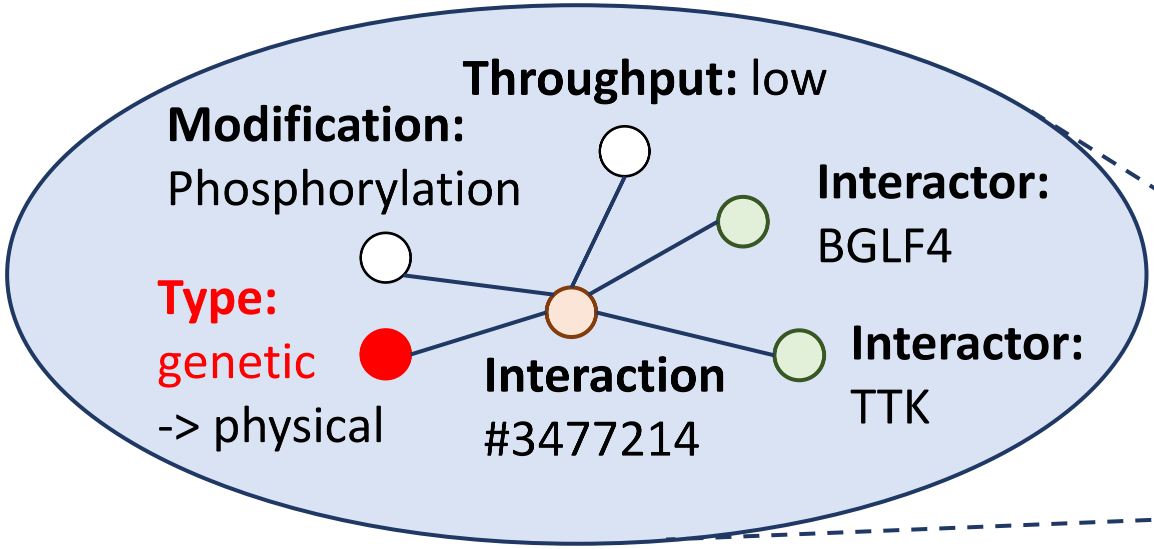
MiniClean: A Single-Machine System for Cleaning Big Graphs

Wenchao Bai, Wenfei Fan, Jiahui Jin, Daji Li, Jian Li,
Shuhao Liu, Mingliang Ouyang, Qiang Yuan

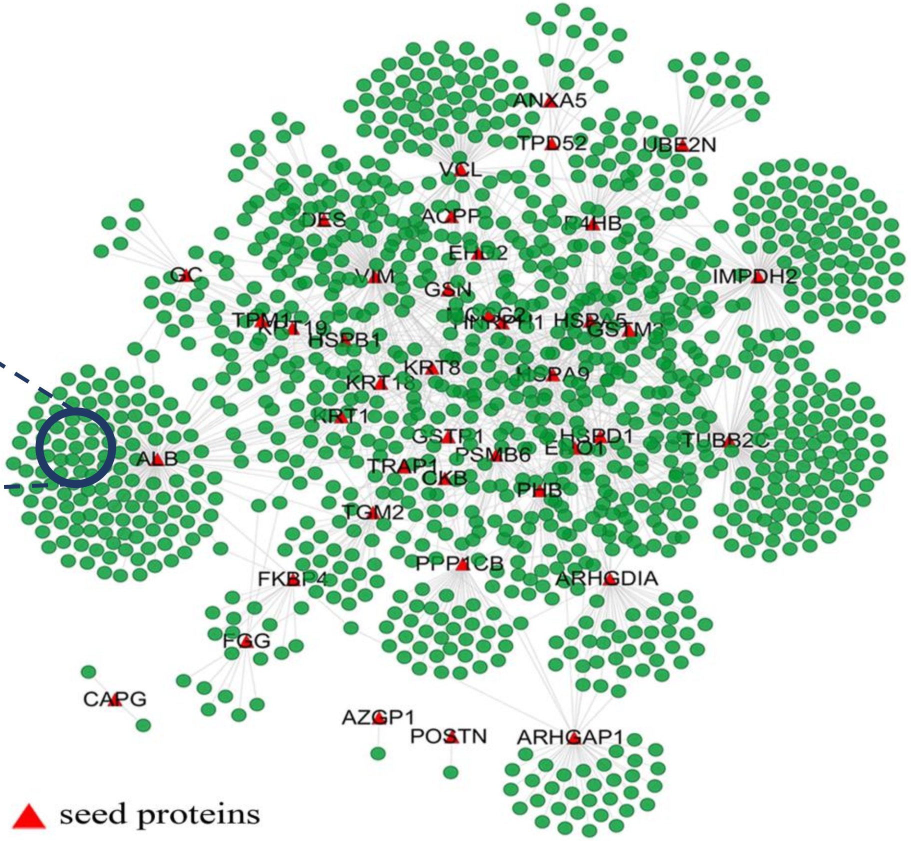
Shenzhen Institute of Computing Science, Southeast University,
University of Edinburgh, Beihang University

- **Introduction**: 00’00 – 08’55
- **Live Demo**: 08’55 – 12’21
- **Evaluation**: 12’21 – 13’38

Graph Data Quality and Graph Applications



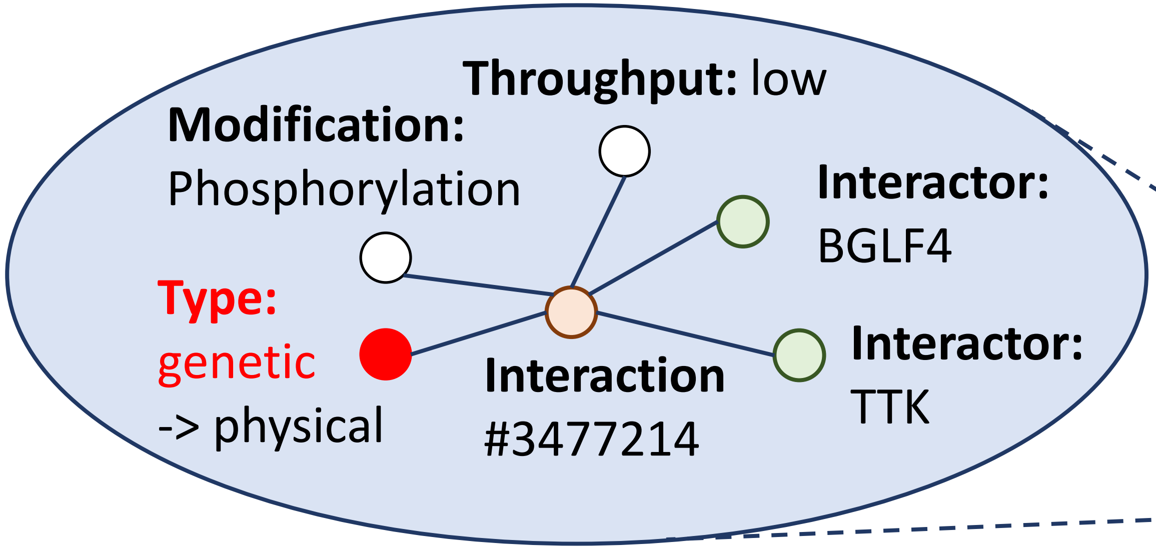
A mislabeled instance in BioGRID[2]



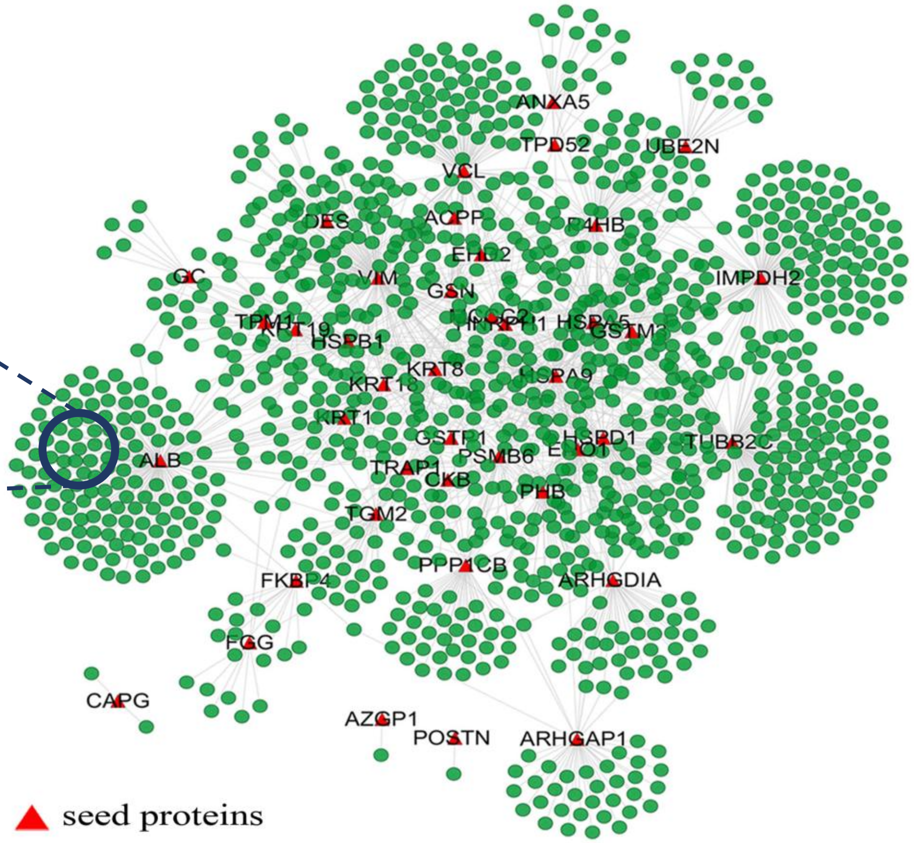
[1] Chen, Chen, et al. "Construction and analysis of protein-protein interaction networks based on proteomics data of prostate cancer." *International journal of molecular medicine* 37.6 (2016): 1576-1586.

