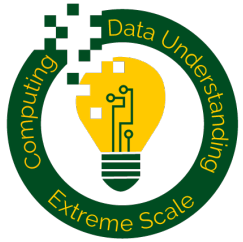
Optimizing Visualization Performance on Power-Limited Supercomputers

Dissertation Defense



Stephanie Labasan

4 March 2019

LLNL-PRES-758484

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

UNIVERSITY OF OREGON

Lawrence Livermore National Laboratory

Dissertation Question

- **“Can Visualization Be Optimized on Power-Limited Supercomputers?”**

- Involves the intersection of two topics:
  - Power-constrained high performance computing
  - Scientific visualization

Outline

- Background and Motivation

- Dissertation Results: Visualization workloads are different than simulation workloads…leads to two questions:
  1. How different are visualization workloads?

     Characterize Data Intensity

  2. How does this affect power optimization strategies?

     Power Scheduling

- Synthesis and Future Work

Outline

- Background and Motivation

- Dissertation Results: Visualization workloads are different than simulation workloads…leads to two questions:
  1. How different are visualization workloads?

     Characterize Data Intensity

  2. How does this affect power optimization strategies?

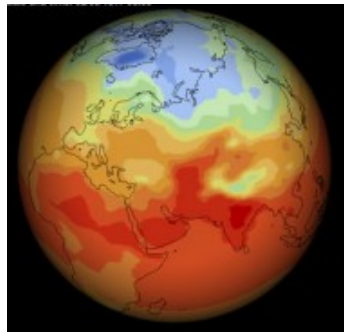
     Power Scheduling

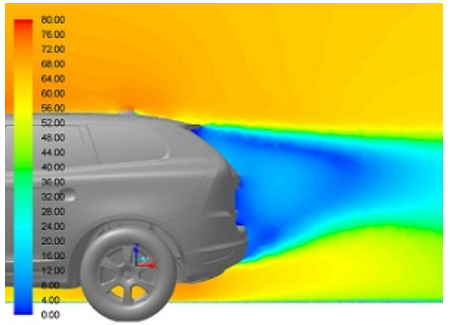
- Synthesis and Future Work

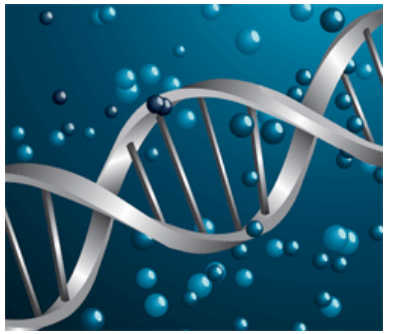
What is high performance computing (HPC)?

- Enabling technology for scientific discoveries through simulation
  - Weather prediction, design simulation, genomics research, medical imaging, and more!
- Leverages computational capacity of several thousand interconnected processing units
- Performance measured in FLOPS = floating-point operations per second
  - Linpack: highly-optimized compute-intensive standard benchmark

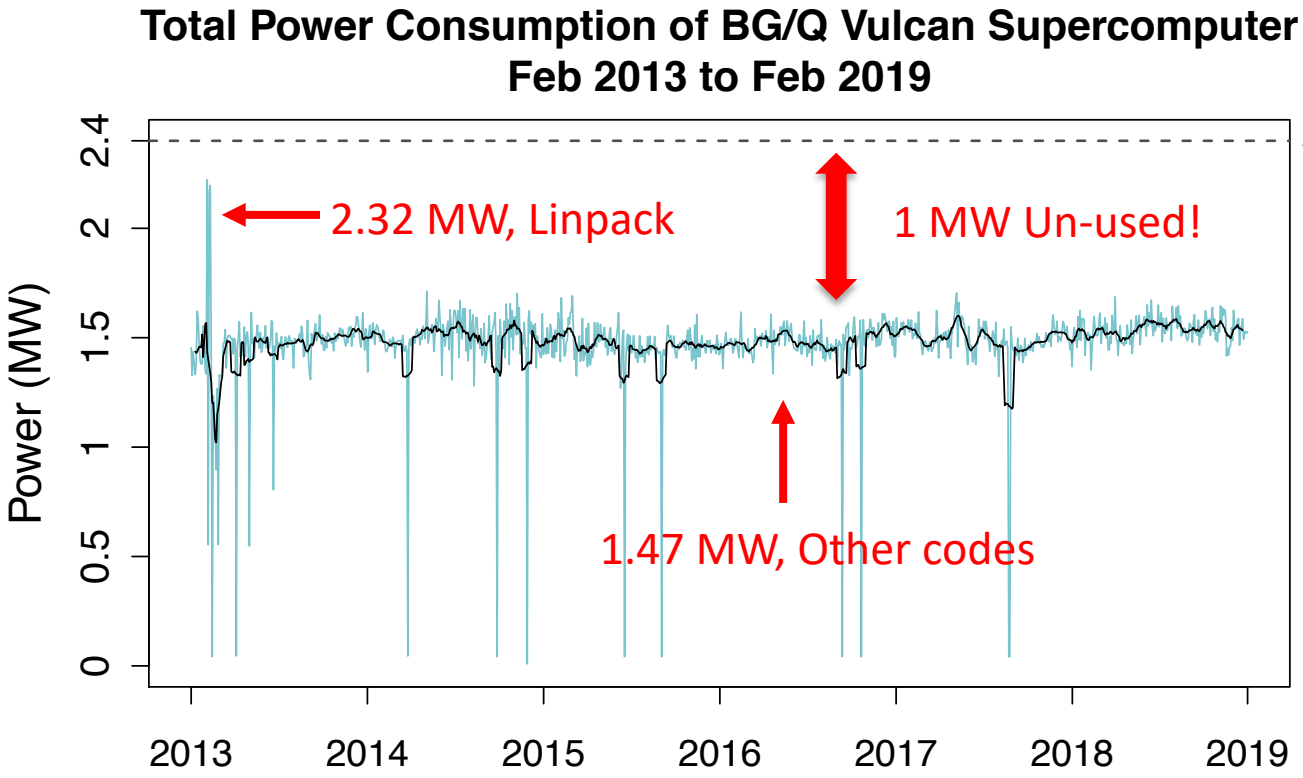
1 TeraFLOPS	10^{12}	1997 (ASCI Red, Sandia)
1 PetaFLOPS	1K TFLOPS	2008 (Roadrunner, LANL)
1 ExaFLOPS	1M TFLOPS	~2023



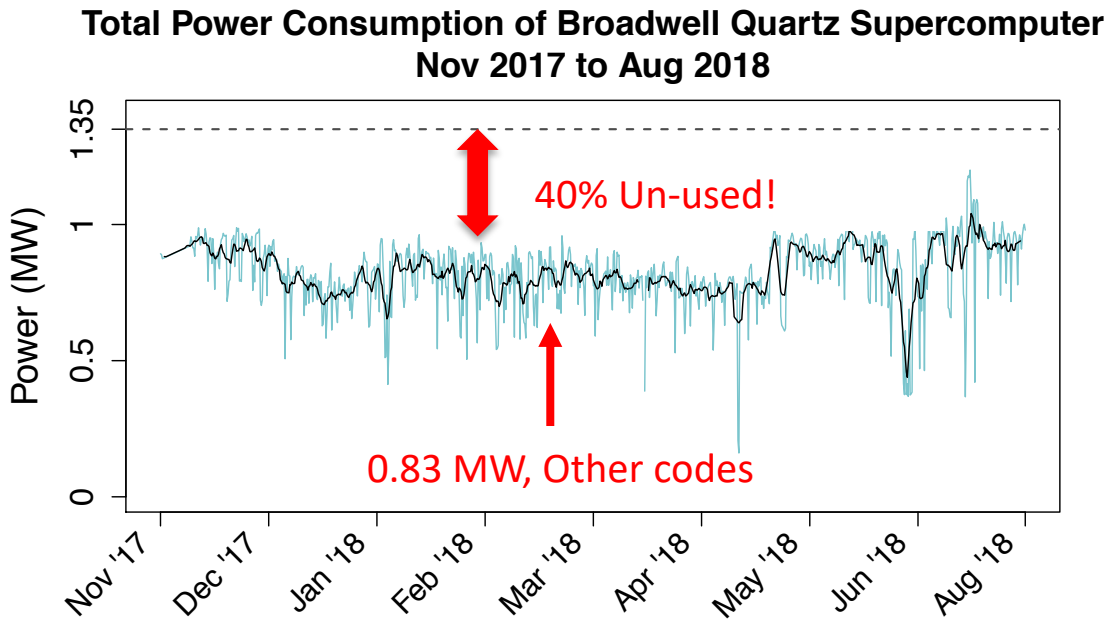




Current system power utilization is sub-optimal



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

Total system power usage and costs are rising

- Power becoming an increasingly scarce resource
  - Expensive cost
  - Limitations enforced by supercomputer facilities

- Target is 20-40 MW system power usage for 1 Exaflop (SciDAC 2016)
  - Need to effectively schedule limited power resources

- **Important premise for this research**: To meet power consumption goals at exascale, need to innovate power-efficient techniques not only in hardware, but also in software

In a power-limited environment, performance of visualization routines will be affected

- Central manager coordinating power allocations across system
- Lots of research focusing on simulations under a power limit
- We are moving towards a workflow where the visualization and analysis are concurrently running with the simulation
- **This research**: visualization component under a power limit

Visualization in a power-limited environment is an important topic

- #1: Visualization is important for scientific insight

- #2: Visualization can use significant resources on the HPC resource (10%-20% of the total runtime), meaning optimizing visualization will be beneficial

- #3: The computing workload for visualization is different than that of simulation
  - Lots of operations (*e.g.*, isovolume, contour, clip, ray tracing)
  - Data intensive, not compute intensive
  - Can be highly variable and imbalanced

Visualization in a power-limited environment is an important topic

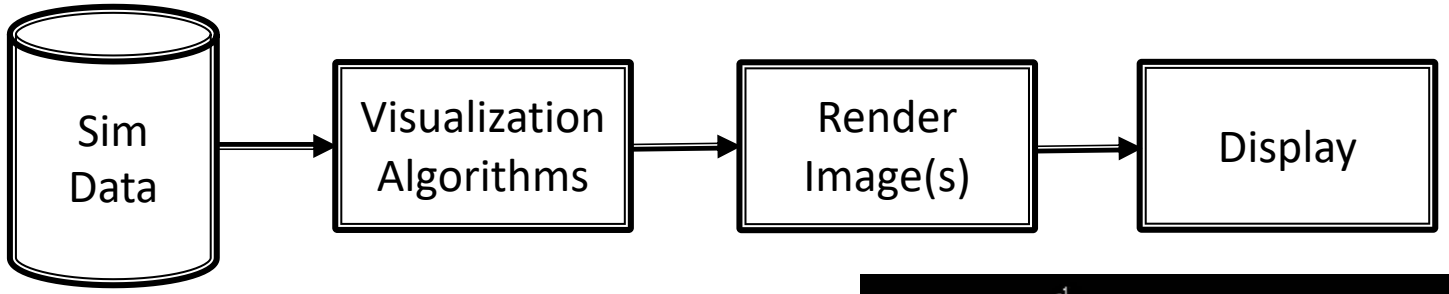
- #1: Visualization is important for scientific insight

- #2: Visualization can use significant resources on the HPC resource (10%-20% of the total runtime), meaning optimizing visualization will be beneficial

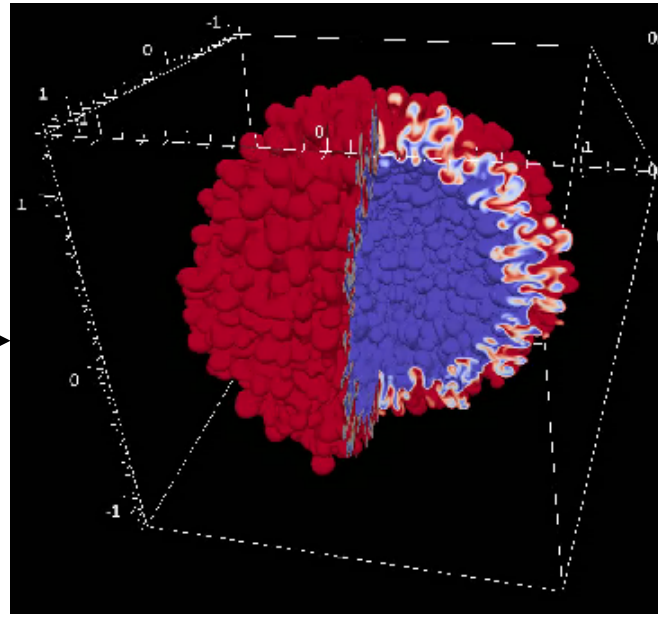
- #3: The computing workload for visualization is different than that of simulation
  - Lots of operations (*e.g.*, isovolume, contour, clip, ray tracing)
  - Data intensive, not compute intensive
  - Can be highly variable and imbalanced

Visualization is important to understanding scientific simulation data

- Central actor in the scientific discovery process:
  - Three use cases:
    - Communicate
    - Validate
    - Explore




```
2.14484 2.13043 2.11357 2.09448 2.07345 2.12172 2.13936 2.15521 2.16894
2.18022 2.1888 2.19483 2.19966 2.20665 2.21965 2.23869 2.25985 2.2795
2.29598 2.30887 2.31819 2.32403 2.32651 2.32577 2.322 2.31549 2.30665
2.29597 2.28405 2.27154 2.25906 2.24724 2.23658 2.22748 2.22021 2.21485
2.2114 2.20969 2.20948 2.21048 2.21231 2.21461 2.21696 2.21895 2.22014
2.22008 2.2183 2.21437 2.2079 2.1986 2.18632 2.17108 2.15304 2.13249
2.10979 2.15935 2.17844 2.1958 2.21118 2.22435 2.23523 2.24414 2.2523
2.26215 2.27613 2.29422 2.31388 2.33243 2.34835 2.36102 2.37024 2.37593
```



Visualization in a power-limited environment is an important topic

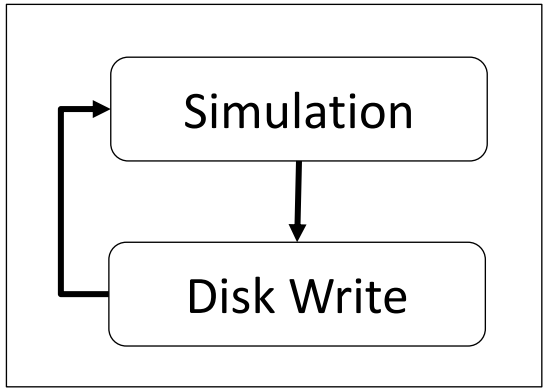
- #1: Visualization is important for scientific insight

- #2: Visualization can use significant resources on the HPC resource (10%-20% of the total runtime), meaning optimizing visualization will be beneficial

- #3: The computing workload for visualization is different than that of simulation
  - Lots of operations (*e.g.*, isovolume, contour, clip, ray tracing)
  - Data intensive, not compute intensive
  - Can be highly variable and imbalanced

