Fast protein database as a service with kAamer

Maxime Déraspe1,2, Sébastien Boisvert3, François Laviolette2,5, Paul H Roy4, and Jacques Corbeil1

1Department of Molecular Medicine at Université Laval, Québec, Canada
2Big Data Research Center at Université Laval, Québec, Canada
3Bodycad, 2035 rue du Haut-Bord, Québec, Québec, Canada
4Infectious Disease Research Centre at Université Laval, Québec, Canada
5Department of Computer Science at Université Laval, Québec, Canada

Identification of proteins is one of the most computationally intensive steps in genomics studies. It usually relies on aligners that don’t accommodate rich information on proteins and require additional pipelining steps for protein identification. We introduce kAamer, a protein database engine based on amino acid k-mers, that supports fast identification of proteins with complementary annotations. Moreover, the databases can be hosted and queried remotely.

Main

One fundamental task in genomics is the identification and annotation of DNA coding regions that translate into proteins via a genetic code. Protein databases increase in size as new variants, orthologous and paralogous genes are being sequenced. This is particularly true within the microbial world where bacterial proteomes’ diversity follows their rapid evolution. For instance, UniProtKB (Swiss-Prot / TrEMBL) (1) and NCBI RefSeq (2) contain over 100 million bacterial proteins and that number grows rapidly.

Identification of proteins often relies on accurate, but slow, alignment software such as BLAST or hidden Markov model (HMM) profiles (3, 4). Although other approaches (such as DIAMOND (5)) have considerably improved the speed of searching proteins in large datasets, from a database standpoint much can be done to offer a more versatile experience. One such approach would be to expose the database as a permanent service making use of computational resources for increased performance (i.e. memory mapping) and leveraging the cloud for remote analyses via a Web API. Another approach would be to extend the result set with comprehensive information on protein targets to facilitate subsequent genomics and metagenomics analysis pipelines.

Alignment software usually relies on a seed-and-extend pattern using an index (two-way indexing in DIAMOND) to make local alignments between query and target sequences. However, there is a plethora of research techniques to bypass the computational cost of alignment. Alignment-free sequence analyses usually adopt k-mers (overlapping subsequences of length k) as the main element of quantification. They are extensively used in DNA sequence analyses ranging from genome assemblies (6) to genotyping variants (7), as well as genomics and metagenomics classification (8–10).

In the present study, we introduce kAamer, a fast and comprehensive protein database engine that was named after the usage of amino acid k-mers which differs from the usual nucleic acid k-mers. We demonstrate the usefulness and efficiency of our approach in protein identification from a large dataset and antibiotic resistance gene identification from a pan-resistant bacterial genome.

The database engine of kAamer is based on log-structured merge-tree (LSM-tree) Key-Value (KV) stores (11). LSM-trees are used in data-intensive operations such as web indexing (12, 13), social networking (14) and online gaming (15, 16). KAAmer uses Badger (17), an efficient implementation in Golang of a WiscKey KV (key-value) store (16). WiscKey’s LSM-tree design is optimized for solid state drives (SSD) and separates keys from values to minimize disk I/O amplification. Disk I/O amplification is typical of LSM-trees due to its vertical design in which keys and values need to be read and rewritten in multiple levels of the tree. Therefore, kAamer will obtain peak performance with modern hardware such as NVMe SSDs. Furthermore, traditional block devices such as SATA solid-state drives that offer good throughput in input/output (I/O) operations per second (IOPS) will effectively accommodate use cases where many queries are sent simultaneously. A kAamer database includes three KV stores (see Figure 1A): one to provide the information on proteins (protein store) and two to enable the search functionalities (k-mer store and combination store).

The k-mer store contains all the 7-mers found in the sequence dataset and the keys to the combination store, which uniquely serves the combination of proteins held by k-mers. The fixed k-mer size at 7 was chosen to fit on 4 bytes and keep a manageable database size while offering good specificity over protein targets. The k-merized design of a kAamer database provides an interesting simplicity for the search tasks which will give an exact match count of all 7-mers between a protein query and all targets from a protein database. The result sets using this strategy are not guaranteed to return the same homologous targets that would be obtained with alignment or HMM search and is therefore less suitable for distant homolog retrieval (< 50% identity). Nonetheless, kAamer also supports alignment on the result set without sacrificing speed as shown in Figure 1B. The main drawback of a kAamer database (in the version at the time of writing: 0.4) is the disk space and time required to build a database that is greater than its benchmarked competitors, although it compares favorably to Ghostz (18) for these parameters.

