Ditto: Efficient Serverless Analytics with Elastic Parallelism

Chao Jin, Zili Zhang, Xingyu Xiang, Songyun Zou,
Gang Huang, Xuanzhe Liu, Xin Jin

北京大學
PEKING UNIVERSITY

Serverless computing

AWS Lambda

Azure Functions

Google Cloud Functions

Knative

Fine-grained resource elasticity

- Auto-scaling
- Concurrency from 1 to 1,000

Fine-grained billing

- 1 MB memory granularity
- 1 ms time granularity

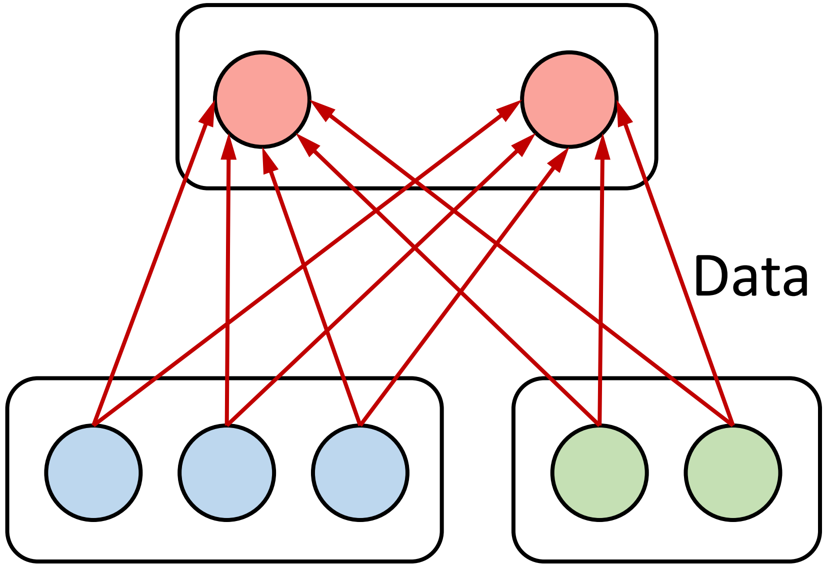
Serverless analytics

Big data & SQL-like query

Locus (NSDI'19)
NIMBLE (NSDI'21)

Databricks SQL Serverless
Azure Synapse Analytics
Google BigQuery

Job Execution DAG

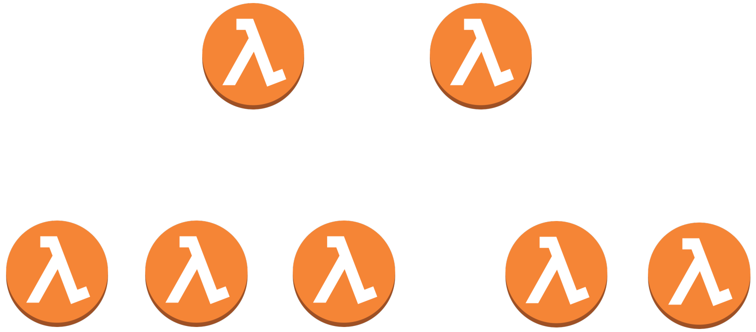


Data parallelism

Deploy

Data dependency

Serverless functions



Job completion time (JCT)

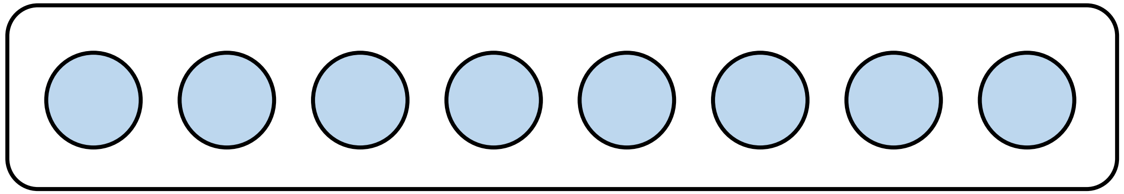
Cost ($\sum_{funcs}$ time×memroy)

Degree of Parallelism: a new problem

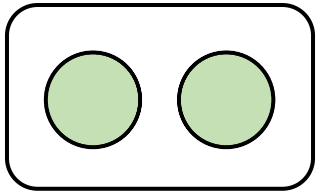
Fine-grained resource elasticity

Can existing parallelism configuration solutions optimize the performance goals in serverless settings?

