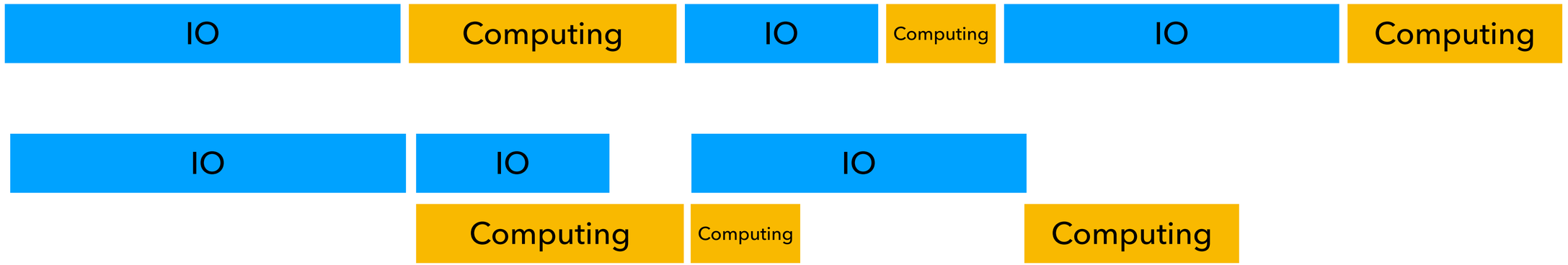
PipeGraph: Pipelining in Dependency-Driven Graph Processing

Basic Ideas

Yuantian Ding 1/26/2021

Pipelining

Do each step at same time



- Target
  - Make IO and Computing take same time.
  - Make IO time be a constant.

Pipelining - Our Solution

Do each step at same time

- Make IO and Computing take same time. => Dependency-Driven Method Like Lumos [ATC'19] / Graph Compressing
- Make IO time be a constant. => Partition Balancing

Much More Efficient

Lumos: Dependency-Driven Graph Processing

An IO Efficient Approach

- Assume computing PR(v) only depend on its in-edge(not out-edge).
- Abstract PageRank into Partial Aggregation operation.

Graph Computation

$$\mathrm{PR}(v^t) = \underbrace{1-d}_{f} + d \times \overbrace{\sum_{\mathrm{inedges}(v)}}^{\bigoplus} \mathrm{PR}(u^{t-1}) \;/\; |\mathrm{outedges}(u)|$$

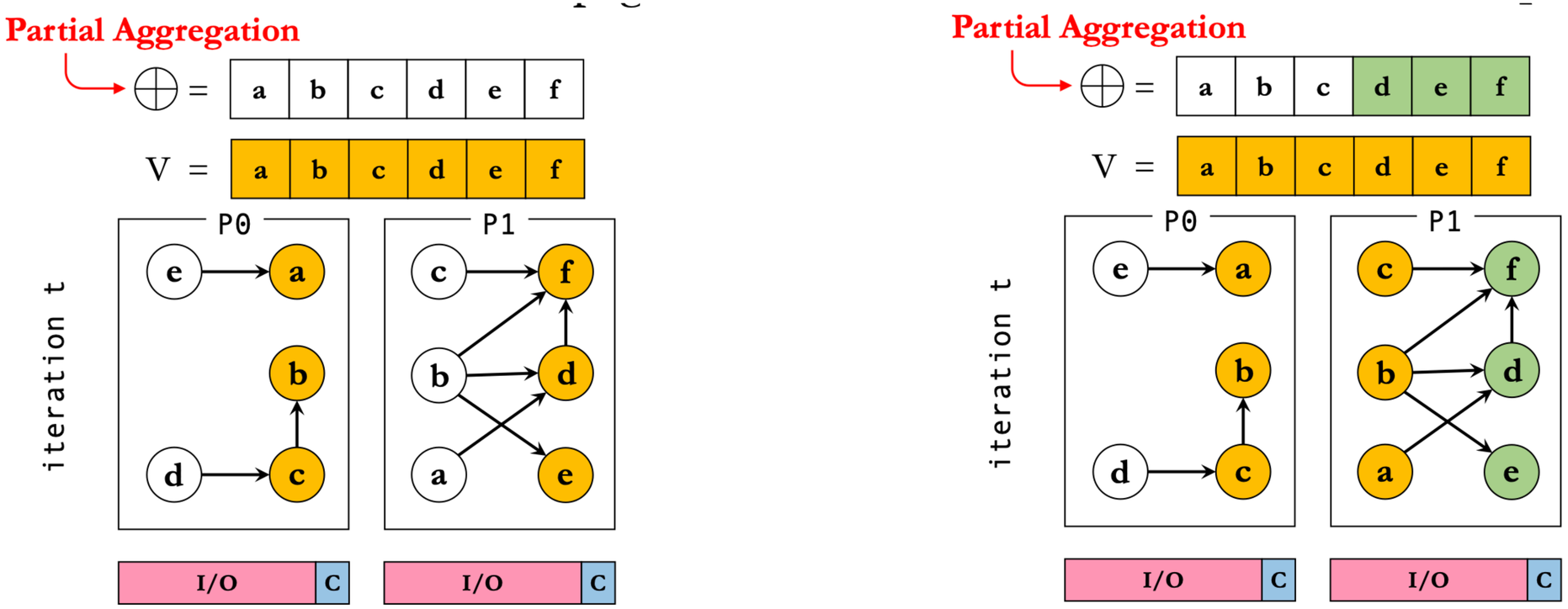
$$v^t = f\Big(\bigoplus_{\mathrm{inedges}(v)} (u^{t-1})\Big)$$

