IEEE 2024 International Conference on Web Services (ICWS)

SFSM: A Serverless Function Scheduling Method for FaaS Applications over Edge Computing

Hao Tian, Cheng Chen, Fei Dai, Wanchun Dou

南京大學 NANJING UNIVERSITY

西南林業大學 SOUTHWEST FORESTRY UNIVERSITY

Serverless Computing

aws Lambda

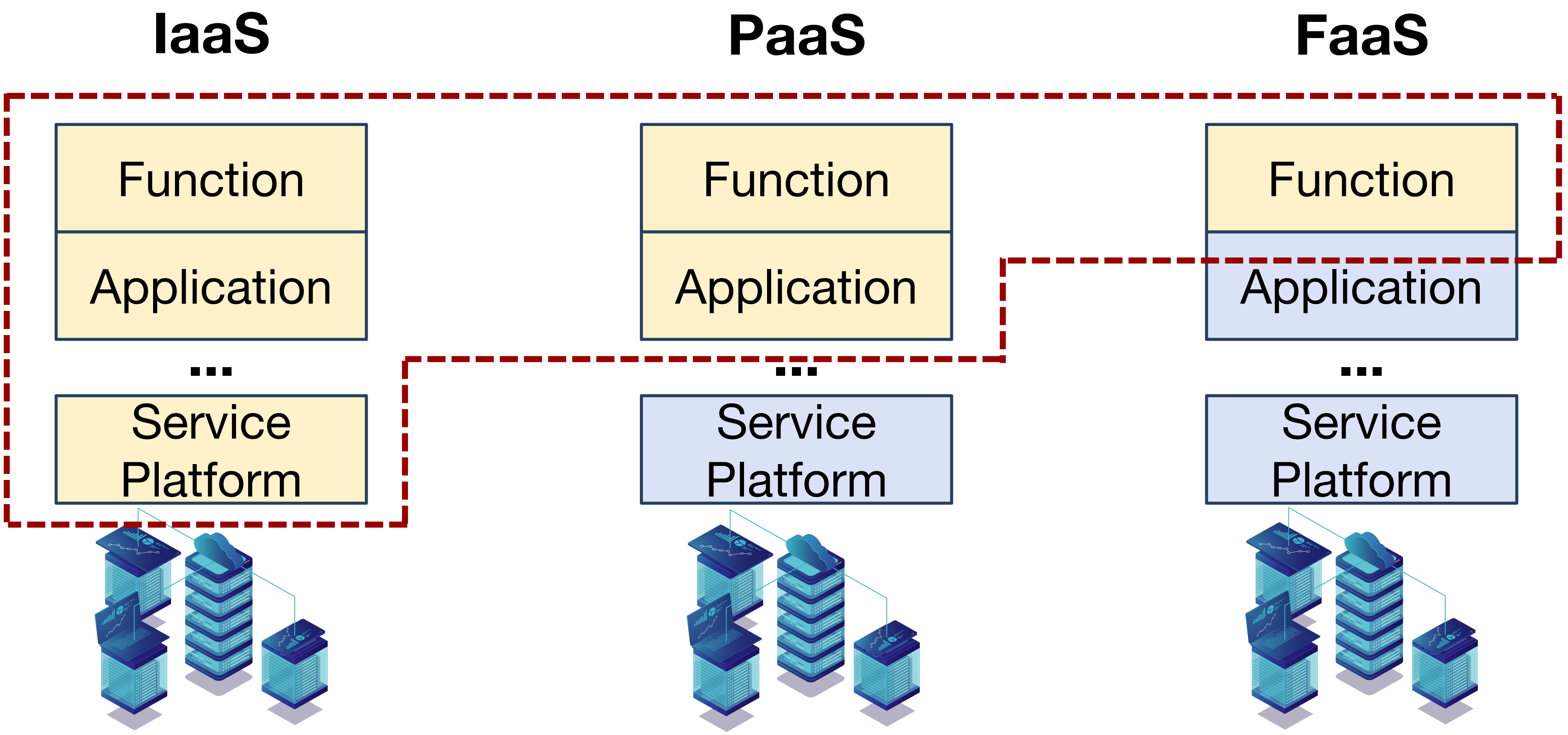
Google Cloud Functions

Azure Functions

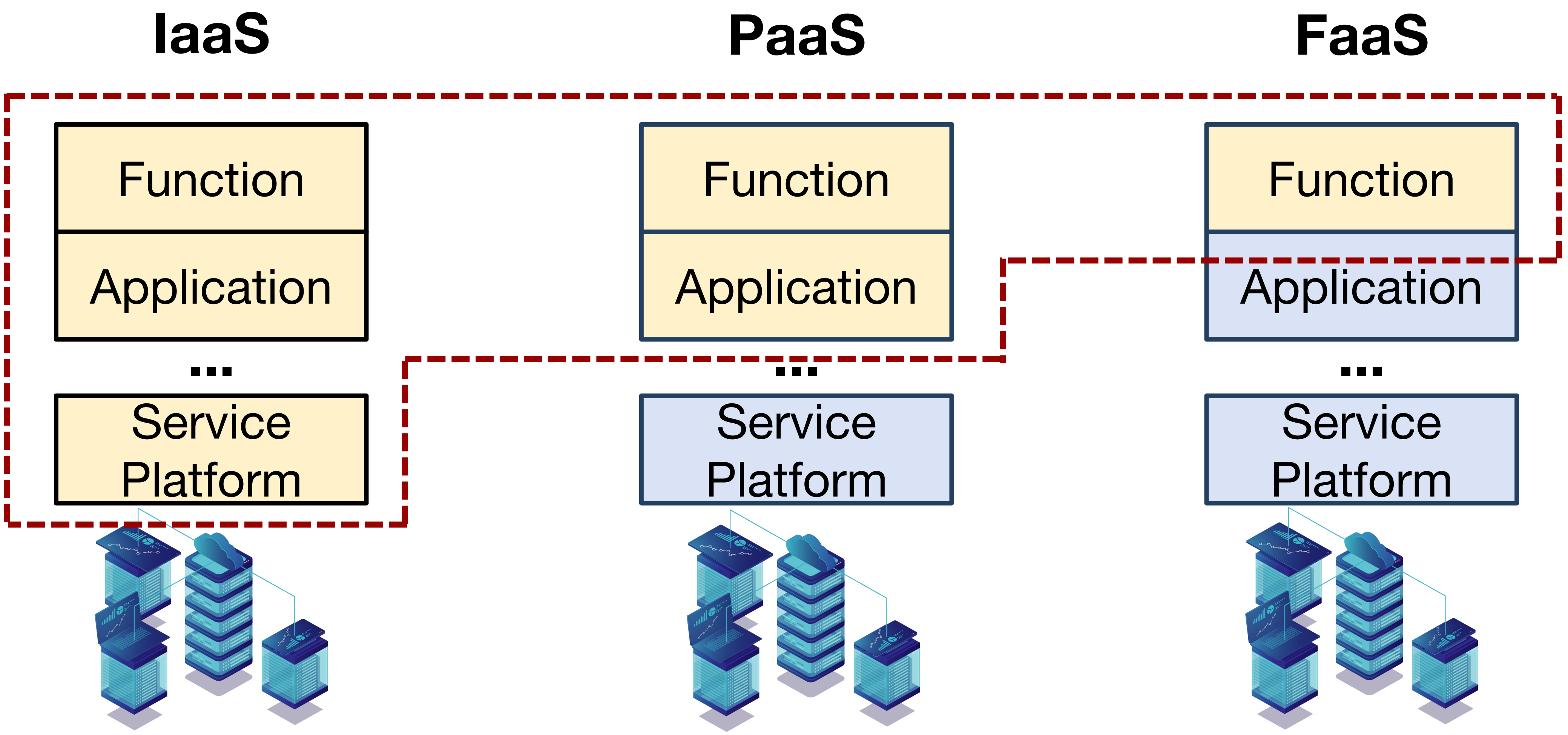
IBM Cloud Functions

[1] AWS Lambda: https://aws.amazon.com/lambda
[2] Google Cloud Functions: https://cloud.google.com/functions
[3] Microsoft Azure Functions: https://azure.microsoft.com/en-us/products/functions
[4] IBM Cloud Functions: https://cloud.ibm.com/functions

Function as a Services (FaaS)



