Register automata as a model for local computation in distributed query evaluation

Tony Tan
National Taiwan University

Collaboration with: Jonny Daenen, Frank Neven, Frederic Servais, Jan Van den Bussche (Hasselt University), Nicole Schweikardt (Humboldt University, Berlin), Stijn Vansummeren (Free University of Brussels)

Popular distributed systems

Systems in which parallelisation is achieved via key-value paradigm.

- Apache Hadoop, which is based on Google's map-reduce.

  [Yahoo's Hadoop; Dean and Ghemawat OSDI'04]

- Spark.

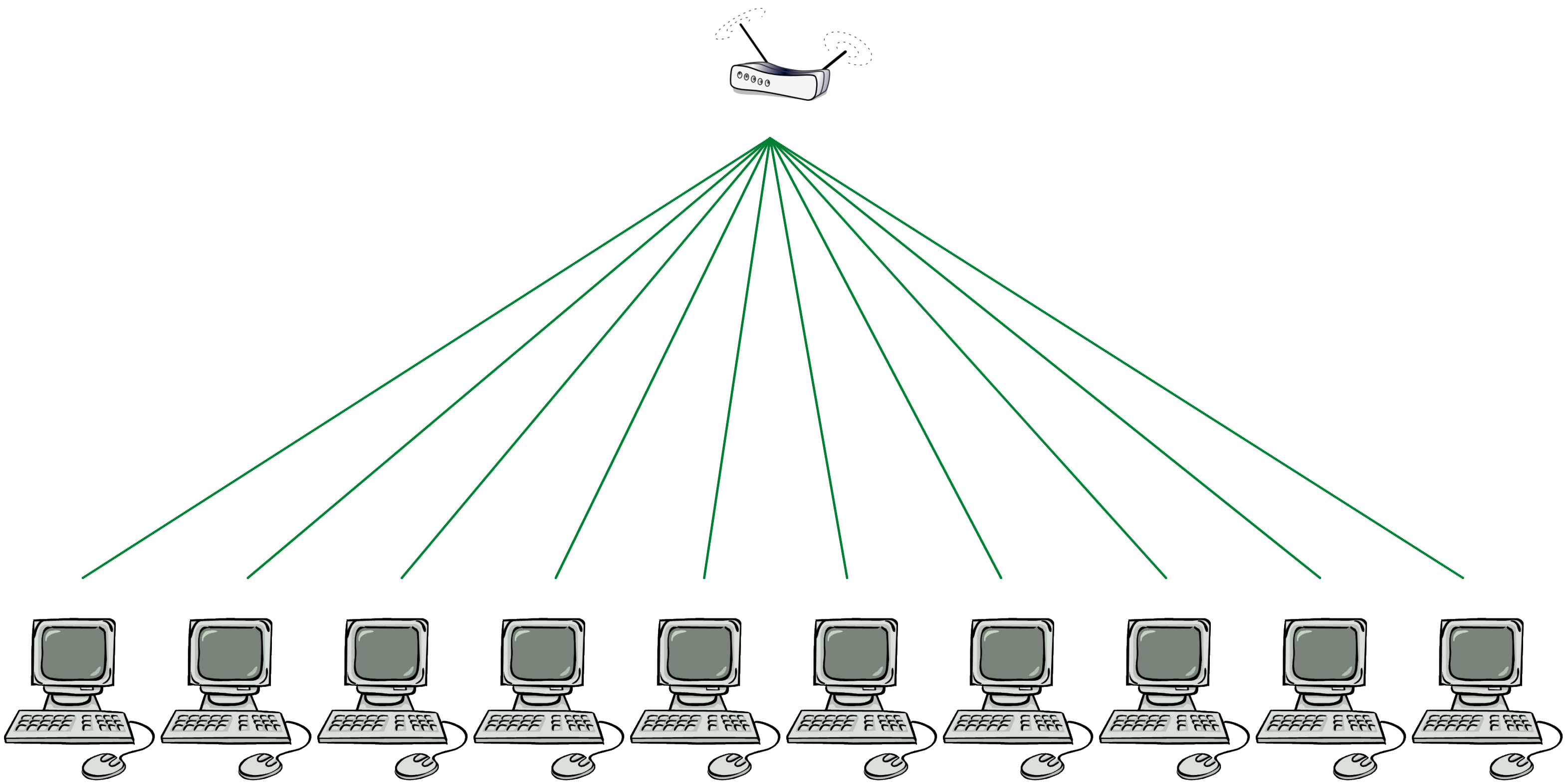
  [Zaharia, et. al. NSDI'12; Zaharia's PhD thesis 2014]

Hadoop and Spark

Some reasons for their popularity:

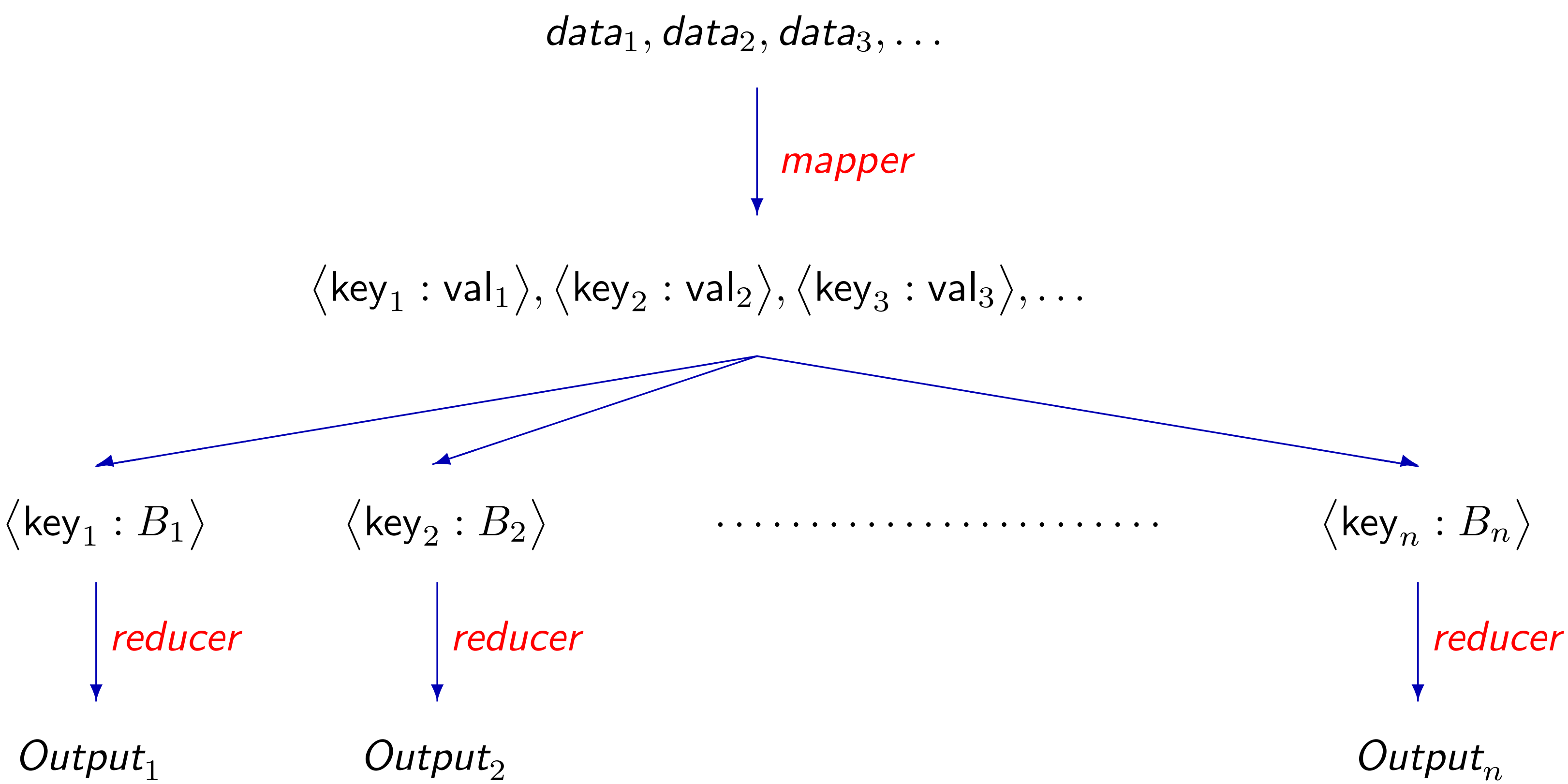
- They are free.
- They take care of the communication.
- Programmers only specify the local computation.
- Do not require special network architecture.

Hadoop and Spark



Protocol: Local computation, communication, local computation, communication, . . .

Key-value paradigm in Hadoop



Key-value paradigm in Spark using RDD (Resilient Distributed Datasets)

D :	$data_1$	$data_2$	$data_3$	$\cdots\cdots\cdots$
		`E = D.mapToPair(<func>)`		
E :	$\langle \mathsf{key}_1 : \mathsf{val}_1 \rangle$	$\langle \mathsf{key}_2 : \mathsf{val}_2 \rangle$	$\langle \mathsf{key}_3 : \mathsf{val}_3 \rangle$	$\cdots\cdots\cdots$
		`F = E.reduceByKey(<func>)`		
F :	$\langle \mathsf{key}_1 : v_1 \rangle$	$\langle \mathsf{key}_2 : v_2 \rangle$	$\langle \mathsf{key}_3 : v_3 \rangle$	$\cdots\cdots\cdots$

Brief review on relational algebra (core of SQL)

- *Union*: $A \cup B$.
- *Intersection*: $A \cap B$.
- *Difference*: $A - B$.
- *Projection*: $\pi_{i_1,\ldots,i_k}(A)$.
- *Selection*: $\sigma_{i=j}(A)$.
- *Semijoin*: $A \ltimes_{i=j} B$.

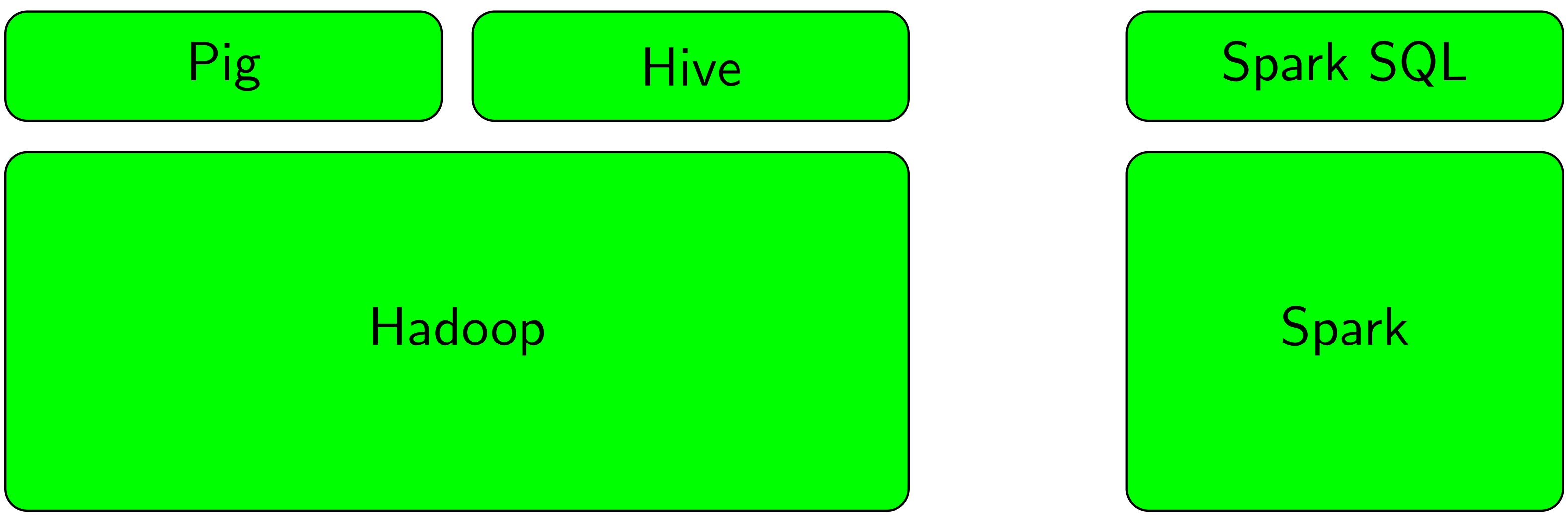
$$A \ltimes_{i=j} B \quad := \quad \Big\{ \bar{a} \in A \ \mid \ \text{there is } \bar{b} \in B \text{ such that } a_i = b_j \Big\}$$