Higher DoP
Faster, lower JCT



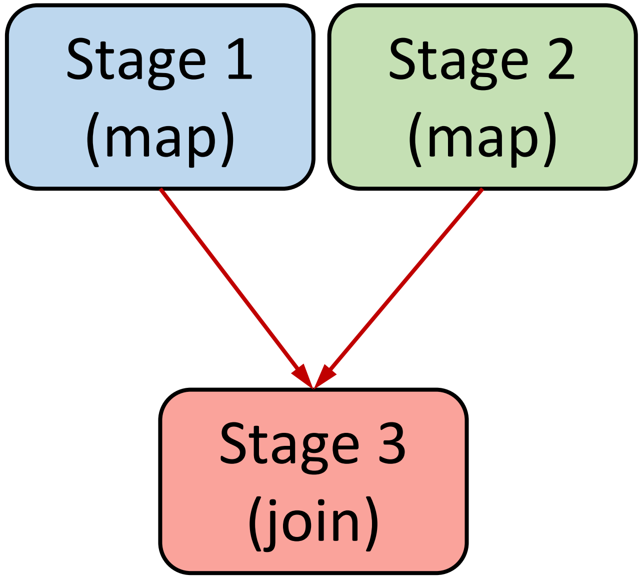
Lower DoP
Lower cost

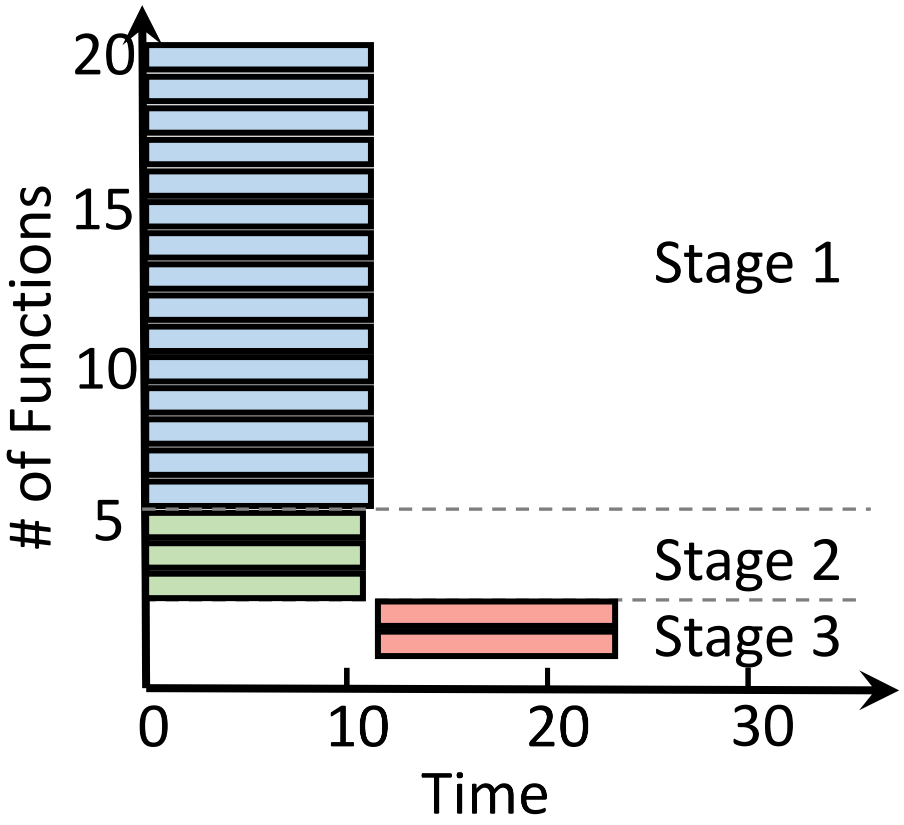


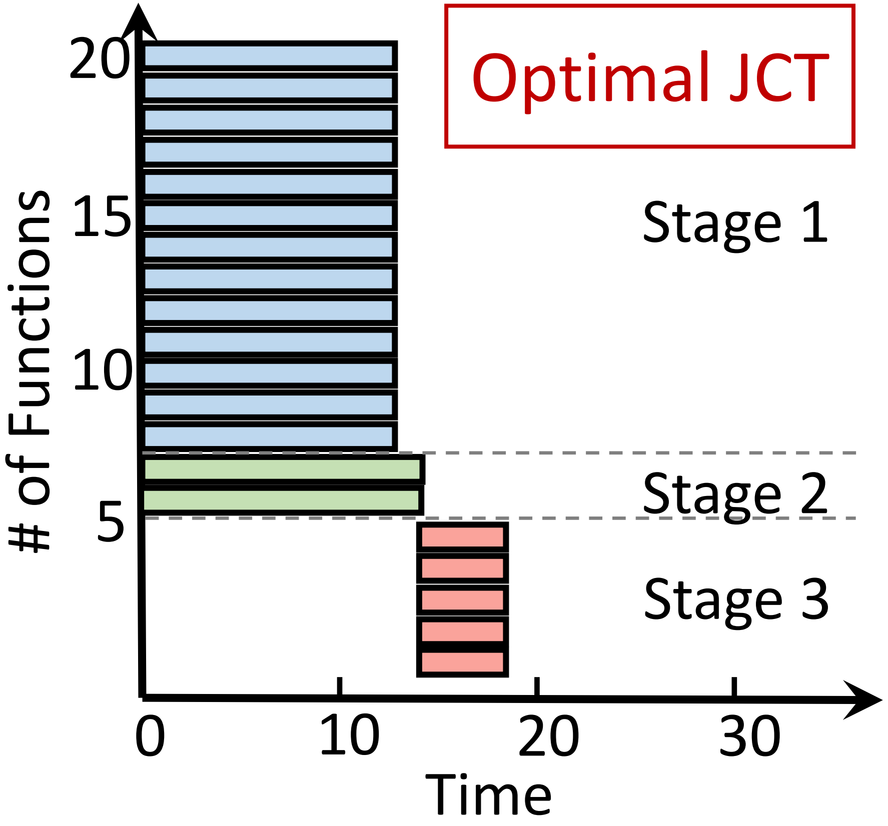
NIMBLE: a **data** perspective

DoP proportional to input **data** size

Caerus: NIMBLE Task Scheduling for Serverless Analytics







Elastic parallelism

Data? ✗

Time? ✓

Job completion time (JCT)

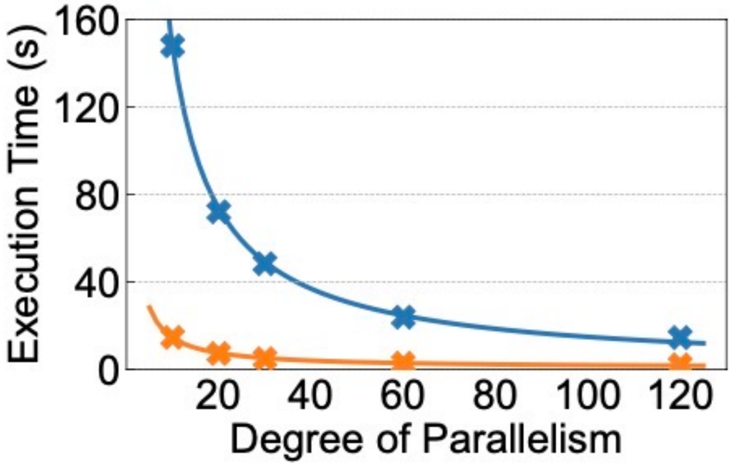
Cost ($\sum_{funcs} \text{time} \times \text{memroy}$)

Main idea:

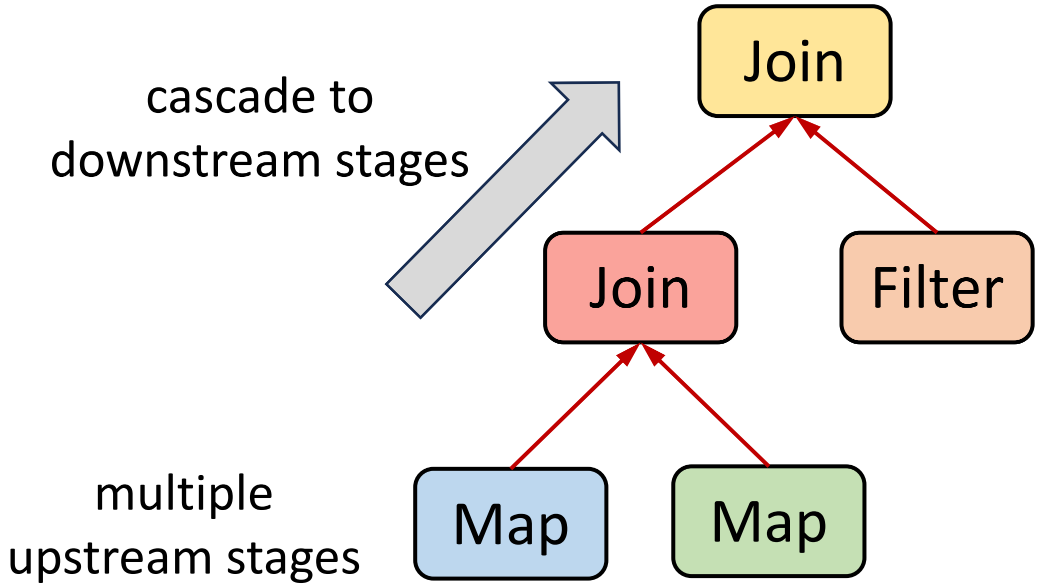
- Match the **resource elasticity** of serverless computing with **parallelism scheduling** in data analytics
- Optimize serverless performance goals directly from a perspective of **time**

