Design and Performance Analysis of a DRAM-based Statistics Counter Array Architecture

Chuck Zhao[1] Hao Wang[2] Bill Lin[2] **Jim Xu**[1]

[1]Georgia Tech

[2]UCSD

October 2nd, 2008

Broader High-Level Question

- What are the “cross-layer" opportunities between evolving technologies and network measurement functions?
- Will use wirespeed statistics counting as a concrete example where previous approaches have treated DRAM as a “blackbox" with overly pessimistic assumptions.
- Other “cross-layer" opportunties possible with evolving technologies (e.g., solid state disks, many cores, etc).

Statistics Counting Wish List

- Fine-grained network measurement
  Possibly tens of millions of flows (and counters)
- Wirespeed statistics counting
  8 ns update time at 40 Gb/s
- Arbitrary increments and decrements
  e.g., byte counting for variable-length packets
- Different number representations
  unsigned and signed integers, floating point numbers
  e.g., entropy-based algorithms need floating point

Conventional Wisdom

- SRAM is needed for speed requirements, but DRAM is needed to provide the storage capacity

  e.g., 10 million counters x 64-bits = 80 MB, prohibitively expensive (infeasible for on-chip)
- SRAM is either infeasible or very expensive, but DRAM makes it difficult to support high line rates

  e.g., 50 ns DRAM random access times typically quoted, 2 x 50 ns = 100 ns » 8 ns required for wirespeed updates (read, increment, then write)

Conventional Wisdom

- The prevailing view that DRAM is too slow is generally held for other structures

  e.g., Bloom filters, flow tables, etc.
- A different view: DRAM is **plenty fast** for network measurement primitives if one considers modern advances in DRAM architectures (e.g. driven by video games)
- Will use statistics counting as driving example

Hybrid SRAM/DRAM architectures

- Based on premise that DRAM is too slow, hybrid SRAM/DRAM architectures have been proposed

  e.g., Shah'02, Ramabhadran'03, Roeder'04, Zhao'06
- All based on following idea:
  1. Store full counters in DRAM (64-bits)
  2. Keep say a 5-bit SRAM counter, one per flow Wirespeed increments on 5-bit SRAM counters
  3. "Flush" SRAM counters to DRAM before they "overflow"
  4. Once "flushed", SRAM counter won't overflow again for at least say another $2^5 = 32$ (or 2^b in general) cycles

But, Still Requires Significant SRAM

For 16 million counters
e.g. UNC traces [Zhao'06] had 13.5 million flows

- 10 to 57 MB needed far exceed available on-chip SRAM
- On-chip SRAM needed for other network processing
- SRAM amount depends on "how often" SRAM counters have to be flushed - if arbitrary increments are allowed (e.g. byte counting), more SRAM needed
- Integer specific, no decrements

Main Observation

- Modern DRAMs are fast

  Driven by insatiable appetite for extremely aggressive memory data rates in graphics, video games, and HDTV at commodity pricing, just $0.01/MB currently, $20 for 2GB!