- *Join*: $A \bowtie_{i=j} B$.

$$A \bowtie_{i=j} B \quad := \quad \Big\{ (\bar{a}, \bar{b}) \in A \times B \ \mid \ \ a_i = b_j \Big\}$$

Database engines on top of Hadoop and Spark

- Pig [Gates, et. al. VLDB'09; Olston, et. al. SIGMOD'08]
- Hive [Thusoo, et. al. ICDE'10]
- Shark and Spark SQL [Xin, et. al. SIGMOD'13 & '15]



Queries on Pig/Hive/Spark SQL

In general, users don't need to mention:

- the number of available servers,
- how to distribute and partition the data.

Users write queries as if on a single processor.

Pig/Hive/Spark SQL

Question: What are the necessary and sufficient *local* computation to evaluate relational algebra?

Main theme of the talk

Three models of computation in key-value paradigm.

- DSA (Distributed Streaming with register Automata)
- DST (Distributed Streaming with register Transducers)
- DSTJ (Distributed Streaming with register Transducers and Joins)

The mappers are "generic mappers," i.e., projection + finite tags.

The reducers are various models with limited computation that process the values in streaming fashion using finite memory.

DSA, DST, DSTJ

$$\textit{tuple}_1, \textit{tuple}_2, \textit{tuple}_3, \ldots$$

$$\Big\downarrow \; \color{red}{\textit{mapper} = \text{projection} + \text{finite tags}}$$

$$\big\langle \text{key}_1 : \text{val}_1 \big\rangle, \big\langle \text{key}_2 : \text{val}_2 \big\rangle, \big\langle \text{key}_3 : \text{val}_3 \big\rangle, \ldots$$

$$\big\langle \text{key}_1 : B_1 \big\rangle \qquad \big\langle \text{key}_2 : B_2 \big\rangle \qquad \cdots\cdots\cdots\cdots \qquad \big\langle \text{key}_n : B_n \big\rangle$$

$$\Big\downarrow \textit{reducer} \qquad \Big\downarrow \textit{reducer} \qquad \qquad \Big\downarrow \textit{reducer}$$

$$\textit{Output}_1 \qquad \textit{Output}_2 \qquad \qquad \textit{Output}_n$$

An example: $R(1,2) \mapsto \big\langle S(1) : T(1,2) \big\rangle$

Boolean DSA, DST, DSTJ:
$(\mathit{map}_1, \mathit{red}_1, \mathit{map}_2, \mathit{red}_2, \ldots, \mathit{map}_\ell, \mathit{red}_\ell, \mathit{agg})$

$$\mathit{data}_1, \mathit{data}_2, \mathit{data}_3, \ldots$$

$$\downarrow \mathit{map}_1$$

$$\langle \mathsf{key}_1 : \mathsf{val}_1 \rangle, \langle \mathsf{key}_2 : \mathsf{val}_2 \rangle, \langle \mathsf{key}_3 : \mathsf{val}_3 \rangle, \ldots$$

$$\langle \mathsf{key}_1 : B_1 \rangle \qquad \langle \mathsf{key}_2 : B_2 \rangle \qquad \cdots\cdots\cdots\cdots \qquad \langle \mathsf{key}_n : B_n \rangle$$

$$\downarrow \mathit{red}_1 \qquad \downarrow \mathit{red}_1 \qquad \qquad \downarrow \mathit{red}_1$$

$$\mathit{Output}_1^1 \qquad \mathit{Output}_2^1 \qquad \qquad \mathit{Output}_n^1$$

$$\downarrow \mathit{map}_2$$

$$\vdots$$

$$\mathit{Output}_1^\ell \qquad \mathit{Output}_2^\ell \qquad \qquad \mathit{Output}_n^\ell$$

$$\downarrow \mathit{agg}$$

yes/no

Non-Boolean DSA, DST, DSTJ:

$(\mathit{map}_1, \mathit{red}_1, \mathit{map}_2, \mathit{red}_2, \ldots, \mathit{map}_\ell, \mathit{red}_\ell, \cancel{\mathit{agg}})$

$\mathit{data}_1, \mathit{data}_2, \mathit{data}_3, \ldots$

$\downarrow \mathit{map}_1$

$\langle \mathsf{key}_1 : \mathsf{val}_1 \rangle, \langle \mathsf{key}_2 : \mathsf{val}_2 \rangle, \langle \mathsf{key}_3 : \mathsf{val}_3 \rangle, \ldots$

$\langle \mathsf{key}_1 : B_1 \rangle \quad \langle \mathsf{key}_2 : B_2 \rangle \quad \cdots\cdots\cdots \quad \langle \mathsf{key}_n : B_n \rangle$

$\downarrow \mathit{red}_1 \quad \downarrow \mathit{red}_1 \quad \downarrow \mathit{red}_1$

