

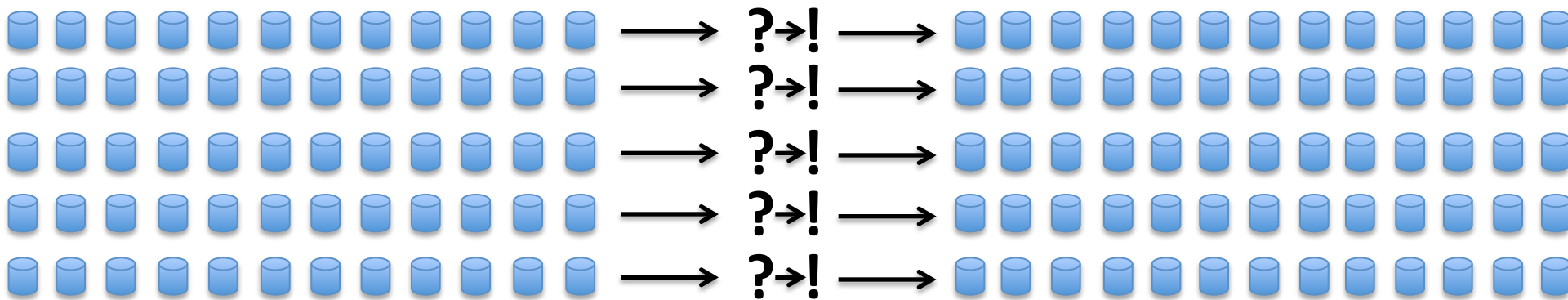
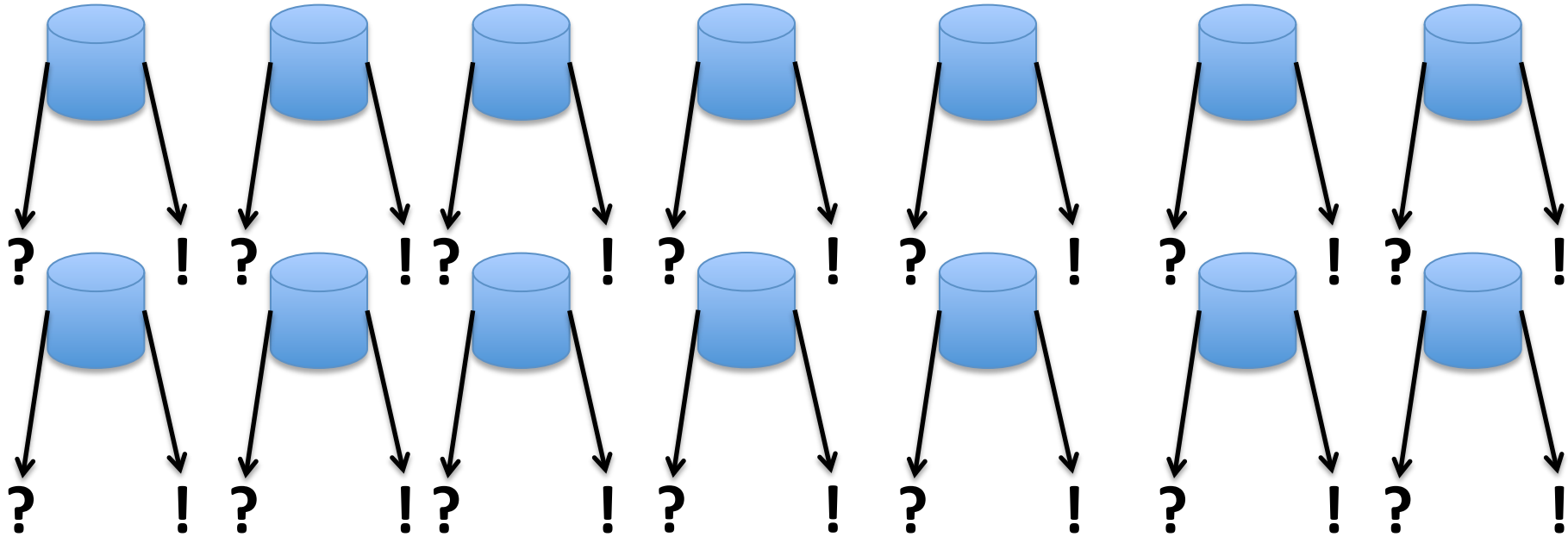
Auto-Parallelizing Stateful Distributed Streaming Applications

Scott Schneider^{*}, Martin Hirzel^{*},
Bugra Gedik⁺ and Kun-Lung Wu^{*}

^{}IBM Research*

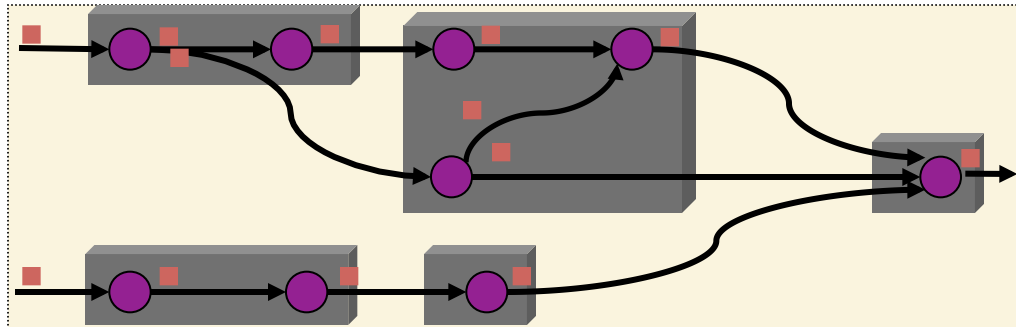
⁺Bilkent University

Big Data



Programming Model

- Streams applications
 - Described as data-flow graphs
 - An instance of a flow graph is a job in the system
 - Flow graphs consist of
 - **Tuples**: structured data item
 - **Operators**: Reusable stream analytics
 - **Streams**: Series of tuples with a given schema



