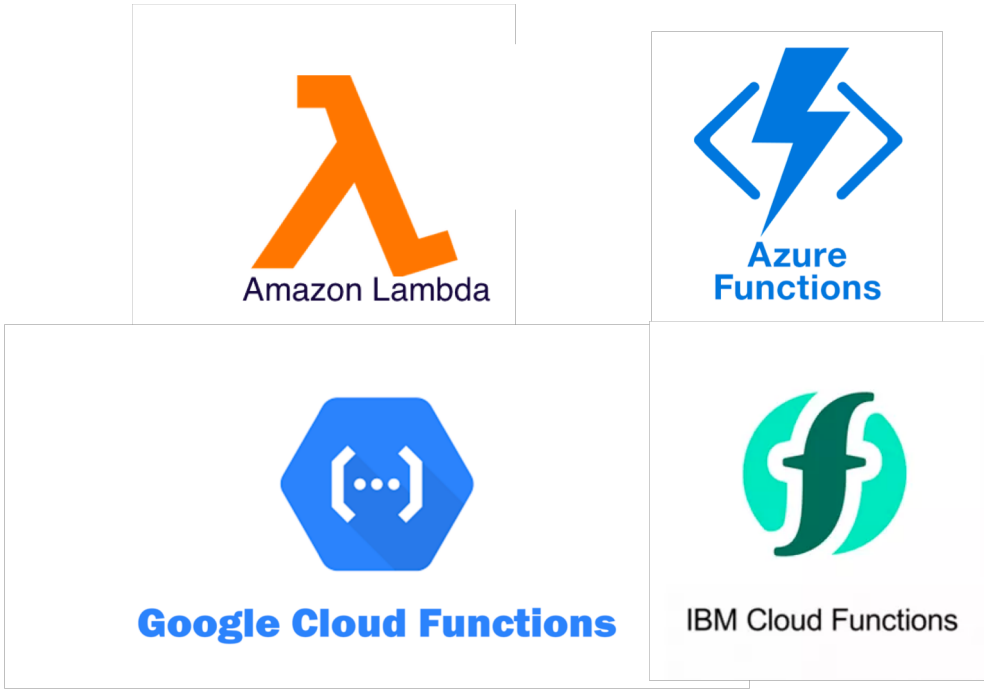
Locus

Shuffling *Fast* and *Slow* on Serverless Architecture

Qifan Pu
Shivaram Venkataraman
Ion Stoica

Serverless Computing

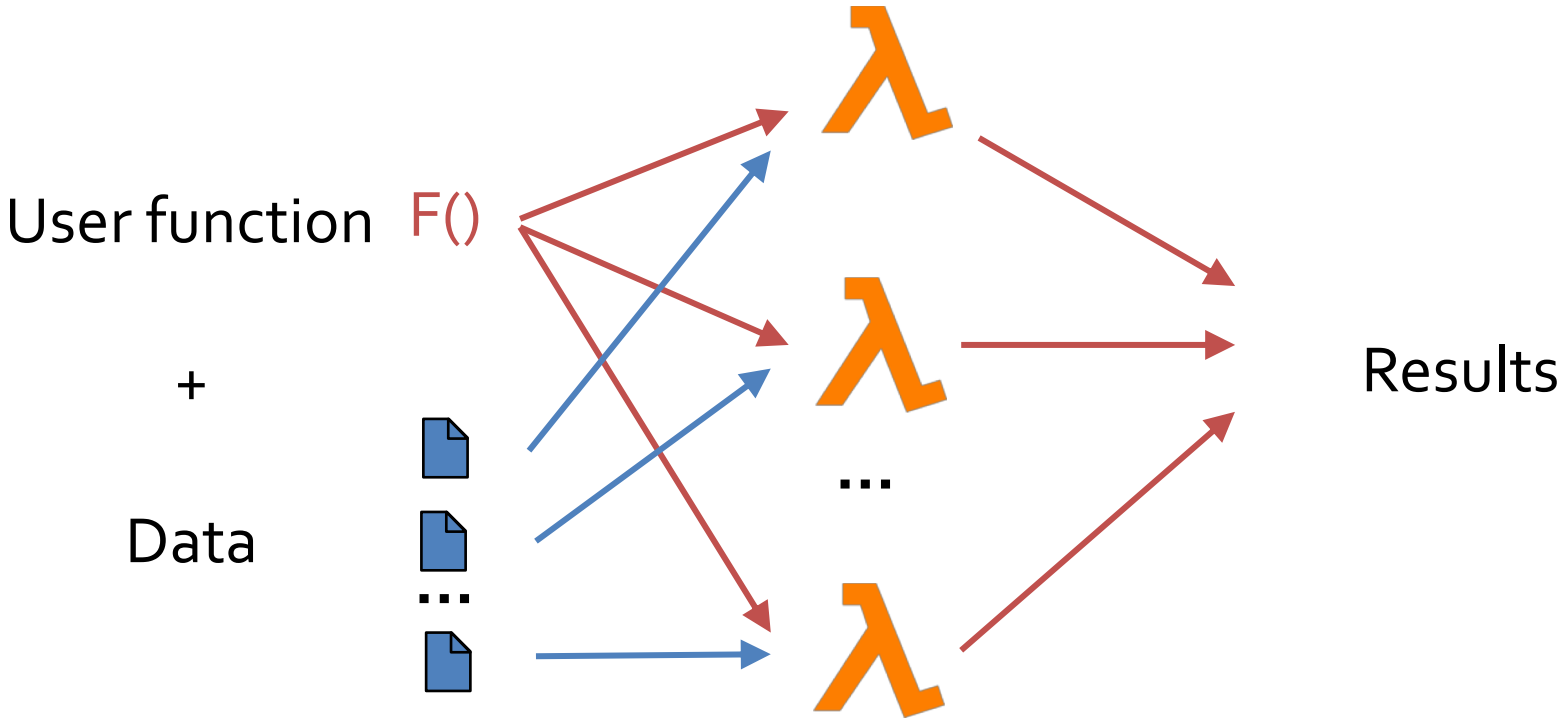


...

Serverless Analytics

Launch short-lived cloud workers with

transparent elasticity and fine-grain **usage billing**



i.e., Map()

Serverless Analytics

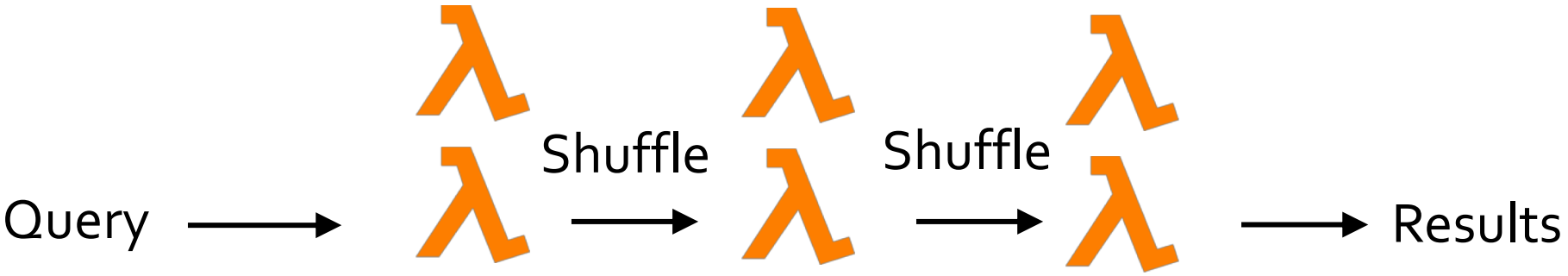
So far, a great fit for embarrassingly parallel analytics

ExCamera (NSDI'17):	video encoding
PyWren (SoCC'17):	scaling python functions
NumPyWren:	large-block matrix computation
Stanford gg compiler:	distributed compiling
AWS Redshift Spectrum:	ETL

General Serverless Analtyics

To enable general analytics, one needs to implement shuffle.

- Key operation for join() and groupby()
- 70+% of TPC-DS queries use shuffle
- Most expensive operation in production clusters

