Accelerating Bioinformatics with Big Data Technologies

Dylan Hutchison

Group 110 Computing and Analytics

6 August 2012

LINCOLN LABORATORY
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

Outline

- **Introduction**

- **Approach**

- **Results**

- **Summary**

DNA Sequence Matching

Goal

- Quickly compare two sets of DNA

Applications

- Identification
- Mixture Analysis
- Kinship Analysis
- Ancestry Analysis

Uses: disease outbreaks, criminal investigations, personal medical services, …

- **Challenge: sequence matching takes a long time, can we make it faster?**

Outline

- Introduction
- **Approach**
- Results
- Summary

Sequence Matching ⇔ Sparse Matrix Multiply in D4M

RNA Reference Set

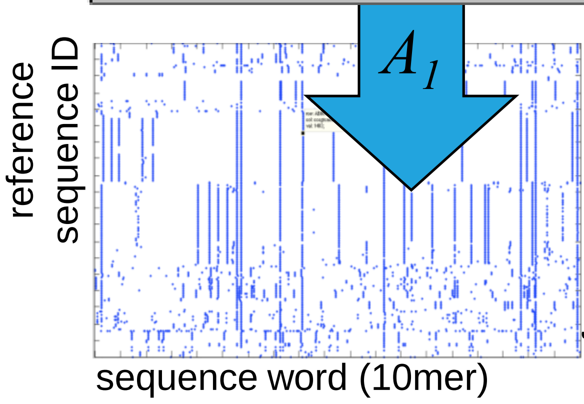
reference bacteria

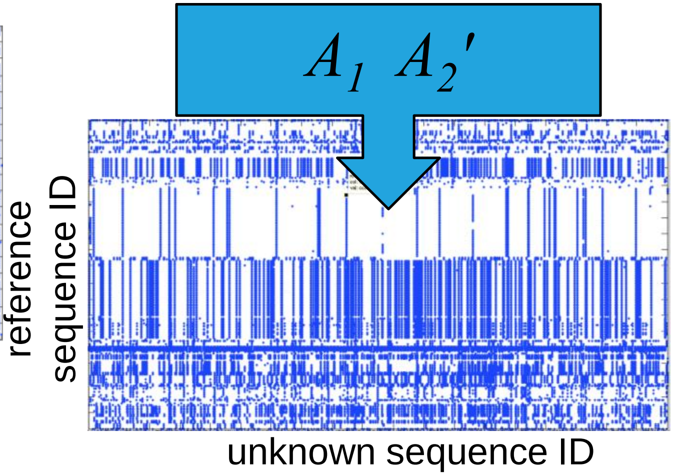
SeqID	sequence
AB000106.1_1-1343	ggaatctgcccttgggttcgg
AB000278.1_1-1410	caggcctaacacatgcaagt
AB000389.1_1-1508	ttgatcctggctcagattgaa
AB000390.2_1-1428	catgcaagtcgagcggaaac

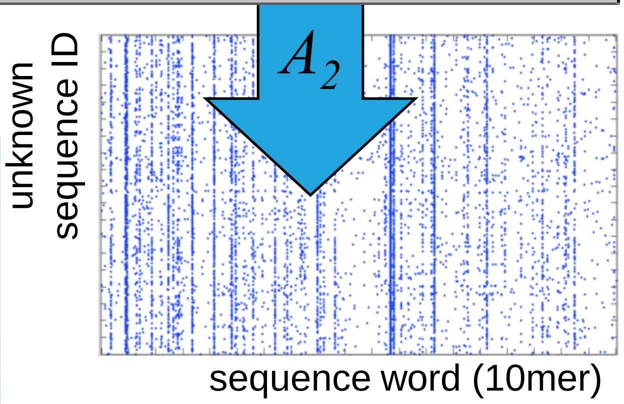
Collected Sample

unknown bacteria

SeqID	sequence
G6J0L4R01AUYU3	TAGATACTGCTGCCTCCCG
G6J0L4R01DLKJM	TTTTTTTCGTGCTGCTGCCT
G6J0L4R01D0SEN	TTATCGGCTGCTGCCTCCC
G6J0L4R01EOS3L	AGGTTGTCTGCTGCCTCTA





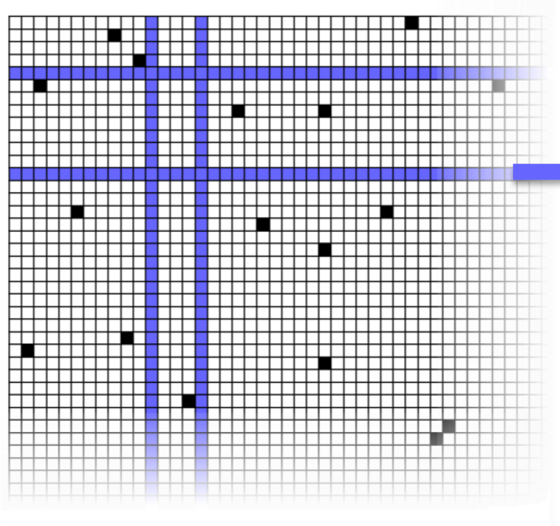


	Database Sequence				Sample Sequence				Matchup		
						SeqU1	SeqU2				
					tcccgtagga	7				SeqU1	SeqU2
	tagatactgc	agatactgct	gatactgctg		agatactgct	33	29	⇨	Seq1	1	2
Seq1	1	2	3		ctcccgtagg		82		Seq2		1
Seq2	55		43		gatactgctg		42				

D4M Stores Giant Sparse Matrices in Accumulo Triple Store Database

Triple Store

Distributed Database



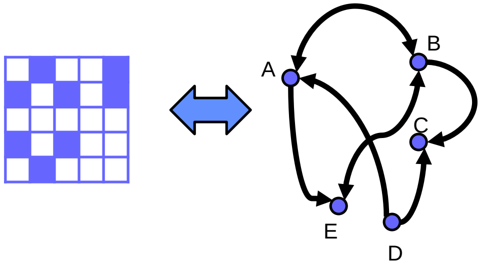
D4M

Dynamic
Distributed
Dimensional
Data
Model

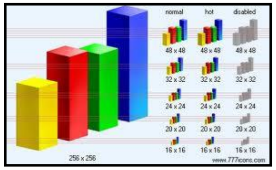
Query:
T(:,ggaatctgcc)

