BENCHMARKING PLATFORMS FOR LARGE-SCALE GRAPH PROCESSING AND RDF DATA MANAGEMENT

The LDBC Approach

Your hosts today

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

Agenda

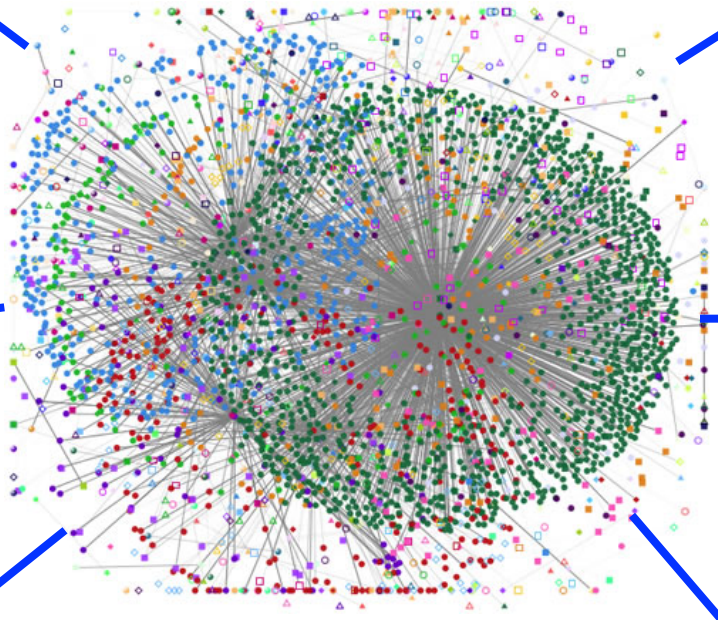
- Introduction to Linked Data
- LDBC Social Network Benchmark (SNB)
- Graphalytics
  - Systems and models
  - Methodology for performance evaluation of graph-processing platforms
  - Graphalytics architecture
- The hour of benchmarking
  - Hands-on Graphalytics
    - Results analysis & lessons learned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

Linked data

Size matters

The need for benchmarking

The data deluge: large-scale graphs



LinkedIn

300M users

??? edges

YAHOO!

friendster

270M MAU
200+ avg followers

>54B edges

1.2B MAU 0.8B DAU
200+ avg followers

>240B edges

XFIRE

The data deluge: large-scale graphs

LinkedIn

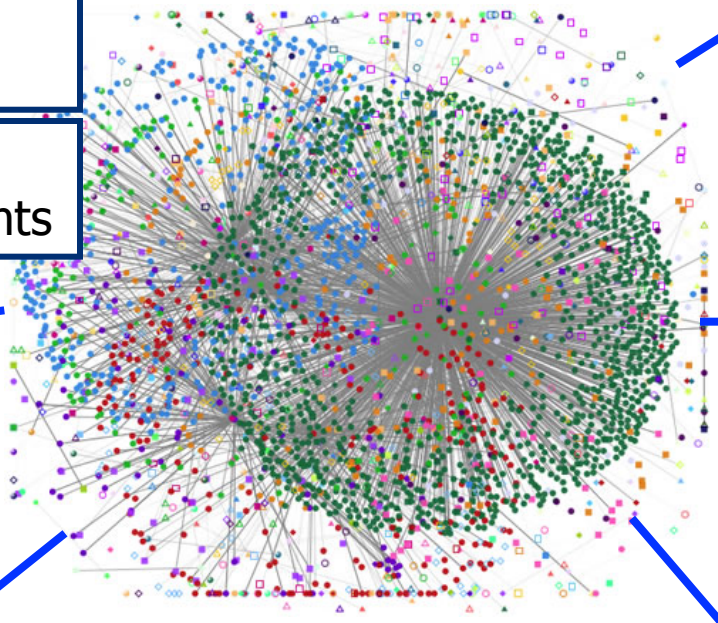
ORACLE

Oracle 1.2M followers,
132k employees

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

YAHOO!

friendster



270M MAU
200+ avg followers

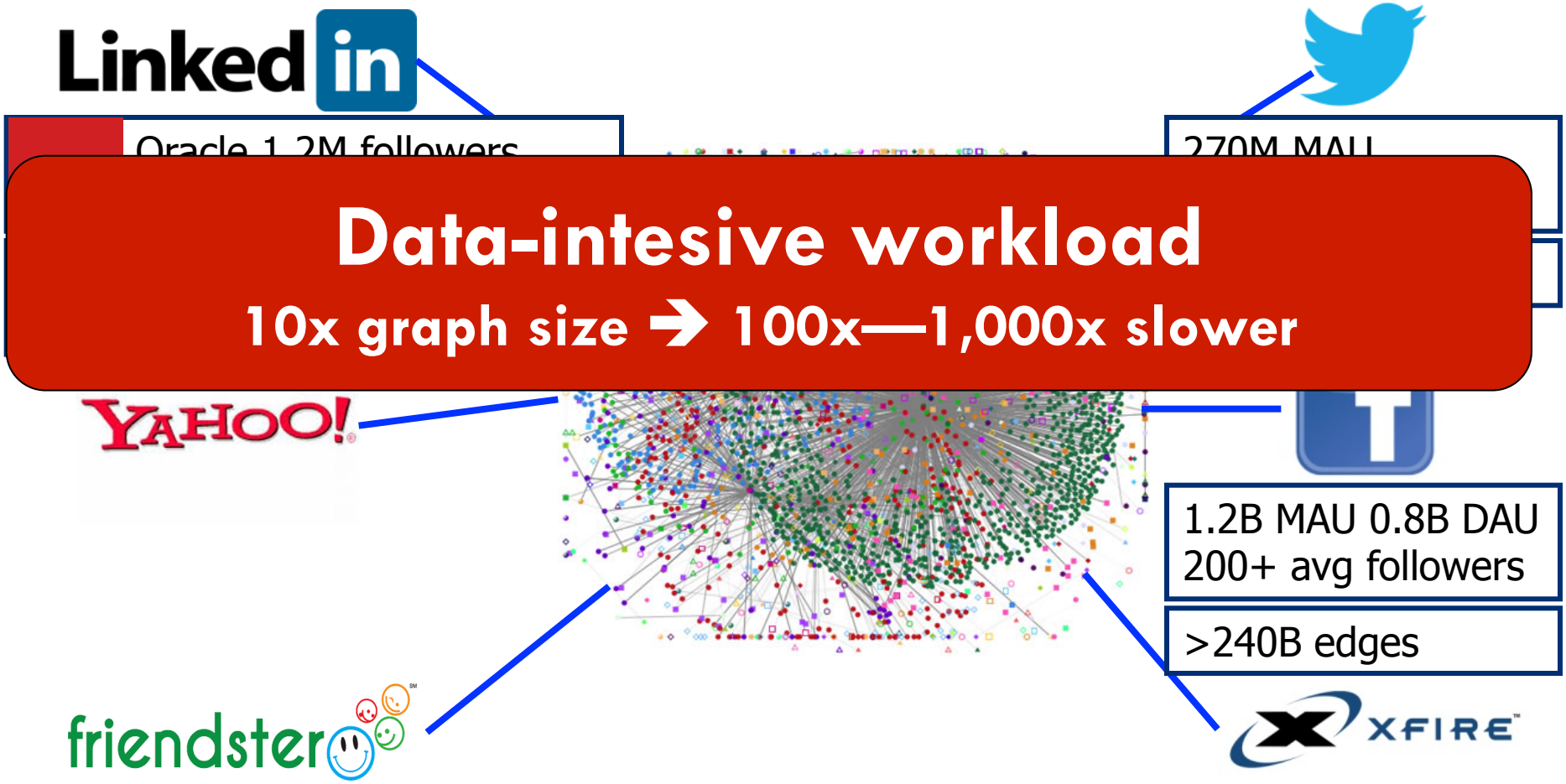
>54B edges

1.2B MAU 0.8B DAU
200+ avg followers

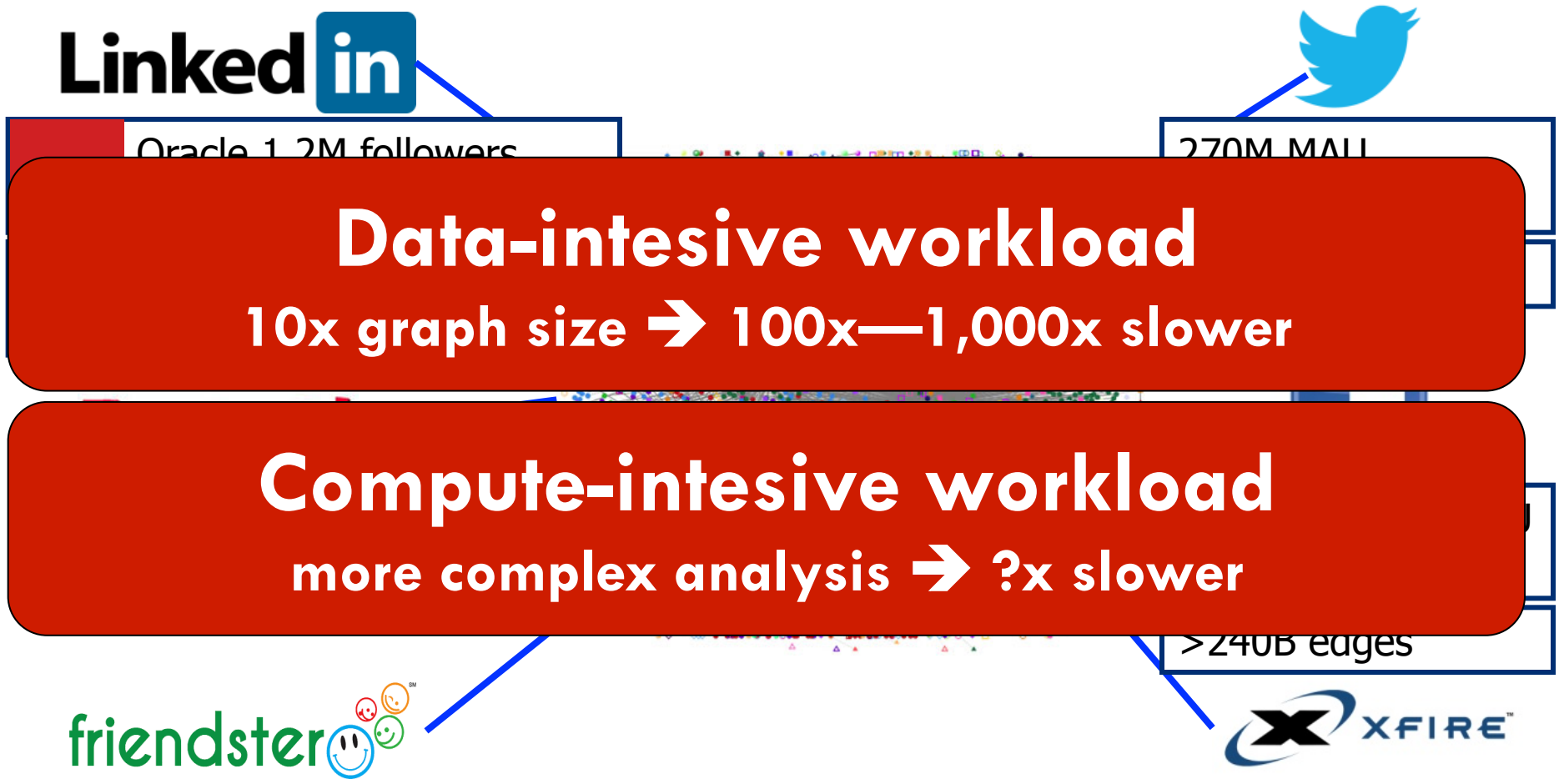
>240B edges

XFIRE

The data deluge: large-scale graphs



The data deluge: large-scale graphs



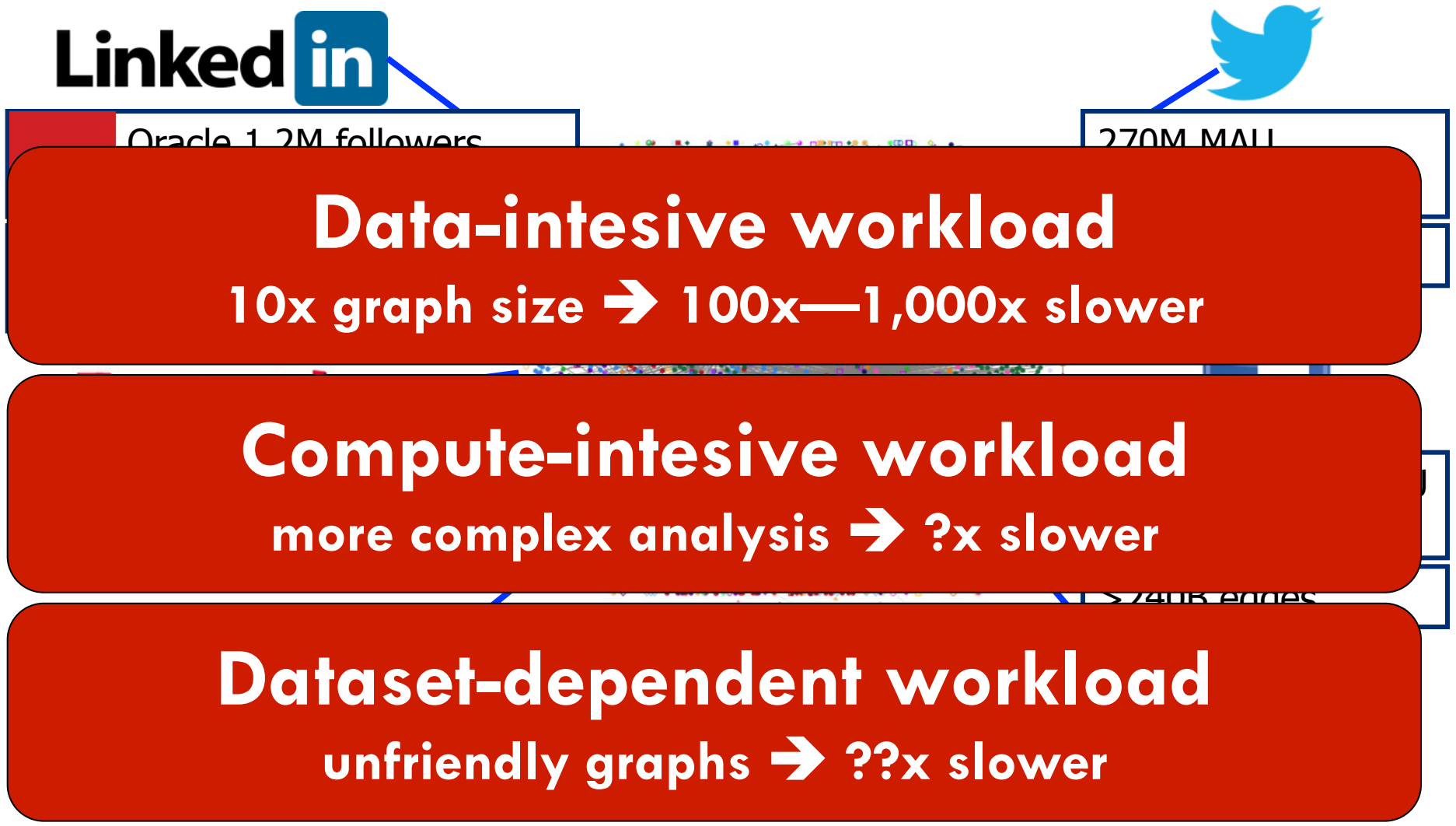
Data-intesive workload

10x graph size → 100x—1,000x slower

Compute-intesive workload

more complex analysis → ?x slower

The data deluge: large-scale graphs



Graphs at the Core of Our Society: The LinkedIn Example

100M Mar 2011, 69M May 2010

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/

Graphs at the Core of Our Society: The LinkedIn Example

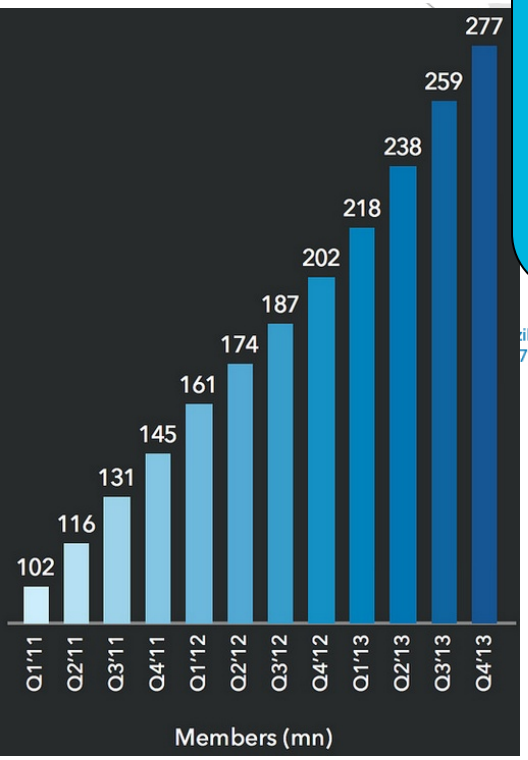
Sources: Vincenzo Cosenza, The State of LinkedIn, http://vincos.it/the-state-of-linkedin/
via Christopher Penn, http://www.shiftcomm.com/2014/02/state-linkedin-social-media-dark-horse/

Graphs at the Core of Our Society: The LinkedIn Example

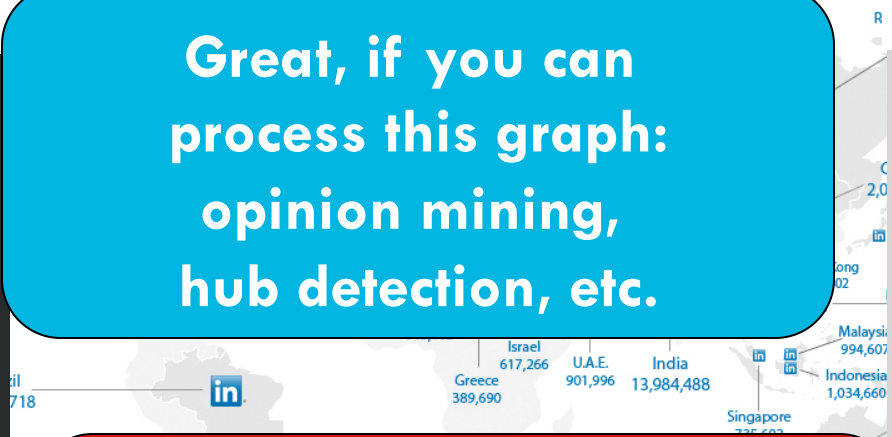
The State of **LinkedIn**

3-4 new users every second

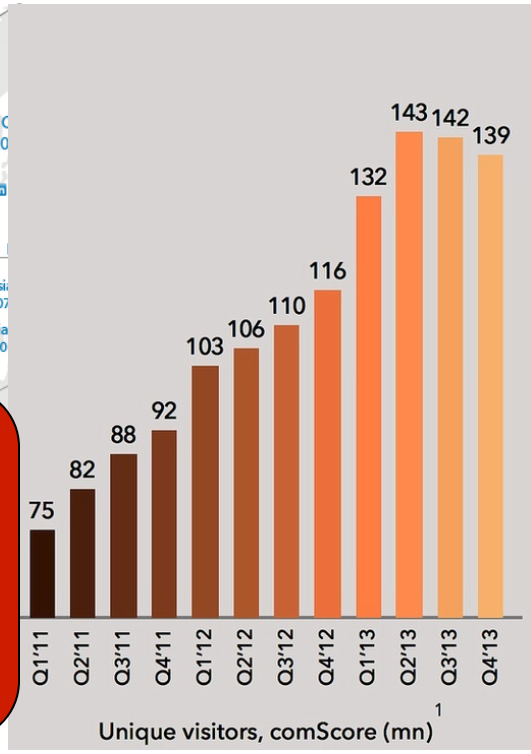
but fewer visitors (and page views)



Great, if you can process this graph: opinion mining, hub detection, etc.



