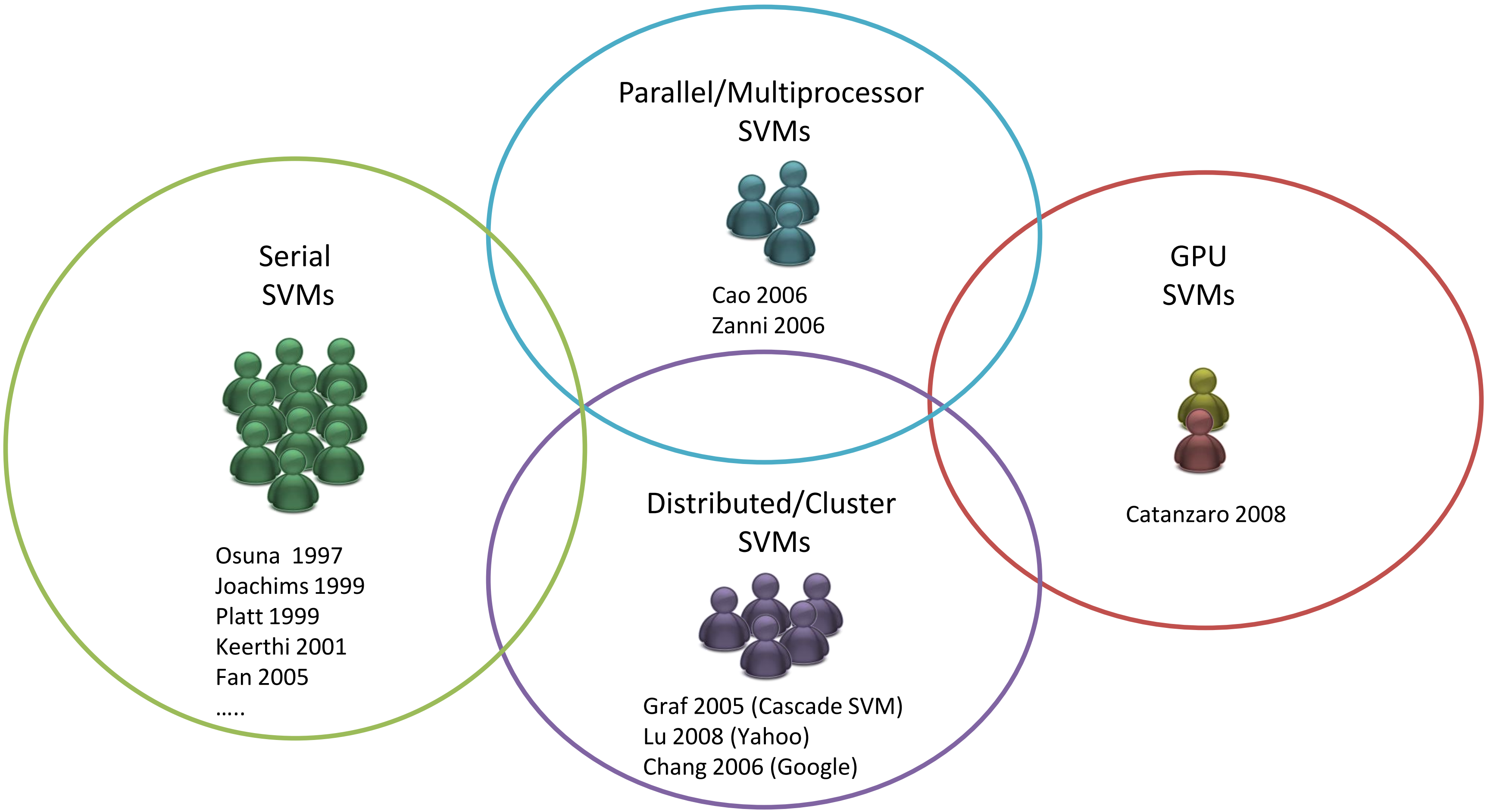
Multiclass Classification using SVMs on GPUs

Sergio Herrero

6.338J Applied Parallel Computing

Large Scale SVMs



Multiclass SVM

l samples $(\bar{x}_1, y_1), \ldots, (\bar{x}_l, y_l)$ $\bar{x}_i \in R^n,$ $y_i \in Y \; \forall i$ $Y = \{1, .., M\}$

R M x N Output code M classes $R_{ij} \in \{-1,0,1\}$

N tasks

$f^k(\bar{x})$ $(\bar{x}_1, R_{y_1 k}), \ldots, (\bar{x}_l, R_{y_l k})$ $k = 1..N$ $\longrightarrow$ $\hat{f}^k(\bar{x})$

$$\hat{y} = \underset{y \in Y}{\operatorname{argmax}} \left\{ \sum_{k=1}^{N} R_{yk} \hat{f}^k(\bar{x}) \right\}$$

