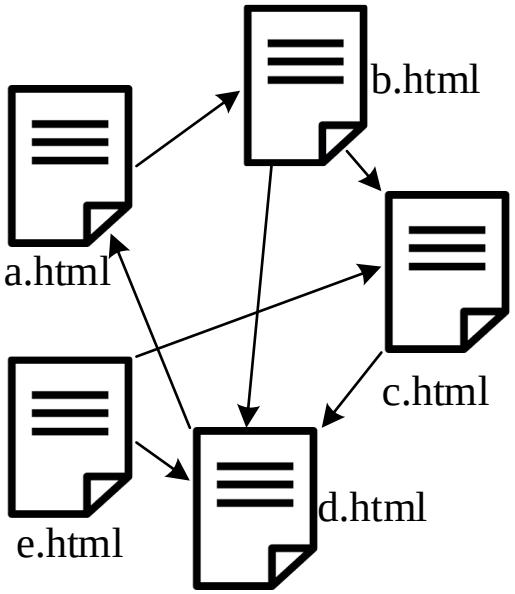
Automating Incremental Graph Processing with Flexible Memoization

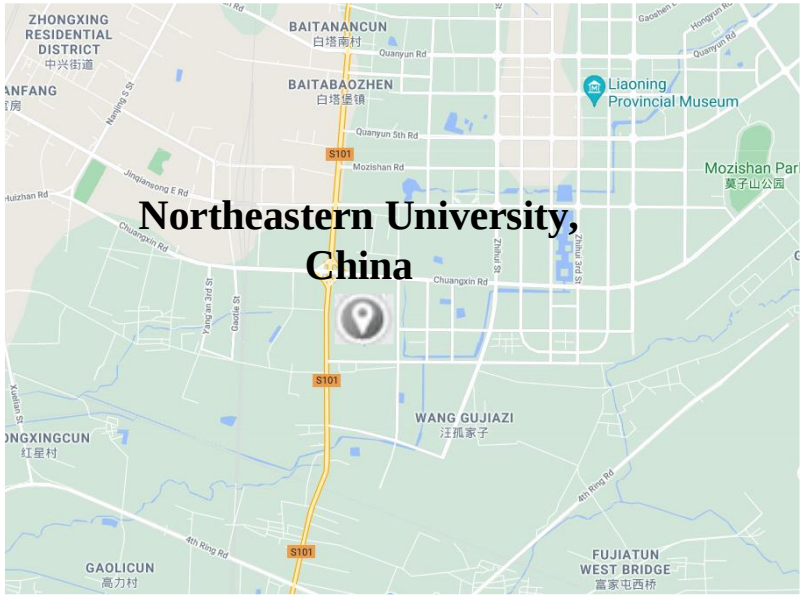
Shufeng Gong, Chao Tian, Qiang Yin, Wenyuan Yu, Yanfeng Zhang, Liang Geng, Song Yu, Ge Yu, Jingren Zhou

Northeastern University
Alibaba Group

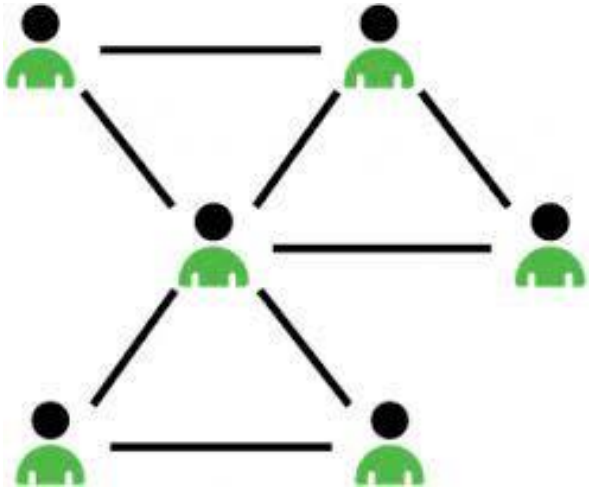
Graph Computation



Web Graph

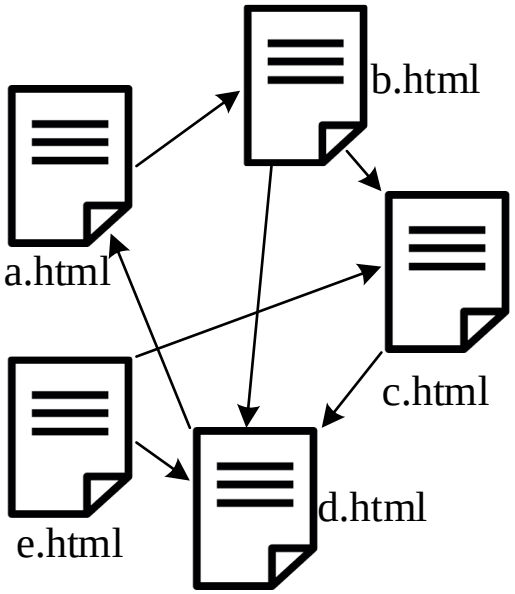


Road Graph

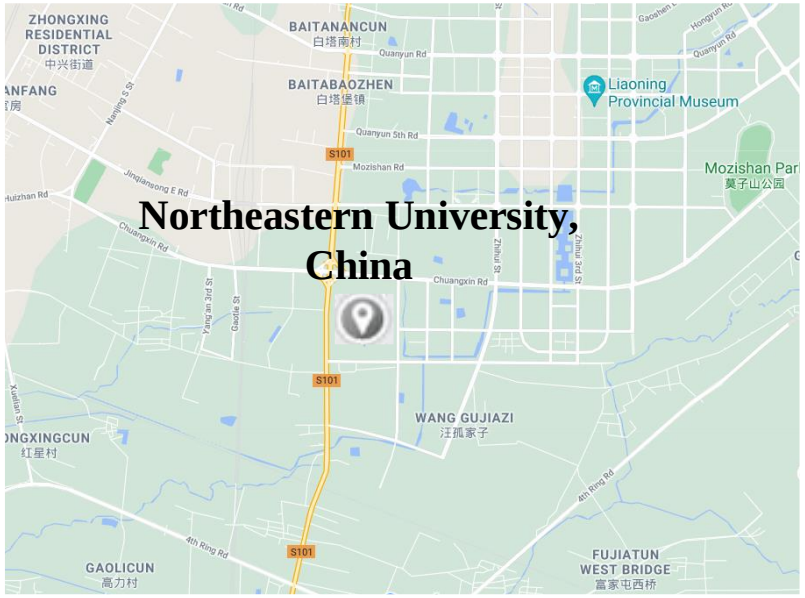


Social Graph

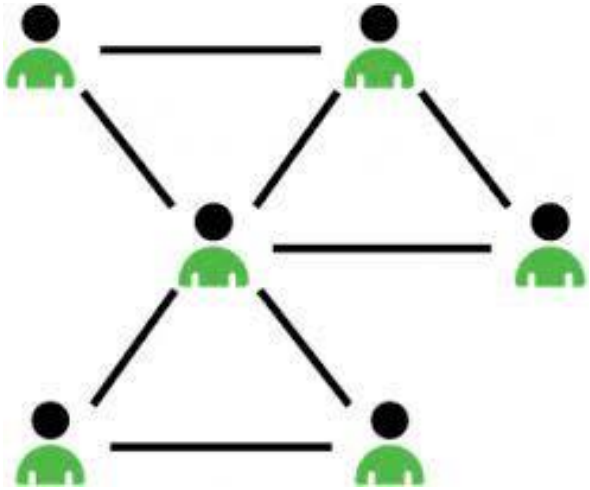
Graph Computation



Web Graph



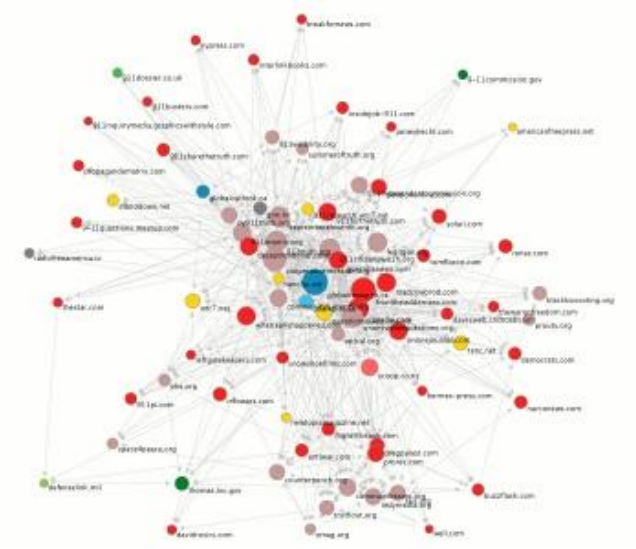
Road Graph



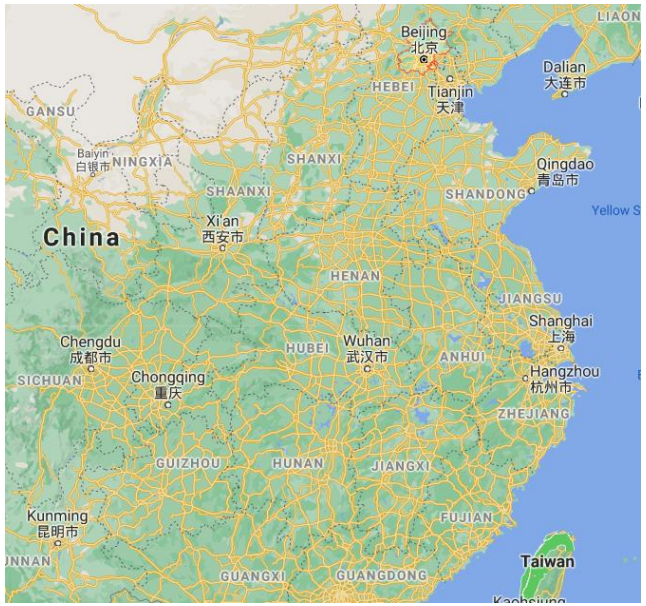
Social Graph

We can discover valuable information from these graphs with graph analytics algorithms, such as *PageRank*, *SSSP*, and *Connected Components*.

