Linear Algebra Optimization

Aidan Levy

Why optimize Linear Algebra with OpenMPI?

- Linear algebra is the foundation of scientific computing
- Real world problems need fast matrix operations
  - Computer graphics
  - Large calculations
- Sequential processing is too slow for these large matrices
- So, my goal is to use OpenMPI in C to massively parallelize matrix functions.

Real world impact of faster computations in linear algebra

- Scientific simulations
  - Weather
  - Physics engines
- Machine Learning
  - Matrix multiplication in training models
- Cryptography and image processing
- Performance gains with matrix size and process count

What’s happening under the hood?

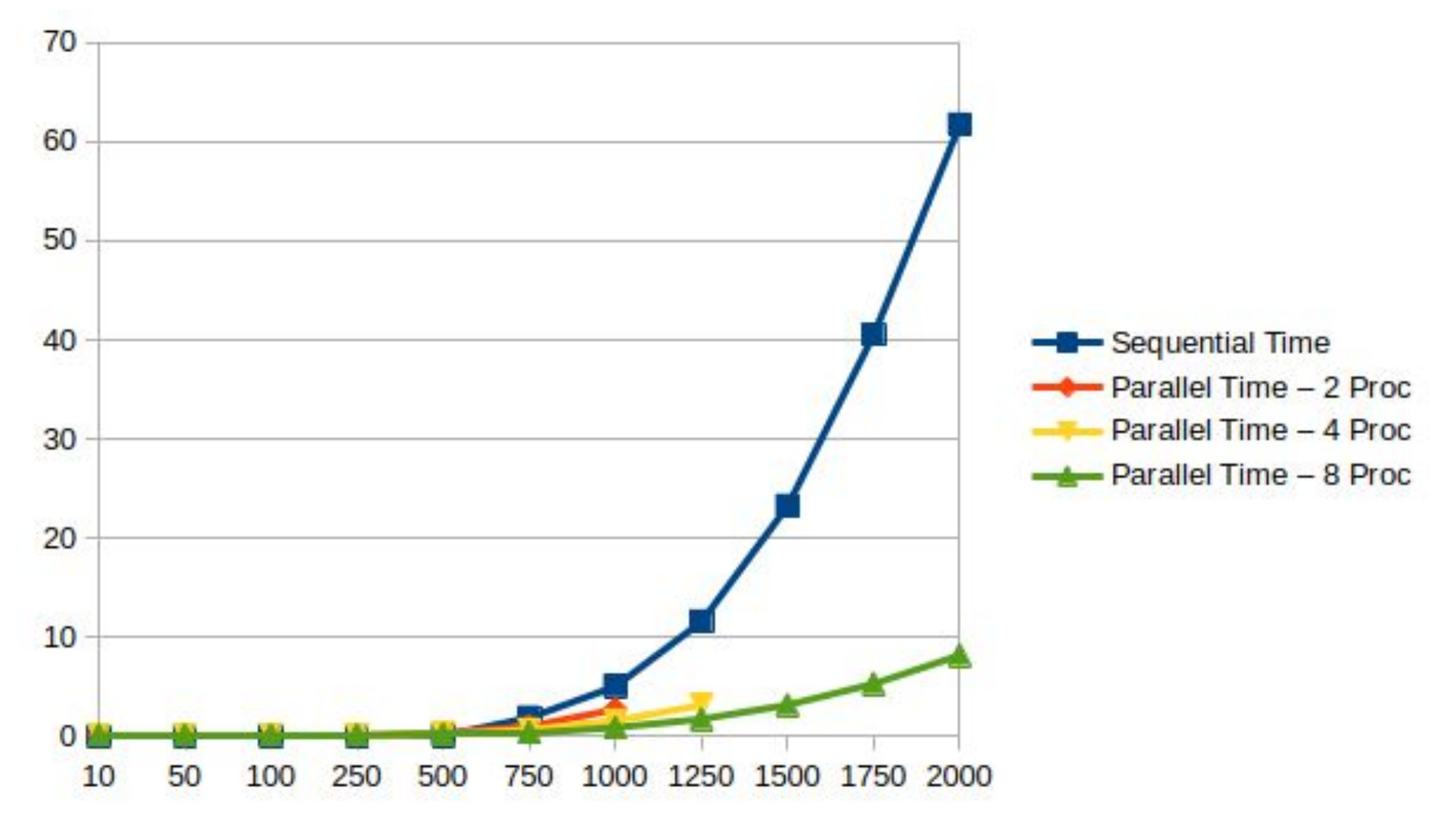
- Matrix multiplication:
  - C = A * B
- LU Decomposition:
  - A = L * U
- REF, RREF
- O(n^3), great for parallelism

