A Control Scheme for Batching DRAM Requests to Improve Power Efficiency

Krishna Kant
George Mason University
Fairfax, VA 22030
kkant@gmu.edu

ABSTRACT

This paper discusses a closed-loop control algorithm to coordinate power management of memory ranks and thereby achieve power savings beyond independent rank power management while bounding the throughput degradation.

Categories and Subject Descriptors: B.3.m
General Terms: Algorithms, Design, Performance
Keywords: DRAM, coordinated power management

1. INTRODUCTION

Because of the increasing contribution of memory power on overall system power, intelligent management of power for memory ranks is becoming critical. In this paper, we consider coordinated power management of multiple ranks that could span across a single DIMM, across a single or all memory channels, or even across all sockets.

The technique proposed in this paper is a closed loop control that batches the traffic adaptively to enhance power efficiency while ensuring that the throughput degradation due to power management stays below some desired bound. The basic idea is to improve energy efficiency by batching of requests [1] and the batching is achieved by periodically making a rank inactive so that newly arriving requests queue up but are not scheduled. When such a rank runs out of ongoing requests, it is placed in a low power mode until the inactive period expires. Such a scheme helps save power beyond what power management of an individual rank could deliver. The additional latency introduced does degrade the performance, and the closed loop control is designed to limit the degradation to a specified target value. The scheme is somewhat similar to the one in [2] where the additional latency due to power measurement is measured and used in closed loop control.

2. COORDINATED INSTANCE CONTROL

Figure 1 illustrates the coordinated control of ranks pictorially. There are N ranks each having its own request queue. Each rank itself may involve multiple servers (or "banks") as shown in the figure by small circles. Normally, each arriving request will begin service as soon as a suitable server becomes available. In the proposed scheme, however, only certain ranks are considered "eligible" and able to schedule new requests. The eligible subset remains so for a certain *gating period*, denoted G, after which another subset is made eligible. In our algorithm, G is not a constant, but a parameter that is adjusted dynamically in order to keep the throughput degradation within an acceptable bound ϵ_t.

A rank may have nonzero ongoing requests when it is marked as ineligible. All ongoing requests are allowed to finish normally on an ineligible rank but no new requests are scheduled. When all the servers become idle, the resource rank is placed in a low power state even though there may be some waiting requests. This works fine for DRAM since the request management is done by the memory controller rather than the DRAM.

The memory access behavior of a workload and hence the impact of memory path latency on performance is generally quite complex, therefore, we use an online monitoring approach to estimate and react to the throughput degradation. The algorithm is designed specifically to allow inexpensive HW implementation and suitable parameter choice can even avoid multiplications/divisions.

For online monitoring we divide time into successive windows of size $W_c + W_u$, where W_c denotes the period during which gating control is effected and W_u the period during which it is not. We call W_u as the *probe period* during which we probe for the full (or undegraded) throughput λ_c. Note that the probe period needs to be long enough for the system to recover from any throughput restriction and yet be small enough so that the control is in effect most of the time.

During the W_c duration, only K (out of N) ranks are made eligible, whereas during the W_u all ranks are eligible. During W_c periods in successive cycles, the set of eligible ranks are changed systematically to ensure fairness. When a gating period is about to end, the new rank is chosen as the one with the longest queue. This works quite well, even though it is suboptimal.

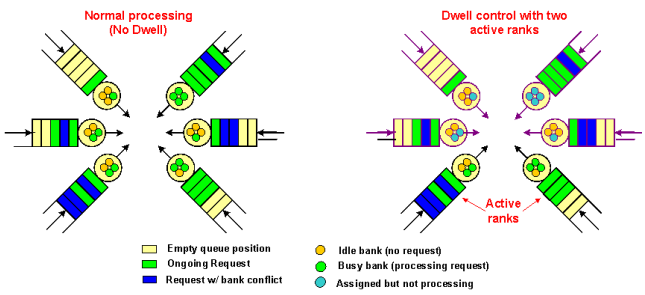

Figure 1: Illustration of coordinated rank control

SIGMETRICS'11, June 7–11, 2011, San Jose, California, USA.
ACM 978-1-4503-0262-3/11/06.

The gating period G is estimated based on the observed throughput degradation, denoted ϵ_o. Let N_c and N_u denote the total number of memory transactions completed during the windows W_c and W_u respectively. Then the degraded throughput is $\lambda_c = N_c/W_c$, and the unperturbed throughput is $\lambda_u = N_u/W_u$. Therefore, $\epsilon_o = (1 - \lambda_c/\lambda_u)$. In order to reduce jitter in measured throughput, it is useful to exponentially smooth N_c and N_u values over successive cycles. Let ϵ_t denote the maximum tolerable throughput degradation. The value of G is adjusted based on the "error signal" $\epsilon_o - \epsilon_t$. The control mechanism attempts to correct large errors quickly while avoiding ping-pongs.

