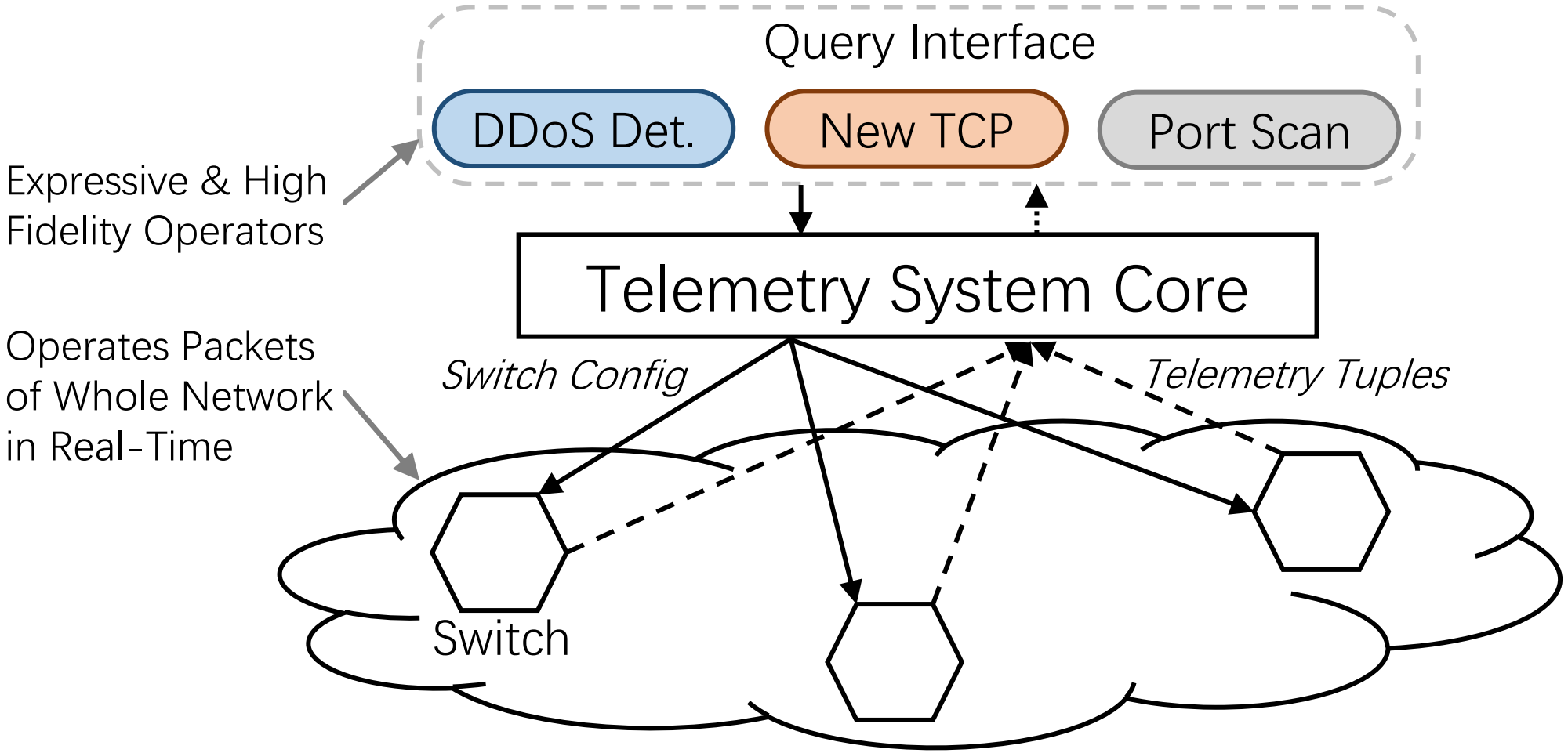
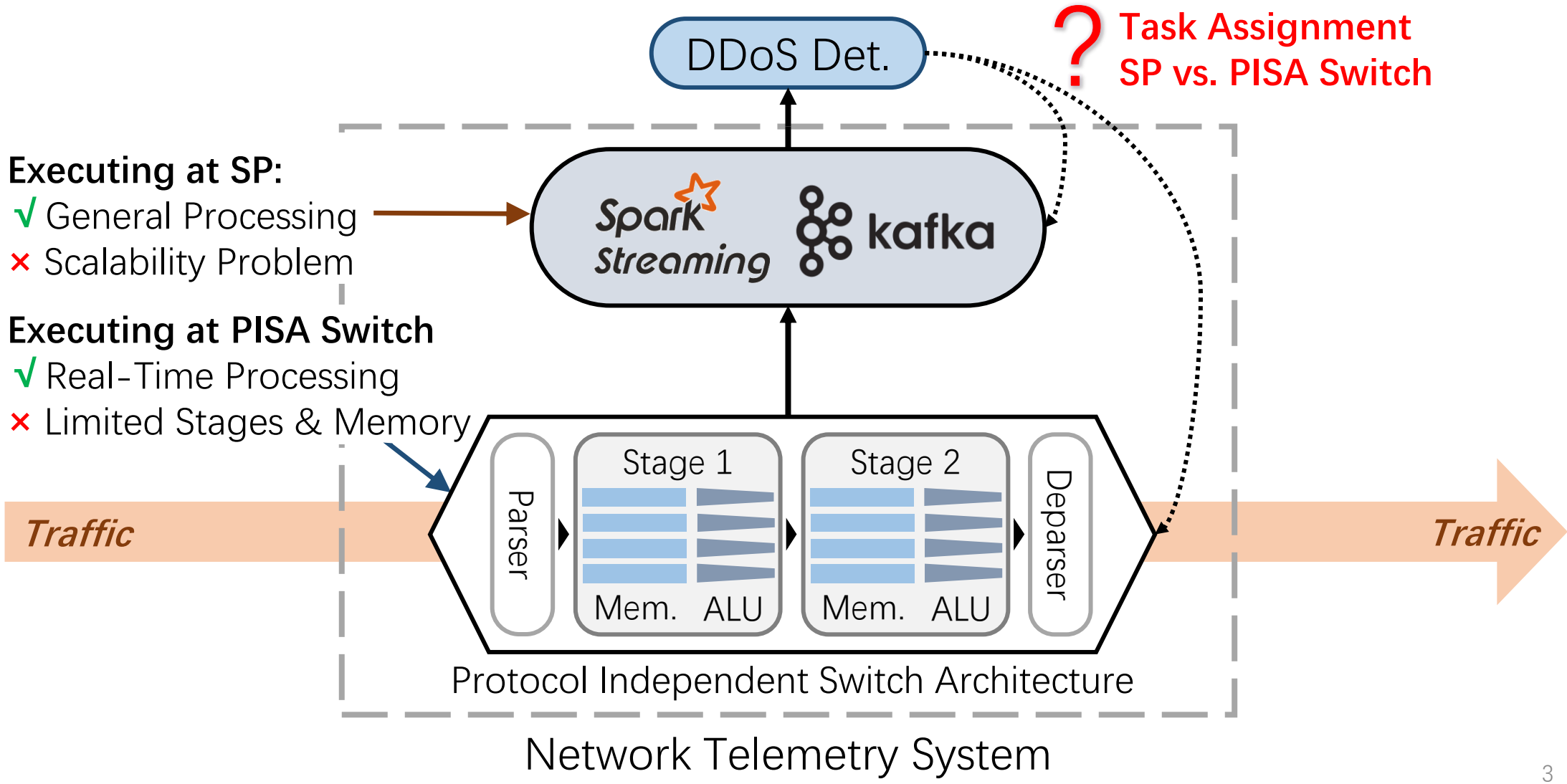
Concerto: Cooperative Network-Wide Telemetry with Controllable Error Rate

