Dynamic GPGPU Power Management Using Adaptive Model Predictive Control

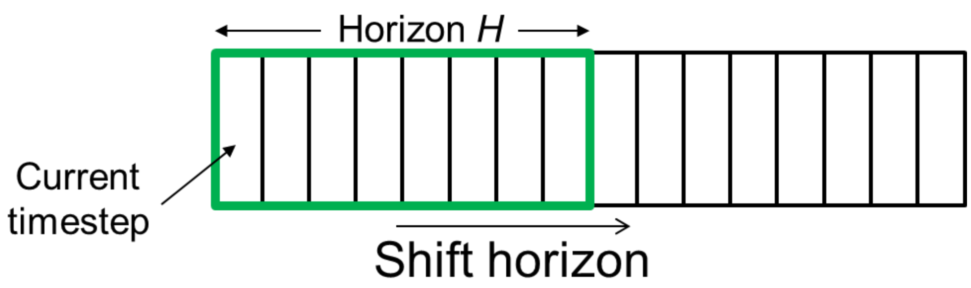
Abhinandan Majumdar[*], Leonardo Piga[†], Indrani Paul[†], Joseph L. Greathouse[†], Wei Huang[†], David H. Albonesi[*]

[*]Computer Systems Laboratory, Cornell University

[†]Advanced Micro Devices, Inc.

Model Predictive Control (MPC)

- Previous dynamic power management policies ignore future application behavior
  - Leads to performance loss or wasted energy
- MPC *looks into the future* to determine the best configuration for the current optimization time step

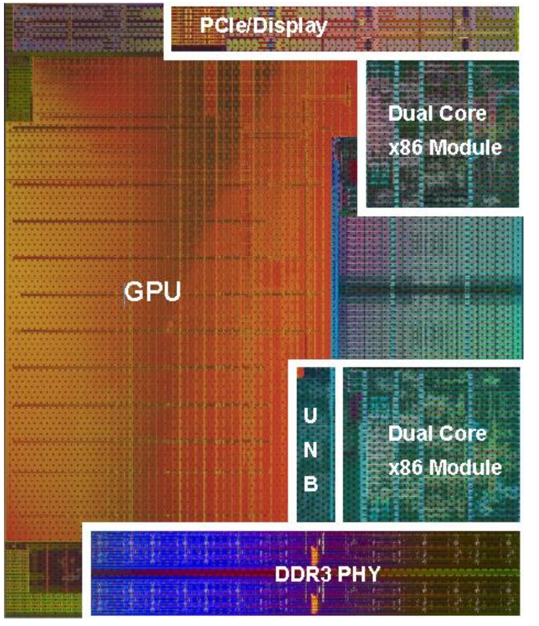


- Though effective in many domains, overheads far too high for short timescales of dynamic power management
- <u>Our approach</u>: An approximation of MPC that dramatically improves GPGPU energy-efficiency with orders of magnitude lower overhead

Dynamic GPGPU Power Management

- Attempts to maximize performance within power constraints

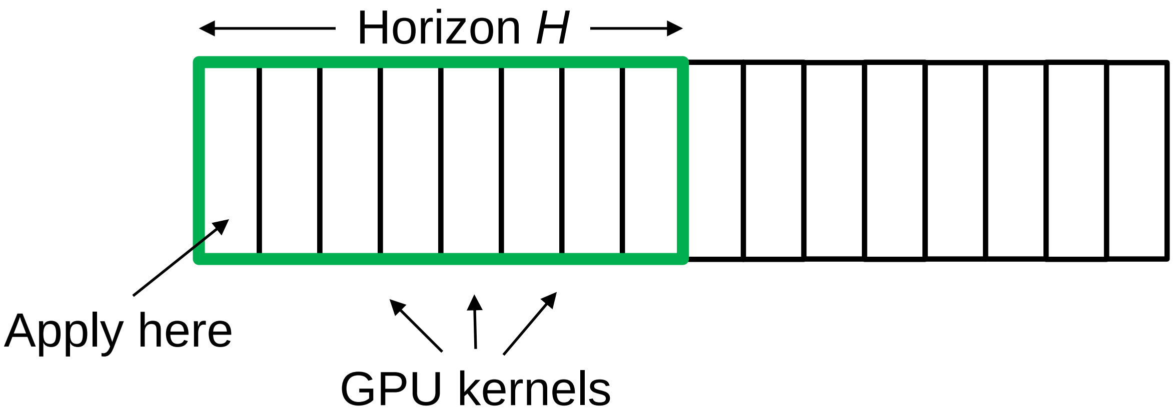
- Hardware knobs
  - Number of active GPU Compute Units
  - DVFS states



- <u>Goal</u>: Reduce the energy of the GPU application phases compared to the baseline power manager while matching its performance

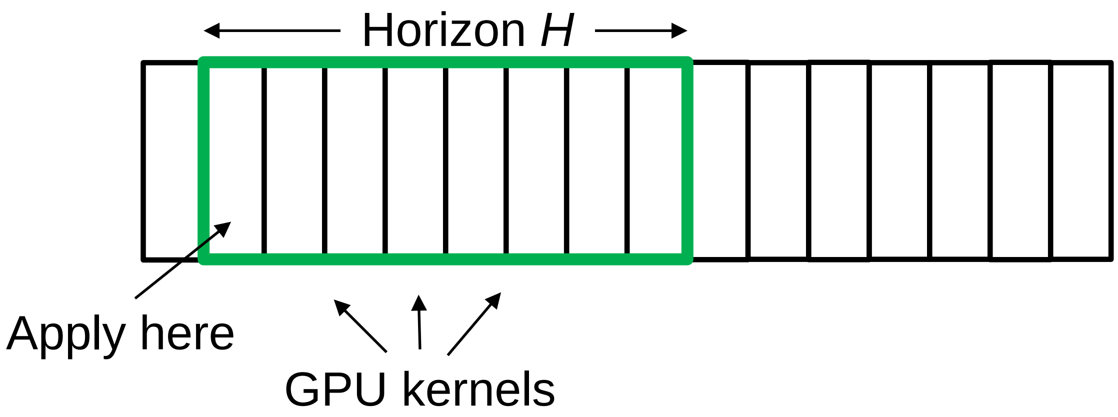
Applying MPC to Dynamic Power Management

- General Idea



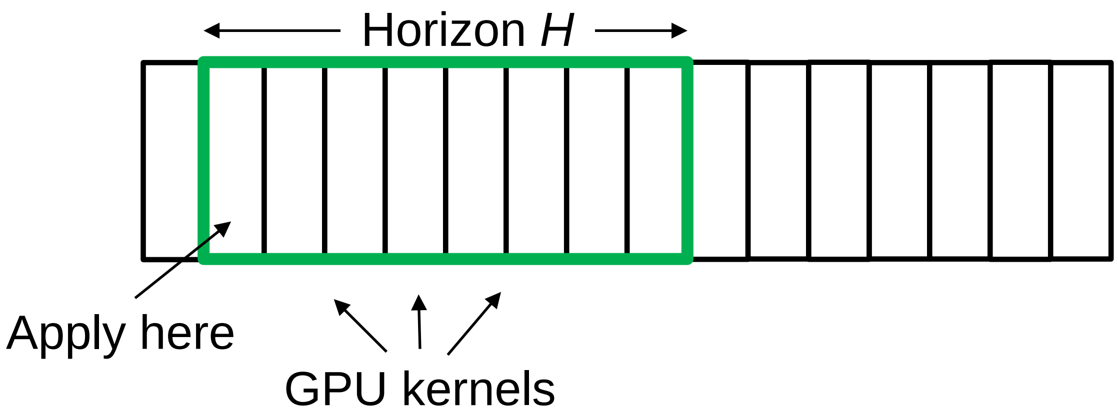
Applying MPC to Dynamic Power Management

- General Idea



Applying MPC to Dynamic Power Management

- General Idea



- MPC has high overhead
  - Complexity scales exponentially with *H*
  - Minimizing energy under performance cap with discrete HW settings is fundamentally NP-hard
  - Usually requires dedicated optimization solvers, such as CVX, lpsolve, etc.

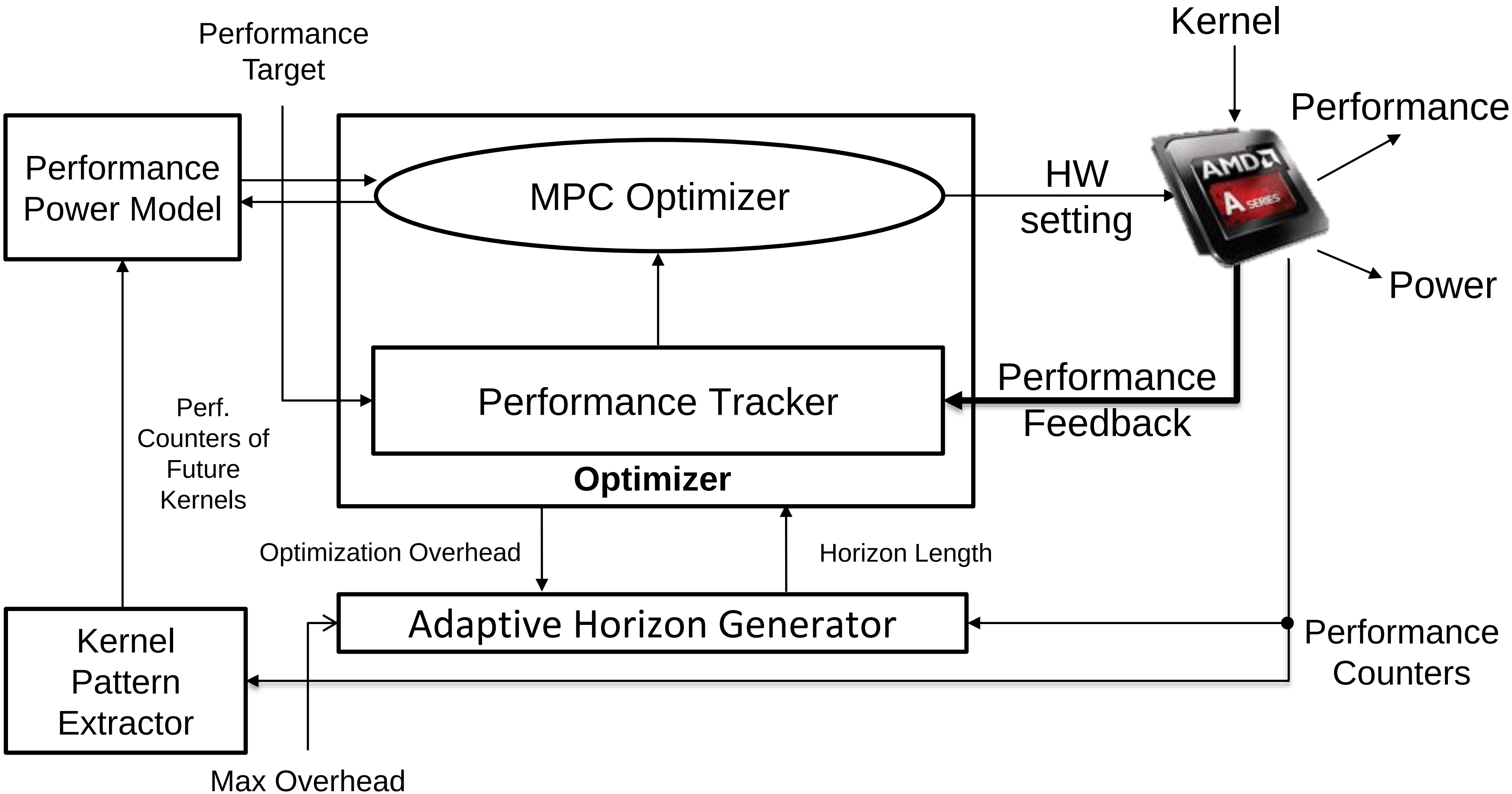
Approximations to MPC

- Greedy Hill Climbing to reduce the search space

- Static search order heuristic to make MPC optimization polynomial rather than exponential

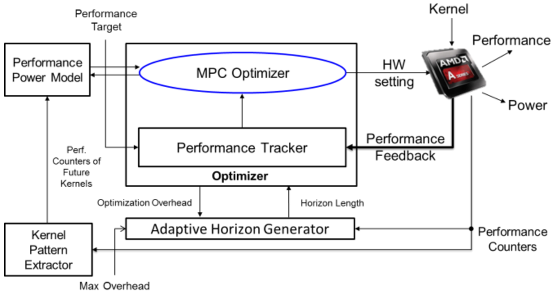
- Dynamically tuning of the horizon length *H* to limit the optimization overhead

MPC Power Manager Architecture

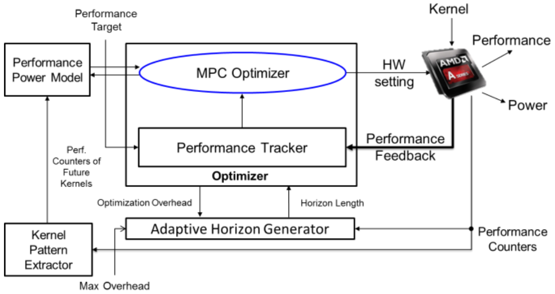


MPC Optimizer

- <u>Objective:</u> Find minimum energy HW setting for each kernel without impacting overall performance



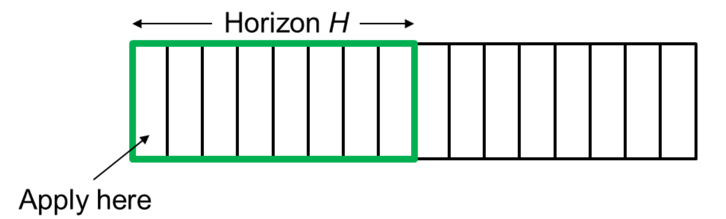
MPC Optimizer



- Objective: Find minimum energy HW setting for each kernel without impacting overall performance
- Greedy Hill Climbing optimization
  - Select the HW knob with highest energy sensitivity
  - Search for the lowest energy configuration using hill climbing
- MPC Search Heuristic
  - Determines a static order without backtracking
  - Search cost becomes polynomial
  - Details in the paper
- 65X average reduction in total cost evaluations

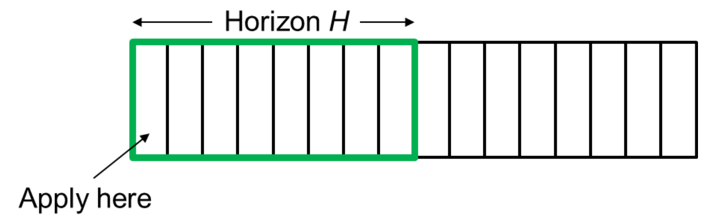
Adaptive Horizon Generator

- Longer horizon improves savings but increases MPC optimization time



Adaptive Horizon Generator

- Longer horizon improves savings but increases MPC optimization time

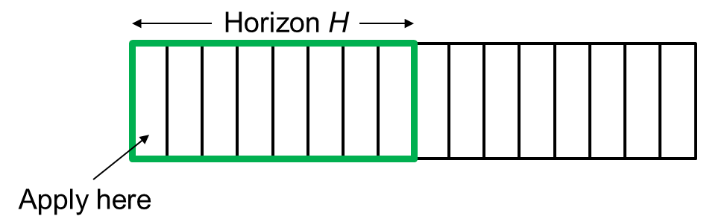


- Limit the overhead to a slowdown factor α by dynamically varying the horizon length

$$H_i \leq \frac{N}{\mathbb{N}} \frac{(1 + \alpha - \frac{1}{i})\frac{i \times T_{total}}{N} - \sum_{j=1}^{i-1}(T_j + T_{MPC,j})}{T_{PPK}}$$

Adaptive Horizon Generator

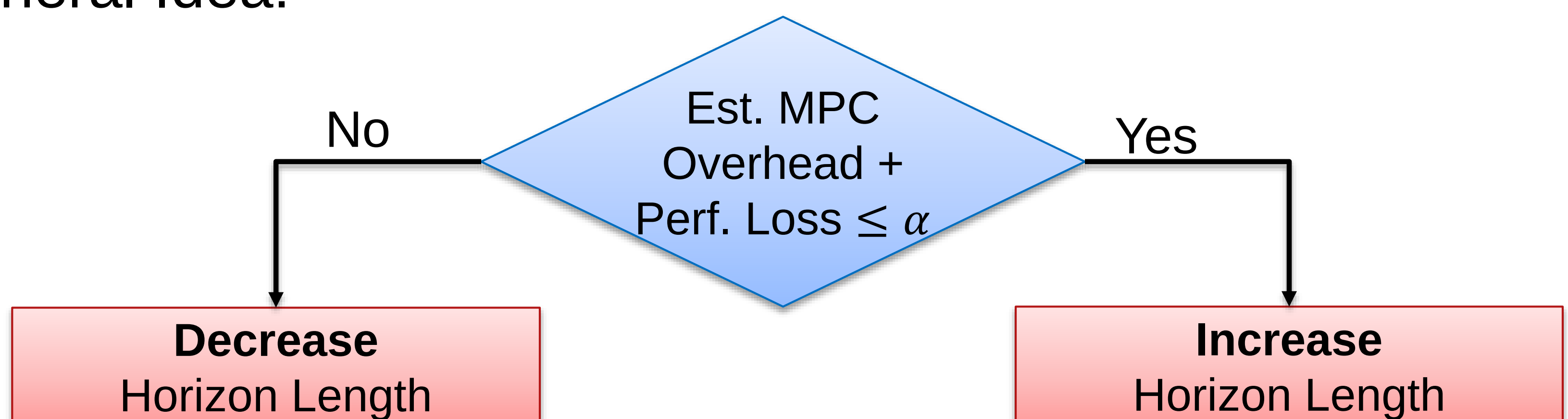
- Longer horizon improves savings but increases MPC optimization time



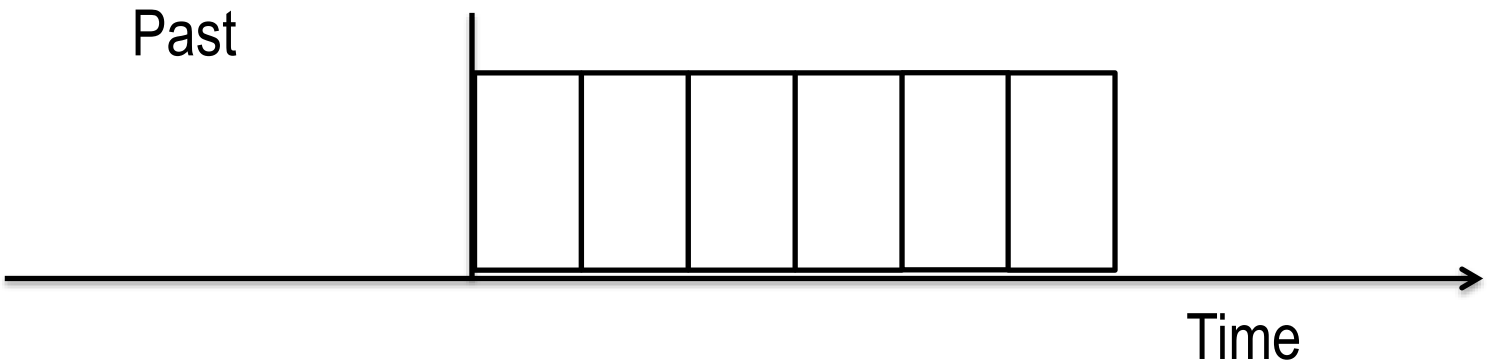
- Limit the overhead to a slowdown factor α by dynamically varying the horizon length

$$H_i \leq \frac{N}{\mathbb{N}} \frac{(1+\alpha-\frac{1}{i})\frac{i\times T_{total}}{N} - \sum_{j=1}^{i-1}(T_j + T_{MPC,j})}{T_{PPK}}$$

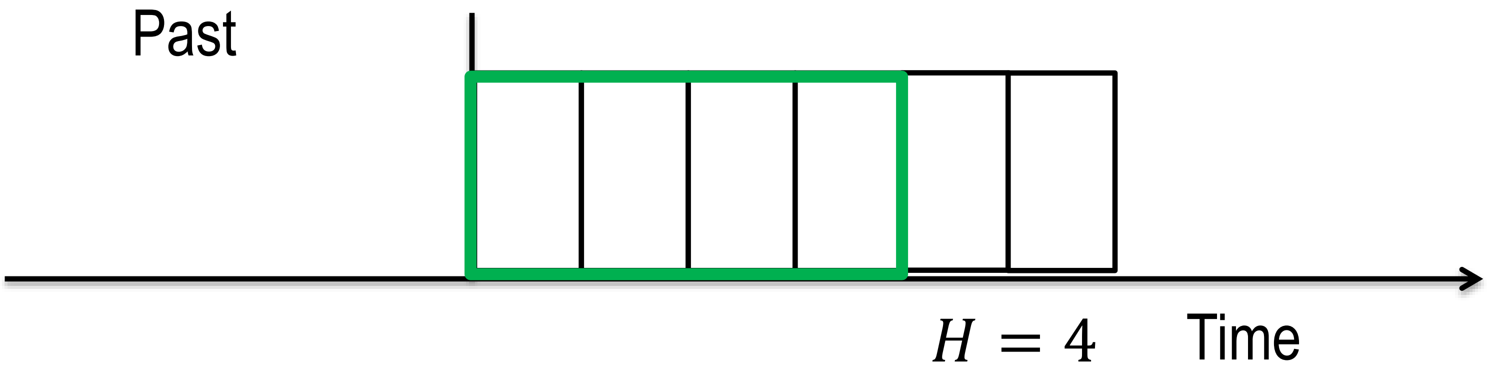
- General Idea:



MPC in Action – An Example



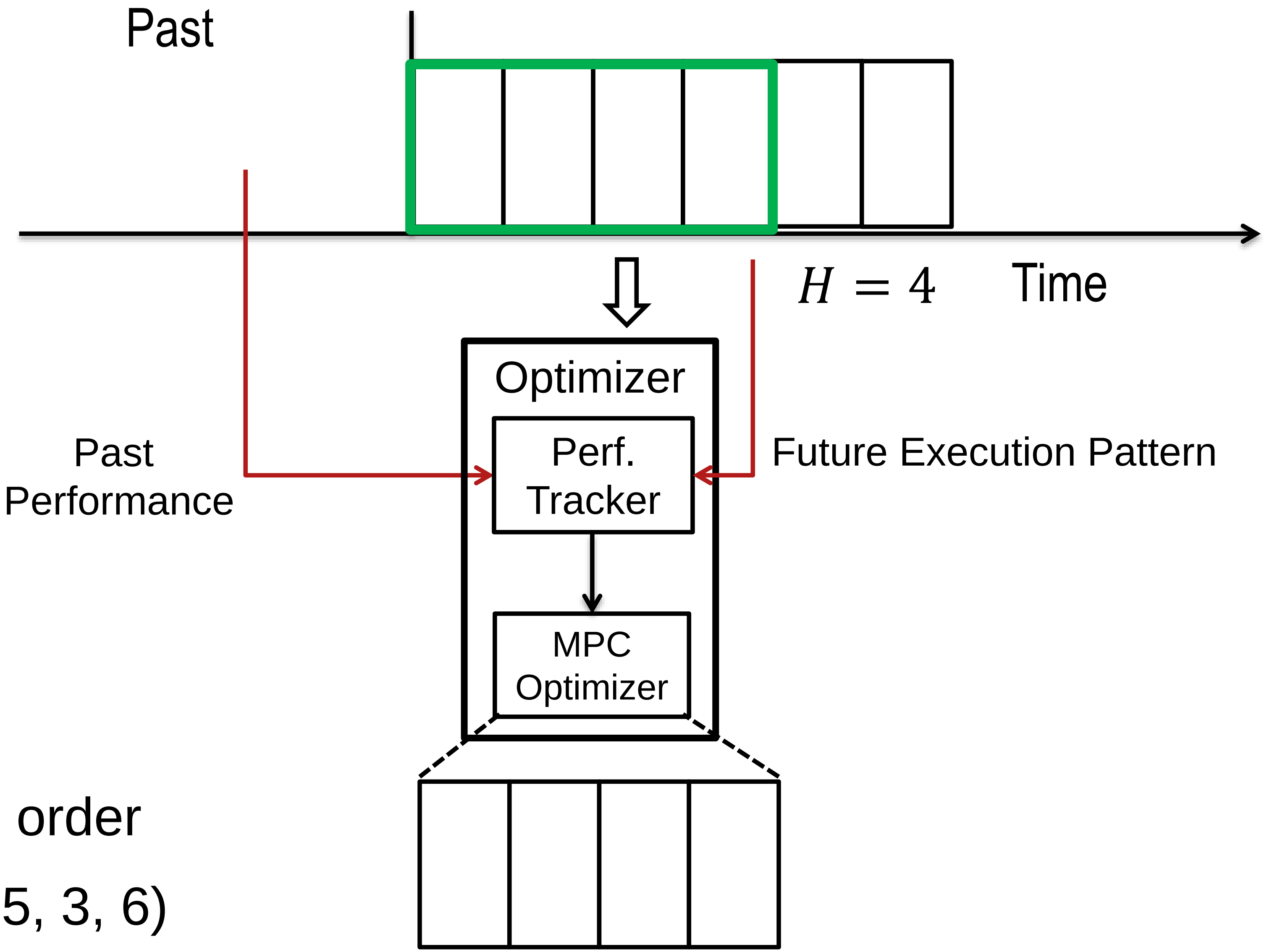
MPC in Action – An Example



Search order

(2, 1, 4, 5, 3, 6)

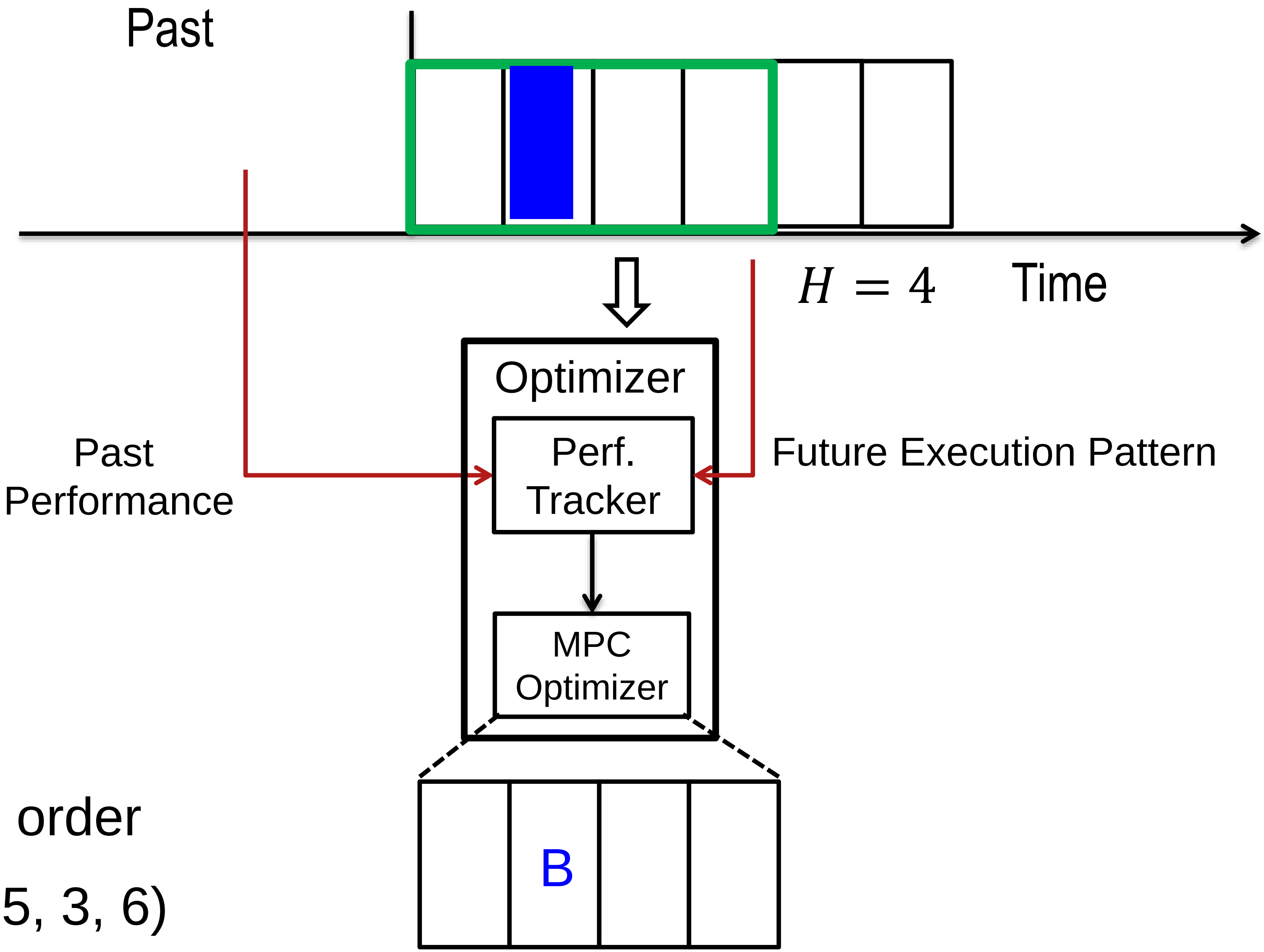
MPC in Action – An Example



Search order

(2, 1, 4, 5, 3, 6)

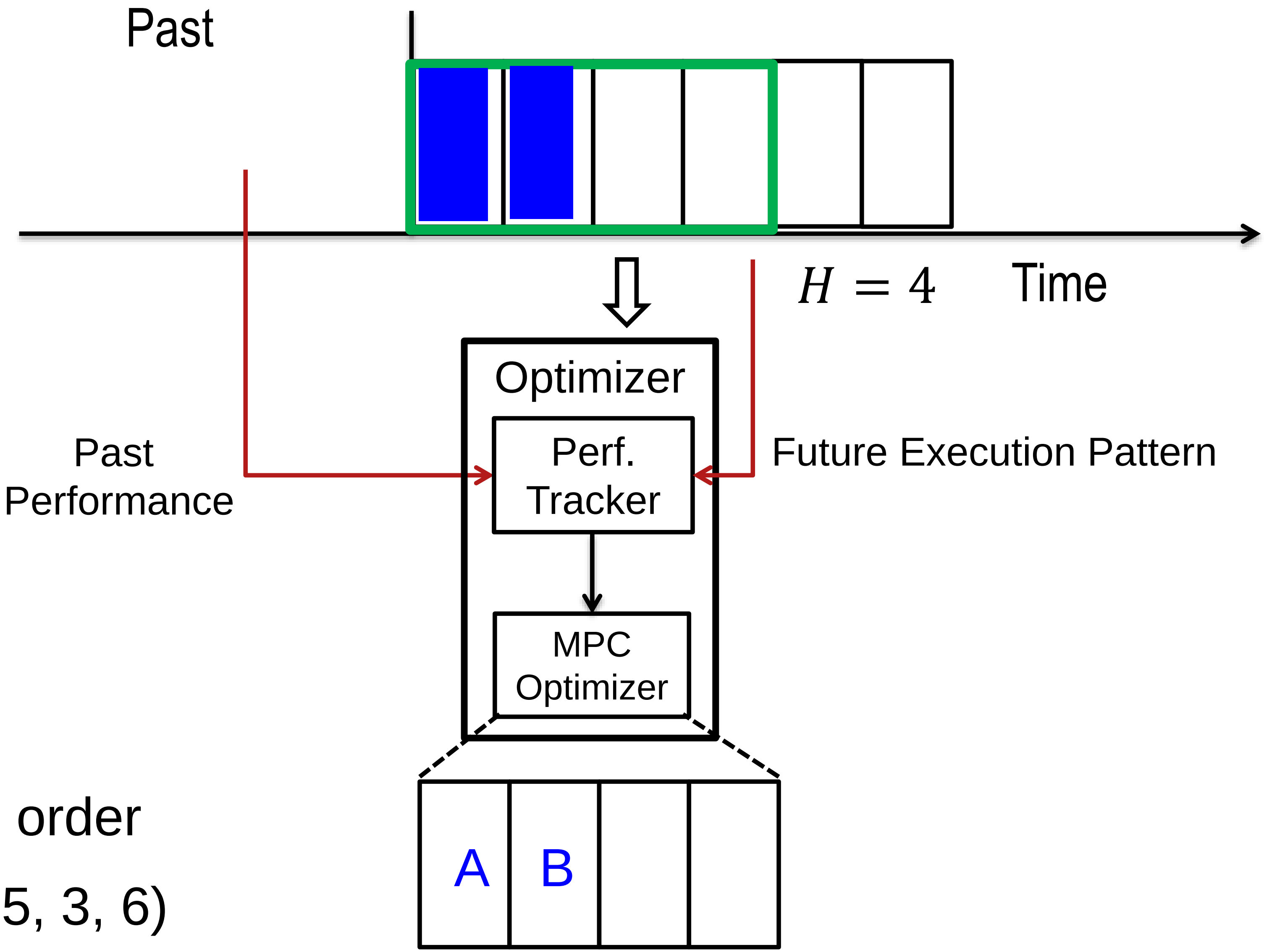
MPC in Action – An Example



Search order

(2, 1, 4, 5, 3, 6)

