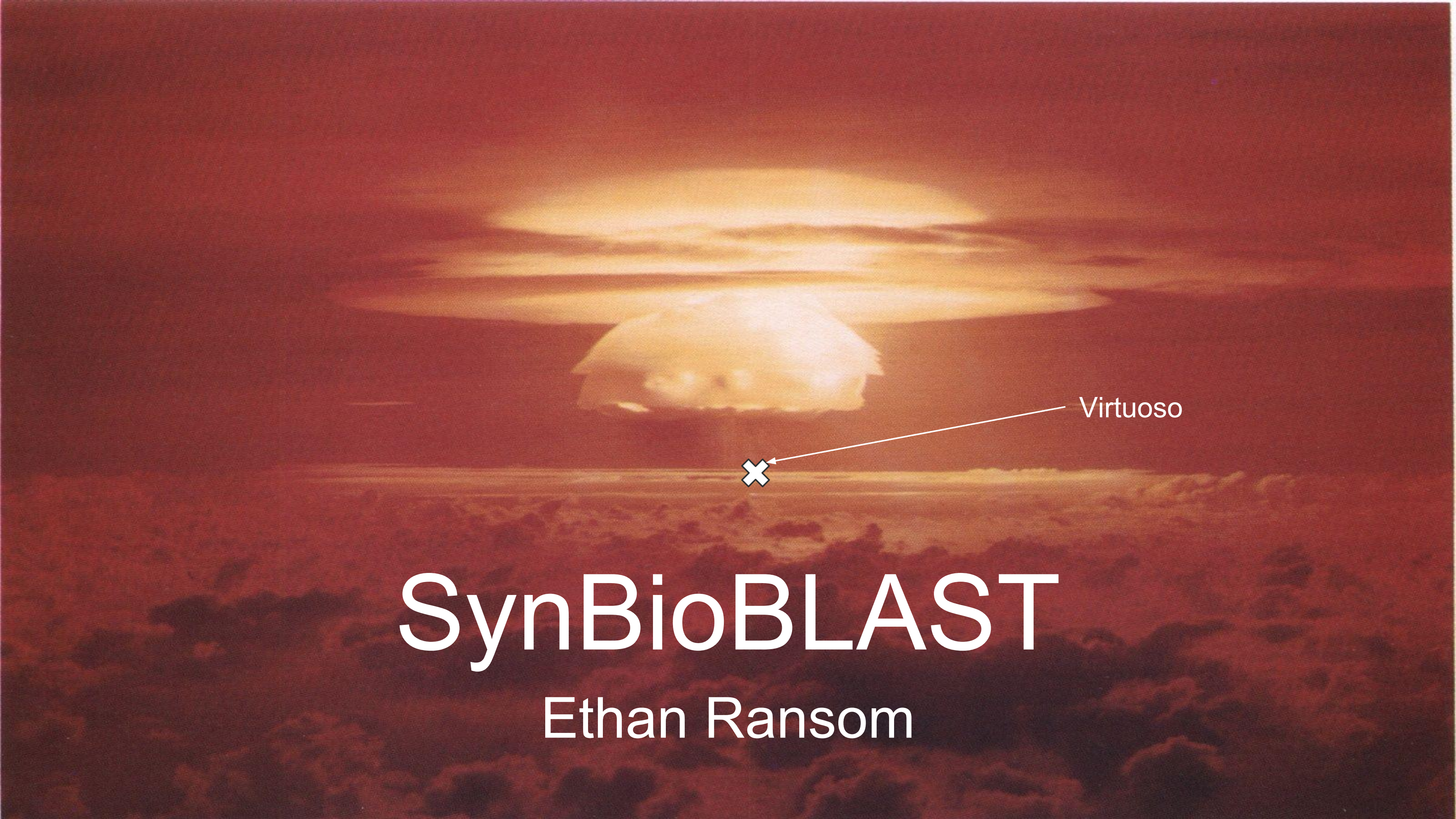


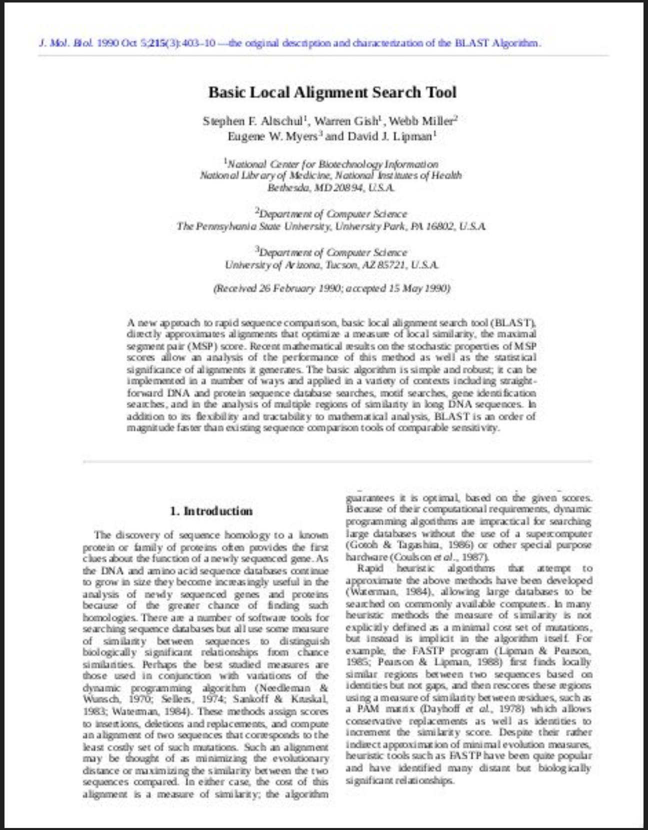
SynBioBLAST

Ethan Ransom

Has this ever happened to you?

