X-Stream: Edge-centric Graph Processing using Streaming Partitions

A. Roy, I. Mihailovic, W. Zwaenepoel

Presented by: Samuil Stoychev

Graphs Processing

- Growing use in <u>**social networks**</u>, <u>**web rankings**</u> and others.
- Modern graphs can contain <u>**billions of edges**</u>.

Table 1: Popular benchmark graphs.

Graph	Vertices	Edges
LiveJournal [9]	4.8M	69M
Twitter 2010 [31]	42M	1.5B
UK web graph 2007 [10]	109M	3.7B
Yahoo web [8]	1.4B	6.6B

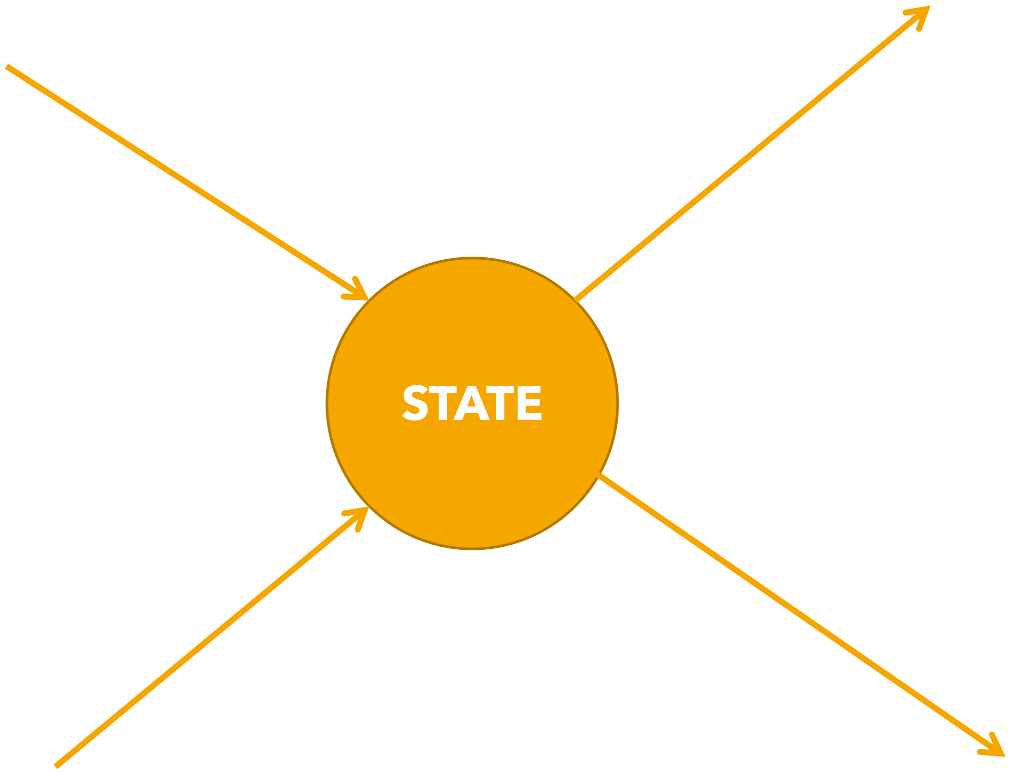
Source: "One Trillion Edges: Graph Processing at Facebook-Scale" (Ching et al., 2015)

Graph Processing Frameworks

- <u>**Distributed**</u> *(scale out)* frameworks:
  - Giraph
  - Pregel
  - Powergraph
- <u>**Single-machine**</u> *(scale up)* frameworks:
  - Graphchi
  - X-Stream

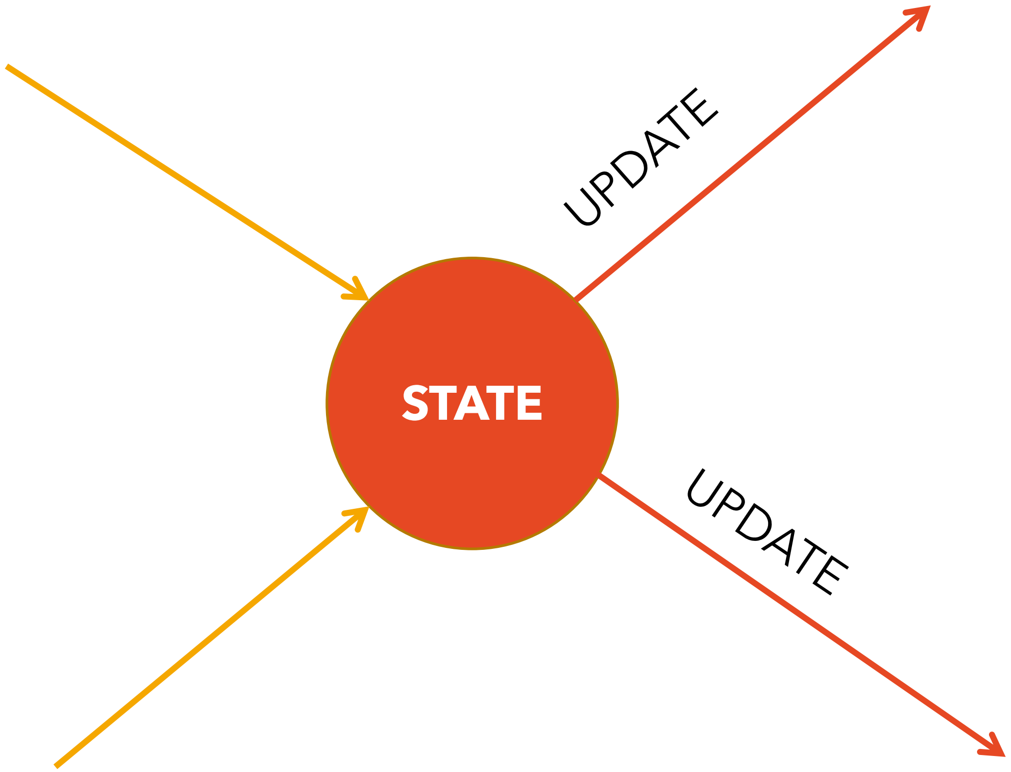
The Scatter-Gather Programming Model

- State is maintained in the vertices.
- User provides a **scatter** and a **gather** function.
- Scatter propagates updates to neighbours
- Gather accumulates updates from neighbours.



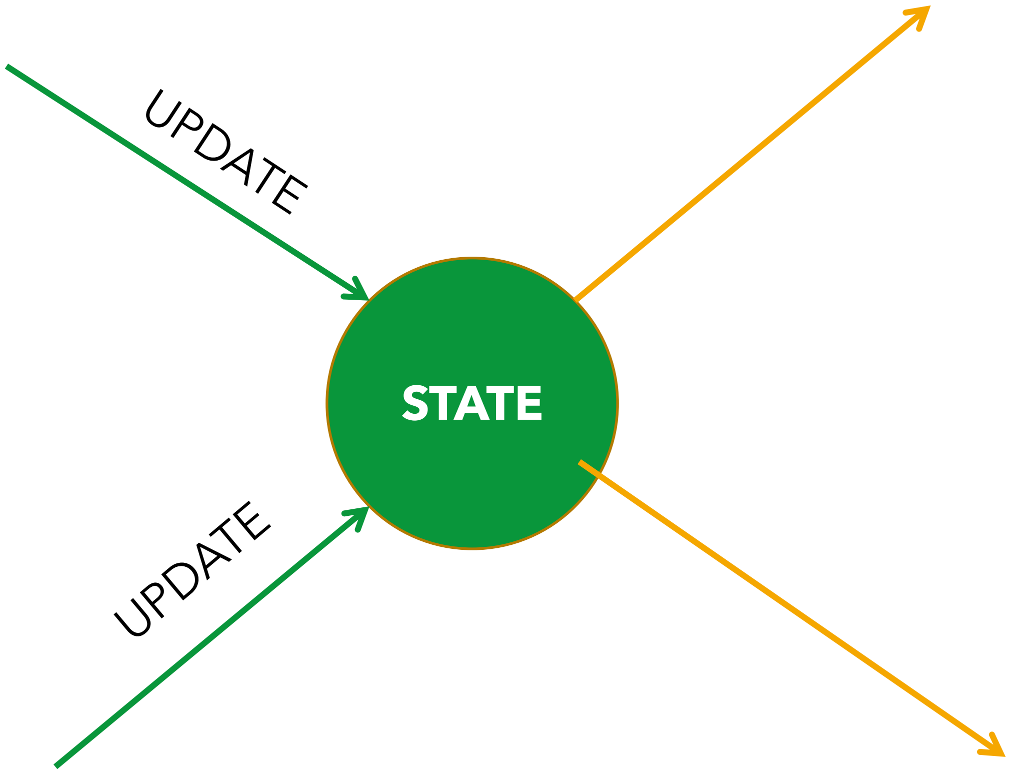
The Scatter-Gather Programming Model

- State is maintained in the vertices.
- User provides a **scatter** and a **gather** function.
- Scatter propagates updates to neighbours
- Gather accumulates updates from neighbours.



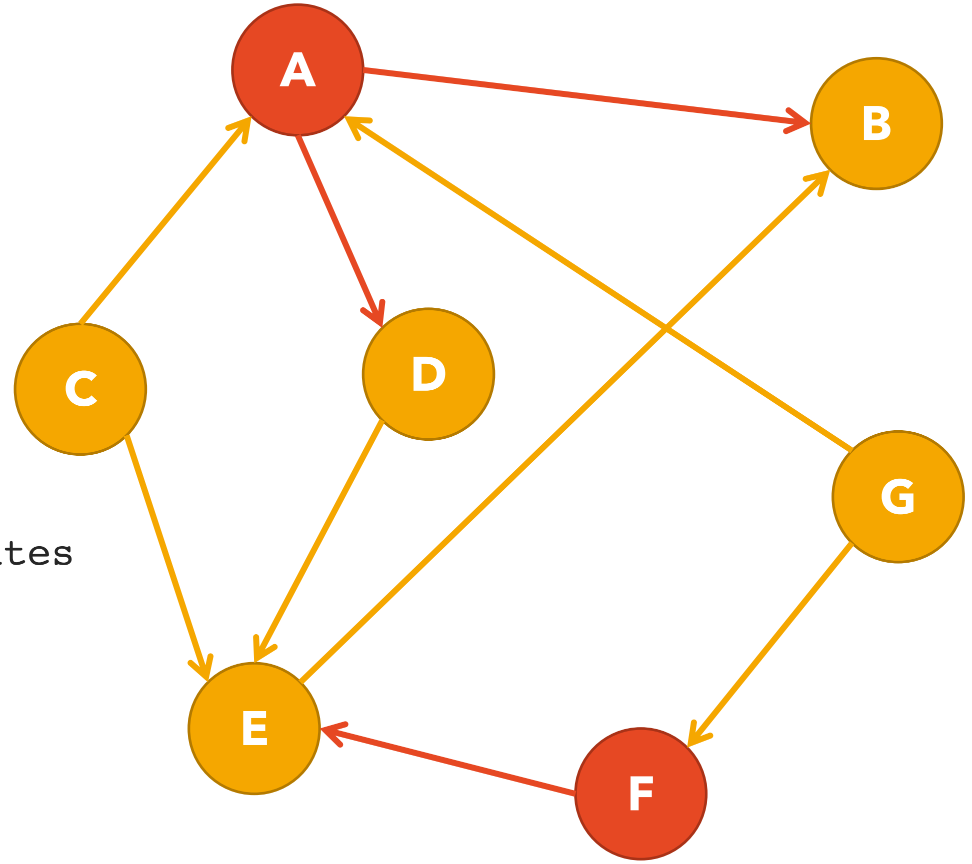
The Scatter-Gather Programming Model

- State is maintained in the vertices.
- User provides a **<u>scatter</u>** and a **<u>gather</u>** function.
- Scatter propagates updates to neighbours
- Gather accumulates updates from neighbours.