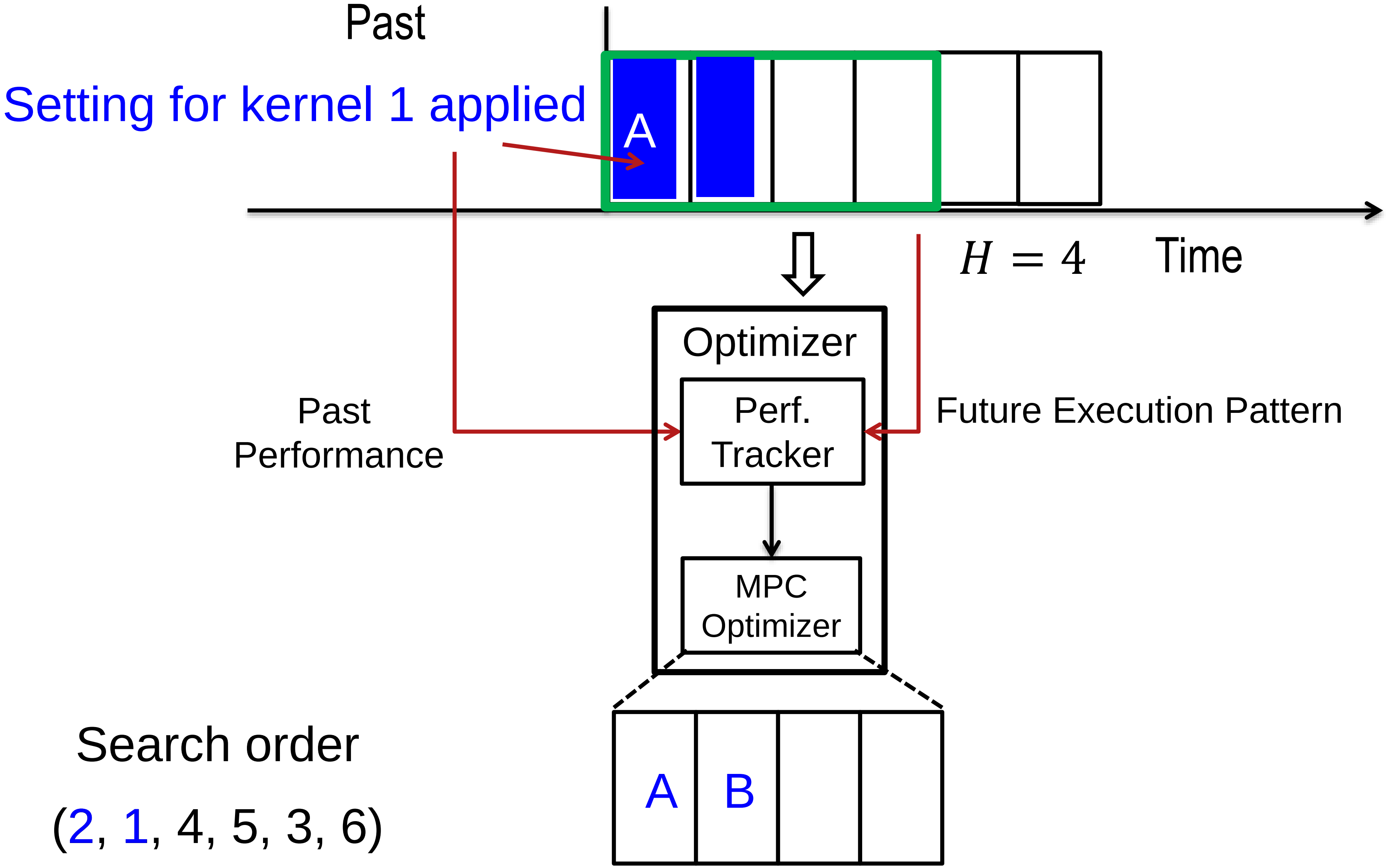
MPC in Action – An Example



Search order

(2, 1, 4, 5, 3, 6)

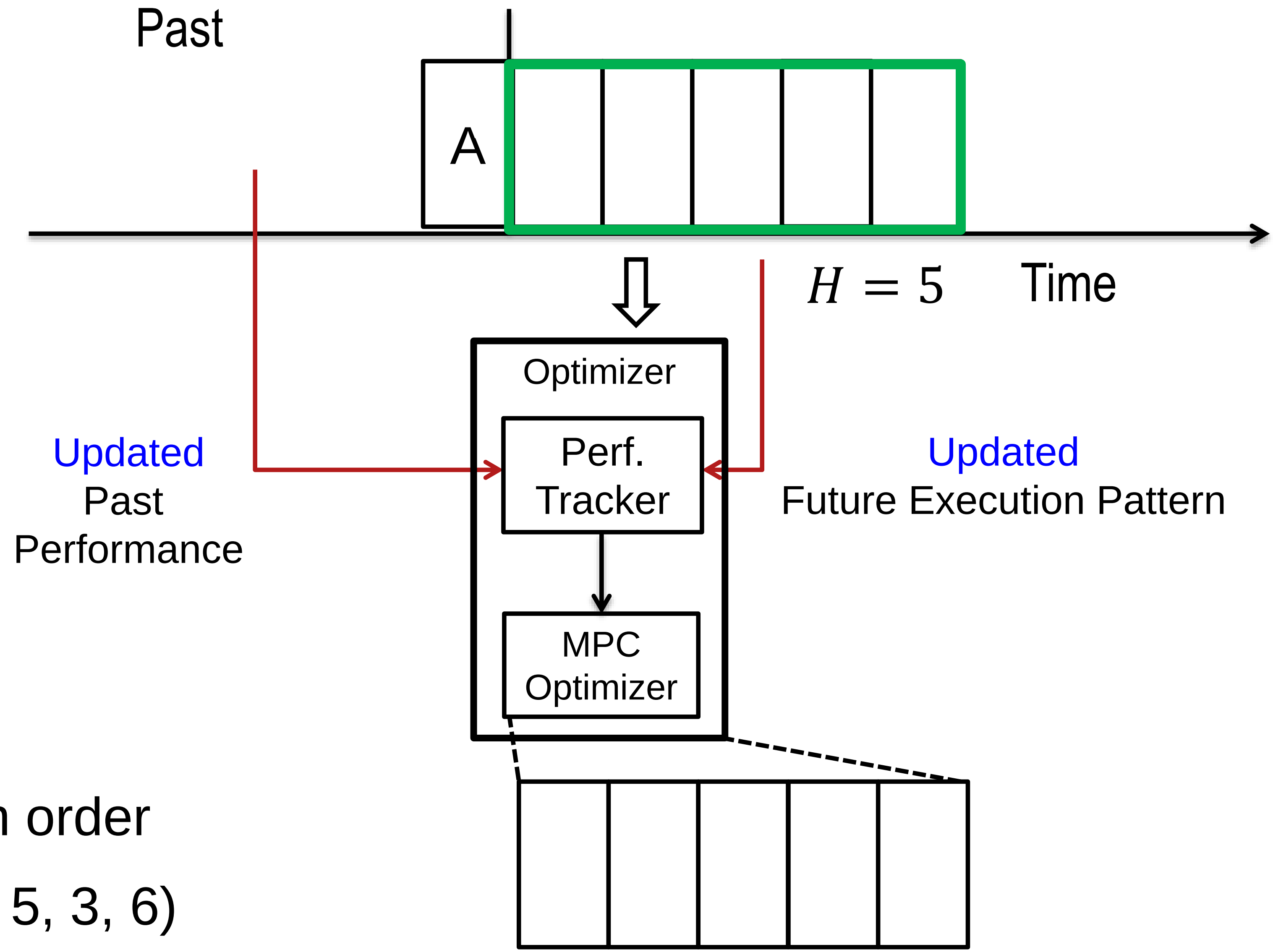
MPC in Action – An Example



Search order

(2, 1, 4, 5, 3, 6)

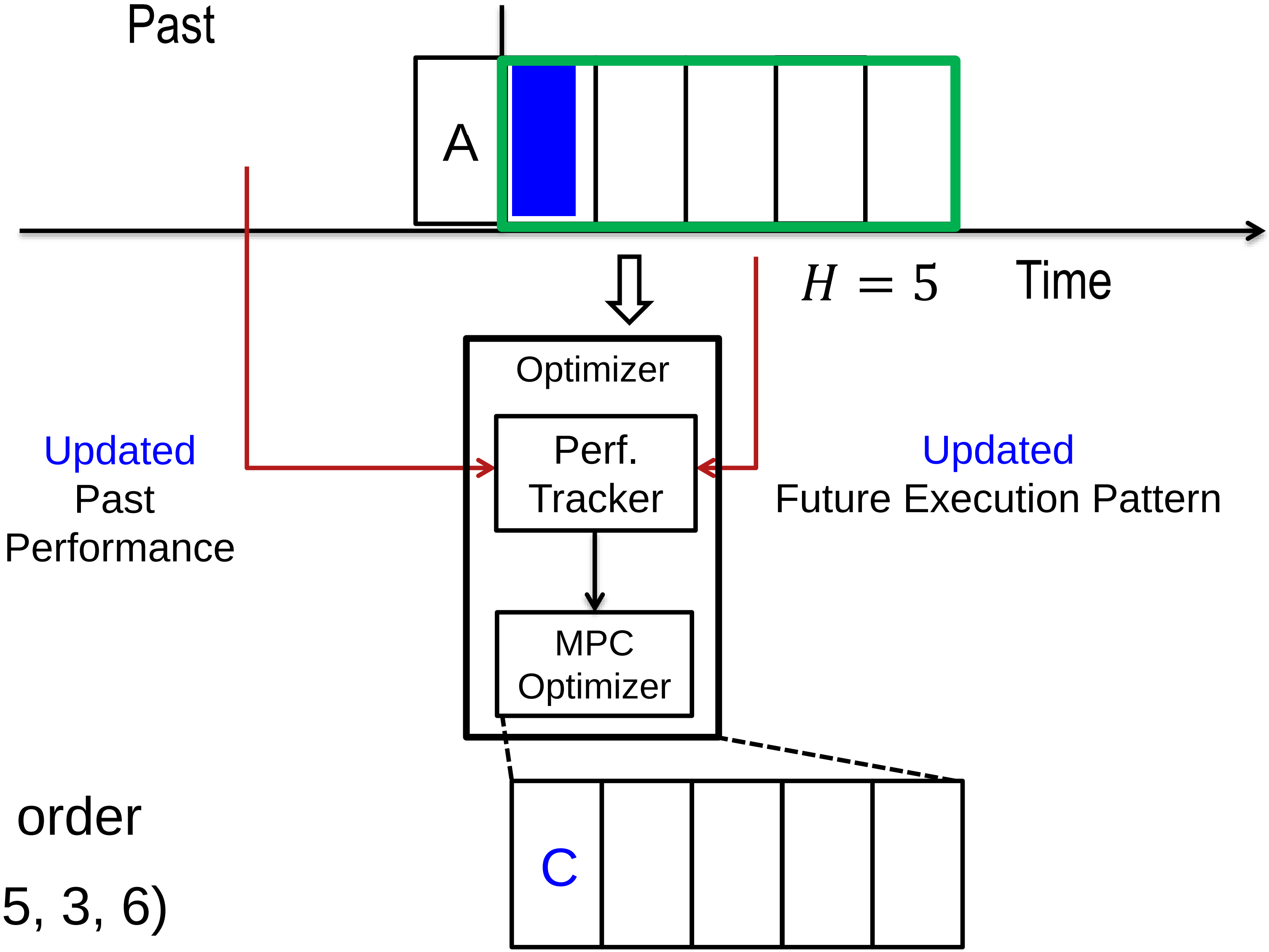
MPC in Action – An Example



Search order

(2, 1, 4, 5, 3, 6)

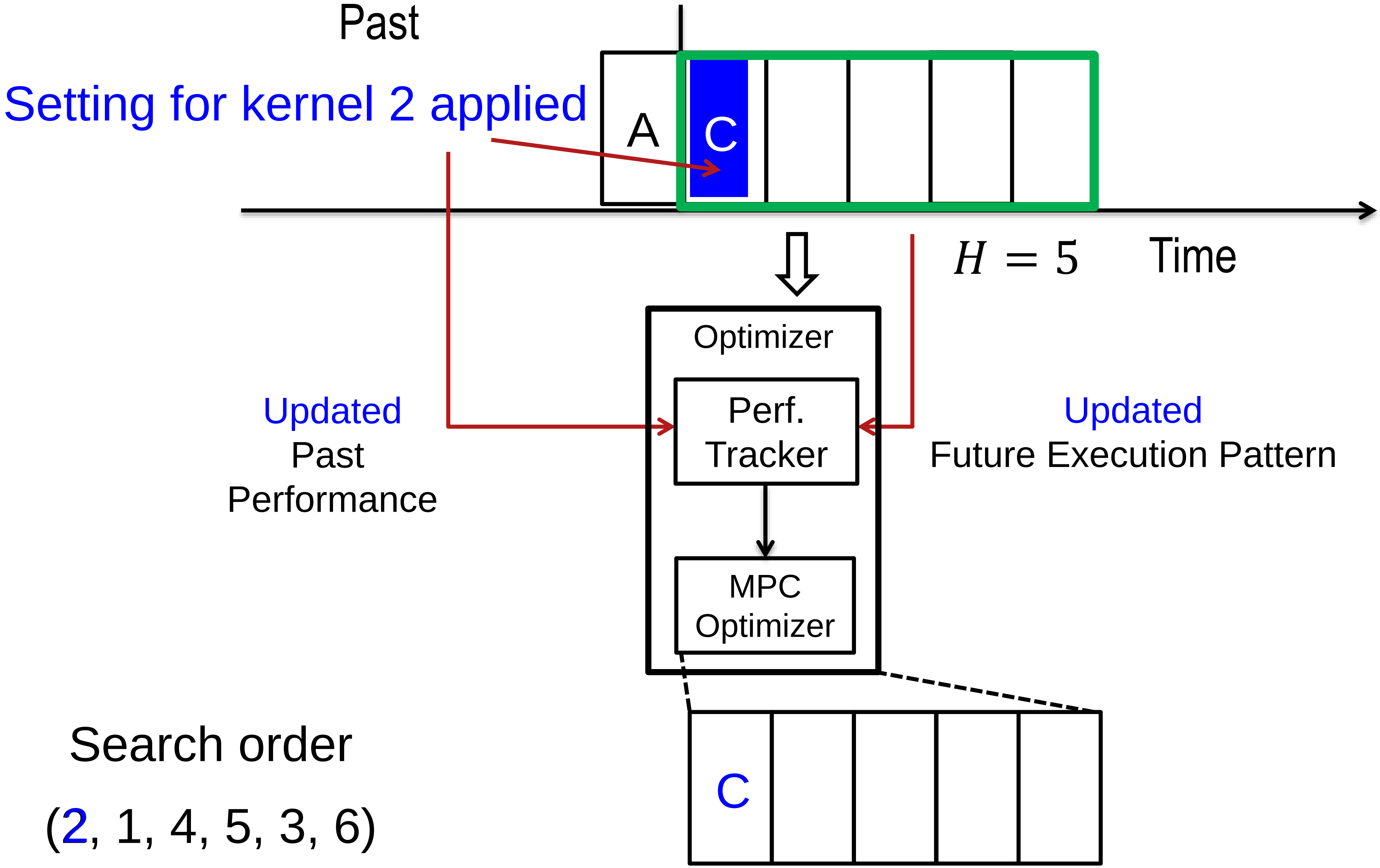
MPC in Action – An Example



Search order

(2, 1, 4, 5, 3, 6)

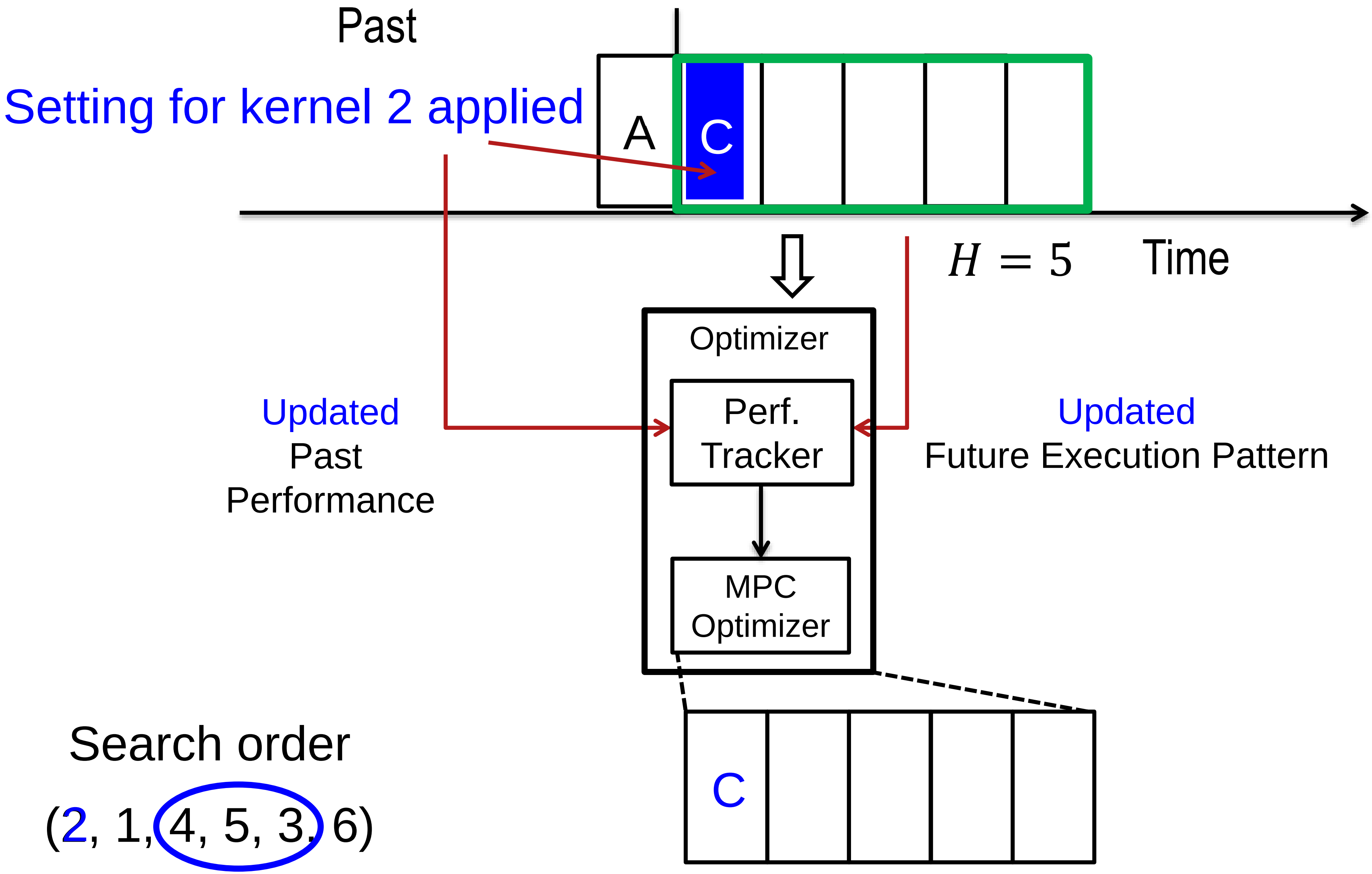
MPC in Action – An Example



Search order

(2, 1, 4, 5, 3, 6)

MPC in Action – An Example

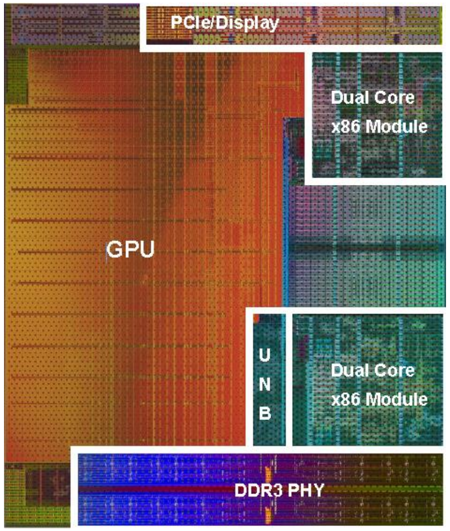


Search order

(2, 1, 4, 5, 3, 6)

Experimental Testbed

- AMD A10-7850K APU
  - 2 out-of-order dual core CPUs
  - GPU contains 512 processing elements (8 CUs) at 720 MHz



- DVFS states

CPU P States	Voltage (V)	Freq (GHz)
P1	1.325	3.9
P2	1.3125	3.8
P3	1.2625	3.7
P4	1.225	3.5
P5	1.0625	3
P6	0.975	2.4
P7	0.8875	1.7

NB P States	Freq (GHz)	Memory Freq (MHz)
NB0	1.8	800
NB1	1.6	800
NB2	1.4	800
NB3	1.1	333

GPU P States	Voltage (V)	Freq (GHz)
DPM0	0.95	351
DPM1	1.05	450
DPM2	1.125	553
DPM3	1.1875	654
DPM4	1.225	720

NB and GPU share the same voltage rail

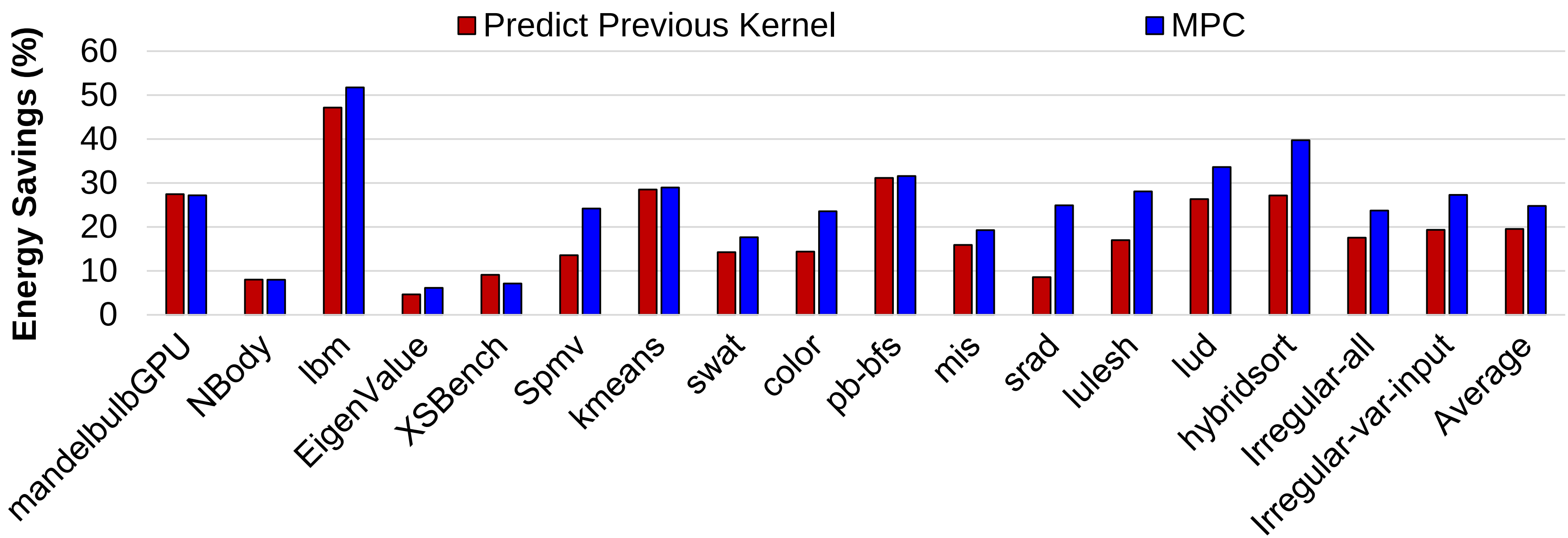
- Total HW configuration: 336

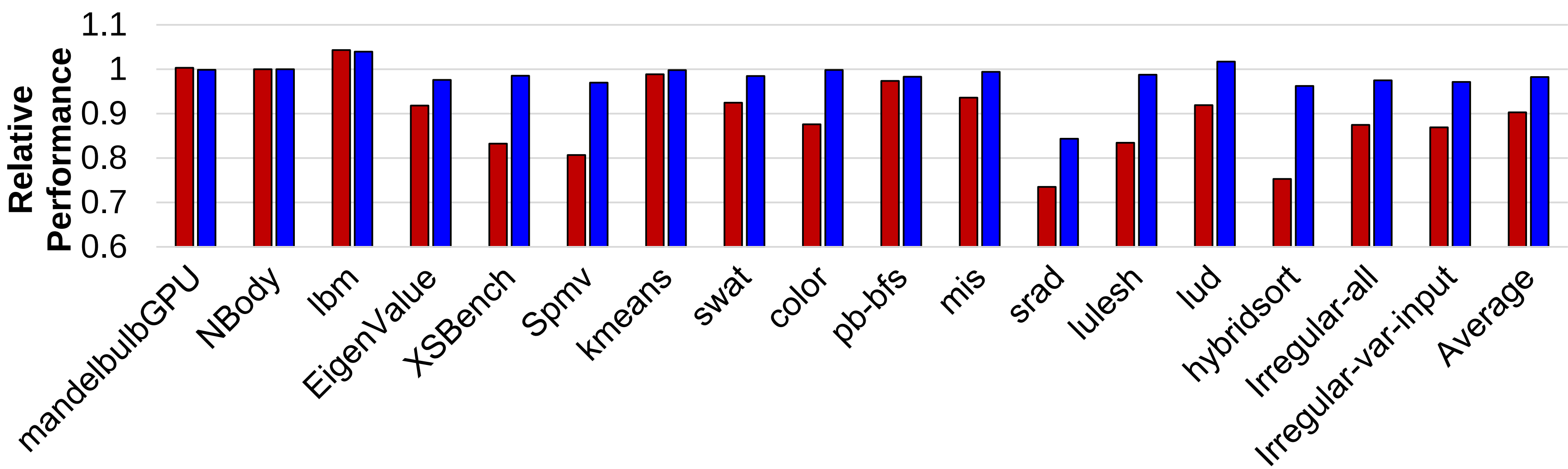
Experimental Setup

- 15 GPGPU Benchmarks
- Baseline scheme
  - AMD Turbo Core
- Predict Previous Kernel (PPK)
  - Assume last kernel repeats
  - State-of-the-art: Harmonia ISCA'15, McLaughlin et al. ASBD'14
- Maximum overhead α = 5%

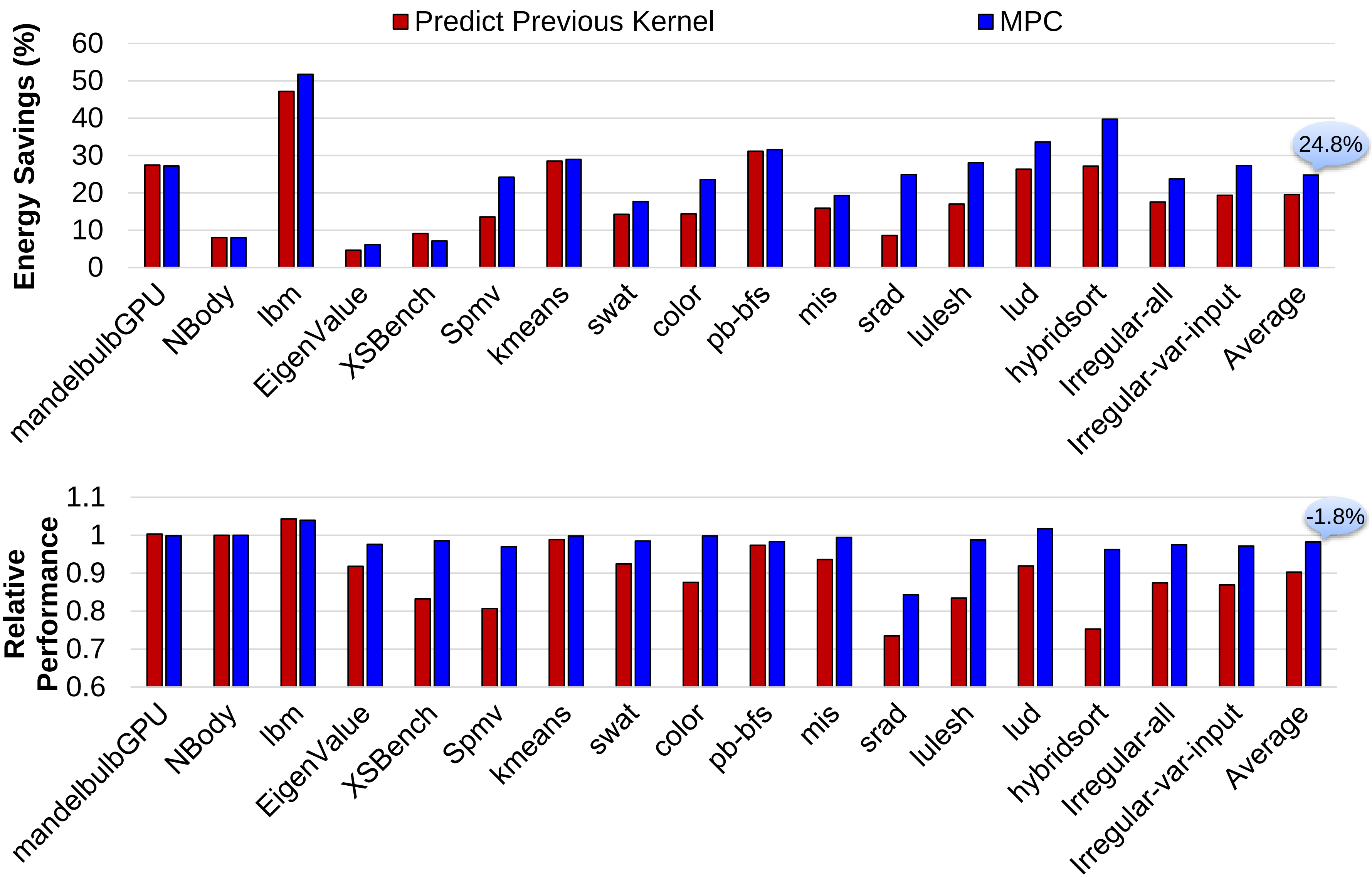
Category	**Benchmarks**	**Benchmark Suite**	**Regular Expression**
Regular	mandelbulbGPU	Phoronix	A^{20}
	Nbody	AMD APP SDK	A^{10}
	juliaGPU	Phoronix	A^{10}
Irregular with repetitive pattern	EigenValue	AMD APP SDK	$(AB)^{5}$
	XSBench	Exascale	$(ABC)^{2}$
Irregular with non-repetitive pattern	Spmv	SHOC	$A^{10}B^{10}C^{10}$
	Kmeans	Rodinia	AB^{20}
Irregular with kernels varying with input	swat	OpenDwarfs	Complex pattern
	color	Pannotia	
	pb-bfs	Parboil	
	mis	Pannotia	
	srad	Rodinia	
	lulesh	Exascale	
	lud	Rodinia	
	hybridsort	Rodinia	