Ever been unsure if a component isn’t in SynBioHub or if you just can’t find it?

Want to see all uses of a particular component without assuming submitters have built their SBOL correctly?



J. Mol. Biol. 1990 Oct 5;**215**(3):403–10 —the original description and characterization of the BLAST Algorithm.

Basic Local Alignment Search Tool

Stephen F. Altschul[1], Warren Gish[1], Webb Miller[2]
Eugene W. Myers[3] and David J. Lipman[1]

[1]*National Center for Biotechnology Information*
National Library of Medicine, National Institutes of Health
Bethesda, MD 20894, U.S.A.

[2]*Department of Computer Science*
The Pennsylvania State University, University Park, PA 16802, U.S.A.

[3]*Department of Computer Science*
University of Arizona, Tucson, AZ 85721, U.S.A.

(Received 26 February 1990; accepted 15 May 1990)

A new approach to rapid sequence comparison, basic local alignment search tool (BLAST), directly approximates alignments that optimize a measure of local similarity, the maximal segment pair (MSP) score. Recent mathematical results on the stochastic properties of MSP scores allow an analysis of the performance of this method as well as the statistical significance of alignments it generates. The basic algorithm is simple and robust; it can be implemented in a number of ways and applied in a variety of contexts including straight-forward DNA and protein sequence database searches, motif searches, gene identification searches, and in the analysis of multiple regions of similarity in long DNA sequences. In addition to its flexibility and tractability to mathematical analysis, BLAST is an order of magnitude faster than existing sequence comparison tools of comparable sensitivity.

1. Introduction

The discovery of sequence homology to a known protein or family of proteins often provides the first clues about the function of a newly sequenced gene. As the DNA and amino acid sequence databases continue to grow in size they become increasingly useful in the analysis of newly sequenced genes and proteins because of the greater chance of finding such homologies. There are a number of software tools for searching sequence databases but all use some measure of similarity between sequences to distinguish biologically significant relationships from chance similarities. Perhaps the best studied measures are those used in conjunction with variations of the dynamic programming algorithm (Needleman & Wunsch, 1970; Sellers, 1974; Sankoff & Kruskal, 1983; Waterman, 1984). These methods assign scores to insertions, deletions and replacements, and compute an alignment of two sequences that corresponds to the least costly set of such mutations. Such an alignment may be thought of as minimizing the evolutionary distance or maximizing the similarity between the two sequences compared. In either case, the cost of this alignment is a measure of similarity; the algorithm guarantees it is optimal, based on the given scores. Because of their computational requirements, dynamic programming algorithms are impractical for searching large databases without the use of a supercomputer (Gotoh & Tagashira, 1986) or other special purpose hardware (Coulson *et al.*, 1987).

Rapid heuristic algorithms that attempt to approximate the above methods have been developed (Waterman, 1984), allowing large databases to be searched on commonly available computers. In many heuristic methods the measure of similarity is not explicitly defined as a minimal cost set of mutations, but instead is implicit in the algorithm itself. For example, the FASTP program (Lipman & Pearson, 1985; Pearson & Lipman, 1988) first finds locally similar regions between two sequences based on identities but not gaps, and then rescores these regions using a measure of similarity between residues, such as a PAM matrix (Dayhoff *et al.*, 1978) which allows conservative replacements as well as identities to increment the similarity score. Despite their rather indirect approximation of minimal evolution measures, heuristic tools such as FASTP have been quite popular and have identified many distant but biologically significant relationships.

