Remote Memory References at Block Granularity

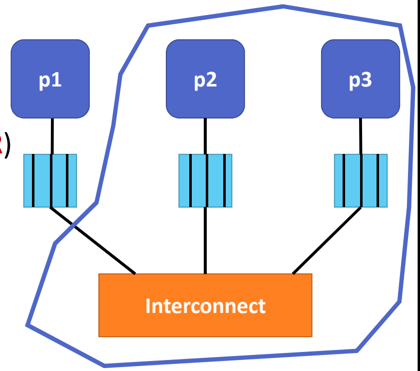
Hagit Attiya and Gili Yavneh, Technion

Remote Memory References

Access to local memory is cheap

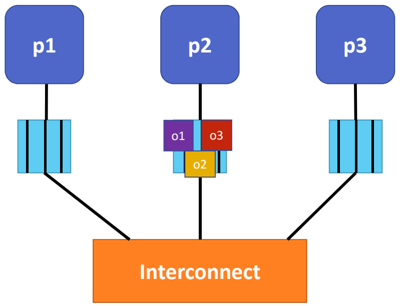
Access to remote memory is not

A remote memory reference (**RMR**) happens when a process accesses the remote memory



RMRs (CC Model) by Example

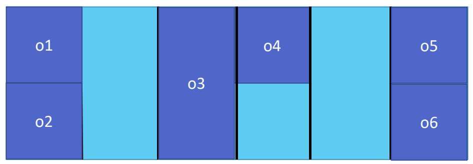
- p2 reads o1
- p1 reads o2 ✓
- p3 writes o2 ✓
- p3 reads o2
- p3 reads o1 ✓



RMRs at Block Granularity

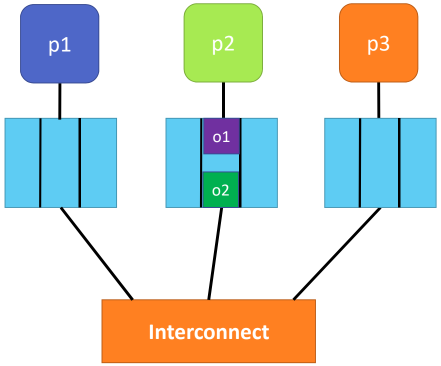
Memory traffic happens in **blocks**, typically holding > 1 object

A **block RMR** occurs if a process accesses an object in a block that is not in its local memory



Block RMRs: Example (CC Model)

- p1 reads o1 ✓
- p3 writes o2 ✓
- p1 reads o1 ✓
- p2 reads o2 ✓
- p2 reads o1
- p1 reads o2
- p3 writes o2

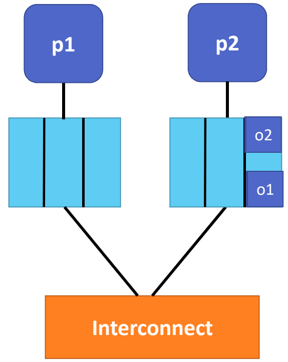


Why Block RMRs?

👍 Bringing one block to the local memory moves all objects in the block, saving future block RMRs (spatial locality)

👎 Different processes accessing different objects placed at the same block cause extra block RMRs (false sharing)

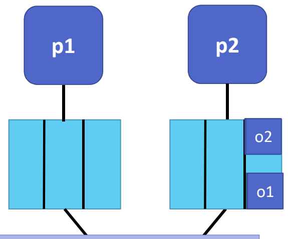
Goal: Place objects into blocks so as to minimizes the number of block RMRs



Why Block RMRs?

👍 Bringing one block to the local memory moves all objects in the block, saving future block RMRs (spatial locality)

👎 Different processes accessing different objects placed at the same block cause extra block RMRs (false sharing)



For an access sequence, the **minimum** number of **block RMRs** over all possible placements
≤ number of **RMRs** for the sequence

Our Results I

Finding an optimal placement is **NP-hard** when objects have **different sizes**, for **single process** and **known access sequence**

Reduction from **bin packing**

All other results assume objects have the same size

Our Results II: CC Model

Finding an optimal placement is **NP-hard** for blocks with ≥ **3** objects, two processes, and **known access sequence**

Validating a hunch in [Bolosky & Scott, 1993]

By reduction from **graph partitioning**

Some similarity to [Petrank & Rawitz, POPL 2002]

Our Results II: CC Model

Finding an optimal placement is **NP-hard** for blocks with ≥ **3** objects, two processes, and **known access sequence**

Polynomial time algorithm to find an **optimal** placement given an access sequence, for blocks with ≤ **2** objects

Finding a **maximum weighted matching** for a graph computed from the access sequence, using linear programming

Graph Partitioning

Finding an optimal placement is **NP-hard** for blocks with ≥ **3** objects, two processes, and **known access sequence**

Input: Undirected graph $G = (V, E)$, positive integer weights $w(v)$ for each vertex and $l(e)$ for each edge, and positive integers K, J

Question: Can V be partitioned into disjoint sets $V_1, \dots V_m$ such that:

- For every i, $\sum_{v \in V_i} w(v) \leq K$
- $\sum_{e \in E'} l(e) \leq J$ for E', the edges with endpoints in different sets

NP-Complete for K≥3, even with unit (1) vertex and edge weights

Block Placement Decision Problem (CC Model)

Given an access sequence $(p_{i_1}, a_1, o_{j_1}) \dots (p_{i_m}, a_m, o_{j_m})$, for processes $p_1 \dots p_n$ and objects $o_1 \dots o_n$, and an integer R