Graph Computation



Web Graph

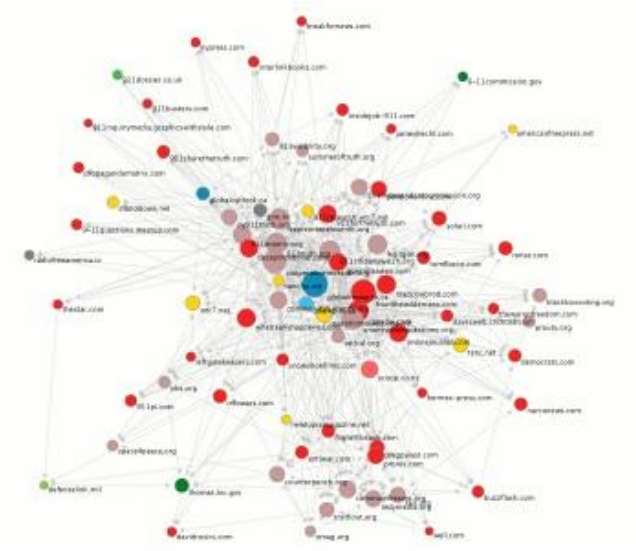


Road Graph

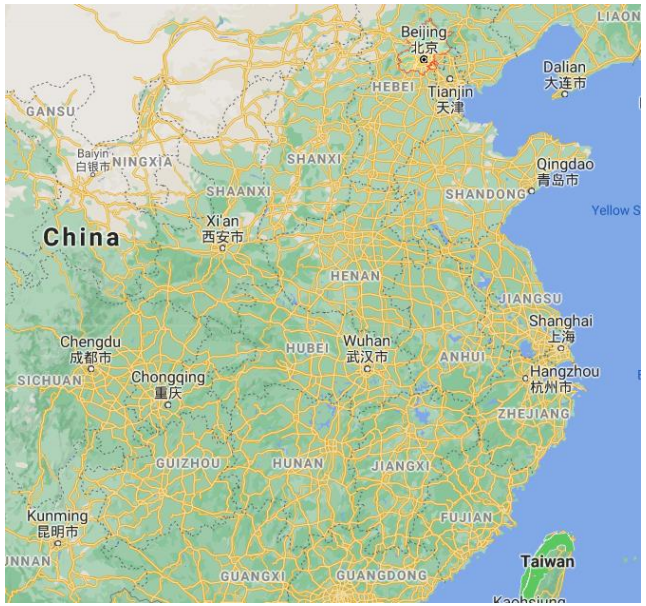
Social Graph

In fact, the graphs in our real life are very large.
It is difficult for us to analysis them.

Graph Computation



Web Graph



Road Graph

Social Graph

For this, some parallel or distributed graph processing systems have been proposed to help us to analyze these large graphs.

Graph Processing Systems

Prege: a system for large-scale graph processing. [Malewicz G, et. al. 2010]

PowerGraph: Distributed graph-parallel computation on natural graphs [Joseph E G, et. al. 2012]

From "Think Like a Vertex" to "Think Like a Graph [Yuanyuan T, et. al. 2013]

Maiter: An asynchronous graph processing framework for delta-based accumulative iterative computation [Yanfeng Z, et. al. 2013]

Gemini: A computation-centric distributed graph processing system. [Xiaowei Z, et. al. 2016]

Grape: Parallelizing Sequential Graph Computations. [Wenfei F, et. al. 2017]

…

…

Graph Processing Systems

Prege: a system for large-scale graph processing. [Malewicz G, et. al. 2010]

PowerGraph: Distributed graph-parallel computation on natural graphs [Joseph E G, et. al. 2012]

From "Think Like a Vertex" to "Think Like a Graph [Yuanyuan T, et. al. 2013]

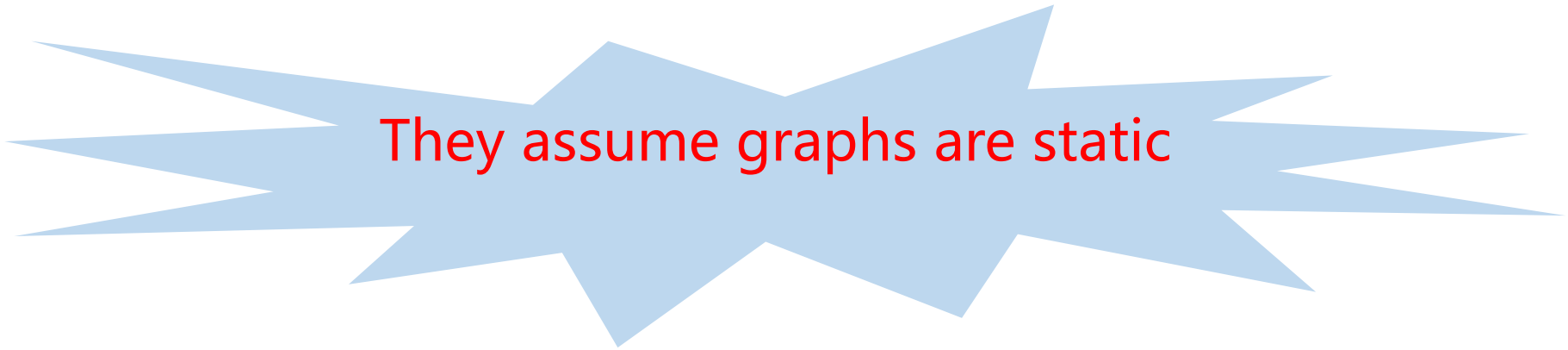
Maiter: An asynchronous graph processing framework for delta-based accumulative iterative computation [Yanfeng Z, et. al. 2013]

Gemini: A computation-centric distributed graph processing system. [Xiaowei Z, et. al. 2016]

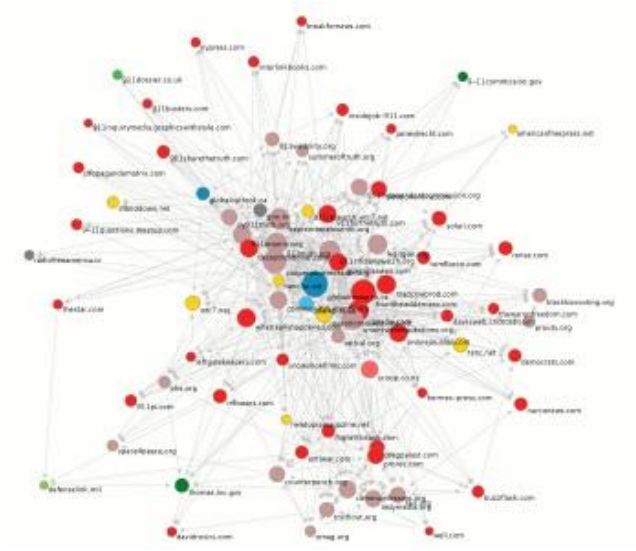
Grape: Parallelizing Sequential Graph Computations. [Wenfei F, et. al. 2017]

