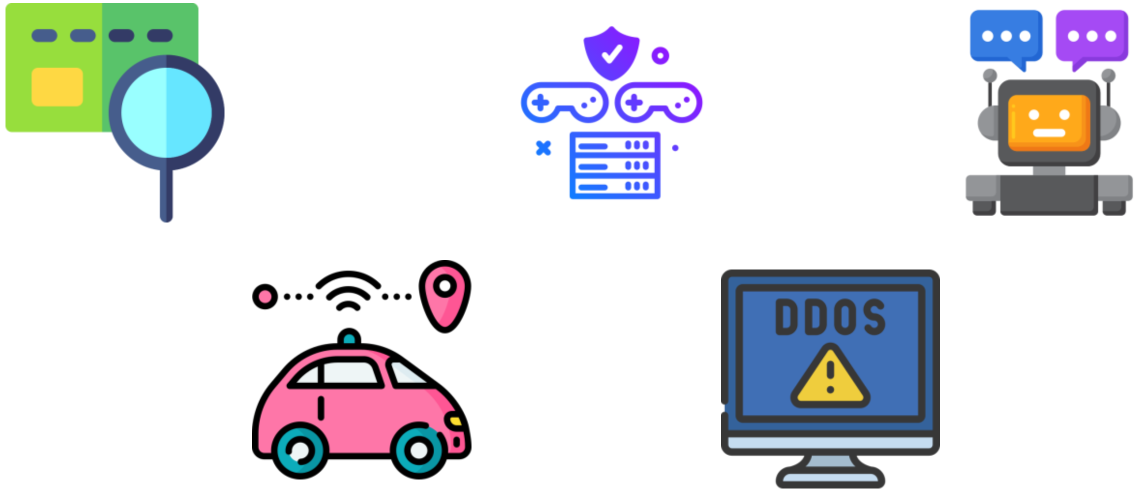
Fast Prototyping of Distributed Stream Processing Applications with *stream2gym*

Md. Monzurul Amin Ifath, Miguel Neves, Israat Haque
Dalhousie University

DALHOUSIE UNIVERSITY

Stream Processing Applications



Stream processing applications have increased by 300% in the last decade.

Stream Processing Applications

80% of the Fortune 100 companies currently use at least one stream processing platform.

Existing Testing Tools

- Preoperatory testing modules.
- Stream processing application benchmarking.
- Testing specific quality attribute.

Approach	Testing	Quality attribute	Stateful operation	Platform support	Open source
DiffStream	Differential	Performance Scalability	No	SPE	Yes
TRAK	Unit	Reliability	No	ESP	No
Gadget	Benchmarking	Performance Scalability	Yes	SPE, DS	Yes
Karimov	Benchmarking	Performance Scalability	Yes	SPE	No
Chintapalli	Benchmarking	Performance	Yes	ESP, SPE, DS	Yes

ESP = Event Streaming Platform,
SPE = Stream Processing Engine, DS = Data Store.

Existing Testing Tools

- Preoperatory testing modules.
- Stream processing application benchmarking.
- Testing specific quality attribute.

- Not suitable for system testing.
- Setting up from scratch – network and application.
- Require advance expertise.

Approach	Testing	Quality attribute	Stateful operation	Platform support	Open source
DiffStream	Differential	Performance Scalability	No	SPE	Yes
TRAK	Unit	Reliability	No	ESP	No
Gadget	Benchmarking	Performance Scalability	Yes	SPE, DS	Yes
Karimov	Benchmarking	Performance Scalability	Yes	SPE	No
Chintapalli	Benchmarking	Performance	Yes	ESP, SPE, DS	Yes
stream2gym	$\boxed{\text{System}}$	$\boxed{\text{Performance Reliability Scalability}}$	Yes	$\boxed{\text{ESP, SPE, DS}}$	Yes

ESP = Event Streaming Platform,
SPE = Stream Processing Engine, DS = Data Store.

Motivation

What if developers could promptly test their application pipeline on a local, low-cost, large-scale setup?

Motivation

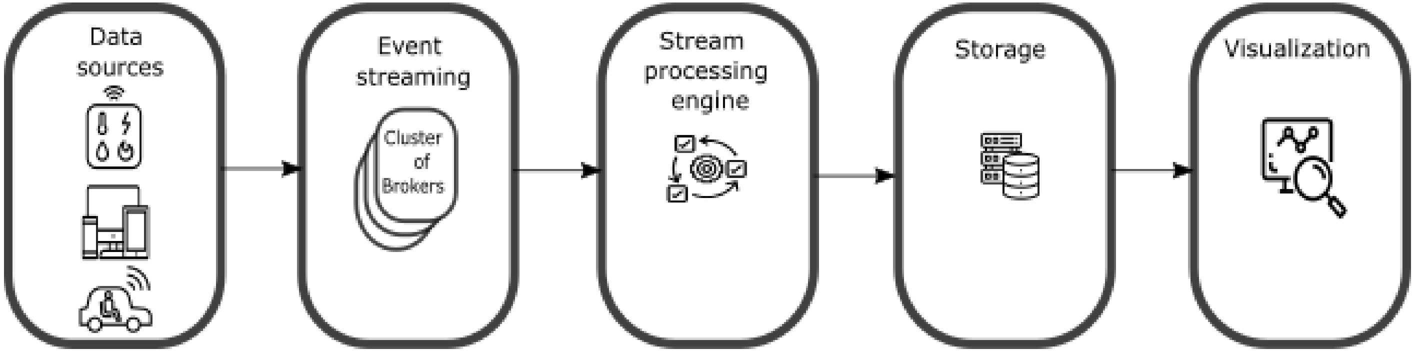
What if developers could promptly test their application pipeline on a local, low-cost, large-scale setup?