Challenge 1:
Optimal parallelism for arbitrary DAGs

- Accurate prediction of the execution time under dynamic parallelism configurations



- Consider data dependencies

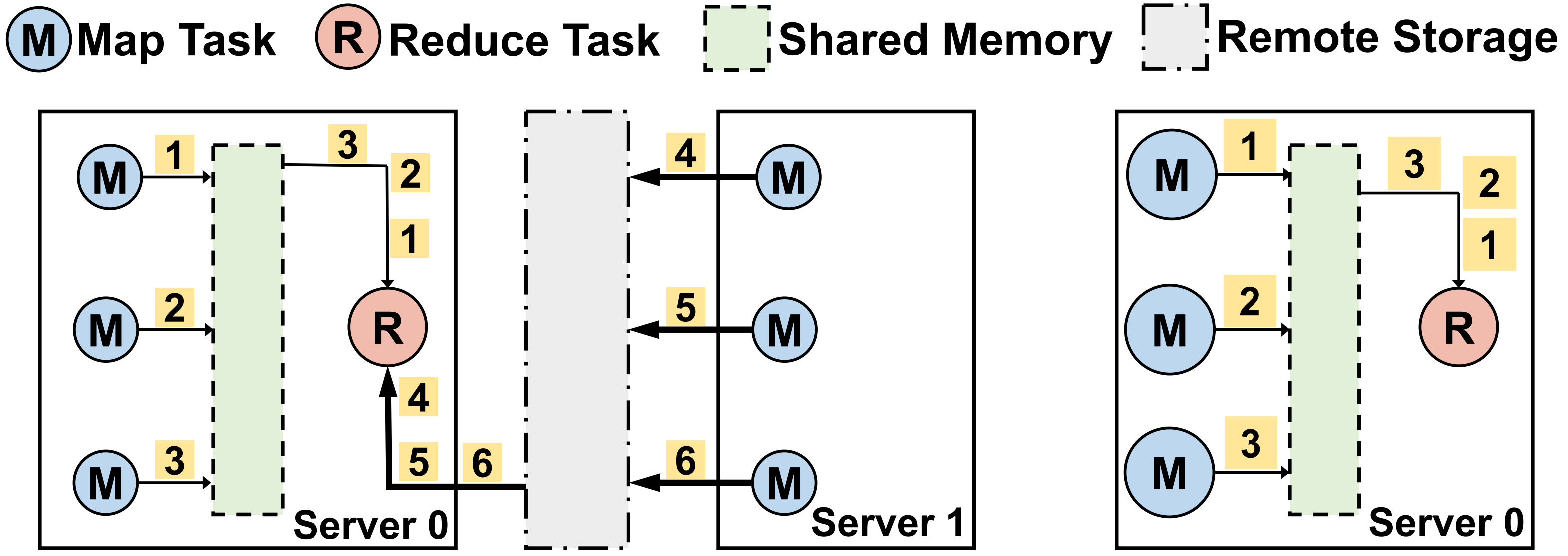


Challenge 2:
Coupling of parallelism and placement

- Co-optimize parallelism configuration and function placement

Shared memory

SPRIGHT (SIGCOMM'22)
Pheromone (NSDI'23)



High DoP with heavy data shuffle time

Low DoP with almost zero data shuffle time

Ditto design outline

Challenge 1: How to find the optimal parallelism for arbitrary DAGs?

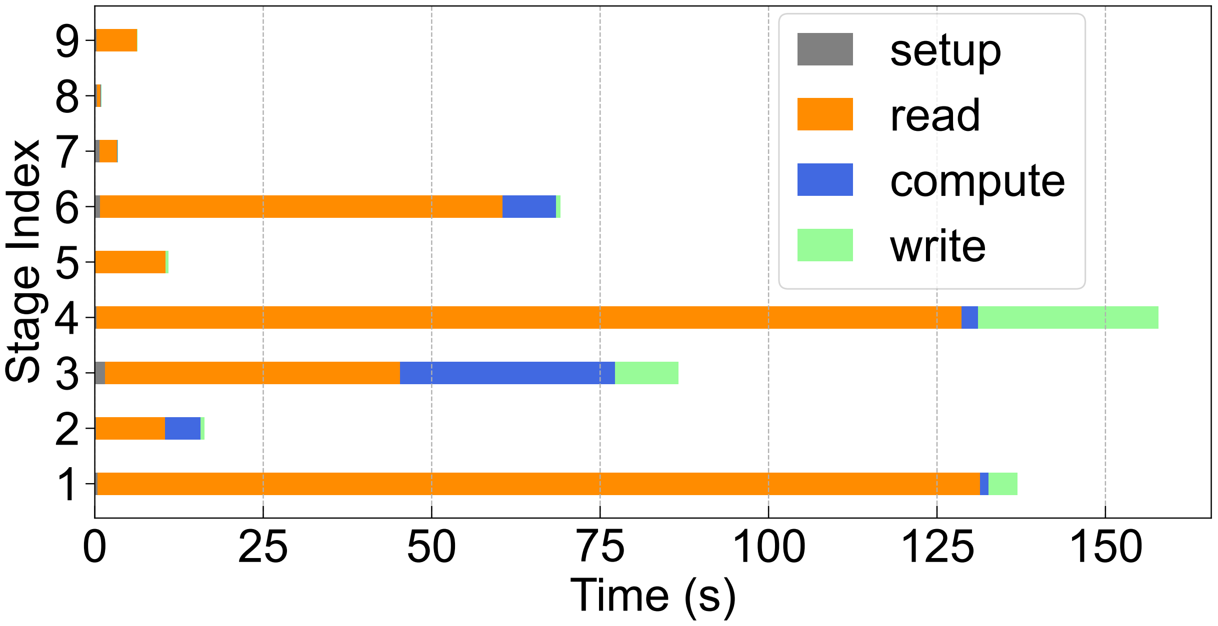
- Execution time model → Time under dynamic parallelism
- DoP ratio computing → Optimal parallelism configuration

Challenge 2: How to optimize the coupled parallelism and placement?

