UNIVERSITY OF CAMBRIDGE

Pregel: A System for Large-Scale Graph Processing

Authors: G. Malewicz, M. Austern, A. Bik, J. Dehnert, I. Horn, N. Leiser, and G. Czajkowski

Year of Publication: 2010

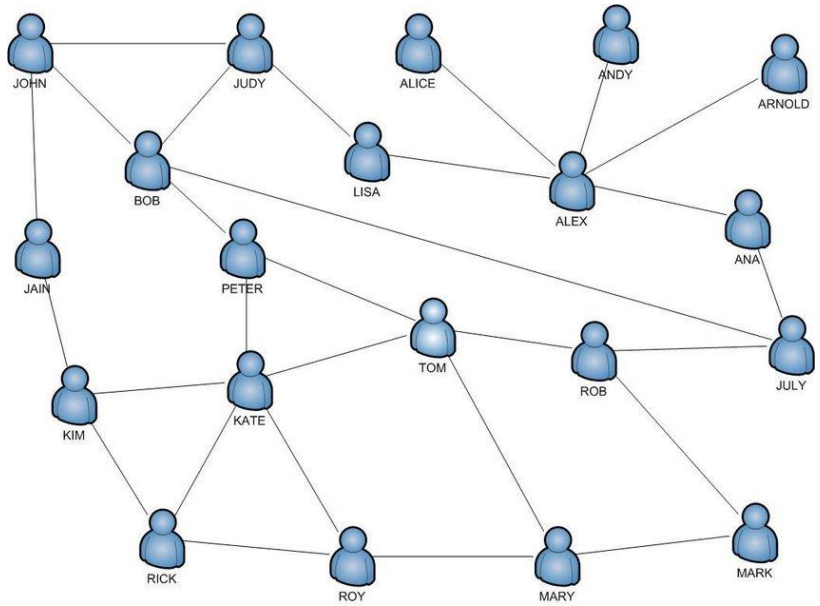
Presenter: Hetong Shen

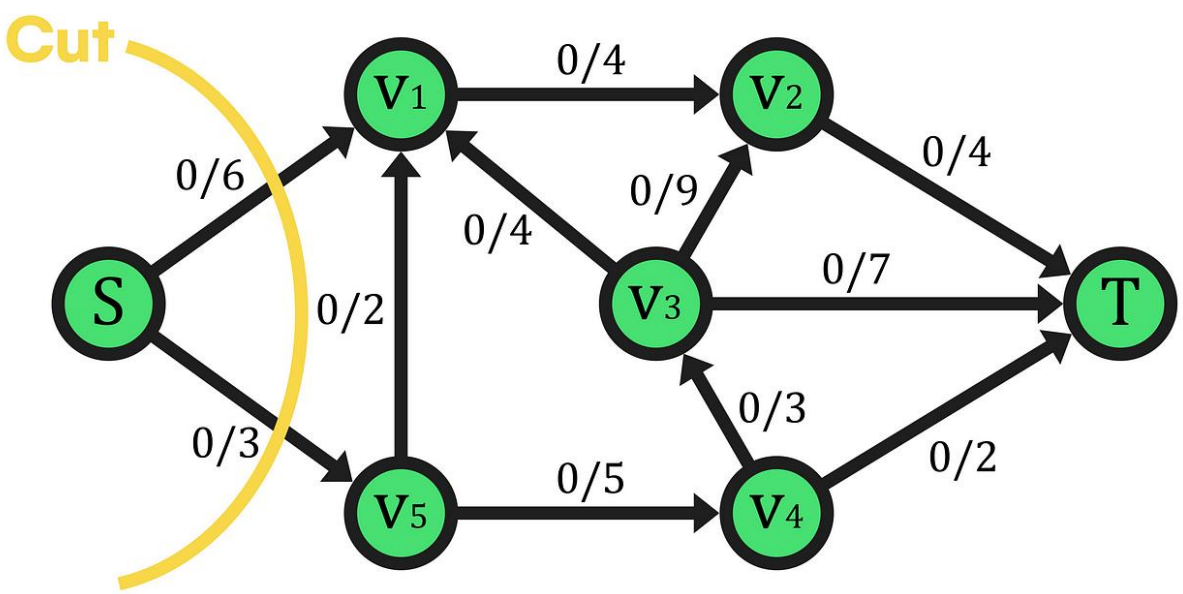
CRSid: HS899

Department of Computer Science **R244**

Subject: Large Graph

- Main motivation: Web Graph and social networks
- Other topics of large graphs: transportation routes, citation relationships, etc.
- Graph Computing problems: min-cut, SSSP, etc.





