signal collect

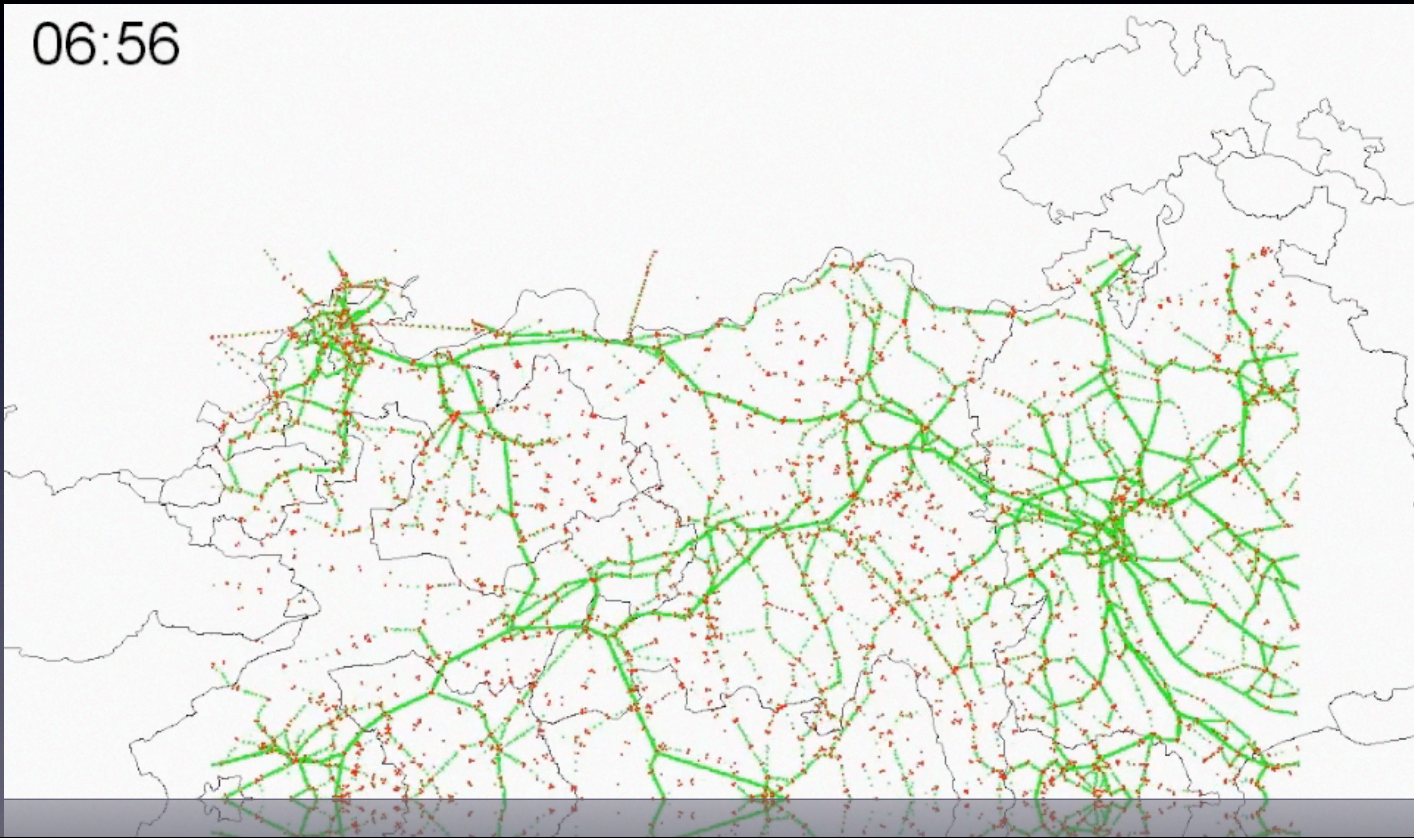
Processing Web-Scale Graphs in Seconds

Abraham Bernstein, Philip Stutz, Mihaela Verman

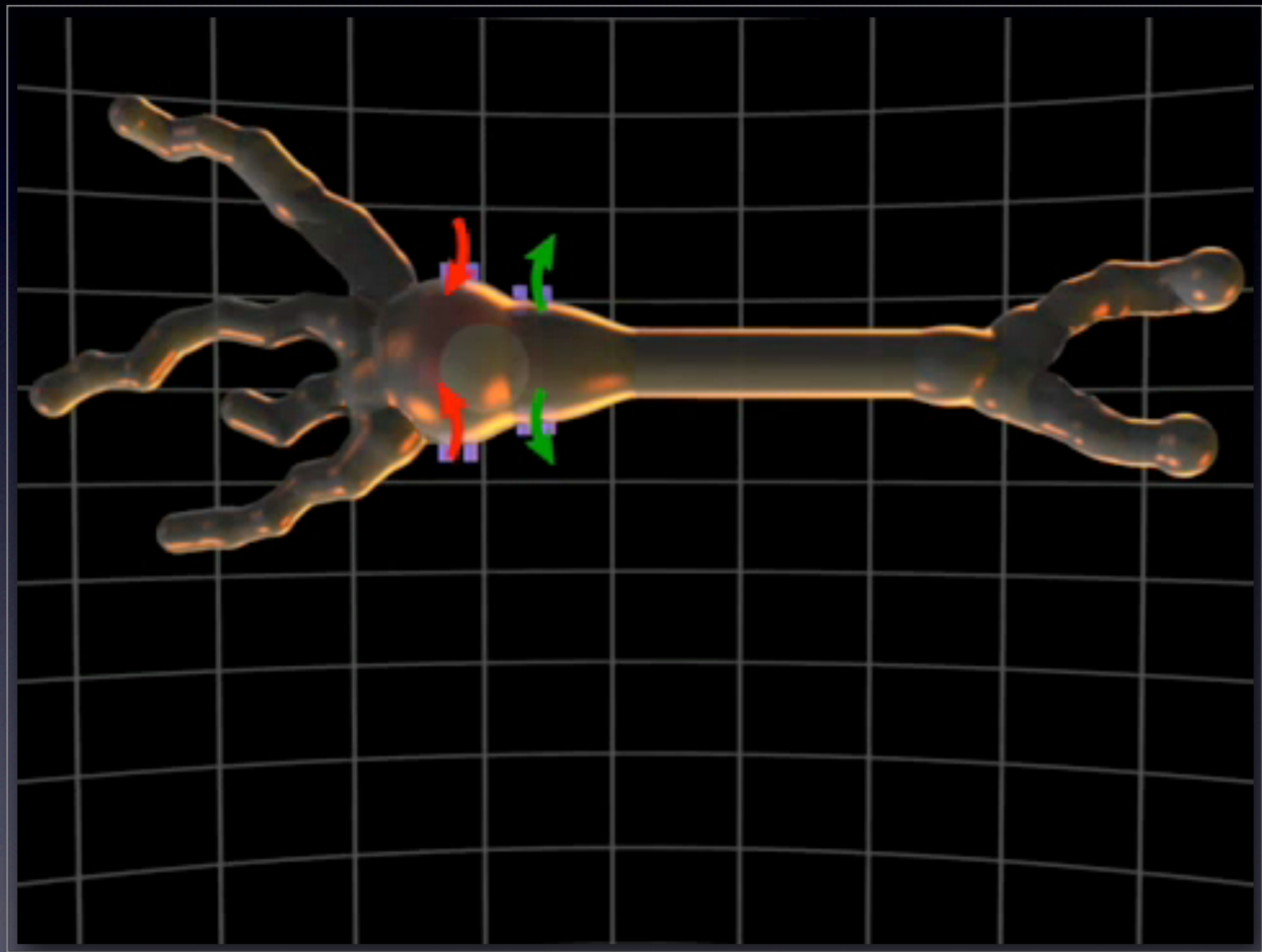
DDIS Dynamic and Distributed Information Systems

