Parallel PageRank Computation using MPI

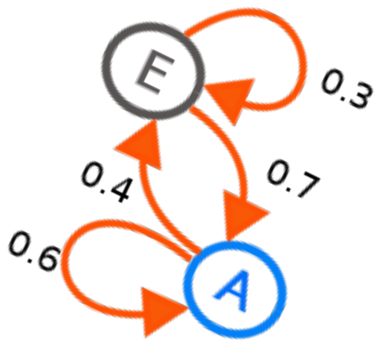
CSE 633 Parallel Algorithms (Fall 2012)

Xiaoyi (Eric) Li
Email: xiaoyili@buffalo.edu

Outline

- Markov Chains
- PageRank Computation
- Parallel Algorithm
- Message Passing Analysis
- Experiments and result

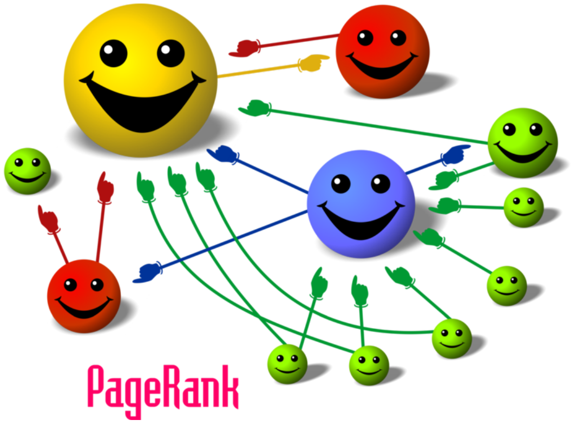
Markov Chains



- Markov Chain:
  - A Markov chain is a discrete-time stochastic process consisting of N states.
- Transition Probability Matrix:
  - A Markov chain is characterized by an N*N transition probability matrix P.
  - Each entry is in the interval [0,1].
  - A matrix with non-negative entries that satisfies $\forall i, \sum_{j=1}^{N} Pij = 1$
  - Chain is acyclic
  - There is a unique steady-state probability vector π.
    - η(i,t) is the number of visits to state i in t steps.
    - π(i)>0 is the steady-state probability for state i.

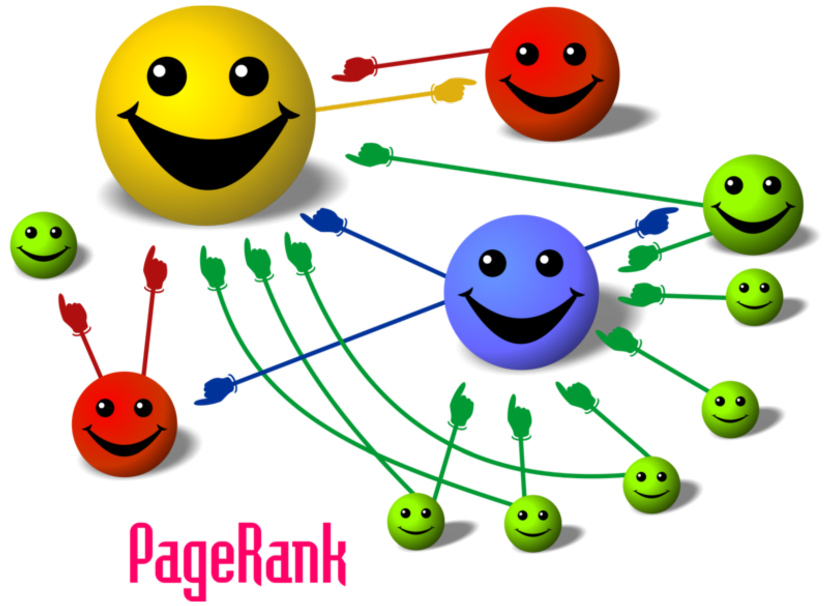
$$\lim_{t->\infty} \frac{\eta(i,t)}{t} = \pi(i)$$