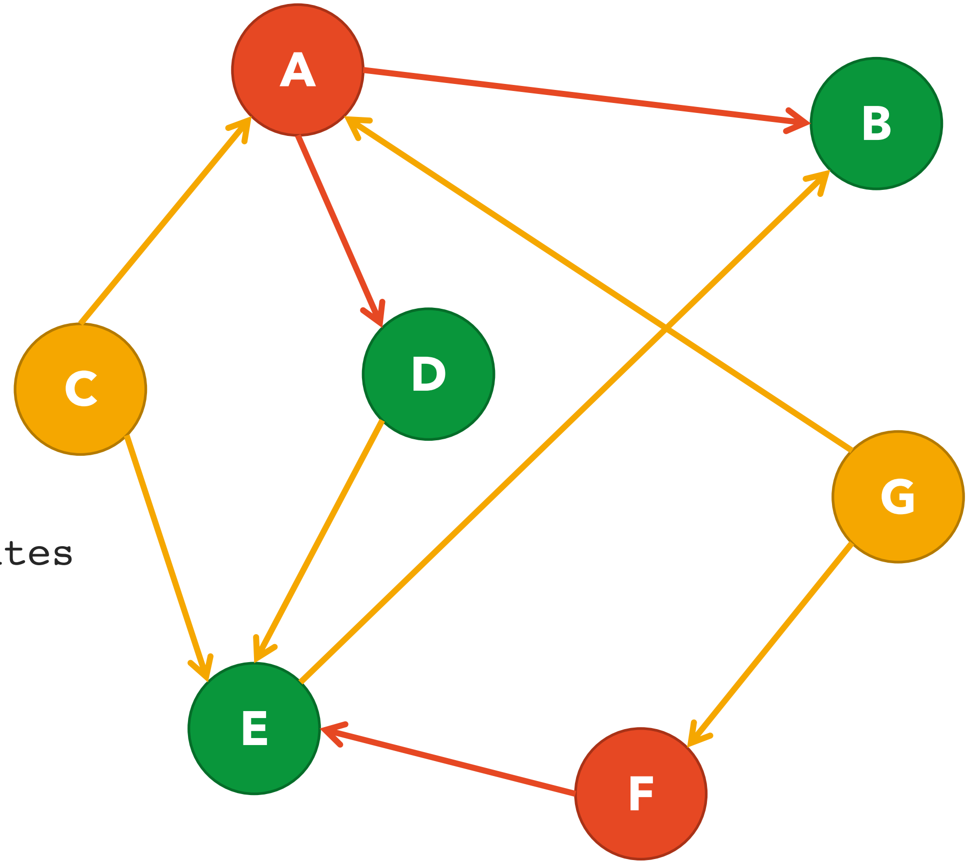
The Scatter-Gather Programming Model

```
vertex_scatter(vertex v)
        send updates over outgoing edges of v

vertex_gather(vertex v)
        apply updates from inbound edges of v

while not done
        for all vertices v that need to scatter updates
                vertex_scatter(v)
        for all vertices v that have updates
                vertex_gather(v)
```



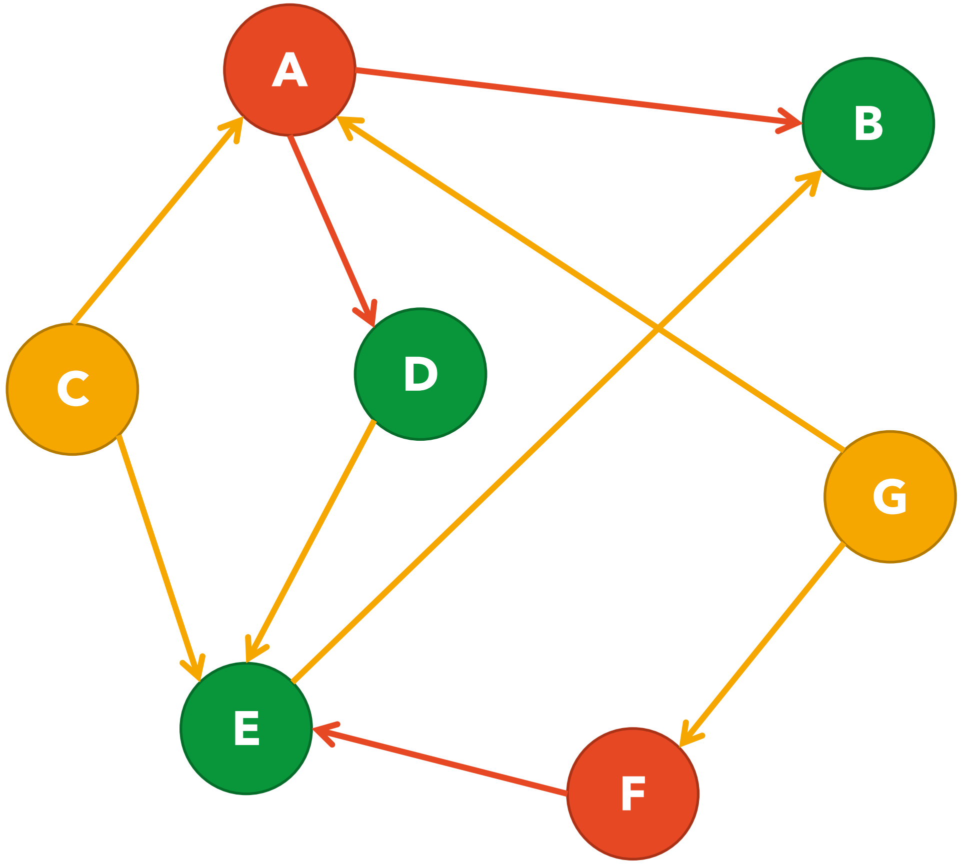
The Scatter-Gather Programming Model

```
vertex_scatter(vertex v)
        send updates over outgoing edges of v

vertex_gather(vertex v)
        apply updates from inbound edges of v

while not done
        for all vertices v that need to scatter updates
                vertex_scatter(v)
        for all vertices v that have updates
                vertex_gather(v)
```



The Scatter-Gather Programming Model

- Simple but powerful interface.
- Sufficient to express a variety of algorithms.
- Used by Pregel and Powergraph.



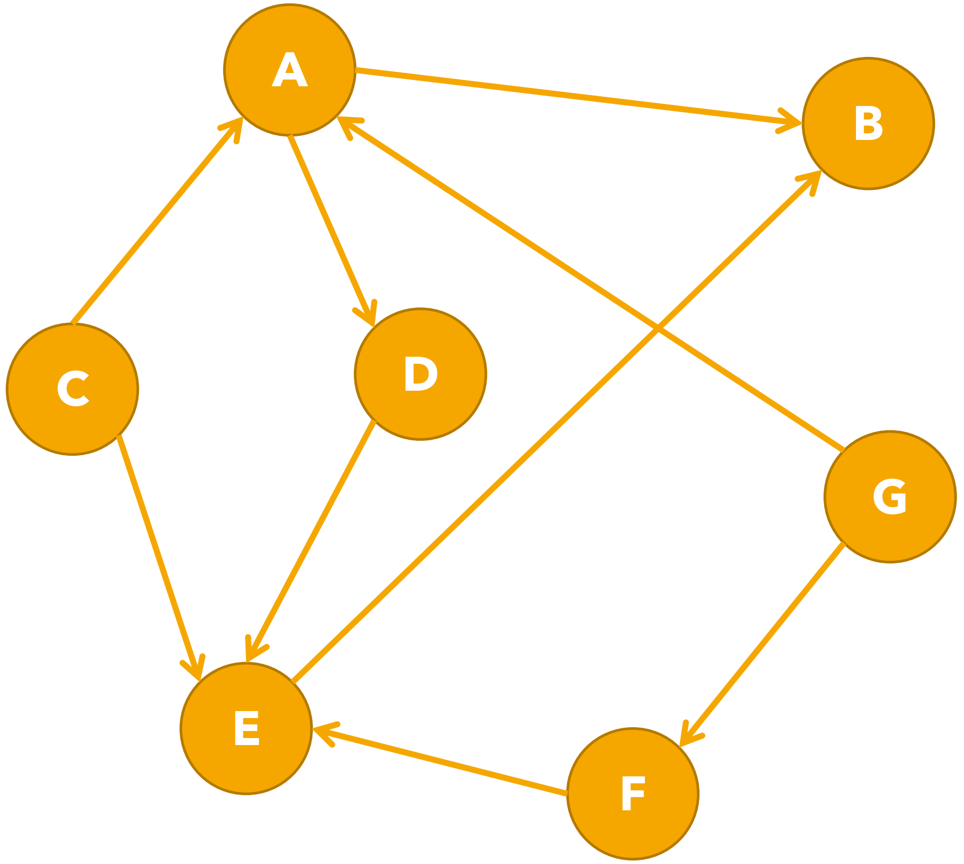
Vertex-Centric Scatter-Gather (BFS)