University of Zurich UZH

Graphs in a Smart World



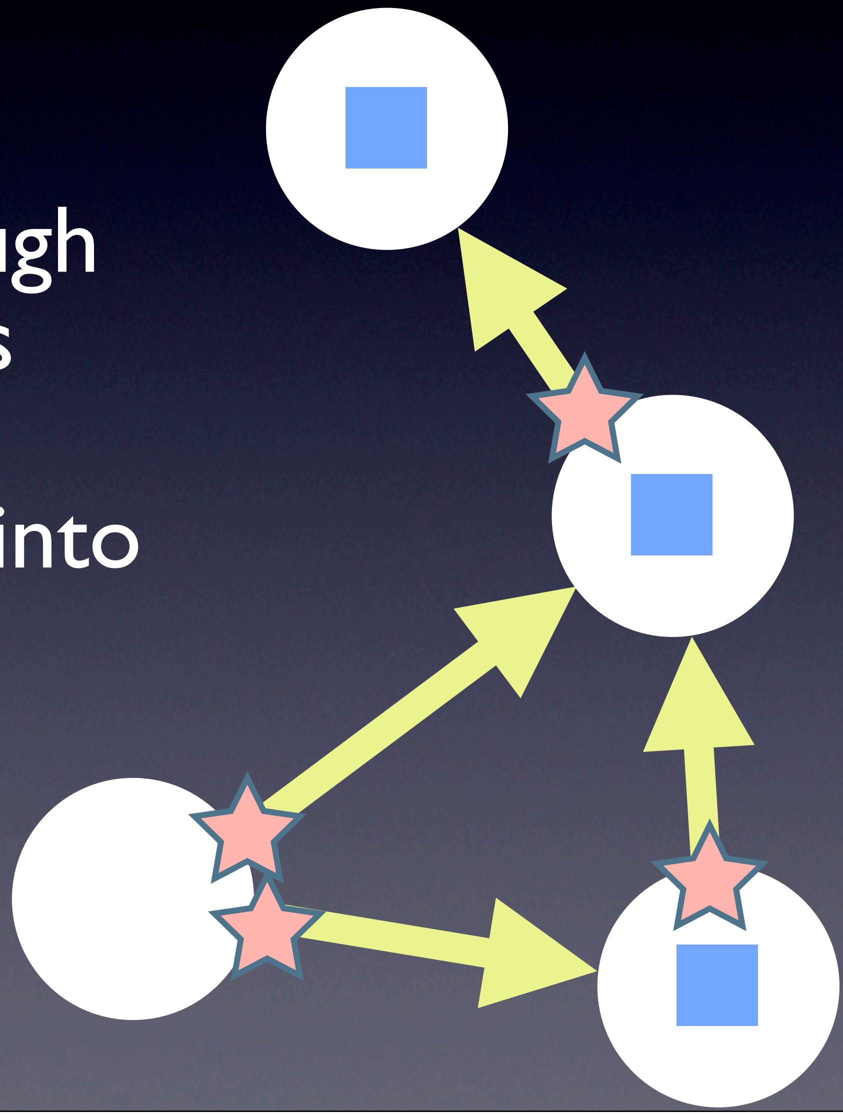
Processing Graphs Naturally



Processing Graphs Naturally
Vertices as Actors

Vertices interact through **signals** along edges

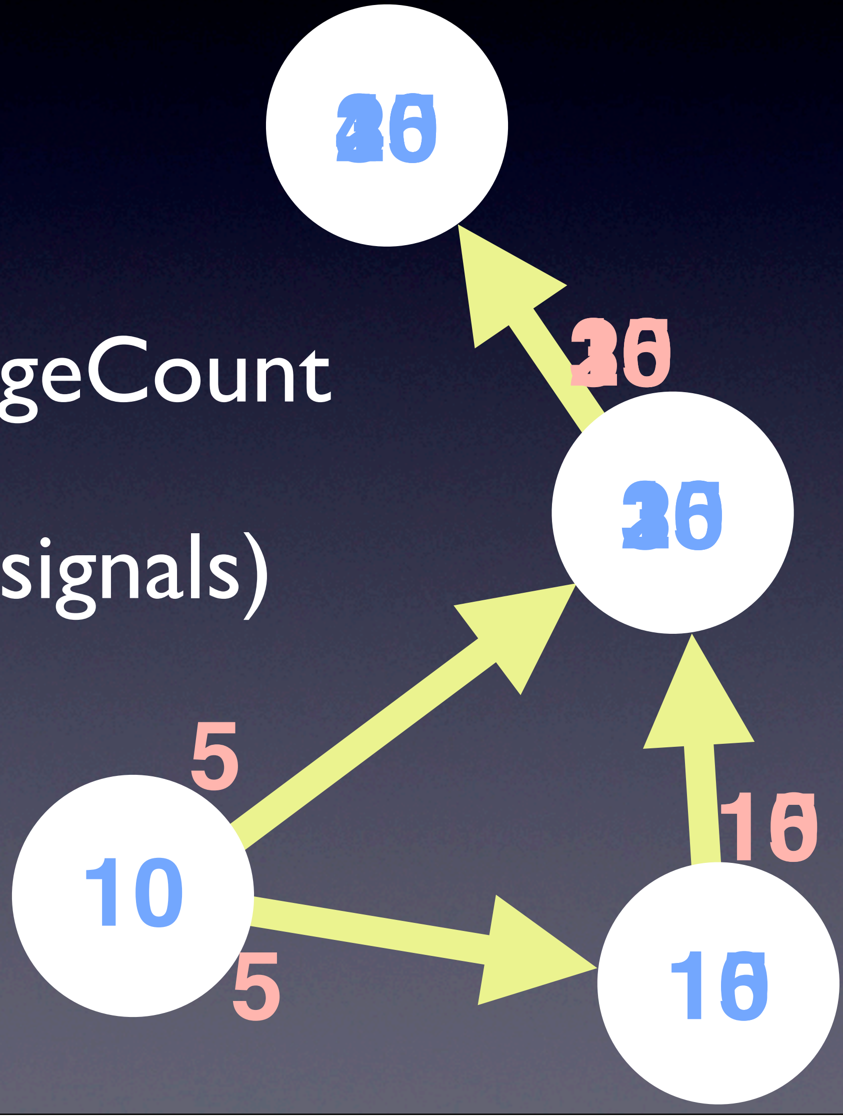
Which are **collected** into new vertex states



