Limitations of data reuse in streaming iterative algorithms

Erik Bodzsar and Andrew A. Chien

Large-Scale Systems Group

THE UNIVERSITY OF CHICAGO
COMPUTER SCIENCE

STREAMING ITERATIVE ALGORITHMS

Traverse all data
High degree of parallelism

Can we reorder computation to exploit data reuse and save I/O?

SCALE-OUT PROBLEMS

MapReduce

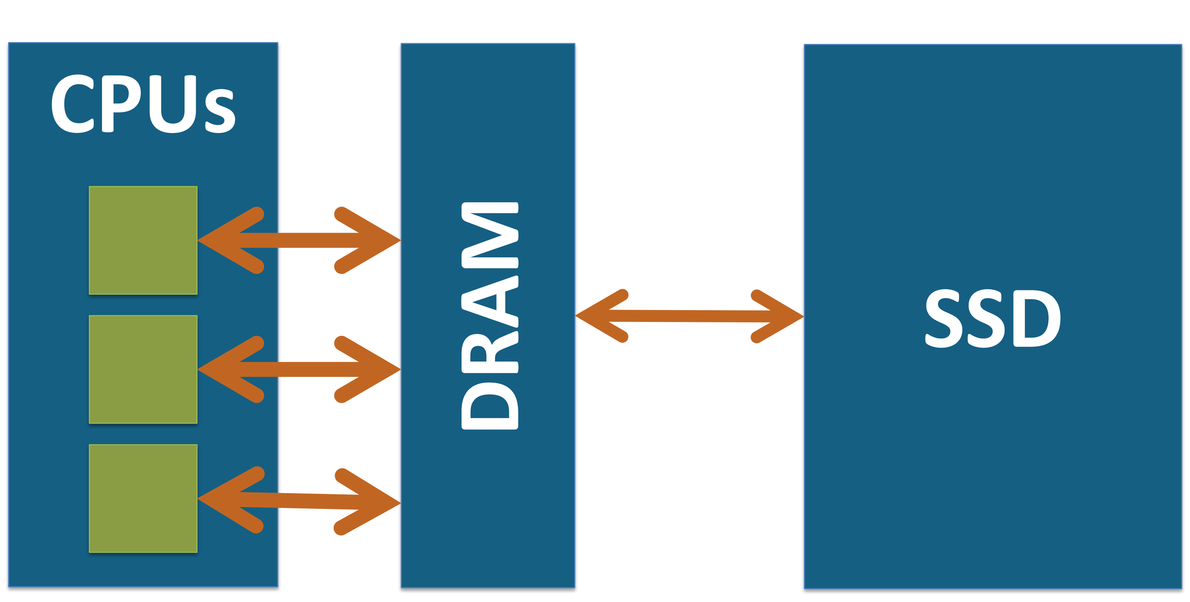
Restrictive programming model

In-memory systems

High cost
Low fault-tolerance

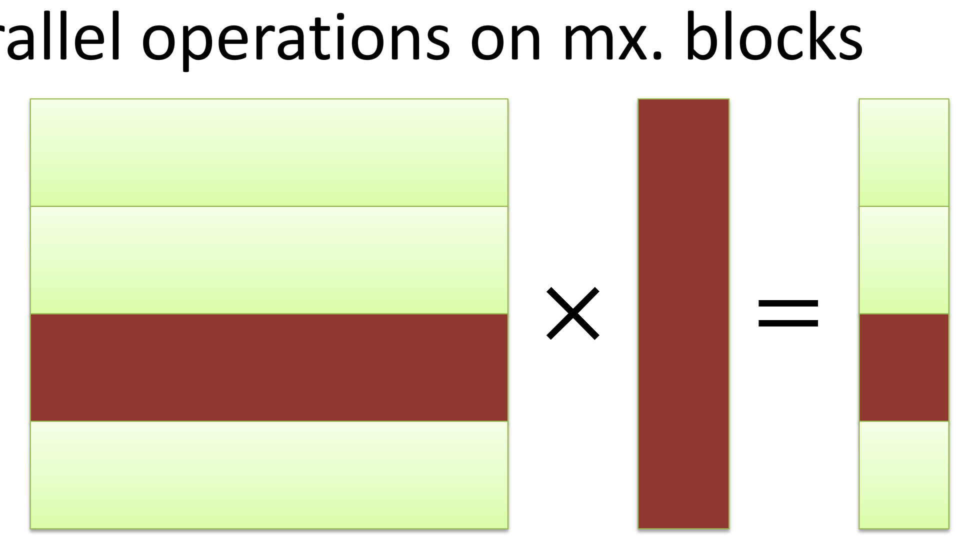
Scale up!

