Parallel n-body simulation using MPI

Prashant Mishra

Department of Computer Science and Engineering
University at Buffalo

December 9, 2018

Presentation Outline

Introduction

Parallelization

Results

Video!

Conclusion

1 Introduction

2 Parallelization

3 Results

4 Video!

5 Conclusion

A bit of background

- Given n bodies with masses and initial position and velocities, how will they evolve over time under gravitational interaction?

- Interest in the problem arose initially by the desire to understand motions of celestial bodies.

- If n=2, we get exact solutions (they are conic sections).

- If $n > 2$, the equations can not be solved analytically and we must look at numerical solutions.

Naive numerical solution

We have n bodies and their masses $\{m_1, m_2, m_3 \cdots m_n\}$, initial positions $\{{}^{0}\mathbf{x_1}, {}^{0}\mathbf{x_2}, {}^{0}\mathbf{x_3} \cdots {}^{0}\mathbf{x_n}\}$ and velocities $\{{}^{0}\mathbf{v_1}, {}^{0}\mathbf{v_2}, {}^{0}\mathbf{v_3} \cdots {}^{0}\mathbf{v_n}\}$.

The acceleration of body i is :
$\mathbf{a_i} = \mathbf{F_i}/m_i = (\sum_k Gm_i m_k \frac{(\mathbf{x_k} - \mathbf{x_i})}{|\mathbf{x_k} - \mathbf{x_i}|^3})/m_i = \sum_k Gm_k \frac{(\mathbf{x_k} - \mathbf{x_i})}{|\mathbf{x_k} - \mathbf{x_i}|^3}$

And thus, for a time step Δt we can update positions and velocities :

$${}^{\mathbf{t+1}}\mathbf{x_i} - {}^{\mathbf{t}}\mathbf{x_i} = \Delta \mathbf{x} = {}^{\mathbf{t}}\mathbf{v_i}\Delta t + \tfrac{1}{2}{}^{\mathbf{t}}\mathbf{a_i}\Delta t^2$$
$${}^{\mathbf{t+1}}\mathbf{v_i} - {}^{\mathbf{t}}\mathbf{v_i} = \Delta \mathbf{v} = {}^{\mathbf{t}}\mathbf{a_i}\Delta t$$

Every time-step has a runtime $O(n^2)$.

Presentation Outline

1 Introduction

2 Parallelization

3 Results

4 Video!

5 Conclusion

Parallelization

What if we divide the calculation into parts? Here's an overview:

- Master core reads initial data and broadcasts it to the other cores.
- Each processor is responsible for n/p part of the velocity and position update.
- The position has to be distributed to all later for force calculation so we do an MPI_Allgather after every time step.

Minor details

• If I have p number of processors and n bodies, I assign a size $2p * \lceil n/p \rceil$ for positions and velocities, and $p * \lceil n/p \rceil$ for mass wherein the last few positions are left unused. This is to simplify my MPI_Allgather and MPI_Broadcast operation.

• Upside is that the runtime during computation decreases by a factor of p. Downside is one would be spending $O(p^2)$ time on inter-process communication (not really though!).

Minor details

• This is a typical input (I create input using random numbers):

```
 [mass        position_x        position_y
veocity_x     velocity_y]
 [0.234364244112 0.847433736937 0.763774618977
-0.0734792922782 -0.0013694738724 2]
 [0.549491064789 0.651592972723 0.788723351136
-0.121842123968 -0.141495757043]
 [0.93576510392 0.432767067905 0.762280082458
-0.149368183995 -0.0163838417836]
```

• I used cyclic boundary conditions to ensure that the positional coordinates remain between 0 and 1 (thought this might help in visualization later).

Presentation Outline

1 Introduction

2 Parallelization

3 Results

4 Video!

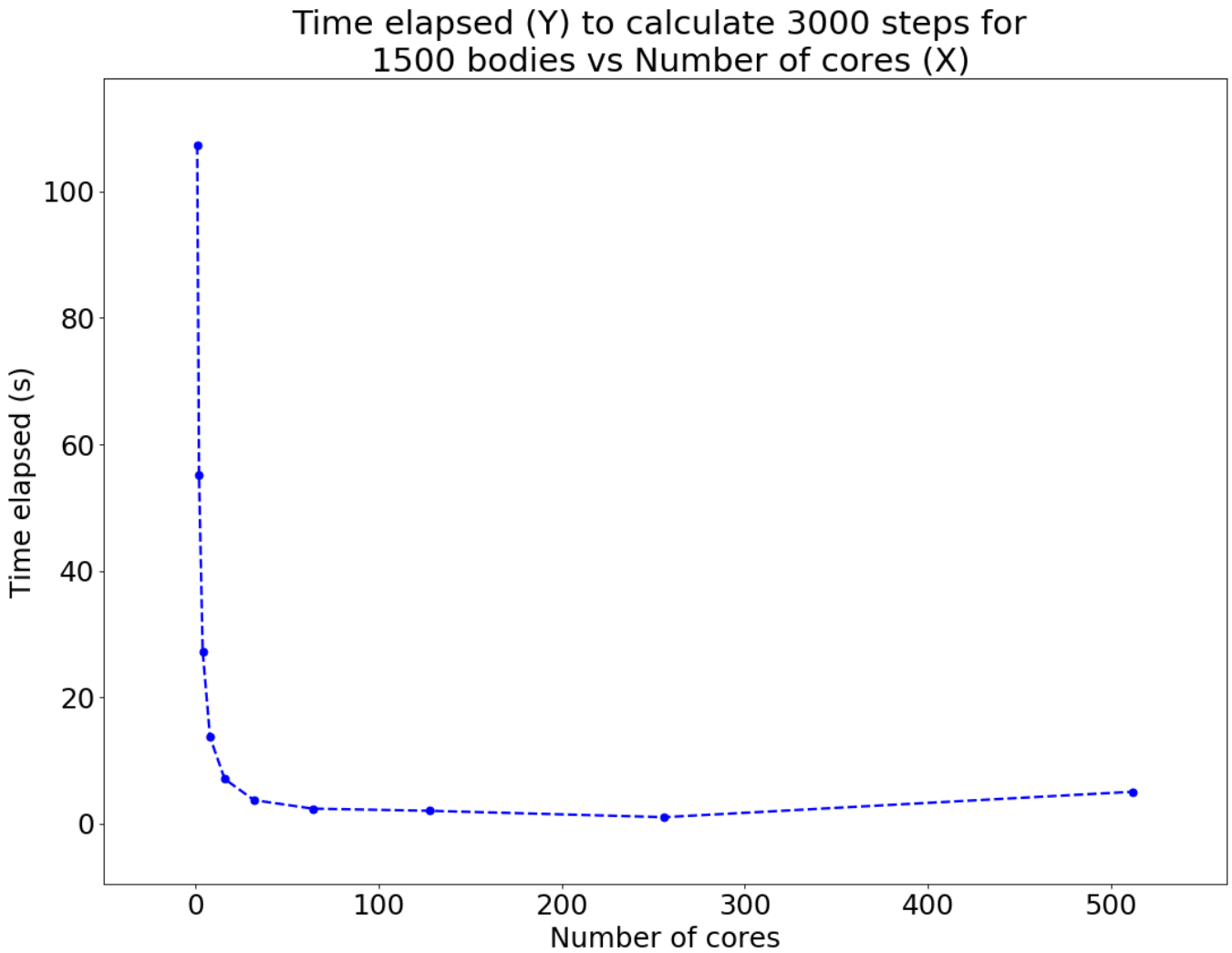
5 Conclusion

Problem size unchanged (1500 particles, 3000 steps)

Cores (Servers x Cores/Server)	Time
1 (1x1)	107.32 seconds
2 (1x2)	55.1237 seconds
4 (1x4)	27.268 seconds
8 (1x8)	13.652 seconds
16 (2x8)	7.0213 seconds
32 (4x8)	3.6874 seconds
64 (8x8)	2.3351 seconds
128 (16x8)	1.992 seconds
256 (32x8)	0.984 seconds
512 (64x8)	5.009 seconds

$$\texttt{Efficiency} = \frac{\texttt{Expected runtime assuming problem scales perfectly}}{\texttt{Actual runtime}}$$

Results


Time elapsed (Y) to calculate 3000 steps for
1500 bodies vs Number of cores (X)
Time elapsed (s)
0
20
40
60
80
100
0
100
200
300
400
500
Number of cores

