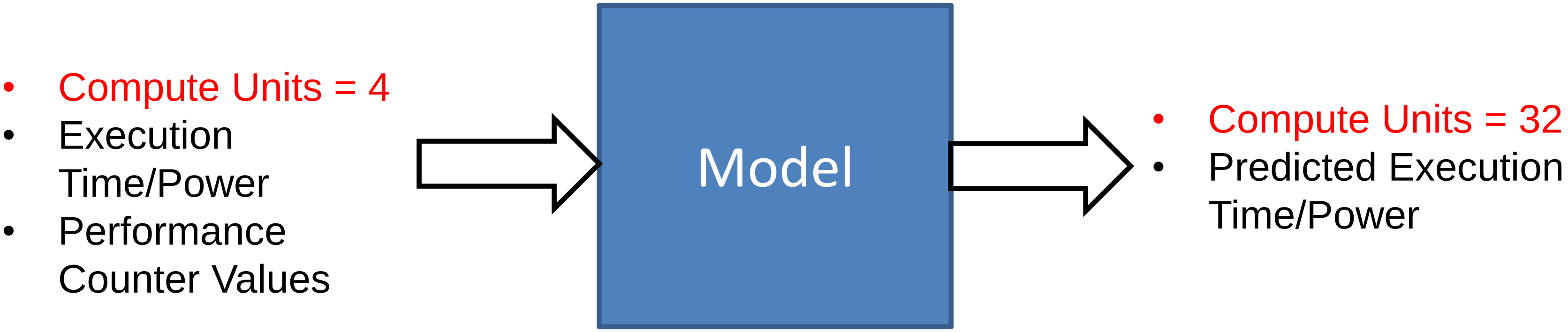
GPGPU Performance and Power Estimation Using Machine Learning

Gene Wu – UT Austin
Joseph Greathouse – AMD Research
Alexander Lyashevsky – AMD Research
Nuwan Jayasena – AMD Research
Derek Chiou – UT Austin

Goals

- Create GPU power & performance scalability models:
  - Capable of predicting for a wide range of settings
    - Number of Compute Units(CUs) or parallel cores
    - GPU Core(Engine) Frequency
    - Memory Frequency
  - Predict many hardware configurations from data gathered on a single configuration



- Compute Units = 4
- Execution Time/Power
- Performance Counter Values

Model

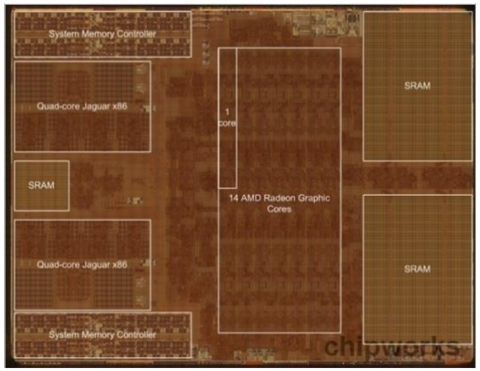
- Compute Units = 32
- Predicted Execution Time/Power

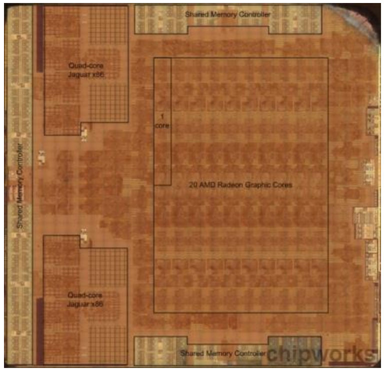
Why Power and Performance Estimation?

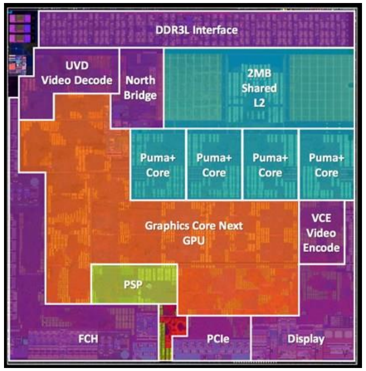
- Feedback to programmer.

Why Power and Performance Estimation?

- Feedback to programmer.
- HW Design Exploration (e.g. Semi-Custom)

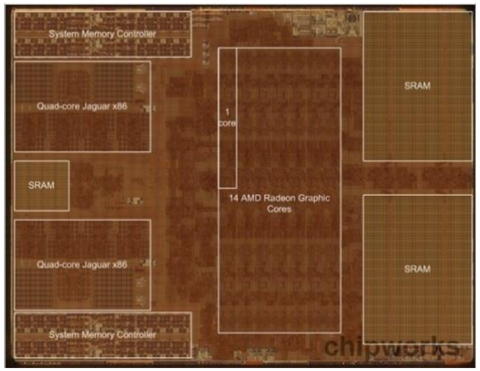


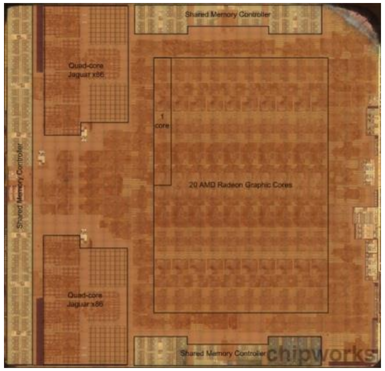


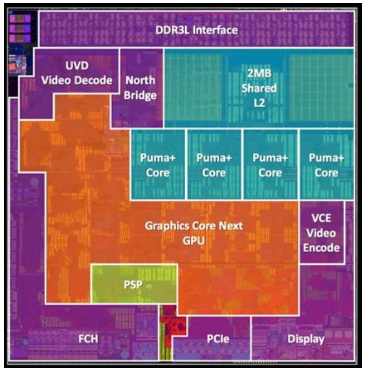


Why Power and Performance Estimation?

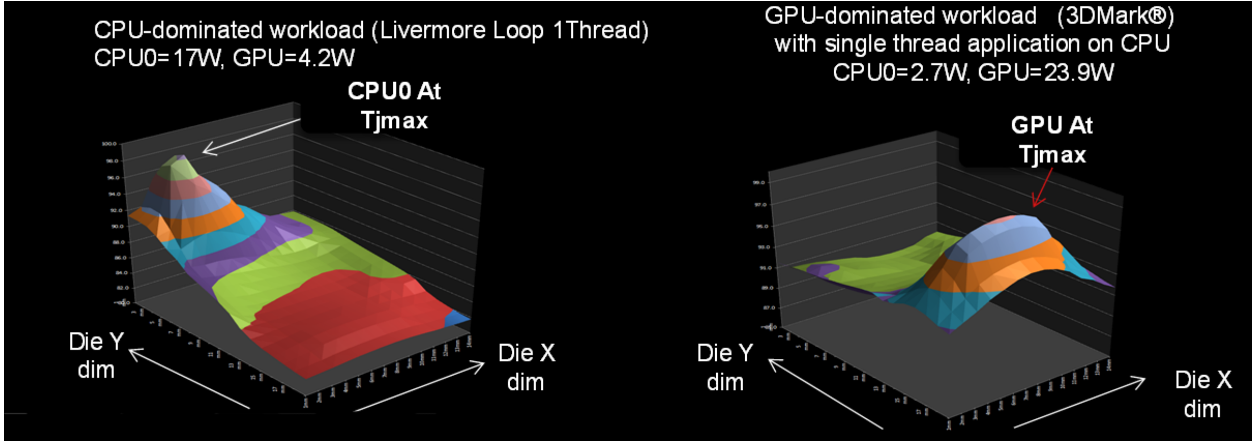
- Feedback to programmer.
- HW Design Exploration (e.g. Semi-Custom)







- Online reconfiguration (e.g. DVFS)


CPU-dominated workload (Livermore Loop 1Thread)
CPU0=17W, GPU=4.2W

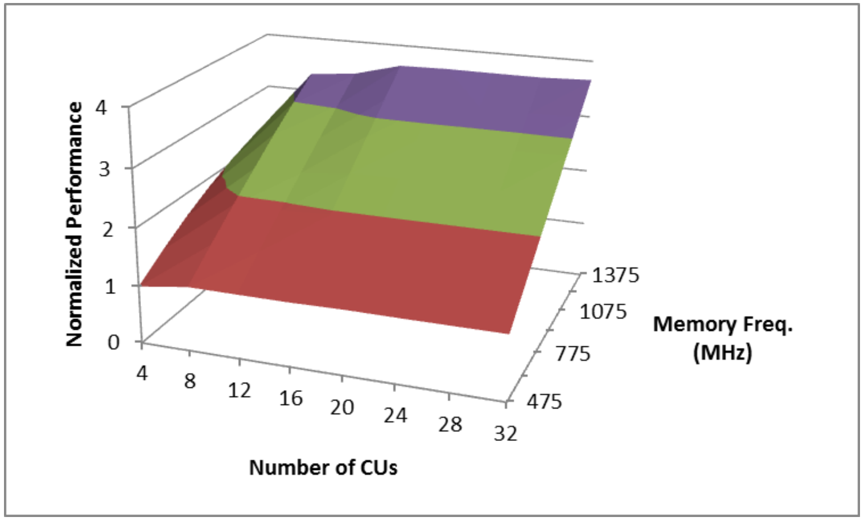
GPU-dominated workload (3DMark®)
with single thread application on CPU
CPU0=2.7W, GPU=23.9W

Outline

- Goals
- Model Overview
- Model Construction
- Results

Base to Target Config. Execution

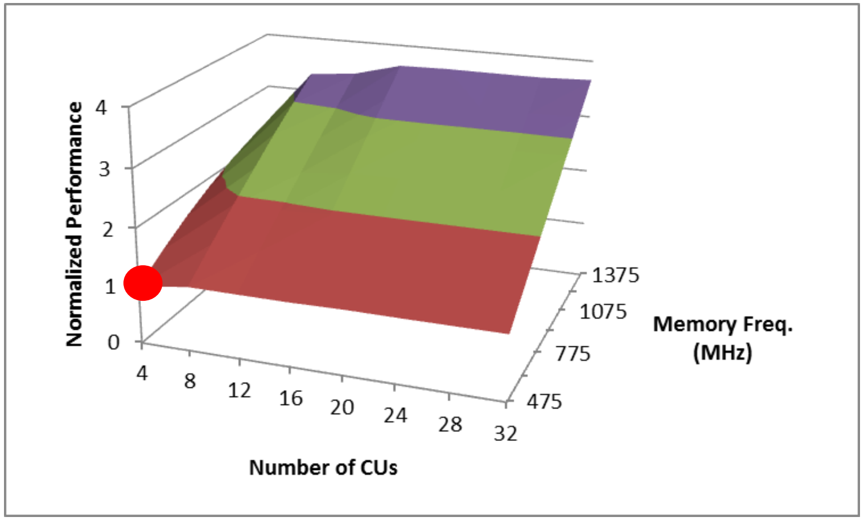
- Hardware Configuration
  - Compute unit(CU) count
  - Engine frequency
  - Memory frequency



- The hardware configuration from which measurements are taken is the Base Hardware Configuration
- The hardware configuration that we wish to predict performance/power at is the Target Hardware Configuration

Base to Target Config. Execution

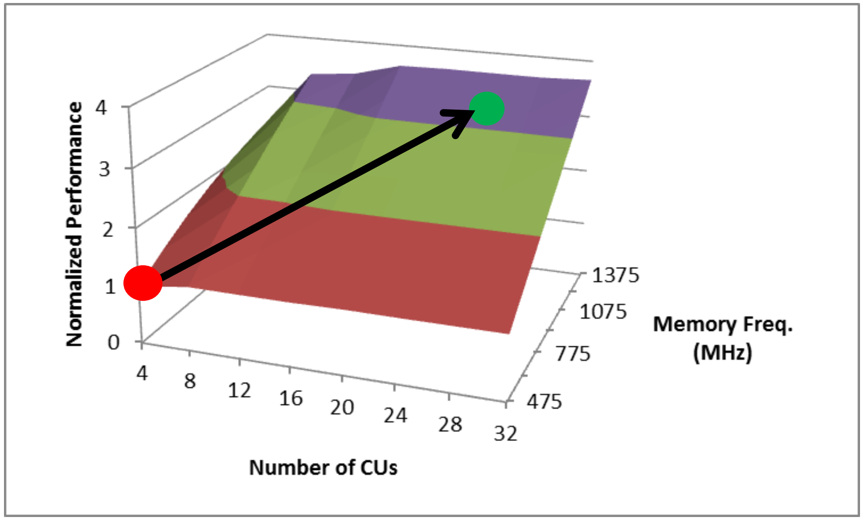
- Hardware Configuration
  - Compute unit(CU) count
  - Engine frequency
  - Memory frequency



- The hardware configuration from which measurements are taken is the Base Hardware Configuration
- The hardware configuration that we wish to predict performance/power at is the Target Hardware Configuration

Base to Target Config. Execution

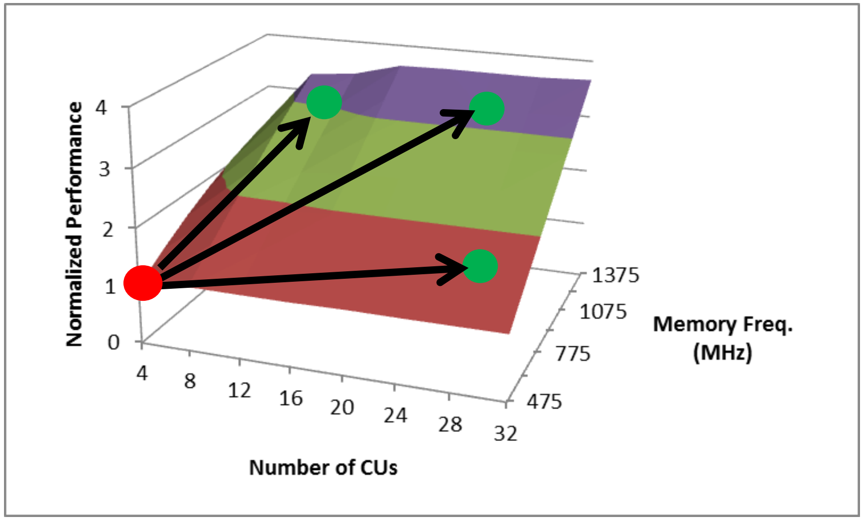
- Hardware Configuration
  - Compute unit(CU) count
  - Engine frequency
  - Memory frequency



- The hardware configuration from which measurements are taken is the Base Hardware Configuration
- The hardware configuration that we wish to predict performance/power at is the Target Hardware Configuration

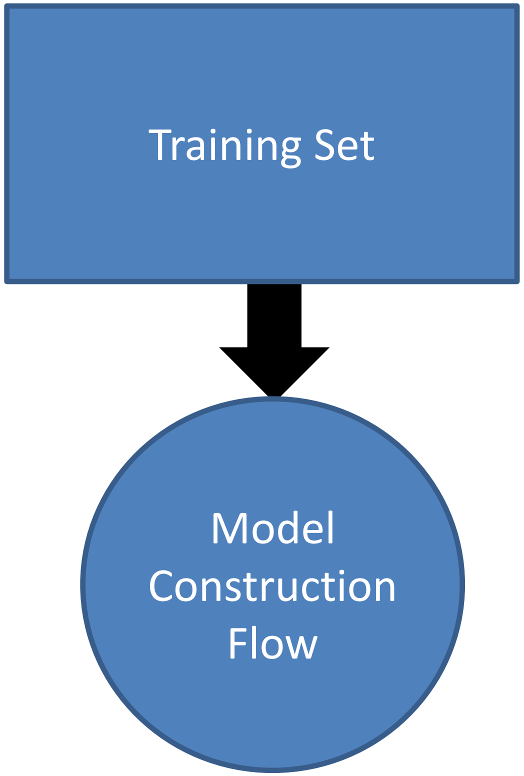
Base to Target Config. Execution

- Hardware Configuration
  - Compute unit(CU) count
  - Engine frequency
  - Memory frequency



- The hardware configuration from which measurements are taken is the Base Hardware Configuration
- The hardware configuration that we wish to predict performance/power at is the Target Hardware Configuration

