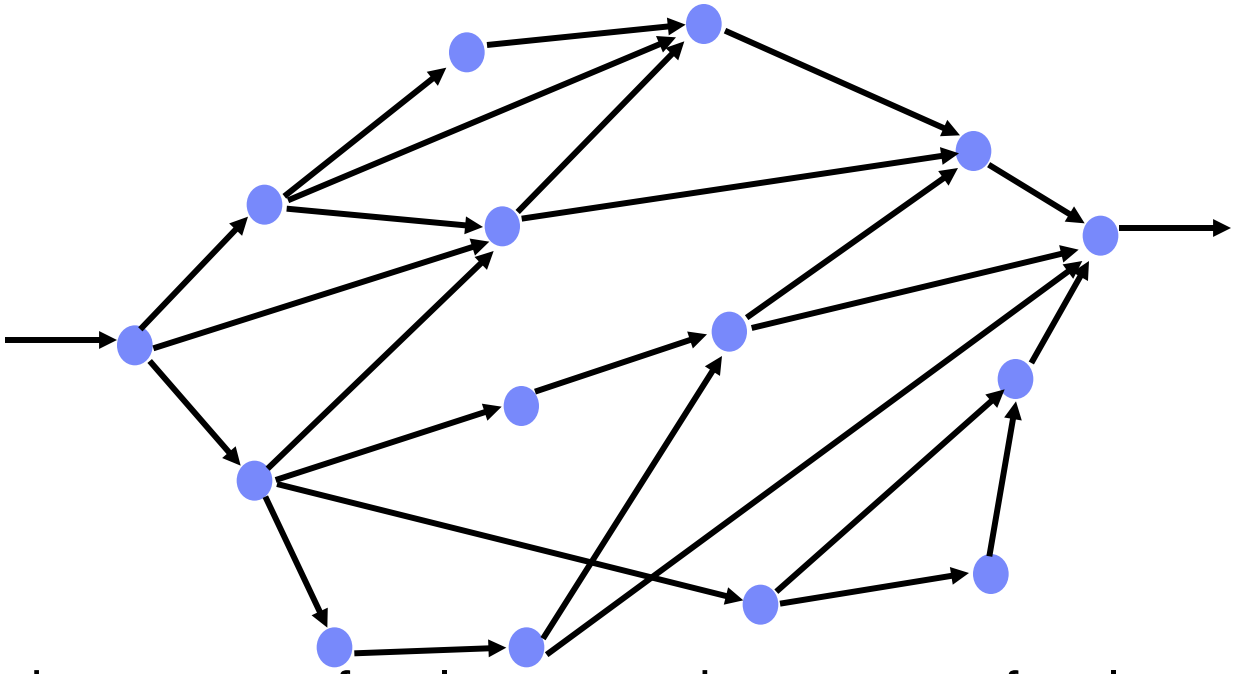
IBM Research

COLA: Optimizing Stream Processing Applications Via Graph Partitioning

Rohit Khandekar, Kirsten Hildrum, Sujay Parekh, Deepak Rajan, Joel Wolf, Kun-Lung Wu, Henrique Andrade, and Bugra Gedik

Streaming application in SPADE

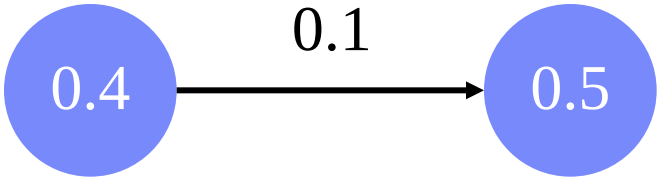


- Input is a stream of tuples, output is a stream of tuples
- May be distributed over many hosts
- Operators perform transformations on the data
- Many operators are quite simple, eg:
  - Drop tuple if $temp_c < 37$
  - $temp_c \rightarrow temp_f = 9/5\ temp_c + 32$

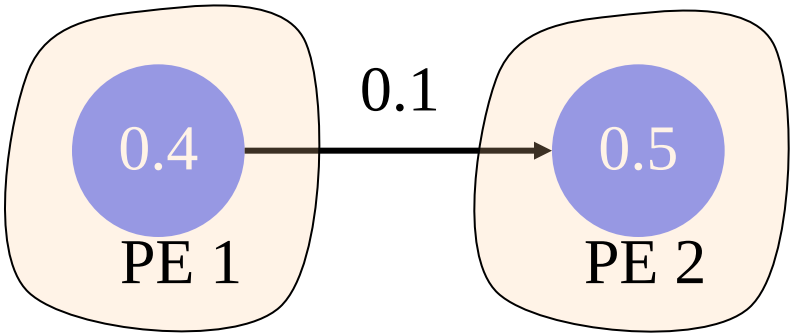
Simple operators mean high communication cost

- **Problem: The cost of unpackaging and packaging a tuple can be high relative to the cost of a simple operation**
  - Receiving $temp_c$ and sending $temp_f$ may be greater than computing $temp_f$ from $temp_c$.
  - This is obviously wasteful, and paid many times over can limit the data rate the application can process.
- **Solution: SPADE provides a way to fuse operators into PEs.**
  - The code for them is compiled together, and the operators form a single process.
  - Passing a tuple within a PE is a function call
- **Caveat: Fusing too many operators together means that the resulting PEs runs out of CPU cycles on the node.**