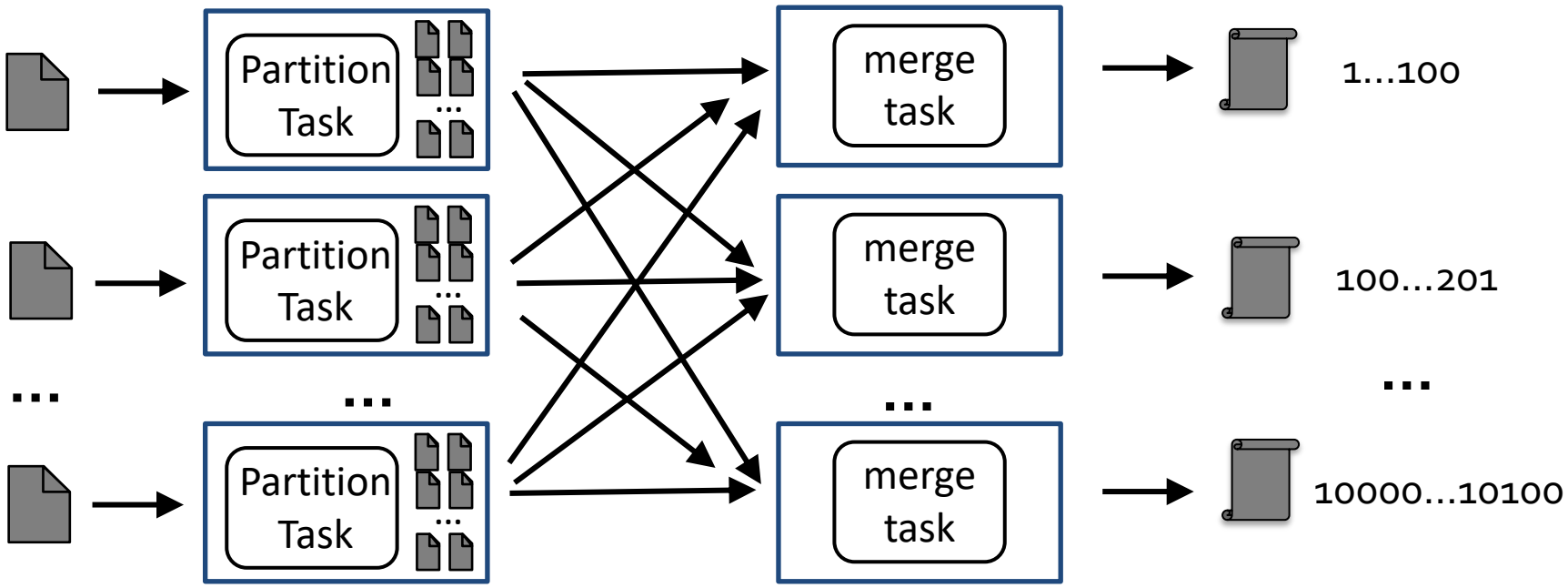


*How to perform cost-efficient **shuffle** on a serverless architecture?*

An Shuffle Example: Cloud Sort

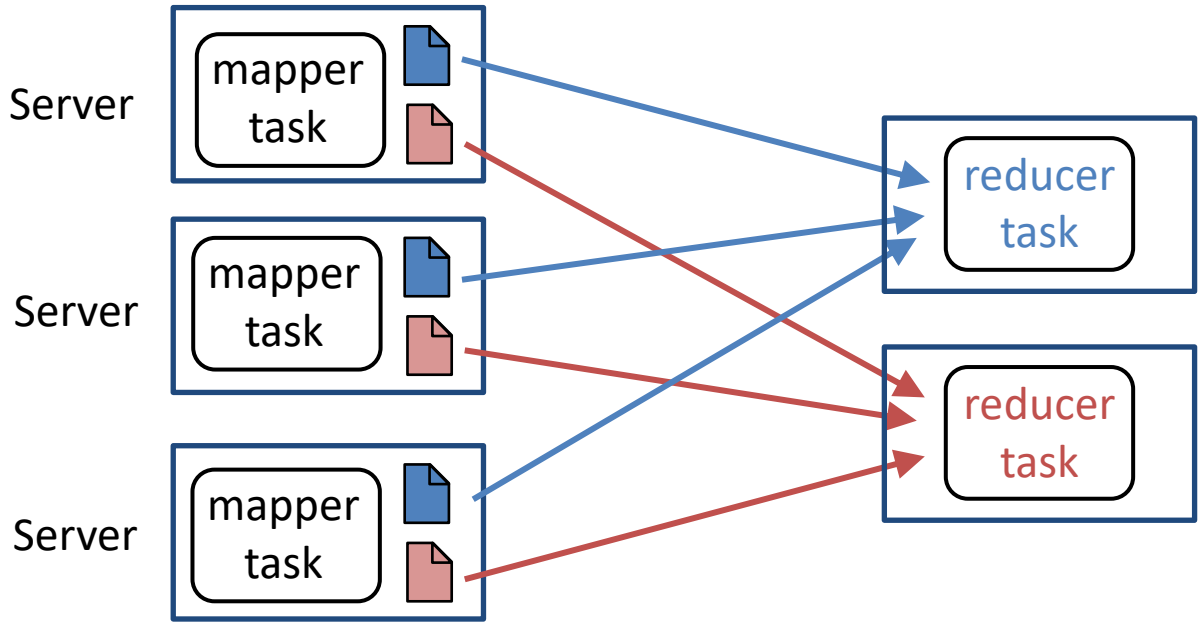
Lowest cost to sort 100TB of data

- Record: Apache Spark, 50min / $144



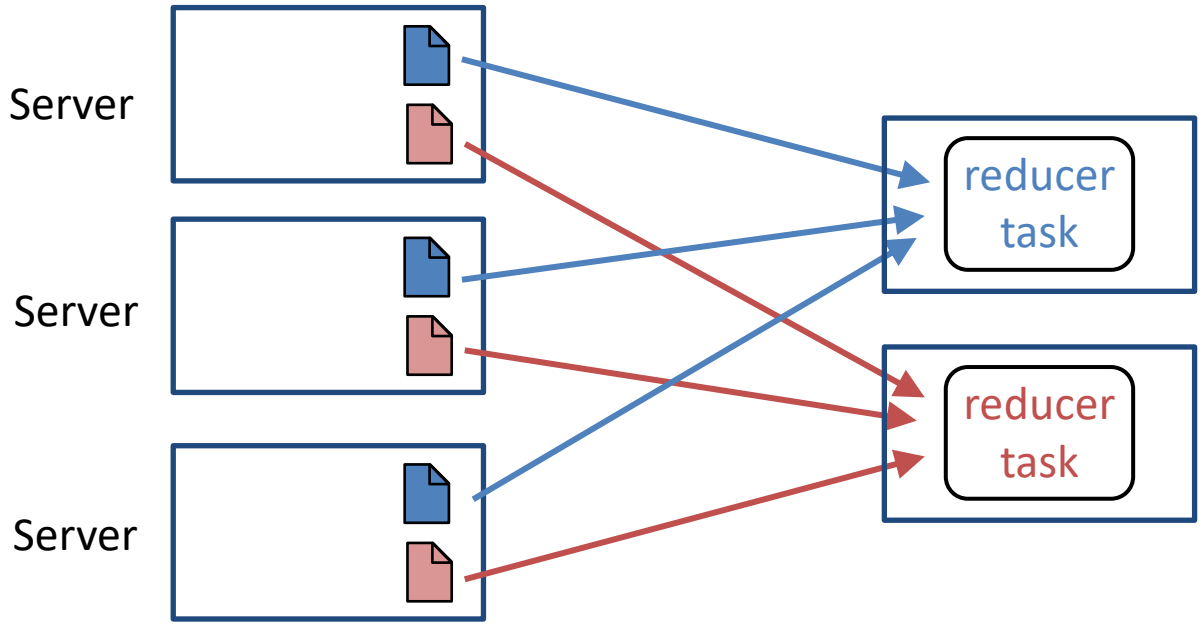
Traditional Analytics

Data communicated directly between servers that execute tasks.



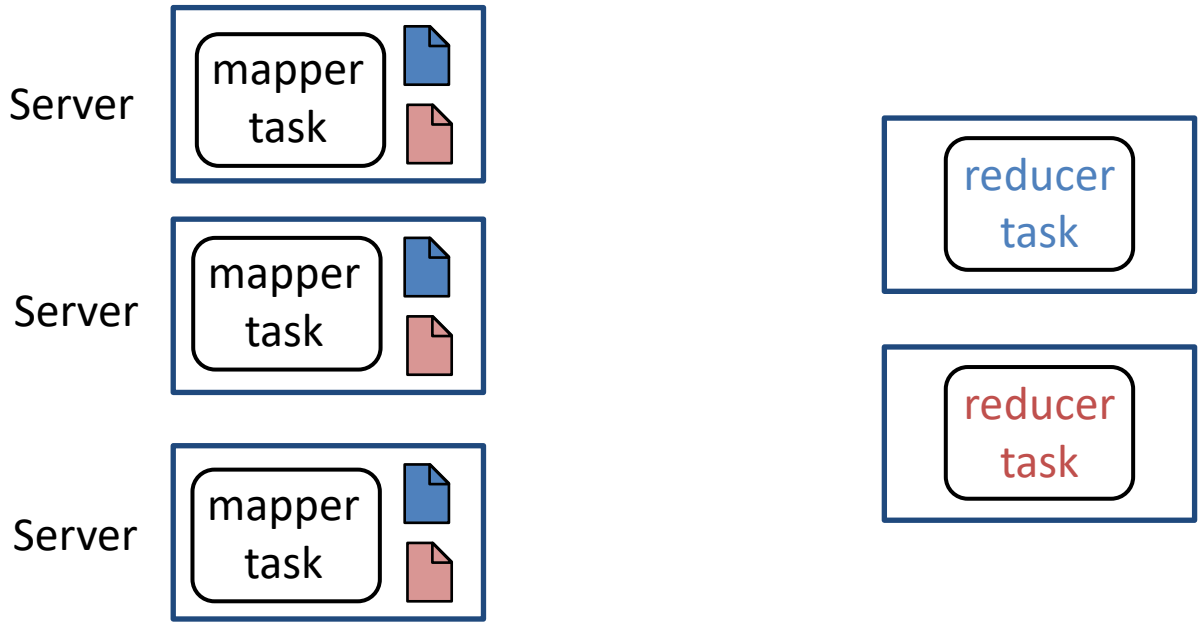
Traditional Analytics

Data communicated directly between servers that execute tasks.



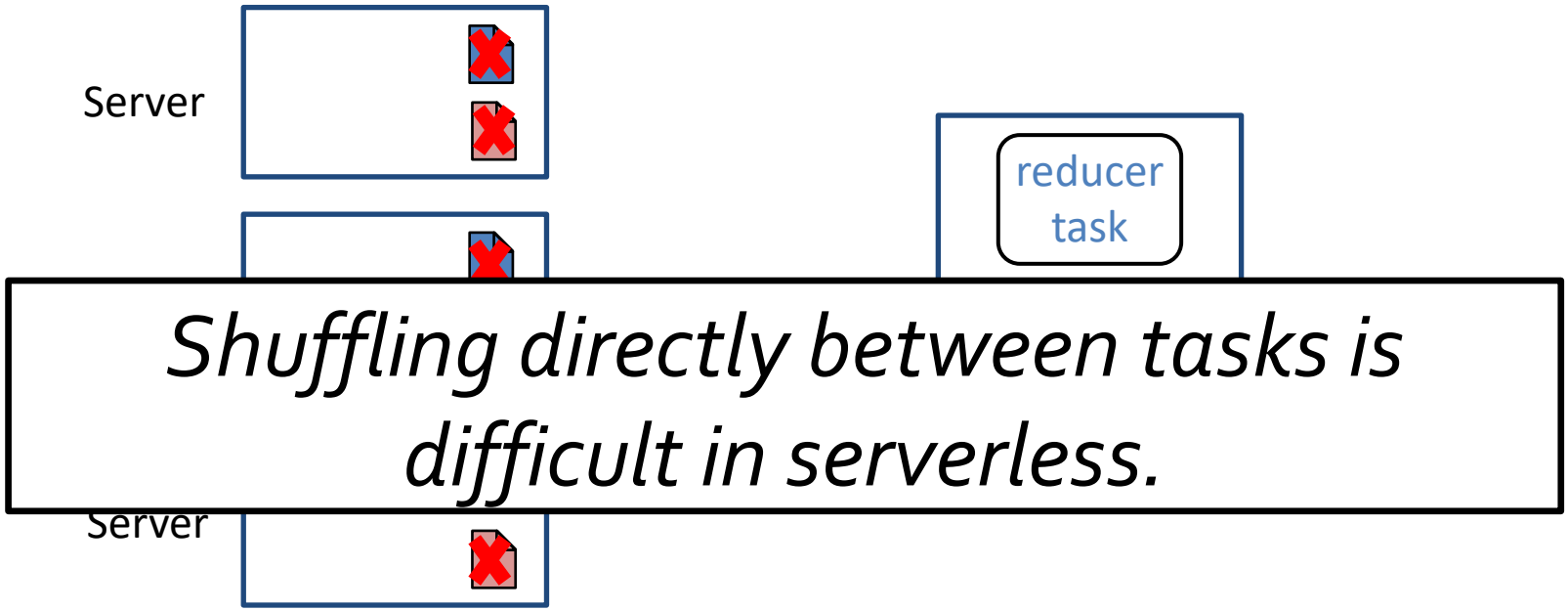
How to shuffle in serverless?

