A Randomized Interleaved DRAM Architecture for the Maintenance of Exact Statistics Counters[*]

Bill Lin[1], Jun (Jim) Xu[2], Nan Hua[2], Hao Wang[1], Haiquan (Chuck) Zhao[2]
[1]University of California, San Diego, [2]Georgia Institute of Technology
{billlin,wanghao}@ucsd.edu, {jx,nanhua,chz}@cc.gatech.edu

ABSTRACT

We extend a previously proposed randomized interleaved DRAM architecture [1] that can maintain wirespeed updates (say 40 Gb/s) to a large array (say millions) of counters. It works by interleaving updates to randomly distributed counters across multiple memory banks. Though unlikely, an adversary can conceivably overload a memory bank by triggering frequent updates to the same counter. In this work, we show this "attack" can be mitigated through caching pending updates, which can catch repeated updates to the same counter within a sliding time window. While this architecture of combining randomization with caching is simple and straightforward, the primary contribution of this work is to rigorously prove that it can handle with overwhelming probability all adversarial update patterns, using a combination of tail bound techniques, convex ordering theory, and queueing analysis.

Categories and Subject Descriptors: C.2.3
General Terms: Algorithms, Measurement, Theory, Performance.
Keywords: Convex ordering, tail bound, statistics counters.

1. INTRODUCTION

Fig. 1(a) depicts our basic randomized counter architecture [1]. Given a SRAM/DRAM random access latency ratio of μ (e.g. μ = 4ns/40ns = 1/10), we use $B > 1/\mu$ memory banks to store the counters. The basic idea is to randomly distribute the counters evenly across the B memory banks so that with high probability each memory bank will receive about one out of B counter updates on average. This is achieved by applying a pseudorandom permutation function $\pi : \{1, \ldots, N\} \rightarrow \{1, \ldots, N\}$ to a counter index to obtain a permuted index, which corresponds to a memory location in the k^{th} memory bank, where $k = \pi(i) \mod B$, at address location $a = \lfloor \pi(i)/B \rfloor$. For each memory bank, we maintain a small update request queue Q_k. To update counter c_i that is stored in the k^{th} memory bank, an update request gets inserted into Q_k. These request queues are then serviced in an *interleaving* order. Among the salient features of this architecture is its ability to support arbitrary increments/decrements and different number representations (including floating point), which are in general not supported by any other existing statistics counter architectures.

With the proposed index permutation scheme, it is difficult for an adversary to purposely trigger a large number of consecutive updates to the same memory bank with updates to distinct counters since the pseudorandom permutation function used is *not* known to the outside world. However, though unlikely, an adversary can conceivably overload a memory bank by triggering frequent updates to the same counter. Under such adversarial conditions, the DRAM bank in which this counter resides will be overloaded.

To safeguard our scheme against such adversarial situations, we introduce a *fully associative cache* (of size C) in front of our interleaved memory architecture, as shown in Fig. 1(b). The purpose of this cache is to catch repeated updates to the same counter within a sliding window of C cycles as to avoid triggering new memory operations; This sliding window mechanism can be implemented using a FIFO replacement policy[1] for the fully associative cache. That is, if a new counter update arrives for counter c_i, we can lookup the cache to see if there is already a pending update request to this counter. If there is, then we can just simply modify that request rather than triggering a new one (e.g. change from "+1" to "+2"). Since these two updates will result in only one (instead of two) eventual DRAM updates, the arrival processes to the update request queues should still behave like or be dominated by a Geom/D/1 queue within a sliding window of C cycles. We present in the next section a theoretical analysis for this modified architecture under adversarial conditions.