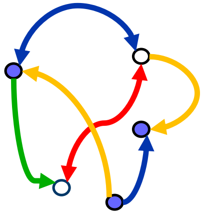
Associative Arrays

Numerical Computing Environment



A D4M query returns a sparse matrix or graph from a triple store…





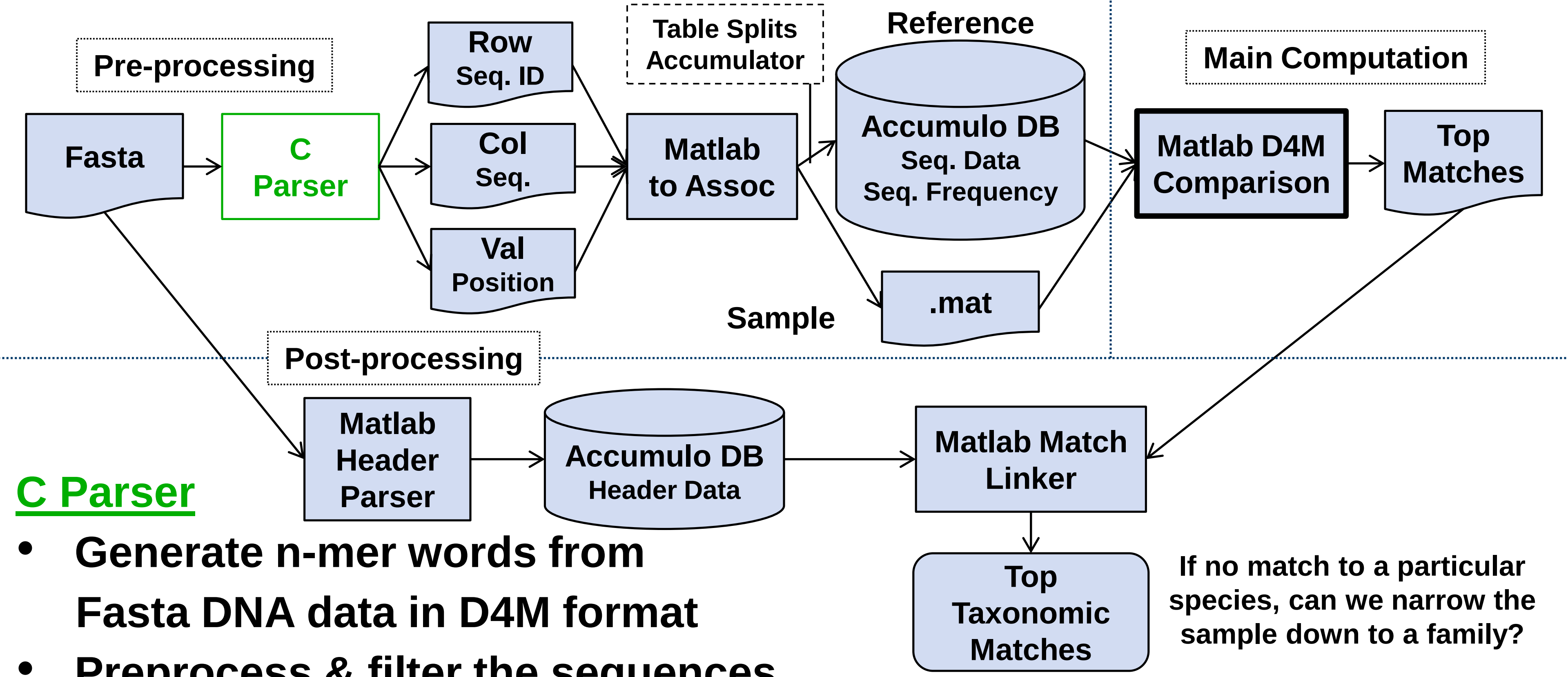
…for statistical signal processing or graph analysis in Matlab

Triple store are high performance distributed databases for heterogeneous data

Database Advantage

- Store word frequency
- Only compare against the most frequent words

Sequence Processing Pipeline



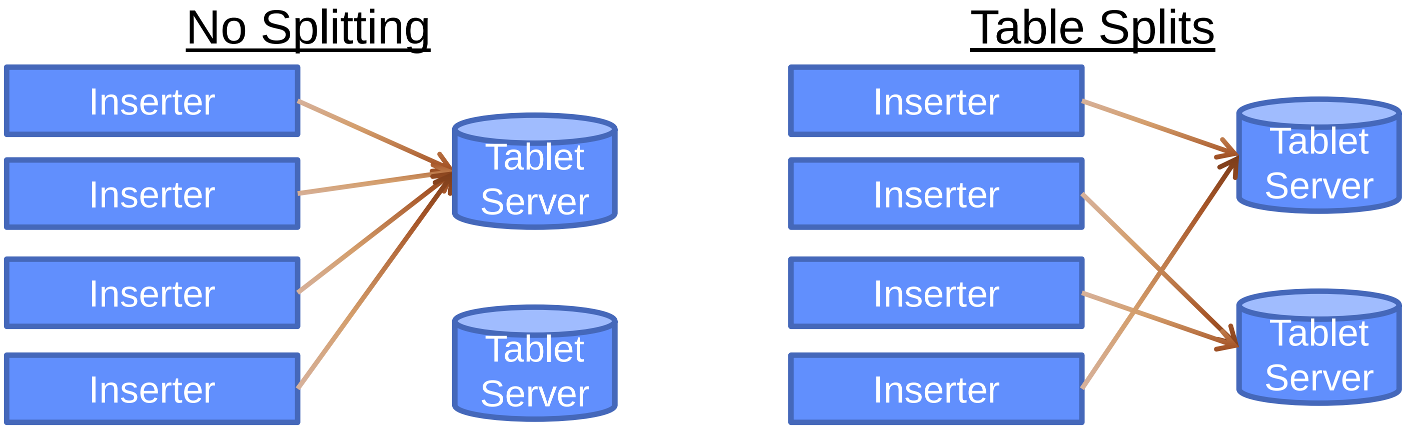
If no match to a particular species, can we narrow the sample down to a family?

C Parser

- Generate n-mer words from Fasta DNA data in D4M format
- Preprocess & filter the sequences
  - Ignore bad, common sequences
  - Break output files into manageable chunks, say 5MB
  - Generate reverse complement sequences
  - Break up big sequences into subsequences to preserve locality

D4M Contributions:
Table Splits

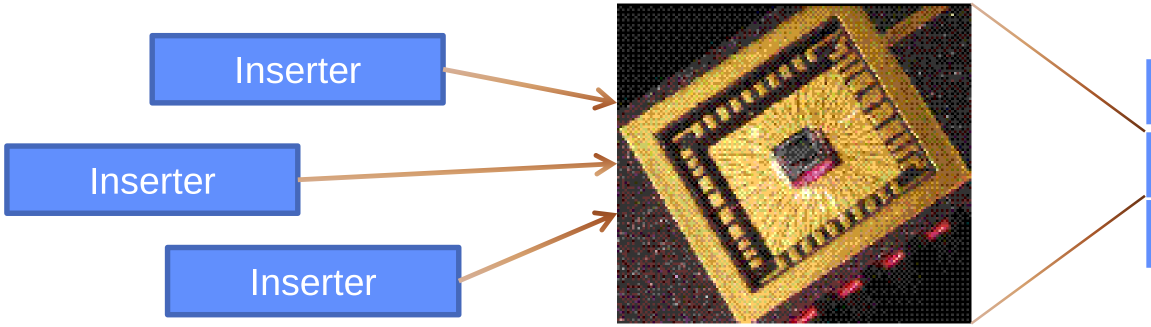
- **Initial inserts bottleneck on one tablet server until it fills up and splits**
- **Performance booster: pre-split table among several tablet servers for instant parallel insertion**
  - **Use advanced knowledge of row data patterns to choose splits**
- **Created functions to set, merge and query table splits**



D4M Contributions: Accumulator Columns

- **Introduced Accumulo’s combiner to D4M**
  - **Example: Document word counting**
    - **Row ID = Document ID**
    - **Column = Word**
    - **Value = Count**
  - **Insert (Doc1, ‘bird’, 2) → DB has (Doc1, ‘bird’, 2)**
  - **Insert (Doc1, ‘bird’, 3) → DB has (Doc1, ‘bird’, 5)**
- **Works with any commutative operation**
  - **Addition, maximum, minimum, etc.**
- **Very handy for D4M users**