[2] BioGRID | Database of Protein, Chemical, and Genetic Interactions

Graph Data Quality and Graph Applications



A mislabeled instance in BioGRID[2]



- *Errors (e.g., duplicates, conflicts) are common.*
- *Errors can misguide data-driven decision making.*

[1] Chen, Chen, et al. "Construction and analysis of protein-protein interaction networks based on proteomics data of prostate cancer." *International journal of molecular medicine* 37.6 (2016): 1576-1586.
[2] BioGRID | Database of Protein, Chemical, and Genetic Interactions

Graph cleaning is crucial

Approaches to Graph Cleaning

- **Strawman 1**: ML models (e.g., Ditto, KGClean).
  - Transform graphs into embeddings.
  - Detect errors (duplicates/conflicts) via binary classification.
  - Predictions are probabilistic and hard to explain.
- **Strawman 2**: Logic rules.
  - Apply rules via pattern matching and predicate verification.
  - Deduce dependencies to detect and correct errors.
  - Difficult to find rules that cover all the cases.

Take advantage of both ML and logic reasoning

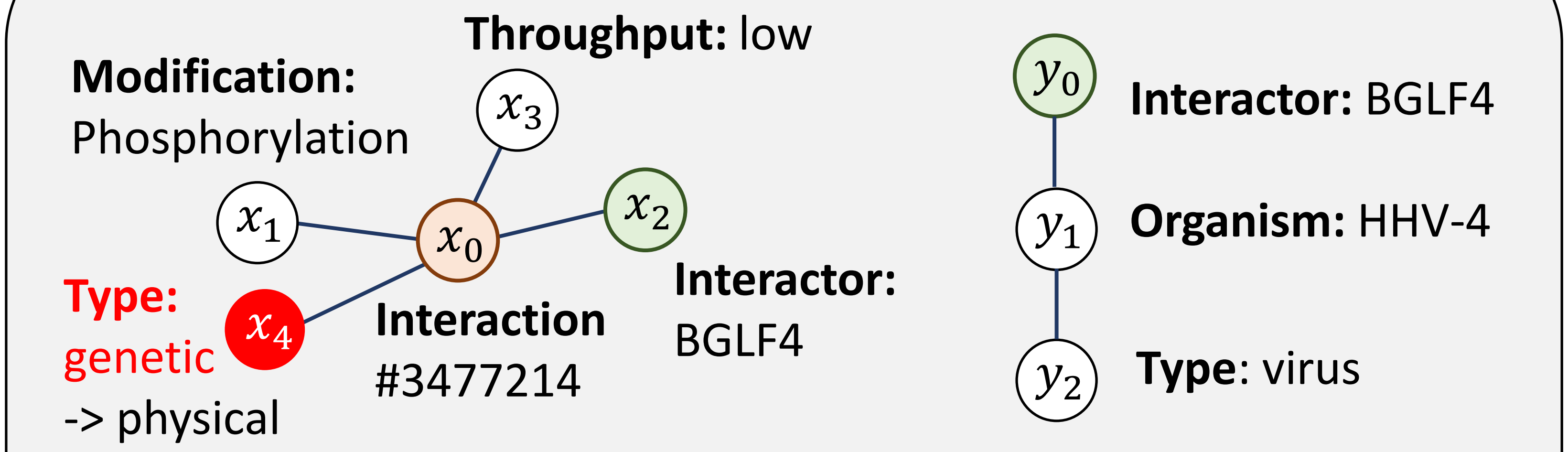
Approaches to Graph Cleaning

- **Strawman 1**: ML models (e.g., Ditto, KGClean).
  - Transform graphs into embeddings.
  - Detect errors (duplicates/conflicts) via binary classification.
  - Predictions are probabilistic and hard to explain.
- **Strawman 2**: Logic rules.
  - Apply rules via pattern matching and predicate verification.
  - Deduce dependencies to detect and correct errors.
  - Difficult to find rules that cover all the cases.

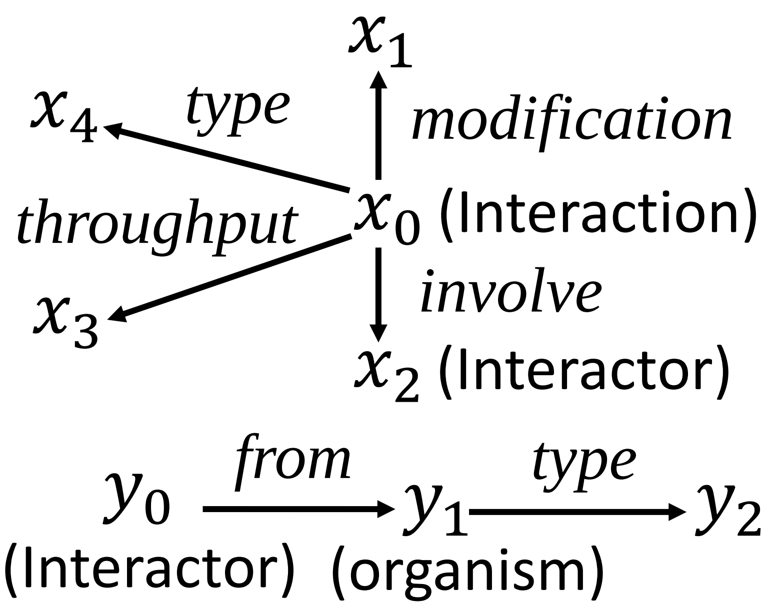
- **Our Solution**: ML models + Logic rules.
  - $M_1(x_0, x_1) > \delta_1 \wedge X_1 \rightarrow x_4.\mathrm{val} = \mathrm{physical}$
  - $x_4.\mathrm{val} = \mathrm{genetic} \wedge X_2 \rightarrow M_2(x_0, x_3) > \delta_2$

Take advantage of both ML and logic reasoning

Approaches to Graph Cleaning



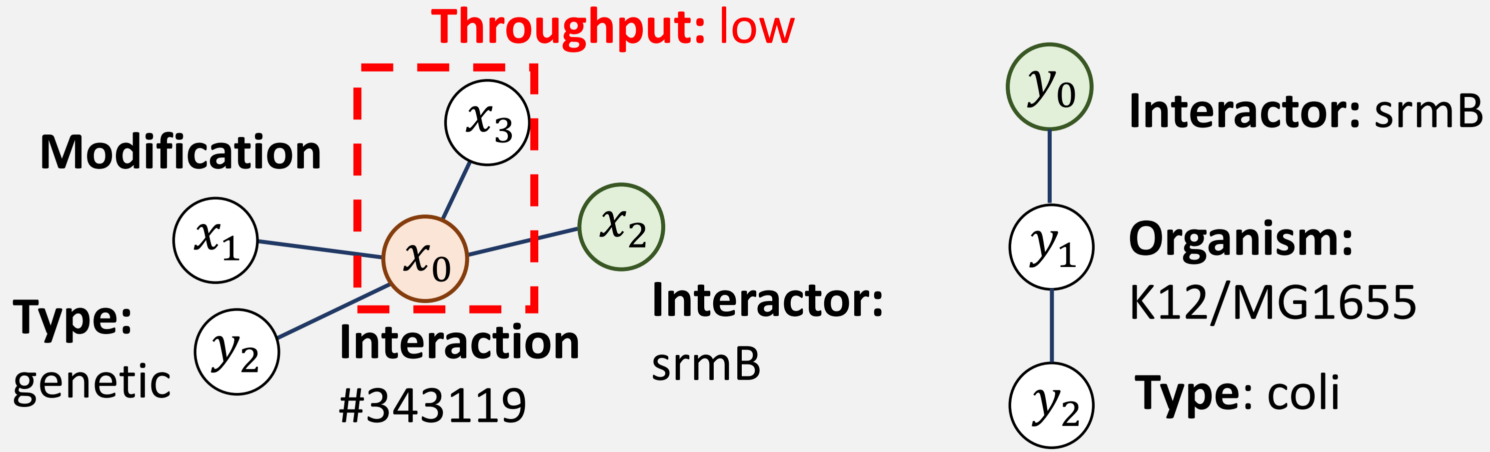
$$X_1 \coloneqq < x_2.\mathrm{id} = y_0.\mathrm{id} \wedge y_2.\mathrm{val} = \mathrm{virus} >$$