Fusion example

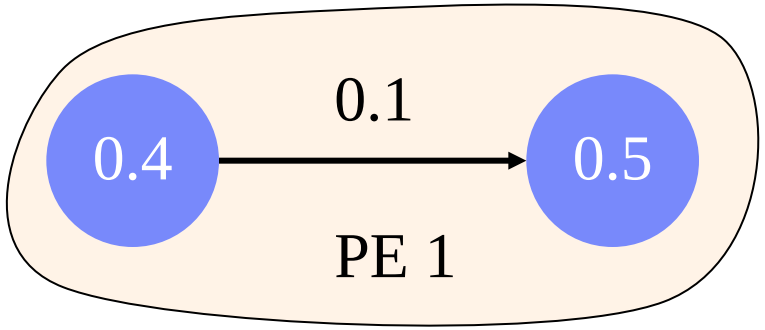


Unfused



PE1 has size 0.5
PE2 has size 0.6
Total size: 1.1

Fused



Single PE has size 0.9
Total size: 0.9

Fusing operators: Competing Goals

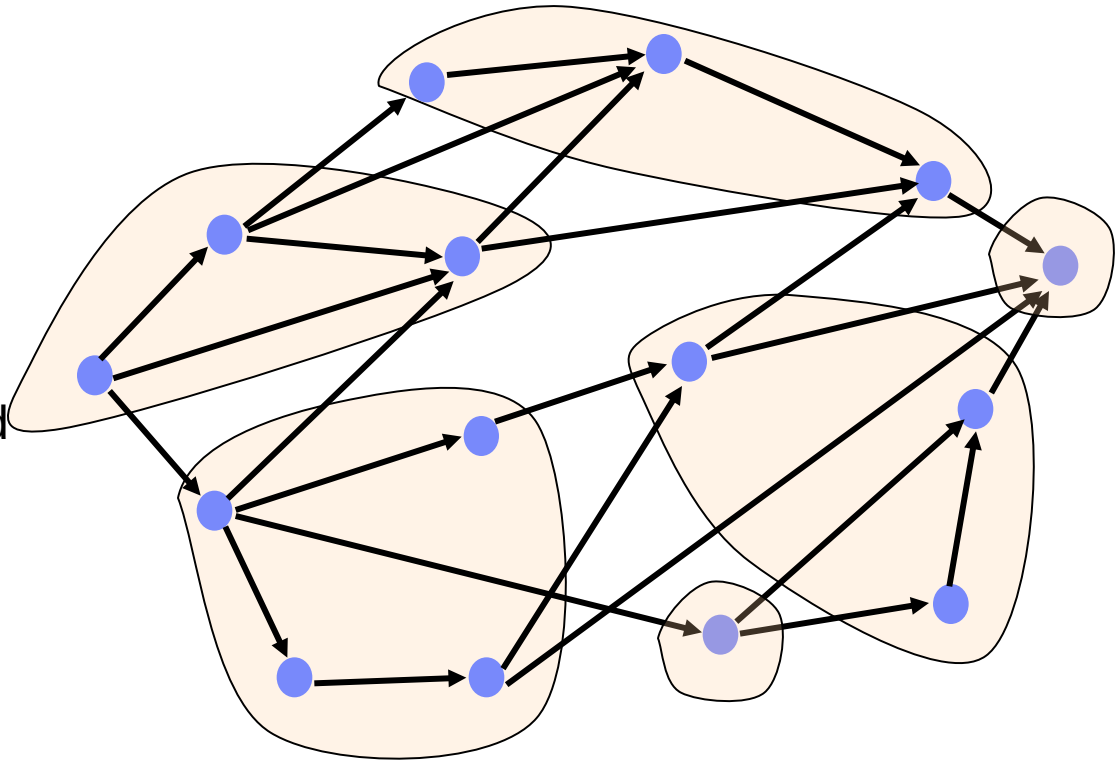
- **Lower communication cost**
  - Best: Fuse all operators together.
  - Worst: One operator per PE.
- **Ensure that no piece is too large (and provide scheduler flexibility)**
  - Best: One operator per PE.
  - Worst: Fuse all operators together.
- **Our algorithm: Fuses operators to lower communication cost while keeping the pieces small enough to allow for good scheduling.**

Outline

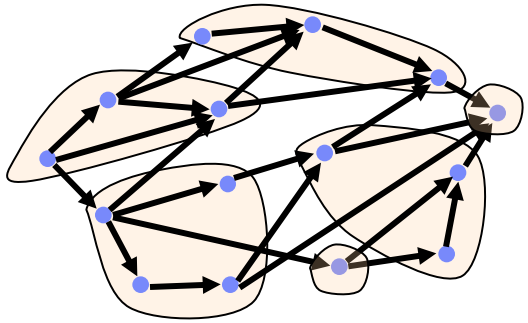
- Our algorithmic approach
  - **Graph Partitioner**
  - **Oracle**
- Additional fusion constraints handled
- Experimental results
- NOTE: Talk presents simplified algorithm, please see paper for more detailed scheme

The SPADE fusion problem, mathematically

- **Input:**
  - Directed graph
  - Vertices: operators
    - Weights = CPU fractions
  - Arcs: streams
    - Weights = CPU fraction needed for packaging and unpackaging tuples (communication cost)
- **Output:** A fusion of operators into clusters (PEs).

