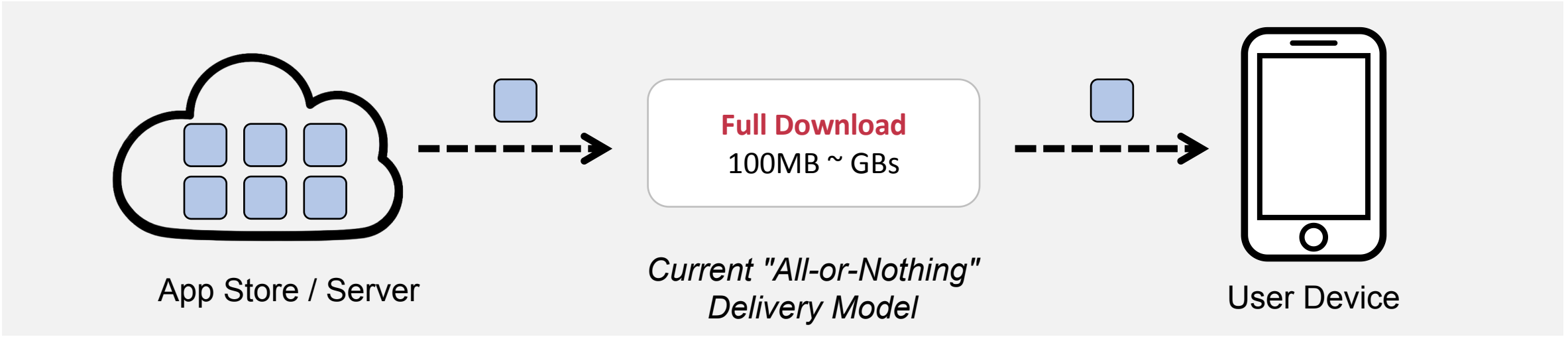
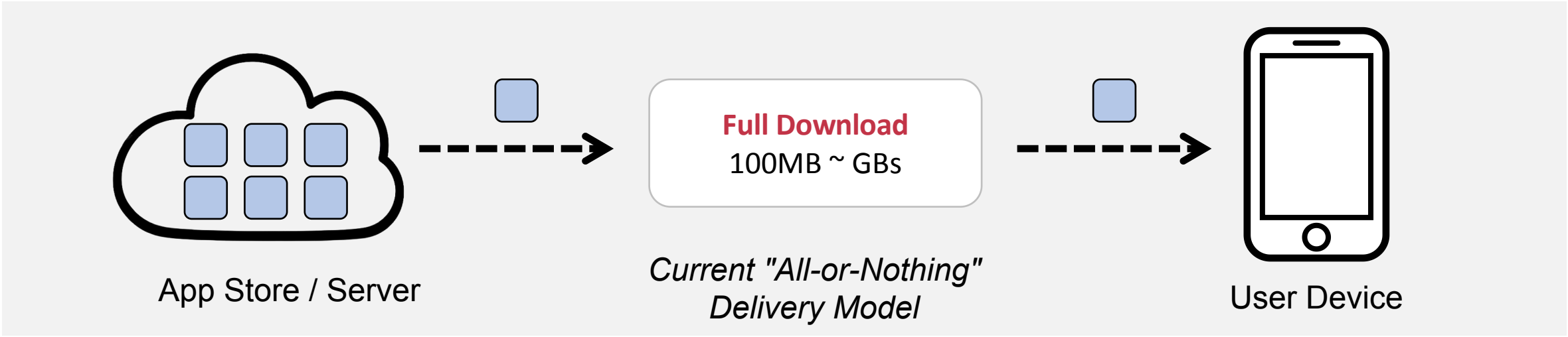
Bridging Storage and Execution: A Semantic Virtual Bus for On-Demand Application Streaming

Jun Lu, Jialin Li, Yaoxue Zhang, Ju Ren

What's wrong with today's app delivery?



What's wrong with today's app delivery?



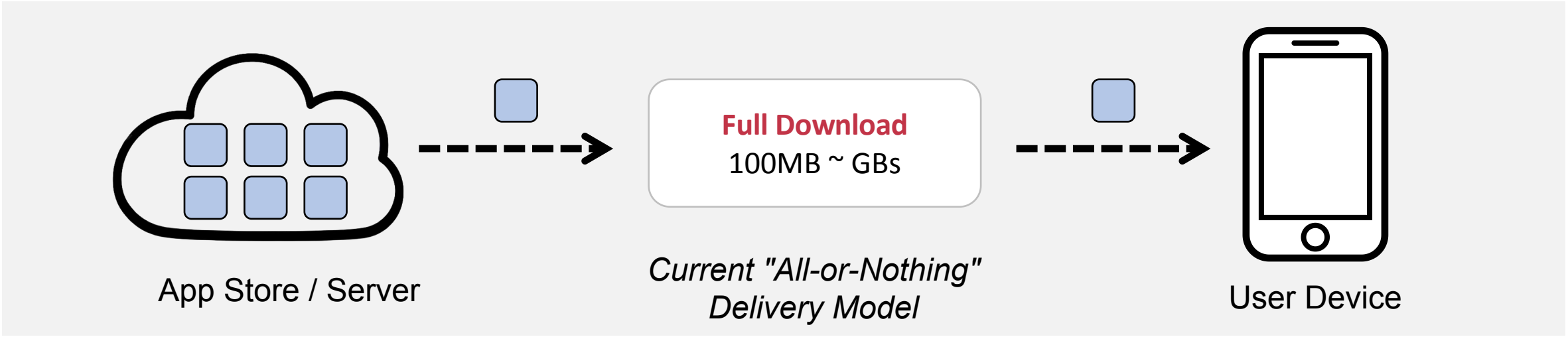
Users delay or avoid updates

- **Persistent vulnerabilities**
- **Wasted developer effort**

Application sizes keep growing

- **Long download & installation time**
- **Poor experience for short-term usage**

What's wrong with today's app delivery?



Users delay or avoid updates

- **Persistent vulnerabilities**
- **Wasted developer effort**

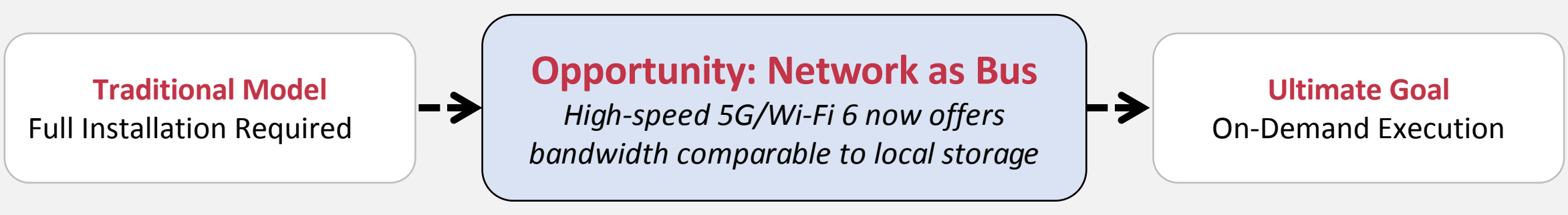
Application sizes keep growing

- **Long download & installation time**
- **Poor experience for short-term usage**

Both problems stem from the same root cause: **full installation before use**

Why existing solutions fall short?

Emerging Paradigm Shift vs. Traditional Model



Why existing solutions fall short?

Emerging Paradigm Shift vs. Traditional Model

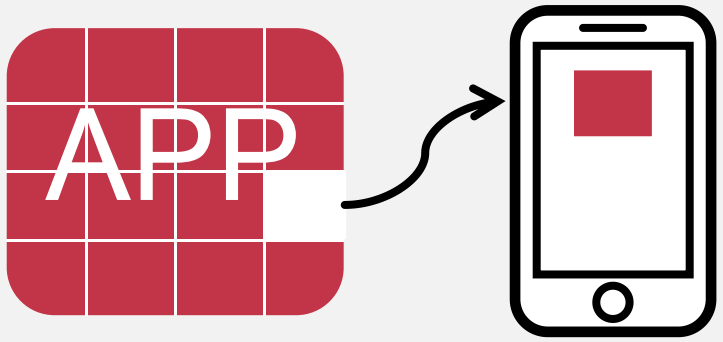
Traditional Model
Full Installation Required

→

Opportunity: Network as Bus
High-speed 5G/Wi-Fi 6 now offers bandwidth comparable to local storage

→

Ultimate Goal
On-Demand Execution



Framework-Bound

Intrusive → cannot support existing apps

Why existing solutions fall short?

Emerging Paradigm Shift vs. Traditional Model

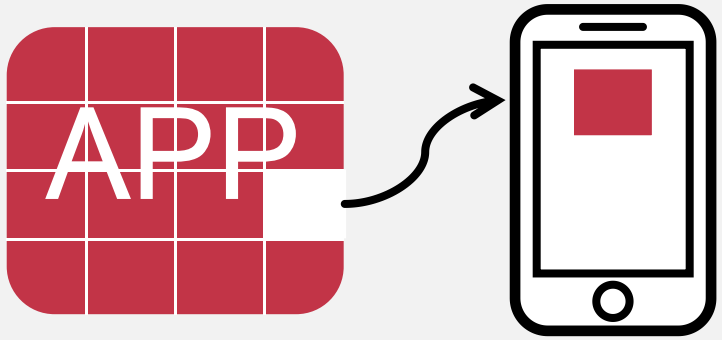
Traditional Model
Full Installation Required

→

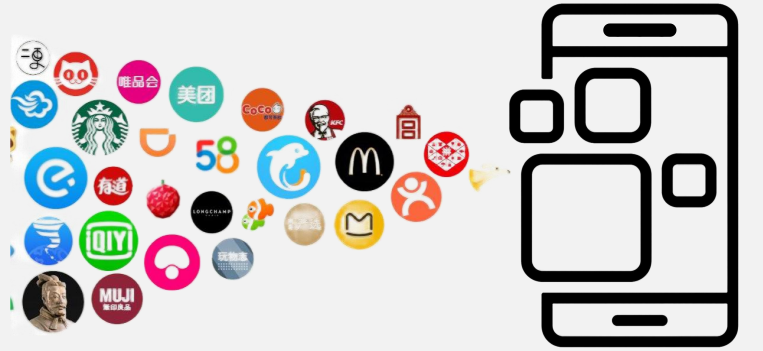
Opportunity: Network as Bus
High-speed 5G/Wi-Fi 6 now offers bandwidth comparable to local storage

→

Ultimate Goal
On-Demand Execution



Framework-Bound
Intrusive → cannot support existing apps



Web/Mini-Programs
Non-native runtime → performance loss

Why existing solutions fall short?