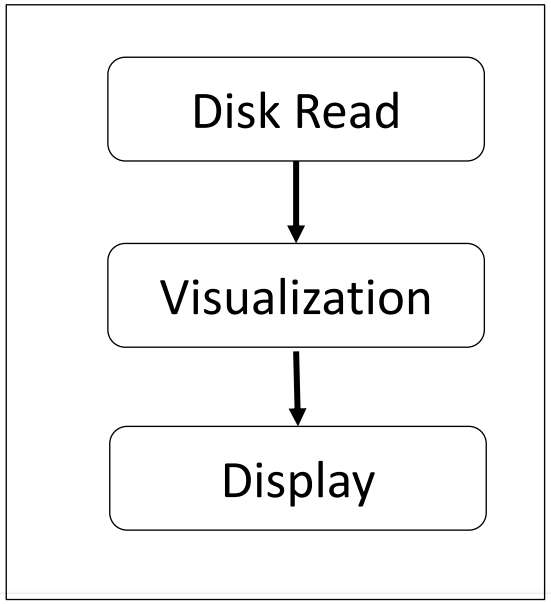
Visualization workflow is traditionally *post hoc*

- First, simulation writes data to disk at regular time steps



Time step 20, 40, 60, …

- Then, visualization program reads in data, performs operations, and displays results to user
  - Data can be explored by scientists at leisurely time scales



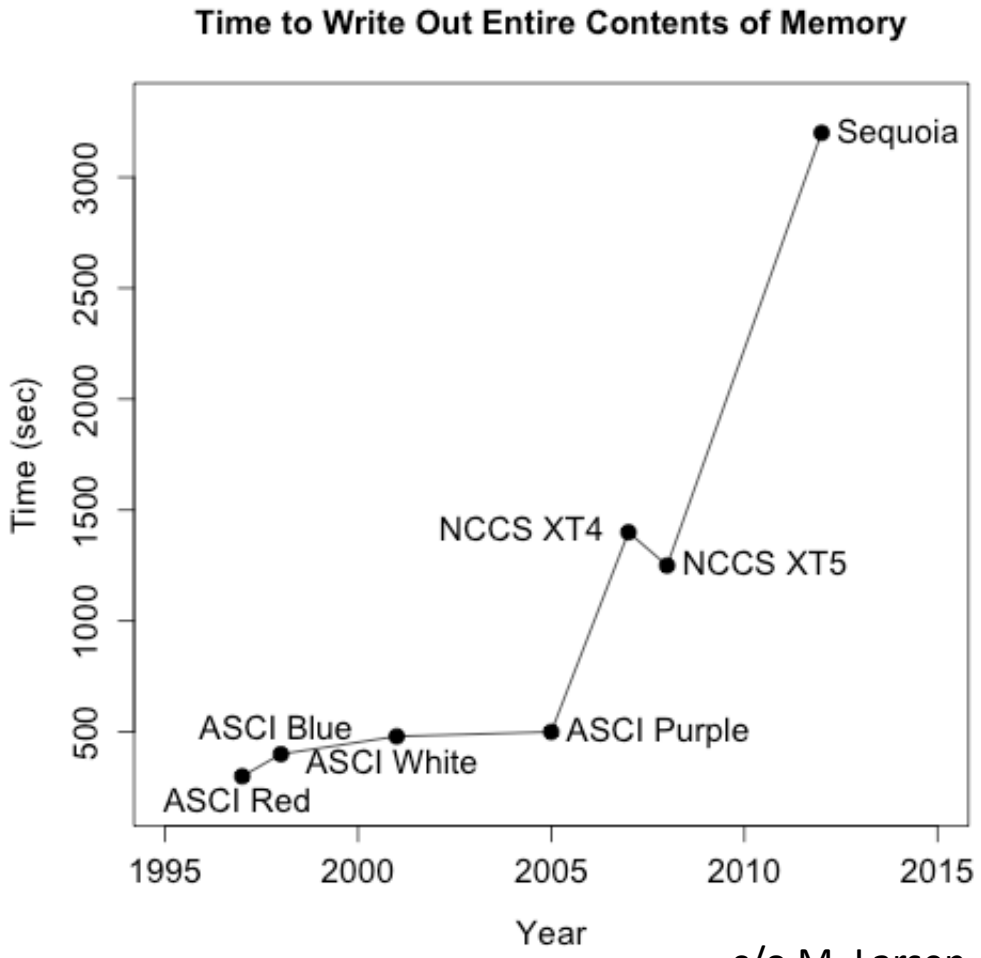
Visualization algorithms, render images

c/o V. Adhinarayanan

Post hoc processing not feasible at exascale

- Compute – and thus ability to generate data – is increasing faster than I/O
- Simulation codes should throttle rate of data saved
- I/O gap limits ability to save data with high frequency
  - Result is *temporal sparsity*

Step 0 Step 500 Step 1000



c/o M. Larsen, B. Whitlock

Transition to *in situ* visualization to bypass I/O gap

- Data analysis and visualization occur *while* simulation is running
- No storage of simulation's state needed
- Changes to HPC ecosystem pose challenges for visualization
  - Total execution time may increase since compute resources are shared

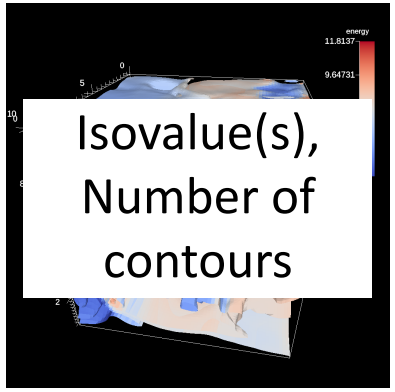
Visualization in a power-limited environment is an important topic

- #1: Visualization is important for scientific insight

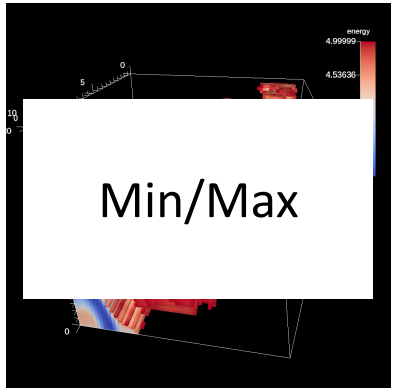
- #2: Visualization can use significant resources on the HPC resource (10%-20% of the total runtime), meaning optimizing visualization will be beneficial

- #3: The computing workload for visualization is different than that of simulation
  - Lots of operations (*e.g.*, isovolume, contour, clip, ray tracing)
  - Data intensive, not compute intensive
  - Can be highly variable and imbalanced

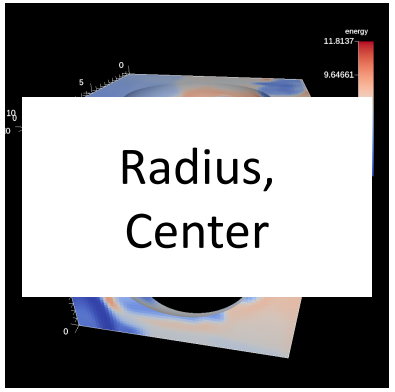
There are many visualization algorithms



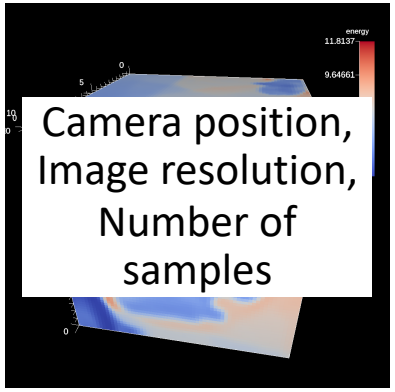
Contour



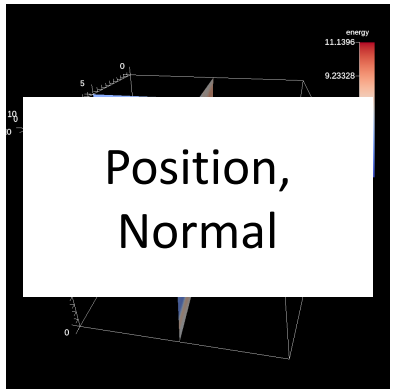
Threshold



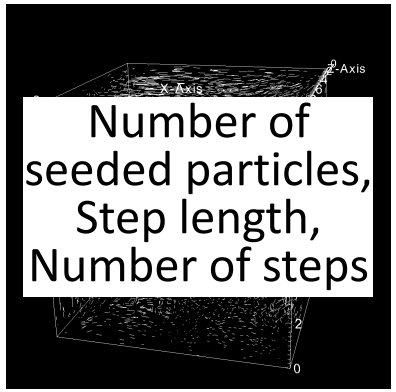
Spherical Clip



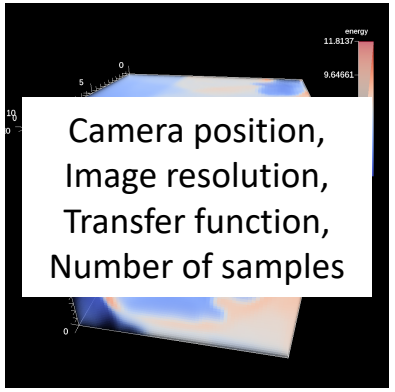
Ray Tracing



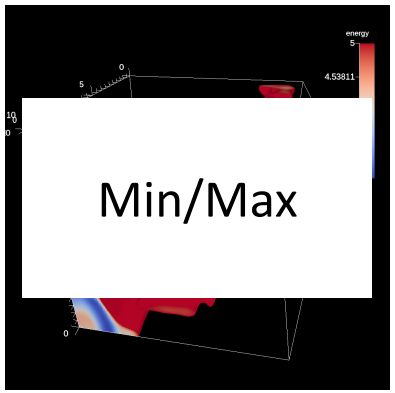
Slice



Particle Advection



Volume Render



Isovolume

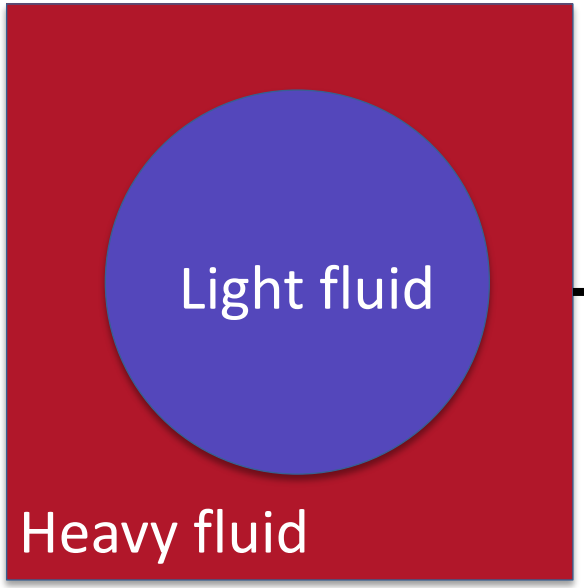
- Execution behaviors differ across algorithms
- Unique configurations for each algorithm
- Pipeline algorithms together
- Output data generated conditionally based on input value

CloverLeaf 200th cycle

In situ load imbalance example: Ares

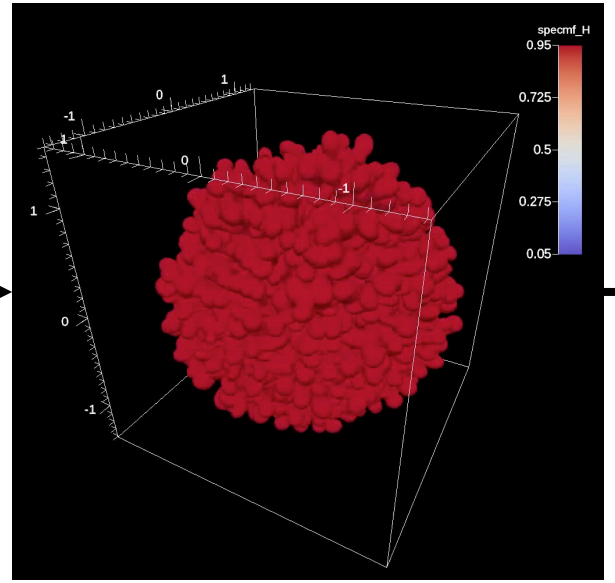
RT Mixing Layer in a Convergent Geometry

- 4π, 1.52B zones
- 29,375 cycles
- ALE Hydrodynamics
- Dynamic Species



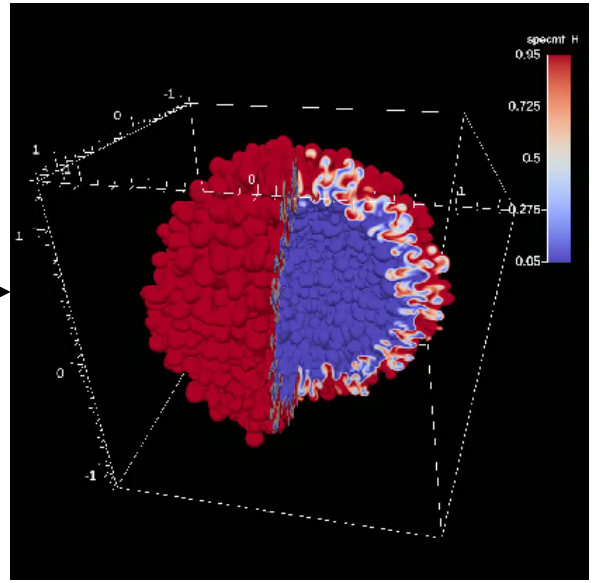
Slice of Ares data set

100% of ranks



Isovolume of mixing layer

~50% of ranks



Clip quarter of data set

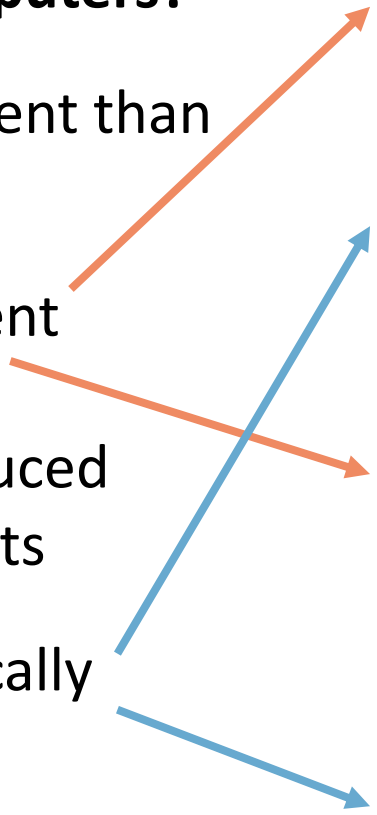
~37.5% of ranks

We cannot address load imbalances directly by redistributing work, since memory is shared between simulation and visualization. Instead, we redistribute power to the source of imbalance.

c/o M. Larsen

Dissertation Plan

- Dissertation Question: **"Can Visualization Be Optimized on Power-Limited Supercomputers?"**
- Focus: Visualization workloads are different than simulation workloads
- First: Understand variation within different visualization workloads
  - Explore execution behaviors with reduced CPU clock frequencies and power limits
- Second: Exploit the variation by dynamically re-allocating power

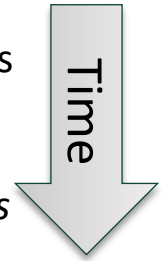


Labasan et al. "*Exploring Tradeoffs Between Power/Performance for a Vis Algorithm.*" IEEE Symposium on Large Data Analysis and Visualization (LDAV), 10/2015.

