CS301 Project Report
Project Title: Sample Sort

Team members:

(1) Aayush Kapadia (201401407)

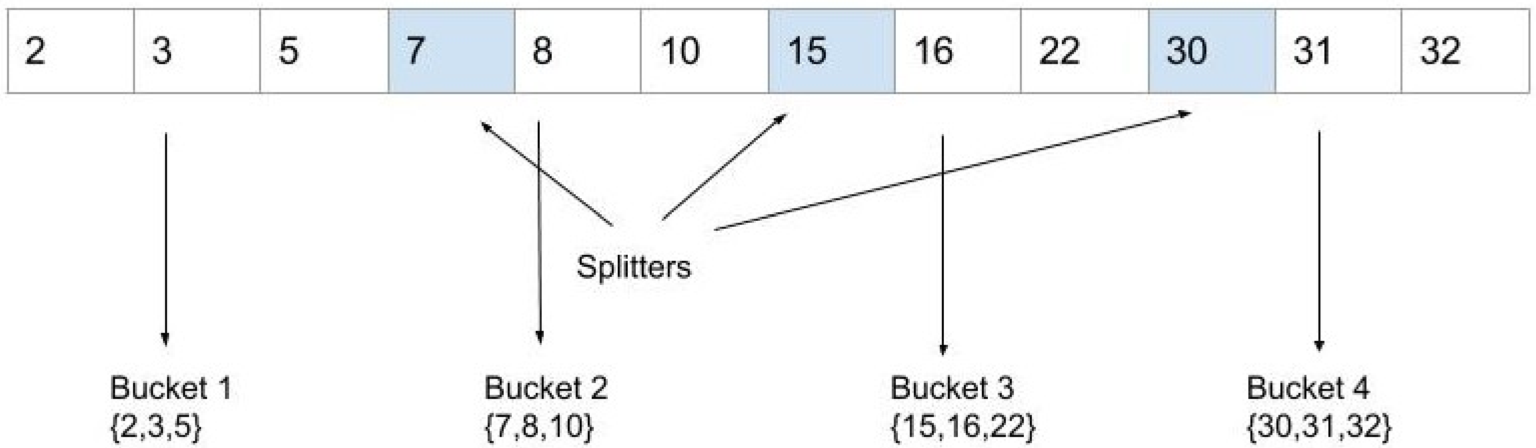
(2) Mit Naria (201401448)

Introduction

- Sample sort is a parallel sorting algorithm.
- Sample sort is a comparison based sorting algorithm.
- Sample sort follows PSRS algorithm design strategy. [Parallel Sorting by Regular Sampling] [explained in upcoming slides.]

Definition of Splitters

- Splitters: Splitters are the elements which divides sorted array into small blocs[chunks or buckets].
- Example:
- Here, a sorted array of 12 elements is splitted in 4 blocs [chunks or buckets] by 3 splitters.



PSRS algorithm design strategy

- This algorithm design consists of 5 stages:

1. Local sorting
2. Generation of local splitters by using a sampling algorithm
3. Generation of global splitters
4. Distributing buckets to nodes
5. Local sorting of a bucket

Sample sort algorithm

1. Local sorting (Quicksort) .
2. Generation of local splitters. (each node selects (p-1) splitters) .
3. Generation of global splitters. (root node gathers p*(p-1) splitters and sorts them using quicksort and selects (p-1) splitters from total p*(p-1) splitters. They are transmitted to each node.)
4. Distributing buckets to nodes. (buckets are prepared by using new (p-1) splitters and send to respective node. Bucket[i] goes node with rank=i. After that each node merges the buckets received.)
5. Local sorting of a bucket (Quicksort).

Algorithm example

Theoretical analysis of parallel algorithm

Note: We have assumed that communication latency is O(log(p)).

- Step 1: O((n/p)*log(n/p)) [Sorting]
- Step 2: O(p) [Choosing local splitters]
- Step 3: O(p*log(p)) [Collecting local splitters] + O((p^2 * log(p))) [Sorting local splitters] + O(p) [Choosing global splitters] + O(p*log(p)) [Distributing global splitters]
- Step 4: O(n/p) [Buckets generation] + O((n/p)*log(p)) [Buckets transmission] + O((n/p)*log(p)) [Merging buckets]
- Step 5: O((n/p)*log(n/p)) [Sorting]

Theoretical analysis of parallel algorithm

- Computational complexity: O((n/p)*log(n/p)) + O((p^2 * log(p))) + O((n/p)*log(p))
- Since n>>p , log(n/p) ~ log(n) , Overall computational complexity:

  O((n/p)*(log(n)+log(p))) .
- Communication complexity: O(p*log(p)) + O((n/p)*log(p))
- Since (n/p) >> log(p) and (n/p)>>p, Overall communication complexity:

  O(n/p) .
- Total bound: O((n/p)*(log(n)+log(p))) + O(n/p) .
- We compare this parallel algorithm to serial randomized quicksort algorithm. [most widely used comparison based sorting algorithm] .

Theoretical analysis of serial algorithm and expected speedup

- Total bound (serial code) = O(n*log(n))
- Theoretical speedup =

  O(n*log(n)) /{O((n/p)*(log(n)+log(p))) + O(n/p) } .
- Expected speedup (predication based approach):
- To calculate approximate speedup, we simply say that parallel algorithm actually sorts twice.[Local sorting]. Thus, on p processors, calculation is p times faster but 2 times slower. Thus, predicated speedup = p/2 . [This speedup is just based on observation.]

Scalability

- From theoretical analysis of parallel algorithm:

  $$k(n,p) = O(p*\log(p)) + O((n/p)*\log(p))$$

- We can ignore first term in k(n,p).
- Thus, $k(n,p) = O((n/p)*\log(p))$.
- Therefore $To(n,p) = O(n*\log(p))$.
- $T(n,1) = O(n*\log(n))$.
- Isoefficiency function:

  $$T(n,1) \quad >= C * To(n,p)$$

  $$n*\log(n) >= C * n * \log(p)$$

  $$\log(n) \quad >= C * \log(p)$$

  $$n \quad >= p^C \qquad ....\text{equation}(1)$$

Scalability

- Scalability function:
- Since M(n) = n , Scalability function M(f(p))/p = $p^{(C-1)}$.
- If C=1 then system will perfectly scalable.

