Scalable Data-driven PageRank: Algorithms, System Issues, and Lessons Learned

Joyce Jiyoung Whang
Dept. of Computer Science & Engineering
SKKU

February 24, 2017

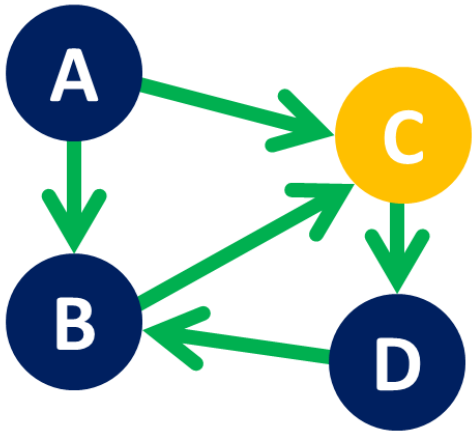
Parallel Graph Mining Algorithms

- Work Activation
  - Topology-driven
  - Data-driven
- Data Access Pattern
  - Pull (Read)
  - Pull-Push (Read-Write)
  - Push (Write)
- Scheduling
  - FIFO/LIFO
  - Task-specific priority scheduling

Scalable Data-driven PageRank[1]

- PageRank: the importance of web pages
  - Given a graph $G = (\mathcal{V}, \mathcal{E})$, a PageRank vector $\mathbf{x}$
  - $\mathcal{S}_v$: the set of incoming neighbors of node v
  - $\mathcal{T}_v$: the set of outgoing neighbors of node v

$$x_v^{(k+1)} = \alpha \sum_{w \in \mathcal{S}_v} \frac{x_w^{(k)}}{|\mathcal{T}_w|} + (1 - \alpha)$$



$$\mathrm{C} = \alpha\left(\frac{A}{2} + \mathrm{B}\right) + (1 - \alpha)$$

[1] J. Whang, A. Lenharth, I. Dhillon, and K. Pingali. Scalable Data-driven PageRank: Algorithms, System Issues, and Lessons Learned. *Euro-Par*, 2015.

Topology-driven PageRank

- Power Method: processing all the nodes at each iteration
  - Work activation: topology-driven
  - Data access pattern: pull (read-mostly access pattern)
  - Scheduling: unordered

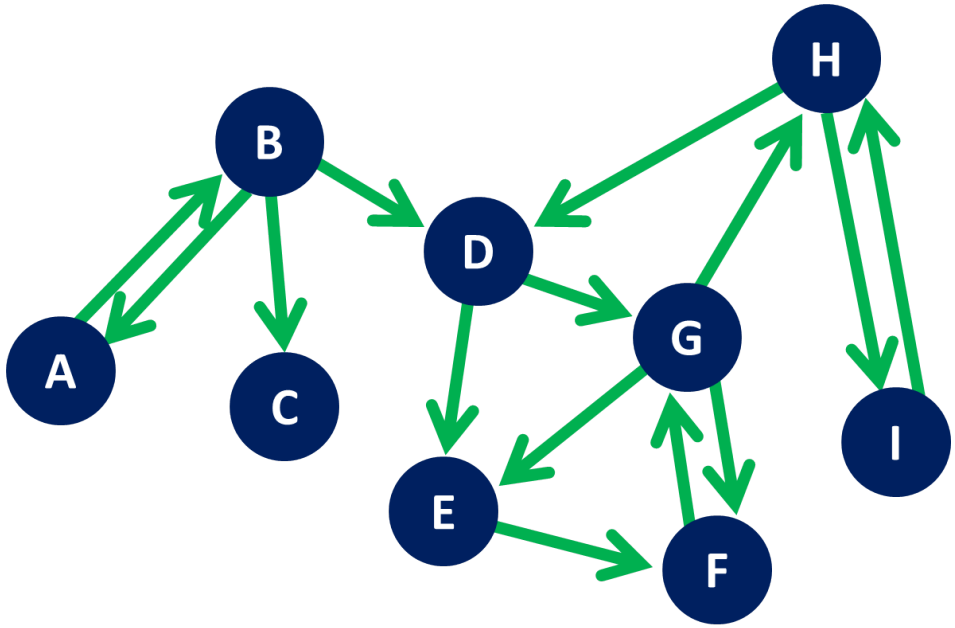


Figure 1: Topology-driven PageRank updates the PageRank values of all the nodes at each iteration.

Topology-driven to Data-driven

- Data-driven PageRank
  - Dynamically maintaining a working set
  - Only compute values which might change
  - Order matters

- On a Twitter graph (51 million nodes, 3 billion edges)
  - Topology-driven: 4.3 billion PageRank updates
  - Data-driven (basic): 1.6 billion PageRank updates
  - Data-driven (best): 0.4 billion PageRank updates

Data-driven PageRank

- Data-driven PageRank
  - Initialize the worklist: the entire vertex set.
  - Pick a node from the worklist, and compute the nodes PageRank.
  - Add its outgoing neighbors to the worklist.

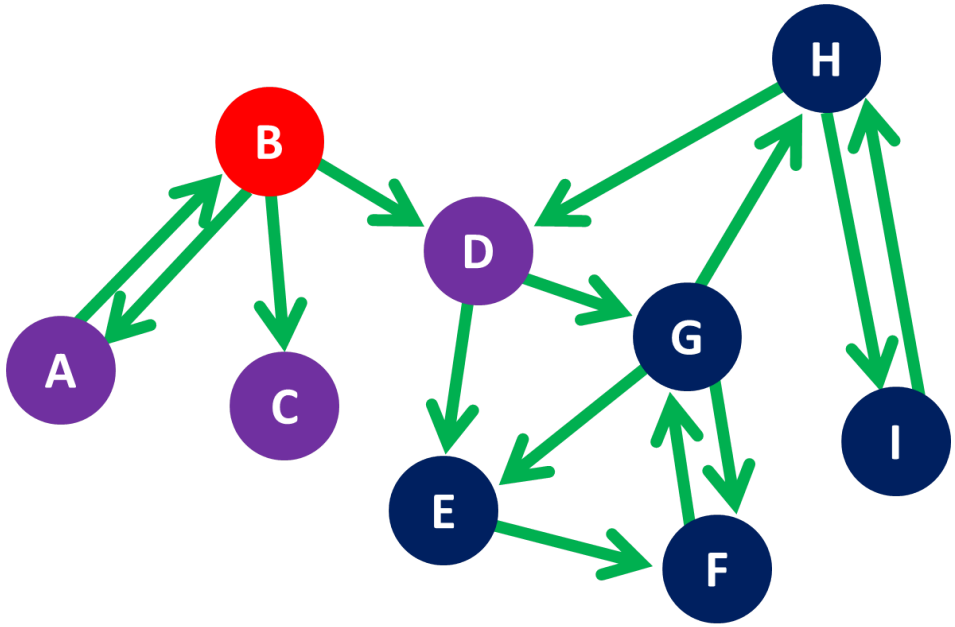


Figure 2: When B's PageRank is updated, A, C, and D should be updated.

