MAXIMUM SUM SUBSEQUENCE

Mahak Mukhi

Mohana Bhunekar

WHAT DOES THE PROBLEM SAY?

- Determine a subsequence of data set that sums to the maximum value with respect to any other subsequence of the data set.

- More formally, if we are given a sequence $X = \langle x_0, x_1, x_2, \ldots, x_n \rangle$ we are required to find a set of indices u and v, $u \leq v$, such that the subsequence $\langle x_u, x_{u+1}, \ldots, x_v \rangle$ has the largest possible sum, $x_u + x_{u+1} + \ldots + x_v$, among all subsequences of *X*.

- This problem is non-trivial only if the given sequence has both positive and negative values

IS THAT EVEN A THING?

- Protein and DNA sequence analysis
  - To locate biologically meaningful segments, e.g. conserved segments , GC-rich regions, non-coding RNA genes, and transmembrane segments.
  - A common approach is to assign a value to each residue, and then look for consecutive subsequences with high sum or average.

- Traffic Monitoring
  - Example: Say for a bridge, given the numbers of vehicles entering and exiting at various passes, you can determine the busiest routes.

ARCHITECTURE

- We worked this problem on a Linear Array
- But, we don't have a Linear Array!
  - But we always can simulate the same behavior virtually.
  - We wrote the MPI code such that the program behaves as if it were running on a Linear Array.
- It comes with a cost though.
  - The communication time is more than what is intuitive.

ALGORITHM

- The algorithm that I am considering to implement is from "Algorithms Sequential & Parallel: A Unified Approach" by Russ Miller and Laurence Boxer.
- First compute the parallel prefix sums $S = \{p_0, p_1, \ldots, p_{n-1}\}$ of $X = \{x_0, x_1, \ldots, x_{n-1}\}$, where $p_i = x_0 \otimes \ldots \otimes x_i$.
- Next, compute the parallel postfix maximum of S so that for each index i, the maximum pj, $j \geq i$, is determined, along with the value j.
- Let m_i denote the value of the postfix-max at position i, and let a_i be the associated index, i.e., $p_{ai} = \max \{p_i, p_{i+1}, \ldots, p_{n-1}\}$.

ALGORITHM CONTINUED

- Next, for each i, compute $b_i = m_i - p_i + x_i$, the maximum prefix value of anything to the right minus the prefix sum plus the current value.

- Finally, the solution corresponds to the maximum of the b_i's, where u is the index of the position where the maximum of the b_i's is found and $v = a_u$.

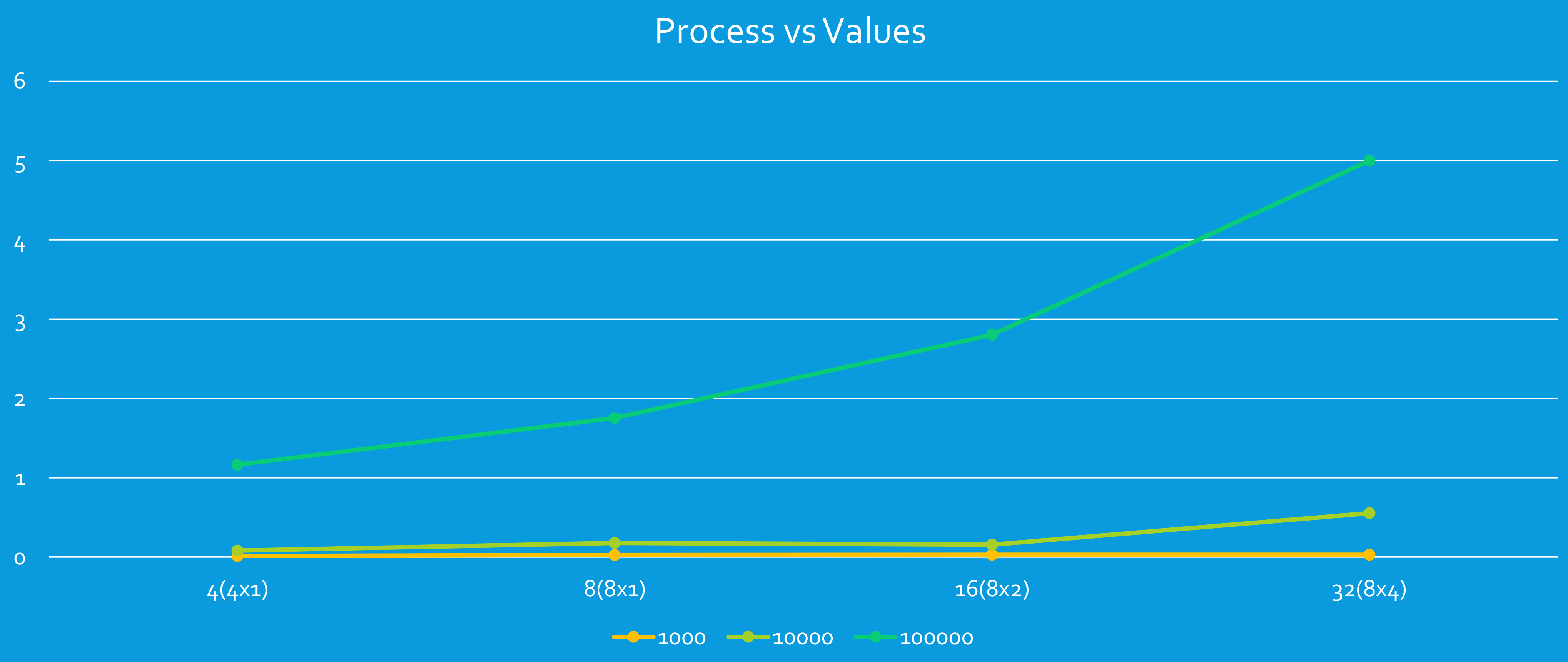
TIME COMPLEXITY

- This algorithm runs in $\Theta(n)$ time on a Linear Array.
- And the optimal cost of $\Theta(n)$ is achieved with $n^{1/2}$ processors.

