Highly Fault-Tolerant Parallel Computation

John Z. Sun
Massachusetts Institute of Technology

October 12, 2011
Outline

• Preliminaries

• Primer on Polynomial Coding

• Coding Strategy
Recap

- **von Neumann (1952)**
  - Introduced study of reliable computation with faulty gates
  - Used computation replication and majority rule to ensure reliability
  - **Main statement**: If any gate can fail with probability $\epsilon$, then the output gate will fail with constant probability $\delta$ by constructing bundles of $r = f(\delta, \epsilon)$ wires. The “blowup” of such a system is $O(r)$.
  - **Alternative statement**: An error-free circuit of $m$ gates can be reliably simulated with a circuit composed of $O(m \log m)$ unreliable components

- **Dobrushin and Ortyukov (1977b)**
  - Rigorously expanded von Neumann’s architecture using *exactly* $\epsilon$ wire probability of error

- **Pippenger (1985)**
  - Gave an explicit construction to the above analysis
  - **Main statement**: There is a constant $\epsilon$ such that, for all circuits $C$, there is a way to replace each wire in $C$ with a bundle of $O(r)$ and an amplifier of size $O(r)$ so that the probability that any bundle in the circuit fails to represent its intended value is at most $w2^{-r}$. The blowup of such a simulation is $O(r)$.

Can we do better?
Computation via Local Codes

- Elias (1958)
  - Focused on multiple instances on pairs of inputs on a particular Boolean function
  - Showed fundamental differences between xor and inclusive-or
  - For the latter, showed that repetition coding is best
- Winograd (1962) and others
  - Further development of negative results along the lines of Elias (see Pippenger 1990 for a summary)
- Taylor (1968)
  - Used LDPC codes for reliable storage in unreliable memory cells
  - Can be extended to other linear functionals
Main Result

- Spielman moves beyond local coding to get improved performance
- **Setup**: Consider a parallel computation machine $M$ with $w$ processors running $t$ time units
- **Result**: $M$ can be simulated using a faulty machine $M'$ with $w \log^{O(1)} w$ processors and $t \log^{O(1)} w$ time steps such that probability of error is $< t2^{-w^{1/4}}$

**Novelty:**
- Using processors (finite state machines) rather than logic
- Running parallel computations to allow for coding
- Using heterogenous components
Notation

Definition
For a set $S$ and integer $d$, let $S^d$ denote the set of $d$-tuples of elements of $S$.

Definition
For sets $S$ and $T$, let $S^T$ denote the set of $|T|$-tuples of elements of $S$ indexed by elements of $T$.

Definition
A pair of functions $(E, D)$ is an encoding-decoding pair if there exists a function $l$ such that

$E : \{0, 1\}^n \rightarrow \{0, 1\}^{l(n)}$

$D : \{0, 1\}^{l(n)} \rightarrow \{0, 1\}^n \cup \{?\},$

satisfying $D(E(\bar{a})) = \bar{a}$ for all $\bar{a}$ in $\{0, 1\}^n$. 
Definition

Let \((E, D)\) be an encoding-decoding pair. A parallel machine \(M'\) \((\epsilon, \delta, E, D)\)-simulates a machine \(M\) if

\[
\operatorname{Prob}\{D(M'(E(\vec{a}))) = M(\vec{a})\} > 1 - \delta,
\]

for all inputs \(\vec{a}\) if each processor produces the wrong output with probability less than \(\epsilon\) at each time step.