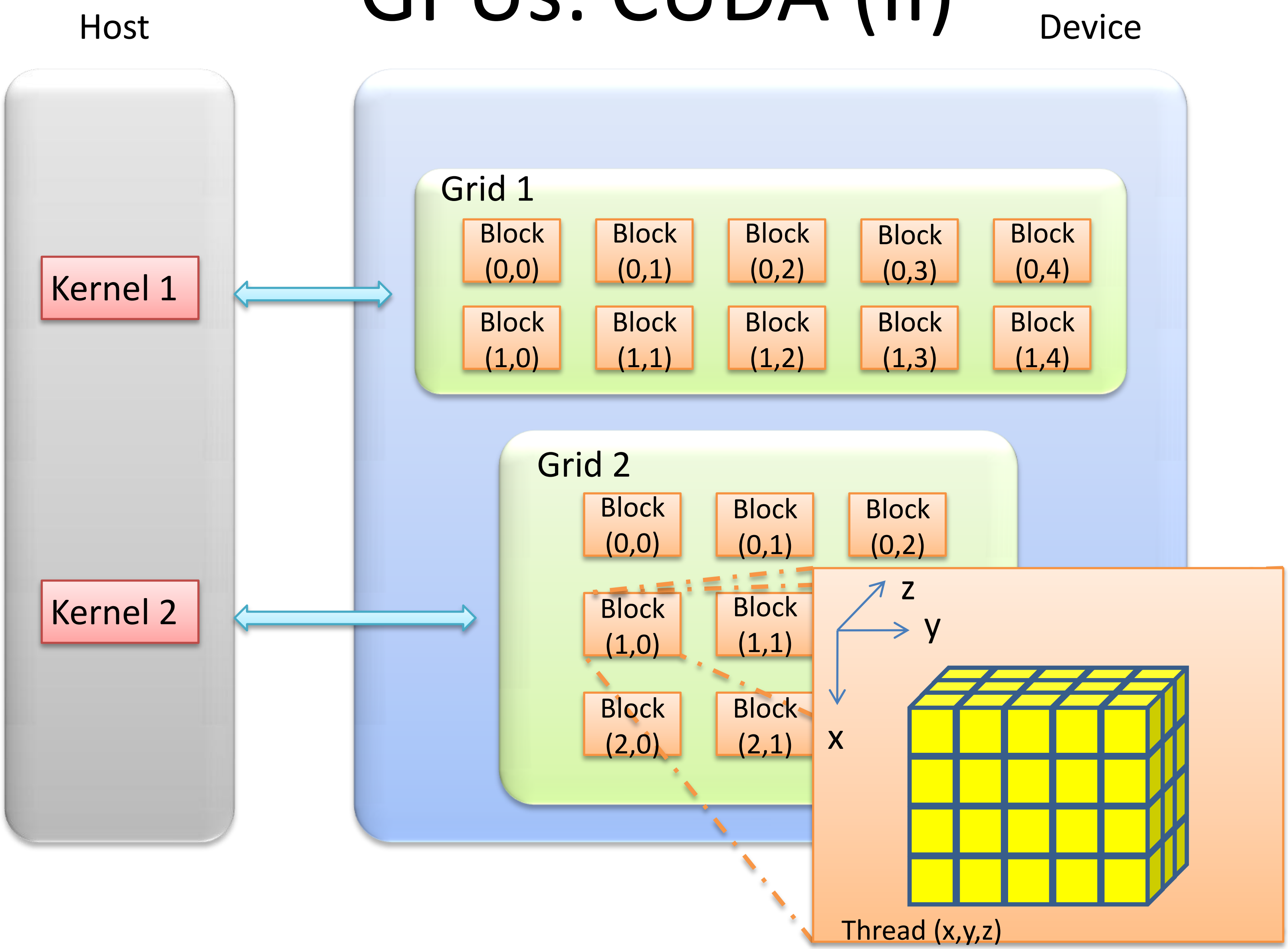
$$\hat{y} = \underset{y \in Y}{\operatorname{argmin}} \left\{ \sum_{k=1}^{N} Loss(R_{yk}, \hat{f}^k(\bar{x})) \right\}$$

GPUs: CUDA (I)