Results

Parallel n-body simulation using MPI

Prashant Mishra

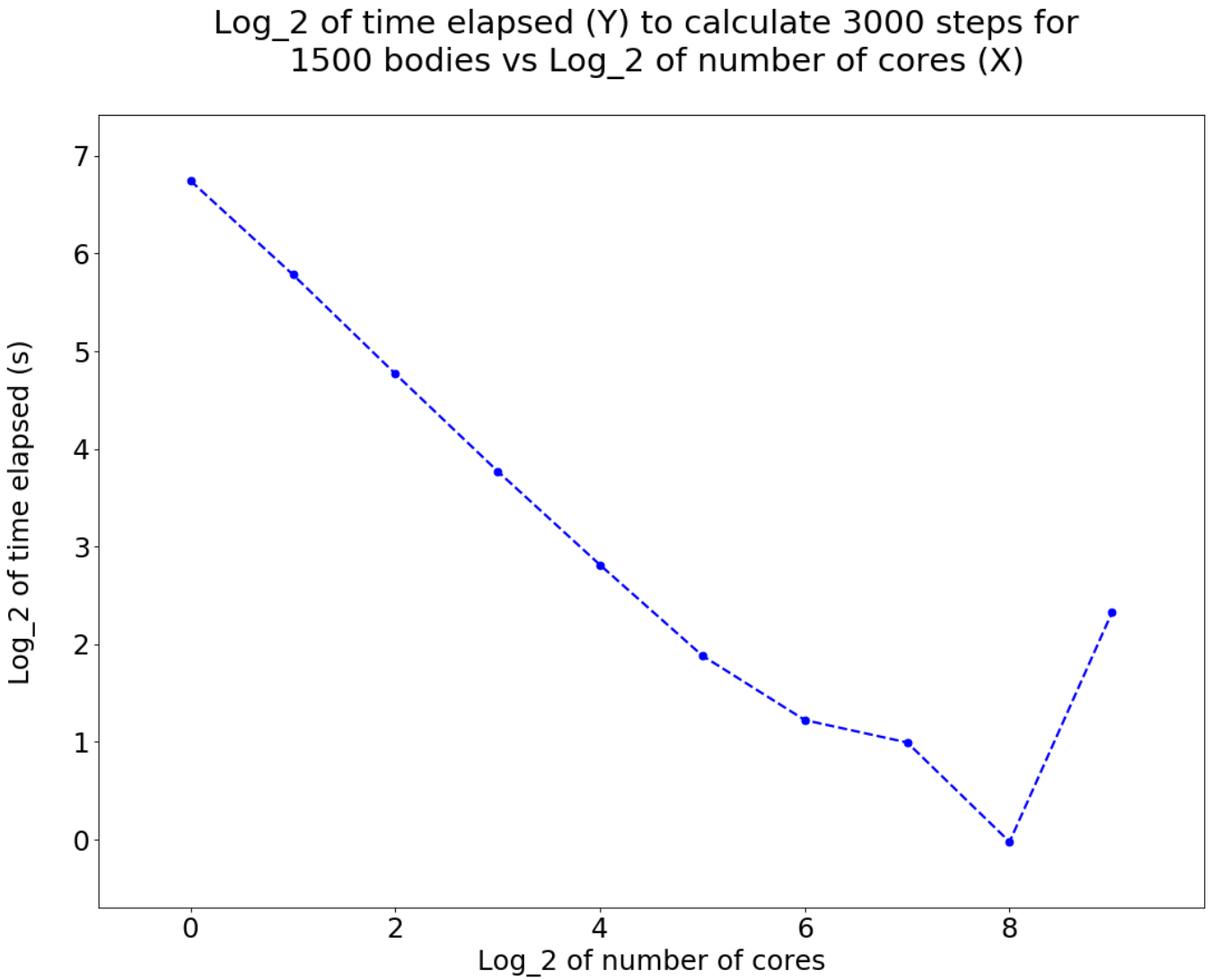
Introduction

Parallelization

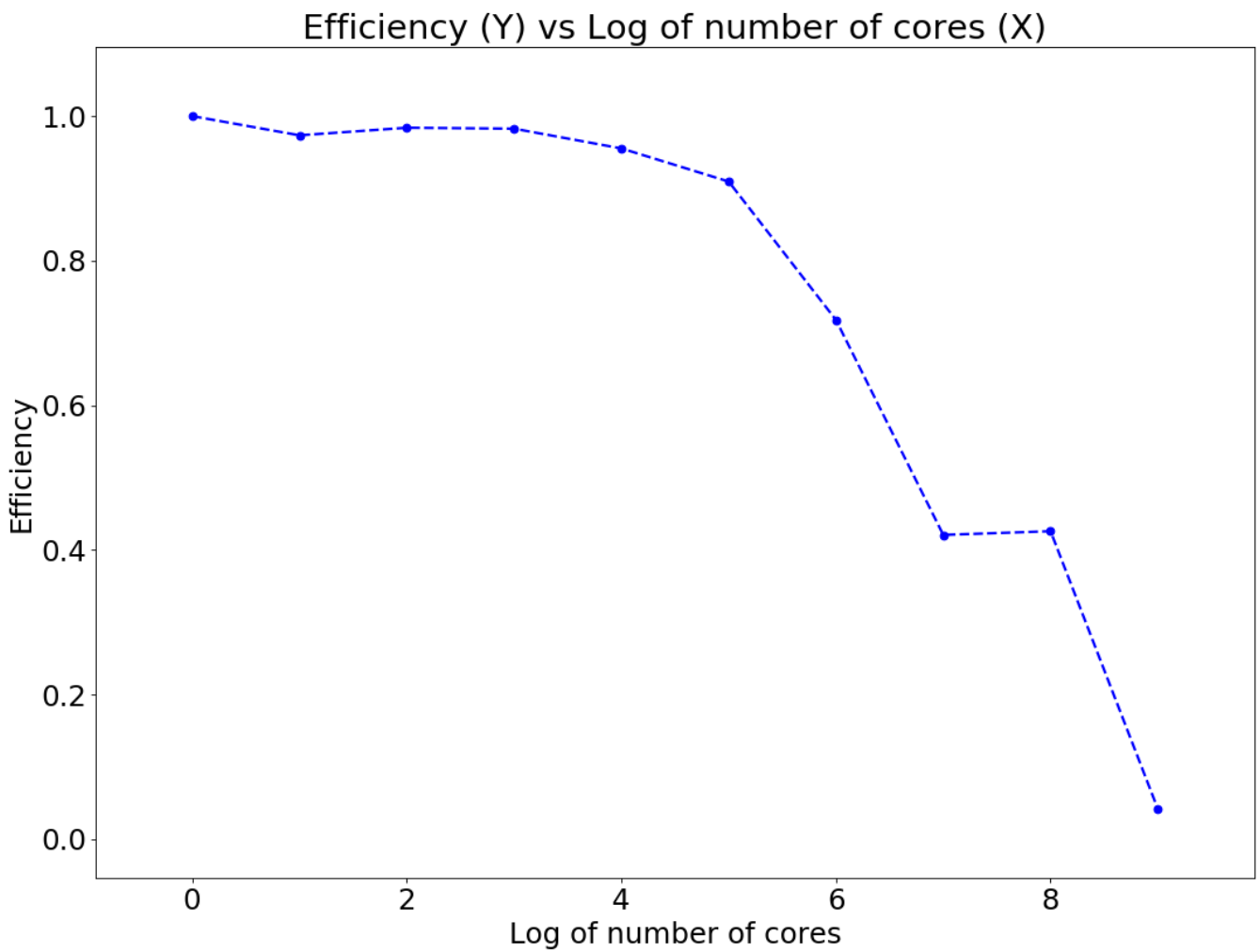
Results

Video!