Simplified PageRank

signal = state / outEdgeCount

collect = 10 + sum(signals)



Example Algorithms

PageRank (Data-Graph)	
`initialState`	`baseRank`
`collect(...)`	`return baseRank + dampingFactor * sum(signals)`
`signal(...)`	`return source.state * edge.weight / sum(edgeWeights(source))`

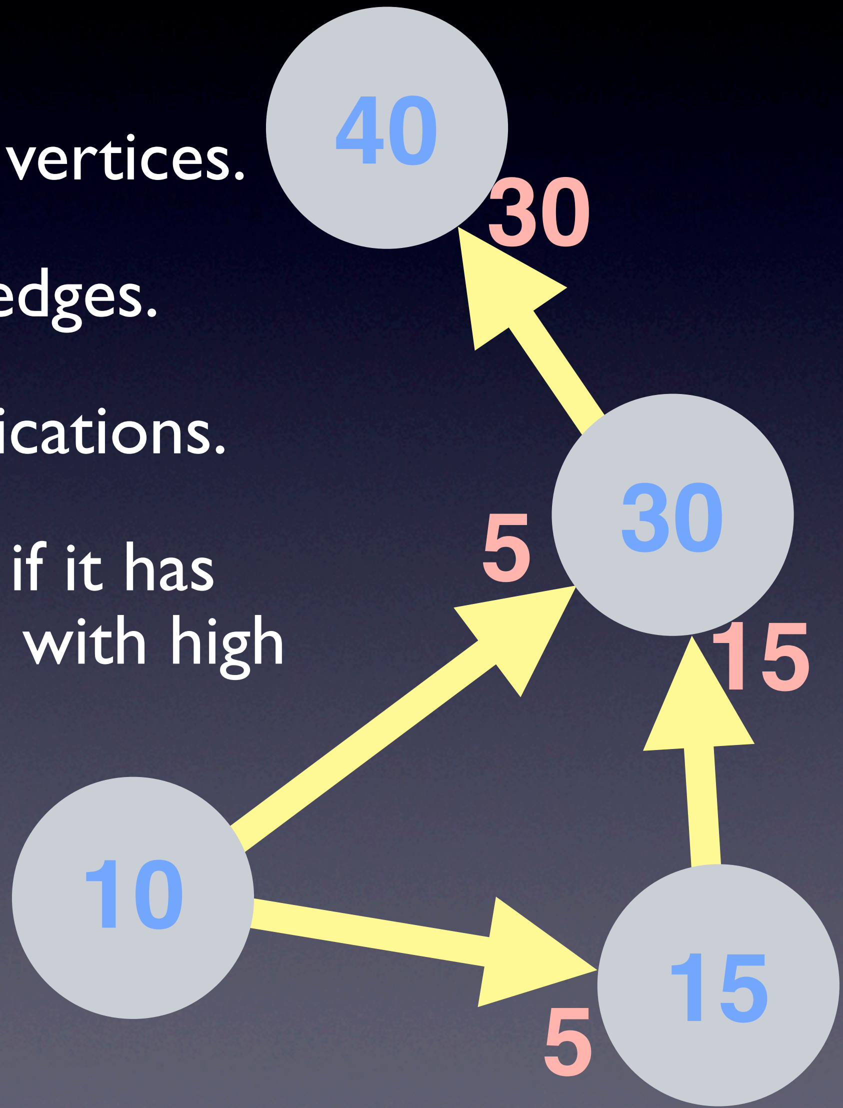
Citation Graph

Publications represented as vertices.

Citations represented as edges.

Use simplified PR to rank publications.

"A publication has a high rank if it has citations from other publications with high ranks."



Example Problem

Goal: Implement simplified PageRank to identify the publications with the highest ranks in the dataset.

Dataset from **DBLP** with 114 657 citations between 26 907 ids of computer science publications.

Represented as LOD triples:

publication1URL	citesURL	publication2URL .
publication1URL	citesURL	publication3URL .
publication3URL	citesURL	publication5URL .

...

Source: http://dblp.l3s.de/dblp++.php

Code for Example

```scala
import com.signalcollect._
import java.io.FileInputStream
import org.semanticweb.yars.nx.parser.NxParser

class Publication(id: String, initialState: Double = 10) extends
DataGraphVertex(id, initialState) {
  type Signal = Double
  def collect = initialState + signals.sum
}

class Citation(targetId: String) extends DefaultEdge(targetId) {
  type Source = Publication
  def signal = source.state / source.outgoingEdges.size
}
```

```scala
object Example extends App {
  val graph = GraphBuilder.build
  Parser.processCitations("./citations.nt", processCitation)
```

Results

(only 1 entry per author)

1

2

3

A Relational Model of Data for Large Shared Data Banks

E. F. Codd
IBM Research Laboratory, San Jose, California

Future users of large data banks must be protected from having to know how the data is organized in the machine (the internal representation). A prompting service which supplies such information is not a satisfactory solution. Activities of users at terminals and most application programs should remain unaffected when the internal representation of data is changed and even when some aspects of the external representation are changed. Changes in data representation will often be needed as a result of changes in query, update, and report traffic and natural growth in the types of stored information.