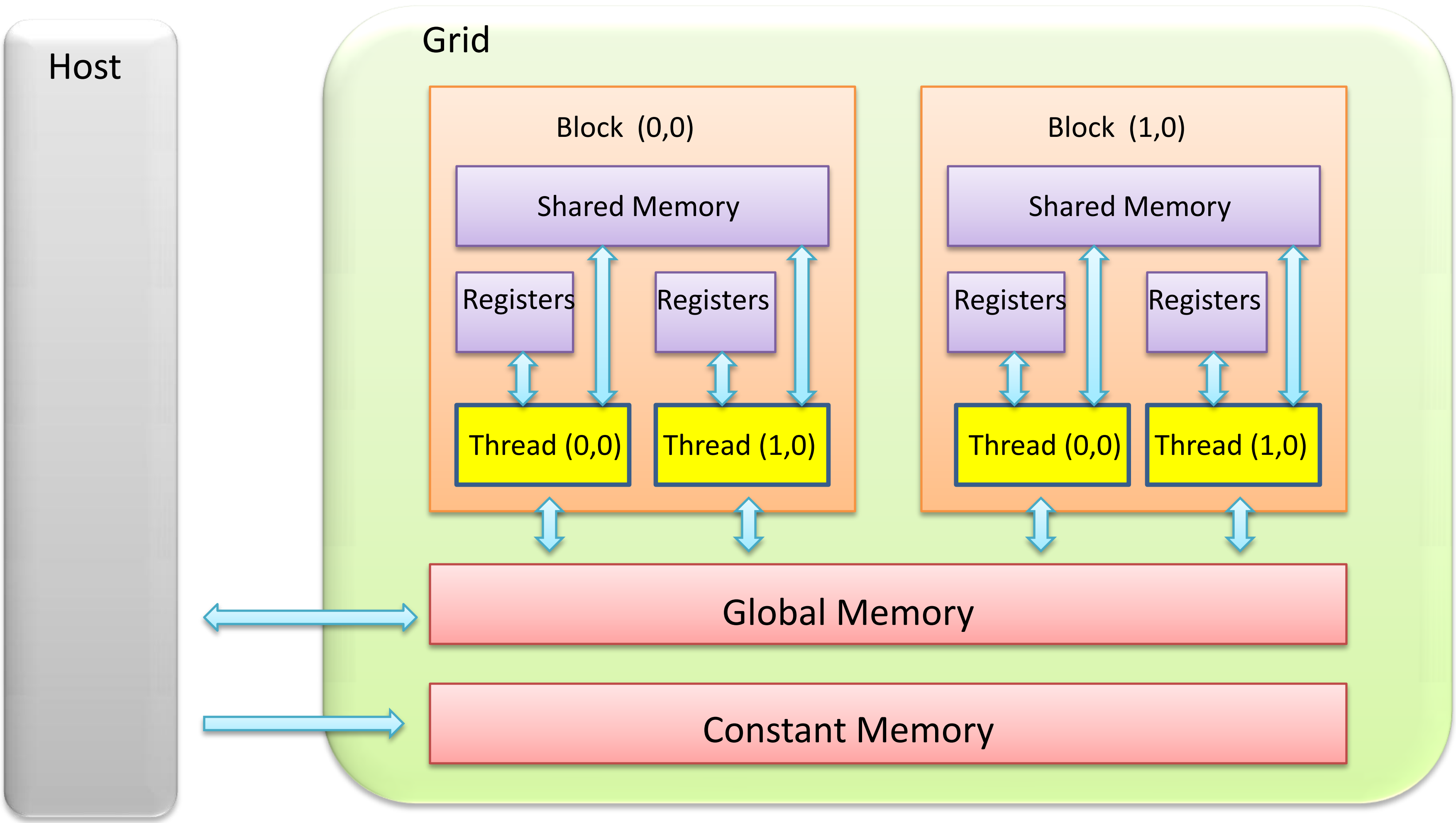
- CUDA Programming model
- Three key abstractions:
  - Hierarchy of thread groups
  - Shared memory
  - Barrier Synchronization

- Advantages:
  - High throughput in floating point computation (1 TFlop)
  - Aggressive Memory system (4 GB)
  - Fast memory bandwidth (102 GB/s)

GPUs: CUDA (II)


Host
Device
Kernel 1
Kernel 2
Grid 1
Block (0,0)
Block (0,1)
Block (0,2)
Block (0,3)
Block (0,4)
Block (1,0)
Block (1,1)
Block (1,2)
Block (1,3)
Block (1,4)
Grid 2
Block (0,0)
Block (0,1)
Block (0,2)
Block (1,0)
Block (1,1)
Block (2,0)
Block (2,1)
z
y
x
Thread (x,y,z)

GPUs: CUDA (III)



Parallel SMO

$$I_0^k = \{i: y_i^k = 1, 0 < \alpha_i^k < C\} \cup \{i: y_i^k = -1, 0 < \alpha_i^k < C\}$$

$$I_1^k = \{i: y_i^k = 1, \alpha_i^k = 0\}$$
$$I_2^k = \{i: y_i^k = -1, \alpha_i^k = C\}$$
$$I_3^k = \{i: y_i^k = 1, \alpha_i^k = C\}$$
$$I_4^k = \{i: y_i^k = -1, \alpha_i^k = 0\}$$

$$f_i^{p,k} = \sum_{j=1}^{l} \alpha_j^k y_j^k k(\bar{x}_j, \bar{x}_i) - y_i^k$$

$$b_{up}^{p,k} = \min\{f_i^{p,k}: i \in I_0^k \cup I_1^k \cup I_2^k \cup l^p\}$$

$$I_{up}^{p,k} = \operatorname*{argmin}_i f_i^{p,k}$$

$$b_{low}^{p,k} = \max\{f_i^{p,k}: i \in I_0^k \cup I_3^k \cup I_4^k \cup l^p\}$$

$$I_{low}^{p,k} = \operatorname*{argmax}_i f_i^{p,k}$$

$$b_{up}^k = \min\{b_{up}^{p,k}\}$$

$$I_{up}^k = \arg_{I_{up}^{p,k}} b_{up}^k$$

$$b_{low}^k = \max\{b_{low}^{p,k}\}$$

$$I_{low}^k = \arg_{I_{low}^{p,k}} b_{low}^k$$

$$\alpha_{I_{up}}^{new,k} = \alpha_{I_{up}}^{old,k} - \frac{y_{I_{up}}^k (f_{I_{low}}^{old,k} - f_{I_{up}}^{old,k})}{\eta}$$