Labasan et al. "*PaViz: A Power-Adaptive Framework for Optimizing Vis Performance.*" Eurographics Symposium on Parallel Graphics and Visualization (EGPGV), 06/2017.

Labasan et al. "*Power/Performance Tradeoffs for Vis Algorithms.*" Accepted for publication at IEEE International Parallel and Distributed Processing Symposium (IPDPS).

Labasan et al. "*Evaluating Predictive and Adaptive Power Strategies for Optimizing Vis Performance.*" In preparation for submission at IEEE Transactions on Parallel and Distributed Systems (TPDS).



Outline

- Background and Motivation

- Dissertation Results: Visualization workloads are different than simulation workloads…leads to two questions:
  1. How different are visualization workloads?
     - Is there an opportunity for power savings in data intensive algorithms?

  2. How does this affect power optimization strategies?

Characterize Data Intensity

Power Scheduling

- Synthesis and Future Work

Are there power saving opportunities for visualization workloads?

- **Motivation**: Power is becoming a constrained resource in HPC
  - **Goal**: Save energy/power
- **Idea**: Reduce CPU clock frequency or enforce hard power limit
  - These ideas are well-suited specifically for visualization
    - Visualization is data intensive
  - **Proposition for users**: Run X% slower, save Y% in energy/power
- **Study**: Explore a set of visualization algorithms and investigate which factors most affect X and Y

- Two Results:
  - **Initial exploration, show it's possible**
    - Looked at 1 algorithm (isosurfacing) under DVFS

      Labasan et al. "*Exploring Tradeoffs Between Power/Performance for a Vis Algorithm.*" IEEE Symposium on Large Data Analysis and Visualization (LDAV), 10/2015.
  - **Thorough study, understand conditions where variation occurs**
    - Looked at 8 algorithms under power limits

      Labasan et al. "*Power/Performance Tradeoffs for Vis Algorithms.*" Accepted for publication at IEEE International Parallel and Distributed Processing Symposium (IPDPS)

Are there power saving opportunities for visualization workloads?

- **Motivation**: Power is becoming a constrained resource in HPC
  - **Goal**: Save energy/power
- **Idea**: Reduce CPU clock frequency or enforce hard power limit
  - These ideas are well-suited specifically for visualization
    - Visualization is data intensive
  - **Proposition for users**: Run X% slower, save Y% in energy/power
- **Study**: Explore a set of visualization algorithms and investigate which factors most affect X and Y

- Two Results:
  - **Initial exploration, show it's possible**
    - Looked at 1 algorithm (isosurfacing) under DVFS

      Labasan et al. "*Exploring Tradeoffs Between Power/Performance for a Vis Algorithm.*" IEEE Symposium on Large Data Analysis and Visualization (LDAV), 10/2015.
  - **Thorough study, understand conditions where variation occurs**
    - Looked at 8 algorithms under power limits

      Labasan et al. "*Power/Performance Tradeoffs for Vis Algorithms.*" Accepted for publication at IEEE International Parallel and Distributed Processing Symposium (IPDPS).

Save energy? Save power?

Energy $Energy = Power * Time$

- Total work done
- For a single application execution, minimize energy consumption ($$)
- Energy-to-solution

Power $\text{Power} = \frac{Energy}{Time}$

- Rate of energy usage
- Reduce rate at which application consumes energy
- Increase concurrent jobs (maximize throughput)

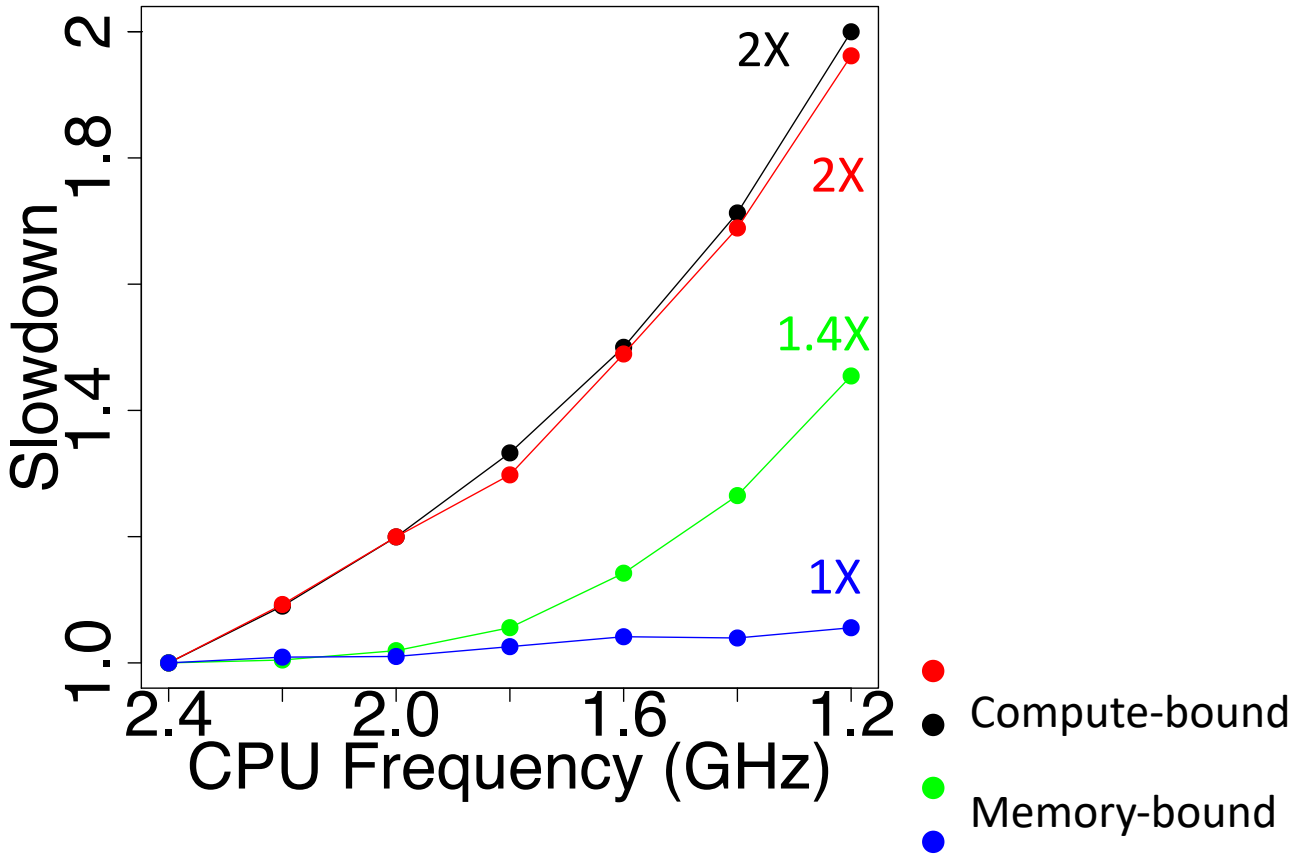
Saving power may/may not save $$ or energy

Run at 40W: take 10 seconds = 400J

Run at 30W: ↓ take 15 seconds = 450J ↑

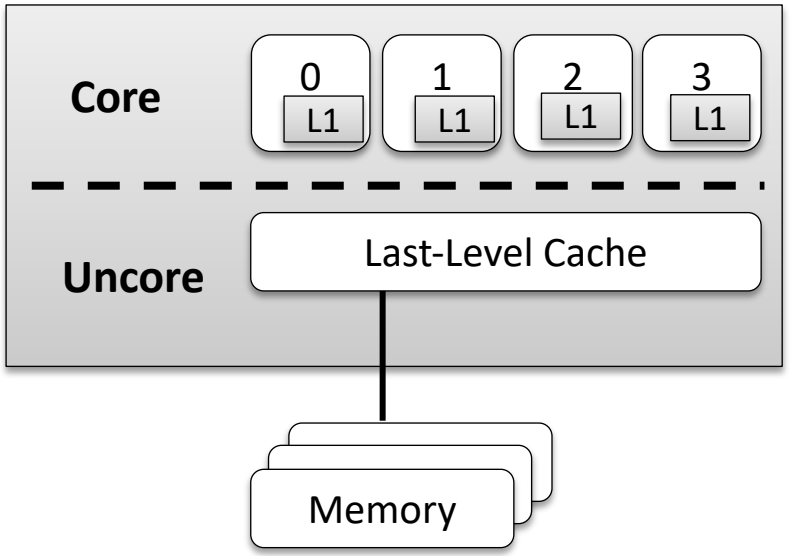
Power-Saving Technique: Reduce Clock Frequency

- **Outcome**: Takes longer to run, but uses less power
- **Why**: Non-linear relationship between frequency and power
  - Reduced clock frequency results in less power consumption
  - But, subcomponents still consume power at the same rate
  - → Will lead to power savings
  - → May lead to energy savings, may not



Shared clock frequency does not work for data intensive applications

- Ideal outcome:
  - Reduce clock frequency
  - Achieve energy/power savings
  - Runtime does not change



- Architecture with separate clock frequencies for cache and compute unit (*e.g.*, Intel Haswell)
  - Enables reduction of clock frequency without impacting cache

- Some visualization algorithms are data intensive

Research Questions & Methodology

- **Data intensity**: If you are forced to reduce the clock frequency, how much slowdown will you see?
- **Energy proposition**: If you are forced to use Y% less energy, you will run X% slower
- **Power proposition**: If you are forced to use Z% less power, you will run X% slower

Explored isosurfacing algorithm with many configurations:

- CPU frequency (7-11 options)
- Data set (6 options)
- Algorithm Implementation (2 options)
- Hardware (2 options)

Phase 1 Results: Vary CPU Clock Frequency

- Reducing clock frequency by 2.2X:
  - Runtime: increases by 1.9X
  - Energy: decreases by 1.4X
  - Power: decreases by 2.7X
- Implementation tends towards compute intensive, but not as strongly as a traditional compute bound benchmark

Isosurfacing algorithm has favorable energy/power savings with reduction in frequency

F	F_{rat}	T	T_{rat}	E	E_{save}	P	P_{rat}
3.5GHz	1X	1.29s	1X	74.3J	1X	57.4W	1X
3.3GHz	1.1X	1.32s	1X	69.4J	1.1X	52.6W	1.1X
3.1GHz	1.1X	1.38s	1.1X	66.7J	1.1X	48.2W	1.2X
2.9GHz	1.2X	1.42s	1.1X	63.4J	1.2X	44.8W	1.3X
2.7GHz	1.3X	1.50s	1.2X	61.5J	1.2X	40.9W	1.4X
2.5GHz	1.4X	1.62s	1.3X	60.9J	1.2X	37.5W	1.6X
2.3GHz	1.5X	1.78s	1.4X	53.7J	1.4X	30.1W	1.9X
2.1GHz	1.7X	1.93s	1.5X	53.8J	1.4X	27.9W	2.1X
2.0GHz	1.8X	1.95s	1.5X	52.1J	1.4X	26.8W	2.2X
1.8GHz	1.9X	2.13s	1.7X	51.1J	1.4X	24.1W	2.4X
1.6GHz	2.2X	2.40s	1.9X	51.4J	1.4X	21.4W	2.7X

Baseline Implementation, Enzo-10M, CPU1, OpenMP

Phase 2 Results: Vary Algorithm Implementation (VTK)

- Compute intensive: F_{rat} and T_{rat} are highly correlated
- Number of instructions retired differs between implementations (102 billion vs. 7 billion)

VTK's implementation shifts instruction mix from data to compute intensive and propositions become less favorable

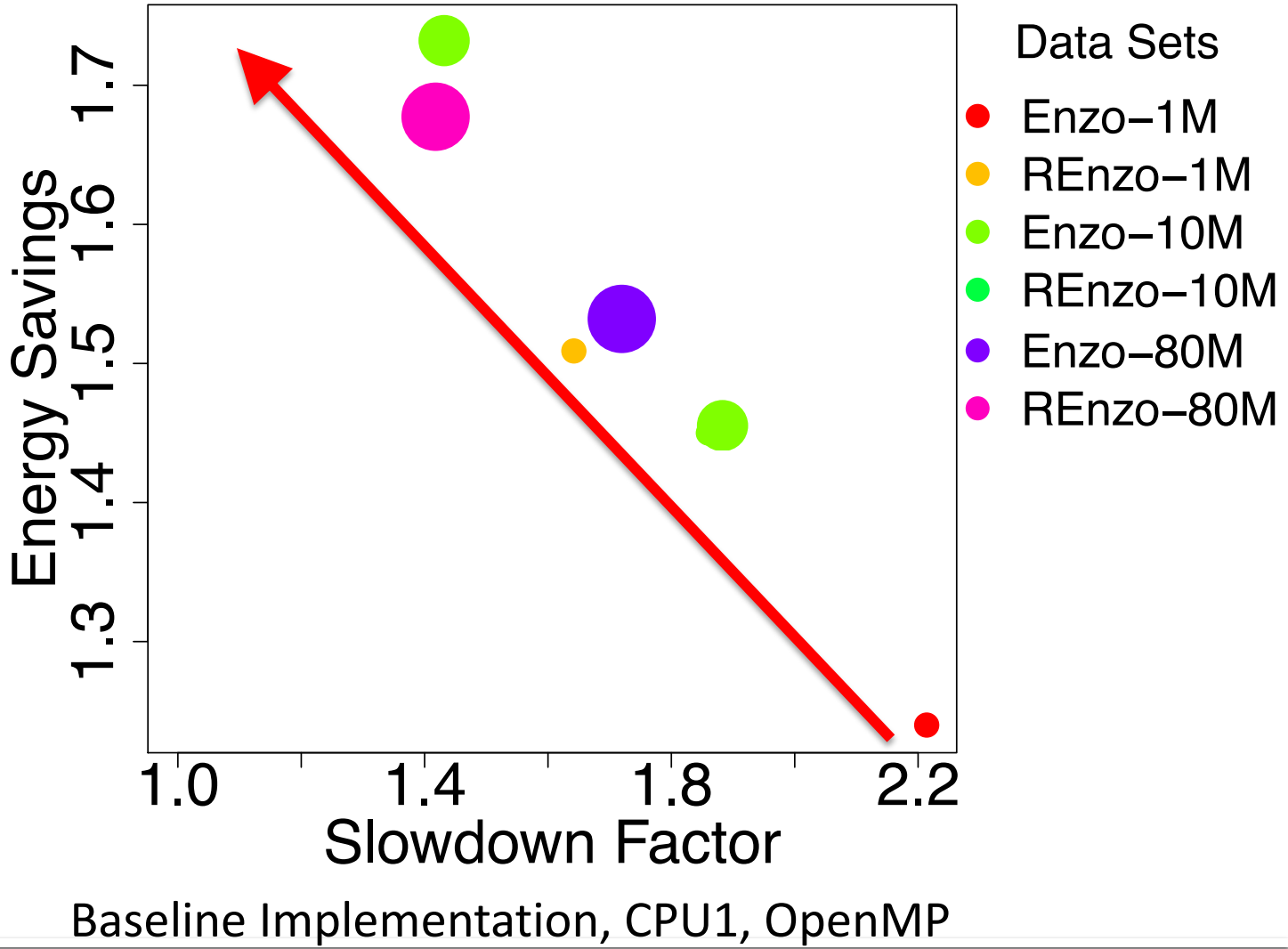
F	F_{rat}	T	T_{rat}	E	E_{save}	P	P_{rat}
3.5GHz	1X	16.06s	1X	1056J	1X	65.8W	1X
3.3GHz	1.1X	16.57s	1X	992J	1.1X	59.9W	1.1X
3.1GHz	1.1X	17.64s	1.1X	950J	1.1X	53.9W	1.2X
2.9GHz	1.2X	19.00s	1.3X	928J	1.1X	48.8W	1.3X
2.7GHz	1.3X	20.85s	1.3X	914J	1.2X	43.9W	1.5X
2.5GHz	1.4X	21.82s	1.4X	876J	1.2X	40.1W	1.6X
2.3GHz	1.5X	24.01s	1.4X	784J	1.3X	32.7W	2X
2.1GHz	1.7X	26.09s	1.7X	763J	1.4X	29.3W	2.2X
2.0GHz	1.8X	27.43s	1.7X	768J	1.4X	28.0W	2.4X
1.8GHz	1.9X	30.67s	1.9X	764J	1.4X	24.9W	2.6X
1.6GHz	2.2X	34.17s	2.1X	756J	1.4X	22.1W	3X

