# ZipLine: An Optimized Algorithm For Elastic Bulk Synchronous Parallel Model

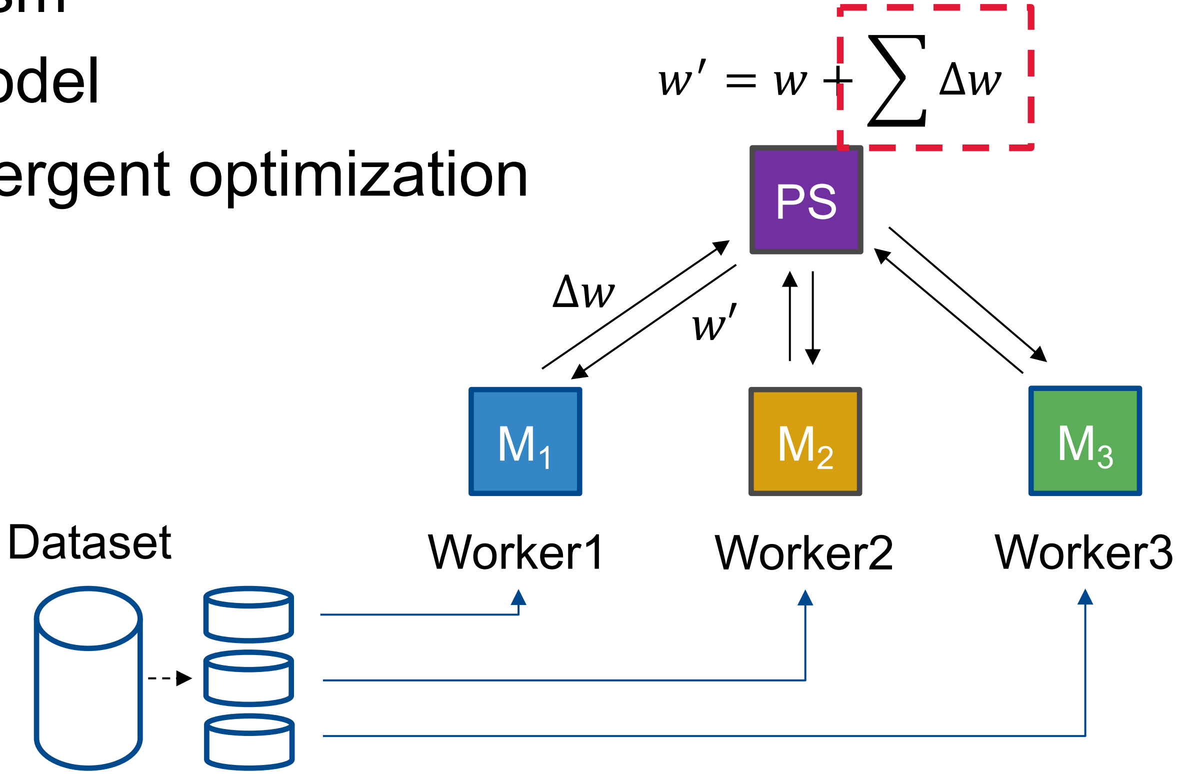
Xing Zhao, Manos Papagelis, Aijun An, Bao Xin Chen, Junfeng Liu and Yonggang Hu

October, 2021

# Background

# Parameter Server Framework

- A **centralized** distribution framework
- Data parallelism
- Train DNN model
- Iterative convergent optimization



# Problems for Distributed Deep Learning

- Synchronization among worker costs (waiting time)
  - Faster worker waits for slower worker

- Learning performance (convergence speed and final accuracy)
  - Reducing synchronizations
    - Decrease waiting time
    - Increase iteration throughput --- usually leads to fast convergence
  - Reducing synchronizations produces staled gradients to the DNN model updates
    - Fast convergence but low accuracy
    - Hinder the DNN model from learning

- How to balance between fast convergence and high accuracy?

