Find us online: graphalytics.ewi.tudelft.nl
https://github.com/tudelft-atlarge/graphalytics/

BENCHMARKING PLATFORMS FOR LARGE-SCALE GRAPH PROCESSING

SC15 Tutorial

The LDBC Approach

Your hosts today

- Alexandru Iosup, Tim Hegeman
  - Delft University of Technology, The Netherlands
- Ana Lucia Varbanescu
  - University of Amsterdam, The Netherlands
- Arnau Prat Perez
  - UPC Barcelona, Spain
- Mihai Capota
  - Intel Labs, USA
- Thomas Manhardt
  - Oracle Labs, USA

+ team members:
Wing Lung Ngai,
M. Biczak (TU Delft)
Josep Larriba-Pey (UPC)
Peter Boncz (CWI)
Alex Averbuch (Neo4j)

Before we proceed

- Download and Install VirtualBox from

  http://virtualbox.com

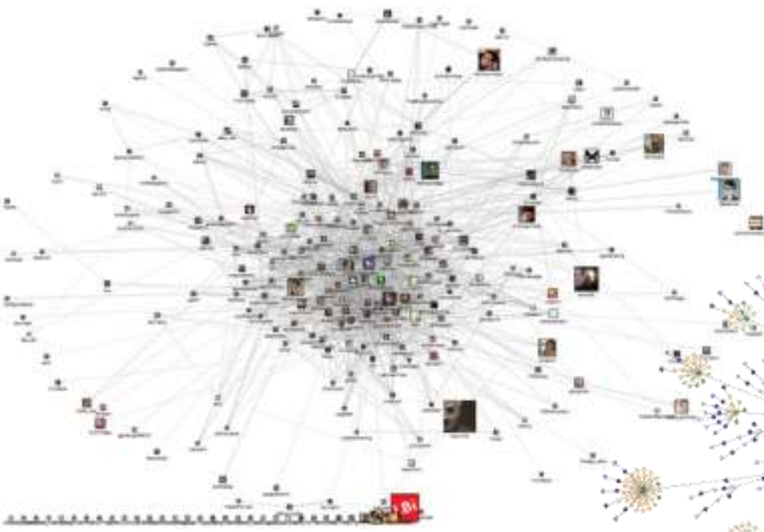
  During the break before the hands-on session (or now)

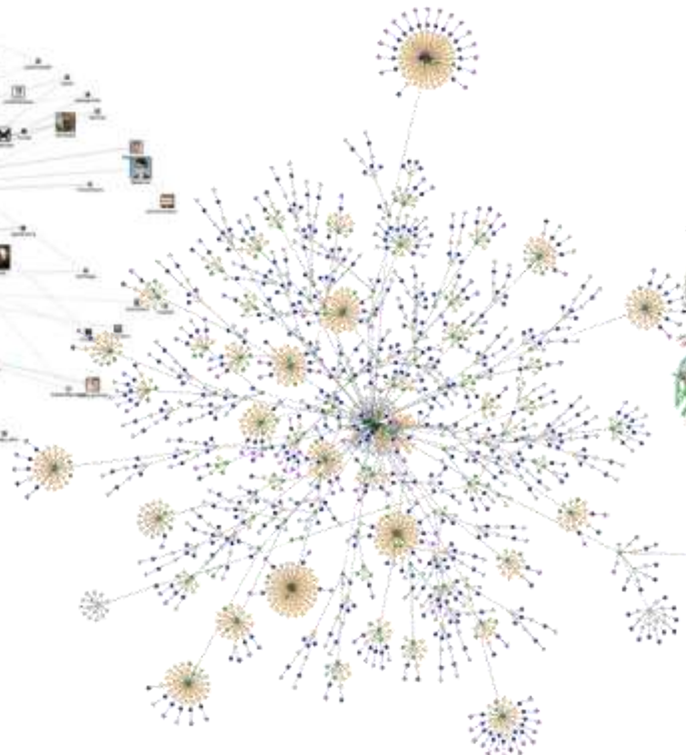
Linked data

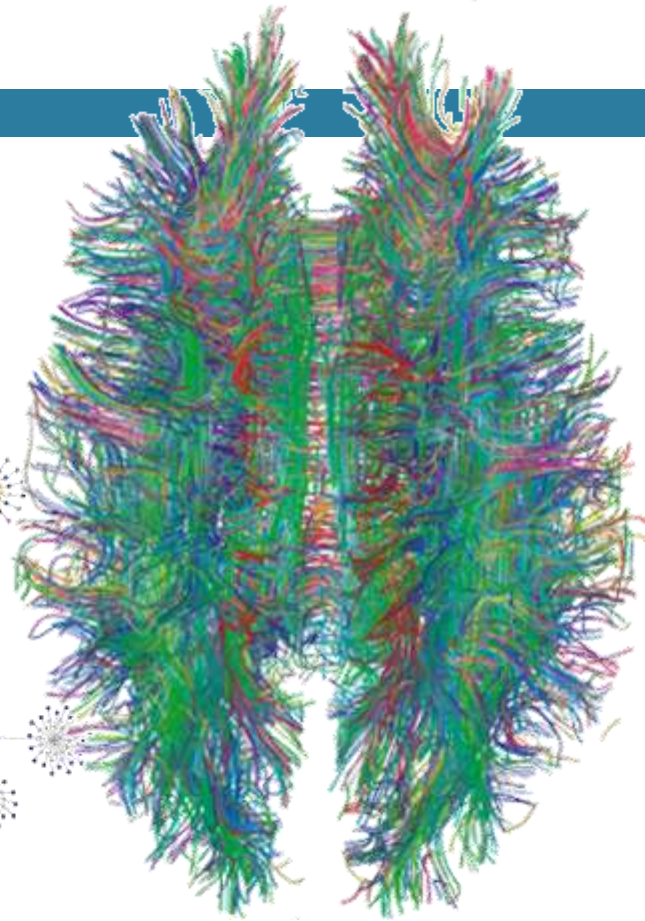
Size matters

The need for benchmarking

The data deluge: large-scale graphs







Social network

~1 billion vertices
~100 billion connections

Web graph

~50 billion pages
~1 trillion hyperlinks

Brain network

~100 billion neurons
~100 trillion connections

Source: Smith, CHI'10; Blog webpage; Gigandet et al., PLoS ONE 3(12)]

Graphs Are at the Core of Our Society: The LinkedIn Example

400 million

Q3 2015

A very good resource for matchmaking workforce and prospective employers

Vital for your company's life, as your Head of HR would tell you

Vital for the prospective employees

Tens of "specialized LinkedIns": medical, mil, edu, gov, ...

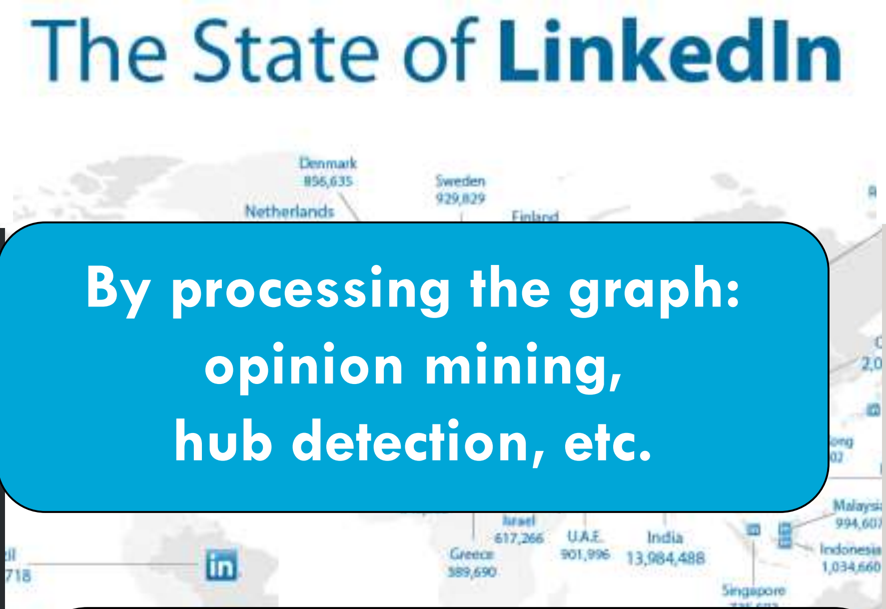
2

ay 2010

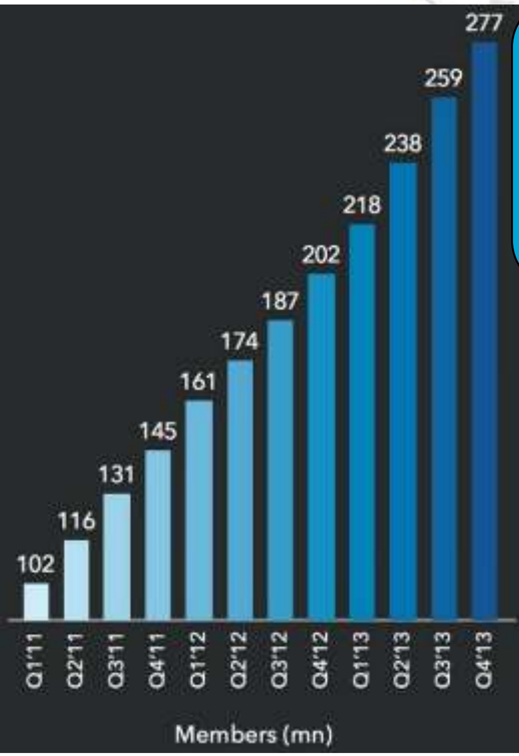
Sources: Vinc
via Christopher Penn, http://www.shiftcomm.com/2014/02/state-linkedin-social-media-dark-horse/

LinkedIn's Service/Ops Analytics

3-4 new users every second

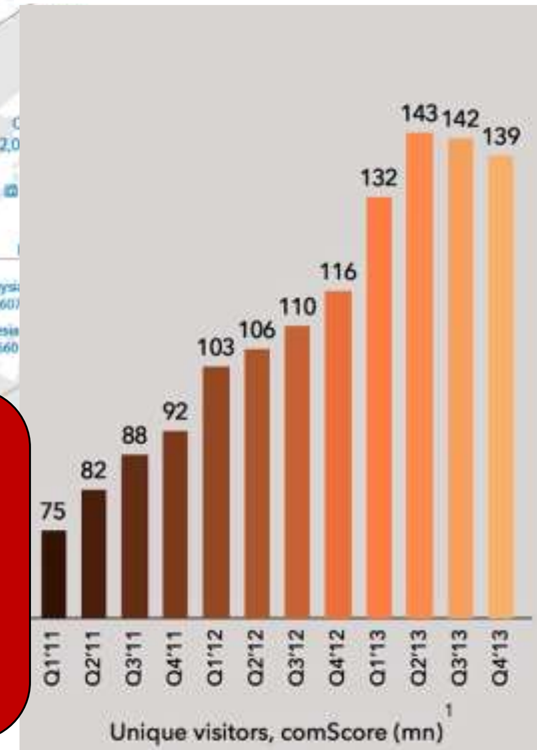


but fewer visitors (and page views)



By processing the graph: opinion mining, hub detection, etc.

100+ million questions of customer retention, of (lost) customer influence, of ...



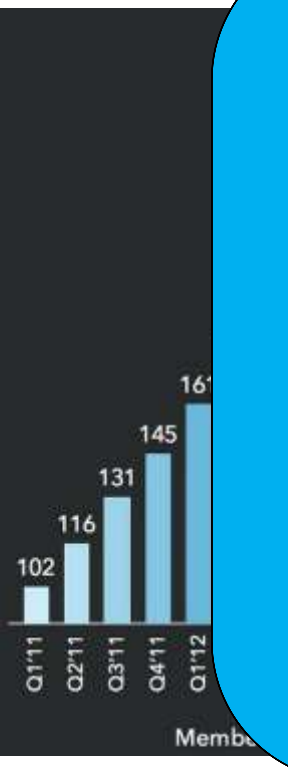
Sources: Vincenzo Cosenza, The State of LinkedIn, http://vincos.it/the-state-of-linkedin/
via Christopher Penn, http://www.shiftcomm.com/2014/02/state-linkedin-social-media-dark-horse/

Why Analytics?

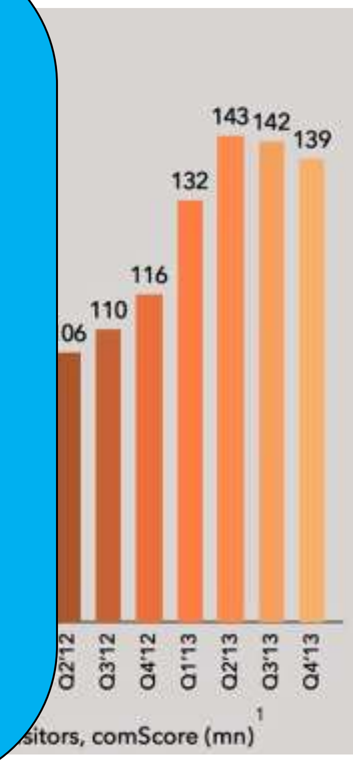
The State of LinkedIn

3-4 new users every second

but fewer visitors (and page views)



Periodic and/or continuous analytics at full scale

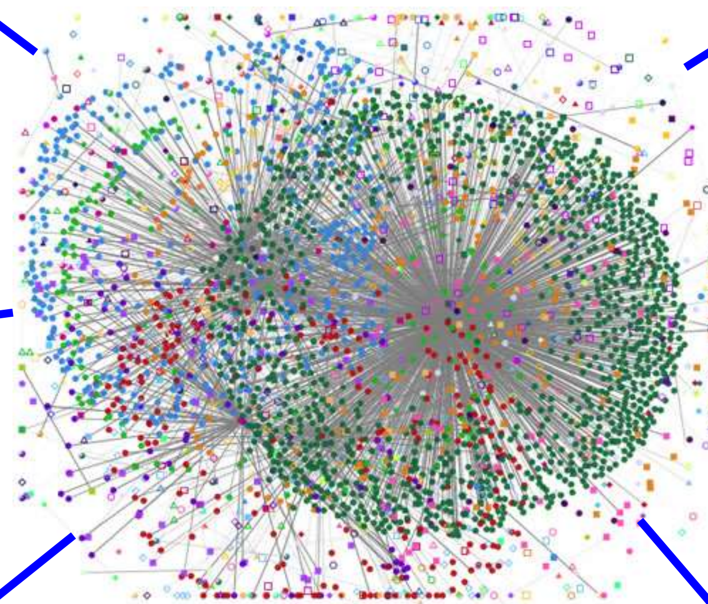


Sources: Vincenzo Cosenza, The State of LinkedIn, http://vincos.it/the-state-of-linkedin/
via Christopher Penn, http://www.shiftcomm.com/2014/02/state-linkedin-social-media-dark-horse/

The data deluge: graphs everywhere!

400M users

??? edges



270M MAU
200+ avg followers

>54B edges

1.2B MAU 0.8B DAU
200+ avg followers

>240B edges

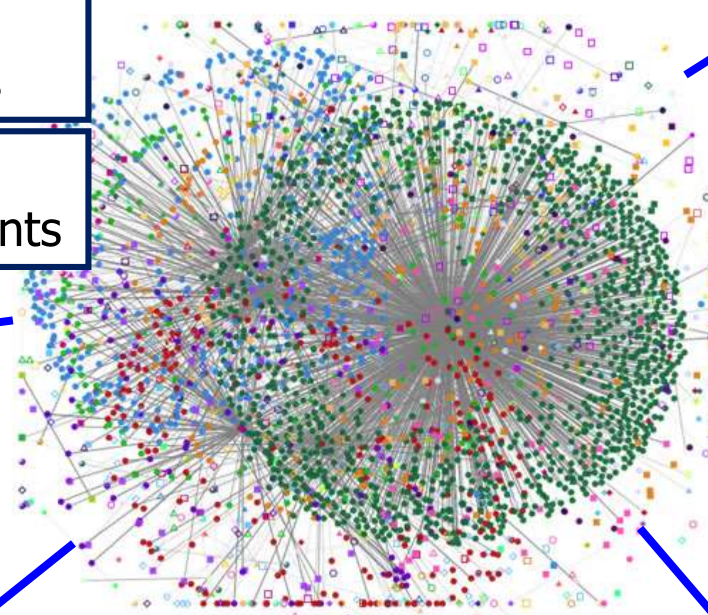
The data deluge: graphs everywhere!

Oracle 1.2M followers, 132k employees

company/day:
40-60 posts, 500-700 comments

270M MAU
200+ avg followers

>54B edges



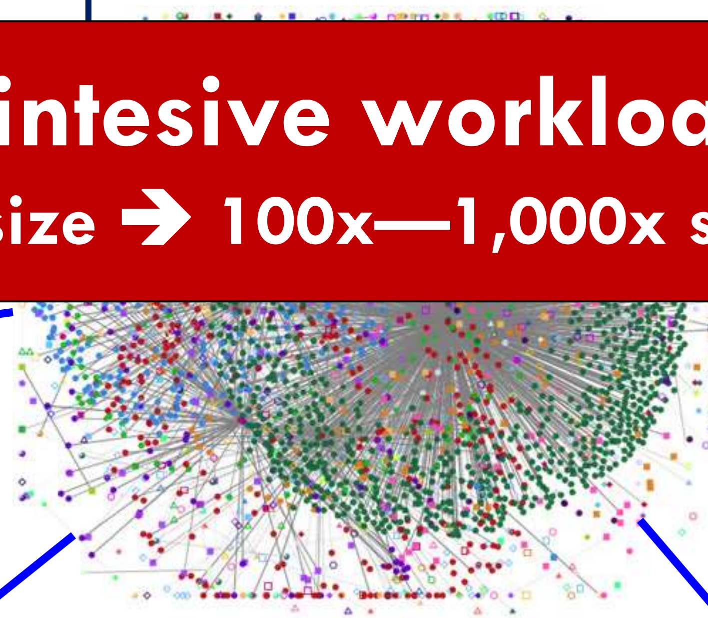
1.2B MAU 0.8B DAU
200+ avg followers

>240B edges

The data deluge: graphs everywhere!

Oracle 1.2M followers

270M MAU



Data-intesive workload

10x graph size → 100x—1,000x slower

1.2B MAU 0.8B DAU
200+ avg followers

>240B edges

The data deluge: graphs everywhere!

LinkedIn

Oracle 1.2M followers

270M MAU

Data-intesive workload

10x graph size → 100x—1,000x slower

Compute-intesive workload

more complex analysis → ?x slower

>240B edges

friendster

XFIRE

The data deluge: graphs everywhere!

Oracle 1.2M followers

270M MAU

>240B edges

Data-intesive workload

10x graph size → 100x—1,000x slower

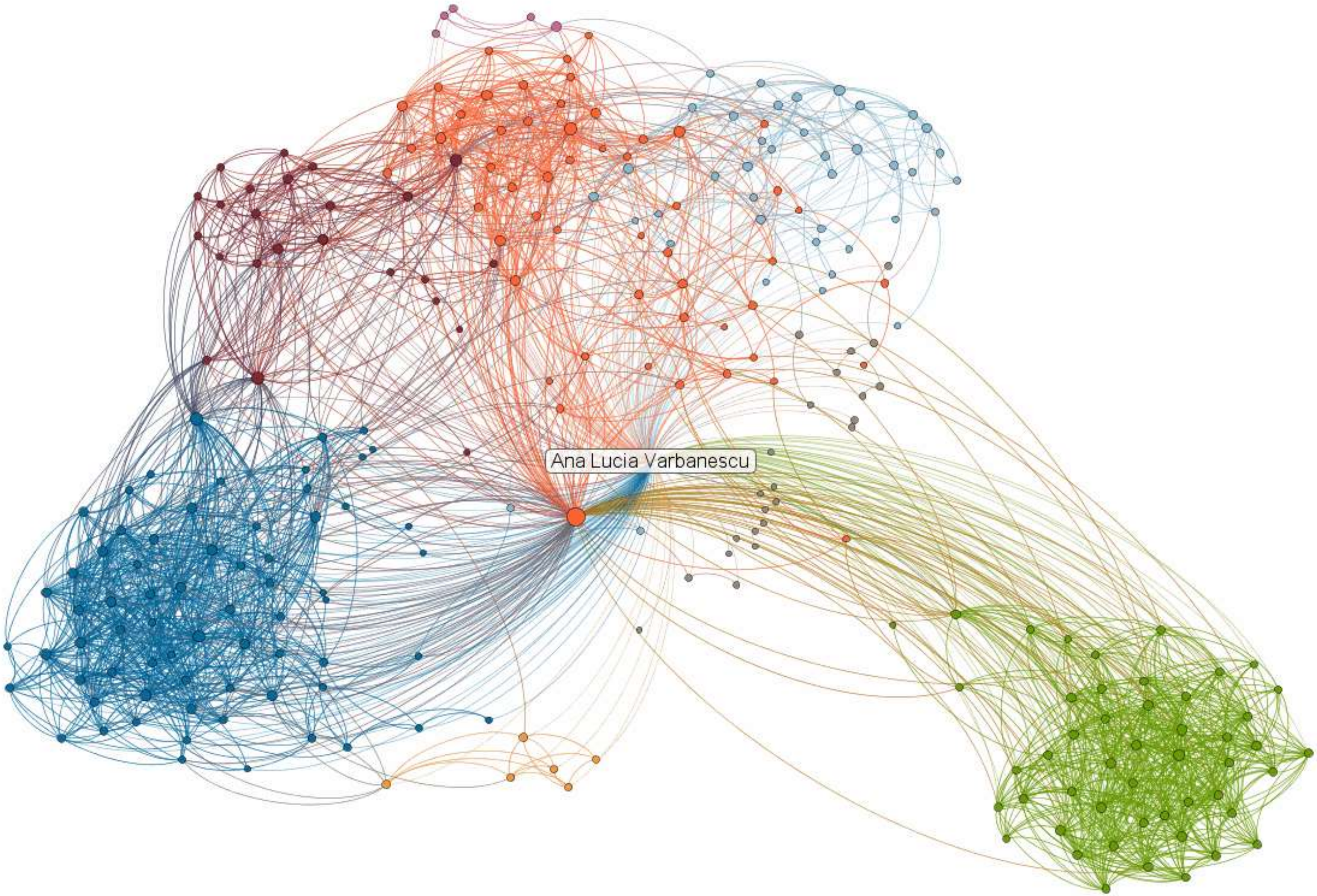
Compute-intesive workload

more complex analysis → ?x slower

Dataset-dependent workload

unfriendly graphs → ??x slower

LinkedIn Maps **Ana Lucia Varbanescu's Professional Network**
as of November 28, 2013



Your network is so large...

Sorry, but your network is too large to be computed, we are working to increase the limit, stay tuned!