Energy-Performance Gains

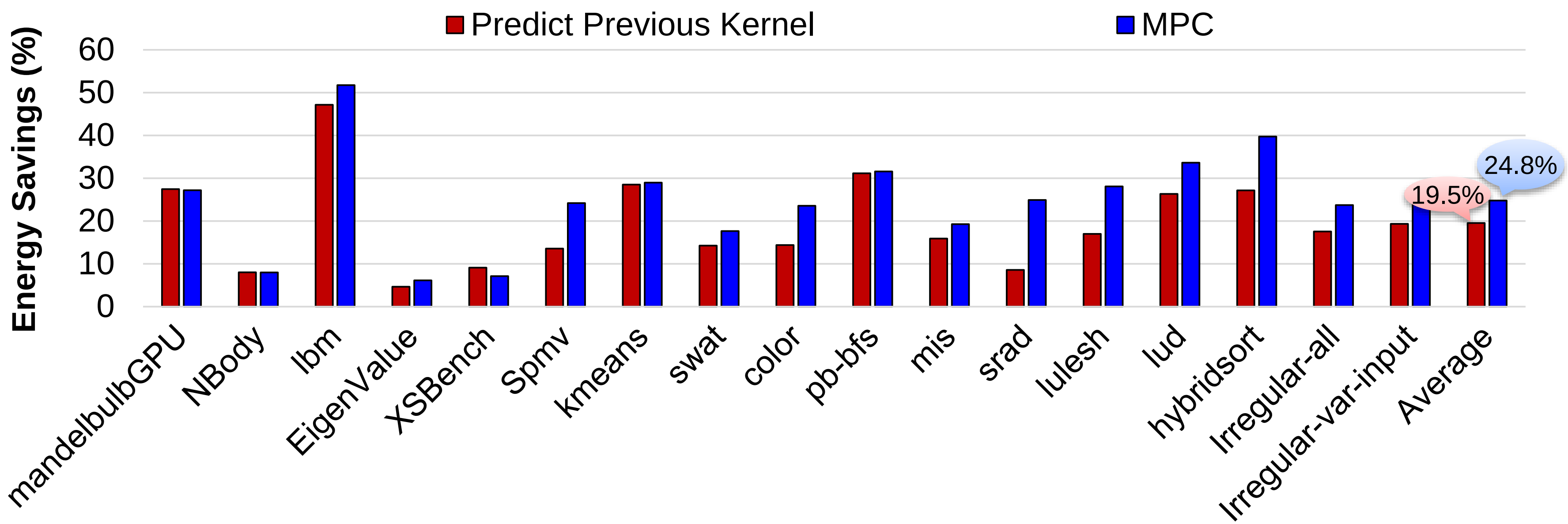


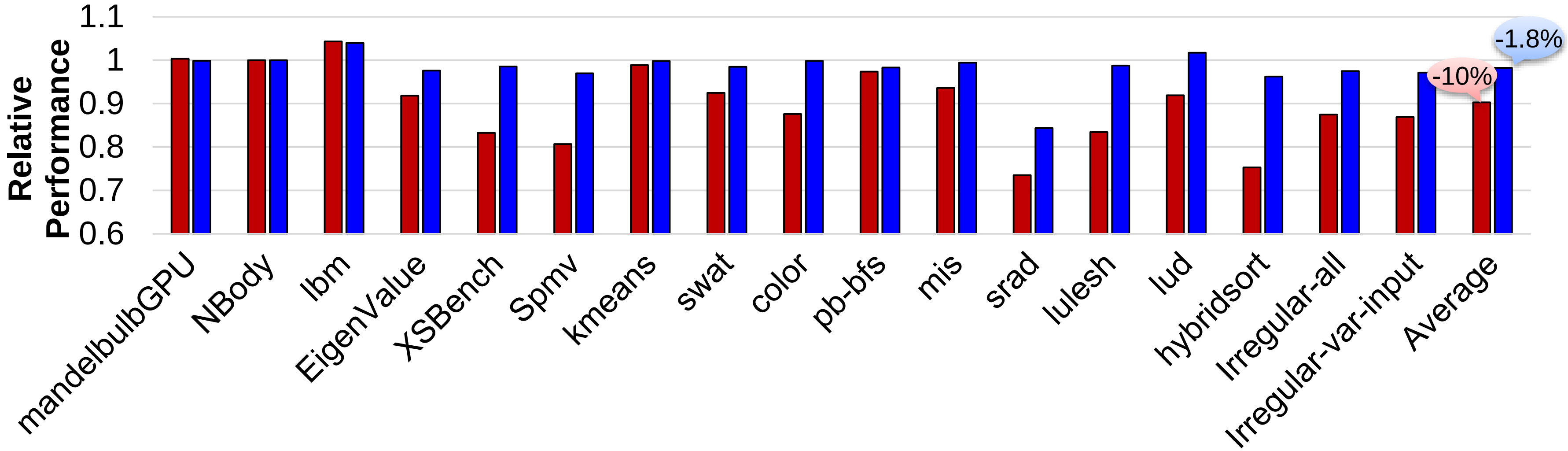


Energy-Performance Gains

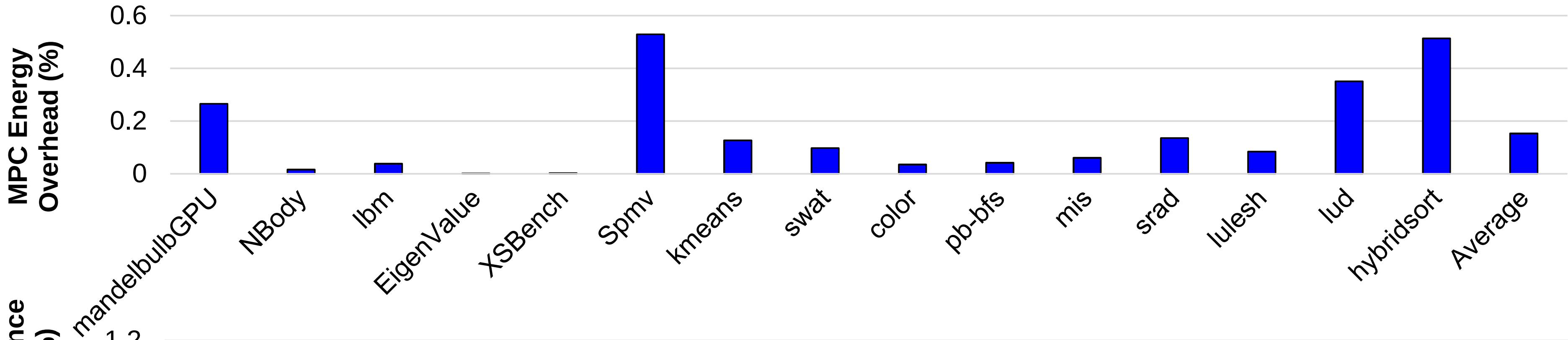


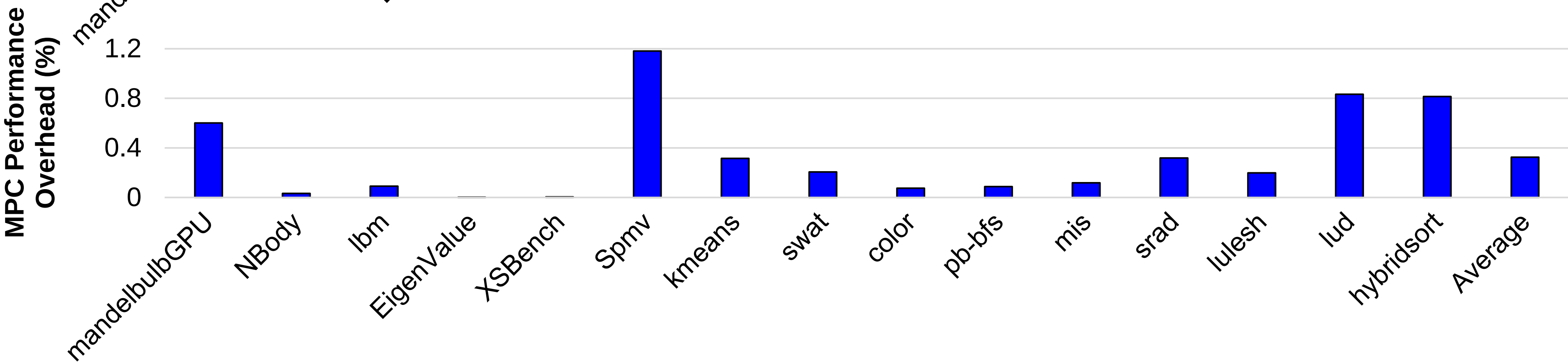
Energy-Performance Gains

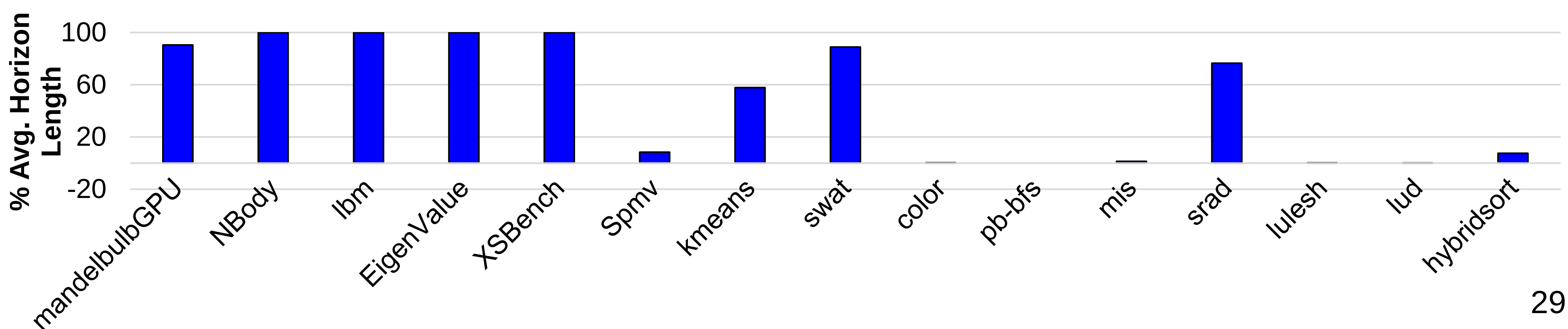




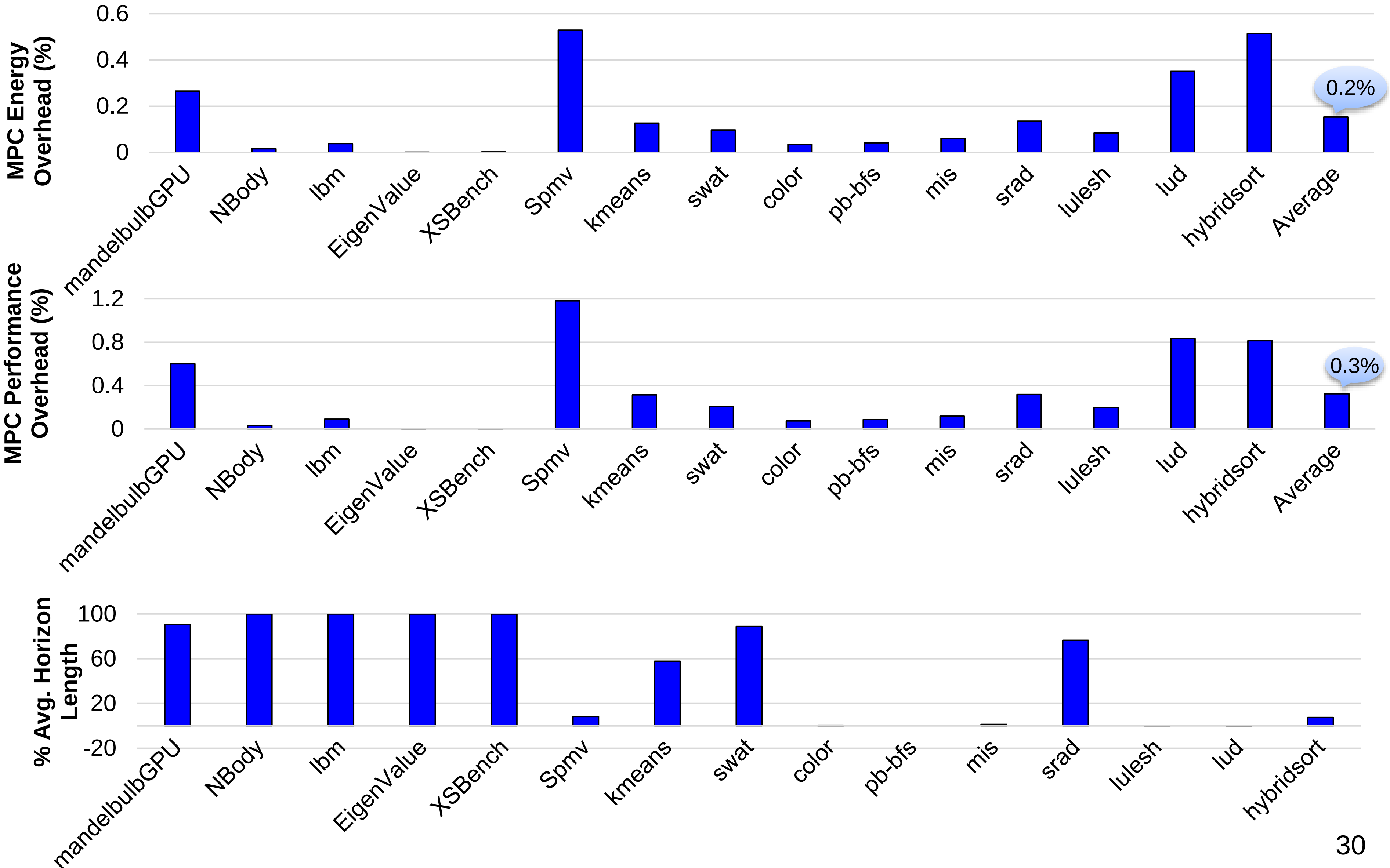
MPC Overhead





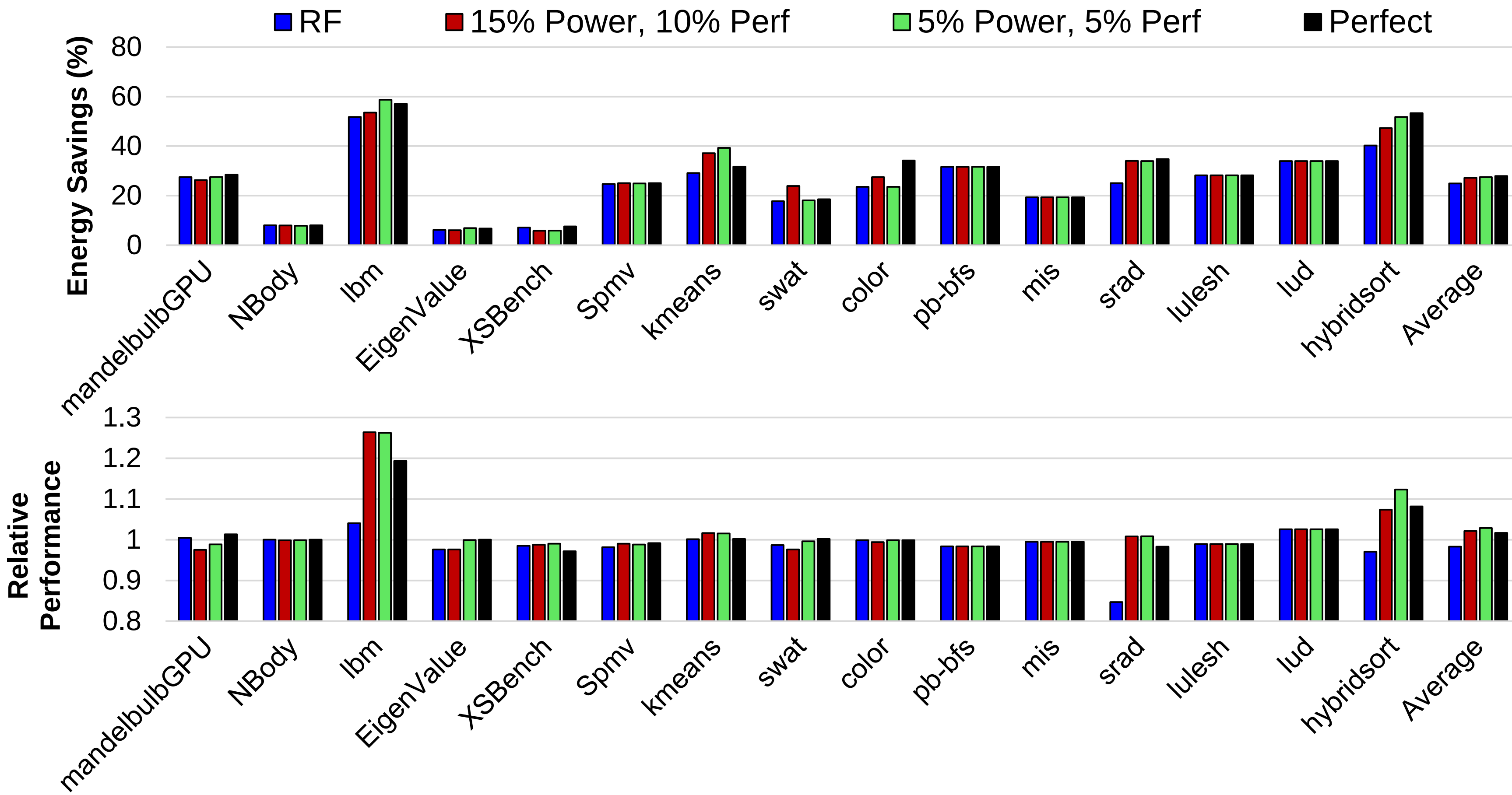


MPC Overhead



Ramification of Prediction Inaccuracy

- RF: 12% Power, 25% Perf
- 15% Power, 10% Perf: Wu et al. [HPCA 2015]
- 5% Power, 5% Perf: Paul et al. [ISCA 2015]



Conclusions

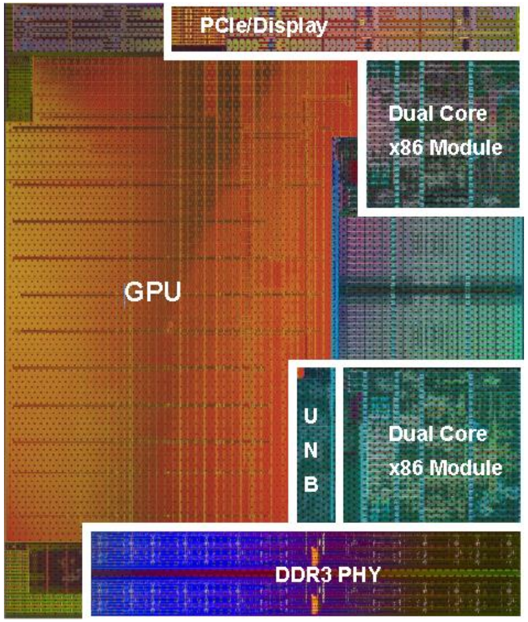
- MPC looks into the future to determine the best configuration for the current optimization time step
- Though effective in many domains, overheads far too high for short timescales of dynamic power management
- We devise an approximation of MPC that dramatically improves GPGPU energy-efficiency with orders of magnitude lower overhead
- Our approach reduces energy by 24.8% with only a 1.8% performance impact

Questions

Backup Slides

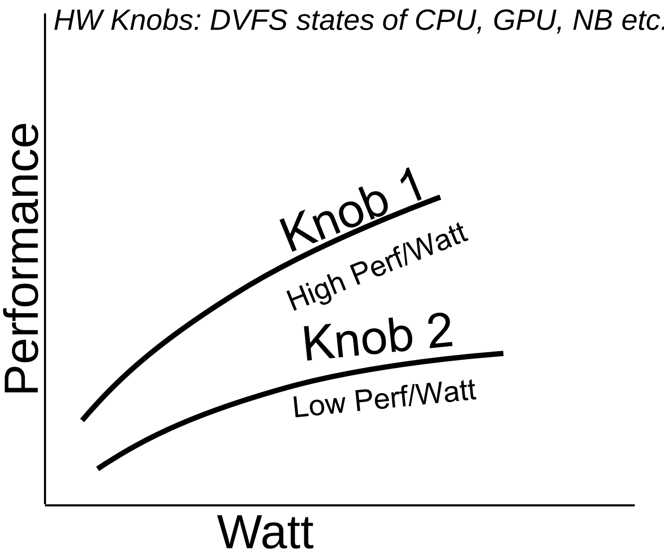
Performance-aware Power Management

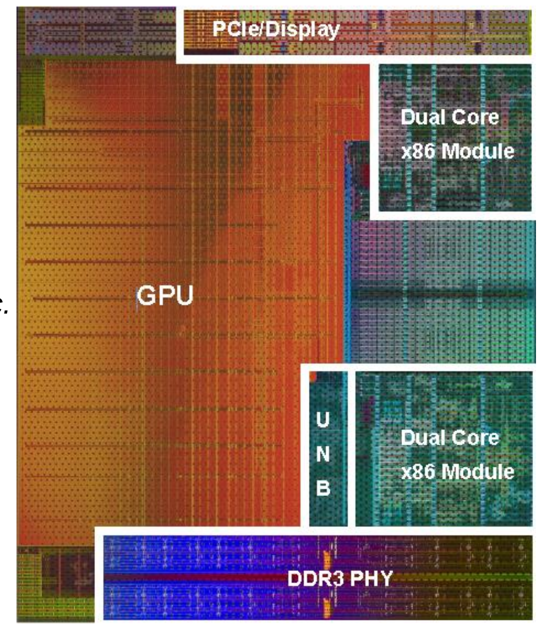
- Optimum node-level power efficiency is a complex function of SOC configuration, workload characteristics, programming model, and node-level objective/constraints



Performance-aware Power Management

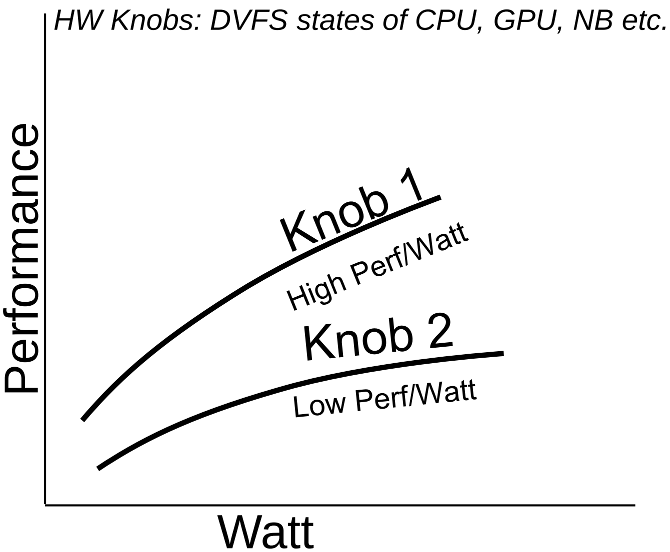
- Optimum node-level power efficiency is a complex function of SOC configuration, workload characteristics, programming model, and node-level objective/constraints

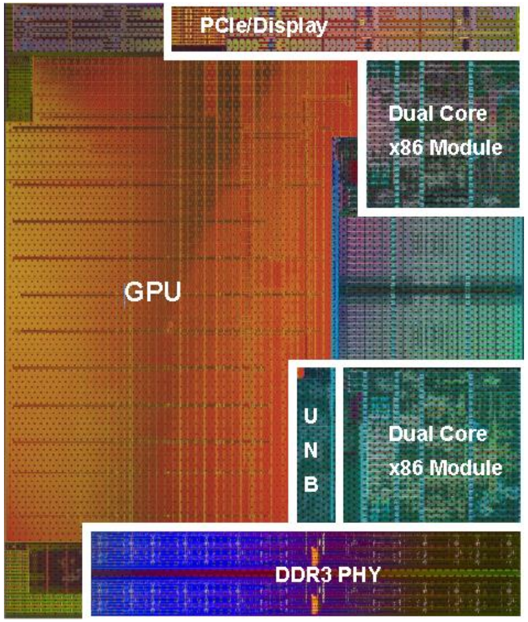


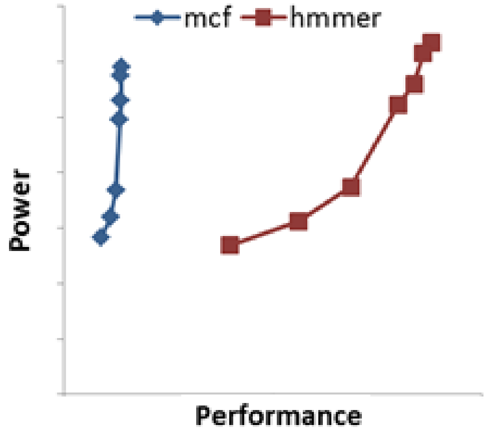


Performance-aware Power Management

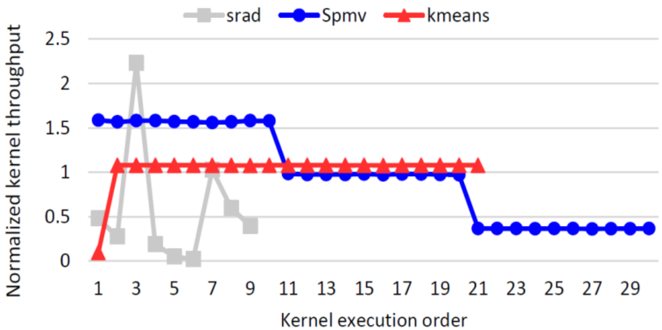
- Optimum node-level power efficiency is a complex function of SOC configuration, workload characteristics, programming model, and node-level objective/constraints







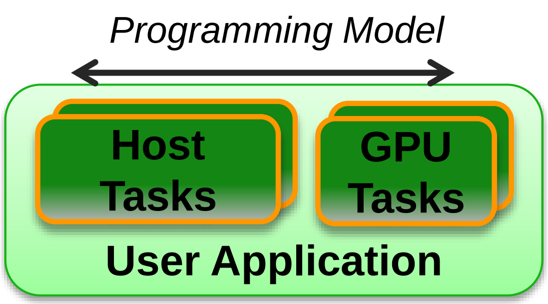
Non-linear power vs. performance curves

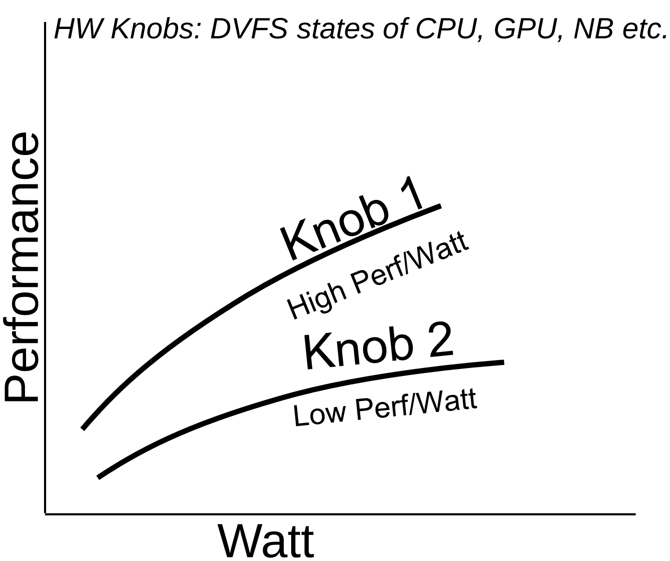


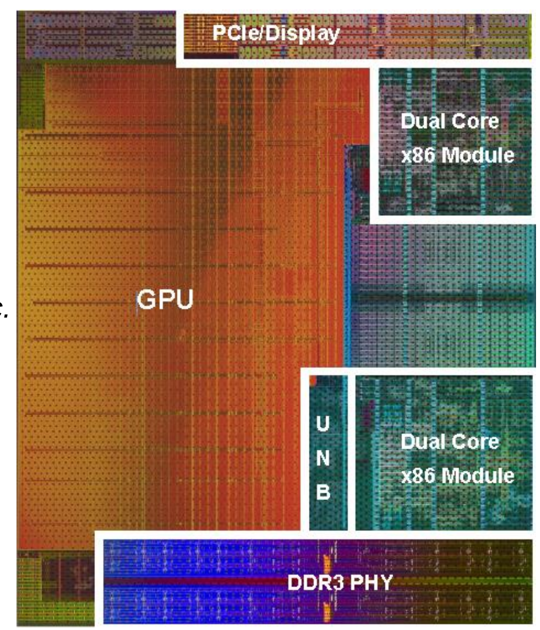
Time-varying Workload

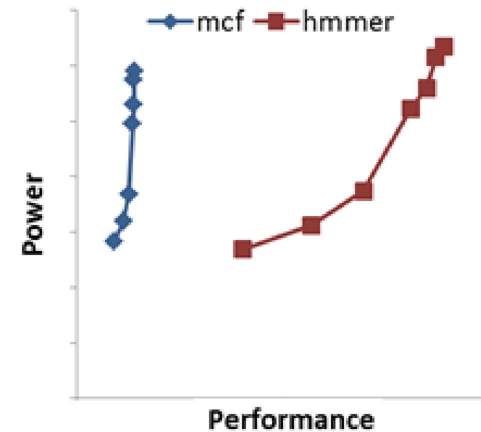
Performance-aware Power Management

- Optimum node-level power efficiency is a complex function of SOC configuration, workload characteristics, programming model, and node-level objective/constraints

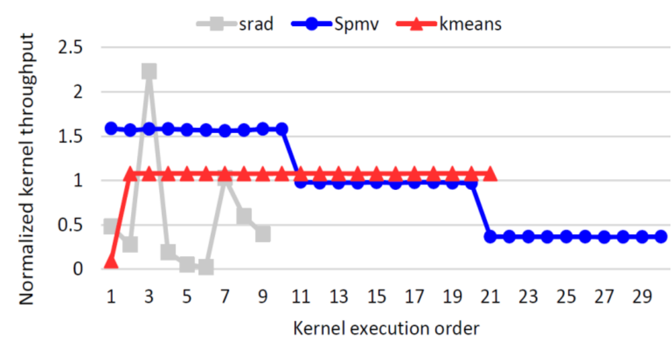








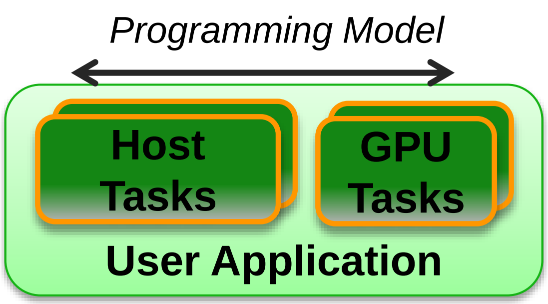
Non-linear power vs. performance curves

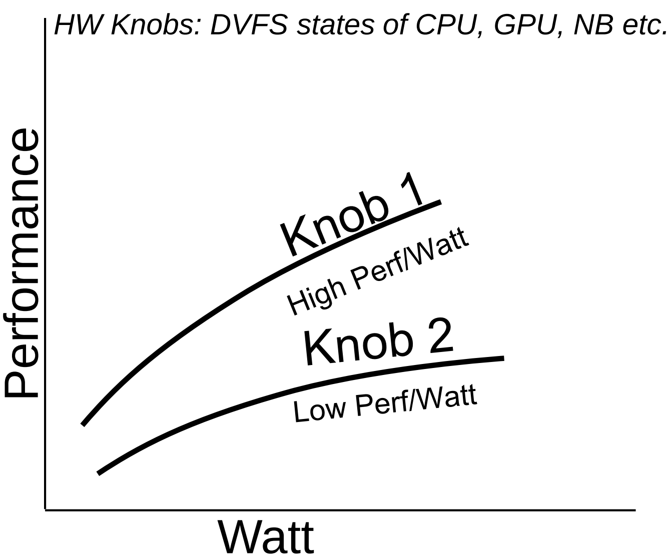


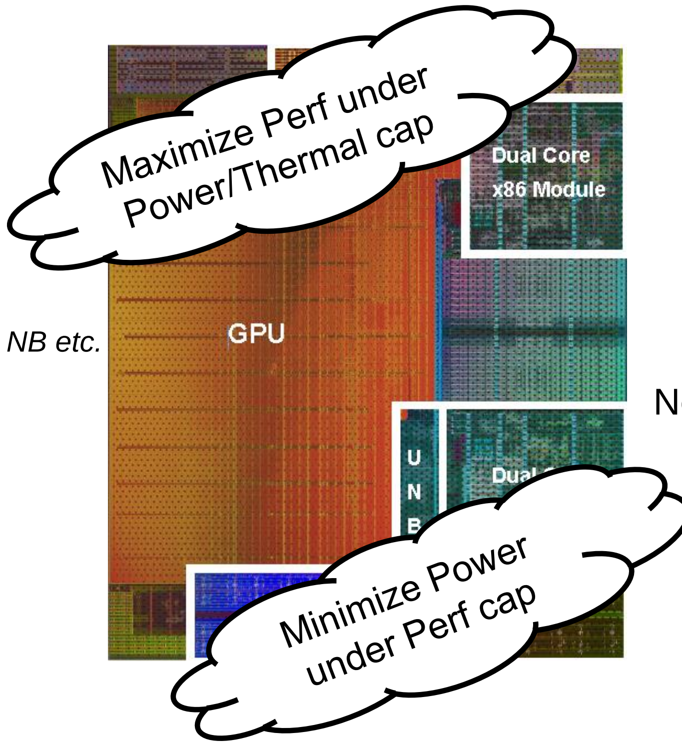
Time-varying Workload

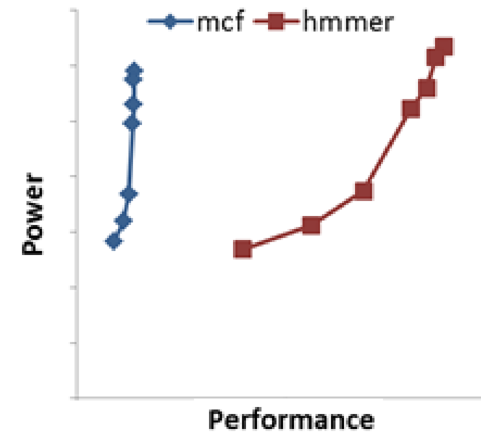
Performance-aware Power Management

- Optimum node-level power efficiency is a complex function of SOC configuration, workload characteristics, programming model, and node-level objective/constraints

