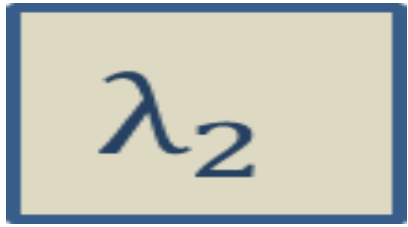
SONIC: Application-aware Data Passing for Chained Serverless Applications

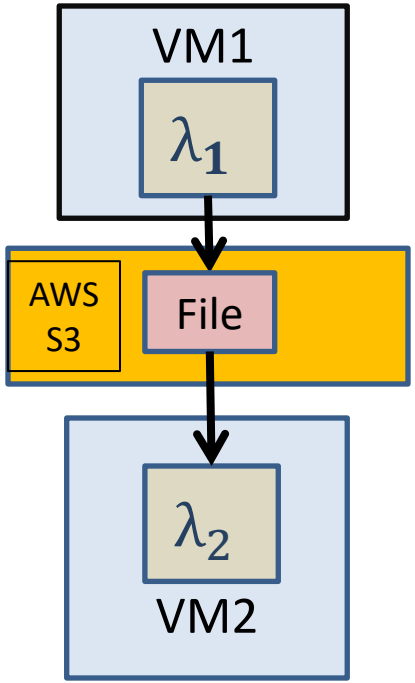
Ashraf Mahgoub[1], Karthick Shankar[2], Subrata Mitra[3], Ana Klimovic[4], Somali Chaterji[1], Saurabh Bagchi[1]

1: Purdue University; 2: Carnegie Mellon University; 3: Adobe Research; 4: ETH Zurich

Background: Serverless Computing DAG & Data Passing



- FaaS are becoming increasingly popular for complex workflows: Video Analytics, ML pipelines, Genomics
- Cloud provider performs administrative tasks (Scaling, Scheduling, Maintenance)
- Applications are represented as a series of stateless functions that pass data among themselves (a DAG)
- Since function's placement is hidden from users, ***direct communication*** between the functions is infeasible
  - Direct communication also requires both sending and receiving functions to execute simultaneously, FaaS frameworks usually provide no such guarantees
- **State-of-practice:** Use a remote storage (e.g., AWS S3)



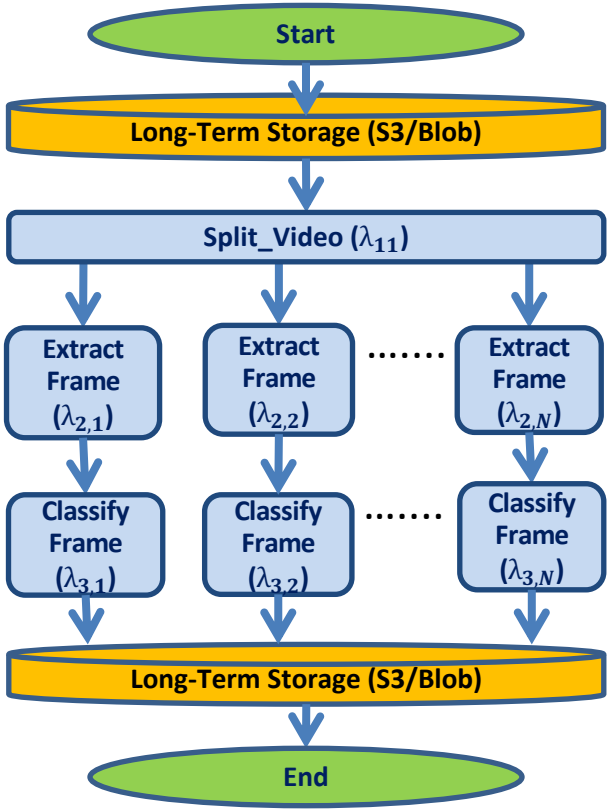
Workflow Example: Video Analytics DAG

Any application DAG can be represented as a **state machine** in AWS Step Function