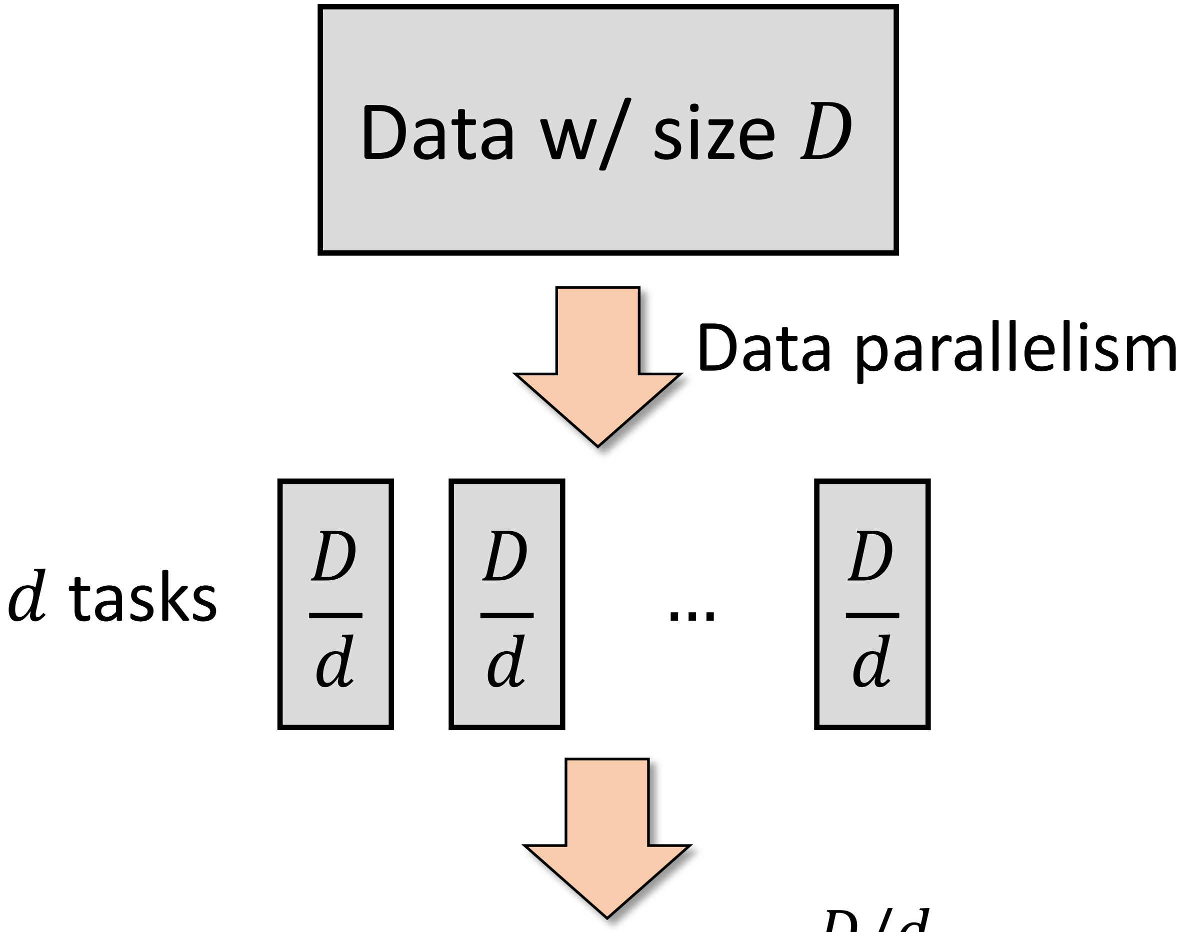
- Greedy grouping → Eliminate high data shuffling overhead
- Joint iterative optimization → Co-scheduling

