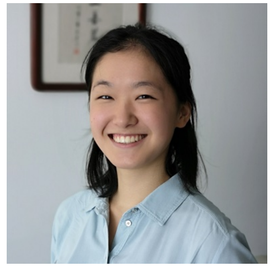
Weave

Efficient and Expressive Oblivious Analytics at Scale

Mahdi Soleimani



Grace Jia

Anurag Khandelwal

Yale

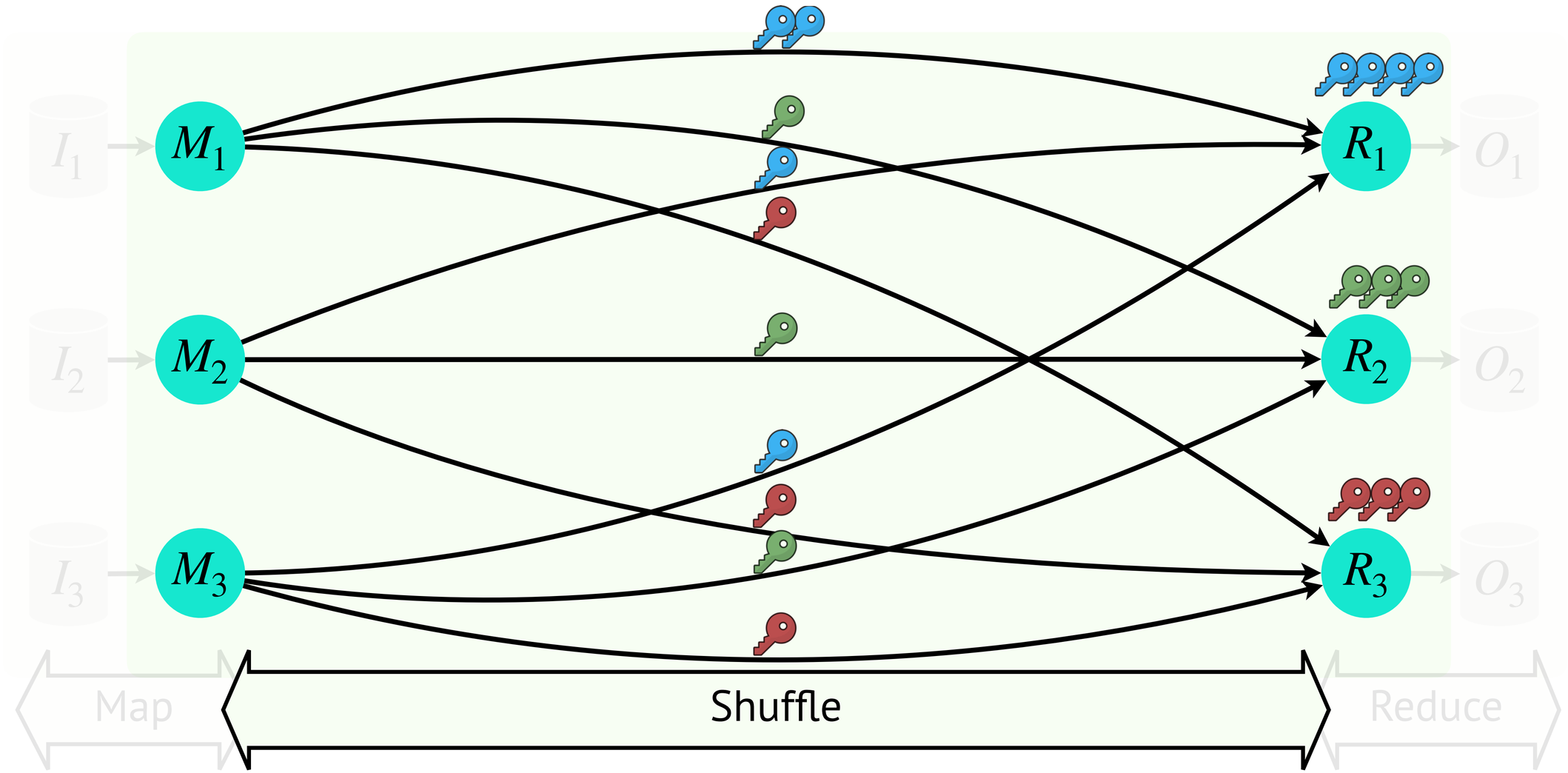
MapReduce

Mappers produce the <key, value> pairs from the input data

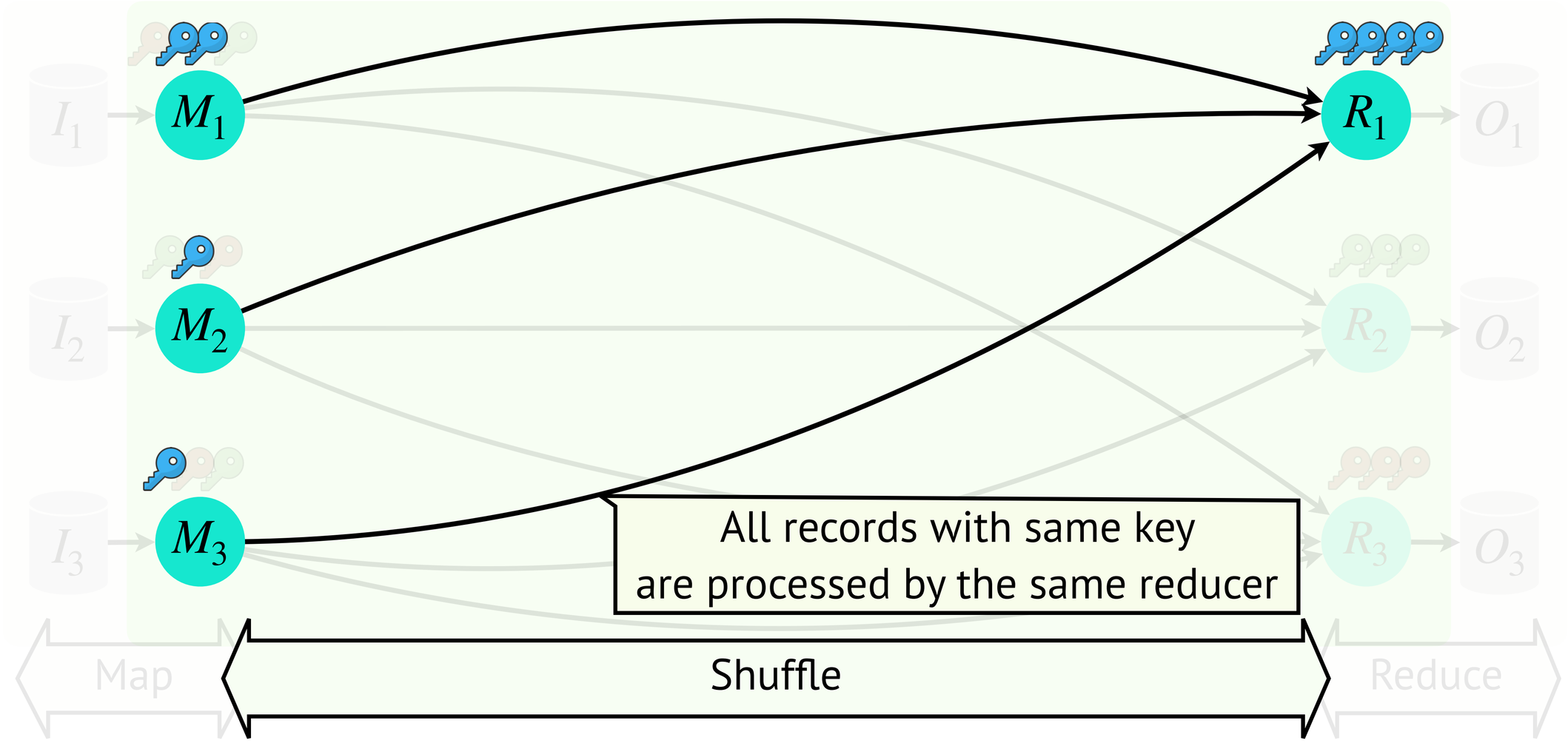
Reducers process the map output records for each key

MapReduce

Records are shuffled by key among the reducers



MapReduce


I_1
I_2
I_3
M_1
M_2
M_3
R_1
R_2
R_3
O_1
O_2
O_3
All records with same key
are processed by the same reducer
Shuffle
Map
Reduce

Secure MapReduce

Requires **confidential execution** of jobs on highly sensitive data.

