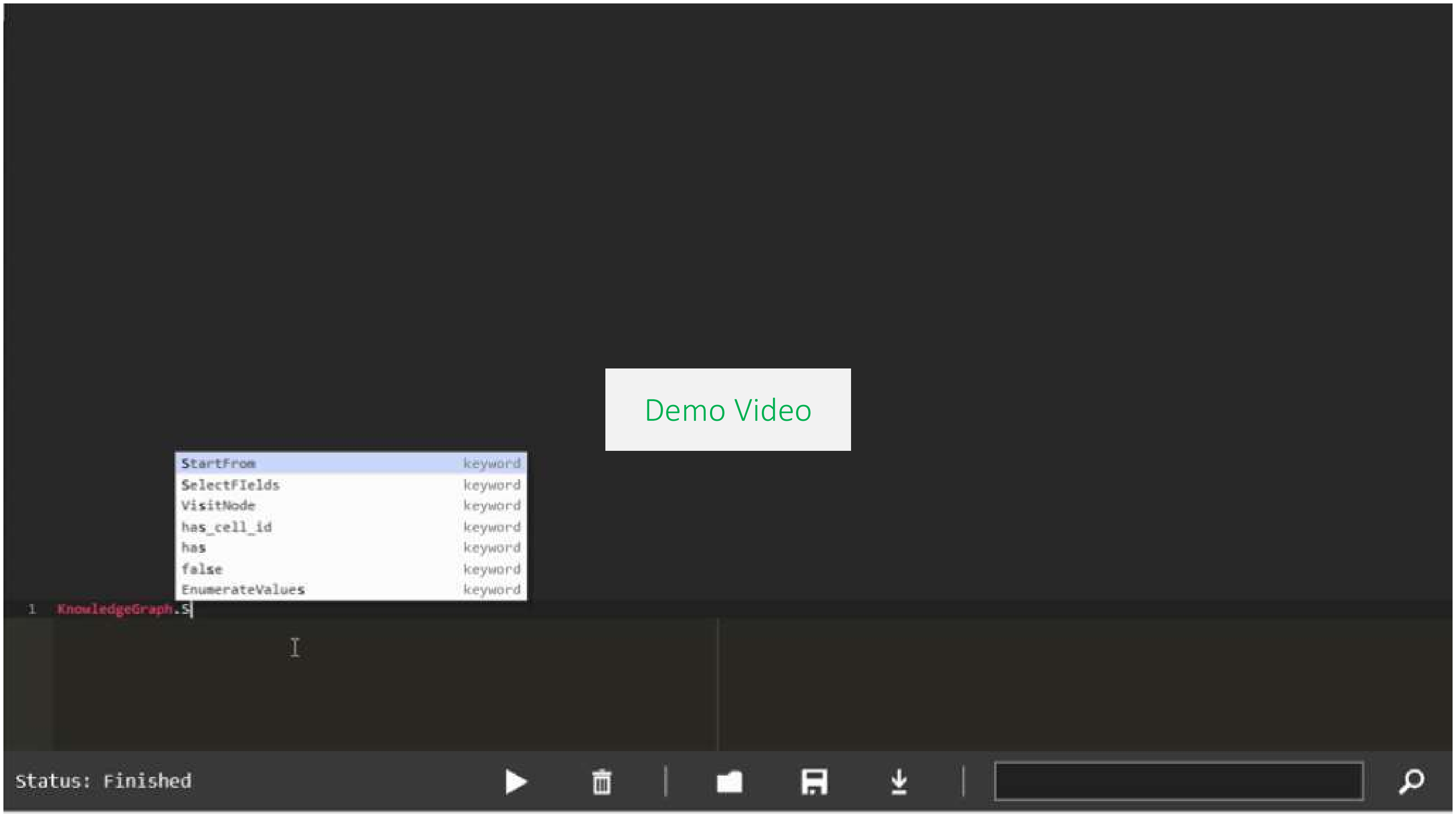
Real-time Knowledge Graph Serving

Bin Shao

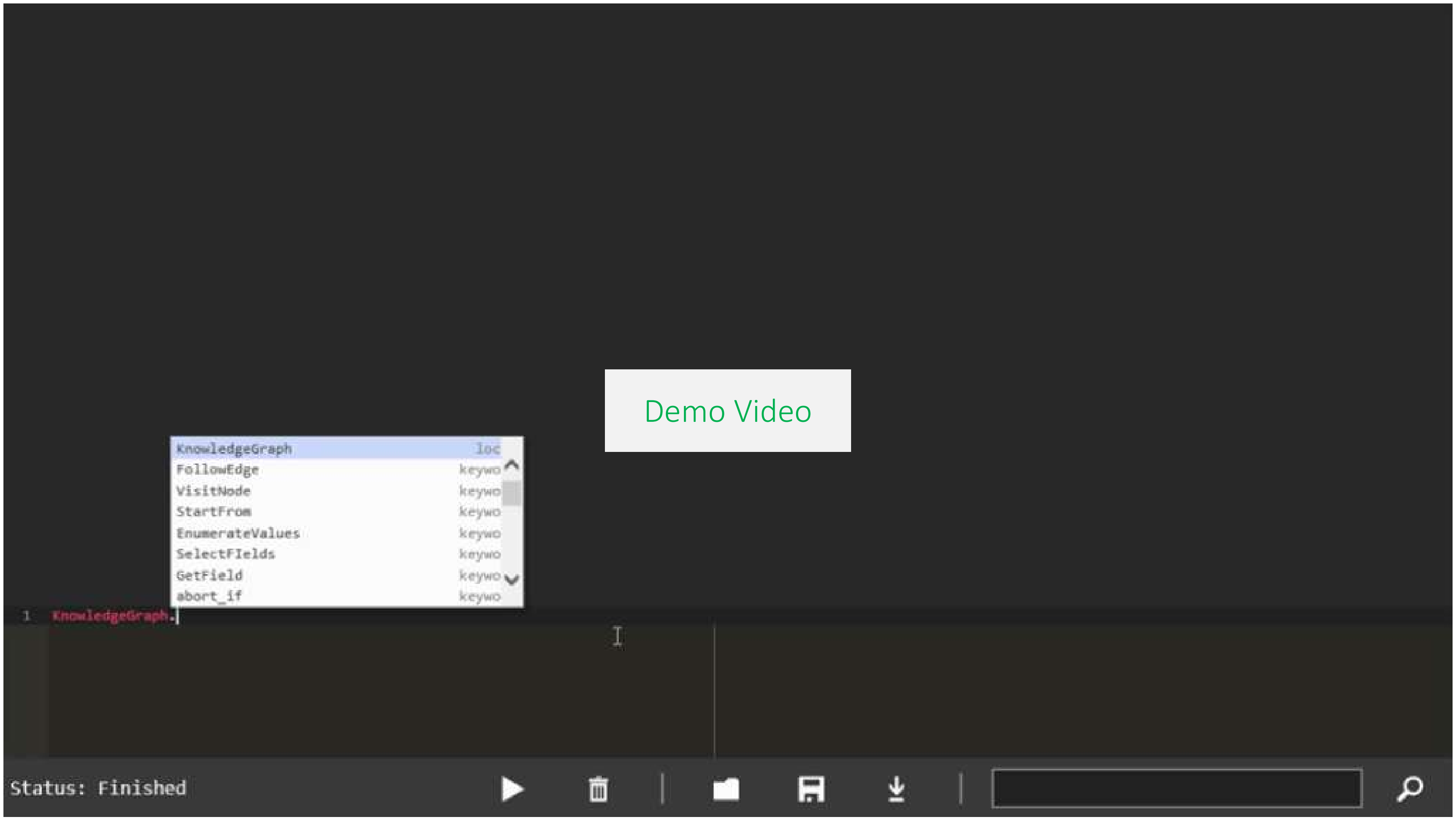
Microsoft Research Asia (Beijing, China)

This talk is about knowledge graph serving from a pragmatic point of view …

Appetizer



Find the people that have the same profession with Bill Gates, and speak at least 3 languages.



Find the triangles containing the vertex ‘Beijing’ with a sampling rate of 4%.

Outline

- Knowledge graph serving scenarios
- General design principles of knowledge graph serving systems
- Representative graph systems
- Real-time query processing
- Knowledge serving application: symbolic reasoning

Knowledge Serving Scenarios

A real-life relation search scenario

A News Headline

Tom Cruise Admits **Katie Holmes** Divorced Him To Protect **Suri** From Scientology

1. **Tom Cruise** – people.person.marriage – (marriage) – time.event.person – **Katie Holmes**
2. **Tom Cruise** – people.person.children – (**Suri Cruise**) – people.person.parent – **Katie Holmes**
3. **Tom Cruise** – film.actor.film – (Bambi Verleihung 2007) – film.filmactor – **Katie Holmes**
4. . . .

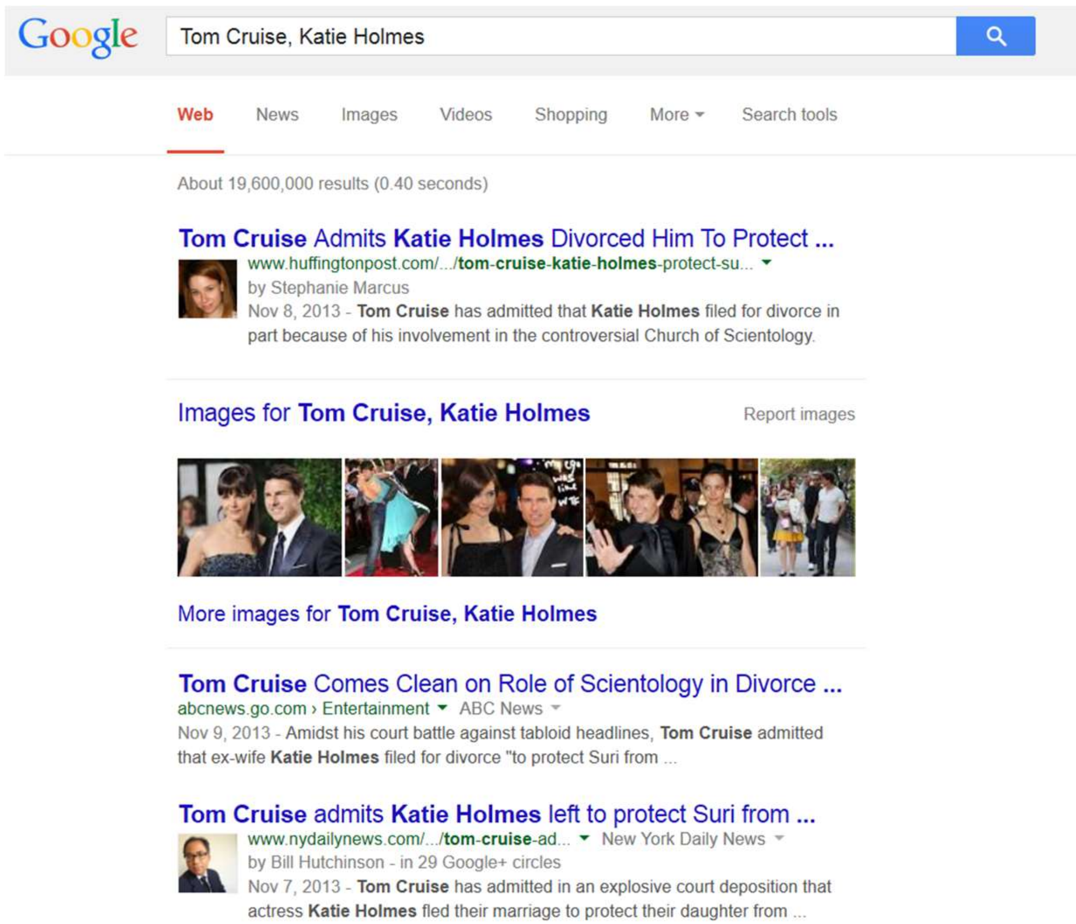
Relation search in knowledge graph

Entity A $\cdots \rightsquigarrow$ **Entity B**

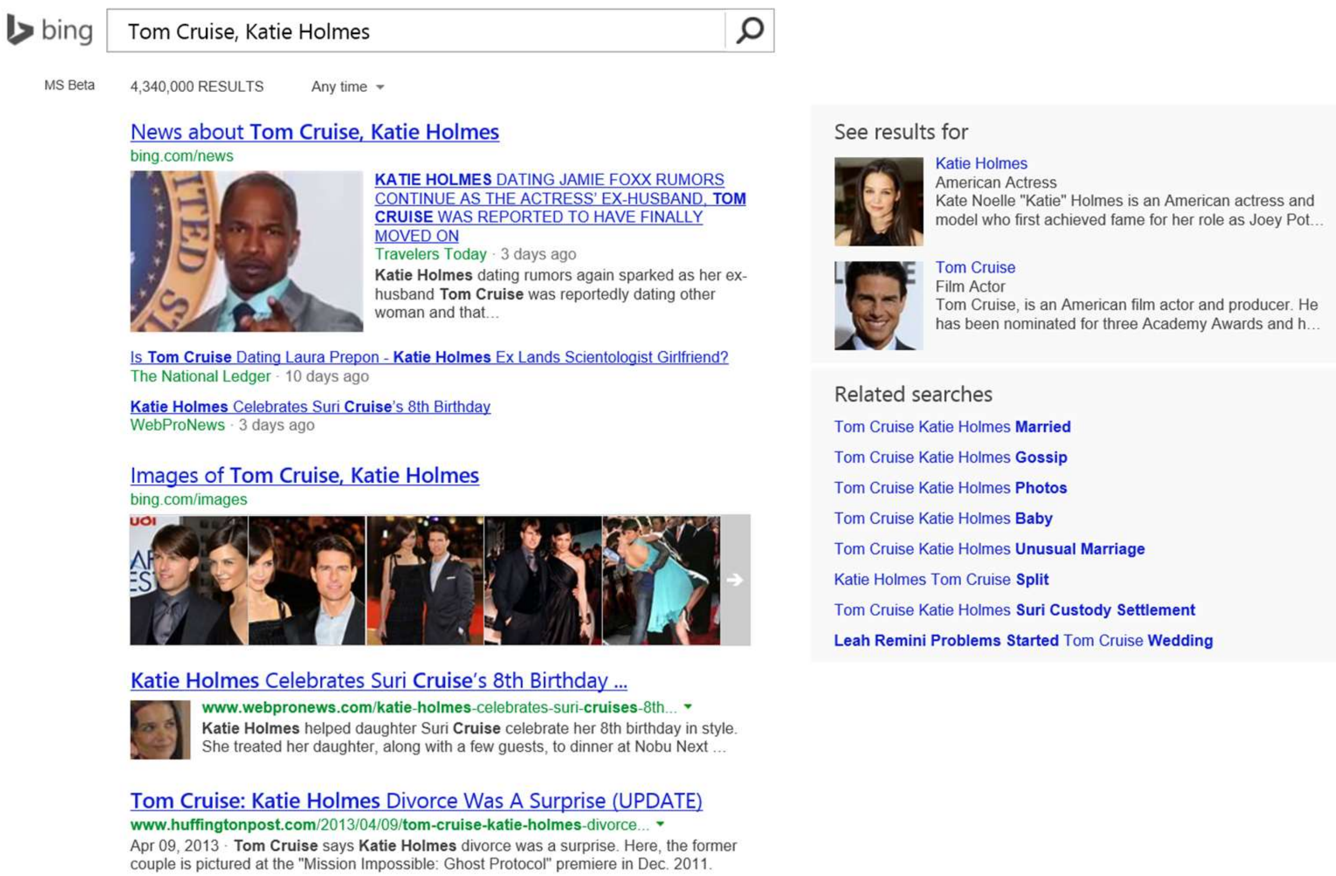
Multi-hop Relation Search

- Discover the **hidden relations** between entities
- Enable more than what entity indexes can support