Data-driven PageRank

- Data-driven PageRank
  - Initialize the worklist: the entire vertex set.
  - Pick a node from the worklist, and compute the nodes PageRank.
  - Add its outgoing neighbors to the worklist.

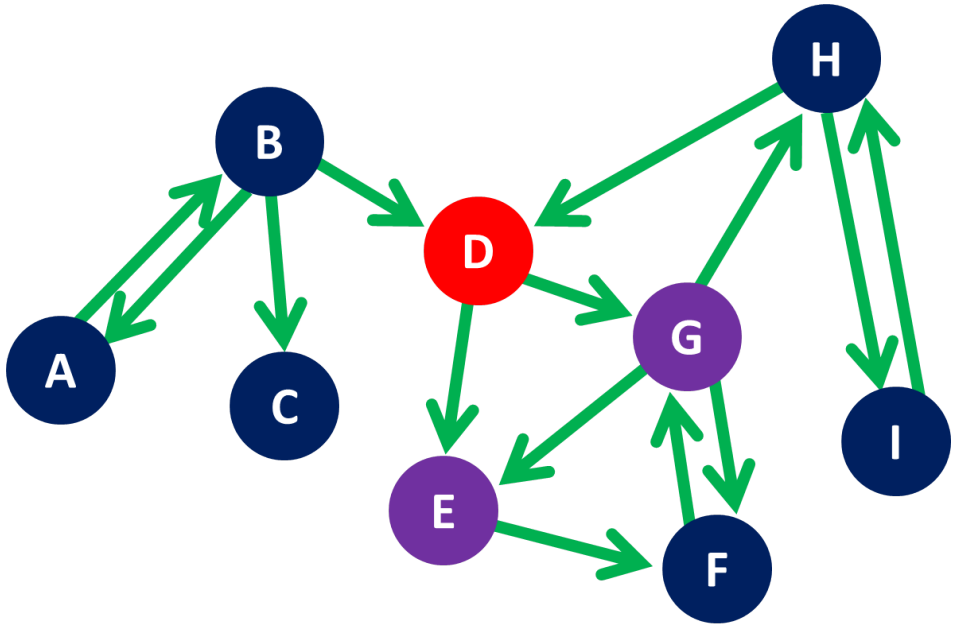


Figure 3: When D's PageRank is updated, E and G should be updated.

Convergence of Data-driven PageRank

The PageRank $\mathbf{x}$ is computed as follows:

$$\mathbf{x} = \alpha \boldsymbol{P}^T \mathbf{x} + (1-\alpha)\mathbf{e}$$

where $\boldsymbol{P}$ is a row-stochastic matrix ($\boldsymbol{P} = \boldsymbol{D}^{-1}\boldsymbol{A}$) and $\mathbf{e}$ is the vector of all ones. This is the linear system:

$$(\boldsymbol{I} - \alpha \boldsymbol{P}^T)\mathbf{x} = (1-\alpha)\mathbf{e}.$$

and the residual:

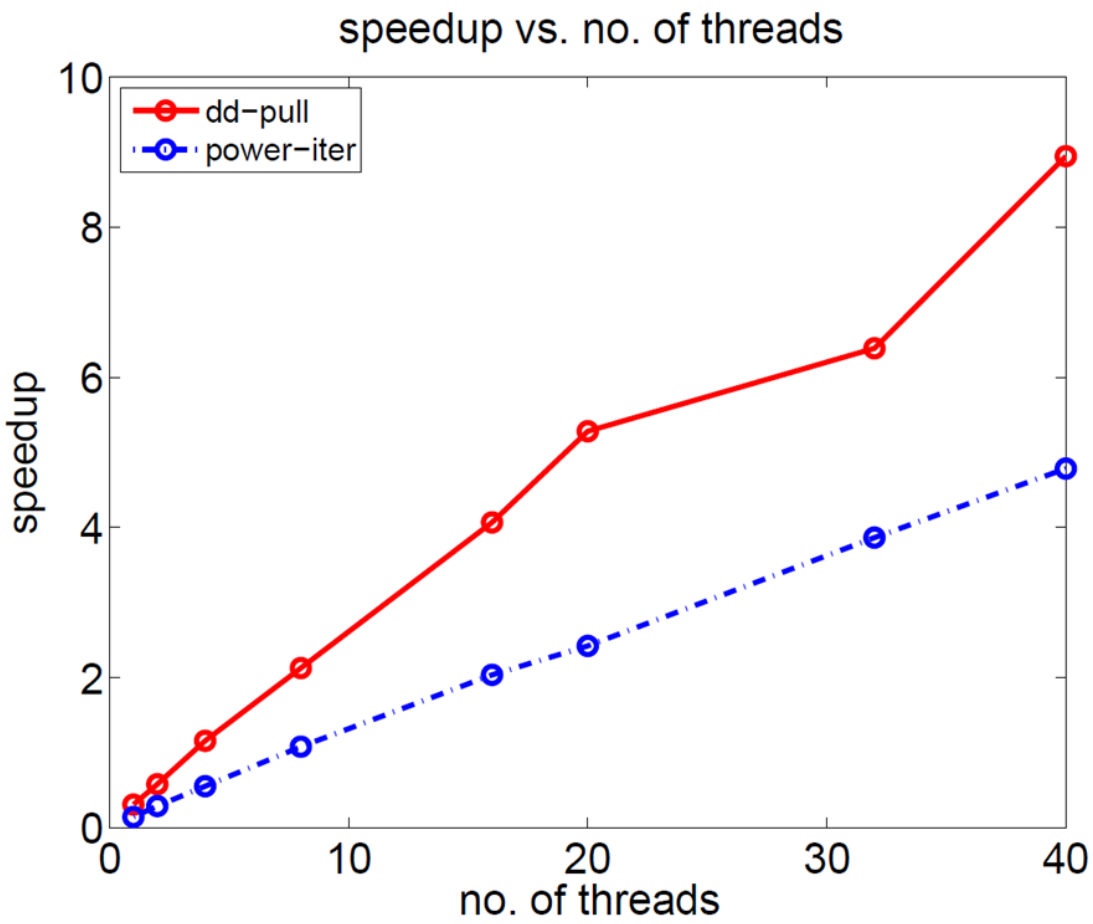
$$\mathbf{r} = (1-\alpha)\mathbf{e} - (\boldsymbol{I} - \alpha \boldsymbol{P}^T)\mathbf{x} = \alpha \boldsymbol{P}^T \mathbf{x} + (1-\alpha)\mathbf{e} - \mathbf{x}.$$