CVS Health

STATE STREET

Secure MapReduce

Data encryption

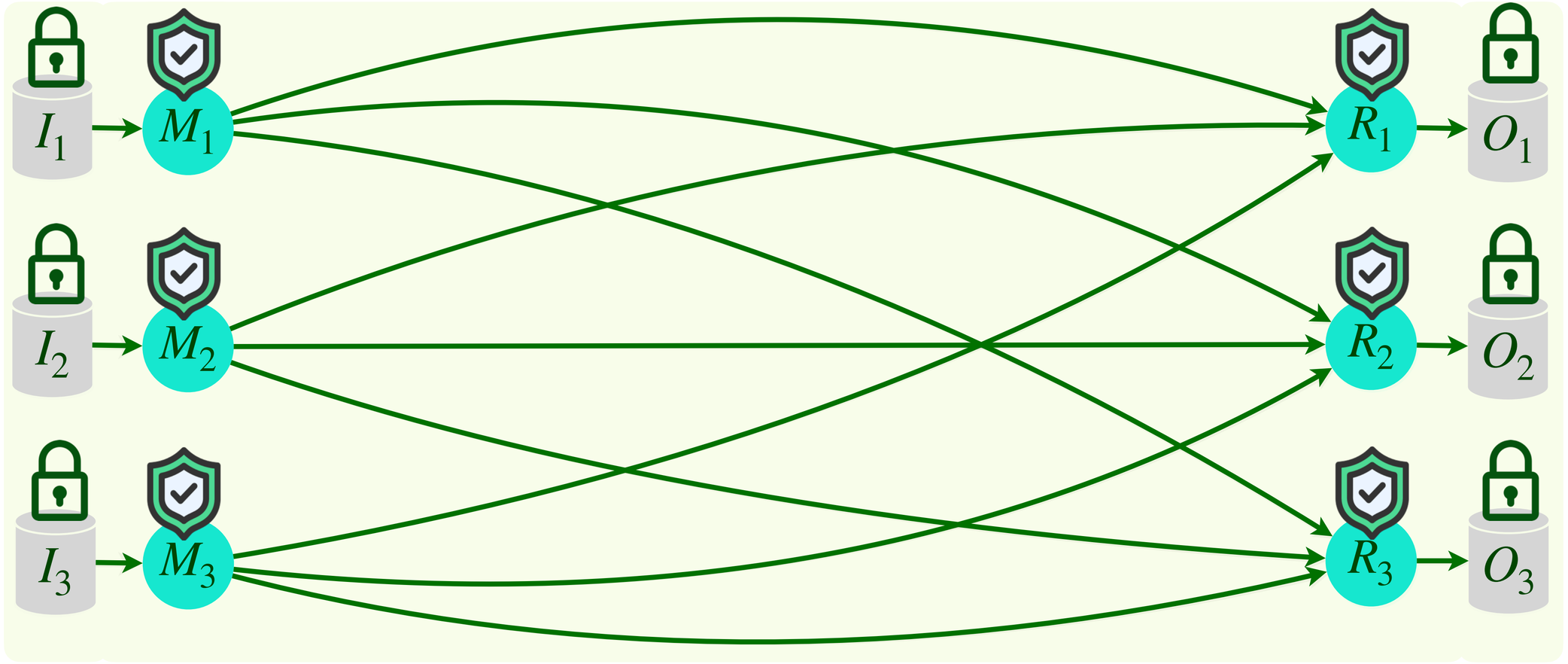
Protects data **at rest**

Trusted Execution

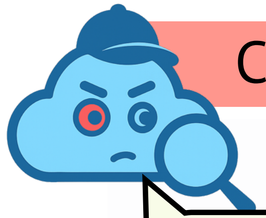
Provides **execution integrity** and **confidentiality**

Encrypted Transfer

Protects **in-flight** data



Secure MapReduce



Cloud Adversary

Honest But ***Curious***

Observes (but does not ***modify***) traffic between workers

Access Pattern Leakage:

Encrypted traffic traces can be used to ***identify private user data***

Access Pattern Leakage

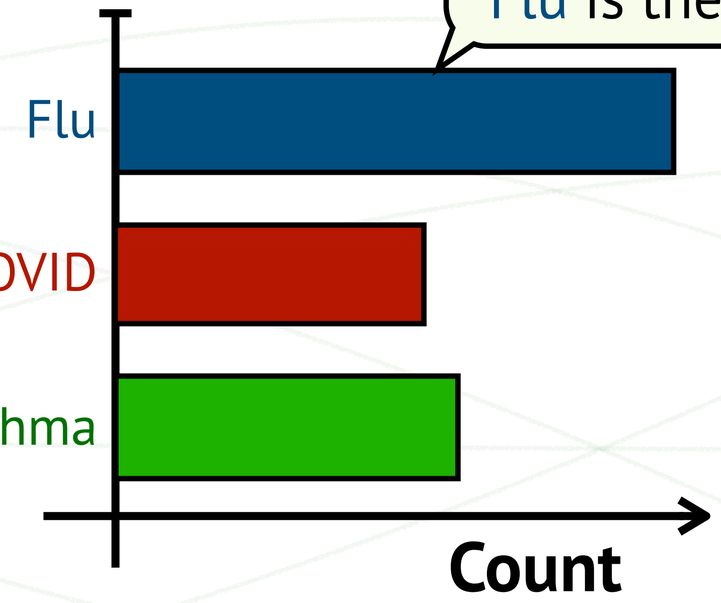
Job: *Count* Records by *Disease*

Disease data analytics

2018 data M_1


2019 data M_2


2020 data M_3



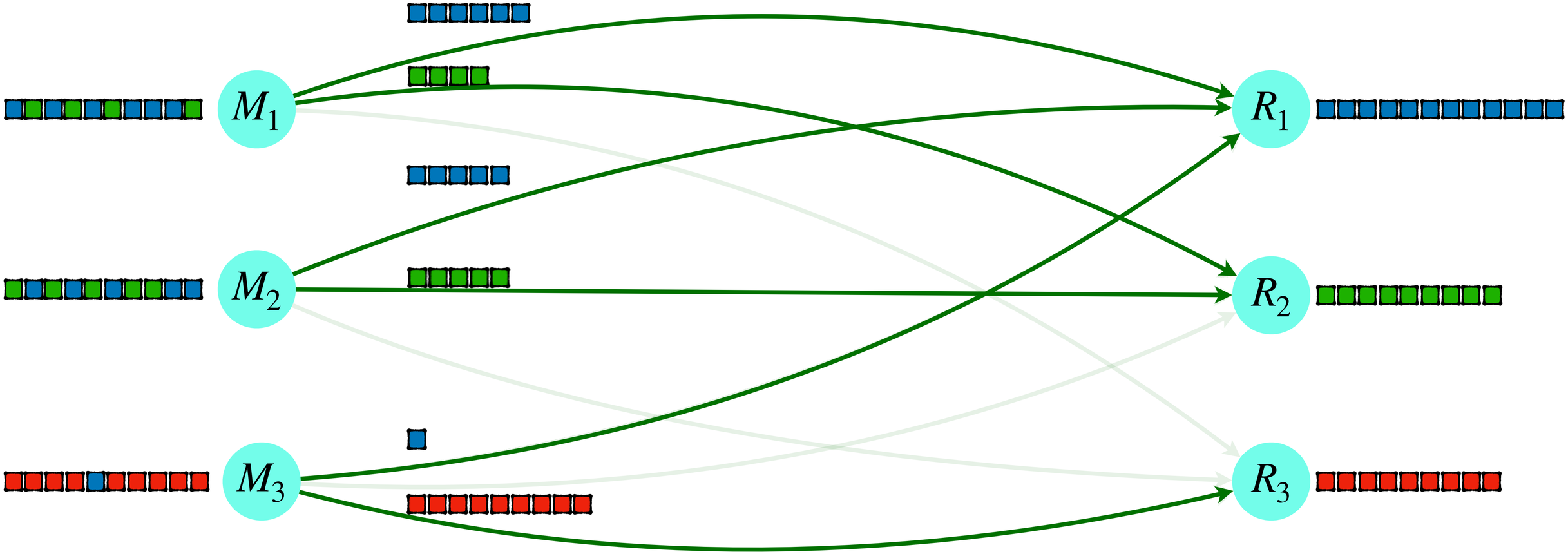
Flu is the most common disease

Prior Knowledge

COVID records appear only in 2020

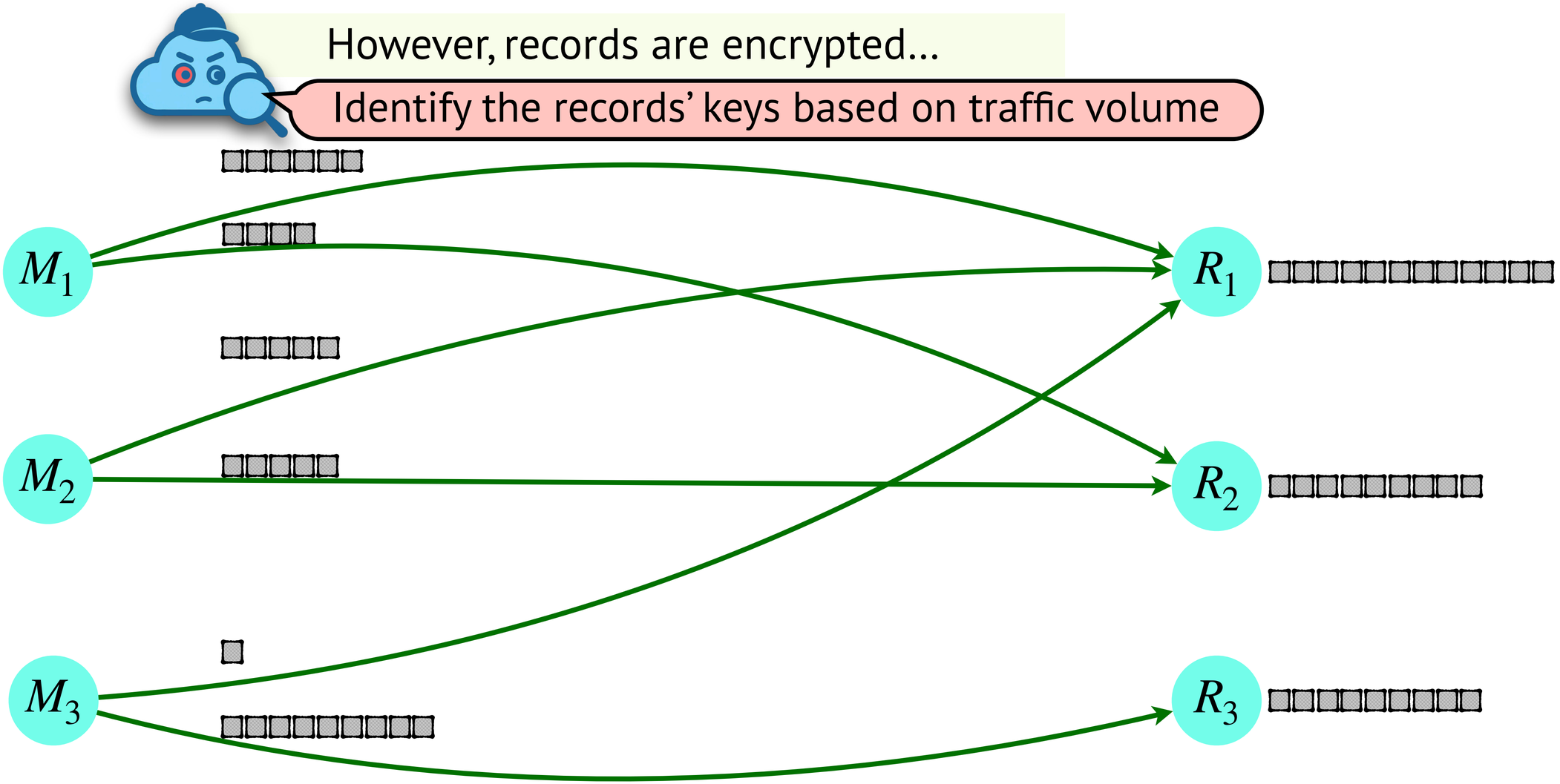
Access Pattern Leakage

Aggregate records by disease at reducers



Disease Flu Asthma COVID

Access Pattern Leakage

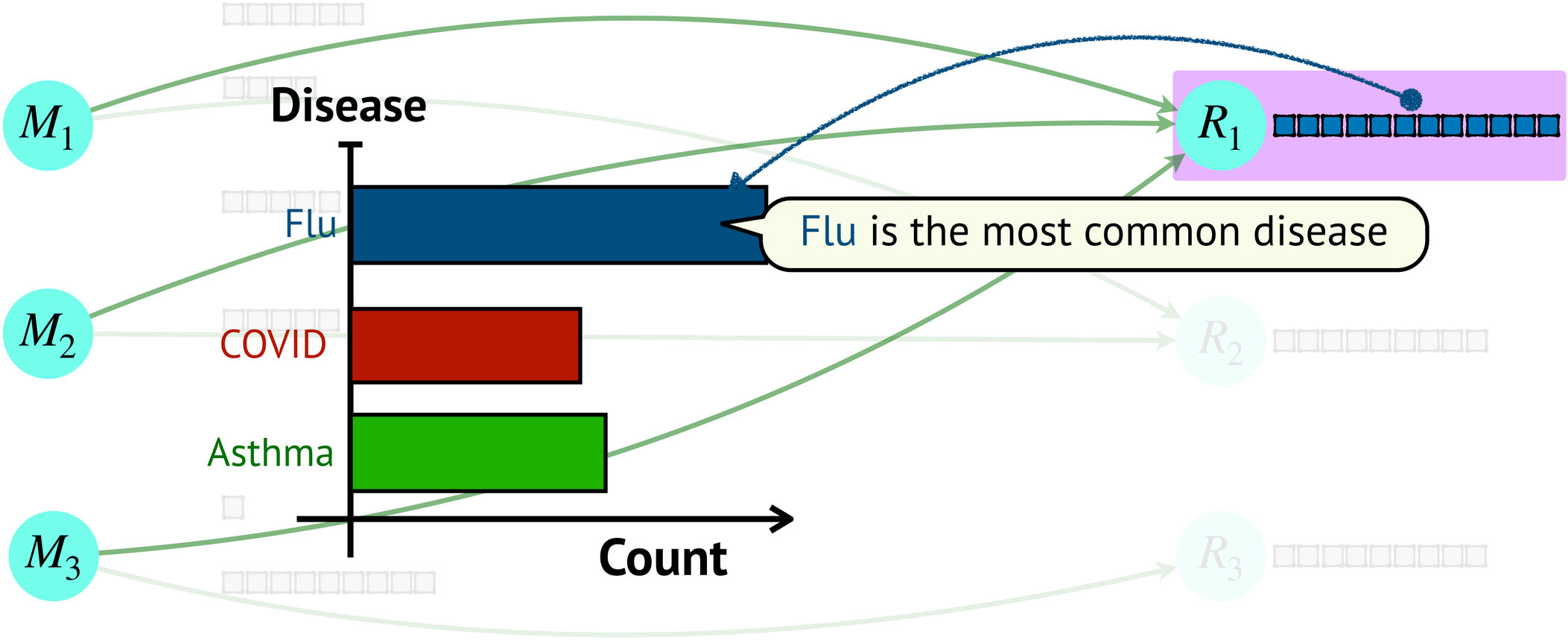


Encrypted **Disease** Flu Asthma COVID

Access Pattern Leakage

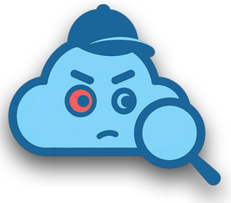
Distribution Based Leakage: Traffic volume to reducers depends on *data distribution*

Reducer R_1 receives most records → R_1 is processing the **Flu** records



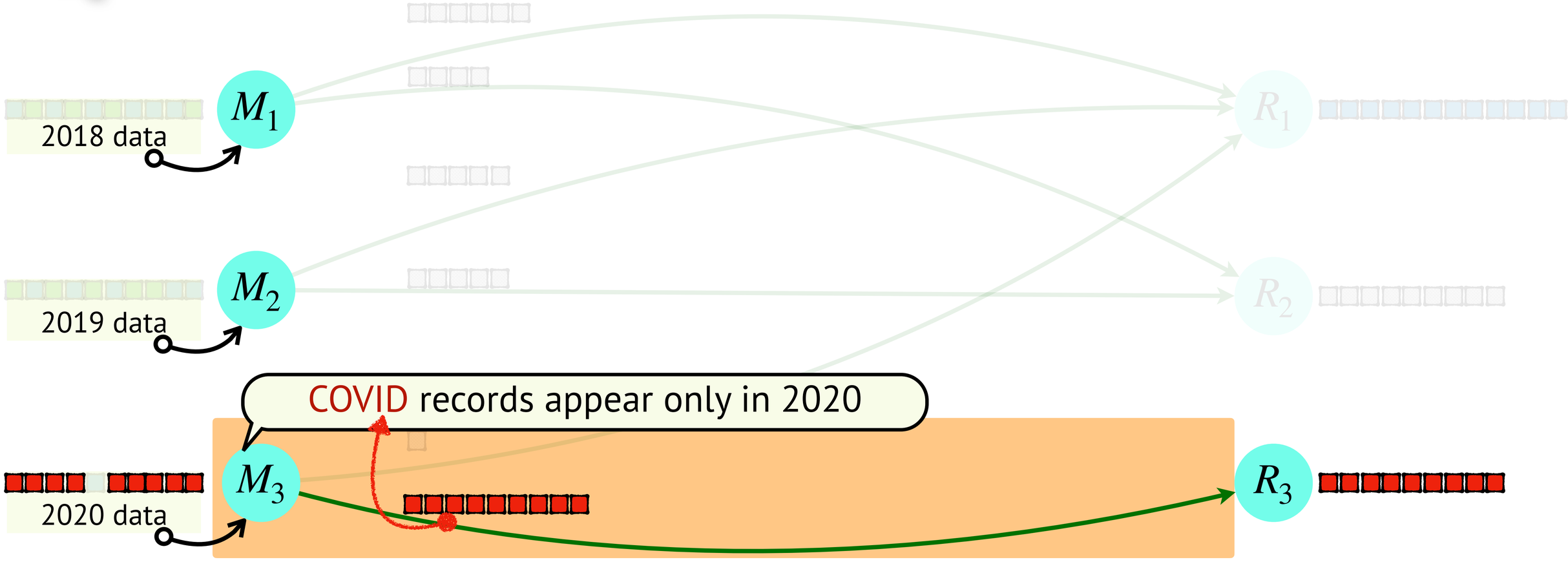
Encrypted | **Disease** | Flu | Asthma | COVID

