Frequency and 6 CUs
  - Right (G3_Circuit): High Frequency and 5 CUs

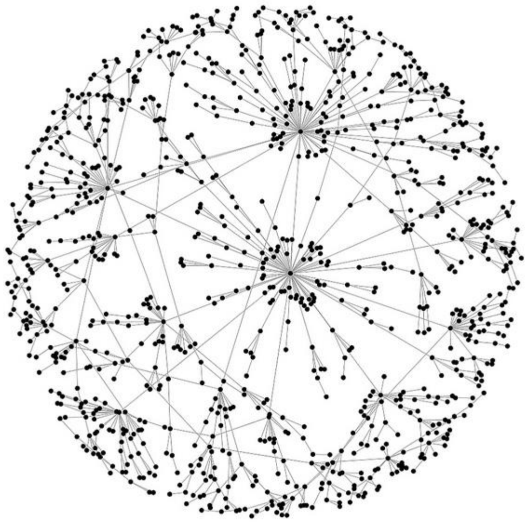
LEVERAGING BOTH DEGREES OF FREEDOM

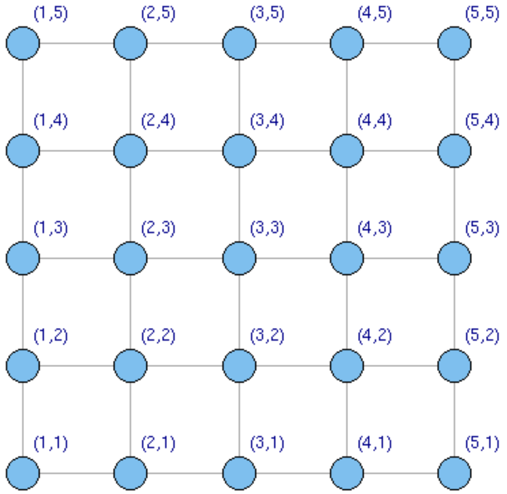
- Sometimes it is better to boost frequency than CUs (af)
- Sometimes it is better to boost CUs than frequency (del)
- Boost both degrees somewhat rather than boosting one maximally (in)
- Reduce one degree to be able to boost the other (pack)



AN ALGORITHMIC APPROACH

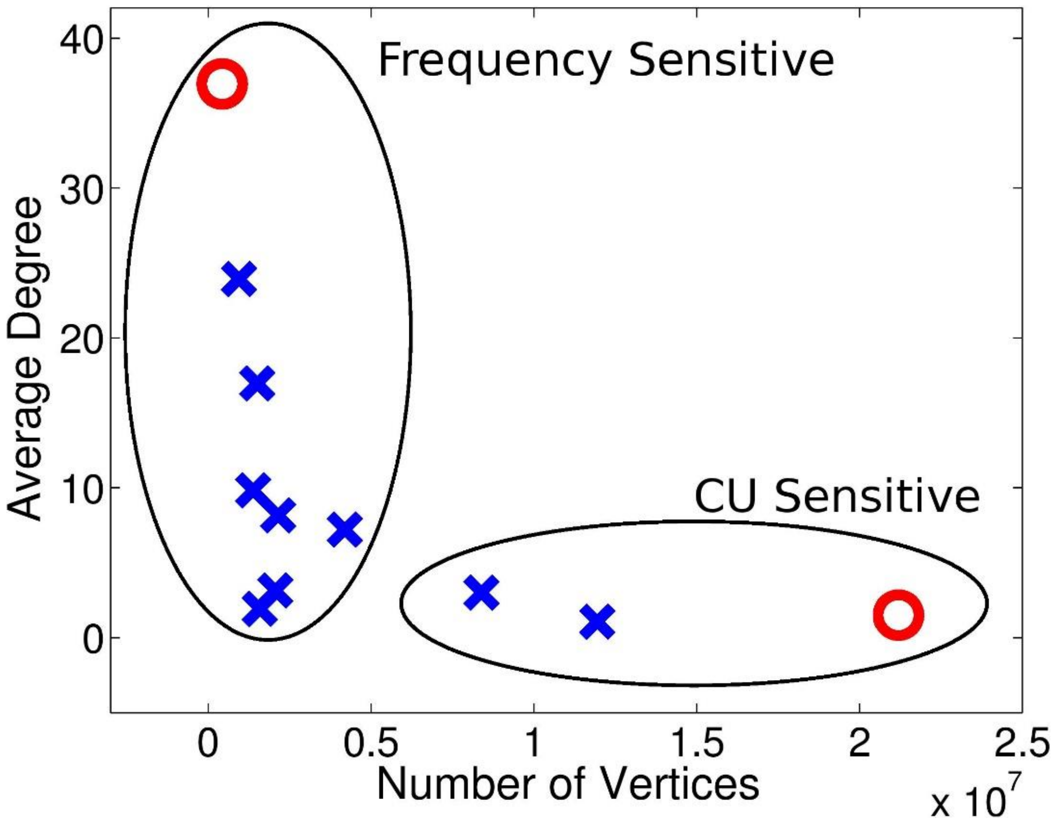
- How to determine the best configuration for a given graph and power cap?
- Intuition: Graphs tend to be more sensitive to either latency or parallelism
  - Use simple, offline, graph metrics to determine this sensitivity
    - Number of nodes
    - Average degree
  - Diameter would be ideal, but that requires too much preprocessing