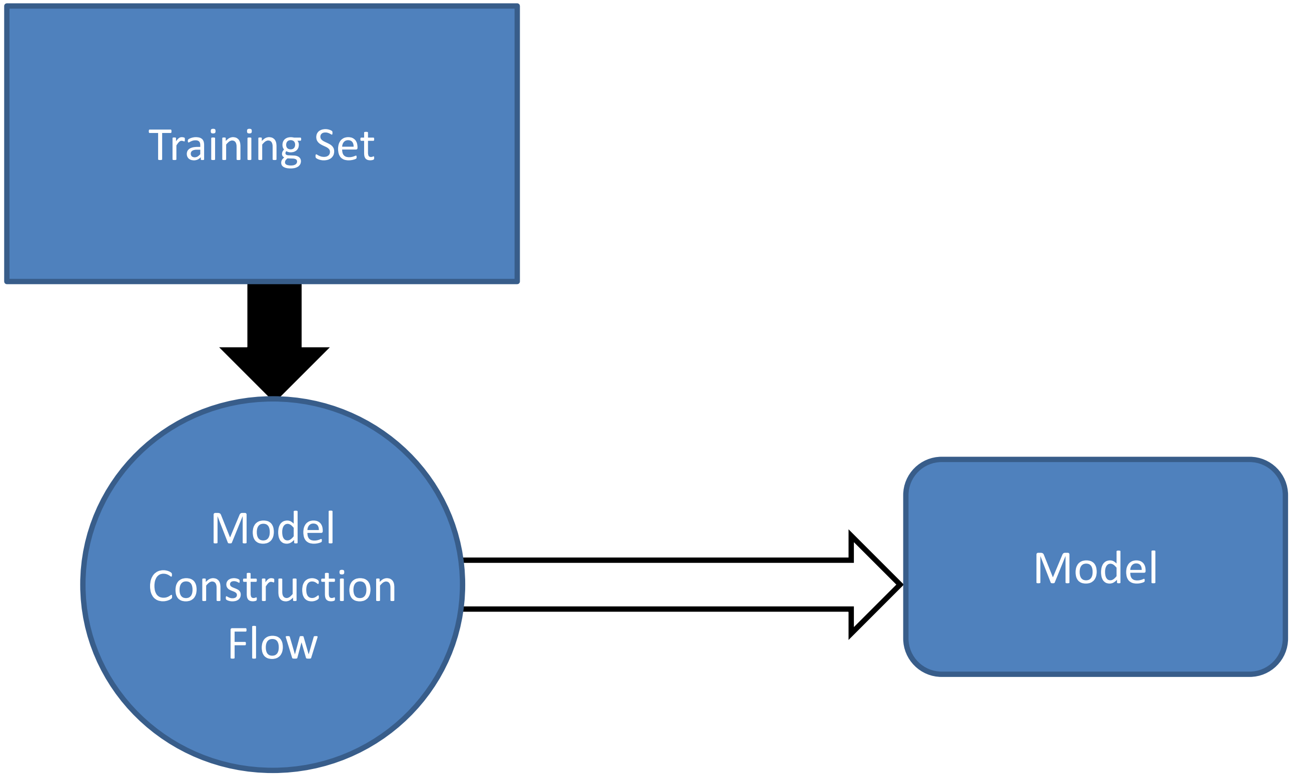
Model Construction and Usage Flow



Training Set

Model Construction Flow

Model Construction and Usage Flow

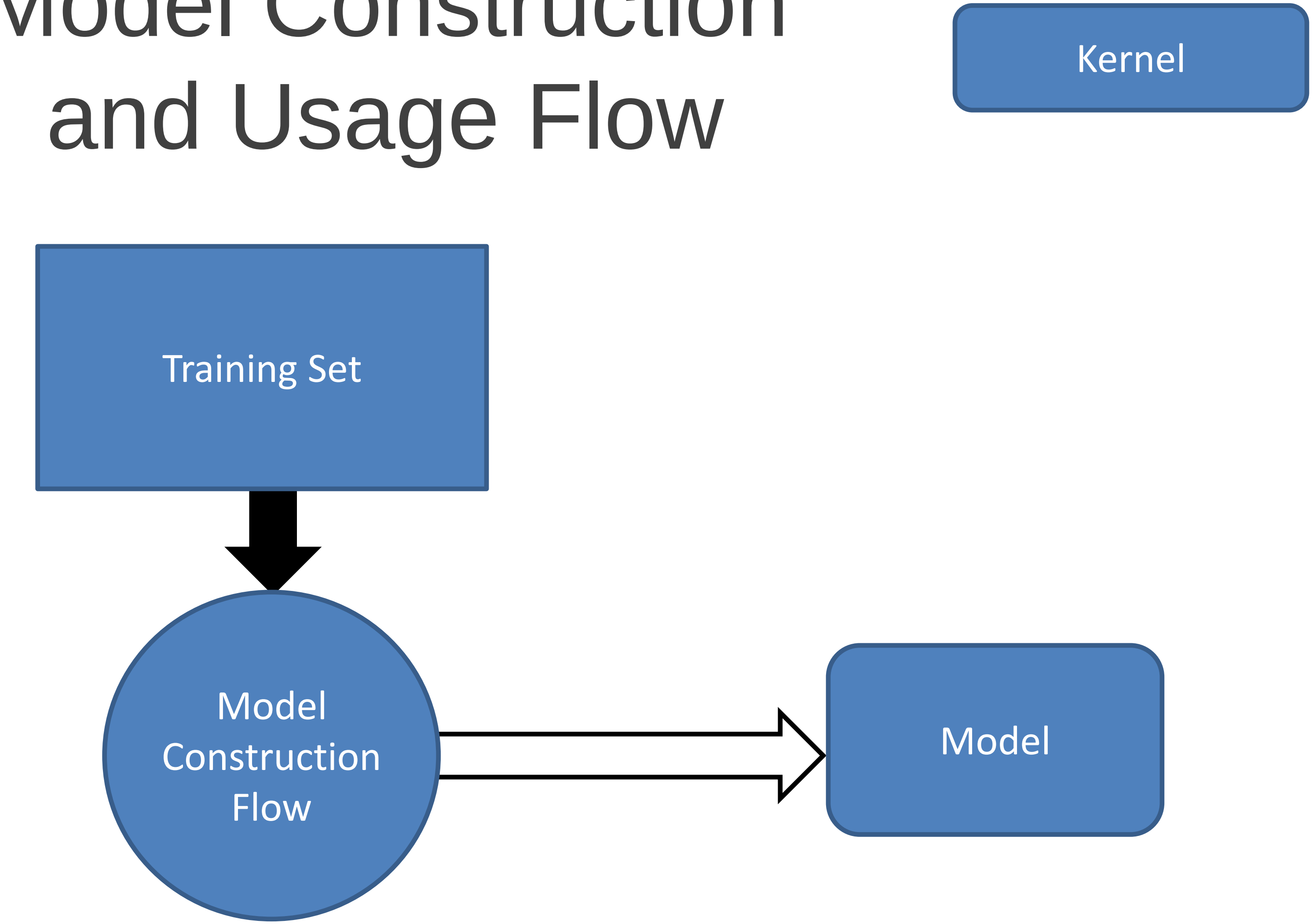


Training Set

Model Construction Flow

Model

Model Construction and Usage Flow



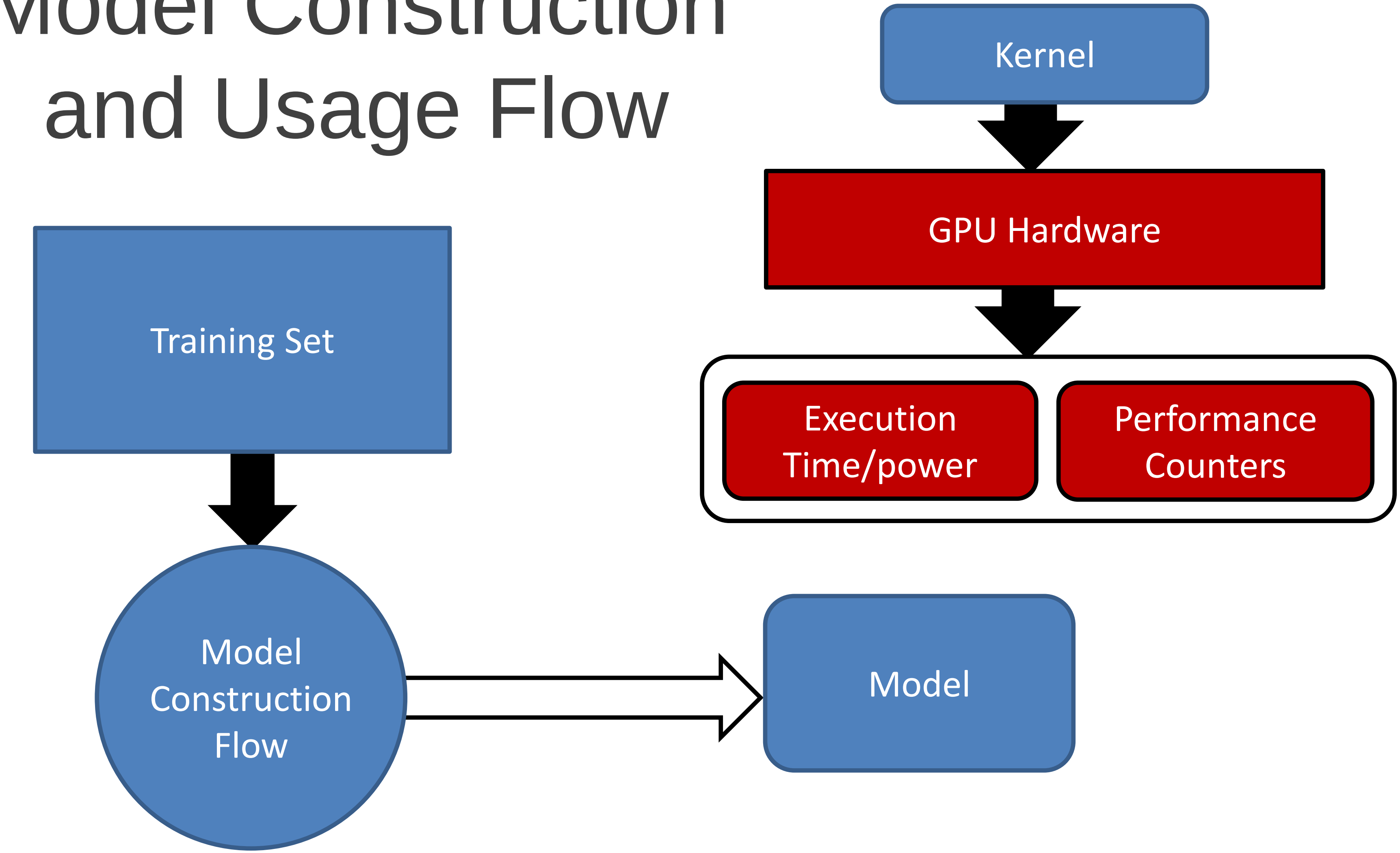
Kernel

Training Set

Model Construction Flow

Model

Model Construction and Usage Flow



Kernel

GPU Hardware

Execution Time/power

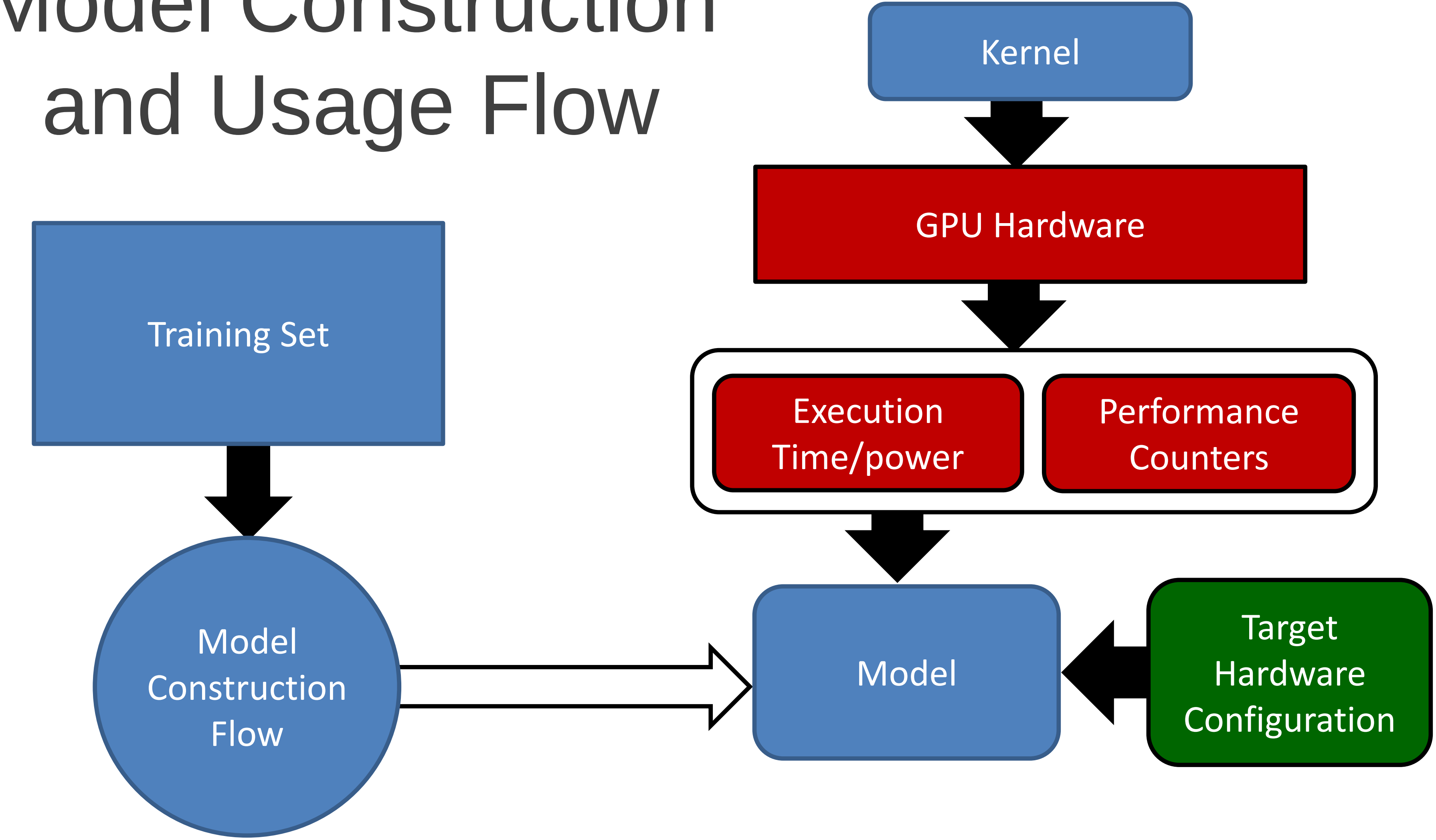
Performance Counters

Training Set

Model Construction Flow

Model

Model Construction and Usage Flow



Kernel

GPU Hardware

Execution Time/power

Performance Counters

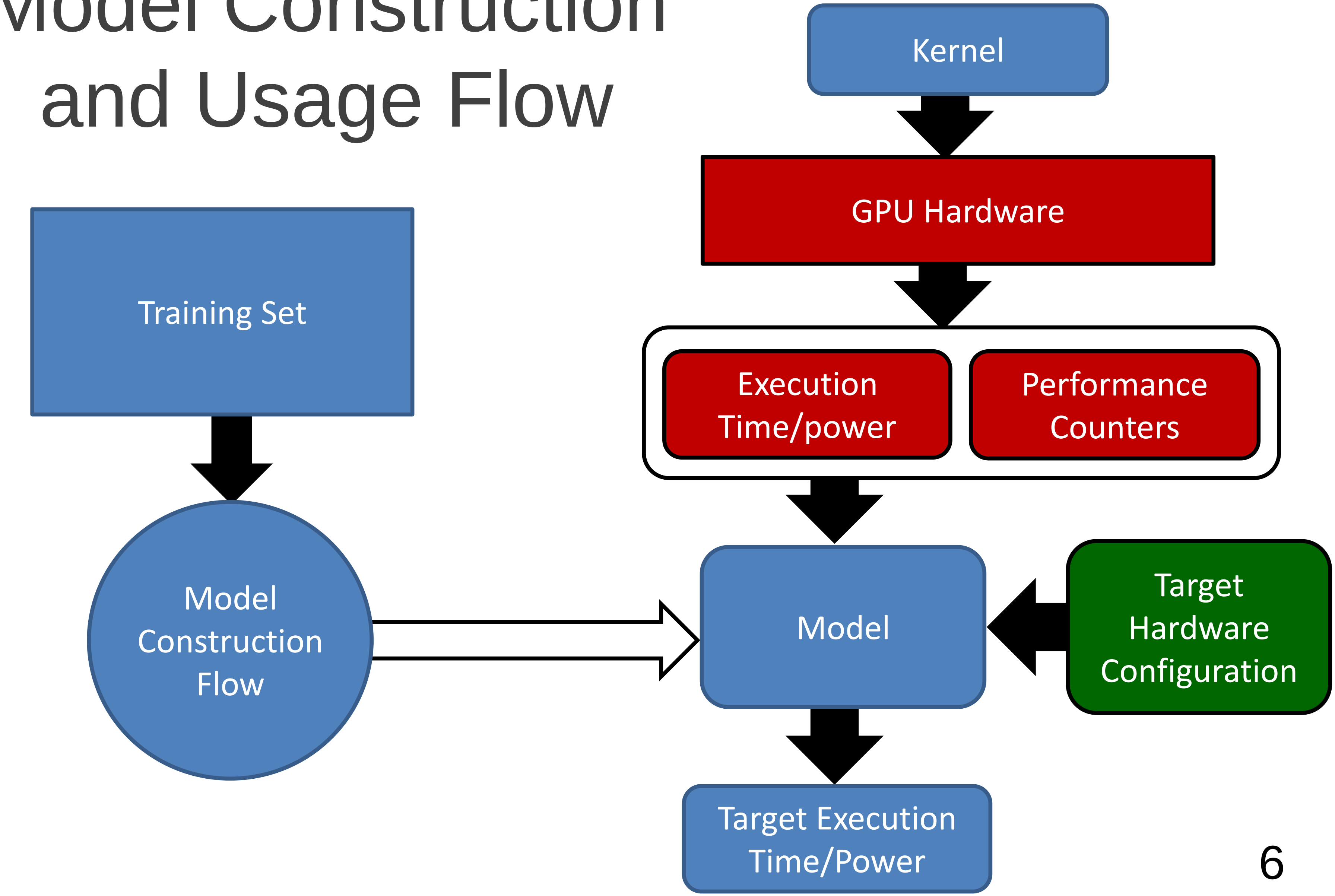
Training Set

Model Construction Flow

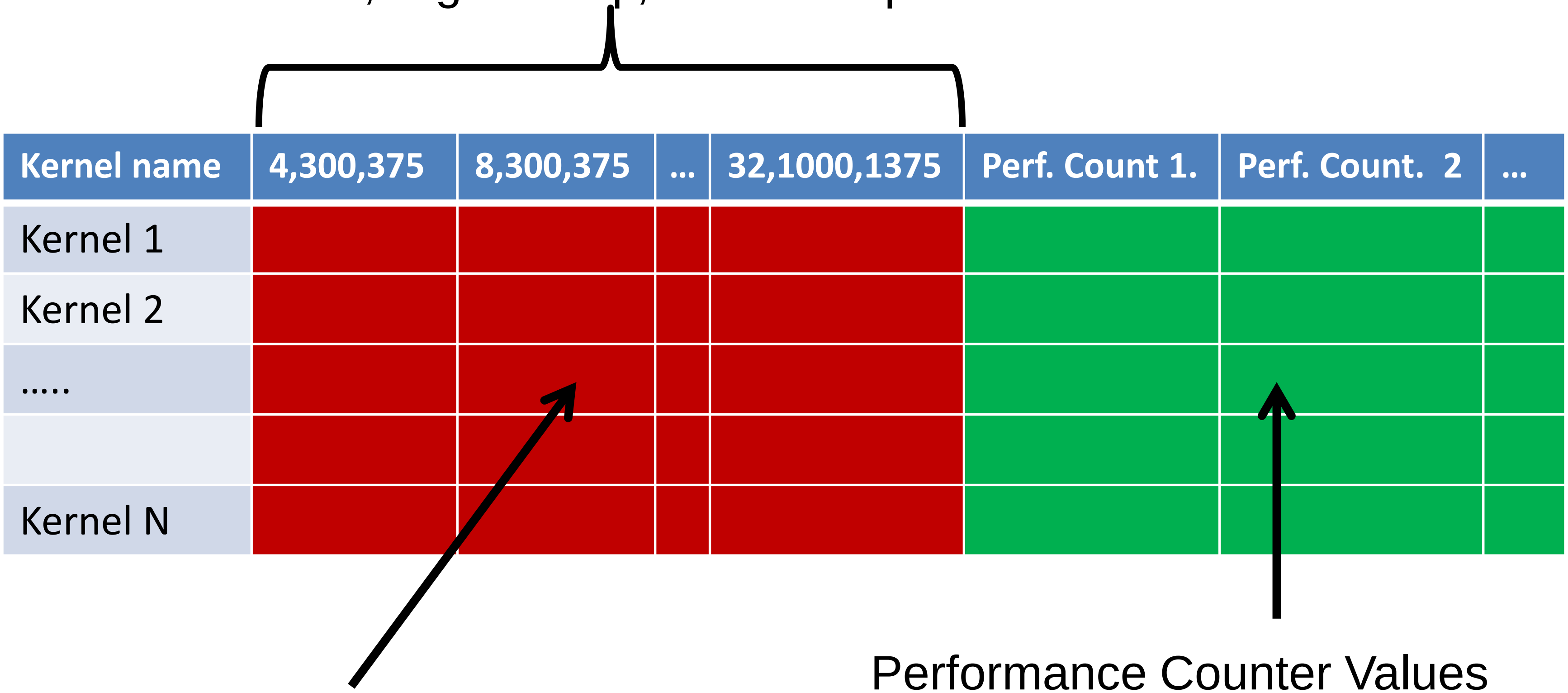
Model

Target Hardware Configuration

Model Construction and Usage Flow



Training Set



CU count,Engine freq., Mem. Freq.

