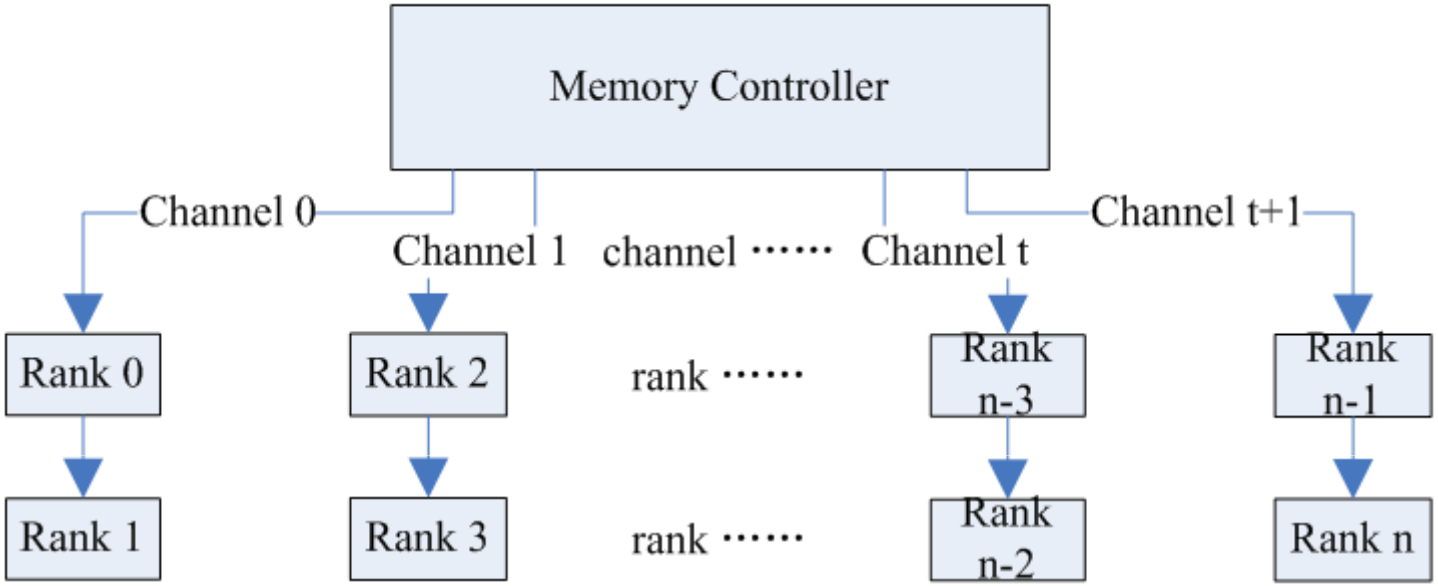
中国科学技术大学
University of Science and Technology of China

A Scheduling of Periodically Active Rank of DRAM to Optimize Power Efficiency

Gangyong Jia
University of Science and Technology of China

First Workshop on Highly-Reliable Power-Efficient Embedded Designs
February 2013

DRAM Organization



- Relationship among channels, DIMMs, ranks, banks
  - A memory system can contain multiple channels
  - each channel is associated with 1 or 2 DIMMs
- A rank is the smallest physical unit for power management
- Banks can be accessed parallel

Optimize Power Efficiency

- Goal: optimize power efficiency for multi-core DRAM
- Study: propose a periodically active rank scheduling (PARS)
  - partition all threads in the system into groups
  - modify page allocation policy to achieve threads in the same group occupies the same rank but different bank of DRAM
  - sequentially schedule threads in one group after another while only active running group's ranks to retain other ranks low power status

Group Partition

Algorithm 1: group partition

After creating a new thread *T*, *T* ∈ *A*, *A* is an application, *G* is the group of all kernel threads