Tasks are short-lived



How to shuffle in serverless?

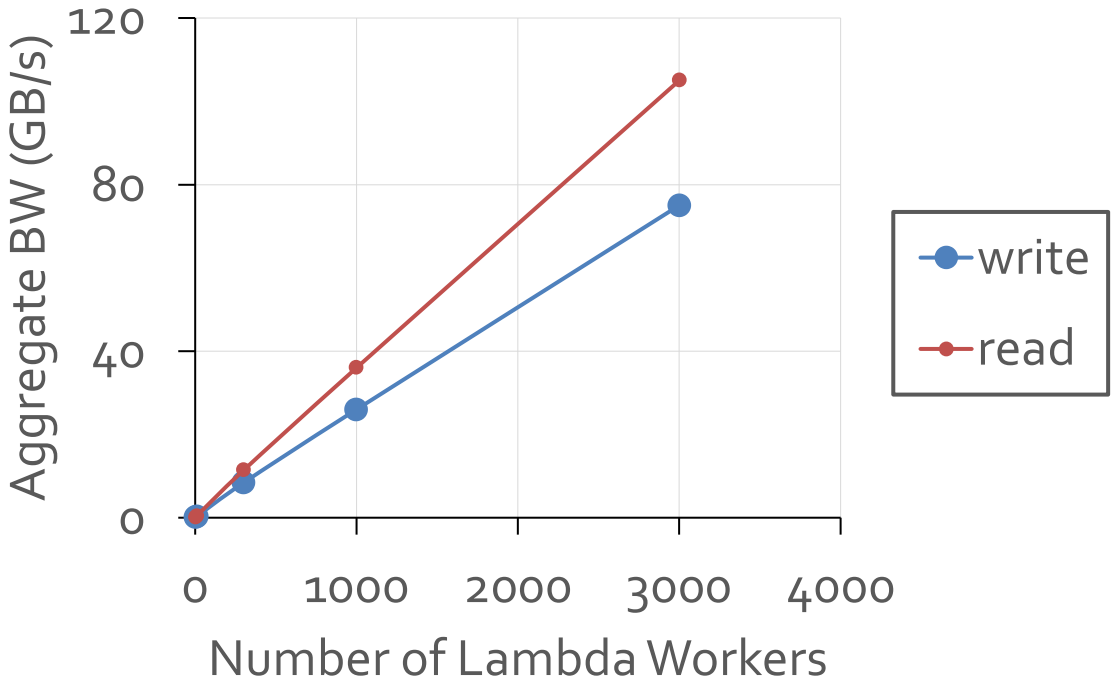
Tasks are short-lived



How to shuffle in serverless?

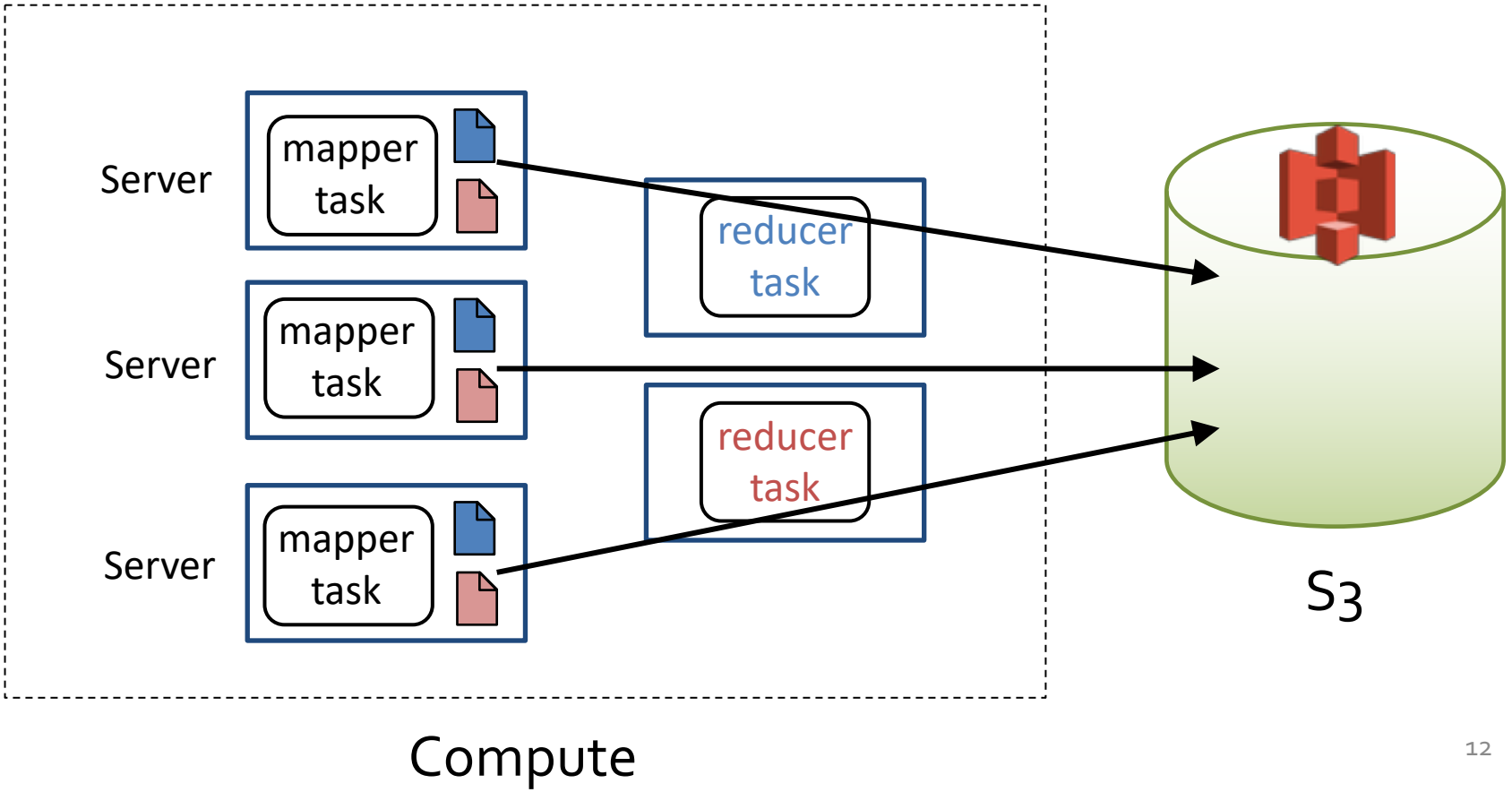
How about using S3?

- Many frameworks already use it for input/results.
- Cheap capacity and elastic bandwidth



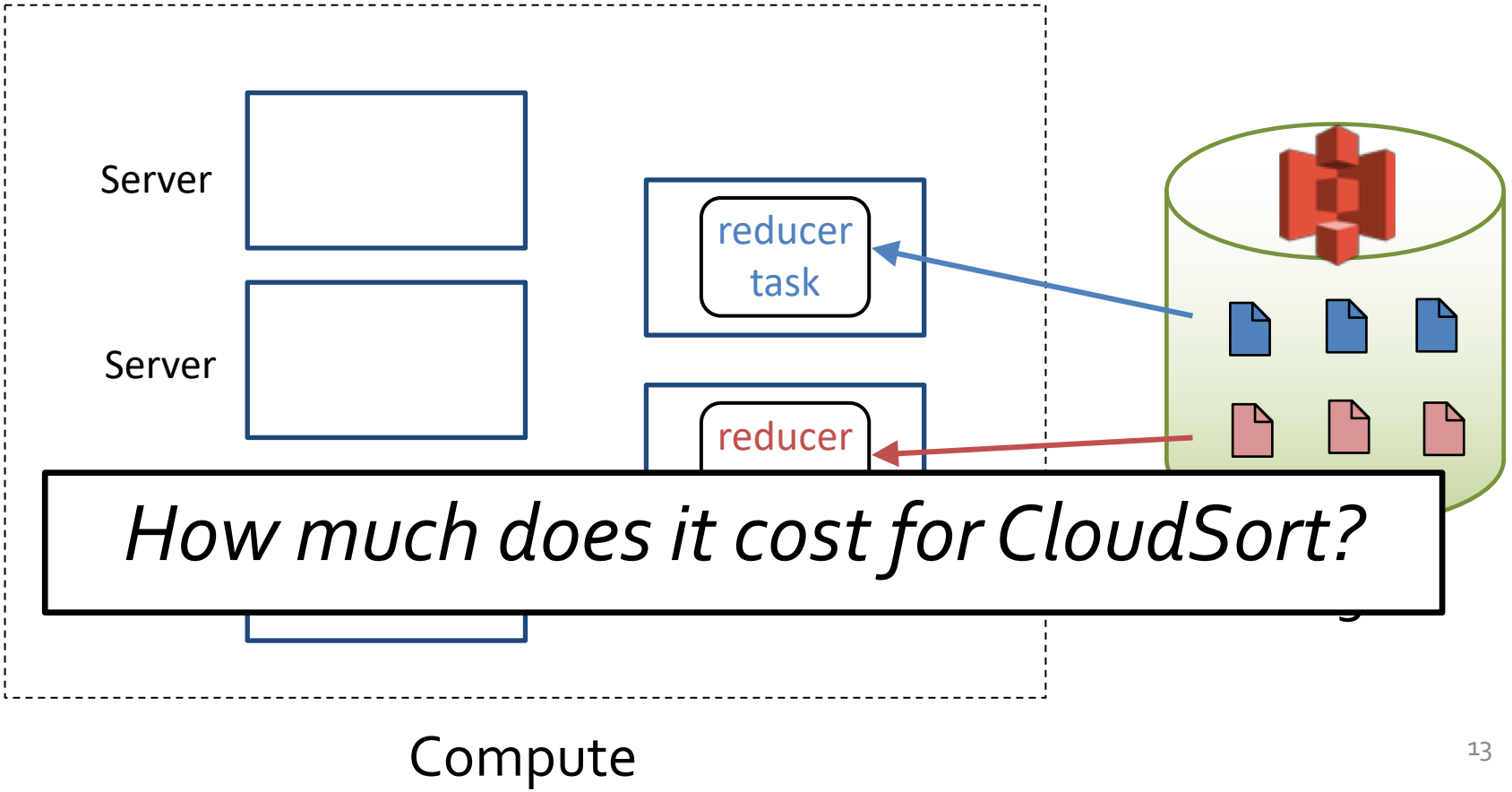
How to shuffle in serverless?

