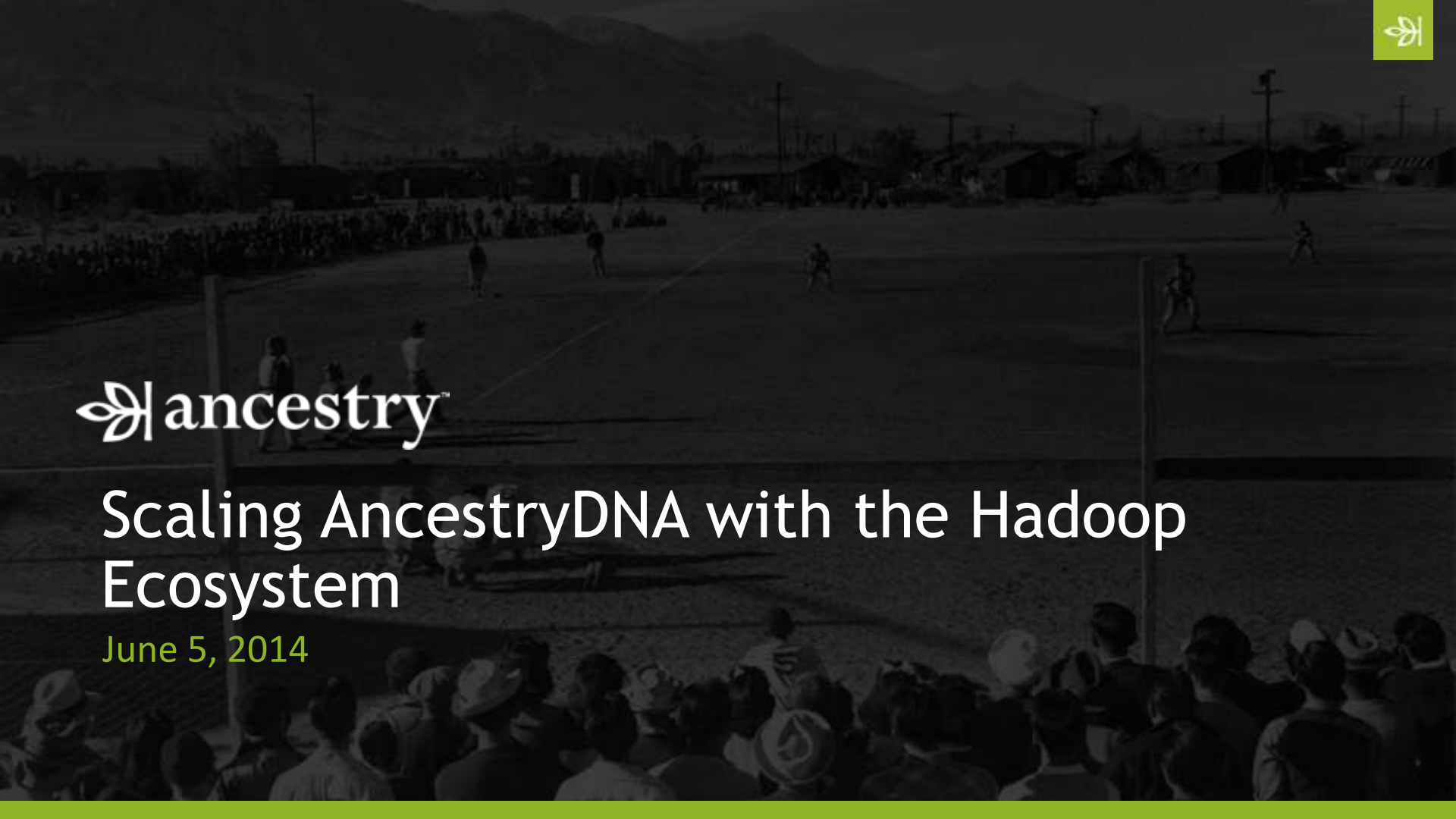
EXHIBIT

8



Scaling AncestryDNA with the Hadoop Ecosystem

June 5, 2014

What Ancestry uses from the Hadoop ecosystem

- Hadoop, HDFS, and MapReduce

- HBase
  - Columnar, NoSQL data store, unlimited rows and columns

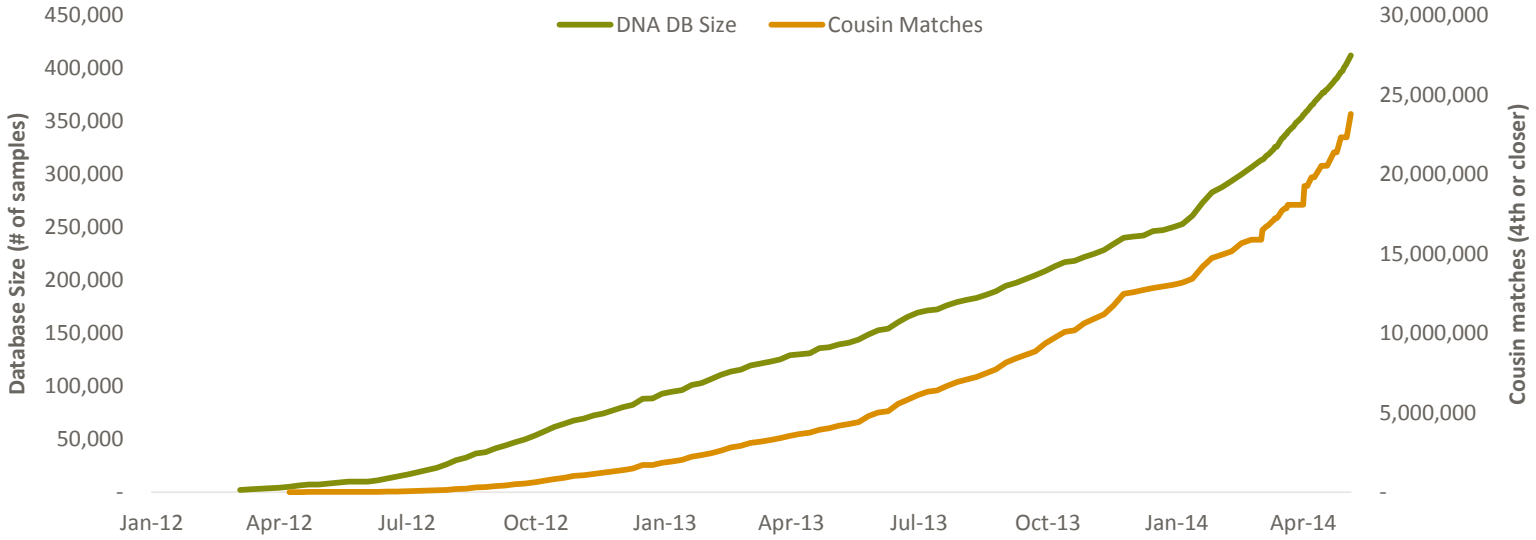
- Azkaban
  - Workflow

What will this presentation cover?

- Describe the problem
  - Discoveries with DNA
  - Three key steps in the pipeline process
- Measure everything principle
- Three steps with Hadoop
  - Hadoop as a job scheduler for the ethnicity step
  - Scaling matching step
  - MapReduce implementation of phasing
- Performance
- What comes next?

