Design of a Novel Statistics Counter Architecture with Optimal Space and Time Efficiency

Qi Zhao   Jun (Jim) Xu
College of Computing
Georgia Institute of Technology

Zhen Liu
IBM T.J. Watson Research Center

SIGMETRICS 2006/PERFORMANCE 2006

Problem Statement

- To maintain a large array (say millions) of counters that need to be incremeted (by 1) in an arbitrary fashion (i.e., $A[i_1] + +$, $A[i_2] + +$, ...)
- Increments may happen at very high speed (say one increment every 10ns) – has to use high-speed memory (SRAM)
- Values of some counters can be very large
- Fitting everything in an array of “long” (say 64-bit) SRAM counters can be expensive
- Possibly lack of locality in the index sequence (i.e., i_1, i_2, ...) – forget about caching

Main Idea in Previous Approaches [SIPM:2001,RV:2003]

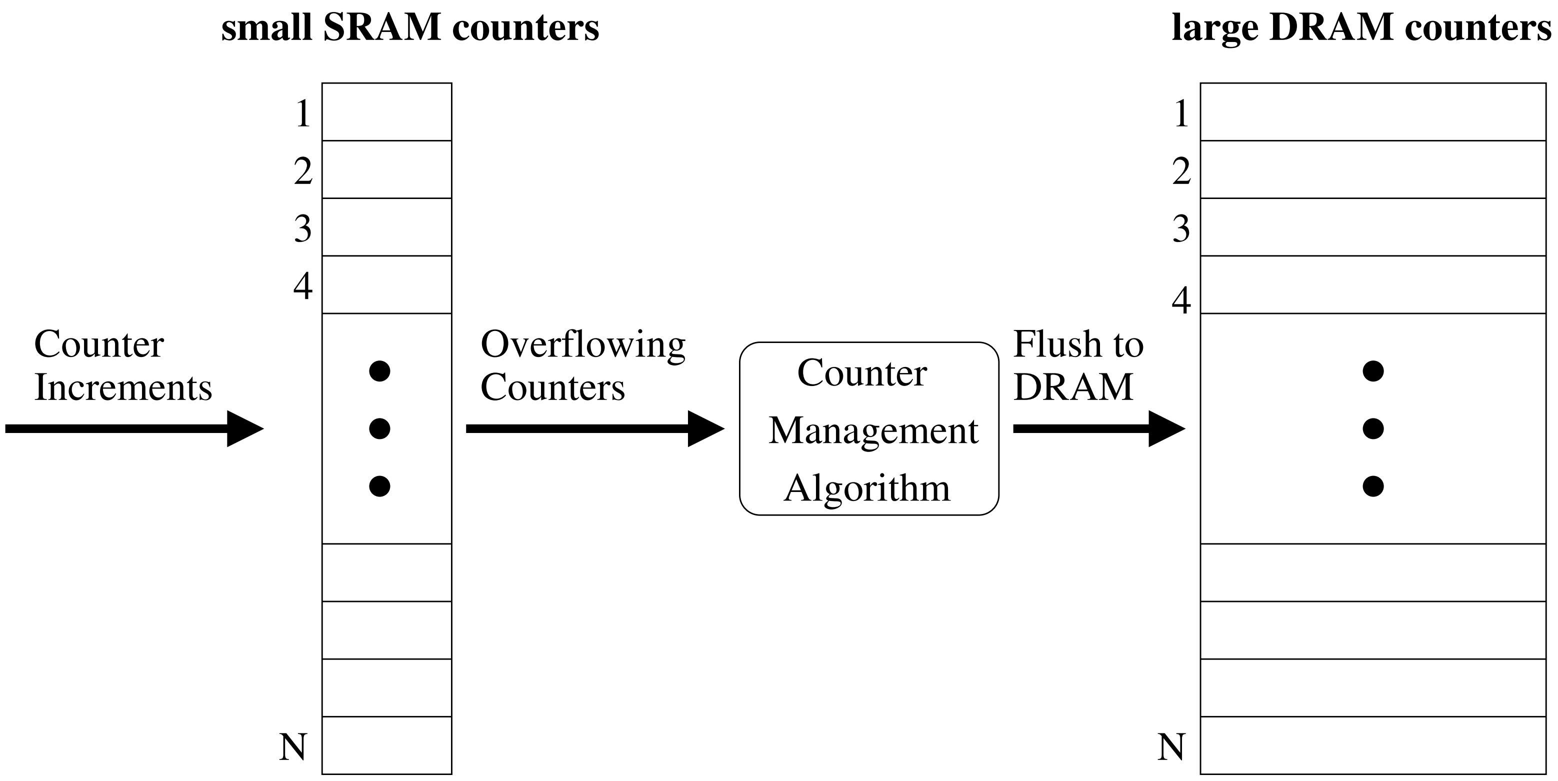

Figure 1: Hybrid SRAM/DRAM counter architecture

CMA used in [SIPM:2001]

- D. Shah, S. Iyer, B. Prabhakar, and N. McKeown, “Maintaining statistics counters in router line cards”, *Hot Interconnects 2001*
- Implemented as a priority queue (fullest counter first)
- About 28 bits per counter (when S/D is 10)
- Need pipelined hardware implementation of a heap.