…

…

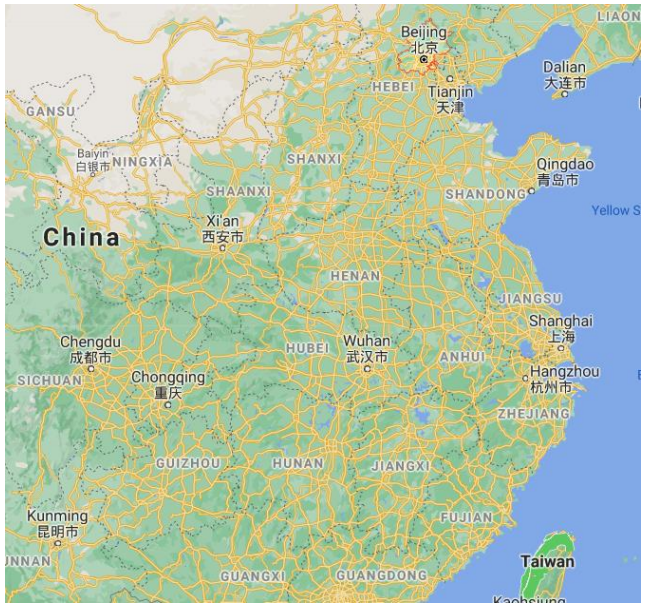


Evolving Graphs



Web Graph

web page additional/deletion

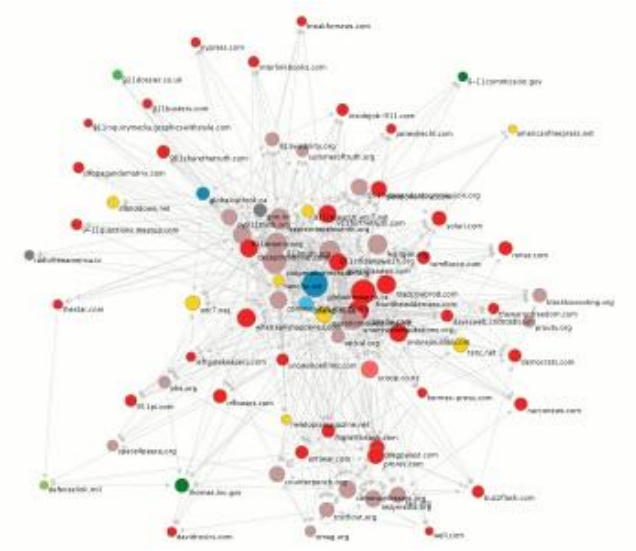


Road Graph

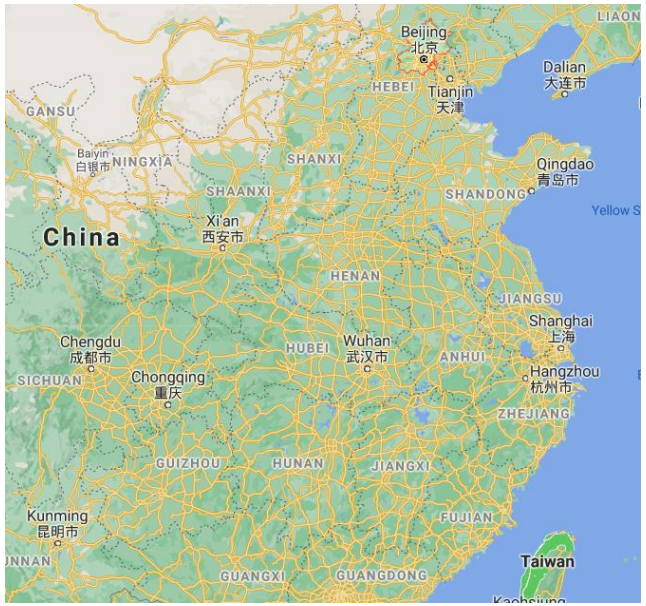
Social Graph

In our real life, graphs are evolving all the time

Evolving Graphs



Web Graph



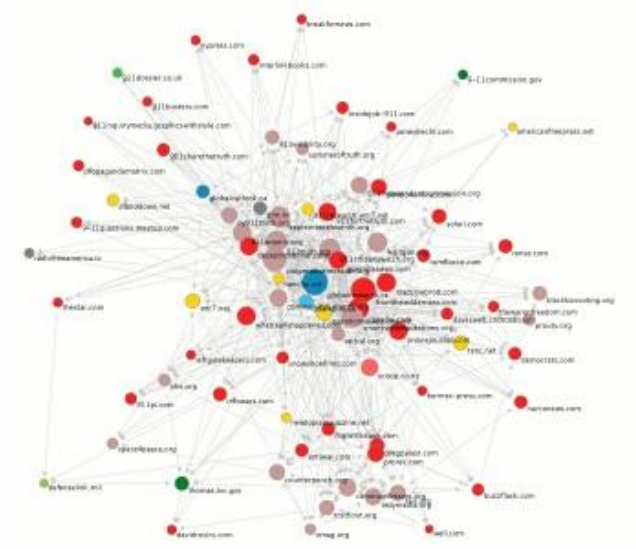
Road Graph

Social Graph

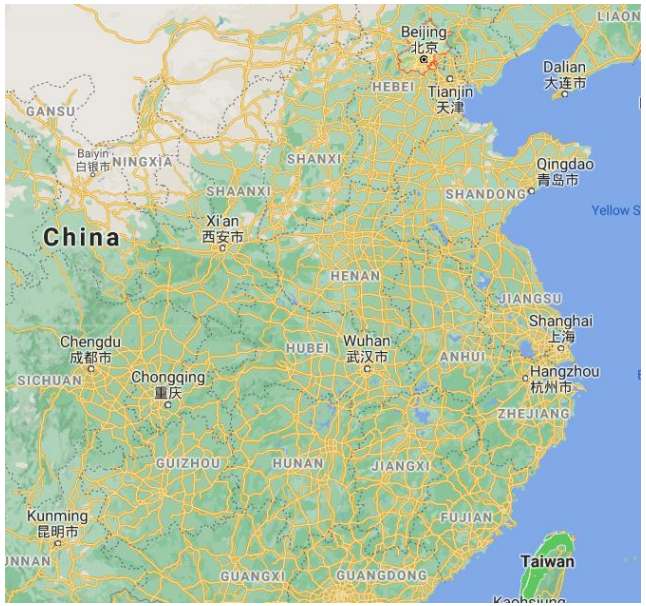
road obstruction/opening

In our real life, graphs are evolving all the time

Evolving Graphs



Web Graph



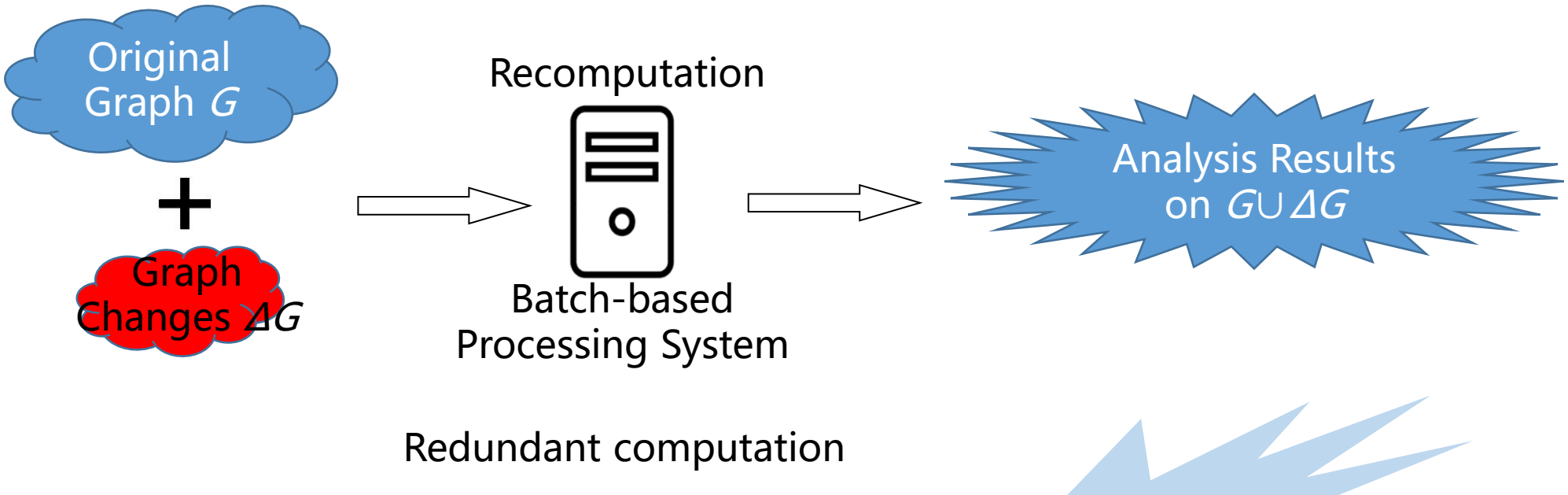
Road Graph

Social Graph

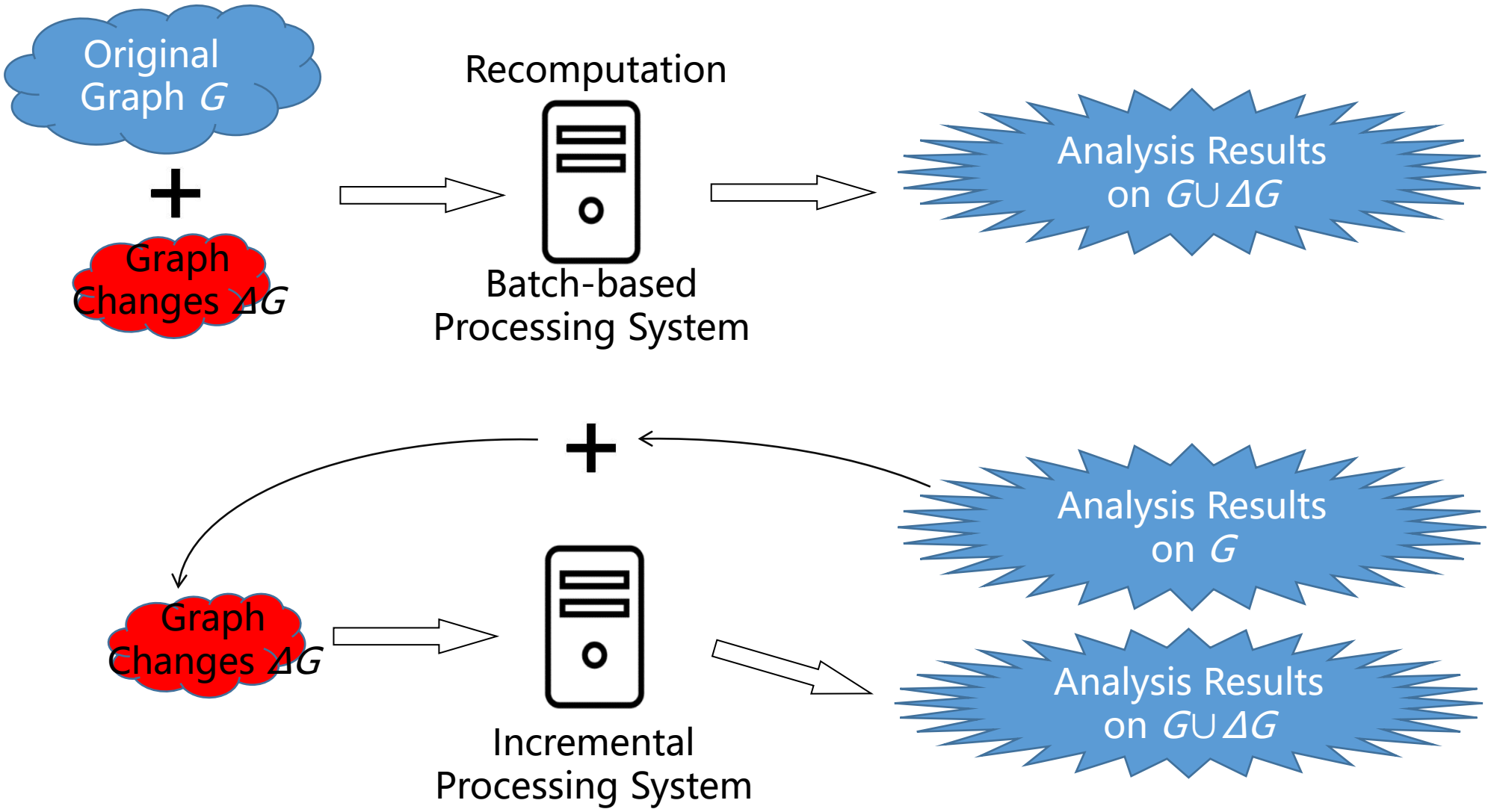
netizens
follow/unfollow

In our real life, graphs are evolving all the time

Incremental Graph Processing



Incremental Graph Processing



Incremental Graph Processing Systems

1. Graphbolt: Dependency-drivensynchronous processing of streaming graphs. [Mugilan M., et. al. 2018]

2. Kickstarter: Fast and accurate computations on streaming graphs via trimmed approximations. [Keval V., et. al, ASPLOS 2017.

3. Graphin: An online high performance incremental graph processing framework. [Dipanjan S., et. al. 2016].

4. Tornado: A system for real-time iterative analysis over evolving data. [Xiaogang S., et. al. 2016]

…

…

1. Only focus on a specific class of algorithms
2. Require no-trivial incremental computation logical

Incremental Graph Processing Systems

require no-trivial incremental computation logical

