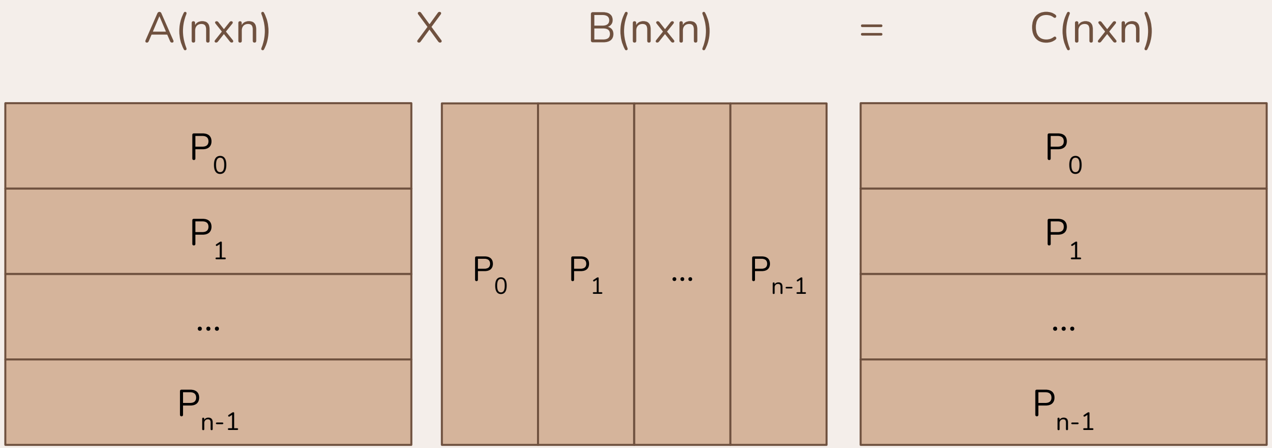
Group Assignment 1

Joshua Jenkins, Peter Nguyen, Grace Hsieh,
Pablo Martinez Castro

Algorithm Design: Problem 1

- Matrix Multiplication for Square (nxn) Matrices
  - 1D Layout



Results: Runtime

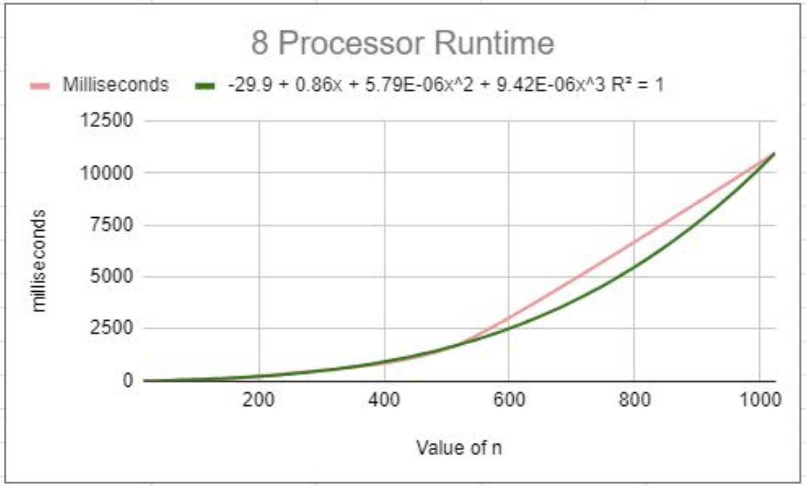
Run time (seconds)	P = 1	P = 2	P = 4	P = 8
n = 100	0.003	3.123	0.268	0.0656*
n = 1000	3.572	2.602	1.384	1.256
n = 5000	778.339	305.047	560.1	**1181.91****
n = 10000	5618.92	2415.694	5958.35	**9429.149****

*Calculated Value; 100 % 8 != 0

**HPC Context Switching

8 Processor Runtime Analysis - Estimation

n	milliseconds
8	0.618517
16	1.83251
32	3.76017
64	16.3313
128	58.191
256	387.012
512	1667.43
1024	10976.3