VTK Implementation, Enzo-10M, CPU1, OpenMP

Phase 3 Results: Vary Data Set

- Smallest data set suffers largest impact to runtime with minimal energy savings
- Energy savings: Improves for data sets with increased data intensity
  - Cache misses differ across data sets

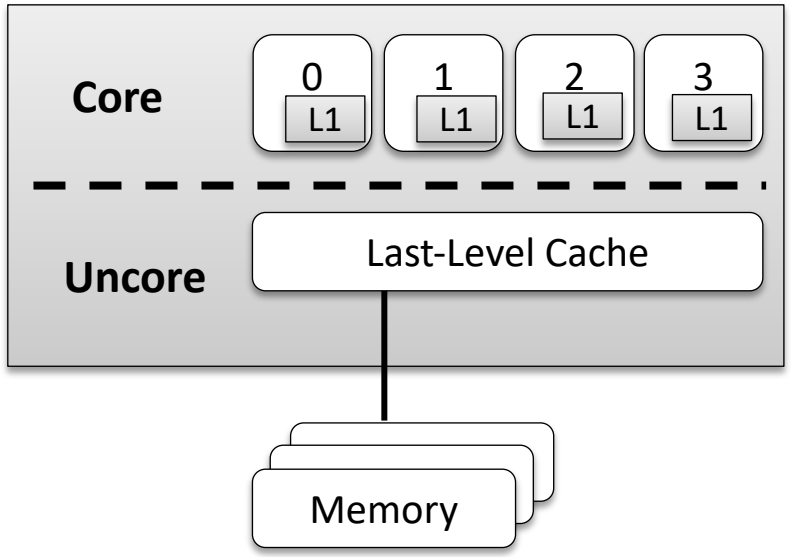
Increasing data set complexity and size produces more favorable tradeoffs



Baseline Implementation, CPU1, OpenMP

Phase 4 Results: Vary Processor Architecture

- Change in clock frequency affects core and uncore on Intel Ivy Bridge
- Propositions now unfavorable



F	F_{rat}	T	T_{rat}	E	E_{rat}	P	P_{rat}
2.4GHz	1X	2.265s	1X	549J	1X	242.4W	1X
2.2GHz	1.1X	2.479s	1.1X	558J	1X	225W	1.1X
2.0GHz	1.2X	2.695s	1.2X	571J	1X	211.9W	1.1X
1.8GHz	1.3X	3.024s	1.3X	573J	1X	189.5W	1.3X
1.6GHz	1.5X	3.385s	1.5X	631J	0.9X	186.4W	1.3X
1.4GHz	1.7X	3.836s	1.7X	668J	0.8X	174.1W	1.4X
1.2GHz	2X	4.466s	2X	697J	0.8X	156W	1.6X

Baseline Implementation, Enzo-10M, **CPU2**, MPI

Hardware with shared clock frequency between core and uncore results in poor tradeoffs

Takeaways demonstrate opportunities for saving energy/power by reducing clock frequencies

- Isosurfacing algorithm is sufficiently data intensive
  - Performance slowdown of 40% led to:
  - Power savings of 2.8X
  - Energy savings of 1.9X
- Increasing data set complexity (*e.g.*, size, irregular access patterns) results in better tradeoffs between runtime and energy/power
- VTK implementation shifts instruction counts, resulting in less favorable tradeoffs between runtime and energy/power savings
- **Next research question**: Is there variation among other visualization algorithms?

Are there power saving opportunities for visualization workloads?

- **Motivation**: Power is becoming a constrained resource in HPC
  - — **Goal**: Save energy/power
- **Idea**: Reduce CPU clock frequency or enforce hard power limit
  - — These ideas are well-suited specifically for visualization
    - Visualization is data intensive
  - — **Proposition for users**: Run X% slower, save Y% in energy/power
- **Study**: Explore a set of visualization algorithms and investigate which factors most affect X and Y

- Two Results:
  - — **Initial exploration, show it's possible**
    - Looked at 1 algorithm (isosurfacing) under DVFS

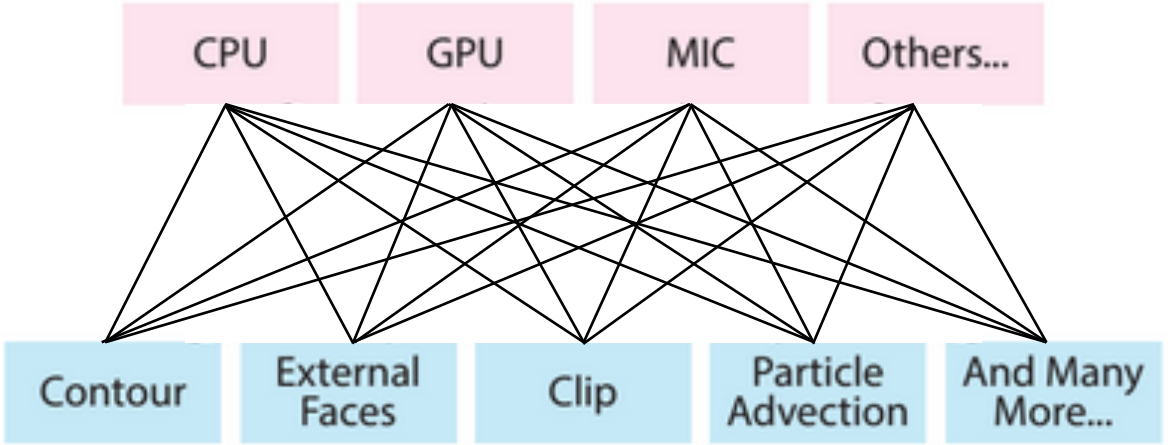
    Labasan et al. "*Exploring Tradeoffs Between Power/Performance for a Vis Algorithm.*" IEEE Symposium on Large Data Analysis and Visualization (LDAV), 10/2015.
  - — **Thorough study, understand conditions where variation occurs**
    - Looked at 8 algorithms under power limits

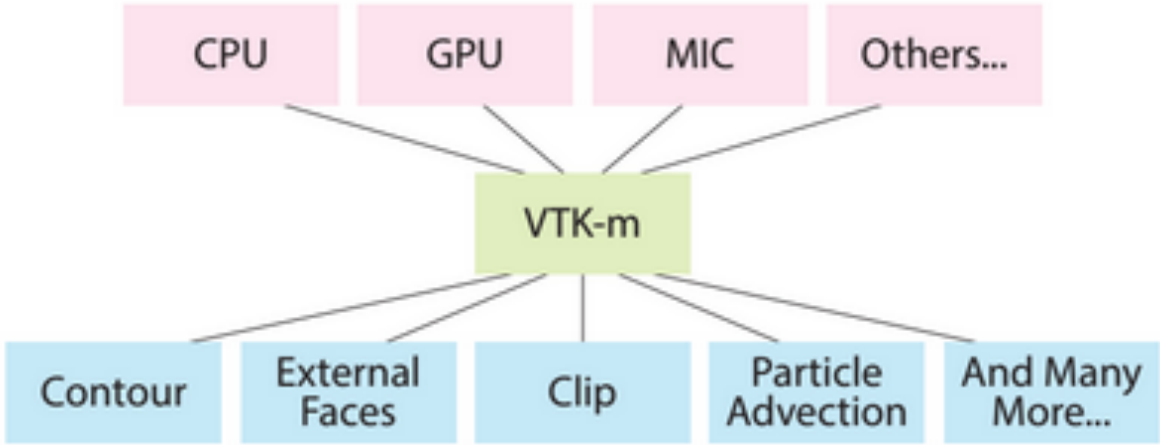
    Labasan et al. "*Power/Performance Tradeoffs for Vis Algorithms.*" Accepted for publication at IEEE International Parallel and Distributed Processing Symposium (IPDPS).

VTK-m: Accelerating VTK for many-core architectures

- Targets on-node parallelism using data parallel primitives

- Framework for simplifying the design of visualization algorithms on future architectures

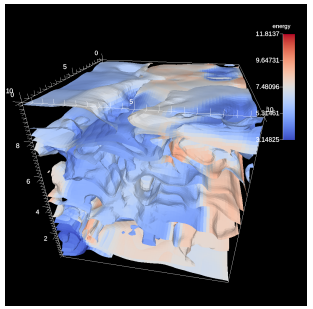


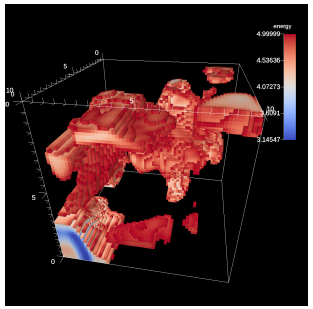


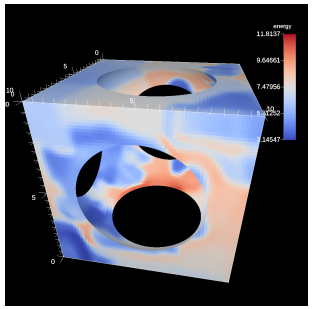
http://m.vtk.org

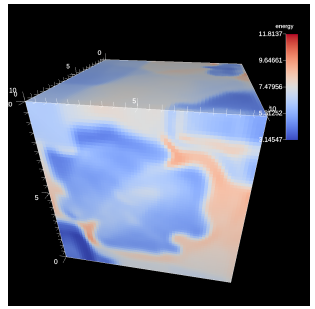
Study Factors

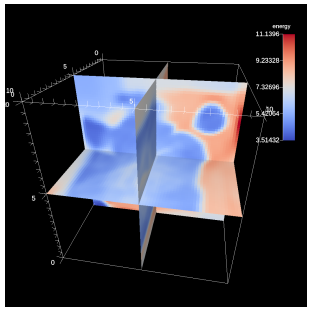
- Processor-level power limit → LLNL RZTopaz Broadwell node
  - 9 options
  - CPU cap ranges from 120W (TDP) down to 40W
- Visualization algorithm
  - 8 options

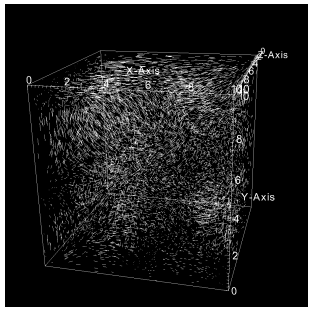

Contour


Threshold

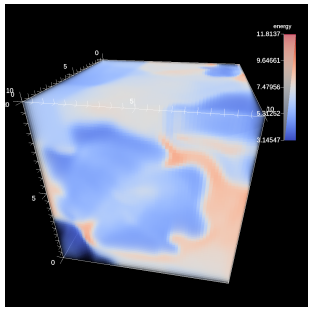

Spherical Clip

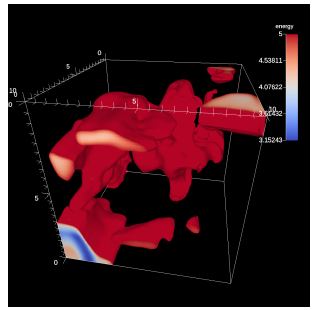

Ray Tracing


Slice



Particle Advection

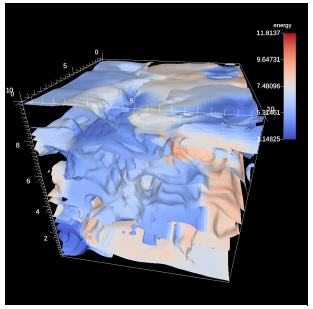

Volume Render

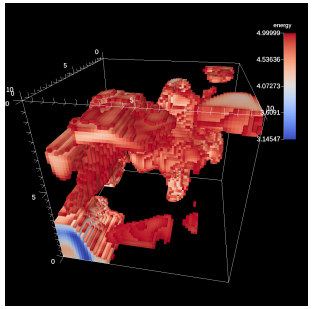

Isovolume

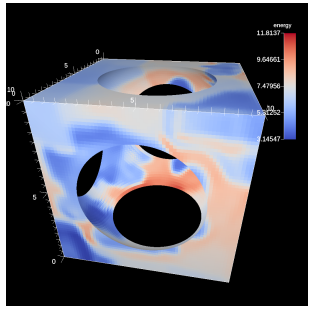
Result from this study:
Most algorithms provide opportunities for power savings

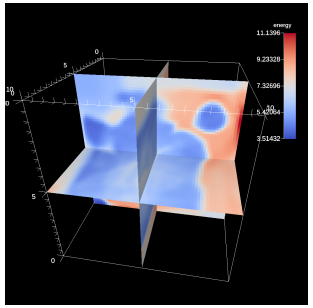
Power Opportunity ✓

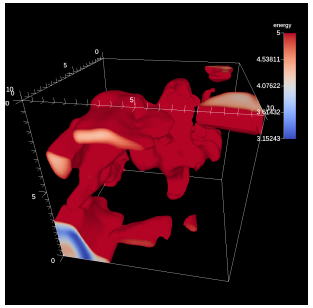
- Performance remains constant until lowest power limit