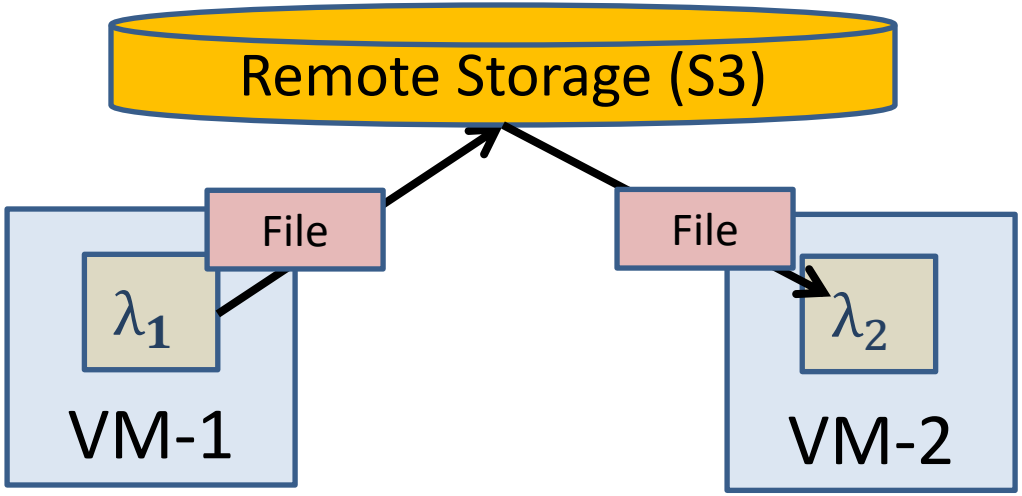
1. *Split-Video:* Takes a video clip as an input (loaded from S3/Blob Storage), and generates smaller video chunks of the same length (10 sec)

2. *Extract-Frame:* Takes a video chunk as input and extracts a representative frame from that chunk

3. *Classify-Frame:* Performs object classification for extracted frame and writes the classification results to Storage

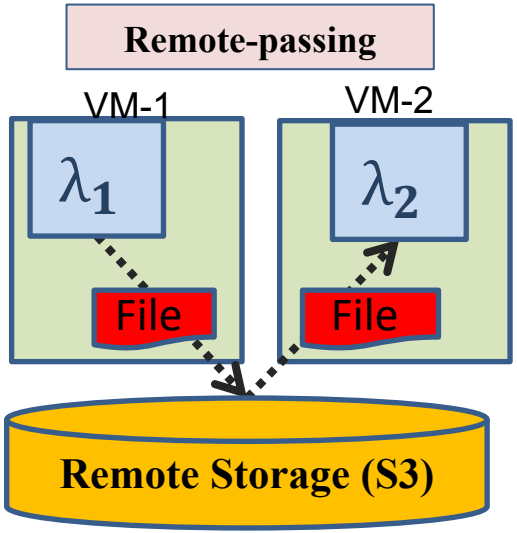


Serverless Data-Passing Challenges



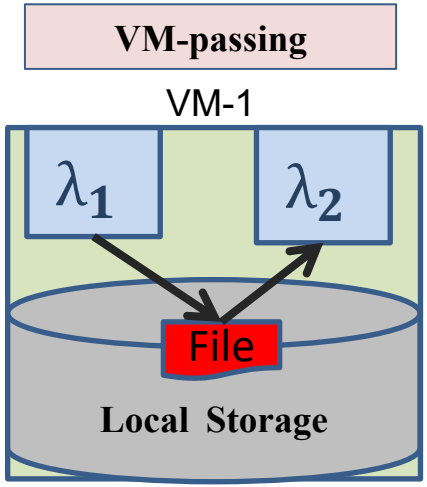
- The state-of-practice technique to communicate between $\lambda's$ is to use a Remote Storage (e.g., S3)
  - Step Functions supports passing direct **JSON** payloads of very small sizes (≤ 256KB)
- Pros:
  - **Allows for flexible scheduling:** Poses no limitation on where the $\lambda's$ execute
- Cons:
  - **Increased latency:** Data needs to be transferred over the network twice, to and from the remote storage

