Rethinking Design Metrics for Datacenter DRAM

Manu Awasthi
Samsung Semiconductor, Inc.
601 McCarthy Blvd., Milpitas, CA
manu.awasthi@ssi.samsung.com

ABSTRACT
Over the years, the evolution of DRAM has provided a little improvement in access latencies, but has been optimized to deliver greater peak bandwidths from the devices. The combined bandwidth in a contemporary multi-socket server system runs into hundreds of GB/s. However datacenter scale applications running on server platforms care largely about having access to a large pool of low-latency main memory (DRAM), and in the best case, are unable to utilize even a small fraction of the total memory bandwidth. In this extended abstract, we use measured data from the state-of-the-art servers running memory intensive datacenter workloads like Memcached to argue for main memory design to steer away from optimizing traditional metrics for DRAM design like peak bandwidth so as to be able to cater the growing needs to the datacenter server industry for high density, low latency memory with moderate bandwidth requirements.

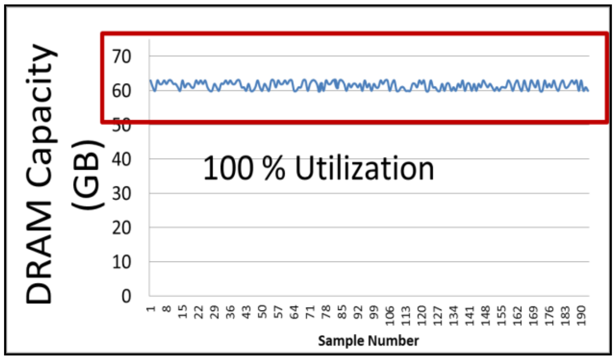
Keywords
DRAM, memory, performance metrics, datacenters.

1. INTRODUCTION
A large fraction of user-facing, web based services that are in use today, rely on backend datacenters. These datacenters host a multitude of user-facing applications, with each of these applications servicing millions of users, concurrently. In order to support the latency and throughput requirements of these applications, the application relies on a number of helper applications and services. For example, an application might be based primarily on servicing users by querying a NoSQL database, but in order to effectively service *all* users within reasonable time, the application will require the support of web serving, load balancing and data caching services, among others. In order to display targeted products or advertisements to each user, the services of a (probably Hadoop based) data analytics engine [3] might also be required. Hence, in order to effectively run one full-fledged user facing application, services of four or five distinct ``supporting'' applications might be required.

MEMSYS '15, October 05-08, 2015, Washington DC, DC, USA
ACM 978-1-4503-3604-8/15/10.
http://dx.doi.org/10.1145/2818950.2818973

Each one these applications and services has a distinct set of characteristics which requires the server hardware and software design to be optimized for the particular use case under consideration. Hence, in order to support the varying needs of these complex, distributed applications, the datacenter is divided into multiple *tiers* of servers – each tier configured specifically for a class of applications or services. Each of these services has distinct requirements in terms of hardware and software support. For example, a typical datacenter that supports web 2.0 style applications might have five to six different tiers of servers [3]. A number of studies have characterized applications running on each of these tiers. All studies confirm two theories (i) all datacenter applications are memory and I/O intensive, with very few of them being compute intensive, and (ii) raw compute almost never becomes the bottleneck for any of these applications – the bottlenecks appear elsewhere in the system before compute does.

2. DATACENTER APPLICATIONS
Since datacenter applications service large number of users, one of the important metrics that needs to be optimized, especially from a user's perspective, is *service latency*. An important class of applications that are critical for achieving low service latencies across users, types and sizes of request sizes, are in-memory key-value (KV) stores. Primary examples of such applications are Memcached and REDIS [3]. Architecturally, the datacenter tier running the in-memory KV stores typically comprises of a number of servers that form a distributed, shared-nothing, caching tier between the web-service frontend and the server tier that runs the database (or a data store) service. The main responsibility of this tier is to cache the most recently accessed pieces of data (hot data) in the server's DRAM. In the absence of this caching tier, the end user will experience very high service latencies because all the requests will be redirected to the servers running the data store service. Servicing those requests will involve retrieving data from the server's disk/storage, leading to larger access times. Moreover, increased load on the data servers leads to increased queuing delays, further exacerbating the service latency problem. Hence, as the number of users increases, the role of the in-memory caching tier becomes increasingly important to meet the service level agreements/requirements (SLAs/SLRs) for the services under consideration. Similar to KV stores, applications at other tiers of the datacenter also require access to DRAM, although, after a certain point, the applications are not sensitive to DRAM [3, 4, 5].