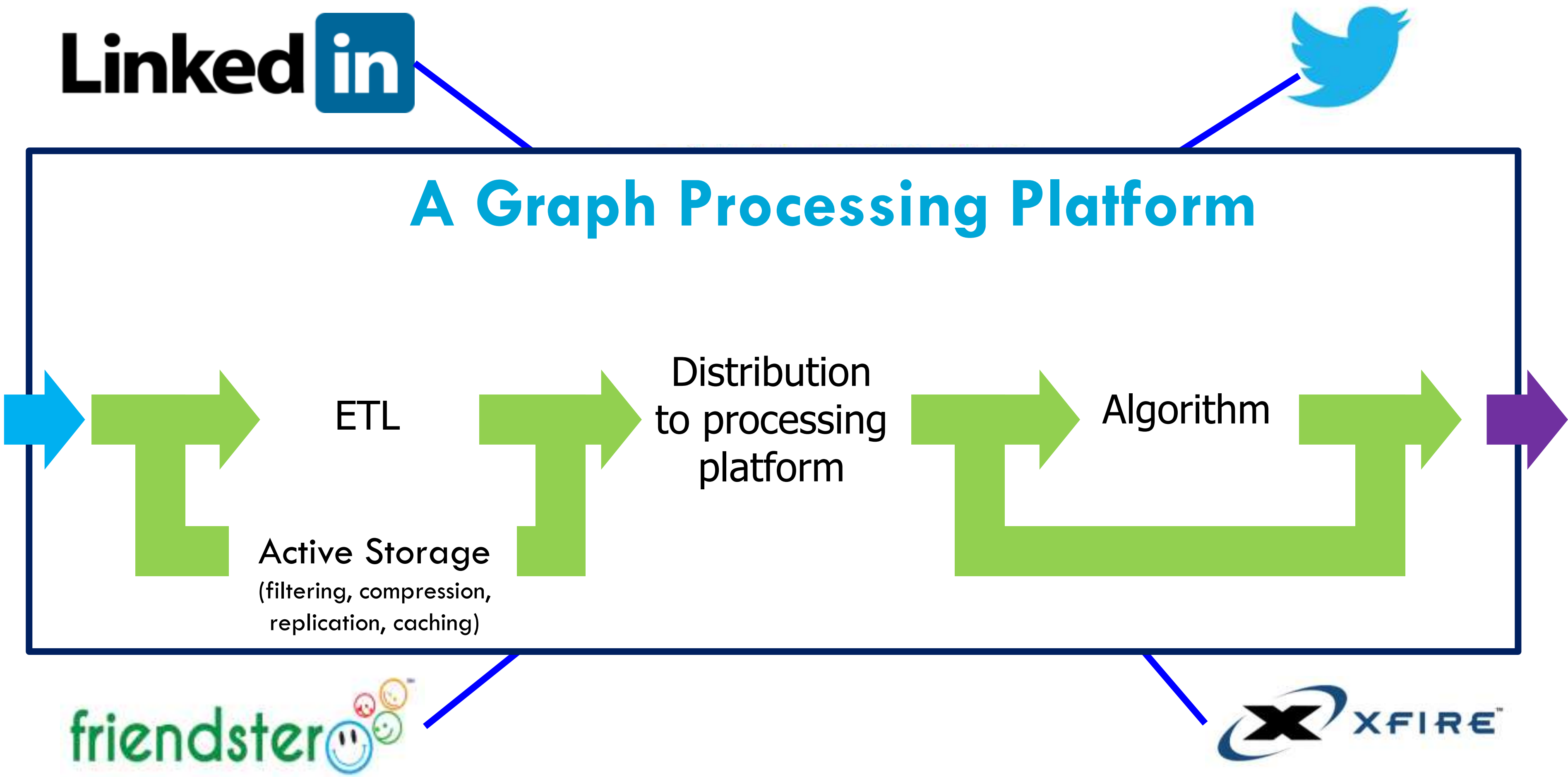
The “sorry, but…” moment

The “sorry, but…” moment

Supporting multiple users

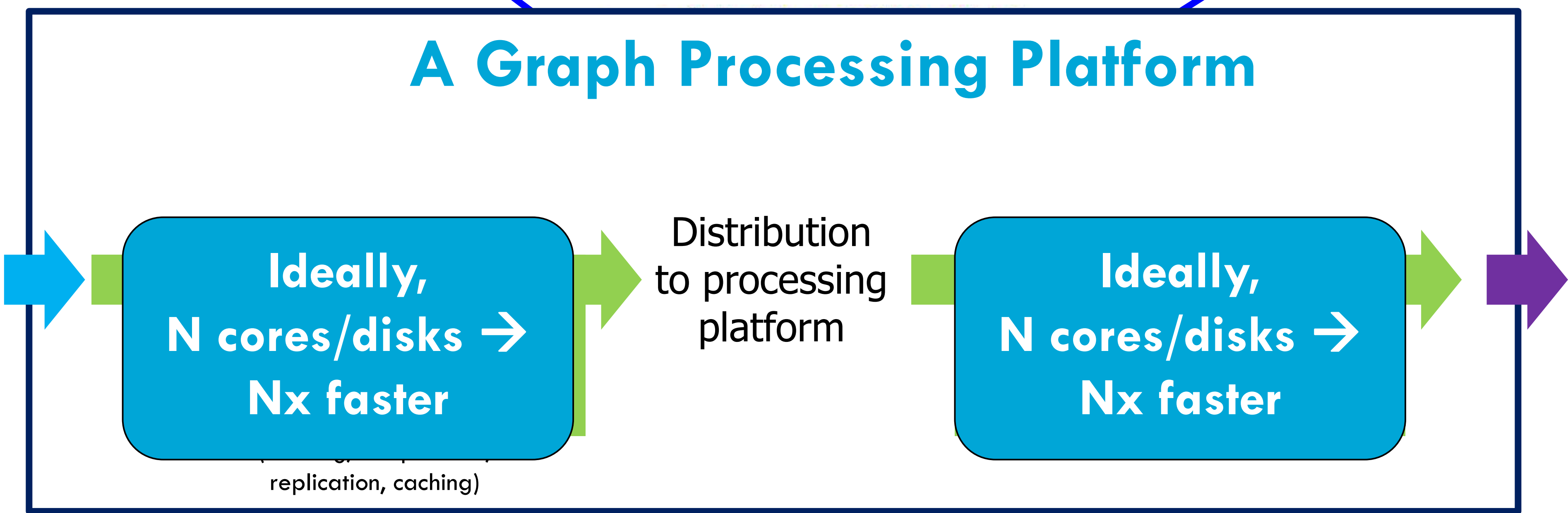
10x number of users → ????x slower

Graph Processing @large



Interactive processing not considered in this presentation.
Streaming not considered in this presentation.

Graph Processing @large



Interactive processing not considered in this presentation.

Streaming not considered in this presentation.

Graph-Processing Platforms

Platform= the combined hardware, software, and programming system that is being used to complete a graph processing task

Which to choose?

What to tune?

In this tutorial…

- Graphalytics = benchmarking graph analytics
- Analytics = graph processing @large
- Platform = hardware and/or software and/or programming model we can tune and change
- (Graph) Processing system = computing system that includes one or more platforms (for graph processing)
- Choke point = system component or workload characteristic, or combinations thereof, that lead to poor system performance

Agenda

- Introduction to Linked Data
- LDBC Approach
- Graphalytics
  - Systems and models
  - Methodology for performance evaluation of graph-processing platforms
  - Graphalytics architecture
- The hour of benchmarking
  - Hands-on Graphalytics
    - Results analysis & lessons learned
  - Fine-grained in-depth analysis with Granula
- Summary & Panel/open discussion

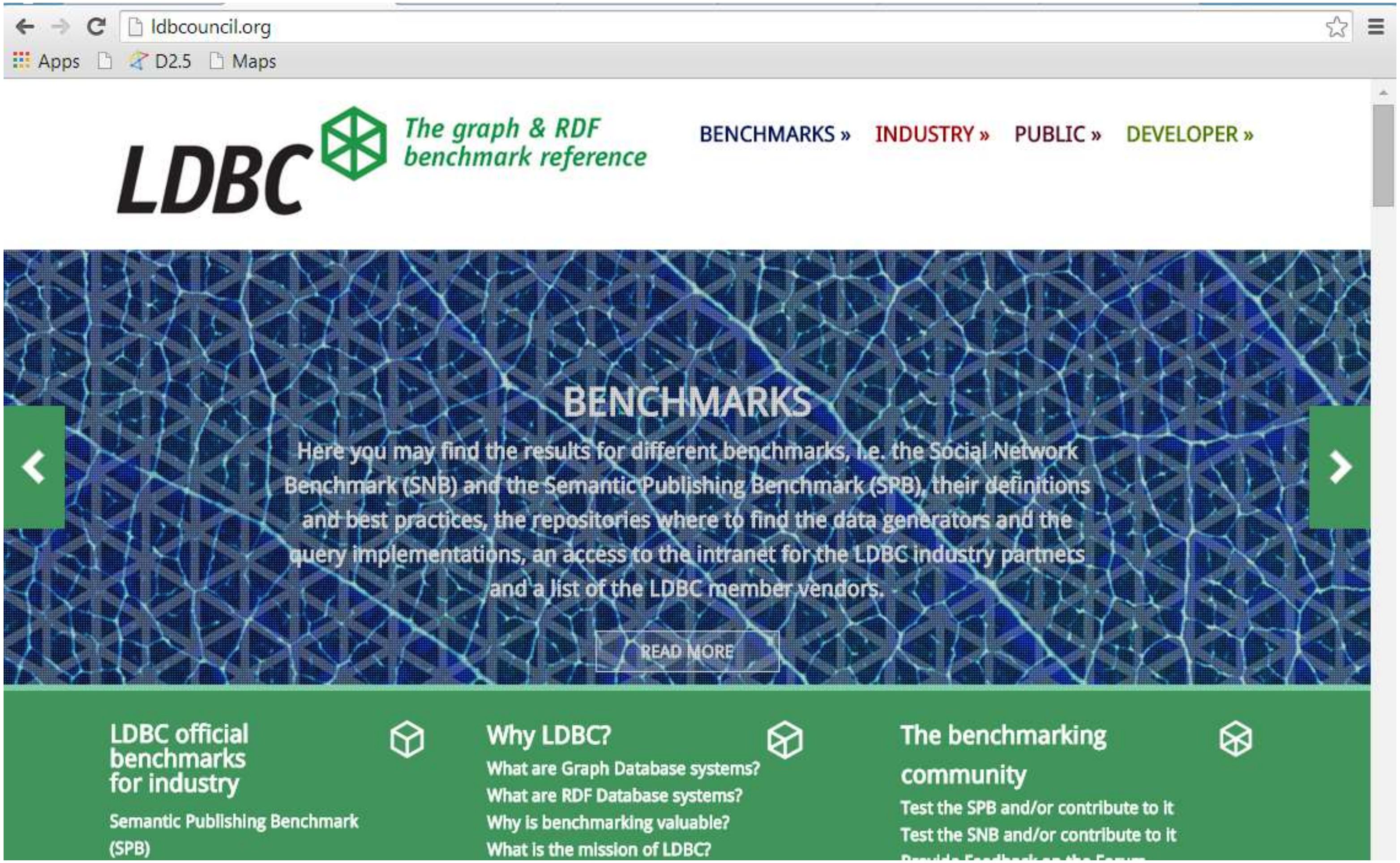
About the Linked Data Benchmarking Council (LDBC)

Benchmarking graph-processing activities

LDBC

The graph & RDF benchmark reference

ldbcouncil.org



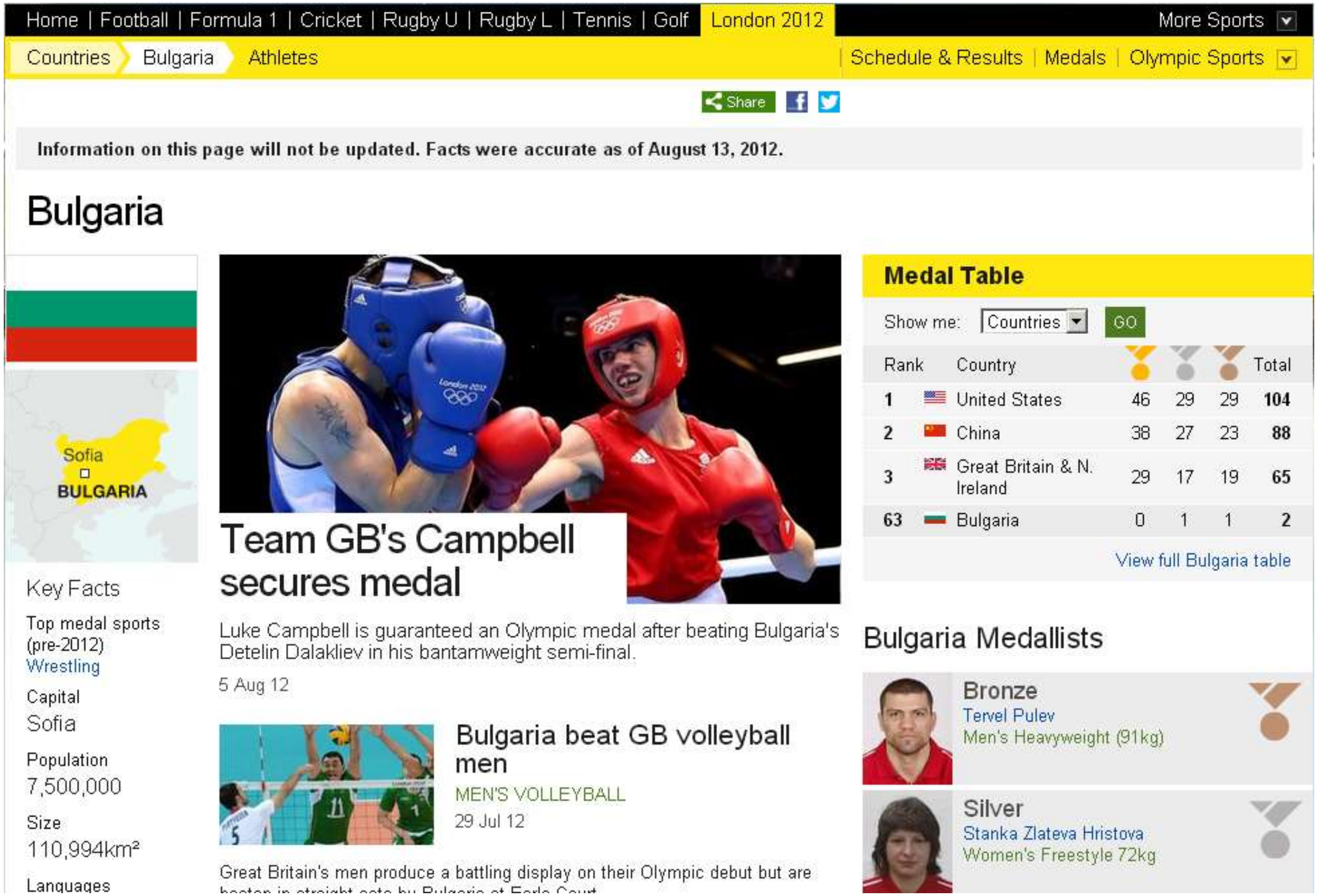
Audience

- For **developers** facing graph processing tasks
  - recognizable scenario to compare merits of different products and technologies
- For **vendors** of graph processing technology
  - checklist of features and performance characteristics
- For **researchers**, both industrial and academic
  - challenges in multiple choke-point areas such as graph query optimization and (distributed) graph analysis

LDBC Task Forces

- Semantic Publishing Benchmark Task Force
  - Develops industry-grade RDF benchmark
- Social Network Benchmark Task Force
  - Develops benchmark for graph data management systems
  - Broad coverage: three workloads
- Graph Analytics Task Force → Graphalytics
  - Spin-off from the SNB task force
- Graph Query Language Task Force
  - Not strictly about benchmarking
  - Studies features of graph database query languages

Semantic Publishing Benchmark (SPB)



SPB scope

- The scenario involves a media/ publisher organization that maintains semantic metadata about its Journalistic assets (articles, photos, videos, papers, books, etc), also called Creative Works
- The Semantic Publishing Benchmark simulates:
  - Consumption of RDF metadata (Creative Works)
  - Updates of RDF metadata, related to Annotations
- Aims to be an industrially mature RDF database benchmark (SPARQL1.1, some reasoning, text and GIS queries, backup&restore)

Social Network Benchmark (SNB)

- Intuitive: everybody knows what a SN is
  - Facebook, Twitter, LinkedIn, …
- SNs can be easily represented as a graph
  - Entities are the nodes (Person, Group, Tag, Post, ...)
  - Relationships are the edges (Friend, Likes, Follows, …)
- Different scales: from small to very large SNs
  - Up to billions of nodes and edges
- Multiple query needs:
  - interactive, analytical, transactional
- Multiple types of uses:
  - marketing, recommendation, social interactions, fraud detection, ...

LDBC Social Network Benchmark (SNB)

Synthetic graph generation

LDBC

The graph & RDF benchmark reference

Why a synthetic graph generator?

- Real graphs are sometimes difficult to obtain
  - Not practical to distribute TeraBytes of data
  - Privacy concerns
- Real data do not always have the desired characteristics
  - Many dimensions to be tested (size, distributions, structural characteristics, etc.) as they can affect the performance of the tested systems
  - Difficult to obtain real data for all the desired dimension combinations

Generator's features (wish list)

- Scalable
  - From GigaBytes to TeraBytes of data
- Realistic
  - Distributions: attributes, degrees, etc.
  - Correlations: attributes, edges, etc.
  - Structural characteristics: clustering coefficient, largest connected component, diameter, etc.
- Flexible
  - Allow choosing the characteristics of the generated data
  - Support different output formats

LDBC DATAGEN

- DATAGEN is a fork of S3G2[1]
- Developed during LDBC European Project as the data generator for the LDBC Social Network Benchmark Workloads
- Available at: https://github.com/ldbc/ldbc_snb_datagen

[1] Pham, Minh-Duc, Peter Boncz, and Orri Erling. "S3g2: A scalable structure-correlated social graph generator." Selected Topics in Performance Evaluation and Benchmarking. Springer Berlin Heidelberg, 2013. 156-172.

LDBC DATAGEN

- Generates a Social Network graph
  - Uses dictionaries extracted from Dbpedia to populate the dataset with realistic attributes
    - e.g. Person names, countries, companies, tags (interests)
  - Correlated attributes
    - e.g. Person names with countries, correlations between tags, etc.
  - Realistic distributions
    - Facebook-like degree distribution, attribute distributions etc.
  - Event-based user activity generation
    - Mimick spikes of activity around specific events

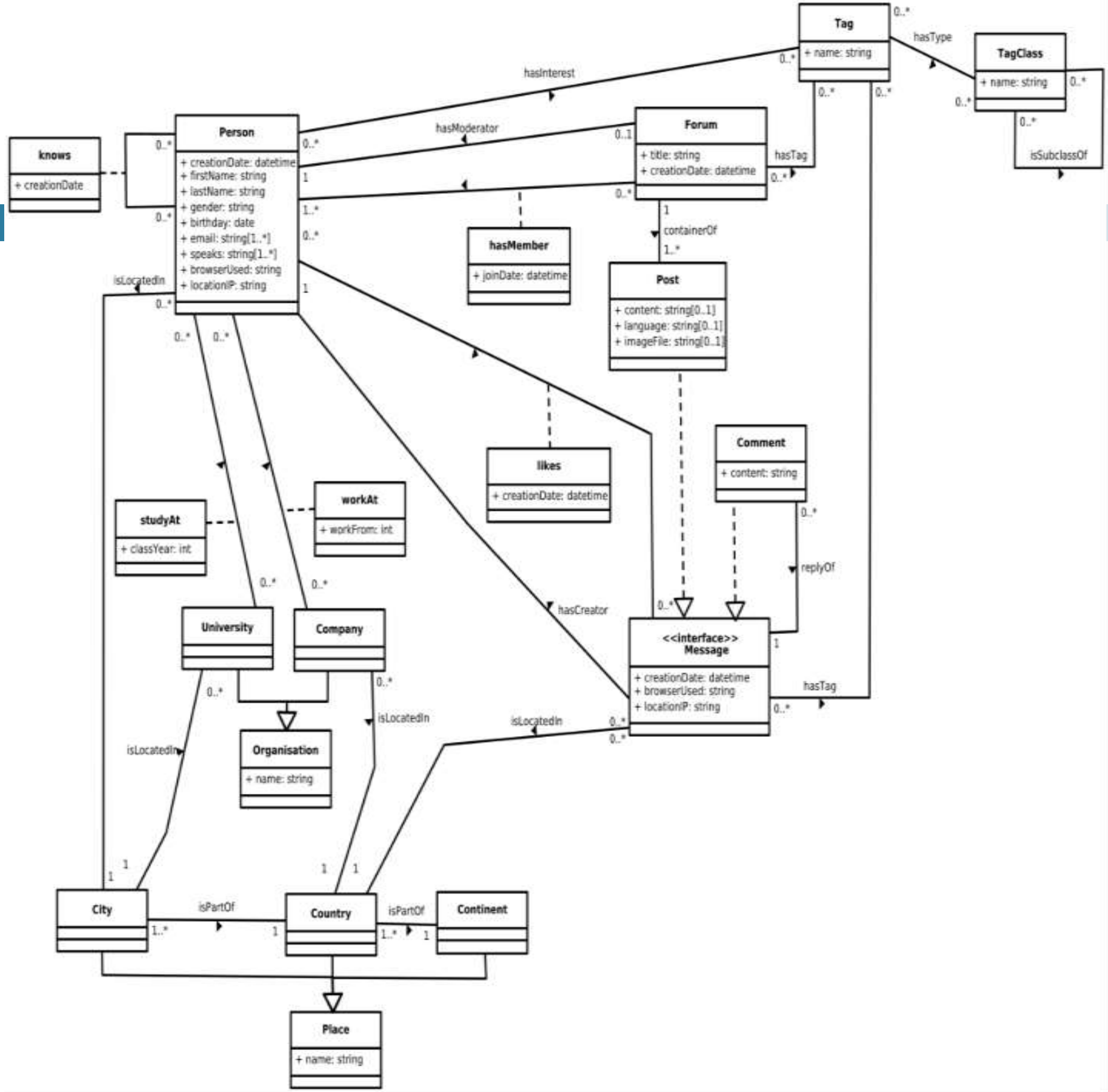
LDBC DATAGEN

- Built on top of Hadoop
  - Able to generate Terabytes of data with a small commodity cluster
  - Billion edge graphs in few hours

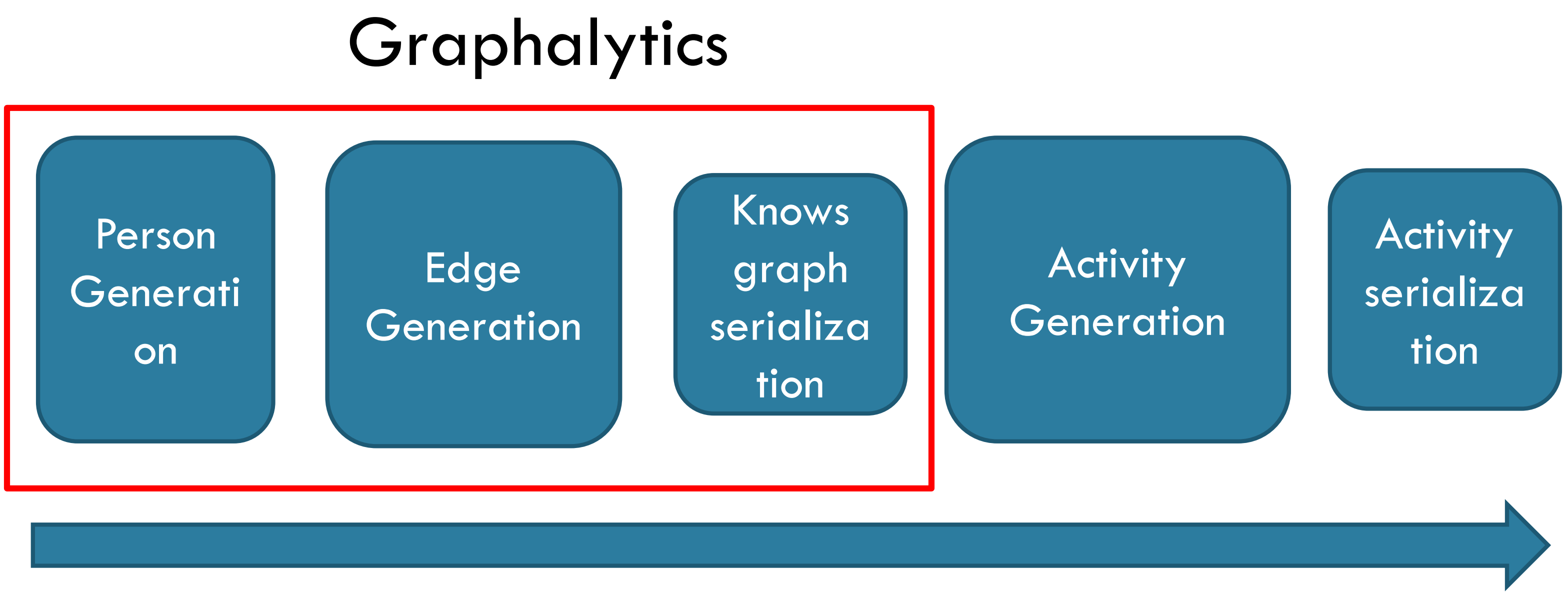
- Deterministic

LDBC DATAGEN

- Continuously evolving towards a more flexible data generator
  - Support for different degree distributions: Zipf, MOEZipf, Geometric, Discrete Weibull, etc.
  - Able to tune structural characteristics of the network (e.g. clustering coefficient, assortativity, etc.)
  - Custom data serializers
  - A more flexible schema definition


Tag
+ name: string
TagClass
+ name: string
hasType
hasInterest
isSubclassOf
hasModerator
Forum
+ title: string
+ creationDate: datetime
hasTag
knows
+ creationDate
Person
+ creationDate: datetime
+ firstName: string
+ lastName: string
+ gender: string
+ birthday: date
+ email: string[1..*]
+ speaks: string[1..*]
+ browserUsed: string
+ locationIP: string
hasMember
+ joinDate: datetime
containerOf
Post
+ content: string[0..1]
+ language: string[0..1]
+ imageFile: string[0..1]
isLocatedIn
likes
+ creationDate: datetime
Comment
+ content: string
studyAt
+ classYear: int
workAt
+ workFrom: int
replyOf
hasCreator
University
Company
<<interface>>
Message
+ creationDate: datetime
+ browserUsed: string
+ locationIP: string
Organisation
+ name: string
City
Country
Continent
isPartOf
Place
+ name: string