Streaming Programming Models

Synchronous

- Static selectivity
 - e.g., $1 : 3$

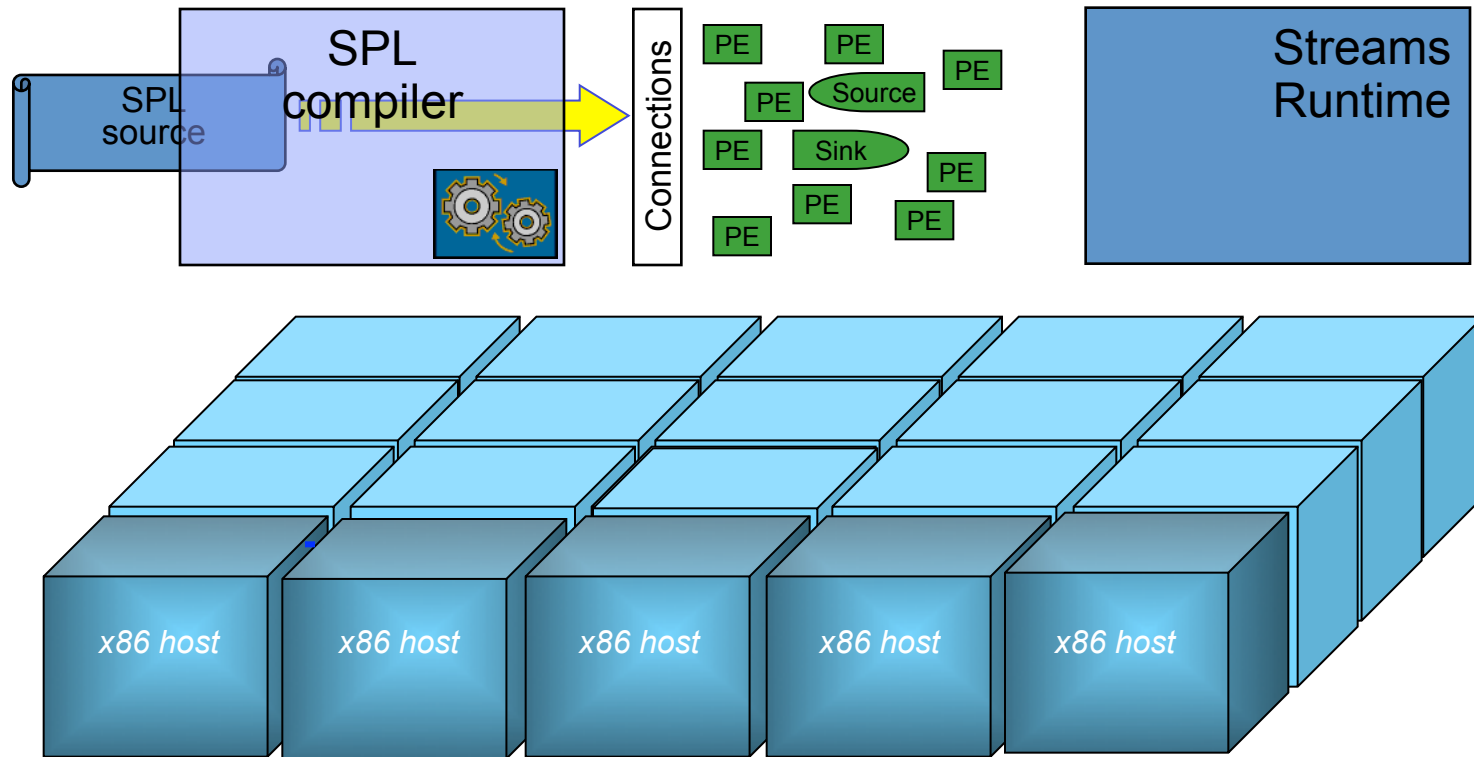
```
for i in range(3):  
    result = f(i)  
    submit(result)
```
 - In general, $m : n$ where m and n are statically known
- Always has static schedule

Asynchronous

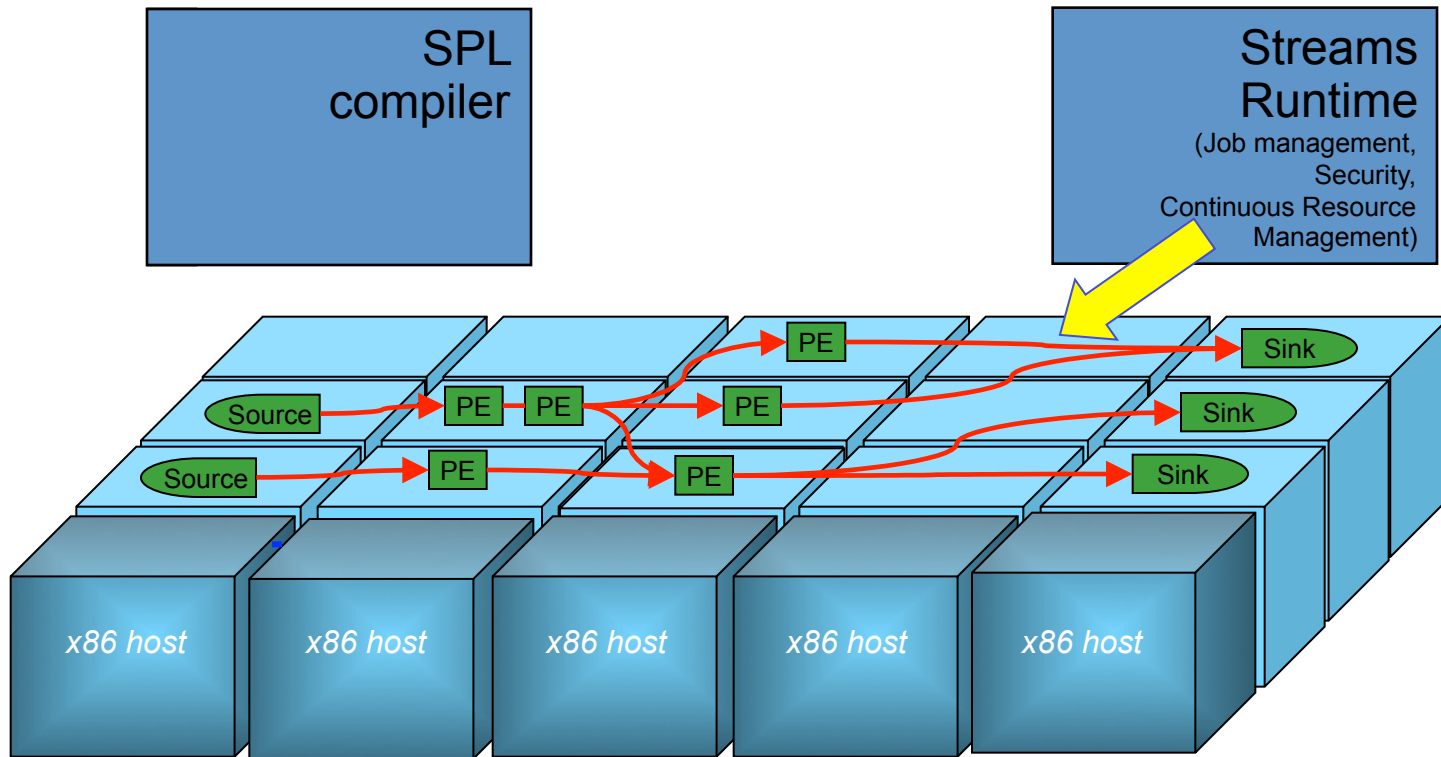
- Dynamic selectivity
 - e.g., $1 : [0,1]$

```
if input.value > 5:  
    submit(result)
```
 - In general, $1 : *$
- In general, schedules cannot be static

