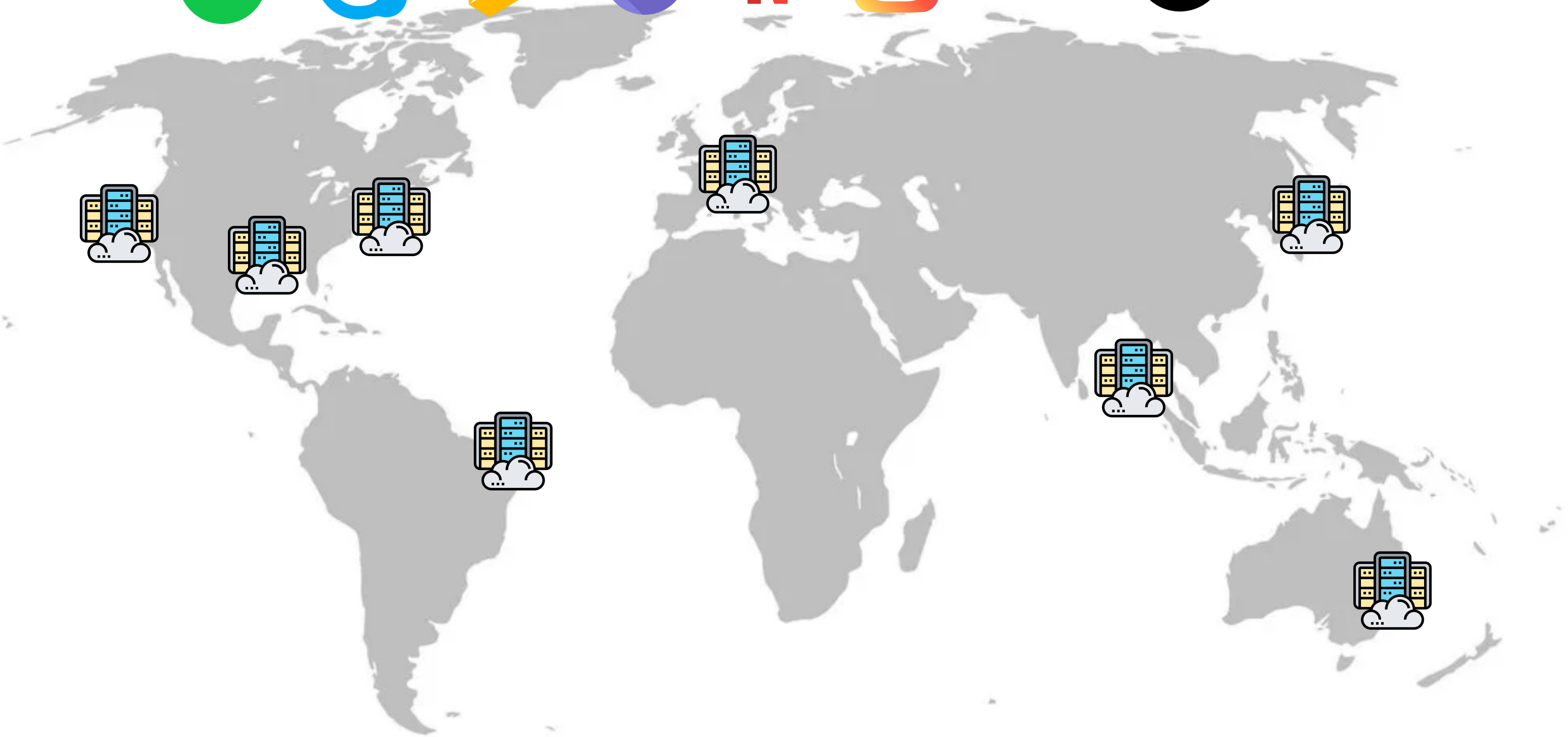
SWAN: WAN-aware Stream Processing on Geographically-distributed Clusters

Won Wook SONG, Myeongjae Jeon, and Byung-Gon Chun

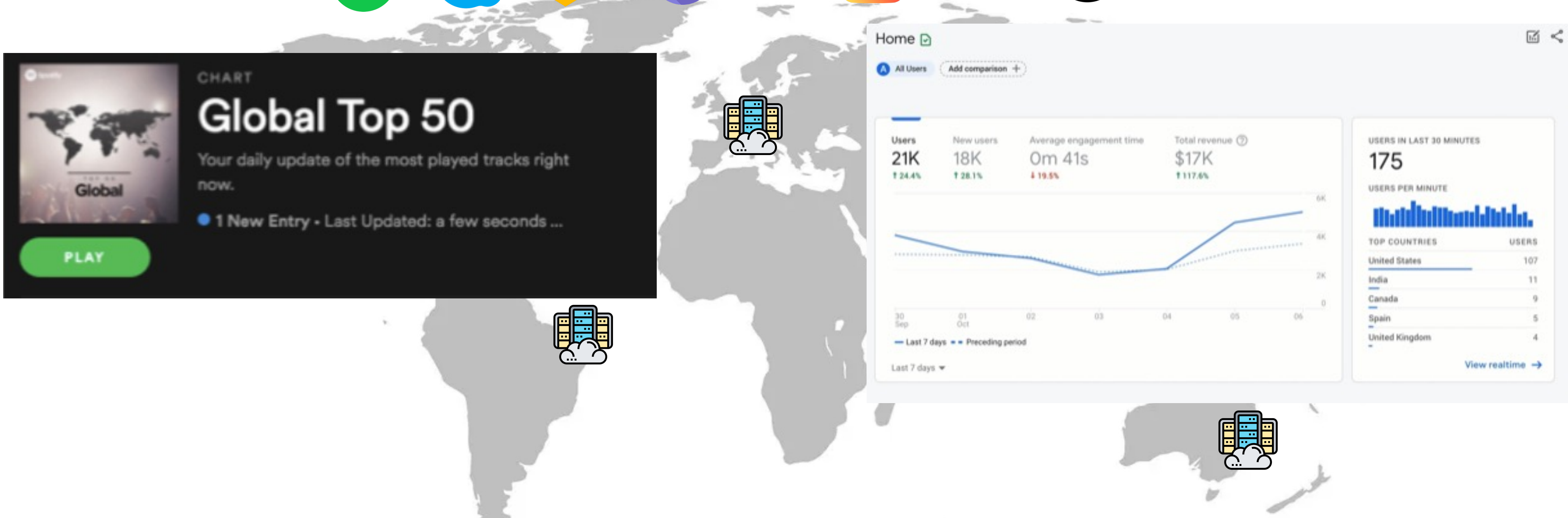
Wide-area Streaming Analytics



Applications placed on multiple DCs
to provide low latency access

image: Flaticon.com

Demand for Analyzing Data from Multiple Datacenters



Need to extract business insights from global log data and metrics

e.g. average, success rate, requests per document, top K, hot items..

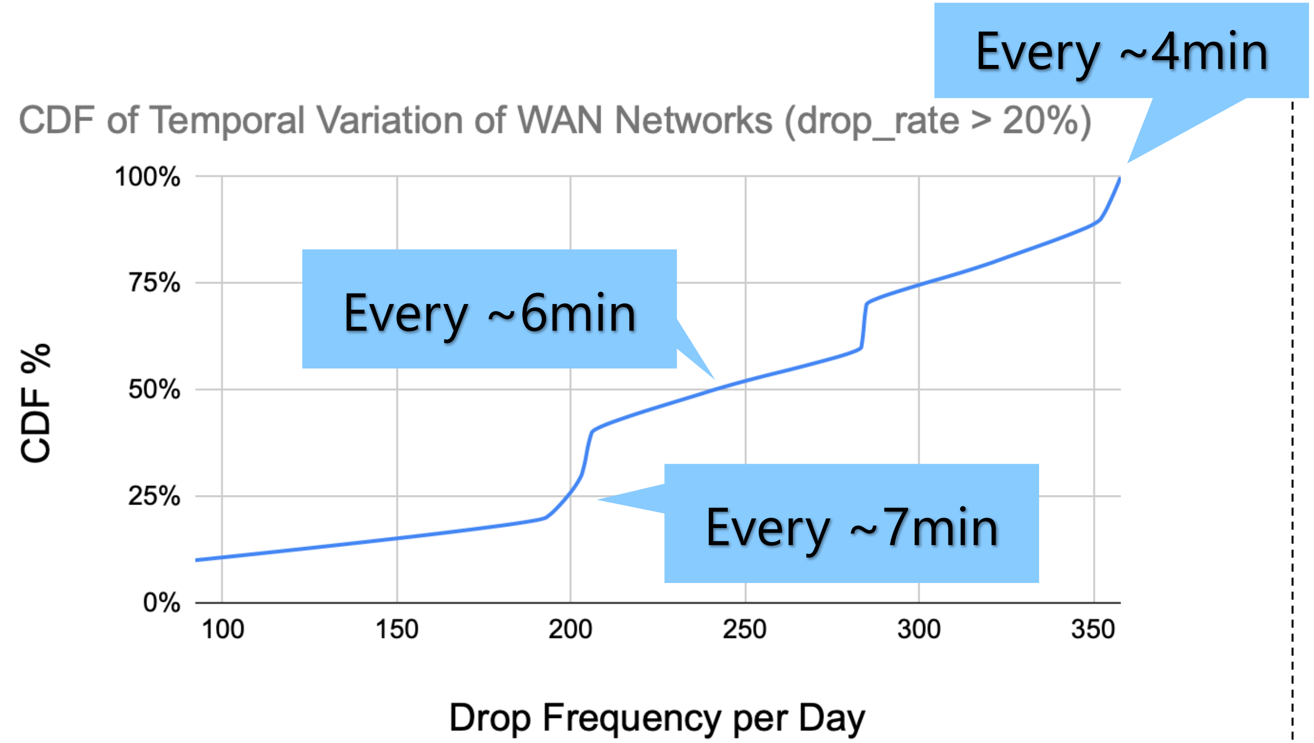
image: Flaticon.com

WAN Characteristics

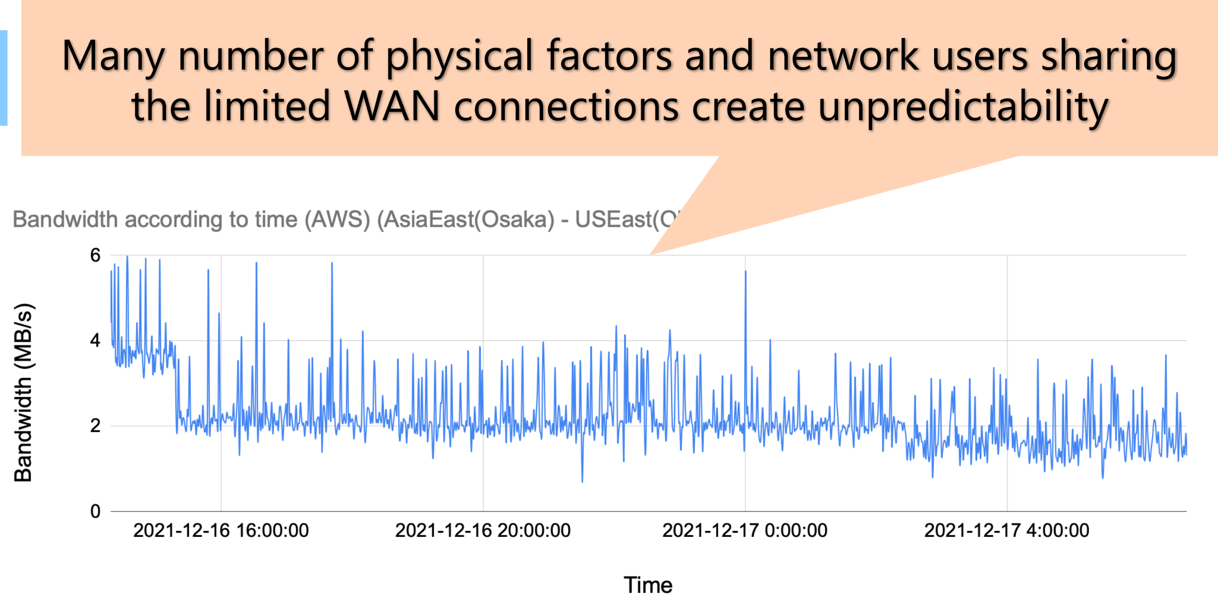
- Observe a GCP Cluster of *16 nodes* across *8 regions* over *3 continents*
  - e2-standard-4 (4vCPUs, 16GB Memory)
  - Asia: Taiwan, Mumbai
  - Europe: Finland, Belgium, Netherlands
  - N. America: Iowa, South Carolina, Oregon
- Observe WAN networks between AWS nodes from 5 regions
  - Asia: Osaka / Europe: Ireland / N. America: Canada, Ohio, Oregon

