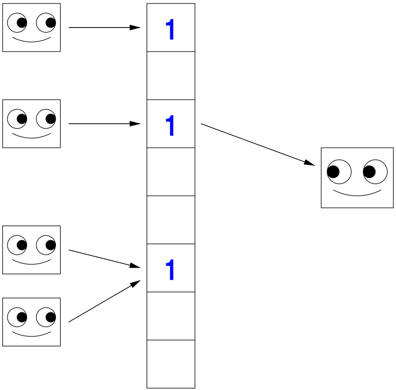
Approximate Shared-Memory Counting Despite a Strong Adversary

James Aspnes Keren Censor

January 5th, 2009

Model

- Processes can read and write shared **atomic registers**.
- Read on an atomic register returns value of last write.
- Timing of operations is controlled by an adversary.
- Cost of a high-level operation is number of low-level operations (register reads and writes) used.

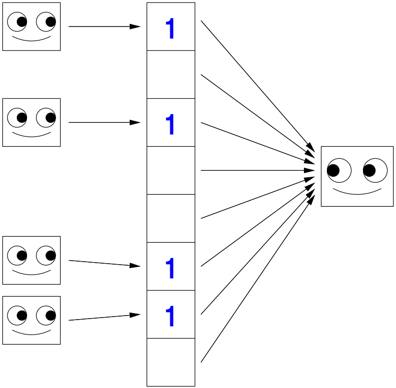


Approximate counting

- Each of n processes increments a **shared counter** at most once.
- Counter read operation should return number of increments within δ relative error with high probability.
- Cost of read should be $\ll n$.
  - $O(n/\log n)$ is enough for our intended application.
  - $\widetilde{O}(n^{4/5+\epsilon})$, for any fixed ϵ, is what we achieve.
- Counter must work despite **strong adversary** that can see internal states of processes.

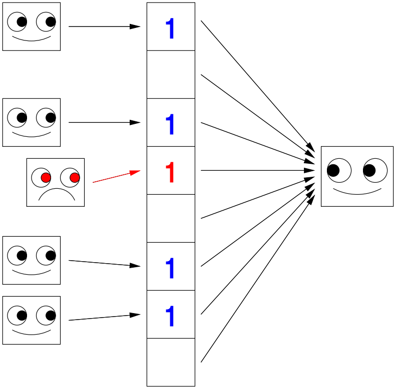
Counting by collect

- Each process writes its increment to a separate register.
- To read the counter, read all registers and add them up. (This takes $\Theta(n)$ time!)
- Counter read always includes writes that finish before read starts.



Latecomers

- If a write starts before the collect finishes, reader may or may not read it.
- OK as long as total returned by collect doesn't exceed number of writes finished or in progress.



Approximate counting

We want a counter that acts like the simple collect, but will sacrifice accuracy for speed.
Counter read is δ-accurate if it:

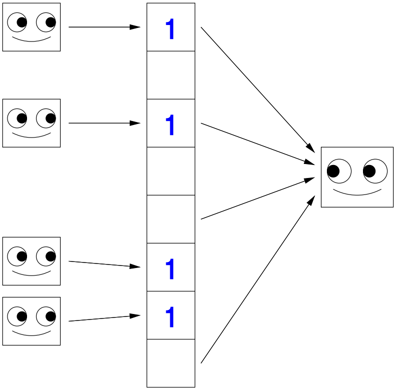
1. Returns at least $(1 - \delta)$ times the number of increments that finish before the read starts.
2. Returns at most $(1 + \delta)$ times the number of increments that start before the read finishes.

(This is a pretty weak guarantee.)

Counting by sampling

Instead of reading all registers, randomly sample s registers and multiply by n/s.

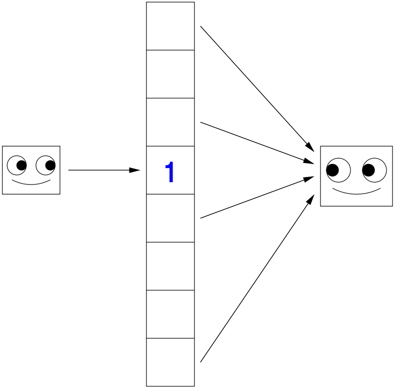
- With no concurrent increments, gives predictable additive error w.h.p. (standard Chernoff bounds).



