CWI Centrum Wiskunde & Informatica

Adaptive parallelization in multi-core systems

Mrunal Gawade, Martin Kersten

CWI, Amsterdam

Erasmus University Rotterdam
29th Nov 2013

monetdb

Why Parallelize Adaptively?

- Multi-core plan space is very large

  (as compared to a serial plan space)

- Statistics maintenance overheads

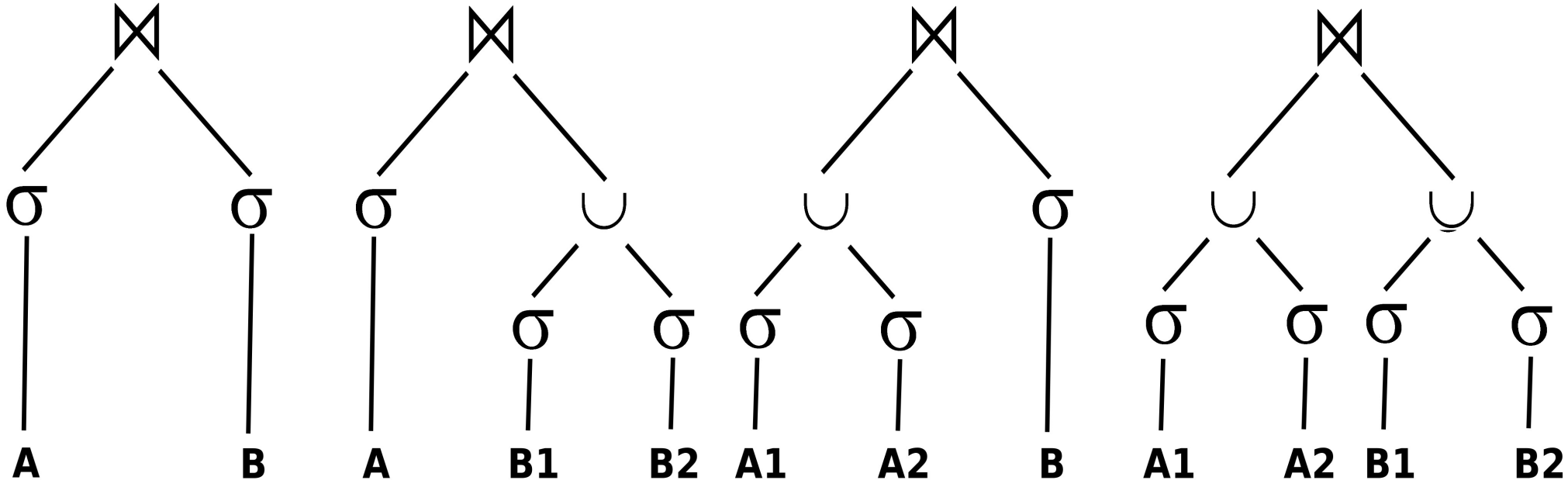
  (cost based optimizers)

- Optimal degree of parallelism is hard to find

  (resource variations)