image: Flaticon.com

WAN Characteristics 1: Temporal Variability



Networks have varying drop frequencies

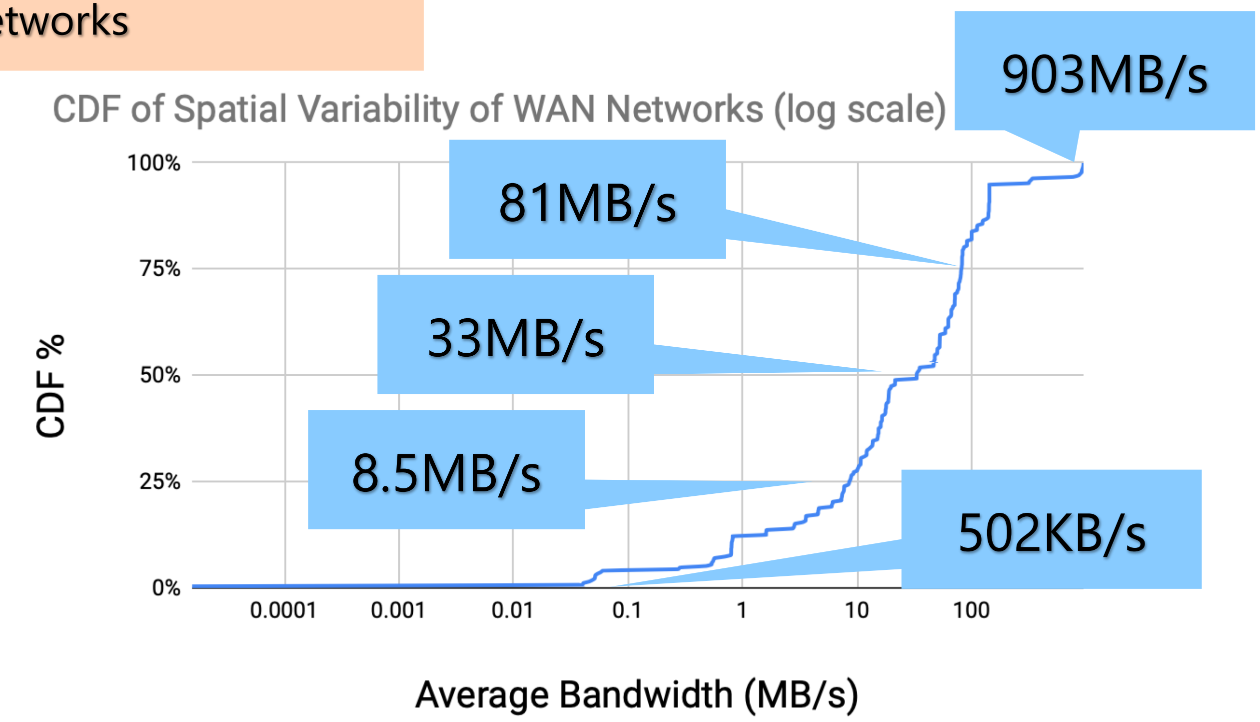


A network example showing bandwidth fluctuation over *time*

image: Flaticon.com

WAN Characteristics 2: Spatial Variability

ISPs operate different infrastructures/equipments between LAN networks



Average bandwidths vary among different *locations*

image: Flaticon.com

Stream Processing System Requirements

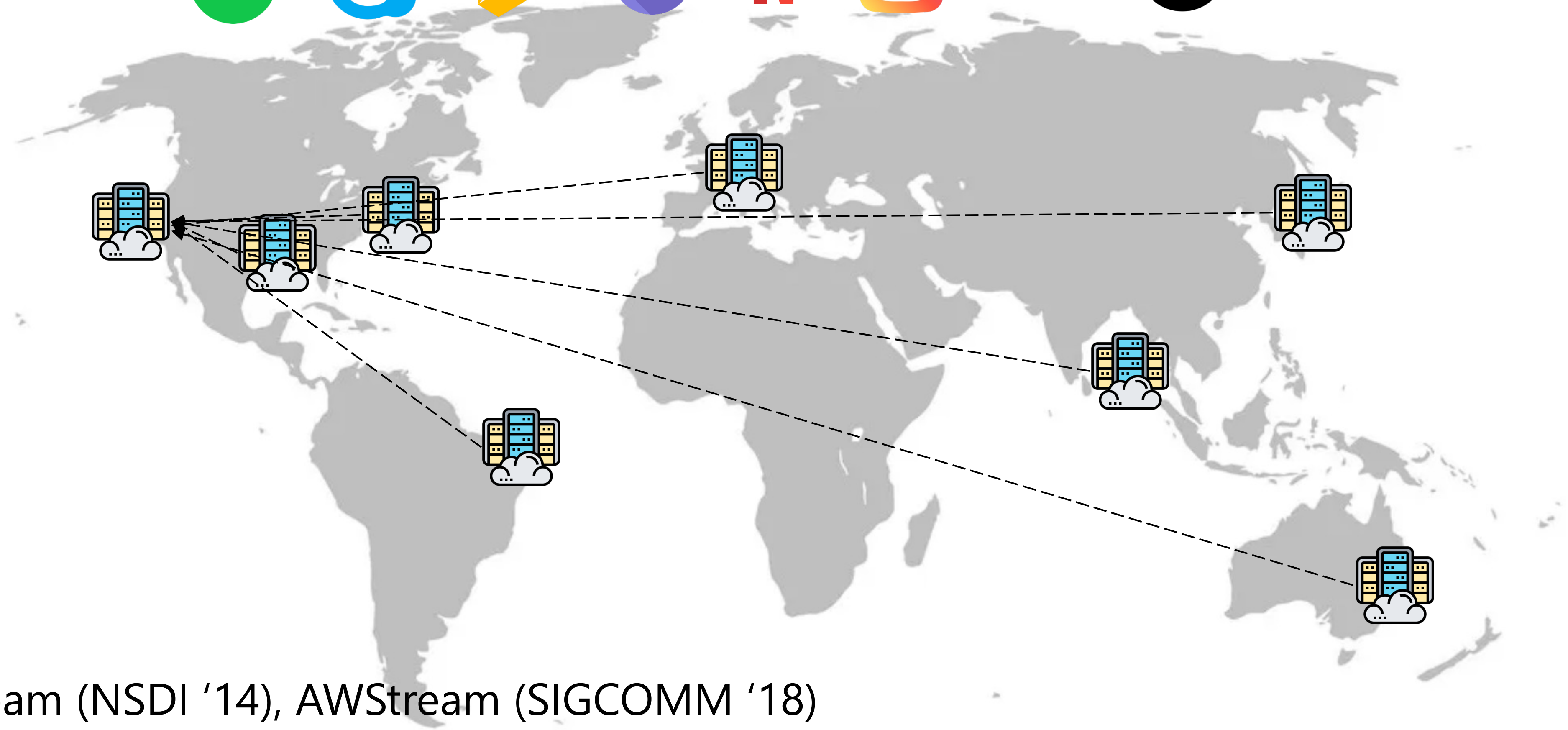
Low latency

High throughput

Correctness

Fast Adaptation

Existing Approach 1: Centralized Processing



Ex. JetStream (NSDI '14), AWStream (SIGCOMM '18)

Aggregate data to a single datacenter to use a conventional stream data analytics engine

image: Flaticon.com

Centralized Processing are Inaccurate or App-Specific

1. Pre-aggregation, degradation, statistical approximation for reducing the latency are often **app-specific**
2. Existing approaches of degrading raw data affects the **result accuracy**

Cannot be applied to workloads like fraud detection, billing, transactional analysis

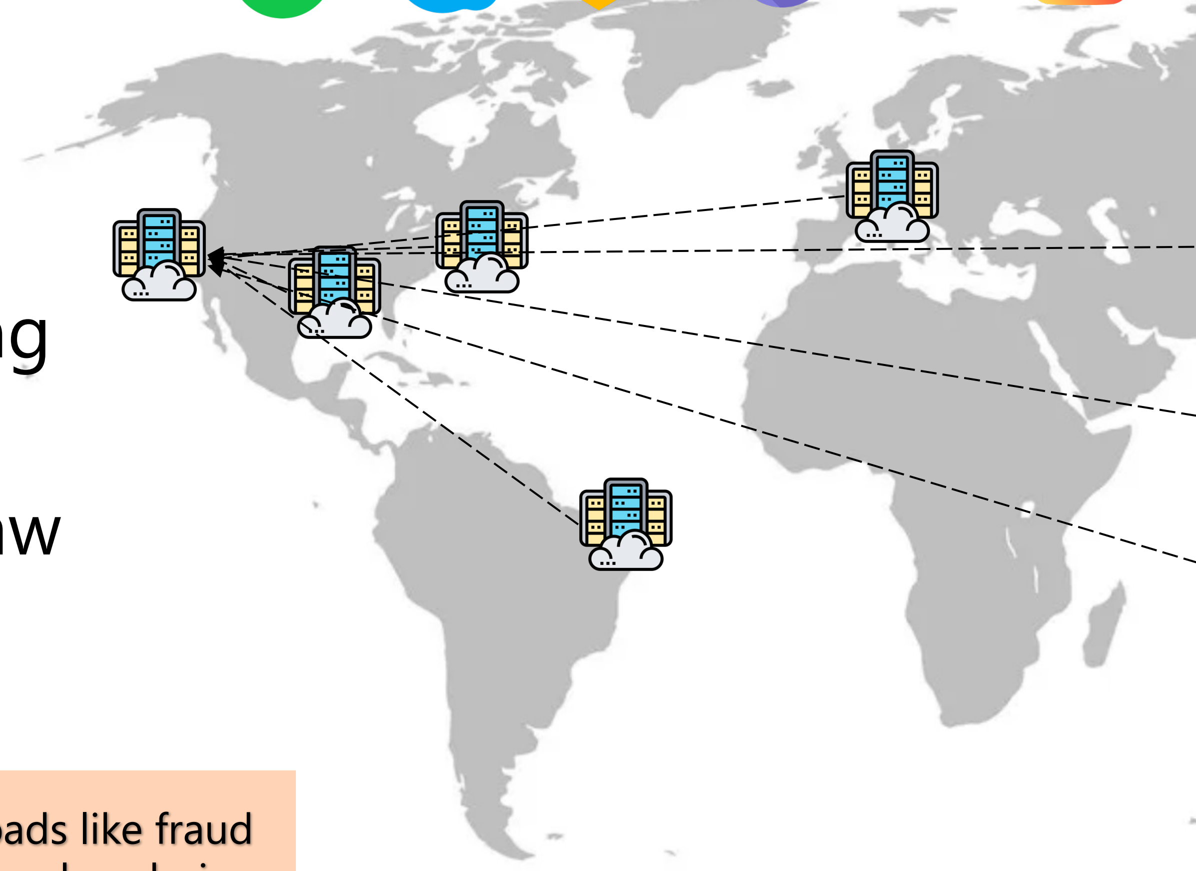
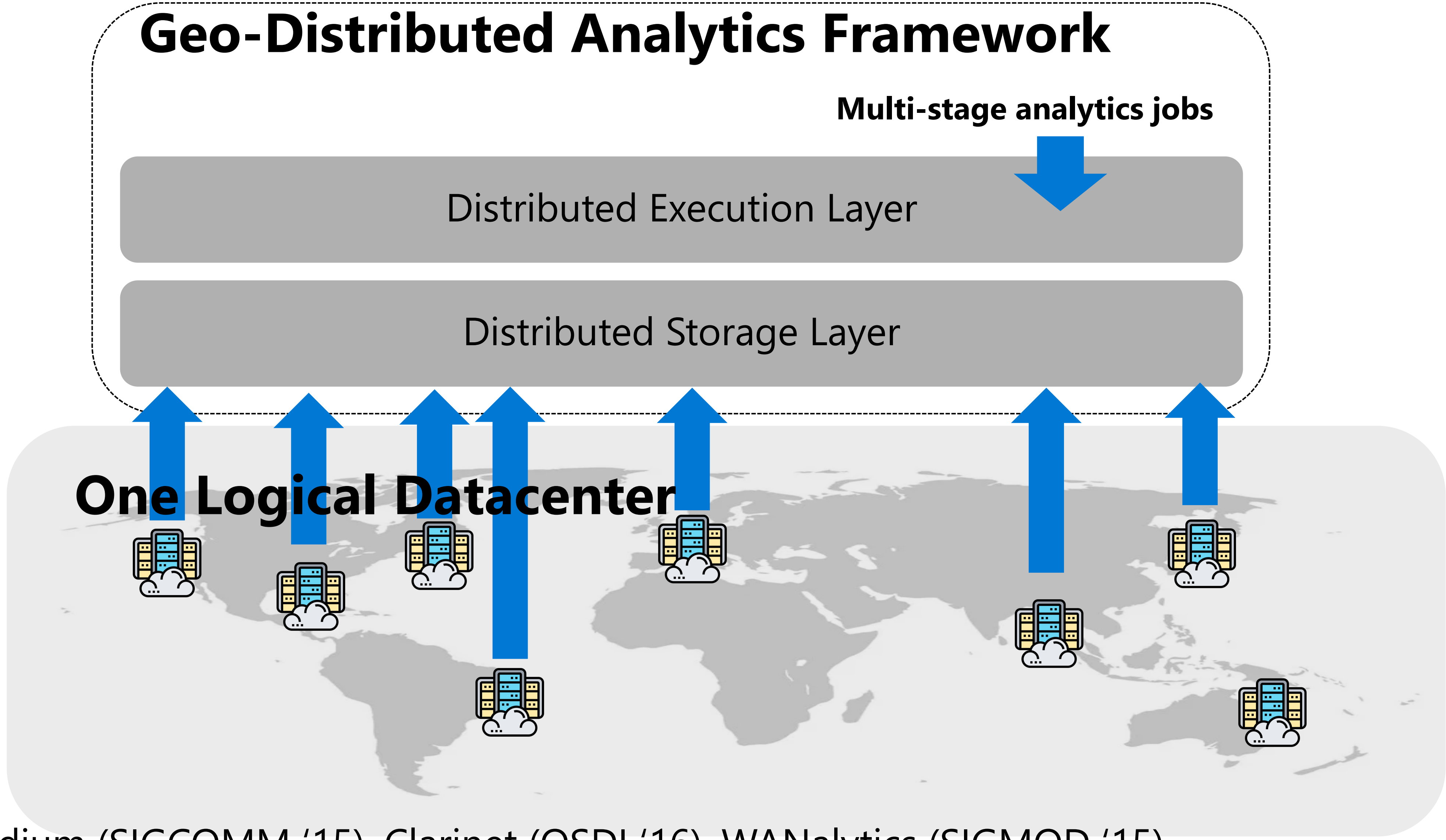


image: Flaticon.com

Approach 2: A Single Geo-Distributed Logical Cluster



Ex. Iridium (SIGCOMM '15), Clarinet (OSDI '16), WANalytics (SIGMOD '15)

image: Flaticon.com

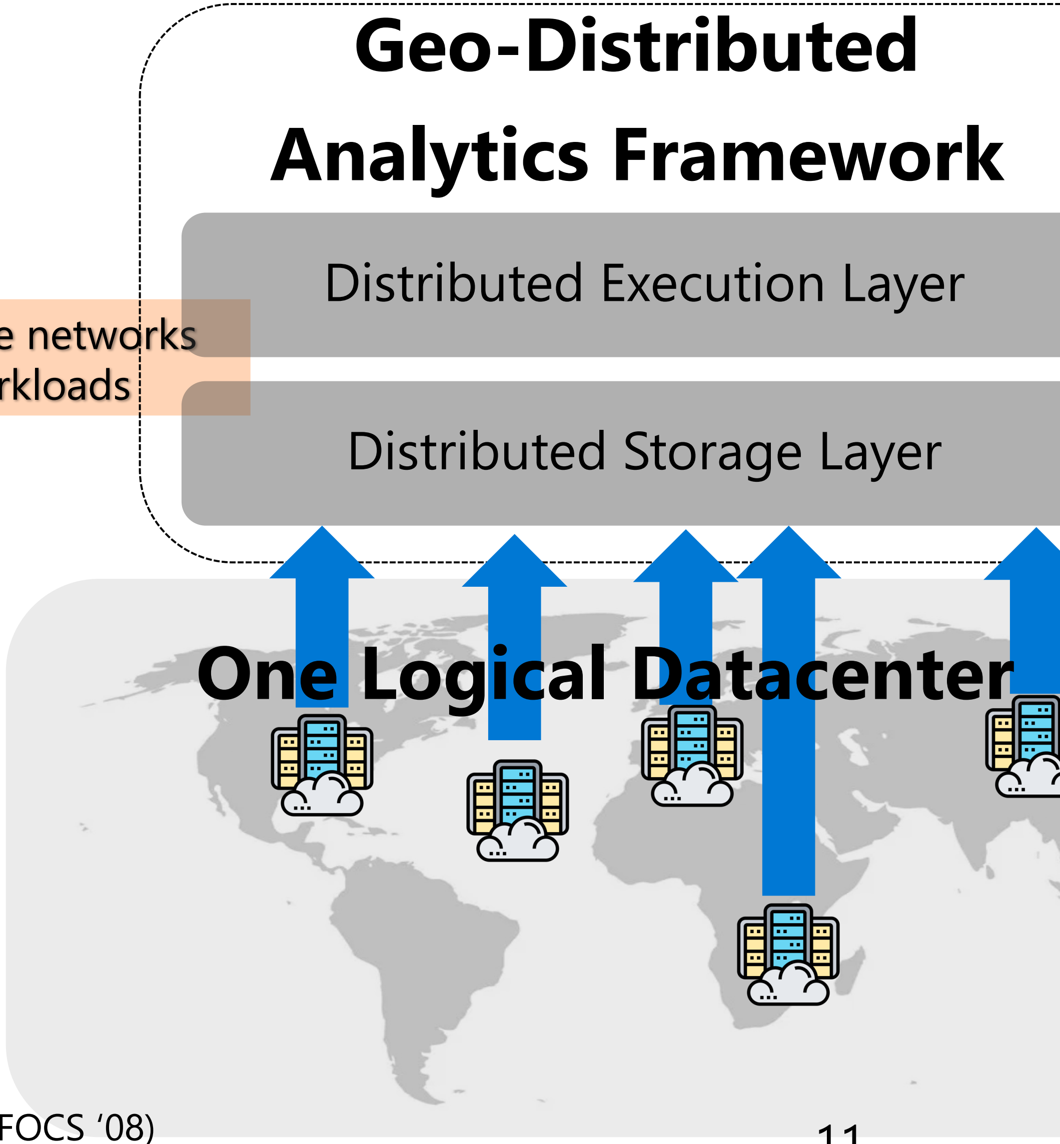
Existing ILP-based Geo-Distributed Systems are Static