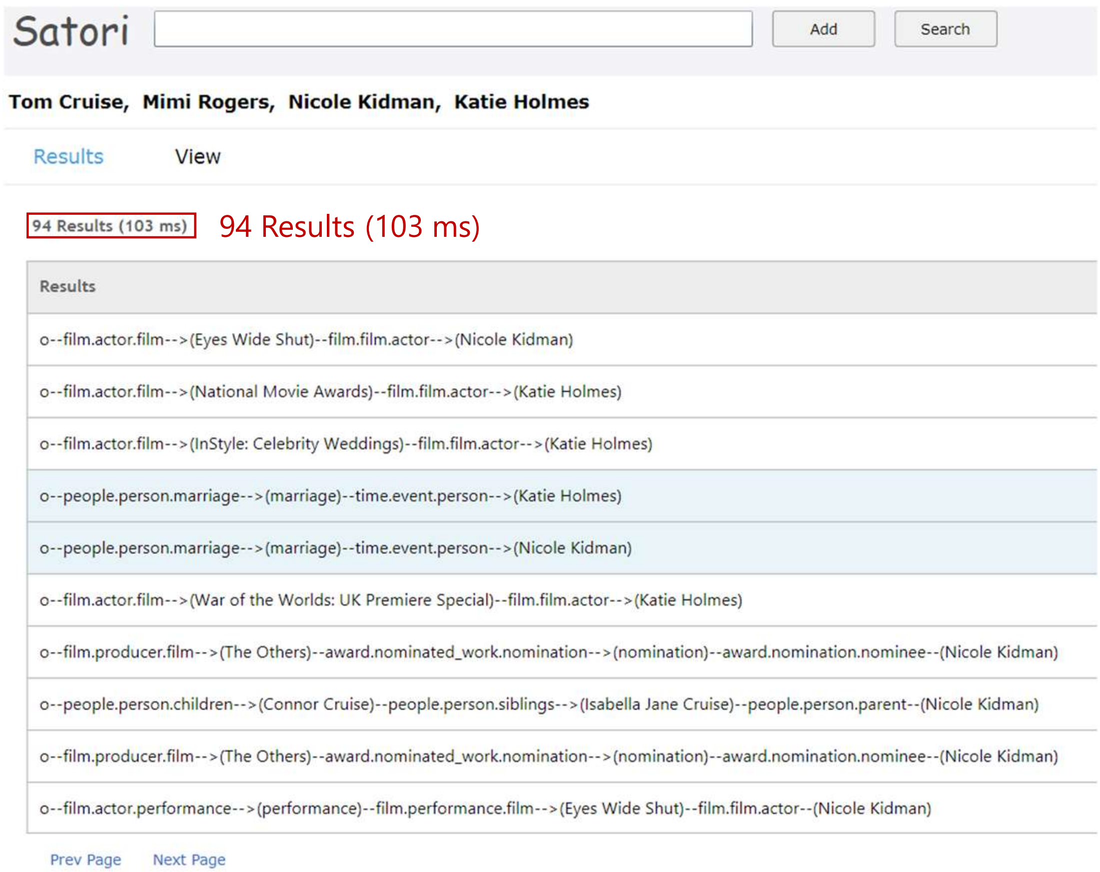
Search results of Google



Search results of Bing

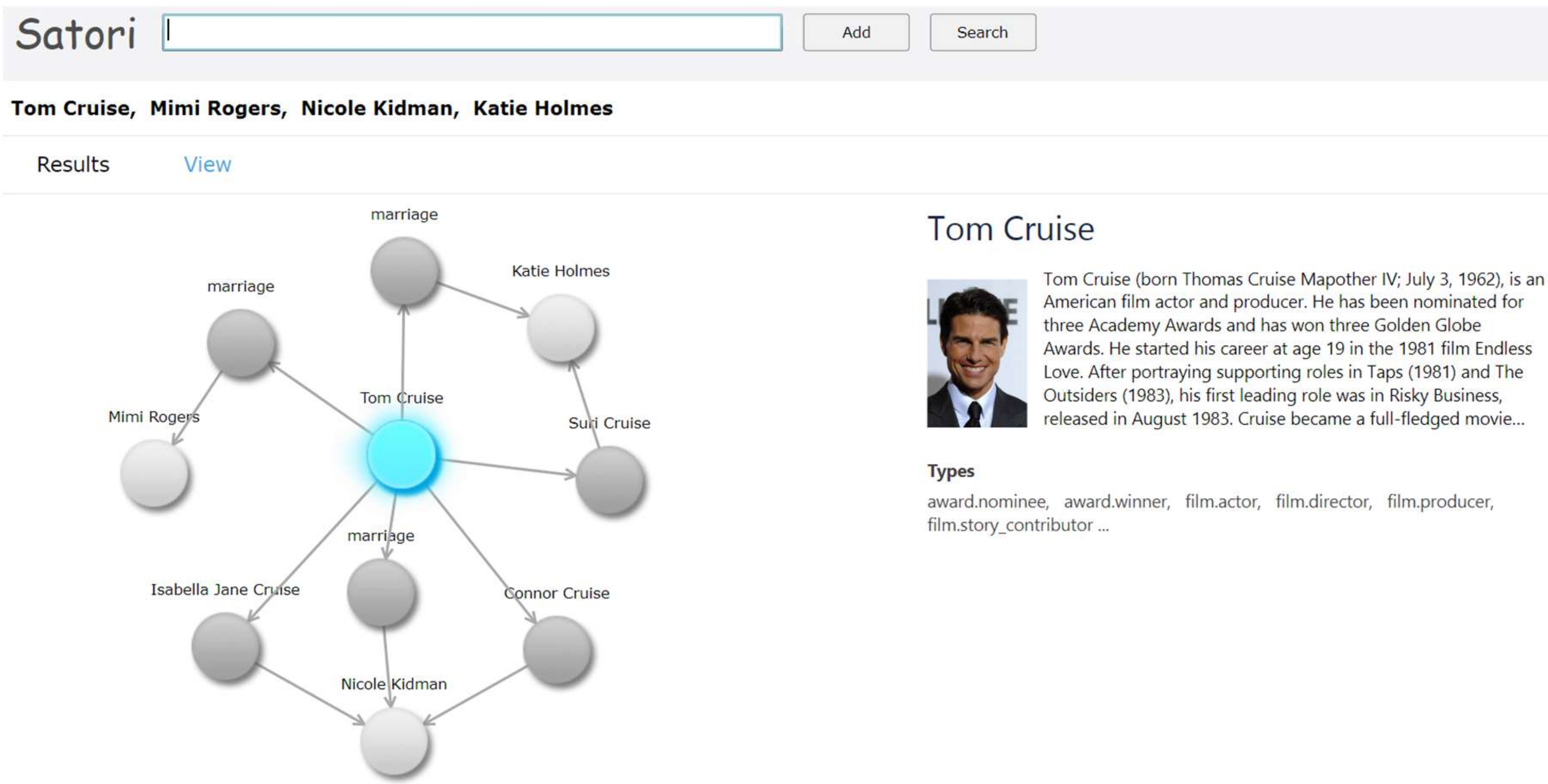


Relation search in knowledge graph



94 Results (103 ms)

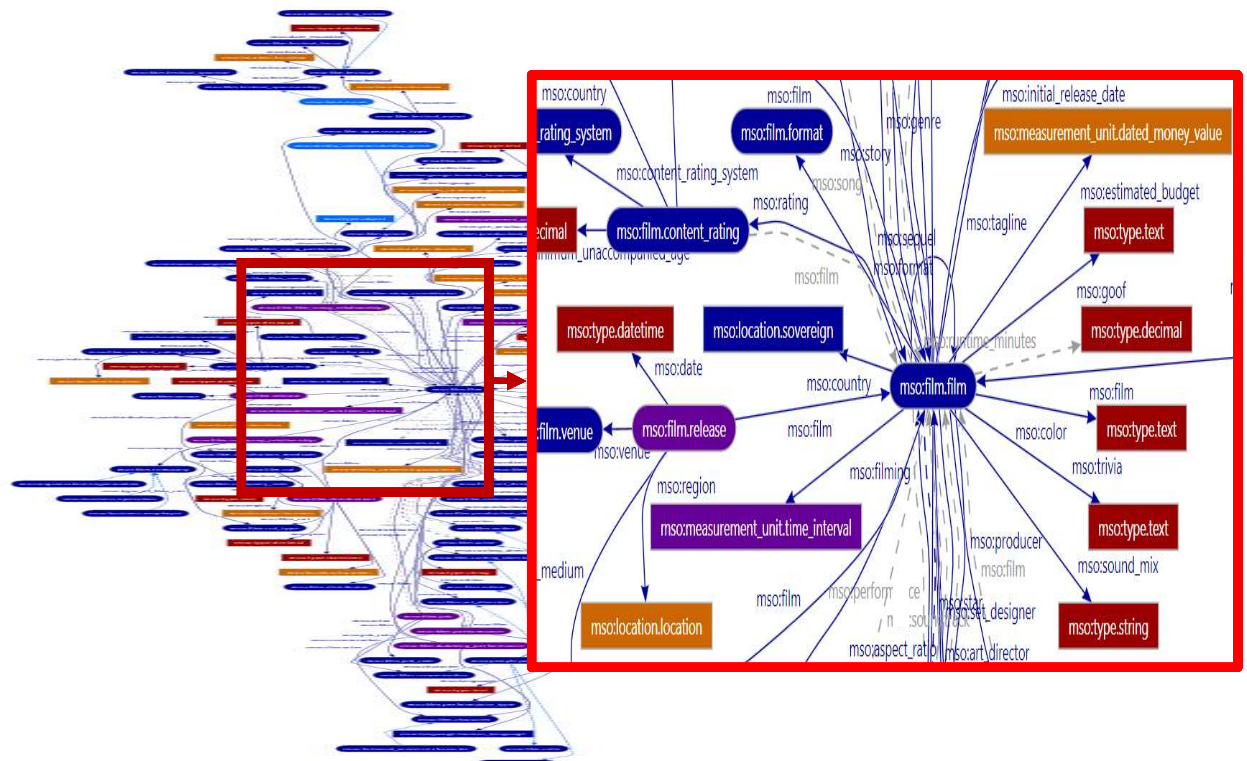
Relation search in knowledge graph



General Design Principles

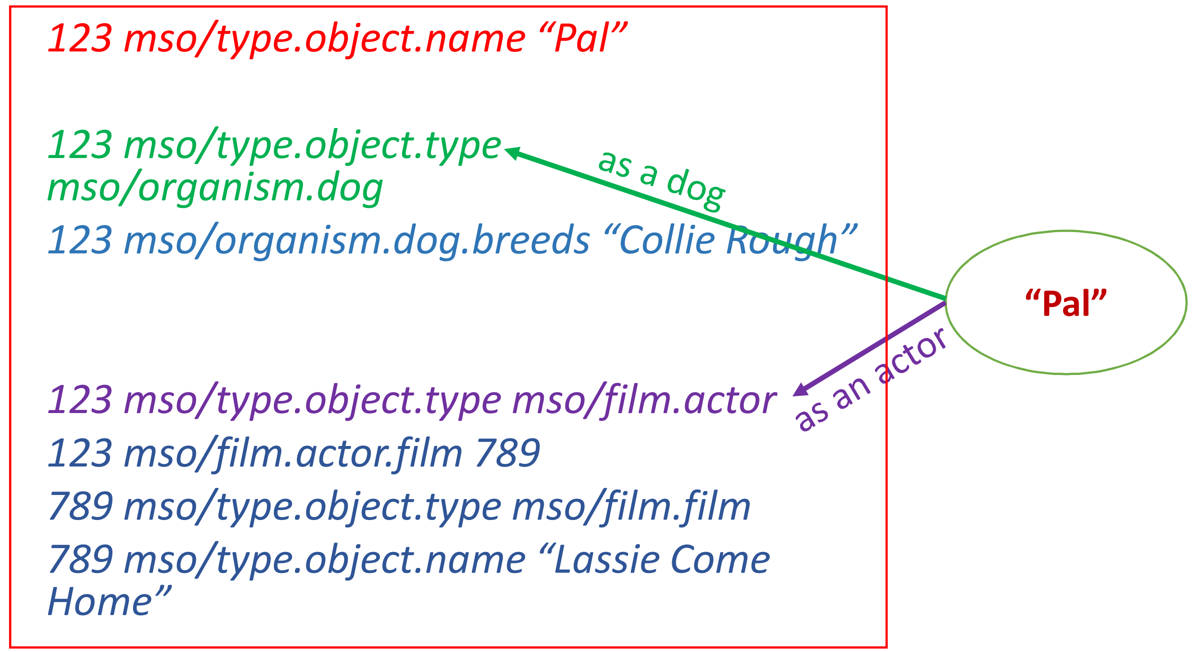
Challenges of serving knowledge graphs

- Data size
  - in the scale of terabytes
- Complex data schema
  - Rich relations

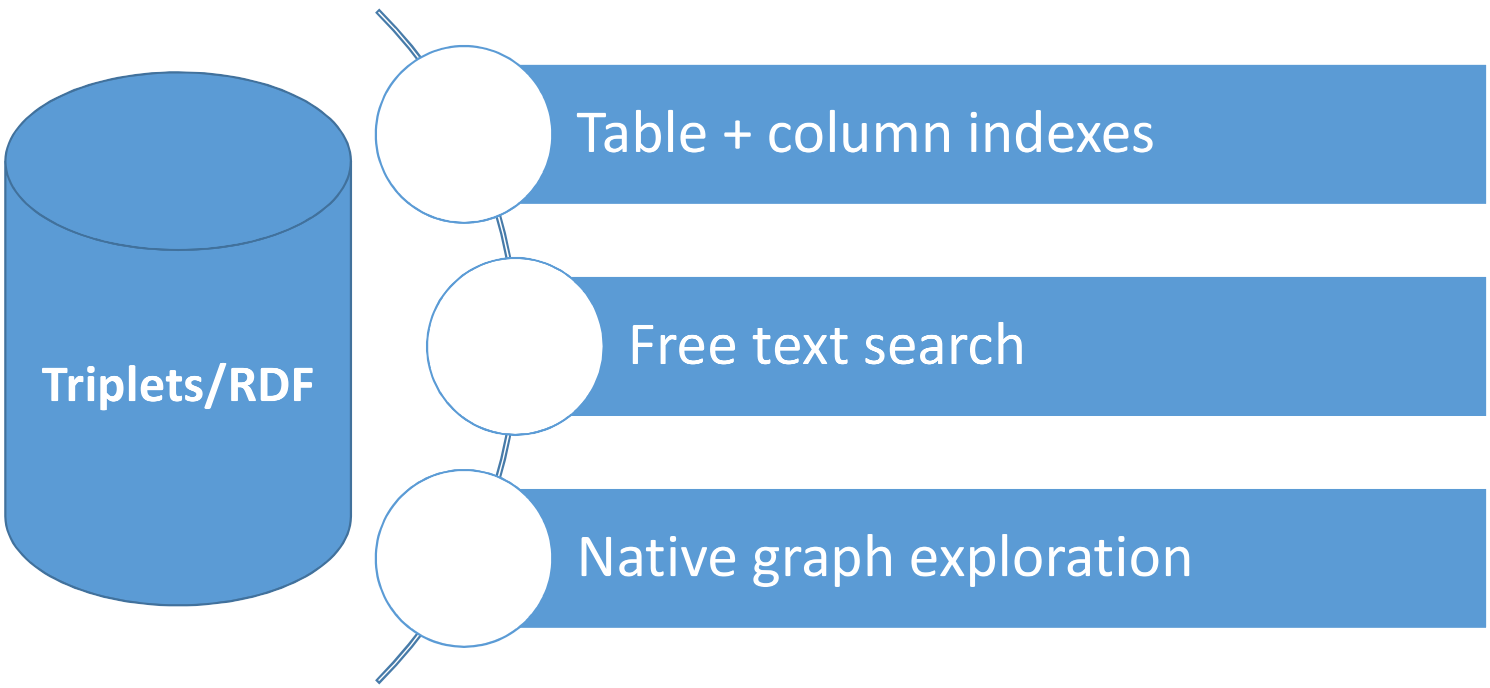


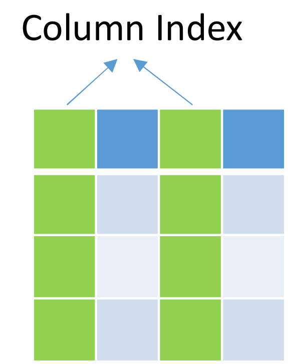
Challenges of serving knowledge graphs

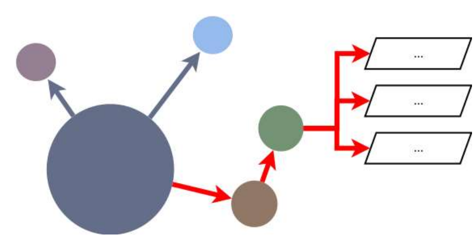
- Data size
  - In the scale of terabytes
- Complex data schema
  - Rich relations
  - Multi-typed entities



How to serve knowledge?



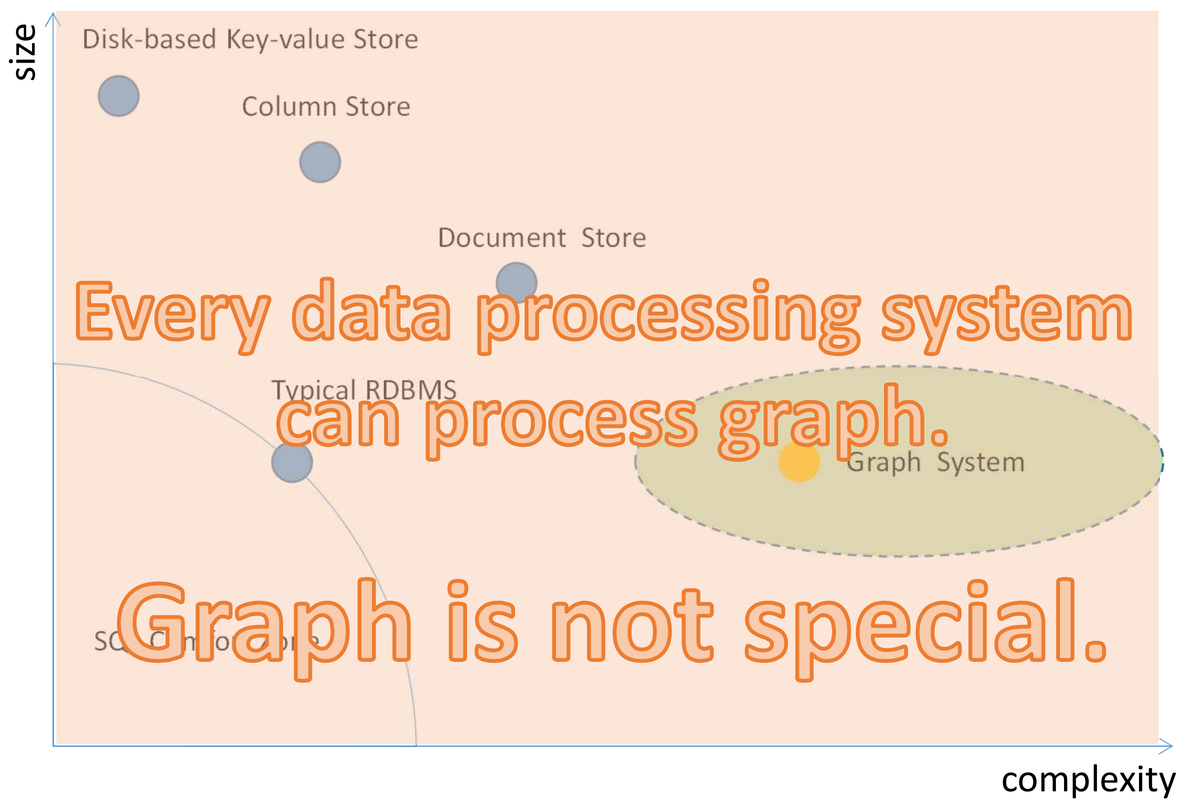




The needs ultimately determine the design

The first important rule: there is no one-size-fits-all system!

First rule: no one-size-fits-all system

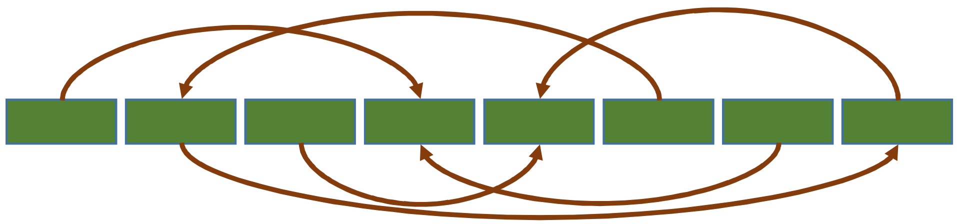


Every data processing system can process graph.

Graph is not special.

Characteristics of parallel graph processing

- Random access (poor locality)
  - For a node, its adjacent nodes cannot be accessed without "jumping" in the storage no matter how you represent a graph
  - Not cache-friendly, data reuse is hard



- It is hard to partition data
  - Difficult to extract parallelism by partitioning data
  - Hard to get an efficient "divide and conquer" solution

- Data driven
  - the structure of computations is not known in advance

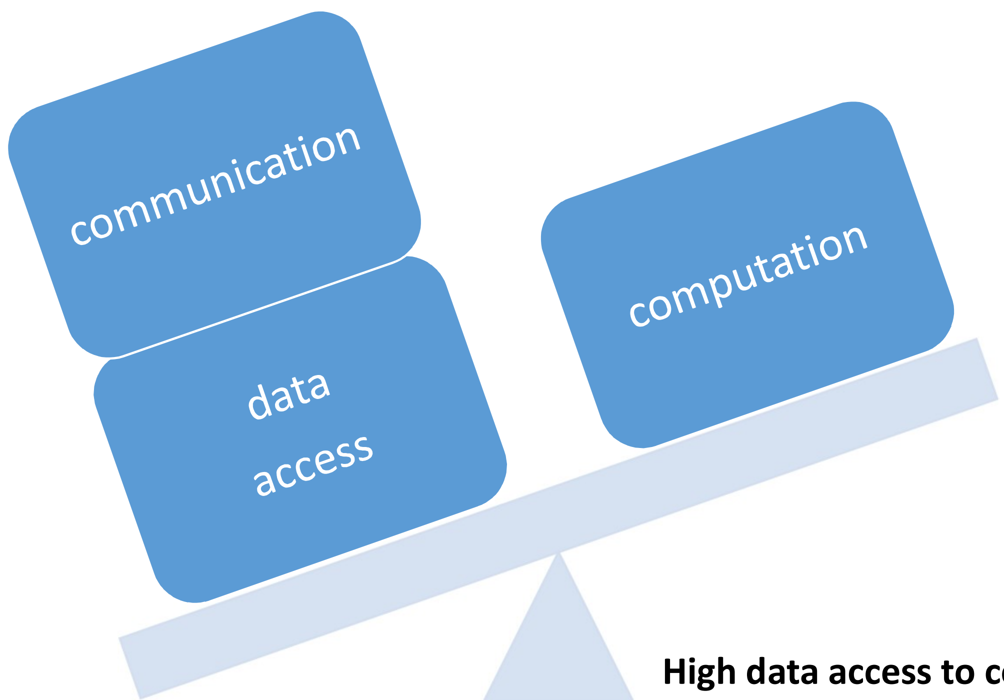
- High data access to computation ratio

Reference: Challenges in parallel graph processing

Online queries vs. offline analytics

- Online query processing is usually optimized for response time
- Offline analytics is usually optimized for throughput
- Compared to offline analytics, it is harder to optimize online queries
  - Online queries are sensitive to latency
  - It is difficult to predict the data access patterns of online queries

Query response time:
data access + communication + computation



System design choice

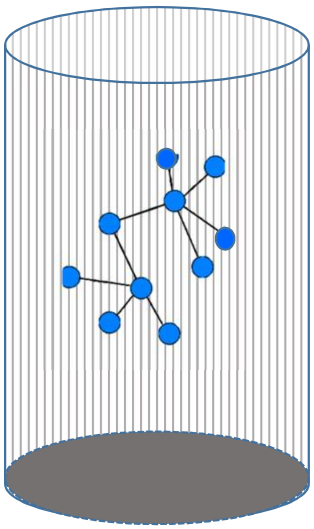
- Main storage (storage backend)
- Index
- Communication paradigm: two-sided vs. one-sided
- Scale out or scale up
- ACID transactions or not

System design choice

- **Main storage (storage backend)**
- Index
- Communication paradigm: two-sided vs. one-sided
- Scale out or scale up
- ACID transactions or not

Graph may be in the jail of storage

- Many existing data management systems can be used to process graphs
- Many of them are mature, but not for graphs
  - RDBMS, MapReduce
  - The commonest graph operation “traversal” incurs excessive amount of joins



Graph in the Jail of the storage

Traverse graph using joins in RDBMS

ID	name	….
1	N1	…
2	N2	…
3	N3	…
4	N4	…
5	N5	…
6	N6	…
…	…	…

Node Table: N

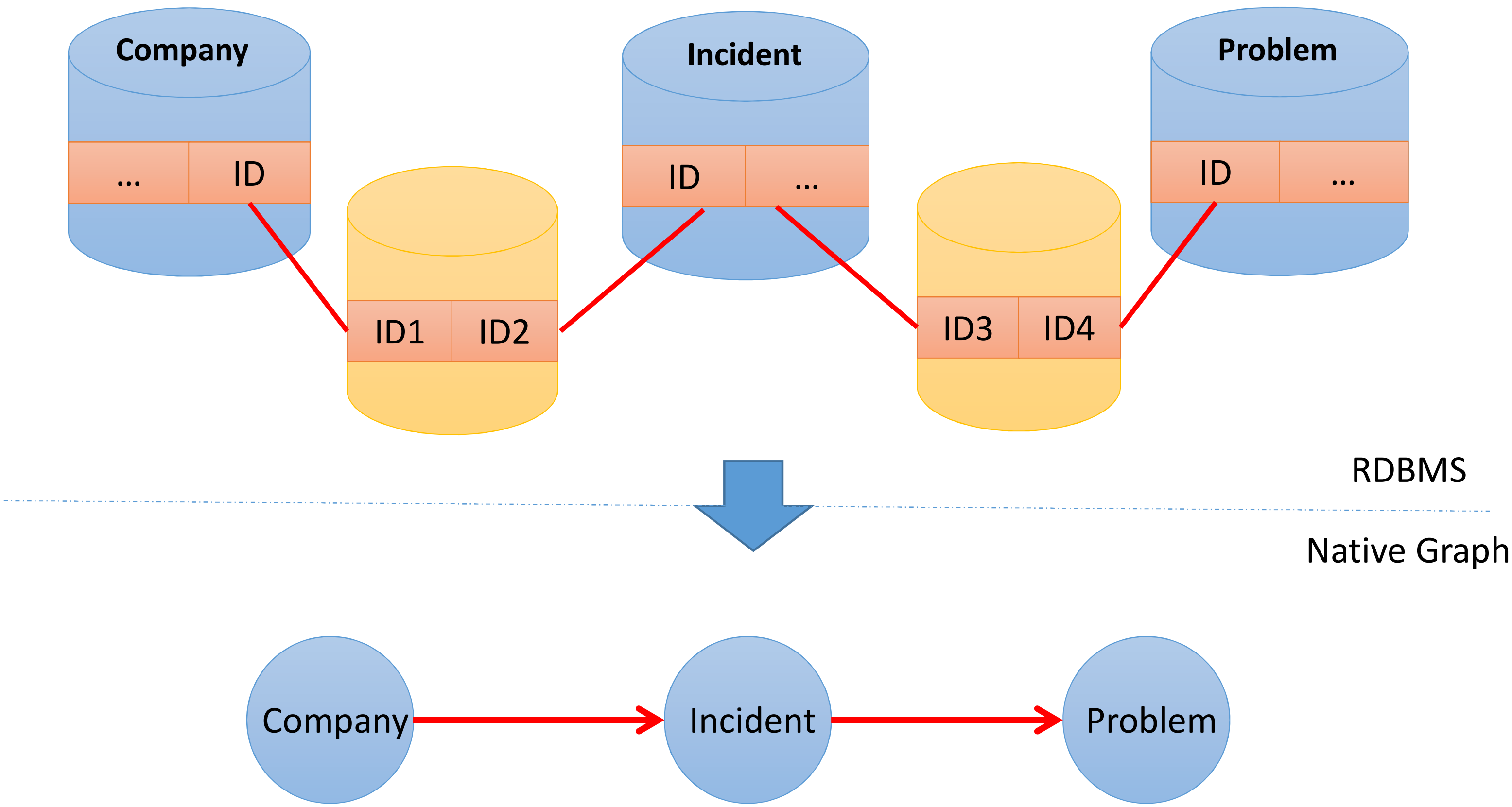
src	dst
1	3
2	4
2	1
4	3
1	5
1	6
…	…

Edge Table: E

Get neighbors of N1

```
SELECT *
FROM N
LEFT JOIN  E ON N.ID = E.dst
WHERE E.src = 1
```

Multi-way join vs. graph traversal



System design choice

- Main storage (storage backend)
- **Index**
- Communication paradigm: two-sided vs. one-sided
- Scale out or scale up
- ACID Transactions or not

Index

It is costly to index graph structures, use it wisely.