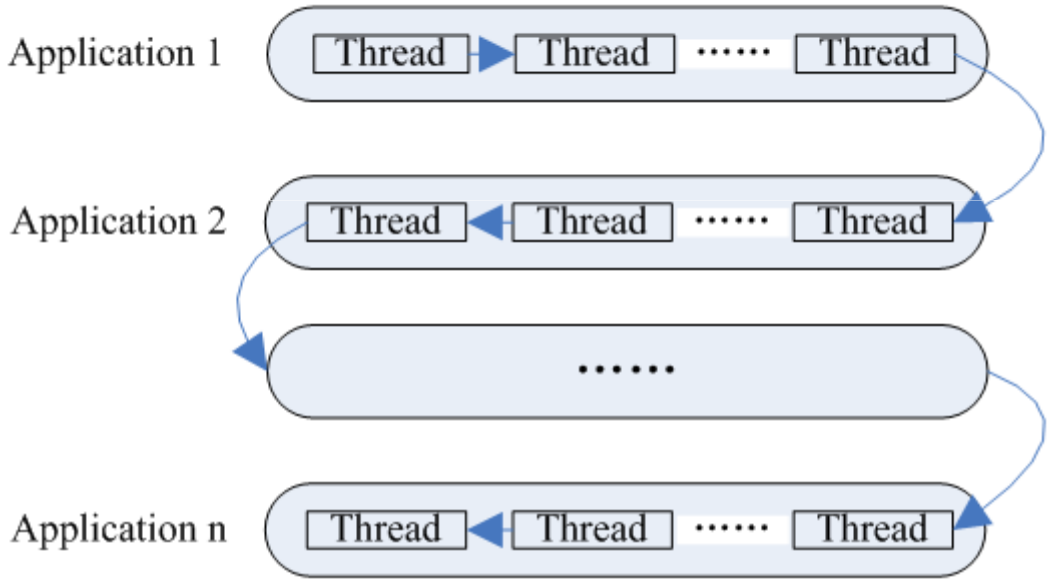
begin

1: whether the application *A* is already existing in system
2: **if** *T* is kernel thread **then**
3: insert *T* into the back of group *G*;
4: return;
5: **else if** *A* is already existing **then**
6: find group G_1, $A \in G_1$;
7: insert *T* into the back of *A*;
8: return;
9: **else** *A* is a new one **then**
10: find the group of lightest load, G_2;
11: insert *A* into the back of group G_2;
12: insert *T* into the back of *A*;
13: *A* is partitioned into group G_2;
14: return;

End

Group Partition

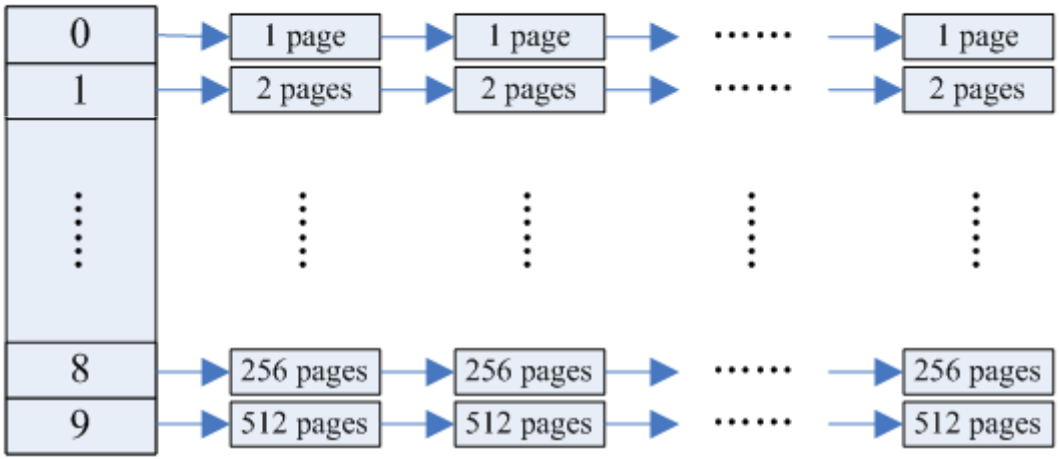
all threads of the same application are listed sequentially,
all applications in the same group are listed sequentially



