STRETCH: Scalable and Elastic Deterministic Streaming Analysis with Virtual Shared-Nothing Parallelism

Hannaneh Najdataei, Yiannis Nikolakopoulos, Marina Papatriantafilou, Philippas Tsigas, Vincenzo Gulisano

13th International Conference on Distributed and Event-Based Systems
June 24-28, 2019, Darmstadt
Motivation

Improve performance by:

• **Scale Down** the amount of data (computing resources)
• **Scale Up** the computing resources on a node via parallel processing
• **Scale Out** the computing to distributed nodes

Phillip B. Gibbons, Keynote Talk IPDPS’15
Motivation

Phillip B. Gibbons, Keynote Talk IPDPS’15

Scale Up before Scale Out

Scale Up

Scale Out

Big Data:
Scale Down, Scale Up, Scale Out
**Scale Up** before **Scale Out**

- Often order of magnitude better performance if data fits in memory of multicore
- Multicores have 1-12 TB memory
- Even when data doesn’t fit, will still want to take advantage of Scale Up whenever you can

---

[Gedik et al., TPDS14][Cardellini et al., HPCS16][Carbone et al., VLDB17][De Matteis et al., PDP17]
Motivation

**Scale Up** before **Scale Out**
- Often order of magnitude better performance if data fits in memory of multicore
- Multicores have 1-12 TB memory
- Even when data doesn’t fit, will still want to take advantage of Scale Up whenever you can

Phillip B. Gibbons, Keynote Talk IPDPS’15

Big Data: Scale Down, Scale Up, Scale Out

Adjusting resources on node level for stateful streaming analysis
What is stream processing?
Data stream processing

Data -> Continuous Query -> Results

- Data flow
- Operators
Stream Processing Operators

**State** is the memory of the operator

- Stateless
- Stateful
Stream Processing Operators

*State* is the memory of the operator

- **Stateless**
  - E.g. filter

- **Stateful**

```
tuple <t,x>
```

```<3,1><2,4><1,3><4,3>```
Stream Processing Operators

State is the memory of the operator

- Stateless
  - E.g. filter

- Stateful
  - E.g. aggregate
Stream Processing Performance

- **Throughput**
  Number of tuples processed per time unit
Stream Processing Performance

- Throughput
- Latency
  Time difference between receiving a tuple and producing the corresponding results
Challenges

- Process large amount of data
- Handle fluctuations in data rate

Scalability and Elasticity

Data Stream Processing

Motivation
Stream Processing Scalability

• Pipeline parallelism
Stream Processing Scalability

- Pipeline parallelism
- Data parallelism

Determinism: Consistent results independent of tuples’ inter-arrival times
Stream Processing Elasticity

Provisioning

Decommissioning

operator

instance₁

instance₂

instance₃

instance₄

operator

operator

instance₁

instance₂

instance₃
Stream Processing Efficiency

Parallelism

Reconfiguration

Virtual
Shared-nothing

Shared-nothing

Shared

Architecture
Proposed Framework

- STRETCH
- Scalability and Elasticity
- Data Stream Processing
- Motivation
STRETCH Framework

Components:

• State manager
  • Virtual shared-nothing parallelism
Virtual Shared-nothing Parallelism
STRETCH Framework

Components:

• State manager
  • Virtual shared-nothing parallelism
• Elastic ScaleGate (ESG)
ScaleGate

- Sort concurrent arriving tuples based on timestamp
- Lock-free data structure
Elastic Gate

- Sort concurrent arriving tuples based on timestamp
- Lock-free data structure

✓ Changing number of readers/sources at runtime
STRETCH Framework

Control Unit

operator

State manager

Buckets

instance_1

instance_2

ESG_in

t=2

ESG_out

t=2

t=3

t=3

t=6

t=8

t=5

t=9

STRETCH Framework

Control Unit

operator

Buckets

State manager

instance₁

instance₂

ESG_{out}
STRETCH Framework
STRETCH Framework

Control Unit

operator

State manager

Buckets

ESG_{in} \quad t_s=8 \quad \text{ESG}_{out}

\text{if } \text{ state } 1 \quad \text{if } \text{ state } 2
STRETCH Framework
Performance Evaluation

- Setup: Intel Xeon E5-2695
- Use case: ScaleJoin
Performance Evaluation

- Setup: Intel Xeon E5-2695
- Use case: ScaleJoin
Conclusion

• Virtual shared-nothing parallelism
• Adaptive reconfiguration of processing units
• Intra-node resource utilization
• Deterministic execution

➢ Scale up/scale out
➢ Automatic control unit

Hannaneh Najdataei
✉ hannajd@chalmers.se