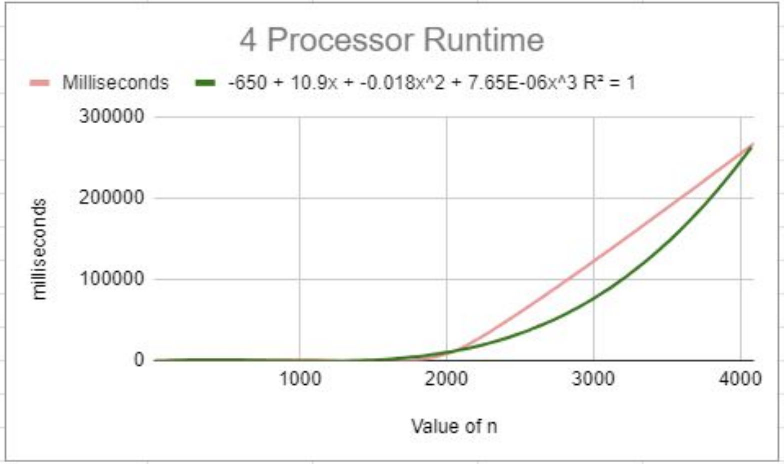


8 Processors			
nxn	Milliseconds	Seconds	Minutes
100	65.5779	<1 Second	<1 Minute
1000	10255.89	10.25589	<1 Minute
5000	1181914.85	1181.91485	19.69858083
10000	9429149.1	9429.1491	157.152485

Based on:
-29.9 + 0.86x + 5.79E-06x^2 + 9.42E-06x^3
(line of best fit)

4 Processor Runtime Analysis - Estimation

n	milliseconds
4	0.05146
8	0.027399
16	0.032998
32	0.067823
64	0.363247
128	4.49998
256	23.6028
512	214.537
1024	1378.9
2048	11427.9
4096	267636

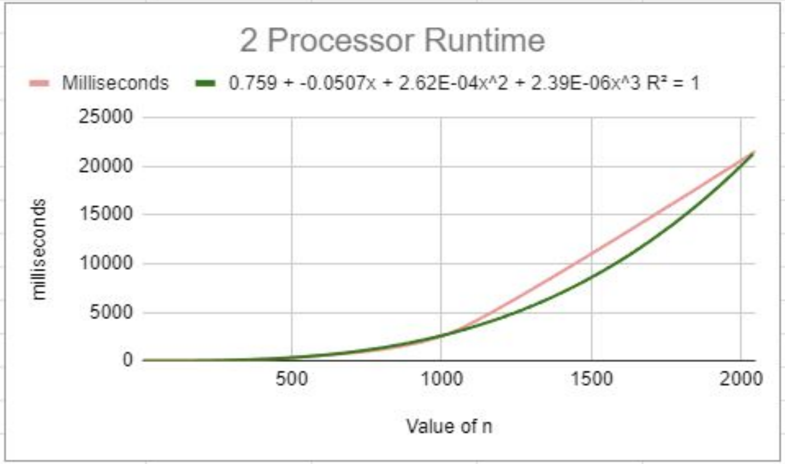


4 Processors			
nxn	Milliseconds	Seconds	Minutes
100	267.65	0.26765	<1 Minute
1000	1324.320484	1.324320484	<1 Minute
5000	560100	560.1	9.335
10000	5958350	5958.35	99.30583333

Based on:
-650 + 10.9x + -0.018x^2 + 7.65E-06x^3
(line of best fit)

2 Processor Runtime Analysis - Estimation

n	milliseconds
2	0.017772
4	0.009123
8	0.009576
16	0.016966
32	0.073625
64	0.536466
128	5.13548
256	40.0838
512	367.397
1024	2789.51
2048	21527
4096	530133

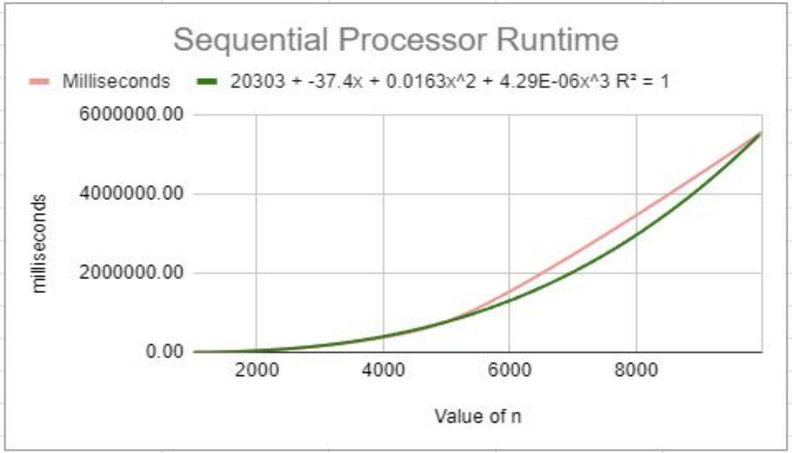


2 Processors			
nxn	Milliseconds	Seconds	Minutes
100	3.123411375	<1 Second	<1 Minute
1000	2602.059	2.602059	0.04336765
5000	305047.259	305.047259	5.084120983
10000	2415693.759	2415.693759	40.26156265