Challenges for Large Graphs

- Efficient processing of large graphs is challenging due to their sizes:
  - 1, poor locality of memory access
  - 2, very little work per vertex
  - 3, changing degree of parallelism over the course of execution

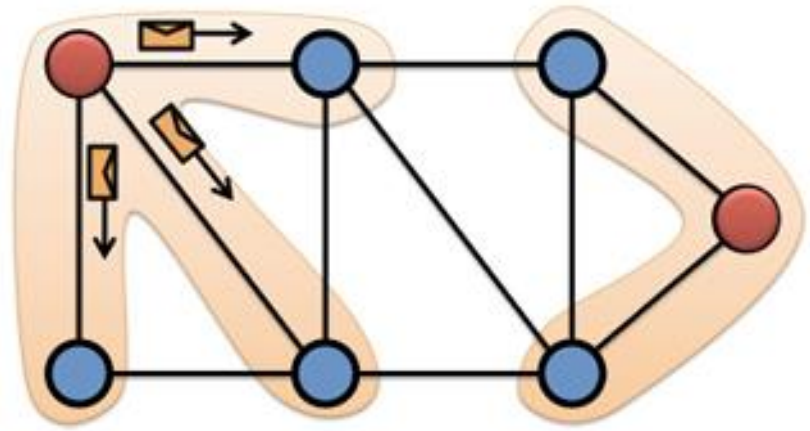
Existing Works

- Customize distributed infrastructure
  - substantial implementation effort
- Existing distributed computing platform, such as MapReduce
  - MapReduce expresses graph as a chained states
  - More communication and serialization overhead
  - More coordination needed
- Single-computer graph algorithm library, such as BGL and JDSL
  - BGL: Boost Graph Library
  - JDSL: Data Structures and Algorithms in JAVA
  - Not scalable
- Existing parallel graph system, such as Parallel BGL
  - No fault tolerance
  - Not scalable as it holds remote cells

No scalable general-purpose system for implementing arbitrary graph algorithms over arbitrary graph representations in a large-scale distributed environment!

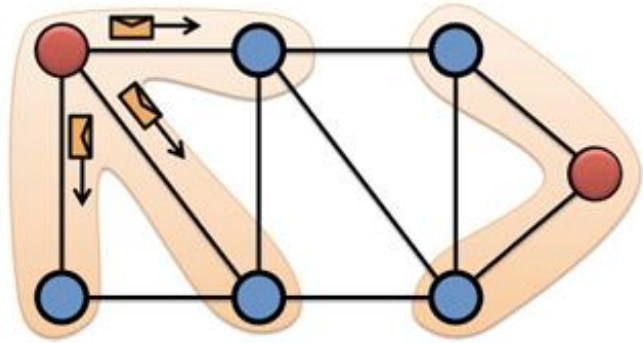
Pregel

- Programs are expressed as a sequence of iterations
- A vertex can (see next page):
  - Receive messages sent in the previous iteration
  - Send messages to other vertices
  - Modify its own state and edges
  - Mutate graph topology

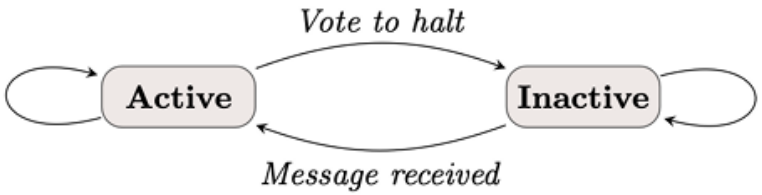


Model

- Input to Pregel: directed graph
  - Vertex and edge: modifiable, user defined value
  - Computations are done on vertices, not edges
- Sequence of iterations/Computations: Supersteps
  - Invokes a user defined function for each vertex
  - Read messages sent to vertex V in superstep S
  - Send messages to other to be received at superstep S + 1
  - Modify V and outgoing edges
  - No order of execution is detected



