Our study is at the intersection of overprovisioning and visualization

- **Problem**: Power is a critical constraint for next generation supercomputers
- **Solution**: Hardware overprovisioning (...and also hardware improvements)
  — “More nodes,” but enforce power limits, degrading performance

- **This study**:
  - **Premise**: Visualization algorithms under power limits will differ from HPC applications
    - Visualization is data intensive
  - **What we did**: Consider a representative set of visualization algorithms and investigate tradeoffs between slowdown and power usage
  - **Outcome**: Understand execution characteristics under power limits
  - **Why this is important**: Exploit characteristics for better performance
Outline

- Motivation
  - Power/Overprovisioning
  - Why Visualization?
  - Overprovisioning and Visualization
- Experimental Overview
- Results
- Conclusion and Future Work

Current system power utilization is sub-optimal

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

### Total Power Consumption of BG/Q Vulcan Supercomputer
Feb 2013 to Feb 2019

- 2.32 MW, Linpack
- 1 MW Un-used!
- 1.47 MW, Other codes

### Total Power Consumption of Broadwell Quartz Supercomputer
Nov 2017 to Aug 2018

- 2.4 MW: Assumes all nodes consume peak power at the same time
- 0.83 MW, Other codes
- 40% Un-used!
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
  - Need to maximize limited power resources

- **Key Premise:** To meet power consumption goals at exascale, need to innovate power-efficient techniques not only in hardware, but also in software.

---------------------

Improve utilization of power resources with overprovisioning

**Traditional**

- All nodes can run at peak power simultaneously

**Overprovisioning**

- Increase compute capacity by adding more nodes
  - Not all nodes can run at max power simultaneously
- Limit power usage per node
  - Example: 2X nodes, each running at 1/2 power → benefits common use case

For overprovisioning to be effective, all aspects of the HPC ecosystem (including visualization!) must be reconsidered with power in mind.
Limit power usage with hard power limits

- **Outcome**: Takes longer to run, but uses less power
- **Why**: Non-linear relationship between power and CPU frequency
  - Lower CPU frequency results in less power usage
  - But, subcomponents still consume power at the same rate
- Application behavior under power limits varies

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

- Central manager coordinating power allocations across system
  - Given a power constraint, adjust application accordingly

- We are moving towards a workflow where the visualization and analysis are concurrently running with the simulation (i.e., in situ)

- **This study**: Visualization component under a power limit
Motivation

— Power/Overprovisioning
— Why Visualization?
— Overprovisioning and Visualization

Experimental Overview

Results

Conclusion and Future Work

Why visualization?

#1: Visualization is important for scientific insight

#2: Visualization can use significant resources on the HPC system (10-20% of the total runtime)

#3: With overprovisioning, visualization workloads respond differently than simulation workloads
   — (discussed in overprovisioning/visualization section)
#1: Visualization is important for understanding scientific simulation data

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

#2: Visualization will share resources with the simulation

- Traditional approach is *post hoc* processing
  - **Not feasible at exascale:** Compute – and thus ability to generate data – is increasing faster than I/O

- Data analysis and visualization occur *while* simulation is running

- Changes to HPC ecosystem pose challenges for visualization
  - Total execution time may increase since compute resources are shared
  - Optimizing visualization will benefit overall performance
Outline

- Motivation
  - Power/Overprovisioning
  - Why Visualization?
  - Overprovisioning and Visualization

- Experimental Overview
- Results
- Conclusion and Future Work

Overprovisioning and visualization merits special attention

- **Why?** Visualization is data intensive
  - Need visualization-centric strategies to thrive in this constrained environment
Hundreds of visualization algorithms, not practical to optimize each individually

- Isovalue(s), Number of contours
- Min/Max
- Radius, Center
- Camera position, Image resolution, Number of samples
- Contour
- Threshold
- Spherical Clip
- Ray Tracing
- Execution behaviors differ across algorithms
- Unique configurations for each algorithm
- Position, Normal
- Number of seeded particles, Step length, Number of steps
- Camera position, Image resolution, Transfer function, Number of samples
- Min/Max
- Pipeline algorithms

Our study is foundational for understanding the execution behaviors for a representative set of all algorithms under power limits.

Overprovisioning and visualization merits special attention

- **Why?** Visualization is data intensive
  → Need visualization-centric strategies to thrive in this constrained environment

Research Questions

- How do visualization workloads perform under power limits?
- If you must run at a lower power limit, how much slowdown will you see?
Outline

- Motivation
  - Power/Overprovisioning
  - Why Visualization?
  - Overprovisioning and Visualization

- Experimental Overview
- Results
- Conclusion and Future Work

Study Factors

- Processor-level power limit
- Visualization algorithm
- Data set size
- Single node of LLNL RZTopaz system (Intel Broadwell)
  - 36 cores per node running at 2.1 GHz
- Leverage `msr-safe` kernel driver to set power limit from user space
- Enforce power limits with Intel’s RAPL
  - Acceptable limits between 120W down to 40W

https://github.com/llnl/msr-safe
Study Factors

- Processor-level power limit
- Visualization algorithm
- Data set size

- Contour
- Threshold
- Spherical Clip
- Ray Tracing

- Slice
- Particle Advection
- Volume Render
- Isovolume

Study Factors

- Processor-level power limit
- CloverLeaf3D: Hydrodynamics proxy application
- Data set sizes per node
  - $32^3, 64^3, 128^3, 256^3$
- Number of cells per node range from 32K to 16M

- Data set size
Methodology

- Study conducted in 3 phases

- Phase 1: Vary power limit for a single visualization algorithm and data set size

- Following phases varies 1 factor at a time to investigate effects
  - Phase 2: Vary visualization algorithm
  - Phase 3: Vary data set size