Based on:
0.759 + -0.0507x + 2.62E-04x^2 + 2.39E-06x^3
(line of best fit)

Sequential Runtime Analysis - Estimation

n	milliseconds
100	3.18
1000	3572.38
5000	778339
10000	5572350



1 Processor				
nxn	Milliseconds	Seconds	Minutes	Hours
100	3.18	<1 Second	<1 Minute	<1 Hour
1000	3572.38	3.57	<1 Minute	<1 Hour
5000	778339	778.34	12.97	<1 Hour
10000	5572350	5572	92.87	1.55

Based on:
20303 + -37.4x + 0.0163x^2 + 4.29E-06x^3
(line of best fit)

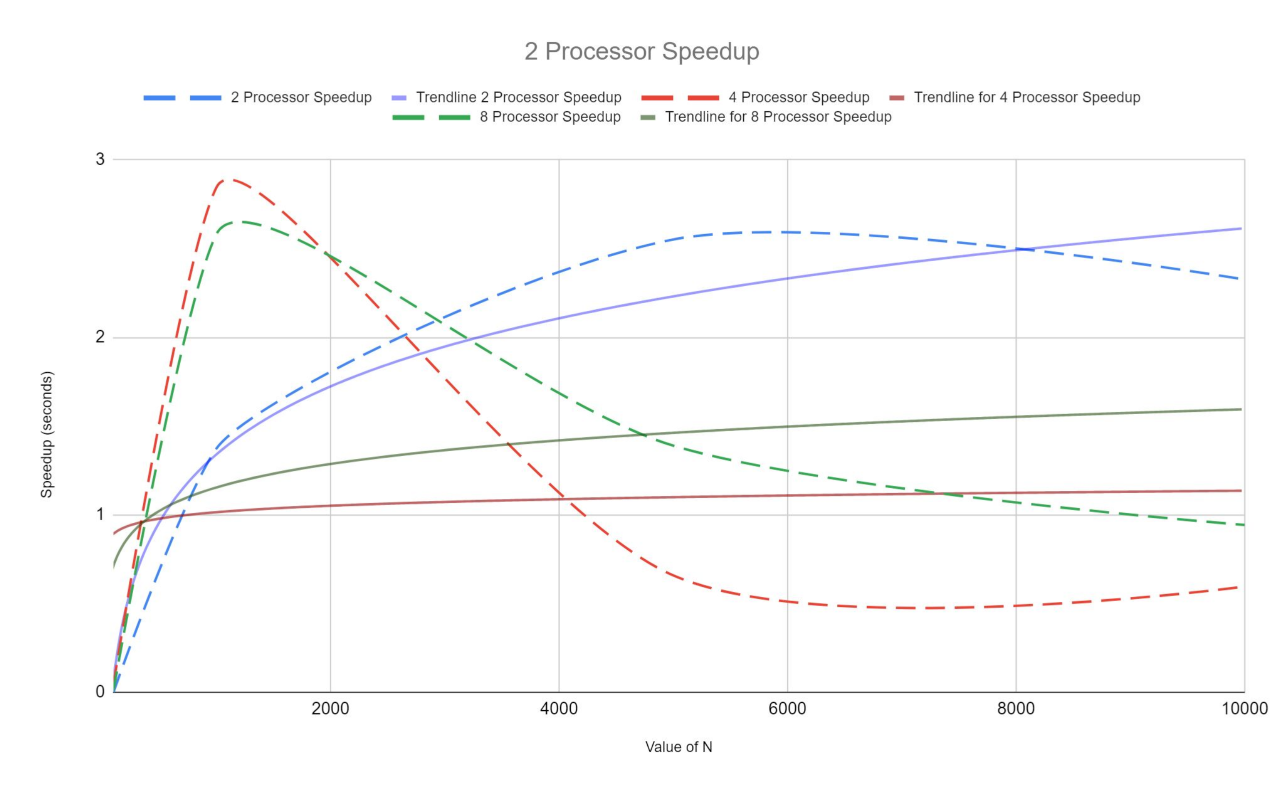
Results: Speedup

Speedup = Sequential Time / Parallel Time

Speedup (S)	**P = 2**	**P = 4**	**P = 8**
n = 100	0.00096	0.0112	0.0457
n = 1000	1.373	2.581	2.844
n = 5000	2.551	1.390	**0.659***
n = 10000	2.326	0.943	**0.596***

*HPC Context Switching

Results: Speedup (cont.)