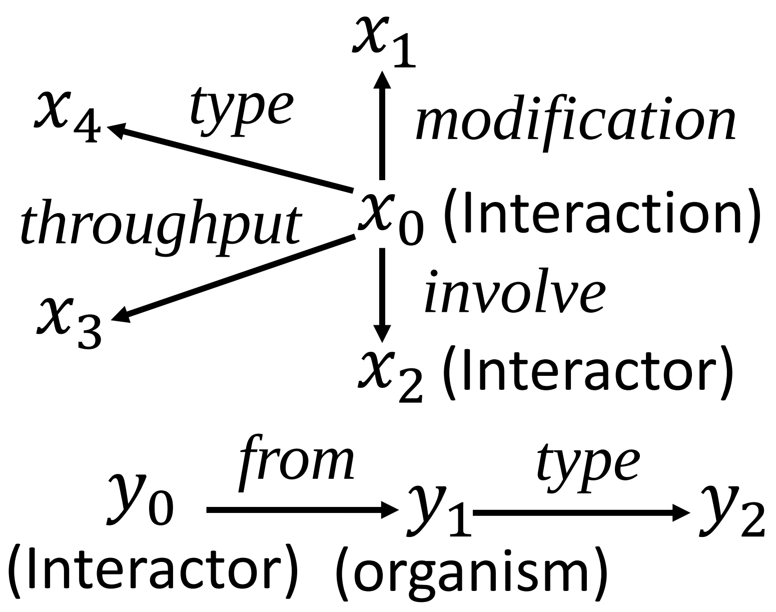
- **Our Solution**: ML models + Logic rules.
  - $M_1(x_0, x_1) > \delta_1 \wedge X_1 \rightarrow x_4.\mathrm{val} = \mathrm{physical}$
  - $x_4.\mathrm{val} = \mathrm{genetic} \wedge X_2 \rightarrow M_2(x_0, x_3) > \delta_2$

Take advantage of both ML and logic reasoning

Approaches to Graph Cleaning



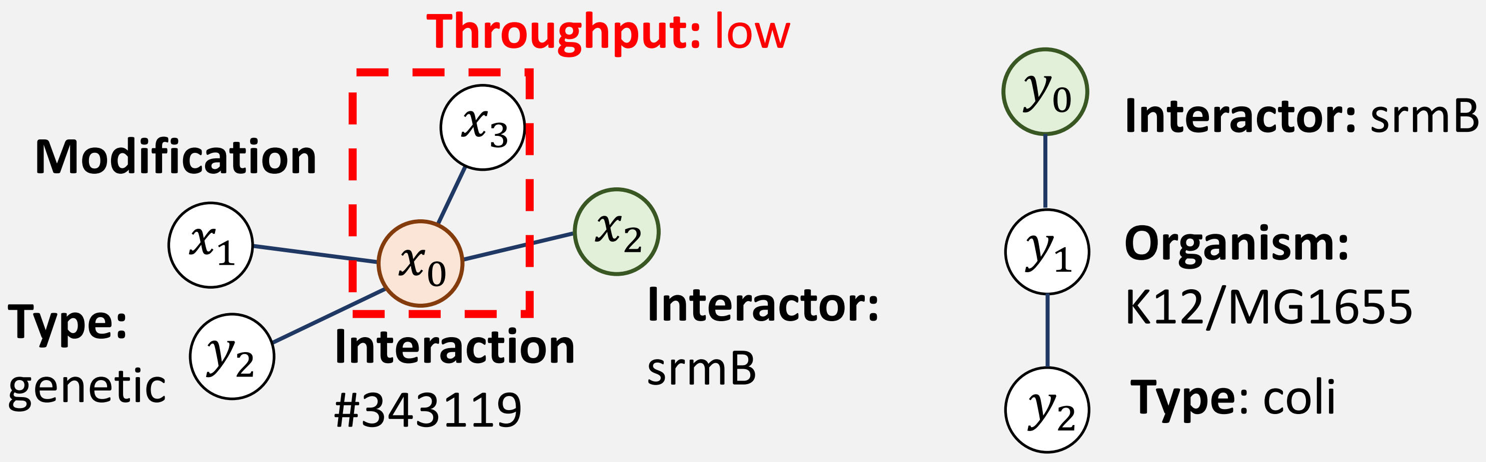
$$X_2 \coloneqq < x_2.\mathrm{id} = y_0.\mathrm{id} \wedge y_2.\mathrm{val} = \mathrm{coli} >$$



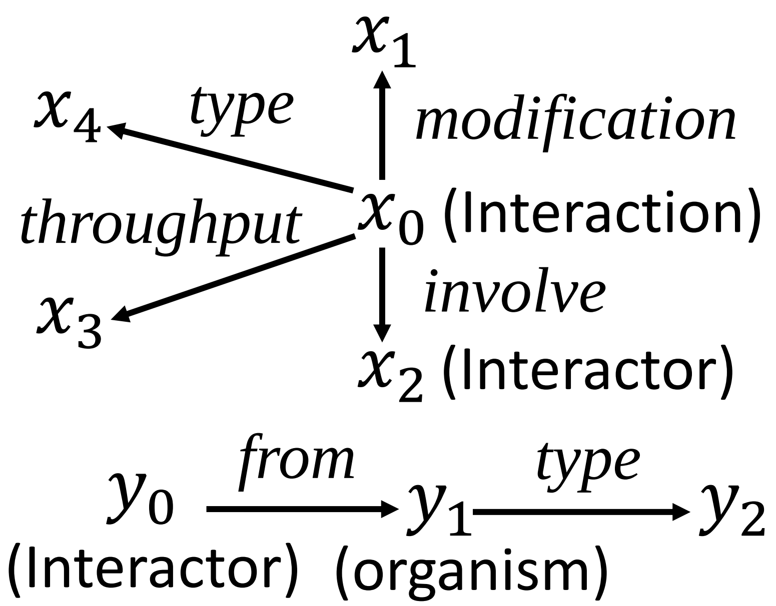
- **Our Solution**: ML models + Logic rules.
  - $M_1(x_0, x_1) > \delta_1 \wedge X_1 \rightarrow x_4.\mathrm{val} = \mathrm{physical}$
  - $x_4.\mathrm{val} = \mathrm{genetic} \wedge X_2 \rightarrow M_2(x_0, x_3) > \delta_2$

Take advantage of both ML and logic reasoning

Approaches to Graph Cleaning



$$X_2 := < x_2.\mathrm{id} = y_0.\mathrm{id} \wedge y_2.\mathrm{val} = \mathrm{coli} >$$

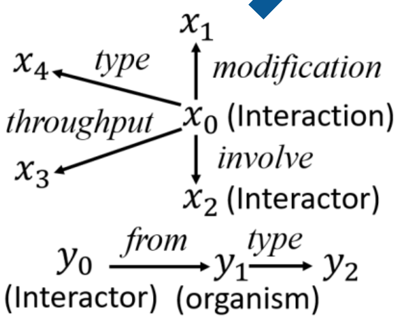


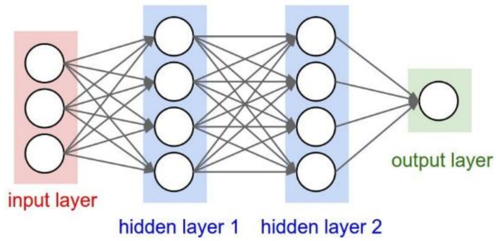
- **Our Solution**: ML models + Logic rules.
  - $M_1(x_0, x_1) > \delta_1 \wedge X_1 \rightarrow x_4.\mathrm{val} = \mathrm{physical}$
  - $x_4.\mathrm{val} = \mathrm{genetic} \wedge X_2 \rightarrow M_2(x_0, x_3) > \delta_2$

- *Generalizability (ML)*
- *Reliability (Logic)*
- *Explainability (Logic)*

Graph Cleaning Rules (GCRs)