1Go programming language (https://golang.org/)
2Non-Volatile Memory Express
In order to test the efficiency of our database search engine, we used all (114,830,954) the non-fragmented proteins of the UniprotKB (Swiss-Prot / TrEMBL) bacterial proteins dataset (release 2019_08). Sixteen different protein query datasets were randomly and uniquely chosen from the original database, with size ranging from 1 protein to 10,000 proteins. We added the kAAmer search in k-mer match mode (without alignment; named “kaamer-kmatch”) for comparison purposes. We also corrected the kAAmer alignment mode (“kaamer-aln”) in “kaamer-aln+opendb”, by adding the time it took to open the database before running the queries (230 seconds). However, kAAmer’s purpose is to be used as a persistent service so the database opening time becomes insignificant the more you query the database. The four software included in the benchmark are Blastp (v2.9.0+) (3), Ghostz (v1.0.2) (18), Diamond (v0.9.25) (5) and kAAmer (v0.4). Figure 1B illustrates the wallclock times of the alignment software in comparison with kAAmer for protein homology searches. See the Methods section for the hardware used in the benchmarks. We observe that with the larger query datasets (10,000 proteins), kAAmer in alignment mode completes its search and alignments in just 3 minutes 2 seconds (and 2 minutes 26 seconds without alignment). In comparison, Diamond, the second fastest aligner, achieved the 10,000 query task in 11 minutes 57 seconds. Thus, at the maximal benchmarked query size, kAAmer shows an increase in speed of almost 4x (3.93x). When the search incorporates fewer protein queries the gain of kAAmer is more substantial (up to 82x with only 10 protein queries) because Diamond and its double indexing is optimized to perform better when the number of queries increases. It is worth mentioning that when correcting for the database opening (square symbol in Figure 1B.), the kAAmer gain in speed drops and it only surpasses Diamond when there are over 4000 protein queries. However, as stated earlier, kAAmer is rather suited to act as a permanent and flexible database service that will store structured protein information and offer a quick homology search over that protein database. Also, with sufficient random access memory (RAM), data is going to be cached by the operating system (OS) which will increase the performance of kAAmer. For the other benchmarked software, Ghostz took over 33 minutes to realise the task with the 10,000 queries, which is 11 times slower than kAAmer. For Blastp, we stopped the benchmark at 2,000 protein queries since it was already taking over 7 hours to complete the task (at least 700x slower than kAAmer).

In order to accomodate real-use cases we built relevant kAAmer databases and investigated their usage in typical bacterial genomics analyses. It should be noted that annotation of genomes and gene identification rely heavily on the quality of the underlying database. What kAAmer has to offer is the inclusion of the protein information within the database combined with an efficient search functionality to facilitate downstream analyses. Therefore, we also provide utility scripts to demonstrate these use cases. The first use case was to identify antibiotic resistance genes (ARG) in a bacterial genome and test its accuracy related to other
Methods

Design of kAAmer. kAAmer design was influenced by our requirement that protein databases would be permanently hosted (on premise or in the cloud), queried remotely and would have room to scale as sequence databases grow in size. It also needed to be multithreaded for protein searches and would support alignment for more accurate remote homology findings. We opted for a Key-Value store engine that would reside on disk and be optimized for SSDs. We used the Go programming language for its versatility and efficiency. The Key-Value stores use the Badger (17) engine and protein annotations are encoded using Protocol Buffers (23).

Database building. kAAmer is first used to build a database in which all amino acid k-mers are associated with proteins in which they are found. It consists of three KV stores to hold the database information (k-mer store, combination store and protein store). The first KV store (k-mer store) keeps the association of every k-mer (key) with a hash value (key length: 8 bytes) that is the entry to the combination store. The k-mer store is fixed at 7 amino acids to fit k-mer keys onto 32 bits (4 bytes) and thus maintain a manageable final database size while keeping a k-mer size long enough for specificity. The second KV store (combination store) is used to hold all the unique sets of protein identifiers. The method used to build this store can relate to the flyweight design pattern, the hash consign technique, and the coloured de Bruijn graphs (7, 8). Indeed, hash values are reused to access identical objects and therefore minimize memory usage. The set of protein identifiers are the keys to the last store (protein store) which contains the protein information found in the raw annotation file. The raw input file can be either in the EMBL format, GenBank format, TSV format or in FASTA format.

Querying a database. Once we have a database, we expose it with the kAAmer server that listens over HTTP for incoming requests. The benefits of using such a service are two fold. First, the database is opened once and is memory mapped to increase the performance of protein searches. Second, the kAAmer server can be hosted virtually anywhere, in the cloud for instance, and be queried remotely by the kAAmer client. Note that it is preferable that the latency (time required for a message to be transported over HTTP) between the server and client be as low as possible. kAAmer supports protein query and translated DNA query with the number of sequences ranging from 1 to 10 thousand. Each set of sequences was in its own FASTA file to be queried with the number of threads set to 10 and no threshold was provided. All software were run with default parameters, except for the number of threads set to 32 and the maximum number of results per query at 10.