$\mathit{Output}_1^1 \quad \mathit{Output}_2^1 \quad \mathit{Output}_n^1$

$\downarrow \mathit{map}_2$

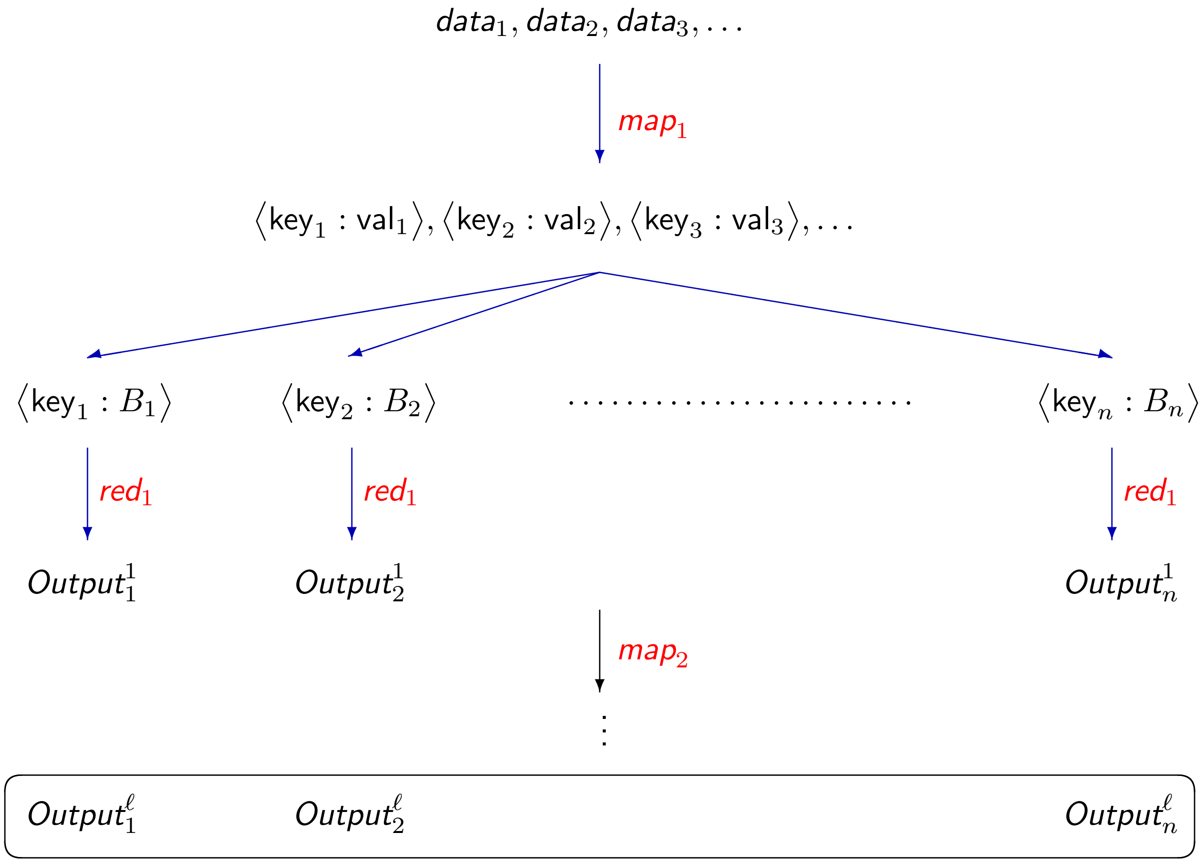
$\vdots$

$\mathit{Output}_1^\ell \quad \mathit{Output}_2^\ell \quad \mathit{Output}_n^\ell$

$\cancel{\downarrow \mathit{agg}}$

~~yes/no~~

Non-Boolean DSA, DST, DSTJ: $(map_1, red_1, map_2, red_2, \ldots, map_\ell, red_\ell)$



The big picture for DSA, DST and DSTJ

$$\begin{array}{ccccc} \text{DSA} & \subsetneq & \text{DST} & \subsetneq & \text{DSTJ} \\ \cup\!\mid & & \cup\!\mid & & \cup\!\mid \\ \text{Rel. alg. without join} & & \text{Semi-join algebra} & & \text{Relational algebra} \end{array}$$

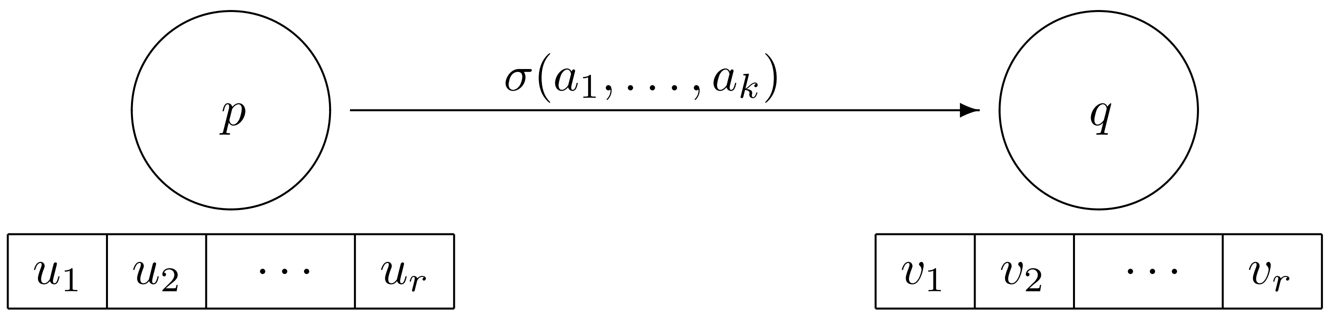
For Boolean case: DSA captures FO over bounded degree databases.

DSA: $(\mathit{map}_1, \mathit{red}_1, \mathit{map}_2, \mathit{red}_2, \ldots, \mathit{map}_\ell, \mathit{red}_\ell, \mathit{agg})$

- $\mathit{map}_1, \ldots, \mathit{map}_\ell$ are generic mappers, i.e., projections + finite tags.
- $\mathit{red}_1, \ldots, \mathit{red}_\ell$ are deterministic register automata "with output."
- agg is deterministic register automaton.