1. Computing the best query execution plan with task placement and schedules is *NP-hard**
2. Existing works apply slow *ILPs*, in a *greedy* manner
3. Dynamic re-optimization is 25x slower than conventional approaches for handling temporal variations

Suitable for stable networks and batch workloads

Limited optimization capabilities

Requires checkpoint & replay of continuous operators



*Mastrolilli et. al: (Acyclic) job shops are hard to approximate (FOCS '08)
*Monaldo et. al: Improved bounds for flow shop scheduling (ICALP '09)

image: Flaticon.com

Comparison on Different Systems

	Centralization through Degradation	ILP-based Geo-distributed Systems	*SWAN*
Real-time data processing	Dynamic (Stream)	Static (Batch)	*Dynamic (Stream)*
Logical geo-distributed cluster	X	O	*O*
Quick network optimization algorithm	O	X	*O*
Application-agnostic	X	O	*O*
Dynamic optimization	O	X	*O*

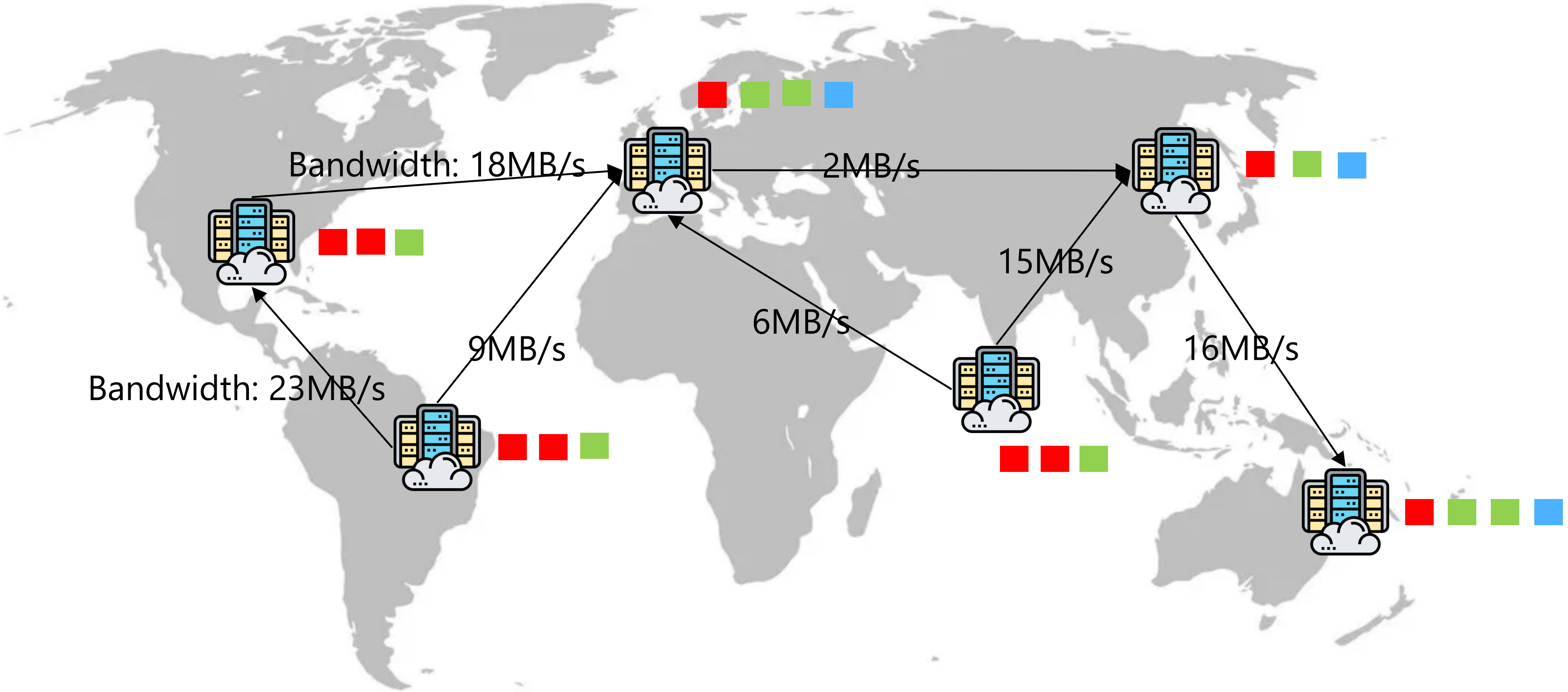
SWAN Design

Key Techniques and Effects

1. **Good heuristics over an expensive solver to perform timely dynamic optimizations**

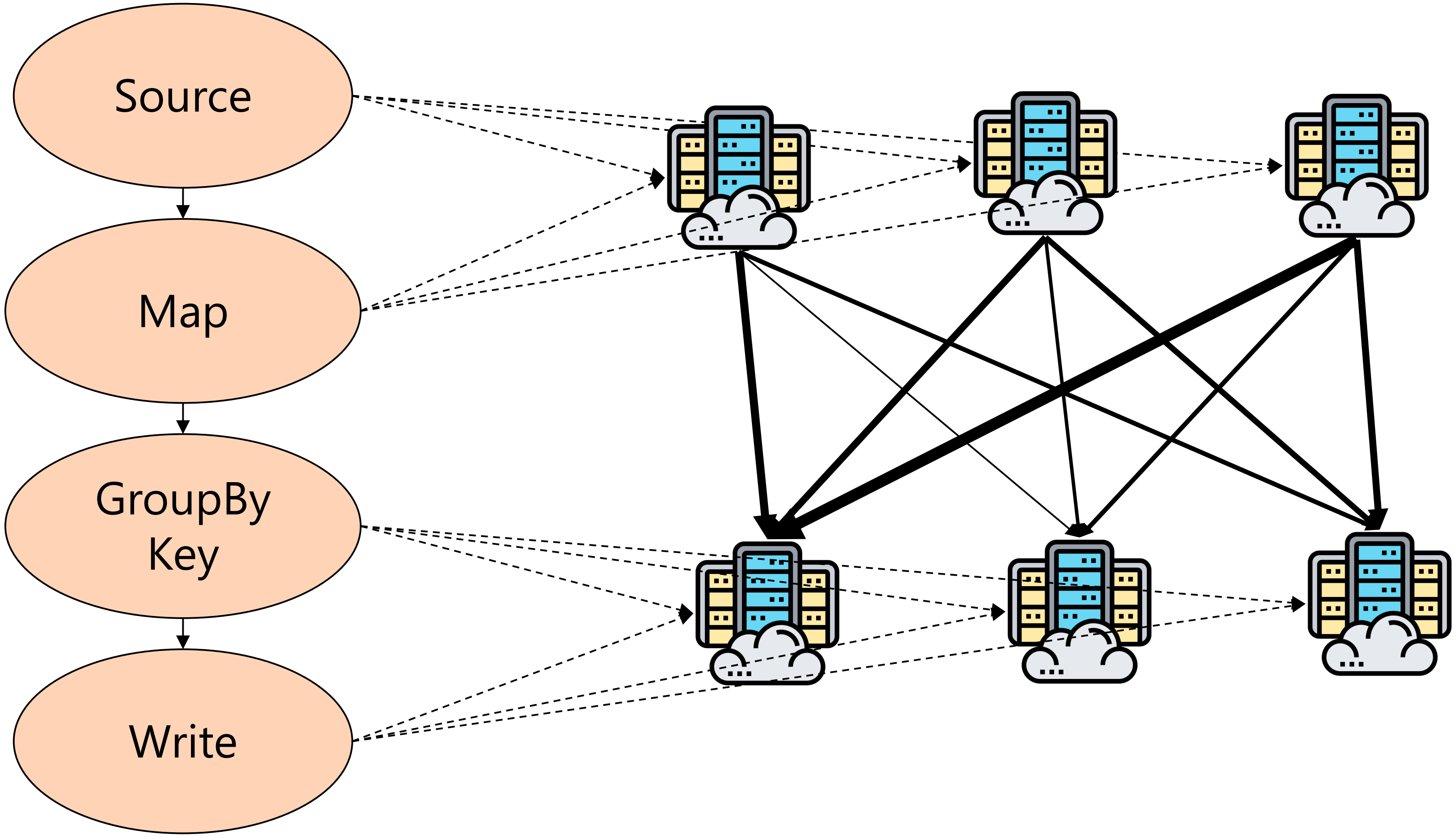
2. **Query rewriting to fully cover promising longer paths with higher bandwidths**

SWAN Heuristics



Requirement 1: Tasks should be scattered more or less evenly, to utilize the pool of CPU/memory resources and prevent network congestion

SWAN Heuristics



Requirement 2: Distribute the tasks proportional to upstream bandwidth capacities

SWAN Scheduling Algorithm

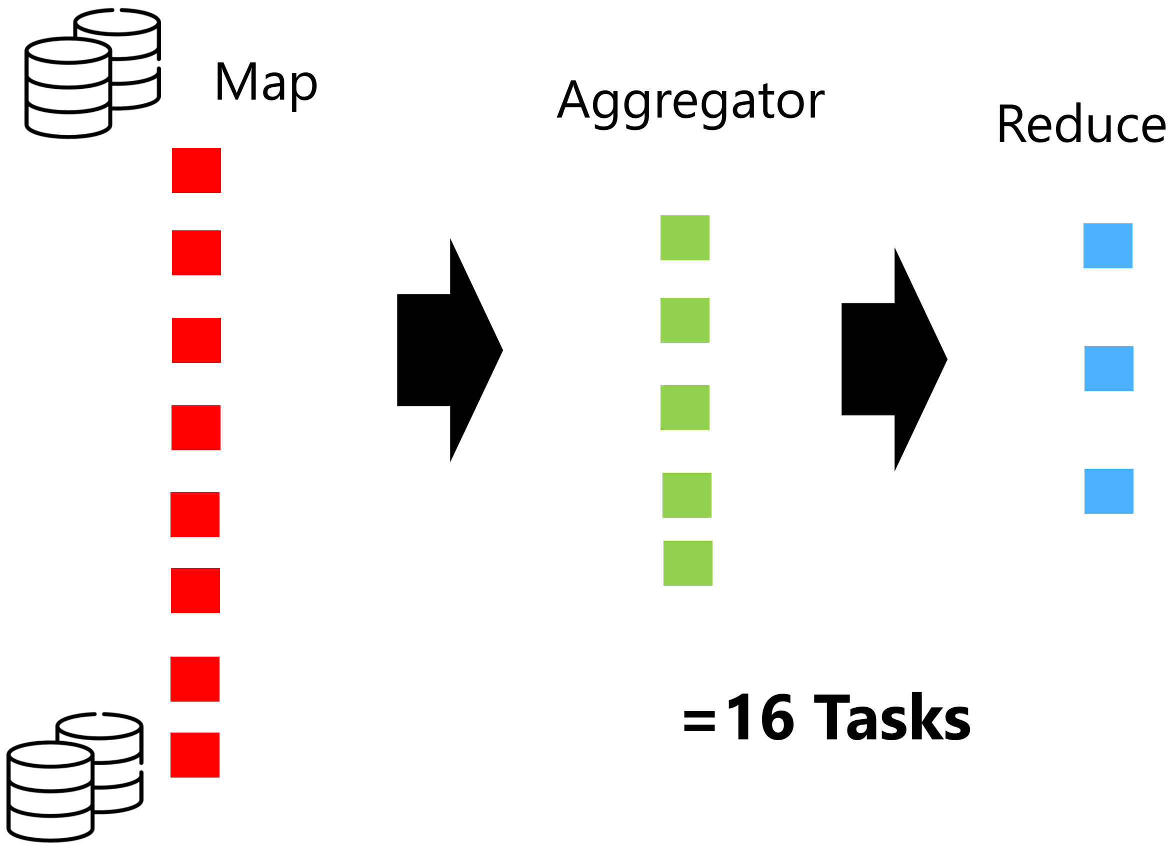
1. **Set an upper limit for the number of tasks for each site**

2. **Calculate the potential network cost for the additional task placed on a specific site.**

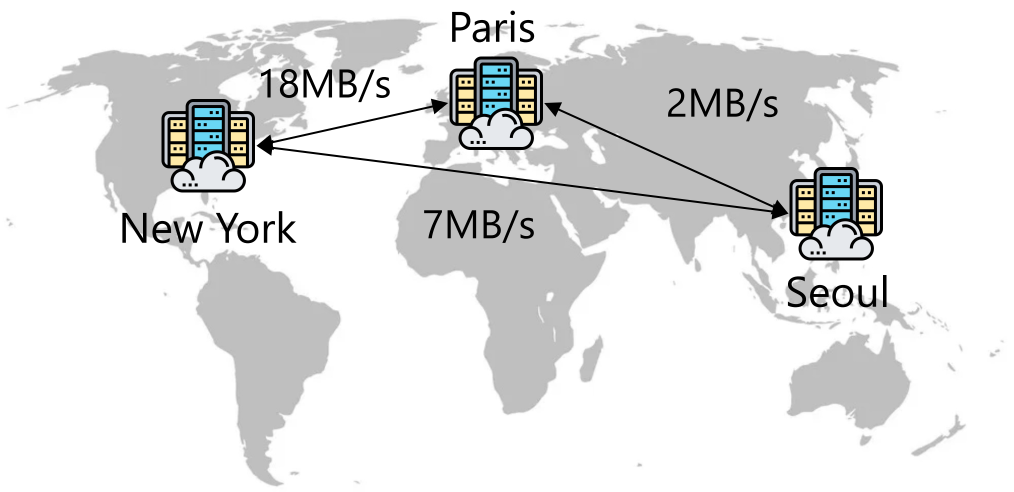
3. **Get the specific number of tasks to place on each site, based on the remaining task slots and the potential network cost**