When the j-th node is processed, the residual is decreased by $\mathbf{r}_j^{(k)}(1-\alpha)$.

$$\mathbf{e}^T \mathbf{r}^{(k+1)} = \mathbf{e}^T \mathbf{r}^{(k)} - \mathbf{r}_j^{(k)}(1-\alpha).$$

Topology-driven vs. Data-driven PageRank

- Speedup against the best single-threaded algorithm (Twitter graph)



Work activation:
Data-driven
Data access pattern:
Pull-based
Schedule:
FIFO

Work activation:
Topology-driven
Data access pattern:
Pull-based
Schedule:
Unordered

Pull to Pull-Push

- The basic pull-based data-driven PageRank
  - Each node adds all the out-neighbors to the worklist.
- Think PageRank as a linear system
  - Each node maintains its residual value.
  - Stopping criteria: the maximum residual $<$ threshold
- Pull-Push PageRank
  - Residual computation:
    Read/write operations on the neighbors
  - If an outgoing-neighbor's residual $<$ threshold,
    then, do not add the neighbor to the worklist.
    $\Rightarrow$ Filter out some work in the worklist.

Pull-Push-based PageRank

- Pull-Push-based PageRank
  - **Pulls** (reads) its neighbors' values.
  - **Pushes** (updates) its neighbors' values.

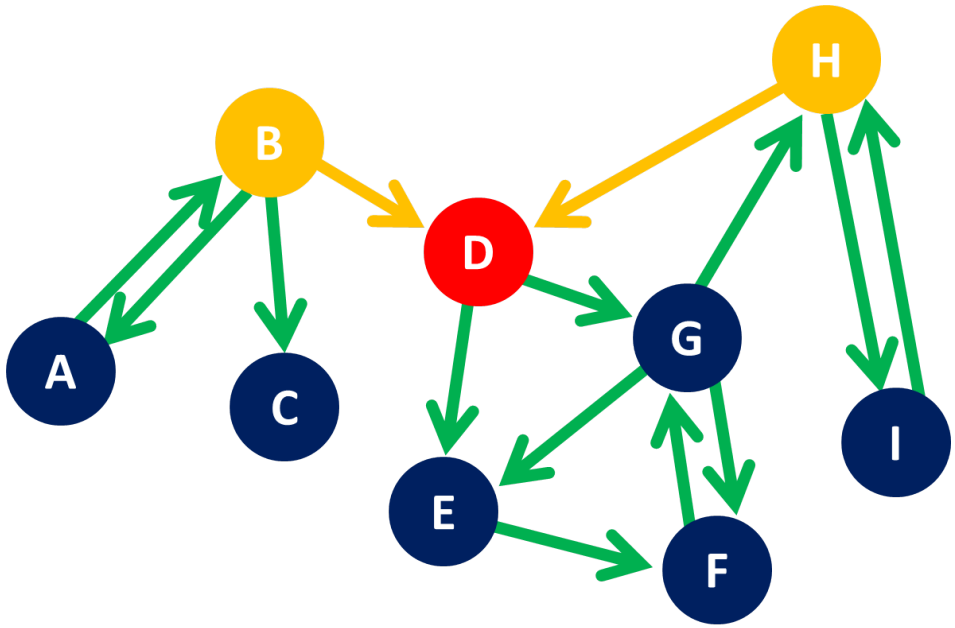


Figure 4: D's PageRank is updated by reading B's and H's PageRank.

Pull-Push-based PageRank

- Pull-Push-based PageRank
  - Stopping criteria: repeat until the residual is less than a threshold.
  - X's PageRank $\rightarrow$ Adds residuals to X's outgoing neighbors.

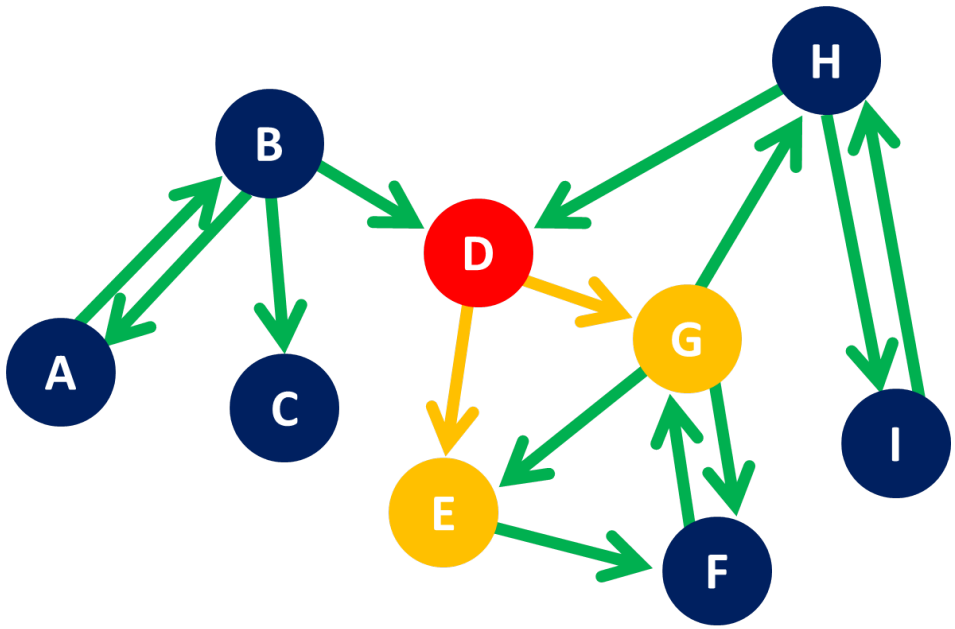


Figure 5: If D's PageRank is updated, it adds the residuals to E and G.

Pull-Push-based PageRank

- Pull-Push-based PageRank
  - X's residual is less than the threshold $\rightarrow$ do not add it to the worklist.
  - Filter out work in the worklist.

