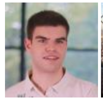
Graphalytics = From Benchmarking to Performance Engineering, leading to Massivizing Graph-Processing Systems

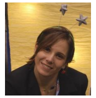
DELFT DATA SCIENCE

COMMIT/

TU Delft



Tim Hegeman, Wing-Lung Ngai, and Stijn Heldens.



Presentation developed jointly with Ana Lucia Varbanescu.

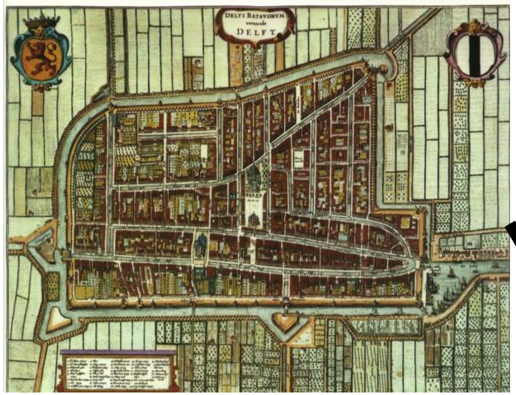
Several slides developed jointly with Yong Guo.

@AIosup

dr. ir. Alexandru Iosup

Distributed Systems Group

(TU) Delft – the Netherlands – Europe



founded 13th century
pop: 100,000

founded 1842
pop: 15,000

Delft

pop: 16.5 M

Barcelona

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

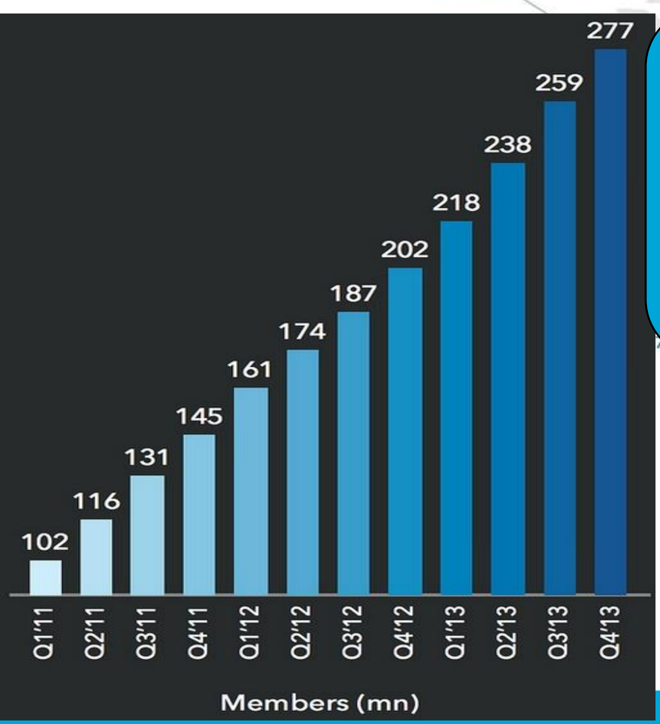
Tens of "specialized LinkedIns": medical, mil, edu, gov, ...

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's Service/Ops Analytics

The State of LinkedIn

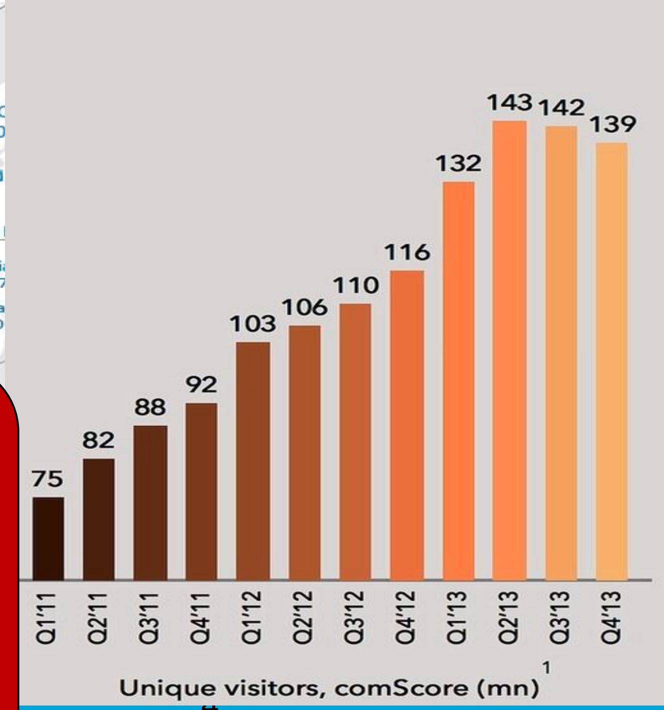
3-4 new users every second



By processing the graph: opinion mining, hub detection, etc. Always new questions about whole dataset.

100+ million questions of customer retention, of (lost) customer influence, of ... Plus new hypotheses to test on whole dataset.

but fewer visitors (and page views)



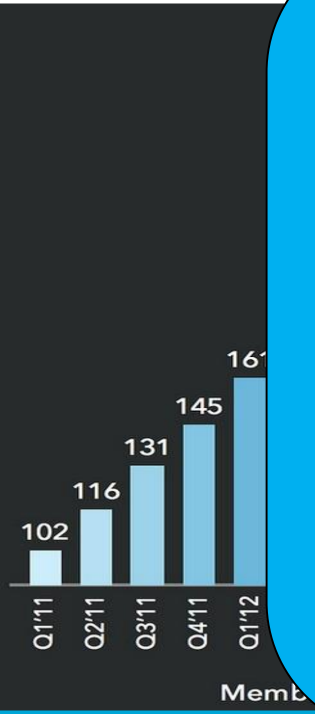
Sources: Vincenzo Cosenza, The State of LinkedIn, http://vincos.it/the-state-of-linkedin/
via Christopher Penn, http://www.shiftcomm.com/2014/02/state-linkedin-social-media-dark-horse/

Why Analytics?

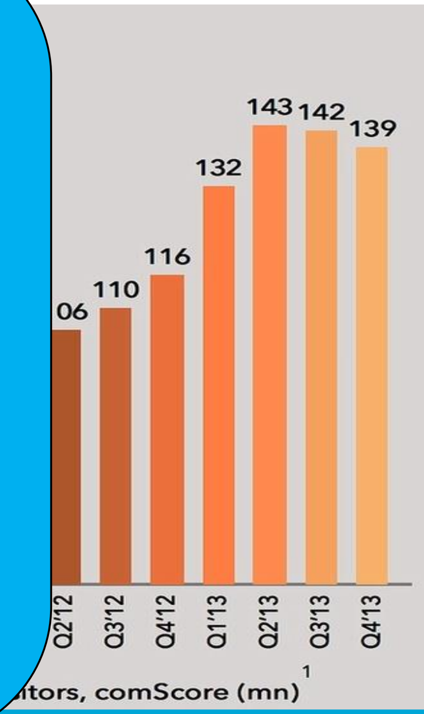
The State of LinkedIn

3-4 new users every second

but fewer visitors (and page views)

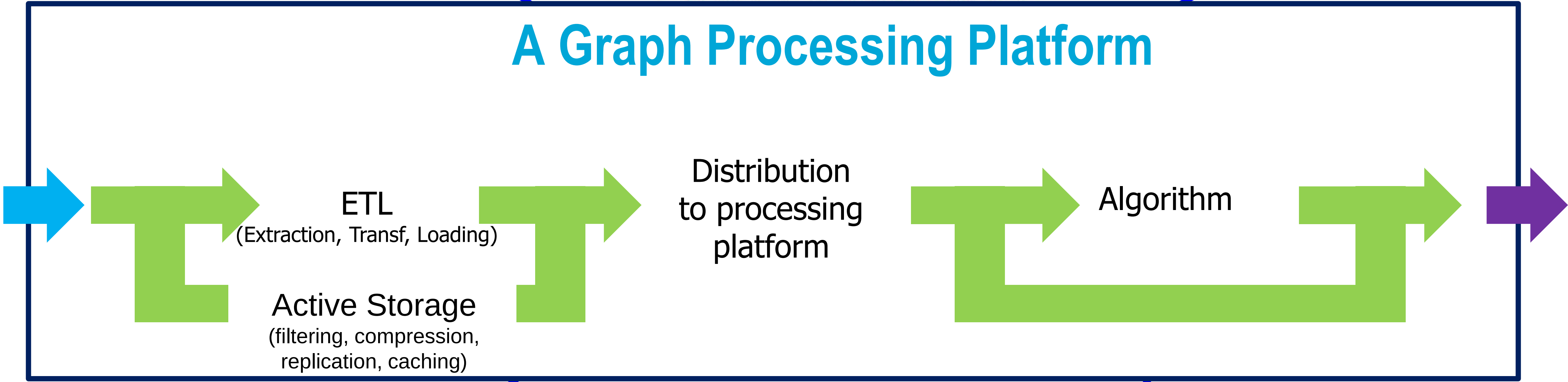


Periodic and/or continuous full-graph analytics



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/

How to do Analytics? Graph Processing @large



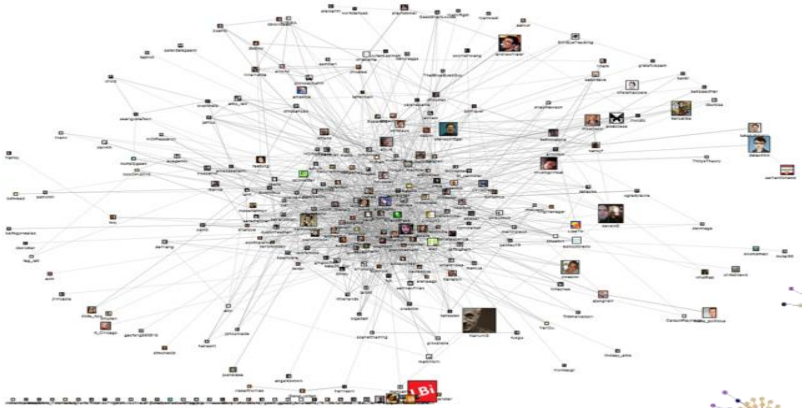
Interactive processing not considered in this presentation.
Streaming not considered in this presentation.

Graph-processing is at the core of our society

The Data Deluge vs. Analytics

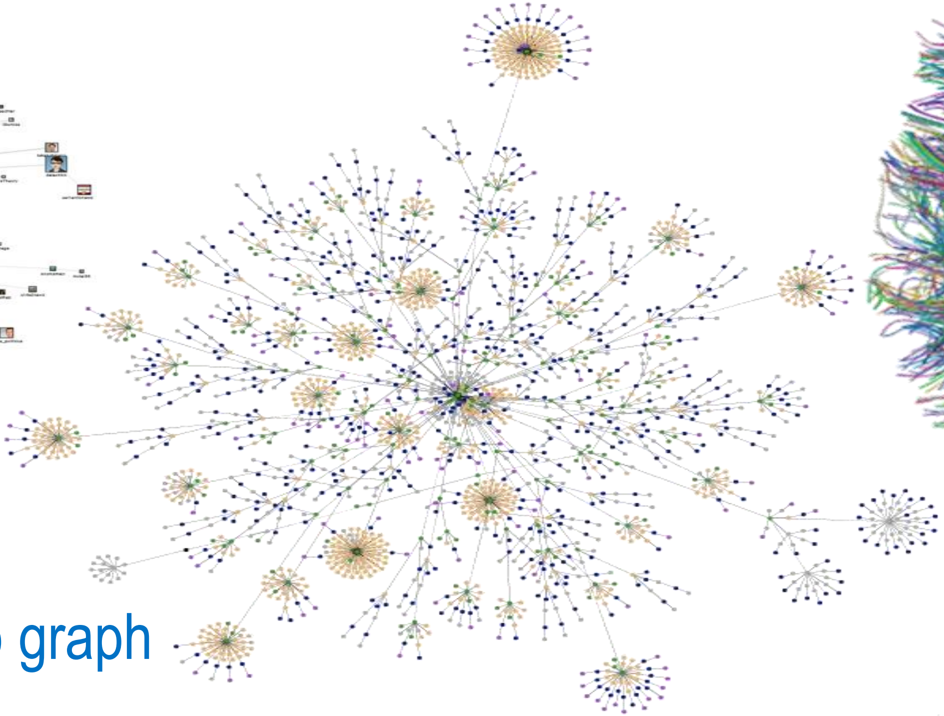
Graph processing @large
Which to select? What to tune? What to re-design?
A performance comparison of graph-processing systems
Take-home message

The data deluge: large-scale graphs



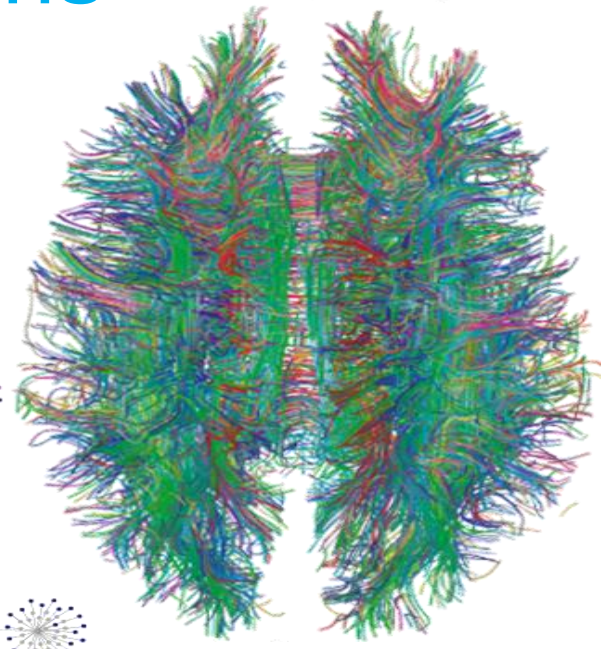
Social network

~1 billion vertices
~100 billion connections



Web graph

~50 billion pages
~1 trillion hyperlinks



Brain network

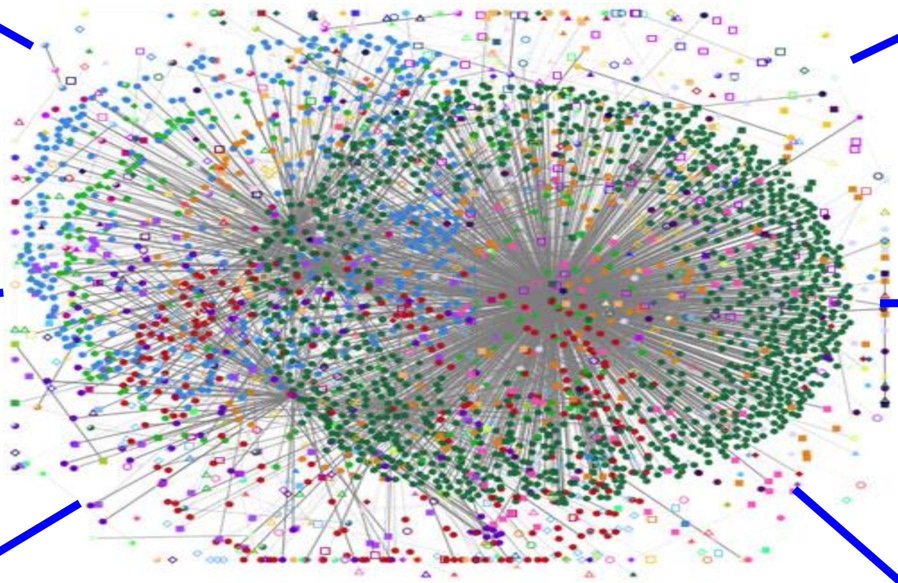
~100 billion neurons
~100 trillion connections

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

The data deluge: graphs everywhere!