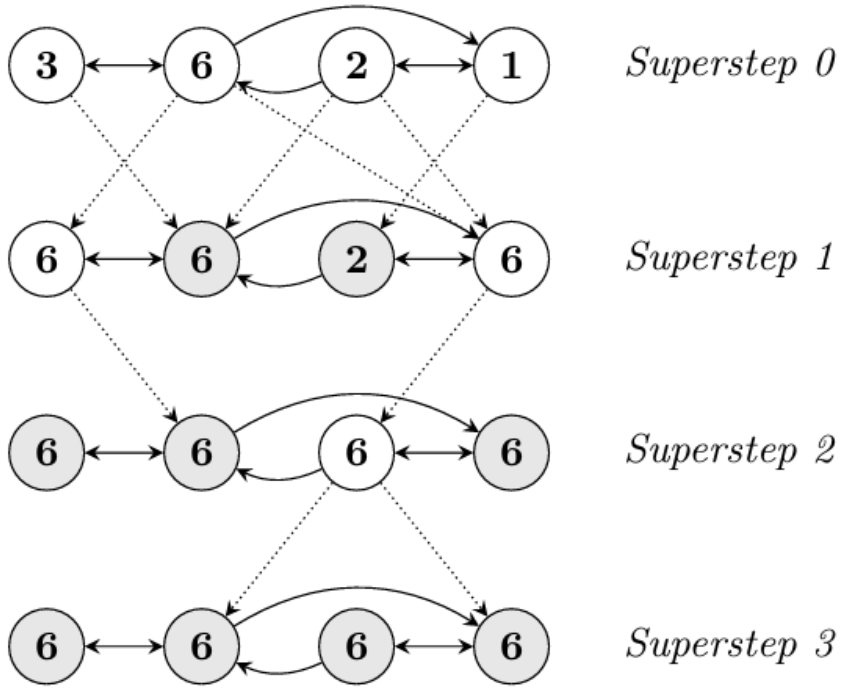
  - Superstep 0: all vertices are active
  - Worker loop through active vertices
  - When there is no work to do, the vertex is deactivated
  - Must be an external message to reactive the deactivated vertex



Model (Continue) and Example

- Termination: all vertices are deactivated.
- Output: the set of values output by the vertices
  - Very flexible
  - directed graph
  - a set of disconnected vertices

- Maximum Value Example



Architecture

- Graph divided into partitions (hash-based division/random division)
- Execute on a cluster of machines; one of them become master
- Master determines the number of partitions
- Master assign one or more partition to worker machine
- The master instructs each worker to execute supersteps
- As long as there are active vertices, the master directs the workers to proceed, and each worker responds with the count of active vertices for the next superstep
- Repeat the above step until halts
- Master instructs workers to save a portion of graph

Characteristic and Novelty of Pregel

- Vertex-centric computation
- Bulk synchronous parallel model (BSP)
- Pure message passing model, no shared memory

- Scalable, flexible, efficient, and (fault tolerant)!

Application - SSSP

- Find the shortest distance from a source vertex s to every other vertex in the graph

```
class ShortestPathVertex
    : public Vertex<int, int, int> {
  void Compute(MessageIterator* msgs) {
    int mindist = IsSource(vertex_id()) ? 0 : INF;
    for (; !msgs->Done(); msgs->Next())
      mindist = min(mindist, msgs->Value());
    if (mindist < GetValue()) {
      *MutableValue() = mindist;
      OutEdgeIterator iter = GetOutEdgeIterator();
      for (; !iter.Done(); iter.Next())
        SendMessageTo(iter.Target(),
                      mindist + iter.GetValue());
    }
    VoteToHalt();
  }
};
```

Result - SSSP Scale With Worker Tasks

- Binary tree
- Vertices = 1 billion
- Edges = 1 billion – 1
- Workers: 50 to 800