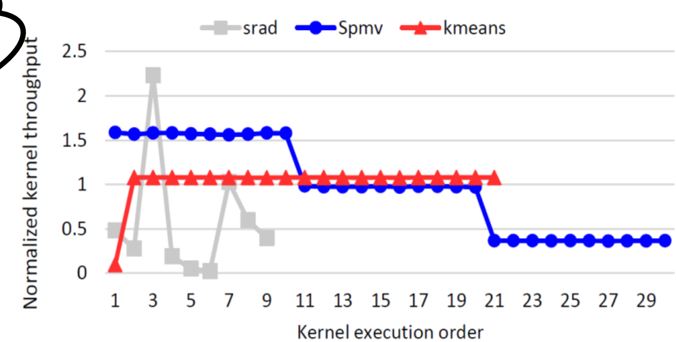








Non-linear power vs. performance curves

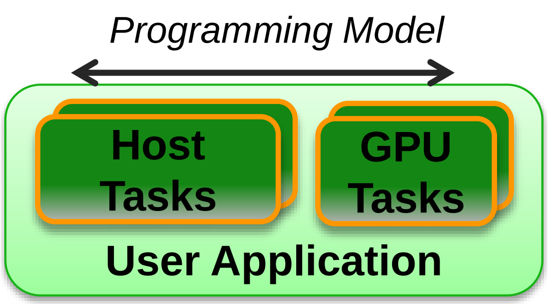


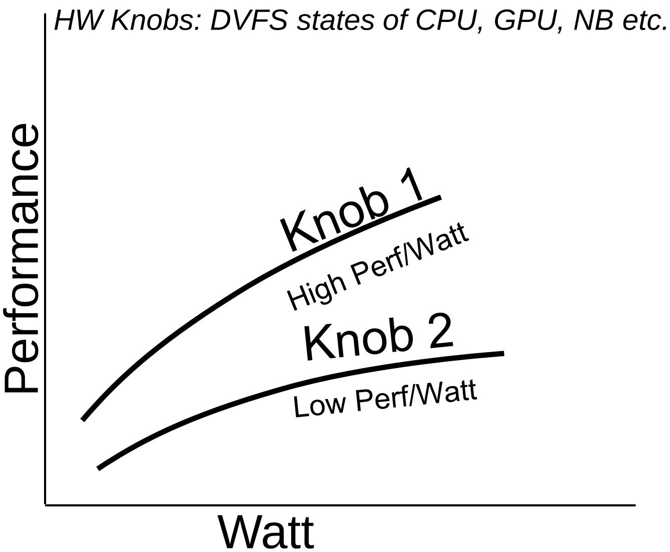
Time-varying Workload

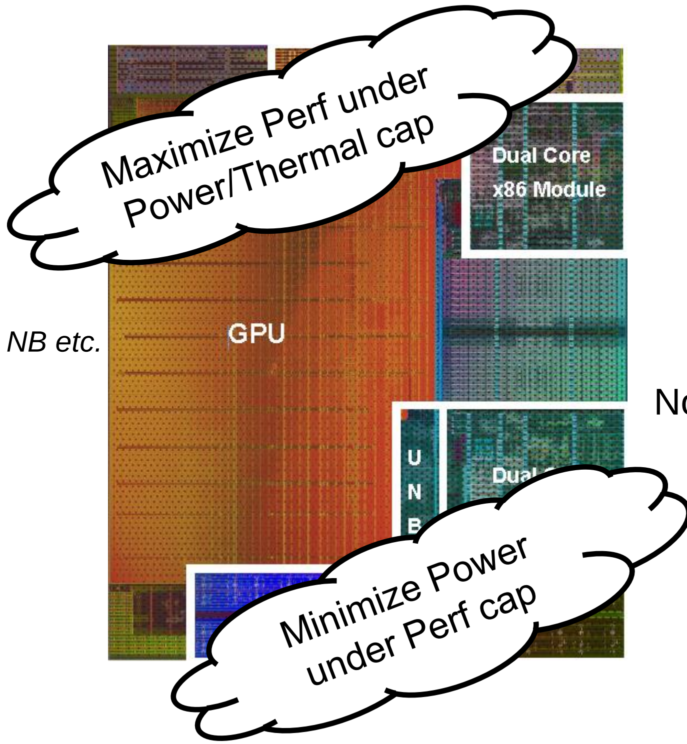
Performance-aware Power Management

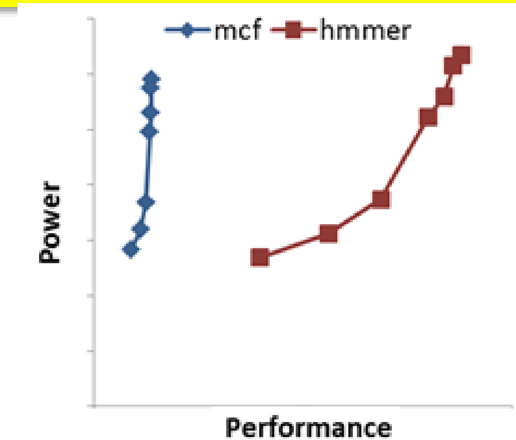
- Optimum node-level power efficiency is a complex function of SOC configuration, workload characteristics, programming model, and node-level objective/constraints

Reduce energy of the GPGPU kernel phase while performing better than a target

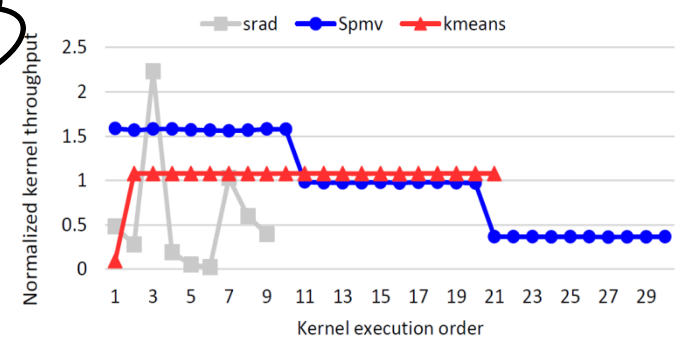








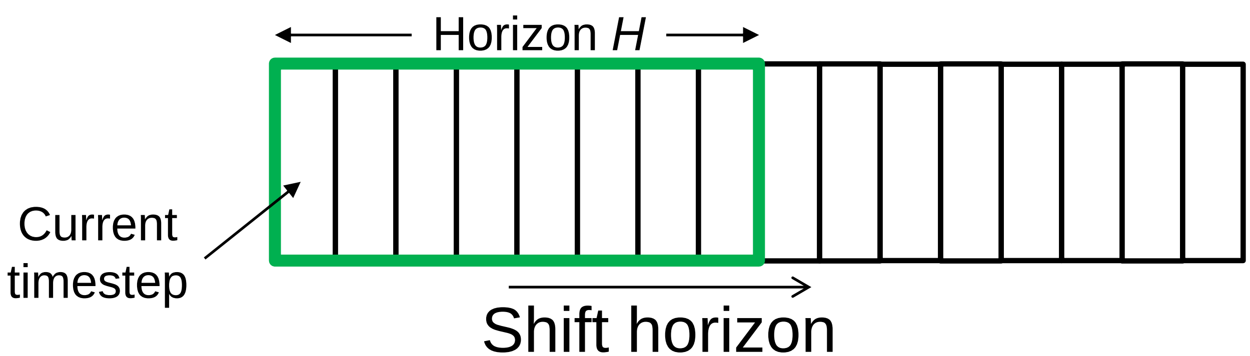
Non-linear power vs. performance curves



Time-varying Workload

Model Predictive Control (MPC)

- MPC *looks into the future* to determine the best configuration for the current optimization time step



- Though effective in many domains, overheads far too high for short timescales of dynamic power management

- <u>Our approach</u>: An approximation of MPC that *dramatically improves* GPGPU energy-efficiency with *orders of magnitude lower* overhead

Dynamic GPGPU Power Management

- CPU and GPU consume significant power in servers
- Previous approaches to dynamic power management are locally predictive and ignore future kernel behavior
  - Performance loss or wasted energy

Dynamic GPGPU Power Management

- CPU and GPU consume significant power in servers
- Previous approaches to dynamic power management are locally predictive and ignore future kernel behavior
  - Performance loss or wasted energy

Model Predictive Control

Proactively looks ahead into the future

Dynamic GPGPU Power Management

- CPU and GPU consume significant power in servers
- Previous approaches to dynamic power management are locally predictive and ignore future kernel behavior
  - Performance loss or wasted energy

Model Predictive Control

Proactively looks ahead into the future

- Applying MPC is challenging for short timescales of dynamic power management

Dynamic GPGPU Power Management

- CPU and GPU consume significant power in servers
- Previous approaches to dynamic power management are locally predictive and ignore future kernel behavior
  - Performance loss or wasted energy

Model Predictive Control

Proactively looks ahead into the future

- Applying MPC is challenging for short timescales of dynamic power management
- **Goal:** Approximations to MPC that save GPGPU energy within the timescales of a typical server operation without degrading performance

Dynamic GPGPU Power Management

- **Goal:** Reduce GPGPU energy within the timescales of a typical server operation without degrading performance

- **Computationally Intensive**
  - NP-Hard
  - Challenging for short timescales of dynamic power management

- **Idea:** Improve energy efficiency by looking into future phases
  - Model Predictive Control (MPC)
  - Dynamically vary computation to limit performance overhead

Traditional Energy Management

- **Static**
  - Predefined set of decisions
- **Reactive**
  - Act after sensing a change in behavior
- **Locally Predictive**
  - Predict the immediate behavior

Traditional Energy Management

- **Static**
  - Predefined set of decisions
- **Reactive**
  - Act after sensing a change in behavior
- **Locally Predictive**
  - Predict the immediate behavior

Performance loss or wasted energy

Traditional Energy Management

- **Static**
  - Predefined set of decisions
- **Reactive**
  - Act after sensing a change in behavior
- **Locally Predictive**
  - Predict the immediate behavior

Performance loss or wasted energy

Proactive Energy Management

Adapt from past and look-ahead into the future

Motivating Future Awareness

Motivating Future Awareness

- **Baseline**
  - AMD Turbo Core

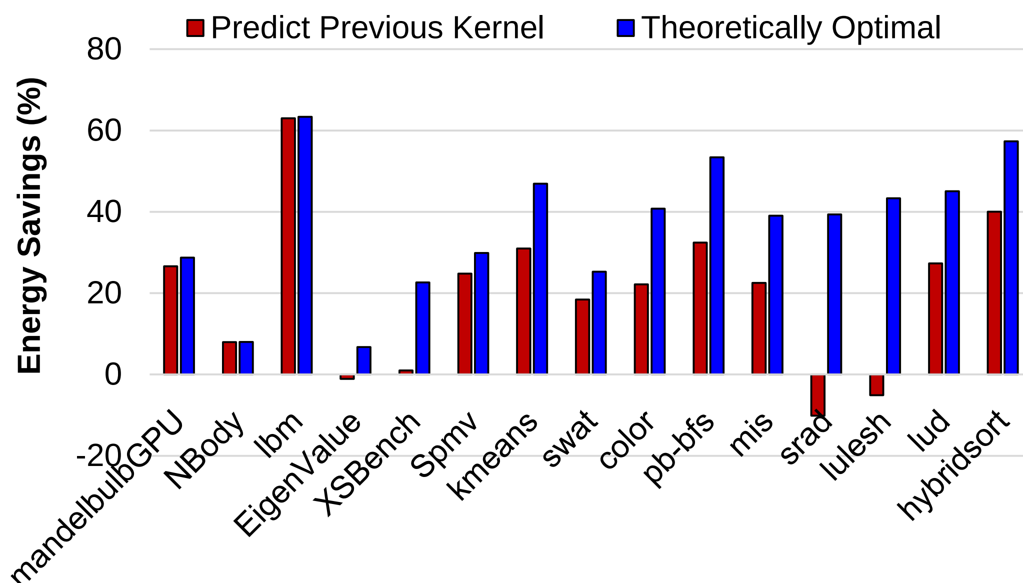
- **Hardware Knobs**
  - DVFS states, GPU CUs

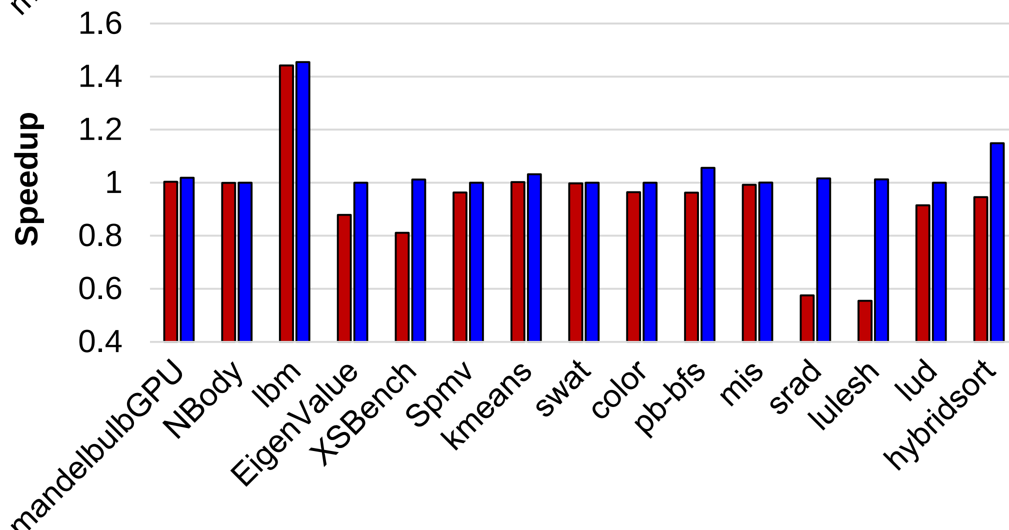
Motivating Future Awareness

- **Baseline**
  - AMD Turbo Core
- **Hardware Knobs**
  - DVFS states, GPU CUs
- **Predict Previous Kernel (PPK):**
  - Assume last kernel repeats
  - State-of-the-art: Harmonia ISCA’15, McLaughlin et al. ASBD’14
- **Theoretically Optimal (TO):**
  - Perfect knowledge of future
  - Impractical

Motivating Future Awareness

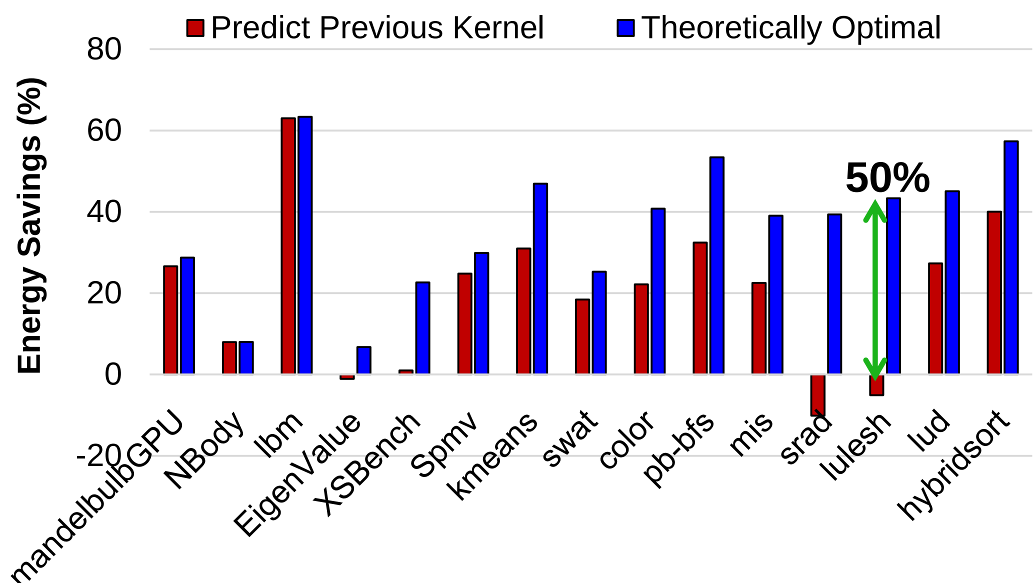
- **Baseline**
  - AMD Turbo Core
- **Hardware Knobs**
  - DVFS states, GPU CUs
- **Predict Previous Kernel (PPK):**
  - Assume last kernel repeats
  - State-of-the-art: Harmonia ISCA'15, McLaughlin et al. ASBD'14
- **Theoretically Optimal (TO):**
  - Perfect knowledge of future
  - Impractical

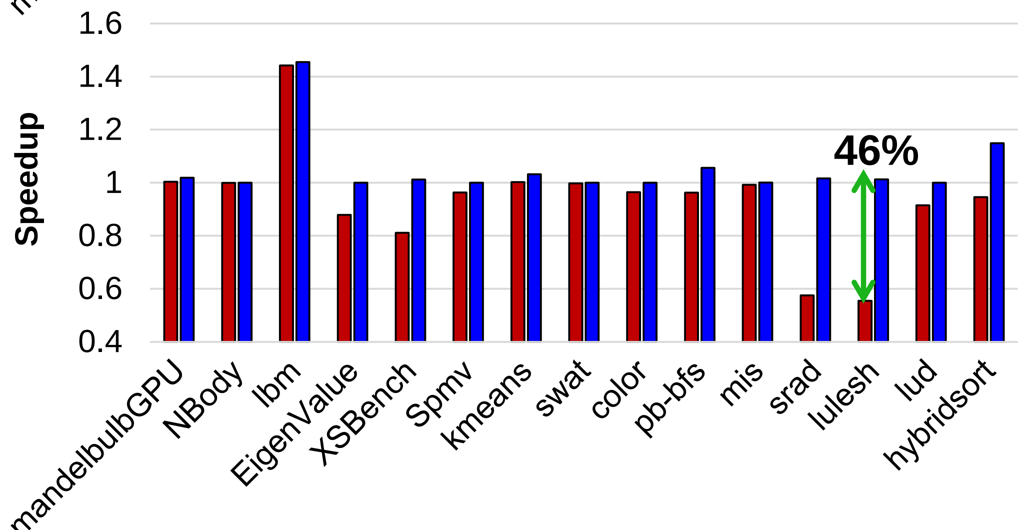




Motivating Future Awareness

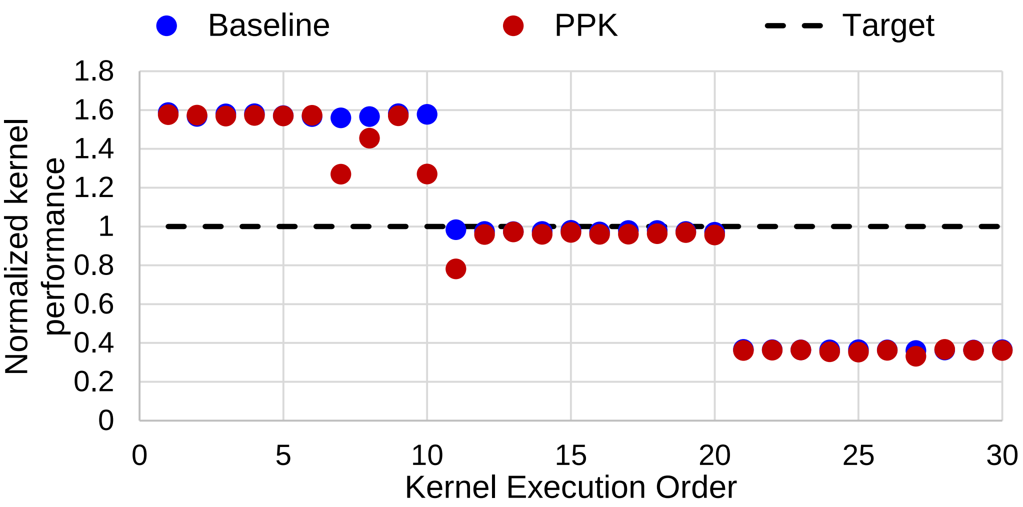
- **Baseline**
  - AMD Turbo Core
- **Hardware Knobs**
  - DVFS states, GPU CUs
- **Predict Previous Kernel (PPK):**
  - Assume last kernel repeats
  - State-of-the-art: Harmonia ISCA'15, McLaughlin et al. ASBD'14
- **Theoretically Optimal (TO):**
  - Perfect knowledge of future
  - Impractical