400M users

??? edges



270M MAU
200+ avg followers

>54B edges

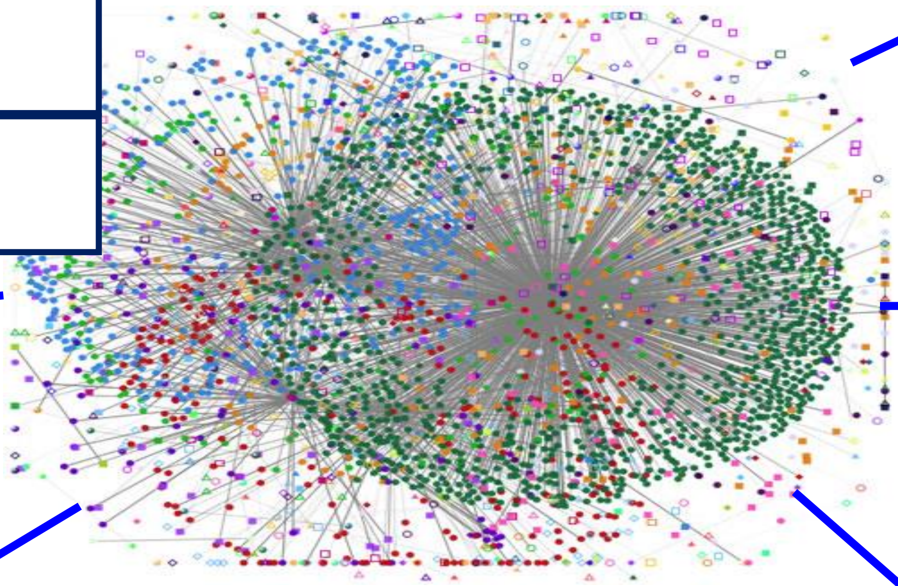
1.2B MAU 0.8B DAU
200+ avg followers

>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 vs. Analytics

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 vs. Analytics

racle 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

The data deluge vs. Analytics

Data-intesive workload

10x graph size → 100x—1,000x slower

Compute-intesive workload

more complex analysis → ?x slower

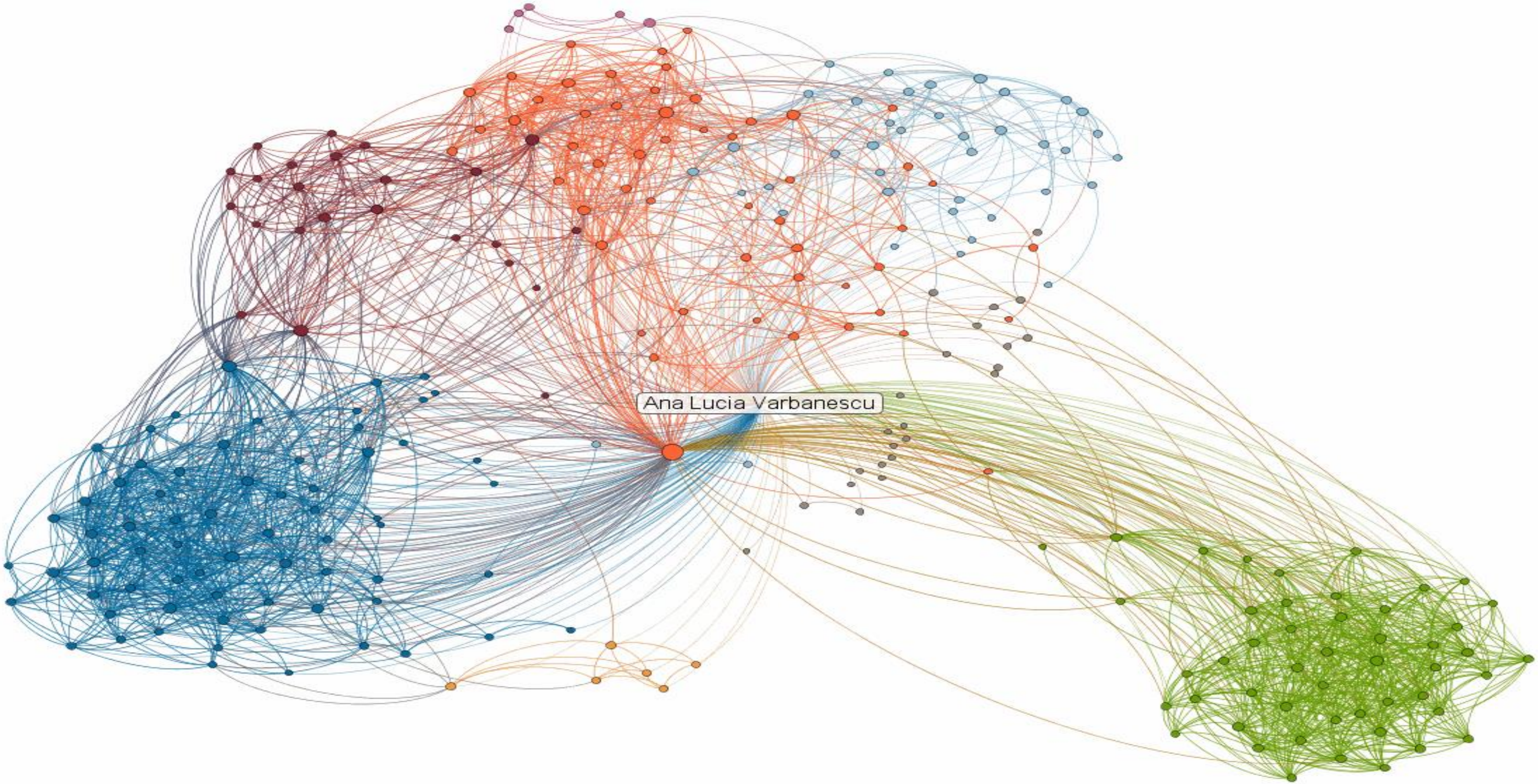
Dataset-dependent workload

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

Supporting multiple users

10x number of users ➔ ????x slower

What would you do to solve this?

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

Supporting multiple users

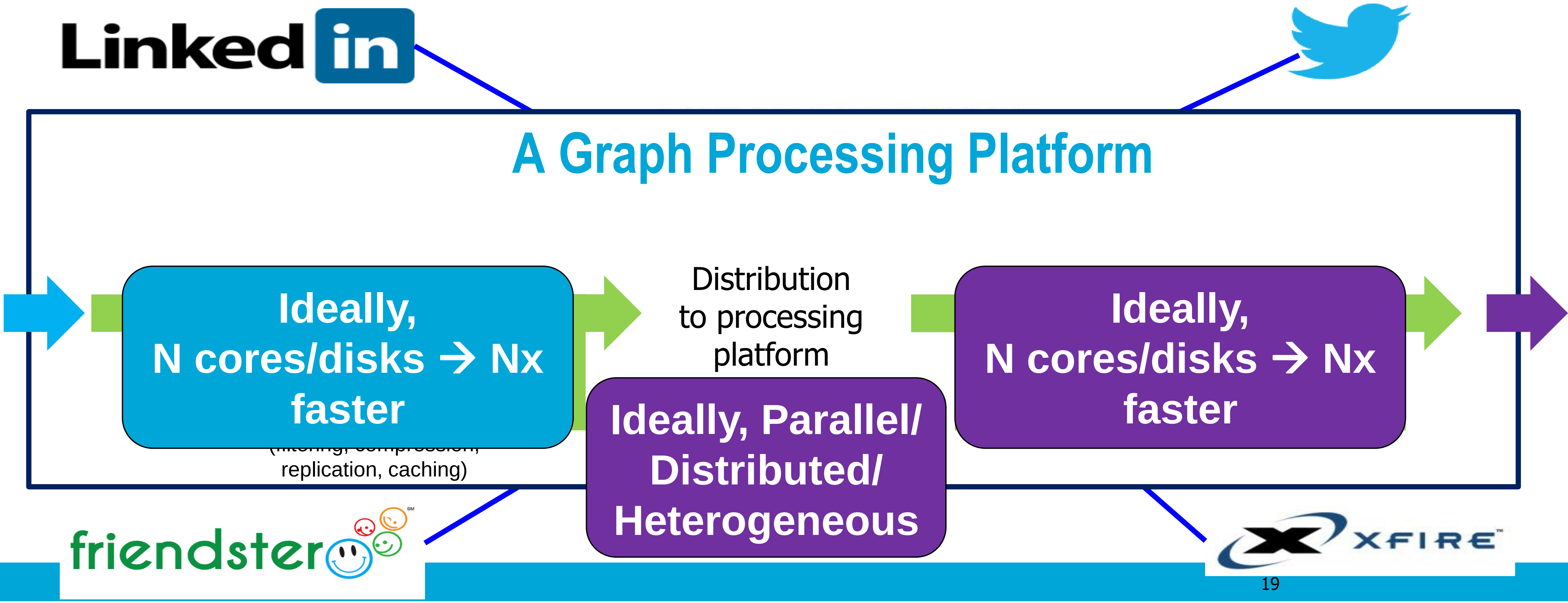
10x number of users ➔ ????x slower

Graph-processing is at the core of our society
The data deluge vs. Analytics

Graph Processing @Large

Which to select? What to tune? What to re-design?
A performance comparison of graph-processing systems
Take-home message

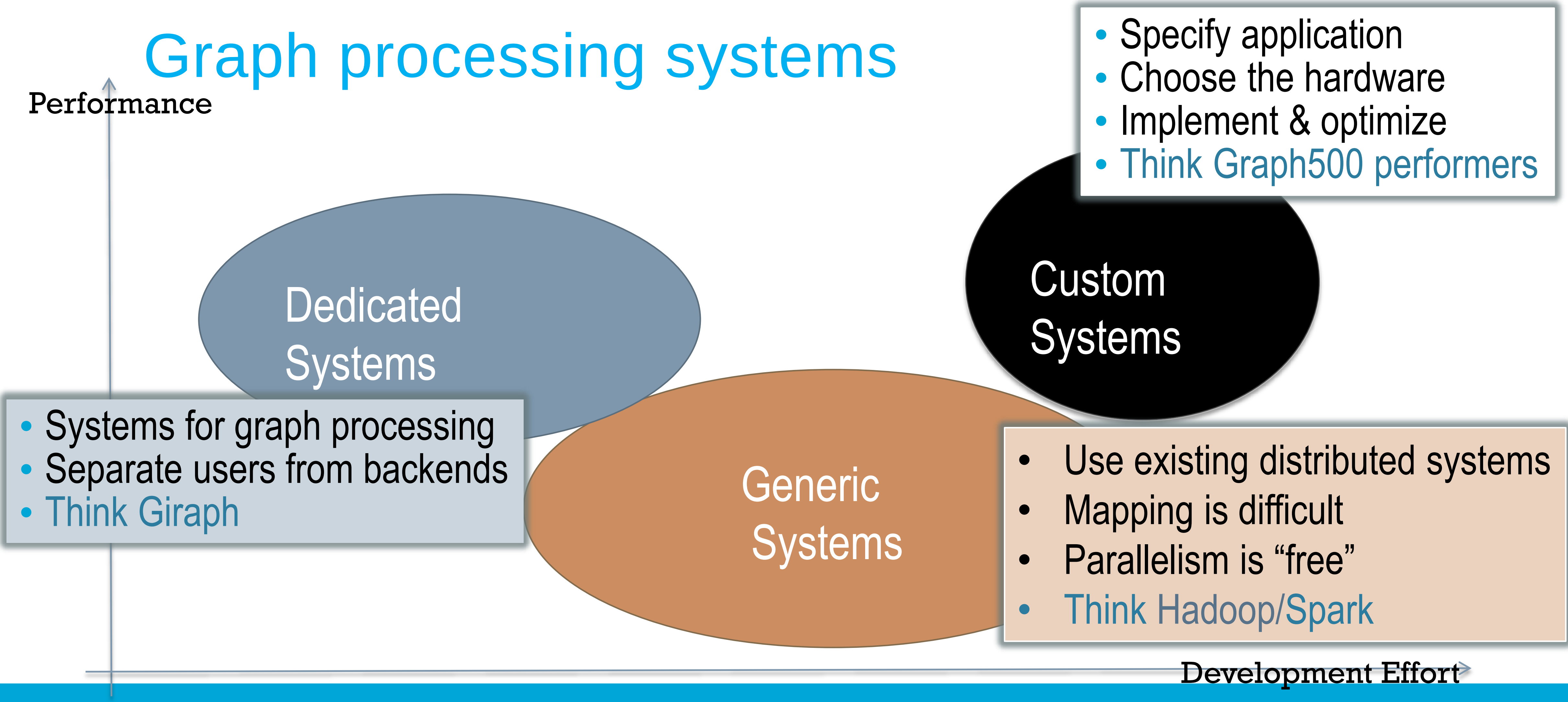
Graph Processing @large



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

Delft University of Technology

Graph processing systems



System diversity (CPU-based)

Dedicated Systems

Generic

ORACLE PGX

Intel Graphmat

Neo4j
the graph database

Trinity

hadoop

YARN

StratoSphere
Above the Clouds

APACHE GIRAPH

GraphLab

GraphX

System diversity (GPU-enabled)

Dedicated Systems

Generic

medusa-gpu

Medusa: Simplified Graph Processing on GPUs

NetSysLab

mapgraph Beta

Massively Parallel Graph processing on GPUs

TOTEM

Gunrock

High-performance Graph Primitives on GPU

VertexAPI2

Graph-Processing Platforms

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

Intel Graphmat

2

Which to choose?

What to tune?

What to re-design?

Graph-processing is at the core of our society
The data deluge vs. Analytics
Graph processing @large

Graphalytics:
Which system to select?
What to tune? What to re-design?

A performance comparison of graph-processing systems
Take-home message

Benchmarking, but … What Is a Benchmark?

Benchmark definition must include:

- *Data schema*: data representation
- *Workloads*: formalize datasets + algorithms
- *Performance metrics:* from performance to non-traditional to cost-related
- *Execution rules:* how to run the benchmark tests, parameter values, etc.

Desirable support for stakeholders:

- Live addition of results
- Curation of added results
- Auditing results

What is the performance of graph-processing platforms?

Metrics Diversity

Graph Diversity

Algorithm Diversity

- Graph500
  - Single application (BFS), Single class of synthetic datasets. @ISC16: future diversification.
- Few existing platform-centric comparative studies
  - Prove the superiority of a given system, limited set of metrics
- GreenGraph500, GraphBench, XGDBench
  - Issues with representativeness, systems covered, metrics, …

What is the performance of graph-processing platforms?

