CSCI 2570 Introduction to Nanocomputing

Coded Computation III

John E Savage

Lecture Topic

- This talk is based on Dan Spielman's paper **Highly Fault-Tolerant Parallel Computation** *Procs 37th Annl IEEE Conf. Foundations of Computer Science*, pp. 154-163, 1996.

- **Spielman's goal:** To realize circuits with unreliable gates more efficiently than the "von Neumann" method.

- **The approach:** To replace the repetition code with a more efficient one.

Computing with Encoded Data

- Recall $\sigma_{i,j}^* = \phi(\sigma_{i,j-1}, \sigma_{i+d,j-1}, w_{i,j})$ on j^{th} step where $\phi: S^3 \rightarrow S$ is next-state function of a processor.
- The codewords Σ_j, Σ_j^d and W_j contain current state of a node, its neighbor and its instruction. We can apply ϕ to components in S, not those in F.
- To handle values in F not S, extend ϕ to the interpolation polynomial $\Phi(r,s,t)$, where r,s,t in F such that for i,j,k in H, $\Phi(i,j,k) = \phi(\sigma^i,\sigma^j,\sigma^k)$ where $\sigma^i,\sigma^j,\sigma^k$ are elements of S.

Computing with Encoded Data

- To handle values in *F* not *S*, extend ϕ to the interpolation polynomial $\Phi(r,s,t)$, where *r*,*s*,*t* in *F* such that for h_i,h_j,h_k in *H*, $\Phi(h_i,h_j,h_k) = \phi(\sigma^i,\sigma^j,\sigma^k)$ where $\sigma^i,\sigma^j,\sigma^k$ are corresponding elements of *S*.

- Form

$$\Phi(r,s,t) = \sum_{i,j,k} \phi(\sigma^i,\sigma^j,\sigma^k) \frac{\prod_{u\neq i}(r-h_t)}{\prod_{u\neq i}(h_i-h_t)} \frac{\prod_{u\neq j}(s-h_t)}{\prod_{u\neq j}(h_i-h_t)} \frac{\prod_{u\neq i}(r-h_t)}{\prod_{u\neq i}(h_i-h_t)}$$

Encoded Hypercube Computation

- Σ_j and W_j are RS codewords. Is Σ_j^d also RS? Is Σ_j^d the set of values of a polynomial over F?
- Index elements of the original hypercube on $N = 2^n$ nodes by $H = GF(2^n)$. Let $F = GF(2^m)$.
- Index of neighbor in direction d is obtained by adding β, an n-tuple with a single 1, $\beta \in H \subseteq F$.
- Adding β to elements in F permutes codeword components. RS interpolation polynomial for Σ_j is mapped to another interpolation polynomial. Thus, Σ_j^d is another RS code with polynomial of same degree.

Putting It All Together

- When no errors at each step, RS codewords $\Sigma_j = \{m_j(i) \text{ for } i \in F\}$, $\Sigma_j^d = \{m_j(i+\beta) \text{ for } i \in F\}$, and $W_j = \{n_j(i) \text{ for } i \in F\}$ are created.

- Compute by extending $\sigma_{i,j}^* = \phi(\sigma_{i,j-1}, \sigma_{i+d,j-1}, w_{i,j})$ to $\Phi(m_j(x), m_j(x+\beta), n_j(x))$ for $x \in F$ and applying it.

- Let c = degree(Φ). Then, $\Phi(m_j(x), m_j(x+\beta), n_j(x))$ has degree c(N-1) and its values over F form an RS codeword of higher degree.

Error Models

- Errors occur independently on gates during the computation of Φ or during degree reduction.
- Conditions needed on coded computation:
  - Encode step inputs and outputs with same code.
  - Design step operations so that a fraction $\leq \theta$ of outputs are in error for each step, $\theta = O(\varepsilon)$, with probability *p*.

Degree Reduction

- Decode RS codeword resulting from computation.
- Re-encode new states using the original RS code.
- Do the resulting operations satisfy all the requirements?
- First condition holds by design.
- Second condition holds if
  - Errors not compounded (von Neumann); let error rate= θ
  - The RS code based on Φ can correct enough errors.
  - If $\theta \leq (|F| - c(|H|-1))/2$, each step decodes correctly.
  - Probability p depends on code length $|F|$ and θ.

Extension to Two Dimensions

- Spielman replaces 1D RS code with a 2D RS code for two reasons:
  - To keep the size of the decoder small, and
  - To ensure that errors experienced by a decoder are statistically independent.
    - Use separate decoder for each row/column RS code
    - Decoding error in one dimension causes many errors in decoder output but only one error in the other dimension.

Two Dimensional RS Code

- 2D RS obtained from 2D interpolation polynomials $m(x,y)$, where (x,y) in H^2. (Replace H by H^2.)

- A degree reduction is done in two steps:
  - Degree reduce on rows; reduce on columns
  - Must show correctness.
  - Can't correct as many errors but decoder smaller.

Deterministic RS Decoding Algorithm

Theorem The encoding and decoding functions $E_{H,F} : F^H \rightarrow F^F$ and $D_{H,F} : F^F \rightarrow F^H \cup \{?\}$ for RS codes can be computed by circuits of size $|F| \log^{O(1)}|F|$. Corrects $k \leq (|F| - |H|)/2$ errors.

Proof Due to Justesen [76] and Sarwate [77].

Kaltofen-Pan Probabilistic RS Decoding Algorithm

Theorem The decoding function $D^k_{H,F}$ can be computed by a **randomized** parallel algorithm that takes $\log^{O(1)} |F|$ time on $(k^2 + |F|) \log^{O(1)} |F|$ processors to correct $k \leq (|F| - |H|)/2$ errors. The algorithm succeeds with prob. 1-1/|*F*|.

- Spielman uses this algorithm with $k = \sqrt{|F|}$ to keep number of processors reasonable.

Decoding of Noisy Computation

Lemma If a) each column in 2D RS code has at most fraction β errors, b) fraction ϵ of degree reductions fail at each stage, and c) bivariate ϕ has degree *c*, a k-error correcting decoder will produce a result that has at most fraction ϵ of outputs in error if $k > \max(2\beta,\epsilon)|F|$ and $c|H| < (1-\epsilon)|F|$.

Proof ϕ combines two words with fraction β errors to produce one with fraction 2β errors. Correct by columns, leaving only errors by decoding circuits. Correct by rows, leaving only errors by decoding circuits. Need $c|H| < (1-\epsilon)|F|$ to ensure that code is RS. (It must be result of interpolating data.)

Putting It All Together

- Use either Kaltofen-Pan decoder (KP) that corrects $k = \sqrt{|F|}$ errors or Justesen-Sarwate algorithm (JS) correcting $k \leq (|F| - |H|)/2$ errors.
- KP: $\log^{O(1)} w$ steps & correct $|F|^{1/2} = w^{1/4}$ errors
- JS: Levelize circuit where *w* is circuit width.
- Both do $|F| \log^{O(1)}|F|$ operations per time step.

- Send *k* sets decoded outputs to majority gates
- Failure if ≥ ½ majority gate inputs are wrong.