Shuffle with S3



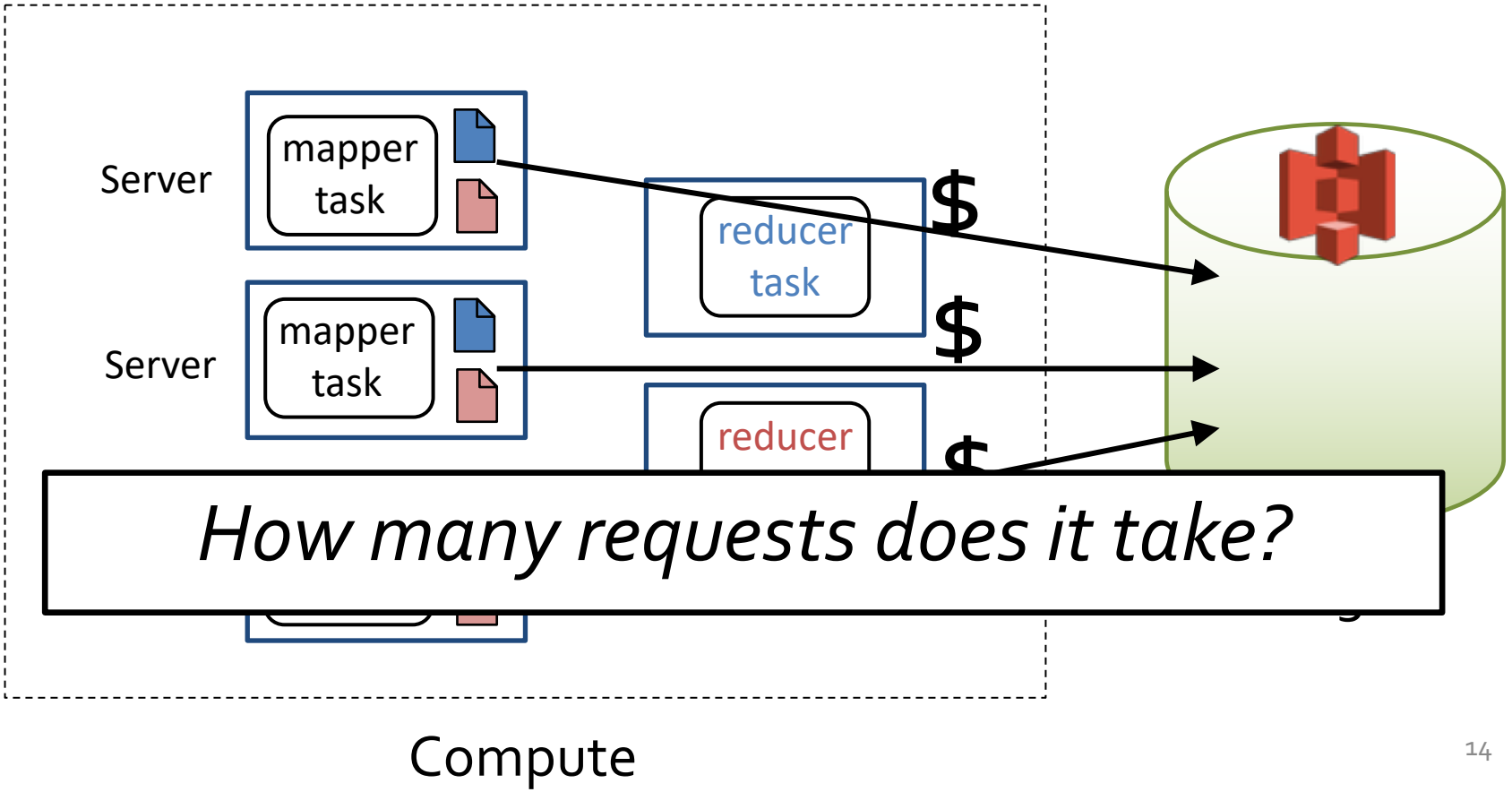
How to shuffle in serverless?

Shuffle with S3

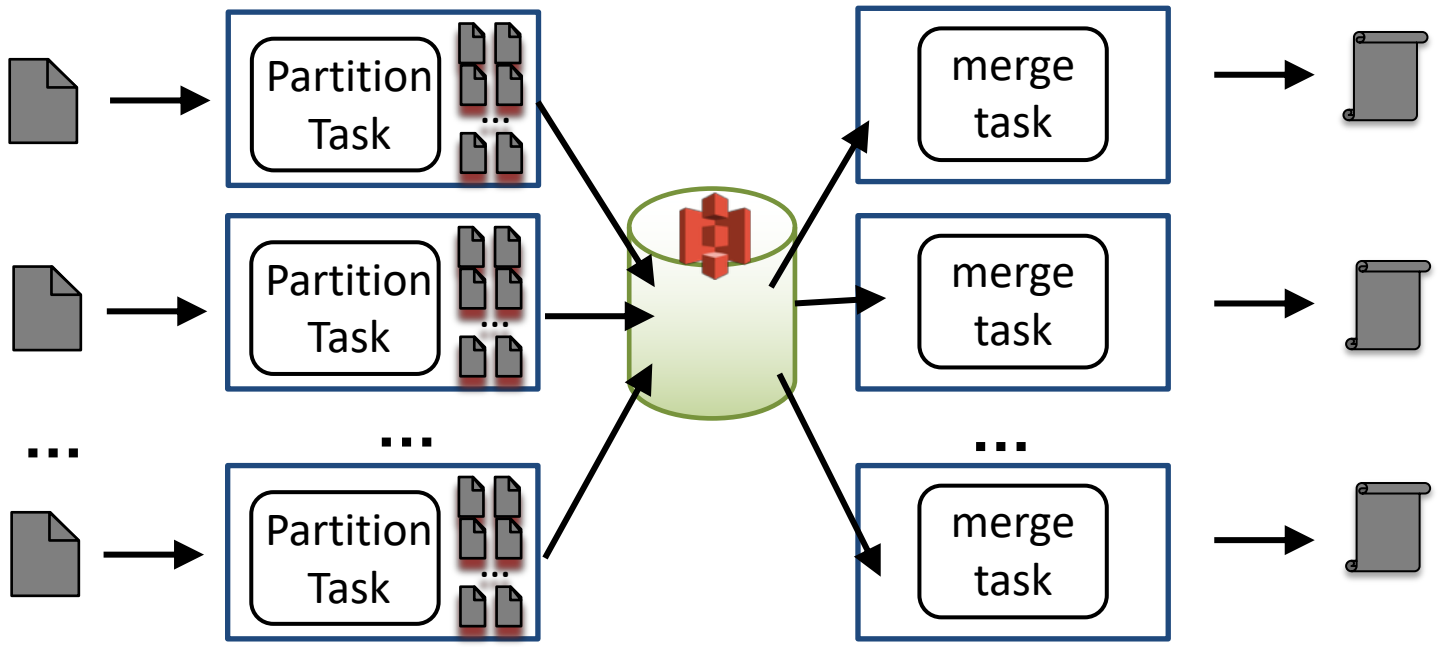


How to shuffle in serverless?

