A Systemic Approach to Maximize Heterogeneous System Performance

Thomas Randall

Outline

- **Overview**
- Components of the Proposal
  - Immersion Cooling Study
  - Optimizing for Generations of Hardware
  - Efficient and Transferable Multi-Scale Tuning
- Timeline

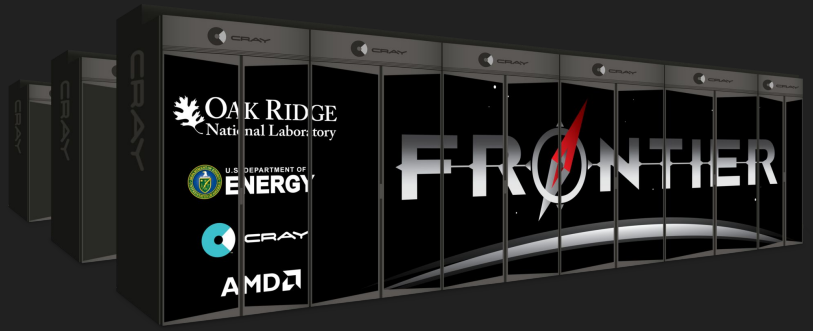
What is a “Systemic Approach to Performance”?

- Different hardware systems
  - CPU, GPU, custom accelerators

- Different performance needs
  - Latency, energy efficiency, throughput

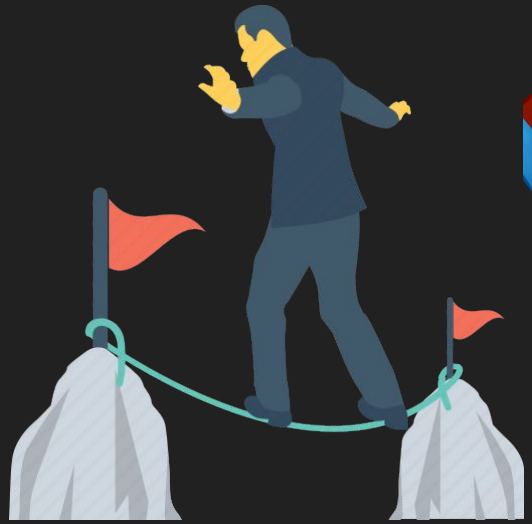
Who needs this?

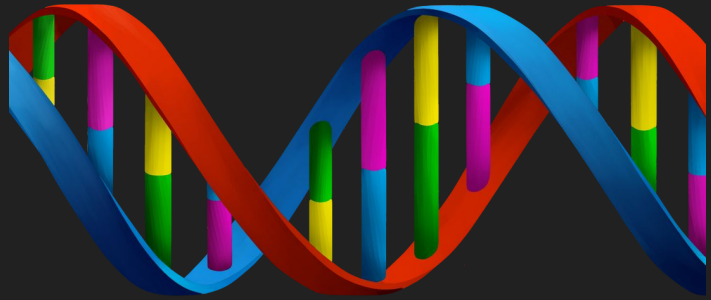
- Focus on High Performance Computing (HPC)
  - Cloud, Industry, Government datacenters and supercomputers



Why do we need it?

- Unilateral approaches: missing specificity
- Rely on end-users: limited resource
- Ever-changing demands: continuous evolution





Primary Goals and Challenges

- Cooperation between hardware and software
  - Not always intuitive
  - Trade offs
- Forward-looking flexibility
  - Allow future improvement
  - Shifts in state-of-the-art and practice
- Minimal cost for maximum benefit
  - Near-infinite maximum cost
  - Premature optimization is the root of all evil – Donald Knuth

Core Components

- Hardware
  - Dennard Scaling, Moore’s Law, Amdahl’s Law
  - New explosion of architectures
  - **Key focus: Energy efficiency for sustained performance**
- Software
  - Capable software stacks
  - Sea of frameworks
  - **Key focus: Algorithm design for hardware acceleration**
- Tuning
  - The glue to hold everything together… longer
  - **Key focus: Improve efficiency of re-usable information**

Outline

- Overview
- **Components of the Proposal**
  - Immersion Cooling Study
  - Optimizing for Generations of Hardware
  - Efficient and Transferable Multi-Scale Tuning
- Timeline

Proposed Works

- Immersion Cooling Study
  - Liquid immersion cooling technology
  - Designed for energy efficiency
  - Opportunity to sustain software performance
- Optimizing for Generations of Hardware
  - Word2Vec algorithm and performance demands
  - Effective acceleration for many kinds of GPUs at once
  - Opportunity for hardware advancements to benefit software
- Efficient and Transferable Multi-Scale Tuning
  - Automatic performance tuning for any hardware-software combination
  - Novel transfer learning technique
  - Opportunity to reduce continuous tuning costs and expand scope

Specific Contributions

- Immersion Cooling Study
  - Quantifying “greater than air” advantage
  - First empirical study of energy and performance impacts
- Optimizing for Generations of Hardware
  - Reduce cost of optimization through architecture-leaning approach
  - Design for lasting performance improvement
  - Limit depth of specialization required
- Efficient and Transferable Multi-Scale Tuning
  - Reducing barrier to entry
  - Increase scope of benefits gained
  - Overcome shortcomings of existing techniques