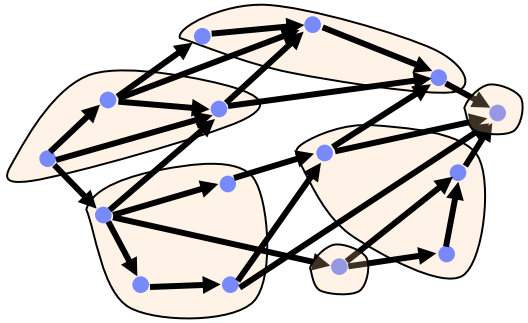


The SPADE fusion problem



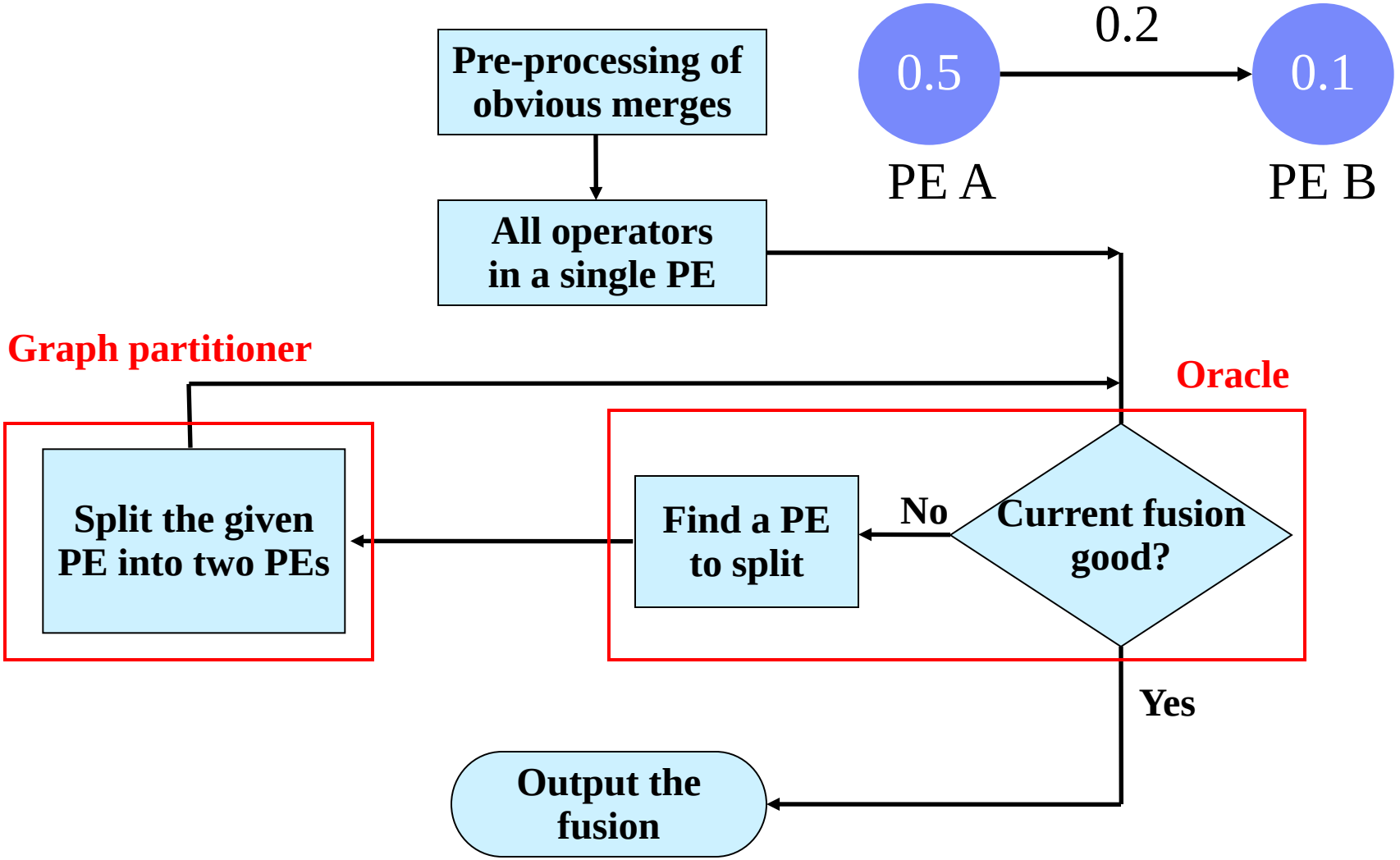
- **Objective**
  - Maximize the throughput when deployed
- **Surrogate objective**
  - Minimize the communication cost between PEs
- **Constraints**
  - Size of any PE (= total operator cost inside + total communication cost at the boundary) is bounded
  - The PEs fit well onto hosts
  - Several other constraints (later)

