Processing large graphs in parallel

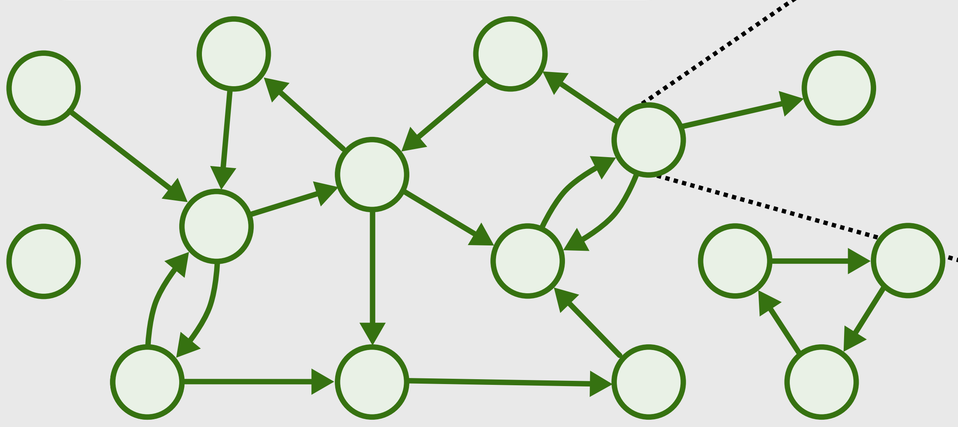
E. Krepska, T. Kielmann, W. Fokkink, H.Bal

VU University Amsterdam, Netherlands

HipG A framework to easily write distributed graph algorithms

- Push-button automatic parallelization
- Easy programming (exposed vertex/edge)
- Structure-driven fine-grained computations
- Handles billions of vertices and edges
- Efficient w.r.t. memory and computation

User implements graph vertices:



MyVertex

```
// custom data
boolean visited;

// custom methods
visit();
```

Visits all reachable vertices

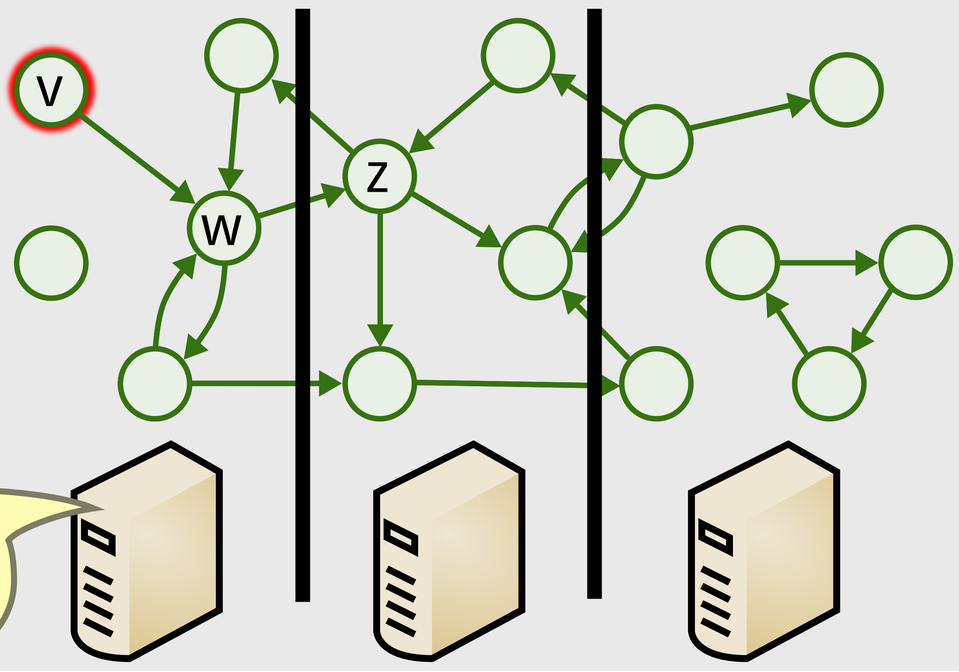
Java

```
public void visit() {
  if (!visited) {
    visited = true;
    while(hasNeighbor(i))
      neighbor(i++).visit();
  }
}
```