Execution time model: a **time** perspective

- Long running: 10 to 1000 seconds
- Data I/O dominates



Time breakdown for TPC-DS Q95



$$T_{exec} \sim T_{IO} \sim \frac{D/d}{Bandwidth}$$

Execution time model: a **time** perspective

$$\boxed{T(s)} = \boxed{\frac{\alpha}{d}} + \boxed{\beta}$$

Execution time of stage s

Parallelized time

Inherent time

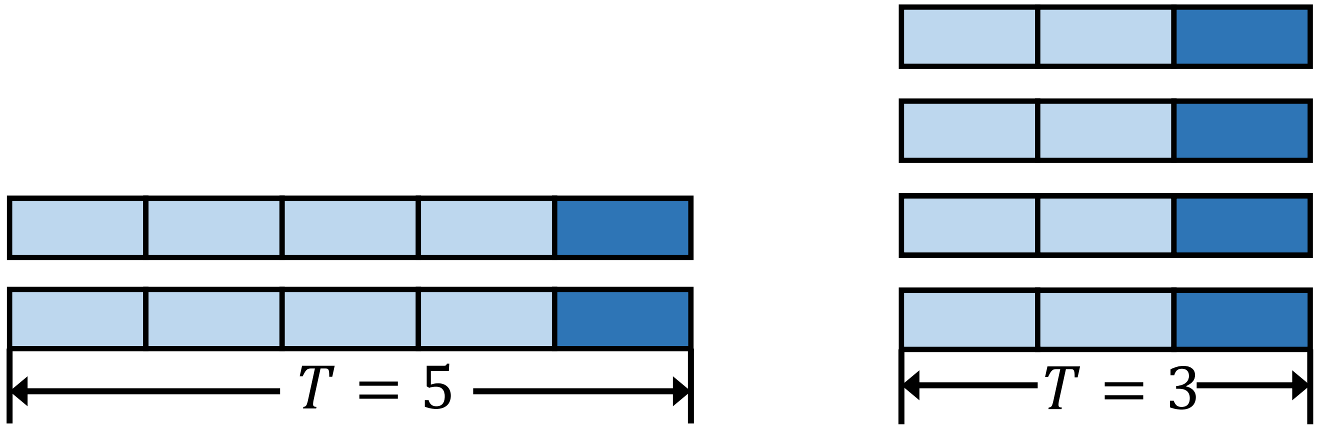
d: degree of parallelism, DoP

α: the parallelized time parameter

$\alpha = 8,\ \beta = 1$

Parallelized time unit

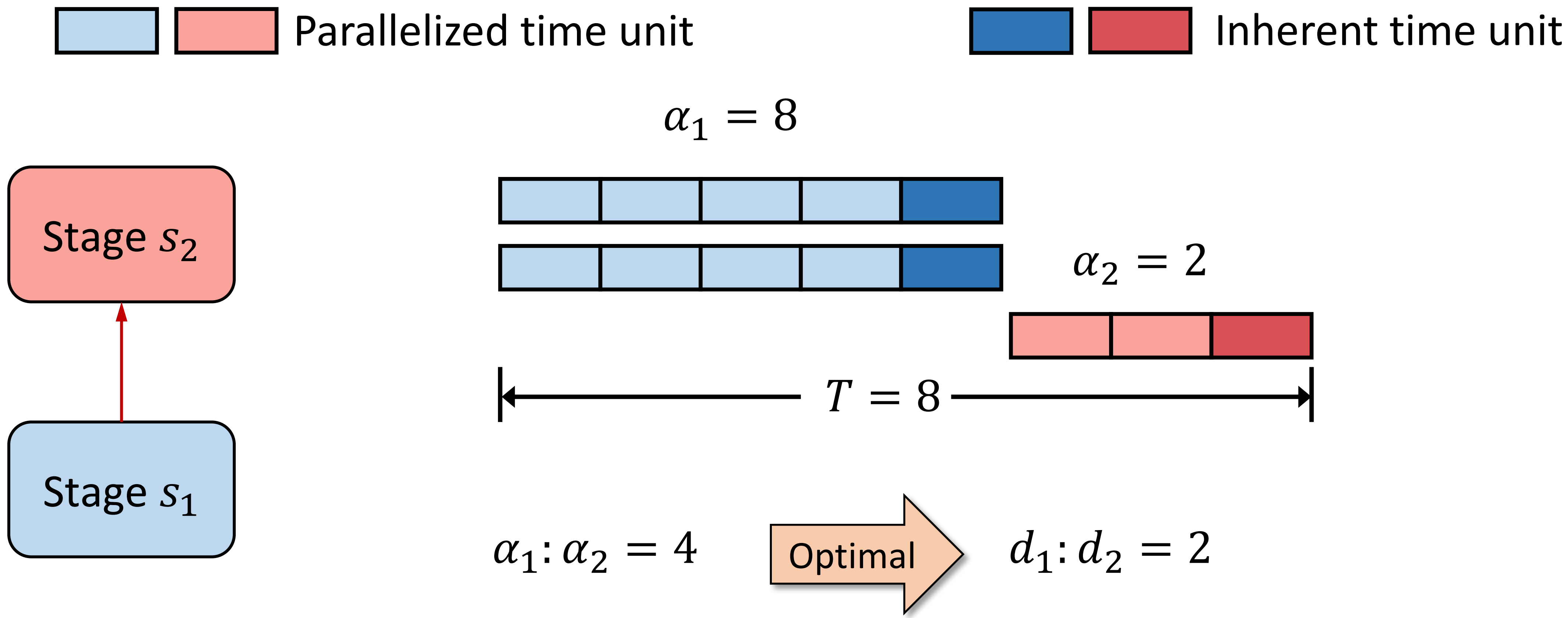
Inherent time unit



$d = 2$ $\quad$ $d = 4$

DoP ratio computing

Intra-path DoP ratio: minimize the sum of the two stages' execution time

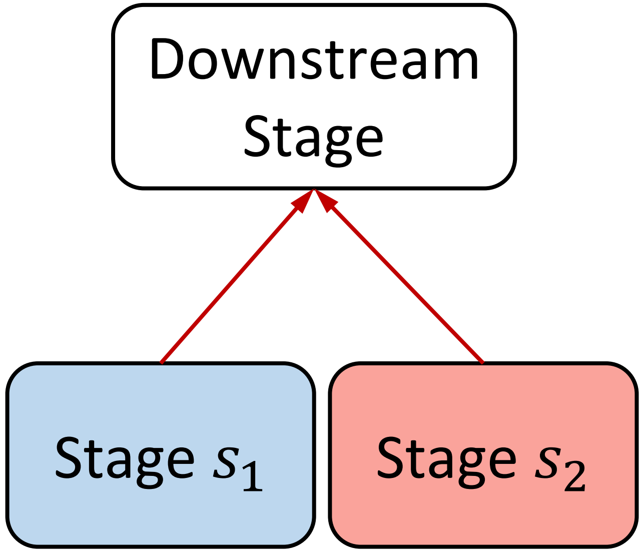


$$d_1 : d_2 = \sqrt{\alpha_1 : \alpha_2}$$

DoP ratio computing

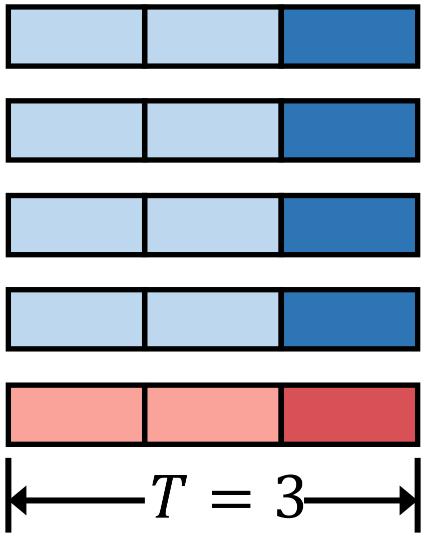
Inter-path DoP ratio: balance the two stages' time