Register automata

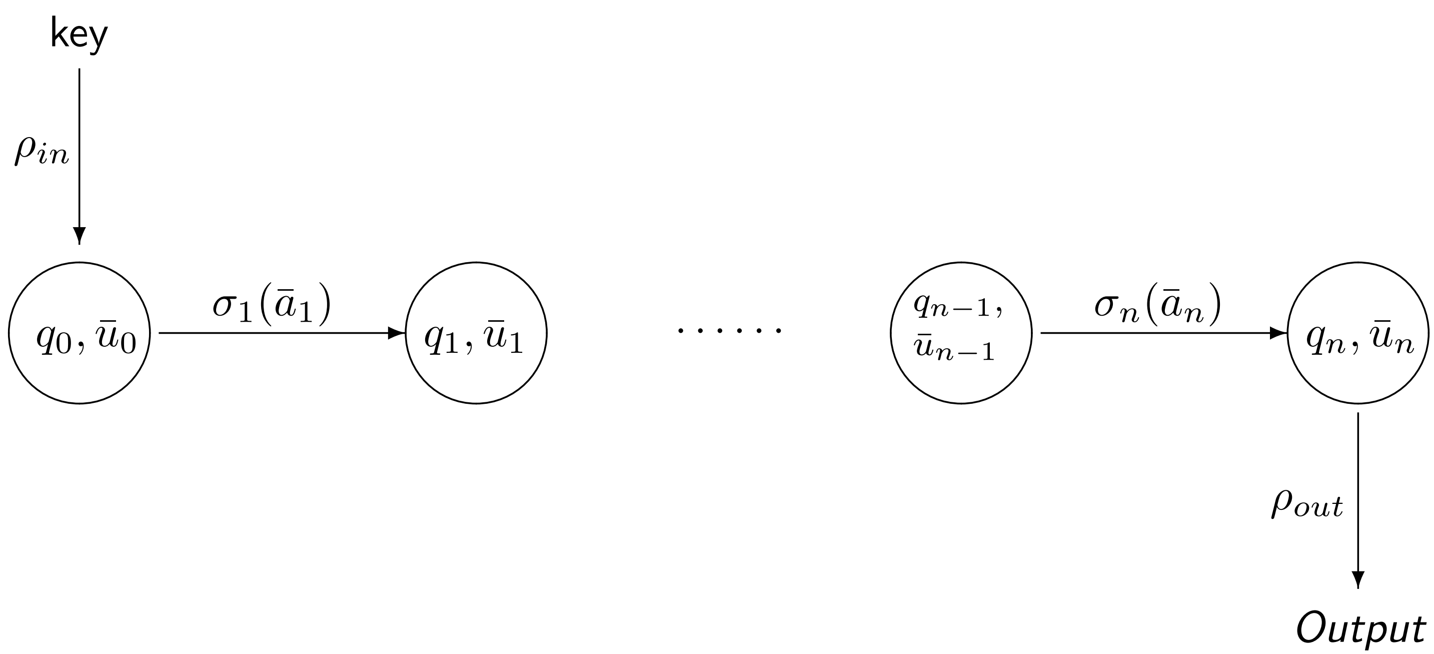
Finite state automaton with a fixed number of registers.



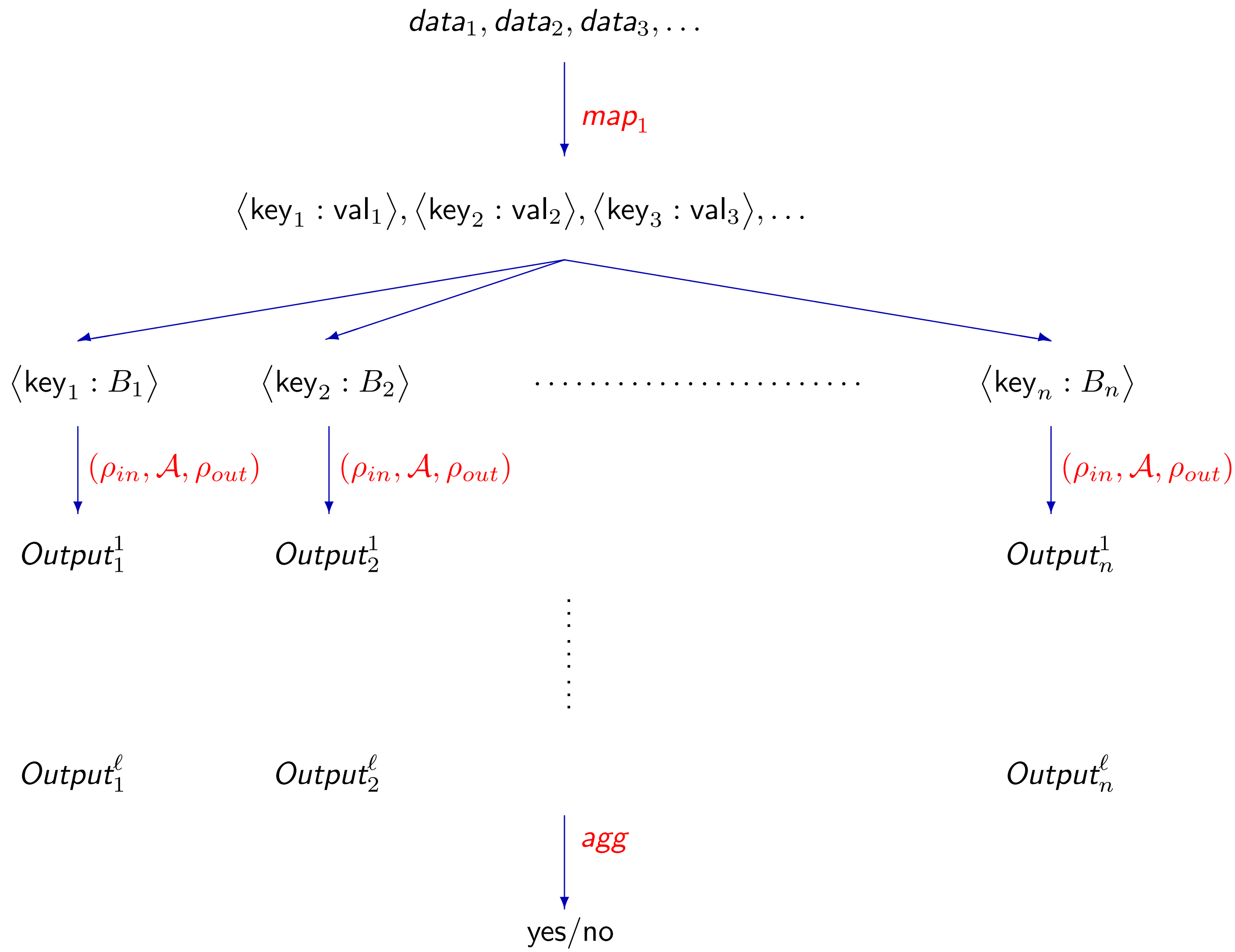
Originally studied by Francez, Kaminski and Shemesh.