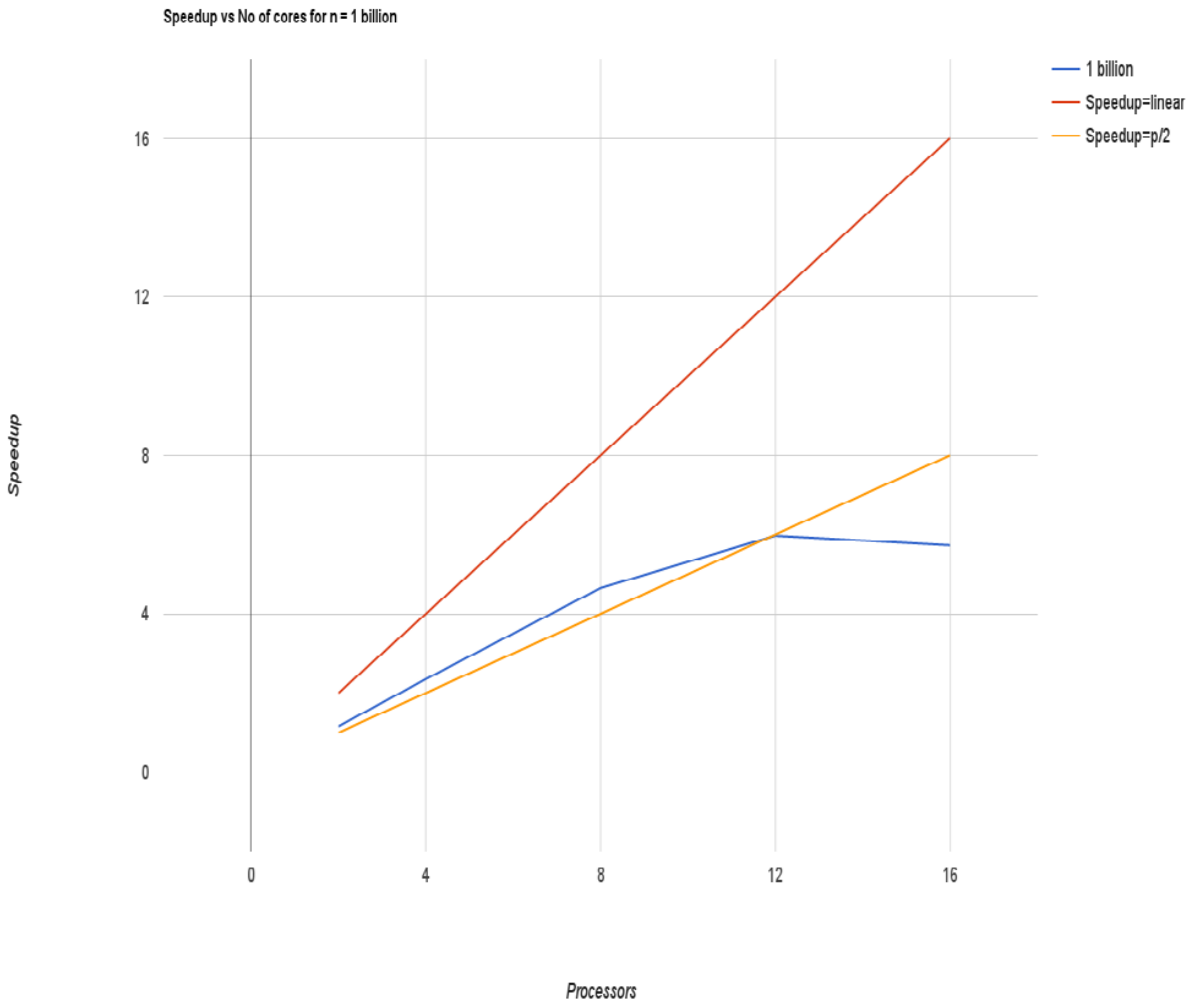
  C = (1 - efficiency)/efficiency .

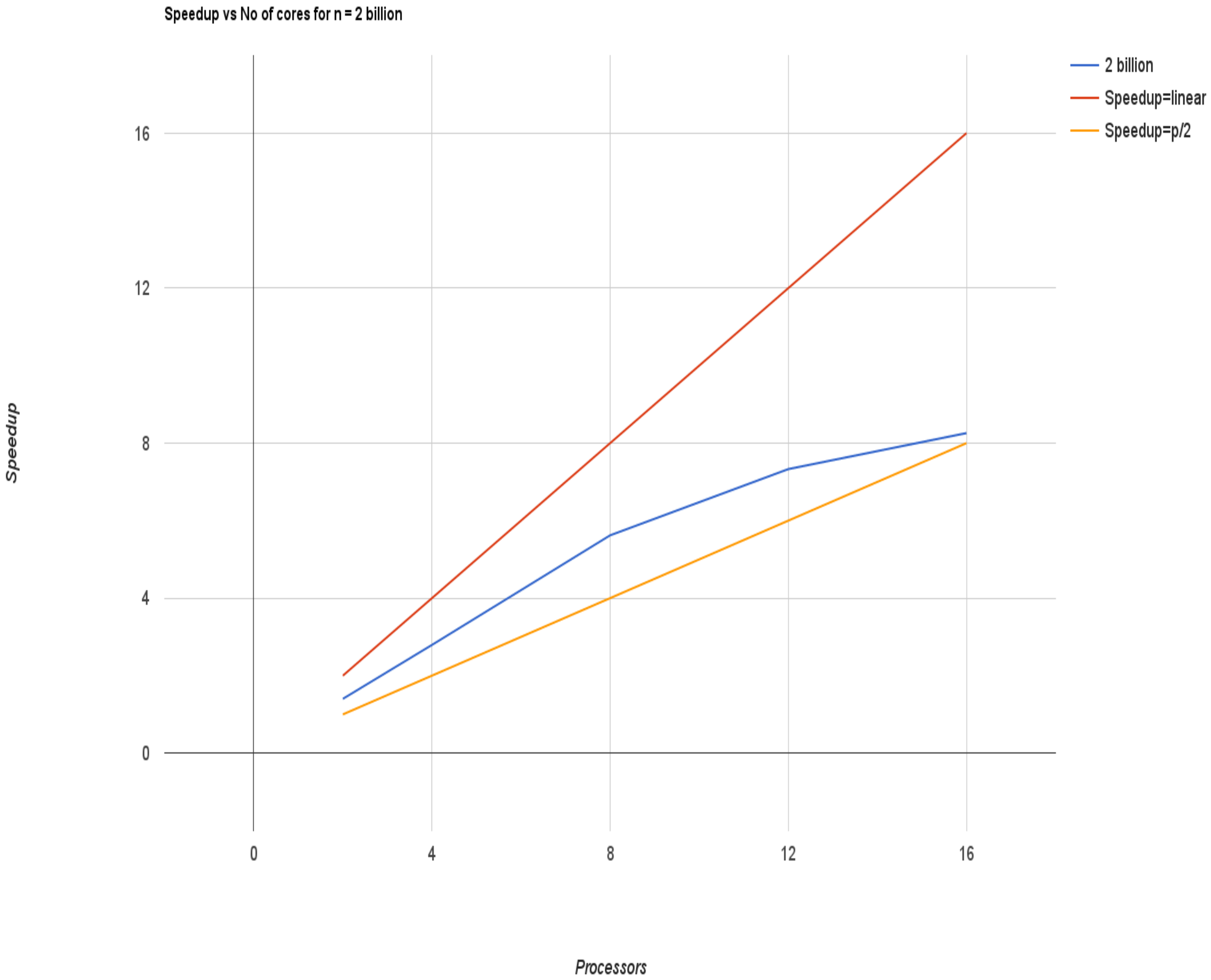
  Suppose x = maximum allowed efficiency which results into perfectly scalable system.

