Machine Learning on Big Data

SLIDES WILL BE AVAILABLE: HTTP://CS.MARKUSWEIMER.COM

Goals

- **Enable you to contribute from a Distributed Systems and Databases background to Machine Learning problems.**

- Show what Machine Learning on big data amounts to in practice
  - In terms of the process
  - In terms of the needed and emerging computational abstractions

- Discuss the systems available to that end

This tutorial is not …

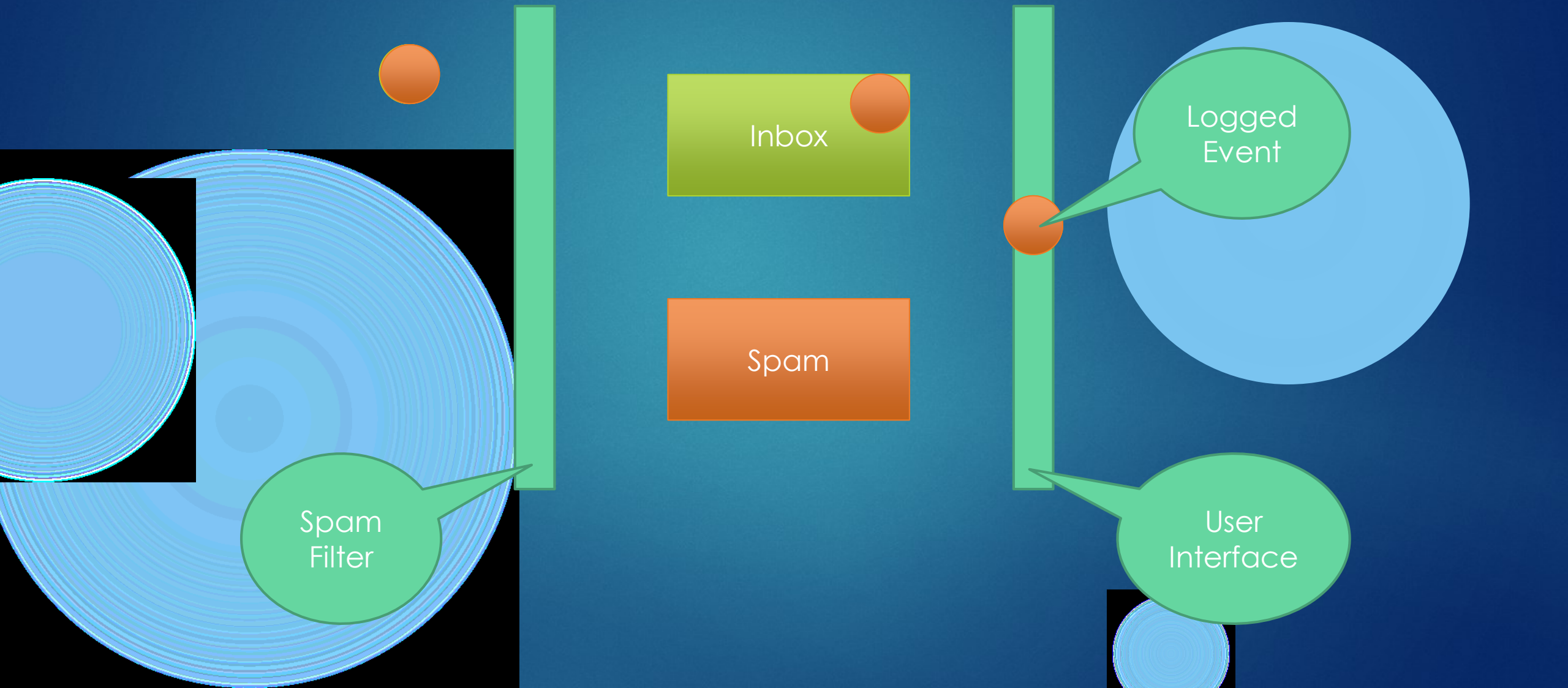
- A Machine Learning class
  - Plenty available on the web: Andrew Ng, Alex Smola, …
- A Distributed Systems class
  - You know more about that than at least one of the presenters
- About Machine Learning in MapReduce
  - Recommended: Vijay Narayanan and Milind Bhandarkar's KDD 2011 tutorial on that subject

A (very) quick intro to ML

“Machine Learning is Programming by Example”

- Used when:
  - Programming is hard (e.g. topic detection, bioinformatics)
  - Program changes all the time (recommender systems, antispam)

Example: Spam Filter



Example: Recommender Systems

	Item 1	Item 2	Item 3	Item 4	Item 5	…
Alice	★★★★★	★★☆☆☆	★★★☆☆	★★★★☆	★☆☆☆☆	
Bob	★★★★★	★★★★★	★★★★★	★★★★★	★★★★★	
Carol	★★★☆☆	★★☆☆☆	★★★★☆	★★★★★	★★★★☆	
Dave	★★★★★	★★☆☆☆	★★★☆☆		★☆☆☆☆	
Eve		★☆☆☆☆	★★★☆☆	★★★★☆	★★★☆☆	
…						

Example: Recommender Systems

	Item 1	Item 2	Item 3	Item 4	Item 5	…
Alice	★★★★★	★★☆☆☆	★★★☆☆	★★★★☆	★☆☆☆☆	
Bob	★★★★★	★★★★★	★★★★★	★★★★★	★★★★★	
Carol	★★★☆☆	★★☆☆☆	★★★★☆	★★★★★	★★★★☆	
Dave	★★★★★	★★☆☆☆	★★★☆☆	?	★☆☆☆☆	
Eve	?	★☆☆☆☆	★★★☆☆	★★★★☆	★★★☆☆	
…						

Example: Recommender Systems

	Item 1	Item 2	Item 3	Item 4	Item 5	…
Alice	★★★★★	★★☆☆☆	★★★☆☆	★★★★☆	★☆☆☆☆	
Bob	★★★★★	★★★★★	★★★★★	★★★★★	★★★★★	
Carol	★★★☆☆	★★☆☆☆	★★★★☆	★★★★★	★★★★☆	
Dave	★★★★★	★★☆☆☆	★★★☆☆	★★★★☆	★☆☆☆☆	
Eve	?	★☆☆☆☆	★★★☆☆	★★★★☆	★★★☆☆	
…						

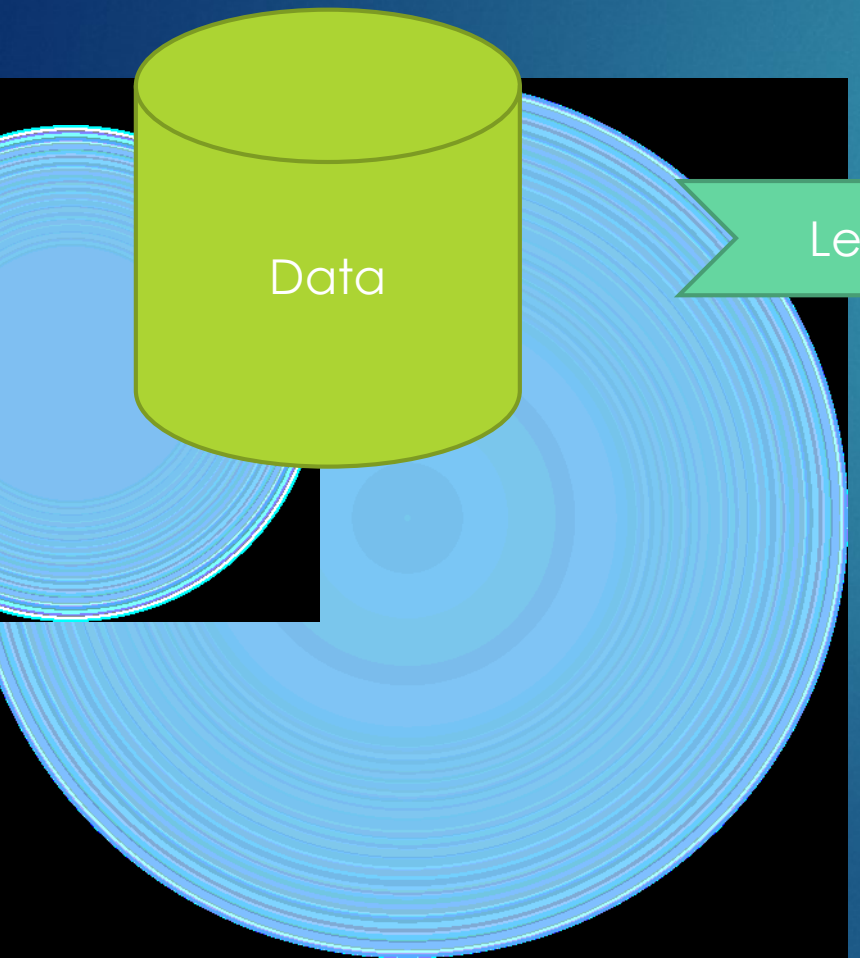
Example: Recommender Systems

	Item 1	Item 2	Item 3	Item 4	Item 5	…
Alice	★★★★★	★★☆☆☆	★★★☆☆	★★★★☆	★☆☆☆☆	
Bob	★★★★★	★★★★★	★★★★★	★★★★★	★★★★★	
Carol	★★★☆☆	★★☆☆☆	★★★★☆	★★★★★	★★★★☆	
Dave	★★★★★	★★☆☆☆	★★★☆☆	★★★★☆	★☆☆☆☆	
Eve	?	★☆☆☆☆	★★★☆☆	★★★★☆	★★★☆☆	
…						

Example: Recommender Systems

	Item 1	Item 2	Item 3	Item 4	Item 5	…
Alice	★★★★★	★★☆☆☆	★★★☆☆	★★★★☆	★☆☆☆☆	
Bob	★★★★★	★★★★★	★★★★★	★★★★★	★★★★★	
Carol	★★★☆☆	★★☆☆☆	★★★★☆	★★★★★	★★★★☆	
Dave	★★★★★	★★☆☆☆	★★★☆☆	★★★★☆	★☆☆☆☆	
Eve	★★☆☆☆	★☆☆☆☆	★★★☆☆	★★★★☆	★★★☆☆	
…						

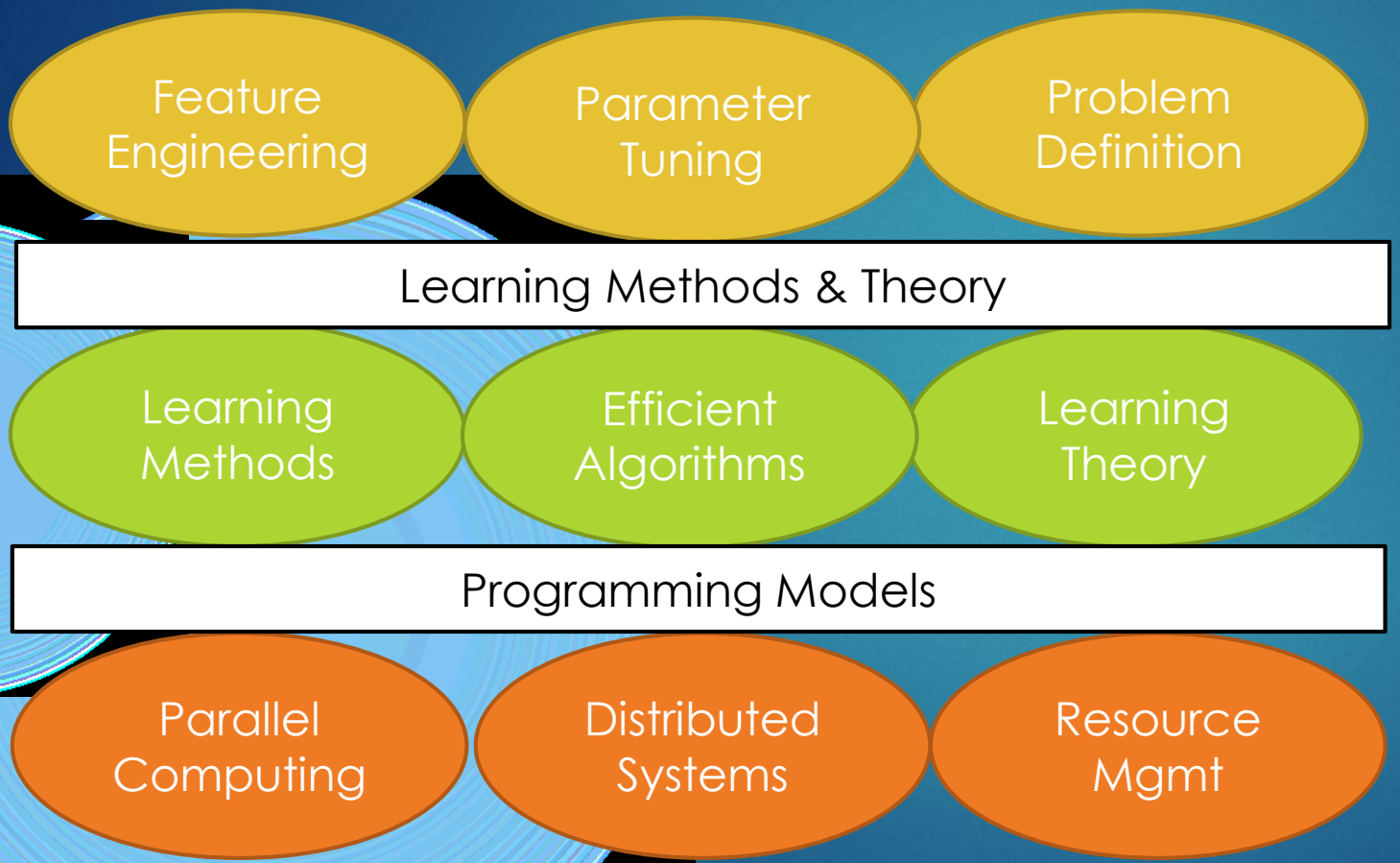
Machine Learning



Supervised
- Classification
- Regression
- Recommender

Unsupervised
- Clustering
- Dimensionality reduction
- Topic modeling

Influences of ML Success



Domain Expert
Data Scientist

Machine Learning
Researcher

You

Today

On Model Power and Data

More Data

- Improves generalization
- Facilitates more powerful models
- Is often cheap
- Increase I/O Cost

More Powerful Models

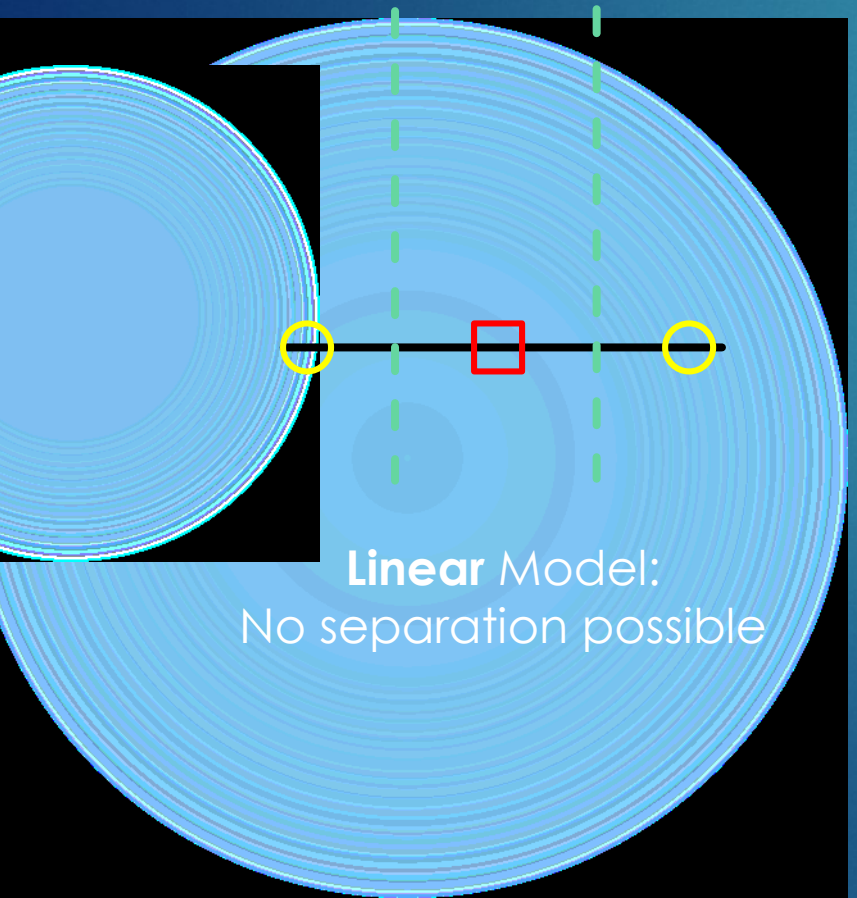
- Improve model fidelity and Generalization
- Require more training data
- Increase computational cost

Observation:
Today, we almost exclusively focus on simple models using lots of data.

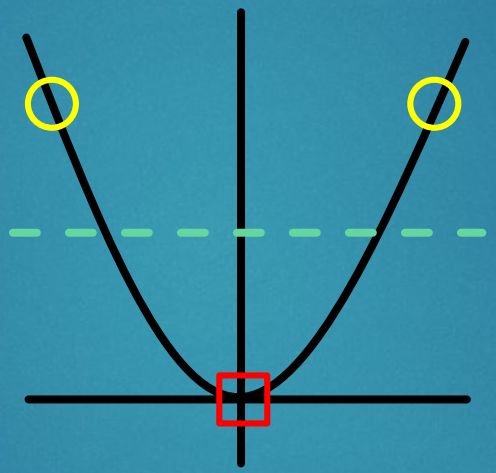
Conjecture:
That is because it is too damn hard to write complex, *distributed* learning algorithms.

→ **Machine Learning needs your help**

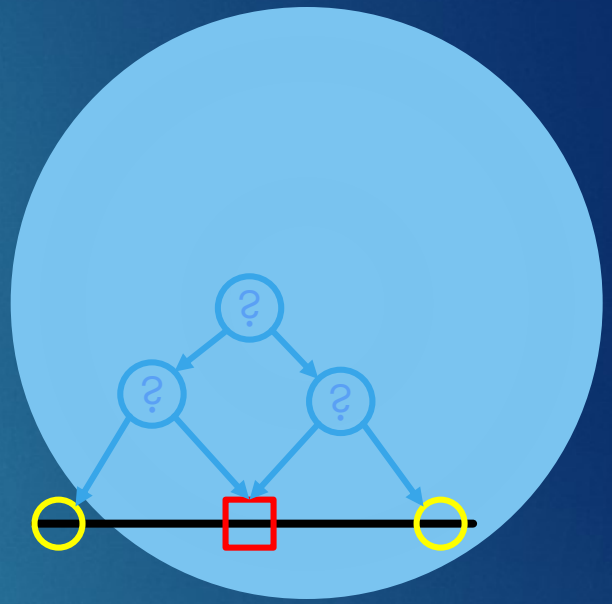
What does model power mean?