DSA: The reducer $\mathit{red}_i = (\rho_{in}, \mathcal{A}, \rho_{out})$

On input $\langle$key : $B\rangle$, where $B = \{\sigma_1(\bar{a}_1), \ldots, \sigma_n(\bar{a}_n)\}$



Boolean DSA: $M = (map_1, red_1, \ldots, map_\ell, red_\ell, agg)$



DSA & Relational Algebra

Theorem. *Non-Boolean DSA can perform intersection, union, set difference, projection and selection.*

Theorem. *For every FO sentence (rel. algebra) φ and an integer m, there is DSA $M_{\varphi,m}$ such that for every database* DB *of degree $\leq m$, $M_{\varphi,m}$ accepts* DB *if and only if* DB $\models \varphi$.

- It still holds for FO with modulo counting ($\exists^{i \bmod m} x$).
- The proof is via Hanf locality.

DSA cannot detect the existence of walk

Definition. ℓ-WALK $= \{G \mid G$ contains a walk of length $\geq \ell\}$.

Theorem. *There is no DSA that accepts 5-WALK.*

Theorem. *On bounded degree graphs, there is ℓ-round DSA that accepts 2^{ℓ}-WALK.*

Theorem. *There is no ℓ-round DSA that accepts $2^{\ell+1}$-WALK on graphs with degree at most 2.*

The big picture for DSA

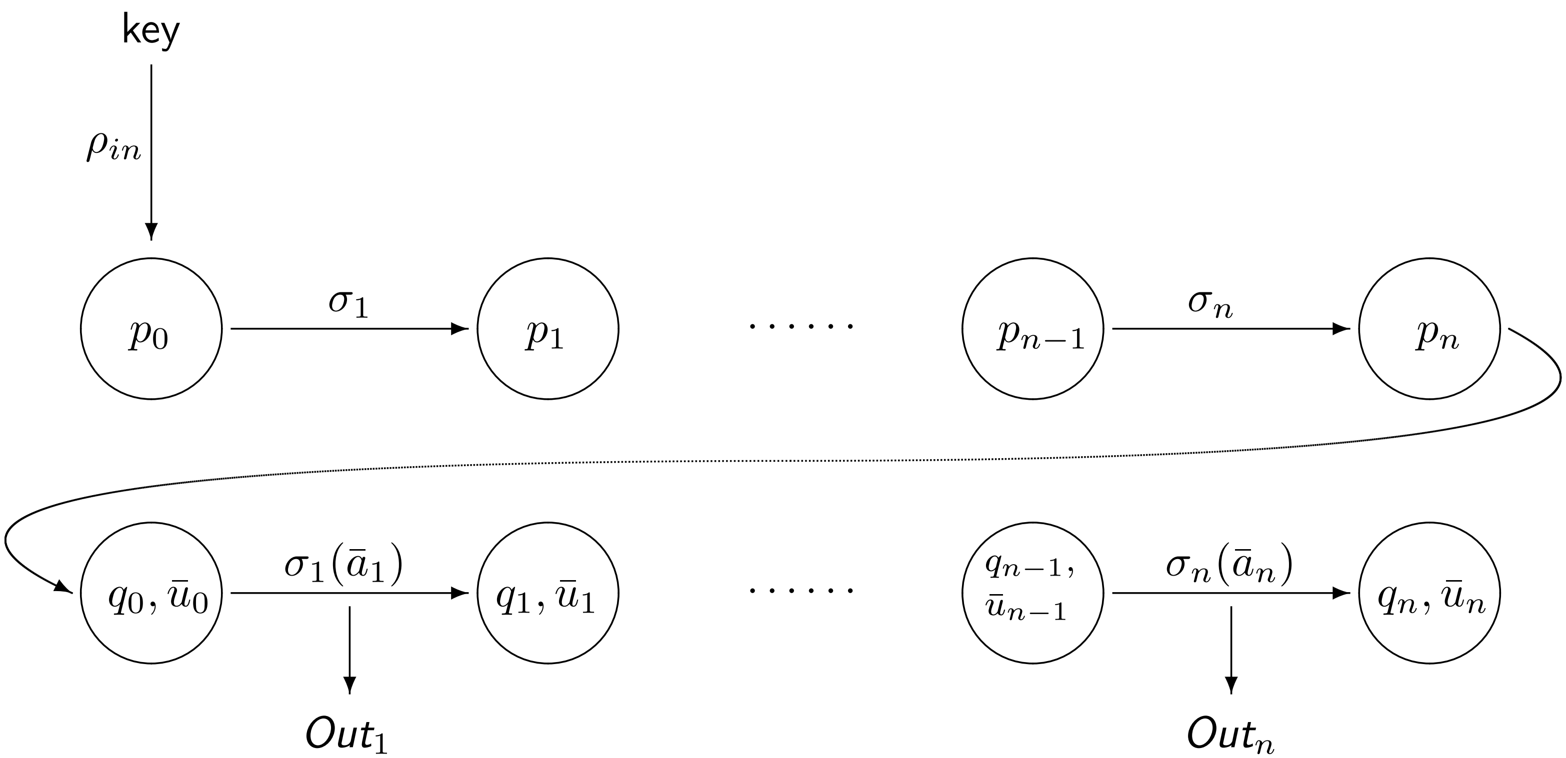
DSA	DST	DSTJ
$\cup\!\mid$		
Rel. alg. without join	Semi-join algebra	Relational algebra
(FO over bounded deg. DB)		