Emerging Paradigm Shift vs. Traditional Model

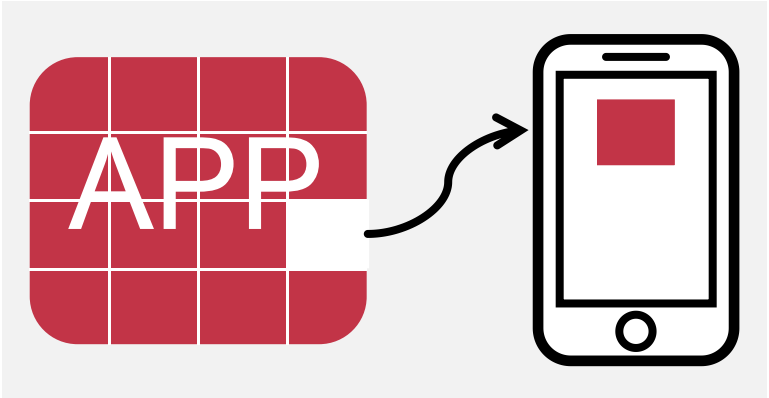
Traditional Model
Full Installation Required

→

Opportunity: Network as Bus
High-speed 5G/Wi-Fi 6 now offers bandwidth comparable to local storage

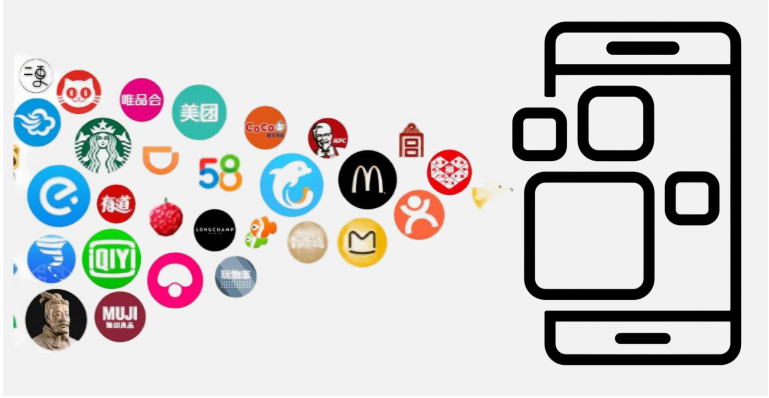
→

Ultimate Goal
On-Demand Execution



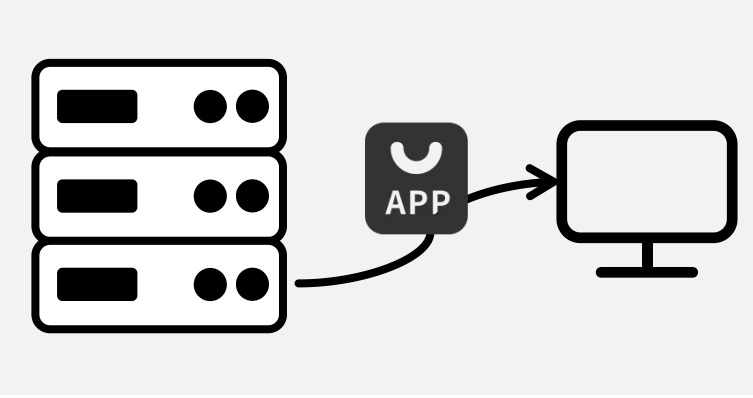
Framework-Bound

Intrusive → cannot support existing apps



Web/Mini-Programs

Non-native runtime → performance loss



Network FS (NFS)

Stateless for execution → design for data access

Why existing solutions fall short?

Emerging Paradigm Shift vs. Traditional Model

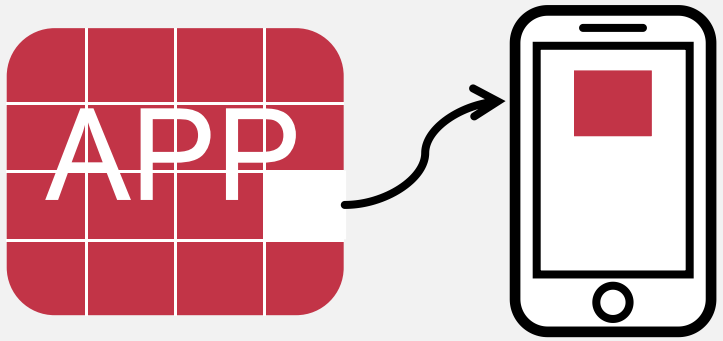
Traditional Model
Full Installation Required

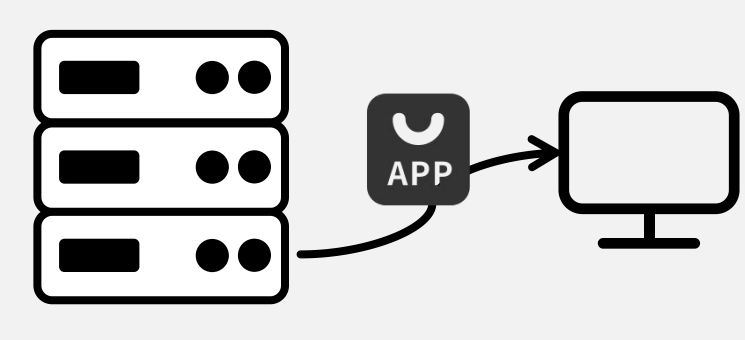
→

Opportunity: Network as Bus
High-speed 5G/Wi-Fi 6 now offers bandwidth comparable to local storage

→

Ultimate Goal
On-Demand Execution





Framework-Bound
Intrusive → cannot support existing apps

Web/Mini-Programs
Non-native runtime → performance loss

Network FS (NFS)
Stateless for execution → design for data access

Existing approaches fail to balance transparency, efficiency, and compatibility.