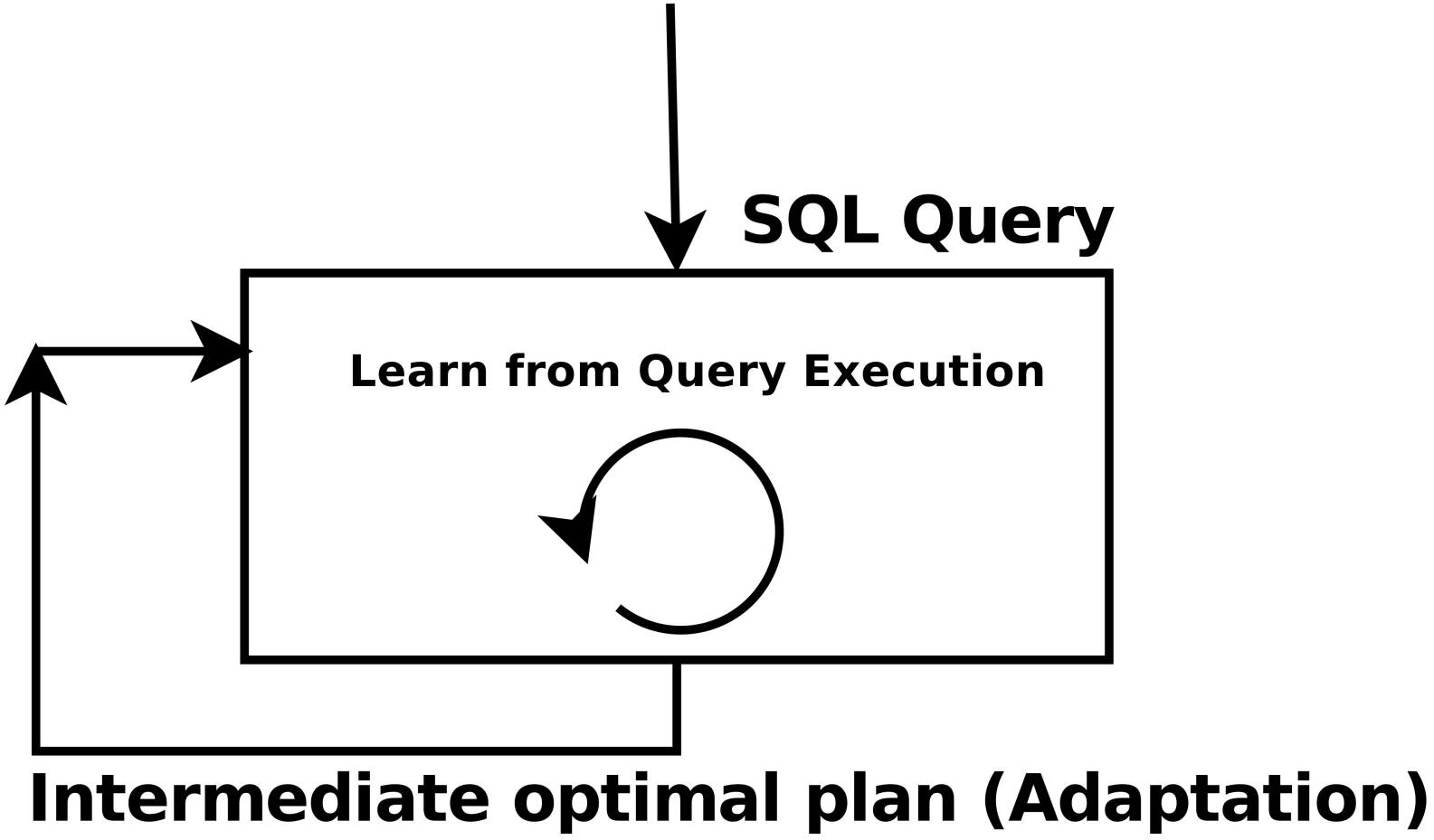
Multi-core Plan Space

Join (Select(A), Select(B))

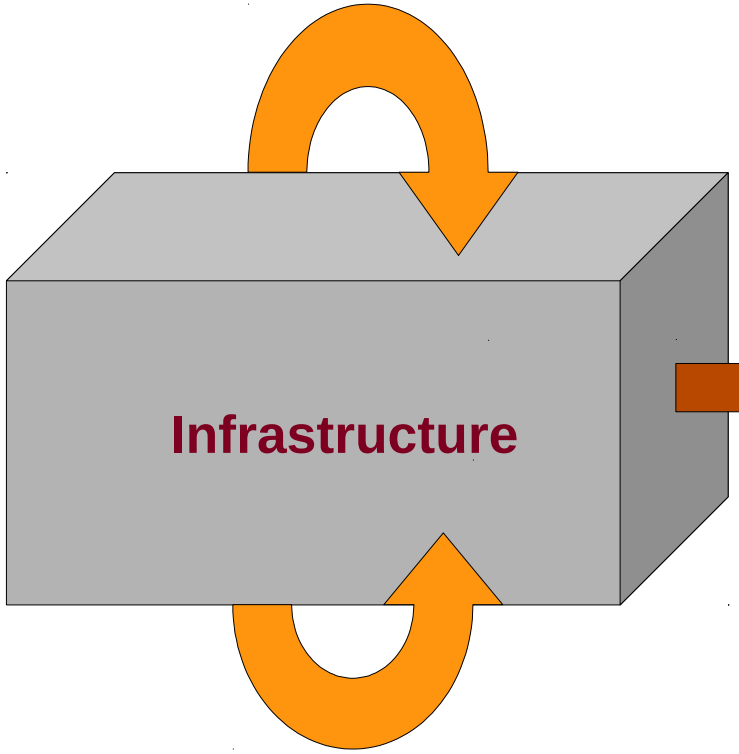


Plan 1 **Plan 2** **Plan 3** **Plan 4**

Bird's Eye View

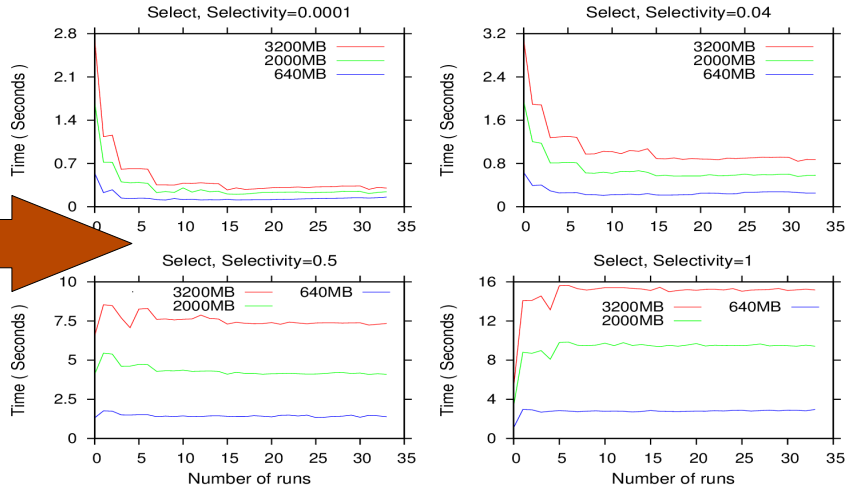


Contributions



Convergence Algorithm

Select & Join Operator Analysis



Infrastructure

Design

- Query execution time is the decision metric
- Store profiled history of plans
- Mutate an old plan to generate a new plan
- Parallelize the most expensive operator

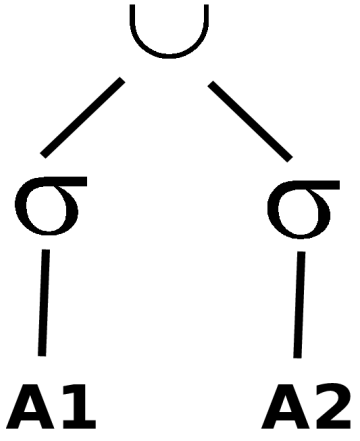
  (from previous query invocation)

Select Operator Mutation

X_1 = select (A, low, high);