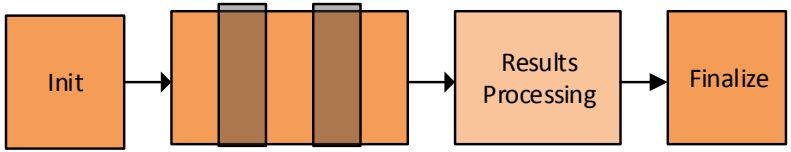
Discoveries with DNA

- Autosomal DNA test that analyzes 700,000 SNPs
- Over 400,000 DNA samples in our database
- Identified 30 million relationships that connect the genotyped members through shared ancestors



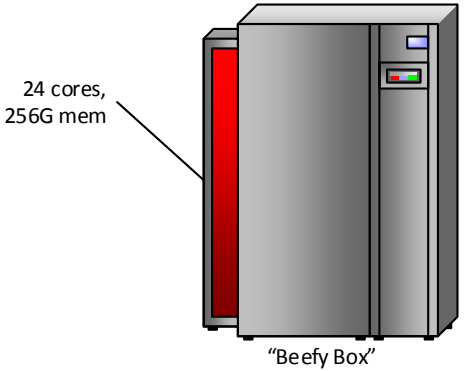
Three key steps in the pipeline

- What is a pipeline?



1. Ethnicity (AdMixture)
2. Matching (GERMLINE and Jermline)
3. Phasing (Beagle and Underdog)

- First pipeline executed on a single, beefy box.
  - Only option is to scale ***vertically***



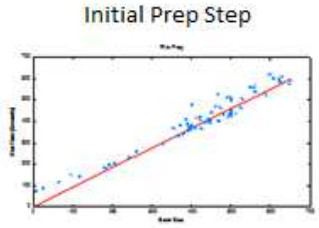
Measure everything principle

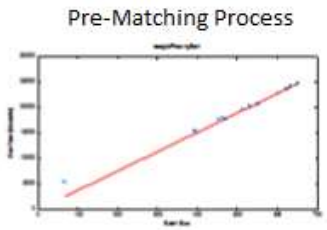
- Start time, end time, duration in seconds, and sample count for every step in the pipeline. Also the full end-to-end processing time.
- Put the data in pivot tables and graphed each step
- Normalize the data (sample size was changing)
- *Use the data collected to predict future performance*

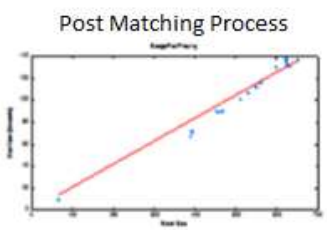
Challenges and pain points

Performance degrades when DNA pool grows

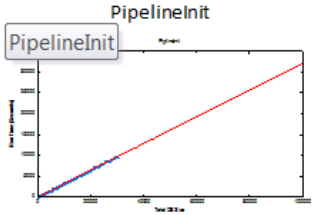
- Static (by batch size)



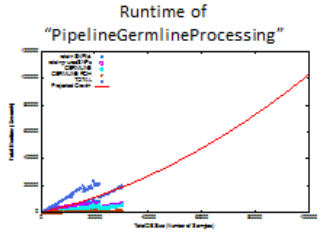


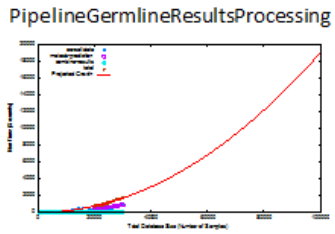


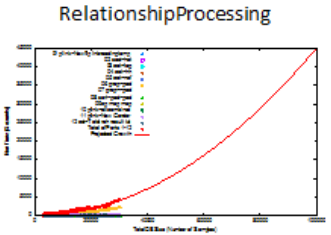
- Linear (by DNA pool size)



- Quadratic (matching related steps) – time bomb







Ethnicity step on Hadoop

Using Hadoop as a job scheduler to scale AdMixture

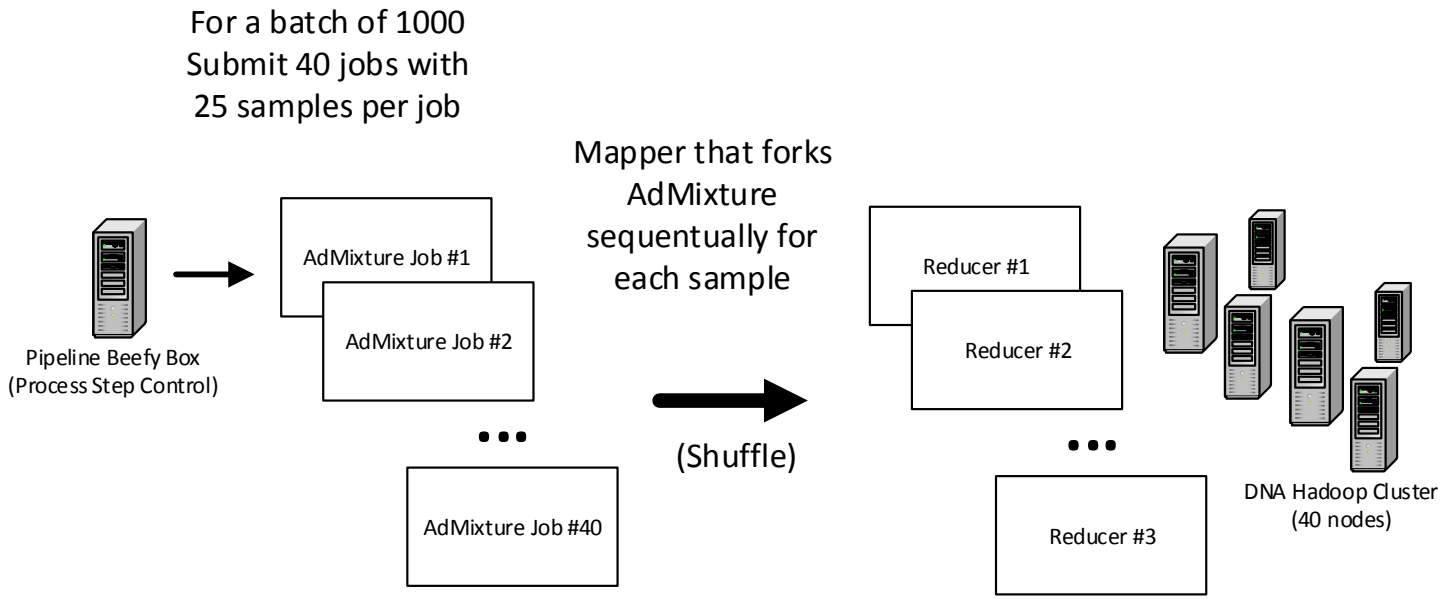
First step with Hadoop

- What was in place?
  - Smart engineers with no Hadoop experience
  - Pipeline running on a single computer that would ***not*** scale
  - New business that needed a scalable solution to ***grow***

First step using Hadoop

- Run AdMixture step in parallel on Hadoop
  - Self contained program with set inputs and outputs
  - Simple MapReduce implementation
  - Experience running jobs on Hadoop
  - Freed up CPU and memory on the single computer for the other steps