stream2gym

Outline

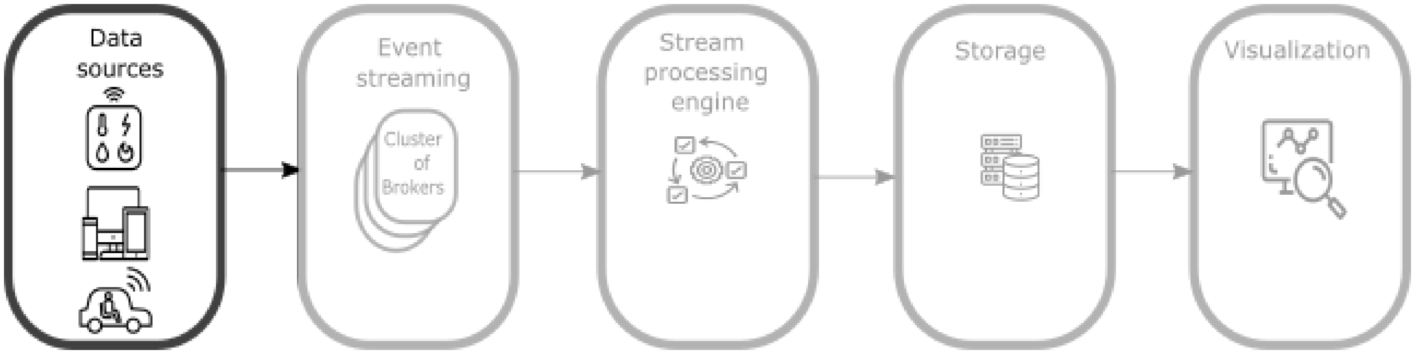
- Background
- Design
- Implementation
- Evaluation
- Conclusion

Stream Processing Pipeline (1)



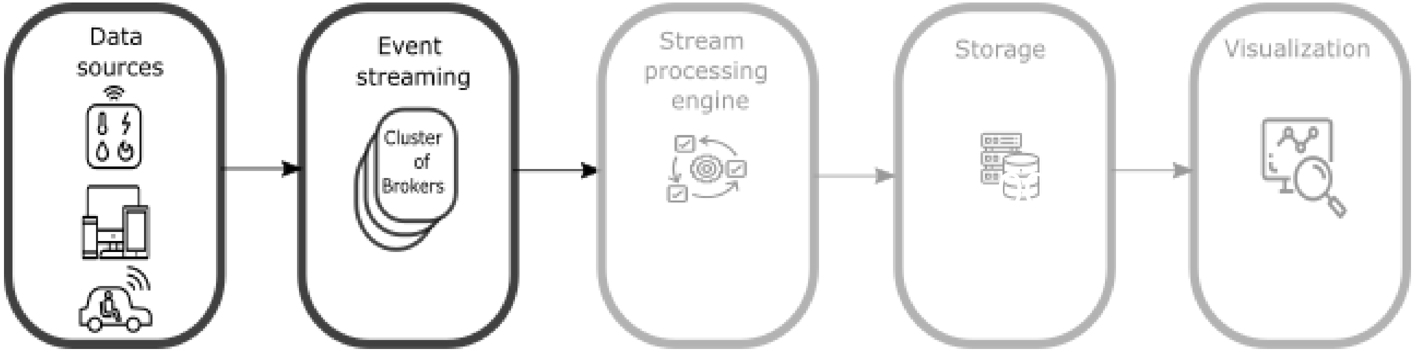
- Applications are typically developed in context of data processing pipeline.
- Pipelines commonly consist of data sources, streaming platforms, engines, storage, and visualization.

Stream Processing Pipeline (2)



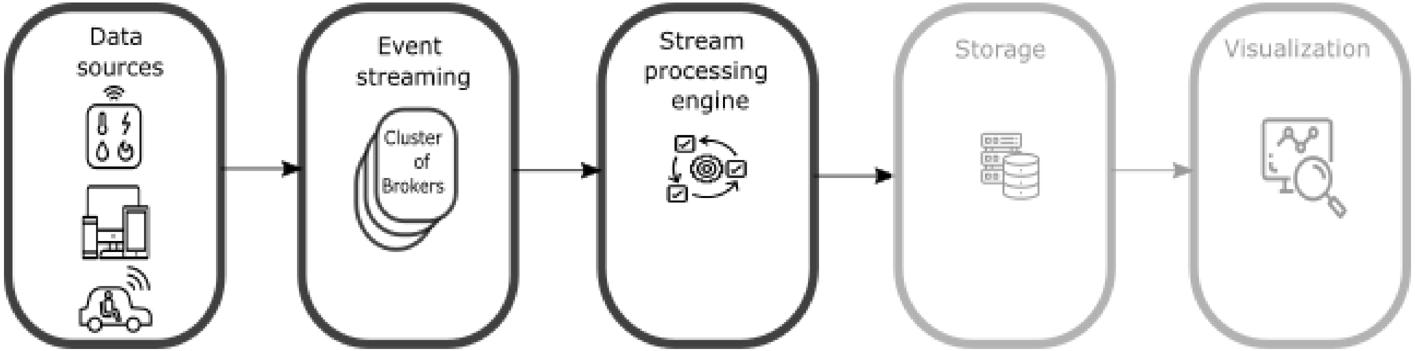
- Data sources (producers) are the origin of data.
- E.g. sensors, web servers, self-driving cars.