Increase capacity with SSDs
Big data, small memory
Manage I/O explicitly
Exploit application knowledge



PROGRAMMING MODEL & IMPLEMENTATION

Parallel operations on mx. blocks

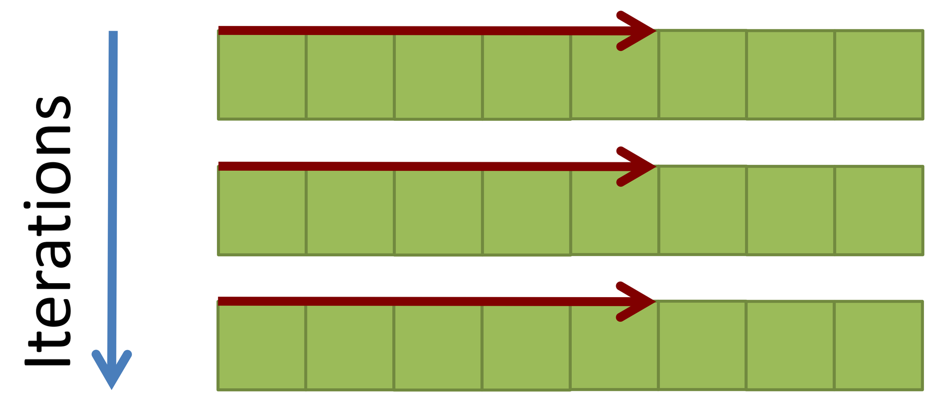


Blockus system; based on Presto, a distributed execution engine for R

SCHEDULING POLICIES

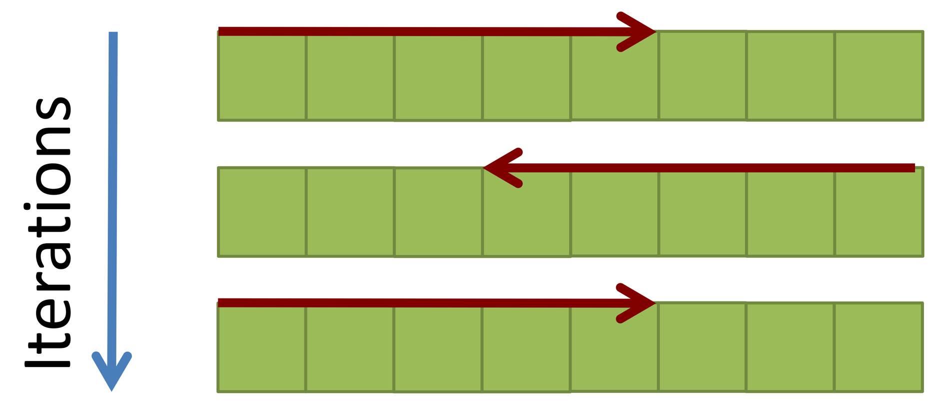