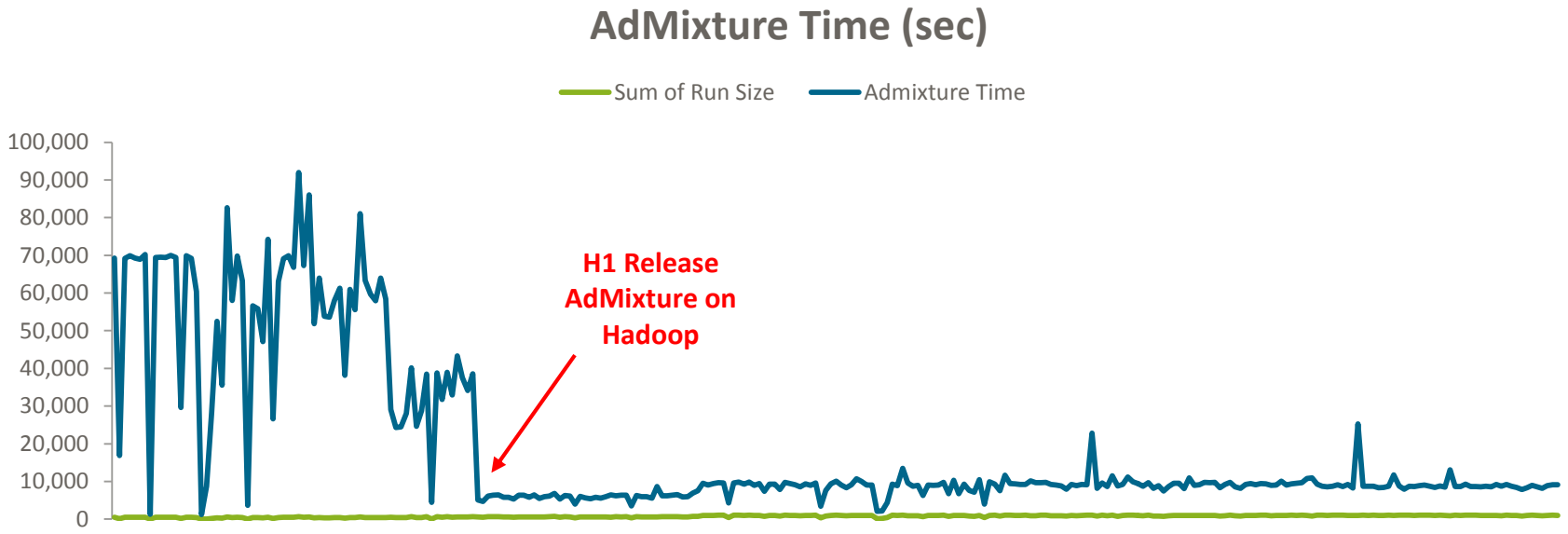
What did we do? (Don’t cringe…)



1. Mapper -> Key: User ID, Ethnicity Result
2. Reducer -> Key: User ID, Array [Ethnicity Result]

Performance results

- Went from processing 500 samples in 20 hours
  to processing 1,000 samples in 2 ½ hours
- Reduced Beagle phasing step by 4 hours



Provided valuable experience and bought us time

GERMLINE to Jermline

Moving the matching step to MapReduce and HBase

Introducing … GERMLINE!

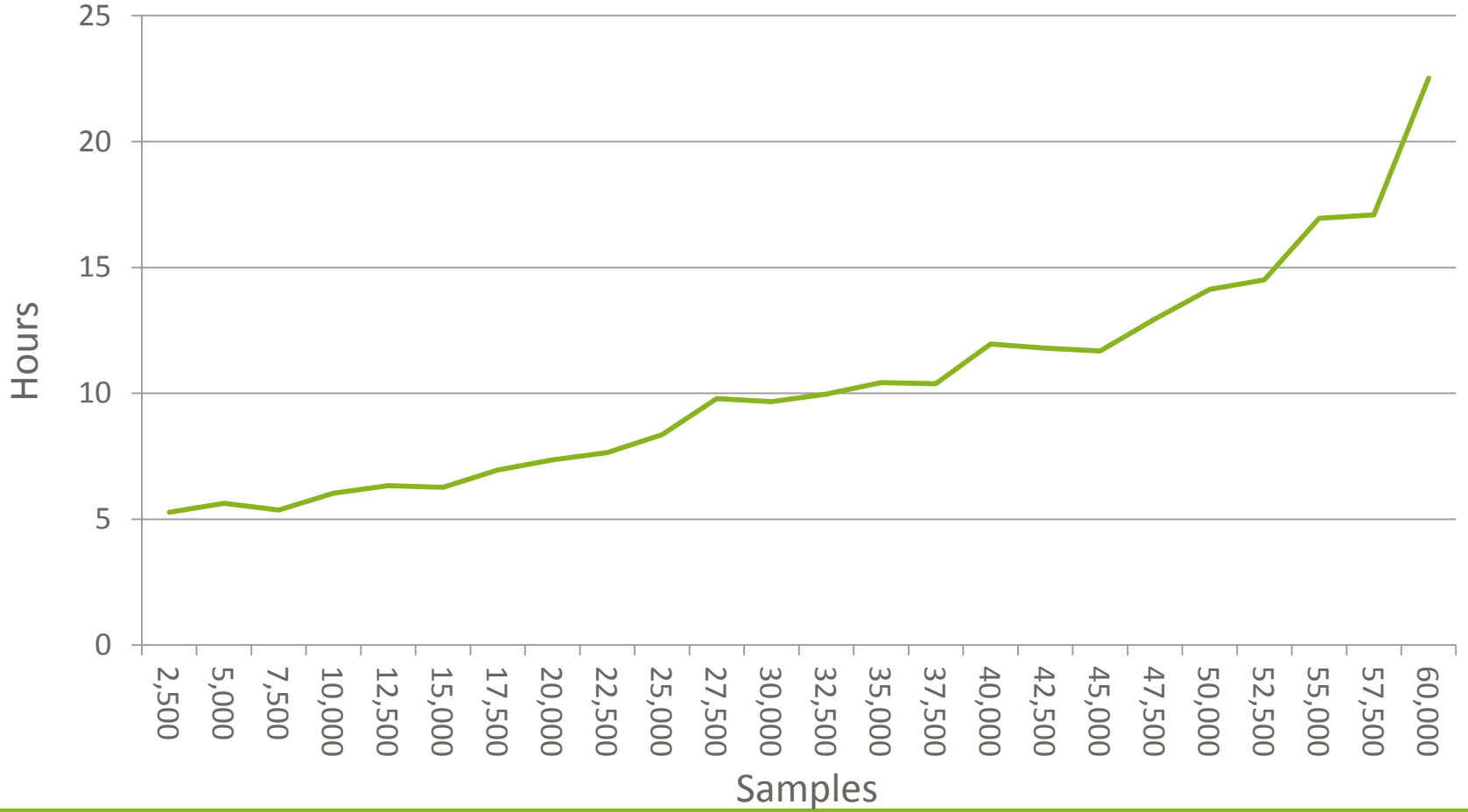
- GERMLINE is an algorithm that finds hidden relationships within a pool of DNA
- Also refers to the reference implementation of that algorithm written in C++. You can find it here:

  http://www1.cs.columbia.edu/~gusev/germline/

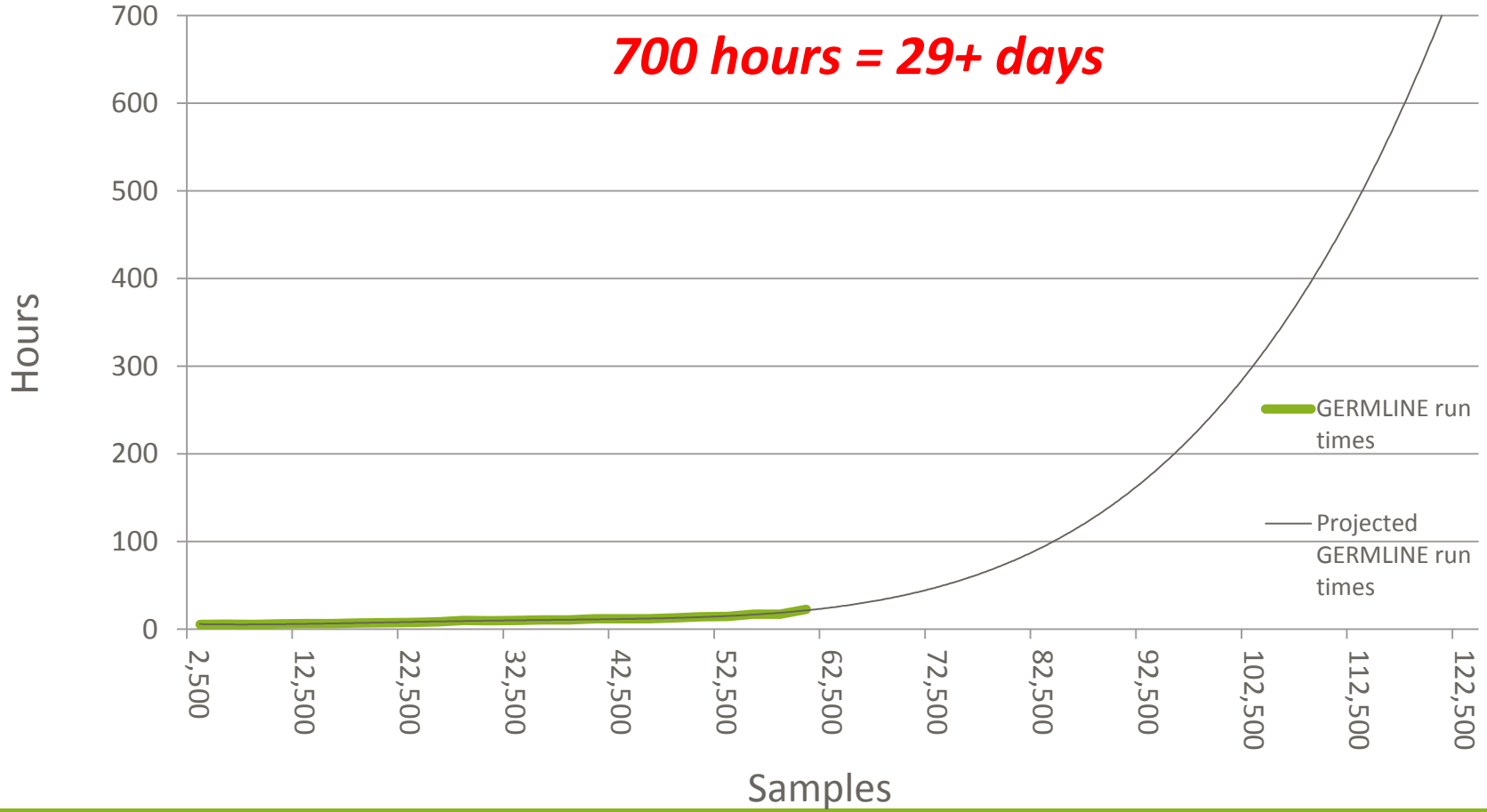
So what's the problem?

- GERMLINE (the implementation) was not meant to be used in an industrial setting
  - Stateless, single threaded, prone to swapping (heavy memory usage)
  - GERMLINE performs poorly on large data sets
- Our metrics predicted exactly where the process would slow to a crawl
- Put simply: GERMLINE couldn't scale

GERMLINE run times (in hours)



Projected GERMLINE run times (in hours)



DNA matching walkthrough

Simplified example of showing how the code works

DNA matching : How it works

Cersei Baratheon

- Former queen of Westeros
- Machiavellian manipulator
- Mostly evil, but occasionally sympathetic

The Input

Cersei : ACTGACCTAGTTGAC
Joffrey : TTAAGCCTAGTTGAC

Joffrey Baratheon

- Pretty much the human embodiment of evil
- Needlessly cruel
- Kinda looks like Justin Bieber

DNA matching : How it works

Separate into words

	0	1	2
Cersei	ACTGA	CCTAG	TTGAC
Joffrey	TTAAG	CCTAG	TTGAC

DNA matching : How it works

Build the hash table

	0	1	2
Cersei	ACTGA	CCTAG	TTGAC
Joffrey	TTAAG	CCTAG	TTGAC

ACTGA_0 : Cersei
TTAAG_0 : Joffrey
CCTAG_1 : Cersei, Joffrey
TTGAC_2 : Cersei, Joffrey

DNA matching : How it works

Iterate through genome and find matches

0 1 2

Cersei : ACTGA CCTAG TTGAC

Joffrey : TTAAG CCTAG TTGAC

ACTGA_0 : Cersei

TTAAG_0 : Joffrey

CCTAG_1 : Cersei, Joffrey

TTGAC_2 : Cersei, Joffrey

Cersei and Joffrey match from position 1 to position 2

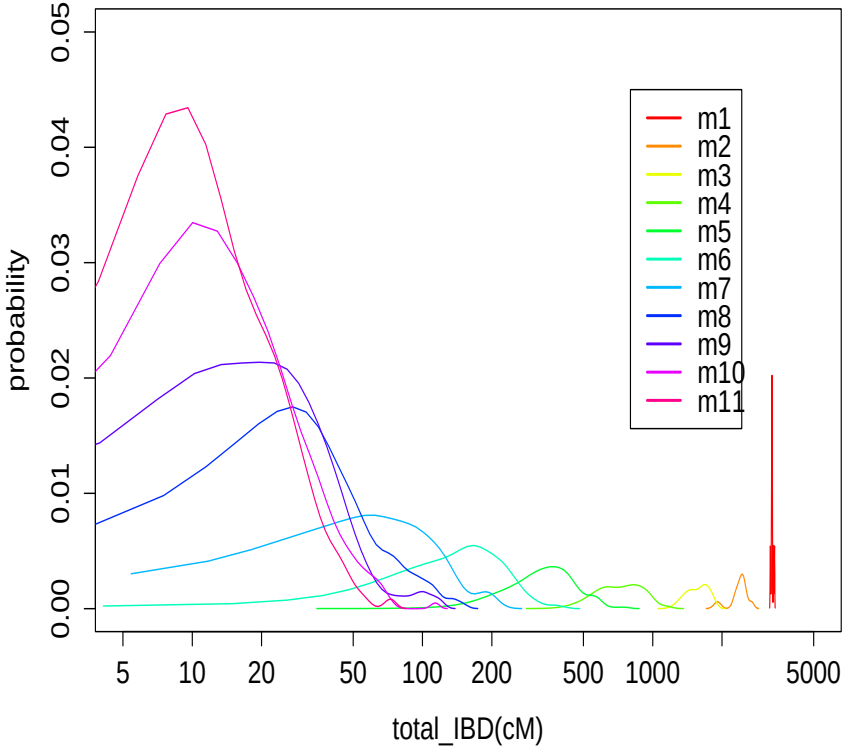
Does that mean they're related?

?

...maybe

IBD to relationship estimation

- We use the total length of all shared segments to estimate the relationship between two genetic relatives
- This is basically a classification problem



But wait...what about Jaime?

