PxBLAT: An Efficient and Ergonomics Python Binding Library for BLAT

Yangyang Li[1] **and Rendong Yang**[1,✉]

[1] Department of Urology, Northwestern University Feinberg School of Medicine, Chicago, IL 60611

PxBLAT provides an ergonomic and efficient Python binding library for BLAST-like alignment tool (BLAT), designed to enhance the user experience and performance of genomic analysis tasks. By providing a pythonic interface to BLAT, PxBLAT simplifies the usage of BLAT, allowing incorporating its functionality directly into their Python-based bioinformatics workflows. Furthermore, A new parallelization and non-blocking model enables improved efficiency and intuitive user interface, thereby reducing the complexity of genomic data analysis tasks.

Software Libraries | Sequence Analysis | BLAT

Correspondence: *rendong.yang@northwestern.edu*

Introduction

The continual advancement of genome sequencing technologies has led to an exponential increase in available genomic data. Tools to analyze and manipulate these data have become critically important in both research and clinical contexts. BLAT (**?**) is a standout in the bioinformatics field for its capability to perform rapid genome sequence alignments. It offers a faster alternative to Basic Local Alignment Search Tool (BLAST) (**?**) for aligning DNA sequences to the human genome (**?**). Despite its widespread use and acceptance in the bioinformatics community, interfacing with BLAT can present challenges, particularly when integrating it within a broader Python-based analytical pipeline. Since BLAT is implemented by C programming language and only provides a variety of utilities with a command line interface (CLI). So far, we lack a holistic approach to enhance both the performance and the usability of BLAT.

On the other hand, Python's rise as a favored programming language in bioinformatics is well-documented, due to its ease of use, extensive libraries, and versatility (**?**). Various binding libraries have been developed to extend Python's reach into other computing languages, improving the flexibility and interoperability of bioinformatics tools. For instance, Biopython (**?**), one of the most prominent bioinformatics libraries, provides interfaces to tools like BLAST (**?**), Clustal (**?**), and others. Nonetheless, to date, no comprehensive Python binding library for BLAT has been developed.

Here we propose PxBLAT, a modern Python library designed to streamline and enhance the interaction with BLAT, thereby making it more efficient and ergonomic. PxBLAT serves as a bridge, bringing the high-performance capabilities of BLAT into the Python environment, which is widely regarded for its readability, simplicity, and extensive library support. By improving the usability of BLAT and seamless integration within Python, PxBLAT opens up new possibilities for efficient genomic analysis. We provide evidence of its performance improvements, demonstrate its ergonomic advantages, and discuss its potential applications in genomic research. The overarching aim of this work is to fill the observed gap by providing a Python binding library specifically tailored for BLAT, addressing both its efficiency and ergonomic concerns.

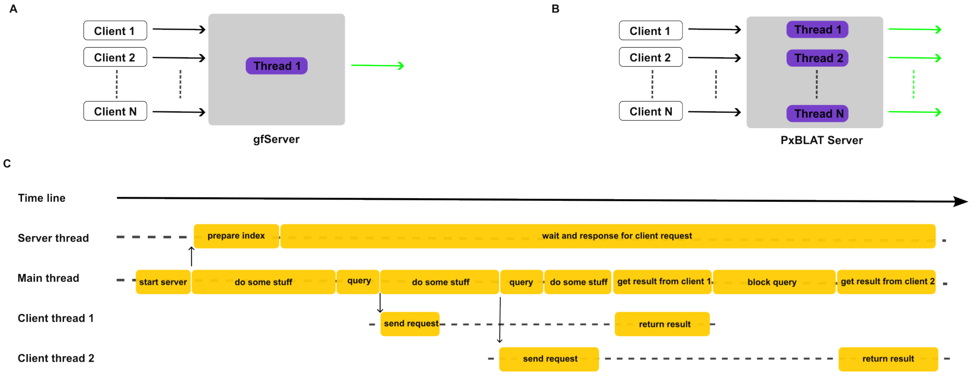

Figure 1. These are cells.

(**A**) This is a regular figure with a legend as a caption underneath. Inset: 3X zoom. Scale bar, 10 µm.

Implementation

The design of PxBLAT follows the pythonic principles of readability and simplicity. It is built to be intuitive, reducing the learning curve for users familiar with Python and bioinformatics tools. In line with this, we focused on minimizing the complexity while maximizing the usability and performance. Hence, We employ existing C codebase and reimplement utilities of BLAT(V37.1) suit including *faTwoBit*, *gfServer* and *gfClient* CLI using Cpp programming language (**?**). Then, we create PxBLAT via Pybind11 (**?**). This approach enabled direct access to the functions of the BLAT program without modifying its original source code, preserving the integrity and performance of BLAT while extending its capabilities.