SWAN Scheduling Algorithm Example

Physical Plan



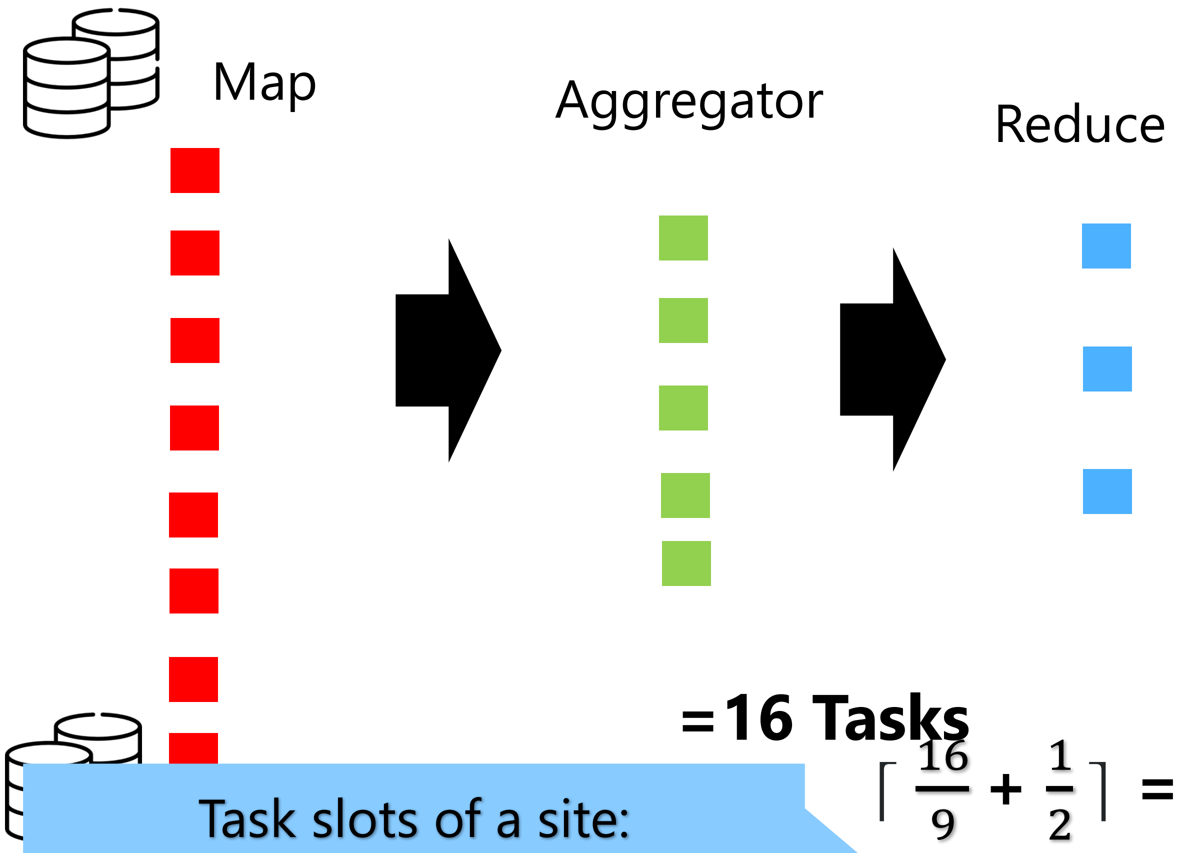
=16 Tasks

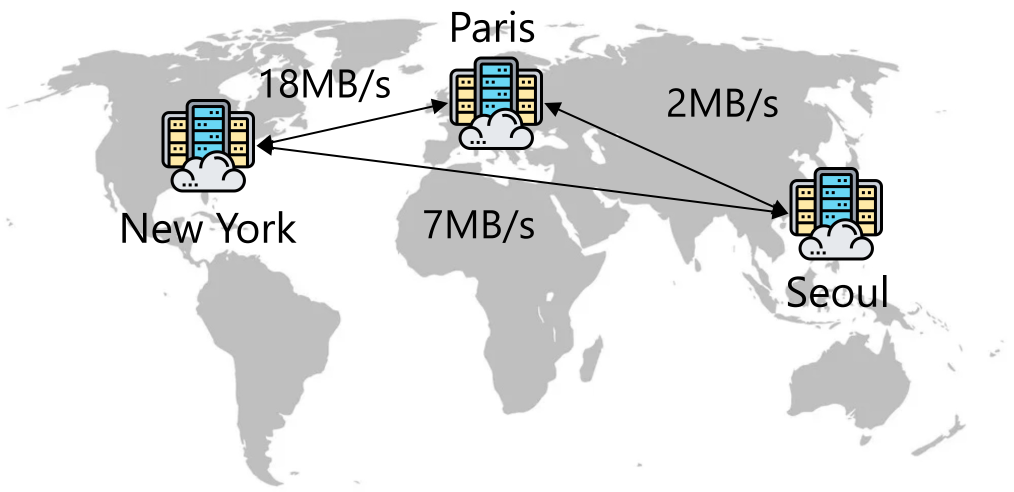


= 3 sites with 3 nodes each

SWAN Scheduling Algorithm Example

Physical Plan





= 3 sites with 3 nodes each

=16 Tasks

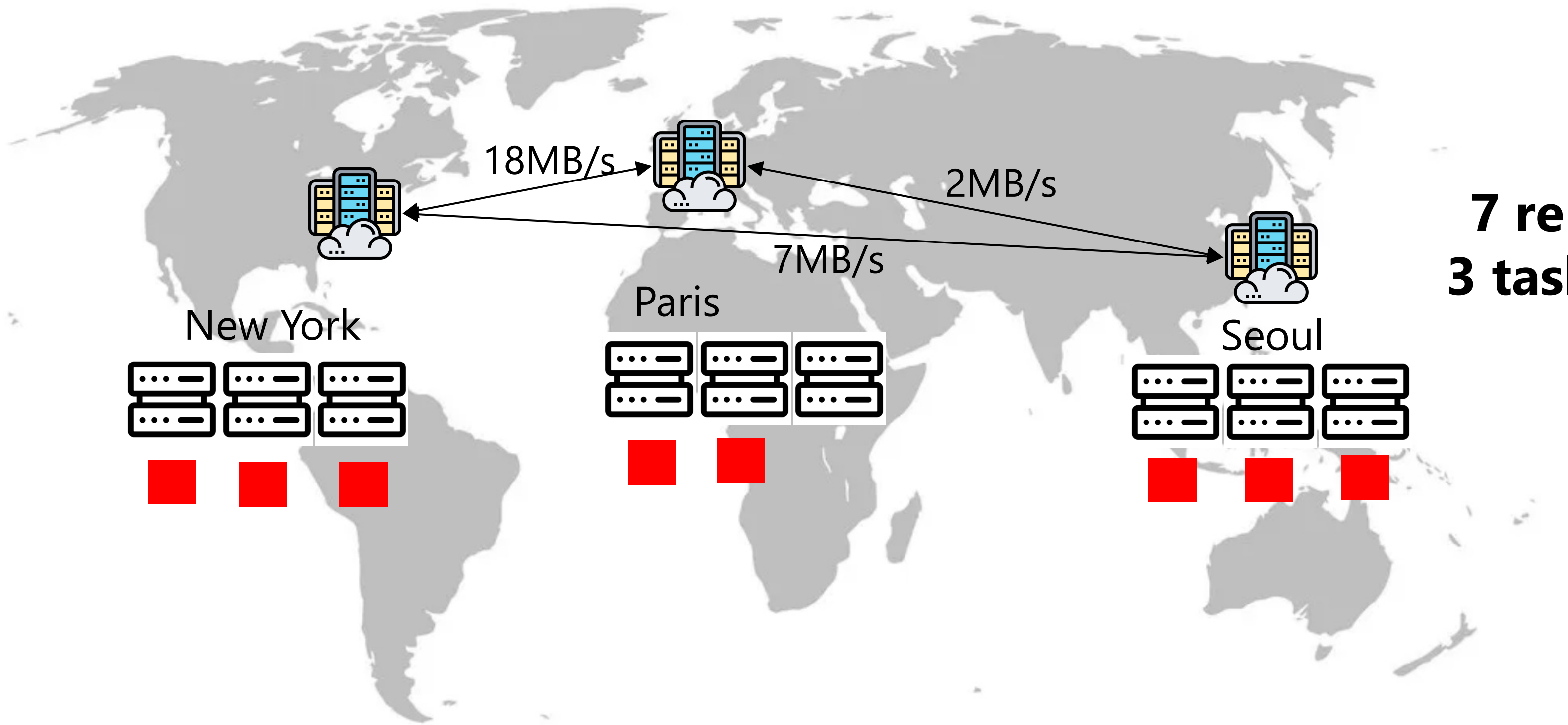
Task slots of a site:

$$\sum_{node \in site} \left\lceil \frac{\sum tasks\_count}{\sum node\_count} + \frac{1}{2} \right\rceil$$

$\left\lceil \frac{16}{9} + \frac{1}{2} \right\rceil$ **= 3 task slots per node**

$3 \times 3 =$ **9 task slots per site**

SWAN Scheduling Algorithm Example



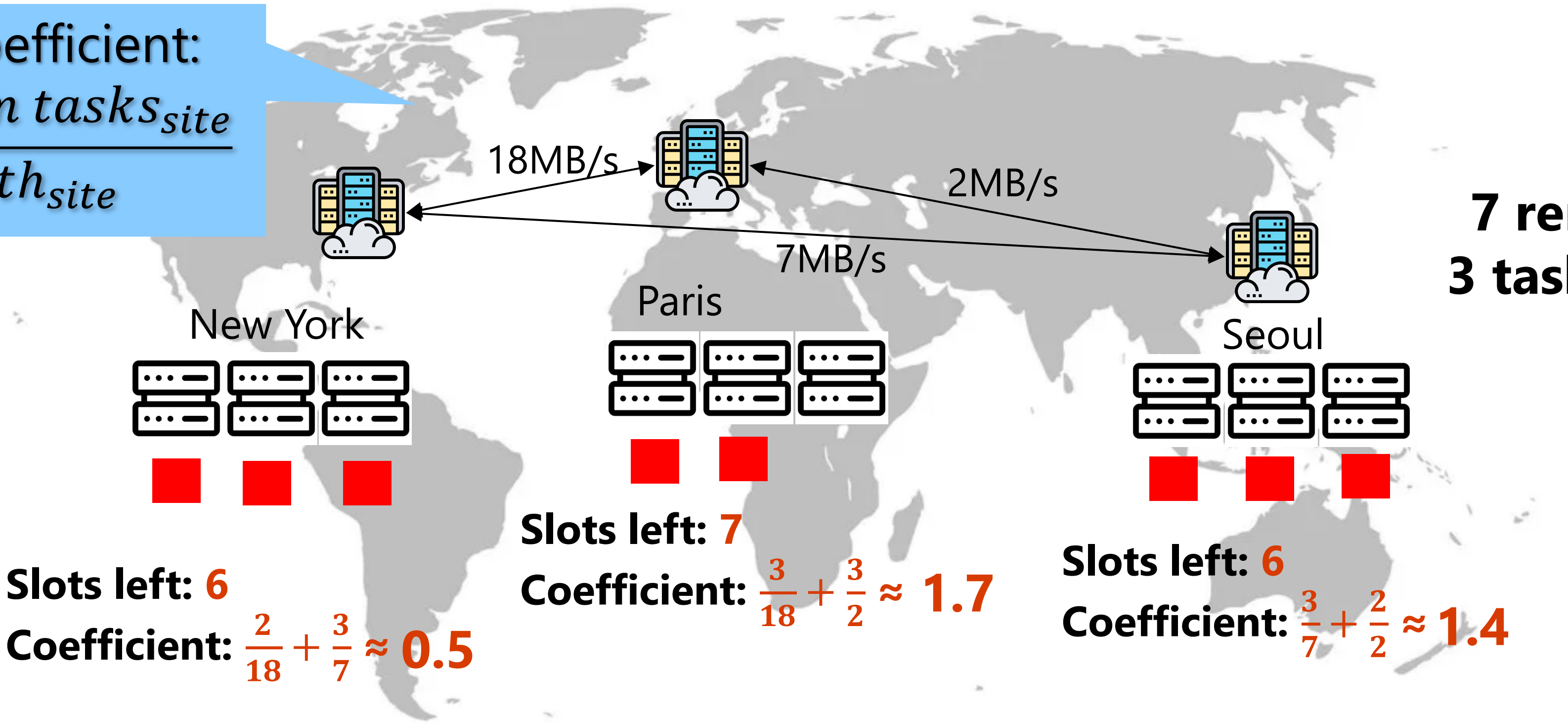
7 remaining tasks,
3 task slots per node

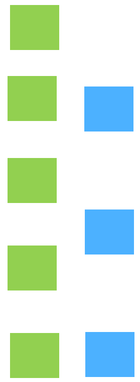
Data sources are distributed across the globe

SWAN Scheduling Algorithm Example

Network cost coefficient:

$$\sum \frac{\#\ of\ upstream\ tasks_{site}}{bandwidth_{site}}$$





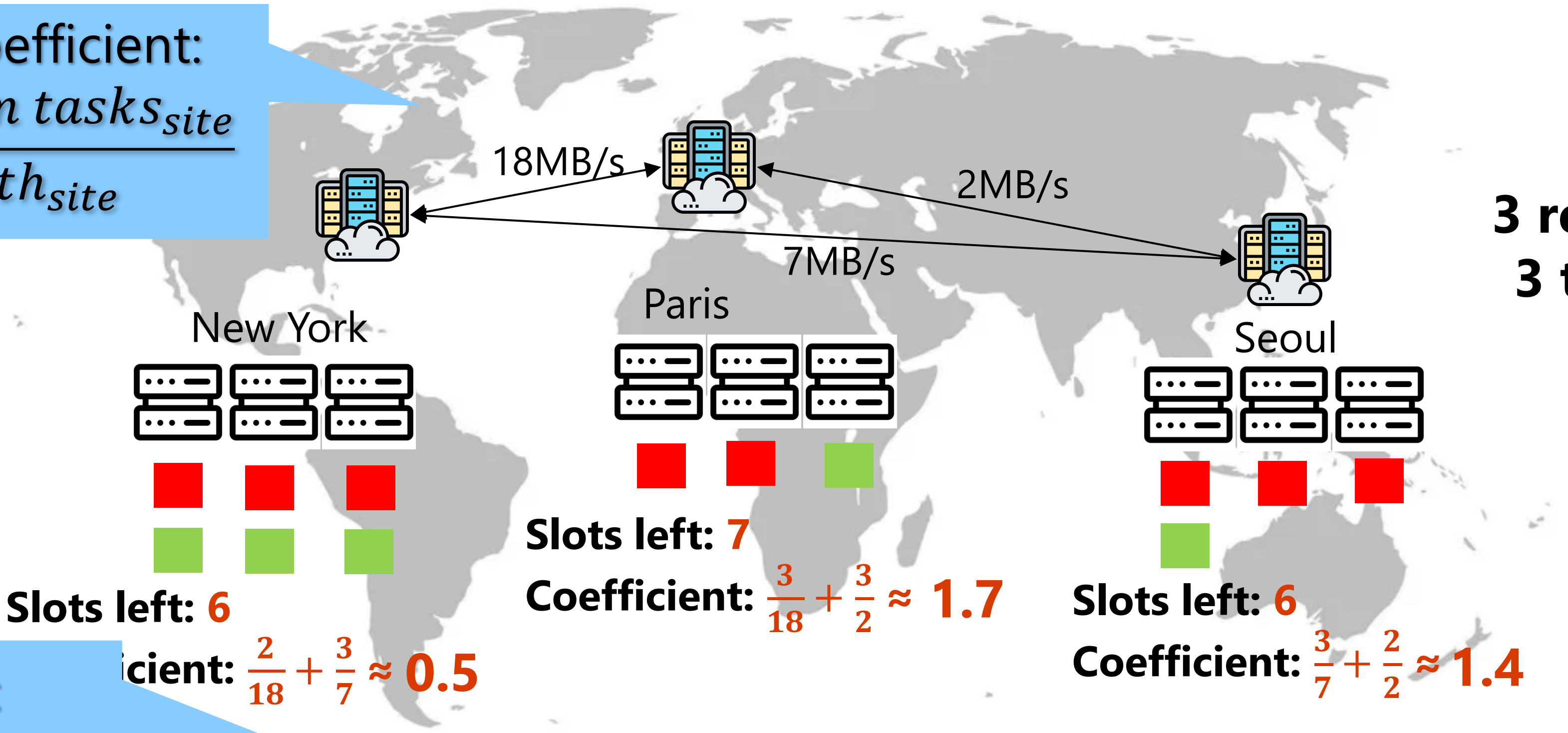
7 remaining tasks,
3 task slots per node