  Then C = (1-x)/x = 1 (because system is perfectly scalable.)

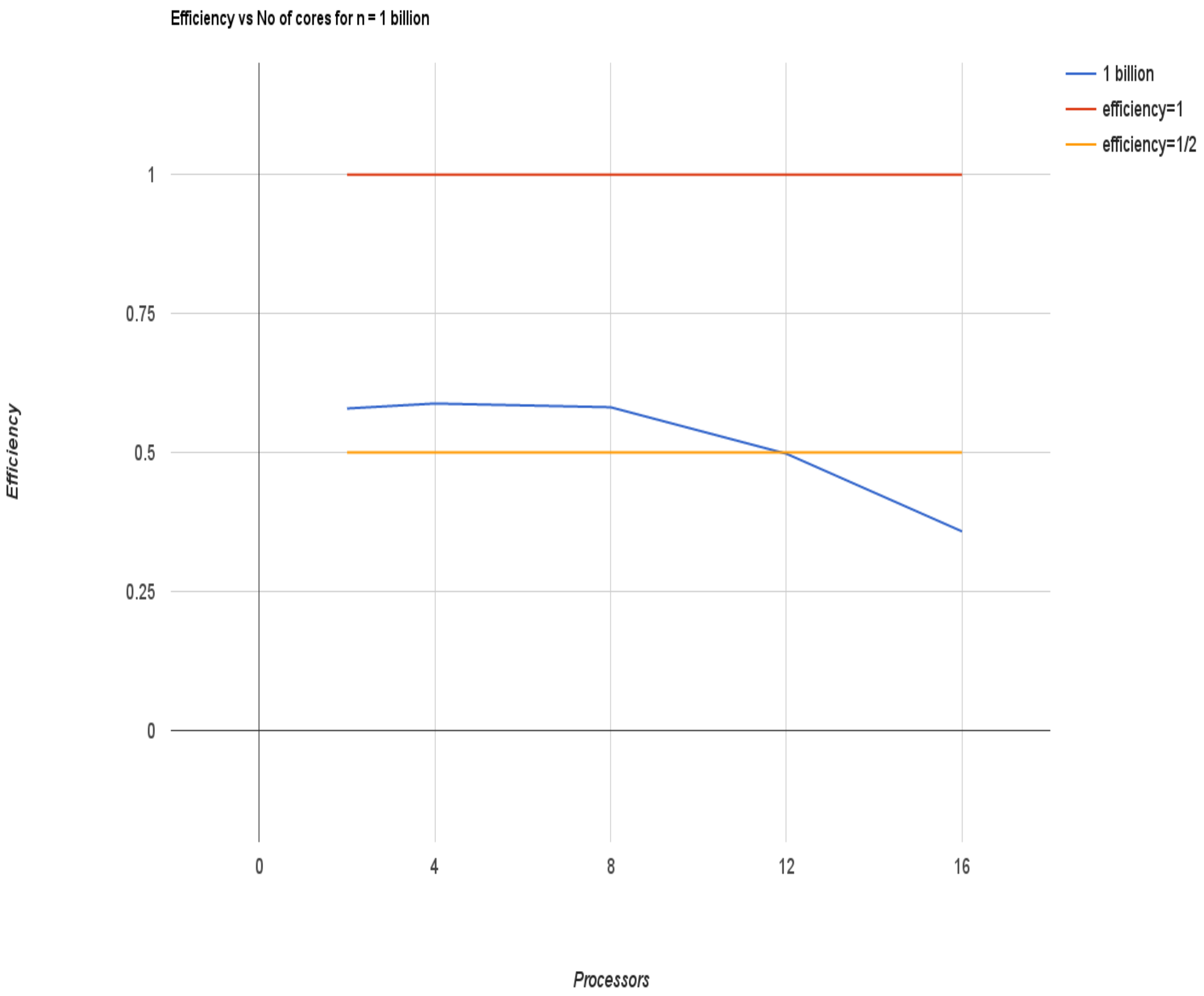
  This only happens when x = ½.
- Thus we can maintain efficiency of 1/2 and speedup of p/2 by increasing n linearly with p. (i.e. put C=1 in equation(1) thus isoefficiency function becomes n >= p).

