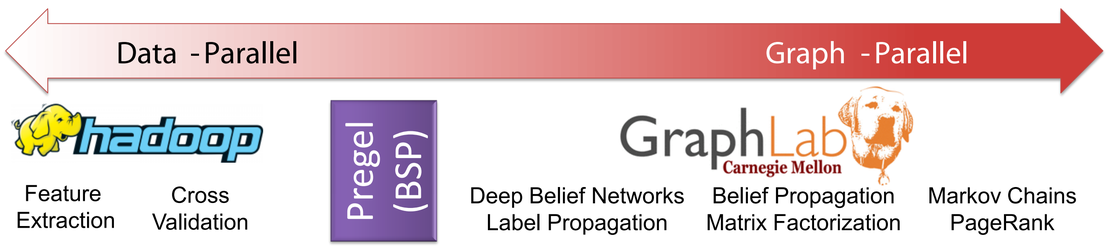
# GRAPHLAB2: THE NEXT PARALLEL ABSTRACTION FOR ML

Yucheng Low, Joseph Gonzalez, Aapo Kyrola, Haijie Gu, Danny Bickson, Carlos Guestrin, Alex Smola, Joseph M. Hellerstein (CMU)

## EXECUTIVE SUMMARY

- Popular parallel abstractions like *MapReduce* and *Pregel* do **not efficiently express** many machine learning algorithms:
  - *MapReduce*: does not express graph computation
  - *Pregel*: does not express asynchronous computation
- To fill this critical void we introduced **GraphLab**, a new abstraction **targeted** at advanced **machine learning** algorithms and capable of expressing **adaptive asynchronous graph computation**

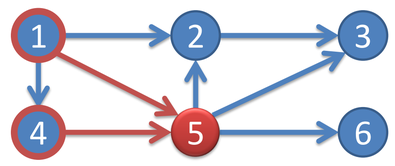



- GraphLab has been successfully applied to many important large-scale, real-world machine learning problems
- Through our work on GraphLab we have identified **Natural Graphs** as **key challenge** to scaling graph computation
- To address the challenge of natural graphs we introduce **GraphLab2** which significantly extends the original GraphLab abstraction

## THE GRAPHLAB ABSTRACTION

- **Running Example:** *PageRank*
  - Rank of a page is a weighted sum of the rank of neighbor pages:

$$R[i] = \alpha + (1-\alpha) \sum_{(j,i)\in E} \frac{1}{L[j]} R[j]$$

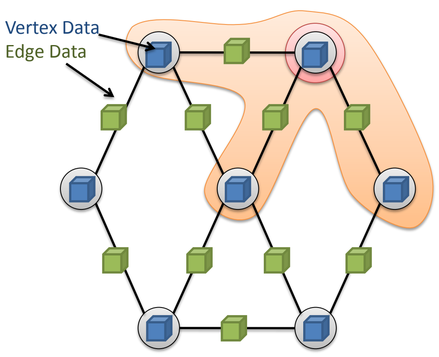



$$R[5] = \alpha + (1-\alpha)\left(\frac{R[1]}{3} + R[4]\right)$$

  - α is the random reset probability
  - L[j] is the number of links on page j
- Abstraction:

**Data Graph**

A **graph** with arbitrary data associated with each vertex and edge.



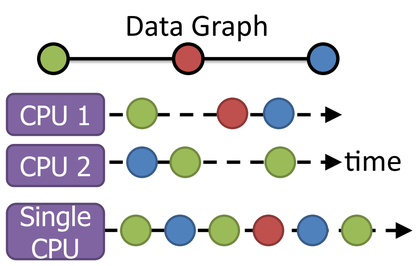

Update Function

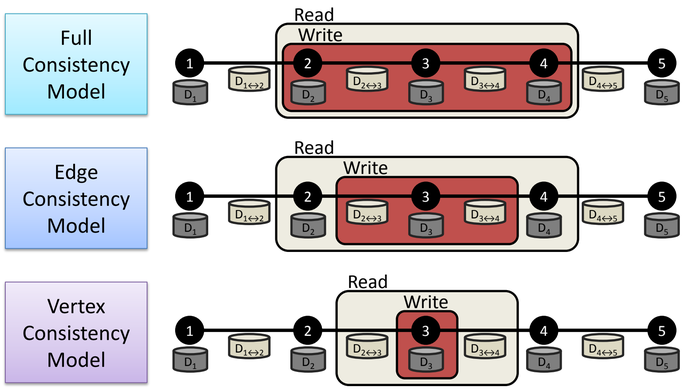
A user defined program transforms the data in the scope of a vertex.

```
struct  pagerank  : public  iupdate <graph, pagerank> {
  void operator()  (icontext_type  & context) {
    vertex_data  & vdata = context.vertex_data();
    double  sum = 0;
    foreach  ( edge_type  edge, context.in_edges() )
      sum += 1/context.num_out_edges(edge.source()) *
        context.vertex_data(edge.source()).rank;
    double  old_rank = vdata.rank;
    vdata.rank = RESET_PROB + (1-RESET_PROB) * sum;
    double  residual = abs(vdata.rank – old_rank) /
                              context.num_out_edges();
    if  (residual > EPSILON)
      context.reschedule_out_neighbors(pagerank ());
  }
};
```