Conclusion



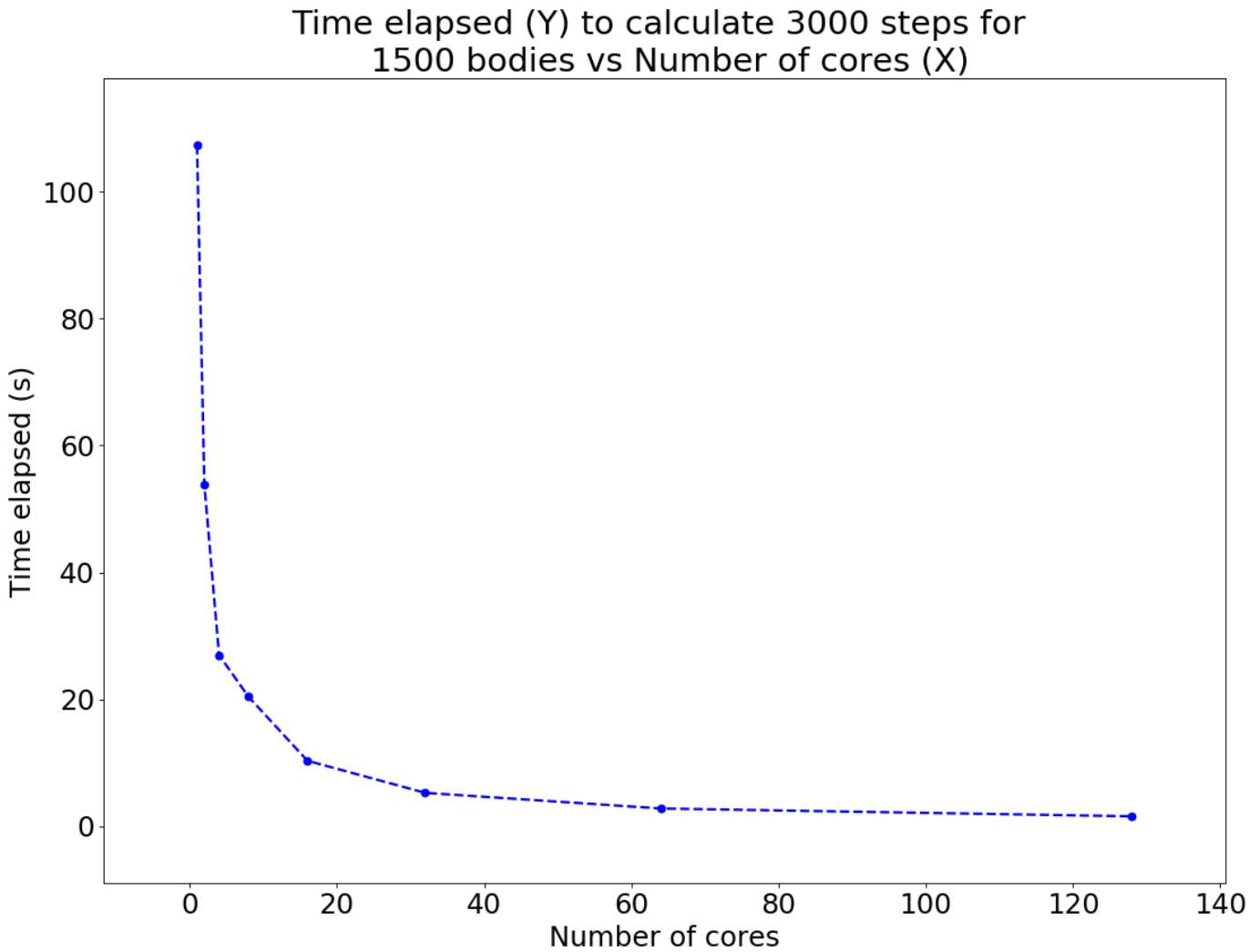
Results


Efficiency (Y) vs Log of number of cores (X)
Efficiency
1.0
0.8
0.6
0.4
0.2
0.0
0
2
4
6
8
Log of number of cores

Problem size unchanged (1500 particles, 3000 steps)

Cores (Servers x Cores/Server)	Time
1 (1x1)	107.32 seconds
2 (2x1)	53.817 seconds
4 (4x1)	26.968 seconds
8 (8x1)	20.38 seconds
16 (16x1)	10.307 seconds
32 (32x1)	5.243 seconds
64 (64x1)	2.7806 seconds
128 (128x1)	1.5457 seconds

Results


Time elapsed (Y) to calculate 3000 steps for
1500 bodies vs Number of cores (X)
Time elapsed (s)
0
20
40
60
80
100
Number of cores
0
20
40
60
80
100
120
140