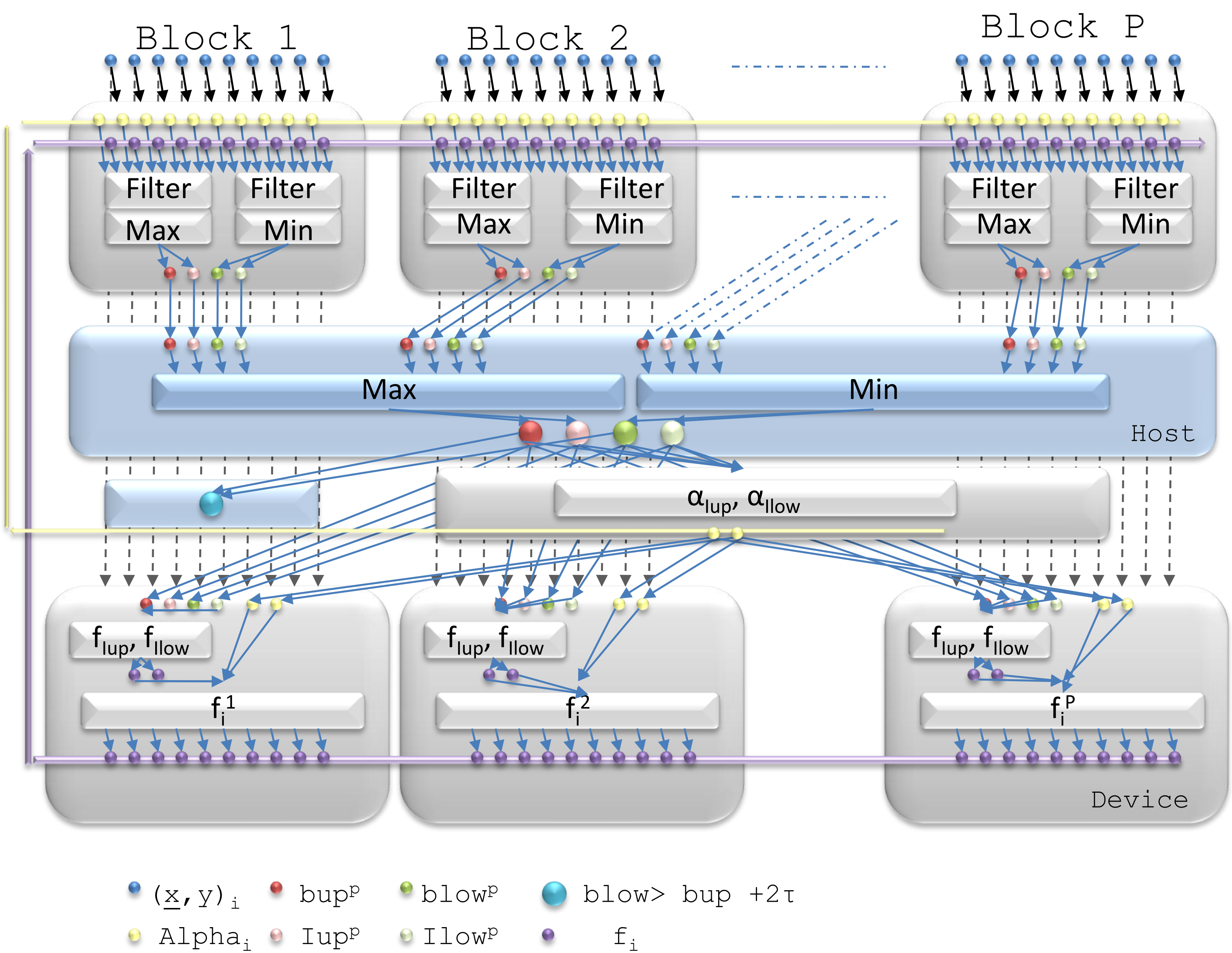
$$\alpha_{I_{low}}^{new,k} = \alpha_{I_{low}}^{old,k} + s(\alpha_{I_{up}}^{old,k} - \alpha_{I_{up}}^{new,k})$$

$$s = y_{I_{up}}^k y_{I_{low}}^k$$
$$\eta = 2k\left(\bar{x}_{I_{low}}, \bar{x}_{I_{up}}\right) - k(\bar{x}_{I_{low}}, \bar{x}_{I_{low}}) - k\left(\bar{x}_{I_{up}}, \bar{x}_{I_{up}}\right)$$

$$f_i^{p,new,k} = f_i^{p,old,k} + \left(\alpha_{I_{low}}^{new,k} - \alpha_{I_{low}}^{old,k}\right) y_{I_{low}}^k k\left(\bar{x}_{I_{low}}, \bar{x}_i\right) + \left(\alpha_{I_{up}}^{new,k} - \alpha_{I_{up}}^{old,k}\right) y_{I_{up}}^k k\left(\bar{x}_{I_{up}}, \bar{x}_i\right)$$

Initialize:
  $\alpha_i{}^k = 0,\ f_i^{p,k} = -y_i^k,\ i \in l^p,$
  $p = 1 \ldots P,\ k = 1 \ldots N$
Calculate:
  $b_{up}^{p,k},\ I_{up}^{p,k},\ b_{low}^{p,k},\ I_{low}^{p,k},\ p = 1 \ldots P,\ k = 1 \ldots N$
Obtain:
  $b_{up}^k,\ I_{up}^k, b_{low}^k, I_{low}^k,\ k = 1 \ldots N$