SONIC's Data-Passing Alternatives



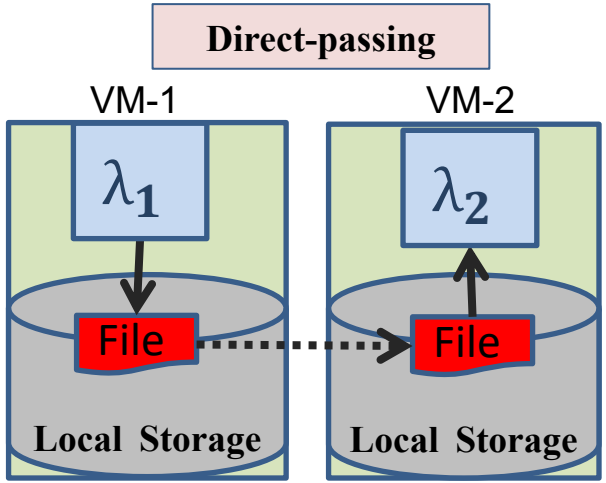
Data transfer over the network

Data is written/read from local storage



Forces λ_2 to run on the same VM as λ_1

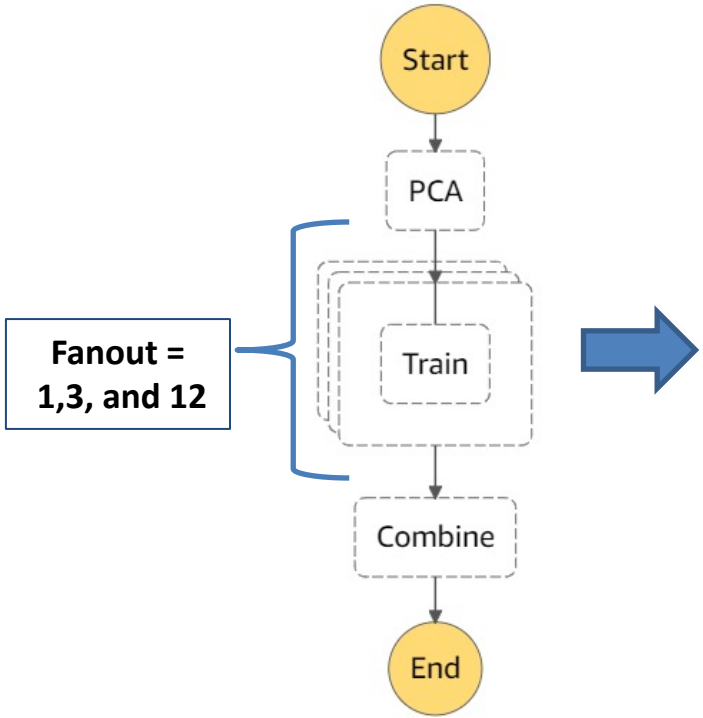
- Infeasible if λ_2 cannot fit in VM-1
- Also infeasible if λ_2 waits for data from a third lambda λ_3 (i.e., Fan-in)

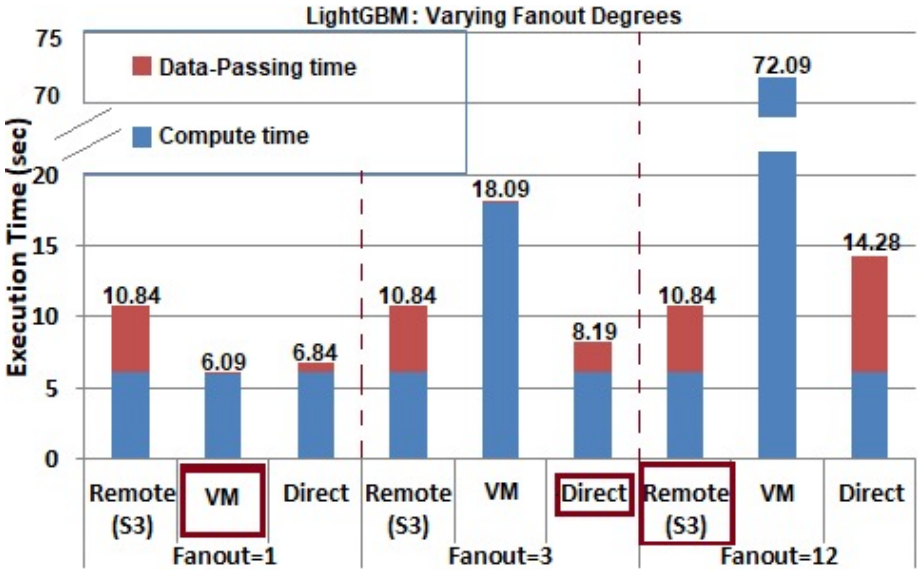


Copies the local state after λ_1 executes to **VM-2** before executing λ_2

- Data-passing speed depends on **VM-1** and **VM-2** bandwidths
- If there exists more than one instance of λ_2 (i.e., Fanout), **VM-1**'s network bandwidth becomes the bottleneck