The SPADE fusion problem

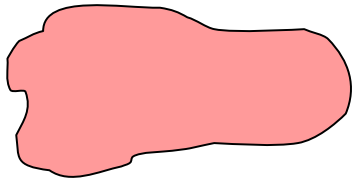


- It is unlikely that efficient algorithms exists to solve the problem optimally (NP-hard)
- Use approximation algorithms or heuristics to find near-optimal solutions quickly

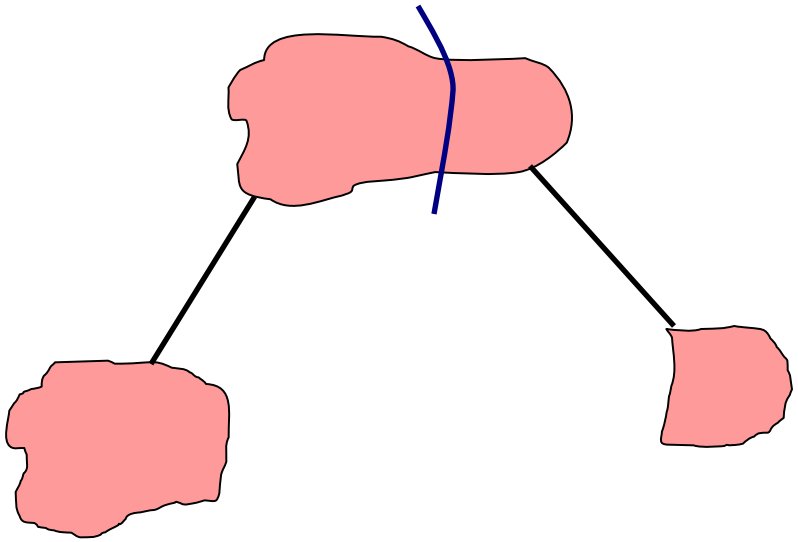
Our approach: Outline



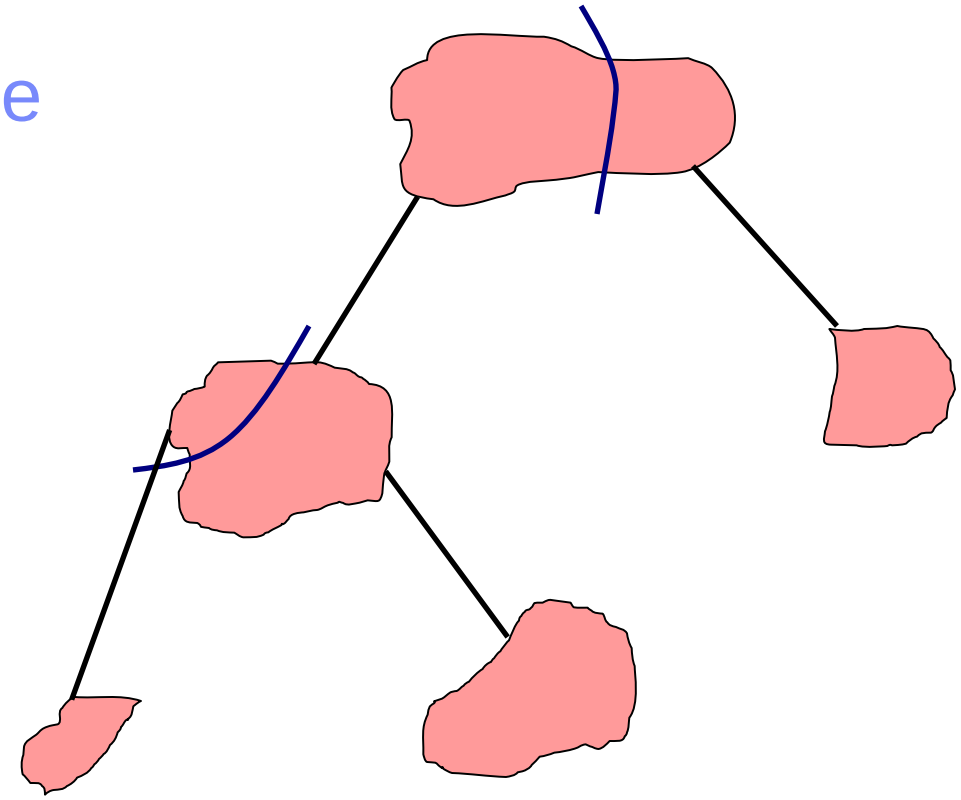
Binary Tree



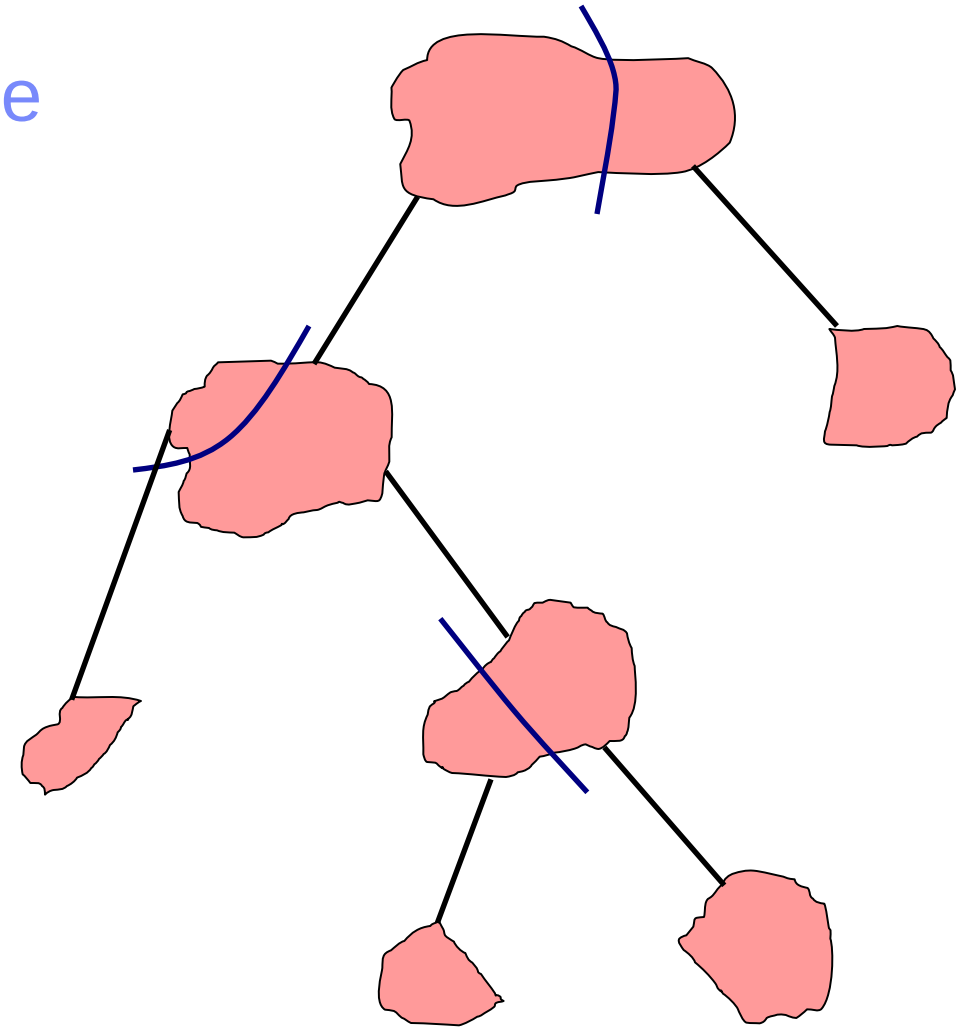
Binary Tree