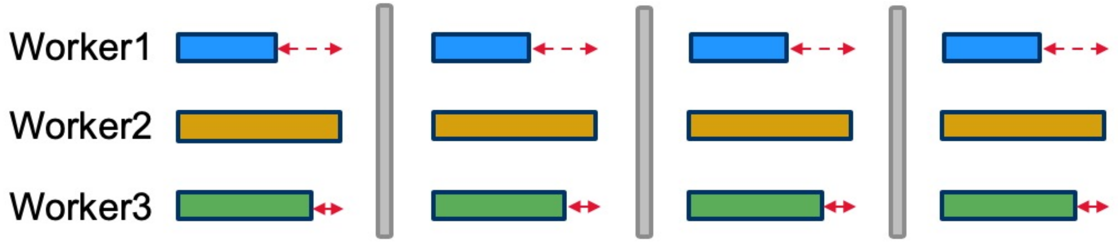
# Priors: BSP, SSP and ASP

# Bulk Synchronous Parallel (BSP)

- Distributed Deep Neural Networks Training
  - Iterative convergent optimization
    - Stochastic Gradient Descent (SGD)

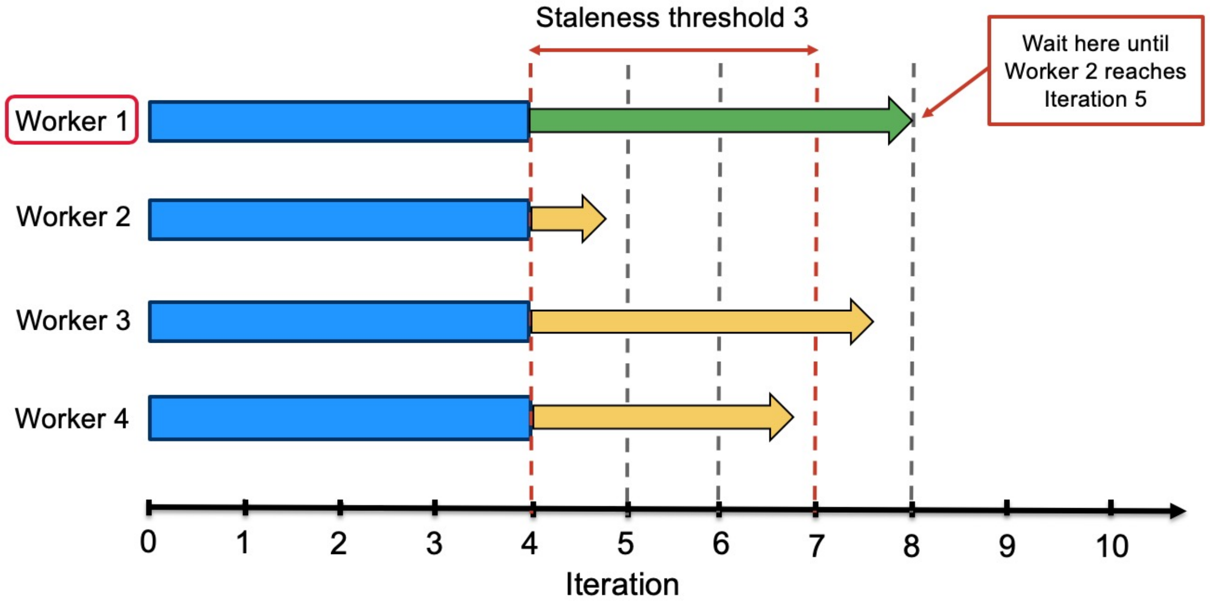
- BSP guarantees convergence by using SGD
  - Parameters synchronization is mandatary on every iteration
    - Good for homogeneous environment
    - Bad for heterogeneous environment

**Straggler** Problem: fast worker waiting for the slowest worker



# Stale Synchronous Parallel (SSP)

- **Blind** to each machine's ***processing capacity***
- Staleness threshold ***s*** is a **fixed** *hyper-parameter* requiring **fine-tuning** to find the *optimal* value of ***s***
- Fine-tuning is required **again** to find *optimal* ***s*** when **changes** happen to
  - Other hyper-parameters (e.g., learning rate, decay)
  - DNN Model
  - Running environment