Kernel name	4,300,375	8,300,375	…	32,1000,1375	Perf. Count 1.	Perf. Count. 2	…
Kernel 1	🟥	🟥	🟥	🟥	🟩	🟩	🟩
Kernel 2	🟥	🟥	🟥	🟥	🟩	🟩	🟩
…..	🟥	🟥	🟥	🟥	🟩	🟩	🟩
	🟥	🟥	🟥	🟥	🟩	🟩	🟩
Kernel N	🟥	🟥	🟥	🟥	🟩	🟩	🟩

Execution Times/Power

Performance Counter Values gathered on base hardware configuration

Outline

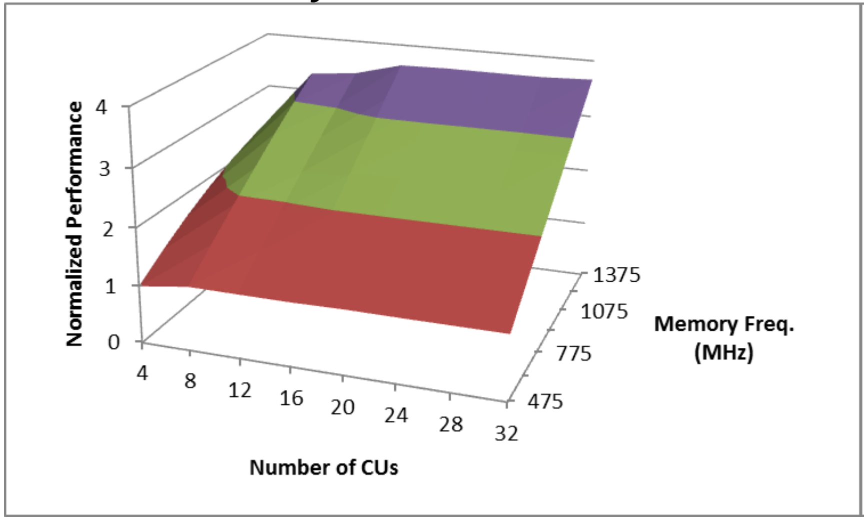
- Goals
- Model Overview
- Model Construction
- Results

Model Construction

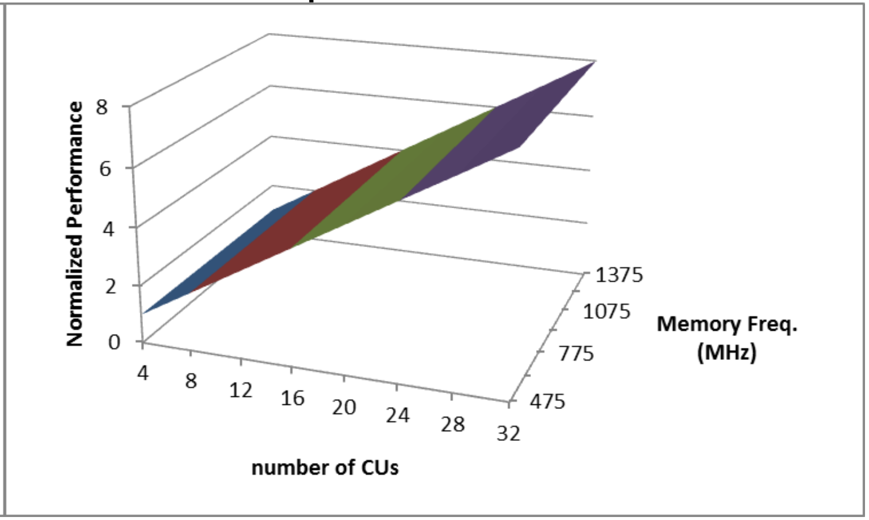
- Phase 1: Form clusters of training kernels that scale similarly
- Phase 2: Build a classifier to map kernel performance counter values to specific clusters

Kernel Scaling Behaviors

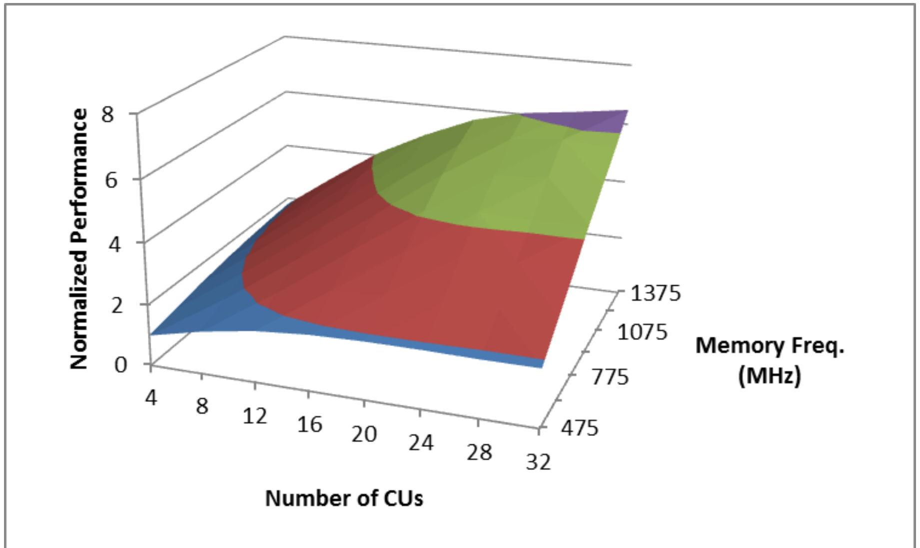
Memory Bound



Compute Bound



Balanced

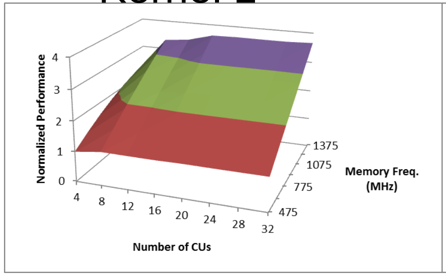


- Found many other patterns during this study

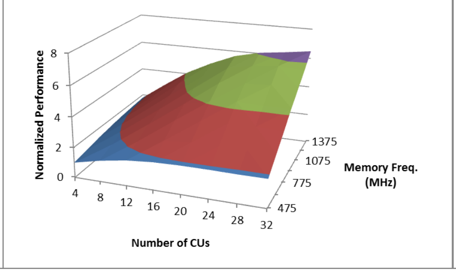
Phase 1:Clustering

Training Set

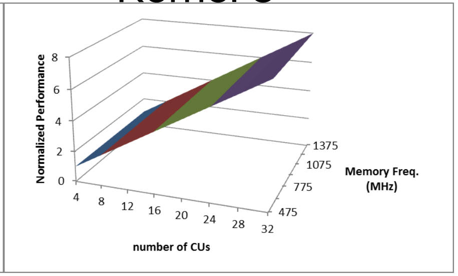
Kernel 1



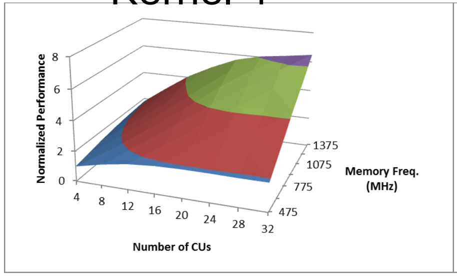
Kernel 2



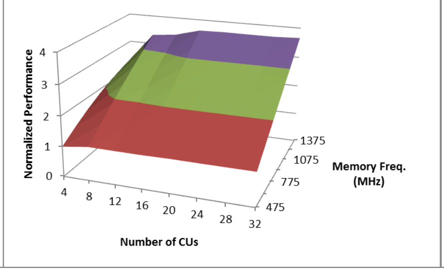
Kernel 3



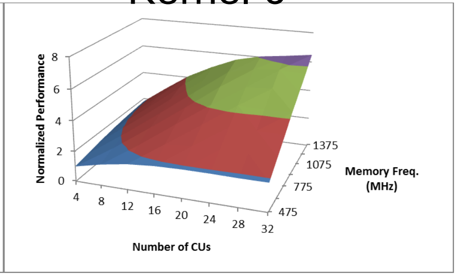
Kernel 4



Kernel 5



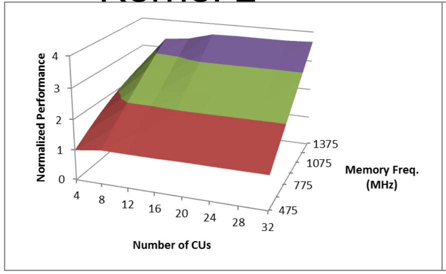
Kernel 6



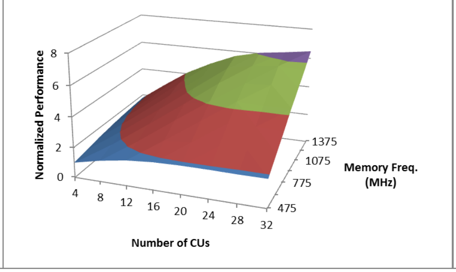
Phase 1:Clustering

Training Set

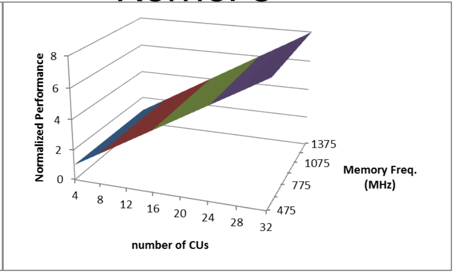
Kernel 1



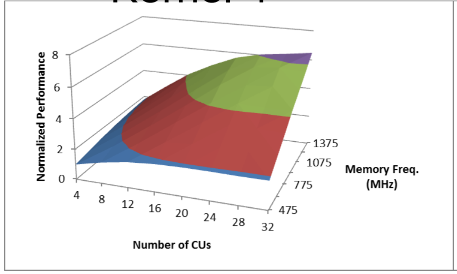
Kernel 2



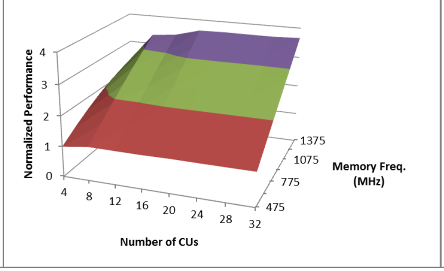
Kernel 3



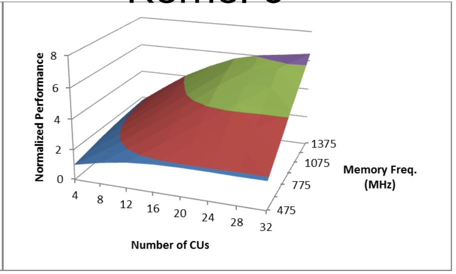
Kernel 4



Kernel 5



Kernel 6



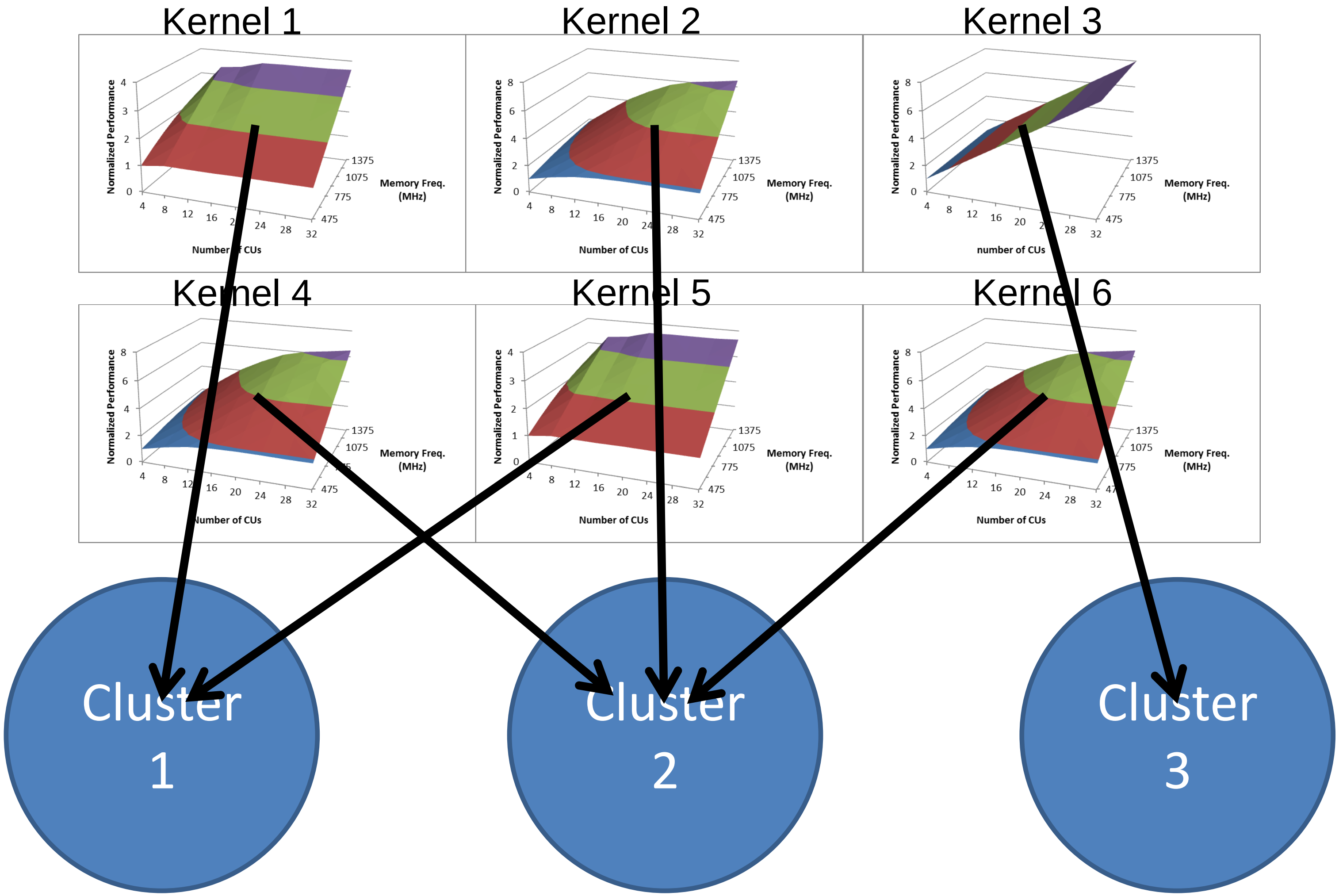
Cluster 1

Cluster 2

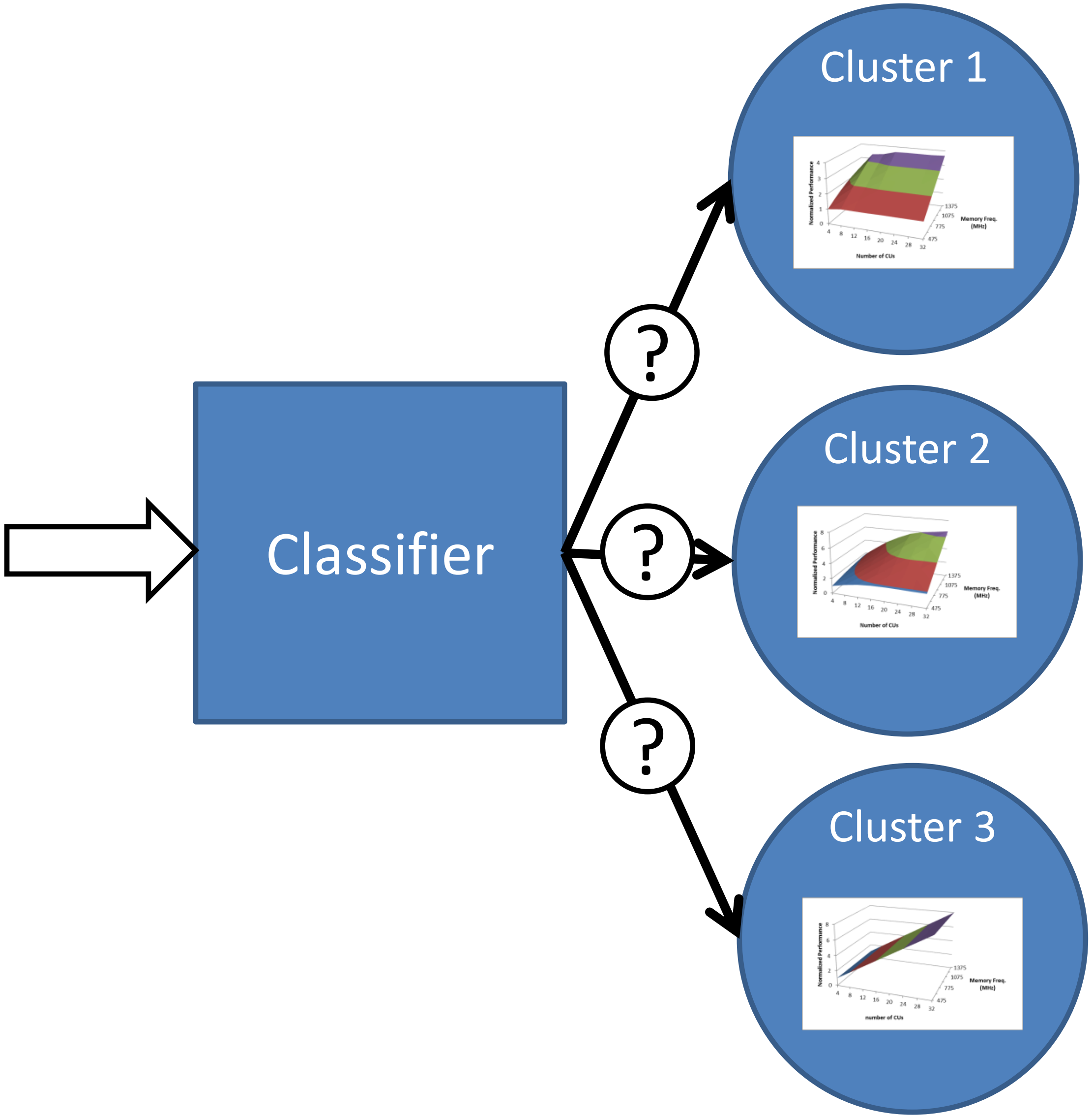
Cluster 3

Phase 1:Clustering

Training Set



Phase 2:Classification



Performance Counter Values (from base configuration)

Classifier

?