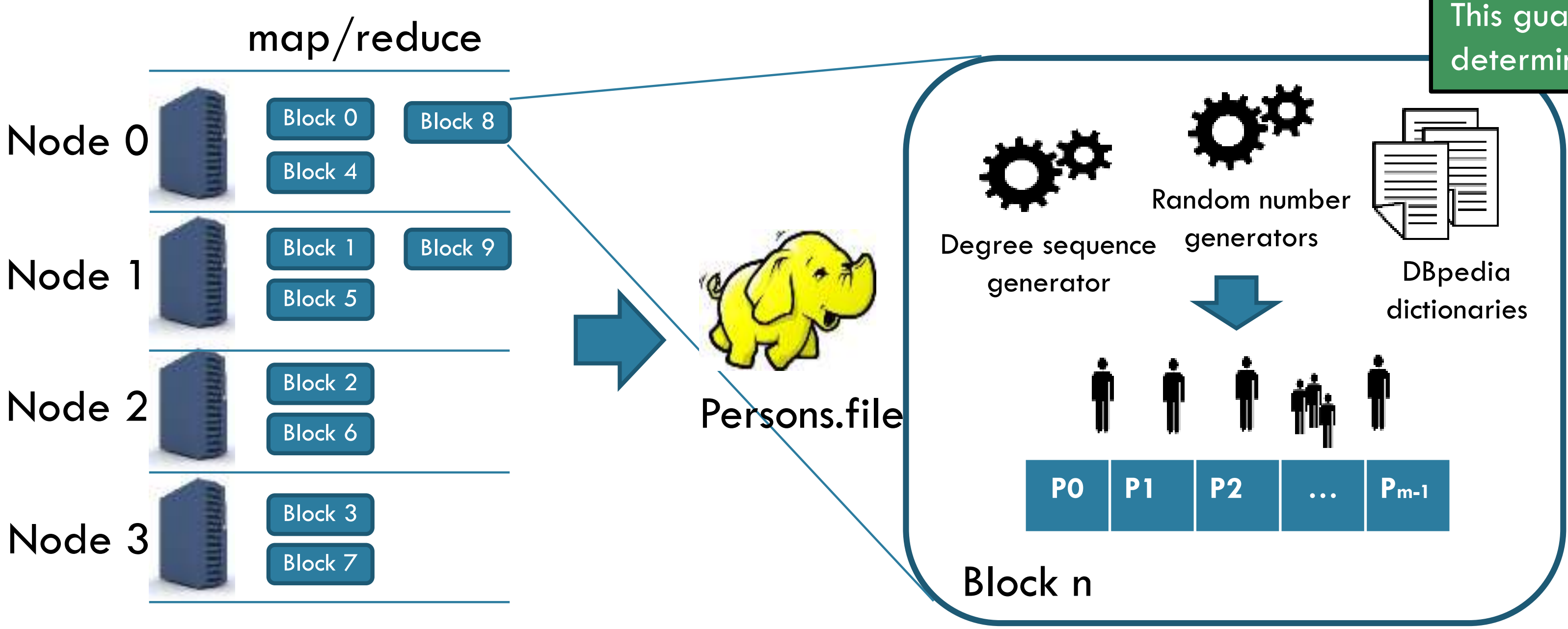
Generation Process



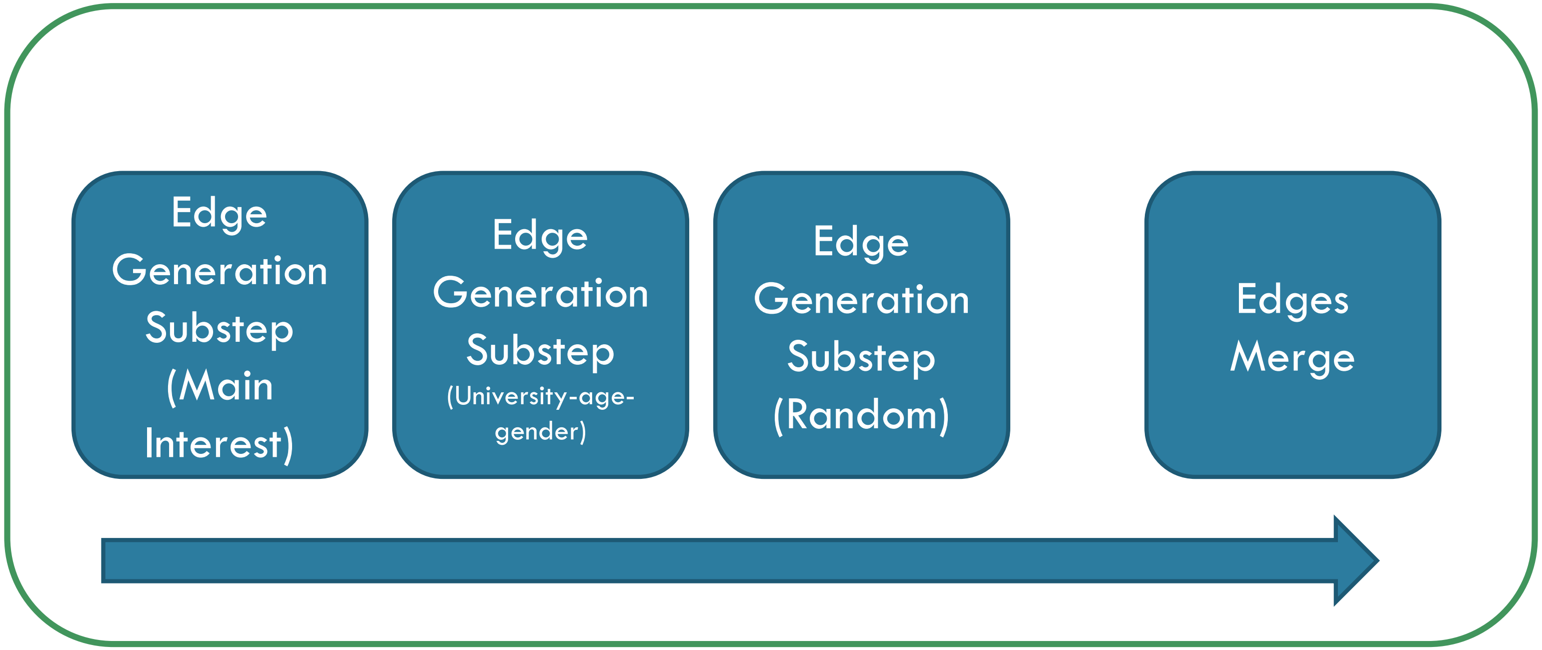
Person Generation

- A 4-machine cluster
- 100,000 Person network
- Block size m= 10,000 => 10 blocks in total

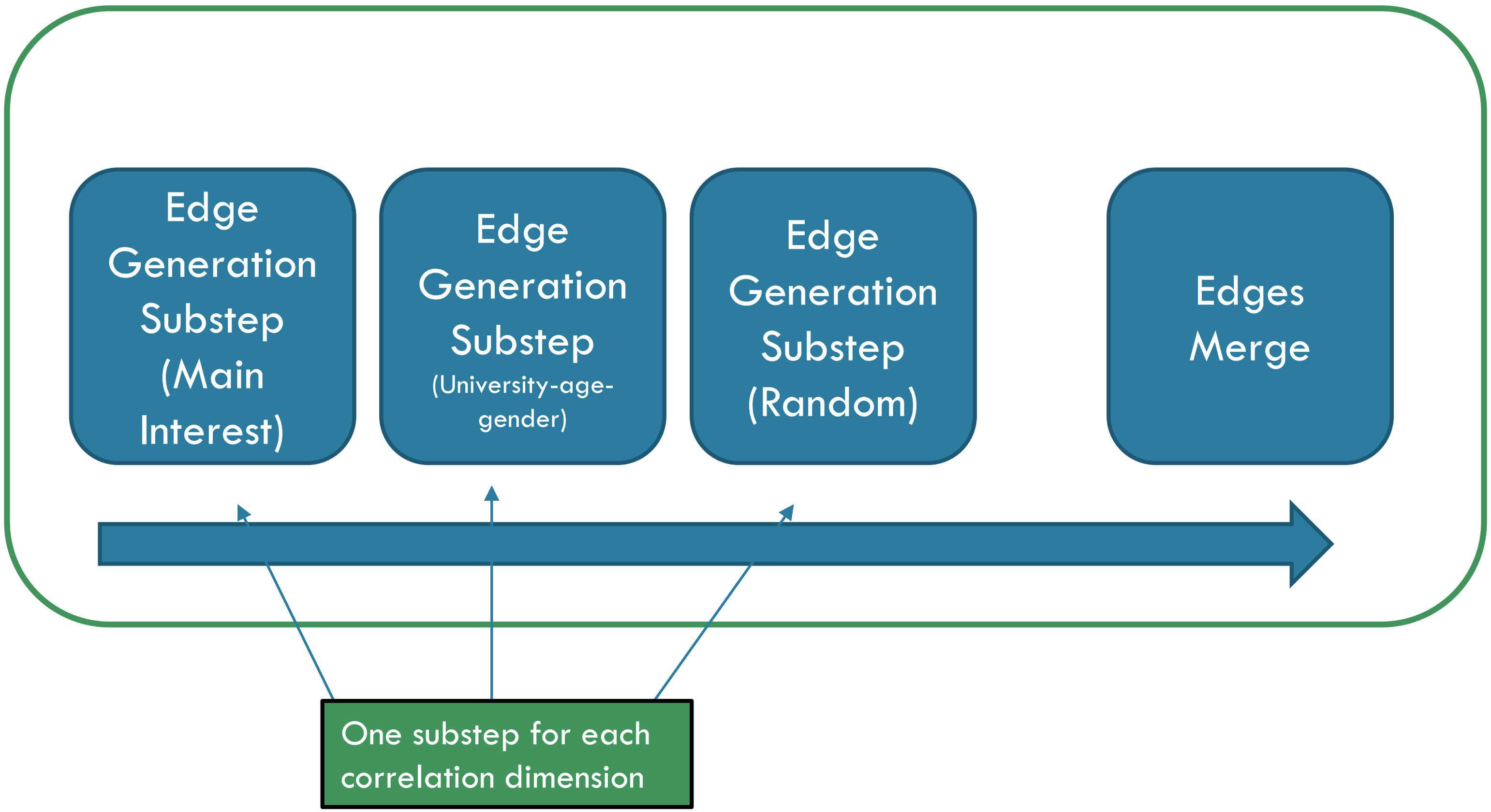
Each block has its own independent state, which depends only on the block id. This guarantees determinism.



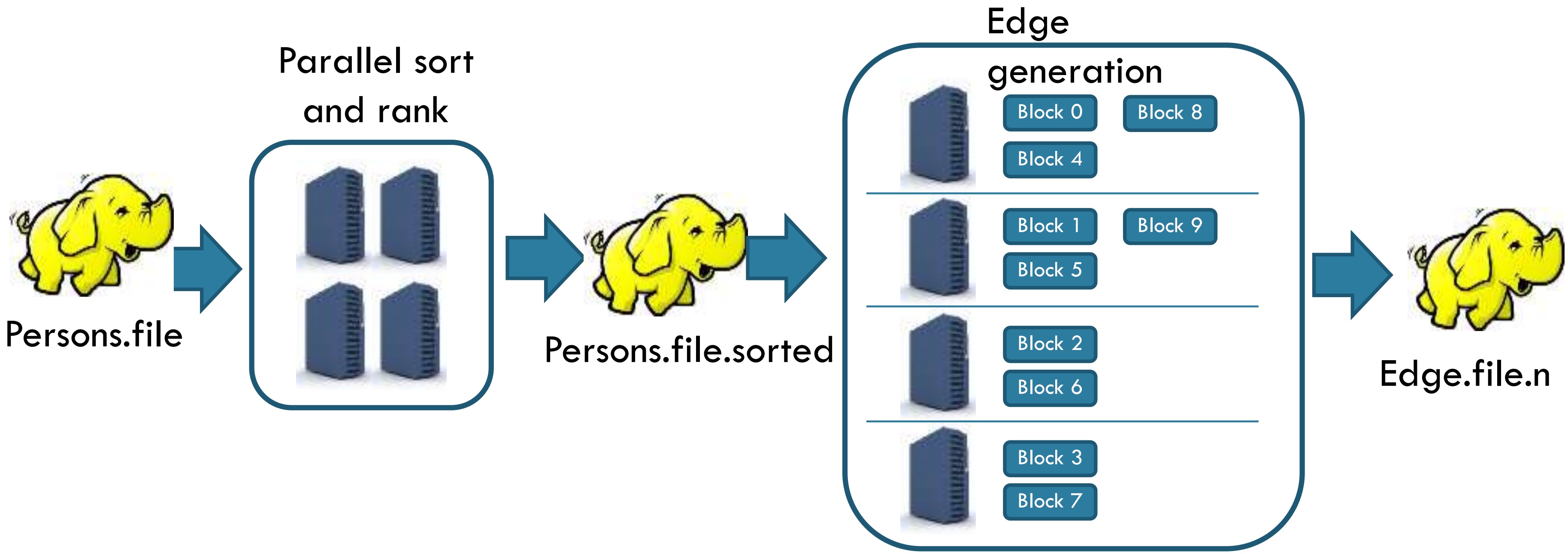
Edge Generation



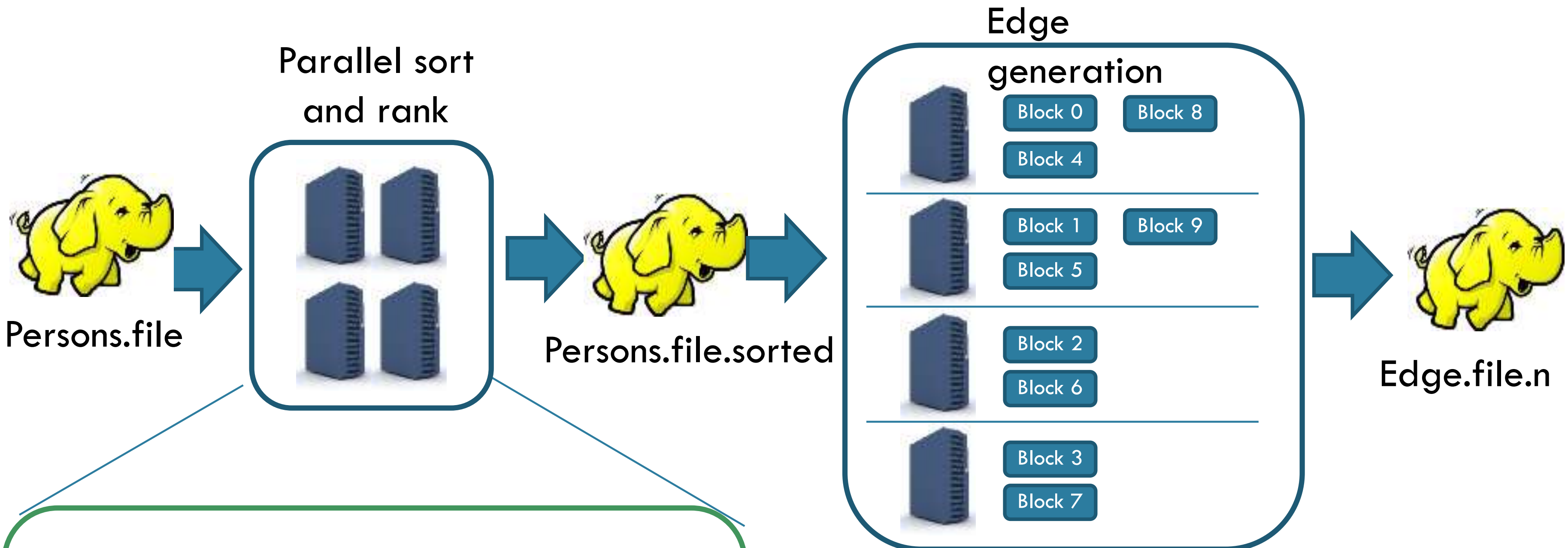
Edge Generation



Edge Generation Substep

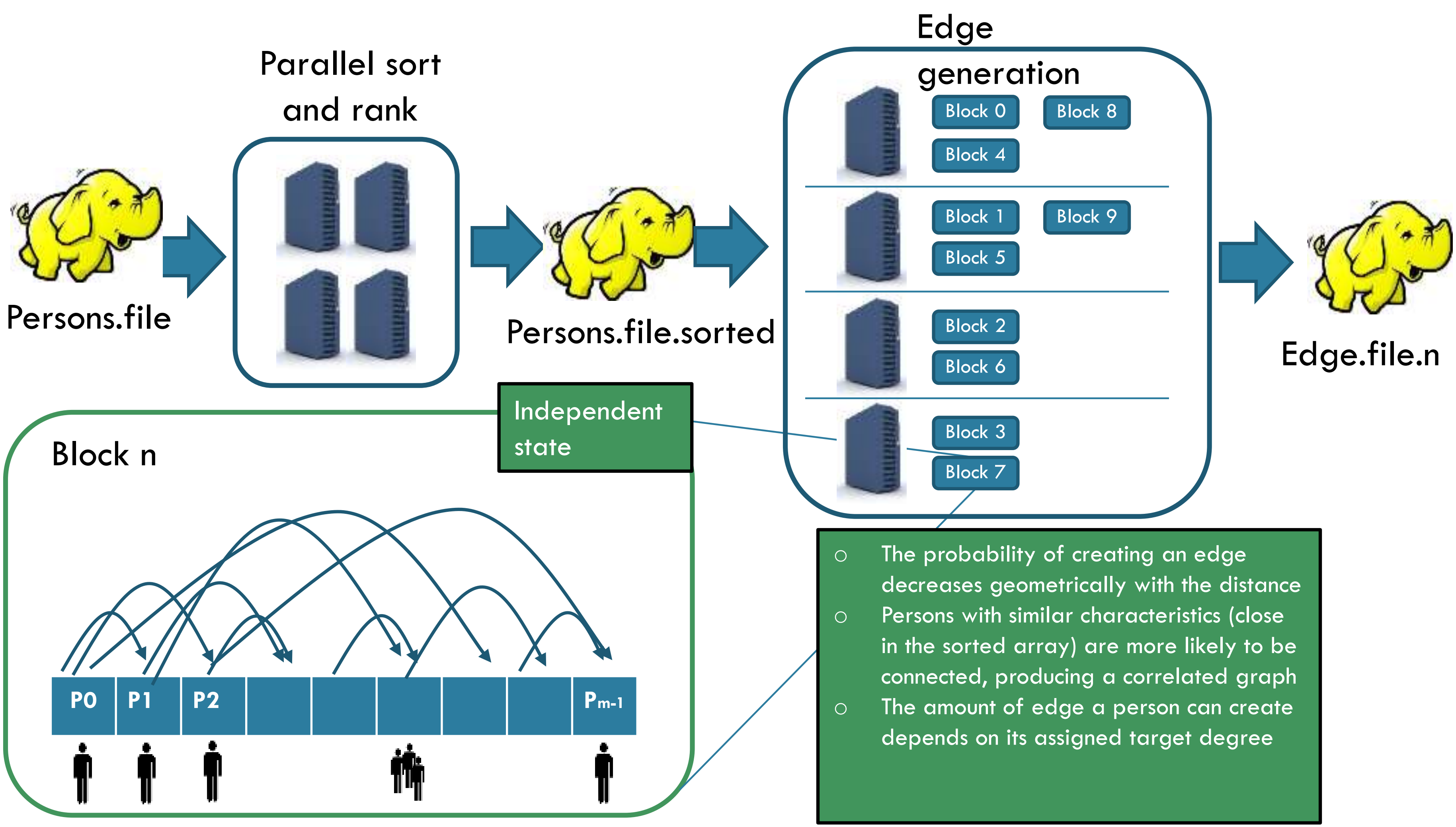


Edge Generation Substep

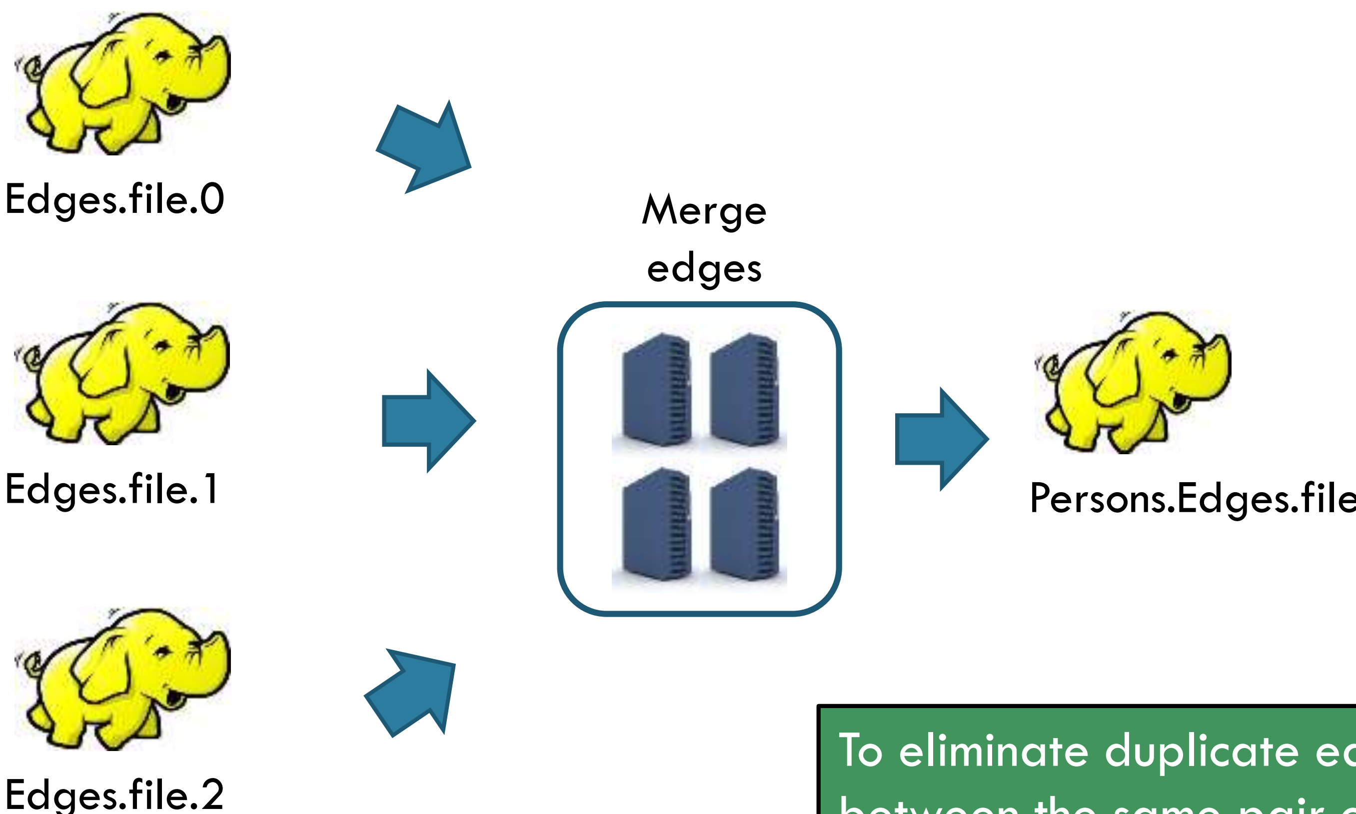


- Sort by correlation dimension:
  - e.g. Main interest, University-age, random
- Set Person keys as their position in the sorted array (between 0 and N-1)

Edge Generation Substep



Edge Merge



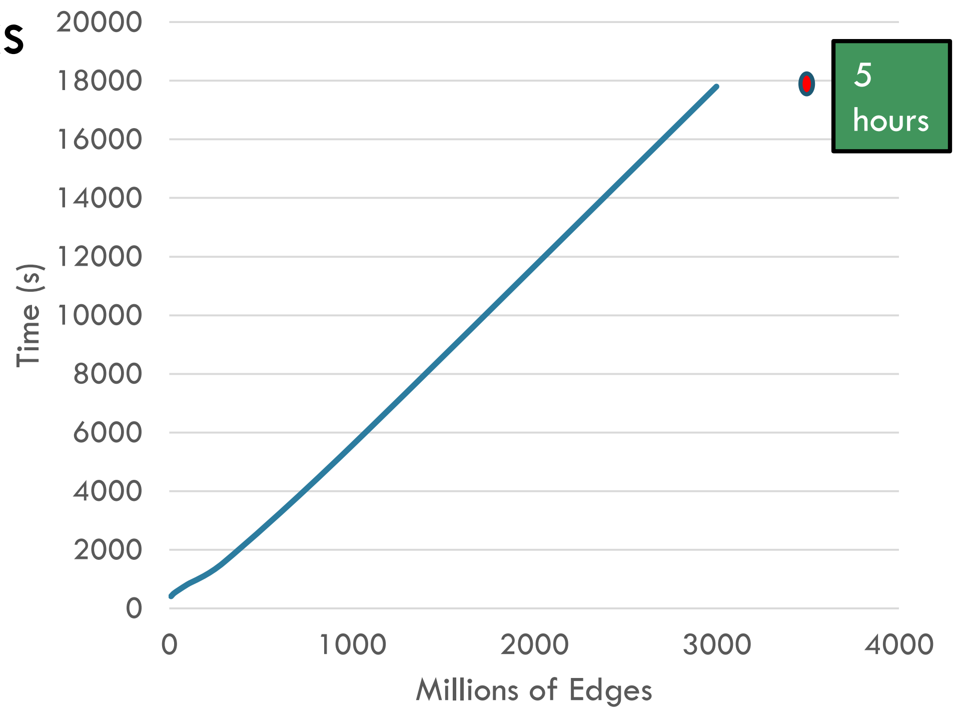
To eliminate duplicate edges between the same pair of Persons

Knows graph serialization

- Finally, *Persons.Edges.file* is read and serialized into HDFS using a configurable serializer.
- Serializers implement ldbc.snb.datagen.serializer* interfaces
  - To write to HDFS
  - To directly bulk load data into the Database System
- Provided CSV serializers
  - Can output compressed files

Performance snapshot

- Cluster with four nodes:
  - Intel Xeon E5530 @ 2.4 Ghz (4 cores, Year 2010)
  - 32Gb of RAM
  - 7200 rpm spinning disks
  - 1 master, 3 slaves
  - 12 reducers in total



Scale Factors

- Provided Scale Factors for LDBC SNB Interactive and Graphalytics
- Scale factors are just configuration presets of DATAGEN

Scale Factor	#Persons	#Edges
Graphalytics.10	235,000	10,000,000
Graphalytics.30	592,500	30,000,000
Graphalytics.100	1,167,000	100,000,000
Graphalytics.300	4,350,000	300,000,000
Graphalytics.1000	12,750,000	1,000,000,000
Graphalytics.3000	32,500,000	3,000,000,000

Final remarks

- The generated Graph is structurally correlated
  - Persons tend to be connected with similar people
- Characteristics typical from real social networks
  - 6-degrees of separation, large connected component, moderately large clustering coefficient, skewed distribution
- Very good scalability: current experiments show linear scalability
- Rapidly evolving to support new features such as tuning structural properties of the graph, or being able to change the generated schema

Questions?

