Resource Allocation for Multiple Concurrent In-network Stream-processing Applications

Anne Benoit Henri Casanova* Veronika Rehn-Sonigo
Yves Robert

LIP, École Normale Supérieure de Lyon
France

*University of Hawai'i at Manoa
USA

HeteroPar'2009
August 25, 2009

Introduction and Motivation

Operator-mapping problem for in-network stream processing

- Applications structured as trees of operators
- Execution in steady-state
- Multiple data objects are continually updated at various locations on a network
- Multiple concurrent applications

Applications?

- Processing of data in a sensor network
- Video surveillance
- Continuous queries on distributed relational databases
- Network monitoring

Introduction and Motivation

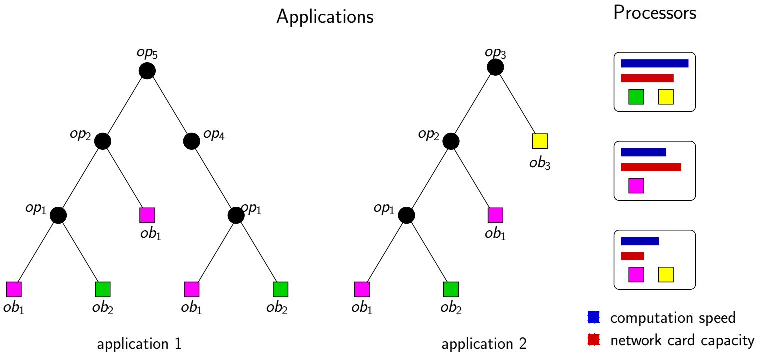
Operator-mapping problem for in-network stream processing

- Applications structured as trees of operators
- Execution in steady-state
- Multiple data objects are continually updated at various locations on a network
- Multiple concurrent applications

Applications?

- Processing of data in a sensor network
- Video surveillance
- Continuous queries on distributed relational databases
- Network monitoring

Rule of the Game

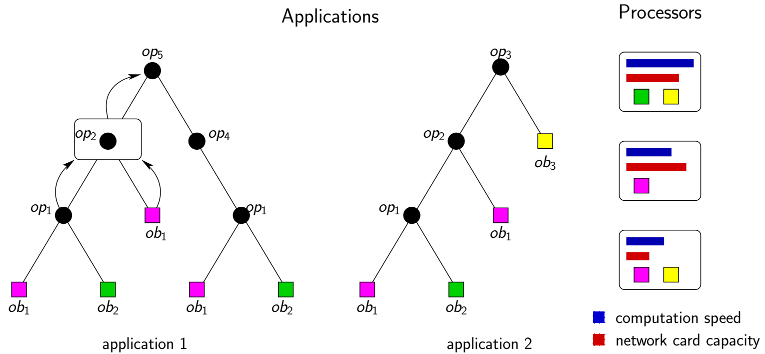


Goal

Minimize some cost function of the target platform while matching all application requirements.

Assess impact of reusing intermediate results.

Rule of the Game

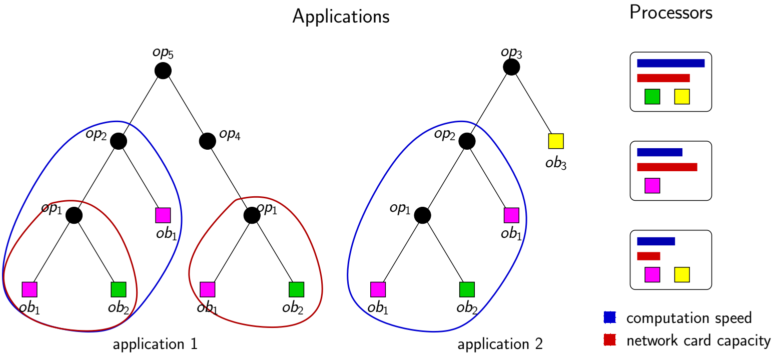


Goal

Minimize some cost function of the target platform while matching all application requirements.

Assess impact of reusing intermediate results.