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

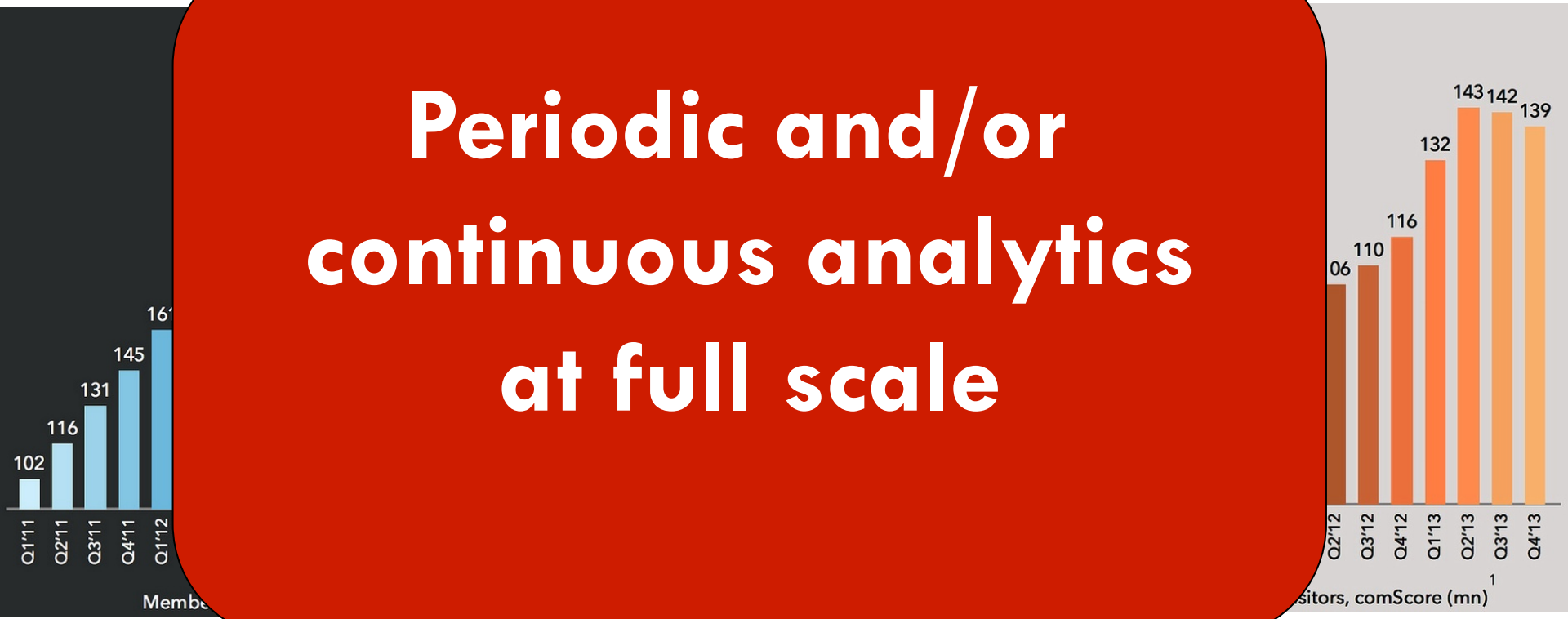


Sources: Vincenzo Cosenza, The State of LinkedIn, http://vincos.it/the-state-of-linkedin/
via Christopher Penn, http://www.shiftcomm.com/2014/02/state-linkedin-social-media-dark-horse/

Graphs at the Core of Our Society: The LinkedIn Example

3-4 new users every second

but fewer visitors (and page views)



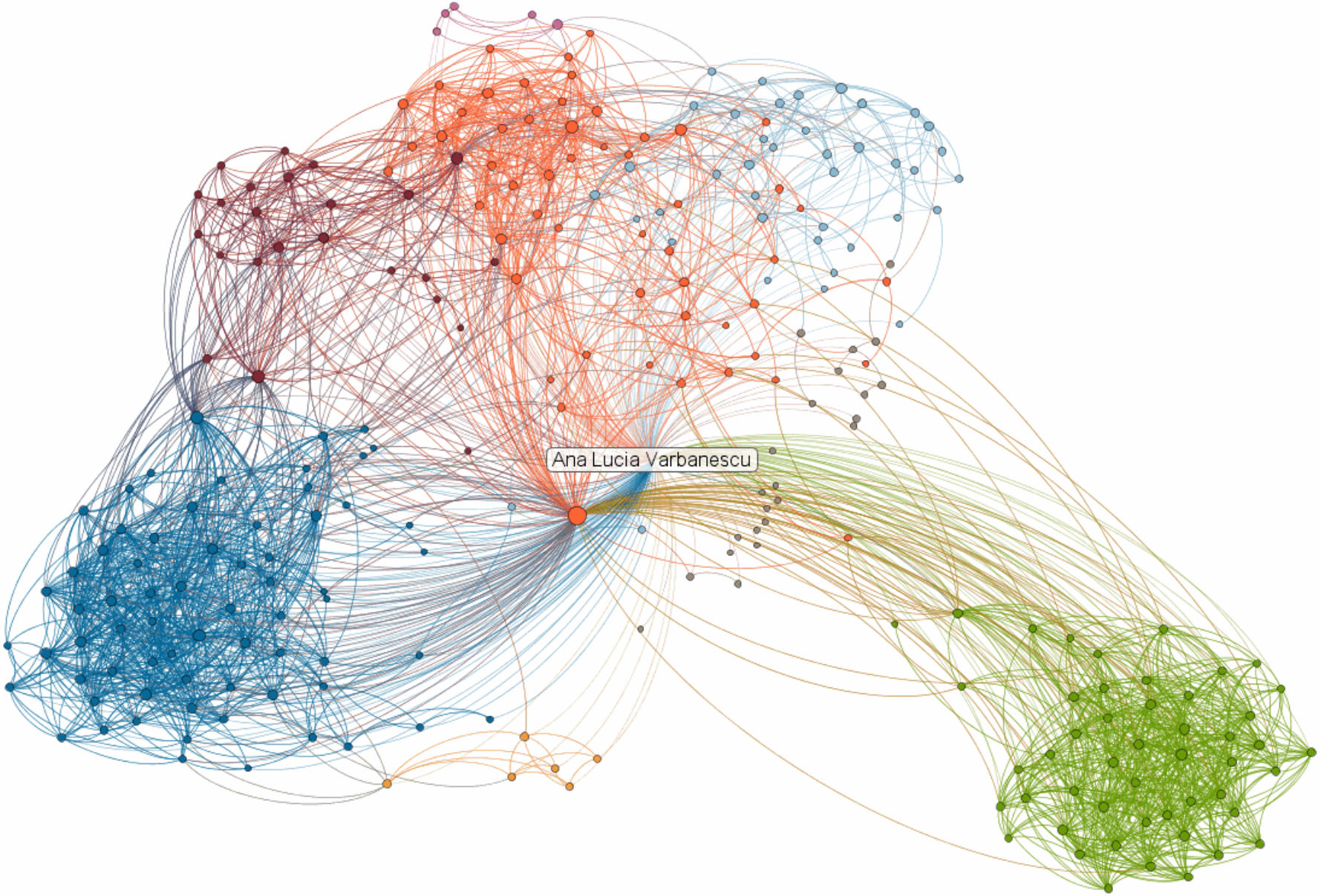
Periodic and/or continuous analytics at full scale

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/

LinkedIn Maps

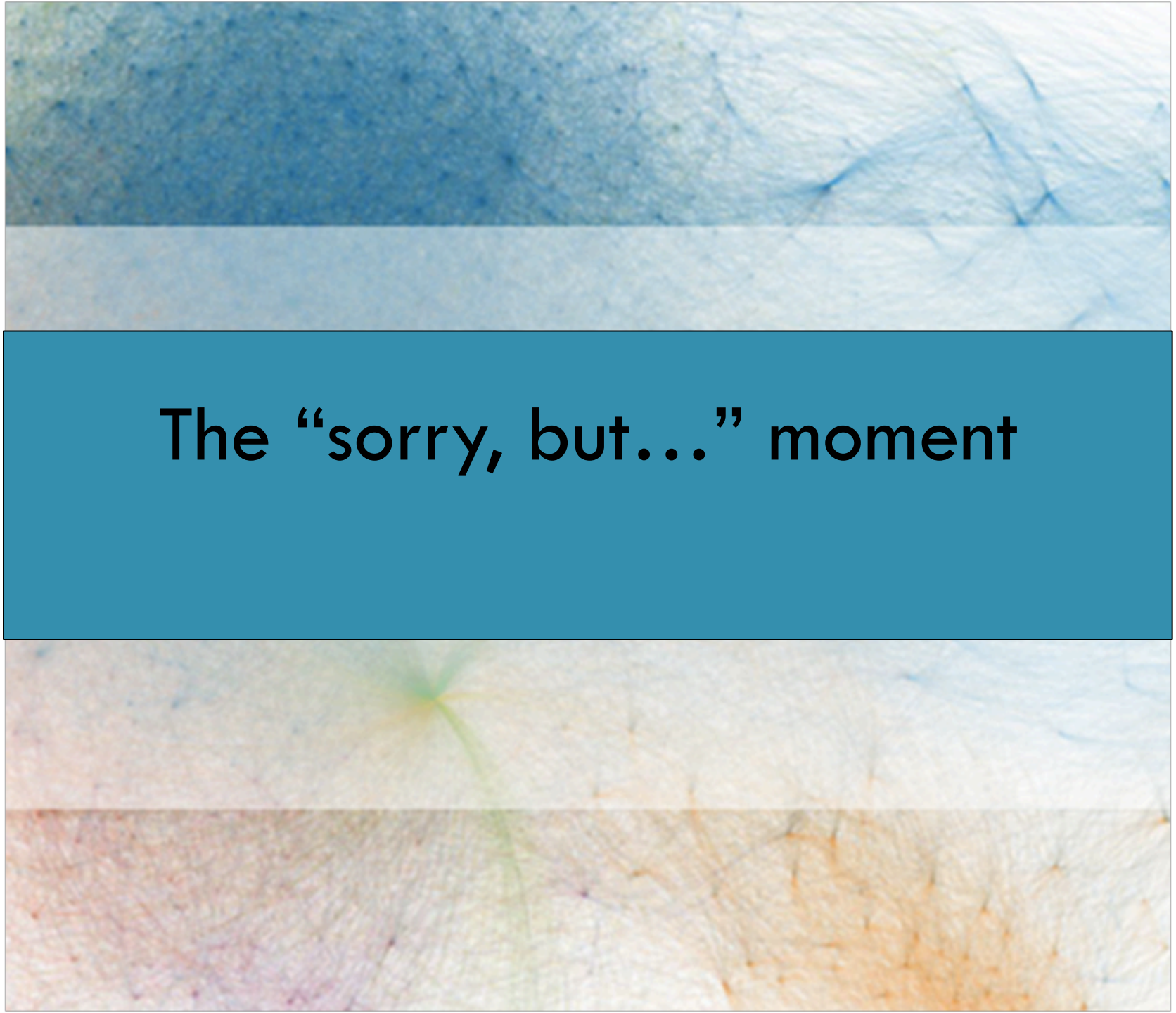
Ana Lucia Varbanescu's Professional Network

as of November 28, 2013

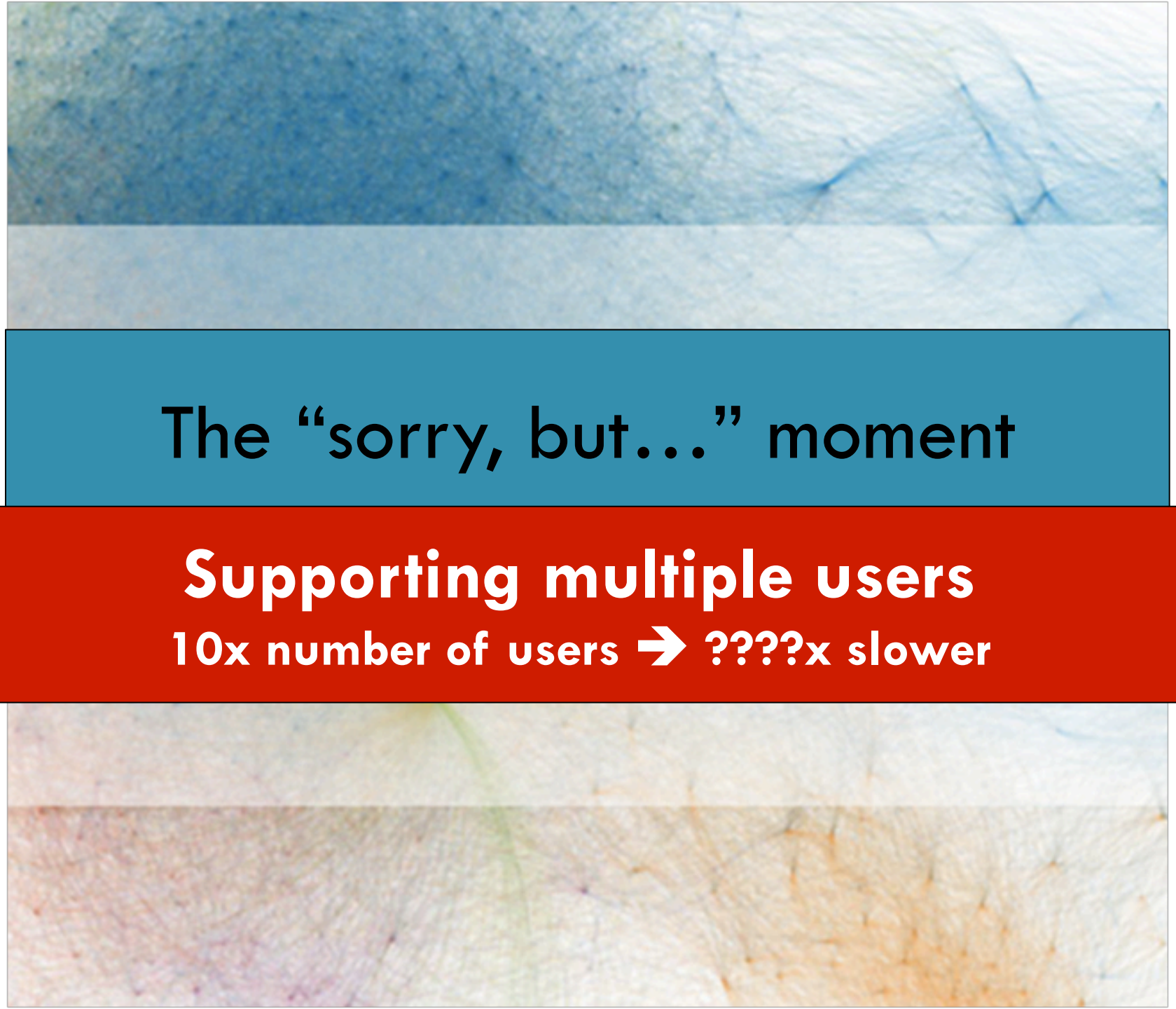


Your network is so large...

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

In this talk …

- Graphalytics = Graph analytics
- Analytics = any form of graph processing
- Platform = hardware and/or software we can tune and change as a whole
- (Graph) Processing system = computing system that includes one or more platforms (for graph processing)

Agenda

- Introduction to Linked Data
- LDBC Social Network Benchmark (SNB)
- 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

LDBC Social Network Benchmark (SNB)