- A1 = partition (A, 2, 0);
- A2 = partition (A, 2, 1);

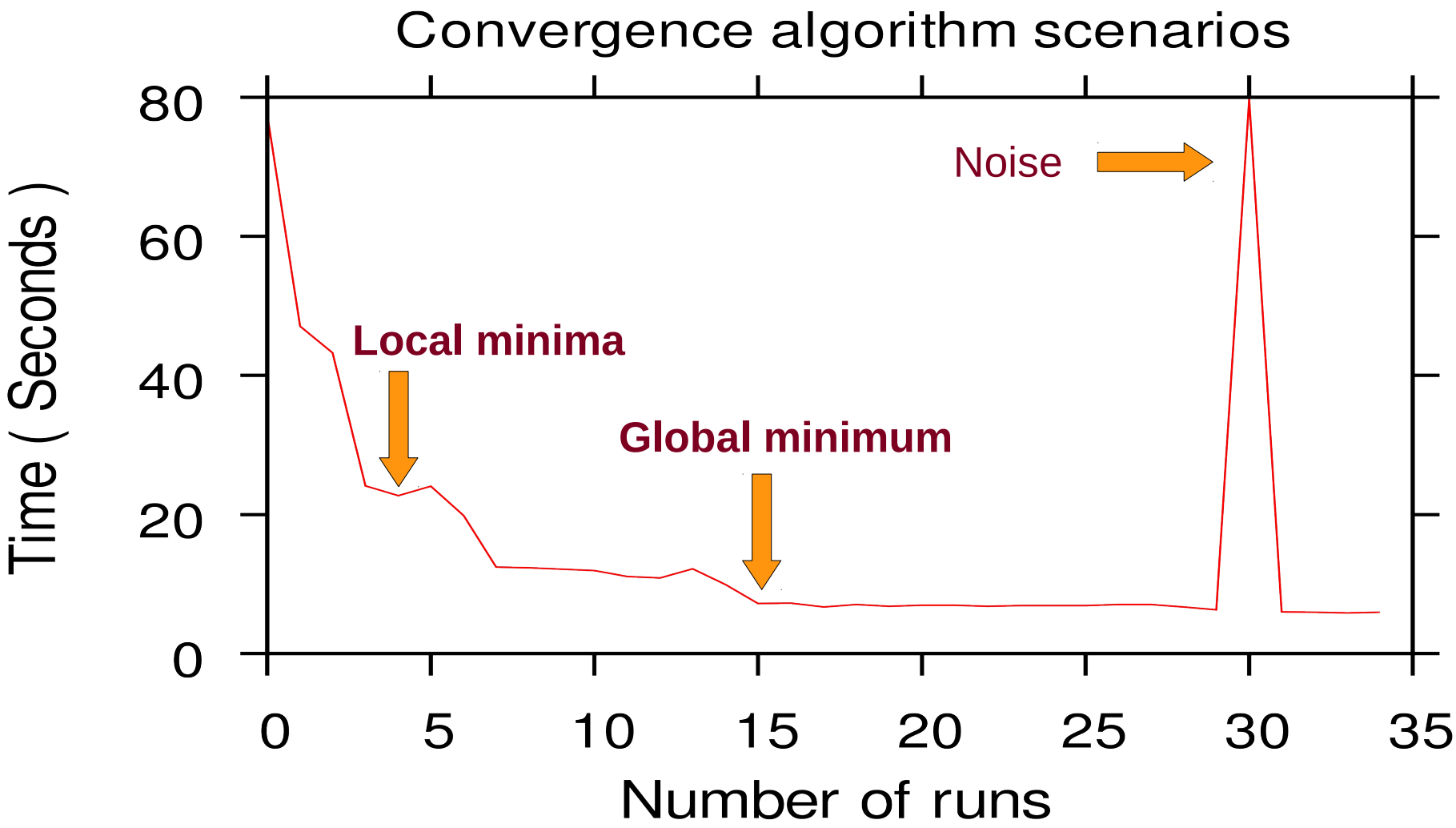
- Y_1 = select (A1, low, high);
- Y_2 = select (A2, low, high);



- **X_1** = pack(Y_1, Y_2);

Convergence Algorithm

Sample Convergence Case



Convergence Algorithm

ROI = Rate Of Improvement

CET = Current run Execution Time.

PET = Previous run Execution Time.

ROI = (CET – PET) / MAX(CET, PET)

Credit = Credit + (ROI x Number Of Cores)

Debit = Debit + (ABS(ROI) x Number Of Cores)

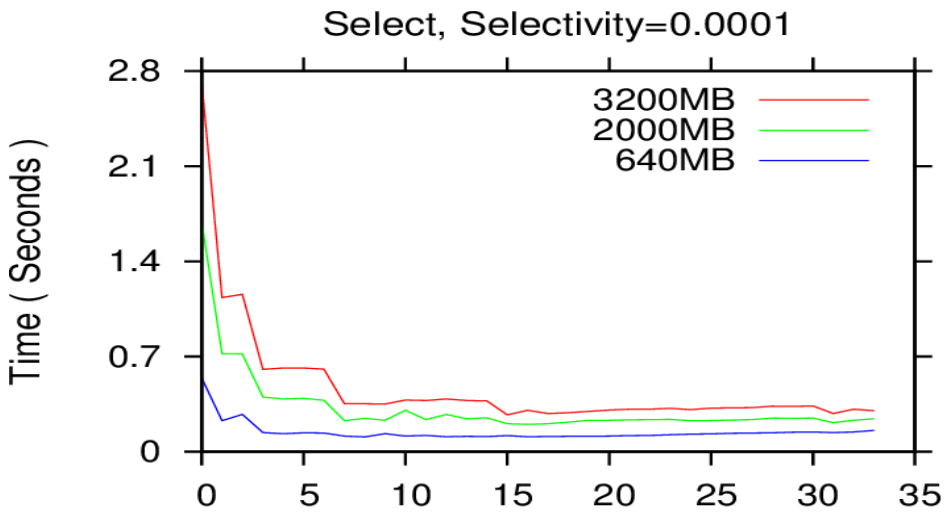
Convergence Runs = Credit - Debit = *f(ROI)*

