PaViz: A Power-Adaptive Framework for Optimizing Visualization Performance

EGPGV, Barcelona, Spain

June 12-13, 2017

Stephanie Labasan, *Lawrence Livermore, Univ of Oregon*
Matthew Larsen, Hank Childs, and Barry Rountree

LLNL-PRES-731862
This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344. Lawrence Livermore National Security, LLC

Lawrence Livermore National Laboratory

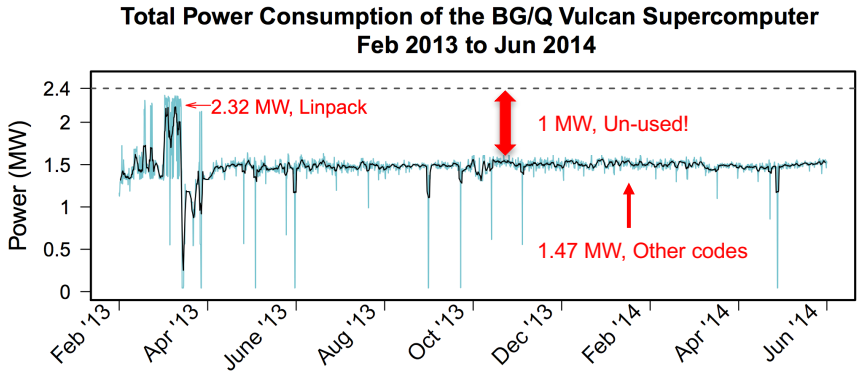
This study was motivated by the power limitations at exascale

- **Goal**: Improve power utilization to maximize performance in a power-constrained environment
- **Method**: Enforce power caps, affecting the CPU frequency and voltage
  - **Proposition for users**: Slow down some nodes in order to speed up other nodes, improving overall execution time
- **Study**: Use a performance model for rendering to dynamically rebalance power to where it will do the most good

Presentation Outline

- Background
- PaViz and Research Questions
- Study Overview
- Results
- Takeaways

Current system power utilization is sub-optimal



2.4 MW: Assumes all nodes consume peak power at the same time

- Limits the size of the system, reducing system throughput
- Power capacity left on the table, more science could be done
- Wasted money spent on infrastructure

c/o T Patki

U.S. Department of Energy's perspectives on exascale challenges

Systems	2009	"~~2018~~" "2023"	Difference Today & ~~2018~~ 2023
System peak	2 Pflop/s	1 Eflop/s	O(1000)
Power	6 MW	20 MW	~3-5
System memory	0.3 PB		
Node performance	125 GF		100)
Node memory BW	25 GB/s		
Node concurrency	12		1000)
Total Node Interconnect BW	3.5 GB/s		
System size (			0)-O(1000)
Total Concurr			O(100)
Storage			0)-O(100)
IO			O(100)
MTTI	Days	O(1 day)	- O(10) c/o P Beckmann

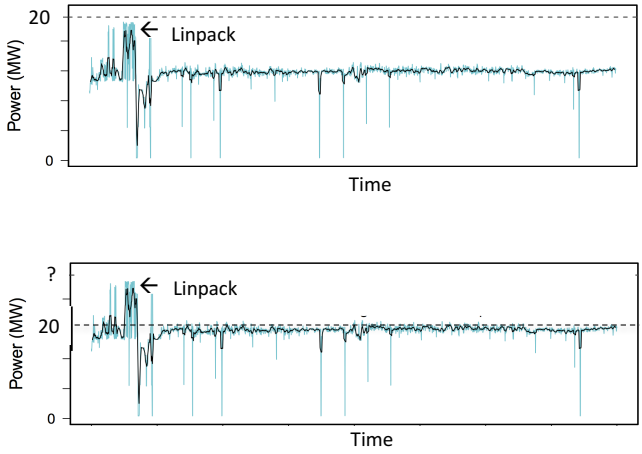
Systems	2009	"~~2018~~" "2023"
System peak	2 Pflop/s	1 Eflop/s
Power	6 MW	20 MW

Hardware innovations have been the primary solution for power savings. However, to reach 1 Exaflop at the desired power budget, software improvements may be necessary.