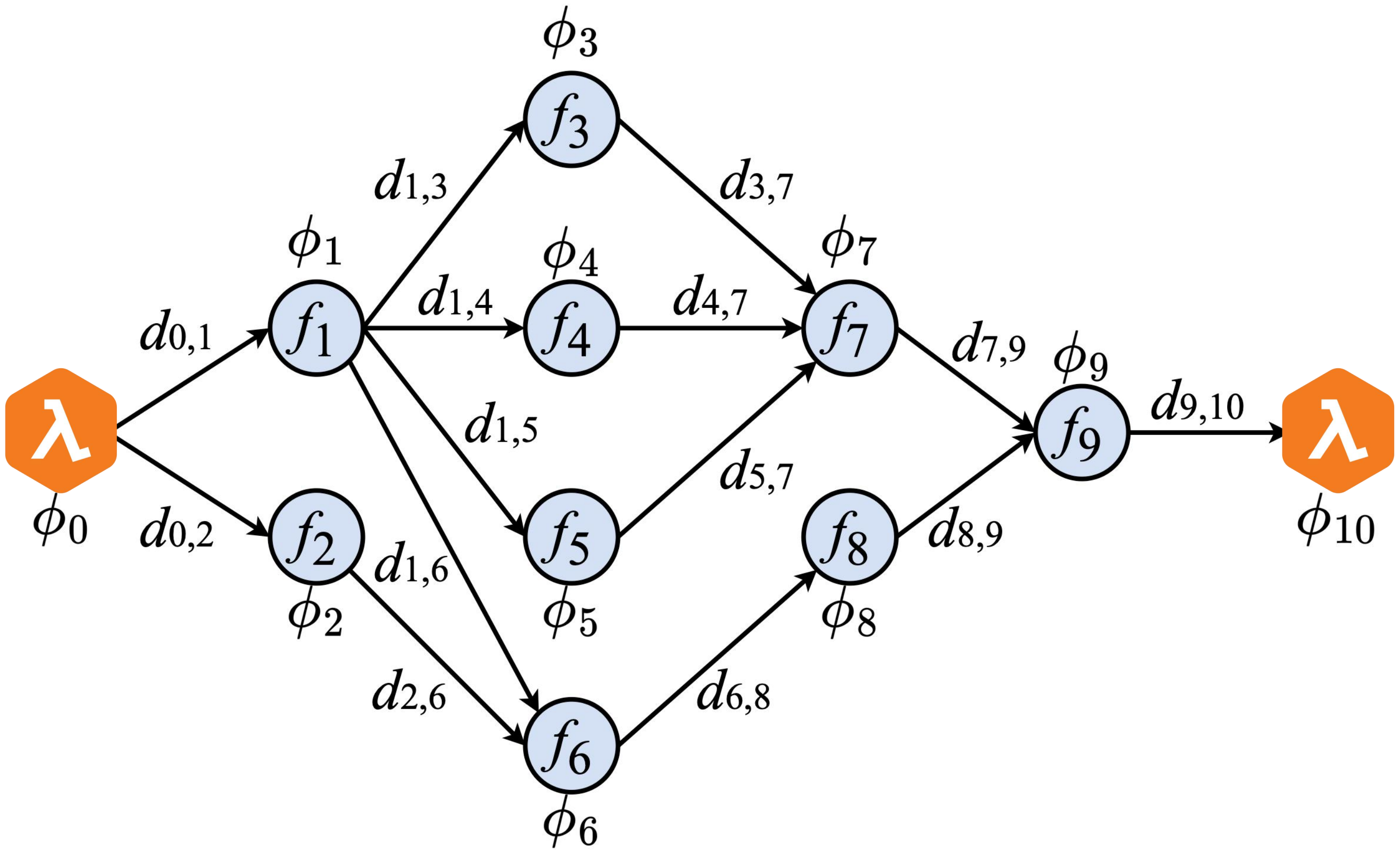
Function as a Services (FaaS)



Pay only for what you use!

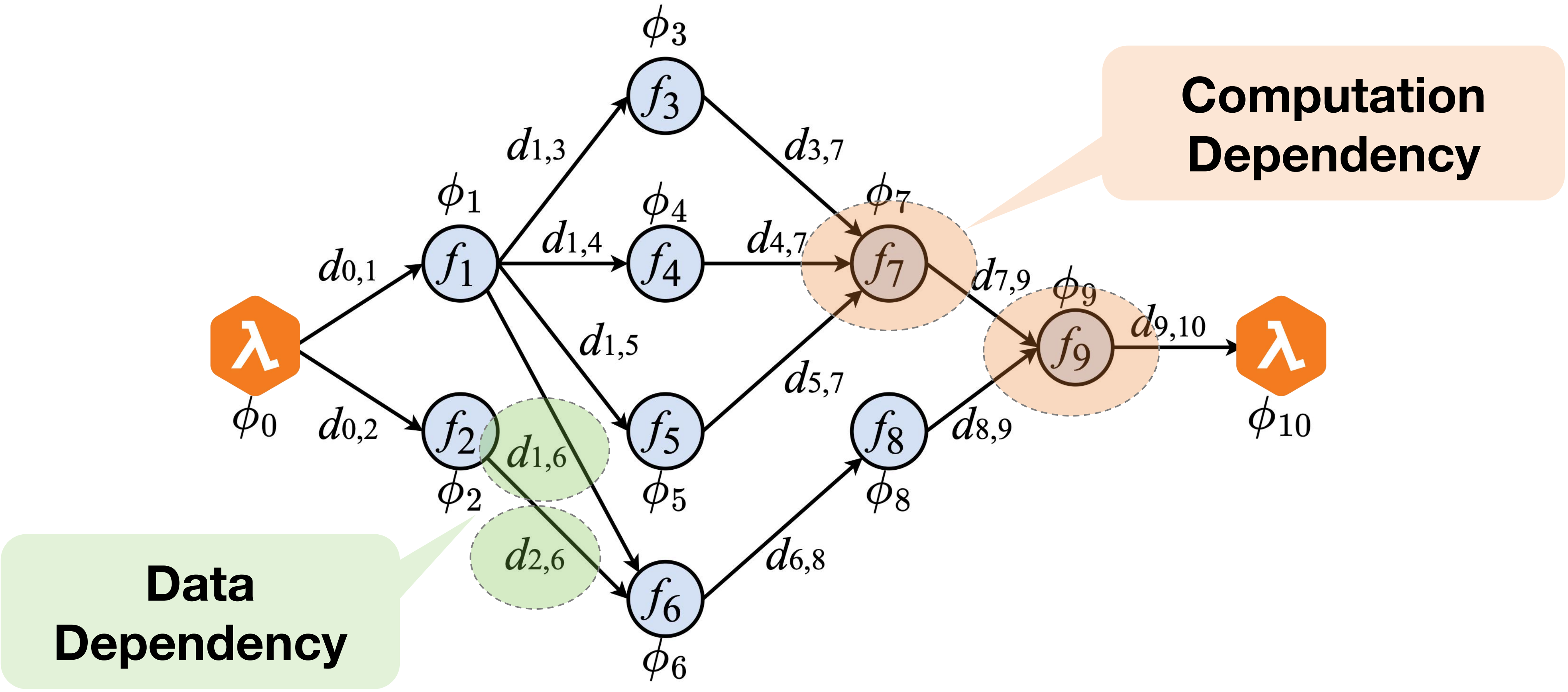
Serverless Workloads

- Abstract as a Direct Acyclic Graph (DAG)
- Constrained by computation & data dependencies



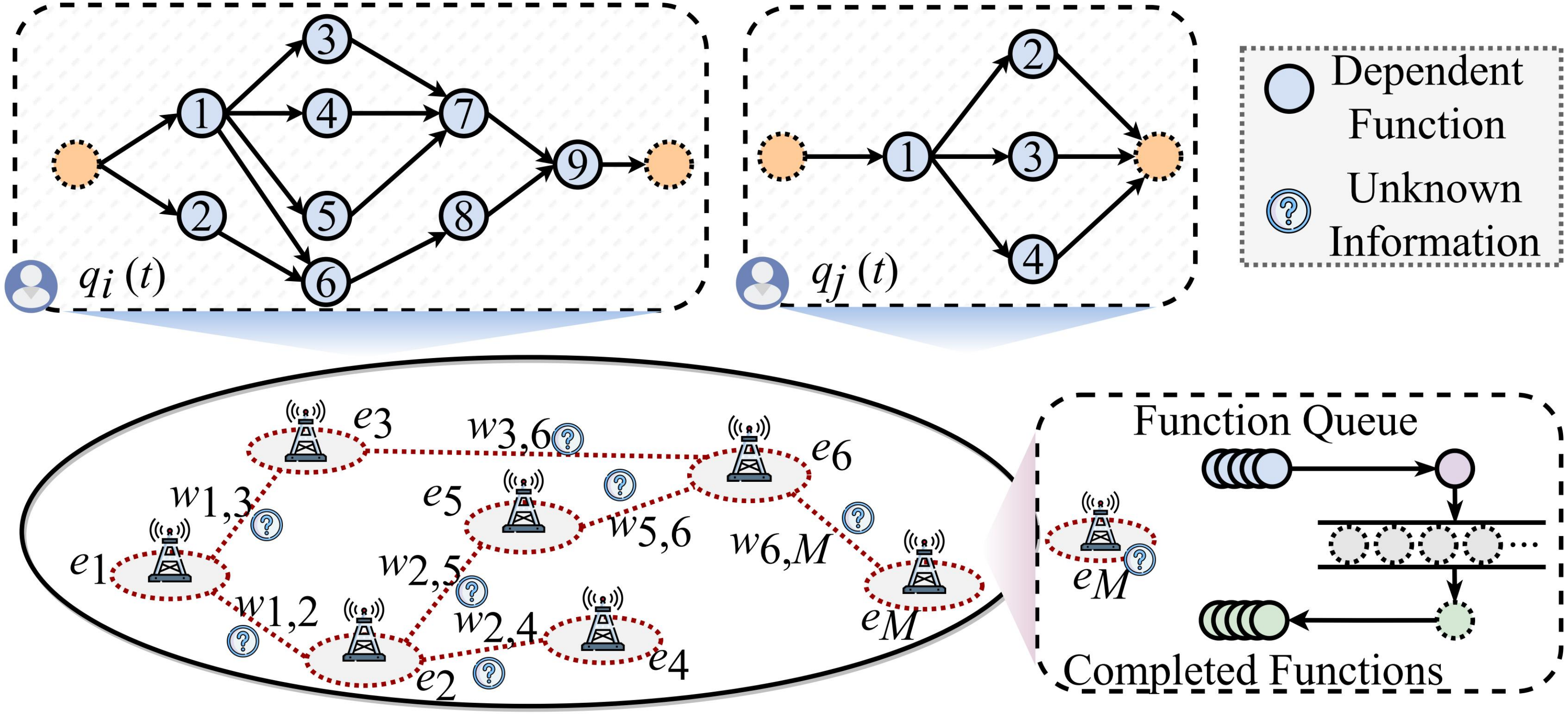
Serverless Workloads

- Abstract as a Direct Acyclic Graph (DAG)
- Constrained by computation & data dependencies



