ECLIPSE: An Extreme-Scale Linear Program Solver for Web-Applications




Kinjal Basu
LinkedIn AI

Amol Ghoting
LinkedIn AI

Rahul Mazumder
MIT

Yao Pan
LinkedIn AI

Agenda

Overview

Introduction

Large-Scale Linear Programs (LP) has several applications on web

Problems of Extreme Scale

$$\min_{x} \; c^T x \quad \text{s.t.} \quad Ax \leq b$$

- Billions to Trillions of Variables
- Ad-hoc Solutions
  - Splitting the problem to smaller sub-problem → No guarantee of optimality
- Exploit the Structure of the Problem
- Solve a Perturbation of the Primal Problem.
  - Smooth Gradient
  - Efficient computation

Motivating Example

Friend or Connection Matching Problem

- Maximize Value
  - Total invites sent is greater than a threshold
  - Limit on invitations per member to prevent overwhelming members

$$
\begin{aligned}
\max_x \quad & \sum_{i,j} x_{ij} p^1_{ij} && \text{(Total Value)} \\
\text{s.t.} \quad & \sum_{i,j} x_{ij} p^2_{ij} \geq b_0 && \text{(Total Invite Constraint)} \\
& \sum_i x_{ij} p^2_{ij} \leq b_j, && j \in \{1, \ldots, J\}, \\
& \sum_j x_{ij} = 1, && i \in \{1, \ldots, I\}
\end{aligned}
$$

- p^1 - Value Model
- p^2 - Invitation Model
- x_{ij} - Probability of showing user j to user i

Scale:

- $I \approx 10^8$
- $J \approx 10^4$
- $n \approx 10^{12}$

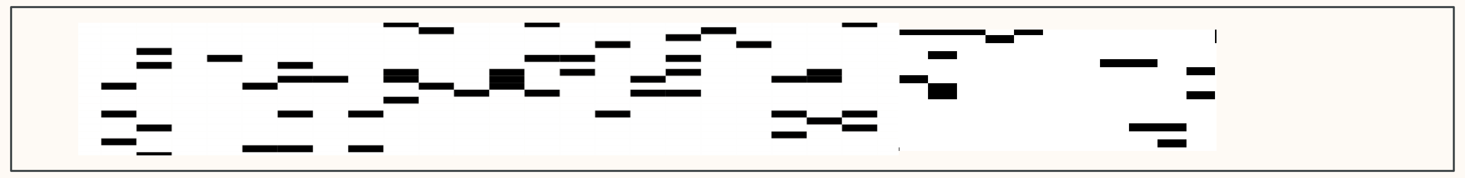
(1 Trillion Decision Variables)

General Framework

$$\min_{x} \quad c^T x$$
$$\text{s.t.} \quad Ax \le b$$
$$x_i \in \mathcal{C}_i, \quad i \in [I]$$

- Users i, Items j, and x_{ij} is the association between (i, j)
- $n = IJ$ can range in 100s of millions to 10s of trillions
- $\mathcal{C}_i$ are simple constraints (i.e. allows for efficient projections)

$$A$$



$$\begin{pmatrix} D_{11} & \cdots & D_{1I} \\ \vdots & \cdots & \vdots \\ D_{m_2 1} & \cdots & D_{m_2 I} \end{pmatrix}$$

$A^{(1)}$ ← Global Constraints
Cohort Level Constraints
Eg: Total Invite Constraint

$A^{(2)}$ ← Item level constraints
Eg: Limits on invitation per user

ECLIPSE: Extreme Scale LP Solver

Solving The Problem

Primal LP: $$P_0^* := \min_x \quad c^T x \qquad \text{s.t.} \qquad Ax \le b, \quad x_i \in \mathcal{C}_i, i \in [I]$$

Old idea: Perturbation of the LP (Mangasarian & Meyer '79; Nesterov '05; Osher et al '11...)