A
B
C
D
E
F
G

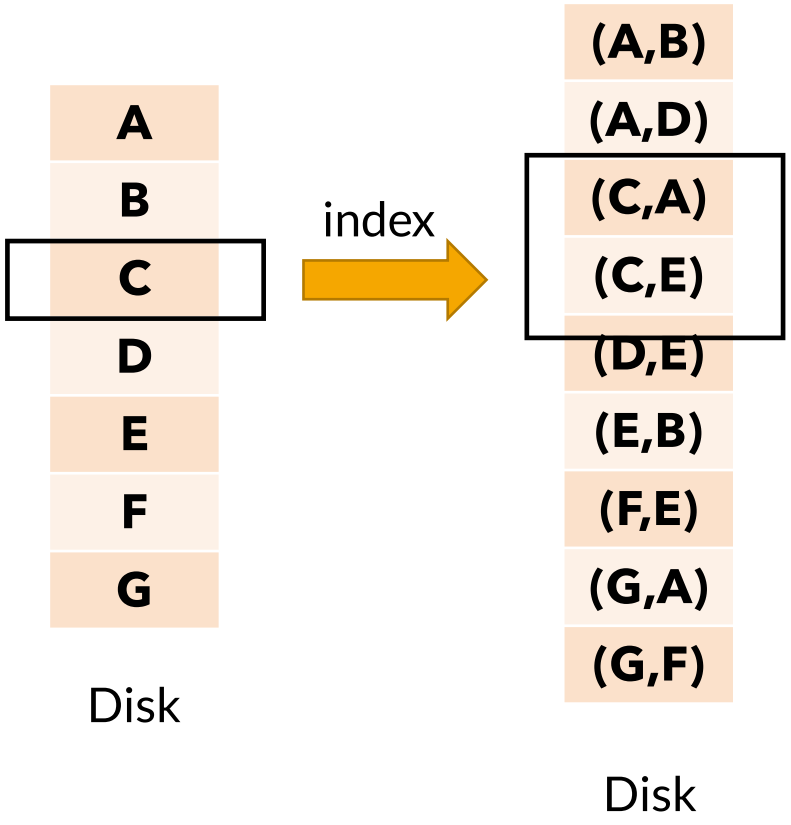
Disk

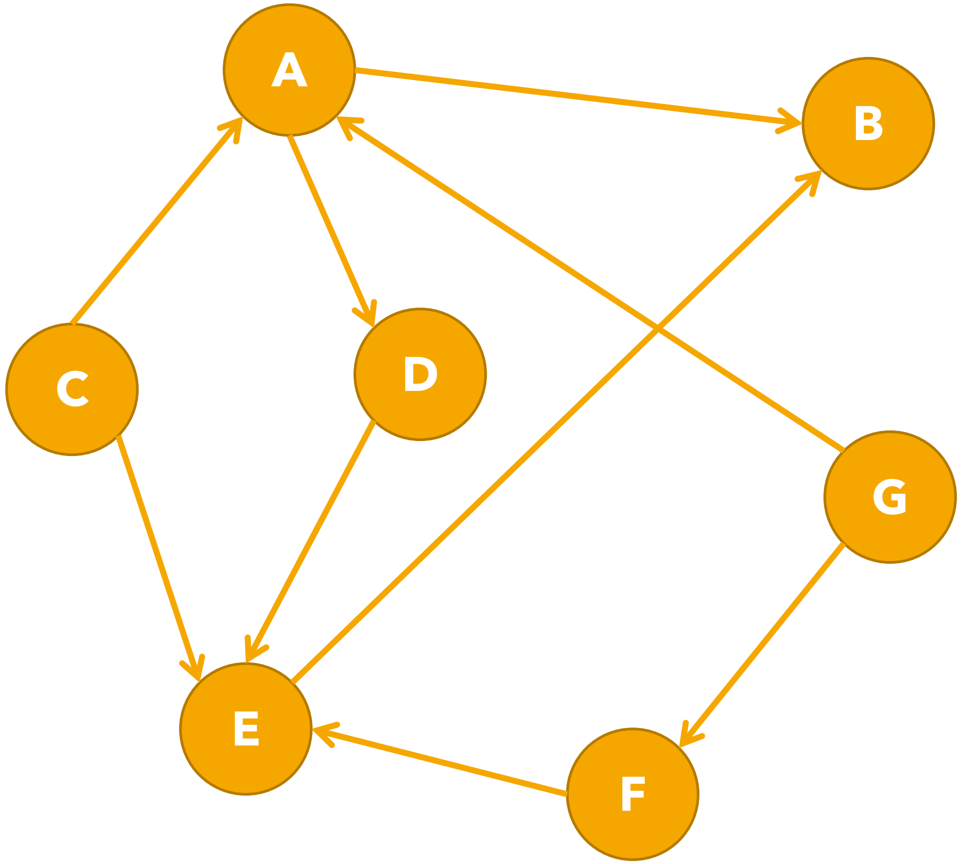
(A,B)
(A,D)
(C,A)
(C,E)
(D,E)
(E,B)
(F,E)
(G,A)
(G,F)

Disk



Vertex-Centric Scatter-Gather (BFS)





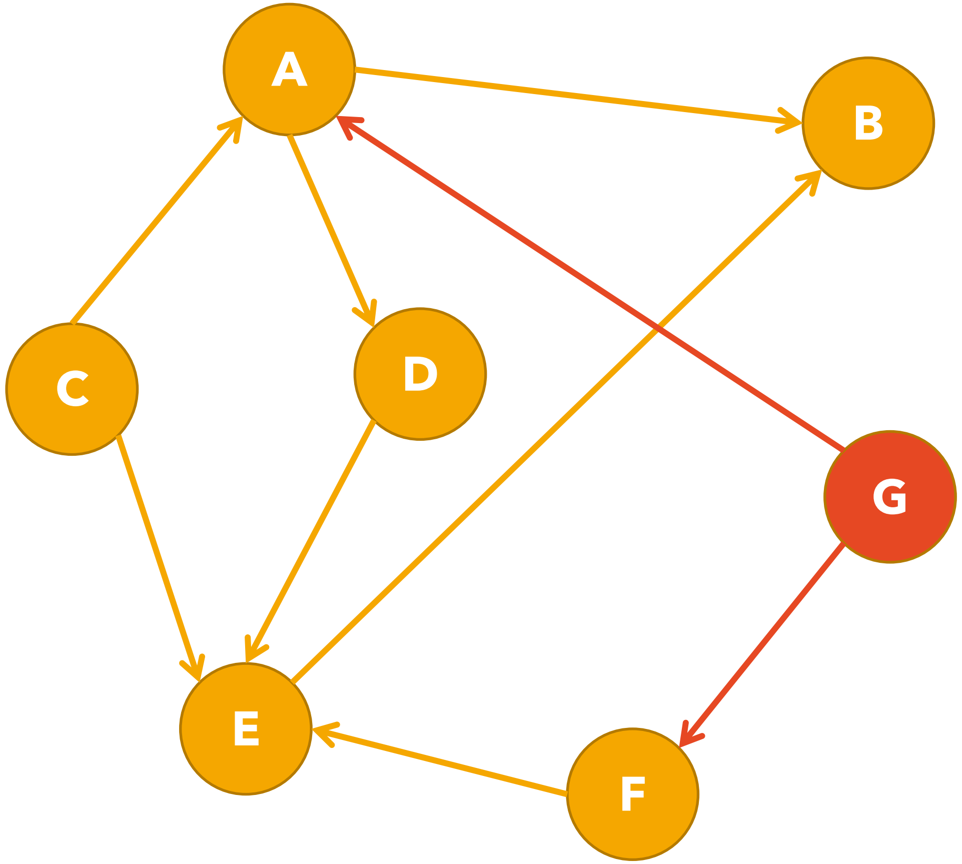
Vertex-Centric Scatter-Gather (BFS)

A
B
C
D
E
F
G

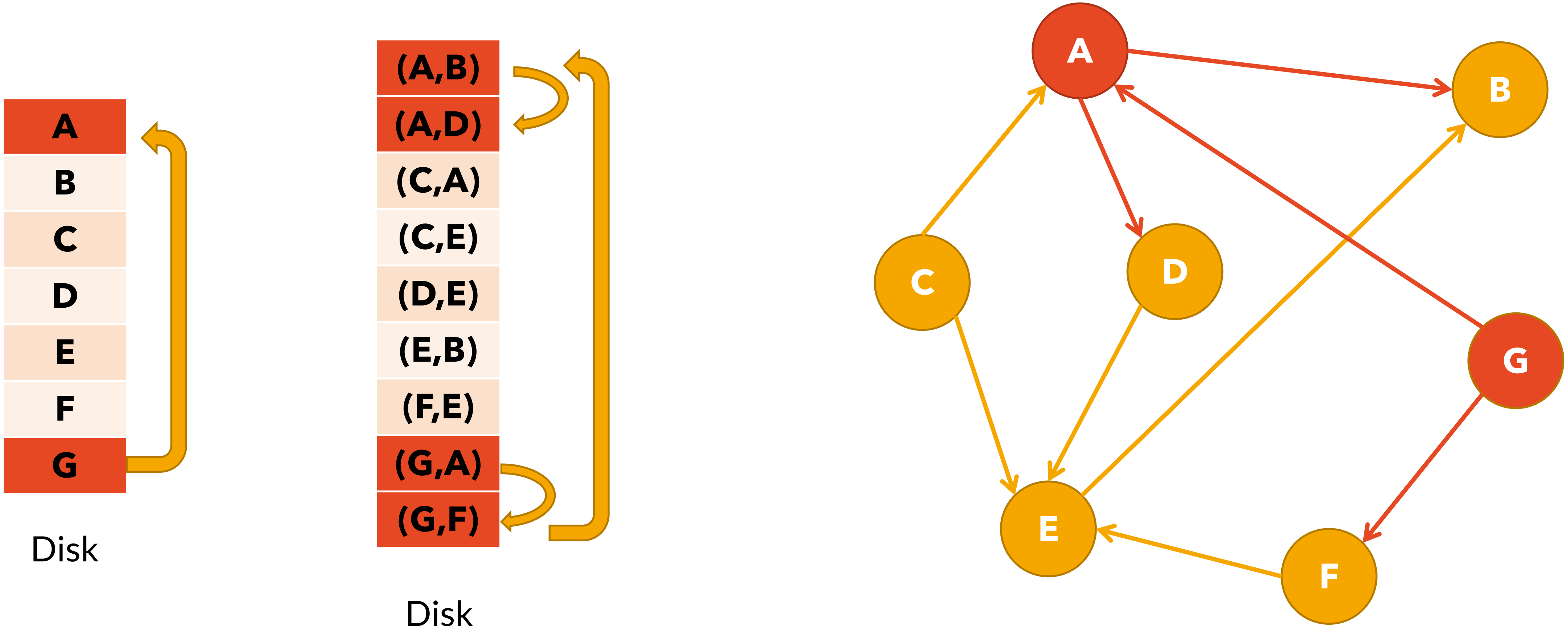
Disk

(A,B)
(A,D)
(C,A)
(C,E)
(D,E)
(E,B)
(F,E)
(G,A)
(G,F)