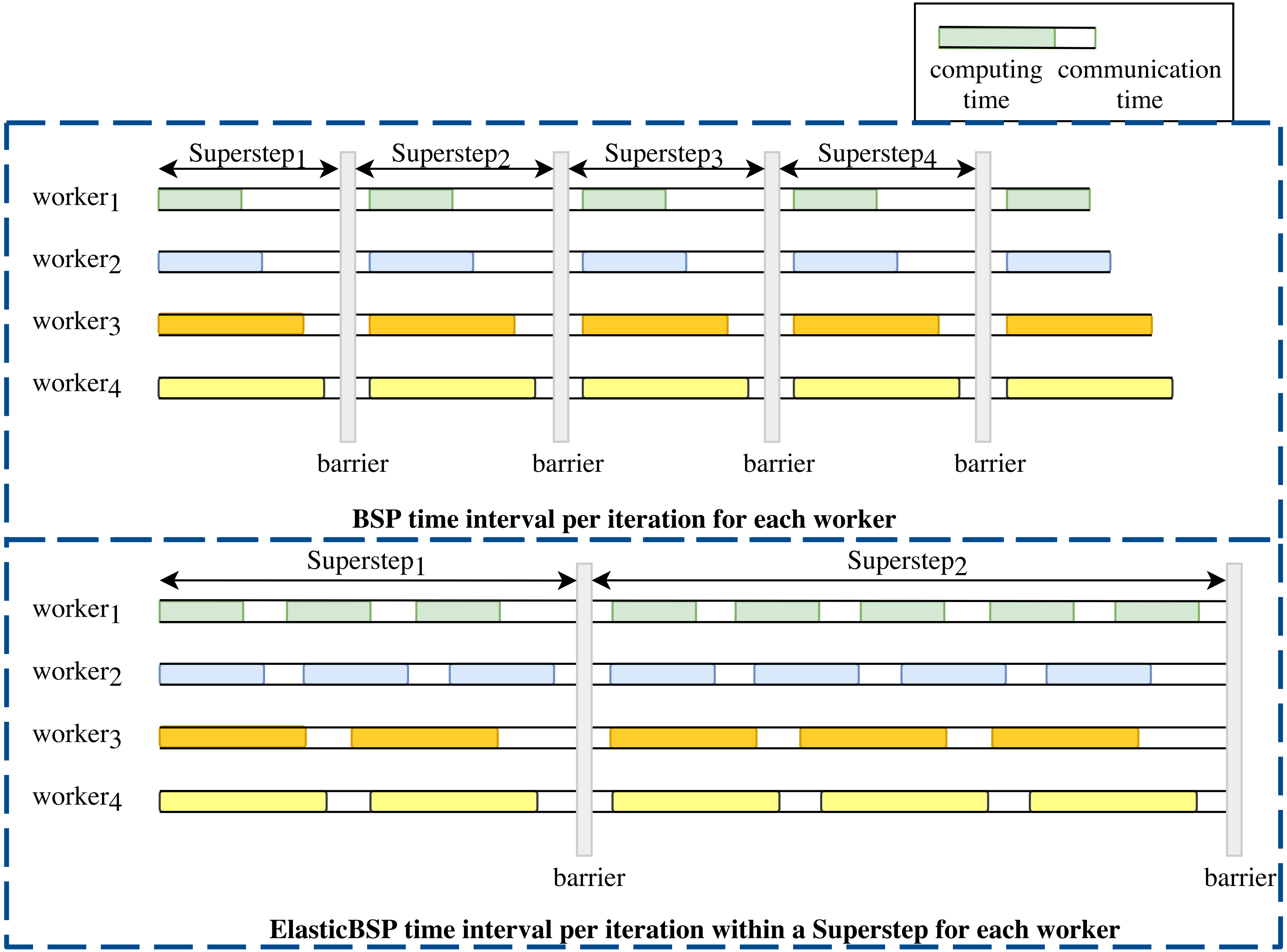


# Asynchronous Parallel (ASP)

- Each worker computing gradients independently
- No synchronizations among workers
  - Maximum iteration throughput
  - The largest number of stale gradients updates to the DNN models
  - Fast convergence but usually end up getting low accuracy