Parallelized time unit   Inherent time unit



$\alpha_1 = 8$

$\alpha_2 = 2$



$\alpha_1 : \alpha_2 = 4$ Optimal $d_1 : d_2 = 4$

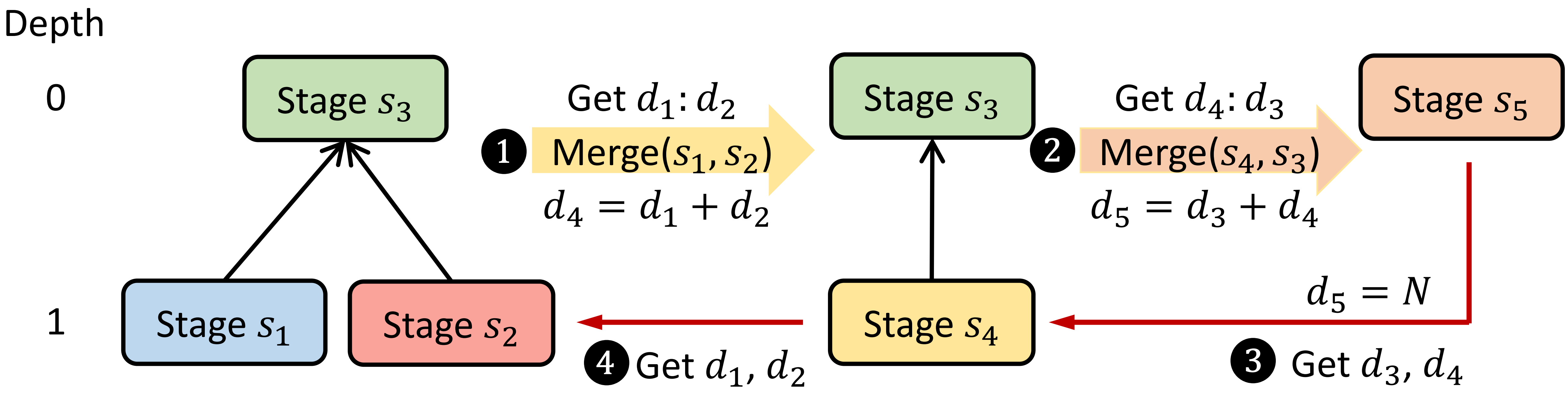
$$d_1 : d_2 = \alpha_1 : \alpha_2$$

DoP ratio computing

Stage merging: a new stage also conforms to the execution time model

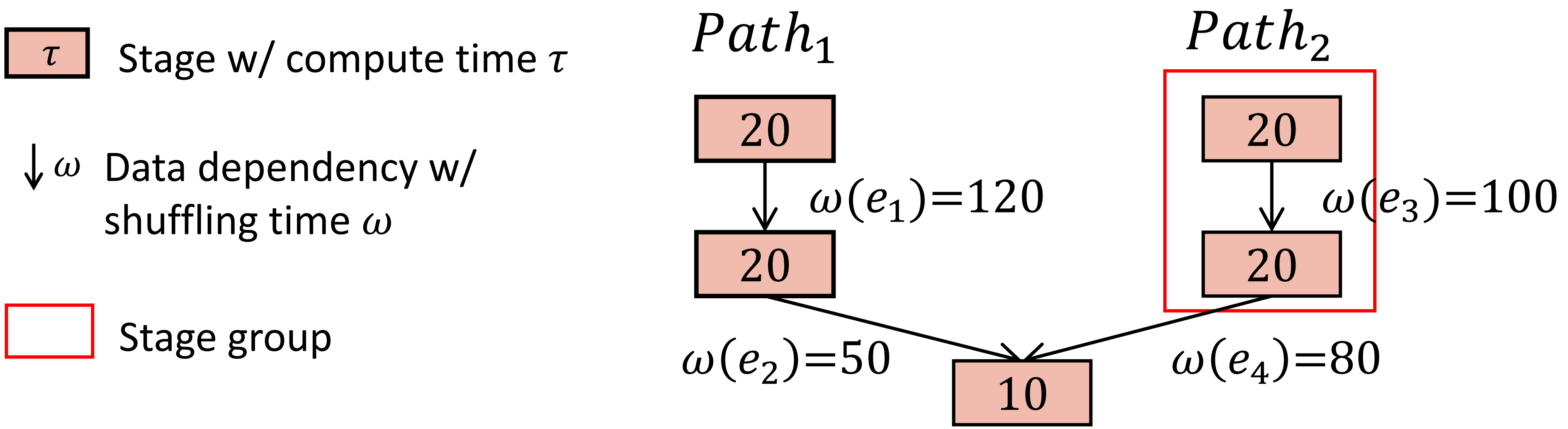
d_i : degree of parallelism of stage s_i

N : total number of functions



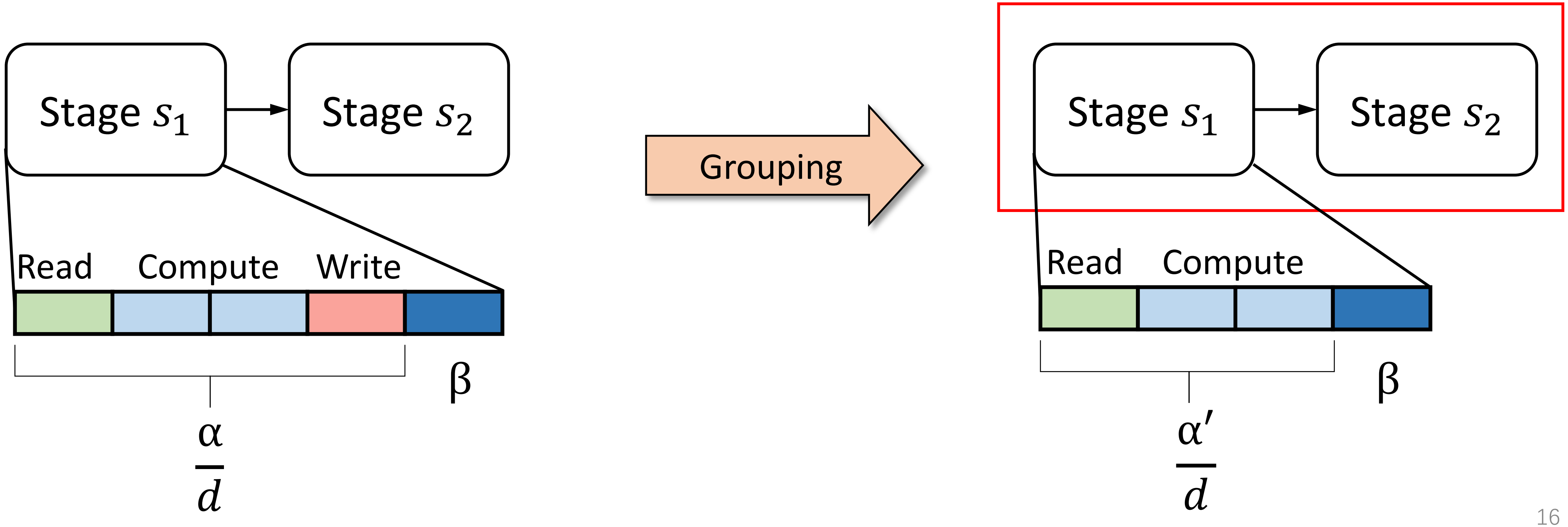
Greedy grouping

- **Stage group**: stages that should communicate via shared memory
  - **NP-hard**
- **Greedy order**: group stages with high shuffling overhead
  - For JCT optimization, the highest on the critical path first