Primal QP: $$P_\gamma^* := \min_x \quad c^T x + \frac{\gamma}{2} x^T x \qquad \text{s.t.} \qquad Ax \le b, \quad x_i \in \mathcal{C}_i, i \in [I]$$

Dualize

Dual QP: $$g_\gamma(\lambda) := \min_{x \in \prod \mathcal{C}_i} \left\{ c^T x + \frac{\gamma}{2} x^T x + \lambda^T (Ax - b) \right\}$$

Key Observation: $\text{length}(\lambda)$ is small

Solve the Dual QP: $$g_\gamma^* := \max_{\lambda \ge 0} \; g_\gamma(\lambda) = P_\gamma^*$$

Strong duality

Solving The Problem

Primal: $$P_0^* := \min_x \quad c^T x \qquad \text{s.t.} \qquad Ax \le b, \quad x_i \in \mathcal{C}_i, i \in [I]$$

$$x_\gamma^* \in \operatorname*{argmin}_x \; c^T x + \frac{\gamma}{2} x^T x \qquad \text{s.t.} \qquad Ax \le b, \quad x_i \in \mathcal{C}_i, i \in [I]$$

- Observation-1: Exact Regularization (Mangasarian & Meyer '79; Friedlander Tseng '08)

  $\exists \bar{\gamma} > 0$ such that x_γ^* solves LP for all $\gamma \le \bar{\gamma}$

Dual: $$g_\gamma(\lambda) := \min_{x \in \prod \mathcal{C}_i} \left\{ c^T x + \frac{\gamma}{2} x^T x + \lambda^T (Ax - b) \right\}$$

$$g_\gamma^* := \max_{\lambda \ge 0} \; g_\gamma(\lambda)$$

- Observation-2: Error Bound (Nesterov '05)

  $|g_\gamma^* - P_0^*| = O(\gamma)$

Solving The Problem

$$\max_{\lambda \geq 0} \; g_\gamma(\lambda)$$

- Observation-1: Dual objective is smooth (implicitly defined)
  [Nesterov '05]

$$\lambda \mapsto g_\gamma(\lambda) \text{ is } O(1/\gamma)\text{-smooth}$$

- Observation-2: Gradient expression (Danskin's Theorem)

$$\nabla g_\gamma(\lambda) = \underbrace{A\hat{x}(\lambda)} - b$$

$$\hat{x}(\lambda) \in \operatorname*{argmin}_{x \in \prod \mathcal{C}_i} \left\{ c^T x + \frac{\gamma}{2} x^T x + \lambda^T (Ax - b) \right\}$$

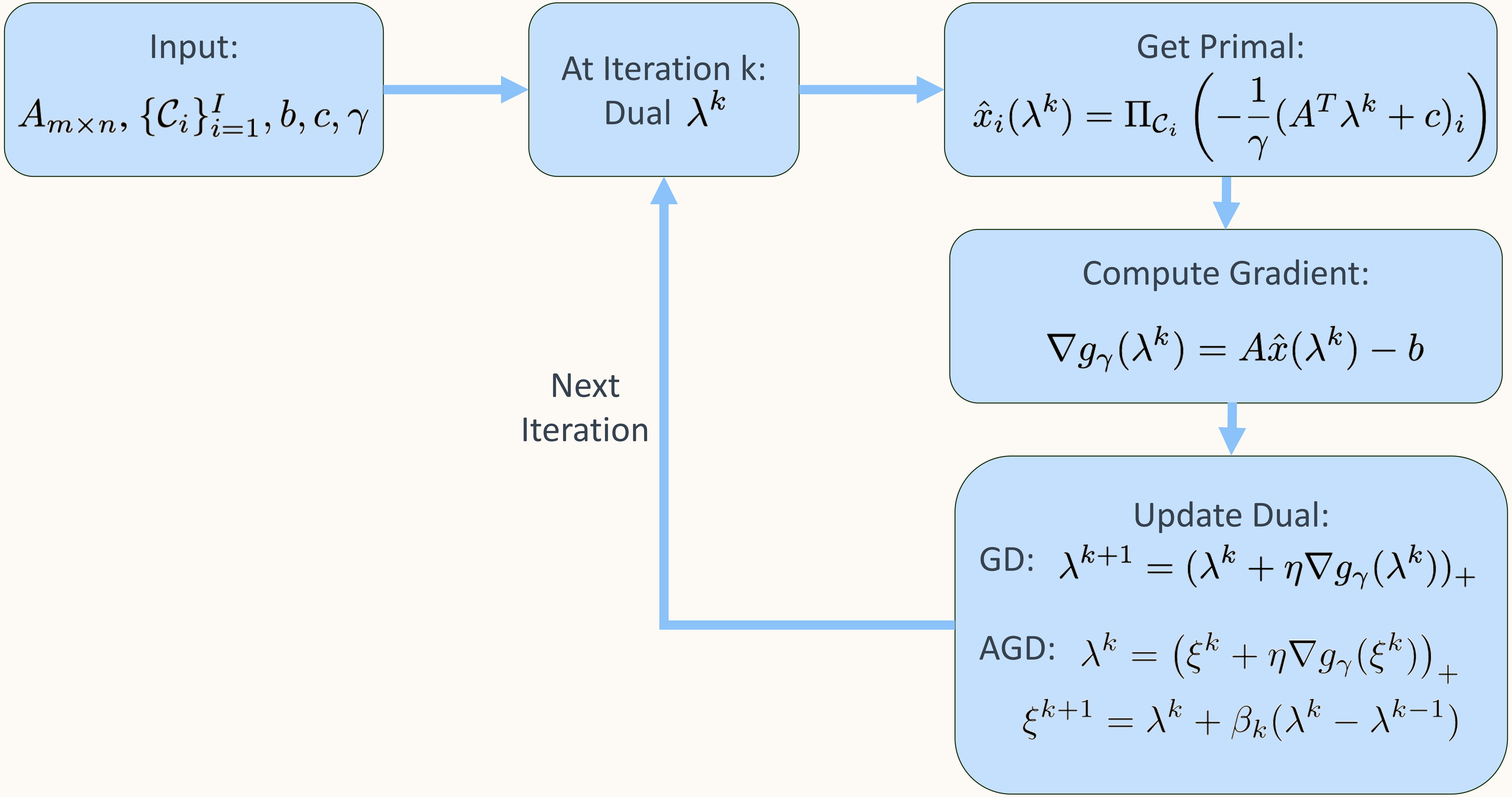
$$\hat{x}_i(\lambda) = \Pi_{\mathcal{C}_i}\left(-\frac{1}{\gamma} (\underbrace{A^T \lambda} + c)_i \right)$$