1. Graphbolt: Dependency-drivensynchronous processing of streaming graphs. [Mugilan M., et. al. 2018]

only for monotonous and single dependency algorithms

2. Kickstarter: Fast and accurate computations on streaming graphs via trimmed approximations. [Keval V., et. al. ASPLOS 2017

require no-trivial incremental computation logical

3. Graphin: An online high performance incremental graph processing framework. [Dipanjan S., et. al. 2016].

4. Tornado: A system for real-time iterative analysis over evolving data. [Xiaogang S., et. al. 2016]

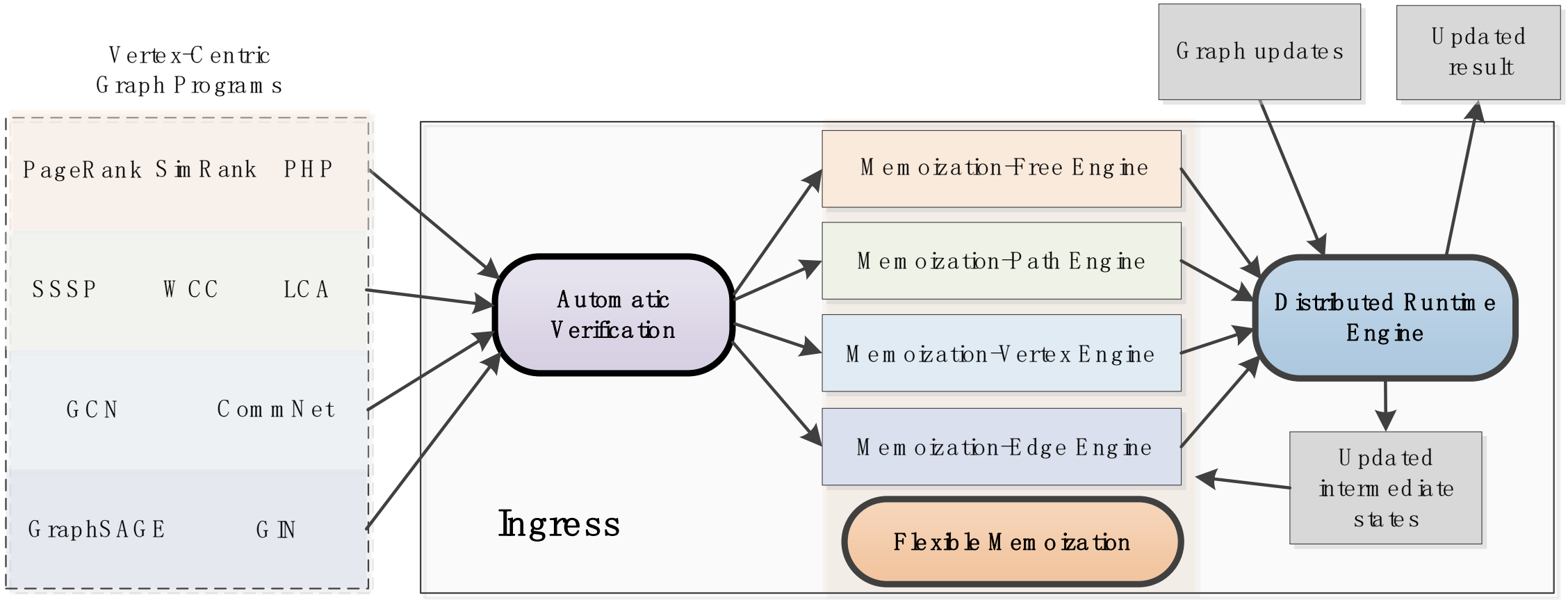
only for algorithms that converge to same fix-point from different initial states

…

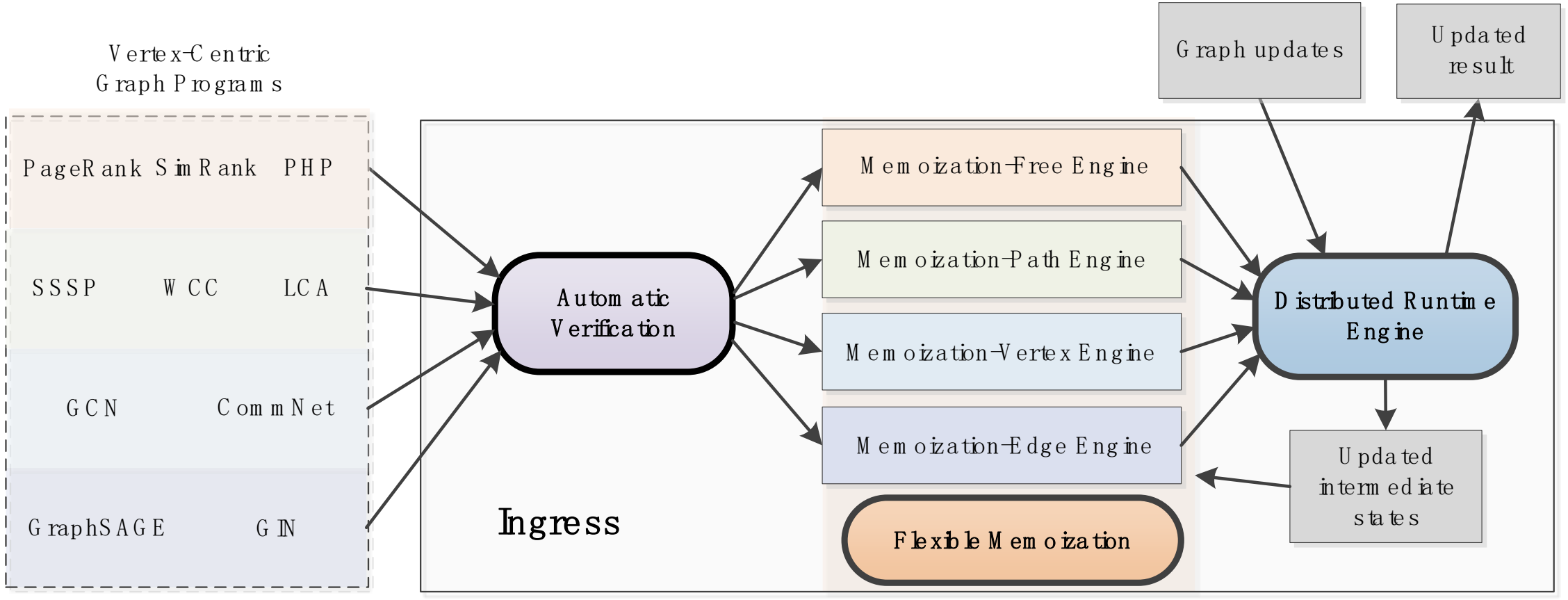
…

1. Only focus on a specific class of algorithms
2. Require no-trivial incremental computation logical

Ingress


Vertex-Centric
Graph Programs
PageRank SimRank PHP
SSSP WCC LCA
GCN CommNet
GraphSAGE GIN
Automatic
Verification
Memoization-Free Engine
Memoization-Path Engine
Memoization-Vertex Engine
Memoization-Edge Engine
Flexible Memoization
Ingress
Graph updates
Updated
result
Distributed Runtime
Engine
Updated
intermediate
states

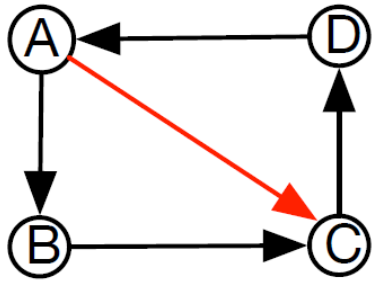
Ingress



1. Ingress is able to incrementalize batch vertex-centric algorithms without users' intervention.
2. Ingress supports all kinds of vertex-centric computations with optimize memory utilization.