$\mathcal{Q}[x_0, y_0](X \rightarrow p_0)$





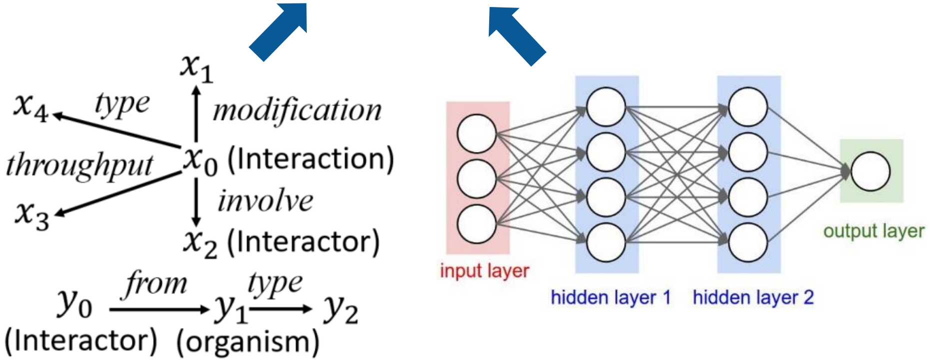
- ✓ $\mathcal{Q}[x_0, y_0]$: dual star pattern
- ✓ $X \rightarrow p_0$: dependency
- ✓ X: precondition (conjunction of predicates)
- ✓ p_0: consequence (predicate)

Dependency deduction:

- ML predicates $M(x.\bar{A}, y.\bar{B})$: for ML classification & regression
- Variable and constant predicates $x.A \oplus y.B, x.A \oplus c$: for value associations

Graph Cleaning Rules (GCRs)

$Q[x_0, y_0](X \to p_0)$



- ✓ $Q[x_0, y_0]$: dual star pattern
- ✓ $X \to p_0$: dependency
- ✓ X: precondition (conjunction of predicates)
- ✓ p_0: consequence (predicate)

Dependency deduction:

- ML predicates $M(x.\bar{A}, y.\bar{B})$: for ML classification & regression
- Variable and constant predicates $x.A \oplus y.B$, $x.A \oplus c$: for value associations

Functionalities

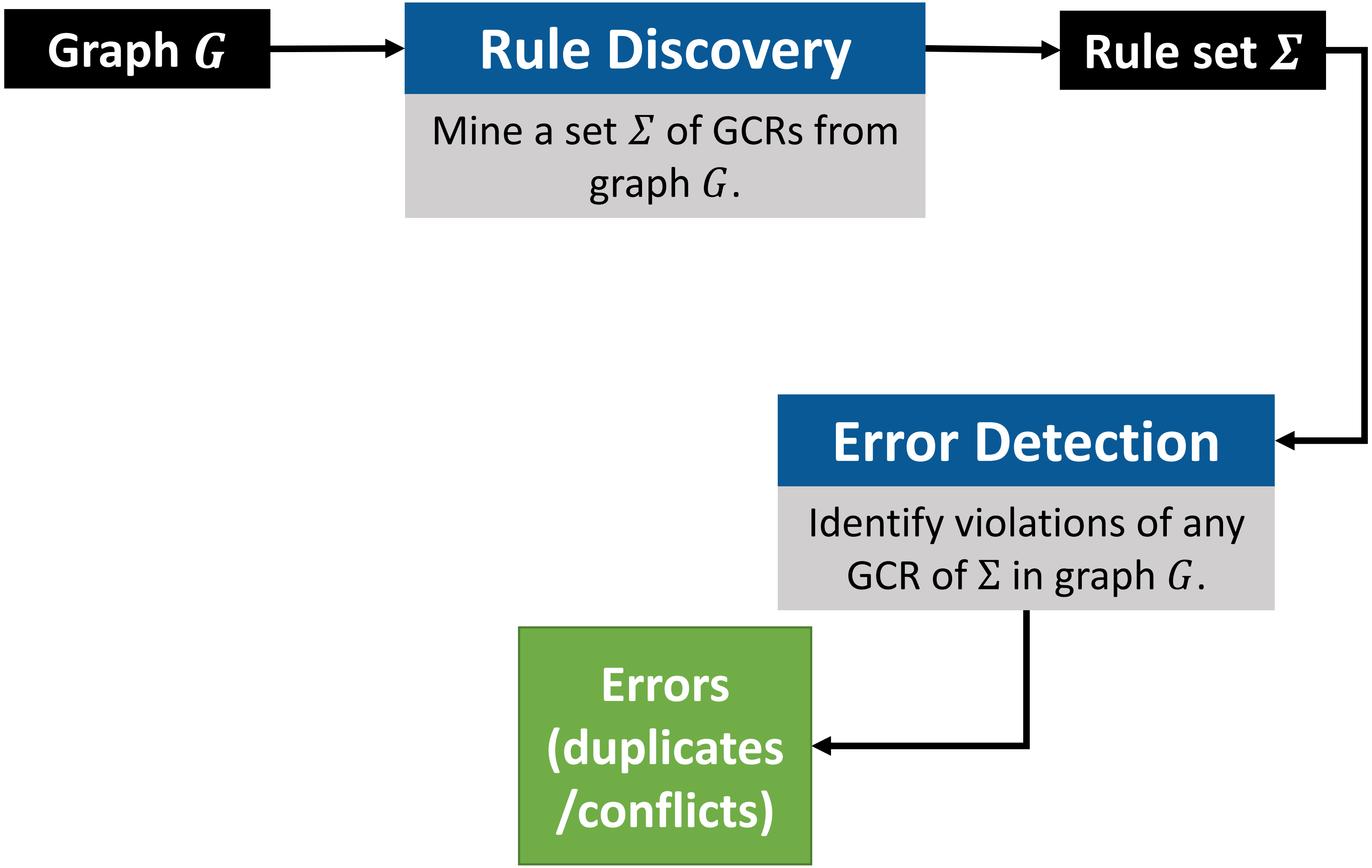
- ✓ *Conflict resolution*
- ✓ *Entity resolution*
- ✓ *ML explanation*

Cleaning with GCRs

Graph G → **Rule Discovery** → **Rule set Σ**

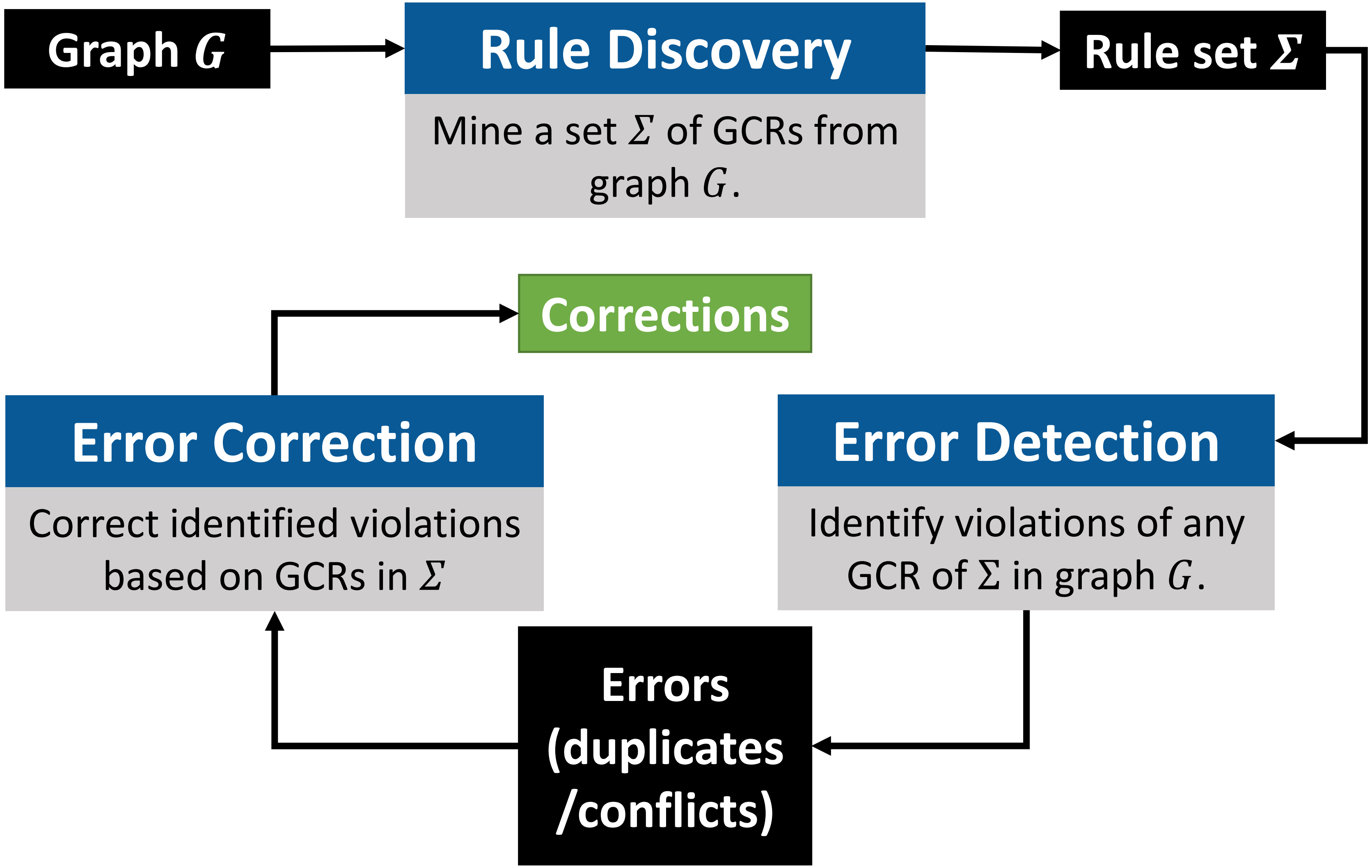
Mine a set Σ of GCRs from graph G.