Index-based subgraph matching

Algorithms	Index Size	Index Time	Update Cost
Ullmann [Ullmann76], VF2 [CordellaFSV04]	-	-	-
RDF-3X [NeumannW10]	**O**(m)	**O**(m)	**O**(d)
BitMat [AtreCZH10]	**O**(m)	**O**(m)	**O**(m)
Subdue [HolderCD94]	-	**Exponential**	**O**(m)
SpiderMine [ZhuQLYHY11]	-	**Exponential**	**O**(m)
R-Join [ChengYDYW08]	**O**($nm^{1/2}$)	**O**(n^4)	**O**(n)
Distance-Join [ZouCO09]	**O**($nm^{1/2}$)	**O**(n^4)	**O**(n)
GraphQL [HeS08]	$O(m + nd^r)$	$O(m + nd^r)$	**O**(d^r)
Zhao [ZhaoH10]	**O**(nd^r)	**O**(nd^r)	**O**(d^L)
GADDI [ZhangLY09]	$O(nd^L)$	$O(nd^L)$	**O**(d^L)

Reference: Sun VLDB 2012

Index-based subgraph matching

Algorithms	Index Size for Facebook	Index Time for Facebook	Query Time on Facebook (s)
Ullmann [Ullmann76], VF2 [CordellaFSV04]	-	-	$>$1000
RDF-3X [NeumannW10]	1T	$>$20 days	$>$48
BitMat [AtreCZH10]	2.4T	$>$20 days	$>$269
Subdue [HolderCD94]	-	$>$ 67 years	-
SpiderMine [ZhuQLYHY11]	-	$>$ 3 years	-
R-Join [ChengYDYW08]	$>$175T	$> 10^{15}$ years	$>$200
Distance-Join [ZouCO09]	$>$175T	$> 10^{15}$ years	$>$4000
GraphQL [HeS08]	$>$13T(r=2)	$>$ 600 years	$>$2000
Zhao [ZhaoH10]	$>$12T(r=2)	$>$ 600 years	$>$600
GADDI [ZhangLY09]	$> 2 \times 10^5$T (L=4)	$> 4 \times 10^5$ years	$>$400

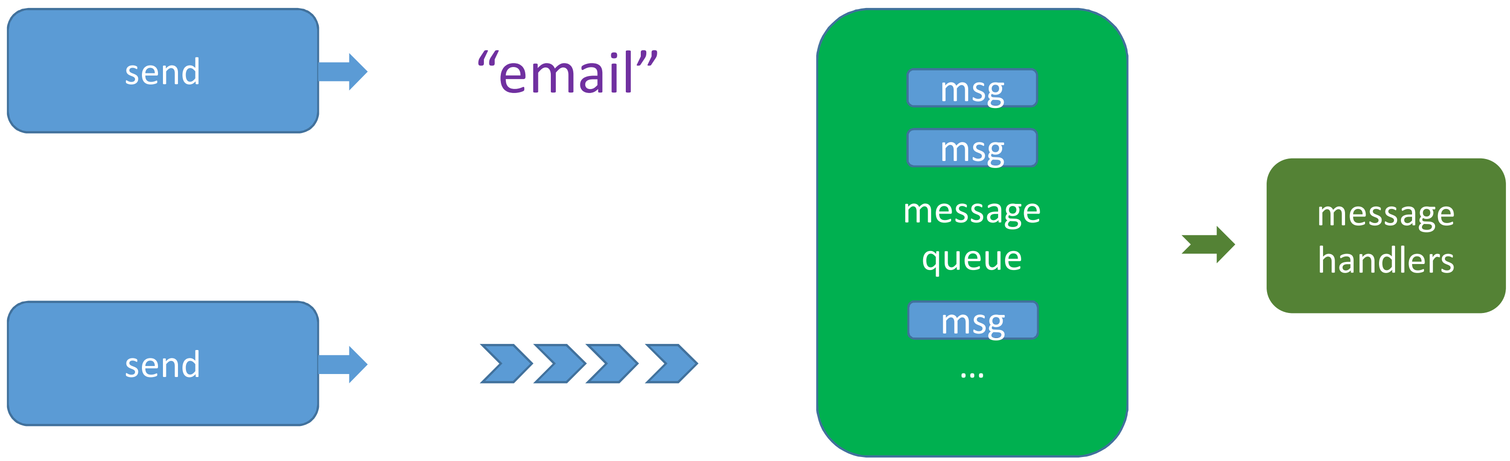
Reference: Sun VLDB 2012

System design choice

- Main storage (storage backend)
- Index
- **Communication paradigm: two-sided vs. one-sided**
- Scale out or scale up
- ACID transactions or not

Two-sided communication

One-sided communication



System design choice

- Main storage (storage backend)
- Index
- Communication paradigm: two-sided vs. one-sided
- **Scale out or scale up**
- ACID transactions or not

Design choice: scale-up vs. scale-out

- Supercomputer model
  - Programming model simple and efficient
    - shared memory address space
  - Expensive
  - Hardware is your ultimate limit

- Distributed cluster model
  - Programming model is complex
  - Relatively cheaper
  - Flexible to meet a variety of needs

Scale “OUT”, not “UP”

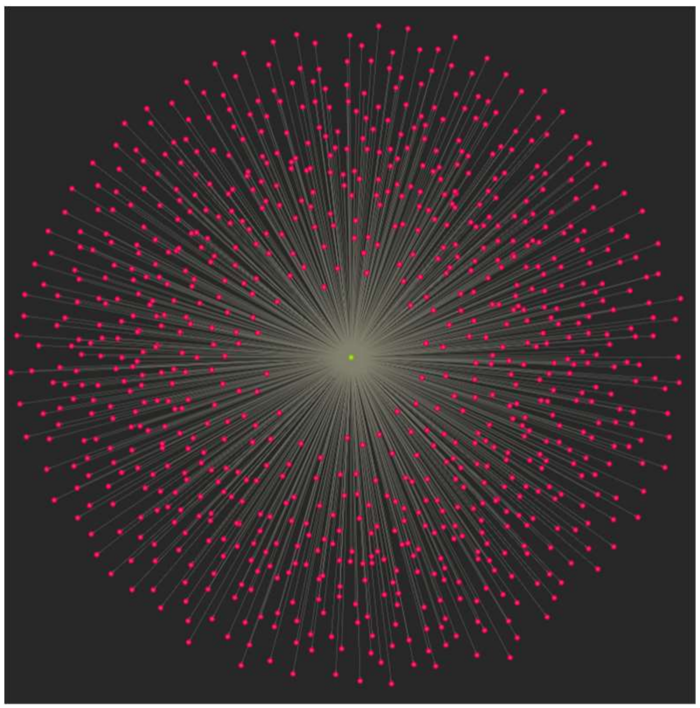
System design choice

- Main storage (storage backend)
- Index
- Communication paradigm: two-sided vs. one-sided
- Scale out or scale up
- **ACID transactions or not**

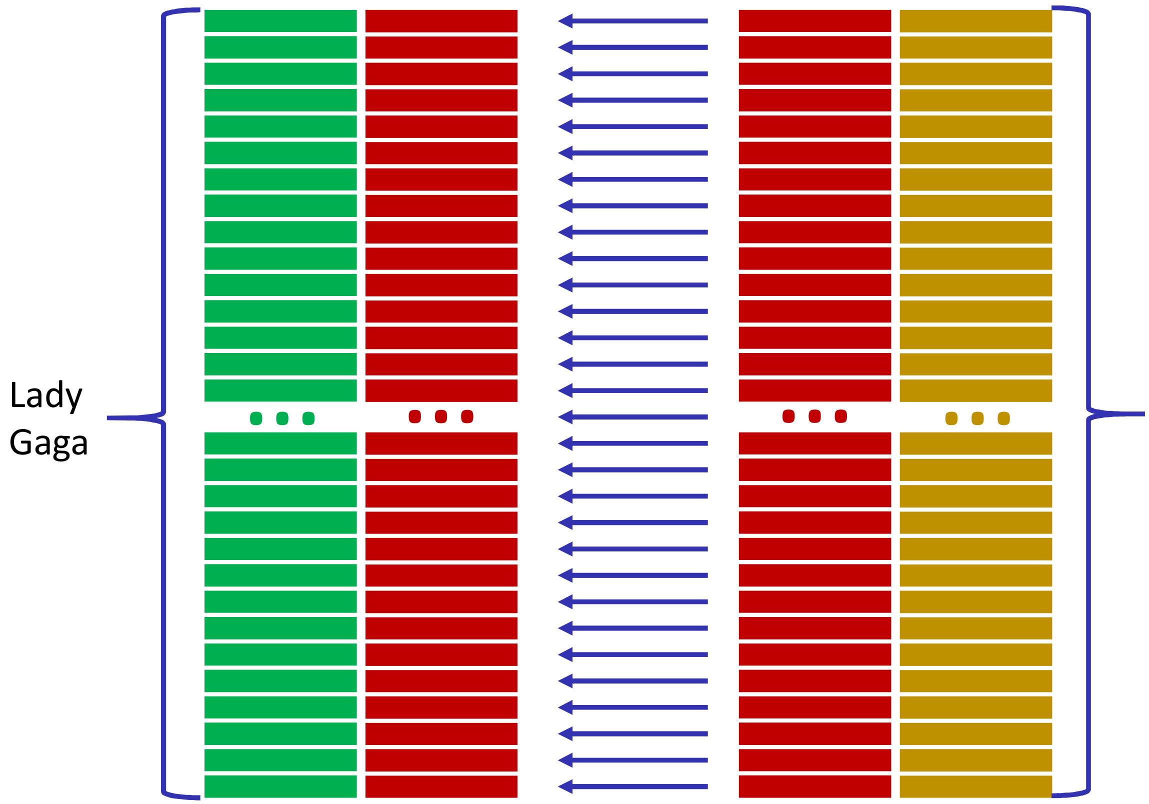
Think twice before diving into transactions

- Pros
  - Strong data consistency guarantee
- Cons
  - The hell of referential integrity
  - The disaster of cascading rollback
  - Multi-round network communications per commit for distributed transactions

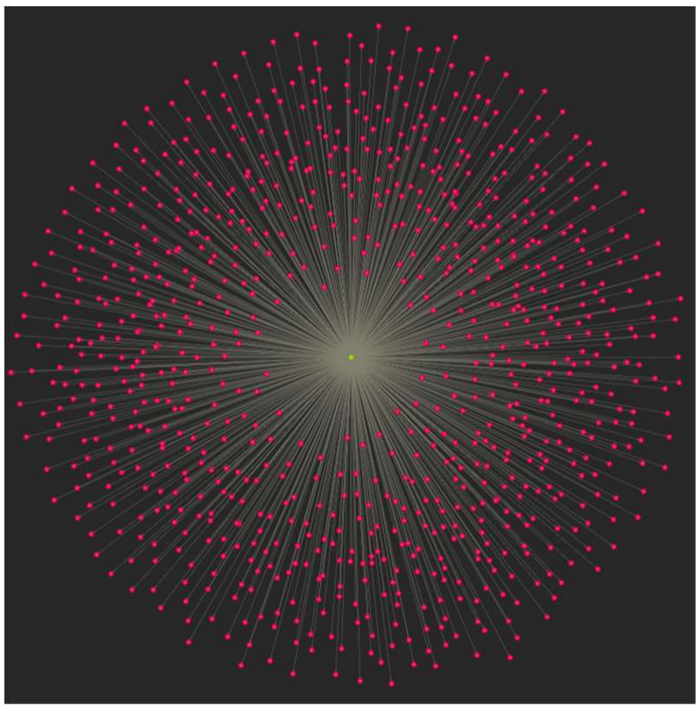
The hell of referential integrity

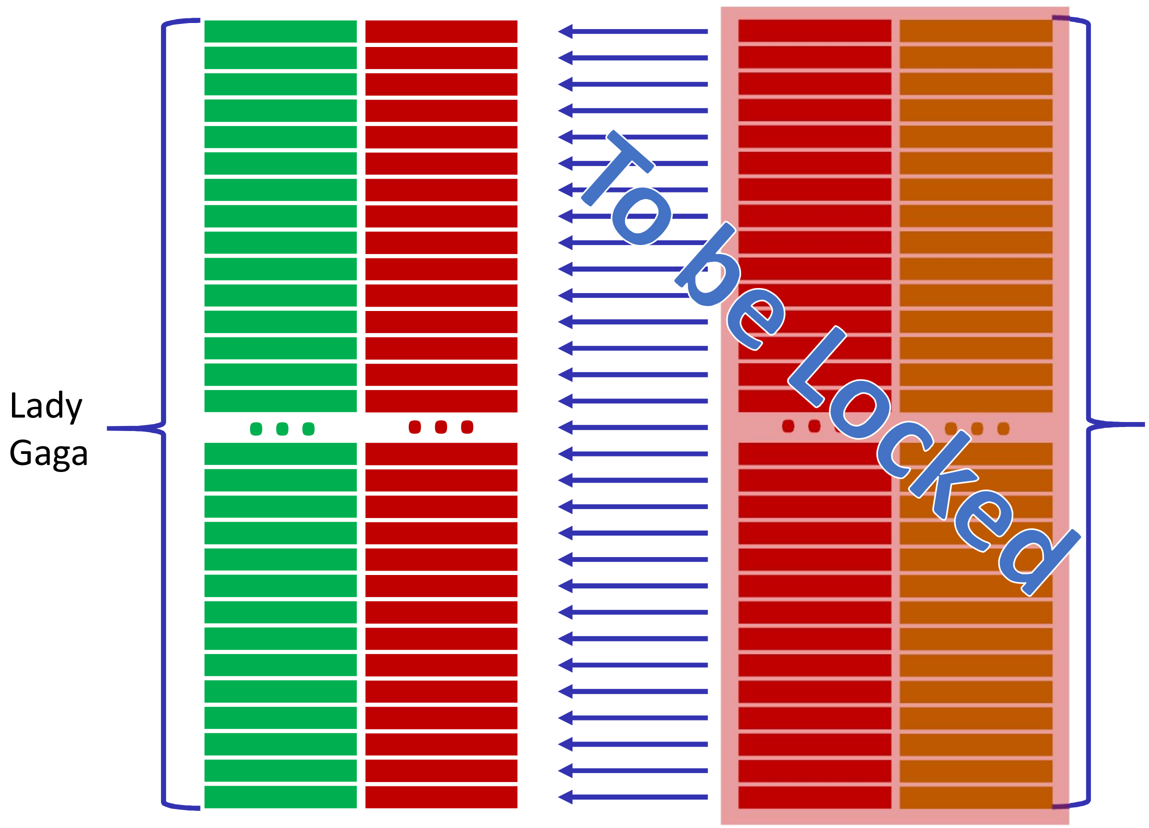


Lady Gaga in Freebase

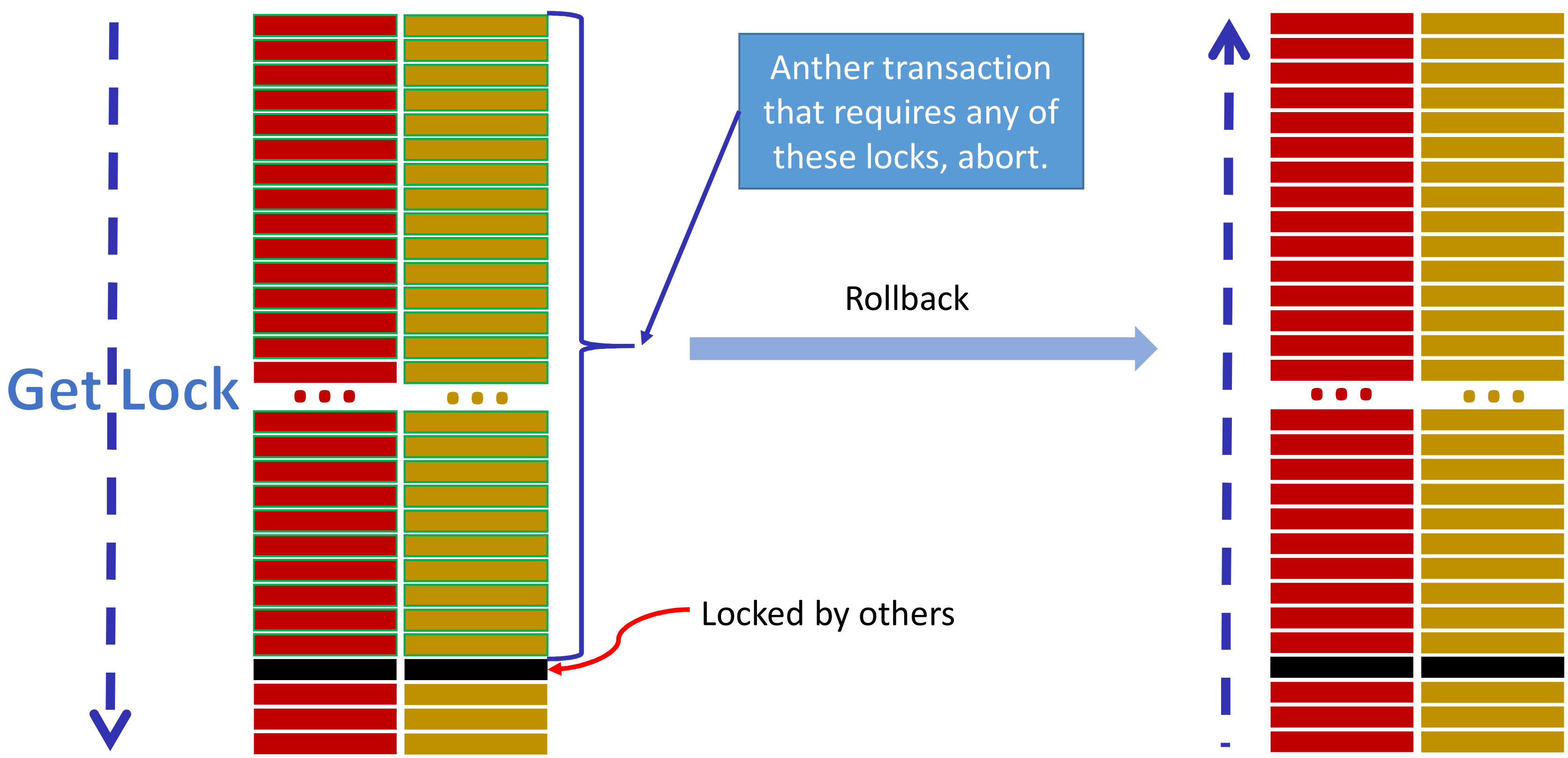


The hell of referential integrity





The disaster of cascading rollback



Representative Graph Systems

Existing systems

- Mature data processing systems
  - RDBMS
  - MapReduce systems
- Systems specialized for certain graph operations
  - PageRank, ……
- General-purpose graph processing systems
  - Neo4j, Trinity, Horton, HyperGraphDB, TinkerGraph, InfiniteGraph, Cayley, Titan, PEGASUS, Pregel, Giraph, GraphLab, GraphChi, GraphX …

Representative graph processing systems

	Property graphs	Online query	Data sharding	In-memory storage	Atomicity & Transaction
★ Neo4j	Yes	Yes	No	No	Yes
★ Trinity	Yes	Yes	Yes	Yes	Atomicity
★ Horton	Yes	Yes	Yes	Yes	No
★ HyperGraphDB	No	Yes	No	No	Yes
★ FlockDB	No	Yes	Yes	No	Yes
★ TinkerGraph	Yes	Yes	No	Yes	No
★ InfiniteGraph	Yes	Yes	Yes	No	Yes
★ Cayley	Yes	Yes	SB	SB	Yes
★ Titan	Yes	Yes	SB	SB	Yes
★ MapReduce	No	No	Yes	No	No
★ PEGASUS	No	No	Yes	No	No
★ Pregel	No	No	Yes	No	No
★ Giraph	No	No	Yes	No	No
★ GraphLab	No	No	Yes	No	No
★ GraphChi	No	No	No	No	No
★ GraphX	No	No	Yes	No	No

Representative graph processing paradigms

- MapReduce for graph processing
- Vertex-centric graph computation
- Matrix arithmetic
- Graph embedding

MapReduce for Graph Processing

MapReduce

- High latency, yet high throughput general purpose data processing platform
- Optimized for offline analytics on large data partitioned over hundreds of machines

Processing graph using MapReduce

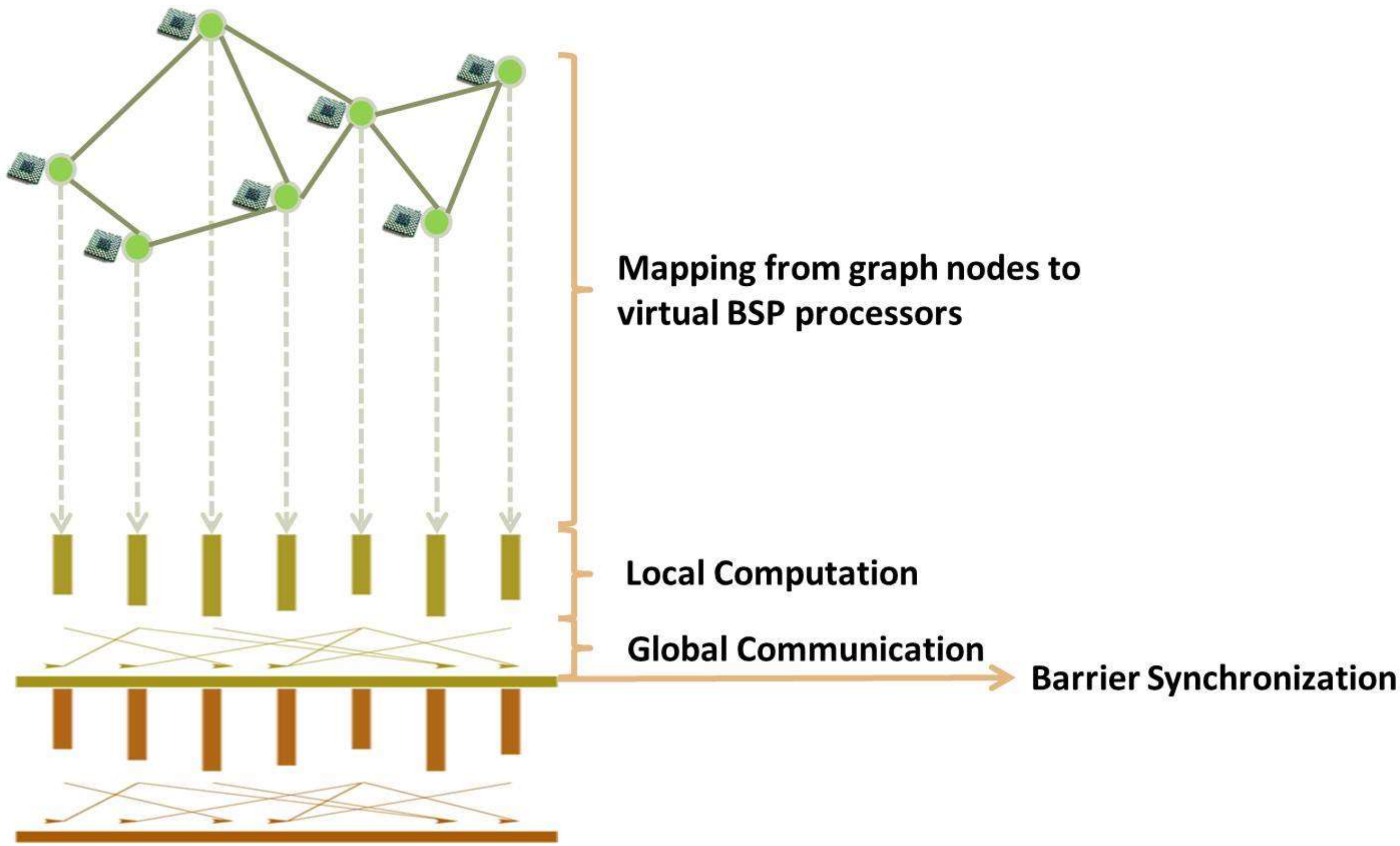
- No online query support
- The data model of MapReduce cannot represent graph natively
  - Graph algorithms cannot be expressed intuitively
- Inefficiency for graph processing
  - Intermediate results of each iteration need to be materialized
  - Entire graph structure need to be sent over network iteration after iteration, this incurs a large amount of unnecessary data movements

MapReduce

- De facto of distributed large data processing
- Great scalability: supports extremely large data, but unfortunately not for graphs

Vertex-centric graph computation

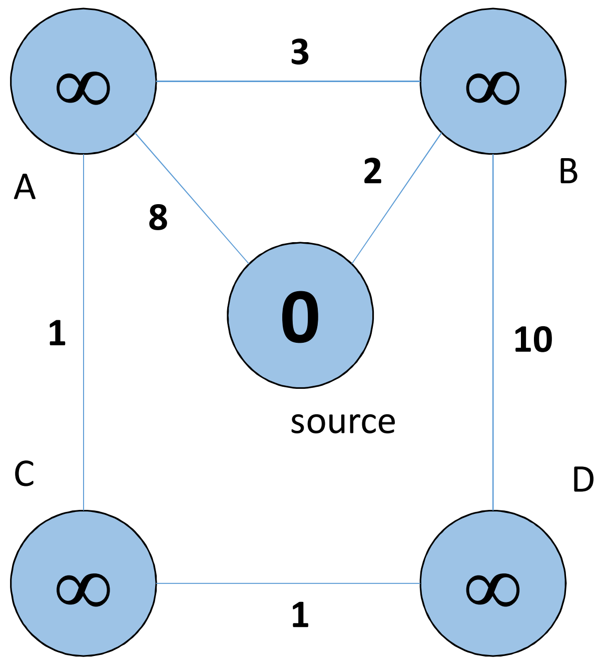
Basic idea: think like a vertex!