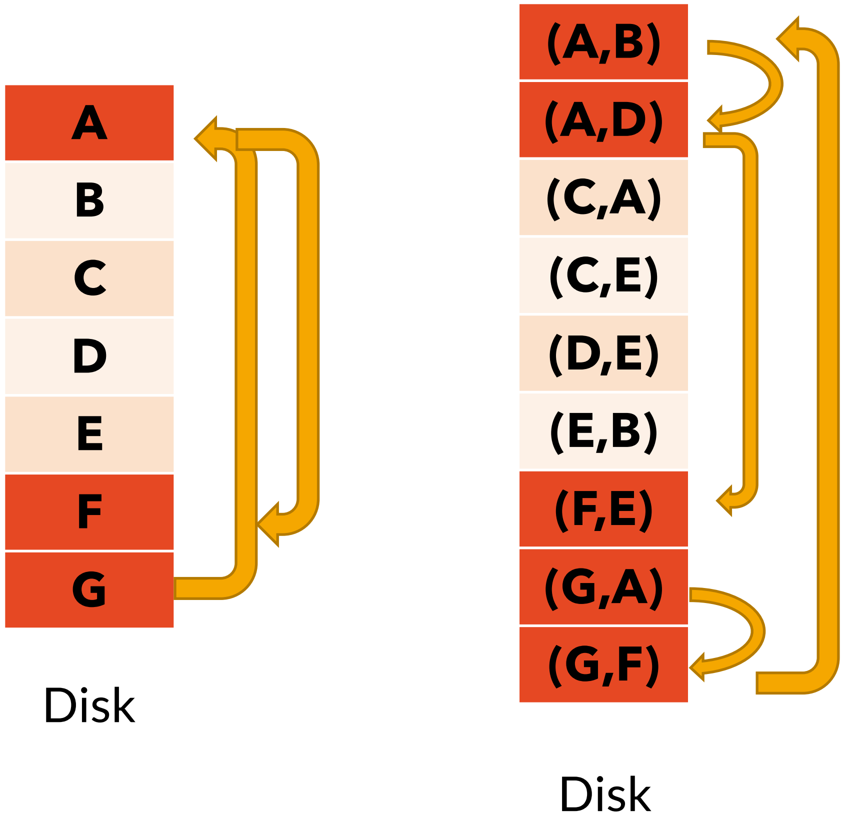
Disk

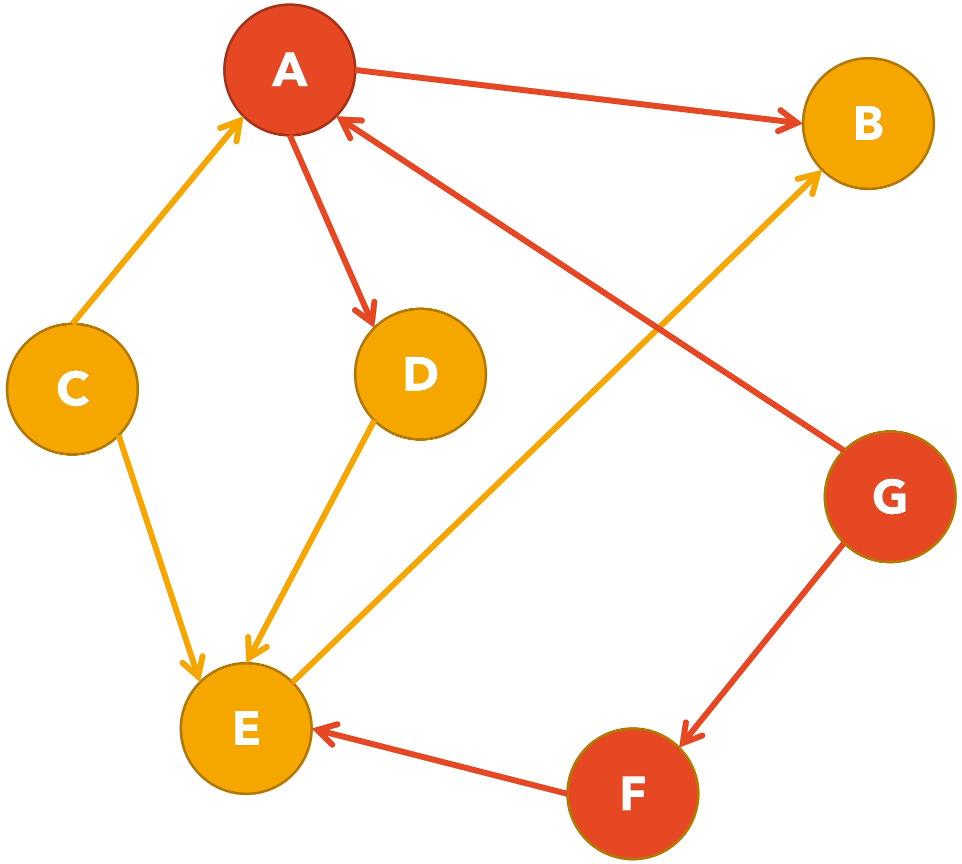


Vertex-Centric Scatter-Gather (BFS)

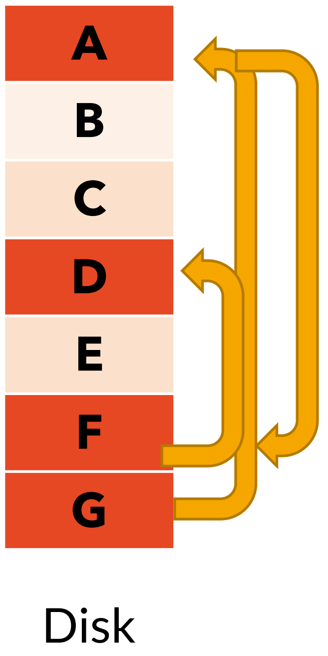


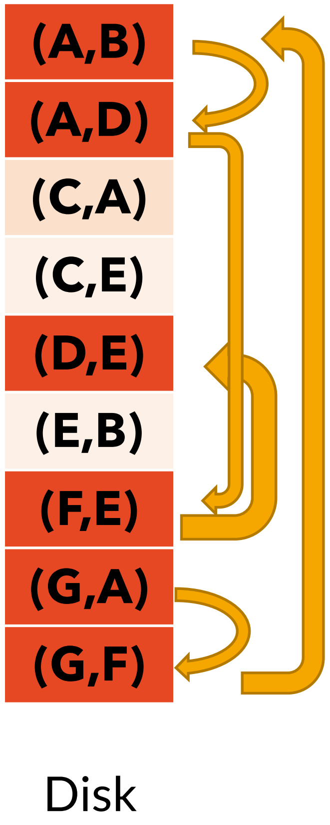
Vertex-Centric Scatter-Gather (BFS)

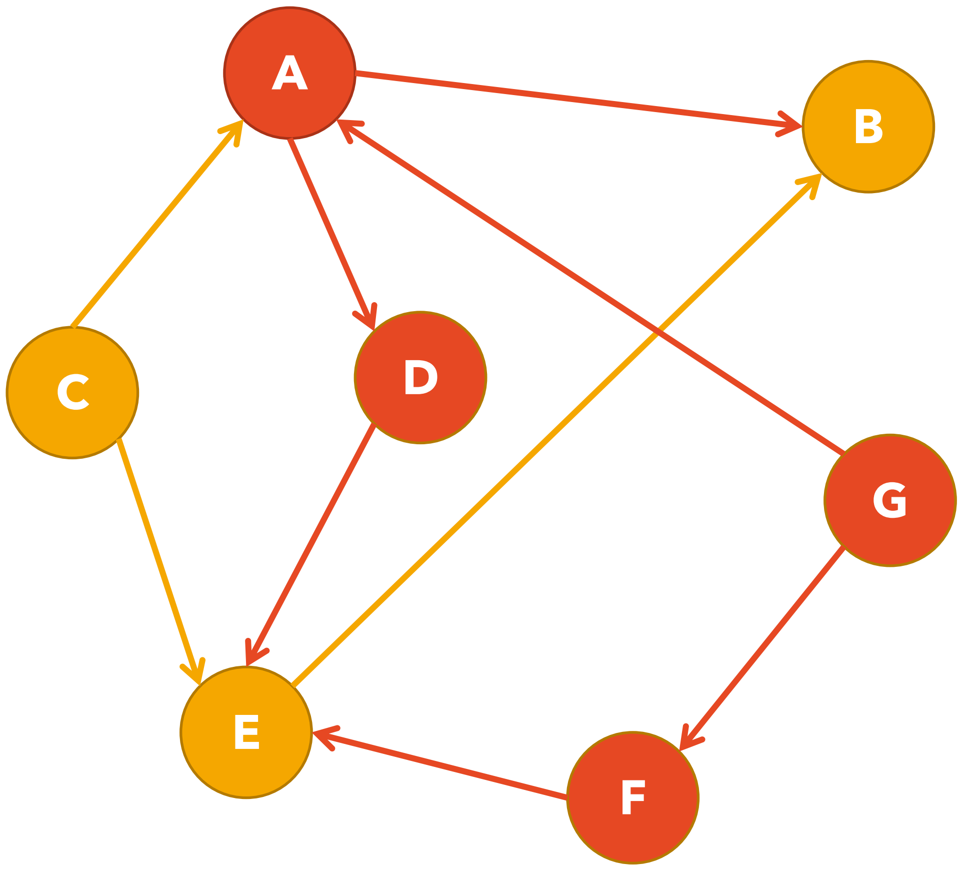




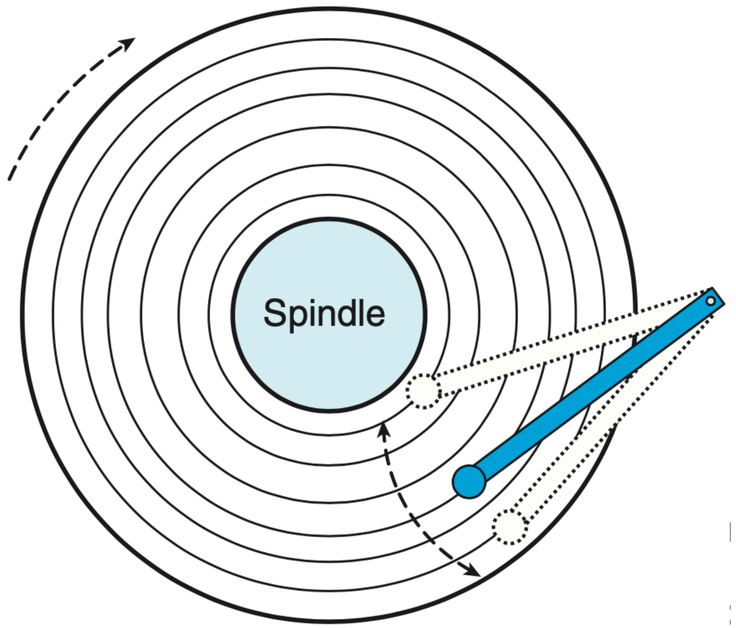
Vertex-Centric Scatter-Gather (BFS)







Random vs Sequential Access



Source: “Computer Systems: A Programmer’s Perspective” (Bryant and O’Hallaron)

- Random memory access is slower than sequential memory access.
- Especially problematic for disk devices.
- Programs need to exploit locality to achieve efficient memory access.

Random vs Sequential Access

Medium	Read (MB/s)		Write (MB/s)	
	Random	Sequential	Random	Sequential
RAM (1 core)	567	2605	1057	2248
RAM (16 cores)	14198	25658	10044	13384
SSD	22.5	667.69	48.6	576.5
Magnetic Disk	0.6	328	2	316.3

- Magnetic disk reads are 500+ times slower for random access.
- The gap in performance is bigger for slower media.

X-Stream

- Graph processing on a <u>single shared-memory machine</u>.
- Minimises random memory access through:
  - <u>Edge-centric scatter-gather</u>
  - <u>Streaming partitions</u>
- Supports both in-memory and out-of-core graphs.

Edge-Centric Scatter-Gather

```
edge_scatter(edge e)
      send update over e

update_gather(update u)
      apply update u to u.destination

while not done
      for all edges e
            edge_scatter(e)
      for all updates u
            update_gather(u)
```

- Streaming (or iterating over) edges instead of vertices.

