Welcome to Dissertation Defense of Jyotikrishna Dass

- Presentation will be Recorded
- Non-Committee attendees are requested
  - during presentation, to ensure
    - Audio is MUTE
    - Video is OFF
  - after presentation, join for Q&A and comments

Efficient and Scalable Machine Learning for Distributed Edge Intelligence

PhD Dissertation

Jyotikrishna Dass
Ph.D. Candidate
Computer Science and Engineering

email: dass.jyotikrishna@tamu.edu

TEXAS A&M UNIVERSITY

Artificial Intelligence is Everywhere

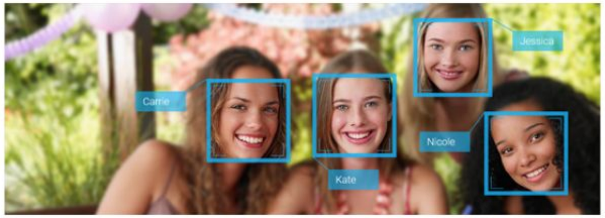


Figure: Face Recognition

Figure: Machine Translation

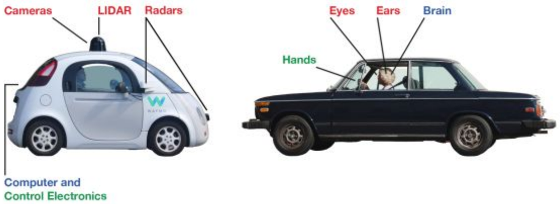


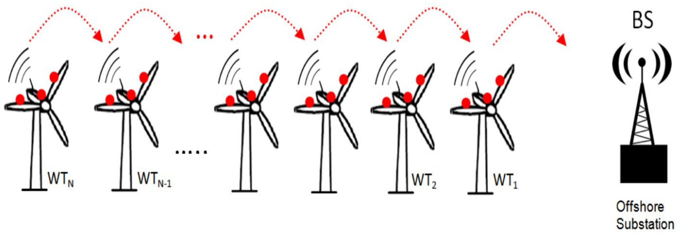
Figure: Self Driving Cars

Figure: Speech Processing

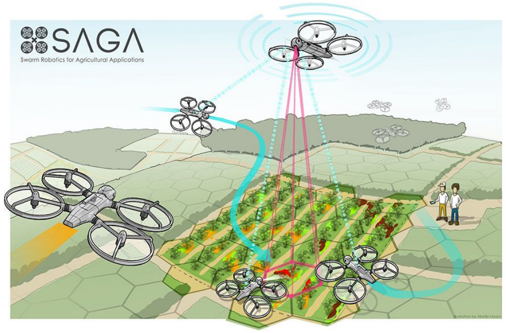
Picture Credits. Google Images

So are, Internet of Things

Intelligent Traffic



Smart Wind Farms

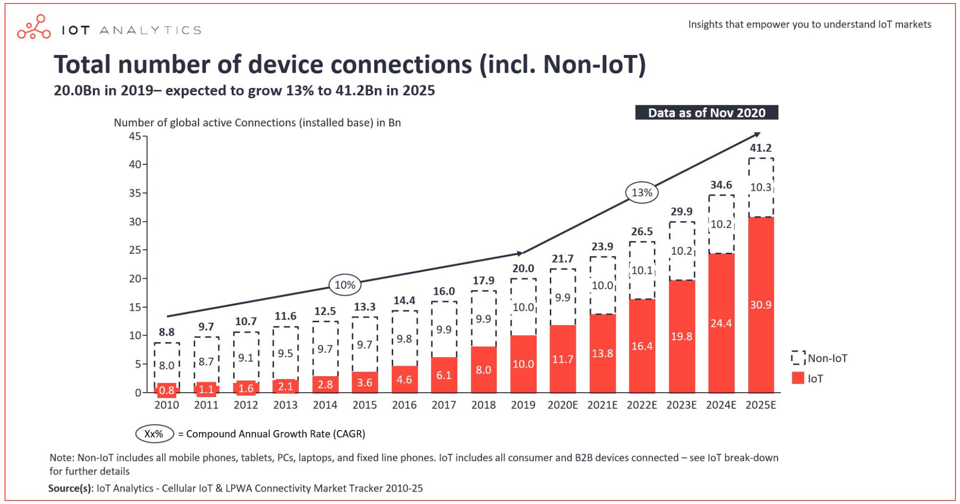


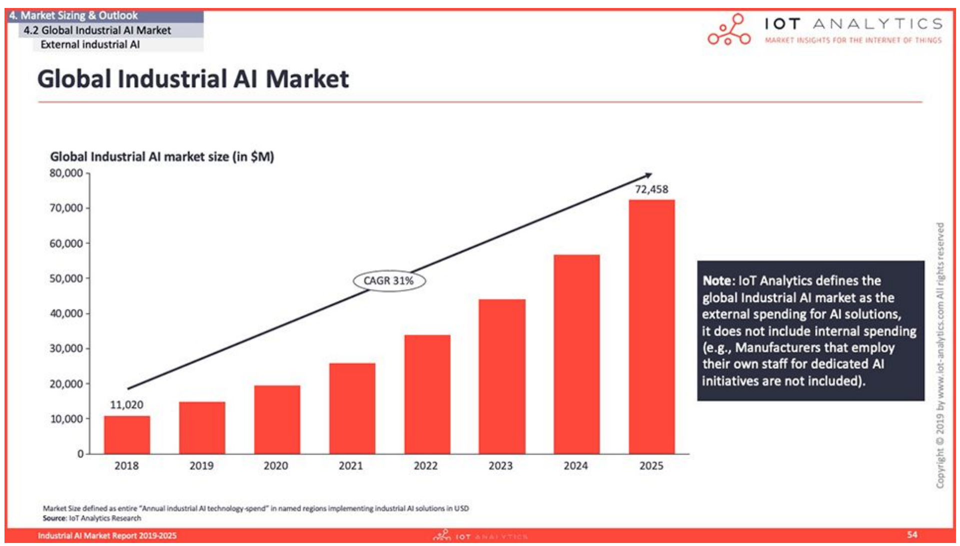
Precision Agriculture

Disaster Relief

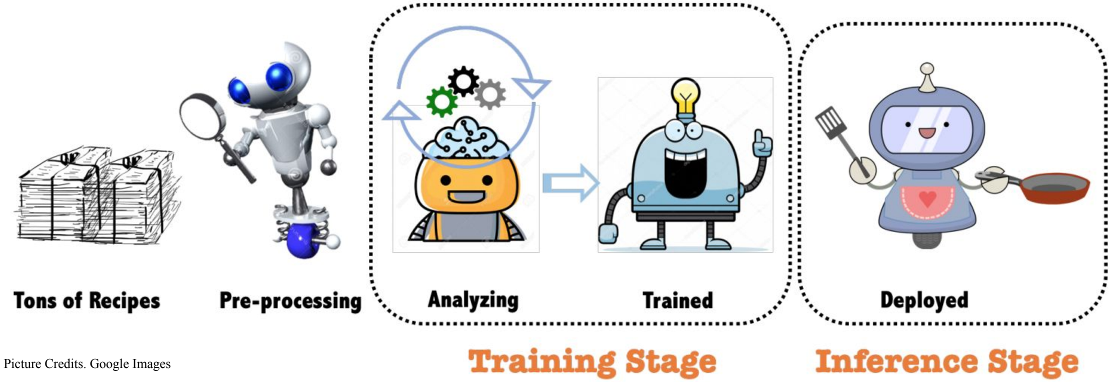
Picture Credits. Google Images

Market Trends: AI + IoT





Machine Learning Process

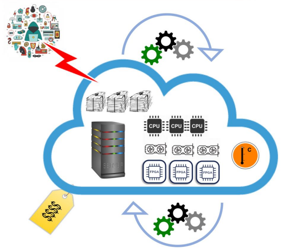


Picture Credits. Google Images

ML Training, usually

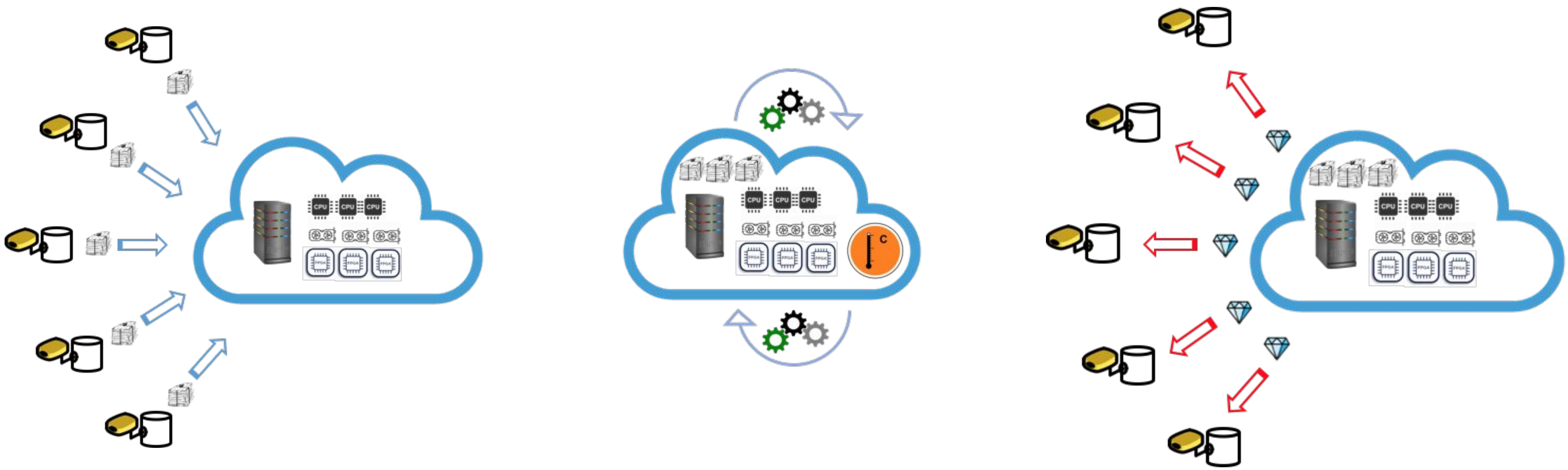
- Requires high memory to store large-scale training dataset (bigdata)
- Involves lots of iterations
- Has huge computational cost
- Is inherently sequential

Training is EXPENSIVE !

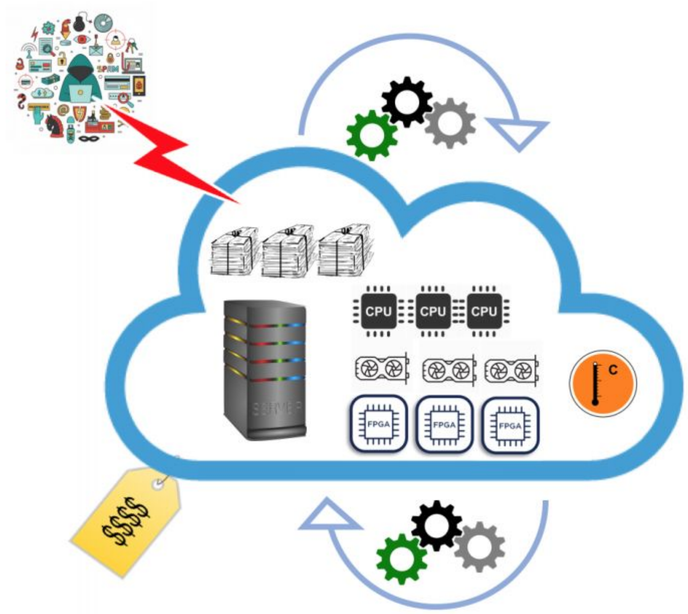


Centralized ML Framework

1. Collect data at Edge devices and transfer to HPC cloud server
2. Train ML model in the HPC cloud server using training algorithms
3. Deploy the trained model back at Edge devices for inference



Challenges of Centralized ML

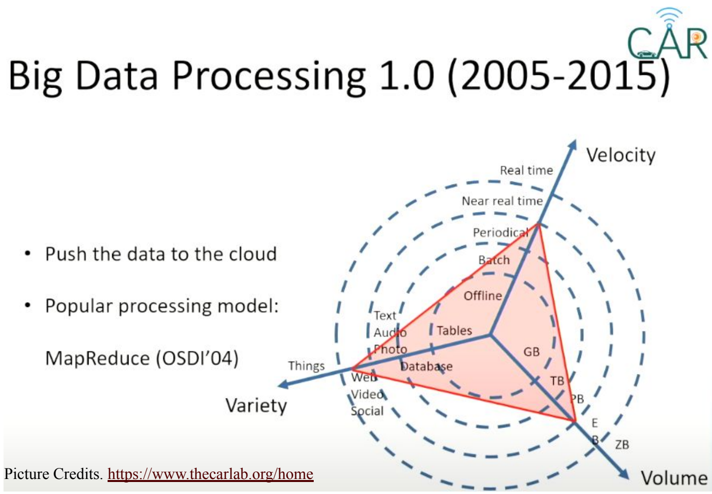


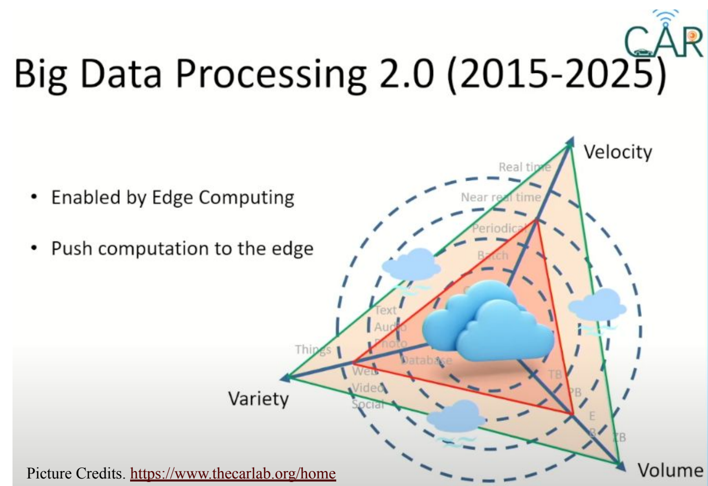
At HPC cloud server

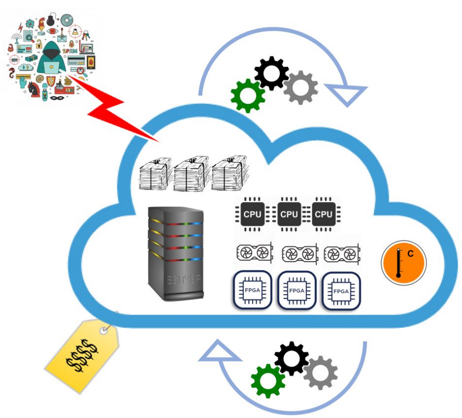
- Privacy at Risk
- High energy dissipation
- Expensive in cost $$$$
- High network latency

and, today there are Billions of connected small devices at Edge (IoT) having new requirements that can not be satisfied by Centralized framework

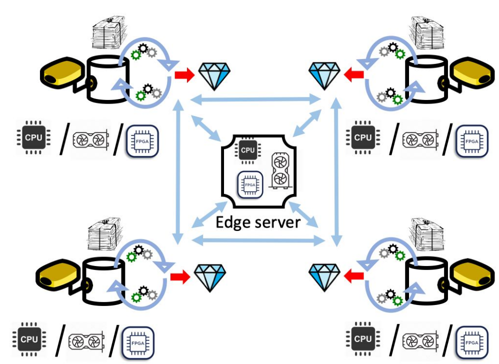
Paradigm Shift





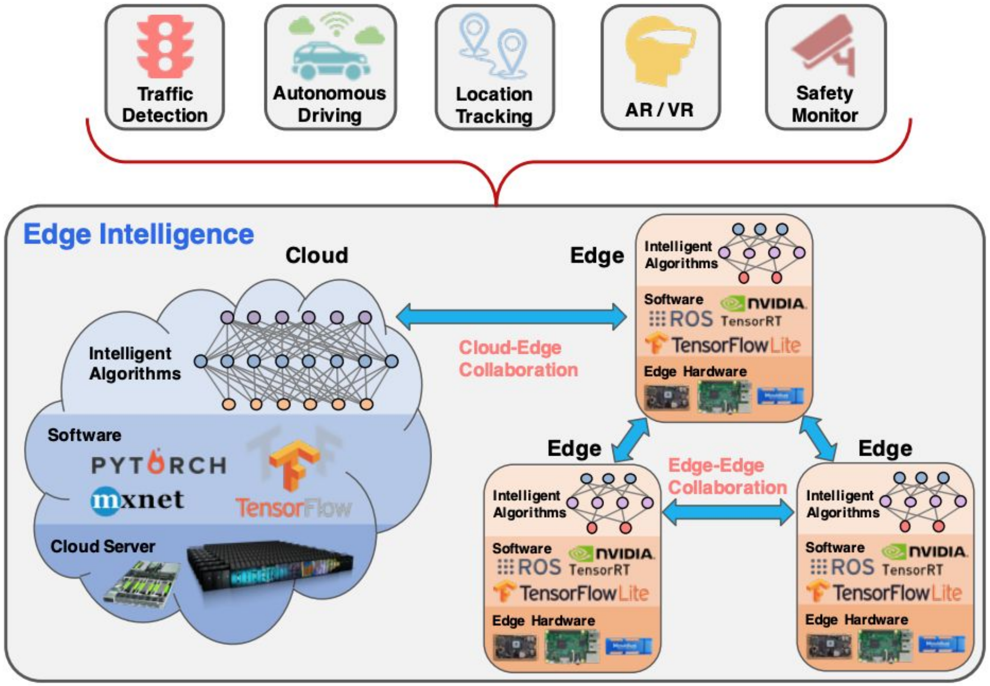


Cloud



Edge

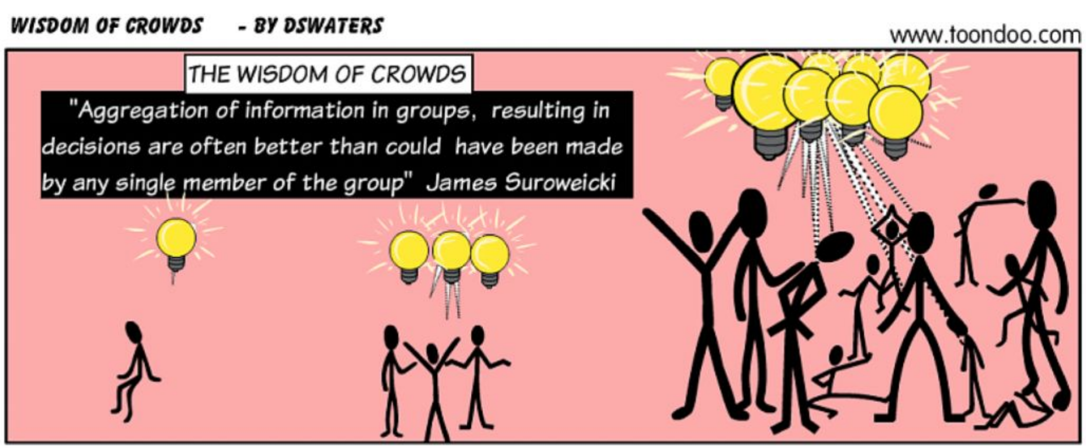
Edge Intelligence



Zhang, Xingzhou, Yifan Wang, Sidi Lu, Liangkai Liu, and Weisong Shi. "Openei: An open framework for edge intelligence." In 2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS 2019)

Hypothesis

Decentralize Machine Learning via
Efficient and Scalable Algorithms and Architectures
for Distributed Edge Intelligence



Picture Credits. Google Images

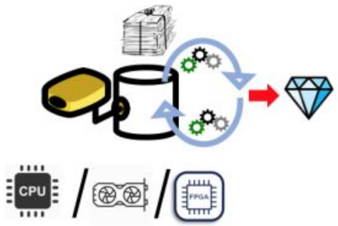
How to Decentralize ML?

