Randomized Smoothing Networks

Maurice Herlihy, Brown University
Srikanta Tirthapura, Iowa State University
Distributed Load Balancing

Example: Routing Tasks to Processors on a Grid
In a $k$-smoothing Network, the numbers of Tokens on different output wires differ by at most $k$
How to Build Smoothing Data Structures?

• Centralized Solution

• Random routing
  – Distributed and local
  – The smoothness can diverge without bound as more tokens enter the network

• Balancing Networks
Overview of Talk

• Describe Smoothing Networks built out of balancers

• Our Result: Randomized Smoothing Networks better than Deterministic Ones

• Analysis of the Randomized Butterfly (or Block) network
Basic Component: Balancer
Balancer
Balancer
Smoothing Networks: Scalable Distributed Data Structures

Bitonic[2]

Bitonic[4]
Prior Work: Deterministic Smoothing Networks

- The Network Balancers are initialized in a very specific way.
- (Usually) All balancers are pointing up.

Depth = 3, width = 4
Our Work: Randomized Smoothing Networks

- Random balancers – initialized as follows
  - With prob = ½, up
  - With prob = ½, down

- Resulting Initial Network State also random

- How good are the smoothing properties of randomized networks?
Benefits of Randomized Networks

1. Easy Initialization
   - Each balancer sets itself to a random state
   - Completely Local Actions

2. Easy to recover from faults
   - Global reconfiguration unnecessary

3. Better Smoothing Properties
Butterfly Network: Inductive Construction

Width = 2

Width = 4

Width = 8
Our Main Result

- Many Randomized Networks produce much smoother outputs than Regular Deterministically Initialized Networks

- For a Butterfly network of width $w$:
  - The output of a randomized network has smoothness $2.83\sqrt{\log w}$ with high probability
  - For any deterministic initialization, worst case output smoothness is no better than $O(\log w)$
Implications

- $2.83 \sqrt{\log w}$ nearly constant for practical purposes
  - If $w = 10^9$, this is 14
  - Smoothness independent of the number of tokens entering the network

- Network is very simple and easy to construct

- No further constants hidden in $O(\ldots)$
Previous Work

• Aspnes Herlihy and Shavit, 1991
  – Bitonic and Periodic Counting networks (which are also 1-smoothing)
  – Isomorphic to sorting networks
  – Width $w$, depth $O(\log^2 w)$

• Klugerman and Plaxton, 1992
  – Better depth constructions of counting networks
  – Width $w$, depth $\approx O(\log w)$
  – Constants are very high, not practical
Previous Work (2)

• Aiello, Venkatesan and Yung, 1993
  – Counting and Smoothing Networks using random and deterministic balancers
  – An alternate definition of a randomized balancer
    • Our results apply to their definition of random balancers also
  – In contrast, we use only randomized balancers

• Herlihy and Tirthapura, 2003
  – Worst case smoothness of deterministic Butterfly, Periodic and Bitonic networks
  – Matching upper and lower bounds
  – Self-stabilizing constructions
Analysis: Random balancer

a, b = numbers of tokens entering the balancer
c, d = numbers of tokens exiting the balancer

\[ r = \begin{cases} + \frac{1}{2} & \text{with probability } \frac{1}{2} \\ - \frac{1}{2} & \text{with probability } \frac{1}{2} \end{cases} \]

\[ c = \frac{a+b}{2} + D(a+b) \cdot r \]
\[ d = \frac{a+b}{2} - D(a+b) \cdot r \]

D = odd characteristic function
Analysis (2): Butterfly[8]

\[ y_1 = f(x_1, x_2, x_3, \ldots, x_8, r_1, r_2, \ldots, r_7) \]

\[ E[y_1] = (x_1 + x_2 + \ldots + x_8)/8 \]

Randomized Smoothing Networks

Randomized Smoothing Networks
Analysis (3)

Random variable $y_1$ is hard to work with directly.

Define Alternate Random Variable $z_1$:

$$z_1 = \frac{(r_1+r_2+r_3+r_4)}{4} + \frac{(r_5+r_6)}{2} + r_7 + \frac{(x_1+\ldots+x_8)}{8}$$
Analysis (4)

- Random Variable $z_1$ is easier to handle, since it is a linear function of $r_1, r_2, \ldots r_7$

\[ \Pr [y_1 > \delta] \leq 2 \Pr [z_1 > \delta] \]

- Use Hoeffding’s inequality to bound the tail of $z_1$, and hence $y_1$
Theorem

The output of Butterfly[w] with random balancers is $2.83\sqrt{\log w}$ smooth with probability at least $(1-4/w)$, irrespective of the input sequence.

Contrast: For any deterministic initialization, there exist inputs which lead to $(\log w)$ smooth outputs.
Conclusions

• Simple network (butterfly) with small depth and very good smoothness

• Easy, local reconfiguration after faults
Open Questions

• Do there exist simple, log-depth networks with constant output smoothness (with high probability)?

• Can the upper bound on the butterfly network be improved?

• Matching Lower bounds?

• Smoothing networks for tokens of unequal weights?