Yiran Li Kevin Gao Xin Jin Wei Xu

Network Telemetry Provides Useful Status Knowledge

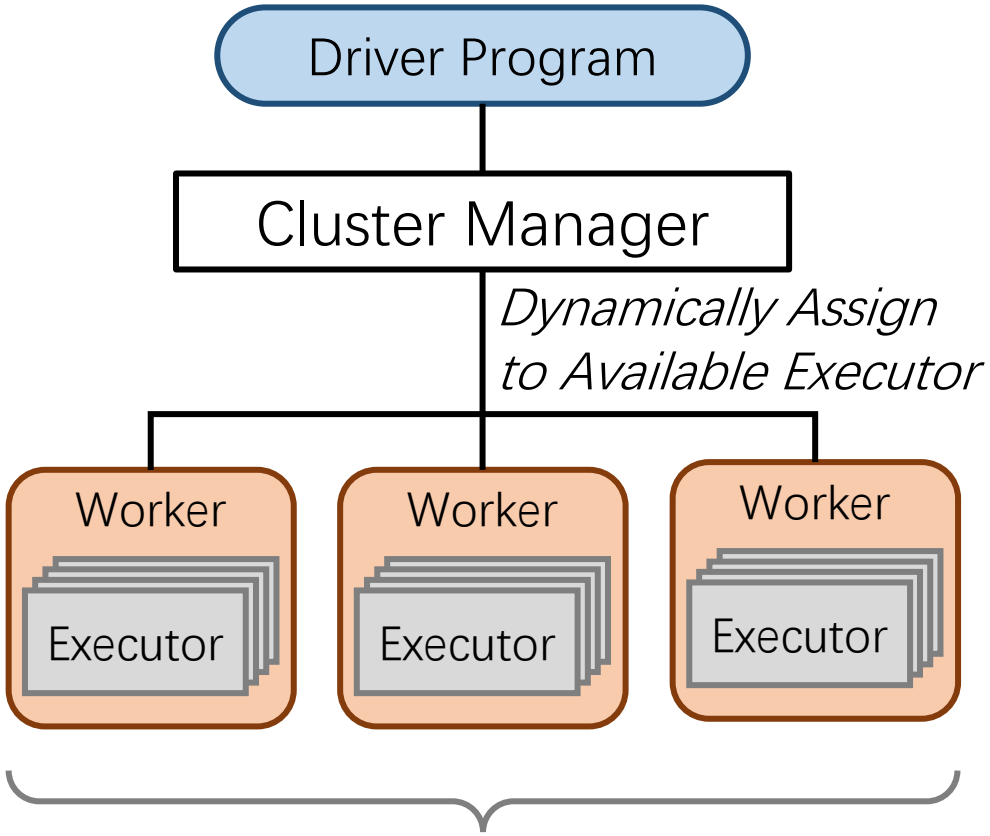


Executing Location: Stream Processor vs. PISA Switch



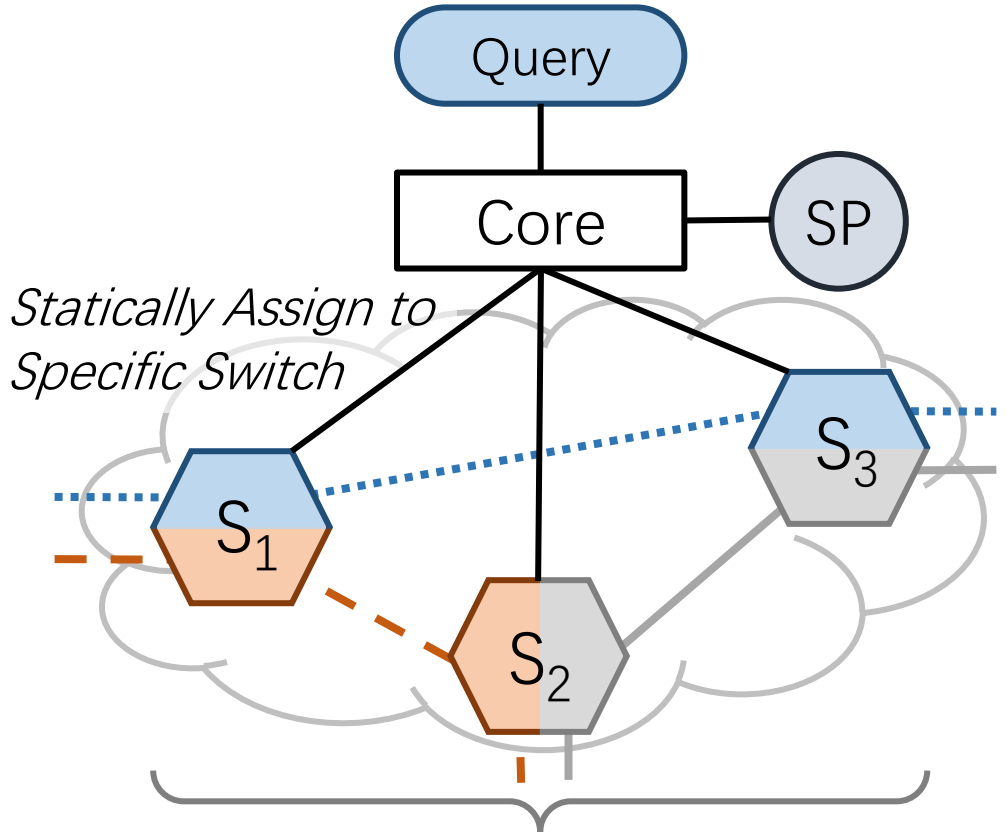
Different Switches Play Different Roles

Big Data Frameworks



vs.

Telemetry Systems



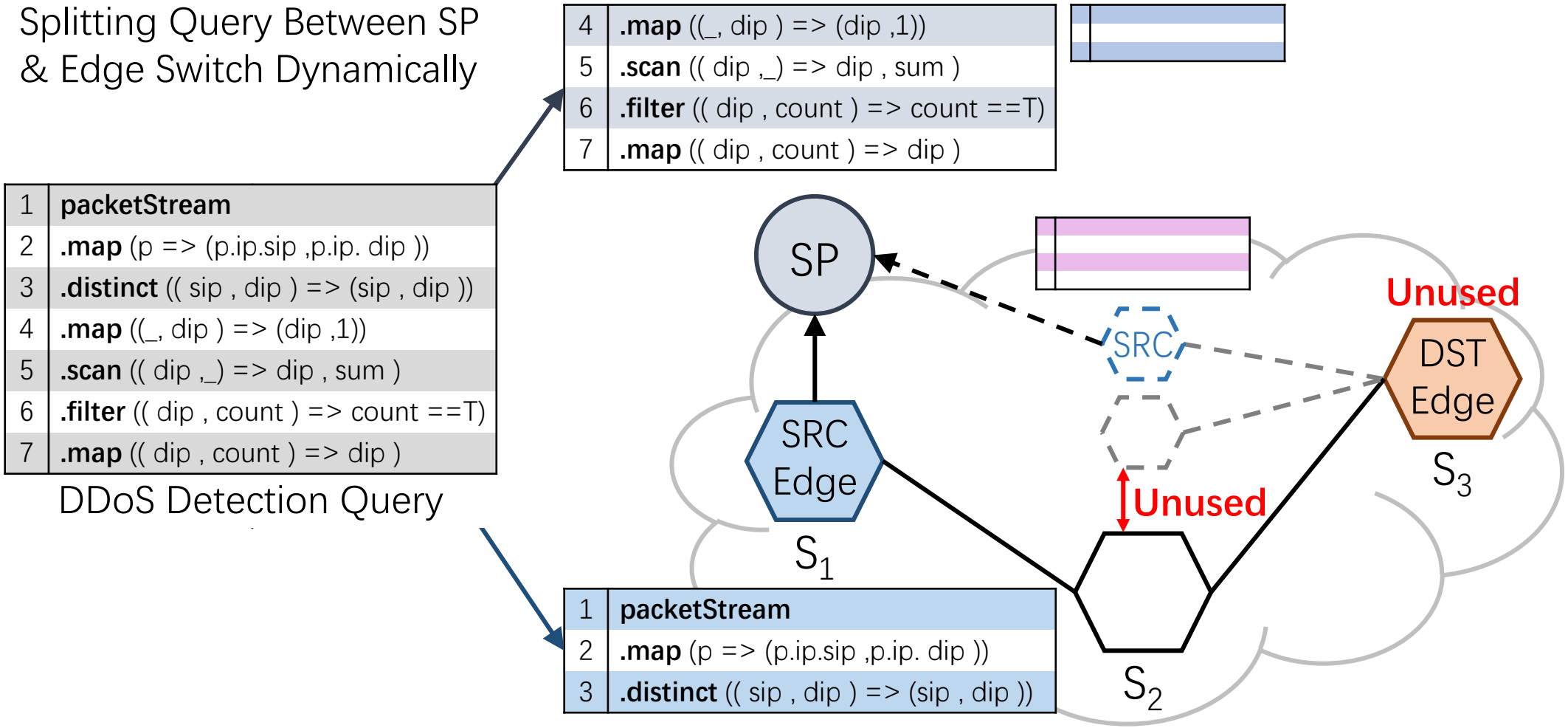
Using Switches Independently Is Insufficient

Splitting Query Between SP & Edge Switch Dynamically

1	**packetStream**
2	**.map** (p => (p.ip.sip ,p.ip. dip))
3	**.distinct** ((sip , dip) => (sip , dip))
4	**.map** ((_, dip) => (dip ,1))
5	**.scan** ((dip ,_) => dip , sum)
6	**.filter** ((dip , count) => count ==T)
7	**.map** ((dip , count) => dip)

DDoS Detection Query

4	**.map** ((_, dip) => (dip ,1))
5	**.scan** ((dip ,_) => dip , sum)
6	**.filter** ((dip , count) => count ==T)
7	**.map** ((dip , count) => dip)



