Auto-tuning MapReduce applications for multicores

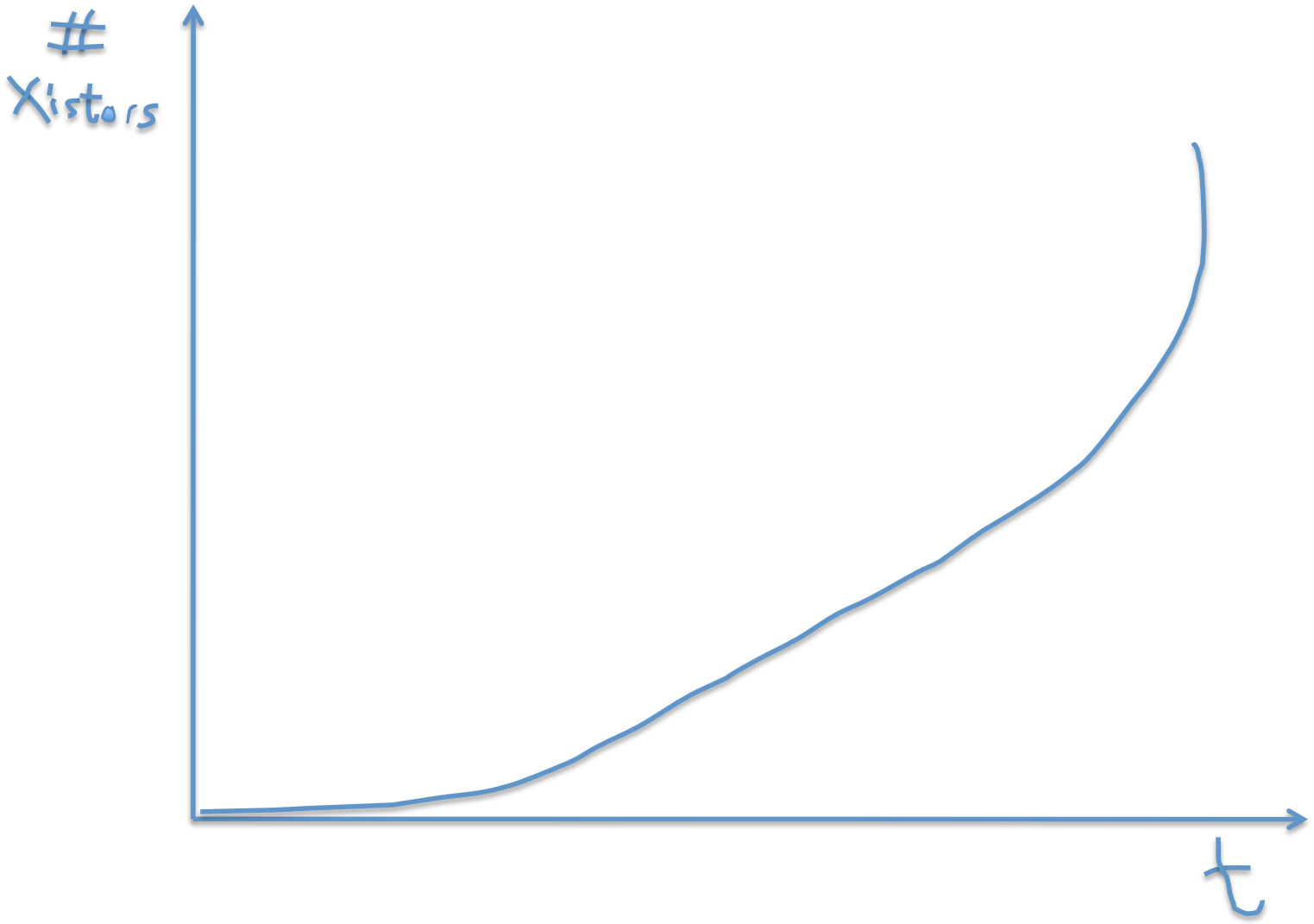
Jeremy Singer

http://www.dcs.gla.ac.uk/~jsinger

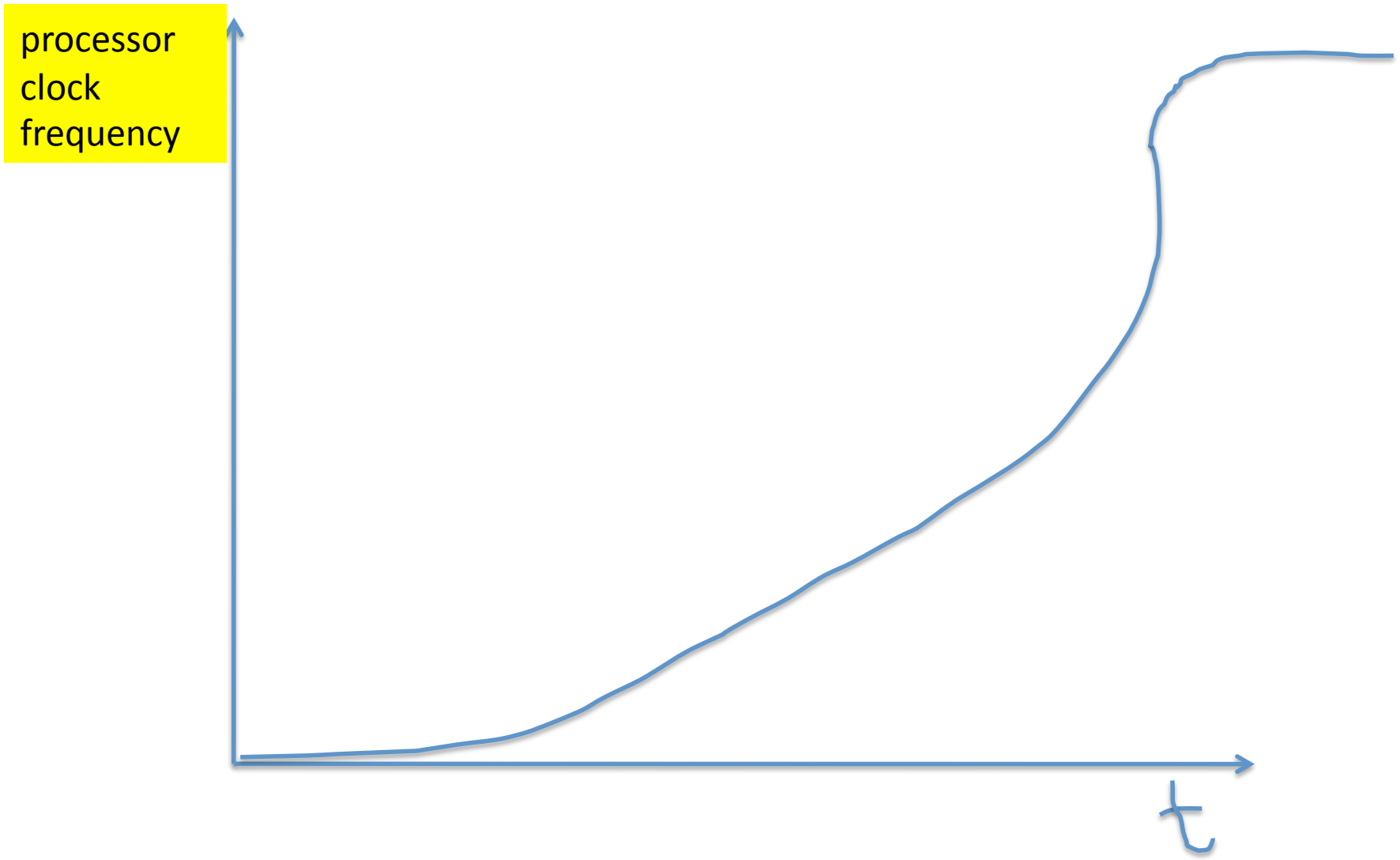
University of Glasgow | School of Computing Science