Calculate the distance coefficient and remaining slots for each stage and site

SWAN Scheduling Algorithm Example

Network cost coefficient:

$$\sum \frac{\#\ of\ upstream\ tasks_{site}}{bandwidth_{site}}$$



3 remaining tasks,
3 tasks per node

Distribution factor:

$$\frac{task\_slots\_left_{site}}{cost\_coefficient_{site}}$$

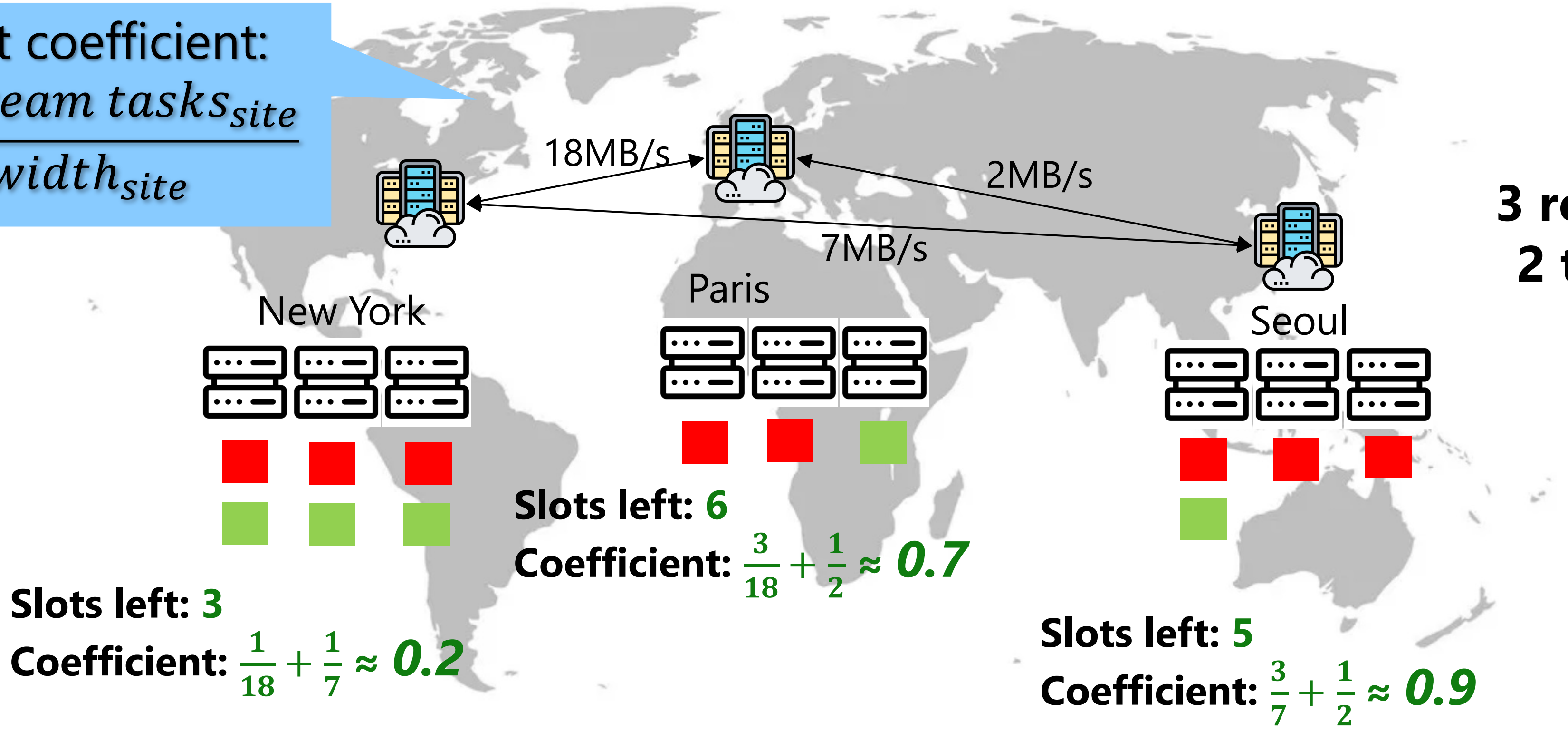
Distribution goal ≈ $\frac{6}{0.5} : \frac{7}{1.7} : \frac{6}{1.4}$ → *3:1:1*

Place tasks on sites where the distribution ratio is most proportional to [remaining slots / network cost coefficient]

SWAN Scheduling Algorithm Example

Network cost coefficient:

$$\sum \frac{\#\ of\ upstream\ tasks_{site}}{bandwidth_{site}}$$



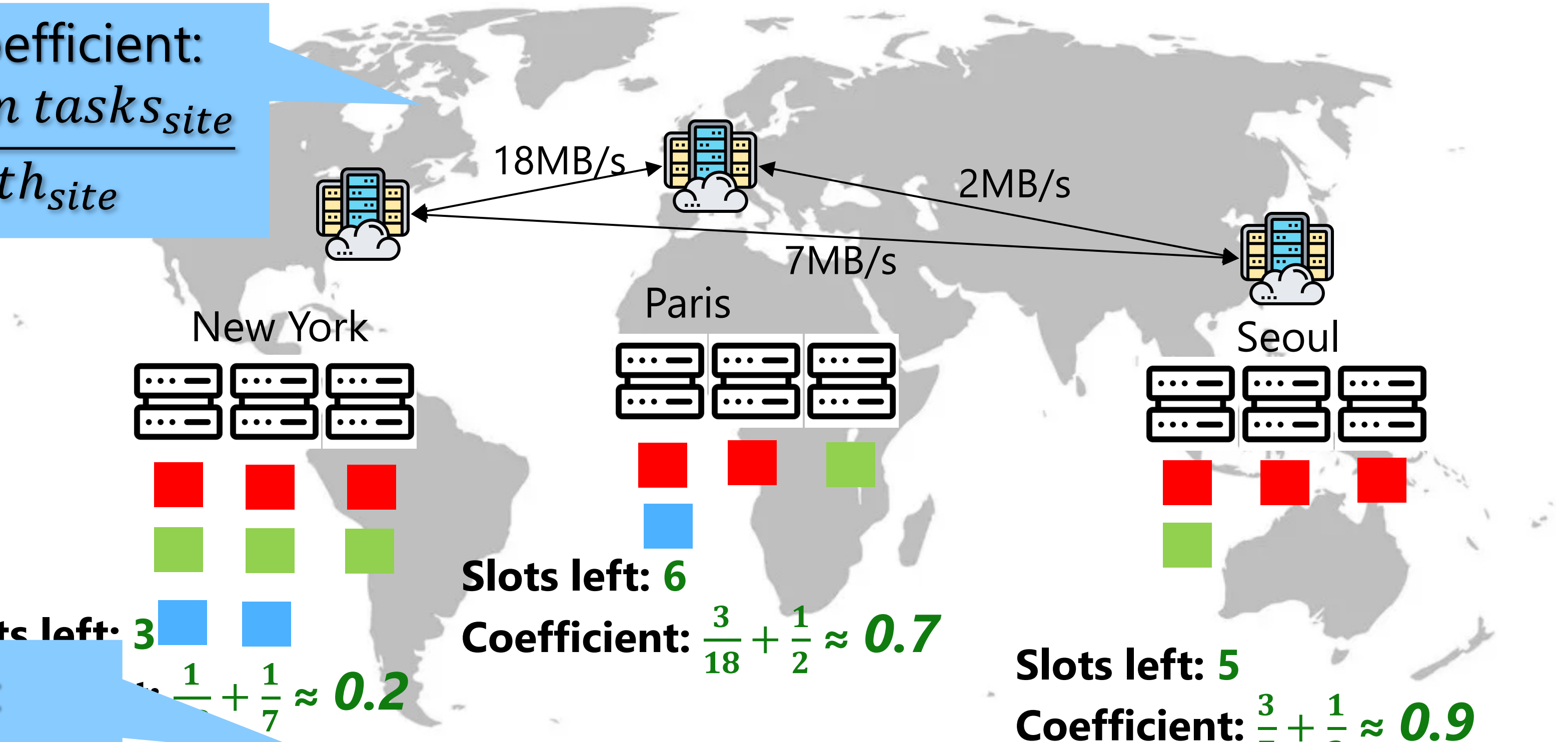
3 remaining tasks, 2 tasks per node

Place tasks on sites where the distribution ratio is most proportional to [remaining slots / network cost coefficient]

SWAN Scheduling Algorithm Example

Network cost coefficient:

$$\sum \frac{\#\ of\ upstream\ tasks_{site}}{bandwidth_{site}}$$



Distribution factor:

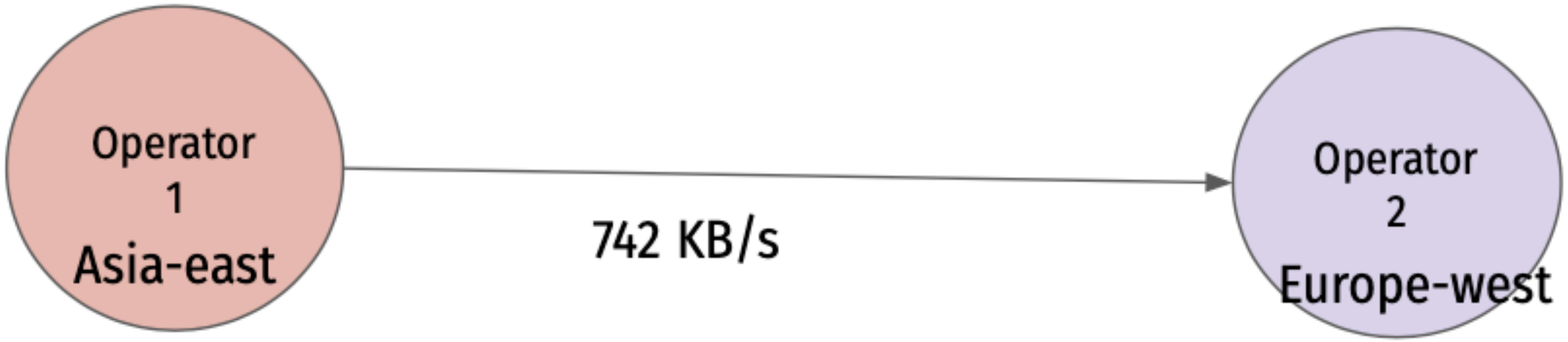
$$\frac{task\_slots\_left_{site}}{cost\_coefficient_{site}}$$

Distribution goal $\approx \frac{3}{0.2} : \frac{6}{0.7} : \frac{5}{0.9} \rightarrow$ ***2:1:0***

Place tasks on sites where the distribution ratio is most proportional to [remaining slots / network cost coefficient]

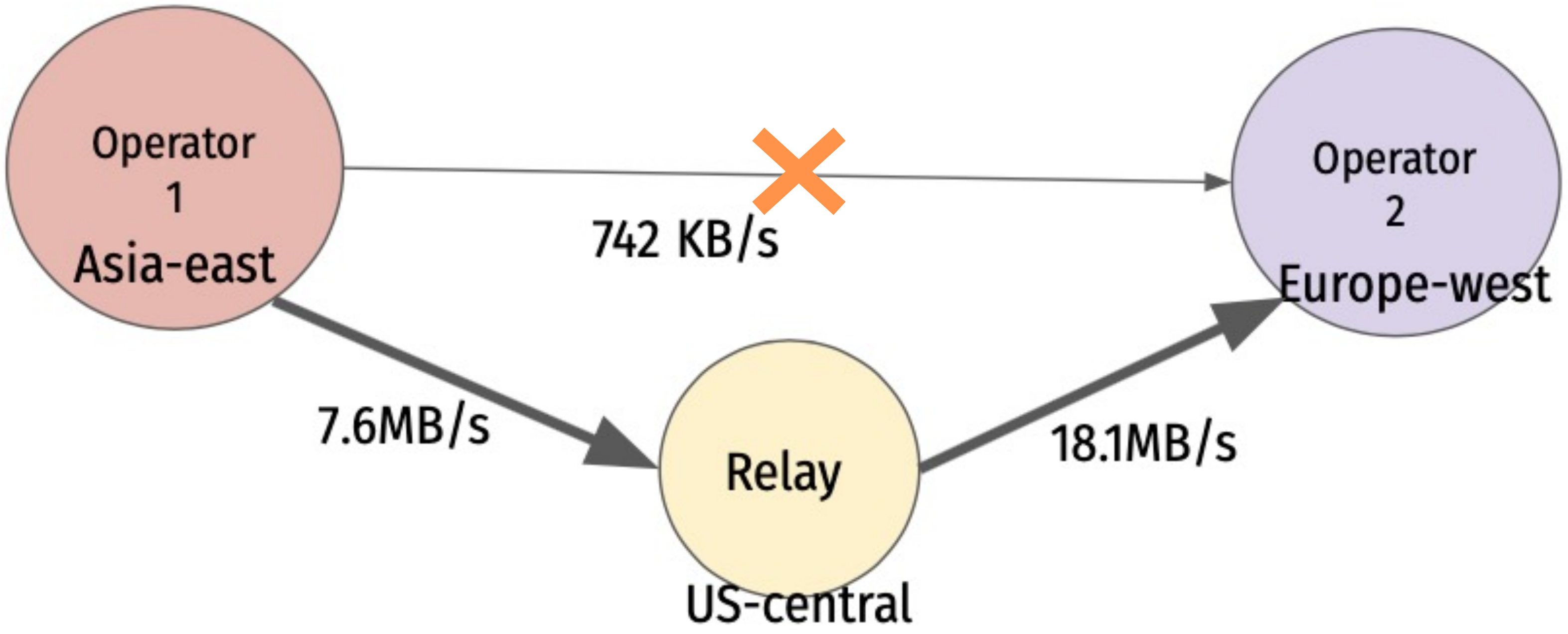
Providing More Flexibility with Relay Operators

Region 1	Region 2	Average Bandwidth
Asia-east	Europe-west	742KB/s
Asia-east	US-central	7.6MB/s
Europe-west	US-central	18.1MB/s