Outline

- Motivation
  - Power/Overprovisioning
  - Why Visualization?
  - Overprovisioning and Visualization

- Experimental Overview

- Results

- Conclusion and Future Work
Phase 1 Results: Vary Processor-Level Power Limit

- Reducing power limit by 3X:
  - Runtime increases by 1.2X
  - Frequency decreases by 1.2X

- Contour algorithm does not see a slow down proportional to power limit

- Favorable power savings

<table>
<thead>
<tr>
<th>Contour</th>
<th>(P)</th>
<th>(P_{ratio})</th>
<th>(T)</th>
<th>(T_{ratio})</th>
<th>(F)</th>
<th>(F_{ratio})</th>
</tr>
</thead>
<tbody>
<tr>
<td>120W</td>
<td>1.0X</td>
<td>33.474s</td>
<td>1.00X</td>
<td>2.55GHz</td>
<td>1.00X</td>
<td></td>
</tr>
<tr>
<td>110W</td>
<td>1.1X</td>
<td>33.543s</td>
<td>1.00X</td>
<td>2.41GHz</td>
<td>1.06X</td>
<td></td>
</tr>
<tr>
<td>100W</td>
<td>1.2X</td>
<td>33.579s</td>
<td>1.00X</td>
<td>2.55GHz</td>
<td>1.00X</td>
<td></td>
</tr>
<tr>
<td>90W</td>
<td>1.3X</td>
<td>33.519s</td>
<td>1.00X</td>
<td>2.55GHz</td>
<td>1.00X</td>
<td></td>
</tr>
<tr>
<td>80W</td>
<td>1.5X</td>
<td>33.617s</td>
<td>1.00X</td>
<td>2.54GHz</td>
<td>1.01X</td>
<td></td>
</tr>
<tr>
<td>70W</td>
<td>1.7X</td>
<td>30.371s</td>
<td>0.91X</td>
<td>2.54GHz</td>
<td>1.00X</td>
<td></td>
</tr>
<tr>
<td>60W</td>
<td>2.0X</td>
<td>30.394s</td>
<td>0.91X</td>
<td>2.50GHz</td>
<td>1.02X</td>
<td></td>
</tr>
<tr>
<td>50W</td>
<td>2.4X</td>
<td>31.066s</td>
<td>0.93X</td>
<td>2.52GHz</td>
<td>1.01X</td>
<td></td>
</tr>
<tr>
<td>40W</td>
<td>3.0X</td>
<td>39.198s</td>
<td>1.17X</td>
<td>2.07GHz</td>
<td>1.23X</td>
<td></td>
</tr>
</tbody>
</table>

Changes for Phase 2

- Changes: Execute all visualization algorithms under varying power limits

- Goal: Investigate the data intensity of additional visualization algorithms
### Phase 2 Results: Vary Visualization Algorithm

<table>
<thead>
<tr>
<th><strong>Power Opportunity</strong></th>
<th></th>
<th><strong>Power Sensitive</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>✓ Performance remains constant until lowest power cap</td>
<td></td>
<td>✓ Performance degrades relative to power cap</td>
</tr>
<tr>
<td>✓ Compute-bound workloads</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Contour**
- **Threshold**
- **Spherical Clip**
- **Slice**
- **Isovolume**
- **Ray Tracing***

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

### (cont’d) Phase 2 Results: Comparing power opportunity and power sensitive algorithms

<table>
<thead>
<tr>
<th><strong>Power Opportunity: Threshold</strong></th>
<th></th>
<th><strong>Power Sensitive: Particle Advection</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>✓ Iterate over the cells in data set</td>
<td></td>
<td>✓ Place a massless particle at a seed location</td>
</tr>
<tr>
<td>✓ Remove cells that do not have values in the specified range</td>
<td></td>
<td>✓ Displace the particle according to the vector field</td>
</tr>
<tr>
<td>✓ Simple computation dominated by loads and stores of each cell</td>
<td></td>
<td>✓ Result is an “integral curve” corresponding to the trajectory the particle travels – Euler, RK4, ...</td>
</tr>
</tbody>
</table>

- **Threshold**
- **Particle Advection**
(cont’d) Phase 2 Results: Power opportunity algorithms are data intensive, while power sensitive algorithms are compute intensive.
Changes for Phase 3

- Changes: Execute all visualization algorithms with different data set sizes
- Goal: Investigate the impacts of data set size on power and performance tradeoffs

Phase 3 Results: Vary Data Set Size

![Graphs showing variations in Processor Power Cap (W), Instructions Per Cycle (IPC), and data set size.](image)
Motivation
- Power/Overprovisioning
- Why Visualization?
- Overprovisioning and Visualization

Experimental Overview

Results

Conclusion and Future Work

Takeaways inform how visualization algorithms can adapt to imposed power limits

- VTK-m contour implementation is sufficiently data intensive to avoid a significant slowdown when the power limit is reduced
- Most algorithms consume low amounts of power, providing opportunities for energy/power savings (*e.g.*, data intensive)
- Tradeoffs become more favorable with a smaller data set size per node

**Future work**: Integrate these learnings into a power-aware runtime system, such that more informed decisions can be made about how to allocate power in an in situ workflow
Thank you!

- VTK-m contour implementation is sufficiently data intensive to avoid a significant slowdown when the power limit is reduced.
- Most algorithms consume low amounts of power, providing opportunities for energy/power savings (e.g., data intensive).
- Tradeoffs become more favorable with a smaller data set size per node.

**Future work:** Integrate these learnings into a power-aware runtime system, such that more informed decisions can be made about how to allocate power in an in situ workflow.

“Power and Performance Tradeoffs for Visualization Algorithms”
Stephanie (Labasan) Brink (brink2@llnl.gov)