Verifiable Stream Computation and Arthur-Merlin Communication

Justin Thaler, Yahoo Labs
Joint Work with:
Amit Chakrabarti, Dartmouth
Graham Cormode, University of Warwick
Andrew McGregor, Umass Amherst
Suresh Venkatasubramanian, University of Utah
Outsourcing

- Many applications require outsourcing computation to untrusted service providers.
  - Main motivation: commercial cloud computing services.
  - Also, weak peripheral devices; fast but faulty co-processors.
  - Volunteer Computing (SETI@home, World Community Grid, etc.)

- User requires a guarantee that the cloud performed the computation correctly.
Goals of Verifiable Computation

- Goal 1: Provide user with a correctness guarantee.
- Goal 2: User must operate within the restrictive data streaming paradigm (models a user who lacks the resources to store the input locally).
Interactive Proofs

Cloud Provider

Business/Agency/Scientist
Interactive Proofs

Cloud Provider

Data

Business/Agency/Scientist
Interactive Proofs

Cloud Provider

Business/Agency/Scientist

Data

Summary
Interactive Proofs

Cloud Provider

Data

Business/Agency/Scientist

Summary

Question

Answer
Interactive Proofs

Cloud Provider

Data

Question

Answer

Challenge

Response

Business/Agency/Scientist

Summary
Interactive Proofs

Cloud Provider

Question

Answer

Challenge

Response

Challenge

Response

Business/Agency/Scientist

Data

Summary
Interactive Proofs

Cloud Provider

Data

Question

Answer

Challenge

Response

Challenge

Response

Business/Agency/Scientist

Accept or Reject
Interactive Proofs

• Prover $P$ and Verifier $V$.

• $P$ solves problem, tells $V$ the answer.
  – Then $P$ and $V$ have a conversation.
  – $P$’s goal: convince $V$ the answer is correct.

• Requirements:
  – 1. Completeness: an honest $P$ can convince $V$ to accept.
  – 2. Soundness: $V$ will catch a lying $P$ with high probability (secure even if $P$ is computationally unbounded).
Streaming Interactive Proofs (SIP) Model [CTY12]

- After both observe stream, P and V have a conversation.

- Fits cloud computing well: streaming pass by V can occur while uploading data to cloud.

- V never needs to store entirety of data.
Costs of SIPs

- Two main costs: amount communication, and V’s working memory. Both must be sublinear (ideally polylogarithmic) in input size.
- Other costs: running time, number of messages.
History of Streaming Interactive Proofs

- [CTY12] introduced streaming interactive proofs (SIPs), gave logarithmic cost protocols for many problems.
- Earlier work [CCM09] had introduced a more restricted model corresponding to one-message SIPs.
- [KP13, GR13, CTY12, CCMTV14, KP14] study variants of these models.
- [CMT12, TRMP13] gave efficient implementations of protocols from [CCM09, CMT10] (and from the literature on “classical” interactive proofs).
Talk Outline

• Part 1: Exponentially more efficient two-message SIPs for many problems.
• Part 2: New communication models that allow us to investigate the **limitations** of constant-round SIPs.
Part I: Exponentially More Efficient Constant-Round SIPs
INDEX Problem

- Data stream specifies a vector $\mathbf{x}$ followed by an index $i$. Goal is to output $x_i$.
- Requires $\Omega(n)$ space in the standard streaming model.
Prior Work on SIPs for INDEX

- [CCM09/CCMT14]: A 1-message protocol with space and comm. costs $O(\sqrt{n})$. Showed this is optimal.
- [CTY12]: A $(2k-1)$-message protocol with cost $O(n^{1/(k+1)})$.
- All of these protocols based on public-coin sum-check techniques [LFKN90].
- [KP13] claimed a matching lower bound for any $k>0$. 
Prior Work on SIPs for INDEX

- [CCM09/CCMT14]: A **1-message** protocol with space and comm. costs $O(\sqrt{n})$. Showed this is optimal.
- [CTY12]: A $(2k-1)$-message protocol with cost $O(n^{1/(k+1)})$.
- All of these protocols based on public-coin **sum-check** techniques [LFKN90].

- We show [KP13] lower bound only applies to “public coin” SIPs.
- We give a 2-message protocol with cost $O(\log n \log \log n)$.
- Later, we’ll build on this protocol to solve more complicated problems (NNS, RangeCount, PatternMatching, Median, etc).
- Protocol adapts result of [Raz05] on **IP/rpoly**. See also [CKLR11].
The 2-message SIP for INDEX
A general technique

- Arithmetization: Given function $g$ defined on small domain, replace $g$ with its **multilinear extension** $\tilde{g}$ as a polynomial defined over a large field.

- Can view $\tilde{g}$ as error-corrected encoding of $g$: If two (boolean) functions differ in one location, their multilinear extensions will differ in almost all locations.