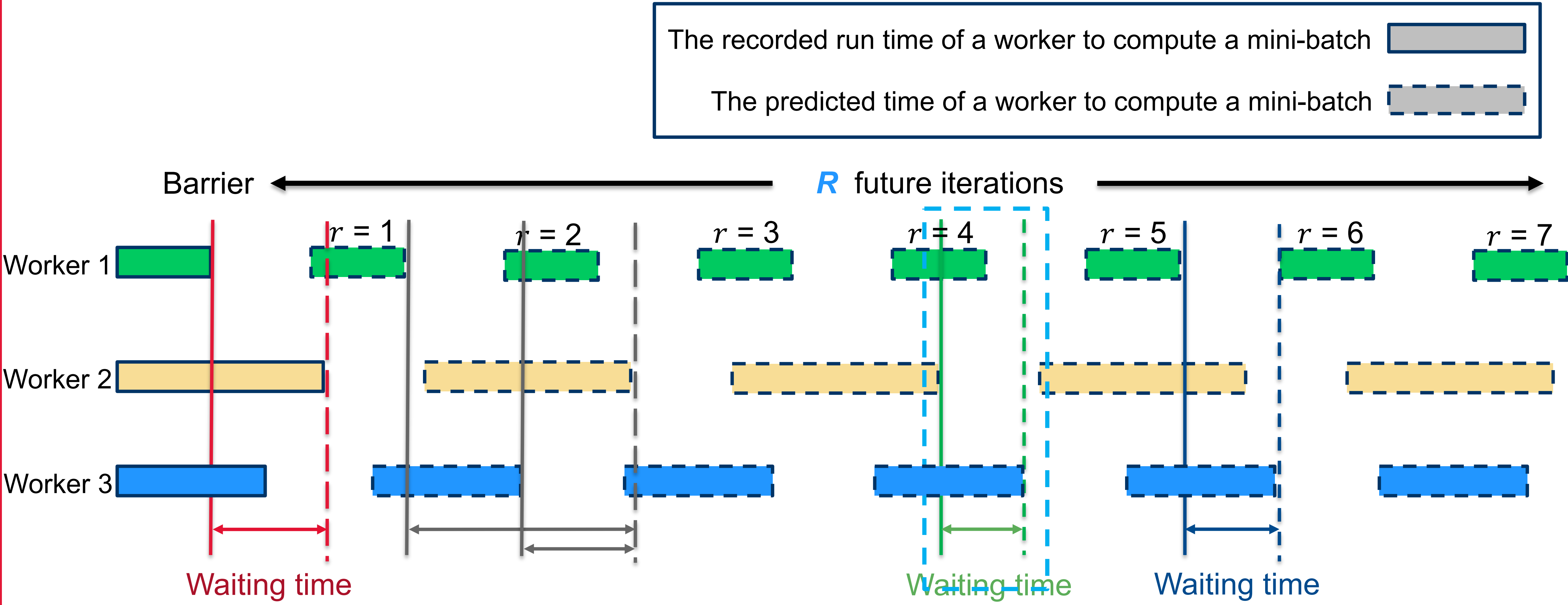
# Our Method

# Objective: Elastic Bulk Synchronous Parallel



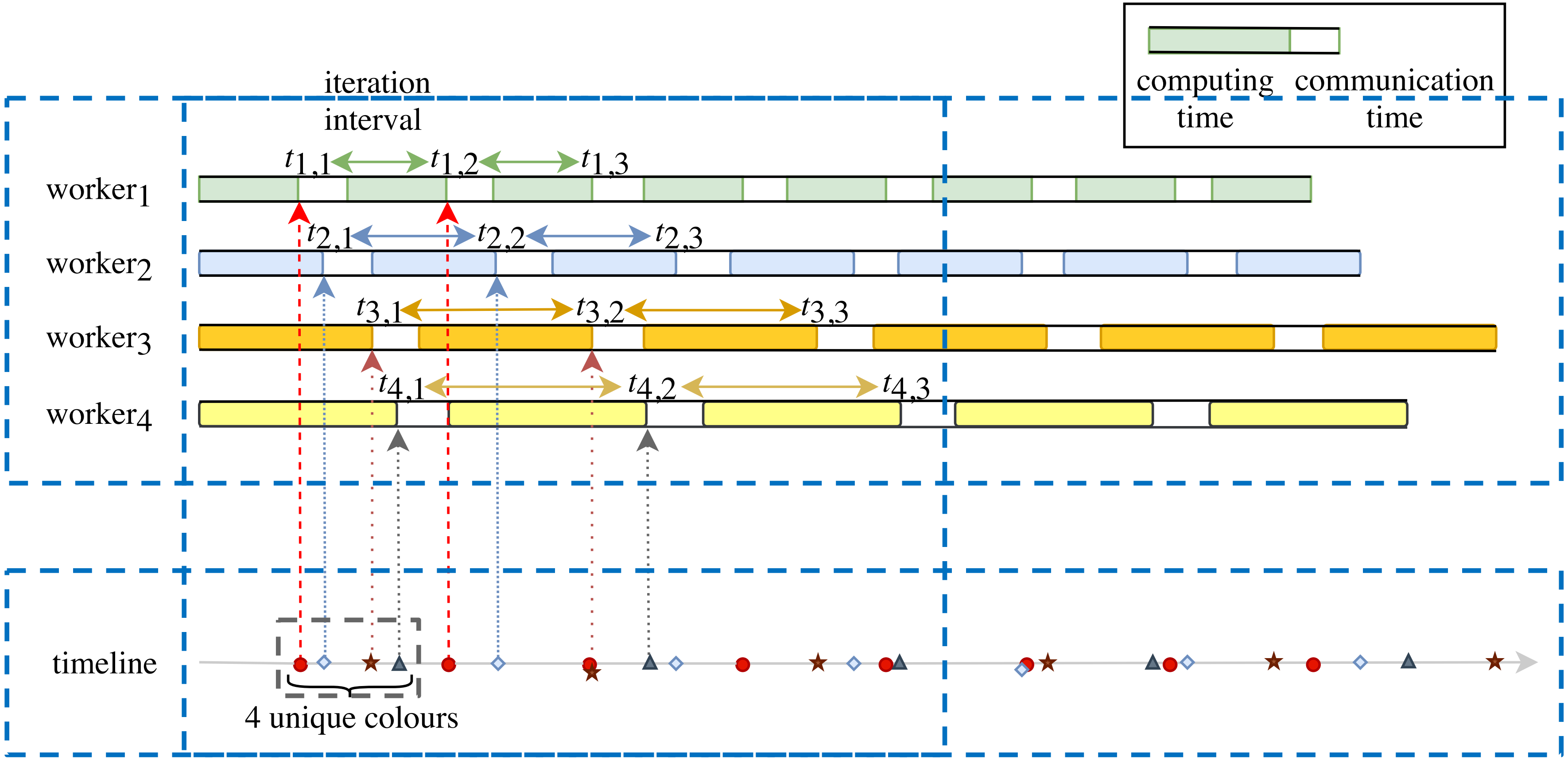
# Intuition about Our Algorithm

# Prediction and Search Method of ElasticBSP



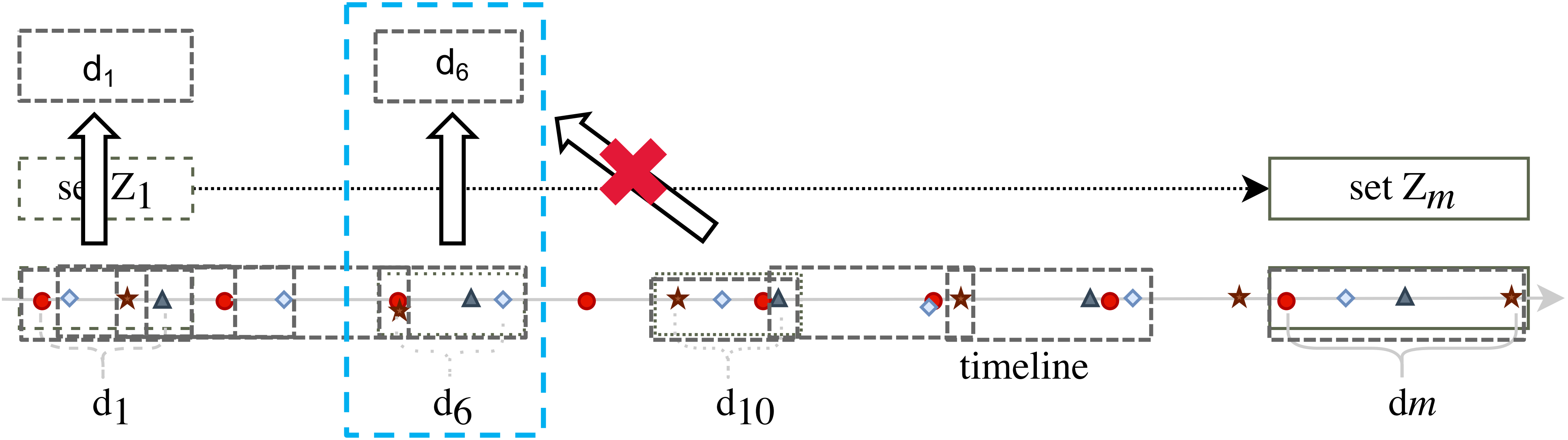

Search optimal waiting time for all *n* workers

# ZipLine Algorithm – Step 1: Preprocess Data




**time