Access Pattern Leakage



Order Based Leakage: Map output sizes depend on input data *order*

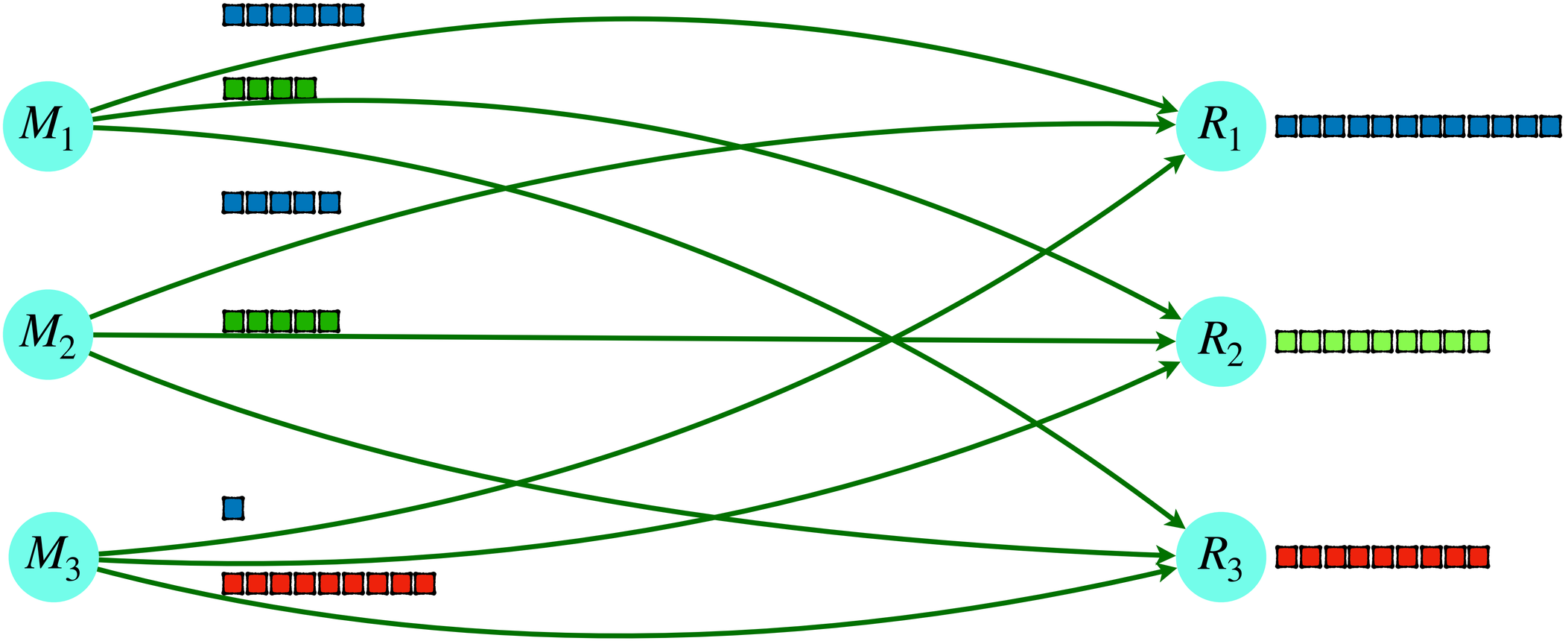
R_3's records are exclusively from M_3 ⟶ R_3 is processing **COVID** records



Encrypted | **Disease** | Flu | Asthma | COVID

Preventing Access Pattern Leakage

Full dataset reconstruction [ISC'19, CCS'20, CCS'15, NSDI'17]



Encrypted **Disease** Flu Asthma COVID

Preventing Access Pattern Leakage

Obliviousness ⇒ ***Traffic*** patterns between workers ***independent*** of dataset

Prevent ***distribution based leakage*** and ***order based leakage***

Encrypted **Disease** Flu Asthma COVID

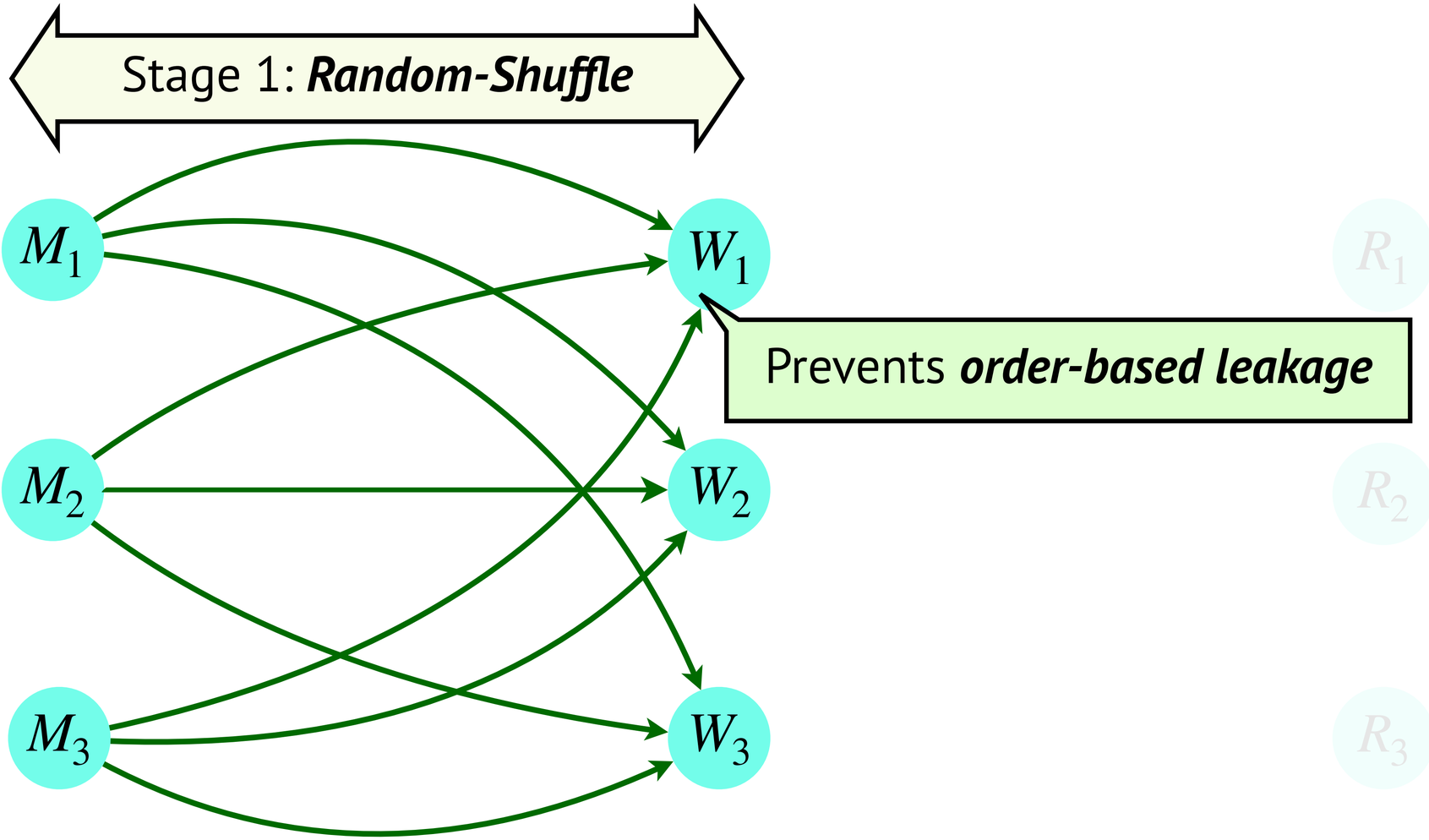
Prior Solutions

	Oblivious Sort Based [Opaque, NSDI' 17]	**Oblivious Shuffle Based** [SnB, CCS' 15]
	Distributed Circuit Sort	**Melbourne Shuffle**
Privacy	Obliviousness via data ***distribution agnostic*** shuffling	
Overhead	$\mathcal{O}(\log(\hat{n}))$, **44x in practice**	$\mathcal{O}(\log(\hat{n}))$, **17x in practice**
Functionality	Only associative jobs	No sort-based jobs

Can a ***secure solution*** achieve a ***small constant factor*** *overhead without limiting* ***functionality****?*