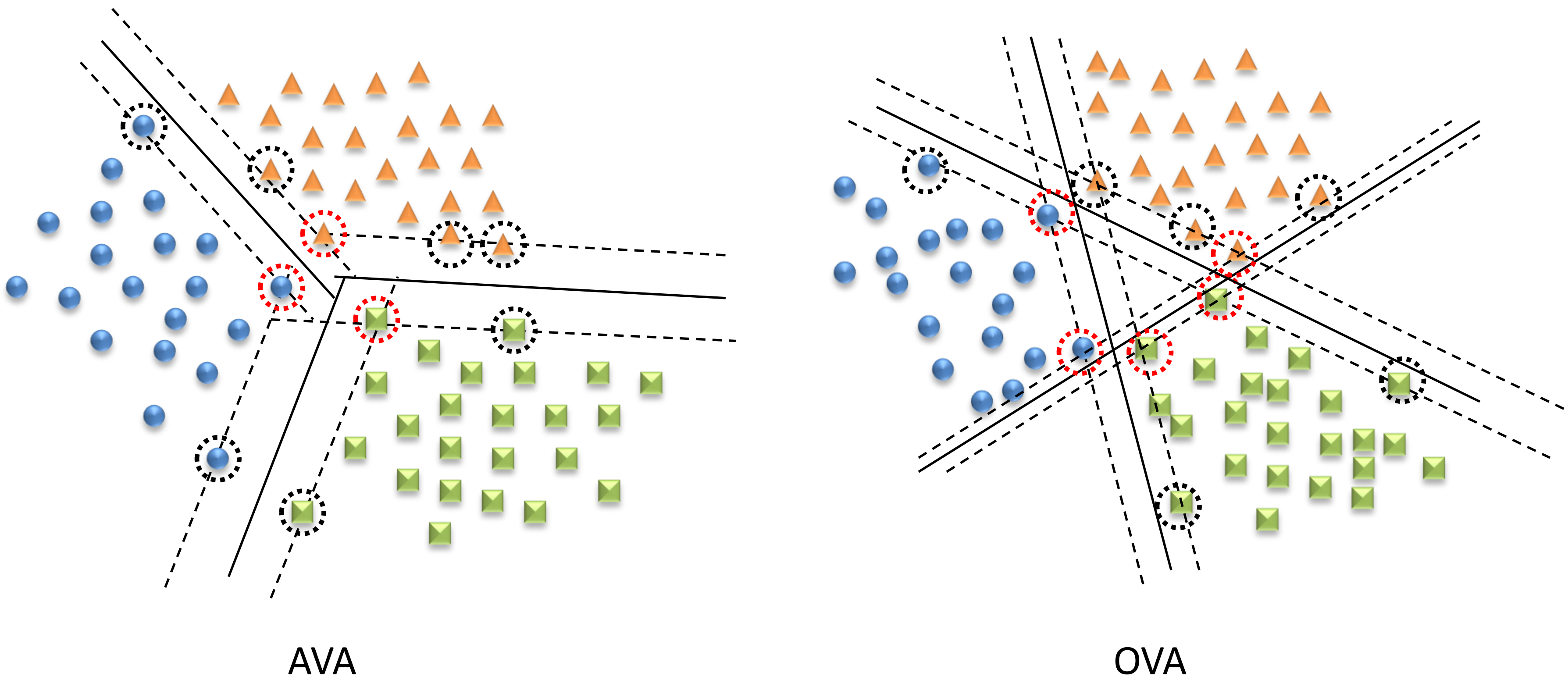
Iterate task k until $b_{low}^k > b_{up}^k + 2\tau$
  Optimize $\alpha_{I_{up}}^k, \alpha_{I_{low}}^k$
  Update $f_i^{p,k}$, $p = 1 \ldots P$
  Calculate $b_{up}^{p,k},\ I_{up}^{p,k},\ b_{low}^{p,k},\ I_{low}^{p,k},\quad p = 1 \ldots P$
  Obtain $b_{up}^k,\ I_{up}^k, b_{low}^k, I_{low}^k$
Repeat



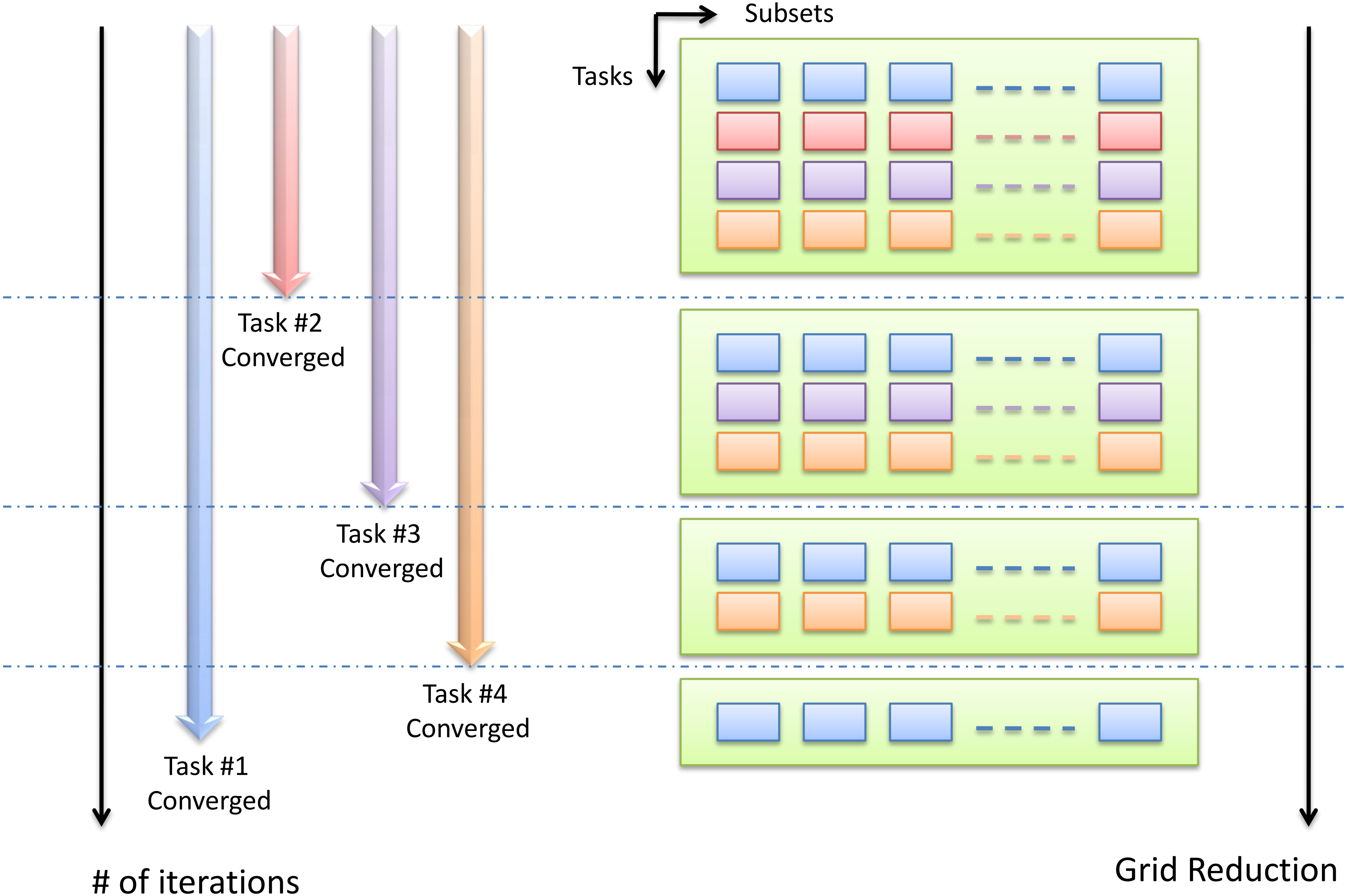
Block 1
Block 2
Block P
Filter
Max
Filter
Min
Max
Min
Host
$\alpha_{lup}, \alpha_{llow}$
f_{lup}, f_{llow}
f_i^1
f_i^2
f_i^P
Device
$(\underline{x}, y)_i$
bup^p
$blow^p$
$blow > bup + 2\tau$
$Alpha_i$
Iup^p
$Ilow^p$
f_i