Making training CHEAP

$\Downarrow$

will REDUCE the need for HPC cloud server

Bring training on edge



Edge Device

- Has less storage
- Handles light computations
- Single low power CPU/GPU/FPGA
- Low cost $$

$\Downarrow$

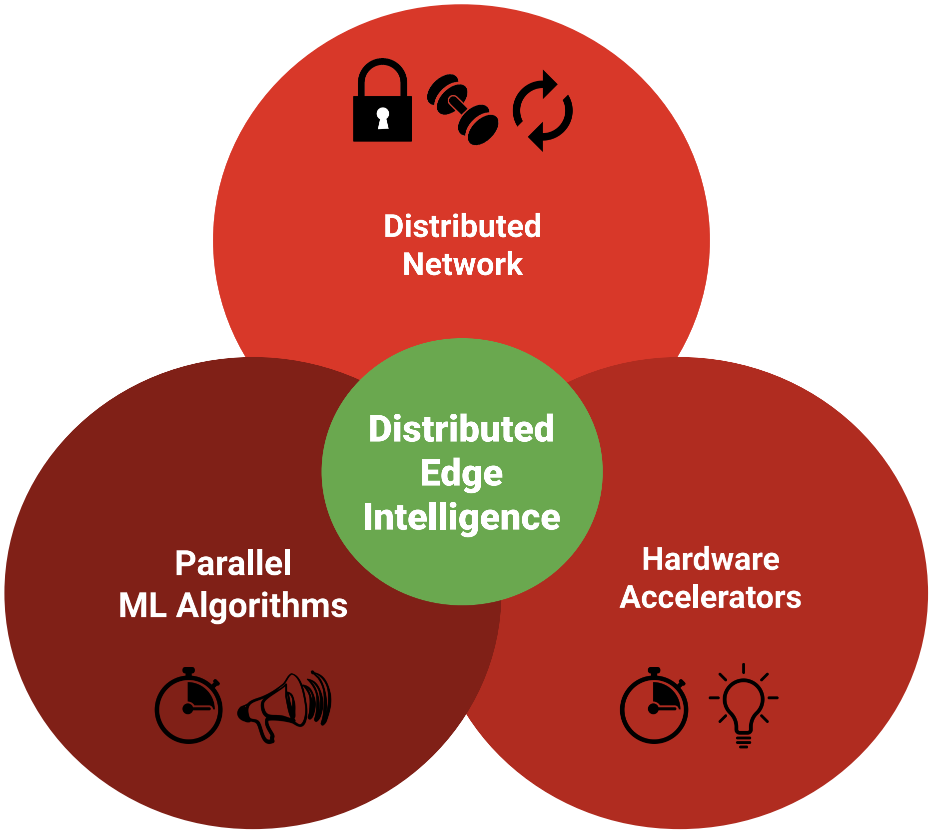
and AVOID centralization

Go for decentralized/distributed framework with parallel training

Requirements for Edge Intelligence

01	Protect Data Privacy	• Keep data decentralized and local on devices • Design privacy-preserving ML algorithms
02	Reduce Latency	• Efficient and scalable training algorithms • Cheap inference calculations to enable real-time analytics
03	Save Bandwidth	• Communicate less data during training • Reduce synchronizations and idling during training
04	Energy Efficiency	• Efficient computation and communication process • Build energy-efficient hardware accelerators for Green AI
05	Build Robust Models	• Devise fault-tolerance for device failures or stragglers • Accurate and robust model to data perturbations
06	Streaming Data	• Incremental federated learning to update the global model • Discard data after each update for memory and privacy

Research Focus



01	Protect Data Privacy
02	Reduce Latency
03	Save Bandwidth
04	Energy Efficiency
05	Build Robust Models
06	Streaming Data

Research Contributions

1. Relaxed Synchronization for Parallel QP Problems
2. Householder Sketch for Machine Learning
3. Memory-efficient Framework for Distributed ML
4. Communication-efficient Framework for Scalable ML
5. Multiple FPGA-based System for Energy-efficient ML
6. Rapid Incremental Solver for Federated ML

Relaxed Synchronization for Parallel QP Problems

01	Protect Data Privacy	
02	Reduce Latency	
03	Save Bandwidth	
04	Energy Efficiency	
05	Build Robust Models	
06	Streaming Data	

Parallel QP Problems

Formulation

$$\min_{\boldsymbol{x}} \sum_{i=1}^{p} \left(\frac{1}{2} \boldsymbol{x}_i^T \boldsymbol{K}_i \boldsymbol{x}_i + \boldsymbol{c}_i^T \boldsymbol{x}_i \right)$$

$$\text{subject to} \quad \sum_{i=1}^{p} \boldsymbol{A}_i \boldsymbol{x}_i = \boldsymbol{b}$$

where,
$\boldsymbol{x}_i, \boldsymbol{c}_i \in \mathbb{R}^{\frac{n}{p}}$, $\boldsymbol{b} \in \mathbb{R}^m$, $\boldsymbol{A}_i \in \mathbb{R}^{m \times \frac{n}{p}}$, and $\boldsymbol{A} = [\boldsymbol{A}_1, \ldots, \boldsymbol{A}_p] \in \mathbb{R}^{m \times n}$
$\boldsymbol{K}_i \in \mathbb{R}^{\frac{n}{p} \times \frac{n}{p}}$ is symmetric positive definite matrix, and
$\boldsymbol{K} = \mathrm{diag}(\boldsymbol{K}_1, \ldots, \boldsymbol{K}_p) \in \mathbb{R}^{n \times n}$
n : Number of variables, m : Number of constraints,
p : Number of separable (parallel) sub-problems

Lagrangian $\mathcal{L}_i$ for each sub-problem, $i = 1, \ldots, p$

$$\mathcal{L}_i(\boldsymbol{x}_i, \boldsymbol{\beta}) = \left(\frac{1}{2} \boldsymbol{x}_i^T \boldsymbol{K}_i \boldsymbol{x}_i + \boldsymbol{c}_i^T \boldsymbol{x}_i \right) + \boldsymbol{\beta}^T (\boldsymbol{A}_i \boldsymbol{x}_i - \boldsymbol{b})$$

where, $\boldsymbol{\beta} \geq \boldsymbol{0}_m \in \mathbb{R}^m$ is a vector of Lagrangian dual variables

Parallel Dual Ascent

Dual sub-function: $g_i(\boldsymbol{\beta}) = \min_{\boldsymbol{x}_i} \mathcal{L}_i(\boldsymbol{x}_i, \boldsymbol{\beta})$
Dual problem: $\max_{\boldsymbol{\beta}} g(\boldsymbol{\beta})$

Parallel Dual Ascent steps

Gradient method - involves iterating through the following steps until convergence (error in β falls below stopping threshold)

Step 1: Minimization of Lagrangian for each sub-problem i

$$\begin{aligned} \boldsymbol{x}_i^{t+1} &= \arg\min_{\boldsymbol{x}_i} \mathcal{L}_i(\boldsymbol{x}_i, \boldsymbol{\beta}^t) \\ &= -\left(\boldsymbol{K}_i\right)^{-1} (\boldsymbol{A}_i^T \boldsymbol{\beta}^t + \boldsymbol{c}_i) \end{aligned}$$

solved in parallel using p machines which then broadcast local $\boldsymbol{A}_i \boldsymbol{x}_i^{t+1}$ during every iteration

Step 2: Dual variable update (Global Gathering of $\boldsymbol{A}_i \boldsymbol{x}_i^{t+1}$)

$$\boldsymbol{\beta}^{t+1} = \boldsymbol{\beta}^t + \eta(\sum_{i=1}^{p} \boldsymbol{A}_i \boldsymbol{x}_i^{t+1} - \boldsymbol{b})$$

$\eta > 0$ is the step size, $\boldsymbol{\beta}^{t=0} = \boldsymbol{0}_m$.

Synchronization is Necessary and Unavoidable in Step 2

- **Results in idling**
- **Leads to waste of computing time**

Relax Synchronization

We propose,

LSDA: Lazily Synchronized Dual Ascent

- Do not synchronize at every iteration
- Communicate data periodically
- Minimize frequency of communication

TSDA: Tightly Synchronized Dual Ascent

$$\boldsymbol{\beta}^{t+1} = \boldsymbol{\beta}^t + \eta(\sum_{i=1}^{p} \boldsymbol{A}_i \boldsymbol{x}_i^{t+1} - \boldsymbol{b}) = \boldsymbol{\beta}^t + \eta \sum_{i=1}^{p} \left(\boldsymbol{A}_i \boldsymbol{x}_i^{t+1} - \frac{1}{p} \boldsymbol{b} \right)$$

LSDA: Lazily Synchronized Dual Ascent

$$\boldsymbol{\beta}^{t+1} = \boldsymbol{\beta}^t + \eta \sum_{i=1}^{p} \left(\boldsymbol{A}_i \boldsymbol{x}_i^{kP+1} - \frac{1}{p} \boldsymbol{b} \right), \quad kP \le t < (k+1)P,$$

where, $k \in \mathbb{N}$, and $P \geq 1$ is the synchronization period

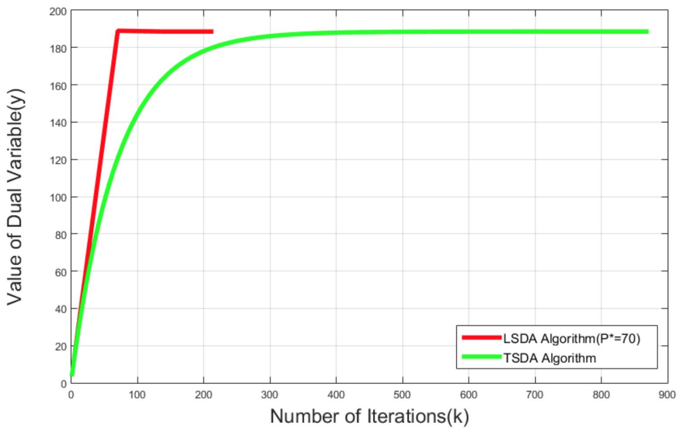
Synchronize across p parallel processes after every P iterations

Results (1/2)

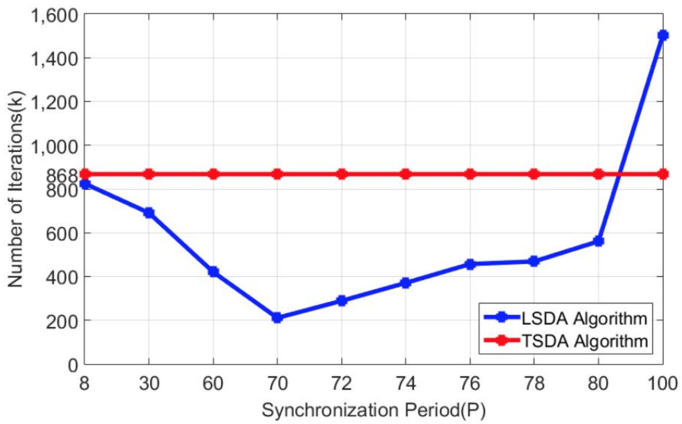
The data was synthetically generated with random values uniformly distributed over $[-1, 1]$. The problem specifics are as follows:

Experimental Setup

1) Number of instances in synthetic dataset, $d = 200,000$.
2) Step size, $\alpha = 0.27$.
3) Optimal Synchronization Period, $P^* = 70$.
4) Stopping threshold, $\epsilon = 10^{-5}$.
5) Cluster Size, $N = \{10, 20, 32, 40\}$.

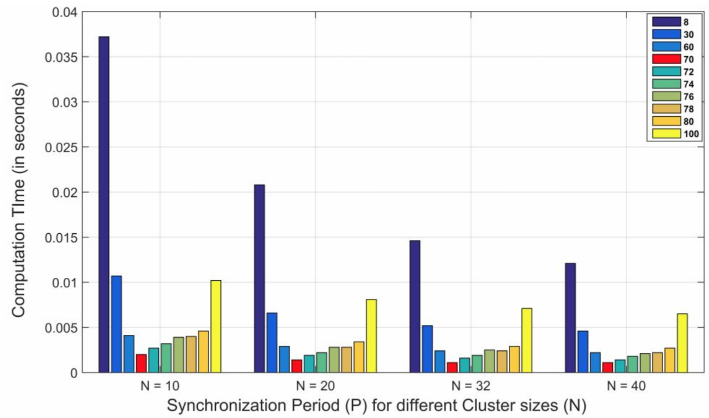


LSDA converges to the optimal solution of the dual variable significantly faster than the TSDA



Communication frequency is minimum at optimal synchronization period, P=70

Results (2/2)



Computation time is minimum at P=70 irrespective of number of parallel workers

Computing Performance Comparison between TSDA & LSDA algorithm

	TSDA algorithm	LSDA algorithm
No. of iteration	868	211
Sync. period	1	70
No. of Sync.	868 (=868/1) times	3 (=211/70) times
Comm. delay reduction	99.65%	
Speedup	160 times	

Research Contributions

✔ Relaxed Synchronization for Parallel QP Problems

[IEEE IPDPS'16] K. Lee, R. Bhattacharya, **J. Dass**, V. N. S. P. Sakuru, R. N. Mahapatra, "*A Relaxed Synchronization Approach for Solving Parallel Quadratic Programming Problems with Guaranteed Convergence,*"

❑ Householder Sketch for Machine Learning

❑ Memory-efficient Framework for Distributed ML

❑ Communication-efficient Framework for Scalable ML

❑ Multiple FPGA-based System for Energy-efficient ML

❑ Rapid Incremental Solver for Federated ML

Householder Sketch for Machine Learning

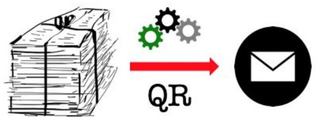
Sketching

A compressed mapping of few or all data points ($\boldsymbol{x}$) in a data set to generate **data summary** called *Sketch ($\boldsymbol{s}$)* to preserve or approximate the covariance matrix, i.e.,

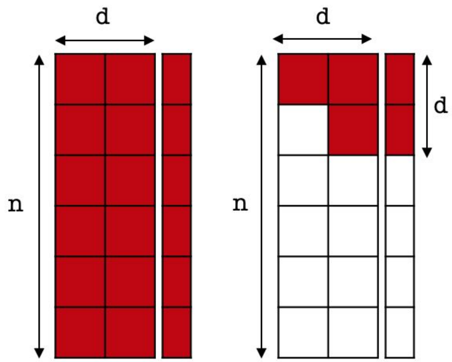
$$\boldsymbol{S}^T\boldsymbol{S} \cong \boldsymbol{X}^T\boldsymbol{X}$$

Householder Sketch

Theorem 3.2 (Householder Sketch)**.** *Let* $\mathbf{X} \in \mathbb{R}^{n \times d}$ *be the original data matrix,* $\mathbf{y} \in \mathbb{R}^{n}$ *be the corresponding output label or response vector, and* $n \gg d$*. Let* $\mathbf{X} = \mathbf{QR}$ *be Householder QR decomposition. Then,* $(\mathbf{R}, \mathbf{Q}^T\mathbf{y})$ *is a memory-efficient and theoretically accurate sketch of original data* $(\mathbf{X}, \mathbf{y})$ *such that* $\mathbf{X}^T\mathbf{X} = \mathbf{R}^T\mathbf{R}$*, and has memory footprint of* $\left(\frac{d(d+3)}{2}\right)$ *elements, computed in time* $O(nd^2)$.



Householder Sketch
(summary)



	$\{X,y\}$	$\{R,Q^Ty\}$
memory consumption	$O(nd)$	$O(d^2)$

Householder-Sketch for LMS

Least-Mean-Squares

$$\min_{\mathbf{w}} f(\|\mathbf{X}\mathbf{w} - \mathbf{y}\|_2) + g(\mathbf{w}).$$

$$\min_{\mathbf{w}} f(\|\mathbf{Q}\mathbf{R}\mathbf{w} - \mathbf{y}\|_2) + g(\mathbf{w}).$$

$$\|\mathbf{X}\mathbf{w} - \mathbf{y}\|_2 = \|\mathbf{Q}\mathbf{R}\mathbf{w} - \mathbf{y}\|_2 = \|\mathbf{Q}\mathbf{R}\mathbf{w} - \mathbf{Q}\mathbf{Q}^T\mathbf{y}\|_2 = \|\mathbf{Q}\|_2 \, \|\mathbf{R}\mathbf{w} - \mathbf{Q}^T\mathbf{y}\|_2 = \|\mathbf{R}\mathbf{w} - \mathbf{Q}^T\mathbf{y}\|_2$$

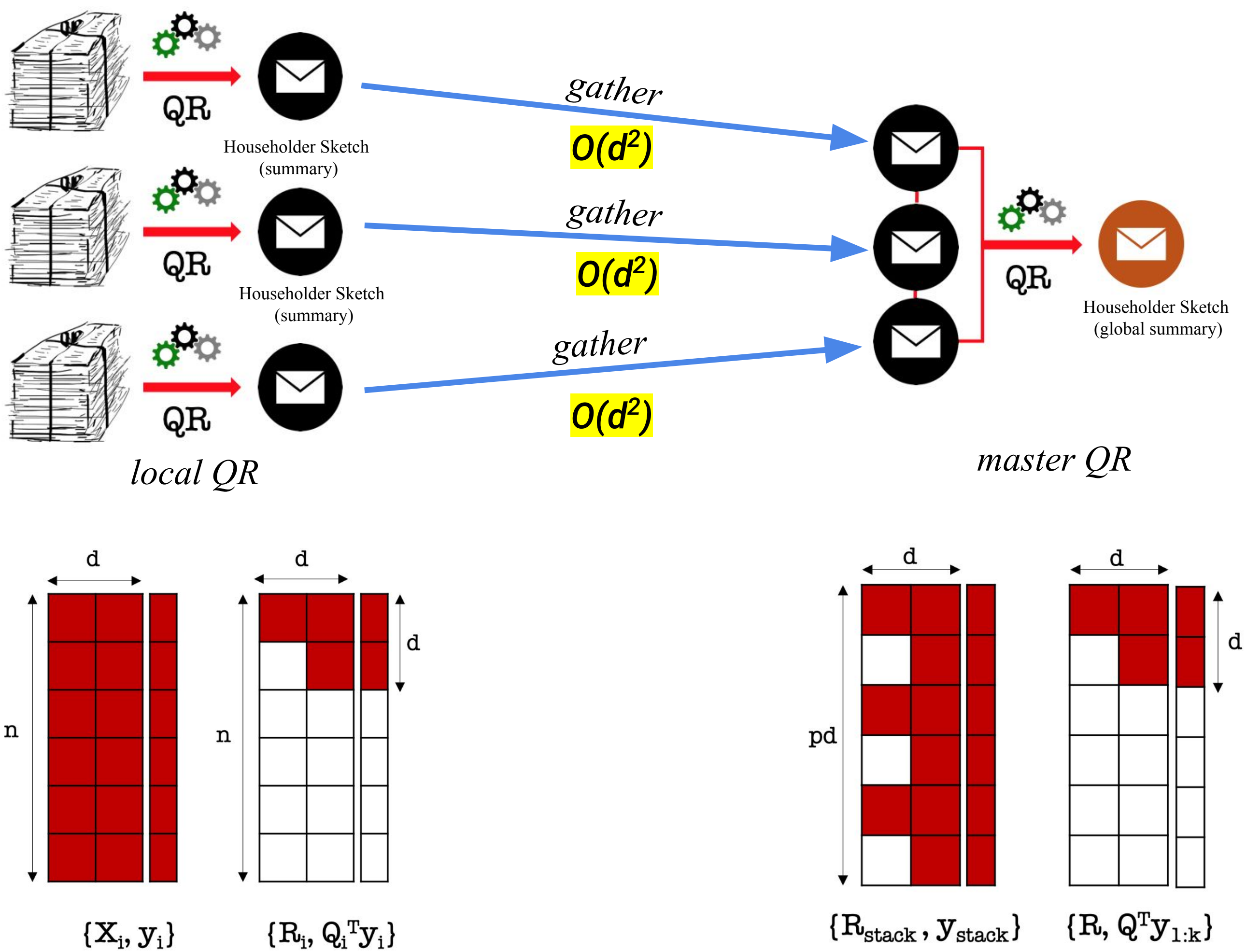
(LMS) Accurate Sketch $R^T R \cong X^T X$ (LMS-QR)

$$\mathbf{w} = \mathbf{X}^T\boldsymbol{\beta} = \mathbf{R}^T\mathbf{Q}^T\boldsymbol{\beta} = \mathbf{R}^T\bar{\boldsymbol{\beta}}$$

