Microsoft

Interactive Genomics: Querying Genomes in the Cloud

Name: George Varghese
(with Bafna, Kozanitis, Pandya)
Affiliation: MSR and UCSD

Microsoft Research
Faculty Summit

Problem Statement

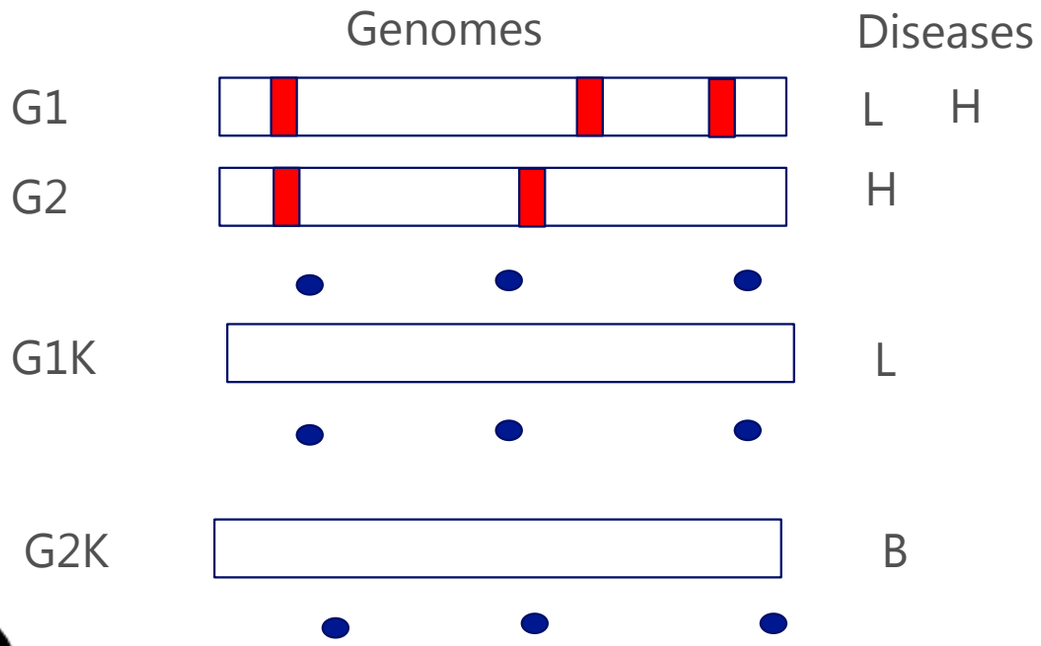
- Genomic Hardware/Data Revolution:
  - Hardware costs falling: <$1000 soon?
  - more genomic data produced: millions soon?
  - Electronic medical records soon: HITECH act
  - Cancer genomics hot: Gleevec, Herceptin
- Genomic Software issues/bottlenecks
  - Batch oriented software (days for analysis)
  - Frameworks, script oriented, hard to write
  - Sharing rare

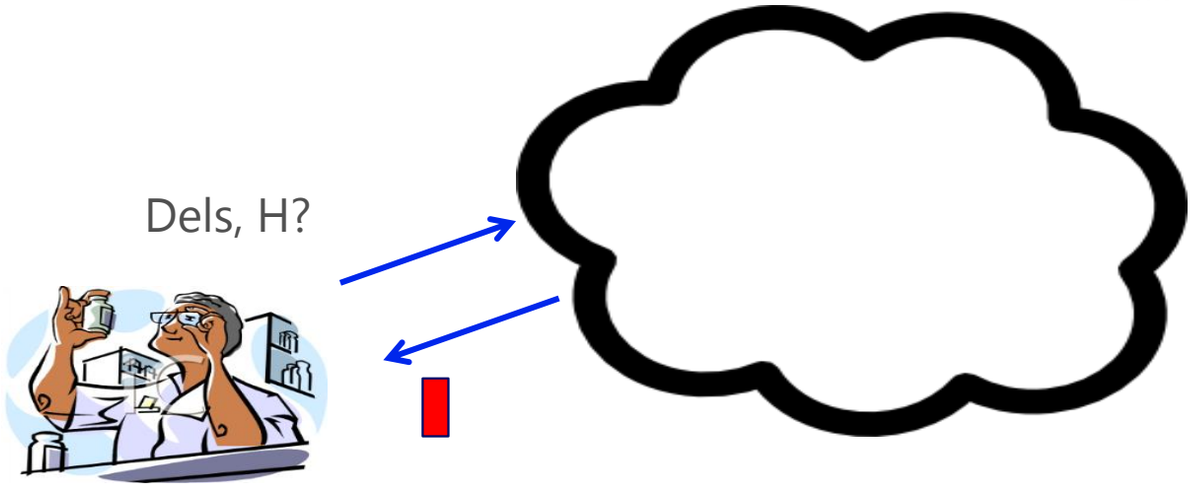
What abstractions can help genomic software?