PageRank Computation



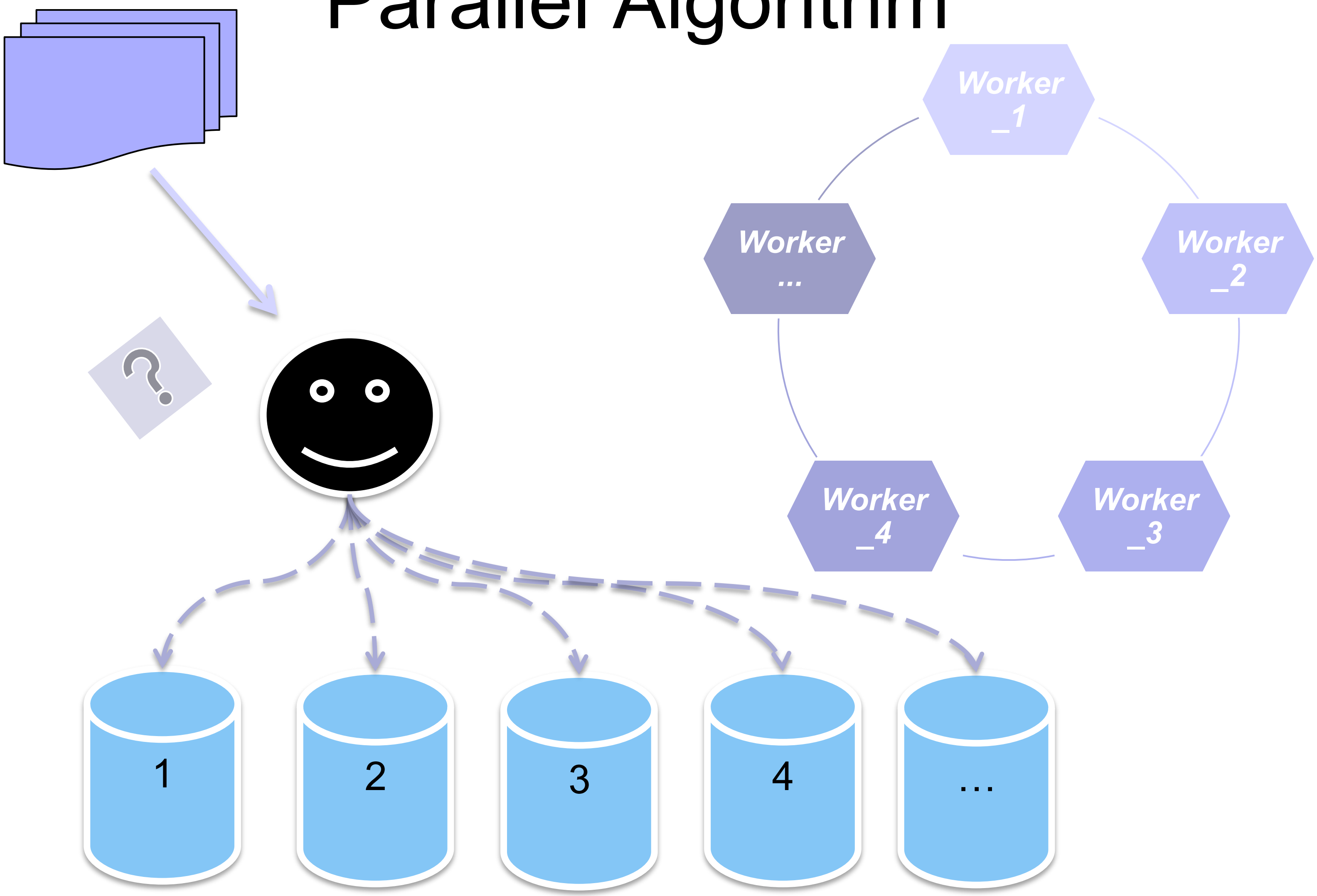
- Target
  - Solve the steady-state probability vector π, which is the PageRank of the corresponding Web page.
- Method
  - Iteration.
  - Given an initial probability distribution vector x0
  - x0*P = x1, x1*P = x2 … Until the probability distribution converges. (Variation in the computed values are below some predetermined threshold.)