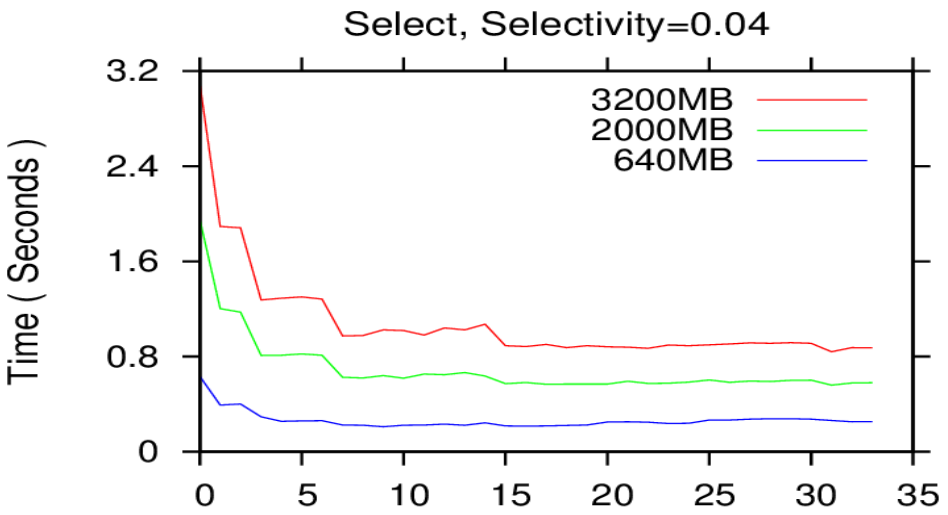
Select Operator Analysis

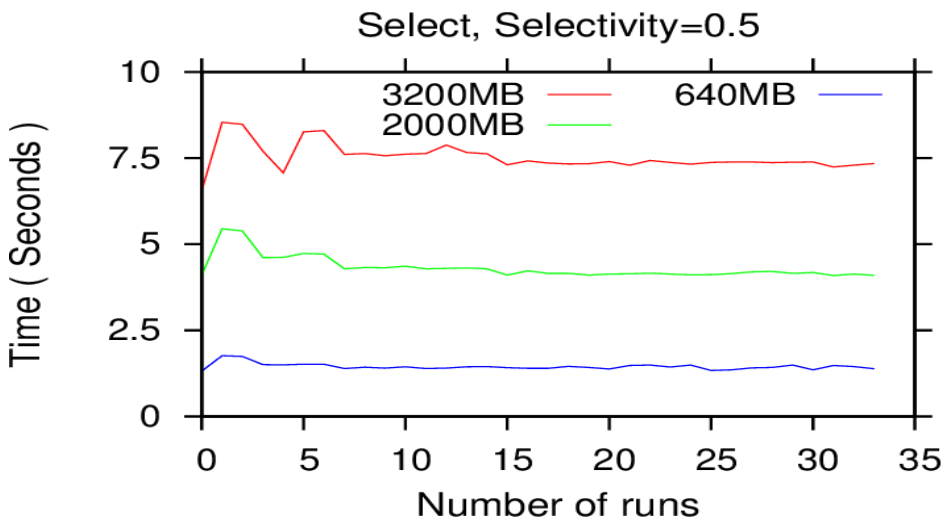
Experiment platform

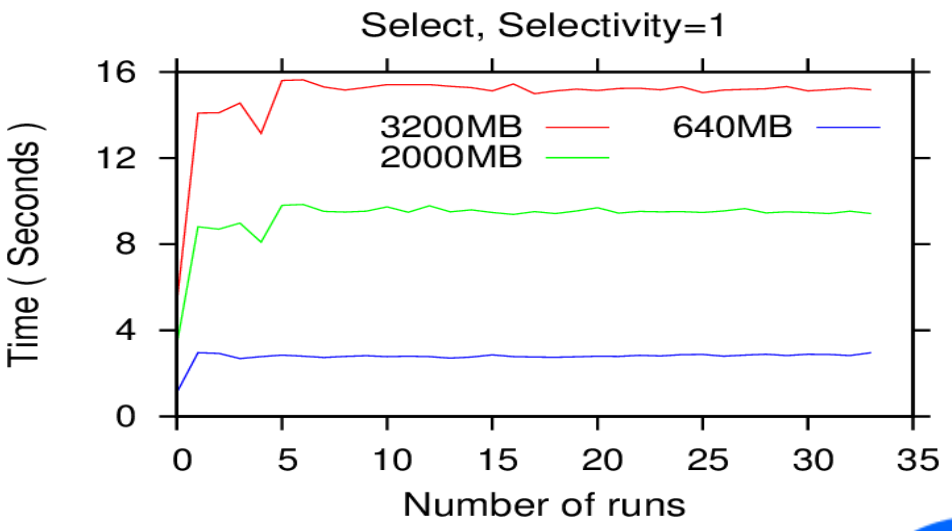
- Intel Xeon E5-2650@ 2.00GHz CPU (Dual socket)
  16 physical cores, 32 threads with hyper-threading
- Cache shared L3 = 20MB.
- 256 GB of 4 channel DDR3 RAM.
- MonetDB on Fedora Core 16.