Significance and Impact

- Immersion Cooling Study
  - Environmental benefits from energy efficiency
  - Sustain higher performance through thermal management
- Optimizing for Generations of Hardware
  - Reduce memory traffic by 89%
  - 1.4-4.3x speedup over multiple hardware generations
- Efficient and Transferable Multi-Scale Tuning
  - Permit greater opportunity for performance tuning
  - Up to 12.81x additional speedup in short-term evaluations

Outline

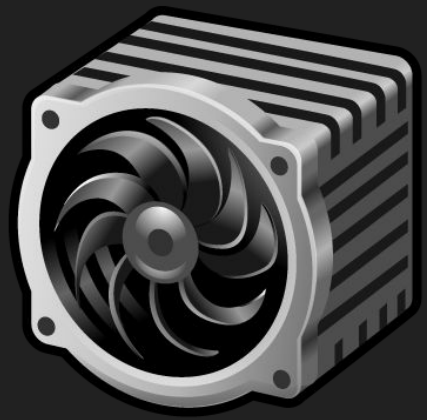
- Overview
- Components of the Proposal
  - **Immersion Cooling Study**
  - Optimizing for Generations of Hardware
  - Efficient and Transferable Multi-Scale Tuning
- Timeline

Cooling down computer systems

- Electric energy lost as heat
- Denser compute → greater heat
  - Least efficient delivery → demanding components
- Too much heat → performance loss
  - Lower clock rate
  - System down time
- Cooling components relocate heat
  - Recover faster
  - Sustain performance

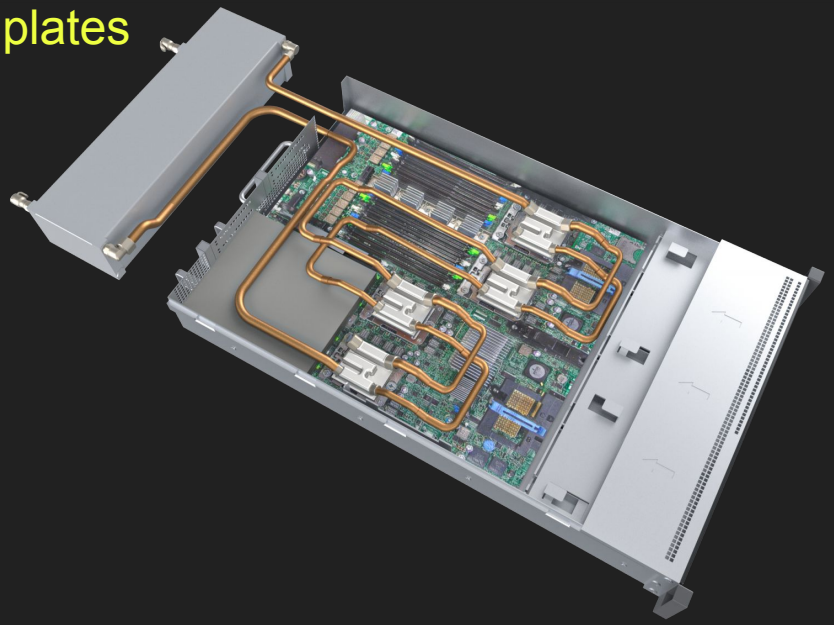
Issues of energy and efficiency

- LBNL: ~2% of US electricity consumption in data centers
  - International Energy Agency: Europe is reported around ~1.5%
- McKinsey & Company: 40% of data center energy use for cooling
  - Year-over-year demand increases by 10%
- Air-based cooling cannot maintain pace
  - Current GPUs often throttle
  - Future chips with kW-scale TDP
- Water conducts 30x more heat per unit volume