Agenda

- Introduction to Linked Data
- LDBC Approach
- Graphalytics
  - Systems and models
  - Methodology for performance evaluation of graph-processing platforms
  - Graphalytics architecture
- The hour of benchmarking
  - Hands-on Graphalytics
    - Results analysis & lessons learned
  - Fine-grained in-depth analysis with Granula
- Summary & Panel/open discussion

Systems and models

Graph processing @ scale

- The characteristics of graph processing
  - Poor locality
  - Unstructured computation
  - Variable parallelism
  - Low computer-to-memory ratio
- @ Scale: resources matter
  - Distributed processing is mandatory
  - Parallel processing is very useful

Implementing graph applications is already difficult. Dealing with large scale systems on top (below, in fact) them is even harder.

Graph processing systems

- Provide simplified ways to develop graph processing applications
  - Typical scenario: analytics on single- or multi-node platfoms
  - Heterogeneity is becoming popular

- Target *productivity* and *performance*
  - Productivity => ease-of-implementation, development time
  - Performance => optimized back-ends / engines /runtimes
  - Portability comes “for free”

- Both commercial and academic, many open-source

Graph processing systems

Performance

- Systems for graph processing
- Separate users from backends
- Think Totem, Medusa, ….
- Think Giraph, GraphLab, PGX

Dedicated

Custom

- Specify application
- Choose the hardware
- Implement & optimize
- Think Graph500

Before Graphalytics: we did extensive performance evaluation of tens of systems (presented next), with considerable effort but without a unified view

- Use existing large scale distributed systems
- Mapping is difficult
- Parallelism is "free"
- Think MapReduce

Development Effort

GPU-enabled dedicated systems

Platforms we have evaluated

- Accelerated, Dedicated
  - Medusa
  - Totem
  - MapGraph
- In progress…
  - Ligra
  - Gunrock

Medusa

- Enables the use of GPUs for graph processing
  - Single-node, multiple GPUs
  - In-memory processing
- Simple API that hides GPU programming
  - Edge- / vertex-granularity that enables fine-grained parallelism.
  - API calls are grouped in kernels
  - Kernels are scheduled on one or multiple GPUs
- Run-time for communicating with the GPU

Totem

- Enables *single-node* heterogeneous computing on graphs
  - C+CUDA+API for specifying applications
  - Based on BSP
- Partitions the data (edge-based) between CPUs and GPUs
  - Based on processing capacity
  - Minimizing the overhead of communication
    - Buffer schemes, aggregation, smart partitioning
- Shows promising performance
  - BFS
  - PageRank
  - Betweenness centrality

MapGraph

- Target at high performance graph analytics on GPUs.
- API based on the Gather-Apply-Scatter (GAS) model as used in GraphLab.
  - Productivity-oriented API
- Single GPU available and Multi-GPU ready
  - Also available in a CPU-only version

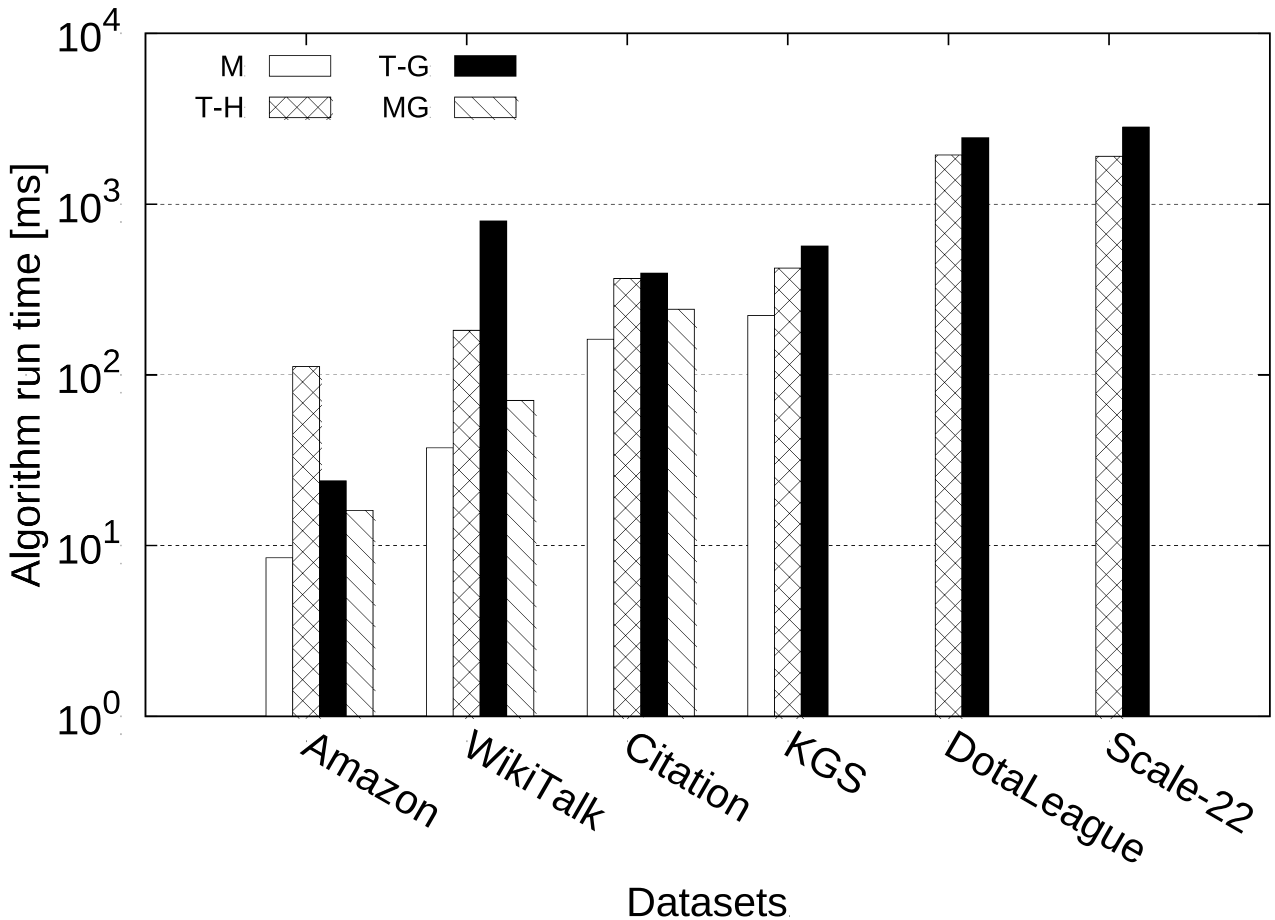
Evaluation setup

- Use GPU-enabled graph platforms to compare their performance*
- Datasets:
  - SNAP repository
  - The Game Trace Archive
  - Graph500 generated benchmarks
    - Scale-22/Synth
- Algorithms
  - BFS (traversal)
  - **PageRank**
  - Weakly connected components

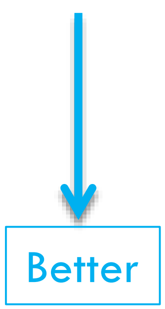
*Yong Guo et. al: An Empirical Performance Evaluation of GPU-Enabled Graph-Processing Systems. CCGrid 2015, May 2015

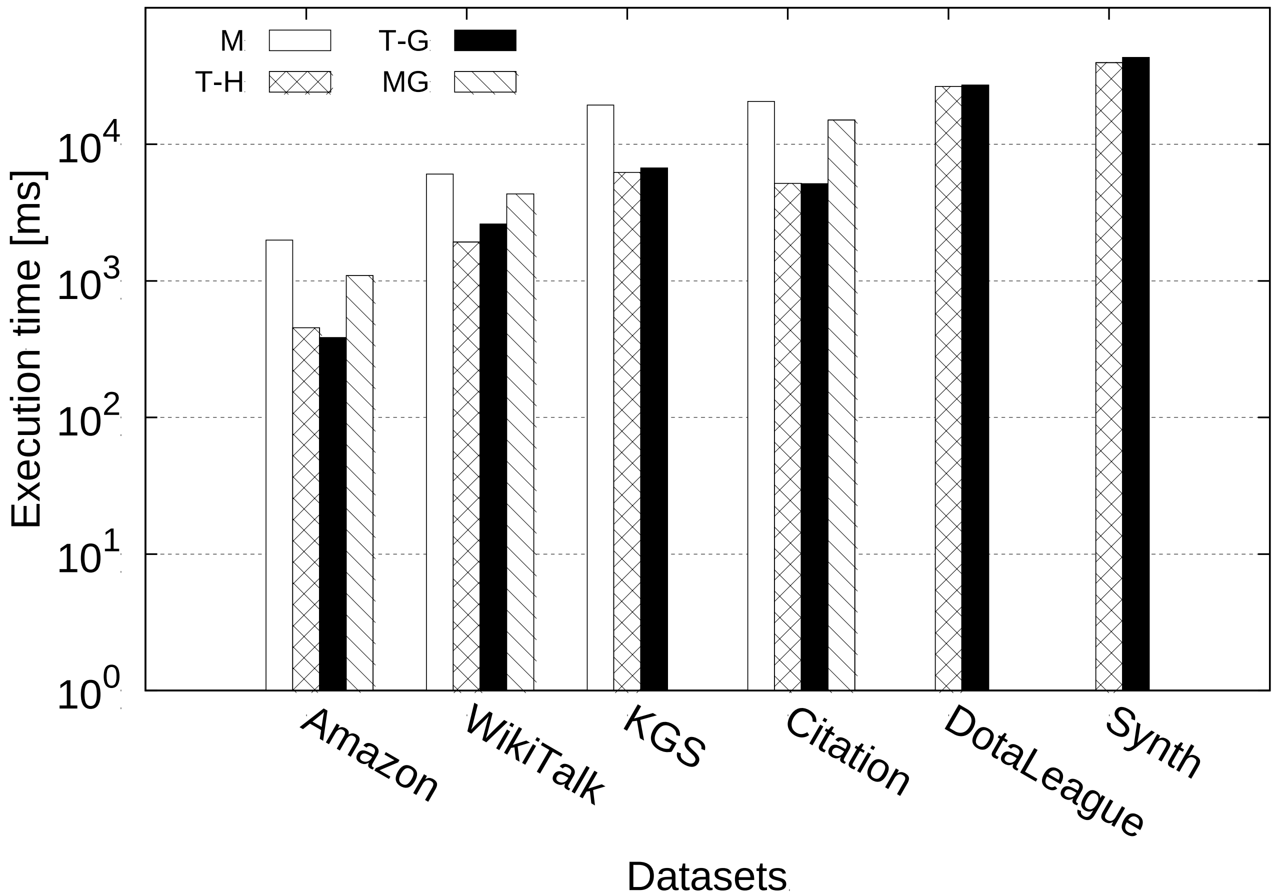
PageRank [algorithm]

Better



PageRank [full]





Lessons learned

- Brave attempts to enable the use of GPUs *inside* graph processing *systems*
- Every system has its own quirks
  - Lower level programming allows more optimizations, better performance
  - Higher level APIs allow more productivity
- No clear winner, performance-wise

- Challenge:
  - Distributed accelerated graph-processing

Distributed/Large Scale platforms

Interesting platforms

- Distributed or non-distributed
- Dedicated or generic

Oracle Labs
PGX

Distributed (Generic)

Distributed
(Dedicated)

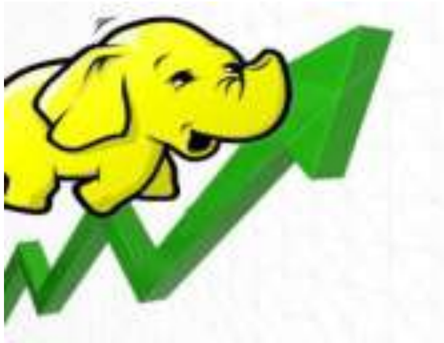
Non-distributed
(Dedicated)

Hadoop (Generic)

- The most popular MapReduce implementation
  - Generic system for large-scale computation
- Pros:
  - Easy to understand model
  - Multitude of tools and storage systems
- Cons:
  - Express the graph application in MapReduce
  - Costly disk and network operations
  - No specific graph processing optimizations

Hadoop2 with YARN (Generic)

- Next generation of Hadoop
  - Supports old MapReduce jobs
  - Designed to facilitate multiple programming models (frameworks, e.g., Spark)
- Separates resource management (YARN) and job management
  - MapReduce uses resources provided by YARN



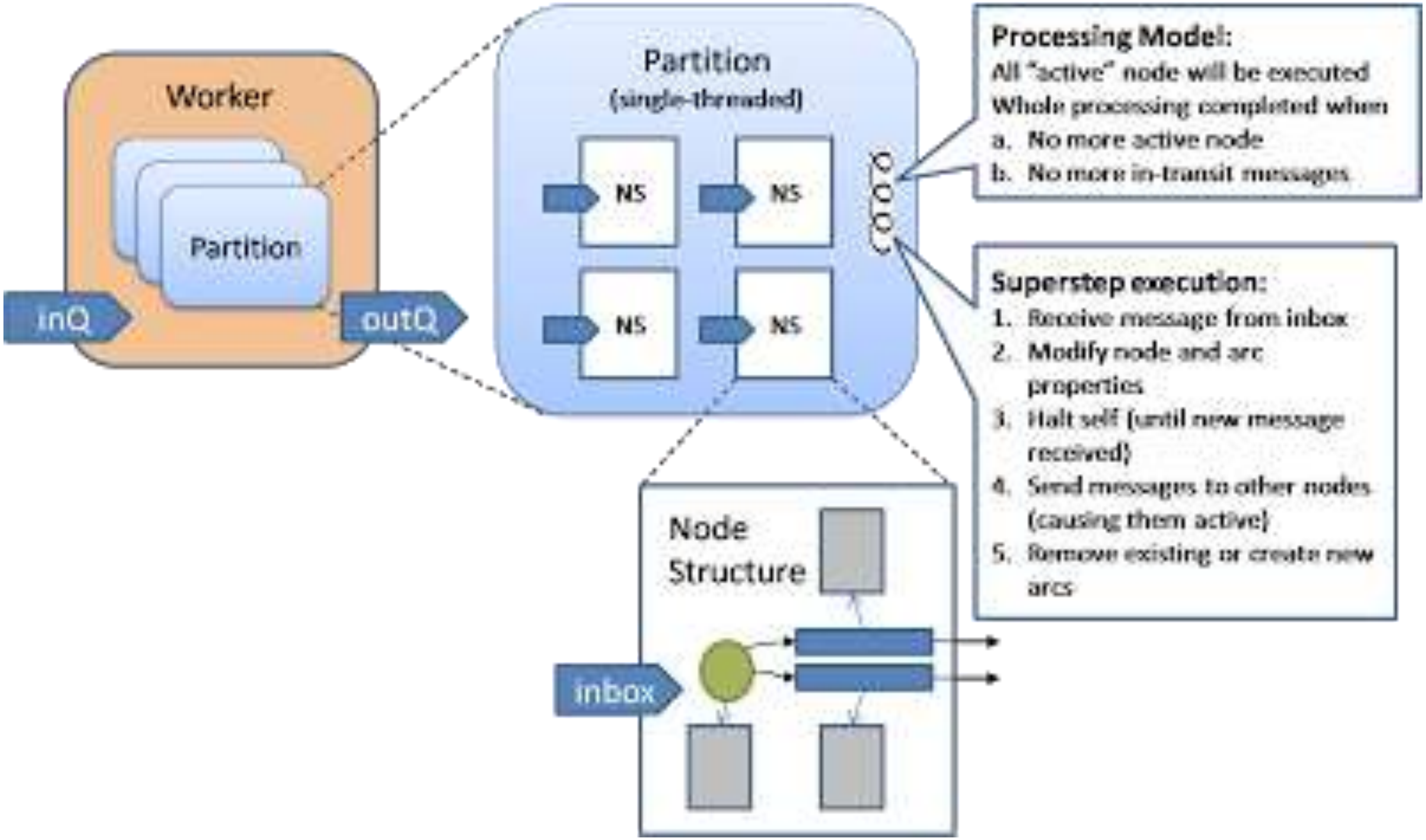
Stratosphere (Generic)

- Now Apache Flink
- Nephele resource manager
  - Scalable parallel engine
  - Jobs are represented as DAGs
  - Supports data flow in-memory, via network, or via files
- PACT job model
  - 5 second-order functions (MapReduce has 2): Map, Reduce, Match, Cross, and CogGroup
  - Code annotations for compile-time plans
  - Compiled as DAGs for Nephele

Pregel: dedicated graph-processing

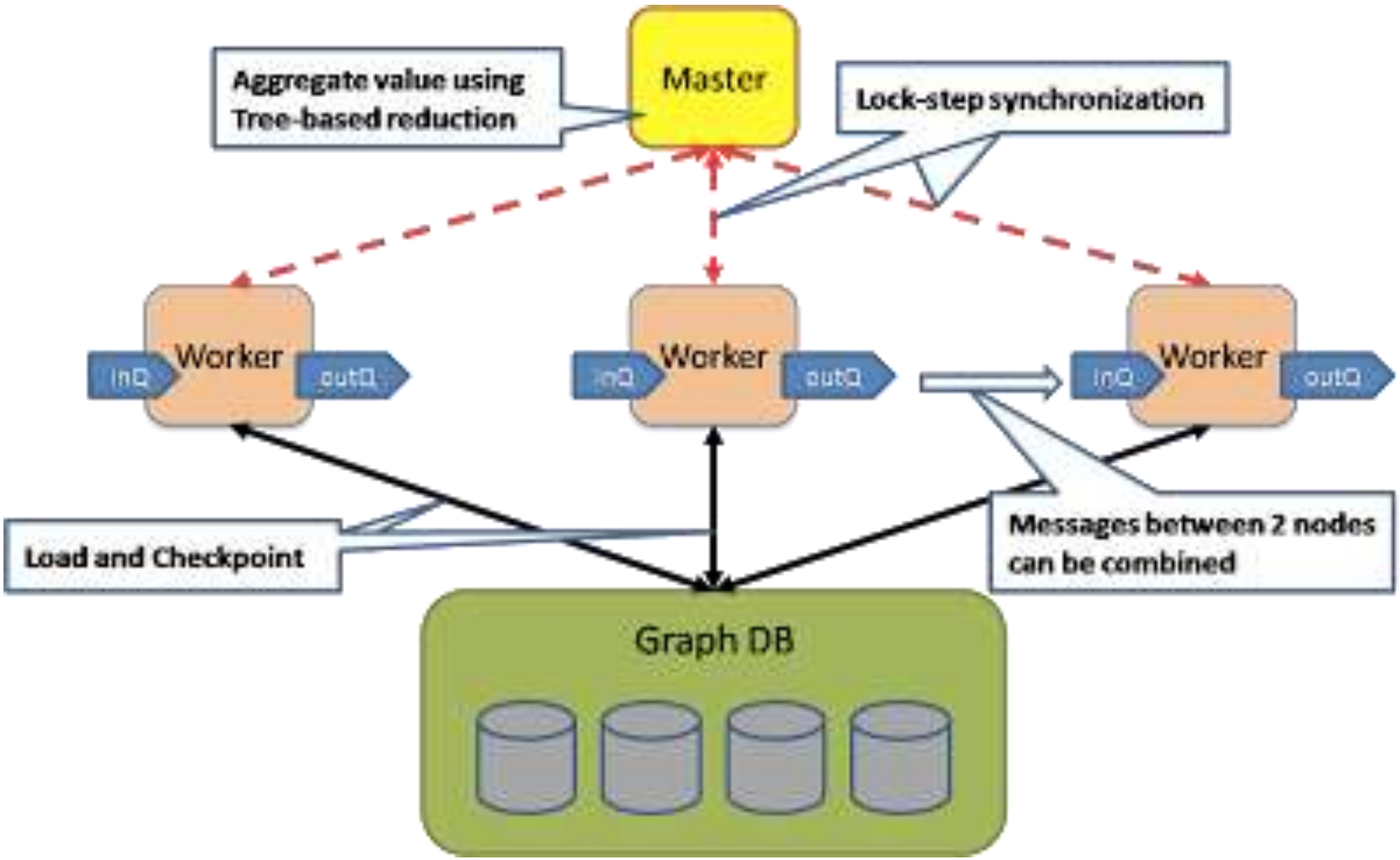
- Proposed a vertex-centric model for graph processing
  - Graph-to-graph transformations
- Front-end:
  - Write the computation that runs on all vertices
  - Each vertex can vote to halt
    - All vertexes halt => terminate
  - Can add/remove edges and vertices
- Back-end:
  - Uses the BSP model
  - Message passing between nodes
    - Combiners, aggregators
  - Checkpointing for fault-tolerance

Pregel



Worker
Partition
inQ
outQ
Partition
(single-threaded)
NS
NS
NS
NS
Processing Model:
All "active" node will be executed
Whole processing completed when
a. No more active node
b. No more in-transit messages
Superstep execution:
1. Receive message from inbox
2. Modify node and arc properties
3. Halt self (until new message received)
4. Send messages to other nodes (causing them active)
5. Remove existing or create new arcs
Node Structure
inbox

Pregel



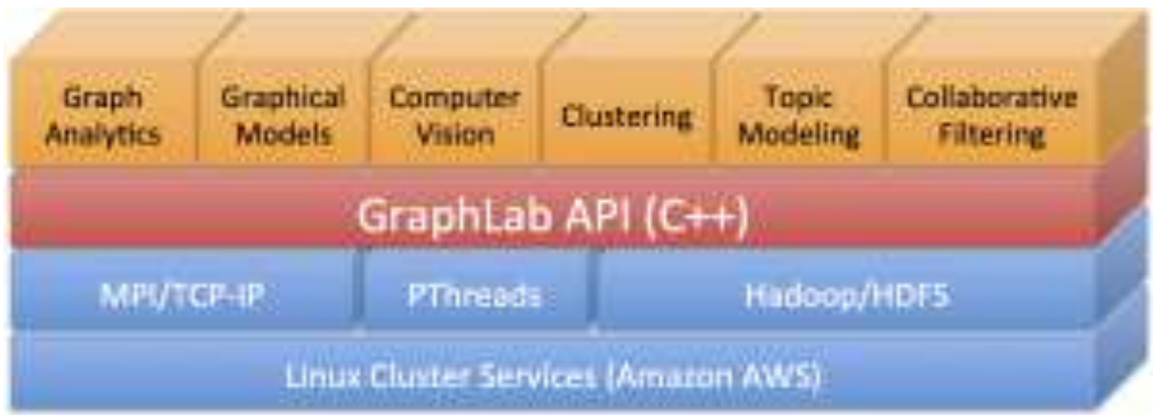
Aggregate value using Tree-based reduction
Master
Lock-step synchronization
Worker
inQ
outQ
Worker
inQ
outQ
Worker
inQ
outQ
Load and Checkpoint
Messages between 2 nodes can be combined
Graph DB

Apache Giraph (Dedicated)

- Based on the Pregel model
- Uses YARN as back-end (yet another framework)
- In-memory
  - Limitations in terms of partition sizes
  - Spilling to disk added recently, removes memory limitations
- Enables
  - Iterative data processing
  - Message passing, aggregators, combiners

GraphLab (Dedicated)

- Distributed programming model for machine learning
  - Provides an API for graph processing, C++ based (now Python)



- All in-memory
- Supports asynchronous processing
- GraphChi is its single-node version, Dato as GraphLab company

Neo4J (Dedicated)