example of one group list

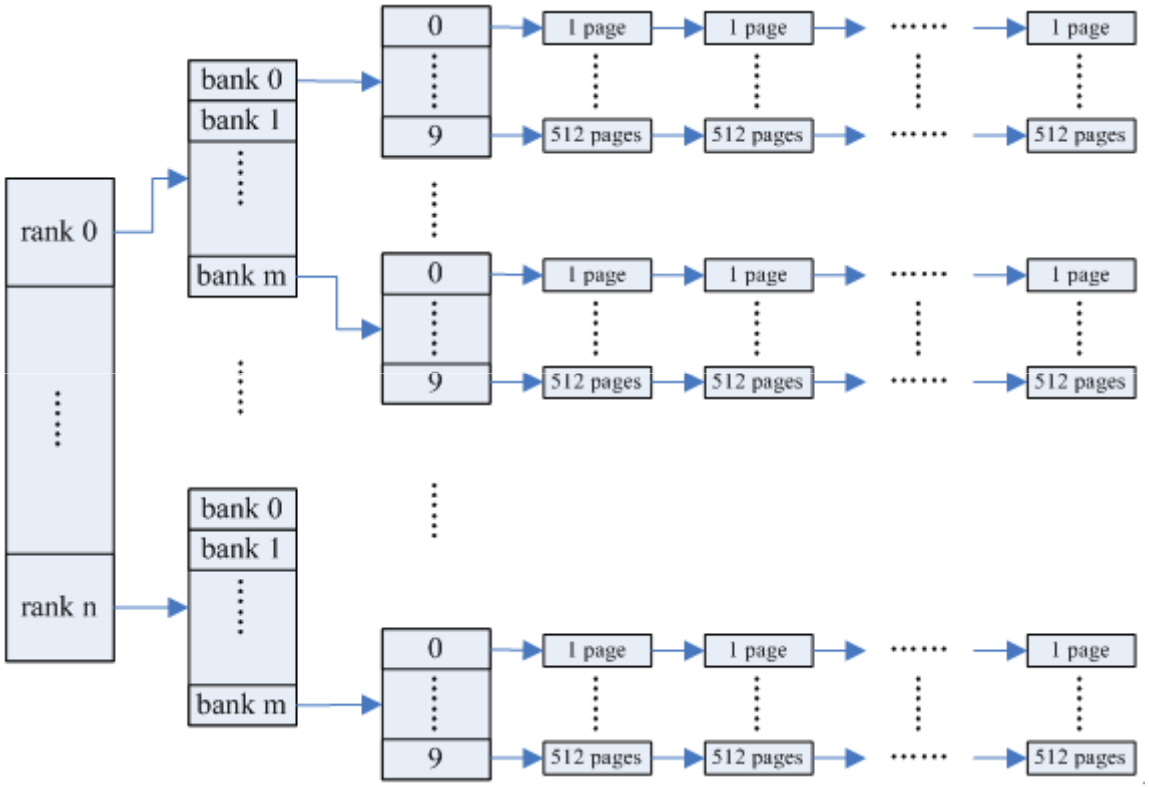
Page Allocation

In the buddy system, the continuous 2^{order} pages (called a block) are organized in the free list with the corresponding order, which ranges from 0 to a specific upper limit.



physical pages management of buddy system

Page Allocation



physical pages management of our system

Page Allocation

Algorithm 2: page allocation

Thread *T* accesses an unmapped virtual address, OS kernel allocates pages

begin

1: find the group *G* which $T \epsilon G$;

2: according to the id of *G*, find corresponding rank, *R*;

3: calculate $Bi = Tid \% B$, *Tid* is the thread id of *T*;

4: identify the right order free list of *Bi* in *R*;