Stream Processing Pipeline (3)



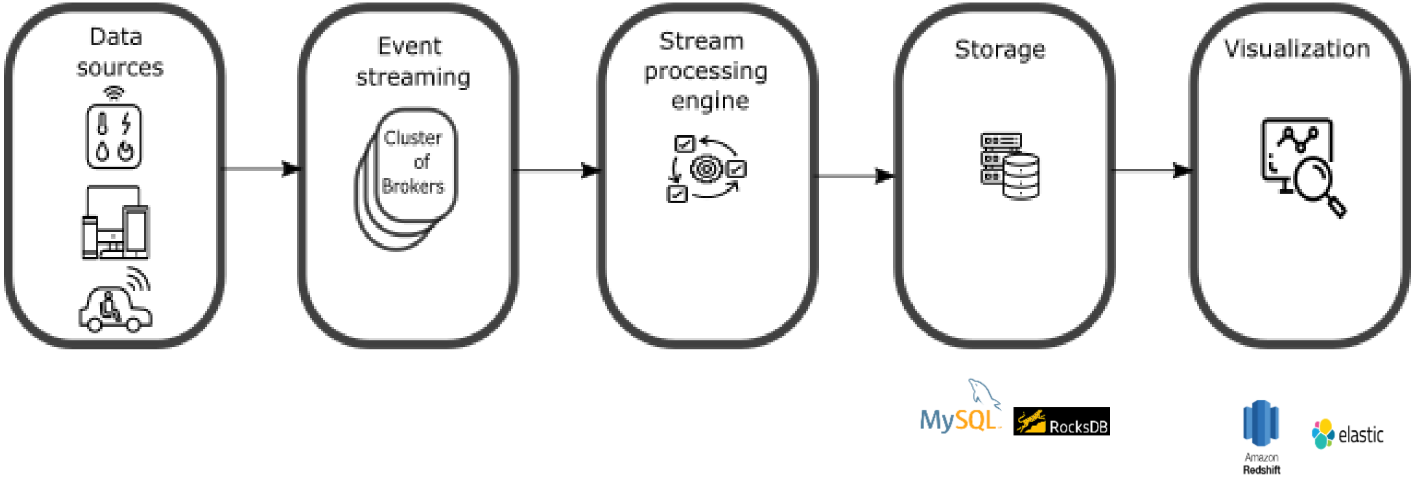
- Event streaming platform (ESP): transporter of data in the pipeline.
- Data or events are stored into different Topics.

Stream Processing Pipeline (4)



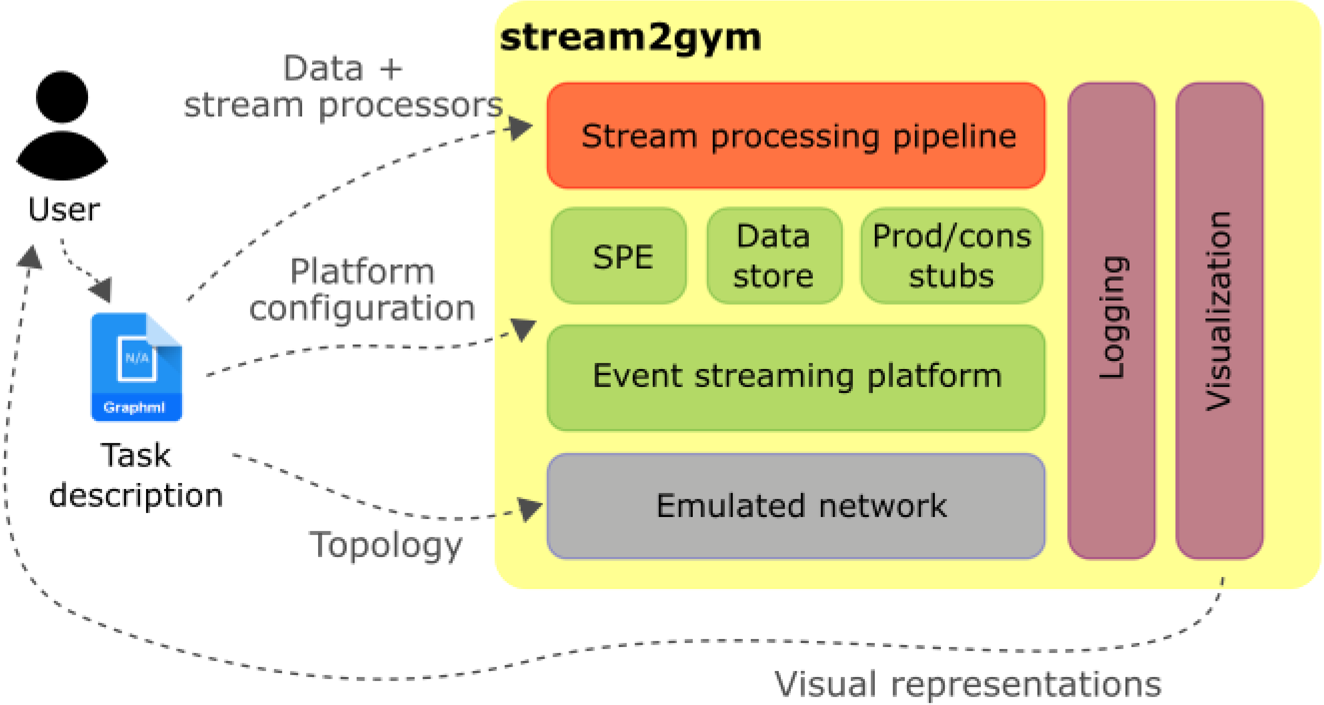
- Stream processing engine (SPE): real time data analysis component.
- E.g. operations performed: joining, aggregation, filtering, windowing.

