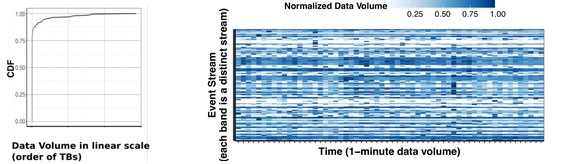
Move Fast and Meet Deadlines: Fine-grained Real-time Stream Processing with Cameo

Le Xu±, Shivaram Venkataraman‡, Indranil Gupta±, Luo Mai[fl], and Rahul Potharaju°

±University of Illinois at Urbana Champaign, ‡University of Wisconsin-Madison, [fl]University of Edinburgh, °Microsoft

Resource provisioning for data stream applications is challenging because:
- Long tail streams cause resource over-provisioning
- Temporal variation makes resource prediction difficult
- Latency requirements vary across jobs



Data volume varied by streams

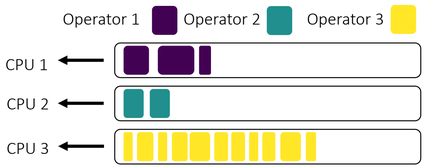
Data volume varied by time

Cameo's Goal: Real-time, fine-grained resource multiplexing for multi-tenant data stream applications. Cameo's contributions include:
- Deriving operator priorities dynamically by analyzing pending messages.
- Scheduling mechanism and programmable interface that enables plugged in scheduling policy.
- Scheduling at message-level granularity with low overhead.

WHY FINE-GRAINED SCHEDULING?

Slot-Based Approach

STORM, HERON, Spark Streaming



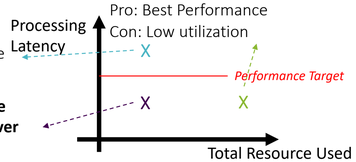
Event-Driven Approach

Orleans, akka

- Today's SPEs mostly adopt slot-based approach that binds operators to CPUs.
  - Actual resource requirement is difficult to predict
  - Requires to provision for peak load to avoid performance degradation
- Event-driven framework achieves high utilization by sharing work among CPUs.
  - Difficult to enforce performance isolation without application awareness

Event-Driven: No Resource Isolation:
Pro: High Utilization
Con: Compromised Performance

Slot-Based: Full Resource Isolation:
Pro: Best Performance
Con: Low utilization



Cameo: Providing performance guarantee without resource over provisioning

SCHEDULING WITH APPLICATION AWARENESS

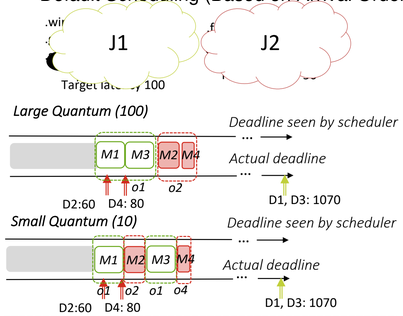
Message	M1	M2	M3	M4
Time of Origin	0	30	40	40

Message	M1	M2	M3	M4
Execution Cost	30	20	30	10

Message	M1	M2	M3	M4
Frontier Time	1000	30	1000	40

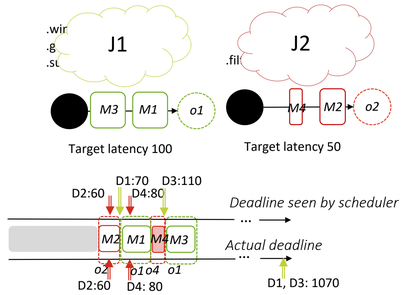
Message	M1	M2	M3	M4
Start Deadline	1070	60	1070	80

Default Scheduling (Based on Arrival Order)



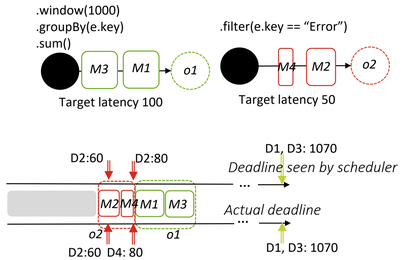
Scheduling based on arrival order causes violations for latency-sensitive jobs (2 violations).

Topology-aware Scheduling



Topology-awareness reduces violation(s) by identifying message(s) with tight latency constraints (1 violation).