Key Idea: Semantic Virtual Storage Bus

Goal 1: Transparency

No installation, seamless execution

Goal 2: Efficiency

Hide latency over variable networks

Goal 3: Compatibility

Run unmodified applications

Key Idea: Semantic Virtual Storage Bus

Goal 1: Transparency

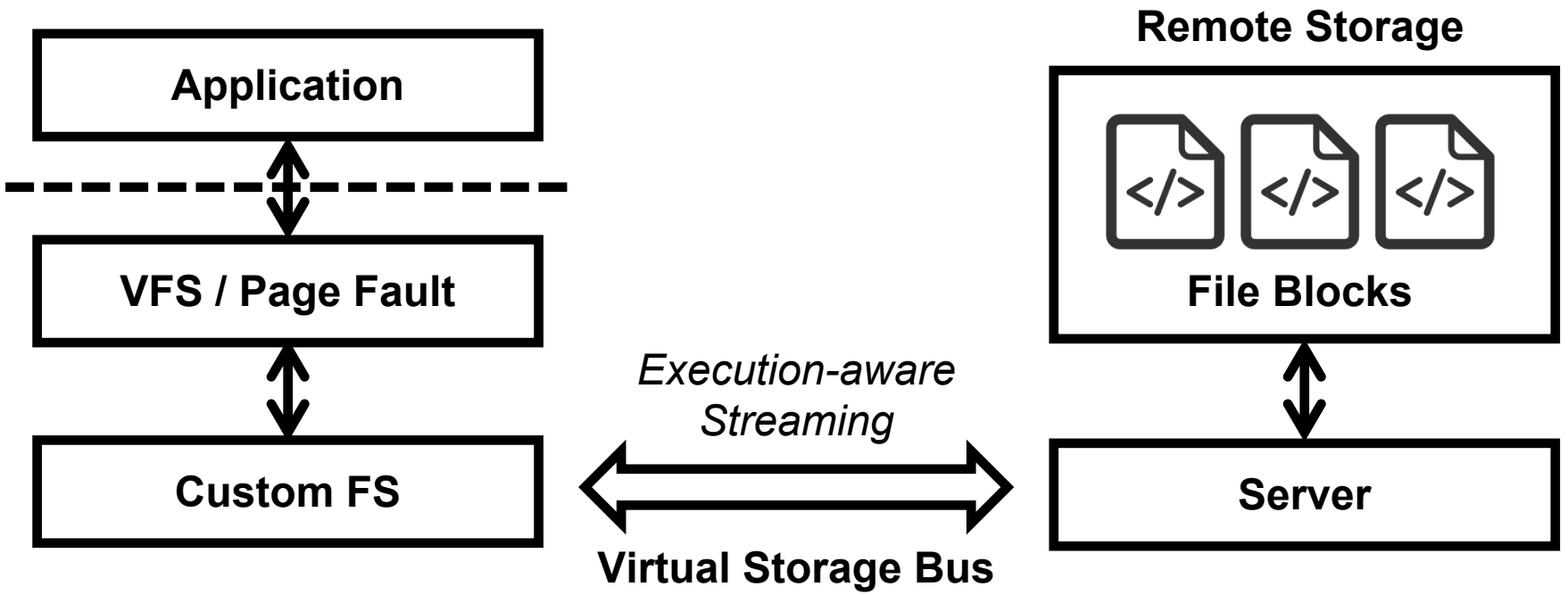
No installation, seamless execution

Goal 2: Efficiency

Hide latency over variable networks

Goal 3: Compatibility

Run unmodified applications



Key Idea: Semantic Virtual Storage Bus

Goal 1: Transparency

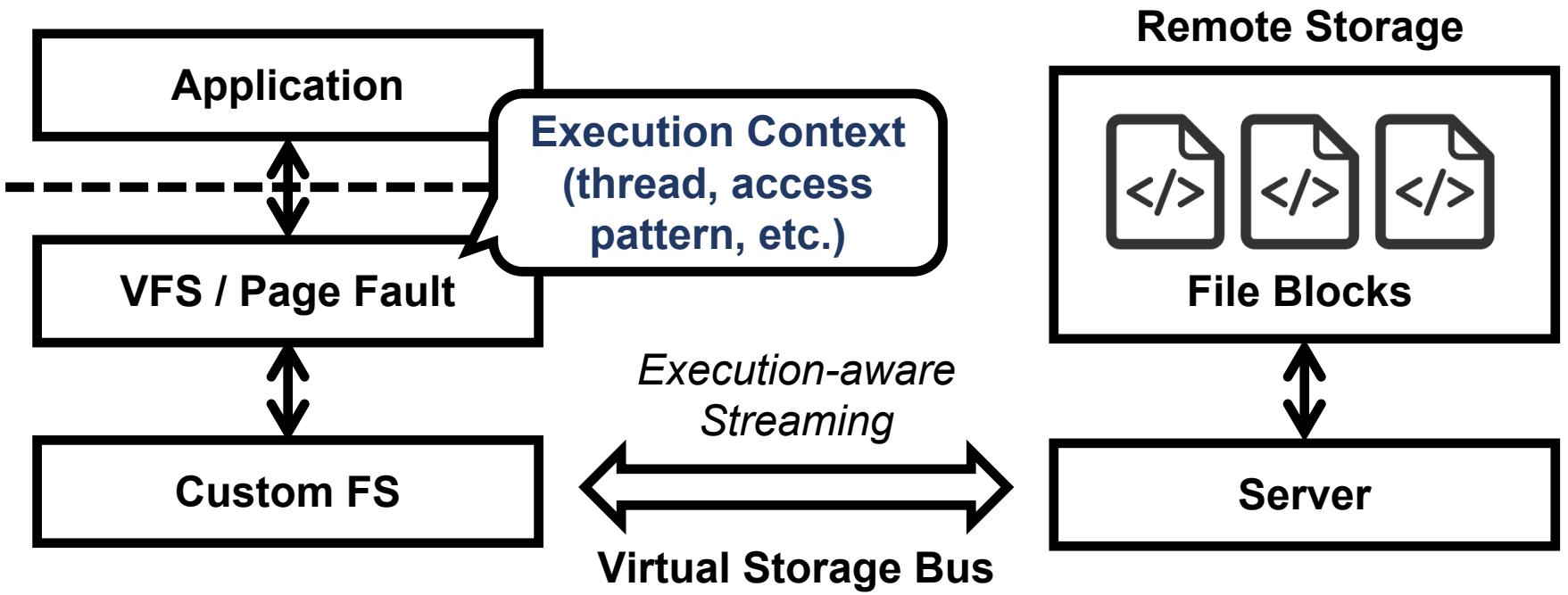
No installation, seamless execution

Goal 2: Efficiency

Hide latency over variable networks

Goal 3: Compatibility

Run unmodified applications



Key Idea: Semantic Virtual Storage Bus

Goal 1: Transparency

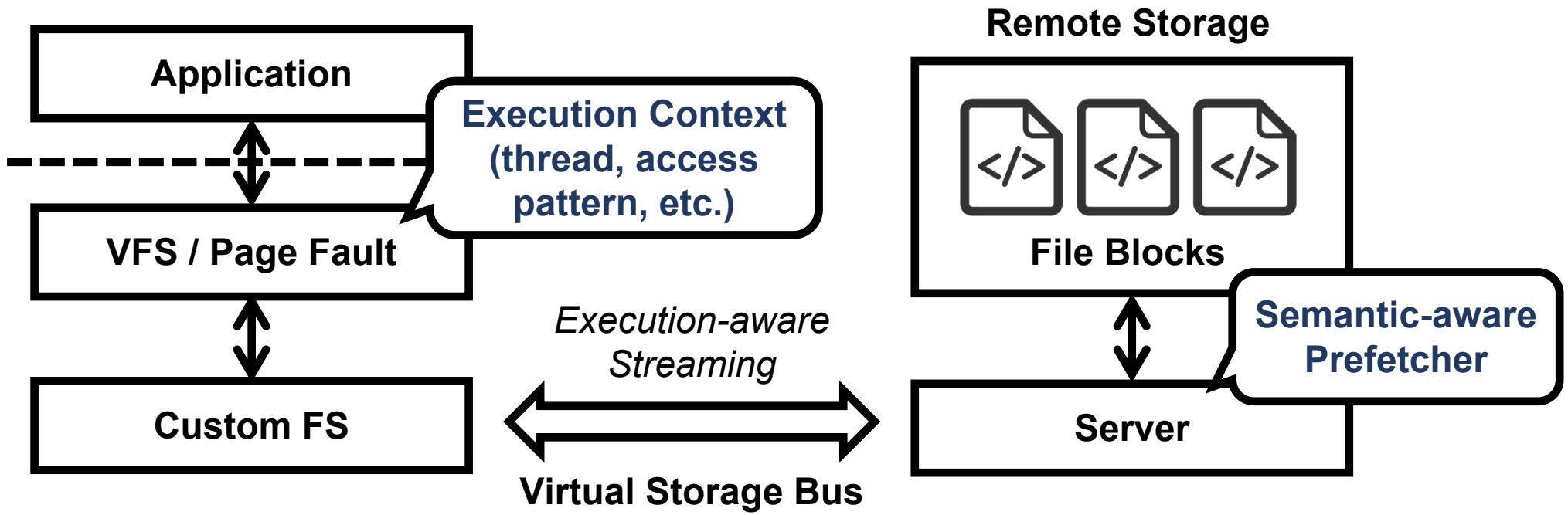
No installation, seamless execution

Goal 2: Efficiency

Hide latency over variable networks

Goal 3: Compatibility

Run unmodified applications



Key Idea: Semantic Virtual Storage Bus

Goal 1: Transparency

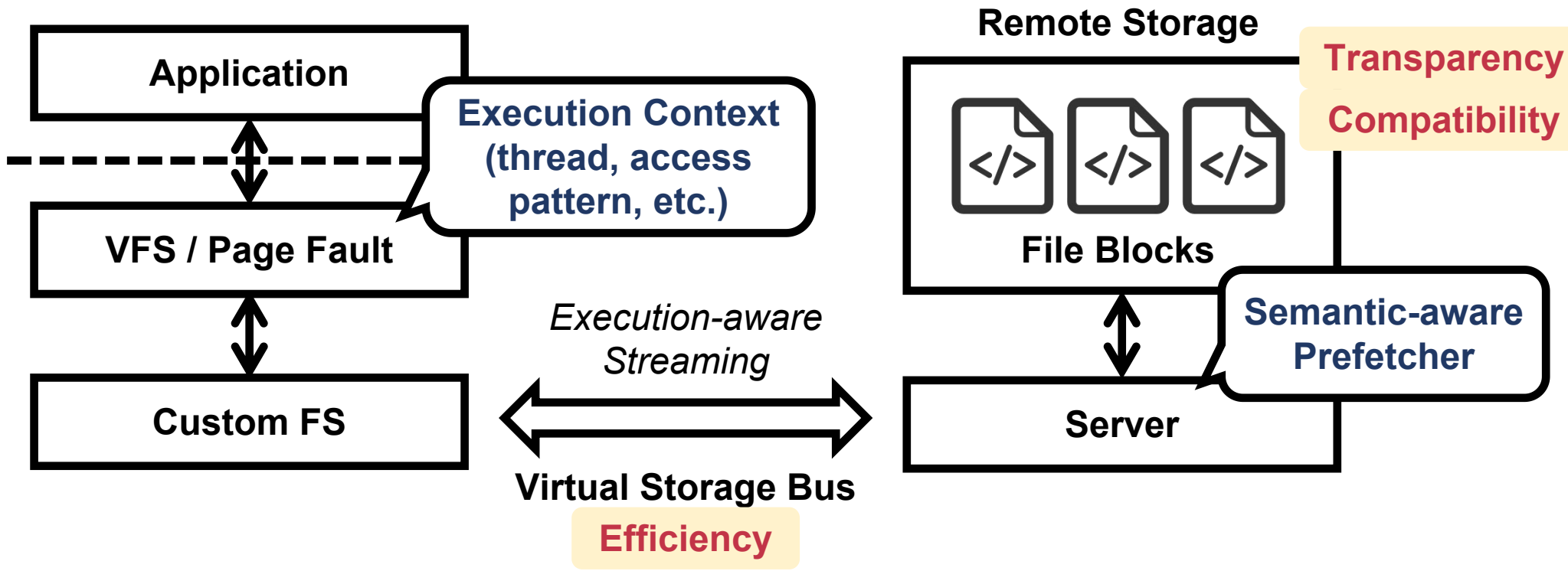
No installation, seamless execution

Goal 2: Efficiency

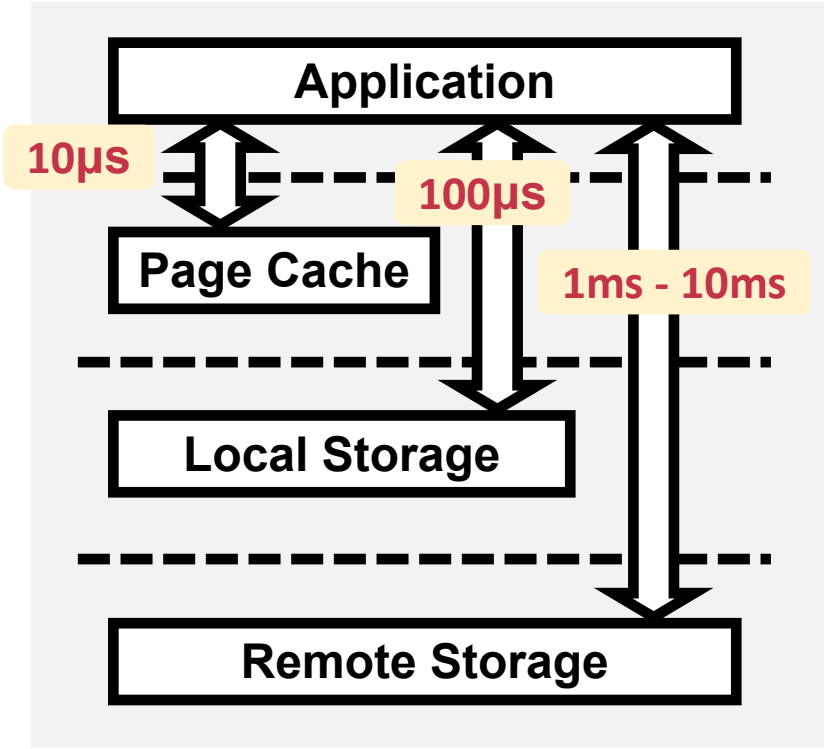
Hide latency over variable networks

Goal 3: Compatibility

Run unmodified applications



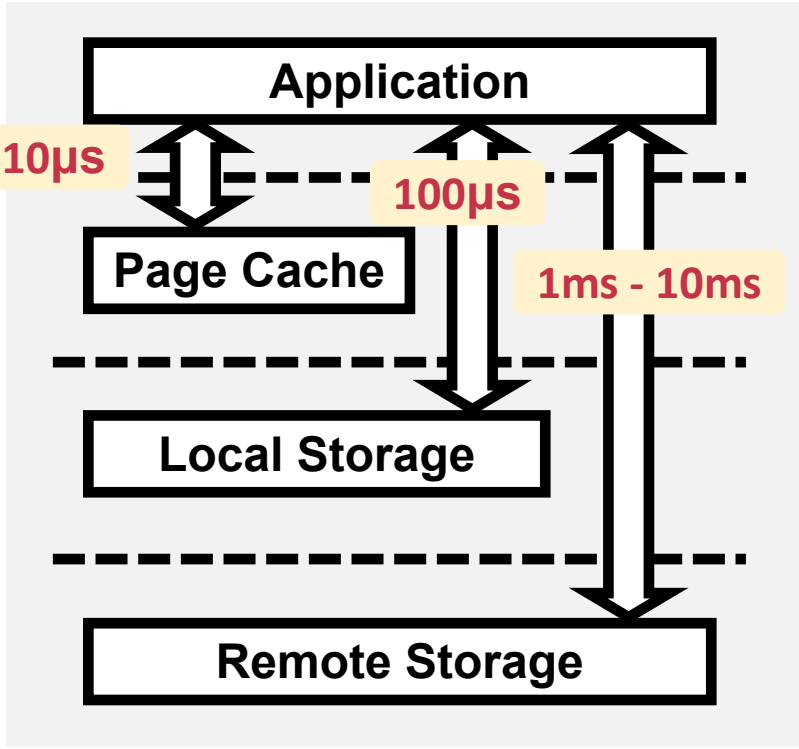
Two Fundamental Challenges



Latency Gap

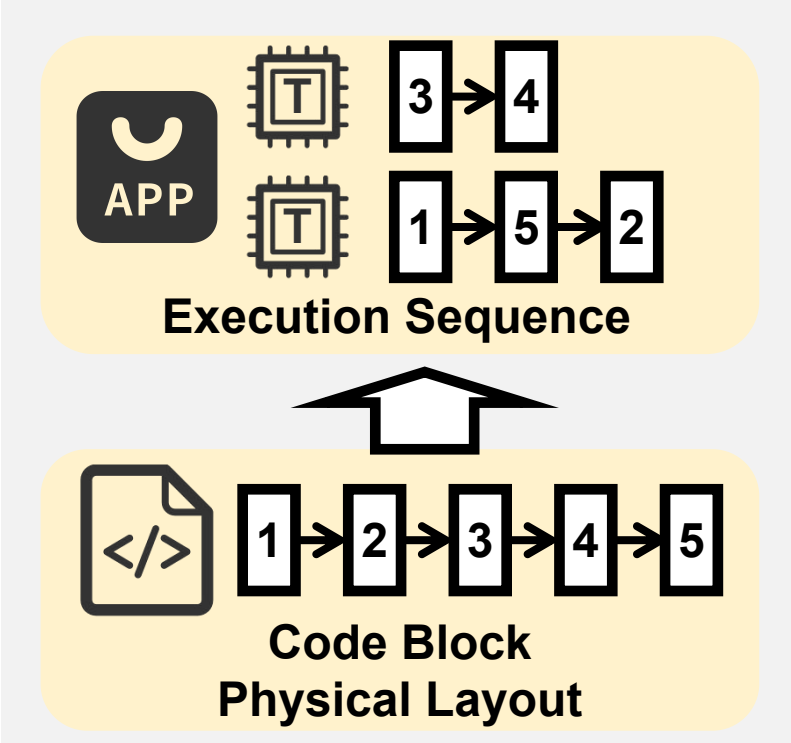
Execution requires **µs-level** response
Network delivers **ms-level** latency
→ **100×–1000× mismatch**

