A Study Towards Optimal Data Layout for GPU Computing

Eddy Z. Zhang, Han Li and Xipeng Shen

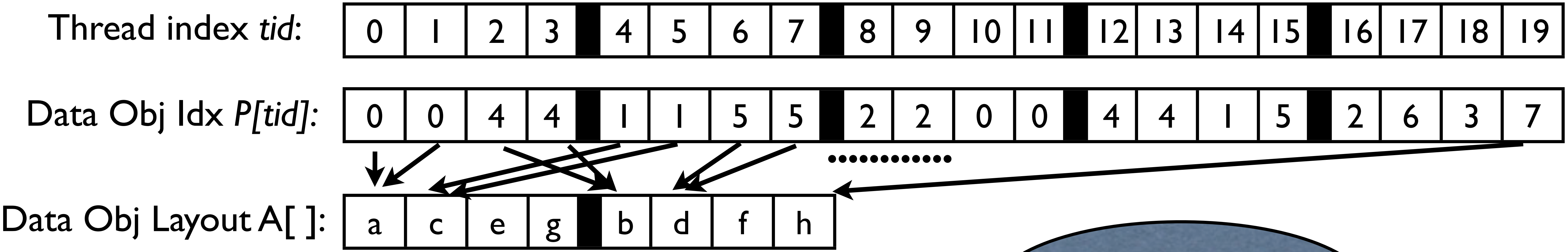
Presenter: Mingzhou Zhou

The College of William and Mary

Problem Description

Irregular Memory References

- * A mem. transaction -- read/write a consecutive memory segment at once
- * A thread warp -- execute only when all data for all threads in the warp is ready
- * Random and complicated patterns
- * Example: thread warp size - 4, mem. segment size - 4. **Access A[P[tid]]**



Thread index *tid*:	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
Data Obj Idx *P[tid]*:	0	0	4	4	1	1	5	5	2	2	0	0	4	4	1	5	2	6	3	7
Data Obj Layout A[ ]:	a	c	e	g	b	d	f	h												

9 Mem. Transactions

Data Layout Transformation

* Complexity -- NP Completeness if not using any extra memory space or thread relocation

Lack of a study for optimal mapping, previous studies are based on simple heuristics

Duplication Approach

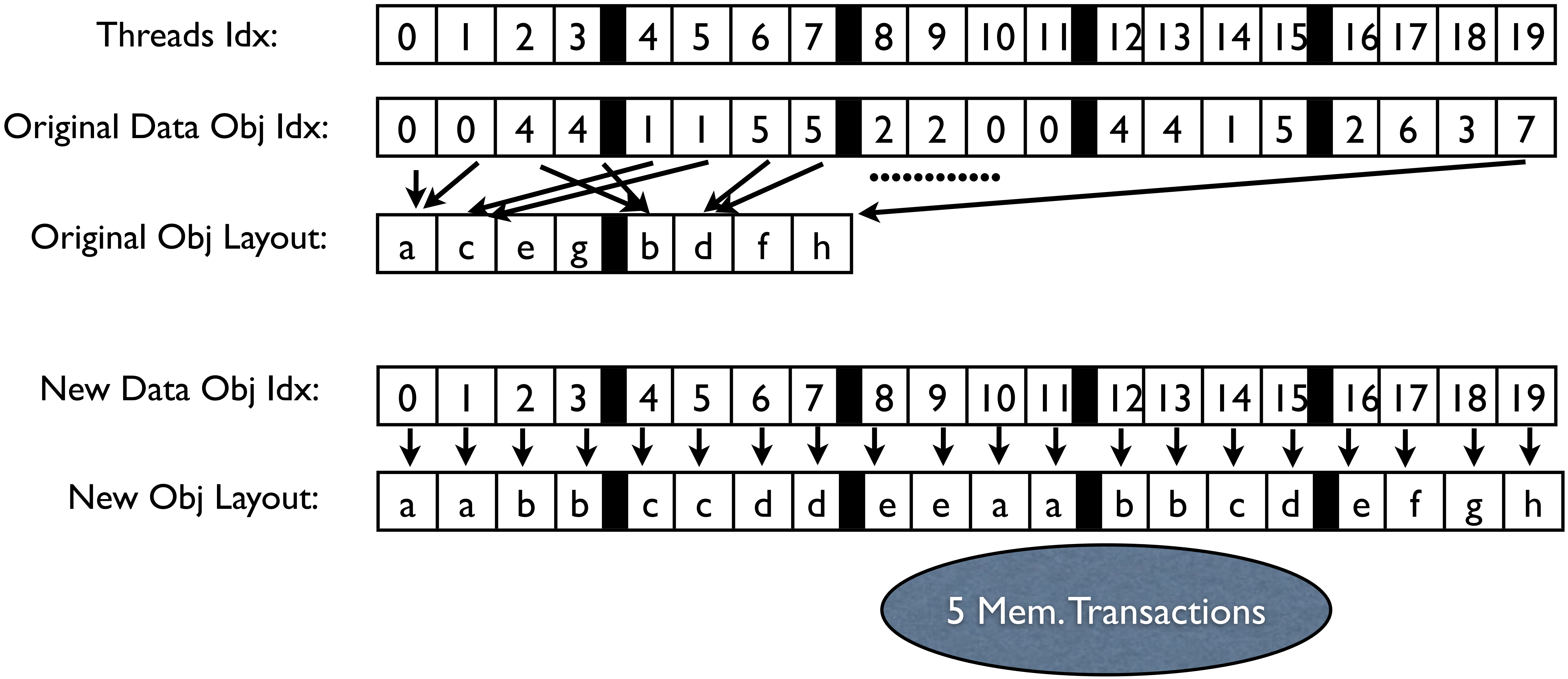
Transform data layout only.

