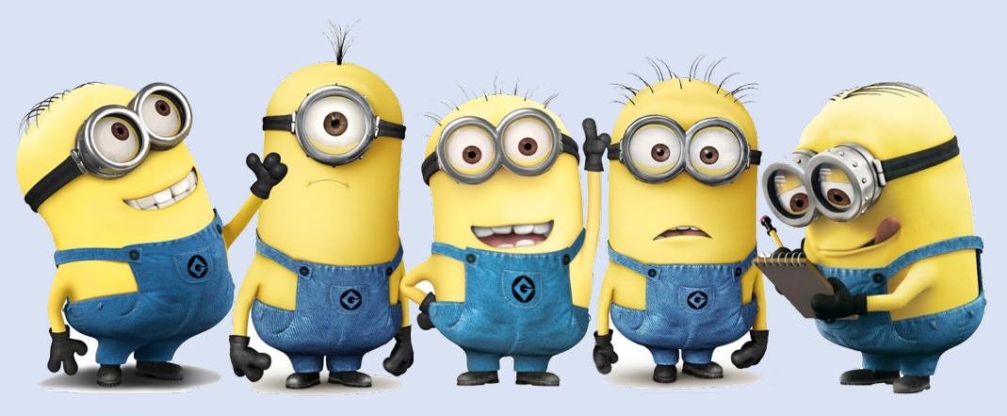
Nanopore Sequencing Technology and Tools:

Computational Analysis of the Current State, Bottlenecks and Future Directions

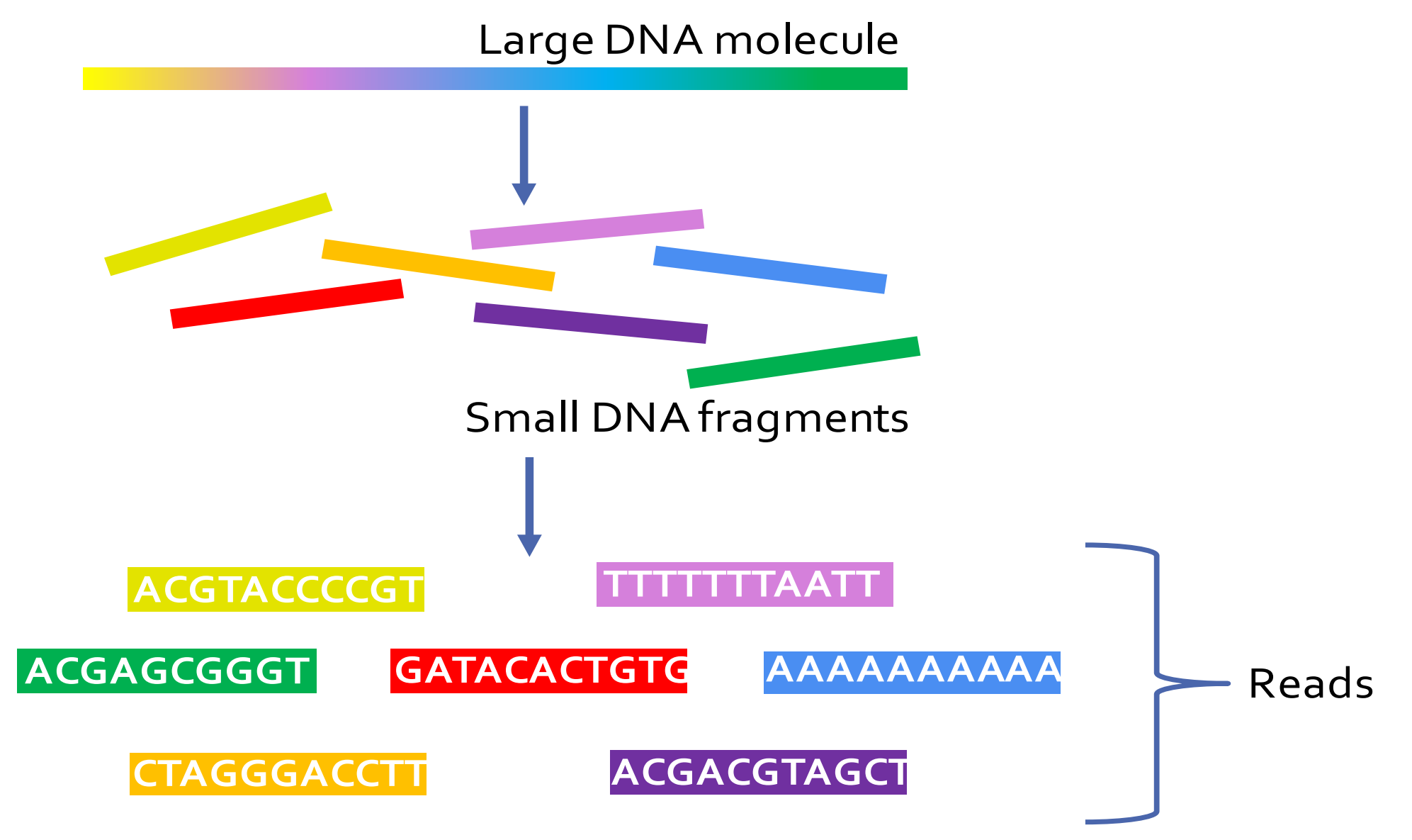
Damla Senol[1], Jeremie Kim[1,3], Saugata Ghose[1], Can Alkan[2] and Onur Mutlu[3,1]