InfoSphere Streams Runtime

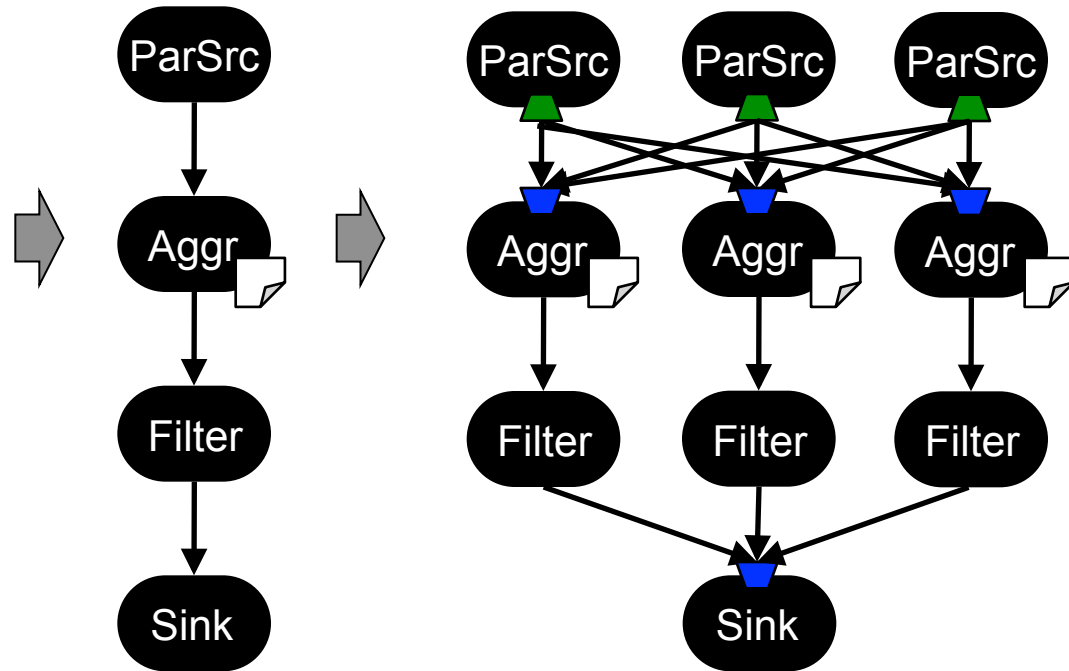


InfoSphere Streams Runtime



What we do

```
composite Main {  
  type  
    Entry = int32 uid, rstring server,  
            rstring msg;  
    Sum = uint32 uid, int32 total;  
  graph  
    stream<Entry> Msgs = ParSource() {  
      param servers: "logs.*.com";  
      partitionBy: server;  
    }  
  
    stream<Sum> Sums = Aggregate(Msgs) {  
      window Msgs: tumbling, time(5),  
              partitioned;  
      param partitionBy: uid;  
    }  
  
    stream<Sum> Suspects = Filter(Sums) {  
      param filter: total > 100;  
    }  
  
    () as Sink = FileSink(Suspects) {  
      param file: "suspects.csv";  
    }  
}
```



Overview

Compiler:

- Apply parallel transformations
- Pick routing mechanism (e.g., hash by key)
- Pick ordering mechanism (e.g., seq. numbers)

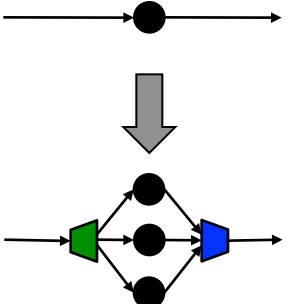
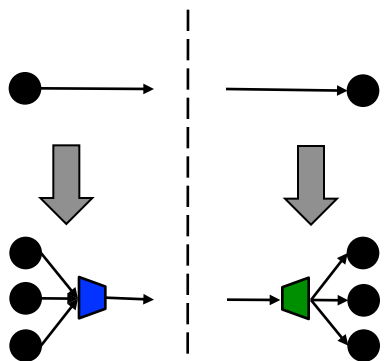
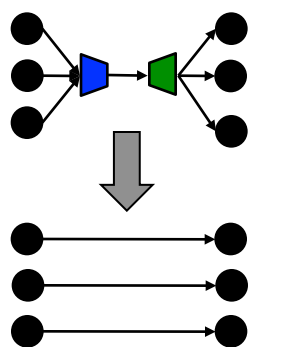
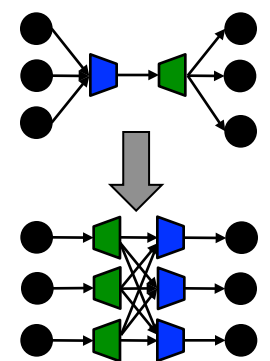