1	**packetStream**
2	**.map** (p => (p.ip.sip ,p.ip. dip))
3	**.distinct** ((sip , dip) => (sip , dip))

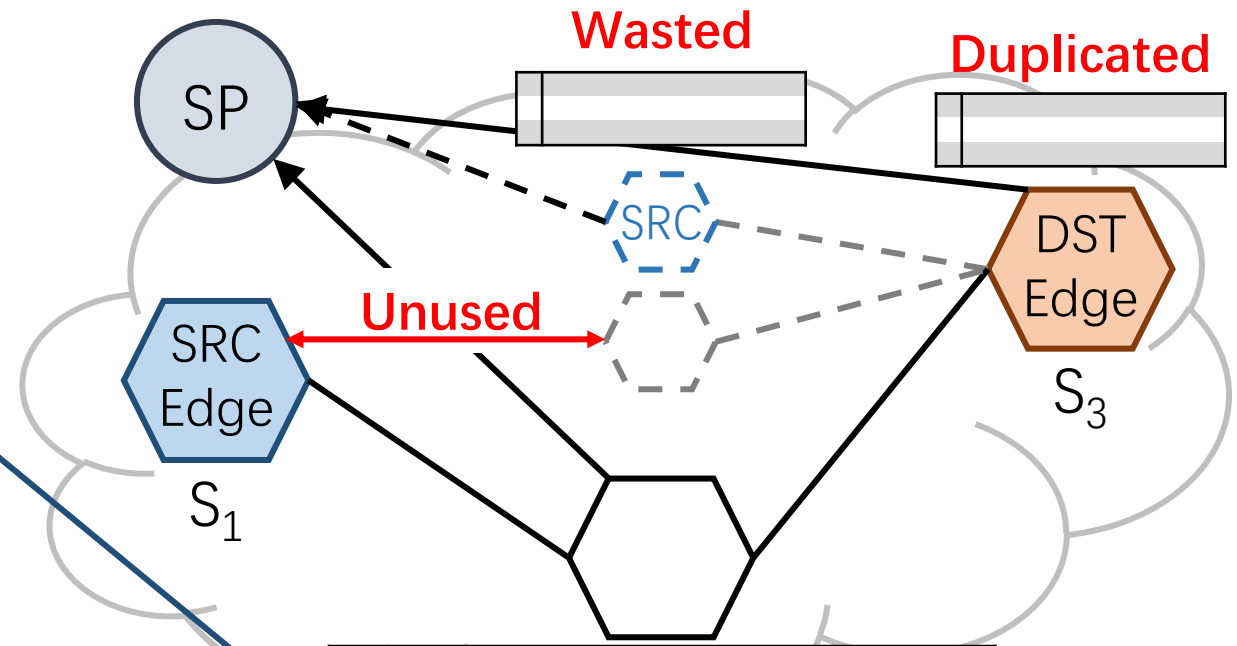
Using Switches Independently Is Insufficient (Cont.)

Static Splitting & Using Switches Independently

```
1  packetStream
2  .map (p => (p.ip.sip ,p.ip. dip ))
3  .distinct (( sip , dip ) => (sip , dip ))
4  .map ((_, dip ) => (dip ,1))
5  .scan (( dip ,_) => dip , sum )
6  .filter (( dip , count ) => count ==T)
7  .map (( dip , count ) => dip )
```

DDoS Detection Query

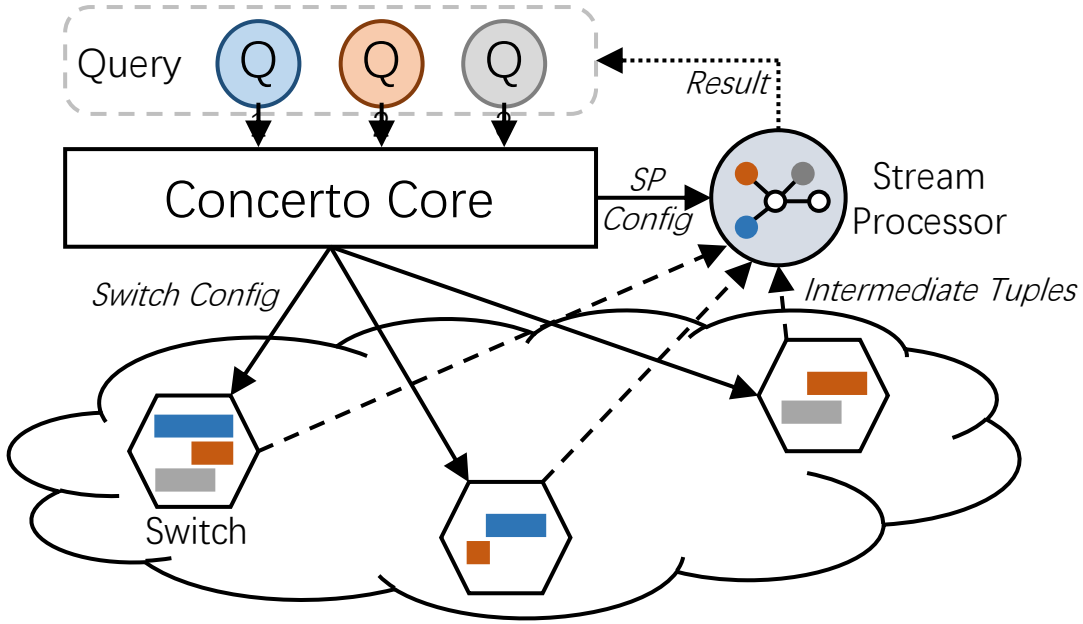
```
4  .map ((_, dip ) => (dip ,1))
5  .scan (( dip ,_) => dip , sum )
6  .filter (( dip , count ) => count ==T)
7  .map (( dip , count ) => dip )
```