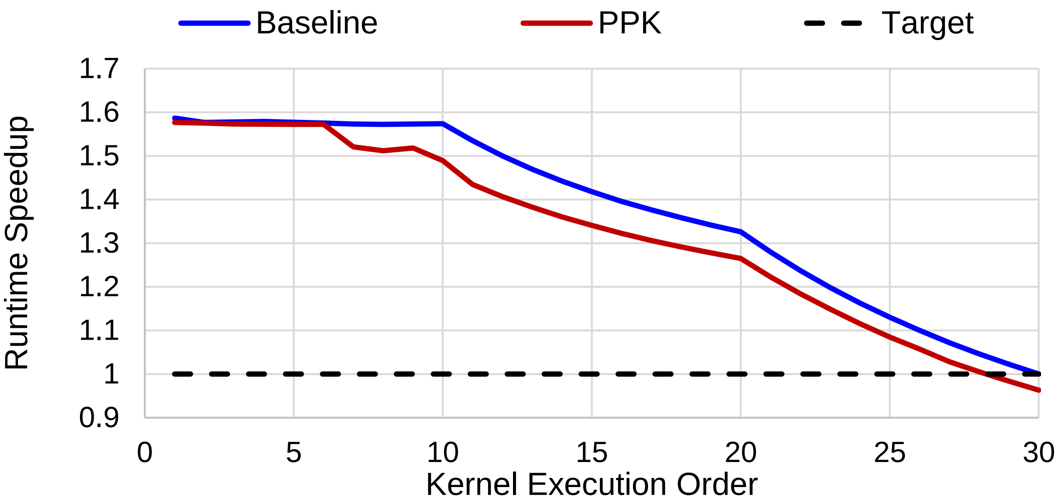




Ramifications of Ignoring Future

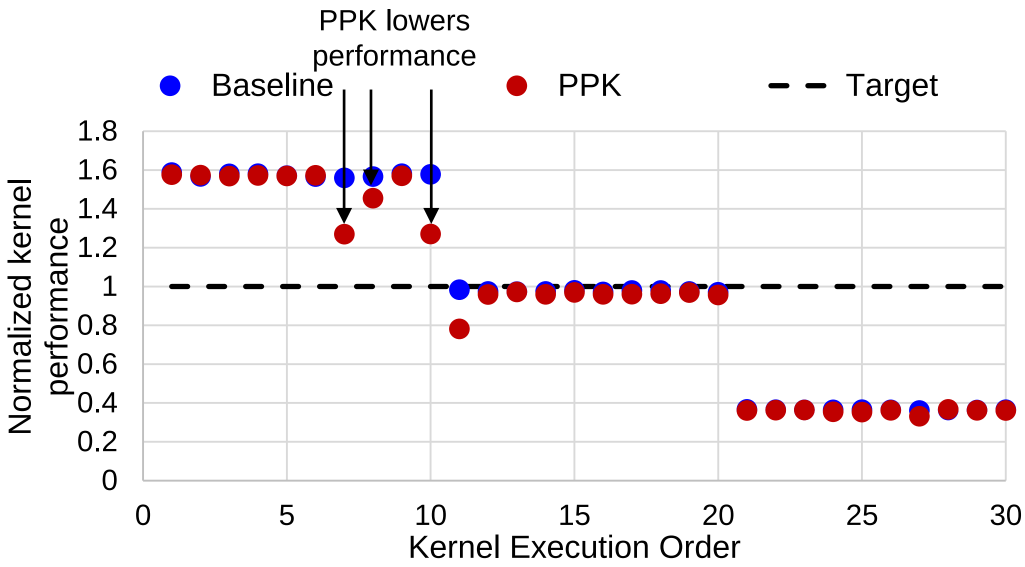
- Spmv

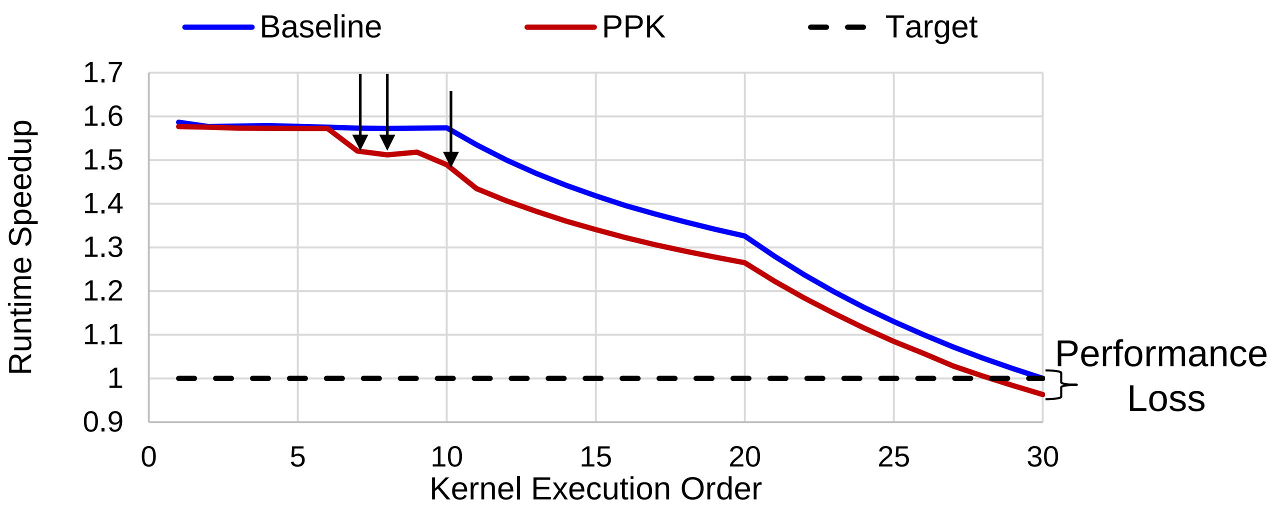




Ramifications of Ignoring Future

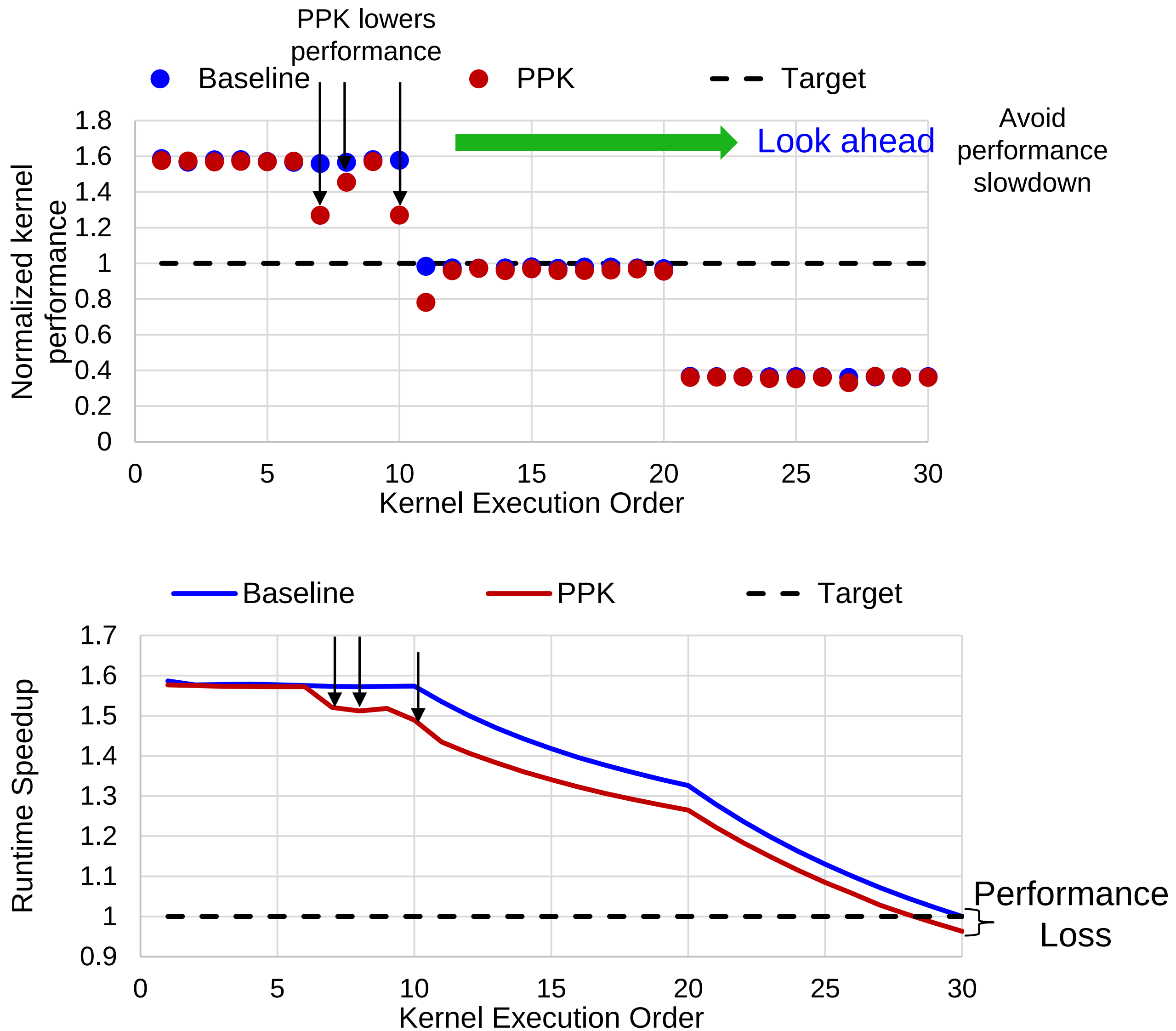
- Spmv





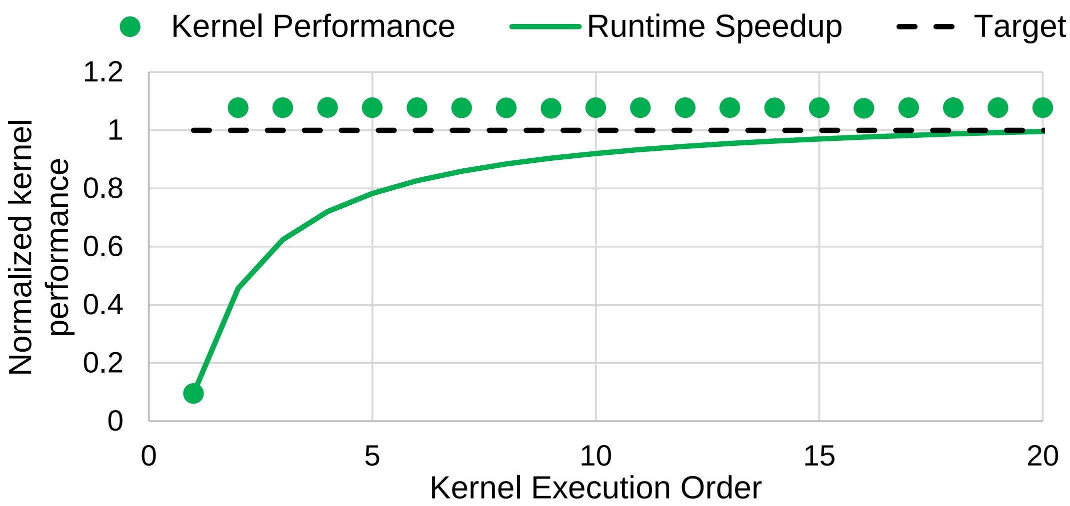
Ramifications of Ignoring Future

- Spmv



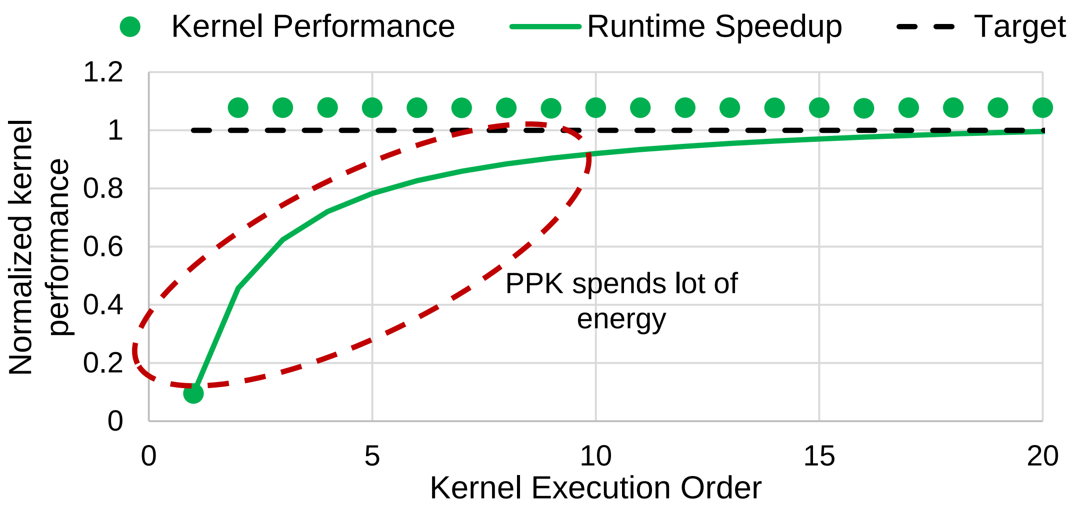
Ramifications of Ignoring Future

- Kmeans



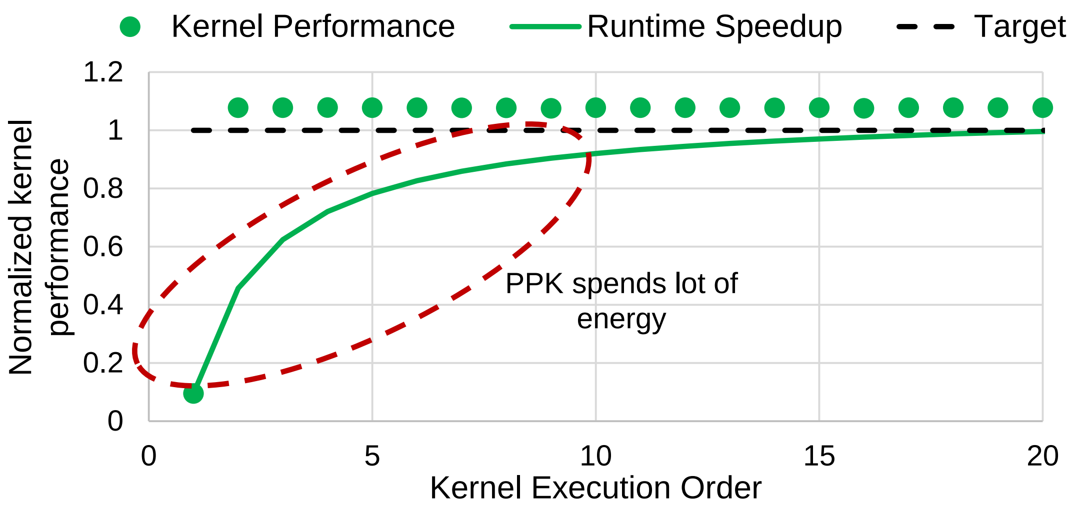
Ramifications of Ignoring Future

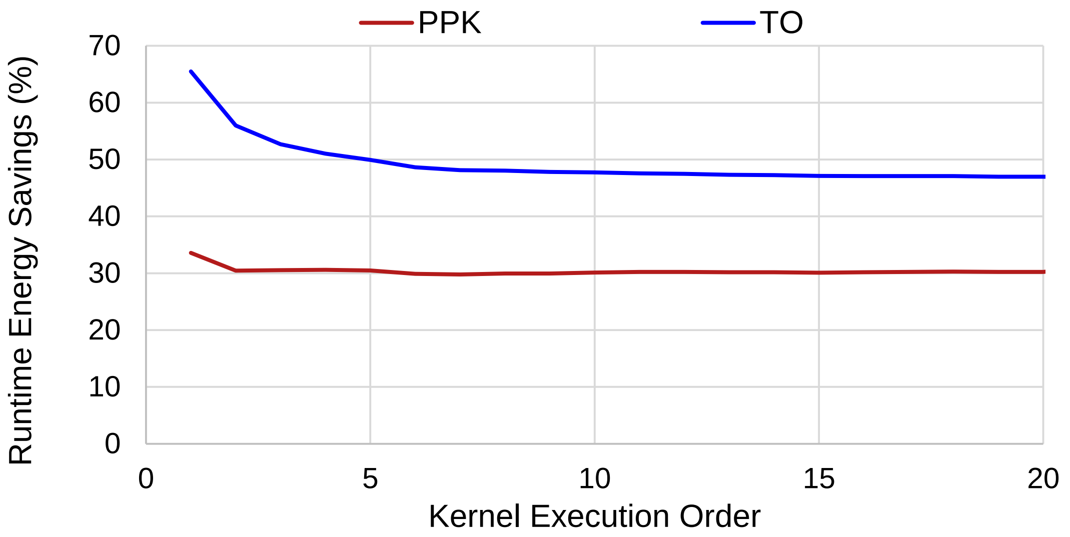
- Kmeans



Ramifications of Ignoring Future

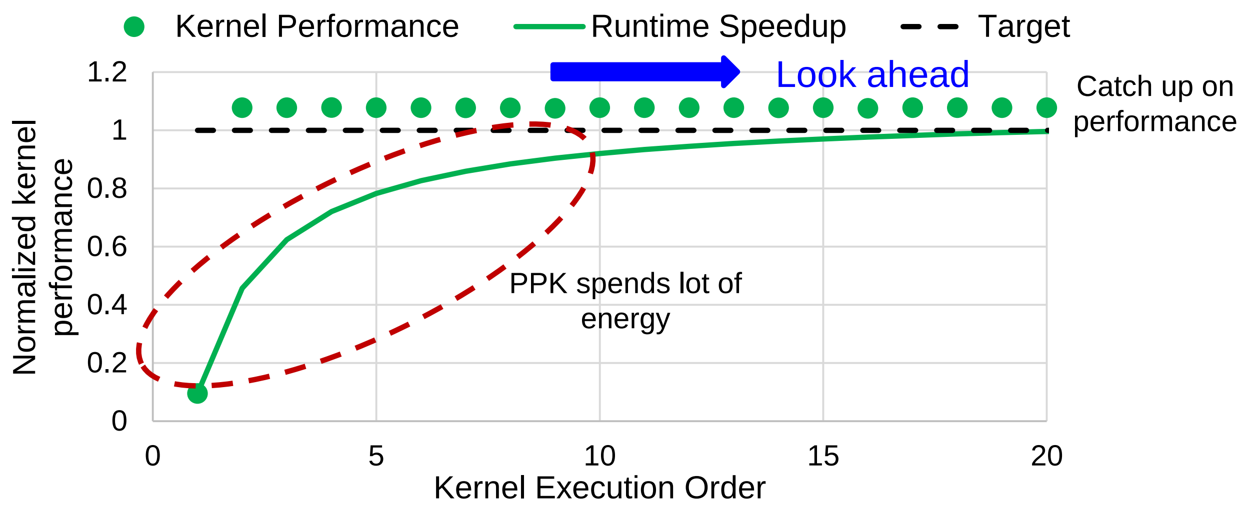
- Kmeans

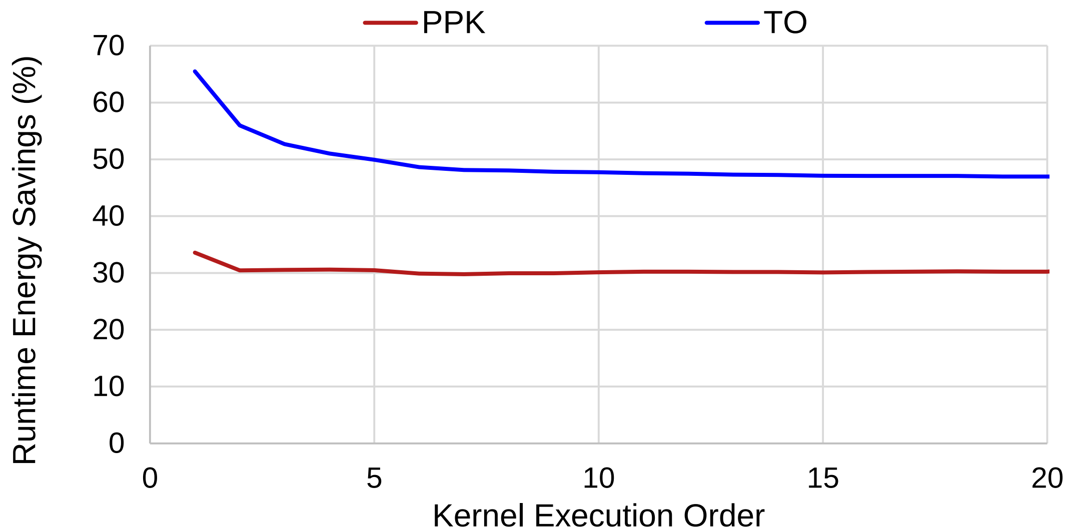




Ramifications of Ignoring Future

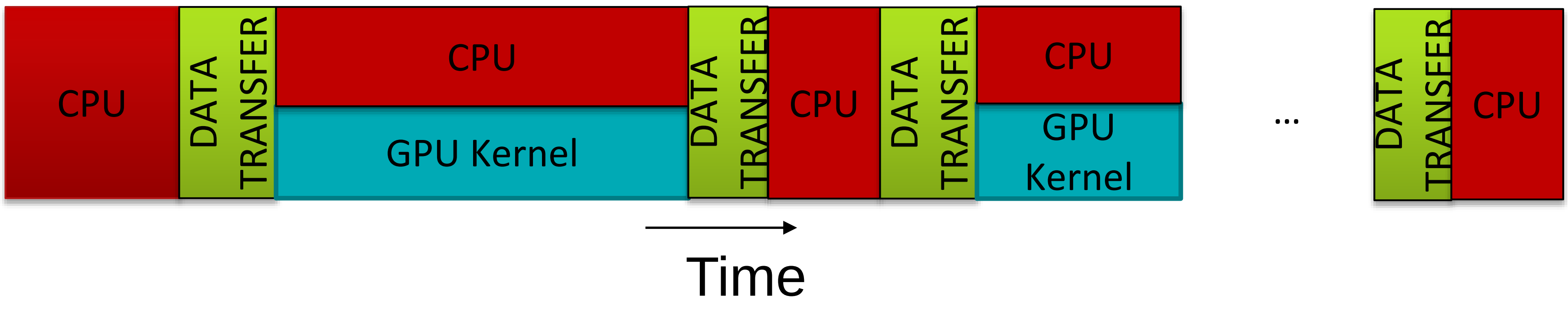
- Kmeans





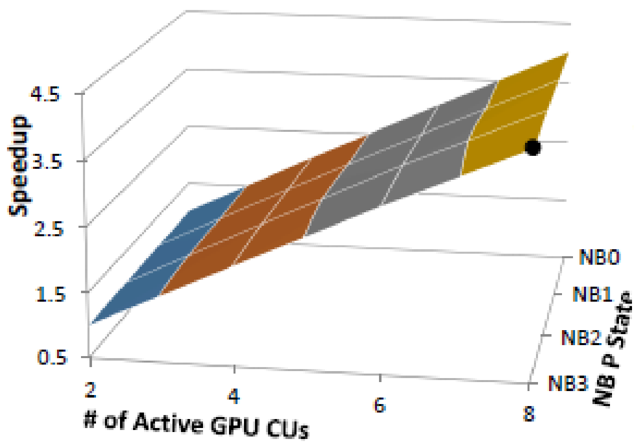
Background

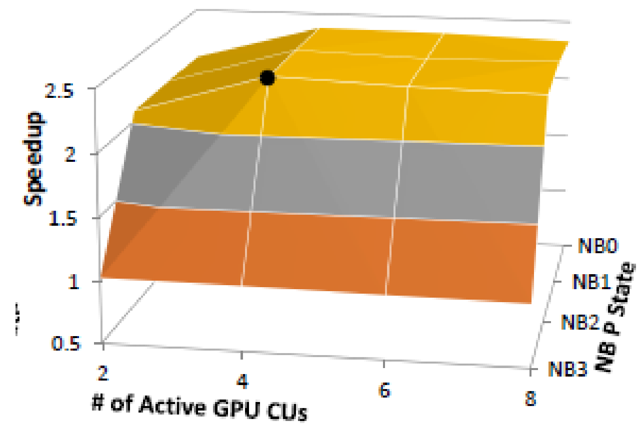
- Typical GPGPU application phase



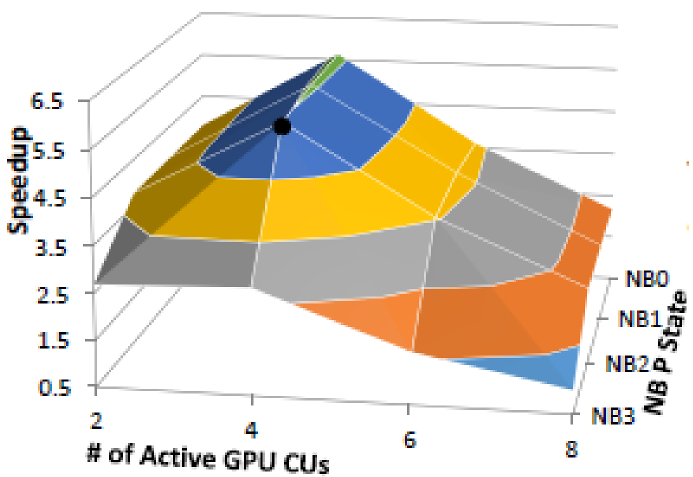
Kernel Performance Scaling

Energy-optimal configuration differ across kernels

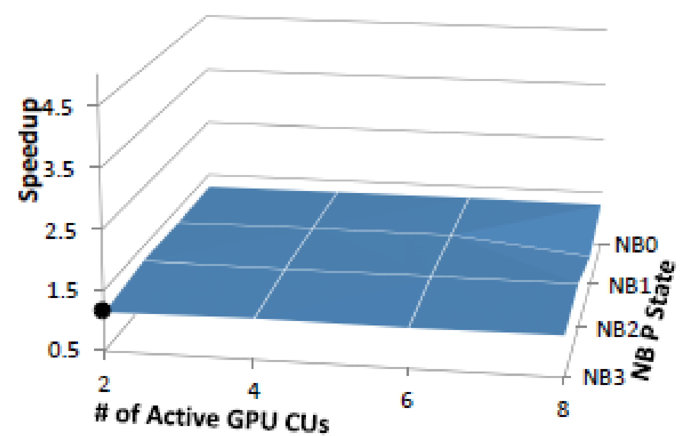




(b) Memory-bound: *readGlobalMemoryCoalsced*



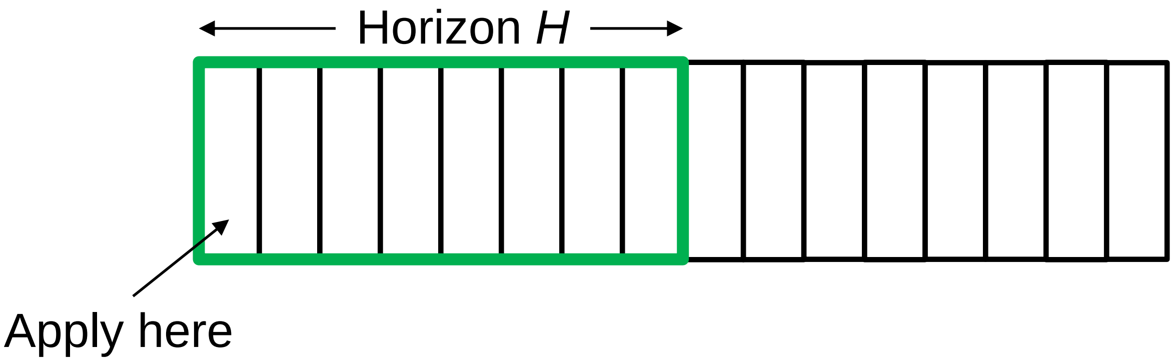
(c) Peak: *writeCandidates*



(d) Unscalable: *astar*

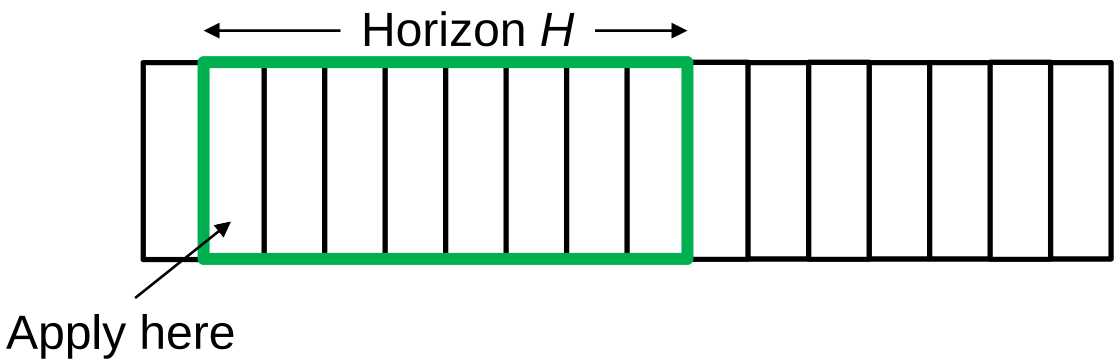
Model Predictive Control (MPC)

- General Idea



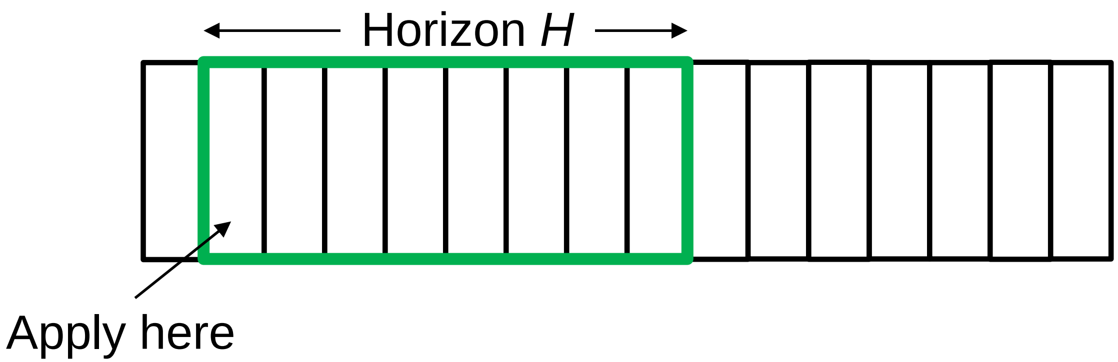
Model Predictive Control (MPC)

- General Idea



Model Predictive Control (MPC)

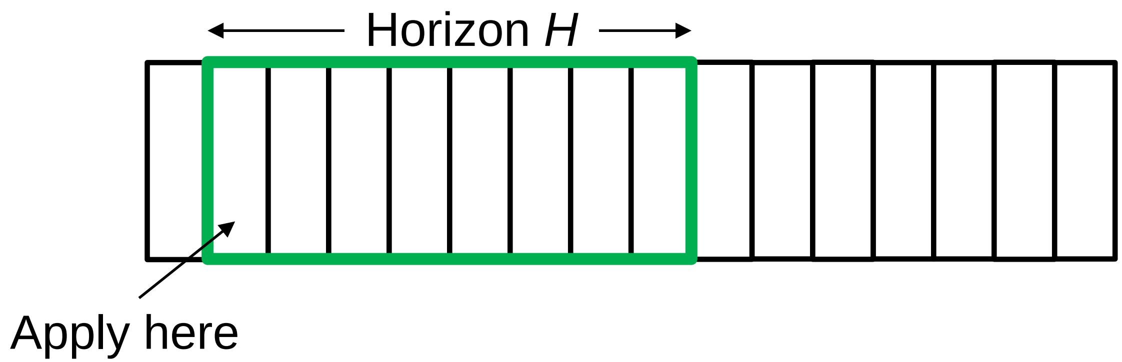
- General Idea



- MPC Components
  - Accurate system model
  - Future input forecast
  - Optimization

Model Predictive Control (MPC)

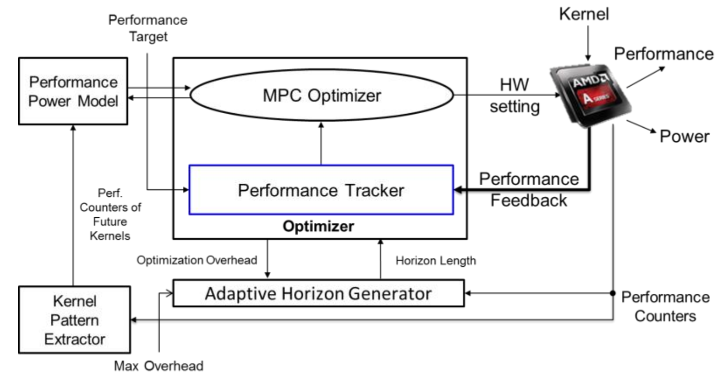
- General Idea



- MPC Components
  - Accurate system model → Power and performance prediction model
  - Future input forecast → Kernel pattern extractor
  - Optimization → Greedy and heuristic based optimizer

Feedback-based Performance Tracker

- Switch the optimization goal based on past performance and the target



Performance met previously?

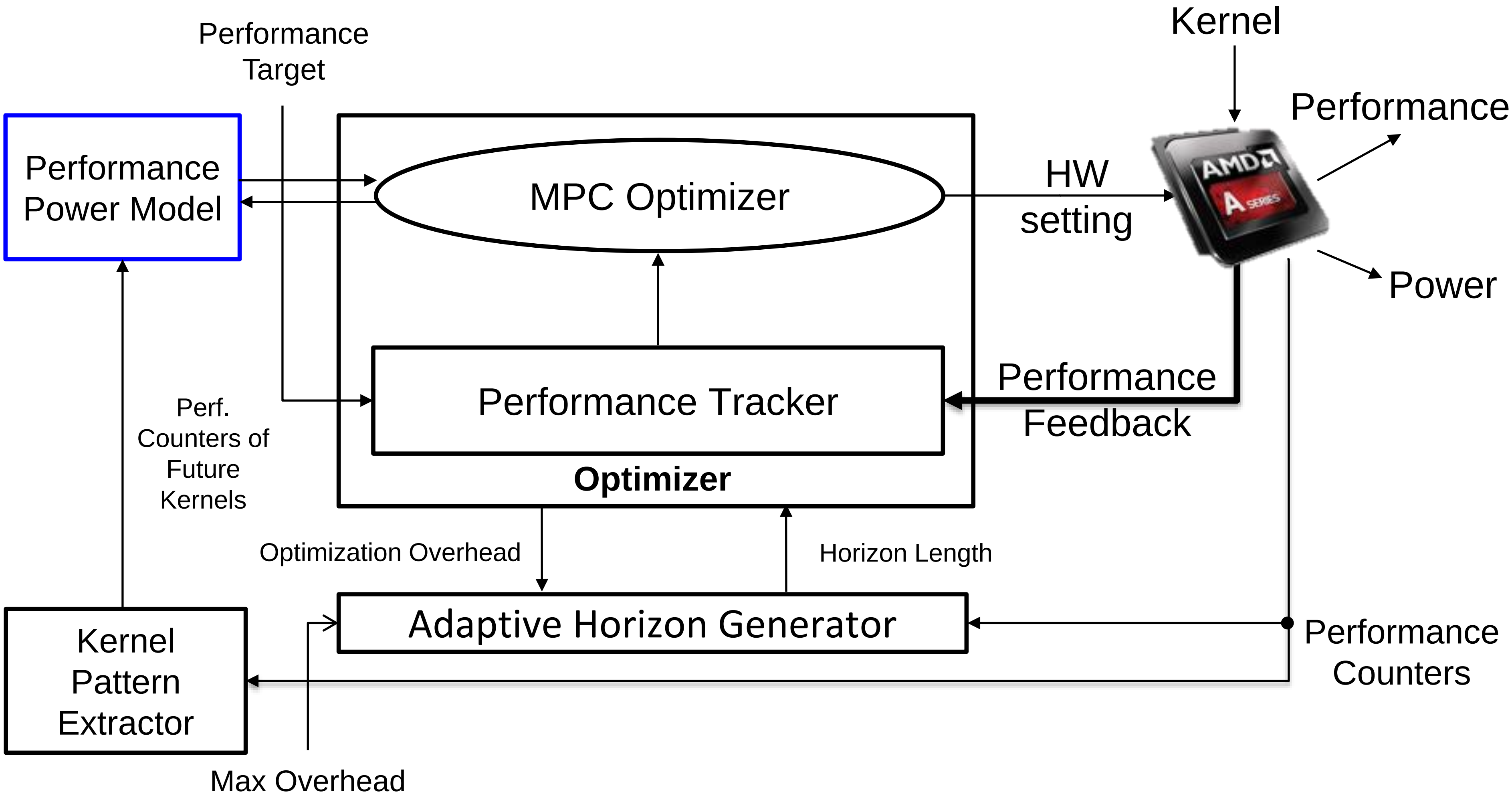
No

Relax optimization
Spend energy
↓
Reduce performance loss

Yes

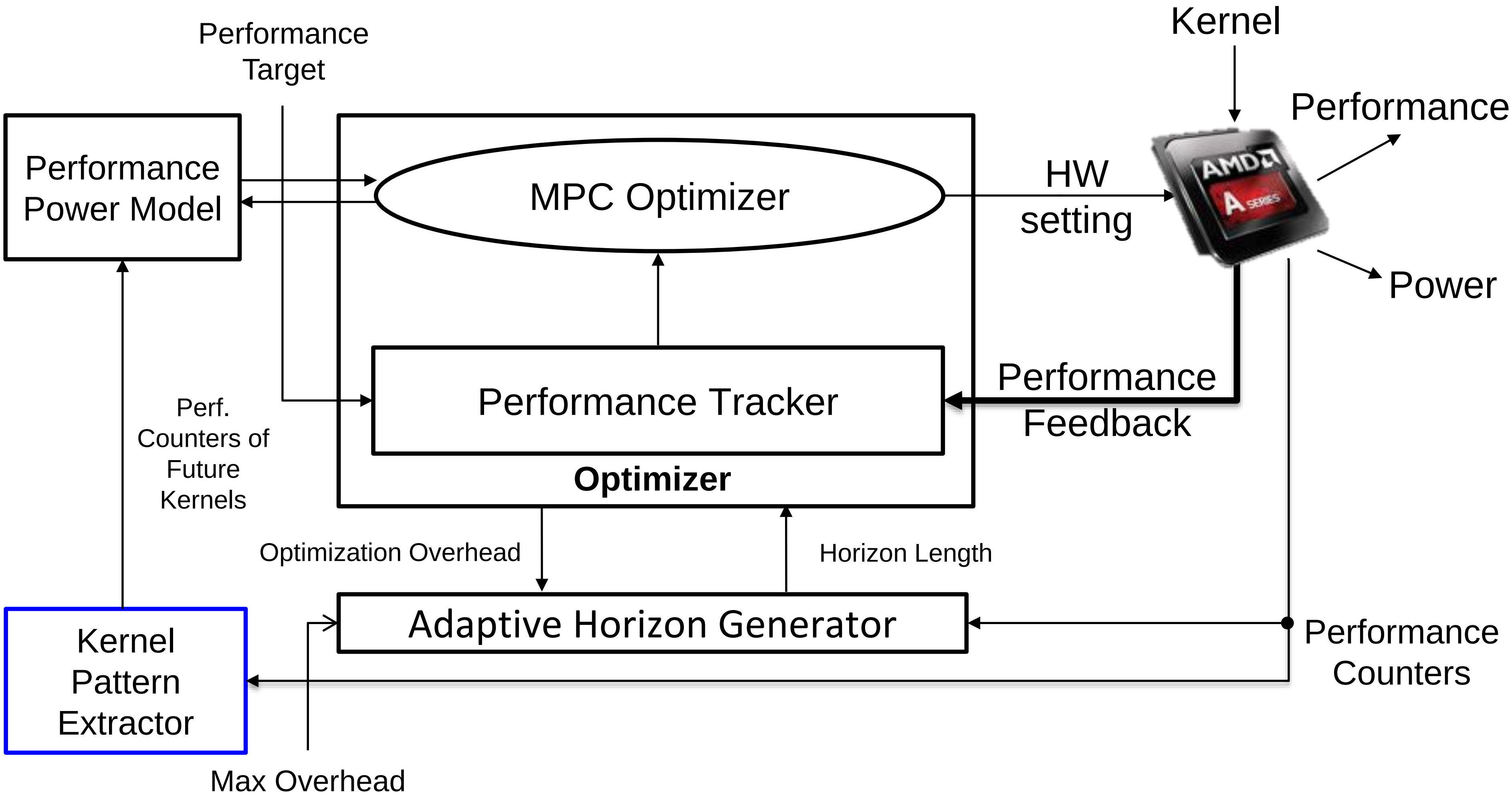
Optimize Aggressively
Reduce energy
↓
Avoid performance loss

Performance Power Model



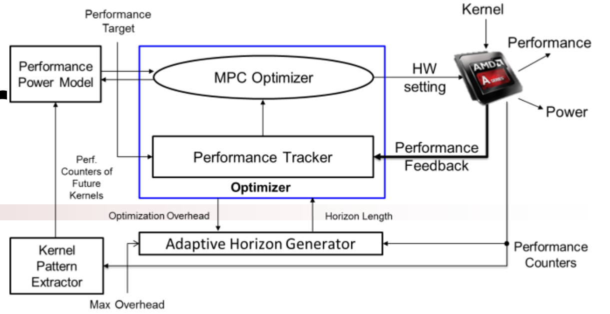
- Trained offline using Random Forest Learning Algorithm
- Estimates performance and power for any HW configuration

Kernel Pattern Extractor



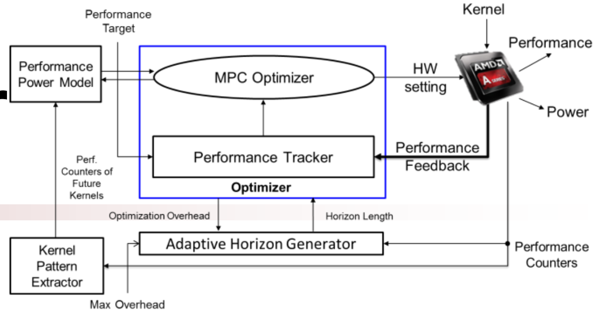
- Extracts kernel execution pattern upon the first encounter
- Stores the performance counters of dissimilar kernels and retunes it

MPC Optimizer



- Greedy Hill Climbing optimization
  - Using the predictor, select the HW knob with highest energy sensitivity
  - Search for low energy configuration using hill climbing

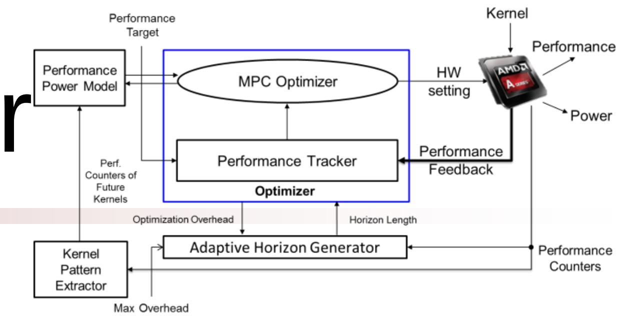
MPC Optimizer



- Greedy Hill Climbing optimization
  - Using the predictor, select the HW knob with highest energy sensitivity
  - Search for low energy configuration using hill climbing