Gives work to other vertices

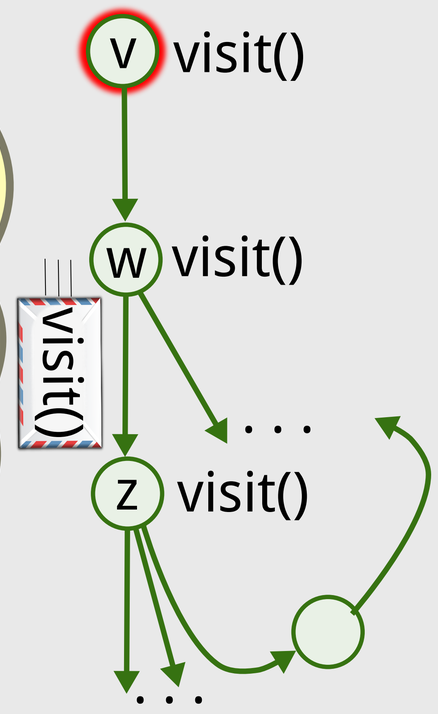
This is sequential code!

Automatic parallelization for a distributed computer:



Vertices distributed between memories of many machines

Methods on non-local vertices translated into asynchronous messages

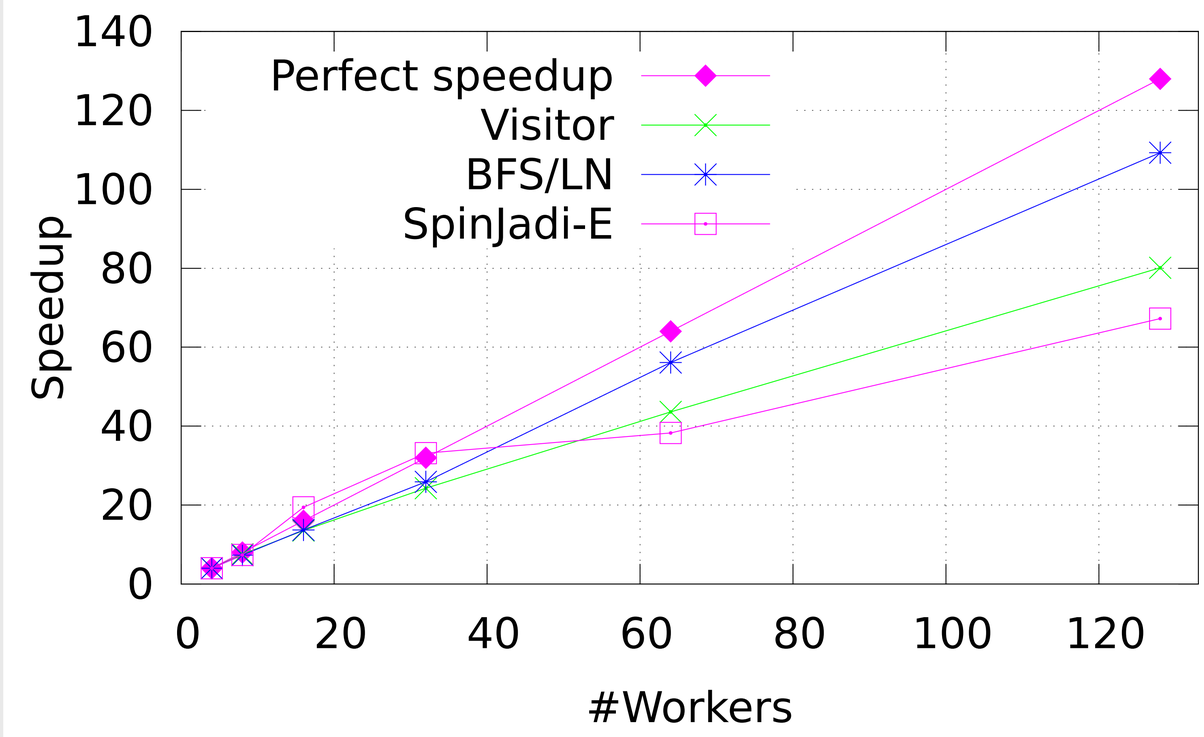


Termination detection:

```
v.visit();
barrier();
```

Results & examples

- Breadth-first search (BFS)
- Model checking (SpinJadi)
- Genetic networks



Speedup of selected HipG applications run using 4-128 workers (2 per machine) on the DAS-4/VU cluster, on graphs with up to 10^{10} of vertices and edges. On 64 machines obtained efficiency of 60-80%.

More features:

- Global operations
- Divide-and-conquer graph computations
- On-the-fly computations (concurrent with graph generation)

Talk to us!

We're looking for new applications

Contact: **e.krepska@vu.nl** Download/papers: **http://www.cs.vu.nl/~ekr/HipG**