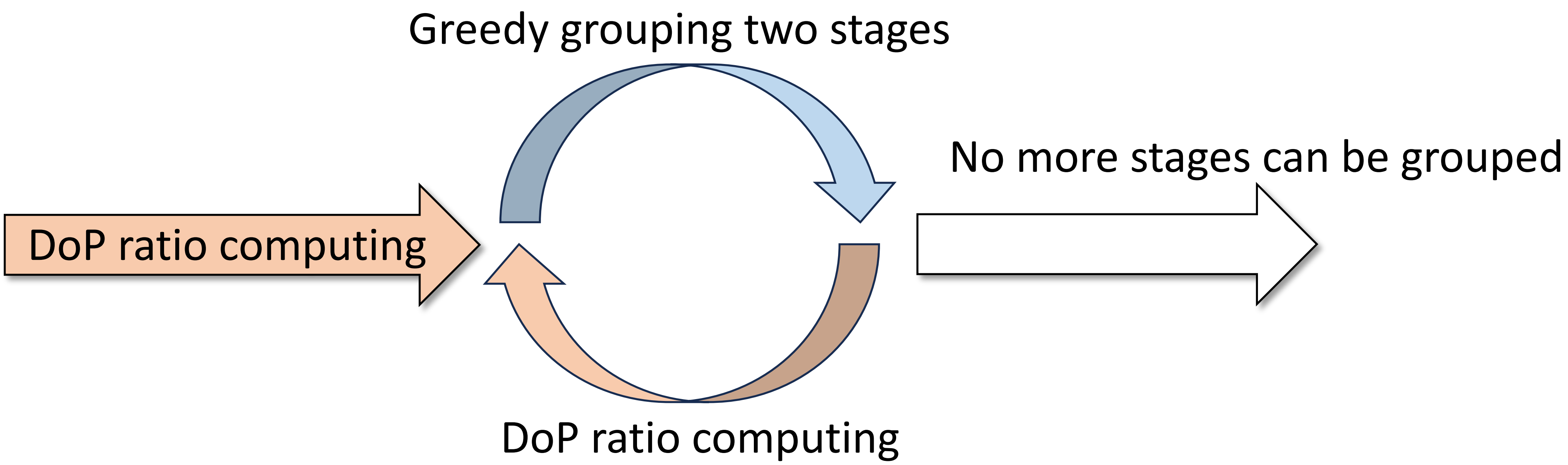


Joint iterative optimization

- $\boldsymbol{\alpha}$ will decrease as the I/O time reduces to zero after grouping
  - Model the I/O and compute parts of $\boldsymbol{\alpha}$ separately
  - Combine with DoP ratio computing into joint optimization



Joint iterative optimization



- Each stage is a group initially
- In each iteration
  - group two stages (or stage groups) with the highest shuffling overhead
  - recalculate the new optimal parallelism configuration

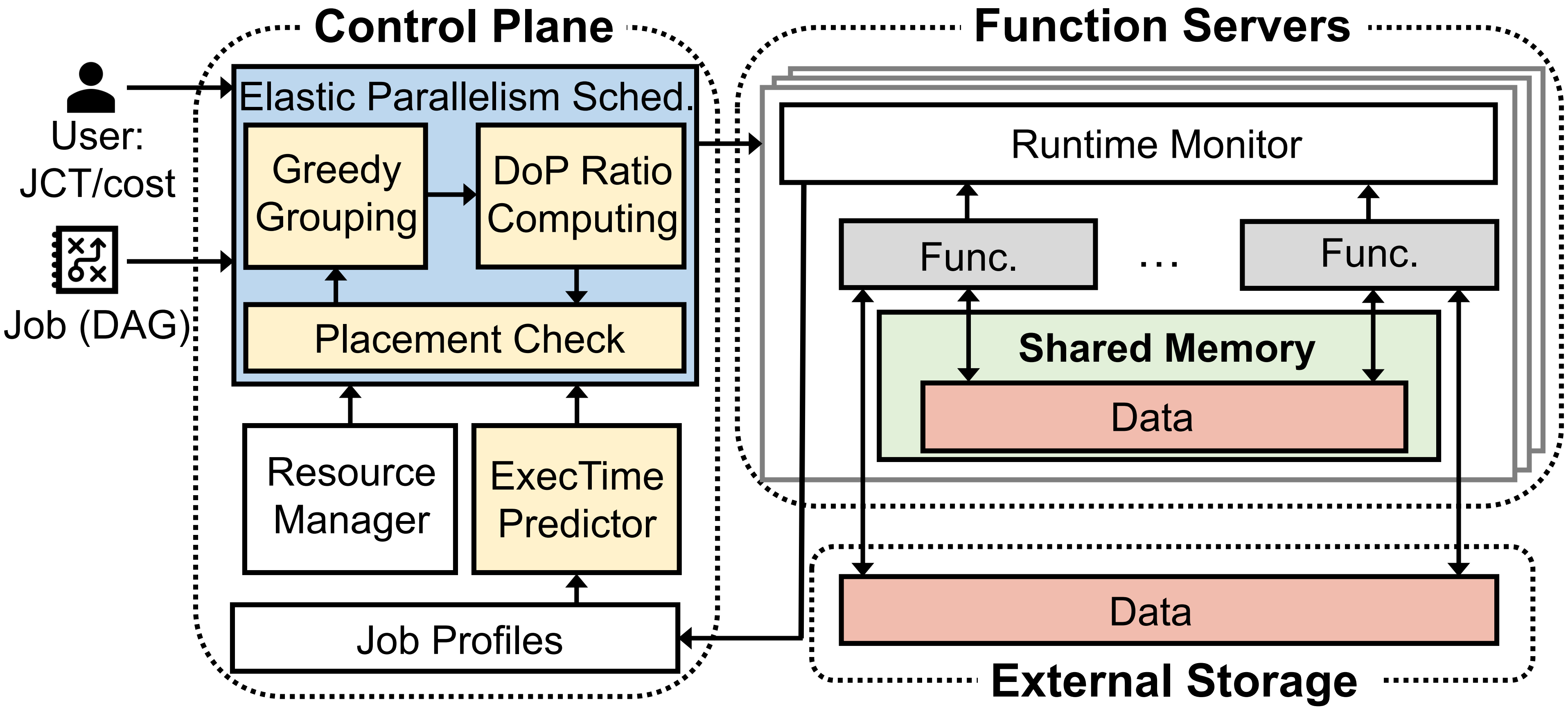
Cost optimization

- DoP ratio computing applies **serverless cost model**
  - Function cost: consider the resource usage
  - Total cost: the sum of all function costs
- Greedy grouping groups stages with **highest shuffling cost first**

- Please refer to our paper for more details!