- Key bottleneck: Matrix-vector multiplication
- Simple projection operation

ECLIPSE Algorithm

- Proximal Gradient Based methods (Acceleration, Restarts)
- Optimal convergence rates.

Overall Algorithm



Applications

Volume Optimization

Maximize Sessions

- Total number of emails / notifications bounded
- Clicks above a threshold
- Disablement below a threshold

Generalized from global to cohort level systems and member level systems

$$
\begin{aligned}
\max_{x} \quad & x^T p^1 && \text{(Total Sessions)} \\
\text{s.t.} \quad & x^T 1 \leq c_1 && \text{(Sends are Bounded)} \\
& x^T p^2 \geq c_2 && \text{(Clicks above a threshold)} \\
& x^T p^3 \leq c_3 && \text{(Disables below a threshold)} \\
& 0 \leq x \leq 1 && \text{(Probability Constraint)}
\end{aligned}
$$

Multi-Objective Optimization

- Maximize Metric 1
  - Metric 2 is greater than a minimum
  - Metric 3 is bounded
  - …

- Most Product Applications

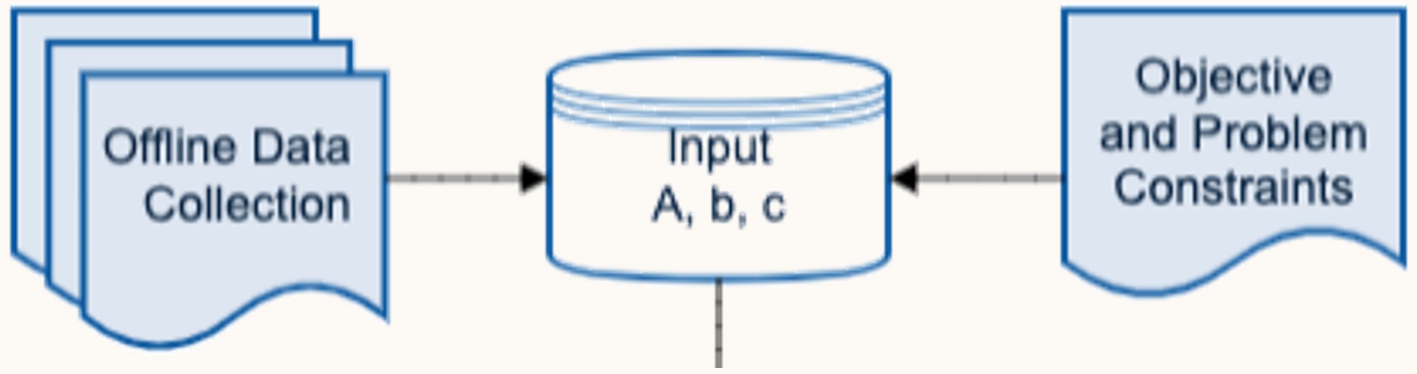
- Engagement vs Revenue
- Sessions vs Notification / Email Volume
- Member Value vs Annoyance