2. ANALYSIS

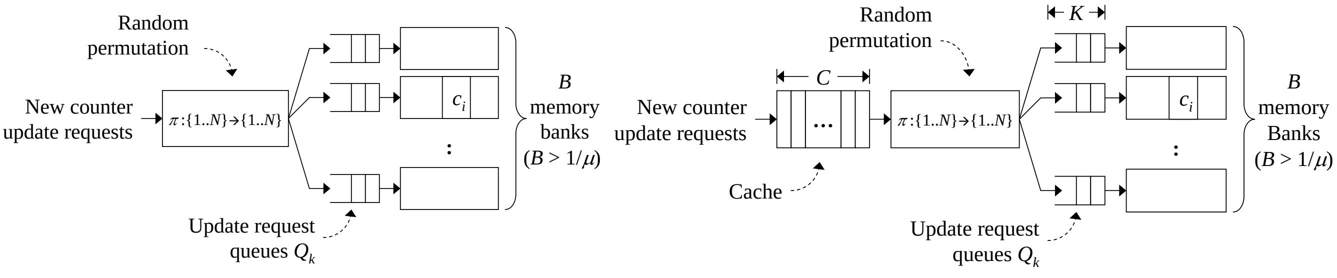
In this section, we prove the aforementioned main theoretical result of this paper, that is, when $B > 1\mu$ (i.e., the system is not overloaded), a small amount of cache combined with index randomization will withstand all counter update sequences, including the adversarial ones, with overwhelming probability. The main idea of our proof is as follows. Given any arbitrary update sequence over a time period $[s, t]$ (viewed as a parameter setting), we are able to obtain a tight stochastic bound of the number on arrivals of counter update requests to a DRAM bank during $[s, t]$ using various tail bound techniques. Since our scheme has to work with all possible update sequences, our bound clearly has to be the worst case (i.e. the maximum) stochastic bound over all of them. However, the space of all such sequences is too large to enumerate. Fortunately, we discovered that the aforementioned number of arrivals under all these update sequences are dominated under a particular (i.e., worst-case) sequence in the convex order. Since $e^{\theta x}$ is a convex function, we are able to dominate the moment generating functions (MGFs) of the number of arrivals under all other update sequences by that under the worst-case update sequence. The final tail bound is obtained by simply applying the Chernoff bound to this worst-case MGF. To the best of our knowledge, we are the first to combine convex ordering theory with the Chernoff bound to obtain a worst-case large deviation theory result.

2.1 Union bound – the first step

In this section, we bound the probability of overflowing a request queue Q. Let $D_{0,n}$ be the event that one or more requests are dropped because Q is full during during time interval $[0, n]$ (in units of cycles). K is the size of each request queue, B is the number of DRAM banks, μ is the SRAM/DRAM random access

[*]This work is supported in part by NSF grants CNS-0519745, CNS-0626979, CNS-0716423, CAREER Award ANI-0238315, a gift from Cisco, and the UCSD Center for Networked Systems.

[1] We choose FIFO for two reasons: (1) it is not hard to prove that under the aforementioned adversarial conditions, FIFO will perform no worse than any other fancy policy such as LRU; (2) no other policy is as simple to analyze as FIFO although the math with FIFO has already been incredibly messy.



Random permutation
New counter update requests
$\pi:\{1..N\}\to\{1..N\}$
c_i
B memory banks ($B > 1/\mu$)
Update request queues Q_k
K
C
Cache
Random permutation
New counter update requests
$\pi:\{1..N\}\to\{1..N\}$
c_i
B memory Banks ($B > 1/\mu$)
Update request queues Q_k

(a) Basic architecture.

(b) Extended architecture to handle adversaries.

Figure 1: Memory architecture for a randomized DRAM-based counter scheme.

latency ratio, C is the size of the cache. In the following, we shall fix n and will therefore shorten $\tilde{D}_{0,n}$ to $\tilde{D}$. Note that $Pr[\tilde{D}]$ is the overflow probability for just one out of B such queues. The overall overflow probability can be bounded by $B \times Pr[\tilde{D}]$ (union bound).

$$\Pr[\tilde{D}] \leq \sum_{0\leq s\leq t\leq n} \Pr[D_{s,t}] \qquad (1)$$

Here, $D_{s,t}$, $0 \leq s < t \leq n$, represents the event that the number of arrivals during the time interval $[s,t]$ is larger than the maximum possible number of departures in the queue, by more than K. Formally, letting $X_{s,t}$ denote the number of update requests (to the DRAM bank) generated during time interval $[s,t)$, we have

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

The inequality (1) is a direct consequence of the following lemma, which states that if the event $\tilde{D}$ happens, at least one of the events $\{D_{s,t}\}_{0\leq s<t\leq n}$ must happen.

