Meces: Latency-efficient Rescaling via Prioritized State Migration for Stateful Distributed Stream Processing Systems

Rong Gu, Han Yin, Weichang Zhong, Chunfeng Yuan, Yihua Huang

State Key Laboratory for Novel Software Technology, Nanjing University, China
Stream Processing Engines (SPEs) are widely adopted for real-time processing.
Motivation

SPEs usually call for dynamic rescaling due to varying workloads[1]

Rescaling in SPEs usually comes with state migration

Related Work

1. **Full Restart & Partial Pause**
   - Related works: Spark (*SOSP*’13), Heron(*SIGMOD*’15), Flink(*VLDB*’17), Flux(*ICDE*’03), Seep(*SIGMOD*’13)
   - Method: Pauses and resumes whole or part of the task when redistributing states
   - **Shortcomings:** blocks processing and causes **latency spikes** during rescaling

2. **Replicated-Dataflow**
   - Related works: ChronoStream(*ICDE*’15), Gloss(*ASPLOS*’18)
   - Method: Executes a new dataflow in parallel with the old one until finishing the state migration
   - **Shortcomings:** high resource usage during rescaling

3. **Proactive**
   - Related Work: Megaphone(*VLDB*’19), Rhino(*SIGMOD*’20)
   - Method: Adds extra behavior to non-rescaling periods to relieve the pressure during state migration
   - **Shortcomings:** incurs extra overhead to a non-rescaling dataflow
Full Restart & Partial Pause

Related works: Spark (SOSP'13), Heron (SIGMOD'15), Flink (VLDB'17), Flux (ICDE'03), Seep (SIGMOD'13)

Method: Pauses and resumes whole or part of the task when redistributing states

Shortcomings: blocks processing and causes latency spikes

Replicated-Dataflow

Related works: ChronoStream (ICDE'15), Gloss (ASPLOS'18)

Method: Executes a new dataflow in parallel with the old one until finishing the state migration

Shortcomings: high resource usage during rescaling

Proactive

Related Work: Megaphone (VLDB'19), Rhino (SIGMOD'20)

Method: Adds extra behavior to non-rescaling periods to relieve the pressure during state migration

Shortcomings: incurs extra overhead to a non-rescaling dataflow

Existing state migration approaches suffer from latency spikes, or high resource usage, or major disruptions

Common limitations: not taking into account the order in which operator state migrates
Prioritized Migration

Example: a key-count stream processing job

- Source
- Count
- Sink
Prioritized Migration

- Wait for the arrival of its corresponding state
- Block subsequent records in the queue
Prioritized Migration

- Block all records until the migration ends
Prioritized Migration

- Minimize the time spent in the waiting queue
Prioritized Migration:

- **Hot keys**: those being processed or about to be processed by downstream operator tasks
- State of **hot keys** needs to be prioritized so that the stream processing proceeds without blocking
Meces: On-the-fly Rescaling via Prioritized State Migration

• Fetch-on-demand state accessing during rescaling
• Coordinated by control messages*

* Inspired by previous works:
**Meces: Design and Mechanisms**

**Fetch-on-demand State Accessing**

(a) Instances rescaling  
(b) Key space redistribution
Fetch-on-demand State Accessing

(1) Triggering controlling messages
Fetch-on-demand State Accessing

(2) Aligning phase
Meces: Design and Mechanisms

Fetch-on-demand State Accessing

Maintaining Exactly-once Semantics during the Migration Stage.
Meces: Design and Mechanisms

Finer Granularity of State Migration

Split Key-groups into Sub-groups
Meces: Design and Mechanisms

Finer Granularity of State Migration

Split one Migration stage into Gradual-Fetch steps.
Meces: Design and Mechanisms

Meces System Architecture

- Non-intrusive design: Not affecting non-rescaling periods
- Runtime code transparent to users: Little effort for code migration
Evaluation

Latency Performance during Rescaling

Compared Systems
- Flink (stopping the whole job when rescaling)
- Order-Unaware (online block-based state migration without order prioritization)

Scenario
- Key-count job
- Scale out after running for 600s

The latency peak of Meces is significantly lower.
Evaluation

Latency Performance during Rescaling

Workload
- NEXMark Q1~Q8

Meces lowers the latency peak by orders of magnitude.
Evaluation

Time breakdown during Rescaling

Workload
- Key-count Job

(a) Order-Unaware
(b) Meces
(c) Distribution of Migration-Cost

- Long-duration blocks are converted into short-duration fetch operations.
- Reducing the queuing cost for subsequent records.
Evaluation

Comparison with Megaphone [VLDB'19]

Workload
- Key-count Job

- Meces incurs no overhead during non-rescaling
- Meces reduces latency peak significantly during rescaling
Evaluation

Comparison with Rhino [SIGMOD'20]

Workload
- Key-count Job

(a) Rhino on Flink
(b) Order-Unaware
(c) Meces

• Meces reduces latency peak by one magnitude during rescaling
• Meces incurs no network overhead during non-rescaling
Conclusion

- Meces: an on-the-fly rescaling mechanism for stateful distributed stream processing engines
  - Prioritized migration of hot states
  - Coordination protocol based on control messages
  - A hierarchical state data organization and a gradual state migration
  - Implemented on top of Apache Flink
Thank You!

Rong Gu @ Nanjing University

gurong@nju.edu.cn