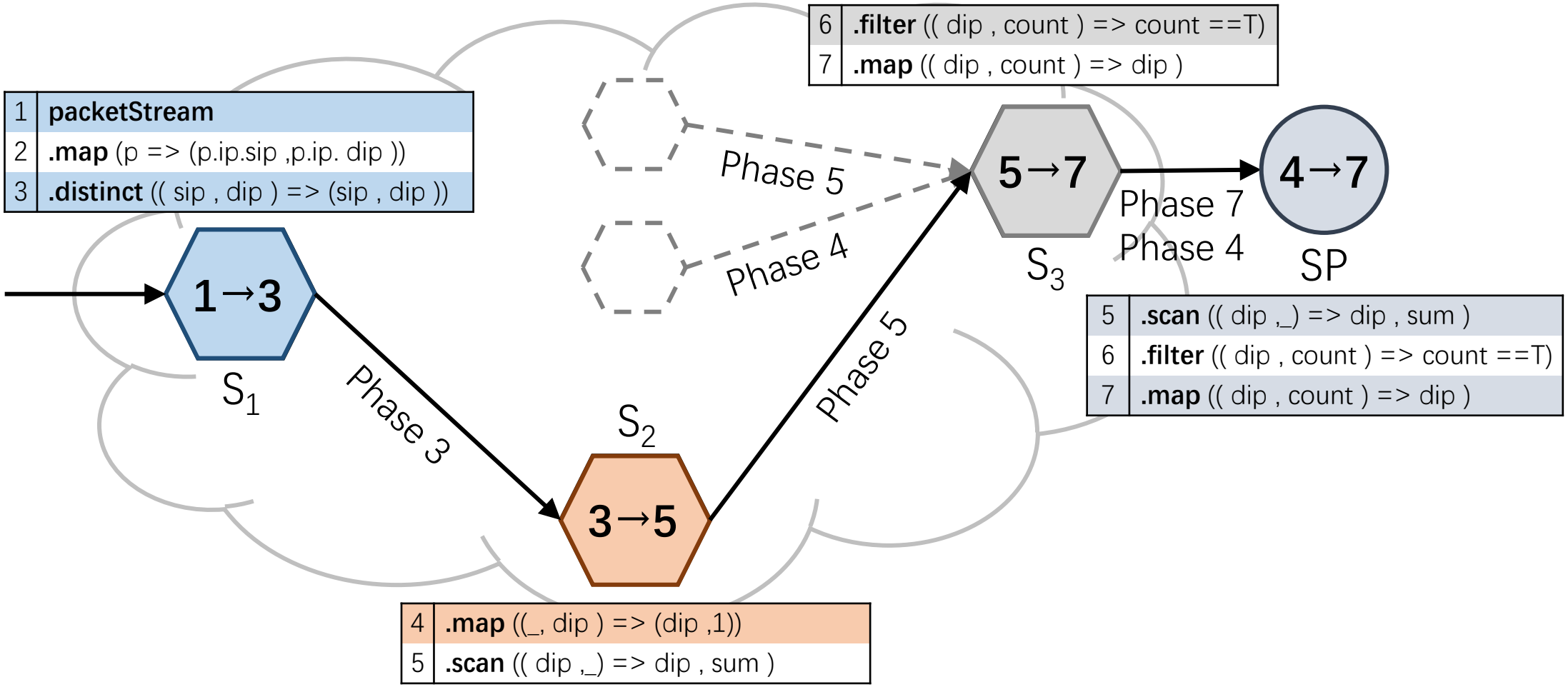

```
1  packetStream
2  .map (p => (p.ip.sip ,p.ip. dip ))
3  .distinct (( sip , dip ) => (sip , dip ))
```

Concerto: Cooperative Network-Wide Telemetry

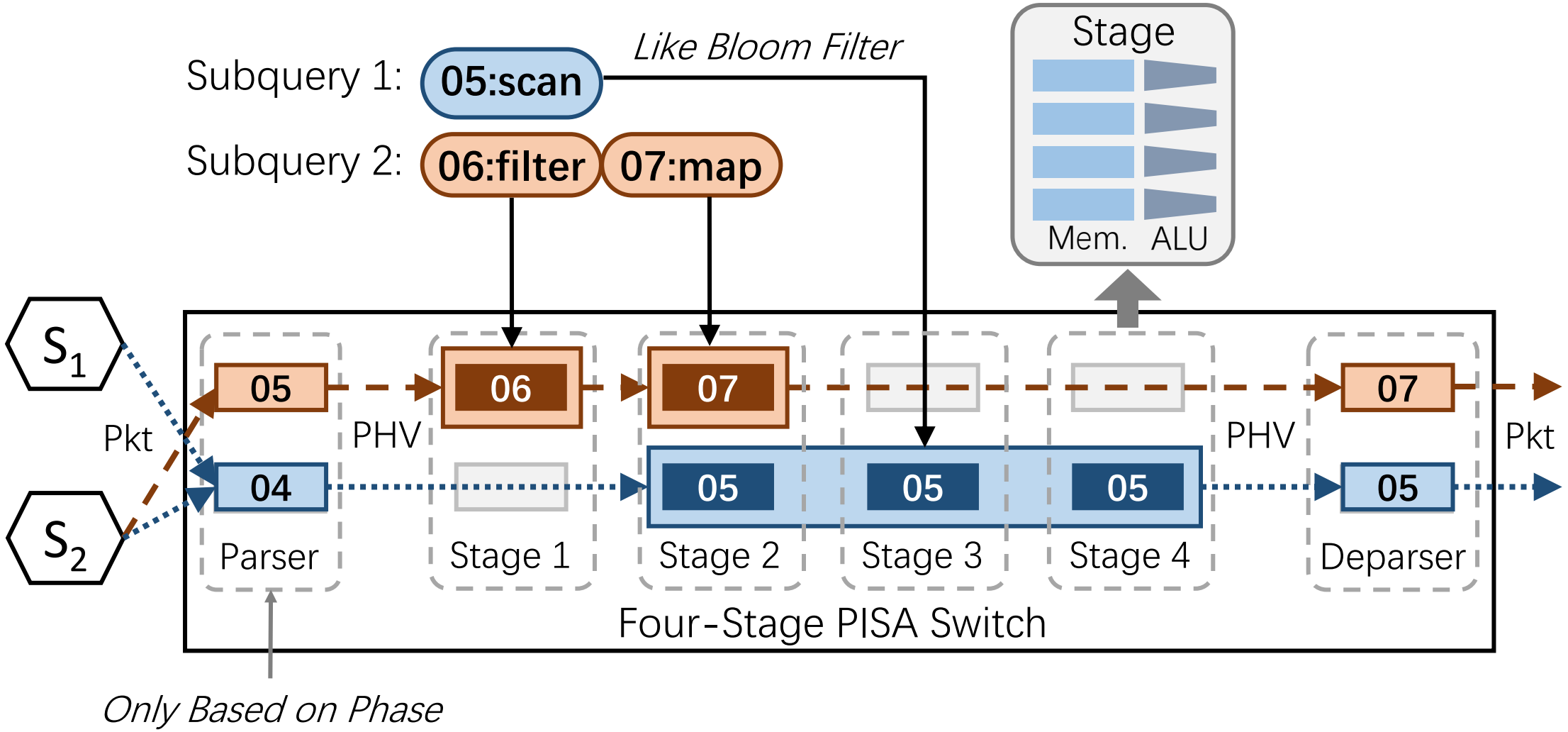
- Challenge
  - Splitting queries among switches while meeting resource & network constraints
- Cooperative query execution model
  - Splitting query to multiple PISA switches
  - Each switch processes tuples locally
  - Various operations on different switches
  - Best-effort tuple processing
- Automatic query placement
  - Analyzing query restrictions from AST
  - Formulating query placement as MIP
- Result
  - Reduce the stream processor's workload by up to $19\times$
  - Achieve $10^4\times$ lower error rate with the same workload