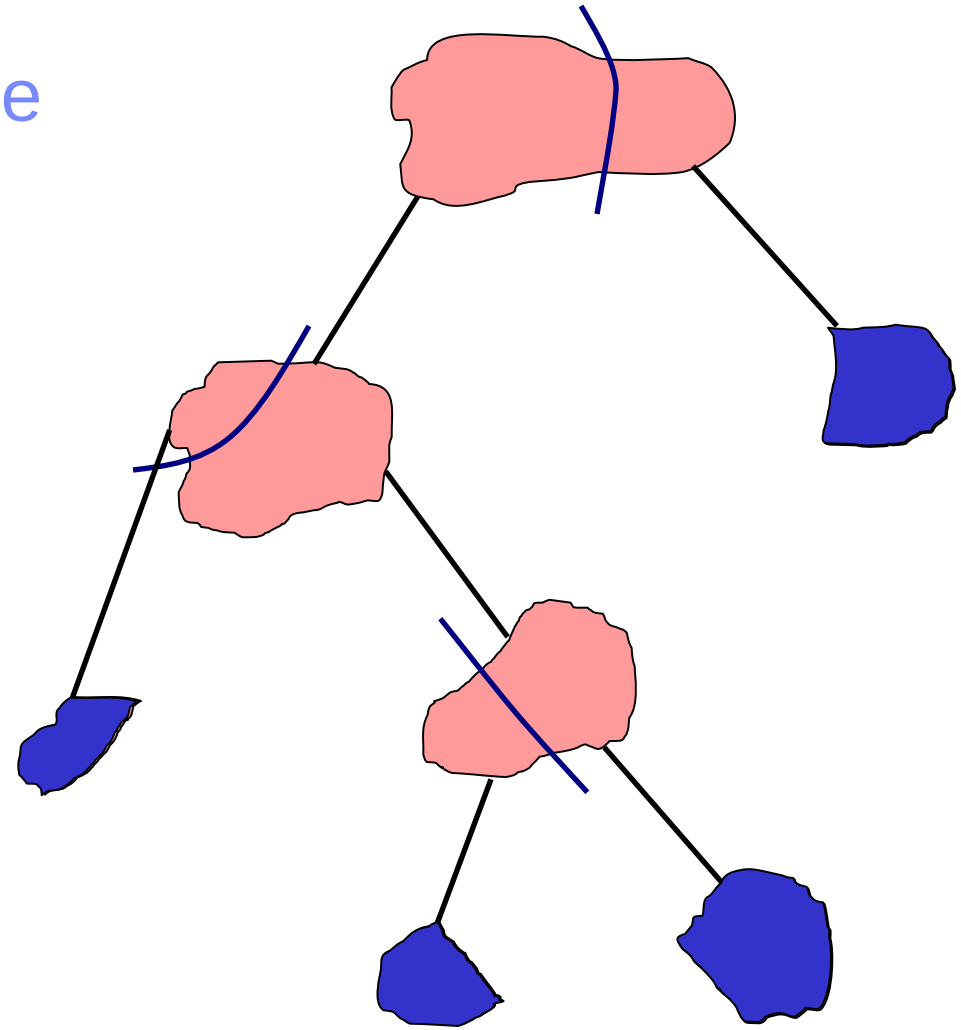
Binary Tree



Binary Tree

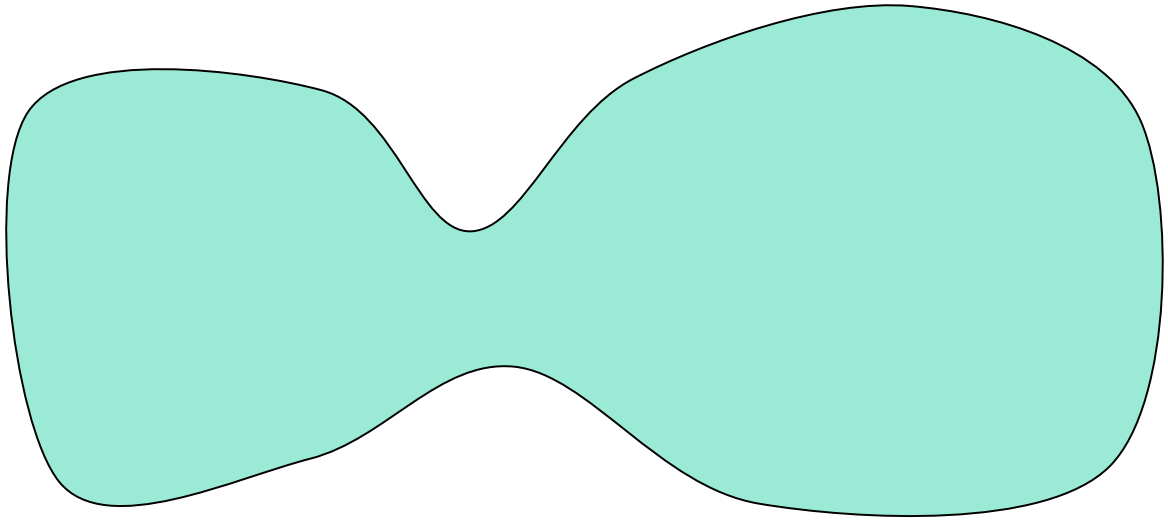


Binary Tree



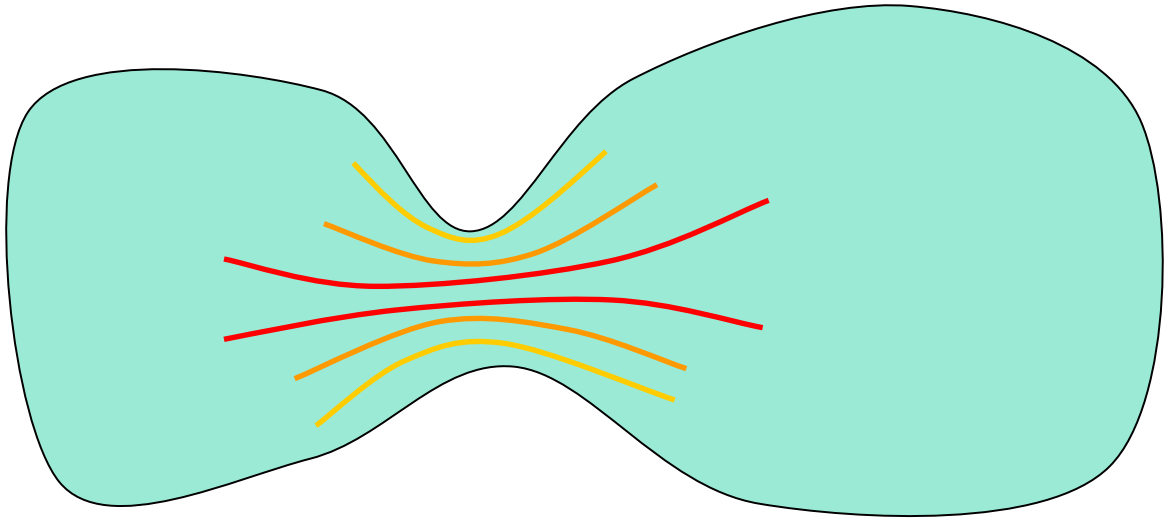
A fusion corresponds to a frontier in this binary tree.