Default

Traverse data in fixed order
Corresponds to for loop



Reversing

Reverse traversal in every iteration
Reuse final blocks of prev. iteration

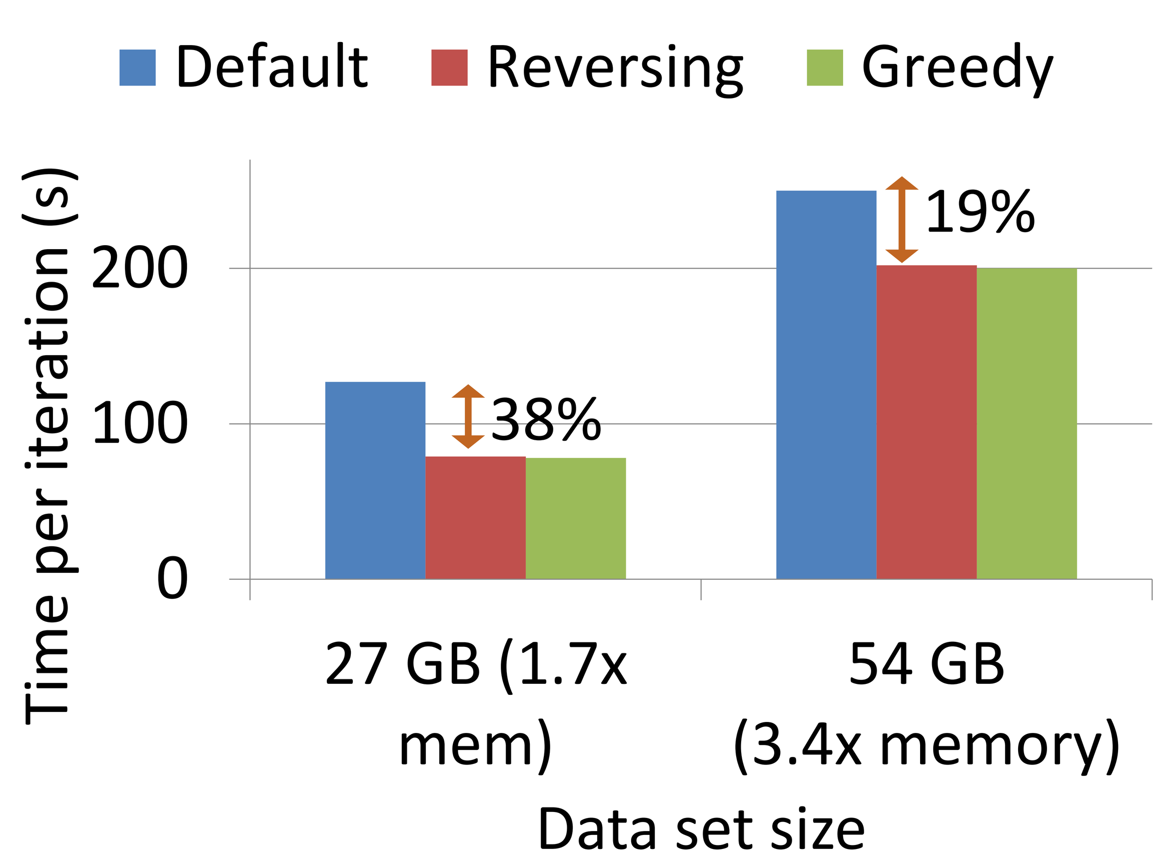


Greedy

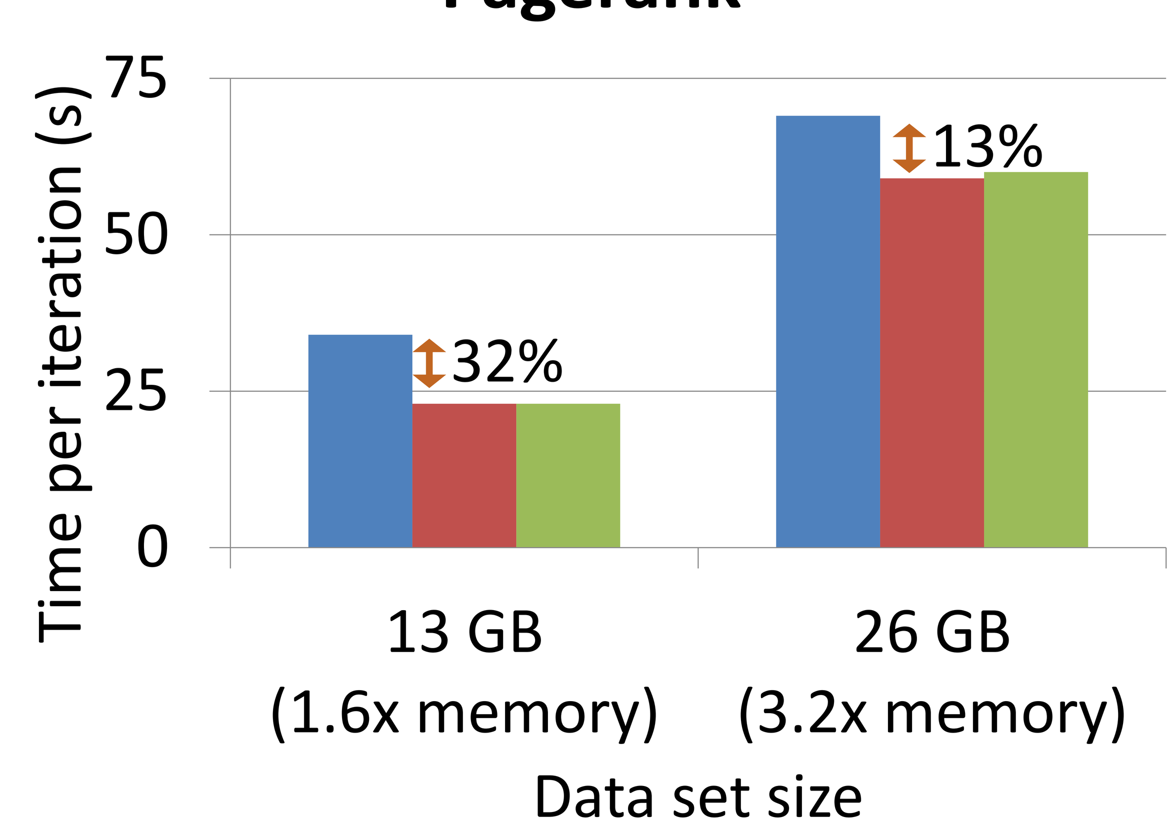
Keep track of in-memory data
Schedule task with least I/O

RESULTS I.