Stream Processing Pipeline (5)



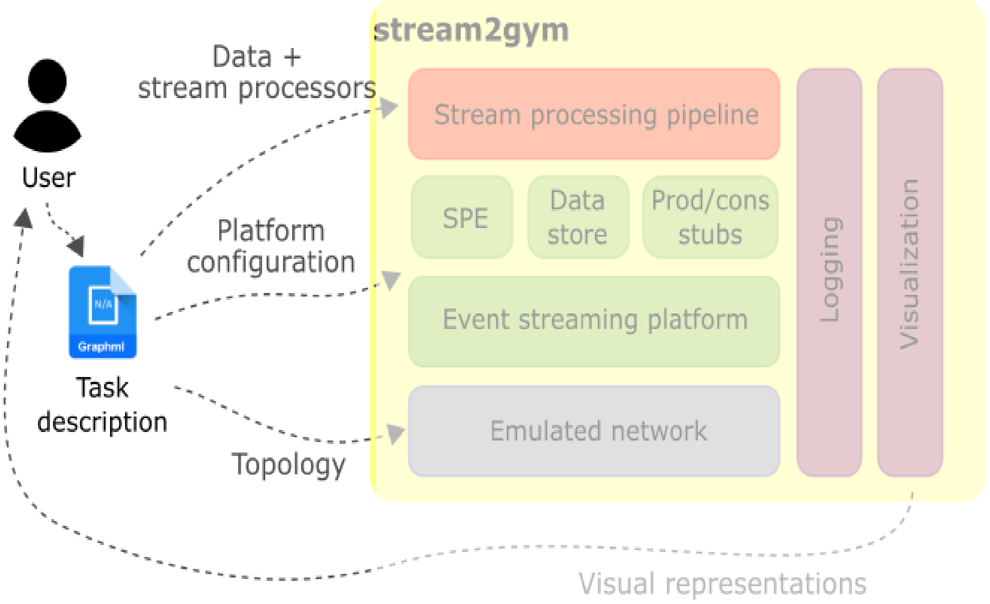
- Traditional consumers: storage and visualization components.
- Storage: persistent data storage, key-value store.
- Visualization component: dashboards.

Architecture & Workflow



Architecture & Workflow

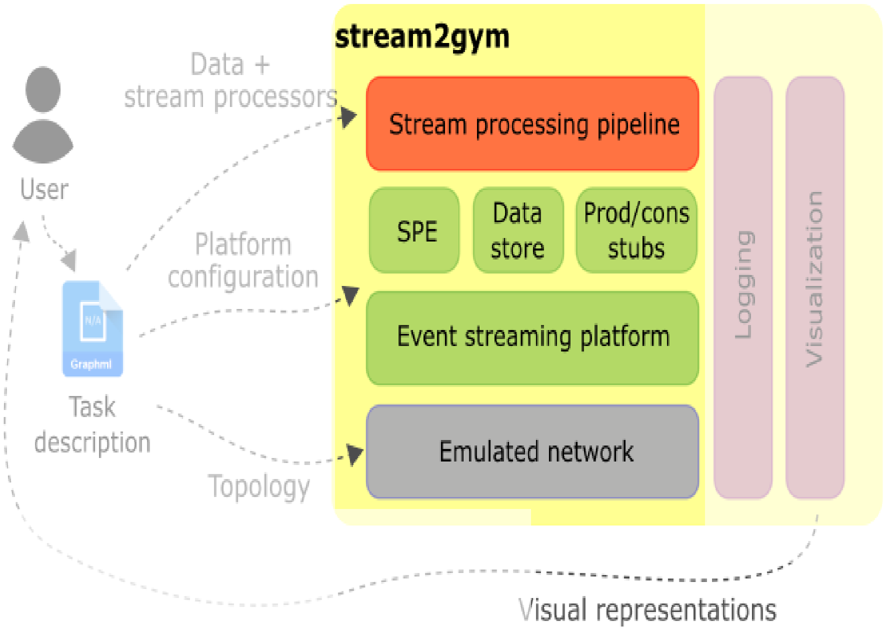
- Input parameters
  - Streaming application.
  - Configuration parameters.
  - Network topology.