Partial Aggregation

$$\mathrm{inedges}(v) = \mathrm{X} \cup \mathrm{Y} \implies \bigoplus_{\mathrm{inedges}(v)} (u^t) = \bigoplus \Big(\bigoplus_{\mathrm{X}} (u^t) \;,\; \bigoplus_{\mathrm{Y}} (u^t) \Big)$$

Lumos: Dependency-Driven Graph Processing

Computing Future Value

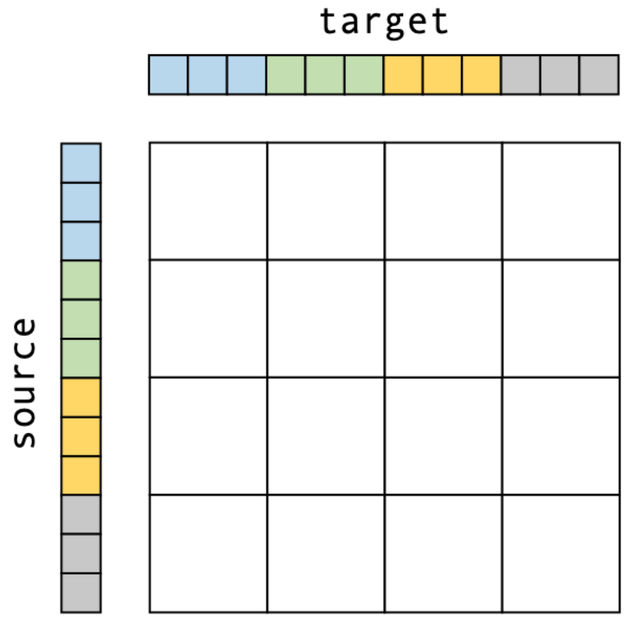


- First Iteration: Computing all vertex one pass.
- In the last partition(P1), we can compute the value for next iteration.

Lumos: Dependency-Driven Graph Processing

Computing Future Value

- Base on these ideas, Vora et al propose Lumos [ATC'19].
- Based on data layout in GridGraph [ATC'15].