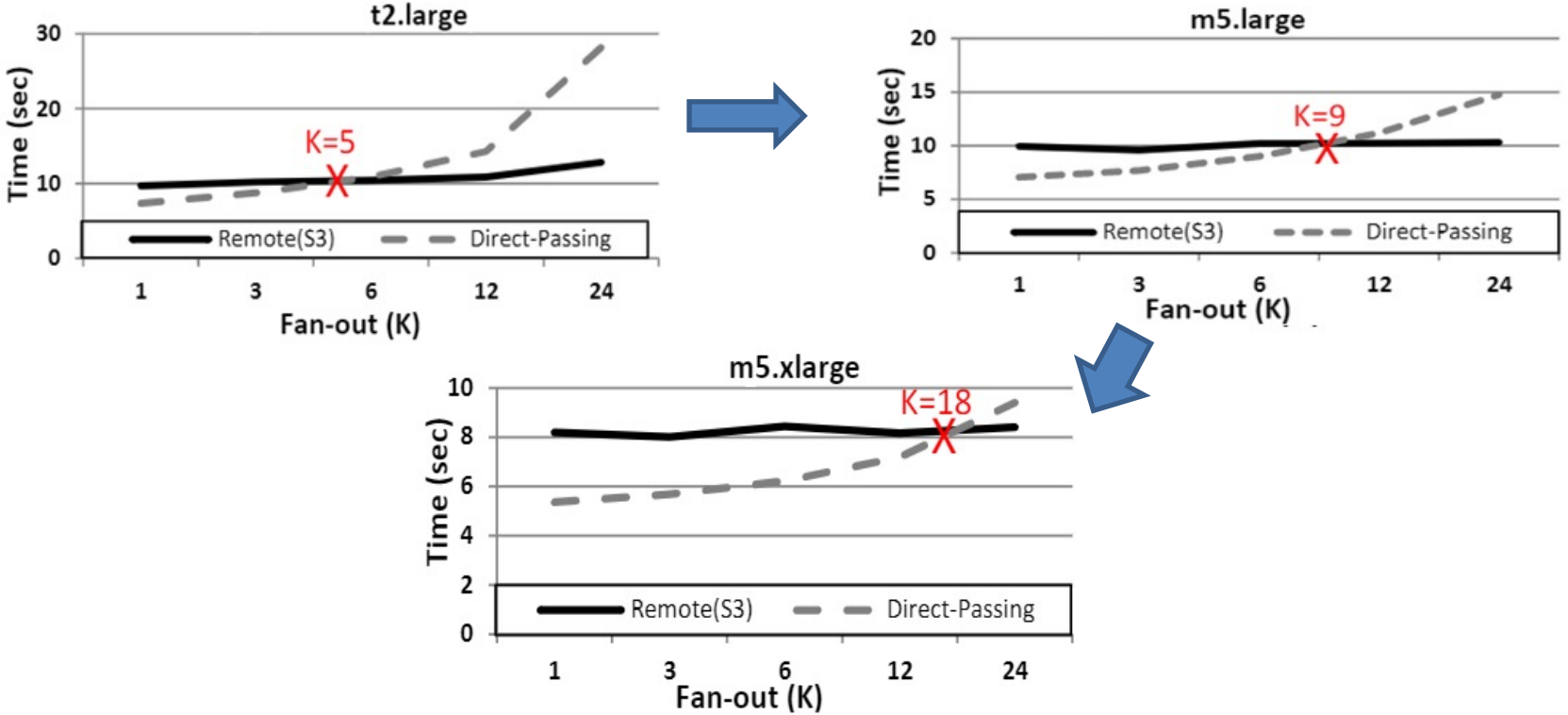
Data-passing performance trade-off





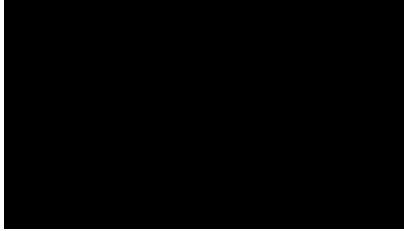
Direct-Passing vs Remote Storage

- With higher network bandwidth, the crossover point between Direct-passing and Remote-passing shifts to higher fanout values



OUR SOLUTION: SONIC

Hybrid Data-Passing Approach



- ***SONIC*** **jointly** optimizes the lambda placement for every function, and data-passing method for every edge in the DAG
- First, we profile the DAG and monitor the following metrics:
  - Memory footprint for every function
  - Execution time for every function
  - Input/output file size for every function
  - Fanout degree in every stage
- These parameters vary w.r.t. the DAG's input size
  - For example, analyzing a 1 min video vs. 30 min video
- We use these parameters to identify the best data-passing method and the corresponding lambda placement for each pair of dependent stages in the DAG

SONIC'S API: Data-passing Abstraction

1. **SONIC** abstracts the selection of data passing methods from application developers