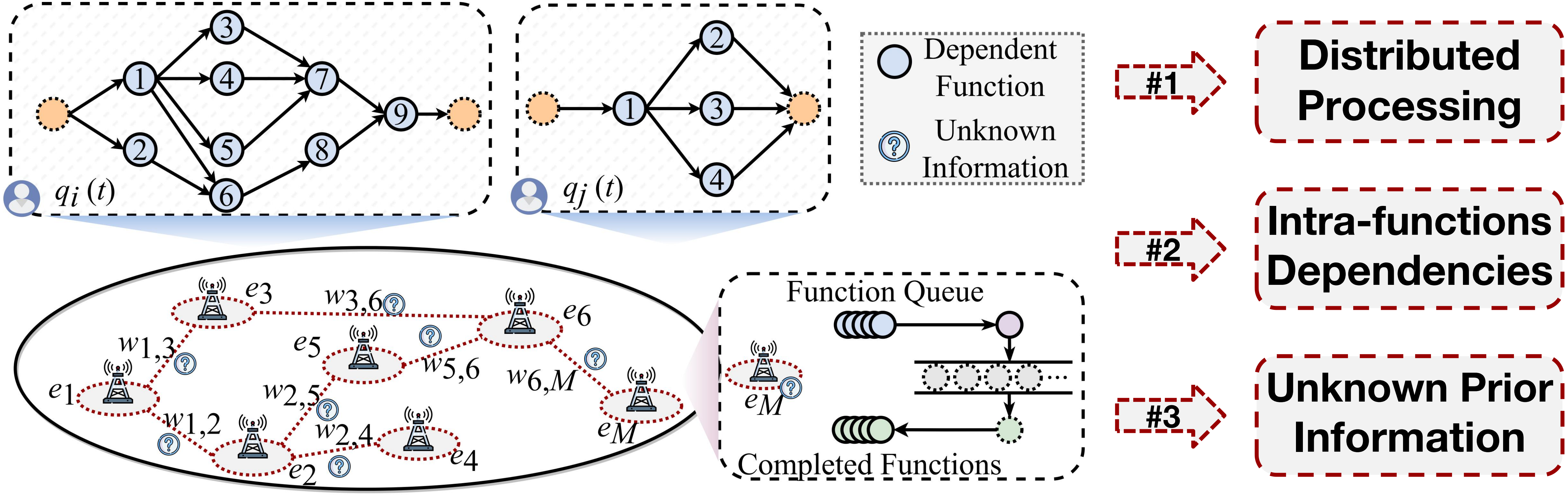
Serverless Edge Computing

- Unleash resources **from cloud to network edge**
- To meet **latency-sensitive** FaaS application demands



- **Low Latency**
- **High Scalablity**
- **Cost-Efficiency**

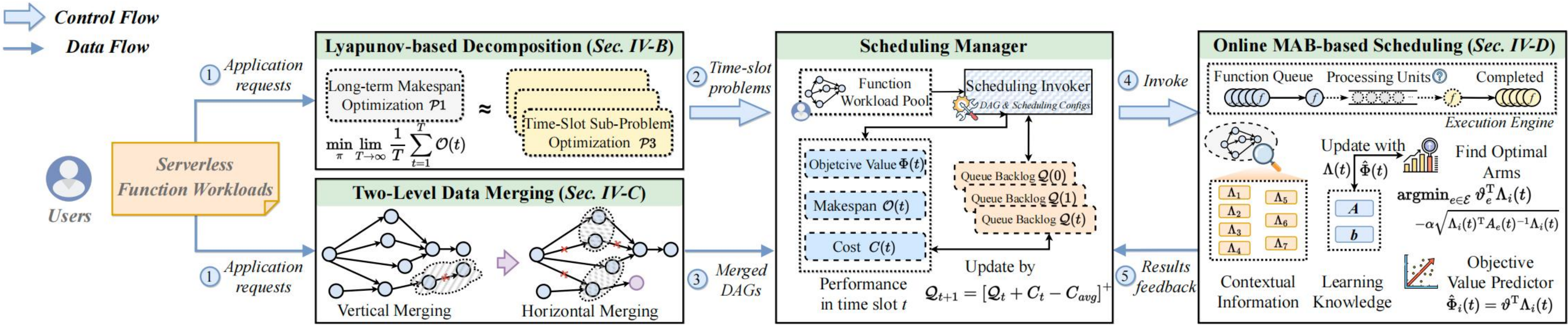
Motivation



How to schedule serverless functions in edge coputing systems with no priors?

Our Solution: SFSM

SFSM Overview

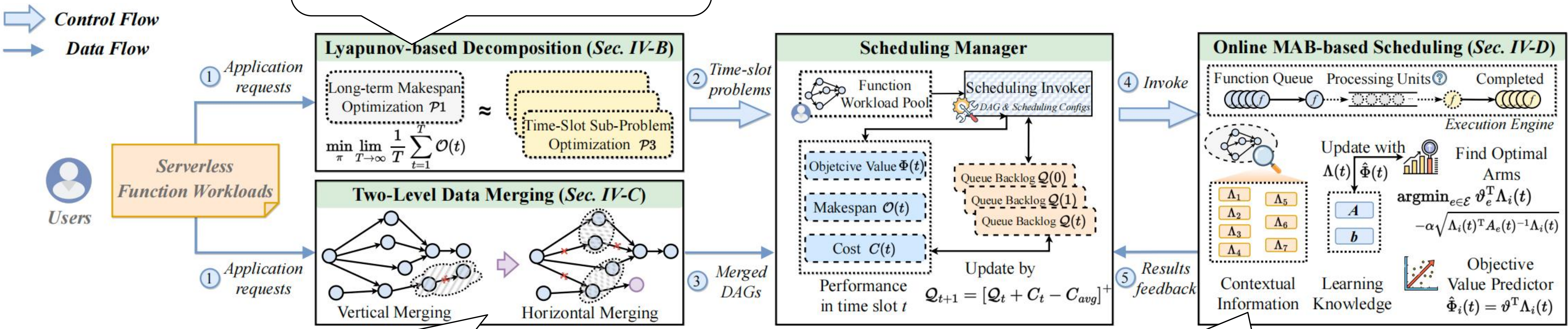


Our Solution: SFSM

(Step 1)

Layapunov-based Decomposition

SFSM Overview



Two-Level Data Merging

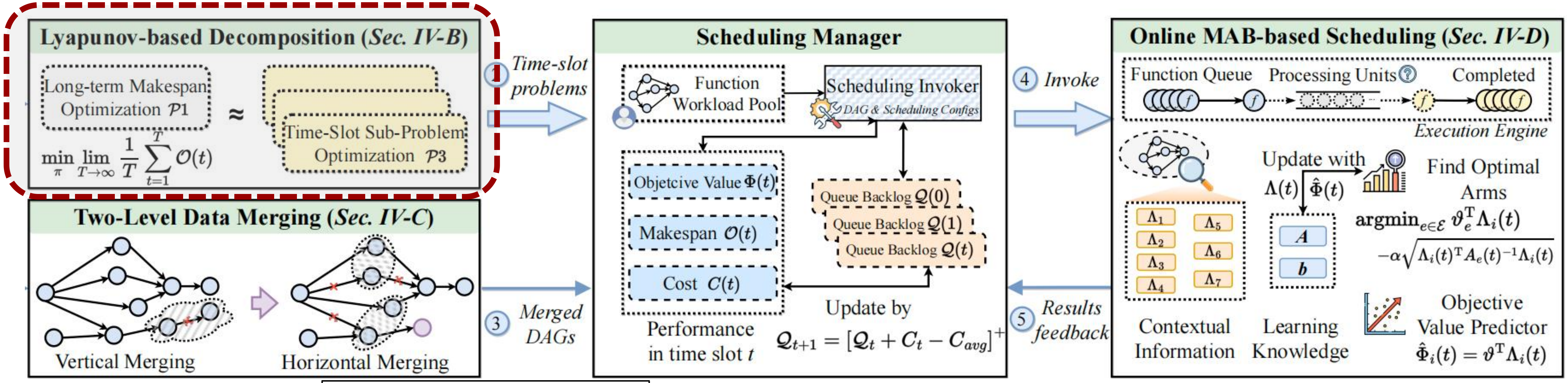
(Step 2)

Online MAB-based Scheduling

(Step 3)

Design of SFSM

Step 1: Layapunov-based Decomposition



$$\mathcal{P}1: \quad \min_{\pi} \lim_{T\to\infty} \frac{1}{T}\sum_{t=1}^{T} \mathcal{O}(t)$$

Long-term averaged makespan

$$\text{s.t.} \quad \mathcal{O}^{es}_{i,k}(t) \ge \max_{j\in\nu(f_i)} \{\mathcal{O}^{com}_{j,k}(t) + \mathcal{O}^{tr}_{i,j,k}(t)\},$$

$$\mathcal{O}^{es}_{i,k}(t) \ge \mathcal{O}^{conf}_{i,k}(t), \forall i \in \mathcal{F}, k \in \mathfrak{A},$$

$$\lim_{T\to\infty} \frac{1}{T}\sum_{t=1}^{T} C(t) \le C_{avg},$$

$$\rho_{\pi_{f_i}(t),k}(t) \le \zeta_{\pi_{f_i}(t)}, \forall i \in \mathcal{F}, k \in \mathfrak{A},$$

$$\sum_{k=1}^{|\mathbf{q}|} \rho_{\pi_{f_i}(t),k}(t) \le \zeta_{\pi_{f_i}(t)}, \forall i \in \mathcal{F}.$$

Lyapunov-drift-plus-penalty

$$\mathcal{P}2: \quad \min_{\pi} \mathbb{E}\left[\Delta(\mathcal{Q}(t)) + V \cdot \mathcal{O}(t) | \mathcal{Q}(t)\right]$$

Approximated by upper bound

$$\mathcal{P}3: \quad \min_{\pi} \Phi(t) = B + V \cdot \mathcal{O}(t) + \mathcal{Q}(t)\left(C(t) - C_{avg}\right)$$

Design of SFSM

Step 1: Layapunov-based Decomposition

Algorithm 1: Long-term problem decomposition by Lyapunov optimization

---

Input: $\mathcal{U}$, $\mathfrak{A}$, C_{avg}, V, $\mathcal{Q}(0)$;
Output: $\pi(t)$ in each time slot $\forall t \in \mathcal{T}$;

for $t = 1$ *to* T **do**
    for $i = 1$ *to* I **do**
        if *Request* i *arrives* **then**
            Obtain the merged $\mathcal{A}_i(t)$ by **Alg.** 2;
            Derive the scheduling decision $\pi_i(t)$ by **Alg.** 3;
            Calculate $\mathcal{O}_i(t)$ and $C_i(t)$;
        end
        Update the virtual cost queue by Eq. (9);
        Set $\pi(t) \leftarrow \pi(t) \cup \pi_i(t)$;
    end
end
Return $\{\pi(1), \pi(2), \cdots, \pi(T)\}$.

