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

   This paper presents Carrefour, which is a memory placement algorithm for data-intensive applications, which aims at eliminating the congestion on memory controllers and interconnects. Although NUMA hardware in the old days has been optimized with the priority of locality, congestion on memory controllers and interconnects, caused by memory traffic from data-intensive applications, has become a more important factor for optimization than locality today.

   It was good that we demonstrated this through a variety of workloads. It is meaningful because it is the first paper that said balancing is more important than locality. The technique of balancing memory pressure and increasing locality by combining three algorithms (Page relocation, Page replication, Page interleaving) was also good.

   However, it is a weak point in that we do not know if it can be used in NUMA architectures not AMD in that this paper collected global and per application metrics only using hardware counters. Another weakness is that there was no consideration for task migration.

   So, it would have been better if the results of the experiment included not only memory page migrations and replications, but also task migrations.

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

   Point out that the row-buffer miss that occurs when accessing rows that are not in the row-buffer of the memory bank has 2~5 times higher overhead than the row-buffer hit. This row-buffer miss occurs when multiple threads access the same memory bank at the same time (sequential access), which is called row-buffer conflict. So, this paper proposes a novel kernel-level memory allocator, called M3 (M-cube, Multi-core Multi-bank Memory allocator) to solve the row-buffer conflict problem.

   It has two design consideration. First, it reduces row-buffer conflict by randomizing page access pattern. It can drop the probability of accessing same bank to (1/Total_number_of_banks). Second, support parallelism by using as many banks as possible and using the banks as evenly as possible.

   It was good that this paper propose novel randomizing algorithm to reduce row-buffer conflict, and introduce notion of memory container to improve parallelism.

   But this approach works when the number of banks is much larger than the number of cores as the probability of bank-conflicts would be low. In addition, the weakness of the random algorithm characteristics is that the worst case scenario is the same.

   So it would have been better if we had shown a graph of performance results based on core and number of banks.