Rule of the Game

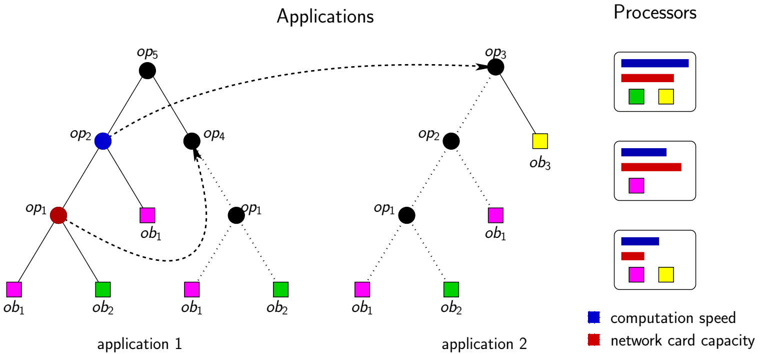


Goal

Minimize some cost function of the target platform while matching all application requirements.

Assess impact of reusing intermediate results.

Rule of the Game

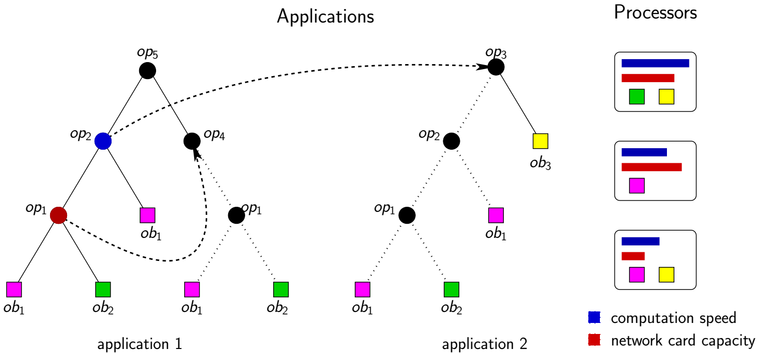


Goal

Minimize some cost function of the target platform while matching all application requirements.

Assess impact of reusing intermediate results.

Rule of the Game



Goal

Minimize some cost function of the target platform while matching all application requirements.

Assess impact of reusing intermediate results.

Major Contributions

Theory
- Definition operator-placement problem
- Problem complexity
- Linear programming formulation

Practice
- Polynomial heuristics
- Experiments to compare heuristics and evaluate their performance

Major Contributions

Theory
- Definition operator-placement problem
- Problem complexity
- Linear programming formulation

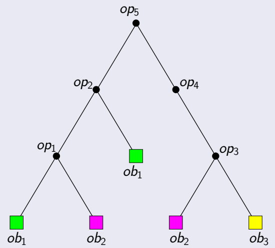
Practice
- Polynomial heuristics
- Experiments to compare heuristics and evaluate their performance

Outline of the Talk

The Application Model

- $\mathcal{K}$ applications
- $\mathcal{OP} = \{op_1, op_2, \dots\}$ set of operators
- $\mathcal{OB} = \{ob_1, ob_2, ob_3, \dots\}$ basic objects
- Computation of operator op_p: w_p operations, δ_p size of output

Application tree



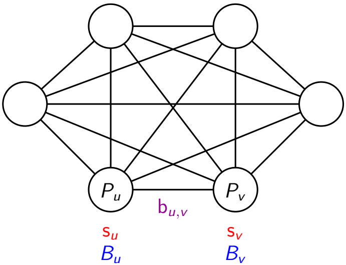
For application k:
$\rho^{(k)}$ application throughput

Object ob_j
- d_j size of ob_j
- $f_j^{(k)}$ download frequency
- $rate_j^{(k)} = d_j \times f_j^{(k)}$ bandwidth consumption

Platform and Communication Model

The platform

- $\mathcal{P}$ processors, fully connected graph (i.e., a clique)
- s_u: compute speed of proc. $P_u \in \mathcal{P}$
- B_u: network card capacity of $P_u \in \mathcal{P}$
- $b_{u,v} (= b_{v,u})$: bandwidth of bidirectional link between P_u and P_v



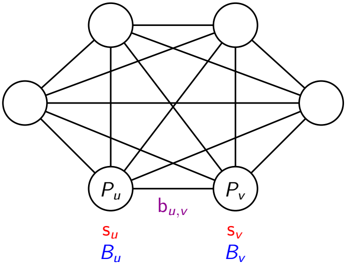
Communication Model

Full-overlap, bounded multi-port model: processor P_u can be involved in computing, sending data, and receiving data simultaneously.