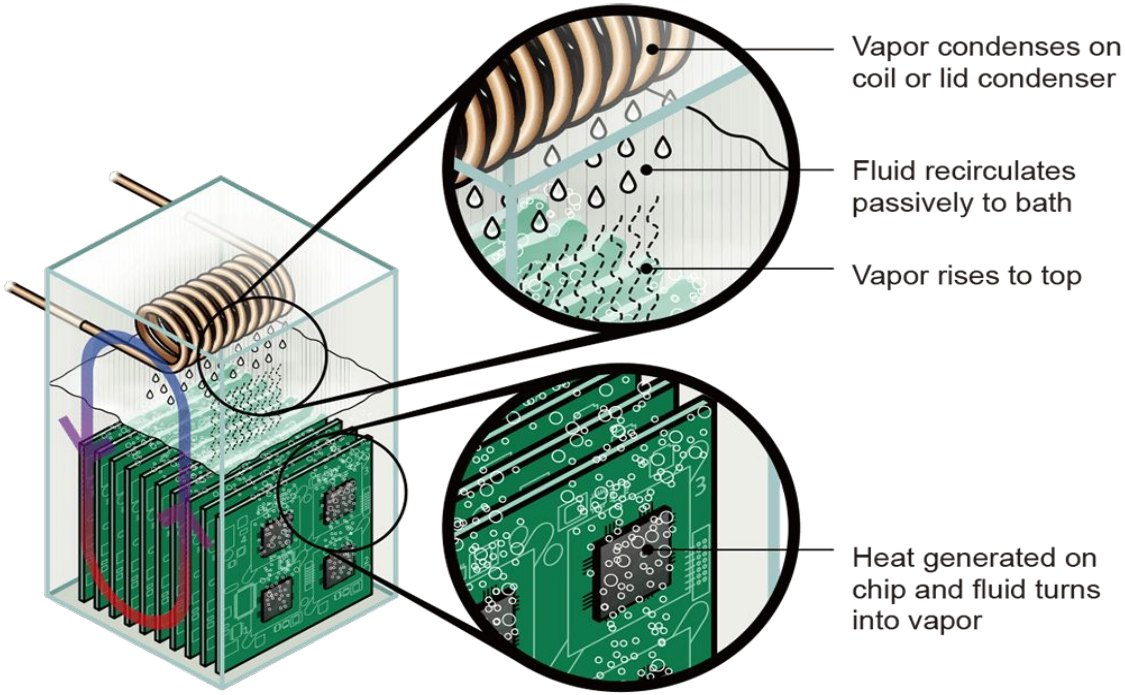
State of the Art: Direct Liquid Cooling

- Circulate chilled facility water through cold plates
  - Microchannels for fluid circulation
  - Other fluids can be used
- Simple in-rack setup
  - Efficiency improved by thermal conditions
- Fluid completely contained in system
  - Safety for computing components
  - Some reduction in effectiveness
  - Minimal complication for maintenance



State of the Art: Dual Phase Immersion Cooling

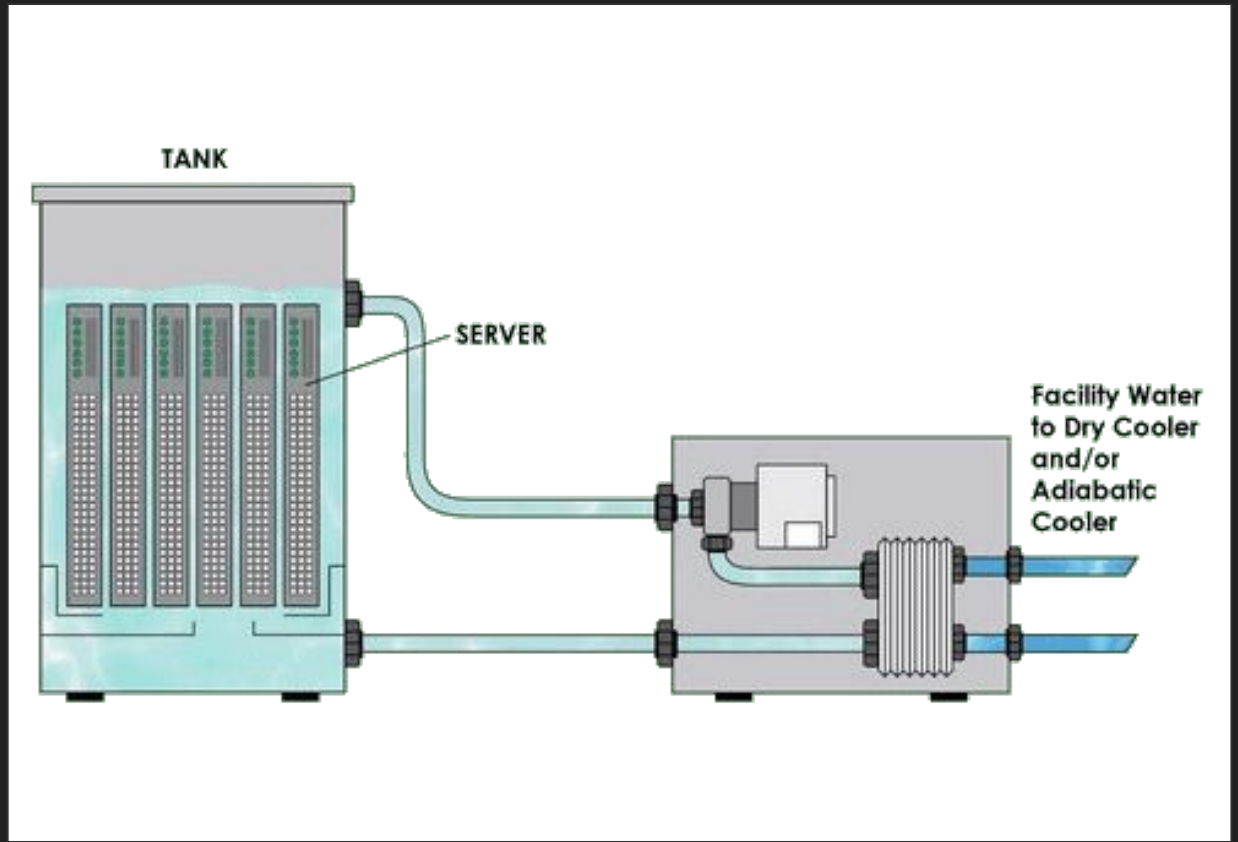
- Thermodynamic heat transfer
- No moving parts
- Gas complicates maintenance
- Fluids are not environmentally friendly
- Nonzero risk of combustion!



