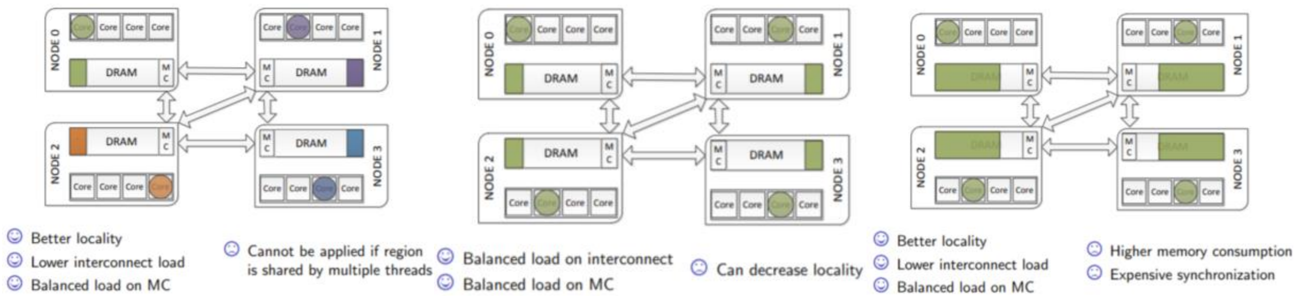
Traffic management: a holistic approach to memory placement on NUMA systems, ASPLOS 2013

Traditionally, in NUMA system, accessing data in a remote node was considered as main overhead in utilizing system. So, the operating systems designed for optimizing access locality. This kind of approach makes forked working threads gathered in same NUMA nodes.

However, as modern NUMA hardware has much smaller remote wire delays, the cost of remote access get lower. This paper issue that memory control and interconnection congestion can be the main overhead

So the paper suggest Carrefour, memory placement algorithm for handling traffic congestion in NUMA system. Carreafour works based on three mechanisms

Page co-location Page interleaving Page replication



NODE 0
NODE 1
NODE 2
NODE 3
Core
DRAM
MC
☺ Better locality
☺ Lower interconnect load
☺ Balanced load on MC
☹ Cannot be applied if region is shared by multiple threads
☺ Balanced load on interconnect
☺ Balanced load on MC
☹ Can decrease locality
☺ Better locality
☺ Lower interconnect load
☺ Balanced load on MC
☹ Higher memory consumption
☹ Expensive synchronization

The allocation algorithm divided into two Global Decision and Page Local Decision

Global Decision phase

Decide whether to enable Carreafour or not and each of three mechanism based on profiled information.

Page Local Decision Phase

Based on decision made on global phase, do actual page allocation

Pros

The main contribution of this paper I think is suggesting that the remote latency overhead, which most studied part, may not be a main bottleneck. It provides a new perspective that memory control and interconnect congestion can be a major overhead.

Cons

Profiling of Carrefour incurs CPU and memory overhead. Do not suggest clear solution for the shortage. And it is better to disable Carrefour in many cases (ex. write heavy workloads)

Idea

Need to suggest the way to mitigate the overhead of profiling.

Regularities considered harmful: forcing randomness to memory accesses to reduce row buffer conflicts for multi-core, multi-bank systems, ASPLOS 2013

Suggest new memory allocation algorithm, for exploit row buffer hit (mitigate row buffer conflict) and memory parallelism.

M3 Design

• Support **parallelism** by using many banks evenly as possible

➔ Memory container

Divides the entire memory into multiple memory containers, with a memory container being exclusively assigned to a core.

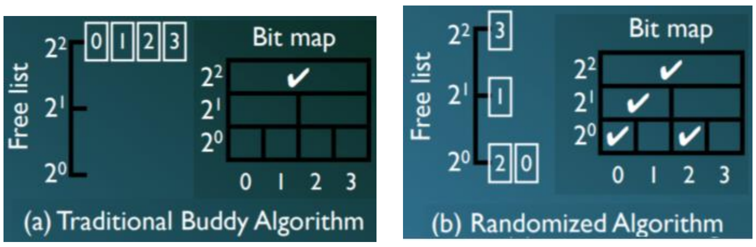
• Reduce **row-buffer conflict** by randomizing page access pattern

➔ Randomizing algorithm

Individual page frame management for randomness

Downward search to disperse allocation sequence

+ Versus Buddy algorithm



Pros

Exploit memory parallelism and row buffer hit

Cons

the traditional buddy algorithm was for preventing memory fragmentation. But the paper does not handle the fragmentation problem

Idea

Should suggest the way to handle memory fragmentation, which traditionally handled on buddy algorithm