BLAST: the most-cited paper of the 1990s

Demo: using BLAST

SynBioHub

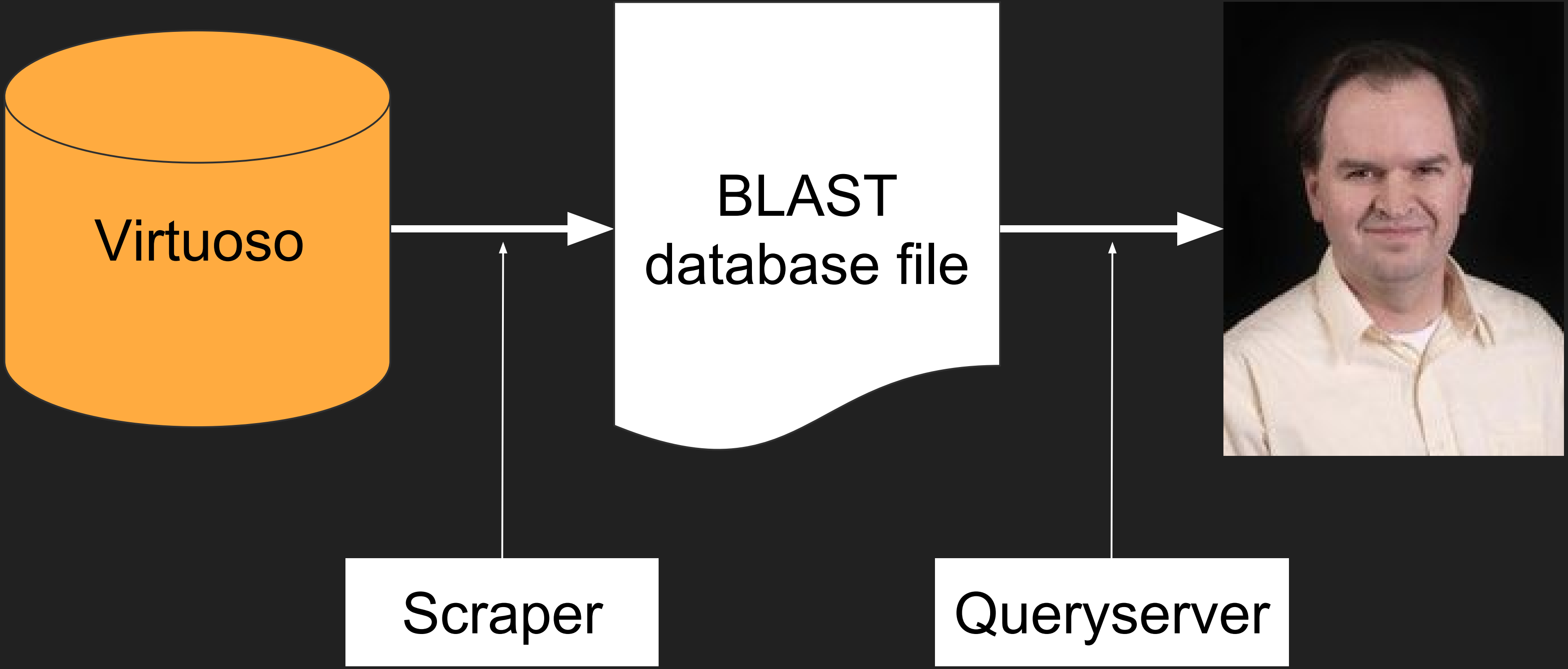
+

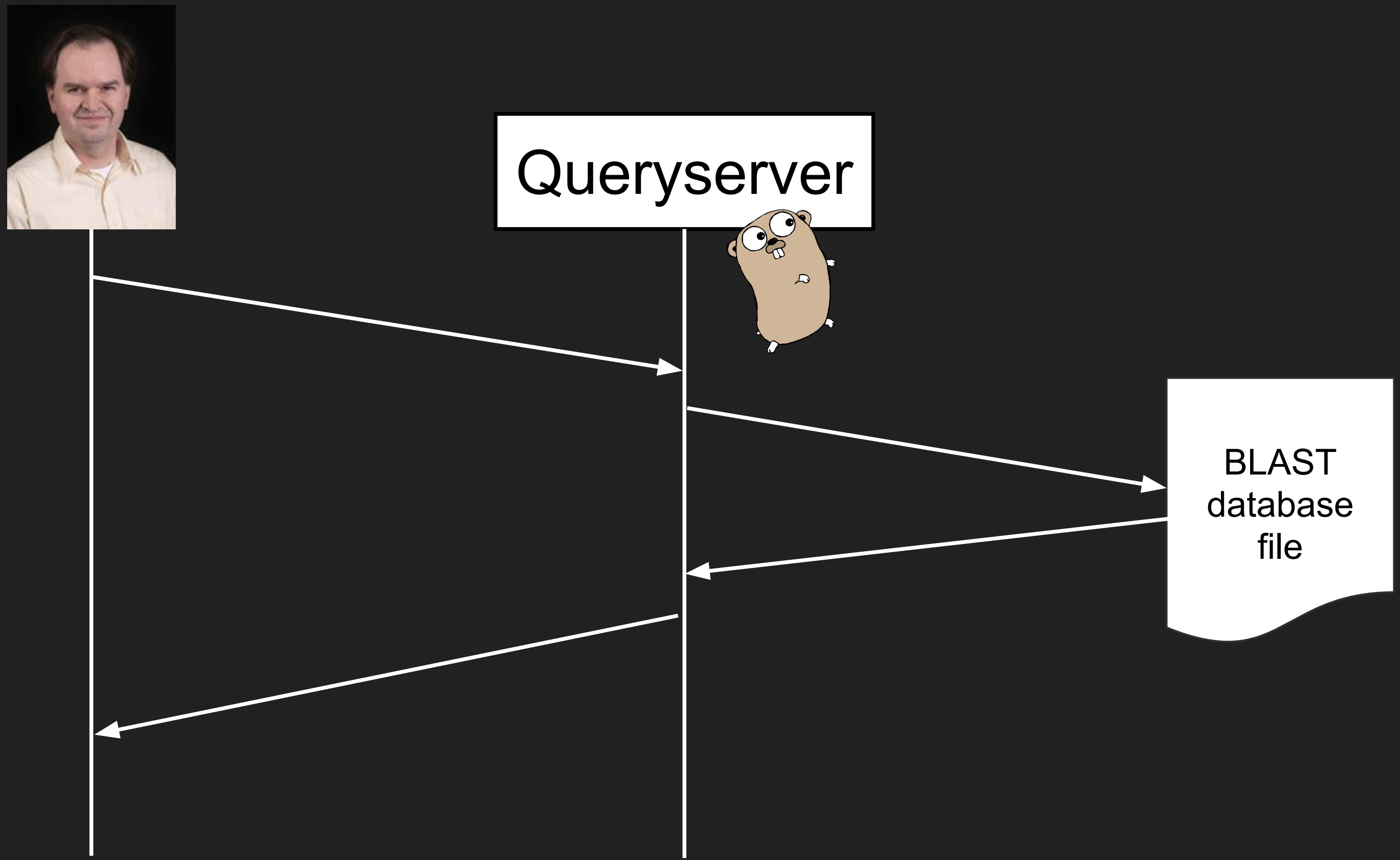
Nucleotide BLAST

nucleotide ▸ nucleotide

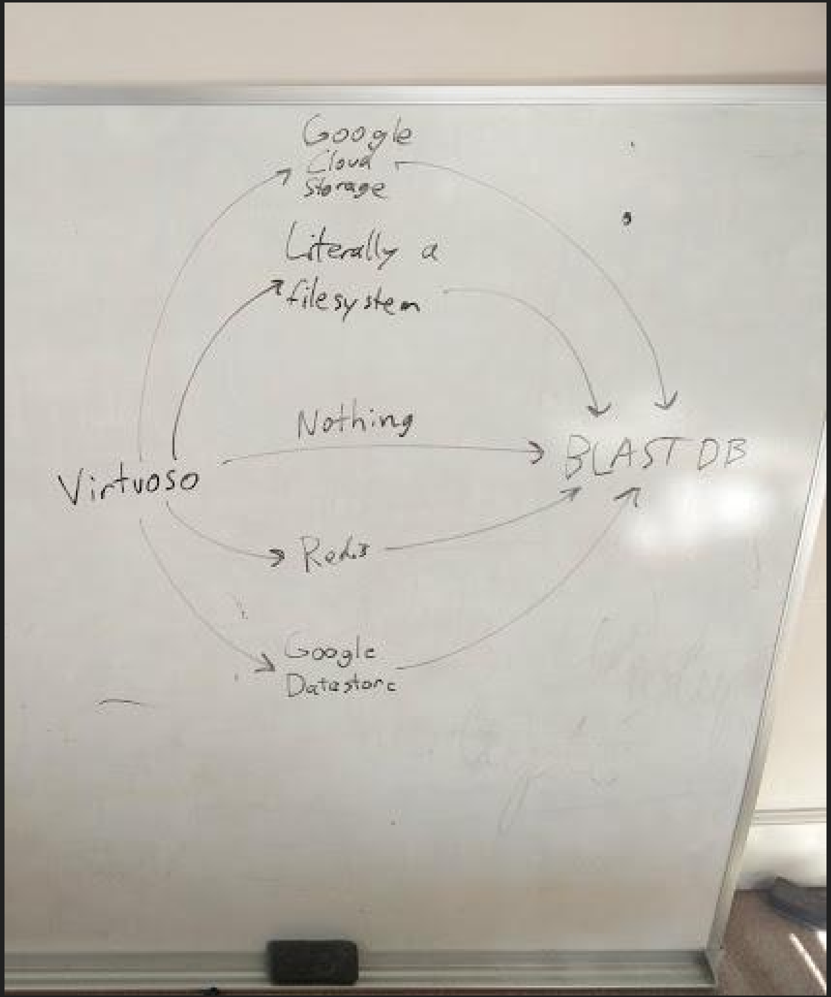