Linear Model:
No separation possible



Polynomial model:
Linear separation possible



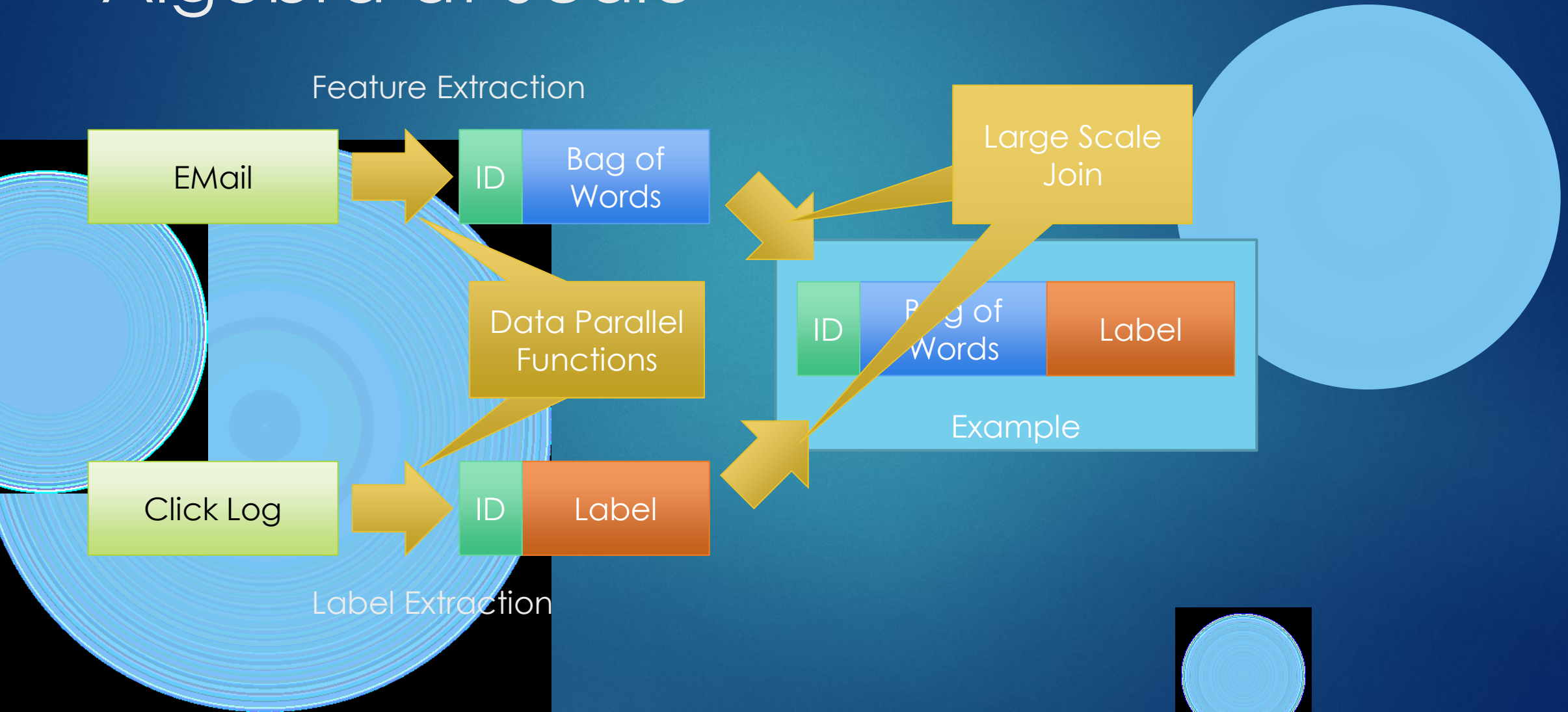
Tree Model:
Separation possible

The Machine Learning Workflow

Machine Learning Workflow

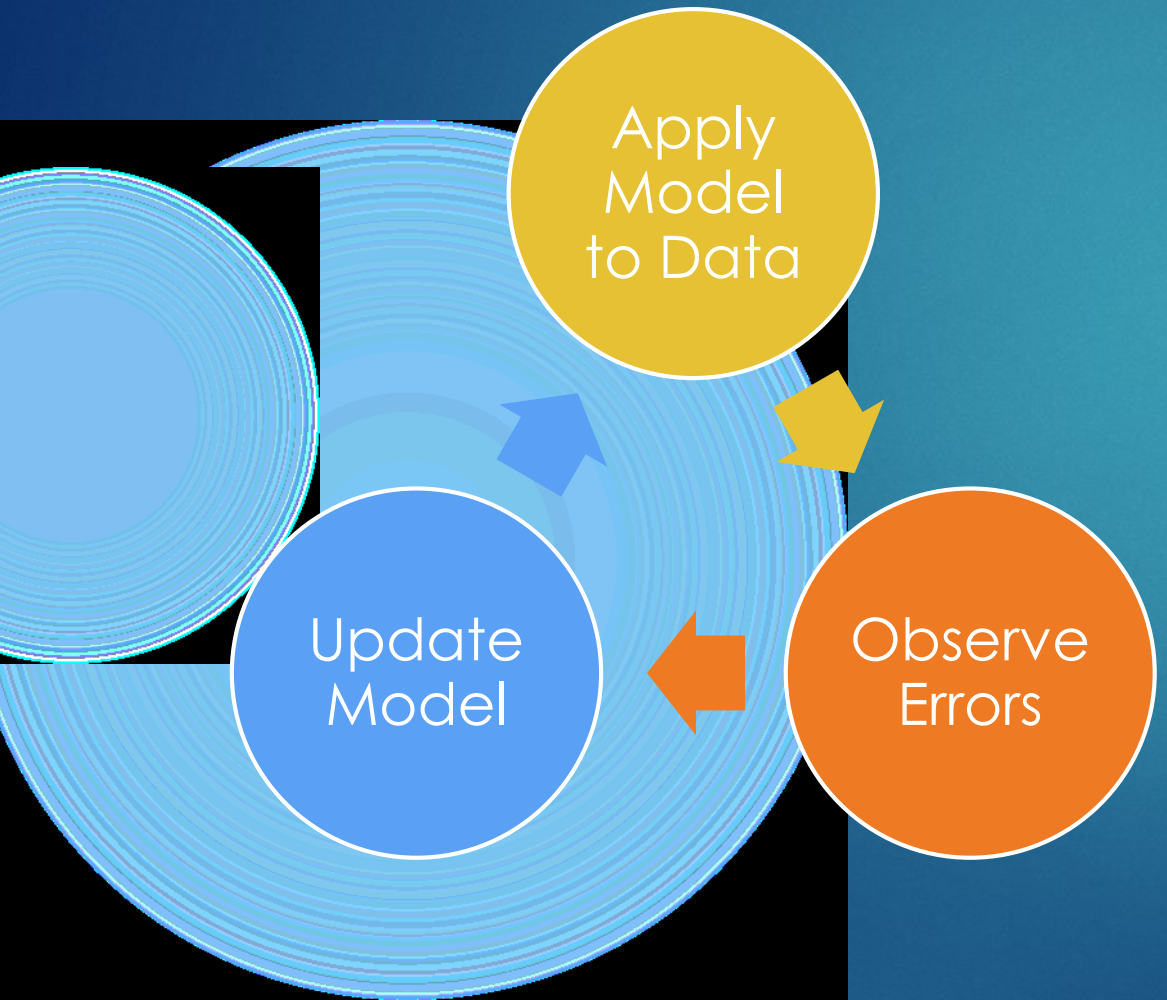
- Step I: Example Formation
  - Feature and Label Extraction
- Step II: Modeling
- Step III: Evaluation (and eventually Deployment)

Example Formation: Relational Algebra at Scale



Modeling (30,000ft)

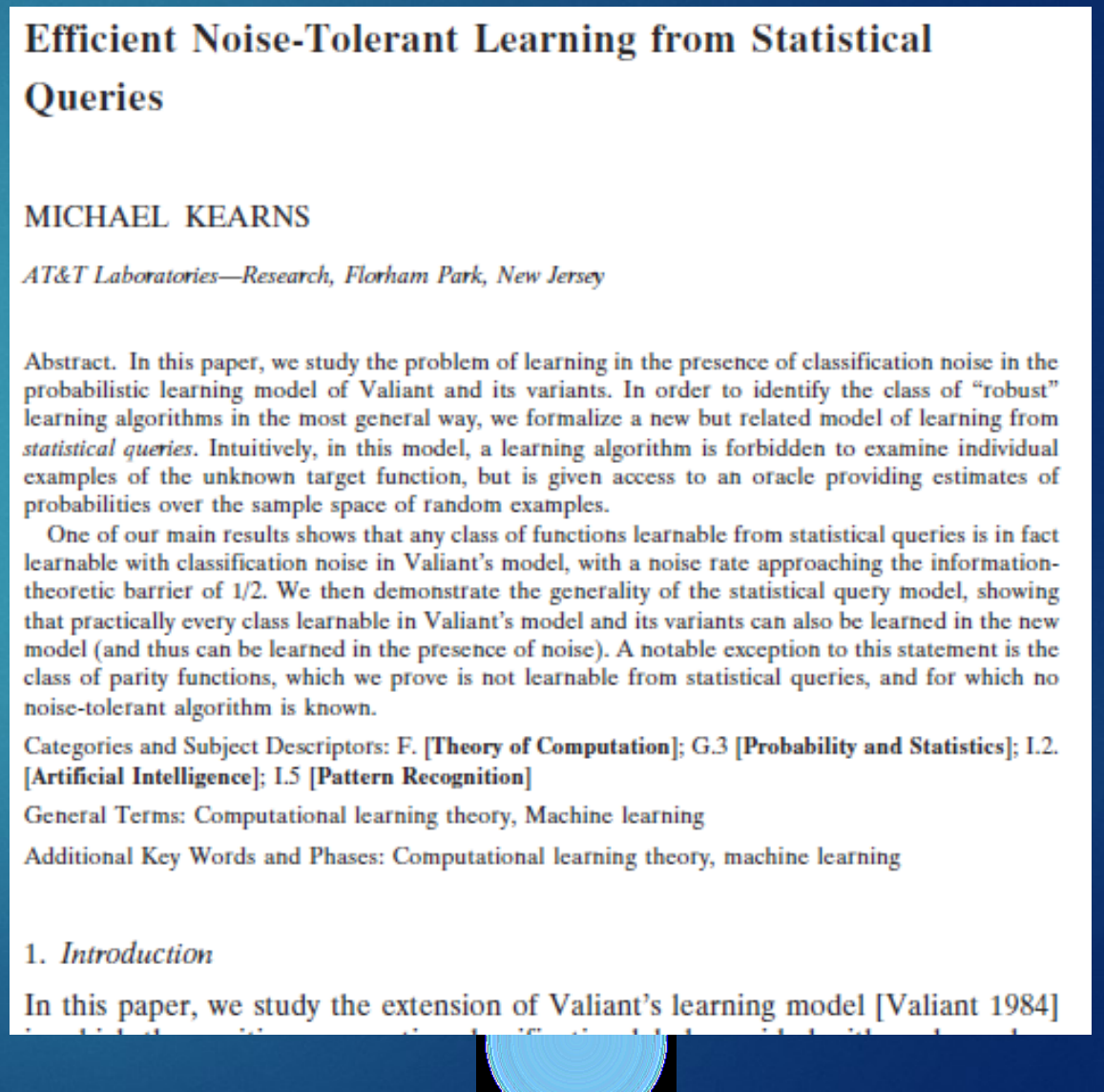
- Learning is Iterative



- Computational models are **emerging**:
  - **Statistical Query Model:** Algorithm operates on statistics of the dataset
  - **Graphical Models:** Heavy message passing, possibly asynchronous.
  - **Many more:** Custom solutions

The Statistical Query Model

- Learning algorithm can access the learning problem only through a statistical query oracle.
- The statistical query oracle returns an estimate of the expectation of a function f(x,y) (averaged over the data distribution).



Efficient Noise-Tolerant Learning from Statistical Queries

MICHAEL KEARNS

AT&T Laboratories—Research, Florham Park, New Jersey

Abstract. In this paper, we study the problem of learning in the presence of classification noise in the probabilistic learning model of Valiant and its variants. In order to identify the class of "robust" learning algorithms in the most general way, we formalize a new but related model of learning from *statistical queries*. Intuitively, in this model, a learning algorithm is forbidden to examine individual examples of the unknown target function, but is given access to an oracle providing estimates of probabilities over the sample space of random examples.

One of our main results shows that any class of functions learnable from statistical queries is in fact learnable with classification noise in Valiant's model, with a noise rate approaching the information-theoretic barrier of 1/2. We then demonstrate the generality of the statistical query model, showing that practically every class learnable in Valiant's model and its variants can also be learned in the new model (and thus can be learned in the presence of noise). A notable exception to this statement is the class of parity functions, which we prove is not learnable from statistical queries, and for which no noise-tolerant algorithm is known.

Categories and Subject Descriptors: F. **[Theory of Computation]**; G.3 **[Probability and Statistics]**; I.2. **[Artificial Intelligence]**; I.5 **[Pattern Recognition]**

General Terms: Computational learning theory, Machine learning

Additional Key Words and Phases: Computational learning theory, machine learning

1. *Introduction*

In this paper, we study the extension of Valiant's learning model [Valiant 1984]

Example: Learn a Regression Model

- **Given:** Dataset *X* of Examples, each consisting of:
  - a Feature Vector x (e.g. words in a document) and
  - a Label y (e.g. the fraction of people that clicked it).
- **Desired:** A predictor
  - "Find a function *f(x)* such that *f(x)* ≈ *y* for unseen data"
  - More precisely, find:

$$\hat{f} = \underset{f}{\operatorname{argmin}} \sum_{(x,y)\epsilon\, X} (f(x) - y)^2$$

The model: Linear Models

- The set of functions f(x) is infinite → We can't find a solution
- Approach: Restrict the set of functions to **linear** functions
  - We choose linear functions parameterized by the weight vector w :

$$f(x) \coloneqq f_w(x) \coloneqq \langle w, x \rangle = \sum_i w_i x_i$$

  - Hence, we can now search for $\hat{w}$:

$$\hat{w} = \underset{w}{\operatorname{argmin}} \sum_{(x,y)\epsilon\, X} (f_w(x) - y)^2 = \sum_{(x,y)\epsilon\, X} (\langle w, x \rangle - y)^2$$

Objective Function

The learning algorithm: Batch Gradient Descent

- The objective function is convex and differentiable in w
- Algorithm (simplified):

Start with a random w_0

Until convergence:

Compute the gradient

$$\partial_w = \sum_{(x,y)\epsilon\, X} 2\,(\langle w, x\rangle - y)$$

Statistical Query

Apply the gradient to the model

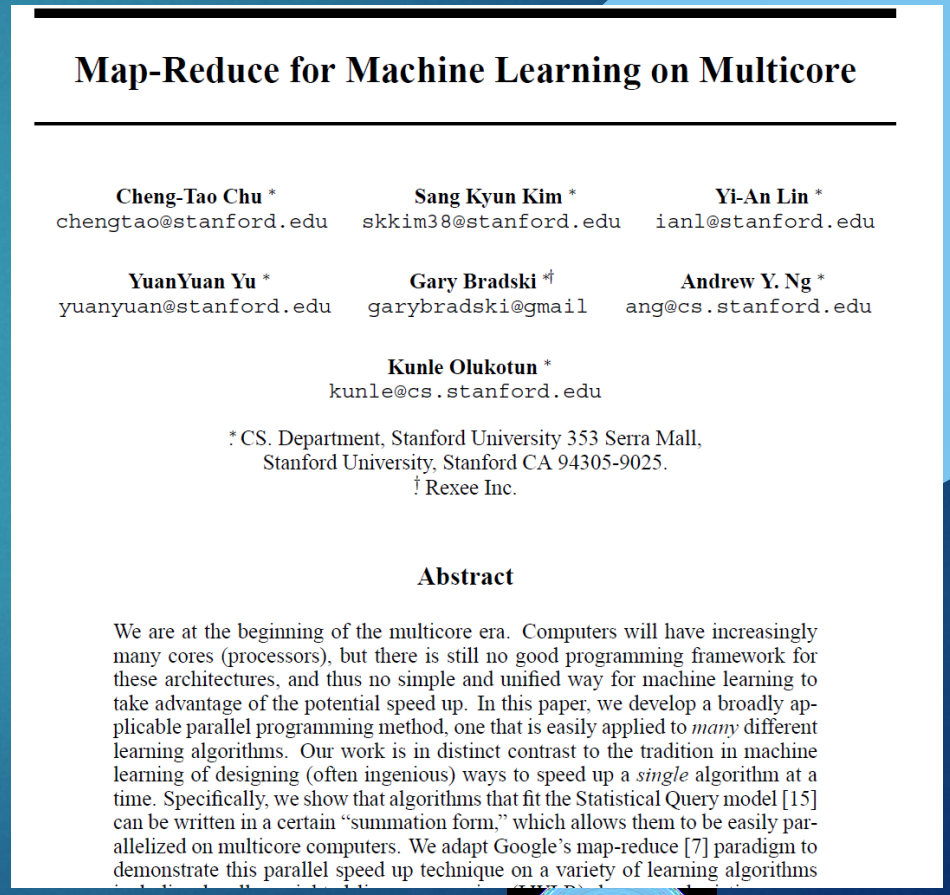
$$w_{t+1} = w_t - \partial_w$$

Example: Summary

1. Define Problem: **Regression**
2. Choose Model Class: **Linear Models**
3. Choose Learning Algorithm: **Gradient Descent**
4. Express Learning Algorithm: **Statistical Queries**
5. Execute Algorithm at scale: **Stay tuned …**

SQM and MapReduce

- Rephrase oracle in summation form.
- **Map:** Calculate function estimates over sub-groups of data.
- **Reduce:** Aggregate the function estimates from various sub-groups.
- Each statistical query turns into one MapReduce job



Map-Reduce for Machine Learning on Multicore

Cheng-Tao Chu * chengtao@stanford.edu
Sang Kyun Kim * skkim38@stanford.edu
Yi-An Lin * ianl@stanford.edu
YuanYuan Yu * yuanyuan@stanford.edu
Gary Bradski *† garybradski@gmail
Andrew Y. Ng * ang@cs.stanford.edu
Kunle Olukotun * kunle@cs.stanford.edu

* CS. Department, Stanford University 353 Serra Mall, Stanford University, Stanford CA 94305-9025.
† Rexee Inc.