Counting by sampling

Instead of reading all registers, randomly sample s registers and multiply by n/s.

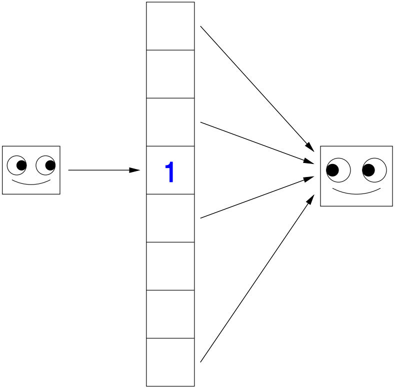
- With no concurrent increments, gives predictable additive error w.h.p. (standard Chernoff bounds).
- Danger of **undercount** with $\ll n/s$ increments.



Counting by sampling

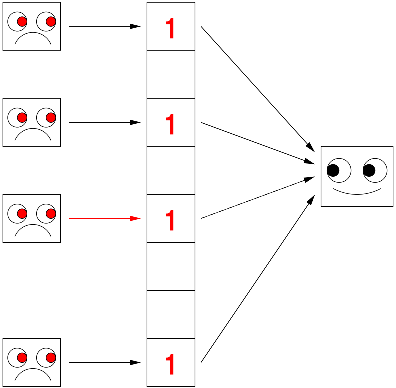
Instead of reading all registers, randomly sample s registers and multiply by n/s.

- With no concurrent increments, gives predictable additive error w.h.p. (standard Chernoff bounds).
- Danger of **undercount** with $\ll n/s$ increments.
- Danger of **overcount** if adversary controls concurrent writes.



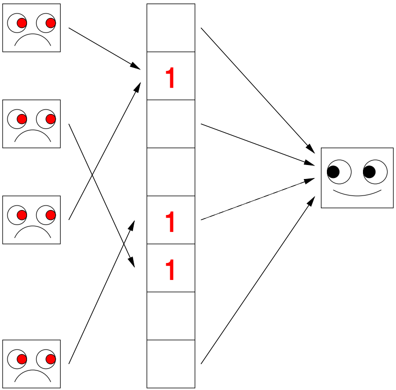
Potemkin village attack

- Strong adversary controls all timing and can see where reader is about to look.
- So it rushes an increment into each register the reader is about to read.
- Amazing! Ones everywhere!
- Reader always returns n.



Two-sided sampling

- Incrementers also write to random locations.
- Collisions are reduced by using $N \gg n$ registers.
- Adversary can't cause overcount with late increments: each new increment only increases chance of 1 in target register by $1/N$.
- But undercount problem gets worse: granularity is now N/s.



Sampling counter: details

Fix small $\epsilon > 0$ and let $s = n^{4/5+\epsilon}$, $N = n^{6/5+\epsilon/4}$.

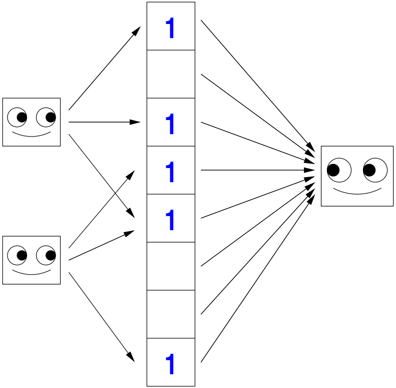
- Expected increments lost to collisions is $O(n^2/N) = O(n^{4/5-\epsilon/4})$.
- Completed increments are sampled with standard deviation $O((N/s)\sqrt{s}) = O(n^{4/5-\epsilon/4})$; stock Chernoff bounds give bound on undercounts.
- Concurrent increments may depend in odd ways on behavior of adversary, but a supermartingale argument and appropriate tail bound give a similar bound on overcounts.