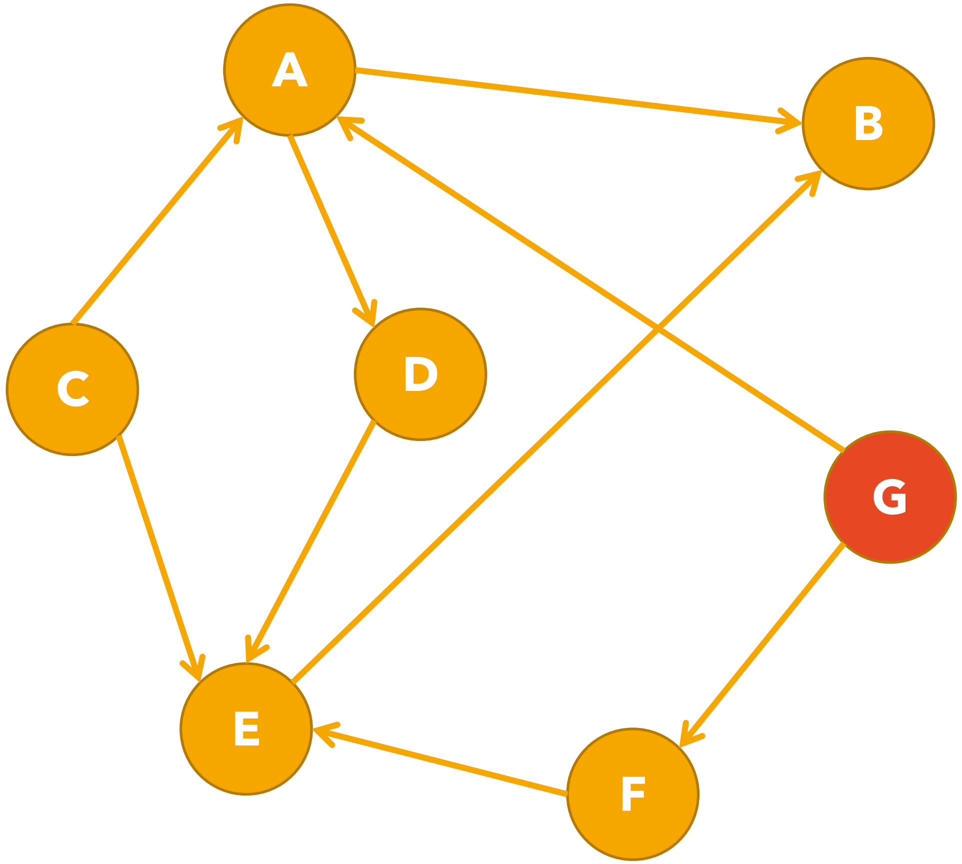
Edge-Centric Scatter-Gather (BFS)

A
B
C
D
E
F
G

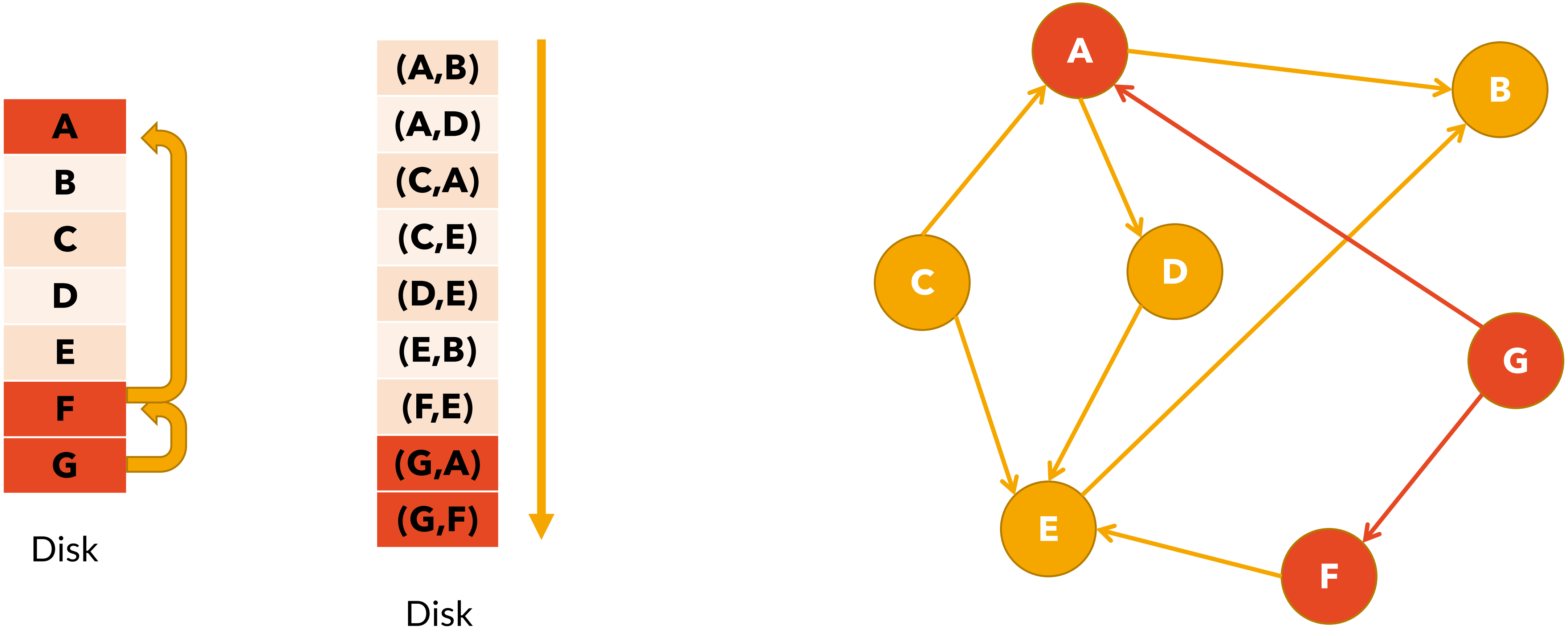
Disk

(A,B)
(A,D)
(C,A)
(C,E)
(D,E)
(E,B)
(F,E)
(G,A)
(G,F)

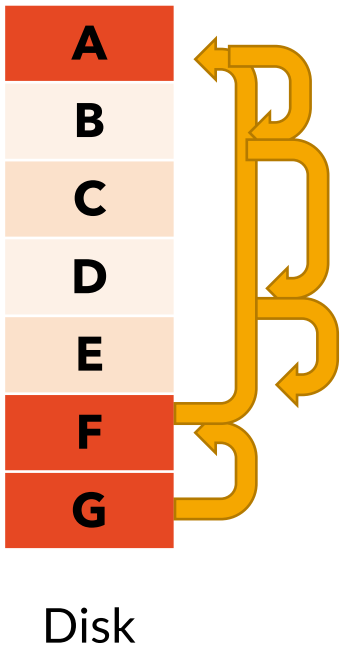
Disk

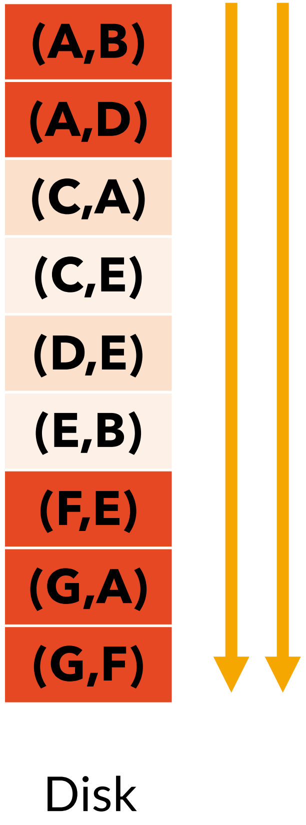


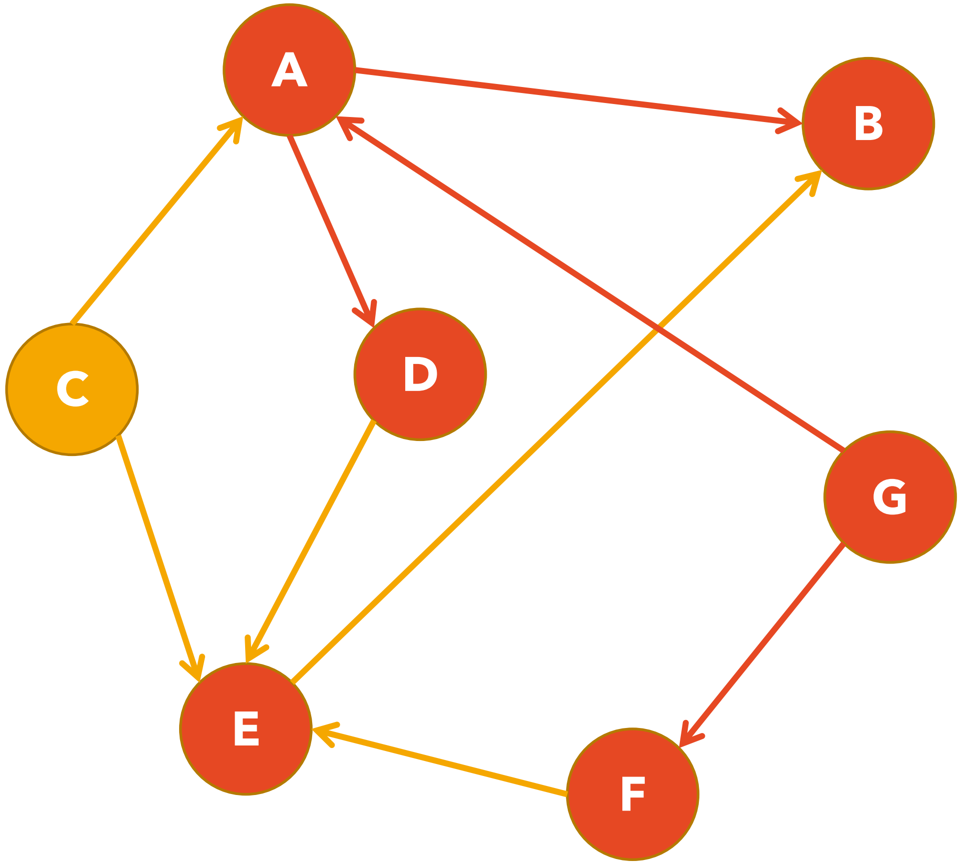
Edge-Centric Scatter-Gather (BFS)



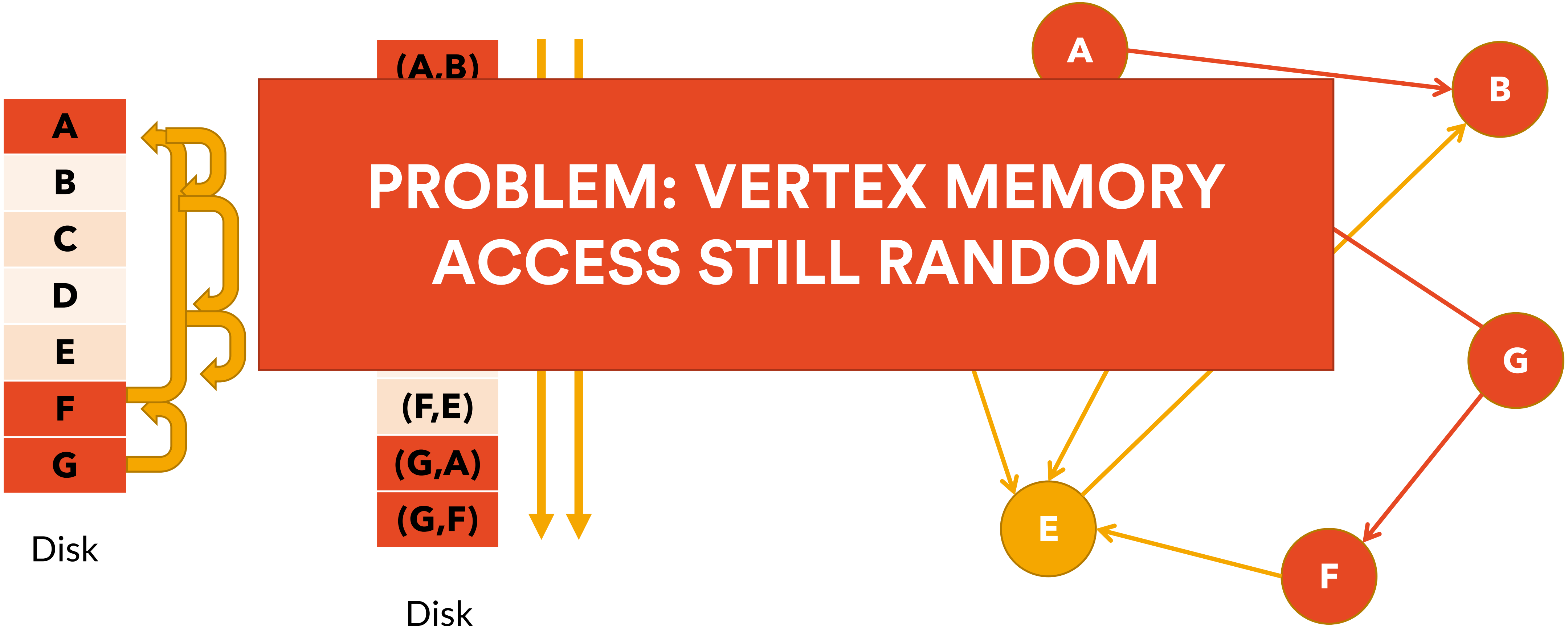
Edge-Centric Scatter-Gather (BFS)