**Consistency Models**

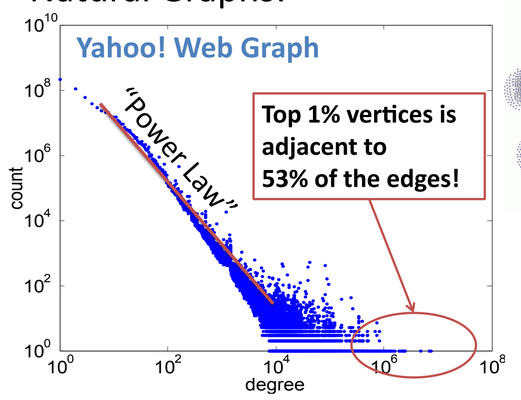
Ensures serializability:

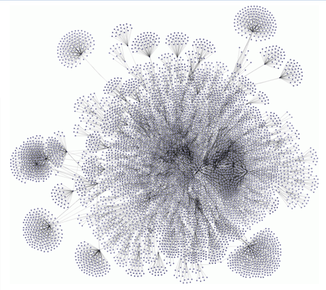


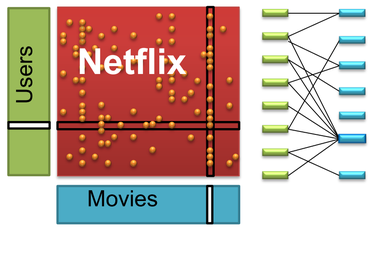




## THE CHALLENGE OF NATURAL GRAPHS

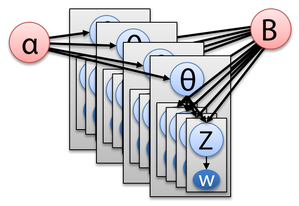
- Natural Graphs:



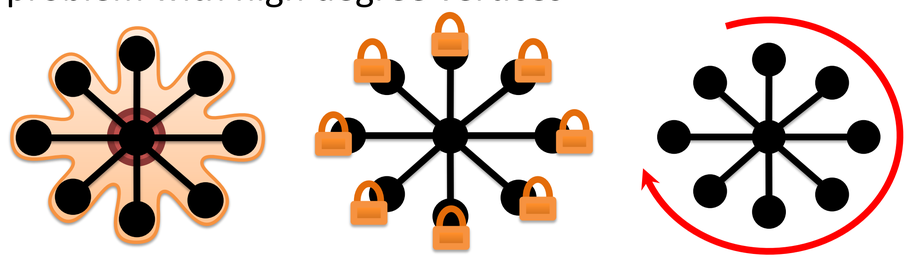







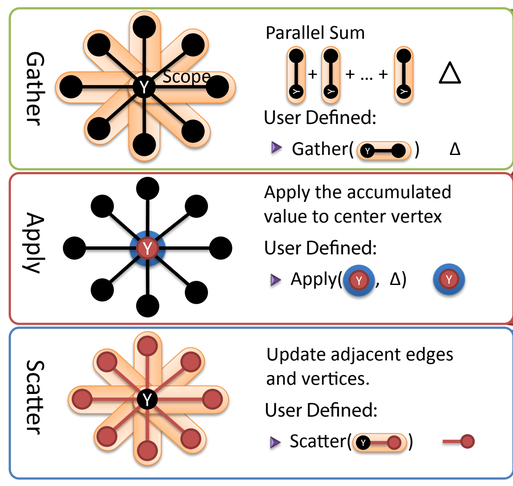



- The problem with high degree vertices




## NEW FEATURE IN GRAPHLAB2

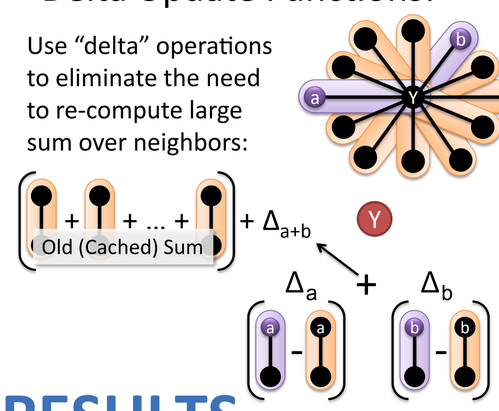
- Factorized Update Functor:




```
struct  pagerank  : public  iupdate <graph , pagerank > {
  double accum = 0 , residual = 0 ;
  void gather (icontext_type  & ctx, edge_type & edge) {
    accum += 1/ctx.num_out_edges(edge.source()) *
             ctx.vertex_data(edge.source()).rank; }
  void merge(pagerank & other) {accum += other.accum;}
  void apply (icontext_type  & context) {
    vertex_data& vdata = context.vertex_data();
    double old_value = vdata.rank;
    vdata.rank =  RESET_PROB + (1-RESET_PROB)*accum;
    residual = fabs(vdata.rank – old_value) /
               context.num_out_edges();
  }
  void scatter  (icontext_type  & ctx, edge_type & edge) {
    if (residual > EPSILON)
      ctx.schedule(edge.target(), pagerank ());
  }
};
```

- Delta Update Functions:

Use "delta" operations to eliminate the need to re-compute large sum over neighbors:

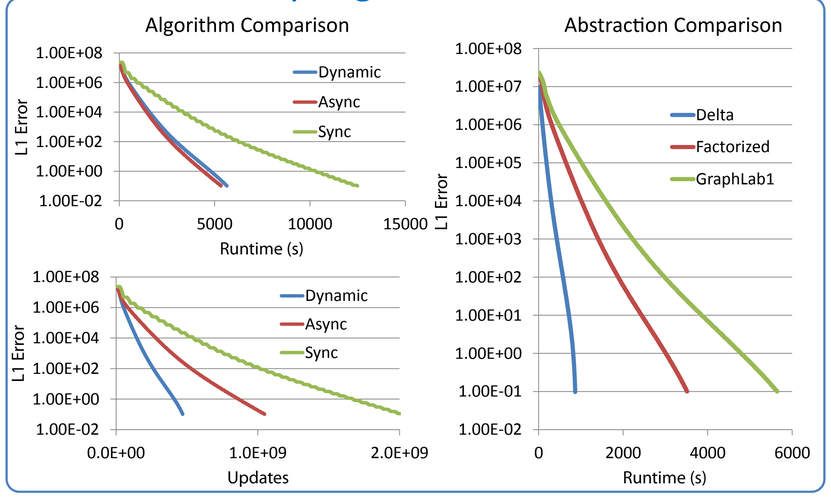



```
struct  pagerank  : public  iupdate <graph, pagerank> {
  double delta;
  pagerank (double d) : delta(d) { }
  void operator+= (pagerank & other) {
    delta += other.delta; }
  void operator()  (icontext_type  & context) {
    vertex_data  & vdata = context.vertex_data();
    vdata.rank += delta;
    if(abs(delta) > EPSILON) {
      delta = delta * (1 – RESET_PROB) *
              1/context.num_out_edges(edge.source());
    context.schedule_out_neighbors(pagerank(delta));
    } }
};
```

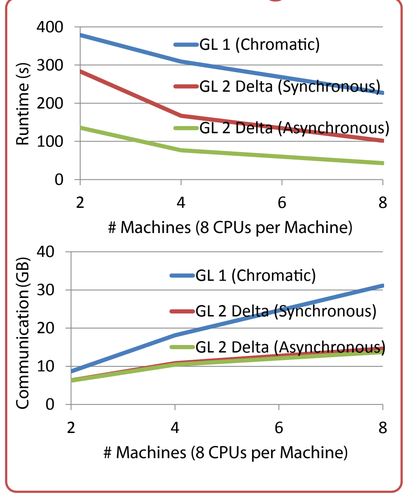
## RESULTS

**PageRank on 25.5M Vertex 355M Edge Web Graph**

Shared Memory Pagerank




Distributed Pagerank




Intel Science & Technology Center for Cloud Computing

http://graphlab.org

Carnegie Mellon University
Georgia Tech
intel
PRINCETON UNIVERSITY
UC Berkeley