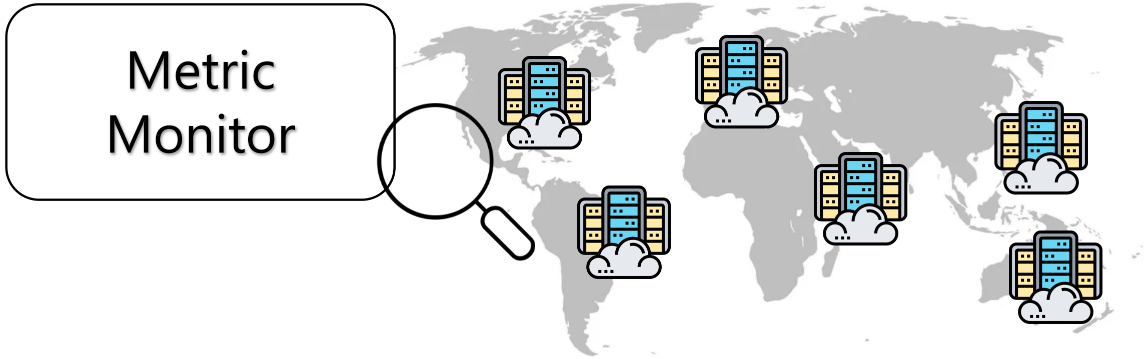
Providing More Flexibility with Relay Operators

Region 1	Region 2	Average Bandwidth
Asia-east	Europe-west	742KB/s
Asia-east	US-central	7.6MB/s
Europe-west	US-central	18.1MB/s



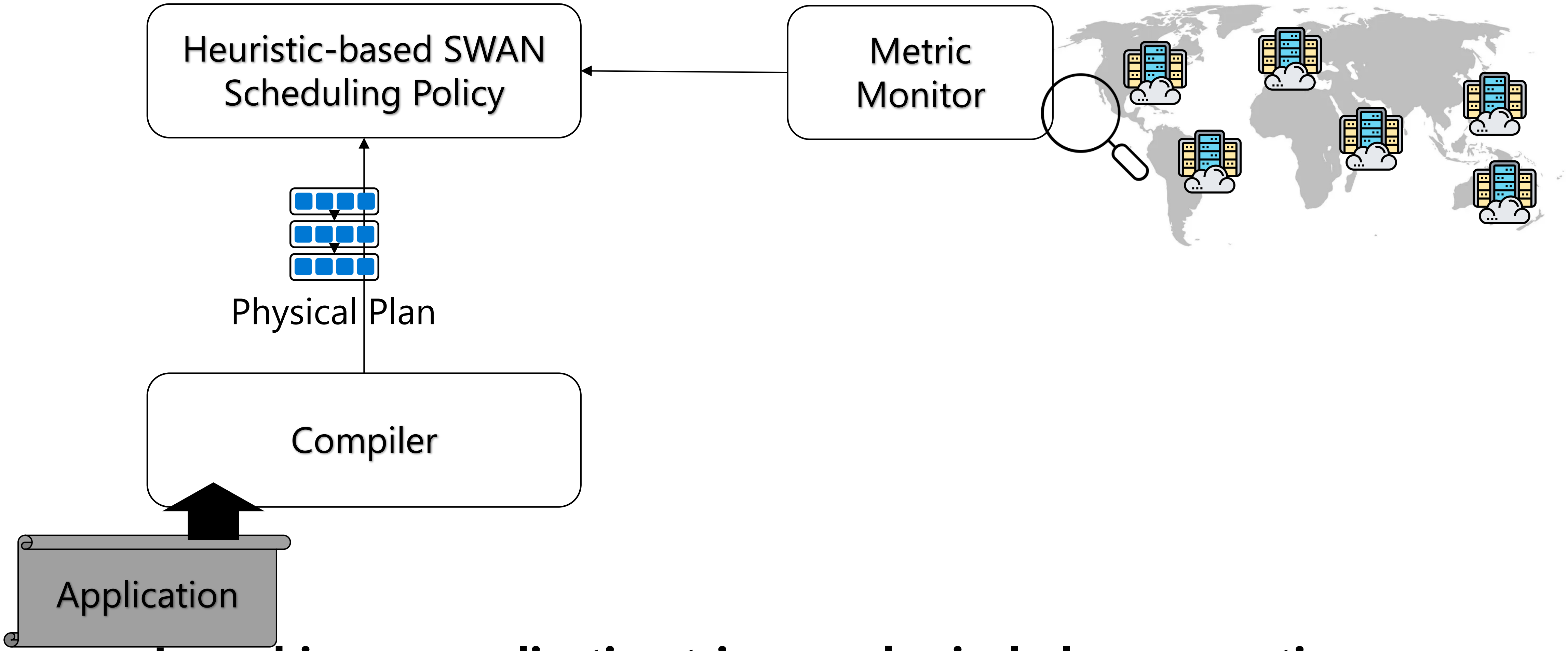
SWAN Implementation

SWAN Implementation



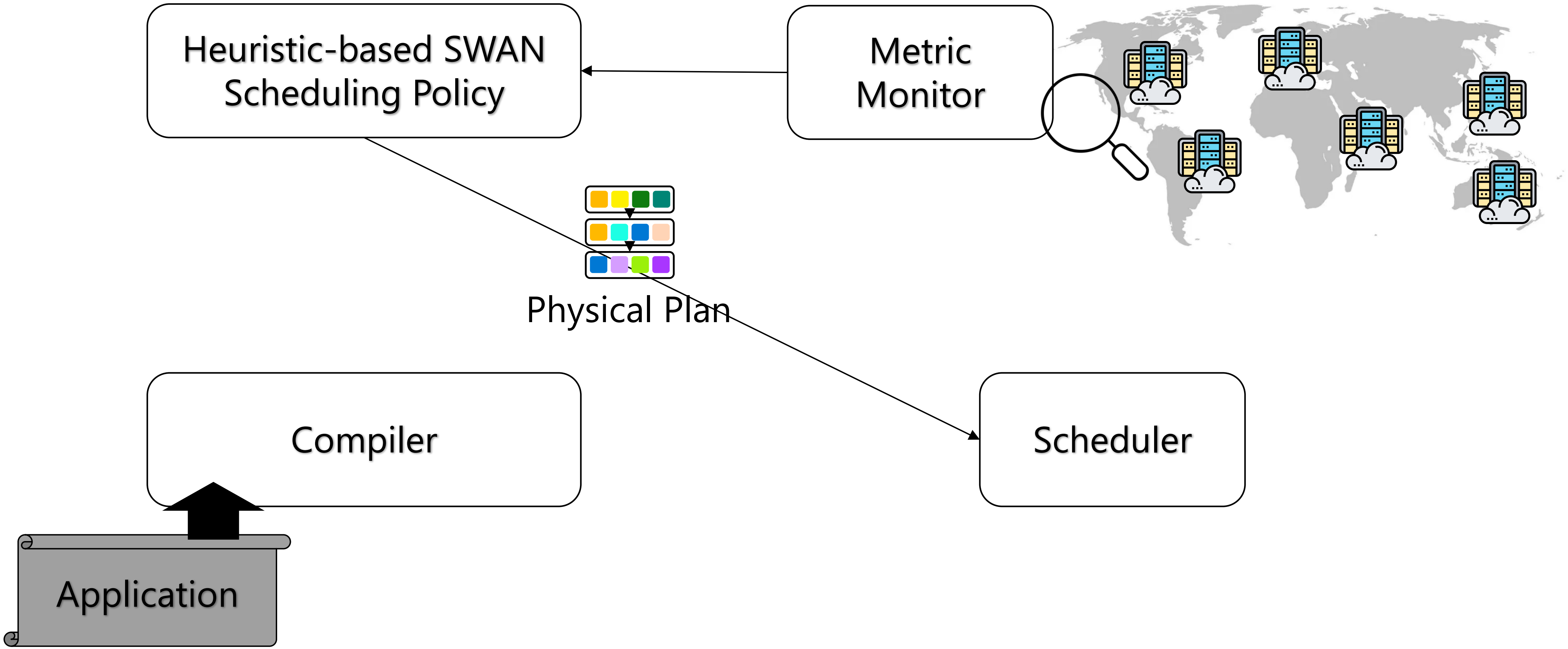
Metric monitor keeps track of the global cluster networks asynchronously

SWAN Implementation



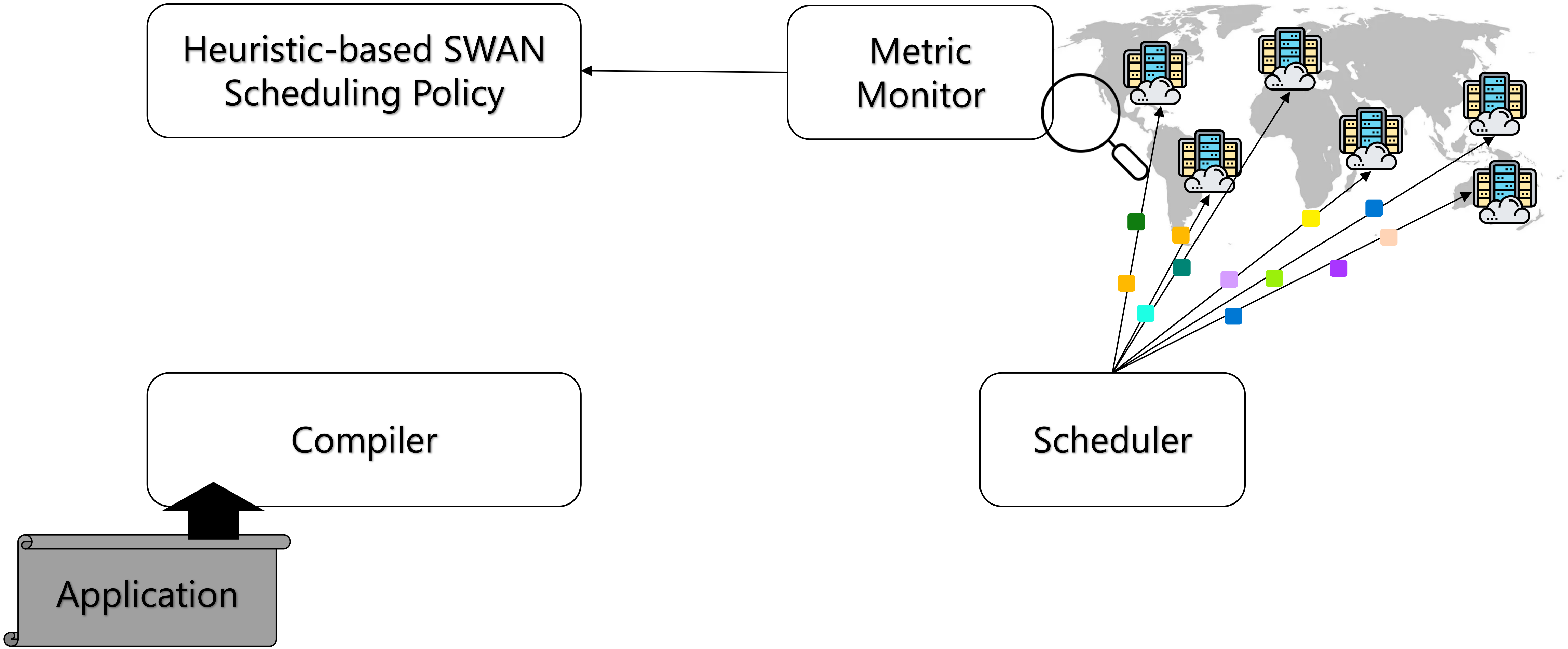
Launching an application triggers physical plan generation, which is submitted to the scheduling policy

SWAN Implementation



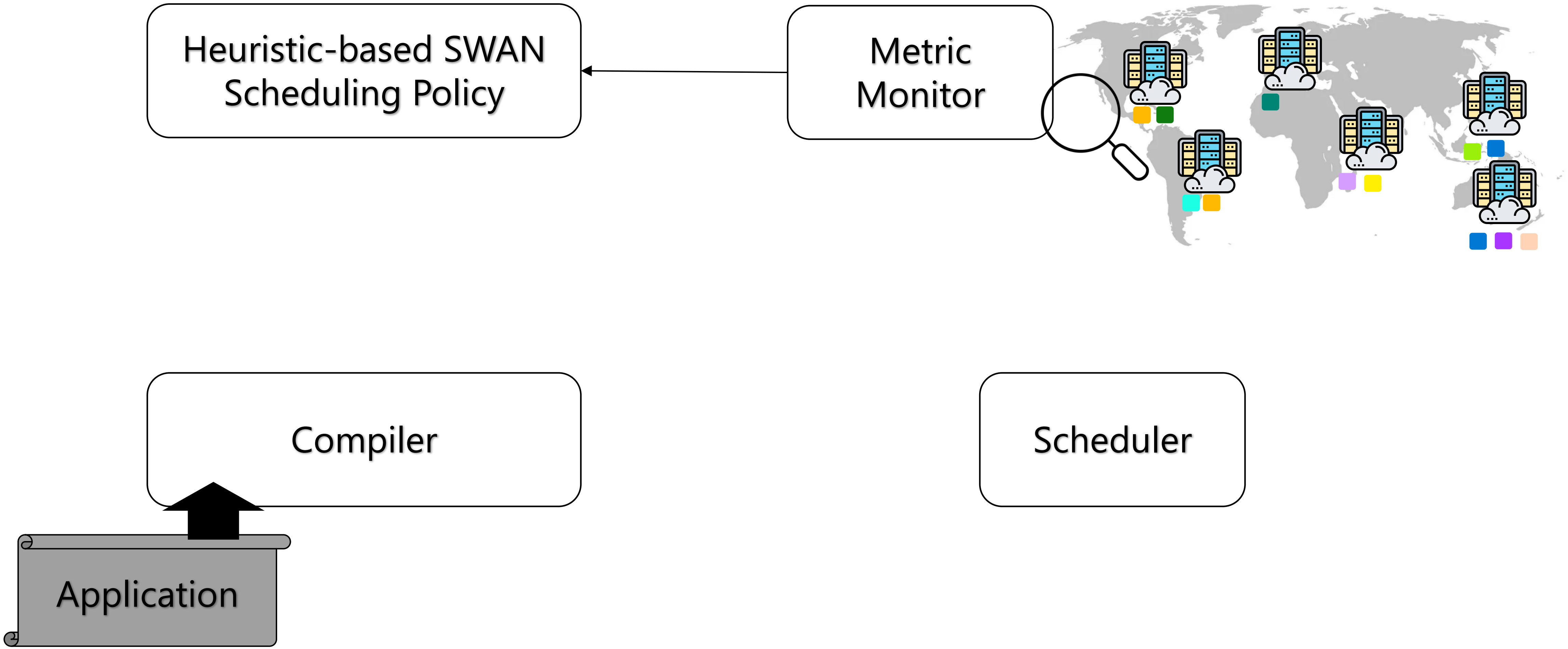
The scheduling policy allocates each task to a node and submits the plan to the scheduler