The Passive 2-Phase Immersion Cooling Cycle

State of the Art: Single Phase Immersion Cooling

- No evaporation
- Circulate fluid, similar to DLC
- Maintenance is simpler
- Liquid-liquid heat exchange is very efficient
- Large industry focus



Previous studies

- Foundational effectiveness
- Design and properties of dielectric fluid
- Number of pumps and their arrangement
- Effects of long-term immersion on computing hardware
- Capital + Maintenance costs

Proposed empirical study

- Non-generalizable, but starting point is necessary
  - Especially relative to air-based cooling
- System under study: Submer SmartPod v3
  - White mineral oil dielectric fluid
  - Heat exchange with 11C facility water
  - Comparable hardware in CRAC and fan environment for comparison
- Thermal characteristics to observe
  - Challenge: Represent broad variety of HPC workload conditions
  - Insight: Heat exchange within the system
  - Challenge: Safely maintain performance at limits

Applications of Interest

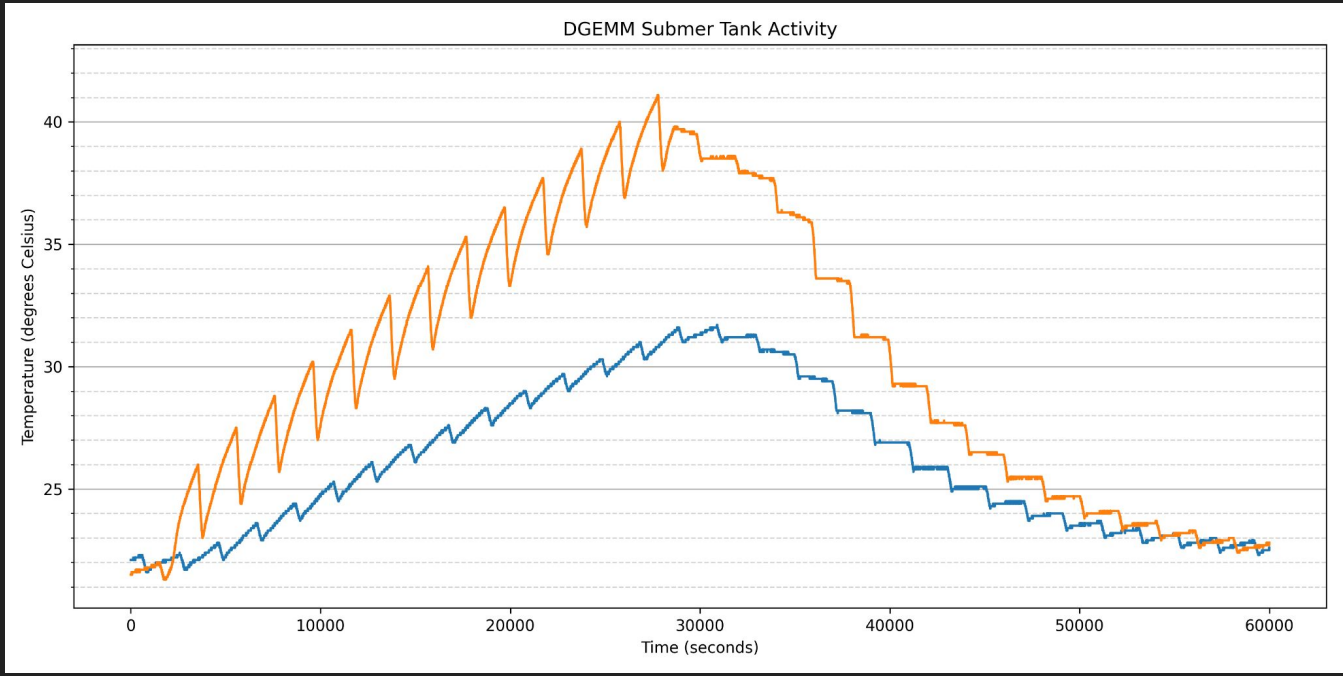
- GPU Application Selection
  - Memory-Intensive
    - Stream
    - EMOGI
  - Compute-Intensive
    - DGEMM
    - MD5 Cracking
  - Machine learning
    - MLPerf
- CPU Application Selection
  - NPB
  - HPLinpack

Remaining Work

- GPU Application Selection
  - Memory-Intensive
    - ~~Stream~~
    - ~~EMOGI~~
  - Compute-Intensive
    - ~~DGEMM~~
    - ~~MD5 Cracking~~
  - Machine learning
    - MLPerf
- CPU Application Selection
  - NPB
  - HPLinpack
- Air-cooled replication