Results

Parallel n-body simulation using MPI

Prashant Mishra

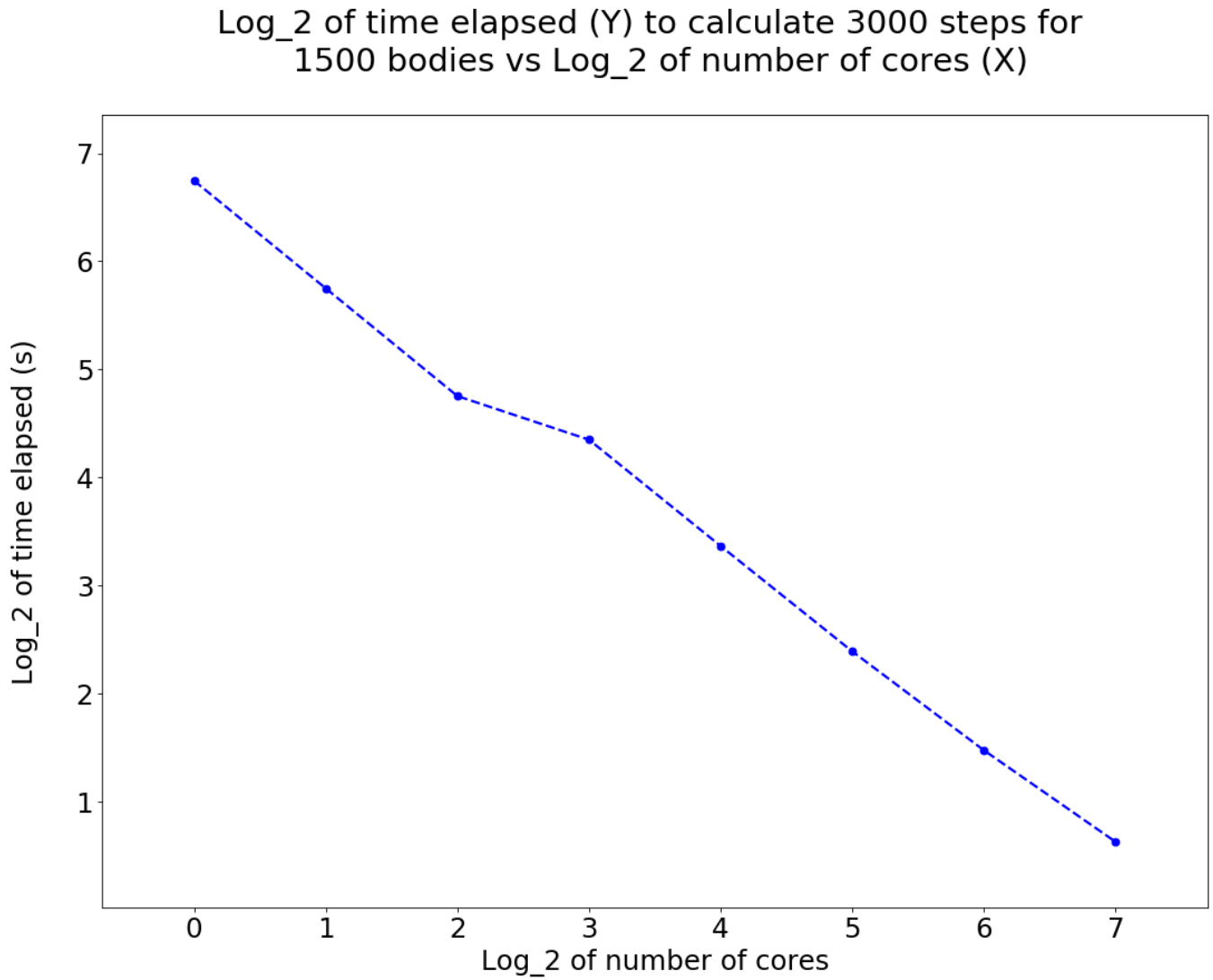
Introduction

Parallelization

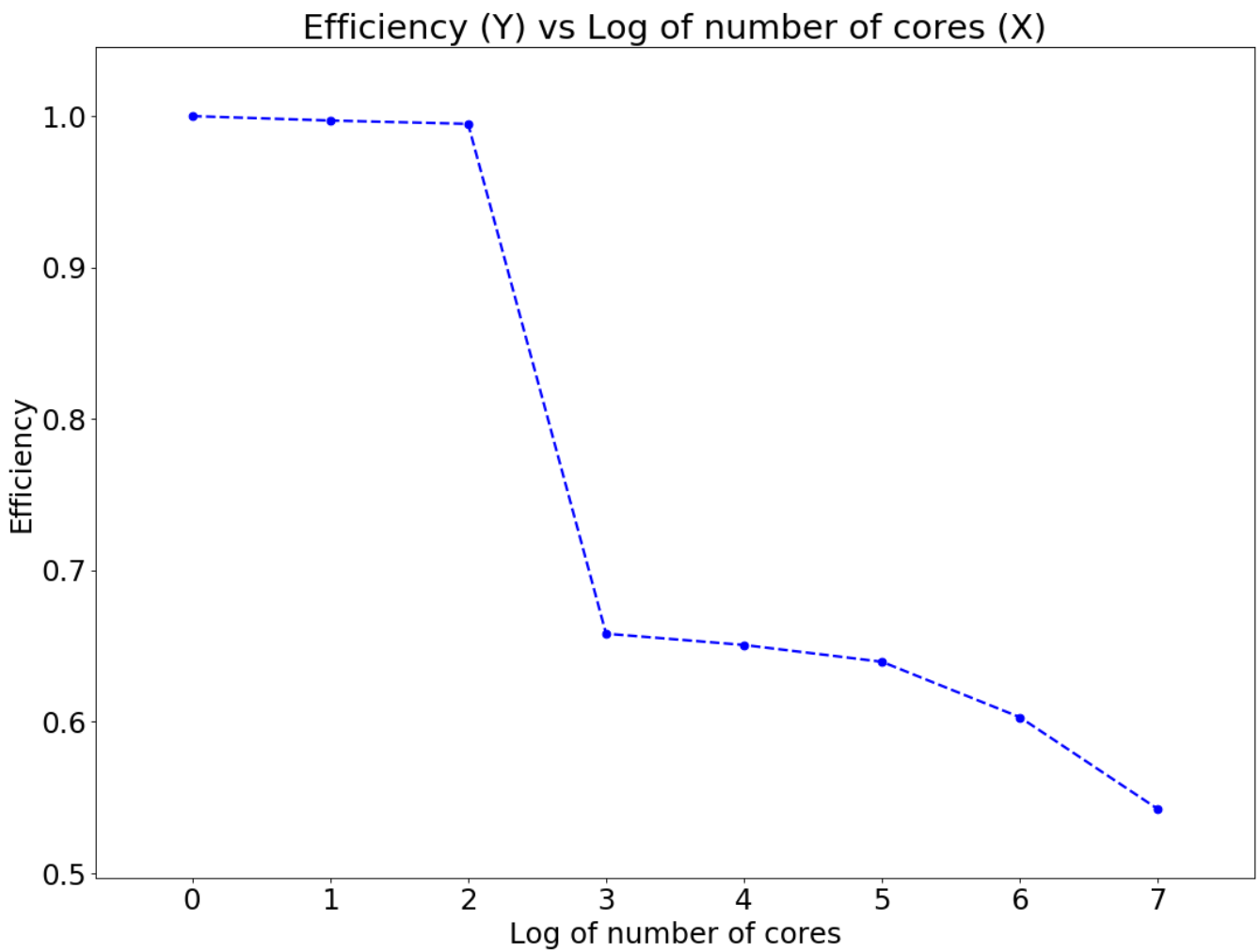
Results

Video!

Conclusion



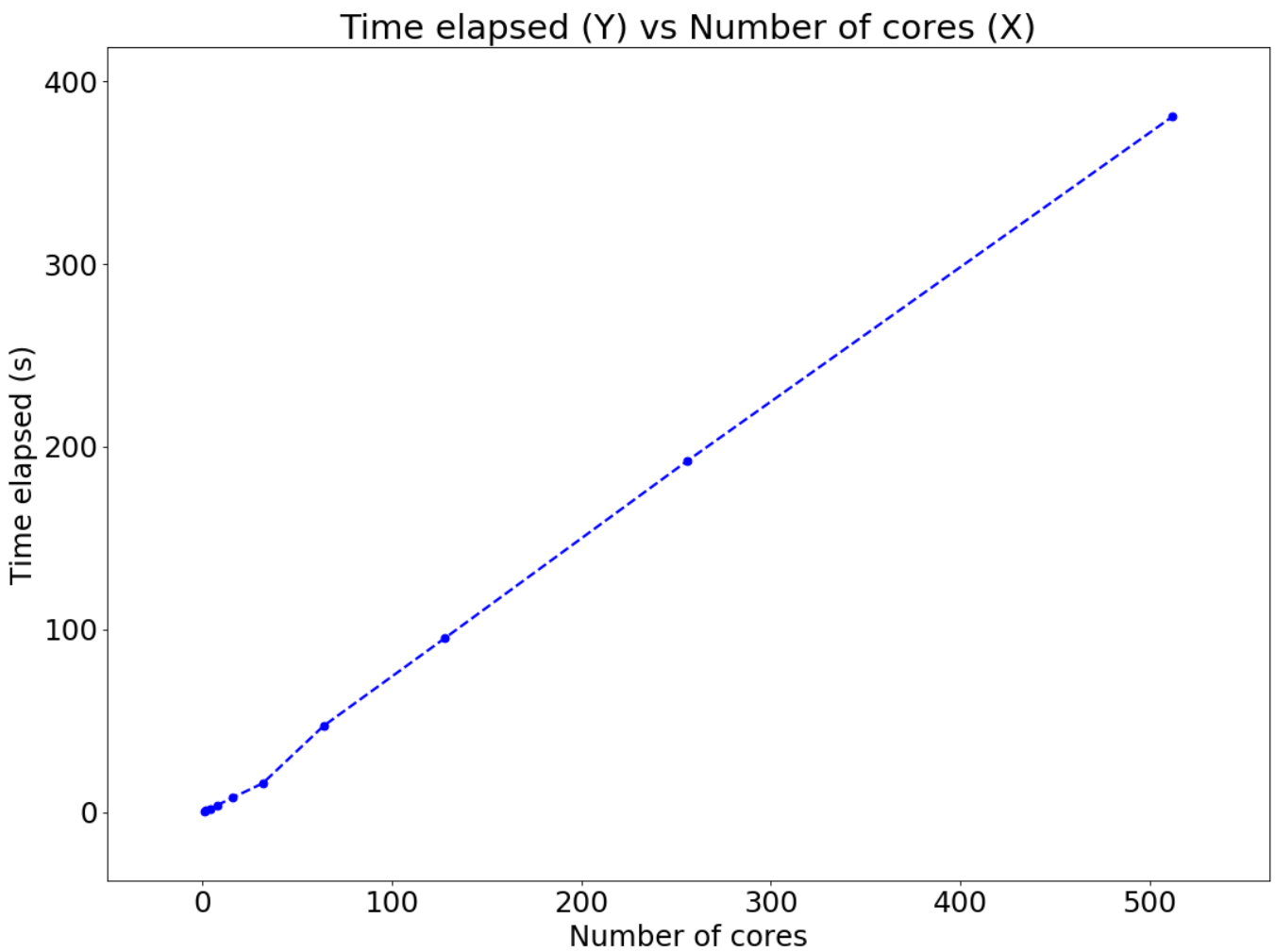
Results


Efficiency (Y) vs Log of number of cores (X)
Efficiency
1.0
0.9
0.8
0.7
0.6
0.5
0
1
2
3
4
5
6
7
Log of number of cores