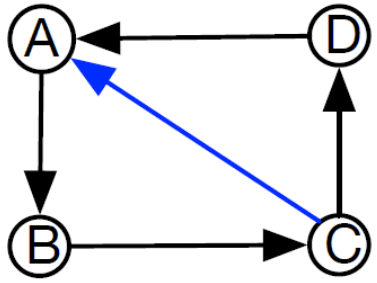
Message-driven Differentiation

Message-driven: the state of each vertex is decided by the received messages from its neighbors.



(1). Graph G

The state of C is decided by the messages received from A and B.



(2). $G \oplus \Delta G$

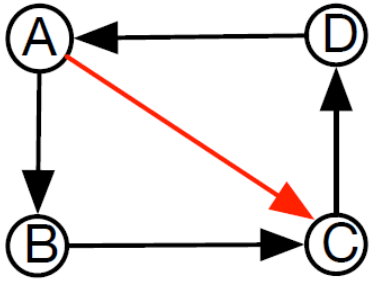
The state of C is decided by the messages received from B.

Reduce the problem of finding the differences among two runs of a batch vertex-centric algorithm to identifying the changes to messages.

Message-driven Differentiation

Message-driven: the state of each vertex is decided by the received messages from its neighbors.

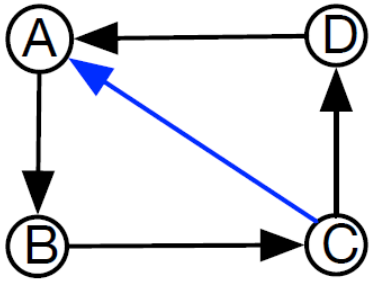
A sends C a message (invalid message),
C do not send A a message (missing message)



(1). Graph G

Messages	A	B	C	D
Cancellation	$-dx_A^*/2 \rightarrow B, C$	$\emptyset$	$-dx_C^* \rightarrow D$	$\emptyset$
Compensation	$dx_A^* \rightarrow B$	$\emptyset$	$dx_C^*/2 \rightarrow A, D$	$\emptyset$

(3). PageRank(x_A^*, x_C^* are PagreRank scores of A and C in G)



(2). $G \oplus \Delta G$

Messages	A	B	C	D
Cancellation	$\perp \rightarrow C$	$\emptyset$	$\perp \rightarrow D$	$\emptyset$
Compensation	$\emptyset$	$2 \rightarrow C$	$\emptyset$	$\emptyset$

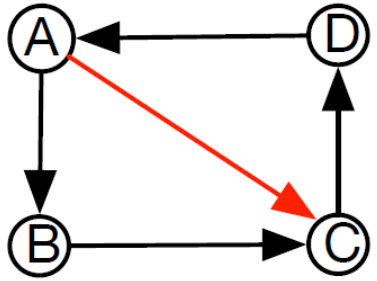
(4). SSSP (source = A)

A sends C a cancellation message,
C sends A a compensation message

Message-driven Differentiation

Message-driven: the state of each vertex is decided by the received messages from its neighbors.

A sends C a message (invalid message),
C do not send A a message (missing message)

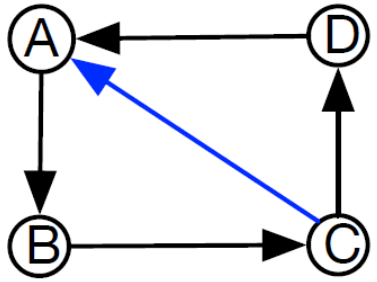


(1). Graph G

Messages	A	B	C	D
Cancellation	$-dx_A^* \ldots 2 \rightarrow B, C$	$\emptyset$	$-dx_C^* \rightarrow D$	$\emptyset$
Compensation	$\ldots B$	$\emptyset$	$dx_C^*/2 \rightarrow A, D$	$\emptyset$

Large Memory Overhead

(3). PageR… scores of A and C in G)



(2). $G \oplus \Delta G$

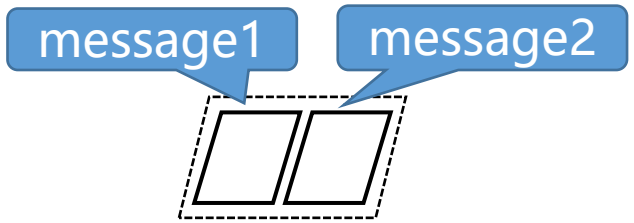
Messag…	A	B	C	D
Cancellation	$\perp \rightarrow C$	$\emptyset$	$\perp \rightarrow D$	$\emptyset$
Compensation	$\emptyset$	$2 \rightarrow C$	$\emptyset$	$\emptyset$

(4). SSSP (source = A)