Is there a **B-Block Placement** $O_1 \dots O_l$ such that:

- Each object appears in exactly one block O_j
- Each block contains at most B objects, and
- The number of block RMRs for the access sequence $\leq R$

Reduction from Graph Partitioning I

Given an input to the graph partitioning problem with unit edge and vertex weights and $K \geq 3$, obtain an input to the B-block placement decision problem:

- Two processes p_1, p_2
- Object o_i for each vertex v_i
- The number of objects per block, $B = K$
- The maximum number of block RMRs, $R = 4(|E| + J)$

Partitioning into sets provides a block placement

Reduction from Graph Partitioning II

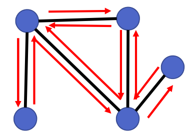
For each edge $e = (v_i, v_j)$:

$$\pi_e = (p_1, w, o_i), (p_1, w, o_j), (p_2, w, o_i), (p_2, w, o_j)$$

- If o_i, o_j are in the same block, 2 block RMRs are incurred during π_e
- Otherwise, 4 block RMRs are incurred during π_e

List edges in DFS order, each traversed twice: $e_1, e_2 \ldots e_{2|E|}$, and consider

$$\pi = \pi_{e_1} \cdot \pi_{e_2} \cdot \ldots \cdot \pi_{e_{2|E|}}$$



Reduction from Graph Partitioning II

For each edge $e = (v_i, v_j)$:

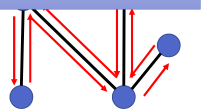
$$\pi_e = (p_1, w, o_i), (p_1, w, o_j), (p_2, w, o_i), (p_2, w, o_j)$$

- If o_i, o_j are in the same block, 2 block RMRs are incurred during π_e
- Otherwise, 4 block RMRs are incurred during π_e

Can adapt the sequence to be a **traversal** on the graph

List edges in DFS order, each traversed twice: $e_1, e_2 \ldots e_{2|E|}$, and consider

$$\pi = \pi_{e_1} \cdot \pi_{e_2} \cdot \ldots \cdot \pi_{e_{2|E|}}$$



Reduction from Graph Partitioning III

- Let E' be the edges with endpoints in different blocks (sets)
- The total number of RMRs is: $2 \cdot 2|E \setminus E'| + 4 \cdot 2|E'| = 4(|E| + |E'|)$

Endpoints in the same block

Endpoints in different blocks

- If there is a placement with $< R = 4(|E| + J)$ block RMRs then $4(|E| + |E'|) \leq R = 4(|E| + J)$ and $|E'| \leq J$
- Otherwise, for every placement the number of block RMRs is $4(|E| + |E'|) > R = 4(|E| + J)$ and $|E'| > J$

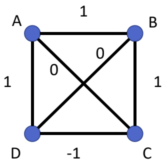
CC Model with 2 Objects per Block

Polynomial time algorithm to find an **optimal** placement given an access sequence, for blocks with ≤ **2** objects

Reduce to finding a **maximum weighted matching**

Input: undirected graph G=(V,E), weights $w(e)$ for each edge.

Output: set of pairwise non-adjacent edges with maximum weight



Can be found using linear programing in O(V^2E)

CC Model with 2 Objects per Block: Reduction

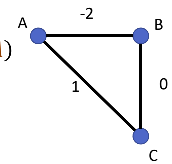
Given input to the B-Block placement problem, define a complete weighted graph:

- Vertex for every object
- The weight of each edge captures the number of block RMRs **saved** by placing the objects in the **same block**
  − extra block RMRs **paid** for placing them in the **same block**

A matching in the graph corresponds to a 2-block placement

CC Model with 2 Objects per Block: Weights

- For every pair of consecutive accesses to the same object *o*:
  - If they are accessed by the same process: for each object *o′* written in between by another process, deduct 1 from *(o, o′)*
  - If they are accessed by different processes: for each object *o′* accessed in between by the same process as the last access, add 1 to *(o, o′)*
- Example: weights for the sequence
  $(p_1, w, A), (p_2, w, B), (p_1, w, A), (p_2, w, B), (p_3, w, C), (p_3, w, A)$



CC Model with 2 Objects per Block: Weights

- For every pair of consecutive accesses to the same object *o*:
  - If they are accessed by the same process: for each object *o′* written in between by another process, deduct 1 from *(o, o′)*
  - If they are accessed by different processes: for each object *o′* accessed in between by the same process as the last access, add 1 to *(o, o′)*

Claim: The number of block RMRs for the 2-block placement corresponding to a matching is the number of RMRs for the sequence, minus the weight of the matching

Corollary: The maximum weighted matching corresponds to a 2-block placement with the minimal number of block RMRs

DSM Model is More Complicated…

- If **cache coherence** is supported and **cost of invalidation is negligible**, results are like the CC model
- Without **cache coherence**, an optimal placement can be found in **polynomial time** for known access sequences
  - An object is placed in the local memory of the process most accessing it
  - This variant is mostly theoretical

Wrap-up and Extensions

- Block RMRs estimate the cost of remote memory references, taking into account the block organization of memory (CC and DSM flavors)
- Given an access sequence, it is NP-hard to find an optimal placement, if a block can hold ≥ 3 objects; otherwise, it is polynomial

☞ Bounded local memory [Lavaee, POPL 2016]

☞ Find an (almost) optimal placement for a family of accesses sequences, e.g., search tree traversals

☞ Minimize the expected number of block RMRs, when the access sequence is chosen from some distribution