Synthetic graph generation

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

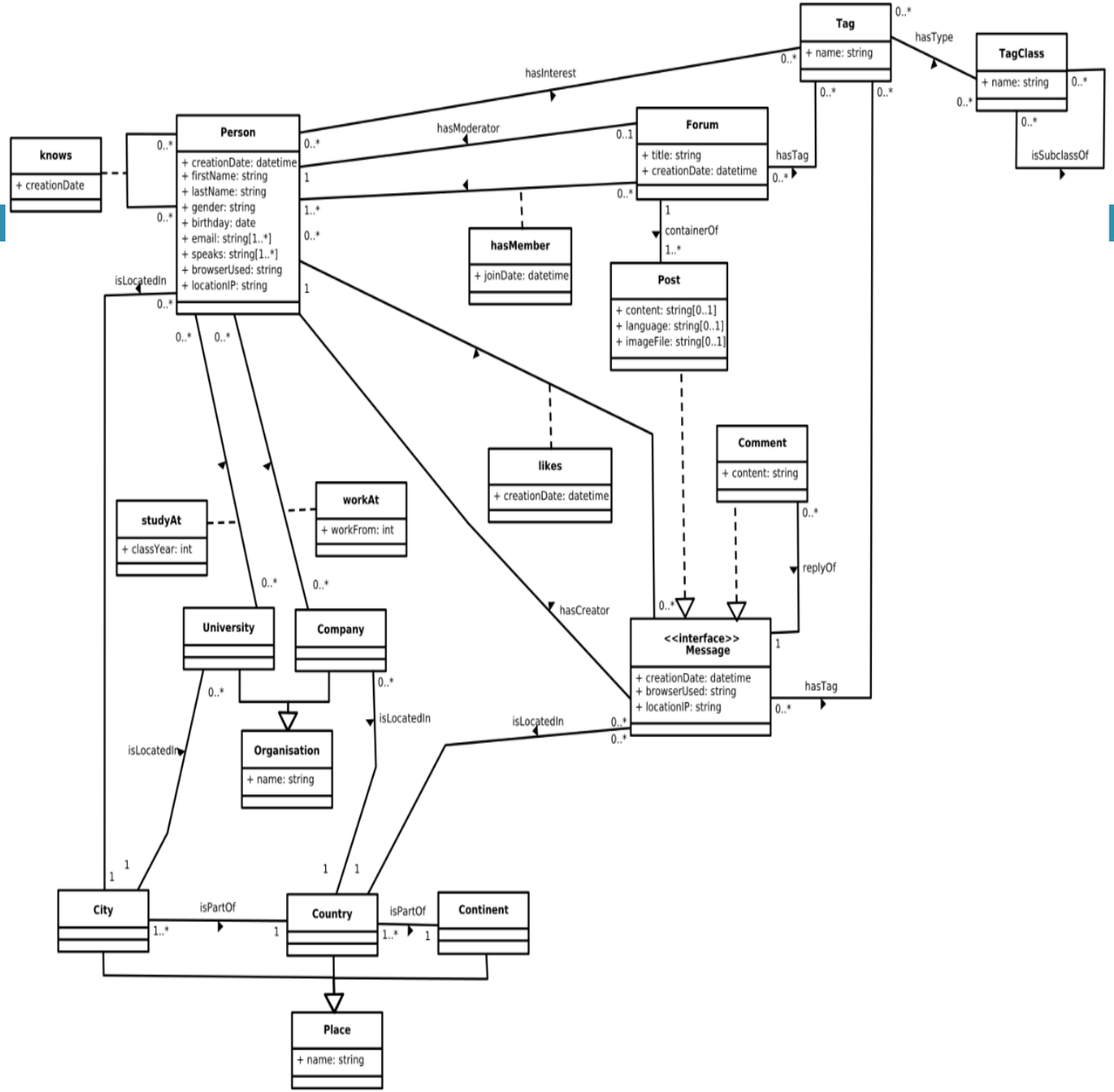
LDBC DATAGEN

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

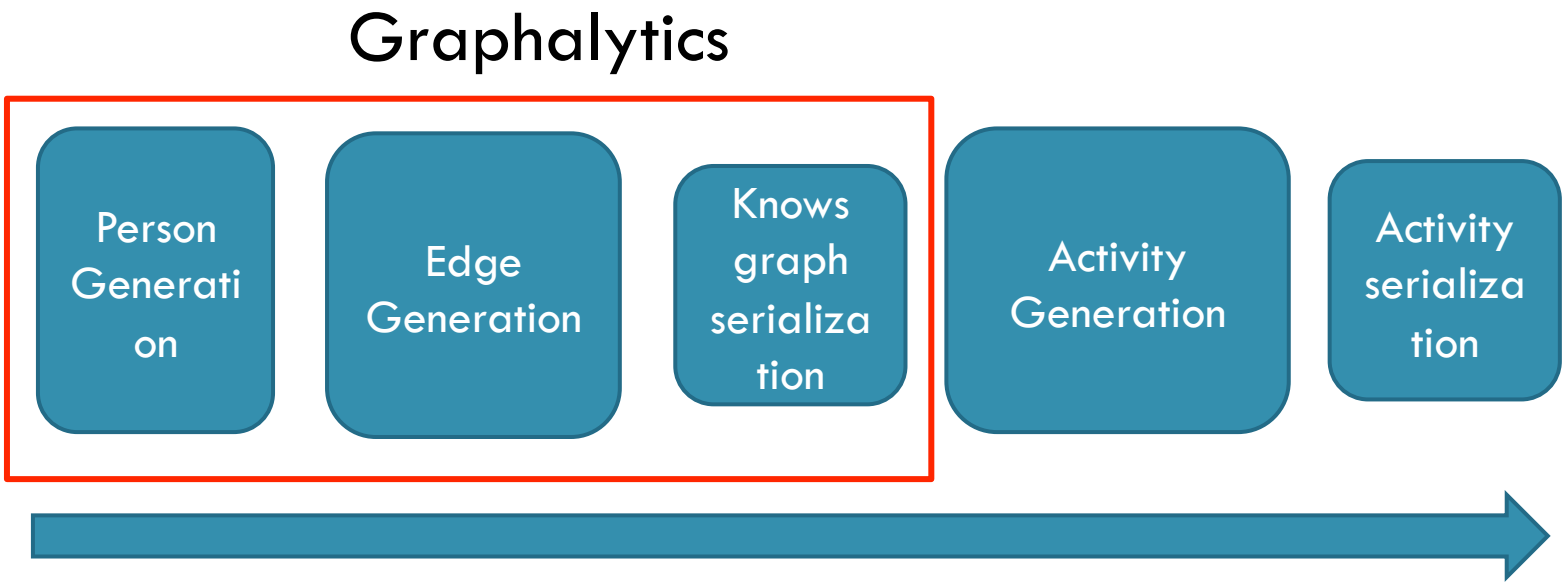
- Deterministic

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

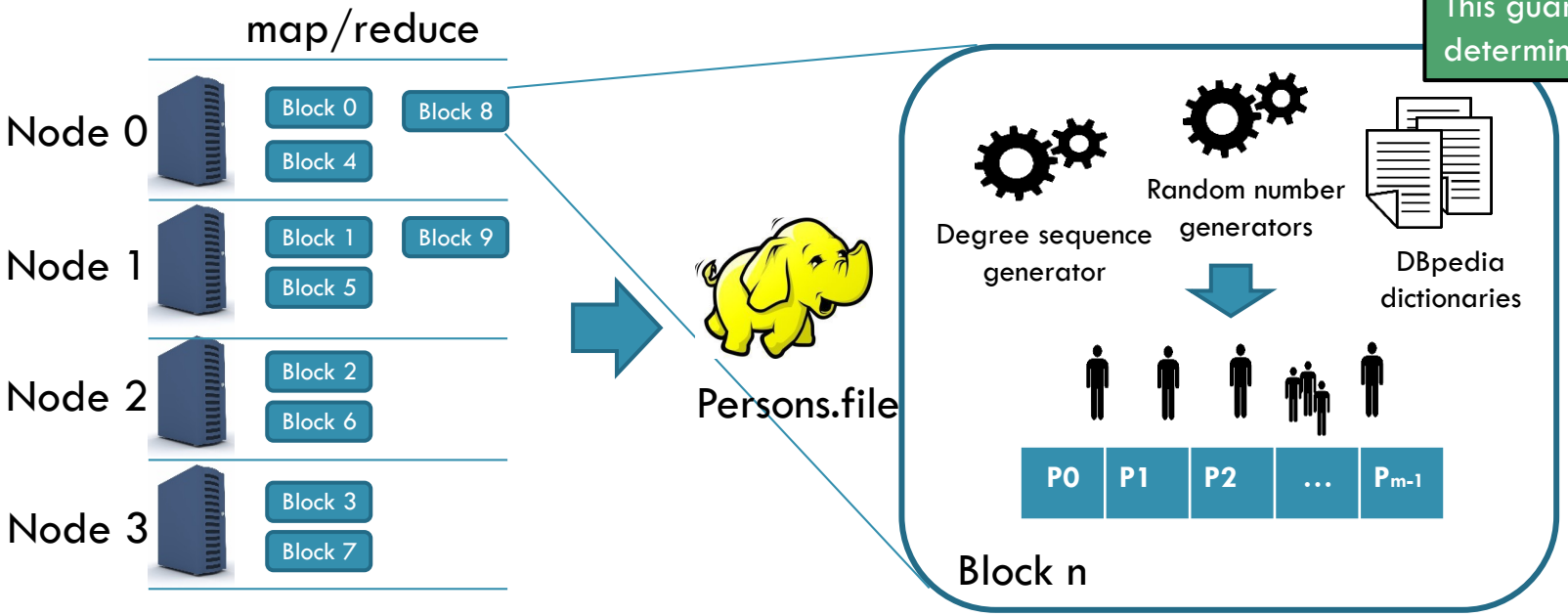

Tag
+ name: string
hasType
TagClass
+ name: string
isSubclassOf
hasInterest
hasModerator
Person
+ creationDate: datetime
+ firstName: string
+ lastName: string
+ gender: string
+ birthday: date
+ email: string[1..*]
+ speaks: string[1..*]
+ browserUsed: string
+ locationIP: string
knows
+ creationDate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

Generation Process

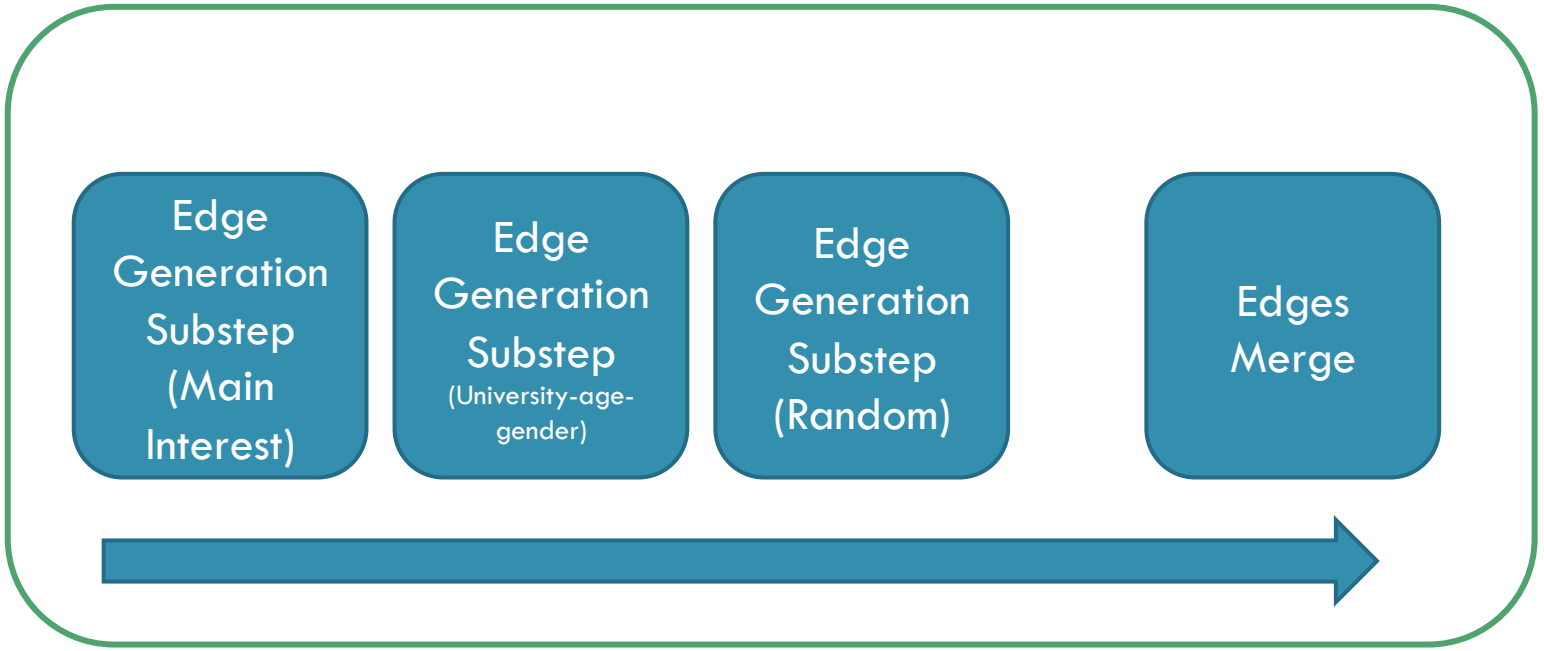


Person Generation

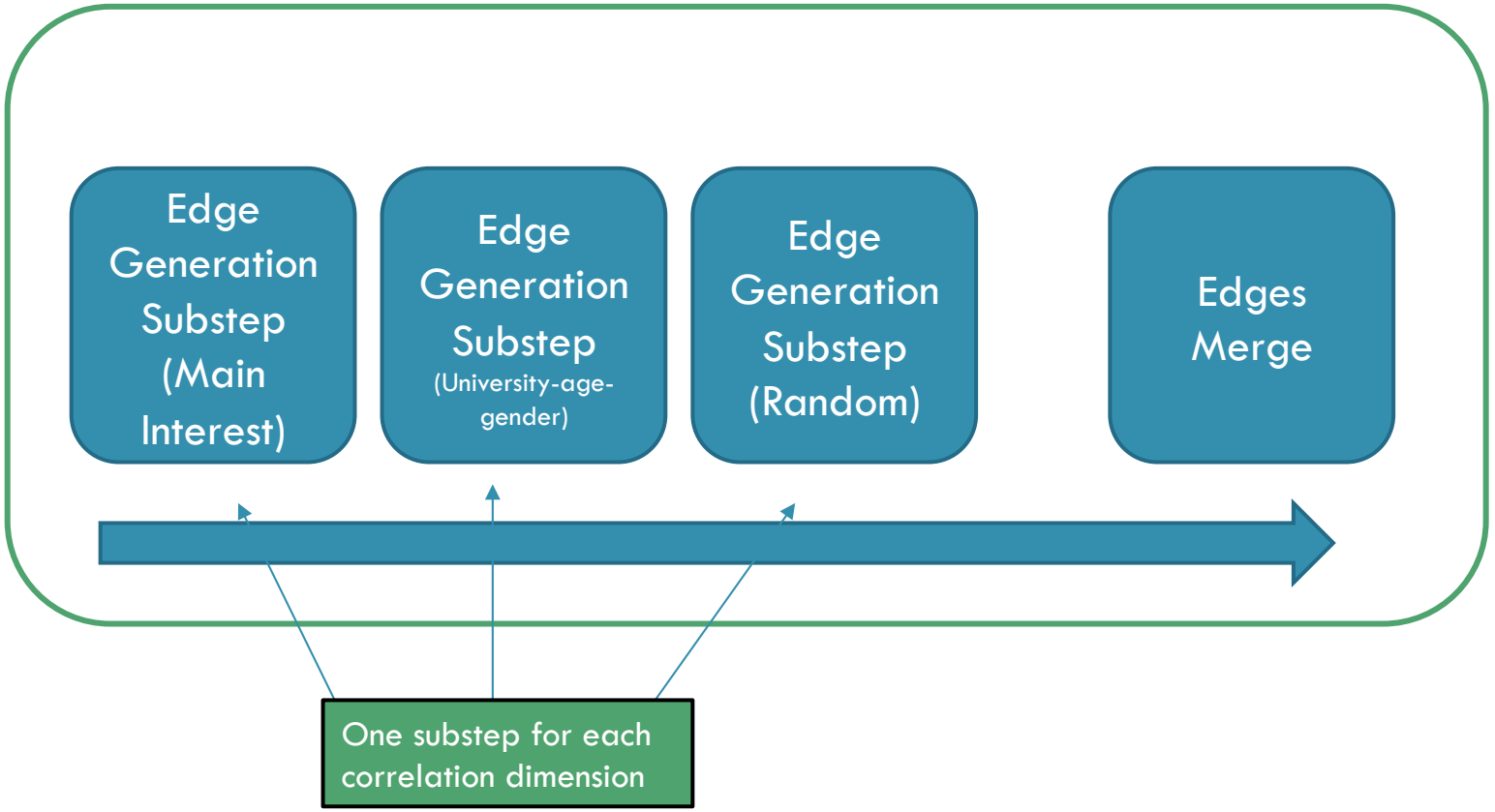
- A 4-machine cluster
- 100,000 Person network
- Block size m= 10,000 => 10 blocks in total