A sends C a cancellation message,
C sends A a compensation message

Flexible Memoization

Memoization-Free

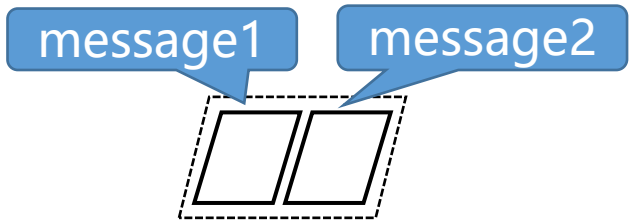


vertex state

Delta-PageRank
Delta-PHP

Flexible Memoization

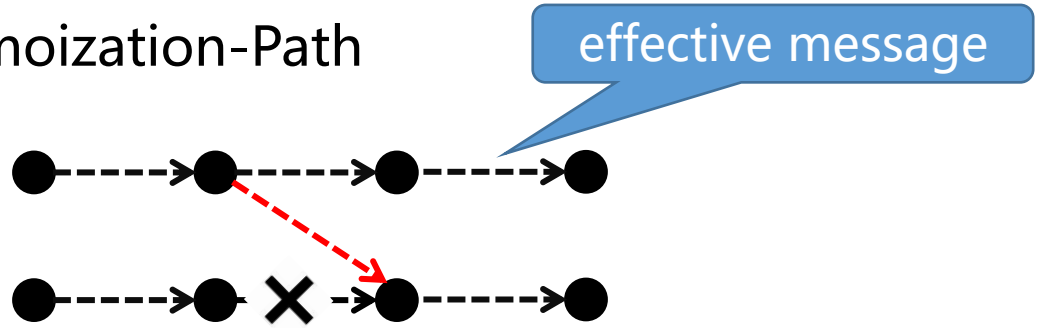
Memoization-Free



vertex state

Delta-PageRank
Delta-PHP

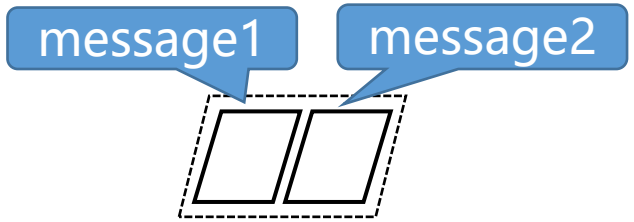
Memoization-Path



SSSP, *CC*, *SSWP*

Flexible Memoization

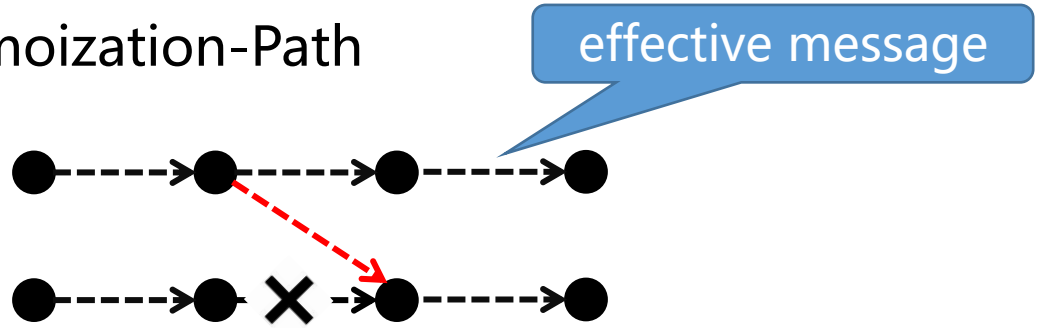
Memoization-Free



vertex state

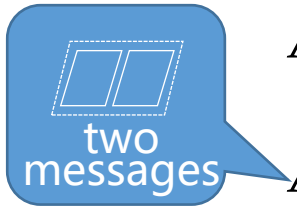
Delta-PageRank
Delta-PHP

Memoization-Path



SSSP, *CC*, *SSWP*

Memoization-Vertex



A_{g0} B_{g0} C_{g0} D_{g0} ⋯

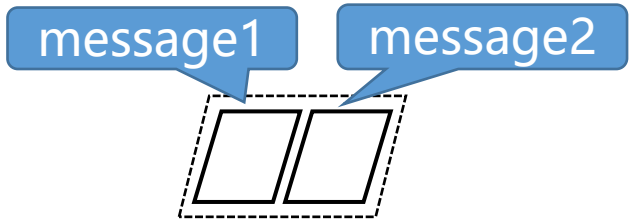
A_{g1} B_{g1} C_{g1} D_{g1} ⋯

A_{g2} B_{g2} C_{g2} D_{g2} ⋯

GCN, *CommNet*

Flexible Memoization

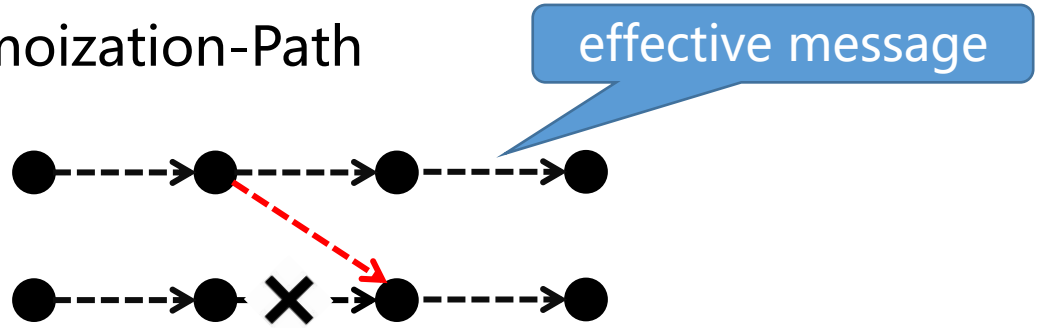
Memoization-Free



vertex state

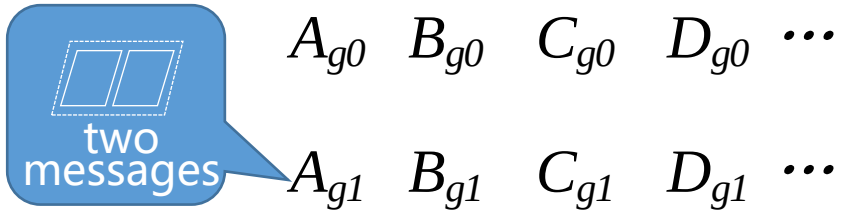
Delta-PageRank
Delta-PHP

Memoization-Path



SSSP, CC, SSWP

Memoization-Vertex



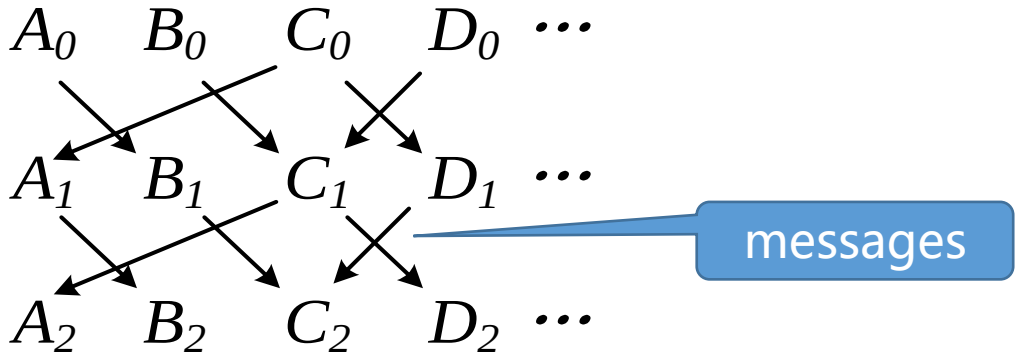
A_{g0} B_{g0} C_{g0} D_{g0} ⋯

A_{g1} B_{g1} C_{g1} D_{g1} ⋯

A_{g2} B_{g2} C_{g2} D_{g2} ⋯

GCN, CommNet

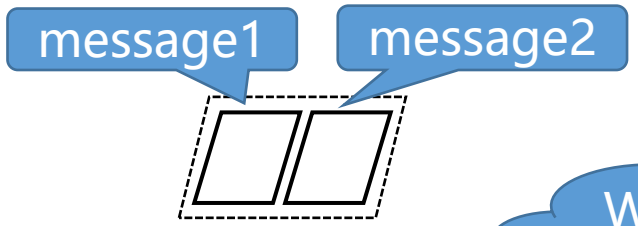
Memoization-Edge



GraphSAGE, GIN

Flexible Memoization

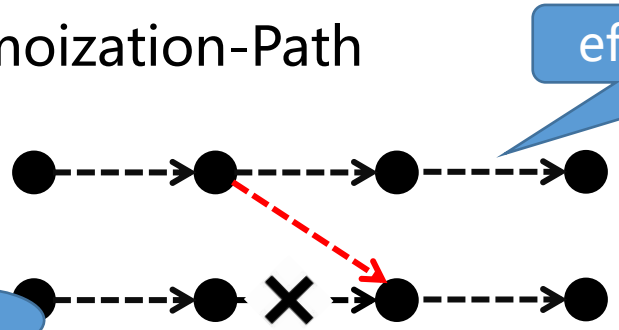
Memoization-Free



vertex state

De
Delta-PHP

Memoization-Path



SSSP, CC, SSWP

Which is optimal selection?

Vertex Memoization

A_{g0} B_{g0} C_{g0} D_{g0} …

A_{g1} B_{g1} C_{g1} D_{g1} …

A_{g2} B_{g2} C_{g2} D_{g2} …

GCN, CommNet

D_0 …

D_1 …

messages

A_2 B_2 C_2 D_2 …

GraphSAGE, GIN

The aggregation of received messages

aggregation function

the set of received messages

$$m_v^i = \mathcal{H}(M_v^{i-1}),$$

$$x_v^i = \mathcal{U}(x_v^{i-1}, m_v^i),$$

$$m_{v,w}^i = \mathcal{G}(x_v^i, m_v^i, P_E(v, w)) \quad (\forall w \in \mathrm{Nbr}(v))$$

$$m_v^i = \mathcal{H}(M_v^{i-1}),$$
$$x_v^i = \mathcal{U}(x_v^{i-1}, m_v^i),$$
$$m_{v,w}^i = \mathcal{G}(x_v^i, m_v^i, P_E(v, w)) \quad (\forall w \in \mathrm{Nbr}(v))$$

update function

vertex state

$$m_v^i = \mathcal{H}(M_v^{i-1}),$$

$$x_v^i = \mathcal{U}(x_v^{i-1}, m_v^i),$$

$$m_{v,w}^i = \mathcal{G}(x_v^i, m_v^i, P_E(v, w)) \quad (\forall w \in \mathrm{Nbr}(v))$$

message from *v* to *w*

Message generation function

Edge property

$$
\begin{aligned}
m_v^i &= \mathcal{H}(M_v^{i-1}), \\
x_v^i &= \mathcal{U}(x_v^{i-1}, m_v^i), \\
m_{v,w}^i &= \mathcal{G}(x_v^i, m_v^i, P_E(v, w)) \quad (\forall w \in \mathrm{Nbr}(v))
\end{aligned}
$$

Free Memoization:

$$
\begin{aligned}
&(\mathbf{C1})\ \mathcal{U}(M \setminus M') = \mathcal{U}(M \cup \{\mathcal{U}^- \circ \mathcal{U}(M')\}) \quad (\forall M' \subseteq M) \\
&(\mathbf{C2})\ \mathcal{U}(\{\mathcal{U}(M)\} \cup M') = \mathcal{U}(M \cup M') \\
&(\mathbf{C3})\ \mathcal{U} \circ \mathcal{G} \circ \mathcal{U}(M) = \mathcal{U} \circ \mathcal{G}(M)
\end{aligned}
$$

Path Memoization:

$$
\begin{aligned}
&(\mathbf{C2})\ \mathcal{U}(\{\mathcal{U}(M)\} \cup M') = \mathcal{U}(M \cup M') \\
&(\mathbf{C3})\ \mathcal{U} \circ \mathcal{G} \circ \mathcal{U}(M) = \mathcal{U} \circ \mathcal{G}(M) \\
&(\mathbf{C4})\ \mathcal{U}(M) = m_c \in M.
\end{aligned}
$$

Vertex Memoization:

$$
(\mathbf{C5})\ \mathcal{H}(M \setminus M') = \mathcal{H}(M \cup \{\mathcal{H}^- \circ \mathcal{H}(M')\}) \quad (\forall M' \subseteq M)
$$

Edge Memoization: None

$$
\begin{aligned}
m_v^i &= \mathcal{H}(M_v^{i-1}), \\
x_v^i &= \mathcal{U}(x_v^{i-1}, m_v^i), \\
m_{v,w}^i &= \mathcal{G}(x_v^i, m_v^i, P_E(v, w)) \quad (\forall w \in \mathrm{Nbr}(v))
\end{aligned}
$$