Result: After $n^{4/5}$ increments, probability that a single call to sampling read is δ-inaccurate is at most $\exp\left(-\frac{\delta^2 n^{\epsilon/2}}{2}\right)(1+o(1))$ $=$ small.

Small numbers of increments

Use a second counter for few increments:

- Each incrementer now writes $D = \widetilde{O}(\log^{O(1/\epsilon)} n)$ of $\widetilde{O}(n^{4/5+\epsilon})$ registers.
- Reader reads all the registers and divides by D.
- Write locations are chosen using an **expander** $\Rightarrow k$ increments give between $(1-\delta)Dk$ and Dk ones.
- Fails only after sampling counter starts working.



Combined counter

The full counter combines the two components:

- Incrementer increments sampling counter first, then expander counter.
- Reader checks expander counter first, then checks sampling counter if expander overflows.
- Since sampling counter is always $\geq$ expanding counter, sampling counter is only used in its accurate range.

Result: δ-accurate approximate counter w.h.p. in $\widetilde{O}(\log^{O(1/\epsilon)} n)$ register writes per increment and $\widetilde{O}(n^{4/5+\epsilon})$ register reads per counter read.

Randomized consensus

- Want n processes to agree on a bit despite asynchrony and up to $n-1$ **halting failures**.
- Impossible for deterministic algorithms with even one failure (Fischer-Lynch-Paterson 1985; Loui and Abu-Amara 1987).
- Possible using randomization even with strong adversary.

Randomized consensus: total work

- Exponential-time algorithm (Abrahamson 1988).
- Reduction to **random voting** (Aspnes-Herlihy 1989).
  - Generate $\Theta(n^2)$ random ± 1 votes.
  - Use counter to test if we have enough votes.
  - $\Omega(n)$ standard deviation beats votes hidden in dead processes with constant probability.
  - First polynomial-time algorithm ($O(n^6)$).
- Only check termination every $\Theta(n/\log n)$ votes (Bracha-Rachman 1991) $\Rightarrow$ $O(\log n)$ amortized cost to check counter $\Rightarrow$ $O(n^2 \log n)$ **total work** (but same **individual work**).
- Use termination flag to stop voting when one process notices termination (Attiya-Censor 2007) $\Rightarrow$ only need to check every $\Theta(n)$ votes $\Rightarrow$ $O(1)$ amortized cost per vote $\Rightarrow$ $O(n^2)$ total work. Also shown to be optimal.

Randomized consensus: individual work

Simple ± 1 voting may force one process to generate all $\Omega(n^2)$ votes itself. What if we want each process to only do $O(n)$ operations?

- **Weighted voting** (Aspnes-Waarts 1996).
  - Faster processes cast bigger votes.
  - Have to check termination slightly more often to avoid runaway big votes.
  - With Bracha-Rachman-style termination test, individual work is $O(n \log^2 n)$ ($= O(n^2 \log^2 n)$ total work, worse that Bracha-Rachman).
- Attiya-Censor termination bit reduces cost to $O(n \log n)$ (Aspnes-Attiya-Censor 2008).
  - Main limitation is each process still checks counter $\Omega(\log n)$ times $\Rightarrow \Omega(n \log n)$ cost with simple counter.
- New result: (AAC 2008) + sublinear counter + much pushing and shoving $\Rightarrow O(n)$ individual work. This is optimal by previous lower bound of (Attiya-Censor 2007).

What's left?

- Randomized consensus: pretty much done (in this model).
- Further counter improvements:
  - Practical time complexity.
  - Exact counting.
  - Linearizability.
  - Unbounded increments.

What's left?

- Randomized consensus: pretty much done (in this model).
- Further counter improvements:
  - Practical time complexity.(*)
  - Exact counting.(*)
  - Linearizability.
  - Unbounded increments.

(*) Can get deterministic exact counting with $O(\log^2 n)$ cost for increments and $O(\log n)$ for reads. (Aspnes-Attiya-Censor, in preparation.)