Spartan Parallel

Kunming Jiang

Sep 21, 2023

1 Introduction

The problem is consisted of P R1CS instances of the form $\{A_i, B_i, C_i\}$, where each A_i, B_i, C_i are $X \times Y$ matrices, X represents the number of constraints and Y represents the number of (inputs + witnesses). For each instance $\{A_i, B_i, C_i\}$, there are Q_i (input, witnesses) vectors of length Y that claim to be a correct execution of the instance. We name these (input, witnesses) vectors $z_{i,0}, \dots z_{i,Q_i}$. The goal is to prove the claim for all such $z_{i,j}$ using SNARK.

For simplicity, assume all P, Q_i, X, Y are powers of two. Define $Q_{\texttt{max}} \leftarrow \max_i Q_i$ and $Q_{\texttt{sum}} \leftarrow \sum_i Q_i$. Let $p = \log P$, $q_i = \log Q_i$, $q_{\texttt{max}} = \log Q_{\texttt{max}}$, $x = \log X$, and $y = \log Y$. We want the total runtime of the Prover ($\mathcal{P}$) to be $O(Q_{\texttt{sum}} \cdot X \cdot \log(Q_{\texttt{sum}} \cdot X))$, and that of the Verifier ($\mathcal{V}$) to be $O(\log(P \cdot Q_{\texttt{max}} \cdot Y))$.

2 Starting from Spartan

We begin by modifying Spartan to support data-parallelism. In section 3 we perform a complexity analysis and in 4 we resolve the complexity blow-up.

The modified protocol is shown below:

1. $\mathcal{P}$ commits witnesses and sends the commit $\mathcal{C}$ to $\mathcal{V}$.
2. $\mathcal{V}$ samples $\tau \in_R \mathbb{F}^{p+q_{\texttt{max}}+x}$ and sends τ to $\mathcal{P}$.
3. $\mathcal{P}$ computes $Az, Bz, Cz, \widetilde{\text{eq}}_\tau : \mathbb{F}^{p+q_{\texttt{max}}+x} \to \mathbb{F}$, defined as:
$$Az(t_p, t_q, t_x) = A(t_p, t_x) \cdot z(t_p, t_q, t_x)$$
$$\widetilde{\text{eq}}_\tau(t) = 1 \text{ if } t = \tau,\ 0 \text{ otherwise}$$
4. $\mathcal{V}$ and $\mathcal{P}$ perform sum-check #1 to prove
$$\sum_{t \in \mathbb{F}^{p+q_{\texttt{max}}+x}} (Az(t) \cdot Bz(t) - Cz(t)) \cdot \widetilde{\text{eq}}_\tau(t) = 0$$
5. By the end of the sum-check, $\mathcal{V}$ samples $r = (r_p, r_q, r_x) \in_R \mathbb{F}^{p+q_{\texttt{max}}+x}$ and $\mathcal{P}$ obtains $v_A = Az(r)$, $v_B = Bz(r)$, $v_C = Cz(r)$. $\mathcal{P}$ sends v_A, v_B, v_C to $\mathcal{V}$.
6. $\mathcal{V}$ checks $(v_A \cdot v_B - v_C) \cdot \widetilde{\text{eq}}_\tau(r) = e_1$, where e_1 is the final claim of the sum-check.

7. $\mathcal{V}$ samples $r_A, r_B, r_C \in_R \mathbb{F}$ and sends to $\mathcal{P}$. $\mathcal{P}$ computes $T = r_A \cdot v_A + r_B \cdot v_B + r_c \cdot v_C$.

8. $\mathcal{P}$ computes
$$ABC : \mathbb{F}^{p+y} \to \mathbb{F} = r_A \cdot A(\cdot) + r_B \cdot B(\cdot) + r_C \cdot C(\cdot)$$
$$Z_{r_q} : \mathbb{F}^{p+y} \to \mathbb{F} \text{ where } Z_{r_q}(t_p, t_x) = z(t_p, r_q, t_x)$$
$$\widetilde{\text{eq}}_{r_p}(t_p) = 1 \text{ if } t_p = r_p, 0 \text{ otherwise}$$

9. $\mathcal{V}$ and $\mathcal{P}$ perform sum-check #2 to prove
$$\sum_{t \in \mathbb{F}^{p+y}} ABC(t) \cdot Z_{r_q}(t) \cdot \widetilde{\text{eq}}_{r_p}(t_p) = T$$
By the end of the sum-check, $\mathcal{V}$ samples $r^* = (r_p^*, r_x) \in_R \mathbb{F}^{p+y}$ and obtains the claim e_2.

10. $\mathcal{P}$ opens the commitment $\mathcal{C}$ and evaluates the witness polynomial on $r_p^*, r_q, r_y[1..]$ as w, sends w to $\mathcal{V}$.

11. $\mathcal{V}$ evaluates the input polynomial on $r_p^*, r_q, r_y[1..]$ as v, computes $v_Z = (1 - r_y[0]) \cdot w + r_y[0] \cdot v$. $\mathcal{V}$ also evaluates $v_{r_p} = \widetilde{\text{eq}}_{r_p}(r_p^*)$

12. $\mathcal{V}$ checks $e_2 = (r_A \cdot v_A + r_B \cdot v_B + r_c \cdot v_C) \cdot v_Z \cdot v_{r_p}$.

3 Identifying Runtime Overhead

3.1 Verifier Cost

Since we only require the Verifier cost to be $O(\log(P \cdot Q_{\texttt{max}} \cdot Y))$, there isn't any improvement we need to do. Here's the cost breakdown:

- In step 2: $O(\log(P \cdot Q_{\texttt{max}} \cdot Y))$ for sampling τ.
- In step 4: $O(\log(P \cdot Q_{\texttt{max}} \cdot Y))$ due to the sum-check having $p + q_{\texttt{max}} + x$ rounds. Assume X and Y are of similar size.
- In step 6: $O(\log(P \cdot Q_{\texttt{max}} \cdot Y))$ for evaluating $\widetilde{\text{eq}}_\tau(r)$.
- In step 9: $O(\log(P \cdot Y))$ due to the sum-check having $p + y$ rounds.
- In step 11: $O(\log(P \cdot Q_{\texttt{max}} \cdot Y))$ for evaluating the input, and $O(\log(P))$ for evaluating $\widetilde{\text{eq}}_{r_p}(r_p^*)$.

3.2 Prover Cost

The prover's cost is currently $O(P \cdot Q_{\texttt{max}} \cdot Y \cdot \log(P \cdot Q_{\texttt{max}} \cdot Y))$, and we need to reduce it to $O(Q_{\texttt{sum}} \cdot X \cdot \log(Q_{\texttt{sum}} \cdot X))$. Here's all the steps that exceeds that complexity:

- Step 1: witness commit
- Step 3: producing Az, Bz, Cz, and $\widetilde{\text{eq}}_\tau$
- Step 4: sum-check 1
- Step 8: compute Z_{r_q}

We now discuss how to improve Prover's cost in these steps.

4 Reducing Complexity

We can first deal with the easier ones:

- We assume that filling an array / matrix with 0s does not take time. Thus

  Idea: express Z as a polynomial over p times a polynomial over q voer a polynomial over x