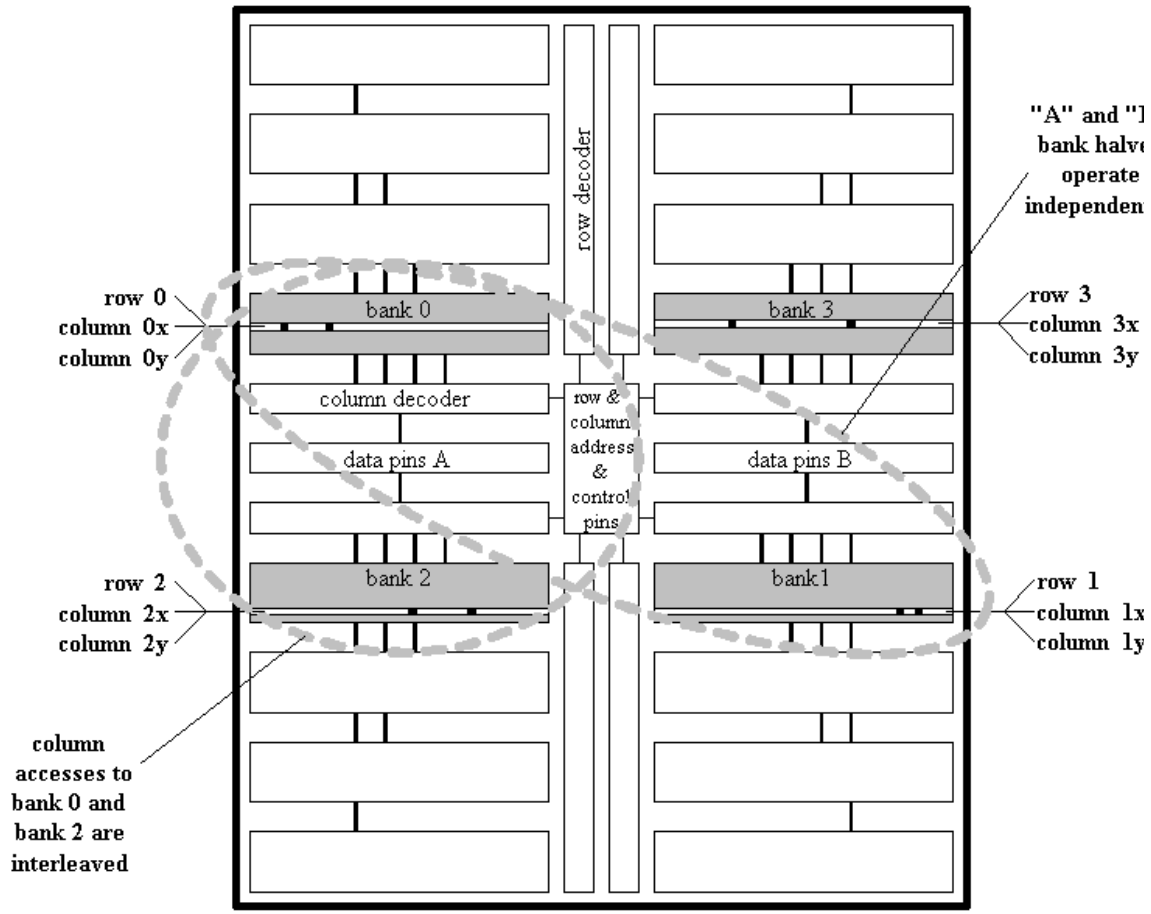
- Example: Rambus XDR Memory

  16 GB/s per 16-bit memory channel
  64 GB/s on dual 32-bit channels (e.g. on IBM cell)

  Terabyte/s on roadmap !