SWAN Implementation



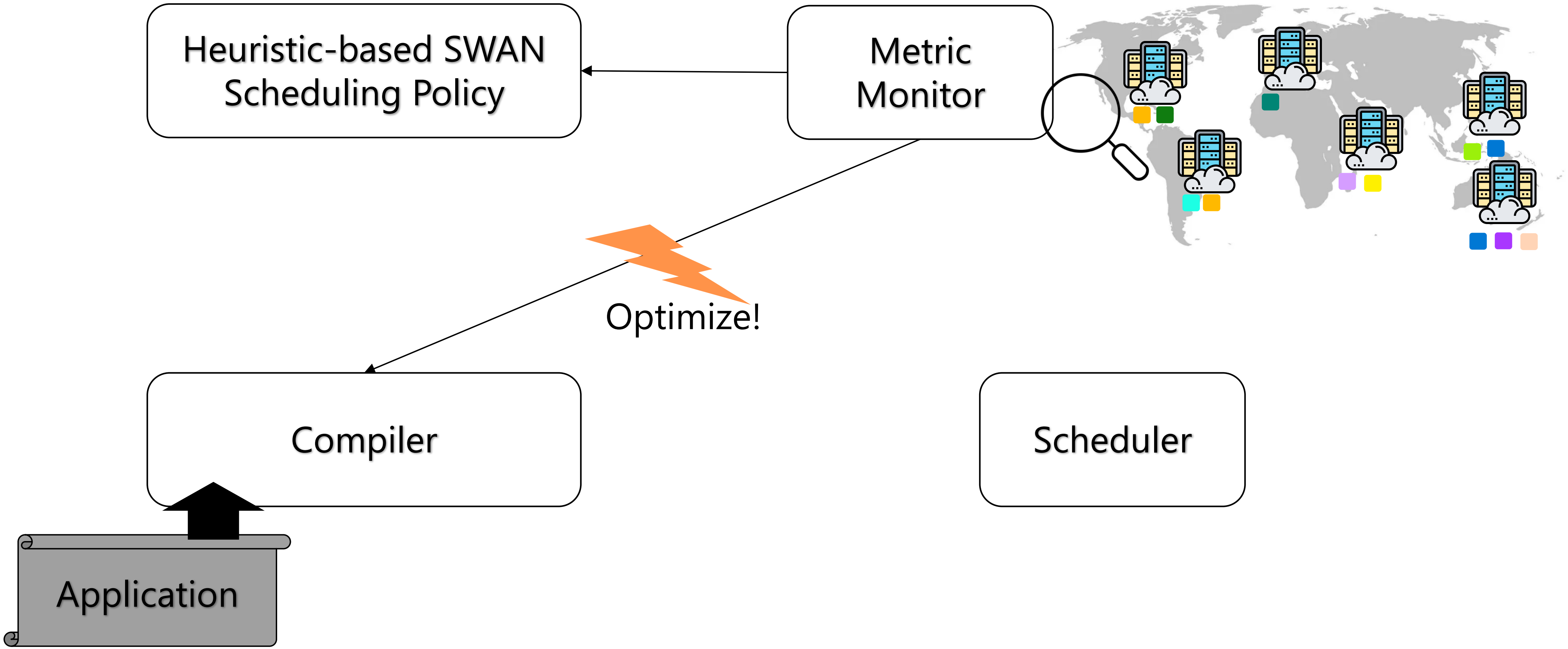
The scheduler distributes tasks to executors according to the plan

SWAN Implementation



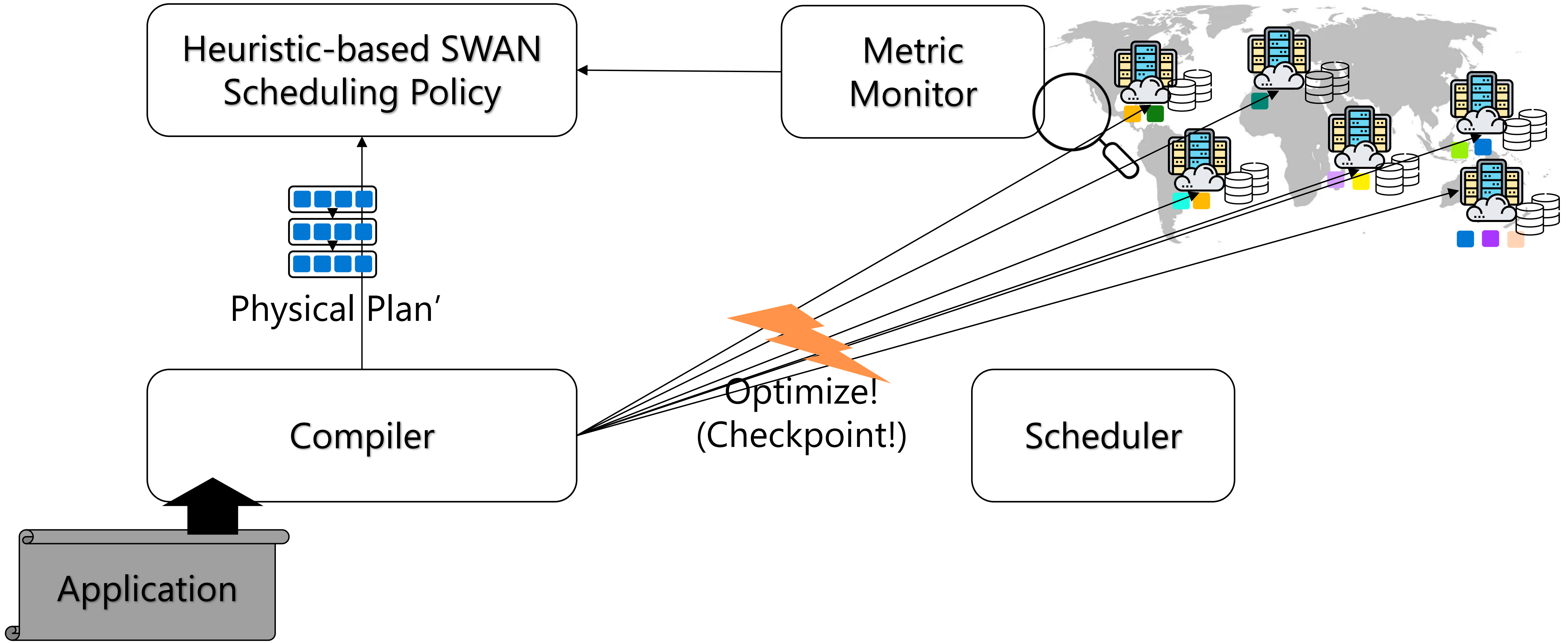
The scheduler distributes tasks to executors according to the plan

SWAN Implementation



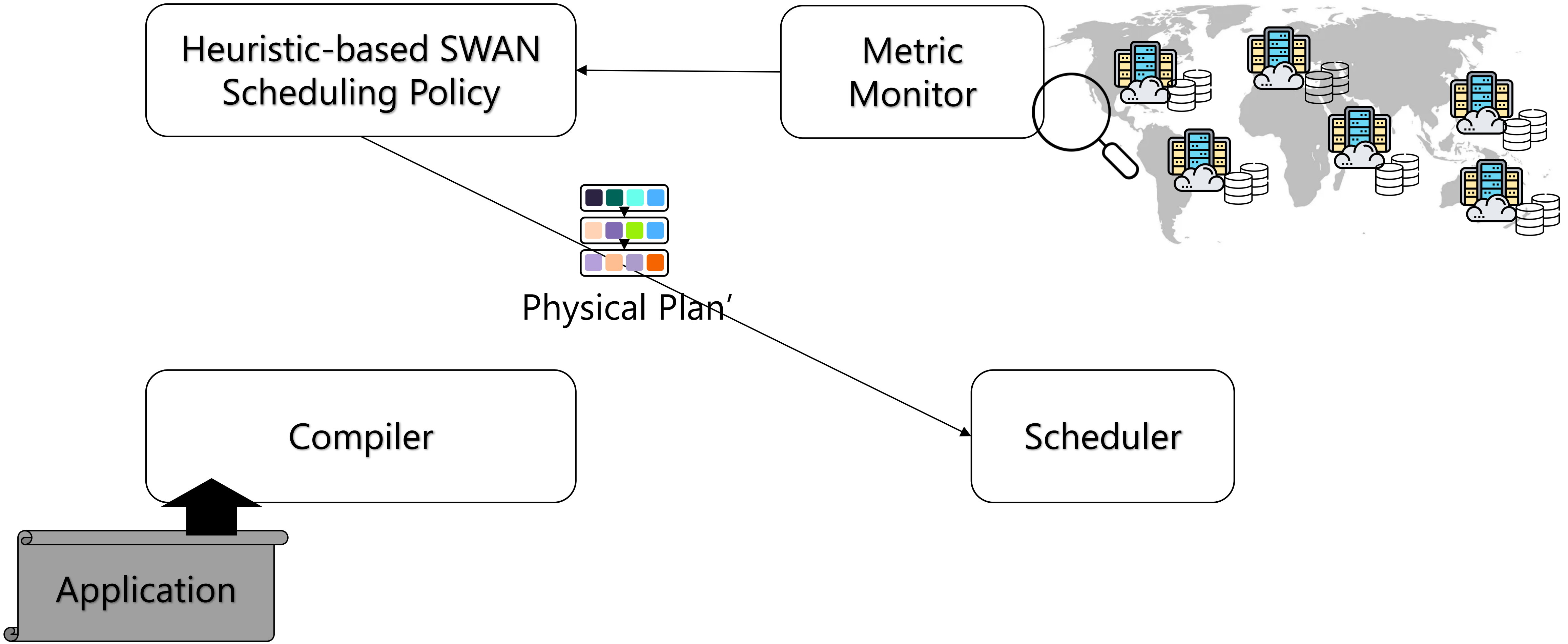
When metrics call for a change (latency rise, network drop, etc.) metric monitor calls for an optimization on the compiler

SWAN Implementation



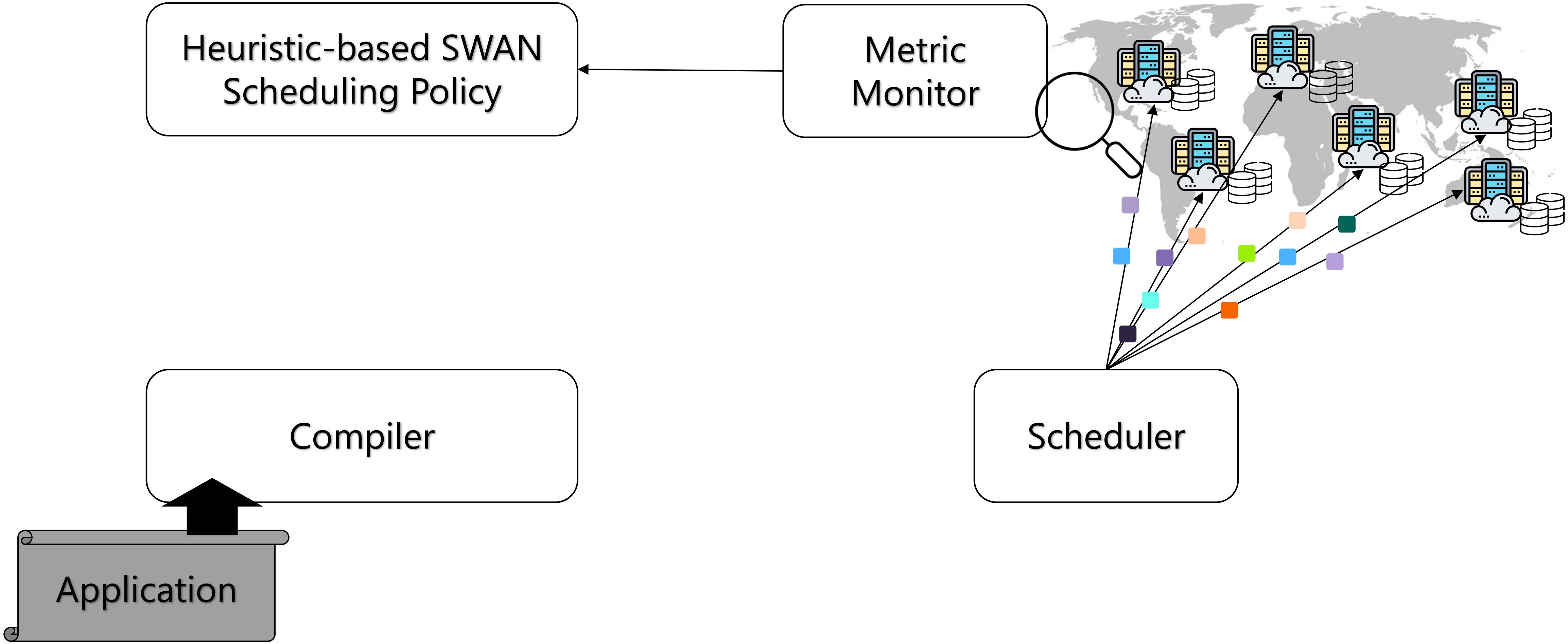
The compiler sends a watermark that tells all nodes to checkpoint their tasks

SWAN Implementation



The physical plan is optimized and re-submitted to the scheduler

SWAN Implementation

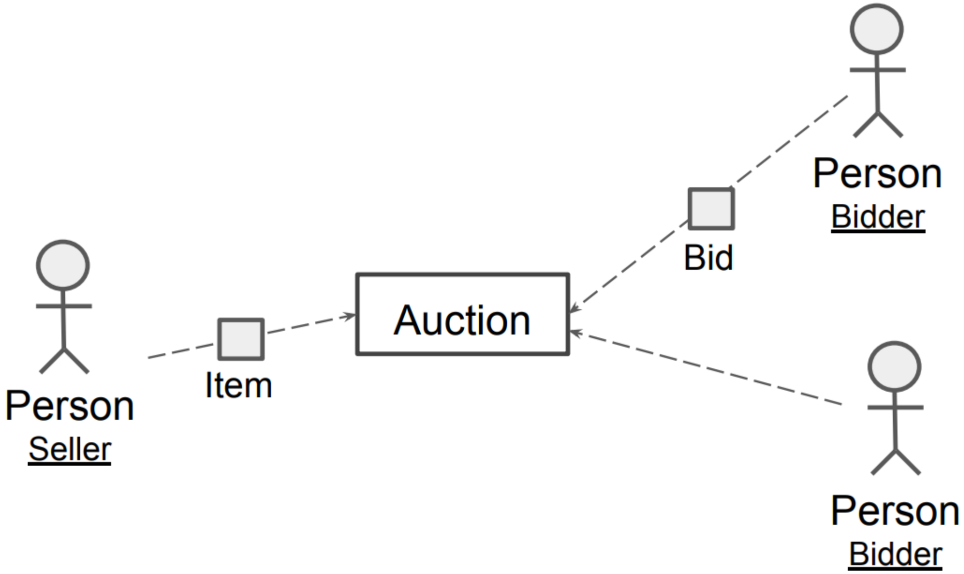


Tasks are migrated according to the new schedule plan and executes from the checkpointed state