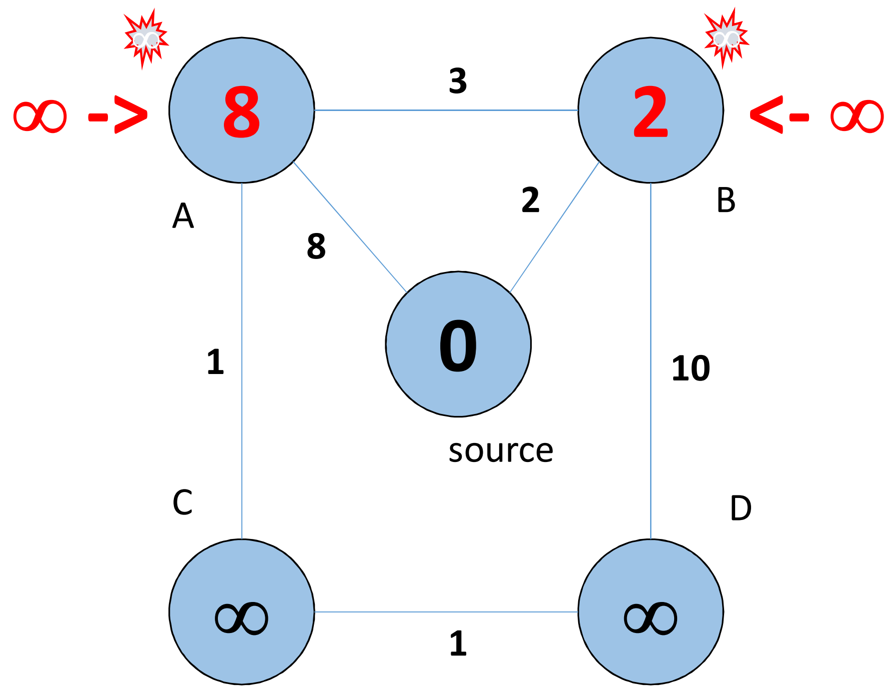
Computation model

- Graph computation is modeled as many supersteps
- Each vertex reads messages sent in the previous superstep
- Each vertex performs computations in parallel
- Each vertex can send messages to other vertices at the end of an iteration

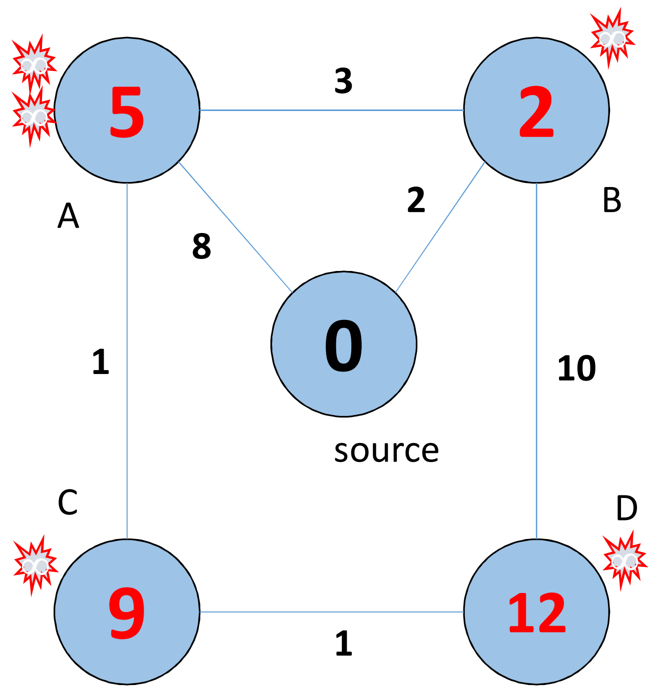
Example: SSSP



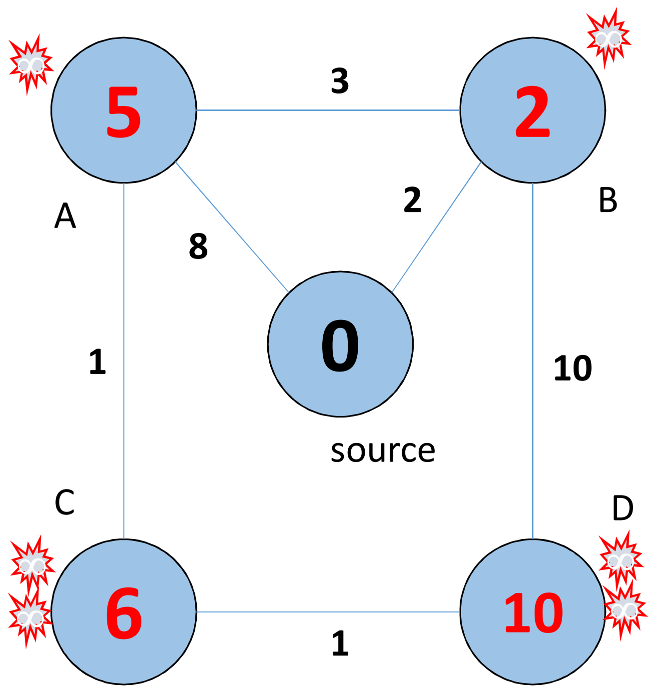
Example: SSSP



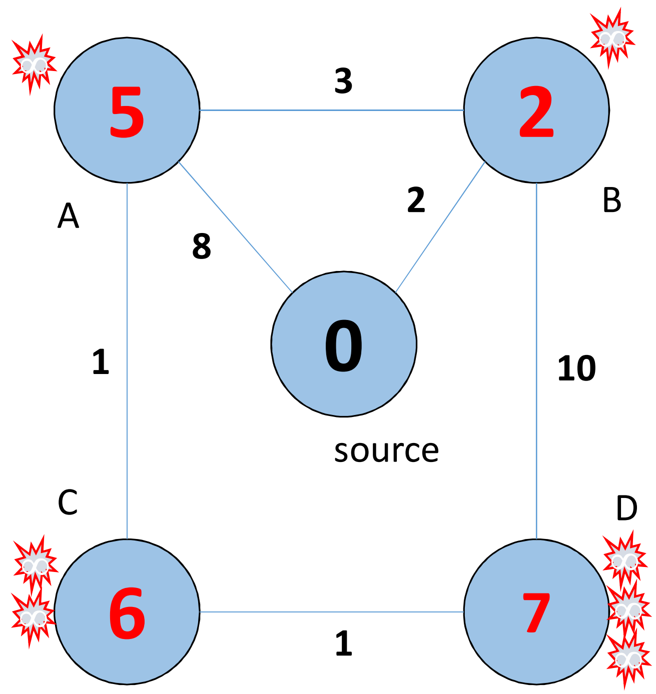
Example: SSSP



Example: SSSP



Example: SSSP

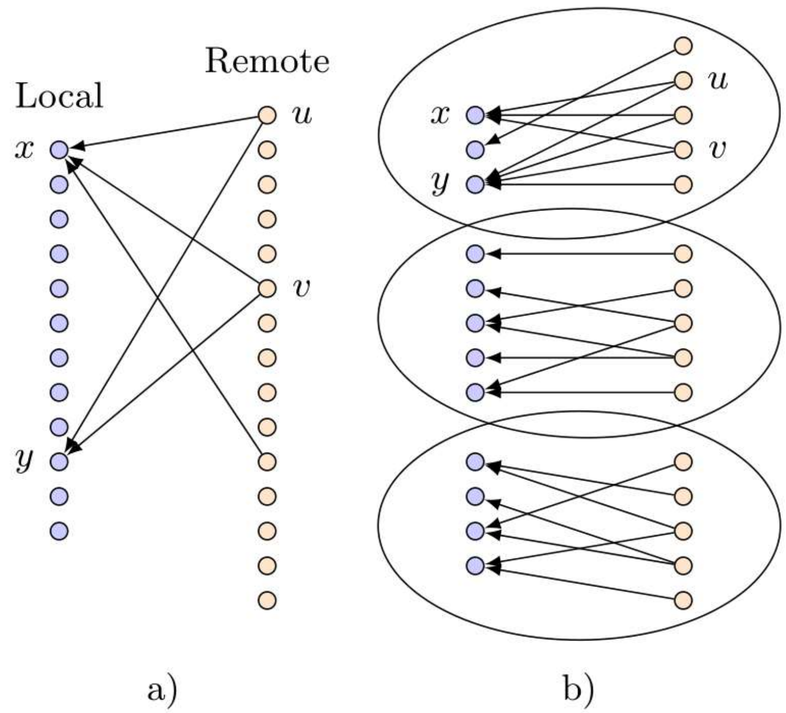


Vertex-centric vs. MapReduce

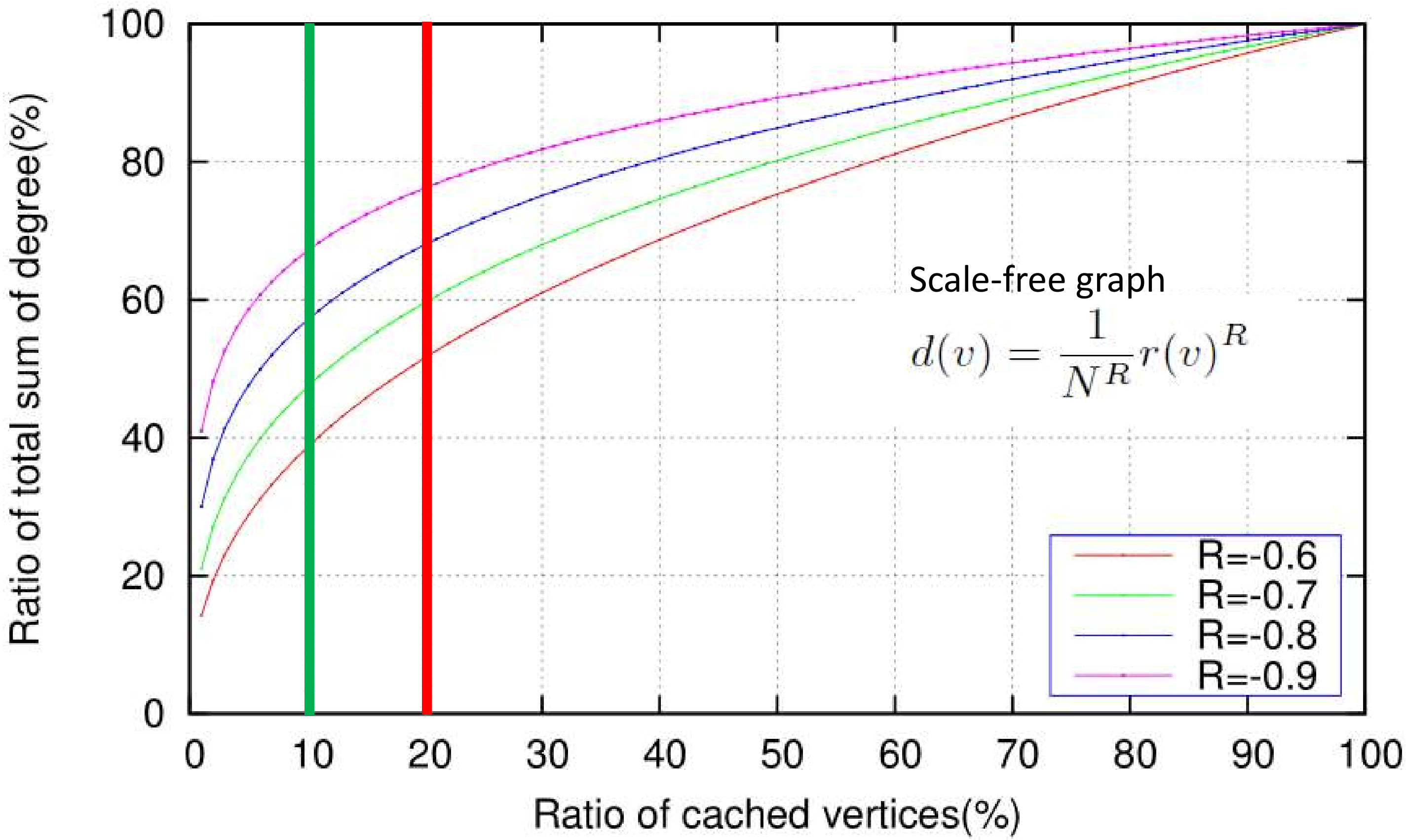
- Exploits fine-grained parallelism at the node level
- Pregel doesn’t move graph partitions over network, only messages among nodes are passed at the end of each iteration
- Many graph algorithms cannot be expressed using vertex-centric computation model intuitively and elegantly

Communication optimization

Bipartite view of a graph on a local machine



Message cache (“80/20” rule in real graphs)



Matrix arithmetic

Representative system: Pegasus

- Open source large graph mining system
  - Implemented on Hadoop

- Convert graph mining operations into iterative matrix-vector multiplications

- Pegasus uses an n by n matrix M and a vector v of size n to represent graphs

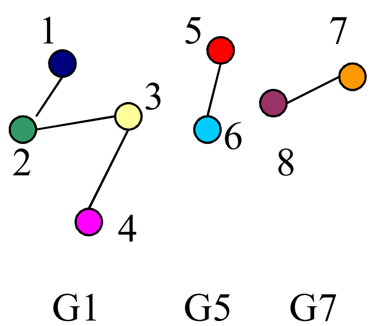
Generalized Iterated Matrix-Vector Multiplication

$M \times v = v'$, where $v'_i \boxed{=} \boxed{\sum_{j=1}^{n}} \boxed{m_{i,j} \times v_j}$

- Three primitive graph mining operations
  - $combine2(m_{i,j}, v_j)$: multiply $m_{i,j}$ and v_j
  - $combineAll_i(x_1, ..., x_n)$: sum n all the multiplication results from $combine2$
  - $assign(v_i, v_{new})$: decide how to update v_i with v_{new}

- Graph mining problems are solved by **customizing** the three operations

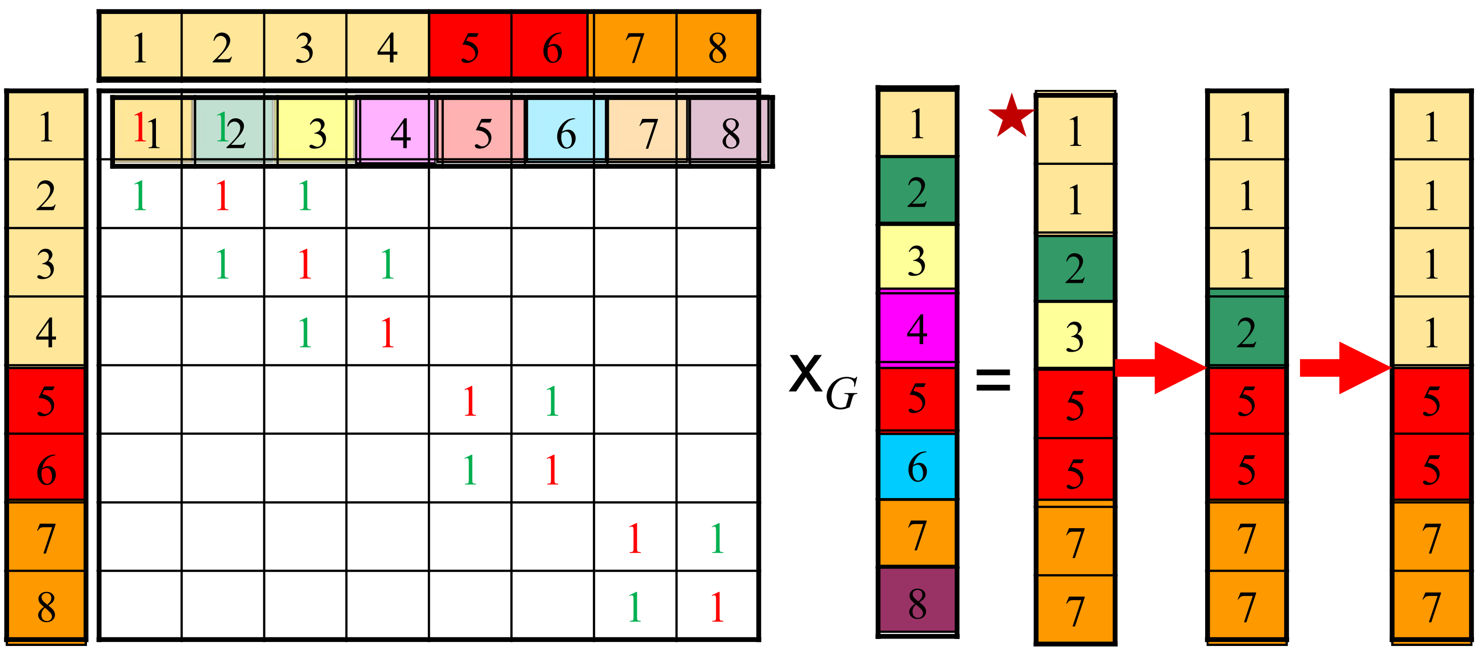
Example: connected components



	1	2	3	4	5	6	7	8
1		1						
2	1		1					
3		1		1				
4			1					
5						1		
6					1			
7								1
8							1	

Source: Pegasus, Kevin Andryc, 2011

Example: connected components



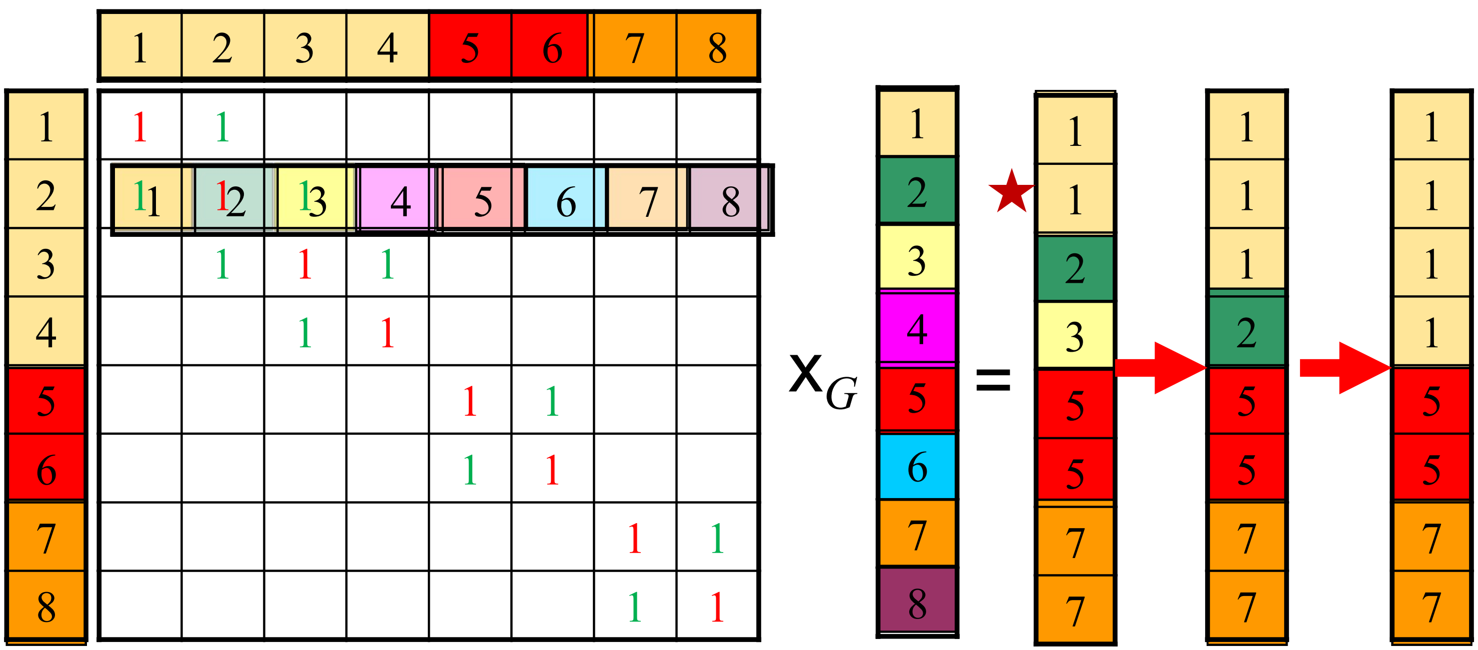
$\texttt{combine2}(m_{i,j}, v_j) = m_{i,j} \times v_j.$

$\texttt{combineAll}_i(x_1, ..., x_n) = \text{MIN}\{x_j \mid j = 1..n\}$

$\texttt{assign}(v_i, v_{new}) = \text{MIN}(v_i, v_{new}).$

Adapted from: Pegasus, Kevin Andryc, 2011

Example: connected components



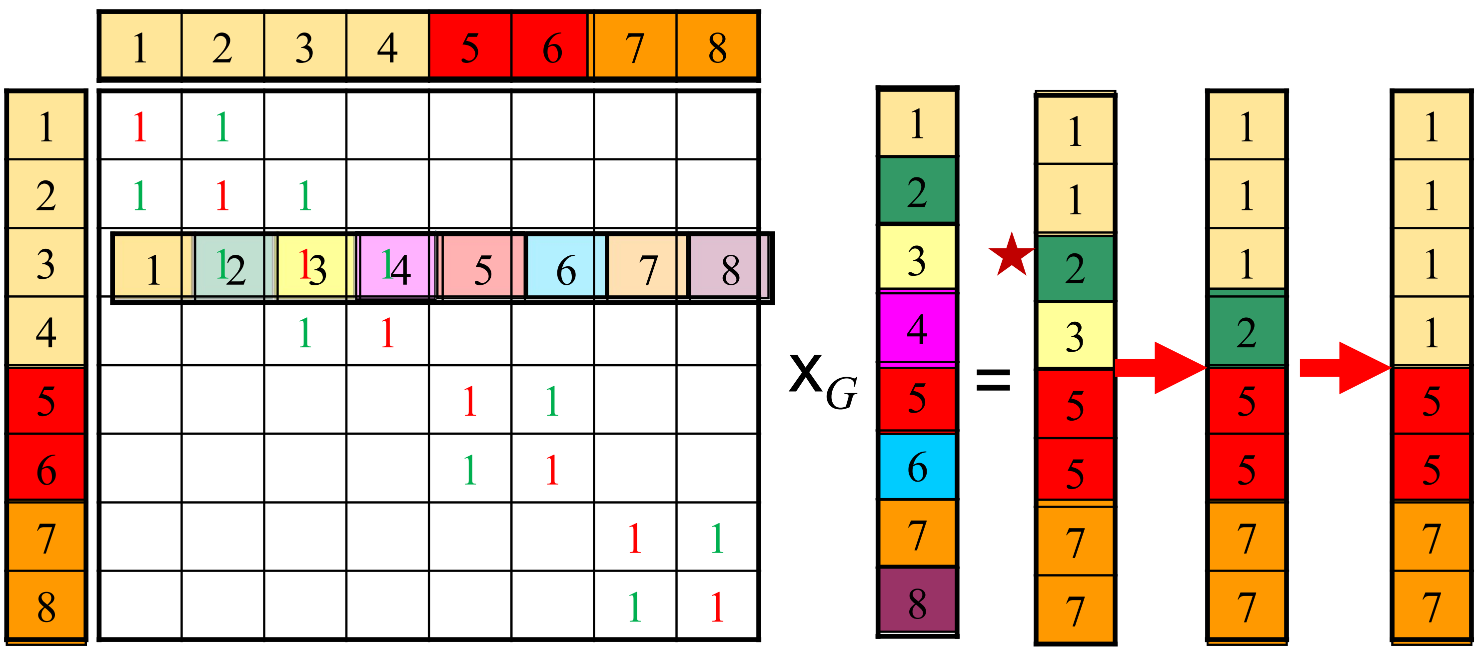
$\texttt{combine2}(m_{i,j}, v_j) = m_{i,j} \times v_j.$