SynBio BLAST

SynBioBLAST

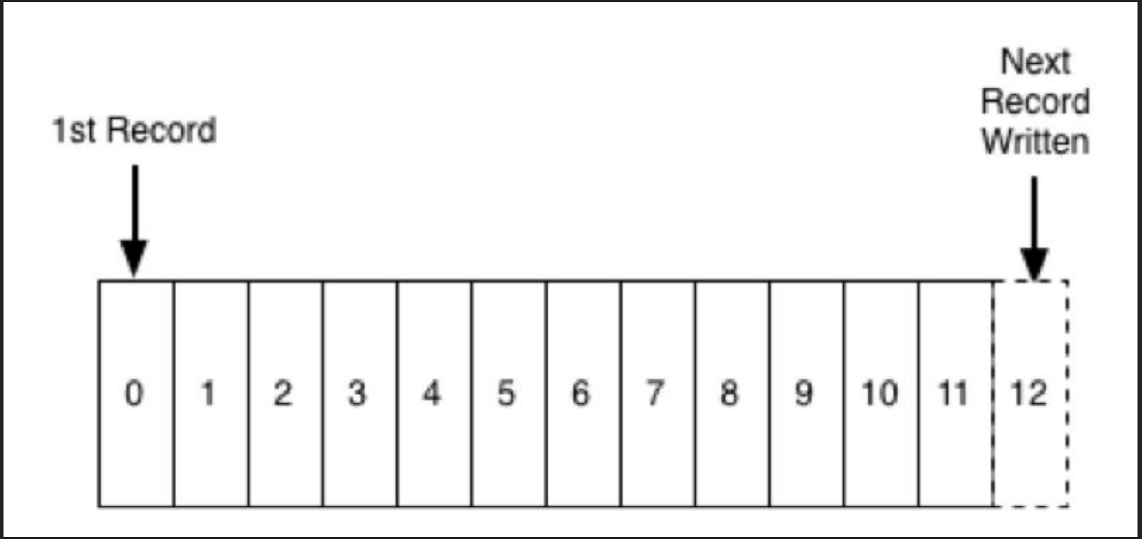



Queryserver
BLAST database file

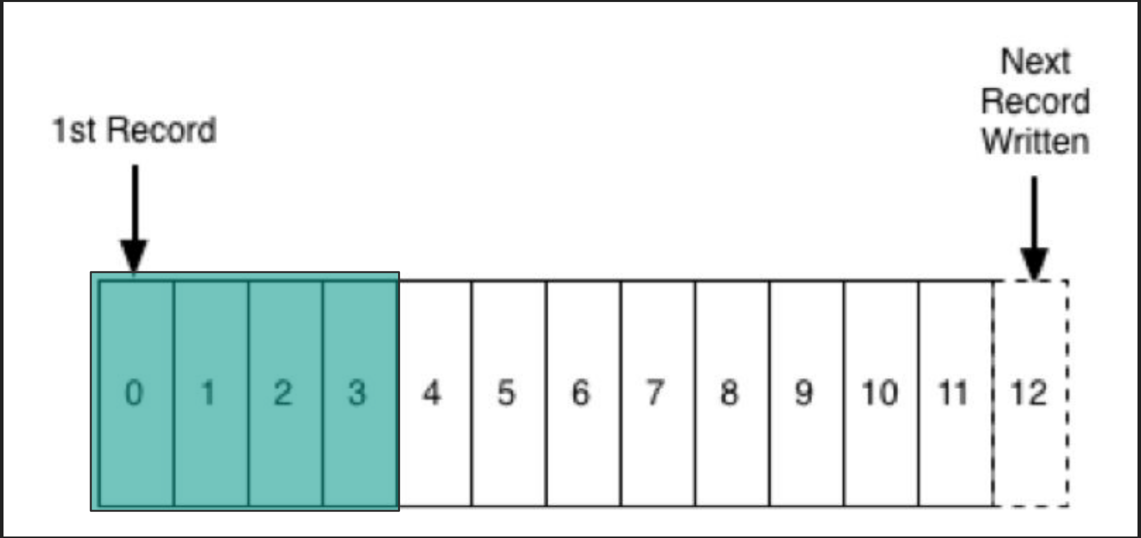
The Scraper or: How I Learned to Stop Worrying and Just Waste CPU Cycles