Evaluation

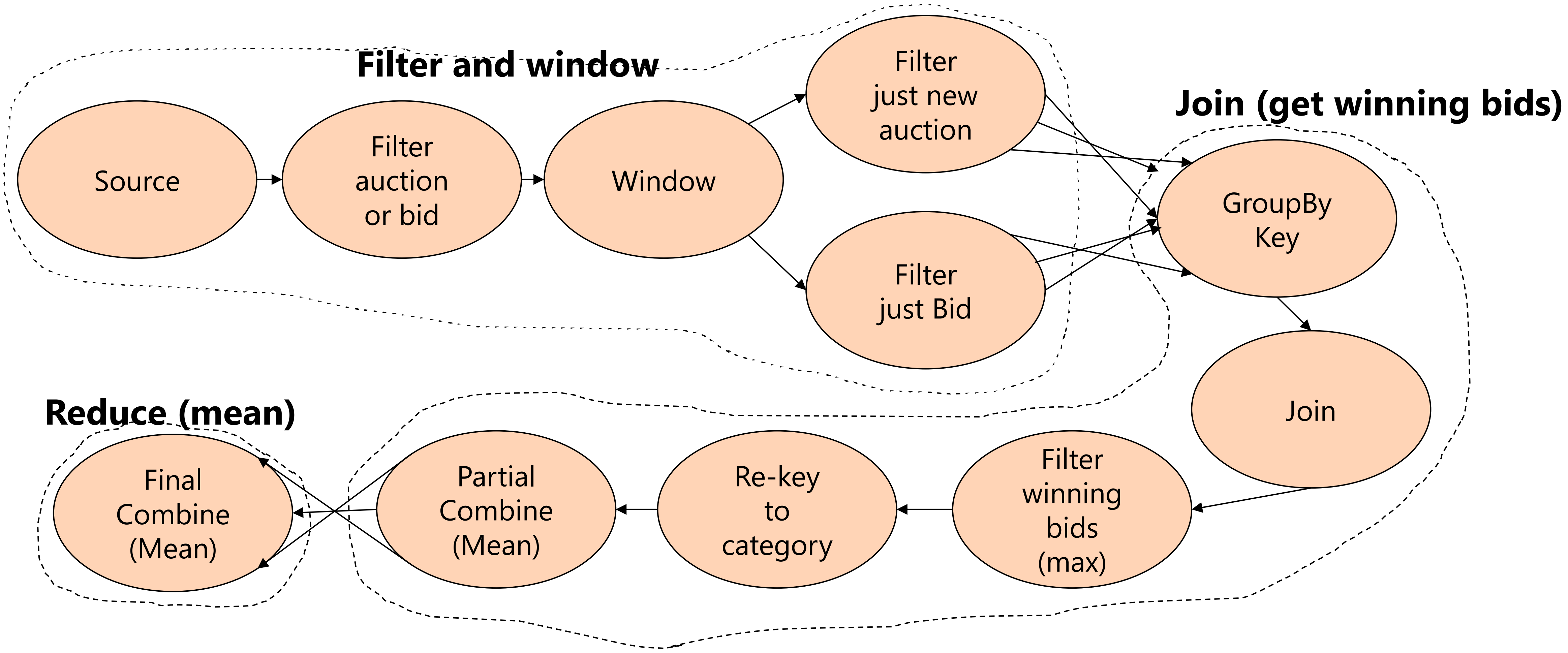
Evaluation Results

- GCP Cluster of *16 nodes* across *8 regions* over *3 continents*
  - e2-standard-4 (4vCPUs, 16GB Memory)
  - Asia: Taiwan, Mumbai
  - Europe: Finland, Belgium, Netherlands
  - N. America: Iowa, South Carolina, Oregon



- NEXMark Benchmark Suite
  - A suite of pipelines, provided by Apache Beam, representing an online auction system
  - Following examples show a case in *Query 4 (average price per category),* which illustrates complex *join* and *aggregation*, involving the most shuffle operations

Evaluation Results: Query 4 Execution DAG

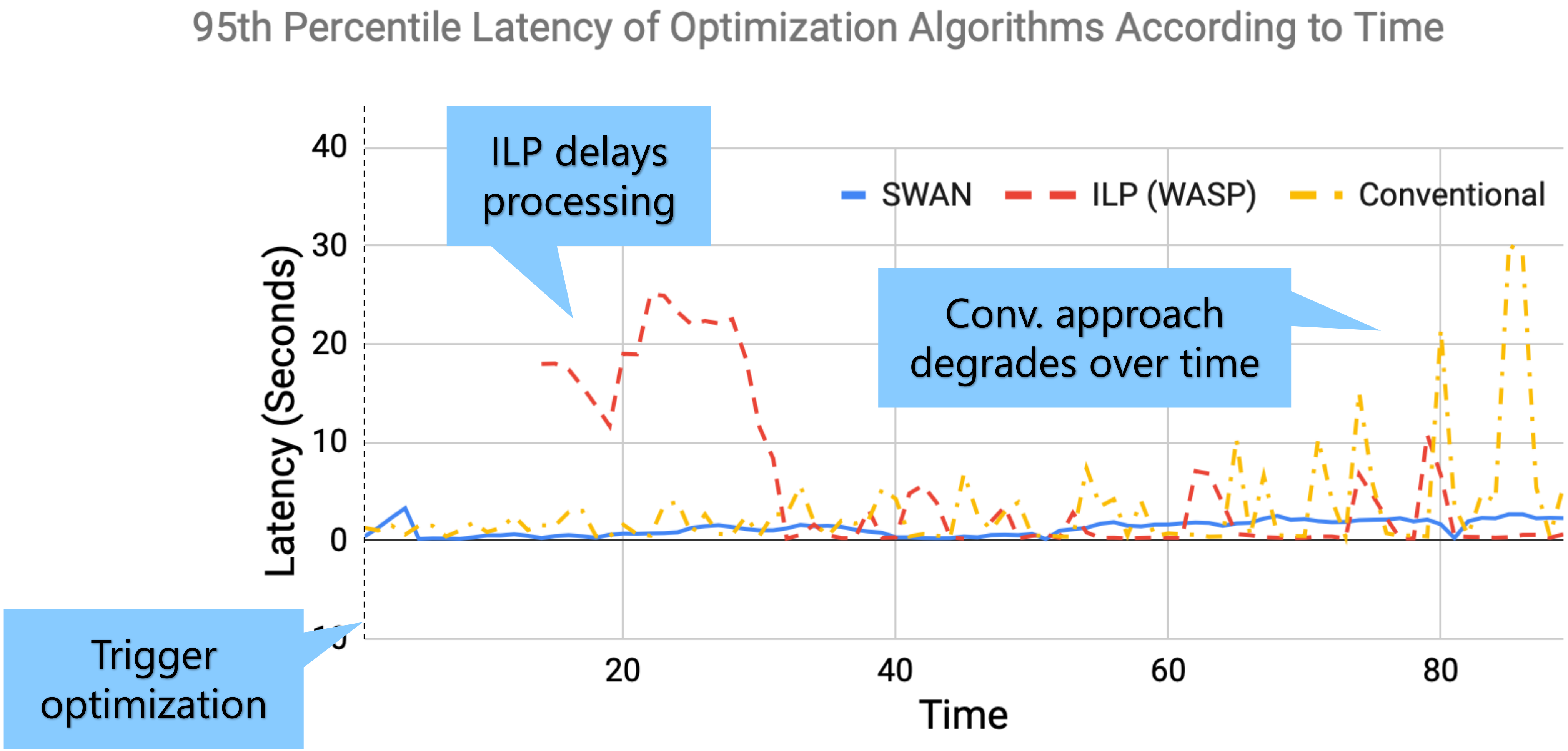


Query 4: average price per category

Evaluation Results: Query 4 Average Price for Category

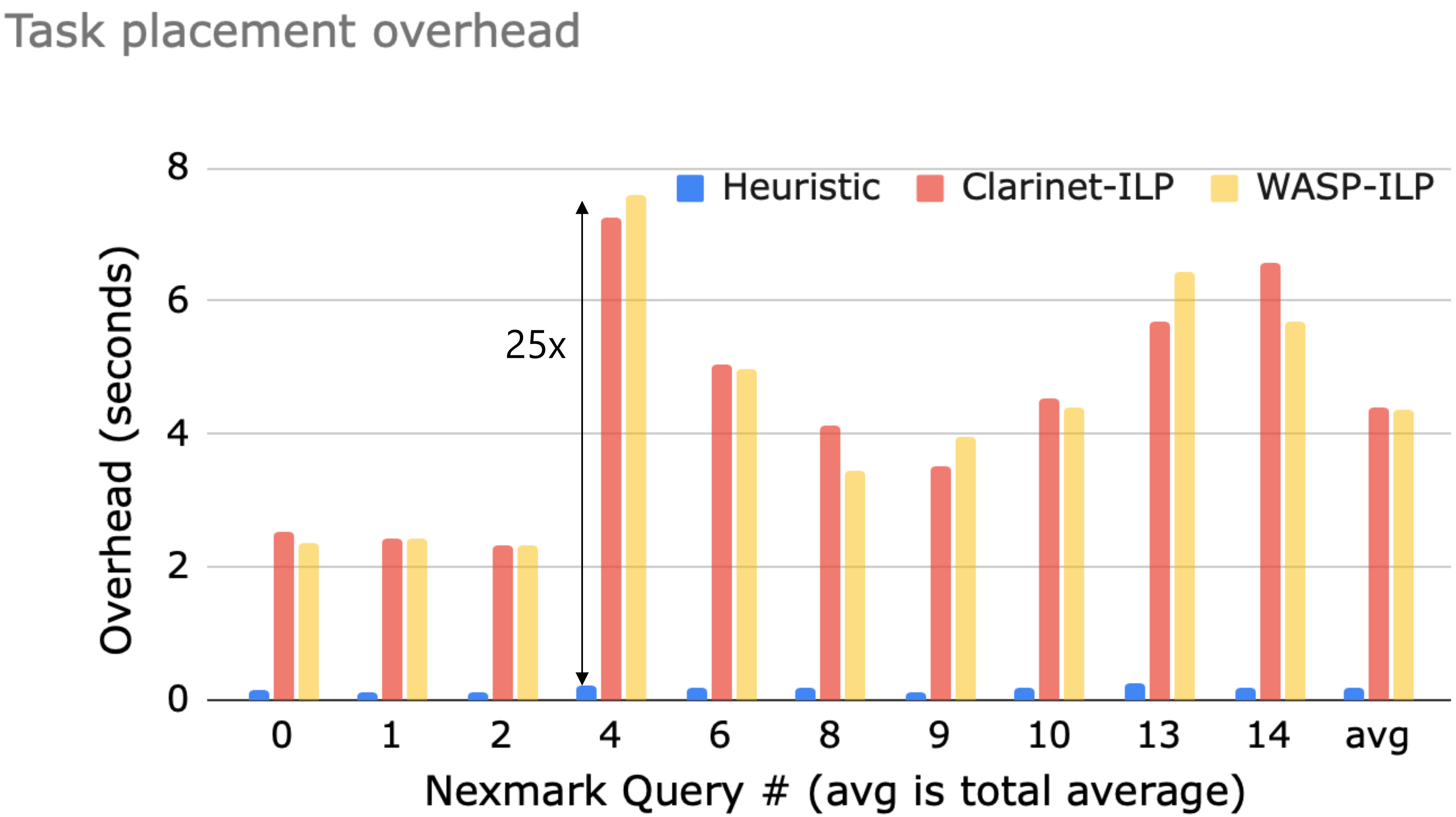
```
SELECT Istream(AVG(Q.final))
FROM Category C, (SELECT Rstream(MAX(B.price) AS final, A.category)
  FROM Auction A [ROWS UNBOUNDED], Bid B [ROWS UNBOUNDED]
  WHERE A.id=B.auction AND B.datetime < A.expires
    AND A.expires < CURRENT_TIME
  GROUP BY A.id, A.category) Q
WHERE Q.category = C.id
GROUP BY C.id;
```

Evaluation Results

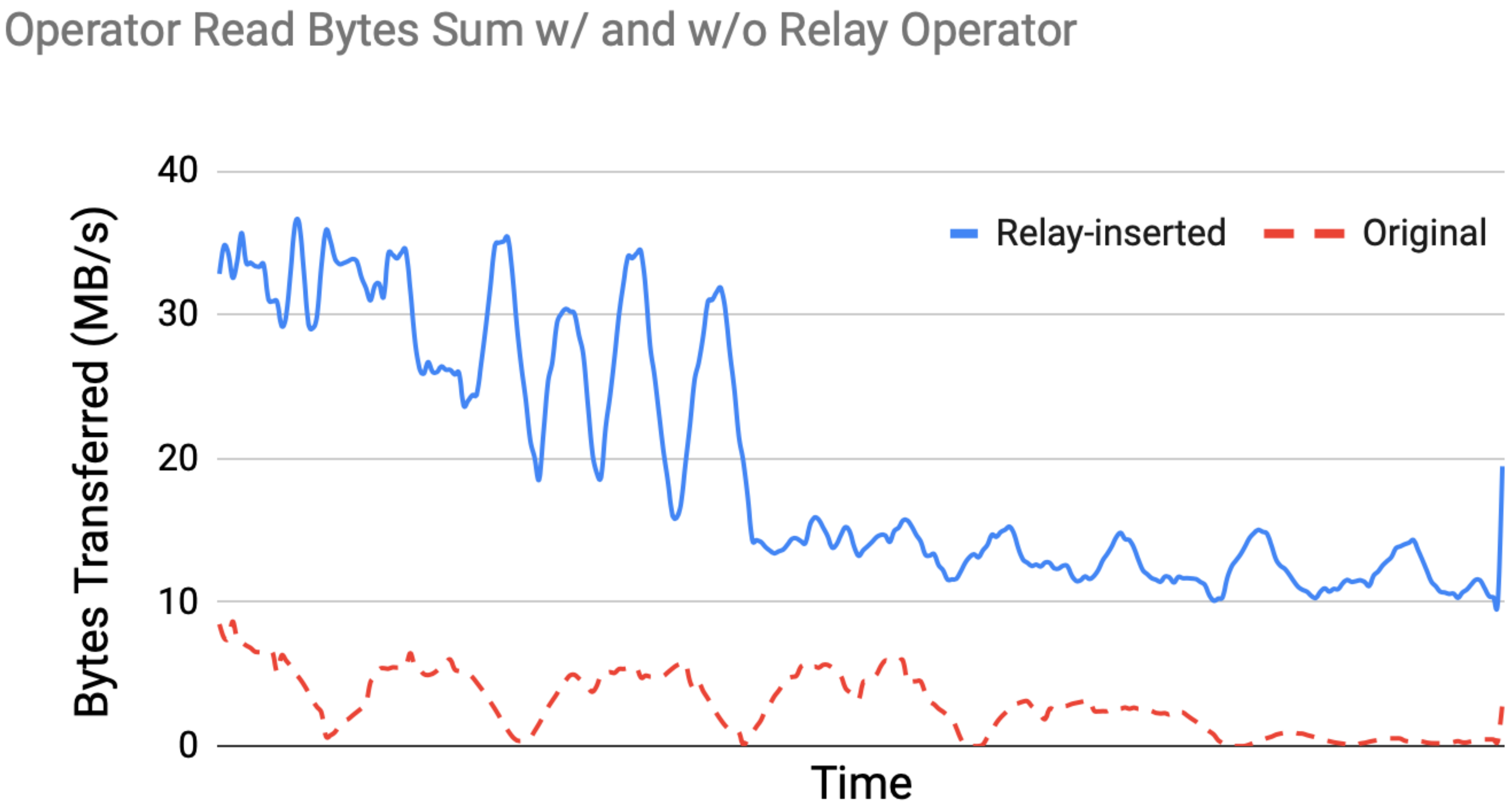


Heuristic approach prevents the delay caused by ILP optimization

Scheduling Overhead of Different Algorithms



Evaluation Results: Relay Operators



Relay operator insertion increases the throughput bytes by leveraging paths with higher bandwidths

Conclusion

- In WAN environments, *spatial* and *temporal* BW variations exist

- Existing stream systems aim to solve *temporal* variation with a centralized approach and degradation methods to maintain low latency
- Existing batch systems aim to solve *spatial* variation for lower network costs with slow ILPs

- SWAN provides a *fast heuristic model* to solve both problems
- SWAN provides *query rewriting methods* to fully cover larger BWs from longer paths

Thank you!