$\texttt{combineAll}_i(x_1, ..., x_n) = \text{MIN}\{x_j \mid j = 1..n\}$

$\texttt{assign}(v_i, v_{new}) = \text{MIN}(v_i, v_{new}).$

Adapted from: Pegasus, Kevin Andryc, 2011

Example: connected components



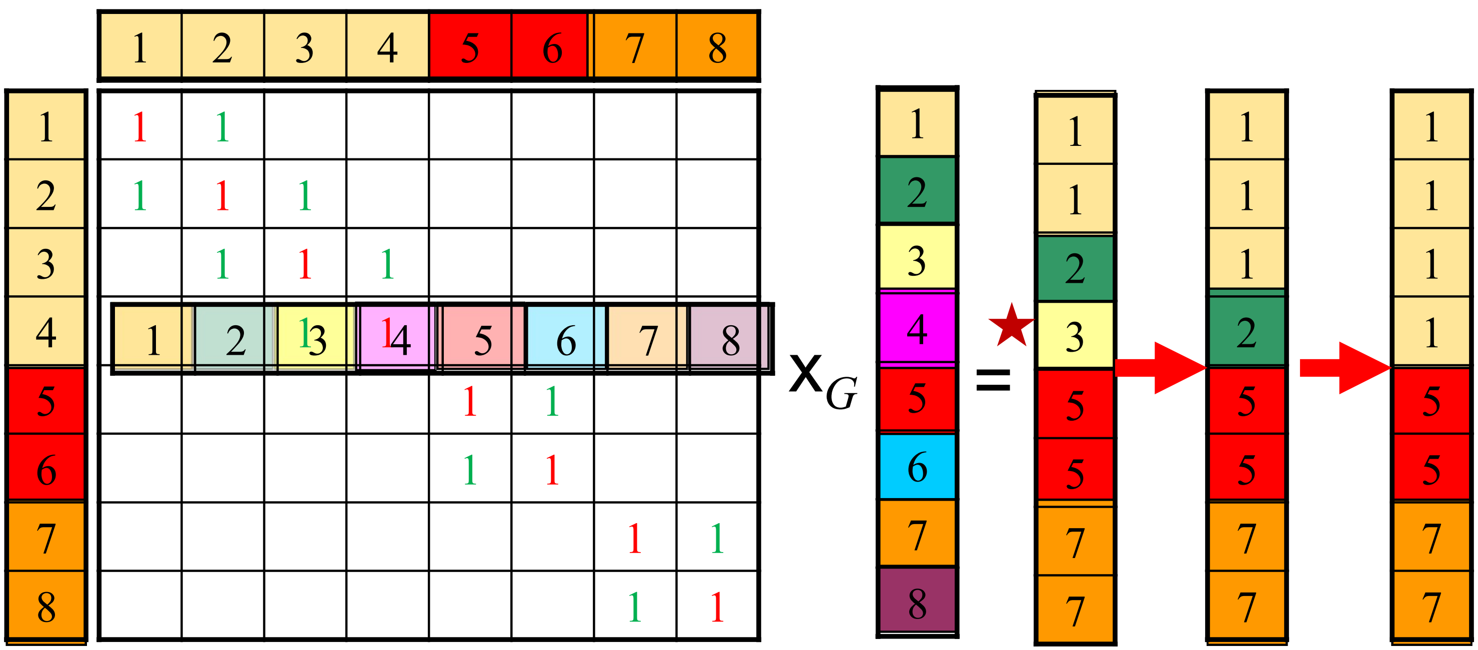
$\texttt{combine2}(m_{i,j}, v_j) = m_{i,j} \times v_j.$

$\texttt{combineAll}_i(x_1, ..., x_n) = \text{MIN}\{x_j \mid j = 1..n\}$

$\texttt{assign}(v_i, v_{new}) = \text{MIN}(v_i, v_{new}).$

Adapted from: Pegasus, Kevin Andryc, 2011

Example: connected components



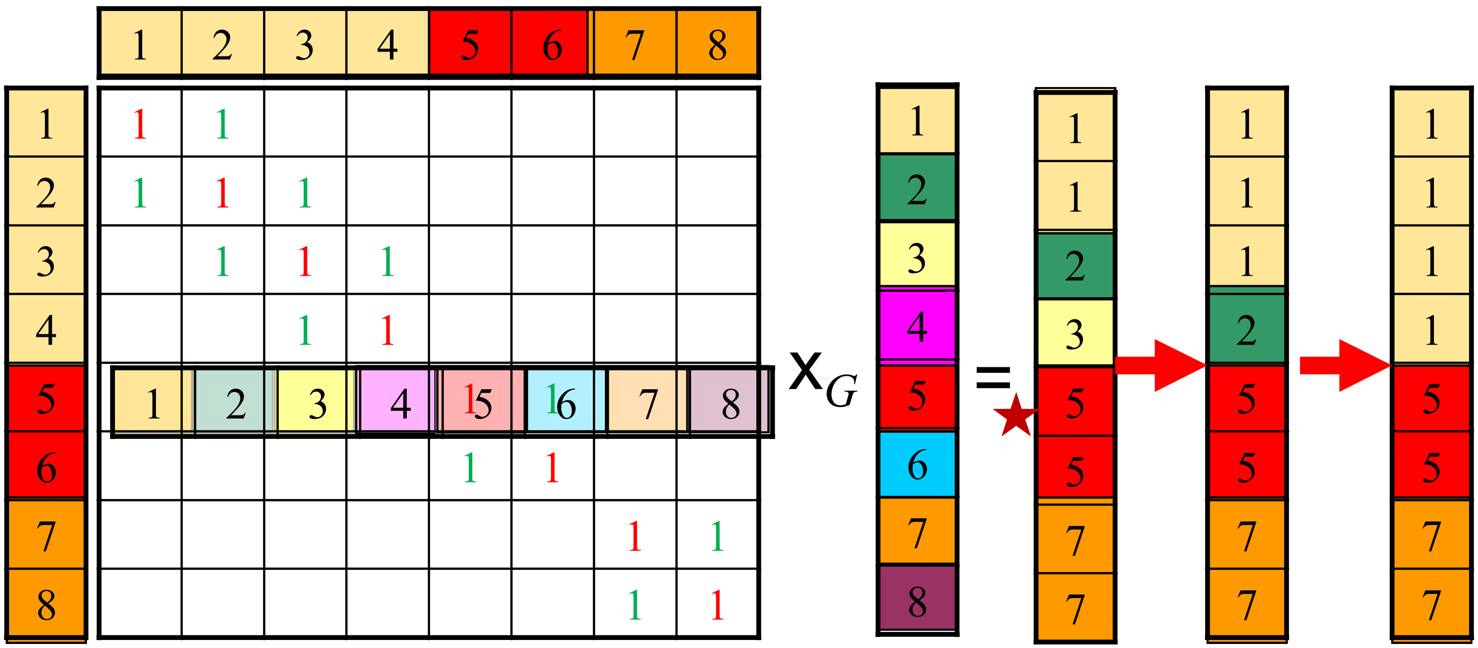
combine2$(m_{i,j}, v_j)$ = $m_{i,j} \times v_j$.

combineAll$_i(x_1, ..., x_n)$ = MIN$\{x_j \mid j = 1..n\}$

assign(v_i, v_{new}) = MIN(v_i, v_{new}).

Adapted from: Pegasus, Kevin Andryc, 2011

Example: connected components



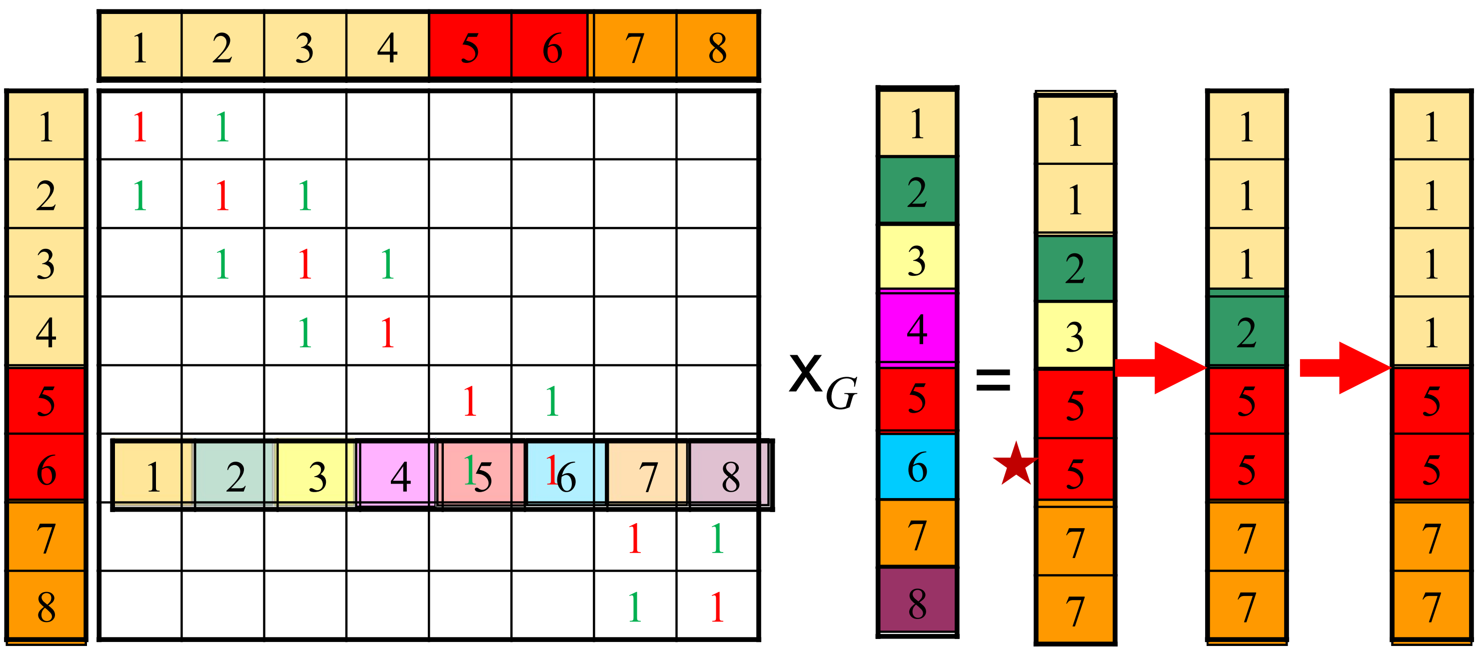
$\texttt{combine2}(m_{i,j}, v_j) = m_{i,j} \times v_j.$

$\texttt{combineAll}_i(x_1, ..., x_n) = \text{MIN}\{x_j \mid j = 1..n\}$

$\texttt{assign}(v_i, v_{new}) = \text{MIN}(v_i, v_{new}).$

Adapted from: Pegasus, Kevin Andryc, 2011

Example: connected components



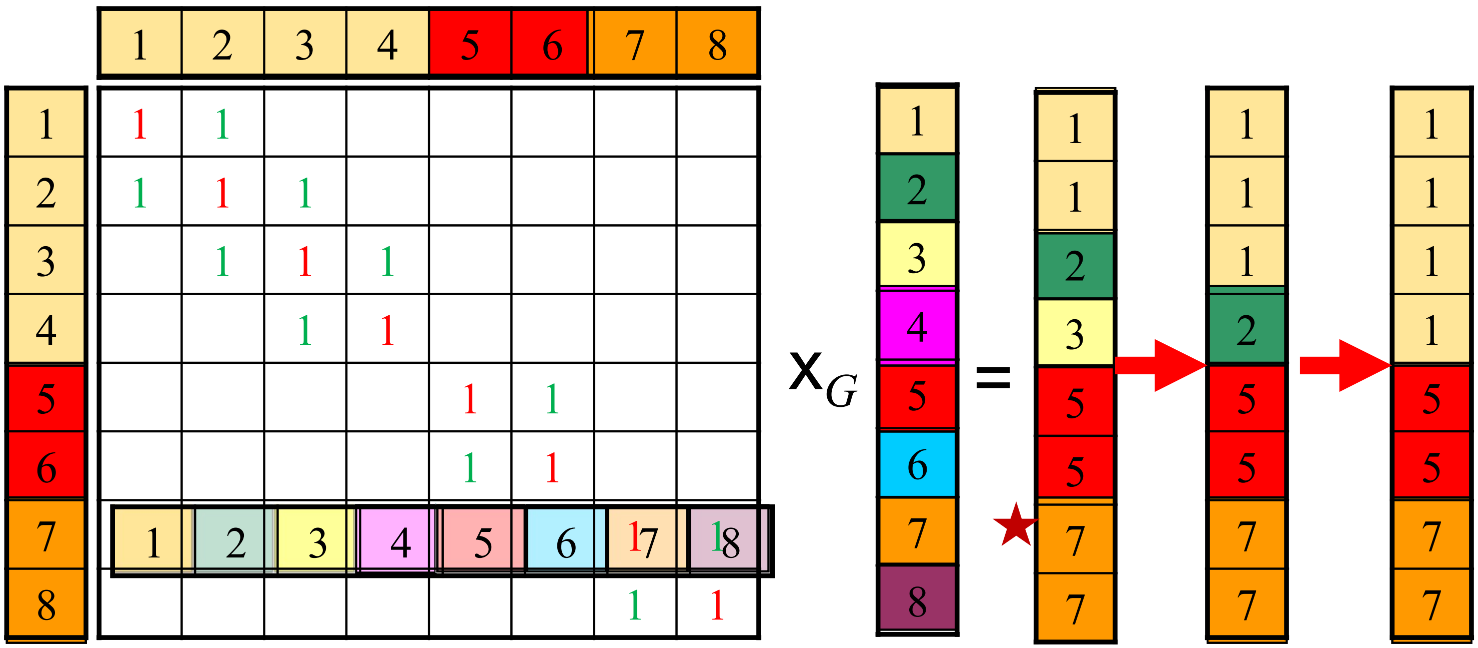
$\texttt{combine2}(m_{i,j}, v_j) = m_{i,j} \times v_j.$

$\texttt{combineAll}_i(x_1, ..., x_n) = \text{MIN}\{x_j \mid j = 1..n\}$

$\texttt{assign}(v_i, v_{new}) = \text{MIN}(v_i, v_{new}).$

Adapted from: Pegasus, Kevin Andryc, 2011

Example: connected components



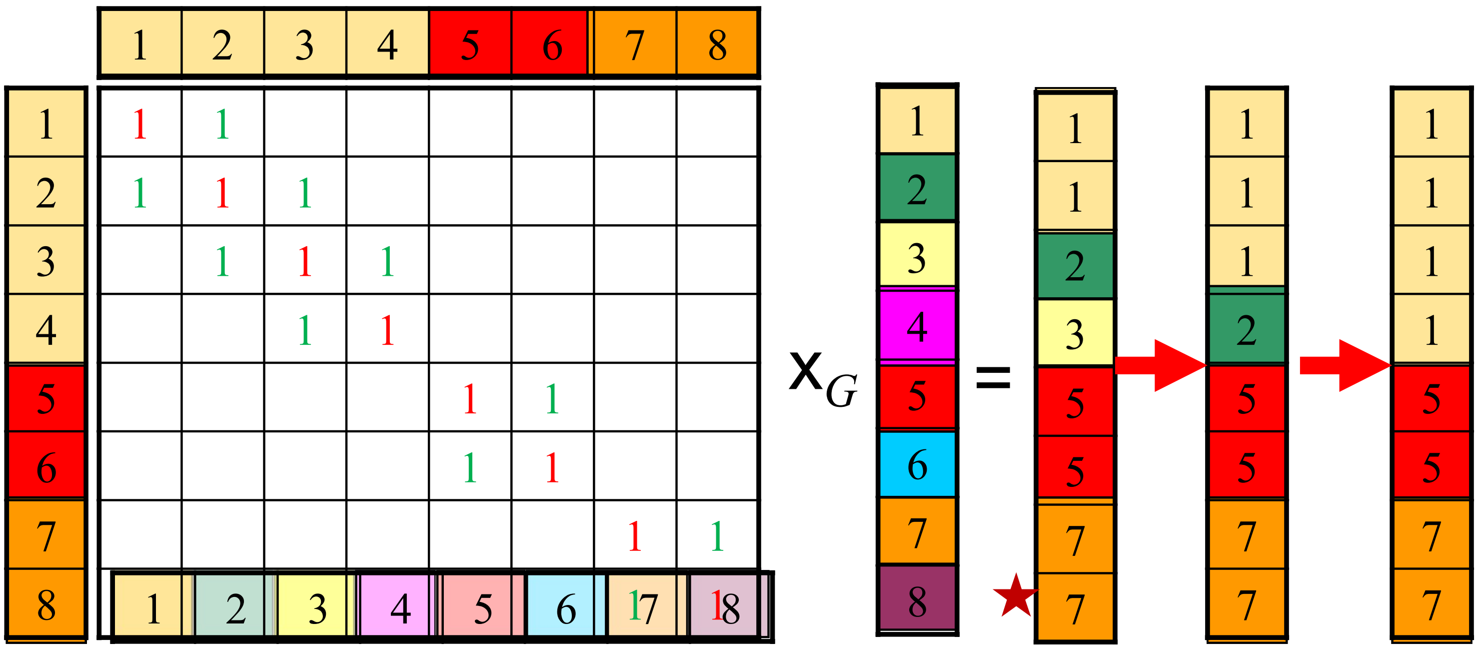
$\texttt{combine2}(m_{i,j}, v_j) = m_{i,j} \times v_j.$

$\texttt{combineAll}_i(x_1, ..., x_n) = \mathrm{MIN}\{x_j \mid j = 1..n\}$

$\texttt{assign}(v_i, v_{new}) = \mathrm{MIN}(v_i, v_{new}).$

Adapted from: Pegasus, Kevin Andryc, 2011

Example: connected components



$\texttt{combine2}(m_{i,j}, v_j) = m_{i,j} \times v_j.$

$\texttt{combineAll}_i(x_1, ..., x_n) = \text{MIN}\{x_j \mid j = 1..n\}$

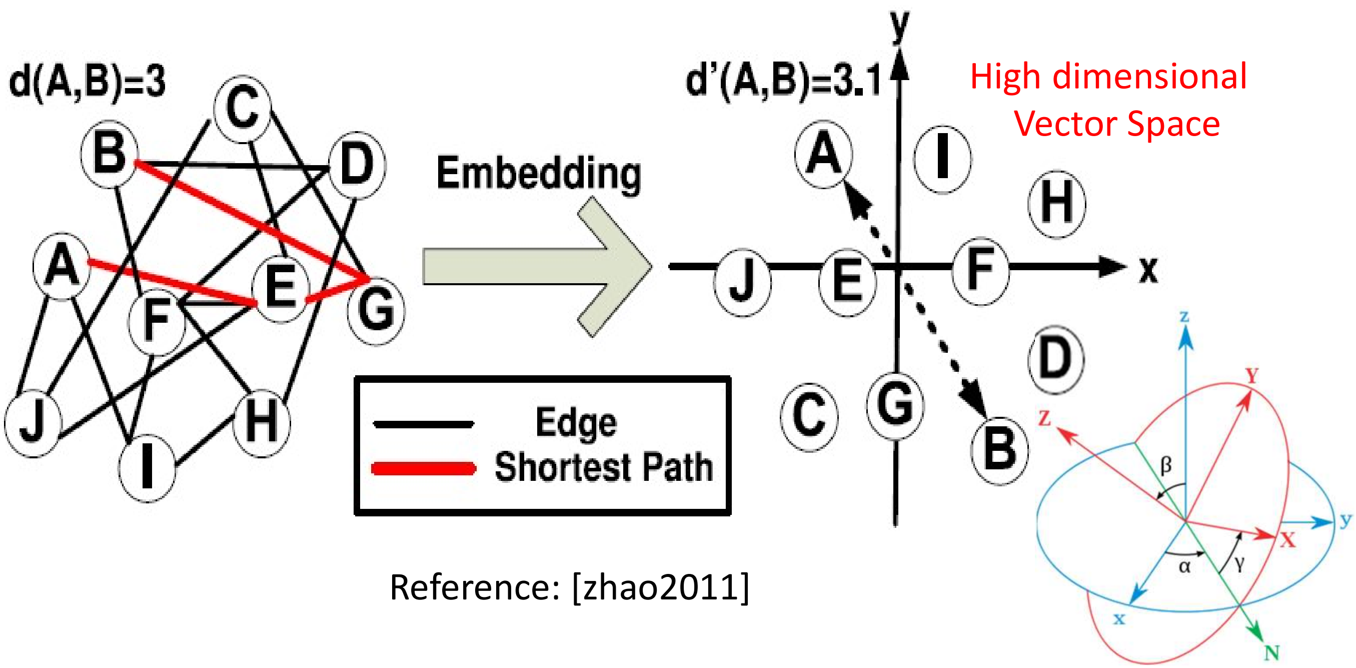
$\texttt{assign}(v_i, v_{new}) = \text{MIN}(v_i, v_{new}).$

Adapted from: Pegasus, Kevin Andryc, 2011

Graph embedding

Graph embedding

- Embed a graph into a geometric space so that distances in the space preserve the shortest distances in the graph



Reference: [zhao2011]

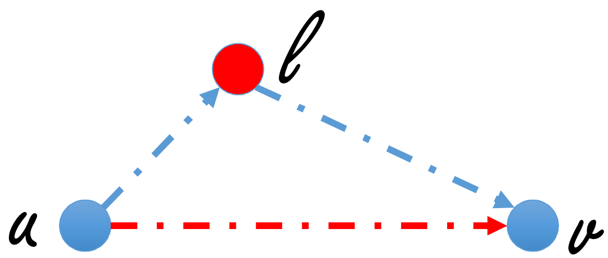
Application: distance oracle

- Choose a small number of landmarks (~100)
  - Heuristics: Degree , betweenness, …
- Calculate the distances from each landmark to all other vertices using *BFS starting from each landmark*
- Calculate the embedding of landmarks using the *downhill simplex method* according to the distances between landmarks
- Calculate the embedding of other vertices using the *downhill simplex method* according to the distances from these vertices to landmarks

Reference: [Qi VLDB2014]

Distance oracle in a nutshell

- Step 1: Using sketch to give the lower and upper bound of the shortest distance between two vertices



$$|\boldsymbol{d}(\boldsymbol{u},\boldsymbol{l}) - \boldsymbol{d}(\boldsymbol{l},\boldsymbol{v})| \le \boldsymbol{d}(\boldsymbol{u},\boldsymbol{v}) \le \boldsymbol{d}(\boldsymbol{u},\boldsymbol{l}) + \boldsymbol{d}(\boldsymbol{l},\boldsymbol{v})$$

Triangle Inequality

$$\boldsymbol{l}(\boldsymbol{u},\boldsymbol{v}) \le \boldsymbol{d}(\boldsymbol{u},\boldsymbol{v}) \le \boldsymbol{r}(\boldsymbol{u},\boldsymbol{v})$$

Distance oracle in a nutshell

- Step 2: Refining results using graph embedding

$$d(u,v) = \begin{cases} \bar{d}(u,v) & \text{if } l(u,v) \leq \bar{d}_{u,v} \leq r(u,v); \\ l(u,v) & \text{if } \bar{d}_{u,v} < l(u,v); \\ r(u,v) & \text{if } \bar{d}_{u,v} > r(u,v); \end{cases}$$

$\bar{d}(u,v)$ is the coordinate distance in the embedding space

Real-time Query Processing

Query processing

- Where do latencies come from?