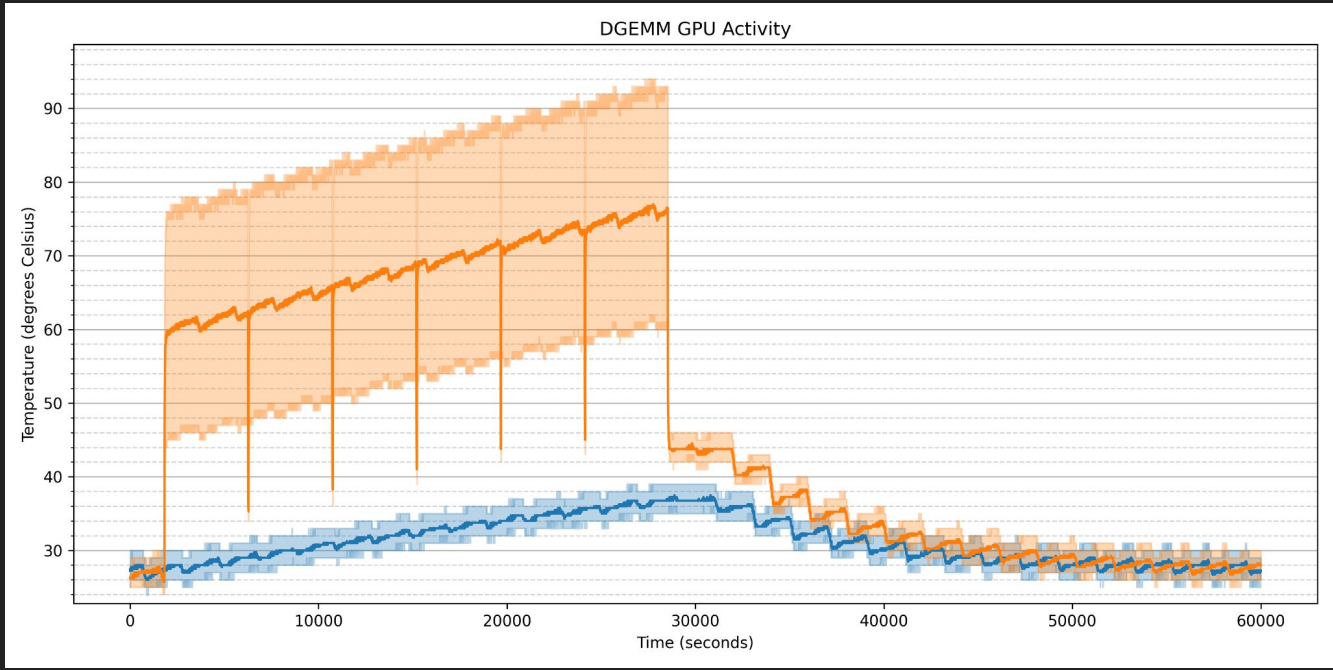
Preliminary Results: DGEMM

- Orange: with application running
- Blue: no application running
- ½ hour warmup
- 8 hours runtime
- 8 hours return to initial conditions
- Resting 22C



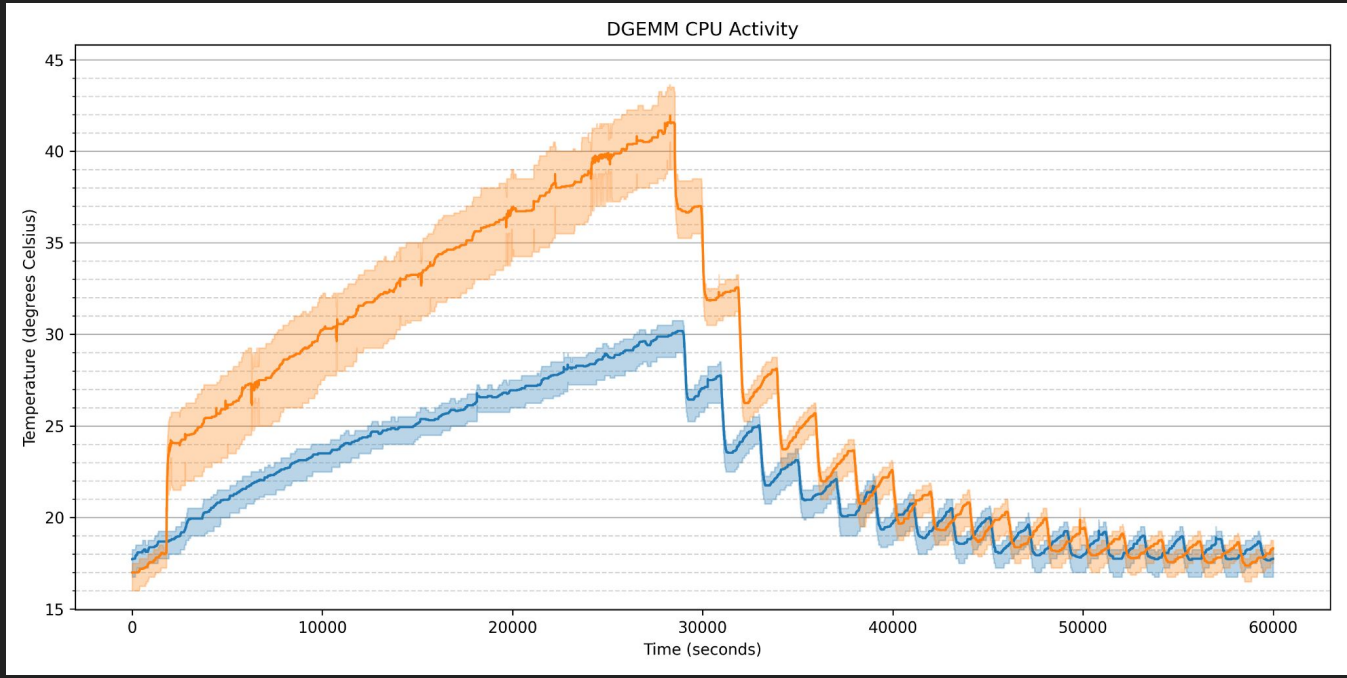
Preliminary Results: DGEMM

- Titan V GPU
  - 12 GB HBM2 @651 GB/s
  - 5120 Cores @14.9 TFLOP/s
- Thermal limits
  - Target: 85C
  - Supported: 89C
  - Max: 91-95C
  - Throttle: 97C
  - Shutdown 100C
- Solid line mean
- Shaded min/max
  - Over supported



Preliminary Results: DGEMM

- AMD EPYC 7351P CPU
  - 16 cores (32 logical)
  - 2.4 GHz
- High CPU activity despite being a GPU application
- Similar time to return to baseline (sub-22C)