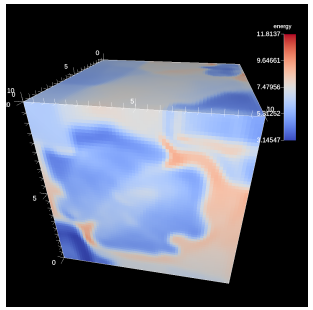

Contour


Threshold


Spherical Clip

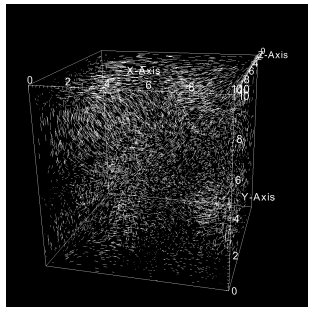

Slice


Isovolume

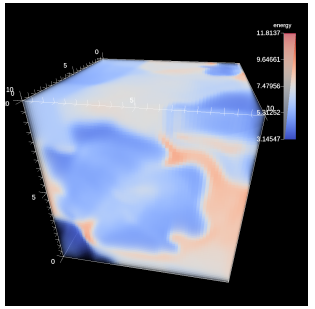

Ray Tracing*

Power Sensitive ✗

- Performance degrades relative to power limit
- Compute-bound workloads



Particle Advection

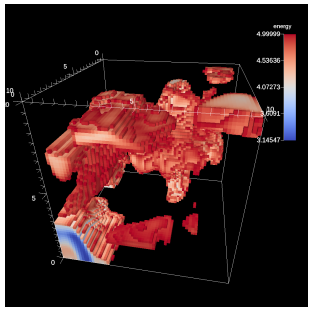

Volume Render

*Ray tracing includes 3 operations: (1) find external faces (50%), (2) build spatial acceleration structure, and (3) trace rays

Comparing power opportunity and power sensitive algorithms (Consider 1 example from each)

Power Opportunity: **Threshold**

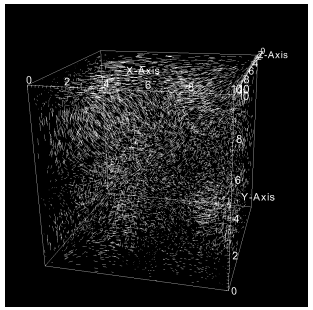
- Iterate over the cells in data set
- Remove cells that do not have values in the specified range
- Simple computation dominated by loads and stores of each cell



Threshold

Power Sensitive: **Particle Advection**

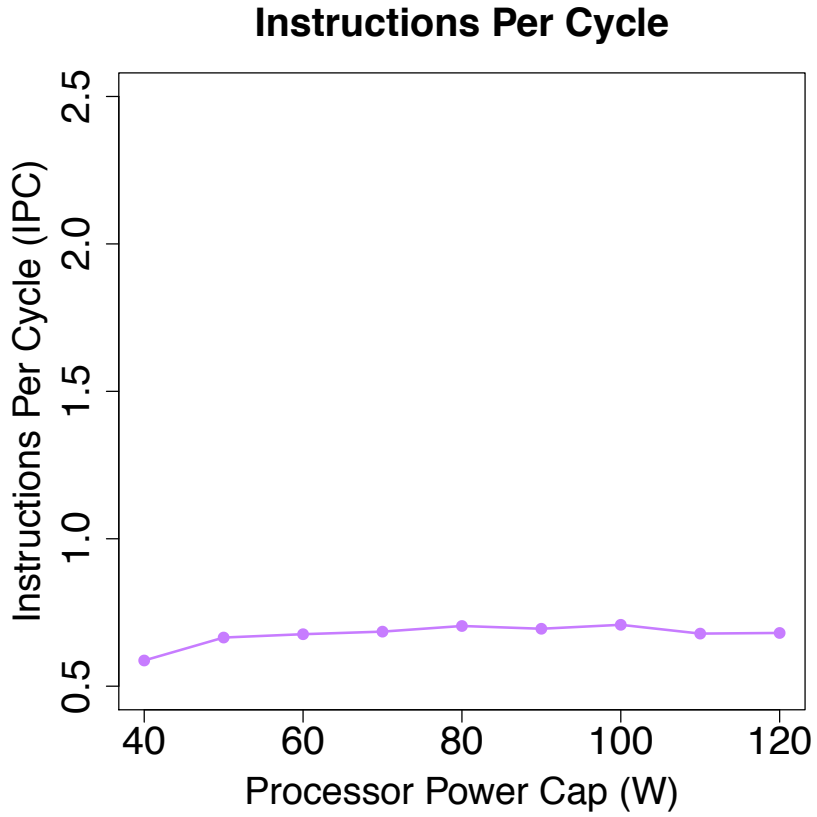
- Place a massless particle at a seed location
- Displace the particle according to the vector field
- Result is an “integral curve” corresponding to the trajectory the particle travels – Euler, RK4, …

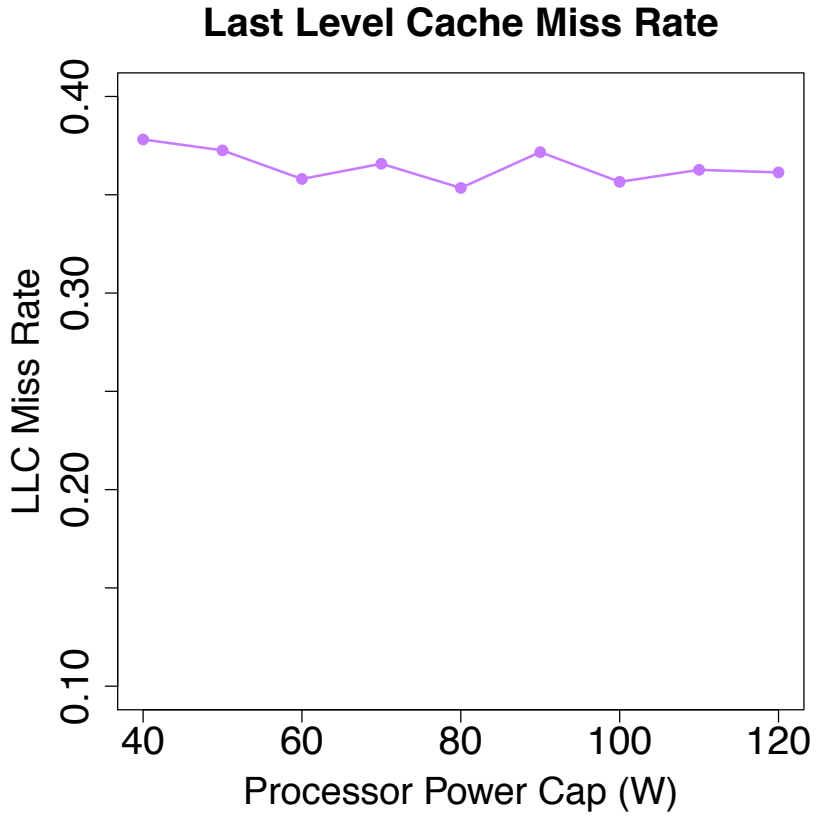


Particle Advection

Power opportunity algorithms are data intensive, while power sensitive algorithms are compute intensive

Power Opportunity



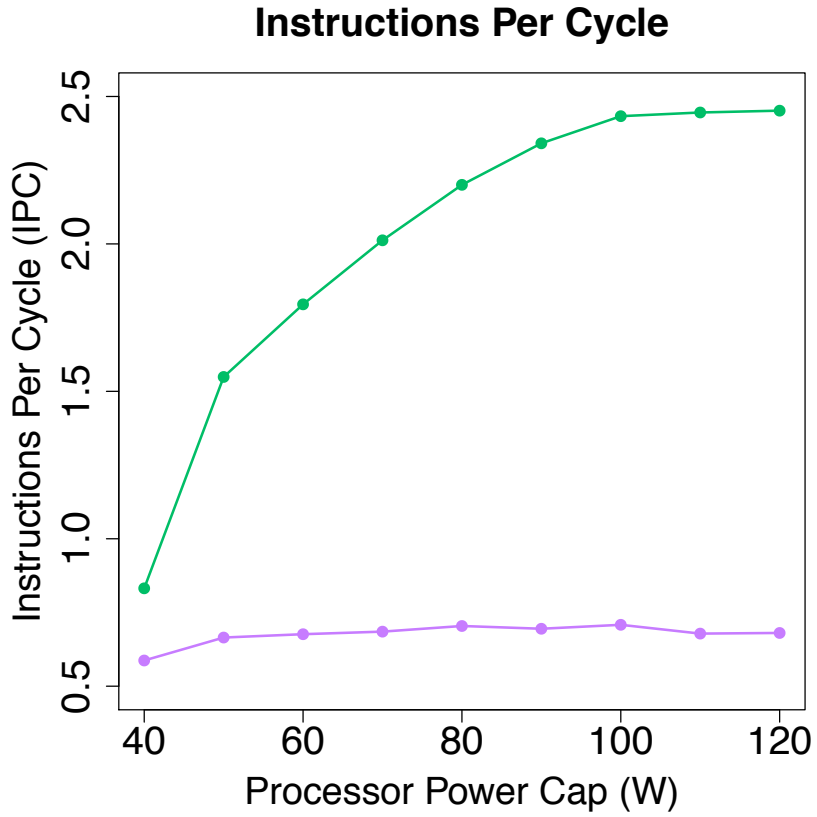


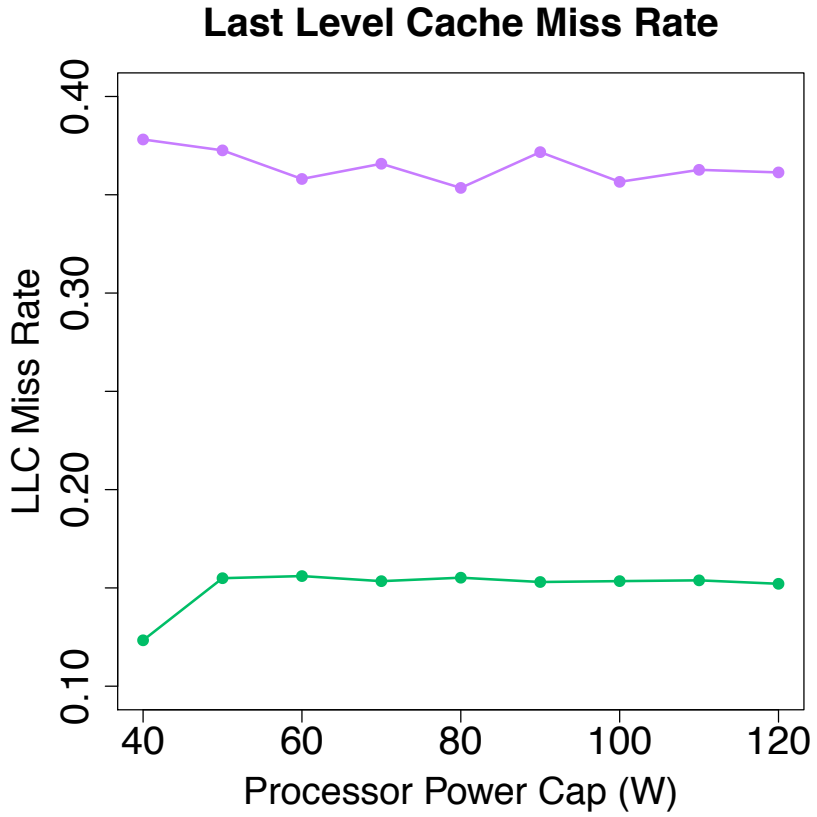
Power Opportunity

Algorithm
threshold

Power opportunity algorithms are data intensive, while power sensitive algorithms are compute intensive

Power Sensitive





Power Sensitive

Algorithm
threshold
particleadvection

Takeaways inform how visualization algorithms can adapt to imposed power limits

- VTK-m implementations are significantly data intensive to avoid a slowdown at a reduced power limit

- Most algorithms consume low amounts of power, providing opportunities for energy/power savings (*e.g.*, data intensive)

Outline

- Background and Motivation

- Dissertation Results: Visualization workloads are different than simulation workloads…leads to two questions:
  1. How different are visualization workloads?

Characterize Data Intensity

  2. How does this affect power optimization strategies?
     - Can we exploit execution behaviors to realize additional performance?

Power Scheduling

- Synthesis and Future Work

How can we exploit execution behaviors to improve performance of visualization algorithms?

- Visualization algorithms can have more variance than simulation codes
  - Motivates an alternative approach for redistributing power: using prediction (rather than adaptation)

- Two Results:
  - **Show prediction is viable**
    - Showed benefit incorporating prediction for volume rendering
  - **Show prediction is beneficial**
    - Extended EGPGV17, compare with runtime system using adaptation (GEOPM)

Labasan et al. "*PaViz: A Power-Adaptive Framework for Optimizing Vis Performance.*" Eurographics Symposium on Parallel Graphics and Visualization (EGPGV), 06/2017.

Labasan et al. "*Evaluating Predictive and Adaptive Power Strategies for Optimizing Vis Performance.*" In preparation for submission at IEEE Transactions on Parallel and Distributed Systems (TPDS).

How can we exploit execution behaviors to improve performance of visualization algorithms?

- Visualization algorithms can have more variance than simulation codes
  - Motivates an alternative approach for redistributing power: using prediction (rather than adaptation)

- Two Results:
  - **Show prediction is viable**
    - Showed benefit incorporating prediction for volume rendering
  - **Show prediction is beneficial**
    - Extended EGPGV17, compare with runtime system using adaptation (GEOPM)

Labasan et al. "*PaViz: A Power-Adaptive Framework for Optimizing Vis Performance.*" Eurographics Symposium on Parallel Graphics and Visualization (EGPGV), 06/2017.

Labasan et al. "*Evaluating Predictive and Adaptive Power Strategies for Optimizing Vis Performance.*" In preparation for submission at IEEE Transactions on Parallel and Distributed Systems (TPDS).