Jaime : TTAAGCCTAGGGGCG

Jaime Lannister

- Kind of a has-been
- Killed the Mad King
- Has the hots for his sister, Cersei

Step one: Update the hash table

	Cersei	Joffrey
2_ACTGA_0	1	
2_TTAAG_0		1
2_CCTAG_1	1	1
2_TTGAC_2	1	1

Already stored in HBase

Jaime : TTAAG CCTAG GGGCG

New sample to add

Key : [CHROMOSOME]_[WORD]_[POSITION]
Qualifier : [USER ID]
Cell value : A byte set to 1, denoting that the user has that word at that position on that chromosome

The jermline way

Step two: Find matches, update the results table

	2_Cersei	2_Joffrey
2_Cersei		{ (1, 2), ...}
2_Joffrey	{ (1, 2), ...}	

Already stored in HBase

Jaime and Joffrey match from position 0 to position 1
Jaime and Cersei match at position 1

New matches to add

Key : [CHROMOSOME]_[USER ID]
Qualifier : [CHROMOSOME]_[USER ID]
Cell value : A list of ranges where the two users match on a chromosome

The jermline way

Hash Table			
	Cersei	Joffrey	Jaime
2_ACTGA_0	1		
2_TTAAG_0		1	1
2_CCTAG_1	1	1	1
2_TTGAC_2	1	1	
2_GGGCG_2			1

Results Table			
	2_Cersei	2_Joffrey	2_Jaime
2_Cersei		{ (1, 2), ...}	{ (1), ...}
2_Joffrey	{ (1, 2), ...}		{ (0,1), ...}
2_Jaime	{ (1), ...}	{ (0,1), ...}	

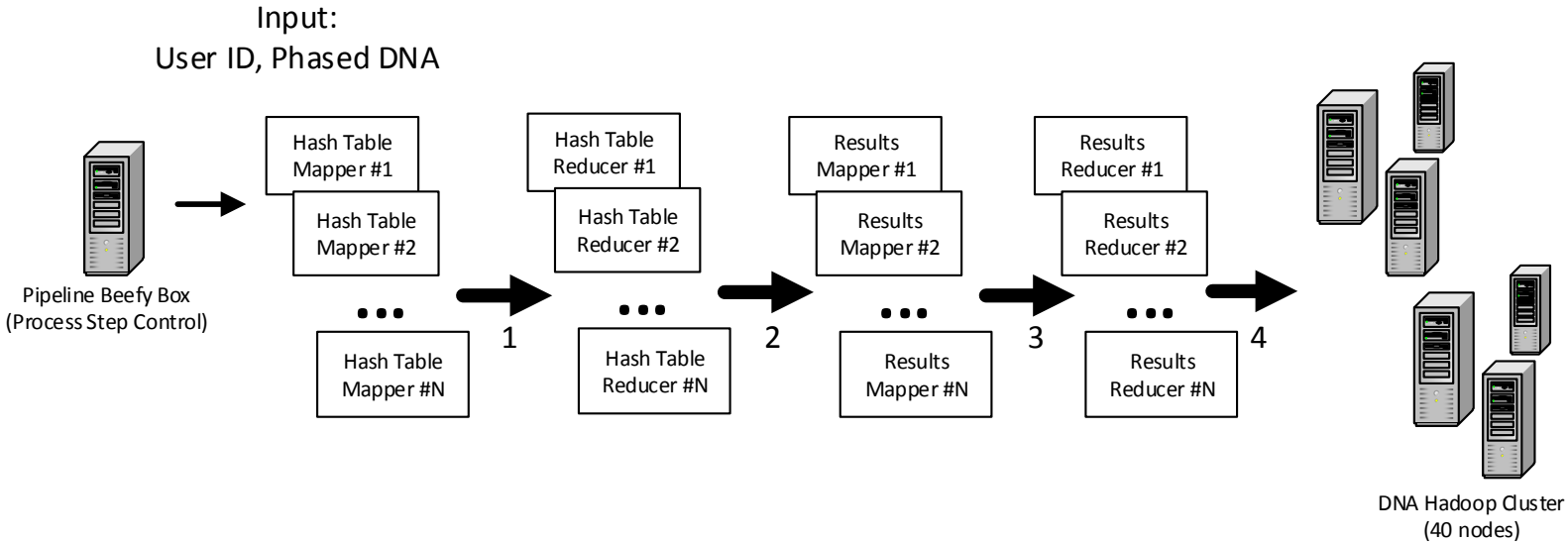
But wait...what about Daenerys, Tyrion, Arya, and Jon Snow?

Run them in parallel with Hadoop!

Parallelism with Hadoop

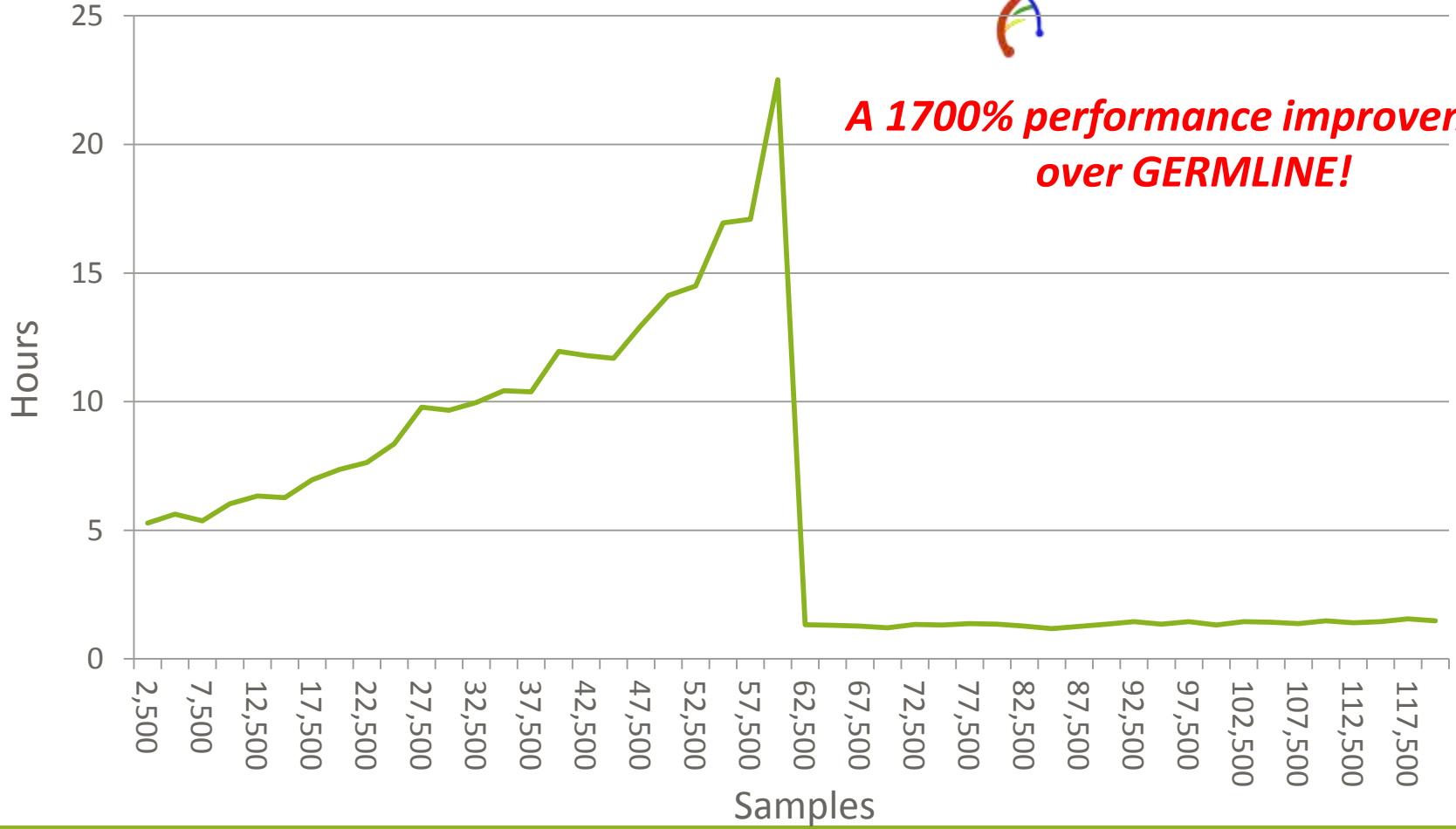
- Batches are usually about a thousand people
- Each mapper takes a single chromosome for a single person
- MapReduce jobs:
  - Job #1: Match words
    - Updates the hash table
  - Job #2: Match segments
    - Identifies areas where the samples match