Metrics Diversity

Graph Diversity

Algorithm Diversity

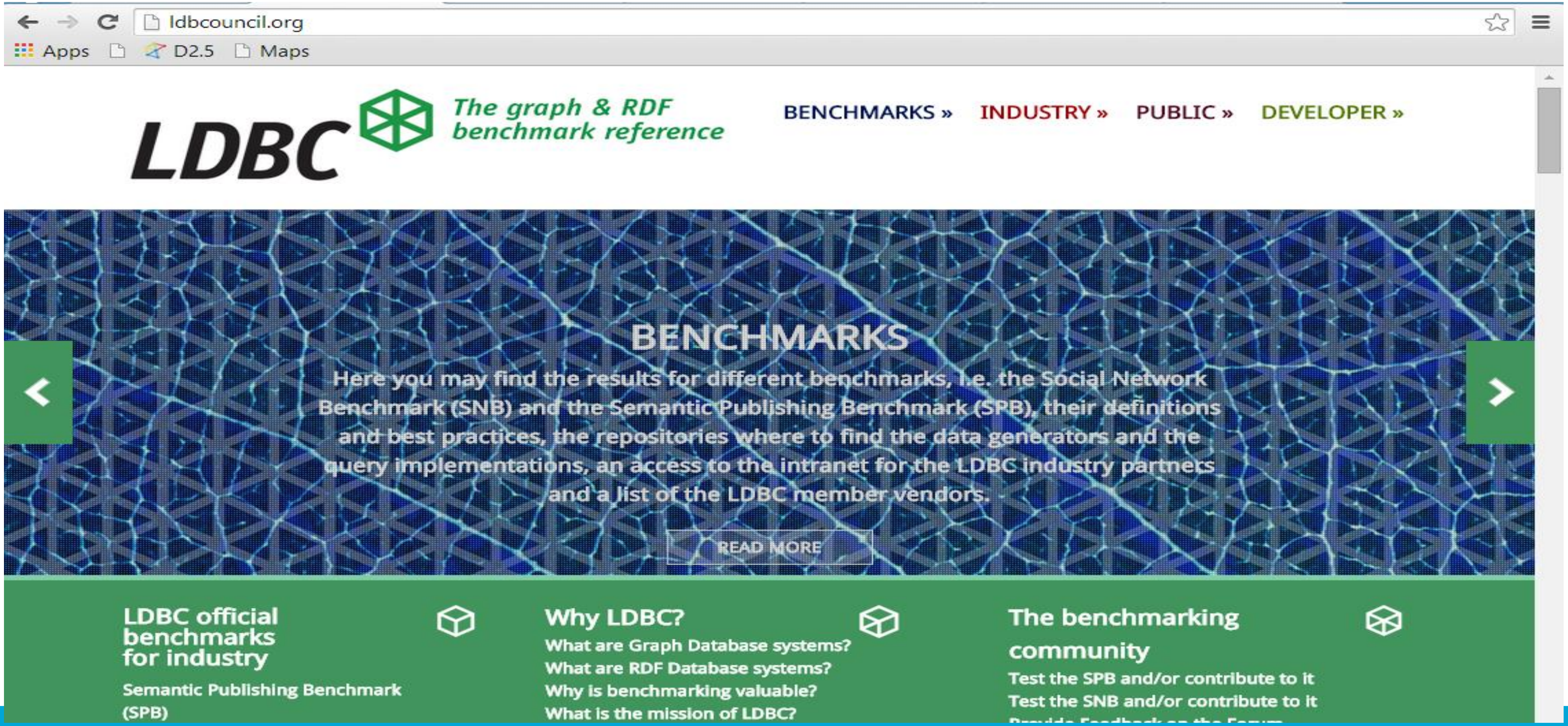
Graphalytics = comprehensive benchmarking suite for graph processing across many platforms

LDBC The graph & RDF benchmark reference

Why is “LDBC” here?

TU Delft
Delft University of Technology

ldbcouncil.org



Delft University of Technology

Graphalytics, in a nutshell

- An LDBC benchmark*
- Advanced benchmarking harness
- Many classes of algorithms used in practice
- Diverse real and synthetic datasets
- Diverse set of experiments representative for practice
- Granula for manual choke-point analysis
- Modern software engineering practices
- Supports many platforms
- Enables comparison of community-driven and industrial systems

LDBC The graph & RDF benchmark reference

UPC CWI

Universiteit van Amsterdam

DELFT DATA SCIENCE

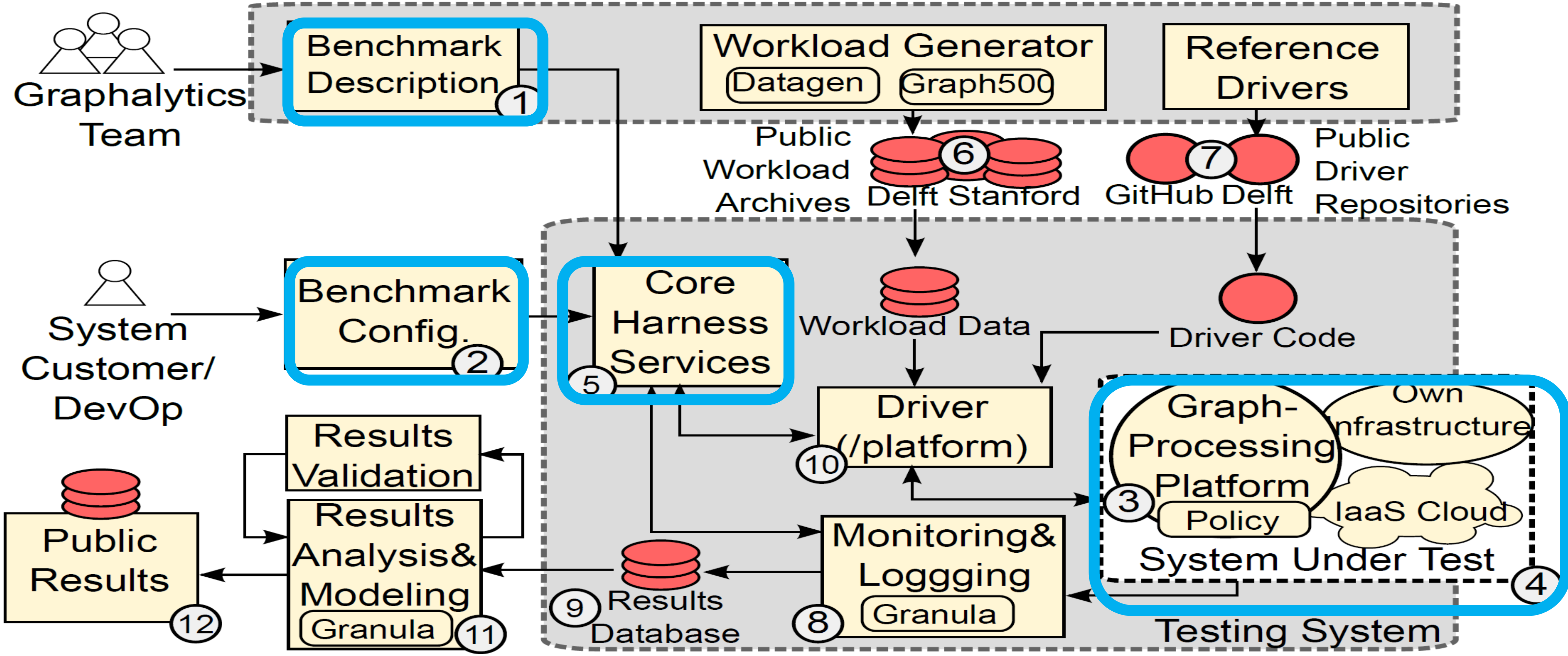
intel

IBM HUAWEI

ORACLE

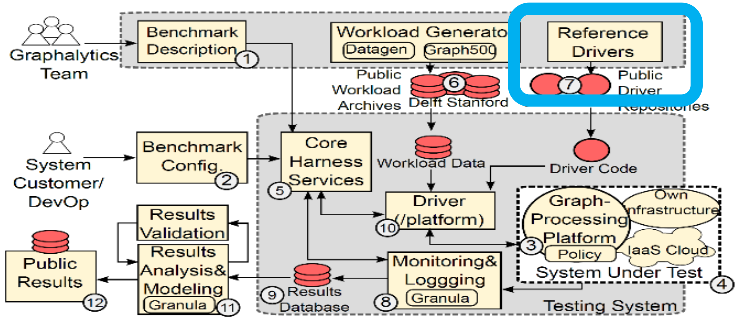
http://graphalytics.ewi.tudelft.nl
https://github.com/tudelft-atlarge/graphalytics/

Benchmarking Harness



Iosup et al. LDBC Graphalytics: A Benchmark for Large Scale Graph Analysis on Parallel and Distributed Platform, VLDB'16.

Graphalytics = Representative Classes of Algorithms and Datasets



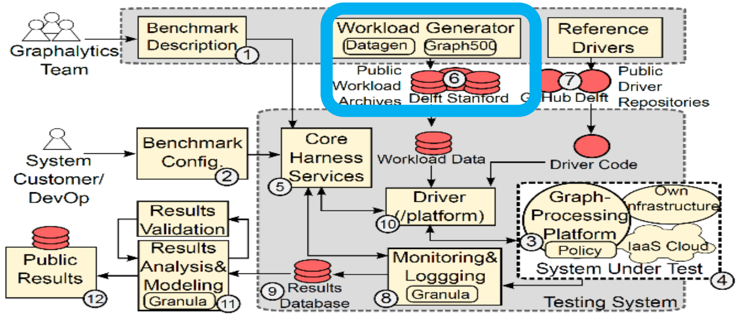
- 2-stage selection process of **algorithms** and datasets

Class	Examples	%
Graph Statistics	Diameter, Local Clust. Coeff, PageRank	20
Graph Traversal	BFS, SSSP, DFS	50
Connected Comp.	Reachability, BiCC, Weakly CC	10
Community Detection	Clustering, Nearest Neighbor, Community Detection w Label Propagation	5
Other	Sampling, Partitioning	<15

+ weighted graphs: Single-Source Shortest Paths (~35%)

Guo et al. How Well do Graph-Processing Platforms Perform? An Empirical Performance Evaluation and Analysis, IPDPS'14.

Graphalytics = Distributed Graph Generation w DATAGEN



- Rich set of configurations
- More diverse degree distribution than Graph500
- Realistic properties, e.g., clustering coefficient and assortativity

Graphalytics

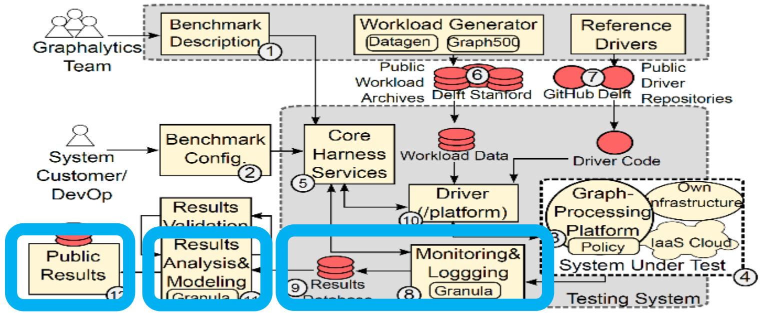
Person Generation → Edge Generation → “Knows” graph serializatio n → Activity Generation → Activity serialization

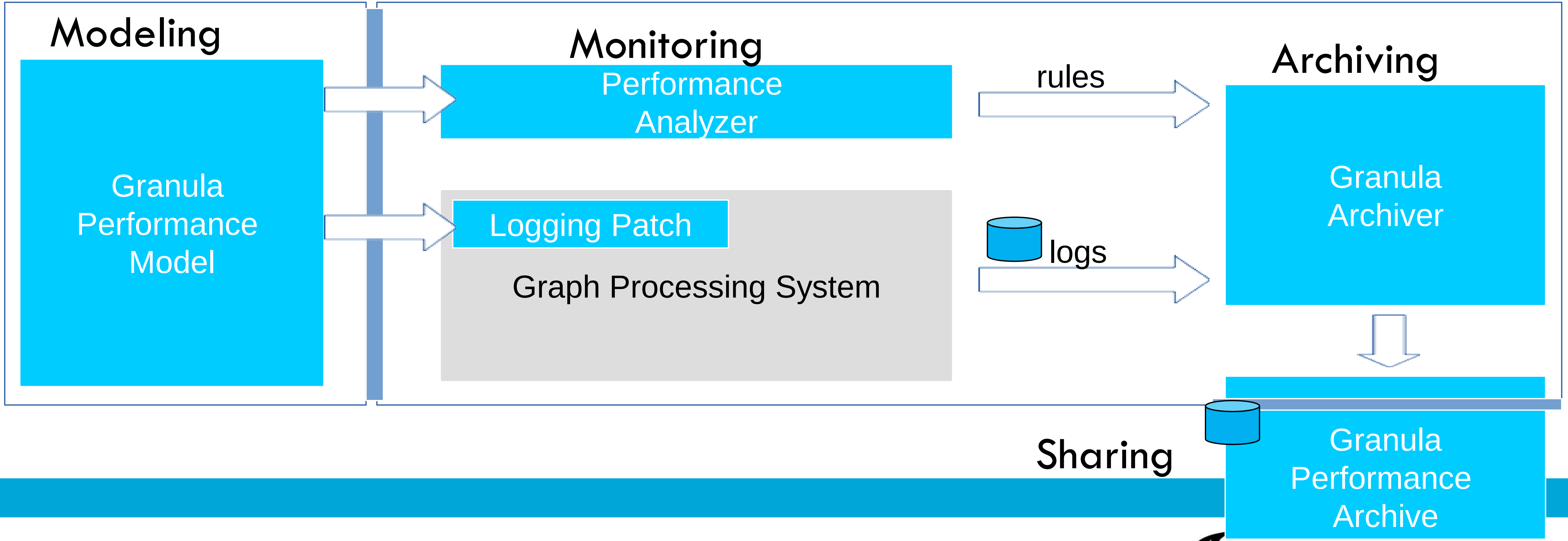
Level of Detail

Graphalytics = Diverse Automated Experiments

Category	Experiment	Algo.	Data	Nodes/ Threads	Metrics
Baseline	Dataset variety	BFS,PR	All	1	Run, norm.
	Algorithm variety	All	R4(S), D300(L)	1	Runtime
Scalability	Vertical vs. horiz.	BFS, PR	D300(L), D1000(XL)	1—16/1—32	Runtime, S
	Weak vs. strong	BFS, PR	G22(S)—G26(XL)	1—16/1—32	Runtime, S
Robustness	Stress test	BFS	All	1	SLA met
	Variability	BFS	D300(L), D1000(L)	1/16	CV
Self-Test	Time to run/part	--	Datagen	1—16	Runtime

Graphalytics = Portable Performance Analysis w Granula





Minimal code invasion + automated data collection at runtime + portable archive (+ web UI) → portable bottleneck analysis