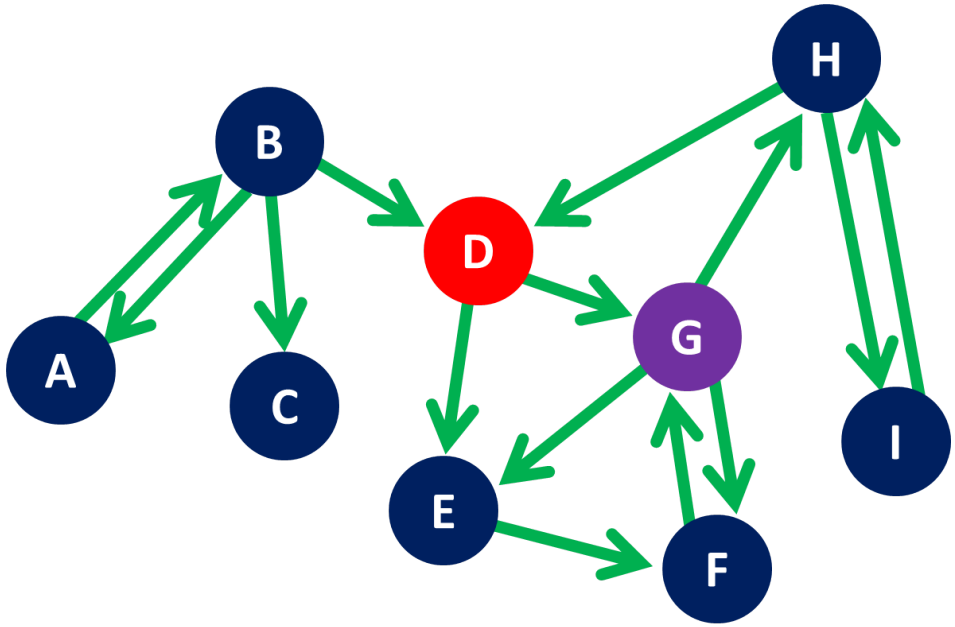
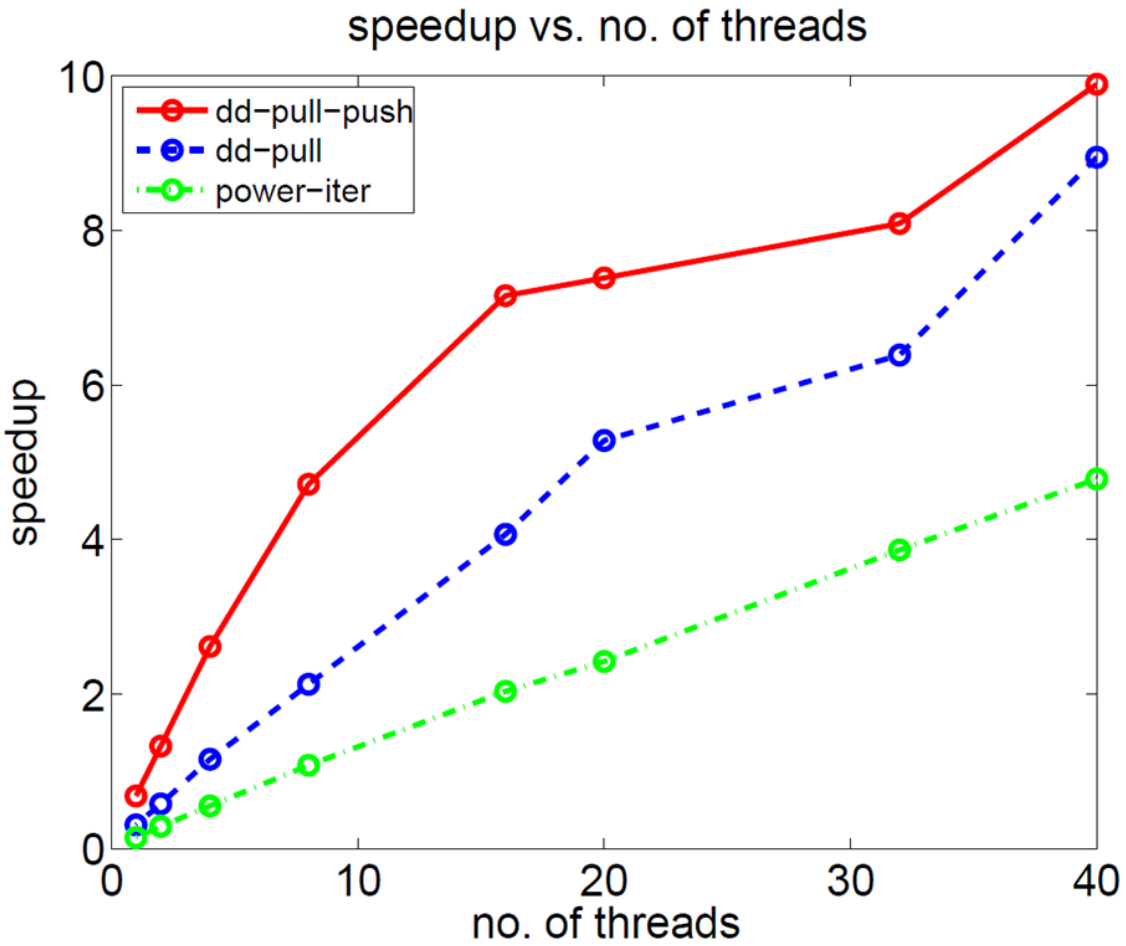


Figure 6: If E's residual is less than a threshold, it is not added to the worklist.

Pull-Push vs. Pull Data-driven vs. Topology-driven

- Speedup against the best single-threaded algorithm (Twitter graph)



Work activation: Data-driven
Data access pattern: Pull-Push
Schedule: FIFO

Work activation: Data-driven
Data access pattern: Pull
Schedule: FIFO

Work activation: Topology
Data access pattern: Pull
Schedule: Unordered

Pull-Push to Push

- Exploiting the problem structure of PageRank

  $x_v^{(k+1)} = x_v^{(k)} + r_v^{(k)}$

  (new PageRank = current PageRank + current residual)

- Push-based PageRank
  - From pull-push PageRank, we can remove the read operations on the incoming neighbors.

    $\Rightarrow$ Avoid extra work (the pull part)

Push-based PageRank

- Push-based PageRank
  - An active node updates its own value.
  - **Pushes** (updates) its neighbors' values.

