WiseFuse: Workload Characterization, and DAG Transformation for Serverless Workflows

Ashraf Mahgoub\textsuperscript{1}, Edgardo Barsallo Yi\textsuperscript{1}, Karthick Shankar\textsuperscript{2}, Eshaan Minocha\textsuperscript{1}, Sameh Elnikety\textsuperscript{3}, Saurabh Bagchi\textsuperscript{1}, Somali Chaterji\textsuperscript{1}

\textsuperscript{1}: Purdue University; \textsuperscript{2}: Carnegie Mellon University  \textsuperscript{3}: Microsoft Research
Introduction: Serverless

- Serverless Functions:
  - Users write the code, and platform deploys and executes the function
  - *Pay-as-you-go* model
Introduction: Serverless Workflows

- Serverless DAG Example: Video Analytics Pipeline

➢ DAG execution is successful after all functions finish execution
DAG Execution

- Serverless DAG Example: **Video Analytics Pipeline**

- Current FaaS platforms execute each function in a *separate* VM
- Users specify the VM size for each function
Performance Bottlenecks

- Two major performance bottlenecks:
  1. Communication latency between in-series functions
  2. Computation skew among in-parallel invocations within the same stage

Diagram:
- Split
- Extract
- Video Chunk
- Classify\(_1\)
- Short Execution Time
- Classify\(_2\)
- Long Execution Time
- Next..
We characterize the impact of the two performance bottlenecks in Azure Durable Functions—DAGs executed over a duration of two weeks. Data set has over 630M invocations.

Top 5% most frequent DAGs:
- Constitute 95% of all DAG invocations
- Invocation rate ≥ 1.6K/day

➢ Workload Characterization (1/3)
➢ DAG structure: short but wide
  - Max Depth: 47 in-series stages
  - Max Width: 10.9K in-parallel invocations
• DAGs with intermediate data size $\geq 1$MB have 9.5$\times$ higher median latency than DAGs with size < 1MB.

• DAGs with skew $\geq 100$ have 17$\times$ higher median latency than DAGs with skew < 100

Real workload traces from Azure Durable Functions
• Direct communication between serverless functions is infeasible

• Accordingly, asynchronous communication through remote storage is used
  – Can add significant delay to the e2e latency since intermediate data sizes are growing
Our solution: **Fusion**

- By fusing the sending and receiving functions together, we can execute them in one VM and leverage local data passing $\Rightarrow$ reduce the e2e latency

➢ Challenge: increases cost if the functions have different resource requirements
Bundling (1/2)

- Each invocation execute in a separate VM
- Straggler dominates the e2e latency
Our solution: **Bundling**

- Execute both invocations in one VM
- Straggler gets additional resources after fast invocation finishes execution
Selecting the bundle size

- Bundle size = 1

Start

- Classify $y_1$
- Classify $y_2$
- Classify $y_3$
- Classify $y_4$

End
Selecting the bundle size

- Bundle size = 2

Start → Classify$_1$ → End
Start → Classify$_2$ → End
Start → Classify$_3$ → End
Start → Classify$_4$ → End
Selecting the bundle size

• Bundle size = 4
Summary of Main Insights

- **Workload Characterization:**
  - Top 5% most frequent DAGs constitute 95% of all DAG invocations
  - Serverless DAGs are short but wide

- **Two major performance bottlenecks:**
  - Communication latency between in-series functions → Fusion
  - Computation skew among in-parallel invocations → Budling
**WISEFUSE’s Design (1/4)**

- First, the DAG Profiler captures the execution time for each function and the entire DAG
  - Information already collected by the platform for billing

- We represent the execution time for each function and the entire DAG as a distribution (CDF)
  - Captures the runtime variability
WISEFUSE’s Design (2/4)

• Second, our modeler breaks down the function’s runtime into download, processing, and upload components

• Our modeler also takes into account the correlation between in-series and in-parallel functions

• We use the function profiles to estimate e2e latency, and estimate the impact of Fusion and Bundling
Afterwards, the DAG Transformer identifies:

- In-series stages that experience high communication latency $\rightarrow$ can benefit from Fusion.
- Parallel stages that experience execution skew $\rightarrow$ can benefit from Bundling
Finally, the Transformer uses **Dynamic Programming** to find the best execution plan for the DAG.

The Execution Plan describes:

1. Which stages to be Fused together
2. How many parallel invocations within a stage to be bundled together
3. The VM size to allocate for each function or function bundle
DAG Transformation

User-defined DAG

Transformed DAG

Start

Split_Video ($\lambda_{1,1}$)

Extract Frame ($\lambda_{2,1}$)

Extract Frame ($\lambda_{2,N}$)

Classify Frame ($\lambda_{3,1}$)

Classify Frame ($\lambda_{3,N}$)

End

Start

Split_Video ($\lambda_{1,1}$)

Extract Frame ($\lambda_{2,1}$)

Extract Frame ($\lambda_{2,N}$)

Classify Frame ($\lambda_{3,1}$ $\rightarrow \lambda_{3,4}$)

Classify Frame ($\lambda_{3,N-3} \rightarrow \lambda_{3,N}$)

End

Function Instance

Assigned VM
Evaluation

• We evaluate **WISEFUSE** on three applications on AWS Lambda

• Model estimation accuracy (using 300 profiling runs):
  
  ➢ Error in P95 e2e latency: ≤ 13%
  
  ➢ Error in estimating the impact of Fusion on e2e latency ≤ 3.4%
  
  ➢ Error in estimating the impact of Bundling on e2e latency ≤ 7%

• **Recall:** 95% of all invocations are for the top 5% most frequent DAGs
  
  – invocation rate ≥ 1.6 K per day
  
  – 300 runs are collected in 5 hours or less
We evaluate the following approaches on AWS Lambda:

1. Three latency target settings for WISEFuse

2. User-Max/User-Min: user-defined using max/min VM sizes

3. Sonic (ATC’21): Selects between three data passing methods for communication

4. Photons (SoCC’20): Colocates parallel invocations together to improve the memory utilization

5. FaastLane (ATC’21): Executes the entire DAG within a single VM
WISEFuse adjusts the execution plan based on the user specified latency target

➢ Higher latency target → Lower cost
WISEFUSE achieves 63% lower cost than User-Max
WISEFUSE achieves 61% lower P95 latency than User-Min
WISEFUSE achieves 47% lower P95 latency over Sonic.
WISEFUSE achieves a lower P95 latency by 62% compared to Photons.
WISEFUSE achieves 39% lower P95 latency compared to Faastlane.
Conclusion

• Workload characterization for serverless DAGs in Azure Durable Functions

• Two major performance bottlenecks in serverless DAGs:
  1. Communication latency between in-series functions → Fusion
  2. Computation skew among in-parallel function invocations → Bundling

• **WiseFuse** uses *Fusion* and *Bundling* operations to derive an optimized execution plan that meets a user-defined latency SLO with low cost

• Experimental evaluation on AWS Lambda
  – **WiseFuse** is superior in meeting tail latency targets with reduced costs
  – **WiseFuse** (Video Analytics) has lower P95 latency by 47%, 62%, and 39% compared to Sonic, Photons and Faastlane respectively