Our proposal

- Interactive genomics: querying genomic data to quickly remove fruitless hypotheses
- Layering: Separate probabilistic inference from deterministic evidence gathering
- Operators: 3 specific operators that abstract noise-tolerant interval computation
- Optimizations: Materialized views, lazy joins . . .
- Prototype: 60 seconds for deletion query on Azure Cloud. 20x more concise, 8x faster than GATK.

Idea 1: Interactive Genomics





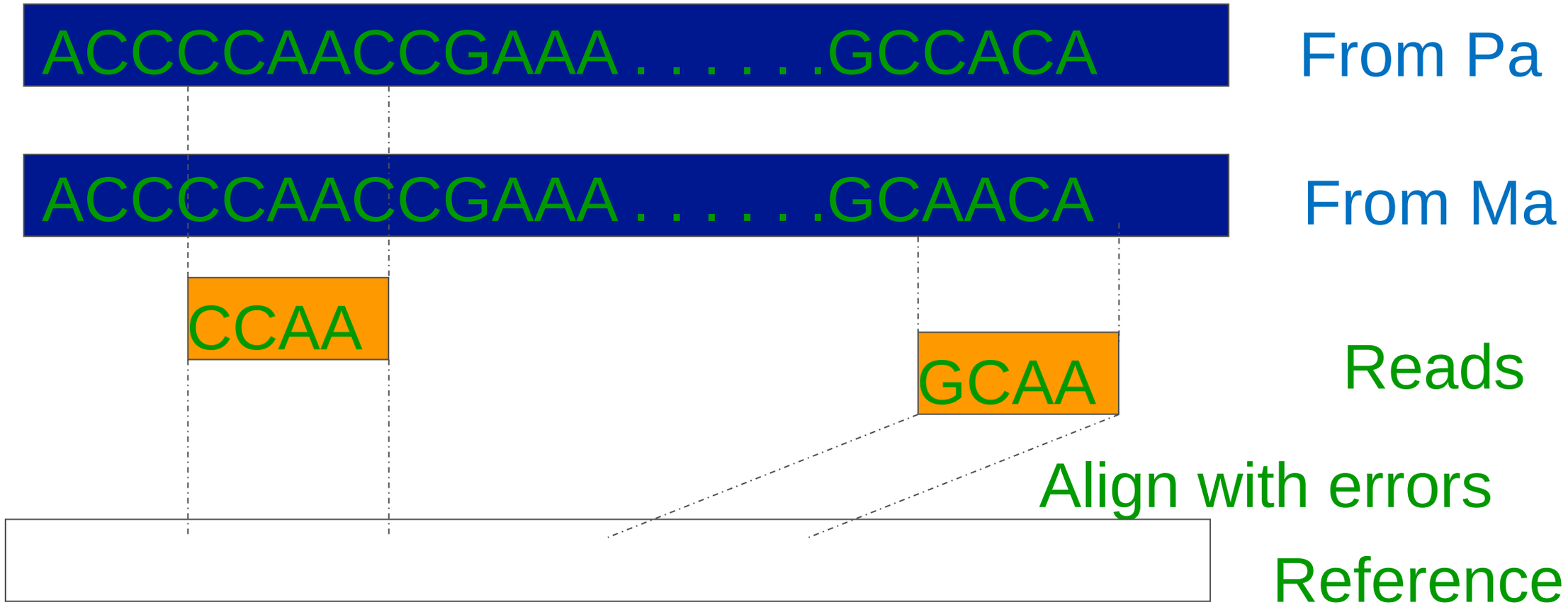
Variation, Disease→ Locations

Discovery

Background

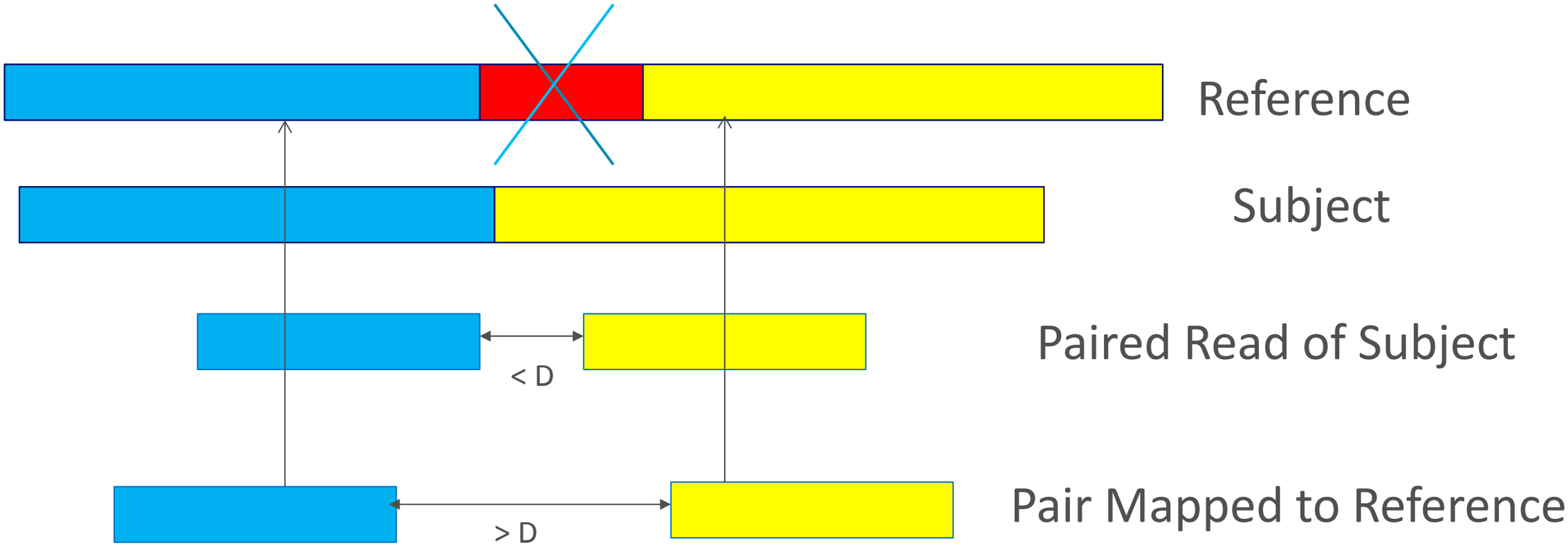
- Model: 2 linear strings 3B long but read as random fragments aligned to a reference
- Big Data: Single DNA:100GB (why?)
- Software steps: 1. Align Reads, 2. Call variants, 3. Correlate variants with disease
- Probabilistic Inference: Randomly sampled fragments + errors in each processing stage