Abstract

We are at the beginning of the multicore era. Computers will have increasingly many cores (processors), but there is still no good programming framework for these architectures, and thus no simple and unified way for machine learning to take advantage of the potential speed up. In this paper, we develop a broadly applicable parallel programming method, one that is easily applied to *many* different learning algorithms. Our work is in distinct contrast to the tradition in machine learning of designing (often ingenious) ways to speed up a *single* algorithm at a time. Specifically, we show that algorithms that fit the Statistical Query model [15] can be written in a certain "summation form," which allows them to be easily parallelized on multicore computers. We adapt Google's map-reduce [7] paradigm to demonstrate this parallel speed up technique on a variety of learning algorithms

BGD in Iterative Map Reduce Update

- Algorithm (simplified):

Start with a random w_0

Until convergence: **Iterative**

Compute the gradient

Reduce $\partial_w = \sum_{(x,y)\epsilon X} 2\left(\langle w, x\rangle - y\right)$ **Map**

Apply the gradient to the model

$w_{t+1} = w_t - \partial_w$

Update

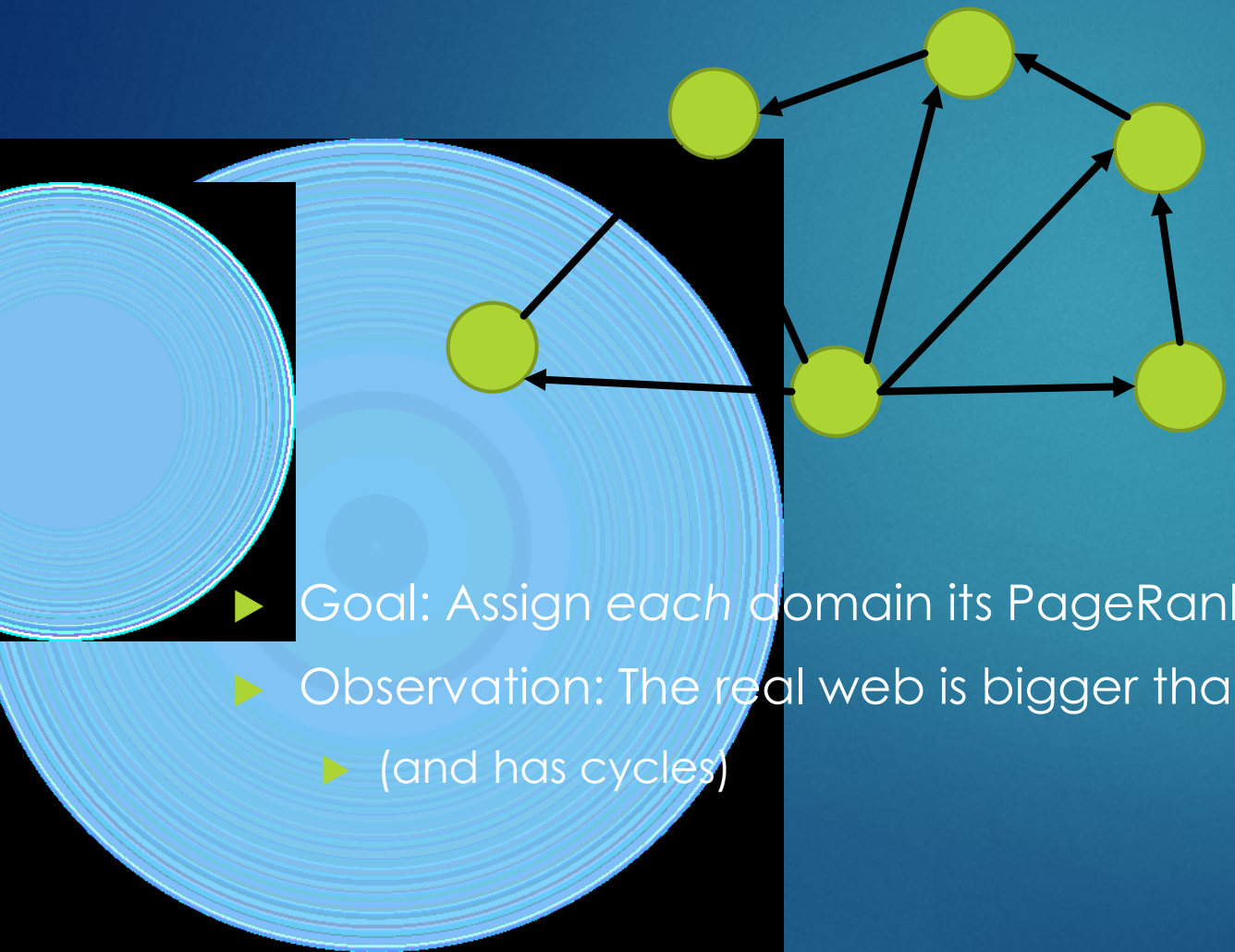
Further Reading

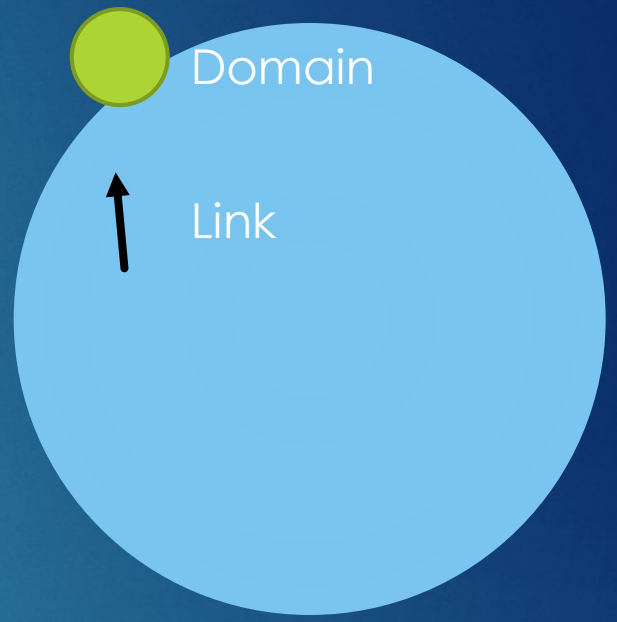
- Paper: Efficient Noise-Tolerant Learning from Statistical Queries
  - http://www.cs.iastate.edu/~honavar/noise-pac.pdf
- KDD 2011 Tutorial on Machine Learning in MapReduce
  - http://www.slideshare.net/hadoop/modeling-with-hadoop-kdd2011
- Paper: Map-Reduce for Machine Learning on Multicore
  - http://cs.stanford.edu/people/ang/papers/nips06-mapreducemulticore.pdf

Beyond SQM

A GRAPH ANALYSIS PERSPECTIVE

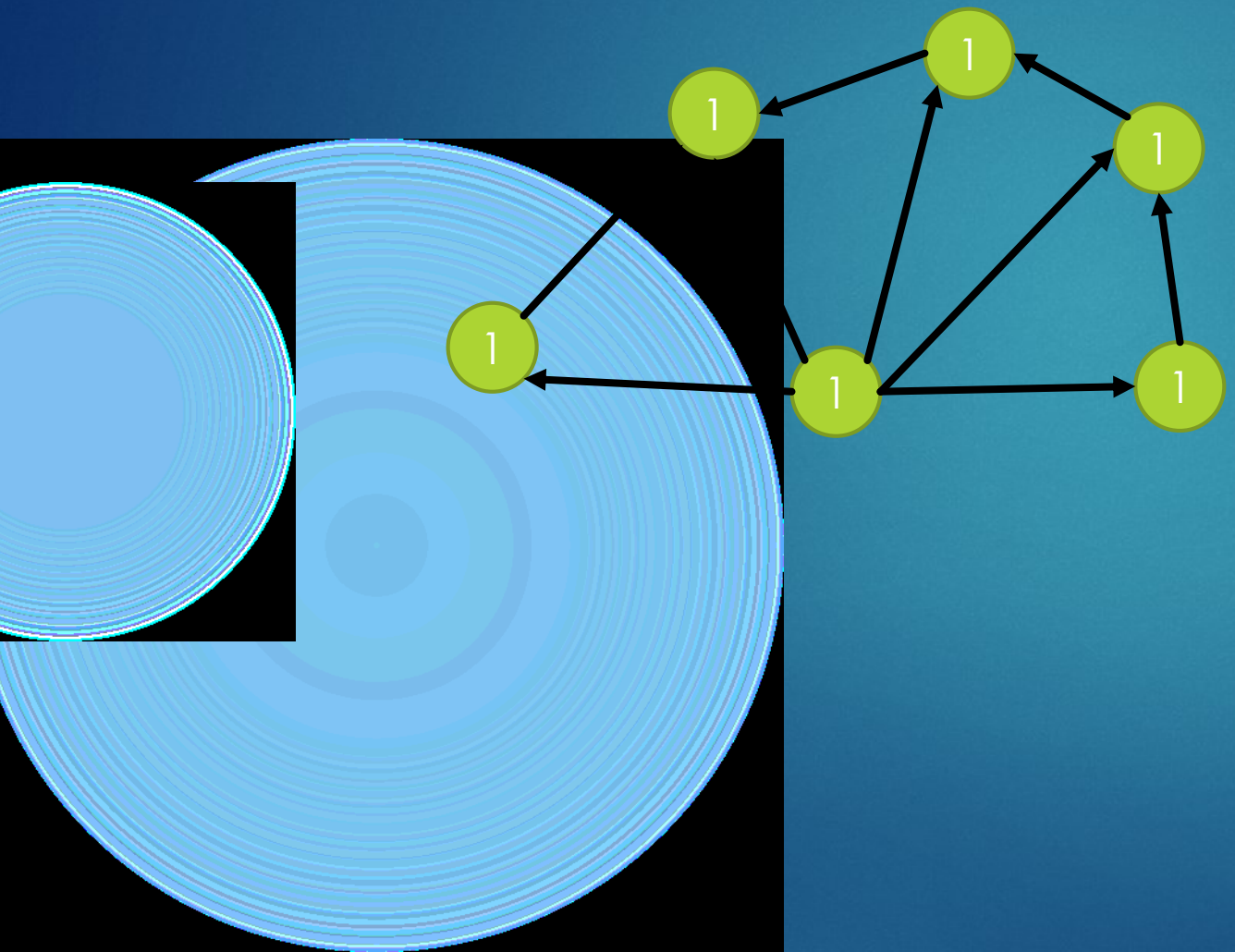
Example: PageRank





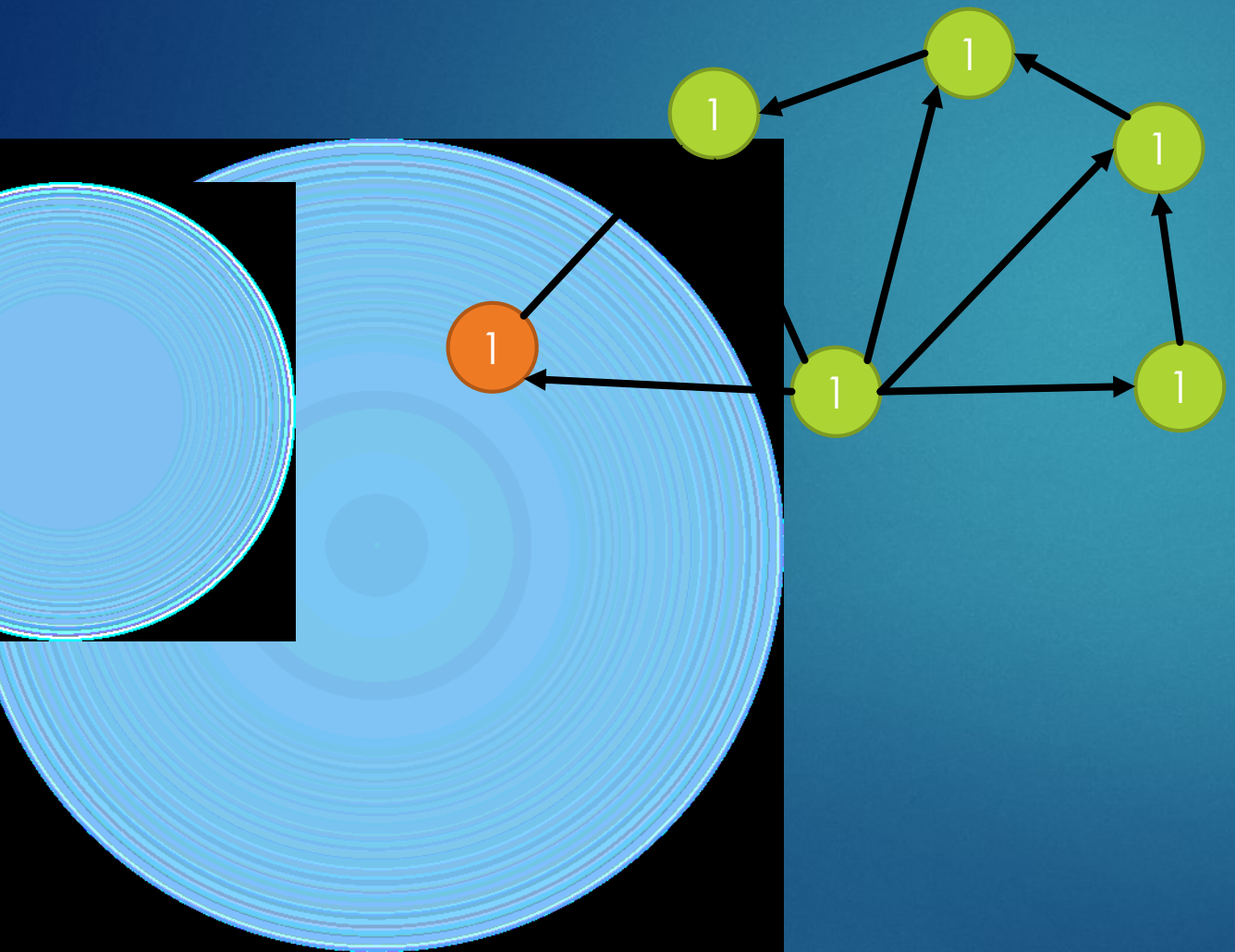
- Goal: Assign *each* domain its PageRank
- Observation: The real web is bigger than RAM
  - (and has cycles)

Example: PageRank



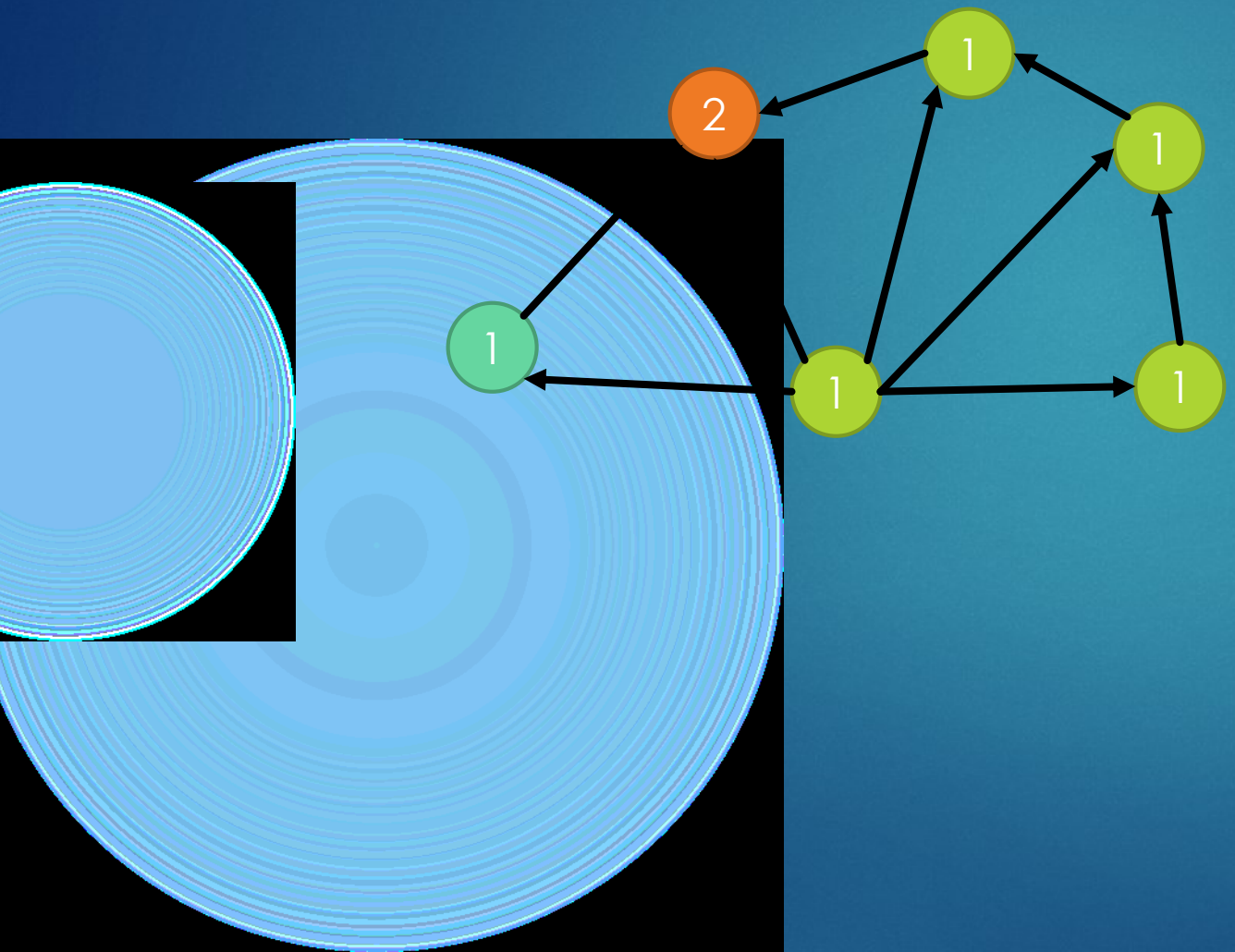
Update

Example: PageRank



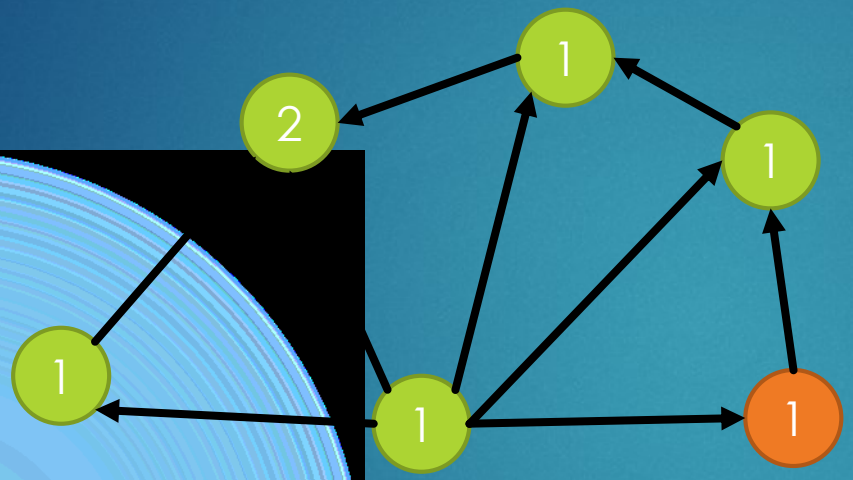
Update

Example: PageRank

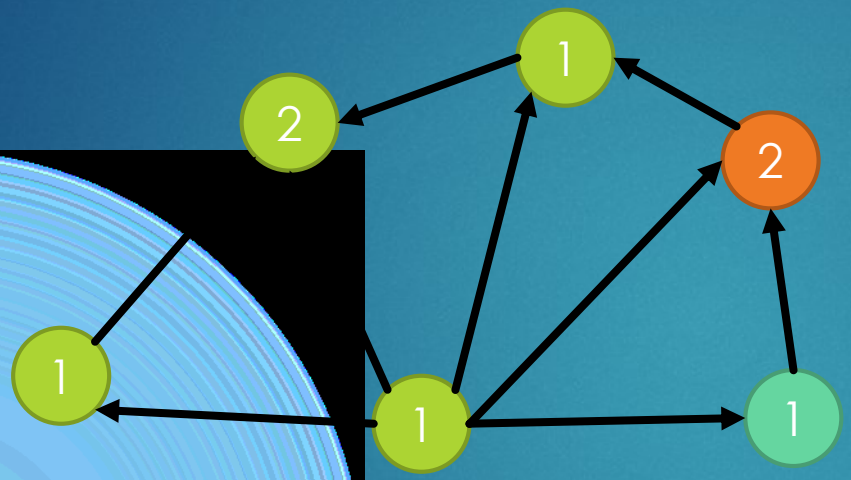


No more Outlinks, we are done!