Stream graph description

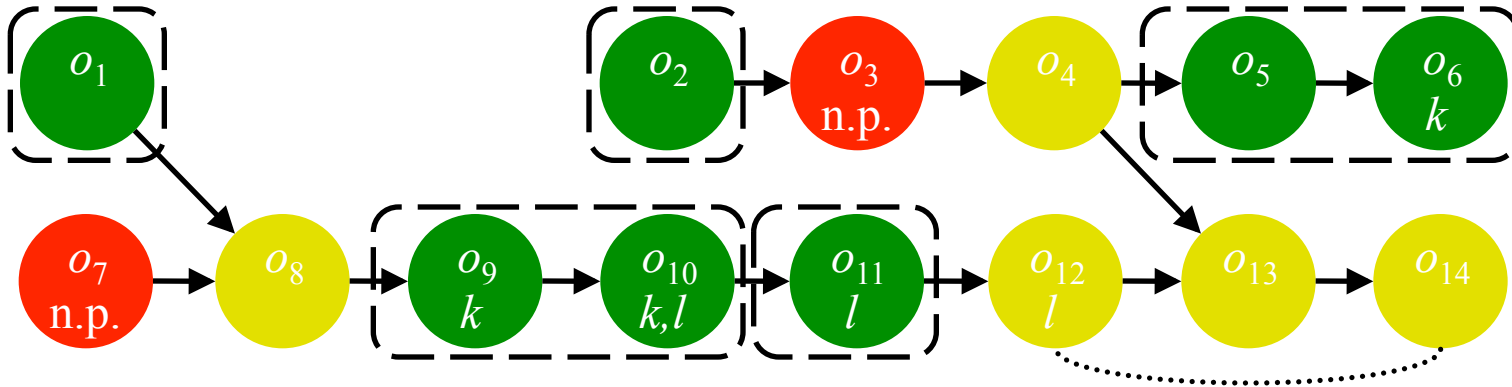
Runtime:

- Replicate segment into channels
- Add split/merge/shuffle as needed
- Enforce ordering

Transformations & Safety Conditions

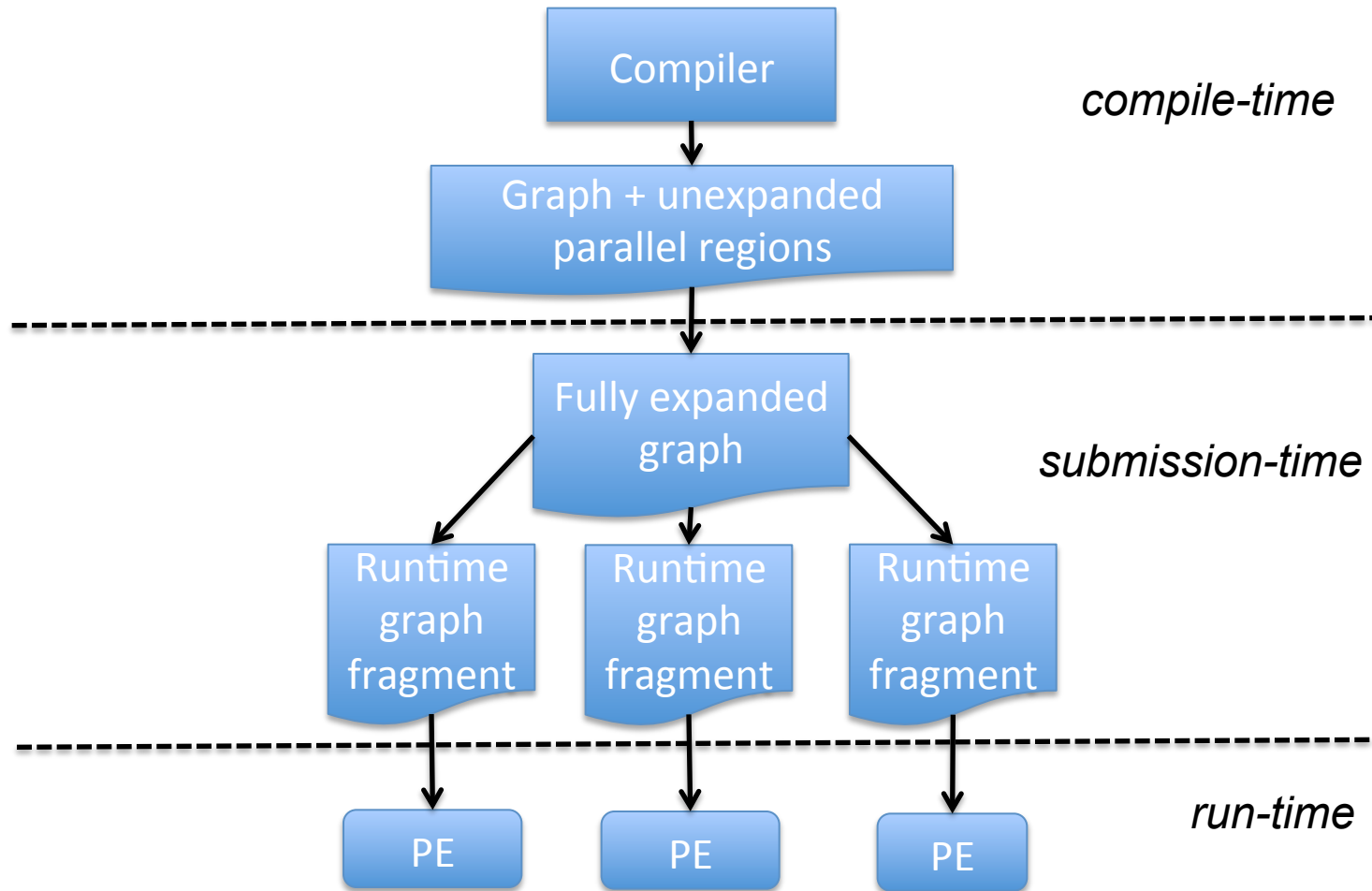
Parallelize non-source/sink	Parallelize sources and sinks	Combine parallel regions	Rotate merge and split
			
<ul style="list-style-type: none"> • stateless <i>or</i> partitioned state • selectivity ≤ 1 • simple chain 	<ul style="list-style-type: none"> • stateless <i>or</i> partitioned state 	<ul style="list-style-type: none"> • stateless <i>or</i> compatible keys • forwarding 	<ul style="list-style-type: none"> • incompatible keys

Select Parallel Segments



- Can't parallelize
 - Operators with >1 fan-in or fan-out
 - Punctuation dependency later on
- Can't add operator to parallel segment if
 - Another operator in segment has co-location constraint
 - Keys don't match