$$
\begin{aligned}
\max_{x} \quad & \sum_{i,j} x_{ij} p_{ij}^{1} & \text{(Metric 1)} \\
\text{s.t.} \quad & \sum_{i,j} x_{ij} p_{ij}^{2} \geq b_0 & \text{(Metric 2)} \\
& \sum_{i,j} x_{ij} p_{ij}^{3} \leq b_1 & \text{(Metric 3)} \\
& \vdots \\
& x_i \in \mathcal{C}_i, \;\; i \in [I]
\end{aligned}
$$

System Infrastructure

System Architecture

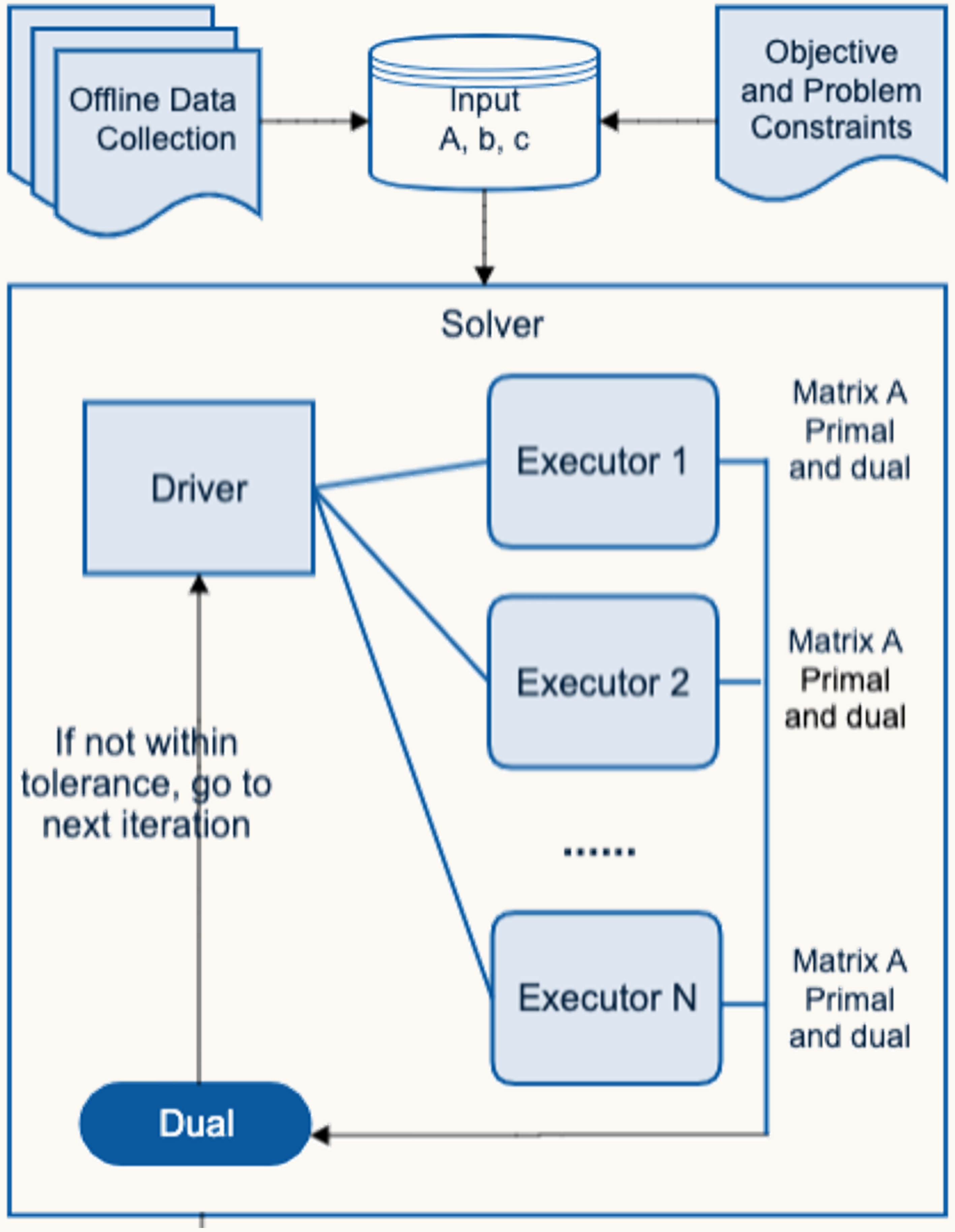
- Data is collected from different sources and restructured to form Input A, b, c