[1] Carnegie Mellon University, Pittsburgh, PA, USA [2] Bilkent University, Ankara, Turkey [3] ETH Zürich, Zürich, Switzerland



Genome Sequencing

Genome sequencing is the process of determining the order of the DNA sequence in an organism's genome.



Long Read Analysis

Long reads
- Sequences with thousands of bases
- Sequences with higher error rates
- Suitable for **de novo assembly**

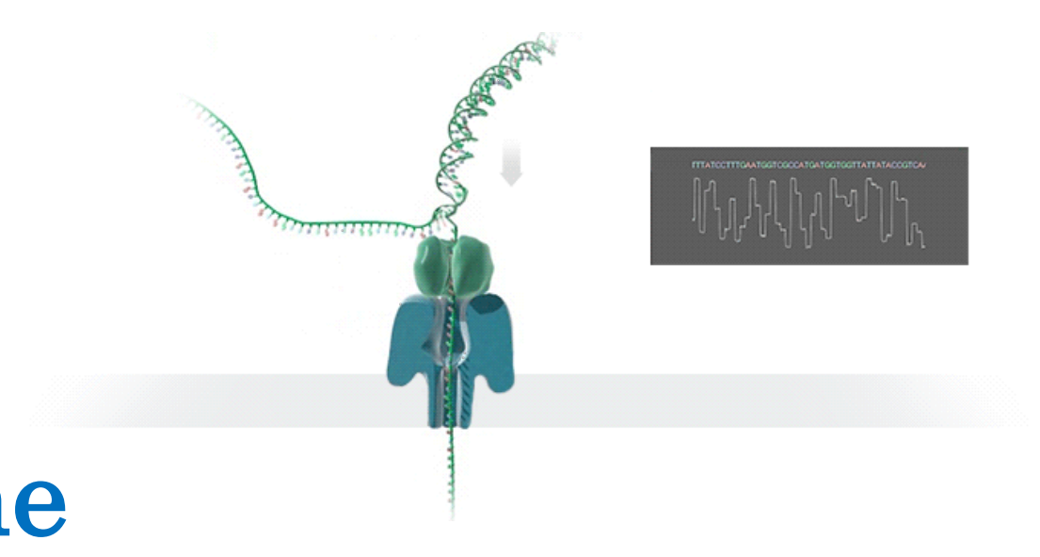
De novo assembly is the method of
- Merging the reads in order to construct the original sequence
- Without the aid of a reference genome

Assembly quality can be improved by using longer reads, since they can cover long repetitive regions.