CMA used in [RV:2003]

- S. Ramabhadran and G. Varghese, "Efficient implementation of a statistics counter architecture", *ACM SIGMETRICS 2003*
- SRAM counters are tagged when they are at least half full
- Scan from left to right to periodically flush (half-full)$^+$ SRAM counters, and maintain a small priority queue to preemptively flush the SRAM counters that rapidly become completely full
- Pipelined hierarchical bitmap data structure to find out "Who's the next (half-full)$^+$?" in $\log(N)$ time
- 8 SRAM bits per counter for storage and 2 bits per counter for the bitmap control logic, when S/D is 10.

Our scheme

- Our scheme only needs 4 SRAM bits when S/D is 10.
- Flush only when an SRAM counter is “completely full” (e.g., when the SRAM counter value changes from 15 to 16 assuming 4-bit SRAM counters).
- Use a small (say hundreds of entries) SRAM FIFO buffer to hold the indices of counters to be flushed to DRAM
- Key innovation: a simple randomized algorithm to ensure that counters do not overflow in a burst large enough to overflow the FIFO buffer, with overwhelming probability
- Our scheme is provably space-optimal (e.g., 3 bits will never work when S/D is 10).

The randomized algorithm

- Set the initial values of the SRAM counters to independent random variables uniformly distributed in $\{0, 1, 2, ..., 15\}$ (i.e., $A[i] := uniform\{0, 1, 2, ..., 15\}$).
- Set the initial value of the corresponding DRAM counter to the negative of the initial SRAM counter value (i.e., $B[i] := -A[i]$).
- Adversaries know our randomization scheme, but not the initial values of the SRAM counters
- We prove rigorously that a small FIFO queue can ensure that the queue overflows with very small probability

A numeric example

- One million 4-bit SRAM counters (512 KB) and 64-bit DRAM counters with SRAM/DRAM speed difference of 12
- 300 slots ($\approx$ 1 KB) in the FIFO queue for storing indices to be flushed
- After 10^{12} counter increments in an arbitrary fashion
- The probability of overflowing from the FIFO queue: less than 10^{-14} in the worst case

Timing diagram of the hardware operation

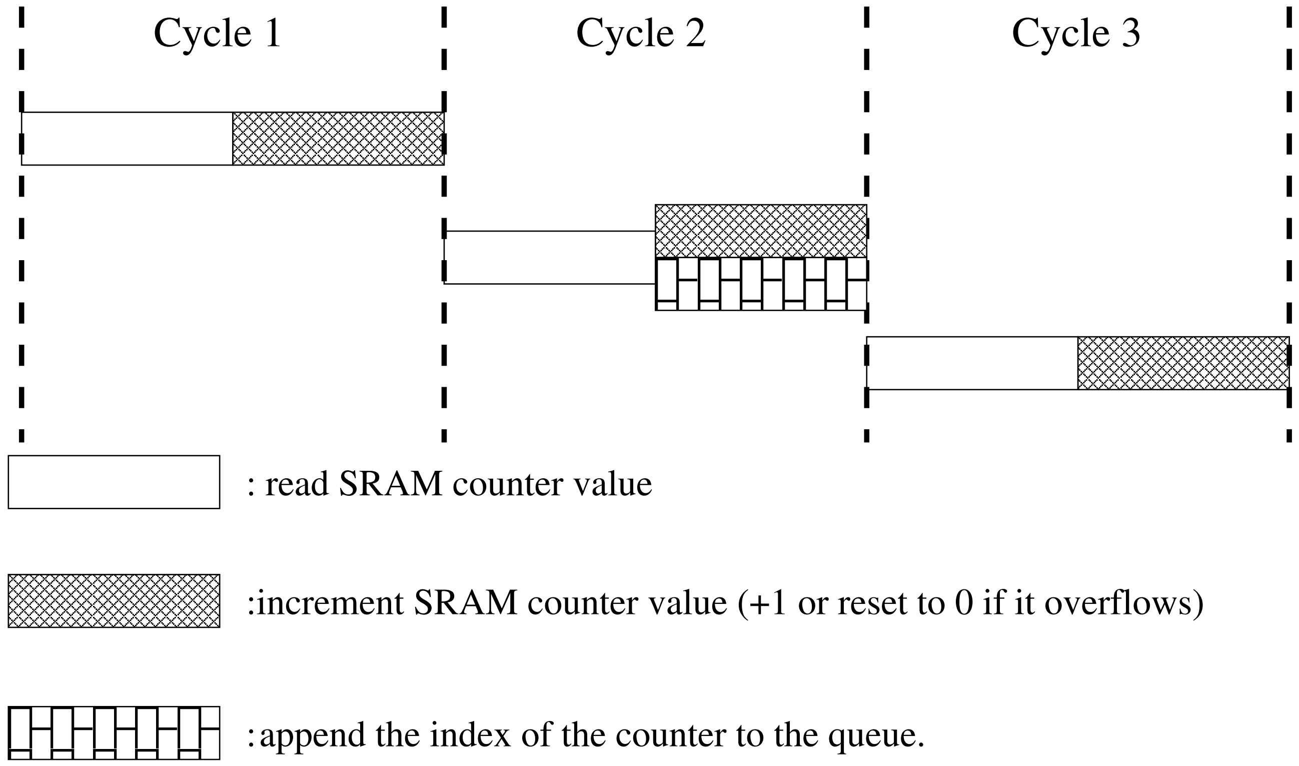
Cycle 1
Cycle 2
Cycle 3
: read SRAM counter value
:increment SRAM counter value (+1 or reset to 0 if it overflows)
:append the index of the counter to the queue.