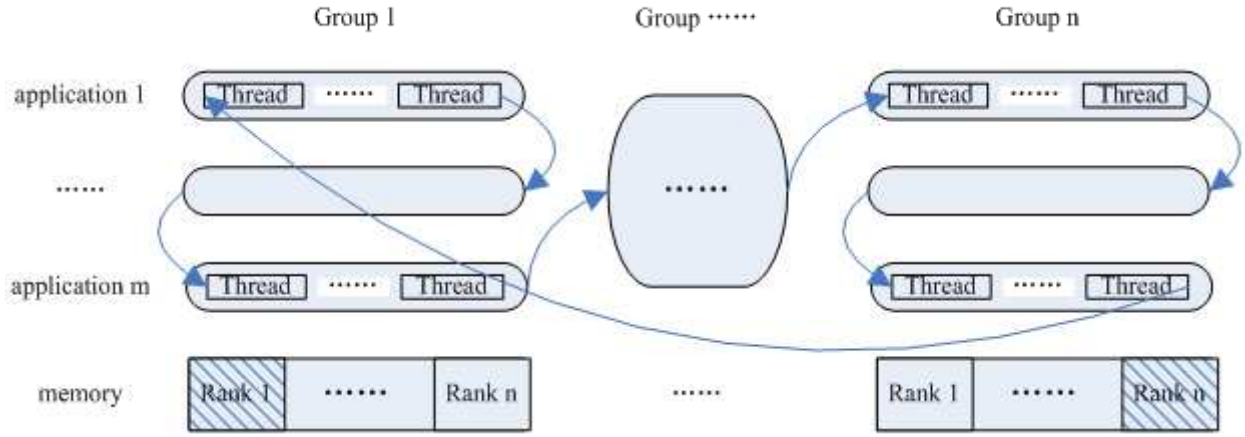
5: allocate on block for *T*;

6: return;

End

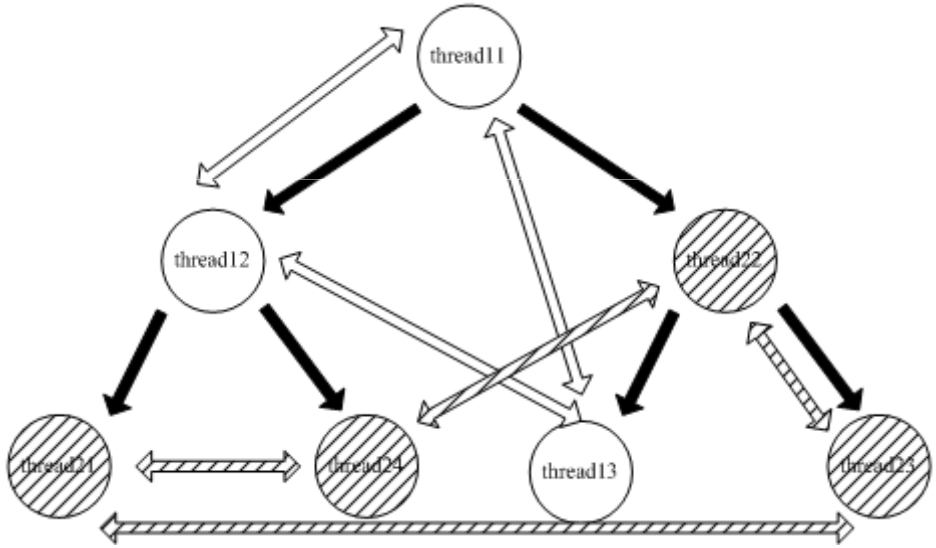
Scheduling and Rank Management Policy

All threads in the system are partitioned into groups, each group occupies only one rank. When a group threads running, only corresponding rank needs to be active, others can be low power. So we coordinate group scheduling with memory rank status management to optimize memory power efficiency.