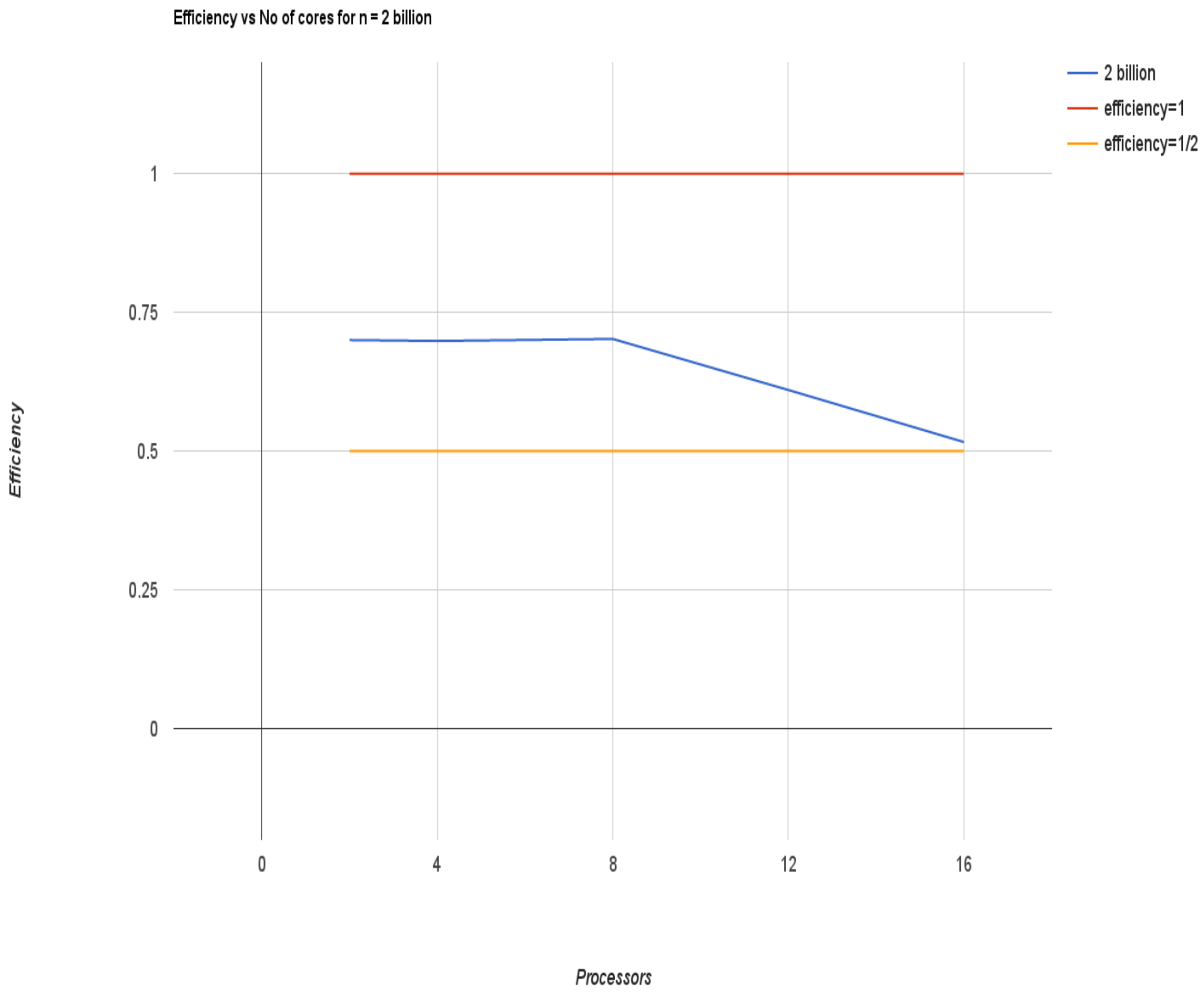
Practical analysis: Amdahl's Law



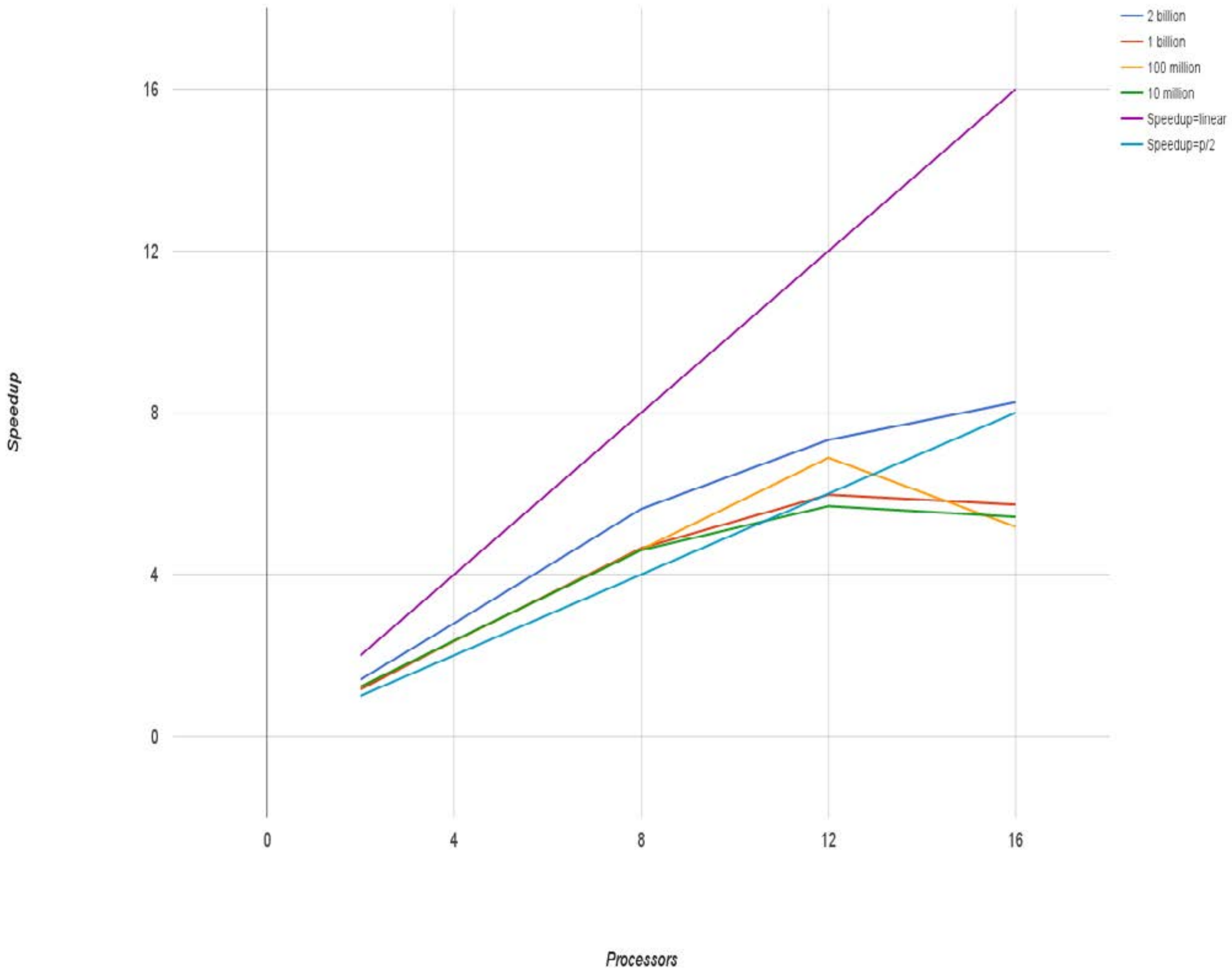


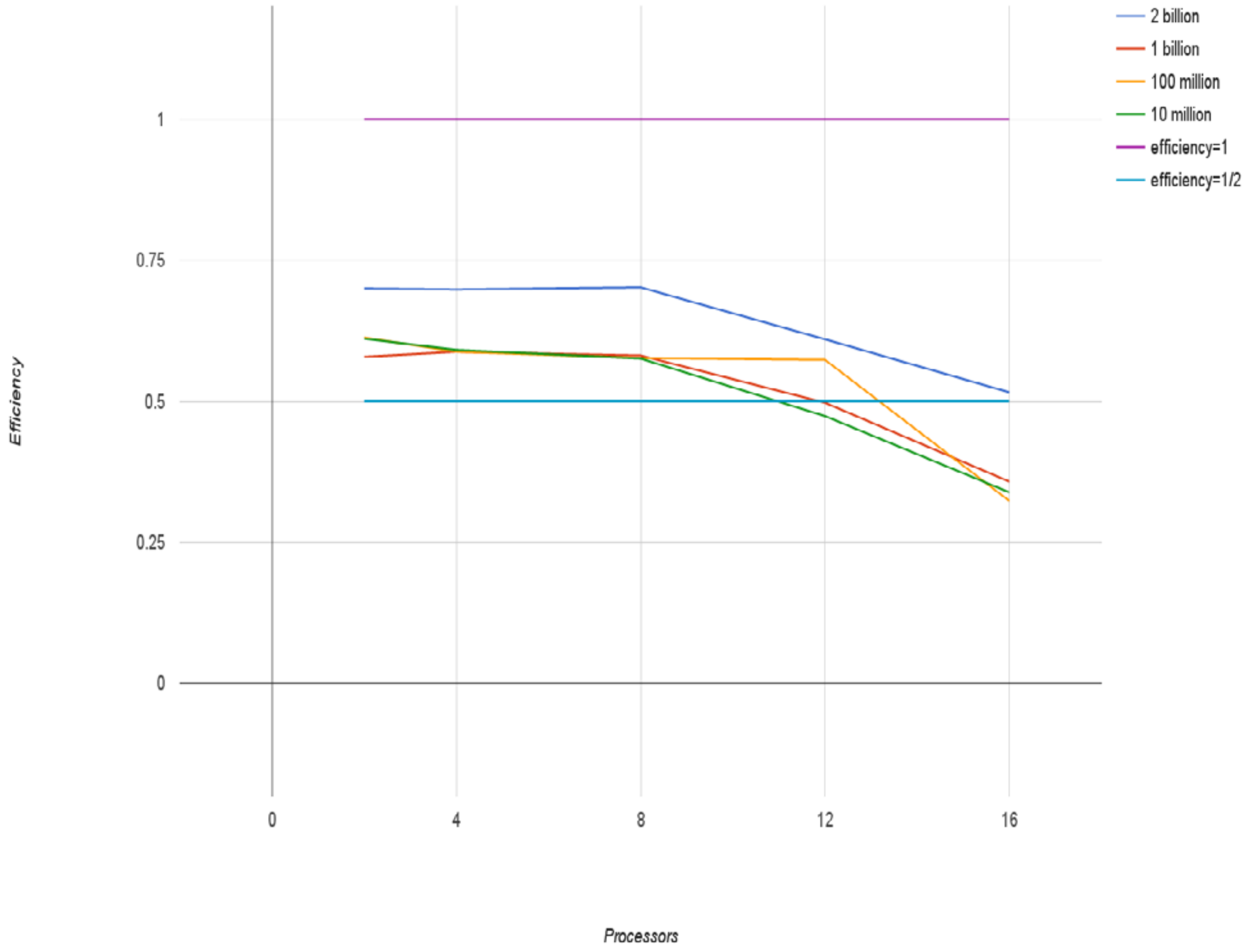
Practical analysis: Amdahl's Law





Practical analysis: Amdahl’s Effect



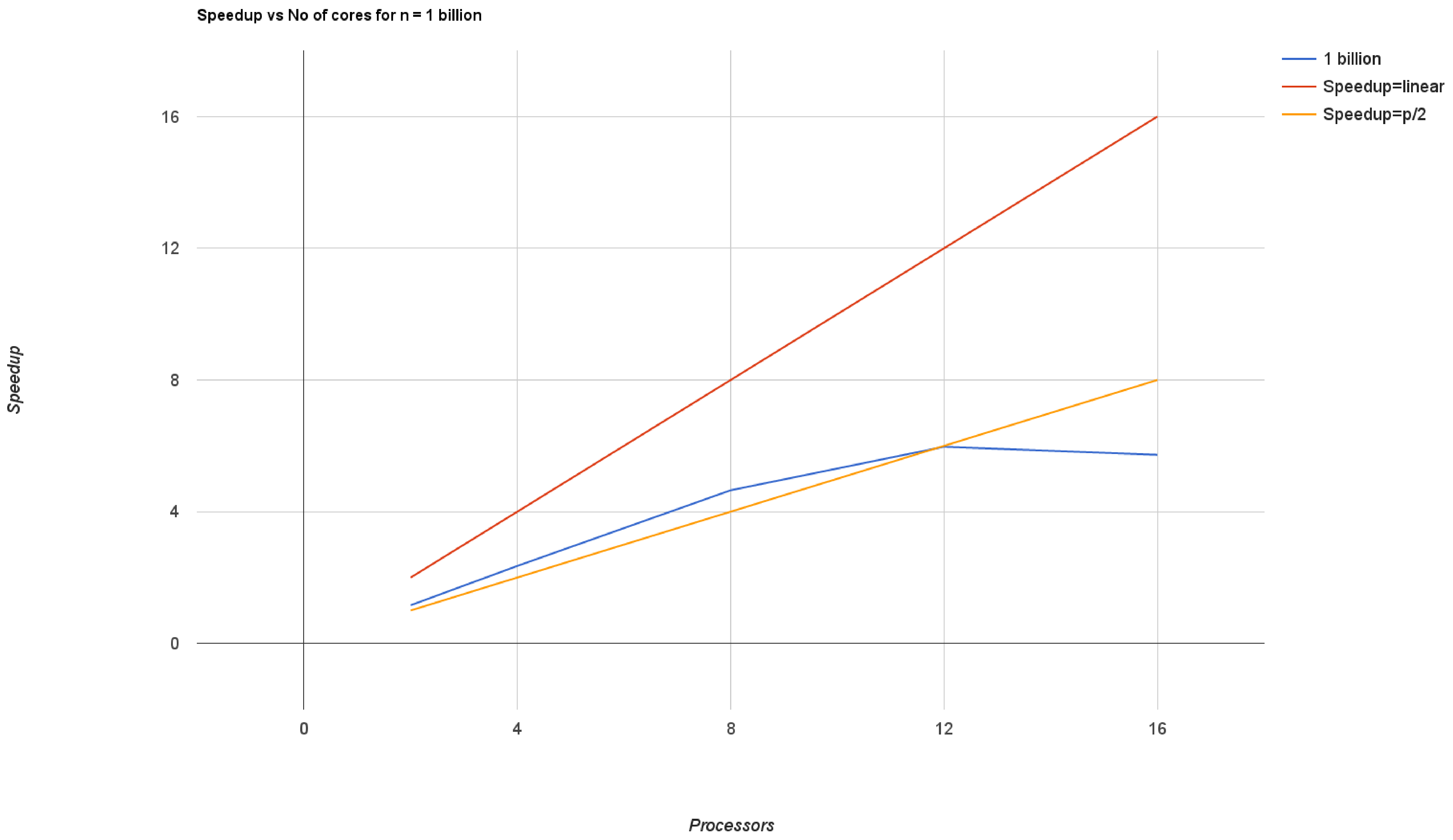