Delft University of Technology

Graphalytics = Modern Software Engineering Process

- Graphalytics code reviews
  - Internal release to LDBC partners (first, Feb 2015; last, Feb 2016)
  - 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/

Implementation status

G=validated, on GitHub
V=validation stage

	MR 2	Gi-raph	Graph X	Power Graph	Graph Lab	Neo4j	PGX.D	Graph Mat	Open G	TOTEM	Map Graph	Medusa
LCC	G	G	G	G	G	G	--	G	G	--	--	--
BFS	G	G	G	G	G	G	G	G	G	V	V	V
WCC	G	G	G	G	G	G	G	G	G	V	V	V
CDLP	G	G	G	G	G	G	G	G	G	--	--	--
P'Rank	--	G	G	G	V	--	G	G	G	V	V	V
SSSP	--	G	G	G	--	--	G	G	G	--	--	--

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

Implementation status

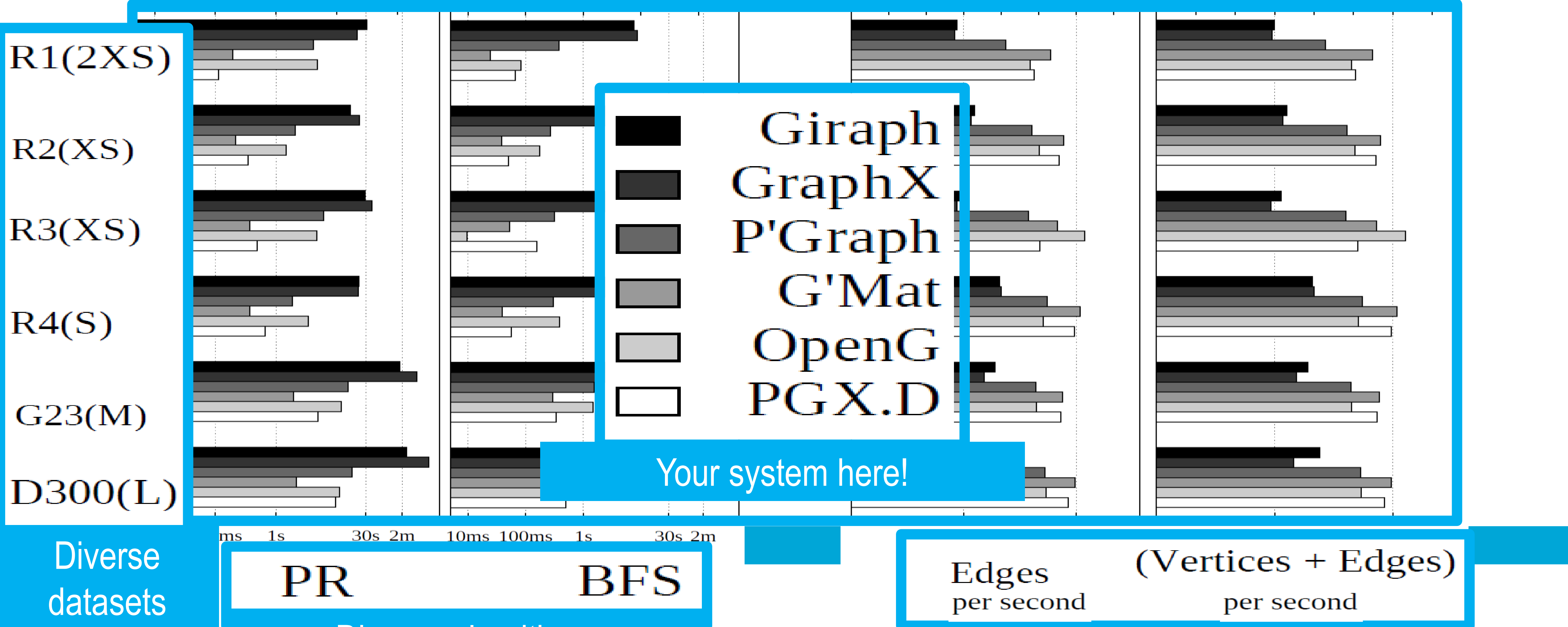
G=validated, on GitHub
V=validation stage

	MR 2	Gi-raph	Graph X	Power Graph	Graph Lab	Neo4j	PGX. D	Graph Mat	Open G	TOTEM	Map Graph	Medusa
LCC	G	G	G	G	G	G	--	G	G	--	--	--
BFS	G	G	G	G	G	G	G	G	G	V	V	V
WCC	G	G	G	G	G	G	G	G	G	V	V	V
CDLP	G	G	G	G	G	G	G	G	G	--	--	--
P’Rank	--	G	G	G	V	--	G	G	G	V	V	V
SSSP	--	G	G	G	--	--	G	G	G	--	--	--

Benchmarking and tuning performed by vendors

Graphalytics Capabilities: An Example

Graphalytics enables deep comparison of many systems at once, through diverse experiments and metrics

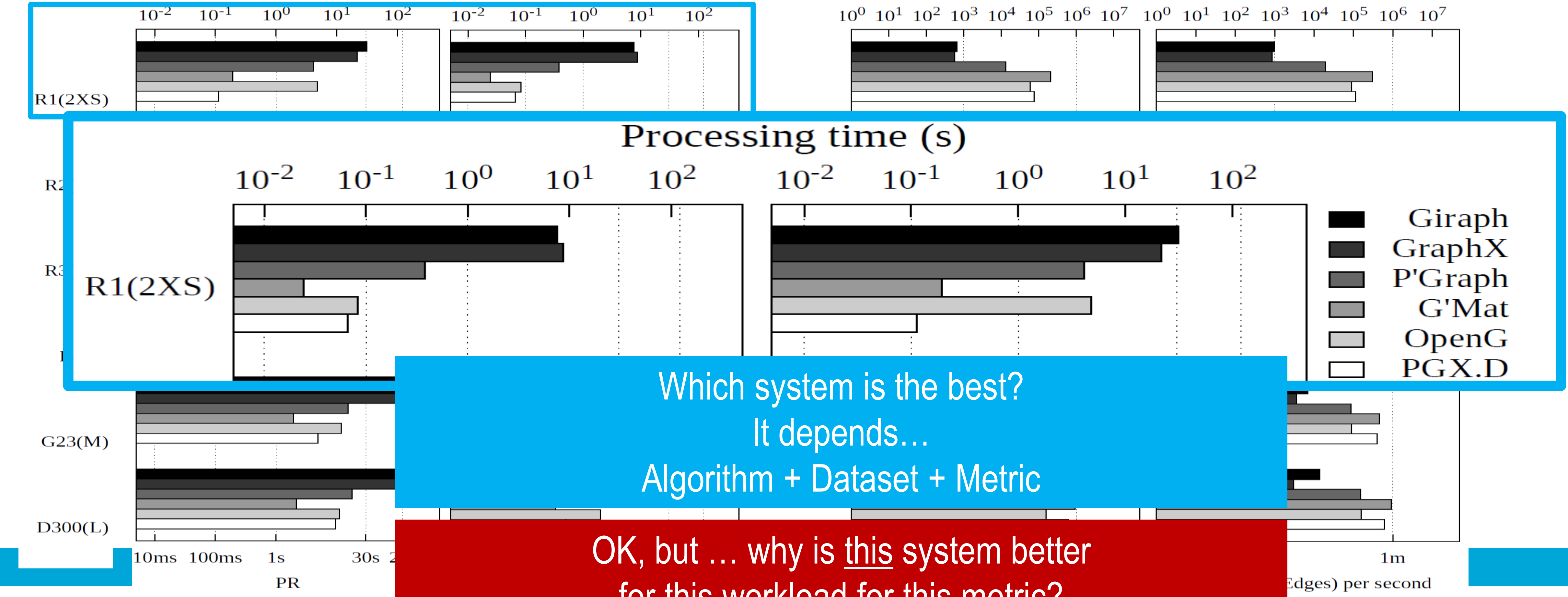


Diverse datasets

Diverse algorithms

Diverse metrics

Processing time (s) + Edges[+Vertices]/s



Which system is the best?
It depends…
Algorithm + Dataset + Metric

OK, but … why is this system better
for this workload for this metric?

Graph-processing is at the core of our society
The data deluge vs. Analytics
Graph processing @large

Graphalytics:
Which system to select?
What to tune? What to re-design?

A performance comparison of graph-processing systems
Take-home message

Coarse-grained vs Fine-grained Evaluation (1)

Coarse-grained Method

system viewed as a black-box

Algorithms, Datasets, Resources

Graph processing system

Coarse-grained metrics

(Overall Execution Time)

Fine-grained Method

system viewed as a white-box

Algorithms, Datasets, Resources

IO operations

Processing

Overheads

Fine-grained metrics

(Stage 3 time, straggler tasks)

Fine-grained evaluation method is more comprehensive

Coarse-grained vs Fine-grained Evaluation (2)

Abstract

Coarse-grained Method

knowledge at conceptual level

Graph Processing Systems

Distributed Infrastructure

Few, coarse-grained results

Granular

Fine-grained Method

knowledge at technical level

hadoop

GraphLab

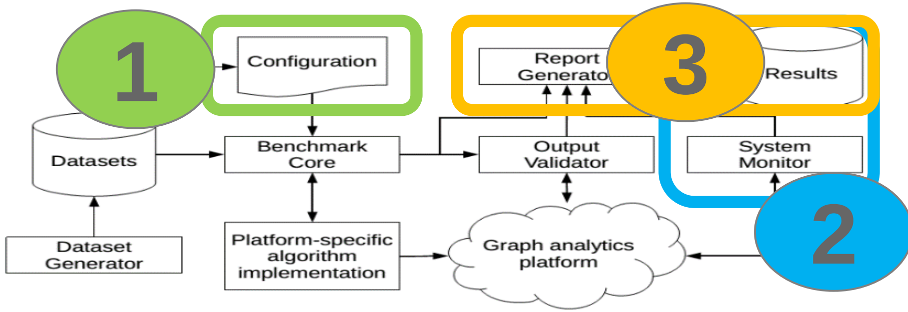
APACHE GIRAPH

Oracle Labs PGX

Many, fine-grained results

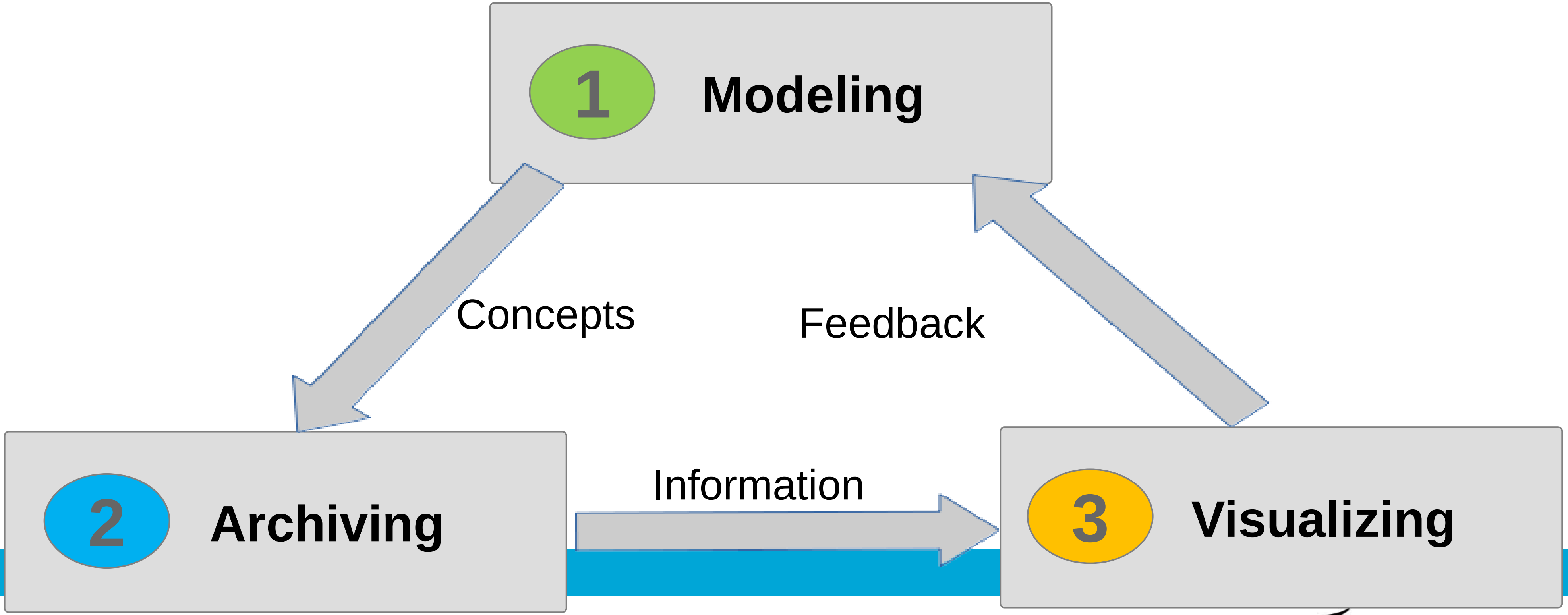
Fine-grained evaluation method is more comprehensive
… but more time-consuming, esp. to implement