* Duplicate data objects.

* Add space overhead: data size = # threads at a data reference.

* Optimal number of memory transactions

* Adaptive partial duplication [Zhang+:ASOLOS'11]



Threads Idx:	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
Original Data Obj Idx:	0	0	4	4	1	1	5	5	2	2	0	0	4	4	1	5	2	6	3	7
Original Obj Layout:	a	c	e	g	b	d	f	h												
New Data Obj Idx:	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
New Obj Layout:	a	a	b	b	c	c	d	d	e	e	a	a	b	b	c	d	e	f	g	h

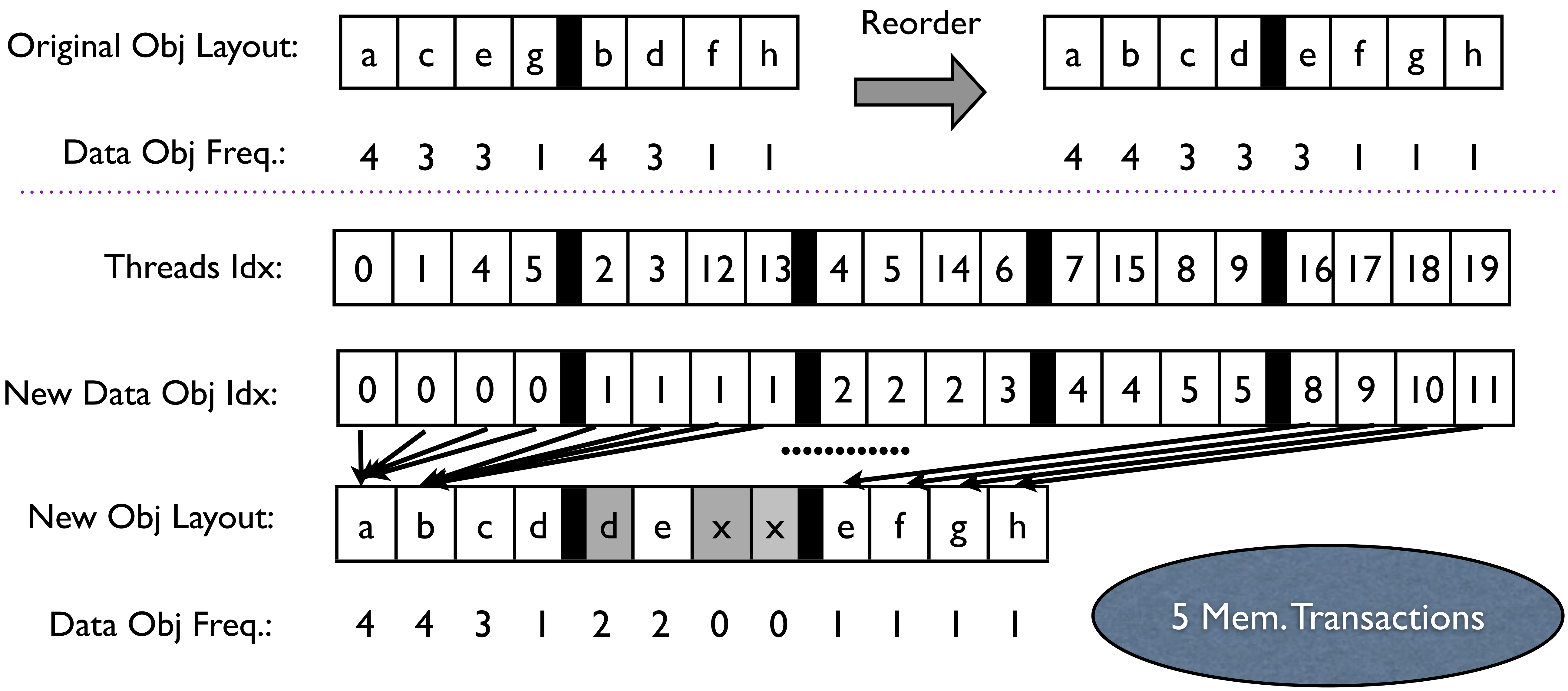
Padding Approach

Reorder Both Threads and Data

* Step 1: Reorder data objects based on access frequencies.