Example: Rambus XDR Memory

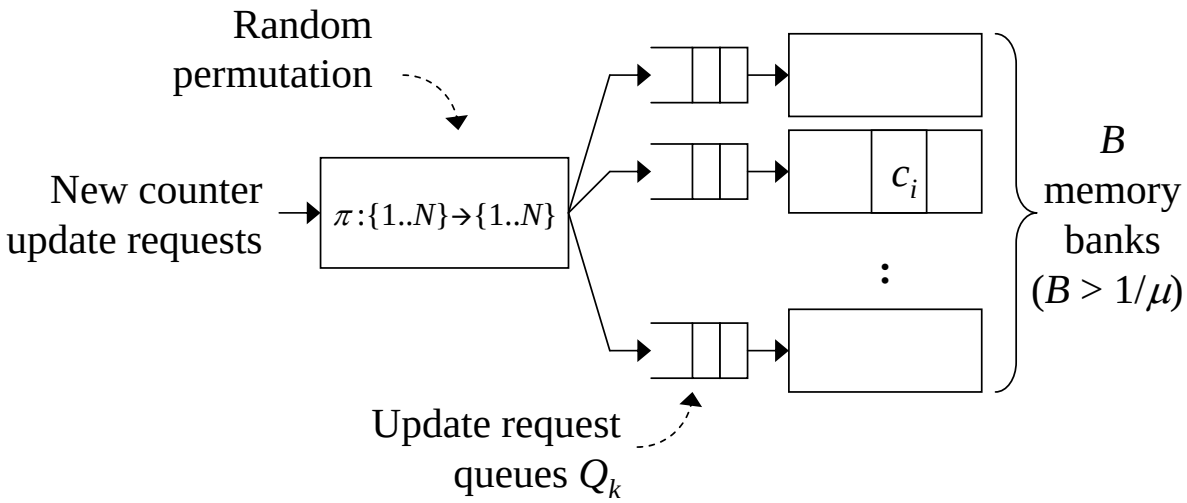
- 16 internal banks



Basic architecture: Randomized Scheme

- Counters randomly distributed across B memory banks

  $B > 1/\mu$, where μ is the SRAM-to-DRAM access latency ratio



Basic architecture: Randomized Scheme

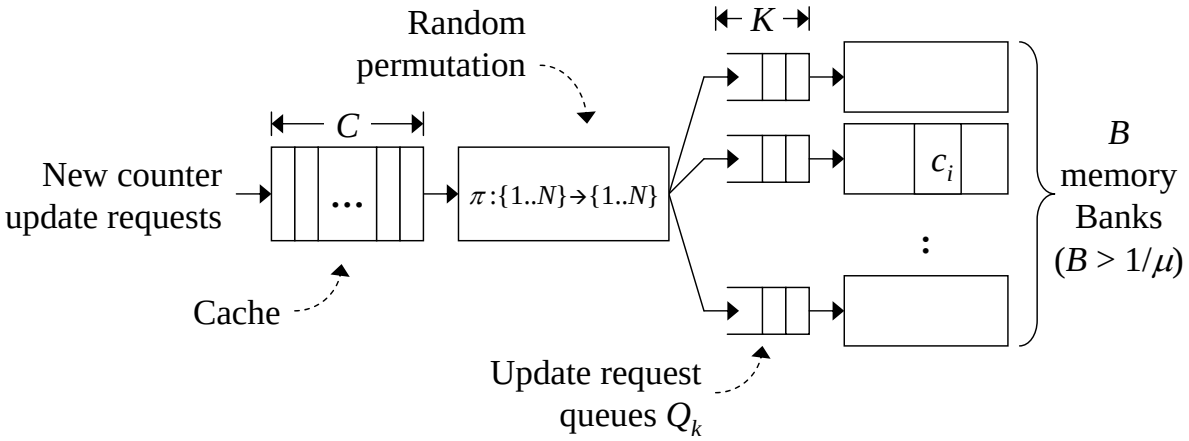
- Conceptually the request queues are serviced concurrently
- In practice, groups of request queues can be serviced round-robin

  Eg. $\mu = 1/16, B = 32$, can use two XDR memory channels, for each channel its 16 banks are serviced round-robin

Extended architecture to handle adversaries.

- Cache module absorbs repeated updates to the same address

  Cache implements FIFO policy



Union Bound

- Want to bound the probability that a request queue will overflow in n cycles

$$\Pr[\text{Overflow}] \leq \sum_{0 \leq s \leq t \leq n} \Pr[D_{s,t}]$$

$$D_{s,t} \equiv \{\omega \in \Omega : X_{s,t} - \mu\tau > K\}$$

- $X_{s,t}$ is the number of updates to the bank during cycles $[s, t]$, $\tau = t - s$, K is length of request queue.
- For total overflow prob. bound multiply by B.

Chernoff Bound

$$\begin{aligned}\Pr[D_{s,t}] &= \Pr[X > K + \mu\tau] \\ &= \Pr[e^{X\theta} > e^{(K+\mu\tau)\theta}] \\ &\leq \frac{E[e^{X\theta}]}{e^{(K+\mu\tau)\theta}} \quad \text{(Markov inequality).}\end{aligned}$$

Since this is true for all $\theta > 0$,

$$\Pr[D_{s,t}] \leq \min_{\theta>0} \frac{E[e^{X\theta}]}{e^{(K+\mu\tau)\theta}}. \tag{1}$$

Want to find worst case update sequence for $E[e^{X\theta}]$.