Existing noninferential, formatted data systems provide users with tree-structured files or slightly more general network models of the data. In Section 1, inadequacies of these models are discussed. A model based on *n*-ary relations, a normal form for data base relations, and the concept of a universal data sublanguage are introduced. In Section 2, certain operations on relations (other than logical inference) are discussed and applied to the problems of redundancy and consistency in the user's model.

The Entity-Relationship Model—Toward a Unified View of Data

PETER PIN-SHAN CHEN
Massachusetts Institute of Technology

A data model, called the entity-relationship model, is proposed. This model incorporates some of the important semantic information about the real world. A special diagrammatic technique is introduced as a tool for database design. An example of database design and description using the model and the diagrammatic technique is given. Some implications for data integrity, information retrieval, and data manipulation are discussed.

The entity-relationship model can be used as a basis for unification of different views of data: the network model, the relational model, and the entity set model. Semantic ambiguities in these models are analyzed. Possible ways to derive their views of data from the entity-relationship model are presented.

Key Words and Phrases: database design, logical view of data, semantics of data, data models, entity-relationship model, relational model, Data Base Task Group, network model, entity set model, data definition and manipulation, data integrity and consistency

System R: Relational Approach to Database Management

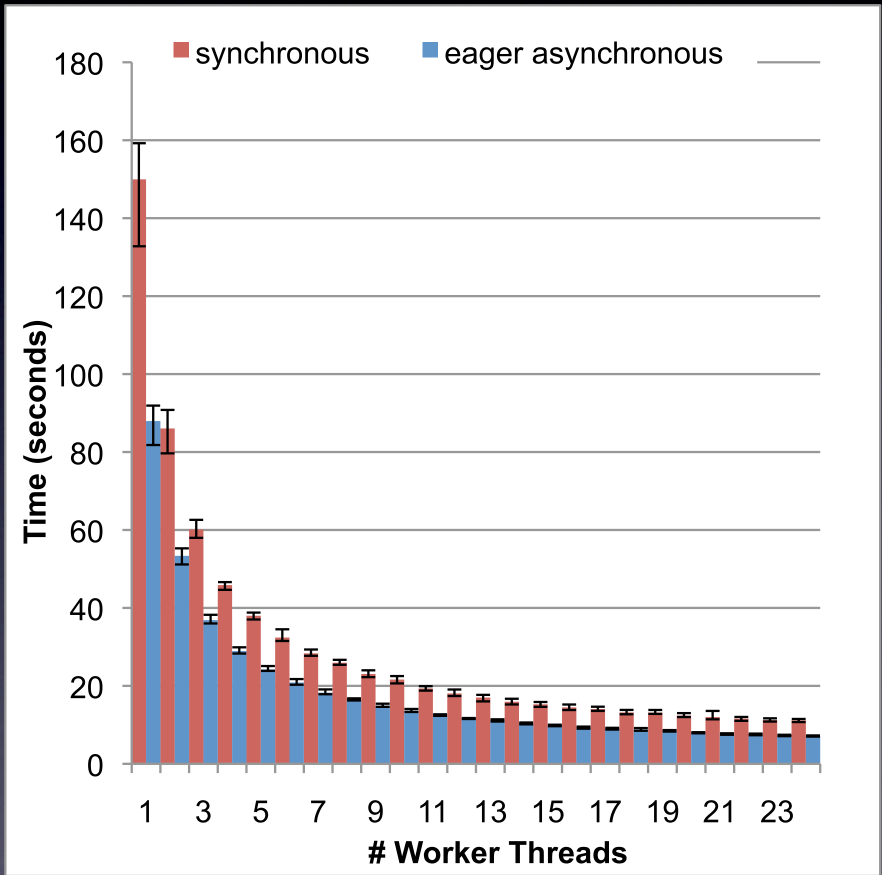
M. M. ASTRAHAN, M. W. BLASGEN, D. D. CHAMBERLIN,
K. P. ESWARAN, J. N. GRAY, P. P. GRIFFITHS,
W. F. KING, R. A. LORIE, P. R. MCJONES, J. W. MEHL,
G. R. PUTZOLU, I. L. TRAIGER, B. W. WADE, AND V. WATSON
IBM Research Laboratory

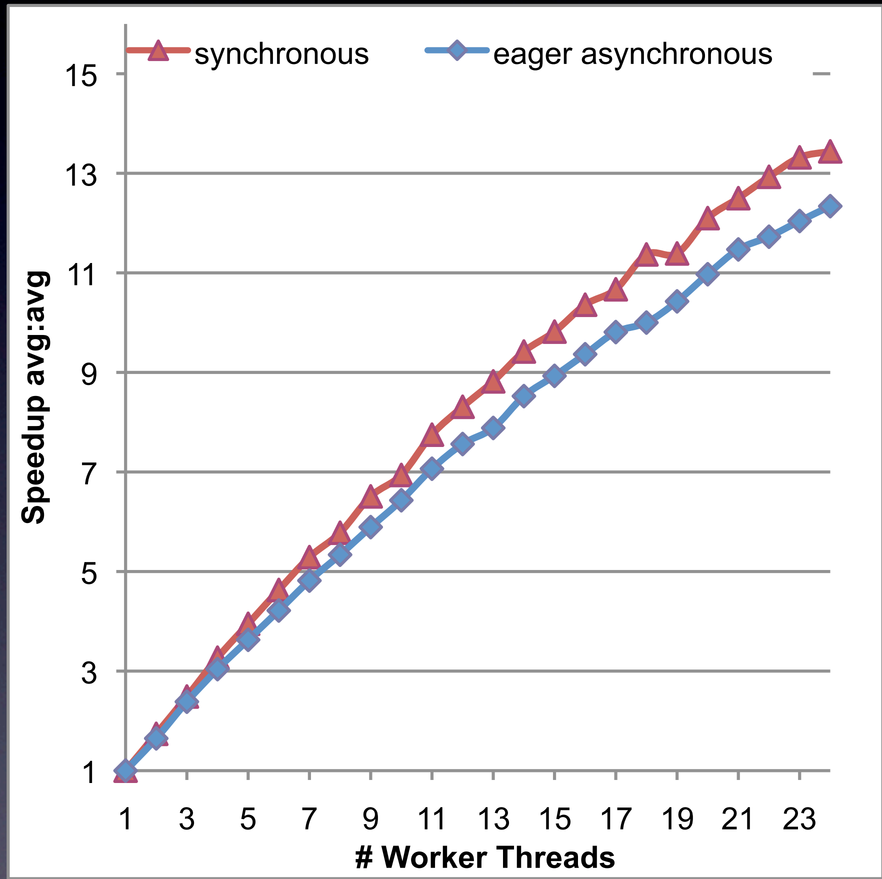
System R is a database management system which provides a high level relational data interface. The system provides a high level of data independence by isolating the end user as much as possible from underlying storage structures. The system permits definition of a variety of relational views on common underlying data. Data control features are provided, including authorization, integrity assertions, triggered transactions, a logging and recovery subsystem, and facilities for maintaining data consistency in a shared-update environment.