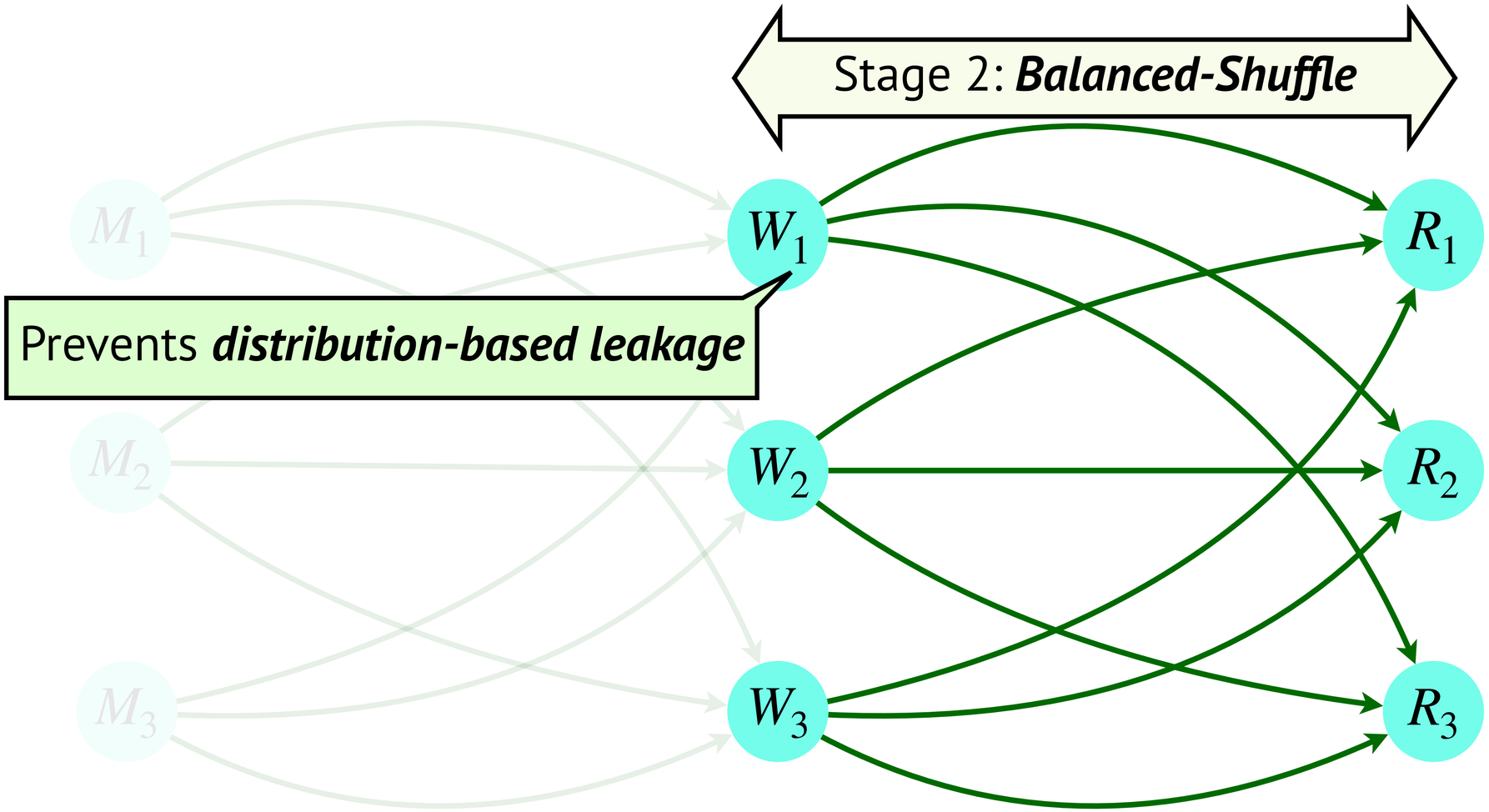
Weave

Key Idea: Obliviousness with distribution-aware shuffling

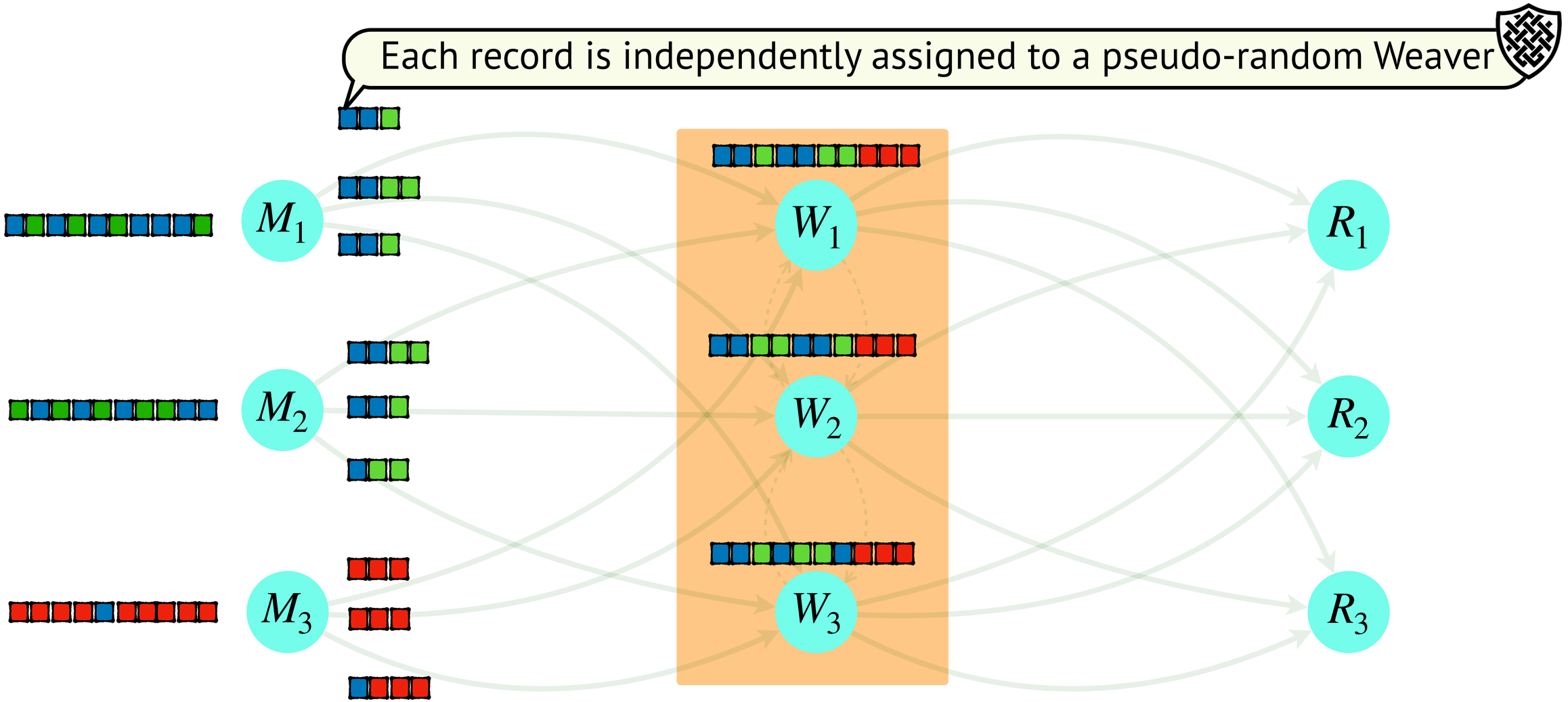


Weave

Key Idea: Obliviousness with distribution-aware shuffling



Random-Shuffle

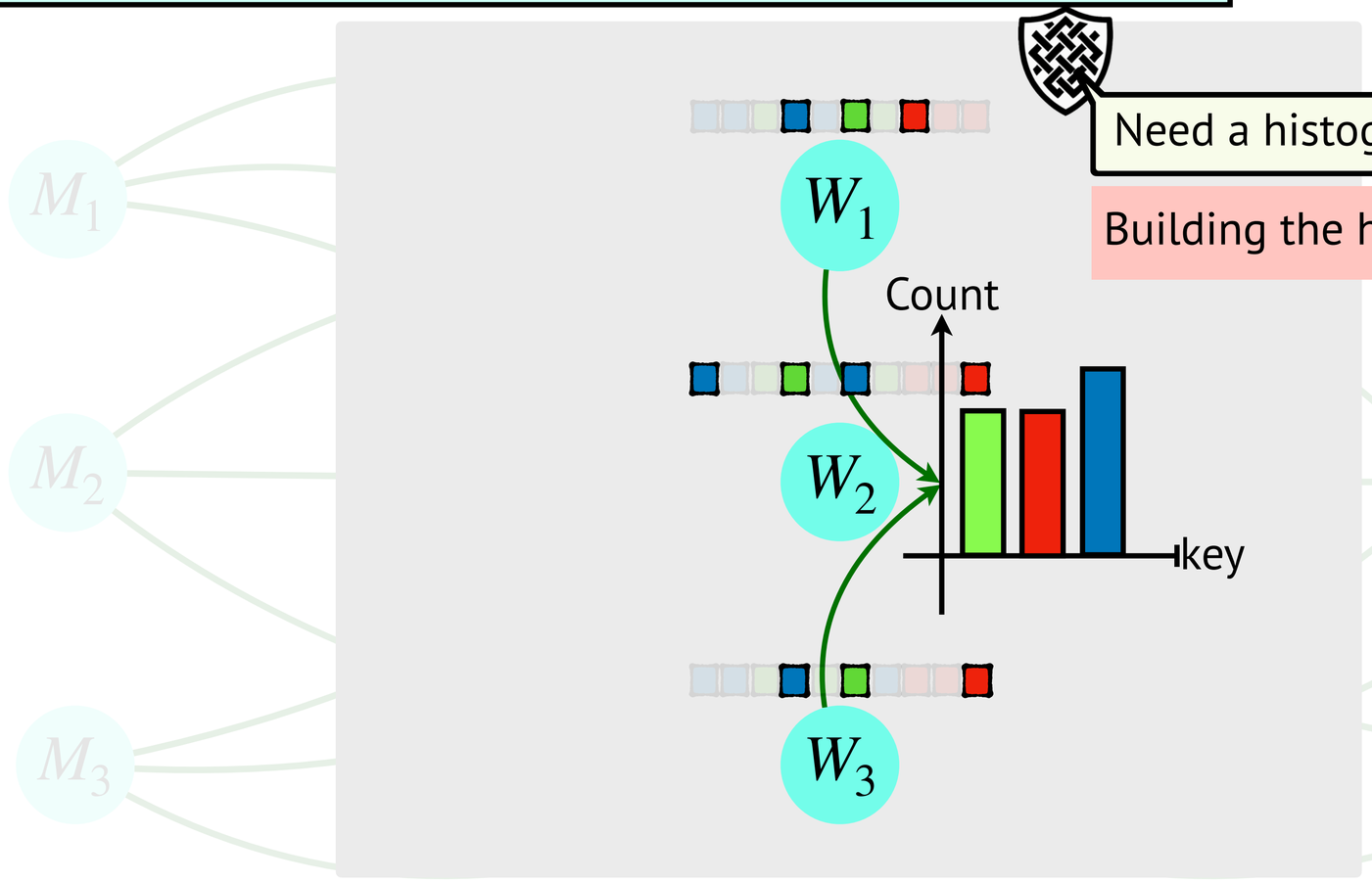


Distribution of records on Weavers is **identical**

Preventing ***order-based leakage***

Balanced-Shuffle

Distribution-aware bin packing of real + fake records



Need a histogram of keys across ***all*** Weavers

Building the histogram does not scale!