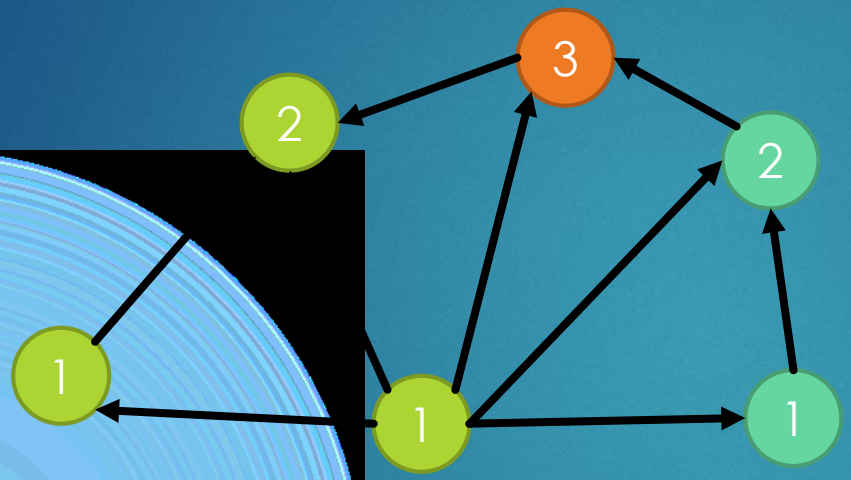
Example: PageRank



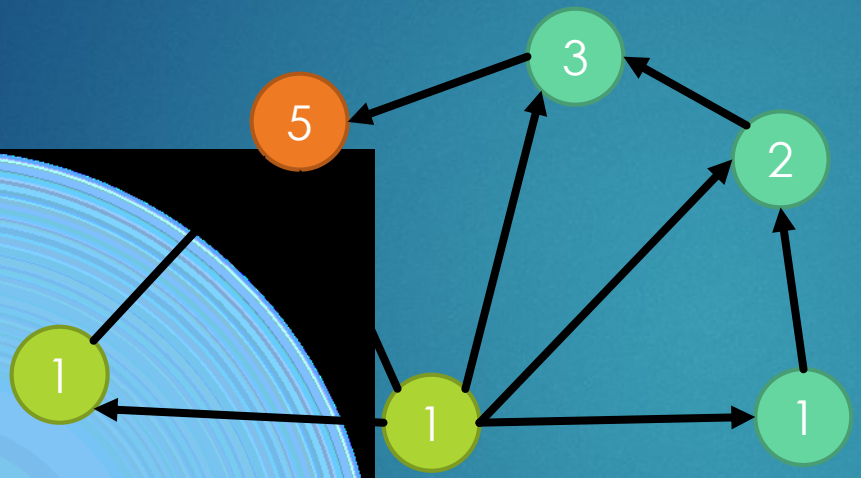
Example: PageRank



Example: PageRank

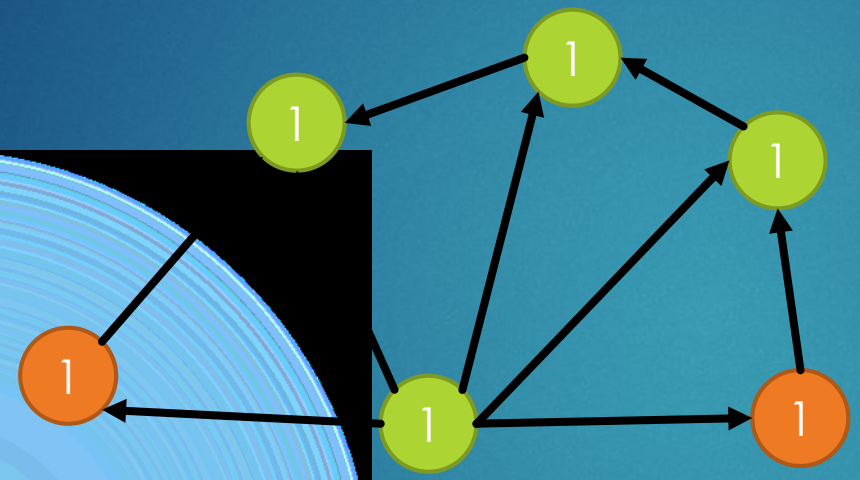


Example: PageRank

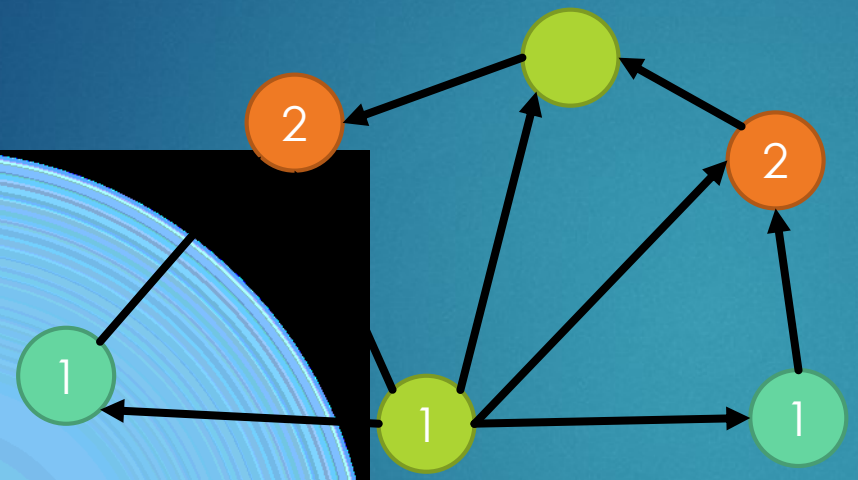


No more Outlinks, we are done!

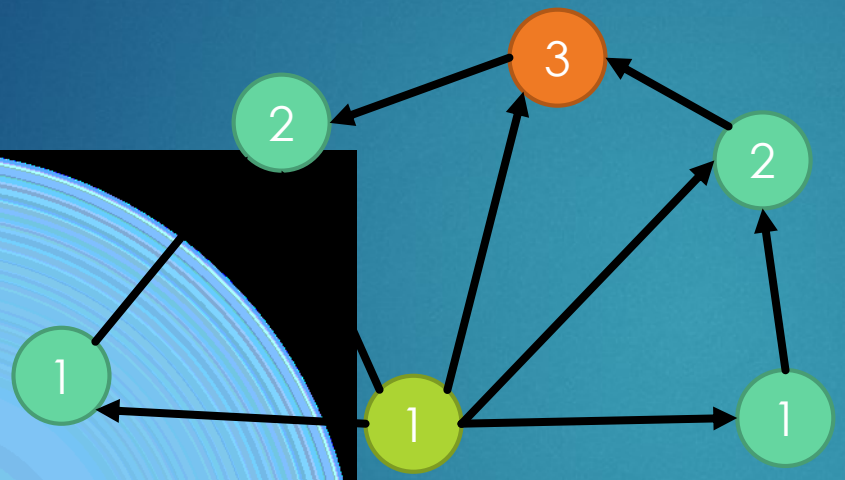
Example: PageRank in Parallel



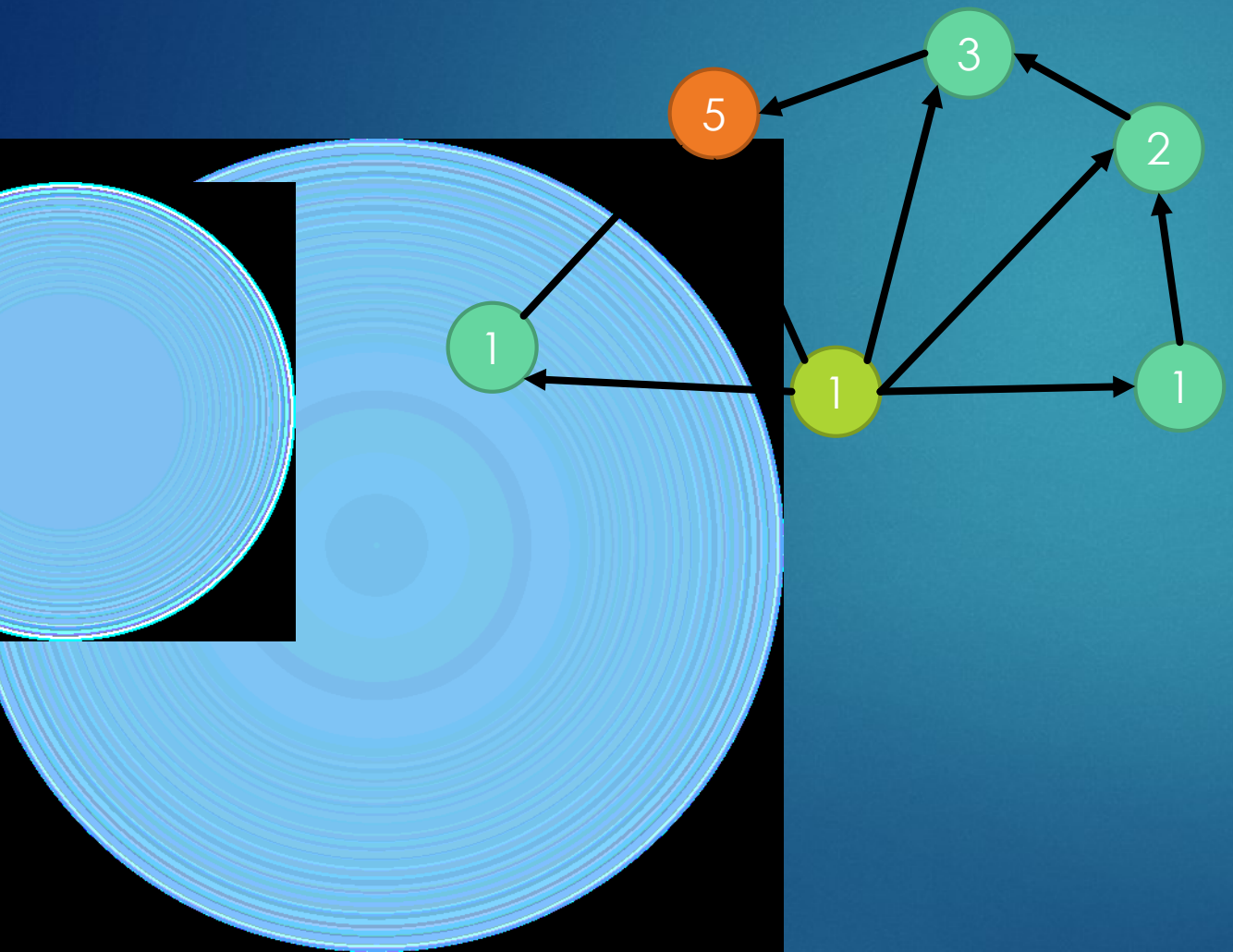
Example: PageRank in Parallel



Example: PageRank in Parallel



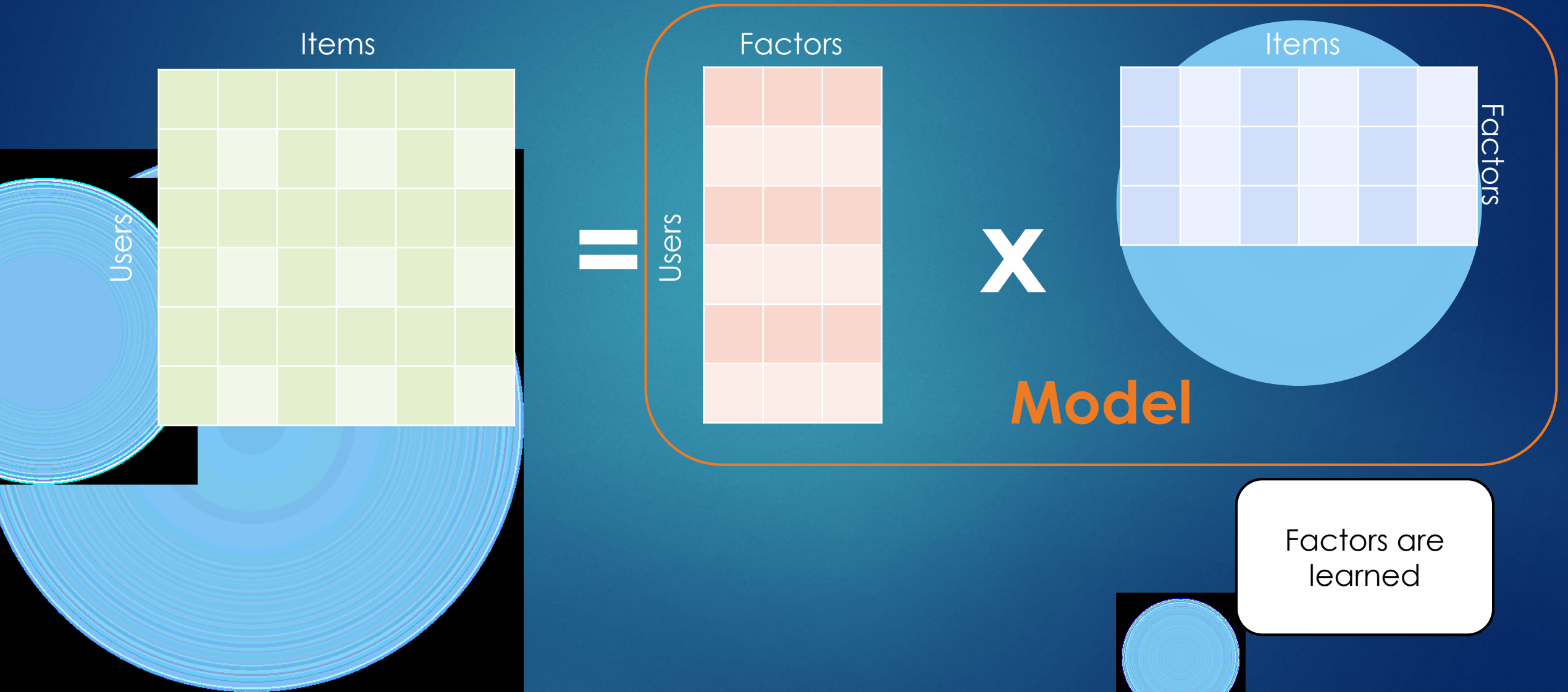
Example: PageRank



No more Outlinks, we are done!