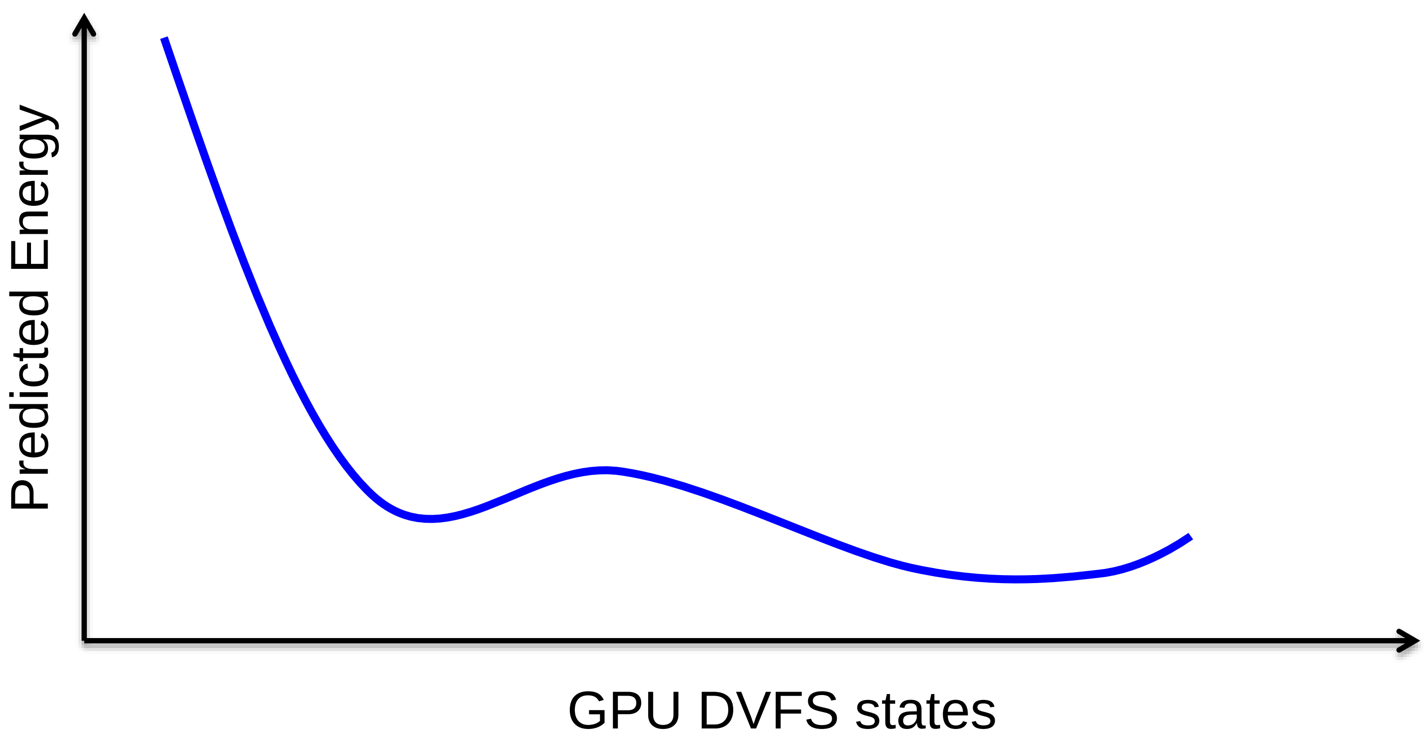
CPU DVFS	NB DVFS	GPU DVFS	GPU CU

MPC Optimizer

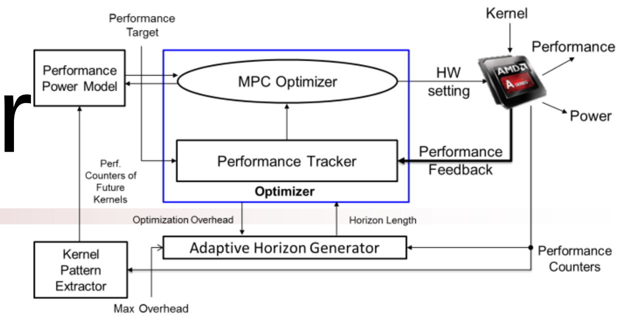


- Greedy Hill Climbing optimization
  - Using the predictor, select the HW knob with highest energy sensitivity
  - Search for low energy configuration using hill climbing

CPU DVFS | NB DVFS | GPU DVFS | GPU CU

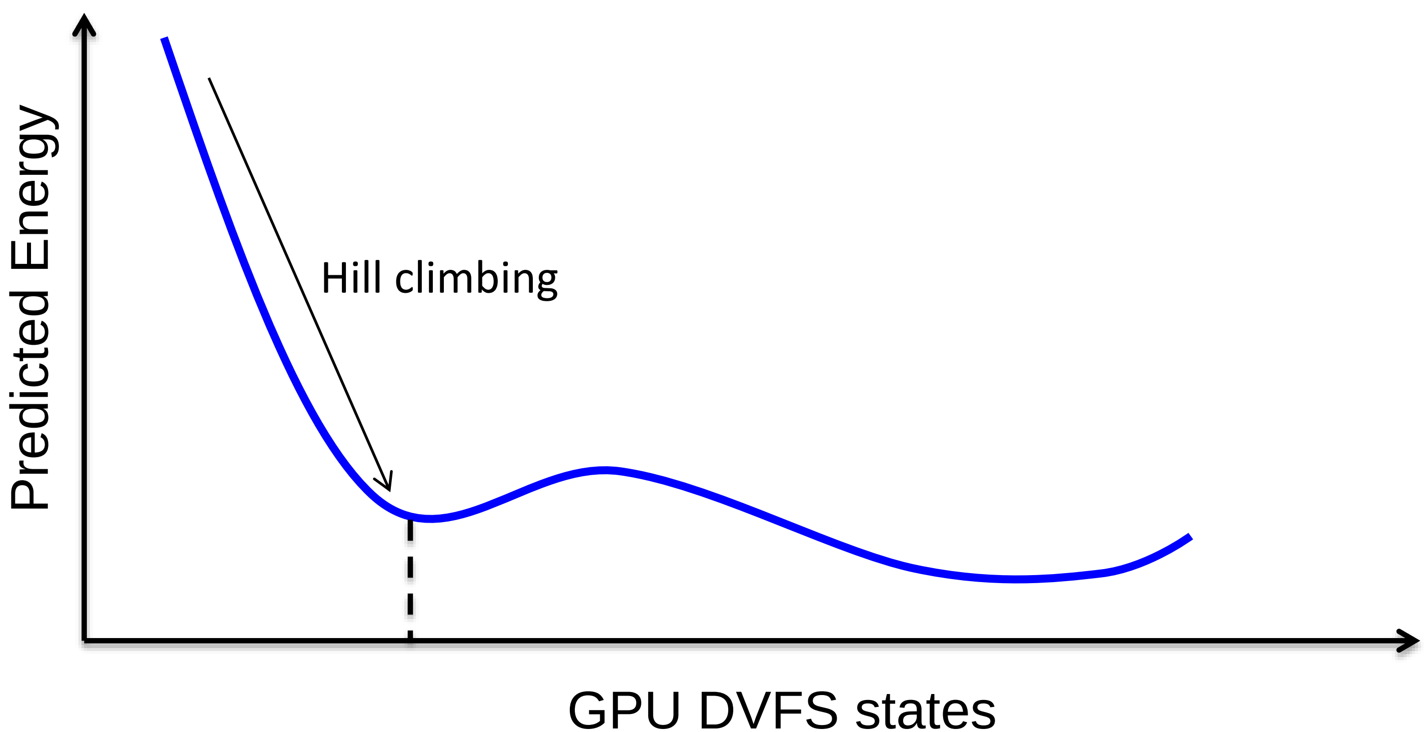


MPC Optimizer

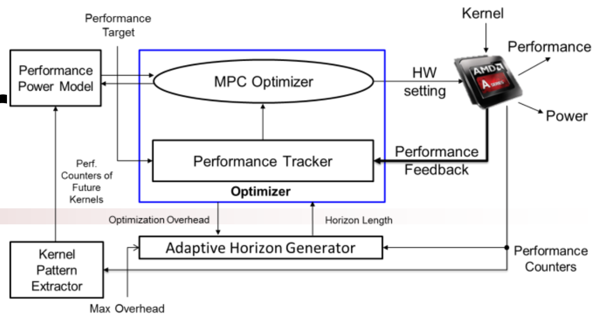


- Greedy Hill Climbing optimization
  - Using the predictor, select the HW knob with highest energy sensitivity
  - Search for low energy configuration using hill climbing

CPU DVFS | NB DVFS | **GPU DVFS** | GPU CU

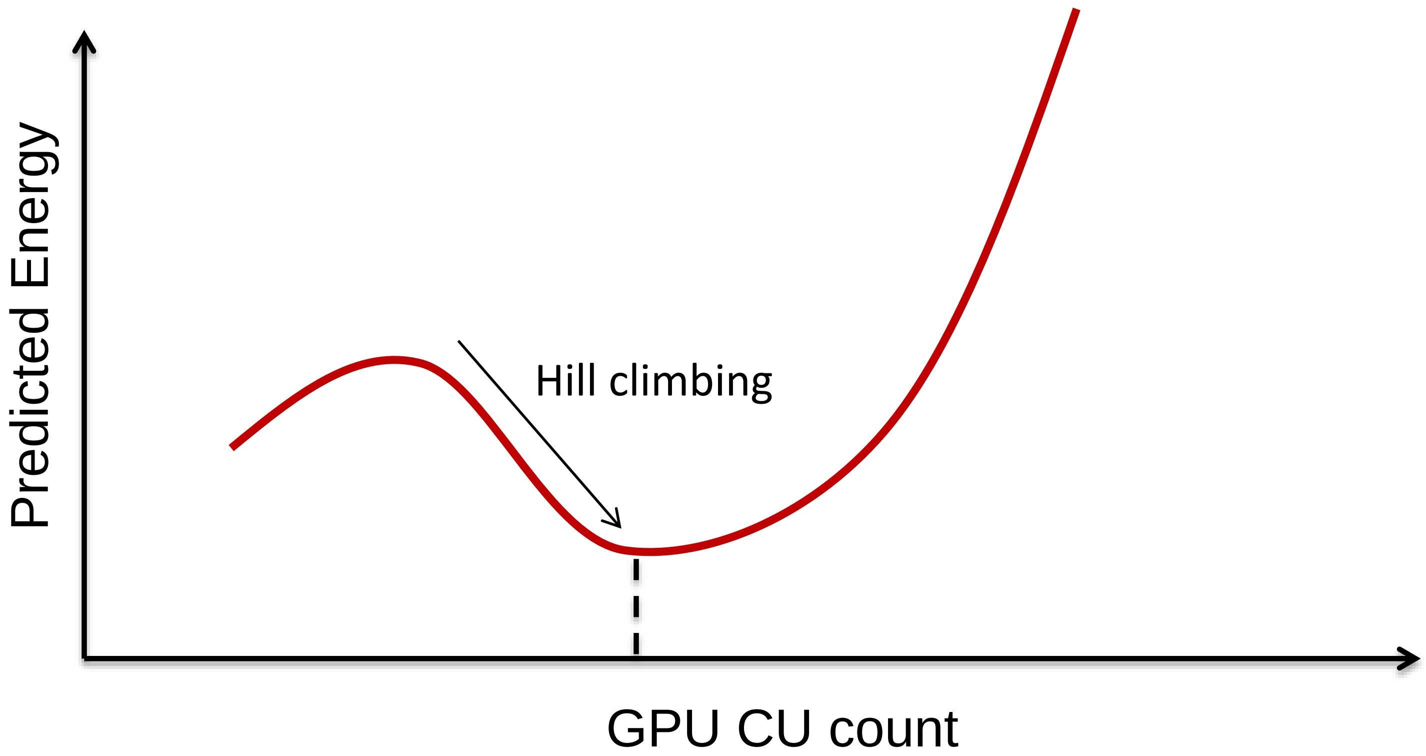


MPC Optimizer

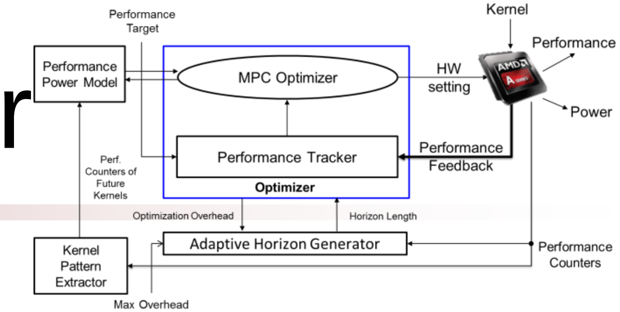


- Greedy Hill Climbing optimization
  - Using the predictor, select the HW knob with highest energy sensitivity
  - Search for low energy configuration using hill climbing

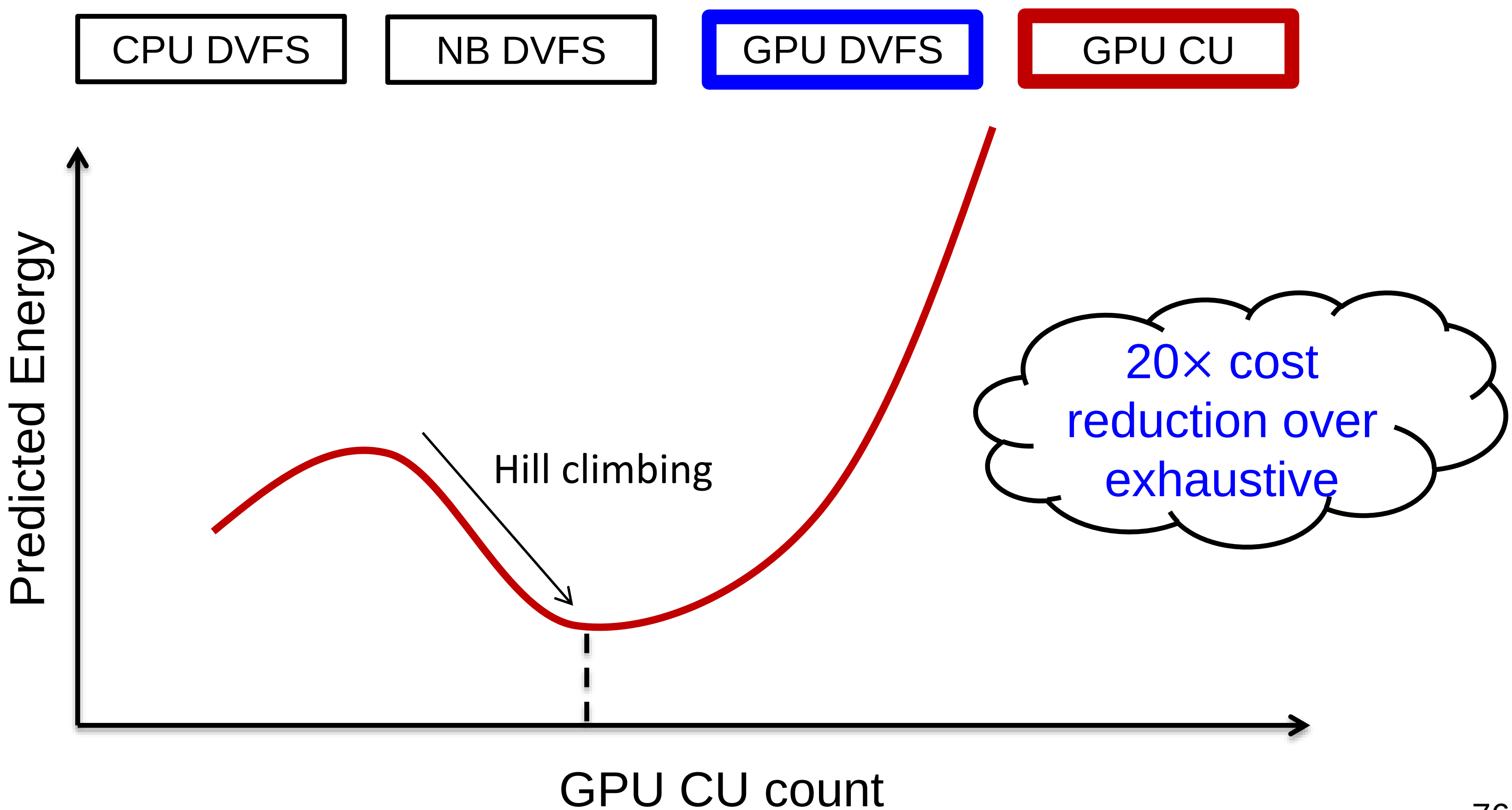
CPU DVFS | NB DVFS | GPU DVFS | GPU CU



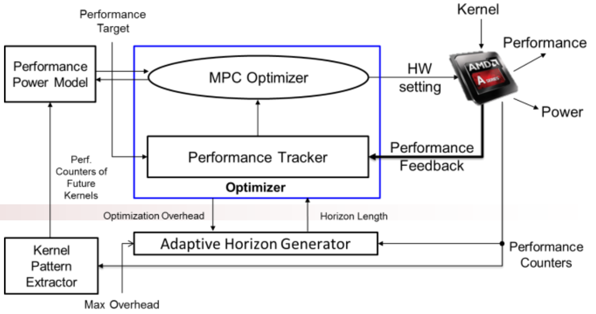
MPC Optimizer



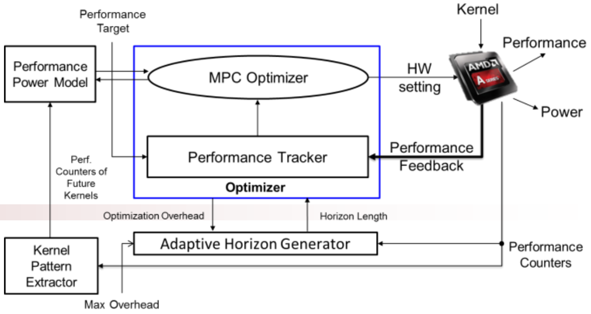
- Greedy Hill Climbing optimization
  - Using the predictor, select the HW knob with highest energy sensitivity
  - Search for low energy configuration using hill climbing



MPC Optimizer

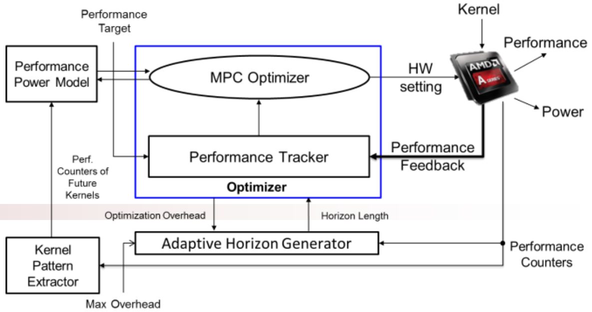


MPC Optimizer



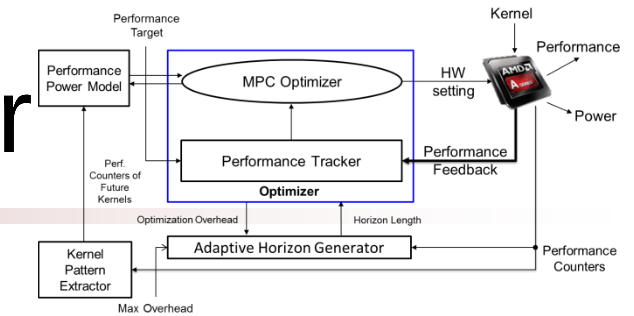
- Greedy Hill Climbing optimization
  - Select the HW knob with highest energy sensitivity and search for low energy configuration using hill climbing

MPC Optimizer

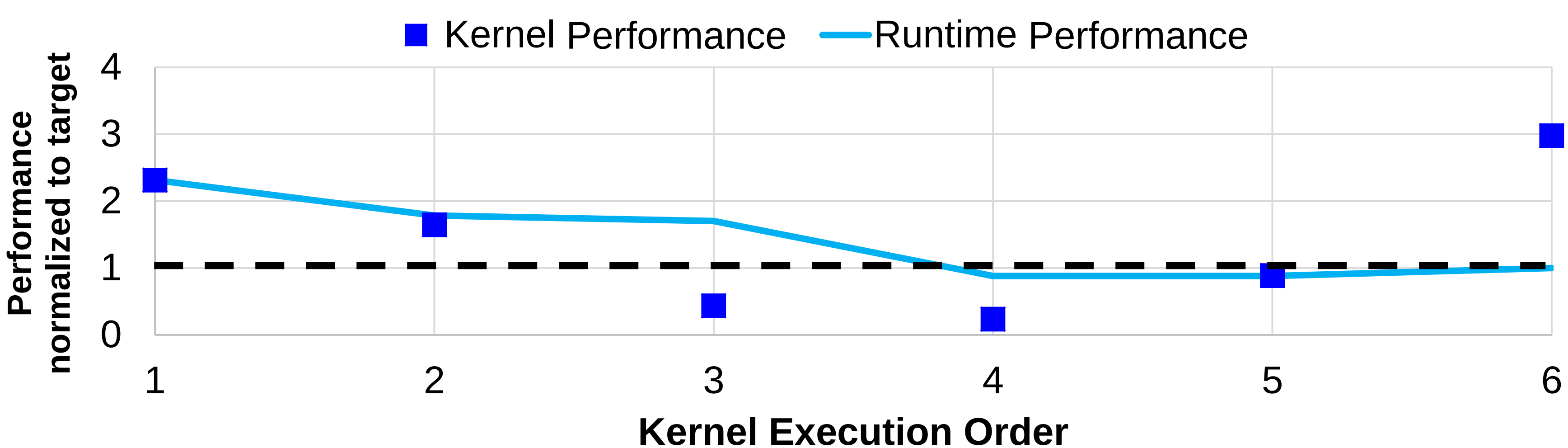


- Greedy Hill Climbing optimization
  - Select the HW knob with highest energy sensitivity and search for low energy configuration using hill climbing
- MPC Search Heuristic
  - Determine a static order requiring no backtracking
  - General Idea
    - High to low performance (e.g. Spmv): Optimize low performing kernels first
    - Low to high performance (e.g. Kmeans): Optimize high performing kernels first
  - Search cost becomes polynomial

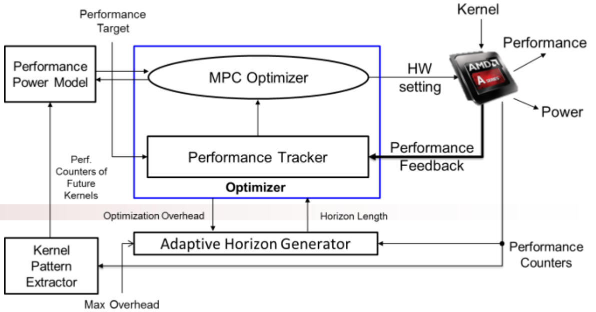
MPC Optimizer



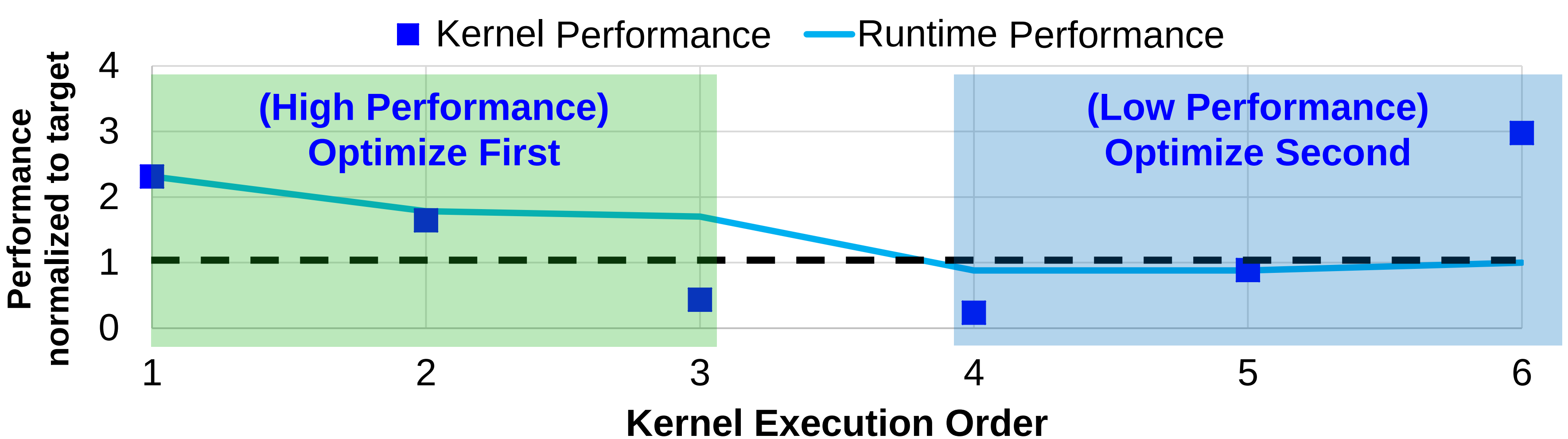
- Greedy Hill Climbing optimization
  - Select the HW knob with highest energy sensitivity and search for low energy configuration using hill climbing
- MPC Search Heuristic
  - Determine a static order requiring no backtracking
  - General Idea
    - – High to low performance (e.g. Spmv): Optimize low performing kernels first
    - – Low to high performance (e.g. Kmeans): Optimize high performing kernels first
  - Search cost becomes polynomial



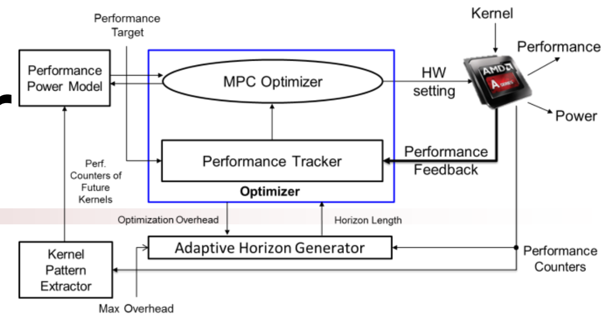
MPC Optimizer



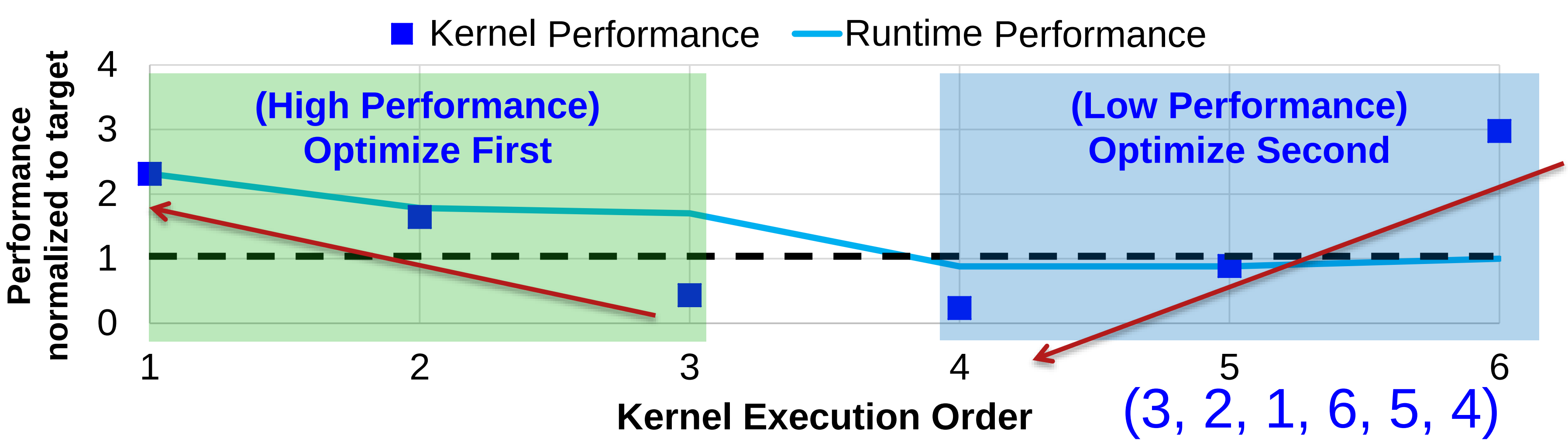
- Greedy Hill Climbing optimization
  - Select the HW knob with highest energy sensitivity and search for low energy configuration using hill climbing
- MPC Search Heuristic
  - Determine a static order requiring no backtracking
  - General Idea
    - High to low performance (e.g. Spmv): Optimize low performing kernels first
    - Low to high performance (e.g. Kmeans): Optimize high performing kernels first
  - Search cost becomes polynomial



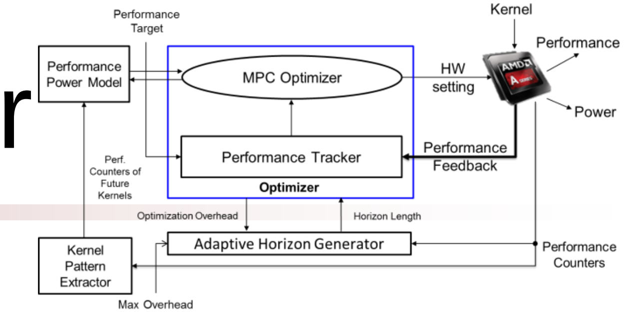
MPC Optimizer



- Greedy Hill Climbing optimization
  - Select the HW knob with highest energy sensitivity and search for low energy configuration using hill climbing
- MPC Search Heuristic
  - Determine a static order requiring no backtracking
  - General Idea
    - High to low performance (e.g. Spmv): Optimize low performing kernels first
    - Low to high performance (e.g. Kmeans): Optimize high performing kernels first
  - Search cost becomes polynomial

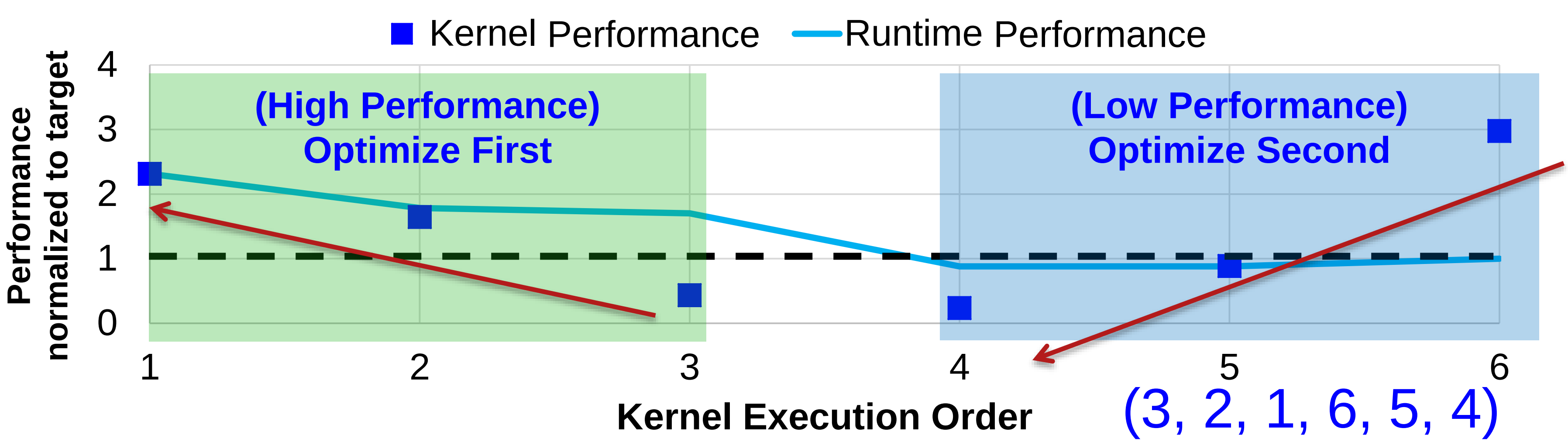


MPC Optimizer

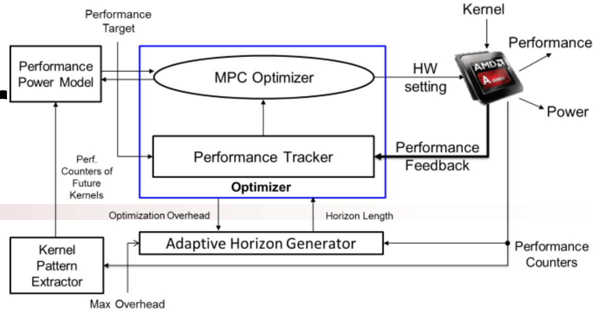


- Greedy Hill Climbing optimization
  - Select the HW knob with highest energy sensitivity and search for low energy configuration using hill climbing
- MPC Search Heuristic
  - Determine a static order requiring no backtracking
  - General Idea
    - High to low performance (e.g. Spmv): Optimize low performing kernels first
    - Low to high performance (e.g. Kmeans): Optimize high performing kernels first
  - Search cost becomes polynomial