This paper contains a description of the overall architecture and design of the system. At the present time the system is being implemented and the design evaluated. We emphasize that System R is a vehicle for research in database architecture, and is not planned as a product.

Key Words and Phrases: database, relational model, nonprocedural language, authorization, locking, recovery, data structures, index structures
CR categories: 3.74, 4.22, 4.33, 4.35

Evaluation
Scalability on one machine



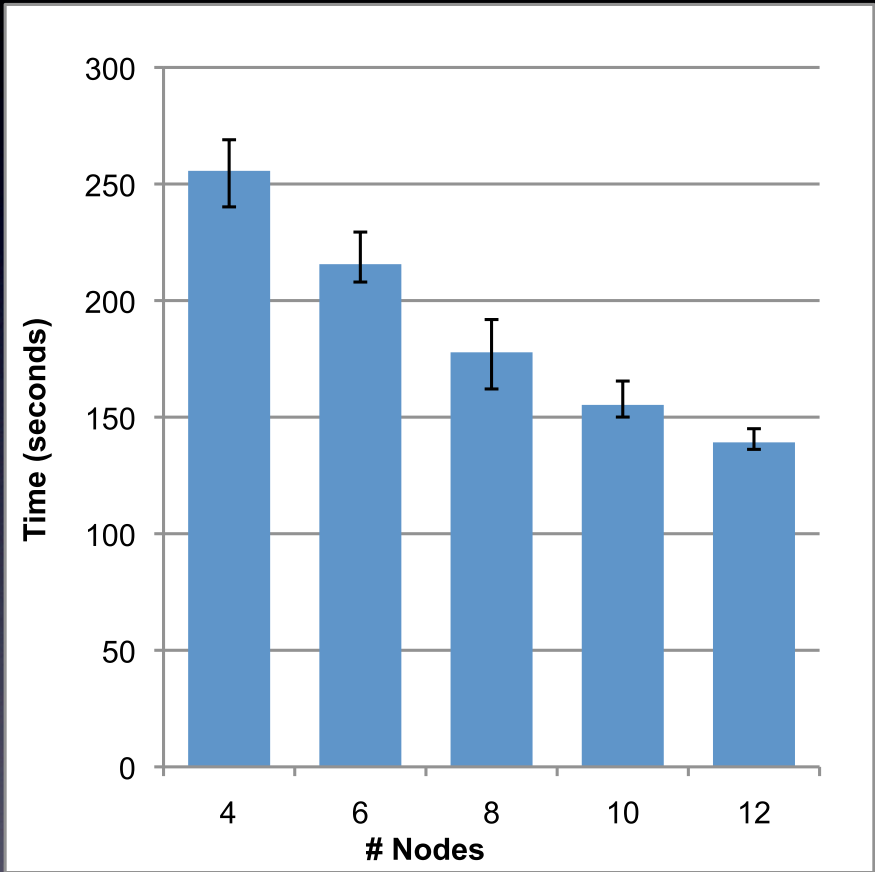


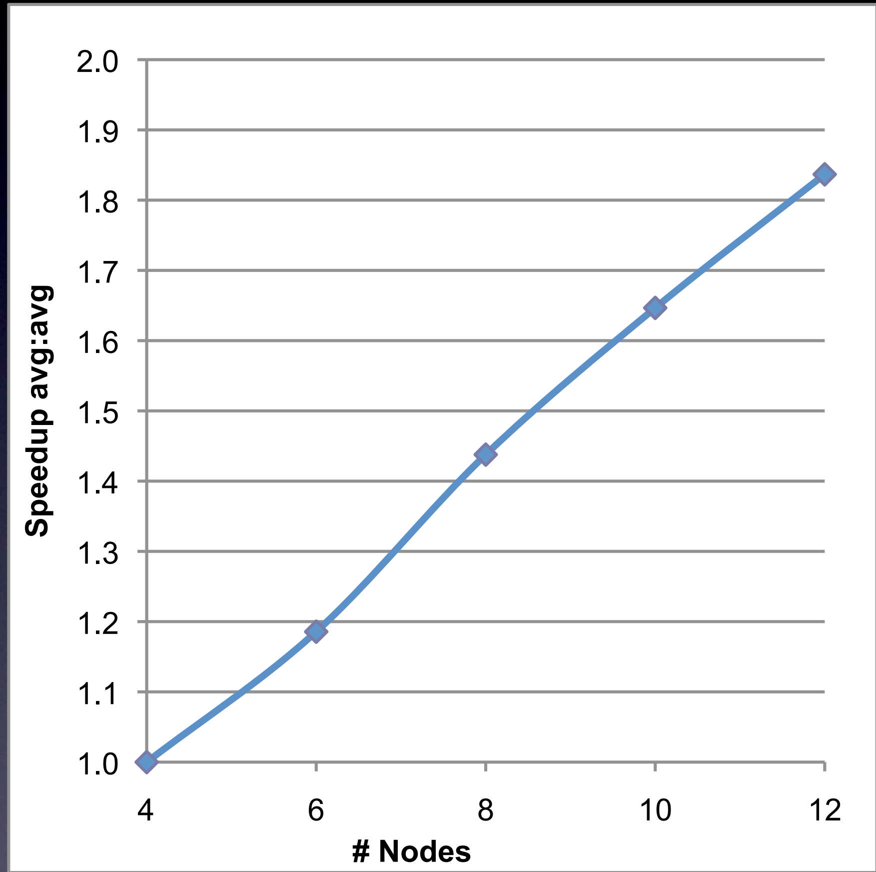
PageRank on web graph dataset, 875 713 vertices (websites) and 5 105 039 edges (links)