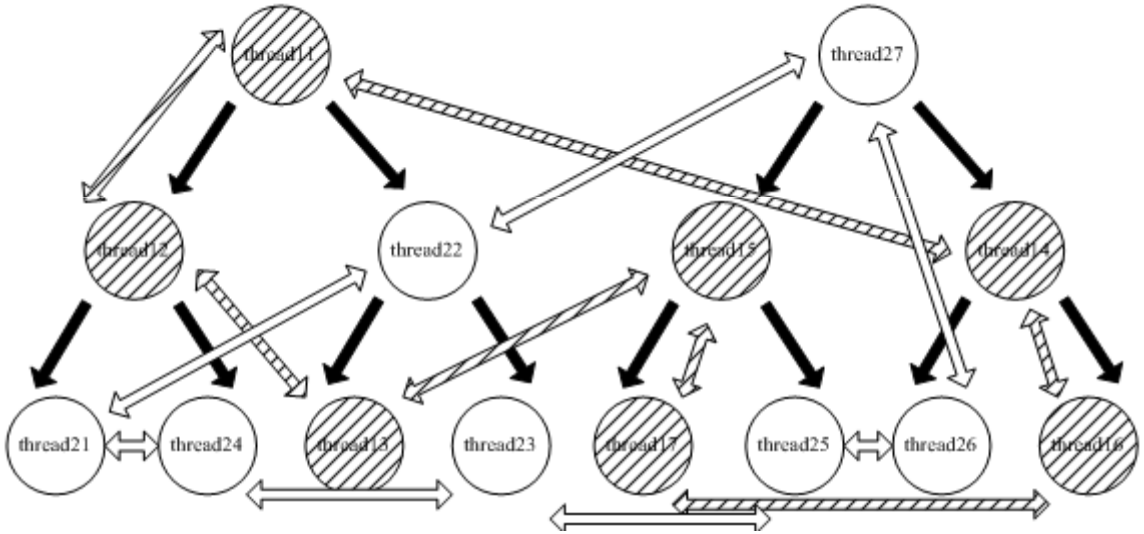
process of threads scheduling and corresponding active rank

All threads in the same thread group are listed together on one core



implement the PARS

All threads in the same thread group are listed together among cores



Experimental Results

- We define the following formula:
  - *Degree of aggregation = total request / switch times*
  - Total request represents the total numbers of request memory
  - switch times represents times of switching between accessing two different ranks

degree of aggregation

- Example
  - we sequentially list a rank numbers of accessing memory, 1, 1, 2, 3, 5, 3, 3, 4.
  - Total request is 8, switch times is 5. The 5 times contains the second 1 to 2; 2 to 3; 3 to 5; 5to 3; third 3 to 4.
  - Degree of aggregation is 8/5.

degree of aggregation

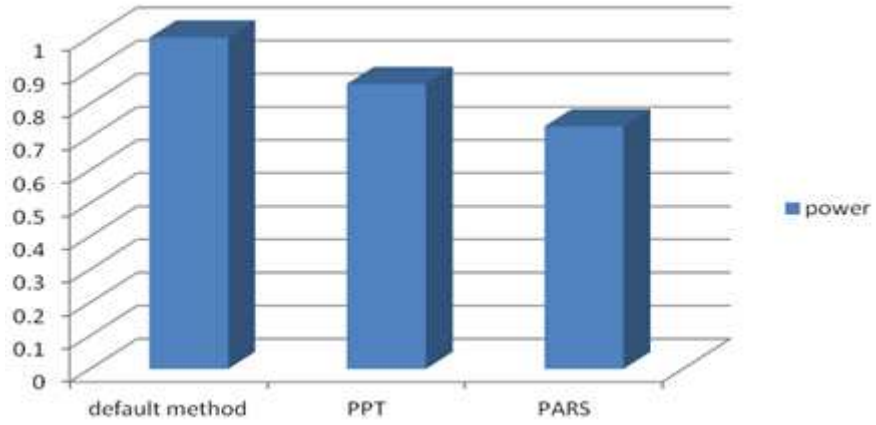
From the below table, we can see PARS is much better than other two methods. From the memory request list, we can obviously find PARS prolong more time accessing on one rank, and reduce much more switch times than other two methods.

compare the degree of aggregation

	Default method	PPT	PARS
Degree of aggregation	10.6	27.1	35.4

power reducing of the PARS

PARS periodically activates one of the ranks according running group, but each time only one memory rank is active except apply a big continuous block which outspace one rank can supply.
Also, our PARS prolongs much more time accessing one the same rank which reduces frequency of switching between ranks.