Shuffle with S3

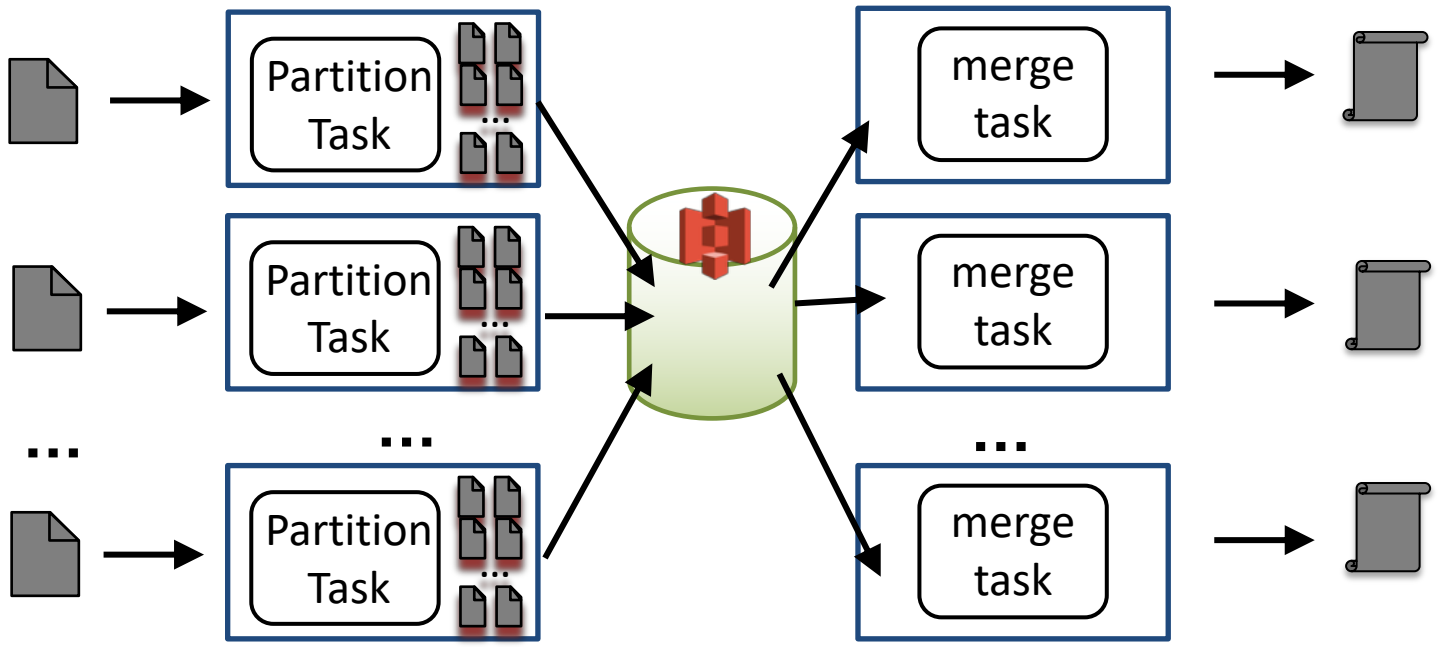


An Shuffle Example: Cloud Sort



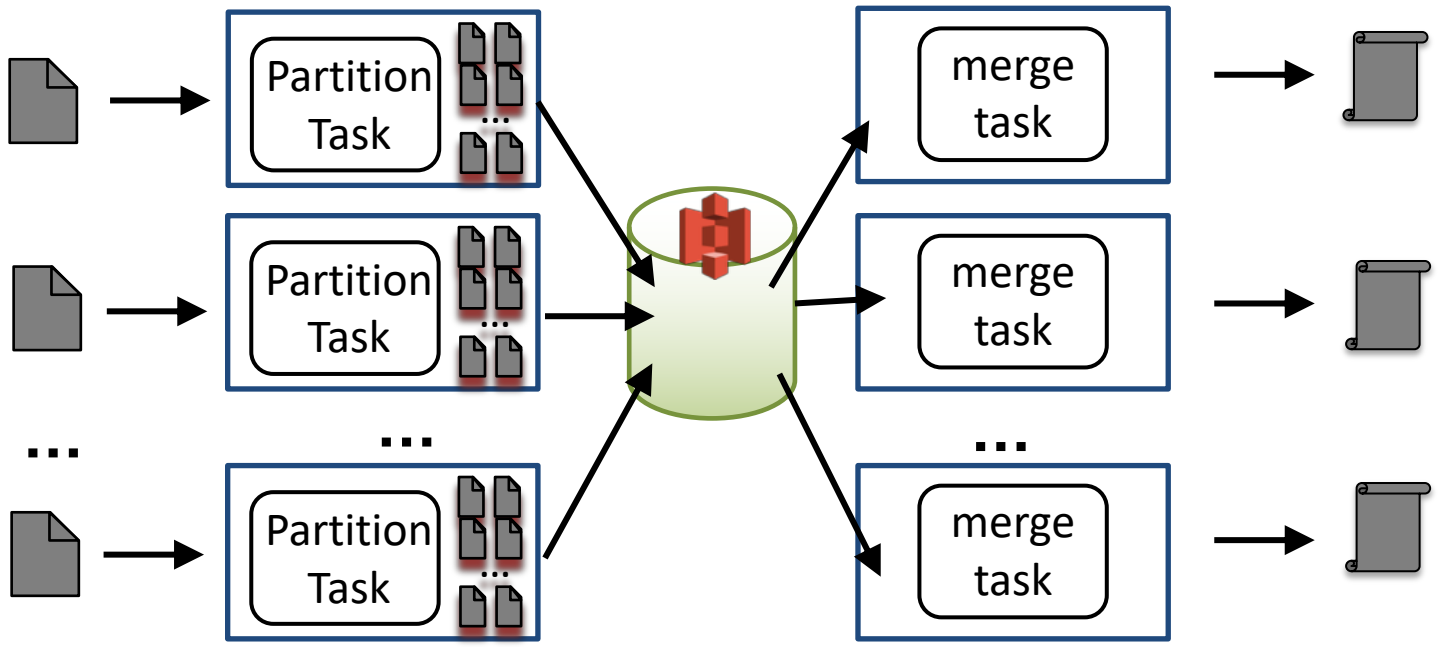
- Total number of transfers:
  - Assuming 1GB serverless containers
  - Num partition/merge tasks = 100TB/1GB = 10^5
  - Number of files: 10^{10} = 10 billion files

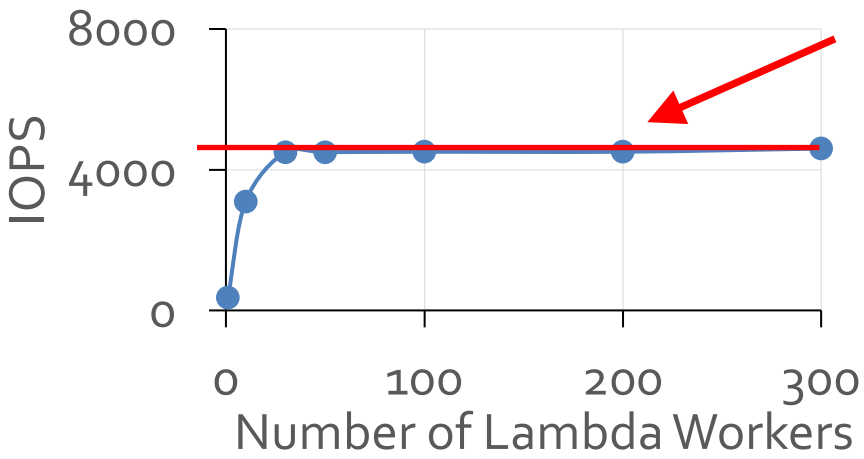
An Shuffle Example: Cloud Sort



- Total number of transfers:
  - Assuming 1GB serverless containers
  - Num partition/merge tasks = 100TB/1GB = 10^5
  - Number of files: 10^{10} = 10 billion files
  - $0.000005/op -> $50k

An Shuffle Example: Cloud Sort

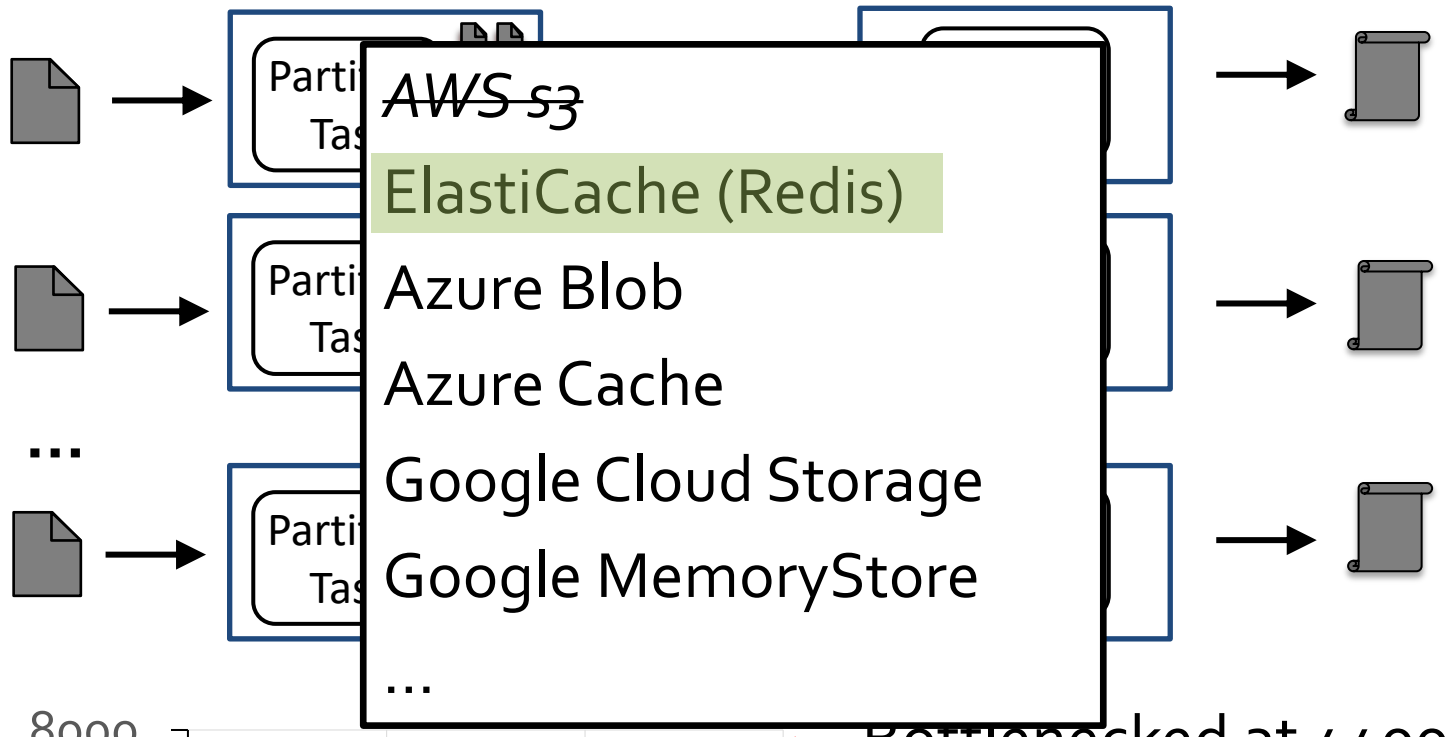




Bottlenecked at 4400 ops/sec!

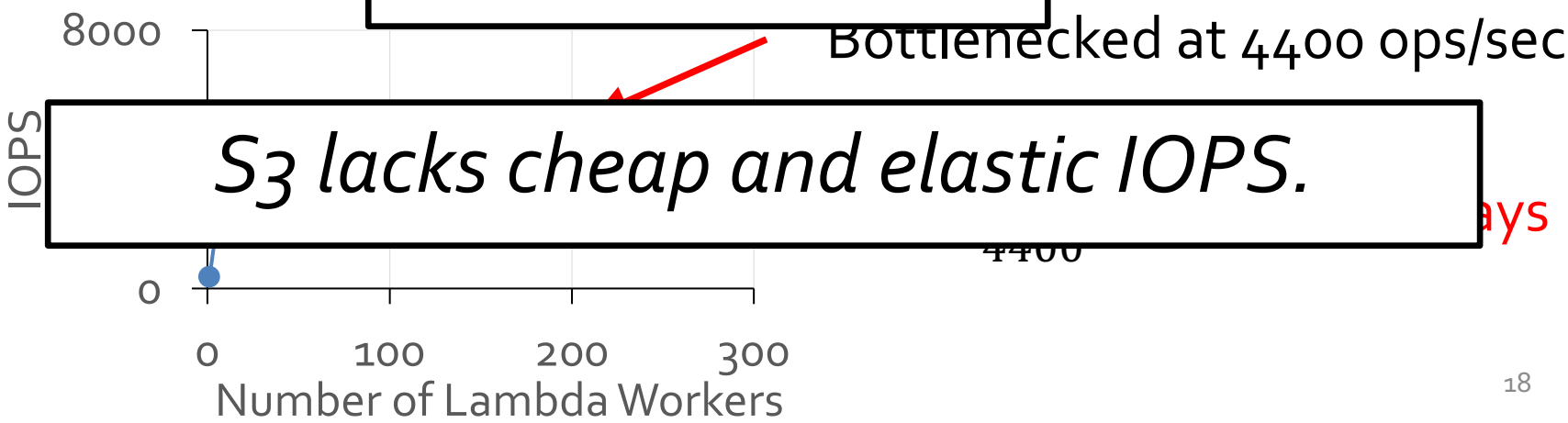
Takes $\frac{10^{10}}{4400}$ seconds = 26 days