Adaptive Parallelization of Select









Speed-up & Partitioning

Size(MB)	Selectivity		# Optimal Partitions
	0.0001	**0.04**	
	Speed-up		
3200	10	3.5	15
2000	7.5	3	7
640	4	2.5	3

TPC-H queries

Queries	Adaptive Parallelization	Heuristic Parallelization
Q 3	21.6 seconds	36 seconds
Q 5	25	28
Q 8	17.7	24
Q 14	4.1	4.1
Q 17	240	250
Q 20	42.5	45

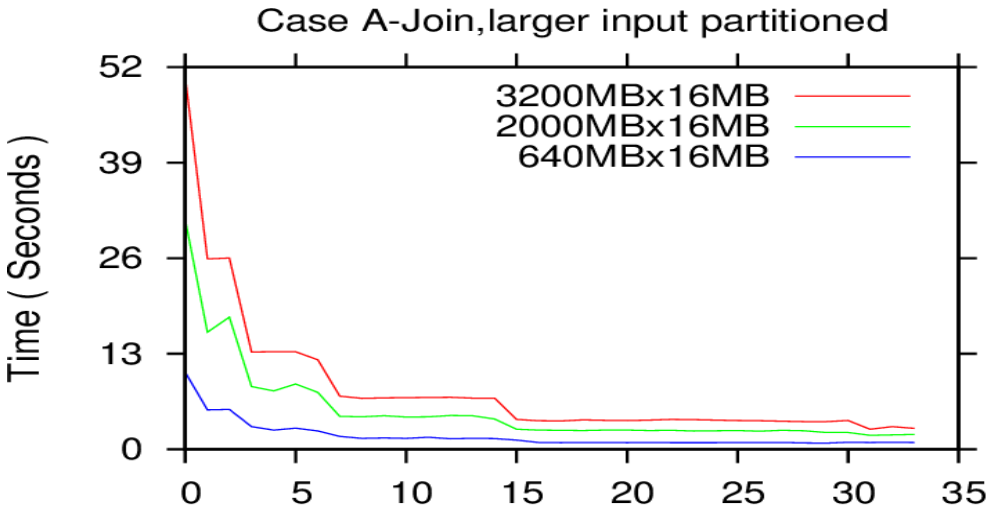
- 11 queries show lower performance (mostly short queries)
- 5 queries not considered

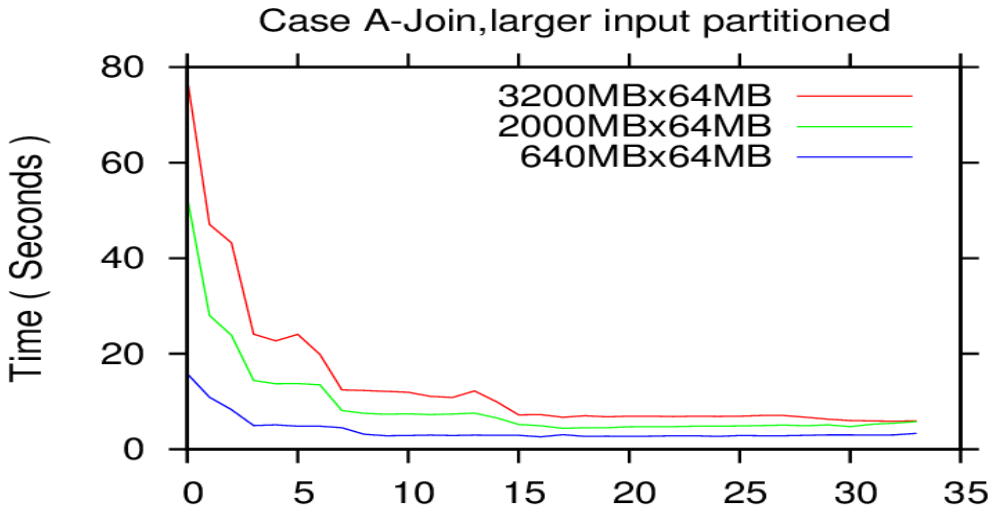
Summary

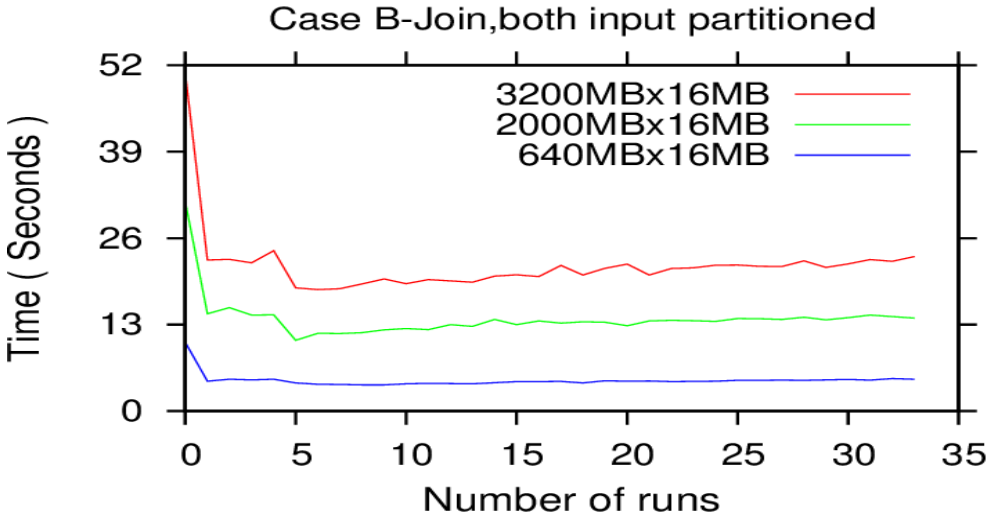
- Detailed analysis of Degree of Parallelism of plans
- Convergence algorithm adjusts dynamically
- High selectivity operator speed-up of an order of magnitude
- Long running TPC-H queries benefit