Compiler *to* Runtime



Runtime

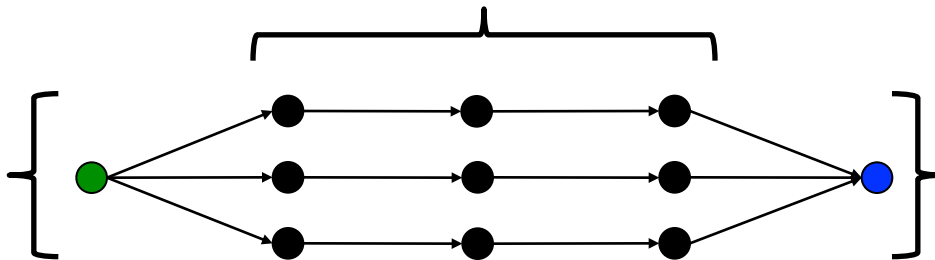
	selectivity = 1	selectivity ≤ 1	selectivity unknown
no state			<i>don't parallelize</i>
partitioned state			<i>don't parallelize</i>
unknown state	<i>don't parallelize</i>	<i>don't parallelize</i>	<i>don't parallelize</i>

Operators in parallel segments:

- Forward seqno & pulse

Split:

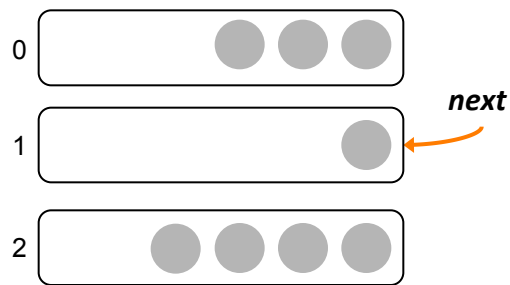
- Insert seqno & pulse
- Routing



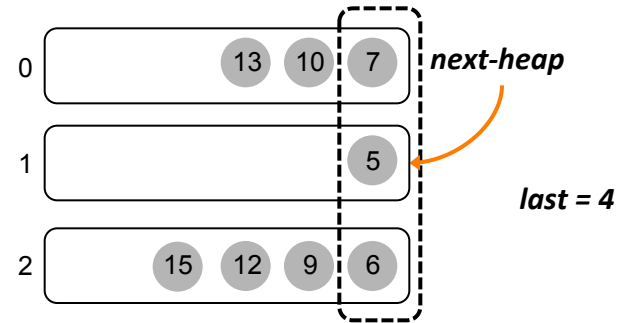
Merge:

- Apply ordering policy
- Remove seqno (if there) and drop pulse (if there)

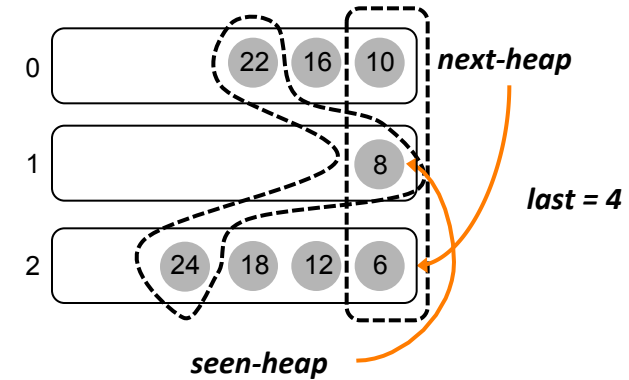
Merger Ordering



Round-Robin

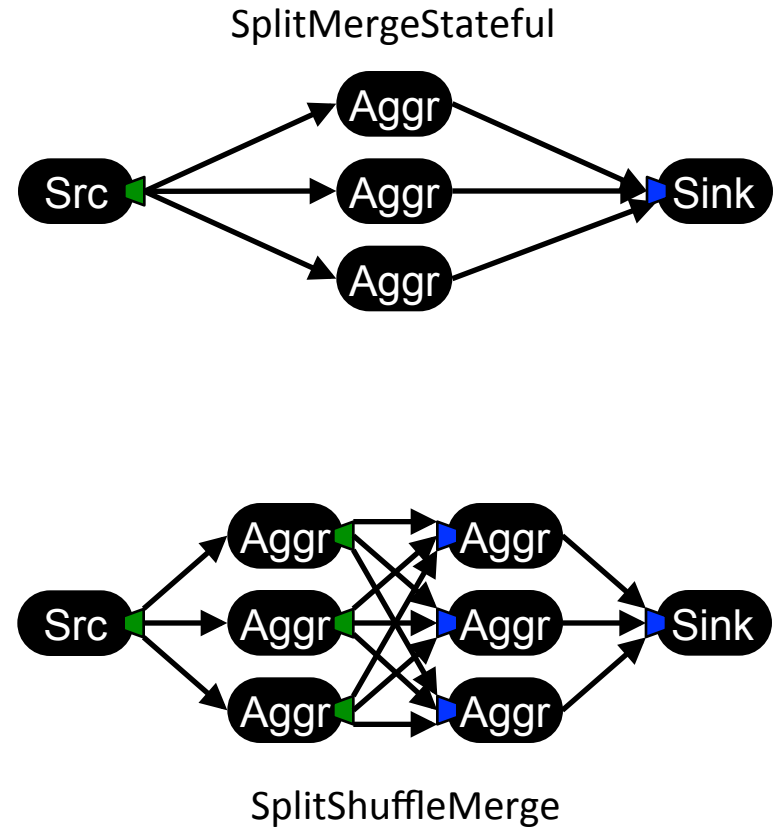
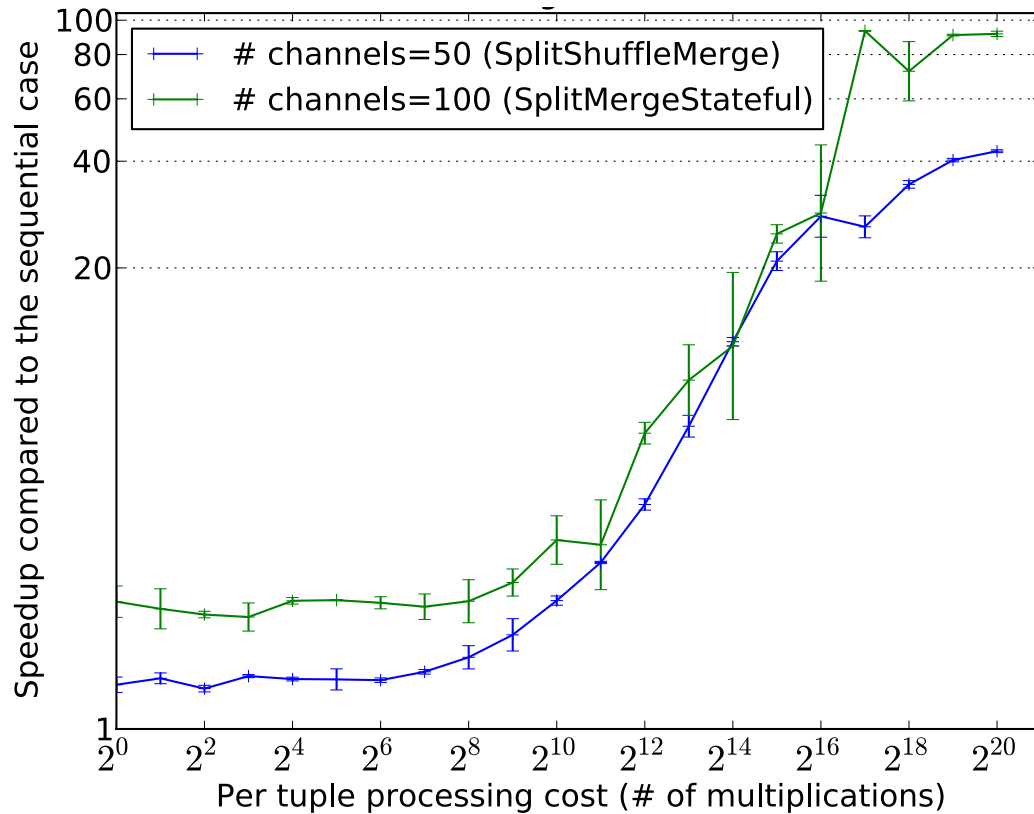