DST: $M = (\mathit{map}_1, \mathit{red}_1, \mathit{map}_2, \mathit{red}_2, \ldots, \mathit{map}_\ell, \mathit{red}_\ell, \mathit{agg})$

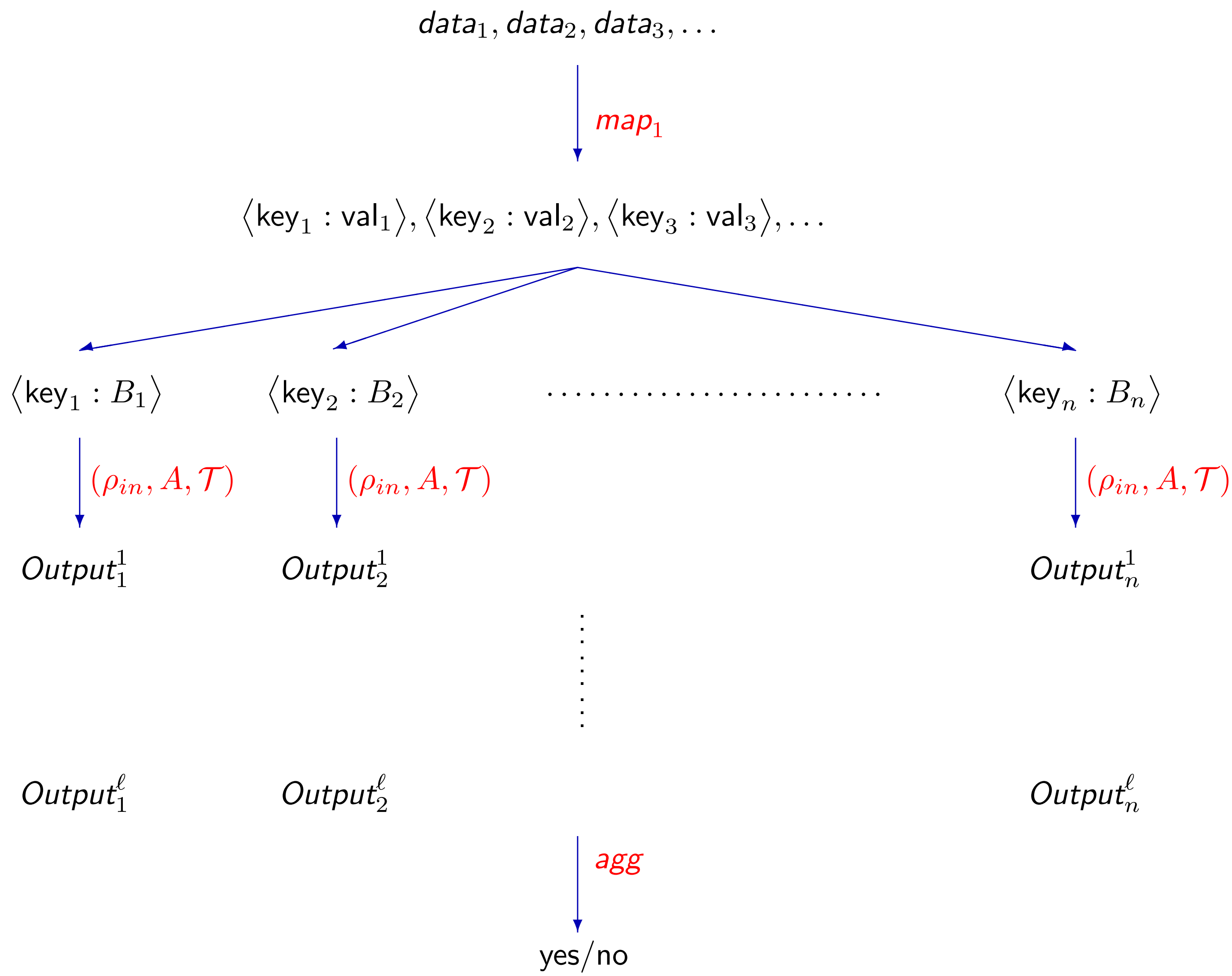
- Each map_i is a generic mapper.
- Each red_i is
  finite state automaton and deterministic "register transducer."
- agg is deterministic register automaton.

DST: The reducer $red_i = (\rho_{in}, A, \mathcal{T})$

On input $\langle \text{key} : B \rangle$, where $B = \{\sigma_1(\bar{a}_1), \ldots, \sigma_n(\bar{a}_n)\}$



Boolean DST: $M = (map_1, red_1, \ldots, map_\ell, red_\ell, agg)$



DST & semi-join algebra

Theorem. *Non-Boolean DST can perform intersection, union, set difference, projection, selection and semi-join, i.e. semi-join algebra.*

DST and walks on graphs

Theorem.

- *For every $\ell \geq 0$, there is an ℓ-round DST that accepts (2ℓ)-$WALK$.*
- *For every $\ell \geq 0$, there is no ℓ-round DST that accepts $(2\ell + 2)$-$WALK$.*
- *$(\ell + 1)$-round DSTs are strictly stronger than ℓ-round DSTs.*

Corollary. *DSTs are strictly more expressive than DSAs.*

Recall that there is no DSA that accepts 5-WALK.