Platform and Communication Model

The platform

- $\mathcal{P}$ processors, fully connected graph (i.e., a clique)
- s_u: compute speed of proc. $P_u \in \mathcal{P}$
- B_u: network card capacity of $P_u \in \mathcal{P}$
- $b_{u,v} (= b_{v,u})$: bandwidth of bidirectional link between P_u and P_v

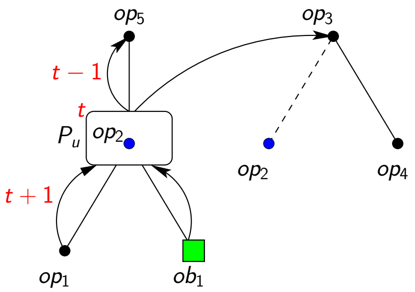


Communication Model

Full-overlap, bounded multi-port model: processor P_u can be involved in computing, sending data, and receiving data simultaneously.

The Mapping Model

- Each processor is in charge of one or several tree nodes
- Node $n_i^{(k)}$ (op_p) mapped on processor P_u
- P_u computes t-th final result
- Sends to parent node(s) (if any) intermediate results for $(t-1)$-th final result
- Receives data from its non-leaf children (if any) for computing the $(t+1)$-th final result



Mapping Model

Constraints

- Application throughput $\rho^{(k)}$: $\forall P_u \in \mathcal{P}$,
$\sum_{p \in a_{op}(u)} \left(\max_{(k,i) \in \bar{a}(u) \mid op(n_i^{(k)}) = op_p} \left(\rho^{(k)}\right) \frac{w_p}{s_u} \right) \leq 1$
- Bandwidth capacity P_u: $\forall P_u \in \mathcal{P}$,
$\sum_{(j,v,k) \in Do(u)} rate_j^{(k)} + \sum_{P_v \in \mathcal{P}} \sum_{(j,u,k) \in Do(v)} rate_j^{(k)} + \sum_{(p,v,k) \in Ch(u)} \delta_p \rho^{(k)} + \sum_{(p,v,k) \in Par(u)} \delta_p \rho^{(k)} \leq B_u$
- Link bandwidth $P_u \longleftrightarrow P_v$: $\forall P_u, P_v \in \mathcal{P}$,
$\sum_{(j,v,k) \in Do(u)} rate_j^{(k)} + \sum_{(j,u,k) \in Do(v)} rate_j^{(k)} + \sum_{(p,v,k) \in Ch(u)} \delta_p \rho^{(k)} + \sum_{(p,v,k) \in Par(u)} \delta_p \rho^{(k)} \leq b_{u,v}$

Optimization Problems

Objective

Map operators onto processors such that a cost function is minimized and all application throughputs are achieved.

- **PROC-NB** minimizes the number of used processors;
- **PROC-POWER** minimizes the compute capacity and/or the network card capacity of used processors (e.g., a linear function of both criteria);
- **BW-SUM** minimizes the sum of the used bandwidth capacities;
- **BW-MAX** minimizes the maximum percentage of bandwidth used on all links.

Outline of the Talk

1. Framework
2. Complexity
3. Heuristics and Experiments
4. Conclusion

Complexity

All optimization problems are NP-hard.

- **PROC-NB** NP-complete in the strong sense even for a simple case: a HOM platform and a single application ($|\mathcal{K}| = 1$), that is structured as a left-deep tree, in which all operators take the same amount of time to compute and produce results of size 0, and in which all basic objects have the same size.
- **PROC-POWER** same proof as for PROC-NB.
- **BW-MAX** Reduction to 2-Partition: download objects with different rates on two processors for a single application.
- **BW-SUM** Reduction to Knapsack problem.

Integer Linear Programming

- Integer LP to solve the different optimization problems
- Many integer variables: no efficient algorithm to solve
- Approach limited to small problem instances

Outline of the Talk

Overview of Heuristics (1)

Heuristics for the PROC-POWER problem, considering the compute capacities of used processors.

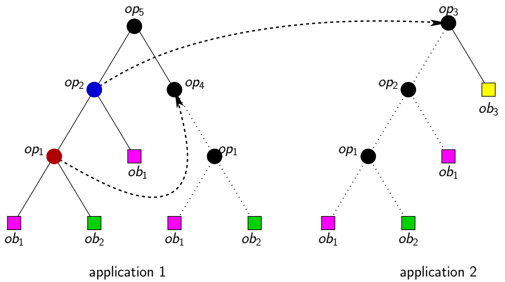
Server selection strategies:

- (S1) Select the fastest processor (blocking);
- (S2) Select the processor with the fastest network card (blocking);
- (S3) Select the fastest processor (non-blocking);
- (S4) Select the processor with the fastest network card (non-blocking).

