Reliability in Distributed Computing and HPC: United We Stand?

Jared Saia
University of New Mexico
Outline

• 3 problems
  • Byzantine Agreement
  • Networks of Noisy Gates
  • Secure Multiparty Computation
• Problem of Mutual Interest?
HPC vs DC

- HPC: Adding nodes makes the problem easier
- DC: Adding nodes makes the problem harder
HPC vs DC

- HPC: Adding nodes makes the problem easier
- DC: Adding nodes makes the problem harder

Example: Byzantine Agreement
Byzantine Agreement

• Many nodes, some are faulty
• Periodically, nodes unite in a decision
• How? Who counts the votes?
Naive: Majority Filtering

Input

Output

0 0

0 0

0 0

0 0

1 0

1 0
Naive: Majority Filtering

Input

Output

Wednesday, August 8, 12
Byzantine Agreement

• Each proc starts with a bit

• Goal: 1) all good procs output the same bit; 2) this bit equals an input bit of a good proc
Byzantine Agreement

- Each proc starts with a bit
- Goal: 1) all good procs output the same bit; 2) this bit equals an input bit of a good proc
- $t$ bad procs controlled by an omniscient adversary
Problem

Input

Output

0 0
0 0
0 0
1 1
1 1
1 1
Majority Filtering + BA

Input

Output

Byzantine Agreement

Wednesday, August 8, 12
All good procs always output same bit
If majority bit held by > 2 good procs, then all procs output majority bit.
Impossibility Result

- 1982: FLP show that 1 fault makes deterministic BA impossible in asynch model

- 2007: Nancy Lynch wins Knuth Prize for this result, called “fundamental in all of Computer Science”
Solution: Randomization

• A randomized algorithm can solve BA [Ben-Or ’83]

• Ben-or’s algorithm solves with probability 1, but requires exponential time in expectation

• Many subsequent improvements
Applications

- Databases, State Machine Replication, Secure Multiparty Computation, Control systems, Sensor Networks, Cloud Computing, etc.
Applications

• Databases, State Machine Replication, Secure Multiparty Computation, Control systems, Sensor Networks, Cloud Computing, etc.

• Peer-to-peer networks
  “These replicas cooperate with one another in a Byzantine agreement protocol to choose the final commit order for updates.” [KBCCEGGRWWWZ '00]

• Game Theory (Mediators)
  “deep connections between implementing mediators and various agreement problems, such as Byzantine agreement” [ADH '08]

• Rule Enforcement
  “… requiring the manager set to perform a Byzantine agreement protocol” [NWD '03]
Recent Improvements

- Decades of work improved runtime to constant expected time [CKS ’05]
- But message cost remained high: $O(n^2)$
- Recent results: each proc. sends $\tilde{O}(\sqrt{n})$ bits [KS ’11]
Recent Improvements

- Decades of work improved runtime to constant expected time [CKS ’05]
- But message cost remained high: $O(n^2)$
- Recent results: each proc. sends $\tilde{O}(\sqrt{n})$ bits [KS ’11]
- Can achieve $O(\log n)$ bits with a random beacon
BA Scalability

Both RBQuery and RBSampler assume a random beacon [MS ’12]
BA for HPC?

- Problem: Factor of 4 blowup in resource cost just to tolerate one “soft” (Byzantine) fault
Networks of Noisy Gates

• Given a function, $f$, that can be computed with $n$ gates

• Must build a network to compute $f$ with unreliable gates

• Gates are unreliable: with probability $\epsilon$ they fault; when they fault, output is incorrect
Networks of Noisy Gates

• Given a function, f, that can be computed with n gates

• Must build a network to compute f with unreliable gates

• Gates are unreliable: with probability $\epsilon$ they fault; when they fault, output is incorrect

• Q: How many unreliable gates do we need to compute f with probability $1-o(1)$
Networks of Noisy Gates

Q: How many unreliable gates do we need to compute f with probability $1-o(1)$

- $O(n \log n)$ gates suffice [Von Neumann ’56]
- $\Omega(n \log n)$ gates necessary [PST ’91]
Upper bound

Executive Organ

Sampler

Restoring Organ
Upper bound

Executive Organ

Sampler

Restoring Organ

x bad outputs on top $\rightarrow$ x/4 “bad” majorities on bottom

Wednesday, August 8, 2012
Upper bound

\[
\text{x bad outputs on top } \rightarrow \text{x/4 “bad” majorities on bottom (for suff. small x)}
\]
\[ \leq \theta \text{ fraction} \]

\[ \leq 2\theta + 2\epsilon \text{ fraction} \]
\[ \leq \theta \text{ fraction} \]

whp by Chernoff bounds

\[ \leq 2\theta + 2\varepsilon \text{ fraction} \]
\$\leq \theta \text{ fraction} \$

whp by Chernoff bounds

\$\leq 2\theta + 2\epsilon \text{ fraction} \$

\$\leq 1/2(\theta + \epsilon) \text{ fraction} \$

if Maj not faulty
whp by Chernoff bounds

\[
\leq \frac{1}{2}(\theta + \epsilon) \text{ fraction if Maj not faulty}
\]

\[
\leq \frac{1}{2}(\theta + \epsilon) + 2\epsilon \text{ fraction if Maj faulty w/ prob } \epsilon
\]

This last term is

\[
\leq \theta \text{ for large } \theta
\]
Issues

• Problem 1: Gates more constrained than processors.
• Problem 2: Faults assumed to be uncorrelated
• How to update the problem for distributed systems?
Issues

• Problem 1: Gates more constrained than processors.
• Problem 2: Faults assumed to be uncorrelated
• How to update the problem for distributed systems?
• Idea: Given a circuit. Use procs to simulate it. \( t < n/3 \) procs controlled by an adversary
n procs want to compute a function $f$ over $n$ inputs. $f$ can be computed with $m$ gates.