Lumos: Dependency-Driven Graph Processing

Computing Future Value

- We use 0 to show in this Grid iteration 0 is computed. Coresponding value of in-edge added to vertex value.
- P1(0) means that all vertices in P1 has completed iteration 0.

src\target	P1(0)	P2(0)	P3(0)	P4(0)	P5(0)
P1	0	0	0	0	0
P2	0	0	0	0	0
P3	0	0	0	0	0
P4	0	0	0	0	0
P5	0	0	0	0	0

Lumos: Dependency-Driven Graph Processing

Computing Future Value

- We first compute (P2-P5, P1).
- Then we read (P1, P1) from disk, add all edge (P1,P1) to vertices in P1.
- We found P1 has complete Iteration 1, we can add all edge (P1, P1) to vertices in P1 and save it in a new place as the future value of Iteration 2.

src\target	P1(0)	P2(0)	P3(0)	P4(0)	P5(0)
P1	0	0	0	0	0
P2	1	0	0	0	0
P3	1	0	0	0	0
P4	1	0	0	0	0
P5	1	0	0	0	0

src\target	P1(1)	P2(0)	P3(0)	P4(0)	P5(0)
P1	1	0	0	0	0
P2	1	0	0	0	0
P3	1	0	0	0	0
P4	1	0	0	0	0
P5	1	0	0	0	0

src\target	P1(1)	P2(0)	P3(0)	P4(0)	P5(0)
P1	2	0	0	0	0
P2	1	0	0	0	0
P3	1	0	0	0	0
P4	1	0	0	0	0
P5	1	0	0	0	0

Lumos: Dependency-Driven Graph Processing

Computing Future Value

- We found that upper triangles can all compute to Iteration 2.

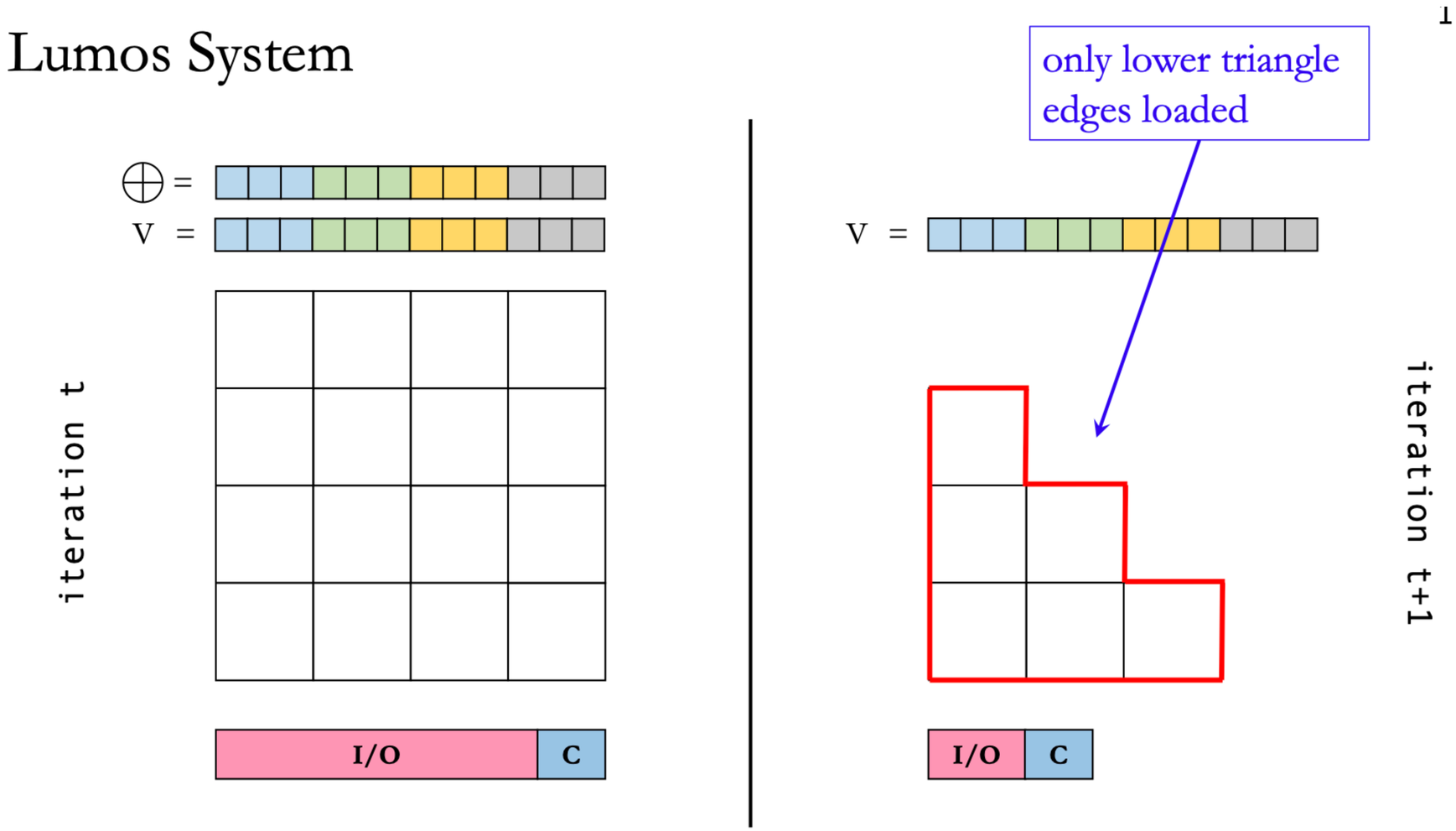
src\target	P1(1)	P2(0)	P3(0)	P4(0)	P5(0)
P1	2	2	2	2	2
P2	1	0	0	0	0
P3	1	0	0	0	0
P4	1	0	0	0	0
P5	1	0	0	0	0