Concern: don’t want to overload Virtuoso
Solution: scrollable cursor a.k.a. a Log

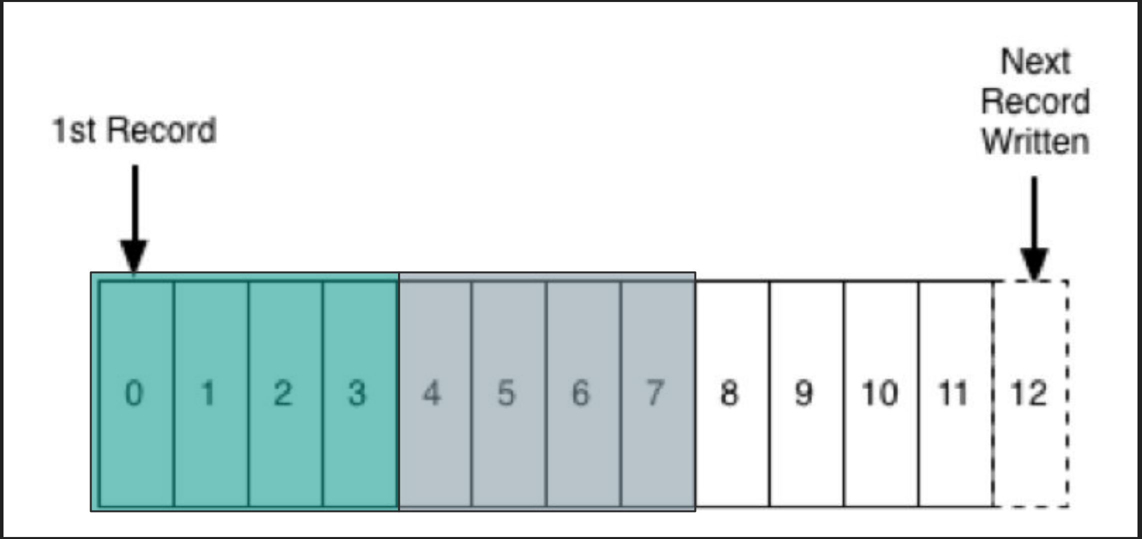


Concern: don’t want to overload Virtuoso
Solution: scrollable cursor a.k.a. a Log