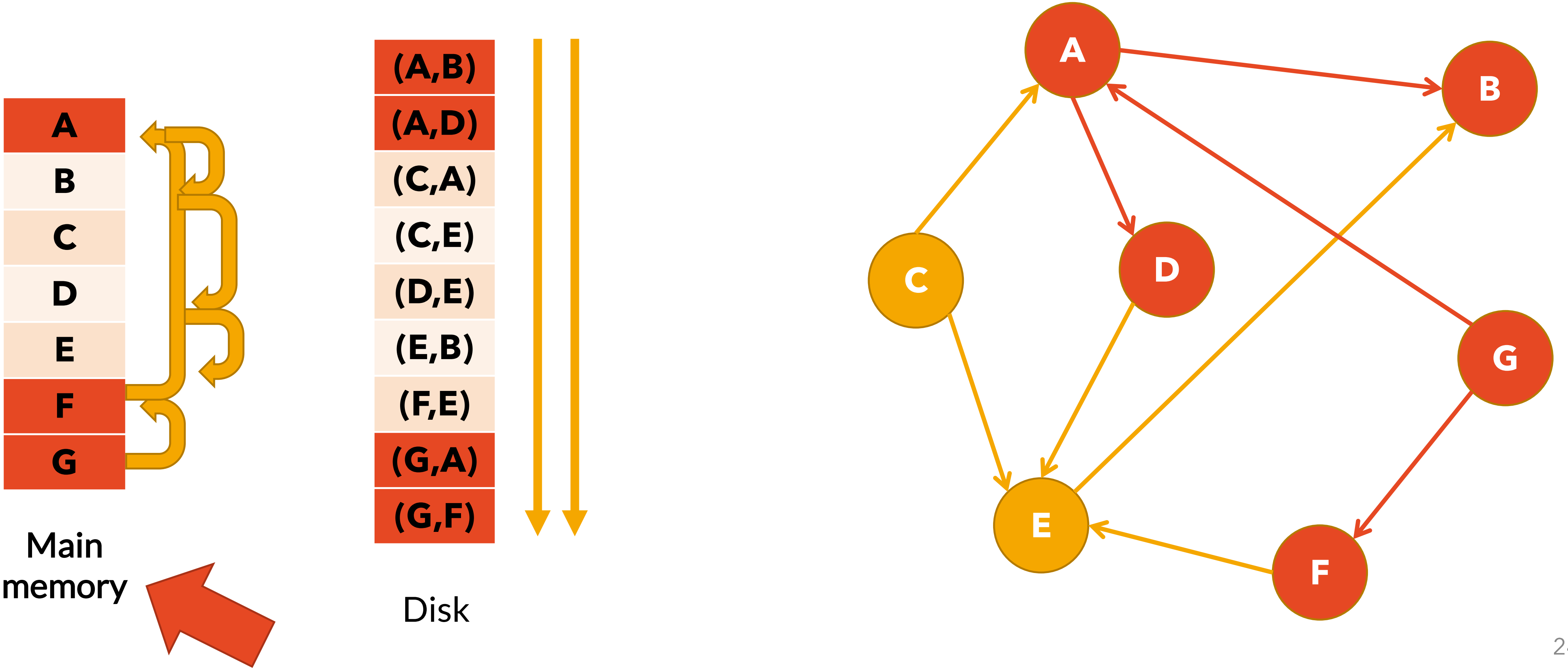


Edge-Centric Scatter-Gather (BFS)

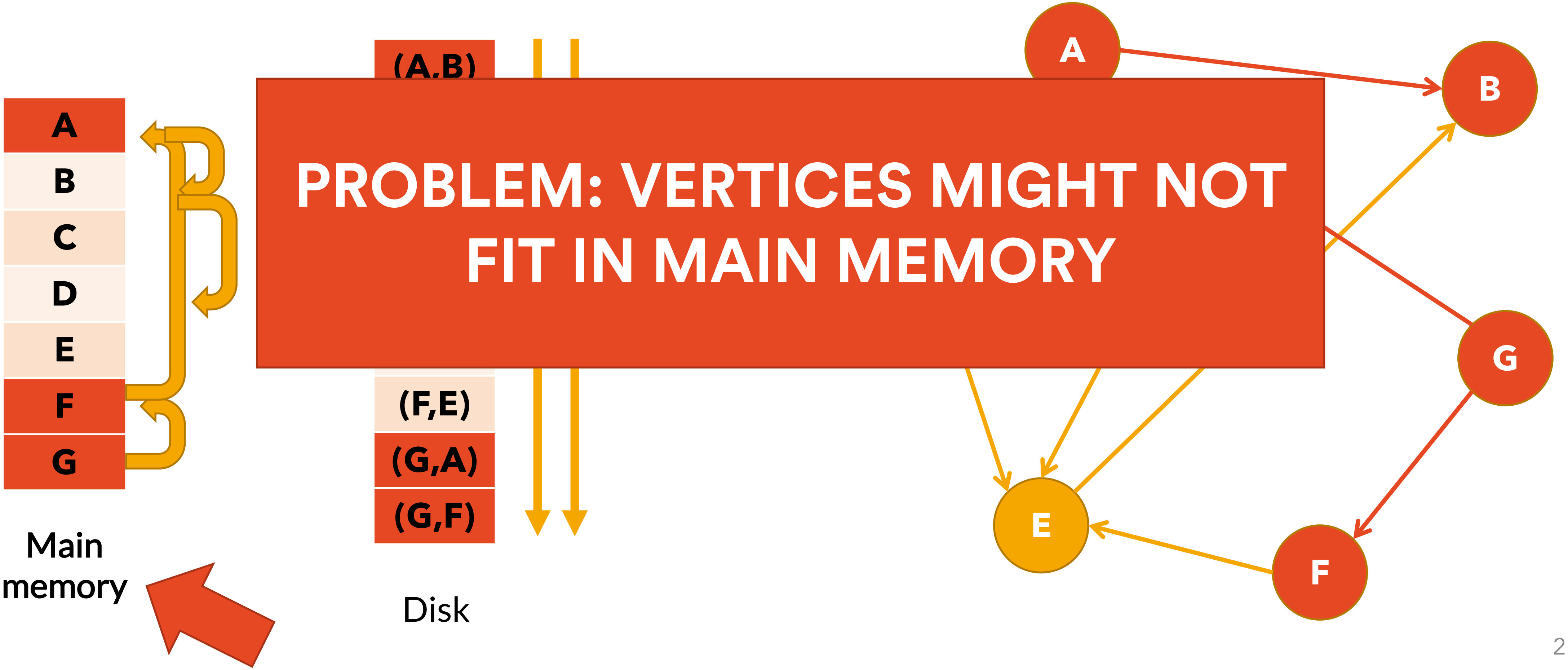


PROBLEM: VERTEX MEMORY ACCESS STILL RANDOM

Edge-Centric Scatter-Gather (BFS)



Edge-Centric Scatter-Gather (BFS)