System Architecture

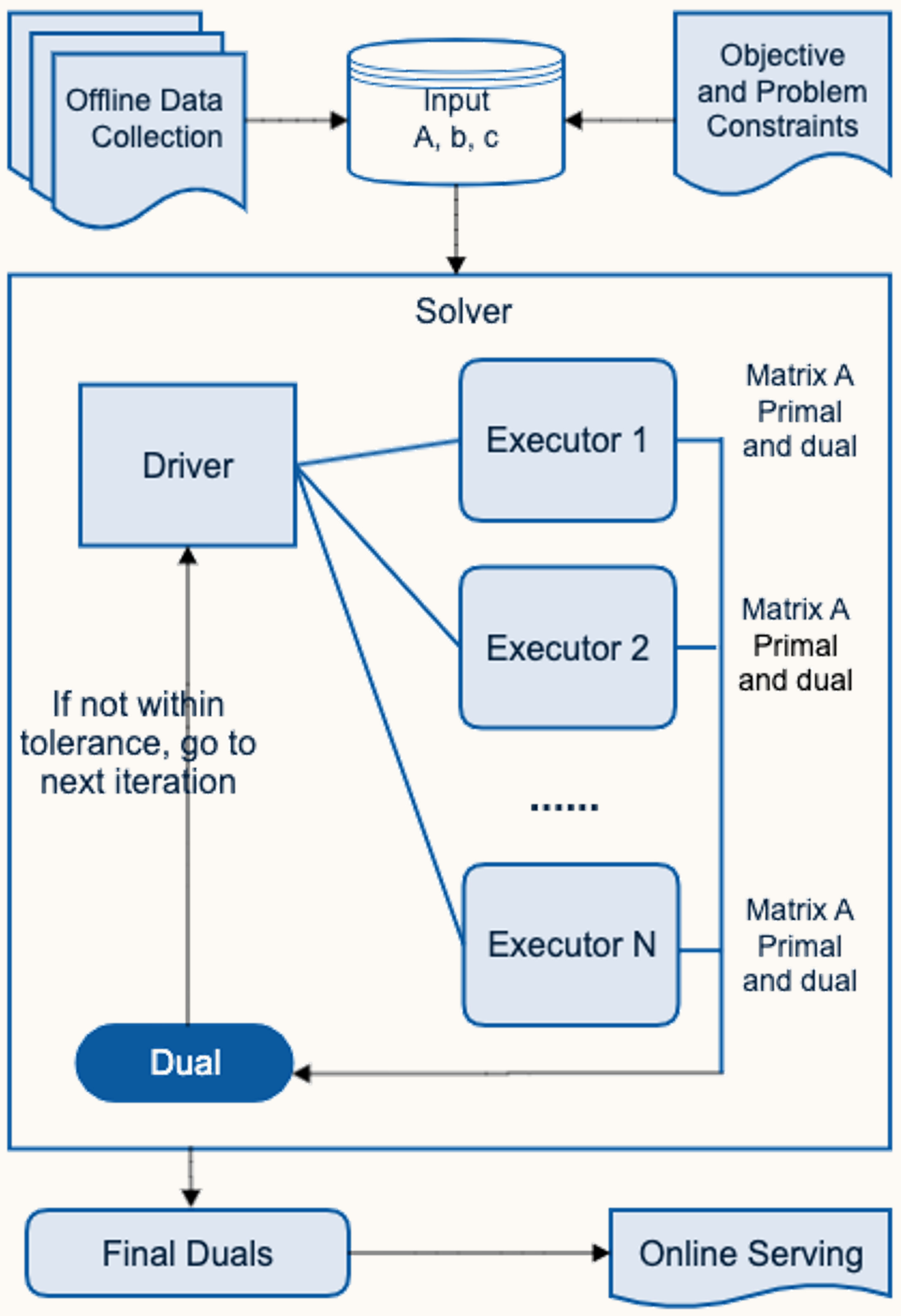
- Data is collected from different sources and restructured to form Input A, b, c