Distributed Householder Sketches

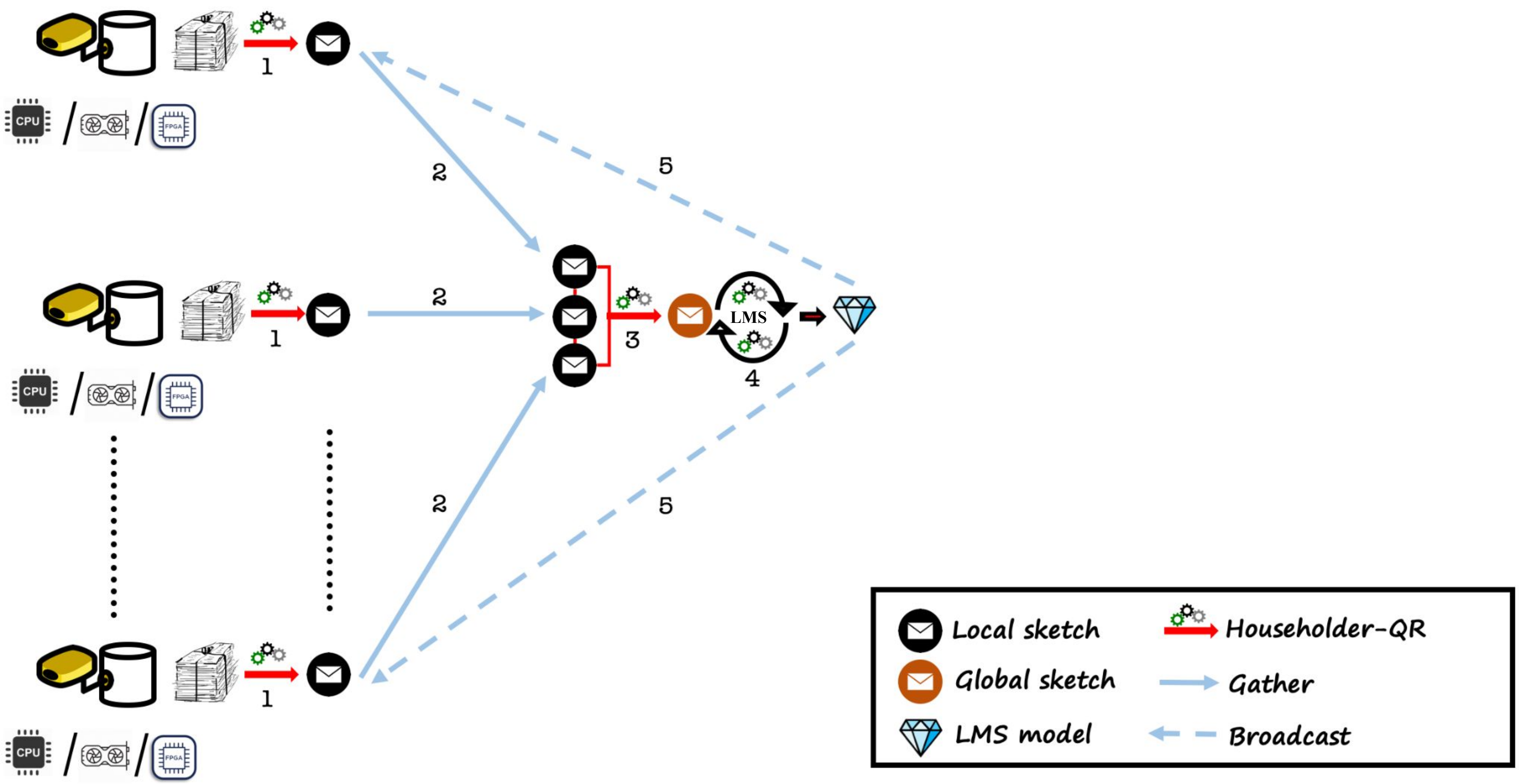
$$\mathbf{X} = \begin{pmatrix} \mathbf{X}_1 \\ \mathbf{X}_2 \\ . \\ . \\ \mathbf{X}_p \end{pmatrix} = \begin{pmatrix} \mathbf{Q}_1\mathbf{R}_1 \\ \mathbf{Q}_2\mathbf{R}_2 \\ . \\ . \\ \mathbf{Q}_p\mathbf{R}_p \end{pmatrix} = \mathtt{diag}(\mathbf{Q}_1, \ldots, \mathbf{Q}_p) \begin{pmatrix} \mathbf{R}_1 \\ \mathbf{R}_2 \\ . \\ . \\ \mathbf{R}_p \end{pmatrix}, \qquad \mathbf{R}_{stack} = \begin{pmatrix} \mathbf{R}_1 \\ \mathbf{R}_2 \\ . \\ . \\ \mathbf{R}_p \end{pmatrix} = \mathbf{Q}_M \mathbf{R}_M$$

$$\mathbf{X} = \mathtt{diag}(\mathbf{Q}_1, \ldots, \mathbf{Q}_p)\mathbf{R}_{stack} = \underbrace{\mathtt{diag}(\mathbf{Q}_1, \ldots, \mathbf{Q}_p)\mathbf{Q}_M}_{\mathbf{Q}}\mathbf{R}_M$$

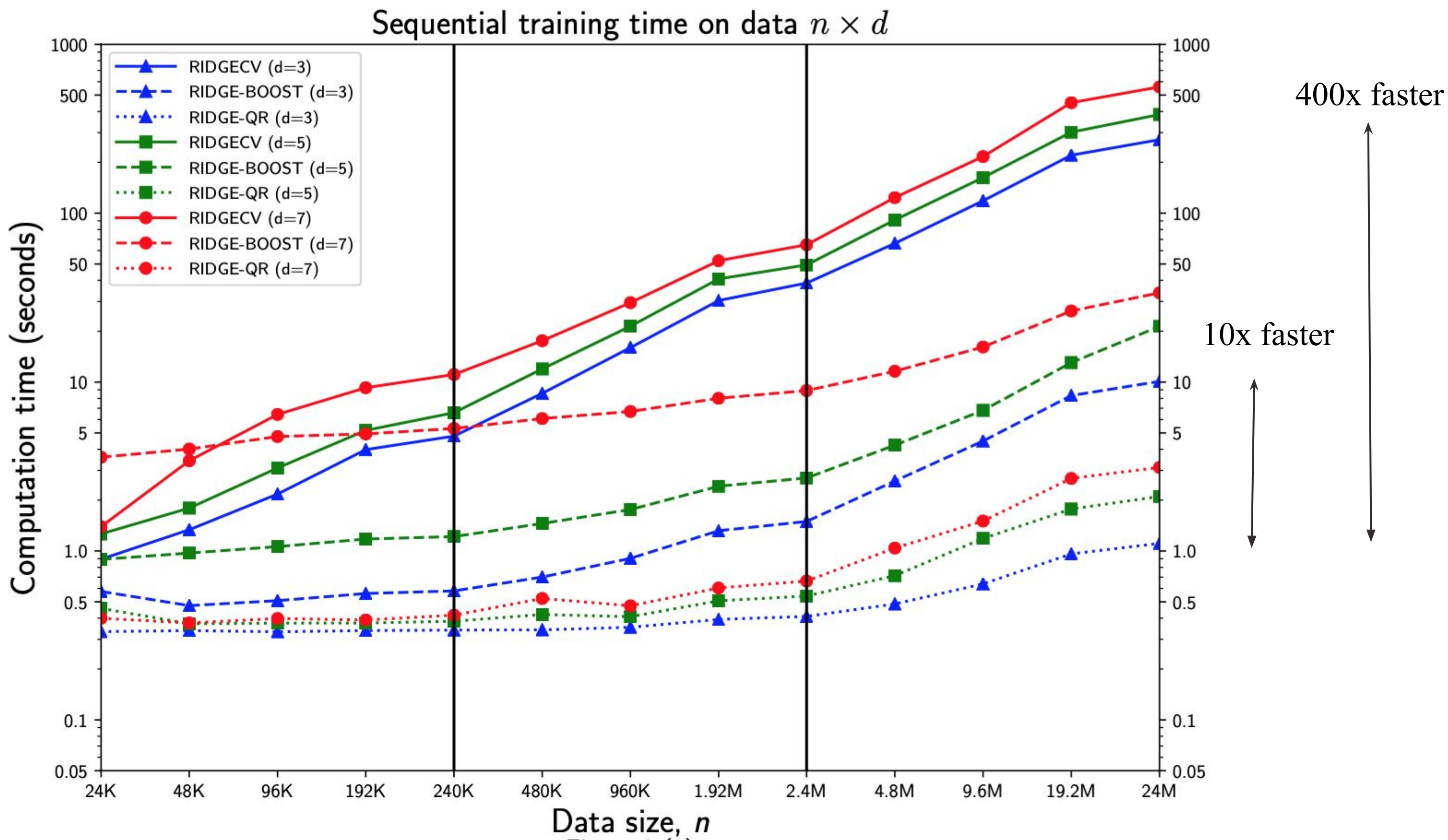


memory consumption $O(nd)$ $O(d^2)$ $O(pd^2)$ $O(d^2)$

Distributed Framework for LMS

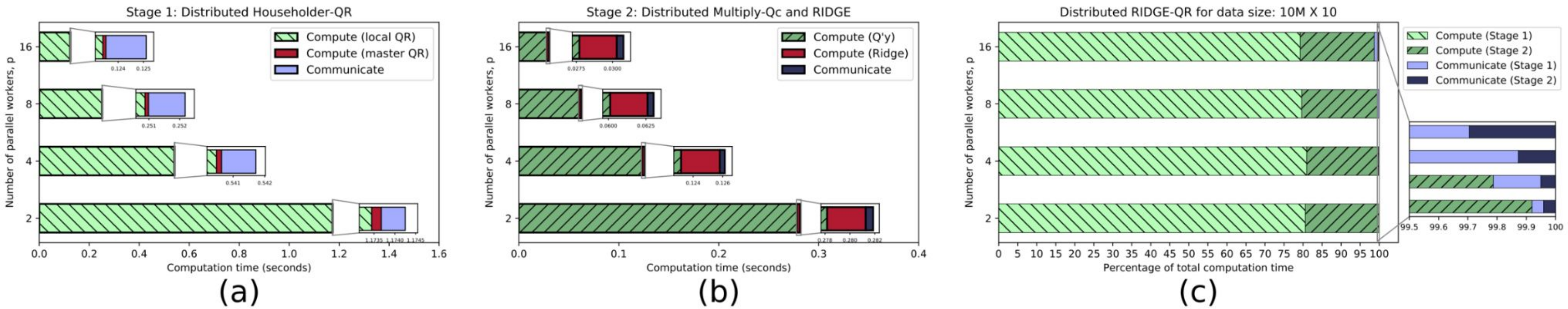


Results (1/2)



Sequential Training Time (RIDGE-QR vs others)

Results (2/2)



Breakdown of DISTRIBUTED RIDGE-QR (10M x 10 dataset) training time with zoomed insets depicting communication time (a): Stage 1: DISTRIBUTED HOUSEHOLDER-QR, (b): Stage 2: DISTRIBUTED MULTIPLY-QC and RIDGE, (c): Combined percentage

Research Contributions

- ✔ Relaxed Synchronization for Parallel QP Problems

[IEEE IPDPS'16] K. Lee, R. Bhattacharya, **J. Dass**, V. N. S. P. Sakuru, and R. N. Mahapatra, "*A Relaxed Synchronization Approach for Solving Parallel Quadratic Programming Problems with Guaranteed Convergence,*"

- ✔ Householder Sketch for Machine Learning

[ICML'21] **J. Dass**, and R. N. Mahapatra, "*Householder Sketch for Accurate and Accelerated Least-Mean-Squares Solvers,*"

- ❑ Memory-efficient Framework for Distributed ML
- ❑ Communication-efficient Framework for Scalable ML
- ❑ Multiple FPGA-based System for Energy-efficient ML
- ❑ Rapid Incremental Solver for Federated ML

Memory-efficient Framework for Distributed ML

Distributed Machine Learning

$\boldsymbol{K}$ *is kernel matrix which is generally non-separable*

(SVM) $$\min_{\boldsymbol{\alpha}} \frac{1}{2}\boldsymbol{\alpha}^T(\boldsymbol{X}\boldsymbol{X}^T + \rho\boldsymbol{I})\boldsymbol{\alpha} + \boldsymbol{e}^T\boldsymbol{\alpha}$$

subject to, $-\boldsymbol{I_n}\boldsymbol{\alpha} \leq \boldsymbol{0}_n$

By using Householder-QR, and $\hat{\square} = \boldsymbol{Q}^T\square$

(QRSVM) $$\min_{\widehat{\boldsymbol{\alpha}}} \frac{1}{2}\widehat{\boldsymbol{\alpha}}^T(\boldsymbol{R}\boldsymbol{R}^T + \rho\boldsymbol{I})\widehat{\boldsymbol{\alpha}} + \hat{\boldsymbol{e}}^T\widehat{\boldsymbol{\alpha}}$$

subject to, $-\mathbf{Q}\widehat{\boldsymbol{\alpha}} \leq \boldsymbol{0}_n$

Memory-efficient Distributed ML

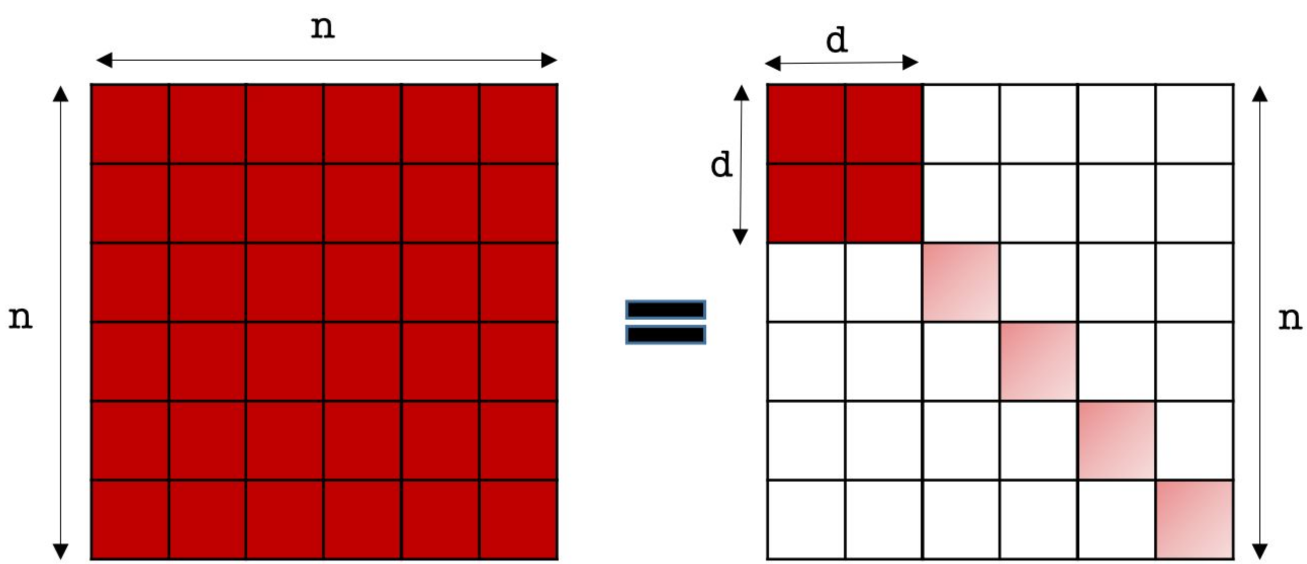
$\boldsymbol{K}$ *is kernel matrix which is generally non-separable*

$$\min_{\boldsymbol{\alpha}} \frac{1}{2} \boldsymbol{\alpha}^T (\boldsymbol{X}\boldsymbol{X}^T + \rho \boldsymbol{I}) \boldsymbol{\alpha} + \boldsymbol{e}^T \boldsymbol{\alpha}$$

subject to, $-\boldsymbol{I}_{\boldsymbol{n}} \boldsymbol{\alpha} \le \boldsymbol{0}_n$

$$\min_{\widehat{\boldsymbol{\alpha}}} \frac{1}{2} \widehat{\boldsymbol{\alpha}}^T (\boldsymbol{R}\boldsymbol{R}^T + \rho \boldsymbol{I}) \widehat{\boldsymbol{\alpha}} + \hat{\boldsymbol{e}}^T \widehat{\boldsymbol{\alpha}}$$

subject to, $-\mathbf{Q} \widehat{\boldsymbol{\alpha}} \le \boldsymbol{0}_n$



$\boldsymbol{X}\boldsymbol{X}^{\mathrm{T}} + \rho \boldsymbol{I}$	$\mathbf{R}\mathbf{R}^{\mathrm{T}} + \rho \mathbf{I}$
Non - Separable	Block-Separable

memory consumption: *Dense O(n^2)* | *Sparse O(d^2)*

Parallel Dual Ascent

Step 1: Minimization of Lagrangian - In Parallel

At core $i = \{1, \ldots, p\}$

$$\hat{\boldsymbol{\alpha}}_{\boldsymbol{i}}{}^{t+1} = \boldsymbol{F_i}{}^{-1}(-\hat{\boldsymbol{\beta}}_{\boldsymbol{i}}{}^{t} + \hat{\boldsymbol{e}}_{\boldsymbol{i}})$$

where,

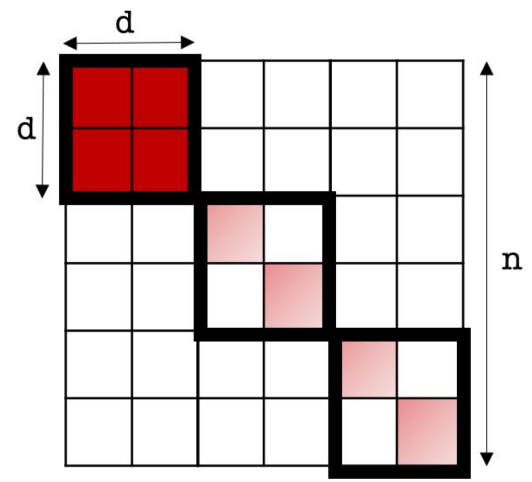
$$\boldsymbol{F_i}{}^{-1} = \begin{cases} \boldsymbol{F_1}{}^{-1} & \text{if} \quad i = 1 \\ -2C & \text{if} \quad i = 2, \ldots, p \end{cases}$$

Step 2: Dual variable update - In Parallel

At core i

$$\hat{\boldsymbol{\beta}}_{\boldsymbol{i}}{}^{t+1} = \hat{\boldsymbol{\beta}}_{\boldsymbol{i}}{}^{t} + \eta^{\star}(-\hat{\boldsymbol{\alpha}}_{\boldsymbol{i}}{}^{t+1})$$

where, $\eta^{\star}$ is the optimal step size, and $\hat{\beta}^{t} = \boldsymbol{Q}^{T}\beta^{t}$

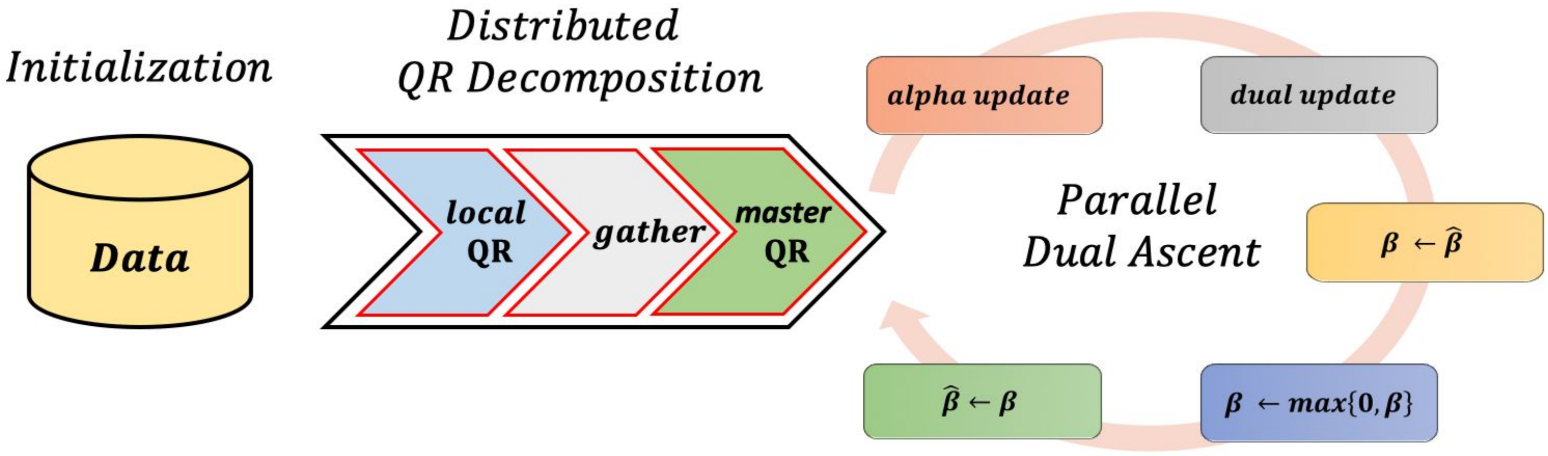


F = diag(F$_1$, F$_2$, F$_3$)

$$(\boldsymbol{R}\boldsymbol{R}^{T} + \rho \boldsymbol{I})$$

Householder-QR based QRSVM renders SVM separable into independent sub-problems!