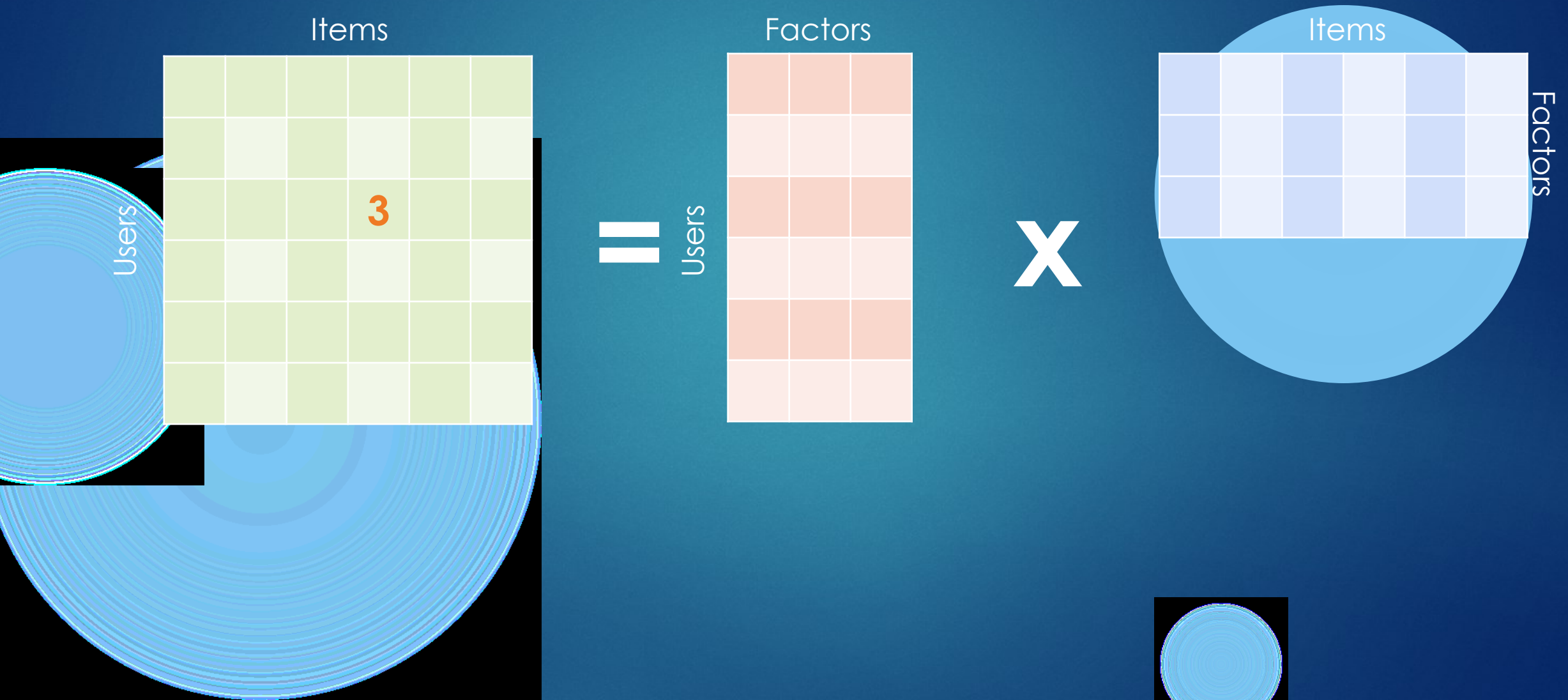
Skip to Summary

Example: Matrix Factorization

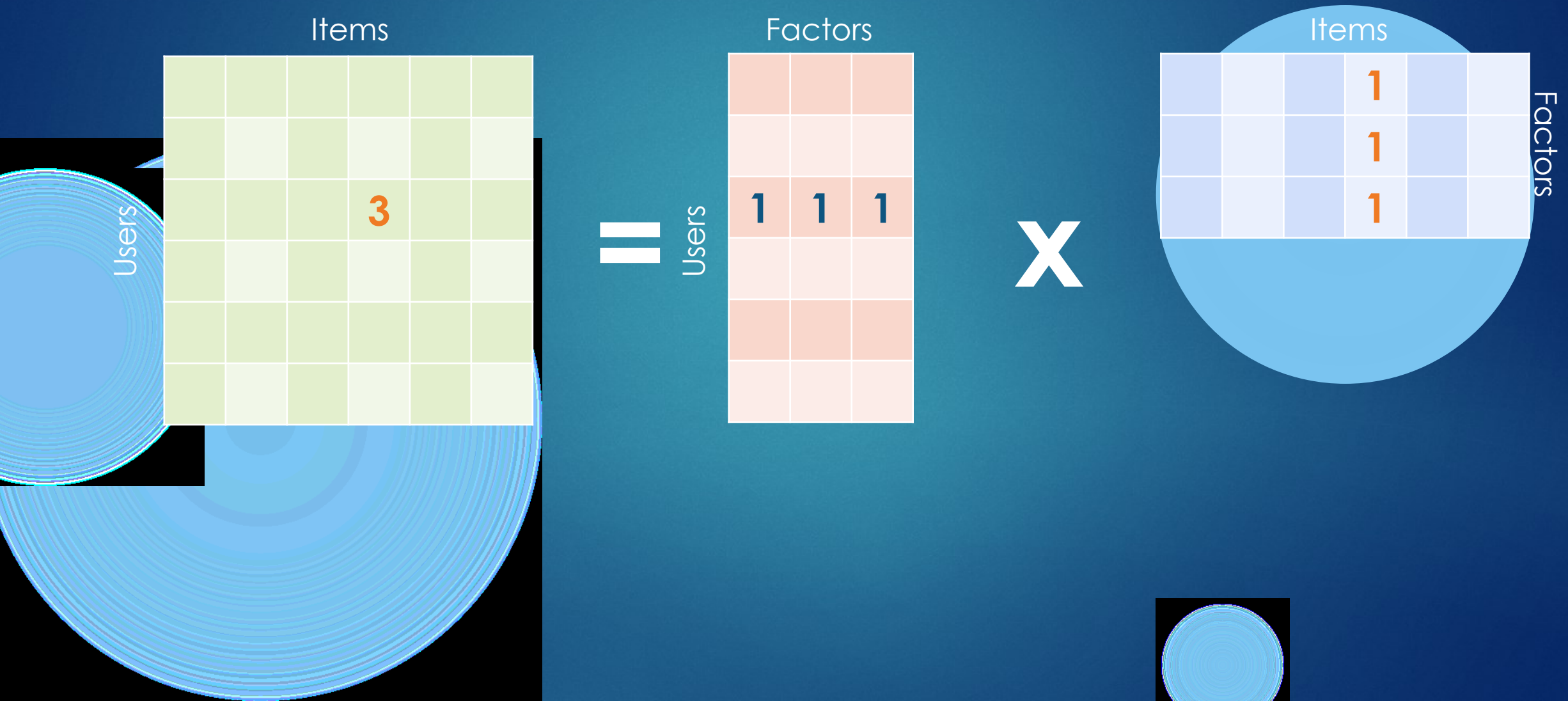


Factors are learned

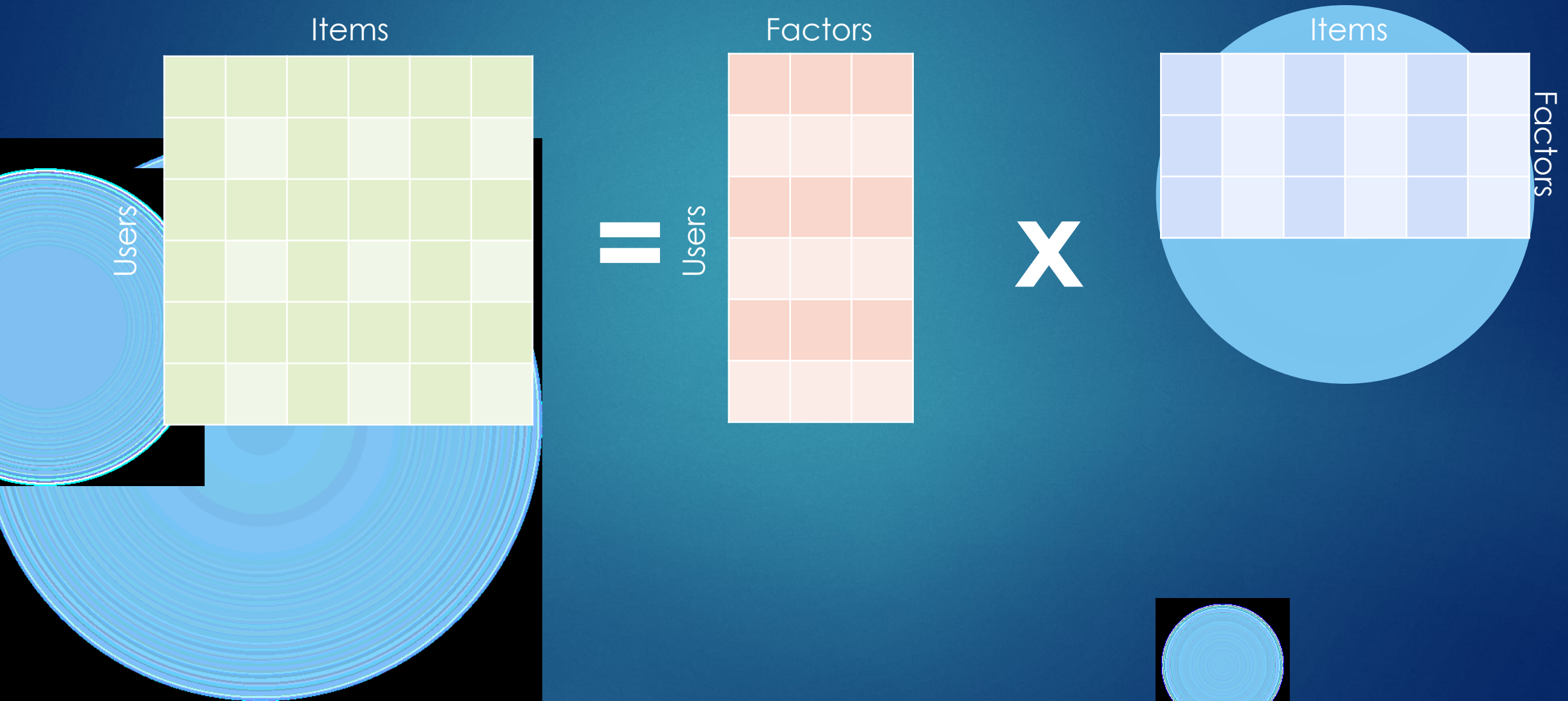
Example: Matrix Factorization



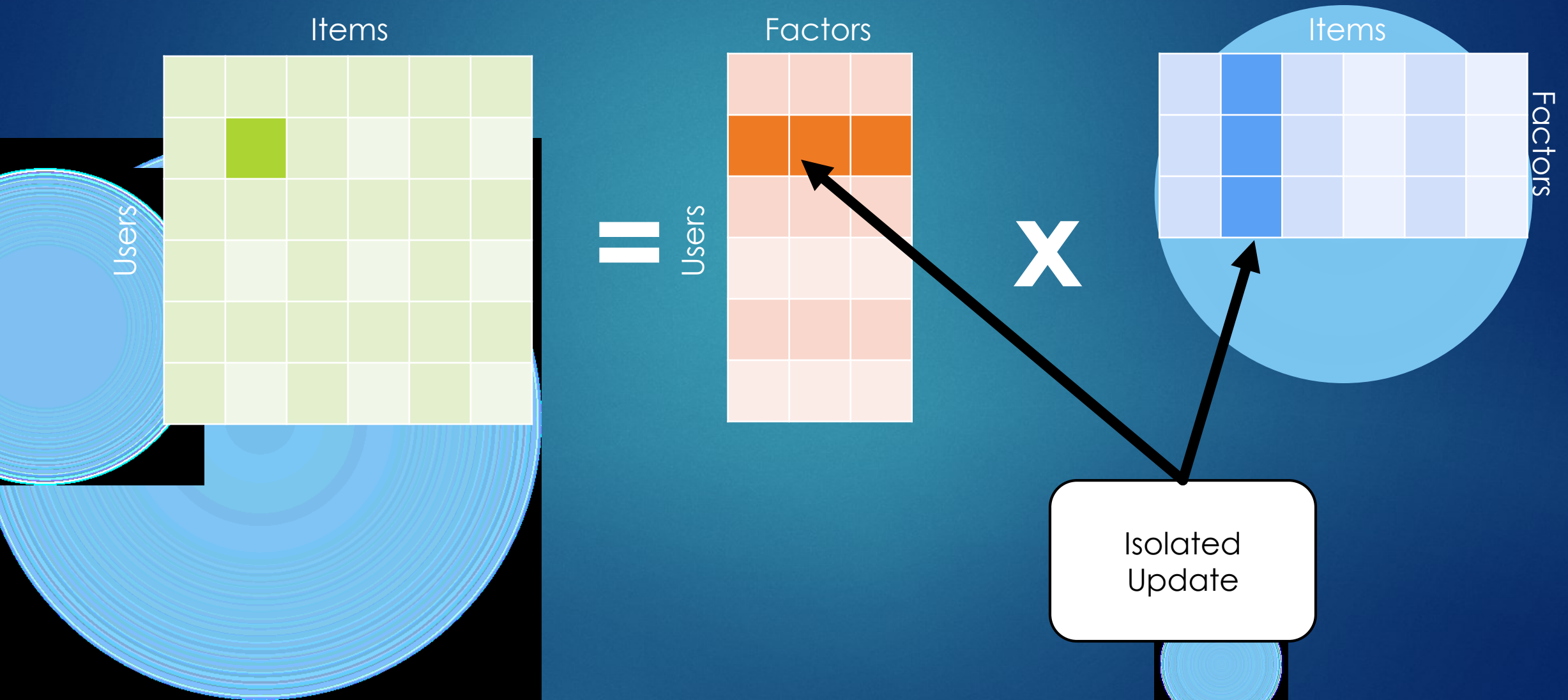
Example: Matrix Factorization



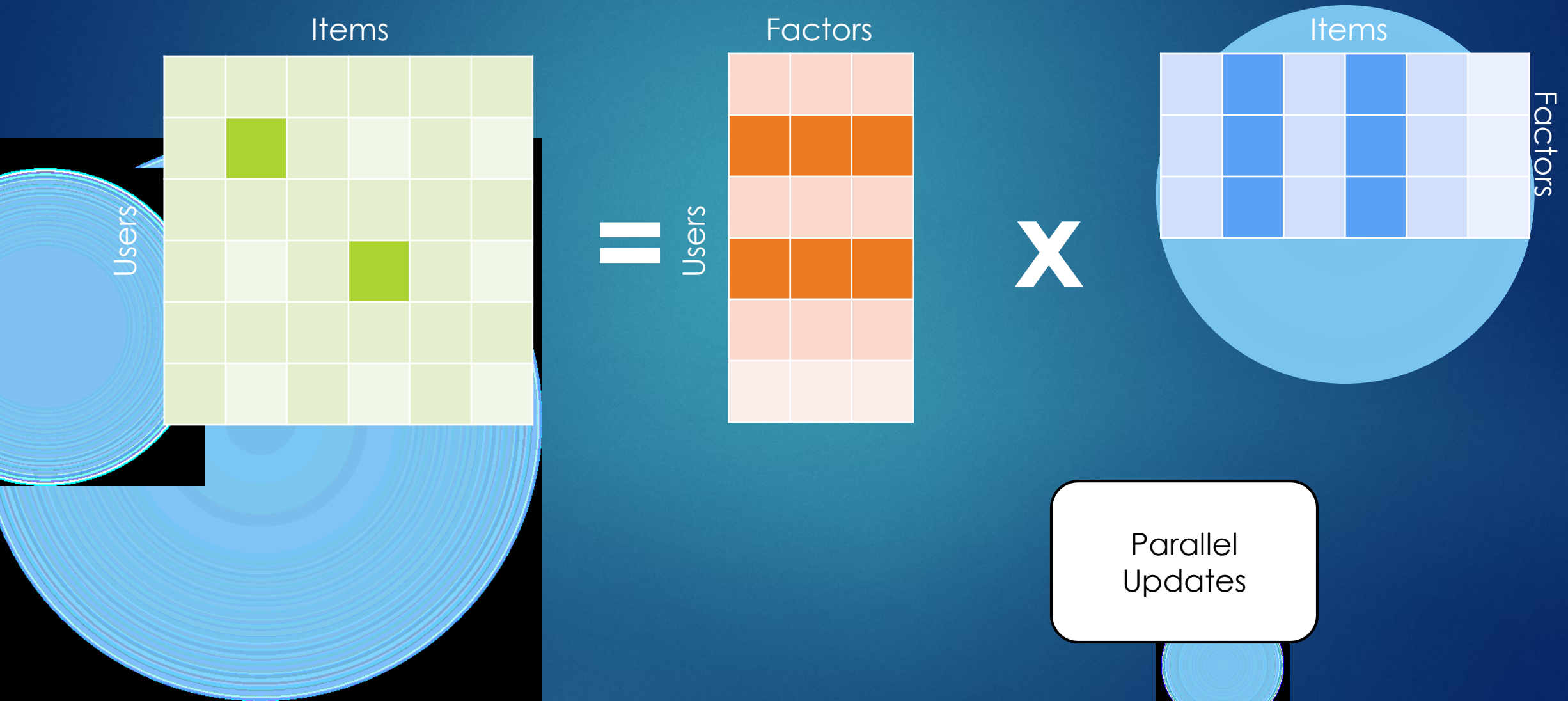
Example: Matrix Factorization



Example: Matrix Factorization

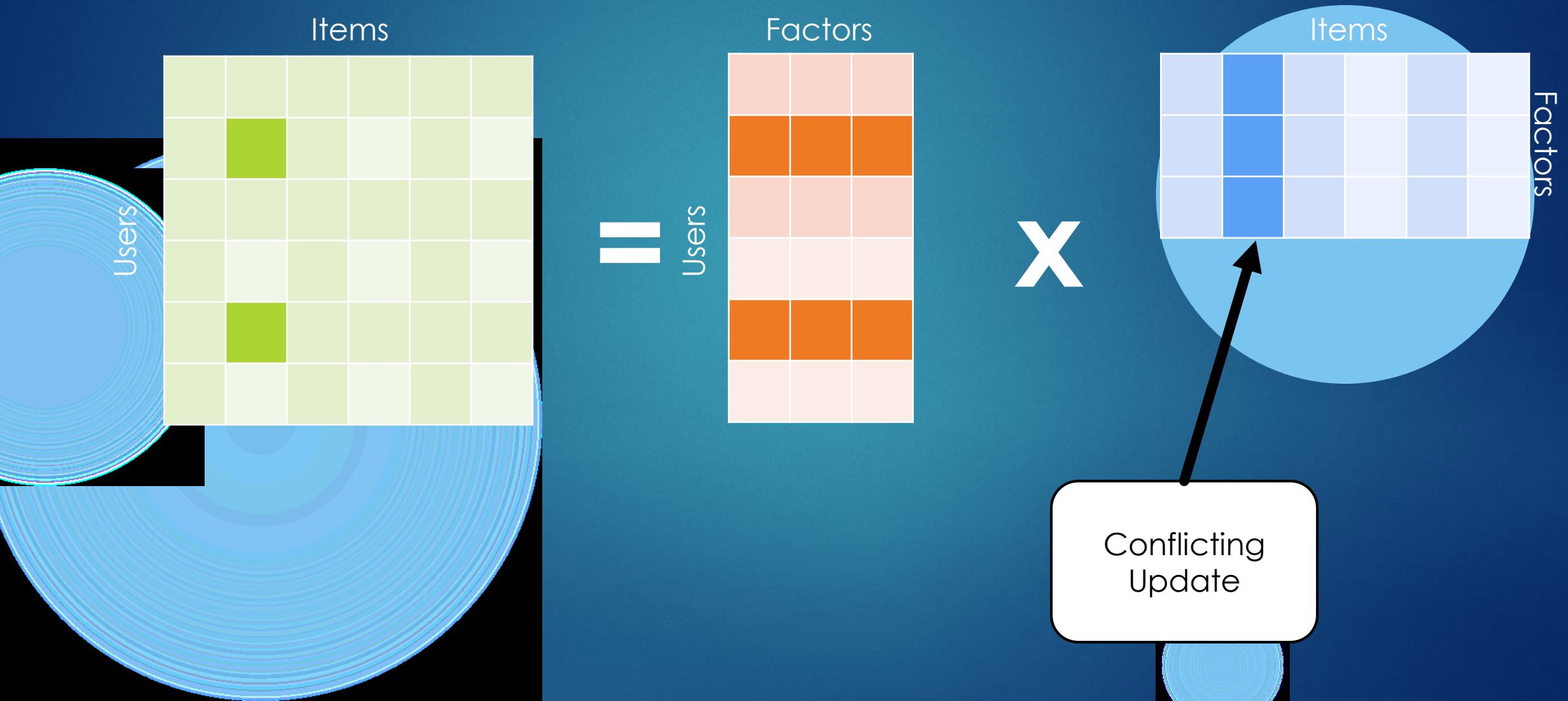


Example: Matrix Factorization

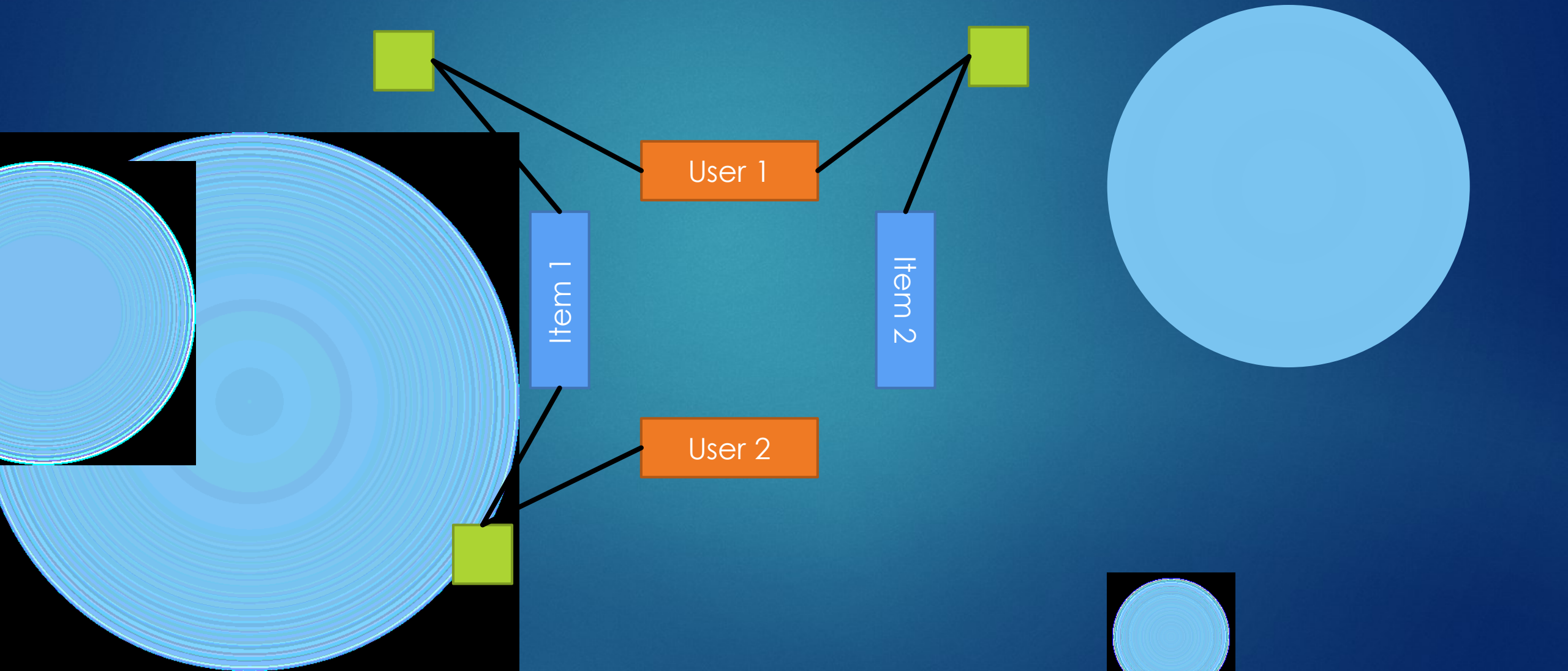


Parallel Updates

Example: Matrix Factorization



Graph View on Matrix Factorization



Beyond SQM: Summary

- **PageRank:**
  - The model structure is identical to the data structure
  - The learning algorithm follows that structure
- **Matrix Factorization:**
  - The model is structured based on the domain (users and items)
  - The learning algorithm may follow an arbitrary access pattern
- **General Graphical Models:**
  - The Model is arbitrarily structured by the designer
  - Arbitrary access patterns