- The solver is called which runs the overall iterations.
  - The data is split into multiple executors and they perform matrix vector multiplications in parallel
  - The driver collects the dual and broadcasts it back to continue the iterations



System Architecture

- Data is collected from different sources and restructured to form Input A, b, c
- The solver is called which runs the overall iterations.
  - The data is split into multiple executors and they perform matrix vector multiplications in parallel
  - The driver collects the dual and broadcasts it back to continue the iterations
- On convergence the final duals are returned which are used in online serving



Detailed Spark Implementation

Data Representation

- Customized DistributedMatrix API
- $A^{(1)}$: BlockMatrix API from Apache MLLib
- $A^{(2)}$: Leverage Diagonal structure and implement DistributedVector API using RDD (index, Vector)

Estimating Primal

- Component wise Matrix Multiplications and Projections are done in parallel
- We cache A in executor and broadcast duals to minimize communication cost.
- The overall complexity to get the primal is $O(J)$

Estimating Gradient

- Most computationally expensive step to get $A\hat{x}(\lambda)$
- The worst-case complexity is $O(n = IJ)$

Experimental Results

Comparative Results

- We compare with a technique of splitting the problem (SOTA):

$$\min_x \; c_k^T x \quad \text{s.t.} \quad A_k x \le b_k, \quad x_i \in \mathcal{C}_i, i \in S_k.$$

$$A = [A_1 : \dots, A_K]$$

$$b = \sum_{k=1}^{K} b_k$$

$$c = (c_1, \dots, c_K)$$

$$\hat{\lambda} = \frac{1}{K} \sum_{k=1}^{K} \hat{\lambda}_k$$

n	Method	Objective	Primal Residual
10^6	ECLIPSE	$\mathbf{3.751 \times 10^5}$	$\mathbf{6.91 \times 10^{-4}}$
	Average 1	3.748×10^5	3.73×10^{-3}
	Average 2	3.747×10^5	1.03×10^{-2}
10^7	ECLIPSE	$\mathbf{3.750 \times 10^6}$	$\mathbf{7.12 \times 10^{-4}}$
	Average 1	3.747×10^6	1.71×10^{-3}
	Average 2	3.747×10^6	3.73×10^{-3}
10^8	ECLIPSE	$\mathbf{3.750 \times 10^7}$	$\mathbf{6.56 \times 10^{-4}}$
	Average 1	3.747×10^7	1.17×10^{-3}
	Average 2	3.747×10^7	1.73×10^{-3}

Table 1. Comparison of our algorithm with the averaging method. Average 1 and 2 correspond to a split size of 10^3 and 10^4 respectively.

Please see the full paper for other comparisons

Real Data Results

- Test on large-scale volume optimization and matching problems
- Spark 2.3 with up to 800 executors
- 1 Trillion use case converged within 12 hours

Problem	Scale n	Time(Hours) ECLIPSE	Time(Hours) SCS
Volume Optimization (9)	10^7	0.8	2.0
	10^8	1.3	>24
	10^9	4.0	>24
Matching Problem (10)	10^{10}	4.5	>24
	10^{11}	7.2	>24
	10^{12}	11.9	>24

Table 2. Running time for Extreme-Scale Problems on real data

SCS: O'Donoghue et al (2016)

Key Takeaways

Key Takeaways

- A framework for solving structured LP problems arising in several applications from internet industry
- Most multi-objective optimization can be framed through this.
- Given the computation resources, we can scale to extremely large problems.
- We can easily scale up to 1 Trillion variables on real data.

Thank you