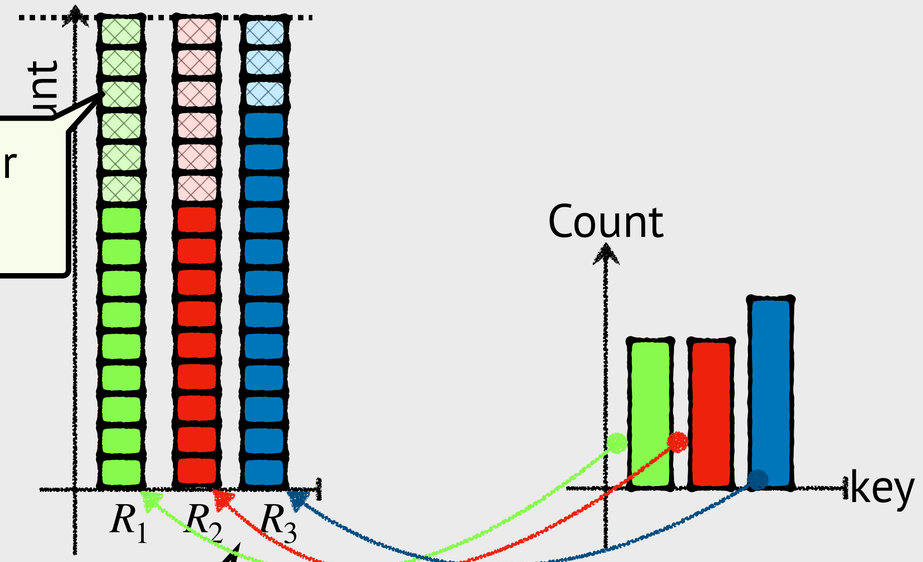
See paper for proof!

Weave **samples 1%** of the records: ensures ***scalability*** without compromising ***security***

Balanced-Shuffle

Distribution-aware **bin packing of real + fake records**

"Fake" records to ensure each reducer receives same number of records



Assign keys to reducers to ***balance number of real records*** across them for efficiency

M_2

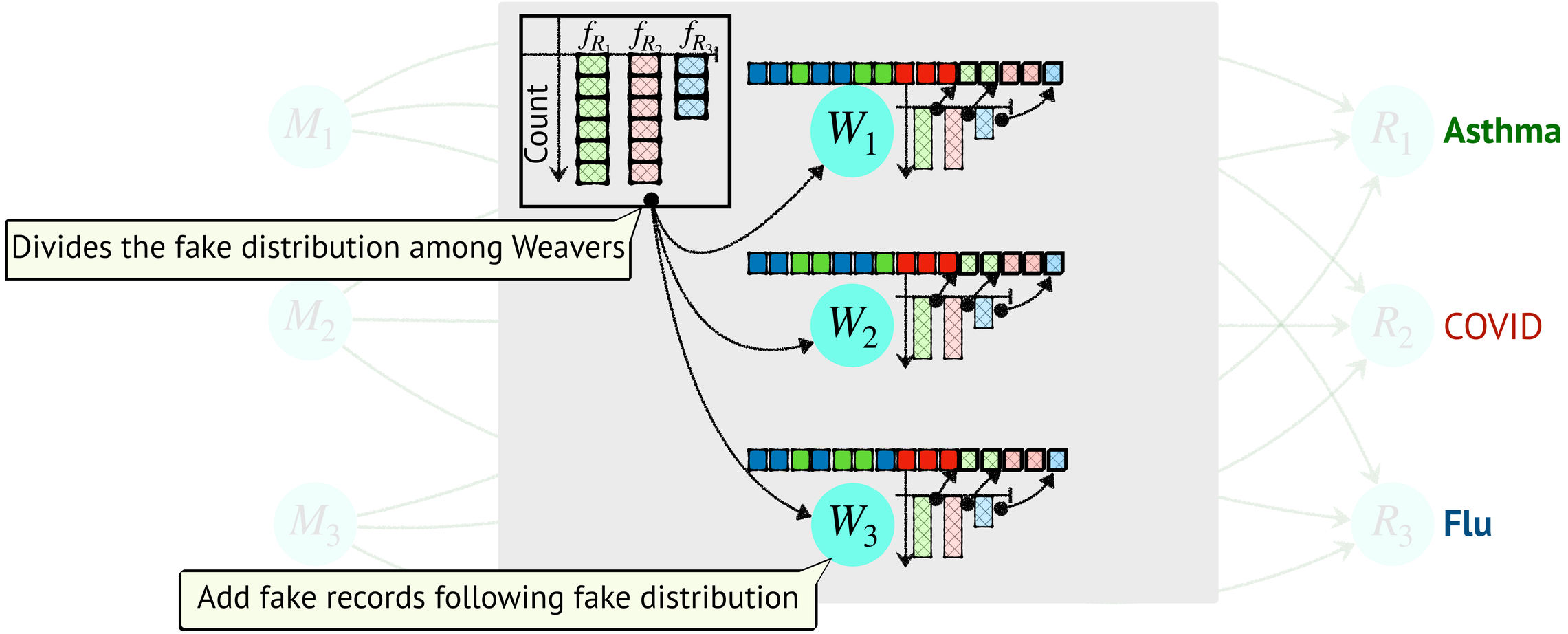
R_1 **Asthma**

R_2 COVID

R_3 **Flu**

The number of records is a ***constant-factor*** multiple of the original, providing ***security*** and ***efficiency***