Emerging Programming Model

- Vertices hold state
- Edges define state dependencies
  - State changes require updates to dependent state
- State transition is defined by user defined functions

Further Reading

- GraphLab: http://graphlab.org/home/publications/

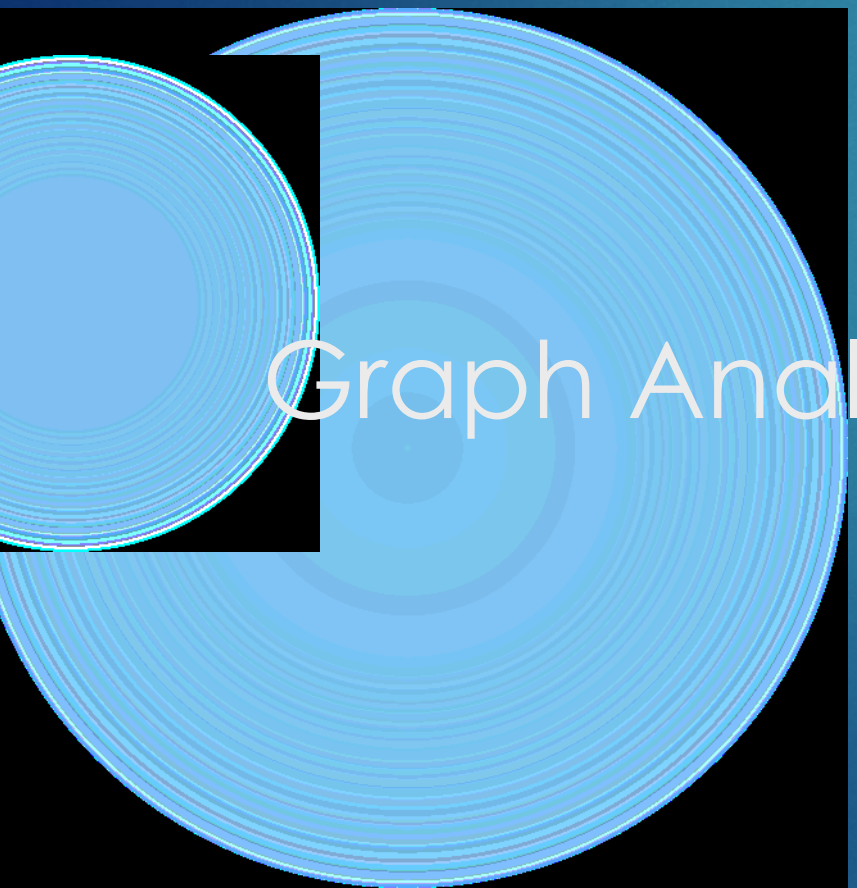
- Ahmed, A., Aly, M., Gonzalez, J., Narayanamurthy, S., and Smola, A.: *Scalable inference in latent variable models*, WSDM '12.
- Rainer Gemulla, Erik Nijkamp, Peter J. Haas, Yannis Sismanis: *Large-scale matrix factorization with distributed stochastic gradient descent*, KDD '11

Take-Away from this part

- Machine Learning works better with more data and more powerful models
- Both of which are enabled by better systems
- Computational models are emerging as an interface between systems and algorithms:
  - Statistical Query Model: Captures many methods and matches iterative Map-Reduce-Update
  - Graph Analysis View: Not as clear a picture, but evolving

System Support for ML

- Assume we have an ML algorithm expressed in:
  - Statistical Query Model (SQM), or
  - Graph Analysis
- How do we run it in a scalable fashion?
- We will cover the following approaches:
  - Graph-analytics systems
  - Dataflow-based systems



Graph Analytics Systems

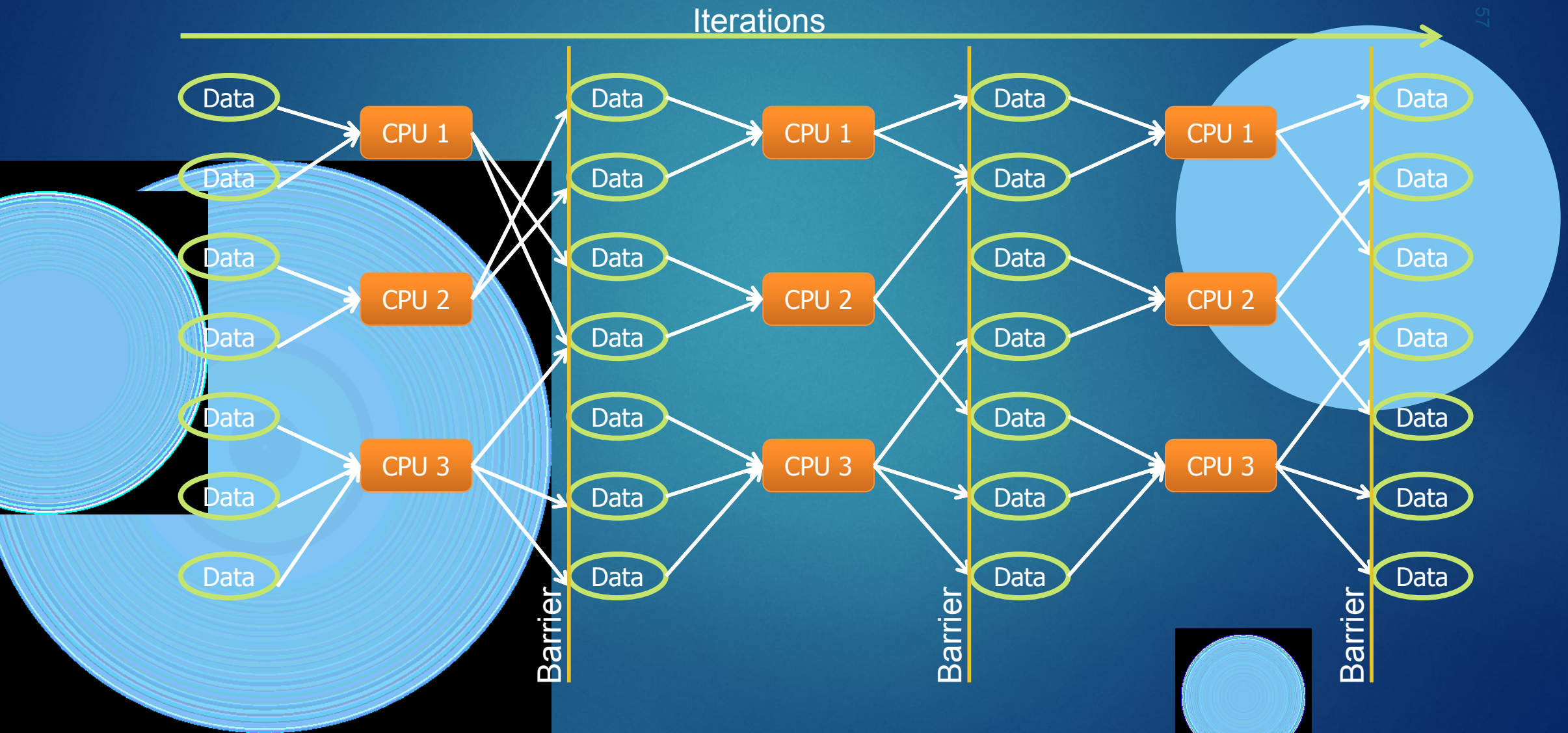
Processing Big Graphs

- Options
  - Create custom distributed infrastructure
  - Just use MapReduce
  - Use a graph library: BGL, LEDA, NetworkX
- Problems
  - Custom solutions do not generalize well
  - MapReduce is not the right programming model
    - And it is inefficient!
  - Graph libraries cannot handle web-scale problems

Google Pregel

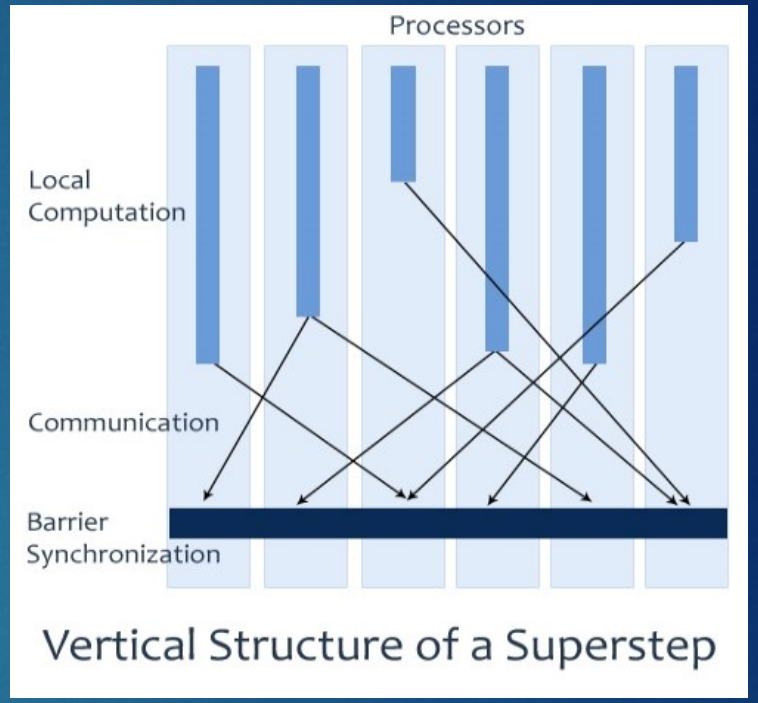
- Graph processing Framework
  - High scalability
  - Fault-tolerance
  - Graph-oriented programming model
- Inspired by Valiant's Bulk Synchronous Parallel (BSP) model
- The Pregel name honors Leonhard Euler
  - The Bridges of Königsberg, which inspired his famous theorem, spanned the Pregel river

Bulk Synchronous Parallel Model



Pregel programming model

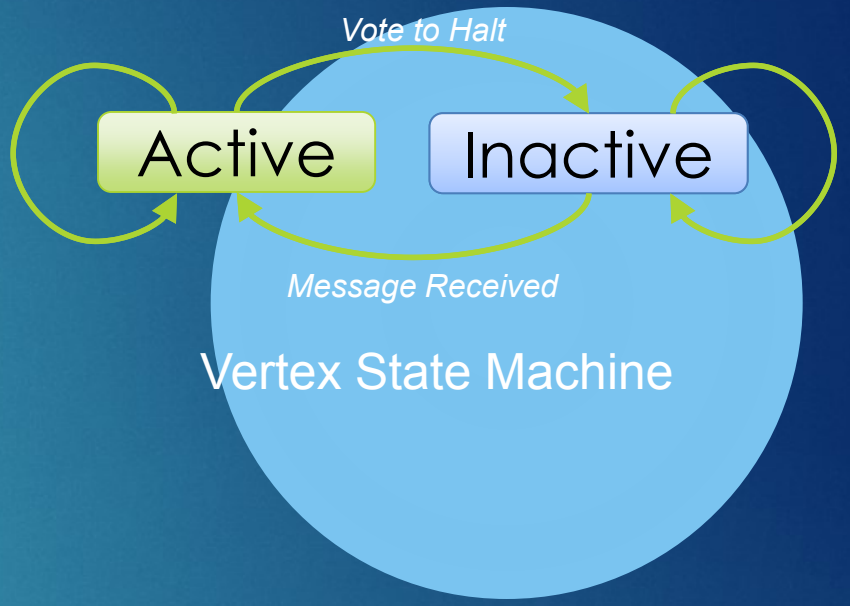
- Model of Computation
  - Input: a graph of vertices and edges
  - Each vertex holds a modifiable user defined value
  - Each edge is associated with a source vertex, value and a destination vertex
- Runtime executes a sequence of iterations called *Supersteps*
  - A user-defined function F is executed at each vertex V
  - F can read messages sent to V in superstep S – 1 and send message to other vertices, which will be received at superstep S + 1
  - F can modify the state of V and its outgoing edges
  - F can change the topology of the graph



Vertical Structure of a Superstep

Algorithm Termination

- Vote to halt protocol
  - In superstep 0, every vertex is active
  - All active vertices participate in the superstep
  - A vertex deactivates itself by "voting to halt"
  - A vertex is reactivated if it receives an external message
- Program terminates when all vertices are simultaneously inactive and there are no messages in transit



The Pregel API in C++

- A Pregel program is written by subclassing the vertex class:

```
template <typename VertexValue,
typename EdgeValue,
typename MessageValue>

class Vertex {
public:
    virtual void Compute(MessageIterator* msgs) = 0;

    const string& vertex_id() const;
    int64 superstep() const;
    const VertexValue& GetValue();
    VertexValue* MutableValue();
    OutEdgeIterator GetOutEdgeIterator();

    void SendMessageTo(const string& dest_vertex,
    const MessageValue& message);

    void VoteToHalt();
};
```

To define the types for vertices, edges and messages

Override the compute function to define the computation at each superstep

To get the value of the current vertex

To modify the value of the vertex

To pass messages to other vertices

Pregel Code for PageRank


```
class PageRankVertex : public Vertex<double, void, double> {
Public:
virtual void Compute(MessageIterator* msgs) {
  if (superstep() >= 1) {
     double sum = 0;
     for (; !msgs->Done(); msgs->Next())
          sum += msgs->Value();
          *MutableValue() = 0.15 / NumVertices() + 0.85 * sum;
     }

     if (superstep() < 30) {
          const int64 n = GetOutEdgeIterator().size();
          SendMessageToAllNeighbors(GetValue() / n);
     } else {
          VoteToHalt();
     }
  } else {
     SendMessageToAllNeighbors(1f / NumVerticies());
  }
};
```

Vertex type with a value and message type of double, and no edge value

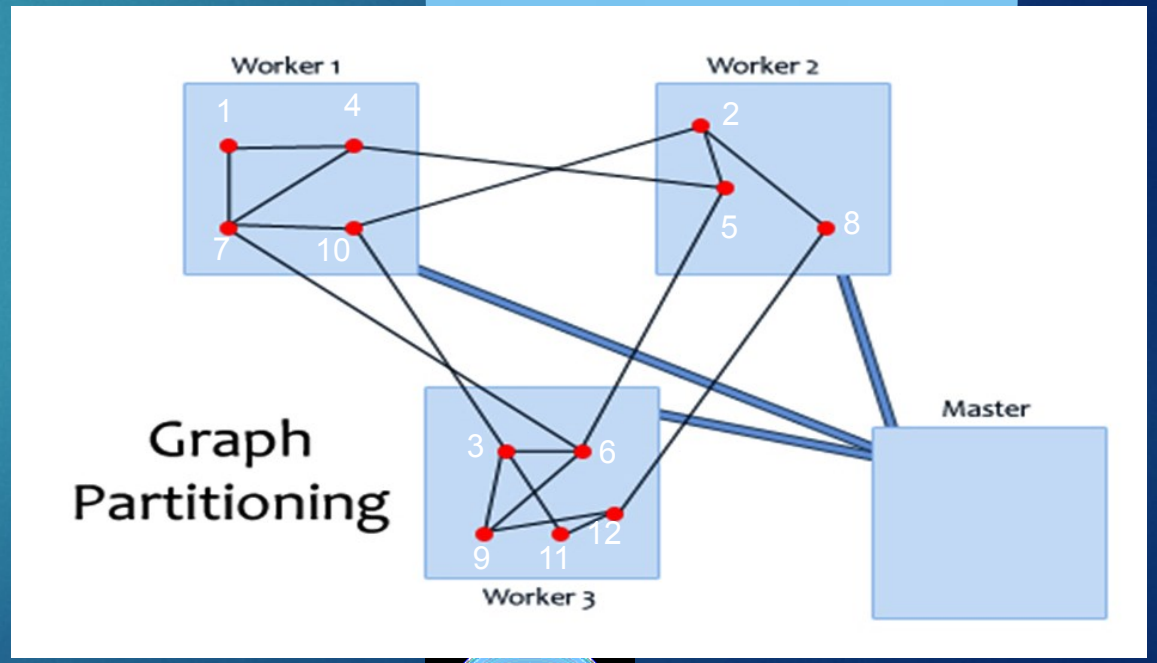
Compute my tentative PageRank

Send my tentative PageRank/n to my neighbors

Send my initial PageRank estimate

Pregel runtime

- Master/Worker Architecture
  - Similar to GFS and MapReduce
  - Master partitions the graph and schedules supersteps
  - Workers execute active vertices
- Graph partitioning
  - The input graph is divided into partitions
  - Default partition function: hash(VertexID) mod N
    - Where N is the # of partitions