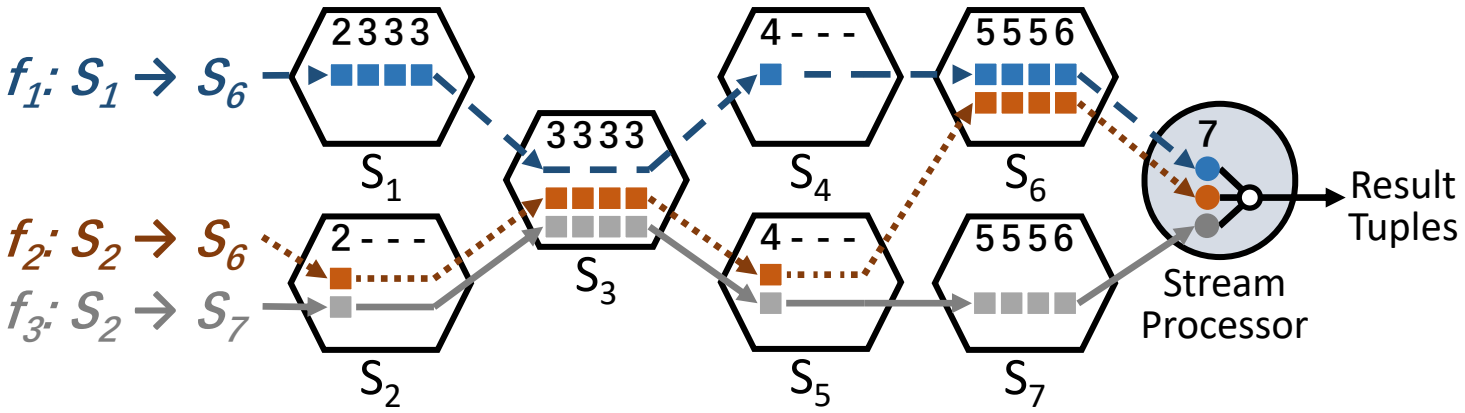
Cooperative Query Execution Model



Query Execution on Switches



Concerto Puts More Operations on Switches




```
1  packetStream
2  .map (p => (p.ip.sip ,p.ip. dip ))
3  .distinct (( sip , dip ) => (sip , dip ))
4  .map ((_, dip ) => (dip ,1))
5  .scan (( dip ,_) => dip , sum )
6  .filter (( dip , count ) => count ==T)
7  .map (( dip , count ) => dip )
```

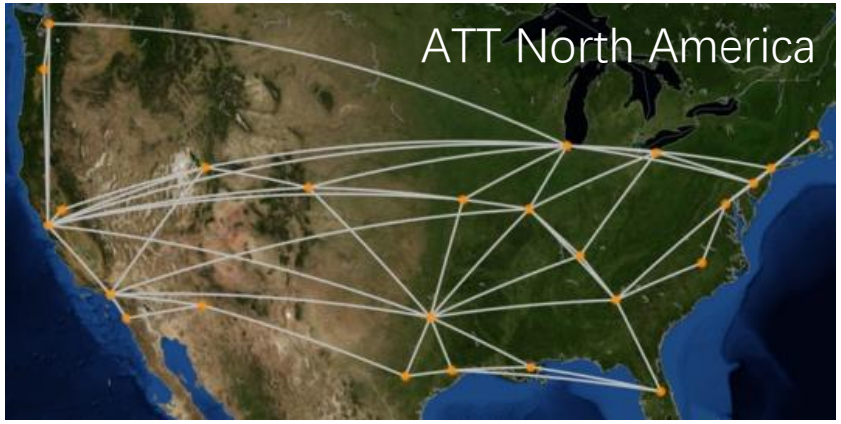
- Switch hardware
  - 4 stages
  - 0.5 Mb of registers at each stage
- Results
  - Stateless filtering: 2.1×10^6
  - Independent stateful: 1.4×10^6
  - Concerto: 86