Two Fundamental Challenges



Latency Gap

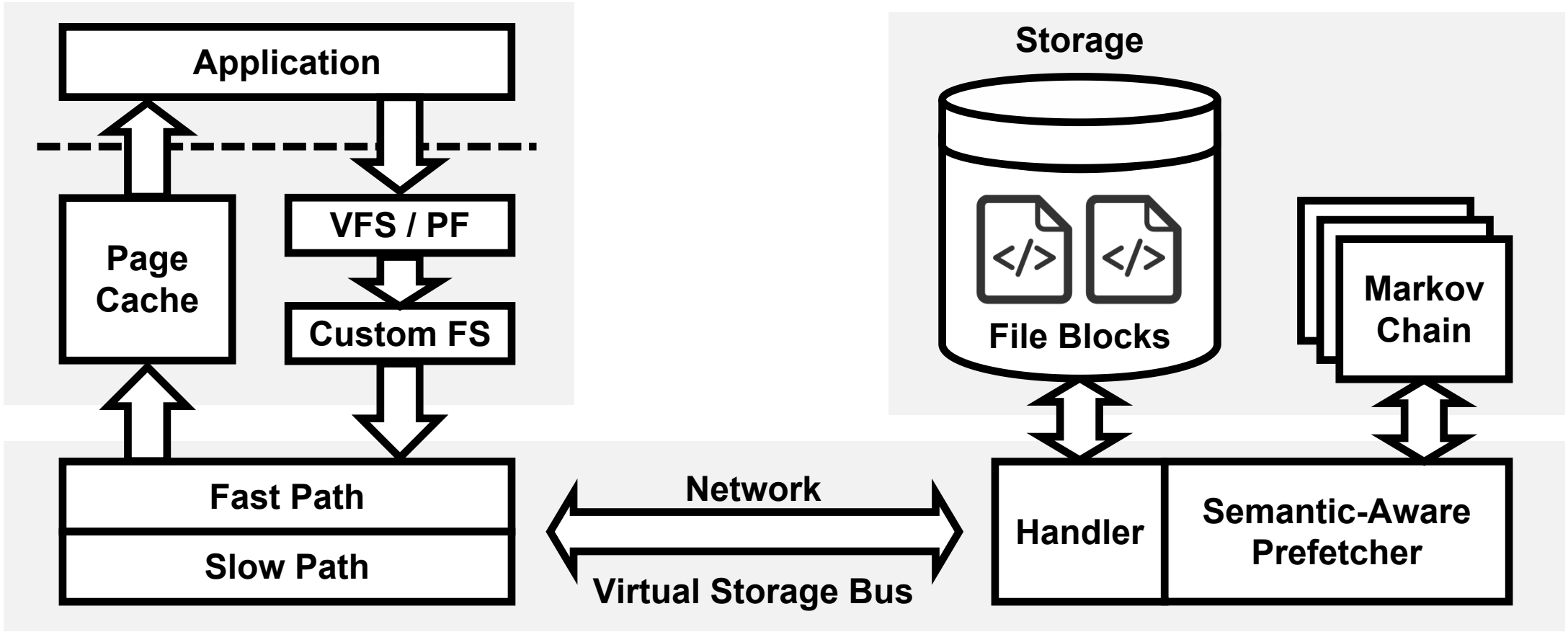
Execution requires **µs-level** response
Network delivers **ms-level** latency
→ **100×–1000× mismatch**



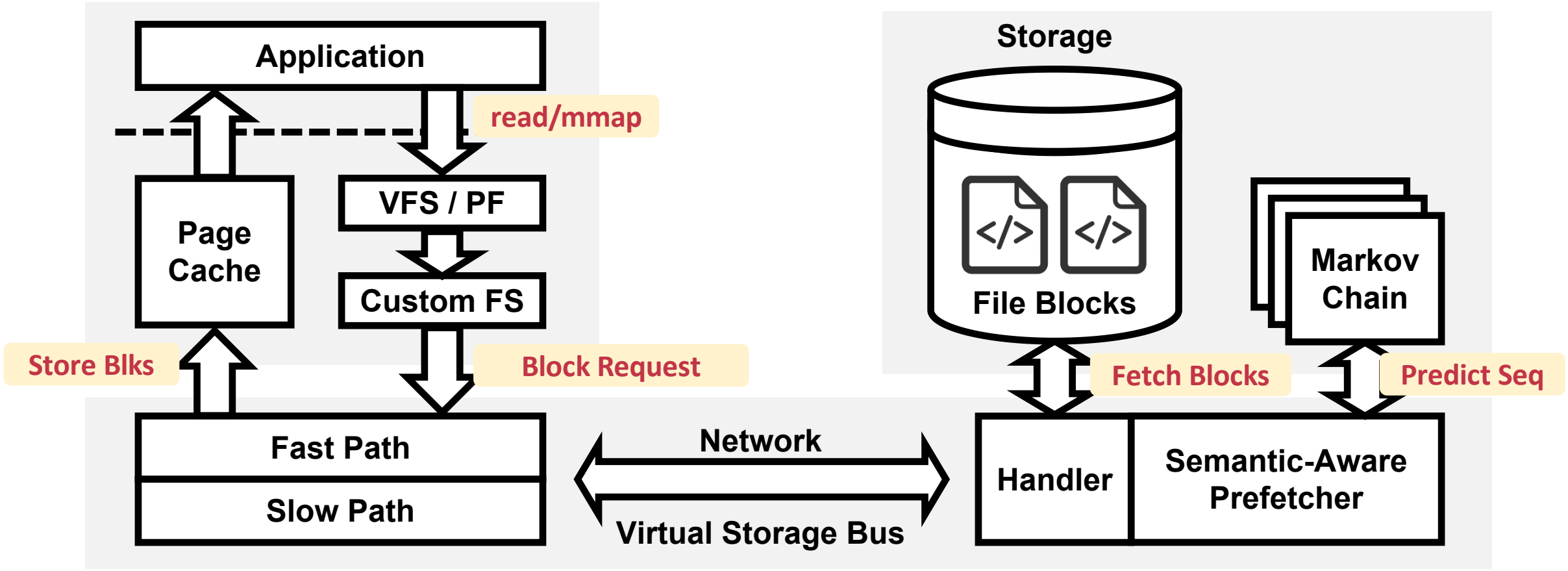
Context Gap

Storage is **stateless**
Execution is **stateful & thread-specific**
→ **generic prefetching is ineffective**