Multicores

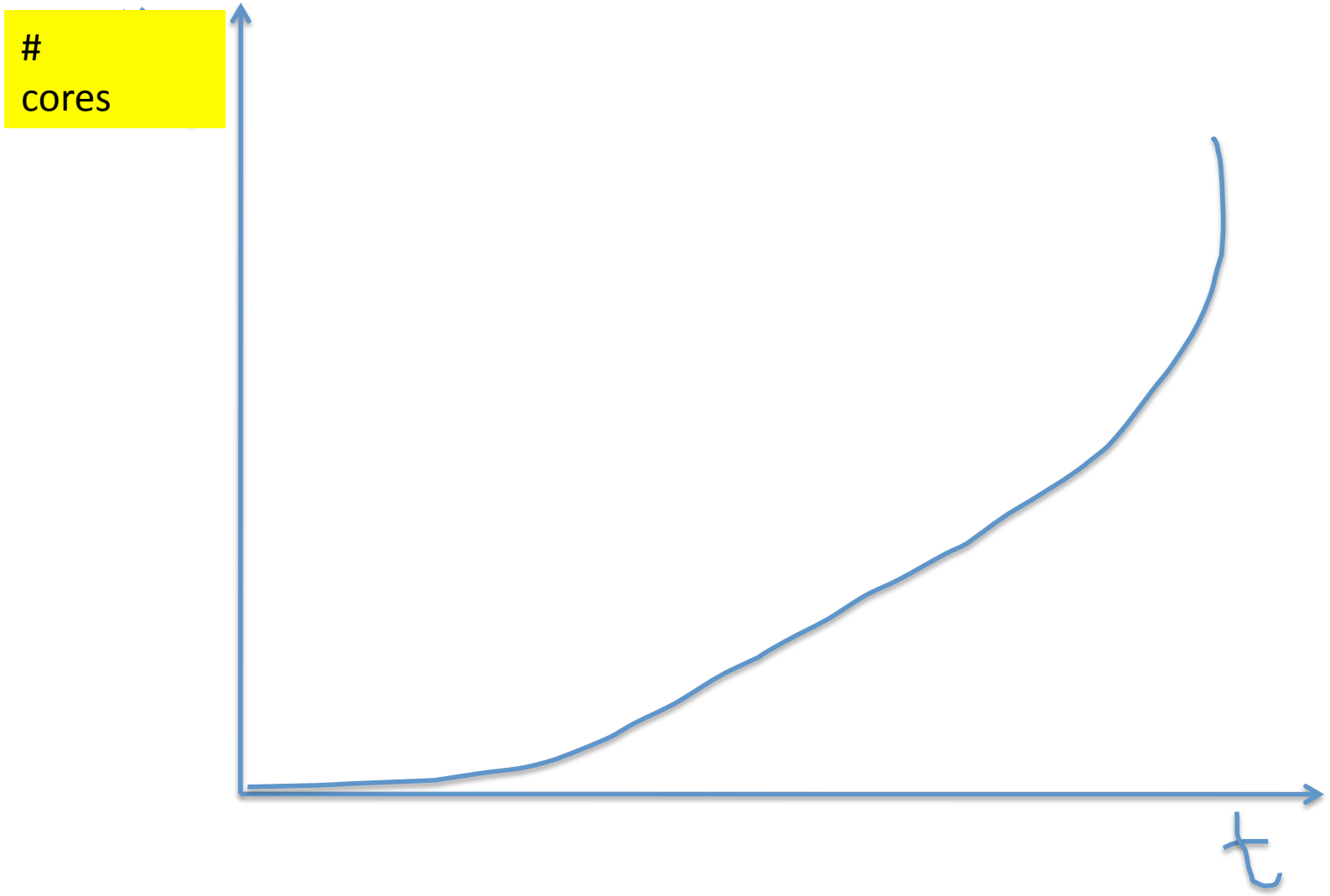
Moore's Law



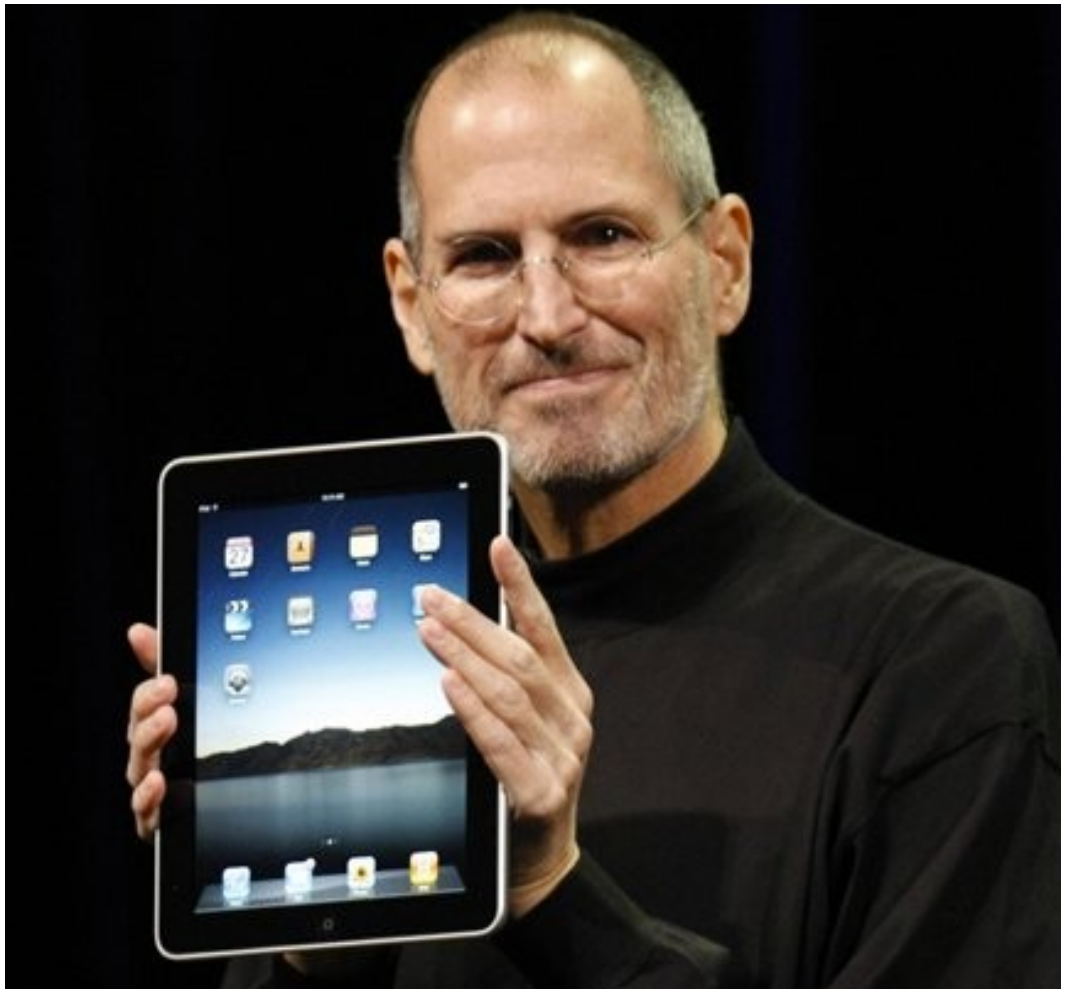
Moore's Law



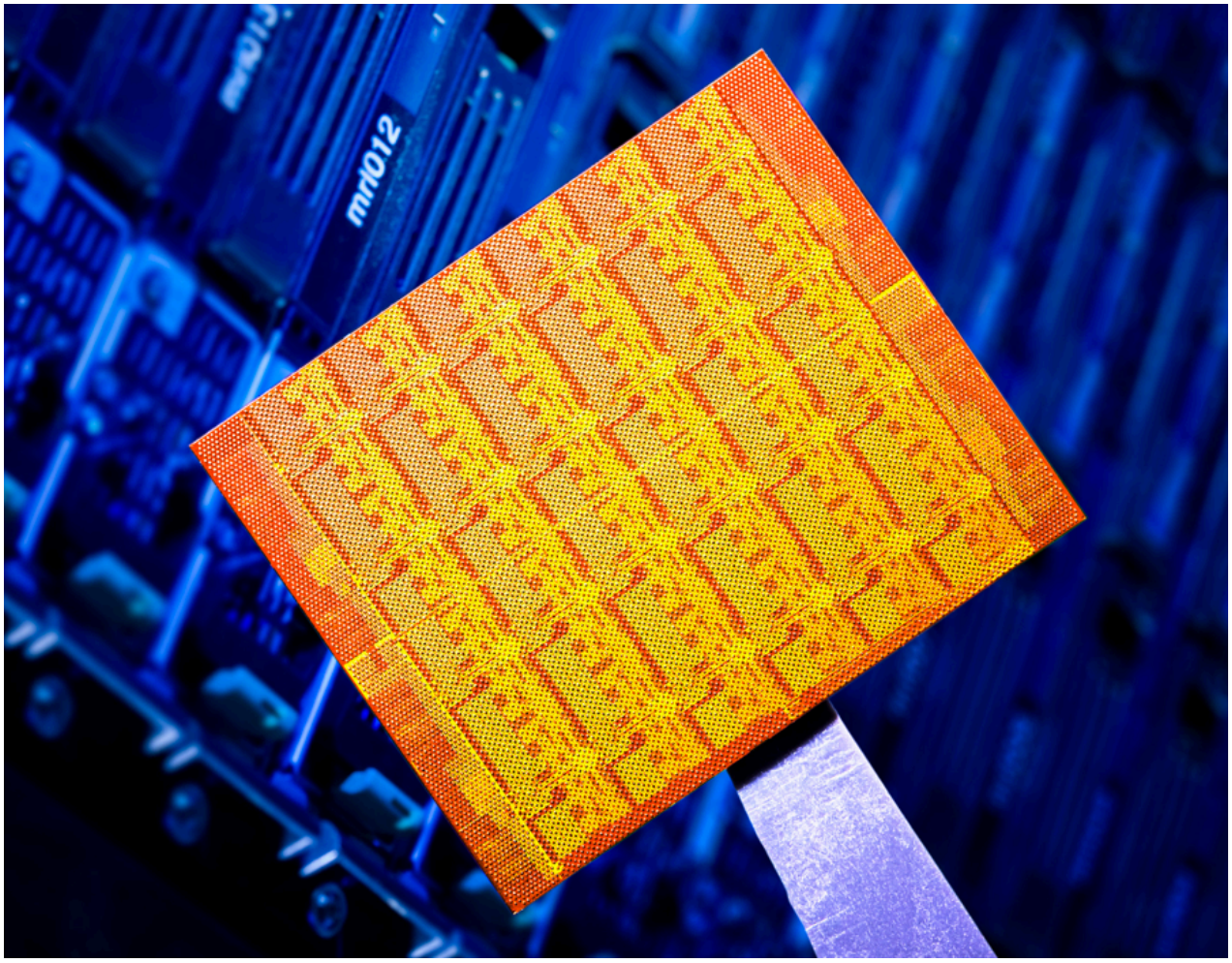
Moore's Law



Multicore is everywhere



Manycore is coming



ProgLangs Solutions for Multicore

MapReduce

A Parallel Design Pattern

- MapReduce
- Given a list of elements:
  - map a function over each element
  - Reduce elements with another function
- E.g. sum of squares: Σx^2
- Map the square fn over all elements
- Reduce with the plus function