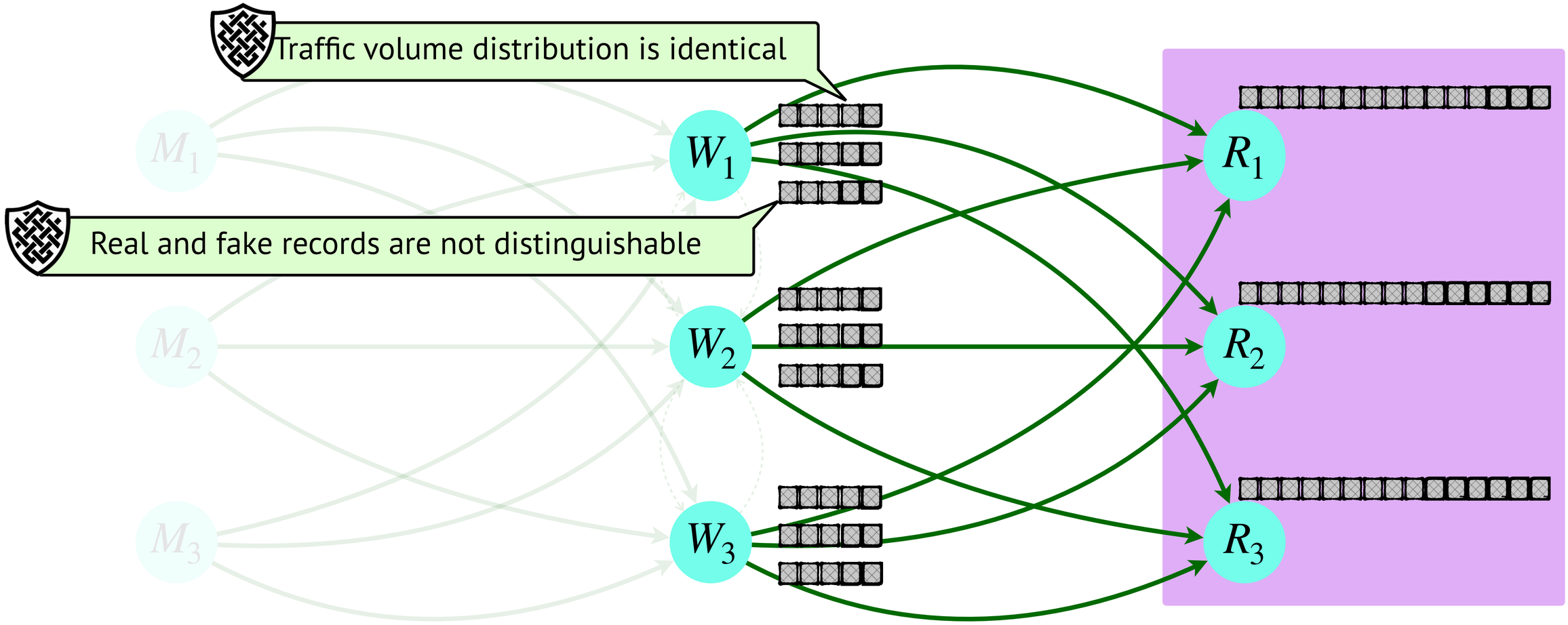
Balanced-Shuffle



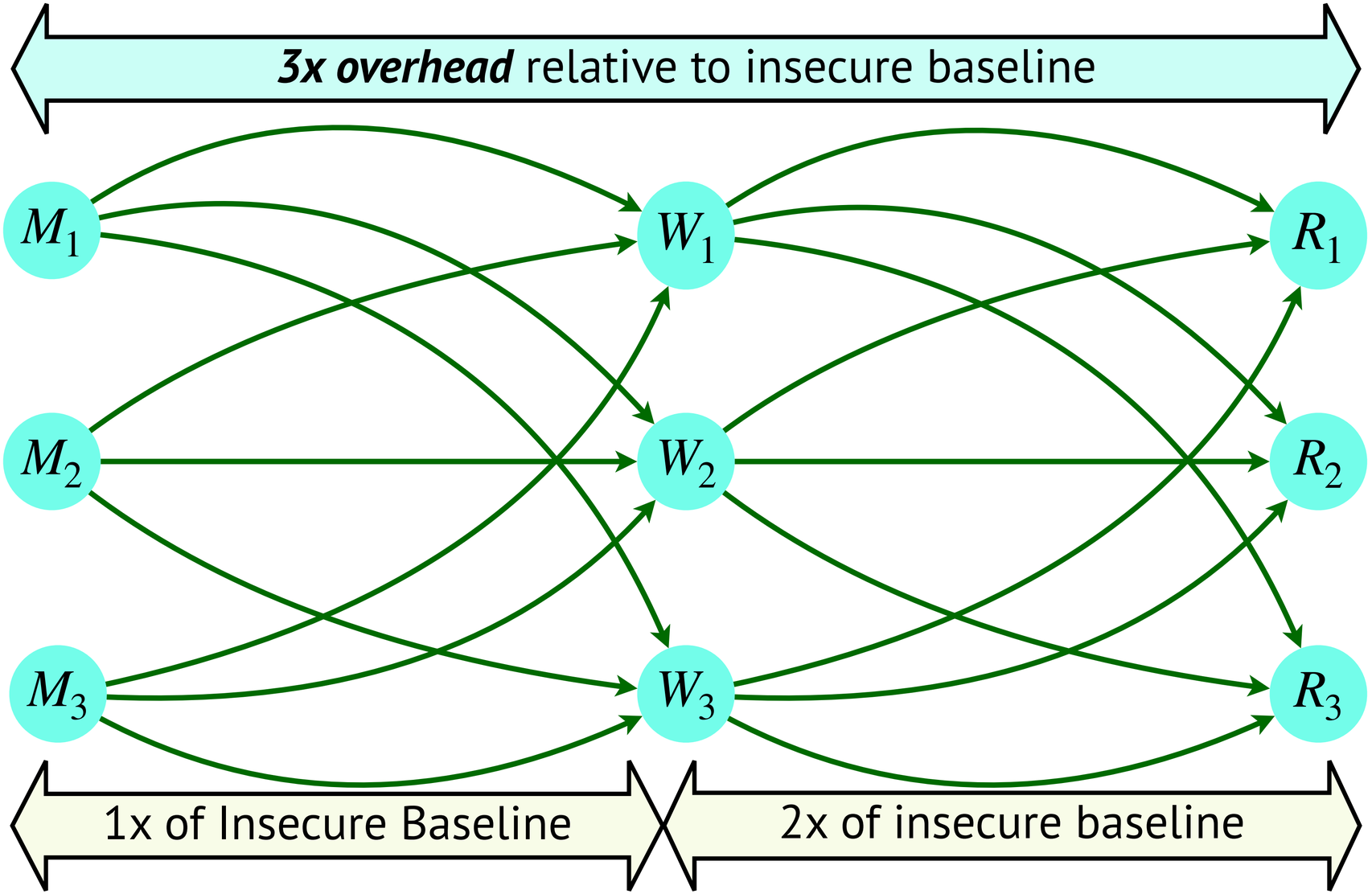
Balanced-Shuffle

Each reducer process same number of records

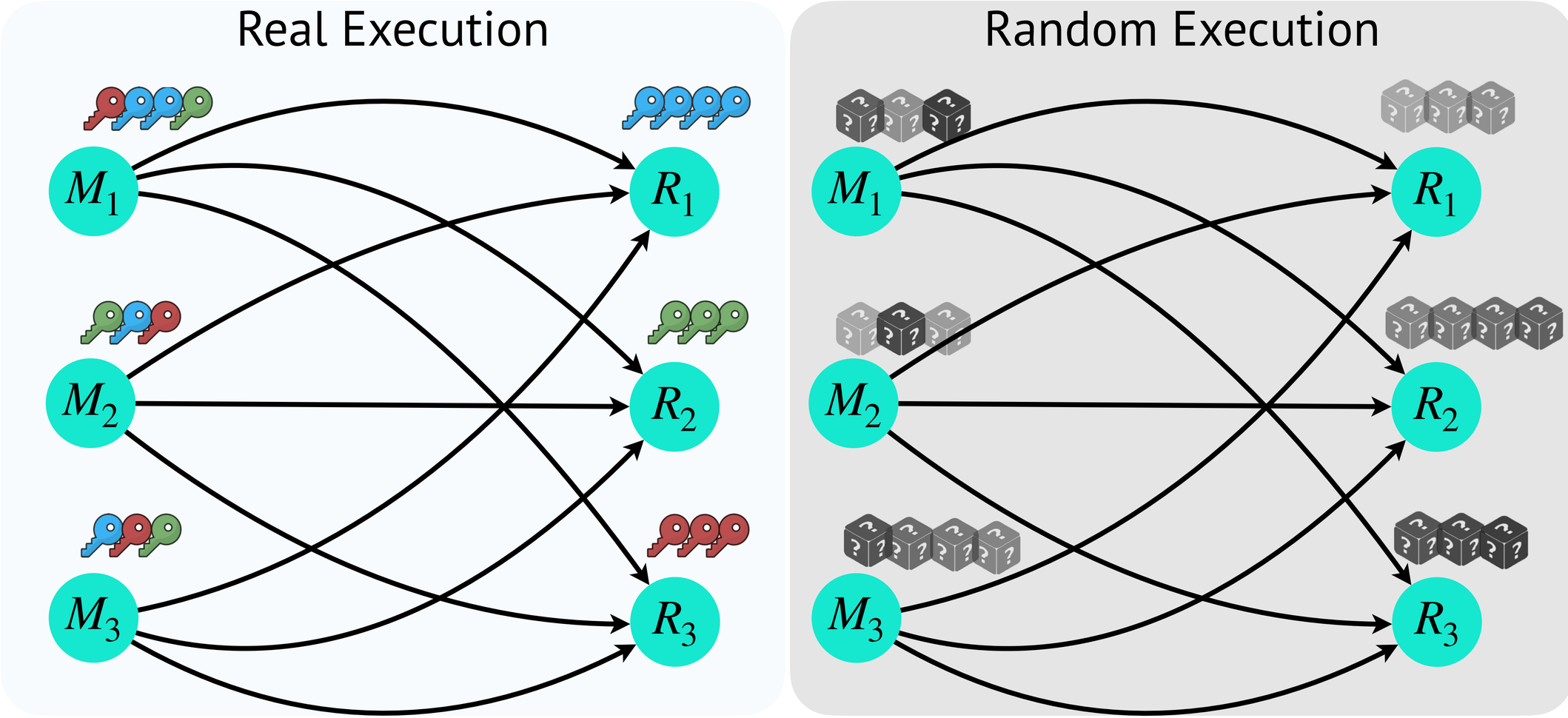
Preventing **Distribution Based** Leakage



Weave Overheads

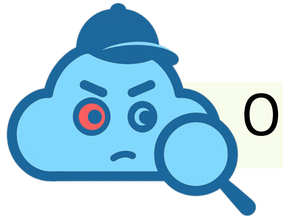


Weave Security:
Real vs Random Execution Indistinguishability



Weave Security:

INDistinguishability under **C**hosen **D**ataset and **J**ob **A**ttack **(IND-CDJA)**



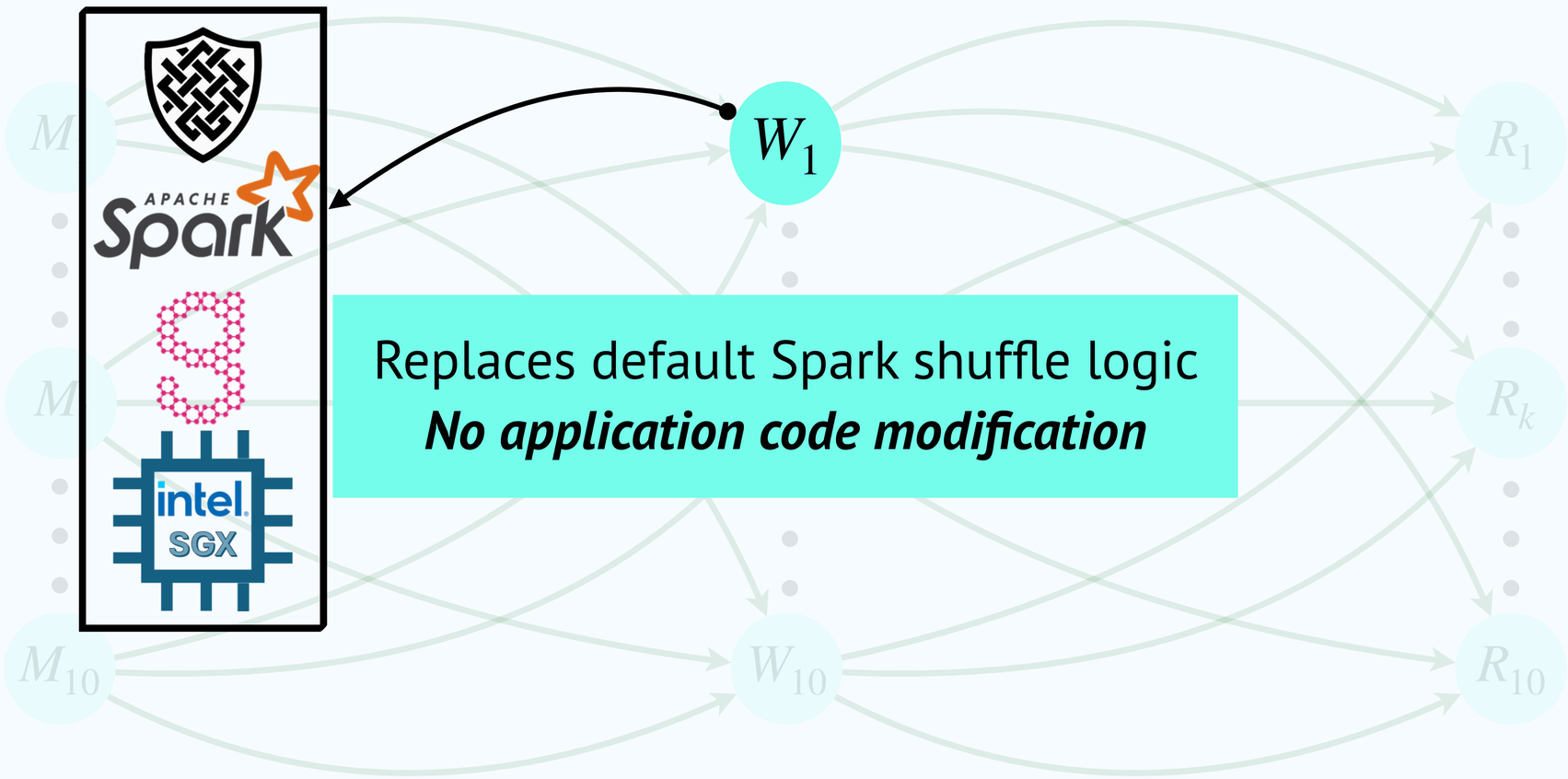
Observes and compares the communication traffic