Query 1: Give me oldest 4 sequences

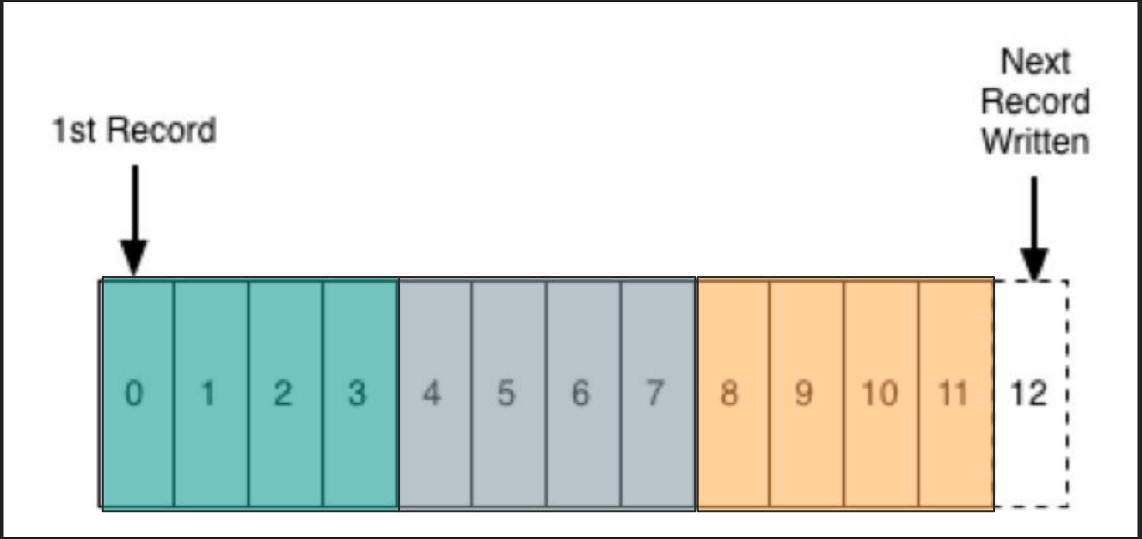
Concern: don’t want to overload Virtuoso
Solution: scrollable cursor a.k.a. a Log



Query 1: Give me oldest 4 sequences

Query 2: Give me oldest 4 starting at 4th oldest

Concern: don’t want to overload Virtuoso
Solution: scrollable cursor a.k.a. a Log



Query 1: Give me oldest 4 sequences

Query 2: Give me oldest 4 starting at 4th oldest

Query 3: Give me oldest 4 starting at 8th oldest

Update algorithm

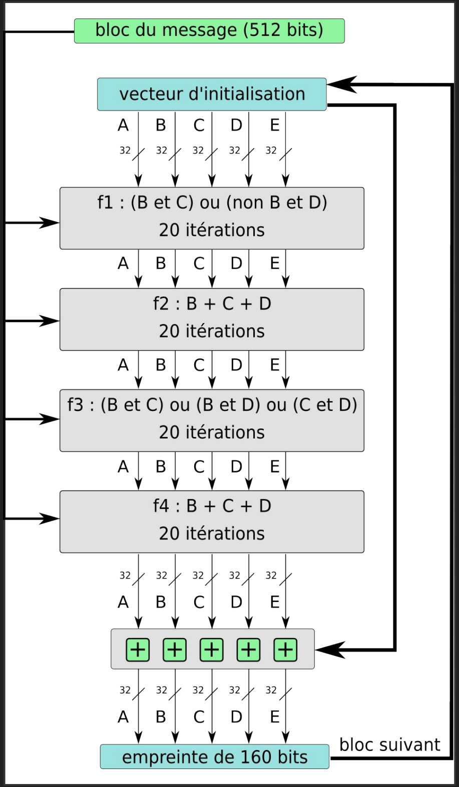
- Loop forever
    - Fetch records from Virtuoso, OFFSET by `offset`
    - Add each record to the BLAST database
    - Increment `offset` by the number of records processed
    - If number of records < batch size
        - Virtuoso doesn’t have anything new for us
        - Sleep for a while (few hours)

Disadvantage: this throws away information about sequences referenced by multiple components