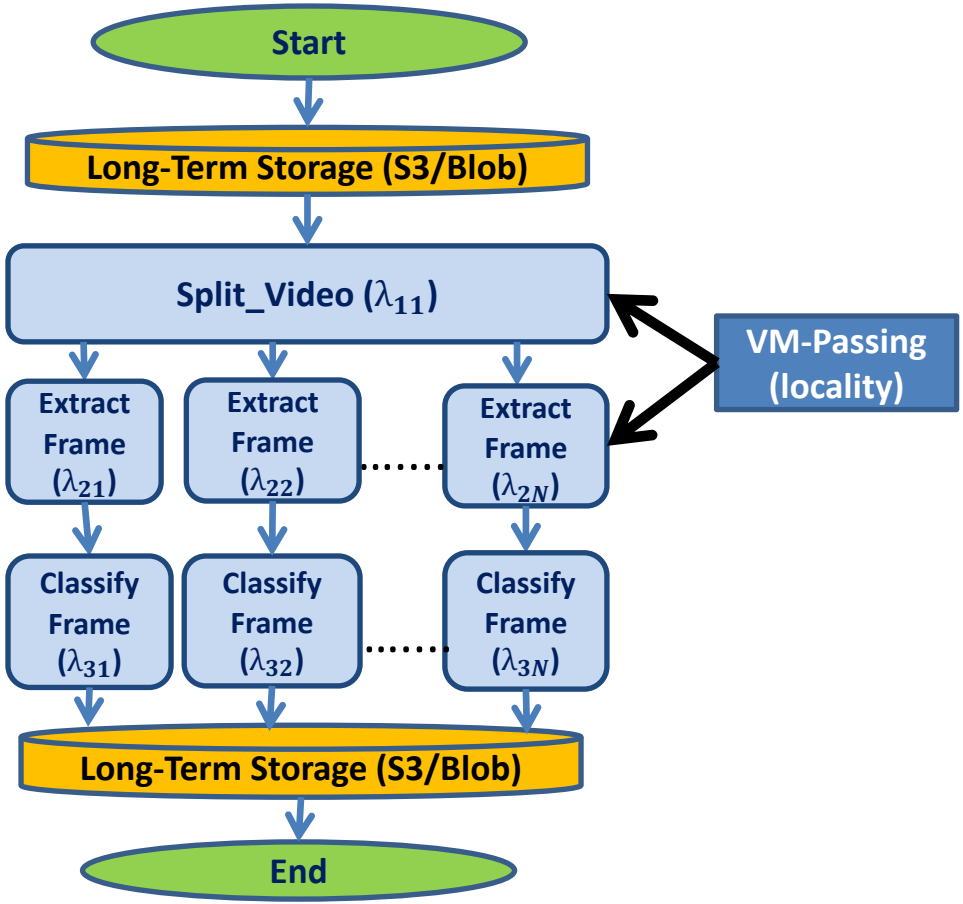
2. Functions write intermediate data to files using a standard file API(read and write), like writing to local storage

3. All λs within a job share a file namespace

4. If an application DAG has an edge $\lambda_s \rightarrow \lambda_r$, SONIC ensures that all of λ_r's input files are present in its local storage before it starts execution.

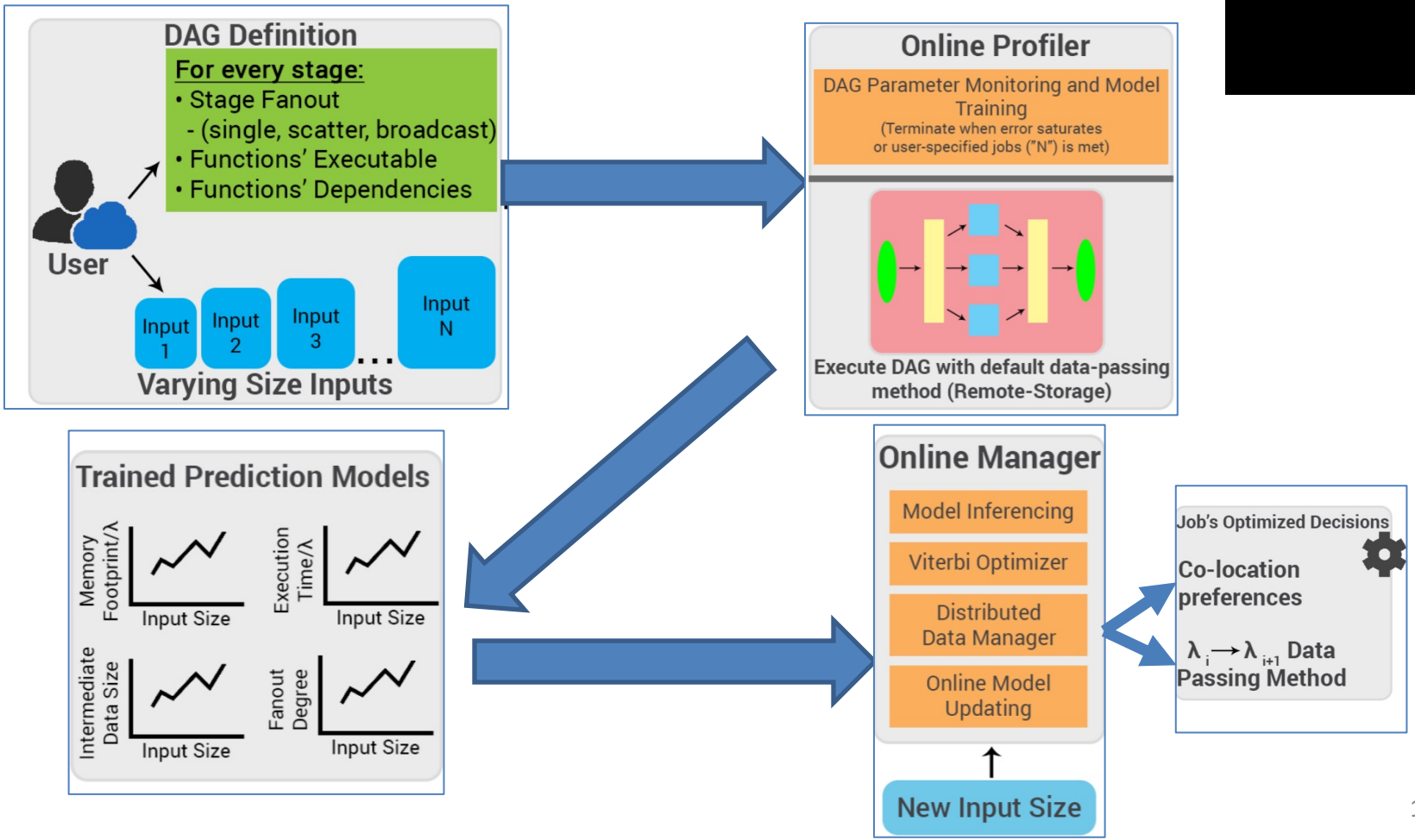
5. Therefore, λ_r reads its input files from the same path as the one that λ_s wrote the files to.