Improvements

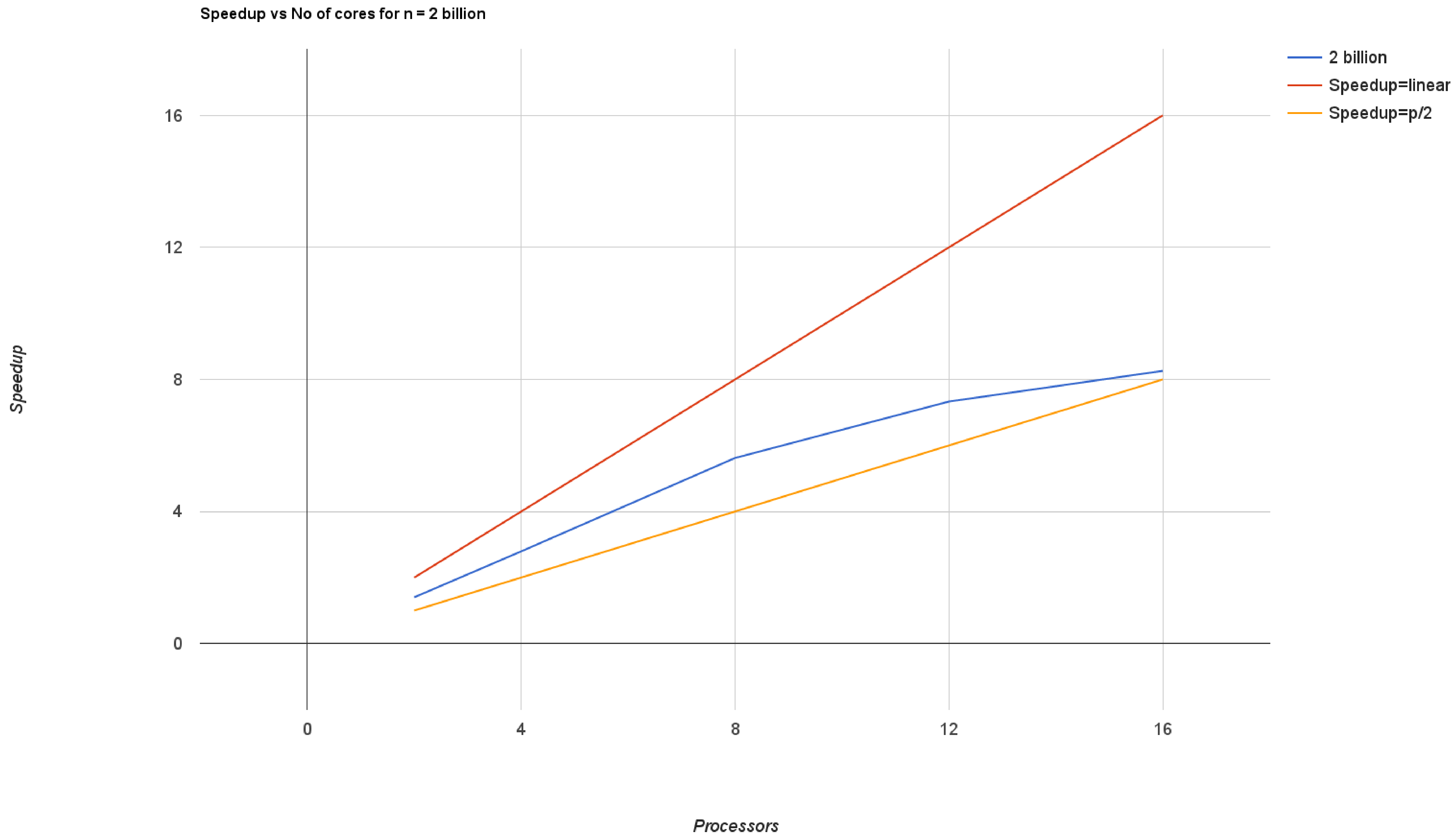
- Based on design strategy of sample sort, several new sorting algorithms have been developed. Most of them focus on 2 things:
- (i) Better Sampling algorithm
- (ii) Decrease communication time
    a) By using shared memory programming model