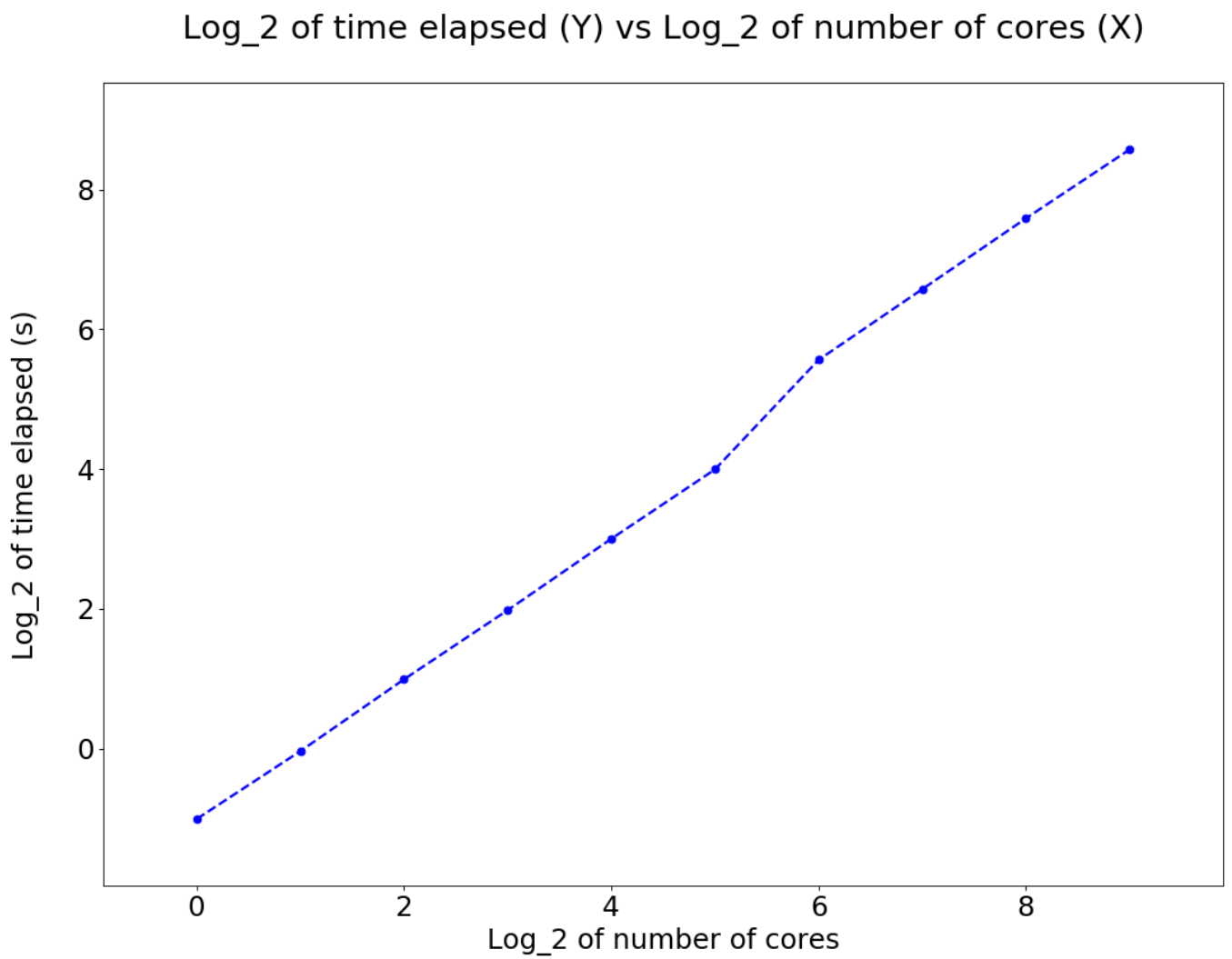
Scaling problem size (3000 steps)

Particles	Cores (Servers x Cores/Server)	Time
100	1 (1x1)	0.4973 seconds
200	2 (1x2)	0.9746 seconds
400	4 (1x4)	1.987 seconds
800	8 (1x8)	3.9455 seconds
1600	16 (2x8)	8.012 seconds
3200	32 (4x8)	15.943 seconds
6400	64 (8x8)	47.943 seconds
12800	128 (16x8)	95.284 seconds
25600	256 (32x8)	192.298 seconds
51200	512 (64x8)	380.687 seconds

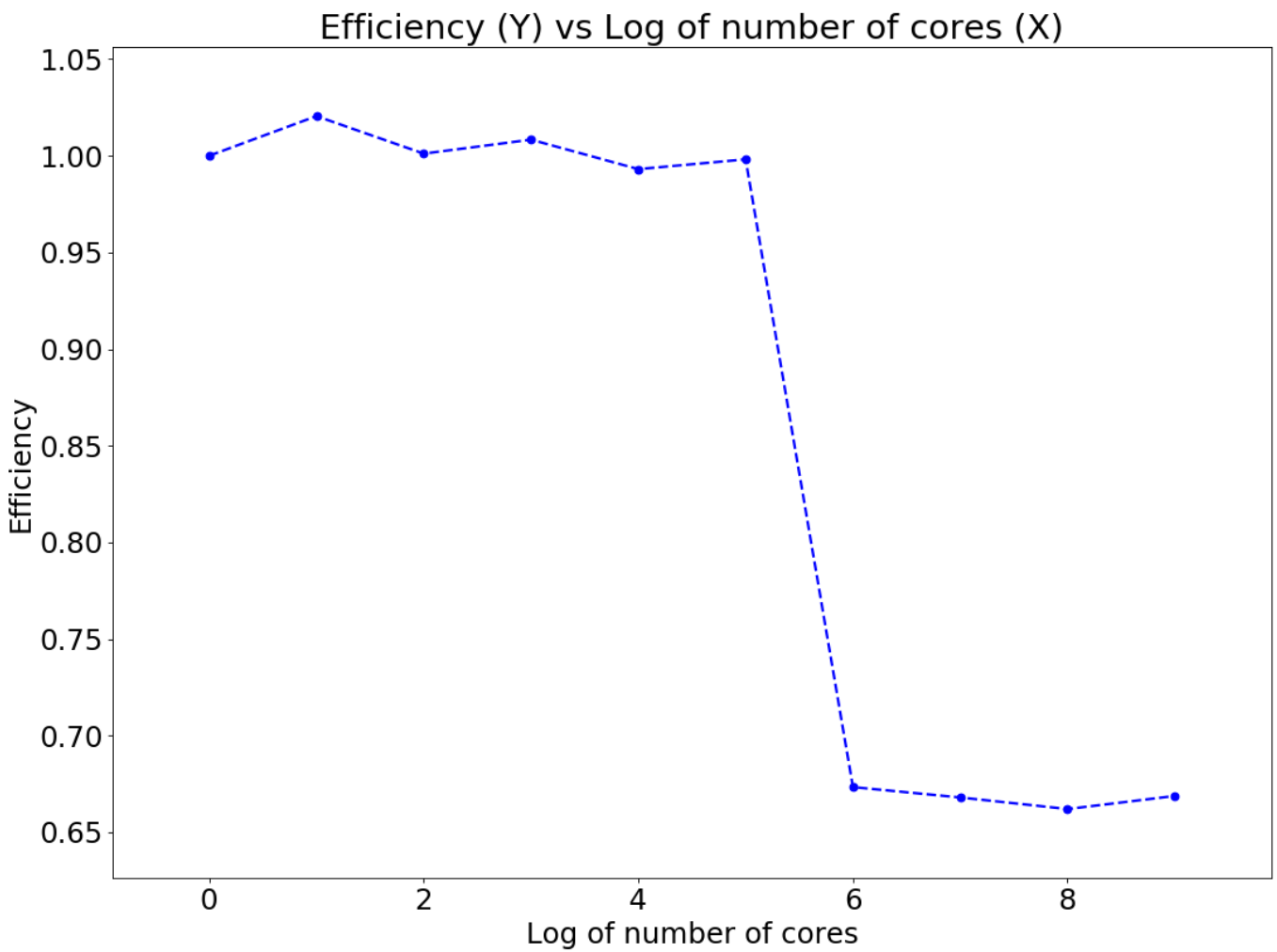
Results


Time elapsed (Y) vs Number of cores (X)
Time elapsed (s)
0
100
200
300
400
0
100
200
300
400
500
Number of cores

Results


Log_2 of time elapsed (Y) vs Log_2 of number of cores (X)
Log_2 of time elapsed (s)
0
2
4
6
8
Log_2 of number of cores
0
2
4
6
8