Furthermore, PxBLAT eliminate the need for intermediate files, which allow all operations to be in memory, these tools can analyze and respond to data inputs almost instantaneously. Input and output file now become optional, which is more flexible. Meanwhile, PxBLAT reduce the need of system calls to interact with BLAT, which may prevent potential safe problem. It provides optional non-blocking operations as well, improving the overall efficiency. Importantly, PxBLAT allow fetching status of BLAT without manipulating log files, especially in concurrent environment. That means PxBLAT reduce possibility of data race, which is annoying issue in concurrent environment. PxBLAT also implement ergonomic features consisting of checking and waiting if server is ready for alignment, retrying different port if current port is occupied, using existing server if server is already started in current computing node. All these features have been tested and work fine in concurrent environment. Compared to *gfServer*, PxBLAT reimplement it using multiple threads model, which enables processing multiple client request concurrently.

Specifically, PxBLAT provides application programming interface (API) including *Server*, *Client* class and other free functions to reproduce BLAT CLI. *Server* and *Client* have same utilities as *gfServer* and *gfClient* respectively but more flexible and efficient. Besides, they own ergonomic features aforementioned. Free functions the library provides, for example, *start_server*, *query_server*, *status_server*, *fa2twobit* etc., serve as potential usage under different contexts. PxBLAT also offers same but more user-friendly CLI as BLAT suit. The library is well tested, and use type hints for all public classes and functions to ensure the quality and correctness via static analyzer. These type annotations also make PxBLAT more pleasant to use inside an Integrated Development Environment (IDE), where the function signatures can be suggested and corrected automatically.

```python
from pxblat.server import Server, Client

with Server() as server:
    client = Client()
    client.start()
    # do some stuffs that consuming time
    # and do not need query result,
    # then get query result
    ret = client.get()
```

Listing 1. Python example

Results

To validate PxBLAT's utility and effectiveness, we undertook a comprehensive evaluation process, assessing the library's performance, usability, and robustness. A series of tests and comparative analysis were carried out to demonstrate the efficiency gains and ergonomic improvements offered by PxBLAT over BLAT usage. We evaluated PxBLAT's efficiency by comparing the execution times of identical alignment tasks performed directly using BLAT and via PxBLAT. Our tests involved datasets of varying sizes to cover a range of typical usage scenarios. The results indicated a significant efficiency gain when using PxBLAT (Figure 1D). In all cases, the execution time was reduced when using PxBLAT compared to BLAT. The reduction in execution time ranged from 1 % to 1 %, clearly demonstrating the benefits of PxBLAT 's direct interfacing approach. In summary, PxBLAT offers tangible benefits in terms of reduced execution time and improved user experience, proving its value as an enhancement to BLAT's functionality.

Through our evaluation, we have demonstrated that PxBLAT not only accelerates the execution of BLAT operations but also presents a user-friendly, pythonic interface that seamlessly integrates with existing Python-based bioinformatics workflows. While the current version of PxBLAT has demonstrated significant advantages in terms of efficiency and ergonomics, we believe that there is potential for further development and enhancement. PxBLAT represents a significant contribution in this regard, offering a blend of performance and usability that stands to benefit a wide range of users.

For gaining the power of concurrent in python, we commonly launch processes due to Global Interpreter Lock (GIL). The overhead of launching process is not trivial as forking a thread, but multiple threads model still empower IO operations Compared to BLAT, PxBLAT benefit the characteristics to speed up multiple clients request spontaneously (Figure 1A and 1B) since we reimplement multithreaded server in Cpp. GIL does not exist in Cpp.

Acknowledgements

Conflict of interest

Funding

Data availability

The PxBLAT, along with the source code, example scripts, and comprehensive documentation, are publicly available in our GitHub repository at https://github.com/cauliyang/pxblat. We have also provided extensive test data, allowing users to validate the functionality and performance of PxBLAT in their own environments. We welcome community contributions to the PxBLAT project. We believe that the open availability of our data and code will foster collaboration and iterative improvement, in line with our vision of creating accessible, efficient tools for bioinformatics.

Reference

Altschul, S. F., Gish, W., Miller, W., Myers, E. W., and Lipman, D. J. Basic local alignment search tool. *Journal of molecular biology*, 215(3):403–410, 1990.

Cock, P. J., Antao, T., Chang, J. T., Chapman, B. A., Cox, C. J., Dalke, A., Friedberg, I., Hamelryck, T., Kauff, F., Wilczynski, B., et al. Biopython: freely available python tools for computational molecular biology and bioinformatics. *Bioinformatics*, 25(11): 1422–1423, 2009.

Higgins, D. G. and Sharp, P. M. Clustal: a package for performing multiple sequence alignment on a microcomputer. *Gene*, 73 (1):237–244, 1988.

Jakob, W., Rhinelander, J., and Moldovan, D. pybind11 — seamless operability between c++11 and python, 2016. https://github.com/pybind/pybind11.

Kent, W. J. Blat—the blast-like alignment tool. *Genome research*, 12(4):656–664, 2002.

Perkel, J. M. Programming: Pick up python. *Nature*, 518(7537):125–126, 2015.

Supplementary Information

Figure S1. This is an endosome.
(**A**) This is a supplementary figure shown as a two-column image with a legend underneath.