A few definitions

Definition (Majorization)

For any n-dimensional vectors a and b, let $a_{[1]} \geq \ldots \geq a_{[n]}$ denote the components of a in decreasing order, and $b_{[1]} \geq \ldots \geq b_{[n]}$ denote the components of b in decreasing order. We say a is *majorized* by b, denoted $a \leq_M b$, if

$$\begin{cases} \sum_{i=1}^{k} a_{[i]} \leq \sum_{i=1}^{k} b_{[i]}, & \text{for } k = 1, \ldots, n-1 \\ \sum_{i=1}^{n} a_{[i]} = \sum_{i=1}^{n} b_{[i]} \end{cases} \tag{2}$$

Eg. $(1, 1, 1,) \leq_M (0, 1, 2) \leq_M (0, 0, 3)$.

A few definitions

Definition (Exchangeable random variables)

A sequence of random variables $X_1, \ldots, X_n$ is called *exchangeable*, if for any permutation $\sigma : [1, \ldots, n] \to [1, \ldots, n]$, the joint probability distribution of the permuted sequence $X_{\sigma(1)}, \ldots, X_{\sigma(n)}$ is the same as the joint probability distribution of the original sequence.

Eg. i.i.d. RVs are exchangeable

Eg. sampling without replacement gives exchangeable RVs

A few definitions

Definition (Convex function)

A real function f is called *convex*, if $f(\alpha x + (1-\alpha)y) \leq \alpha f(x) + (1-\alpha)f(y)$ for all x and y and all $0 < \alpha < 1$.

Definition (Convex order)

Let X and Y be random variables with finite means. Then we say that X is less than Y in convex order (written $X \leq_{cx} Y$), if $E[f(X)] \leq E[f(Y)]$ holds for all real convex functions f such that the expectations exist.

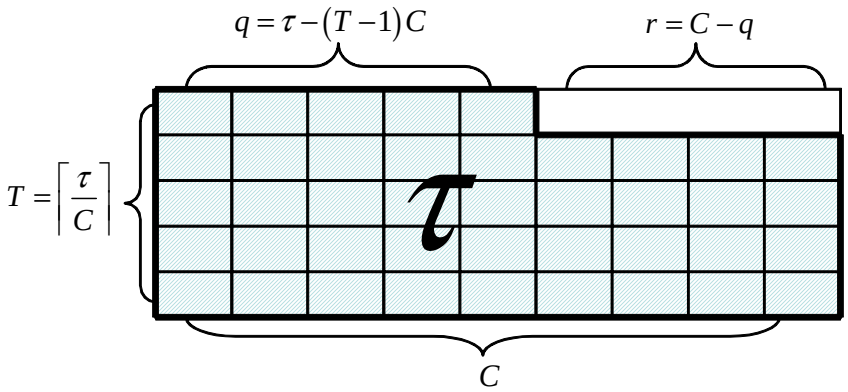
A Useful Theorem

The following theorem from Marshall relates majorization, exchangeable random variables and convex order together.

Theorem

If $X_1, \ldots, X_n$ are exchangeable random variables, a and b are n-dimensional vectors, then
$a \leq_M b$ implies $\sum_{i=1}^{n} a_i X_i \leq_{cx} \sum_{i=1}^{n} b_i X_i$.

Valid splitting pattern of τ



- During time $\tau = t - s$ each counter updated m_i times, $\sum m_i = \tau$
- Access to same address not repeated within C cycles, so $m_i \leq \lceil \tau / C \rceil \equiv T$.
- Number of m_i equal to T is at most q

Worst Case update sequence

- $q + r$ requests for distinct counters $a_1, ..., a_{q+r}$
- repeat $T - 1$ times in total
- q requests for counters $a_1, ..., a_q$
- worst case pattern m^*:
  $m_1^* = ... = m_q^* = T, m_{q+1}^* = ... = m_{q+r}^* = T - 1$, rest 0.

Proof for Worst Case

- $X_m \equiv \sum m_i X_i$, where X_i is indicator R.V. for whether address i is mapped to the bank
- X_i's are exchangeable
- For any m, $m \leq_M m^*$ by design
- From previous theorem, $X_m = \sum m_i X_i \leq_{cx} m_i^* X_i = X_{m^*}$, so m_* is worst case in convex order