HelloWorld in MapReduce

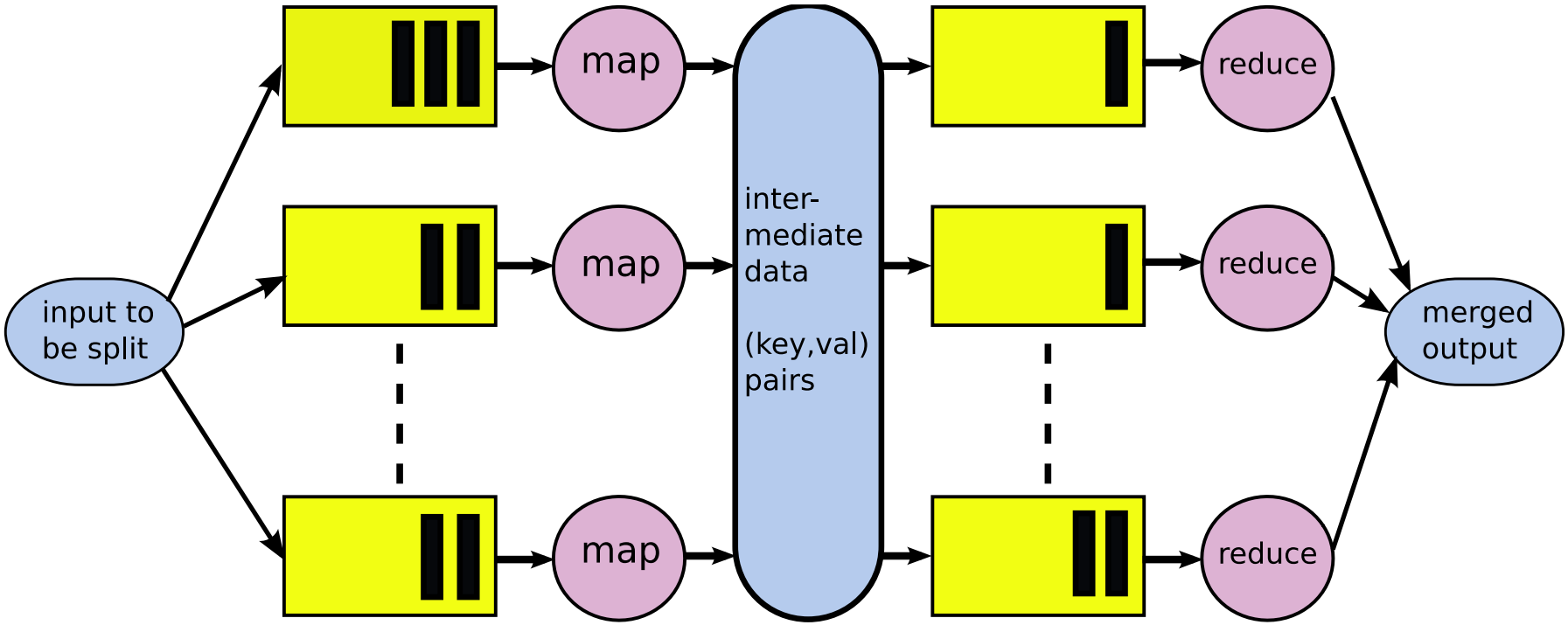
- Google give the wordcount example
- (whiteboard)

MRJ – mapreduce in Java

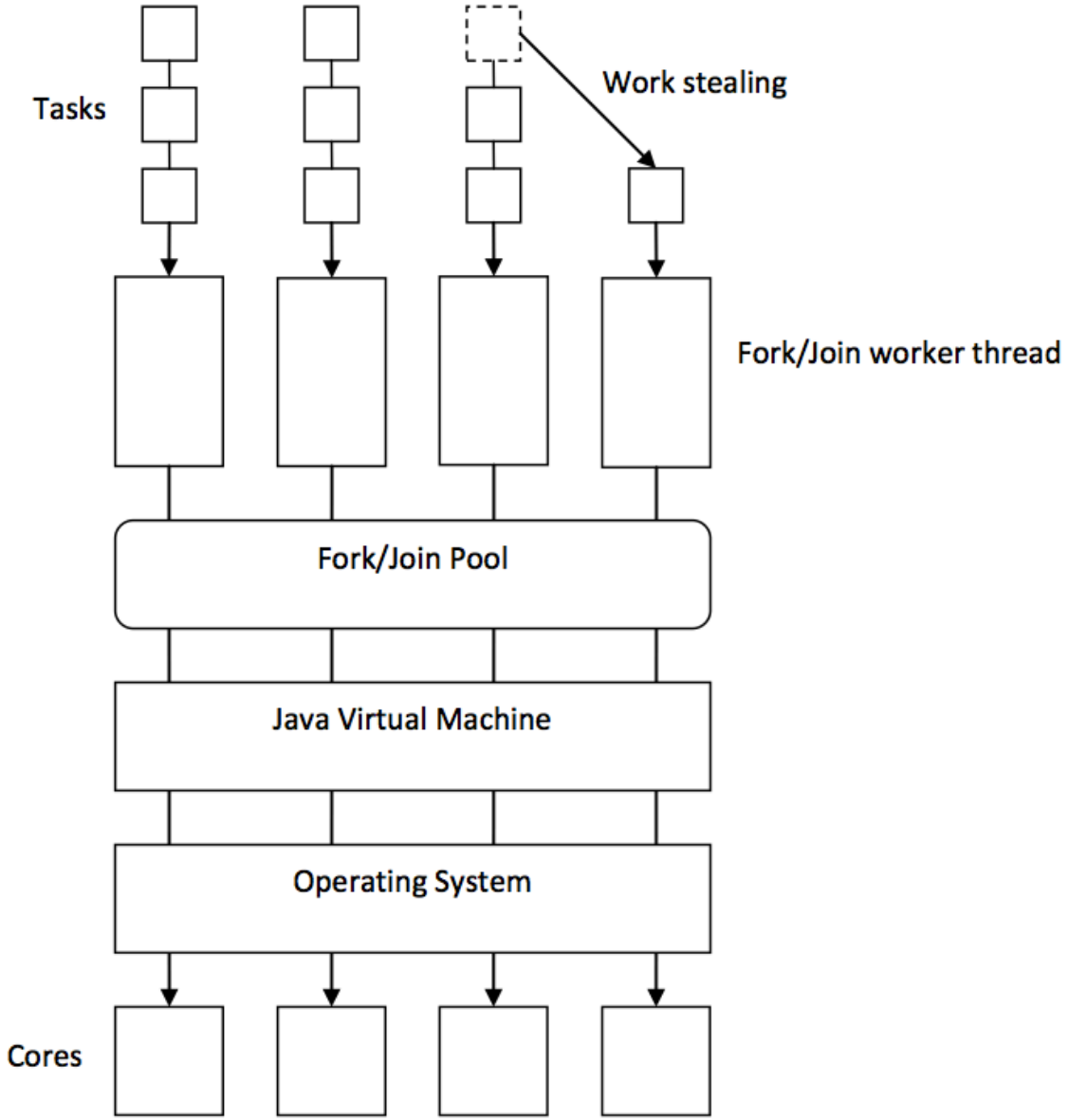
- Philosophically
  - Reducing complexity burden for programmer
  - Let the framework do the work