- Error-correcting properties give $V$ considerable power over $P$. 
The INDEX Problem

- Data stream specifies a vector $\mathbf{x}$ followed by an index $i$. Goal is to output $x_i$. 
The INDEX Protocol, Notation

• View \( \mathbf{x} \) as a function mapping \( \{0,1\}^{\log n} \rightarrow \{0,1\} \) via:
  \[
  \mathbf{x}(j_1,\ldots,j_{\log n}) = x_j, \text{ where } (j_1,\ldots,j_{\log n}) \text{ is the binary representation of } j.
  \]

• Fix a finite field \( \mathbf{F} \) of size at least \( 4\log n \).

• \( \widetilde{\mathbf{x}} \) denotes the multilinear extension of \( \mathbf{x} \) over \( \mathbf{F} \).
The INDEX Protocol, Part 1

- $V$ picks a random vector $\mathbf{r} \in \mathbb{F}^{\log n}$, and evaluates $\tilde{\mathbf{x}}(\mathbf{r})$ in streaming pass over $\mathbf{x}$ (requires space $O(\log n \log |\mathbb{F}|)$).
How Can V Evaluate \( \tilde{x}(r) \)?

- For each \( j \in \{0,1\}^{\log n} \), define \( \delta_j : \{0,1\}^{\log n} \rightarrow \{0,1\} \) via:
  \[
  \delta_j(k) := 1 \text{ if } j = k \text{ and } \delta_j(k) := 0 \text{ otherwise.}
  \]

- Note: \( \tilde{x} = \sum_{j \in \{0,1\}^{\log n}} x_j \tilde{\delta}_j \) as formal polynomials, where \( \tilde{\delta}_j \)
  is the multilinear extension of \( \delta_j \).

- So \( \tilde{x}(r) = \sum_{j \in \{0,1\}^{\log n}} x_j \tilde{\delta}_j(r) \).

- i.e., each entry \( j \) of \( x \) contributes \textbf{independently} to \( \tilde{x}(r) \) \((V \text{ can just keep a running sum while observing stream})\).
The INDEX Protocol, Part 2

Boolean Hypercube $\{0, 1\}^{\log n}$

Extended Hypercube $F^{\log n}$

\(\tilde{x}\)

\(x\)

\(\bullet\) denotes entries of \(x\) that equal 1.
The INDEX Protocol, Part 2

\[ \tilde{x} \]

\[ x \]

\[ \{0, 1\}^{\log n} \]

\[ F^{\log n} \]

\bullet denotes entries of \( x \) that equal 1.

Evaluation point \( r \).
The INDEX Protocol, Part 2

- **Boolean Hypercube** $\{0, 1\}^{\log n}$
- **Extended Hypercube** $F^{\log n}$
- • denotes entries of $x$ that equal 1.
- **Evaluation point** $r$.
- **Query point** $i$. 

Diagram:
- The diagram illustrates the relationship between the boolean hypercube and the extended hypercube.
- Black dots represent the entries of $x$ that equal 1.
- Arrows indicate transformations or mappings between the two hypercubes.
The INDEX Protocol, Part 2

- Boolean Hypercube \( \{0, 1\}^{\log n} \)
- Extended Hypercube \( F^{\log n} \)

- Challenge line \( \lambda \)
- Evaluation point \( r \)
- Query point \( i \)

\( \tilde{x} \)

- \( \bullet \) denotes entries of \( x \) that equal 1.
Extensions of the INDEX Protocol
Polylogarithmic Cost Protocols

- We give polylogarithmic cost protocols for the following problems.
  - Nearest Neighbor Search under many standard metrics ($L_1$, $L_2$, $L_\infty$, etc.)
  - Median and Quantiles.
  - RangeCount Queries.
  - PatternMatching (with wildcards).
Overview of RangeCount Protocol

- RangeCount Problem: Fix a data universe \([n]\) and a range space \(R \subseteq 2^{[n]}\). The input is list of points \(\{x_1, \ldots, x_m\}\) from \([n]\), followed by a range \(R^* \in R\). Goal is to output \(|\{i : x_i \in R^*\}|\).
Overview of RangeCount Protocol

- **RangeCount Problem**: Fix a data universe \([n]\) and a range space \(R \subseteq 2^n\). The input is list of points \(\{x_1, \ldots, x_m\}\) from \([n]\), followed by a range \(R^* \in R\). Goal is to output \(|\{i : x_i \in R^*\}|\).

- **Basic idea**: Reduce to the (Generalized) INDEX problem.
  - Create a “derived stream” consisting entirely of ranges.
  - On stream update \(x_i\), insert a copy of every range \(R\) that \(x_i\) is in.
  - \(V\) needs to know the frequency of \(R^*\) in derived stream. Can answer this with the (Generalized) INDEX protocol.
  - Space and communication costs are only \(O(\log |R| \log \log |R|)\).
  - **Problem**: \(V\) requires \(|R|\) time per stream update!
Online Interactive Proofs (Communication Model)
So How Powerful Are $O(1)$-Round SIPs?

- INDEX has a two-message protocol of logarithmic cost.
- Does a similar protocol exist for “harder” problems such as DISJOINTNESS?
So How Powerful Are $O(1)$-Round SIPs?