?

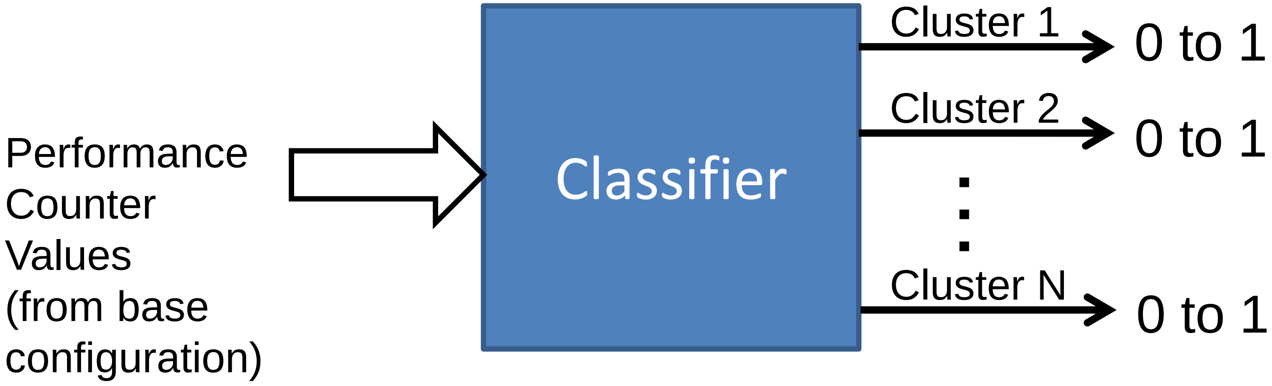
?

Cluster 1

Cluster 2

Cluster 3

Classifier



Performance Counter Values (from base configuration)

Classifier

Cluster 1 → 0 to 1

Cluster 2 → 0 to 1

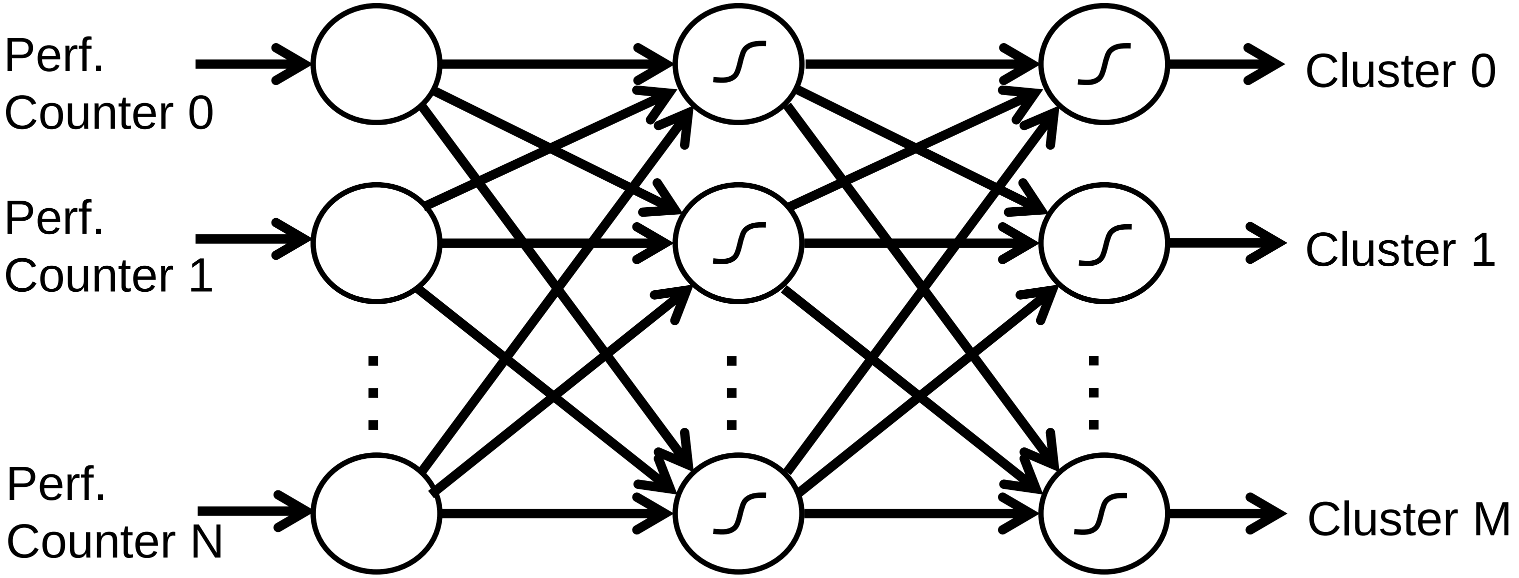
⋮

Cluster N → 0 to 1

- Inputs:
  - Performance counter values
- Outputs:
  - One output per cluster
  - Output values between 0 and 1
  - Cluster with highest output is chosen
  - Ideally a one hot encoding at outputs

Classifier: Neural Network Topology

- 3 layer, fully connected network
  - Input layer: linear
    - Number of neurons equals number of features
  - Hidden layer: sigmoid
    - Number of neurons equals number of clusters
  - Output layer: sigmoid
    - Number of neurons equals number of clusters