interval per iteration for each worker**

# ZipLine Algorithm – Step 2: Scan the Data in $O(Rn \cdot logn)$

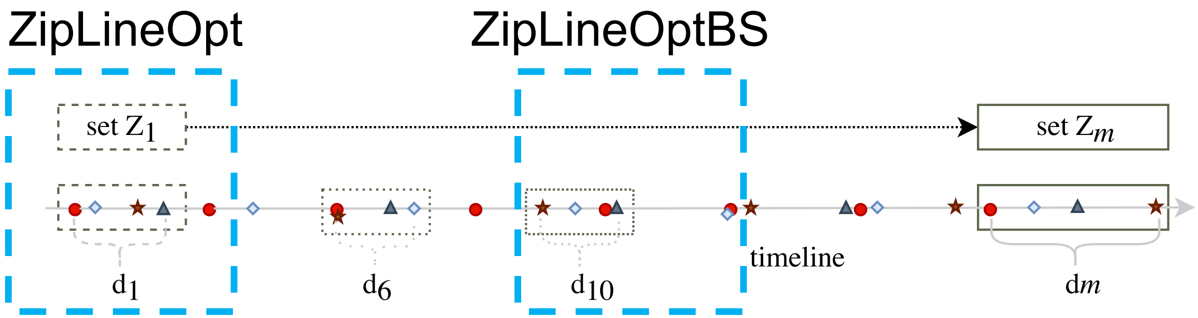



**ZipLine searching for the set Z* with the minimum difference d***

# Optimized ZipLine: ZipLineOpt and ZipLineOptBS

- ZipLineOpt: optimize ‘add’ operation of the scanning set Z
  - Using pruning to skip the search time of adding new element to set Z