* Step 2: Reorder threads according to their data object order from Step 1.

* Step 3: Put data objects into memory segments. Duplicate or pad dummy objects only when necessary.



Original Obj Layout: a c e g | b d f h

Data Obj Freq.: 4 3 3 1 4 3 1 1

Reorder

a b c d | e f g h

4 4 3 3 3 1 1 1

Threads Idx: 0 1 4 5 | 2 3 12 13 | 4 5 14 6 | 7 15 8 9 | 16 17 18 19

New Data Obj Idx: 0 0 0 0 | 1 1 1 1 | 2 2 2 3 | 4 4 5 5 | 8 9 10 11

....

New Obj Layout: a b c d | d e x x | e f g h

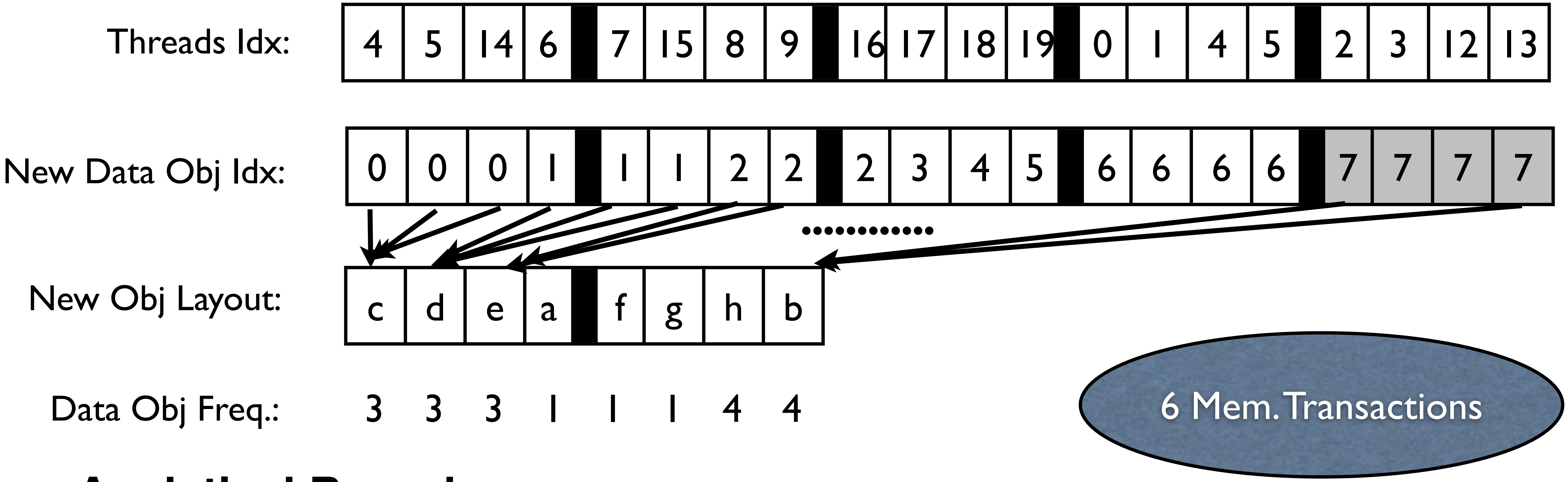
Data Obj Freq.: 4 4 3 1 2 2 0 0 1 1 1 1

5 Mem. Transactions

Approximation with Confidence

Reordering Threads More Aggressively

- * Step 1: Group threads:
  - (a). those accessing a popular object (accesses >= warp size)
  - (b). those accessing alone object (1 access only)
  - (c). others
- * Step 2: Form warps for (a) and (b). Order their objects accordingly.
- * Step 3: Form warps for (c) and the remainders of (a) and (b). Order objects.



Threads Idx: 4 5 14 6 ▮ 7 15 8 9 ▮ 16 17 18 19 ▮ 0 1 4 5 ▮ 2 3 12 13

New Data Obj Idx: 0 0 0 1 ▮ 1 1 2 2 ▮ 2 3 4 5 ▮ 6 6 6 6 ▮ 7 7 7 7

....

New Obj Layout: c d e a ▮ f g h b

Data Obj Freq.: 3 3 3 1 1 1 4 4

6 Mem. Transactions

Analytical Bound

- * Optimal case: Total number of memory transactions = Total number of warps.
- * Upper bound: Optimal + R (|c| + #remainder threads) / W (warp size)

Conclusion

- Data layout transformation is critical for GPU
    - Two algorithms to achieve the optimal
        - duplication & padding (less space)
    - One algorithm to approx. with guarantees
- A first step to reveal the limit
- Future
    - Testing and refining them for practical usage

Questions?