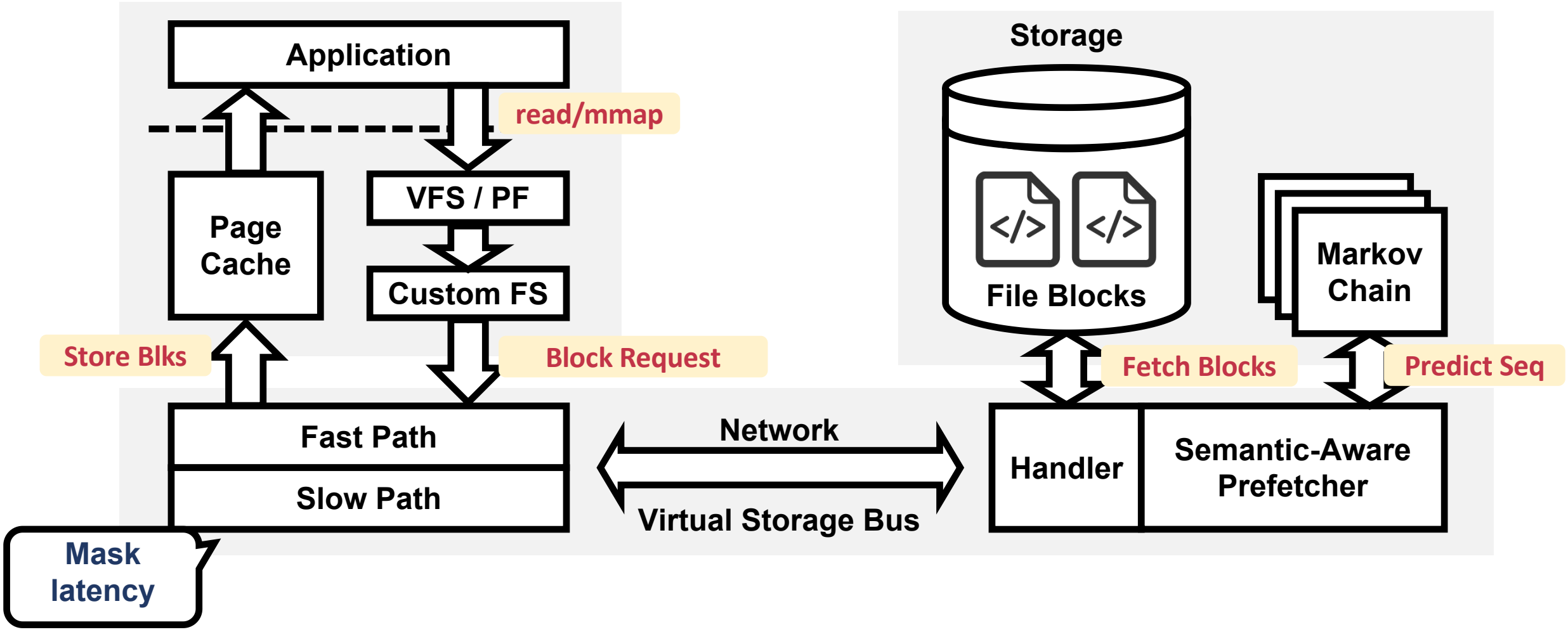
Architecture Overview



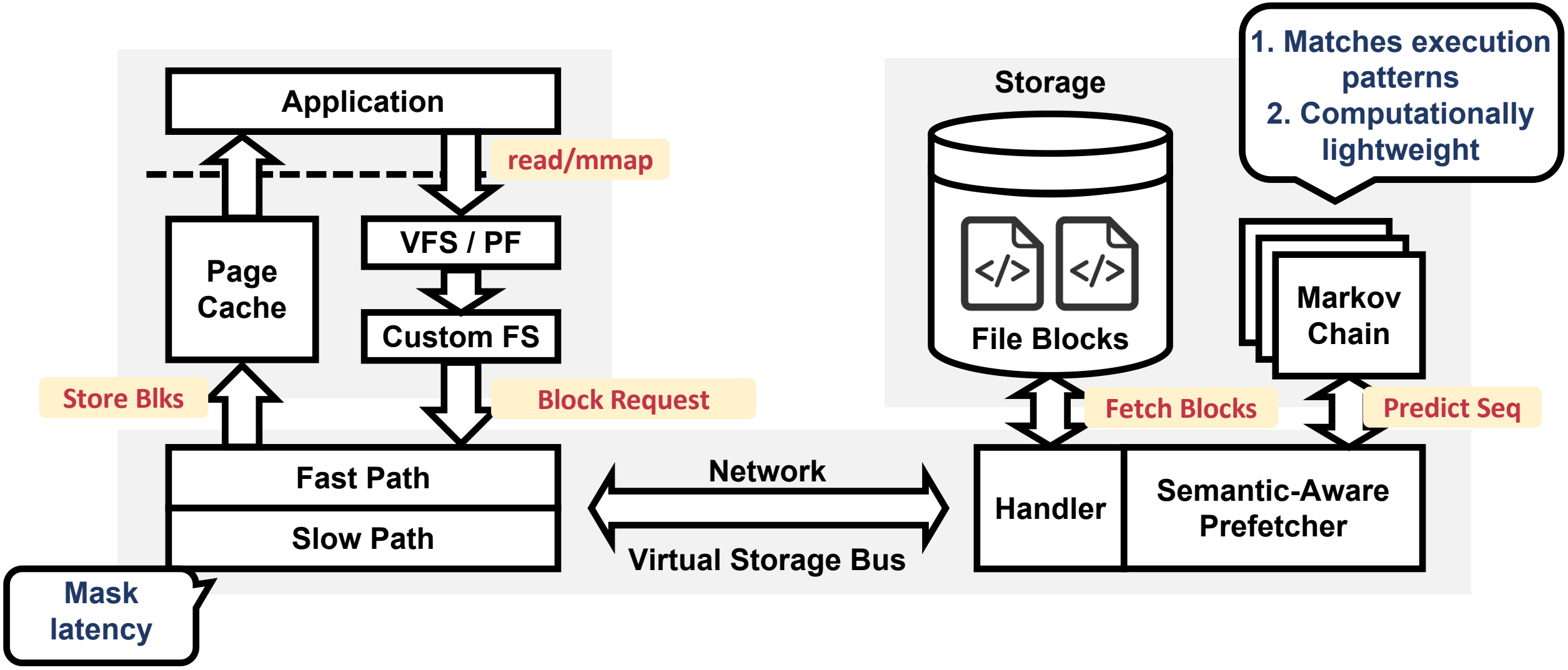
Architecture Overview



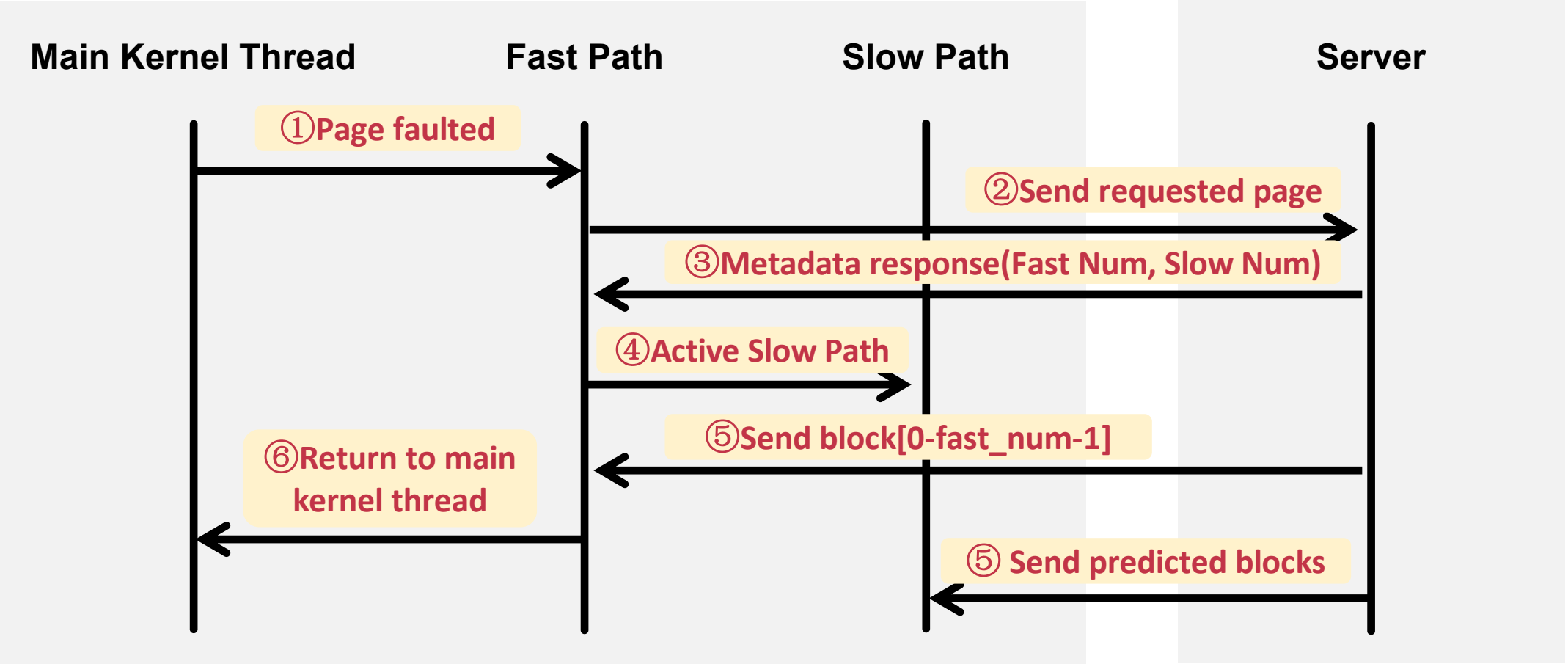
Architecture Overview



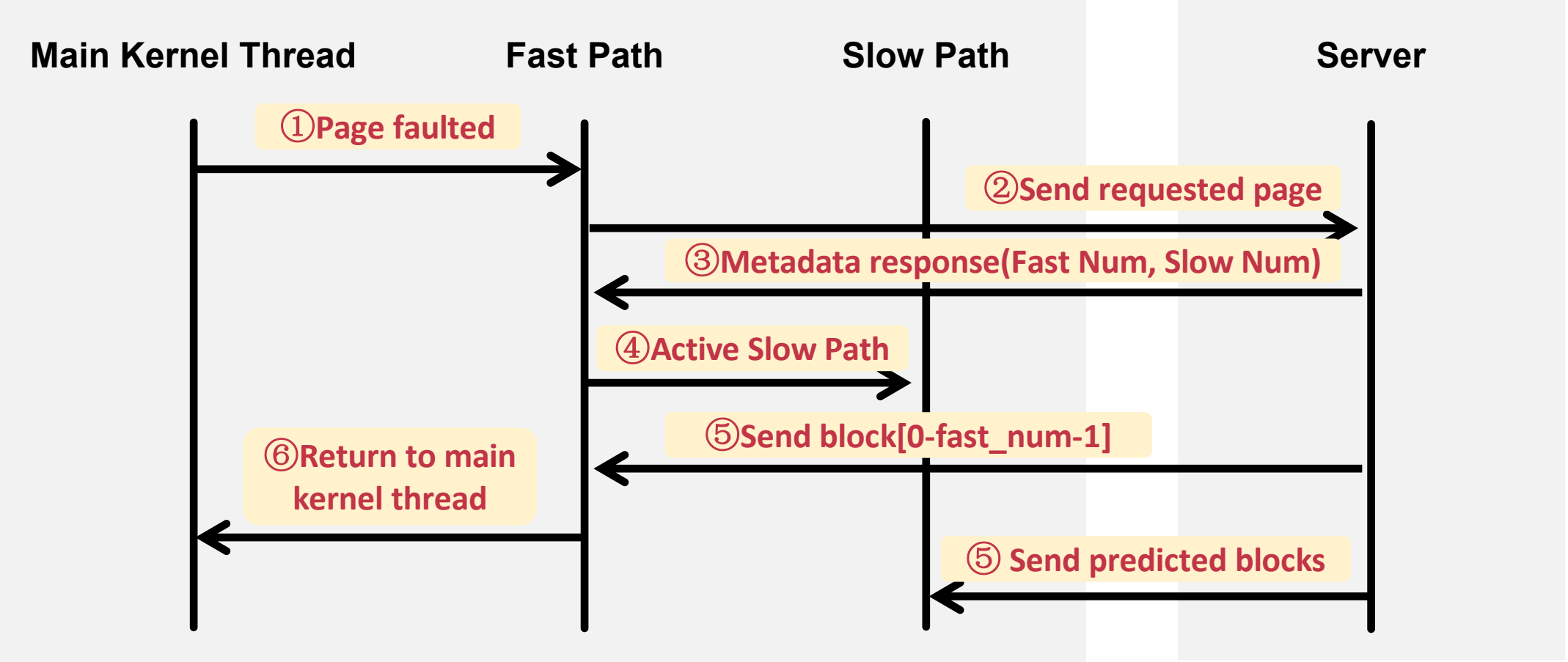
Architecture Overview



Dual-mode block delivery workflow

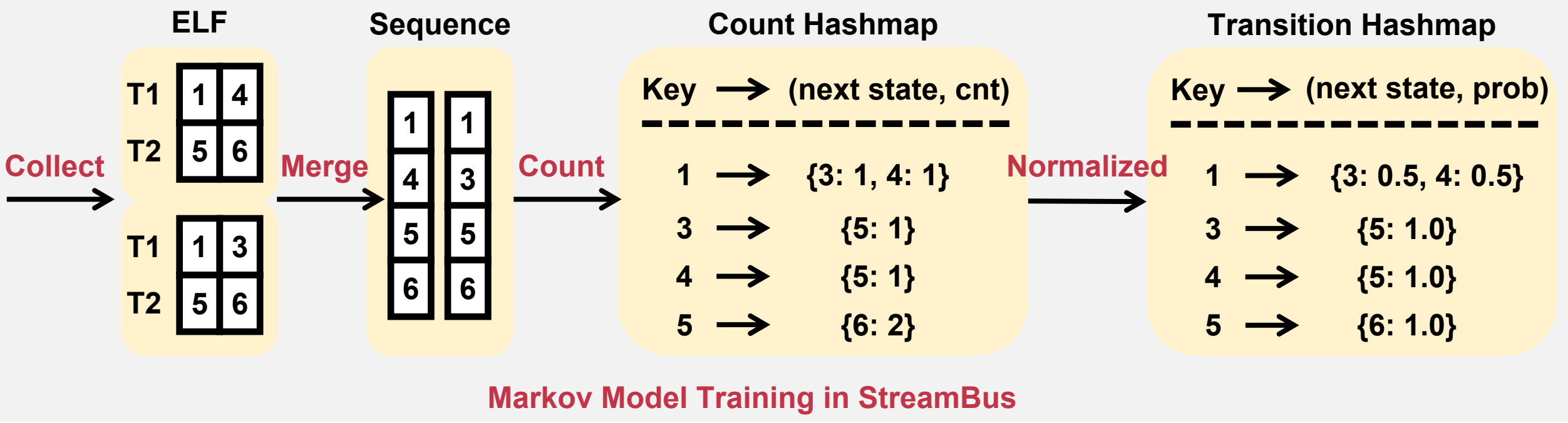


Dual-mode block delivery workflow



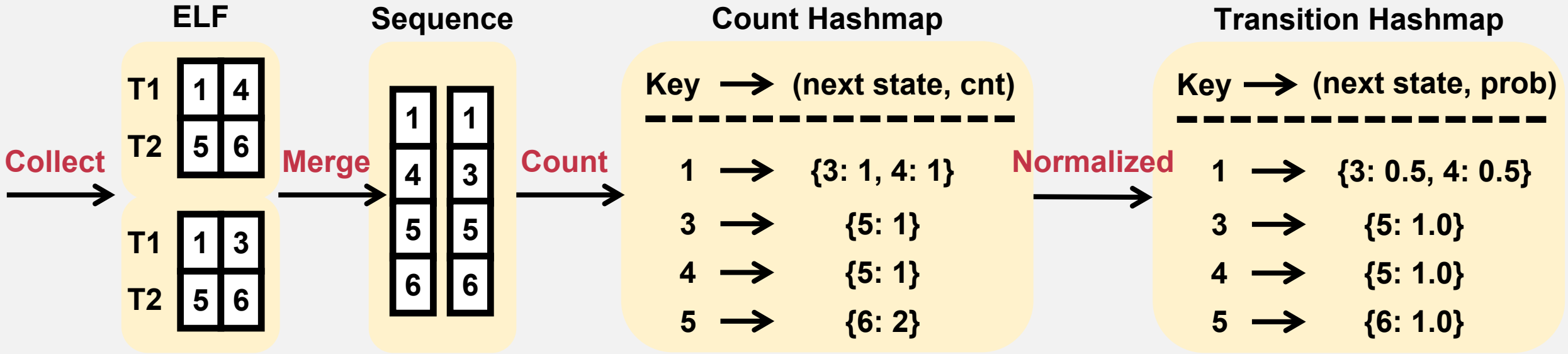
Fast path serves the **current demand**, while **slow path** prepares **future blocks**.