Flow	# Tuples				# Stages	
	t1, t2	*t3, t4*	*t5*	*t6, t7*	*d3*	*d5*
f1	442628	50034	1033	25	3	3
f2	1383594	113584	1739	36	4	3
f3	307941	8874	2194	25	3	3
f1+f2	1826222	163618	2772	61	5	3
f2+f3	1691535	122458	3933	61	4	4
f1+f2+f3	2134163	172492	4966	86	5	4

Evaluation Setup

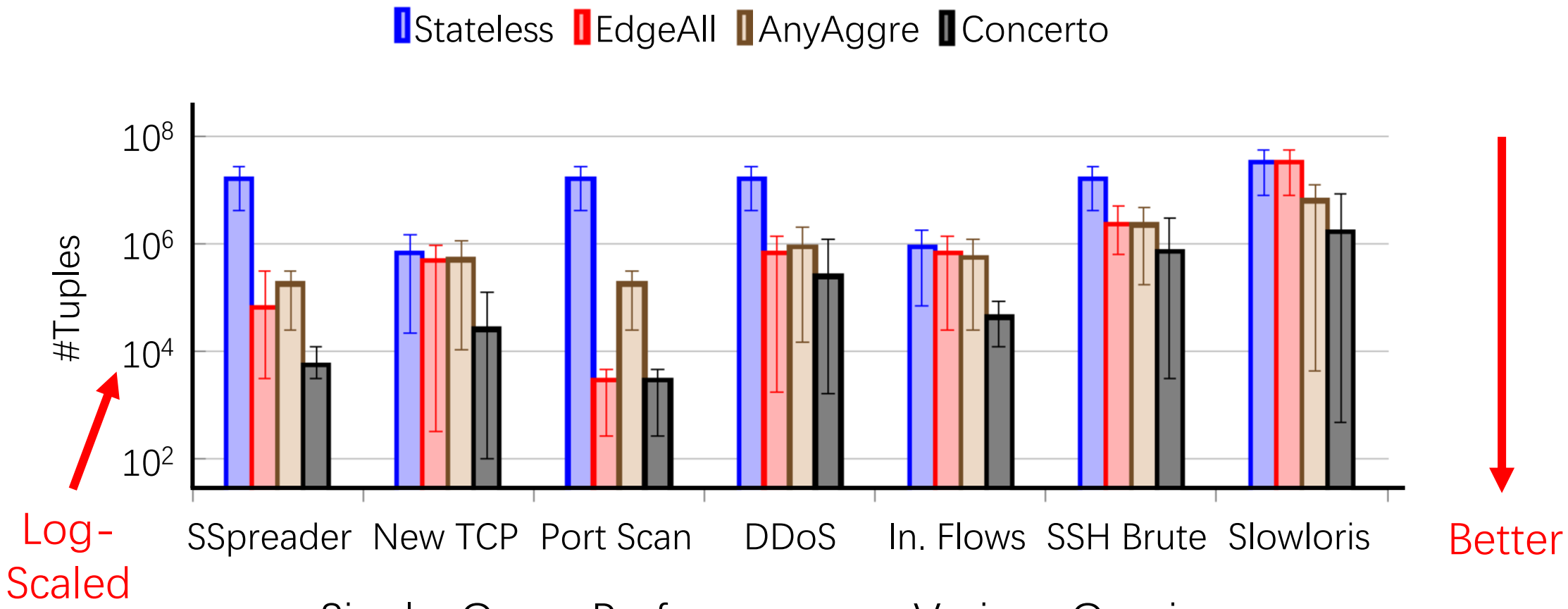
- Questions: workload reduction, error rate guarantee, scalability
- Topology

Topology	# Sites	# Links
Claranet	15	18
ATT North America	25	56
Cesnet-10	52	63
OTEGlobe	93	103