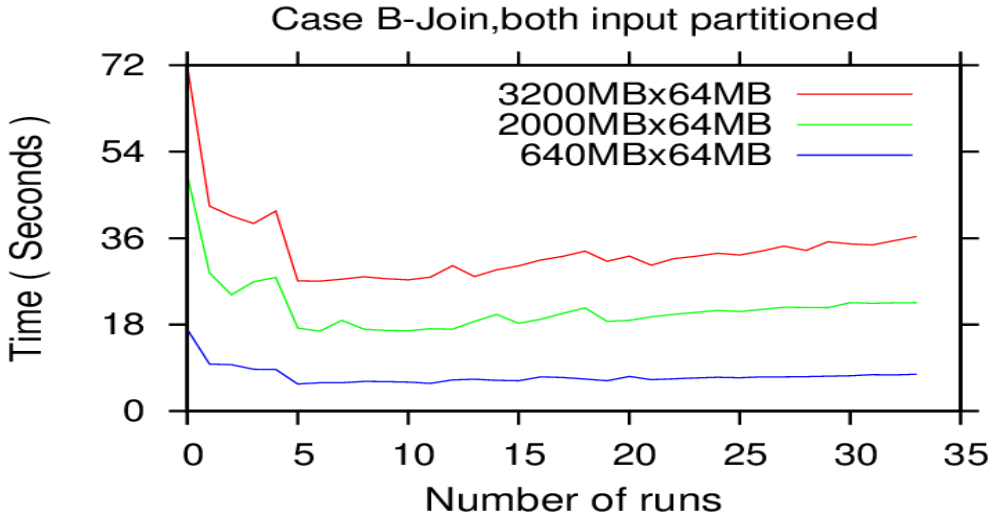
Thank you

Adaptive Parallelization of Join





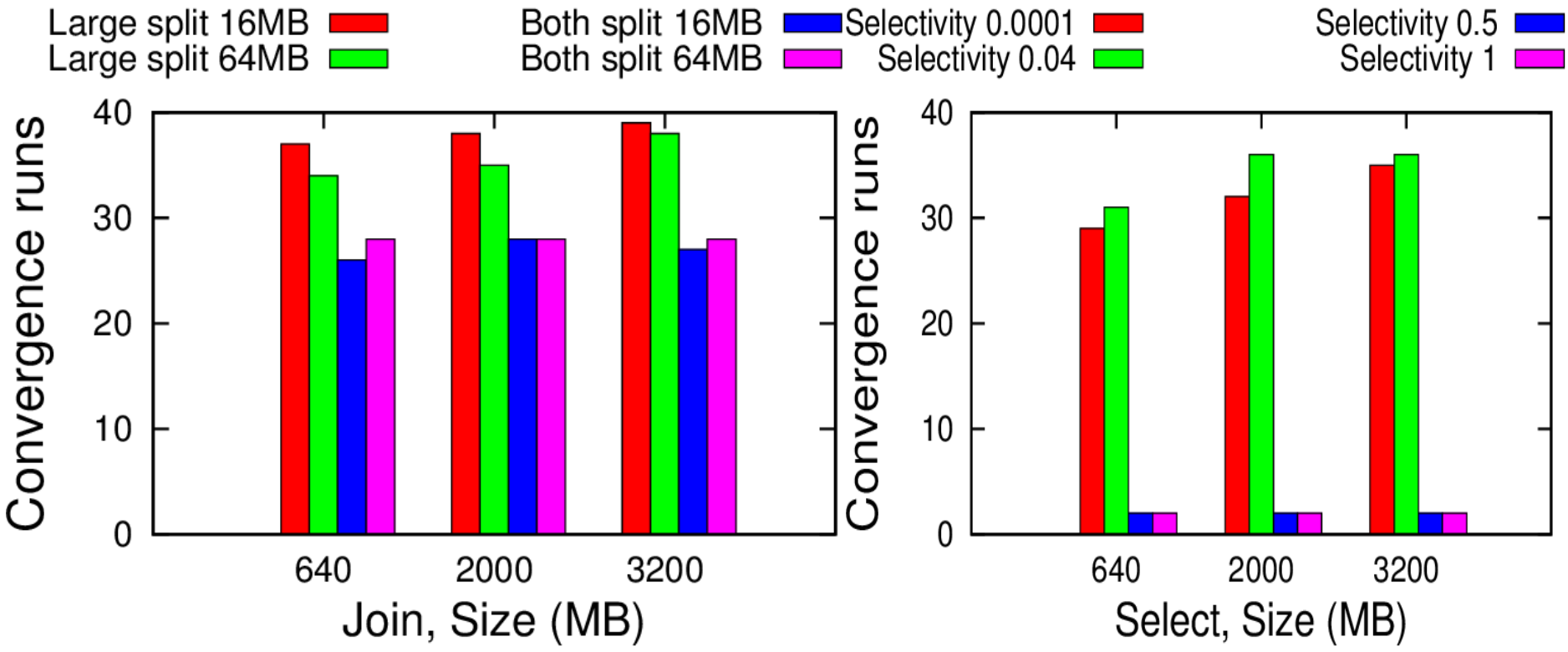




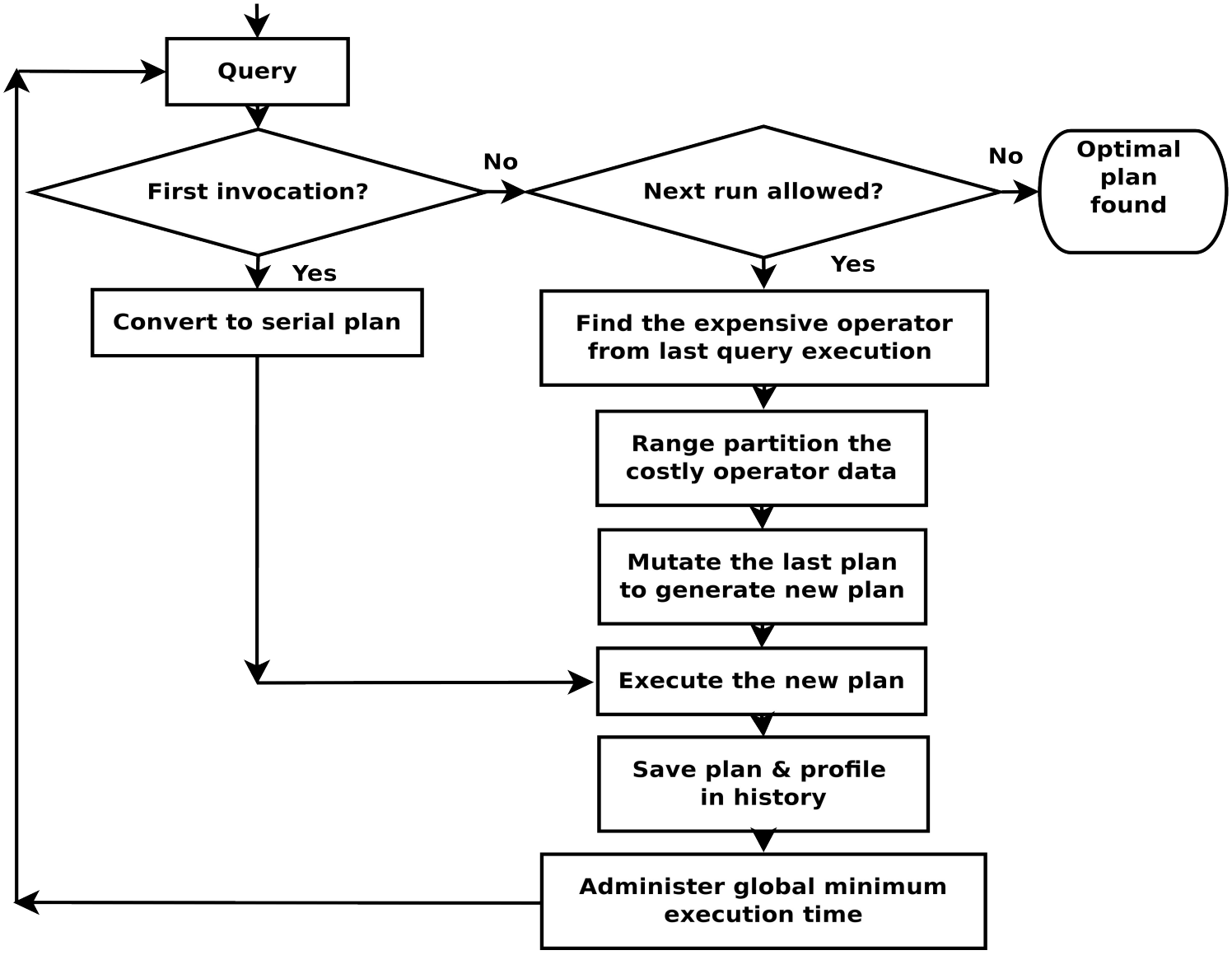
Speed-up & Partitioning

Size (MB)	16	64	16	64	16	64	16	64
	Case A		Case B		Case A		Case B	
	Speed-up				#Partitions			
3200	18.5	12	2.75	2.5	31	31	16	16
2000	16	12	3	3	31	15	16	16
640	11	6	2.5	3	15	15	28	16