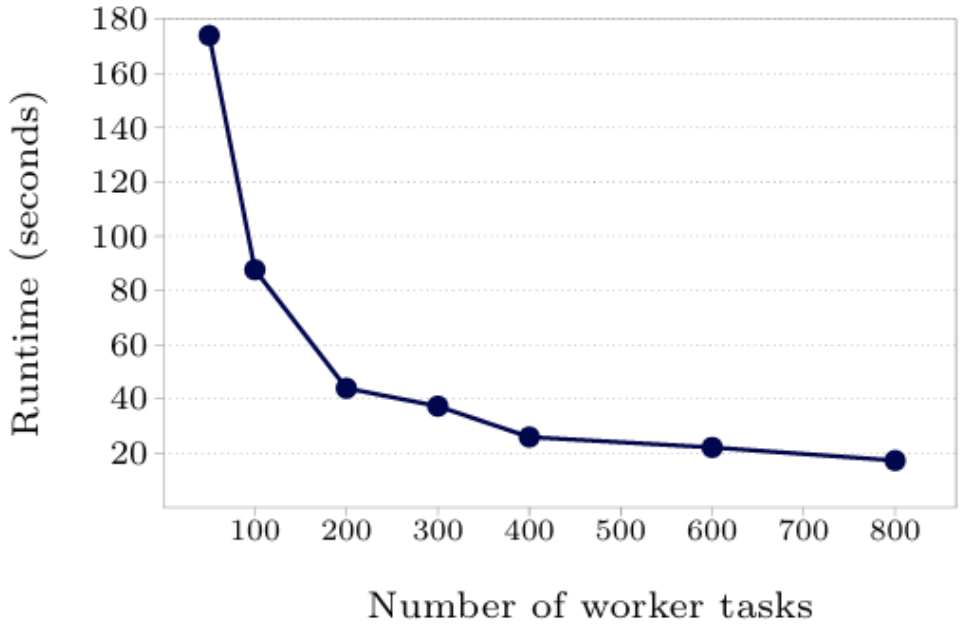


Figure 7: SSSP—1 billion vertex binary tree: varying number of worker tasks scheduled on 300 multi-core machines

Result - SSSP Scale With Graph Size

- Binary tree
- Vertices = 1 billion to 50 billion
- Edges = (1 billion – 1) to (50 billions – 1)
- Workers = 800