Call Alg. 2 (data merging)
&
Alg. 3 (online scheduling)

Design of SFSM

Step 1: Layapunov-based Decomposition

Algorithm 1: Long-term problem decomposition by Lyapunov optimization

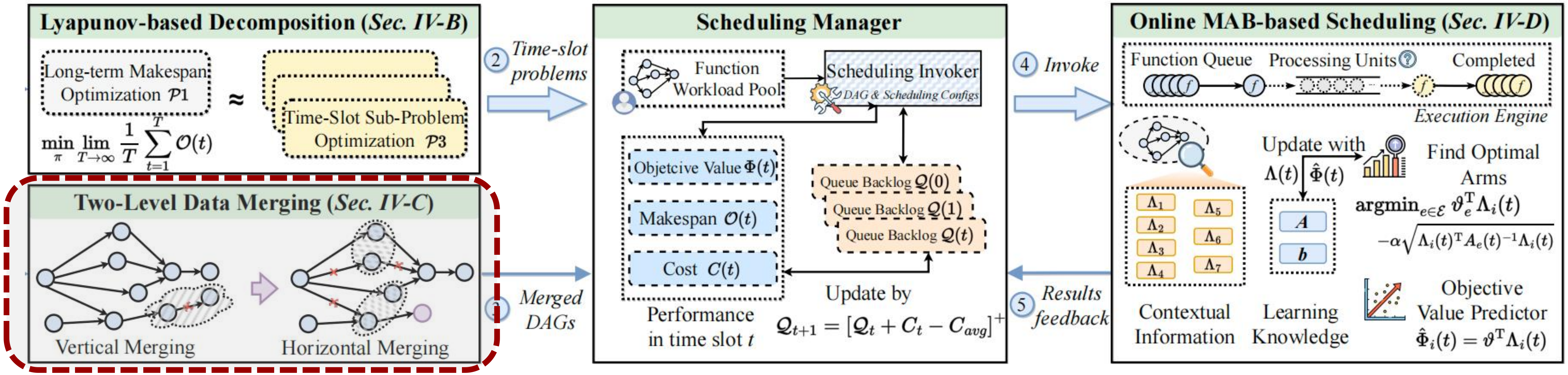
Input: $\mathcal{U}$, $\mathfrak{A}$, C_{avg}, V, $\mathcal{Q}(0)$;
Output: $\pi(t)$ in each time slot $\forall t \in \mathcal{T}$;

for $t = 1$ *to* T **do**
  for $i = 1$ *to* I **do**
    if *Request* i *arrives* **then**
      Obtain the merged $\mathcal{A}_i(t)$ by **Alg.** 2;
      Derive the scheduling decision $\pi_i(t)$ by **Alg.** 3;
      Calculate $\mathcal{O}_i(t)$ and $C_i(t)$;
    end
    Update the virtual cost queue by Eq. (9);
    Set $\pi(t) \leftarrow \pi(t) \cup \pi_i(t)$;
  end
end
Return $\{\pi(1), \pi(2), \cdots, \pi(T)\}$.

Update virtual cost queue based on Lyapunov optimization technique

Design of SFSM

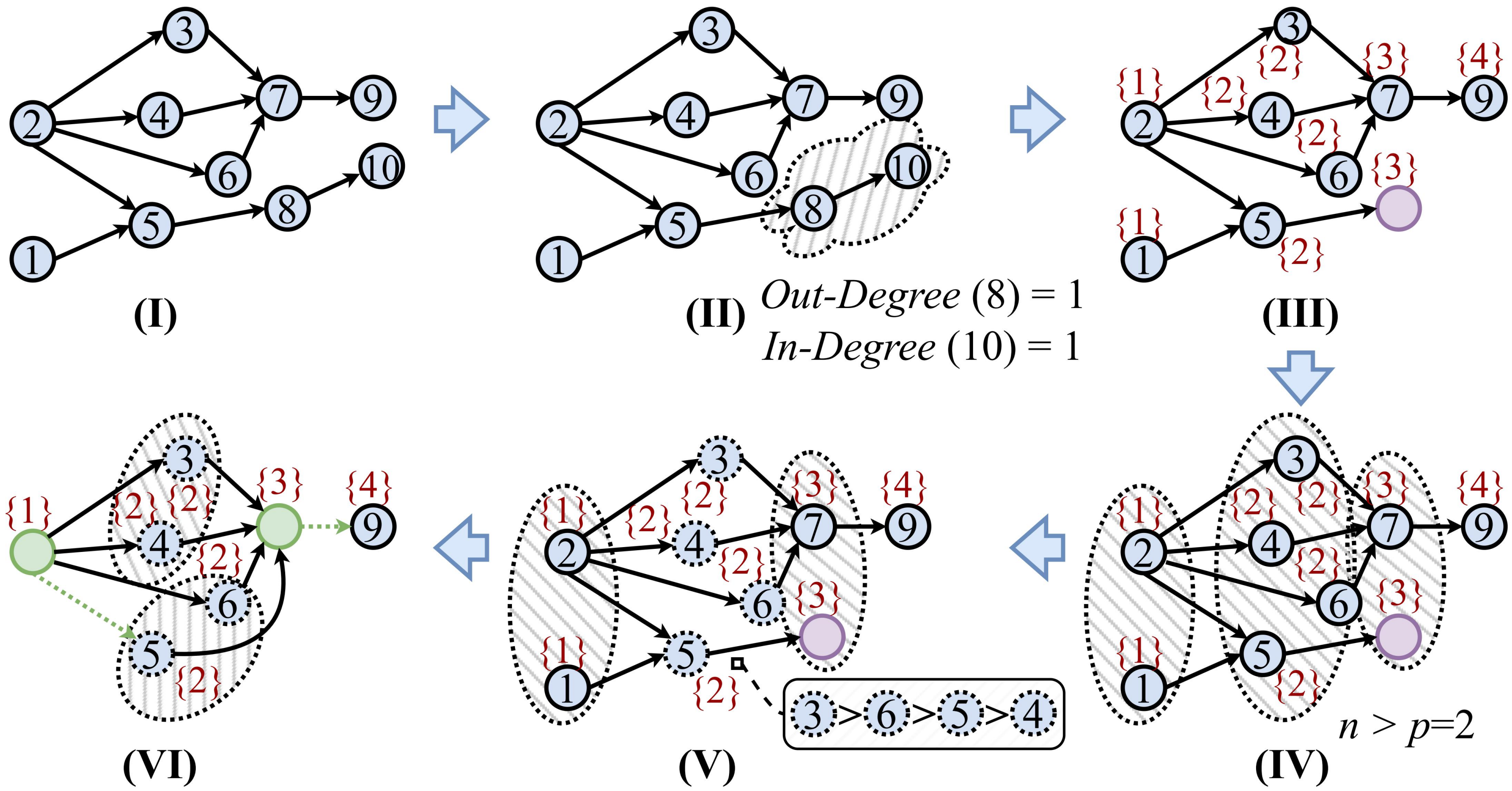
Step 2: Two-Level Data Merging



- **Vertical Merging:** find the node that only has one successor and its successor also has one predecessor.
- **Horizontal Merging:** integrate nodes at the same node levels to improve the parallel performance.