Error Analysis

1. **HPC context switching**
   - One error we saw occur was **context switching**.
   - This caused our program to produce incorrect runtimes for some of our processes.
   - This is due to multiple people using the HPC at the same time.
2. **Line of best fit not accurate**
   - Another error we experienced is the calculation of runtimes not being calculated completely accurately after information was gathered.
   - This was due to using the trendline equations to calculate runtime at certain values of n.
3. **Potential Human Error**
   - Lastly, human error could have occurred. Recognizing and mitigating human error was essential for ensuring the validity of our computational analyses.

Algorithm Design: Problem 2

- A(m x n), B(n x q), C(m x q) - where m ≠ n ≠ q (different sized matrices).
- **Data Distribution**
    - Divide A and B into non-uniform segments, based on their sizes and the number of processors
        - If m = rows of A, q = columns of B, and p = number of processors, each processor would handle m/p rows and q/p columns
    - Processor 0 will handle a larger portion of A and B, will also calculate the final result
        - Each processor p (excluding processor 0), will receive a portion of A and B
    - If the number of rows or column is not divisible evenly by the number of processors:
        - Distribute the remainder

Algorithm Design: Problem 2 Cont.

Algorithm Design:

1. Initialize the MPI environment.
   - `INITIALIZE MPI;`
2. Determine the total number of MPI processes and get the ID of the current process.
   - `DETERMINE MPI_WORLD_SIZE (p_total) AND RANK (pid);`
3. Root process initializes matrices A[m][n] and B'[n][q] (where B' is the transpose of B[n][q]).
   - `IF PID == 0 THEN INITIALIZE A(m,n),[B'<-TRANSPOSE(B[n][q])];`
4. Distribute matrix segments to all processes.
   - `localA, localB', p_total, pid <- DISTRIBUTE_MATRICES(A,B');`
5. Perform 1D matrix multiplication on the different segments.
   - `localC <- MM_1D_DISTRIBUTED(localA,localB',localM,localN,localQ);`
6. Gather all processes and assemble matrix C[m][q]
   - `C <- ASSEMBLE_MATRIX(GATHER_RESULTS(ALL localC's ACROSS ALL pid));`

Problem 2: DISTRIBUTE_MATRICIES()

```
FUNCTION DISTRIBUTE_MATRICES(in[A, B'] -> out[m, n, q, p_total, pid])
    IF pid == 0 THEN
        // Root process sends data to each process
        FOR process FROM 1 TO p_total - 1 DO
            SEND SEGMENT OF A TO process
            SEND SEGMENT OF B' TO process
        END FOR
    ELSE
        // Processes receive their information
        local_A <- RECEIVE SEGMENT OF A
        local_B' <- RECEIVE SEGMENT OF B'
    END IF
END FUNCTION
```

Communication Cost:

$$O(p_{total} * t_s + (m * n) * t_w)$$

Problem 2: `MM_1D_DISTRIBUTED()`

```
FUNCTION MM_1D_DISTRIBUTED(in[localA, localB'] -> out[localM, localN, localQ])
    localC <- INITIALIZE_MATRIX(localM, localQ)
    FOR i FROM 0 TO localM - 1 DO
        FOR j FROM 0 TO localQ - 1 DO
            sum <- 0
            FOR k FROM 0 TO localN - 1 DO
                sum <- sum + (localA[i][k] * localB'[j][k])
            END FOR
            localC[i][j] <- sum
        END FOR
    END FOR
    RETURN localC
END FUNCTION
```

Problem 2: GATHER_RESULTS()

```
FUNCTION GATHER_RESULTS(in[localC, m, q] -> out[C, p_total, pid])
    IF pid == 0 THEN
        // Root process gathers all segments of C
        C <- INITIALIZE_MATRIX(m, q)
        FOR process FROM 0 TO p_total - 1 DO
            RECEIVE SEGMENT OF C FROM process
        END FOR
    ELSE
        // Processes send result back to root processor
        SEND localC TO pid 0
    END IF
END FUNCTION
```

Communication Cost:

$$O(p_{total} * t_s + (m * q) * t_w)$$

Communication Cost & Runtime Analysis

Overall Communication Costs = $O(p_{total} * t_s + (m * n) * t_w) + O(p_{total} * t_s + (m * q) * t_w)$

= $O(p_{total} * t_s + (m * (n + q)) * t_w)$

Overall Runtime = $O(\frac{m * n * q}{p_{total}} + p_{total} * t_s + (m * (n + q)) * t_w)$

Thank you!