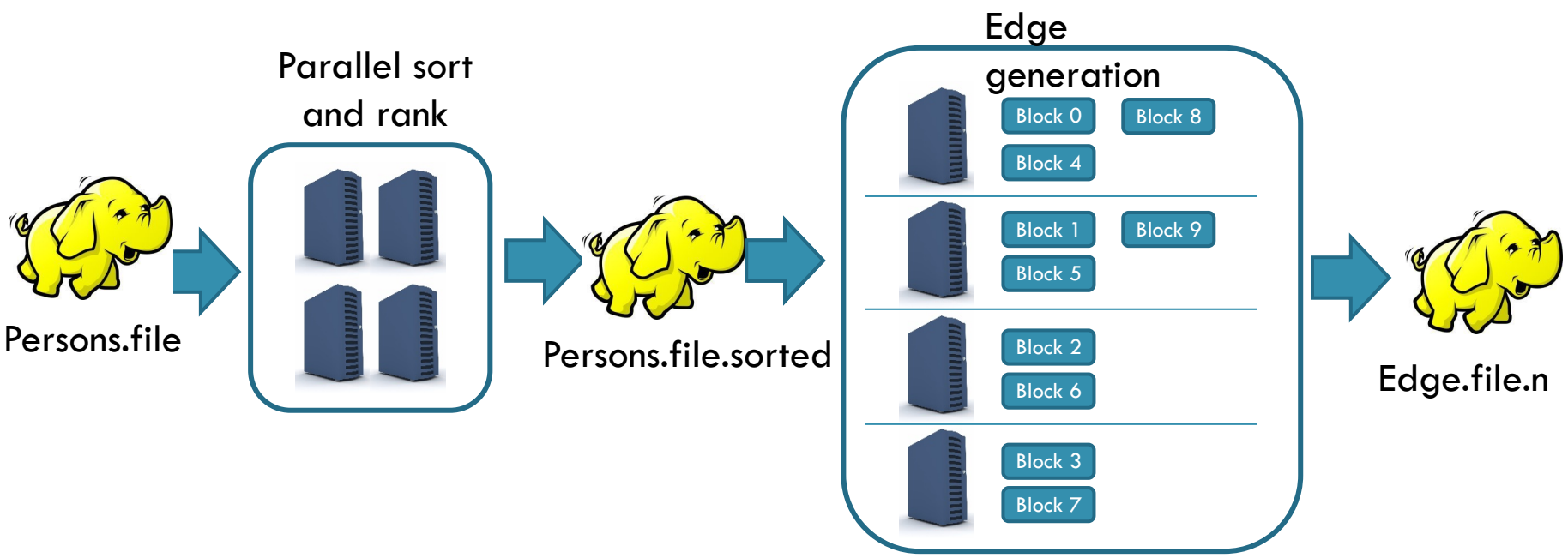
Edge Generation



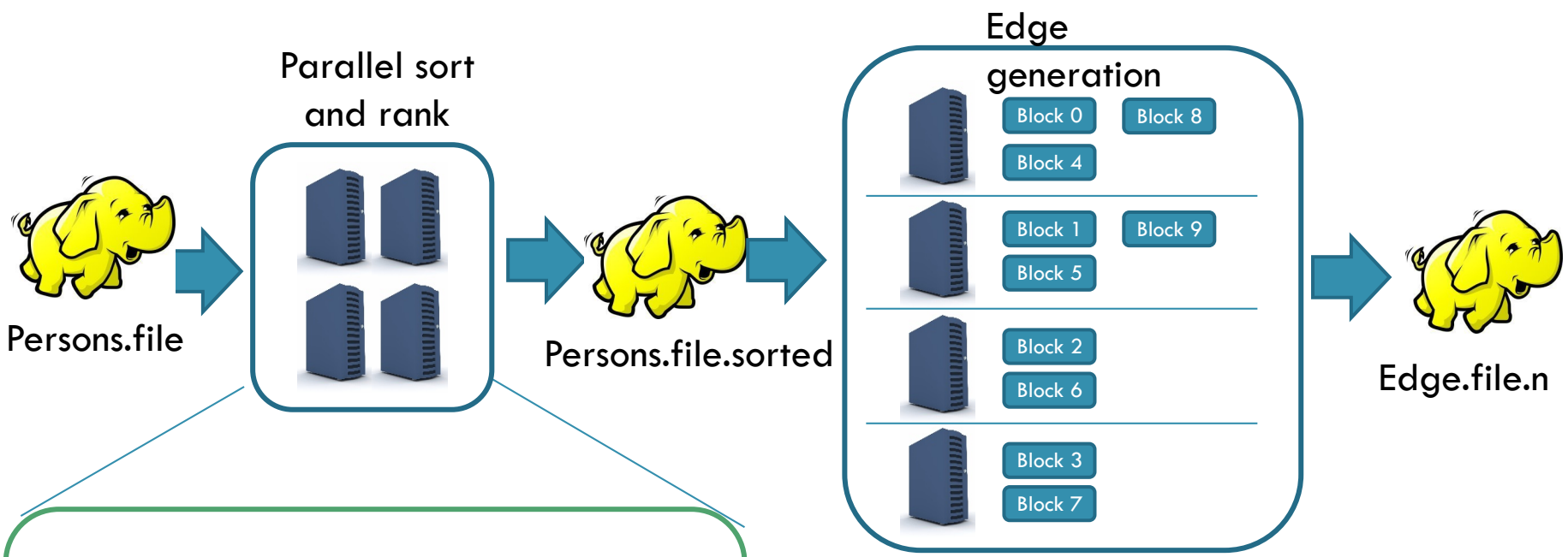
Edge Generation



Edge Generation Substep

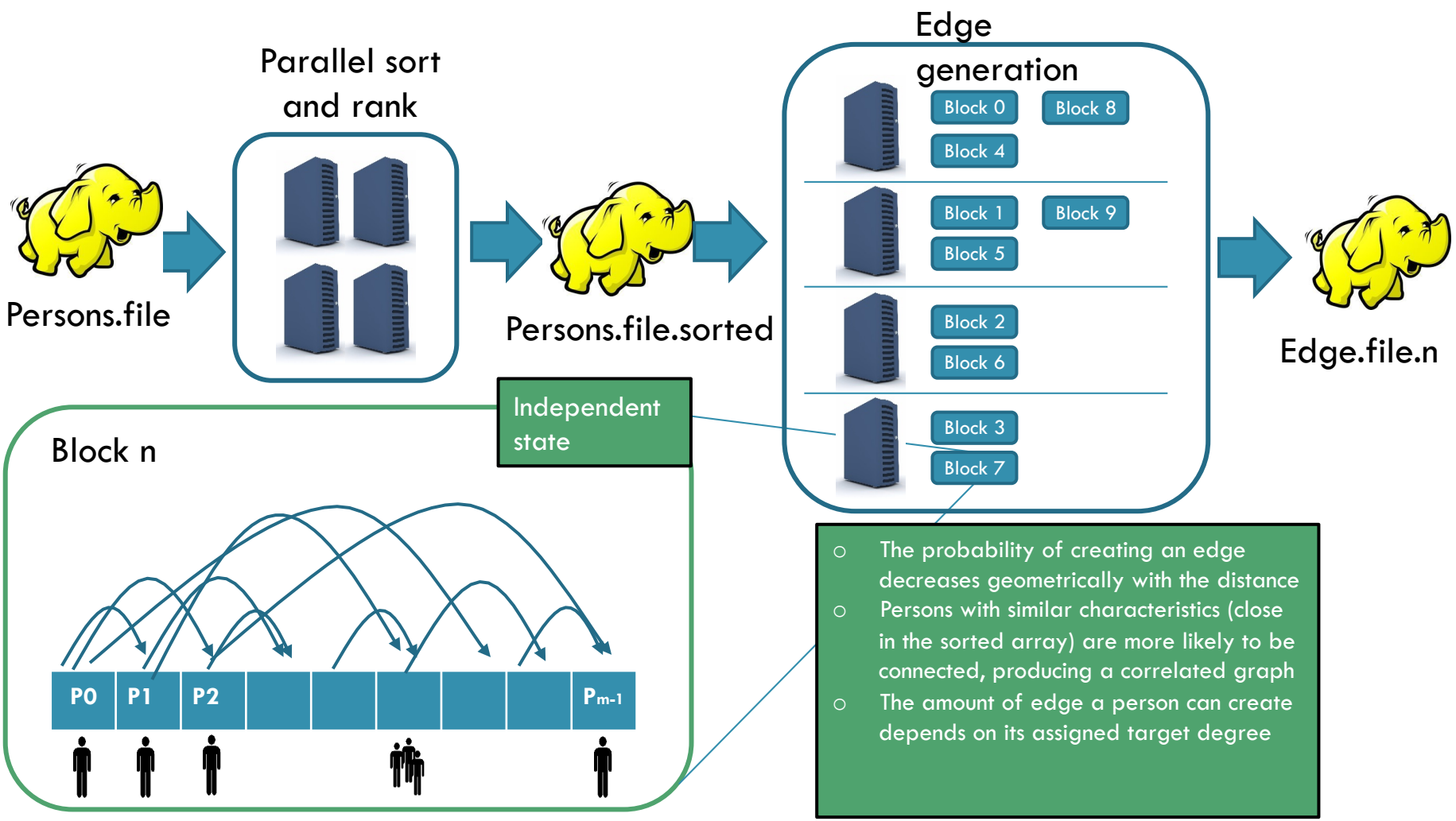


Edge Generation Substep

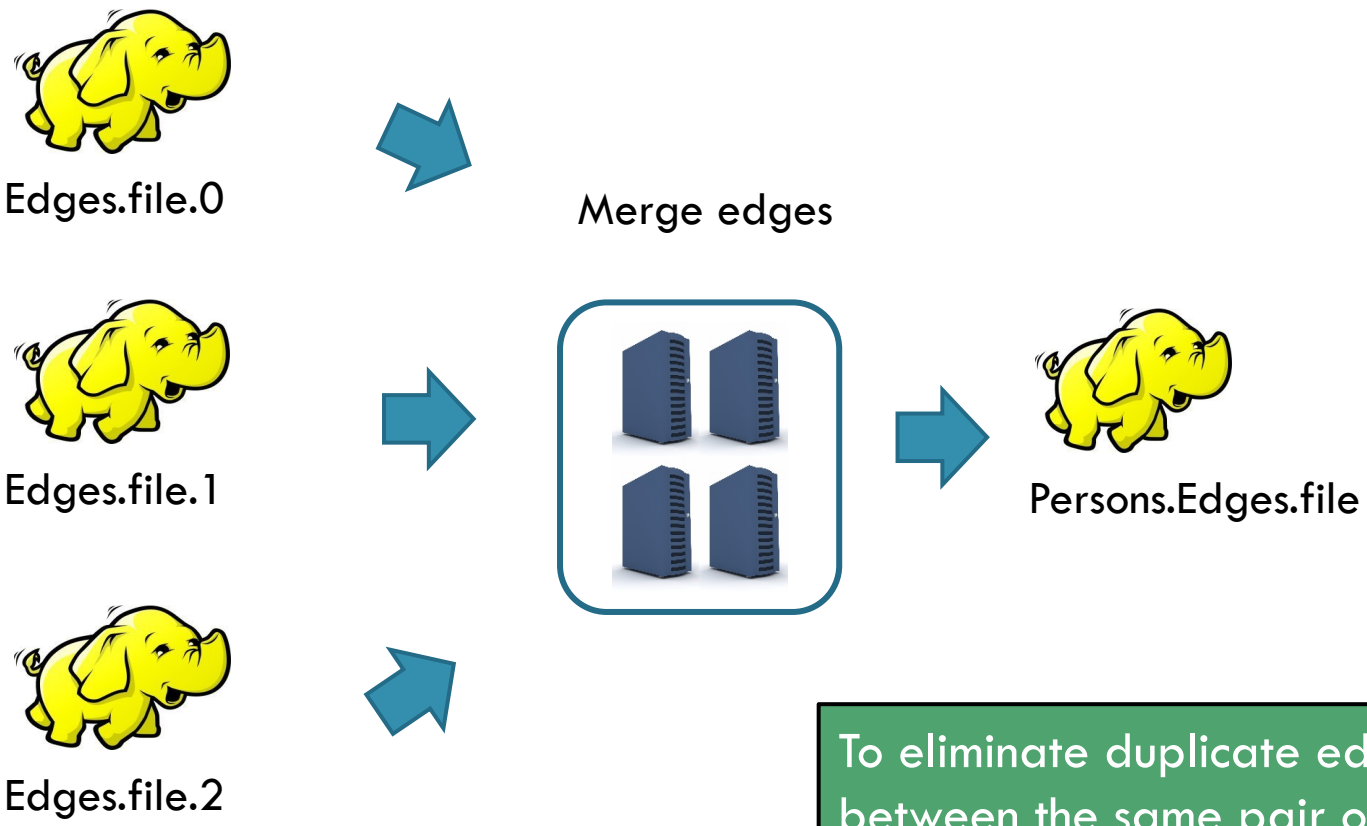


- Sort by correlation dimension:
  - e.g. Main interest, University-age, random
- Set Person keys as their position in the sorted array (between 0 and N-1)

Edge Generation Substep



Edge Merge



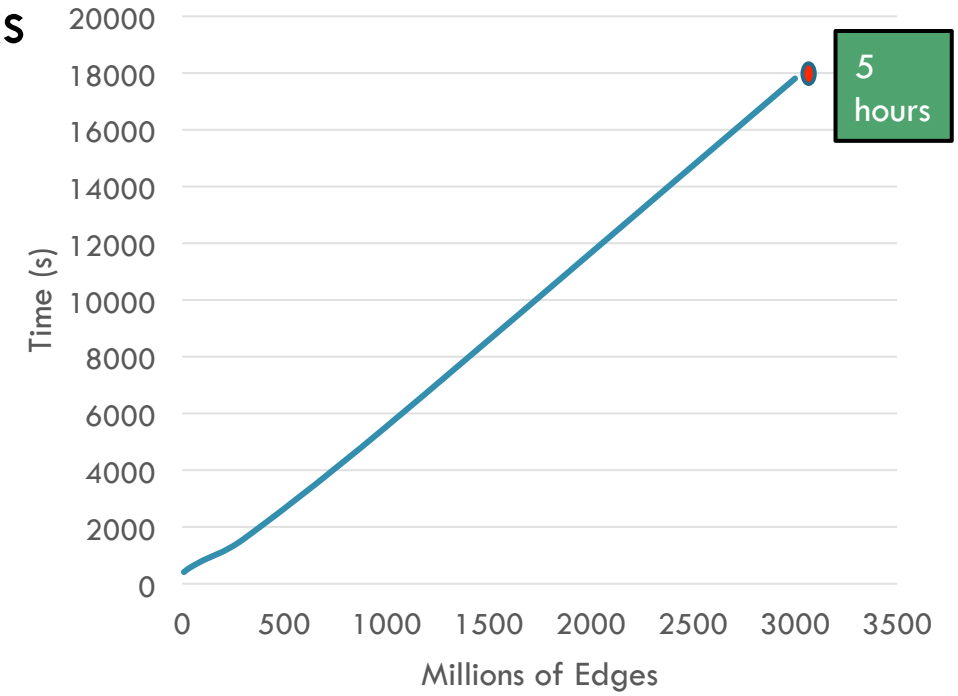
To eliminate duplicate edges between the same pair of Persons

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?

- References:
  - 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.
  - Capota, M., Hegeman, T., Iosup, A., Prat-Pérez, A., Erling, O., & Boncz, P. (2015). Graphalytics: A Big Data Benchmark for Graph-Processing Platforms.
    - http://ldbcouncil.org/sites/default/files/LDBC_D2.2.2.pdf
    - http://ldbcouncil.org/sites/default/files/LDBC_D3.3.34.pdf

Graphalytics

Systems and models

Methodology

Architecture

Agenda

- Introduction to Linked Data
- LDBC Social Network Benchmark (SNB)
- 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

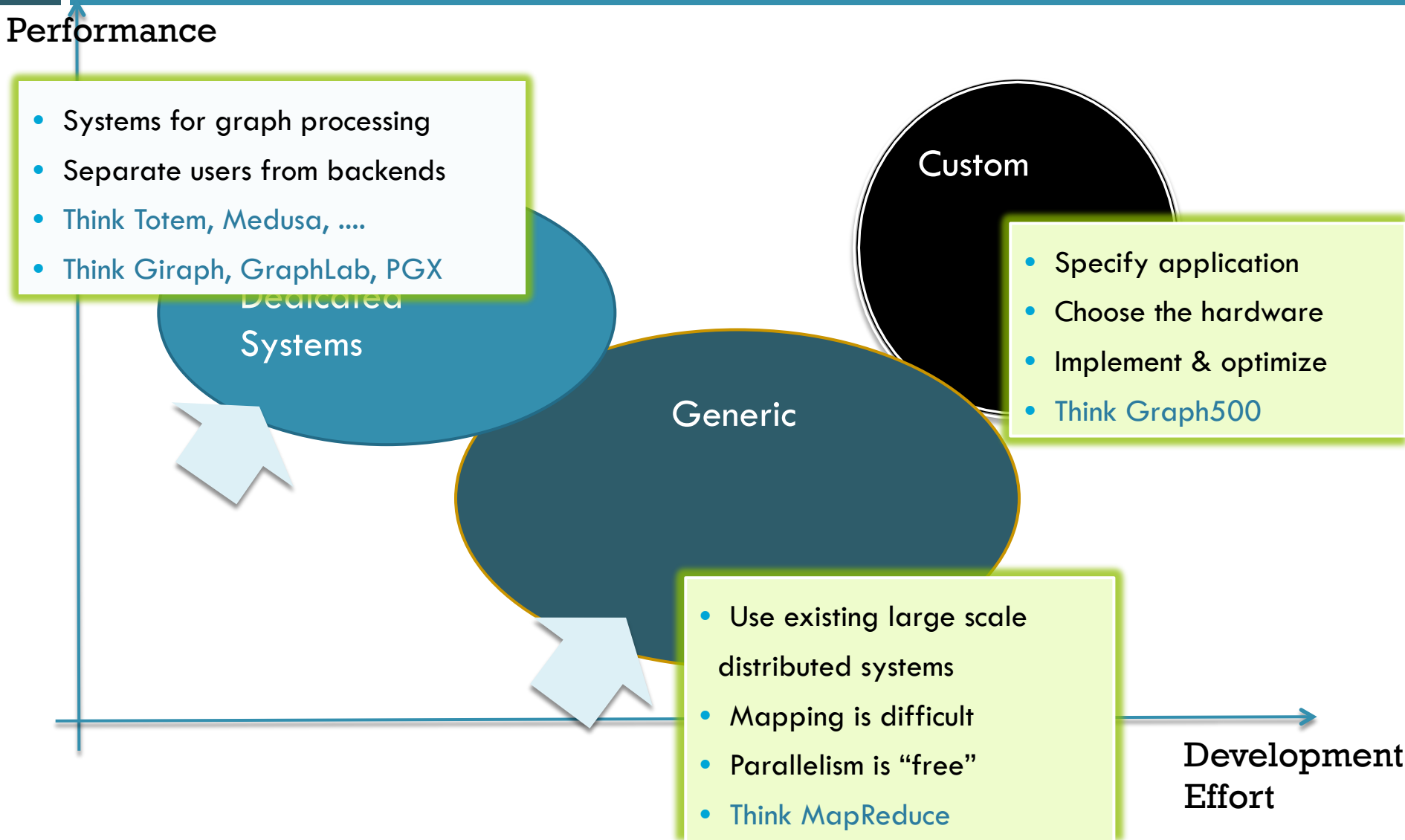
- The characteristics of graph processing
  - Poor locality
  - Unstructured computation
  - Variable parallelism
  - Low computer-to-memory ratio
- @ Scale
  - Distributed processing is mandatory
  - Parallel processing is very useful

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



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

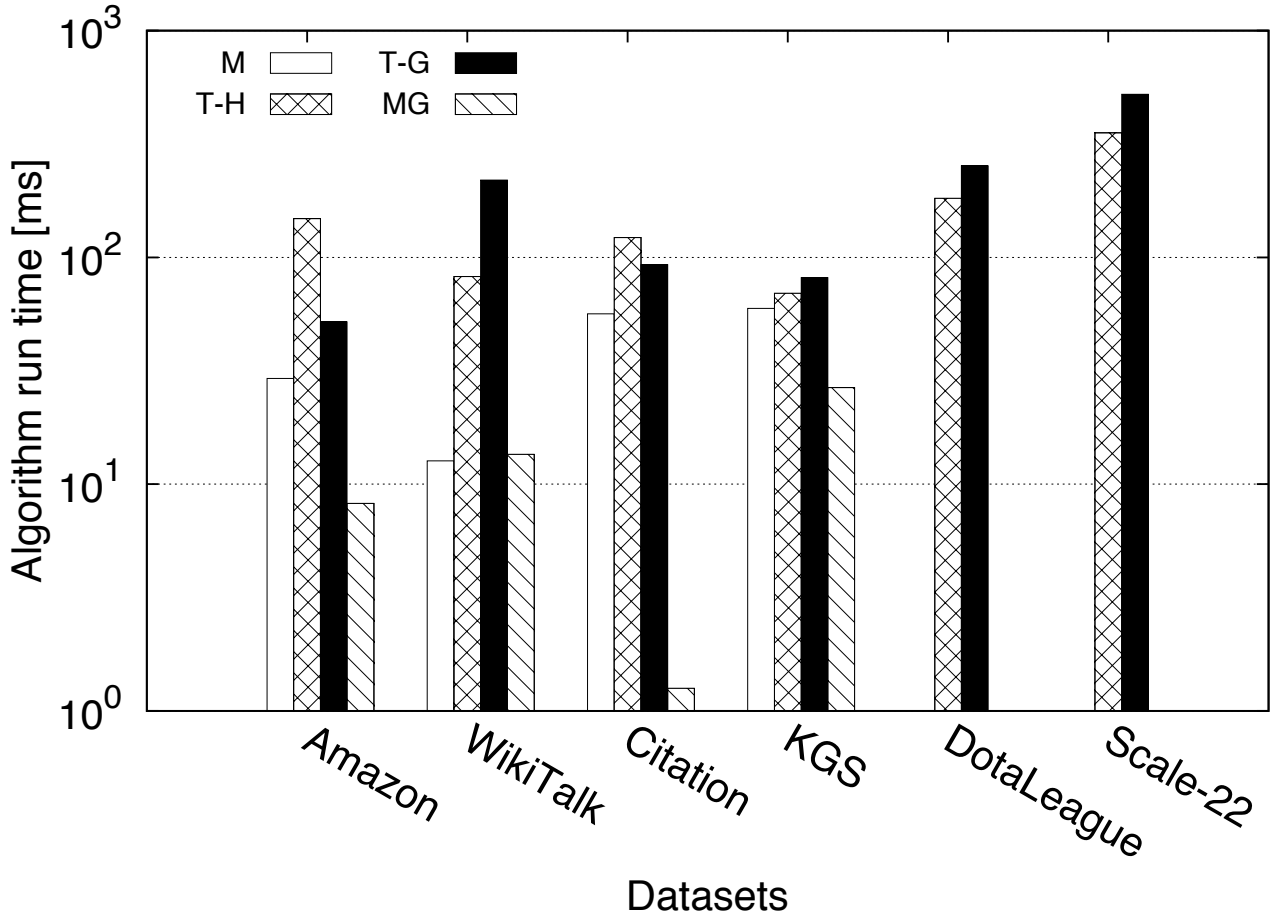
- Target at high performance graph analytics on GPUs.
- API based on the Gather-Apply-Scatter (GAS) model as used in GraphLab.
  - Productivity-oriented API
- Single GPU available and Multi-GPU ready
  - Also available in a CPU-only version

Evaluation setup

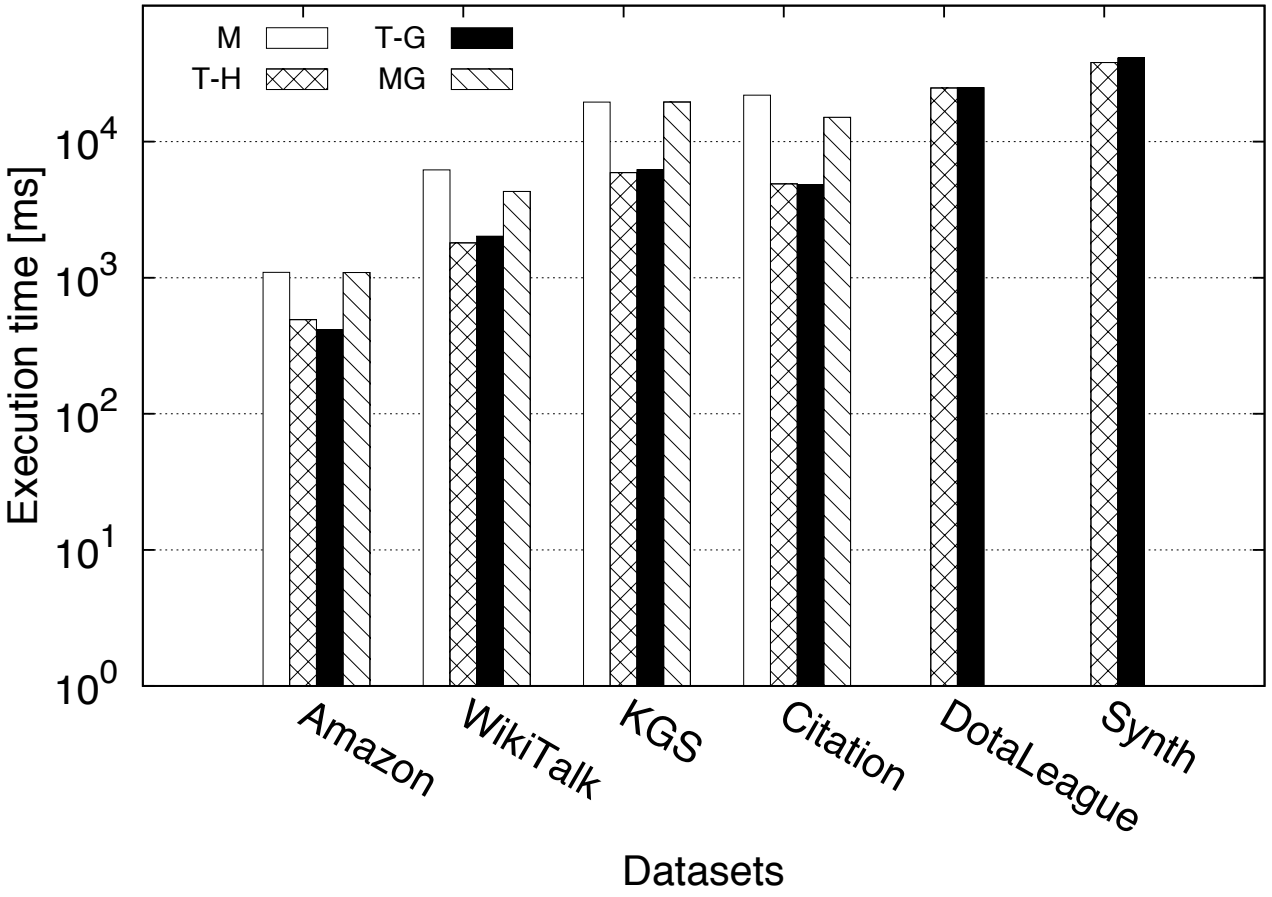
- Use GPU-enabled graph platforms to compare their performance*
- Datasets:
  - SNAP repository
  - Graph500 generated benchmarks
    - Scale-22/Synth
- Algorithms
  - BFS (traversal)
  - PageRank
  - Weakly connected components