src\target	P1(1)	P2(1)	P3(0)	P4(0)	P5(0)
P1	2	2	2	2	2
P2	1	2	2	2	2
P3	1	1	0	0	0
P4	1	1	0	0	0
P5	1	1	0	0	0

src\target	P1(1)	P2(1)	P3(1)	P4(1)	P5(1)
P1	2	2	2	2	2
P2	1	2	2	2	2
P3	1	1	2	2	2
P4	1	1	1	2	2
P5	1	1	1	1	2

Lumos: Dependency-Driven Graph Processing

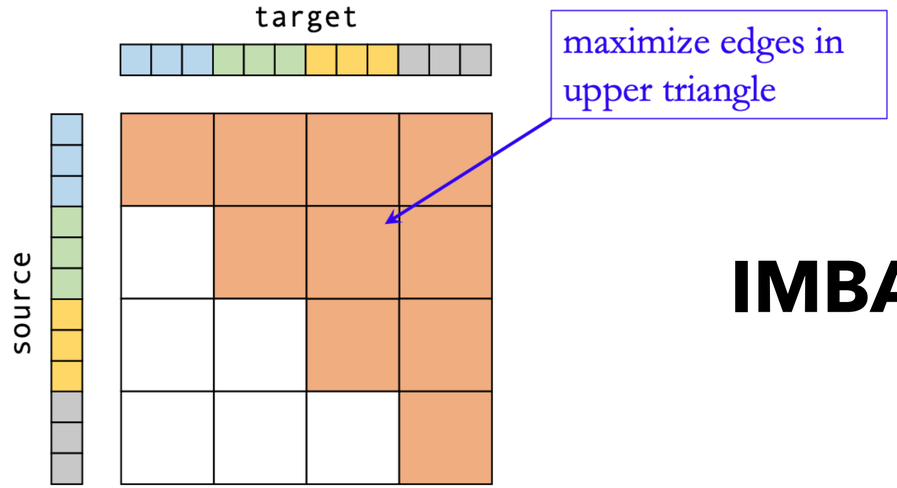
Computing Future Value



Lumos: Dependency-Driven Graph Processing

Computing Future Value

- Though Lumos achieve lower IO than GridGraph, to reduce 1/2 IO, we need to make all edges in upper triangle, which increase the imbalancing in each grid.



IMBALANCE in each grid size!