Machine with two twelve-core AMD Operon 6174 processors and 66 GB RAM

Evaluation

Scalability on a cluster



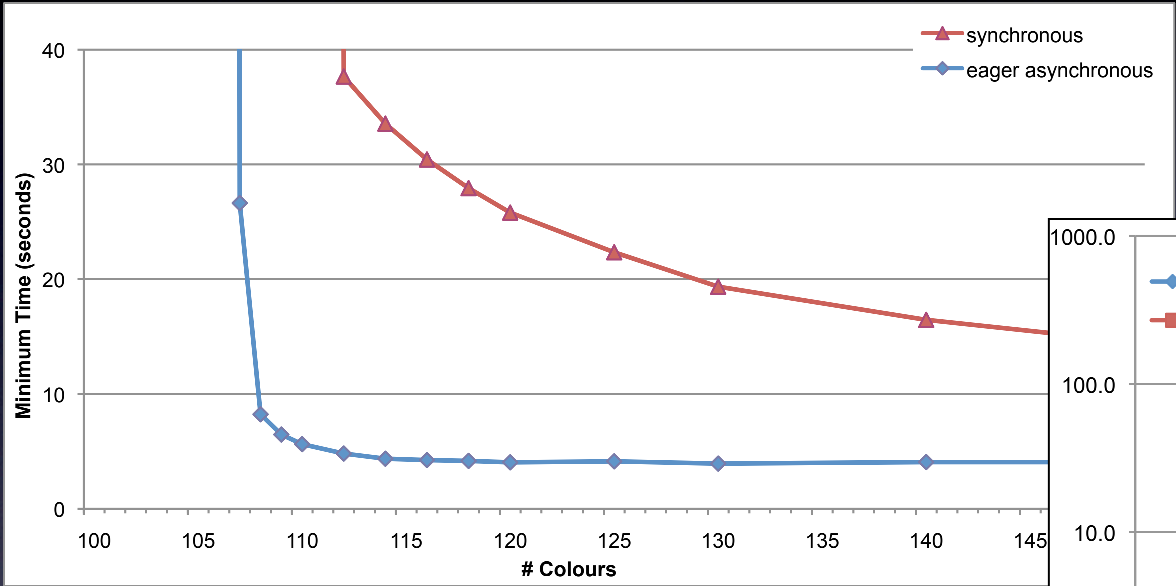


PageRank on 12 machines (24 cores, 66GB RAM each)

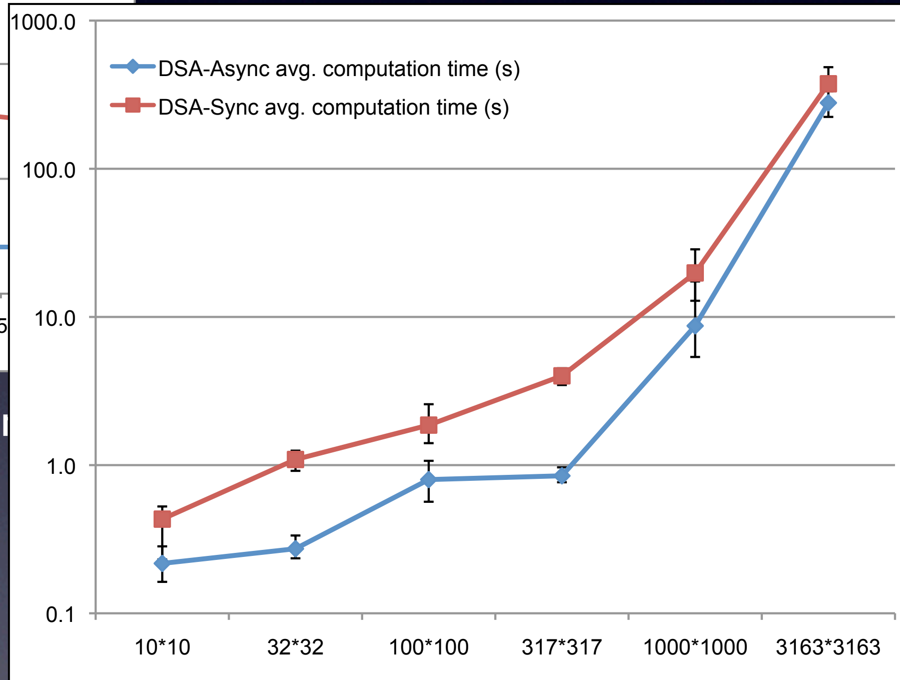
> 1.4 billion vertices, > 6.6 billion edges, 12 machines (24 cores, 66GB RAM each)

Fastest execution time (fully converged): 137s, loading time: 45s

Evaluation
Asynchronous vs. Synchronous



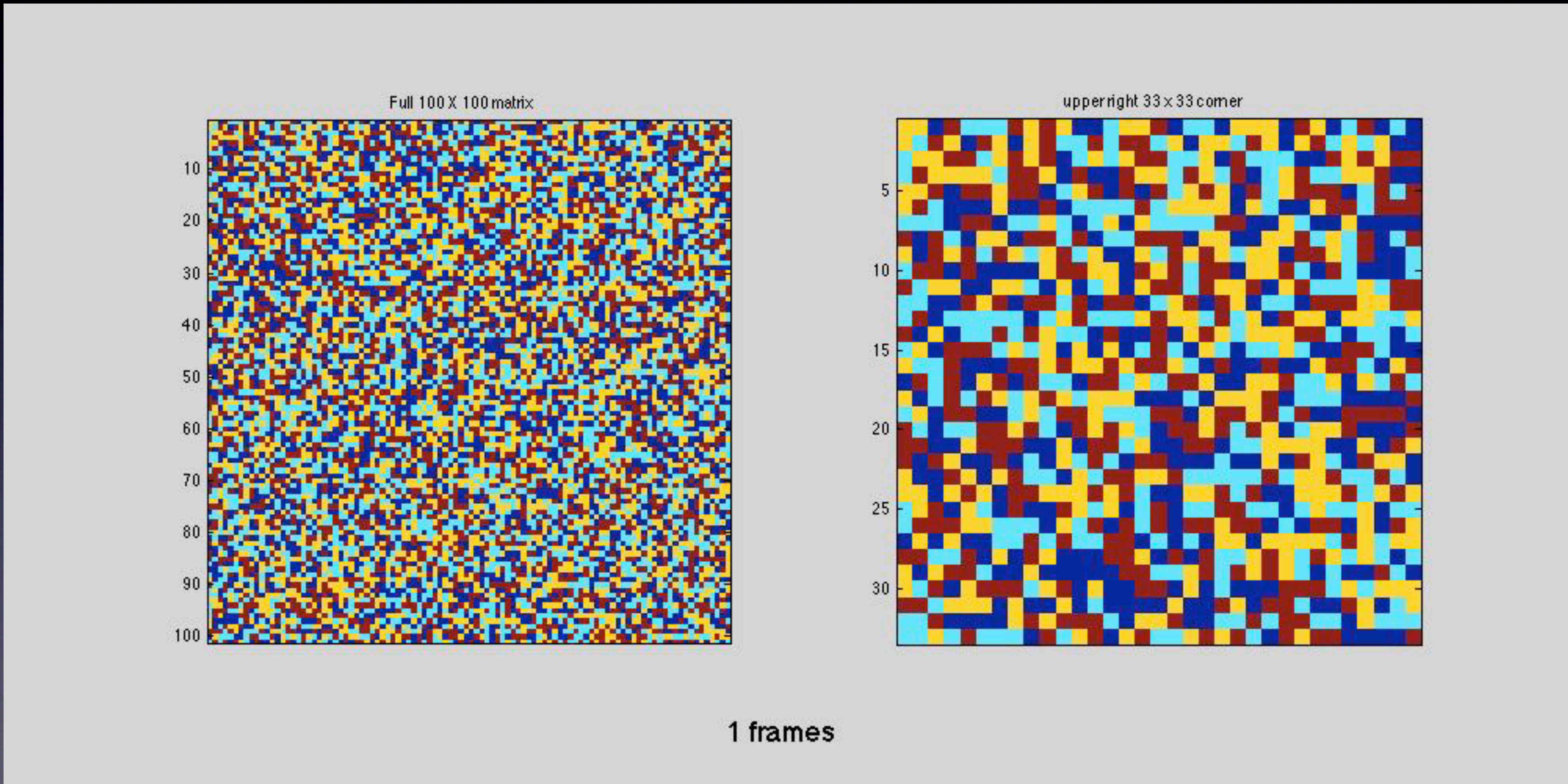
Simple greedy algorithm for solving vertex colori
problem on a 100x100 latin square.