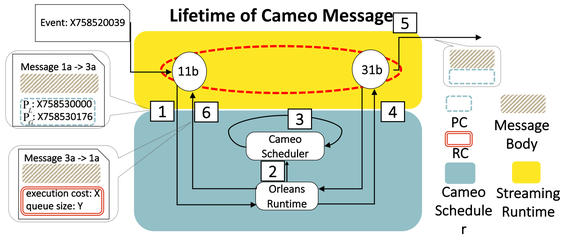
Topology and Semantic –aware Scheduling



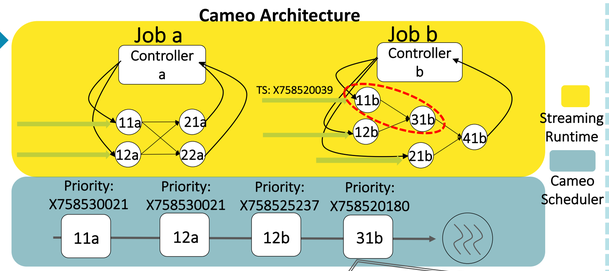
Semantic-awareness further identifies message(s) that can be delayed without violating latency constraints (0 violation).

PROGRAMMABLE SCHEDULING INTERFACE

Cameo enables data-driven, plug-and-play scheduling policy by using both **Scheduling Contexts** and user-defined **Context Handlers**:
- Handler functions are invoked **before** and **after** processing every request and acknowledgement message.
- Handler functions enforces priority generation strategy by **creating**, **interpreting** and **modifying** scheduling contexts.
- Message scheduler determines message and operator priority by observing scheduling context attached to messages. Message scheduler is policy-agnostic and therefore remains stateless.



1. PC generation (BuildCtxAtSource)
2. Request insertion
3. Request scheduled
4. Request message execute (BuildCtxAtOperator)
5. Trigger target operator and reply (PrepareReply)
6. Reply message execute (ProcessCtxFromReply)

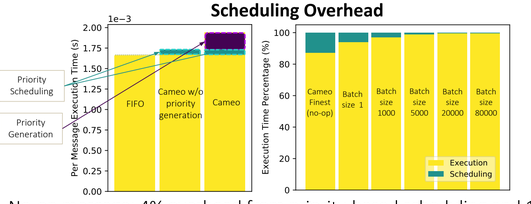


Local Priority	Global Priority	Pending Messages
X758520000	X758520180	Message<RID: 7216>, Message<RID: 7217>
X758530000	X758530176	Message<RID: 7218>, Message<RID: 7219>

Operator Priority: X758520180

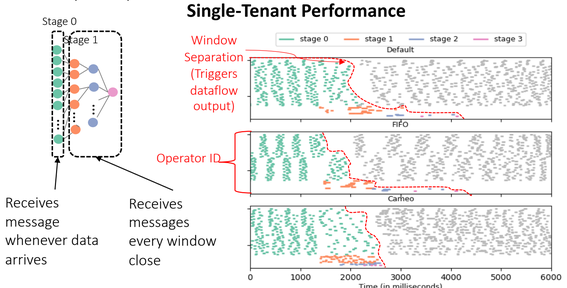
CAMEO IN ACTION

Scheduling Overhead



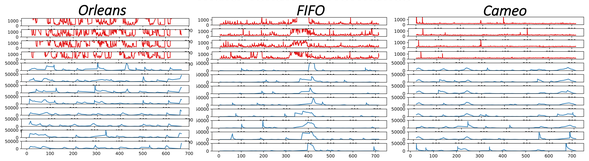
No-op message: 4% overhead from priority-based scheduling and 11% from priority generation. In practice, scheduling overhead is much smaller (<6.4%).

Single-Tenant Performance



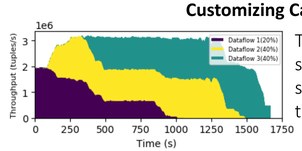
Cameo improves query latency (2.7 - 3.2X) by preventing messages influencing future results to be processed before messages targeting the next result.

Multi-Tenant Performance



Pareto Workload: Cameo weathers transient spikes by reducing tail latency (99 percentile) by 21 – 30X.

Customizing Cameo Strategy



Token-based proportional fair scheduling: Cameo implements scheduling algorithm that targets throughput objectives.