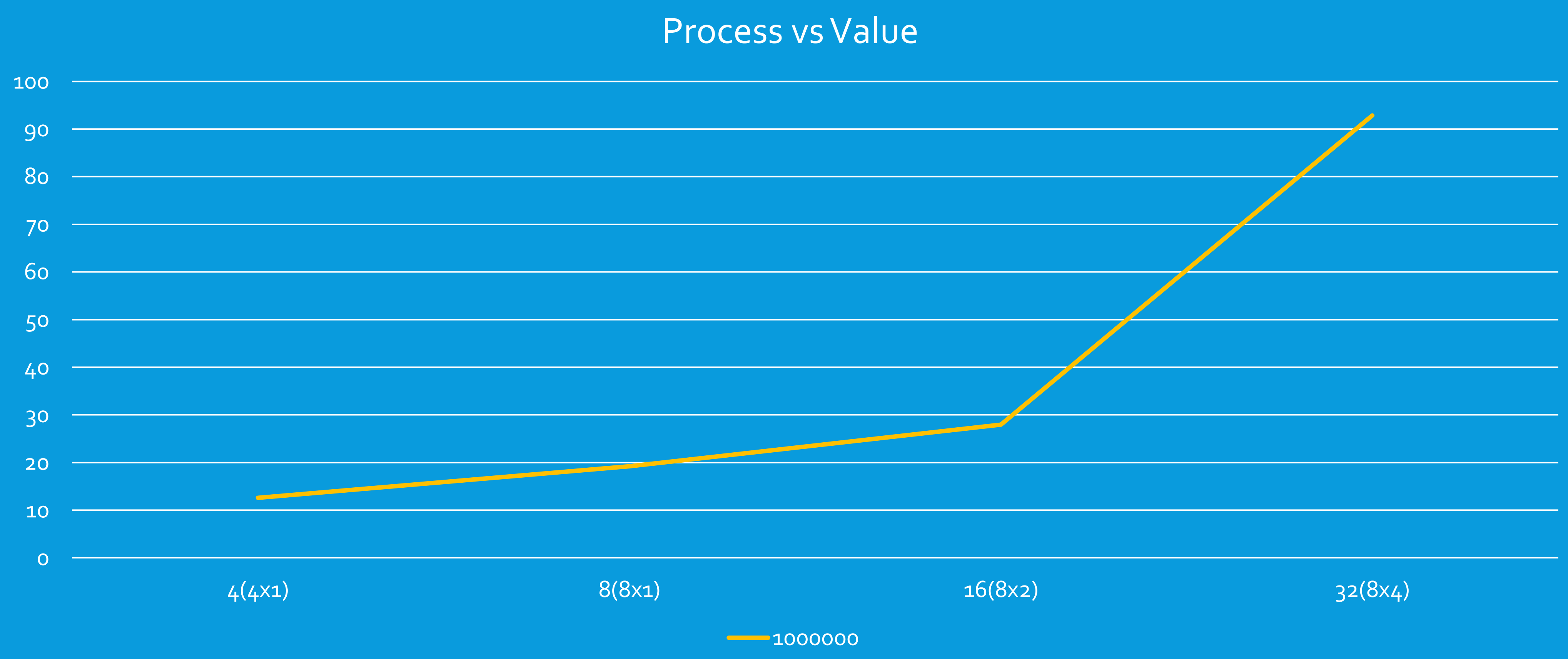
RUNNING TIME

Process\Values	1000	10000	100000	1000000
4(4x1)	0.0142	0.0832	1.1636	12.5648
8(8x1)	0.0271	0.1795	0.7540	19.2160
16(8x2)	0.0301	0.1571	2.8025	27.9722
32(8x4)	0.0306	0.5545	6.7950	92.8327

RUNNING TIME



RUNNING TIME



CONCLUSION

- Linear Array is not a very efficient architecture to go with.
- With the increase in no of processes the running time increases too because of the communication diameter.
- After a while the communication time completely overshadows the execution time.

FUTURE GOALS

- For the rest of the semester, we would like to conduct experiments with even larger data.
- In the next phase we plan to work this out on various other architectures as well.
- A comparative analysis would be great!

ACKNOWLEDGEMENT AND REFERENCES

- "Algorithms Sequential & Parallel: A Unified Approach" by Russ Miller and Laurence Boxer
- http://wordaligned.org/articles/the-maximum-subsequence-problem
- "Genomic Sequence Analysis: A Case Study in Constrained Heaviest Segments" Kun-Mao Chao

LET’S TALK!

- Session open for discussion

Thank you