LEMMA 1. $\tilde{D} = \bigcup_{0\leq s\leq t\leq n} D_{s,t}$

2.2 Bounding individual $\Pr[D_{s,t}]$

In the following, we first obtain sharp tail bounds from the MGF of the random variable X through the standard Chernoff method.

$$\begin{aligned}\Pr[D_{s,t}] &= \Pr[X > K + \mu\tau] = \Pr[e^{X\theta} > e^{(K+\mu\tau)\theta}] \\ &\leq \min_{\theta>0} \frac{E[e^{X\theta}]}{e^{(K+\mu\tau)\theta}} \qquad (2)\end{aligned}$$

where τ is $t-s$, and the last step is due to the Markov inequality.

Then, we aim to bound the moment generating function $E[e^{X\theta}]$ by finding the worst-case sequence, in the *convex ordering* sense (defined next). Note that we resort to convex ordering, because *stochastic order*, which is the conventional technique to establish ordering between random variables and is stronger than *convex order*, does not hold here.

DEFINITION 1 (CONVEX ORDER [2](DEFINITION 1.5.1)). *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.*

Now we will construct the worst-case pattern in terms of convex ordering. Suppose ℓ distinct memory addresses are requested during time interval $[s,t]$, and the count of their appearances are $m_1, ..., m_\ell$, with $\sum_{i=1}^{\ell} m_i = \tau$. We have $X = \sum_{i=1}^{\ell} m_i X_i$, where X_i are i.i.d Bernoulli variables[2] with probability $1/B$. Hence any sequence update sequence could be summarized as a splitting $\{m_1, ..., m_\ell\}$ under the constraint $\sum_{i=1}^{\ell} m_i = \tau$. Another constraint is that none of the counts $m_1, ..., m_\ell$ can exceed T, where $T = \lceil\frac{\tau}{C+1}\rceil$, since the dequeued requests to the same DRAM address should not repeat within any sliding window of $C+1$ cycles, where C is the cache size.

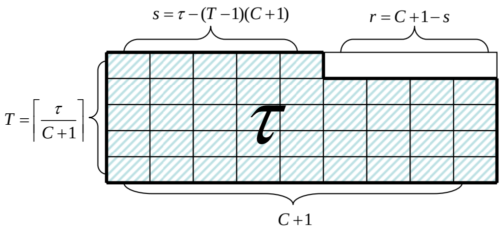


Figure 2: Relationship among s, r, T and τ

A worst-case counter update sequence (in the convex ordering sense) can be constructed as follows: first comes $s + r (= C+1)$ distinct requests $a_1, a_2, ..., a_{s+r}$, then repeats for $T-1$ times in total, where $s = \tau - (T-1)(C+1)$ and $r = (C+1) - s$. The last s requests are $a_1, a_2, ..., a_s$. Let m^* be the splitting pattern of this update sequence, i.e. s requests with count T and r requests with count $T-1$. Let M be the set of all valid splitting patterns. The following theorem and corollary can be proven:

THEOREM 1. *m^* is the worst case splitting pattern in terms of convex ordering, i.e. $X_m \leq_{cx} X_{m^*}, \forall m \in M$.*

COROLLARY 1.

$$E[e^{X\theta}] \leq (\frac{1}{B}e^{T\theta} + (1-\frac{1}{B}))^s(\frac{1}{B}e^{(T-1)\theta} + (1-\frac{1}{B}))^r$$

Combine it with (2), we obtain the following bound for $\Pr[D_\tau]$:

$$\Pr[D_\tau] \leq \min_{\theta>0} \frac{(\frac{1}{B}e^{T\theta} + (1-\frac{1}{B}))^s(\frac{1}{B}e^{(T-1)\theta} + (1-\frac{1}{B}))^r}{e^{(K+\mu\tau)\theta}}$$

3. REFERENCES

[1] B. Lin and J. Xu, “DRAM is Plenty Fast for Wirespeed Statistics Counting,” ACM HotMetrics’08, June 2008.

[2] A. Muller, D. Stoyan. “Comparison Methods for Stochastic Models and Risks”. Wiley, 2002.

[2] We acknowledge that this i.i.d Bernoulli modeling of $X_i's$ is an approximation because this modeling is exact only when the counter indices go through a pseudorandom function while in our case they go through a pseudorandom permutation. However this is a very close approximation when the total size of the DRAM is large, which holds in our case. We have not had any other approximation throughout the paper.