PA-Bench: A framework for benchmarking pairwise aligners

Ragnar Groot Koerkamp
ragnar.grootkoerkamp@inf.ethz.ch
@curious_coding

Daniel Liu
daniel.liu02@gmail.com
@daniel_c0deb0t

ETH zürich UCLA

Code and references:
curiouscoding.nl/notes/pabench-poster/

Introduction

- Pairwise alignment using dynamic programming accounts for a significant fraction of the time spent in many bioinformatics pipelines, and many advancements are still being made.
- Alignment has diverse use-cases and different types of data:
  - Read-to-read, read-to-genome, and genome-to-genome alignment
  - Small indels, structural variation, repetitive regions.
  - Different sequencing technologies (Illumina, PacBio, ONT).
- Generally, **benchmarking is hard**. Most papers have ad-hoc benchmarks that do not fairly capture the diversity of data faced by users, nor guarantee a fair and stable results.
- A scalable and comprehensive benchmark suite for aligners is missing.

Methods

PA-Bench is a rigorous benchmarking framework for pairwise alignment that:

- allows comparing existing and new methods on a wide variety of data;
- helps users to **find the right algorithm** for their data;
- provides a **uniform API** `pa-wrapper` via the `Aligner` trait.
- has a single **binary** `pa-bin` to call existing aligners.

The main component is a tool to orchestrate a set of jobs for benchmarking different aligners on different datasets.

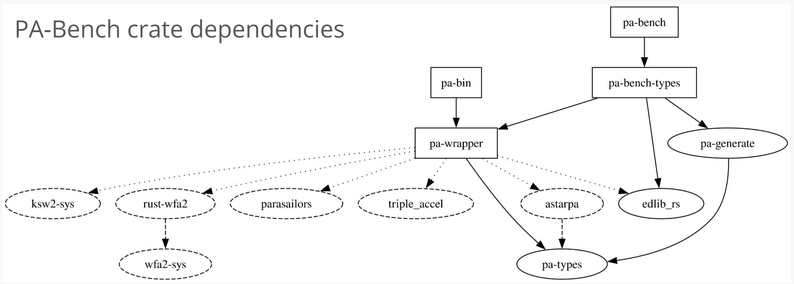
Currently supported are:

- **Parasail** [Daily 2016]
- **Edlib** [Šošić and Šikić 2017]
- **KSW2** [Li 2018, Suzuki and Kasahara 2018]
- **WFA/BiWFA** [Marco-Sola et. al. 2021, 2023]
- **Triple Accel** [Liu 2022]
- **Block Aligner** [Liu and Steinegger 2023]
- **A*PA and A*PA2** [Groot Koerkamp and Ivanov 2024, Groot Koerkamp 2024]

New aligners can be added easily by implementing the trait.

Both exact and approximate methods are supported.

PA-Bench crate dependencies



Input

PA-Bench's main input is an `experiment.yaml` file that allows the user to specify a number of parameters:

- the **time** and **memory limit** to use;
- the **datasets** to run on: a file, URL, or generator parameters,
- whether to compute a **traceback**,
- the **cost model** to use,
- the **aligners** to run.

Jobs are created from the cartesian product of these parameters.

PA-Bench can run multiple **jobs in parallel** and **caches results**, so that only new jobs are run when an experiment is modified.

```yaml
- time_limit: 1h
  mem_limit: 32GiB
  datasets:
    - !Generated
      seed: 31415
      error_rates: [0.05]
      error_models: [Uniform]
      lengths: [1000, 10000, 100000, 1000000]
      total_size: 10000000
  traces: [true]
  costs: [{ sub: 1, open: 0, extend: 1 }]
  algos:
    - !Edlib
    - !Wfa
```

Output

Benchmarking is done by running one job at a time. Each job:

- is run with a **fixed CPU frequency**;
- is **pinned** to its own thread;
- is run with **high priority** (low *niceness*)
- optionally has a **memory** or **time limit**, set using *rlimit*.

Many statistics are collected, such as:

- Wall, system and user time usage.
- Peak and increment of memory usage.
- The CPU frequency when the job started and finished.
- Dataset characteristics like sequence lengths, edit distances, gap lengths.

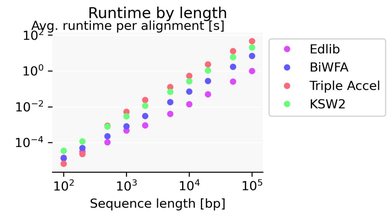
Results are automatically verified:

- The score of the returned CIGAR must match the returned score.
- For exact methods, the score must match the optimal alignment.
- The CPU frequency at start and end must equal the set frequency.

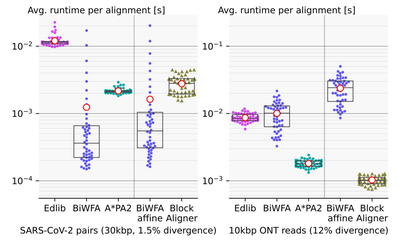
All results (statistics and outputs) are then written to a JSON file that can be parsed and plotted using provided python notebooks.

Results

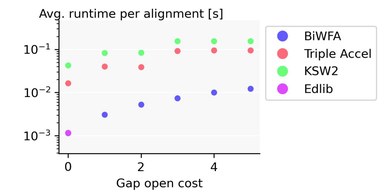
- PA-Bench can benchmark jobs running from milliseconds to hours with **high accuracy and reproducibility**.
- It is easy to compare aligners across various dimensions.
- Runtime scaling with length on random data, for fixed error rate:



- Runtime distribution on real data across unit cost (Edlib, BiWFA, and A*PA2) and affine cost (BiWFA-affine and Block Aligner) models. Note how the relative performance of aligners can vary widely between different types of sequences, and that Block Aligner is approximate.



- Runtime comparison for increasing gap-open cost (sub=extend=1)



- PA-Bench has proven useful for rapidly testing and benchmarking aligners, and it has been used in practice to run the benchmarks for A*PA/A*PA2.
- **Quick benchmark-to-plot feedback** increases productivity.
- The sanity checks on the returned CIGAR have found bugs both during the development of A*PA, and in already published code.

Conclusion

- PA-Bench is a convenient tool that **makes comparing pairwise aligners easy** for developers and users
- Currently, PA-Bench only supports benchmarking global alignment of DNA sequences. Future work include semi-global, local, and e.g. sequence-to-graph alignment, in addition to continuously updated datasets.