Practical PageRank Calculation



$$PR(p_i) = \frac{1-d}{N} + d \sum_{p_j \in M(p_i)} \frac{PR(p_j)}{L(p_j)}$$

Parallel Algorithm



------------------------------------ Initialization --

Worker

- Read bucket, construct local graph and send two index -- *node to update* & *node required* to master

Master

- Received individual index, initialize send & receive buff for each worker.
- Initialize global weights, send weights[index_i] to workers_i

------------------------------------ *Begin iteration* --

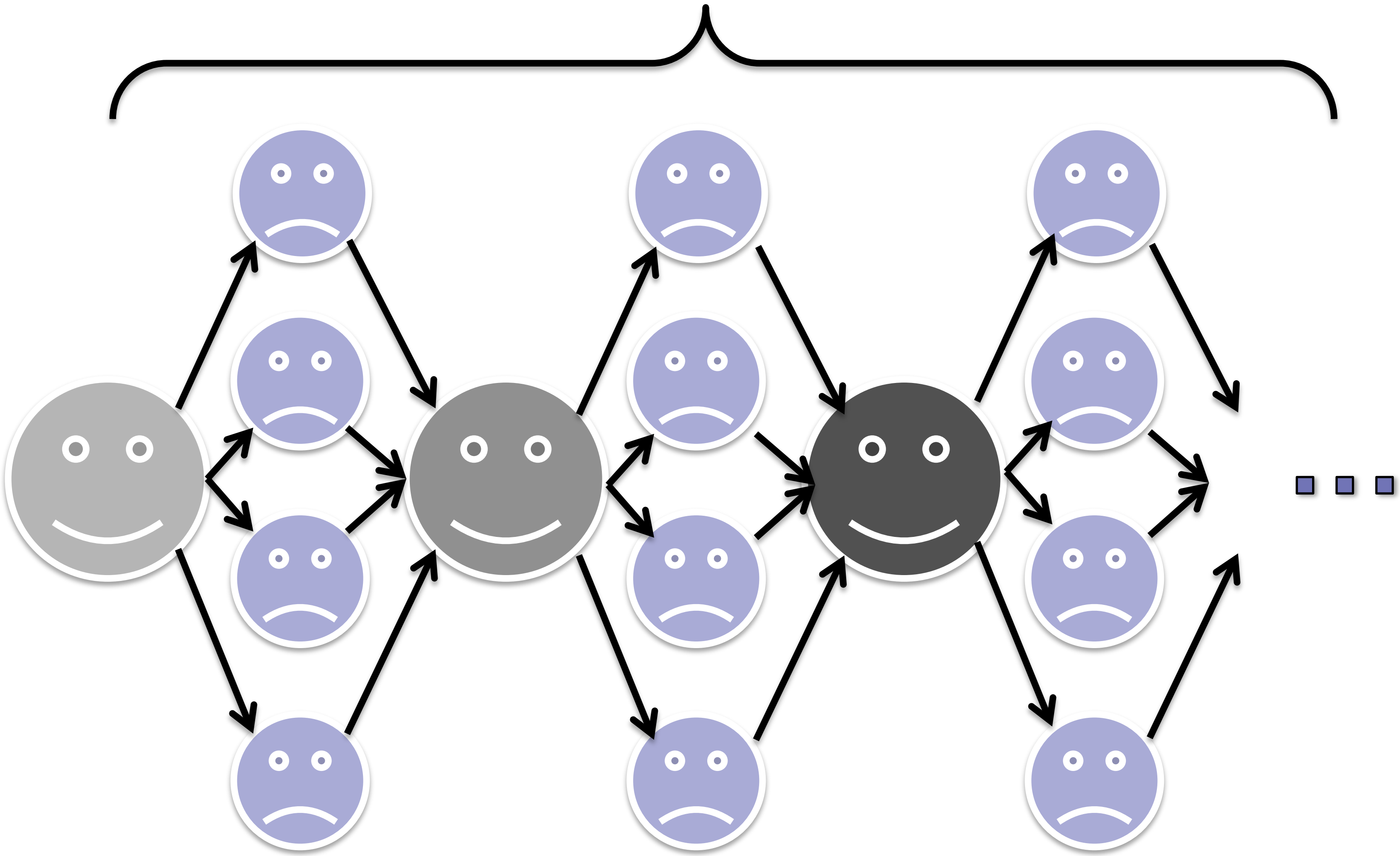
Worker

- Update local graph using received weight. Calculate PageRank once.
- Send the updated score back to master.