Our Solution

Reducing imbalance

- To achieve exactly 1/2 IO, we found another method, which will not cause any imbalance.

src\target	P1(1)	P2(1)	P3(1)	P4(1)	P5(2)
P1	2	2	2	2	2
P2	1	2	2	2	2
P3	1	1	2	2	2
P4	1	1	1	2	2
P5	1	1	1	1	3

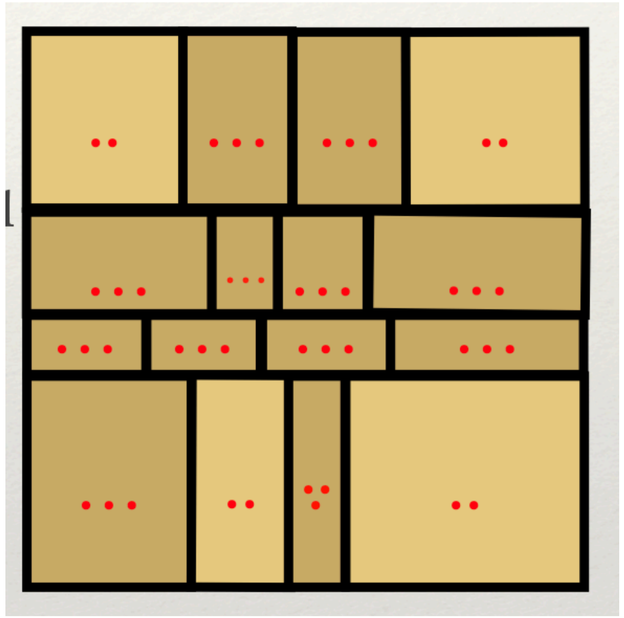
src\target	P1(1)	P2(1)	P3(1)	P4(1)	P5(2)
P1	2	2	2	2	2
P2	1	2	2	2	2
P3	1	1	2	2	2
P4	1	1	1	3	2
P5	1	1	1	3	3

src\target	P1(3)	P2(2)	P3(2)	P4(2)	P5(2)
P1	4	2	2	2	2
P2	3	3	2	2	2
P3	3	3	3	2	2
P4	3	3	3	3	2
P5	3	3	3	3	3

Our Solution

Reducing imbalance

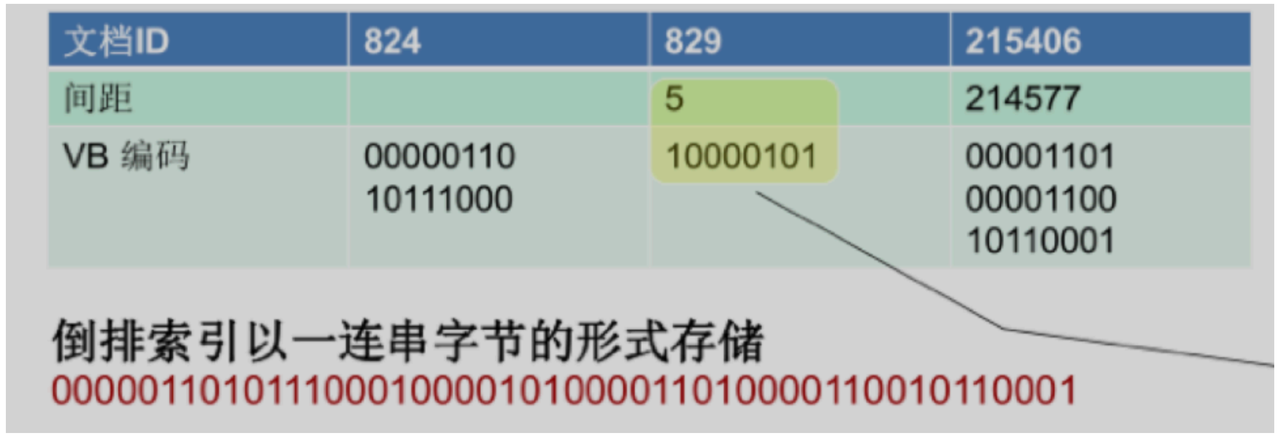
- We also use some partition technique like Graphene [FAST'17]



Compressing CSR Graph

VBCode

- To reduce IO cost, we use VB code to compress CSR Graph.
  - Storage the difference of two vertices id, instead of their id.
  - Save this id-difference in VBCode.
- This Technique always used in a posting of search engine.



Compressing CSR Graph

VBCode

- However, as real world graph follows power law. For many vertices, VB code may have a very low compression rate.
- So for an edge (u, v), if d(v) > d(u), we save this edge in v, and mark it as in-edge, if d(u) > d(v), we save this edge in u, and mark it as out-edge.
- This mark take one bit in every end of VB code.