Figure 2: Hybrid SRAM/DRAM counter architecture

Tail bound analysis – Intuition

- The average departure rate of the FIFO queue is the speed of DRAM (e.g, 1 departure every 12 cycles or with the rate 1/12 when S/D is 12)
- The average arrival rate to the FIFO queue is approximately 1/16, as it takes 16 increments for a counter to become full – and hopefully the randomization makes the arrival process very smooth!
- Actually, our experimental result is very close to that of the Geom/D/1 queue
- However, we are NOT able to prove that our queueing process is stochasticly comparable to (or bounded by) that of a Geom/D/1 queue – only able to prove much weaker tail bounds

Tail bound analysis (1st step)

- Let D be the event that the FIFO queue overflows after n increments.
- Let $D_{s,t}$ be the event that the number of arrivals during the time interval $[s,t]$ is larger than the maximum possible number of departures from the FIFO queue (even if serving continuously), by more than the queue size K.
- Lemma 1: $D \subseteq \bigcup_{0\le s\le t\le n} D_{s,t}$ (proved using standard busy period arguments)
- Therefore

$$\Pr[D] \le \Pr[\bigcup_{0\le s\le t\le n} D_{s,t}] \le \sum_{0\le s\le t\le n} \Pr[D_{s,t}]$$

Bounding $\Pr[D_{s,t}]$ using Chernoff bound

- Let c_j, $j = 1, 2, ..., N$ be the number of increments to counter j during time period $[s, t]$ – note our bound will be independent of these c_j values (note $\sum_{j=1}^{N} = n$)
- Let b_j be the number of “flush to DRAM” requests generated by the counter j during the time interval $[s, t]$
- It can be shown that $b_j - E[b_j]$, $j = 1, 2, ..., N$, are independent Bernoulli RV’s:

$$b_j = \begin{cases} \lfloor \frac{c_j}{2^l} \rfloor & \text{with probability } 1 - \{2^{-l} c_j\}, \\ \lfloor \frac{c_j}{2^l} \rfloor + 1 & \text{with probability } \{2^{-l} c_j\}. \end{cases} \qquad (1)$$

Chernoff bound on sum of independent Bernoulli RV's

- Lemma 3, Let $X_1, X_2, \ldots, X_m$ be mutually independent random variable such that, for $1 \leq j \leq m$, $\Pr[X_j = 1 - p_j] = p_j$ and $\Pr[X_j = -p_j] = 1 - p_j$, where $0 < p_j < 1$. Then, for $X = \sum_{j=1}^{m} X_j$ and $a > 0$,

$$\Pr[X > a] < e^{-2a^2/m}$$

- Applying to the sum of $b'_j s$, we obtain Theorem 2:
  For any $s < t$, let $\tau = t - s$.

$$\Pr[D_{s,t}] \equiv \Pr[b(s,t) - \mu\tau > K] < e^{-2(K+\mu\tau-2^{-l}\tau)^2/\min\{\tau,N\}} \tag{2}$$

Using 2nd Moment Information to Obtain a New Bound of $\Pr[D_{s,t}]$

$$VAR[b(s,t)] \le \begin{cases} \frac{N}{4} & t-s \ge 2^{l-1}N, \\ \frac{(2^l - \frac{t-s}{N})(t-s)}{2^{2l}} & N \le t-s < 2^{l-1}N, \\ \frac{(2^l-1)(t-s)}{2^{2l}} & 0 < t-s < N. \end{cases}$$