    - Skip the search if new element has the same color as the leftmost element of Z

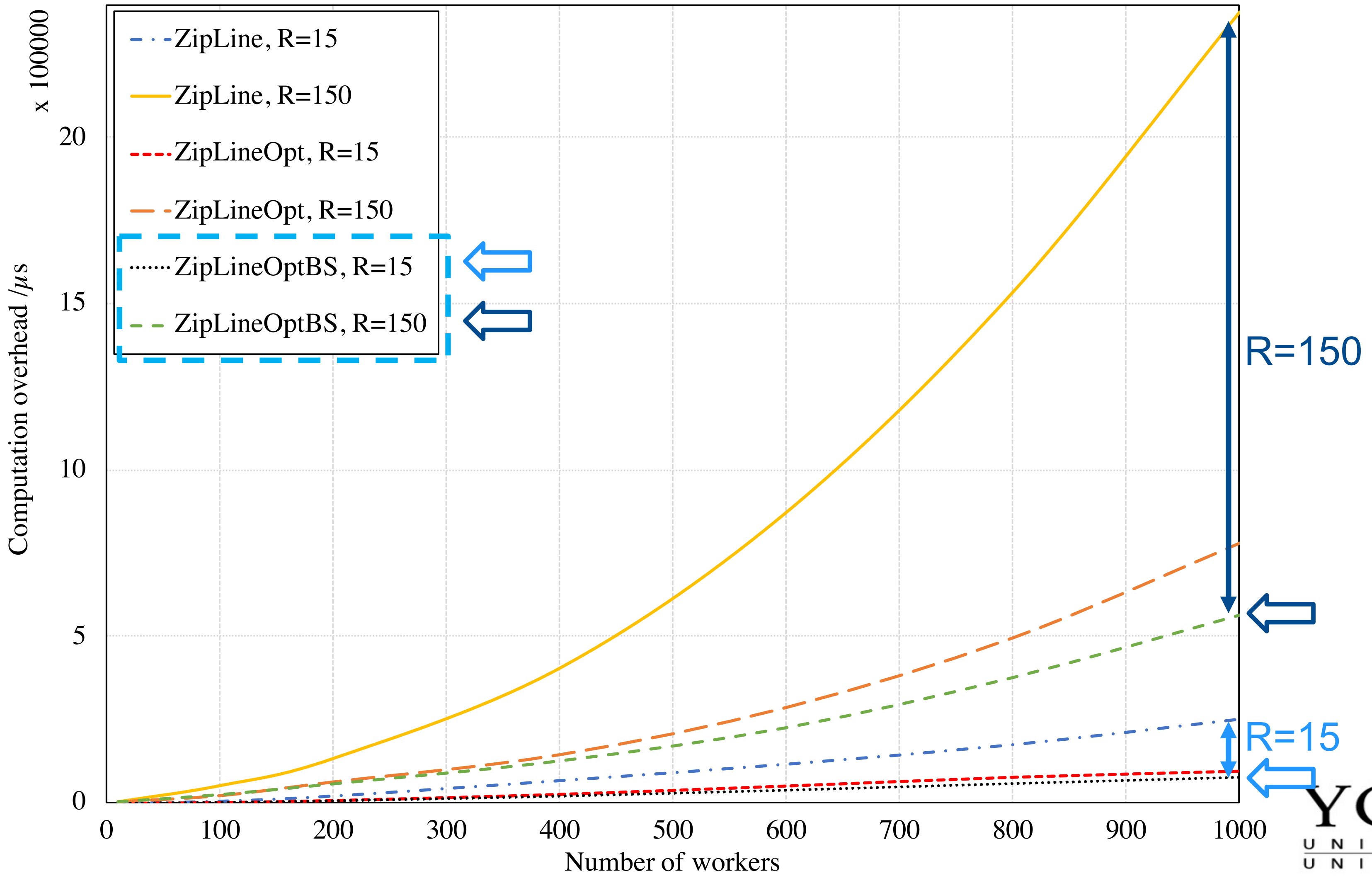
- ZipLineOptBS: further optimize ‘add’ operation of the scanning set Z
  - Using a Matrix containing info (e.g., timestamps) of data points to enable binary searching in set Z
  - Matrix: n (workers) x R (future iterations)