Database Accumulator

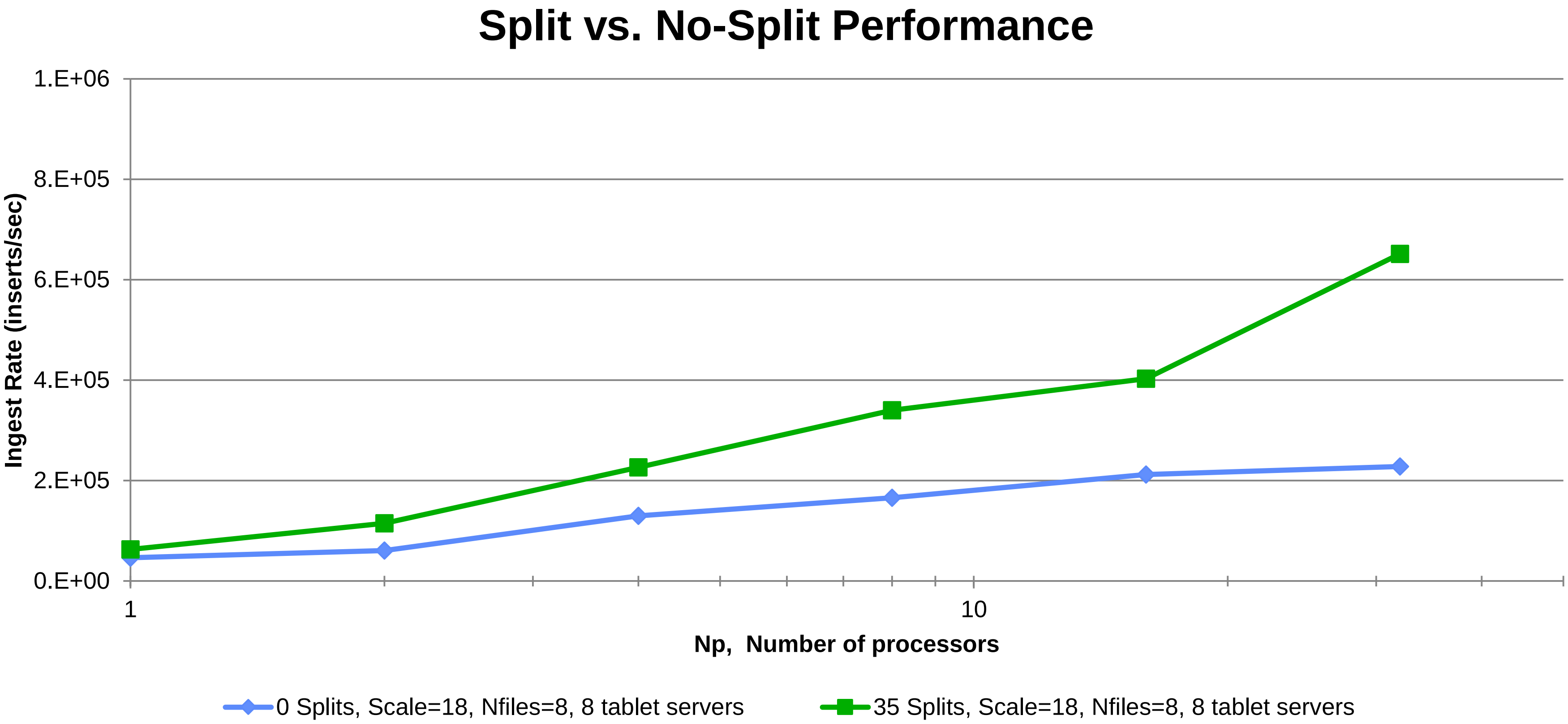


	Col	
Row	SUM	

Outline

- Introduction
- Approach
- Results
- Summary

D4M Performance: Table Splits



pMatlab + tic & toc

Each inserter inserts ~2^{18} rows of ~8 bytes each 8 times = 16 MB/inserter

- Ran in exclusive mode – one inserter process per node

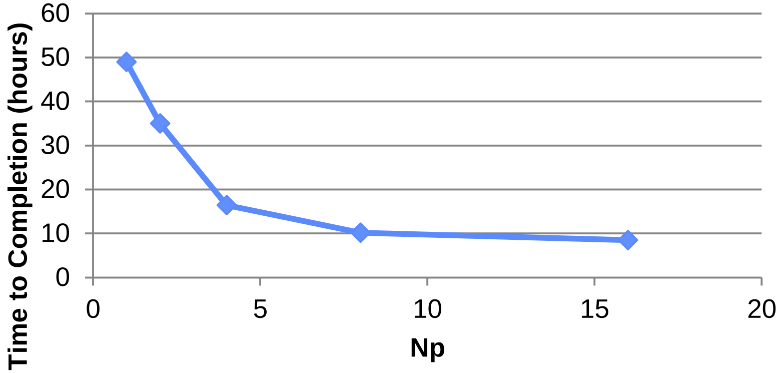
Conclusion: Pre-Splitting tables appropriately can **double ingest rates** at higher Np in multinode database environments

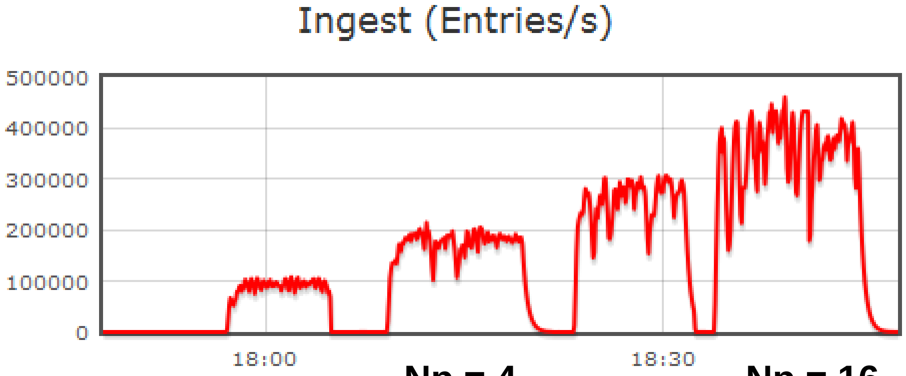
Sample to Human Performance

- **4.5 GB human Fasta file**
- **C Parser took 25 minutes**
- **101 GB of row, col files**
- **Ingest Time: Reduced 50 hours to 10 hours**

Long pre-processing; Fast query comparison!

Extrapolated Human Ingest Run Times

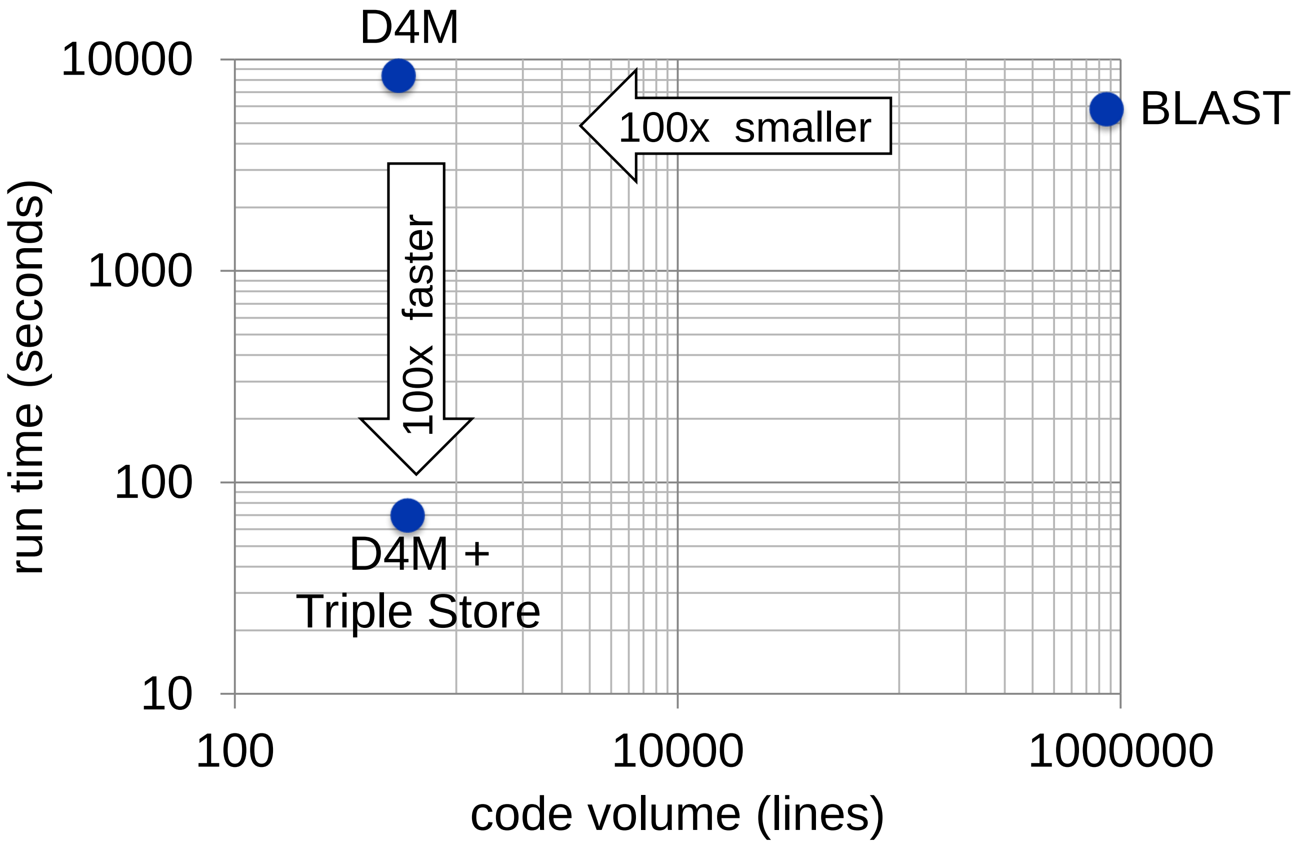




Np = 2 **Np = 4** **Np = 8** **Np = 16**

8 Tablet Server Accumulo Instance

Leveraging "Big Data" Technologies for High Speed Sequence Matching



- **High performance triple store database trades computations for lookups**
- **Used Apache Accumulo database to accelerate comparison by 100x**
- **Used Lincoln D4M software to reduce code size by 100x**