Workflow


Initialization
Data
Distributed
QR Decomposition
local
QR
gather
master
QR
alpha update
dual update
Parallel
Dual Ascent
$\beta \leftarrow \hat{\beta}$
$\hat{\beta} \leftarrow \beta$
$\beta \leftarrow max\{0, \beta\}$

Results

Hardware

- *Ada* Supercomputing Cluster at TAMU
- Intel Xeon E5-2670 v2 (Ivy Bridge-EP), 10-core, 2.5GHz
- 64 GB/node and 16 cores/node
- Message-Passing Interface (MPI), InfiniBand interconnect

Dataset	n	d	**Description**
a9a	32560	123	predict annual income
covtype	464810	54	predict forest cover type

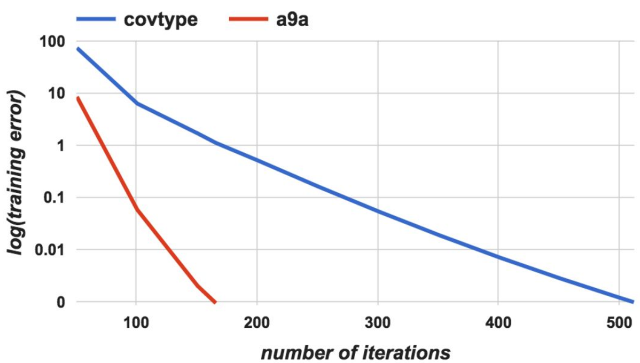
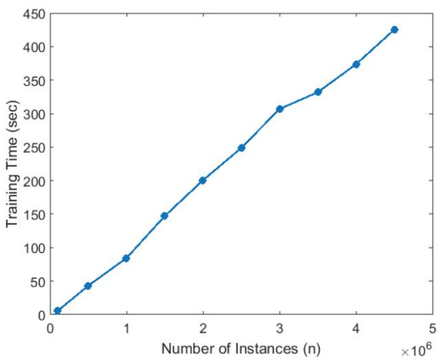
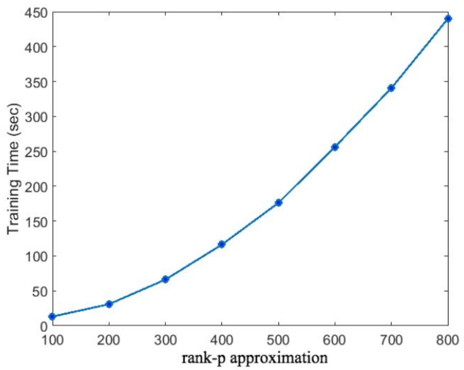


Figure: a9a: t=166 , covtype: t=512, threshold=10^{-3}

Convergence



Linear scaling with #samples



Quadratic scaling with rank

Research Contributions

✔ Relaxed Synchronization for Parallel QP Problems

[IEEE IPDPS'16] K. Lee, R. Bhattacharya, **J. Dass**, V. N. S. P. Sakuru, and R. N. Mahapatra, "*A Relaxed Synchronization Approach for Solving Parallel Quadratic Programming Problems with Guaranteed Convergence*"

✔ Householder Sketch for Machine Learning

[ICML'21] J. Dass, and R. N. Mahapatra, "*Householder Sketch for Accurate and Accelerated Least-Mean-Squares Solvers*"

✔ Memory-efficient Framework for Distributed ML

[ICDCS'17] J. Dass, V. N. S. P. Sakuru, V. Sarin and R. N. Mahapatra, "Distributed QR Decomposition Framework for Training Support Vector Machines"

❑ Communication-efficient Framework for Scalable ML

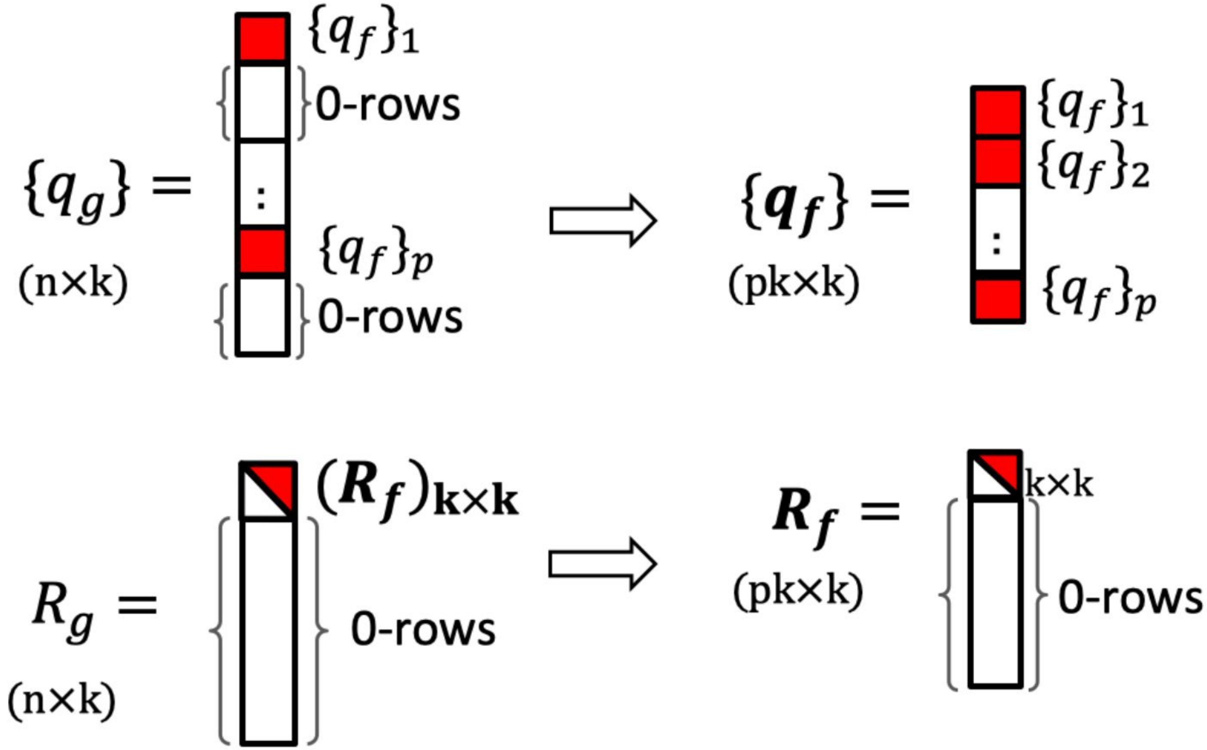
❑ Multiple FPGA-based System for Energy-efficient ML

❑ Rapid Incremental Solver for Federated ML

Communication-efficient Framework for Scalable ML

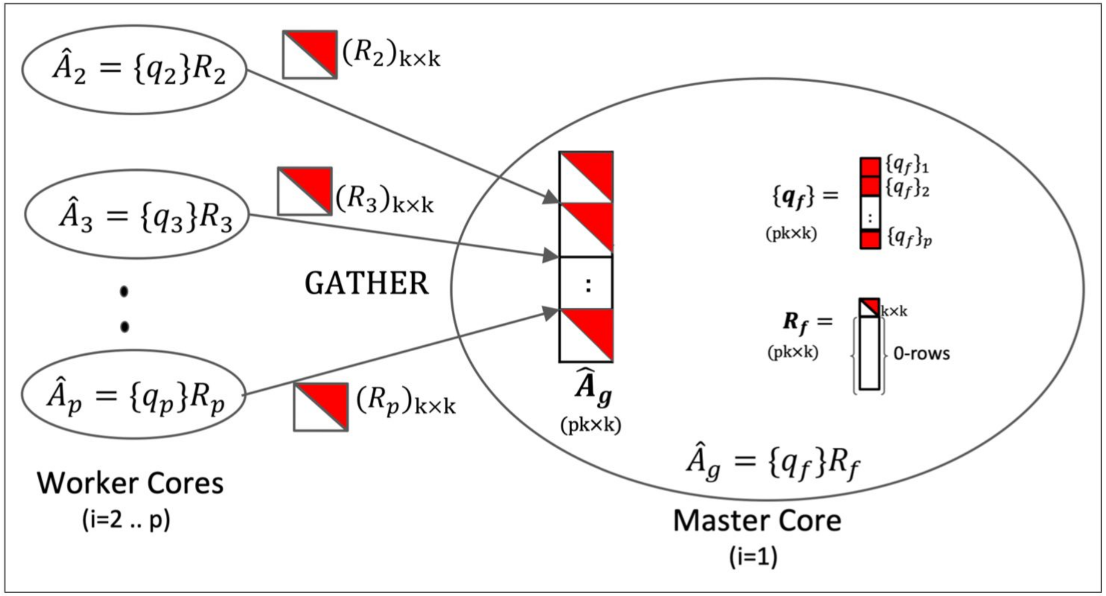
01	Protect Data Privacy
02	Reduce Latency
03	Save Bandwidth
04	Energy Efficiency
05	Build Robust Models
06	Streaming Data

Improved Master QR



At master core: Memory improvement and representation of QR factors

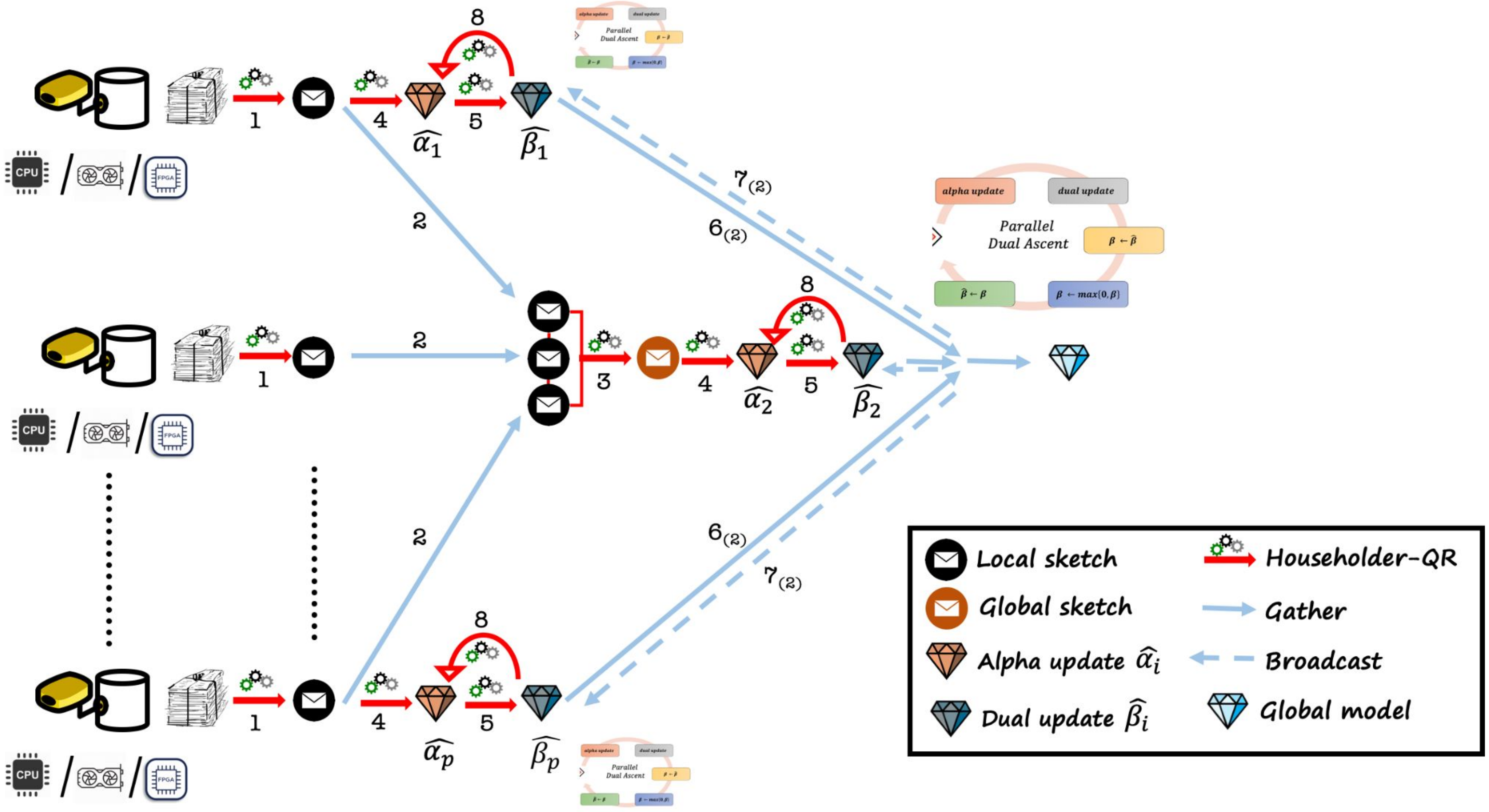
Communication-Efficient Workflow



Communication Efficient ⇒ High Scalability

- During every iteration of parallel dual ascent, each Gather and Scatter involves communicating **negligibly small** $O(k)$ data volume per core with Master core, where, $k \ll \frac{n}{p}$
- Scalable with number of cores (p) and handle larger datasets than previous implementation

Communication-efficient Framework for Scalable ML



Results (1/2)

TABLE : Benchmark Dataset Description

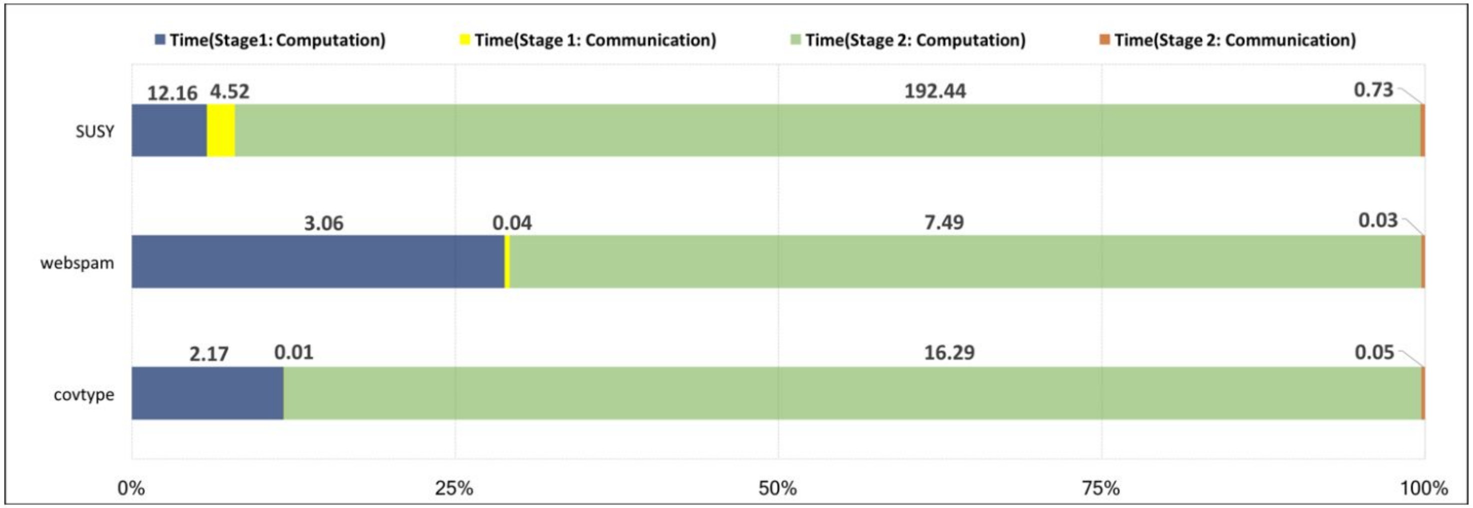
Dataset	#training samples (n)	#features (d)	*k*-rank approx.
covtype.binary	464,810	54	64
webspam(unigram)	350,000	254	128
SUSY	5,000,000	18	128

TABLE : Scalability of distributed-QRSVM.
Training time (in seconds) and Speedup, S_p wrt sequential-QRSVM (S_p is shown in parenthesis).

Dataset	sequential	p=2	p=4	p=8	p=16	p=32	p=64
covtype	268 (1x)	132 (2x)	64 (4x)	33 (8x)	18 (15x)	10 (27x)	6 (45x)
webspam	258 (1x)	120 (2x)	58 (4x)	32 (8x)	19 (14x)	11 (23x)	9 (29x)
SUSY	28,614 (1x)	11,284 (2x)	4,405 (7x)	1,686 (17x)	804 (36x)	380 (75x)	210 (136x)

Near linear scalability on large datasets with increasing parallel workers

Results (2/2)



Overall training time, T_{train} analysis (in seconds) for benchmarks covtype (p = 16), webspam (p = 32) and SUSY (p = 64).

Communication overhead is Negligible.

Research Contributions

✔ Relaxed Synchronization for Parallel QP Problems

[IEEE IPDPS'16] K. Lee, R. Bhattacharya, **J. Dass**, V. N. S. P. Sakuru, and R. N. Mahapatra, "*A Relaxed Synchronization Approach for Solving Parallel Quadratic Programming Problems with Guaranteed Convergence*"

✔ Householder Sketch for Machine Learning

[ICML'21] J. Dass, and R. N. Mahapatra, "*Householder Sketch for Accurate and Accelerated Least-Mean-Squares Solvers*"

✔ Memory-efficient Framework for Distributed ML

[IEEE ICDCS'17] J. Dass, V. N. S. P. Sakuru, V. Sarin and R. N. Mahapatra, "*Distributed QR Decomposition Framework for Training Support Vector Machines*"

✔ Communication-efficient Framework for Scalable ML

[IEEE TPDS'18] J. Dass, V. Sarin and R. N. Mahapatra "*Fast and Communication-Efficient Algorithm for Distributed Support Vector Machine Training*"

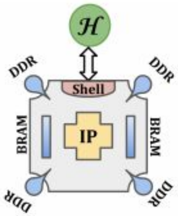
❑ Multiple FPGA-based System for Energy-efficient ML

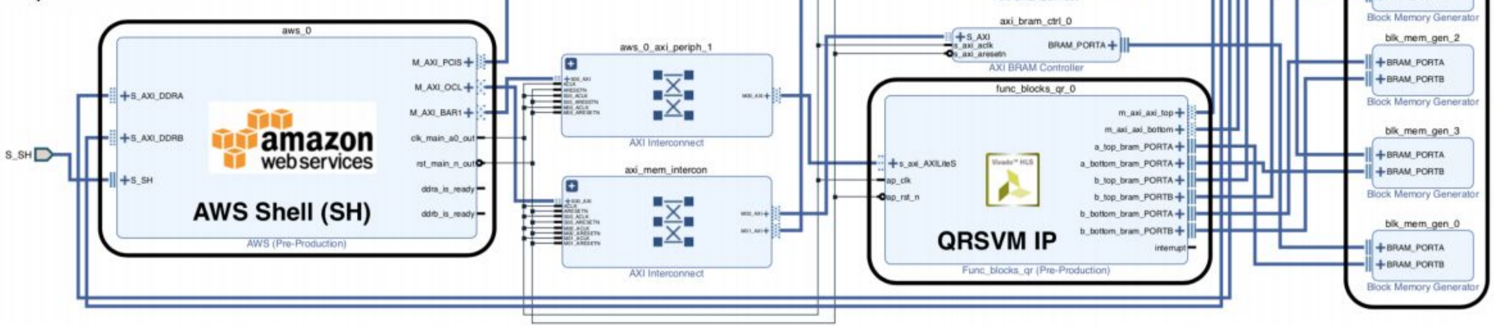
❑ Rapid Incremental Solver for Federated ML

Multiple FPGA-based System for Energy-efficient ML

01	Protect Data Privacy
02	Reduce Latency
03	Save Bandwidth
04	Energy Efficiency
05	Build Robust Models
06	Streaming Data

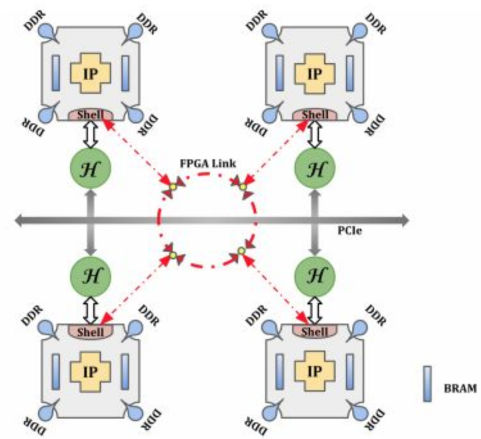
Multiple FPGA-based System





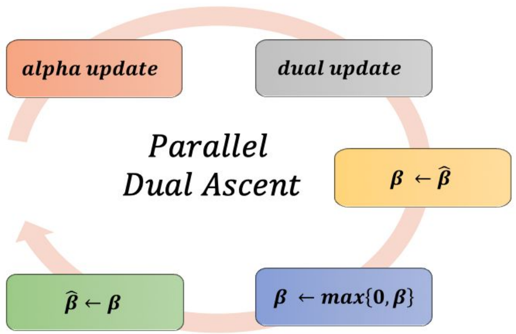
Multiple FPGA Network

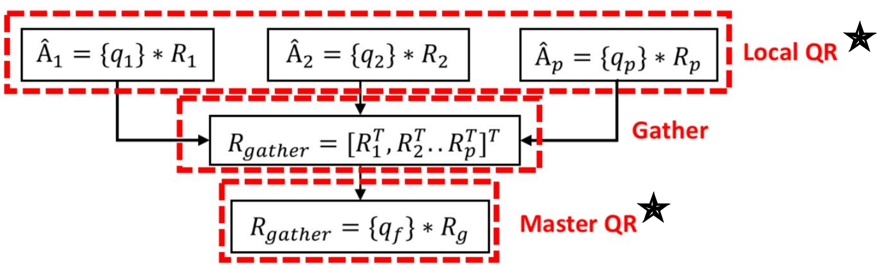
- Each edge device comprises of FPGA IP logic + Host processor
- Illustration for $p = 4$ edges in a network
- A single QRSVM IP is synthesized per FPGA device to operate at clock frequency of *125 MHz* with *39 Watts* of power
- Computational workload is entirely with the FPGA while communication is through Host processor (*PCIe*)
- Each FPGA handles maximum of *256K* samples