An Shuffle Example: Cloud Sort

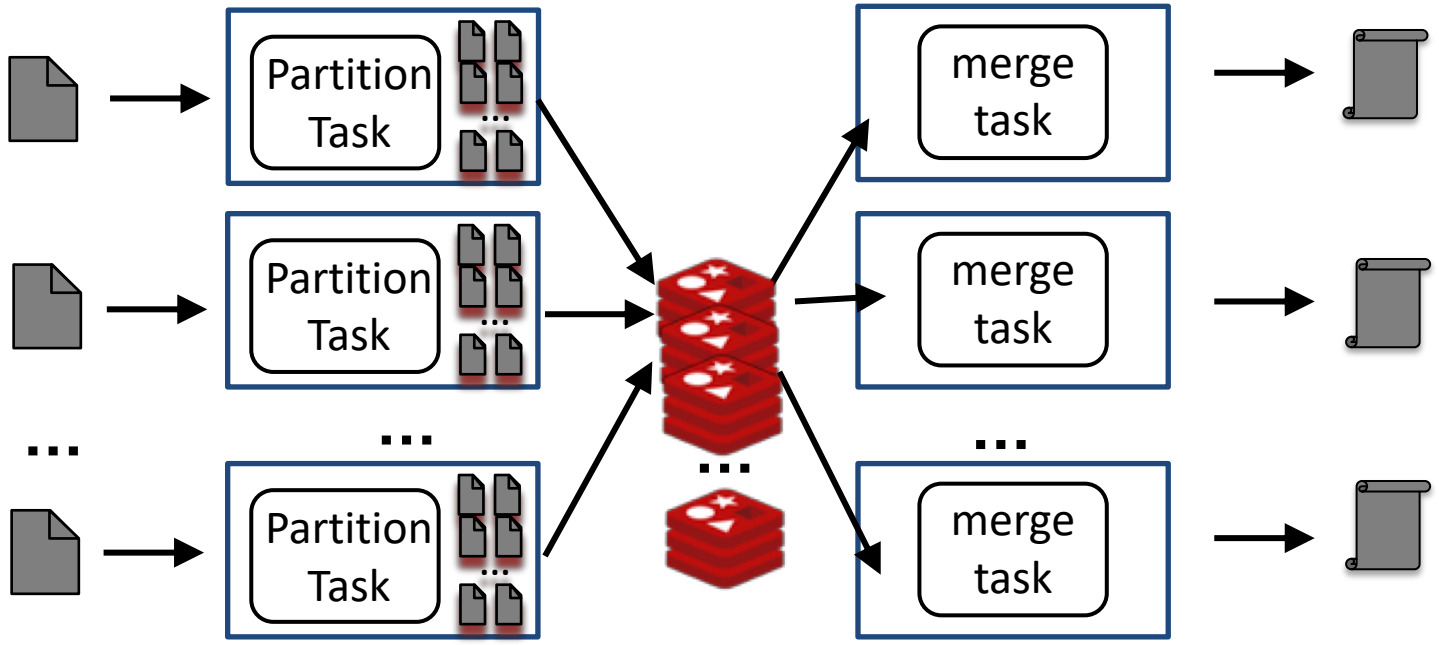


Bottlenecked at 4400 ops/sec!

S3 lacks cheap and elastic IOPS.



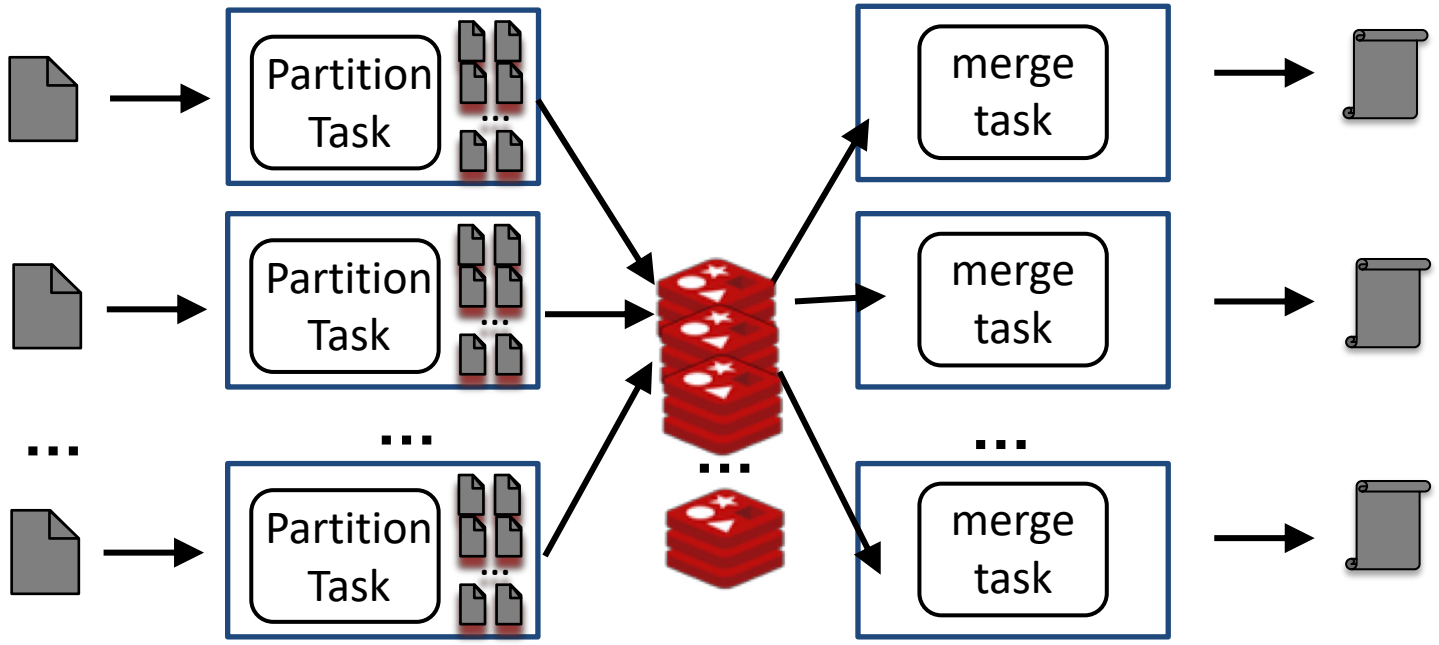
An Shuffle Example: Cloud Sort



Require 158 large (r5.24xlarge) instances!
Costs $1.6K/hour.

Capacity alone is 10X more expansive than record.

An Shuffle Example: Cloud Sort



Require 158 large (r5.24xlarge) instances!

Redis capacity is too expensive for large intermediate data.

Cloud Storage

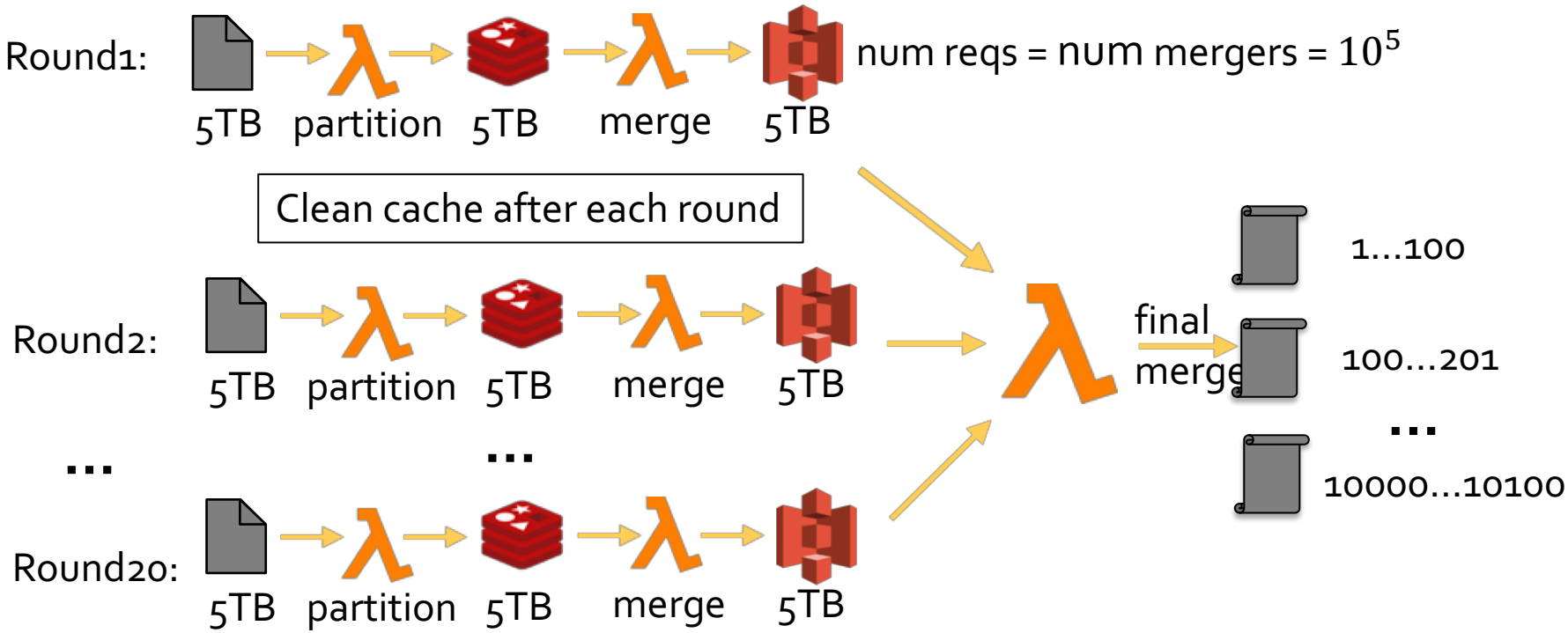
Service	Capacity	IOPS
Slow Storage	High	Low
Fast Storage	Low	High

Can we leverage the strengths of two storages types?