Nanopore Sequencing

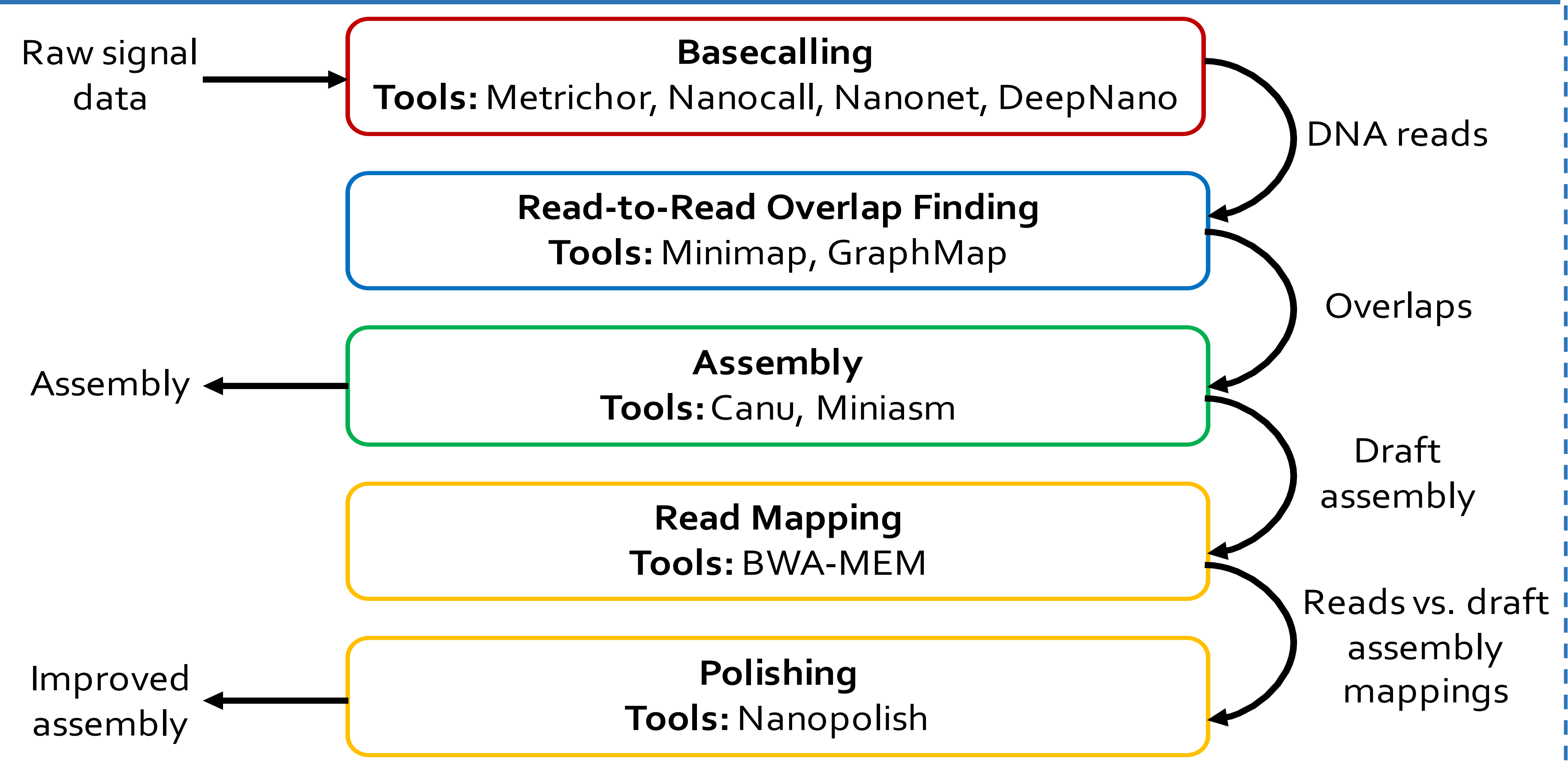
Nanopore sequencing is an emerging DNA sequencing technology.

- Long read length
- Portable and low cost
- Produces data in real-time



Nanopore sequencers rely solely on the electrochemical structure of the different nucleotides for identification and measure the change in the ionic current as long strands of DNA (ssDNA) pass through the nano-scale protein pores.

Pipeline and Current Tools



Problem & Our Goal

Problem

The tools used for nanopore sequence analysis are of critical importance in order to increase the accuracy of the whole pipeline to take better advantage of long reads, and increase the speed of the whole pipeline to enable real-time data analysis.