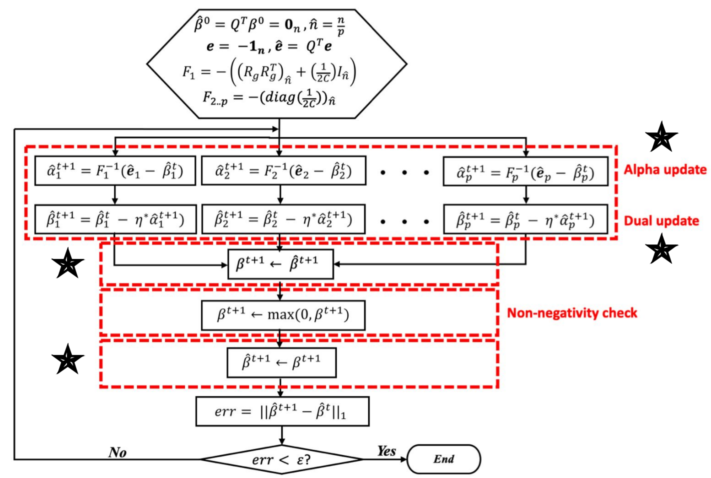


Computational Workflow

Distributed QR Decomposition







Inner Product and SAXPY

Algorithm 6.1 $\{\mathbf{q}_i\}, \mathbf{R}_i \leftarrow \hat{\mathbf{A}}_i$, via Householder algorithm

Data: Matrix $\hat{\mathbf{A}}_i$

1 $\mathbf{Q}_{\hat{n}\times k}, (\hat{\mathbf{A}}_i)_{\hat{n}\times k}$ ▷ $\hat{n}$: samples per compute unit
2 **for** $j \leftarrow 1$ *to* k **do**
3 $\quad \mathbf{q}_{ij} \leftarrow \hat{\mathbf{A}}_i(j:\hat{n}, j)$
4 $\quad \mathbf{q}_{ij}(1) \leftarrow \mathbf{q}_{ij}(1) + sign(\mathbf{q}_{ij}(1)) \times \|\mathbf{q}_{ij}\|_2$ ▷ scalar update
5 $\quad \mathbf{q}_{ij} \leftarrow \frac{\mathbf{q}_{ij}}{\|\mathbf{q}_{ij}\|_2}$ ▷ vector normalization
6 $\quad \hat{\mathbf{A}}_i(j:\hat{n}, j:k) \leftarrow \hat{\mathbf{A}}_i(j:\hat{n}, j:k) - 2\mathbf{q}_{ij} < \mathbf{q}_{ij}, \hat{\mathbf{A}}_i(j:\hat{n}, j:k) >$ ▷ *Algorithm 6.2*
7 $\quad \mathbf{R}_i = \hat{\mathbf{A}}_i(j:\hat{n}, j:k)$
8 **end**
9 $\{\mathbf{q}_i\} \leftarrow [\mathbf{q}_{i1}, \mathbf{q}_{i2}, \ldots, \mathbf{q}_{ik}]$ ▷ set of k-reflectors

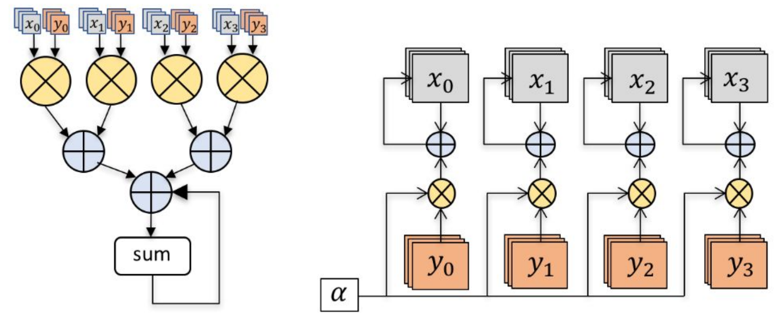
Algorithm 6.2 Computing Step 6 in Algorithm 6.1

Data: Matrix $\hat{\mathbf{A}}_i$

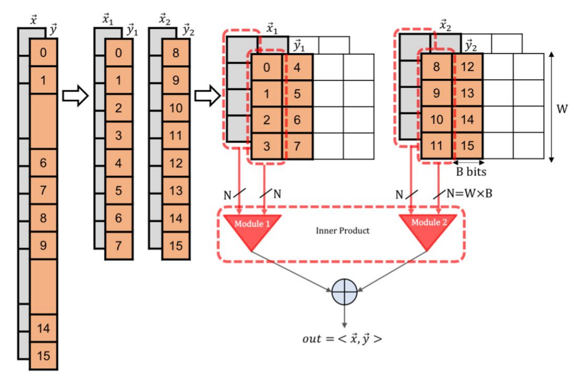
▷ $\hat{\mathbf{A}}_i(j:\hat{n}, j:k) \leftarrow \hat{\mathbf{A}}_i(j:\hat{n}, j:k) - 2\mathbf{q}_{ij} < \mathbf{q}_{ij}, \hat{\mathbf{A}}_i(j:\hat{n}, j:k) >$
for $m \leftarrow j$ *to* k **do**
$\quad \mathbf{a}_{BRAM} \leftarrow \mathbf{A}(j:\hat{n}, m)$ ▷ Load into BRAM
$\quad$ Compute $sum = < \mathbf{q}_{ij}, \mathbf{a}_{BRAM} >$ ▷ Inner Product
$\quad \mathbf{a}_{BRAM} \leftarrow \mathbf{a}_{BRAM} - 2 \times sum \times \mathbf{q}_{ij}$ ▷ saxpy
$\quad \mathbf{A}(j:\hat{n}, m) \leftarrow \mathbf{a}_{BRAM}$ ▷ Write to DDR
end

$$\boldsymbol{\beta} = \mathbf{Q}\hat{\boldsymbol{\beta}} = \mathrm{diag}(\mathbf{Q}_1, \mathbf{Q}_2, ..\mathbf{Q}_i.., \mathbf{Q}_p) \times \mathbf{Q}_g \times \hat{\boldsymbol{\beta}}$$

$$\hat{\boldsymbol{\beta}}_{i[j:\hat{n}]} \leftarrow \hat{\boldsymbol{\beta}}_{i[j:\hat{n}]} - 2\mathbf{q}_{ij} < \mathbf{q}_{ij}, \hat{\boldsymbol{\beta}}_{i[j:\hat{n}]} > \qquad j \leftarrow k \text{ to } 1$$



SIMD hardware kernels for (a) inner product, < x , y > (b) saxpy, x = x + αy. The vectorized kernels process W = 4 elements in each pass with pipeline stages, D=2 (inner product)

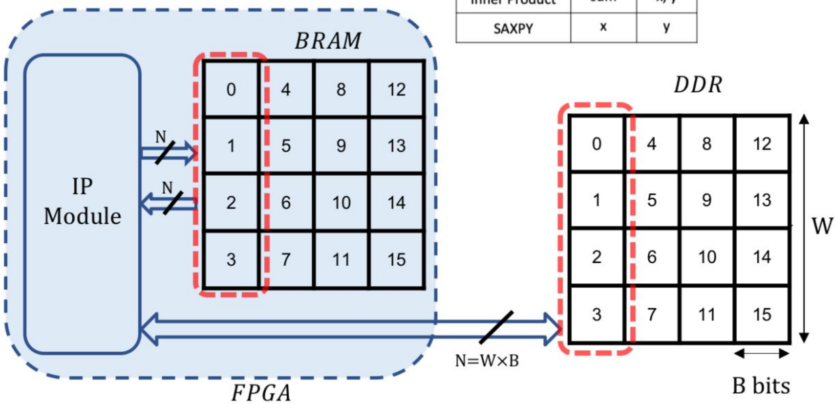


Doubling Throughput for Inner Product

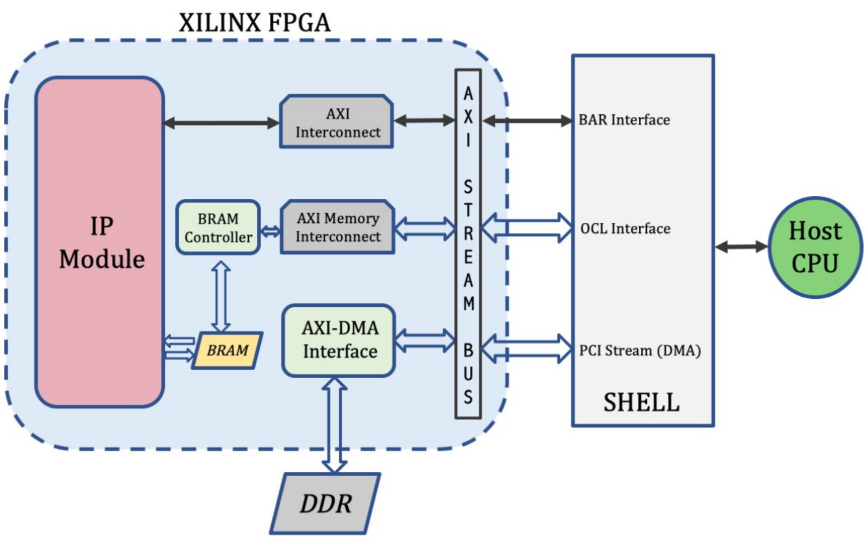
Data Layout + Memory Interface

Memory Assignment

Kernel	BRAM	DDR
Inner Product	sum	x, y
SAXPY	x	y



Data layout in column-major order and memory interface for on-chip Block RAM (Full-duplex) and off-chip DDR (Half-duplex) with the IP. **4MB to synthesize 2x2MB BRAMs on each FPGA. Can store 256K data samples**



FPGA Block Diagram with memory interfacing. Maximum bus width supported by this interface is N = 1024 bits. Support for double-precision (B=64 bits) floating point numbers without any loss of accuracy when compared to software implementation, the **maximum parallel compute units, $W = \lfloor N/B \rfloor = 16$**

Results (1/3)

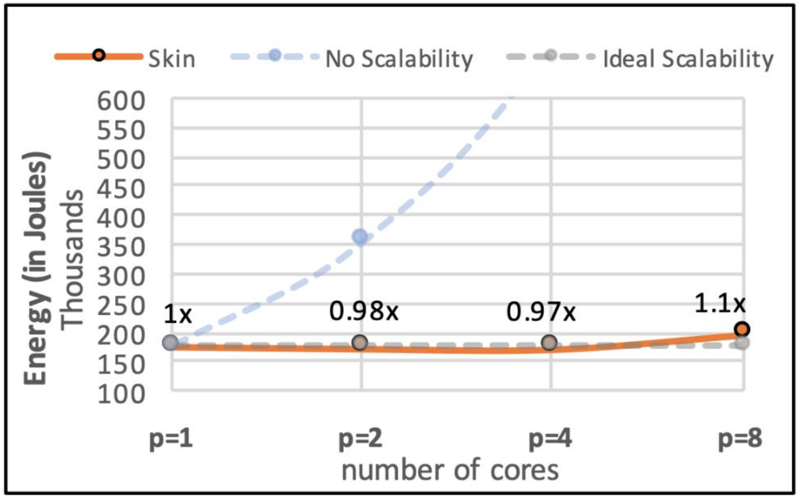
Benchmark	Application	#samples (n)	#features (d)	*k*-rank
MNIST	Image	60,000	780	128
Skin	Health	200,000	3	64
Webspam	Email	350,000	254	128
Covtype	Geography	464,810	54	64
SUSY	Physics	2,000,000	18	128

Datasets

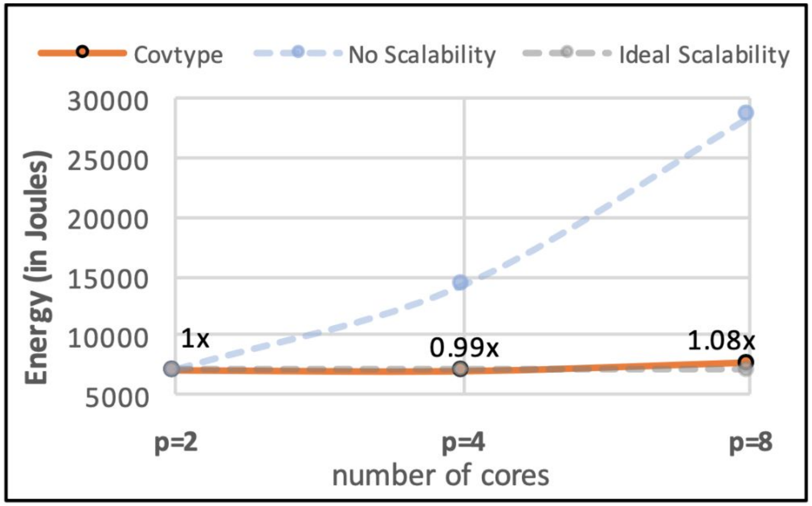
Resource	BRAM	DSP	FF	LUT
Used	1405	1221	545248	449113
Available	2160	6840	2363536	1181768
Utilization	65%	18%	23%	38%

Utilization for FPGA Xilinx Virtex xcvu9p-flgb2104-2-i

Results (2/3)



(a)



(b)

Energy consumption Analysis of Multi-FPGA system under strong scaling scenario

Benchmarks: (a) Skin and (b) Covtype

Nearly CONSTANT (ideal) energy consumption across #FPGA units
Fully-Parallel Implementation

Results (3/3)

TABLE 4
Comparison With Embedded Edge Processor (ARM Cortex A15) and Cloud Processor (Broadwell) Platforms

#units p	MNIST T_p^{FPGA}	T_p^{ARM}	T_p^{Broad}	E_p^{FPGA}	E_p^{ARM}	E_p^{Broad}	Skin T_p^{FPGA}	T_p^{ARM}	T_p^{Broad}	E_p^{FPGA}	E_p^{ARM}	E_p^{Broad}
1	**10.92**	31.06	18.78	**0.43**	0.44	2.72	**4,536**	21,773	7,167	**177**	305	1,039
2	**5.84**	20.12	8.25	**0.45**	0.56	2.40	**2,228**	6,121	3,093	**174**	172	897
4	**3.58**	12.9	4.35	**0.56**	0.72	2.52	**1,108**	3,044	1,607	**173**	170	932

#units p	Webspam T_p^{FPGA}	T_p^{ARM}	T_p^{Broad}	E_p^{FPGA}	E_p^{ARM}	E_p^{Broad}	Covtype T_p^{FPGA}	T_p^{ARM}	T_p^{Broad}	E_p^{FPGA}	E_p^{ARM}	E_p^{Broad}
1	-	895	236.54	-	12.5	34.30	-	1079	292.63	-	15	42.43
2	**76.14**	477	133.99	**5.9**	13.4	38.86	**91.45**	520	160.08	**7.1**	14.5	46.42
4	**39.36**	254	65.92	**6.1**	14.2	38.23	**45.36**	251	77.19	**7**	14	44.77

#units p	#samples n	SUSY T_p^{FPGA}	T_p^{ARM}	T_p^{Broad}	E_p^{FPGA}	E_p^{ARM}	E_p^{Broad}
1	250K	**108.08**	2452	171.29	**4.2**	34.3	24.84
2	500K	**131.02**	3131	232.01	**10.2**	87.7	67.28
4	1M	**176.18**	4189	319.03	**27.5**	234	185

In a multiple compute system, #units, p, corresponds to #FPGA units (QRSVM IP cores), #ARM processors, and #Broadwell processors. Training time (in s), T_p^{FPGA}, T_p^{ARM}, and T_p^{Broad}. Energy consumption (in kJ), E_p^{FPGA}, E_p^{ARM}, and E_p^{Broad}

Broadwell at 145W
ARM at 14W
FPGA at 39W

Multi-FPGA implementation is upto
1.7x faster and **6x lower energy** than the cloud processor (Broadwell)
3x-24x faster and **2x-8x lower energy** than edge processor (ARM)

Research Contributions

✔ Relaxed Synchronization for Parallel QP Problems

[IEEE IPDPS'16] K. Lee, R. Bhattacharya, **J. Dass**, V. N. S. P. Sakuru, and R. N. Mahapatra, "*A Relaxed Synchronization Approach for Solving Parallel Quadratic Programming Problems with Guaranteed Convergence*"

✔ Householder Sketch for Machine Learning

[ICML'21] J. Dass, and R. N. Mahapatra, "*Householder Sketch for Accurate and Accelerated Least-Mean-Squares Solvers*"

✔ Memory-efficient Framework for Distributed ML

[IEEE ICDCS'17] J. Dass, V. N. S. P. Sakuru, V. Sarin and R. N. Mahapatra, "*Distributed QR Decomposition Framework for Training Support Vector Machines*"

✔ Communication-efficient Framework for Scalable ML

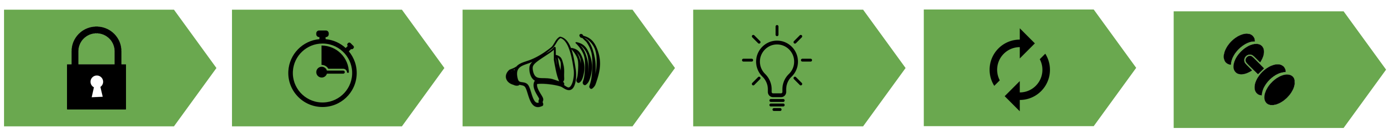
[IEEE TPDS'18] J. Dass, V. Sarin and R. N. Mahapatra "*Fast and Communication-Efficient Algorithm for Distributed Support Vector Machine Training*"

✔ Multiple FPGA-based System for Energy-efficient ML

[ACM FPGA'19 | IEEE TC'20] J.Dass, Y. Narawane, R. N. Mahapatra, and V. Sarin, "*Distributed Training of Support Vector Machine on a Multiple-FPGA System*"

❑ Rapid Incremental Solver for Federated ML

Rapid Incremental Solver for Federated ML

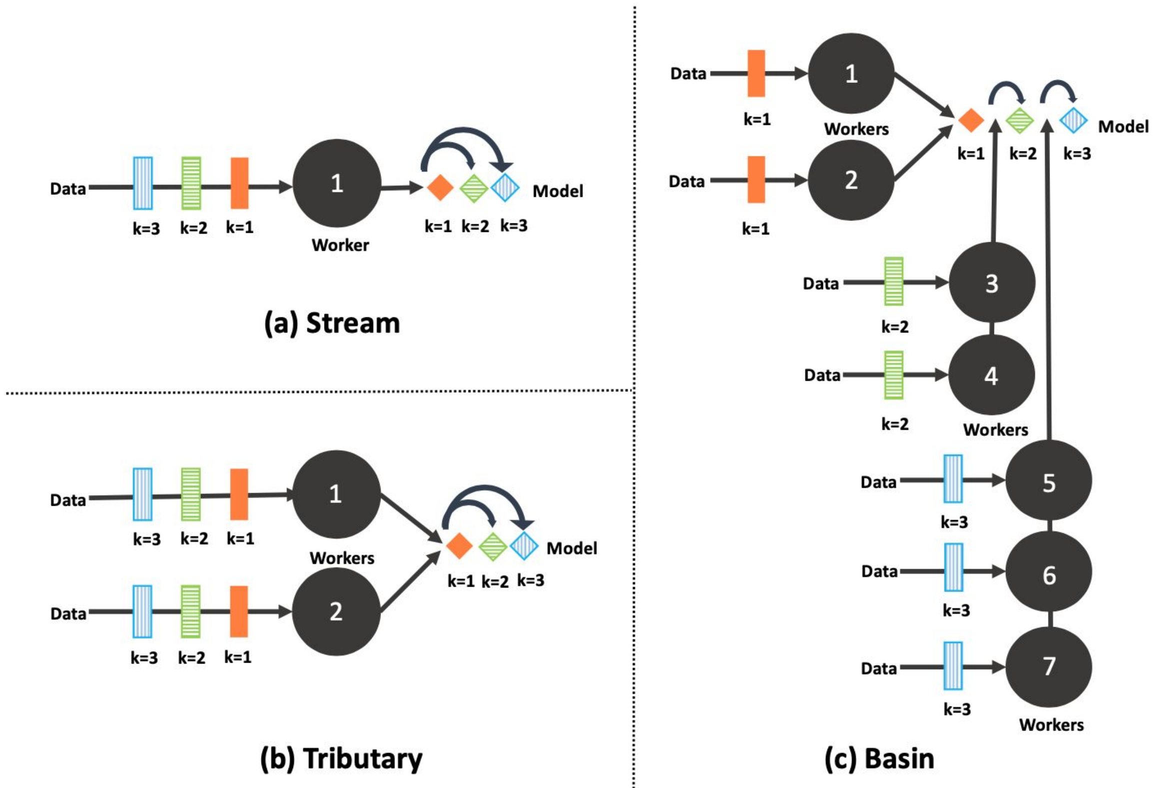


01	Protect Data Privacy
02	Reduce Latency
03	Save Bandwidth
04	Energy Efficiency
05	Build Robust Models
06	Streaming Data