Summary

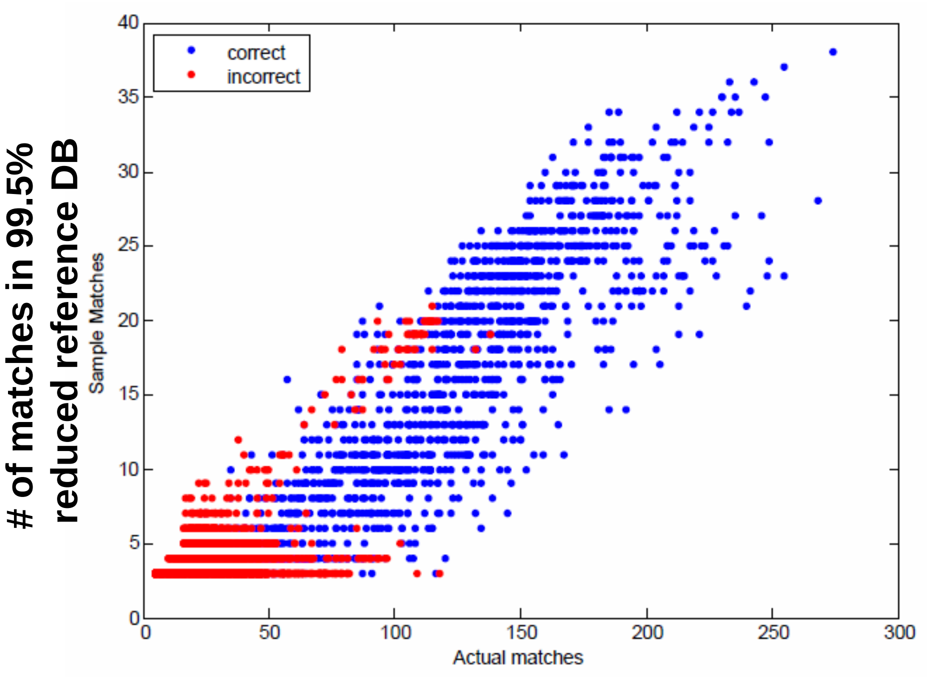
- New Approach to DNA Matching
  - D4M: Sparse Matrix Multiplication + Database Integration
- Speed over depth – excellent first-stage tool
- Promising pending results – estimate big speedup over BLAST
- Scalable: Performance scales with database/query size and number/power of processors
  - Need faster results? Add more/better COTS machines
- Techniques applicable to more bioinformatics problems
  - Protein comparison, SNP analysis, the meaning of life (almost)

Big Thanks to Jeremy Kepner and Darrell Ricke of Group 110, 48!

Backup Slides

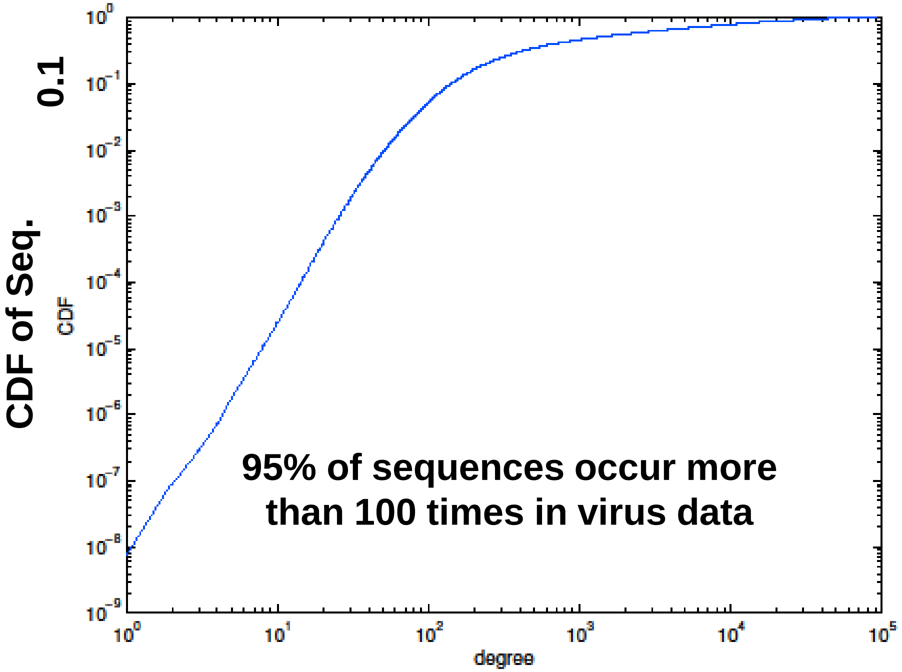
Further Speedup:
Only need the rare sequences!

- Can eliminate 10mers occurring in the reference DB > 100 times (95% of all 10mers) and still correctly match top results
- Implication: Significantly smaller # of entries to scan
  Reduced database size and search space



# of matches in 99.5% reduced reference DB

# of matches in full reference DB



CDF of Seq.

0.1

95% of sequences occur more than 100 times in virus data

Sequence Occurrence in Reference DB

Parallel and Distributed Computing

- General Development: Improving D4M
  - What is D4M? Why is it useful?
    - Link to Accumulo database
  - Some Contributions:
    - Accumulator Columns
    - Table Splits
  - Performance Testing Results

- Specific Application: DNA Matching
  - Background on Bioinformatics
  - The DNA Matching Problem
  - D4M Solution – Matrices and Assoc Arrays
  - Complete Pipeline
  - Optimization
  - Performance & Results

- What's next?

D4M for Dummies

- **Dynamic Distributed Dimensional Data For Matlab**
- **Associative Array**
  - **Sparse matrix – memory efficient**
  - **Easily distributed across multiple machines with *pMatlab***
  - **Linkable with Accumulo Database for Big Data**
    - **Remote computation – Bring the processing to the data**
    - **1-to-1 correspondence between database queries & linear algebra**

      Easier, cleaner implementation than SQL queries
  - ***Doubly* Index-able Triple Store**
    - **O(log n) search time on row and column**
- **Lincoln Lab development**
  - **Currently used in text and cyber analytics**