- Practically
  - Integration with Hadoop open-source project

MapReduce Schematic Diagram



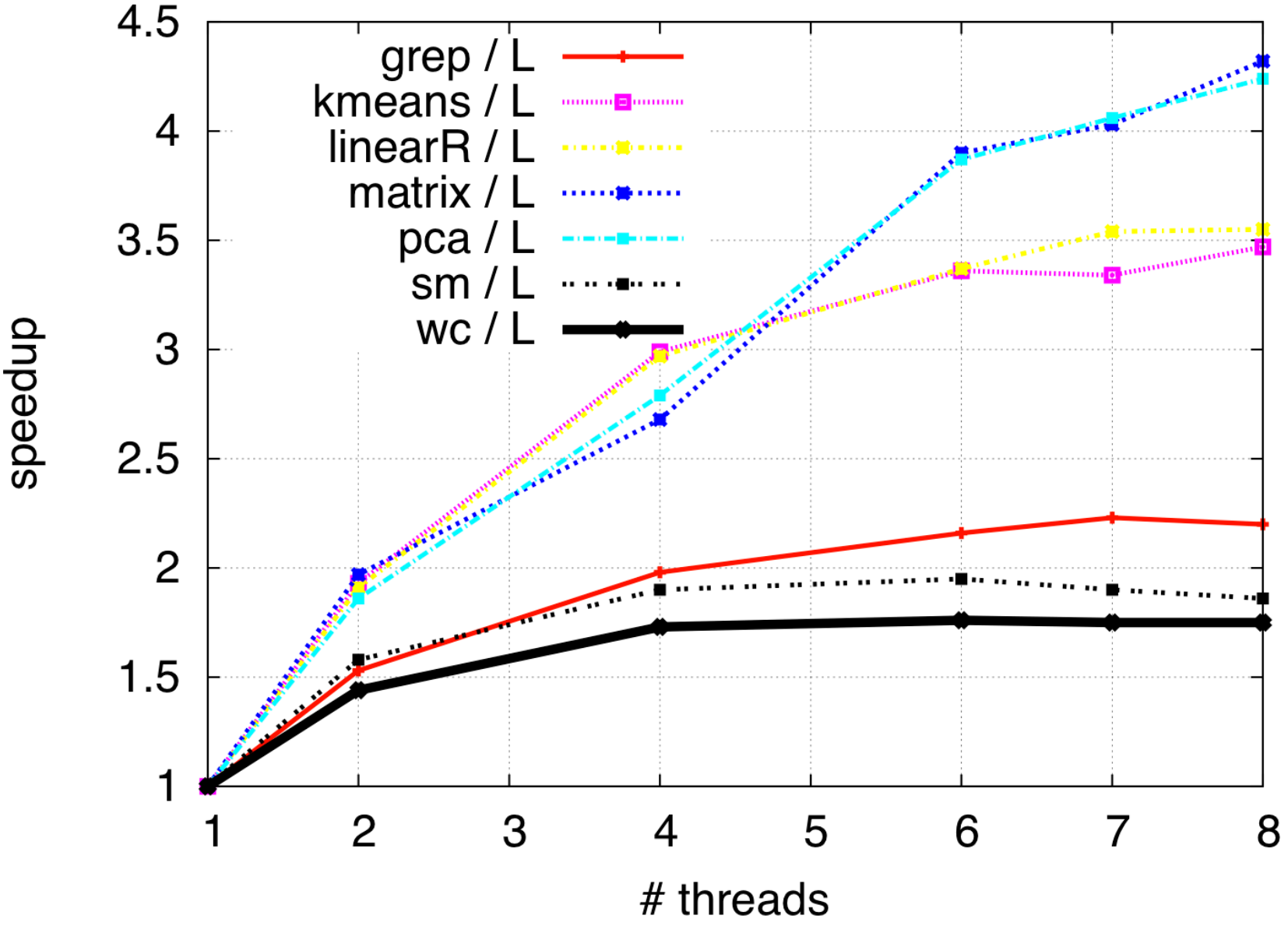
Fork/Join Library



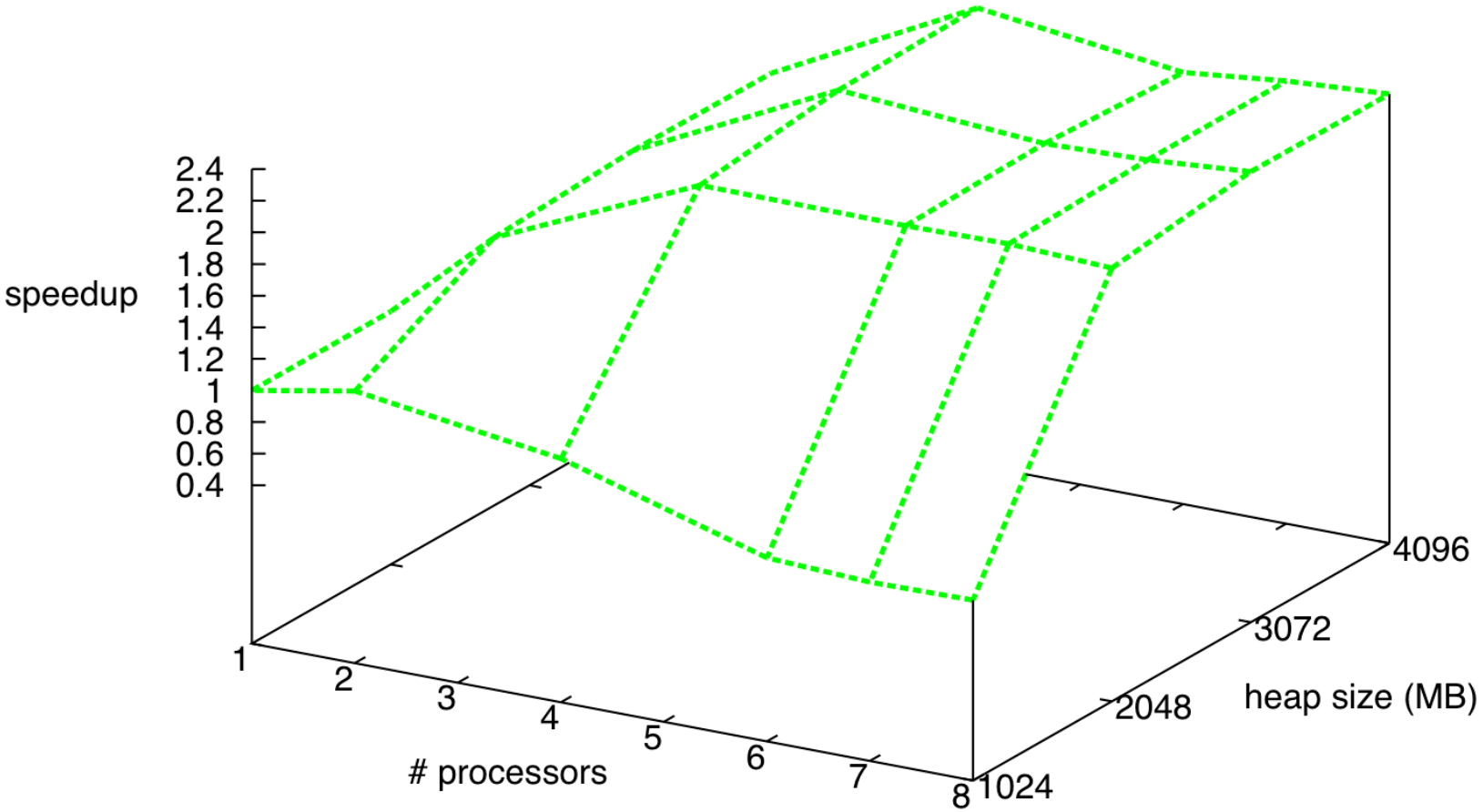
MRJ Benchmarks

benchmark	description
grep	find string occurrences in input text file
kmeans	group 3d points into clusters based on their Euclidean distance
linearR	compute best-fit line for input data file
matrix	dense integer matrix multiplication
pca	principal components analysis on an integer matrix
sm	search input text file for a word
wc	count instances of each unique word in input text file