Thread-Aware Prefetching

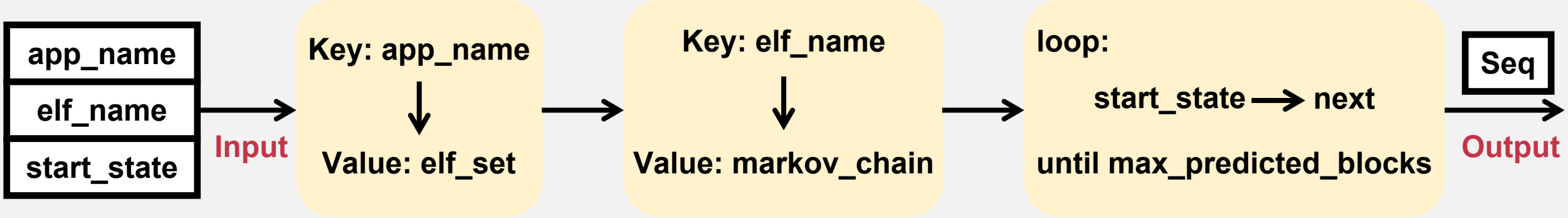


Markov Model Training in StreamBus

Thread-Aware Prefetching



Markov Model Training in StreamBus



StreamBus Inference Procedure

Evaluation Setup

- **Hardware Setup**
  - **Dell OptiPlex 5070 desktop: 1 TB NVMe SSD**
  - **Google Pixel 6 Pro: 128 GB UFS 3.1 storage**
  - **NETGEAR Router: 1 Gbps Ethernet, 802.11ac (5GHz)**
- **Baseline**
  - **Local: NVMe SSD, UFS 3.1 storage**
  - **Network: NFSv4**

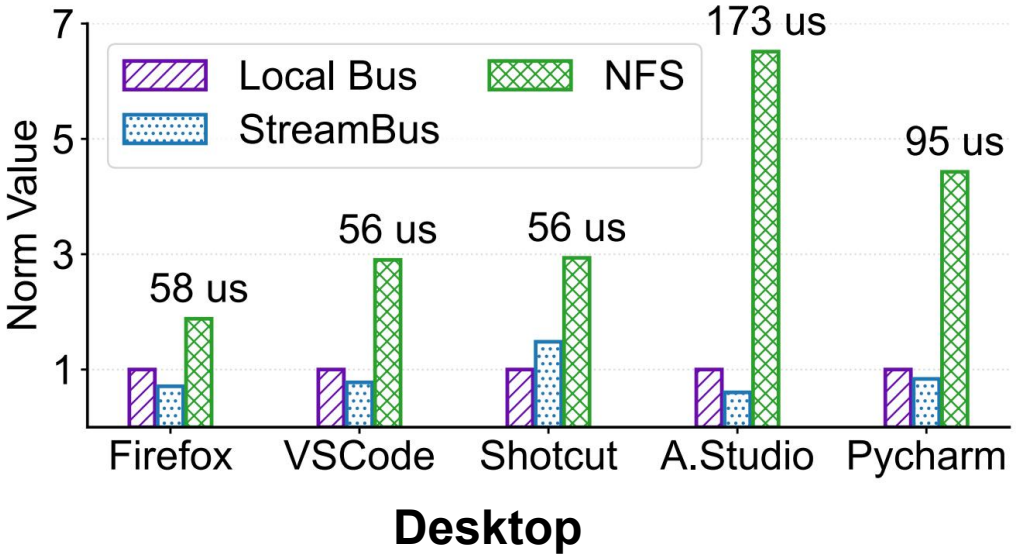
Desktop		Mobile	
Pycharm	3.0GB	Uber	447MB
A. Studio	2.9GB	WPS	377MB
Shotcut	685MB	Spotify	193MB
VSCode	409MB	Whatsapp	126MB
Firefox	260MB	Telegram	122MB

Microbenchmark

- **Up to ~40% higher cache hit rate than both local storage and NFS**

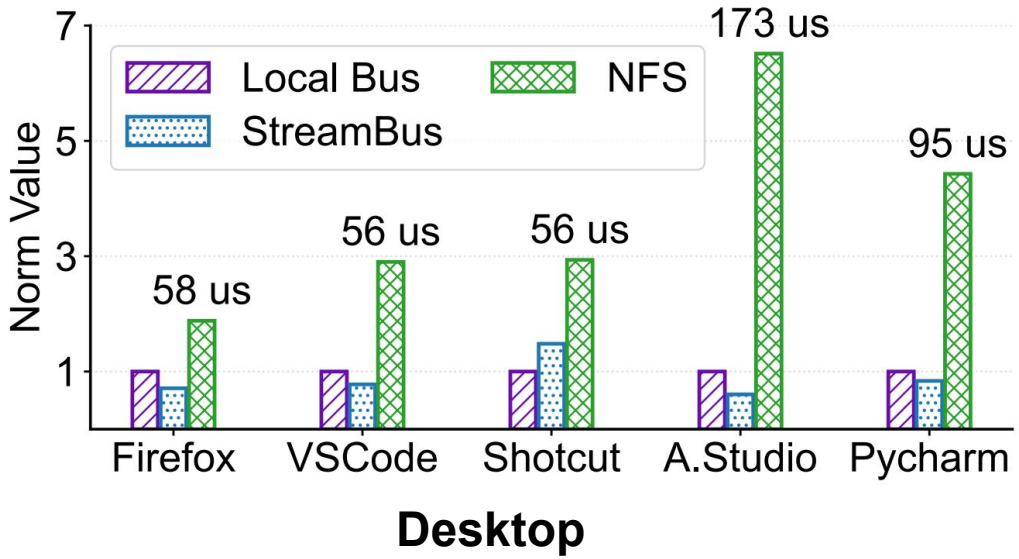
Microbenchmark

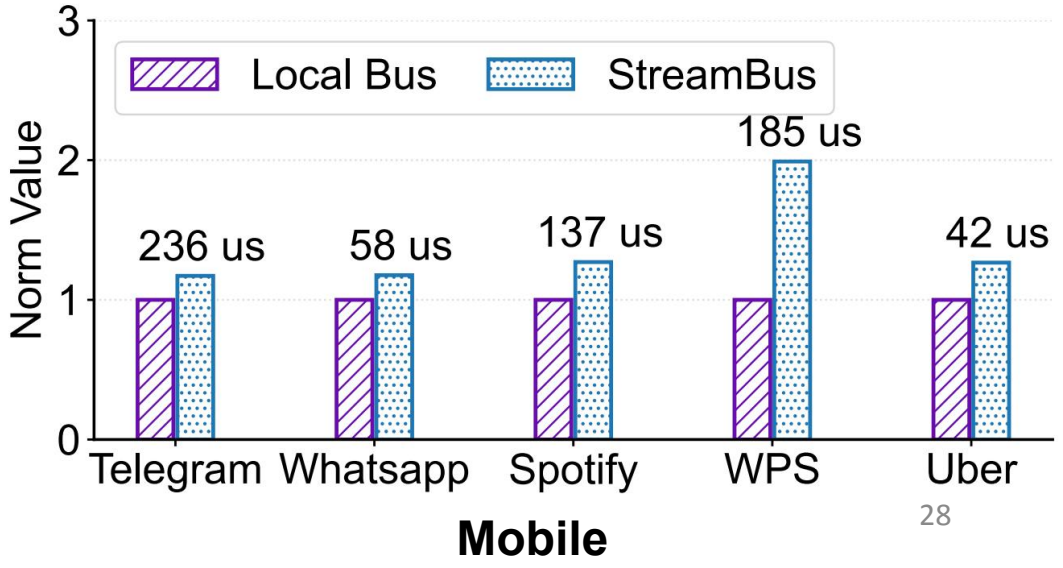
- **Up to ~40% higher cache hit rate than both local storage and NFS**
- **Up to 16–40% lower latency than NVMe SSD on desktop (cold start)**



Microbenchmark

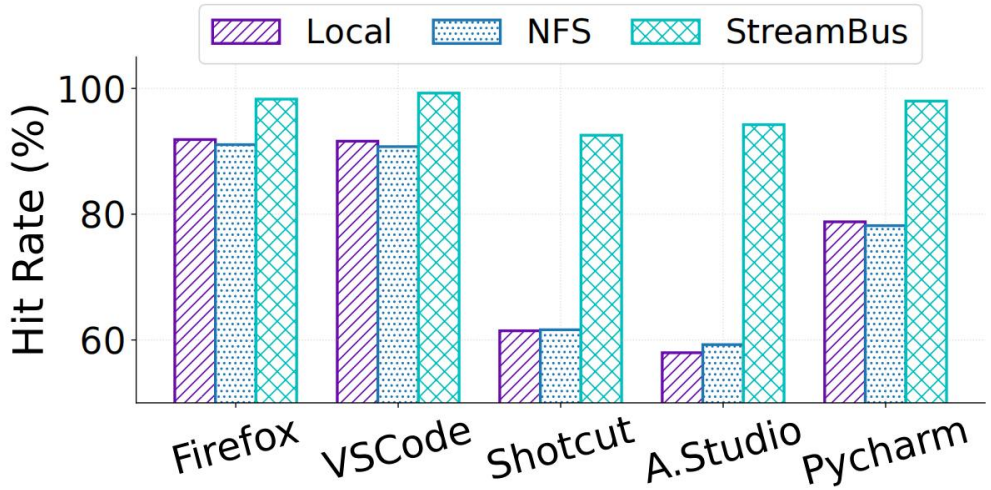
- **Up to ~40% higher cache hit rate than both local storage and NFS**
- **Up to 16–40% lower latency than NVMe SSD on desktop (cold start)**
- **Within 17–27% of local storage latency on mobile**