Greedy Data-Passing Decisions: Pitfalls



1. If we select "VM-Passing" between Split and Extract, all the extracted frames will reside on the same VM
2. This will cause passing between Extract and Classify to be either:
   1. VM-Passing: sacrifices parallelism as we cannot fit all Classify invocations in the same VM
   2. Direct or Remote: Bounded by the single VM's bandwidth and slow
3. Alternatively, we could have selected a non-optimal decision between Split and Extract to minimize the end-to-end latency
4. Specifically, using Direct-passing spreads the extracted frames on several VMs, allowing VM-passing to Classify without sacrificing parallelism

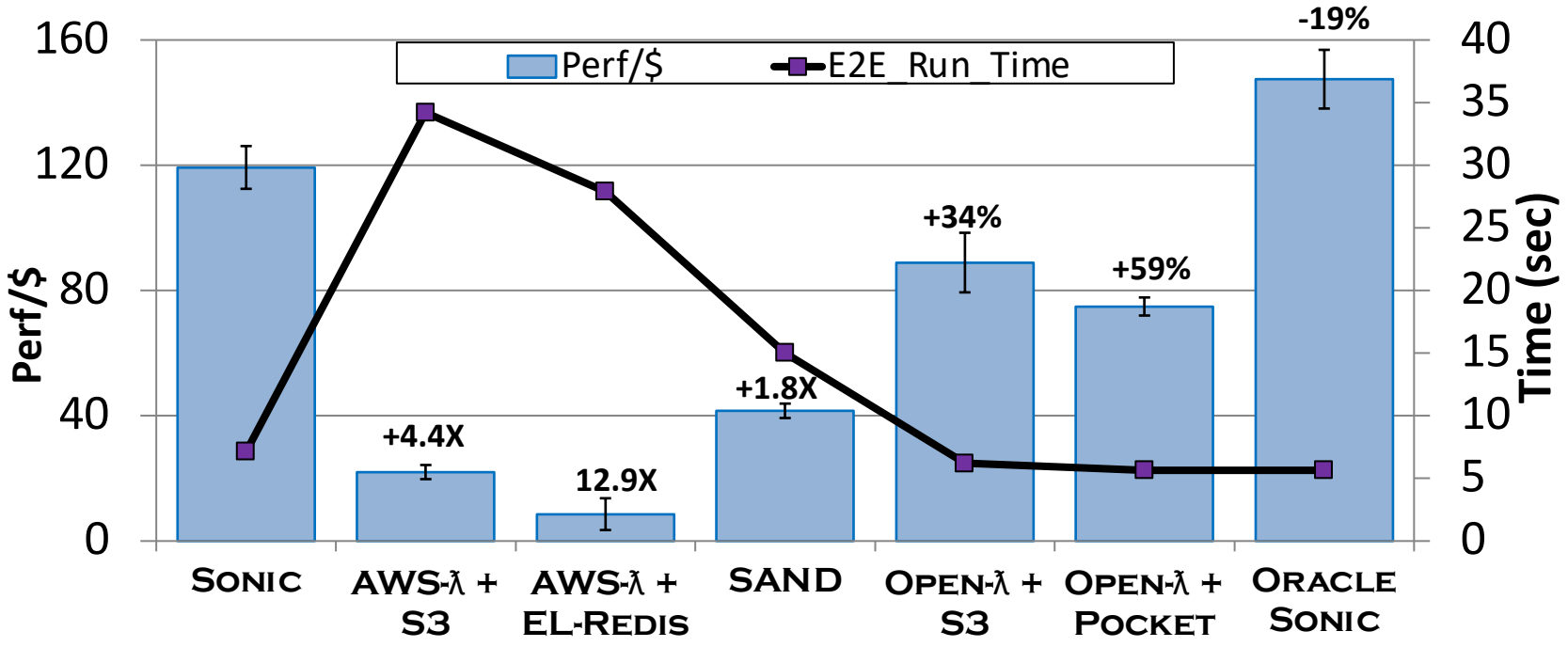
SONIC: Design Overview



Evaluation: Baselines

- OpenLambda [HotCloud'16] + S3: OpenLambda framework deployed on EC2 with S3 as its remote storage. A new VM is created to host each λ in the DAG.
- OpenLambda [HotCloud'16] + Pocket [OSDI'18]: We use Pocket's default storage tier (DRAM) with r5.large instance types.
- SAND [ATC'18]: Leverages data locality by allocating all lambda functions on a single host with rich resources.