Use performance predictions to identify imbalance

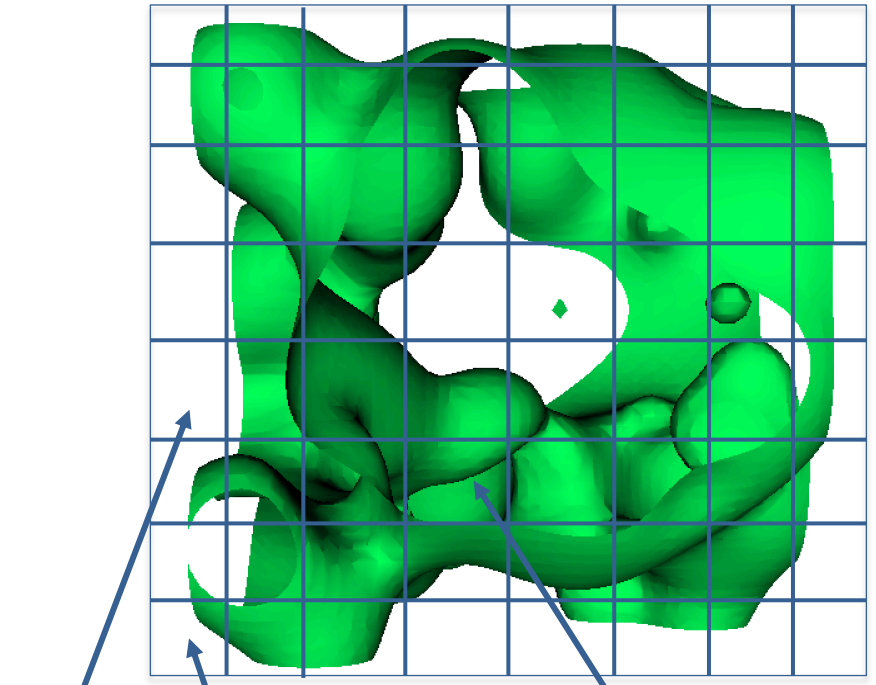
- **Goal**: Improve power utilization to maximize performance in a power-limited environment

- **Method**: Enforce power limit, 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 redistribute per-node power to where it will do the most good

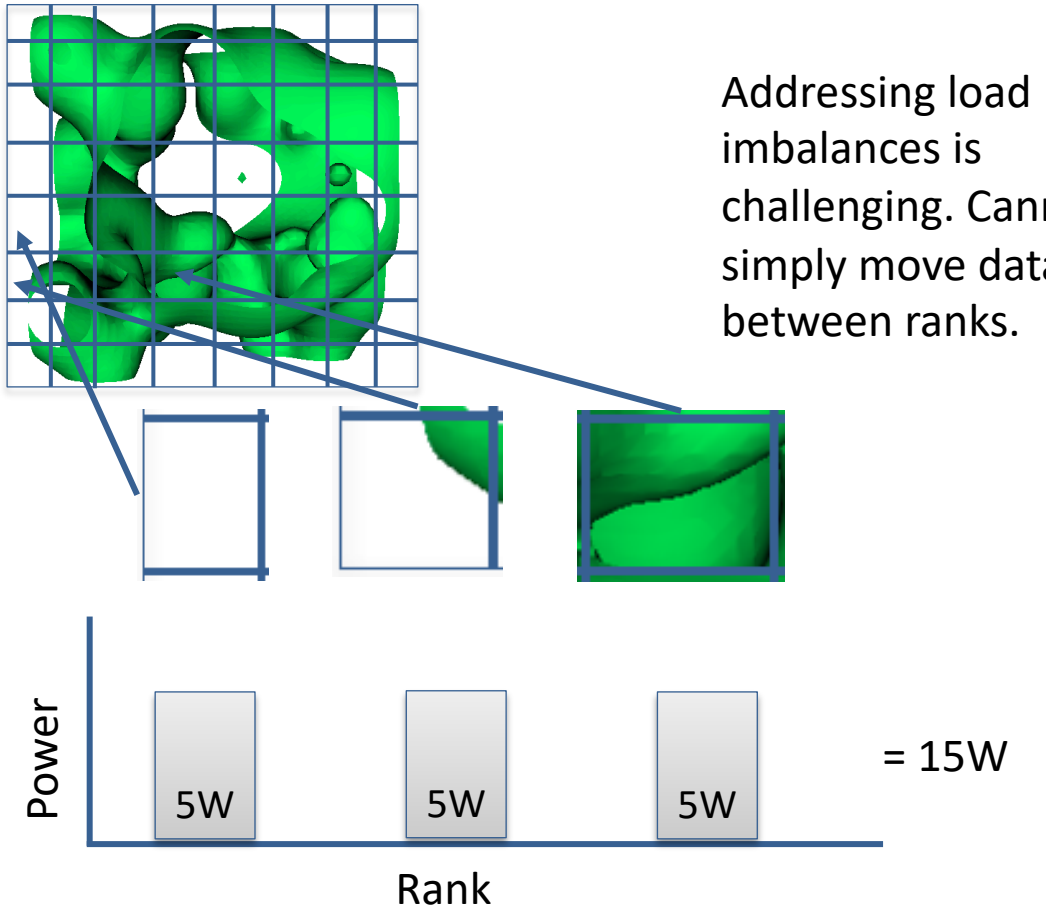
Imbalance in visualization routines poses an opportunity for performance improvements

- 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

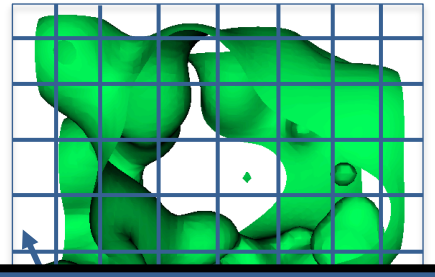
- Canonical example = isosurfacing
  - Geometry (*i.e.*, work) will not be symmetrical
  - Work per node will vary...some nodes will have lots of work, others with no work
- **Give** power to rank with lots of work
- **Take** power away from rank with less work
- Improve overall runtime



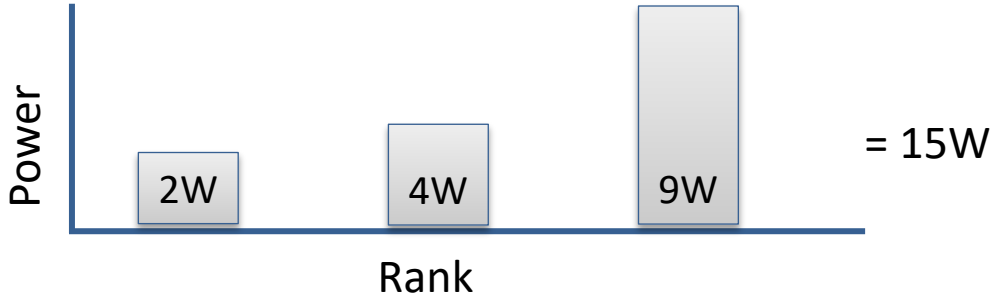
Addressing load imbalances is challenging. Cannot simply move data between ranks.

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 ha
    work
- **Give** pov
- **Take** power away from rank with less work
- Improve overall runtime



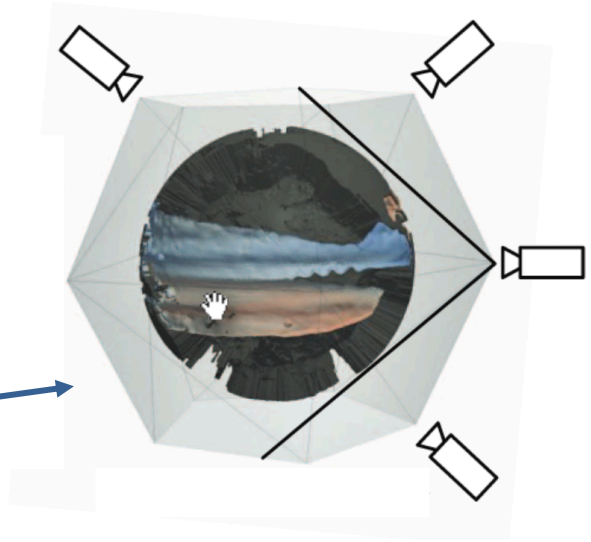
So, which ranks have more work and which have less?
We need to know this before they start executing.
Idea: Use performance modeling.



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

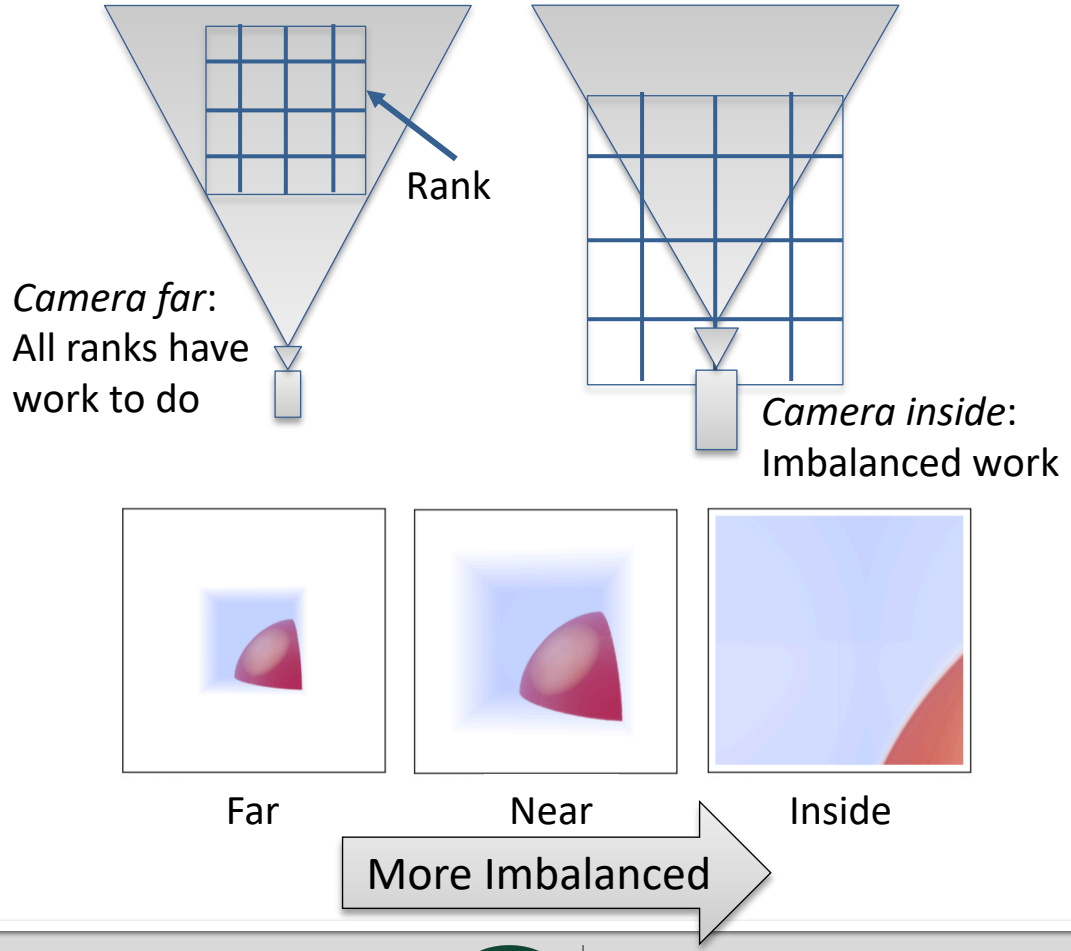
- System focus is on rendering
  - Key phase in the visualization pipeline
  - Incorporates an existing performance model
  - Typically a very fast operation, but could be very time-consuming (CINEMA)



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

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

- (1) MPI+TBB task per node for maximum memory allocation

- 8 tasks (ranks) = 192 cores
- 64 tasks (ranks) = 1,472 cores

Study Overview

- Rendering workload
- MPI task concurrency
- Power scheduling strategy

- (5) strategies explored: *Min*, *Normalized*, *Mean*, *Median*, *Max*

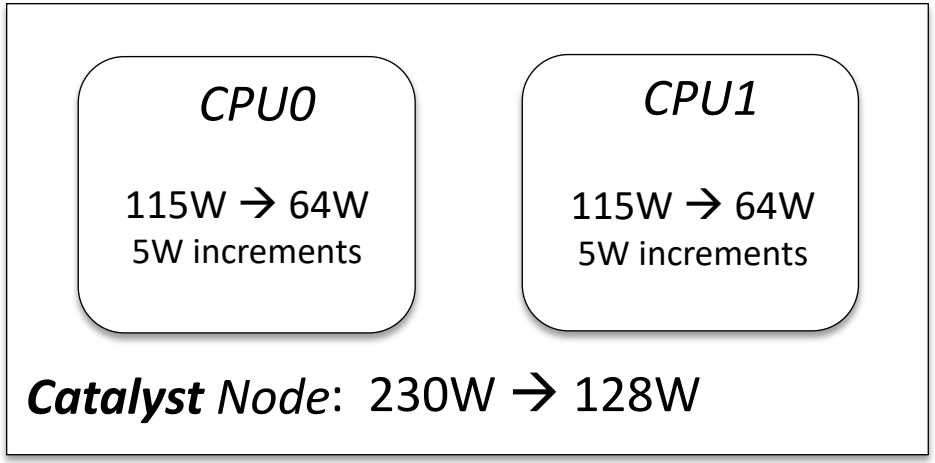
$$P_i = \frac{t_i - t_{min}}{\sum(t_i - t_{min})} \quad \textit{Min}$$

$$P_i = \frac{t_{max} - t_i}{\sum(t_{max} - t_i)} \quad \textit{Max}$$

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

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.

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

* 1 MPI+TBB task per node

Methodology

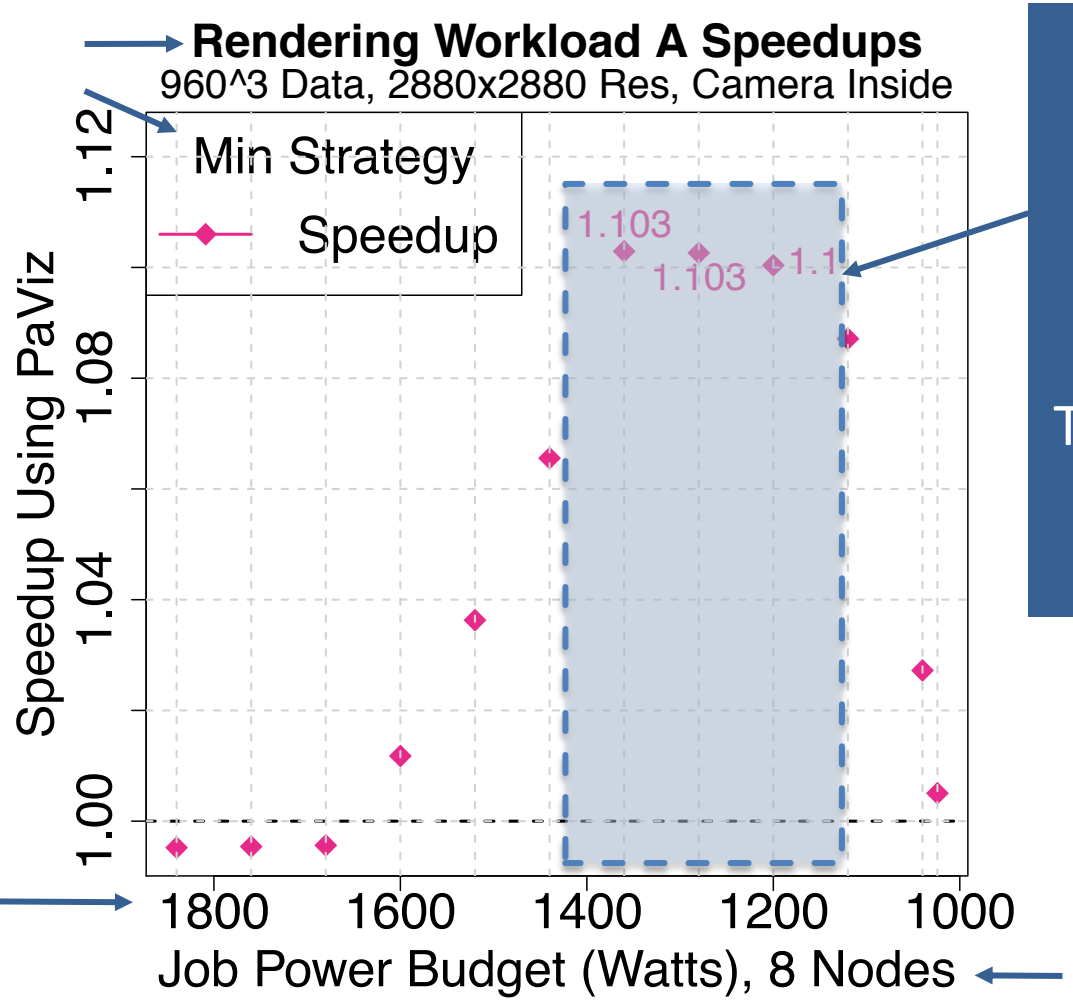
- Rendering workload
- MPI task concurrency
- Power scheduling strategy
- Job power budget