- Index-free query processing

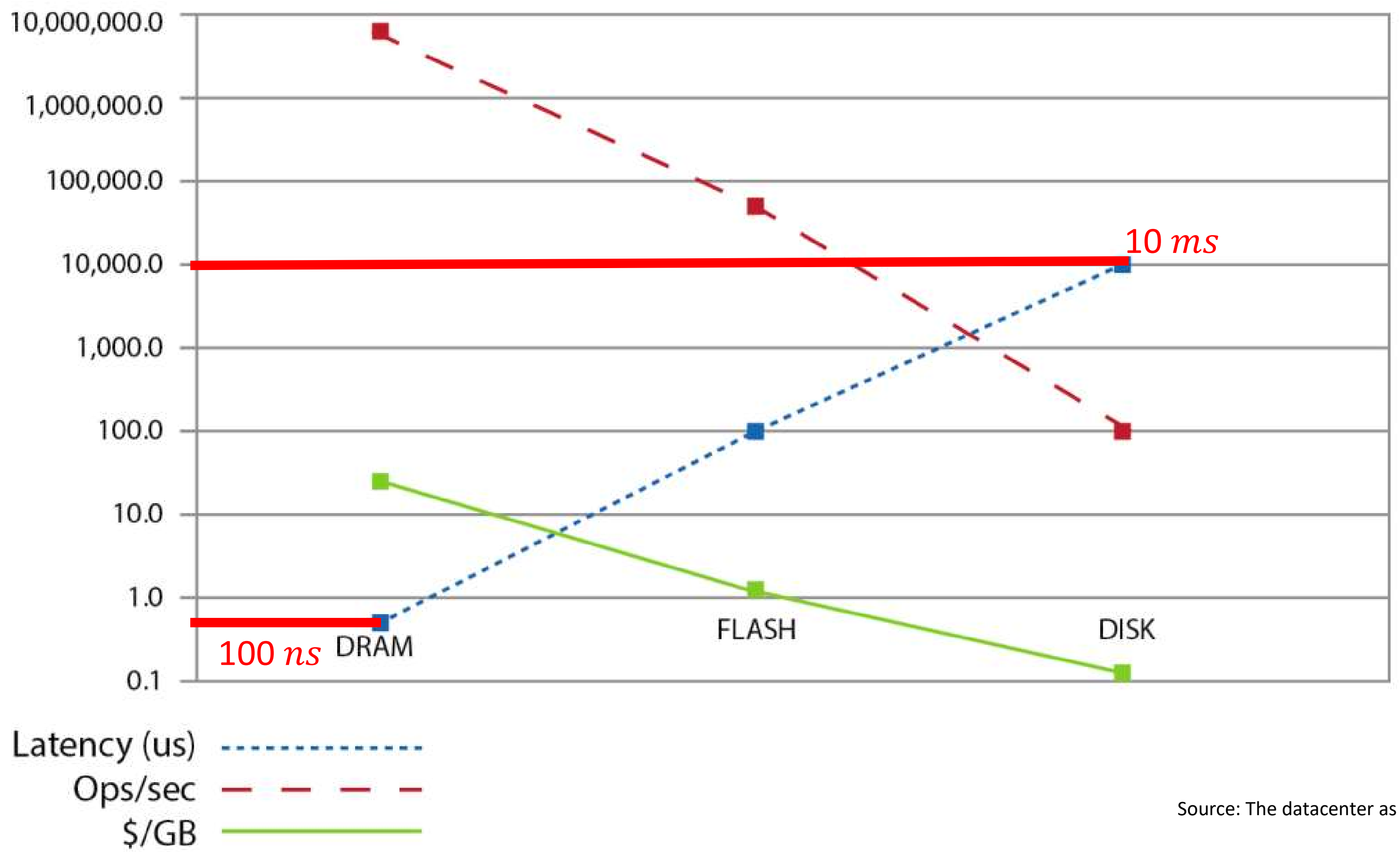
People search challenge in Facebook graph

- Among adult Facebook users, the average number of friends is 338.

$$
\begin{array}{r}
338 \\
+338 \times 338 \\
+338 \times 338 \times 338 \\
=38{,}729{,}054
\end{array}
$$

Can we search a person in one's 3-hop neighborhood within 500 ms?

Latency, Bandwidth, and Capacity



Source: The datacenter as a computer (book)

Disk-based approach

338
+338 x 338
+338 x 338 x 338
=38,729,054 → 387,290,540 ms
= 4.5 days

each disk seek + read: > 10 ms

RAM-based approach

- DRAM latency: 100 ns

10 million reads/writes per second

1 million vertex-level read/write per second

38,729,054 vertices to access, it takes at least 38 seconds.

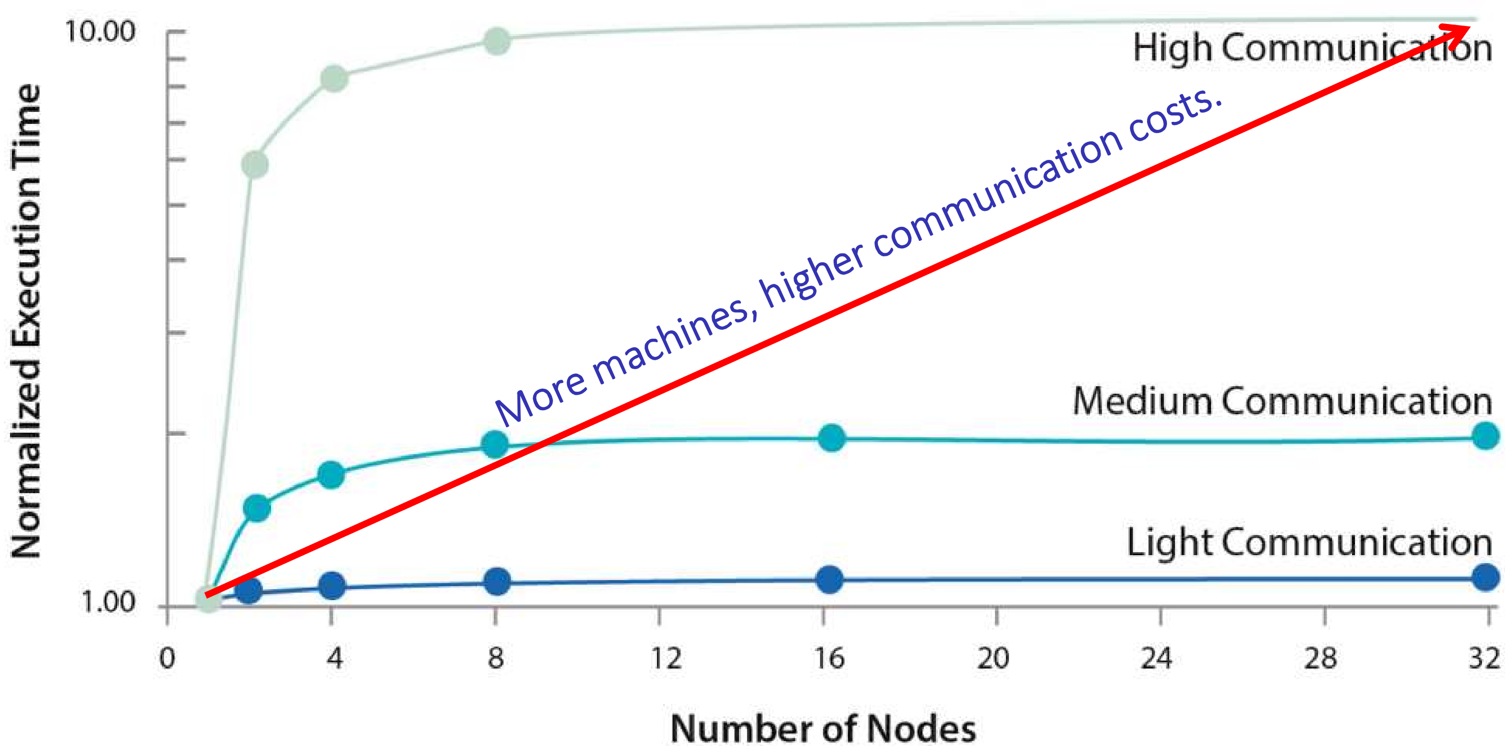
Where do latencies come from?

Algorithms — Single-threaded, multi-threaded, synchronous, asynchronous

Communications — Multi-cores, Multi-nodes

Storage — DRAM, Flash, or Disk

Move computation, instead of data!



Source: The datacenter as a computer (book)

If you care about latency, do not use the shared-memory model in a distributed setting.

Lessons learned so far (how to reduce latencies)

- RAM (hardware sometimes does matter a lot)
  - The stupid buy faster computers, smart ones write better programs?
- Avoid moving data
- ……

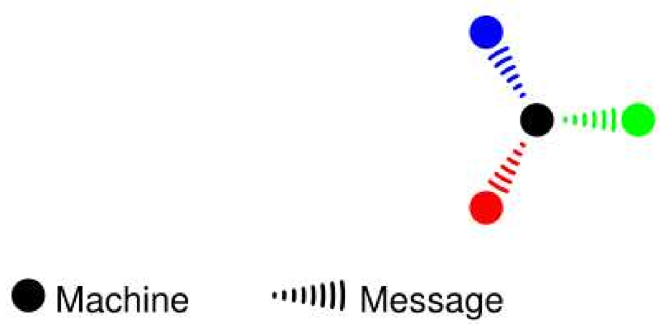
Lessons learned so far (how to reduce latencies)

- RAM (hardware sometimes does matter a lot)
  - The stupid buy faster computers, smart ones write better programs?
- Avoid moving data
- Avoid unnecessary synchronizations

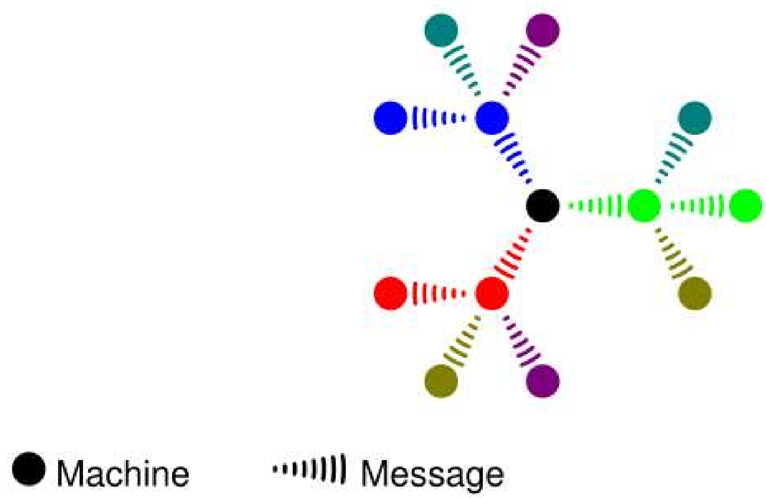
Make programming harder

Fan-out Search

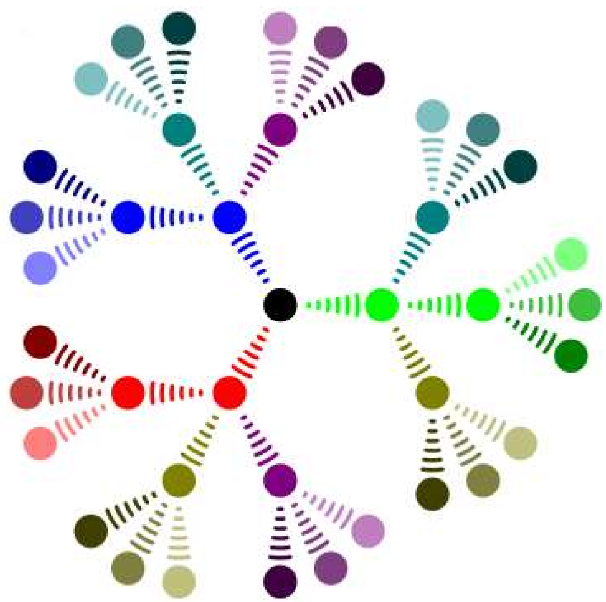
Fan-out Search



Fan-out Search



Fan-out Search



● Machine ····|||| Message

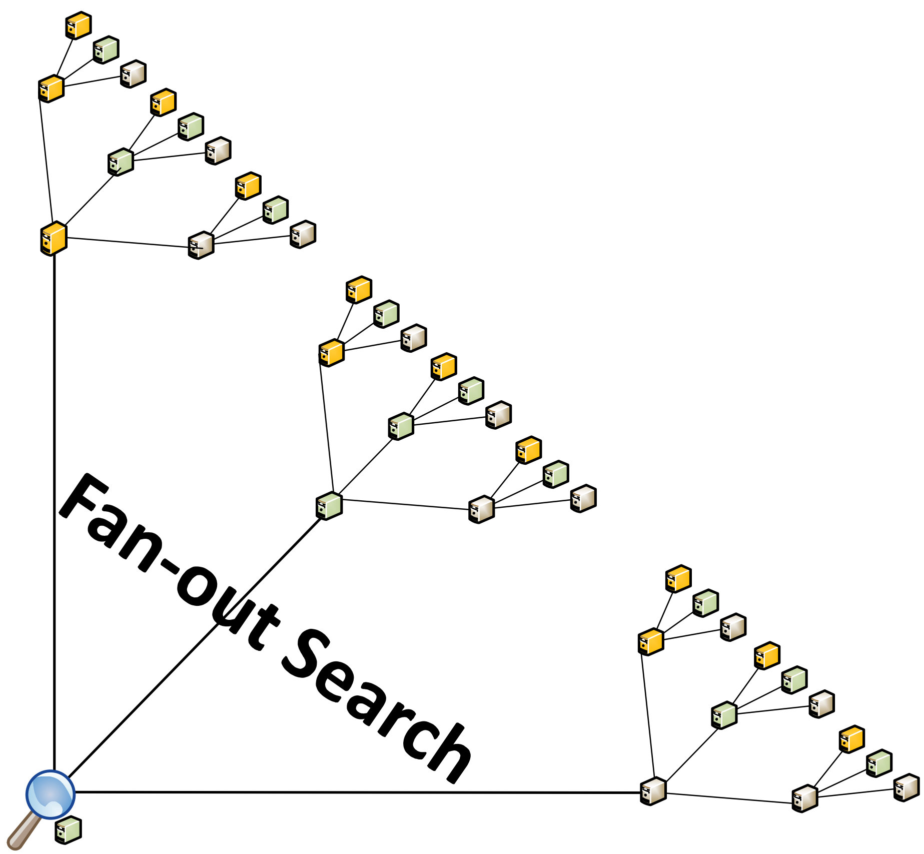
$$MessageCount = \sum_{i=1}^{h} N^i$$

Lessons learned so far

- RAM (Hardware sometimes does matter a lot)
  - The stupid buy faster computers, smart ones write better programs?
- Avoid moving data
- Avoid unnecessary synchronizations

Makes programming harder

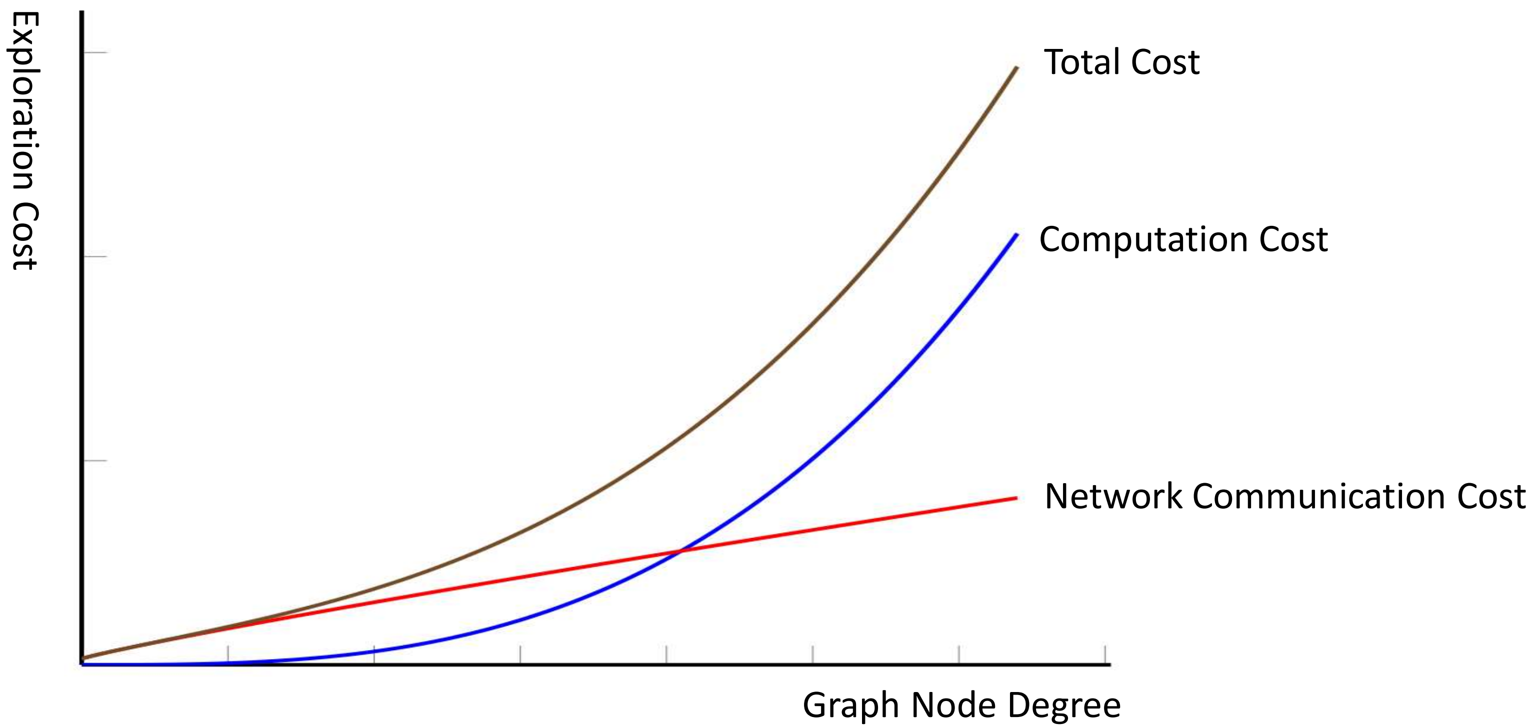
Asynchronous fan-out search



Hop	Msg #	Node # per machine
1	n	$\frac{d}{n}$
2	n^2	$\frac{d^2}{n}$
3	n^3	$\frac{d^3}{n}$

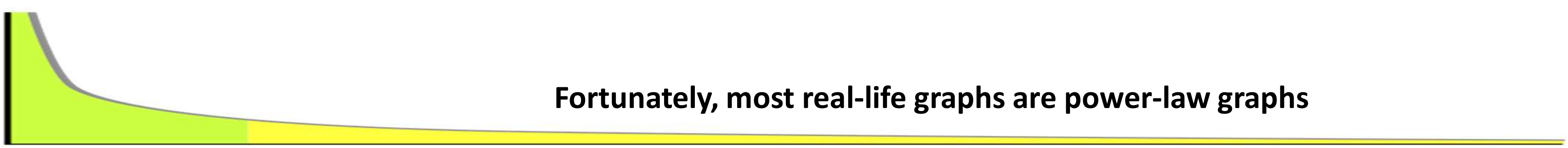
n is the server count
d is the average degree

Cost of Graph Exploration



The scalability of fan-out search

Node # N	Edge # E	Node Degree	Network Message # $p = \sum_{k=0}^{h} M^k$	CPU Workload Per Machine $q = \sum_{i=0}^{h} \frac{d^i}{M}$	Total Cost $f(p) + g(q)$
2.4×10^9	2.4×10^{14}	10^5	4,368 (M=16, h=3)	10^{14}	2 days
2.4×10^9	17.4×10^9	$0 \sim 5000$	4,368 (M=16, h=3)	6.3×10^7	< 120 ms



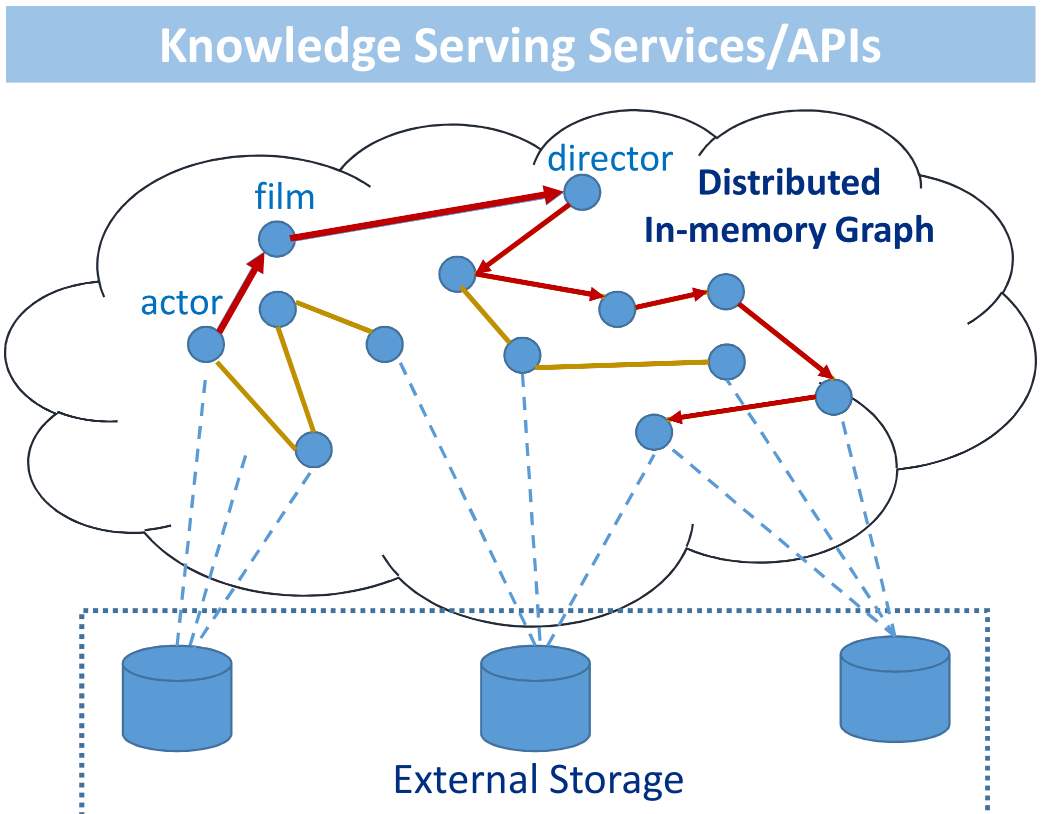
h: *hop count* M: *Machine Count* d: *Average Node Degree*

Online query processing

- Where do latencies come from?