- Very popular graph database
  - Graphs are represented as relationships and annotated vertices
- Single-node system
  - Uses parallel processing
  - Additional caching and query optimizations
  - All in-memory
- The most widely used solutions for medium-scale problems
- Cluster version in development

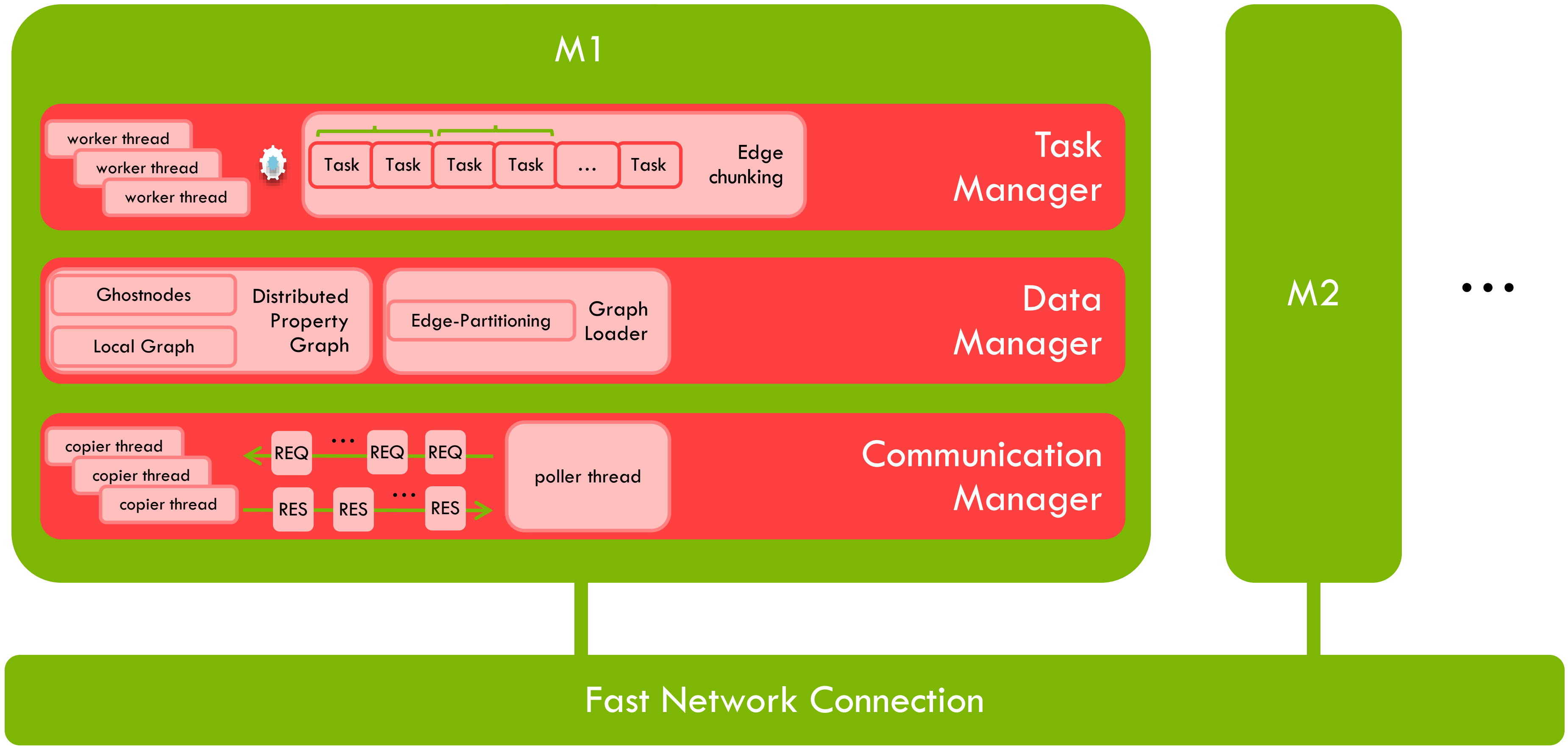
PGX.D (Dedicated)

- Designed for beefy clusters
  - Fully exploits the underlying resources of modern beefy cluster machines
- Low-overhead communication mechanism
  - Lightweight cooperative context switching mechansim
- Support for data-pulling (also data-pushing)
  - Intuitive transformation of classical graph algorithms
- Reducing traffic and balancing workloads
  - Several advanced techniques: Selective Ghostnodes, edge based partitioning, edge chunking

Attend presentation of SC15 article!

ORACLE

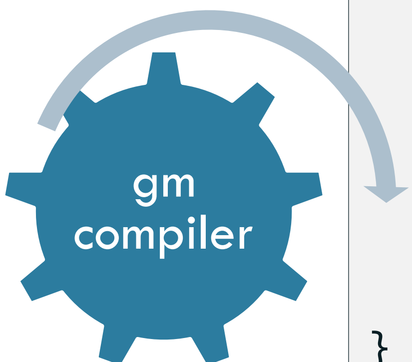
PGX.D: System Design Overview



PGX.D: Programming Model

High level programming model for Neighborhood Iteration Tasks

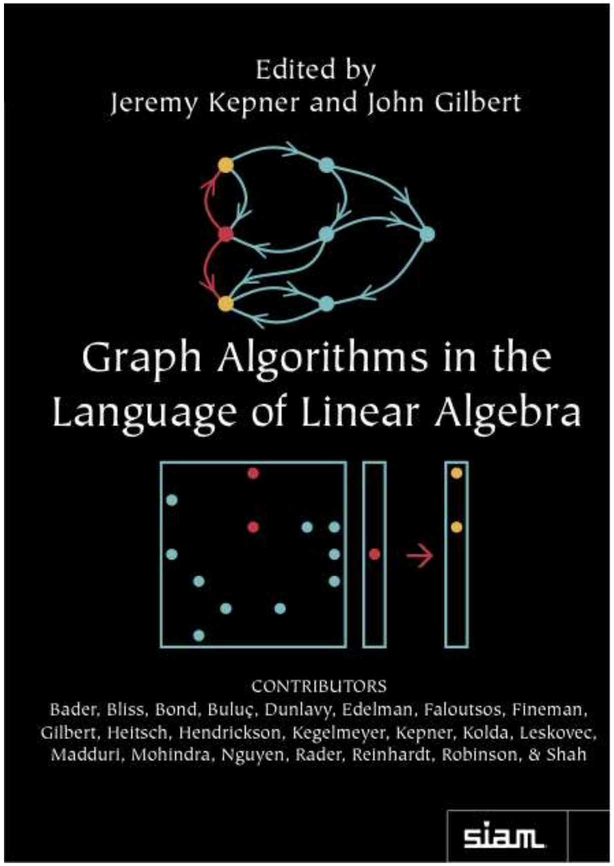
```
foreach(n: G.nodes)
    foreach(t: n.Nbrs)
        n.foo += t.bar
```



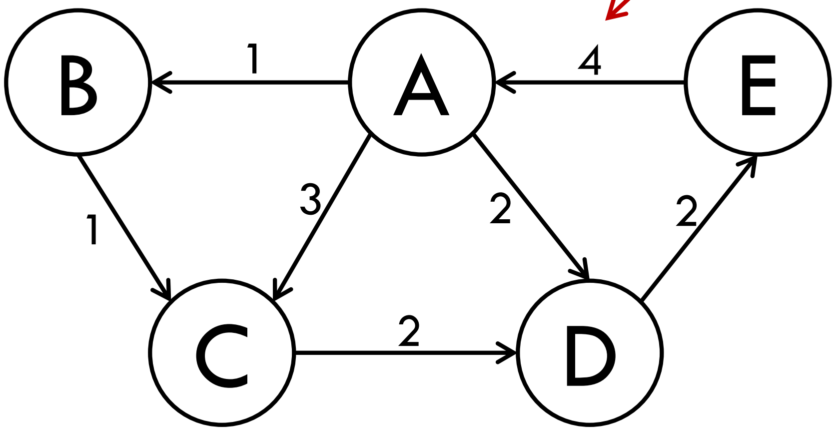

```
class my_task_pull : public innbr_iter_task {
  void run(..) {
    read_remote(get_nbr_id(), bar);
  }
  void read_done(void* buffer,..) {
    int foo_v = get_local<int>(node_id, foo);
    int bar_v= get_data<int>(buffer);
    set_local(node_id, foo_v + bar_v, foo);
  }
}
```

GraphMat (Dedicated)

- Vertex programming as front-end and sparse matrix operations as back-end
  - “Matrix level performance with vertex program productivity”
  - Unifying vertex programming w linear algebra is new



Example



$$G^T = \begin{array}{c c} & \begin{array}{ccccc} A & B & C & D & E \end{array} \\ \begin{array}{c} A \\ B \\ C \\ D \\ E \end{array} & \begin{bmatrix} - & - & - & - & 4 \\ 1 & - & - & - & - \\ 3 & 1 & - & - & - \\ 2 & - & 2 & - & - \\ - & - & - & 2 & - \end{bmatrix} \end{array}$$

<u>A Vertex Program (Single Source Shortest Path) ~ Giraph</u>

SEND_MESSAGE : message := vertex_distance

PROCESS_MESSAGE : result := message + edge_value

REDUCE : result := min(result, operand)

APPLY : vertex_distance = min(result, vertex_distance)

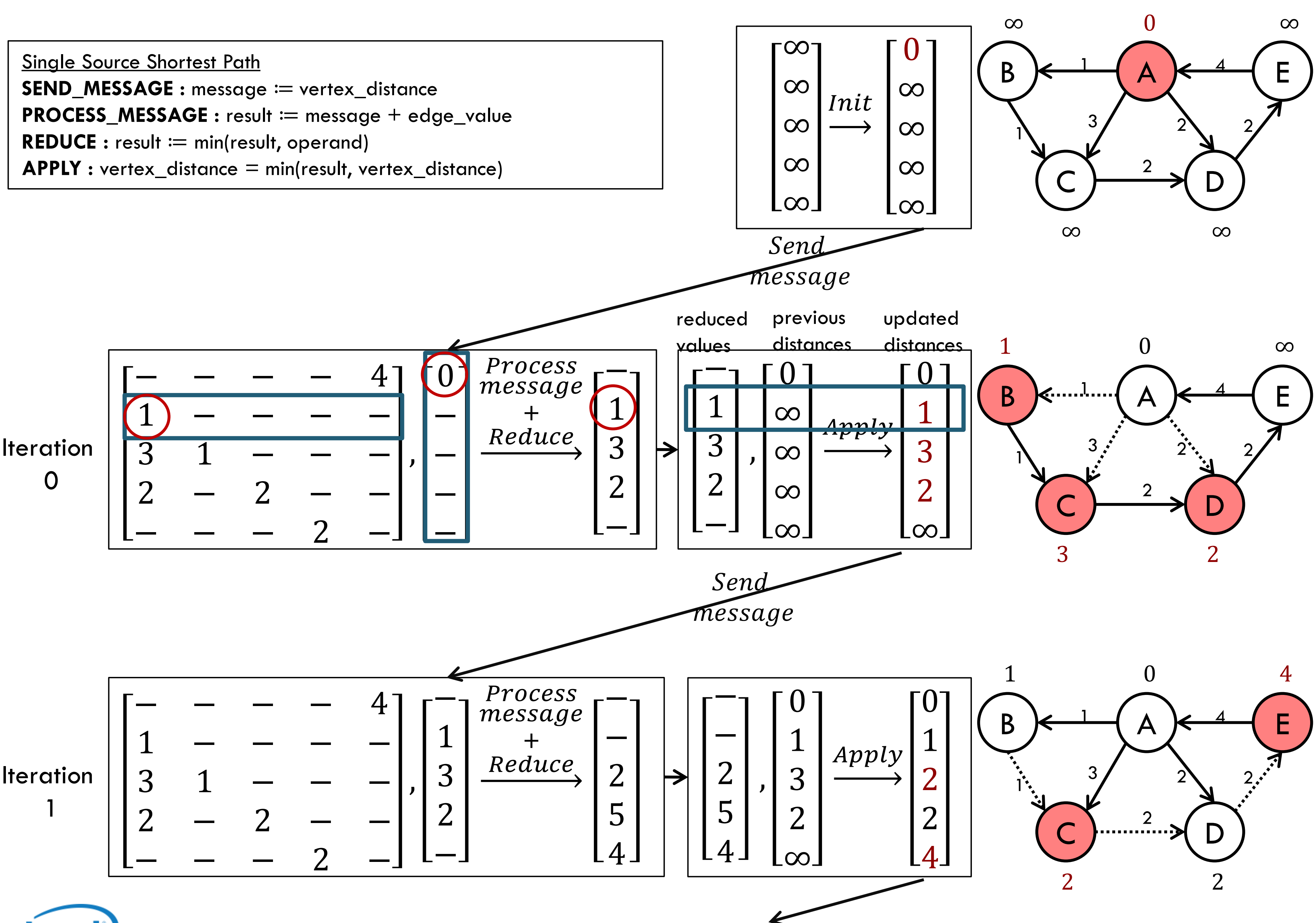
Single Source Shortest Path

SEND_MESSAGE : message ≔ vertex_distance

PROCESS_MESSAGE : result ≔ message + edge_value

REDUCE : result ≔ min(result, operand)

APPLY : vertex_distance = min(result, vertex_distance)



Setup*

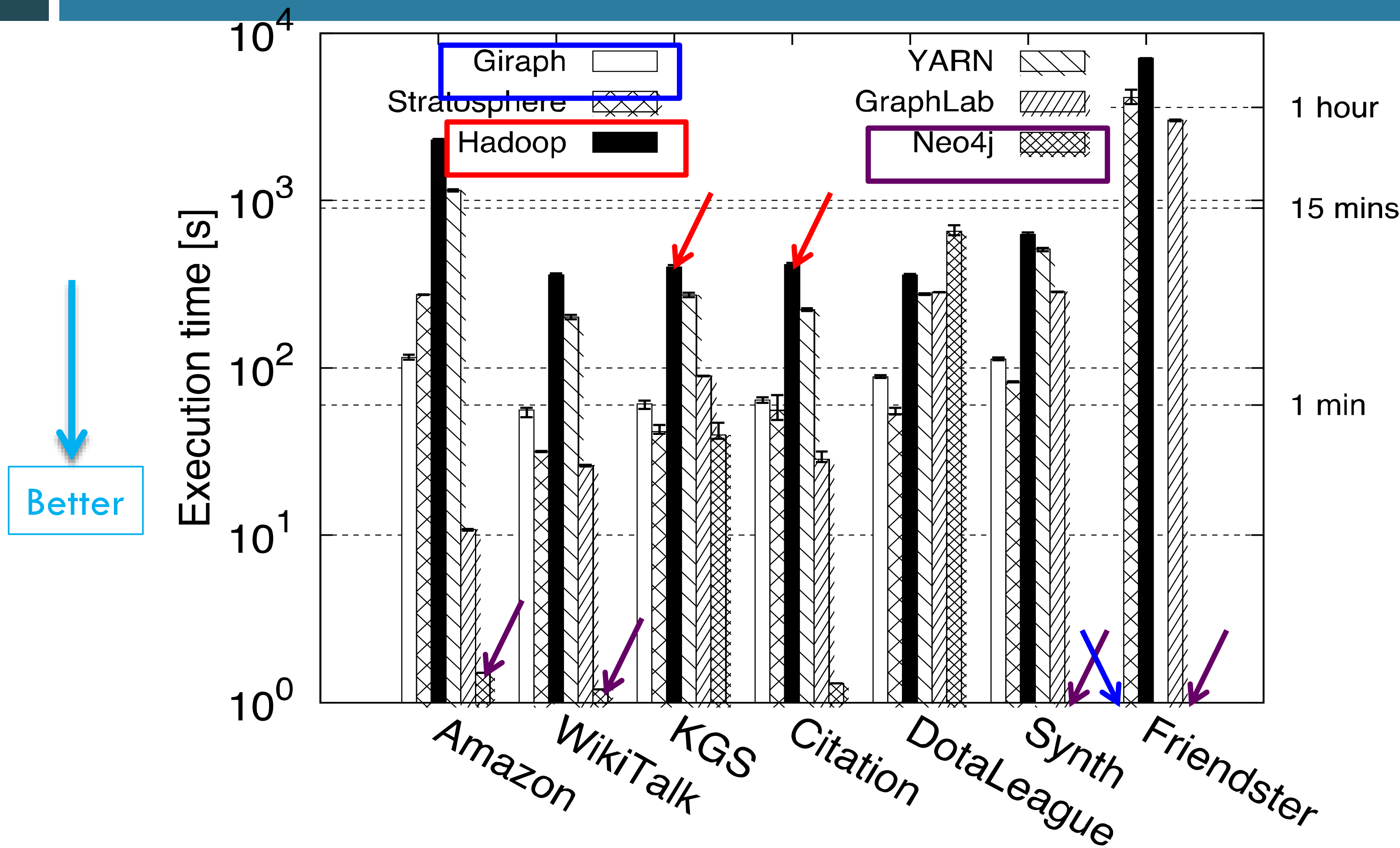
- Benchmarking-like experiment
  - 6 algorithms:
    - Stats, BFS, PageRank, connected components, community detection, graph evolution.
  - 7 data-sets
    - From 1.2M to 1.8B edges, various types, real and synthetic
  - Many platforms
- Implement **all algorithms** on **all platforms**
- Run and compare many aspects, including …
  - Performance
  - Weak / Strong, Horizontal / Vertical scalability
- Estimate usability*

*Y. Guo, M. Biczak, A. L. Varbanescu, A. Iosup, C. Martella, and T. L. Willke. How Well do Graph-Processing Platforms Perform? An Empirical Performance Evaluation and Analysis, IPDPS 2014

Hardware for Main Experiments

- DAS4: a multi-cluster Dutch grid/cloud
  - Intel Xeon 2.4 GHz CPU (dual quad-core, 12 MB cache)
  - Memory 24 GB
  - 1 Gbit/s Ethernet network
- Size
  - Most experiments take 20 working machines
  - Up to 50 working machines
- HDFS used as distributed file system

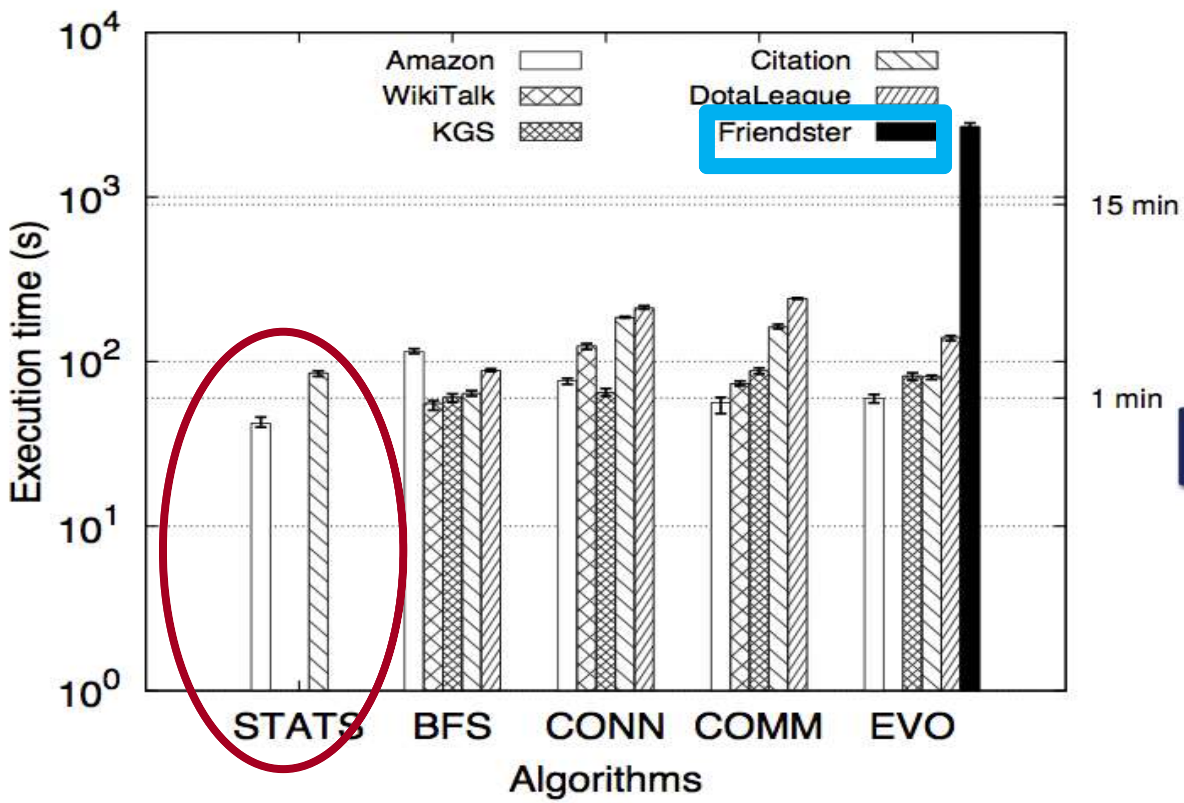
BFS: Results for all-2-all



No platform runs fastest for all graphs, but Hadoop is the worst performer.
Not all platforms can process all graphs, but Hadoop processes everything.

Giraph:
Results for (algo*,platform*)

Better

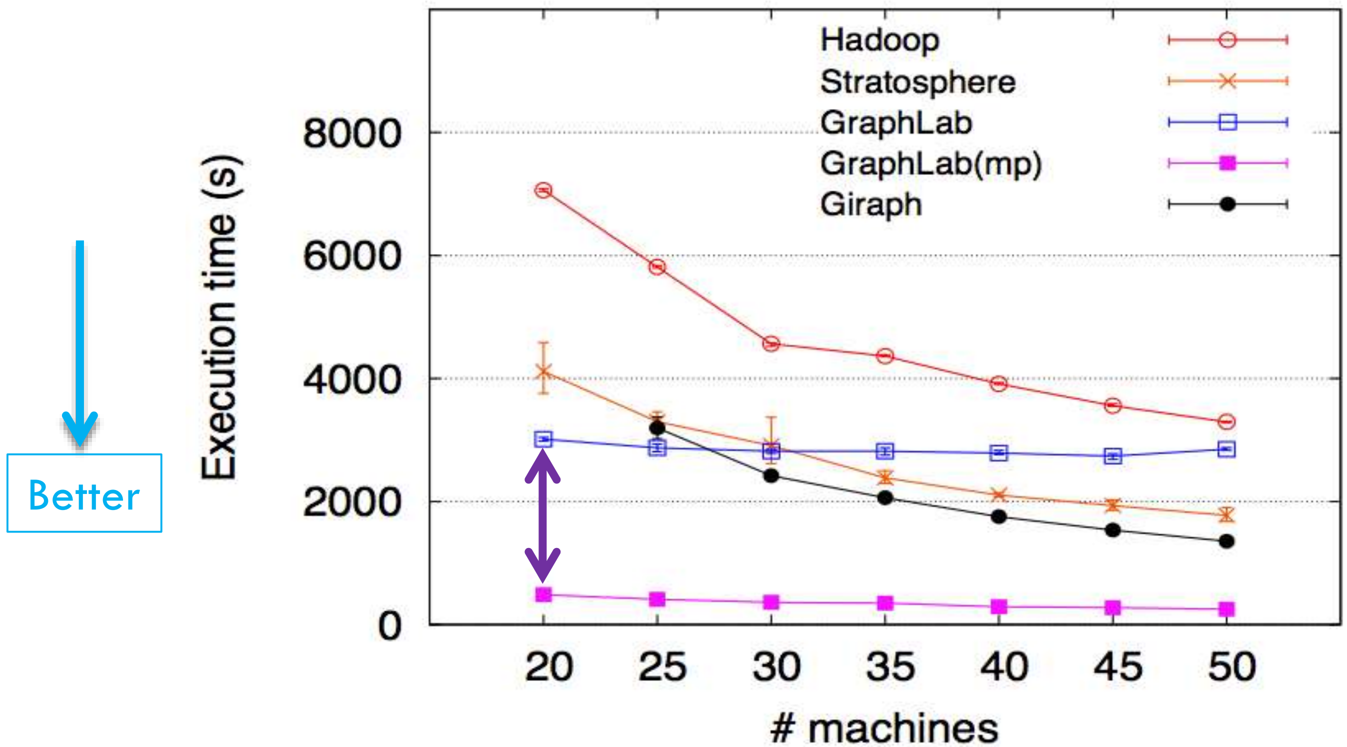


DAS 4

Storing the whole graph in memory helps Giraph perform well

Giraph may crash when graphs or number of messages large

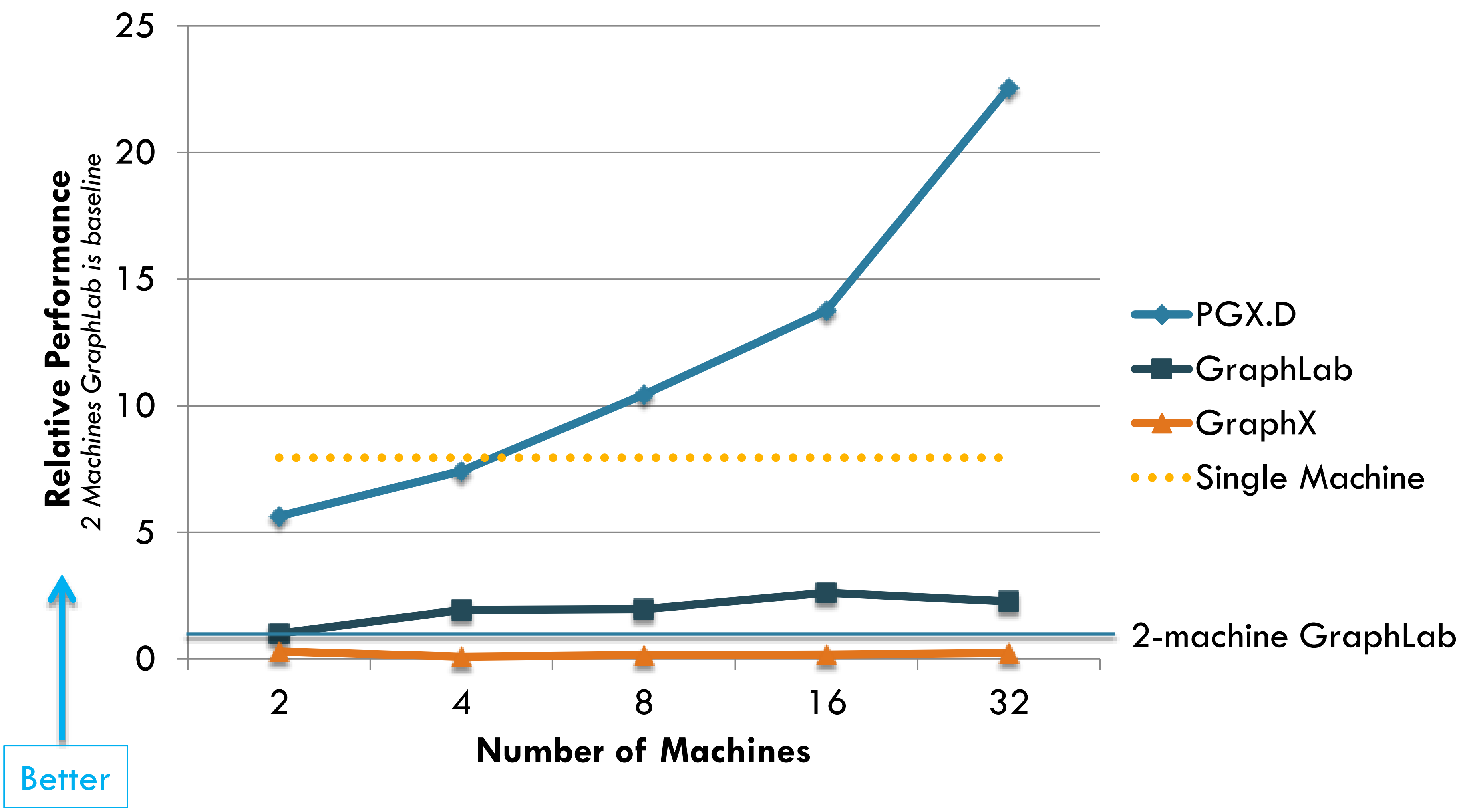
Horizontal scalability: BFS on Friendster (31 GB)