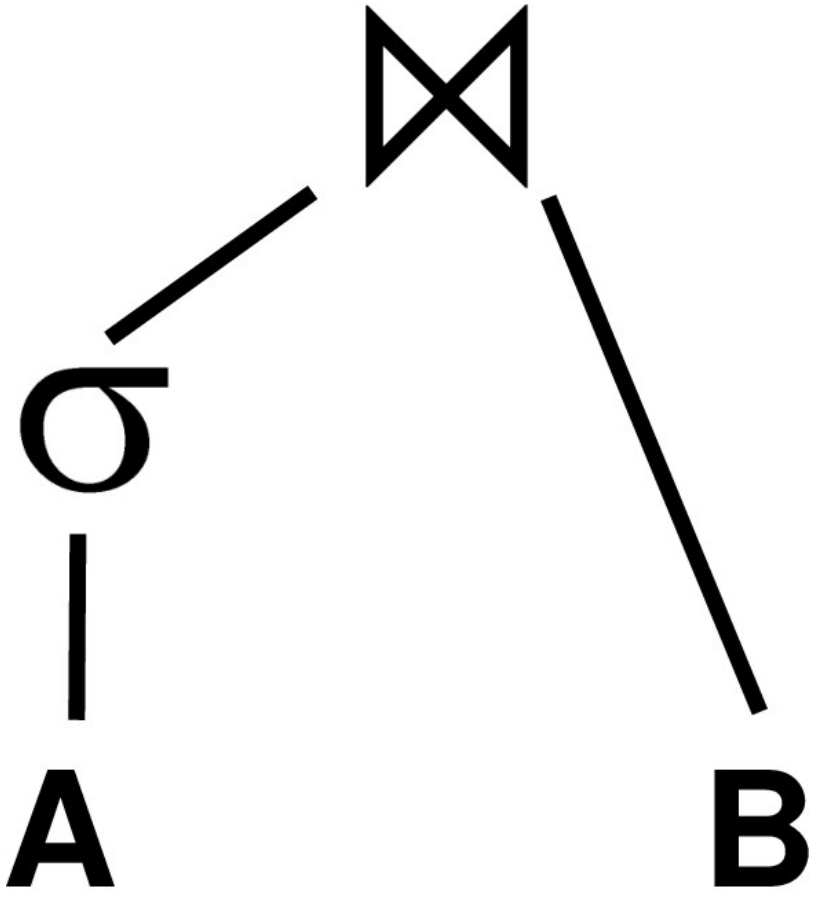
Convergence Runs Dynamic Adjustment



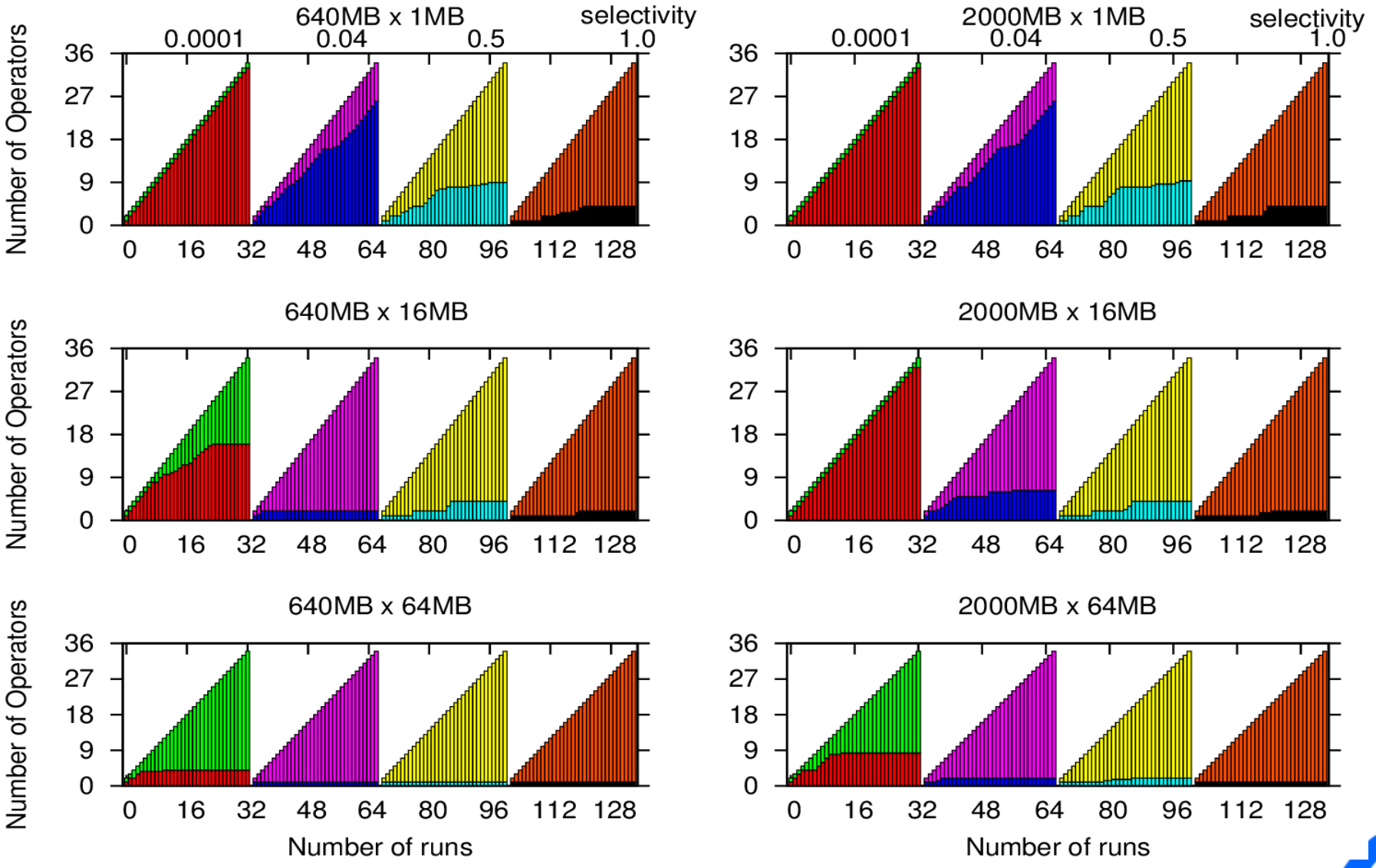
Work-flow


Query
First invocation?
No
Next run allowed?
No
Optimal plan found
Yes
Yes
Convert to serial plan
Find the expensive operator from last query execution
Range partition the costly operator data
Mutate the last plan to generate new plan
Execute the new plan
Save plan & profile in history
Administer global minimum execution time

Select-Join Plan



Select-Join Degree of Parallelism



monetdb

Select-Join execution