- AWS-λ: The commercial FaaS platform using two different remote storage systems: S3 and ElastiCache-Redis.
- Oracle-SONIC: This is SONIC with fully accurate estimation of DAG parameters and no data-passing latency (mimicking local running of all functions).

Evaluation: E2E Latency and Cost



Perf/$ represents the latency normalized by Cost (in $) $\left(\frac{1}{Latency} \cdot \frac{1}{Cost}\right)$

Content Sensitivity

- Our approach uses the job's input *size* only to predict the DAG execution parameters.

- This allows generalizing without performing any application-specific processing.

- In some applications, the execution parameters are also dependent on the input *content*.
  - For example, the intermediate chunk sizes (in MB) in our video analytics application will vary based on the video's bitrate (**video quality**).

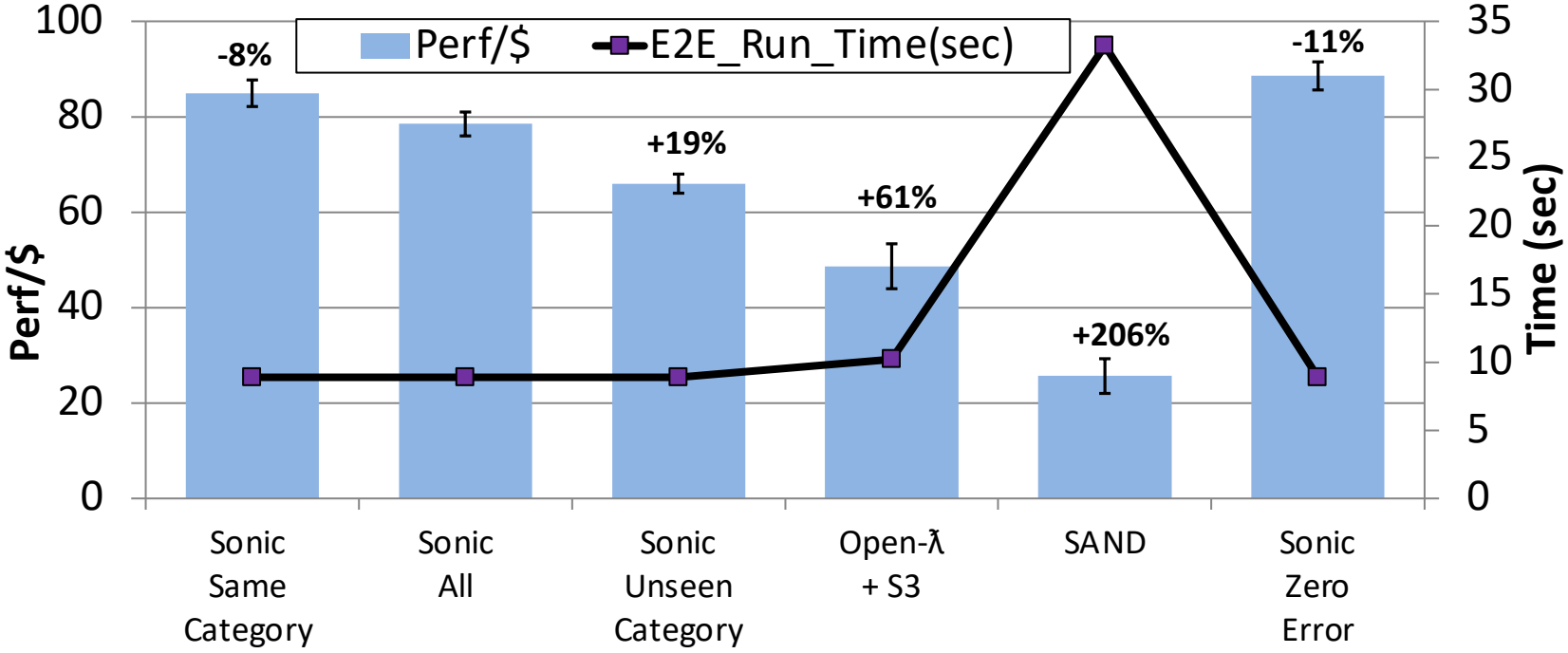
- We want to evaluate how sensitive is ***SONIC*** to this content sensitivity
  - For example, what is the performance of ***SONIC*** executing with test videos different from training?