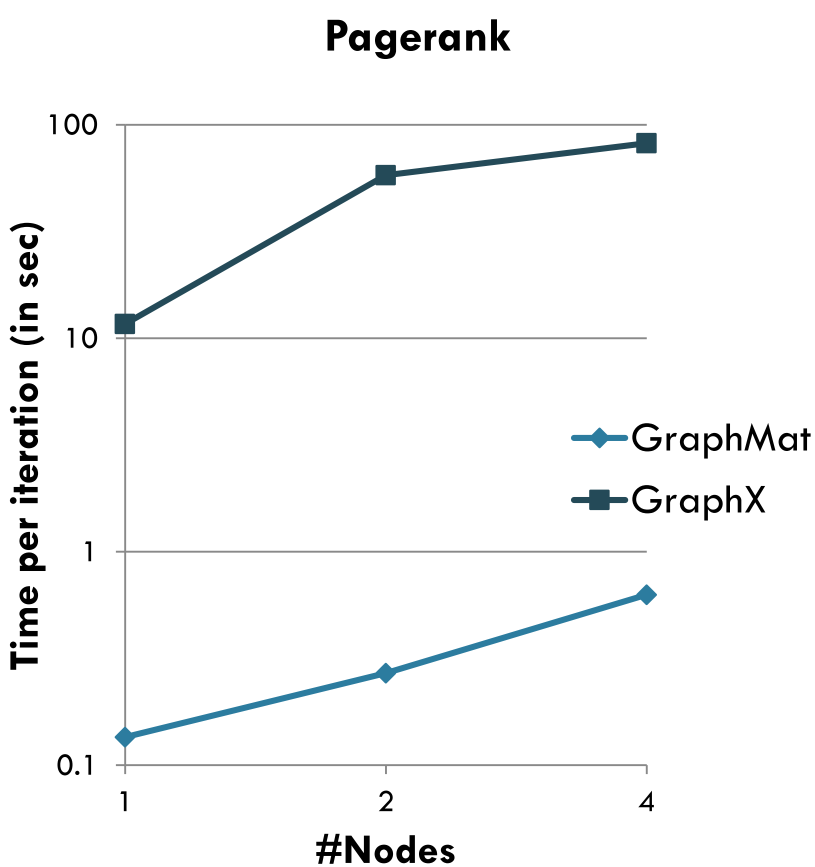
Using more computing machines can reduce execution time

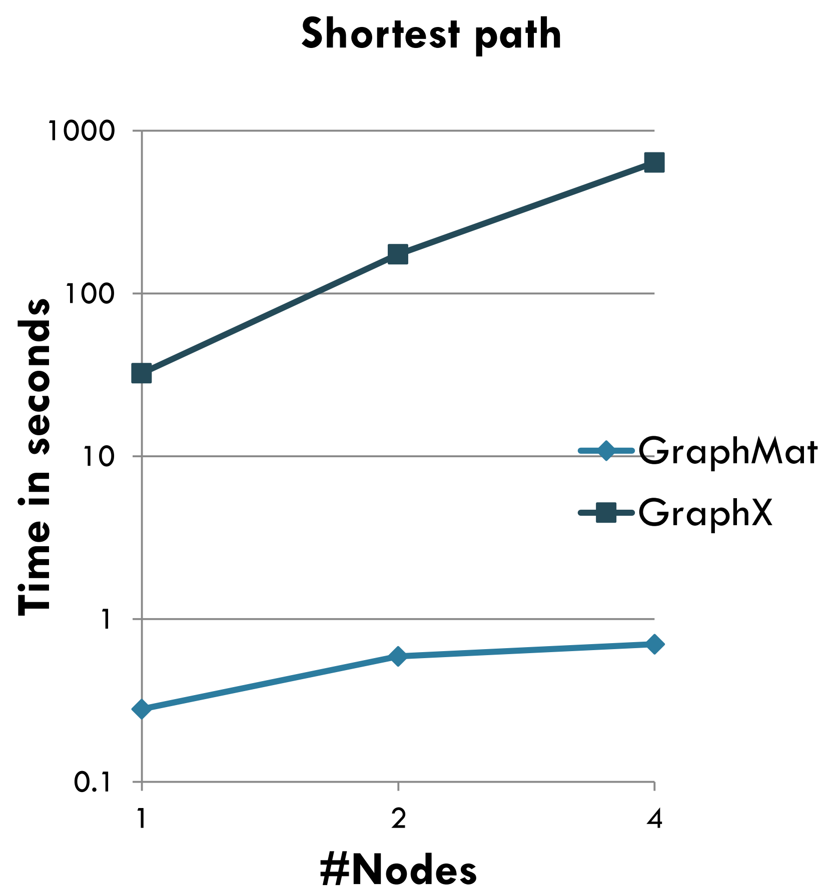
Tuning needed for horizontal scalability, e.g., for GraphLab, split large input files into number of chunks equal to the number of machines

PGX.D: Performance Evaluation
(PageRank, Twitter, Infiniband)



GraphMat Weak Scalability (Preliminary results, Amazon EMR)

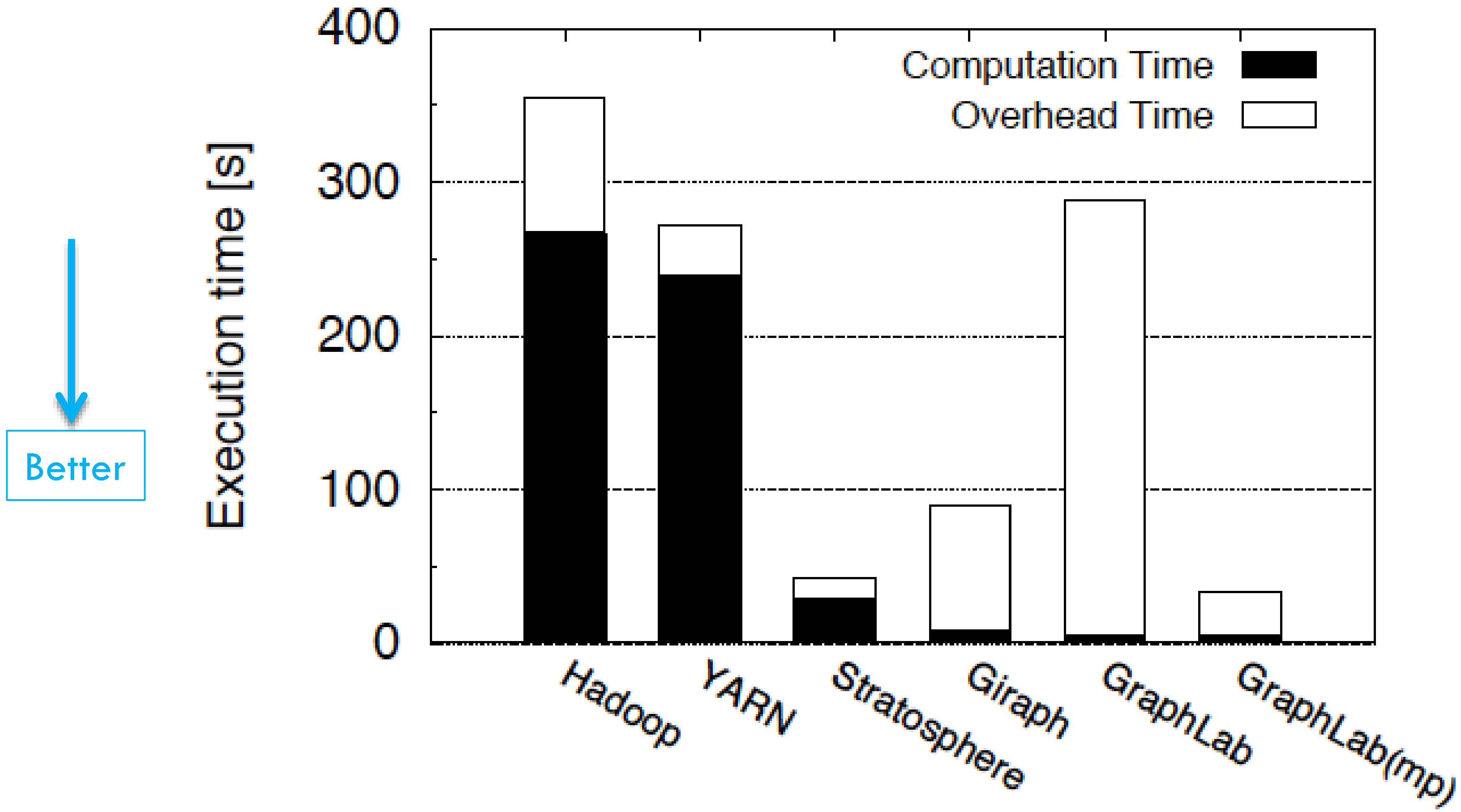




Better

Weak scaling, 128 M edges/node
Graph500 scale 23-24-25

Overhead (BFS, DotaLeague)



We need new metrics, to capture meaning of computation time (more later)
In some systems, overhead is by and large wasted time (e.g., in Hadoop)

Additional Overheads
Data ingestion time

- Data ingestion
  - Batch system: one ingestion, multiple processing
  - Transactional system: one ingestion, one processing

- Data ingestion matters even for batch systems

	Amazon	DotaLeague	Friendster
HDFS	1 second	7 seconds	5 minutes
Neo4J	4 hours	**days**	**n/a**

Productivity

	Hadoop(Java)	Stratosphere(Java)	Giraph(Java)	GraphLab(C++)	Neo4j(Java)
BFS	1 d, 110 loc	1 d, 150 loc	1 d, 45 loc	1 d, 120 loc	1 h, 38 loc
CONN	1.5 d, 110 loc	1 d, 160 loc	1 d, 80 loc	0.5 d, 130 loc	1 d, 100 loc

- Low throughput in terms of LOC for all models
- Days to hours development time for the simpler applications

We need better productivity metrics!

Lessons learned*

- Performance is function of (Dataset, Algorithm, Platform, Deployment)
  - Previous performance studies may lead to tunnel vision
- Platforms have their own drawbacks

Such manual evaluation is never comprehensive or scalable …
Adding PGX.D by hand would take 4-5 weeks!

  - Ease-of-use of a platform is very important

There are 20+ other interesting platforms …
Can we do better than manual ?

  - Strong vs weak scaling still a challenge

*All results and details:
http://www.pds.ewi.tudelft.nl/fileadmin/pds/reports/2013/PDS-2013-004-4.pdf

Questions?

Methodology

From single- to many- (to all ?) evaluations

A systematic approach

Graph Processing Platforms

- Platform: the combined hardware, software, and programming system that is being used to complete a graph processing task.

Neo4j the graph database

hadoop

APACHE GIRAPH

HyperGraphDB

GraphLab

Trinity

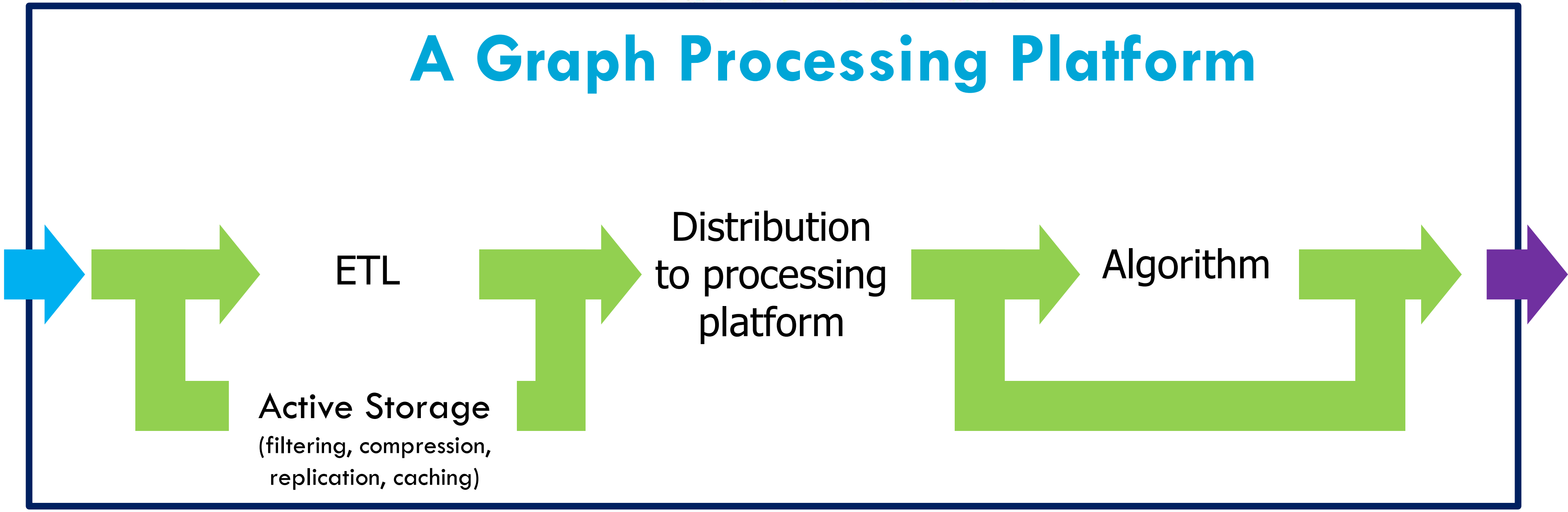
StratoSphere Above the Clouds

PROJECT PEGASUS

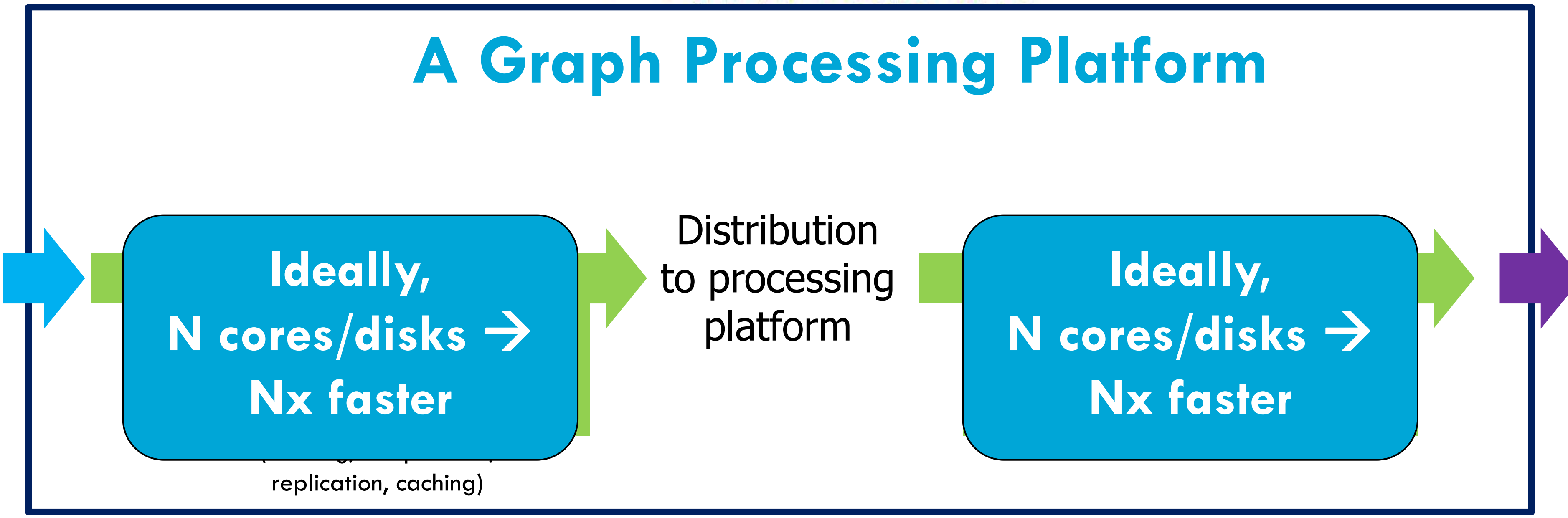
Which to choose?

What to tune?

Abstraction



Objectives: scalability & peformance



What does a benchmark consist of?

- Four main elements:
  - data schema: defines the structure of the data
  - workloads: defines the set of operations to perform
  - performance metrics: used to measure (quantitatively) the performance of the systems
  - execution rules: defined to assure that the results from different executions of the benchmark are valid and comparable

- Software as Open Source (GitHub)
  - data generator, query drivers, validation tools, ...

Evaluating graph-processing platforms

Metrics Diversity

Graph Diversity

Algorithm Diversity

Graphalytics = comprehensive benchmarking suite for graph processing across all platforms

LDBC The graph & RDF benchmark reference

spec Research

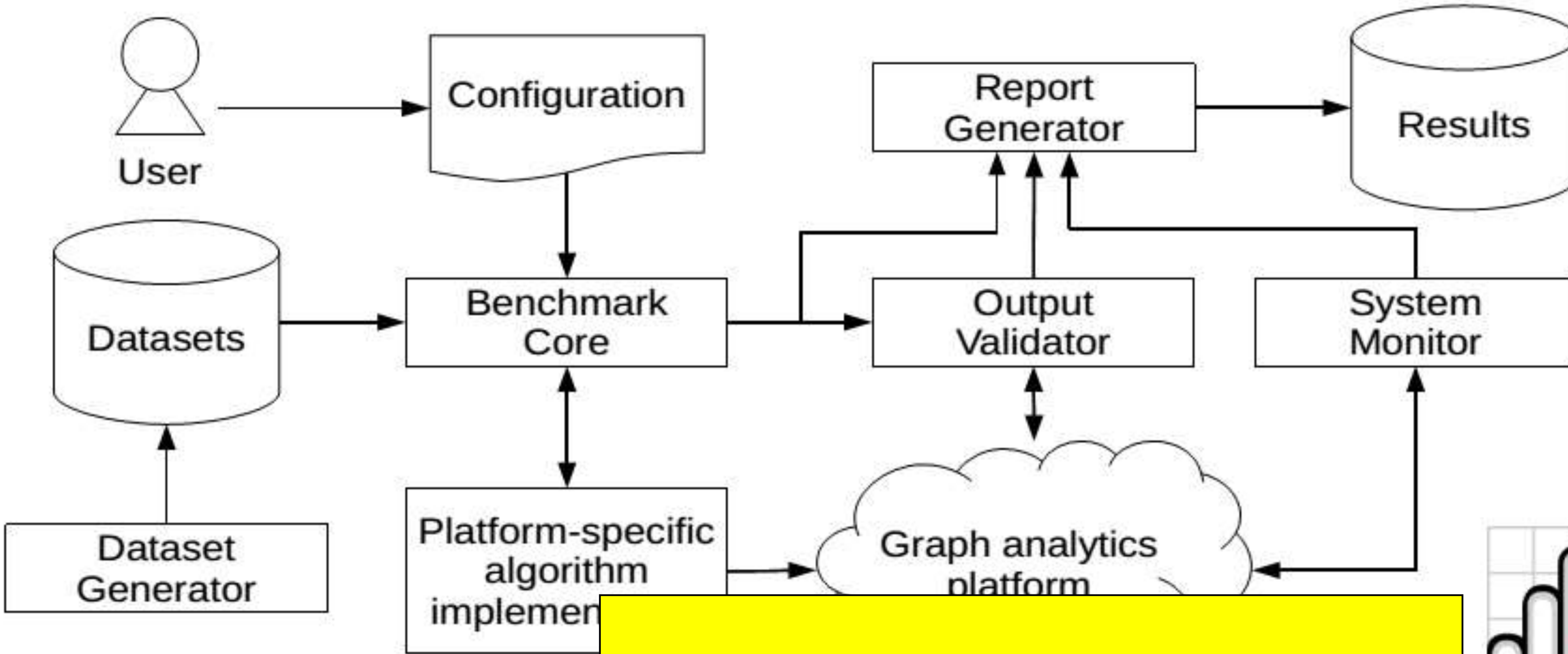
Graphalytics = A Challenging Benchmarking Process

- Methodological challenges
  - Challenge 1. Evaluation process
  - Challenge 2. Selection and design of performance metrics
  - Challenge 3. Dataset selection and analysis of coverage
  - Challenge 4. Algorithm selection and analysis of coverage
- Practical challenges
  - Challenge 5. Scalability of evaluation, selection processes
  - Challenge 6. Portability
  - Challenge 7. Result reporting

LDBC The graph & RDF benchmark reference

Y. Guo, A. L. Varbanescu, A. Iosup, C. Martella, T. L. Willke:
Benchmarking graph-processing platforms: a vision. ICPE 2014.