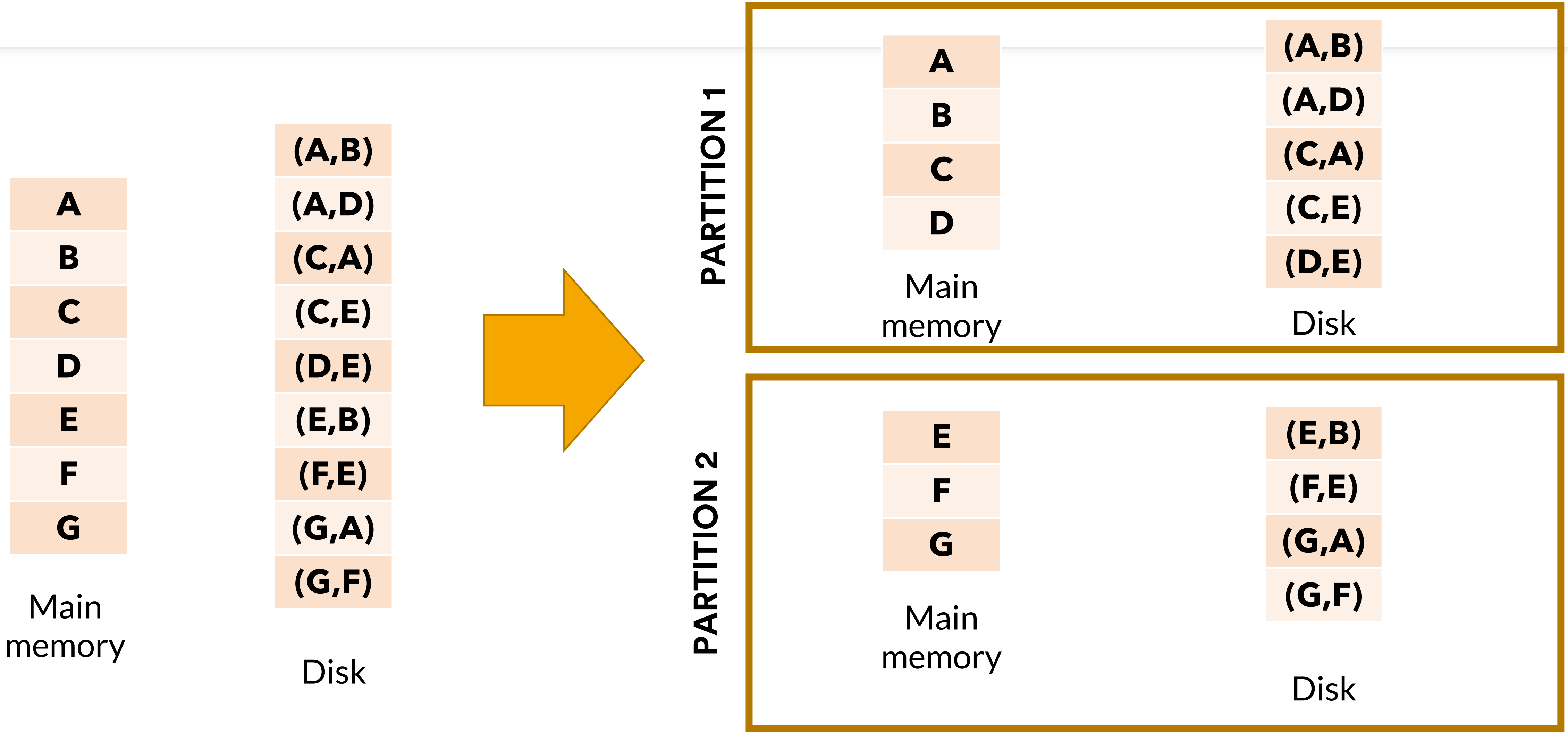


PROBLEM: VERTICES MIGHT NOT FIT IN MAIN MEMORY

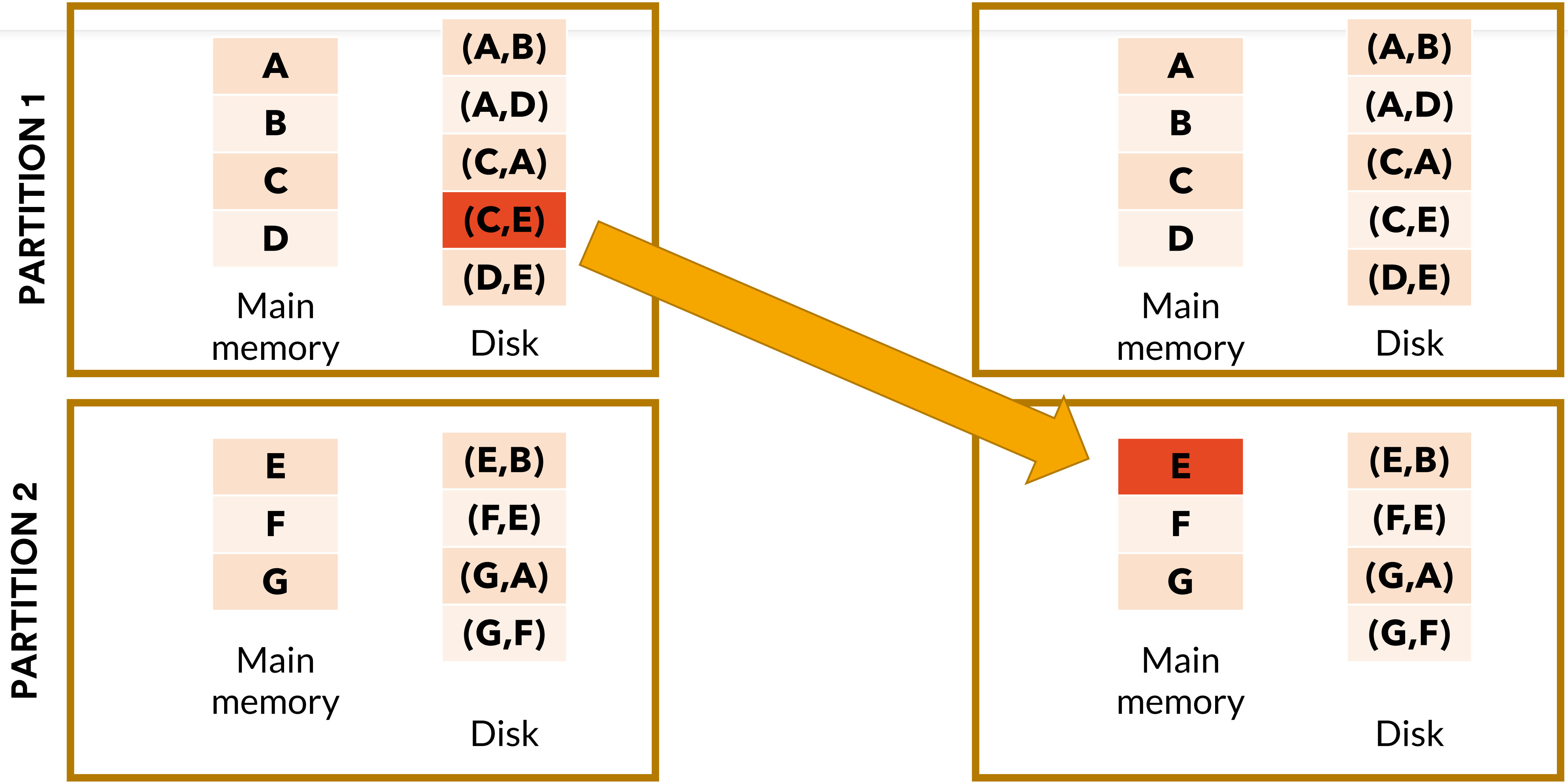
Streaming Partitions

- Split the set of vertices into <u>partitions</u> such that every partition fits in memory.
- A partition also includes all edges whose source vertex is within that partition.

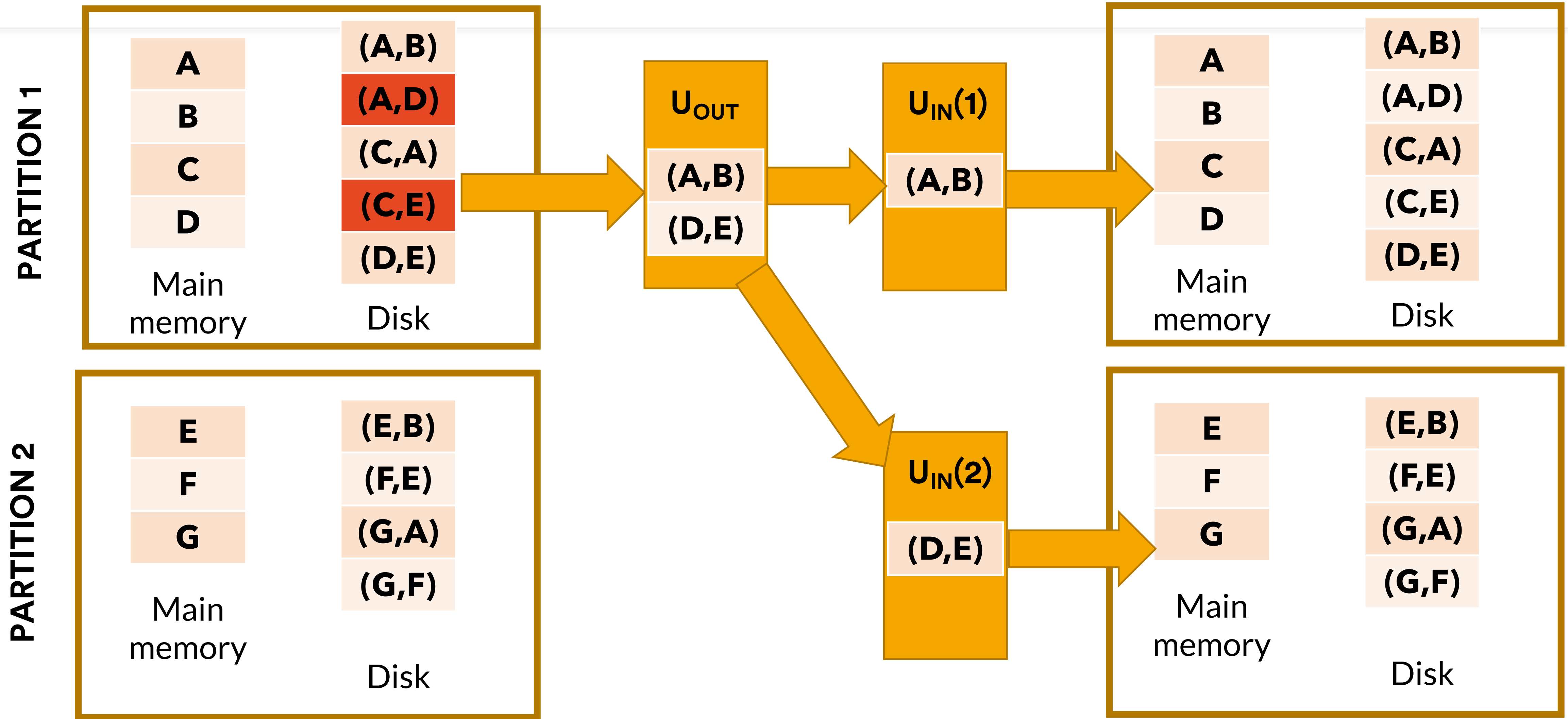
Streaming Partitions



Shuffle Phase



Shuffle Phase



Generalising the Approach

- We showed interaction between:
  - Disk *(Slow Storage)*
  - **Main memory** *(Fast Storage)*
- But the same concept can be applied to
  - Main memory *(Slow Storage)*
  - Cache *(Fast Storage)*
- This allows to apply X-Stream for support both in-memory and out-of-core graphs specifying two engines:
  - Out-of-core Streaming Engine
  - In-memory Streaming Engine

Evaluation Setup

- 1U Server:
  - 64GB main memory
  - 200GB SSD
  - 3TB magnetic disks
- X-Stream evaluated over 10 popular algorithms.
- Both synthetic and real-world datasets.

Algorithmic Performance

	WCC	SCC	SSSP	MCST	MIS	Cond.	SpMV	Pagerank	BP
memory									
amazon0601	0.61s	1.12s	0.83s	0.37s	3.31s	0.07s	0.09s	0.25s	1.38s
cit-Patents	2.98s	0.69s	0.29s	2.35s	3.72s	0.19s	0.19s	0.74s	6.32s
soc-livejournal	7.22s	11.12s	9.60s	7.66s	15.54s	0.78s	0.74s	2.90s	1m 21s
dimacs-usa	**6m 12s**	**9m 54s**	**38m 32s**	4.68s	9.60s	0.26s	0.65s	2.58s	12.01s
ssd									
Friendster	38m 38s	1h 8m 12s	1h 57m 52s	19m 13s	1h 16m 29s	2m 3s	3m 41s	15m 31s	52m 24s
sk-2005	44m 3s	1h 56m 58s	2h 13m 5s	19m 30s	3h 21m 18s	2m 14s	1m 59s	8m 9s	56m 29s
Twitter	19m 19s	35m 23s	32m 25s	10m 17s	47m 43s	1m 40s	1m 29s	6m 12s	42m 52s
disk									
Friendster	1h 17m 18s	2h 29m 39s	3h 53m 44s	43m 19s	2h 39m 16s	4m 25s	7m 42s	32m 16s	1h 57m 36s
sk-2005	1h 30m 3s	4h 40m 49s	4h 41m 26s	39m 12s	7h 1m 21s	4m 45s	4m 12s	17m 22s	2h 24m 28s
Twitter	39m 47s	1h 39m 9s	1h 10m 12s	29m 8s	1h 42m 14s	3m 38s	3m 13s	13m 21s	2h 8m 13s
yahoo-web	—	—	—	—	—	16m 32s	14m 40s	1h 21m 14s	8h 2m 58s