Content Sensitivity (Cont.)

- First, we collect 60 YouTube videos from each of the following categories (News, Entertainment, Nature, Sports, and Cartoon)

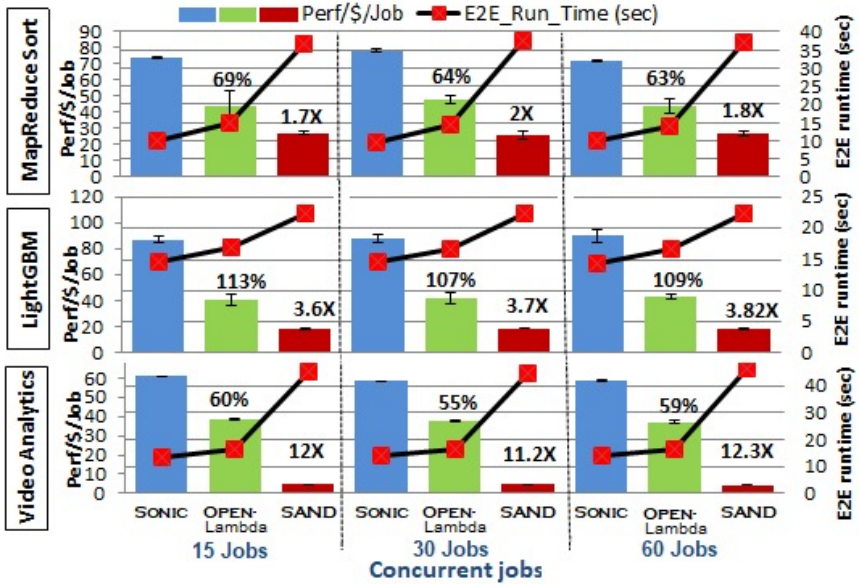
- We compare SAND and OpenLambda+S3 to the following variants of ***SONIC***:
  - Same Category: Test videos are from the same category as the training videos (Sports)
  - All Categories: Training videos are sampled from all categories, including the testing video category
  - Unseen Category: All training videos are from *News* category (has a 25% lower bitrate than the Sports category on average)
  - Zero Error: our approach executing with perfect knowledge of the exact execution parameters

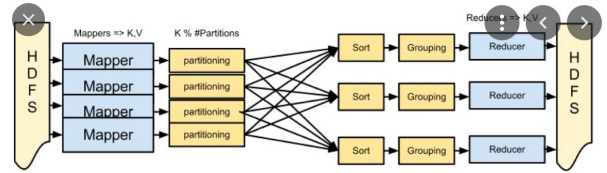
Evaluation: Content Sensitivity

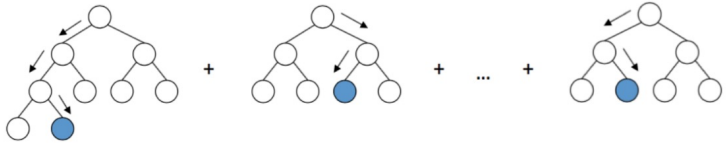


Perf/$ represents the Cost-normalized latency $\left(\frac{1}{Latency} \cdot \frac{1}{Cost}\right)$

Evaluation: Scalability







Conclusion

1. Data passing among serverless functions in an application is challenging
2. We studied 3 different data-passing options between serverless functions and showed that no single method prevails under all conditions (input sizes, network bandwidth, etc.)
3. We present **SONIC** a dynamic and hybrid approach to select the best global data passing method and lambda placement serverless workflows
4. Our solution outperforms all baselines in terms of Cost-normalized latency without sacrificing the raw latency

Ongoing Work

1. How to handle content-dependence in application DAGs
2. How to handle dynamic control flows

Thank You!

Funding:

- NIH R01 (2016-2022)
- NSF (CNS): Collaborative Research: Computer System Failure Data Repository to Enable Data-Driven Dependability
- NSF (CNS)/NIFA: Secure CPS for Real-time Agro-Analytics