Power on future exascale supercomputers

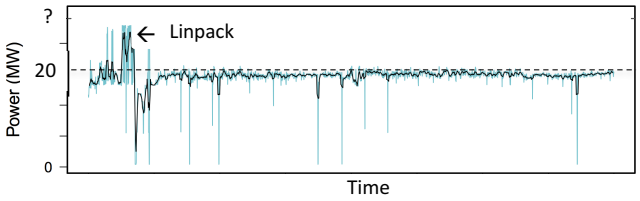
Two scenarios

1. System peak power consumption is ≤ 20 MW
   - *i.e.*, even Linpack stays below this limit
   - Poor power utilization
2. System peak power consumption is > 20 MW
   - Controls needed to enforce power usage at 20 MW power cap
   - Central manager coordinates power allocations
   - Better power utilization and more science completed



Improve power utilization with *overprovisioning*

- Increase compute capacity by adding more nodes
  - Limit power per node



- Not all nodes can run at max power simultaneously – bad for Linpack
- All nodes can be run at (say) 50% power – great for common use case

The entire HPC ecosystem will need to adapt to these constraints, including visualization and analysis routines.

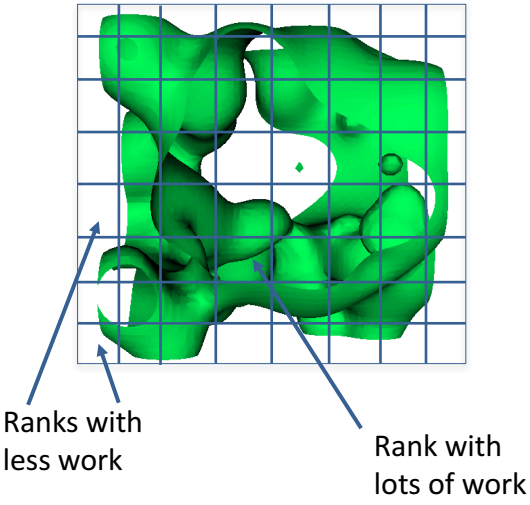
c/o B Rountree

Presentation Outline

- Background
- PaViz and Research Questions
- Study Overview
- Results
- Takeaways

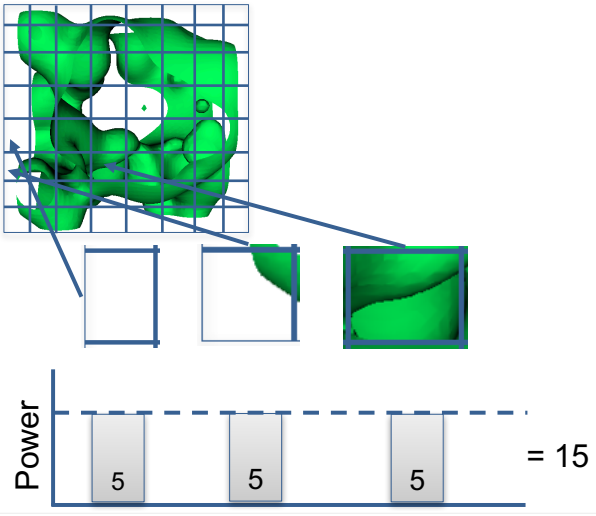
Imbalance in visualization routines poses an opportunity for performance improvements

- Canonical example = isosurfacing
  - Geometry (*i.e.*, work) will not be symmetrical
  - Work per node will vary...some nodes will have lots of work, others with no work
- **Give** power to rank with lots of work
- **Take** power away from rank with less work
- Improve overall runtime



Imbalance in visualization routines poses an opportunity for performance improvements

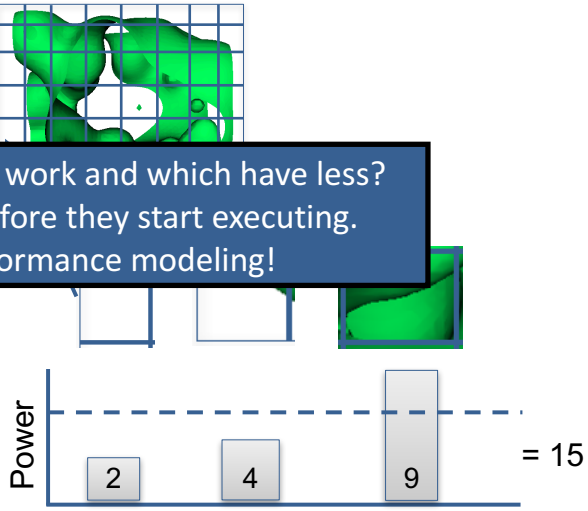
- Canonical example = isosurfacing
  - Geometry (*i.e.*, work) will not be symmetrical
  - Work per node will vary...some nodes will have lots of work, others with no work
- **Give** power to rank with lots of work
- **Take** power away from rank with less work
- Improve overall runtime