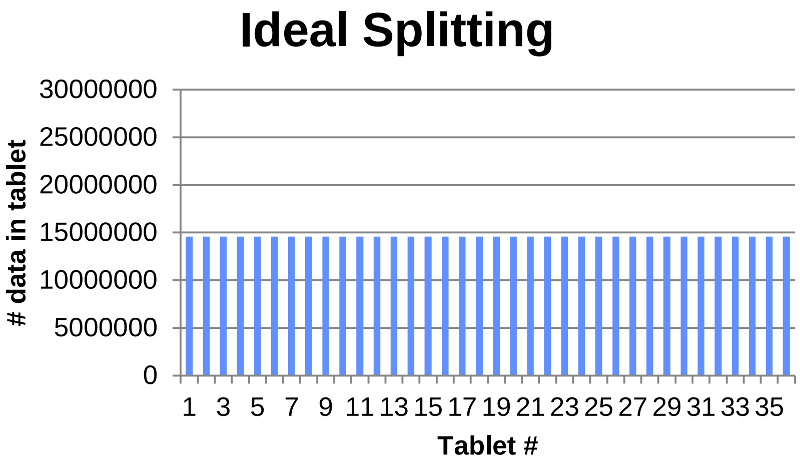
Assoc Array Example

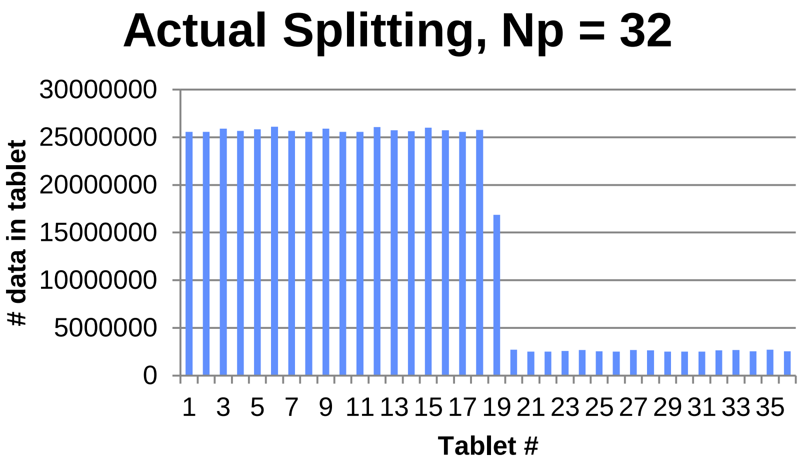
(ACR56360.1,,GGBKTAO01A0YYP,) 5
(ACR56360.1,,GGBKTAO02CAGWF,) 1
(ADK12630.1,,GGBKTAO02CAGWF,) 3
(ADY69164.1,,GGBKTAO01A0YYP,) 128

	GGBKTAO01A0YYP,	GGBKTAO02CAGWF,
ACR56360.1,	5,	1,
ADK12630.1,		3,
ADY69164.1,	128,	

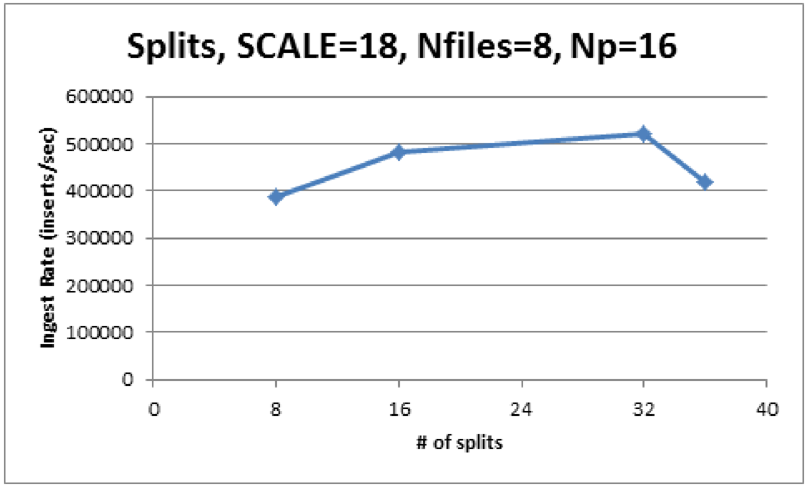
➔ System architecture for rapidly analyzing very large problems ➔

D4M Performance: Why 35 splits?

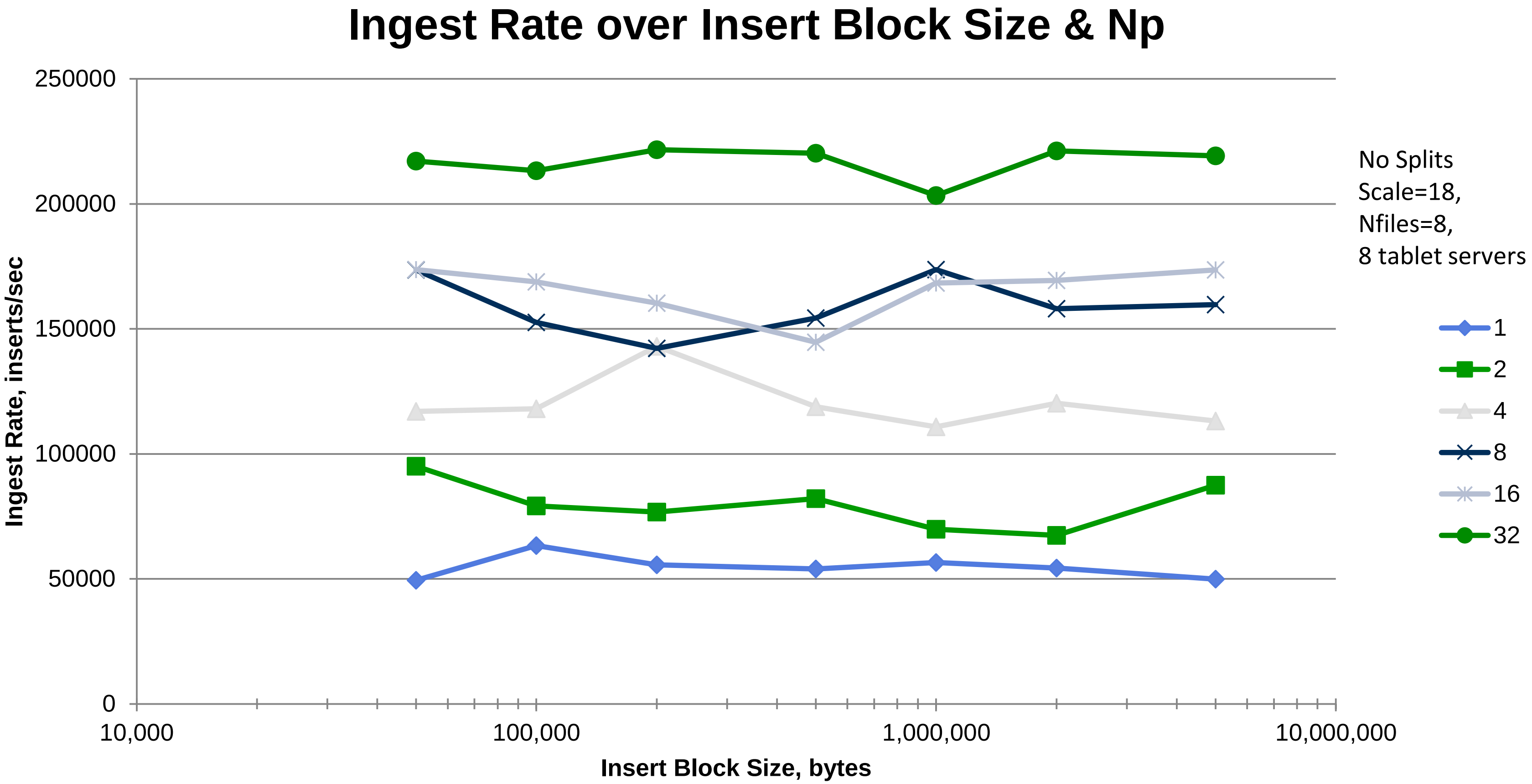




- Goal: Split at points such that we get an even distribution of data among tablets (*load balancing*)
- Challenge: Random power law data
- Solution:
  - Split “string space”
    - 00000000 to 99999999
  - More splits = less error as Accumulo will put sparse tablets on the same tablet server (bottom graph)