    b) By improving communication patterns

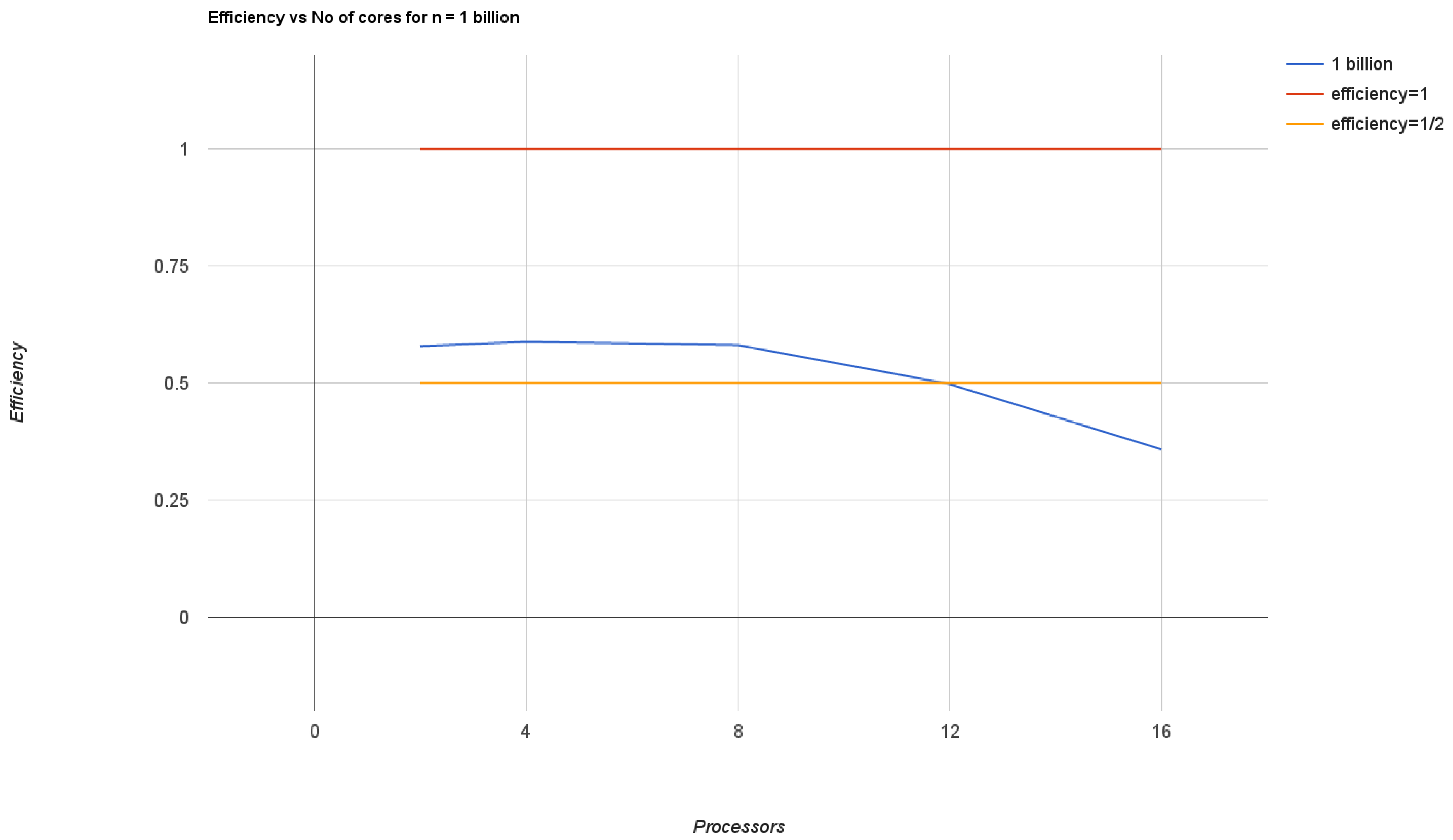
Optional Slides

Below slides have same graphs as shown in slide number 12,13,14. If they are not clearly visible in above slides , you can find them in upcoming slides.