Incremental Federated ML Requirements

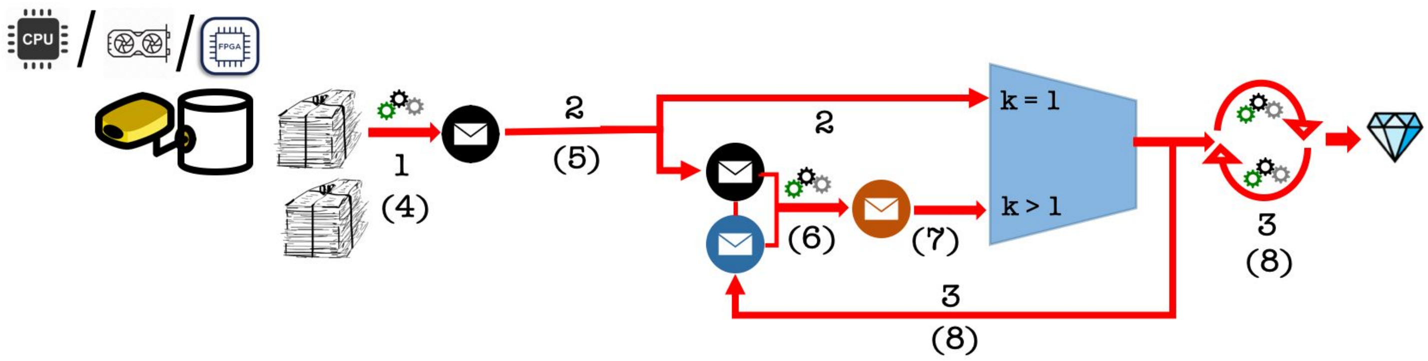
1. Data should **never be saved** on a centralized server **nor shared** among peers
2. Data samples must **not be stored** between successive model updates, i.e., update using **current data only**
3. **No retraining** from scratch is allowed
4. **Accurately** solve for global model in **collaboration** with other workers
5. **Robustness** and **Fault-tolerant** to straggling/offline workers

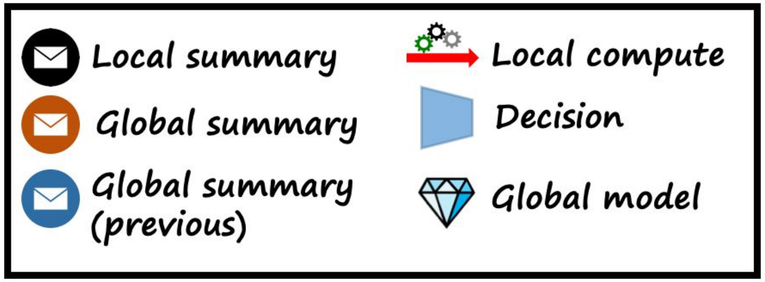
RIVER Setups



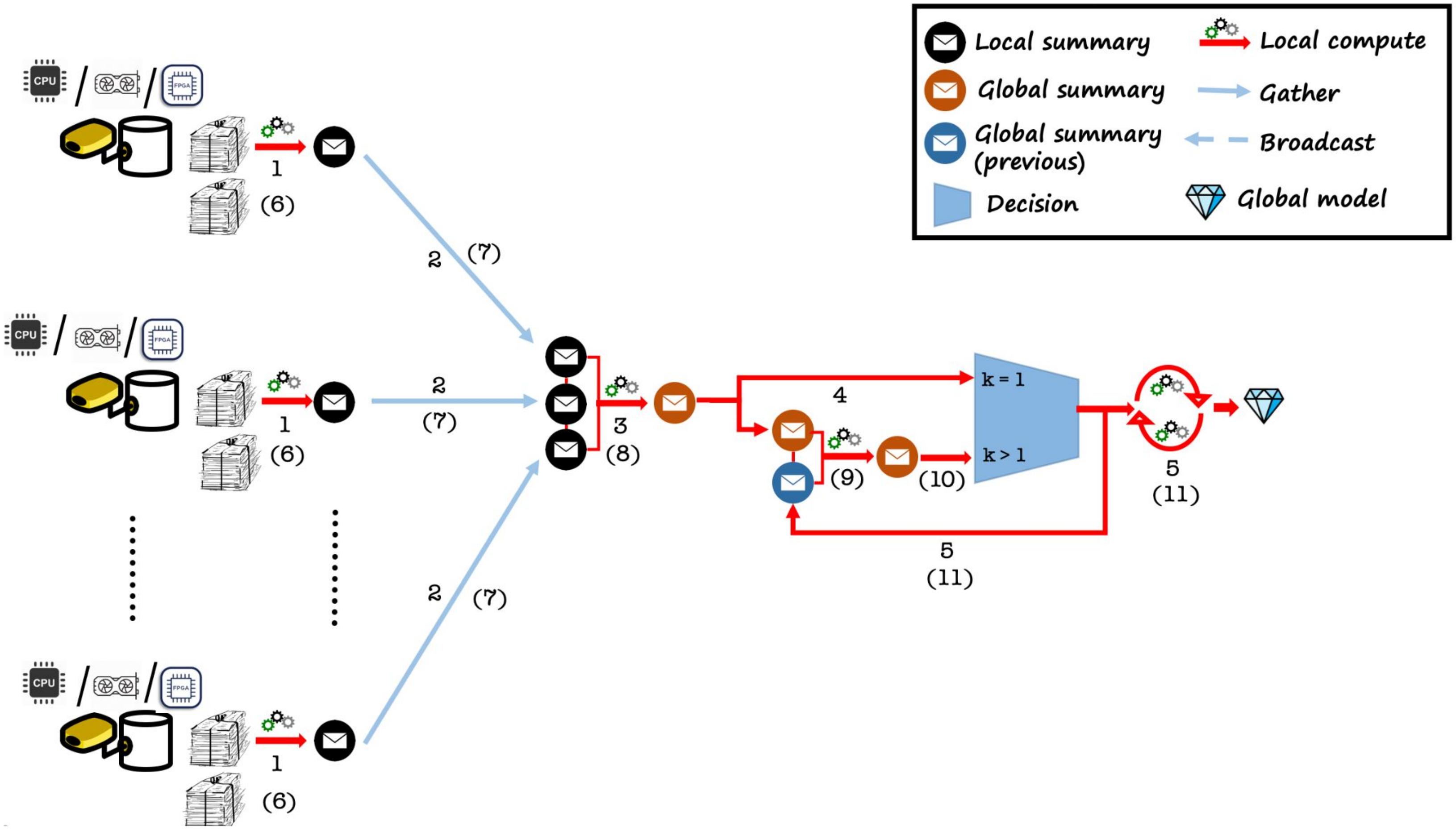
Various setups, round k $\in$ [3] **(a) Stream:** only one worker, where data is generated in every round **(b) Tributary:** fixed number of workers, where data is generated by each worker in every round **(c) Basin:** dynamic number of workers, where in each round different groups of workers participate in the network with their data

RIVER-Stream

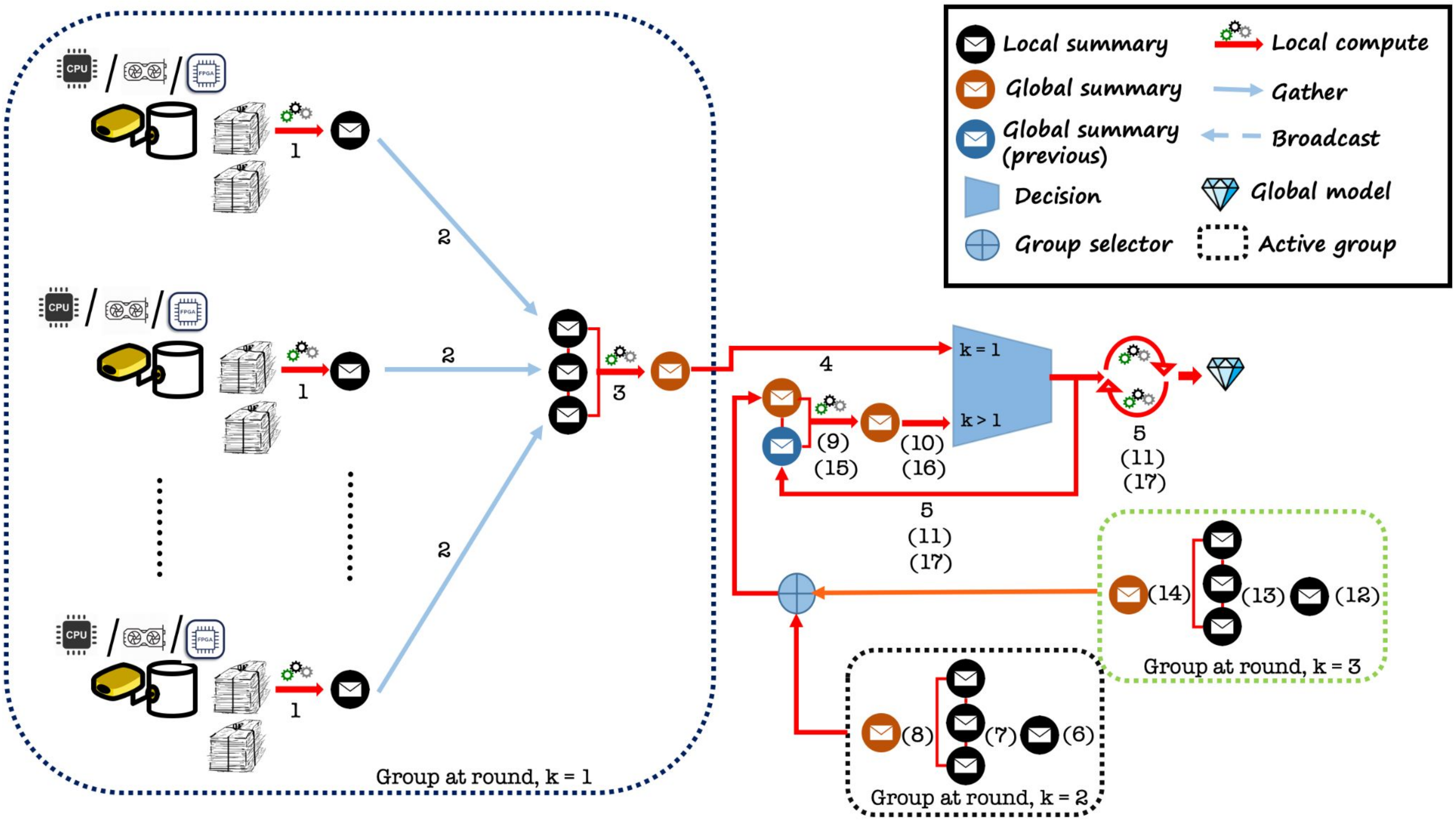




RIVER-Tributary

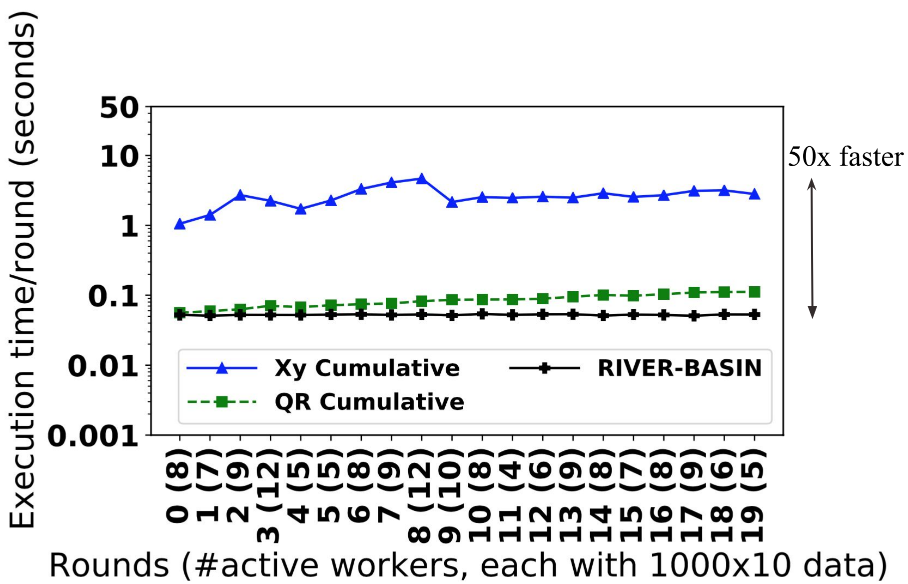


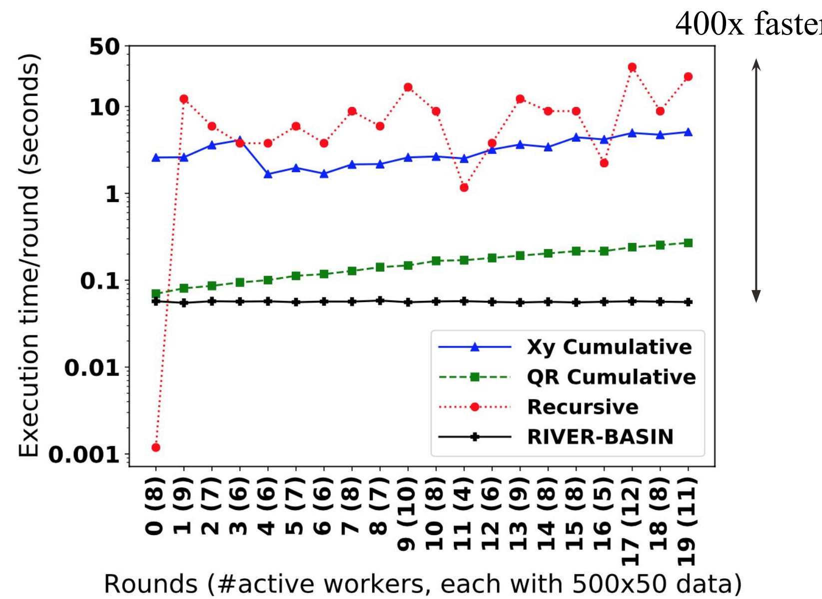
RIVER-Basin



Local summary
Local compute
Global summary
Gather
Global summary (previous)
Broadcast
Decision
Global model
Group selector
Active group
CPU
FPGA
1
2
3
4
k = 1
k > 1
(9)
(15)
(10)
(16)
5
(11)
(17)
(14)
(13)
(12)
Group at round, k = 3
(8)
(7)
(6)
Group at round, k = 2
Group at round, k = 1

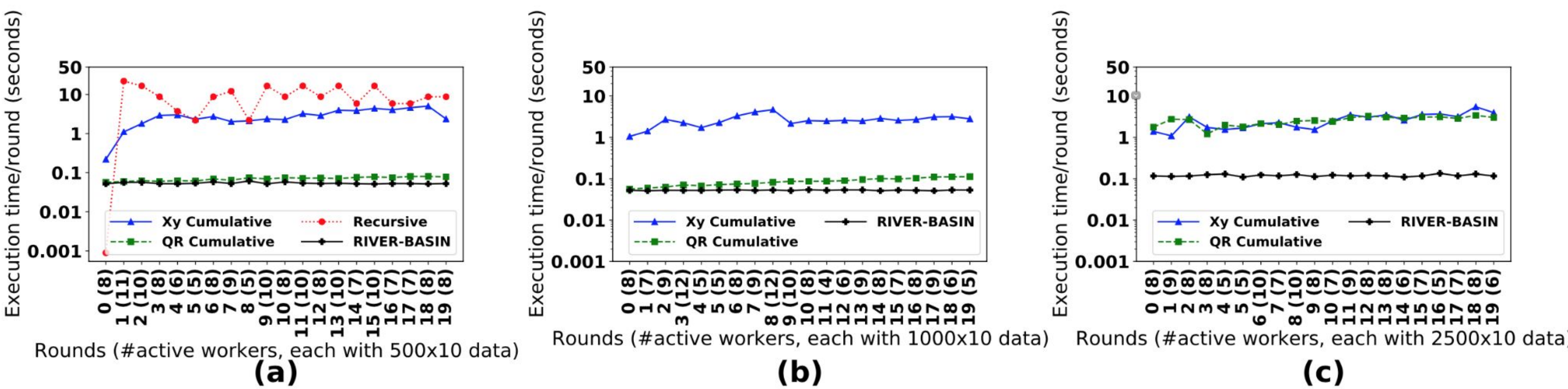
Results (1/5)





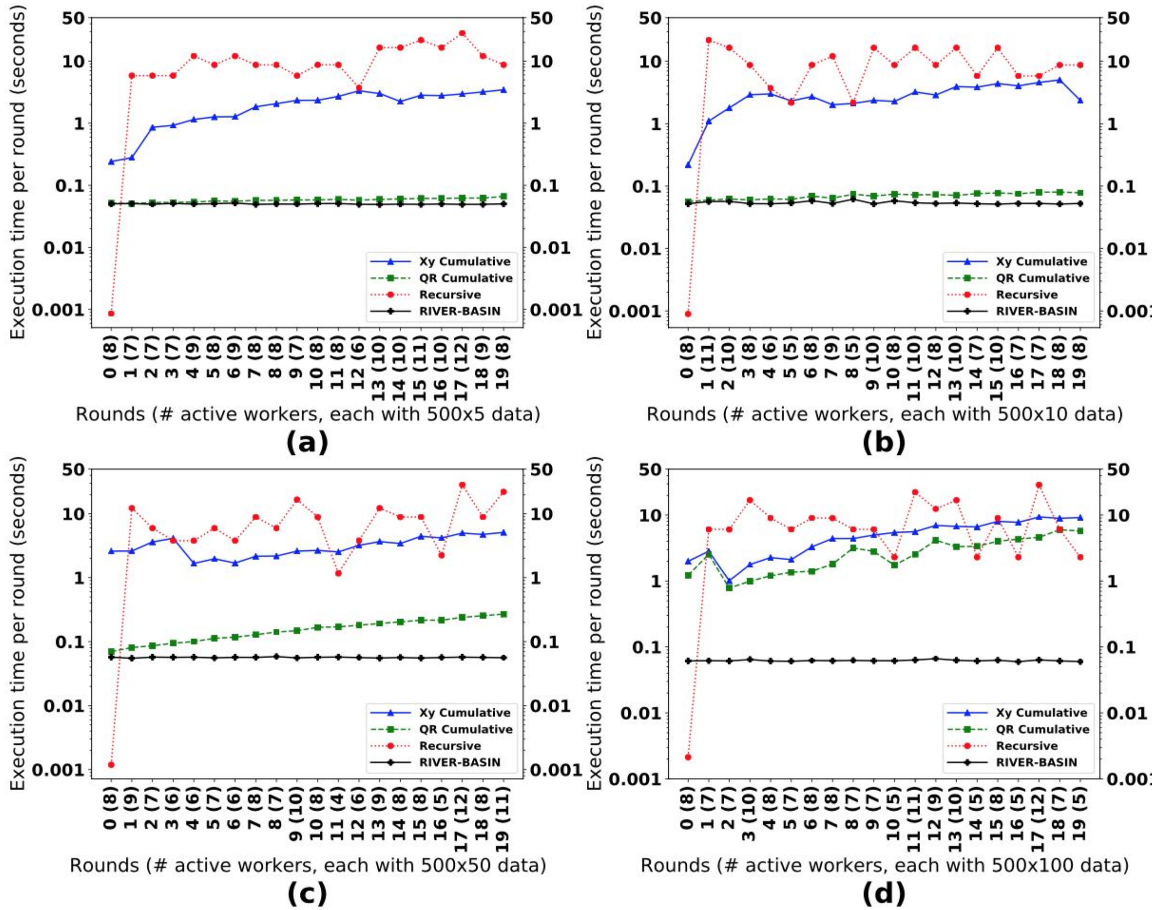
RIVER-BASIN **execution time per streaming round**
(addition of new data samples or workers)
is CONSTANT , i.e., computations depends ONLY on the current data and FASTER