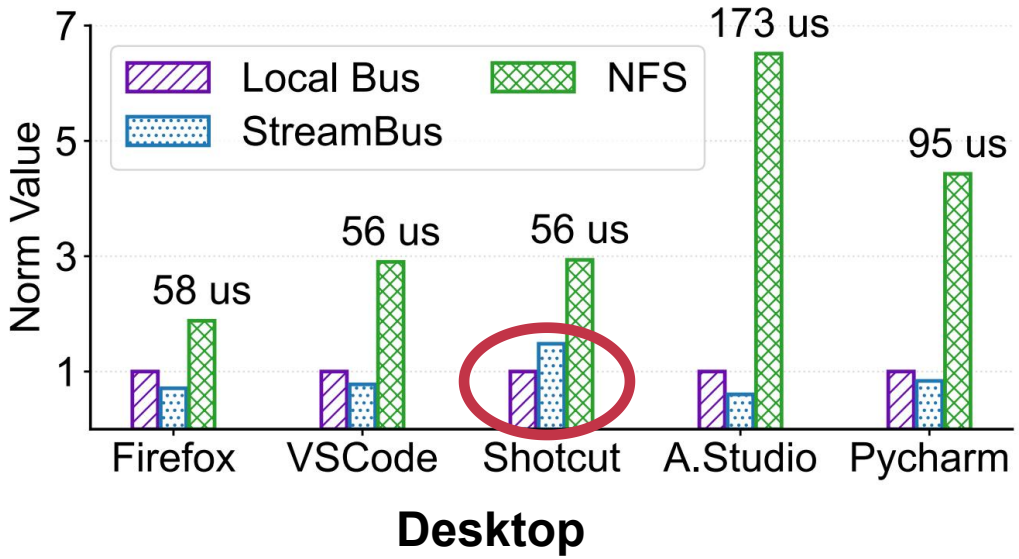


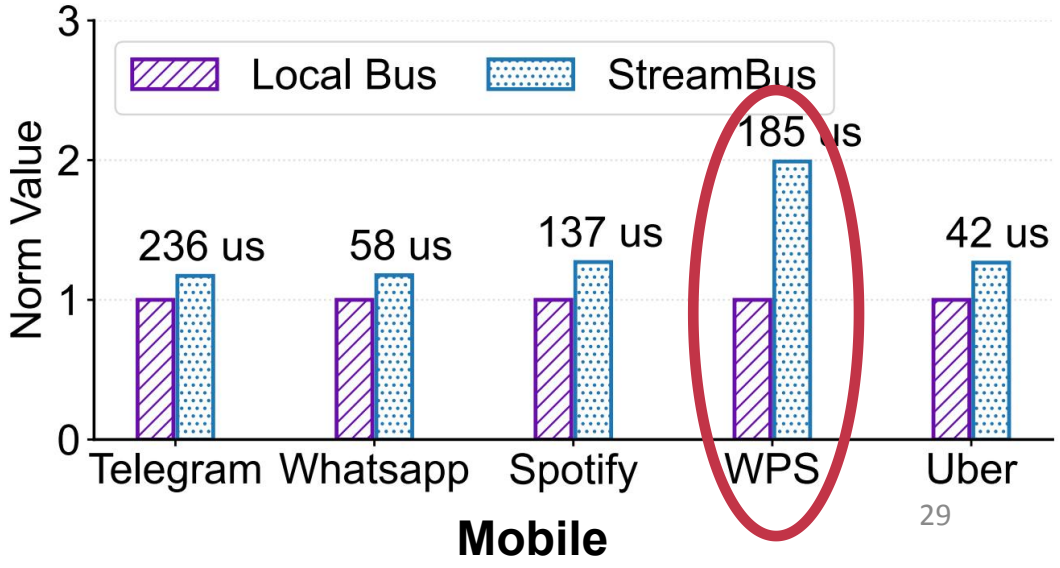


Microbenchmark

- **Up to ~40% higher cache hit rate than both local storage and NFS**
- **Up to 16–40% lower latency than NVMe SSD on desktop (cold start)**
- **Within 17–27% of local storage latency on mobile**

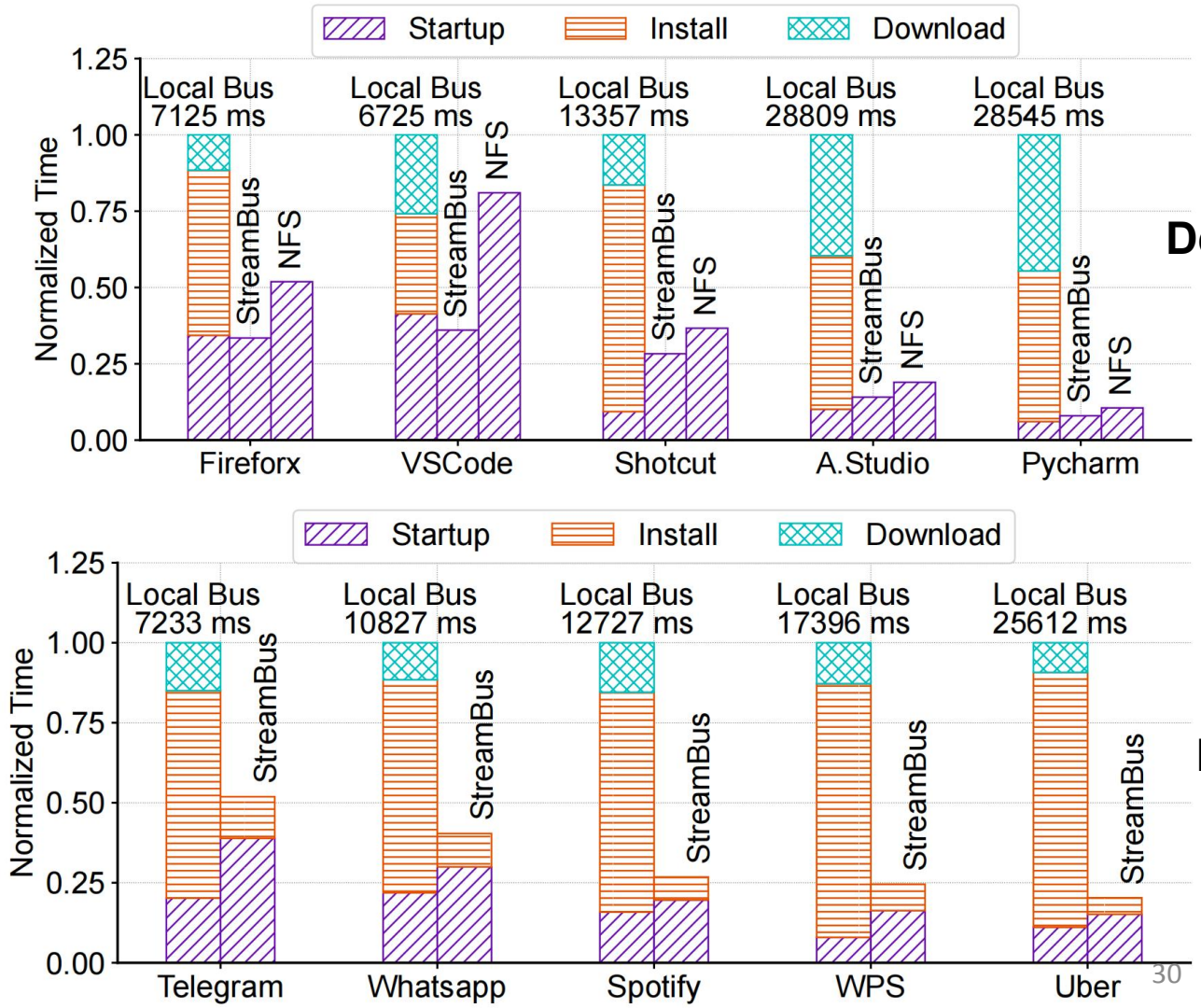






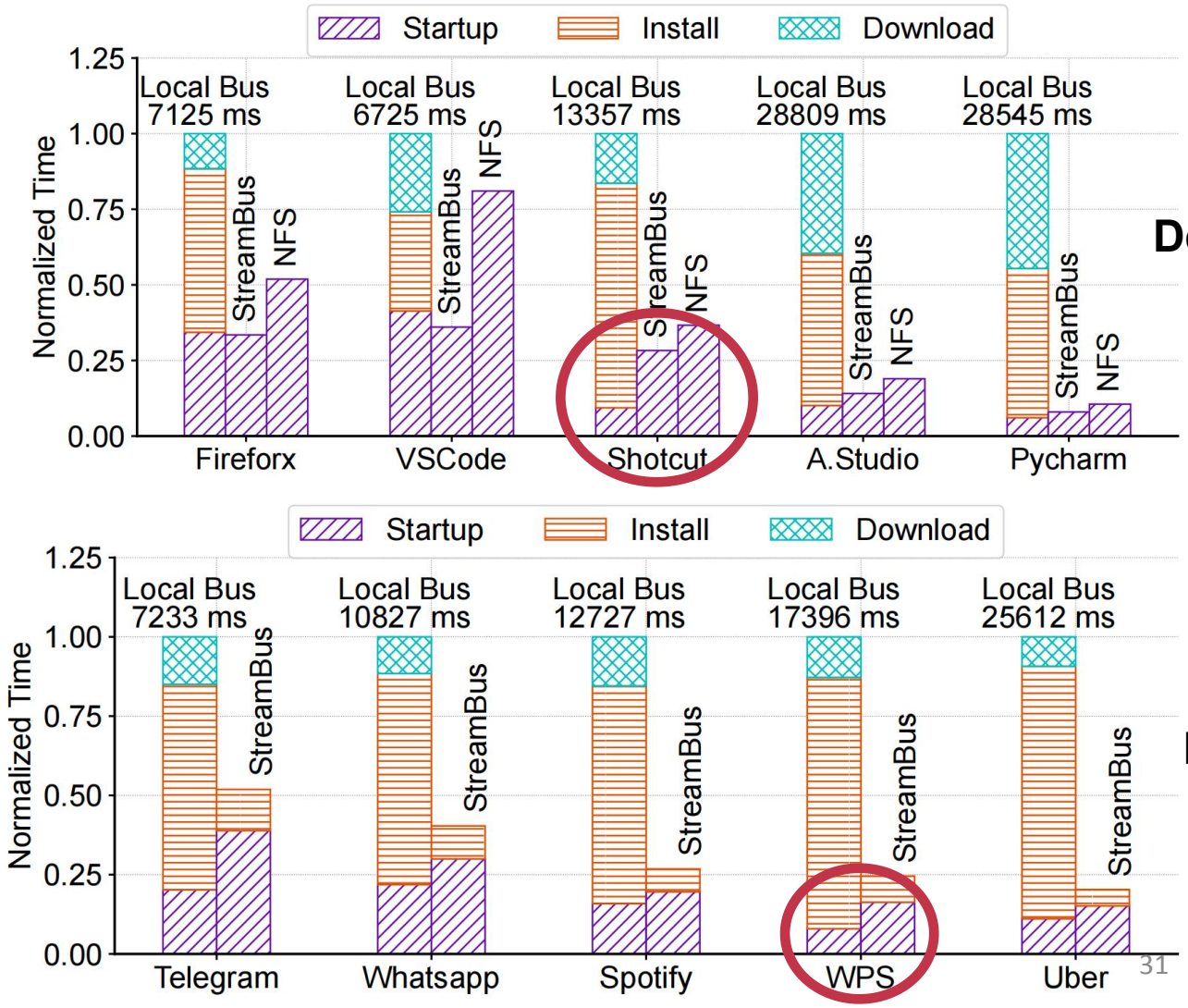
End-to-End Performance

- **Workflow: Download → Install → Startup**
- **↓ 20–55% startup latency vs. NFS**
- **≈ NVMe-level startup on desktop**
- **≤ 40% startup overhead on mobile**