- Index-free query processing

Query processing via graph exploration



Online query example: subgraph matching

Procedure:

1. Break a graph into basic units (edges, paths, frequent subgraphs, …)

2. Build index for every possible basic unit

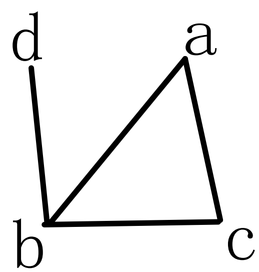
3. Decompose a query into multiple basic unit queries, and join the results

Case study: distributed subgraph matching

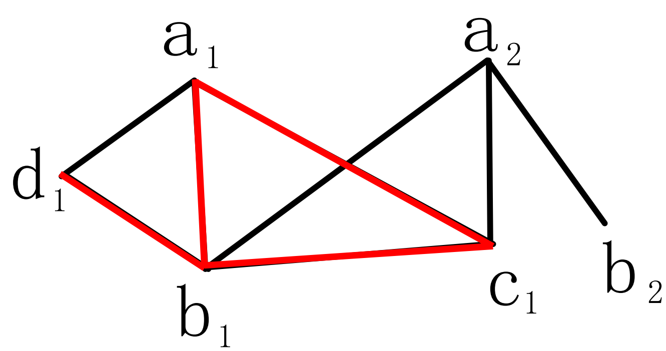
Procedure:

1. Break a query into basic units
2. **Match the basic units in parallel on the fly**
3. Join the results

Subgraph matching

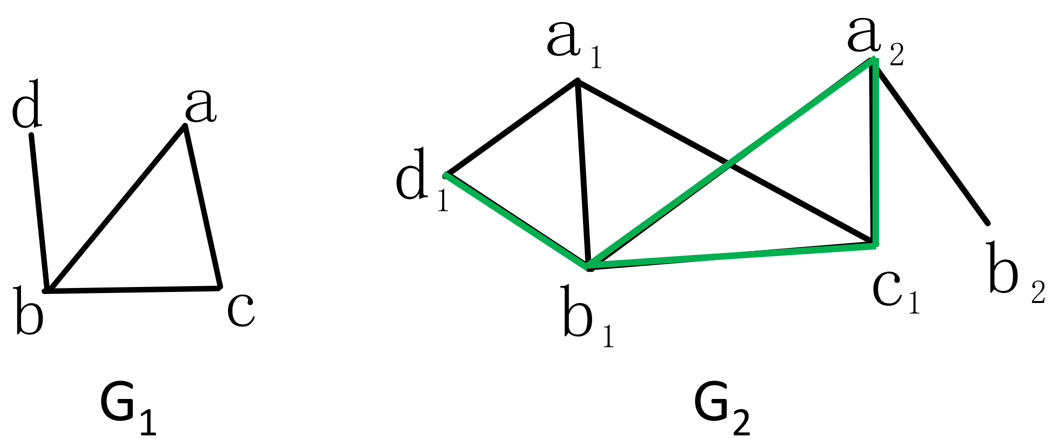


G_1

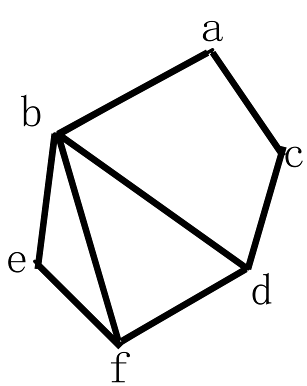


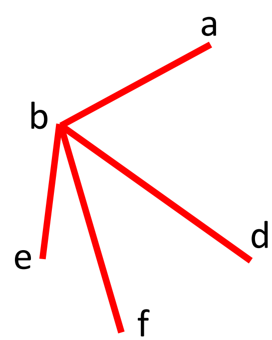
G_2

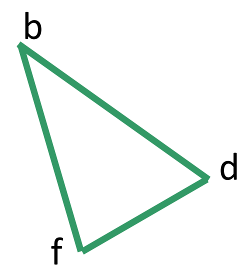
Subgraph matching

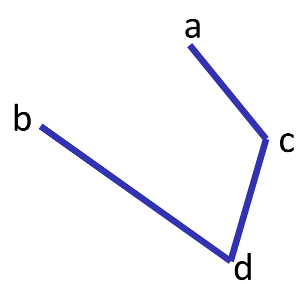


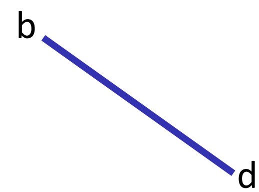
Basic unit for distributed subgraph matching





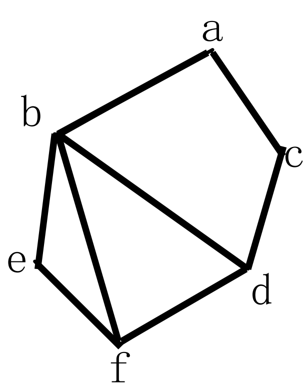


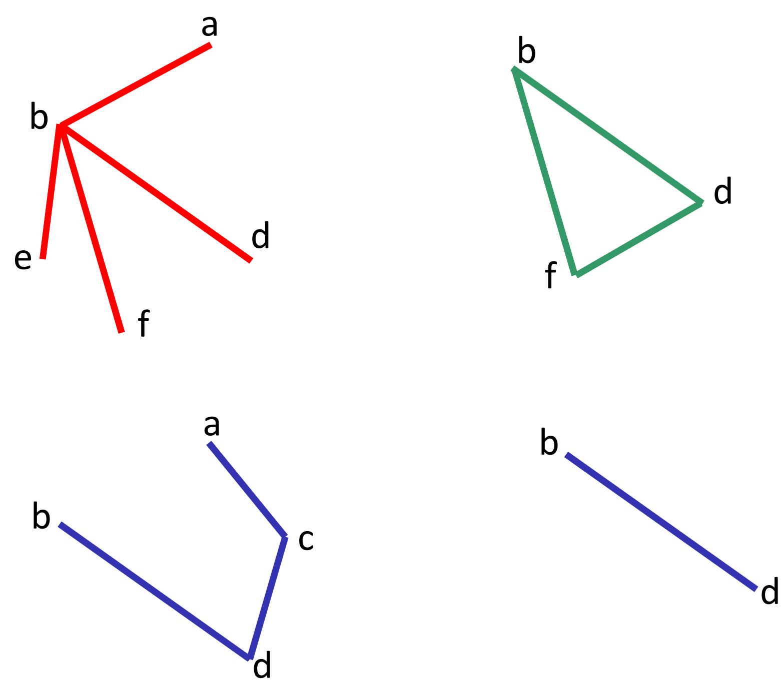




As a basic unit, which one is the best?

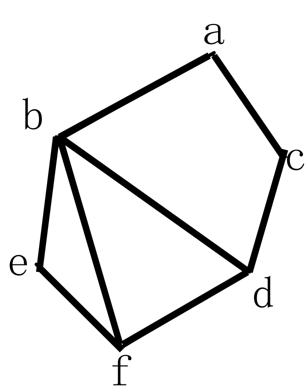
Basic unit for distributed subgraph matching

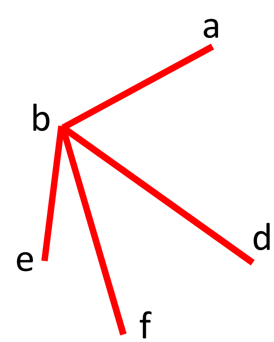




As a basic unit, which one is the best?

Basic unit for distributed subgraph matching



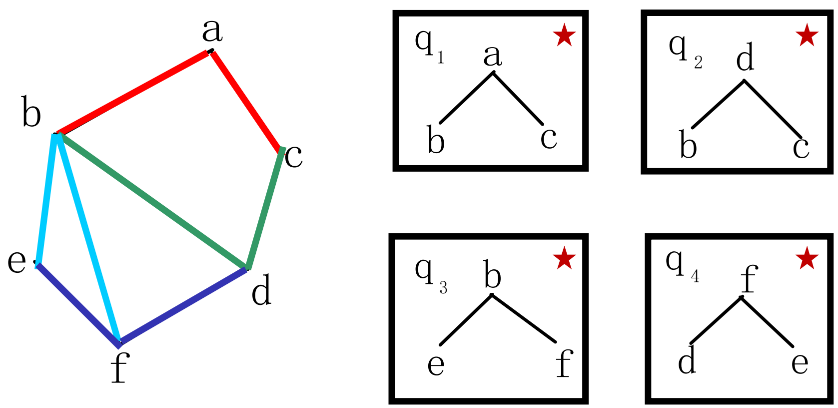


Twig

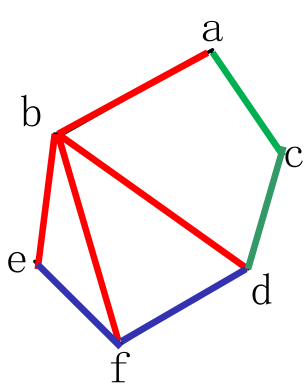
- Easy to decompose
- Height is always one
  - It at most needs to cross the network once

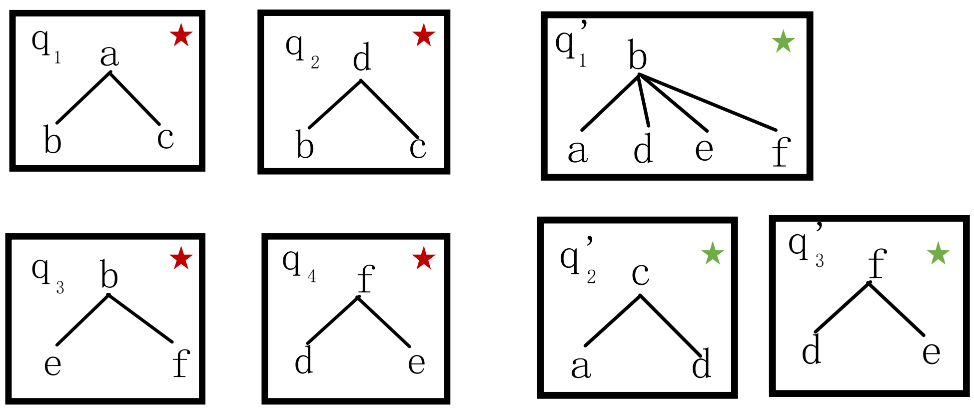
As a basic unit, which one is the best?

Query decomposition



Query decomposition

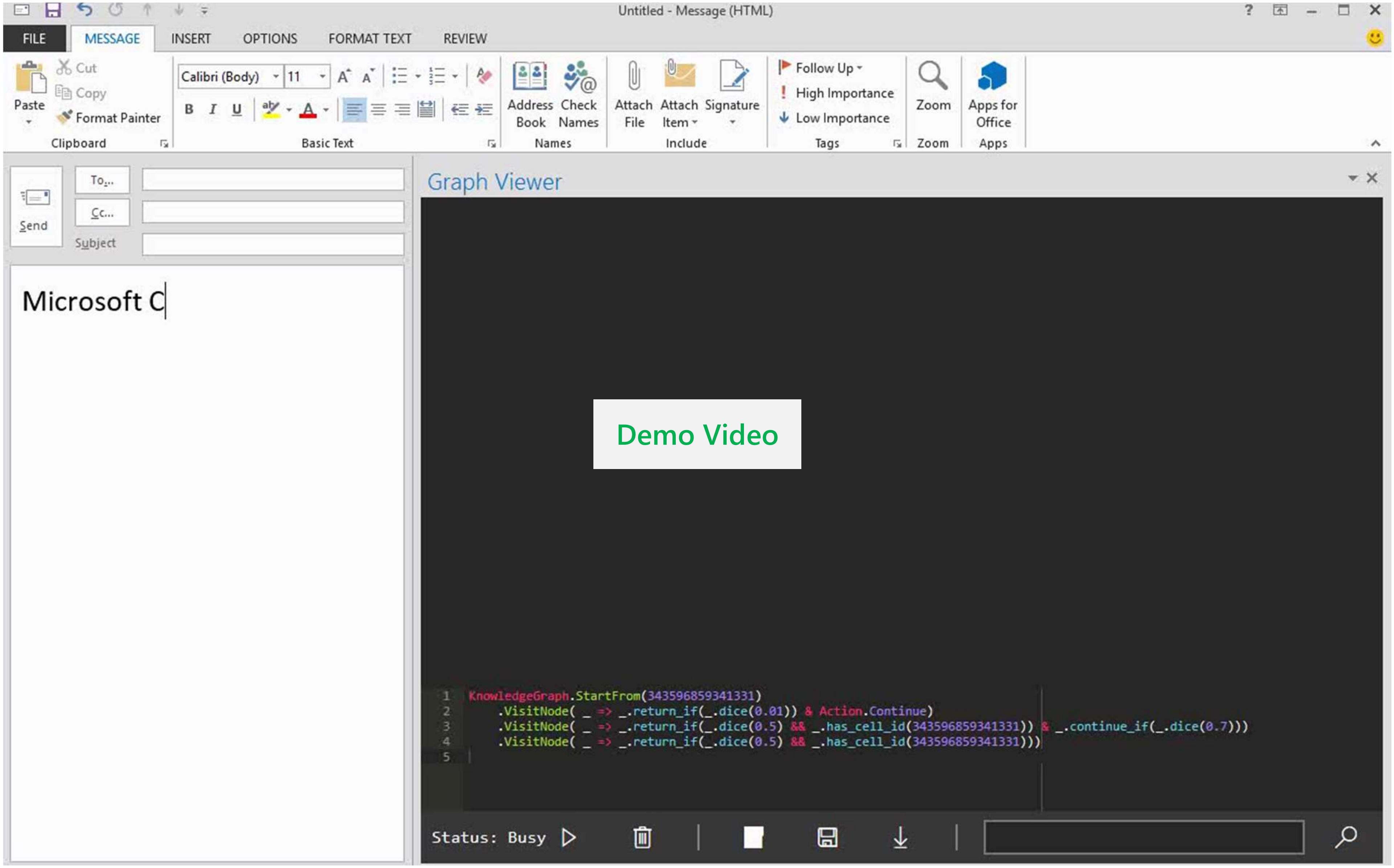




Query optimization problems

- How to choose a good query decomposition
- How to choose a good execution order
- How to choose a good join order

Demo


Untitled - Message (HTML)
FILE
MESSAGE
INSERT
OPTIONS
FORMAT TEXT
REVIEW
Paste
Cut
Copy
Format Painter
Clipboard
Calibri (Body)
11
Basic Text
Address Book
Check Names
Names
Attach File
Attach Item
Signature
Include
Follow Up
High Importance
Low Importance
Tags
Zoom
Zoom
Apps for Office
Apps
Send
To...
Cc...
Subject
Microsoft C
Graph Viewer
Demo Video
KnowledgeGraph.StartFrom(343596859341331)
.VisitNode(_ => _.return_if(_.dice(0.01)) & Action.Continue)
.VisitNode(_ => _.return_if(_.dice(0.5) && _.has_cell_id(343596859341331)) & _.continue_if(_.dice(0.7)))
.VisitNode(_ => _.return_if(_.dice(0.5) && _.has_cell_id(343596859341331)))
Status: Busy

How can we make it fast enough

- Big data
  - hmm, we have a large variety of tools available

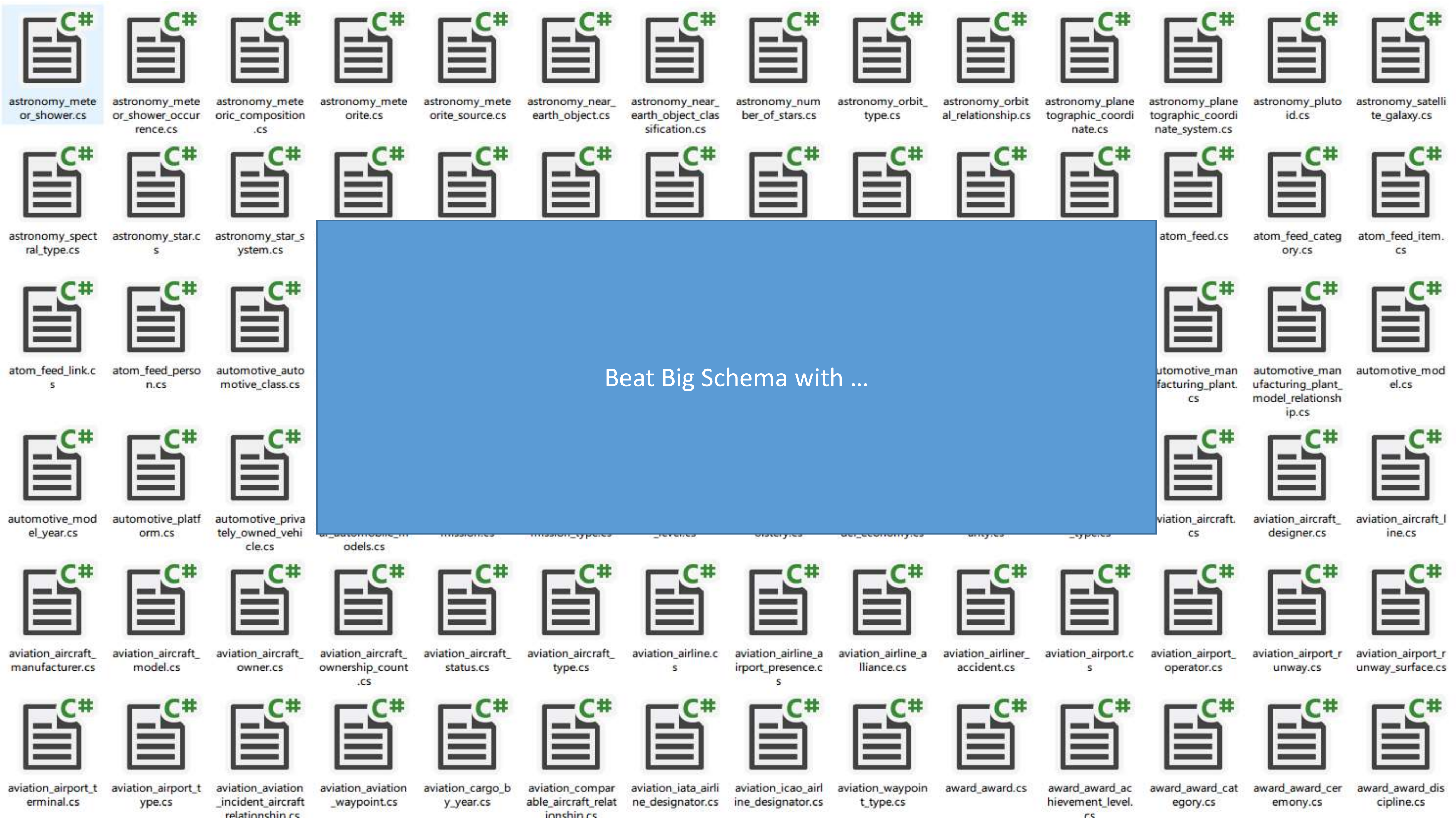
- But, how do we handle "big schema" …

  If we treat everything as texts and build indexes for these piles of words

  ⬇

  - Inefficient data processing (weakly-typed system)
  - Limited search functionality we can provide

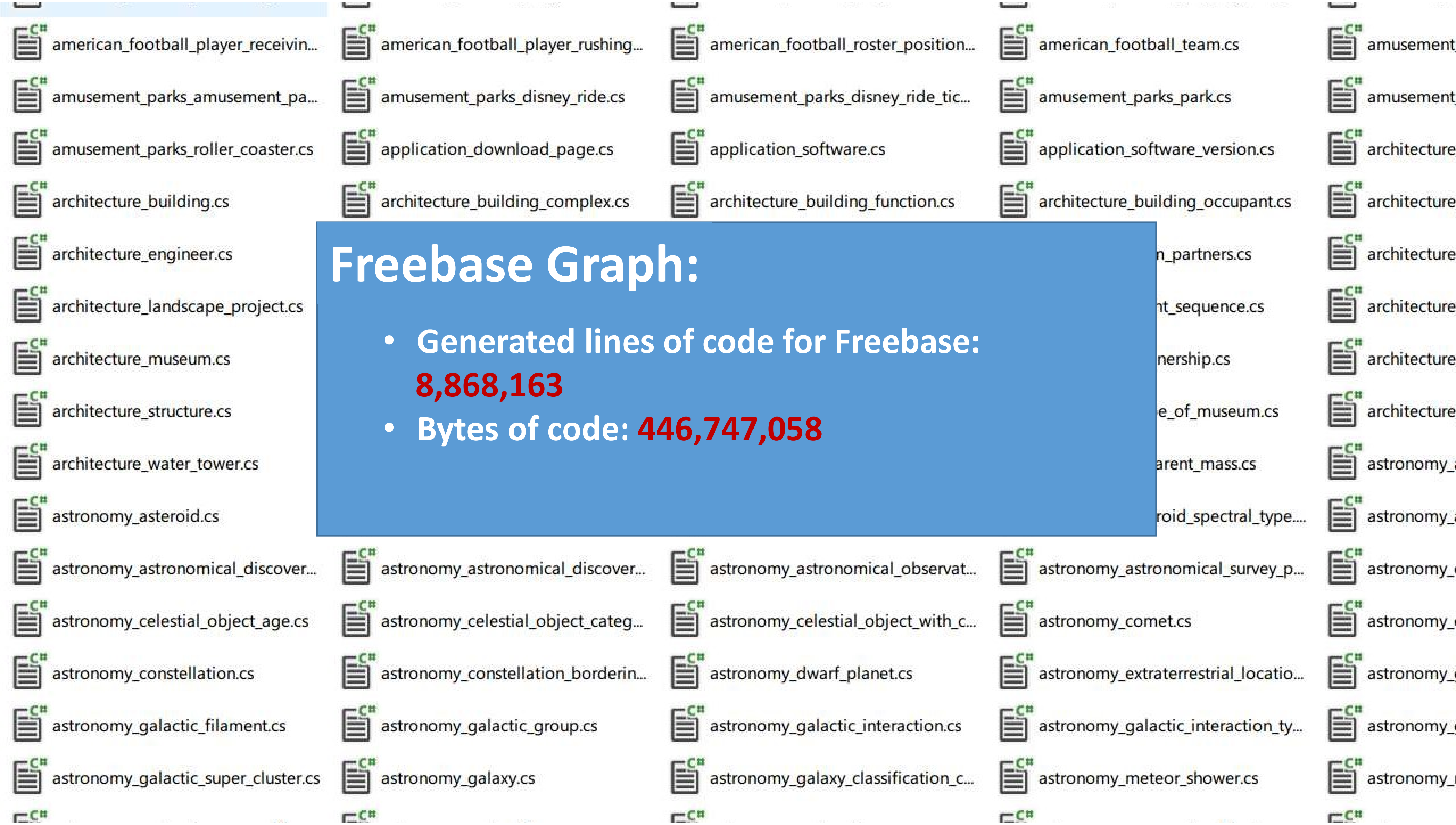
Beat Big Schema with …


astronomy_meteor_shower.cs
astronomy_meteor_shower_occurrence.cs
astronomy_meteoric_composition.cs
astronomy_meteorite.cs
astronomy_meteorite_source.cs
astronomy_near_earth_object.cs
astronomy_near_earth_object_classification.cs
astronomy_number_of_stars.cs
astronomy_orbit_type.cs
astronomy_orbital_relationship.cs
astronomy_planetographic_coordinate.cs
astronomy_planetographic_coordinate_system.cs
astronomy_plutoid.cs
astronomy_satellite_galaxy.cs
astronomy_spectral_type.cs
astronomy_star.cs
astronomy_star_system.cs
atom_feed.cs
atom_feed_category.cs
atom_feed_item.cs
atom_feed_link.cs
atom_feed_person.cs
automotive_automotive_class.cs
utomotive_manfacturing_plant.cs
automotive_manufacturing_plant_model_relationship.cs
automotive_model.cs
automotive_model_year.cs
automotive_platform.cs
automotive_privately_owned_vehicle.cs
viation_aircraft.cs
aviation_aircraft_designer.cs
aviation_aircraft_line.cs
aviation_aircraft_manufacturer.cs
aviation_aircraft_model.cs
aviation_aircraft_owner.cs
aviation_aircraft_ownership_count.cs
aviation_aircraft_status.cs
aviation_aircraft_type.cs
aviation_airline.cs
aviation_airline_airport_presence.cs
aviation_airline_alliance.cs
aviation_airliner_accident.cs
aviation_airport.cs
aviation_airport_operator.cs
aviation_airport_runway.cs
aviation_airport_runway_surface.cs
aviation_airport_terminal.cs
aviation_airport_type.cs
aviation_aviation_incident_aircraft_relationship.cs
aviation_aviation_waypoint.cs
aviation_cargo_by_year.cs
aviation_comparable_aircraft_relationship.cs
aviation_iata_airline_designator.cs
aviation_icao_airline_designator.cs
aviation_waypoint_type.cs
award_award.cs
award_award_achievement_level.cs
award_award_category.cs
award_award_ceremony.cs
award_award_discipline.cs
Beat Big Schema with …