Average computation time over 10 runs for a 6 coloring on grids of varying sizes

Machine with two twelve-core AMD Operon 6174 processors and 66 GB RAM

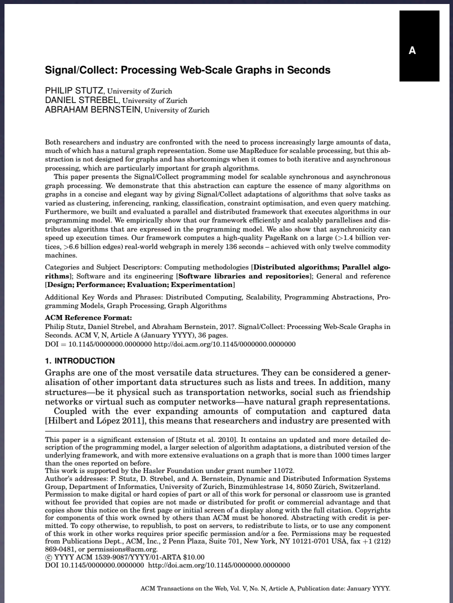
Vertex Coloring in Action



Optimized Version of DSA Running on a MacBoo Pro with 8 workers
(slow, due to lots of IO for logging, bookkeeping, etc.)

Dissemination

- Scientific publications (under review)
- Open Source Software on GitHub
- FOSDEM'13



A

Signal/Collect: Processing Web-Scale Graphs in Seconds

PHILIP STUTZ, University of Zurich
DANIEL STREBEL, University of Zurich
ABRAHAM BERNSTEIN, University of Zurich

Both researchers and industry are confronted with the need to process increasingly large amounts of data, much of which has a natural graph representation. Some use MapReduce for scalable processing, but this abstraction is not designed for graphs and has shortcomings when it comes to both iterative and asynchronous processing, which are particularly important for graph algorithms.

This paper presents the Signal/Collect programming model for scalable synchronous and asynchronous graph processing. We demonstrate that this abstraction can capture the essence of many algorithms on graphs in a concise and elegant way by giving Signal/Collect adaptations of algorithms that solve tasks as varied as clustering, inferencing, ranking, classification, constraint optimisation, and even query matching. Furthermore, we built and evaluated a parallel and distributed framework that executes algorithms in our programming model. We empirically show that our framework efficiently and scalably parallelises and distributes algorithms that are expressed in the programming model. We also show that asynchronicity can speed up execution times. Our framework computes a high-quality PageRank on a large (>1.4 billion vertices, >6.6 billion edges) real-world webgraph in merely 136 seconds – achieved with only twelve commodity machines.

Categories and Subject Descriptors: Computing methodologies [**Distributed algorithms; Parallel algorithms**]; Software and its engineering [**Software libraries and repositories**]; General and reference [**Design; Performance; Evaluation; Experimentation**]

Additional Key Words and Phrases: Distributed Computing, Scalability, Programming Abstractions, Programming Models, Graph Processing, Graph Algorithms

ACM Reference Format:
Philip Stutz, Daniel Strebel, and Abraham Bernstein, 201?. Signal/Collect: Processing Web-Scale Graphs in Seconds. ACM V, N, Article A (January YYYY), 36 pages.
DOI = 10.1145/0000000.0000000 http://doi.acm.org/10.1145/0000000.0000000

