Tigr: Transforming Irregular Graphs for GPU-Friendly Graph Processing

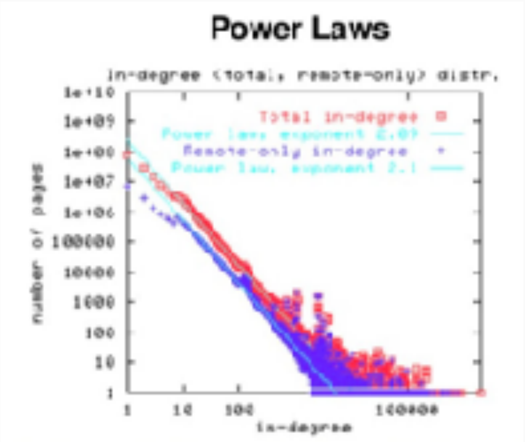
Sadet, Qiu, Zhao (2018)

Presented by Edward Fan

Real-world graphs are irregular

- Power-law or not, large imbalances in vertex degree are common

- Lots of frameworks that we've seen have worked around this
    - Partitioning the graph by edges
    - Splitting work into sub-tasks
    - Duplicating vertices across nodes



Real-world graphs are irregular

- For GPUs, the issue is even more important
- Threads are organized in *warps*
    - A warp is SIMD-like- a group of threads execute the same instructions in parallel over different data
    - There is implicit synchronization- all threads must complete before execution can continue
    - If vertices are assigned to threads, this can lead to a major bottleneck

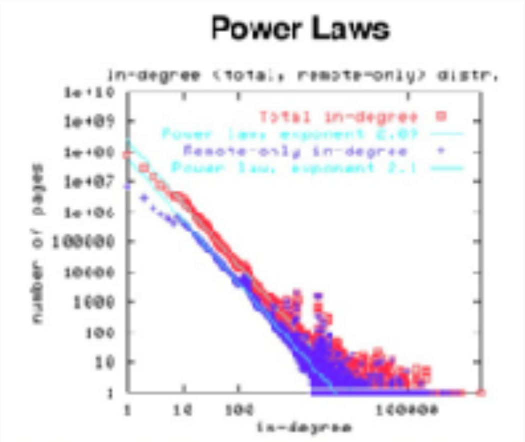


Table 6. Performance Details (SSSP, LiveJournal, K = 8).

	Without Worklist				With Worklist		
	#iter	time / iter.	#instr.	warp eff.	#iter	#instr.	warp eff.
Original	14	29.52	5.3 ×10[illegible]	25.98%	18	9 ×10[illegible]	60.53%
Physical	29	36.68	6.9 ×10[illegible]	91.15%	45	4.6 ×10[illegible]	70.11%
Virtual	14	17.64	7.5 ×10[illegible]	92.81%	13	2.2 ×10[illegible]	85.51%

Tigr: vertex transformations

- Basic idea: split a high degree node into multiple nodes
- Many possible connections, but tradeoffs between propagation time, edge count, and vertex count
- Solution: *uniform-degree tree transformation* (UDT)

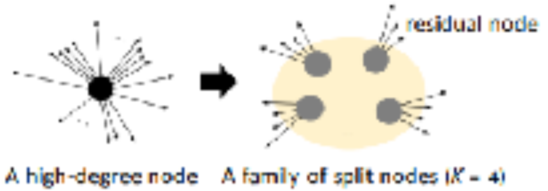


Figure 4. Illustration of Split Transformation.

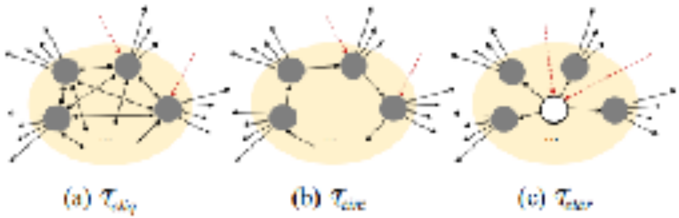


Figure 5. Three Example Connections.

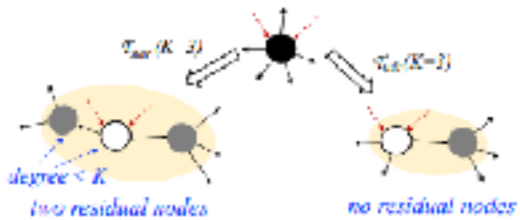


Figure 6. Comparison between $\mathcal{T}_{star}$ and $\mathcal{T}_{udt}$.

Correctness of split nodes

- Many algorithms still work correctly on the split nodes
- Can assign internal edges zero weight (or equivalent)
  - Just need to treat split vertex as if it were a single vertex
- Total in/out-degree does not change, so PageRank still works
  - Sum the split nodes

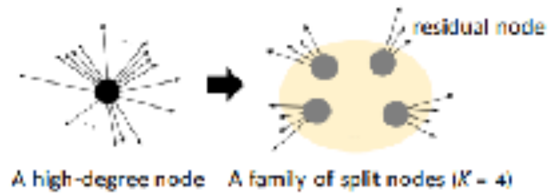


Figure 4. Illustration of Split Transformation.

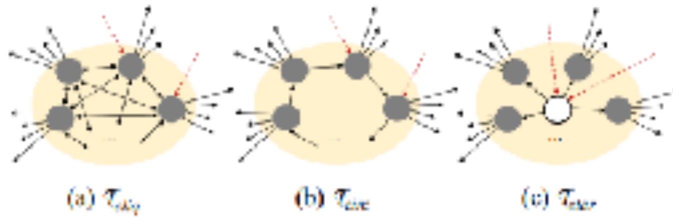


Figure 5. Three Example Connections.