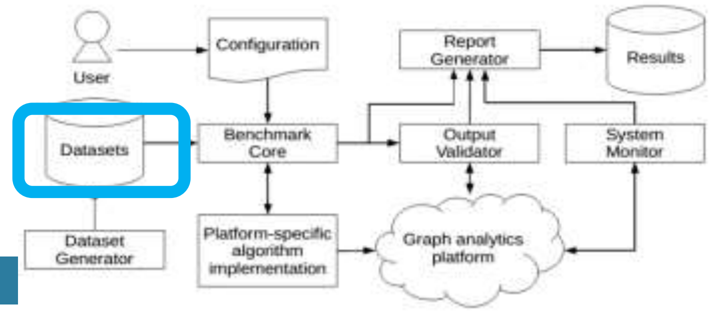
Graphalytics = Advanced Harness



Support of cloud-based platforms technically feasible, methodologically difficult

M. Capota et al., Graphalytics: A Big Data Benchmark for Graph-Processing Platforms. SIGMOD GRADES 2015

Graphalytics = Real & Synthetic Datasets



	Graphs	#V	#E	d	$\bar{D}$	Directivity
G1	Amazon	262,111	1,234,877	1.8	4.7	directed
G2	WikiTalk	2,388,953	5,018,445	0.1	2.1	directed
G3	KGS	293,290	16,558,839	38.5	112.9	undirected
G4	Citation	3,764,117	16,511,742	0.1	4.4	directed
G5	DotaLeague	61,171	50,870,316	2,719.0	1,663.2	undirected
G6	Synth	2,394,536	64,152,015	2.2	53.6	undirected
G7	Friendster	65,608,366	1,806,067,135	0.1	55.1	undirected

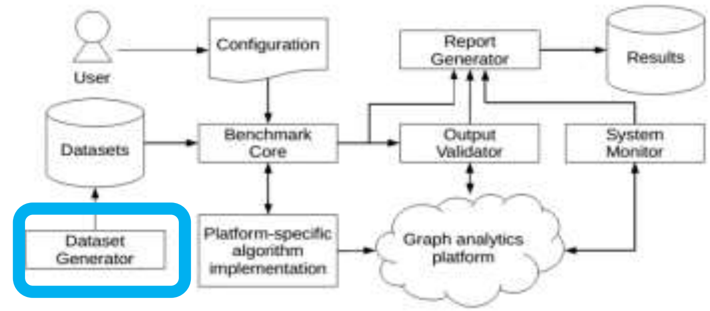
G8: LDBC DATAGEN synthetic graphs (described next)

SNAP GRAPH 500 **The Game Trace Archive**

https://snap.stanford.edu/ http://www.graph500.org/ http://gta.st.ewi.tudelft.nl/

Y. Guo and A. Iosup. The Game Trace Archive, NETGAMES 2012.

Graphalytics = Graph Generation w DATAGEN



DATAGEN Process

- Rich set of configurations
- More diverse degree distribution than Graph500
- Realistic clustering coefficient and assortativity

Graphalytics

Person Generation

Edge Generation

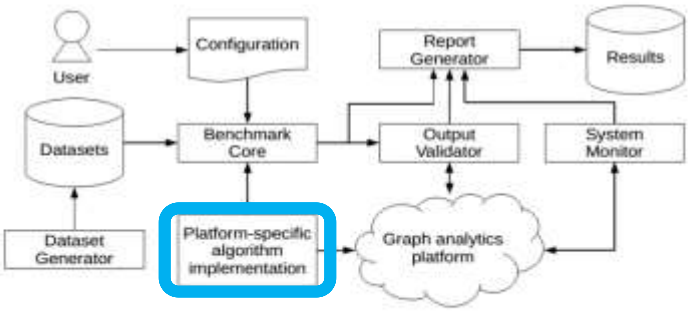
“Knows” graph serialization

Activity Generation

Activity serialization

Level of Detail

Graphalytics = Many Classes of Algorithms



- Literature survey of of metrics, datasets, and algorithms
  - 2009–2013, 120+ articles in 10 top conferences: SIGMOD, VLDB, HPDC,

Class	Examples	%
Graph Statistics	Diameter, Local Clust. Coeff., PageRank	16.1
Graph Traversal	BFS, SSSP, DFS	46.3
Connected Component	Reachability, BiCC, Weakly CC	13.4
Community Detection	Clustering, Nearest Neighbor, Label Propagation	5.4
Graph Evolution	Forest Fire Model, PAM	4.0
Other	Sampling, Partitioning	14.8

Y. Guo, M. Biczak, A. L. Varbanescu, A. Iosup, C. Martella, and T. L. Willke. How Well do Graph-Processing Platforms Perform? An Empirical Performance Evaluation and Analysis, IPDPS'14.

Graphalytics = Choke-Point Analysis

- Choke points are crucial technological challenges that platforms are struggling with
- Examples
  - Network traffic
  - Access locality
  - Skewed execution (stragglers)

Choke-point analyss often require fine-grained analysis of system operation, across many systems

- Challenge: Select benchmark workload based on real-world scenarios, but make sure benchmark covers important choke points

ongoing work

Coarse-grained vs Fine-grained Evaluation (1)

Coarse-grained Method

system viewed as a black-box

Algorithms, Datasets, Resources

Graph processing system

Coarse-grained performance metrics
(Overall Execution Time)

Fine-grained Method

system viewed as a white-box

Algorithms, Datasets, Resources

IO operations

Processing operations

Overheads

Fine-grained performance metrics
(Stage 3 time, straggler tasks)

Fine-grained evaluation method is more comprehensive

Coarse-grained vs Fine-grained Evaluation (2)

Abstract	**Granular**
Coarse-grained Method knowledge at conceptual level	**Fine-grained Method** knowledge at technical level

Graph Processing Systems

Distributed Infrastructure

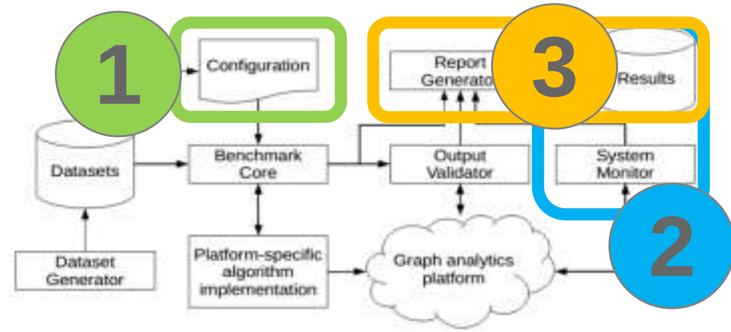
several performance results

Oracle Labs PGX

many performance results

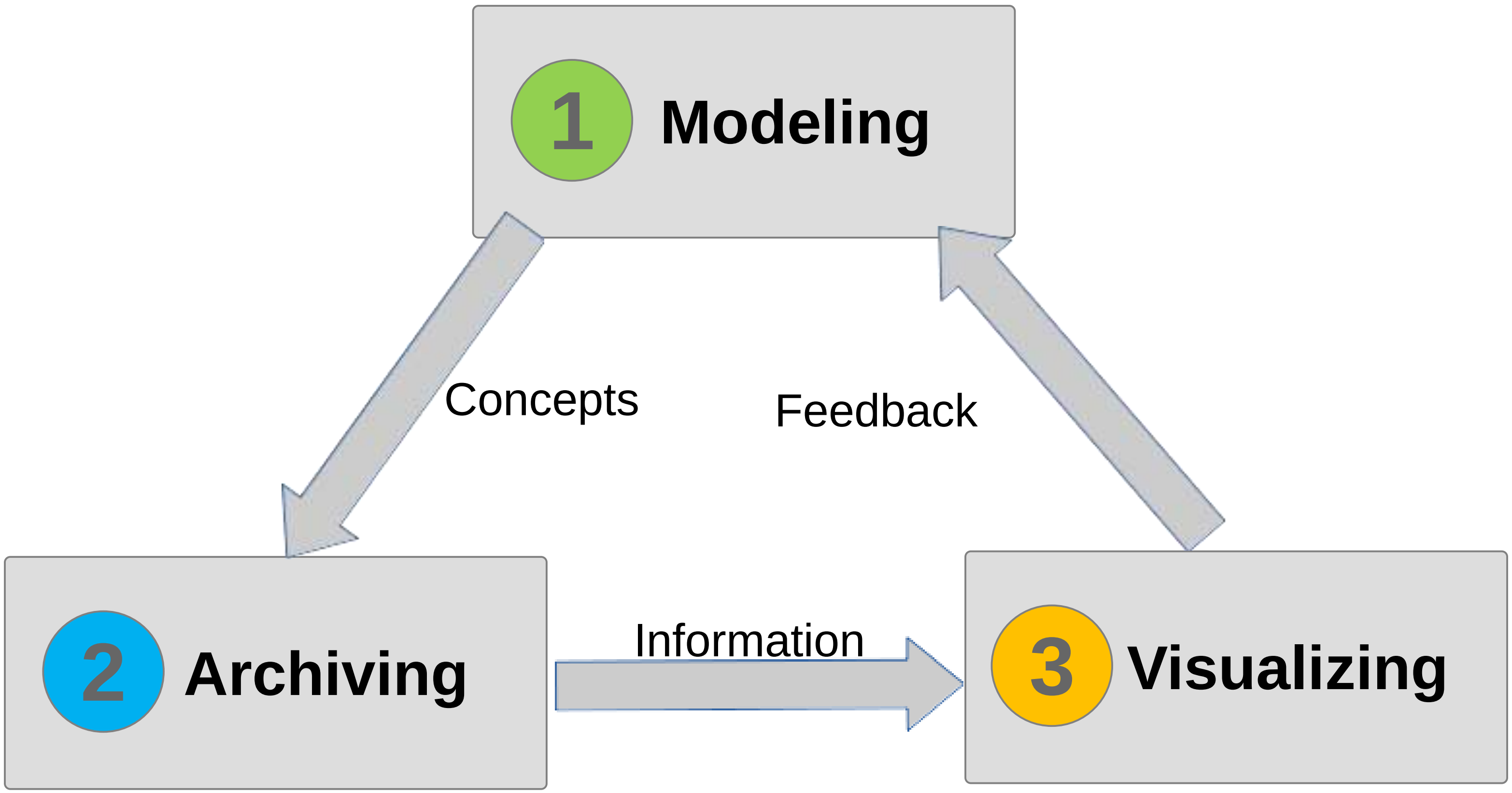
Fine-grained evaluation method is more comprehensive
… but more time-consuming, esp. to implement

Graphalytics: Granula Overview



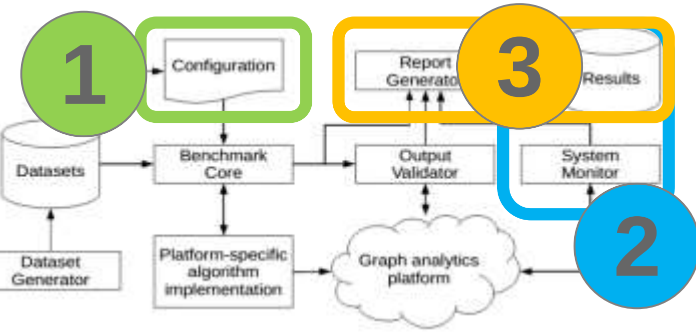
Granular

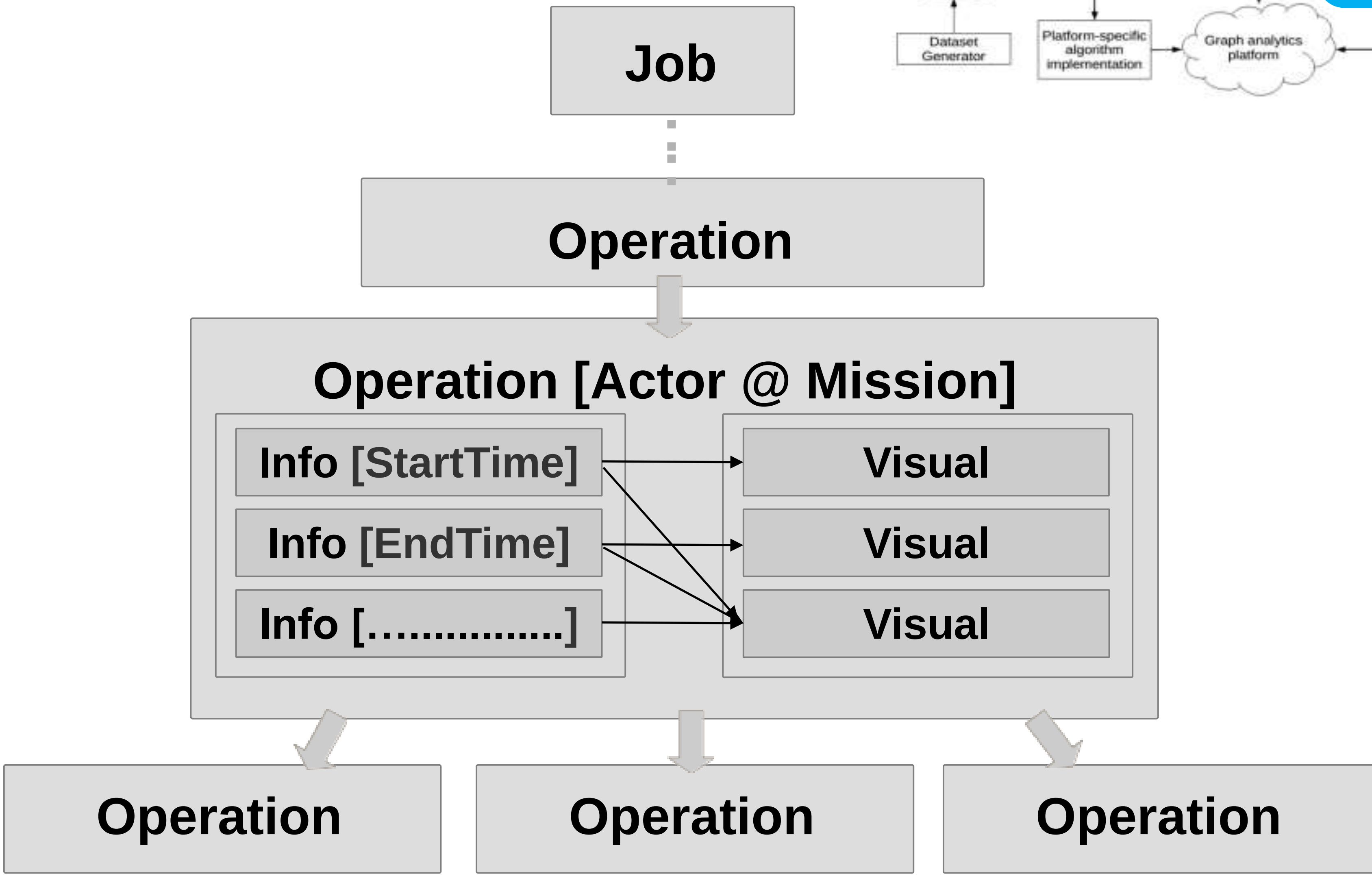
Fine-grained Method



https://github.com/tudelft-atlarge/granula/

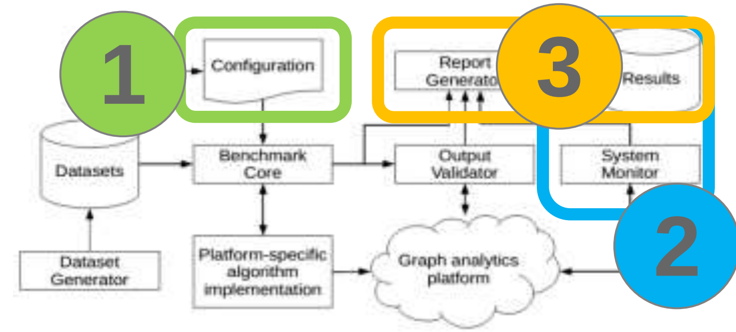
1 Granula Modeller





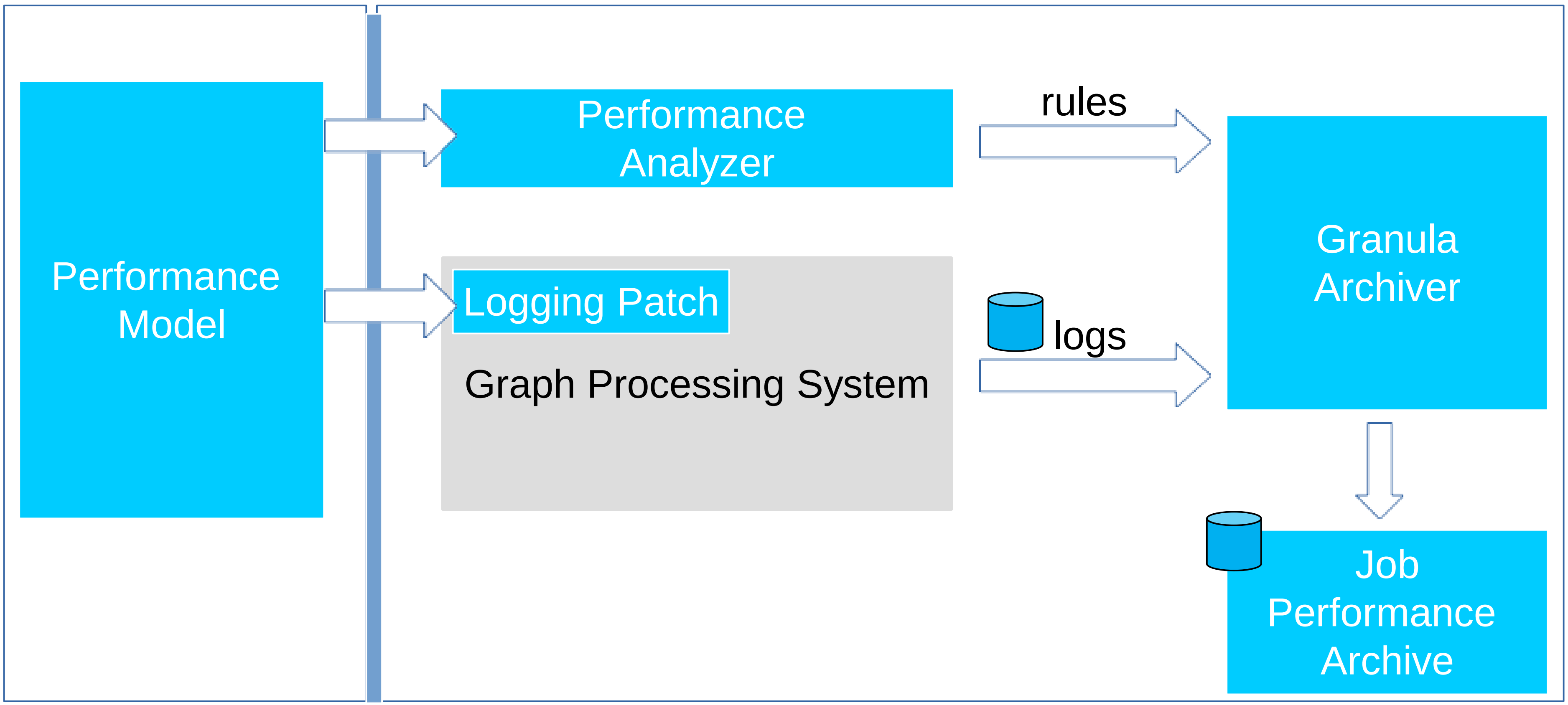
Time-consuming, expert-only, done only once

2 Granula Archiver



Modeling

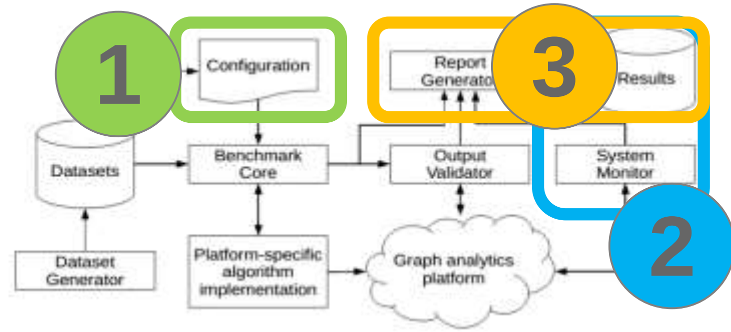
Archiving

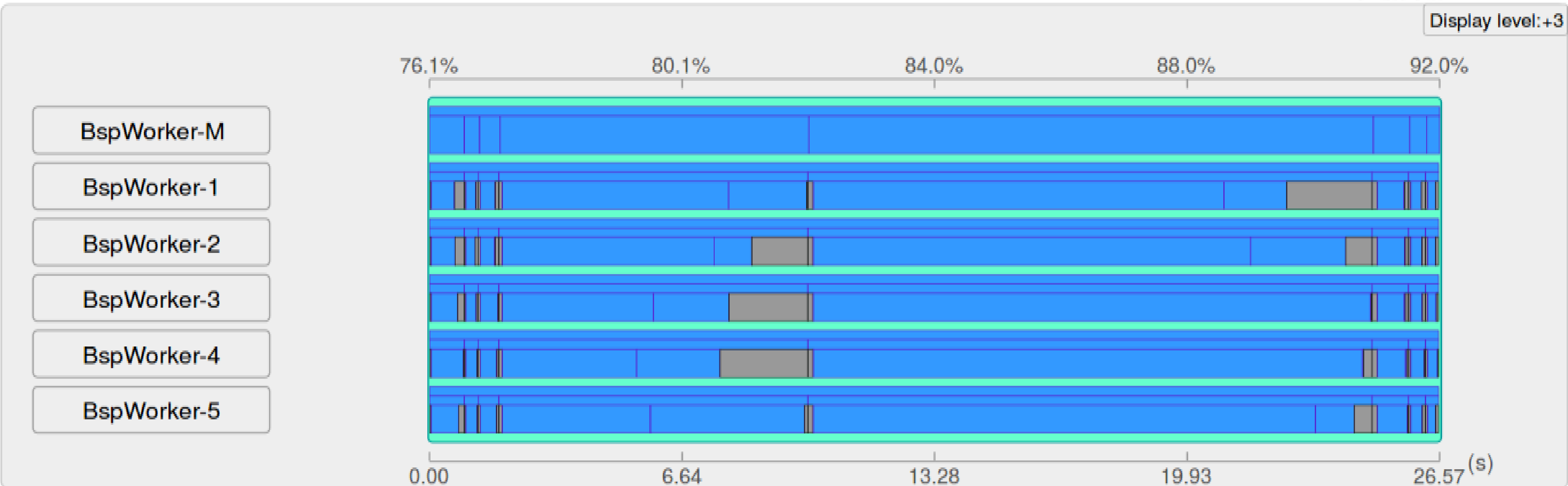


Time-consuming, minimal code invasion,
automated data collection at runtime, portable archive

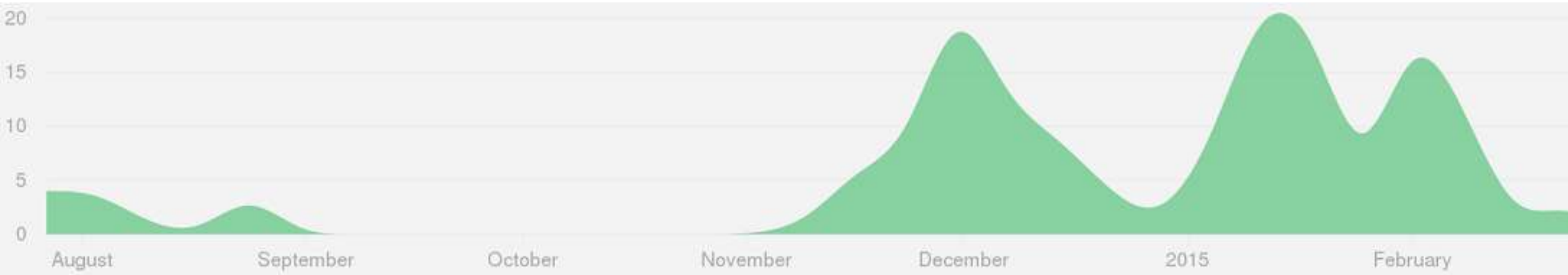
3 Granula Visualizer

Portable choke-point analysis for everyone!





Graphalytics = Modern Software Engineering Process



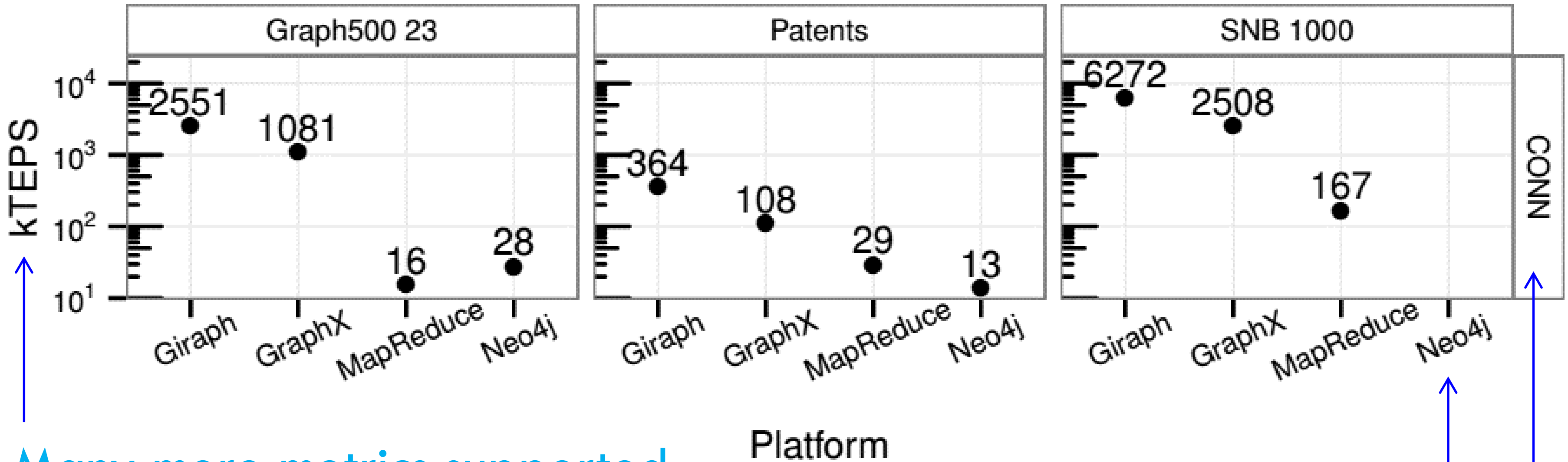
- Graphalytics code reviews
  - Internal release to LDBC partners (Feb 2015)
  - Public release, announced first through LDBC (Apr 2015)
  - First full benchmark specification, LDBC criteria (Q1 2016)
- Jenkins continuous integration server
- SonarQube software quality analyzer

https://github.com/tudelft-atlarge/graphalytics/

Graphalytics in Practice

6 real-world datasets + 2 synthetic generators

Data ingestion not included here!