Master

- Gather individual updates form workers, update the global weight determined by index_i
- Check convergence
- If not, send global weights to workers
- If yes.. Send stop signal and do house keeping

Total number of iterations



Message Passing Analysis

***Without* weight index**

- Each worker send & receive global weight from master:

1M web-nodes, 64 workers:
- 2 * 8 bytes * 1M = 16MB
- 16 * 64 = 1024MB = 1GB
- Total = #iteration * 1GB

***With* weight index**

- Each worker send & receive global weight from master:

1M web-nodes, 64 workers:
- Send: 8 * 1M / #workers ≈ 0.128MB
- Rec: 8 * 1M / (small fraction, e.g #nodes/8) ≈ 1MB
- 1.128 * 64 ≈ 72MB
- Total = #iteration * 72MB

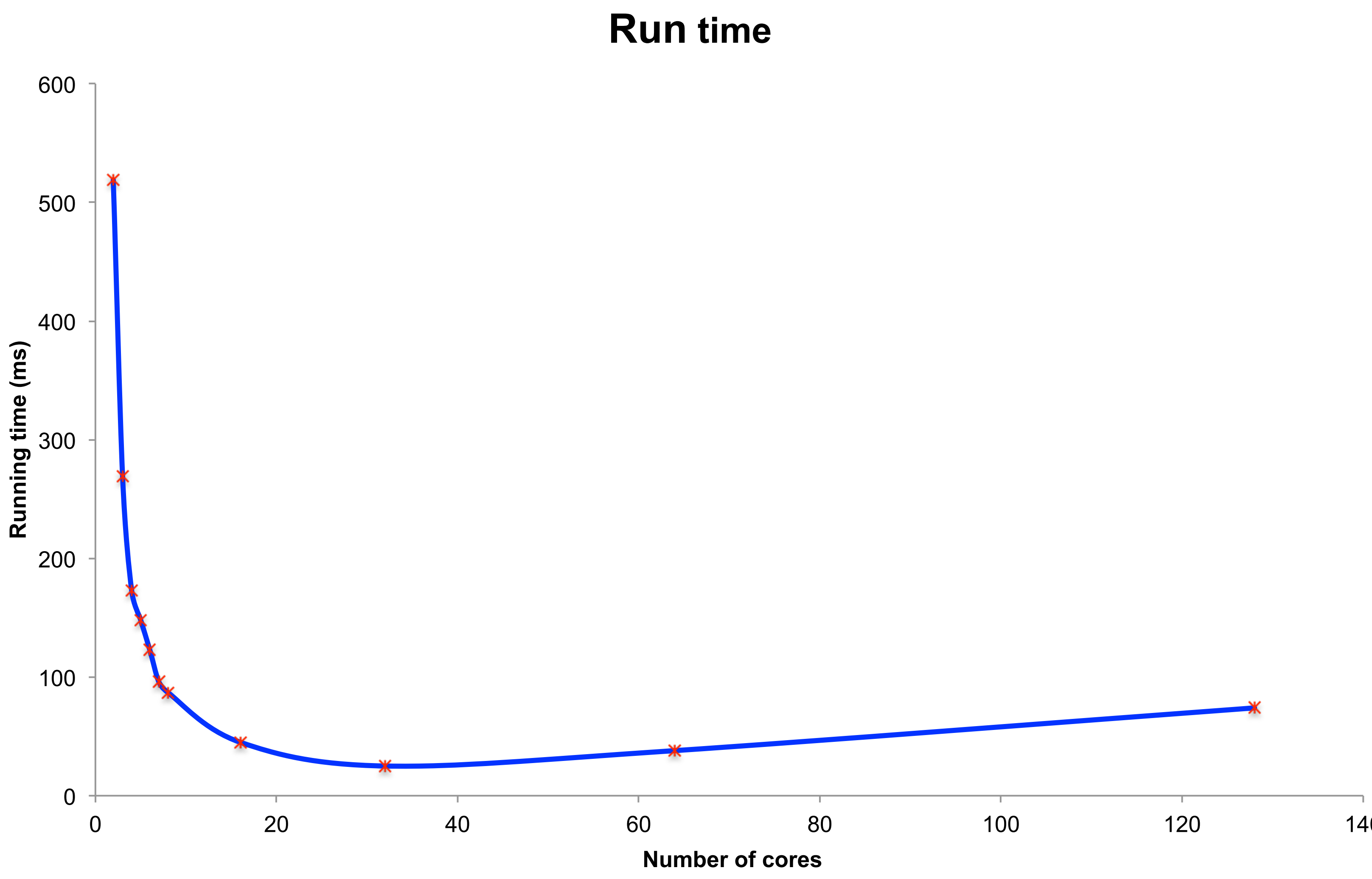
Experiments

- Data: wiki-votes (67035 | 1025563)
- #nodes = 32, IB2, ppn=4

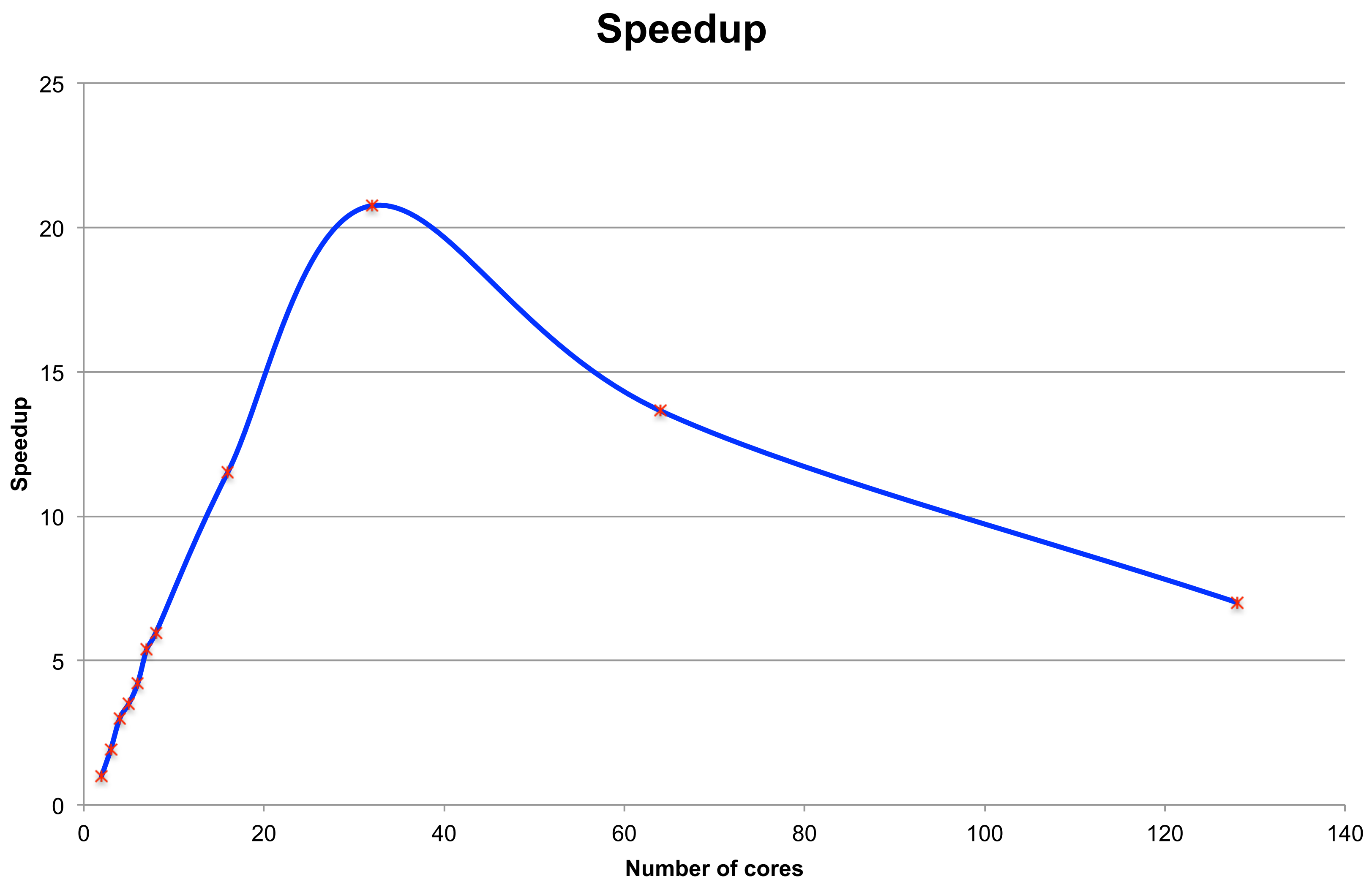
Results

#cores	run time (ms)	speed up	efficiency
2	519	1	1
3	269	1.92936803	0.96468401
4	173	3	1
5	148	3.50675676	0.87668919
6	123	4.2195122	0.84390244
7	96	5.40625	0.90104167
8	87	5.96551724	0.85221675
16	45	11.5333333	0.76888889
32	25	20.76	0.66967742
64	38	13.6578947	0.21679198
128	74	7.01351351	0.05522452