Imbalance in visualization routines poses an opportunity for performance improvements

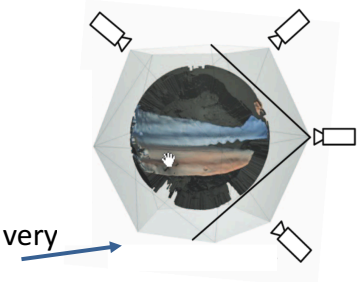
- Canonical example = isosurfacing
  - Geometry (*i.e.*, work) will not be symmetrical
  - Wor
    will
    wor
- **Give** power to rank with lots of work
- **Take** power away from rank with less work
- Improve overall runtime

So, which ranks have more work and which have less?
We need to know this before they start executing.
Idea: Let's use performance modeling!



PaViz: Power-Aware Visualization Runtime Framework

- This study: PaViz
  - Software power-aware runtime framework
  - Uses predicted execution times per node to assign a schedule of power allocations
- System focus is on rendering
  - Incorporates existing performance model
  - Key phase in the visualization pipeline
  - Typically a very fast operation, but could be very time-consuming (CINEMA)



Ahrens et. al "An image-based approach to extreme scale *in situ* visualization and analysis"

Putting It All Together

- Visualization + overprovisioning merits special attention
  - **Why?** Visualization workloads are imbalanced and irregular
  - → Need visualization-centric techniques to thrive in this constrained environment

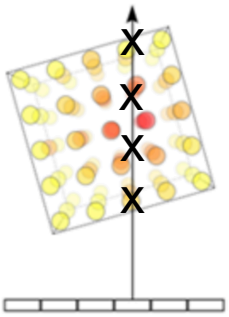
- **Research Question**: Can additional performance be gained by leveraging performance models for visualization algorithms?
  - **Idea**: Alter per node power allocations based on predicted runtime

Presentation Outline

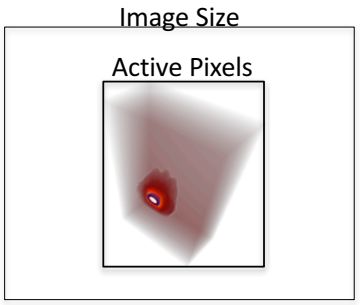
- Motivation
- PaViz and Research Questions
- Study Overview
- Results
- Takeaways

Performance Factors for Volume Rendering

- Image-order algorithm
  - For each pixel, trace a ray through the volume and sample at regular intervals
- Volume rendering performance is influenced by three key factors:
  - Number of unique cells sampled
  - Total number of samples
  - Number of pixels in image vs. number of active pixels



Sample Along The Ray



Screen Space Bounding Box

c/o M Larsen

Volume Rendering Performance Model

- Predicted time to complete volume rendering per node:

$$T_{VR} = c_0 * (AP * CS) + c_1 * (AP * SPR) + c_2$$

Cell Frequency

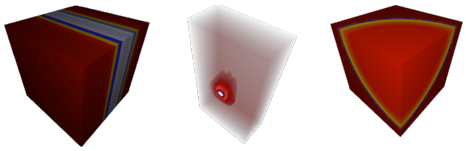
Sample Frequency

- AP = number of active pixels
- CS = number of cells spanned
- SPR = samples per ray

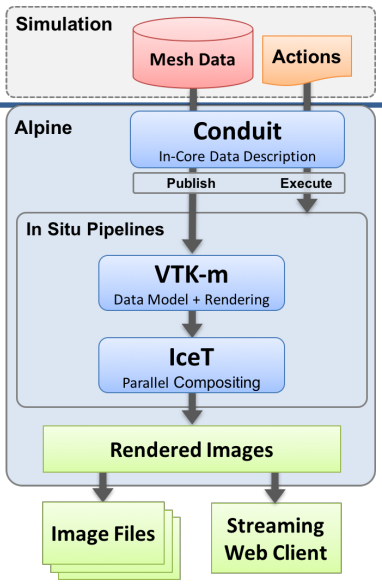
Larsen et al. "Performance Modeling of In Situ Rendering". SC16

