Secure MPC for Federated Genomic Data Analysis

Scott Constable (PhD),
Anshumali Jain (Ms),
Suyash Rathi (Ms),
Yuzhe Tang (AP)

Computation task (1)

- Statistical analysis (for GWAS): Maf, Chi2
  - Goal: Association between a disease and human genetic feature (SNP).
  - Maf: minor allele frequency
    - Genotypes of five individuals: AA, AG, AA, AG, and GG.
    - G is less frequent than A ==> MAF: 0.4
  - Chi2: association test based on frequencies in control/case

$$\chi^2 = \sum_{i,j} \frac{(obs_{i,j} - exp_{i,j})^2}{exp_{i,j}}$$

- Algorithmic model: counting

Computation task (2)

- Secure comparison
  - Hamming distance
  - Approximate edit distance
    - Application optimized
- Algorithmic model:
  - A merge followed by counting differences.

Implementation framework

PCF (from UVA): portable circuit framework

- A C-like language (w. restrictions)
- A compiler: LCCYao
- An interpreter/runtime: BetterYao:
  - Based on garbled circuits/OT
- Note: We tried using GMW protocol which only has low-level circuit interface.

Design: How to express the algorithm in PCF variant of C?

Restrictions and solutions

Limited input-data size

- BetterYao limits input be less than 8000 bits
- Challenging to handle big-data inputs

Solutions

- Partition input data
  - GWAS: independent genotypes, easy partitioning
  - Edit: partition by concatenation of chrome# & pos

Restrictions and solutions

Lack of support for:

- negative number, floating point computation

Solution:

- Simulated by integer computation:
  - “x <<< FPP / y”
  - (FPP is floating point precision)

Performance optimization

Computation level:

- Local computation (5~9X)
- Dynamic input encoding
- Merge: Improving from $O(n^2)$ to linear.

System level:

- Automatic parallelism on multi-core
  - e.g. *xarg* to run multiple processes with bound

Security guarantee

BetterYao enables security protection under various models:

- Semi-honest to malicious

Leaks input size (e.g. # of lines with chrome 1)

System architecture

Implementation:

- By extending PCF platform
- Automatic dynamic code generator
  - Loop length generation (Edit)
  - Data partitioning (GWAS)
- Bash to glue the components

Perf. Results (Networked setting)

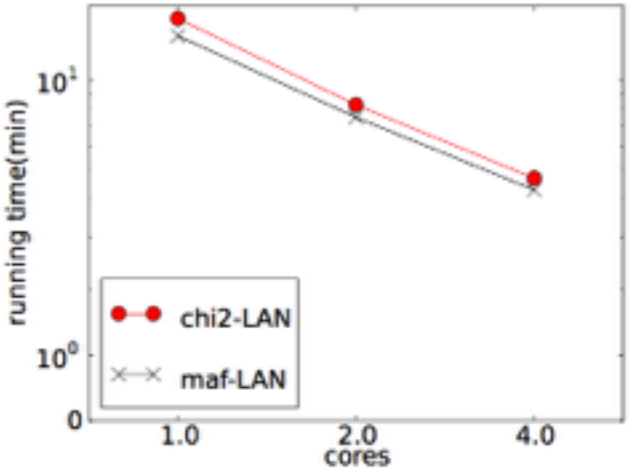
Setups

- Local: on one node: shared memory/caches
- LAN: two homogeneous machines in SU LAN
- Internet: two heterogenous machines respectively in UCSD and IUB

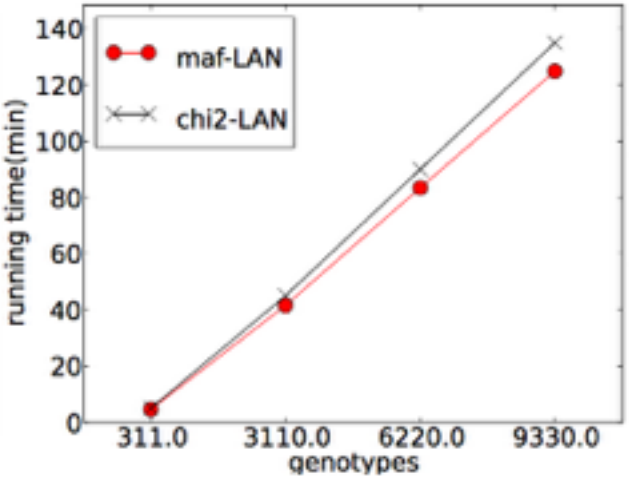
	Local	LAN	Internet
MAF	6m33.200s	4m44.124s	5m55.750s
χ^2	7m13.530s	5m8.721s	6m3.502s

Table 2: Computation running time with various network setups

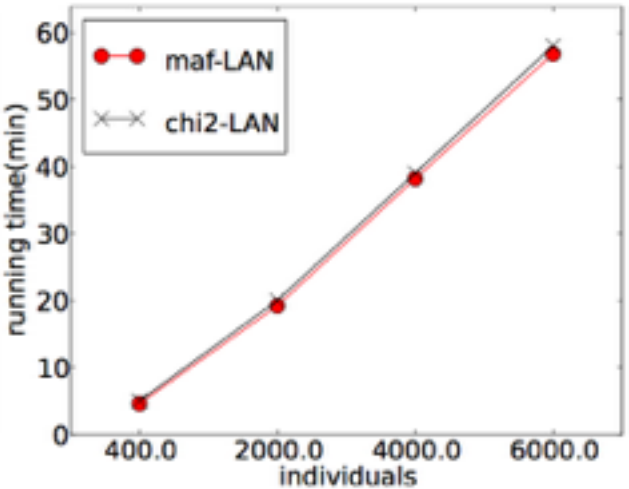
Perf. Results (Data sizes)



(a) Cores



(b) Data size by genotypes



(c) Data size by individuals

Updates to perf. results

On a LAN with 4 core machine:

- MAF: 29.9 seconds (around 5.45 X speed-up)
- Chi2: 56.5 seconds (around 9.33 X speed-up)

Acknowledgement

PCF team: https://github.com/cryptouva/pcf/graphs/contributors

Questions?

Thank you

Contact:
Yuzhe Tang
Assistant Professor
Syracuse University
ytang100@syr.edu
ecs.syr.edu/faculty/yuzhe