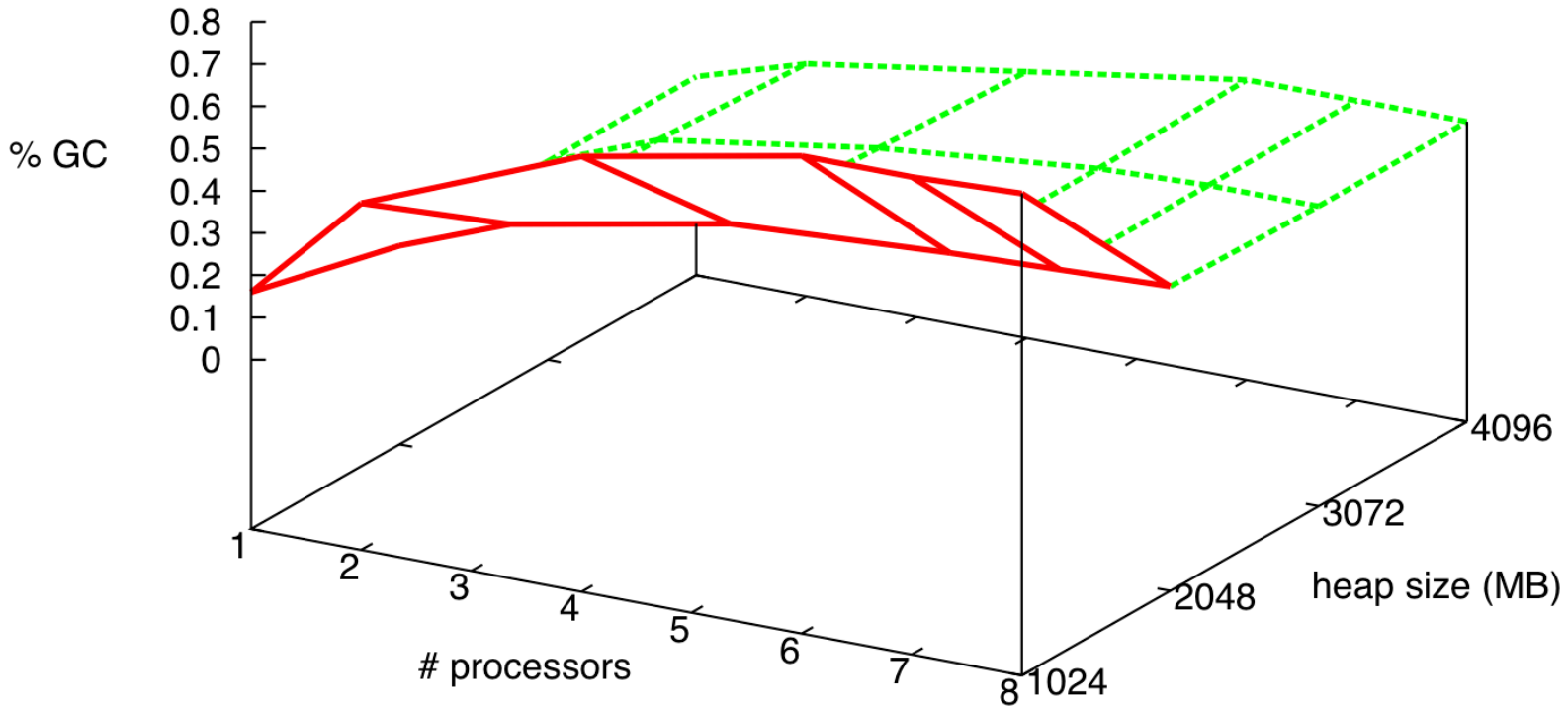
Benchmark Scalability



Grep benchmark anomaly



Grep GC overhead



Different GC policies affect performance

input	Small			Medium			Large		
heap/GB	1	2	4	1	2	4	1	2	4
#threads 8	P	S,P	S,P	C	P	P		C	P
7	P	S,P	S,P	C	P	P		C	P
6	P	S,P	S,P	C	P,C	P		C	P,C
4	P	S,P	S,P	S,P,C	P	P		S,P,C	P,C
2	P	S,P	S,P	P	P	P	C	P	P
1	S	S,P	S,P	S,P,C	S,P,C	S,P,C	C	S,P	S,P

Auto-Tuning

Auto-tuning to predict GC policy

- Predict algorithm and heap layout
- Given benchmark, input, and available memory

- Machine learning
  - Given some training examples
  - Generate predictor model
  - Test on previously unseen examples

Features for examples

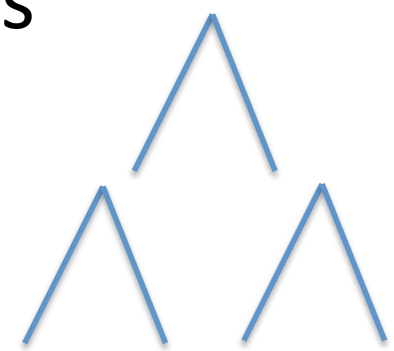
feature	*type*	*how collected*
heap size (MB)	integer	system parameter
# MRJ worker threads	integer	system parameter
# minor GCs (x2)	integer	trial execution
# major GCs (x2)	integer	trial execution
% GC time (x2)	real	trial execution
bytes allocated	integer	trial execution
% String alloc'd	real	trial execution
% int array alloc'd	real	trial execution

Prediction Outputs

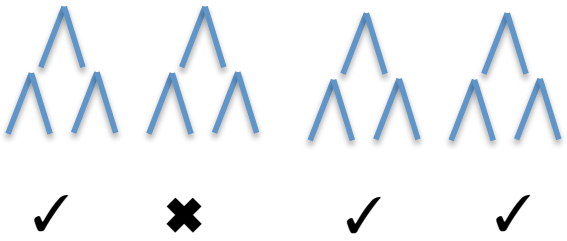
- Garbage collection algorithm:
  - serial, parallel, concurrent
- Heap Layout:
  - Young/old – ratio 1:2, 1:8