Graphalytics:
Granula Overview



Granular

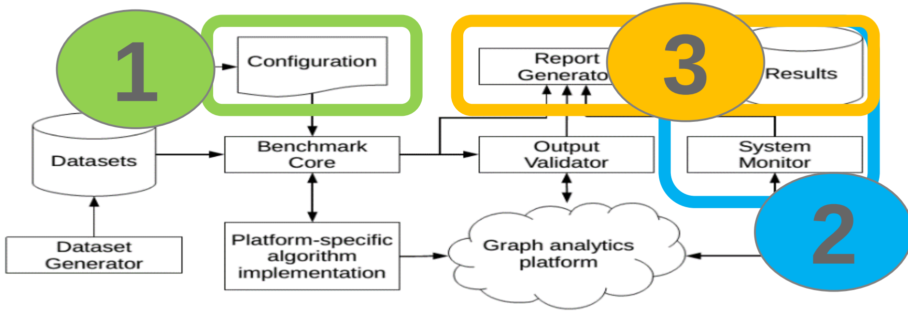
Fine-grained Method

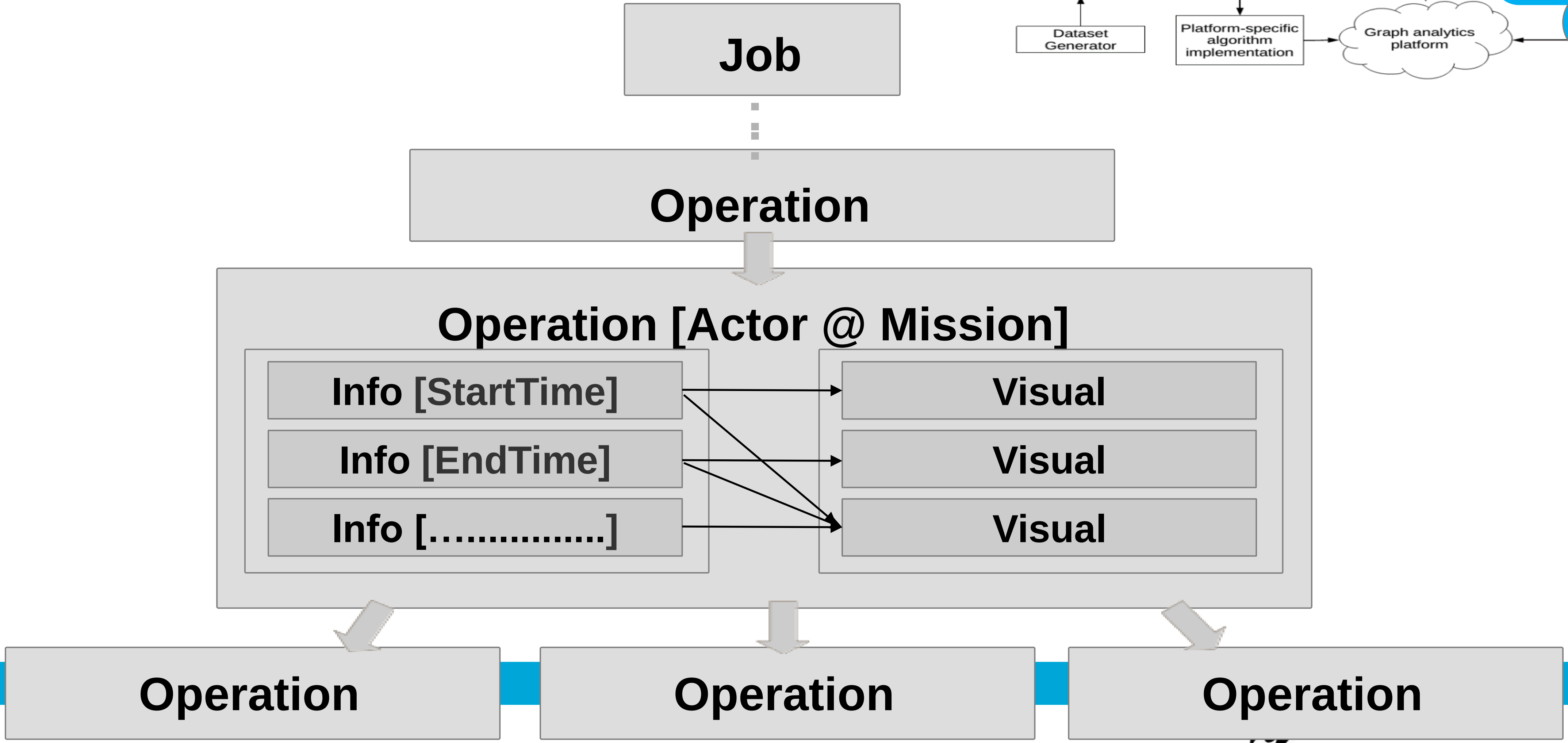


Delft University of Technology

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

1 Granula Modeller



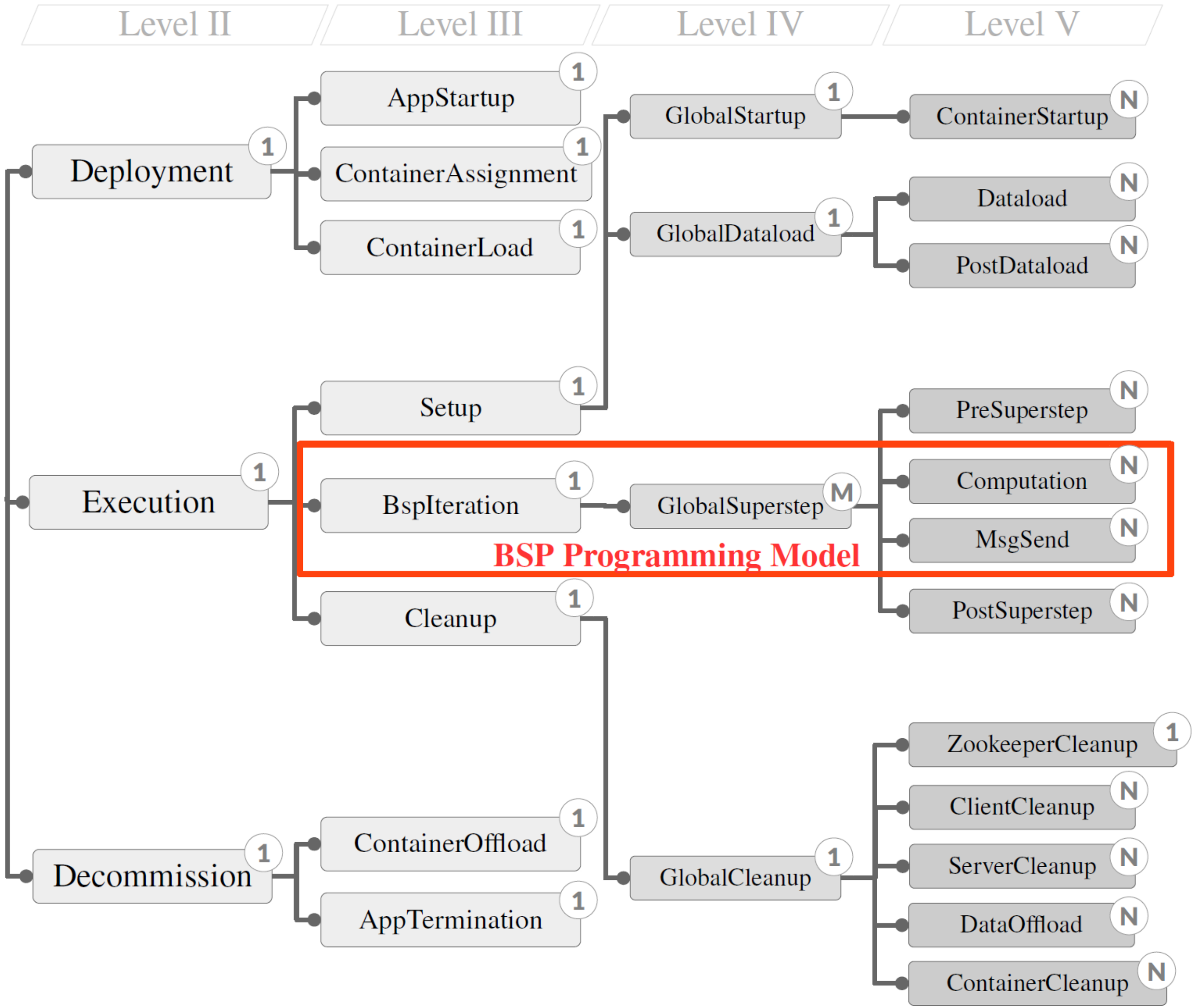


Time-consuming, expert-only,

Done only once per platform, but incrementally

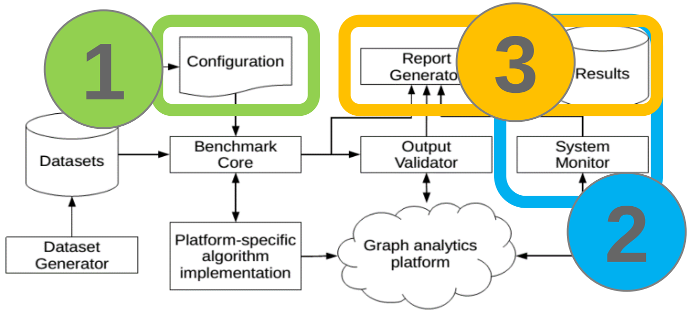
1 Granula Modeller

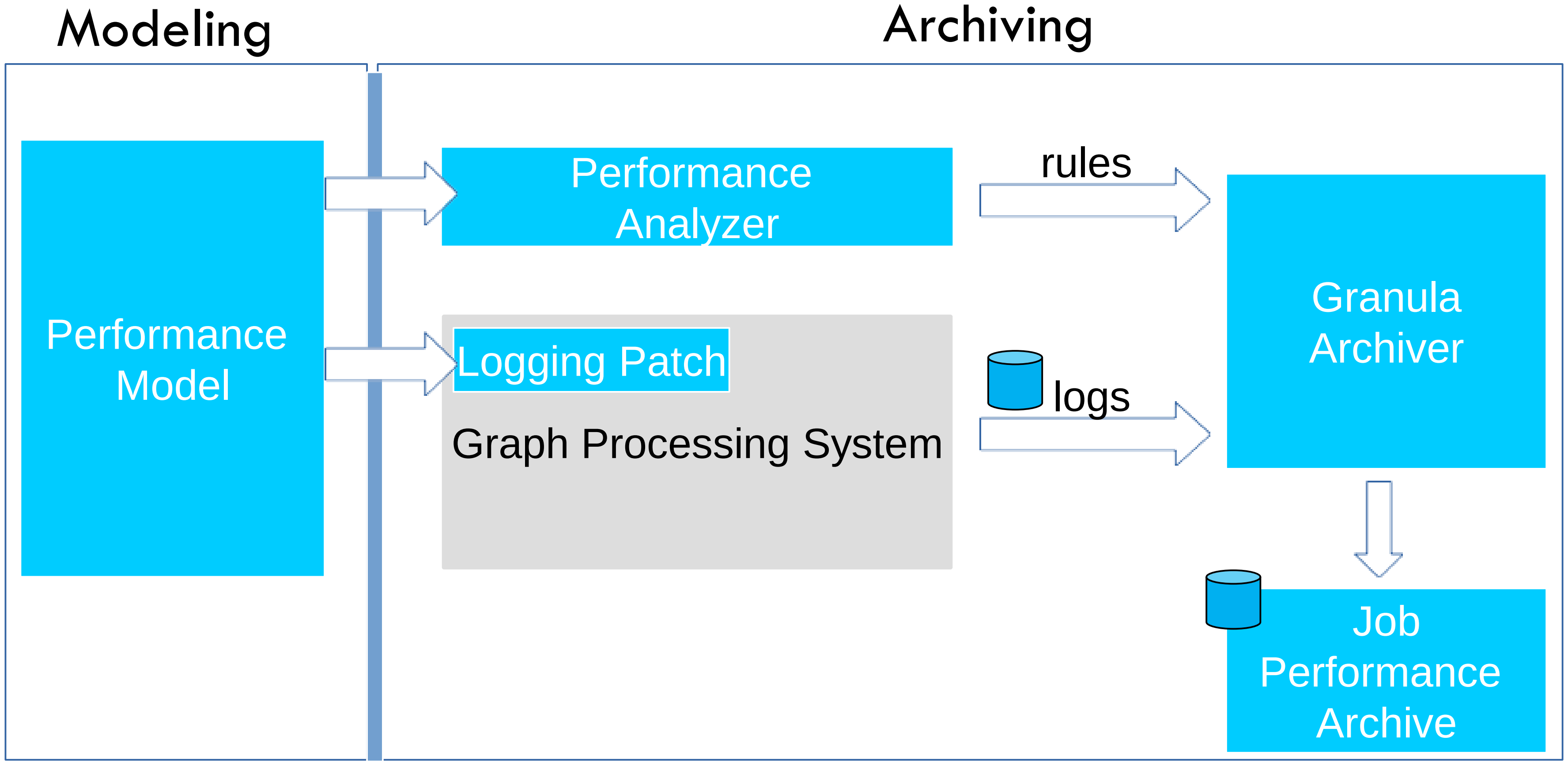
Incremental Model of Graph-Processing in Giraph



TU Delft
Delft University of Technology

2 Granula Archiver

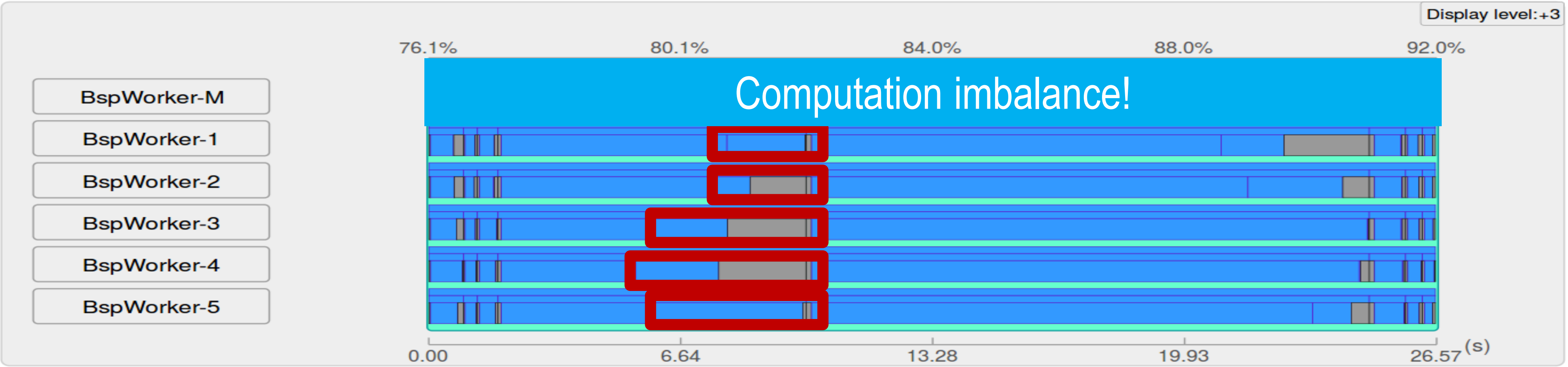


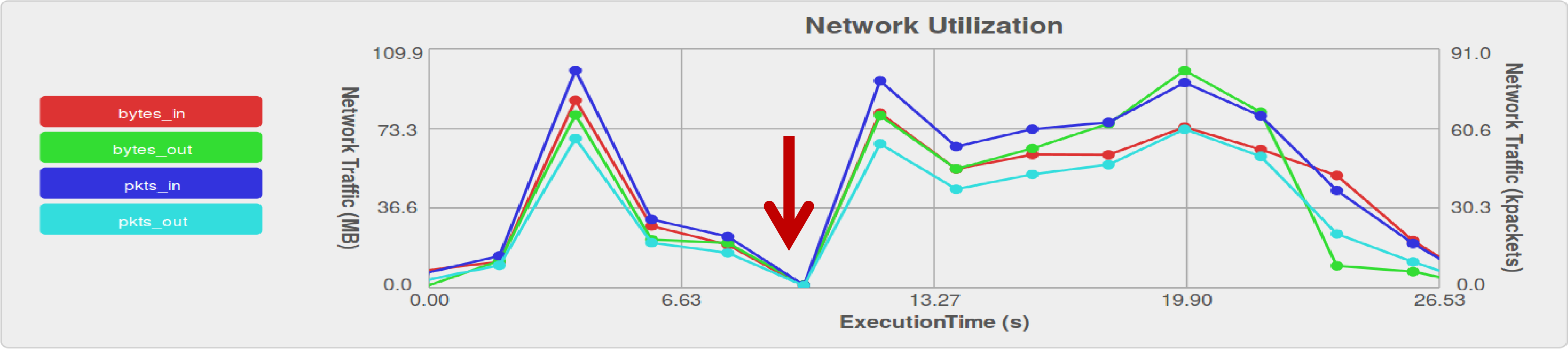


Time-consuming, minimal code invasion,
automated data collection at runtime, portable archive

3 Granula Visualizer

Portable choke-point analysis for everyone!