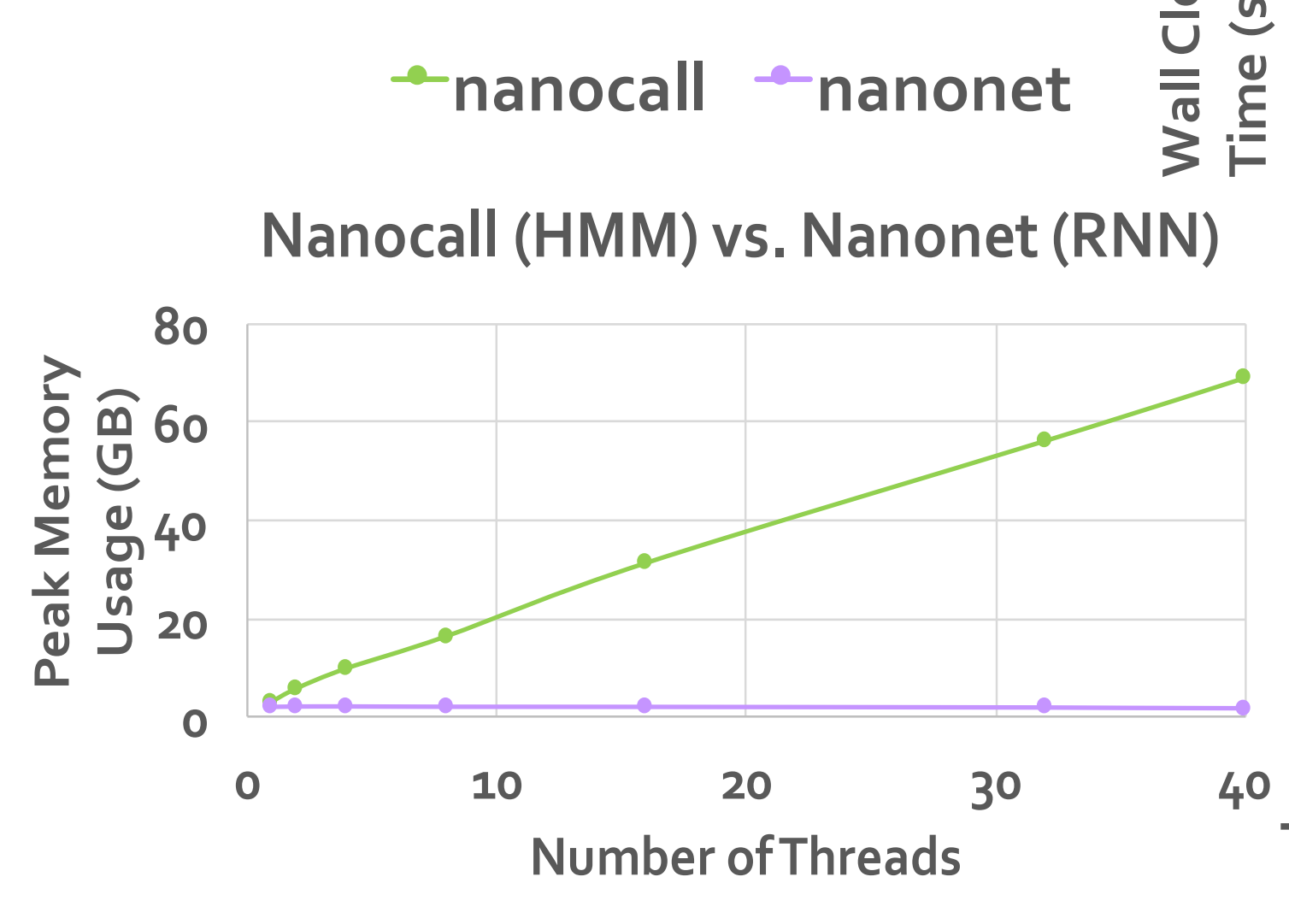
Our Goal
- Comprehensively analyze current publicly available tools in the whole pipeline for nanopore sequence analysis, with a focus on understanding their advantages, disadvantages, and performance bottlenecks.
- Provide guidelines for determining the appropriate tools for each step of the pipeline.