Model of Sequencing Process



With Short Reads, no assembly only alignment

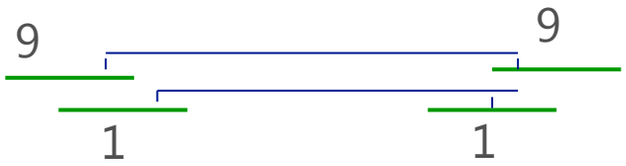
Example: Calling Deletions



Deletion as Interval Processing

Reads

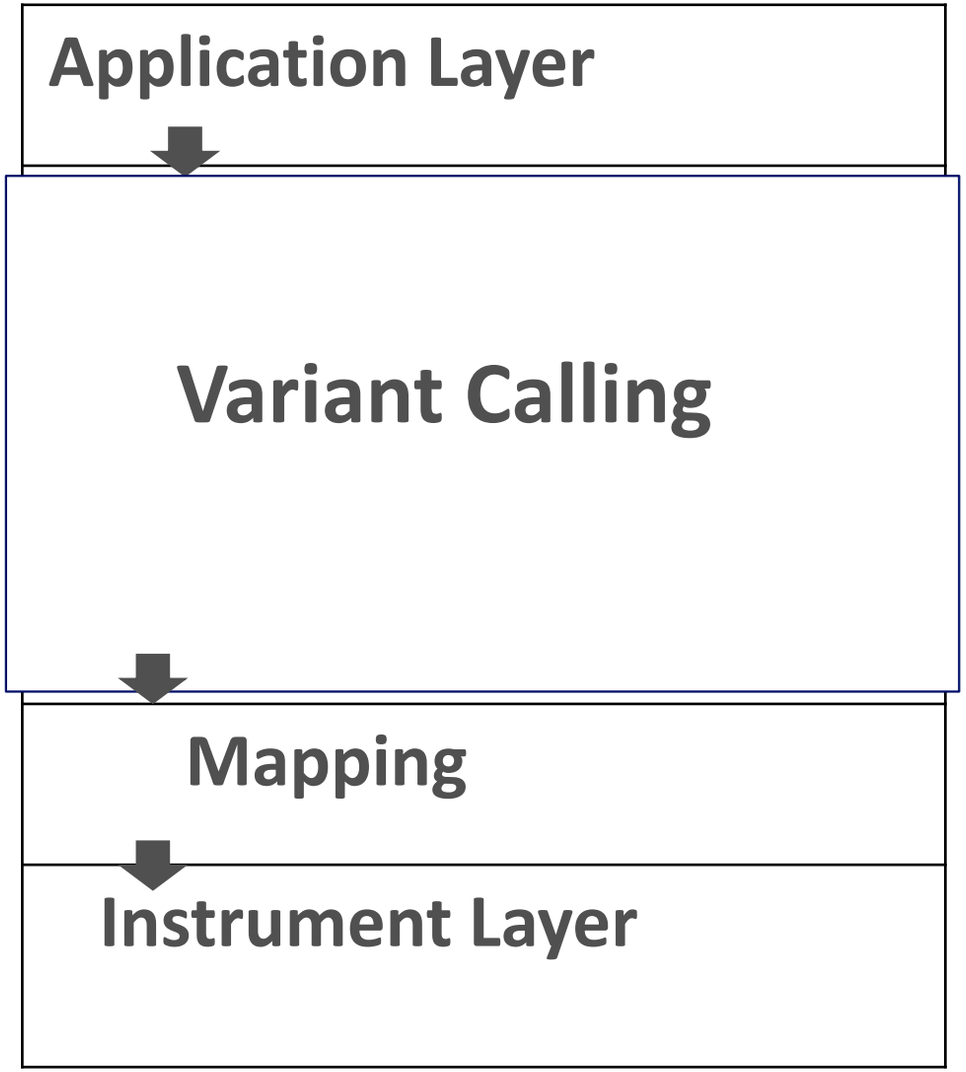
Possibly deleted regions



Evidence:
Two witnesses

Compute
Confidence (low)

Layering today



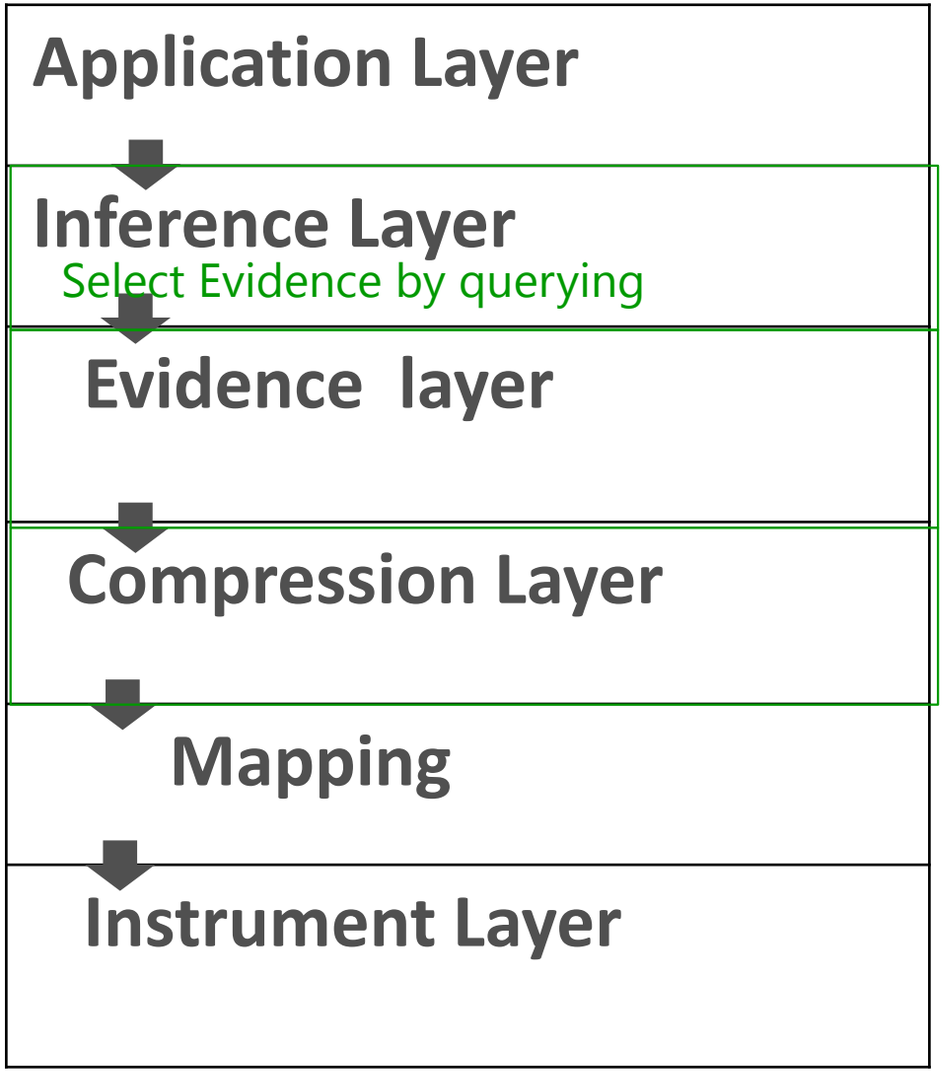
e.g., : cancer genomics, pharmacogenomics