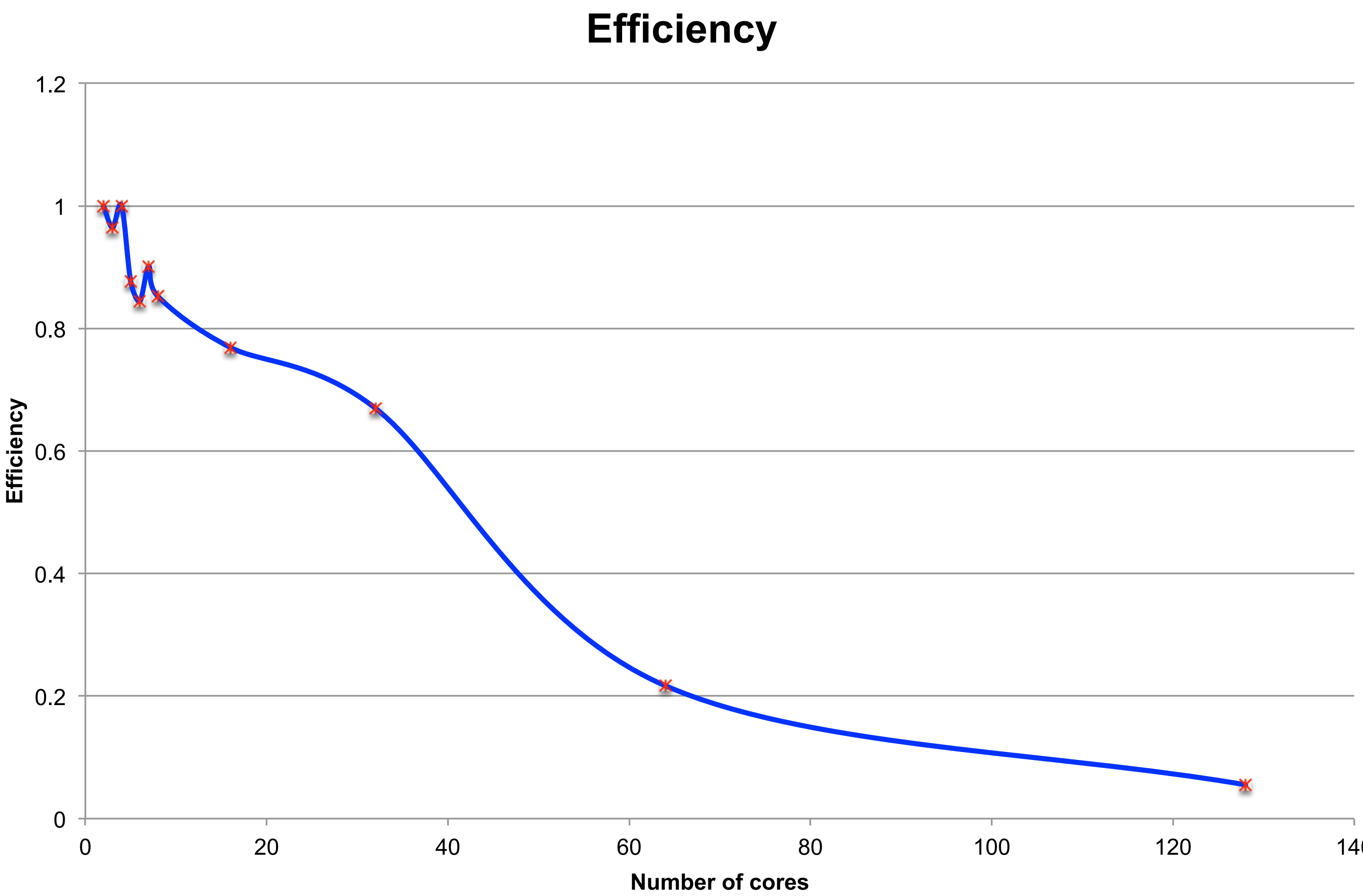
Results


Run time
Running time (ms)
600
500
400
300
200
100
0
0
20
40
60
80
100
120
140
Number of cores

Results


Speedup
25
20
15
10
5
0
Speedup
0
20
40
60
80
100
120
140
Number of cores

Results



Efficiency
1.2
1
0.8
0.6
0.4
0.2
0
Efficiency
0
20
40
60
80
100
120
140
Number of cores

- Questions?