Fault Tolerance in Pregel

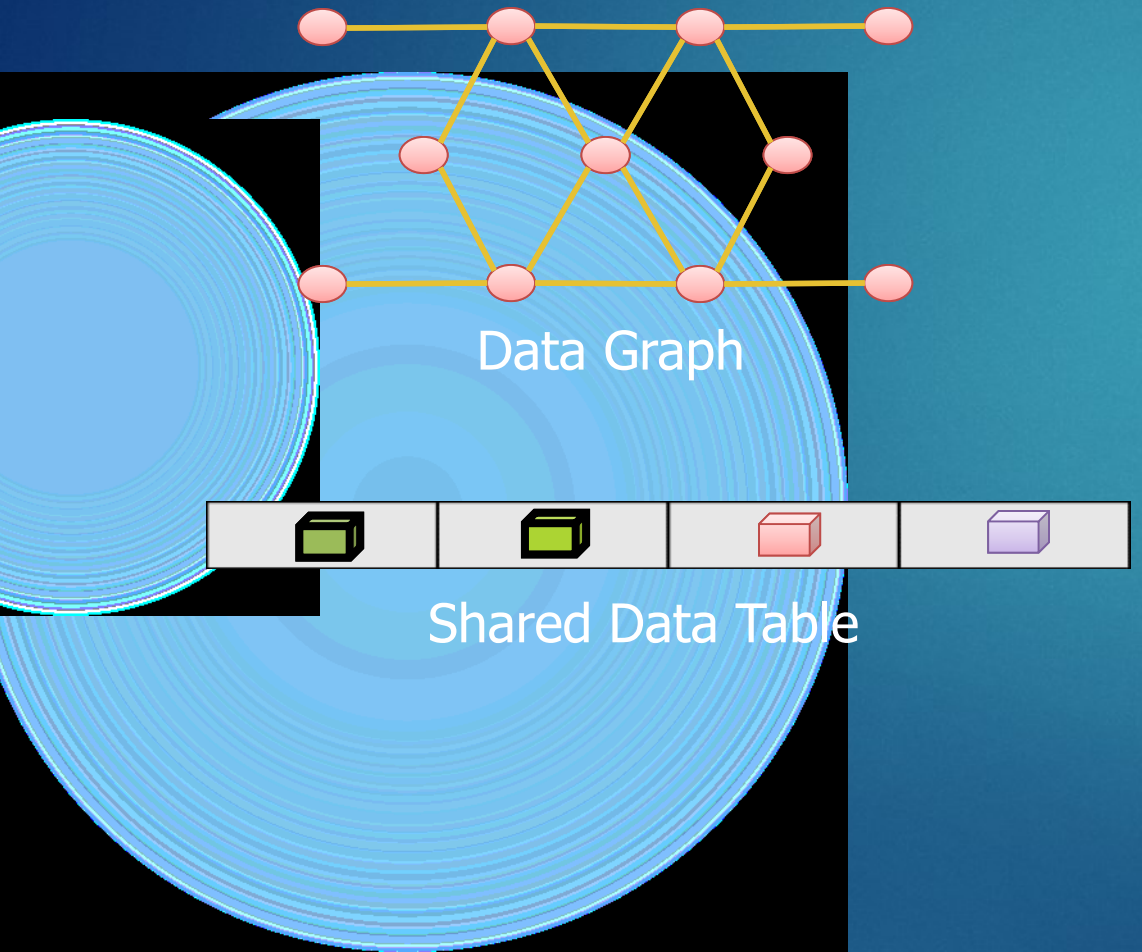
- Fault tolerance is achieved through checkpointing
  - At the start of every superstep the master may instruct the workers to save the state of their partitions in a stable storage (e.g., GFS)
- Master uses ping messages to detect worker failures
- If a worker fails, the master reassigns vertices/input to another available worker and restarts the superstep
  - The new worker reloads the partition state from the most recent checkpoint

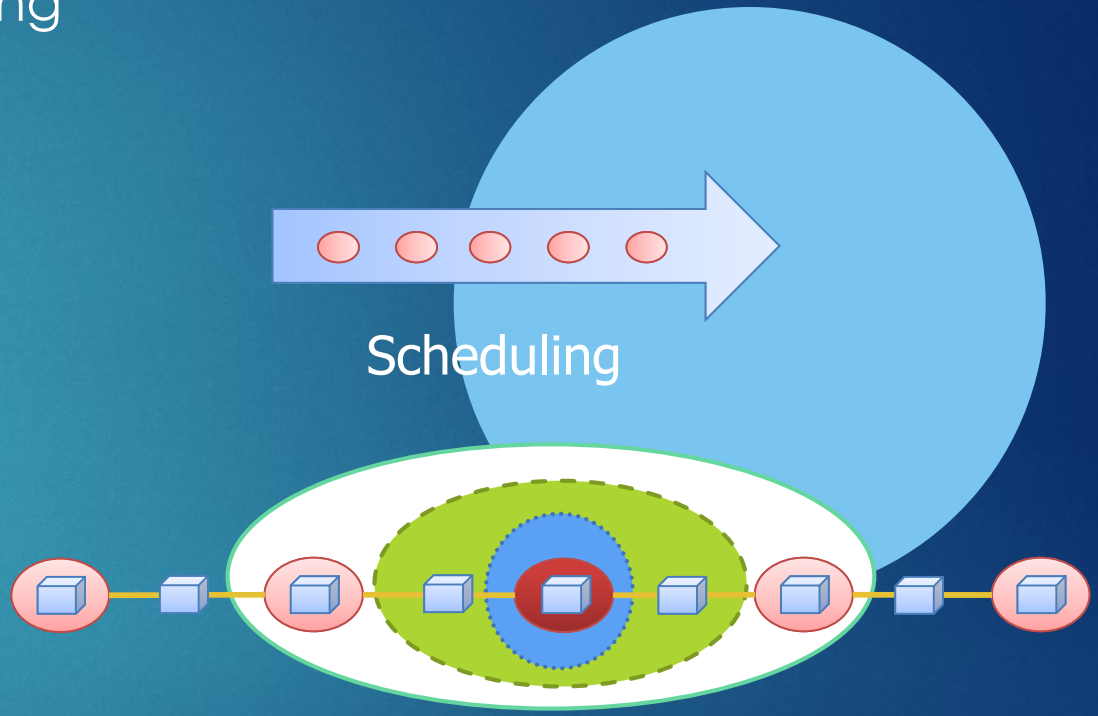
Motivation for GraphLab

- Shortcomings of Google MapReduce
  - Programming model does not fit Graph computations
  - Overheads of running jobs iteratively --- disk access and startup costs
- Shortcomings of Google Pregel
  - BSP model requires synchronous computation
  - Slowest machine determines computation speed
- Shortcomings of Microsoft Dryad
  - Complex programming model

GraphLab

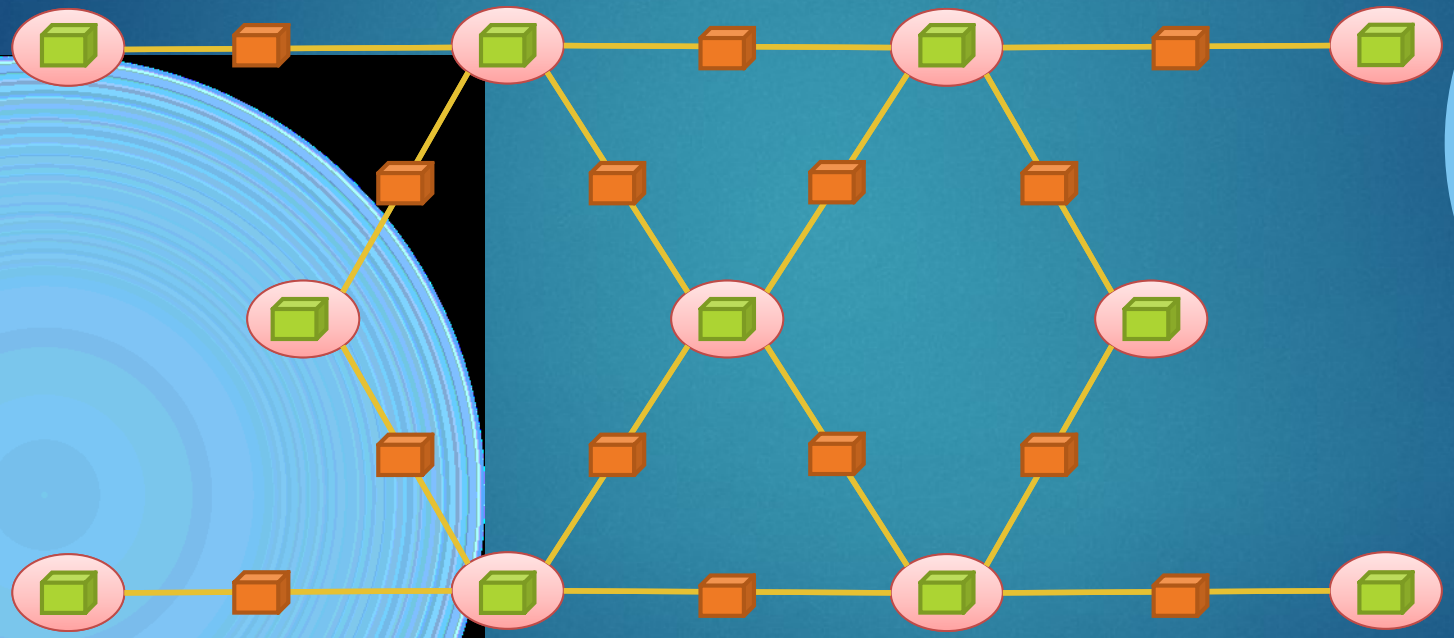
- A framework for parallel machine learning





Data Graph

- Graphs associate data at every vertex and edge

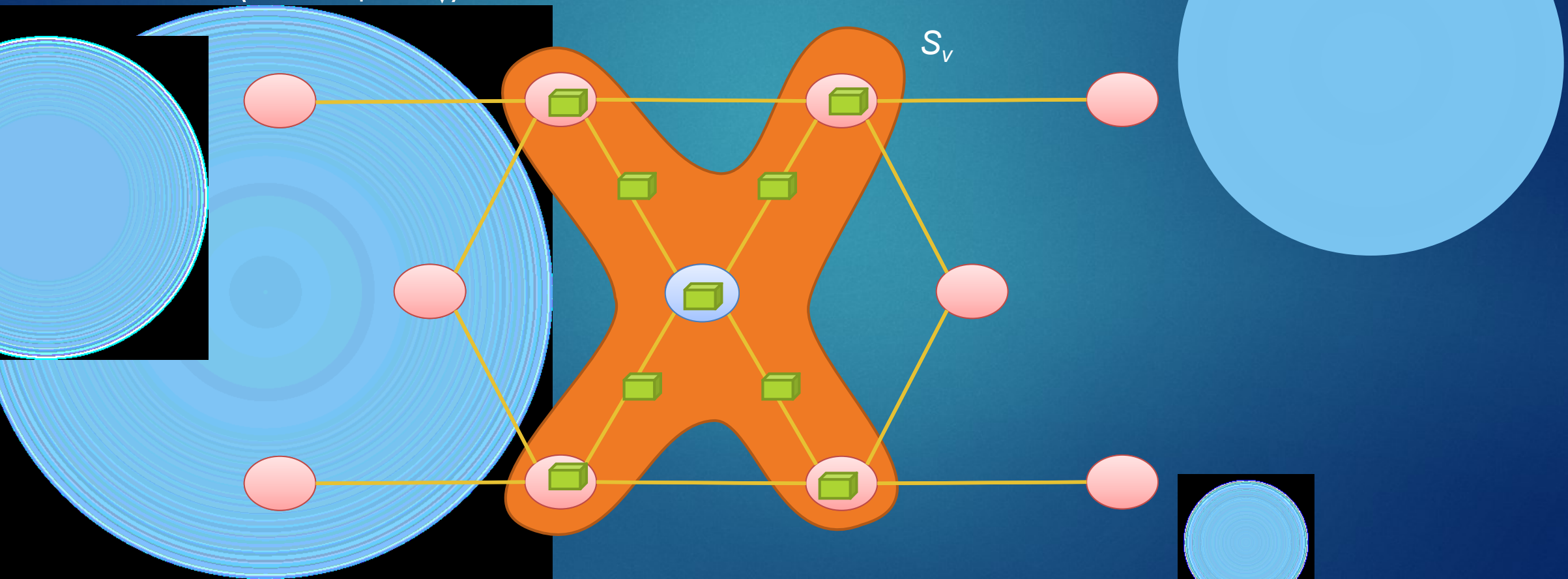


Data Graph

Arbitrary blocks of data can be assigned to vertices and edges

Update Functions

- Data graph is modified via update functions
  - The update function can modify a vertex v and its neighborhood (or scope S_v)



Shared Data Table

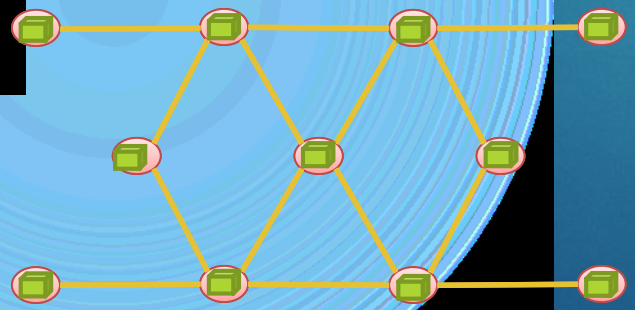
- Certain algorithms require global information shared among all vertices (Algorithm Parameters, Statistics, etc.)
  - GraphLab exposes a Shared Data Table (SDT)
- SDT is an associative map between keys and data blocks
  - T[Key] => Data Block

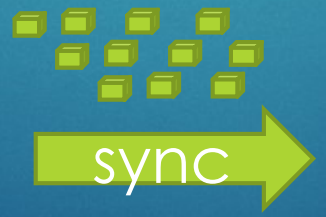
Shared Data Table

- The shared data table is updated using the ***sync mechanism***

Sync Mechanism

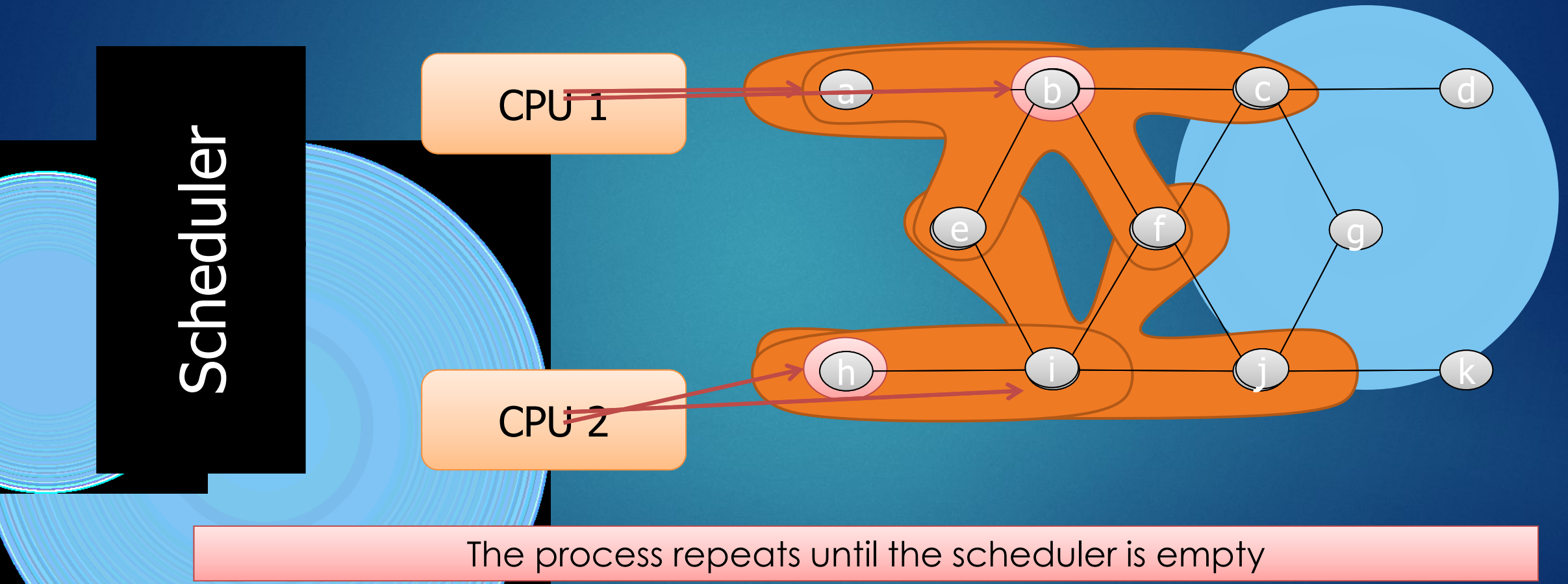
- Similar to "Reduce" in MapReduce
  - User defines fold, merge and apply functions that are triggered during a global sync mechanism
- **Fold** allows the user to aggregate information across all vertices
- **Merge** (optionally) allows a parallel tree reduction
  - Similar to "Combiners" in MapReduce
- **Apply** finalizes the result from fold/merge and commits to the SDT





Shared Data Table

Scheduling in GraphLab



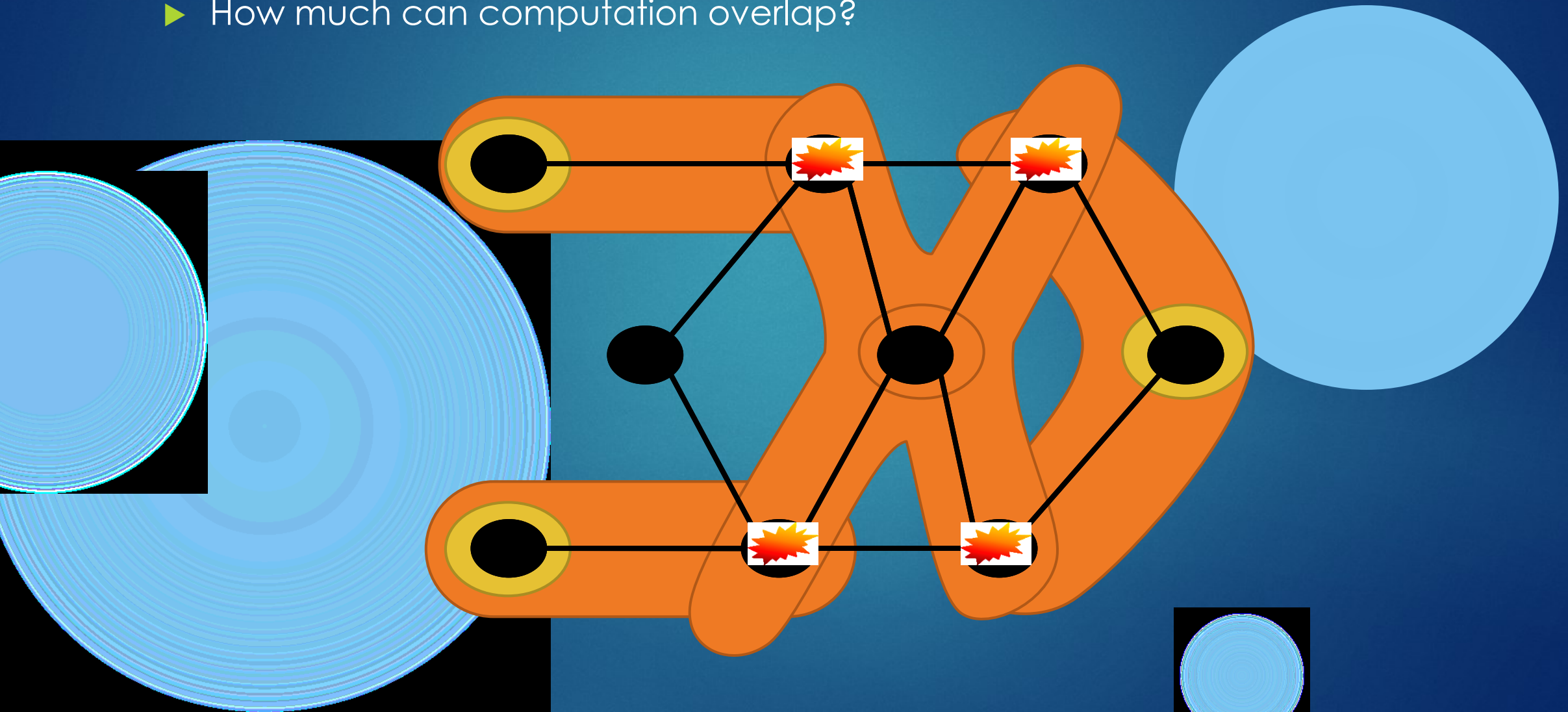
The process repeats until the scheduler is empty