Design of SFSM

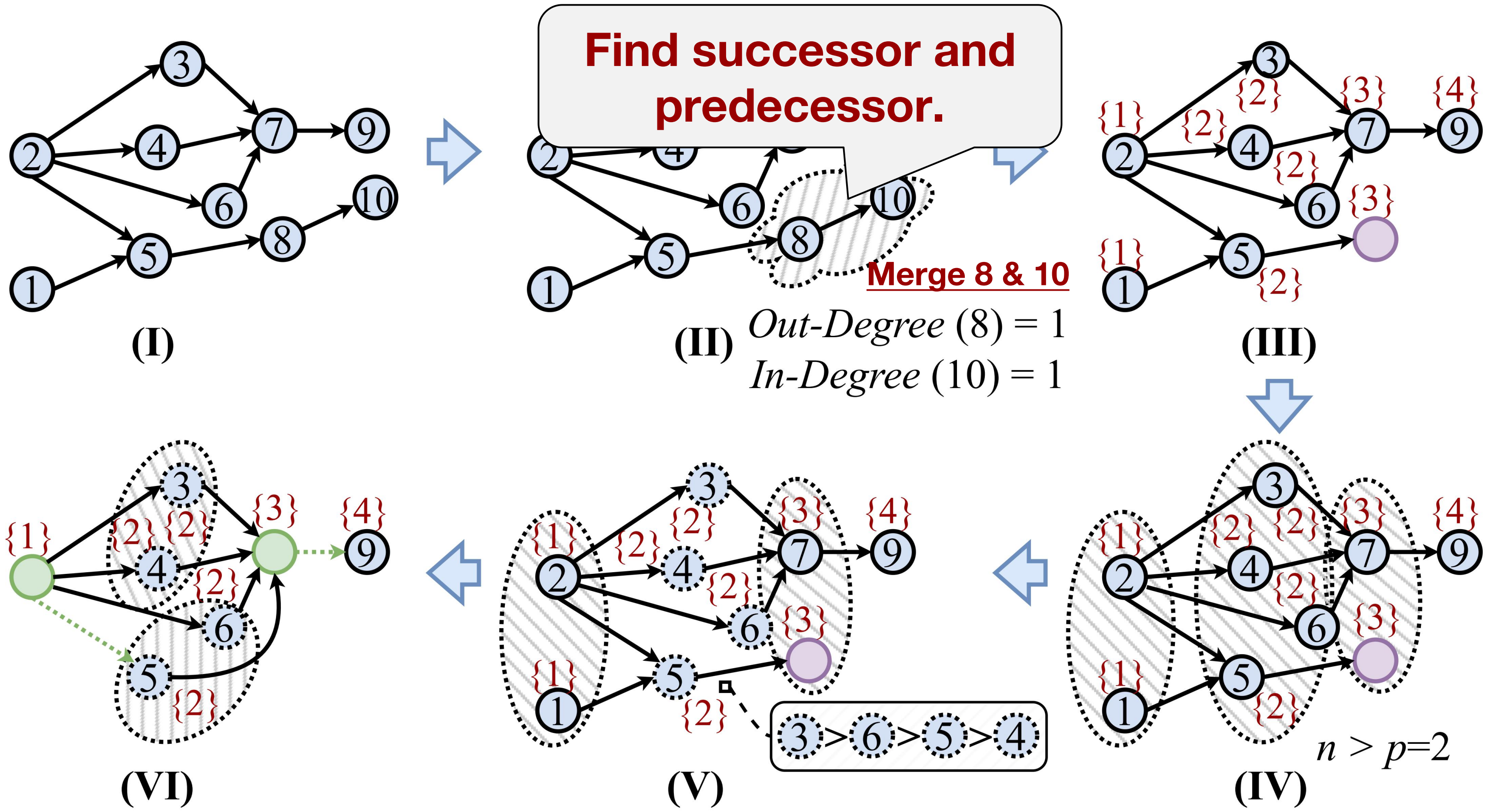
Step 2: Two-Level Data Merging



An example of 10 serverless functions

Design of SFSM

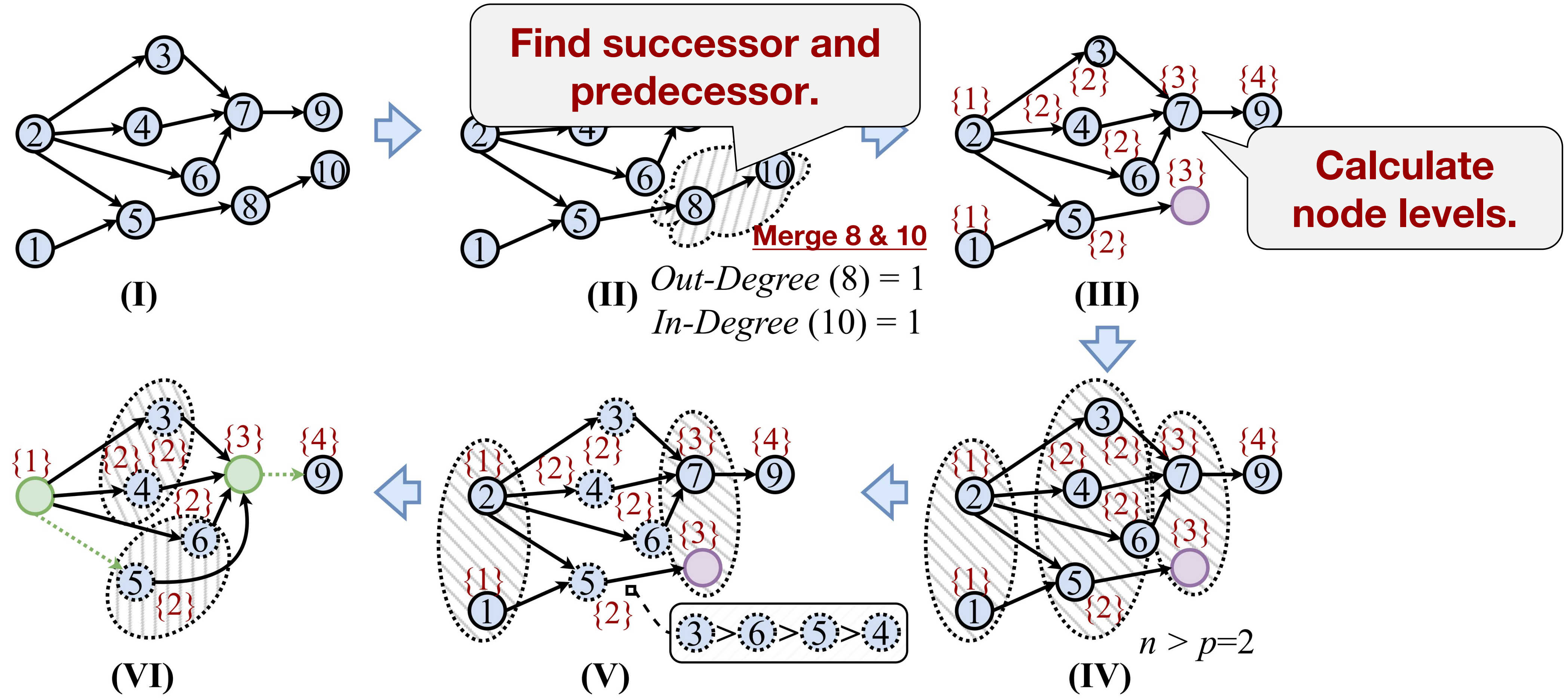
Step 2: Two-Level Data Merging



An example of 10 serverless functions

Design of SFSM

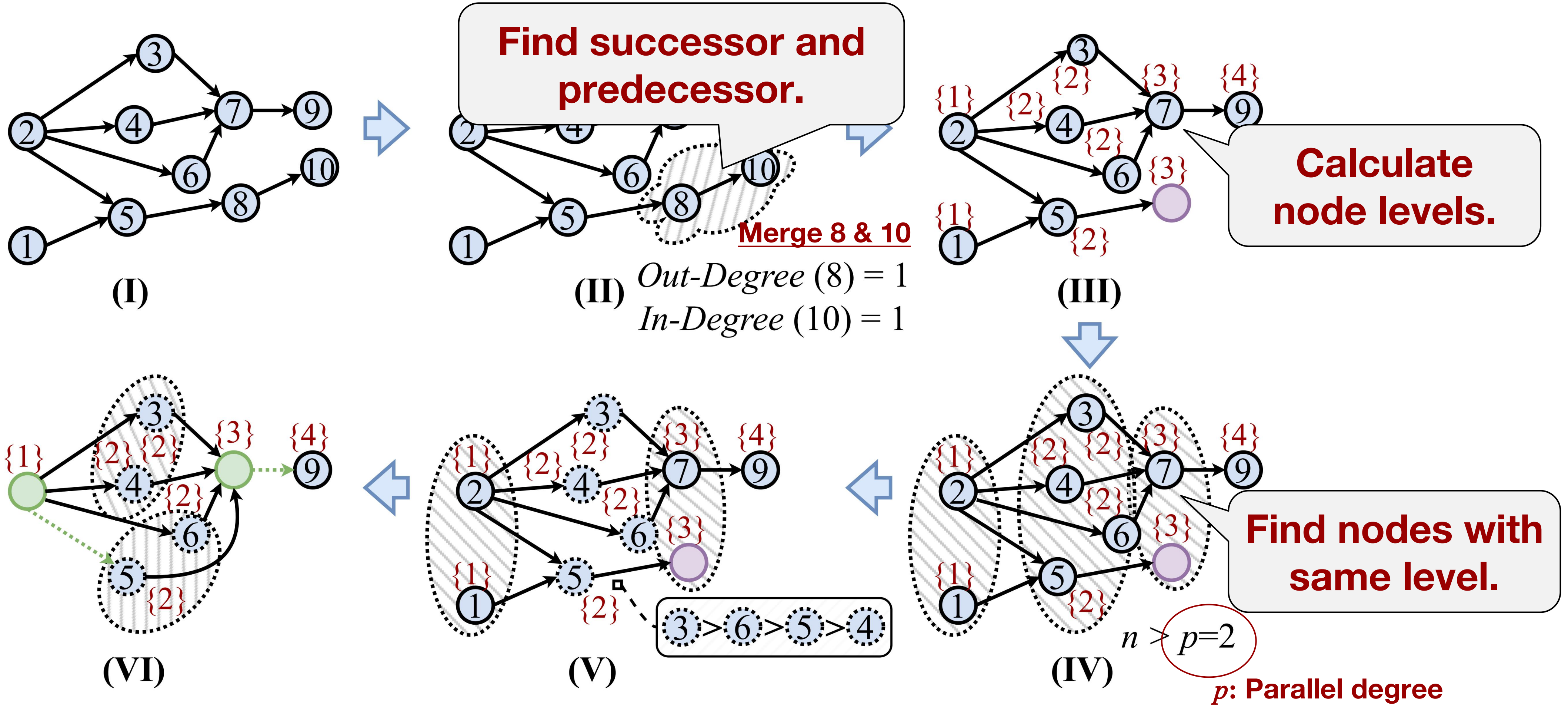
Step 2: Two-Level Data Merging



An example of 10 serverless functions

Design of SFSM

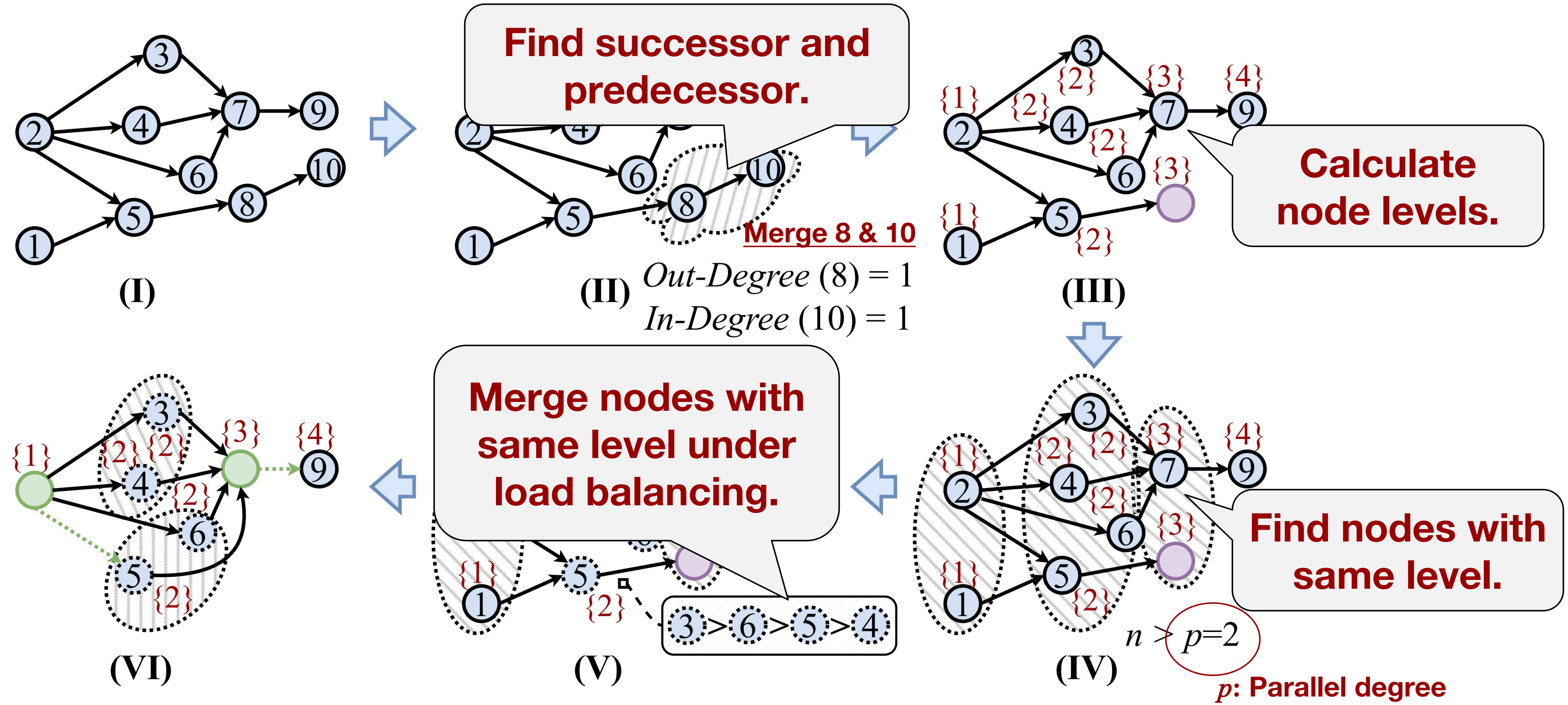
Step 2: Two-Level Data Merging



An example of 10 serverless functions

Design of SFSM

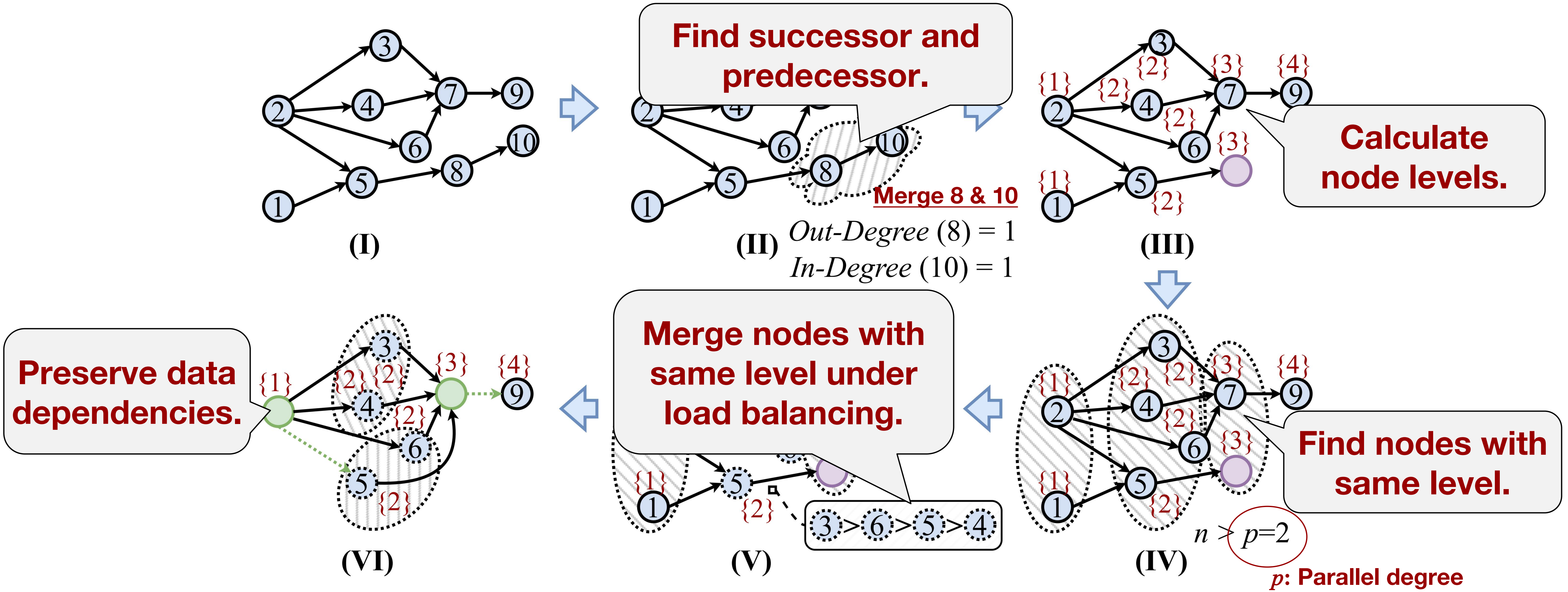
Step 2: Two-Level Data Merging



An example of 10 serverless functions

Design of SFSM

Step 2: Two-Level Data Merging



An example of 10 serverless functions

Design of SFSM

Step 2: Two-Level Data Merging

Algorithm 2: Two-level data merging

Input: $\mathcal{A}$, p, L;
Output: Merged DAG $\mathcal{A}'$;

for f *in topological sort of* $\mathcal{A}$ **do**
  if *out-degree*(f) *is* 1 **then**
    $g \leftarrow suc(f)$;