Results (2/5)



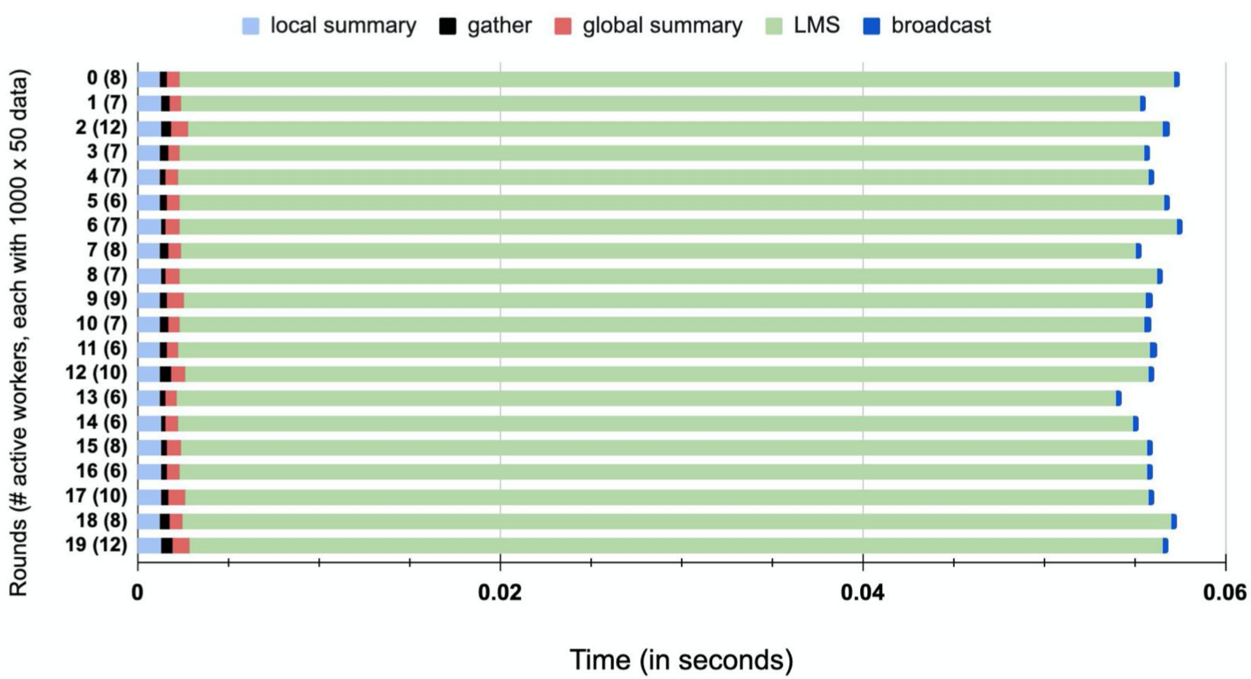
BASIN Scalability across varying streaming batch size, **n**
(a) **500**x10 (b) **1000**x10 (c) **2500**x10

Results (3/5)



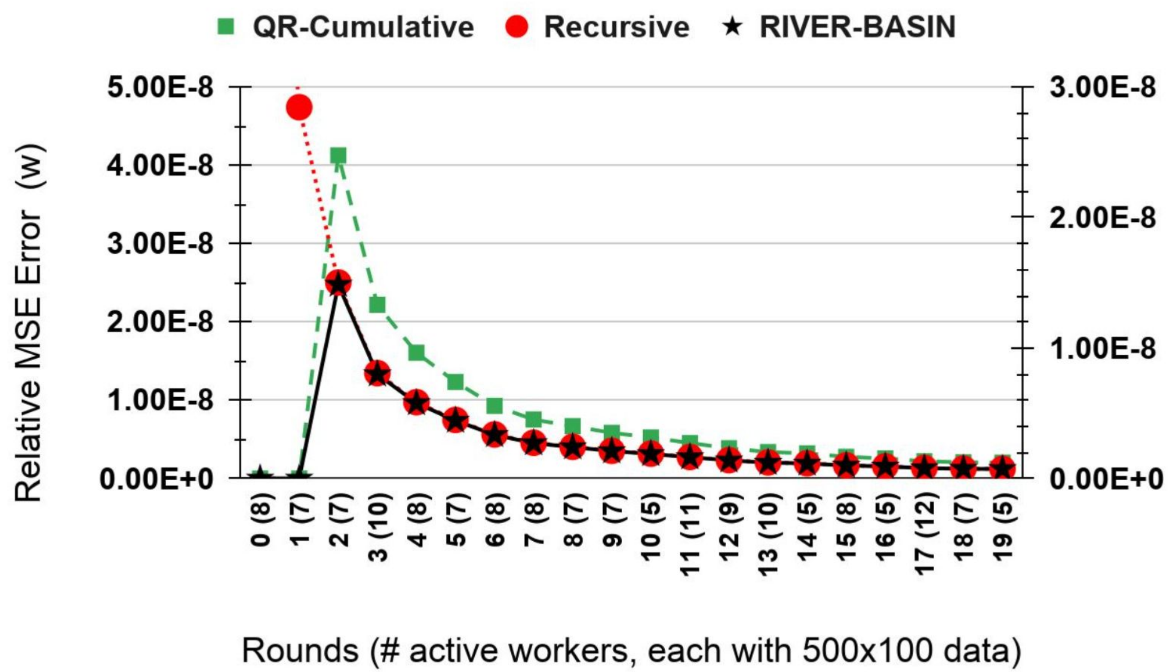
BASIN Scalability across varying feature dimension, **d**
(a) 500x**5** (b) 500x**10** (c) 500x**50** (d) 500x**100**

Results (4/5)



RIVER- BASIN: Timing breakdown analysis

Results (5/5)



BASIN model error relative to Xy-Cumulative

Model is accurately learnt in each round

Research Contributions

✔ Relaxed Synchronization for Parallel QP Problems

[IEEE IPDPS'16] K. Lee, R. Bhattacharya, **J. Dass**, V. N. S. P. Sakuru, and R. N. Mahapatra, "*A Relaxed Synchronization Approach for Solving Parallel Quadratic Programming Problems with Guaranteed Convergence*"

✔ Householder Sketch for Machine Learning

[ICML'21] J. Dass, and R. N. Mahapatra, "*Householder Sketch for Accurate and Accelerated Least-Mean-Squares Solvers*"

✔ Memory-efficient Framework for Distributed ML

[IEEE ICDCS'17] J. Dass, V. N. S. P. Sakuru, V. Sarin and R. N. Mahapatra, "*Distributed QR Decomposition Framework for Training Support Vector Machines*"

✔ Communication-efficient Framework for Scalable ML

[IEEE TPDS'18] J. Dass, V. Sarin and R. N. Mahapatra "*Fast and Communication-Efficient Algorithm for Distributed Support Vector Machine Training*"

✔ Multiple FPGA-based System for Energy-efficient ML

[ACM FPGA'19 | IEEE TC'20] J.Dass, Y. Narawane, R. N. Mahapatra, and V. Sarin, "*Distributed Training of Support Vector Machine on a Multiple-FPGA System*"

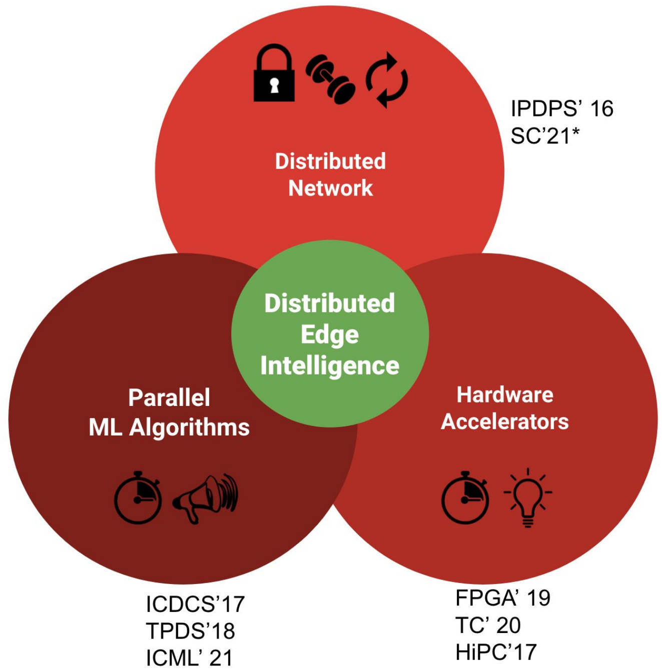
✔ Rapid Incremental Solver for Federated ML

[Under Review, ACM SC'21] J.Dass, N. Purwosumarto, R. N. Mahapatra, and X. Hu, "*Rapid Incremental Solver for Federated Regression*"

Conclusions

Distributed Edge Intelligence Requirements

01	Protect Data Privacy	
02	Reduce Latency	
03	Save Bandwidth	
04	Energy Efficiency	
05	Build Robust Models	
06	Streaming Data	



We proposed,

- **LSDA** for Relax Synchronization (IPDPS)
- **LMS-QR** for Householder Sketch (ICML)
- **QRSVM** for Memory-efficient Distributed Machine Learning (ICDCS)
- **QRSVM.v2** for Communication-efficient Scalable Machine Learning (TPDS)
- **Multi-FPGA** system for Energy-efficient Machine Learning (TC)
- **RIVER** for Incremental learning under Federated ML setups (SC, under Review)

Future Directions

- **Secure Multi-Party Decentralized Machine Learning**
  - Cryptographic methods
  - Differential Privacy
  - Safeguards against malicious workers
  - Privacy-preserving sketch computations

- **Codesigned AutoML Systems for Distributed Edge Intelligence**
  - automatically generate ML model-Accelerator codesigned pair
  - Hardware-aware Neural Architecture Search (HW-NAS)
  - Incorporate design parameters/constraints from distributed computing framework to generate optimal pairs

- **Systems for Lifelong Multi-Agent Learning**
  - Learning from little data, retain the acquired knowledge, share knowledge with other agents and apply to learning in new settings
  - Continuously update model: drone swarms, connected vehicles+users

ACKNOWLEDGMENTS

Dissertation Committee

- Dr. Rabi Mahapatra (advisor)
- Dr. Eun Jung Kim
- Dr. Xia Hu
- Dr. Raktim Bhattacharya

Collaborators

- Dr. Vivek Sarin, *TAMU*
- Dr. Kooktae Lee, *New Mexico Tech*
- Dr. Dharanidhar Dang, *UCSD*
- VNS Prithvi Sakuru, *Amazon*
- Yashwardhan Narawane, NVIDIA
- Rengang “Angelo” Yang, *TAMU*
- Nathan Purwosumarto, *TAMU*

CODES Lab Members

- Biplab Patra, *Microsoft*
- Akash Sahoo, *Samsung*
- Karl Ott, *PhD continuing*
- Dr. Ying Yung “Jerry” Yiu, *Intel*
- and everyone

Special Thanks

- Dr. Aakash Tyagi, *TAMU*
- Dr. Joseph Hurley, *TAMU (retired)*
- To all my students, PTs, and graders

Family, Friends, and Teachers

References

1. **J. Dass**, N. Purwosumarto, R. N. Mahapatra and X. Hu, RIVER: Rapid Incremental Solver for Federated Regression, in ACM/IEEE International Conference for High Performance Computing, Networking, Storage, and Analysis (SC 2021), Under Review

2. **J. Dass**, R. N. Mahapatra, Householder Sketch for Accurate and Accelerated Least-Mean-Squares Solvers, in International Conference on Machine Learning (ICML 2021)

3. **J. Dass**, Y Narawane, R. N. Mahapatra and V. Sarin, Distributed Training of Support Vector Machine on a Multiple- FPGA System, in IEEE Transactions on Computers (TC 2020),

4. **J. Dass**, Y Narawane, R. N. Mahapatra and V. Sarin, FPGA-based Distributed Edge Training of SVM, in ACM/SIGDA 27th International Symposium on Field Programmable Gate Arrays (FPGA 2019), Seaside, CA.

5. **J. Dass**, V. Sarin and R. N. Mahapatra, Fast and Communication-Efficient Algorithm for Distributed Support Vector Machine Training, in IEEE Transactions on Parallel and Distributed Systems (TPDS 2018)

6. D. Dang, **J. Dass** and R. Mahapatra, ConvLight: A Convolutional Accelerator with Memristor Integrated Photonic Computing, in IEEE 24th International Conference on High Performance Computing (HiPC 2017), Jaipur,

7. **J. Dass**, V. N. S. P. Sakuru, V. Sarin and R. N. Mahapatra, Distributed QR Decomposition Framework for Training Support Vector Machines, in IEEE 37th International Conference on Distributed Computing Systems (ICDCS 2017), Atlanta, GA,

8. K. Lee, R. Bhattacharya, **J. Dass**, V. N. S. P. Sakuru and R. N. Mahapatra, A Relaxed Synchronization Approach for Solving Parallel Quadratic Programming Problems with Guaranteed Convergence, in IEEE International Parallel and Distributed Processing Symposium (IPDPS 2016), Chicago, IL

Thank You !

TEXAS A&M UNIVERSITY

APPENDIX

Requirements for Edge Intelligence

01	Keep Data Local	• Keep data decentralized and local on devices • Design privacy-preserving ML algorithms
02	Reduce Latency	• Efficient and scalable training algorithms • Cheap inference calculations to enable real-time analytics
03	Communicate Less	• Communicate less data during training • Reduce synchronizations and idling during training
04	Energy Efficiency	• Efficient computation and communication process • Build energy-efficient hardware accelerators for Green AI
05	Build Robust Models	• Devise fault-tolerance for device failures or stragglers • Accurate and robust model to data perturbations
06	Streaming Data	• Incremental federated learning to update the global model • Discard data after each update for memory and privacy

Use Case

HOUSTON'S SMART CITY VISION

Transportation

Scroll right to see more

find
eniently

Intelligent Transportation System

A powerhouse of smart devices and real-time data helps manage Houston traffic on a comprehensive scale.

More on this project

Dockless Bikes & Scooters

More mobility options means less traffic congestion, convenient travel, better air quality and more.

More on this project

Parking Pay-By-Plate

A modern take on parking meters promises more seamless parking, easier enforcement and cost savings for the city.

More on this project

Parking Analytics

We're managing our city's parking inventory in real time to boost parking satisfaction while driving big efficiencies.

More on this project

Rice/Kinder Foundation Study

Research study addressing our environmentally-friendly vehicle replacement to modern electric vehicles.

More on this project

Autonomous Transit Circulator

Autonomous circulator system fast-tracks travel time, helps with traffic congestion and improves public transit accessibility.

More on this project

Public Safety

Scroll right to see more

Downtown Sandbox

This collaborative hub enables ongoing testing and installation of the latest tech to keep our city safe.

More on this project

Project Edison

This platform alerts authorities and the public during active emergencies, enabling safe passage — and safer communities.

More on this project

Firefighting Drones

Firefighting in Houston is safer than ever with smart drones that assess conditions and identify hazards.

More on this project

Resiliency & Sustainability

Scroll right to see more

Flood Detection Sensors

A low-cost, high-impact solution from Houston's brightest minds uses sensors to alert drivers of high-water conditions.

More on this project

Sanitary Sewer Outflow Monitors

A powerful network of monitors is keeping our citywide sewers in check and our city streets cleaner.

More on this project

Mobile Air Quality Sensors

Sensor-enabled vehicles monitor air pollutants throughout our city to embrace green environment initiatives.

More on this project

Water Innovation Hub

The nation's first large-scale water technology demonstration hub is testing and showcasing innovative solutions to water and wastewater challenges.

More on this project

Bridge To Clean Air

Cutting-edge filtration systems and air monitors work together to purify polluted air protecting neighboring communities.

More on this project

Roadway Flood Warning System

High water sensors and advanced traffic signaling will keep our city safe and informed during flood threats.

More on this project

Lead

A pow
buildi
prote
issues

More

http://houstontx.gov/smartcity/

Gaps in Parallel ML Training

- Expensive **communication** (synchronization) overheads
- **Memory cost** is nearly **quadratic** with sample size
- **Computational cost** per iteration is **quadratic** with sample size
- Some parallel techniques are limited to small workload, thereby, can **not scale**

For efficient parallel ML training

- Reduce the communication (synchronization) overhead via asynchronous or relaxed synchronous techniques
- Incorporate memory-efficient representation of data by inducing sparsity
- Design highly separable and independent sub-problems amenable for parallelization
- Build algorithms than can scale linearly with number of parallel machines

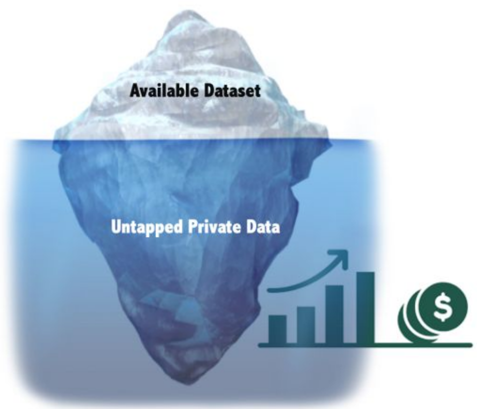
Opportunities

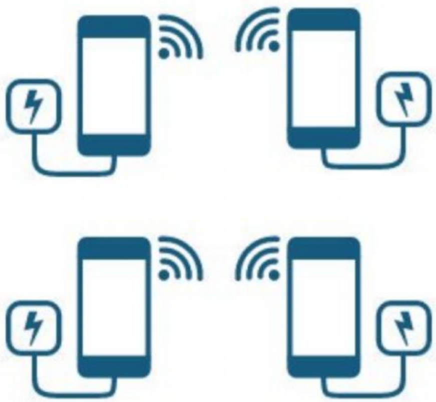
Untapped Private Data

- Available public data with big companies is just a tip of iceberg
- Utilize untapped private data in individual devices for ML with privacy protected

Idle Processing Power

- Billions of connected devices with underutilized processing capacity
- Leverage their idle processing power to build Decentralized processor for ML !