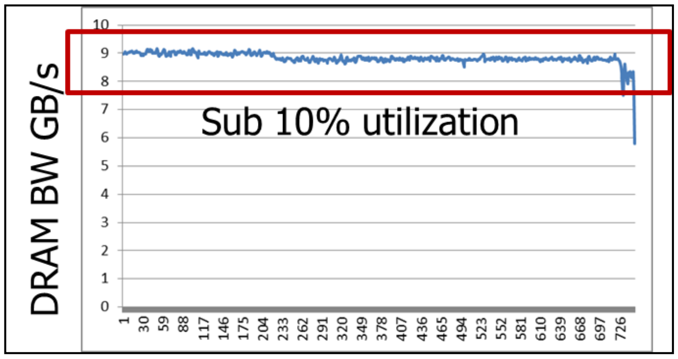


Figure 1. Left DRAM Capacity utilization, Right – DRAM Bandwidth utilization for Memcached, 4 MB value sizes, 99% read traffic

Some of our previous work in this regard [3] characterized the most popular application for each server tier. Of all the applications that were considered, the success of in-memory KV caches depends on availability of a large pool of low latency, high density main memory, the role of which has traditionally been fulfilled by DRAM.

3. THE METRICS OF OPTIMIZATION

The in-memory KV cache requires large quantities of DRAM for successful operation. Using the YCSB framework, we characterized Memcached on state of the art Intel servers for a large number of use cases [3]. The performance was measured using a combination of two metrics – 99th percentile response rates on the client side, as well as cache hit rates on the server side. Two results, as depicted in Figure 1, stood out. Firstly, Memcached performance (per-server cache hit rate) is very sensitive to available DRAM capacity. The application consumes ***all*** DRAM capacity that is allowed to allocate. Secondly, per server (and hence, by association, user level) performance of Memcached is almost independent of available DRAM bandwidth on the server. Our test setup had two sockets with four channels (to DDR3 RDIMMs), capable of delivering 12.8 GB/s per channel. Overall, the system was capable of delivering 102.4 GB/s of peak DRAM bandwidth. In the best case scenario, bandwidth utilization peaked at ~9% of peak. This was observed in cases where large chunks of data (values) were being requested from the cache. In experiments with smaller request sizes, the bandwidth utilization fell even further.

We also experimented with the DRAM capacity and bandwidth requirements of applications at different tiers. Irrespective of the application under consideration and the use case that it was subjected to, the bandwidth utilization never went beyond 35%. However, almost all applications were sensitive to DRAM capacity.

4. DISUSSION AND CONCLUSIONS

Over the years, commodity DRAM (DDR2, DDR3, DDR4 etc.) has been optimized for (i) low latency, (ii) high bandwidth, and (iii) low cost/bit. During the course of time, latter two have taken over the former as first order design constraints. If we track changes in absolute values of these parameters over different generations of DRAM, we find that although peak DRAM bandwidth has increased by over 20x, absolute latencies have only decreased by a few percentage points [6]. The current contenders for future DRAM architectures and standards including Wide I/O, HBM, and HMC have made significant strides in furthering the state of the art in DRAM technology. However, even with the breakthroughs in process technologies and 3D stacking, very little attention has been paid to fundamentally rethink the design metrics that server DRAM should be optimized for, especially given the changing requirements of applications.

In this paper, we argue that for designing datacenter - class main memory/DRAM, we need to refocus our energies on ***metrics that matter*** for the application class under consideration. Using the case of in-memory KV stores as an example, we show that the need of the hour is to stop optimizing the next generation DRAM architectures for delivering higher bandwidth, but rather concentrate our energies on designing high capacity, low latency DRAM, while providing moderate bandwidth provisioning for the system.

5. REFERENCES

[1] Kanev, Set al., Profiling a warehouse scale computer. *In Proceedings of ISCA, 2015*.

[2] Kozyrakis, C., Kansal, A., Sankar, S. and Vaid, K., Server Engineering Insights for Large-Scale Online Services *Micro, IEEE* vol.30, no.4, pp.8,19, July-Aug. 2010.

[3] Awasthi, M., Suri, T., Guz, Z., Shayesteh, A., Ghosh, M., and Balakrishnan, V. System-Level Characterization of Datacenter Applications. *In Proceedings of ICPE*, 2015.

[4] Z. Ren, X. Xu, J. Wan, W. Shi, and M. Zhou. Workload characterization on a production Hadoop cluster: A case study on Taobao. *In Proceedings of IISWC*, 2012.

[5] T. Rabl, et al. Solving big data challenges for enterprise application performance management. *Proc. VLDB Endow*., 5(12):1724–1735, 2012.

[6] Lee, D. and Kim, Y., and Seshadri, V., and Liu, J., and Subramanian, L., and Mutlu, O., Tiered-latency DRAM: A low latency and low cost DRAM architecture. *In Proceedings of HPCA*, 2013