    if *in-degree*(g) *is* 1 *and pre*(g) *is* f **then**
      $\phi_f \leftarrow \phi_f + \phi_g$;
      $suc(f) \leftarrow suc(g)$;
      $del(g)$;
    end
  end
end

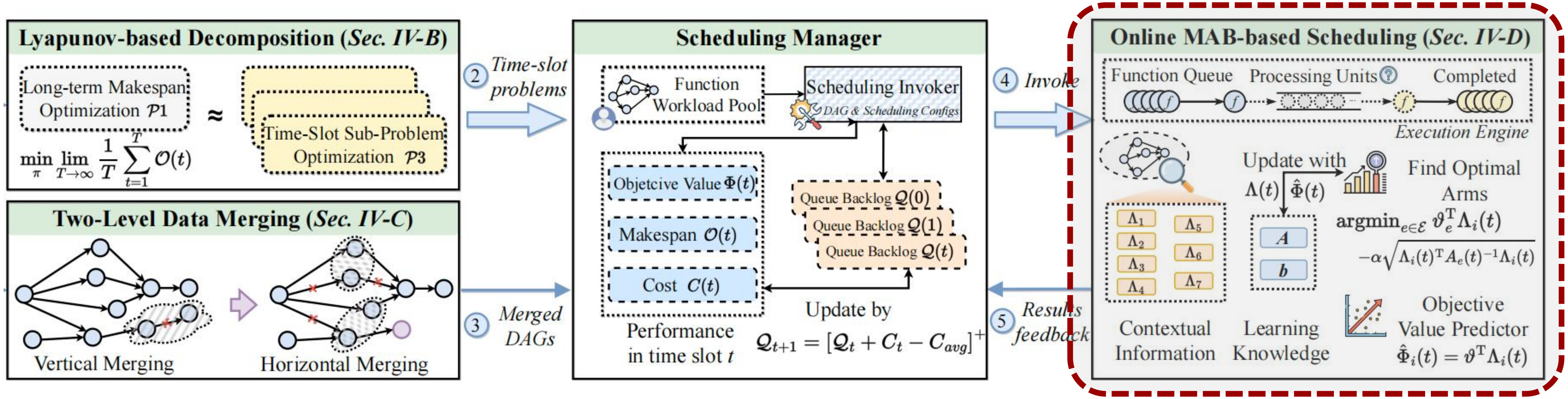
Vertical Merging

for $l = 1$ *to* L **do**
  for f *in group* l **do**
    $\mathbf{w} \leftarrow \mathbf{w} \cup \phi_f$;
  end
  Sort $\mathbf{w}$ in descending order;
  $i \leftarrow 1$, $j \leftarrow |\mathbf{w}|$;
  while $i < j$ **do**
    $n \leftarrow min(p, j - i + 1)$;
    $merge(\mathcal{A}', n)$;
    $\phi_{f1} \leftarrow \{\phi_{f1}, \phi_{f2}, \cdots, \phi_{fn}\}$;
    $suc(f_i) \leftarrow \{suc(f_i), suc(f_j), \cdots, suc(f_{i+\lceil \frac{n}{2} \rceil}), suc(f_{j-\lfloor \frac{n}{2} \rfloor})\}$;
    $pre(f_i) \leftarrow \{pre(f_i), pre(f_j), \cdots, pre(f_{i+\lceil \frac{n}{2} \rceil}), pre(f_{j-\lfloor \frac{n}{2} \rfloor})\}$;
    $del(\mathcal{A}', n)$;
    $i \leftarrow i + \lceil \frac{p}{2} \rceil$, $j \leftarrow j - \lfloor \frac{p}{2} \rfloor$;
  end
end
Return $\mathcal{A}'$.