D4M Performance: Insert Block Size

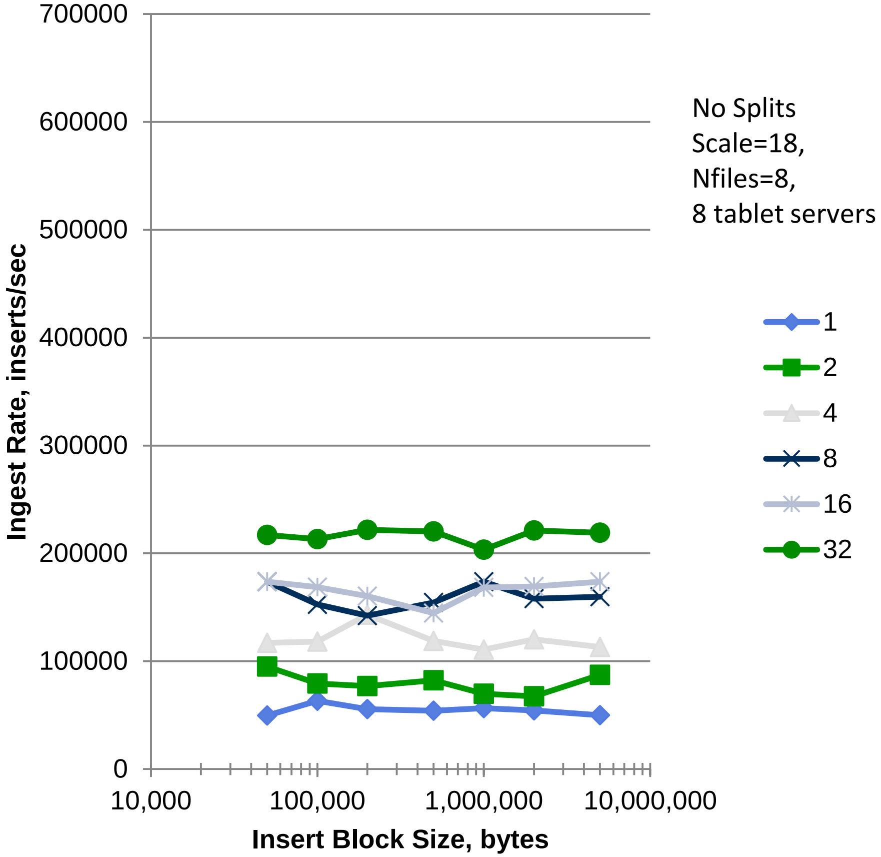


Same setup as before, now varying the insert block size.

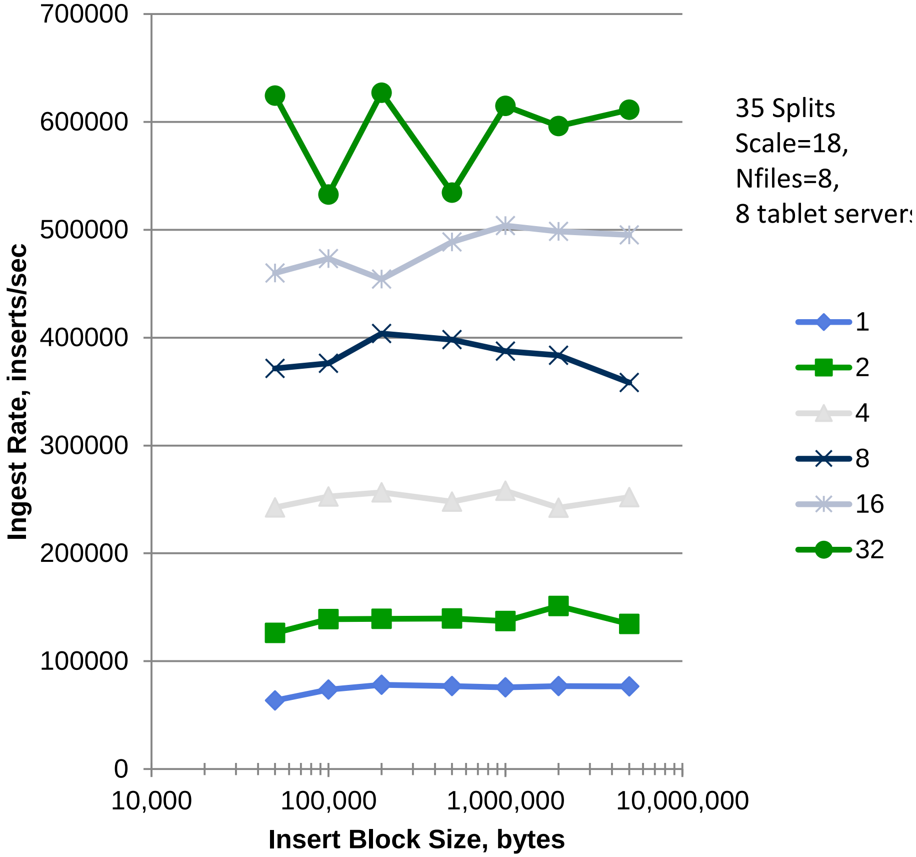
Conclusion: little correlation between insert block size and ingest rate

D4M Performance: Table Splits & Insert Block Size

Without Splits



With 35 Splits



Clear table splitting performance boost across insert block size

Parallel and Distributed Computing

- **General Development: Improving D4M**
  - **What is D4M? Why is it useful?**
    - **Link to Accumulo database**
  - **Some Contributions:**
    - **Accumulator Columns**
    - **Table Splits**
  - **Performance Testing Results**
- **Specific Application: DNA Matching**
  - **Background on Bioinformatics**
  - **DNA Matching Problem**
  - **D4M Solution – Matrices and Assoc Arrays**
  - **Complete Pipeline**
  - **Optimization**
  - **Performance & Results**
- **What’s next?**

DNA Matching

Worked with Darell Ricke of Group 48 -Bioengineering Systems and Technologies

<u>Applications</u>

- **Identification**
- **Mixture Analysis**
- **Kinship Analysis**
- **Ancestry Analysis**

Uses: disease outbreak analysis, criminal investigations, personal services, …

DNA Matching with D4M

- Goal: match a DNA sample against a DB of DNA

Sequence: tagatactgctgcctcccgtagga

Split: tagatactgc

agatactgct

(chunksize 10) gatactgctg

- Substring comparison problem!
- Count # of matches of each sequence chunk
  - Count forward, backward w/ base complement
  - Highest count = highest match chance
- Difficulty: 10GB DNA Fasta file → 260 GB of different sequence data. Need efficient comparison!

Associative Array Approach

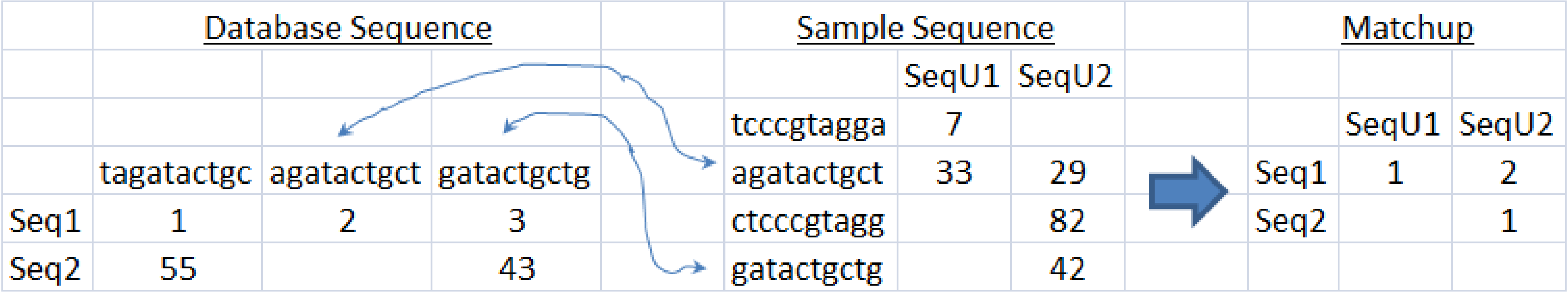
- **Matlab Array of Triples**

(sequence ID, sequence, position in sequence)

(G6J0L4R01AUYU3, tagatactgc, 1)

(G6J0L4R01AUYU3, agatactgct, 2) …

- **Take transpose for fast column lookup.**
- **Multiply to match: A * B’**



Database Sequence

	tagatactgc	agatactgct	gatactgctg
Seq1	1	2	3
Seq2	55		43

Sample Sequence

	SeqU1	SeqU2
tcccgtagga	7	
agatactgct	33	29
ctcccgtagg		82
gatactgctg		42

Matchup

	SeqU1	SeqU2
Seq1	1	2
Seq2		1

Also stored: matchup positions, row & column matchup totals

C Fasta Parser

- **Primary Goal: Generate n-mer words from Fasta data in format Matlab can easily read into an Assoc array**
- **Goal #2: Preprocess & Filter the sequences**
  - **Ignore bad, common sequences**
    - **Too few unique bases**
    - **Unknown bases at end**
  - **Break output files into manageable chunks, say 5MB**
  - **Generate reverse sequences**
  - **Break up big sequences into subsequences to preserve locality**
    - **Matches must occur somewhat close together**
    - **Ex. Break after 10,000 base pairs with overlap of 200**