Applying Chernoff bound

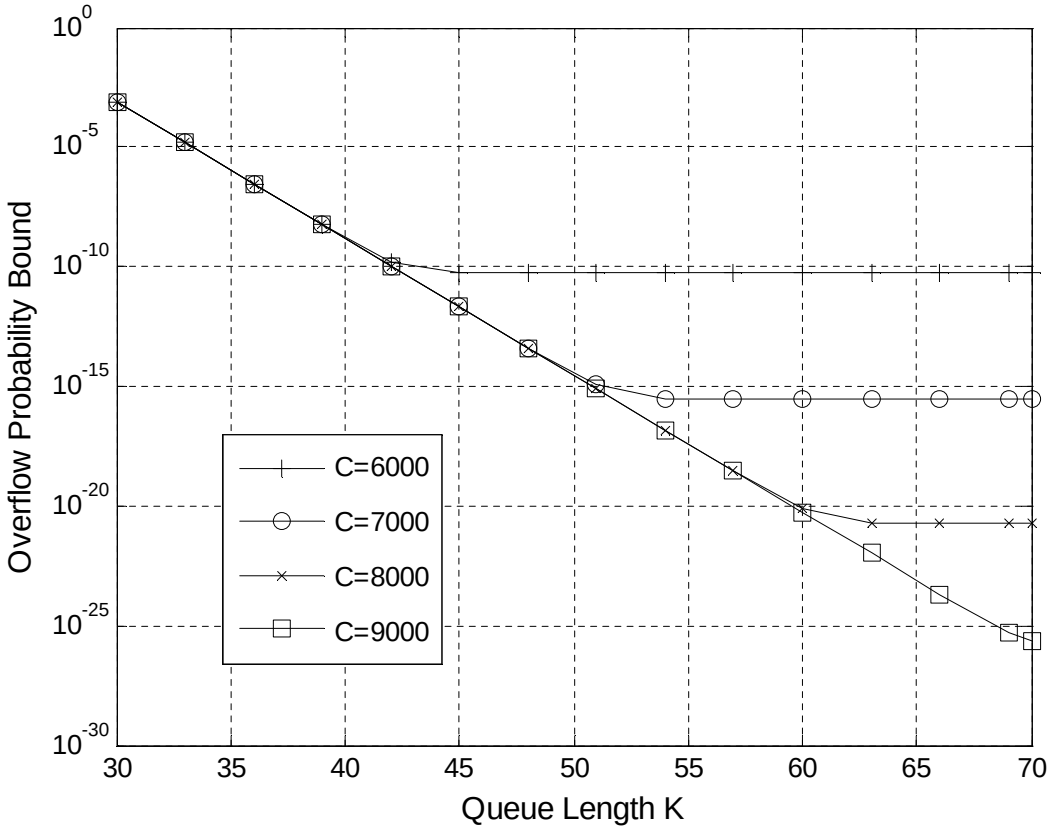
- $e^{x\theta}$ is a convex function, so $X_m \leq_{cx} X_{m^*}$ implies that $E[e^{X_m\theta}] \leq E[e^{X_{m^*}\theta}]$

$$\Pr[D_{s,t}] \leq \min_{\theta>0} \frac{E[e^{X_m\theta}]}{e^{(K+\mu\tau)\theta}}$$
$$\leq \min_{\theta>0} \frac{E[e^{X_{m^*}\theta}]}{e^{(K+\mu\tau)\theta}}$$

- We reduced arbitrary update sequence to one worst case update sequence
- $E[e^{X_{m^*}\theta}]$ can be bounded by sum of i.i.d. random variables (for details see paper)

Overflow Probability

Overflow probability for 16 million counters, $\mu = 1/16$, $B = 32$.



Memory Usage Comparison

	Naive	Hybrid SRAM/DRAM	Ours
Counter DRAM	None	128M DRAM	128M DRAM
Counter SRAM	128M SRAM	8M SRAM	None
Control	None	1.5K SRAM	25K CAM, 5.5K SRAM

Work-in-progress

Generalizing proposed randomized scheme to broader abstraction of “fixed-delay SRAM"

- Enable “read" and “write" memory transactions at SRAM throughput with fixed pipeline delay
- Under fairly broad conditions, not only “block" access typically assumed in graphics
- Per-hop delay at core routers today typically >10ms, corresponding to >1000 cycles » b (e.g. b = 16 cycles is relatively negligible pipeline delay)
- General abstraction makes it possible to extend other known SRAM data structures (e.g. Bloom filters)