Free Memoization:

$$
\begin{aligned}
&(\mathbf{C1})\ \mathcal{U}(M \setminus M') = \mathcal{U}(M \cup \{\mathcal{U}^- \circ \mathcal{U}(M')\}) \quad (\forall M' \subseteq M) \\
&(\mathbf{C2})\ \mathcal{U}(\{\mathcal{U}(M)\} \cup M') = \mathcal{U}(M \cup M') \\
&(\mathbf{C3})\ \mathcal{U} \circ \mathcal{G} \circ \mathcal{U}(M) = \mathcal{U} \circ \mathcal{G}(M)
\end{aligned}
$$

Path Memoization:

$$
\begin{aligned}
&(\mathbf{C2})\ \mathcal{U}(\{\mathcal{U}(M)\} \cup M') = \mathcal{U}(M \cup M') \\
&(\mathbf{C3})\ \mathcal{U} \circ \mathcal{G} \circ \mathcal{U}(M) = \mathcal{U} \circ \mathcal{G}(M) \\
&(\mathbf{C4})\ \mathcal{U}(M) = m_c \in M.
\end{aligned}
$$

Vertex Memoization:

$$
(\mathbf{C5})\ \mathcal{H}(M \setminus M') = \mathcal{H}(M \cup \{\mathcal{H}^- \circ \mathcal{H}(M')\}) \quad (\forall M' \subseteq M)
$$

Edge Memoization: None

Automating Incremental Graph Processing with Flexible Memoization.
Shufeng Gong, Chao Tian, Qiang Yin, et. al.

Experimental Study

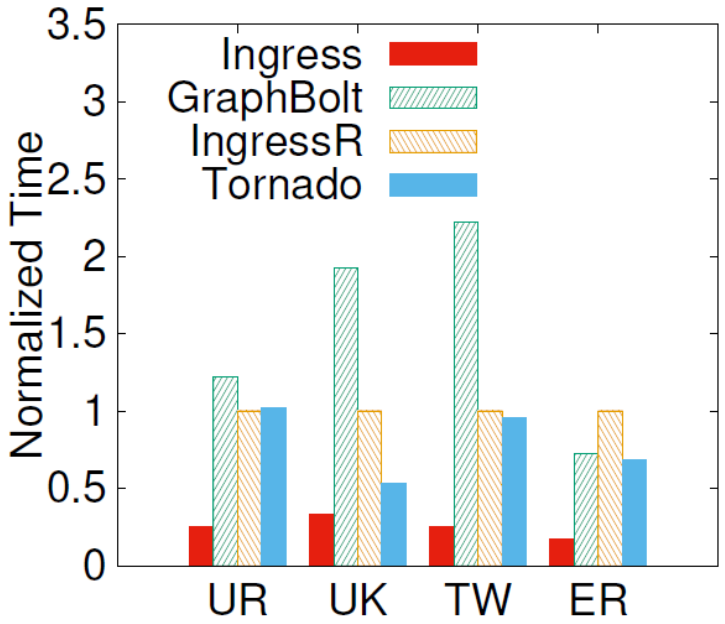
- Competitors
  Tornado[X. Shi], GraphBolt [M. Mariappan]
  KickStarter [K. Vora], IngressR

- Environment
  AliCloud 1 ecs.r6.13xlarge (52vCPU, 384G Memory)
  32 ecs.r6.6xlarge (24vCPU, 192G Memory)

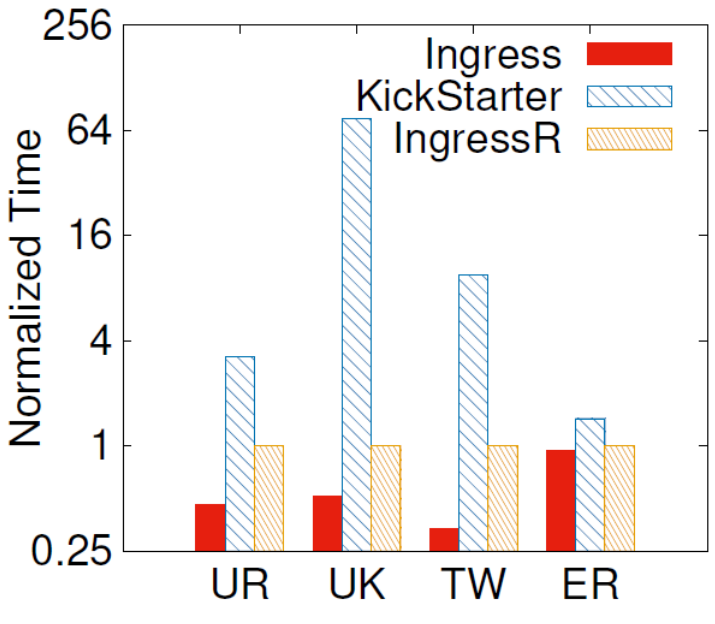
- Datasets

Graph	#Vertices	#Edges	Size
Twitter-2009 (TW) [39]	41,652,230	1,468,365,183	23.99GB
UK-2005 (UK) [1]	39,459,925	936,364,282	16.45GB
Euro-Road (ER) [2]	50,912,018	108,109,320	0.94GB
US-Road (UR) [3]	23,947,347	57,708,624	0.49GB
Friendster (FS) [49]	65,608,366	1,806,067,139	30.14GB

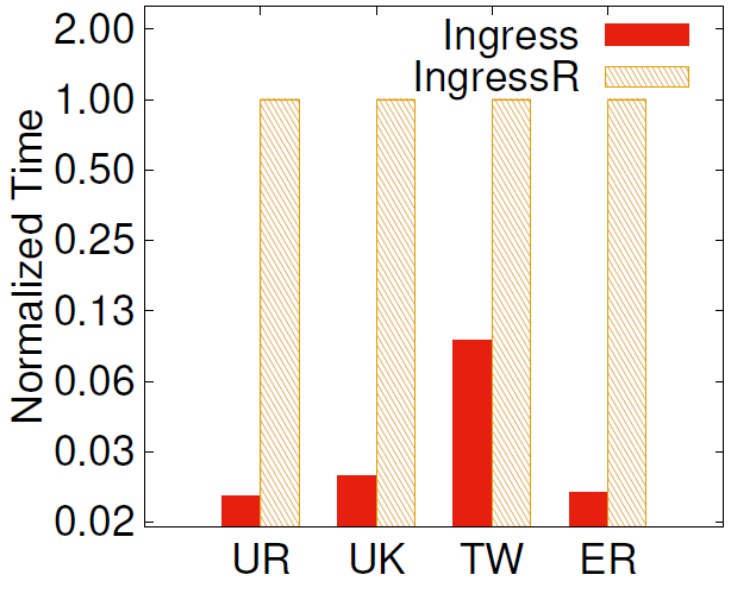
Response Time



PageRank



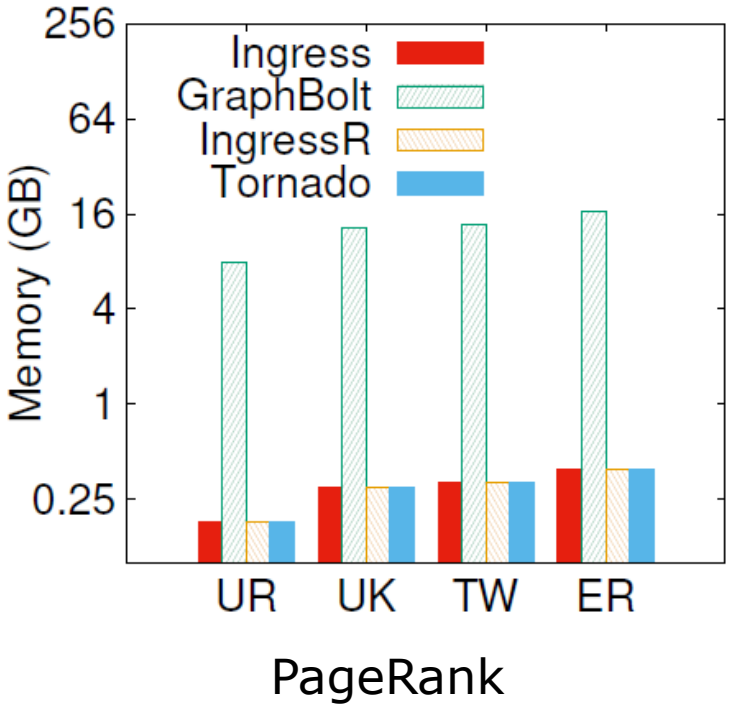
SSSP



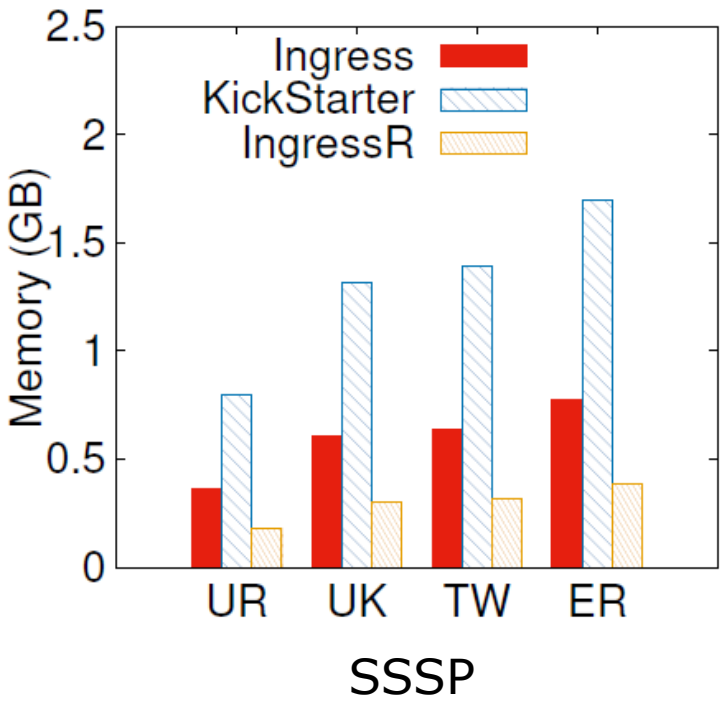
GCN-Forward

Ingress outperforms state-of-the-art incremental graph systems by 15.93× on average.