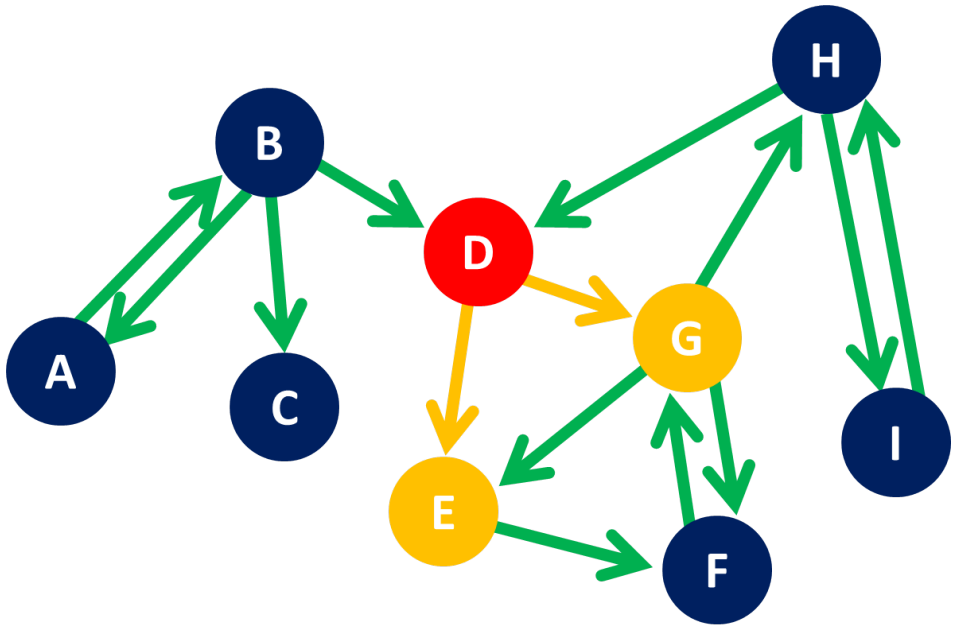
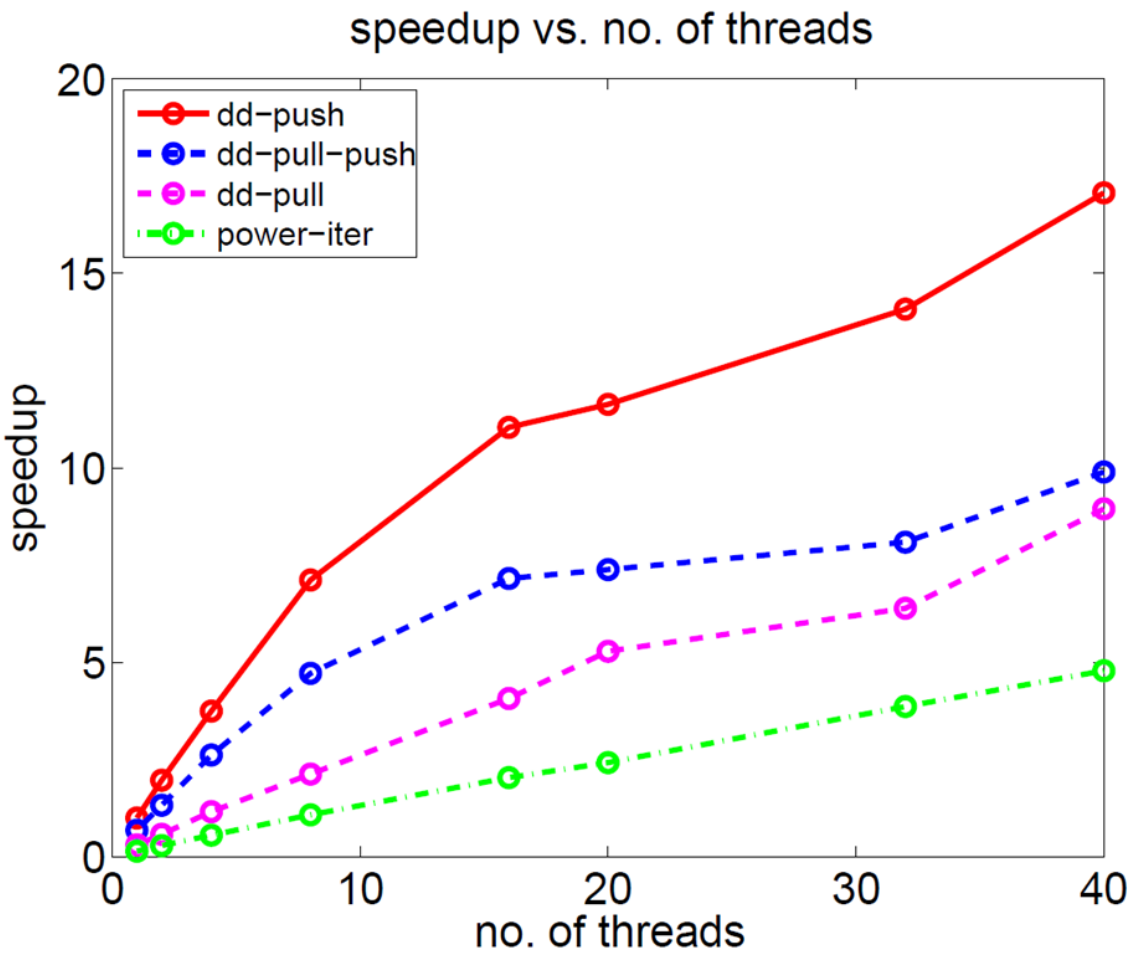


Figure 7: If D's PageRank is updated, it pushes the residuals to E and G.