Sequence Numbers

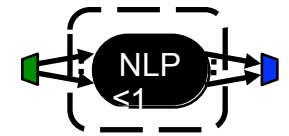
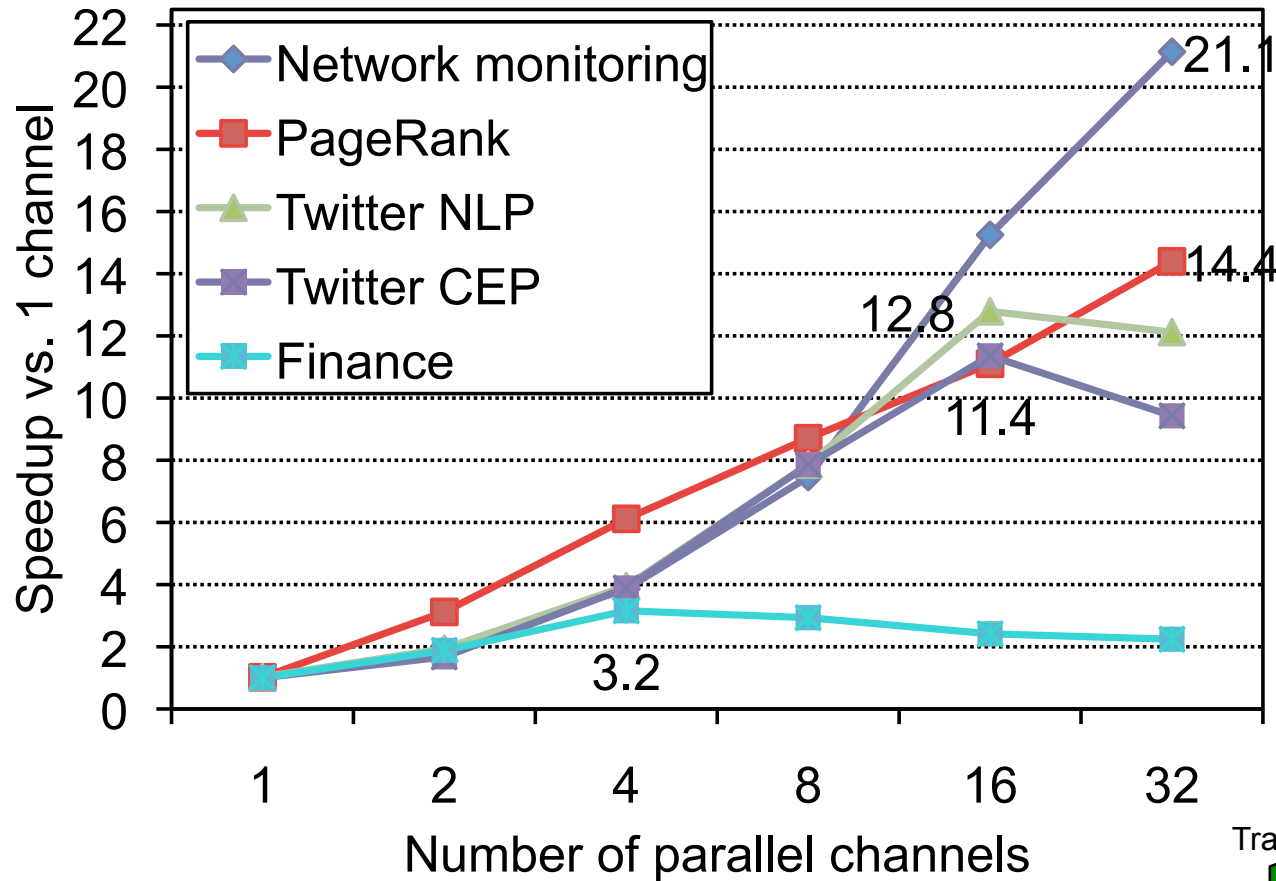