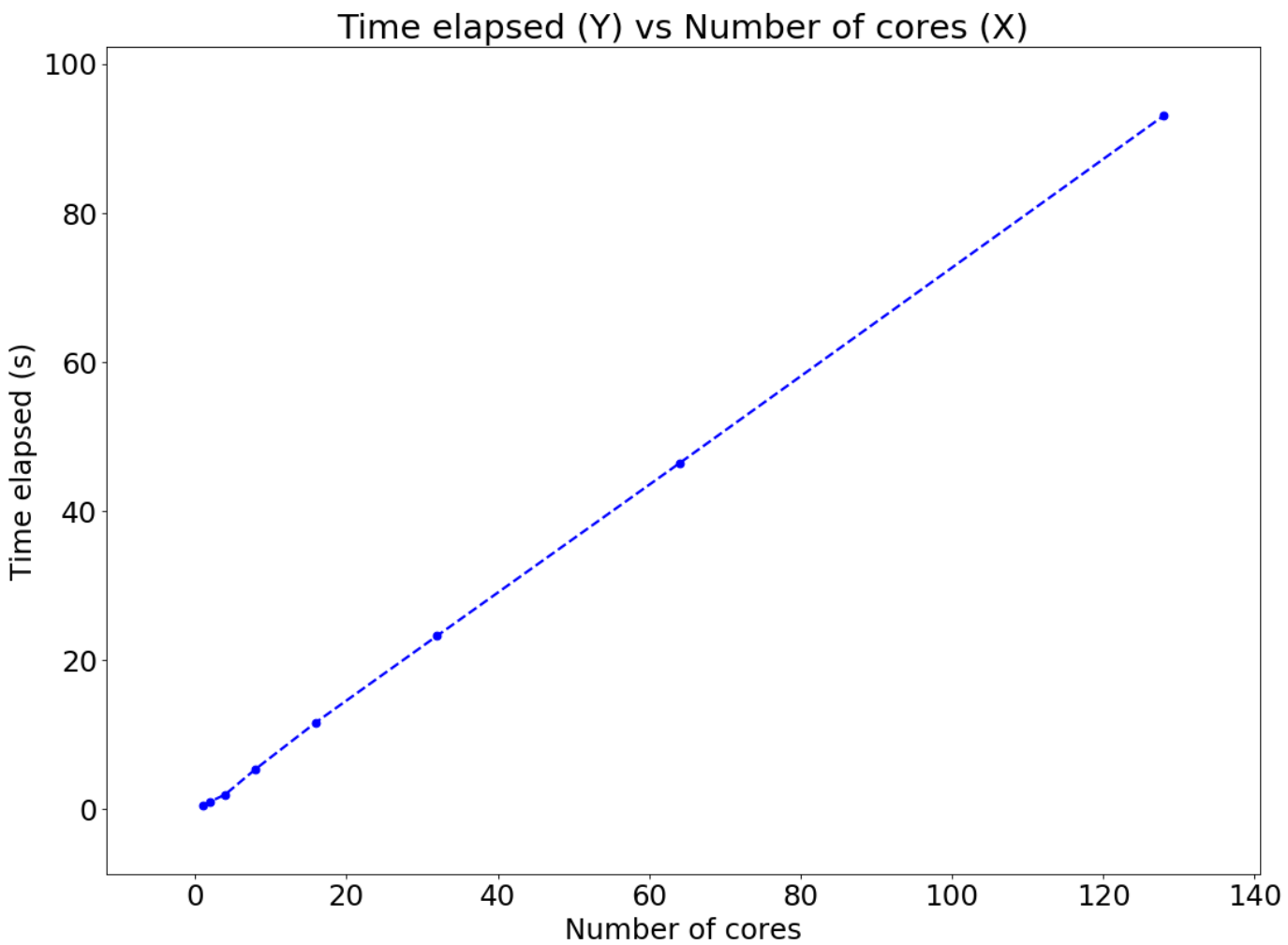
Results


Efficiency (Y) vs Log of number of cores (X)
Efficiency
1.05
1.00
0.95
0.90
0.85
0.80
0.75
0.70
0.65
0
2
4
6
8
Log of number of cores

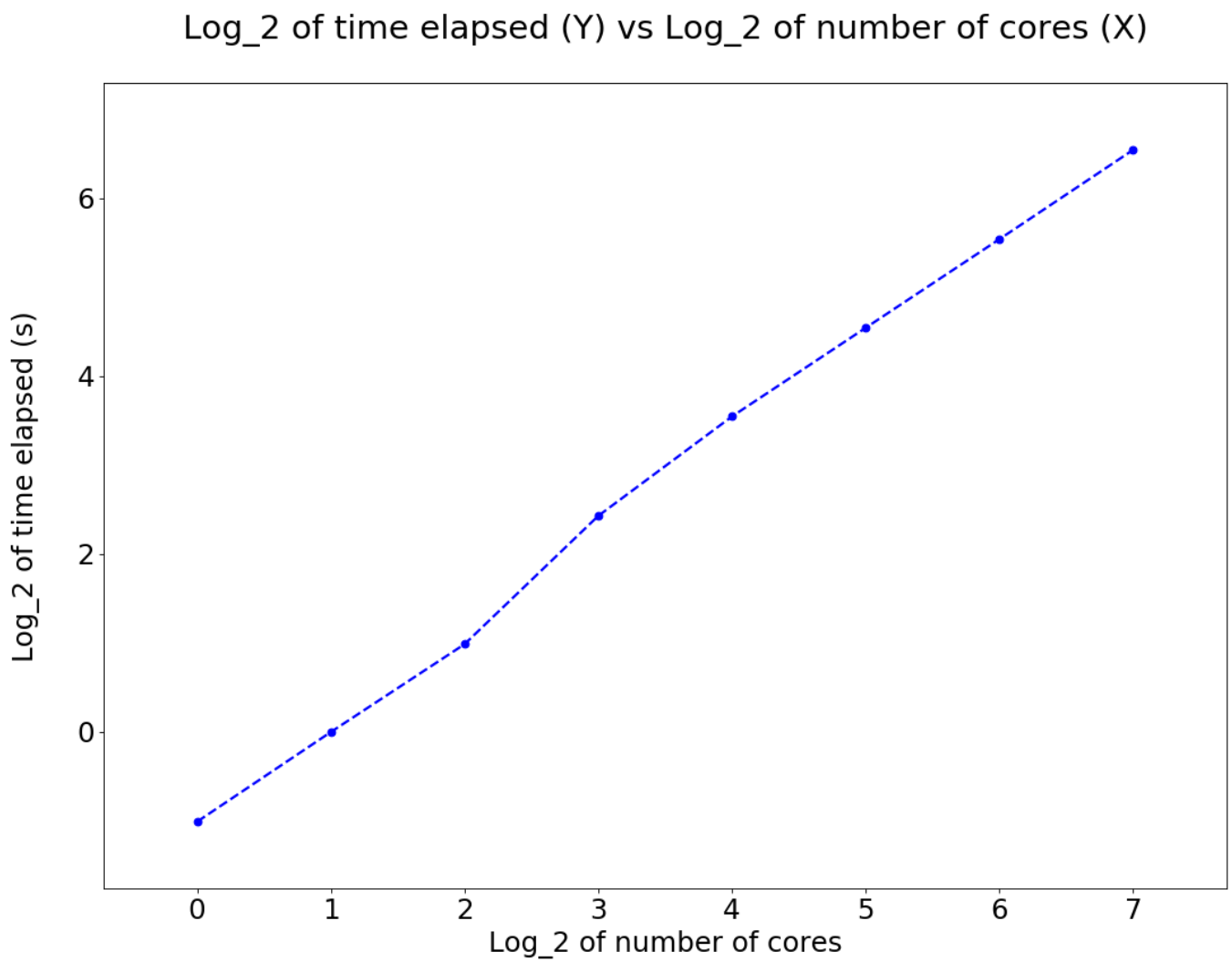
Scaling problem size (3000 steps)

Particles	Cores (Servers x Cores/Server)	Time
100	1 (1x1)	107.32 seconds
200	2 (2x1)	0.9996 seconds
400	4 (4x1)	1.9813 seconds
800	8 (8x1)	5.382 seconds
1600	16 (16x1)	11.673 seconds
3200	32 (32x1)	23.283 seconds
6400	64 (64x1)	46.437 seconds
12800	128 (128x1)	93.099 seconds