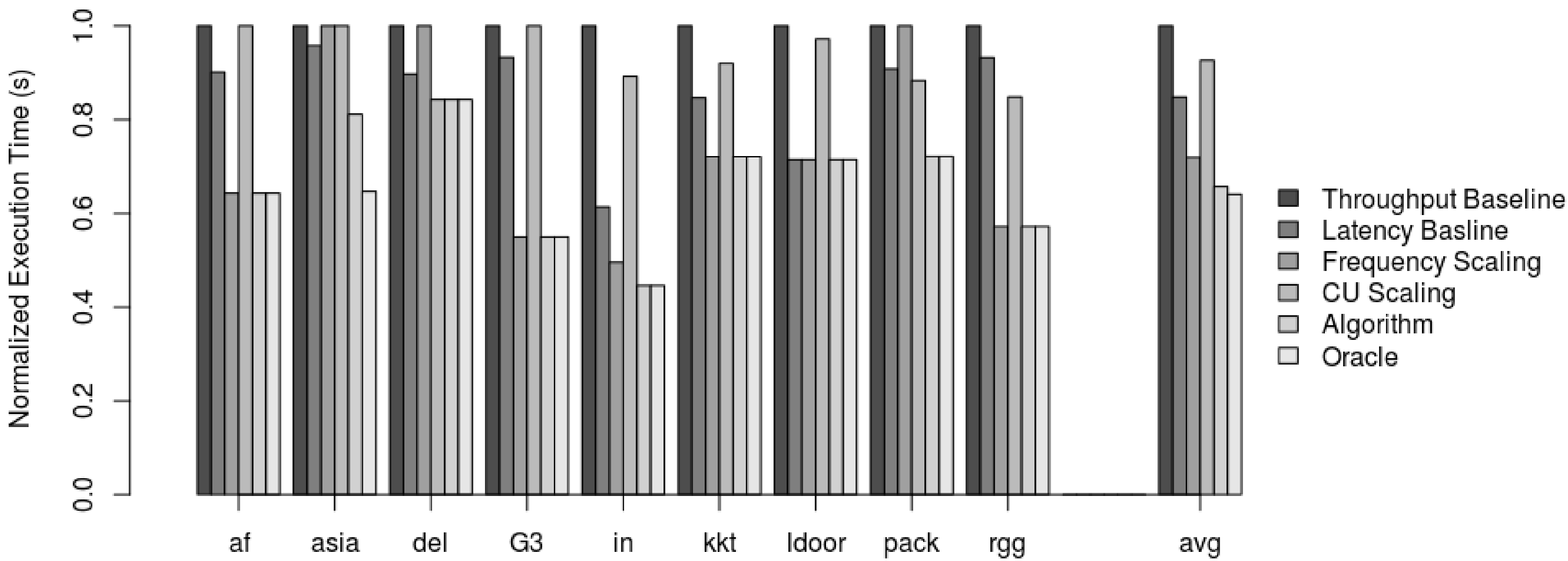


CLUSTERING

- Red circles: training set
- Blue x's: Classified via K-means clustering
- High average degree implies a high potential for load imbalances
  - Scale-free, small world graphs
- Low average degree means more uniform work
  - Meshes, Road networks



STATIC RESULTS



- Algorithm matches the oracle for 8/9 graphs
- CU scaling less helpful
  - Baseline already has 4 active CUs
  - Matter of perspective

CONCLUSIONS

- Power optimizations depends heavily on graph structure
- Frequency boosting is a useful technique
  - Already implemented in contemporary HW
  - We show that CU boosting is also useful
  - …and that combining Frequency and CU boosting is even better
- Simple graph metadata suffices for making power management decisions
  - No preprocessing required
- HW needs to support finer granularities of power management

QUESTIONS

- We would like to thank the NSF and AMD for their support

IMPROVEMENTS: DYNAMIC ALGORITHM

- Choose the best configuration at each iteration of the search
  - Exhaustively test all iterations at all power configurations
  - Choose the fastest of the ones that do not exceed the power cap
  - Refer to this setting as the *Dynamic Oracle*
- Two ways to improve over the static algorithm
  - If the static algorithm classifies a graph incorrectly
  - If the vertex frontiers change significantly in size
    - Scale CUs when frontiers are small
    - Scale frequency when frontiers are large

DYNAMIC RESULTS

- Modest improvements
  - ~5% overall
- More variation in structure than available power states
  - Need finer-grained methods of power management
- Small number of iterations dominate
  - Static case can optimize for these iterations