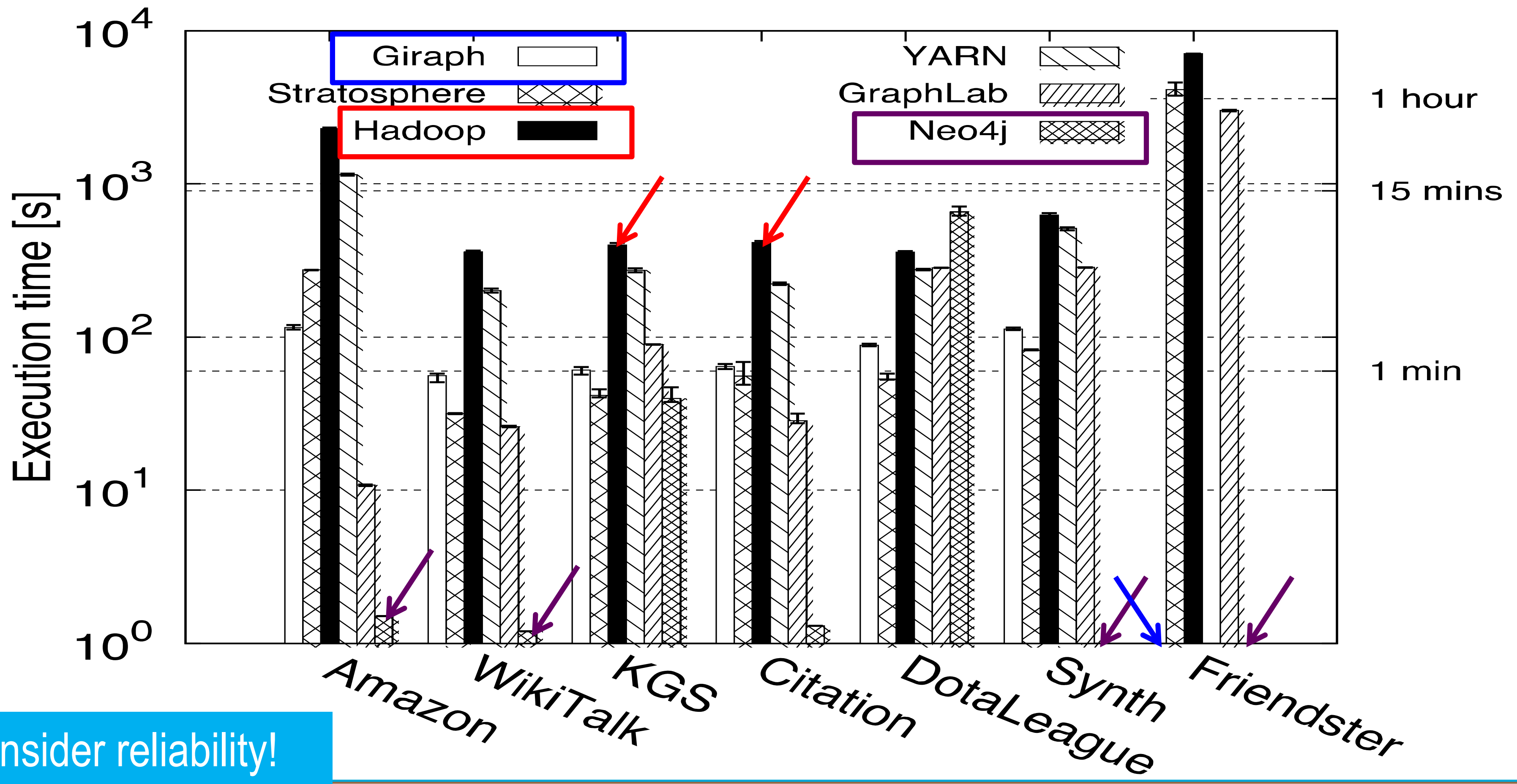


Graph-processing is at the core of our society
The data deluge vs. Analytics
Graph processing @large
Graphalytics: Which system to select? What to tune? What to re-design?

A Performance Comparison of Graph-Processing Systems

Take-home message

BFS Algorithm: All CPU Platforms, All Datasets

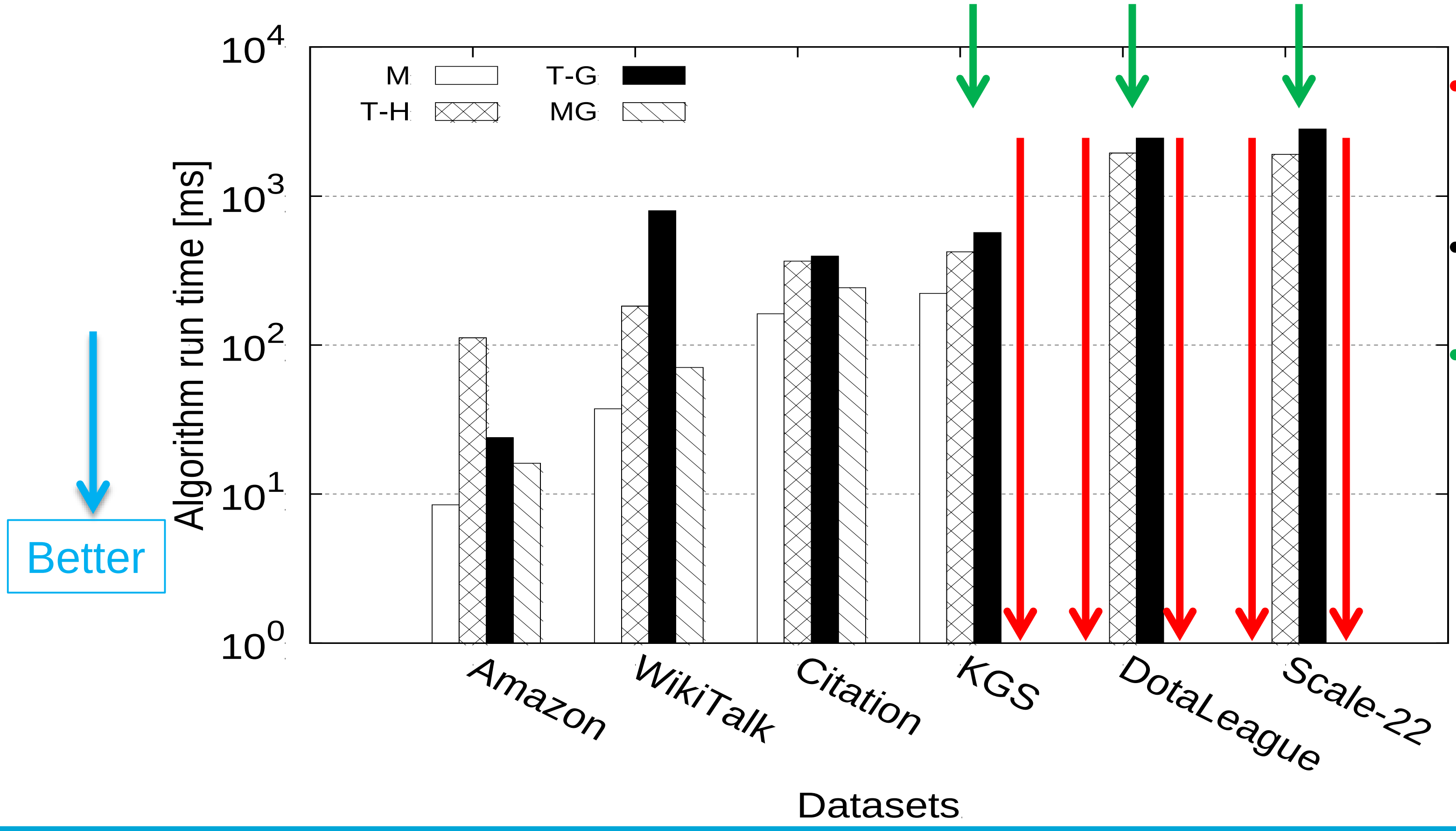


Better

Always consider reliability!

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

PageRank Algorithm: All GPU Platforms, Datasets

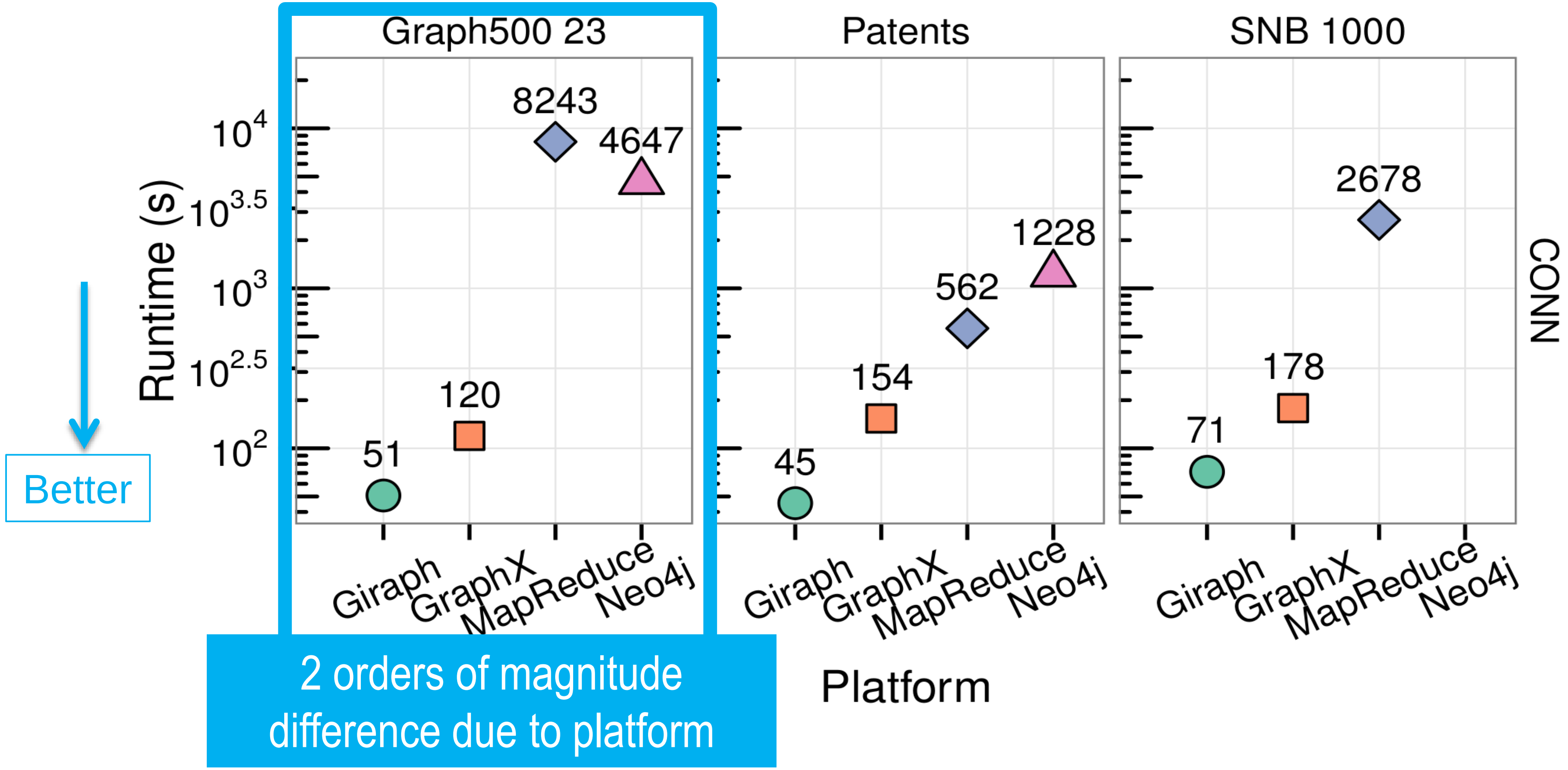


- Medusa, MapGraph fail on larger datasets, with MG failing earlier
- Medusa better for small datasets
- Totem the only system able to process all tried datasets

Always consider reliability!

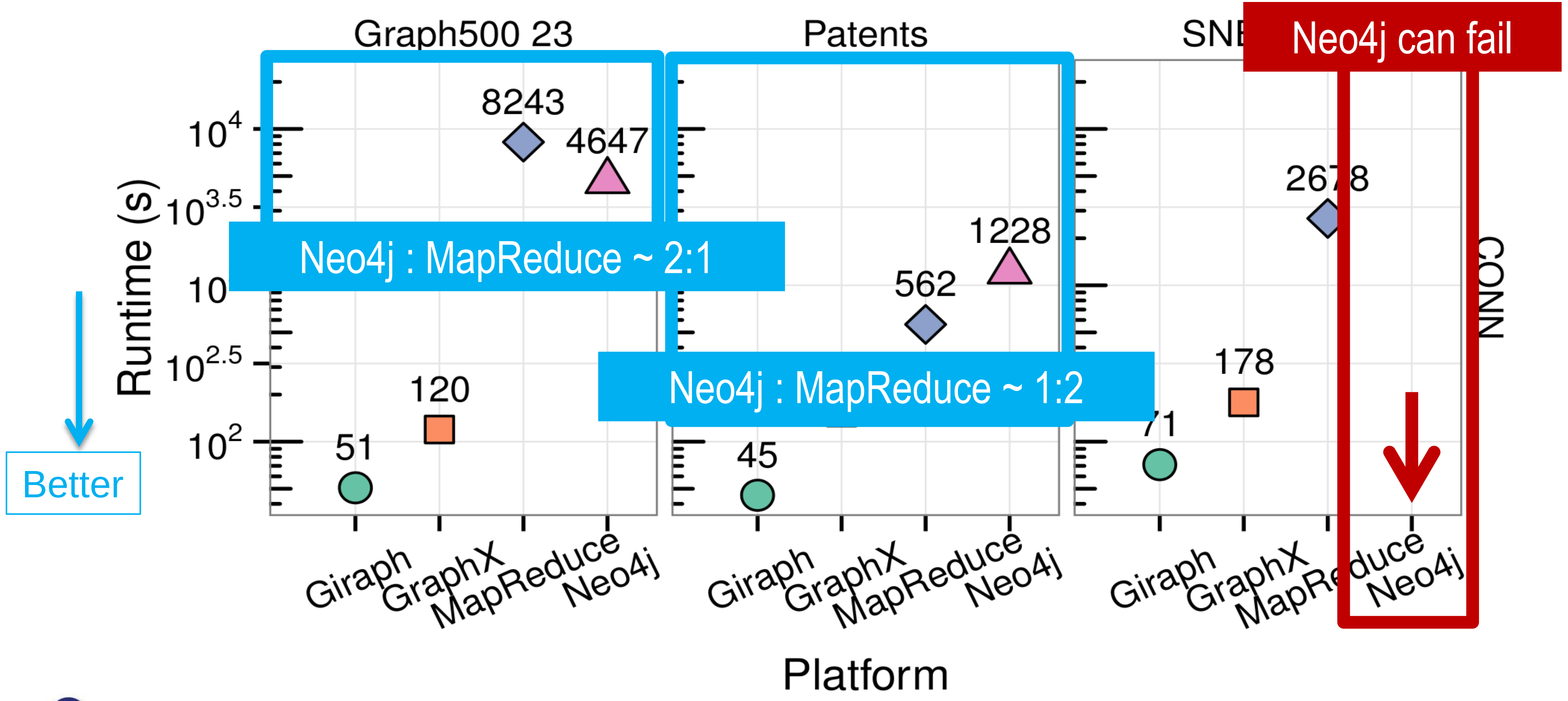
Guo et al. How Well do Graph-Processing Platforms Perform? An Empirical Performance Evaluation and Analysis, IPDPS'14.