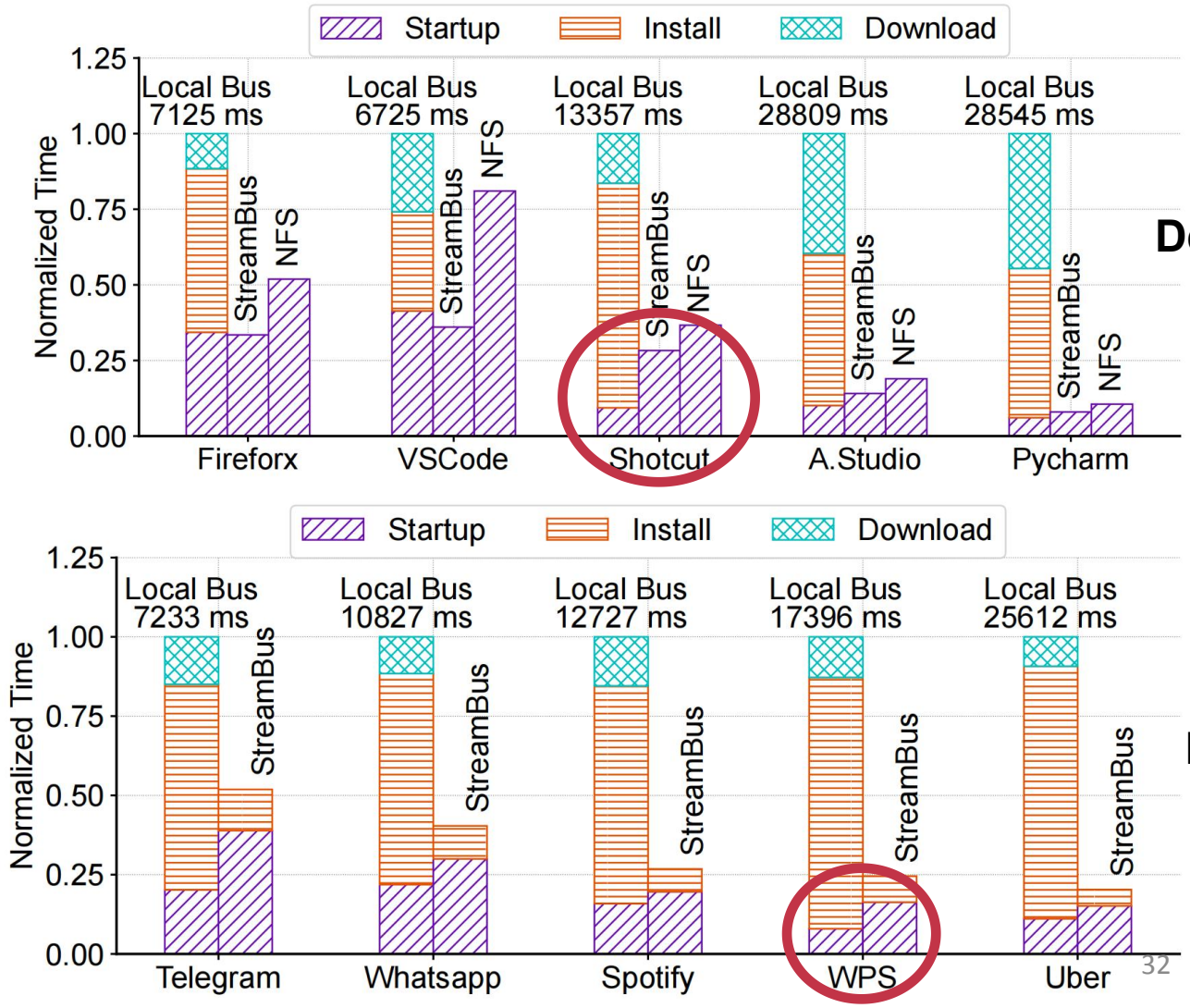
End-to-End Performance

- **Workflow: Download → Install → Startup**
- **↓ 20–55% startup latency vs. NFS**
- **≈ NVMe-level startup on desktop**
- **≤ 40% startup overhead on mobile**



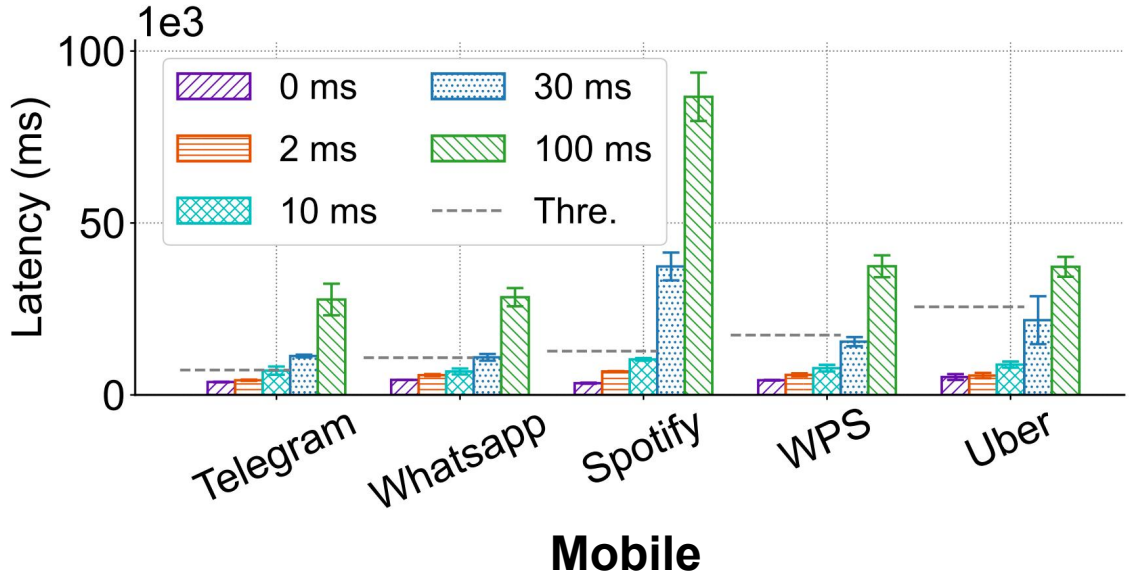
End-to-End Performance

- **Workflow: Download → Install → Startup**
- **↓ 20–55% startup latency vs. NFS**
- **≈ NVMe-level startup on desktop**
- **≤ 40% startup overhead on mobile**
- **↓ 45–78% end-to-end time (desktop)**
- **↓ 36–80% end-to-end time (mobile)**

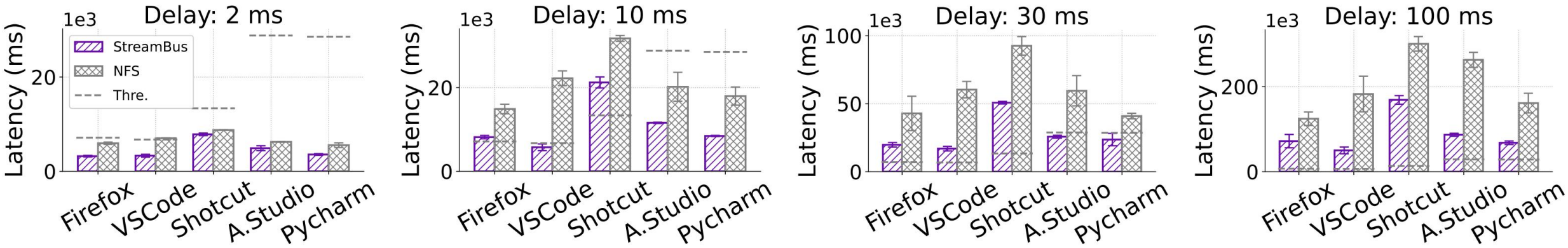


Robustness Under Network Variability

- **Set up: Inject RTT: 2–100 ms (+ loss & jitter)**
- **Metric: End-to-end delay**
- **Dashed line = local install baseline**
- **Performs well under intra-city RTT (2 ms)**
- **Remains competitive under inter-city RTT (10 ms)**



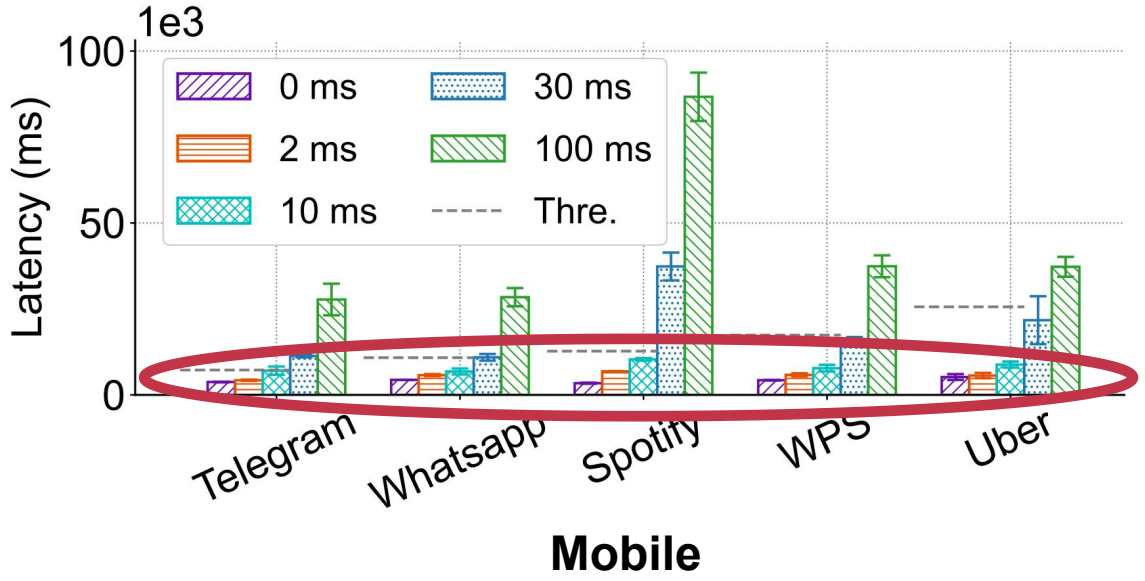
Mobile



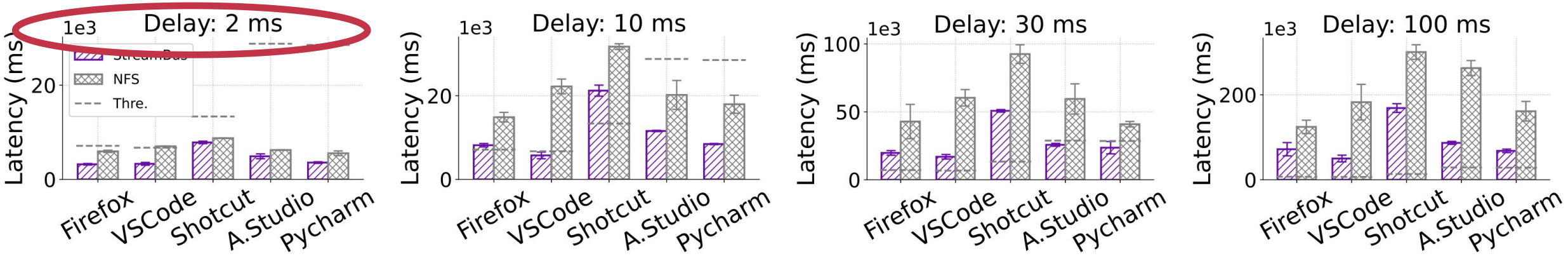
Desktop

Robustness Under Network Variability

- **Set up: Inject RTT: 2–100 ms (+ loss & jitter)**
- **Metric: End-to-end delay**
- **Dashed line = local install baseline**
- **Performs well under intra-city RTT (2 ms)**
- **Remains competitive under inter-city RTT (10 ms)**



Mobile



Desktop

Conclusion

- **StreamBus: Enables on-demand execution without installation via a semantic virtual storage bus.**
- **Performance: Masks network latency, achieving near-local responsiveness**
- **Future: Towards zero-install applications and edge-cloud execution.**

Q & A