Pull vs. Pull-Push vs. Push PageRank

- Speedup against the best single-threaded algorithm (Twitter graph)



Work activation: Data-driven
Data access pattern: Push
Schedule: FIFO

Work activation: Data-driven
Data access pattern: Pull-Push
Schedule: FIFO

Work activation: Data-driven
Data access pattern: Pull
Schedule: FIFO

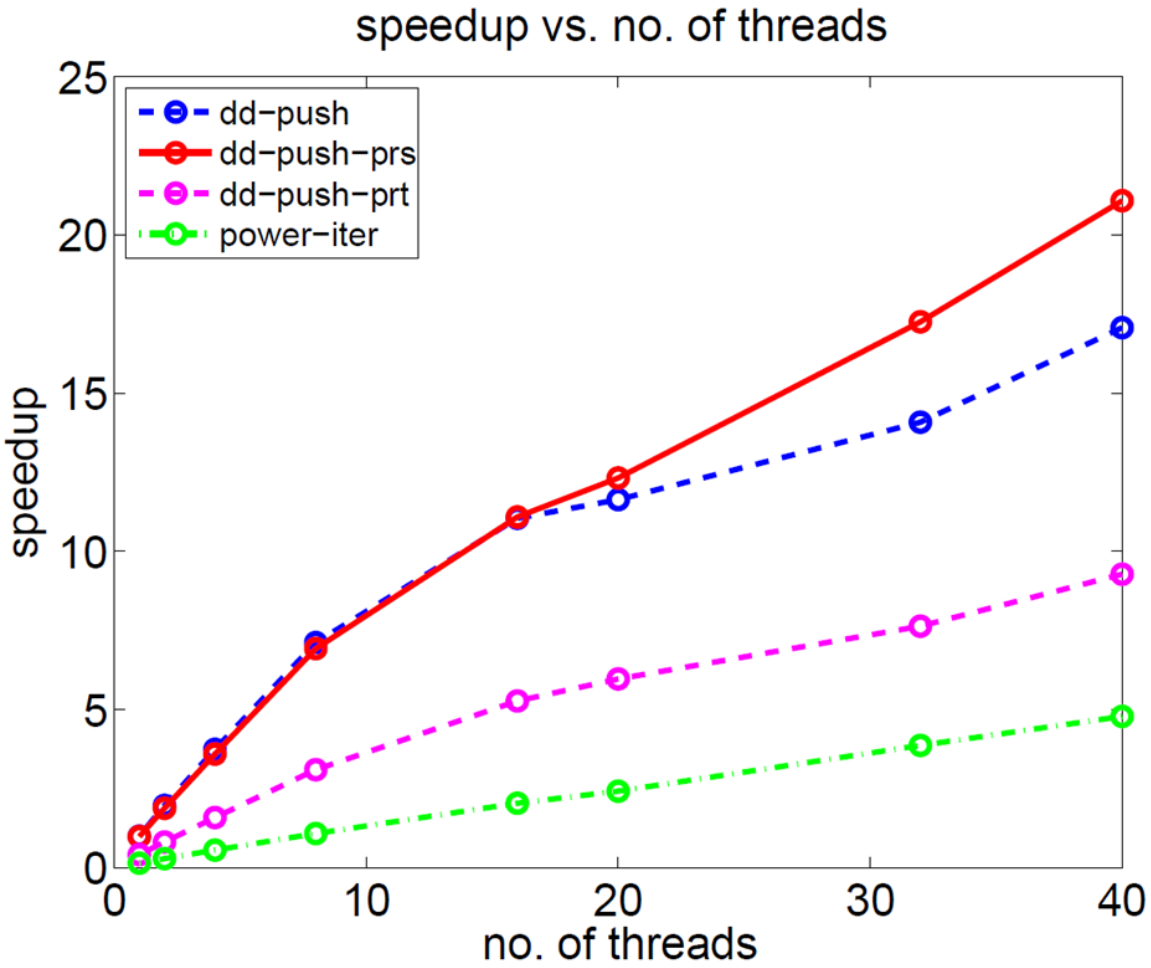
Work activation: Topology
Data access pattern: Pull
Schedule: Unordered

Scheduling

- Order of visiting nodes affects performance.
  - The decrease of the total residual is proportional to the node's residual.
    $\Rightarrow$ Define the priority as the residual per unit work.
- Asynchronous, Autonomous Scheduler
  - OBIM (ordered-by-integer-metric) priority scheduler
  - Each thread uses limited communication to estimate the priority of the high priority work.
  - Potentially generating duplicate tasks in the scheduler
- Bulk Synchronous Scheduler
  - Set-semantics: there are no duplicate work items in the worklist.
  - Periodically, compute the distribution of priorities.
  - Use the distribution to define a threshold for the priority.
  - Process a subset of nodes whose priority is greater than the threshold.

Data-driven PageRank with Different Scheduling

- Speedup against the best single-threaded algorithm (Twitter graph)



Work activation: Data-driven
Data access pattern: Push
Schedule: Priority (Set)

Work activation: Data-driven
Data access pattern: Push
Schedule: FIFO

Work activation: Data-driven
Data access pattern: Push
Schedule: Priority (OBIM)

Work activation: Topology
Data access pattern: Pull
Schedule: Unordered

Experiments

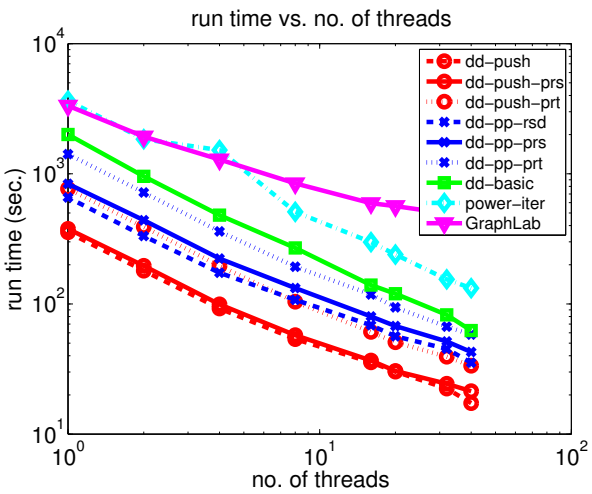
	# nodes	# edges	CSR size
pld	39M	623M	2.7G
sd1	83M	1,937M	7.9G
Twitter	51M	3,228M	13G
Friendster	67M	3,623M	14G