Strawman

- In situ visualization infrastructure connecting simulation to visualization
- Coupled with (3) proxy applications, **CloverLeaf3D**, Kripke, and Lulesh



- DOE Exascale Computing Project Alpine: https://github.com/alpine-dav/alpine



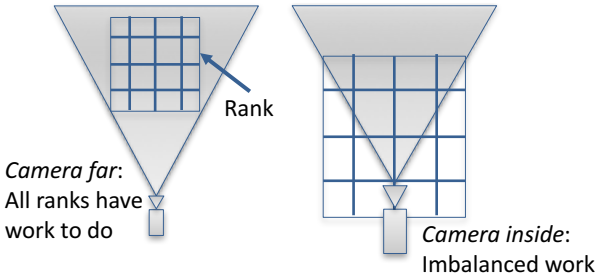
Larsen et al. "Strawman: A Batch In Situ Visualization and Analysis Infrastructure for Multi-Physics Simulation Codes". ISAV15

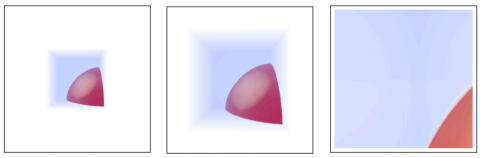
Hardware Architecture

- LLNL's Catalyst supercomputer
  - Dual socket nodes containing Intel Ivy Bridge CPUs each with 12 hyper-threaded cores
  - Nominal CPU frequency of 2.4 GHz
  - Peak power draw of 115W
- CPU power cap with Intel's Running Average Power Limit (RAPL)
  - Over a given time window, the average power usage will not exceed the specified power cap
  - Underlying firmware dithers CPU operating frequency and voltage
- LLNL's *msr-safe* kernel (https://github.com/llnl/msr-safe) enables setting power caps from userspace

Study Overview

- Rendering workload





Wkld	Data Set	Image Res	Cam Pos	Imbalance Factor
A	240^3	2880^2	inside	1.57
B	470^3	1080^2	near	1.16
C	128^3	1920^2	inside	1.58
D	320^3	2048^2	far	1.12

- Camera position affects imbalance level
  - **Imbalance Factor** = max runtime/min runtime
  - High **Imbalance Factor** = more imbalanced

Study Overview

- Rendering workload
- MPI task concurrency

- One MPI task per node for maximum memory partition
- 8 tasks (ranks) = 192 cores
- 64 tasks (ranks) = 1,472 cores

Study Overview

- Rendering workload
- MPI task concurrency
- Power scheduling strategy

- Five strategies explored: *Min*, *Normalized*, *Mean*, *Median*, *Max*

Min

$$P_i = \frac{t_i - t_{min}}{\sum t_i - t_{min}}$$

Normalized

$$P_i = \frac{t_i}{\sum t_i}$$

- P_i = proportion of total power allocated to rank *i*
- Ensure strategy does not over-allocate job power budget ($\sum P_i \leq 1$)

See this paper for details of other power scheduling strategies.

Study Overview

- Rendering workload
- MPI task concurrency
- Power scheduling strategy
- Job power budget

Supercomputer is not overprovisioned. However, we artificially limit the power per node to simulate an overprovisioned environment.

CPU0
115W → 64W
5W increments

CPU1
115W → 64W
5W increments

Node : 230W → 128W

- *8 nodes* : 1,840W → 1,024W
- *64 nodes* : 14,720W → 8,192W

*1 MPI task per node

Methodology

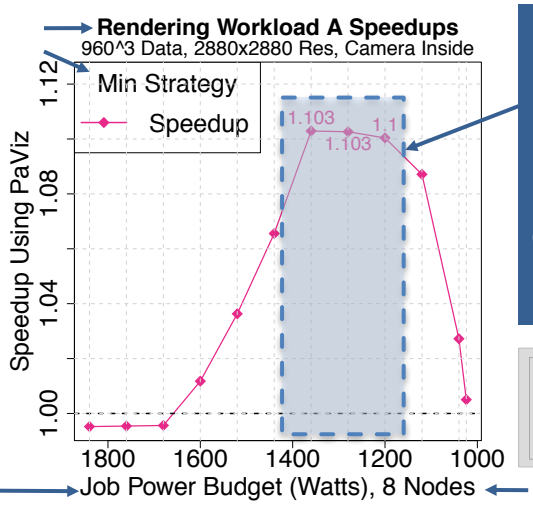
- Rendering workload
- MPI task concurrency
- Power scheduling strategy
- Job power budget