*Accepted in CCGrid'15, Y.Guo et. al: ""

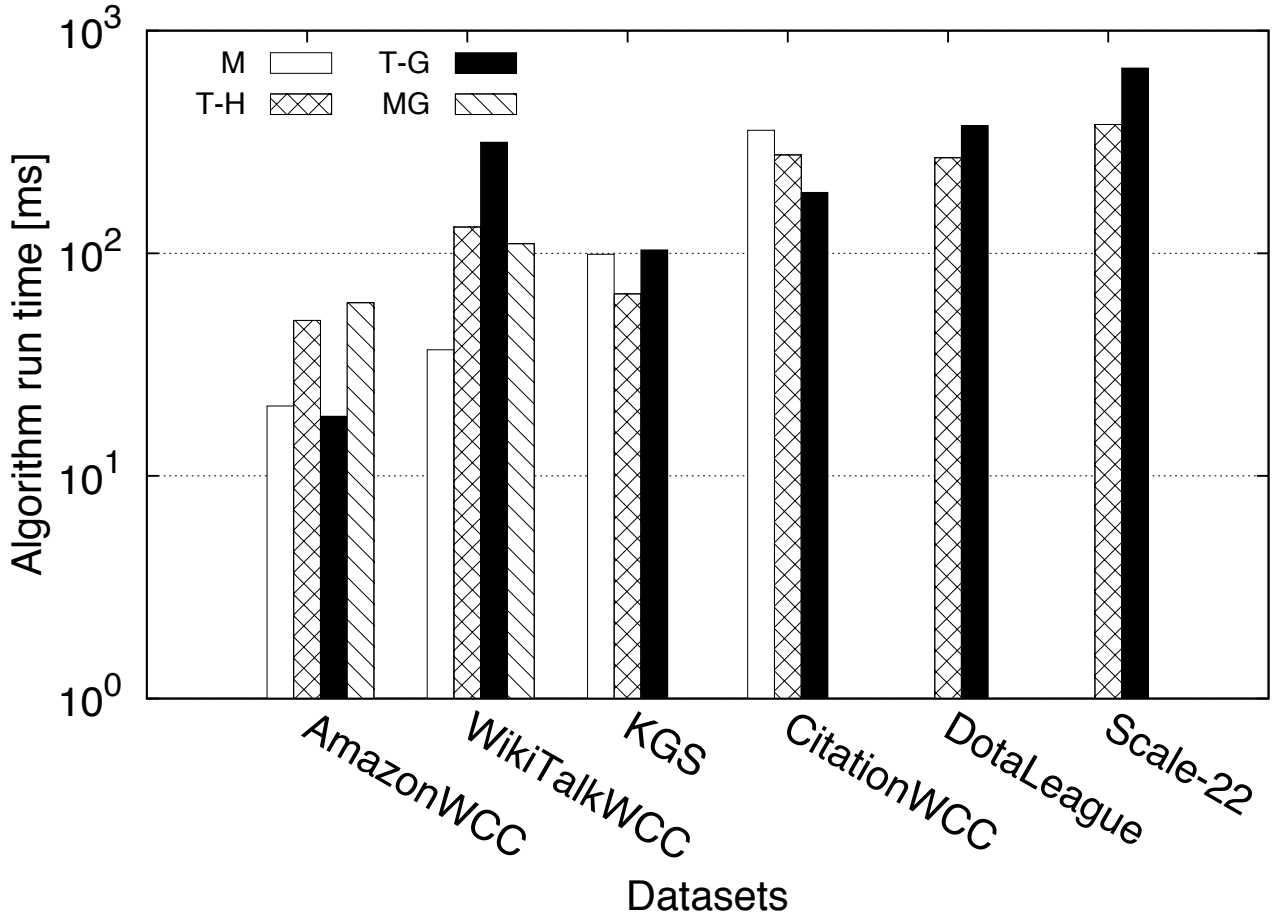
BFS [algorithm]



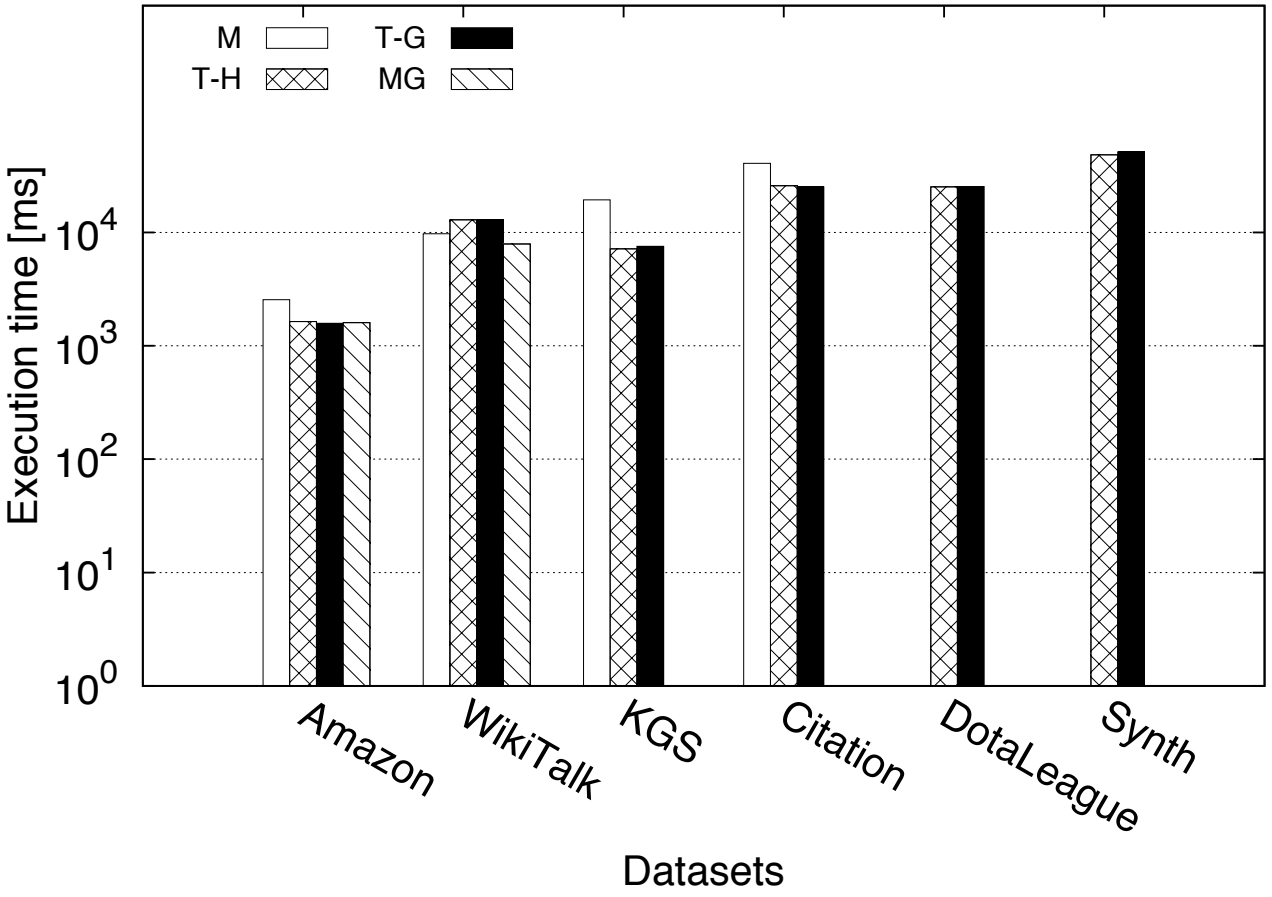
BFS [full]


M
T-G
T-H
MG
Execution time [ms]
10^4
10^3
10^2
10^1
10^0
Amazon
WikiTalk
KGS
Citation
DotaLeague
Synth
Datasets

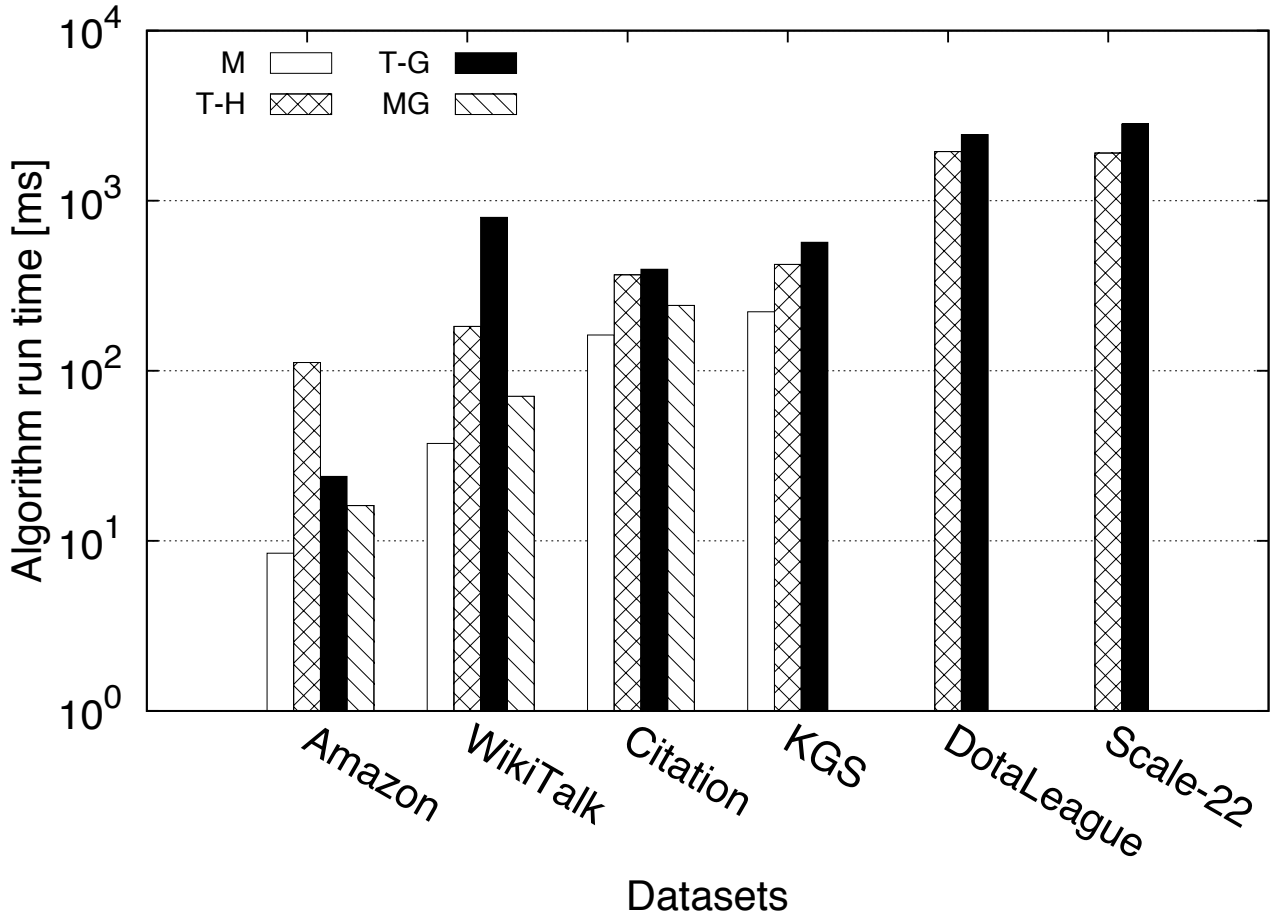
WCC [algorithm]



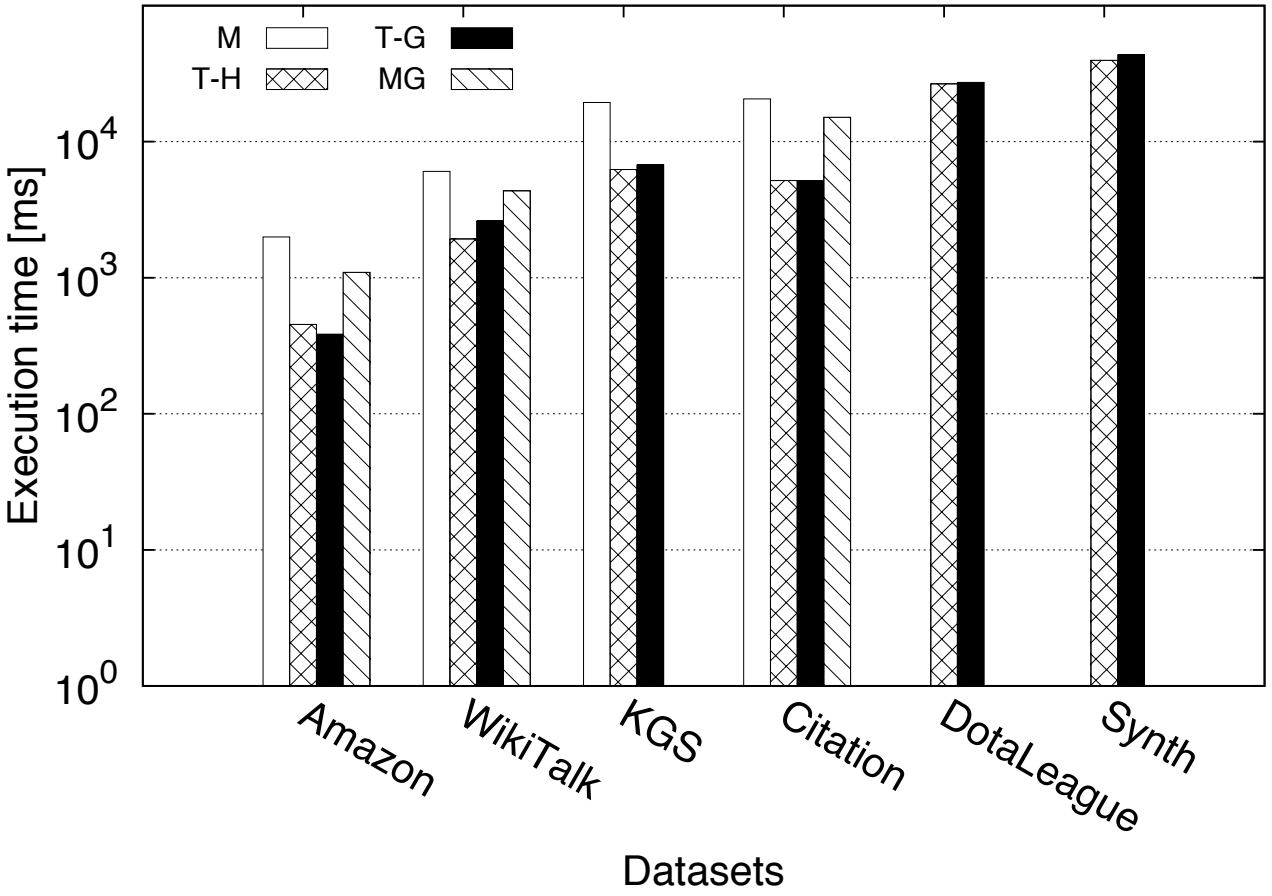
WCC [full]


M
T-G
T-H
MG
Execution time [ms]
10^4
10^3
10^2
10^1
10^0
Amazon
WikiTalk
KGS
Citation
DotaLeague
Synth
Datasets

PageRank [algorithm]



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

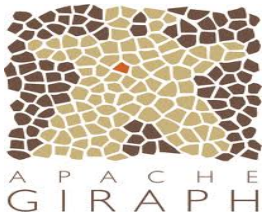
Distributed/Large Scale platforms

Interesting platforms

- Distributed or non-distributed
- Dedicated or generic

Distributed (Generic)

Oracle Labs
PGX



Distributed
(Dedicated)

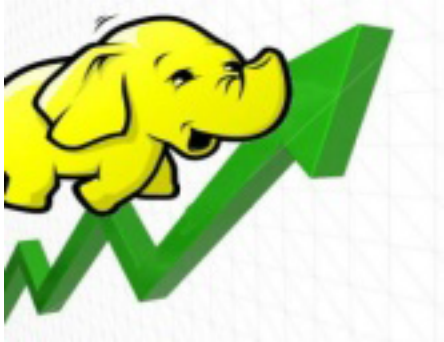
Non-distributed
(Dedicated)

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 the form of 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 manages jobs using resources provided by YARN

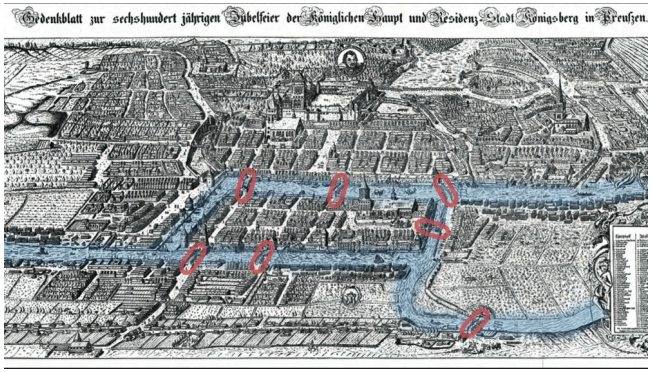


Stratosphere (Generic)

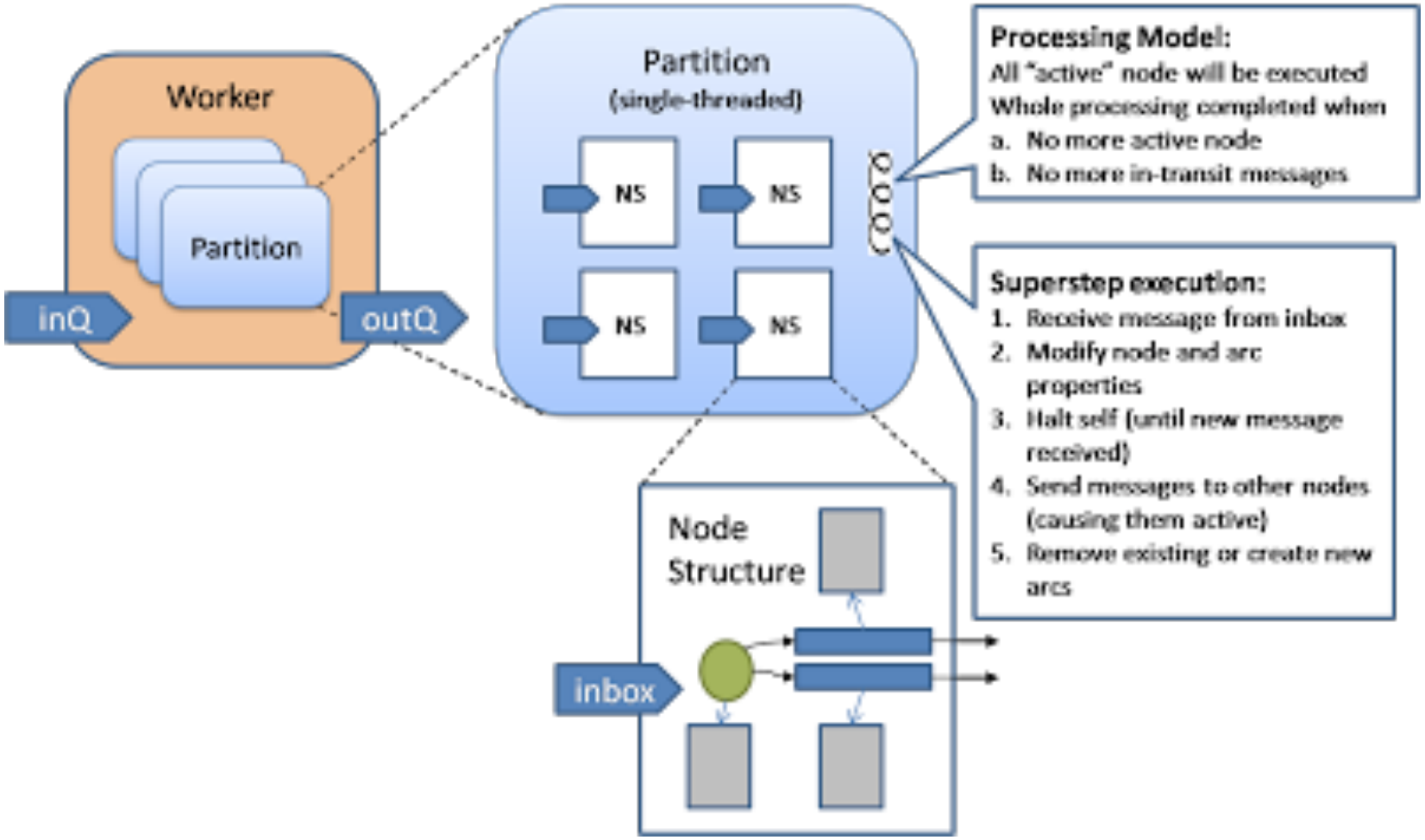
- Now Apache Flink
- Nephele resource manager
  - Scalable parallel engine
  - Jobs are represented as DAGs
  - Supports data flow in-memory, via network, or on files
- PACT job model
  - 5 second-order functions (MapReduce has 2): Map, Reduce, Match, Cross, and CogGroup
  - Code annotations for compile-time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