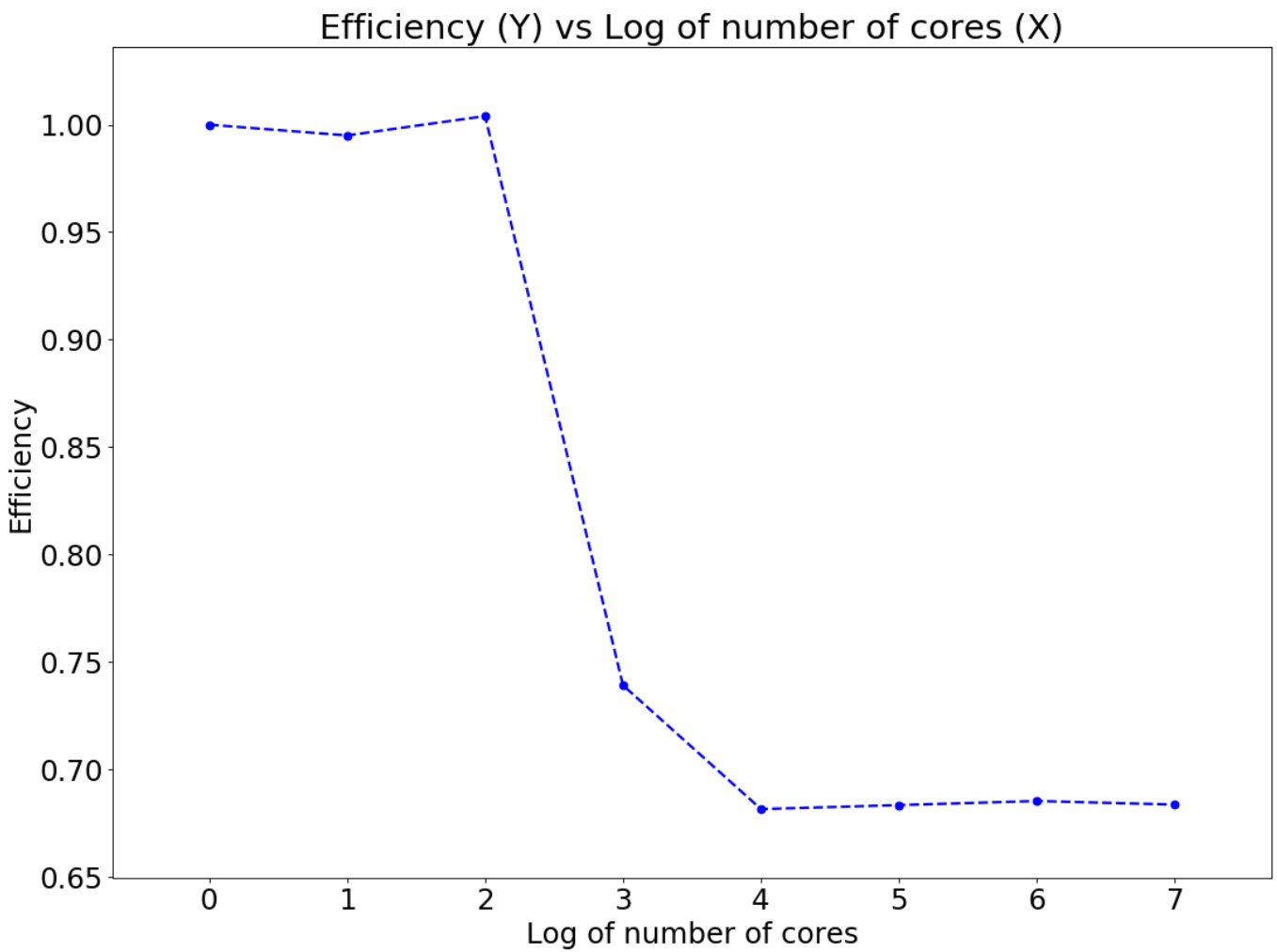
Results


Time elapsed (Y) vs Number of cores (X)
Time elapsed (s)
Number of cores
0
20
40
60
80
100
120
140

Results


Log_2 of time elapsed (Y) vs Log_2 of number of cores (X)
Log_2 of time elapsed (s)
0
2
4
6
0
1
2
3
4
5
6
7
Log_2 of number of cores

Results


Efficiency (Y) vs Log of number of cores (X)
Efficiency
1.00
0.95
0.90
0.85
0.80
0.75
0.70
0.65
0
1
2
3
4
5
6
7
Log of number of cores

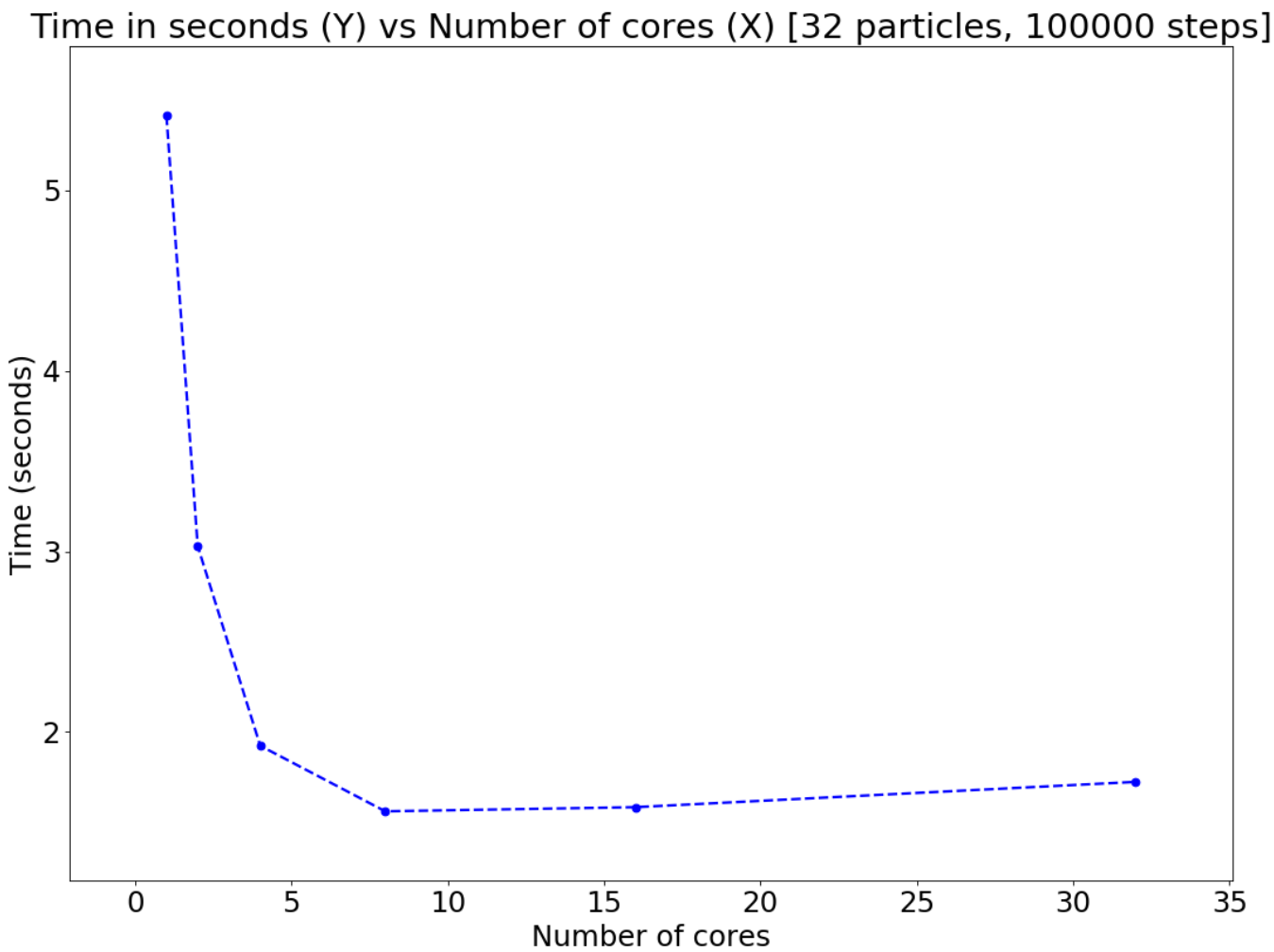
Amdahl's Law

- Although the results show that the problem scales well, it is hard to see Amdahl's law in action.
- This is because, the number of particles being so high, the ratio of communicaton to computation is very low.
- To simulate a higher ratio, we now only use 32 particles for calculations.
- This gives us a U-shaped graph in accordance with Amdahl's law.