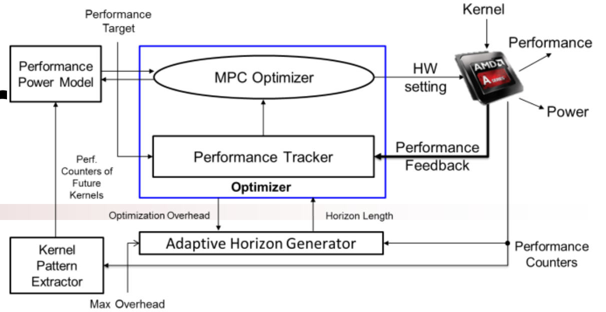
65× reduction in cost evaluation



MPC Optimizer

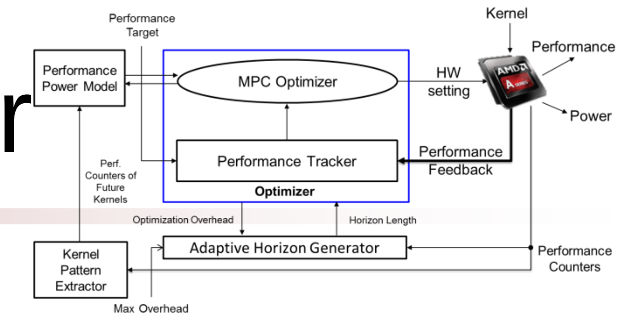


MPC Optimizer

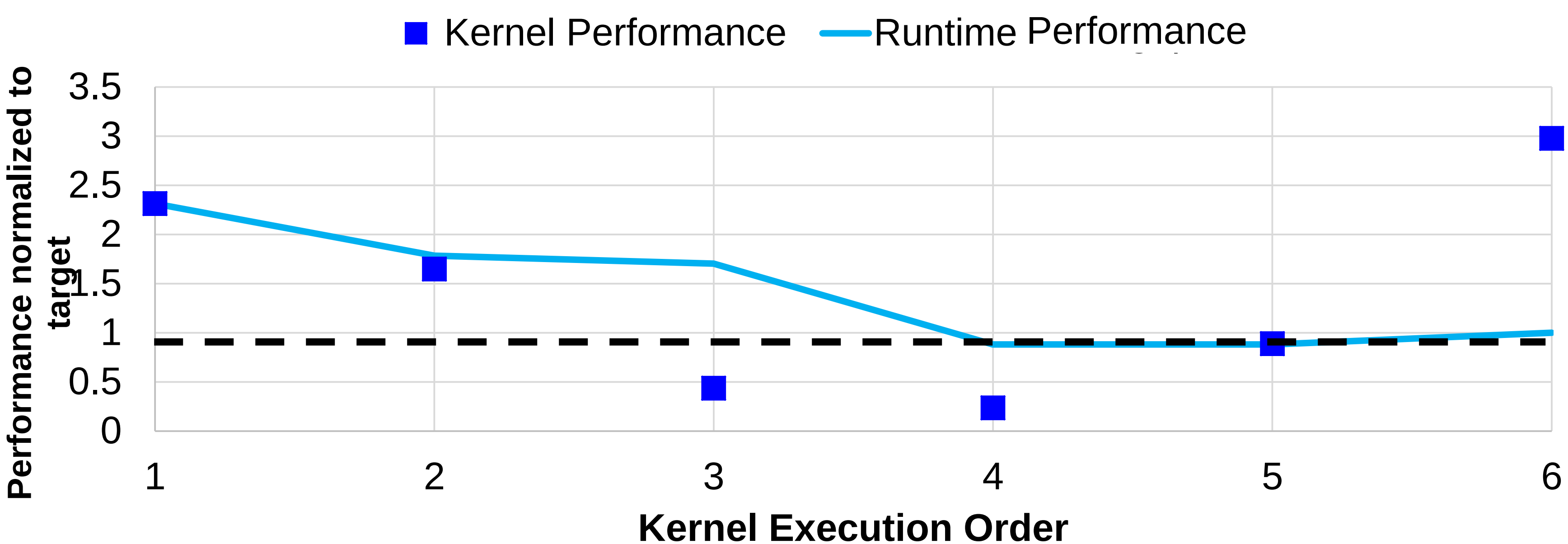


- MPC Search Heuristic
  - Determine a static order requiring no backtracking
  - General Idea
    - High to low performance (e.g. Spmv): Optimize low performing kernels first
    - Low to high performance (e.g. Kmeans): Optimize high performing kernels first
  - Search cost becomes polynomial

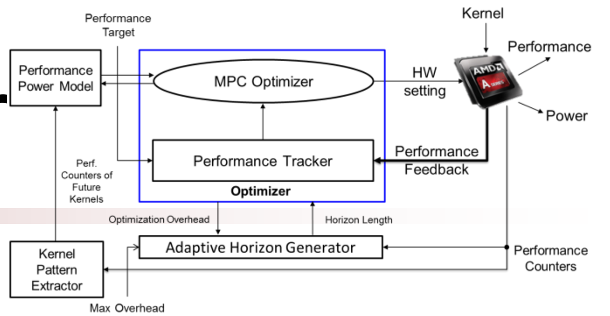
MPC Optimizer



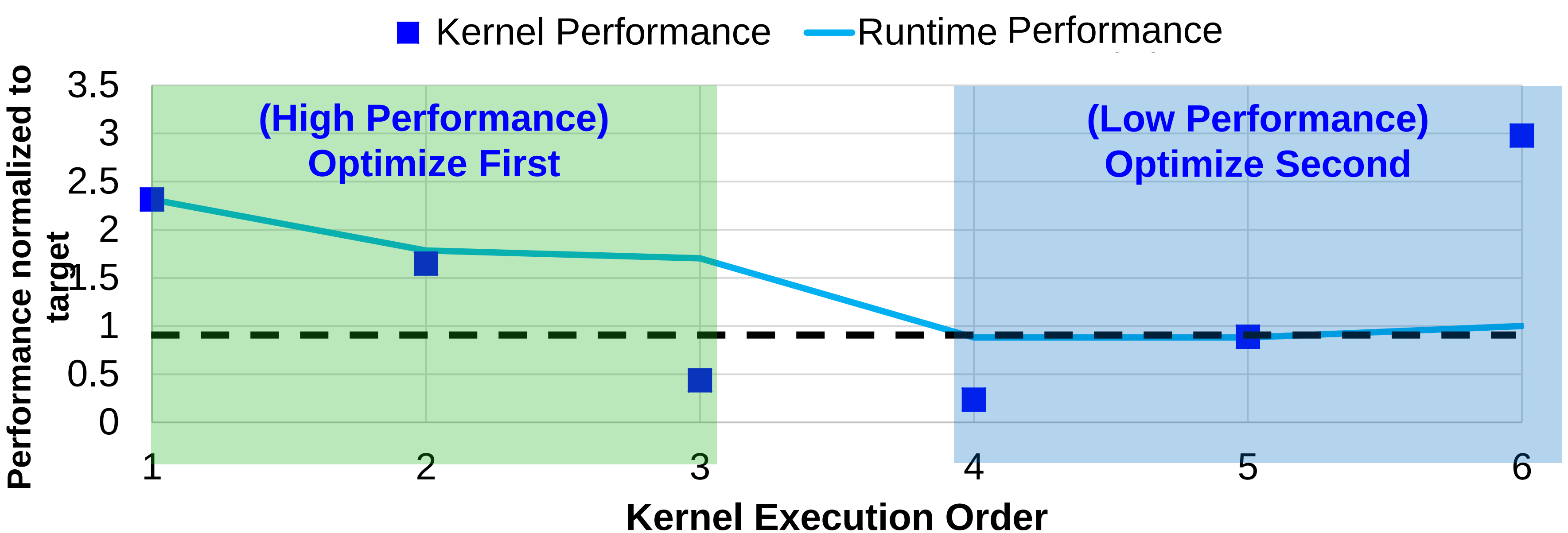
- MPC Search Heuristic
  - Determine a static order requiring no backtracking
  - General Idea
    - High to low performance (e.g. Spmv): Optimize low performing kernels first
    - Low to high performance (e.g. Kmeans): Optimize high performing kernels first
  - Search cost becomes polynomial



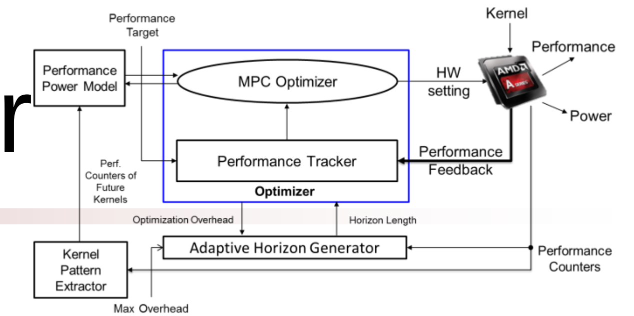
MPC Optimizer



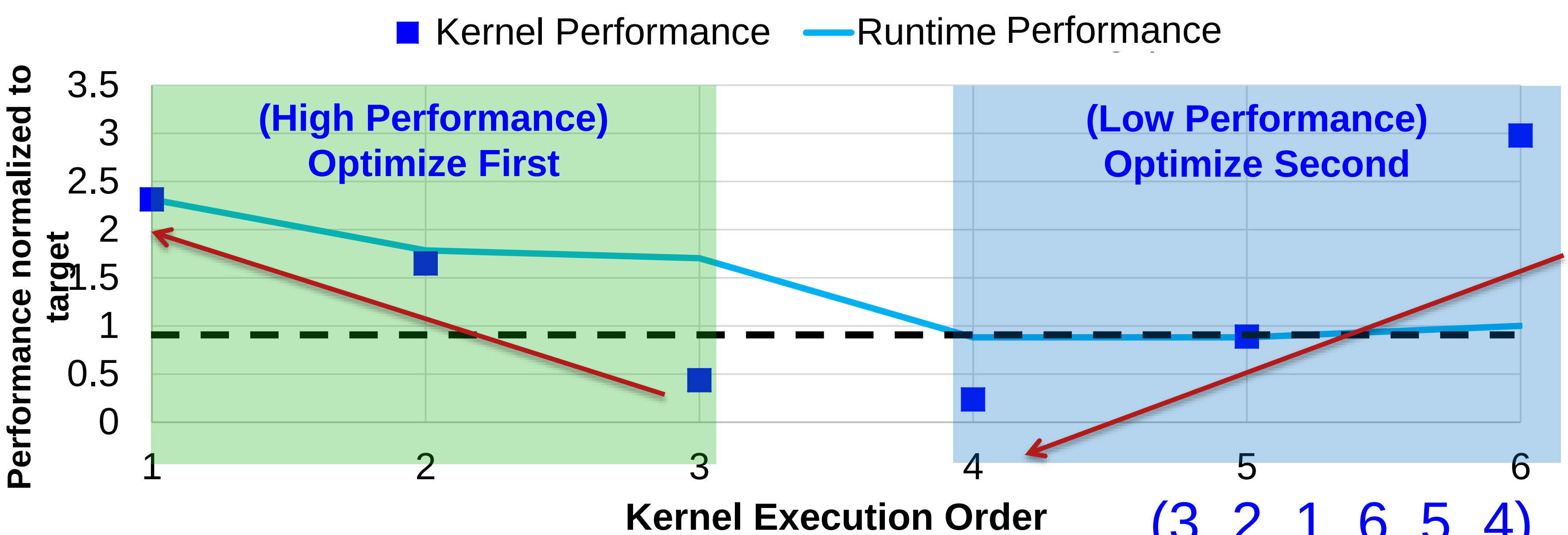
- MPC Search Heuristic
  - Determine a static order requiring no backtracking
  - General Idea
    - High to low performance (e.g. Spmv): Optimize low performing kernels first
    - Low to high performance (e.g. Kmeans): Optimize high performing kernels first
  - Search cost becomes polynomial



MPC Optimizer

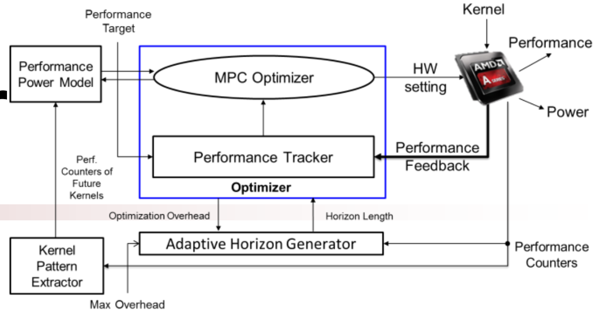


- MPC Search Heuristic
  - Determine a static order requiring no backtracking
  - General Idea
    - High to low performance (e.g. Spmv): Optimize low performing kernels first
    - Low to high performance (e.g. Kmeans): Optimize high performing kernels first
  - Search cost becomes polynomial



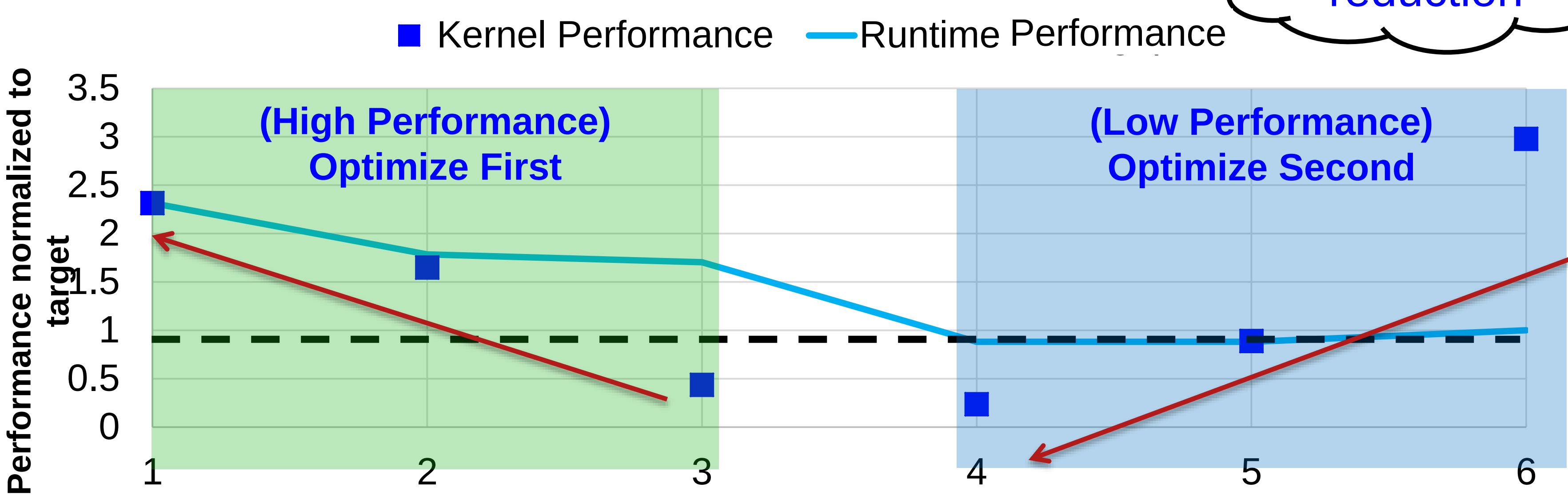
(3, 2, 1, 6, 5, 4)

MPC Optimizer



- MPC Search Heuristic
  - Determine a static order requiring no backtracking
  - General Idea
    - High to low performance (e.g. Spmv): Optimize low performing kernels first
    - Low to high performance (e.g. Kmeans): Optimize high performing kernels first
  - Search cost becomes polynomial

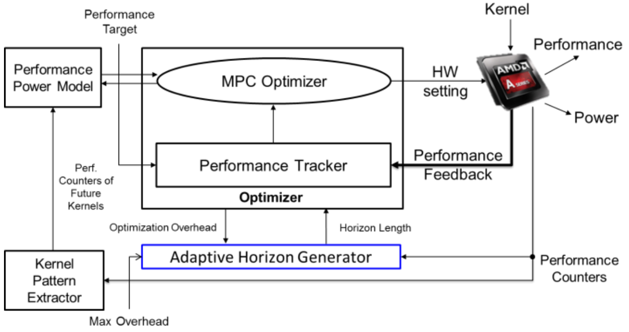
65× cost reduction



(3, 2, 1, 6, 5, 4)

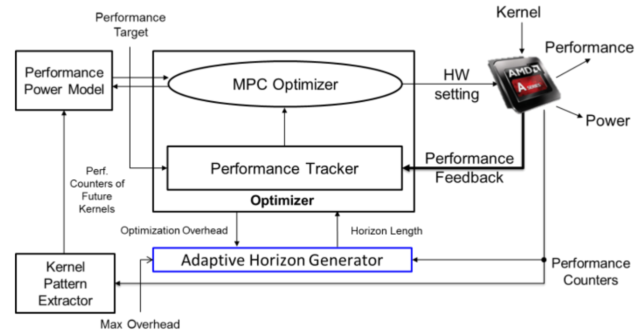
Adaptive Horizon Generator

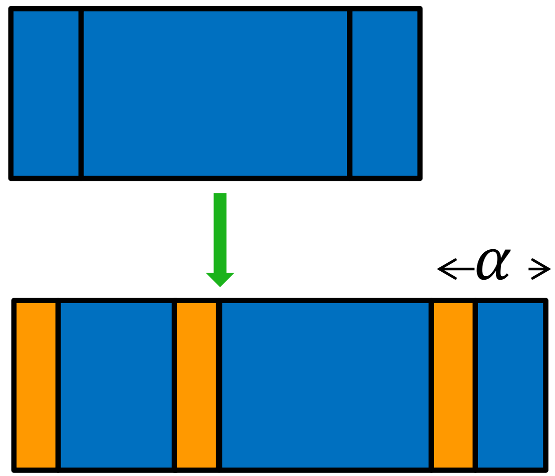
MPC runs between kernel invocations



Adaptive Horizon Generator

MPC runs between kernel invocations



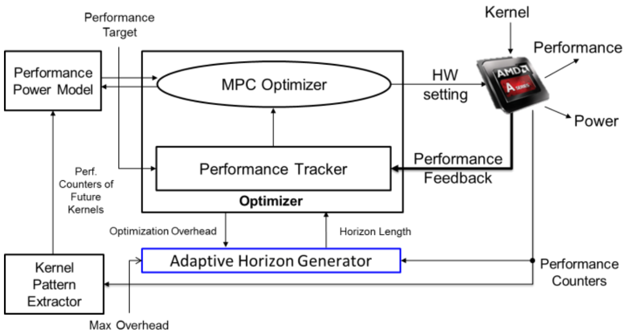


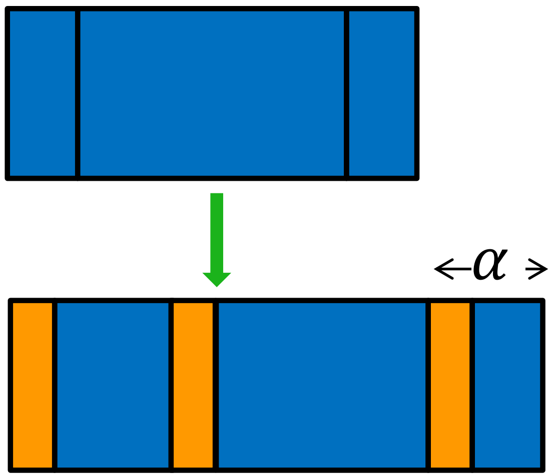
Kernels

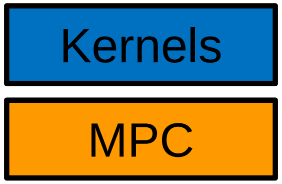
MPC

Adaptive Horizon Generator

MPC runs between kernel invocations

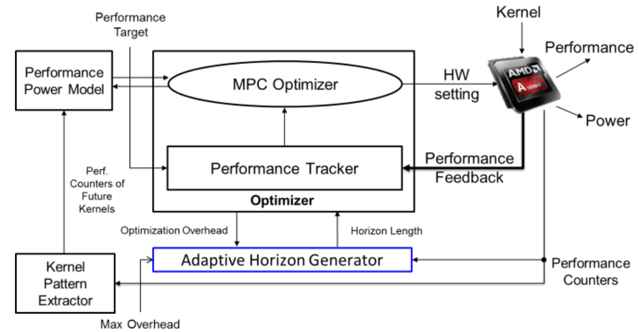




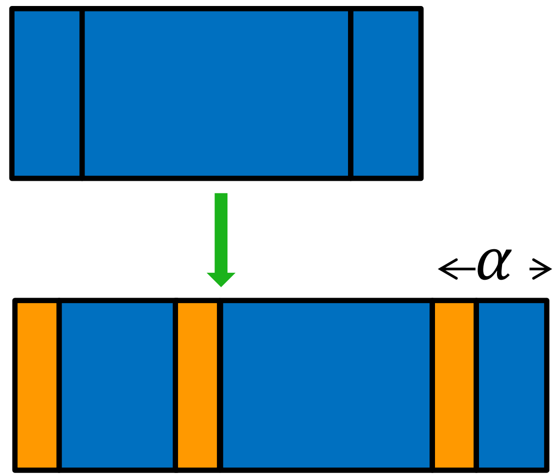


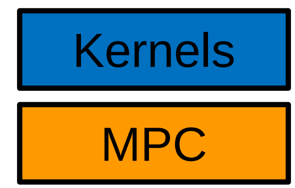
- Longer horizon increases MPC optimization time
- Limit the overhead α by dynamically varying the horizon length

Adaptive Horizon Generator



MPC runs between kernel invocations



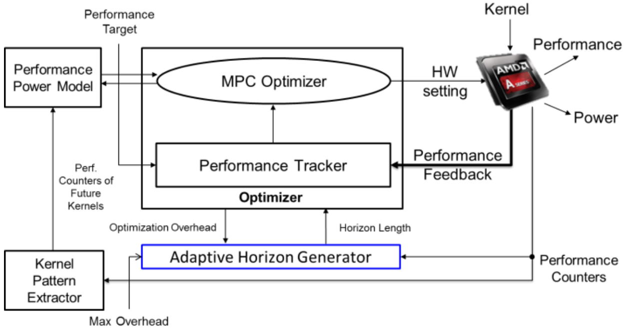


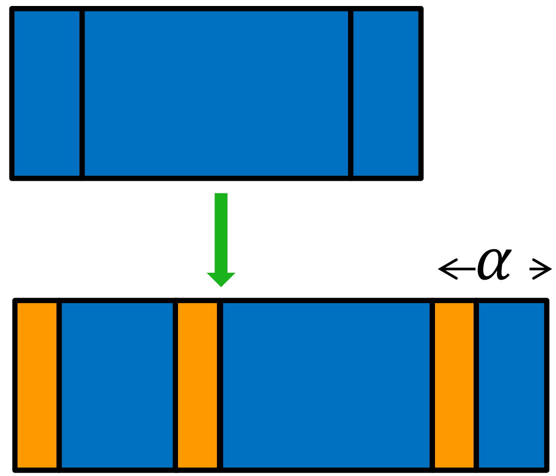
- Longer horizon increases MPC optimization time
- Limit the overhead α by dynamically varying the horizon length
- General Idea:

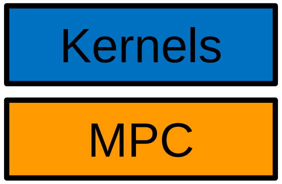
$$\frac{(Total\ Time\ of\ i-1\ kernels) + (Est.MPC\ Opt.\ Time) + (Est.Time\ of\ kernel\ i)}{Baseline\ Time\ of\ i\ kernels} \leq 1+\alpha$$

Adaptive Horizon Generator

MPC runs between kernel invocations







- Longer horizon increases MPC optimization time
- Limit the overhead α by dynamically varying the horizon length
- General Idea:

$$H_i \leq \frac{N}{N} \frac{\left(1+\alpha-\frac{1}{i}\right)\frac{i \times T_{total}}{N} - \sum_{j=1}^{i-1}\left(T_j + T_{MPC,j}\right)}{T_{PPK}}$$

Dynamic GPGPU Power Mgmt. Formulation

- Minimize energy over N GPU kernels such that the performance target is met

$$\min_{\vec{s}} \sum_{i=1}^{N} E_i(s_i)$$

such that

$$\frac{\sum_{i=1}^{N} I_i}{\sum_{i=1}^{N} T_i(s_i)} \geq \frac{I_{total}}{T_{total}}$$

$$s_i \in S \qquad \forall 1 \leq i \leq N$$

$$S = \overrightarrow{cpu} \times \overrightarrow{nb} \times \overrightarrow{gpu} \times \overrightarrow{cu}$$

Theoretically Optimal

- GLPK to solve the Integer Linear Programming (ILP) formulation

$$\min \sum_{i=1}^{N} \sum_{j \in S} E_i(j) X_{ij}$$

such that

$$\sum_{i=1}^{N} I_i - \frac{I_{total}}{T_{total}} \sum_{i=1}^{N} \sum_{j \in S} T_i(j) X_{ij} \geq 0$$

$$\sum_{j \in S} X_{ij} = 1 \qquad \forall\ 1 \leq i \leq N$$

$$X_{ij} \in \{0, 1\} \qquad 1 \leq i \leq N \text{ and } \forall j \in S$$

Predict Previous Kernel (PPK)

- Minimize energy of kernel i such that the runtime performance so far exceeds the target

$$\min_{s_i \in S} E_i(s_i)$$

such that

$$\frac{\sum_{j=1}^{i} I_j}{\sum_{j=1}^{i} T_j(s_j)} \geq \frac{I_{total}}{T_{total}} \qquad \forall 1 \leq i \leq N \text{ and } \forall s_j \in S$$

MPC-based GPGPU Power Manager

- Optimize energy for next H kernels such that the runtime performance at the end of H kernels exceeds the target

$$\min_{\vec{s}} \sum_{j=i}^{i+H-1} E_j(s_j)$$

such that

$$\frac{\sum_{j=1}^{i+H-1} I_j}{\sum_{j=1}^{i+H-1} T_j(s_j)} \geq \frac{I_{total}}{T_{total}} \qquad \forall 1 \leq i \leq N \text{ and } \forall s_j \in S$$

Runtime Performance Tracker

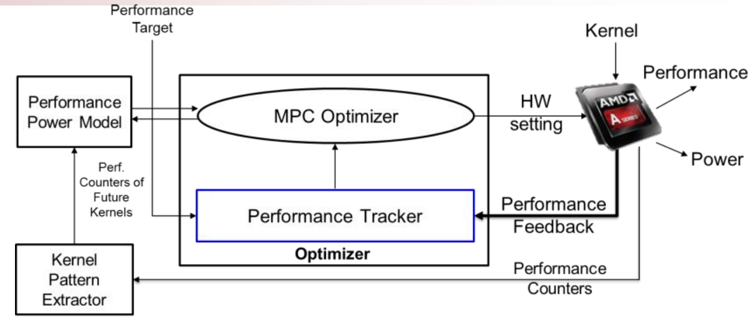
- Performance requirement of kernel, k, is enforced as follows:

$$\frac{\sum_{j=1}^{k-1} I_j + \mathbb{E}[I_k]}{\sum_{j=1}^{k-1} T_j(s_j) + \mathbb{E}[T_k(s_k)]} \geq \frac{I_{total}}{T_{total}}$$

- Kernel time headroom is updated according to:

$$\mathbb{E}[T_k(s_k)] \leq \left(\sum_{j=1}^{k-1} I_j + \mathbb{E}[I_k]\right) \Bigg/ \left(\frac{I_{total}}{T_{total}}\right) - \sum_{j=1}^{k-1} T_j(s_j)$$

Feedback-based Performance Tracker



Performance met previously?

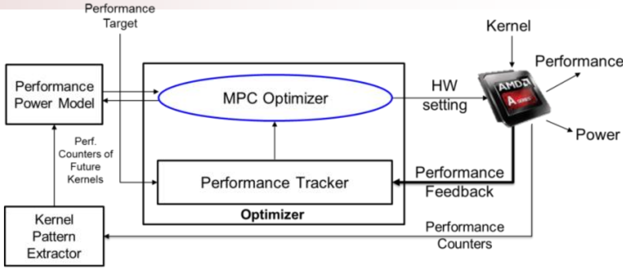
No

Relax optimization
Spend energy
↓
Reduce performance loss

Yes

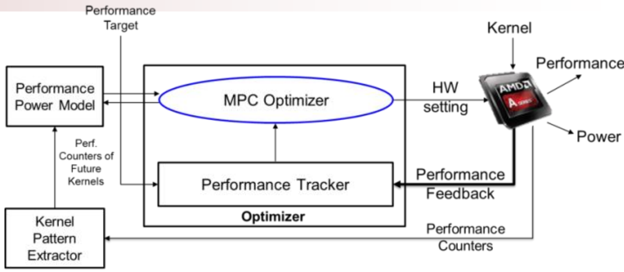
Optimize Aggressively
Reduce energy
↓
Avoid performance loss

MPC Optimizer

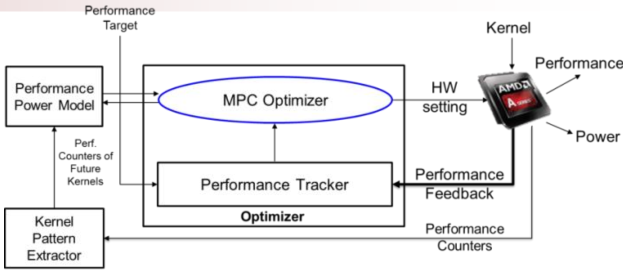


MPC Optimizer

- **Greedy Hill Climbing optimization**
  - Select the HW knob with highest energy sensitivity and search for low energy configuration using hill climbing



MPC Optimizer



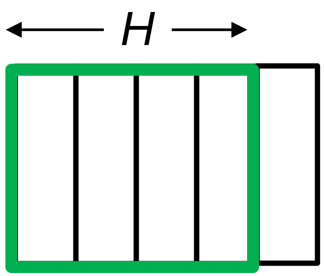
- **Greedy Hill Climbing optimization**
  - Select the HW knob with highest energy sensitivity and search for low energy configuration using hill climbing

- **MPC Search Heuristic**
  - Determine a static order requiring no backtracking
  - General Idea
    - High to low performance (e.g. Spmv): Optimize low performing kernels first
    - Low to high performance (e.g. Kmeans): Optimize high performing kernels first
  - Search cost becomes polynomial

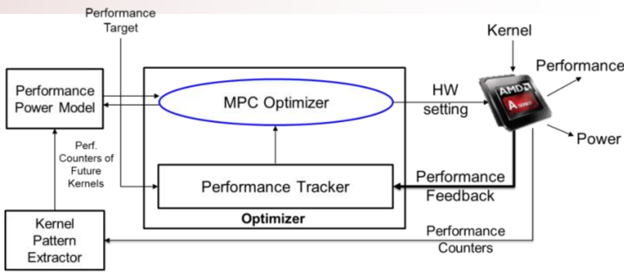
- 65X average reduction in total cost evaluations

An Example

- Search order = (2, 1, 4, 5, 3)



MPC Optimizer



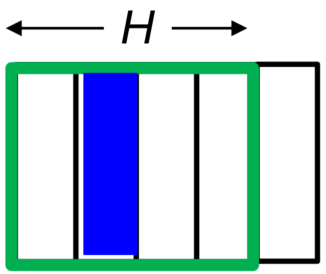
- **Greedy Hill Climbing optimization**
  - Select the HW knob with highest energy sensitivity and search for low energy configuration using hill climbing

- **MPC Search Heuristic**
  - Determine a static order requiring no backtracking
  - General Idea
    - High to low performance (e.g. Spmv): Optimize low performing kernels first
    - Low to high performance (e.g. Kmeans): Optimize high performing kernels first
  - Search cost becomes polynomial

- 65X average reduction in total cost evaluations

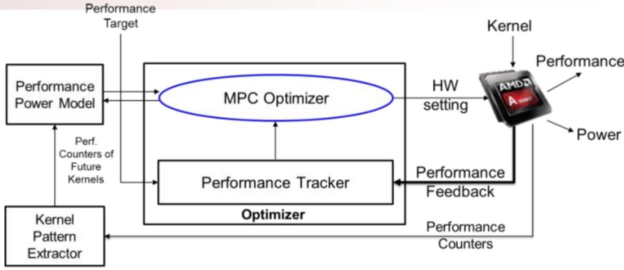
An Example

- Search order = (2, 1, 4, 5, 3)



MPC Optimizer

- **Greedy Hill Climbing optimization**
  - Select the HW knob with highest energy sensitivity and search for low energy configuration using hill climbing

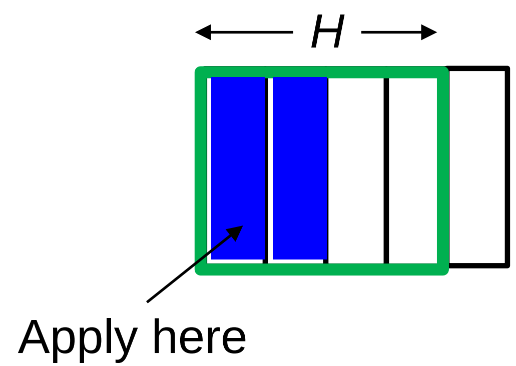


- **MPC Search Heuristic**
  - Determine a static order requiring no backtracking
  - General Idea
    - High to low performance (e.g. Spmv): Optimize low performing kernels first
    - Low to high performance (e.g. Kmeans): Optimize high performing kernels first
  - Search cost becomes polynomial

- 65X average reduction in total cost evaluations

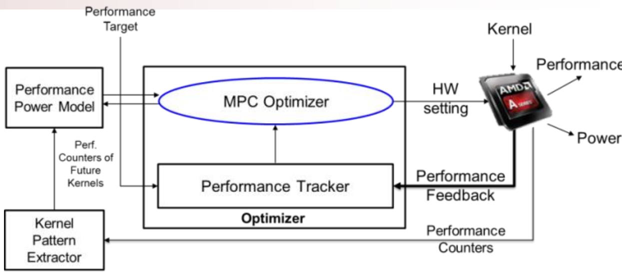
An Example

- Search order = (2, 1, 4, 5, 3)