Pregel: dedicated graph-processing model

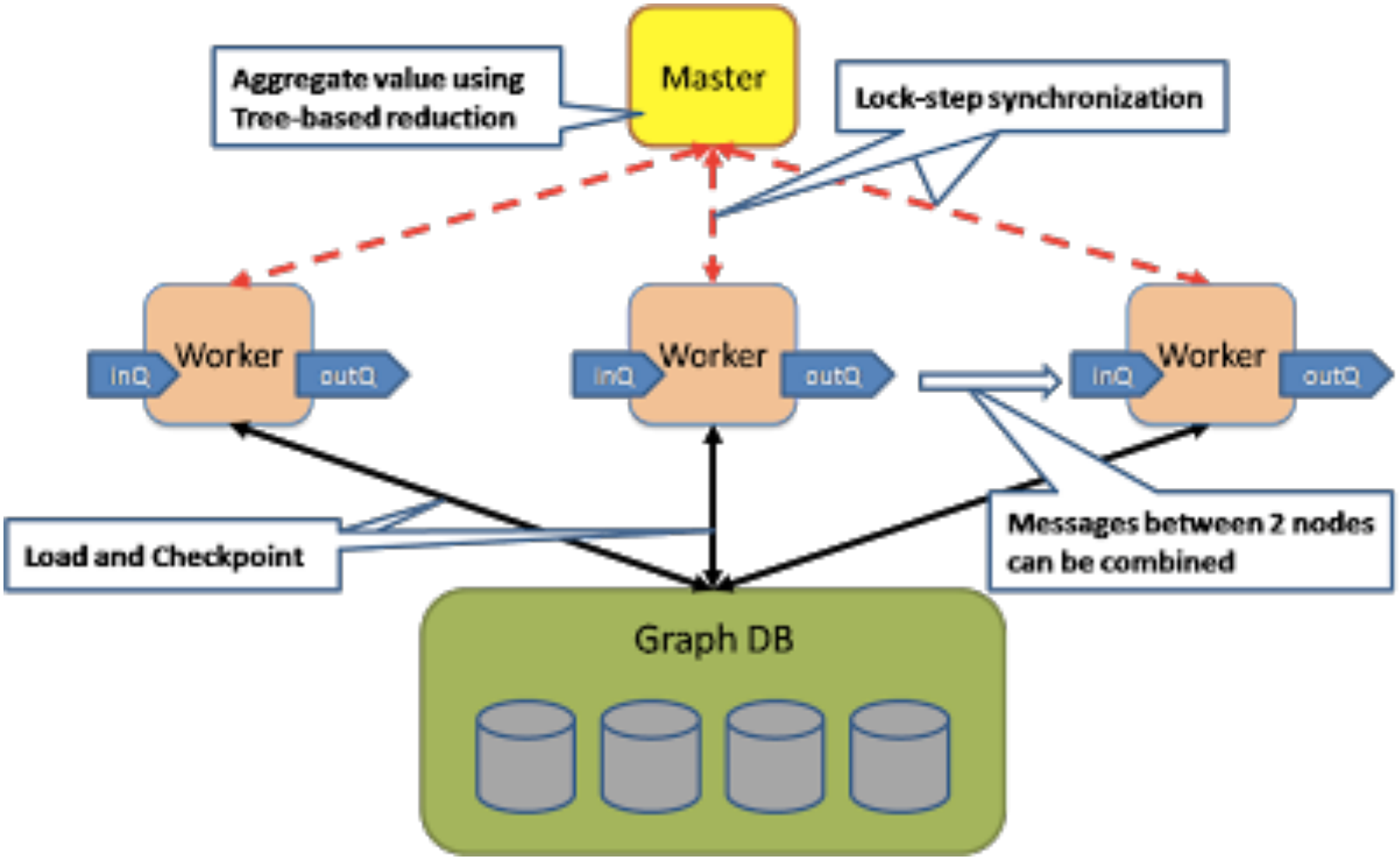
- Proposed a vertex-centric approach to graph processing
  - Graph-to-graph transformations
- Front-end:
  - Write the computation that runs on all vertices
  - Each vertex can vote to halt
    - All vertexes halt => terminate
  - Can add/remove edges and vertices
- Back-end:
  - Uses the BSP model
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



Pregel

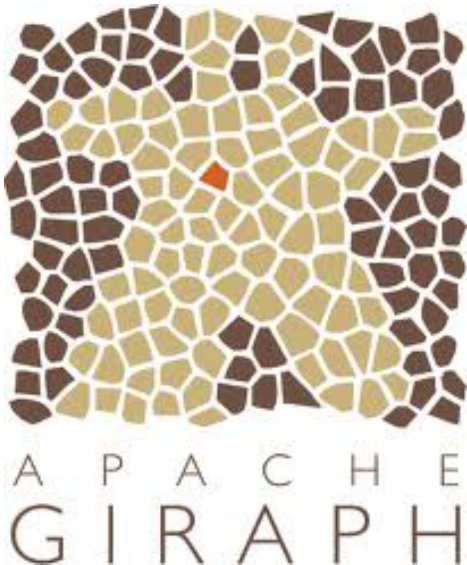

Worker
Partition
inQ
outQ
Partition
(single-threaded)
NS
NS
NS
NS
Node
Structure
inbox
Processing Model:
All "active" node will be executed
Whole processing completed when
a. No more active node
b. No more in-transit messages
Superstep execution:
1. Receive message from inbox
2. Modify node and arc properties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

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

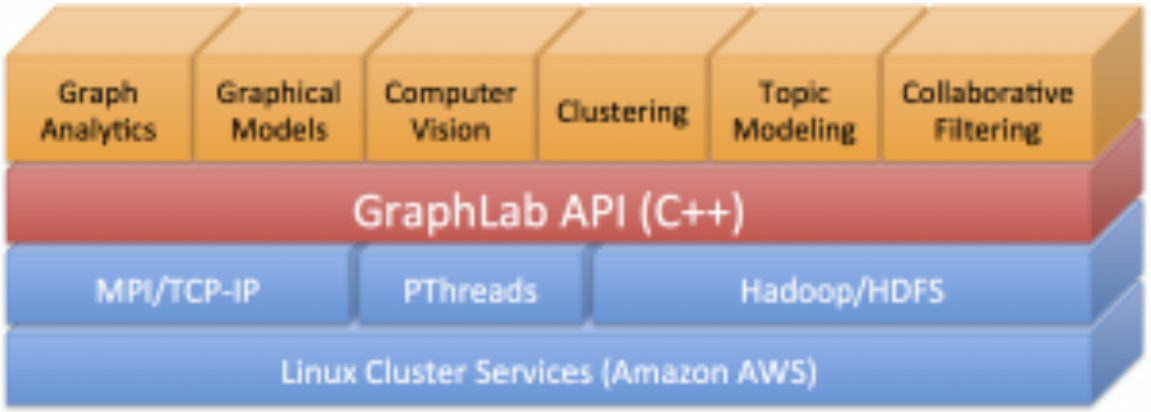
Apache Giraph (Dedicated)

- Based on the Pregel model
- Uses YARN as back-end (yet another framework)
- In-memory
  - Limitations in terms of partition sizes
  - Spilling to disk is work in progress
- 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)

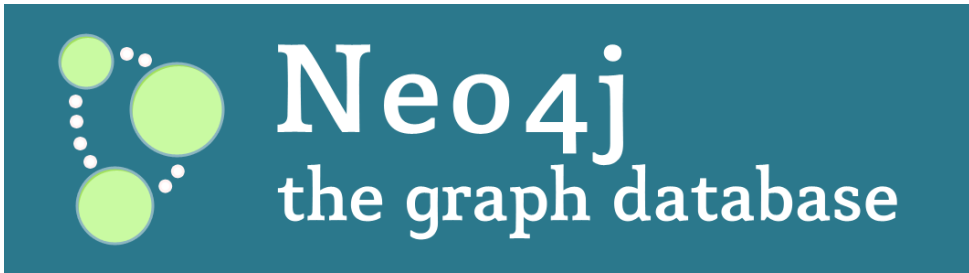


- All in-memory
- Supports asynchronous processing
- GraphChi is its single-node version, Dato as GraphLab company

GraphLab

Neo4J (Dedicated)

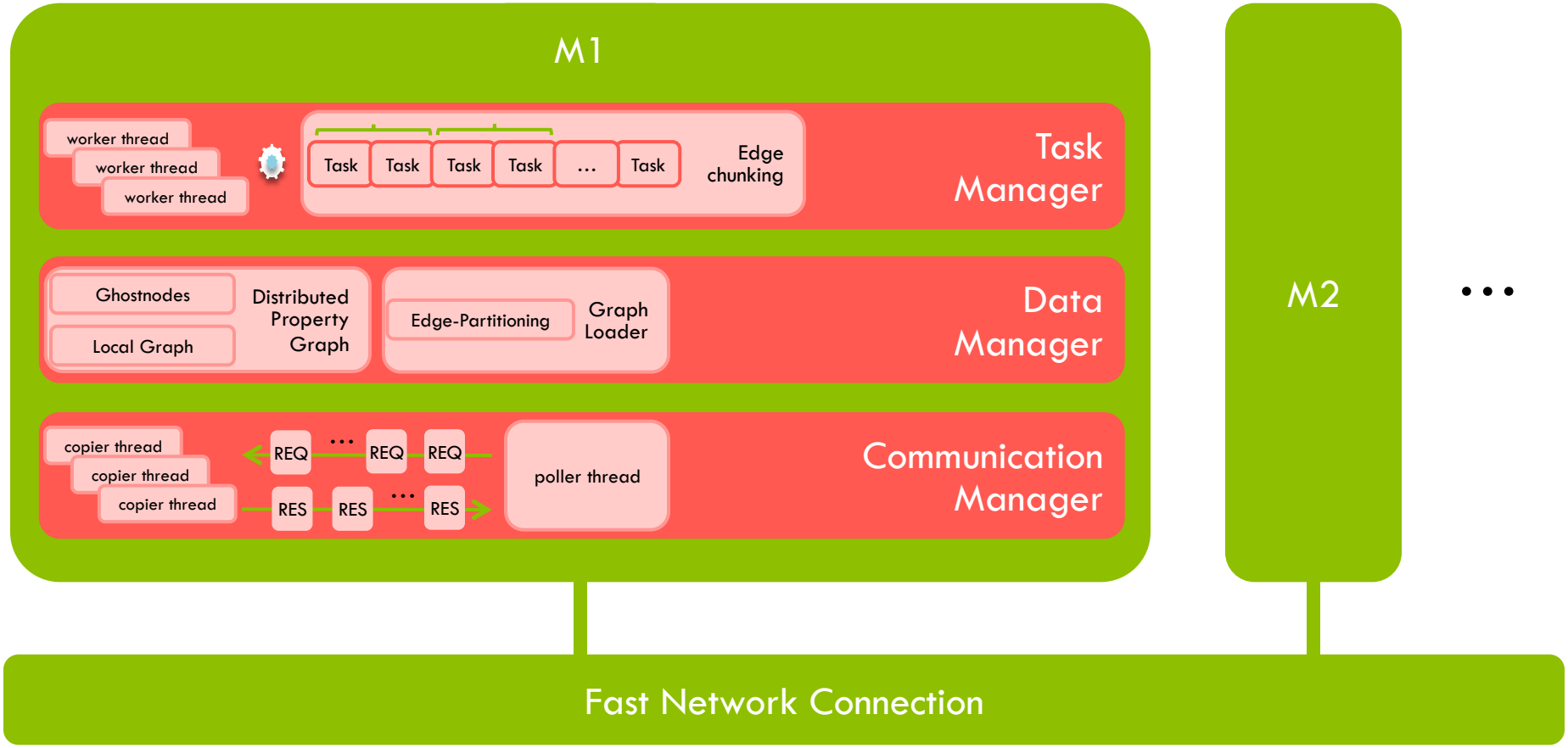
- Very popular graph database
  - Graphs are represented as relationships and annotated vertices
- Single-node system
  - Uses parallel processing
  - Additional caching and query optimizations
  - All in-memory
- The most widely used solutions for medium-scale problems



PGX.D (Dedicated)

- Very fast distributed graph processing system
  - Beats GraphLab and GraphX by orders of magnitude
- Low-overhead communication mechanism
  - Lightweight cooperative context switching mechansim
- Support for data-pulling
  - Intuitive transformation of classical graph algorithms
- Reducing traffic and balancing workloads
  - Several advanced techniques: Selective Ghostnodes, edge based partitioning, edge chunking
- Justification for beefy clusters
  - Fully exploits the underlying resources of modern beefy cluster machines