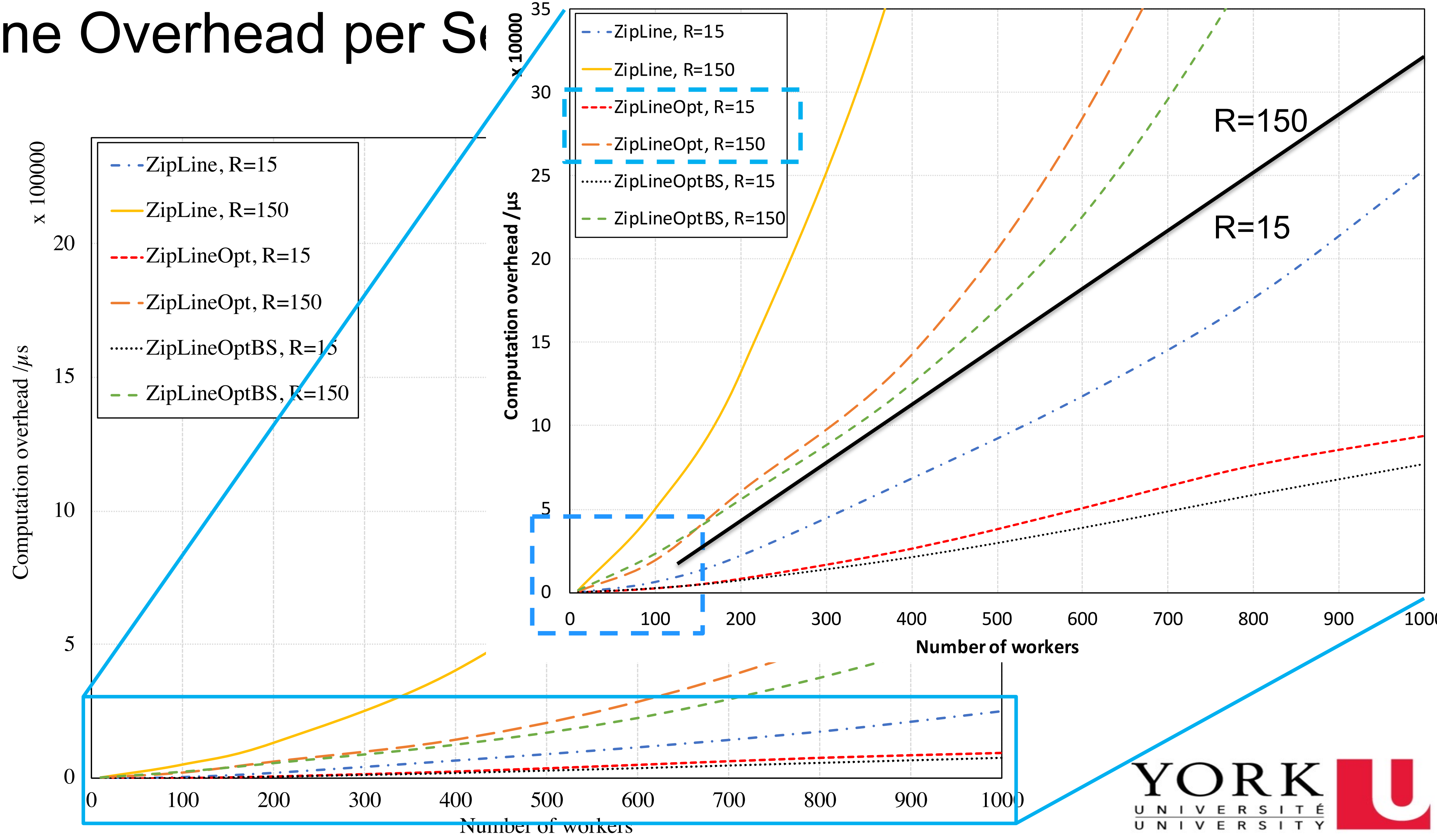

**ZipLine searching for the set Z* with the minimum difference d***

# Experimental Results

# ZipLine Computing Time per Search/Synchronization

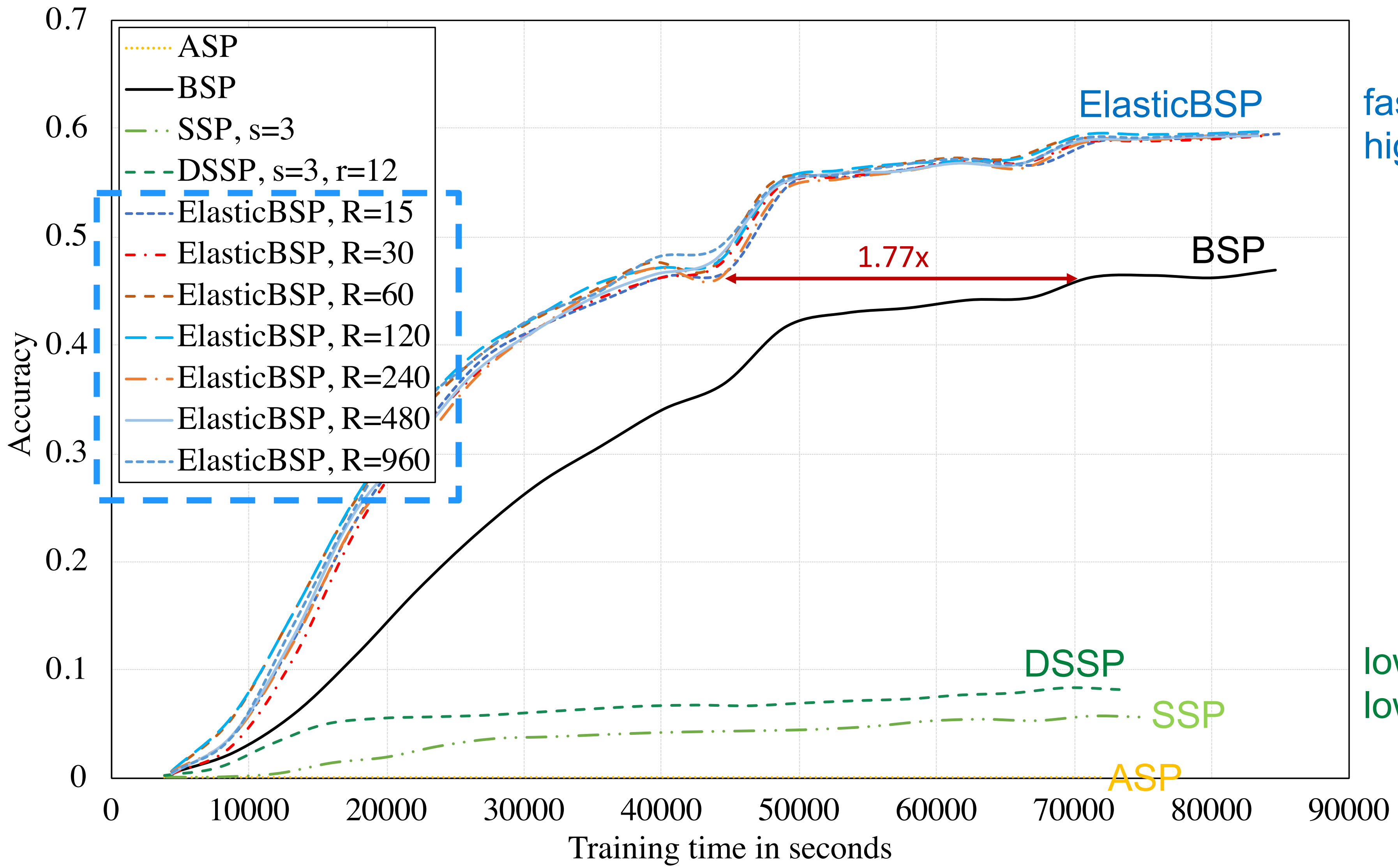


# ZipLine Overhead per Se




YORK UNIVERSITÉ UNIVERSITY

# Vgg16 on ImageNet1K with 4 workers




fast convergence, high accuracy

low convergence, low accuracy

# Conclusion

- ElasticBSP provides **faster convergence** to **higher accuracy** than BSP via
  - Increasing the iteration throughput
  - Limiting the staled gradients by elastic bulk synchronization

- ZipLine finds the **optimal** waiting time per synchronization
  - A greedy one-pass algorithm
  - Optimized versions: ZipLineOpt and ZipLineOptBS
  - ZipLineOptBS has linearithmic time complexity, O($Rn \cdot logn$)
    - $R$ is the # of predicted future iterations per worker
    - $n$ is the number of workers in a cluster