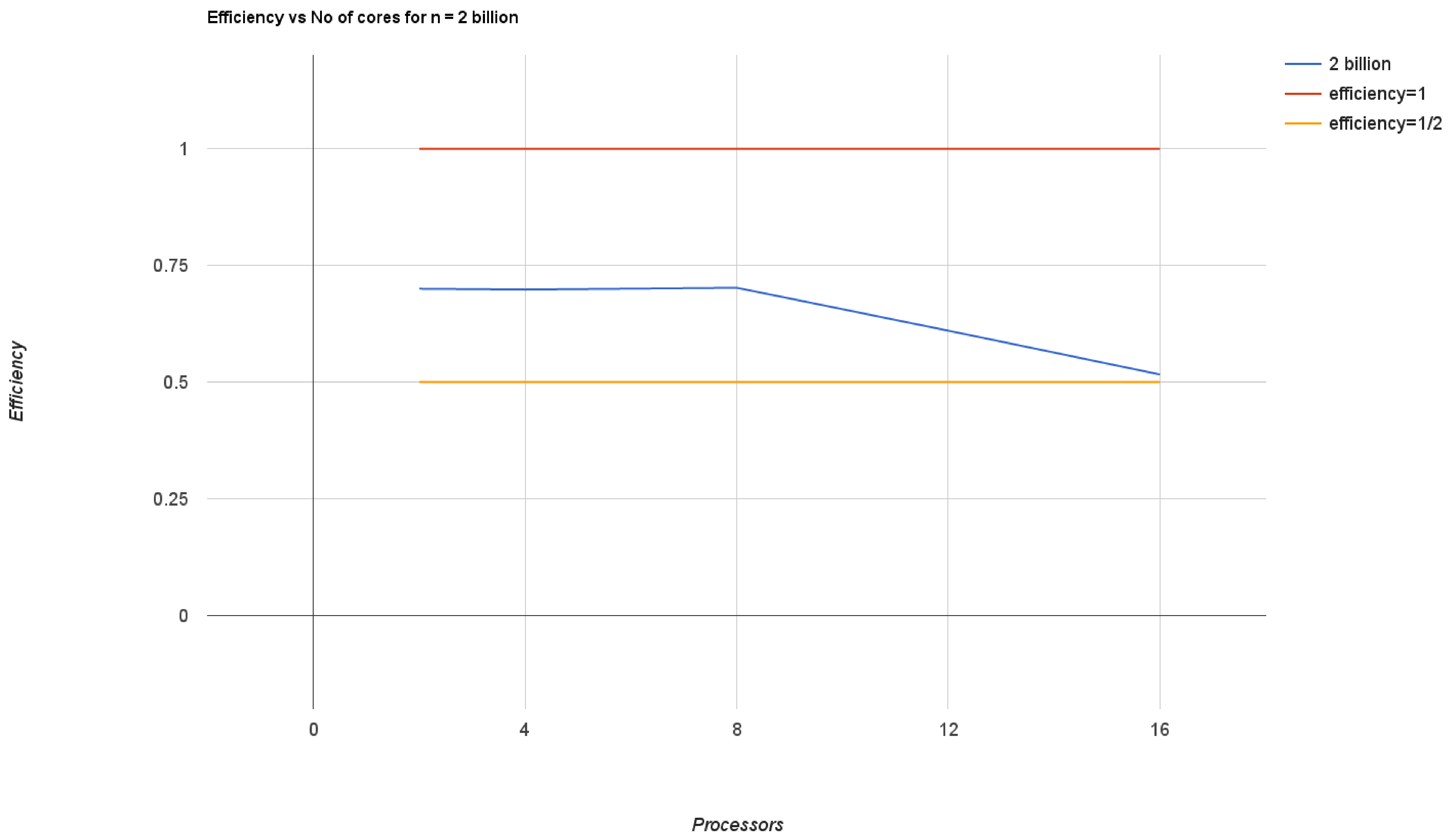


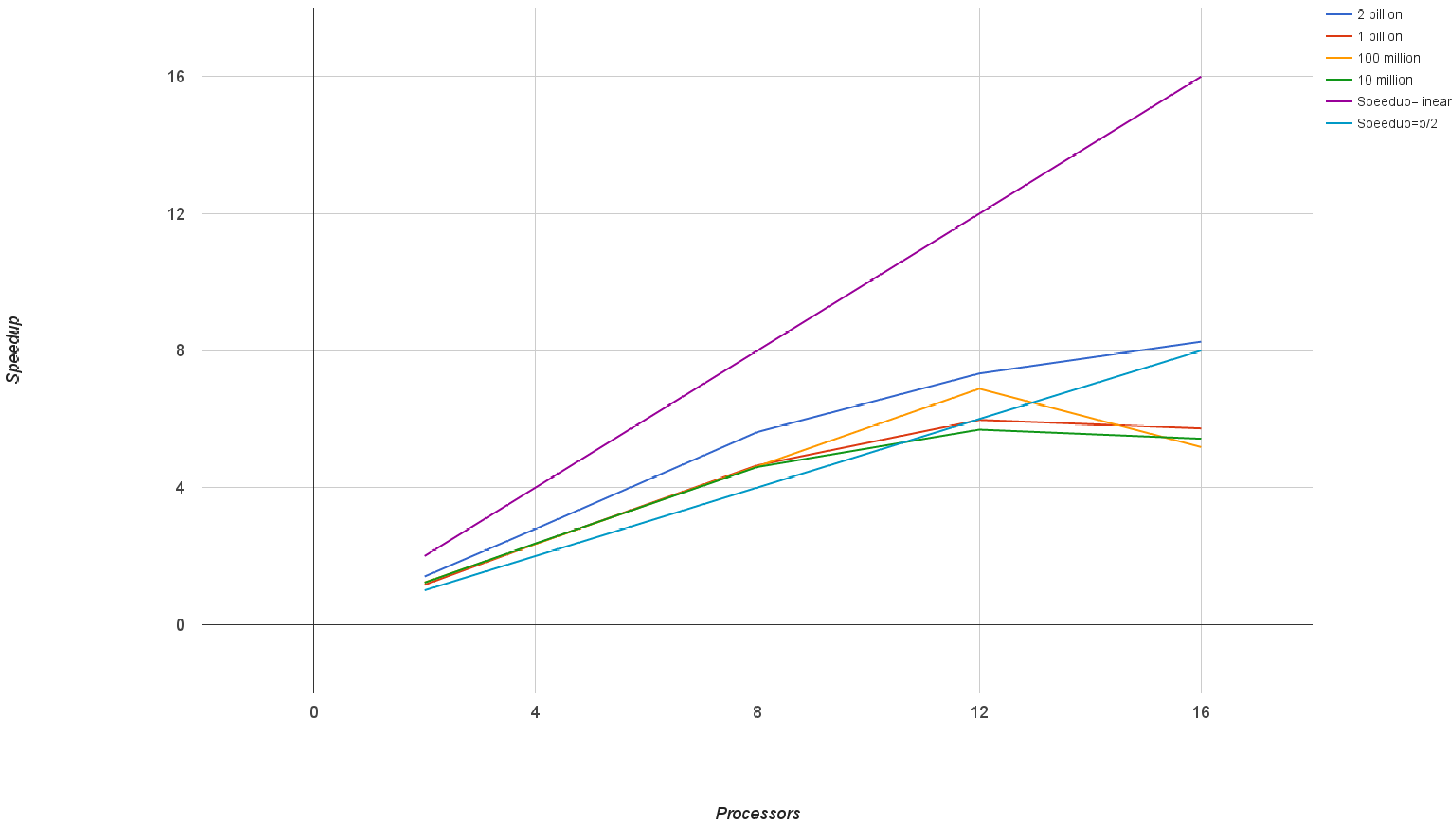
Speedup vs No of cores for n = 1 billion
1 billion
Speedup=linear
Speedup=p/2
Speedup
Processors
0
4
8
12
16
0
4
8
12
16


Speedup vs No of cores for n = 2 billion
2 billion
Speedup=linear
Speedup=p/2
Speedup
Processors
0
4
8
12
16

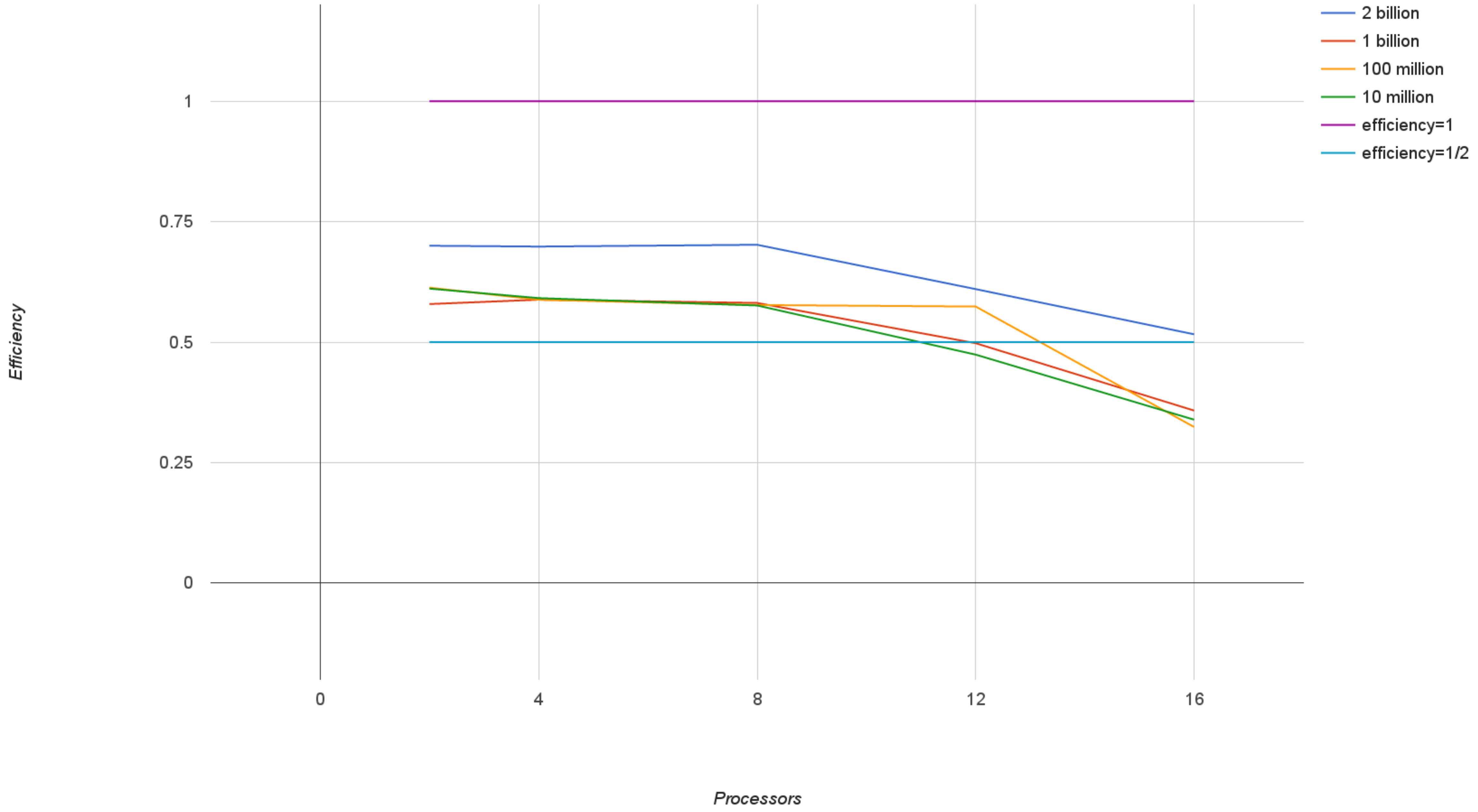


Efficiency vs No of cores for n = 1 billion
1 billion
efficiency=1
efficiency=1/2
1
0.75
0.5
0.25
0
Efficiency
0
4
8
12
16
Processors


Efficiency vs No of cores for n = 2 billion
2 billion
efficiency=1
efficiency=1/2
1
0.75
0.5
0.25
0
Efficiency
0
4
8
12
16
Processors



2 billion
1 billion
100 million
10 million
Speedup=linear
Speedup=p/2
Speedup
Processors
0
4
8
12
16



2 billion
1 billion
100 million
10 million
efficiency=1
efficiency=1/2
Efficiency
1
0.75
0.5
0.25
0
0
4
8
12
16
Processors