Cleaning with GCRs



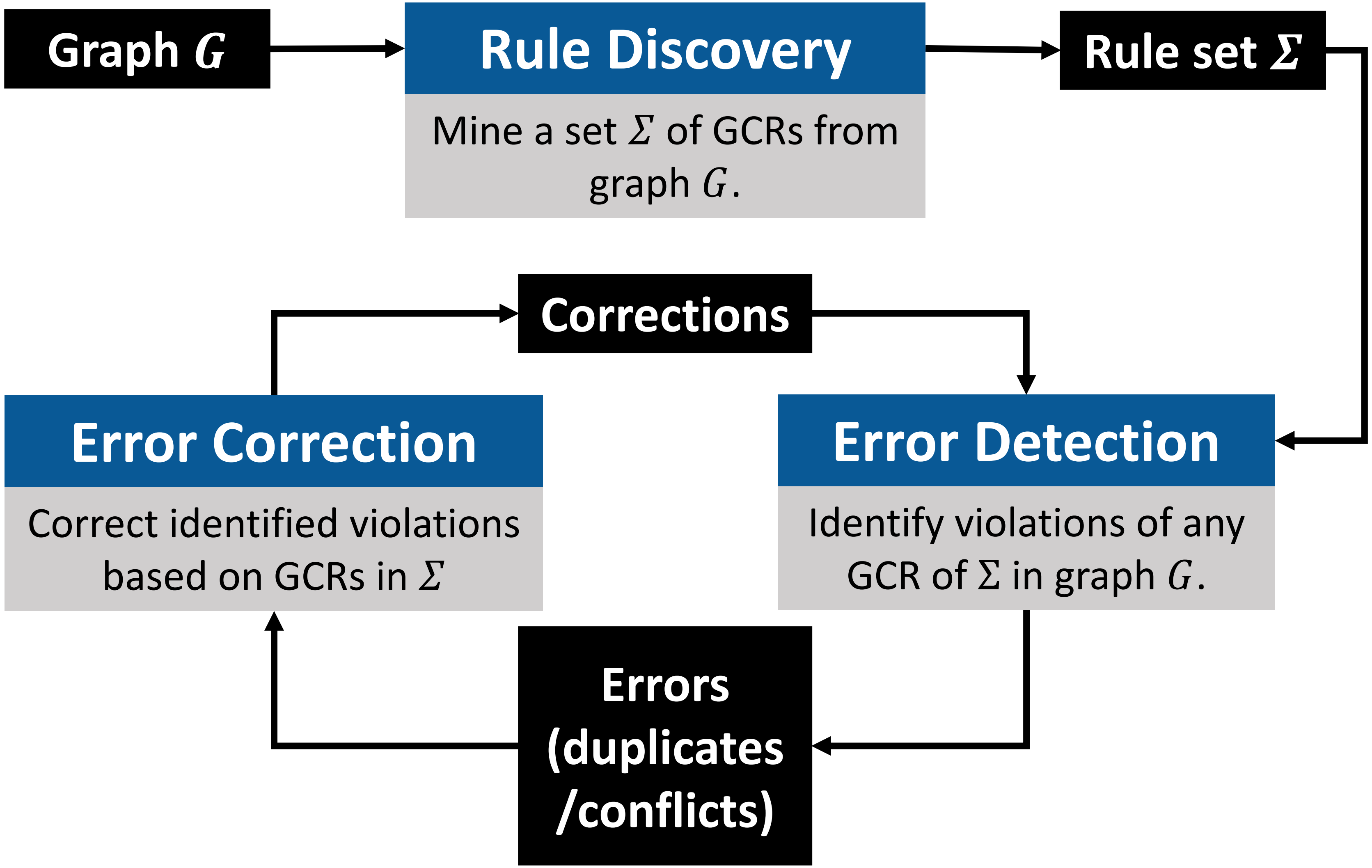
The workflow of graph cleaning

Cleaning with GCRs



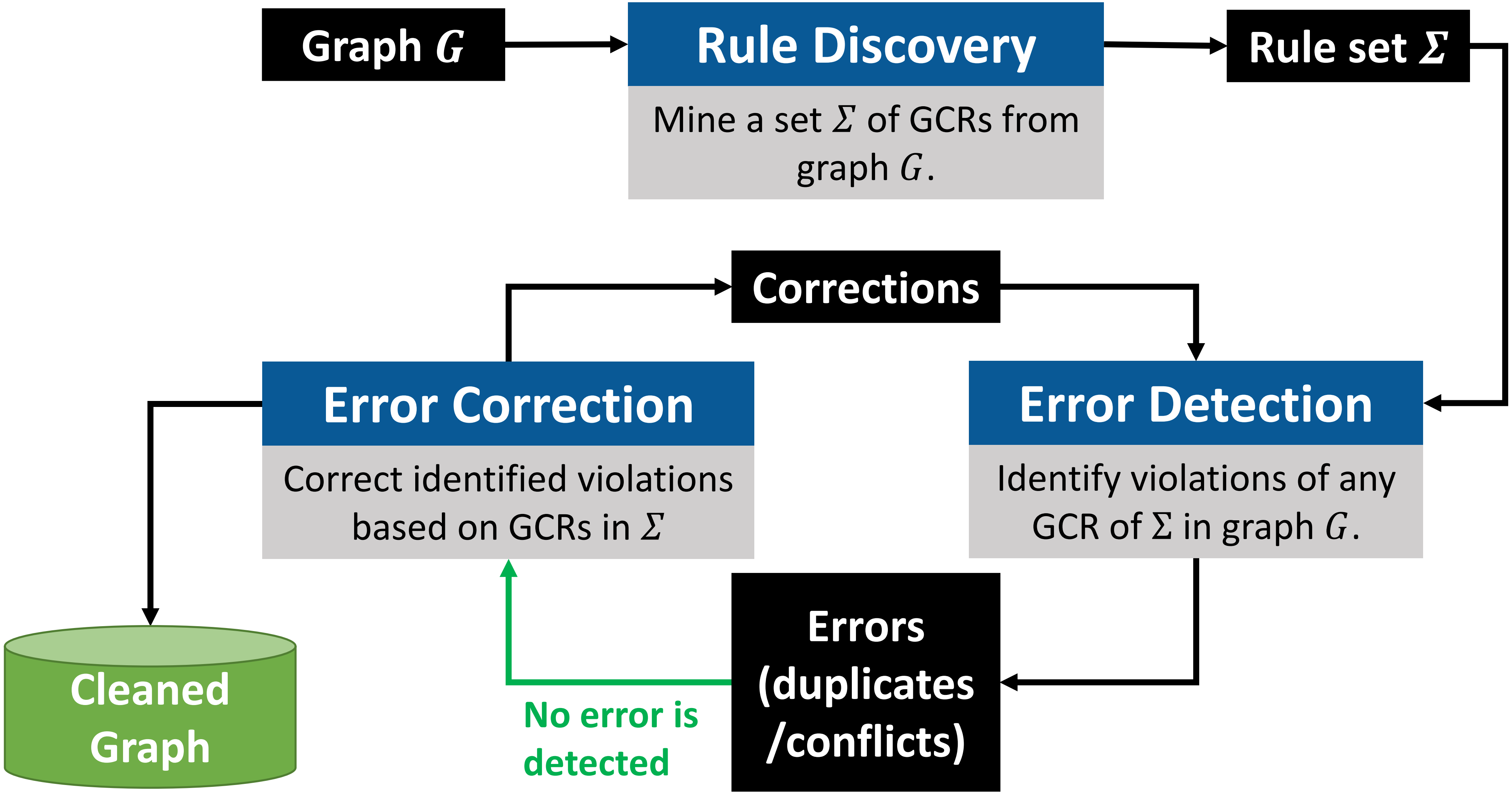
The workflow of graph cleaning

Cleaning with GCRs



The workflow of graph cleaning

Cleaning with GCRs



The workflow of graph cleaning

Single-machine solutions are often preferred:

- **Physical constraints on deployed hardware.**
  - A cluster on premise may not be feasible, e.g., at an edge location.
- **A cloud deployment is not viable.**
  - Prohibitive cost: network bandwidth at $0.05–0.09 per GB[1].
  - Privacy concerns.