Other kAAmer use cases. Apart from the antibiotic resistance gene (ARG) identification use case, we also provide two demonstrations of kAAmer usage in bacterial genome annotation and metagenome profiling. The use cases are documented at https://github.com/zorino/kaamer_analyses and a Python script is provided for each one of the analyses. For the genome annotation, we used the chromosomal sequence of the same Pseudomonas aeruginosa strain (E6130952) as in the antibiotic resistance genes identification. The kAAmer database that was used for the homology detection is a subset of RefSeq from the Pseudomonadaceae family which is available from the kAAmer repository (see Data availability) along with other Bacterial family databases. Essentially the genome annotation script parses the kAAmer results and produces a GFF (General Feature Format) annotation file giving some threshold on the protein homology. The other use case is the profiling of a metagenome based on the MGNify database of the human gut (24). MGNify includes protein annotations from gene ontology, enzyme commission and kegg pathways, among others. The metagenome profiling script will parse the results and produce a summary file by annotation that counts the presence and abundance of each feature.

Data availability

We have built a repository where one can download prebuilt kAAmer database versions of common protein datasets useful in bacterial genomics and metagenomics. The repository is available at https://kaamer.genome.ulaval.ca/kaamer-repo/ and includes datasets from the NCBI, the EBI and other popular data sources.

Code availability

The code is available at https://github.com/zorino/kaamer under the Apache 2 license and the documentation can be found at https://zorino.github.io/kaamer/.

Acknowledgements

This study was financed by the Canada Research Chair in medical genomics (JC). MD was supported by the Fonds de recherche du Québec - Santé (#32279). The authors thank Juan-Manuel Dominguez and Charles Burdet for their comments. Computations were performed under the auspices of Calcul Québec and Compute Canada. The operations of Compute Canada are funded by the Canada Foundation for Innovation (CFI), the National Science and Engineering Research Council (NSERC), NanoQuébec and the Fonds Québécois de Recherche sur la Nature et les Technologies (FQRNT).
Declarations

The authors declare no competing interests.

Bibliography


13. Sanjay Ghemawat and Jeffrey Dean. LevelDB, 2011. DRAFT.


A) Kmerized Protein Sequences

<table>
<thead>
<tr>
<th>p01</th>
<th>k01</th>
<th>k03</th>
<th>k06</th>
<th>k07</th>
<th>k08</th>
</tr>
</thead>
<tbody>
<tr>
<td>p02</td>
<td>k03</td>
<td>k04</td>
<td>k06</td>
<td>k07</td>
<td>k09</td>
</tr>
<tr>
<td>p03</td>
<td>k02</td>
<td>k03</td>
<td>k06</td>
<td>k07</td>
<td>k08</td>
</tr>
</tbody>
</table>

KAAmer Database

<table>
<thead>
<tr>
<th>K-mer Store</th>
<th>Combination Store</th>
<th>Protein Store</th>
</tr>
</thead>
<tbody>
<tr>
<td>k01 → 0x3</td>
<td>[p01, p02]</td>
<td>p01 → 0x3</td>
</tr>
<tr>
<td>k03 → 0x2</td>
<td>[p01, p03]</td>
<td>p02 → 0x3</td>
</tr>
<tr>
<td>k05 → 0x2</td>
<td>[p02, p03]</td>
<td>p03 → 0x3</td>
</tr>
</tbody>
</table>

B) Protein Search Benchmark

Protein Search Benchmark

<table>
<thead>
<tr>
<th>Software</th>
<th>Runtime in Seconds</th>
</tr>
</thead>
<tbody>
<tr>
<td>blastp</td>
<td>0</td>
</tr>
<tr>
<td>ghostz</td>
<td>1</td>
</tr>
<tr>
<td>diamond</td>
<td>2</td>
</tr>
<tr>
<td>kaamer-kmatch</td>
<td>4</td>
</tr>
<tr>
<td>kaamer-aln</td>
<td>6</td>
</tr>
<tr>
<td>kaamer-aln+opendb</td>
<td>7</td>
</tr>
</tbody>
</table>

Fig. 1. A) Design of a KAAmer database. Three key-value stores are created within a database (K-mer Store, Combination Store, Protein Store). Colours indicate the combination (hash) value that are reused in the combination store. Proteins are numbered (p01, p02, p03) and k-mers are numbered (k01, k02, ..., k08). B) Protein search benchmark. Software include blastp (v2.9.0+), ghostz (v1.0.2), diamond (v0.9.25) and KAAmer (v0.4) with and without alignment.

Table 1. Report of the antibiotic resistance genes identification within the pan-resistant Pseudomonas aeruginosa E6130952 strain from KAAmer+NCBI-arg, ResFinder and CARD databases.