Query gives us table of (uri, sequence) pairs

URI	Sequence
https://synbiohub.org/public/igem/BBa_K2000000/1	tccctatcagtgatagagattgacatccctatcagtgatagagata...
https://synbiohub.org/public/igem/BBa_J22146/1	aacaatttctacaaaacacttgatactgtatgagcatacagtataa...
https://synbiohub.org/public/igem/BBa_K2000005/1	tccctatcagtgatagagattgacatccctatcagtgatagagata...
https://synbiohub.org/public/igem/BBa_K2000004/1	tccctatcagtgatagagattgacatccctatcagtgatagagata...
https://synbiohub.org/public/igem/BBa_M1579/1	atgatcctcaccccggaacaagttgcagcagcgcaaaggcc...
https://synbiohub.org/public/igem/BBa_J22146/1	aacaatttctacaaaacacttgatactgtatgagcatacagtataa...
...	...

sha1(“atcgcgggattccc”) ⇨
07f790394a4602a13b80cae
1637df48fdbcc75a5



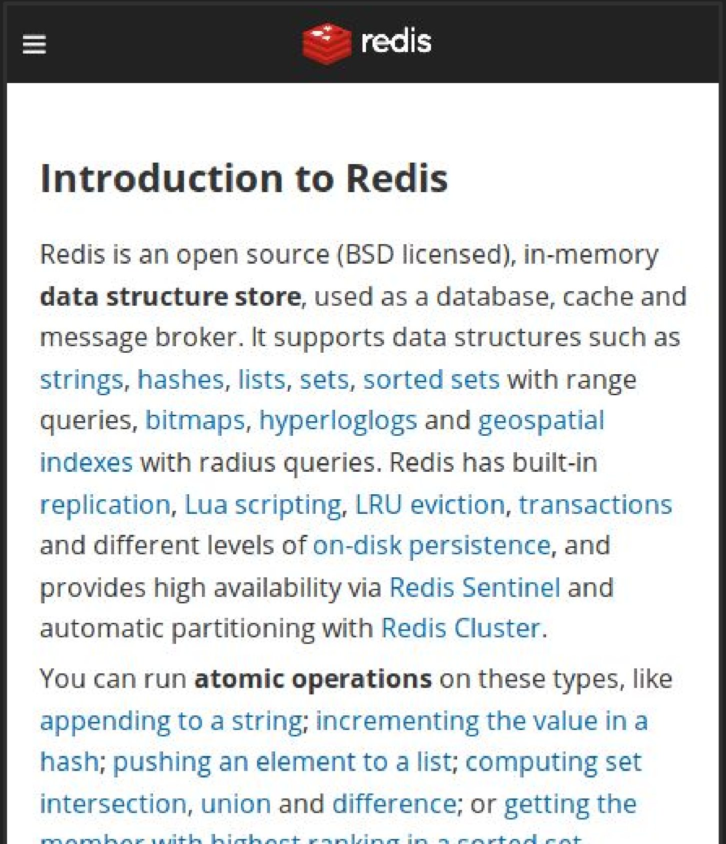
Identify seqs by sha1 hash, keep list of corresp. uris

Sequences = { 683c0aa1fca5dec8e9039744a..., 9b7578f..., 1e9644... }

- https://synbiohub.org/public/igem/BBa_K2000000/1
- https://synbiohub.org/public/igem/BBa_K2000002/1
- https://synbiohub.org/public/igem/BBa_K2000003/1
- https://synbiohub.org/public/igem/BBa_K2000005/1
- https://synbiohub.org/public/igem/BBa_K2000004/1

- https://synbiohub.org/public/igem/BBa_J22151/1
- https://synbiohub.org/public/igem/BBa_J22146/1

For deduplication info:



Introduction to Redis

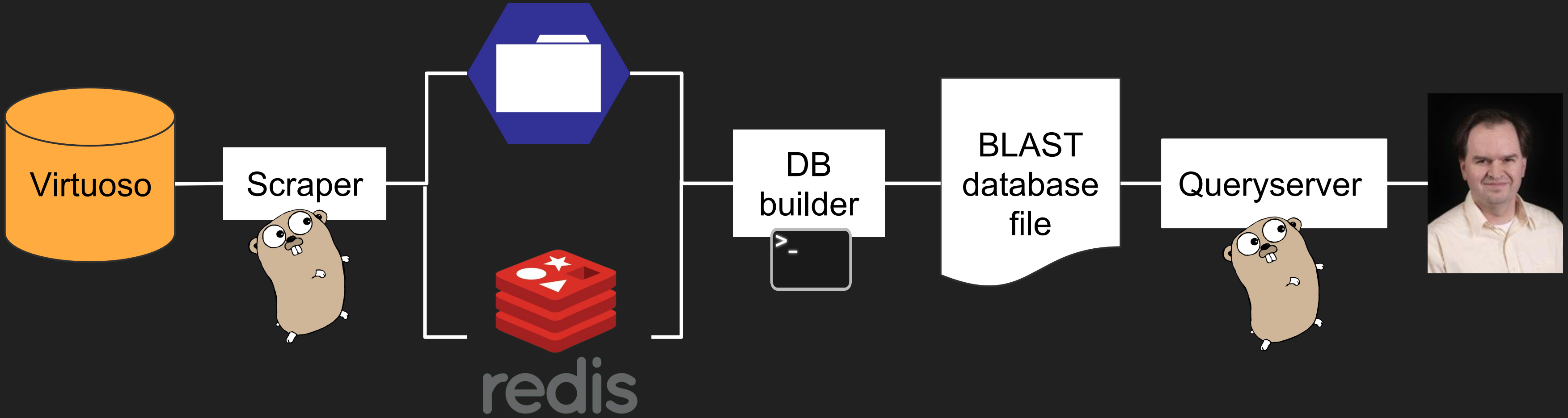
Redis is an open source (BSD licensed), in-memory **data structure store**, used as a database, cache and message broker. It supports data structures such as strings, hashes, lists, sets, sorted sets with range queries, bitmaps, hyperloglogs and geospatial indexes with radius queries. Redis has built-in replication, Lua scripting, LRU eviction, transactions and different levels of on-disk persistence, and provides high availability via Redis Sentinel and automatic partitioning with Redis Cluster.