[1] https://aws.amazon.com/pricing

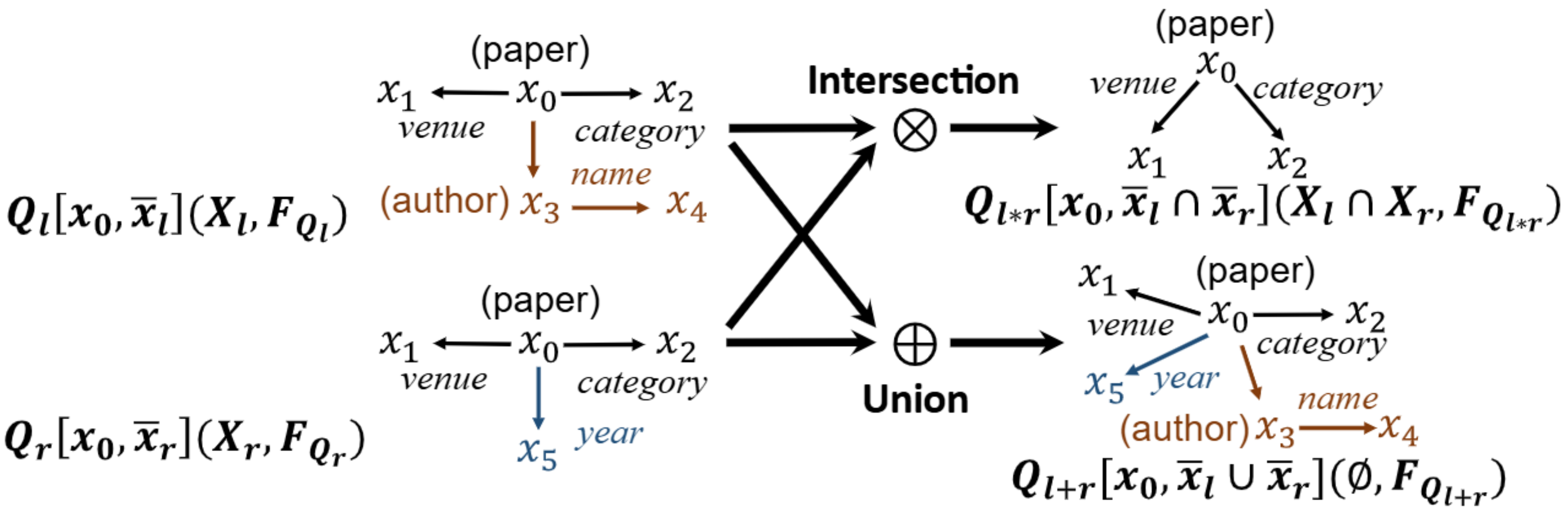
Single-machine graph cleaning is a practical need!

Single-Machine Graph Cleaning: Challenges

- **Computation heavy.**
  - > 9h processing time on a 32-node cluster.
- **Excessive intermediate data.**
  - > 150GB intermediate result, way exceeding the memory capacity of a typical machine.
- **Parallel model for maximum resource utilization.**
  - Idle caused by I/O, data transfer, and task dependencies.

No prior single-machine systems can handle the task!

Optimization 1: Bundled Processing



- ✓ **Motivation**: Repetitive computation by matching common substructures.
- ✓ **Solution**: Bundle similar patterns into a group and match them together.

Optimization 2: Data Compression

Conditional succinct matches

Matches			Conditions		
$x_0.\mathrm{id}$	$x_1.\mathrm{val}$	$x_2.\mathrm{val}$	$x_4.\mathrm{val}$	$x_0.\mathrm{title}$	$x_5.\mathrm{val}$
u_3	OSDI	CS	{J, S}	MR	null
u_4	OSDI	CS	{J, S}	GFS	null
u_1	OSDI	null	{M, P, J, S}	TF	2016
u_2	OSDI	null	{M, P, J, S}	tf	2016

$\Big\}$ Filtered out by $x_2.\mathrm{val} = \mathrm{CS}$

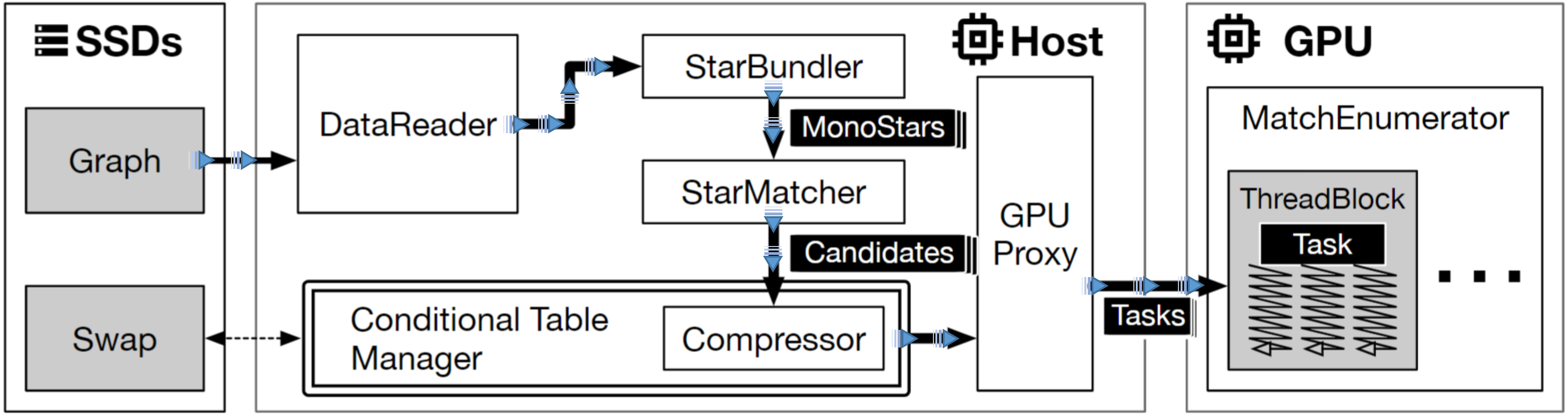
Unfolded matches

$x_0.\mathrm{id}$	...	$x_4.\mathrm{val}$	...
u_3	...	J	...
u_3	...	S	...
...	...	...	...

- ✓ **Motivation**: Intermediate data has up to $O(|V|^{|Q|})$ materialized candidates.
- ✓ **Solution**: Conditional succinct table for compression.