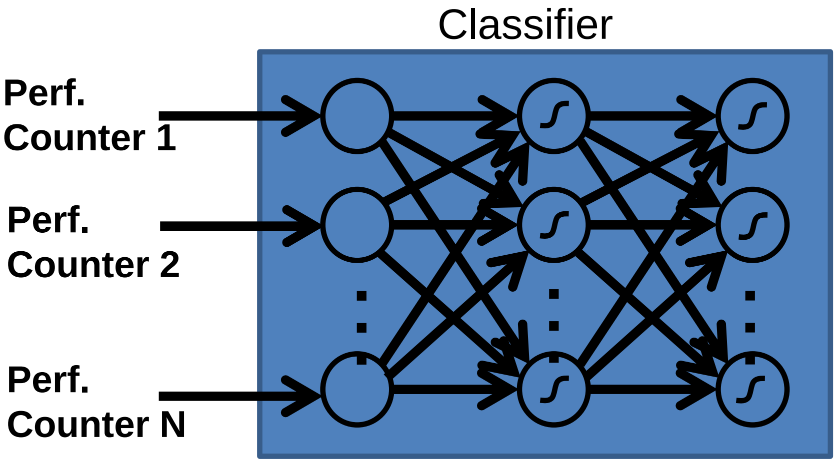
Putting It All Together

**Perf.
Counter 1**

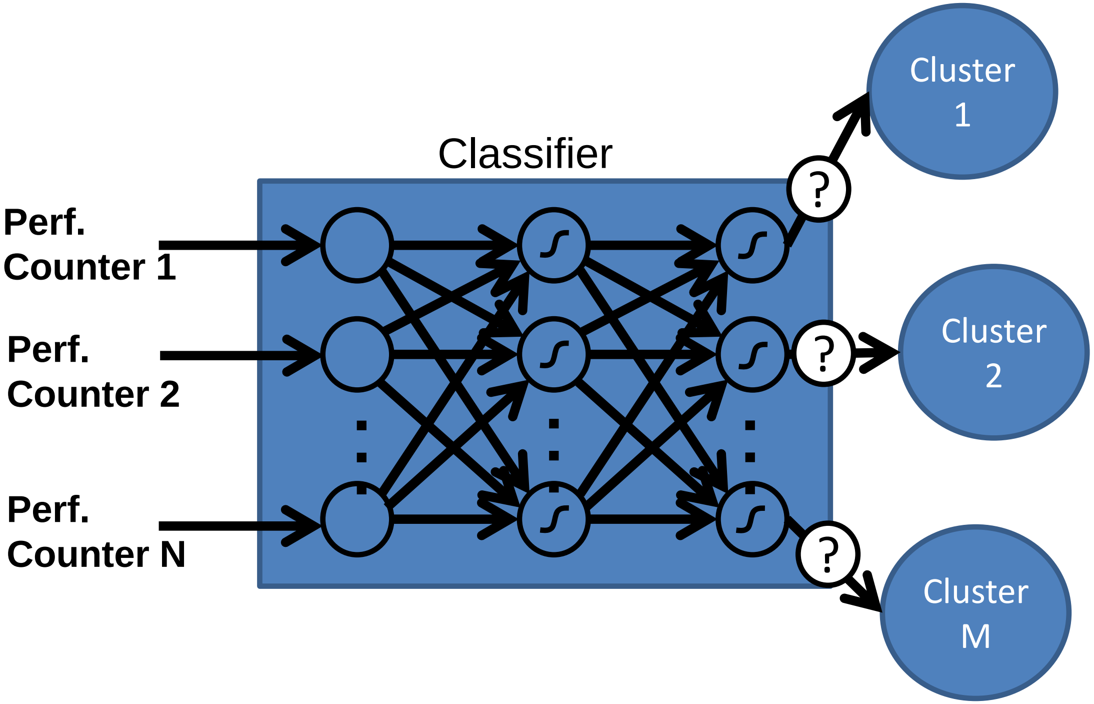
**Perf.
Counter 2**

**Perf.
Counter N**

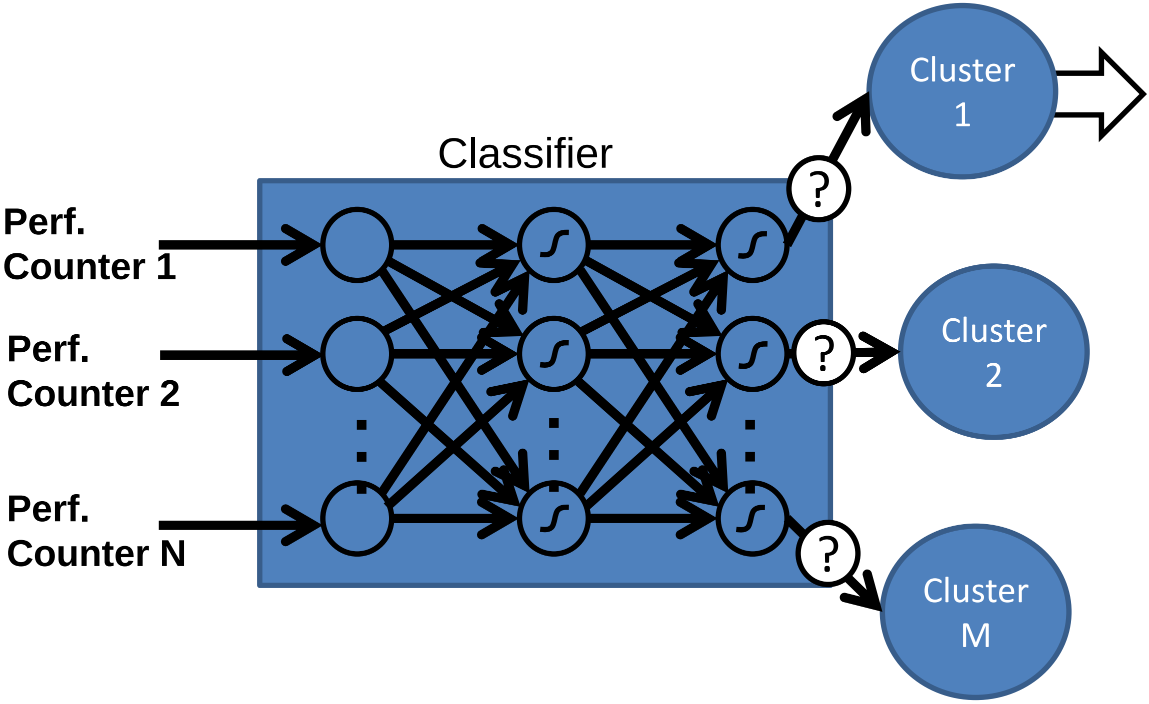
Putting It All Together

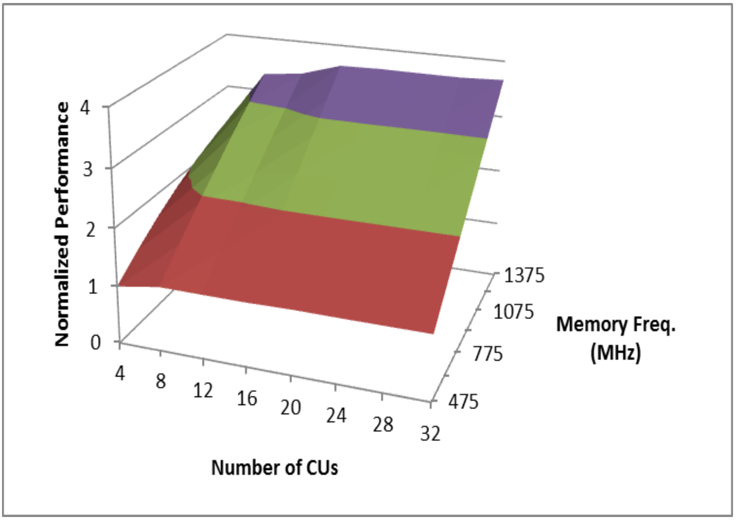


Putting It All Together

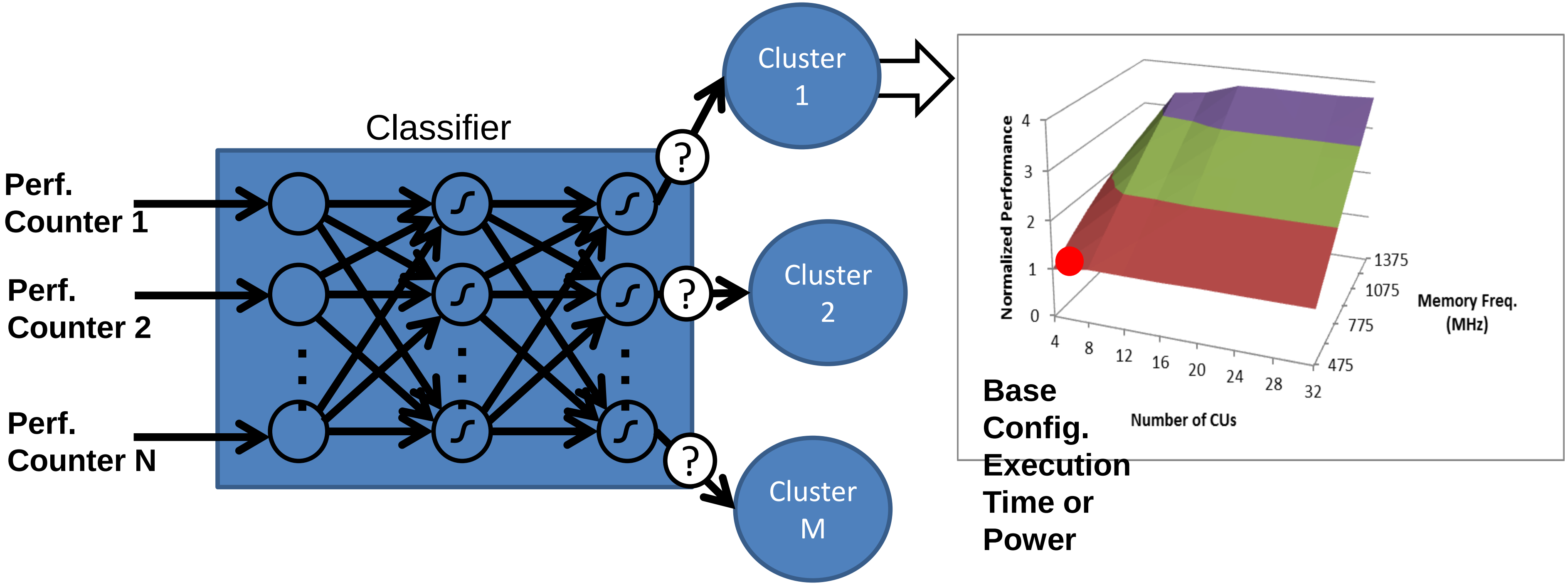


Putting It All Together





Putting It All Together



Perf. Counter 1

Perf. Counter 2

Perf. Counter N

Classifier

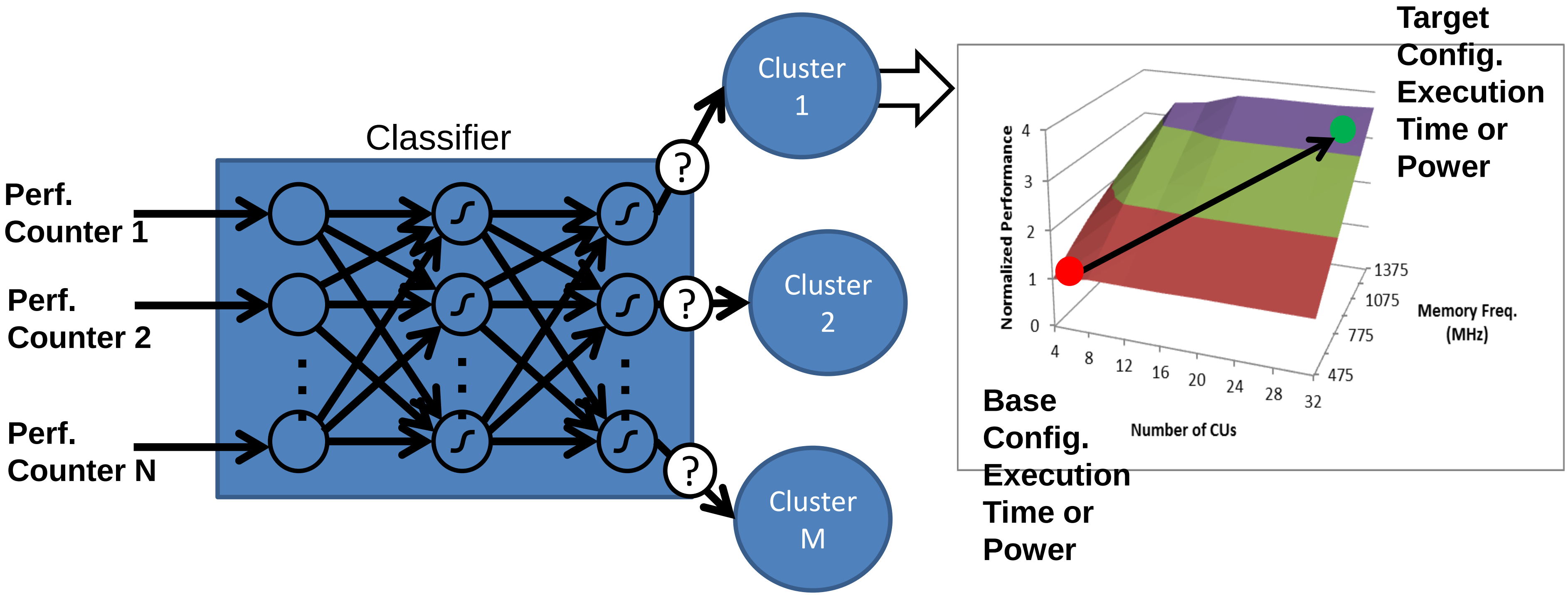
Cluster 1

Cluster 2

Cluster M

Base Config. Execution Time or Power

Putting It All Together



Classifier

Perf. Counter 1

Perf. Counter 2

Perf. Counter N

Cluster 1

Cluster 2

Cluster M

Target Config. Execution Time or Power

Base Config. Execution Time or Power

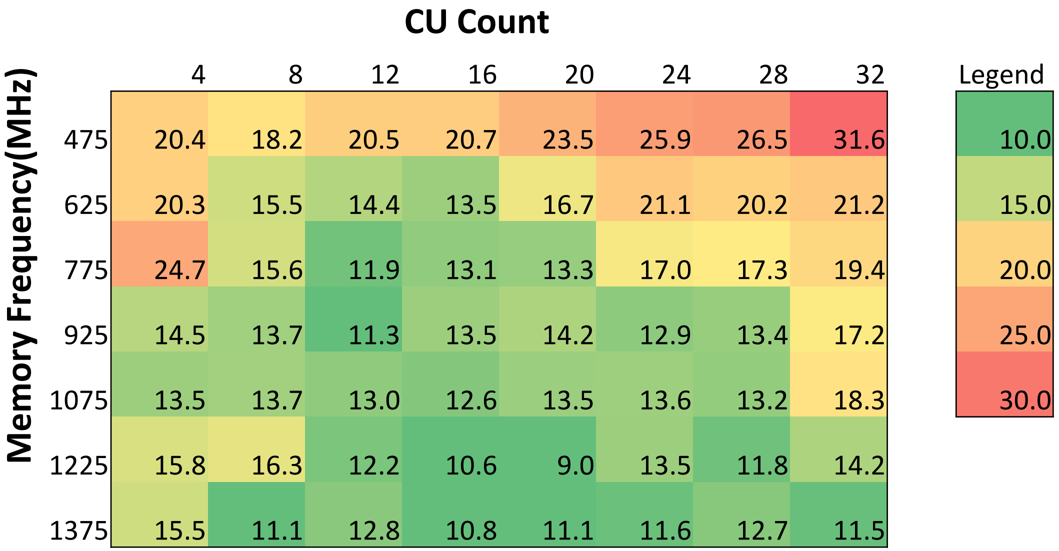
Outline

- Goals
- Model Overview
- Model Construction
- Results

Experimental Setup

- Measurements gathered on a AMD Radeon HD 7970 GPU
- 8 CU settings:
  - 4, 8, 12, 16, 20, 24, 28, 32
- 8 Engine Frequencies:
  - 300, 400, 500, 600, 700, 800, 900, 1000 (MHz)
- 7 Memory Frequencies:
  - 475, 625, 775, 925, 1075, 1225, 1375 (MHz)
- 448(8x8x7) possible hardware configurations
- 108 OpenCL kernels:
  - 86 kernels (80%) for training
  - 22 kernels (20%) for validation

Accuracy vs. Base Configuration



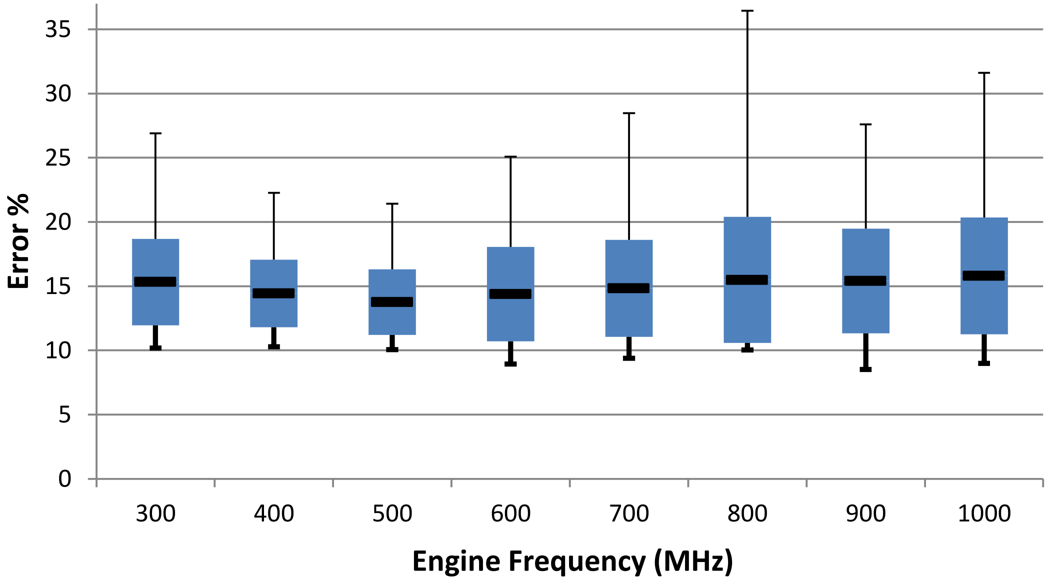
CU Count

Memory Frequency(MHz)	4	8	12	16	20	24	28	32
475	20.4	18.2	20.5	20.7	23.5	25.9	26.5	31.6
625	20.3	15.5	14.4	13.5	16.7	21.1	20.2	21.2
775	24.7	15.6	11.9	13.1	13.3	17.0	17.3	19.4
925	14.5	13.7	11.3	13.5	14.2	12.9	13.4	17.2
1075	13.5	13.7	13.0	12.6	13.5	13.6	13.2	18.3
1225	15.8	16.3	12.2	10.6	9.0	13.5	11.8	14.2
1375	15.5	11.1	12.8	10.8	11.1	11.6	12.7	11.5

Legend
10.0
15.0
20.0
25.0
30.0

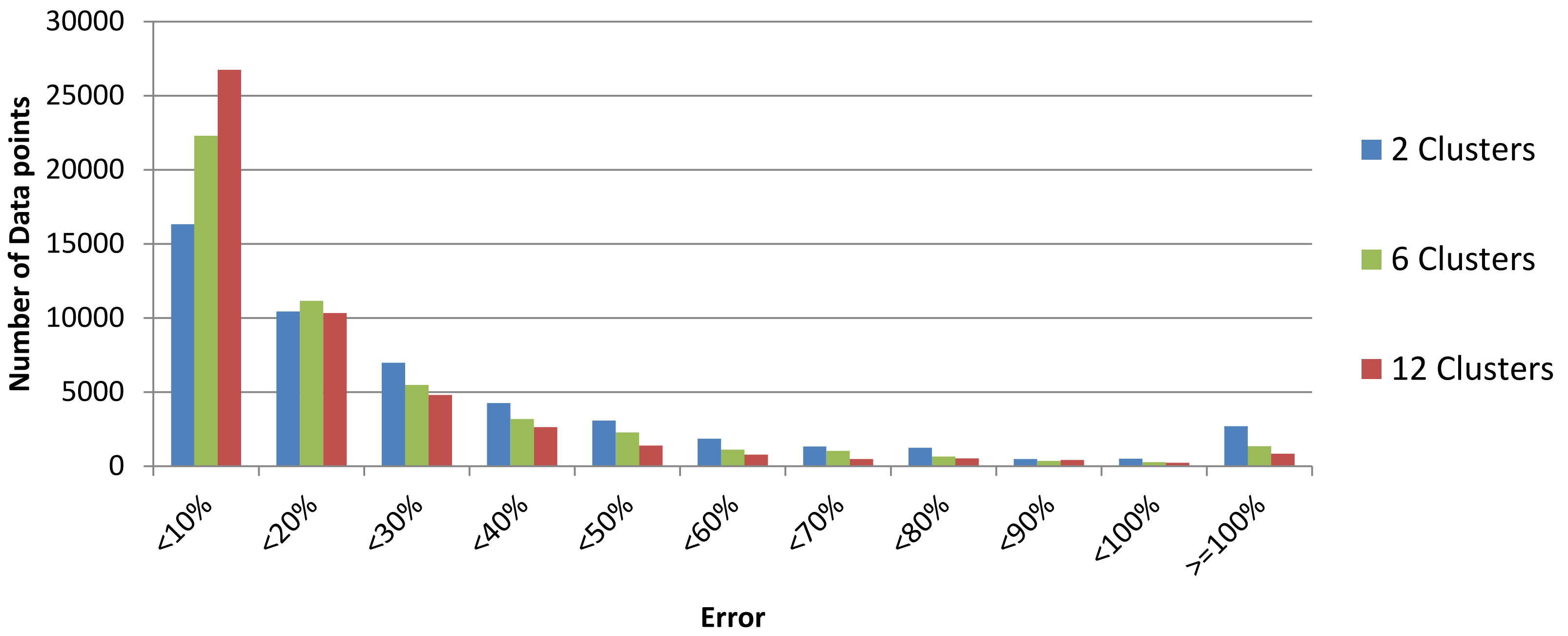
- Base configuration engine frequency fixed at 1000 MHz
- 12 Clusters
- Each entry is the average error of all validation kernels on all 447 possible target configurations (22 kernels x 447 target configs = 9834 predictions)
- Error higher when base configurations has an unbalanced compute to bandwidth ratio

Accuracy vs. Base Configuration



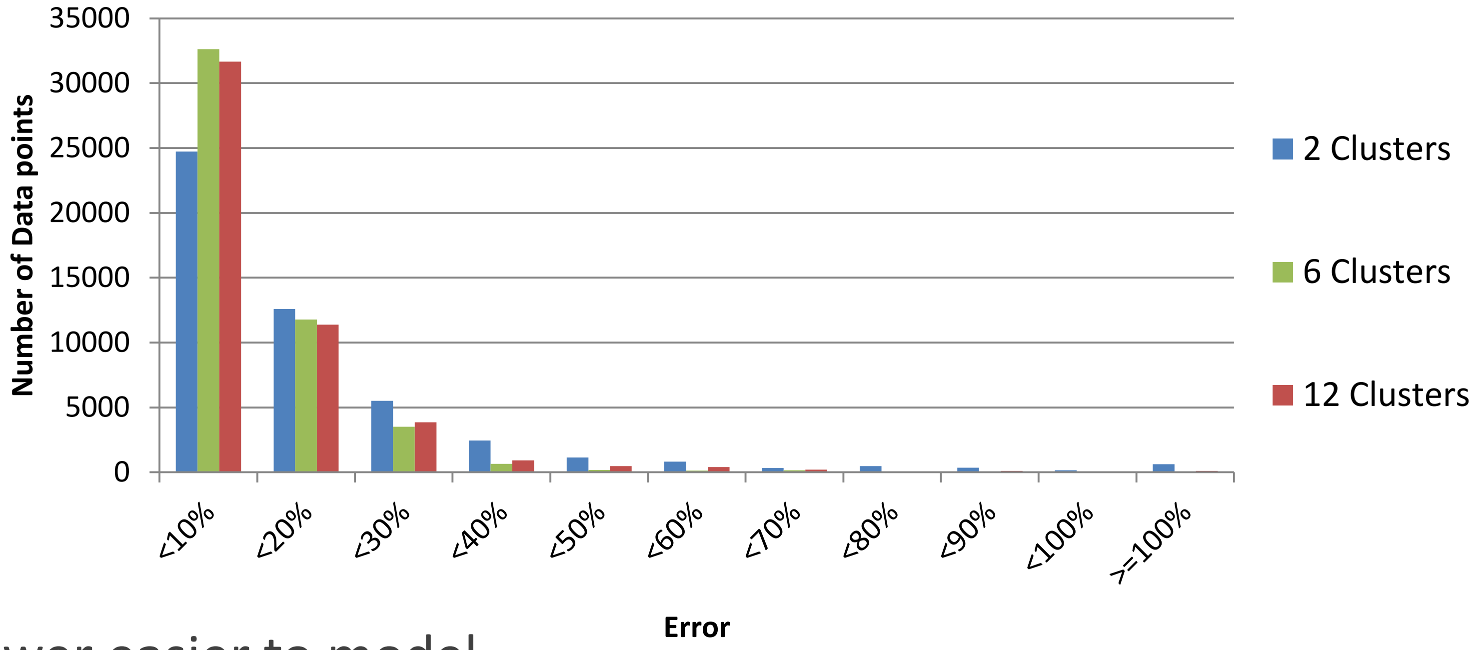
- Lowest error at 500 MHz engine frequency
  - Avg: 13.7%
  - Standard deviation: 2.6%
  - Max: 21.4%
  - Min: 10.1%

Performance Error Distribution



- 447 target configurations
- 22 validation kernels
- 5 base configurations:
  - 32.300.475, 32.300.1375, 32.700.925, 32.1000.475, 32.1000.1375
- 447x22x5 = 49170 total data points

Power Error Distribution



- Power easier to model
- Modeling a model
- Average Error:
  - 2 Clusters: 11.4%
  - 6 Clusters: 9.1%
  - 12 Clusters: 10.1%

Summary

- GPU power and performance models
  - Constructed with K-means clustering and neural networks
- Performance model average error:
  - Around 10% for the best base hardware configurations
- Power model average error:
  - Around 10%
- Less than a millisecond for each prediction

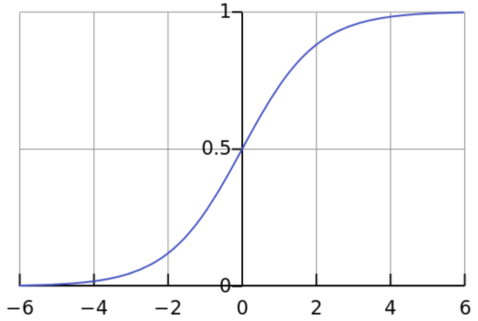
Questions

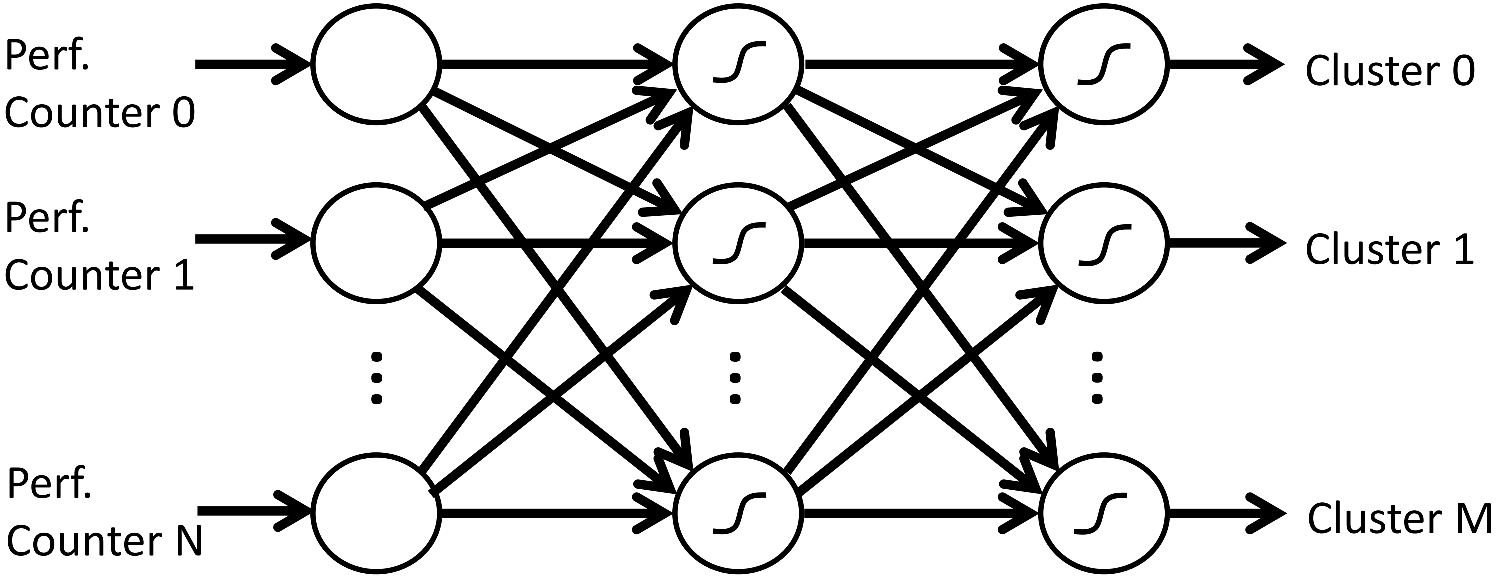
?