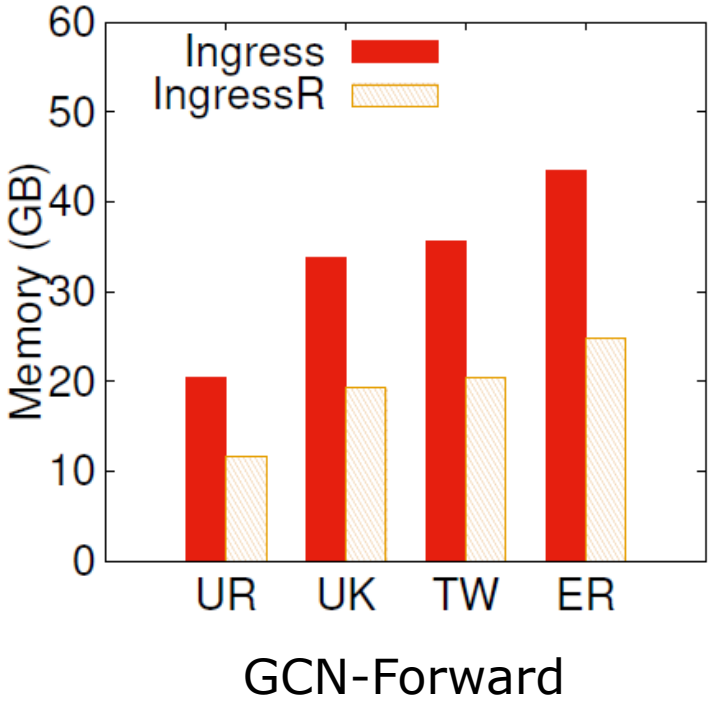
Space Cost



PageRank



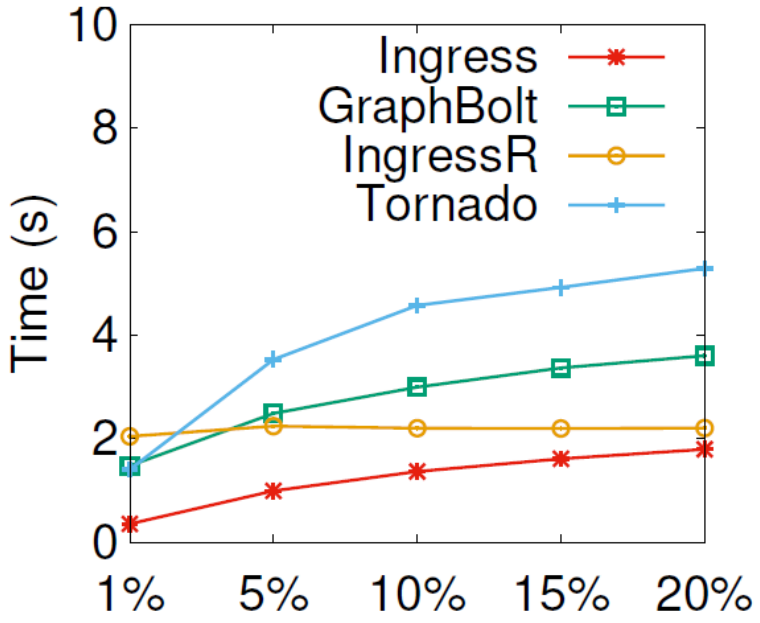
SSSP



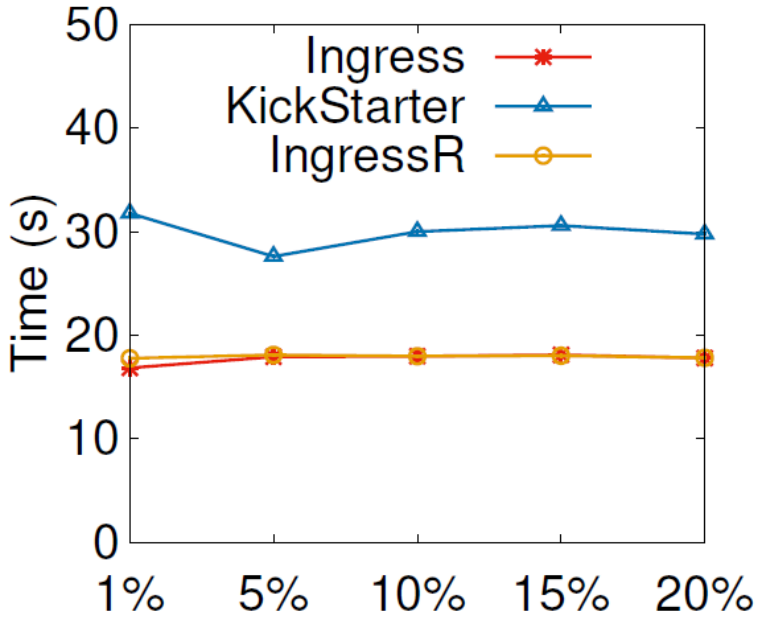
GCN-Forward

Compared to GraphBolt and KickStarter, Ingress always uses less memory.

Sensitivity to Updates



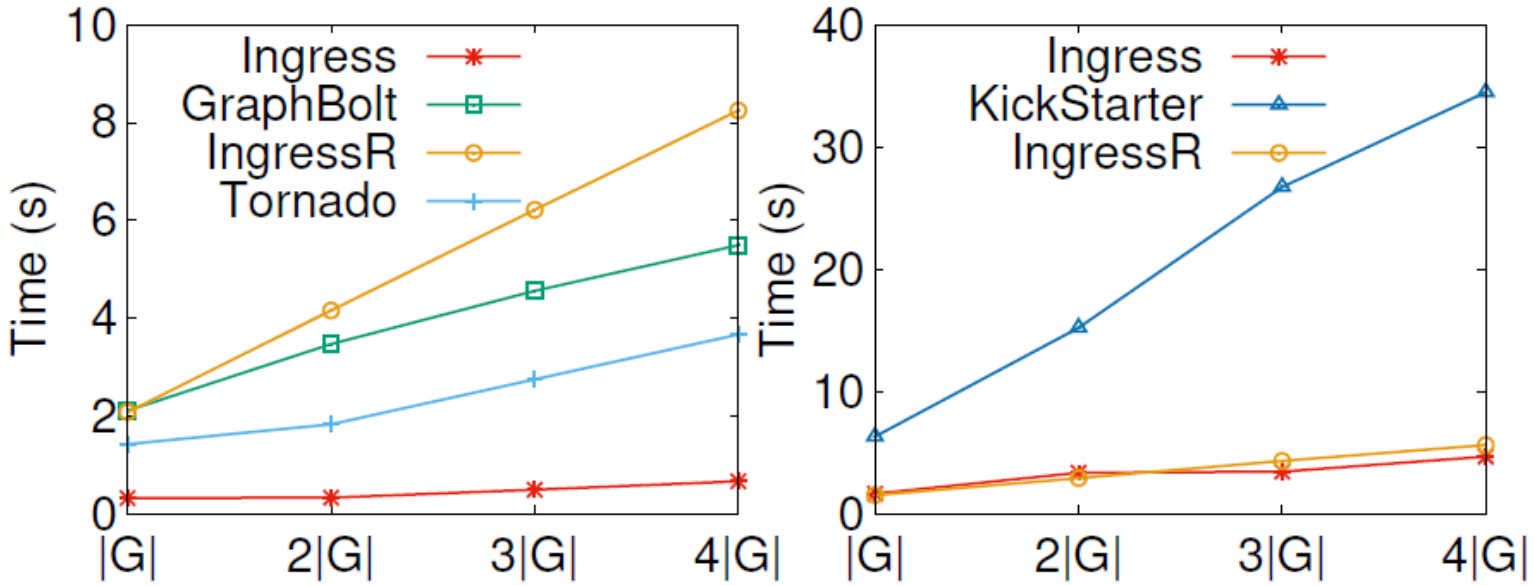
PageRank



SSSP

Almost all the incremental graph processing systems take longer to process larger graph updates.
Ingress show an better performance than Tornado and graphbolt, and comparable performance with KickStarter.

Sensitivity to Graph Size



PageRank

SSSP

Compared with IngressR, the response time of incremental systems Ingress, GraphBolt and KickStarter are less sensitive to the increase of |G|.

Conclusion

- Design a vertex-centric incremental graph processing framework.
- Design four memorization policy and identify their sufficient conditions.
- Implement an incremental graph processing system, Ingress.

Thanks!