Table 1: Input Graphs

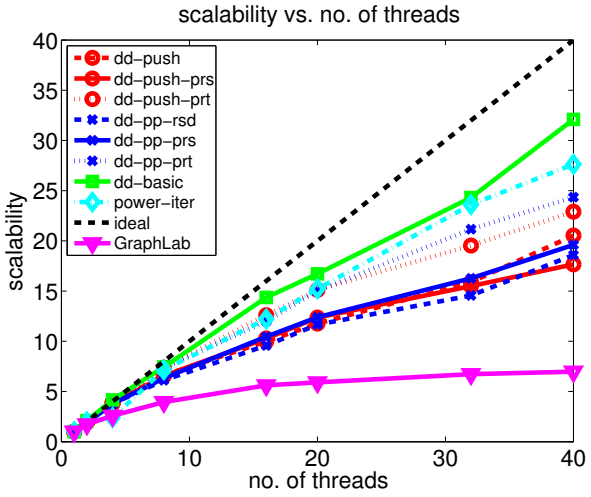
Algorithm	Activation	Access	Schedule
dd-push	Data-driven	Push	FIFOs w/ Stealing
dd-push-prs	Data-driven	Push	Bulk-sync Priority
dd-push-prt	Data-driven	Push	Async Priority
dd-pp-rsd	Data-driven	Pull-Push	FIFOs w/ Stealing
dd-pp-prs	Data-driven	Pull-Push	Bulk-sync Priority
dd-pp-prt	Data-driven	Pull-Push	Async Priority
dd-basic	Data-driven	Pull	FIFOs w/ Stealing
tp	Topology	Pull	Load Balancer