Parallel Tasks (I)

Kernel Caching (Joachims 1999) $\alpha_{I_{up}}^{new,k}$ $\alpha_{I_{low}}^{new,k}$



Parallel Tasks (II)



Performance Results (I)

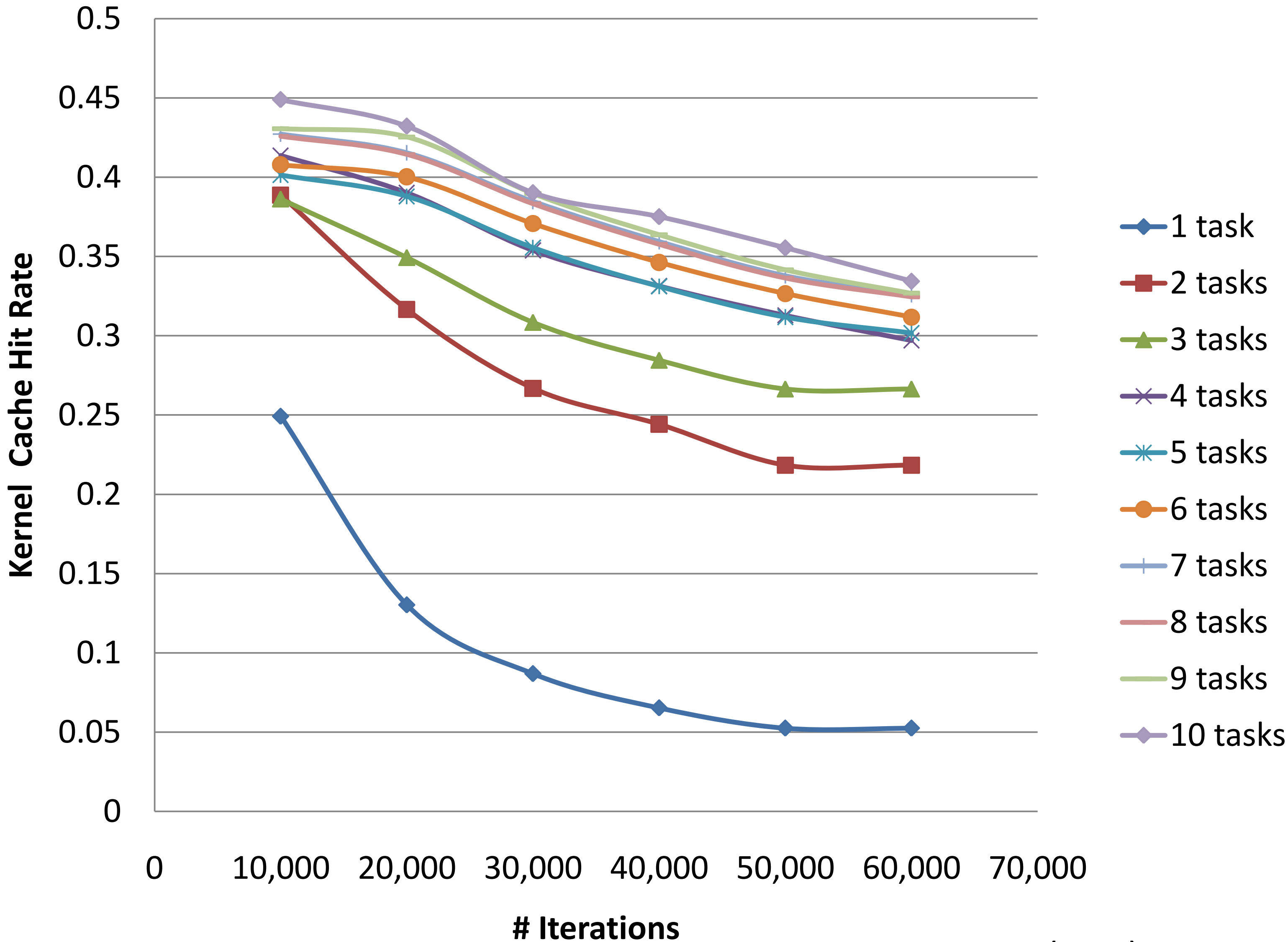
Host-Device Specifications:

Host	Device
Ubuntu 8.10 64bit	Tesla C1060
CPU: Intel Core i7 920 @ 2.67 GHz	# Stream Processors: 240
Memory 6GB (3x2 DDR2)	Frequency of Processors: 1.3GHz
	933 Gflops
	Memory: 4GB DDR3
	Memory Bandwidth: 102GB/s
Host <-> Device	
PCIe x16 (8GB/s)	

Datasets:

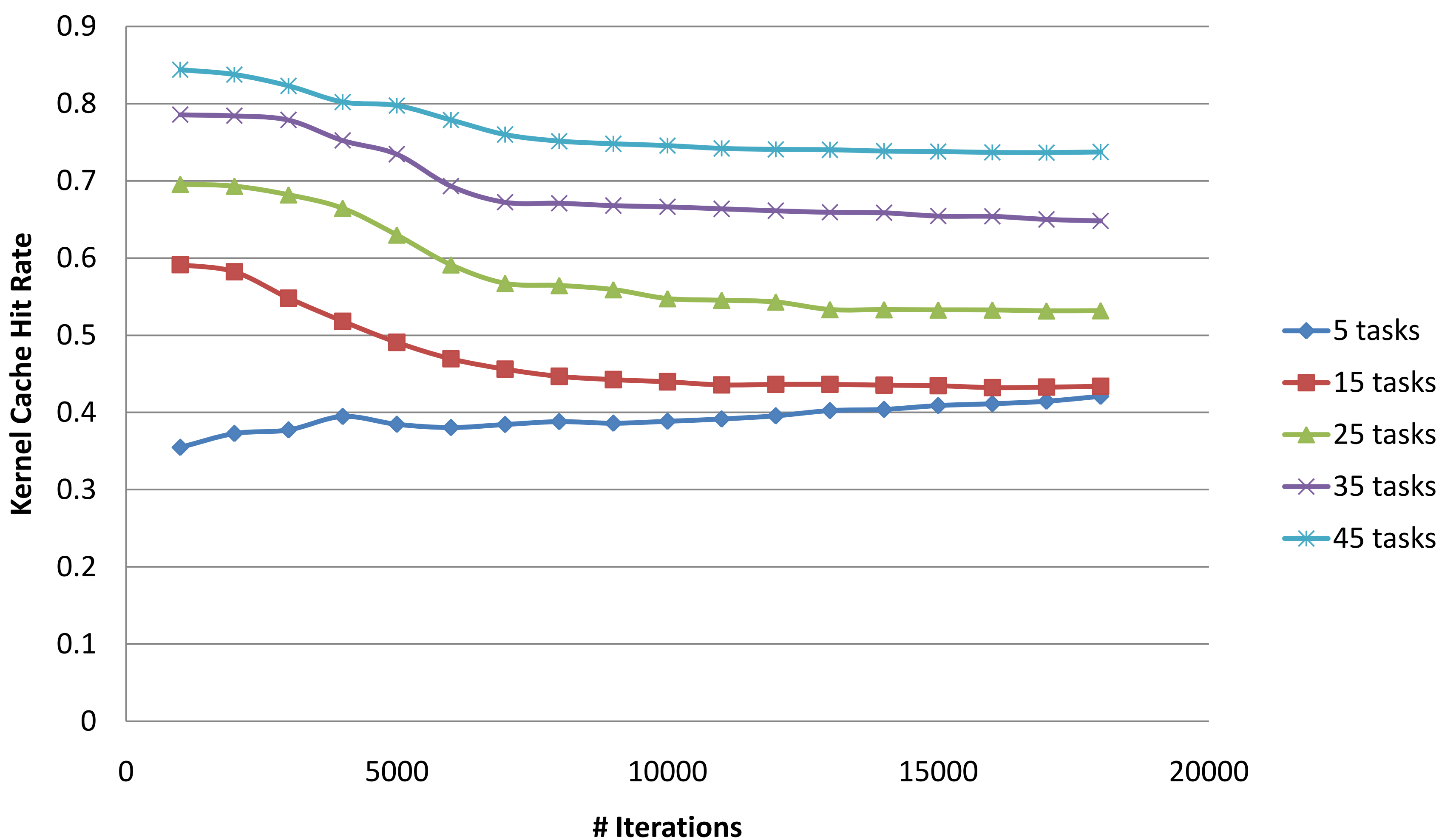
Dataset	# Training Points	# Testing Points	# Features	# Classes	C	β
Adult	32,561	16,281	123	2	100	0.5
MNIST	60,000	10,000	780	10	10	0.125

Performance Results (II)



MNIST (OVA)

Performance Results (III)



MNIST (AVA)

Performance Results (IV)