Matching steps on Hadoop



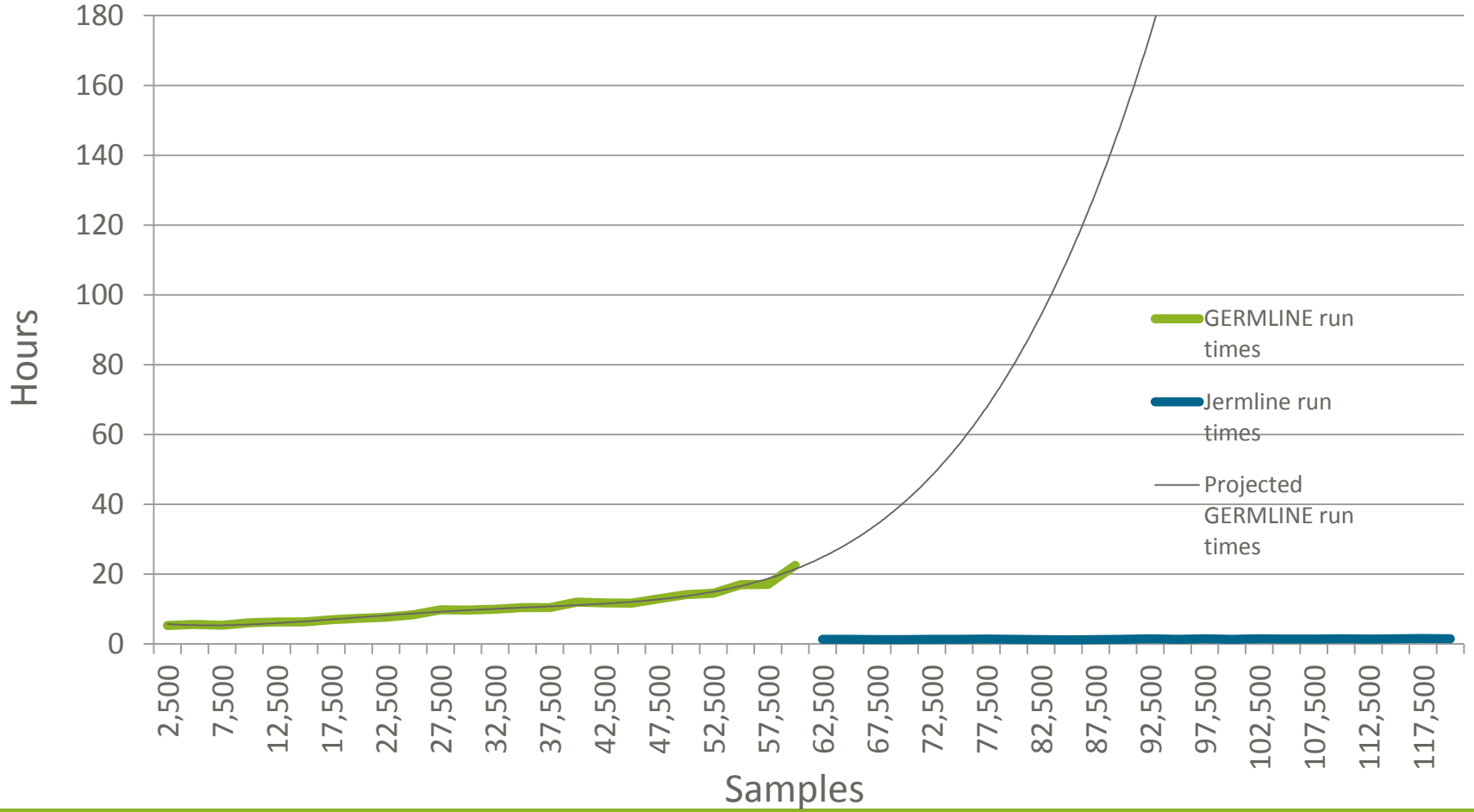
1. Hash Table Mapper -> Breaks input into words and fills the hash table (HBase Table #1)
2. Hash Table Reducer -> default reducer (does nothing)
3. Results Mapper -> For each new user, read hash table, fill in the results table (HBase Table #2)
4. Results Reducer -> Key: Object(User ID #1, User ID #2), Array[Object(chrom + pos, Matching DNA Segment)]

Run times for matching with jermline



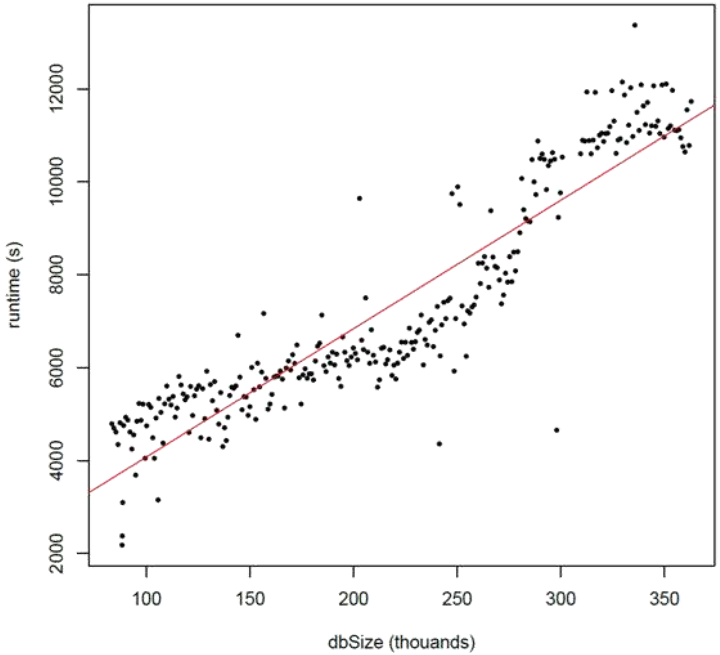
A 1700% performance improvement over GERMLINE!

Run times for matching (in hours)



Jermline performance

- Science team is sure the Jermline algorithm is linear



- Improving the accuracy
  - Found a bug in original C++ reference code
  - Balancing false positives and false negatives
- Binary version of Jermline
  - Use less memory and improve speed
- Paper submitted describing the implementation
  - Releasing as an Open Source project soon