- Varied across all factors

- Vary job power budget for a single rendering workload, 8 MPI tasks, *Min* power scheduling strategy

- Subsequent results varies 1 factor to explore effects

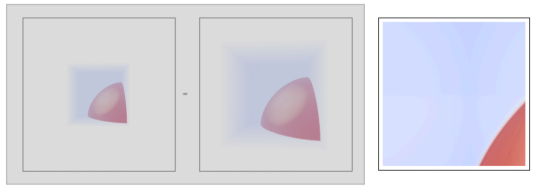
Phase 1 Results: Vary Job Power Budget



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 limit



Camera Inside

More Imbalanced

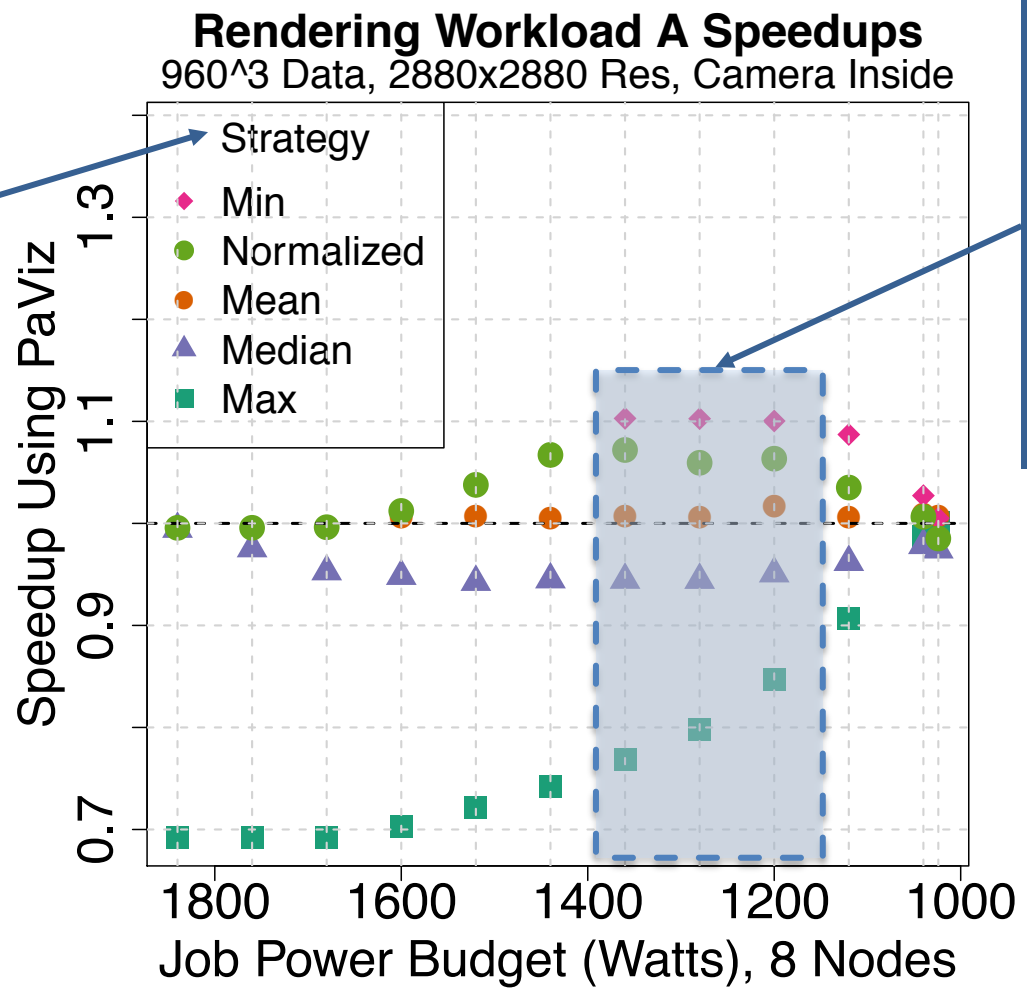
Phase 2 Changes

- **Changes**: Compare *Min* strategy from previous phase with remaining power scheduling strategies on Rendering Workload A

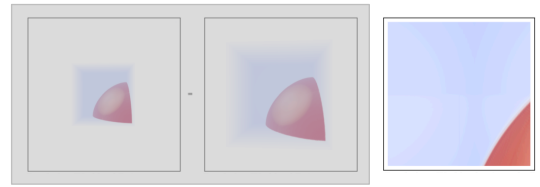
- **Goal**: Explore the effects of each strategy on an imbalanced workload

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

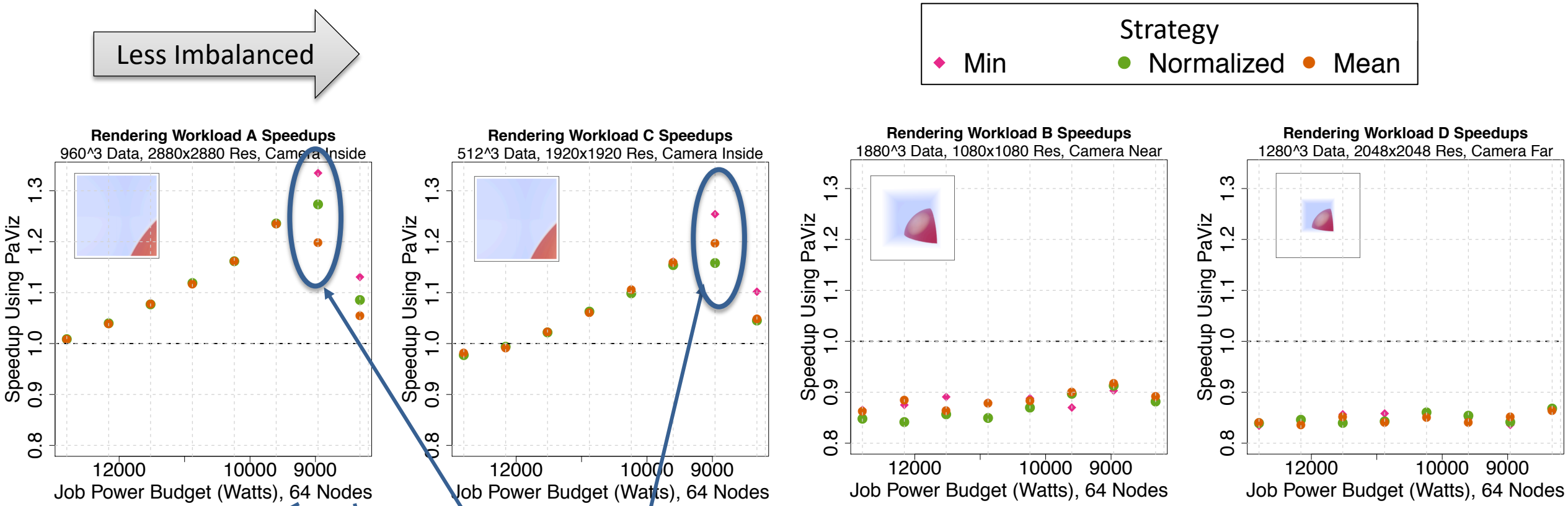
More Imbalanced

Phase 3 Changes

- **Changes**: Compare best performing scheduling strategies on all rendering workloads at high concurrency

- **Goal**: Explore the effects of best strategies on workloads of varying imbalance levels at scale

Phase 3 Results: Compare Rendering Configuration at Scale



64 Nodes

Up to 33% speedup at higher concurrencies on imbalanced workloads. Could not scale up larger due to software limitations.

Takeaways demonstrate the viability of using prediction to re-allocate power

- 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

- **Next research questions**:
  - Can additional performance be realized at higher concurrencies?
  - How will adaptation compare to prediction for visualization workloads?

How can we exploit execution behaviors to improve performance of visualization algorithms?

- Visualization algorithms can have more variance than simulation codes
  - Motivates an alternative approach for redistributing power: using prediction (rather than adaptation)

- Two Results:
  - **Show prediction is viable**
    - Showed benefit incorporating prediction for volume rendering
  - **Show prediction is beneficial**
    - Extended EGPGV17, compare with runtime system using adaptation (GEOPM)

Labasan et al. "*PaViz: A Power-Adaptive Framework for Optimizing Vis Performance.*" Eurographics Symposium on Parallel Graphics and Visualization (EGPGV), 06/2017.

Labasan et al. "*Evaluating Predictive and Adaptive Power Strategies for Optimizing Vis Performance.*" In preparation for submission at IEEE Transactions on Parallel and Distributed Systems (TPDS).

Two approaches for power scheduling

Predictive – PaViz

- Use a performance model of execution times to assign power before execution
- High cost associated in creating model, and fitting it to specific-architecture
- Better suited for highly irregular workloads

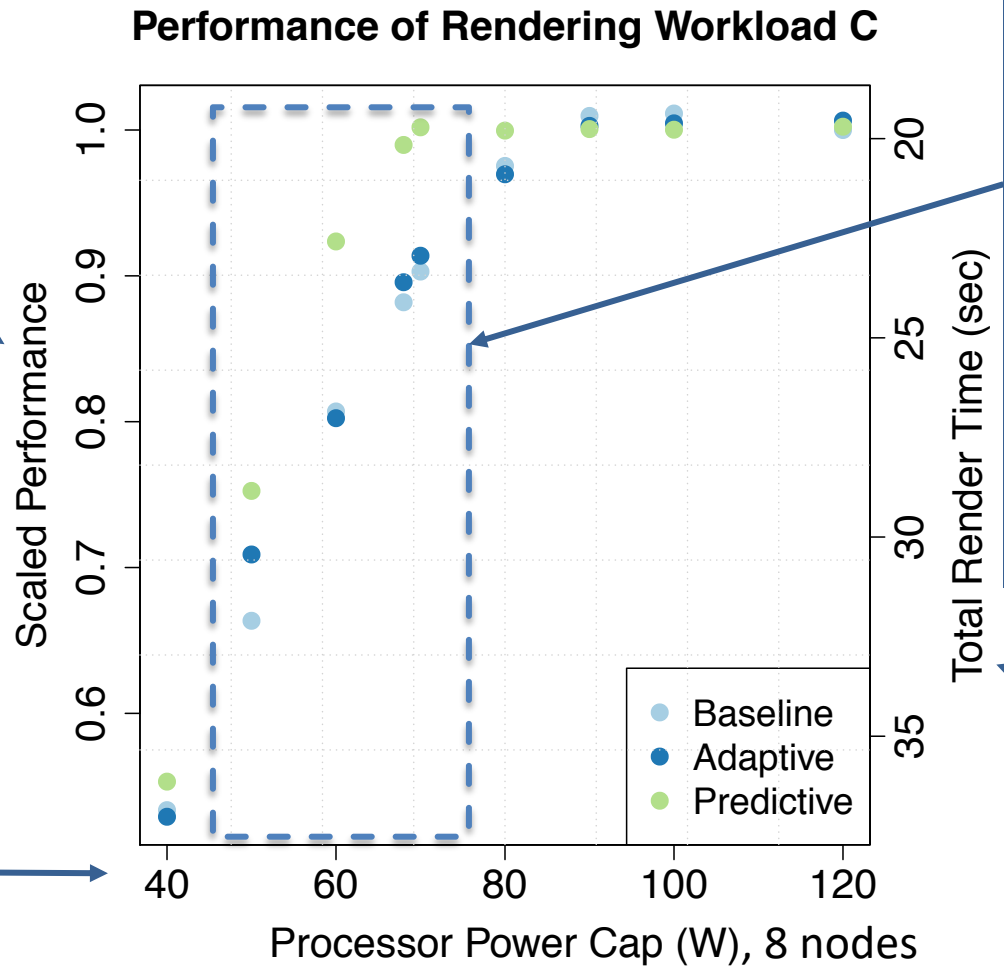
Adaptive – GEOPM

- Monitor execution time and adapt power during execution
- Behaviors from beginning iterations predict behavior of later iterations
- Better suited for regular and predictable workloads

Research Question: How will the predictive and adaptive approaches compare in improving performance for visualization (rendering) workloads?

Results:
Predictive strategy performs better than adaptive strategy

Scaled performance normalized to Baseline at 120W (TDP)



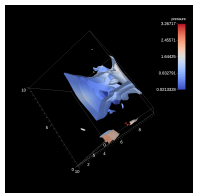
Predictive performs better than adaptive strategy under power constraints for this workload.

This is comparing to baseline strategy of uniform power allocations.

Increasing processor power limit

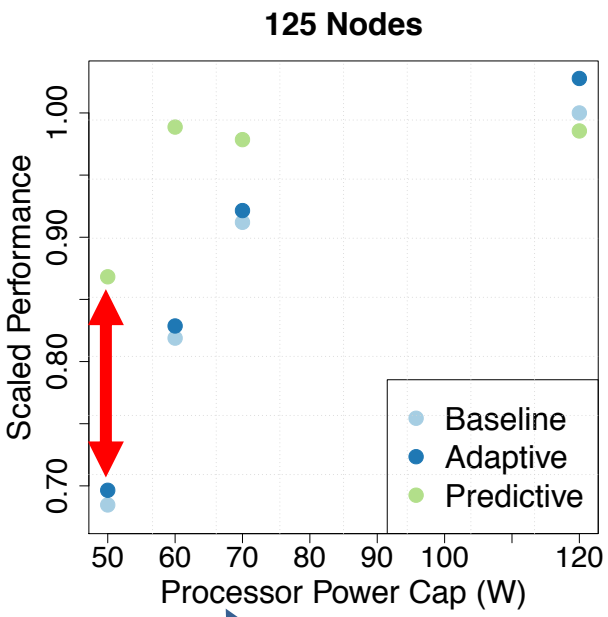
Total rendering time

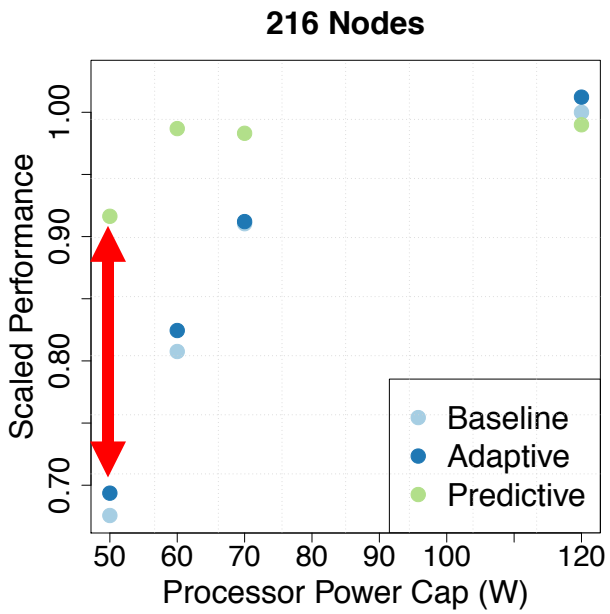
128^3 data set, 0.9 isovalue
1920^2 resolution, 170 images, 300 sim cycles, 6 vis cycles