Locus

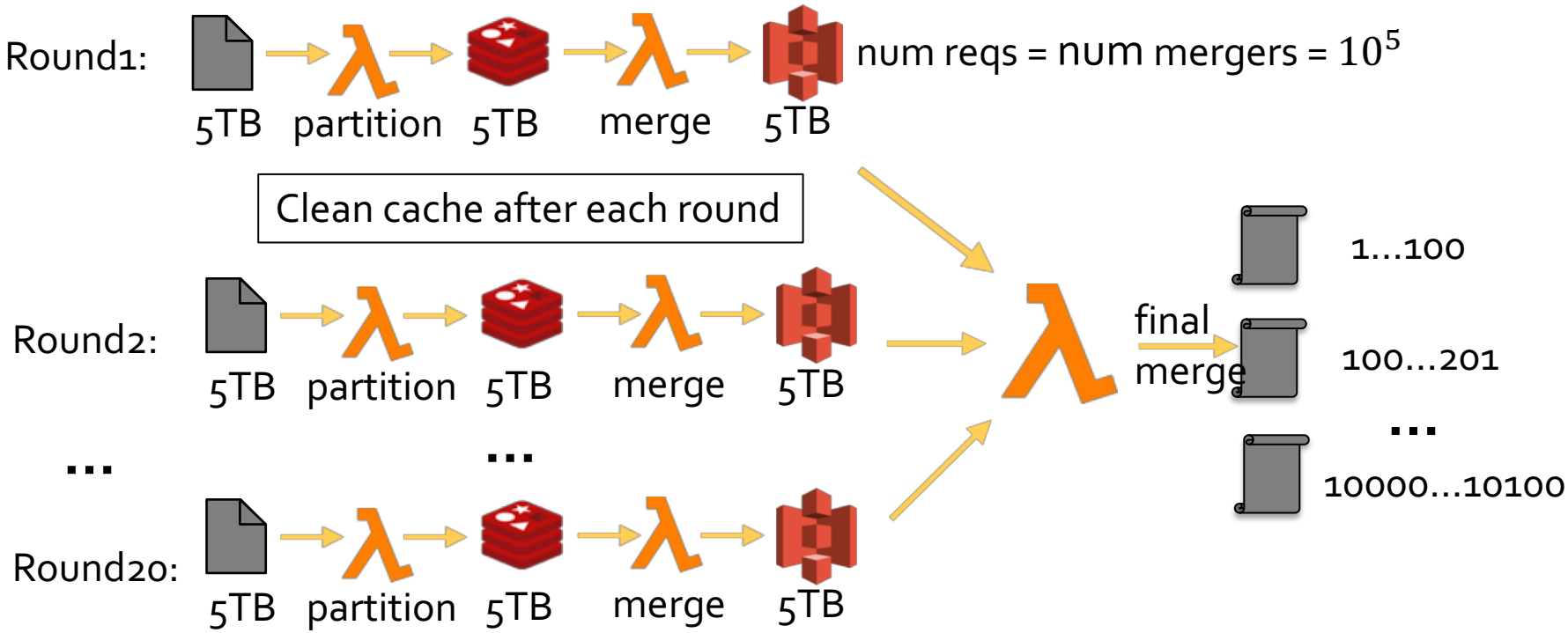
- Hybrid *slow* and *fast* cloud storage to achieve cost-efficient shuffle performance
  - Intuition: use fast storage to absorb IOPS and create bigger chunks for slow storage.

Hybrid sort (100TB)



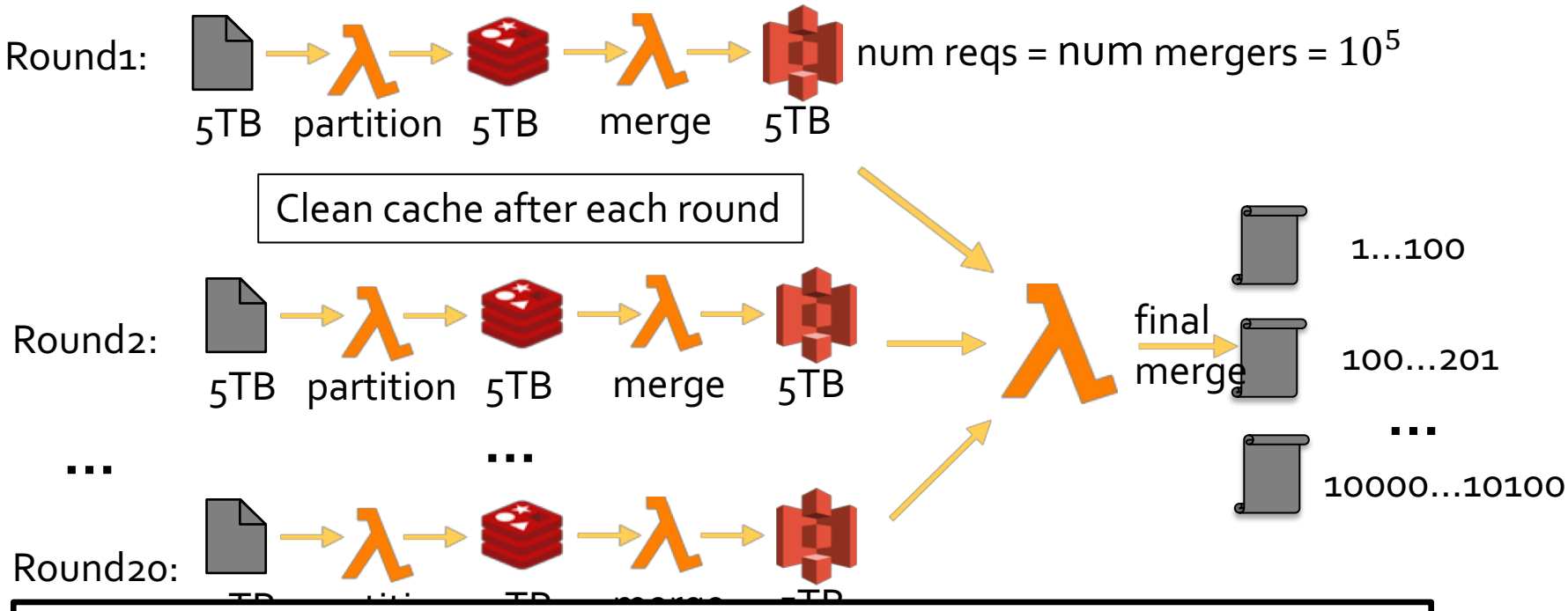
- Total number of S3 reqs: 20×10^5 ($vs.$ 10^{10})
- Total memory cache needed: 5TB (vs. 100TB)

Hybrid sort (100TB)



- Spark record: 50min + $144
- Locus: 50min + $163

Hybrid sort (100TB)



Locus achieves same performance with 5X less resource.

Locus

- Hybrid *slow* and *fast* cloud storage to achieve cost-efficient shuffle performance
  - Intuition: use fast storage to absorb IOPS and create bigger chunks for slow storage.
- When to use hybrid shuffle?
  - Alongside, many other configurations
    - Amount of fast cache
    - Level of parallelism
    - Worker size

Locus Performance Model

Navigating cost-performance with different compute and storage configurations is difficult.

Locus Performance Model

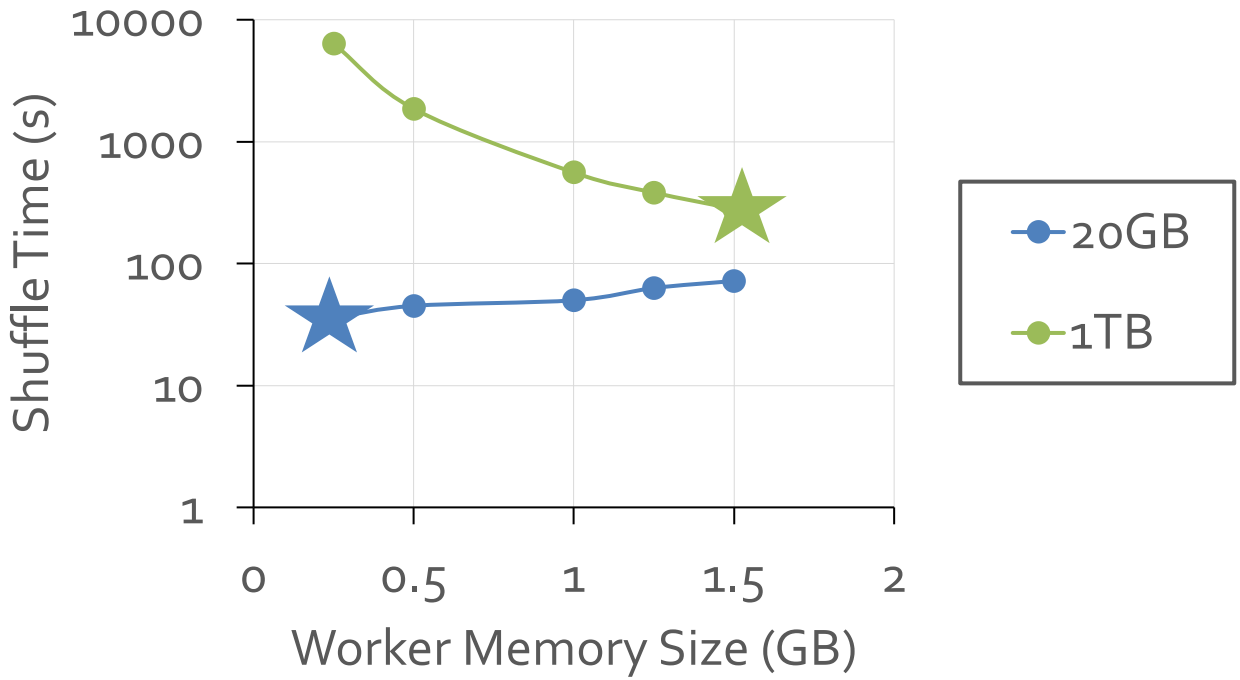
Navigating cost-performance with different compute and storage configurations is difficult.

- How many serverless workers? Which size?

Locus Performance Model

Navigating cost-performance with different compute and storage configurations is difficult.