K-means



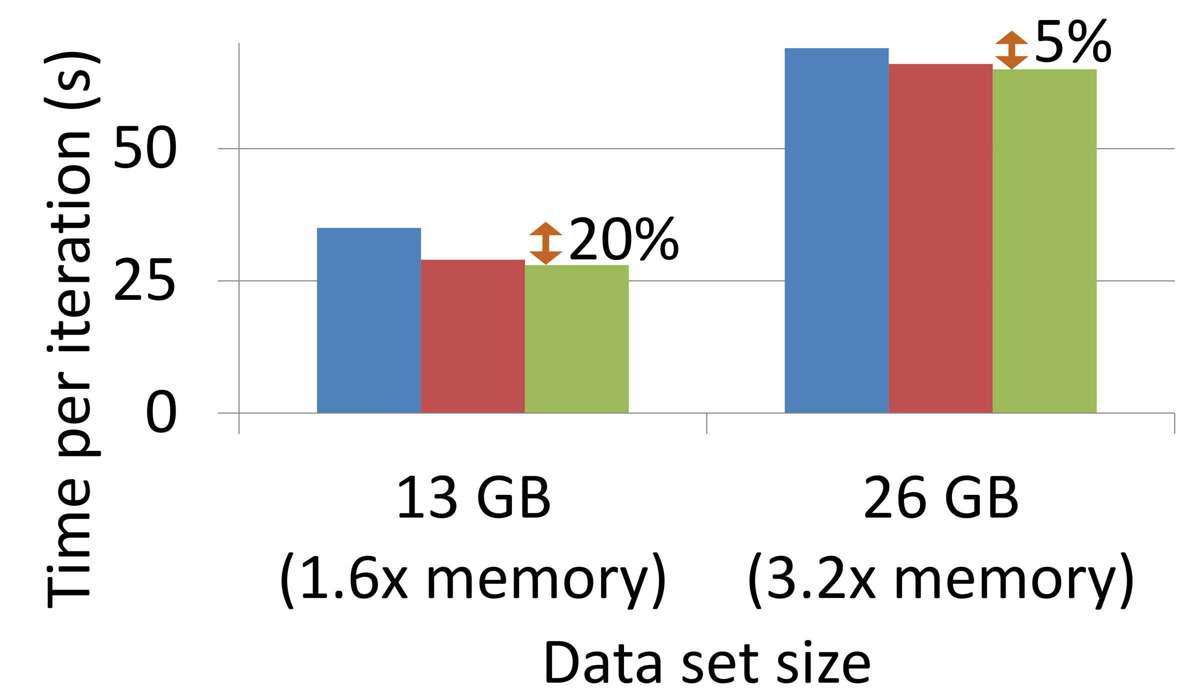
Pagerank



Simple computation structure
Reuse all memory contents

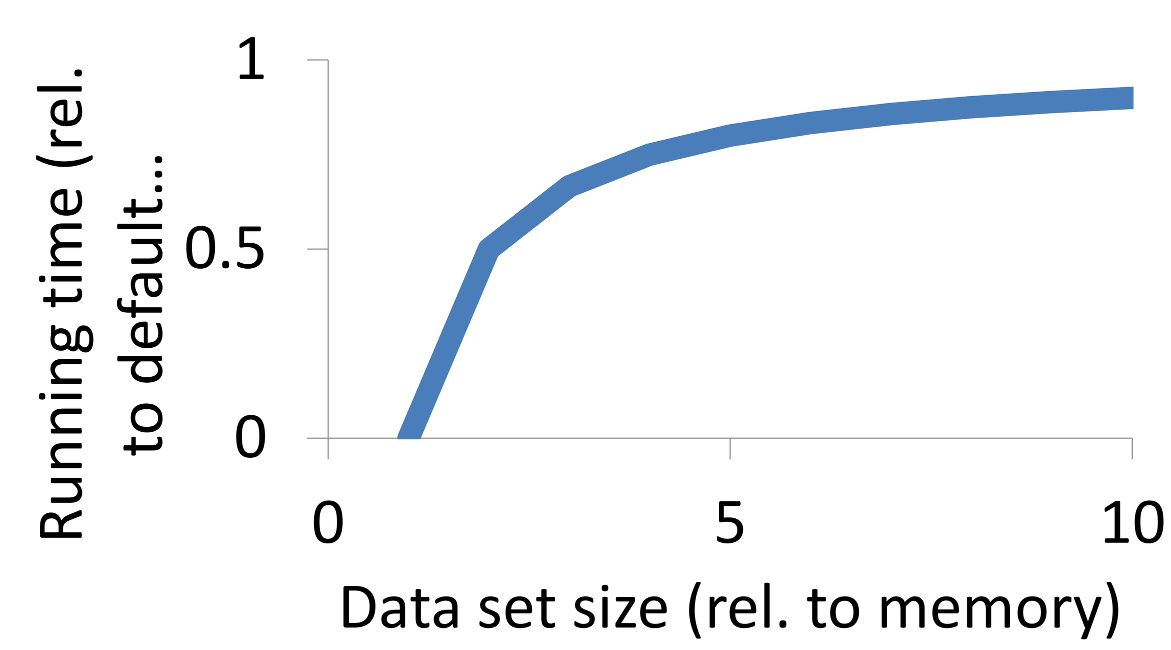
RESULTS II.

Conjugate gradient method



Multiple operations per iteration
Less reordering, less reuse

RUNNING TIME IMPROVEMENT SCALING



CONCLUSIONS

$$\text{speedup} \sim \frac{1}{\text{datasize}}$$

Computation reordering does not scale to big data sizes for streaming iterative algorithms

FUTURE WORK

DAG execution model
Data dependent access pattern
Asynchronous algorithms (conv. vs. I/O)
Caching policies

ACKNOWLEDGEMENTS

The authors gratefully acknowledge support from HP, NSF, DOE, DARPA, Qualcomm, Nvidia

Thanks to Shivaram Venkataraman, Indrajit Roy, Rob Schreiber and Alvin AuYoung for their contributions

REFERENCES

Presto: Distributed Machine Learning and Graph Processing with Sparse Matrices, EuroSys '13
http://www.hpl.hp.com/research/presto.htm
Blockus: http://blockus.cs.uchicago.edu