Overview of Heuristics (2)

Heuristics: Reuse of intermediate results

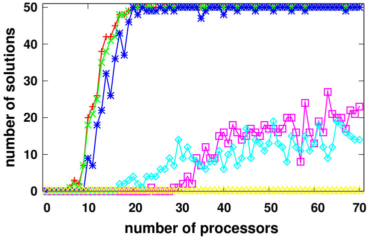
(H1) RandomNoReuse
(H2) Random
(H3) TopDownBFS
(H4) TopDownDFS
(H5) BottomUpBFS
(H6) BottomUpDFS



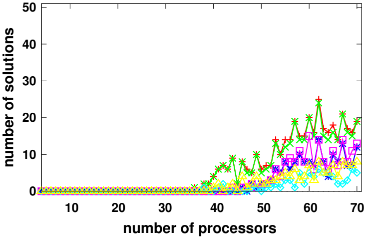
Results

Number of processors increases.
50 runs. 5 applications. 50 operators.

Successful runs.



(S3) Fastest proc.

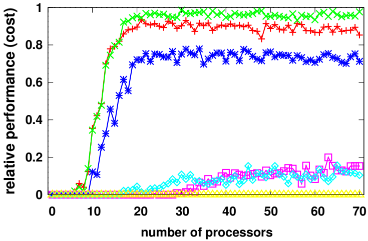


(S3) Fastest proc - no reuse.

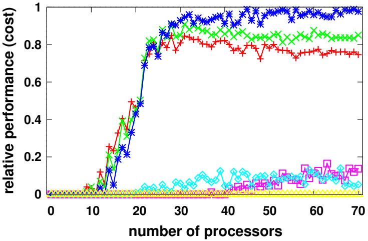
Results

Number of processors increases.
50 runs. 5 applications. 50 operators.

Relative performance.



(S3) Fastest proc.

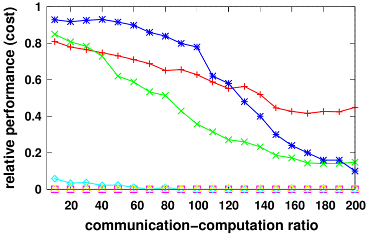


(S1) Fastest proc - blocking.

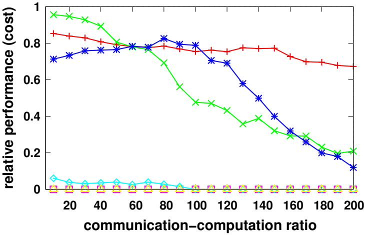
Results

Communication-computation ratio increases.
50 runs. 5 applications. 50 operators.

Relative performance.



(S1) Fastest proc - blocking.



(S2) Fastest netw. card - block.

Summary

- Random approach dramatically bad
- Neglecting reuse limits success rate and quality of solution in terms of cost
- Top Down approach turns out to be the best
- BottomUp only with BFS competitive
- DFS unable to reuse results efficiently (bandwidth)
- Strong dependency of processor selection strategy on solution quality
- Solid combination: TopDownBFS with fastest proc - non-blocking

Outline of the Talk

Related Work

Babu et al., Liu et al.

Execution of continuous queries on data streams

Chen et al., van Rennesse et al.

In-network stream processing systems
These systems all face the same question: where should operators be mapped in the network?

Pietzuch et al., Srivastava et al.

Operator-mapping problem for in-network stream processing

Conclusion

Resource allocation for multiple concurrent in-network stream processing applications

- Multiple concurrent applications
- Reuse of intermediate results
- Formulation of different operator-placement problems
- Complexity analysis: NP-completeness for all optimization problems
- Integer linear programming formulation

Practical side

- Polynomial time heuristics
- Simulation: TopDownBFS with fastest proc - non blocking

Conclusion

Resource allocation for multiple concurrent in-network stream processing applications

- Multiple concurrent applications
- Reuse of intermediate results
- Formulation of different operator-placement problems
- Complexity analysis: NP-completeness for all optimization problems
- Integer linear programming formulation

Practical side

- Polynomial time heuristics
- Simulation: TopDownBFS with fastest proc - non blocking

Perspectives

- Heuristics for the other optimization problems: PROC-NB, BW-SUM, BW-MAX
- More general cost function $w_{i,u}$ (time required to compute operator i onto processor u) $\longrightarrow$ more heterogeneity
- Mutable applications: Operators can be rearranged based on operator associativity and commutativity rules

  Ex: relational database applications