Definition

Let \((E, D)\) be an encoding-decoding pair. A circuit \(C'\) \((\epsilon, \delta, E, D)\)-simulates a circuit \(C\) if

\[
\operatorname{Prob}\{D(C'(E(\vec{a}))) = C(\vec{a})\} > 1 - \delta,
\]

for all inputs \(\vec{a}\) if each wire produces the wrong output with probability less than \(\epsilon\) at each time step.
Remarks

- The **blow-up** of the simulation is the ratio of gates in $C'$ and $C$
- The notion of failure here is at most $\epsilon$ on wires  [Pippenger (1989)]
- Restrict $(E, D)$ to be simple to eliminate them from doing computation rather than $M'$
- In this case, the encoder-decoder pair is same for *all* simulations
  - No recoding necessary between levels of circuits
Reed-Solomon Codes

Fields

- A field $\mathcal{F}$ is a countable set with the following properties
  - $\mathcal{F}$ forms an abelian group under the addition operator
  - $\mathcal{F} - \{0\}$ forms an abelian group under multiplication operator
  - Operators satisfy distributive law
- A Galois field has $q^n$ elements for $q$ prime
- $\mathcal{GF}(q^n)$ isomorphic to polynomials of degree $n - 1$ over $\mathcal{GF}(q)$

Reed-Solomon code

- Consider a message $(f_0, \ldots, f_k)$
- For $n = q$, evaluate $f(z) = f_0 + f_1 z + \ldots + f_{k-1} z^{k-1}$ for each $z \in \mathcal{GF}(q)$
- Codeword associated with message is $(f(1), f(\alpha), \ldots f(\alpha^{q-2}))$
- Minimum distance is $d = n - k + 1$
Extended Reed-Solomon Codes

Definition
Let \( \mathcal{F} \) be a field and let \( \mathcal{H} \subset \mathcal{F} \). We define an encoding function of an extended RS code \( C_{\mathcal{H},\mathcal{F}} \) to be

\[
E_{\mathcal{H},\mathcal{F}} : \mathcal{F}^{\mathcal{H}} \rightarrow \mathcal{F}^{\mathcal{F}},
\]

where the message is mapped to the unique degree-\((|\mathcal{H} − 1)\) polynomial that interpolates it.

The decoding function is

\[
D_{\mathcal{H},\mathcal{F}} : \mathcal{F}^{\mathcal{F}} \rightarrow \mathcal{F}^{\mathcal{H}} \cup \{0\},
\]

where the input is mapped to a codeword of \( C_{\mathcal{H},\mathcal{F}} \) that differ in at most \( k \) places and the output is the inverse mapping to the message space.

The error-correcting function is

\[
D_{k,\mathcal{H},\mathcal{F}} : \mathcal{F}^{\mathcal{F}} \rightarrow \mathcal{F}^{\mathcal{F}} \cup \{0\},
\]

where the input is mapped to a codeword of \( C_{\mathcal{H},\mathcal{F}} \) that differ in at most \( k \) places.
Theorem

The encoding and decoding functions $E_{\cal H, \cal F}$ and $D_{\cal H, \cal F}$ can be computed by circuits of size $|\cal F| \log^O(1) |\cal F|$.

Proof: See Justesen (1976) and Sarwate (1977)

Lemma

The function $D^k_{\cal H, \cal F}$ can be computed by a randomized parallel algorithm that takes time $\log^O(1) |\cal F|$ on $(k^2 |\cal F|) \log^O(1) |\cal F|$, for $k < (|\cal F| - |\cal H|)/2$. The algorithm succeeds with probability $1 - 1/|\cal F|$.

Proof: See Kaltofen and Pan (1994). Requires $k = O(\sqrt{|\cal F|})$. 
Definition

Let $\mathcal{F}$ be a field and let $\mathcal{H} \subset \mathcal{F}$. We define an encoding function of a generalized RS code $C_{\mathcal{H}^2, \mathcal{F}}$ to be

$$E_{\mathcal{H}^2, \mathcal{F}} : \mathcal{F}^{\mathcal{H}^2} \rightarrow \mathcal{F}^{\mathcal{F}^2}.$$

The decoding function is

$$D_{\mathcal{H}^2, \mathcal{F}} : \mathcal{F}^{\mathcal{F}^2} \rightarrow \mathcal{F}^{\mathcal{H}^2} \cup \{0\}.$$