PINet
Programmable and Intelligent Networking

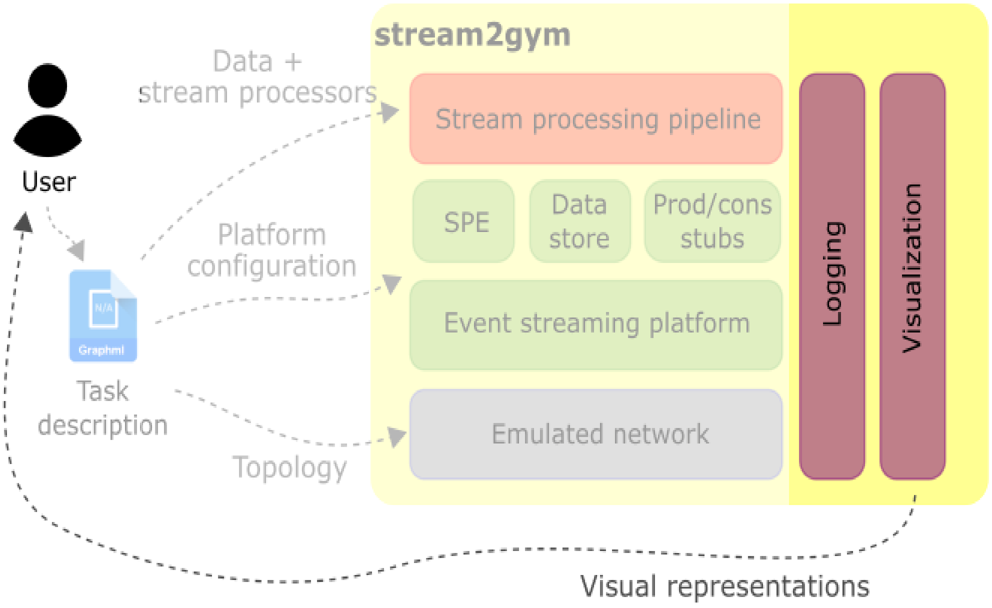
Architecture & Workflow

- Network instantiation over network emulator.
- Automatic mapping of brokers, data sources and sinks.
- ESP and SPE initiation.



Architecture & Workflow

- Logging facility to monitor application and network.
- Visual representations of logged statistics.



API

- Attributes
  - Graph
  - Node
  - Link

Graph attributes	Description
topicCfg	Topic configuration for the event streaming system
faultCfg	Fault configuration (e.g., link down) for reliability tests

Node attributes	Description
prodType	Data source type (used for data ingestion)
prodCfg	Data source configuration
consType	Data sink type (used for data consumption)
consCfg	Data sink configuration
streamProcType	Stream processing engine type (e.g., Spark, Flink, KStream)
streamProcCfg	Stream processing engine configuration
storeType	Data store type (e.g., MySQL, MongoDB, RocksDB)
storeCfg	Data store configuration
brokerCfg	Message broker configuration
cpuPercentage	Cap on overall system CPU usage

Link attributes	Description
lat	Link latency (in milliseconds)
bw	Link bandwidth (in Mbps)
loss	Link loss (%)
st	Source port
dt	Destination port

Implementation

- stream2gym implemented over Mininet.
- Currently supports
  - ❑ Apache Kafka.
  - ❑ Apache Spark Structured Streaming.
  - ❑ MySQL.
- ❑ Code Available (https://github.com/PINetDalhousie/stream2gym)

Use Cases

- Testing stream processing applications
- Emulating network conditions
  - Varying link delay
  - Network partitioning
- Reproducing research work
  - Video analysis framework
  - Traffic monitoring fore enterprise networks

Testing Stream Processing Applications

Application	Components	Features	LoC
Word count	5	Multiple stream processing jobs	167
Ride selection	5	Structured Data, Stateful Processing	142
Sentiment analysis	3	Unstructured Data	72
Maritime monitoring	4	Persistent storage	162
Fraud detection	5	Machine learning prediction	185

- Data source/data sink
- Message brokers
- Training and Inference systems
- Key-value store

(check project repository for application details)

Testing Stream Processing Applications

Application	Components	Features	LoC
Word count	5	Multiple stream processing jobs	167
Ride selection	5	Structured Data, Stateful Processing	142
Sentiment analysis	3	Unstructured Data	72
Maritime monitoring	4	Persistent storage	162
Fraud detection	5	Machine learning prediction	185

Depicts specific features each application deploys

Testing Stream Processing Applications

Application	Components	Features	LoC
Word count	5	Multiple stream processing jobs	167
Ride selection	5	Structured Data, Stateful Processing	142
Sentiment analysis	3	Unstructured Data	72
Maritime monitoring	4	Persistent storage	162
Fraud detection	5	Machine learning prediction	185

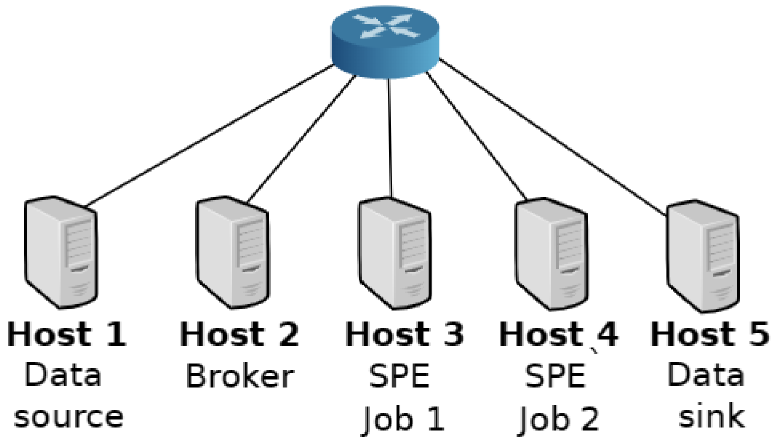
Defines the application using stream2gym API

- Deployed and tested five applications in *stream2gym*.
- Offers flexibility and efficiency.

Emulating Networking Conditions

Varying Link Delay

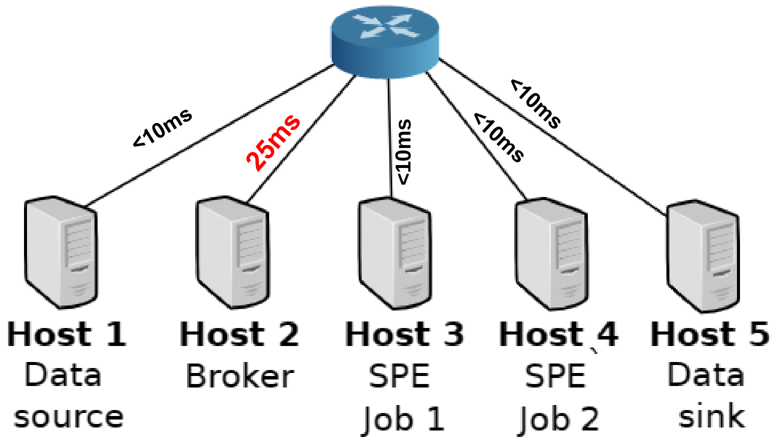
- Testing stream processing in geo-distributed setup is challenging.
- Easy customizations in topologies.
- Link delay increase for a single component.