Reduce I/O overhead by at least 4.7×

Optimization 3: Pipelined Architecture

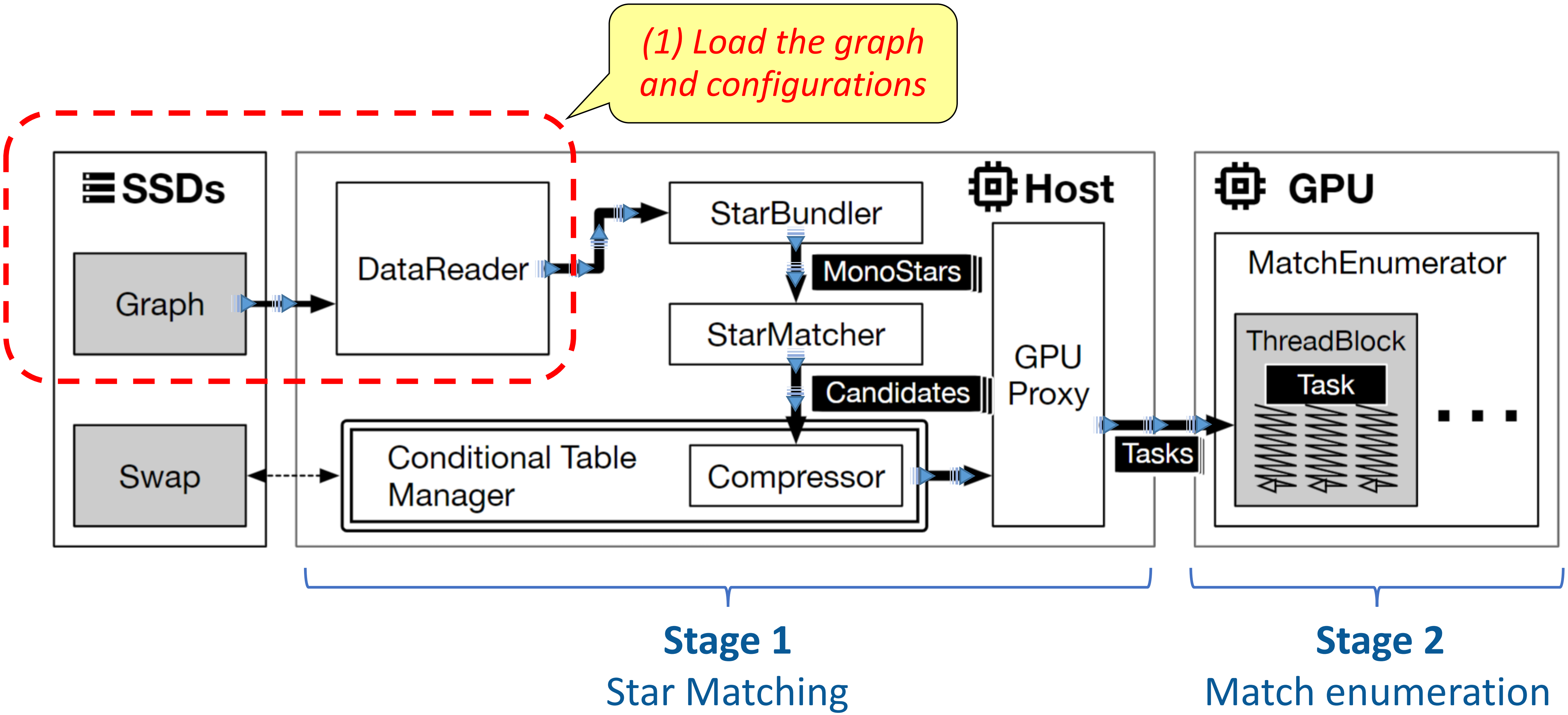


Stage 1
Star Matching

Stage 2
Match enumeration

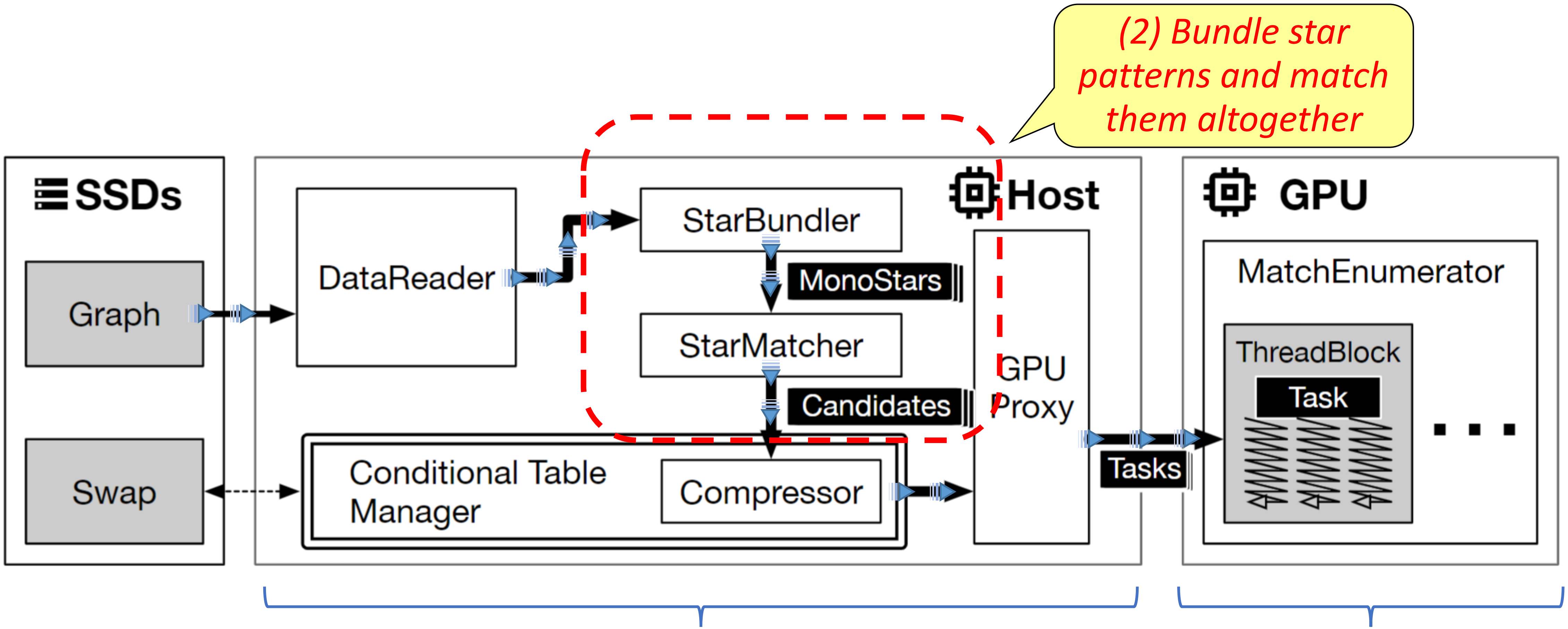
Maximally overlapping I/O, CPU and GPU operations