Horizontal Merging

Design of SFSM

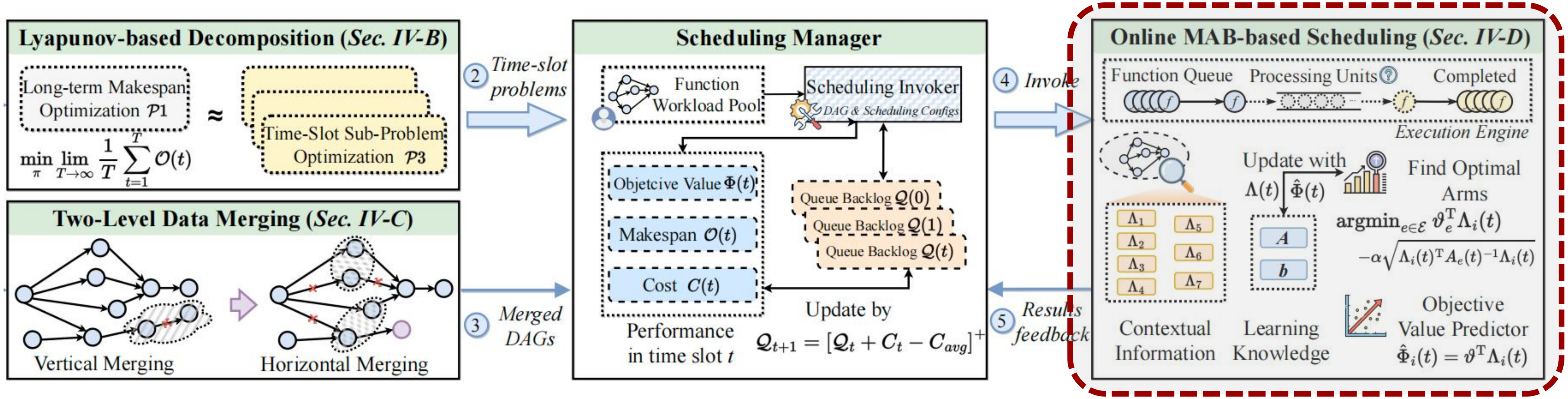
Step 3: Online MAB-based Scheduling



- **Contextual information:** request info, computations, data, etc.
- **Without any priors of edge networks:** i.e., transmission link and computing resources

Design of SFSM

Step 3: Online MAB-based Scheduling



$$\pi_i(t) = \text{argmin}\ \vartheta^{\mathrm{T}} \Lambda_i(t)$$

ridge regression

$$\pi_i(t) = \text{argmin}_{e\in\mathcal{E}} \left[\vartheta_e^{\mathrm{T}} \Lambda_i(t) - \alpha \sqrt{\Lambda_i(t)^{\mathrm{T}} A_e(t)^{-1} \Lambda_i(t)} \right]$$

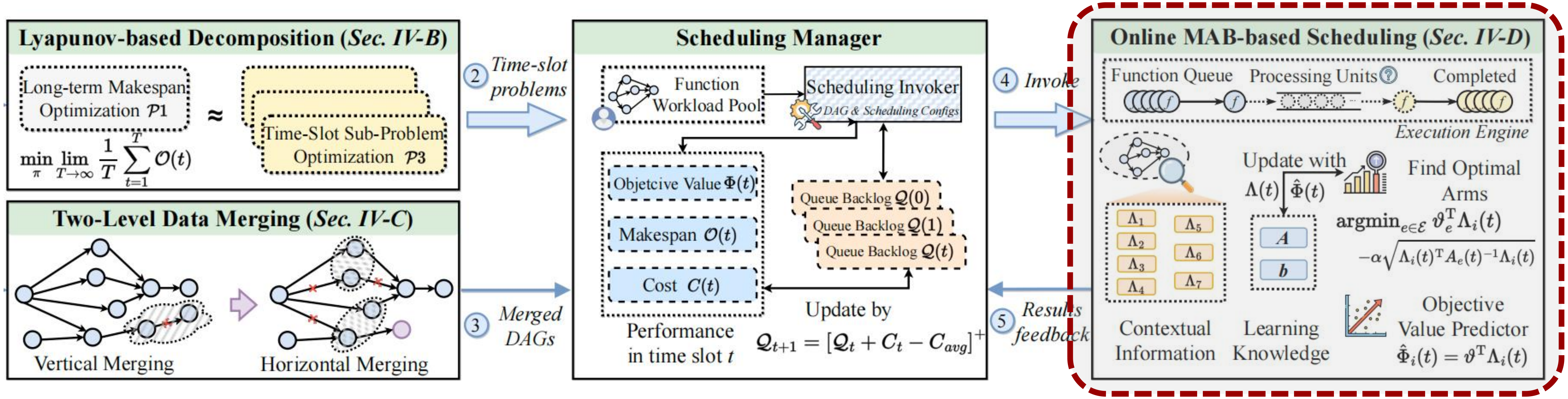
learnable weight

matrix of contextual information

balance exploration and exploitation

Design of SFSM

Step 3: Online MAB-based Scheduling



$$\pi_i(t) = \mathrm{argmin}\, \vartheta^{\mathrm{T}}\Lambda_i(t)$$

ridge regression

$$\pi_i(t) = \mathrm{argmin}_{e\in\mathcal{E}}\left[\vartheta_e^{\mathrm{T}}\Lambda_i(t) - \boxed{\alpha\sqrt{\Lambda_i(t)^{\mathrm{T}}A_e(t)^{-1}\Lambda_i(t)}}\right]$$

exploration

learnable weight

matrix of contextual information

balance exploration and exploitation

Design of SFSM

Step 3: Online MAB-based Scheduling

Algorithm 3: Online serverless function scheduling

Input: $\mathcal{A}'_i$, $\mathcal{U}$;
Output: π_i;

Construct the contextual knowledge $\Lambda_i(t)$;
Initialize A_e and b_e, $\forall e \in \mathcal{E}$;
repeat
  for *f in topological sort of $\mathcal{A}'_i$* **do**
    for $e = 1$ *to* $|\mathcal{E}|$ **do**
      $\hat{\vartheta}_e(h) \leftarrow A_e^{-1}(h-1)b_e(h-1)$;
      $\hat{\Phi}_{f,e}(h) \leftarrow \hat{\vartheta}_e(h)^{\mathrm{T}}\Lambda_i(h) - \alpha\sqrt{\Lambda_i(h)^{\mathrm{T}}A_e(h)^{-1}\Lambda_i(h)}$;
    end
    $\pi_{i,f}(h) \leftarrow \operatorname{argmin}_{e\in\mathcal{E}}\hat{\Phi}_{f,e}(h)$;
    $\pi_i(h) \leftarrow \pi_i(h) \cup \pi_{i,f}(h)$;
  end
  for *Activated* $e \in \mathcal{E}$ **do**
    $A_e(h) \leftarrow A_e(h-1) + \Lambda_i(h)\Lambda_i(h)^{\mathrm{T}}$;
    $b_e(h) \leftarrow b_e(h-1) + \Lambda_i(h)\hat{\Phi}_{f,e}(h)$;
  end
  $\pi_i \leftarrow \pi_i \cup \pi_i(h)$;
until *Learning step h finished*;
Return π_i.

Calculate weight and predict objective value

Design of SFSM

Step 3: Online MAB-based Scheduling

Algorithm 3: Online serverless function scheduling

Input: $\mathcal{A}'_i$, $\mathcal{U}$;
Output: π_i;
Construct the contextual knowledge $\Lambda_i(t)$;
Initialize A_e and b_e, $\forall e \in \mathcal{E}$;
repeat
  for f *in topological sort of* $\mathcal{A}'_i$ **do**
    for $e = 1$ *to* $|\mathcal{E}|$ **do**
      $\hat{\vartheta}_e(h) \leftarrow A_e^{-1}(h-1)b_e(h-1)$;
      $\hat{\Phi}_{f,e}(h) \leftarrow$
      $\hat{\vartheta}_e(h)^{\mathrm{T}}\Lambda_i(h) - \alpha\sqrt{\Lambda_i(h)^{\mathrm{T}}A_e(h)^{-1}\Lambda_i(h)}$;
    end
    $\pi_{i,f}(h) \leftarrow \operatorname{argmin}_{e\in\mathcal{E}}\hat{\Phi}_{f,e}(h)$;
    $\pi_i(h) \leftarrow \pi_i(h) \cup \pi_{i,f}(h)$;
  end
  for *Activated* $e \in \mathcal{E}$ **do**
    $A_e(h) \leftarrow A_e(h-1) + \Lambda_i(h)\Lambda_i(h)^{\mathrm{T}}$;
    $b_e(h) \leftarrow b_e(h-1) + \Lambda_i(h)\hat{\Phi}_{f,e}(h)$;
  end
  $\pi_i \leftarrow \pi_i \cup \pi_i(h)$;
until *Learning step h finished*;
Return π_i.