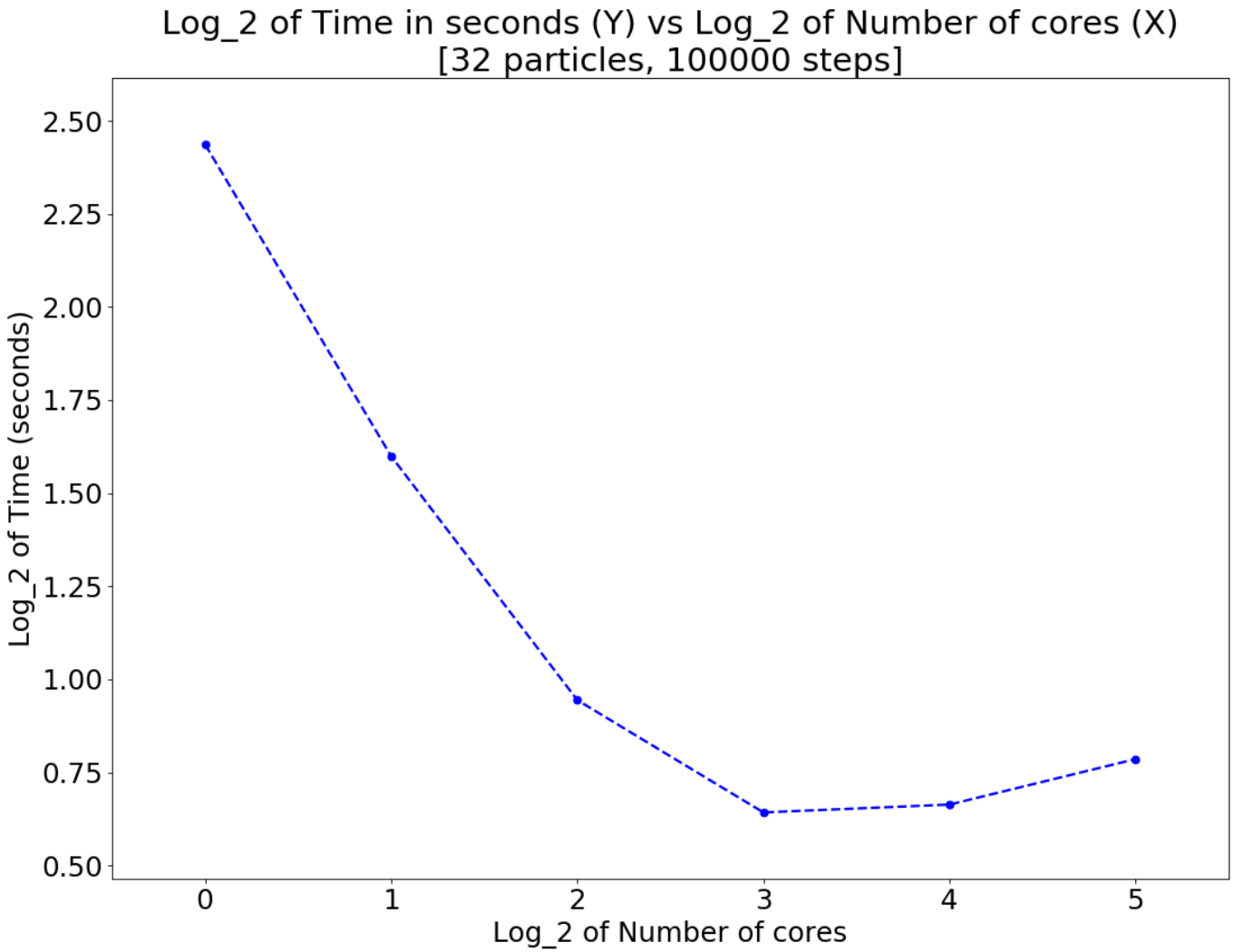
32 particles, 100000 steps

Cores (Servers x Cores/Server)	Time
1 (1x1)	5.412 seconds
2 (2x1)	3.029 seconds
4 (4x1)	1.924 seconds
8 (8x1)	1.561 seconds
16 (16x1)	1.584 seconds
32 (32x1)	1.724 seconds

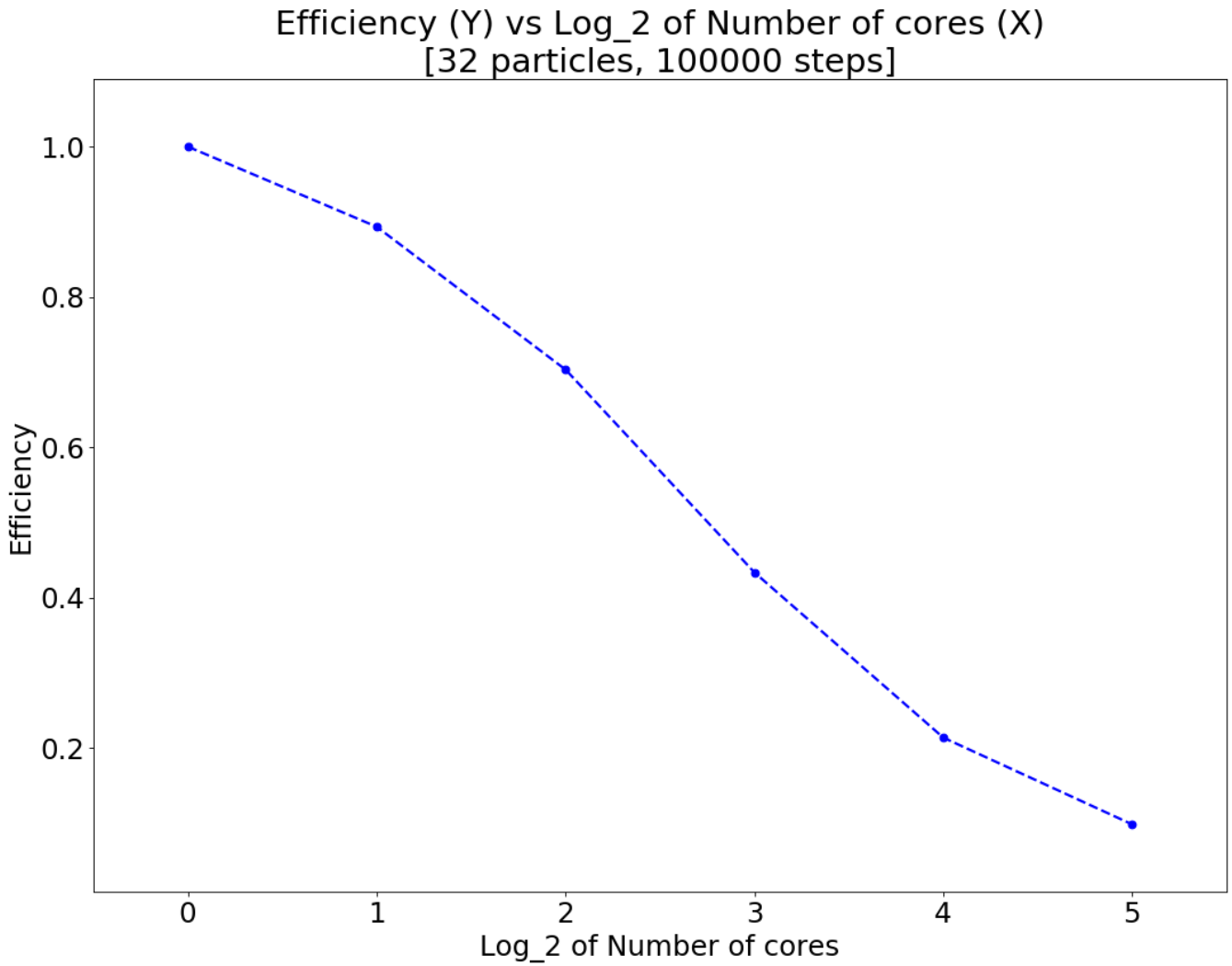
Results


Time in seconds (Y) vs Number of cores (X) [32 particles, 100000 steps]
Time (seconds)
2
3
4
5
0
5
10
15
20
25
30
35
Number of cores

Results


Log_2 of Time in seconds (Y) vs Log_2 of Number of cores (X)
[32 particles, 100000 steps]
Log_2 of Time (seconds)
2.50
2.25
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0
1
2
3
4
5
Log_2 of Number of cores

Results


Efficiency (Y) vs Log_2 of Number of cores (X)
[32 particles, 100000 steps]
Efficiency
1.0
0.8
0.6
0.4
0.2
0
1
2
3
4
5
Log_2 of Number of cores

Presentation Outline

Introduction

Parallelization

Results

Video!

Conclusion

1 Introduction

2 Parallelization

3 Results

4 Video!

5 Conclusion

Videos!

- Created these videos by storing the particle states after each step, using this data and python to create scatter plots, and then using `ffmpeg` to stitch there plots (frames).
- The idea was for these videos to act as a sanity check, that the simulation made sense and the boundary conditions were being met.
- Created videos for 5, 15, 25, 50 and 100 particles as it's hard to see anything after that.
- Note that a specialized PDF reader might be required to see the videos (I used Okular).

A small video of the simulation (5 particles)

A small video of the simulation (15 particles)

A small video of the simulation (25 particles)

A small video of the simulation (50 particles)

A small video of the simulation (100 particles)

Presentation Outline

1 Introduction

2 Parallelization

3 Results

4 Video!

5 Conclusion

Reference & Wrapping up

- The problem scales very well if the number of particles is large! We have to use a relatively low number of particles so that the ratio of communication to computation is high to see Amdahl's law in action.

- Link to a tutorial I found useful:
`http://mpitutorial.com/tutorials/`

- Link to code if anyone is interested:
`https://www.github.com/prashantmishra/n_body_mpi`

- Questions?

Thank You!