Beagle to Underdog

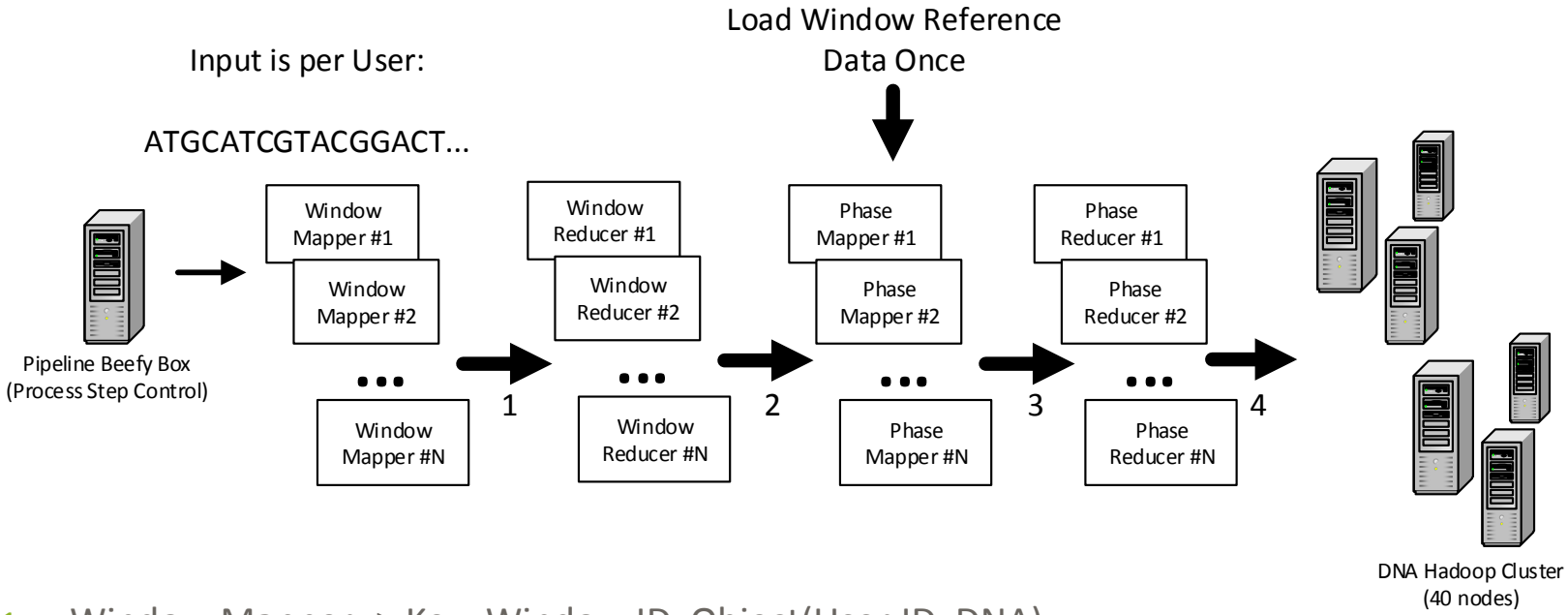
Moving phasing step from a single process to MapReduce

Phasing goes to the dogs

- Beagle
  - Open source, freely available program
  - Multi-threaded process that runs on one computer
  - More accurate with a large sample set

- Underdog
  - Does the same statistical calculations with a larger reference set, which increases accuracy
  - Carefully split into a MapReduce implementation that allows parallel processing
  - Collaboration between the DNA Science and Pipeline Developer Teams

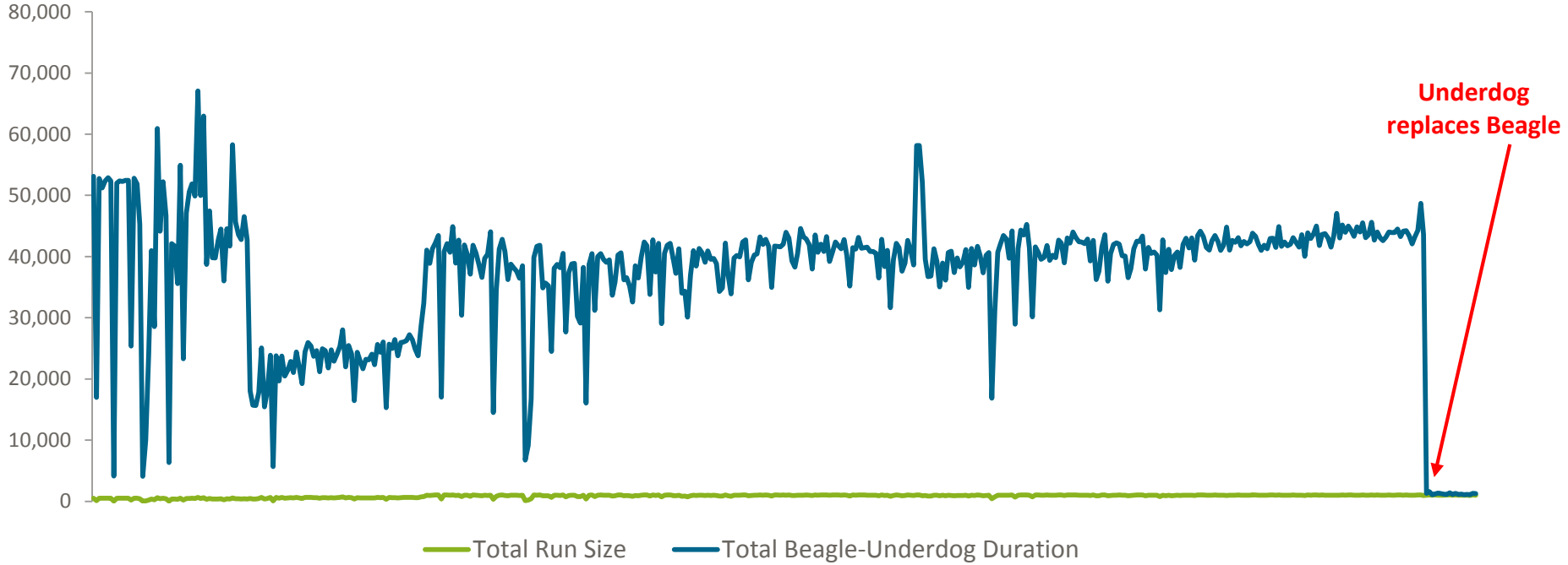
What did we do?



1. Window Mapper -> Key: Window ID, Object(User ID, DNA)
2. Window Reducer -> Key: Window ID, Array[Object(User ID, DNA)]
3. Phase Mapper (loads window data) -> Key: User ID, Object(Window ID, Phased DNA)
4. Phase Reducer -> Key: User ID, Array[Object(Window ID, Phased DNA)]

Underdog performance

- Went from 12 hours to process 1,000 samples
  to under 25 minutes with a MapReduce implementation



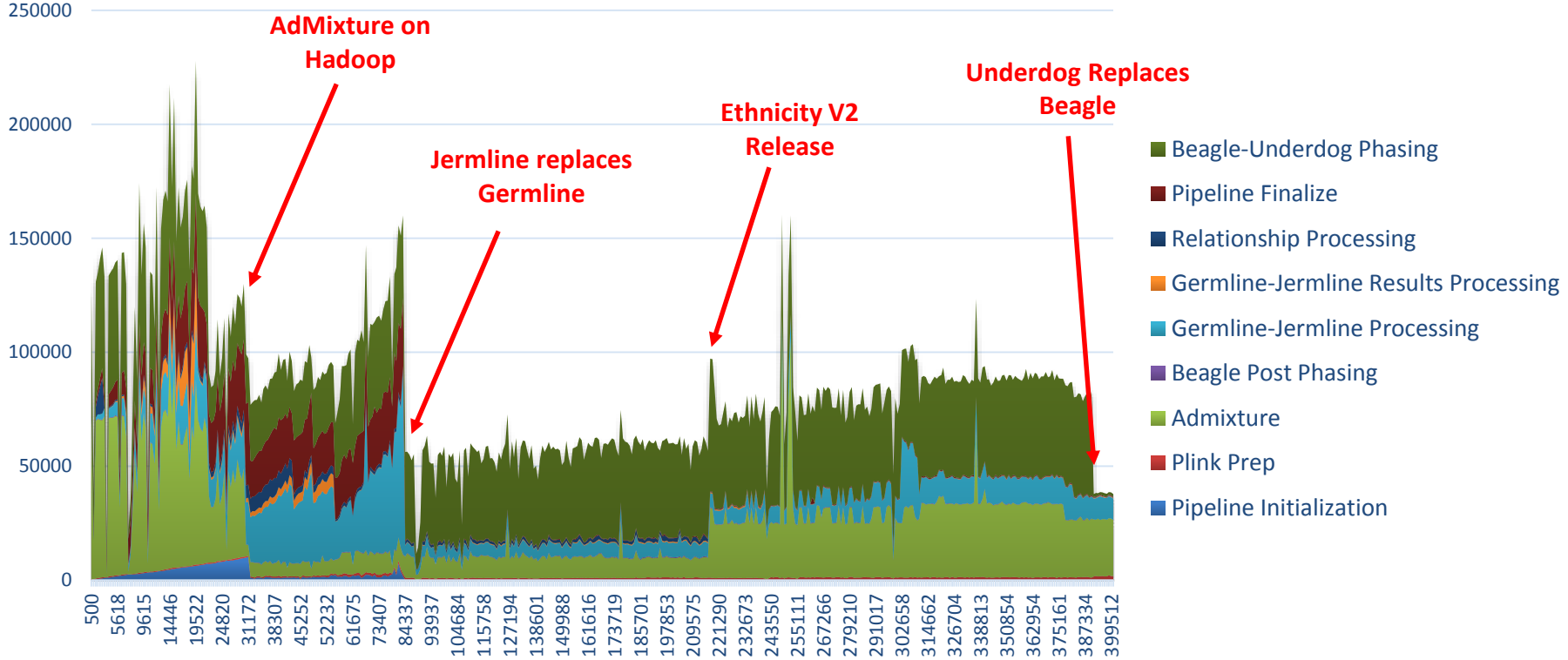
With improved accuracy!