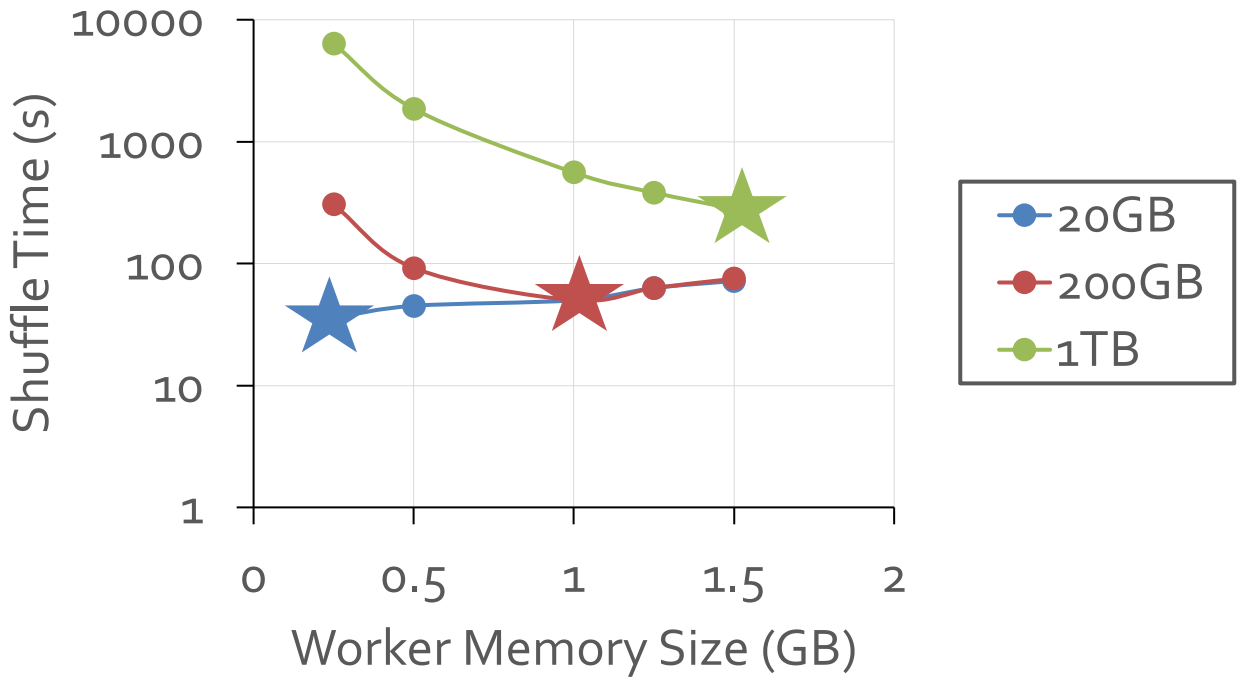
- How many serverless workers? Which size?



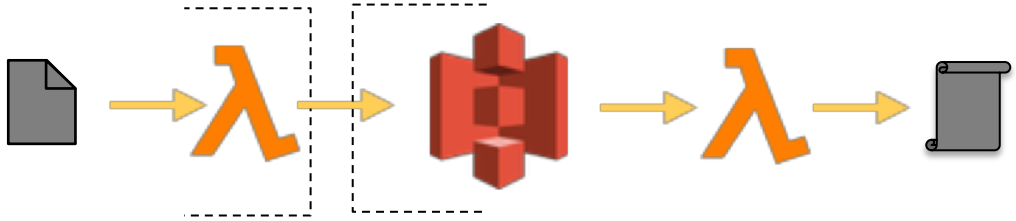
Locus Performance Model

Navigating cost-performance with different compute and storage configurations is difficult.

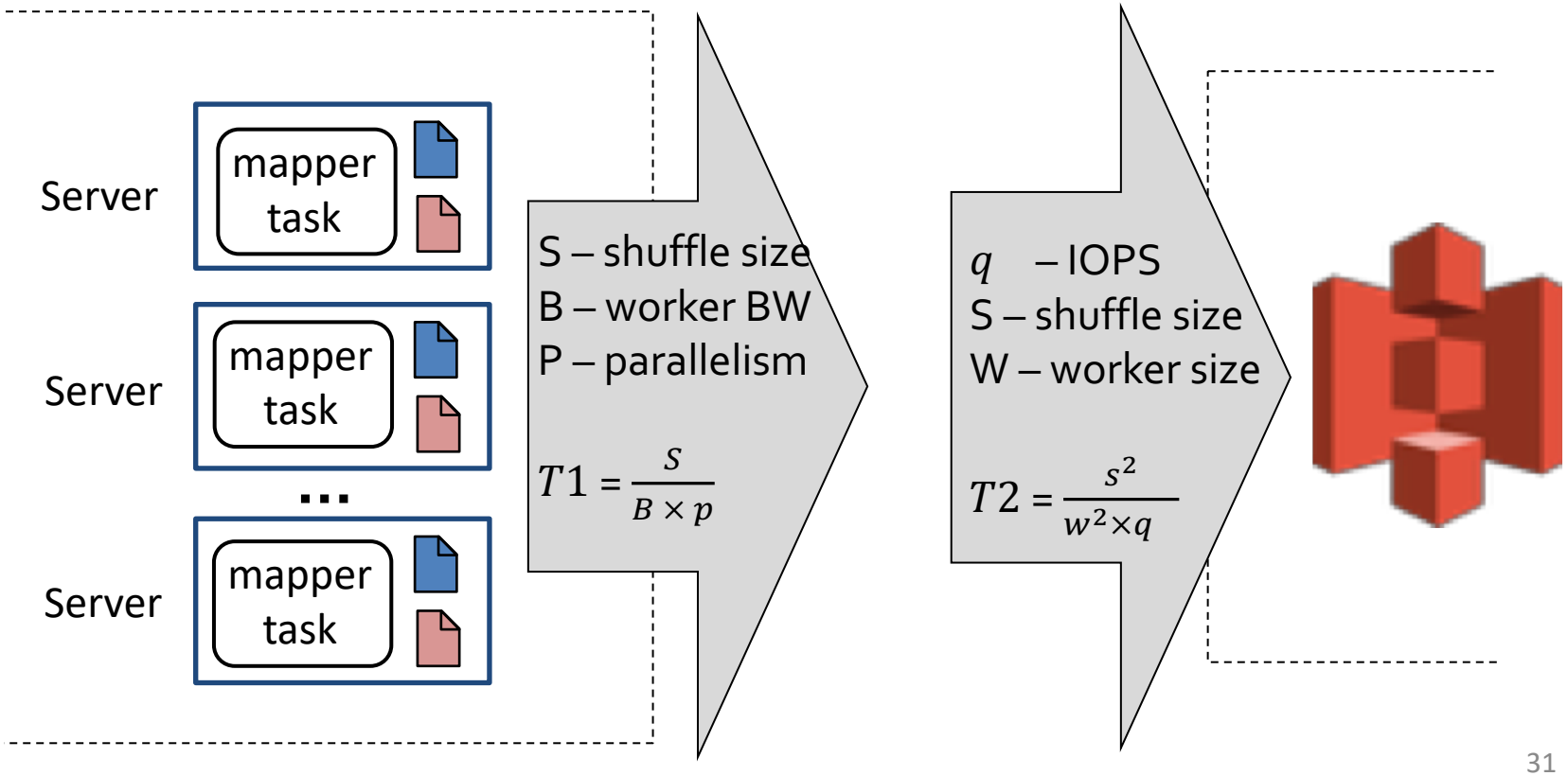
- How many serverless workers? Which size?



Performance model example: slow-storage only shuffle



$$T = \max(T1, T2)$$



Evaluation

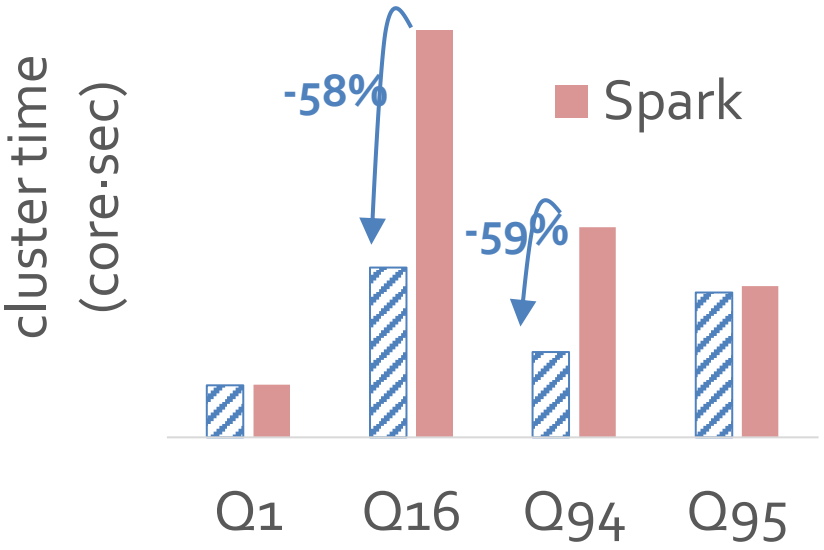
- Implement Locus on top of PyWren
  - *AWS Lambda*
  - *S3 (slow)*
  - *Redis (fast)*

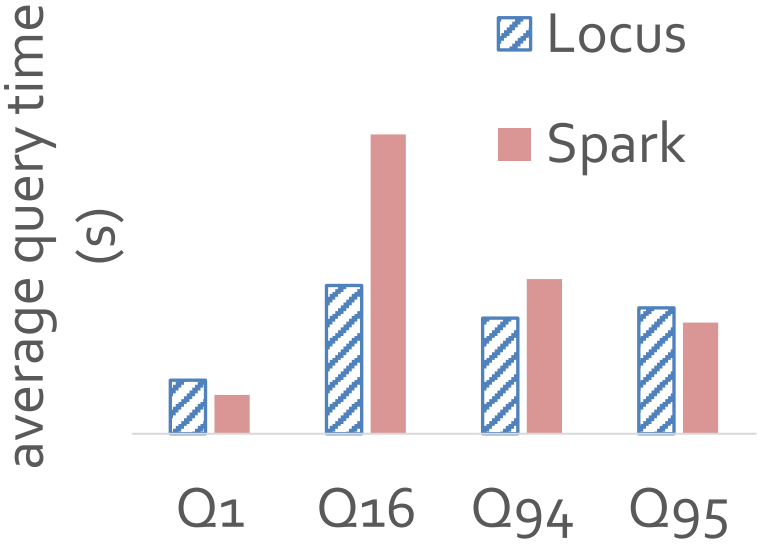
- Workloads:
  - *CloudSort*
  - *TPC-DS*
  - *Big data benchmark*

- Baseline:
  - *PyWren*
  - *Apache Spark on VMs*
  - *Redshift*

Evaluation

TPC-DS





Reduce resource usage by up to 59% while achieving comparable performance!

Related Works

Pocket (OSDI'18)

- New elastic store that scales to application demand

Locus

- By judiciously combining *slow* and *fast* cloud storage, one can achieve cost-efficient shuffle for serverless analytics.

- Locus achieves this by:
  - Design algorithms that leverage storage characteristics
  - A performance model that captures the performance and cost metrics of shuffle operations.

- 4×-500× performance improvements over baseline and reduce resource usage by up to 59% while achieving comparable performance with traditional analytics