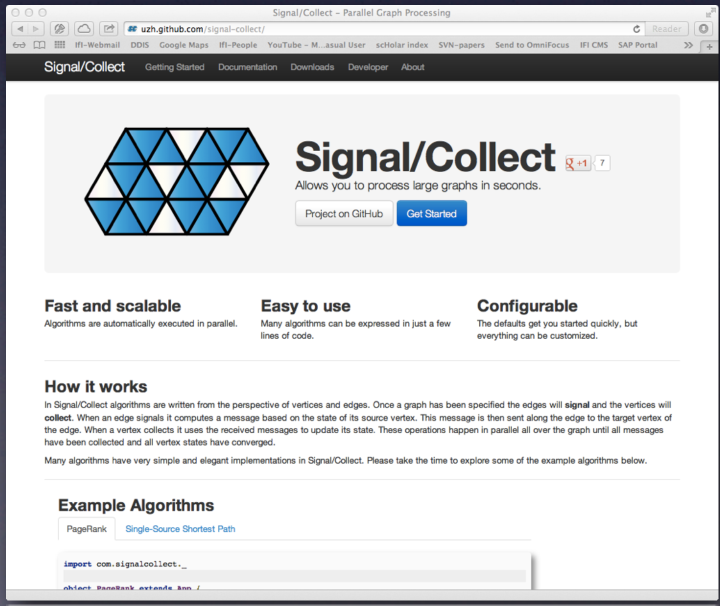
1. INTRODUCTION

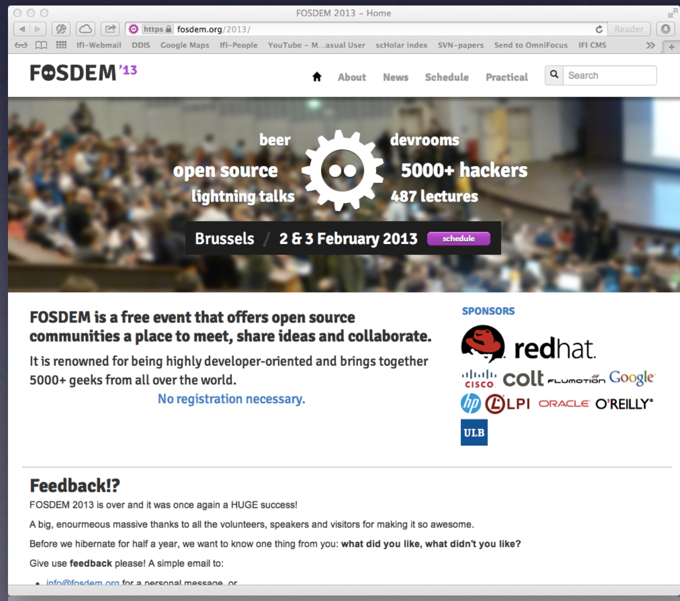
Graphs are one of the most versatile data structures. They can be considered a generalisation of other important data structures such as lists and trees. In addition, many structures—be it physical such as transportation networks, social such as friendship networks or virtual such as computer networks—have natural graph representations.

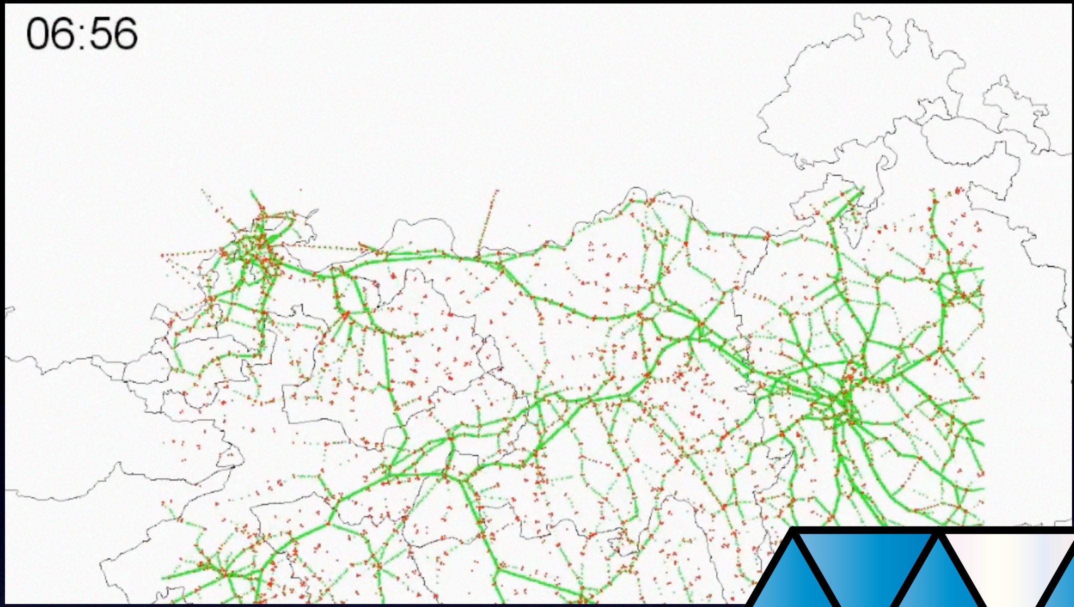
Coupled with the ever expanding amounts of computation and captured data [Hilbert and López 2011], this means that researchers and industry are presented with

This paper is a significant extension of [Stutz et al. 2010]. It contains an updated and more detailed description of the programming model, a larger selection of algorithm adaptations, a distributed version of the underlying framework, and with more extensive evaluations on a graph that is more than 1000 times larger than the ones reported on before.
This work is supported by the Hasler Foundation under grant number 11072.
Author's addresses: P. Stutz, D. Strebel, and A. Bernstein, Dynamic and Distributed Information Systems Group, Department of Informatics, University of Zurich, Binzmühlestrase 14, 8050 Zürich, Switzerland.
Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies show this notice on the first page or initial screen of a display along with the full citation. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, to redistribute to lists, or to use any component of this work in other works requires prior specific permission and/or a fee. Permissions may be requested from Publications Dept., ACM, Inc., 2 Penn Plaza, Suite 701, New York, NY 10121-0701 USA, fax +1 (212) 869-0481, or permissions@acm.org.
© YYYY ACM 1539-9087/YYYY/01-ARTA $10.00
DOI 10.1145/0000000.0000000 http://doi.acm.org/10.1145/0000000.0000000

ACM Transactions on the Web, Vol. V, No. N, Article A, Publication date: January YYYY.

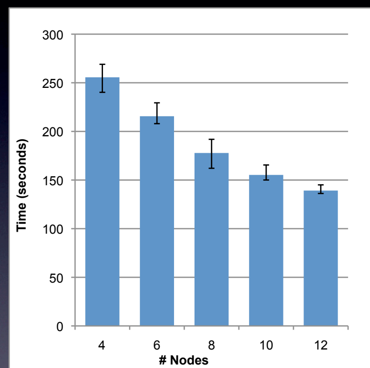


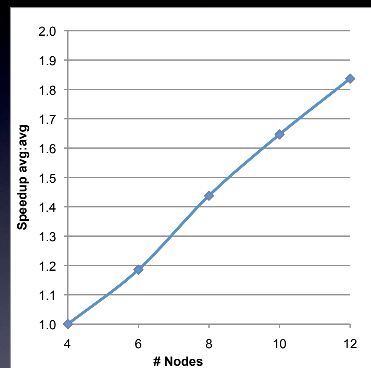




Evaluation

Scalability on a cluster





PageRank on 12 machines (24 cores, 66GB RAM each)
> 1.4 billion vertices, > 6.6 billion edges, 12 machines (24 cores, 66GB RAM each)
Fastest execution time (fully converged): 137s, loading time: 45s