Backup Slides

Classifier: Neural Network Topology

- 3 layer, fully connected network
  - Input layer: linear
    - Number of neurons equals number of features
  - Hidden layer: sigmoid
    - Number of neurons equals number of clusters
  - Output layer: sigmoid
    - Number of neurons equals number of clusters



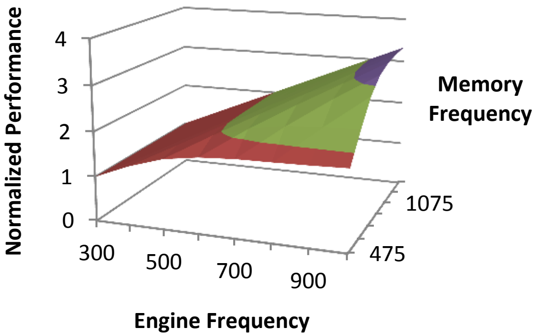


Execution Time Scaling Values

Engine Freq. (MHz)

1000	$t_{3,0}$	$t_{3,1}$	$t_{3,2}$	$t_{3,3}$
800	$t_{2,0}$	$t_{2,1}$	$t_{2,2}$	$t_{2,3}$
600	$t_{1,0}$	$t_{1,1}$	$t_{1,2}$	$t_{1,3}$
400	$t_{0,0}$	$t_{0,1}$	$t_{0,2}$	$t_{0,3}$
	400	600	800	1000

Mem. Freq. (MHz)



- Per kernel in training set
- Fixed CU count in this example

Execution Time Scaling Values

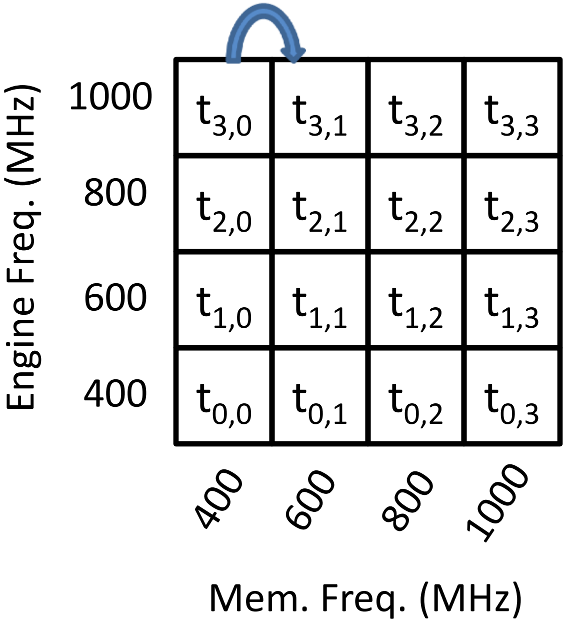
Engine Freq. (MHz)				
1000	$t_{3,0}$	$t_{3,1}$	$t_{3,2}$	$t_{3,3}$
800	$t_{2,0}$	$t_{2,1}$	$t_{2,2}$	$t_{2,3}$
600	$t_{1,0}$	$t_{1,1}$	$t_{1,2}$	$t_{1,3}$
400	$t_{0,0}$	$t_{0,1}$	$t_{0,2}$	$t_{0,3}$
	400	600	800	1000

Mem. Freq. (MHz)

- Per kernel in training set
- Fixed CU count in this example

Execution Time Scaling Values

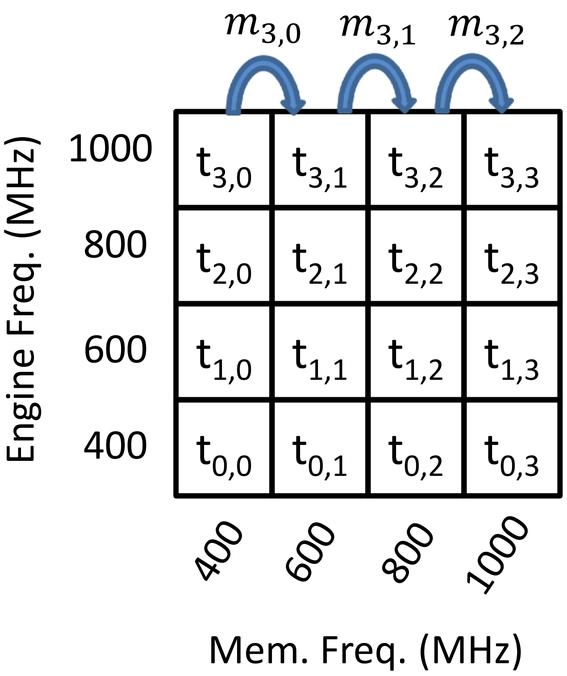
$$m_{3,0} = \frac{t_{3,1}}{t_{3,0}}$$



1000	$t_{3,0}$	$t_{3,1}$	$t_{3,2}$	$t_{3,3}$
800	$t_{2,0}$	$t_{2,1}$	$t_{2,2}$	$t_{2,3}$
600	$t_{1,0}$	$t_{1,1}$	$t_{1,2}$	$t_{1,3}$
400	$t_{0,0}$	$t_{0,1}$	$t_{0,2}$	$t_{0,3}$
	400	600	800	1000

- Per kernel in training set
- Fixed CU count in this example

Execution Time Scaling Values

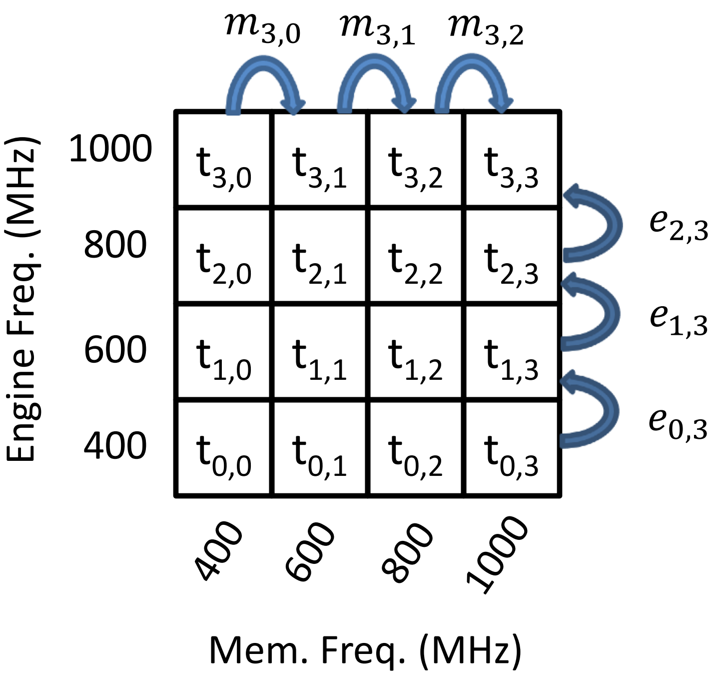


1000	$t_{3,0}$	$t_{3,1}$	$t_{3,2}$	$t_{3,3}$
800	$t_{2,0}$	$t_{2,1}$	$t_{2,2}$	$t_{2,3}$
600	$t_{1,0}$	$t_{1,1}$	$t_{1,2}$	$t_{1,3}$
400	$t_{0,0}$	$t_{0,1}$	$t_{0,2}$	$t_{0,3}$
	400	600	800	1000

$$t_{3,3} = t_{3,0} \prod_{j=0}^{2} m_{3,j}$$

- Per kernel in training set
- Fixed CU count in this example

Execution Time Scaling Values



	400	600	800	1000
1000	$t_{3,0}$	$t_{3,1}$	$t_{3,2}$	$t_{3,3}$
800	$t_{2,0}$	$t_{2,1}$	$t_{2,2}$	$t_{2,3}$
600	$t_{1,0}$	$t_{1,1}$	$t_{1,2}$	$t_{1,3}$
400	$t_{0,0}$	$t_{0,1}$	$t_{0,2}$	$t_{0,3}$

$$t_{3,3} = t_{3,0} \prod_{j=0}^{2} m_{3,j}$$

- Per kernel in training set
- Fixed CU count in this example

Execution Time Scaling Values

Engine Freq. (MHz)				
1000	$t_{3,0}$	$t_{3,1}$	$t_{3,2}$	$t_{3,3}$
800	$t_{2,0}$	$t_{2,1}$	$t_{2,2}$	$t_{2,3}$
600	$t_{1,0}$	$t_{1,1}$	$t_{1,2}$	$t_{1,3}$
400	$t_{0,0}$	$t_{0,1}$	$t_{0,2}$	$t_{0,3}$
	400	600	800	1000

Mem. Freq. (MHz)

$$m_{i,j} = \frac{t_{i,j+1}}{t_{i,j}}$$

Engine Freq. (MHz)			
1000	$m_{3,0}$	$m_{3,1}$	$m_{3,2}$
800	$m_{2,0}$	$m_{2,1}$	$m_{2,2}$
600	$m_{1,0}$	$m_{1,1}$	$m_{1,2}$
400	$m_{0,0}$	$m_{0,1}$	$m_{0,2}$
	400 to 600	600 to 800	800 to 1000

Mem. Freq. (MHz)

$$e_{i,j} = \frac{t_{i+1,j}}{t_{i,j}}$$

Engine Freq. (MHz)				
800 to 1000	$e_{2,0}$	$e_{2,1}$	$e_{2,2}$	$e_{2,3}$
600 to 800	$e_{1,0}$	$e_{1,1}$	$e_{1,2}$	$e_{1,3}$
400 to 600	$e_{0,0}$	$e_{0,1}$	$e_{0,2}$	$e_{0,3}$
	400	600	800	1000

Mem. Freq. (MHz)

- Per kernel in training set
- Fixed CU count in this example

K-means Clustering: the view from 10,000 feet

- Each kernel has a feature vector (x vector)

Kernel 1
$[x_0, x_1, x_2, \dots x_n]$

Kernel 2
$[x_0, x_1, x_2, \dots x_n]$

Kernel 3
$[x_0, x_1, x_2, \dots x_n]$

$\vdots$

Kernel K
$[x_0, x_1, x_2, \dots x_n]$

K-means Clustering: the view from 10,000 feet

- Each kernel has a feature vector (x vector)

Kernel 1
$[x_0, x_1, x_2, \dots x_n]$

Kernel 2
$[x_0, x_1, x_2, \dots x_n]$

Kernel 3
$[x_0, x_1, x_2, \dots x_n]$

⋮

Kernel K
$[x_0, x_1, x_2, \dots x_n]$

Cluster 1
$[y_0, y_1, y_2, \dots y_n]$

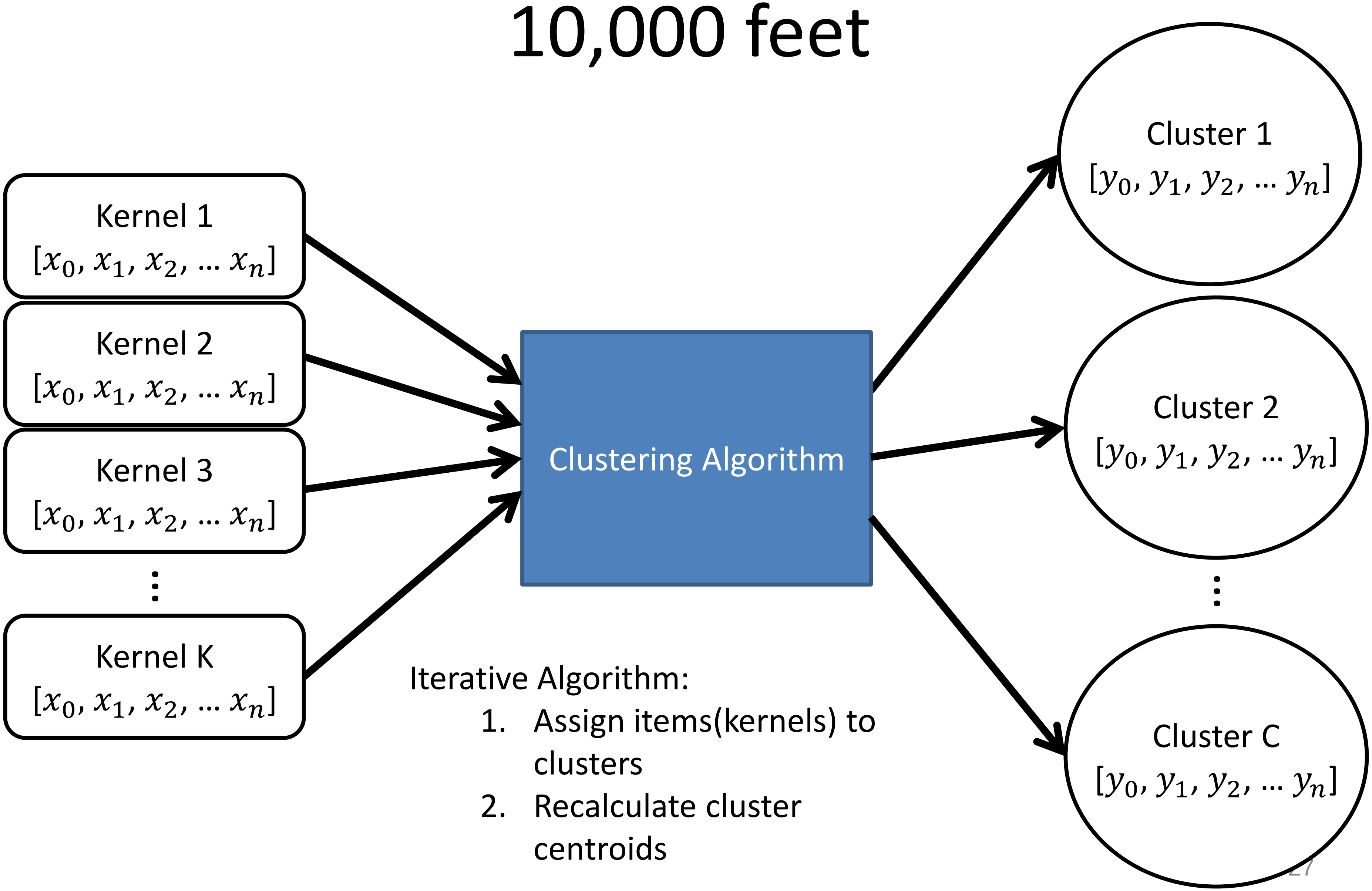
Cluster 2
$[y_0, y_1, y_2, \dots y_n]$

⋮

Cluster C
$[y_0, y_1, y_2, \dots y_n]$

- Each kernel has a cluster has a centroid (y vector)

K-means Clustering: the view from 10,000 feet



Kernel 1
$[x_0, x_1, x_2, \dots x_n]$

Kernel 2
$[x_0, x_1, x_2, \dots x_n]$

Kernel 3
$[x_0, x_1, x_2, \dots x_n]$

⋮

Kernel K
$[x_0, x_1, x_2, \dots x_n]$

Clustering Algorithm

Cluster 1
$[y_0, y_1, y_2, \dots y_n]$

Cluster 2
$[y_0, y_1, y_2, \dots y_n]$

⋮

Cluster C
$[y_0, y_1, y_2, \dots y_n]$

Iterative Algorithm:

1. Assign items(kernels) to clusters
2. Recalculate cluster centroids

K-means Clustering: the view from 10,000 feet

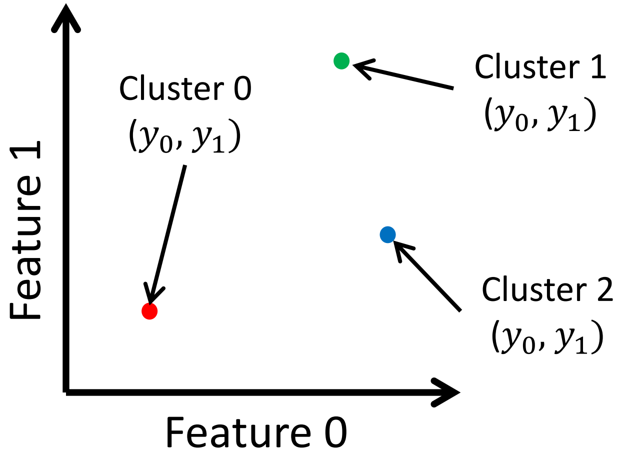
Kernel 1
$[x_0, x_1, x_2, \dots x_n]$

Kernel 2
$[x_0, x_1, x_2, \dots x_n]$

Kernel 3
$[x_0, x_1, x_2, \dots x_n]$

⋮

Kernel K
$[x_0, x_1, x_2, \dots x_n]$



Iterative Algorithm:

1. Assign items(kernels) to clusters
2. Recalculate cluster centroids

Cluster 1
$[y_0, y_1, y_2, \dots y_n]$

Cluster 2
$[y_0, y_1, y_2, \dots y_n]$

⋮

Cluster C
$[y_0, y_1, y_2, \dots y_n]$

K-means Clustering: the view from 10,000 feet

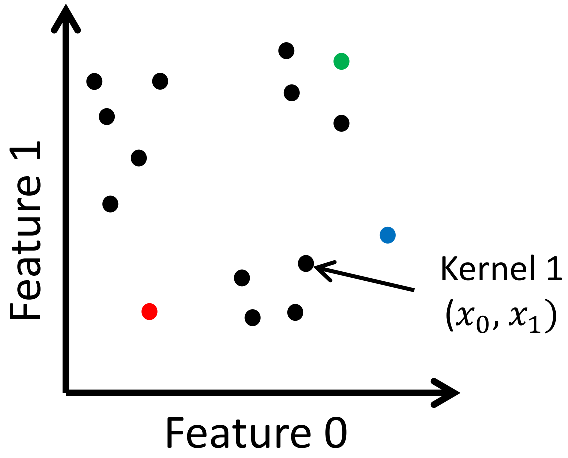
Kernel 1
$[x_0, x_1, x_2, \ldots x_n]$

Kernel 2
$[x_0, x_1, x_2, \ldots x_n]$

Kernel 3
$[x_0, x_1, x_2, \ldots x_n]$

⋮

Kernel K
$[x_0, x_1, x_2, \ldots x_n]$



Iterative Algorithm:

1. Assign items(kernels) to clusters
2. Recalculate cluster centroids

Cluster 1
$[y_0, y_1, y_2, \ldots y_n]$

Cluster 2
$[y_0, y_1, y_2, \ldots y_n]$

⋮

Cluster C
$[y_0, y_1, y_2, \ldots y_n]$

K-means Clustering: the view from 10,000 feet

Kernel 1
$[x_0, x_1, x_2, \ldots x_n]$

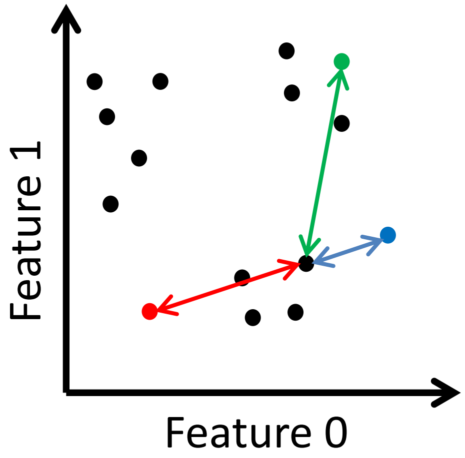
Kernel 2
$[x_0, x_1, x_2, \ldots x_n]$

Kernel 3
$[x_0, x_1, x_2, \ldots x_n]$

⋮

Kernel K
$[x_0, x_1, x_2, \ldots x_n]$

$$distance = \sqrt{(x_0 - y_0)^2 + (x_1 - y_1)^2 + \cdots + (x_n - y_n)^2}$$



Iterative Algorithm:

1. Assign items(kernels) to clusters
2. Recalculate cluster centroids

Cluster 1
$[y_0, y_1, y_2, \ldots y_n]$

Cluster 2
$[y_0, y_1, y_2, \ldots y_n]$

⋮

Cluster C
$[y_0, y_1, y_2, \ldots y_n]$

K-means Clustering: the view from 10,000 feet

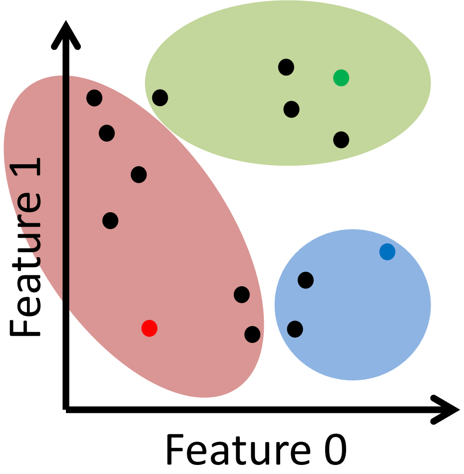
Kernel 1
$[x_0, x_1, x_2, \ldots x_n]$

Kernel 2
$[x_0, x_1, x_2, \ldots x_n]$

Kernel 3
$[x_0, x_1, x_2, \ldots x_n]$

⋮

Kernel K
$[x_0, x_1, x_2, \ldots x_n]$



Iterative Algorithm:

1. Assign items(kernels) to clusters
2. Recalculate cluster centroids

Cluster 1
$[y_0, y_1, y_2, \ldots y_n]$

Cluster 2
$[y_0, y_1, y_2, \ldots y_n]$

⋮

Cluster C
$[y_0, y_1, y_2, \ldots y_n]$

K-means Clustering: the view from 10,000 feet

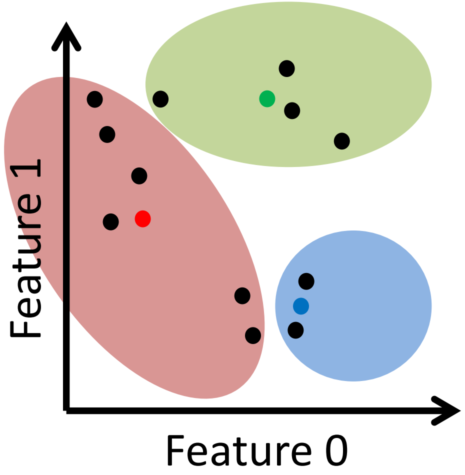
Kernel 1
$[x_0, x_1, x_2, \dots x_n]$

Kernel 2
$[x_0, x_1, x_2, \dots x_n]$

Kernel 3
$[x_0, x_1, x_2, \dots x_n]$

⋮

Kernel K
$[x_0, x_1, x_2, \dots x_n]$



Iterative Algorithm:

1. Assign items(kernels) to clusters
2. Recalculate cluster centroids

Cluster 1
$[y_0, y_1, y_2, \dots y_n]$

Cluster 2
$[y_0, y_1, y_2, \dots y_n]$

⋮

Cluster C
$[y_0, y_1, y_2, \dots y_n]$

K-means Clustering: the view from 10,000 feet

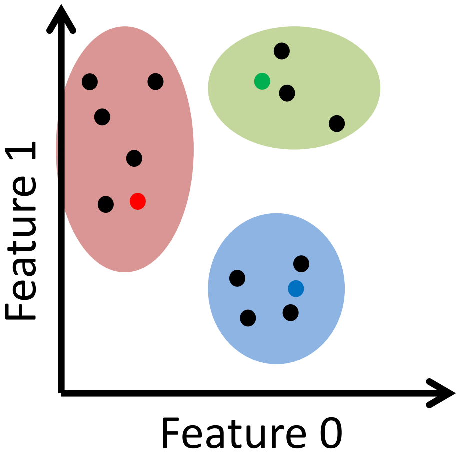
Kernel 1
$[x_0, x_1, x_2, \ldots x_n]$

Kernel 2
$[x_0, x_1, x_2, \ldots x_n]$

Kernel 3
$[x_0, x_1, x_2, \ldots x_n]$

⋮

Kernel K
$[x_0, x_1, x_2, \ldots x_n]$



Iterative Algorithm:

1. Assign items(kernels) to clusters
2. Recalculate cluster centroids

Cluster 1
$[y_0, y_1, y_2, \ldots y_n]$

Cluster 2
$[y_0, y_1, y_2, \ldots y_n]$

⋮

Cluster C
$[y_0, y_1, y_2, \ldots y_n]$

K-means Clustering: the view from 10,000 feet

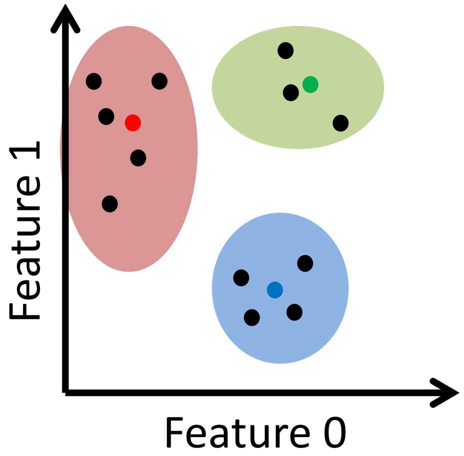
Kernel 1
$[x_0, x_1, x_2, \ldots x_n]$

Kernel 2
$[x_0, x_1, x_2, \ldots x_n]$

Kernel 3
$[x_0, x_1, x_2, \ldots x_n]$

⋮

Kernel K
$[x_0, x_1, x_2, \ldots x_n]$



Iterative Algorithm:

1. Assign items(kernels) to clusters
2. Recalculate cluster centroids

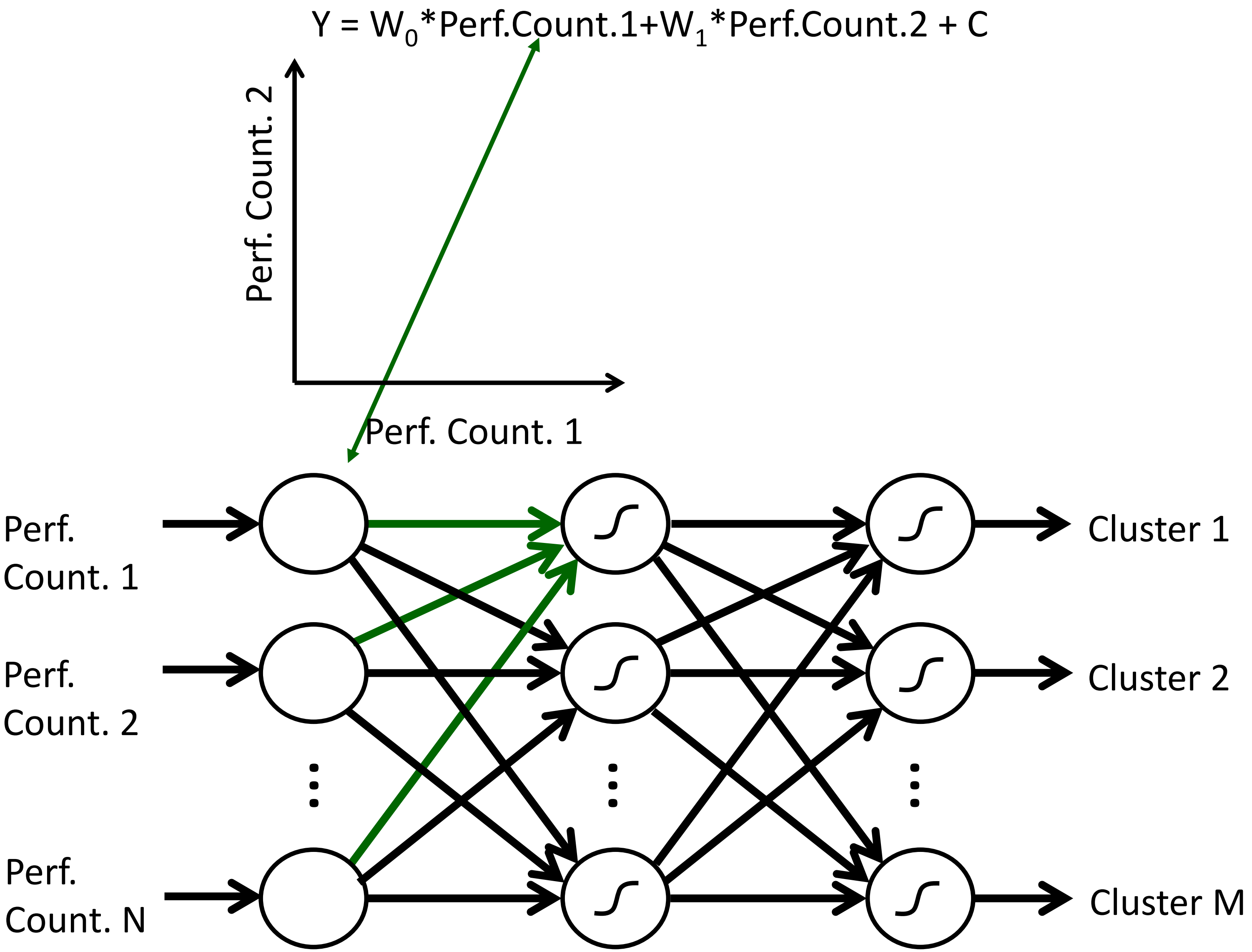
Cluster 1
$[y_0, y_1, y_2, \ldots y_n]$

Cluster 2
$[y_0, y_1, y_2, \ldots y_n]$

⋮

Cluster C
$[y_0, y_1, y_2, \ldots y_n]$

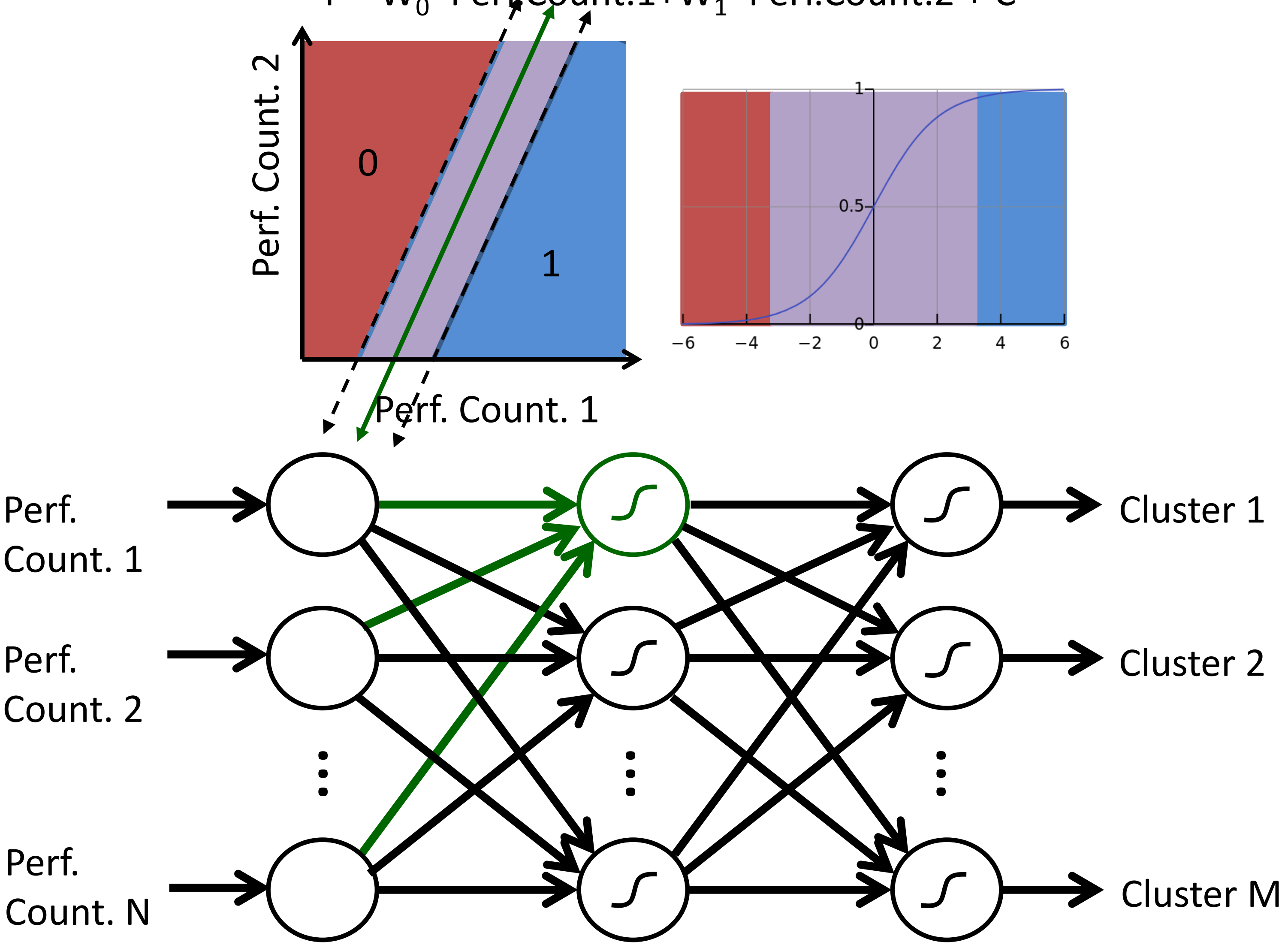
Neural Network



$$Y = W_0 * \text{Perf.Count.1} + W_1 * \text{Perf.Count.2} + C$$

Neural Network

$Y = W_0 * Perf.Count.1 + W_1 * Perf.Count.2 + C$



Perf. Count. 1

Perf. Count. 2

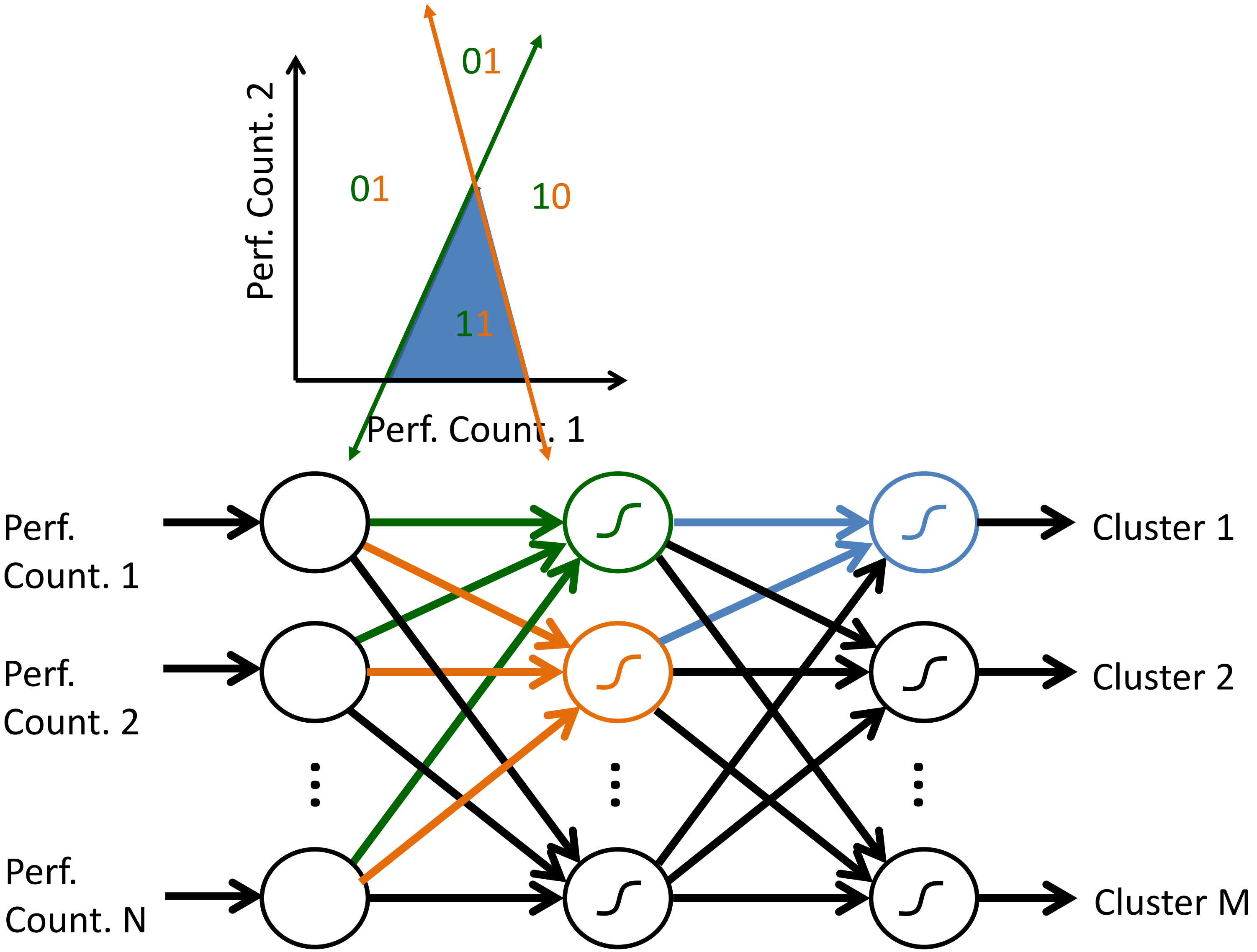
Perf. Count. N

Cluster 1

Cluster 2

Cluster M

Neural Network



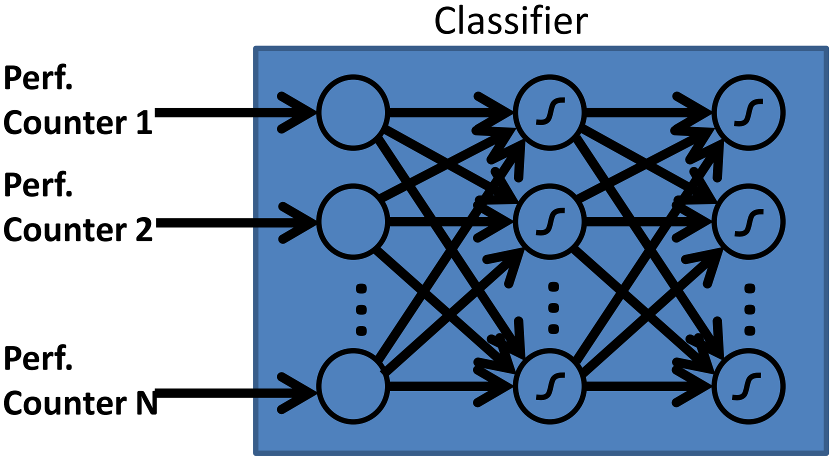
Putting It All Together

**Perf.
Counter 1**

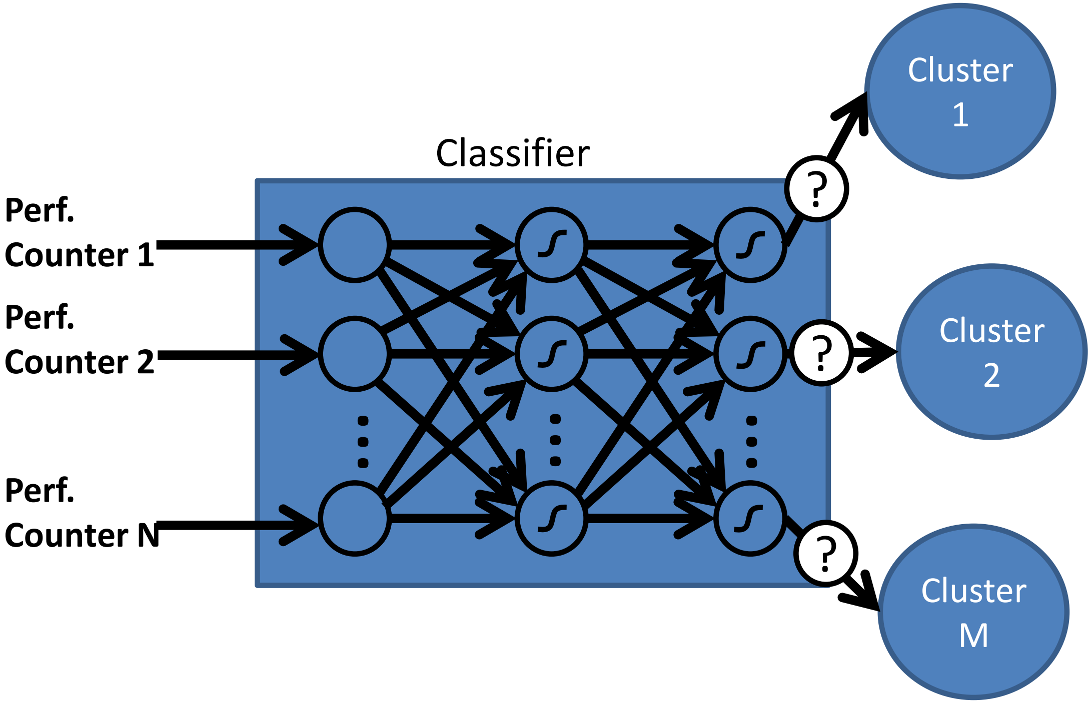
**Perf.
Counter 2**

**Perf.
Counter N**

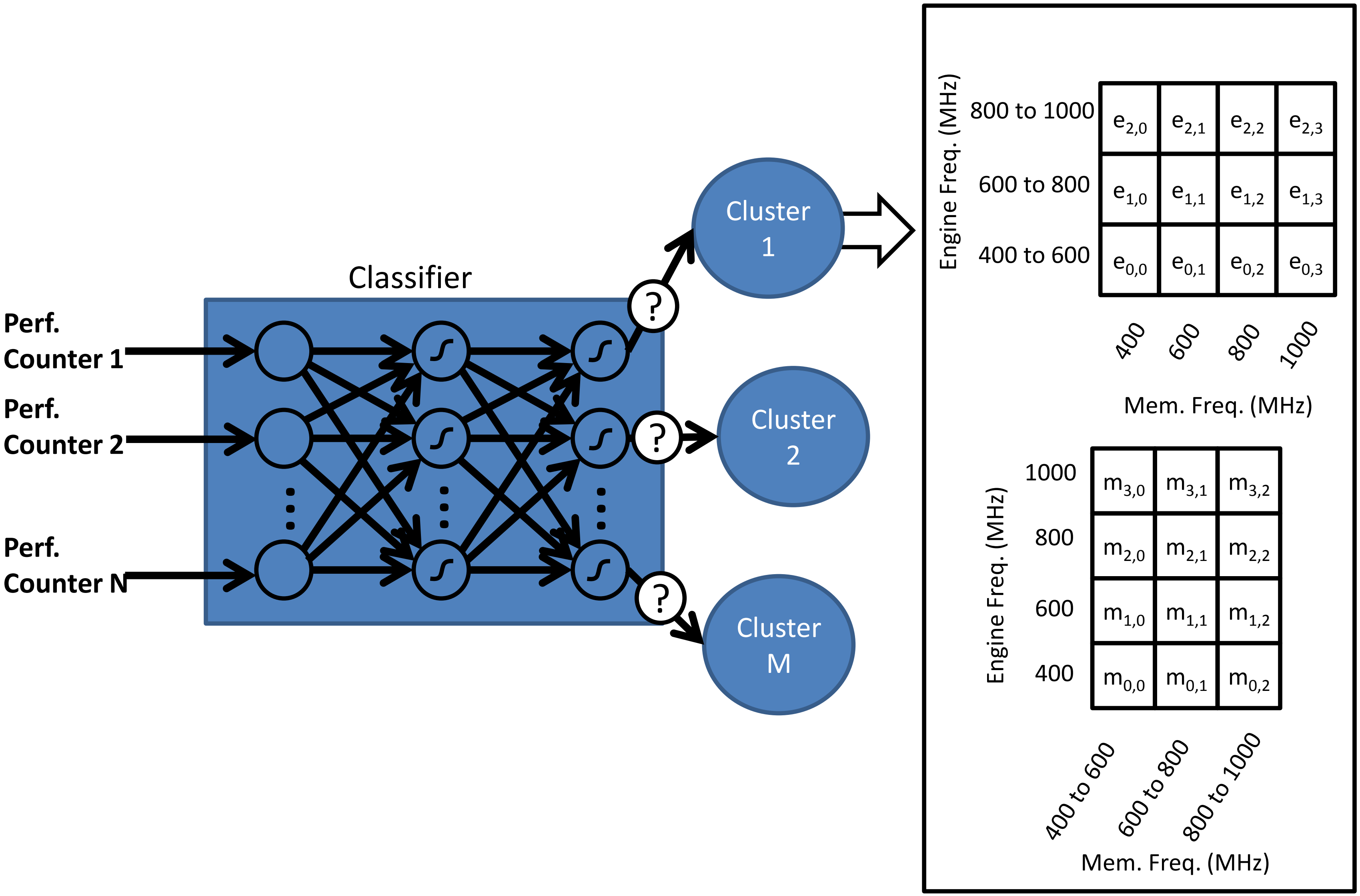
Putting It All Together



Putting It All Together



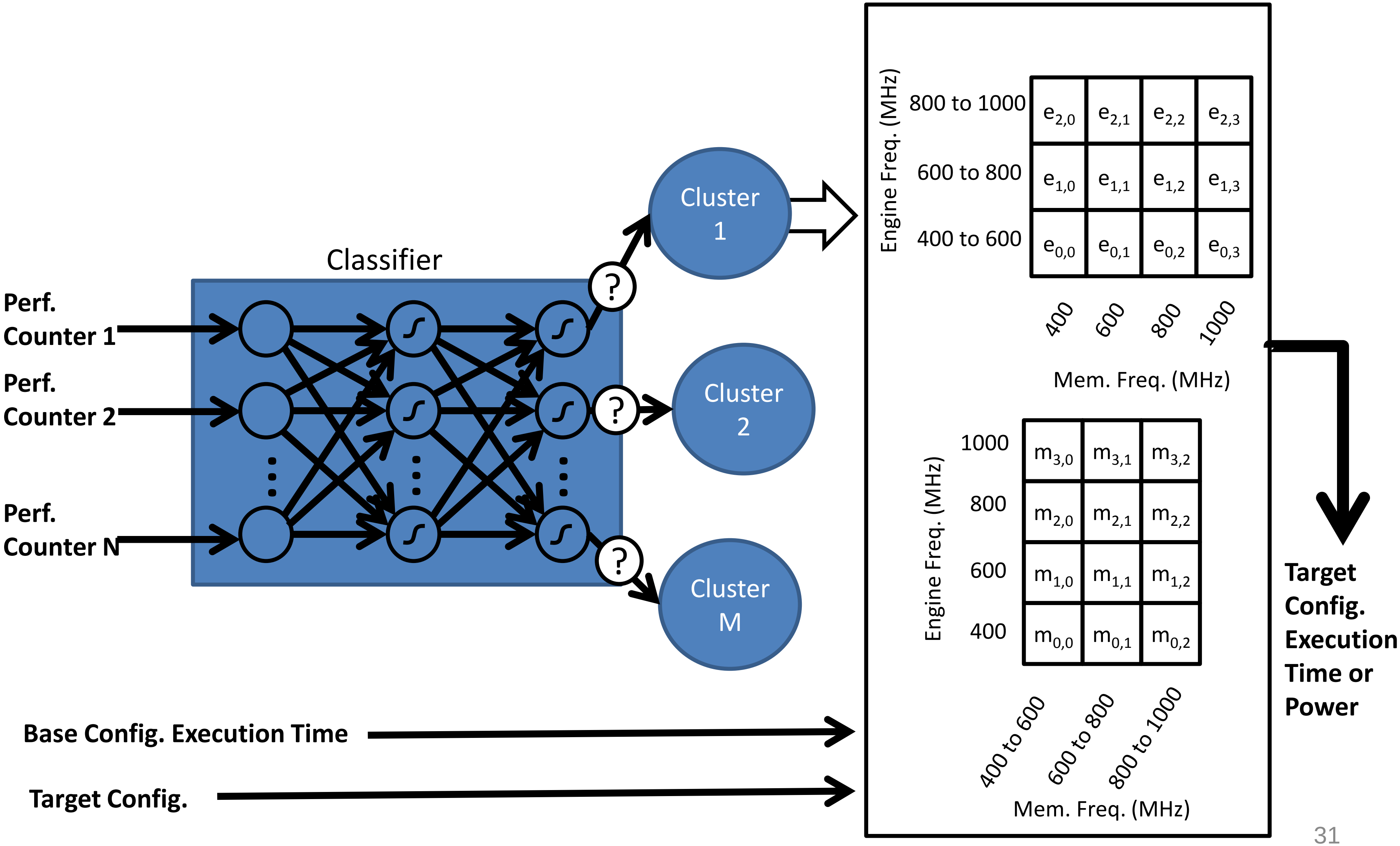
Putting It All Together



Engine Freq. (MHz)	400	600	800	1000
800 to 1000	$e_{2,0}$	$e_{2,1}$	$e_{2,2}$	$e_{2,3}$
600 to 800	$e_{1,0}$	$e_{1,1}$	$e_{1,2}$	$e_{1,3}$
400 to 600	$e_{0,0}$	$e_{0,1}$	$e_{0,2}$	$e_{0,3}$

Engine Freq. (MHz)	400 to 600	600 to 800	800 to 1000
1000	$m_{3,0}$	$m_{3,1}$	$m_{3,2}$
800	$m_{2,0}$	$m_{2,1}$	$m_{2,2}$
600	$m_{1,0}$	$m_{1,1}$	$m_{1,2}$
400	$m_{0,0}$	$m_{0,1}$	$m_{0,2}$

Putting It All Together



Classifier

Perf. Counter 1

Perf. Counter 2

Perf. Counter N

Cluster 1

Cluster 2

Cluster M

Base Config. Execution Time

Target Config.

Engine Freq. (MHz)				
800 to 1000	$e_{2,0}$	$e_{2,1}$	$e_{2,2}$	$e_{2,3}$
600 to 800	$e_{1,0}$	$e_{1,1}$	$e_{1,2}$	$e_{1,3}$
400 to 600	$e_{0,0}$	$e_{0,1}$	$e_{0,2}$	$e_{0,3}$
	400	600	800	1000

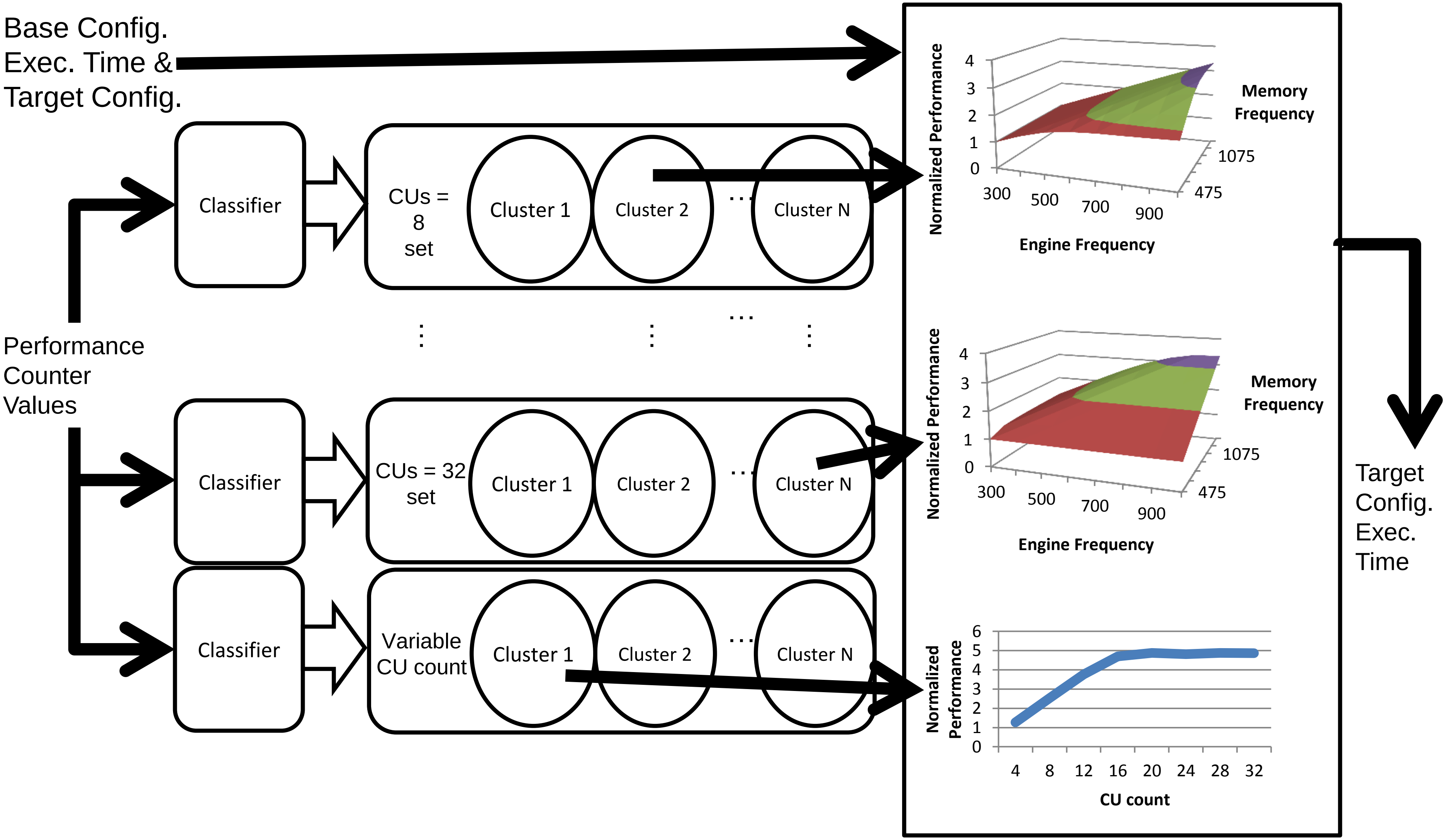
Mem. Freq. (MHz)

Engine Freq. (MHz)			
1000	$m_{3,0}$	$m_{3,1}$	$m_{3,2}$
800	$m_{2,0}$	$m_{2,1}$	$m_{2,2}$
600	$m_{1,0}$	$m_{1,1}$	$m_{1,2}$
400	$m_{0,0}$	$m_{0,1}$	$m_{0,2}$
	400 to 600	600 to 800	800 to 1000

Mem. Freq. (MHz)

Target Config. Execution Time or Power

Model Architecture



Base Config. Exec. Time & Target Config.

Performance Counter Values

Classifier

CUs = 8 set

Cluster 1

Cluster 2

…

Cluster N

Classifier

CUs = 32 set

Cluster 1

Cluster 2

…

Cluster N

Classifier

Variable CU count

Cluster 1

Cluster 2

…

Cluster N

Target Config. Exec. Time