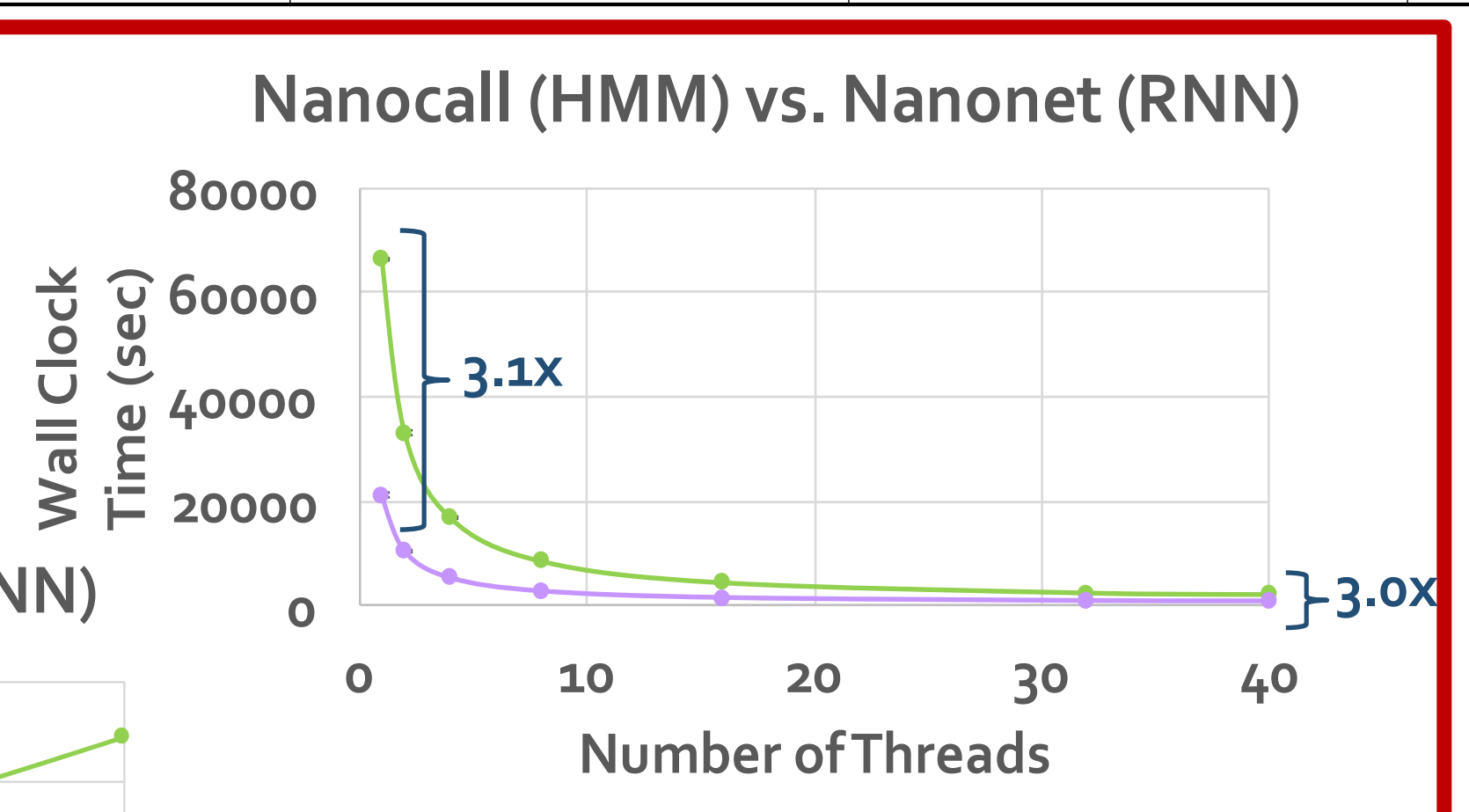
Results and Analysis

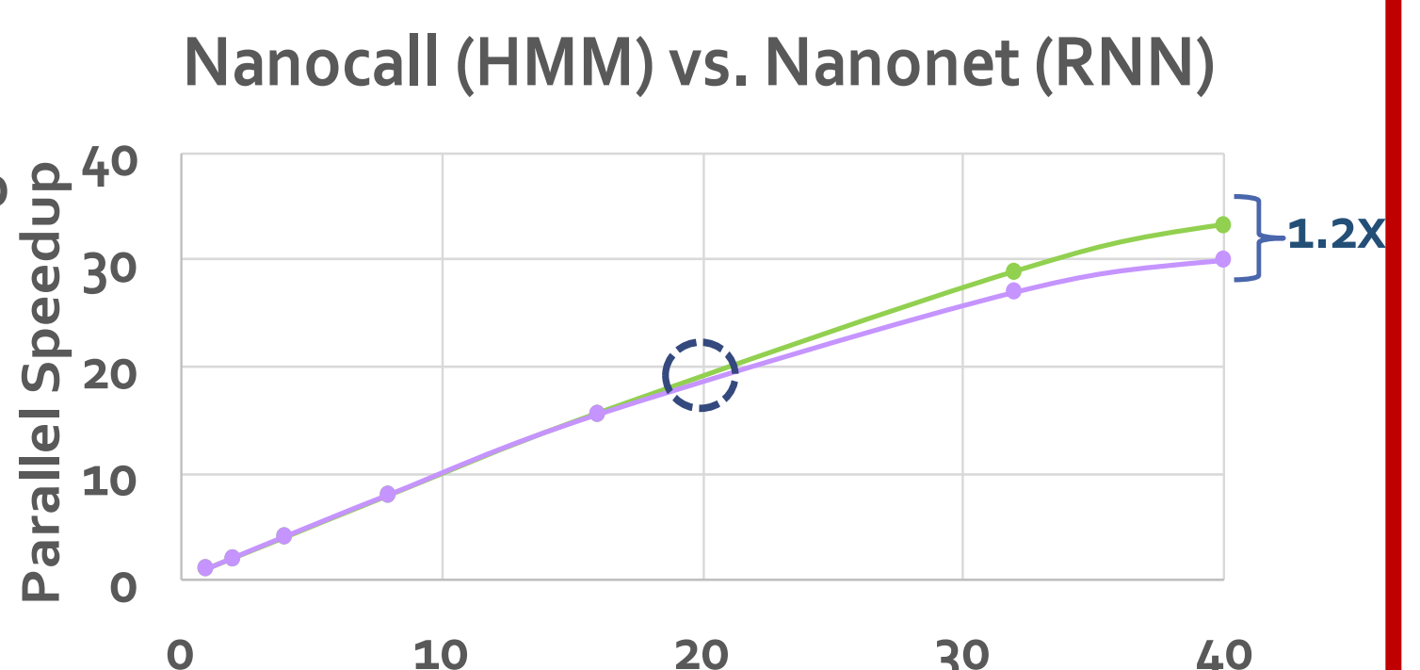
	Step 1 Wall Clock Time (h:m:s)	Step 1 Memory Usage (GB)	Step 2 Wall Clock Time (h:m:s)	Step 2 Memory Usage (GB)	Step 3 Wall Clock Time (h:m:s)	Step 3 Memory Usage (GB)	Number of Contigs	Identity (%)	Coverage (%)
Metrichor + Canu			–	–	44:12:31	5.76	1	98.04	99.31
Metrichor + Minimap + Miniasm	–	–	2:15	12.30	1:19	1.96	1	85.00	94.85
Metrichor + Graphmap + Miniasm			6:14	56.58	1:05	1.82	2	85.24	96.95
Nanonet + Canu			–	–	11:32:40	5.27	1	97.92	98.71
Nanonet + Minimap + Miniasm	17:52:42	1.89	1:13	9.45	33	0.69	1	85.50	93.72
Nanonet + Graphmap + Miniasm			3:18	29.16	32	0.65	1	85.36	92.05
Nanocall + Canu			–	–	–	–	–	–	–
Nanocall + Minimap + Miniasm	47:04:53	37.73	1:15	12.19	20	0.47	5	80.53	96.80
Nanocall + Graphmap + Miniasm			5:14	56.78	16	0.30	3	80.52	95.43
Deepnano + Canu			–	–	1:15:48	3.61	106	92.63	154.07
Deepnano + Minimap + Miniasm	23:54:34	8.38	1:50	11.71	1:03	1.31	1	82.37	91.62
Deepnano + Graphmap + Miniasm			5:18	54.64	58	1.10	1	82.39	91.60

OBSERVATION 1: **Basecalling with Recurrent Neural Networks** performs better than **basecalling with Hidden Markov Models** in terms of accuracy, speed, and memory usage. However, it has scalability limitations due to **data sharing** between threads.

OBSERVATION 2: Sharing the computation of a read between parallel threads provides a constant and **low memory usage**, but data sharing between multiple sockets **degrades the parallel speedup** when number of threads reaches higher values.







OBSERVATION 3: Storing **minimizers** instead of all k-mers does not affect the accuracy of the whole pipeline. However, Minimap has a **lower memory usage** and **higher speed** than GraphMap, since computation is decreased by **shrinking the size of the dataset** that needs to be considered.

OBSERVATION 4: Canu, an assembler with error correction, produces **high-quality assemblies** but is **slow** compared to Miniasm, an assembler without error correction.

Miniasm is suitable for **fast initial analysis**, and the quality of its assembly can be increased with an additional polishing step.

OBSERVATION 5: Nanopolish is compatible only with reads basecalled by Metrichor.

Polishing the draft assembly generated with Canu takes **5h52m** and increases the accuracy **from 98.04% to 99.46%**.

Polishing the draft assembly generated with Miniasm takes **5d2h54m** and increases the accuracy **from 85.00% to 92.31%**.