Optimization 3: Pipelined Architecture



Maximally overlapping I/O, CPU and GPU operations

Optimization 3: Pipelined Architecture

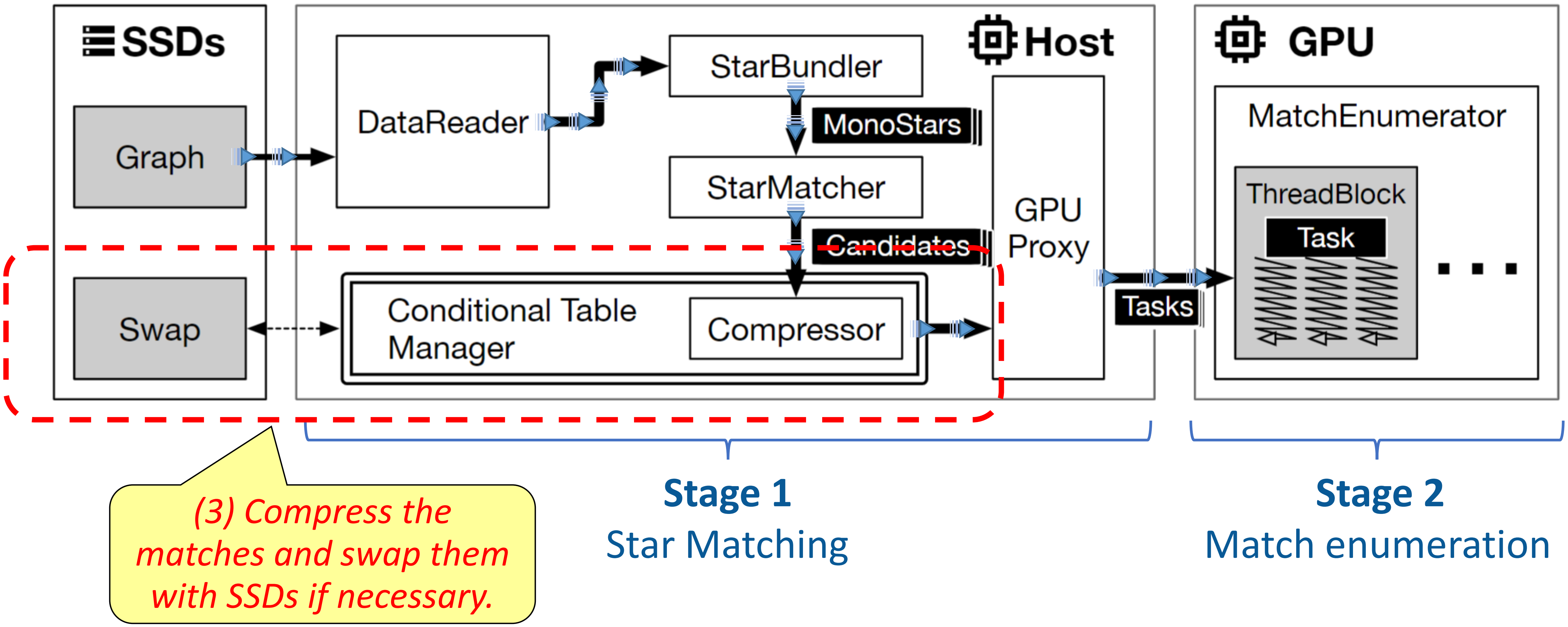


Stage 1
Star Matching

Stage 2
Match enumeration

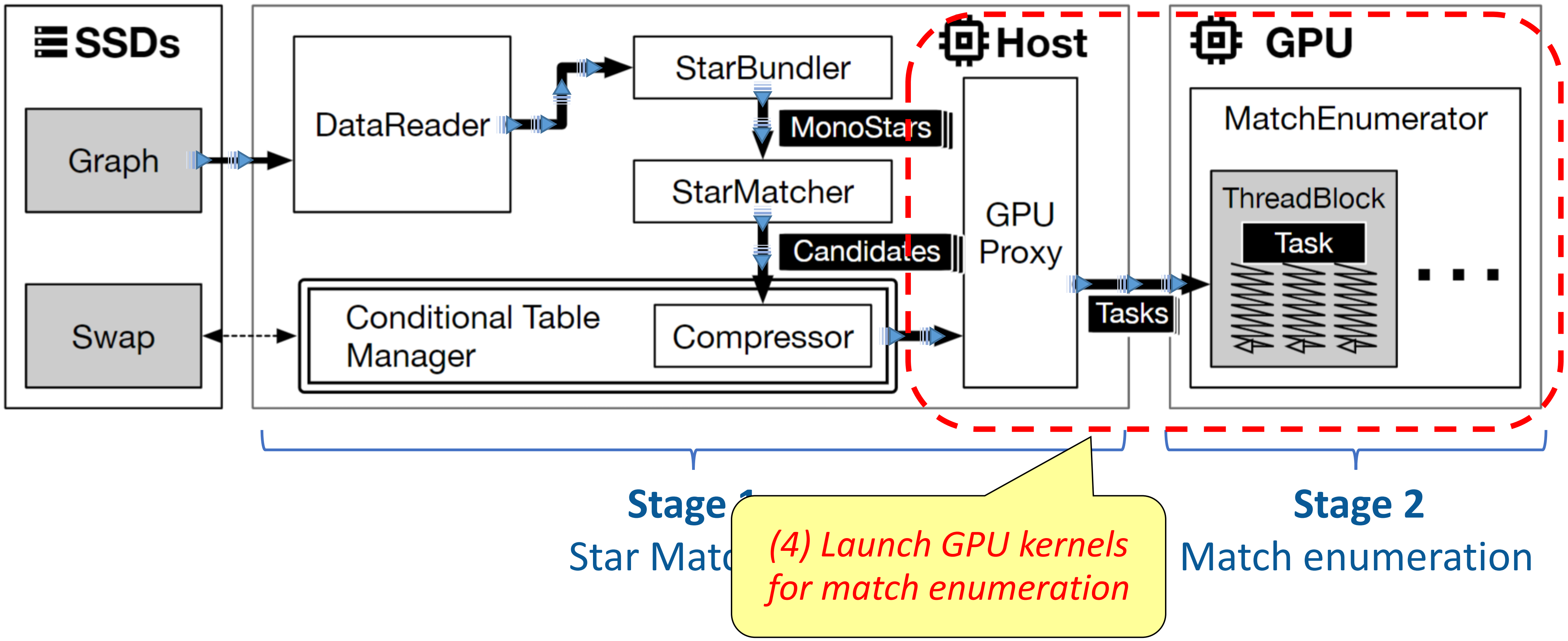
Maximally overlapping I/O, CPU and GPU operations

Optimization 3: Pipelined Architecture



Maximally overlapping I/O, CPU and GPU operations

Optimization 3: Pipelined Architecture



Maximally overlapping I/O, CPU and GPU operations

Guided Tour

In our demonstration, we will walk the participants through

- **Data management**: visualize the input graph and GCRs;
- **GCR discovery**: configure and launch a rule discovery task;
- **GCR application**: inspect the errors detected/corrected by each GCR.

We will use BioGRID, a real PPI graph in our demonstration.

System Evaluation: Accuracy

System	BioGRID			SemScholar		
	Time	ER-F1(%)	CR-F1(%)	Time	ER-F1(%)	CR-F1(%)
MiniClean	312.7s	97.5	97.2	4089.1s	94.6	70.6
Ditto	7.7×	91.0	N/A	4.3×	90.3	N/A
KGClean	11.9×	N/A	54.9	11.9×	N/A	29.8

- **ER tasks**: outperforms ER model ***Ditto*** by 4.3%--6.5%.
- **CR tasks**: outperforms CR model ***KGClean*** by 40.4%--42.3%.
- **Efficiency**: 11.9× faster than ML baselines.

System Evaluation: Accuracy

System	BioGRID			SemScholar		
	Time	ER-F1(%)	CR-F1(%)	Time	ER-F1(%)	CR-F1(%)
MiniClean	312.7s	97.5	97.2	4089.1s	94.6	70.6
Ditto	7.7×	91.0	N/A	4.3×	90.3	N/A
KGClean	11.9×	N/A	54.9	11.9×	N/A	29.8

$\Delta = 6.5\%$ $\Delta = 4.3\%$

- **ER tasks**: outperforms ER model ***Ditto*** by 4.3%--6.5%.
- **CR tasks**: outperforms CR model ***KGClean*** by 40.4%--42.3%.
- **Efficiency**: 11.9× faster than ML baselines.

System Evaluation: Accuracy

System	BioGRID			SemScholar		
	Time	ER-F1(%)	CR-F1(%)	Time	ER-F1(%)	CR-F1(%)
MiniClean	312.7s	97.5	97.2	4089.1s	94.6	70.6
Ditto	7.7×	91.0	N/A	4.3×	90.3	N/A
KGClean	11.9×	N/A	54.9	11.9×	N/A	29.8

$\Delta = 42.3\%$ $\Delta = 40.8\%$

- **ER tasks**: outperforms ER model ***Ditto*** by 4.3%--6.5%.
- **CR tasks**: outperforms CR model ***KGClean*** by 40.8%--42.3%.
- **Efficiency**: 11.9× faster than ML baselines.

Higher accuracy than ML baselines

System Evaluation: Accuracy

System	BioGRID			SemScholar		
	Time	ER-F1(%)	CR-F1(%)	Time	ER-F1(%)	CR-F1(%)
MiniClean	312.7s	97.5	97.2	4089.1s	94.6	70.6
Ditto	7.7×	91.0	N/A	4.3×	90.3	N/A
KGClean	11.9×	N/A	54.9	11.9×	N/A	29.8

- **ER tasks**: outperforms ER model ***Ditto*** by 4.3%--6.5%.
- **CR tasks**: outperforms CR model ***KGClean*** by 40.4%--42.3%.
- **Efficiency**: 11.9× faster than ML baselines.

Higher accuracy than ML baselines

System Evaluation: Efficiency & Scalability

Single-machine (BioGRID)		Scalability (SemScholar)		Ablation study (SemScholar)	
System	**Time (s)**	**System**	**Time (s)**	**System**	**Time (s)**
Blaze	OOM	GCRClean-32-Node	8.09×	noBundle	1.35×
MiniGraph	65.34×	MiniClean-2-GPU	0.64×	noPipelinedPar	1.26×
Hyperblocker	11.92×	MiniClean-4-GPU	0.48×	noIndPar	1.12×
MiniClean-1-GPU	259.6s	MiniClean-1-GPU	4251.08s	MiniClean-2-GPU	2993.7s

- **Single-machine**: 65.34× faster than **MiniGraph**; 11.92× faster than **Hyperblocker**.
- **Multi-machine**: 8.09× faster than 32-node **GCRClean**.
- **Scalability**: speedup by 1.56× with **2 GPUs**; 2.08× with **4 GPUs**.

Better performance and good scalability

System Evaluation: Efficiency & Scalability

Single-machine (BioGRID)		Scalability (SemScholar)		Ablation study (SemScholar)	
System	**Time (s)**	**System**	**Time (s)**	**System**	**Time (s)**
Blaze	OOM	GCRClean-32-Node	8.09×	noBundle	1.35×
MiniGraph	65.34×	MiniClean-2-GPU	0.64×	noPipelinedPar	1.26×
Hyperblocker	11.92×	MiniClean-4-GPU	0.48×	noIndPar	1.12×
MiniClean-1-GPU	259.6s	MiniClean-1-GPU	4251.08s	MiniClean-2-GPU	2993.7s

- **Single-machine**: 65.34× faster than **MiniGraph**; 11.92× faster than **Hyperblocker**.
- **Multi-machine**: 8.09× faster than 32-node **GCRClean**.
- **Scalability**: speedup by 1.56× with **2 GPUs**; 2.08× with **4 GPUs**.

Better performance and good scalability

System Evaluation: Efficiency & Scalability

Single-machine (BioGRID)		Scalability (SemScholar)		Ablation study (SemScholar)	
System	**Time (s)**	**System**	**Time (s)**	**System**	**Time (s)**
Blaze	OOM	GCRClean-32-Node	8.09×	noBundle	1.35×
MiniGraph	65.34×	MiniClean-2-GPU	0.64×	noPipelinedPar	1.26×
Hyperblocker	11.92×	MiniClean-4-GPU	0.48×	noIndPar	1.12×
MiniClean-1-GPU	259.6s	MiniClean-1-GPU	4251.08s	MiniClean-2-GPU	2993.7s

- **Single-machine**: 65.34× faster than **MiniGraph**; 11.92× faster than **Hyperblocker**.
- **Multi-machine**: 8.09× faster than 32-node **GCRClean**.
- **Scalability**: speedup by 1.56× with **2 GPUs**; 2.08× with **4 GPUs**.

Better performance and good scalability

System Evaluation: Efficiency & Scalability

Single-machine (BioGRID)		Scalability (SemScholar)		Ablation study (SemScholar)	
System	**Time (s)**	**System**	**Time (s)**	**System**	**Time (s)**
Blaze	OOM	GCRClean-32-Node	8.09×	noBundle	1.35×
MiniGraph	65.34×	MiniClean-2-GPU	0.64×	noPipelinedPar	1.26×
Hyperblocker	11.92×	MiniClean-4-GPU	0.48×	noIndPar	1.12×
MiniClean-1-GPU	259.6s	MiniClean-1-GPU	4251.08s	MiniClean-2-GPU	2993.7s

- **Single-machine**: 65.34× faster than **MiniGraph**; 11.92× faster than **Hyperblocker**.
- **Multi-machine**: 8.09× faster than 32-node **GCRClean**.
- **Scalability**: speedup by 1.56× with **2 GPUs**; 2.08× with **4 GPUs**.

Better performance and good scalability

Welcome to play with MiniClean!