Algorithmic Performance

	WCC	SCC	SSSP	MCST	MIS	Cond.	SpMV	Pagerank	BP
memory									
amazon0601	0.61s	1.12s	0.83s	0.37s	3.31s	0.07s	0.09s	0.25s	1.38s
cit-Patents	2.98s	0.69s	0.29s	2.35s	3.72s	0.19s	0.19s	0.74s	6.32s
soc-livejournal	7.22s	11.12s	9.60s	7.66s	15.54s	0.78s	0.74s	2.90s	1m 21s
dimacs-usa	**6m 12s**	**9m 54s**	**38m 32s**	4.68s	9.60s	0.26s	0.65s	2.58s	12.01s
ssd									
Friendster	38m 38s	1h 8m 12s	1h 57m 52s	19m 13s	1h 16m 29s	2m 3s	3m 41s	15m 31s	52m 24s
sk-2005	44m 3s	1h 56m 58s	2h 13m 5s	19m 30s	3h 21m 18s	2m 14s	1m 59s	8m 9s	56m 29s
Twitter	19m 19s	35m 23s	32m 25s	10m 17s	47m 43s	1m 40s	1m 29s	6m 12s	42m 52s
disk									
Friendster	1h 17m 18s	2h 29m 39s	3h 53m 44s	43m 19s	2h 39m 16s	4m 25s	7m 42s	32m 16s	1h 57m 36s
sk-2005	1h 30m 3s	4h 40m 49s	4h 41m 26s	39m 12s	7h 1m 21s	4m 45s	4m 12s	17m 22s	2h 24m 28s
Twitter	39m 47s	1h 39m 9s	1h 10m 12s	29m 8s	1h 42m 14s	3m 38s	3m 13s	13m 21s	2h 8m 13s
yahoo-web	—	—	—	—	—	16m 32s	14m 40s	1h 21m 14s	8h 2m 58s

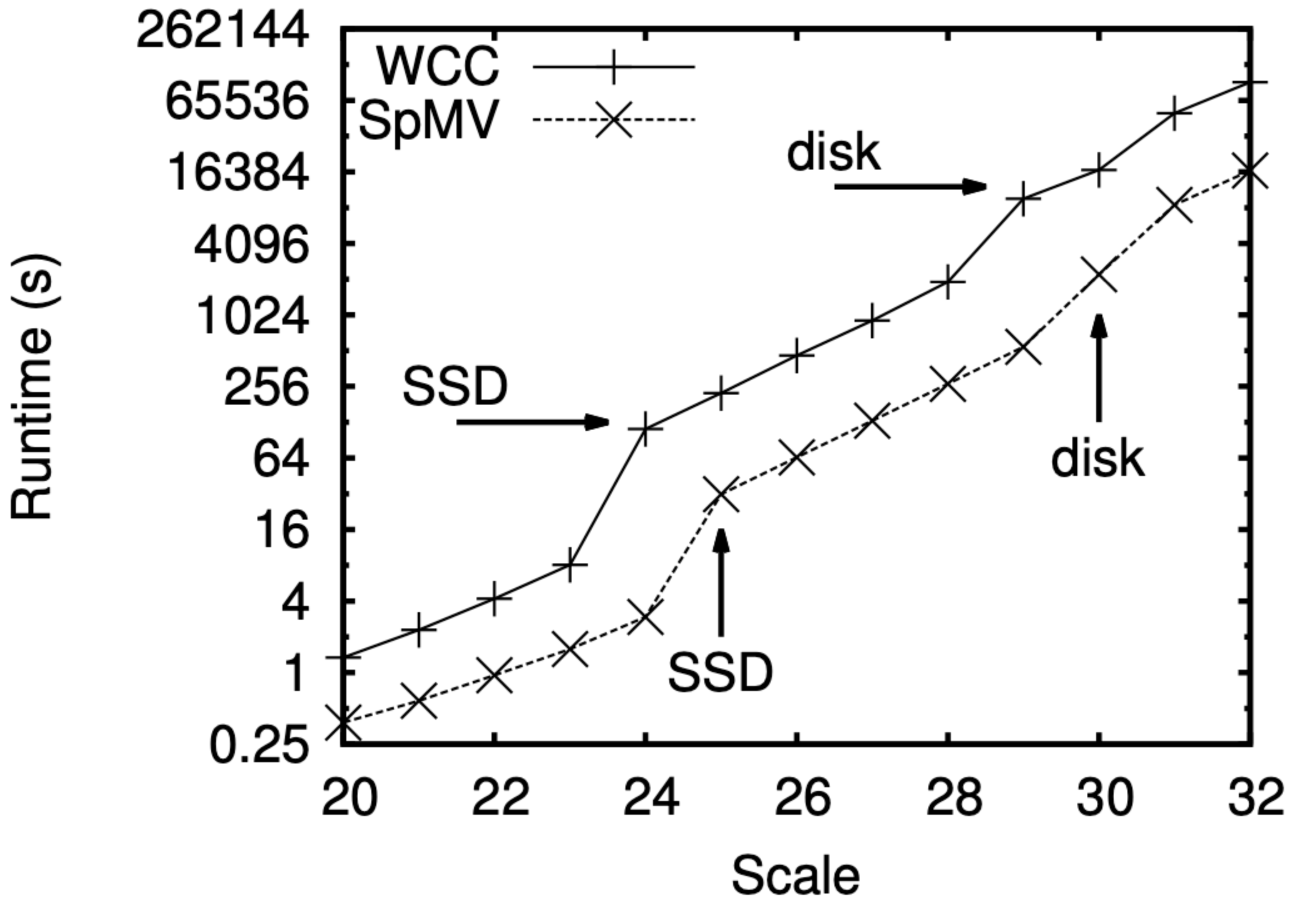
Algorithmic Performance

Graph	# steps
In-memory	
amazon0601	19
cit-Patents	20
soc-livejournal	15
dimacs-usa	**8122**
Out-of-core	
sk-2005	28
yahoo-web	**over 155**

Number of Steps Taken to Cover the Graph by HyperANF

- HyperANF measures the *neighbourhood function* of the graph.
- High numbers indicate those graphs have a **large diameter**.
- They take many scatter-gather iterations to complete, which is why X-Stream performs poorly on them.

Scalability

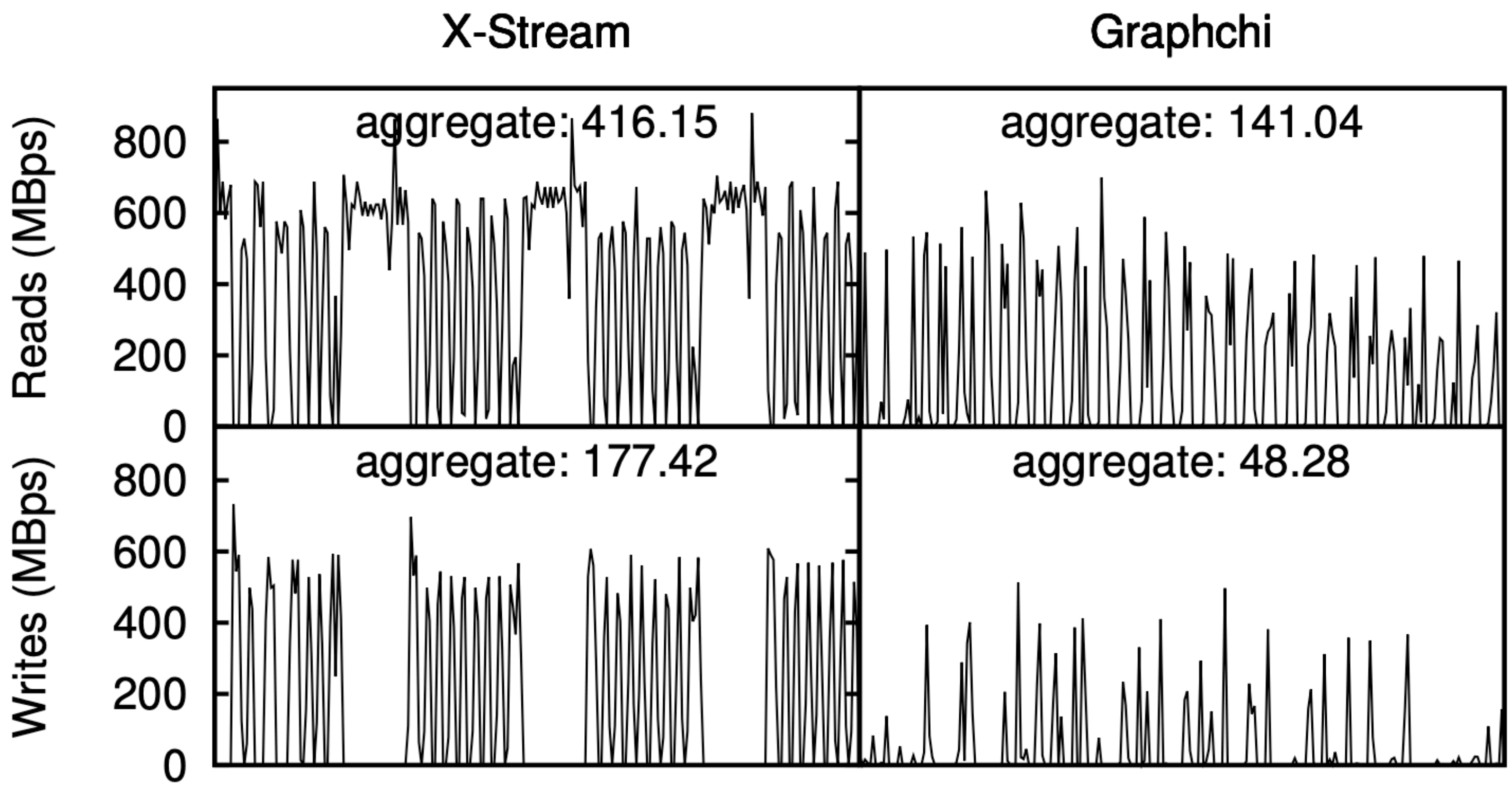


X-Stream vs Graphchi

	Pre-Sort (s)	Runtime (s)	Re-sort (s)
Twitter pagerank			
X-Stream (1)	*none*	397.57 ± 1.83	–
Graphchi (32)	752.32 ± 9.07	1175.12 ± 25.62	969.99
Netflix ALS			
X-Stream (1)	*none*	76.74 ± 0.16	–
Graphchi (14)	123.73 ± 4.06	138.68 ± 26.13	45.02
RMAT27 WCC			
X-Stream (1)	*none*	867.59 ± 2.35	–
Graphchi (24)	2149.38 ± 41.35	2823.99 ± 704.99	1727.01
Twitter belief prop.			
X-Stream (1)	*none*	2665.64 ± 6.90	–
Graphchi (17)	742.42 ± 13.50	4589.52 ± 322.28	1717.50

- X-Stream outperforms Graphchi:
  - No pre-processing cost.
  - X-Stream makes a better use of the SSD, constantly maintaining high bandwidth.

X-Stream vs Graphchi



- X-Stream outperforms Graphchi:
  - No pre-processing cost.
  - X-Stream makes a better use of the SSD, constantly maintaining high bandwidth.

Review

- X-Stream achieved impressive results against existing solutions.
- Points attention to the trade-off between number and cost of memory accesses.
- On the downside, X-Stream’s performance is heavily dependant on the characteristics of the underlying graph.

Impact

- Chaos is the next generation of X-Stream.
- A couple of studies have adopted X-Stream's edge-centric approach:
  - "An FPGA framework for edge-centric graph processing" (S. Zhou et al.)
  - "WolfGraph: The edge-centric graph processing on GPU" (H. Zhu et al.)
- However, the application of edge-centric frameworks seems to be restricted to academia.

Thank you for the attention!