Accuracy (Binary tasks):

Dataset	SVM	Accuracy (%)	# SVs	Difference in b (%)	Iterations
Adult	GPU	82.697624	18668	0.01	115565
	LIBSVM	82.697624	19058		43735
MNIST	GPU	96	43730	0.04	69535
	LIBSVM	96	43756		76385

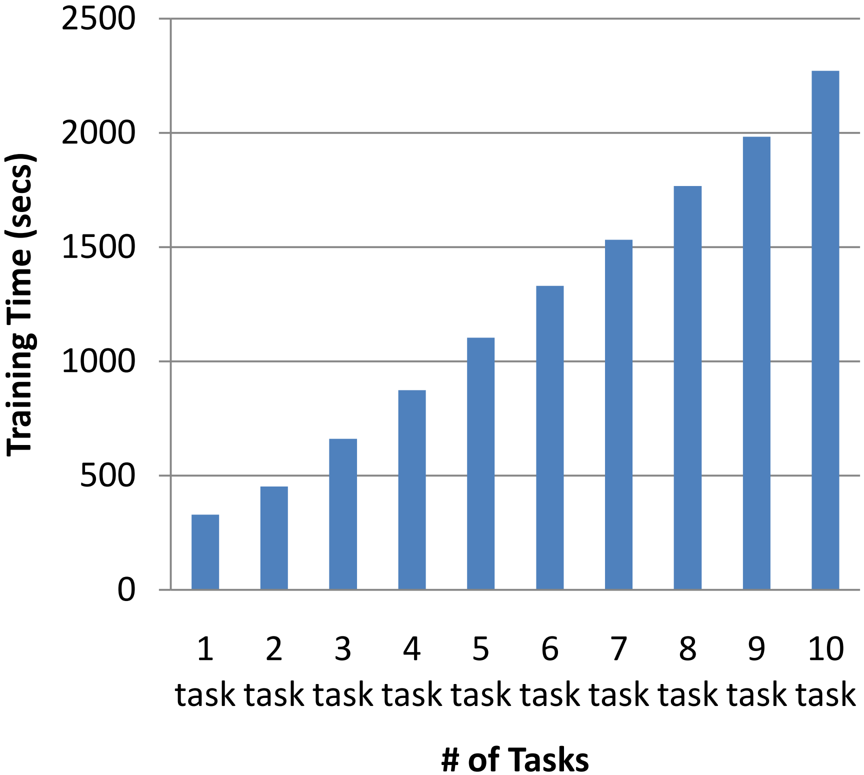
Training Time (Binary & Multiclass):

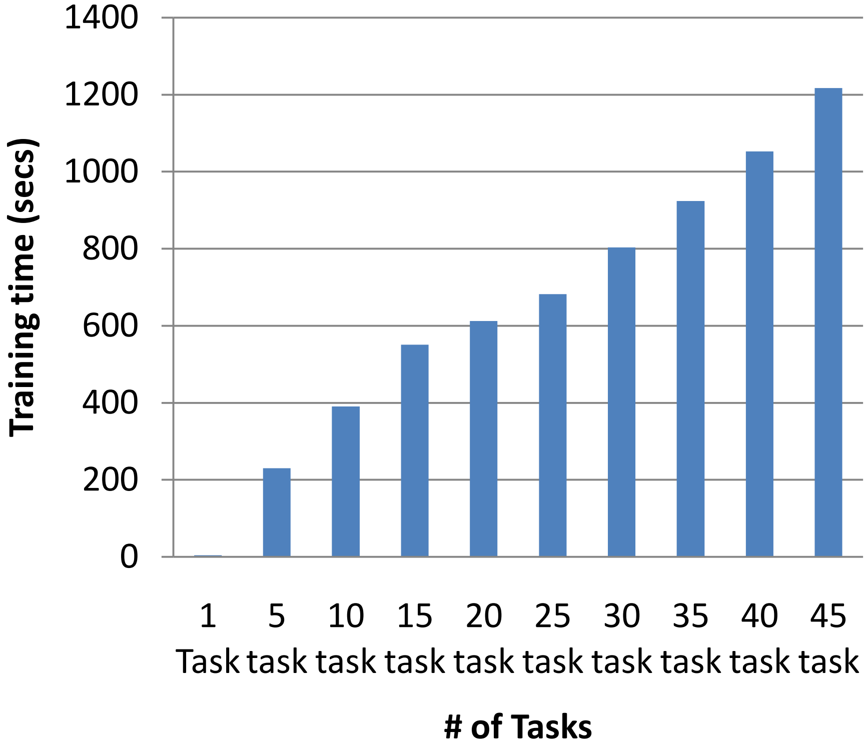
Dataset	GPU (sec)		LIBSVM (sec)	Speedup
Adult	38.0542		479	12.58731
	OVA (10 tasks)	AVA (45 tasks)	AVA (45 tasks)	
MNIST	2272.71	1217.333	27833	22.86392

~ 20 min
☺

~ 7 hours,
53 min
☹

Performance Results (V)





MNIST (OVA)
1172 Blocks per iteration

MNIST (AVA)
5274 Blocks per iteration

Conclusions:

-Naïve implementation of multiclass SVM:
- One order of magnitude of speedup compared to LIBSVM
- Room for improvement
  - -Second order heuristics (Keerthi 2001)
  - -Sparse matrices (Joachims 2006)
  - -Parallel programming experience (me)

-Future work
- Distributed SVM training on multi GPU scenarios (Graf 2005, Lu 2008)