Beat Big Schema with …
Big Code!
atom_feed_category.cs
atom_feed_item.cs
atom_feed_link.cs
atom_feed_person.cs
automotive_automotive_class.cs
automotive_company.cs
automotive_designer.cs
automotive_engine.cs
automotive_exterior_color.cs
automotive_generation.cs
automotive_make.cs
automotive_manufacturing_plant.cs
automotive_manufacturing_plant_model_rel...
automotive_similar_automobile_models.cs
automotive_transmission.cs
aviation_aircraft_line.cs
aviation_aircraft_manufacturer.cs
aviation_airline.cs
aviation_airline_airport_presence.cs
aviation_comparable_aircraft_relationship.cs
aviation_iata_airline_designator.cs
award_ceremony.cs
award_competition.cs
award_competition_type.cs
award_competitor.cs
award_hall_of_fame.cs
award_hall_of_fame_discipline.cs
award_hall_of_fame_inductee.cs
award_hall_of_fame_induction.cs
award_hall_of_fame_induction_category.cs
award_honor.cs
award_judge.cs
award_judging_term.cs
award_long_list_nomination.cs
award_ranked_item.cs
award_ranked_list.cs
award_ranking.cs
award_winner.cs
award_winning_work.cs
baseball_coach.cs
baseball_division.cs

Freebase Graph:

- **Generated lines of code for Freebase: 8,868,163**
- **Bytes of code: 446,747,058**

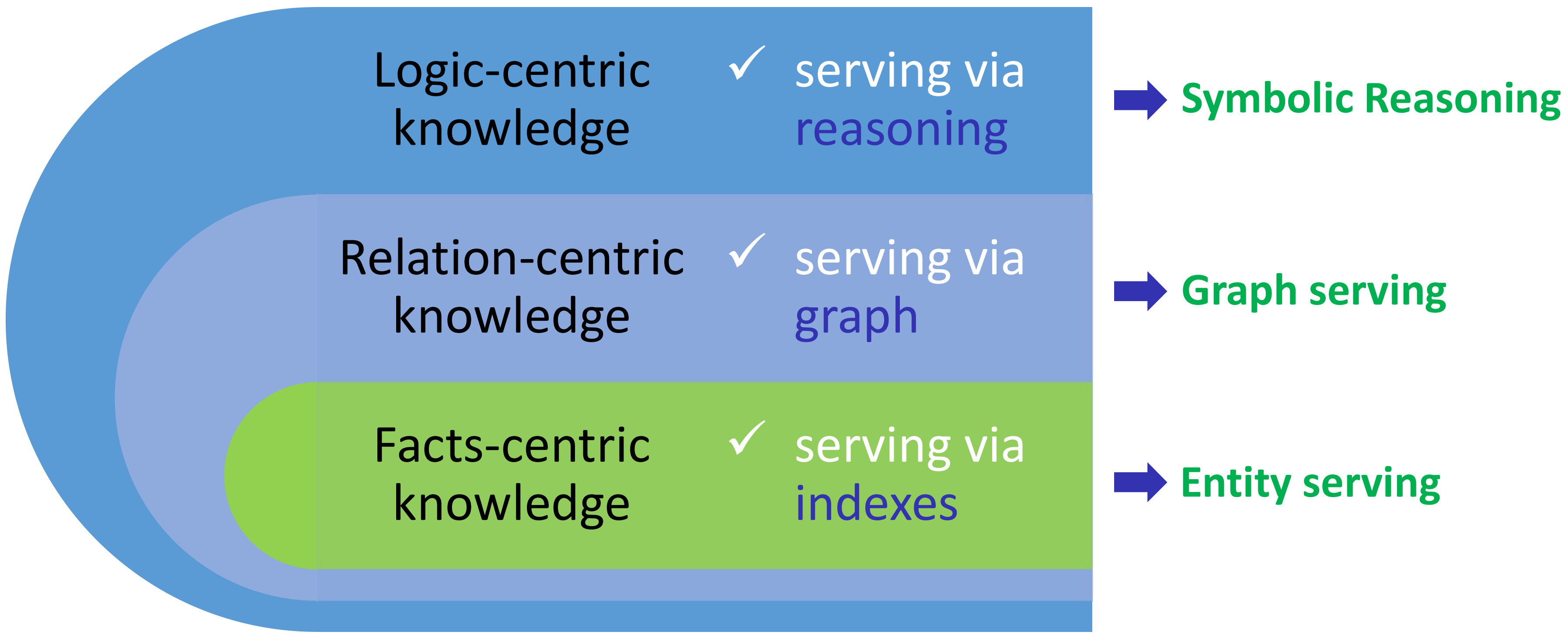


What is the huge amount of code for?

- Provides extremely fine-grained data access methods best matching the data

= Efficiency

Symbolic Reasoning

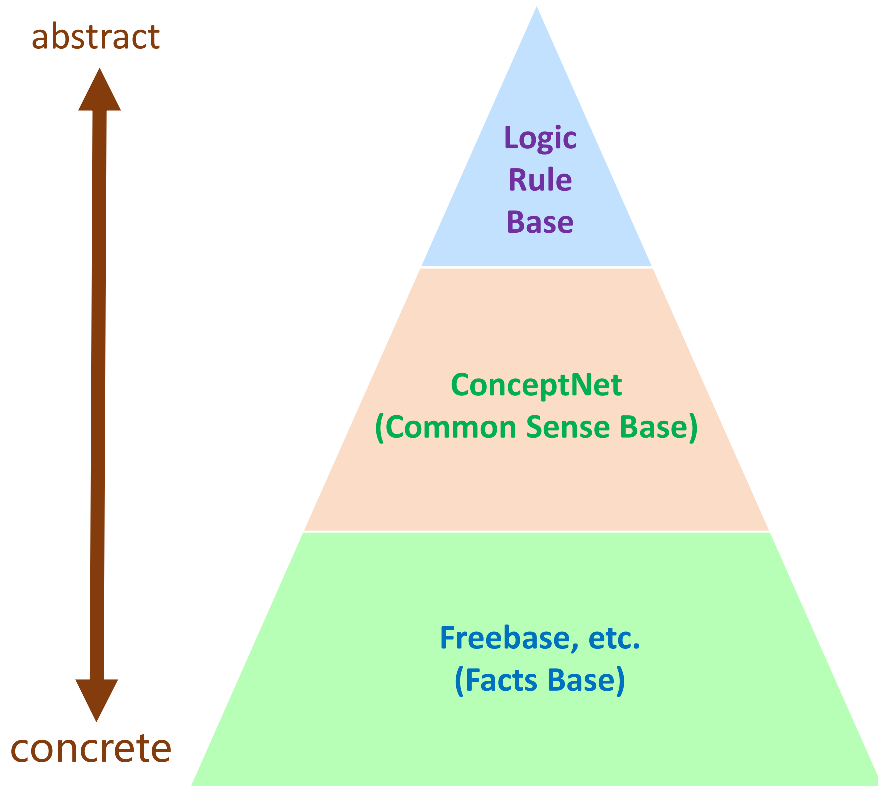


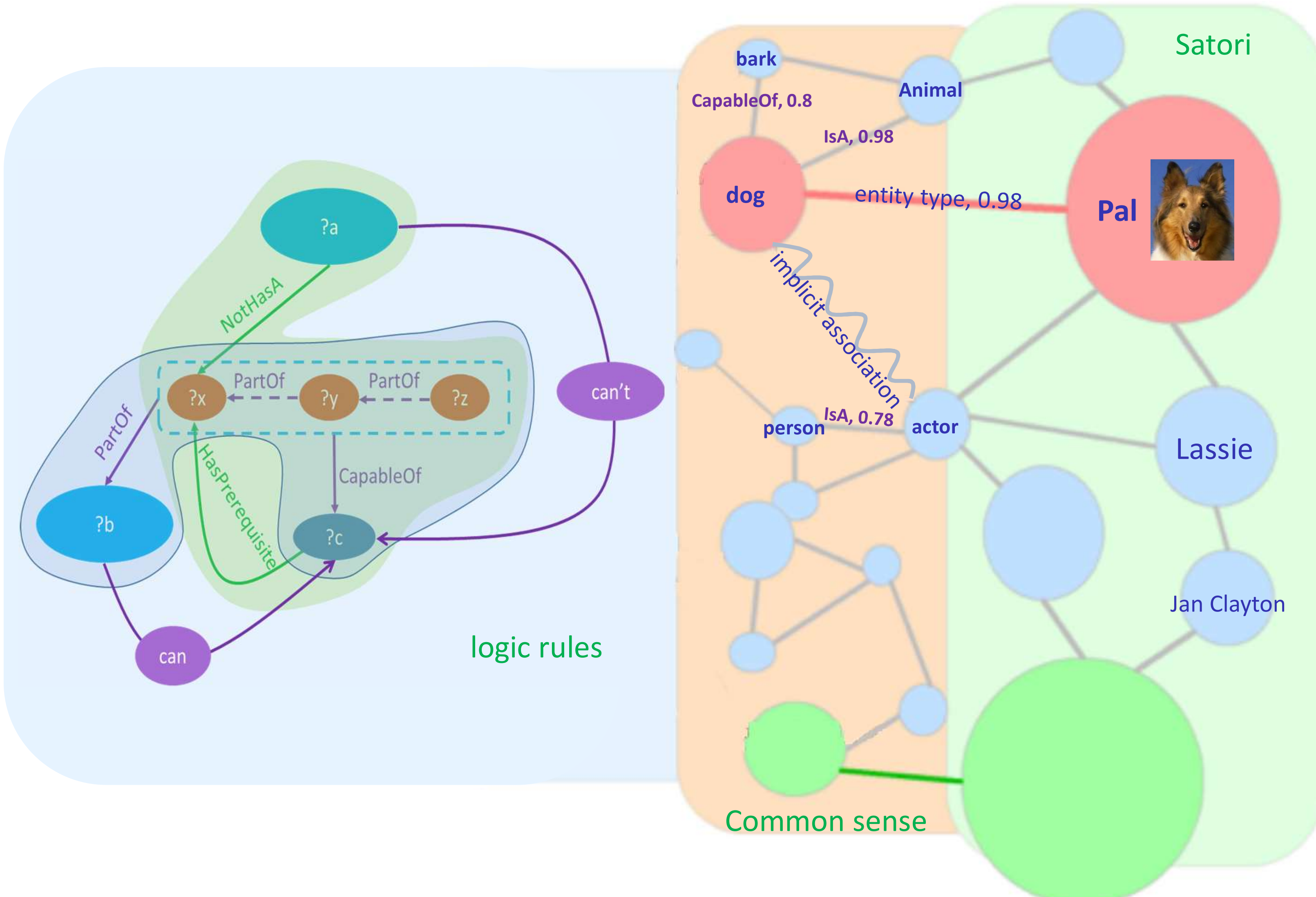
The evolution of knowledge representation

Why is a big knowledge graph not enough?

- Large knowledge graphs have billions of facts
- *However*, it doesn't provide much help in logic reasoning
    - The knowledge is not symbolized logic knowledge
    - Lack of reasoning rules allow machines to do reasoning automatically
    - More importantly, lack of common sense

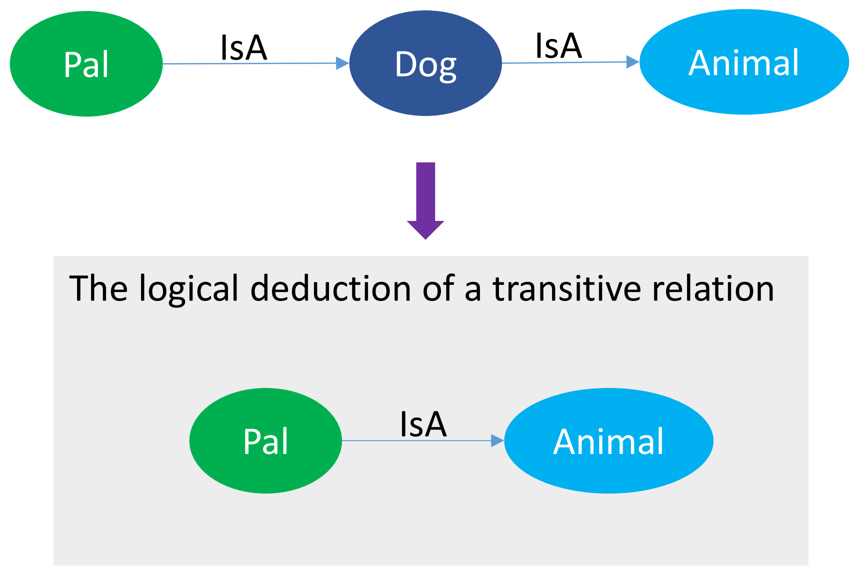
The pyramid of knowledge





?a
NotHasA
?x
PartOf
?y
PartOf
?z
PartOf
?b
HasPrerequisite
CapableOf
?c
can't
can
logic rules
bark
Animal
CapableOf, 0.8
IsA, 0.98
dog
entity type, 0.98
Pal
Satori
implicit association
person
IsA, 0.78
actor
Lassie
Jan Clayton
Common sense

Use graph transformation to do logic deduction



Graph transformation: whenever we see a graph G_a with a certain pattern p, replace it with a graph G_b.

Page: 1 of 13 130%

AI Magazine Volume 3 Number 4 (1982) (© AAAI)

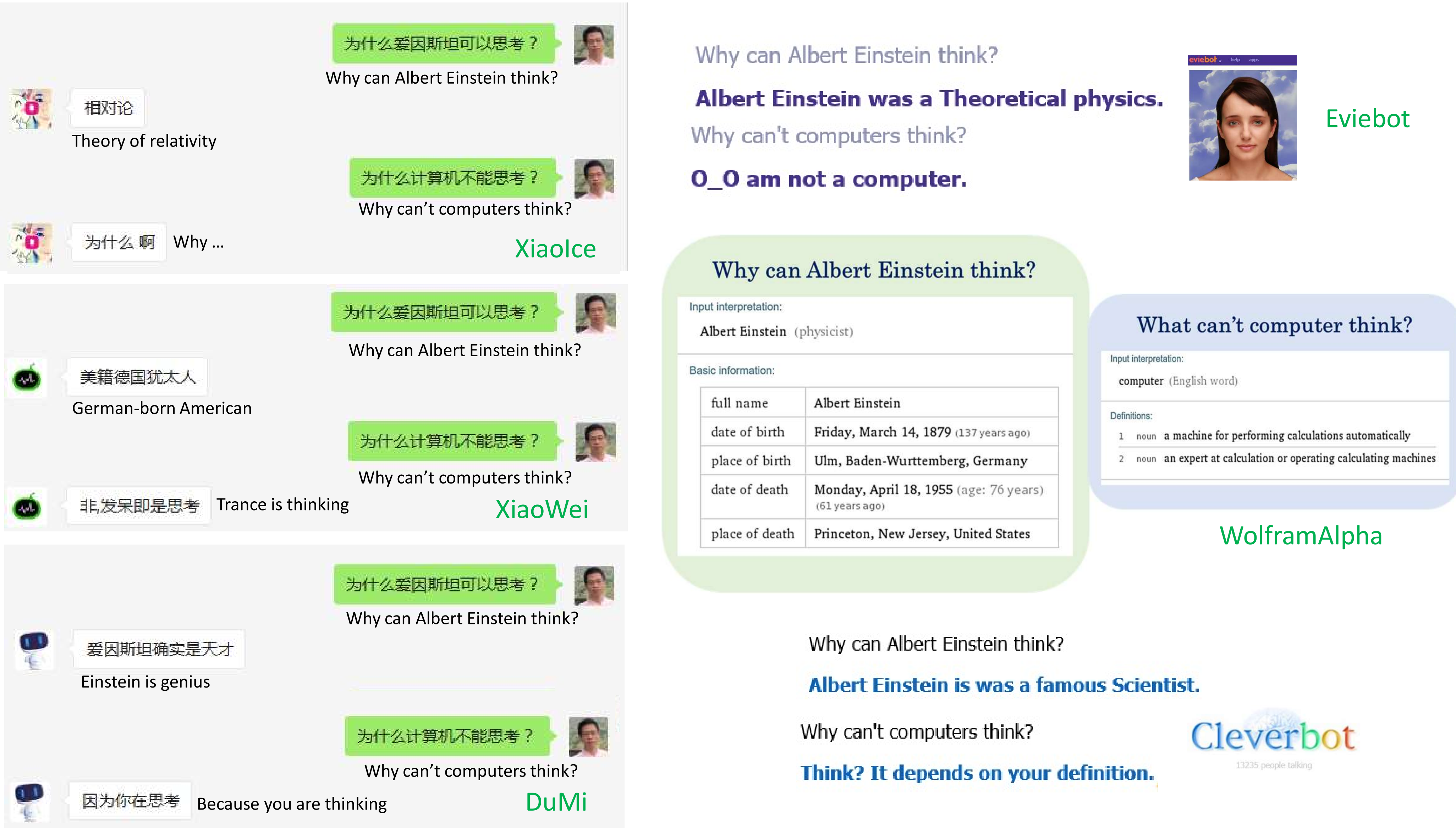
Instead of Minsky's 13-page answer, …

Why People Think Computers Can't

Marvin Minsky

MIT
Cambridge, Massachusetts

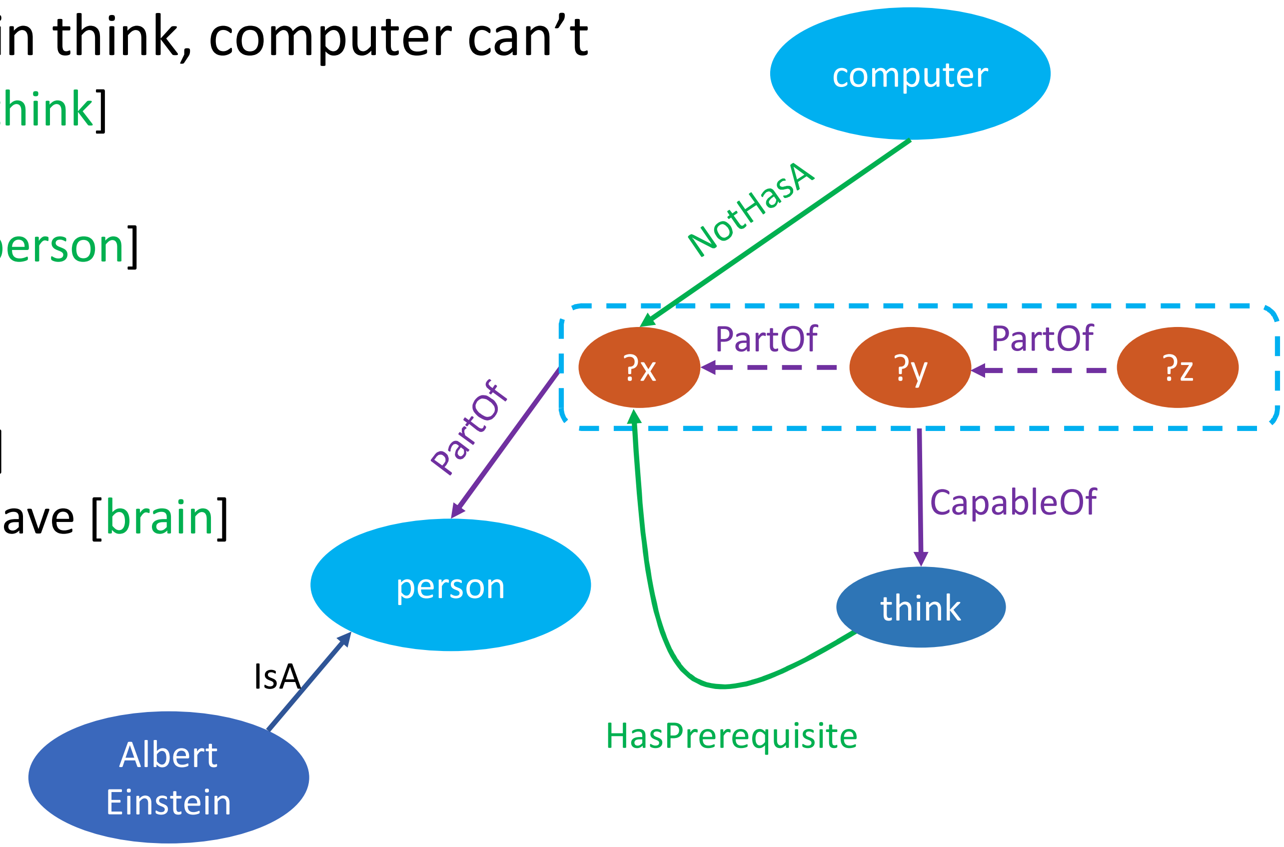
"Why" question


为什么爱因斯坦可以思考？
Why can Albert Einstein think?
相对论
Theory of relativity
为什么计算机不能思考？
Why can't computers think?
为什么 啊
Why ...
XiaoIce
为什么爱因斯坦可以思考？
Why can Albert Einstein think?
美籍德国犹太人
German-born American
为什么计算机不能思考？
Why can't computers think?
非发呆即是思考
Trance is thinking
XiaoWei
为什么爱因斯坦可以思考？
Why can Albert Einstein think?
爱因斯坦确实是天才
Einstein is genius
为什么计算机不能思考？
Why can't computers think?
因为你在思考
Because you are thinking
DuMi
Why can Albert Einstein think?
Albert Einstein was a Theoretical physics.
Why can't computers think?
O_O am not a computer.
Eviebot
Why can Albert Einstein think?
Input interpretation:
Albert Einstein (physicist)
Basic information:
full name
Albert Einstein
date of birth
Friday, March 14, 1879 (137 years ago)
place of birth
Ulm, Baden-Wurttemberg, Germany
date of death
Monday, April 18, 1955 (age: 76 years) (61 years ago)
place of death
Princeton, New Jersey, United States
What can't computer think?
Input interpretation:
computer (English word)
Definitions:
1 noun a machine for performing calculations automatically
2 noun an expert at calculation or operating calculating machines
WolframAlpha
Why can Albert Einstein think?
Albert Einstein is was a famous Scientist.
Why can't computers think?
Think? It depends on your definition.
Cleverbot

Our "shallow" yet reasonable answer

- Why can Albert Einstein think, computer can't
  - [brain] is Capable Of [think]
  - [person] have [brain]
  - [Albert Einstein] is a [person]

  - [think] requires [brain]
  - [computer] does not have [brain]



References

- Andrew Lumsdaine, Douglas Gregor, Bruce Hendrickson, Jonathan Berry. Challenges in parallel graph processing. Parallel Processing Letters 2007.
- Bin Shao, Haixun Wang, Yatao Li. Trinity: a distributed graph engine on a memory cloud. SIGMOD 2013.
- Luiz André Barroso, Urs Hoelzle. The datacenter as a computer: An introduction to the design of warehouse-scale machines. 2009.
- Grzegorz Malewicz, Matthew H. Austern, Aart J. C. Bik, James C. Dehnert, Ilan Horn, Naty Leiser, Grzegorz Czajkowski. Pregel: a system for large-scale graph processing. SIGMOD 2010.
- Aapo Kyrola, Guy Blelloch, Carlos Guestrin. GraphChi: Large-scale graph computation on just a PC. OSDI 2012.
- Zhao Sun, Hongzhi Wang, Haixun Wang, Bin Shao, and Jianzhong Li. Efficient subgraph matching on billion node graphs. PVLDB 2012.
- U Kang, Charalampos E. Tsourakakis, Christos Faloutsos. PEGASUS: A peta-scale graph mining system implementation and observations. ICDM 2009.
- Xiaohan Zhao, Alessandra Sala, Haitao Zheng, Ben Y. Zhao. Fast and Scalable Analysis of Massive Social Graphs. CoRR 2011.
- Zichao Qi, Yanghua Xiao, Bin Shao, Haixun Wang. Toward a Distance Oracle for Billion-Node Graphs. PVLDB 2014.

Thanks!

https://www.graphengine.io/
https://www.binshao.info/