Parsing & DB Ingest Optimization

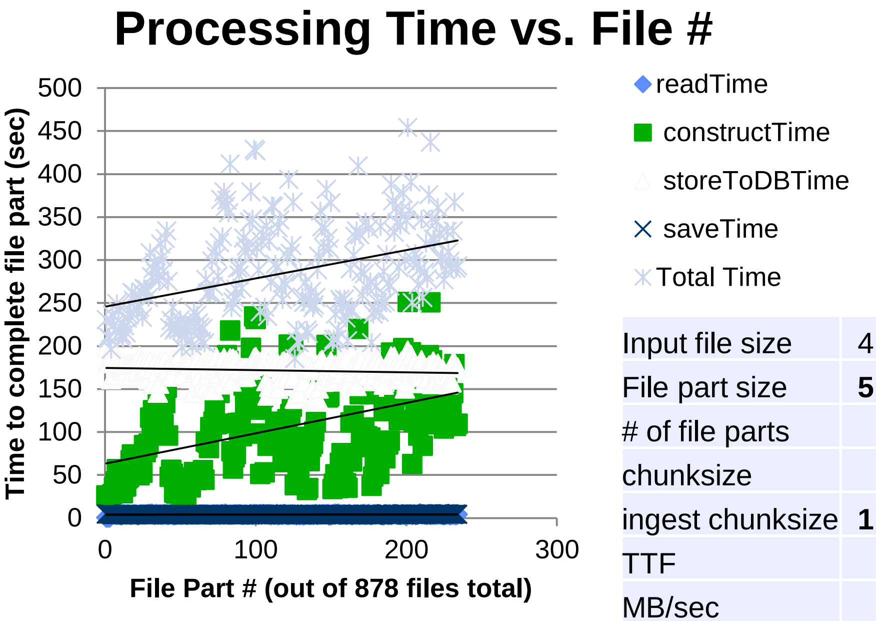
Parameters to optimize

- Input file size
  - Too big = won’t fit in memory
  - Too small = too many files
  - ~5MB file size good
- Ingest chunk size – amt. of data sent to DB at once
  - ~500k worked ok

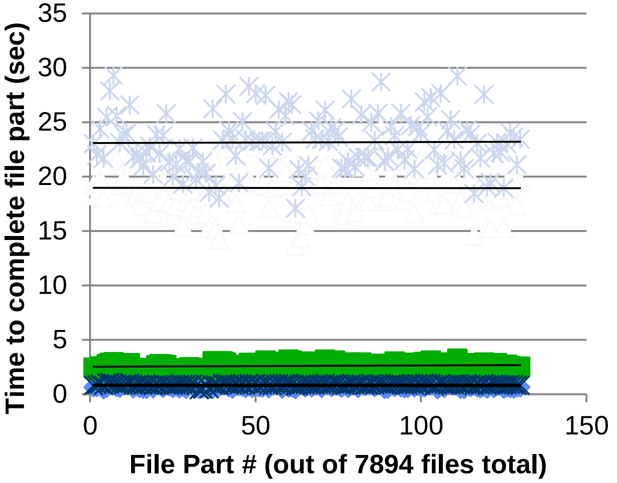
Weird Problem: Assoc construct time increases with file #

- Fixed after adjusting above parameters

Much better!



Input file size	4.82E+09	
File part size	**5.00E+07**	
# of file parts	878	
chunksize	10	
ingest chunksize	**1.00E+06**	
TTF	**69.38**	hrs
MB/sec	0.0193	

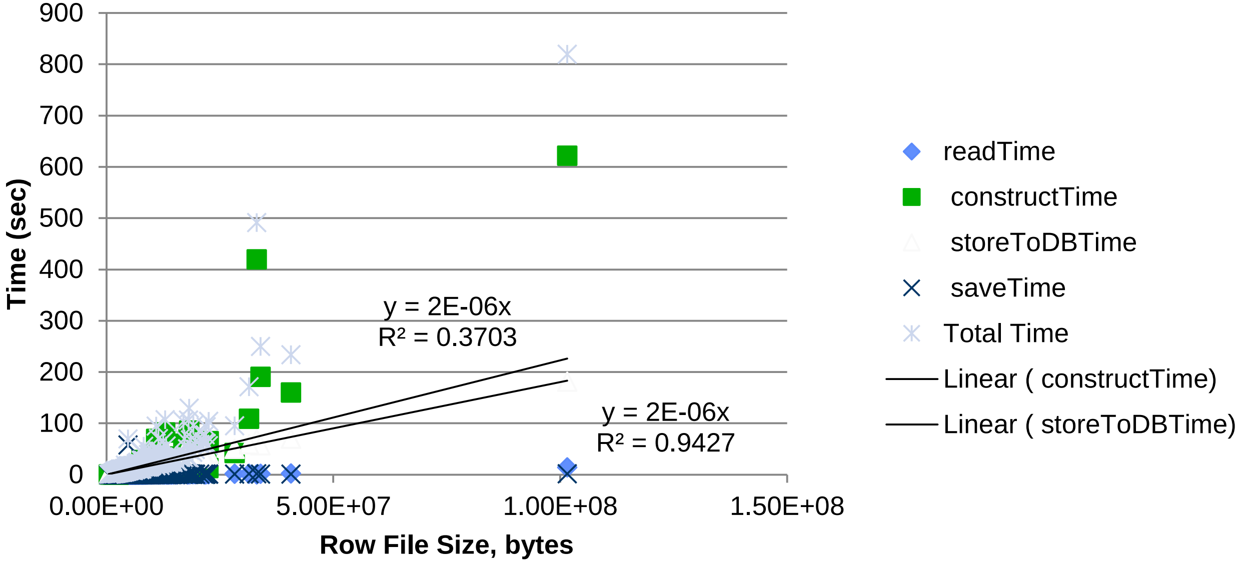


Input file size	4.82E+09	
File part size	**6.20E+06**	
# of file parts	7894	
chunksize	10	
ingest chunksize	**1.00E+05**	
TTF	**50.74**	hrs
MB/sec	0.0264	

File Size Optimization

- **Some virus sequences are really big → big, outlier output files**
- **Huge, nonlinear jump in Assoc construction time**
- **<u>Solution</u>: break DNA in middle of a sequence**

Time Spent Processing File vs. File Size



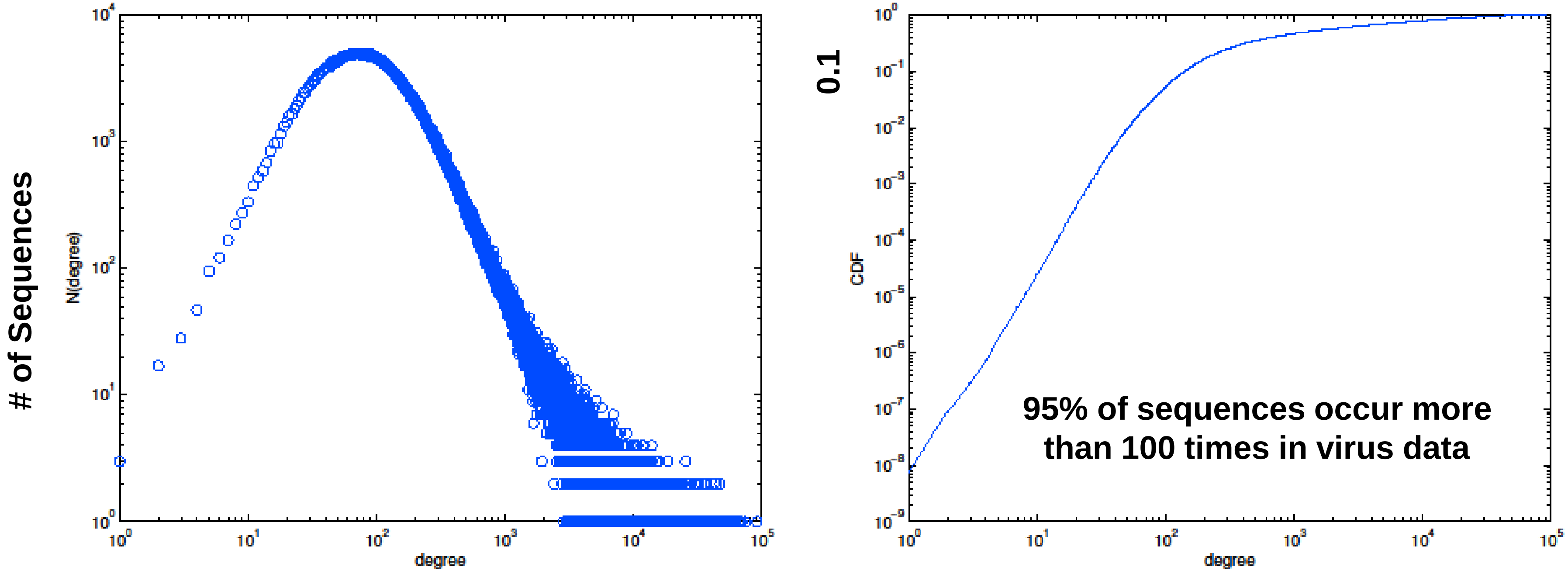
chunksize	10	
# of file splits:	770	
# of sequences	384663	
Input file size:	5.85E+08	Total size
Row avg file size:	6.32E+06	4.87E+09
Col average file size:	6.38E+06	4.92E+09
Val average file size:		0.00E+00
TOTAL output file size		9.78E+09
Output/Input Size	16.72	
TTF	13126.2 sec	
	3.64618 hrs	
MB/sec	0.0446	