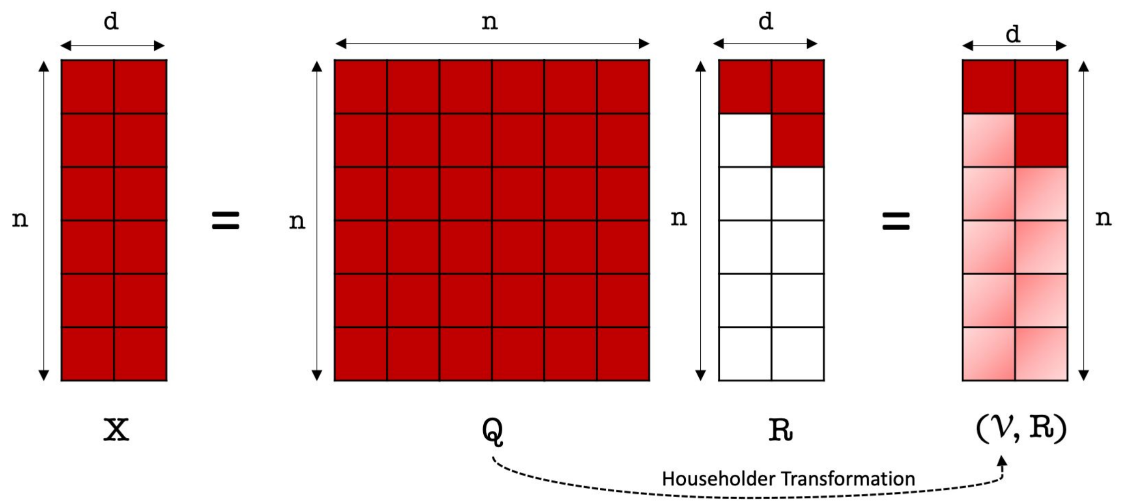




Householder-QR

Theorem 3.1 (Householder-QR [57]). *Let matrix* $\mathbf{X} \in \mathbb{R}^{n\times d}$ *with* $n > d$. *Householder QR decomposition of* $\mathbf{X}$ *generates set of* d *Householder matrices* $\mathcal{H}$ *and an* $n \times d$ *upper trapezoidal matrix* $\mathbf{R}$. *The Householder matrices are stored as a set of* d *Householder reflectors* $\mathcal{V}$. *Total memory footprint of above factors is* nd *elements with time complexity of* $O(nd^2)$ *for* $n \gg d$.

$$X = QR\ ,\ \text{where},\quad Q^TQ = QQ^T = I$$



memory consumption: O(nd) | O(n²) | O(d²) | O(nd)

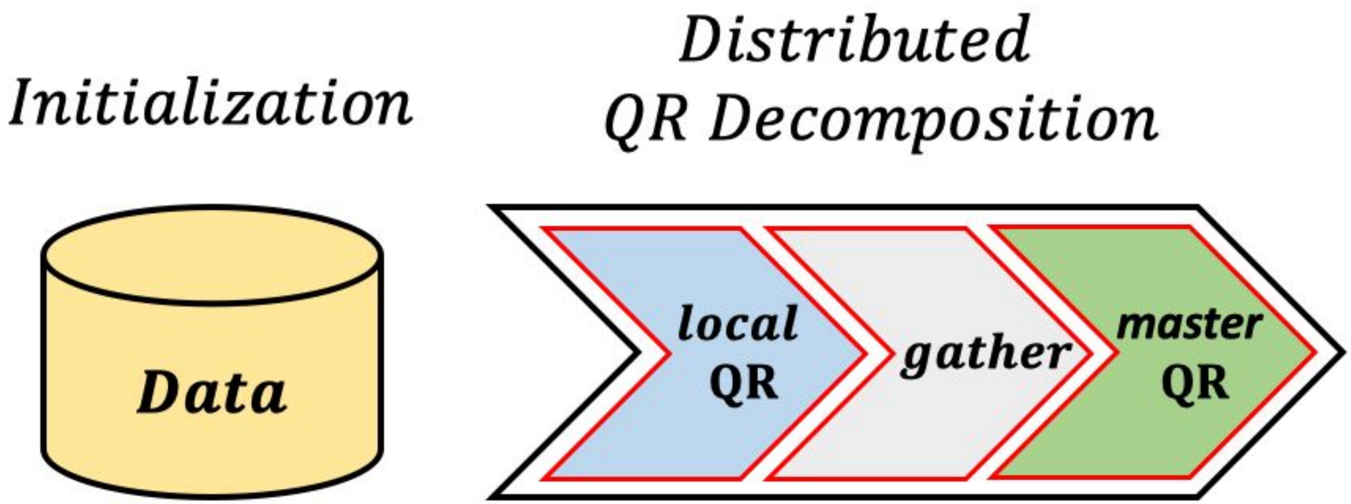
Distributed Householder Sketches

$$\mathbf{X} = \begin{pmatrix} \mathbf{X}_1 \\ \mathbf{X}_2 \\ . \\ . \\ \mathbf{X}_p \end{pmatrix} = \begin{pmatrix} \mathbf{Q}_1\mathbf{R}_1 \\ \mathbf{Q}_2\mathbf{R}_2 \\ . \\ . \\ \mathbf{Q}_p\mathbf{R}_p \end{pmatrix} = \texttt{diag}(\mathbf{Q}_1, \ldots, \mathbf{Q}_p) \begin{pmatrix} \mathbf{R}_1 \\ \mathbf{R}_2 \\ . \\ . \\ \mathbf{R}_p \end{pmatrix} , \qquad \mathbf{R}_{stack} = \begin{pmatrix} \mathbf{R}_1 \\ \mathbf{R}_2 \\ . \\ . \\ \mathbf{R}_p \end{pmatrix} = \mathbf{Q}_M \mathbf{R}_M$$

$$\mathbf{X} = \texttt{diag}(\mathbf{Q}_1, \ldots, \mathbf{Q}_p)\mathbf{R}_{stack} = \underbrace{\texttt{diag}(\mathbf{Q}_1, \ldots, \mathbf{Q}_p)\mathbf{Q}_M}_{\mathbf{Q}}\mathbf{R}_M$$

$$\mathbf{Q}^T\mathbf{c} = \mathbf{Q}_M^T \; \texttt{diag}(\mathbf{Q}_1^T, \ldots, \mathbf{Q}_p^T)\mathbf{c} = \mathbf{Q}_M^T \begin{pmatrix} \mathbf{Q}_1^T\mathbf{c}_1 \\ \mathbf{Q}_2^T\mathbf{c}_2 \\ . \\ . \\ \mathbf{Q}_p^T\mathbf{c}_p \end{pmatrix}$$

Workflow



Initialization
Distributed
QR Decomposition
Data
local
QR
gather
master
QR

Properties

Lemma

(Stability) The dual variable for LSDA algorithm is stable if and only if

$$\rho\Big(\mathbf{A}(\mathbf{P})\Big) < 1. \tag{8}$$

where $\mathbf{A}(\mathbf{P}) := I - P\sum_{i=1}^{N} \alpha_i \left(A_i Q_i^{-1} A_i^T\right)$ *and the symbol* $\rho(\cdot)$ *denotes the spectral radius of the given matrix (i.e., the largest magnitude of the eigenvalue).*

Proposition

(Convergence) Consider the QP problem that is separable. If the condition (8) *holds, then the dual variables* y_{LSDA} *for LSDA and* y_{TSDA} *for TSDA converge to the same fixed-point value* $y^* := \lim_{k\to\infty} y_{TSDA}^{k} = \lim_{t\to\infty} y_{LSDA}^{tP}$.

Theorem

(Optimality) For the given parallel QP problem with LSDA technique, the optimal synchronization period $P^\star$ *is obtained by*

$$P^\star = \max \operatorname*{argmin}_{P\in\mathbb{N}} \max\{|1-\underline{\lambda}(\beta)P|, |1-\bar{\lambda}(\beta)P|\} \tag{9}$$

where $\beta := \sum_{i=1}^{N} \alpha_i A_i Q_i^{-1} A_i^T$, $\underline{\lambda}(\cdot)$ *and* $\bar{\lambda}(\cdot)$ *denote the smallest and the largest eigenvalues of the square matrix, respectively.*

Memory-efficient Distributed ML

We propose,

1. QRSVM: QR decomposition framework for **memory-efficient** modeling and training of SVM
2. Optimal step size calculation for **fast convergence**
3. Distributed QRSVM for decomposing SVM into parallel equivalent sub-problems that are trained **in parallel**

Lagrangian $\mathcal{L}$ of QRSVM

$$\mathcal{L}(\hat{\boldsymbol{\alpha}}, \boldsymbol{\beta}) = \frac{1}{2}\hat{\boldsymbol{\alpha}}^T\left(\mathbf{R}\mathbf{R}^T + \frac{1}{2C}\mathbf{I}_n\right)\hat{\boldsymbol{\alpha}} + (\hat{\mathbf{e}})^T\hat{\boldsymbol{\alpha}} + \boldsymbol{\beta}^T(-\mathbf{Q}\hat{\boldsymbol{\alpha}})$$

where, $\boldsymbol{\beta} \geq \mathbf{0}_n$ is the Lagrangian dual variable.

Dual Decomposition via projected sub-gradient

Dual function: $g(\boldsymbol{\beta}) = \min_{\hat{\boldsymbol{\alpha}}} \mathcal{L}(\hat{\boldsymbol{\alpha}}, \boldsymbol{\beta})$
Dual Problem: $\max_{\boldsymbol{\beta}} g(\boldsymbol{\beta})$

Results (2/2)

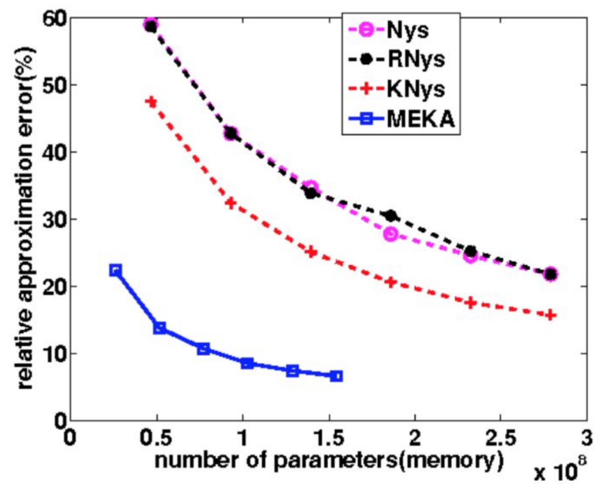


Figure 4.7: Low-rank Gaussian kernel approximation results using MEKA and other methods. [5]

Parameters	a9a	covtype
rank, k	40	64
C	2^{-1}	2^{-1}
γ	2^{-3}	2^{3}
approx. K_{error}	0.51	0.58
#cores, p	16	16
stopping threshold	10^{-3}	10^{-3}
optimal step size, η^*	1.9	1.9
#iterations, t	166	512

Table 4.2: Distributed-QRSVM: Parameter values. [2]

Distributed QRSVM: Timing Discussions for $p = 16$

Stage 1: Distributed QR

1. Computation: $T(p_{localQR}) + T(p_{masterQR})$
2. Communication: $T(c_{gatherR})$

Stage2: Parallel Dual Ascent

1. Computation: $T(p_{pda})$
2. Communication: $T(c_{pda})$ Gather+Scatter

Time details	a9a (in ms)	covtype (in s)
$T(p_{meka})$	460	2.1
$T(p_{localQR})$	24	1.89
$T(p_{masterQR})$	4	0.02
$T(c_{gatherR})$	0.5	0.04
$T(p_{pda})$	1628.1	120.18
$T(c_{pda})$	17.1	0.36
$T(train)$	1674.2	122.50

Optimal step size

Lemma

For any $\eta > 0$, the optimal step size η^* can be computed using

$$\eta^* = P^*\eta, \quad P^* \in \mathbb{N}$$

where,

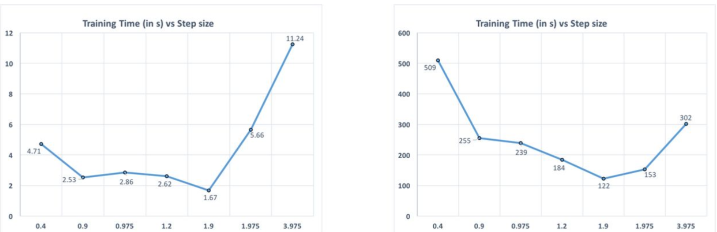
$$P^* = \begin{cases} 1 & \text{if} \quad 0 < \bar{\lambda}^{-1} < 2 \\ \lfloor \bar{\lambda}^{-1} \rceil & \text{if} \quad \bar{\lambda}^{-1} \geq 2 \end{cases}$$

and $\bar{\lambda} = (\lambda_{max}(\mathbf{M}) + \lambda_{min}(\mathbf{M}))/2$

$$\boldsymbol{M} := \eta\left(\boldsymbol{R}\boldsymbol{R}^T + \tfrac{1}{2C}\boldsymbol{I}_n\right)^{-1},$$

$\eta > 0$ is step size

$\lambda_{min}(\cdot)$ and $\lambda_{max}(\cdot)$ eigenvalues of matrix $\boldsymbol{M}$



(a) a9a

(b) covtype

Figure 4.9: Optimal step size: For both datasets, optimal step size is observed to be 1.9. [2]

In prior work,

Observe,

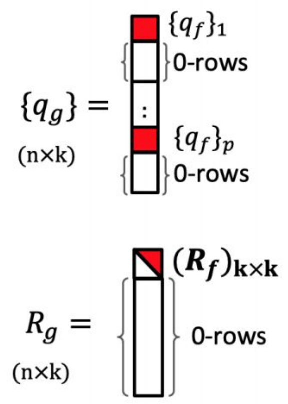


Figure: At Master core

$\beta = \boldsymbol{Q}\hat{\beta}$ using $\{\boldsymbol{q_g}\}$

- Gather $\hat{\beta}_i$ at Master core
- Scatter $(\boldsymbol{Q}_g\hat{\beta})_i$ to each core i

$\hat{\beta} = \boldsymbol{Q}^T\beta$ using $\{\boldsymbol{q_g}\}$

- Gather $(\boldsymbol{Q}_i^T\beta_i)$ at Master core
- Scatter $\hat{\beta}_i$ to each core i

Communication Overhead in distributed QRSVM

During every iteration of parallel dual ascent, each Gather and Scatter involves communicating **huge** $O(\frac{n}{p})$ data volume per core with Master core $\Rightarrow$ Poor Scalability

Results (3/3)

TABLE : Comparing *dis*-QRSVM with PSVM, P-packSVM and [12] on T_{train} (in seconds) for *covtype* dataset.

Algorithm	p=2	p=4	p=8	p=16	p=32	p=64
PSVM	8,562	4,396	2,352	1,270	635	341
P-packSVM	-	-	2,019	1,022	295	110
dis-QRSVM [12]	390	309	271	261	256	454
dis-QRSVM	**132**	**64**	**33**	**18**	**10**	**6**

- data unavailable

TABLE : Comparing proposed *dis*-QRSVM with [12] on Stage 2 computation time, T_{update} and communication time, T_{2g+2s} (in seconds) for *covtype* dataset.

Time	p=2	p=4	p=8	p=16	p=32	p=64
T_{update} [12]	379	304	269	259	252	448
T_{update}(proposed)	**122**	**59**	**30**	**16**	**8**	**5**
T_{2g+2s} [12]	1.63	1.81	0.34	0.05	1.19	3.02
T_{2g+2s}(proposed)	**0.02**	**0.02**	**0.14**	**0.05**	**0.03**	**0.11**

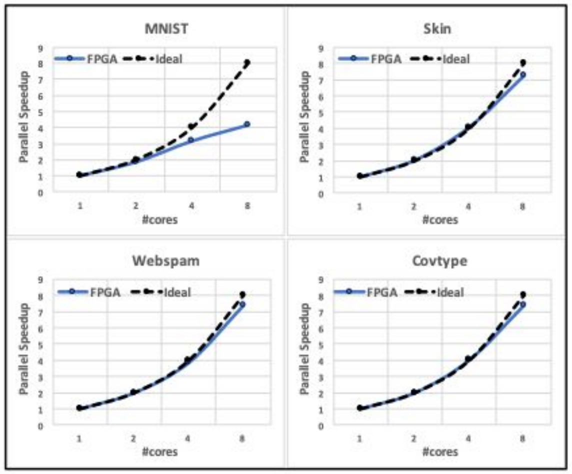
- dis-QRSVM trains upto 70×, upto 60×, and upto 75× faster than PSVM, P-packSVM, and earlier [ICDCS'17] implementation, respectively
- dis-QRSM is communication-efficient and scales better on PDA with $p = \{2, 4, 8, 16, 32, 64\}$ than [ICDCS'17]
- dis-QRSVM can handle larger workloads than [ICDCS'17]

Motivation

- Edge devices are getting efficient in processing data
- Make them capable for accelerating ML training
- Help reduce latency for critical applications
- Lower the dependency on HPC server grade CPUs, GPUs
- Provide energy-efficient modeling and training
- First-of-its-kind work for training SVM in a multiple FPGA environment

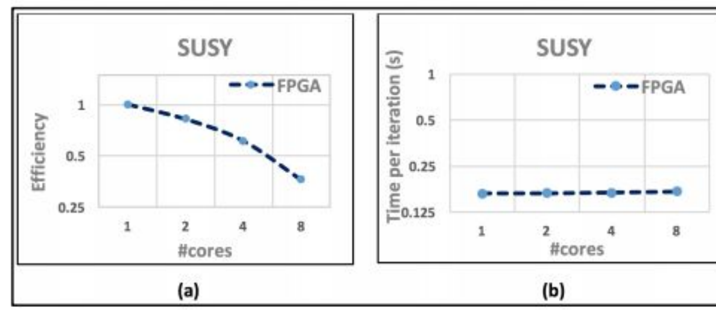
Results (1/3)

Strong Scaling Analysis



- Achieves near **linear parallel speedup** for larger datasets Skin, Webspam, Covtype
- For small dataset MNIST, going beyond *p = 4* seems to be overkill

Weak Scaling Analysis



- Workload per FPGA fixed at *250K* samples
- (a) T_{QR} is constant while T_{DA} ↑ with #iterations
- (b) (T_p^{FPGA}/t) is constant as desired

Results (2/3)

Energy Analysis

- Under strong scaling, the proposed FPGA design follows the **ideal energy consumption** trend that is **constant** across #FPGA units.
- Validates **fully parallel** implementation

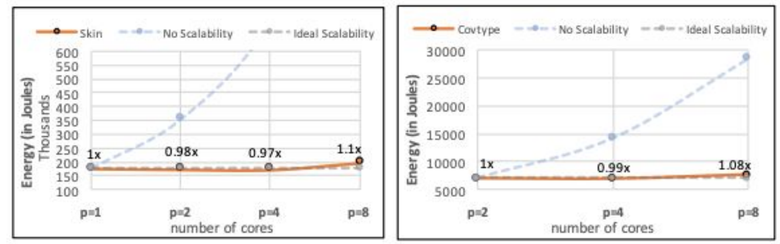


Fig. Energy consumption under Strong Scaling Skin and Covtype

- Under weak scaling, the ideal energy consumption trend is linear while no scalability trend is quadratic.
- The proposed design is closer to being linear than quadratic. Aberration at $p=8$ due to large #iterations for fine tuning model with increasing overall problem size
- ($Energy/p$) is nearly **constant** as expected with uniform workload per device

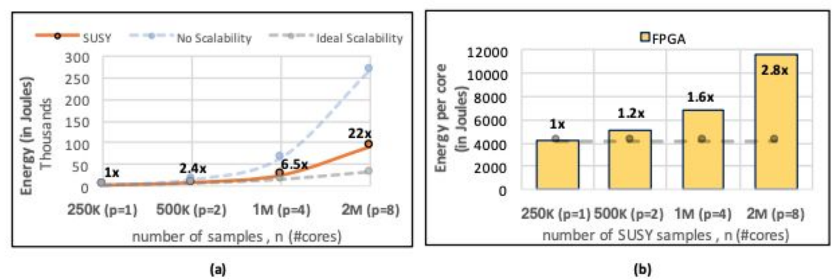
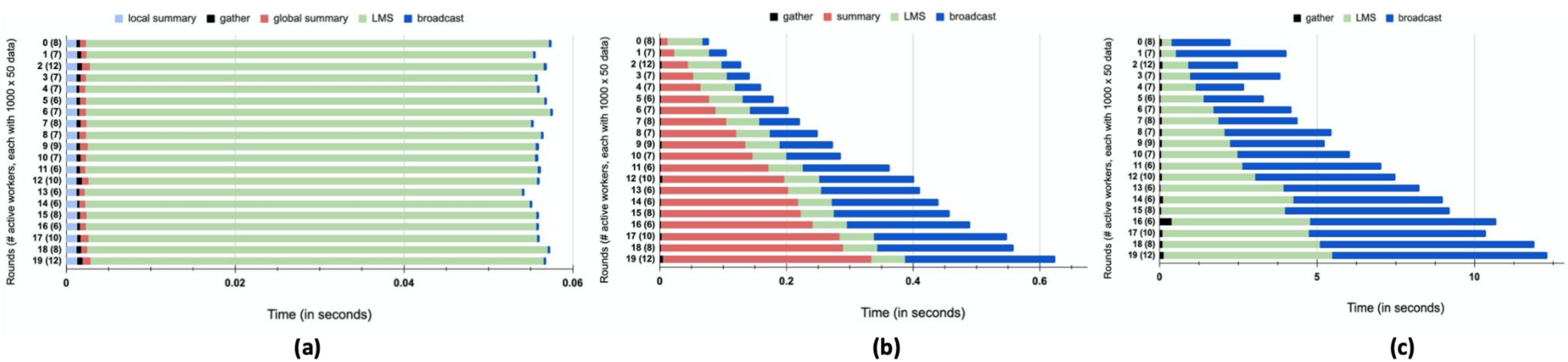
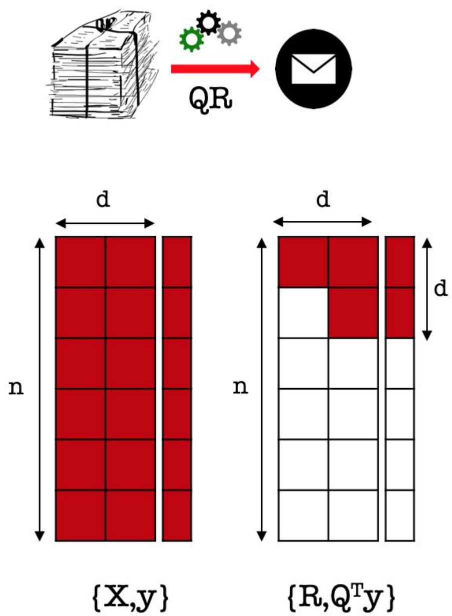


Fig. (a) Total Energy (b) Energy per core consumption under Weak Scaling for SUSY

Results (4/5)



BASIN: Timing breakdown analysis (a) RIVER (b) QR-Cumulative (c) Xy-Cumulative



memory consumption $O(nd)$ $O(d^2)$