Runtime: The Platform Has Large Impact



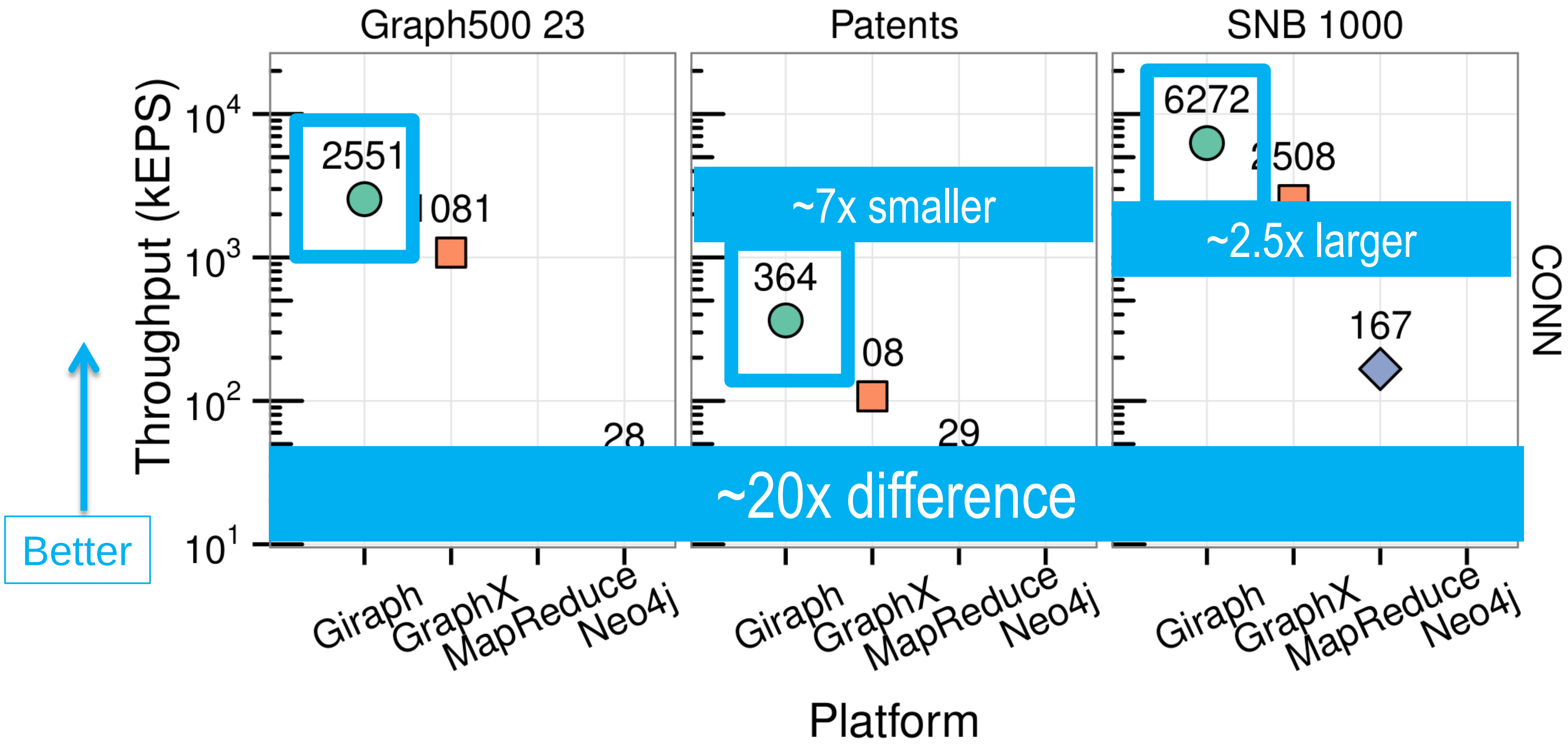
M. Capota et al., Graphalytics: A Big Data Benchmark for Graph-Processing Platforms. SIGMOD GRADES 2015

Runtime: The Dataset Has Large Impact



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

Throughput: The Dataset Has Large Impact

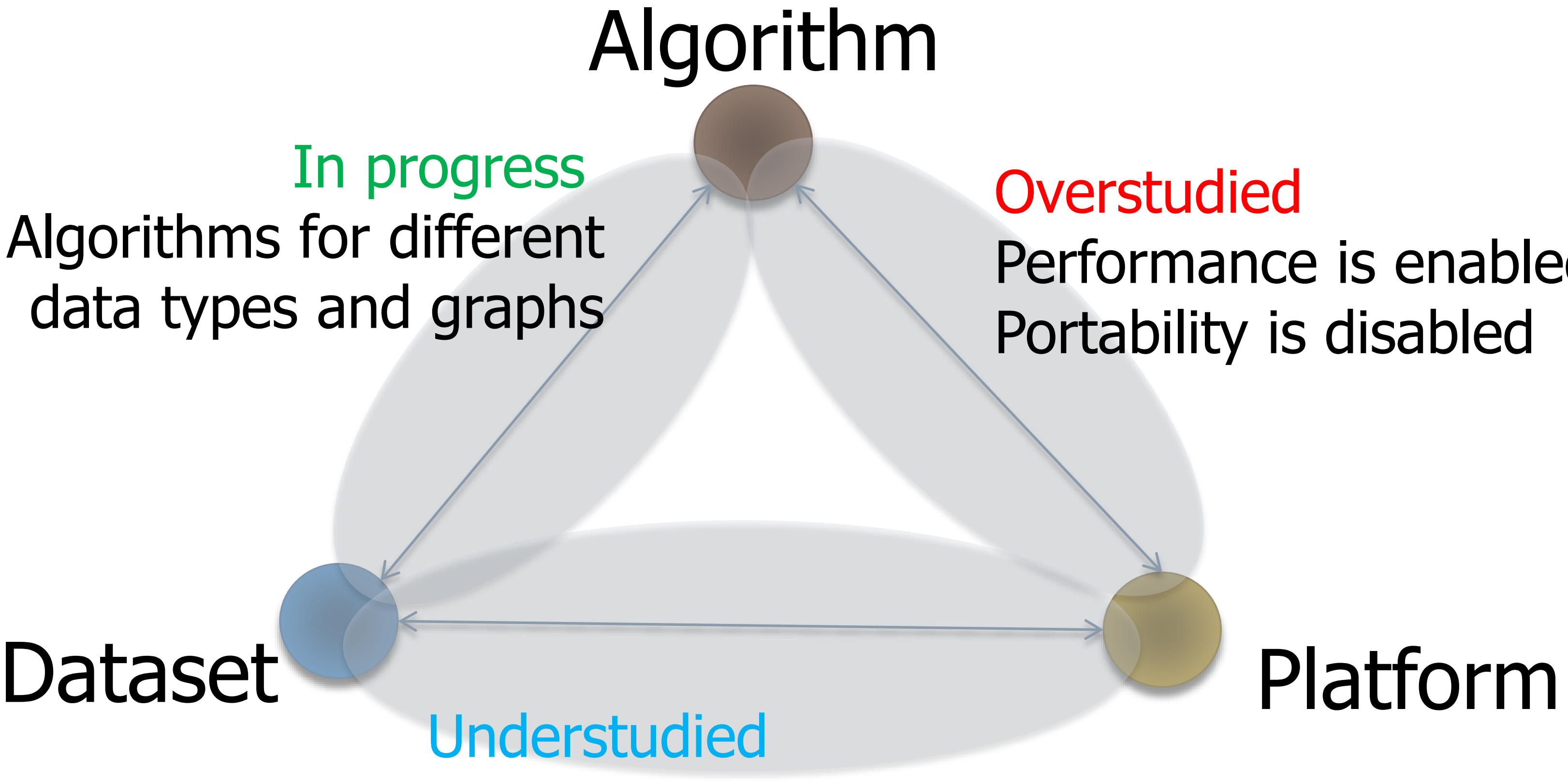


M. Capota et al., Graphalytics: A Big Data Benchmark for Graph-Processing Platforms. SIGMOD GRADES 2015

The Platform-Algorithm-Dataset (PAD) Triangle

for Performance Engineering of Graph-Processing Systems

Introduced by Ana Lucia Varbanescu.



In progress
Algorithms for different data types and graphs

Overstudied
Performance is enabled
Portability is disabled

Understudied
No systematic findings yet
Must be correlated with the algorithm

Lessons learned*

Performance of graph processing is function of
(Dataset, Algorithm, Platform, Deployment)

All current platforms have important drawbacks
(crashes, long execution time, tuning, etc.)
Best-performing is not only low response time
Scalability with cluster size/number of cores varies per system
Ease-of-use of a platform is very important

Graph-processing is at the core of our society
The data deluge vs. Analytics
Graph processing @large
Graphalytics: Which system to select? What to tune? What to re-design?
A performance comparison of graph-processing systems

Take-Home Message

Take-Home Message

Performance

Graphalytics: unified view and benchmarking of tens of systems, with little effort

Granula: iterative, fine-grained, shareable performance evaluation to enable performance engineering

The P-A-D Triangle:

Performance = f (Algorithm, Dataset, Platform, …)

Towards addressing the graph data deluge with

high performance, low development effort

Development Effort

Delft University of Technology

Reading List

- Alexandru Iosup, Tim Hegeman, Wing Lung Ngai, Stijn Heldens, Arnau Prat Perez, Thomas Manhardt, Hassan Chaffi, Mihai Capotă, Narayanan Sundaram, Michael Anderson, Ilie Gabriel Tanase, Yinglong Xia, Lifeng Nai, Peter Boncz, *LDBC Graphalytics: A Benchmark forLarge-Scale Graph Analysis on Parallel andDistributed Platforms*, VLDB'16.
- Mihai Capotă, Tim Hegeman, Alexandru Iosup, Arnau Prat-Pérez, Orri Erling, and Peter Boncz, *Graphalytics: A Big Data Benchmark for Graph-Processing Platforms*, International Workshop on Graph Data Management Experiences and Systems (GRADES), 2015.
- Guo et al., *An Empirical Performance Evaluation of GPU-Enabled Graph-Processing Systems*. CCGRID'15.
- A. Iosup, A. L. Varbanescu, M. Capotă, T. Hegeman, Y. Guo, W. L. Ngai, and M. Verstraaten, *Towards Benchmarking IaaS and PaaS Clouds for Graph Analytics*, Workshop on Big Data Benchmarking (WBDB), 2014.
- 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*, IEEE International Parallel and Distributed Processing Symposium (IPDPS), 2014, pp. 395–404.
- Y. Guo, A. L. Varbanescu, A. Iosup, C. Martella, and T. L. Willke, *Benchmarking graph-processing platforms*, ACM/SPEC International Conference on Performance Engineering (ICPE), 2014, pp. 289–292.

System diversity (GPU-enabled)

Dedicated Systems

Generic

medusa-gpu

Medusa: Simplified Graph Processing on GPUs

NetSysLab

mapgraph Beta

Massively Parallel Graph processing on GPUs

TOTEM

Gunrock

High-performance Graph Primitives on GPU

VertexAPI2

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 use of GPUs (T-G)
- Enables *single-node* heterogeneous (T-H) computing on graphs

- Programming requires expert knowledge of all types of systems
  - 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

MapGraph

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

Lessons learned from GPU-based systems

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

System diversity (CPU-based)

Dedicated Systems

Generic

ORACLE PGX

hadoop

GIRAPH

Neo4j the graph database

YARN

GraphLab

Trinity

StratoSphere Above the Clouds

GraphX

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 (now 6M$ investment from Intel)

Nephele resource manager

- Scalable parallel engine
- Jobs are represented as DAGs
- Supports data flow in-memory, via network, or via files

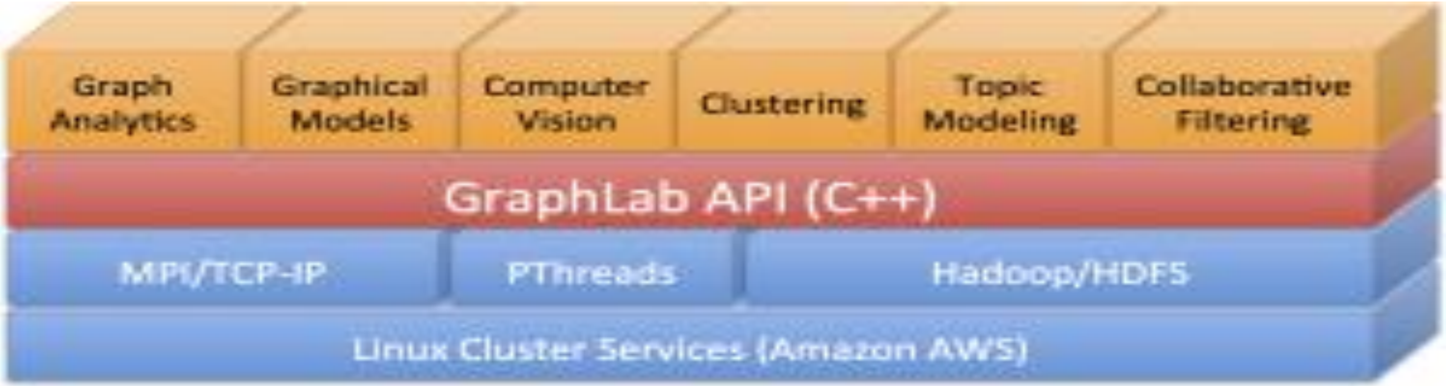
PACT job model

- 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 + Apache Giraph (Dedicated)

- 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

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

Neo4j
the graph database

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!