- Study conducted in 4 phases
- Phase 1: Vary job power budget for a single rendering workload, 8 MPI tasks, *Min* power scheduling strategy
- Each phase varies 1 factor to explore effects

Presentation Overview

- Background
- PaViz and Research Questions
- Study Overview
- Results
- Takeaways

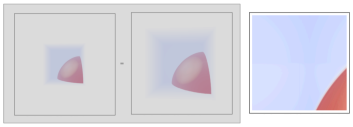
Phase 1 Results: Vary Job Power Budget



10% speedup in execution time by redistributing power to nodes with more work.

This is comparing to current strategy of uniform power distribution across nodes.

Reducing job power budget, more severe power cap



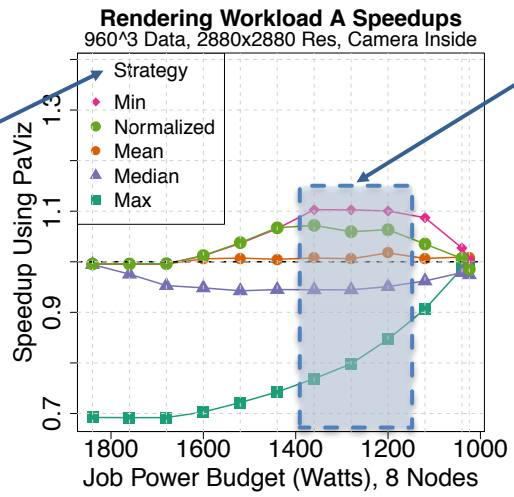
Camera Inside

Phase 2 Changes

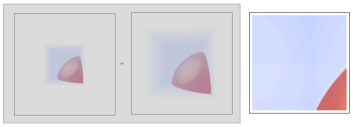
- **Changes**: Compare *Min* strategy from previous phase with remaining power scheduling strategies on Rendering Workload A
- **Goal**: Explore the effects of each strategy on an imbalanced workload

Phase 2 Results: Vary Power Scheduling Strategy

Compare all scheduling strategies.



Min and *Normalized* scheduling strategies perform best under power constraints and in an imbalanced workload.



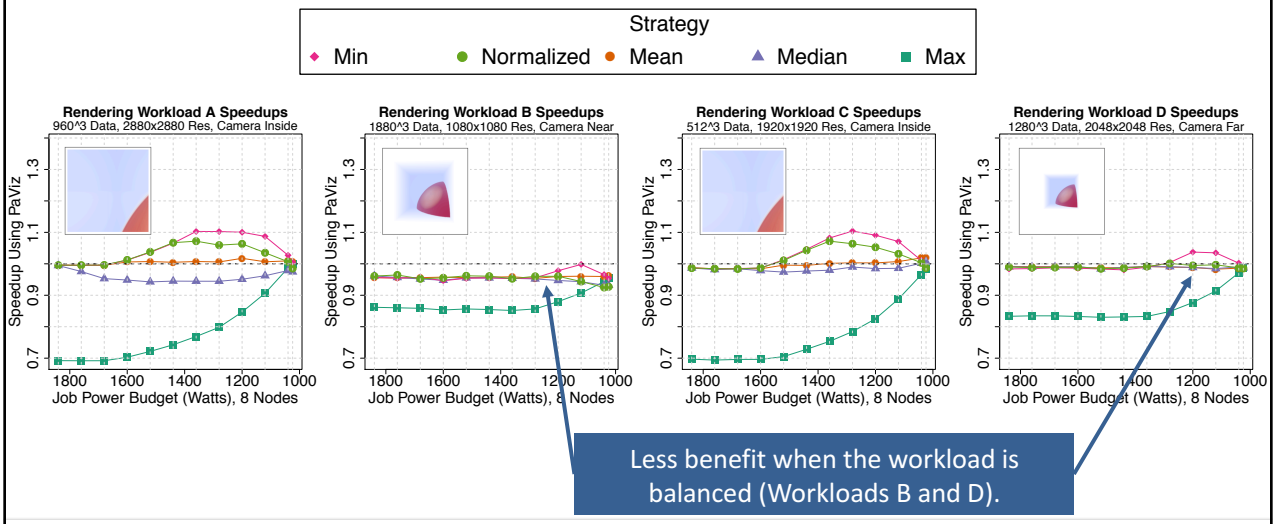
Camera Inside

Phase 3 Changes

- **Changes**: Compare all (5) power scheduling strategies on all rendering workloads
- **Goal**: Explore the effects of each strategy on workloads of varying imbalance levels

Figures for remaining phases are not meant to be readable. I will summarize the findings.