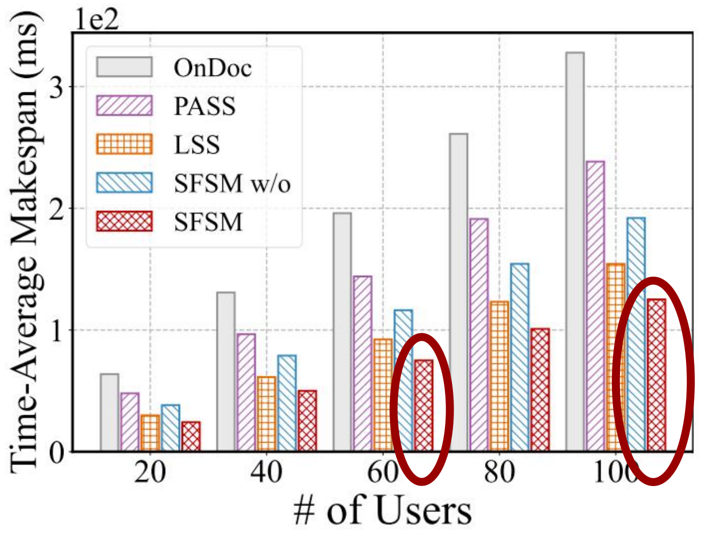
Select the target scheduling edge server with minimum value $\hat{\Phi}_i(t)$

Evaluation Settings

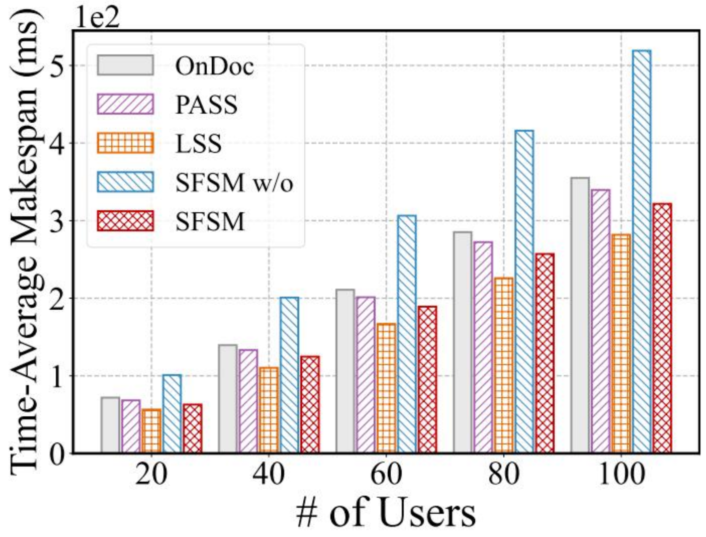
- **Workloads**
  - Alibaba cluster trace[1], which includes about 4000 machines in a periods of 8 days, over 20,000 applications
  - Three request modes, i.e., data-intensive, computation-intensive and both data and computation-intensive
- **Baselines**
  - OnDoc [TOSN 2023]
  - PASS [IWQoS 2022]
  - Latency-myopic Static Scheduling (LSS) [TPDS 2022, IWQoS 2019]
  - SFSM w/o
- **Metircs**
  - Time-Average Makespan
  - Time-Average Cost

[1] https://github.com/alibaba/clusterdata

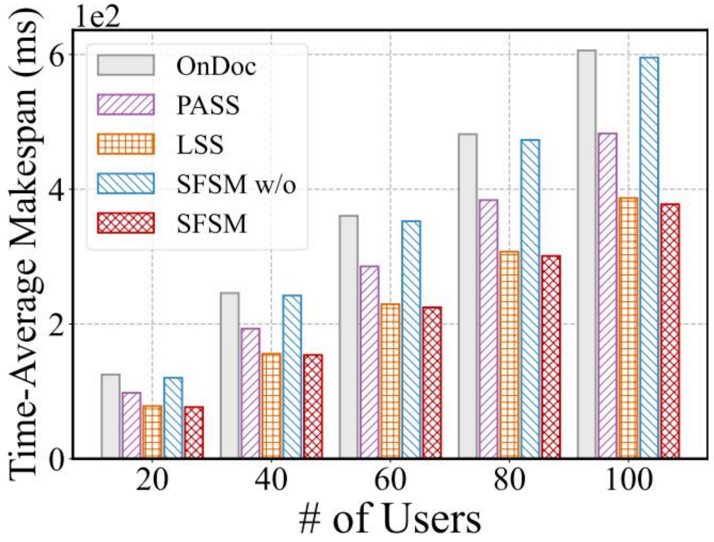
Evaluation: Makespan



(a) data-intensive



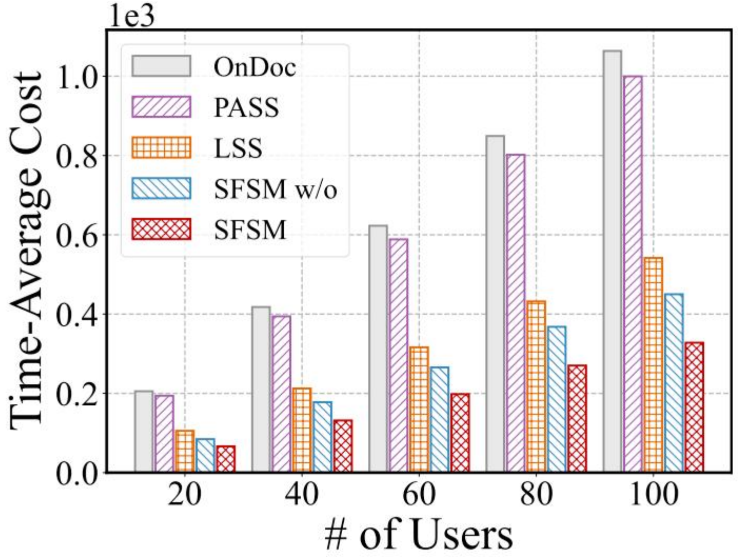
(b) computation-intensive



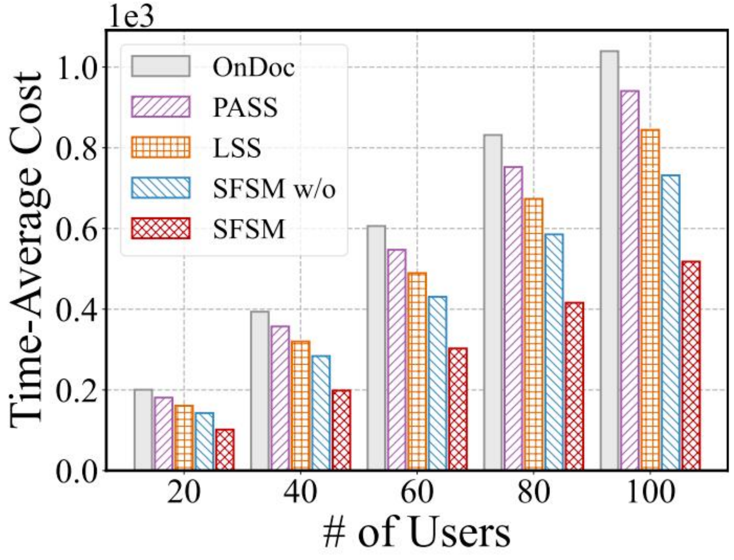
(c) both data and computation-intensive

Lower makespan, especially with intensive data req.

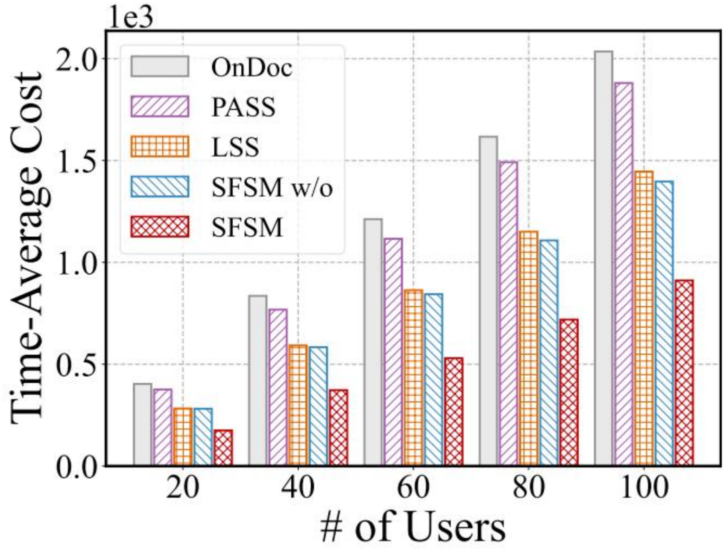
Evaluation: Cost



(a) data-intensive



(b) computation-intensive



(c) both data and computation-intensive

Lower cost within different request modes

Conclusion

- SFSM aims to achieve the **long-term makespan** minimization while improving the execution cost
- Two-level data merging algorithm to **mitigate the transmission overhead** of redundant data
- **Online** contexual shceduling **without priors** to realize fine-grained decisions

Thank You For Your Attention!