PGX.D: Programming Model

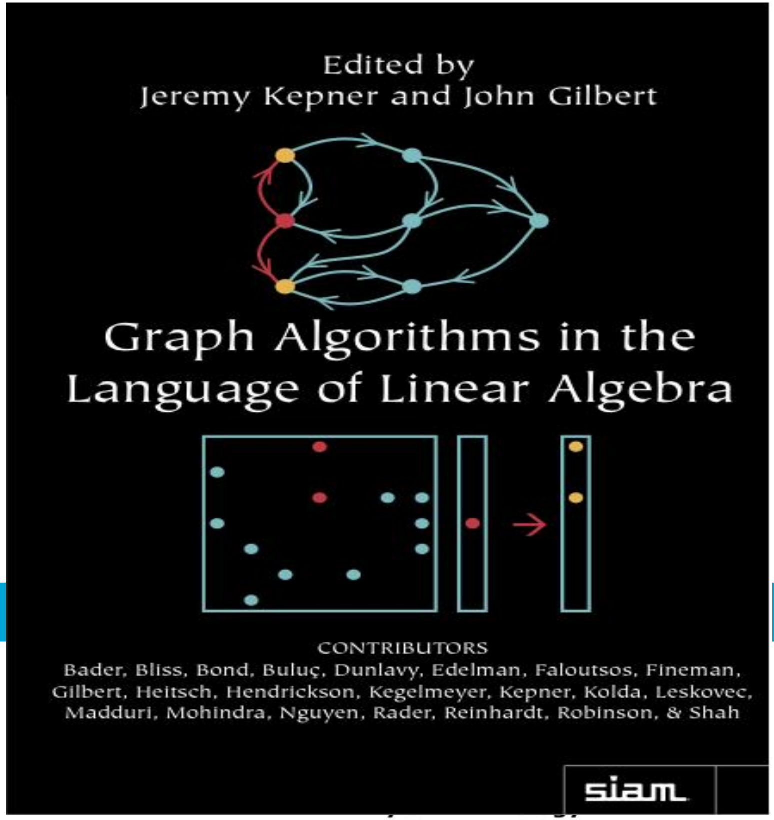
High level programming model for Neighborhood Iteration Tasks

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

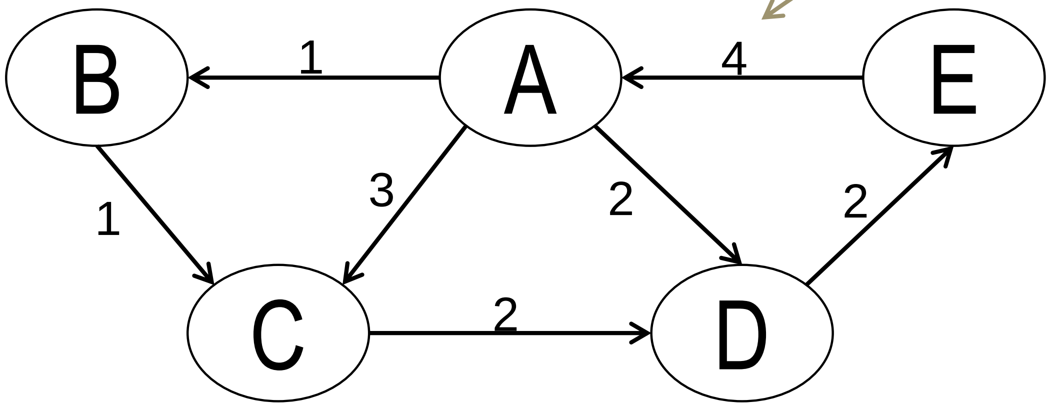

```
class my_task_pull : public innbr_iter_task {
  void run(..) {
    read_remote(get_nbr_id(), bar);
  }
  void read_done(void* buffer,..) {
    int foo_v = get_local<int>(node_id, foo);
    int bar_v= get_data<int>(buffer);
    set_local(node_id, foo_v + bar_v, foo);
  }
}
```

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



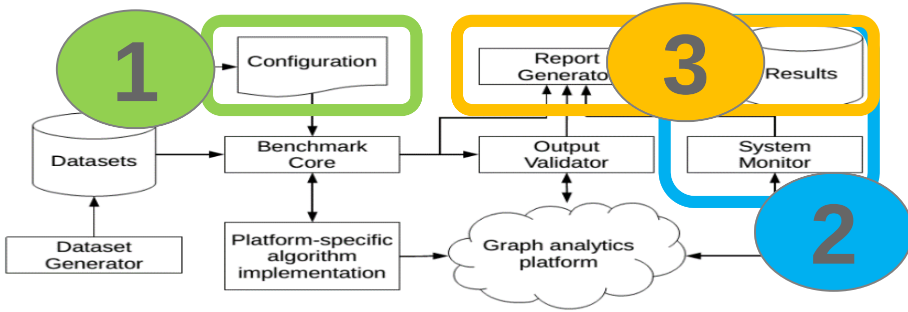
Example

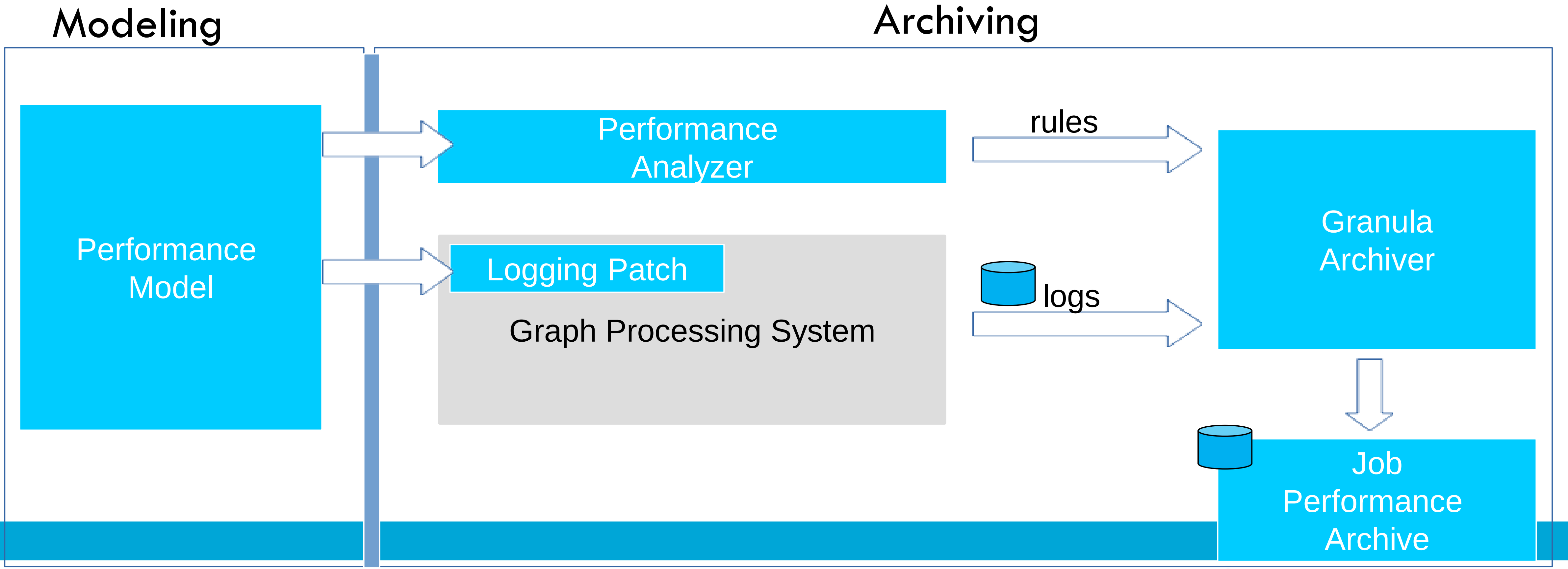


$$G^T = \begin{array}{c} \\ A \\ B \\ C \\ D \\ E \end{array} \begin{array}{c} \begin{array}{ccccc} A & B & C & D & E \end{array} \\ \begin{bmatrix} - & - & - & - & 4 \\ 1 & - & - & - & - \\ 3 & 1 & - & - & - \\ 2 & - & 2 & - & - \\ - & - & - & 2 & - \end{bmatrix} \end{array}$$

<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)

2 Granula Archiver



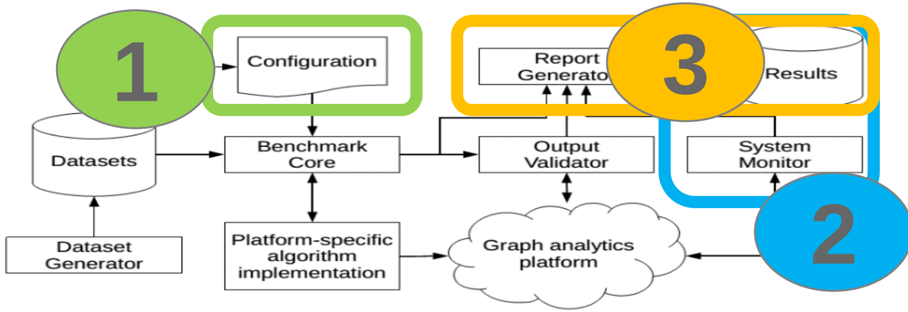


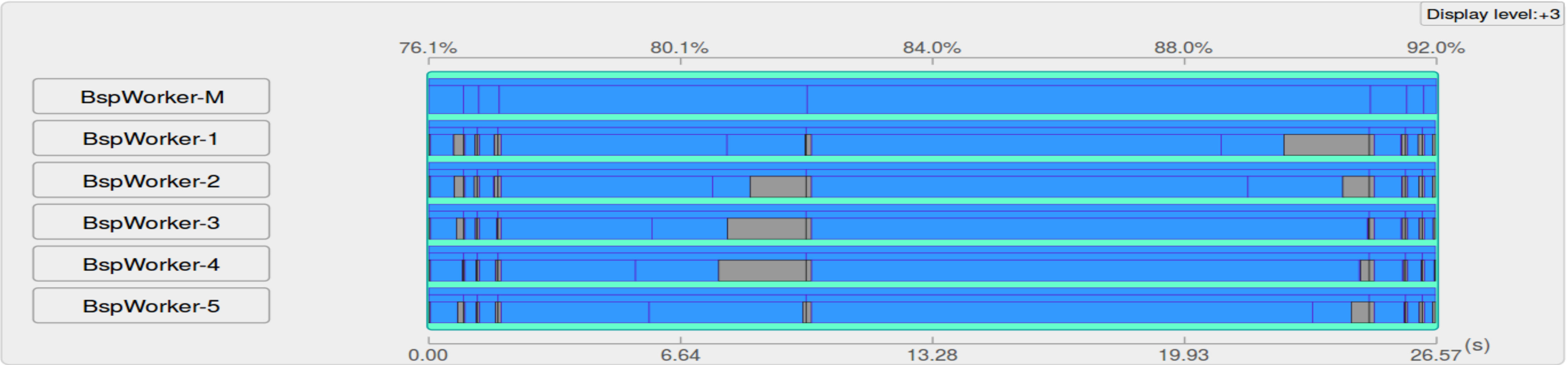
Time-consuming, minimal code invasion,
automated data collection at runtime, portable archive

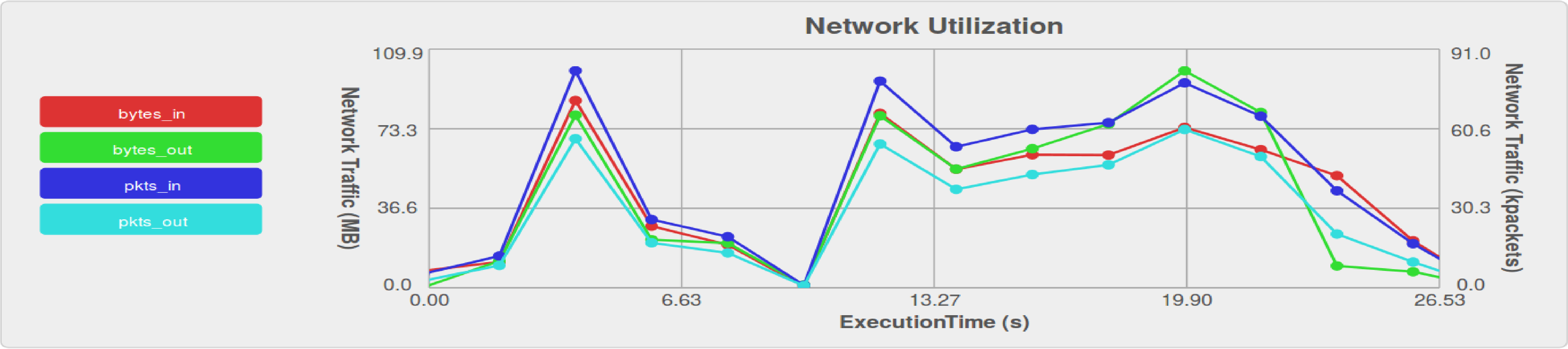
TU Delft
Delft University of Technology

3 Granula Visualizer

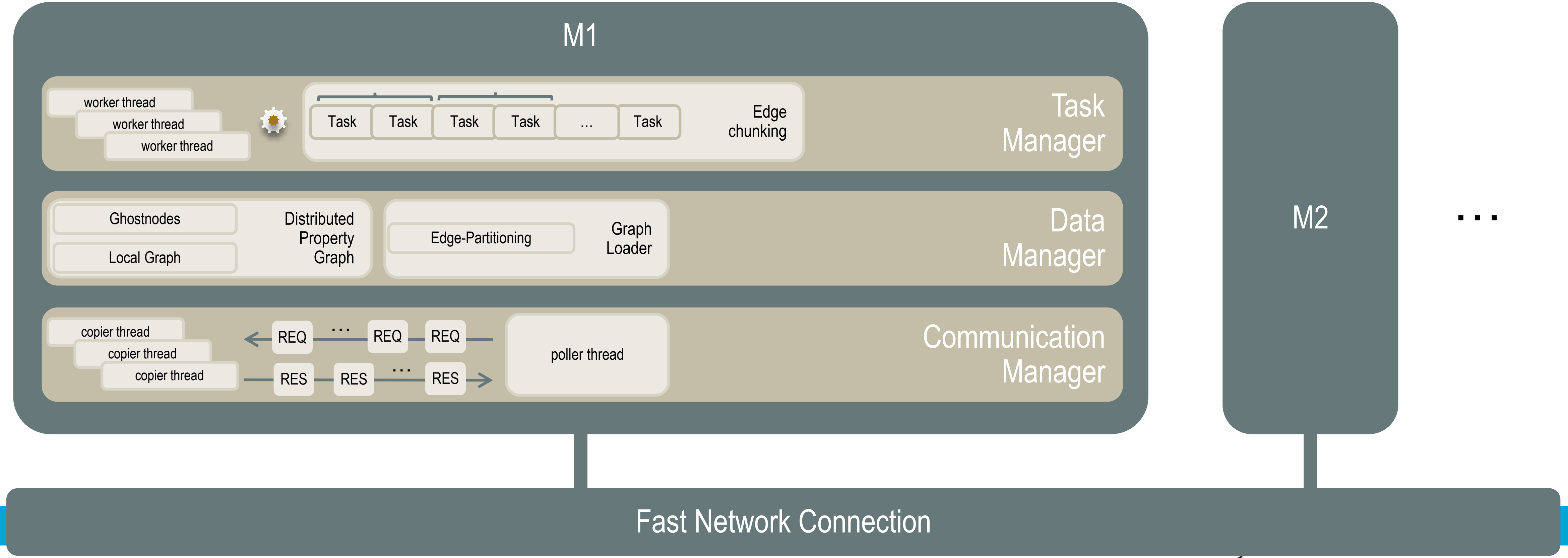
Portable choke-point analysis for everyone!







PGX.D: System Design Overview



Icons