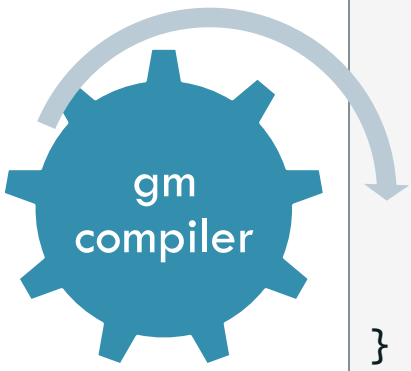
PGX.D: System Design Overview



PGX.D: Programming Model

Intuitive 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);
  }
}
```

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
  - 7 platforms
- Implement **all algorithms** on **all platforms**
- Run and compare …
  - Performance
- 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

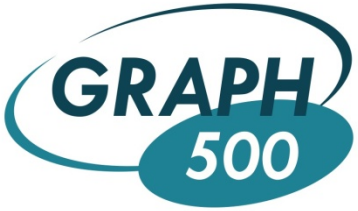
Hardware

- 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

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

https://snap.stanford.edu/



http://www.graph500.org/

The Game Trace Archive

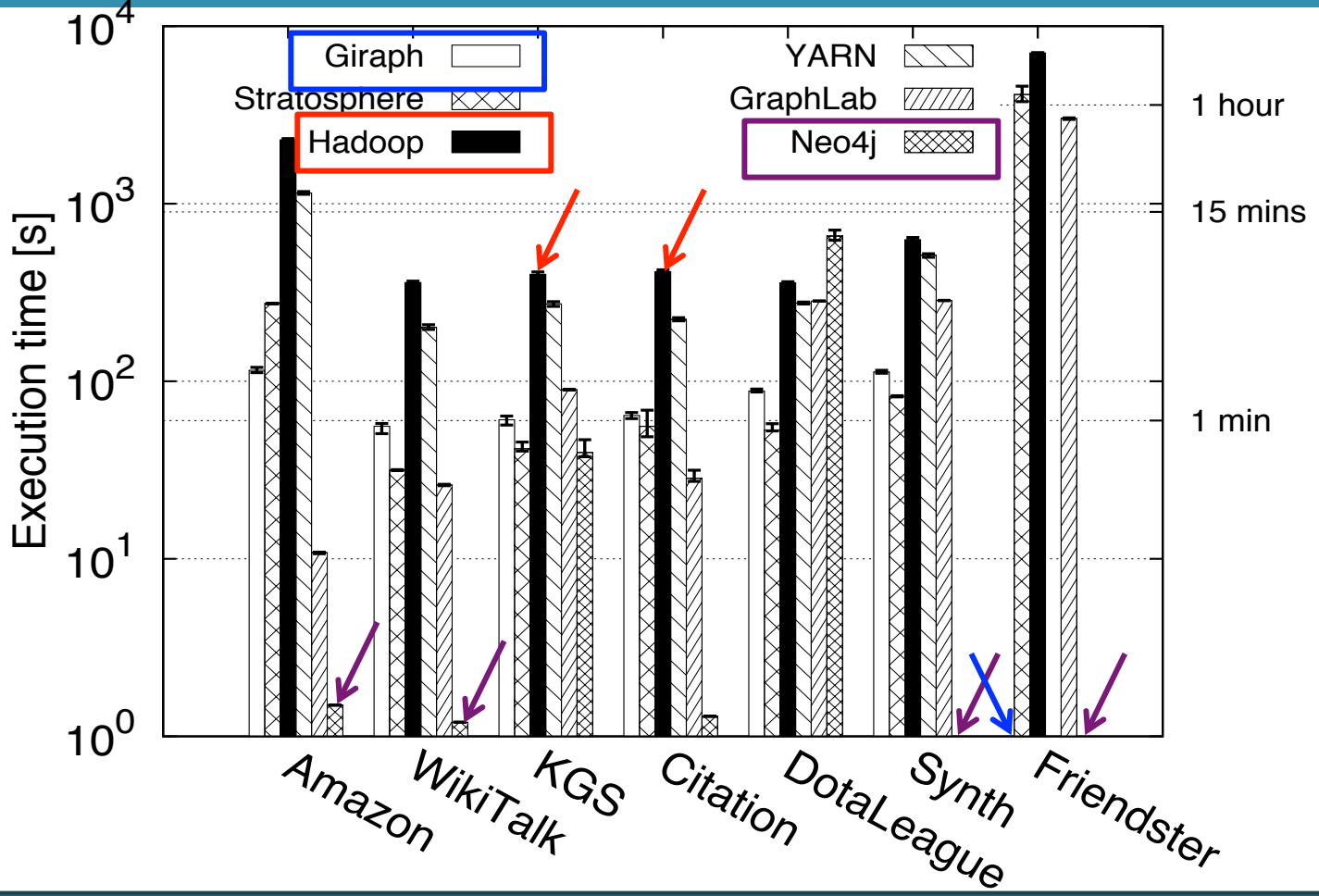
http://gta.st.ewi.tudelft.nl/

Graph-Processing Algorithms

- Literature survey
  - 10 top research conferences: SIGMOD, VLDB, HPDC …
  - 2009–2013, 124 articles

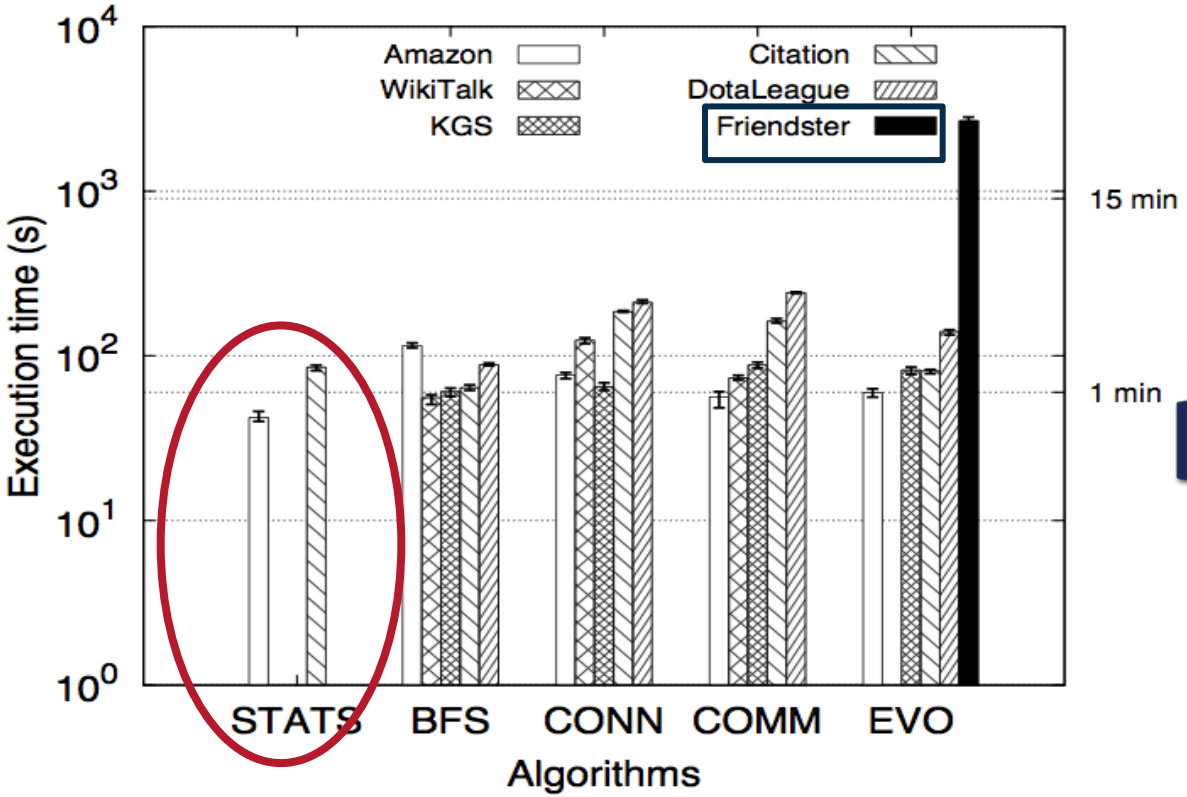
Class	Examples	%
Graph Statistics	Diameter, PageRank	16.1
Graph Traversal	BFS, SSSP, DFS	46.3
Connected Component	Reachability, BiCC	13.4
Community Detection	Clustering, Nearest Neighbor	5.4
Graph Evolution	Forest Fire Model, PAM	4.0
Other	Sampling, Partitioning	14.8

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.

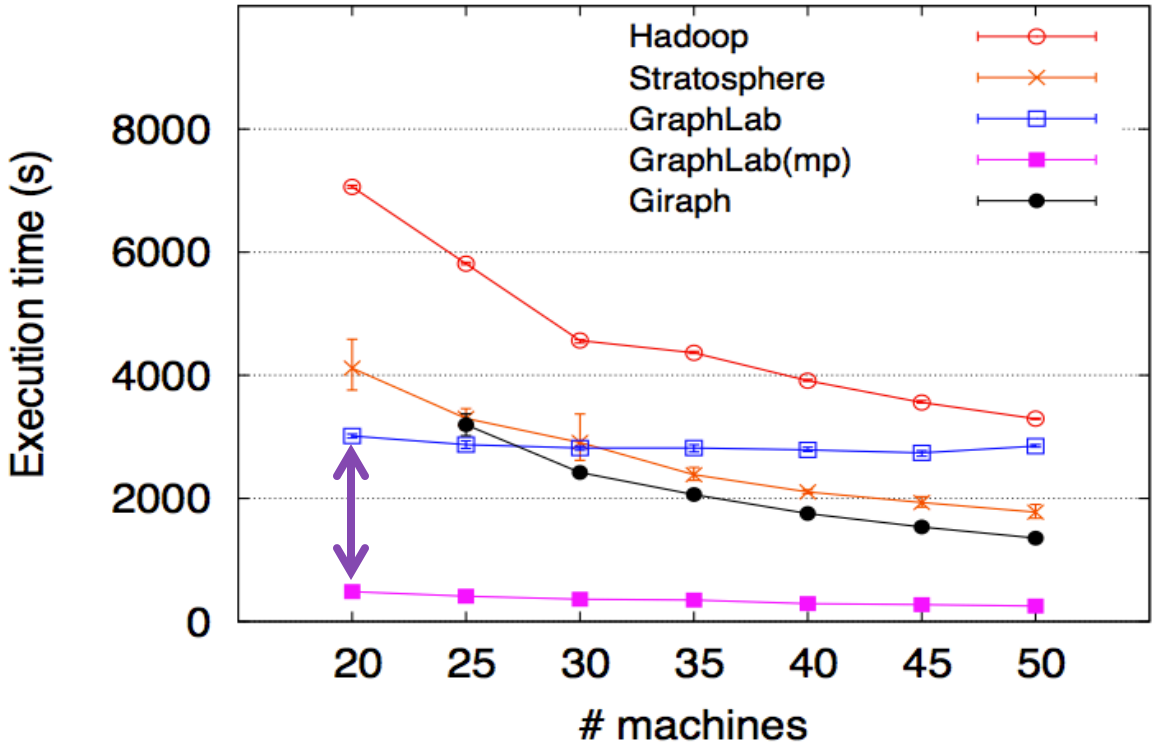
Giraph:
Results for (algo*,platform*)



DAS 4

Storing the whole graph in memory helps Giraph perform well
Giraph may crash when graphs or number of messages large

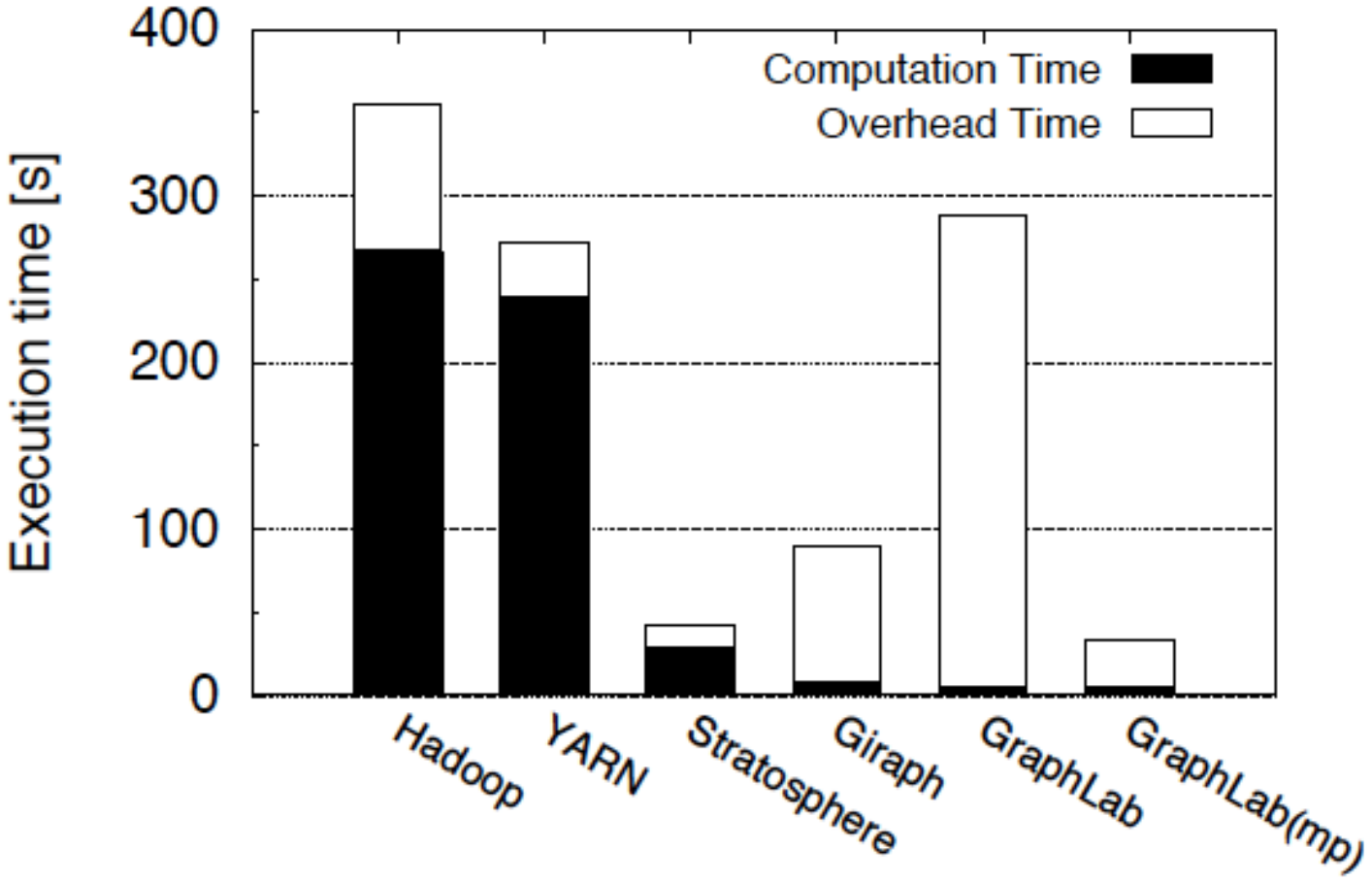
Horizontal scalability: BFS on Friendster (31 GB)



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

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

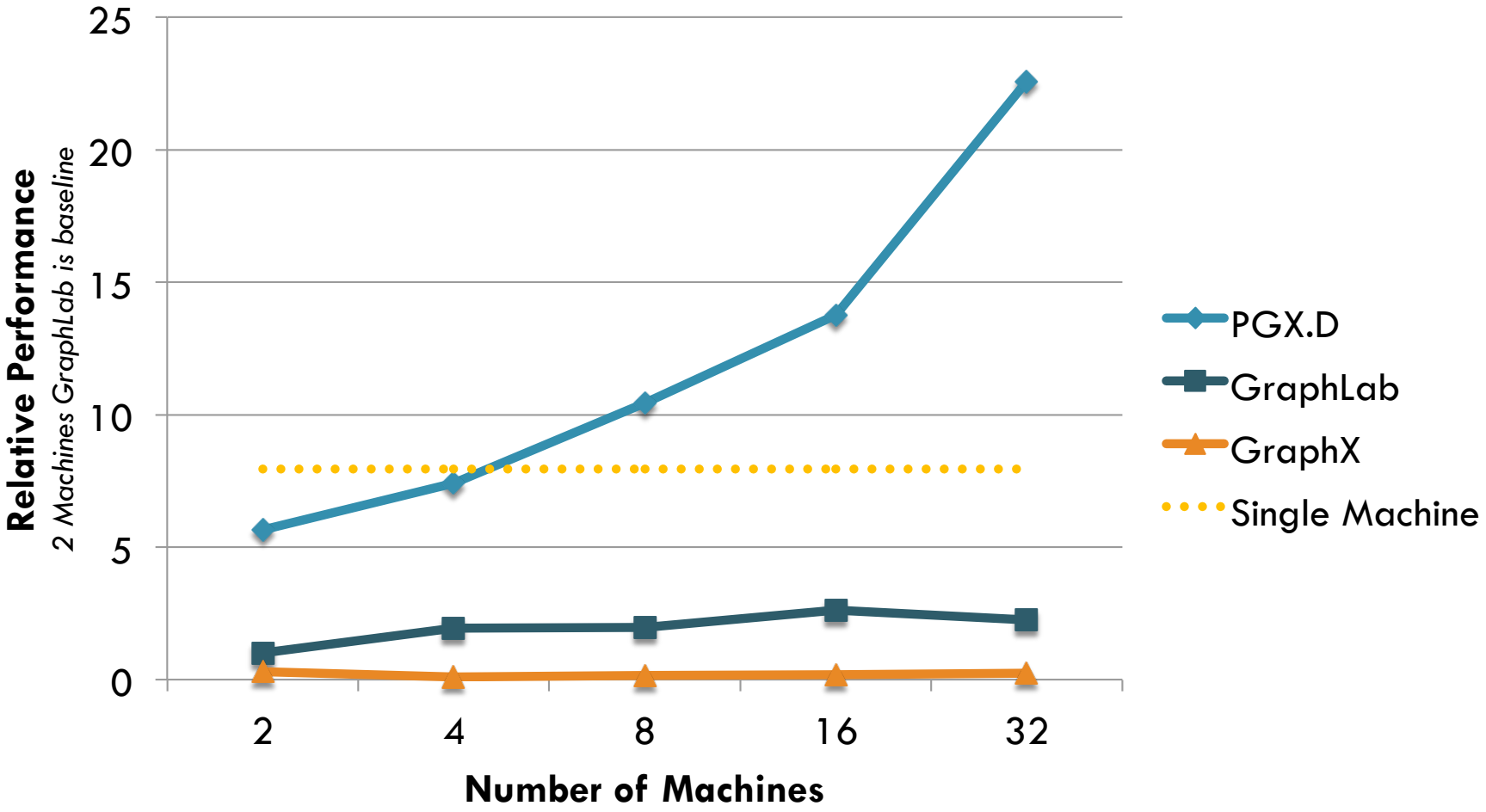
	Hadoop(Java)	Stratosphere(Java)	Giraph(Java)	GraphLab(C++)	Neo4j(Java)
BFS	1 d, 110 loc	1 d, 150 loc	1 d, 45 loc	1 d, 120 loc	1 h, 38 loc
CONN	1.5 d, 110 loc	1 d, 160 loc	1 d, 80 loc	0.5 d, 130 loc	1 d, 100 loc

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

We need better productivity metrics!

PGX.D: Performance Evaluation

(PageRank, Twitter, Infiniband)



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
    - workload scaling tricky

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

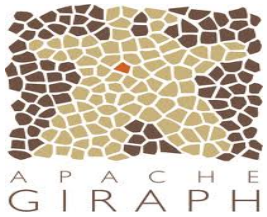
Methodology

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

A systematic approach

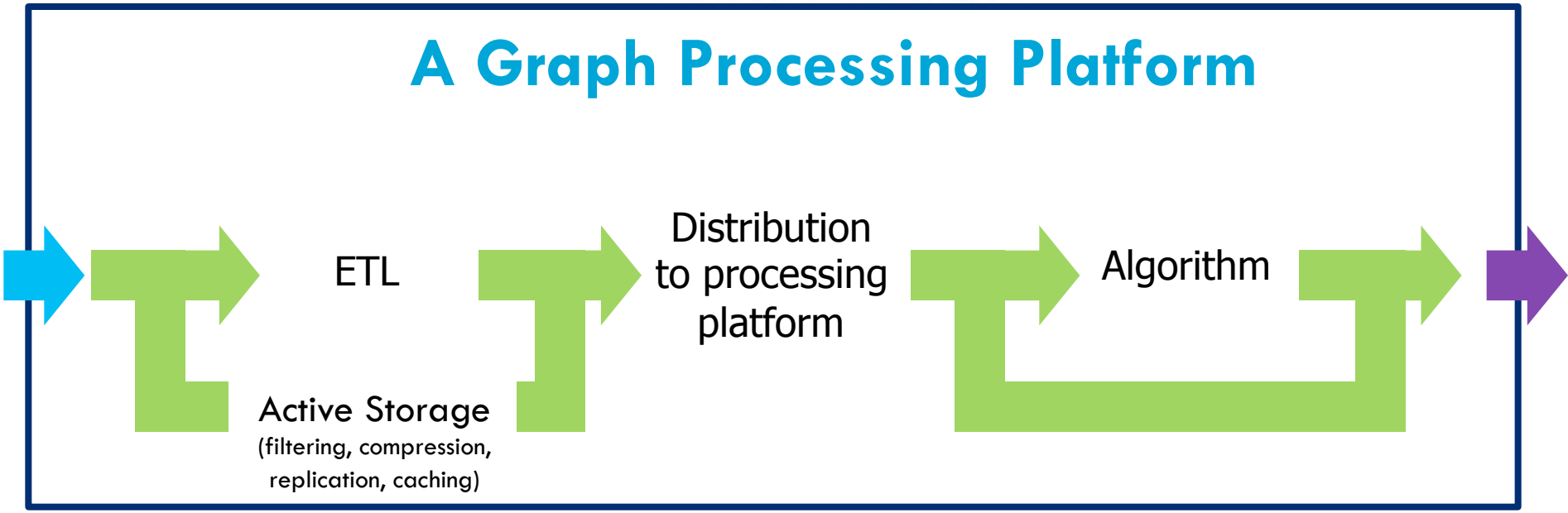
Graph Processing Platforms

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

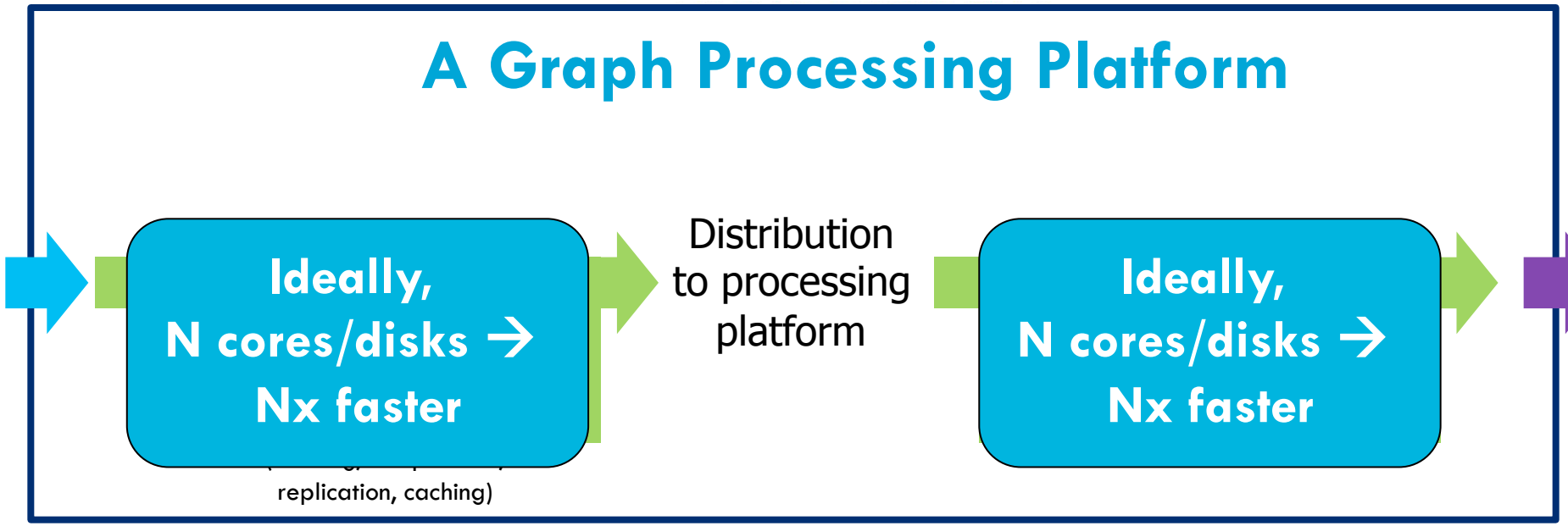


Which to choose?
What to tune?

Abstraction



Objectives: scalability & peformance



Evaluating graph-processing platforms

Metrics Diversity

Graph Diversity

Algorithm Diversity

Graphalytics = comprehensive benchmarking suite for graph processing across all platforms

Graphalytics:
A Challenging Benchmarking Process

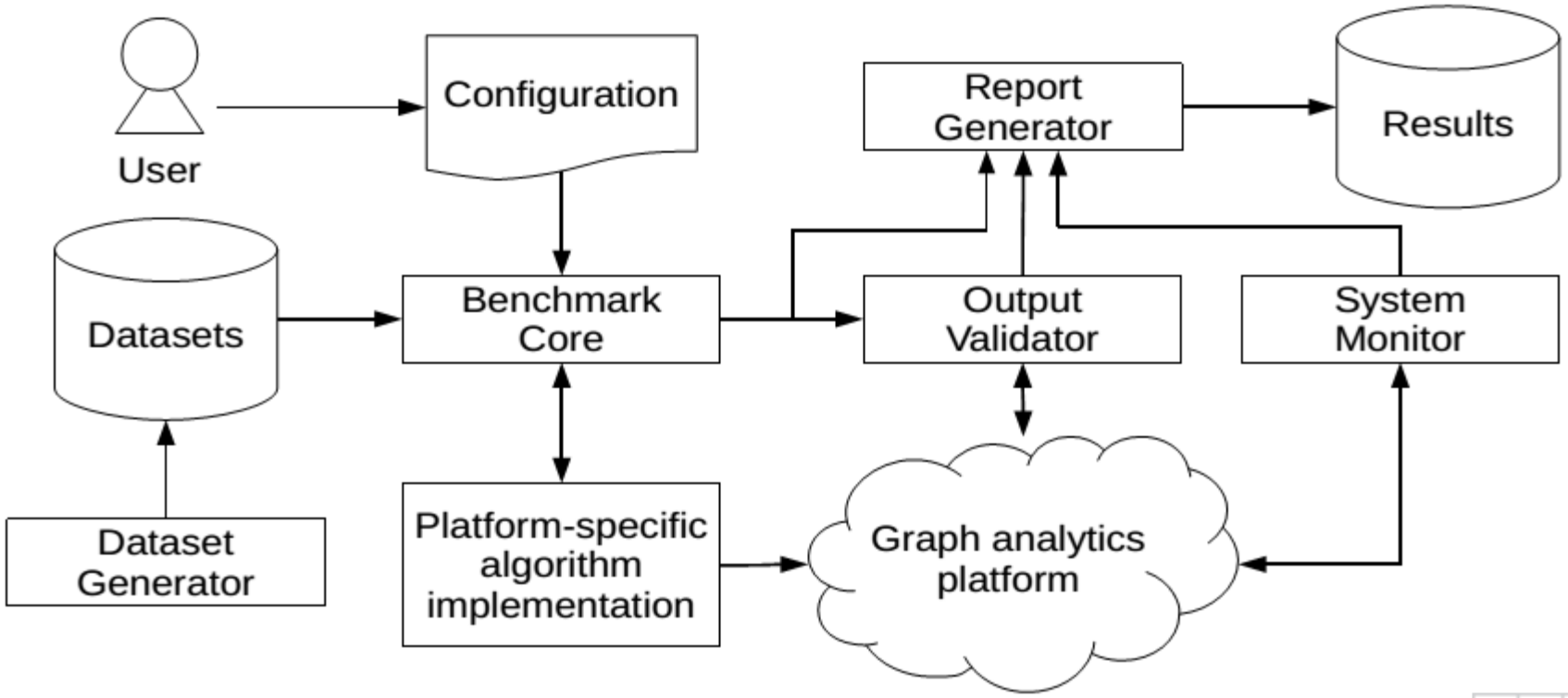
- Methodological challenges
  - Challenge 1. Evaluation process
  - Challenge 2. Selection and design of performance metrics
  - Challenge 3. Dataset selection and analysis of coverage
  - Challenge 4. Algorithm selection and analysis of coverage
- Practical challenges
  - Challenge 5. Scalability of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

LDBC The graph & RDF benchmark reference

UPC CWI ORACLE

Y. Guo, A. L. Varbanescu, A. Iosup, C. Martella, T. L. Willke: Benchmarking graph-processing platforms: a vision. ICPE 2014: 289-292

Graphalytics

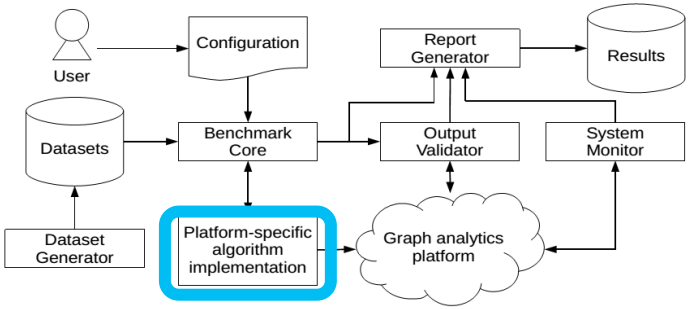


LDBC The graph & RDF benchmark reference

UPC CWI ORACLE

spec Research

Graphalytics:
Many Classes of Algorithms



Literature survey of metrics, datasets, and algorithms

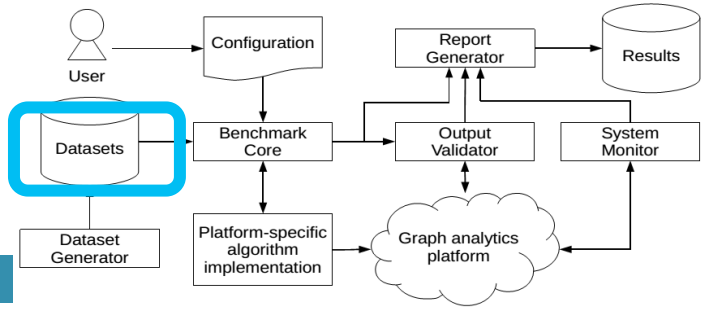
- All 124 articles in 10 top research conferences: SIGMOD, VLDB, HPDC … (2009–2013)

Class	Examples	%
Graph Statistics	Diameter, PageRank	16.1
Graph Traversal	BFS, SSSP, DFS	46.3
Connected Component	Reachability, BiCC	13.4
Community Detection	Clustering, Nearest Neighbor	5.4
Graph Evolution	Forest Fire Model, PAM	4.0