Graph Partitioner: The intuition



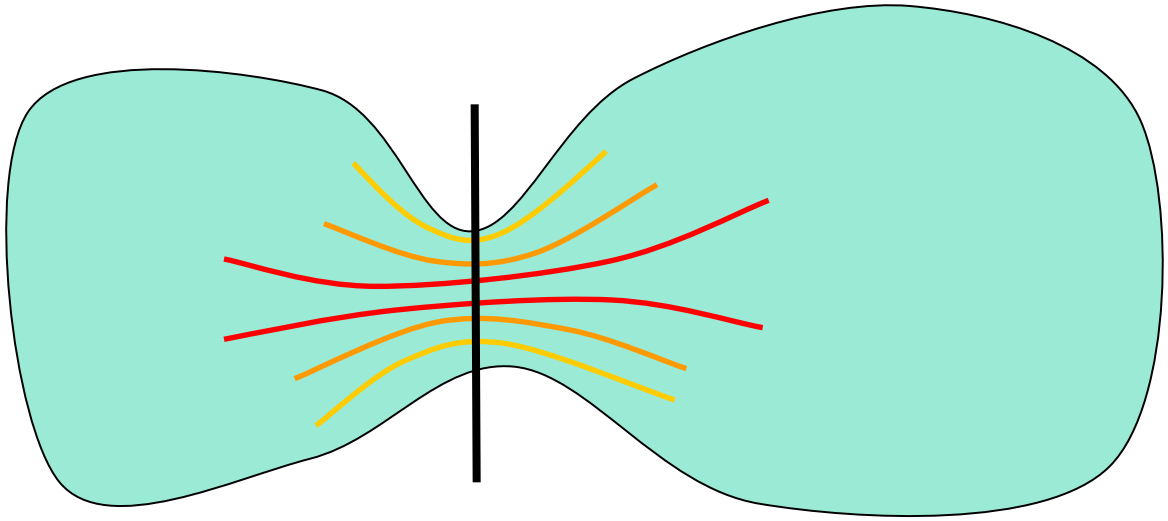
- Think of the operator graph as a system of pipes (arcs) capable of carrying a flow between operators.
- An LP Leighton and Rao [1988] sends a unit flow between every pair of operators.
- The congested arcs (full pipes) correspond to a bottleneck that sends a lot of flow from one side to the other.

Graph Partitioner: The intuition



- Think of the operator graph as a system of pipes (arcs) capable of carrying a flow between operators.
- An LP Leighton and Rao [1988] sends a unit flow between every pair of operators.
- The congested arcs (full pipes) correspond to a bottleneck that sends a lot of flow from one side to the other.

Graph Partitioner: The intuition



- Think of the operator graph as a system of pipes (arcs) capable of carrying a flow between operators.
- An LP Leighton and Rao [1988] sends a unit flow between every pair of operators.
- The congested arcs (full pipes) correspond to a bottleneck that sends a lot of flow from one side to the other.

Graph Partitioner

- Problem: Leighton-Rao uses a linear program with n^2 variables and constraints.
  - **Even with n = 1000, the LP has 1 million variables/constraints. This is too large for CPLEX (state-of-the-art LP solver) to handle.**
- Solution: We implement an approximation algorithm due to [Garg and Könemann 98] to solve this LP.
  - **An LP with a million variables can be solved in less than a second.**

COLA Oracle

- Role Given a fusion, determine if it is good enough. If not, find a PE to split next.
- Objectives
  - Keep the processor loads balanced, and
  - Keep the communication cost low
- Simplest version
  - Use “Longest Processing Time” (LPT) scheme to simulate a scheduler (assign PEs to processors).
  - If the PEs do not fit well, return the largest-size PE to split.
  - Oracle understands that hosts are multicore.
  - (Size of PE is computation cost of operators plus communication cost of incoming and outgoing edges)

Why simulate a scheduler?

- **Even if all PEs fit individually, they could not fit collectively.**
- **Example:**
  - Consider the case of four PEs taking up 51% of the node
    - Requires four hosts
    - Split one of those PEs, only three hosts are required even if there is an increase in total size
      - More PEs → more communication cost → total CPU requirement increases
- **For that reason, oracle tries to assign PEs to host, and doesn't just measure their size.**

COLA Features and Constraints

- **Heterogeneous processors**
- **Resource matching constraints: An operator has a limited set of hosts on which it is permitted**
- **Partition colocation constraints: Two operators must be in same partition**
- **Host colocation constraints: Two operators must be assignable by scheduler to the same host**
- **Partition exlocation constraints: Two operators must be in separate partitions**
- **Host exlocation constraints: Two operators must be assignable by scheduler to separate hosts**
- **High Availability constraints: Ensure that replicated pieces are placed on separate hosts**

Experimental Results

- **Experimental application is VWAP, a stock market data processing application**
  - Two different versions
  - Varied number of available hosts
- **Compare our strategy (COLA) to**
  - NONE: no fusion
  - FINT: built-in SPADE fusion optimizer
- **Compare two competing objectives (communication cost and PE size)**
- **Measure throughput**