Outline

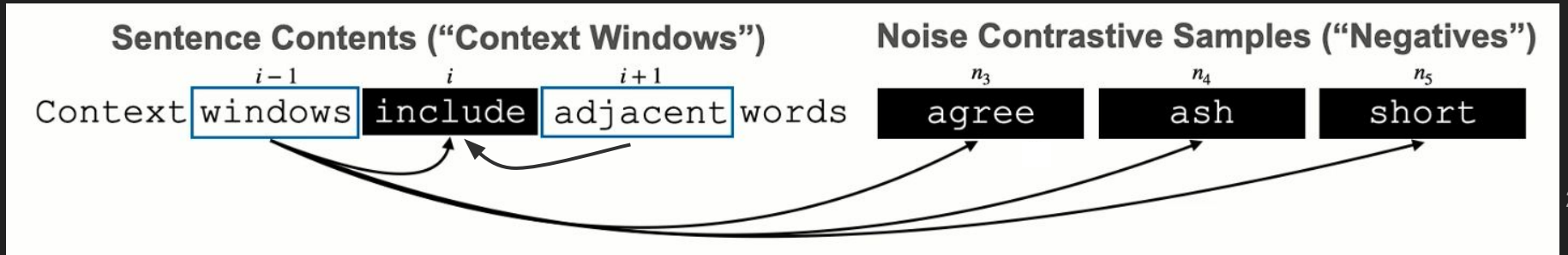
- Overview
- Components of the Proposal
  - Immersion Cooling Study
  - **Optimizing for Generations of Hardware**
  - Efficient and Transferable Multi-Scale Tuning
- Timeline

Introducing Word2Vec

- Natural Language Processing (NLP) machine-learning algorithm
  - Training data: human-written text
  - **Unsupervised** objective: predict word co-occurrence
  - Model output: dense semantic vectors for words
- Downstream uses in NLP
  - Sentiment analysis
  - Machine translation
  - Spam detection
  - Grammar correction
  - Summarization

Bird’s eye view of algorithm

- Text is decomposed into “sentences”
  - Embarrassingly parallel
- Sentences are composed of context windows
  - Words close enough for association
  - Serial order necessary for convergence and correctness
- Noise-contrastive samples (negatives) for each context window
  - Positively correlate words within the window
  - Negatively correlate spurious selection of words



Case for optimization

- Repeated usage
  - Larger dataset → richer model
  - Private and domain-specific datasets mean pretraining not always enough
  - Language evolution necessitates retraining
- Ripe for GPU acceleration
  - Portions are embarrassingly parallel
  - Heavily leans on matrix processing – implicitly PCA
  - Minibatches with simple network architecture

Suboptimal GPU performance

- Initial implementations for K40 GPUs failed to scale to successive hardware generations
- CPUs maintained general performance advantage

Hardware Platform	pWord2Vec (CPU) Millions Words/Sec	Wombat (GPU) Millions Words/Sec
Broadwell CPU, P100 GPU	**10.36**	2.86
Haswell CPU, TitanXP GPU	8.4	3.3
Skylake CPU, V100 GPU	9.32	10.33

Core problems for GPUs

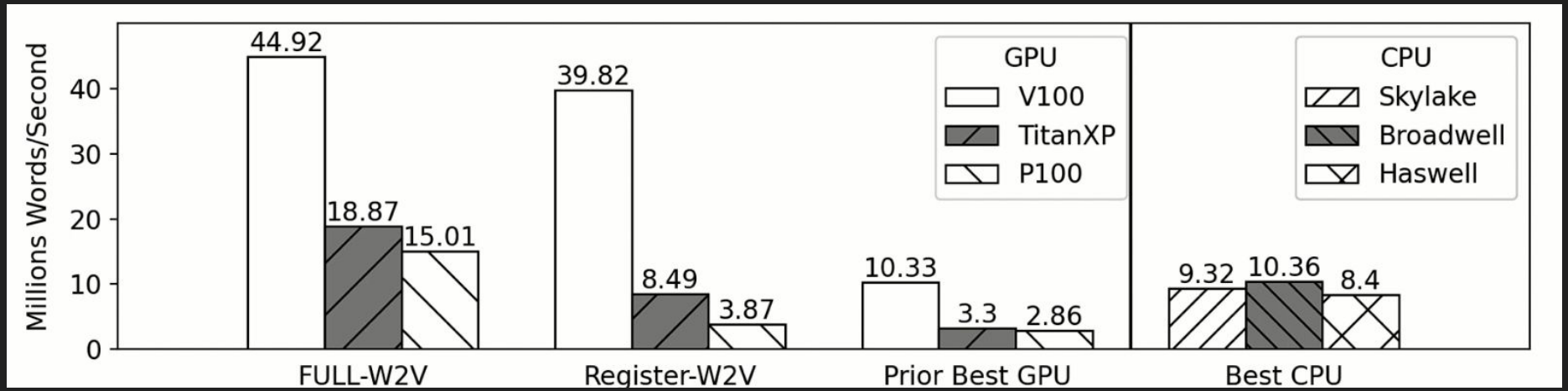
- High cost of memory latency
  - Negatives are cache-averse and explicitly harm locality
  - Repetition of context windows not exploited by matrix tiling
- Low computational intensity
  - Simple network = minimal compute
  - Small matrices limit gains from tiling
- Weak scaling does little to aid performance
  - More scheduling work
  - Not as much computational demand
  - Much more memory demand