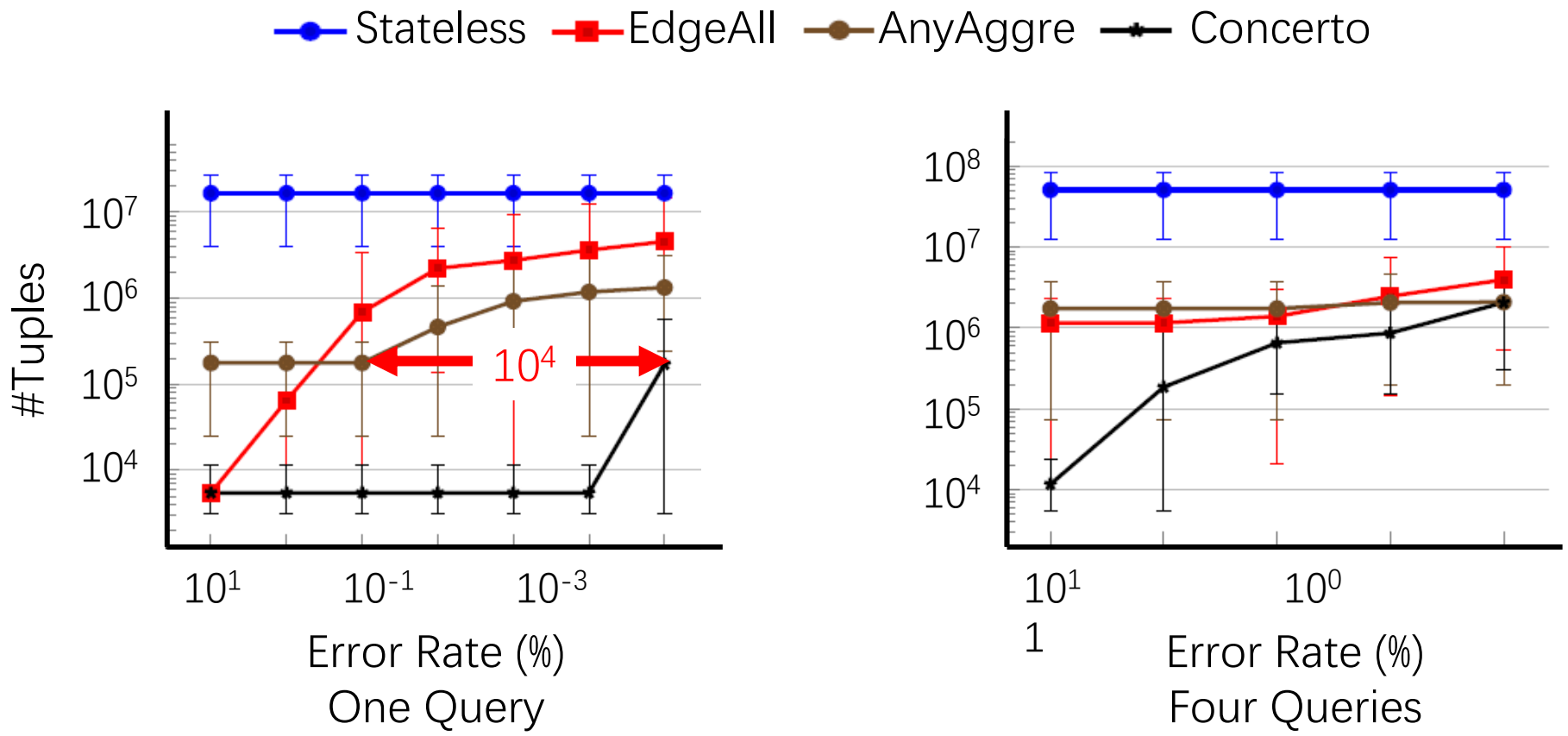


- CAIDA trace: captured at a backbone ISP link from New York to San Paulo
- Compared systems
  - **Stateless**: Everflow, DREAM
  - **EdgeAll**: Sonata
  - **AnyAggre**: OpenSketch, UnivMon, Marple
- Metric: # tuples to the stream processor (same as Sonata)

Concerto Reduces SP's Workload on Various Queries

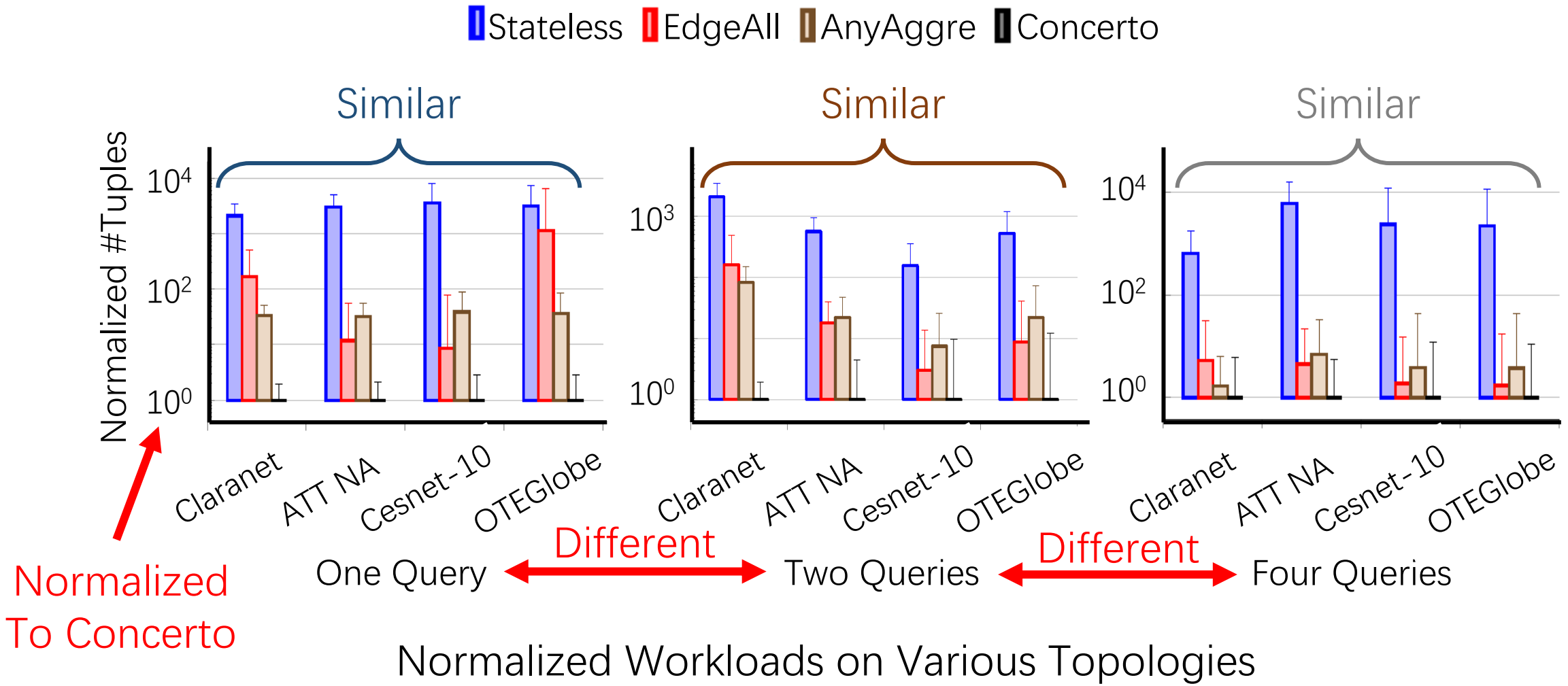


Concerto Achieves Much Lower Error Rate



Error Rate Requirement

Concerto Scales Well



Normalized Workloads on Various Topologies

Conclusion

- We propose a cooperative query execution model
  - Mimics network routing, each switch processes tuples locally
  - Independent of the underlying routing method
  - Applies to arbitrary topology
- We provide a method to automatically compile queries to PISA switches
  - Analyzes the query placement requirement from AST
  - Formulates and optimizes query placement on switches using MIP
- We show that the cooperative query execution of Concerto is effective
  - Reduces the stream processor's workload by as much as 19 times
  - Achieves an error rate of 10^4 times lower than state-of-the-art systems

Thanks!
Q&A