Each proc has one input of $f$

Up to $t < n/3$ procs are bad

[Yao '82]
SMC

- n procs want to compute a function $f$ over $n$ inputs. $f$ can be computed with $m$ gates.
- Each proc has one input of $f$
- Up to $t \leq n/3$ procs are bad

Note: The traditional SMC definition has additional privacy requirements that are ignored here
Applications as Functions

• Auctions

\[ f = \max(x_1, x_2, \ldots, x_n) \]

• Threshold cryptography

\[ f = M^s \mod pq \]

• Information aggregation

\[ f = \sqrt{\sum_{i=1}^{n} \frac{x_i^2}{n} - \left( \frac{\sum_{i=1}^{n} x_i}{n} \right)^2} \]
Applications as Functions

- Auctions

\[ f = \max(x_1, x_2, \ldots, x_n) \]

- Threshold cryptography

\[ f = \mathcal{M}^s \mod \mathcal{P}q \]

1) \( \mathcal{M}, p, q \) are parameters of the function;
2) \( s \) is the \( y \) intercept of a degree \((d - 1)\) function with points given by the \( x_i \) values.

- Information aggregation

\[ f = \sqrt{\frac{\sum_{i=1}^{n} x_i^2}{n} - \left( \frac{\sum_{i=1}^{n} x_i}{n} \right)^2} \]
Previous Work

• $f$ has $n$ variables and requires $m$ gates
• Previous work [see e.g. Goldreich ’98]
  • Each player sends $O(nm)$ messages
  • Each player performs $O(nm)$ computation.
Our Contribution

[DKMS ’12]

• Much improved computation & message cost
  • Each player sends $\tilde{O}(\frac{m + n}{n} + \sqrt{n})$ messages
  • Each player performs $\tilde{O}(\frac{m + n}{n} + \sqrt{n})$ computation.
Our Contribution

[DKMS ’12]

• Much improved computation & message cost
  • Each player sends $\tilde{O}(\frac{m + n}{n} + \sqrt{n})$ messages
  • Each player performs $\tilde{O}(\frac{m + n}{n} + \sqrt{n})$ computation.

• We solve SMPC w.h.p. meaning

  $1 - O(1/n^k)$ for any fixed $k$
Algorithm Overview

- Make critical use of *quorums*

- Each gate is computed by a quorum
Algorithm Overview

• Make critical use of quorums

  has $\theta(\log n)$ procs; less than $1/3$ are bad

• Each gate is computed by a quorum
Tools Used

- Can get all processors to agree on $n$ quorums w.h.p. [KS '11]
- HEAVY-WEIGHT-SMPC algorithm [BGW 88]
Algorithm Overview

- Translate function $f$ to circuit $C$
- Build network $G$ based on $C$
  - Gates $\rightarrow$ Internal nodes
  - Inputs $\rightarrow$ Input nodes
  - Wire $\rightarrow$ Communication Links
- Build quorums
- Each quorum is assigned to a node
Circuit and Network
Propagating Output

- Output reconstruction
- Output propagation
HPC Connection

• Want: Algorithms to monitor, collect and analyze data on large systems
HPC Connection

• Want: Algorithms to monitor, collect and analyze data on large systems

• Problem: Who watches the watchers?
HPC Connection

- Want: Algorithms to monitor, collect and analyze data on large systems
- Problem: Who watches the watchers?
- Need to design resilient tree-like circuit
- Solution: SMC
A Possible Agenda

• Step 1: Focus first on reliable OS tools
A Possible Agenda

• Step 1: Focus first on reliable OS tools tree-like circuits for data aggregation
A Possible Agenda

• Step 1: Focus first on reliable OS tools tree-like circuits for data aggregation

• Step 2: Solve problems based on these tools
A Possible Agenda

- Step 1: Focus first on reliable OS tools tree-like circuits for data aggregation

- Step 2: Solve problems based on these tools

- Step 3: Proven reliability of these tools attracts research attention, funding, etc
A Possible Agenda

• Step 1: Focus first on reliable OS tools tree-like circuits for data aggregation

• Step 2: Solve problems based on these tools

• Step 3: Proven reliability of these tools attracts research attention, funding, etc

• Step 4: Build on algorithmic techniques to full-fledged reliable applications &/or 13 dwarves
Towards a Research Agenda

“Make no little plans”

• **Important** problems span disciplines
• **Succinct** problems are remembered
• **Hard** problems pull in smart people
Questions
Dream Result

• Given: a parallel algorithm for n reliable procs

• Goal:
  • Design a reliable algorithm that is correct even if t procs are unreliable
  • Reliable algorithm has resource costs that are $O(t)$ larger in an additive sense
Problems with SMC

• Problem 1: To tolerate a linear number of faults, SMC requires logarithmic resource blowup

• This is still too large

• Idea: Amortization. Can we do better if same set of processors is used for many computations?
Problems with SMC

- Problem 2: SMC simulates a circuit
- Communication in a circuit is static
- Idea: develop version of SMC that simulates an arbitrary parallel algorithm