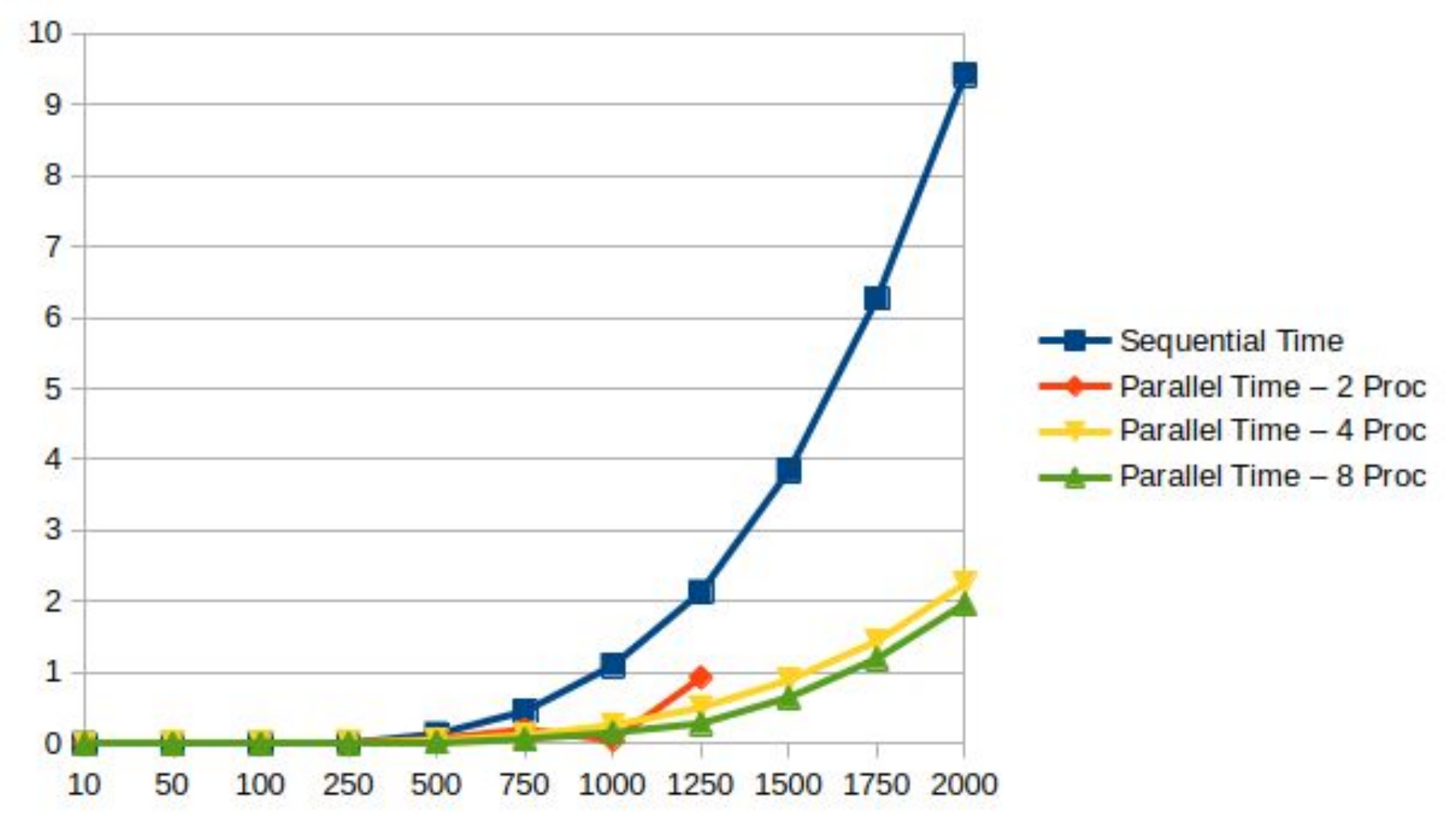
How parallel optimization works

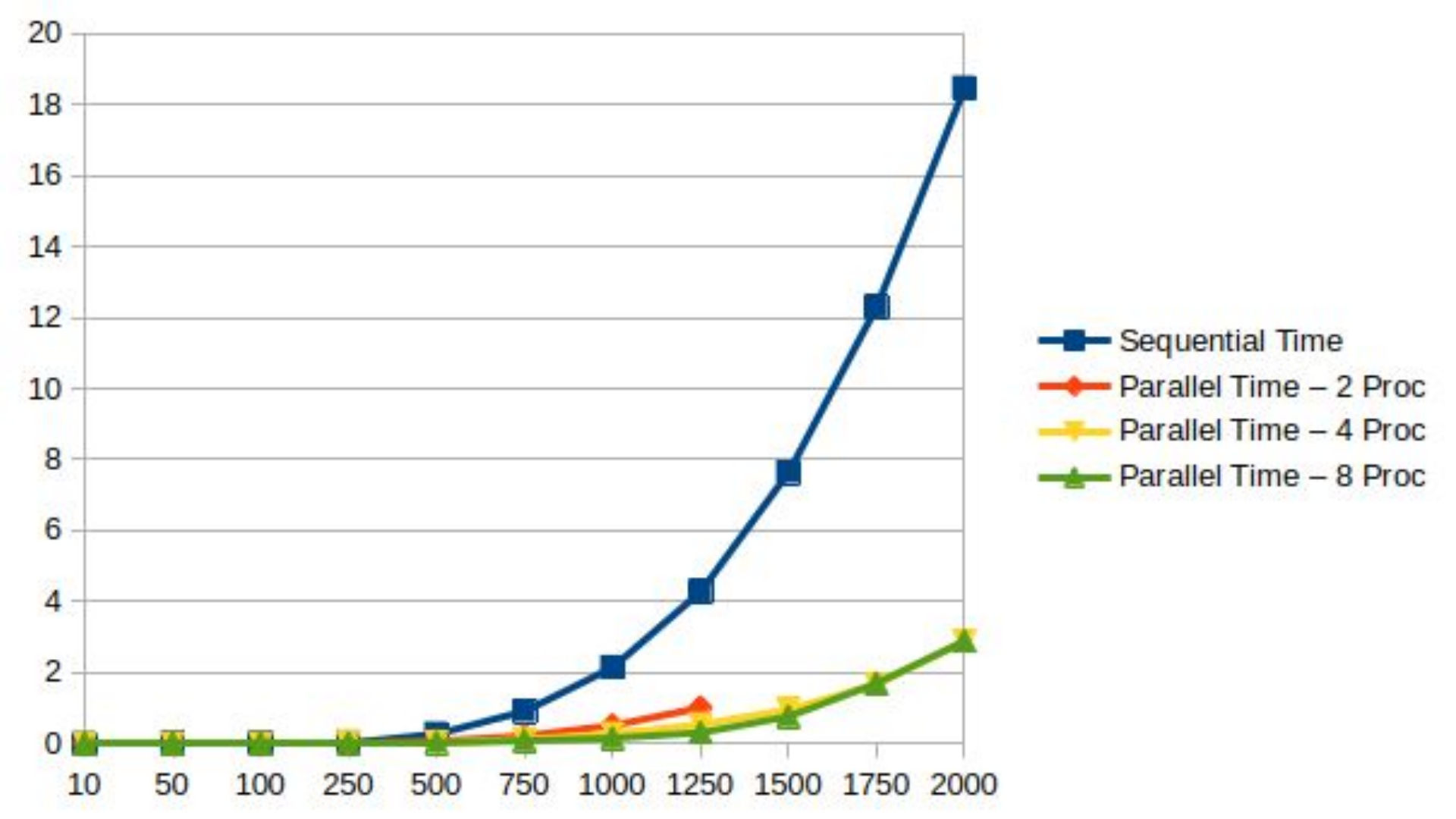
- Use MPI_Scatter, MPI_Gather to split / join matrix rows
- Each rank processes a subset independently
- Synchronization done with MPI_Barrier, MPI_Bcast
- Load balancing is the key to performance

How parallel optimization works

- Tested on matrices from sizes 10x10 -> 2000x2000
- Parallel version is significantly faster with the larger size and more cores


70
60
50
40
30
20
10
0
10
50
100
250
500
750
1000
1250
1500
1750
2000
Sequential Time
Parallel Time – 2 Proc
Parallel Time – 4 Proc
Parallel Time – 8 Proc


10
9
8
7
6
5
4
3
2
1
0
10
50
100
250
500
750
1000
1250
1500
1750
2000
Sequential Time
Parallel Time – 2 Proc
Parallel Time – 4 Proc
Parallel Time – 8 Proc


20
18
16
14
12
10
8
6
4
2
0
10
50
100
250
500
750
1000
1250
1500
1750
2000
Sequential Time
Parallel Time – 2 Proc
Parallel Time – 4 Proc
Parallel Time – 8 Proc

Takeaways

- Parallelism drastically improves speed for large problems
- OpenMPI is effective even on modest hardware
- For the future, I could implement more functions, such as methods to find eigenvalues and / or eigenvectors