MPC Optimizer

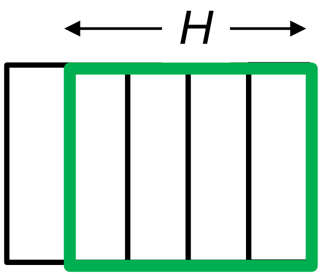
- Greedy hill climbing optimization
  - **Greedy**: Select the HW knob with highest energy sensitivity
  - **Hill Climbing**: Continue finding low energy configuration (within perf target), and stop
  - Reduces search cost by 20X



An Example

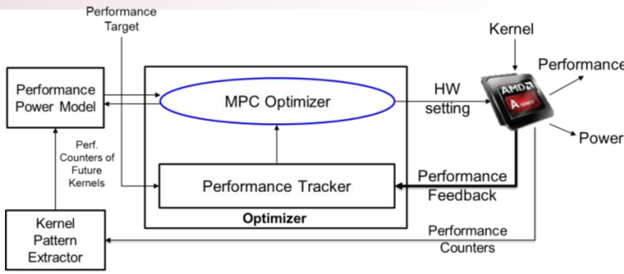
- Search order = (2, 1, 4, 5, 3)

- MPC Search Heuristic
  - Determine a static order requiring no backtracking
  - General Idea
    - High to low performance (e.g. Spmv): Optimize low performing kernels first
    - Low to high performance (e.g. Kmeans): Optimize high performing kernels first
  - Search becomes polynomial



MPC Optimizer

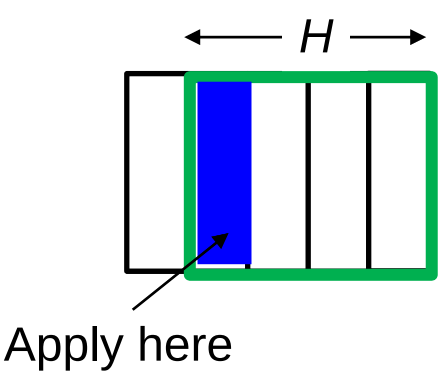
- Greedy hill climbing optimization
  - **Greedy**: Select the HW knob with highest energy sensitivity
  - **Hill Climbing**: Continue finding low energy configuration (within perf target), and stop
  - Reduces search cost by 20X



An Example

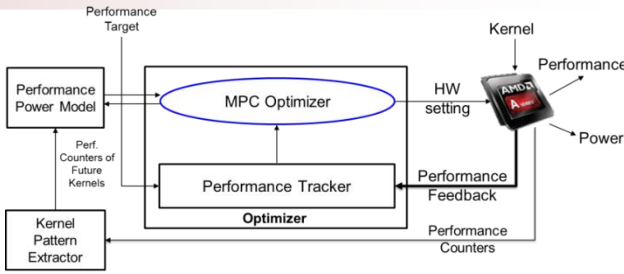
- Search order = (2, 1, 4, 5, 3)

- MPC Search Heuristic
  - Determine a static order requiring no backtracking
  - General Idea
    - High to low performance (e.g. Spmv): Optimize low performing kernels first
    - Low to high performance (e.g. Kmeans): Optimize high performing kernels first
  - Search becomes polynomial



MPC Optimizer

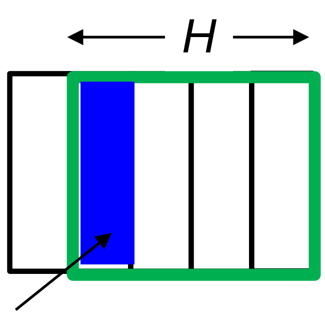
- Greedy hill climbing optimization
  - **Greedy**: Select the HW knob with highest energy sensitivity
  - **Hill Climbing**: Continue finding low energy configuration (within perf target), and stop
  - Reduces search cost by 20X



An Example

- Search order = (2, 1, 4, 5, 3)

- MPC Search Heuristic
  - Determine a static order requiring no backtracking
  - General Idea
    - High to low performance (e.g. Spmv): Optimize low performing kernels first
    - Low to high performance (e.g. Kmeans): Optimize high performing kernels first
  - Search becomes polynomial
- Average reduction in total cost evaluations: 65X



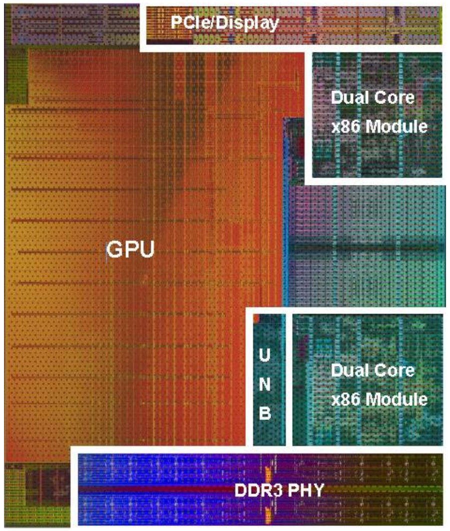
Apply here

GPGPU Benchmarks

Category	Benchmarks	Benchmark Suite	Regular Expression
Regular	mandelbulbGPU	Phoronix	A^{20}
	Nbody	AMD APP SDK	A^{10}
	juliaGPU	Phoronix	A^{10}
Irregular with repetitive pattern	EigenValue	AMD APP SDK	$(AB)^{5}$
	XSBench	Exascale	$(ABC)^{2}$
Irregular with non-repetitive pattern	Spmv	SHOC	$A^{10}B^{10}C^{10}$
	Kmeans	Rodinia	AB^{20}
Irregular with kernels varying with input	swat	OpenDwarfs	Complex pattern
	color	Pannotia	
	pb-bfs	Parboil	
	mis	Pannotia	
	srad	Rodinia	
	lulesh	Exascale	
	lud	Rodinia	
	hybridsort	Rodinia	

Experimental Testbed (Detailed)

- AMD A10-7850K APU
  - 2 out-of-order dual core CPUs
  - GPU contains 512 processing elements (8 CUs) at 720 MHz
    - Each CU has 4 SIMD Vector Units
    - 16 PEs per SIMD vector unit



- DVFS states

CPU P States	Voltage (V)	Freq (GHz)
P1	1.325	3.9
P2	1.3125	3.8
P3	1.2625	3.7
P4	1.225	3.5
P5	1.0625	3
P6	0.975	2.4
P7	0.8875	1.7

NB and GPU share the same voltage rail

NB P States	Freq (GHz)	Memory Freq (MHz)
NB0	1.8	800
NB1	1.6	800
NB2	1.4	800
NB3	1.1	333

GPU P States	Voltage (V)	Freq (GHz)
DPM0	0.95	351
DPM1	1.05	450
DPM2	1.125	553
DPM3	1.1875	654
DPM4	1.225	720

- GPU CUs and DVFS states changed in multiples of 2
- Total HW configuration: 336

Experimental Setup

- **15 GPGPU Benchmarks**
  - Sampled from 73 benchmarks
  - 75% irregular
  - 44% vary with input

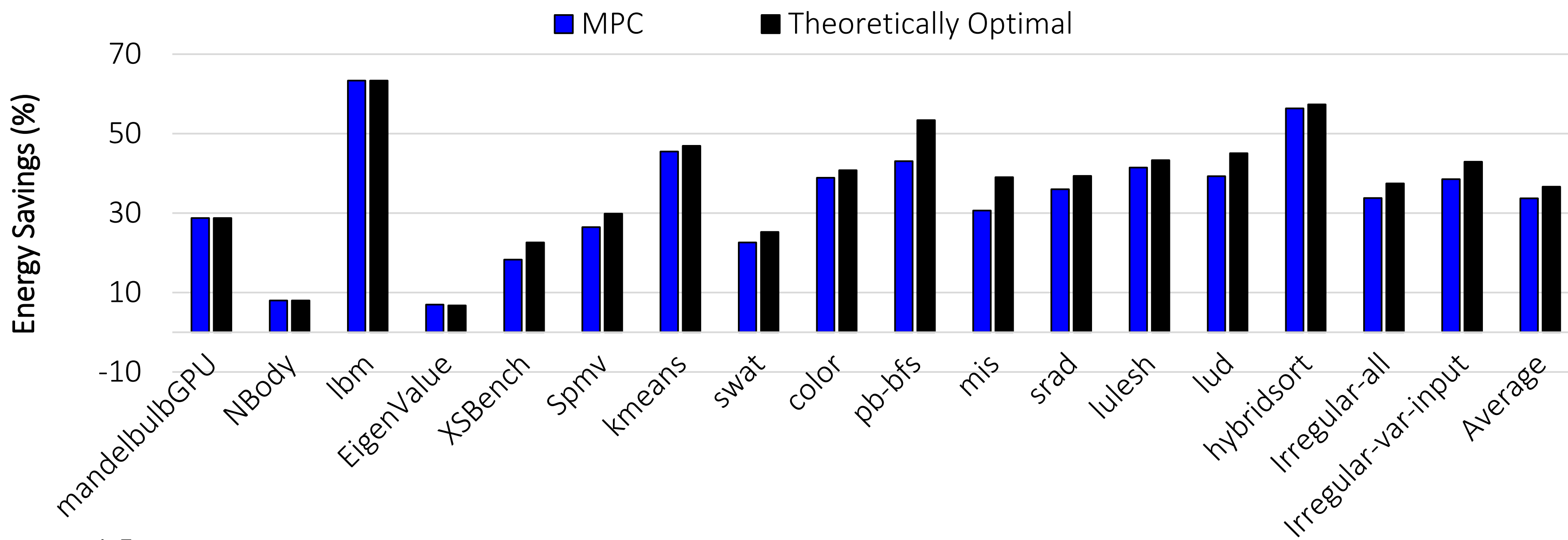
Category	Benchmarks	Benchmark Suite	Regular Expression
Regular	mandelbulbGPU	Phoronix	A^{20}
	Nbody	AMD APP SDK	A^{10}
	juliaGPU	Phoronix	A^{10}
Irregular with repetitive pattern	EigenValue	AMD APP SDK	$(AB)^{5}$
	XSBench	Exascale	$(ABC)^{2}$
Irregular with non-repetitive pattern	Spmv	SHOC	$A^{10}B^{10}C^{10}$
	Kmeans	Rodinia	AB^{20}
Irregular with kernels varying with input	swat	OpenDwarfs	Complex pattern
	color	Pannotia	
	pb-bfs	Parboil	
	mis	Pannotia	
	srad	Rodinia	
	lulesh	Exascale	
	lud	Rodinia	
	hybridsort	Rodinia	

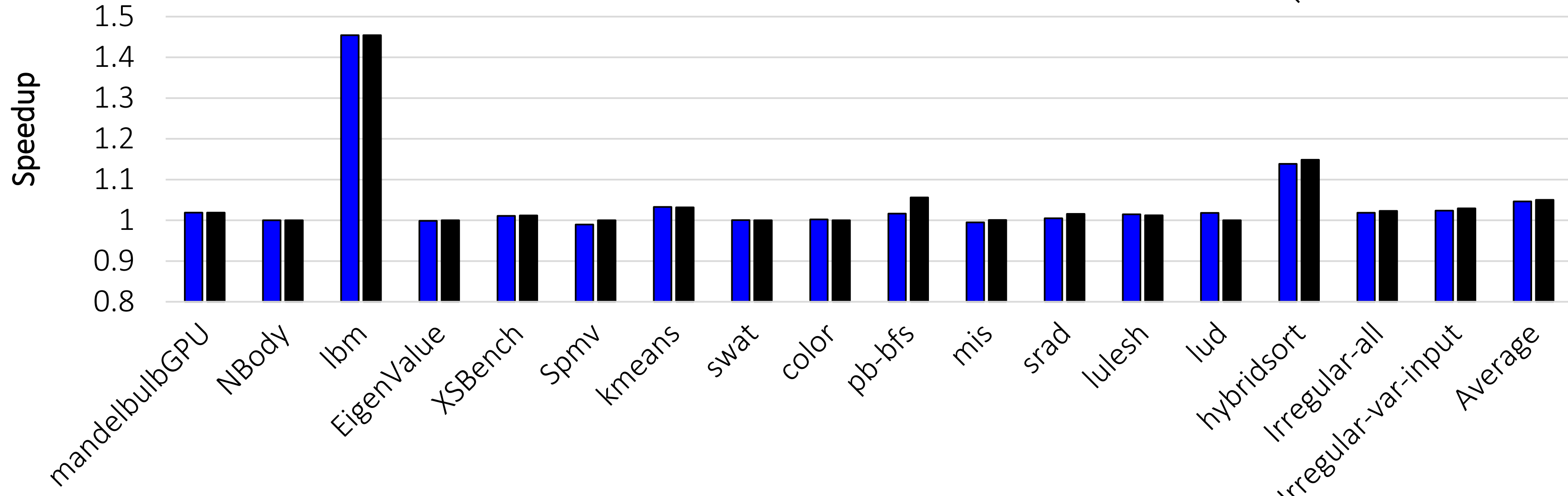
- **Baseline scheme**
  - AMD Turbo Core

- **Algorithms**
  - PPK
  - MPC
  - TO

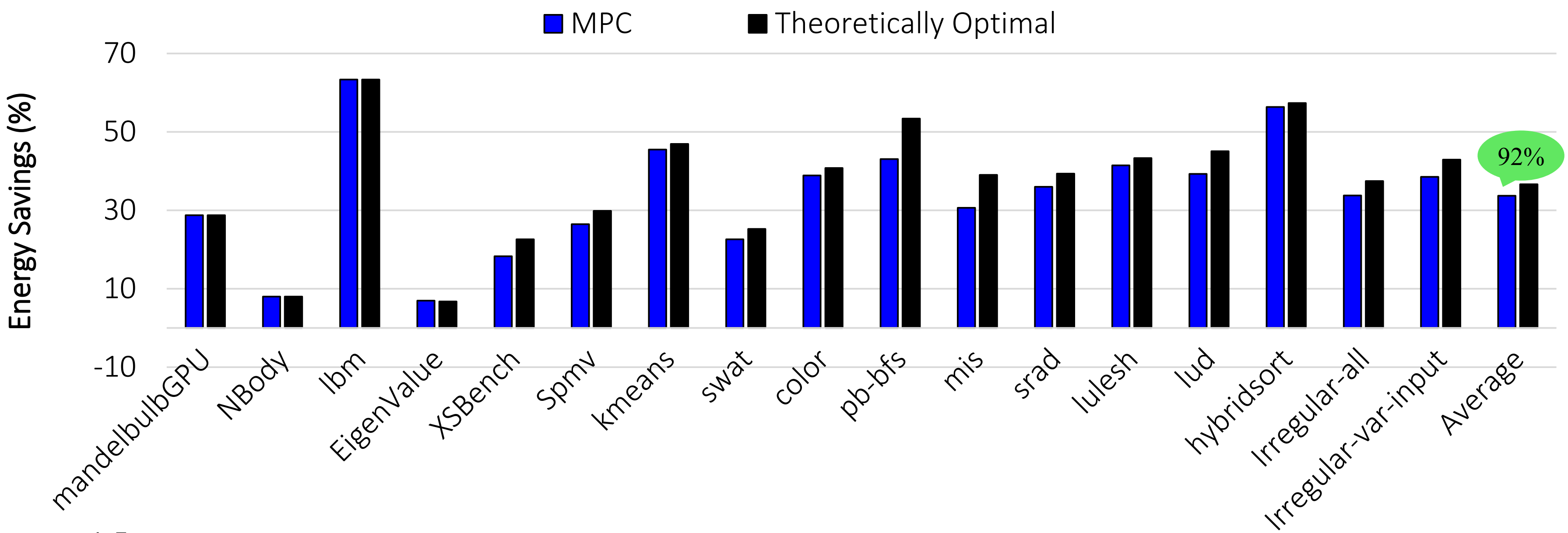
- **Results based on real hardware traces**

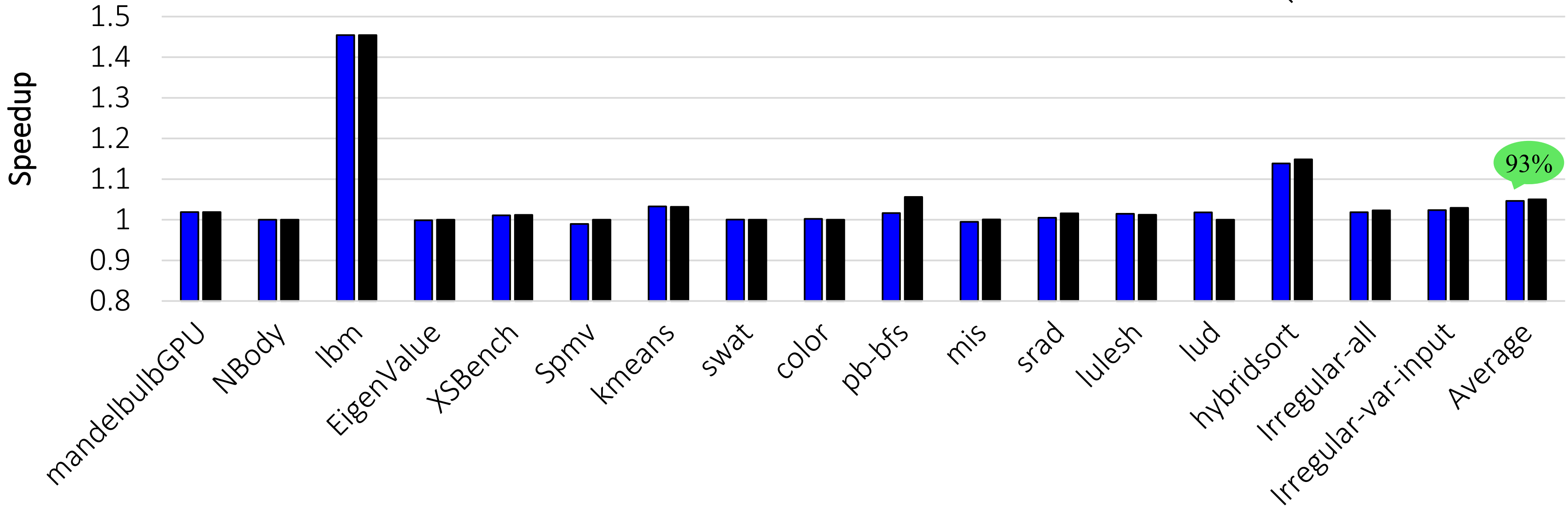
Polynomial MPC w/ Theoretical Optimal



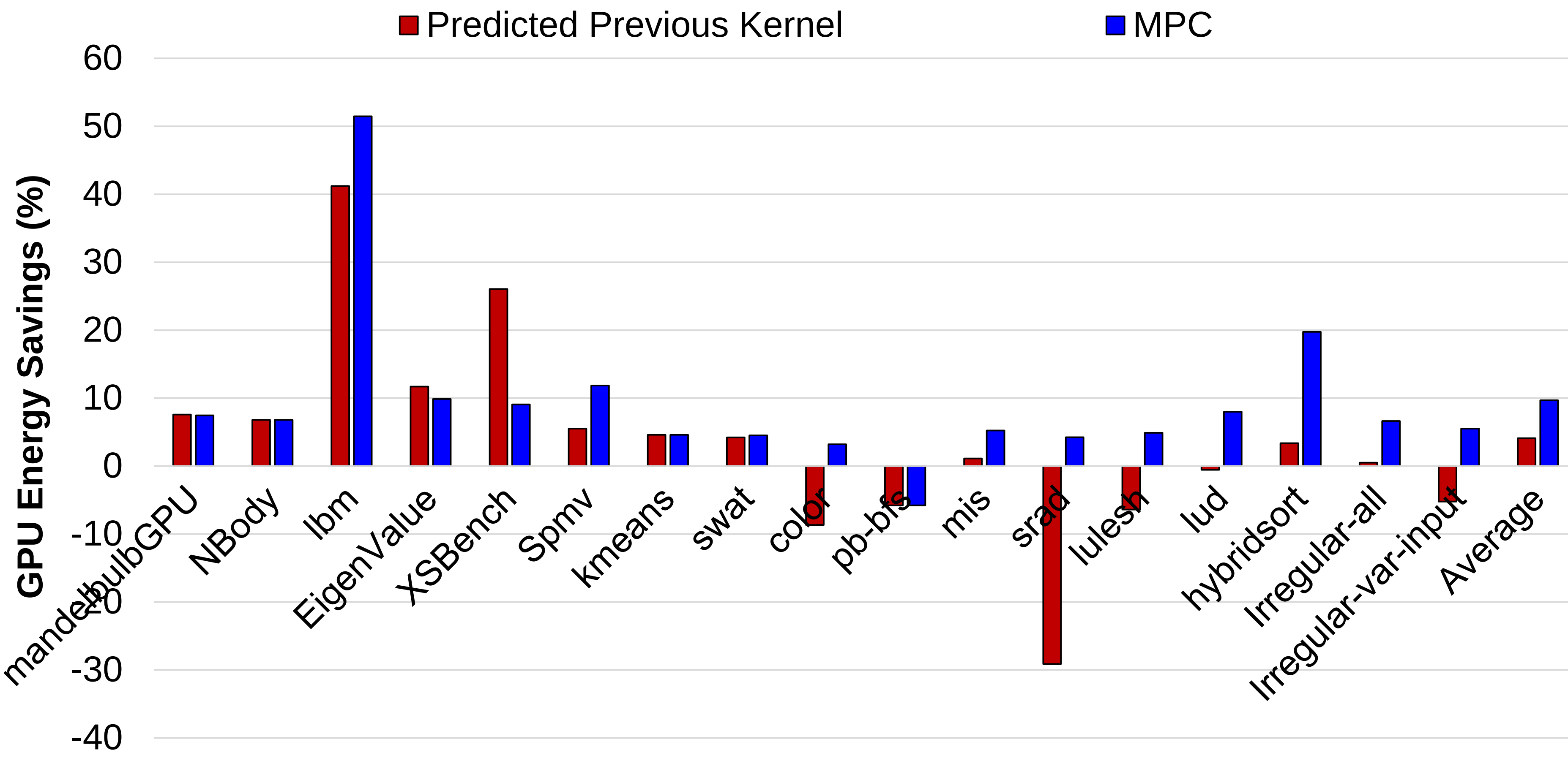


Polynomial MPC w/ Theoretical Optimal

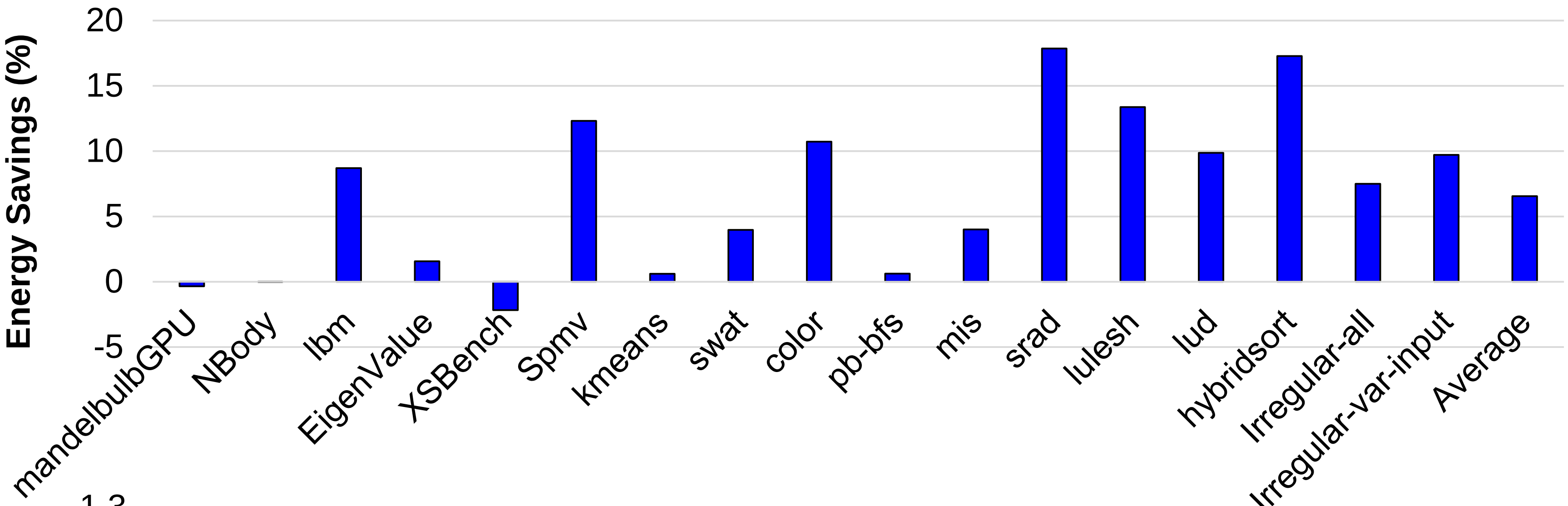


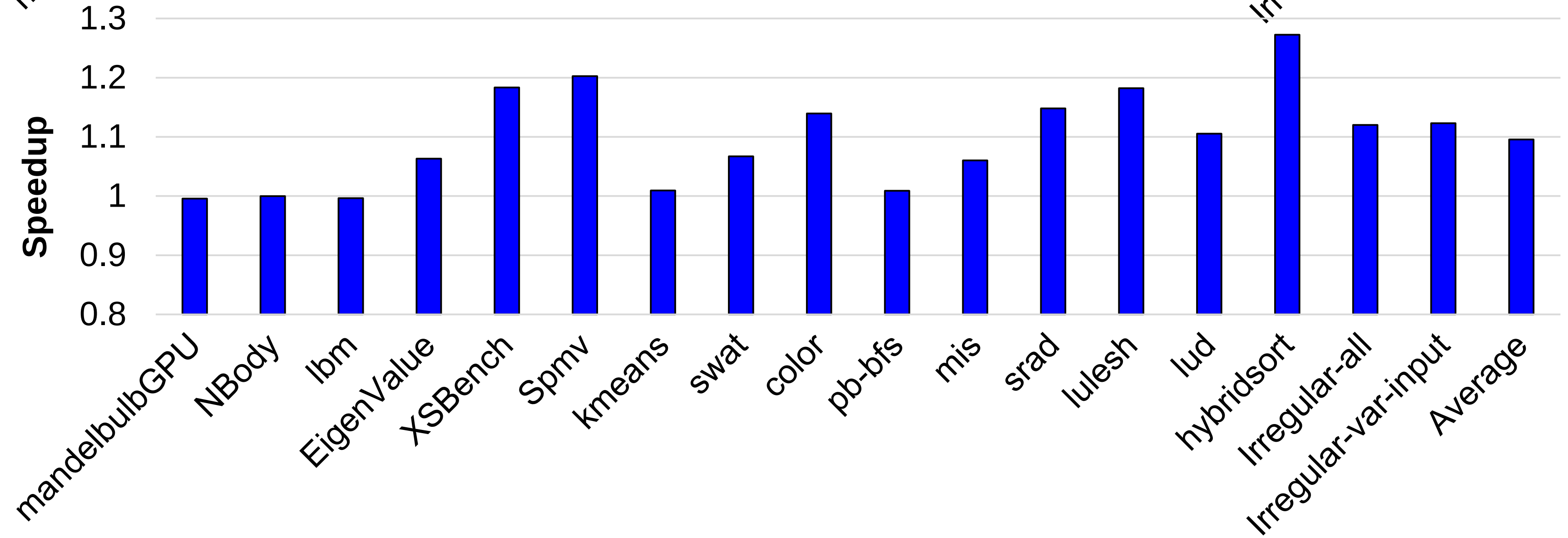


GPU Energy Savings

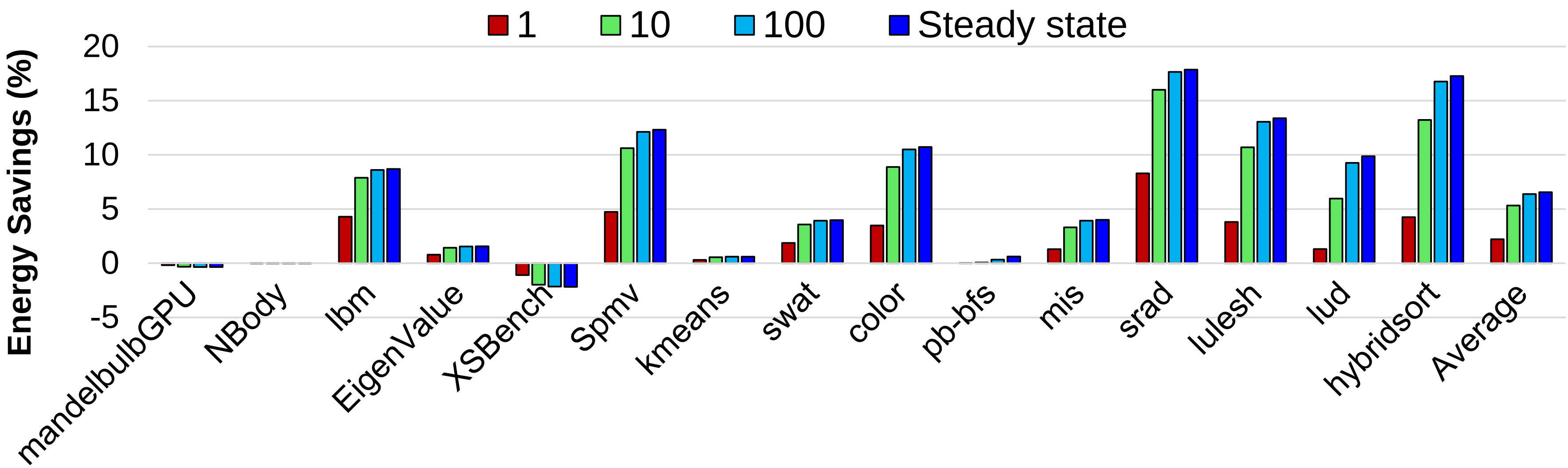


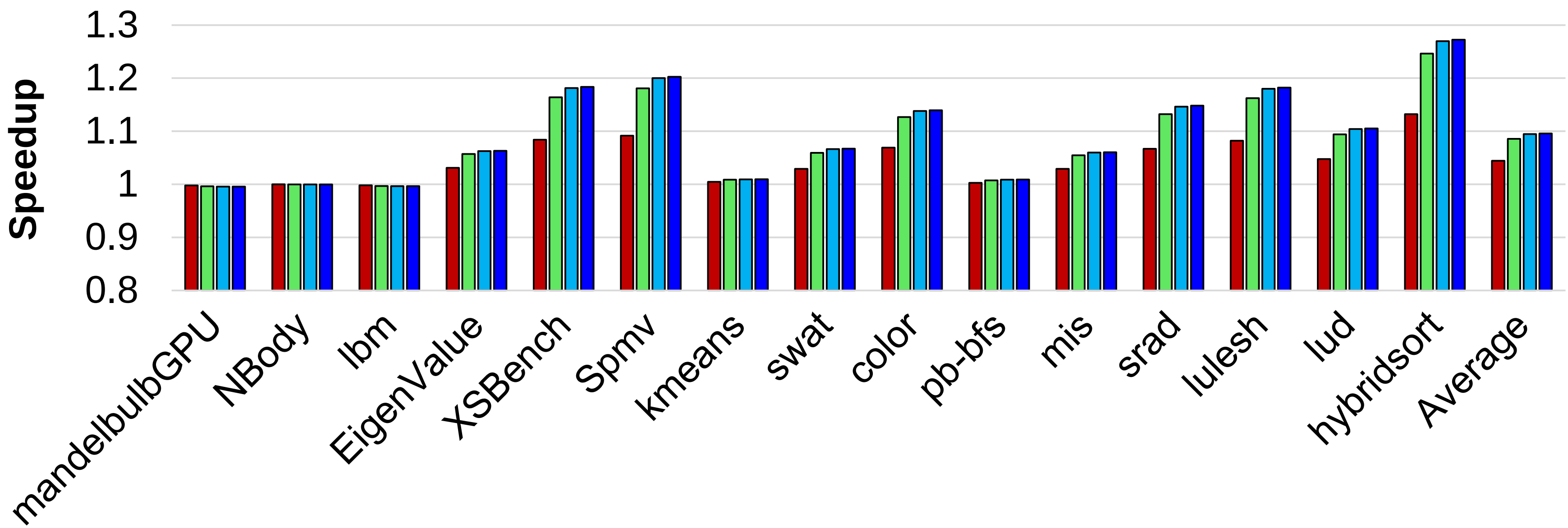
MPC Energy-Performance w.r.t PPK





Amortization of Initial Loses





Final Takeaway

- MPC is a versatile scheme
  - Does not hurt gains of regular benchmarks
  - Improves energy savings
  - Reduces performance loss
- Online MPC reduces performance loss compared to traditional approaches
- Online MPC is resilient to prediction inaccuracy due to
  - Performance feedback
  - MPC's greedy and heuristic approximations depend minimally on prediction models