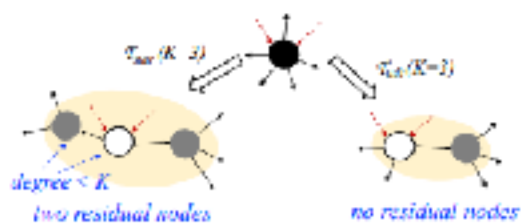


Figure 6. Comparison between $\mathcal{T}_{star}$ and $\mathcal{T}_{ndt}$.

Virtual transformation

- Benefits of vertex splitting, without having to synchronize!
- Slightly modified CSR serves this purpose well

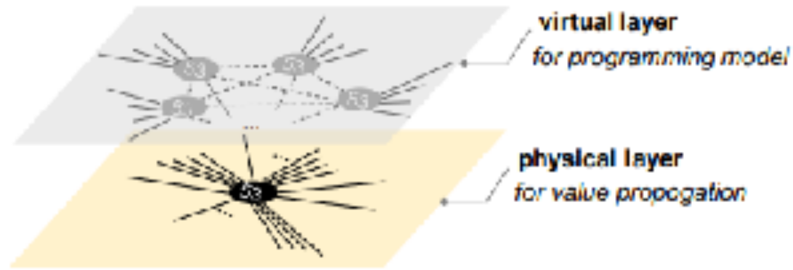


Figure 9. Illustration of Virtual Split Transformation.

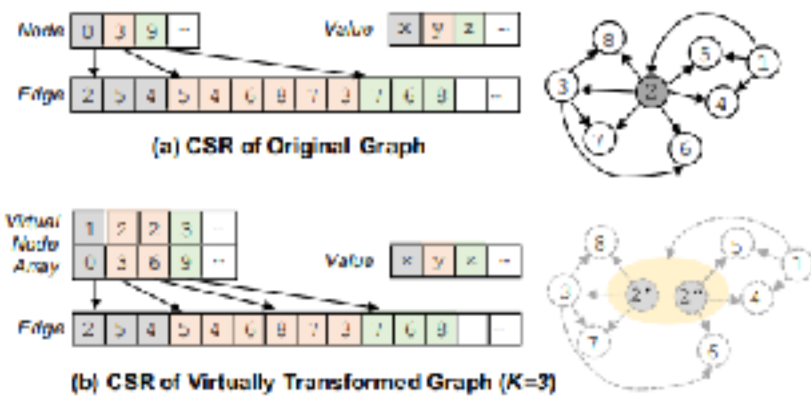


Figure 10. Integrating Virtual Node Array into CSR Format.

Virtualization details

- Use atomic operations to resolve concurrent writes
  - For push execution: no correctness difference (as every vertex needs to be able to accept concurrent pushes)
  - For pull execution: writes must be associative, as neighbors are processed in parallel

- Clever trick for GPU coalescing
  - A normal split would result in strided accesses
  - Solution: *stride the split!*
  - Leads to sequential access

Figure 12. Edge-array Coalescing.

Performance

- On single GPU, beats CuSha and Gunrock most of the time
- Caveat: custom code vs framework limitations

Table 4. Performance Comparison.

execution time: *ms*; the best performance is bolded

Alg.	Dataset	MW	CuSha	Gunrock	Tigr-V+
BFS	pokec	90.32	21.73	28.23	**14.64**
BFS	LiveJournal	149.6	57.62	51.47	**27.76**
BFS	hollywood	89.4	142.26	24.54	**15.9**
BFS	orkut	276.13	129.93	227.83	**77.73**
BFS	twitter	1514.44	1060.85	348.06	**178.53**
BFS	sinaweibo	1160.01	OOM	OOM	**299.24**
SSSP	pokec	94.37	44.49	73.34	**40.77**
SSSP	LiveJournal	228.39	115	127.54	**62.21**
SSSP	hollywood	180.15	331.46	85.49	**44.84**
SSSP	orkut	538.99	279.13	452.89	**159.85**
SSSP	twitter	1570.21	OOM	538.47	**269.75**
SSSP	sinaweibo	1529.09	OOM	1297.46	**699.55**
PR	pokec	20.81	**2.06**	30.67	22.1
PR	LiveJournal	30.63	**4.61**	33.04	34.25
PR	hollywood	16.73	20.35	**12.7**	15.09
PR	orkut	135.65	**16.59**	171.7	156.32
PR	twitter	**216.21**	OOM	243.07	221.49
PR	sinaweibo	445.8	OOM	**441.02**	463.66
CC	pokec	54.94	**17.54**	37.44	42.32
CC	LiveJournal	133.98	49.42	59.54	**47.4**
CC	hollywood	71.08	98.87	89.36	**21.38**
CC	orkut	221.67	**132.37**	170.51	207.93
CC	twitter	1427.73	979.03	685.89	**573.53**
CC	sinaweibo	928.45	OOM	772.52	**579.13**
SSWP	pokec	111.48	52.29	-	**36.86**
SSWP	LiveJournal	333.08	163.38	-	**63.67**
SSWP	hollywood	141.2	239.13	-	**22.63**
SSWP	orkut	479.12	211.38	-	**121.48**
SSWP	twitter	1546.68	OOM		**240.48**
SSWP	sinaweibo	1527.14	OOM	-	**635.23**
BC	pokec	-	-	87.09	**42.86**
BC	LiveJournal	-	-	109.56	**73.61**
BC	hollywood	-	-	55.77	**39.21**
BC	orkut	-	-	399.96	**207.58**
BC	twitter	-	-	732.28	**475.23**
BC	sinaweibo	-	-	1507.25	**1033.97**

Questions?

Thanks!