DST & TRIANGLE

Definition. TRIANGLE $= \{G \mid G$ *contains a triangle*$\}$.

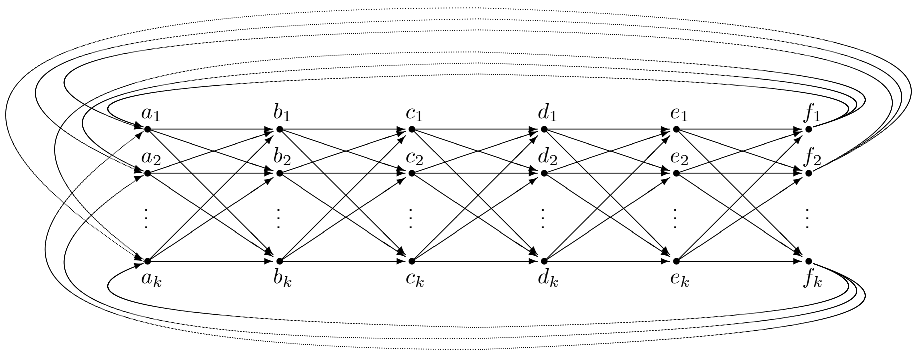
Theorem. *There is no DST that can accept* TRIANGLE*.*

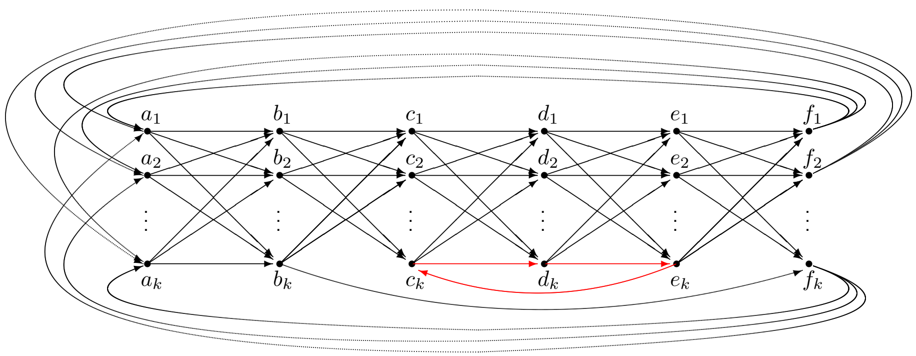
Reason:

- The output of DST is linear in the size of the input database.
- The number of triangles in can be $m^{3/2}$, where m is the number of edges.

DST cannot accept TRIANGLE

For every DST M, there is k such that M cannot differentiate:





Big picture for DSA, DST

DSA	$\subsetneq$	DST	DSTJ
$\cup\!\mid$		$\cup\!\mid$	
Rel. alg. without join		Semi-join algebra	Relational algebra
(FO over bounded deg. DB)			

DSTJ: $M = (\mathit{map}_1, \mathit{red}_1, \ldots, \mathit{map}_\ell, \mathit{red}_\ell, \mathit{agg})$

- Each map_i is a generic mapper.
- Each red_i is either:
  - $(\rho_{in}, A, \mathcal{T})$, as in DST, or
  - a joiner to perform Cartesian product.
- *agg* is deterministic register automaton.

DST: The reducer red_i

On input $\langle \text{key} : B \rangle$

$(\rho_{in}, A, \mathcal{T})$

key $= (\sigma, \bar{a})$

Joiner (α, β, γ)

$\{\gamma(\bar{u}, \bar{v}) \mid \alpha(\bar{u}) \in B$ and $\beta(\bar{v}) \in B\}$

DSTJ & relational algebra

Theorem. *Non-Boolean DSTJ can perform intersection, union, set difference, projection, selection, semi-join and equi-join.*

Theorem. *There is a 2-round DSTJ that accepts* TRIANGLE*.*

Corollary. *DSTJs are strictly stronger than DSTs.*

DSTJ and cycles in graphs

Definition. ℓ-CYCLE $= \{G \mid G$ *contains a cycle of length* $\ell\}$.

Theorem.

- *For every* $\ell \geq 0$*, there is an* ℓ*-round DSTJ that accepts* (2^{ℓ})-CYCLE.
- *For every* $\ell \geq 0$*, there is no* ℓ*-round DSTJ that accepts* $(2^{\ell+1})$-CYCLE.
- $(\ell + 1)$*-round DSTJs are strictly stronger than* ℓ*-round DSTJs.*

Big picture for DSA, DST, DSTJ

DSA	$\subsetneq$	DST	$\subsetneq$	DSTJ
$\cup\!\mid$		$\cup\!\mid$		$\cup\!\mid$
Rel. alg. without join		Semi-join algebra		Relational algebra
(FO over bounded deg. DB)				

Applicability

We built a software that implements the “algorithmic idea” of DSTJ.

- His name is GUMBO.
- He is built on top of Hadoop.
- He performs parallel multiple semijoins. In fact, he can perform guarded fragment queries.
- His performance is comparable with Pig and Hive.
- He has additional novel strategies that make him more effective in evaluating multi-semijoins.
- He can be found here:
`https://zenodo.org/record/51517#.V7hxt5N96fU`

References

- Gumbo: Guarded Fragment Queries over Big Data.

  EDBT 2015 (demo).

  Jonny Daenen, Frank Neven, Tony Tan.

- Distributed Streaming with Finite Memory.

  ICDT 2015.

  Frank Neven, Nicole Schweikardt, Frédéric Servais, Tony Tan.

- Parallel Evaluation of Multi-Semi-Joins.

  Jonny Daenen, Frank Neven, Tony Tan, Stijn Vansummeren.

  PVLDB 9(10): 732-743 (2016)