Sequence Numbers and Pulses

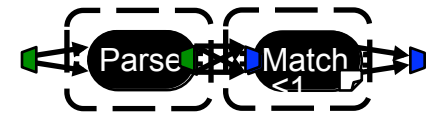
Scalability



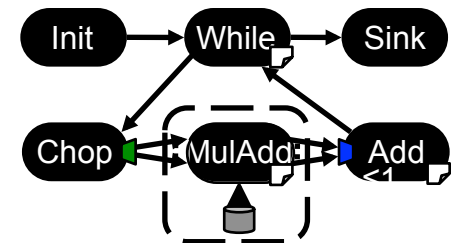
Application Kernels



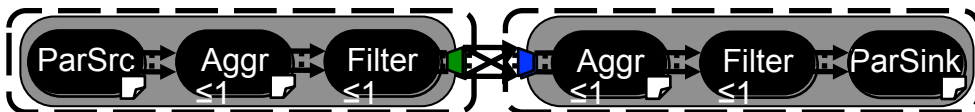
Twitter NLP



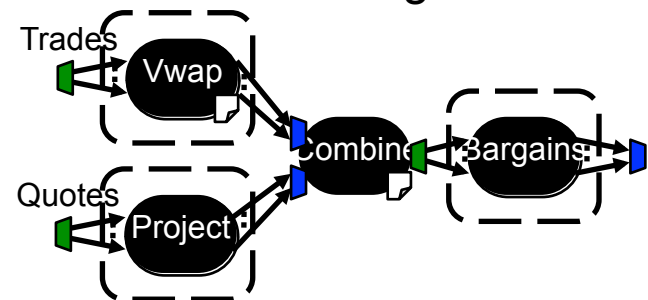
Twitter CEP



PageRank



Network monitoring

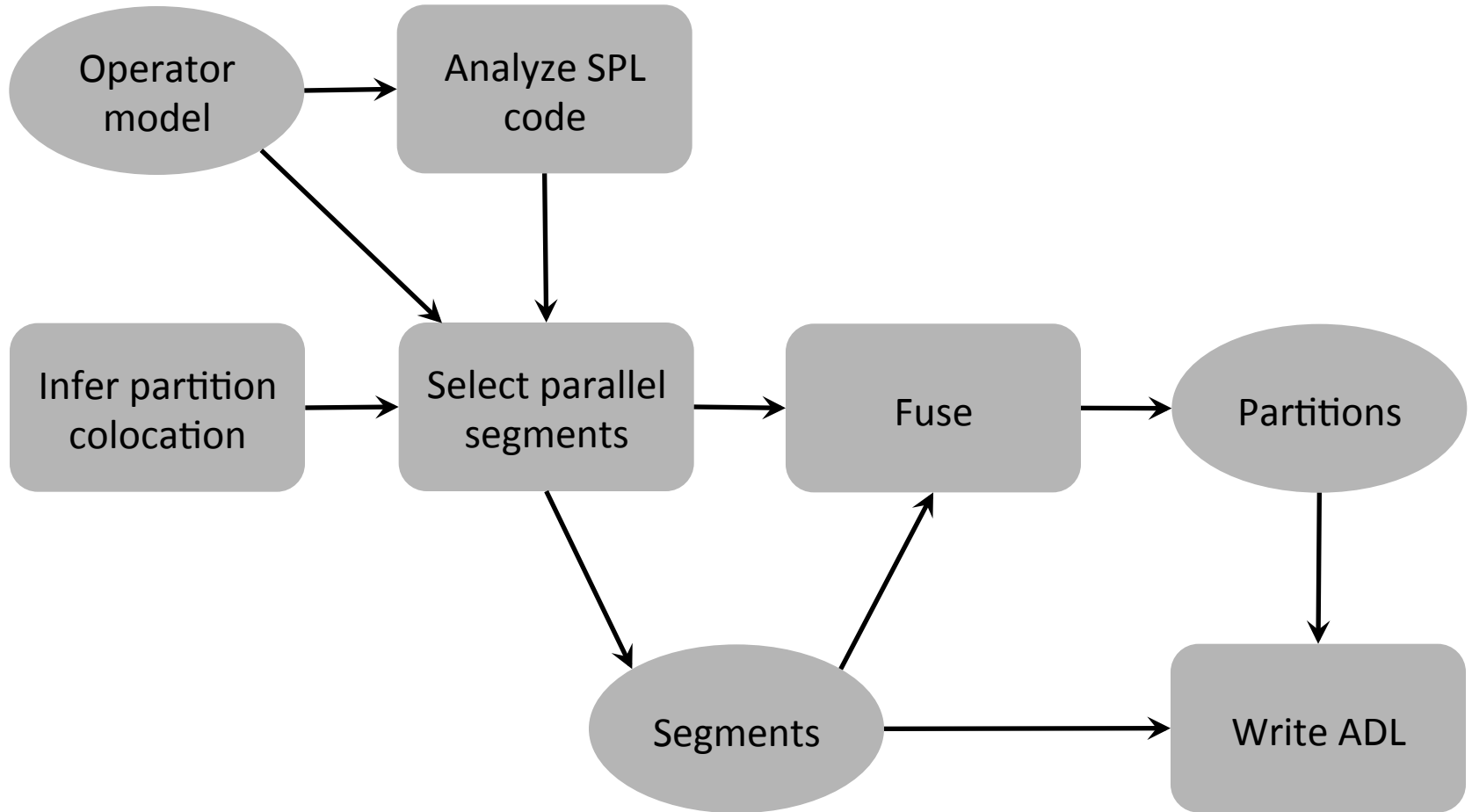


Finance

Questions?