Pipeline Performance

- 586 MB Virus data, 23 MB Sample data

It’s the rare ones!

- Can eliminate 10mers occurring in the reference DB > 100 times (95% of all 10mers) and still correctly match top results
- Implication: Significantly smaller # of entries to scan
  Can reduce database size and search space



95% of sequences occur more than 100 times in virus data

Sequence Occurrence in Reference DB

Conclusion

- **Do you have a ton of data to process?**
- **Are you limited by time or memory?**

➔contact Group 110 and the D4M team!

- **Somewhere mention the technique applies to protein sequences too**

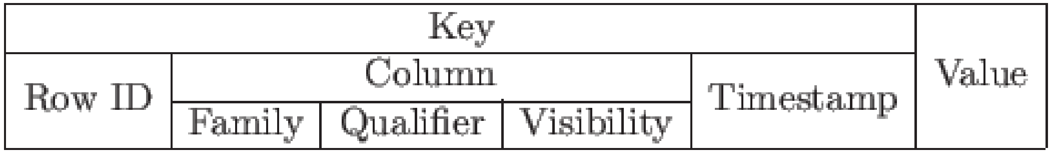
Backup!

- **Index the data in a DB! Fast lookup, comparison**
  1. **Parse big .fa Fasta file → all sequence chunks in D4M-friendly format – .row, .col, .val files**
  2. **Ingest data into DB**
  3. **Parse sample .fa Fasta file → .row, .col, .val**
  4. **Load sample data & compare against data in DB**

Overview

- ➢ **Upgrading D4M support for Accumulo**
  - **Accumulator / combiner columns**
  - **Table Splits**
  - **Deleting Triples**
  - **Performance Testing**
- **D4M Bioinformatics Application**
  - **Background & Approach**
  - **Optimization & Quirks**
  - **Performance & Results**

Accumulo Background



Key					Value
Row ID	Column			Timestamp	
	Family	Qualifier	Visibility		

- Triple Store DB:
- Cell-level access control
- Distributed, Open Source, Java
  - Uses Hadoop File System and Zookeeper Node Mgmt.

D4M Background

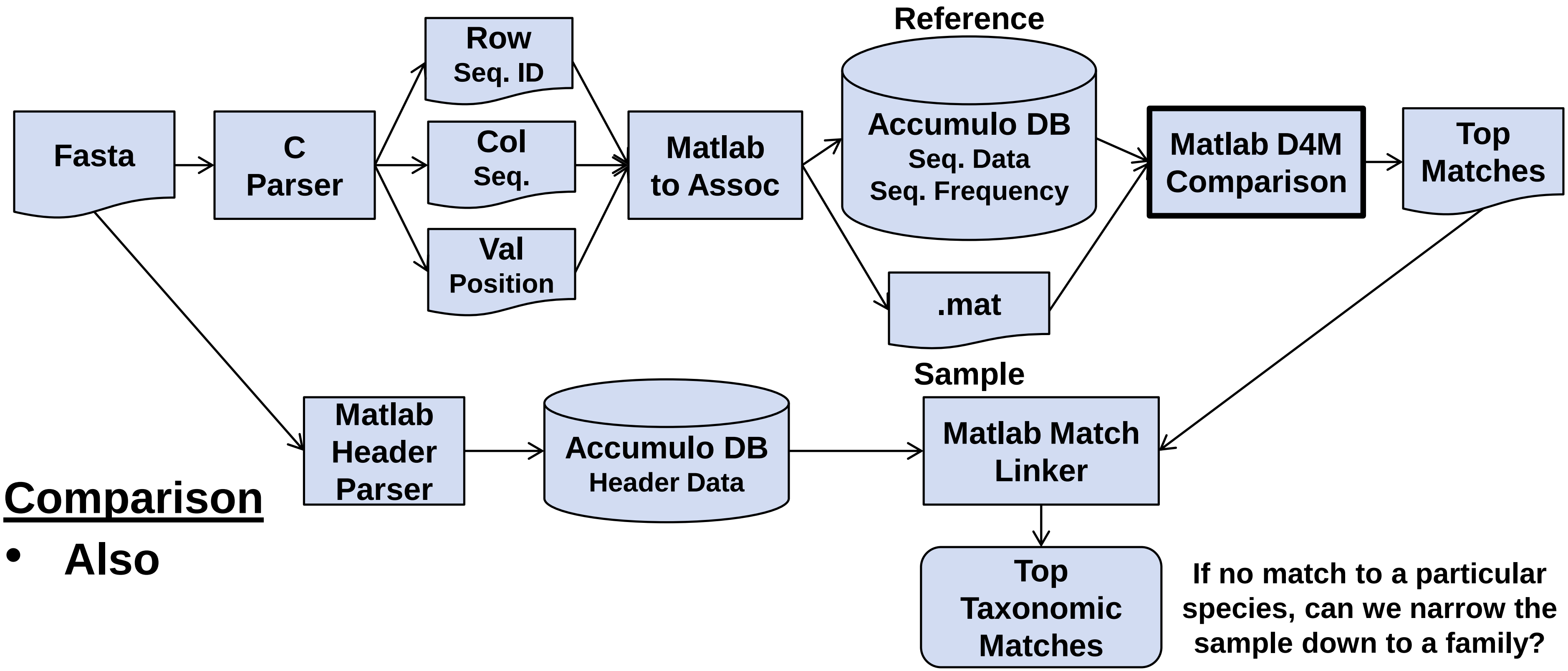
- Dynamic Distributed Dimensional Data
- Lincoln Lab development
- Matlab Software Package
  - Linear Algebra & Graph Theory support
  - Associative Array

- Usage: Text and cyber analytics
  - Lincoln Labs PLSA Cloud Knowledge Service uses Accumulo

Performance Testing Intro

- The hallmark of parallel and distributed computing
- pMatlab – Parallel Matlab
  - Divide work to many nodes using distributed arrays
  - Lincoln Laboratory product
- Multiple nodes insert vast quantities of data into Accumulo instance
- Matlab timing functions: tic toc
  - Measure ingest rate in inserts/sec
- Rigorously vary parameter, measure times

Build Sequence Processing Pipeline



Fasta

C Parser

Row Seq. ID

Col Seq.

Val Position

Matlab to Assoc

Reference

Accumulo DB
Seq. Data
Seq. Frequency

.mat

Matlab D4M Comparison

Top Matches

Matlab Header Parser

Accumulo DB
Header Data

Sample

Matlab Match Linker

Top Taxonomic Matches

If no match to a particular species, can we narrow the sample down to a family?

Comparison

- Also