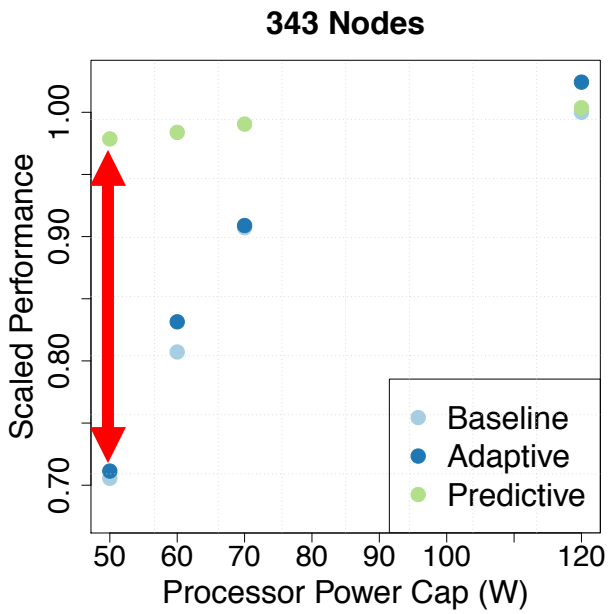


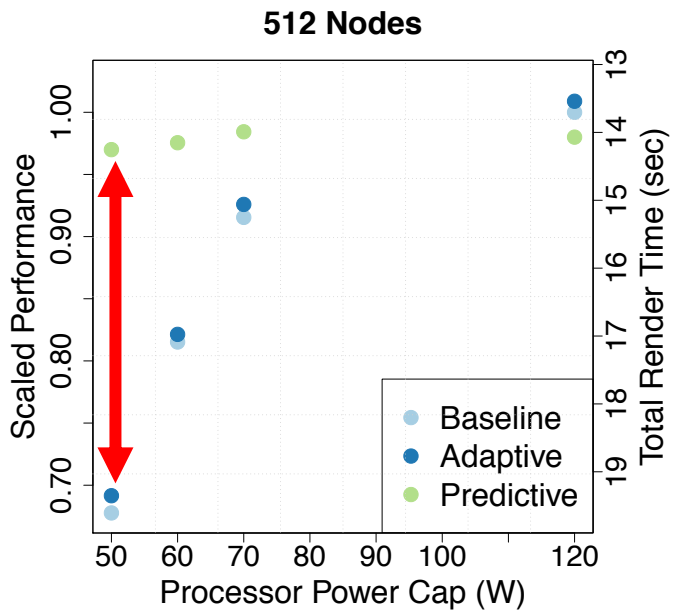
Results:
Increasing concurrency leads to additional performance gains

Increase Concurrency





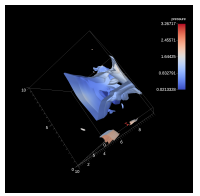




Focus power limits on {50W, 60W, 70W}

Increase Speedup

128^3 data set, 0.9 isovalue
1920^2 resolution, 170 images, 300 sim cycles, 6 vis cycles



Takeaways show better performance using prediction rather than adaptation for scheduling power

- Dynamically scheduling power within a job can be done by using prediction or using adaptation of execution behaviors

- Using prediction to re-allocate power can improve performance by up to 27% over using adaptation for imbalanced workloads, while using the same power overall

Dissertation Question: "Can Visualization Be Optimized on Power-Limited Supercomputers?"

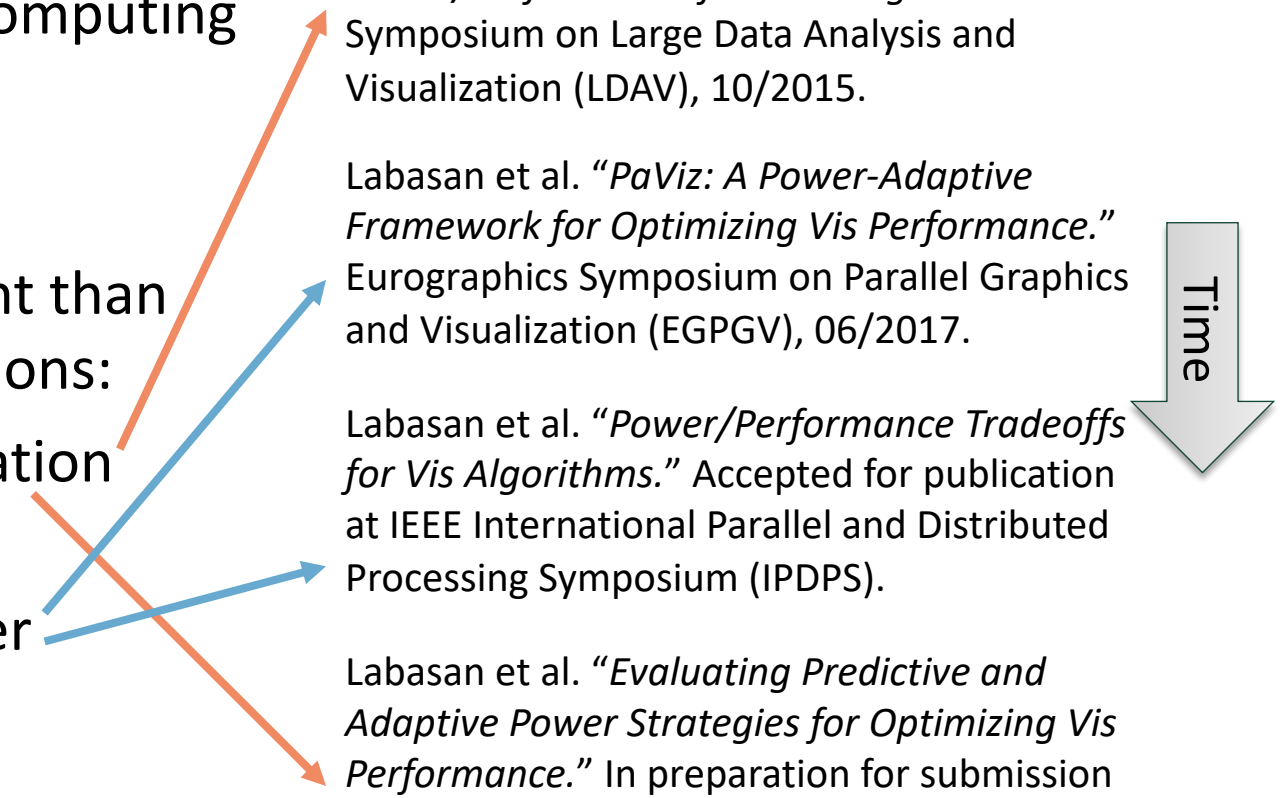
- Involves the intersection of two topics:
  - Power-constrained high performance computing
  - Scientific visualization

- Focus: Visualization workloads are different than simulation workloads...leads to two questions:

Characterize Data Intensity

Power Scheduling

1. How different are visualization workloads?
2. How does this affect power optimization strategies?



Labasan et al. "*Exploring Tradeoffs Between Power/Performance for a Vis Algorithm.*" IEEE Symposium on Large Data Analysis and Visualization (LDAV), 10/2015.

Labasan et al. "*PaViz: A Power-Adaptive Framework for Optimizing Vis Performance.*" Eurographics Symposium on Parallel Graphics and Visualization (EGPGV), 06/2017.

Labasan et al. "*Power/Performance Tradeoffs for Vis Algorithms.*" Accepted for publication at IEEE International Parallel and Distributed Processing Symposium (IPDPS).

Labasan et al. "*Evaluating Predictive and Adaptive Power Strategies for Optimizing Vis Performance.*" In preparation for submission at IEEE Transactions on Parallel and Distributed Systems (TPDS).

Dissertation Question: “Can Visualization Be Optimized on Power-Limited Supercomputers?”

- Dissertation Answer: Yes!

- Because:
  - We evaluated tradeoff opportunities between power and performance for different input parameters (e.g., data set size, algorithm implementation).
  - We exploit these tradeoffs in a performance model and show performance improvement by shifting power in a power-limited environment at large scale.

Future Research Directions

- Explore power and performance tradeoffs for other hardware architectures
  - Gain better understanding of execution behaviors on heterogeneous platforms
- Evaluate when to use prediction or adaptation in dynamically reallocating power to improvement performance for irregular workloads
  - Prediction has large overhead costs, but is better suited for unpredictable workloads
  - Adaptation is online, but may not converge on irregular workloads
- Create additional performance models for more visualization workloads
- Develop a power-aware central resource manager

Optimizing Visualization Performance on Power-Limited Supercomputers

The author wrote this dissertation in support of requirements for the degree Doctor of Philosophy in (field) at (university, location). The research is funded in part by the LLNL Graduate Scholars Program, and is not a deliverable for any United States government agency. The views and opinions expressed are those of the author, and do not state or reflect those of the United States government or Lawrence Livermore National Security, LLC.

Standard practice for overprovisioning

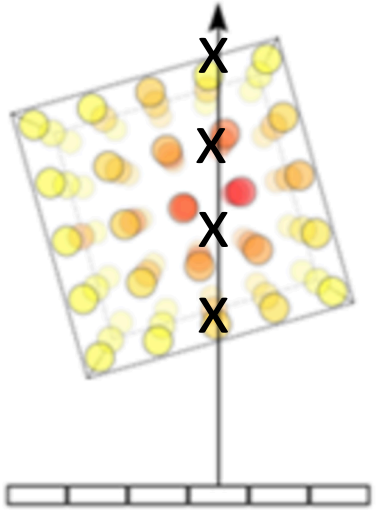
- Setting processor power limits requires user space access to Intel registers
  - This is a large request of large scale systems

- LLNL's solution: msr-safe (github.com/llnl/msr-safe)
  - Kernel driver providing user space access through a whitelisting mechanism
  - Enables overprovisioning research at scale on certain Intel machines
  - Required by Intel's GEOPM, included with OpenHPC release

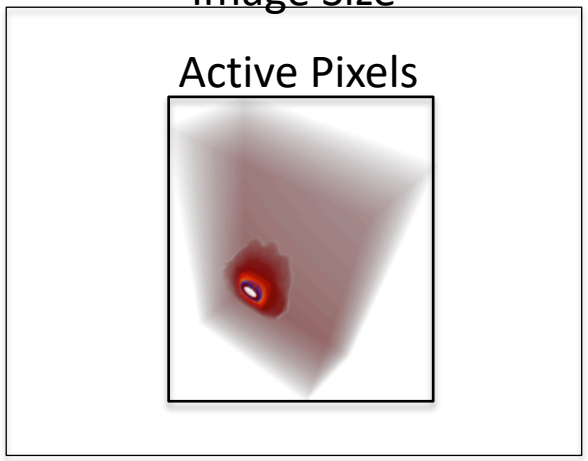
- This is in line with what the overprovisioning community is using

Volume Rendering Performance Model

- 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

- Predicted time to complete volume rendering per node:

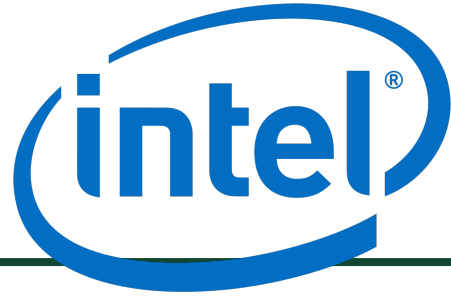
$$T_{VR} = c_0 * (AP * CS) + c_1 * (AP * SPR) + c_2$$

Cell Frequency

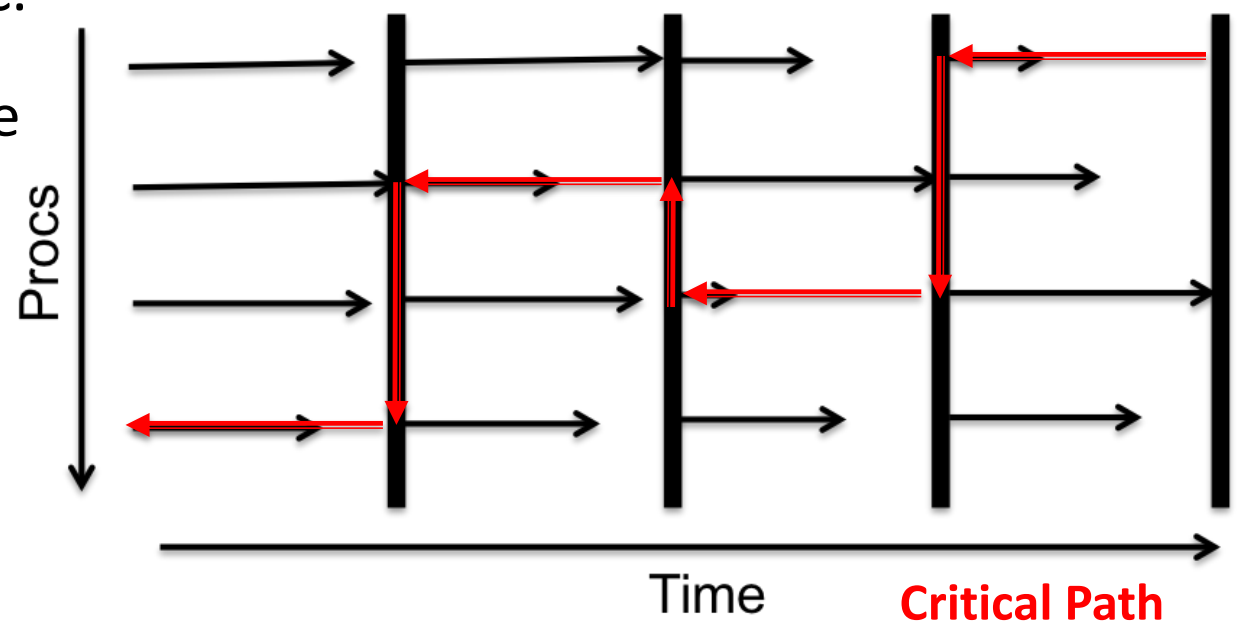
Sample Frequency

Larsen et al. “Performance Modeling of In Situ Rendering”, SC16.

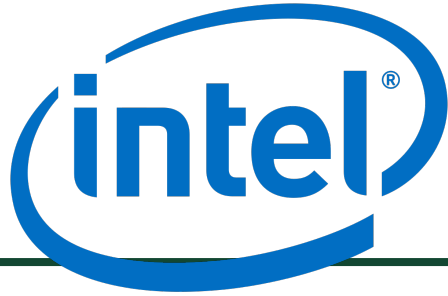
Load imbalances pose a performance challenge



- Massively distributed bulk-synchronous applications
- Global synchronization barriers at computation milestones
- Load imbalance manifestation: manufacturing variation, OS jitter, problem does not divide evenly, etc.
- Performance determined by last compute unit to arrive at the barrier
  - Speed up slower processors
  - Slow down faster processors



Improving performance with intra-node power management



- Improve application performance by leveraging fine-grained power management capabilities within a node
- Track thread progress within a loop, assign relative frequencies to each thread, check progress
- Idea: Should be able to request higher than expected frequency
- Prototyping work influenced next generation architectures

Labasan et al. Mitigating load imbalances through hierarchical performance balancing. US Patent Application No: US20170277576A1. Pending patent.