Table 2: Summary of algorithm design choices

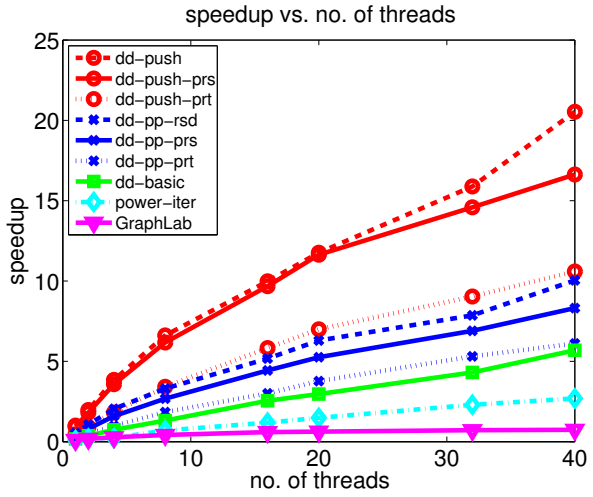
Experiments



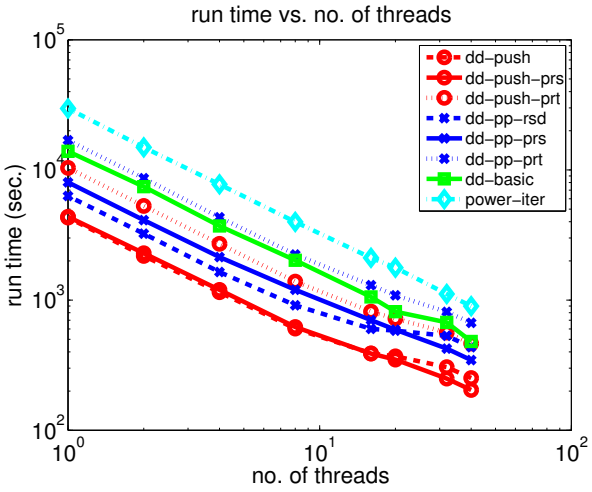
(a) pld run time



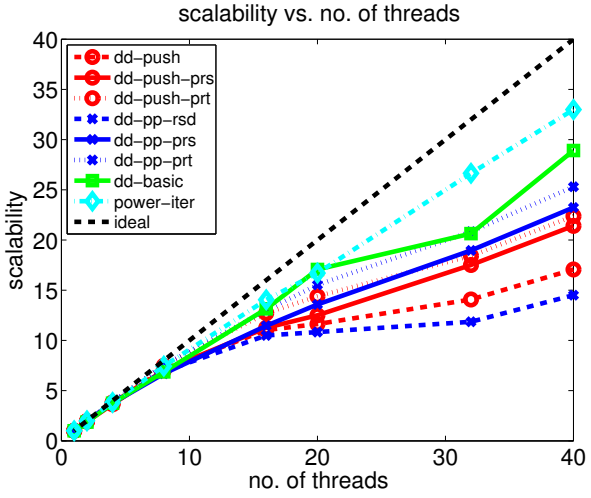
(b) pld scalability



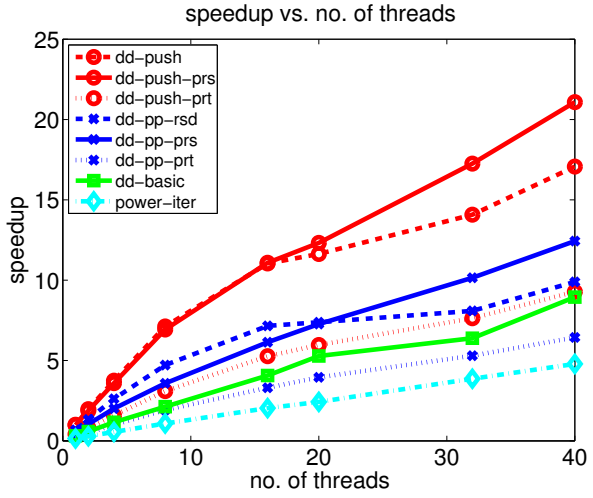
(c) pld speedup



(d) twitter run time

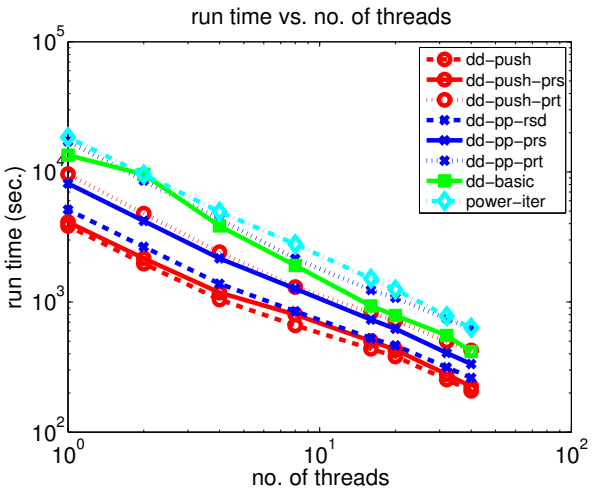


(e) twitter scalability

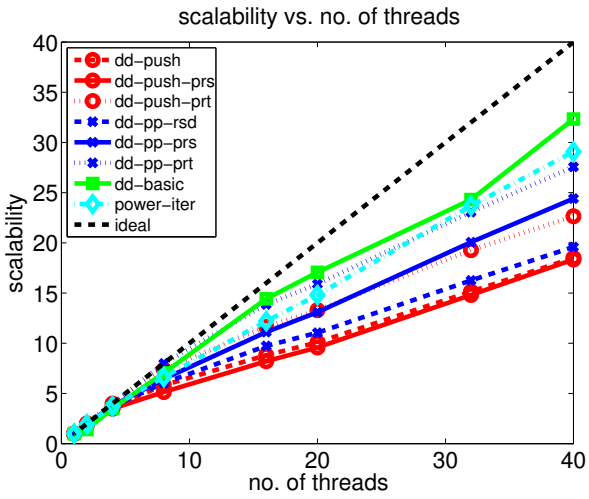


(f) twitter speedup

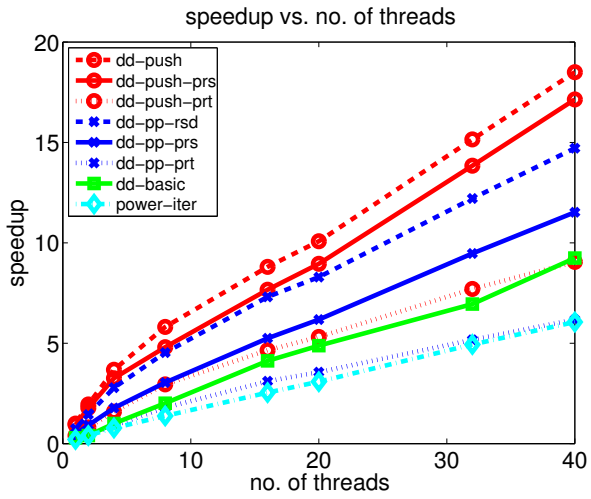
Experiments



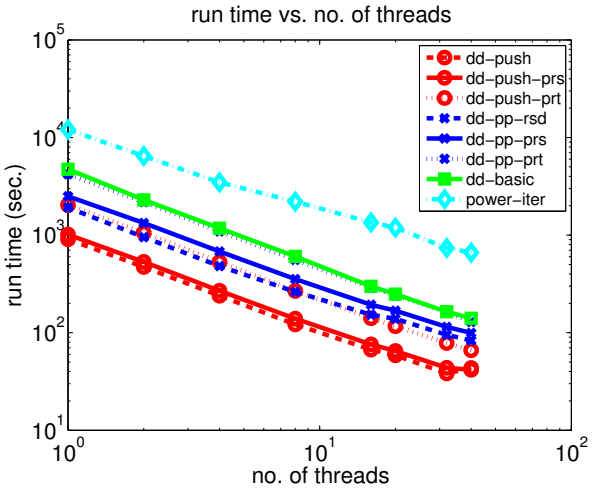
(g) friendster run time



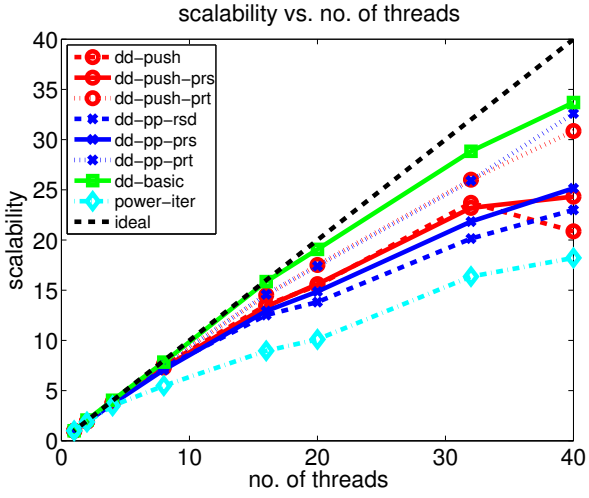
(h) friendster scalability



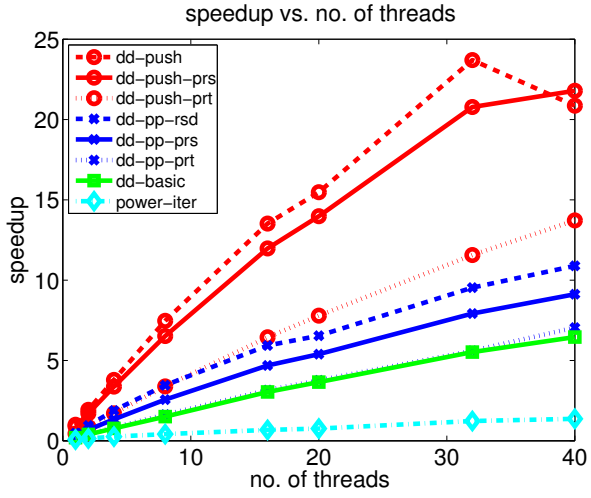
(i) friendster speedup



(j) sd1 run time



(k) sd1 scalability



(l) sd1 speedup

Experiments

- Runtime of different PageRank implementations on pld dataset
  - Using 40 threads, the fastest GraphLabs method takes 478 seconds whereas our push-based PageRank takes 17 seconds.

System	Method	Threads				
		40	32	16	8	1
GraphLab	sync	478 secs.	496 secs.	594 secs.	845 secs.	3,332 secs.
	async-fifo	500 secs.	580 secs.	618 secs.	898 secs.	5,194 secs.
	async-qfifo	788 secs.	804 secs.	970 secs.	1,292 secs.	5,098 secs.
	async-sweep	4,186 secs.	5,162 secs.	9,156 secs.	$>$ 4 hrs.	$>$ 4 hrs.
	async-prt	$>$ 4 hrs.	$>$ 4 hrs.	$>$ 4 hrs.	$>$ 4 hrs.	$>$ 4 hrs.
Galois	power-iter	132 secs.	155 secs.	299 secs.	510 secs.	3,650 secs.
	dd-basic	62 secs.	82 secs.	140 secs.	269 secs.	2,004 secs.
	dd-pp-prt	58 secs.	67 secs.	118 secs.	193 secs.	1,415 secs.
	dd-push	17 secs.	22 secs.	36 secs.	53 secs.	355 secs.

Summary

- Three key algorithm design axes
  - Work activation
  - Data access pattern
  - Scheduling

- Data-driven, push-based PageRank algorithms achieve a significantly superior scalability than standard PageRank implementations.

Big Data Lab

- Email: jjwhang@skku.edu
- Homepage: `http://bigdata.cs.skku.edu/`
- Tel: 031-299-4396
- Office: Engineering Building 2, #27326
- Lab: Engineering Building 2, #26315B