Backups

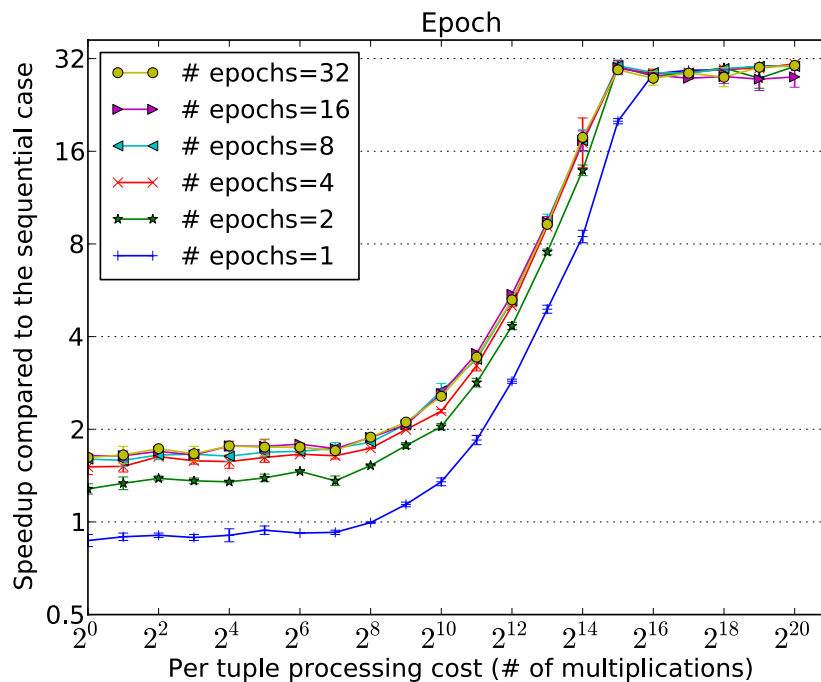
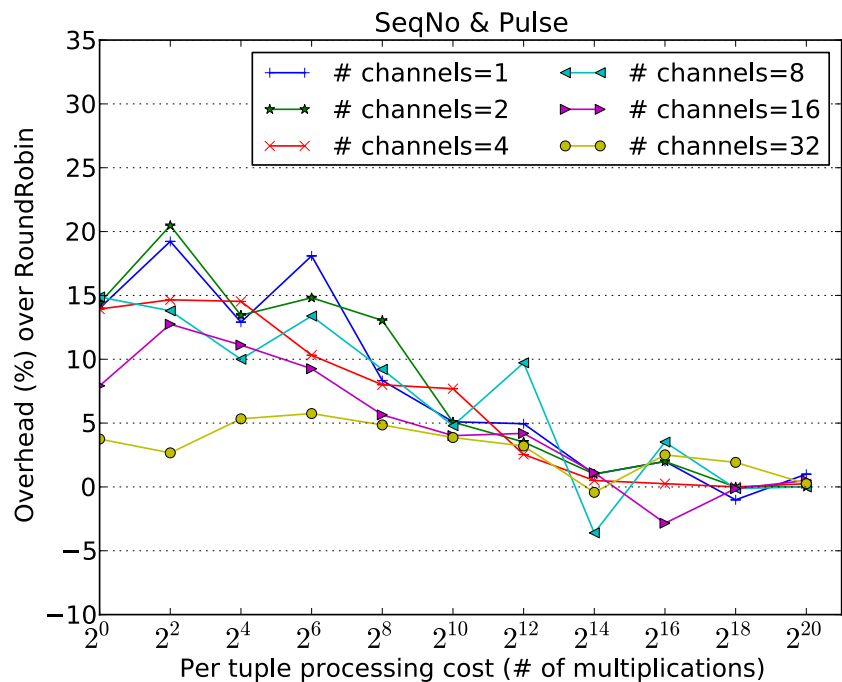
Compiler Changes



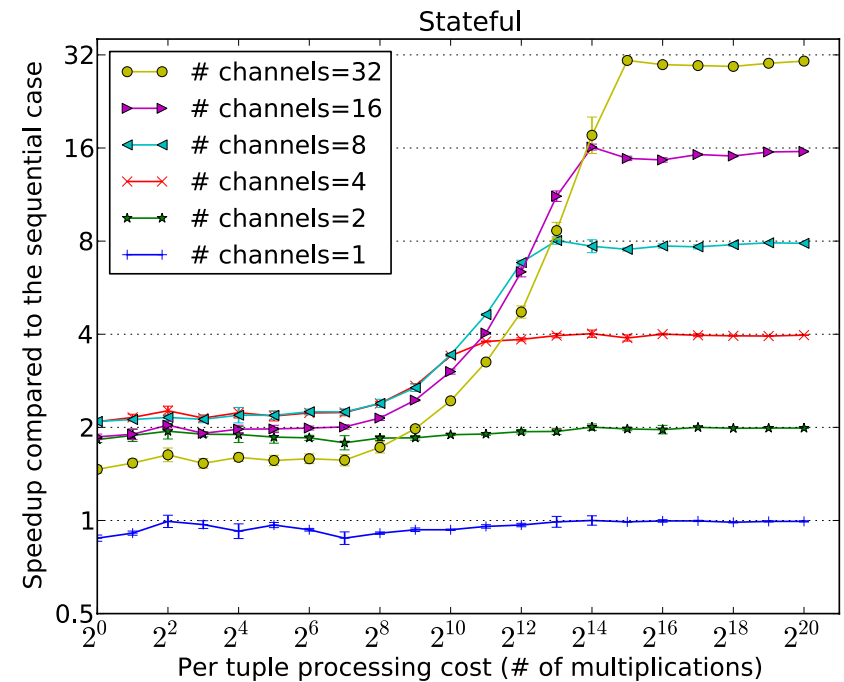
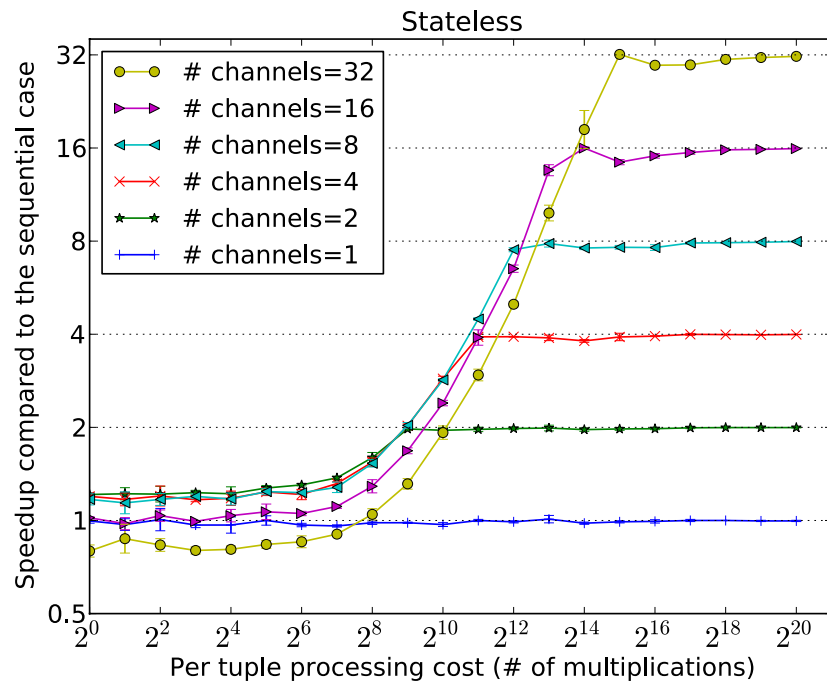
Transport Changes

- TCPSender
 - Added the ability to send to a subset of connections on an output port
- Handshake
 - Modified to include sender identities
- TCPReciever
 - Added support for identifying which connection has delivered a tuple

Overhead



Scalability



Scalability