You can run **atomic operations** on these types, like appending to a string; incrementing the value in a hash; pushing an element to a list; computing set intersection, union and difference; or getting the member with highest ranking in a sorted set

For sequences:

Literally just a folder of FASTA files

Actual Architecture

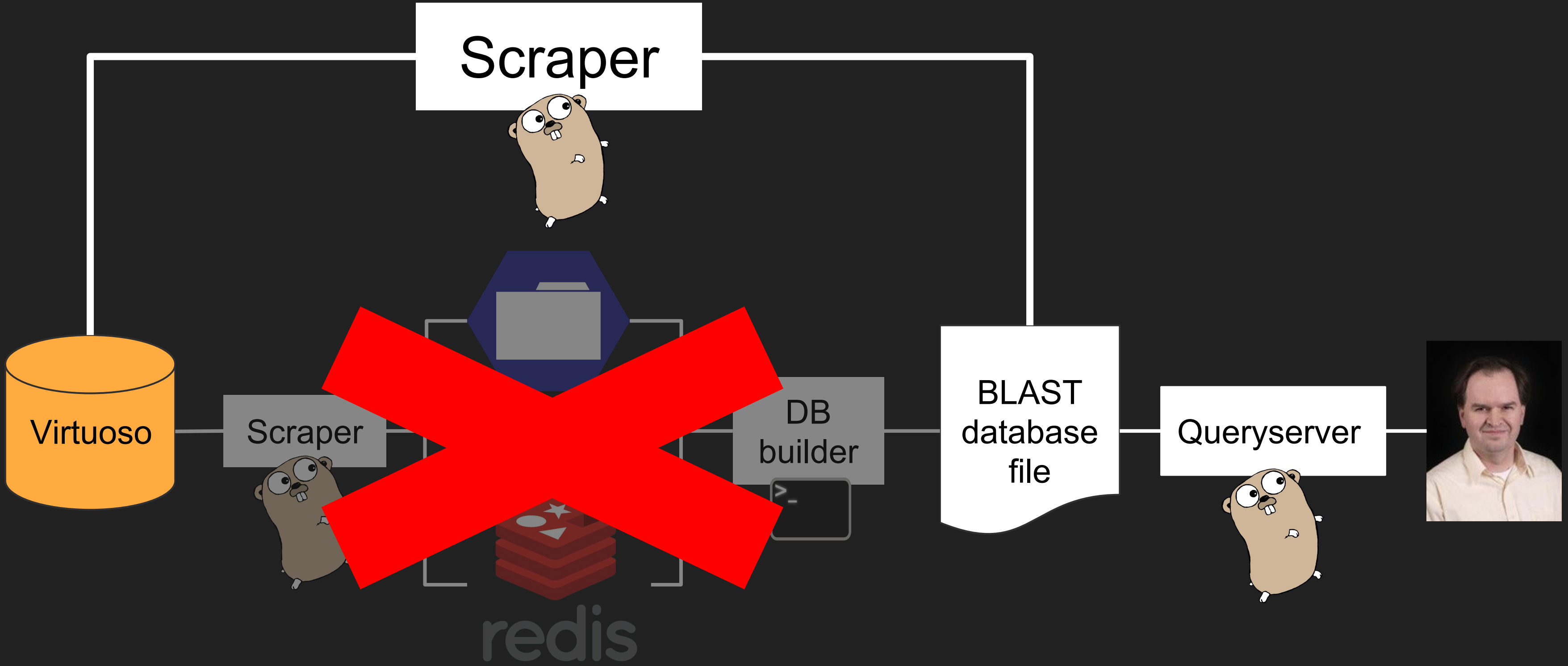


Actual scraping time: like 3 mins lol

Results:

33,325 unique sequences
143mb of FASTA files
~13kb of dedup info in Redis

Could be wayyyyyyyyyy simpler



Demo: searching with SynBioBLAST

Future Work

- Make the web interface look not-terrible
- Set up a Docker container so other SynBioHub instances could set up their own SynBioBLASTs quickly
- Merge into SynBioHub
- BLAST can be optimized for shorter sequences, which seem to be most commonly found in SynBioHub
- Export deduplication information back into SynBioHub

The End: Thanks!