Performance and next steps

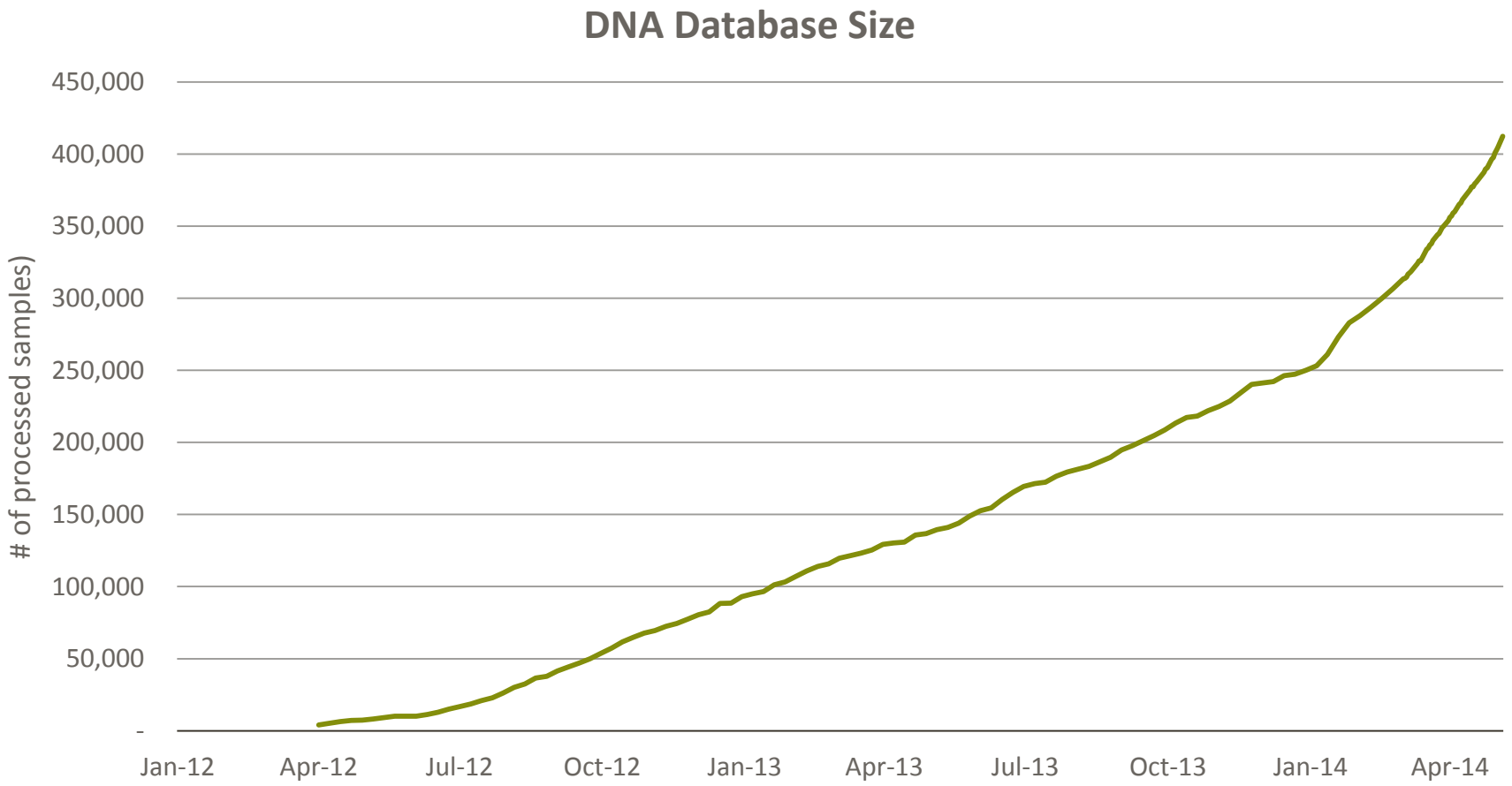
Incremental change

Pipeline steps and incremental change…

- Incremental change over time
- Supporting the business in a “just in time” Agile way



…while the business continues to grow rapidly



What's next? Building different pipelines

- Azkaban

  - Allows us to easily tie together steps on Hadoop
  - Drop different steps in/out and create different pipelines
  - Significant improvement over a hand coded pipeline

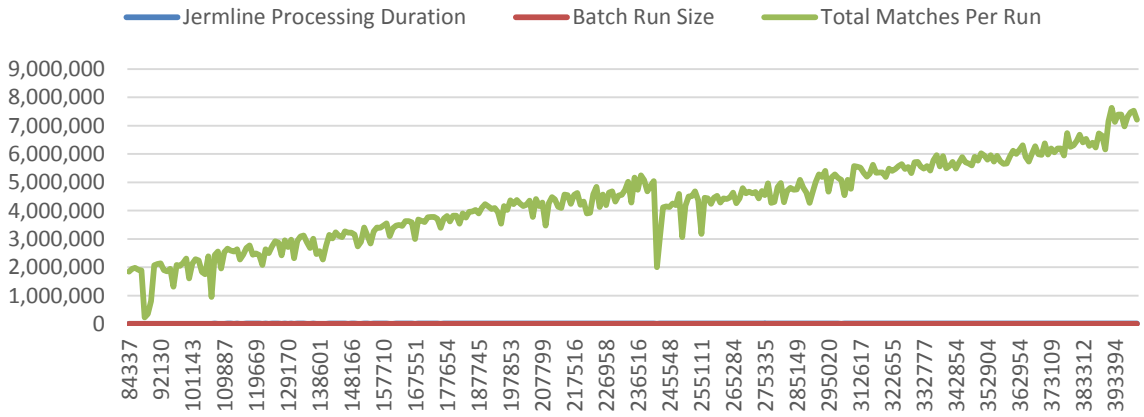
- Cloud

  - New algorithm changes will force a complete re-run of the entire DNA pool
  - Best example: New matching or ethnicity algorithm will force us to reprocess 400K+ samples
  - Solution: Use the cloud for this processing while the current pipeline keeps chugging along

What's next? Other areas for improvement

- Admixture as a MapReduce implementation
  - Last major algorithm that needs to be addressed
  - Expect to get performance improvements similar to Underdog

- Matching growth will cause problems
  - Matches per run increasing
  - Change the handoff



Keep measuring everything and adjust

Questions?

Ancestry is hiring for the DNA Pipeline Team!

Tech Roots Blog: http://blogs.ancestry.com/techroots

byetman@ancestry.com

Special thanks to the DNA Science and Pipeline Development Teams at Ancestry