Emulating Networking Conditions

Varying Link Delay

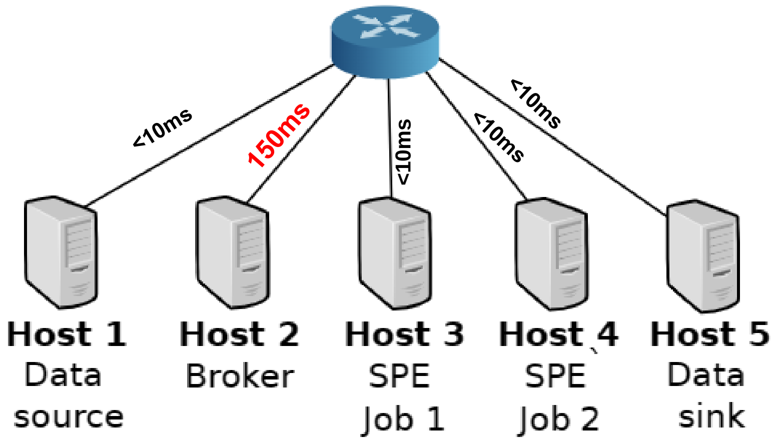
- Testing stream processing in geo-distributed setup is challenging.
- Easy customizations in topologies.
- Link delay increase for a single component.



Emulating Networking Conditions

Varying Link Delay

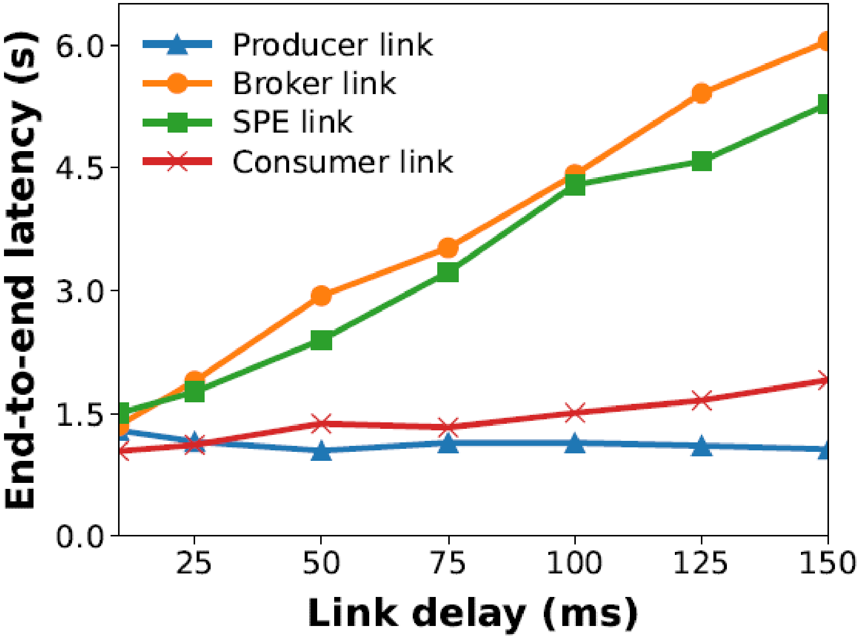
- Testing stream processing in geo-distributed setup is challenging.
- Easy customizations in topologies.
- Link delay increase for a single component.



Emulating Networking Conditions

Varying Link Delay

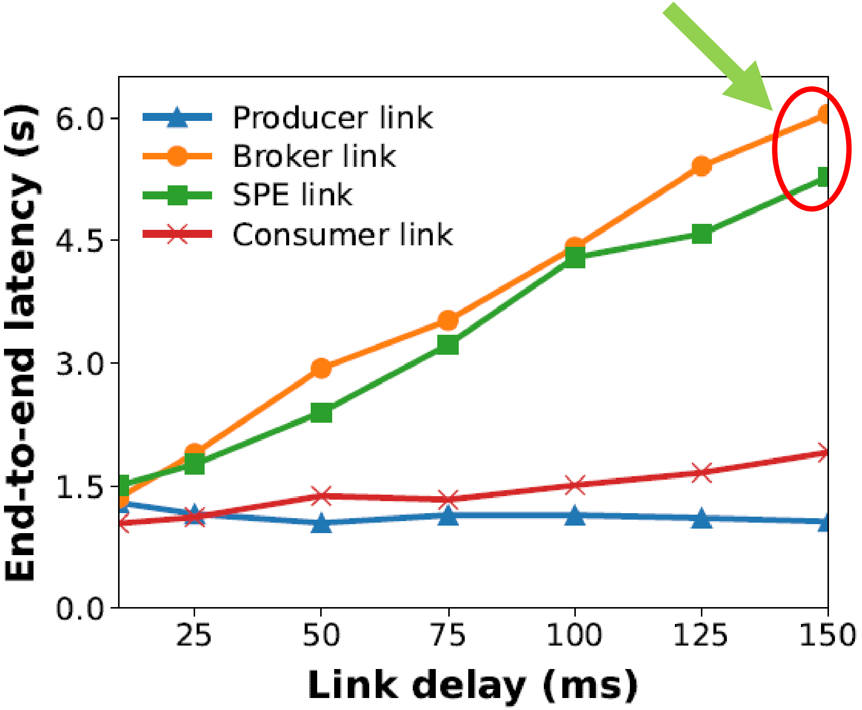
- Higher link delays impact performance of all components.
- Data broker and stream processing engine are more sensitive to networking conditions.
  - Due to higher communication frequency.
  - Distinct networking requirements for each component.