Ditto System

Implement Ditto on top of SPRIGHT (SIGCOMM' 22)



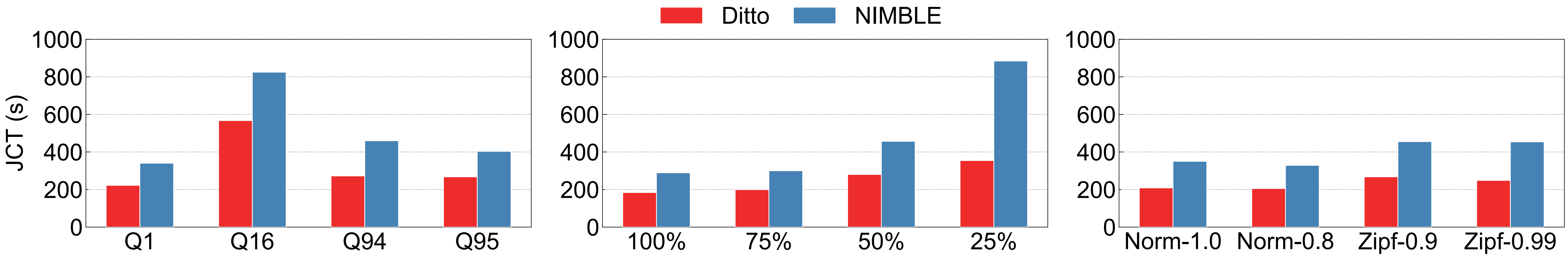
Evaluation

- Setup on AWS
  - Scheduling: one m6i.4xlarge server
  - Compute: eight m6i.24xlarge servers (96 vCPUs & 384 GB DRAM each)
  - Storage: S3
- TPC-DS
  - Q1, Q16, Q94, Q95
    - `groupby, filter, join`
  - 1 TB data

```
select
  count(distinct ws_order_number) as "order count",
  sum(ws_ext_ship_cost) as "total shipping cost",
  sum(ws_net_profit) as "total net profit"
from
  web_sales ws1,
  date_dim,
  customer_address,
  web_site
where
  d_date between '1999-4-01'
  and (cast('1999-4-01' as date) + 60 days)
  and ws1.ws_ship_date_sk = d_date_sk
  and ws1.ws_ship_addr_sk = ca_address_sk
  and ca_state = 'IA'
  and ws1.ws_web_site_sk = web_site_sk
  and web_company_name = 'pri'
```

Evaluation

- Ditto reduces the JCT by 1.3-2.5X compared to NIMBLE



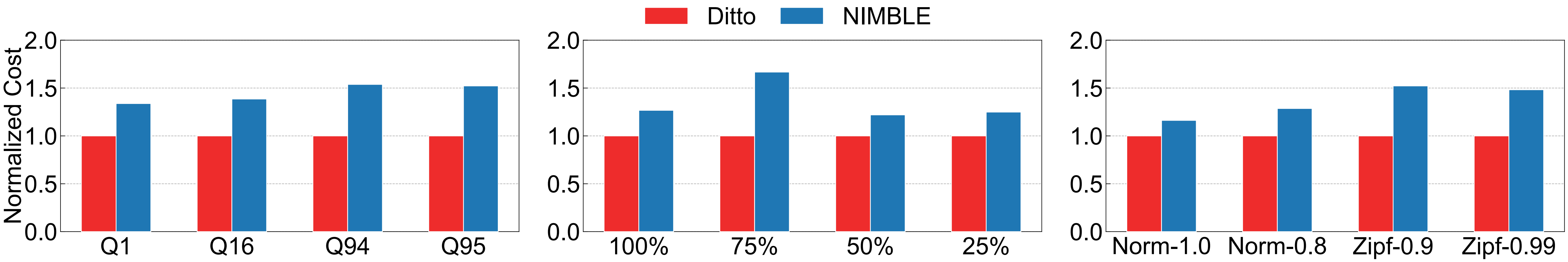
Four queries

Different resource usage

Different resource distributions

Evaluation

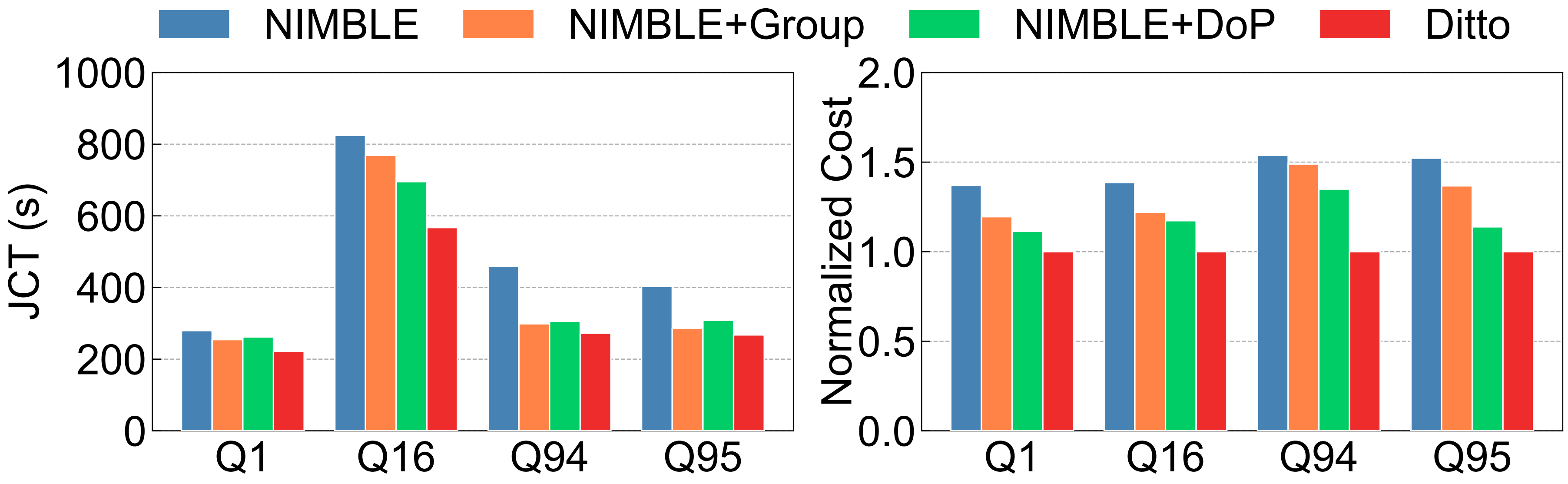
- Ditto reduces the cost by 1.2-1.7X compared to NIMBLE



Four queries | Different resource usage | Different resource distributions

Evaluation

- Ablation experiment to verify the effectiveness of Ditto



Evaluation

- Performance under Redis
- Accuracy of the execution time model
- Execution breakdown for TPC-DS Query 95
- System overhead of Ditto

Conclusion

- **Serverless analytics introduces the elastic parallelism scheduling problem to optimize serverless performance goals, i.e., JCT and cost**
- **Ditto co-optimizes parallelism configuration and function placement from the perspective of time**
  - Execution time model under dynamic parallelism
  - DoP ratio computing to achieve optimal JCT or cost
  - Joint iterative optimization for both parallelism and placement
- **Ditto reduces up to 2.5X in JCT and up to 1.7X on cost compared to NIMBLE**

Thank you!

chaojin@pku.edu.cn