Core opportunities for improvement

- Vector processing reduces data movement
  - Compute everything in register with minimal memory overhead
  - Fixed context width allows for limited negative samples
  - Fuse network operations
- Known data reuse improves cache utilization
  - Explicitly managed L1 cache – GPU shared memory
  - Perfect cache eviction
  - Maximum cache hit rate
- Together: 89% reduction in memory traffic!
  - Maintain computational intensity
  - Maintain semantic correctness

Experiment with 1 Billion Words dataset

- Register-W2V near CPU throughput on TitanXP instead of V100
  - Vs prior GPU: 1.35X (P100), 2.57X (TitanXP), 3.85X (V100)
- FULL-W2V exceeds CPU throughput across all hardware generations
  - Vs prior GPU: 5.2X (P100), 5.7X (TitanXP), 4.3X (V100)



Conclusions

- Better software alignment to hardware became necessary
  - Deep understanding of just hardware or just software insufficient
- Possible to benefit multiple generations of performance
  - Register file and shared memory sizes increased
  - Memory latency bottleneck more important

Outline

- Overview
- Components of the Proposal
  - Immersion Cooling Study
  - Optimizing for Generations of Hardware
  - **Efficient and Transferable Multi-Scale Tuning**
- Timeline

Putting it all together

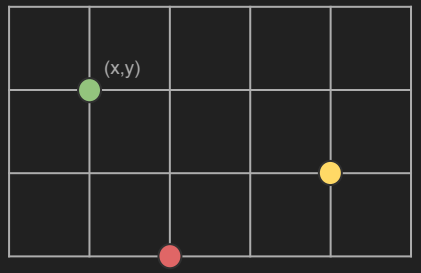
- The best performance usually involves navigating trade-offs
  - Configurable hardware and software settings
    - Source code adjustment
    - Compiler flags
    - Runtime options
    - Environment settings
- Produce the highest performing configuration via tuning
  - Speedup often by an order of magnitude
- Long tail of benefits
  - Repeated executions with better performance
  - Energy savings
  - Reusable libraries and modules

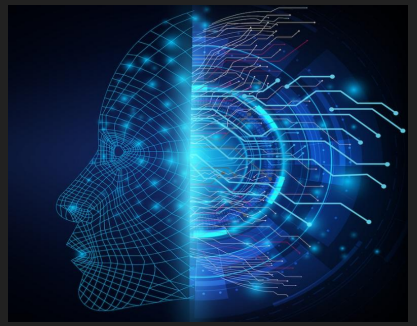
Cost of tuning

- Not everything gets tuned
  - HPC programs execute for hours at a time, tune via microbenchmarks
  - Lack of resources
  - Lack of time
- Simple kernel takes 25 seconds to evaluate performance
  - Ten parameters to tune in source code
    - Three loops are tiled with distinct sizes between 4 and 2048
    - One pair of loops have an optional interchange
    - Six arrays may be independently packed during loop traversals
  - Brute force: 376,320 combinations, 100+ days of serial compute
- Meaningful tuning *requires* sophisticated technique

Existing tuning approaches

- Grid search: simple
  - Coarse granularity - fast but inaccurate
  - Refinement - prone to local minima
- Performance modeling: generalize
  - Limited applicability
  - Immense initial cost
  - Generalization not guaranteed
- Machine learning: specialize
  - Large data demand
  - Cumbersome to handle many tasks





Exploratory surrogate models

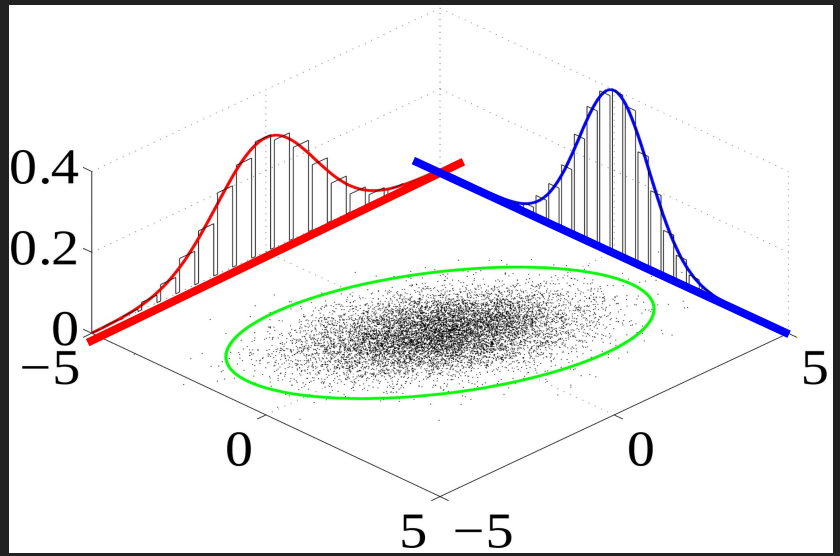
- BLISS, YTOPT, GPTune
- Surrogate represents known information and trend
  - Often includes uncertainty measure
  - Iteratively explore and exploit knowledge
- Powerful convergence and efficiency, but restart from scratch
  - Need capability to leverage known data from a related context