Emulating Networking Conditions

Varying Link Delay

- Higher link delays impact performance of all components.
- Data broker and stream processing engine are more sensitive to networking conditions.
  - Due to higher communication frequency.
  - Distinct networking requirements for each component.

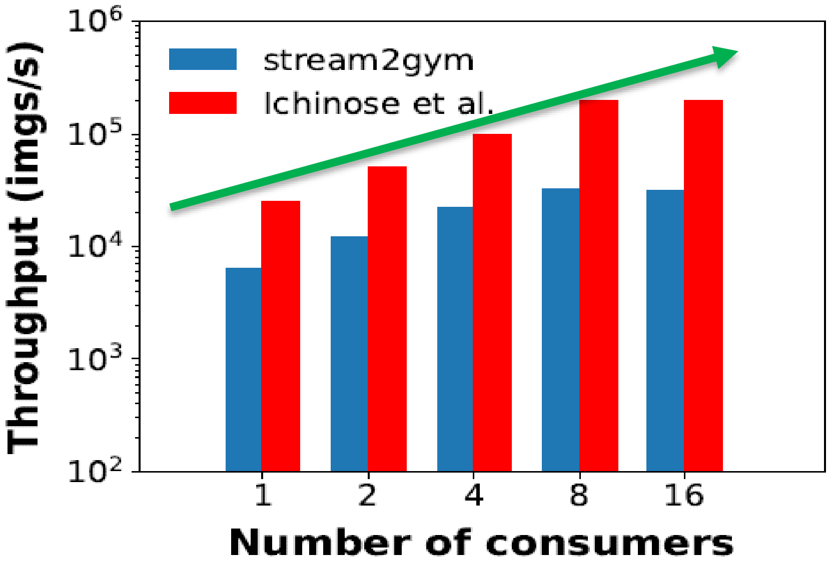


Reproducing Research Work

- Replicate research works from
  - Ichinose et al. [1] on video analysis framework.
  - Ocampo et al. [2] on traffic monitoring of enterprise networks.
- In both cases, *stream2gym* matches original paper results by showing similar patterns.
- Network emulator overhead may affect results slightly.

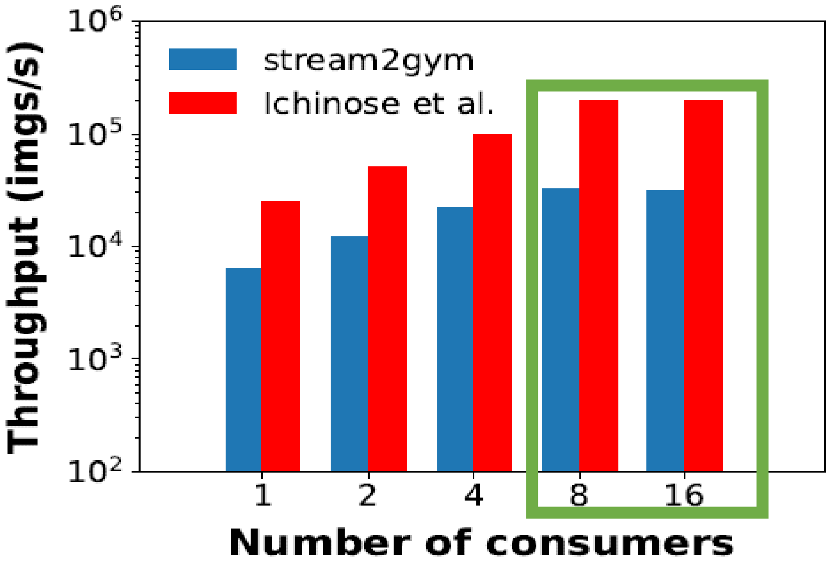
Reproducing Research Work

- Ichinose et al. [1]
  - Video processing in real-time.
  - Performance analysis of ESP in terms of increasing consumers.
  - Throughput increases up to 8 consumers, then plateaus.



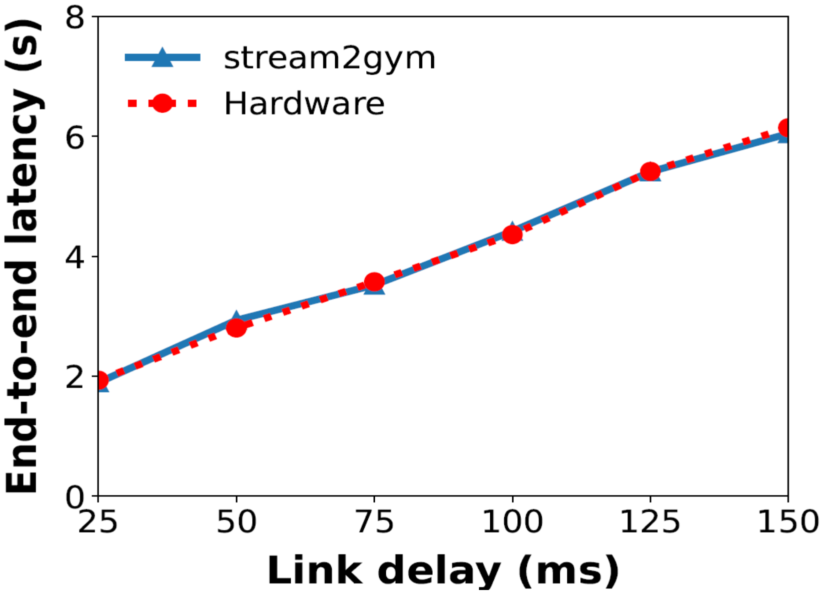
Reproducing Research Work

- Ichinose et al. [1]
  - Video processing in real-time.
  - Performance analysis of ESP in terms of increasing consumers.
  - Throughput increases up to 8 consumers, then plateaus.