Competing objectives: Results

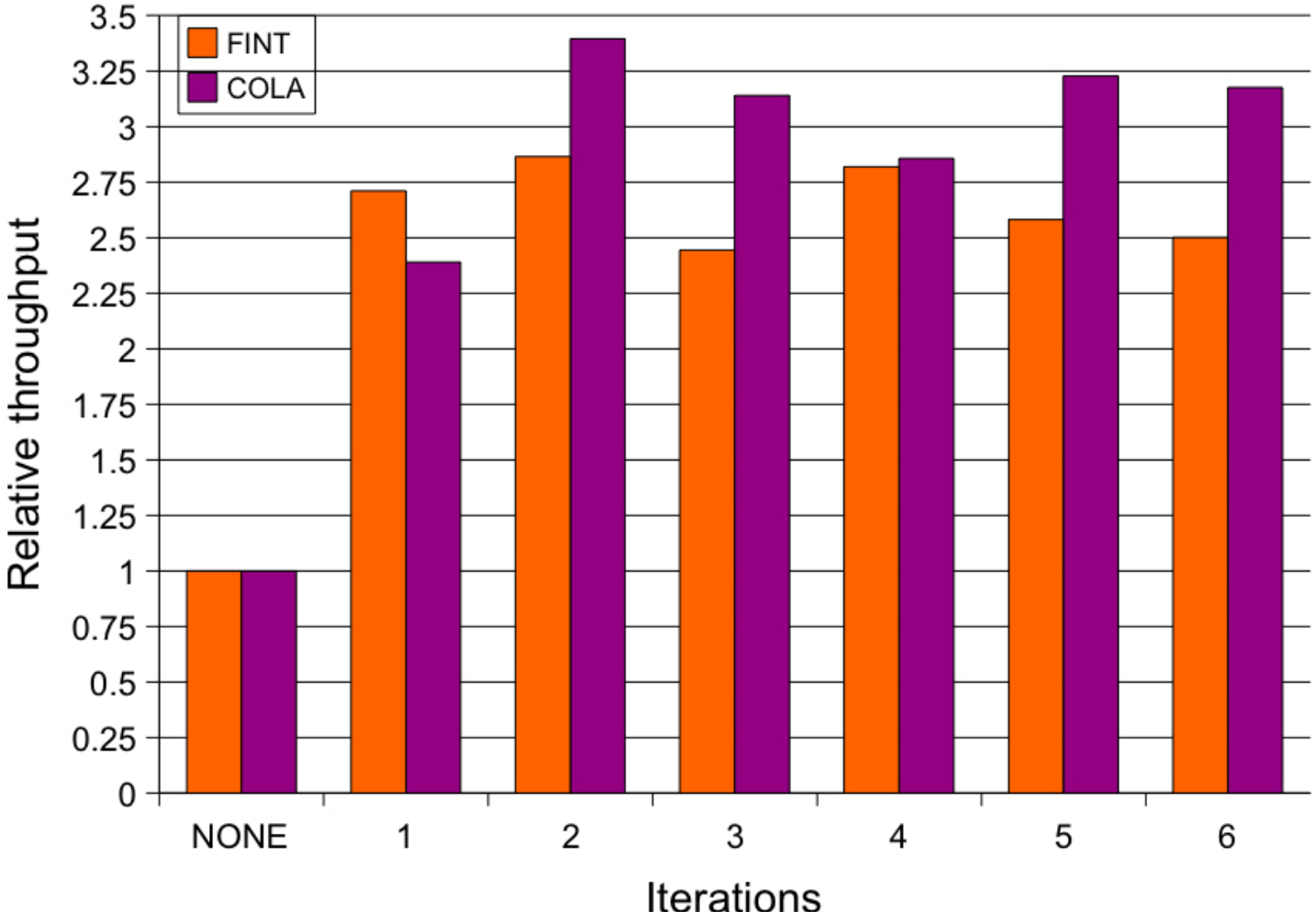
	NONE		FINT		COLA	
Hosts	Comm cost	max PE size	Comm cost	max PE size	Comm cost	max PE size
4	0.3792	0.3218	0.0661	0.5353	0.0332	0.4990
5	0.3703	0.2980	0.0441	0.5476	0.0587	0.4676
6	0.7236	0.8169	0.2698	0.8169	0.1998	0.7560
7	0.6833	0.6697	0.2492	0.6697	0.1860	0.6666

- COLA does better than FINT on cut size without larger PEs
  - In a few cases, the maximum PE size is actually **smaller** than the no fusion case!

Throughput

- At different throughput rates, the operator-costs and communication-costs do not scale linearly
  - **Initial run with a poorly-performing fusion results in low estimates of operator sizes, and we cannot just "scale up" to see how big operators can get.**
  - **Use an iterative scheme**
    - Use NONE to gather initial metrics
    - Have COLA find a fusion based on these metrics
    - Gather metrics with this new fusion
    - Use these new metrics as input to COLA and repeat

Iterating to fight inaccurate data



Results: Throughput

Hosts	NONE	FINT			COLA		
		1st	1st-local	Max	1st	1st-local	Max
4	99.4 (1)	273 (2.75)	295 (2.96)	295 (2.96)	284 (2.86)	286 (2.88)	303 (3.05)
5	97.2 (1)	263 (2.71)	279 (2.87)	279 (2.87)	232 (2.39)	330 (3.40)	330 (3.40)
6	189 (1)	286 (1.51)	286 (1.51)	293 (1.55)	280 (1.48)	379 (2.00)	391 (2.06)
7	179 (1)	268 (1.50)	322 (1.80)	336 (1.88)	267 (1.50)	349 (1.95)	363 (2.03)

- First row shows that COLA usually outperforms FINT
- Notice that as the number of hosts increases, the gain from the fusion strategy goes down

Conclusion

- **Graph partitioning is an effective way of determining which small operators to fuse together into PEs**
- **Future work**
  - Better characterization of a good-throughput fusion
  - Thread aware: PE size limits do not take into account whether the PE is single or multithreaded
  - Fast constraint checking
    - NP hard problem, but “difficult” instances likely to be rare