System is secure when
Real (System)
is indistinguishable from
Random

Weave is IND-CDJA Secure: Identical traffic distributions *for real & random datasets*

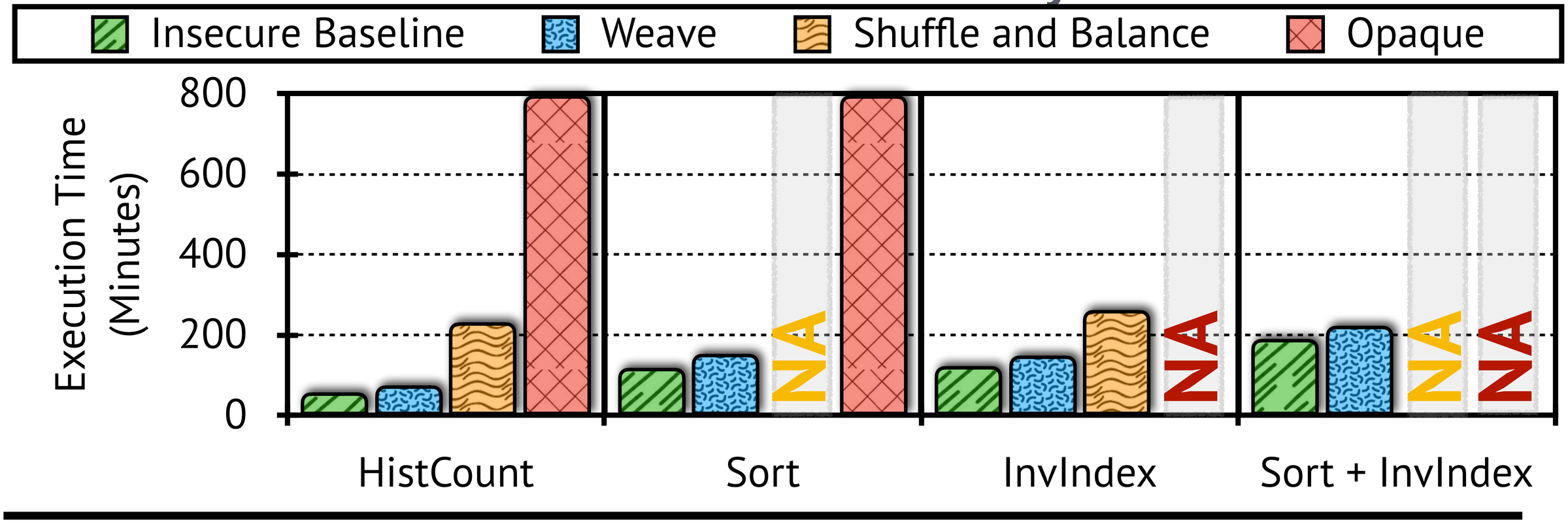
Weave Implementation & Evaluation Setup

Nodes: 20 Azure DCsv3 VMs (16GB RAM, 4 vCPUs)
Network: 10Gbps links with TLS-encrypted traffic



Replaces default Spark shuffle logic
No application code modification

Weave Efficiency

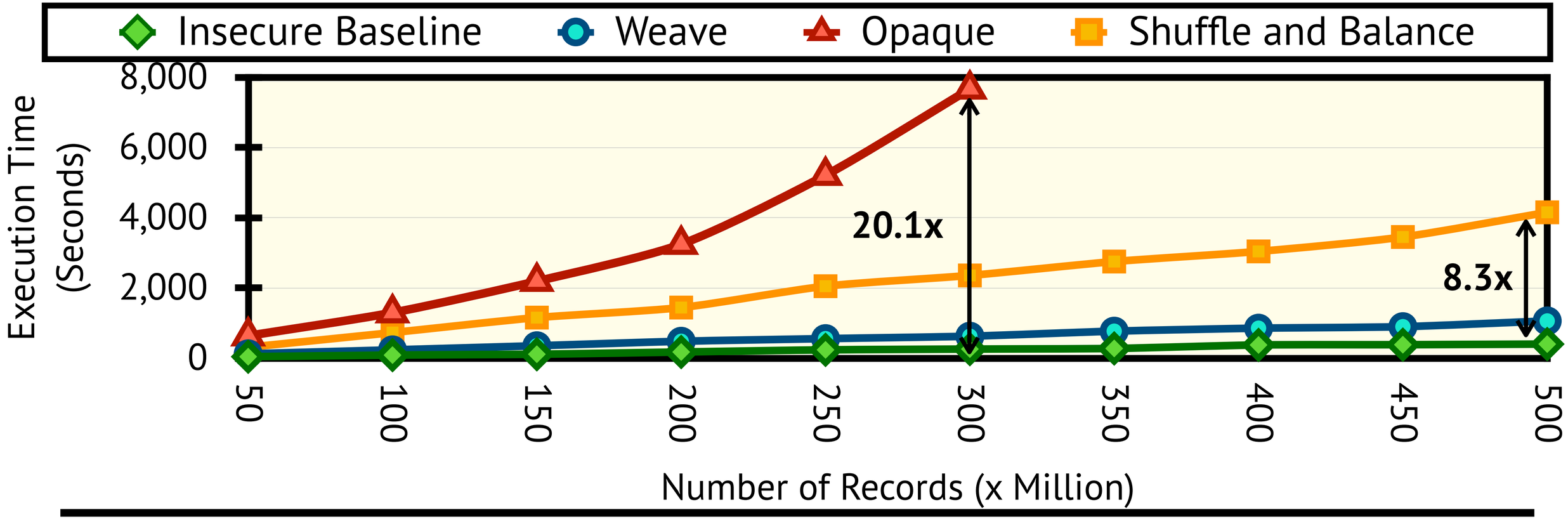


Takeaways

*Weave incurs **<40%** overhead over an insecure baseline*

***6 - 14x** faster than prior approaches* *without limiting functionality*

Weave Scalability with Input Size

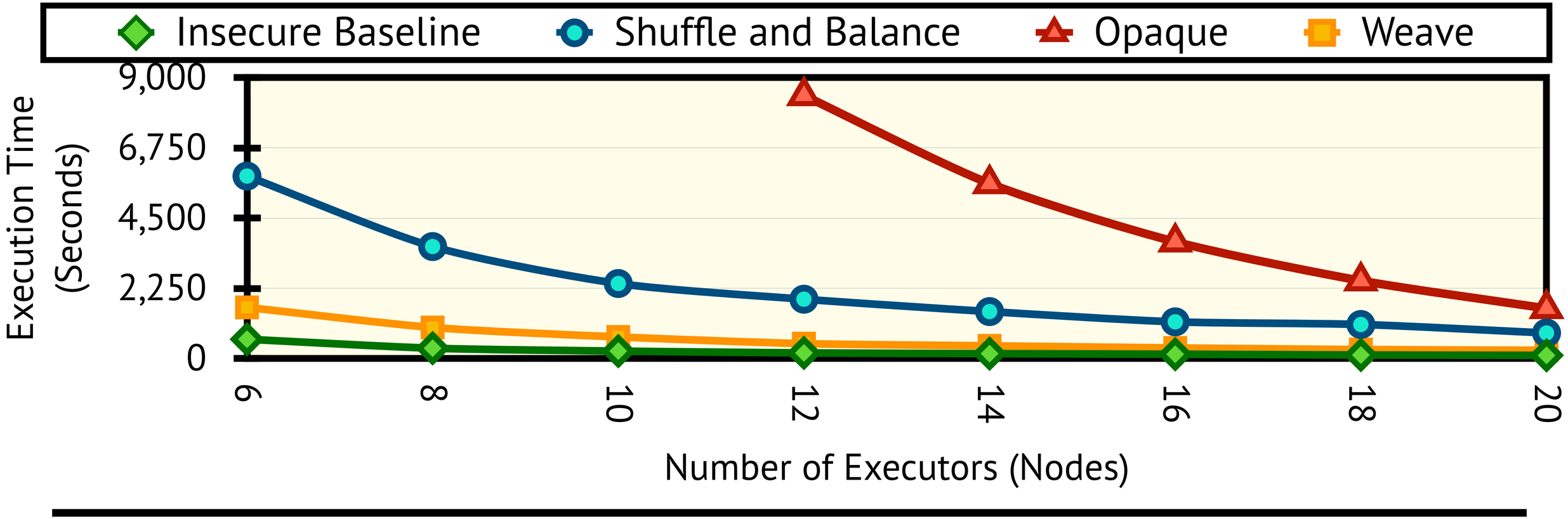


Takeaways

*Weave scales linearly with input size **due to its constant overhead***

With up to 8.3 - 20.1x faster execution at scale

Weave Scalability with Cluster Size



Takeaways

Weave also scales linearly ***with cluster size***

Conclusion

- **Oblivious MapReduce** with **constant overhead**
- **Data distribution aware noise injection** for **security** and **performance**
- Order of magnitude **lower execution time** than prior solutions without compromising on **functionality**
- Code available at https://github.com/yale-nova/weave