3. EXPERIMENTAL RESULTS

Since latency sensitivity is a key factor in determining the power savings potential of power management, we created several workloads with different latency sensitivities (dialed via the transaction buffer size distribution as described above). Based on some experimentation, we chose a transaction buffer size of 10 for "low" latency sensitivity (LS) workloads and 4 for workloads with "normal" latency sensitivity (NS). We show the results for the LS and NS cases in the following. All experimental results were obtained via a detailed (but not cycle accurate) simulator that represents DDR3 memory operation in substantial detail but the CPU simply. In particular, the CPU stall behavior (and hence memory access latency sensitivity) is controlled by properly sizing per hardware thread transaction buffers.

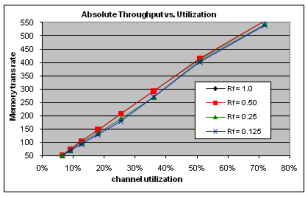

Figure 2: Throughput for LS case

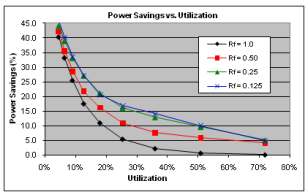

Figure 3: Power savings for LS case

For our experimental setup, we control all 8 ranks of a DDR3 memory in each socket. We define R_f as the fraction of ranks that are kept eligible at a time. We show behavior for R_f =1/8, 1/4, 1/2 and 1. $R_f = 1/8$ means that only one rank in a socket is eligible for scheduling at any given time, and represents the most stringent coordinated control. On the other extreme, $R_f = 1$ corresponds to no coordinated control, and hence reduces to independent control of each rank.

Let us start with the low latency sensitivity case. Fig 2 and 3 show the throughput and relative power savings as a function of channel utilization with R_f as a parameter. The throughput graph is included merely to confirm that the throughput degradation remains roughly contained within the 1% bound specified in the algorithm. The power saving is given as a percentage of total power consumption (at that utilization) without any power management.

The curve for $R_f = 1$ clearly shows significant power savings achievable below about 20% channel utilization range by the fine-grain power management considered in this paper. In contrast, coarse-grain techniques such as page remapping would typically apply only when utilization dips under a few percent or less. *Thus there are significant gains to be had from the fine-grain power management.*

The curves for $R_f < 1$ show that rank coordination further improves power savings *beyond* those achievable by isolated power management at moderate to high utilization levels. (At low utilizations, isolated management itself is adequate.) For example, at 25% channel utilization, $R_f = 1/2$ gives 11% power savings whereas $R_f = 1$ (i.e., uncoordinated control) gives only 5% savings. With $R_f = 1/4$, the power savings go up to 16%. At 50% channel utilization where isolated management has essentially no power savings, with $R_f = 1/2$ and $R_f = 1/4$ we get respectively 5.8% and 9.2% savings. Clearly, these are significant numbers in watts at such high channel utilizations.

We also note that $R_f = 1/8$ and $R_f = 1/4$ behave almost identically. This is because even with $R_f = 1/4$, much of the available power savings have already been squeezed out, and reducing R_f only serves to increase the read latency without any throughput advantage.

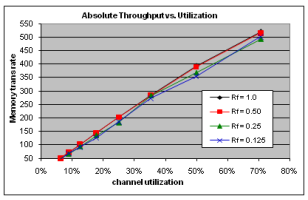

Figure 4: Throughput for MS case

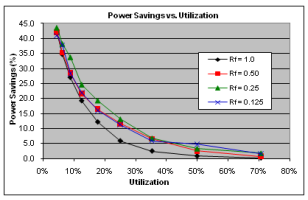

Figure 5: Power savings for MS case

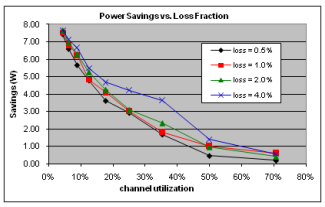

Figure 6: Power savings vs. Loss Frac.

Now we consider similar plots for the medium latency sensitivity case. Fig 4 and 5 show the throughput and power savings as a function of channel utilization with R_f as a parameter. It is seen that $R_f = 1/2$ in this case is enough to squeeze out all the additional power savings over isolated control, and reducing R_f further is not helpful. In fact, $R_f = 1/4$ or smaller degrades the throughput below the specified percentage. The reason for this failure is that the window W_u is inadequate to provide adequate recovery of the throughput.

Fig 6 shows the power savings curves for the medium latency sensitivity case under 4 different target degradation levels. For these curves we used $R_f = 1/2$ to ensure that there is no significant over-control and the target degradation can indeed be maintained. Not surprisingly, as the target degradation increases, more power savings become possible.

4. REFERENCES

[1] A. Papathanasiou and M. Scott, "Energy Efficiency through Burstiness", Proc of the 5th IEEE Workshop on Mobile Computing Systems and Applications (WMCSA'03), pp. 44-53, Oct 2003.

[2] X. Li, Z. Li, F. David, et al., "Performance directed energy management for main memory and disks", Proc. of 11th ASPLOS conf, pp271-283, 2004.