Phase 3 Results: Vary Rendering Workload Configuration

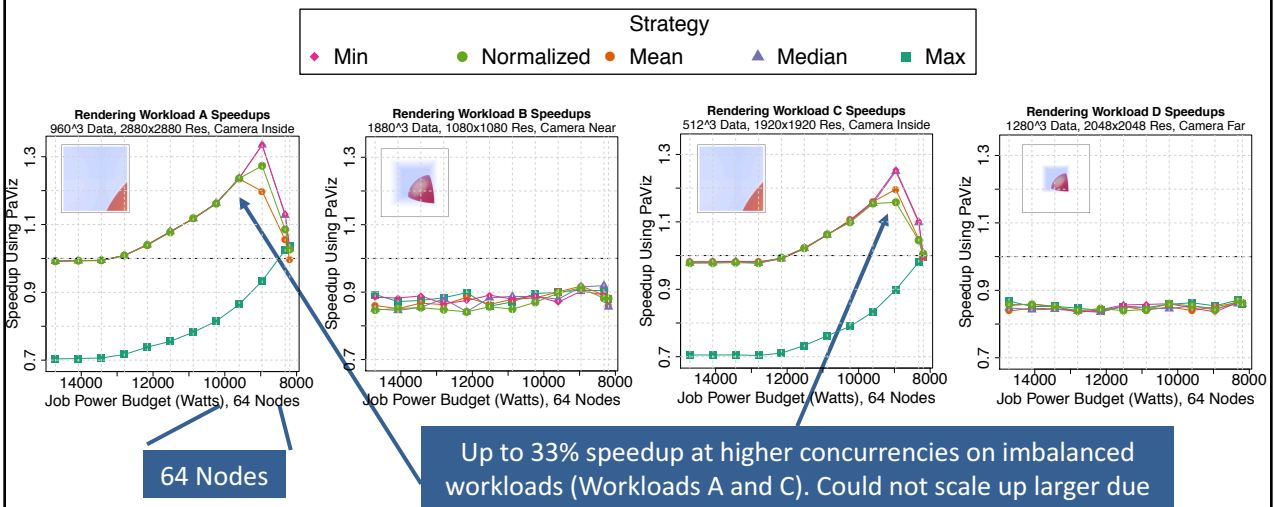


Less benefit when the workload is balanced (Workloads B and D).

Phase 4 Changes

- **Changes**: Repeat previous phase at 64 node concurrency
- **Goal**: Explore the effects at higher scale

Phase 4 Results: Vary Concurrency



64 Nodes

Up to 33% speedup at higher concurrencies on imbalanced workloads (Workloads A and C). Could not scale up larger due to software limitations. Planning to address this in future work.

Presentation Overview

- Background
- PaViz and Research Questions
- Study Overview
- Results
- Takeaways

Conclusion and Future Work

- We present PaViz, a power-adaptive framework for optimizing visualization performance
- Using prediction to re-allocate power among nodes in a job can result in up to 33% speedup over current strategy while using the same power overall
- **Future Work**
  - Can additional performance be realized at higher concurrencies?
  - How will performance models for other algorithms fare?
  - How does PaViz compare to state-of-the-art runtime systems, such as Intel's GEOPM?

Thank you!

- We present PaViz, a power-adaptive framework for optimizing visualization performance
- Using prediction to re-allocate power among nodes in a job can result in up to 33% speedup over current strategy while using the same power overall
- **Future Work**
  - Can additional performance be realized at higher concurrencies?
  - How will performance models for other algorithms fare?
  - How does PaViz compare to state-of-the-art runtime systems, such as Intel's GEOPM?

"PaViz: A Power-Adaptive Framework for Optimizing Visualization Performance"
Stephanie Labasan (labasan1@llnl.gov)


Lawrence Livermore
National Laboratory