e.g., Breakdancer

e.g., BWA, Berkeley SNAP

e.g.,: Illumina, PacBio

Idea 2: Split Evidence and Inference



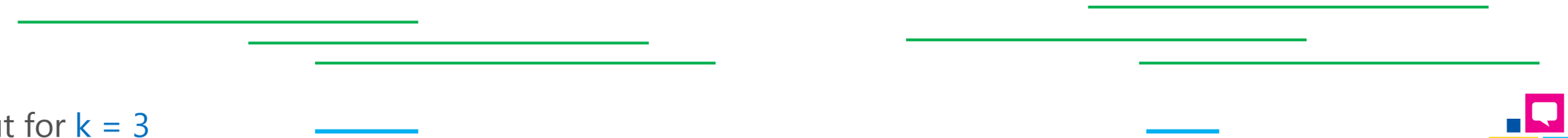
Probabilistic: e.g., Bayesian inference.

Split Variant callers into two layers

Deterministic: storage, retrieval

Idea 3: Interval processing abstractions

- *Intervals* (genes, deletions) are first class
- *Data model*: like SQL, tables with intervals
- GQL (Genome Query Language) *Operators*:
  - Select: A set of rows from each table
  - Join: Two tables based on interval intersection
  - MergeIntervals: Minimal set of disjoint intervals covered by at least k intervals in input.

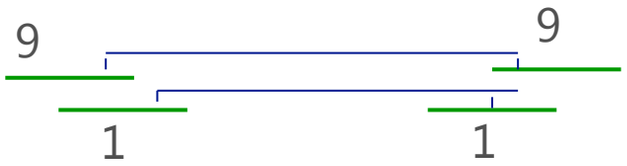


Output for k = 3

Deletion using GQL operators

Reads

Select + Merge
with k = 2



Join back
with Reads

Inference Layer

GQL Deletion Script we ran

```
include<tables.txt>
genome NA18506;

Discordant = select * from READS
where (mate_location - location > 1000)
```

Select pairs with distance > 1000

```
Predicted_deletions =
select merge_intervals( interval_count > 5)
from Discordant
```

Identify regions with 5 such pairs

```
out= select *
from MAPJOIN Predicted_deletions, Discordant
```

Select Reads in these regions

Equivalent in GATK: 150 lines of Java

Deletion Results

- GQL found 113 deleted intervals in Chromosome 1 on a certain genome (NA18506)
- But Conrad et al. (Nature Genetics 2006) found only 8 in the same individual
- Q: How do results compare? Such conflicts are common

Probing further using GQL. . .

- Join with Conrad Intervals to find missing deletions (MD) in Conrad not in GQL Results
- Select Reads with high pair separations in MD. (None)
- Reads within MD should have reduced count (coverage) in MD. (Not found)
- NA18506 is the child of a Yoruban trio. Repeated Query in parent. Deletions in GQL analysis not in Conrad's data were in parent.

GQL allows interactive sifting of results,
See Bioinformatics paper

Other processing examples

- 1. Report change in position X SNP
- 2. Find replicated substrings Copy Number
- 3. Find reversed substrings. Inversions
- 5. Ascribe substrings to Mom/Dad Phasing

While we used deletion as an example, our abstractions apply to these as well

Idea 4: Optimizations

- Materialized views: many whole genome queries require only scanning metadata
- Lazy Joins: only store indices of joined entries, access original columns only at final output.
- Parallelism: each chromosome in a separate Azure VM. Used 24 VMs, $0.96/hour

60 seconds, $1 for hardest query on *single* genome.
Ways to go . . . but interactivity *plausible*

Summary

- Vision: Hypotheses generation in seconds not hours/days: *interactive* genetics.
- Ideas: Evidence-inference separation, GQL interval operators, lazy joins
- Database: mixes existing ideas but crucial to get *whole package* right
- Applications: Cancer Genomics, Newborn genomics, personalized medicine

The Builder of GQL 1.0

Christos Kozanitis, who will be a Postdoc at UCB

More details, experiments etc:
cseweb.ucsd.edu/~vbafna/gqlsystemspaper.pdf