There is implicitly a quasi minimax analysis in it – imaging that the adversary has control over the increment index sequence

A New Tail Bound Theorem

- Given any $\theta > 0$ and $\epsilon > 0$, the following holds: Let $W_j, 1 \leq j \leq m$, m arbitrary, be independent random variables with $EXP[W_j] = 0$, $|W_j| \leq \theta$ and $VAR[W_j] = \sigma_j^2$. Let $W = \sum_{j=1}^{m} W_j$ and $\sigma^2 = \sum_{i=1}^{m} \sigma_j^2$ so that $VAR[W] = \sigma^2$. Let $\delta = \ln(1+\epsilon)/\theta$. Then for $0 < a \leq \delta\sigma$,

$$\Pr[W > a\sigma] < e^{-\frac{a^2}{2}(1-\frac{\epsilon}{3})}$$

- Mapping to our problem, it becomes

$$\begin{array}{ll} \textit{maximize} & \frac{a^2}{2}(1-\frac{\epsilon}{3}) \\ \textit{subject to} & 0 < a \leq \delta\sigma \\ & e^{\delta} - 1 \leq \epsilon < 3 \\ & a\sigma \leq K + \mu\tau - 2^{-l}\tau \end{array}$$

The Hybrid Tail Bound

- Recall that $\Pr[D] \leq \sum_{0 \leq s \leq t \leq n} \Pr[D_{s,t}]$
- We derived the first bound $\Pr[D_{s,t}] \leq \Omega_1(s,t)$ using Chernoff bound
- We derived the second bound $\Pr[D_{s,t}] \leq \Omega_2(s,t)$ using our new tail bound theorem
- The first bound is better for most of the s, t values, BUT the second bound can be much better for some critical s, t values
- We refer to $\Pr[D] \leq \sum_{0 \leq s \leq t \leq n} \min\{\Omega_1(s,t), \Omega_2(s,t)\}$ as the hybrid bound

Numerical Examples

Given $N = 10^6$, $n = 10^{12}$, $\mu = 1/30$ and $l = 5$ bits,

K	First	Second	Hybrid
500	trivial (≥ 1)	trivial (≥ 1)	1.1×10^{-11}
3033	1.4×10^{-6}	trivial (≥ 1)	8.7×10^{-142}

Cost-benefit Comparison

Given $l = 64$ bits, $\mu = 1/30$, and $K = 500$ slots

	Naive	LCF	$LR(b)$	Ours
Counter memory	64Mb SRAM	9Mb SRAM 64Mb DRAM	9Mb SRAM 64Mb DRAM	5Mb SRAM 65Mb DRAM
Control memory	None	20Mb SRAM	2Mb SRAM	10Kb SRAM
Control logic	None	Hardware heap	Aggregated bitmap	FIFO queue
Implementation Complexity	Very low	High	Low	Very Low

Simulation Using Real-world Internet Traffic

- Given $N = 1,000,000$, $n = 10^{12}$, $\mu = 1/30$ and $l = 5$ bits,

Trace	SRAM counter size (in bits)	μ	Queue Size	
			Max	Average
USC	4	1/12	21	1.6
	5	1/30	61	6.0
UNC	4	1/12	23	1.7
	5	1/30	72	7.0

- Computing the hybrid bound, we need 228 slots for the bound to be nontrivial
- The experimental result is in fact very close to that of Geom/D/1 queue (average is 1.6).
- The experimental result is much better than the bound because (1) The input is not adversarial, and (2) The union bound $\Pr[\bigcup_{0\le s\le t\le n} D_{s,t}]$ $\sum_{0\le s\le t\le n} \Pr[D_{s,t}]$ is very lossy

Conclusion

- A simple and efficient counter management algorithm for hybrid SRAM/DRAM counter architecture
- Statistical guarantee for queue overflow probability
- A new tail bound theorem for the sum of independent random variables that can take advantage of both their independence and their overall low variance

Future Work

- Further improve the theoretical bound by possibly ditching the union bound
- Allow for both increments and decrements – this algorithm won't work since an adversary can create thrashing around 0.
- Apply the counter array work to other network applications (e.g., for implementing millions of token buckets).

Thank You!

ANY QUESTIONS?

Concern over heavy traffic through system bus

- Concern: shorter SRAM counter size means that much larger flusing traffic through the system bus, when the SRAM array is on the L1 cache of a network processor
- “Victim of our own success”: previous schemes are constrained by the lower efficiencies of their CMA algorithms, not by the concern that there will be too much bus traffic
- We intend our scheme/algorithm to be generic and we do not want to bind it to any particular architecture choice just like in previous works.
- The heavy traffic over the bus may not be an issue in many scenarioes: (a) a computer architecture can have a dedicated bus between CPU and memory (b) the system is built for network monitoring only (e.g., Sprint’s CMON)