Other		

Future work

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: 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					undirected
G4	Citation					directed
G5	DotaLeague					undirected
G6	Synth					undirected
G7	Friendster	65,608,366	1,806,067,135	0.1	55.1	undirected

Interaction graphs (possible work)

SNAP

https://snap.stanford.edu/

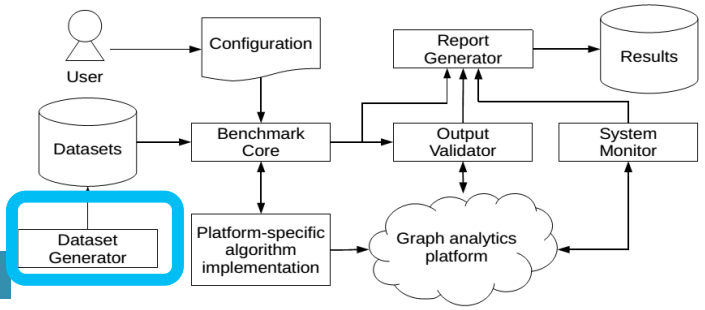
LDBC Social Network Generator

The Game Trace Archive

http://gta.st.ewi.tudelft.nl/

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

Graphalytics: Enhanced LDBC Datagen



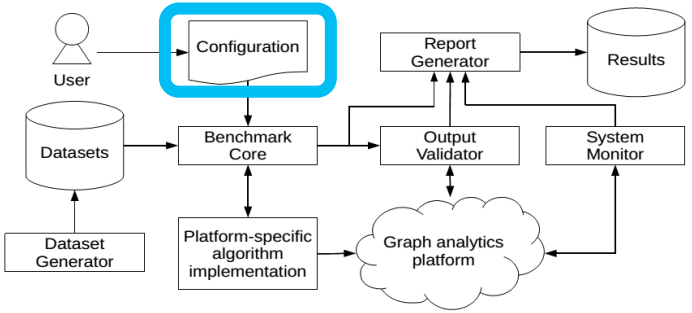
- A battery of graphs covering a rich set of configurations
- Datagen extensions to
  - More diverse degree distributions
  - Clustering coefficient and assortativity

Ongoing work

LDBC D3.3.34
http://ldbcouncil.org/sites/default/files/LDBC_D3.3.34.pdf
Orri Erling et al. The LDBC Social Network Benchmark: Interactive Workload, SIGMOD'15

LDBC The graph & RDF benchmark reference

Graphalytics: Metrics of interest



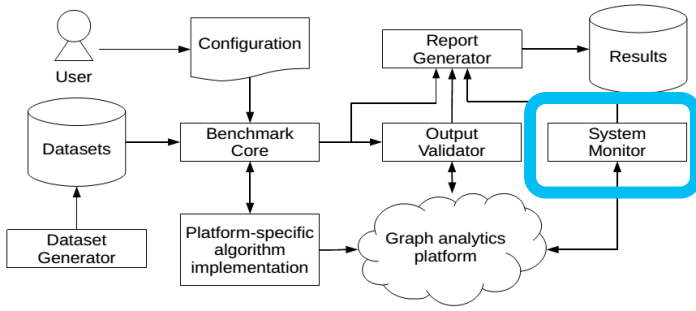
- Raw processing power
  - Execution time
  - Actual computation time
  - Normalized Edges/Vertices per second
- Resource utilization
  - CPU, memory, network
- Scalability
  - Horizontal vs. vertical
  - Strong (fixed work) vs. weak (scaled work)
- Overhead
  - Data ingestion time
  - Other overheads
- Cost?

LDBC The graph & RDF benchmark reference

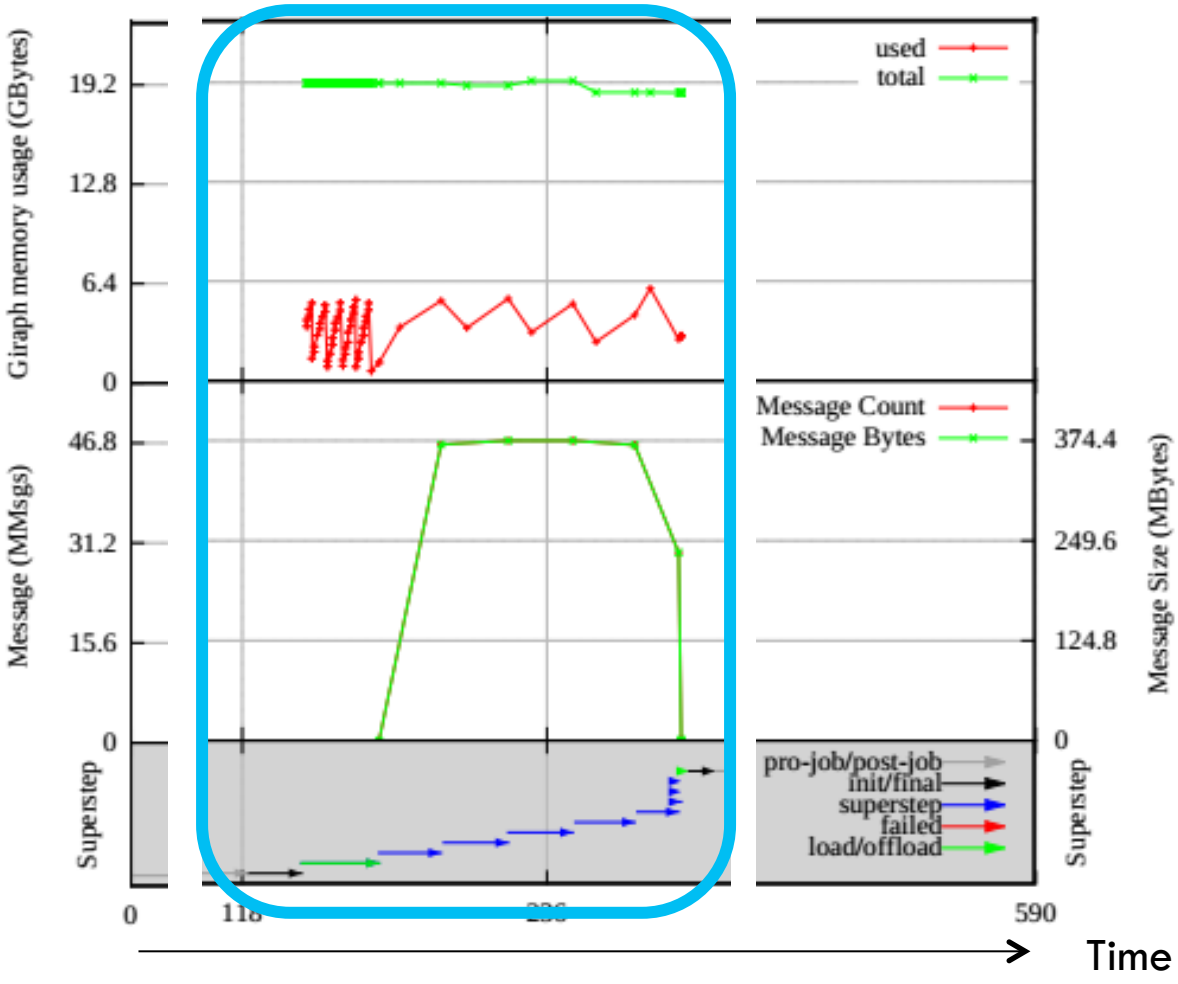
UPC CWI ORACLE

Y. Guo, A. L. Varbanescu, A. Iosup, C. Martella, T. L. Willke: Benchmarking graph-processing platforms: a vision. ICPE 2014: 289-292

Graphalytics: Monitoring & Logging



Diverse metrics: CPU, IOPS, Network, Memory use, …

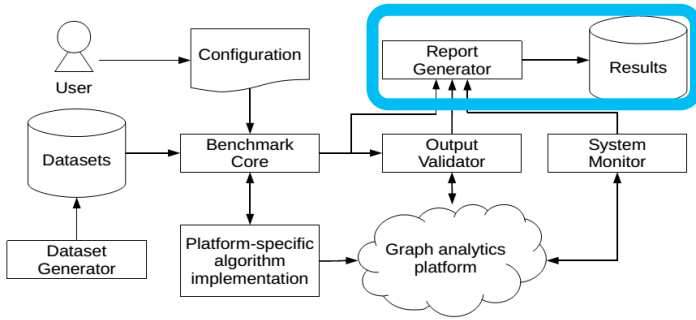


- Automatic analysis matching the programming model

Ongoing work

A. Iosup et al., Towards Benchmarking IaaS and PaaS Clouds for Graph Analytics. WBDB 2014

Graphalytics: Choke-Point Analysis



- Choke points are crucial technological challenges that platforms are struggling with

- Examples
  - Network traffic
  - Access locality
  - Skewed execution

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

near-future work

Graphalytics:
Advanced Software Engineering Process

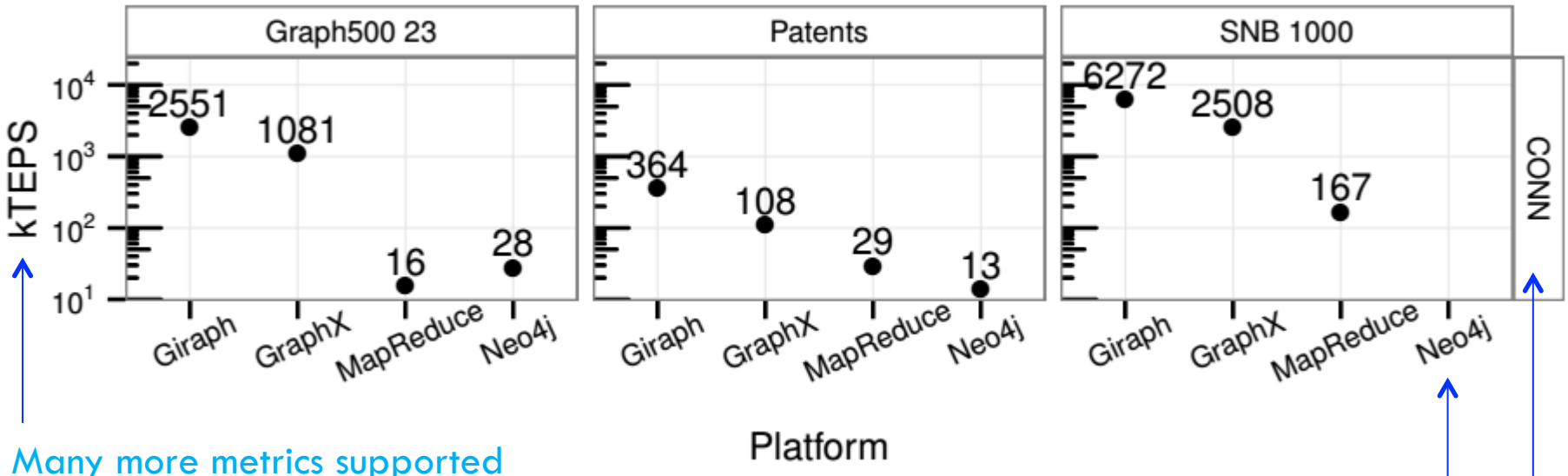
- All significant modifications to Graphalytics are peer-reviewed by developers
  - Internal release to LDBC partners (Feb 2015)
  - Public release, announced first through LDBC (Apr 2015*)
- Jenkins continuous integration server
- SonarQube software quality analyzer

https://github.com/mihaic/graphalytics/

Graphalytics:
Results easy to read/interpret

6 real-world datasets +
2 synthetic generators

Data ingestion not included here!



Many more metrics supported

10 platforms tested w prototype implementation

5 classes of algorithms

- Missing results = failures of the respective systems

The hour of benchmarking

Graphalytics in practice

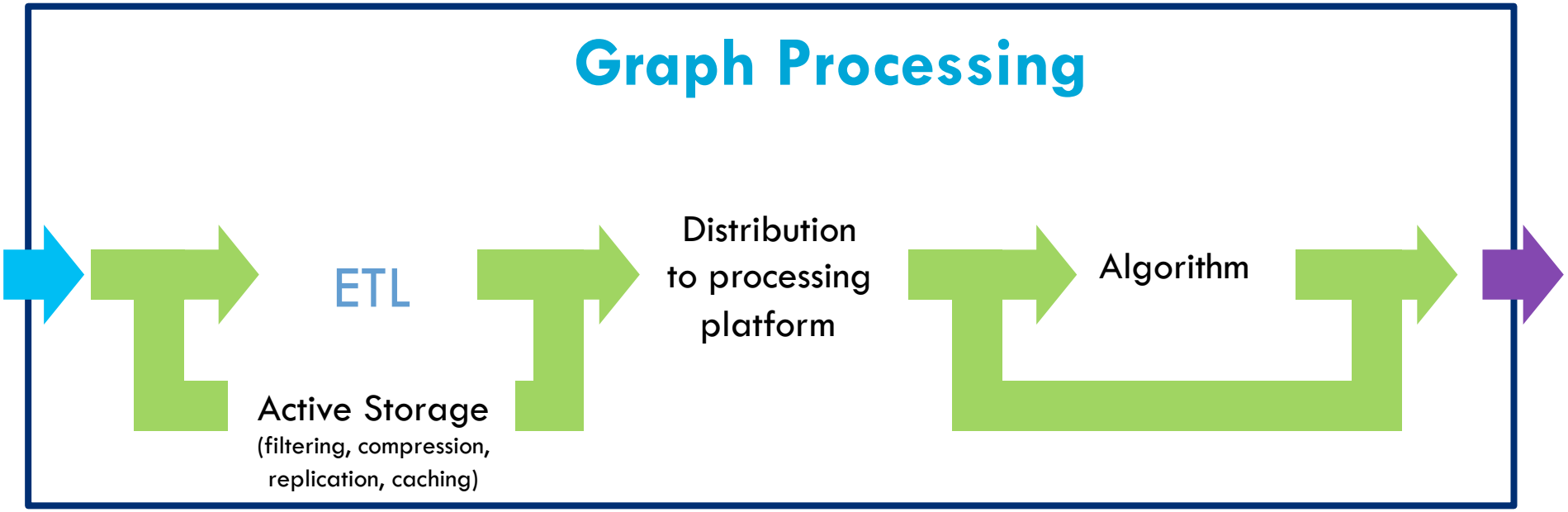
Schedule

- Benchmarking with Graphalytics
  - Get to know the system: step-by-step tutorial
    - Simple example (TBA)
  - Team-work
    - A real-life problem (TBA)
- BONUS: Fine-grained in-depth analysis with Granula
  - Step-by-step tutorial
    - An example (TBA)

Open discussion

Discussion 1

- 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?

SPEC Research Group (RG)

The Research Group of the
Standard Performance Evaluation Corporation

Mission Statement

- Provide a platform for collaborative research efforts in the areas of computer benchmarking and quantitative system analysis
- Provide metrics, tools and benchmarks for evaluating early prototypes and research results as well as full-blown implementations
- Foster interactions and collaborations btw. industry and academia

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 you can program.
  - Graphalytics: use systematic benchmarking

Find us online: graphalytics.ewi.tudelft.nl