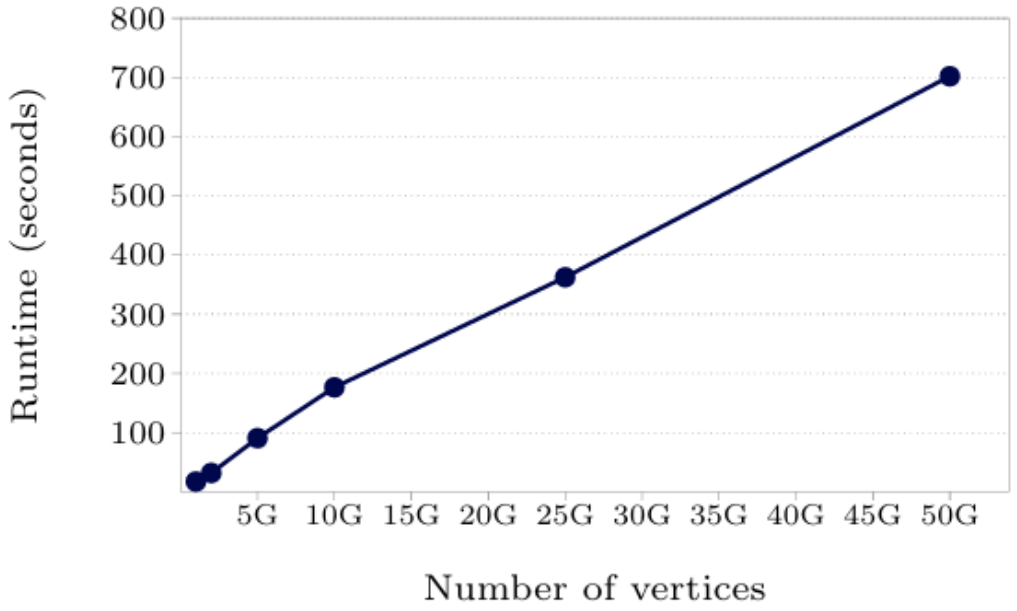
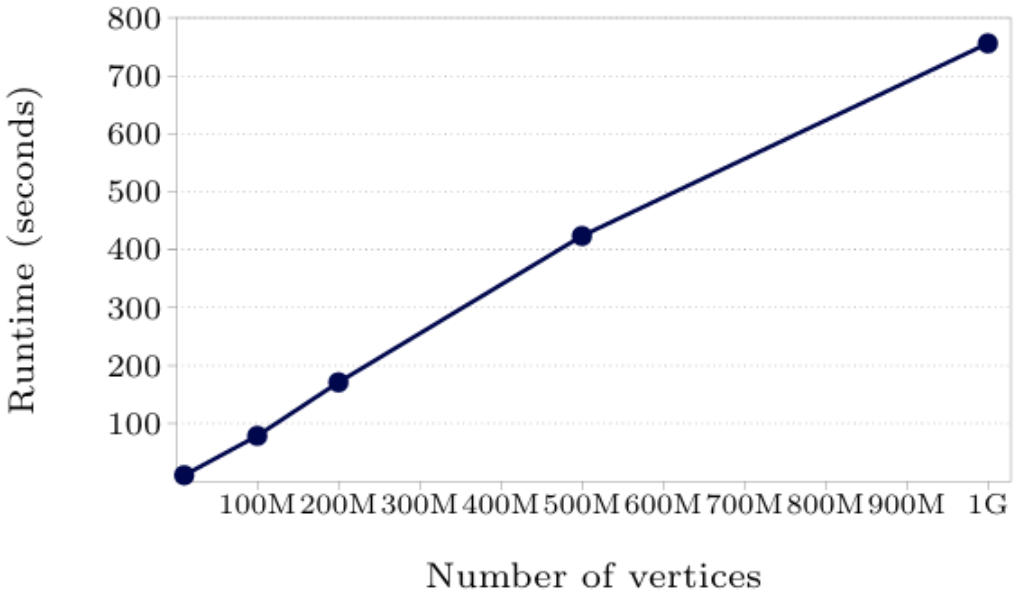


Figure 8: SSSP—binary trees: varying graph sizes on 800 worker tasks scheduled on 300 multicore machines

Result – SSSP on Log Normal Random Graphs

- Random graph
- With log-normal distribution of outdegrees
- Mean outdegree = 127.1
- Vertices = 10 million to 1 billion
- Edges = 1.27 billion to 127 billion
- Workers = 800



Strengths/Contributions of Pregel

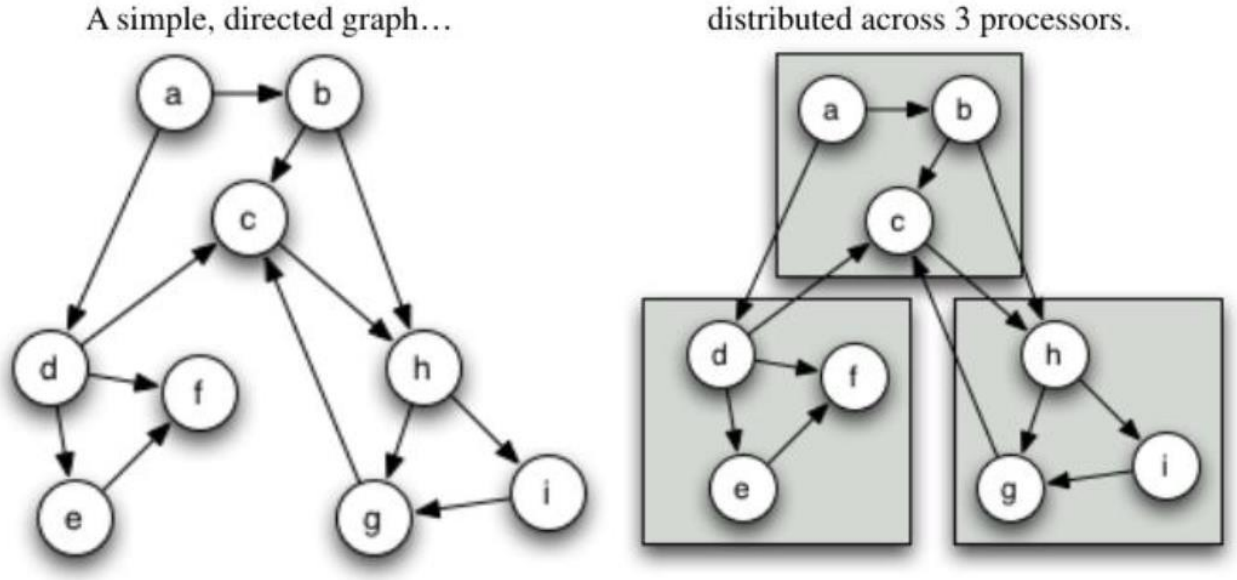
- Production quality C++ API
  - Flexible
  - Lazy evaluation
- Scalable
  - Sparse graph
  - Communication over edges
- Fault tolerance
  - Check points
  - Confined recovery