- INDEX has a two-message protocol of logarithmic cost.
- Does a similar protocol exist for “harder” problems such as DISJOINTNESS?
- To investigate, we introduce two hierarchies of communication models called $OIP_+$ and $OIP$.
- $OIP_+[k]$ can simulate all $k$-message SIPs. So lower bounds against $OIP_+$ protocols imply ones against SIPs.
- $OIP[k]$ is weaker, but can still simulate all known SIPs, and captures the fundamental way SIPs differ from IPs.
AM^{cc} [BFS86]
Goal: Compute $f(x,y)$
Step 1: Random coins are broadcasted.
Step 2: Merlin broadcasts a message to Alice and Bob.
Step 3: Alice and Bob engage in a deterministic communication protocol. Bob outputs a bit.
OIP_+[k]
Step 1: Alice and Bob toss “secret coins” that are hidden from Merlin.
Step 2: Alice sends a single message to Bob.
Step 3: Bob and Merlin engage in $k$-message interaction.
Step 1: Alice and Bob toss “secret coins” that are hidden from Merlin.
Step 2: Bob and Merlin engage in k-message interaction.
Step 3: Alice sends a single message to Bob, who then outputs a bit.
OIP[k] Can Simulate All Known k-message SIPs
OIP[2] protocol of cost $O(\log n \log \log n)$ for INDEX.

Goal: Output $x_i$. 
Step 1: Alice and Bob toss “secret coins” that are hidden from Merlin to choose evaluation point \( r \).
Bob sends Merlin $\lambda$, the line through $r$ and $i$. Merlin responds to univariate polynomial $G(t)$ claimed to equal $\tilde{x}(\lambda(t))$. 
Alice sends Bob $\tilde{x}(r)$. 
A Communication Complexity Zoo

Notation:
- $\text{OIP}^k$ denotes class of functions solved by polylog cost OIP$^k$ protocols, $\text{AM}^{cc}$ functions solved by polylog cost AM$^{cc}$ protocols.
- $\rightarrow$ denotes containment with exponential separation.
- $\leftrightarrow$ denotes equality.
Details and Intuition

Notation:
• Let $\mathbf{R}^{[2,B]}$ denote the class of functions solved by polylog cost protocols in which Bob sends a single message to Alice, and Alice send a single response to Bob (and there is no Merlin).
  • $\mathbf{R}^{[2,B]}$ is the simplest non-online model of communication.
Details and Intuition

Notation:
• Let $R^{[2,B]}$ denote the class of functions solved by polylog cost protocols in which Bob sends a single message to Alice, and Alice send a single response to Bob (and there is no Merlin).
  • $R^{[2,B]}$ is the simplest non-online model of communication.
• Theorem: $OIP^{[2]} = R^{[2,B]}$.

• If we let Bob send two messages to Merlin, we can pretend that both Bob and Merlin can talk to Alice.
  • Which is enough power to simulate all of $AM^{cc}$. 

Details and Intuition
Details and Intuition

Notation:
• Let $R^{2,B}$ denote the class of functions solved by polylog cost protocols in which Bob sends a single message to Alice, and Alice send a single response to Bob (and there is no Merlin).
  • $R^{2,B}$ is the simplest non-online model of communication.
• Theorem: $OIP^{2} = R^{2,B}$.
• So $OIP^{2}$ lets us “pretend” that Bob can send one message to Alice, even though the $OIP^{2}$ model lets neither Bob nor Merlin talk to Alice.
Details and Intuition

Notation:
• Let $\mathbf{R}^{[2,B]}$ denote the class of functions solved by polylog cost protocols in which Bob sends a single message to Alice, and Alice send a single response to Bob (and there is no Merlin).
  • $\mathbf{R}^{[2,B]}$ is the simplest non-online model of communication.
• Theorem: $\mathbf{OIP}^{[2]} = \mathbf{R}^{[2,B]}$.
• So $\mathbf{OIP}^{[2]}$ lets us “pretend” that Bob can send one message to Alice, even though the $\mathbf{OIP}^{[2]}$ model lets neither Bob nor Merlin talk to Alice.
• $\mathbf{OIP}^{[4]}$ lets us pretend that both Bob and Merlin can talk to Alice.
  • This is enough power to simulate all of $\mathbf{AM}^{cc}$. 
Main Findings

- Any OIP[2] or OIP[3] protocol for DISJOINTNESS has cost $\Omega(n^{1/2})$ and $\Omega(n^{1/3})$ respectively. Both bounds are tight.
  - i.e. There is no three-message SIP of polylog cost for DISJOINTNESS using “known techniques”.

- OIP[4] is equivalent to AM$^{cc}$, a communication class beyond the reach of current lower bound methods.
  - i.e. Proving lower bounds on 4-message SIPs may be challenging.

- Generic round-reduction impossible in the OIP hierarchy.
  - In contrast, AM[O(1)] = AM[2] in classical interactive proofs.
Thank you!