Machine Learning Technique

- Decision Trees

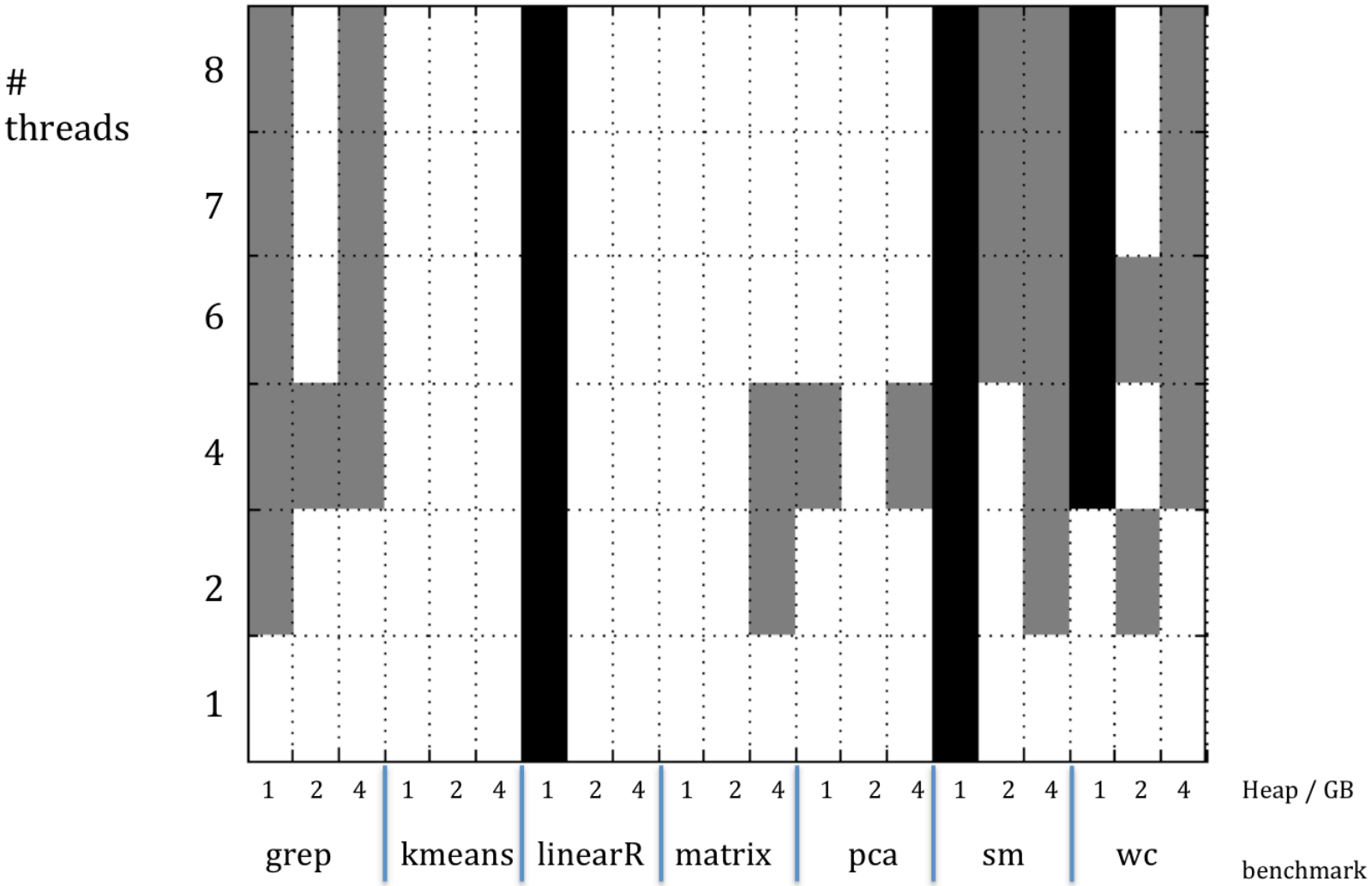


- Random Forest

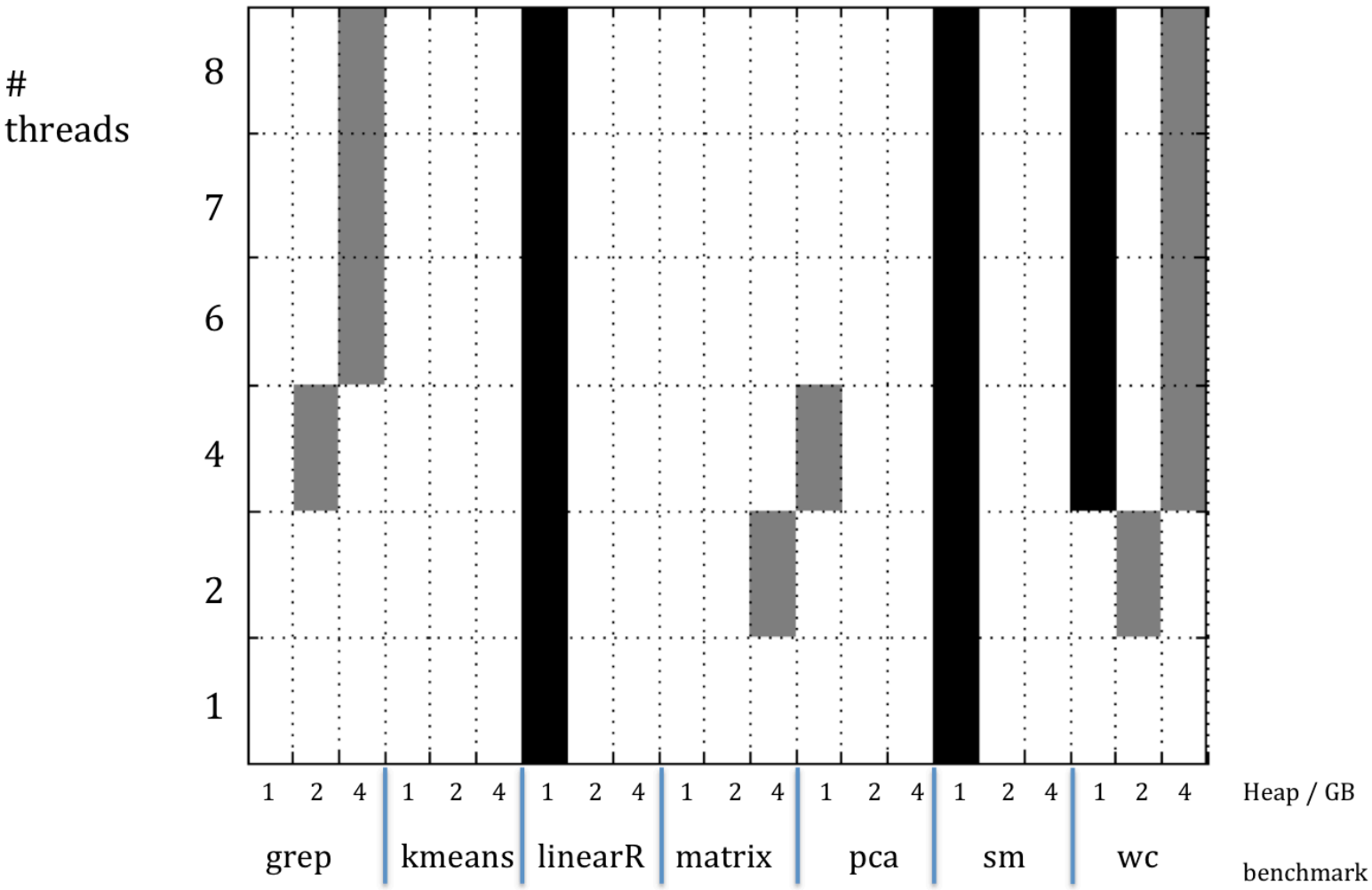


Cascade of Classifiers

Close to optimal?



No worse than default?



Conclusions

- Auto-tuning is beneficial for complex software systems
- Major subtleties:
  - What to tune
  - How to describe example problems
  - Constraining prediction
- Paper at ISMM 2011

http://www.dcs.gla.ac.uk/~jsinger/ismm11.pdf

Garbage Collection Auto-Tuning for Java MapReduce on Multi-Cores

Jeremy Singer
University of Glasgow
jeremy.singer@glasgow.ac.uk

George Kovoor *
kovoor.george@gmail.com

Gavin Brown Mikel Luján
University of Manchester
firstname.lastname@manchester.ac.uk

Abstract

MapReduce has been widely accepted as a simple programming pattern that can form the basis for efficient, large-scale, distributed data processing. The success of the MapReduce pattern has led to a variety of implementations for different computational scenarios. In this paper we present *MRJ*, a MapReduce Java framework for multi-core architectures. We evaluate its scalability on a four-core, hyperthreaded Intel Core i7 processor, using a set of standard MapReduce benchmarks. We investigate the significant impact that Java runtime garbage collection has on the performance and scal-

[3]. The pattern still has its detractors [15]. However, it has been demonstrated to give effective parallelism for important parts of the computer applications spectrum e.g. machine learning [10], databases [37], eScience [16].

Our objective is to investigate the MapReduce pattern in the context of multi-core architectures, rather than within compute clusters, as commonly used by Amazon, Facebook, Google and Yahoo, amongst others.

1.1 Motivation for Multi-Core MapReduce