Many more metrics supported

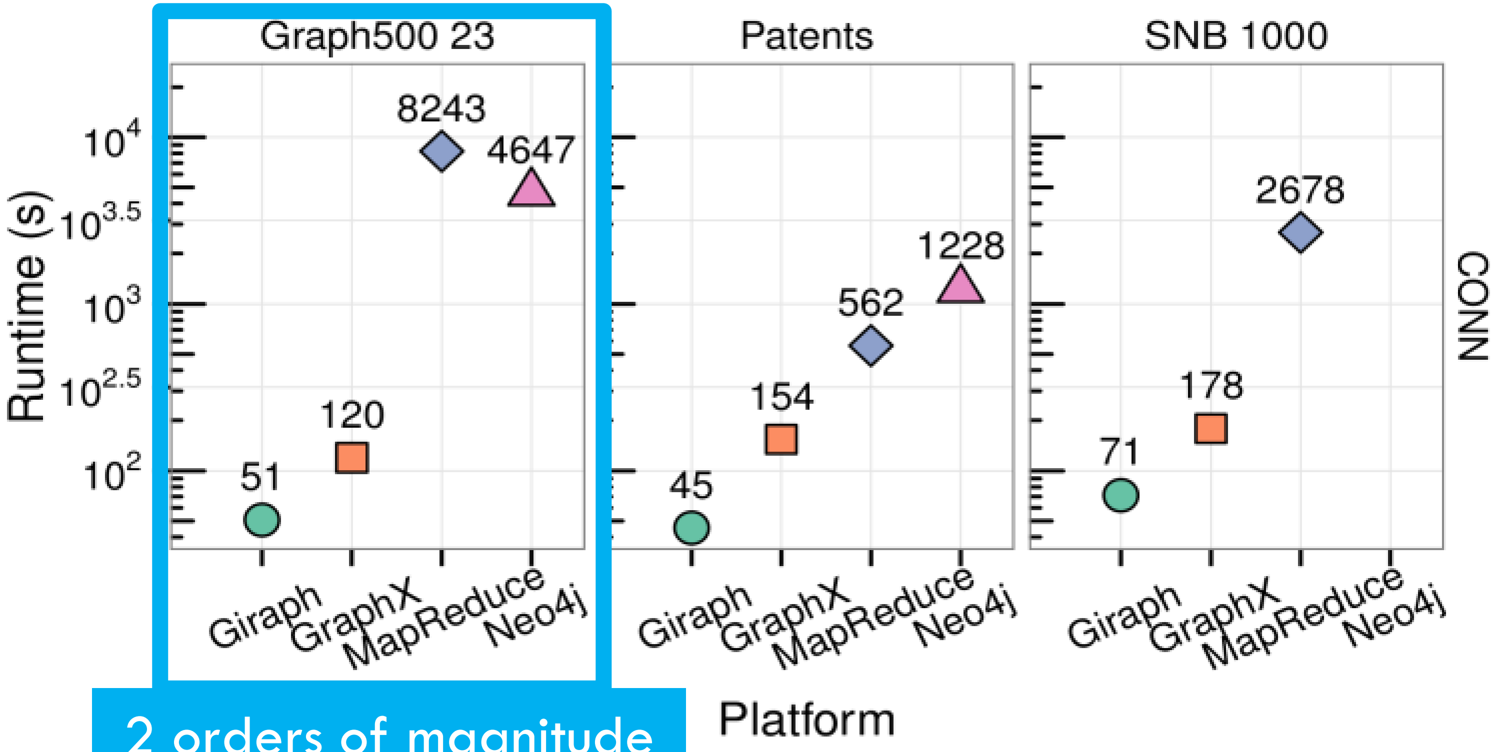
10 platforms tested w prototype implementation

6 classes of algorithms

- Missing results = failures of the respective systems

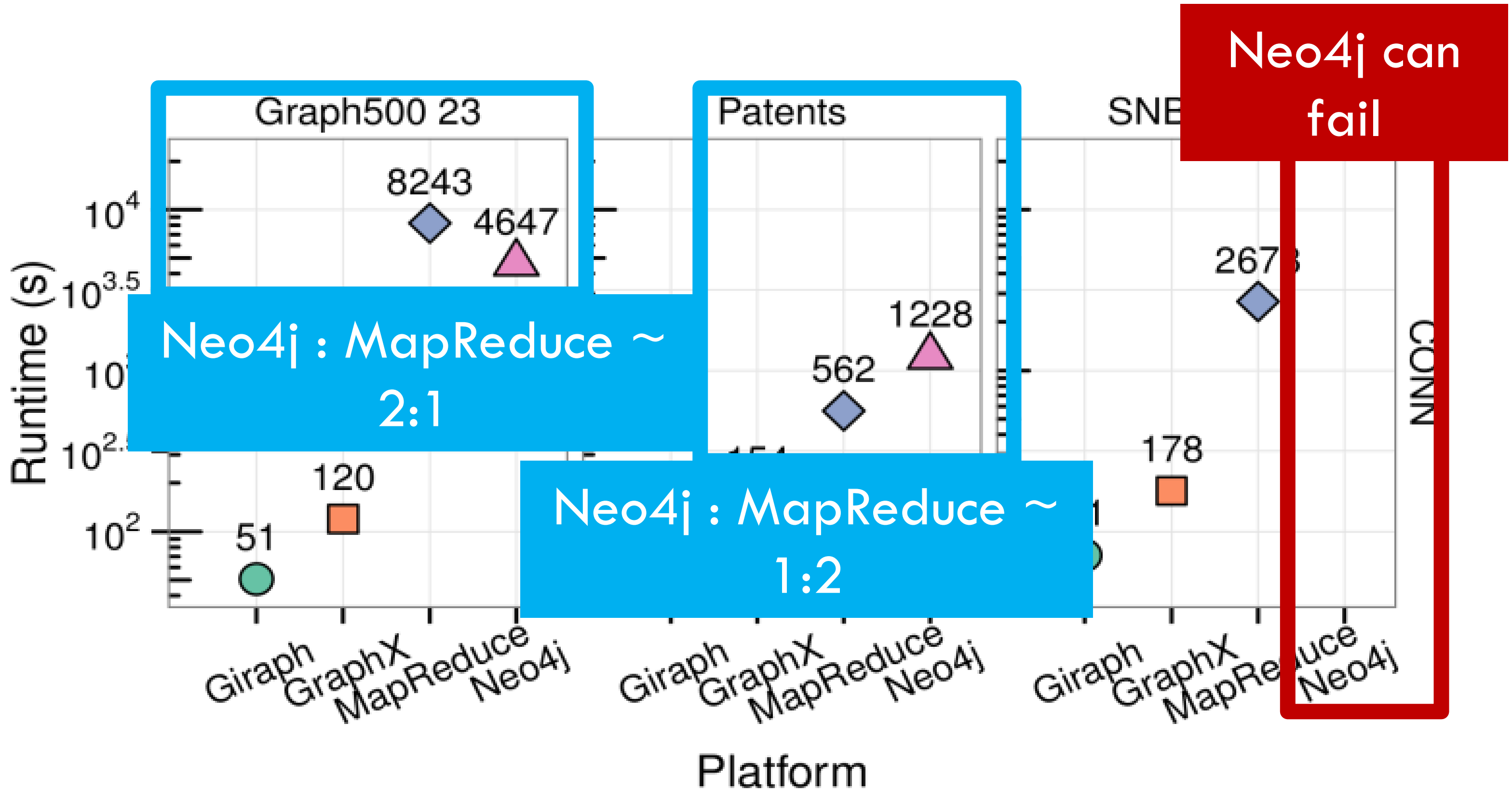
M. Capota et al., Graphalytics: A Big Data Benchmark for Graph-Processing Platforms. SIGMOD GRADES 2015

Runtime: the Platform has large impact

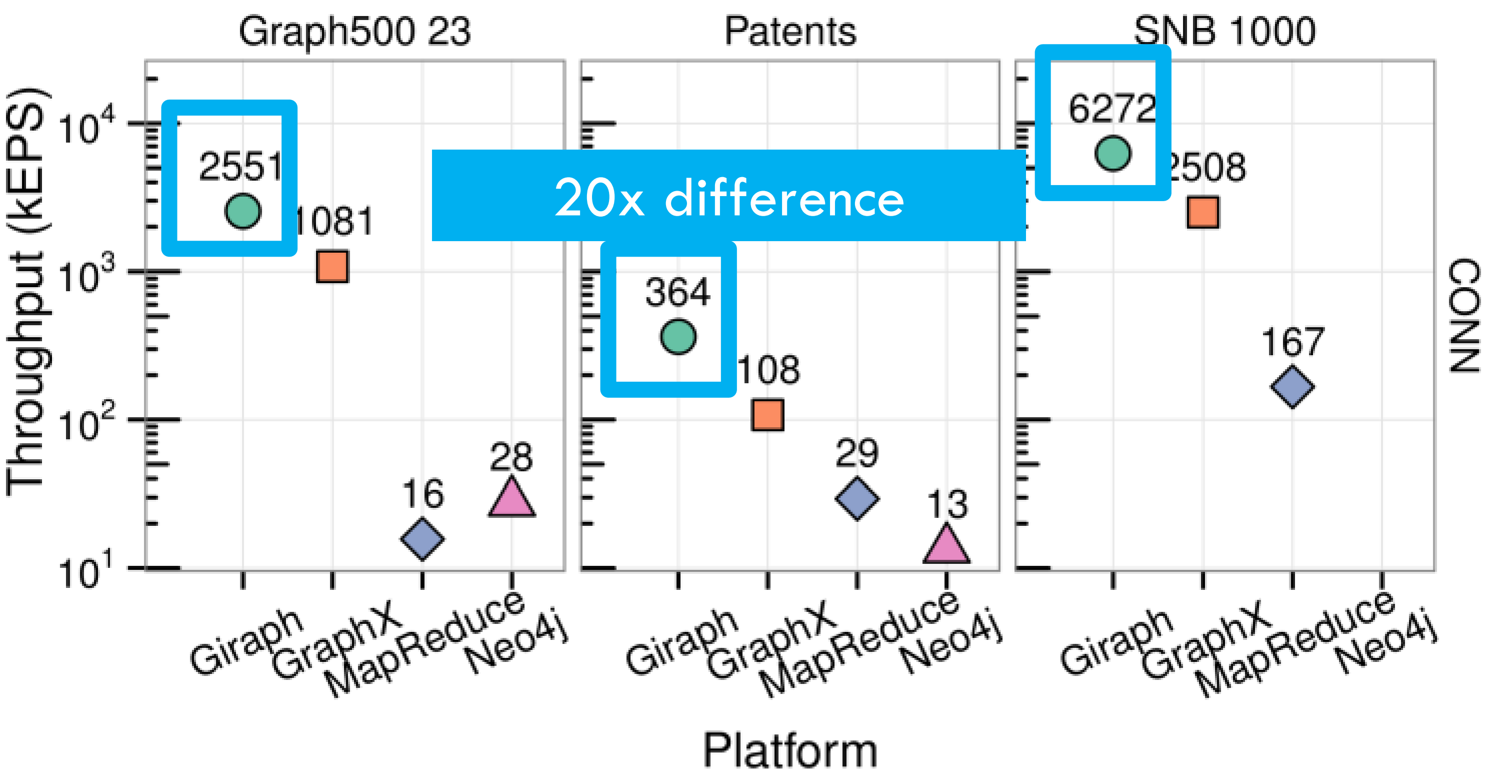


2 orders of magnitude difference due to platform

Runtime: the Dataset has large impact



Throughput: Dataset structure matters!



Graphalytics, in a nutshell

- An LDBC benchmark
- Advanced benchmarking harness
- Diverse real and synthetic datasets
- Many classes of algorithms
- Granula for manual choke-point analysis
- Modern software engineering practices
- Supports many platforms

LDBC The graph & RDF benchmark reference

DELFT DATA SCIENCE

intel

ORACLE

http://graphalytics.ewi.tudelft.nl
https://github.com/tudelft-atlarge/graphalytics/

Implementation status

G=validated, on GitHub
V=validation stage

	MapReduce2	Giraph	GraphX	GraphLab	Neo4j	PGX.D	GraphMat	TOTEM	MapGraph	Medusa
Stats	G	G	G	G	G	--	--	--	--	--
BFS	G	G	G	G	G	V	V	V	V	V
CON	G	G	G	G	G	V	--	V	V	V
CD	G	G	G	G	G	--	--	--	--	--
EVO	G	G	G	--	G	--	--	--	--	--
P'Rank	--	G	V	V	--	V	V	V	V	V

https://github.com/tudelft-atlarge/graphalytics/

Ongoing Work

- Final benchmark definition (Q1 2016)
  - *Data schema:* formalize schema, support stakeholders
  - *Workloads*: formalize datasets + algorithms
  - *Performance metrics:* done
  - *Execution rules:* select parameter values
- Online Live Performance Results (Q4 2016)
  - Live addition of results
  - Curation of added results
  - Auditing results

Questions?

Agenda

- Introduction to Linked Data
- LDBC Approach
- Graphalytics
  - Systems and models
  - Methodology for performance evaluation of graph-processing platforms
  - Graphalytics architecture
- The hour of benchmarking
  - Hands-on Graphalytics
    - Results analysis & lessons learned
  - Fine-grained in-depth analysis with Granula
- Summary & Panel/open discussion

The hour of benchmarking

DATAGEN in practice

Graphalytics in practice

Zooming in with Granula

Schedule

- How to use DATAGEN?
  - Generating graph structure vs rich graphs
- Benchmarking with Graphalytics
  - Comparing Giraph and PGX.D
- Fine-grained in-depth analysis with Granula
  - Performance modeling, archiving, and visualizing
  - In-depth performance evaluation

How to use DATAGEN?

- Step 1: generate structural graph
  - Generate *graphalytics-1* using DATAGEN
  - Inspect resulting graph
- Step 2: generate rich graph
  - Generate *snb-1* using DATAGEN
  - Compare (meta)data included in *snb-1* with *graphalytics-1*

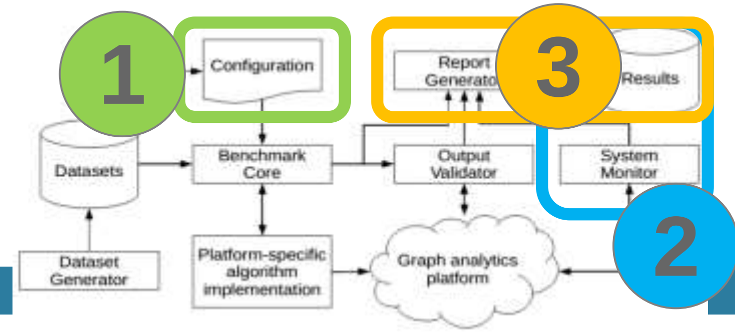
10 minutes – See Handout Section 3

Graphalytics: Benchmarking Giraph and PGX.D

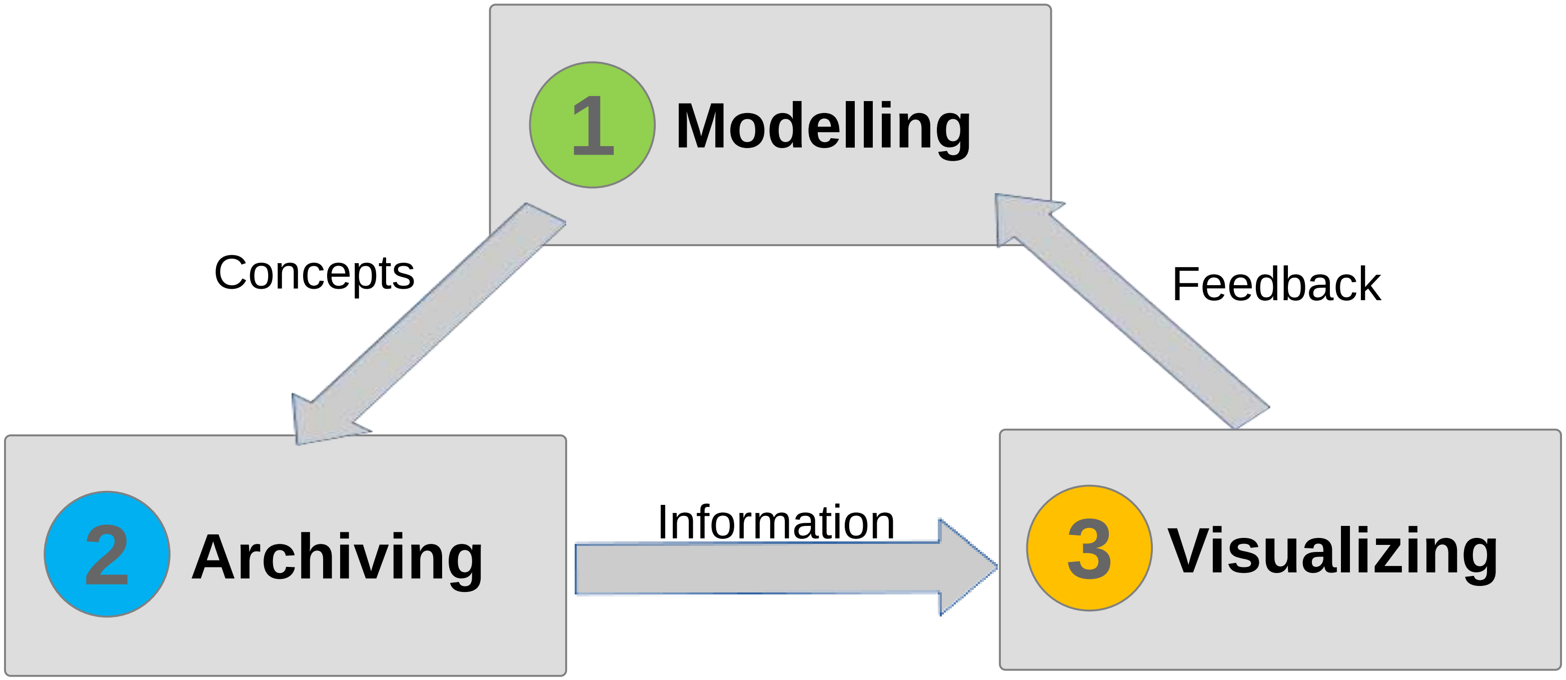
- Step 1: benchmark Giraph
  - Prepare platform
  - Launch pre-configured Graphalytics for Giraph
- Step 2: benchmark PGX.D
  - Launch pre-configured Graphalytics for PGX.D
- Step 3: compare results
  - What can be learned?

20 minutes – See Handout Section 4

Granula Overview



Framework for fine-grained performance evaluation of Big Data Processing (BDP) systems



https://github.com/tudelft-atlarge/granula/

Granula Demo

Live analysis of two benchmark reports for a 5- and 20-node Giraph cluster.

Feel free to follow along on your device, or explore on your own!

Agenda

- Introduction to Linked Data
- LDBC Approach
- Graphalytics
  - Systems and models
  - Methodology for performance evaluation of graph-processing platforms
  - Graphalytics architecture
- The hour of benchmarking
  - Hands-on Graphalytics
    - Results analysis & lessons learned
  - Fine-grained in-depth analysis with Granula
- Summary & Panel/open discussion



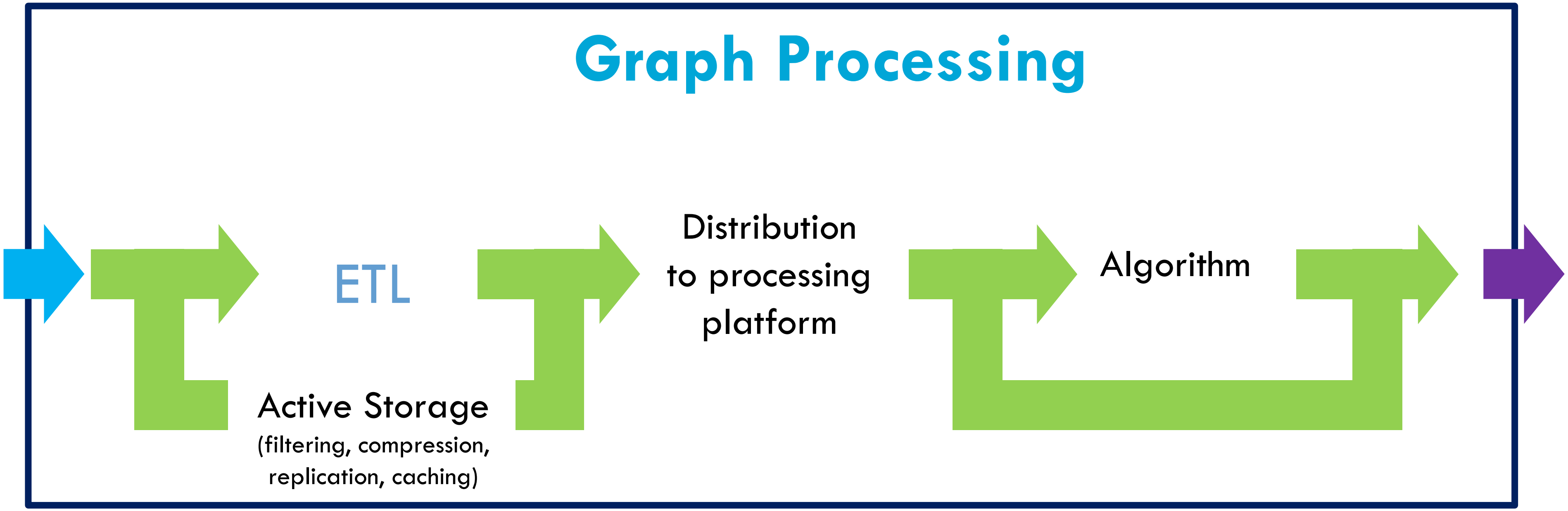
Open discussion


What does a benchmark consist of?

- Four main elements:
  - *data schema*: defines the structure of the data
  - *workloads*: defines the set of operations to perform
  - *performance metrics*: used to measure (quantitatively) the performance of the systems
  - *execution rules*: defined to assure that the results from different executions of the benchmark are valid and comparable
- Software as Open Source (GitHub)
  - data generator, query drivers, validation tools, …

Discussion 1 (execution rules, metrics)

- How much preprocessing should we allow in the ETL phase?
- How to choose a metric that captures the preprocessing?



Discussion 2

- Trade-off between fast dataset submission (reads from the database or full-scale generation) and cost (of storage, of computation).

Discussion 3

- Should we allow platform-specific algorithms or only implementations of exhaustively defined algorithms?

Discussion 4

- How should we asses the correctness of algorithms that produce approximate results?

Discussion 5

- How to setup the platforms? Should we allow algorithm-specific platform setups or should we require only one setup to be used for all algorithms?

Take home message

Summary

- Graph processing is a hot topic for both software and hardware developers
- Challenges in scale and irregularity
- Existing platforms: over 80!
- Choose which one to use
  - Quick: pick a platform where your graph fits and that you can program.
  - **Graphalytics: use systematic benchmarking!**

Find us online: graphalytics.ewi.tudelft.nl
https://github.com/tudelft-atlarge/graphalytics/

Bibliography

- Graphalytics
  - Capota, M., Hegeman, T., Iosup, A., Prat-Pérez, A., Erling, O., & Boncz, P. (2015). Graphalytics: A Big Data Benchmark for Graph-Processing Platforms, GRADES 2015.
  - A. Iosup, A. L. Varbanescu, M. Capota, T. Hegeman, Y. Guo, W.-L. Ngai, M. Verstraaten. Towards Benchmarking IaaS and PaaS Clouds for Graph Analytics, In the WBDB 2014
- LDBC and in particular DATAGEN
  - Erling, Orri, et al. "The LDBC Social Network Benchmark: Interactive Workload." Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data. ACM, 2015.
  - http://ldbcouncil.org/sites/default/files/LDBC_D2.2.2.pdf
  - http://ldbcouncil.org/sites/default/files/LDBC_D3.3.34.pdf
- Performance evaluation of graph-processing systems
  - Y. Guo, A. L. Varbanescu, A. Iosup, C. Martella, T. L. Willke: Benchmarking graph-processing platforms: a vision. ICPE 2014: 289-292
  - Guo et al., An Empirical Performance Evaluation of GPU-Enabled Graph-Processing Systems. CCGRID'15.
  - A. L. Varbanescu, M. Verstraaten, C. de Laat, A. Penders, A. Iosup, H. J. Sips: Can Portability Improve Performance?: An Empirical Study of Parallel Graph Analytics. ICPE 2015: 277-287