Opportunity for reuse

- In HPC, applications often tuned for multiple scales
  - Additional data, same system resource
  - Additional data AND additional system resource
- Transfer techniques improve efficiency of related tuning
  - Faster search = less resource expenditure
  - Still converge to global optimum
- Existing transfer techniques
  - Static tuning spaces
  - Require transfer calibration
  - Result: under-utilized except in highest-impact areas
    - BLAS
    - ML libraries

Proposed transfer model: Gaussian Copula

- Statistical model: multivariate probability distribution
  - Disjoint marginal representation per variable
  - Correlation as joint distribution
- Near-optimal short-term transferred search process
  - Distributions do not require calibration
  - Probability model forecasts budget
  - First generative transfer model for search



Distributions as search

- Challenges
  - Identify useful distribution
  - Adjust distribution intelligently
  - Guarantee utility of randomness
- Insights
  - Existing high-performing data forms distribution
    - Implicitly excludes sub-optimal regions
  - Limited complexity of distribution shifts
    - Especially when not crossing new bottlenecks
  - Hypergeometric sampling
    - Describe expected behavior under distribution

Preliminary Results: Few-shot Polybench tuning

App.	Scale	Peak Speedup (# Evaluation Discovered)				
		GC			BO	GPTune
		1^{st}	Budget	Best	Best	Best
3mm	SM	5.09	5.70 (23)	5.70 (23)	3.03 (26)	5.53 (30)
	ML	5.25	5.57 (29)	5.57 (29)	3.29 (30)	5.16 (16)
	XL	27.10	33.39 (18)	33.39 (18)	20.58 (30)	18.96 (25)

- One application three input data scales (SM, ML, XL)
- Speedup on evaluation number (parenthesized)
  - Limited to 30 evaluations
- Three results for GC technique with transfer
  - First evaluation
  - Within predicted budget
  - Best within 30 evaluations
- Bayesian Optimization: SOTA surrogate tuning from scratch
- GPTune: SOTA transfer learning ML model

Preliminary Results: Few-shot Polybench tuning

- Best results consistently on-budget
  - First result exceeds SOTA 44% tasks
  - Highest speedup 78% tasks
- GC up to 12.81X over SOTA
- SOTA no more than 0.24X over GC

App.	Scale	Peak Speedup (# Evaluation Discovered)				
		GC			BO	GPTune
		1^{st}	Budget	Best	Best	Best
3mm	SM	5.09	5.70 (23)	5.70 (23)	3.03 (26)	5.53 (30)
	ML	5.25	5.57 (29)	5.57 (29)	3.29 (30)	5.16 (16)
	XL	27.10	33.39 (18)	33.39 (18)	20.58 (30)	18.96 (25)
Cov.	SM	21.10	21.98 (21)	21.98 (21)	21.83 (28)	13.30 (30)
	ML	4.13	4.27 (26)	4.27 (26)	3.87 (25)	4.07 (30)
	XL	23.04	23.96 (2)	23.96 (2)	8.43 (12)	17.88 (9)
Floyd-W.	SM	1.01	1.02 (17)	1.02 (17)	1.02 (20)	1.01 (26)
	ML	1.02	1.02 (1)	1.02 (1)	1.01 (25)	1.01 (3)
	XL	0.99	1.00 (29)	1.00 (29)	1.01 (16)	1.01 (20)
Heat3d	SM	1.83	2.03 (5)	2.06 (18)	2.21 (15)	2.30 (28)
	ML	1.89	1.89 (1)	2.06 (10)	2.12 (25)	1.80 (6)
	XL	1.50	2.92 (2)	3.09 (18)	2.16 (13)	2.75 (29)
LU	SM	1.16	1.18 (25)	1.18 (25)	1.12 (30)	1.11 (19)
	ML	1.15	1.20 (24)	1.20 (24)	1.17 (26)	1.19 (5)
	XL	1.00	1.00 (3)	1.00 (3)	0.98 (13)	1.00 (29)
Syr2k	SM	2.06	2.90 (2)	3.32 (18)	2.34 (12)	2.41 (11)
	ML	0.80	1.17 (2)	1.22 (16)	0.93 (29)	0.85 (30)
	XL	0.95	1.09 (2)	1.09 (2)	0.42 (23)	0.85 (26)

Remaining Work

- Prototype has demonstrated key capabilities
  - Transfer autotuning without calibration
  - Budgeting capability
  - Short term only: No continuous learning
- Improvements
  - Tune multiple scales at once (data + system)
  - Incorporate other tools for longer term effectivness

Outline

- Overview
- Components of the Proposal
  - Immersion Cooling Study
  - Optimizing for Generations of Hardware
  - Efficient and Transferable Multi-Scale Tuning
- **Timeline**

Immersion Cooling Study

- Remaining SPLIC experiments
  - To be completed by mid-May
- Air-cooled experiments
  - To be completed by late-May
- Publication target
  - IISWC’24 (late-May submission)

Optimizing for Generations of Hardware

- Work tentatively completed
  - Best Paper award at ICS'21
    - FULL-W2V: Fully Exploiting Data Reuse for W2V on GPU-Accelerated Systems

Efficient and Transferable Multi-Scale Tuning

- Initial prototype successful with single-scale tuning
  - Published ICS'23
    - Transfer-learning-based Autotuning using Gaussian Copula
- Multi-scale tuning requires more novel contributions
  - Collaboration with Iowa State University
    - Currently validating experimental methodology
    - Seeking publication summer 2024
  - Collaboration with Argonne National Lab
    - Currently improving proof of concept
    - Seeking publication early 2025