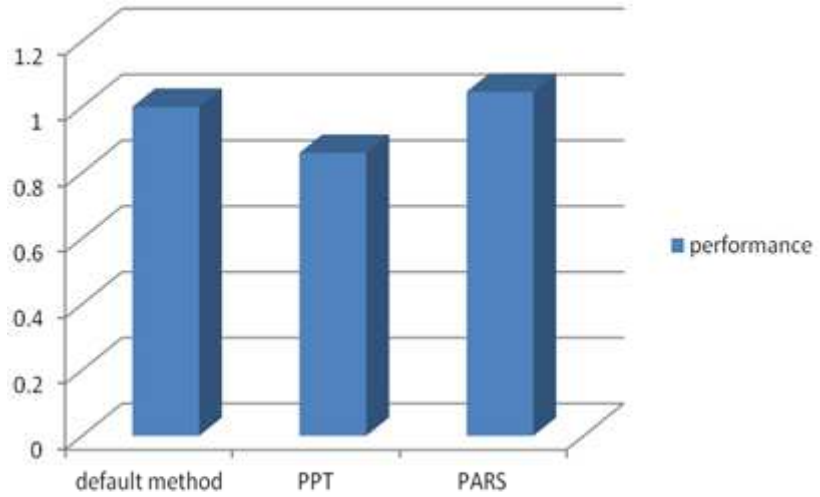


power consumption comparing

performance of PARS

- our PARS optimizes performance in the following parts:
  - partition threads of an application into the same group and priority schedule threads belonging to the same application. The cost of switching between the same application threads is much smaller for sharing the memory address space
  - allocate pages of different banks for threads in the same group. Memory request from different cores almost access different banks, so seldom interfering among threads from different cores and improve parallel

performance of PARS



performance comparing

overhead comparing

	PPT	PARS
L2 cache miss rate	0.094%	0.013%
DTLB misses	26992646	26948934
ITLB misses	17895	12312
ITBL flushes	66	43

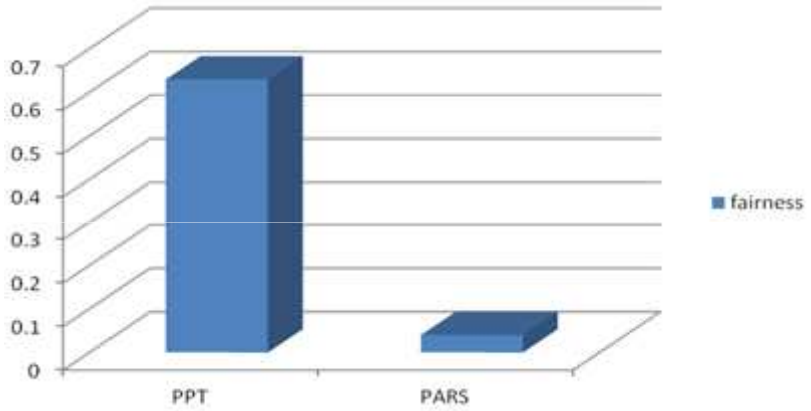
performance of PARS

our page allocation according bank is also very effective, which intensely reduces row buffer miss rate

row buffer miss rate comparing

	Default method	PPT	PARS
Row buffer miss rate	58.2%	60.7%	31.3%

fairness of PARS



fairness comparing

thermal of PARS

average peak temperature comparing

	Default method	PPT	MAS
Peak temperature	85.9	76.1	75.8

contributions

- We firstly coordinate page allocation policy with operating system scheduler to optimize memory power efficiency
- We improve both power efficiency and performance for multi-core
- We propose degree of aggregation parameter to indicate the effect of page allocation policy to retain other memory ranks stay low power as long as possible

Thank You!