Downsides of Pregel

- API cannot be changed with liberty
  - Already a production infrastructure
- Only for sparse graph
  - Challenges of High-Degree Vertices
  - PowerGraph
- Barrier in synchronization
  - Faster workers have to wait to synchronize between supersteps
- Partition based on topology may have better performance
  - Currently, hash-based partitioning

Related Study – Parallel BGL

- Similarities
  - Message Passing Abstraction
  - Vertex-Centric Model
  - Synchronization Mechanisms



Related Study – Parallel BGL (Continue)

Downside of Parallel BGL

- Property maps to hold information associated with vertices and edges in the graph
  - Ex: the property map that access the distance of any vertex in the graph G

    property_map <Graph, vertex_distance_t>::type distance = get(vertex distance, G)
- Ghost cells to hold values associated with remote components
  - Cache values for vertices in other processes

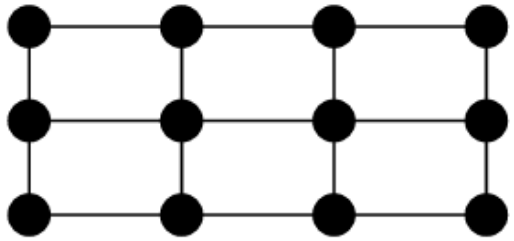
Pregel is better in a way that…

- Better with scaling: Pregel has an explicit message approach to acquiring remote information and does not replicate remote values locally

Further Study - PowerGraph

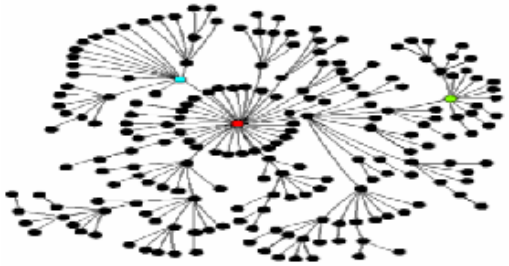
- The graph with high degree vertices: Power-Law Graphs
- Ex: celebrity

Assumed Structure



- ***Small*** neighborhoods
- ***Similar*** degree vertices
- Easy to ***partition***

Natural Graph



- ***Large*** Neighborhoods
- ***Power-Law Degree***
- Difficult to partition

PowerGraph!

- GAS Decomposition
- 3 Vertex partitioning Strategies

Summary/Takeaway Message (Optional)

- Challenges - no efficient computations for large graphs
  - Custom distributed infrastructure – cost too high
  - Existing distributed computing platform – not for graph
  - Single-computer graph algorithm libraries – limited scope
  - Existing parallel graph systems - no fault tolerance
- Solution: Pregel
  - A scalable general-purpose system for implementing arbitrary graph algorithms over arbitrary graph representations in a large-scale distributed environment
- Related systems
  - Parallel BGL (comparison)
  - PowerGraph (advancement?)