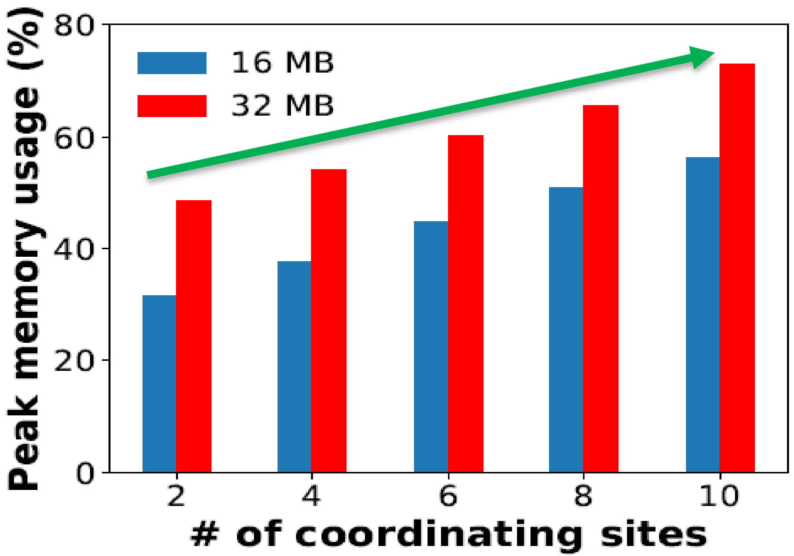
Accuracy

- *stream2gym* results match testbed results almost exactly.
- Conduct component wise series of experiments to confirm validity.



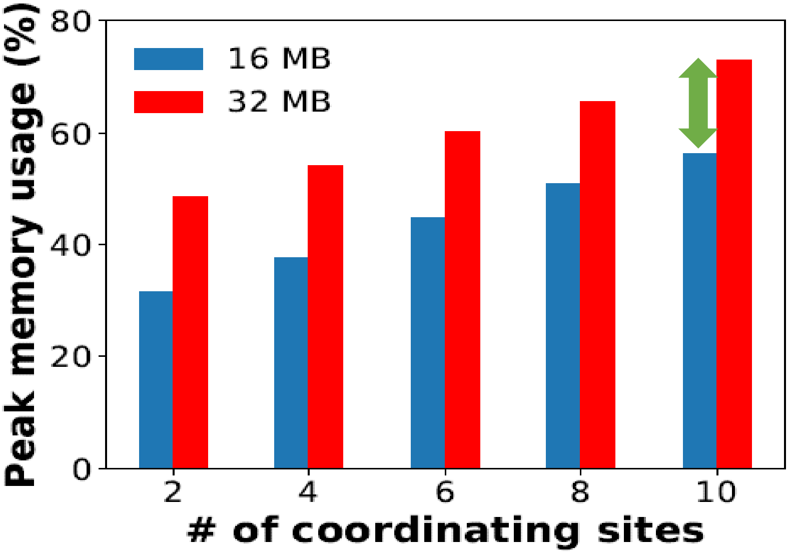
Resource Usage

- Insignificant increase in memory usage even with larger setup (~25%).
- Buffer size at producers affects memory consumption significantly (~18%).
  - Optimized parameter setup may accomplish scaled up topology accommodation.



Resource Usage

- Insignificant increase in memory usage even with larger setup (~25%).
- Buffer size at producers affects memory consumption significantly (~18%).
    - Optimized parameter setup may accomplish scaled up topology accommodation.



Conclusion

- Existing stream processing testing solutions face challenges in terms of providing end-to-end testing.
- *stream2gym* enables automated end-to-end testing by
  - Facilitating high level API for application developers.
  - Abstracting low level network infrastructure.
  - Providing accurate result while consuming negligible resource.
- Working towards
  - More stream processing tool adoption.
  - Automatic parameter tuning.

Questions

Thank You

Email: monzurul.amin@dal.ca

References

[1] A. Ichinose, A. Takefusa, H. Nakada, and M. Oguchi, “A study of a video analysis framework using kafka and spark streaming,” in 2017 IEEE International Conference on Big Data (Big Data), 2017, pp. 2396–2401.

[2] A. F. Ocampo Palacio, T. Wauters, B. Volckaert, and F. De Turck, “Scalable distributed traffic monitoring for enterprise networks with spark streaming,” in ECCWS2018, the 17th European Conference on Cyber Warfare and Security, 2018, pp. 563–569.

[3] M. M. A. Ifath, M. Neves, and I. Haque, “Fast prototyping of distributed stream processing applications with stream2gym,” ser. ICDCS ’23. New York, NY, USA: IEEE, 2023, accepted for the 43rd IEEE International Conference on Distributed Computing Systems.

[4] M. M. A. Ifath, M. Neves, and I. Haque, “Raptor: rapid prototyping of distributed stream processing applications at scale,” in Proceedings of the 17th International Conference on emerging Networking EXperiments and Technologies, ser. CoNEXT ’21. New York, NY, USA: Association for Computing Machinery, 2021, pp. 485–486.