**Encoding**: Run RS encoder on first dimension, then on second.

**Decoding**: Run RS decoder on second dimension, then on first

Can correct up to $((\mathcal{F} - \mathcal{H})/2)^2$ errors, but only $(\mathcal{F} - \mathcal{H})/2$ in each dimension.
Computation on Hypercubes

Network model

- Consider an $n$-dimensional hypercube with processors at each vertex (labeled by a string in $\{0, 1\}^n$)
- Processors are connected via edges in hypercube (strings that differ in only one bit)
- Processors are synchronized and are allowed to communicate with one neighbor during each time step
- At each time step, all communication must happen in the same direction

Proposition

Any parallel machine with $w$ processors can be simulation with polylogarithmic slowdown by a hypercube with $O(w)$ processors.

Processor Model

- Processors are identical finite automata with a valid set of states $S = GF(2^s)$ for some constant $s$
- Processors change state based on a deterministic instruction, its previous state, and state of a neighbor
- Communication direction is deterministic and known to each processor
• FSM previous state $\sigma_{i,t}$, neighbor state $\sigma'_{i,t}$ and instruction $w_{i,t}$ are mapped to set $S \subset \mathcal{F}$

• Encode states and instructions using generalized RS codes denoted $a_{\vec{x}}^{t-1}$, $a_{\vec{x} + \vec{v}_i}^{t-1}$ and $W^t_{\vec{x}}$ respectively

• Compute on encoded data and run error-correction function after noise is applied
Communication

- Let $\mathcal{H}$ be spanned by basis elements $v_1, \ldots, v_{n/2}$
- The processors of an $n$-dimensional hypercube are elements of $\mathcal{H}^2$
- Communication into a node $\vec{x}$ by a neighbor can be represented with $\vec{x} + \vec{v}_i$, where $\vec{v}_i \in \mathcal{H}^2$

Computation

- Consider two operation polynomials $\phi_1(\cdot, \cdot)$ and $\phi_2(\cdot, \cdot)$
- The new state can be calculated as
  $$\phi_2 \left( \phi_1 \left( a_{\vec{x}}^{i-1}, a_{\vec{x}+\vec{v}_i}^{i-1} \right), \hat{W}_i \vec{x} \right)$$
- Communication into a node $\vec{x}$ by a neighbor can be represented with $\vec{x} + \vec{v}_i$, where $\vec{v}_i \in \mathcal{H}^2$
- Run degree reduction - run error-correction code to fix up to errors in output state (skipping details)
Main Theorem

Theorem

There exists some constant $\epsilon > 0$ and a deterministic construction that provides, for every parallel program $M$ with $w$ processors that runs for time $t$, a randomized parallel program $M'$ that $(\epsilon, h2^{-w^{1/4}}, E, D)$-simulates $M$ and runs for time $t \log^{O(1)} w$ on $w \log^{O(1)} w$ processors, where $E$ encodes the $(\log^2 w)$-fold repetition of a generalized Reed-Solomon code of length $w \log^{O(1)} w$ and $D$ can correct any $w^{-3/4}$ fraction of errors in this code.

Proof

- Can simulate $M$ with a $n$-dimensional hypercube with polylogarithmic slowdown if $2^n > w$
- Choose $\mathcal{F}$ to be smallest field $\mathcal{GF}(2^\nu)$ such that $S \subset \mathcal{GF}(2^\nu)$
- Using degree reduction and error-correction function, an arithmetic program can be constructed that computes the same function as $M$ that runs for time $t \log^{O(1)} w$ on $w \log^{O(1)} w$ processors
- This code can tolerate failures in up to $w^{1/4}/\log^{O(1)} w$ processors
- Using repetition, it can be shown that probability of simulation failing is at most $t2^{-w^{1/4}}$
Remarks

- Can prove better results if the number of levels in the circuit is not restricted, allowing for a better error-correcting function
- There is discussion on applications to self-correcting programs
- Directions for future work
  - Greater fault tolerance
  - Constant blow-up, like for Taylor (1968)
  - Construction via other codes