Scheduling in GraphLab

- An update schedule defines the order in which update functions are applied to vertices
  - A parallel data-structure called the scheduler represents an abstract list of tasks to be executed in GraphLab
- Base (Vertex) schedulers in GraphLab
  - Synchronous and Round-robin
- Job Schedulers in GraphLab
  - FIFO and Priority
- Custom schedulers can also be defined
- Termination Assessment
  - If the scheduler has no remaining tasks
  - Or, a termination function can be defined to check for (algorithmic) convergence

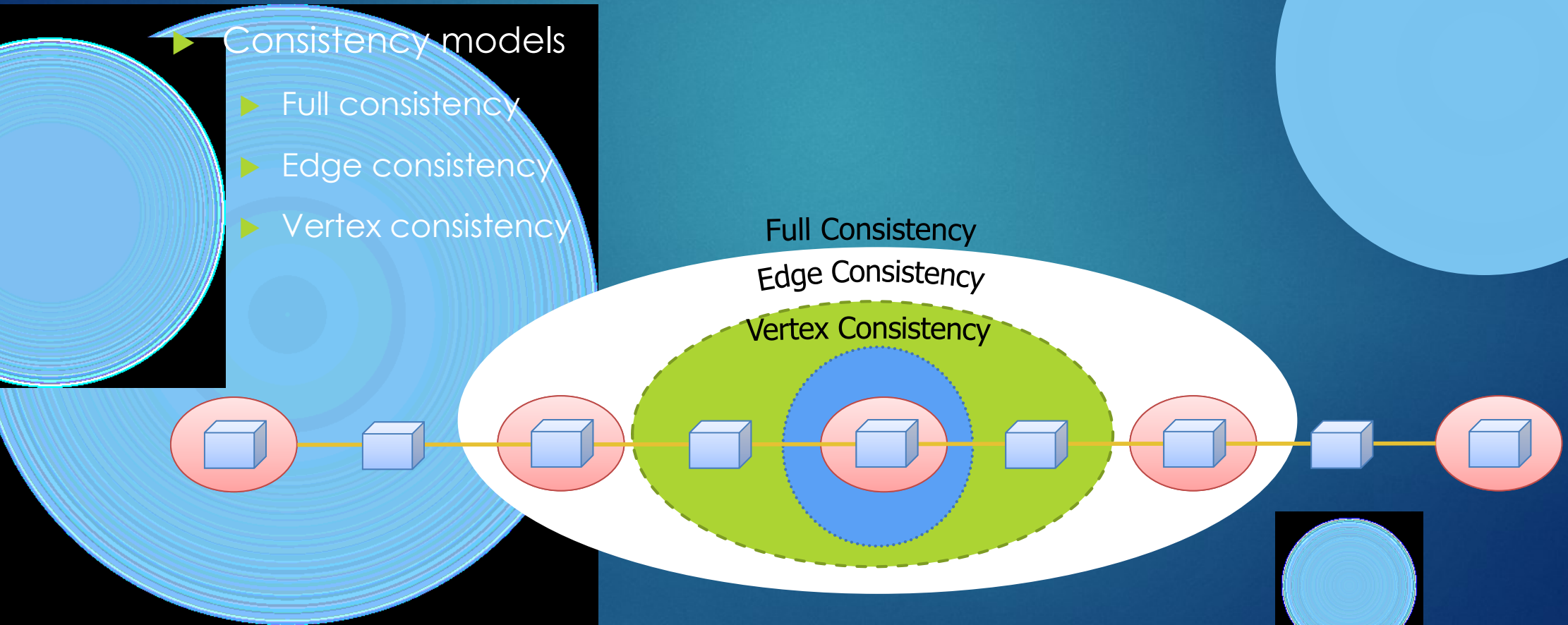
Need for Consistency Models

- How much can computation overlap?



Consistency Models in GraphLab

- GraphLab guarantees sequential consistency
  - Garanteed to give the same result in a sequential execution of the computational steps
- Consistency models
  - Full consistency
  - Edge consistency
  - Vertex consistency

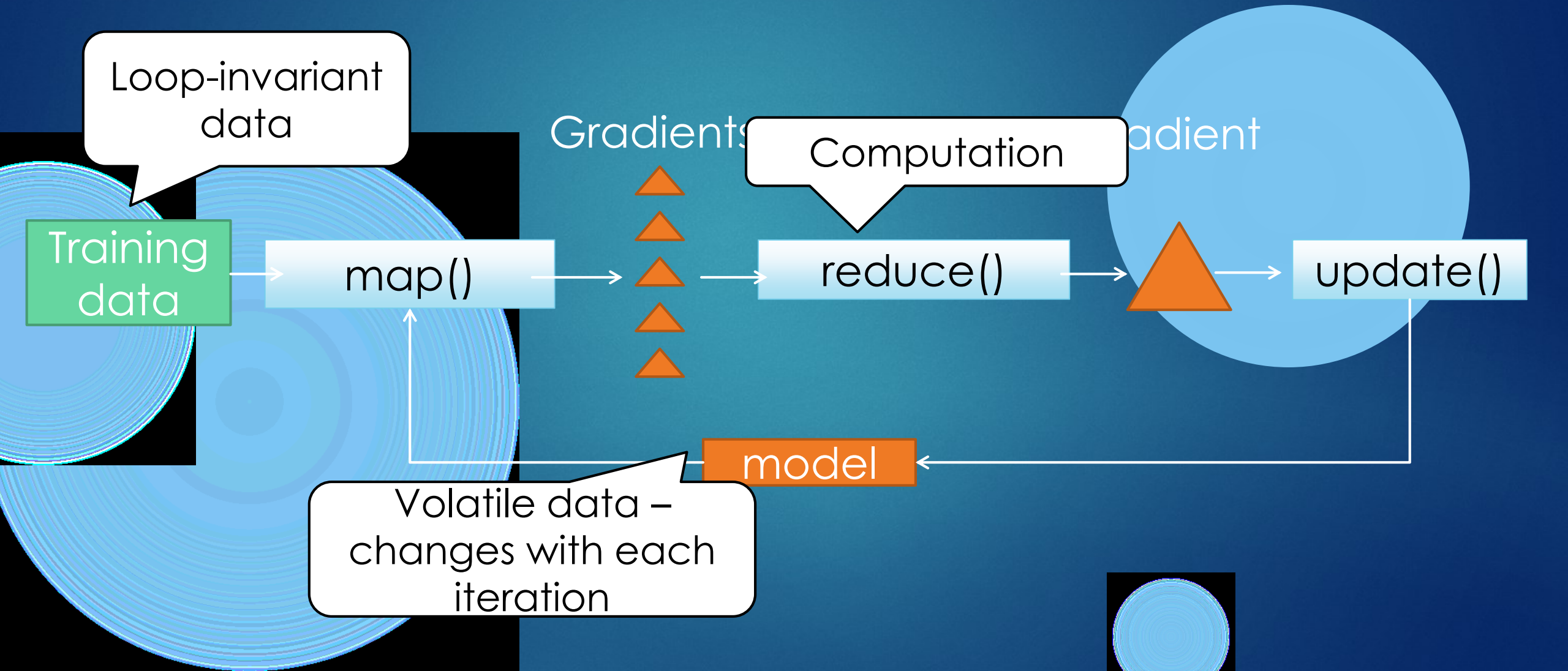


Dataflow Systems

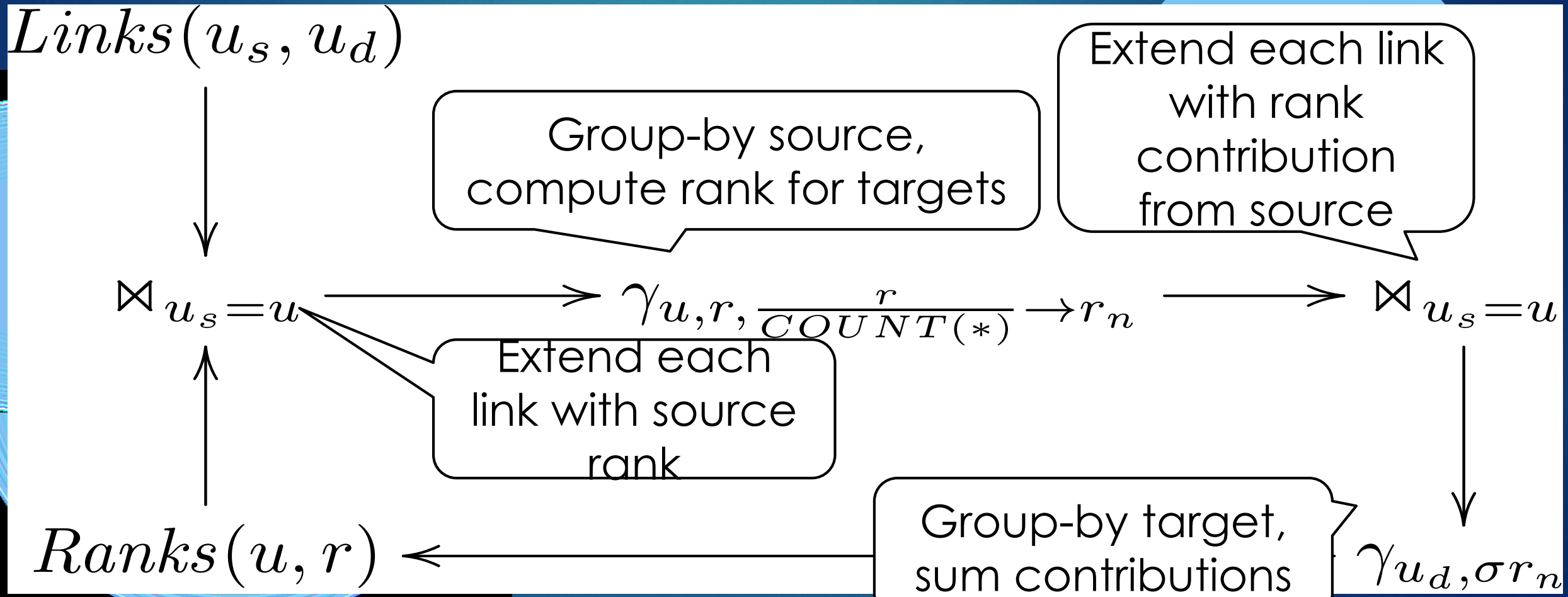
Dataflow Systems

- Computation expressed as graph of operators
  - An operator transforms input to some output
  - Data flows along edges of the graph
- Iteration is captured as a cycle in the dataflow
- Think of a RA query plan + UDFs + iteration
- Scaling through distributed computation
  - Job scheduling/data movement
  - Fault tolerance

Example: BGD

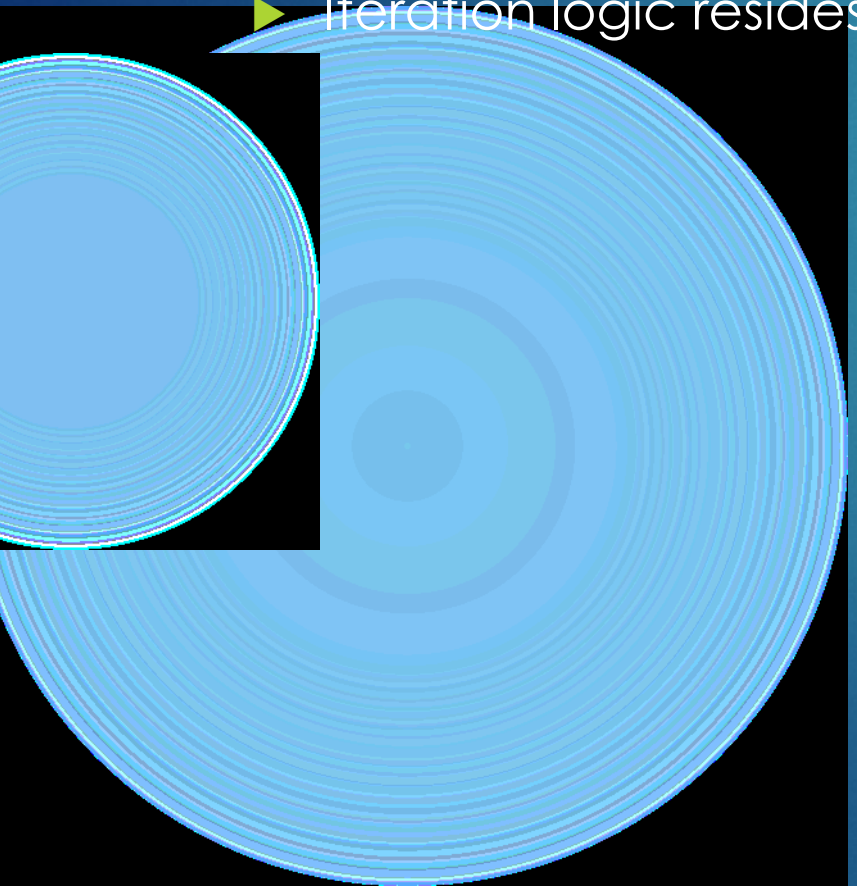

Loop-invariant data
Training data
map()
Gradients
Computation
reduce()
update()
model
Volatile data – changes with each iteration

Example: Pagerank



Baseline: Hadoop

- One iteration is a DAG of MR jobs
- Iteration logic resides in the application



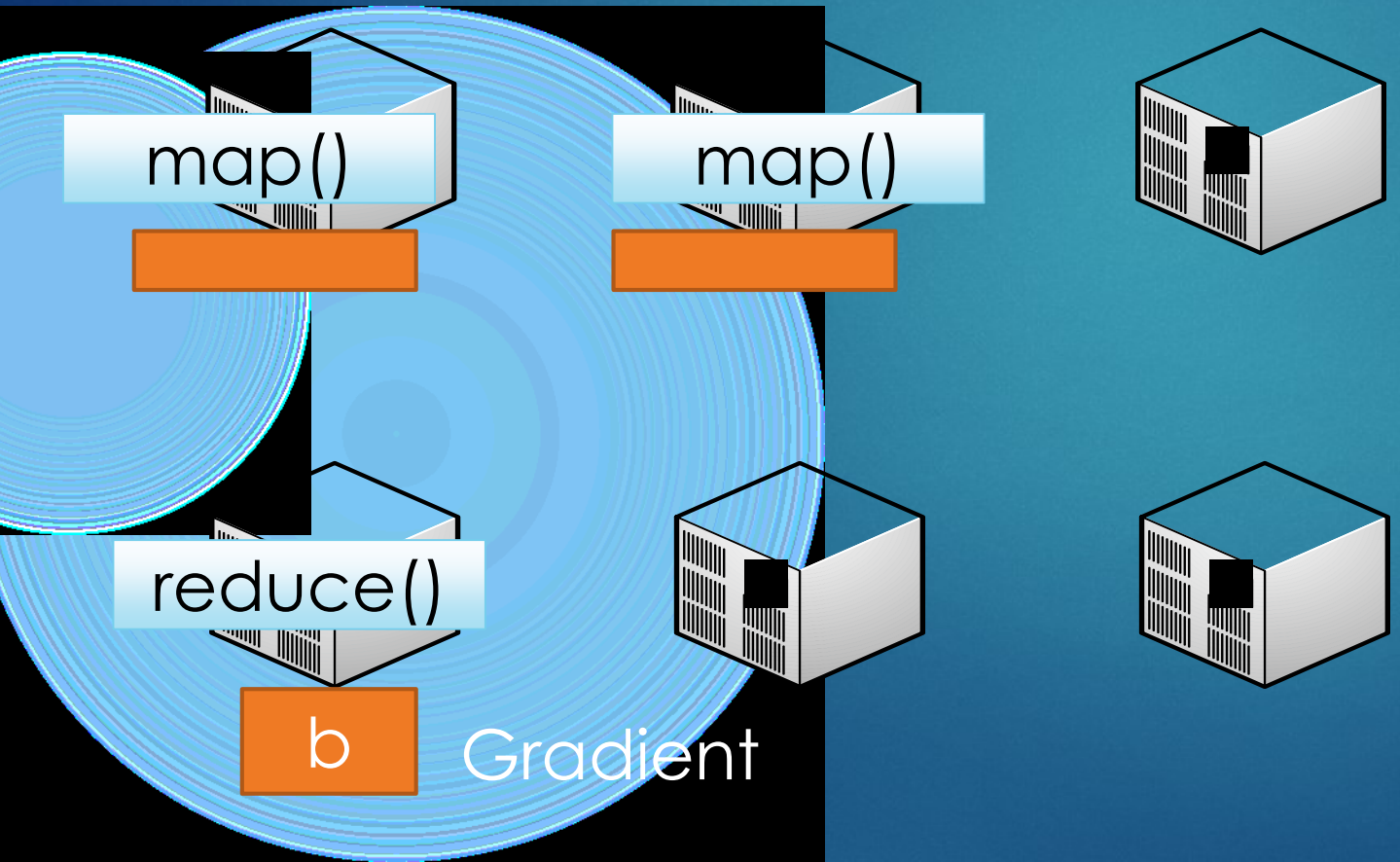
BGD in Hadoop

Training data

1 2

Model

a'



Iteration 1

Application

- Apply update()
- Continue?

Iteration 2

Shortcomings of Baseline

- Every iteration re-reads loop-invariant data
  - Batch Gradient Descent: Training data
  - PageRank: Links table
- Intrinsic overheads of Map/Reduce (cost to start-up tasks and controller)
  - Empirical evidence on startup cost of MapReduce job in Hadoop is in the order of a minute.

Haloop: Hadoop + Loops

Y. Bu et al., "HaLoop: Efficient Iterative Data Processing on Large Clusters", VLDB 2010

- Extended MapReduce API to express iteration:
  - Designate the loop body
  - Specify loop termination (max iterations or convergence)
  - Designate loop-invariant data
- Haloop optimizes access to loop-invariant data
  - Physical caching and indexing of data
  - Loop-aware job scheduling

Data Access in Haloop

Training Data

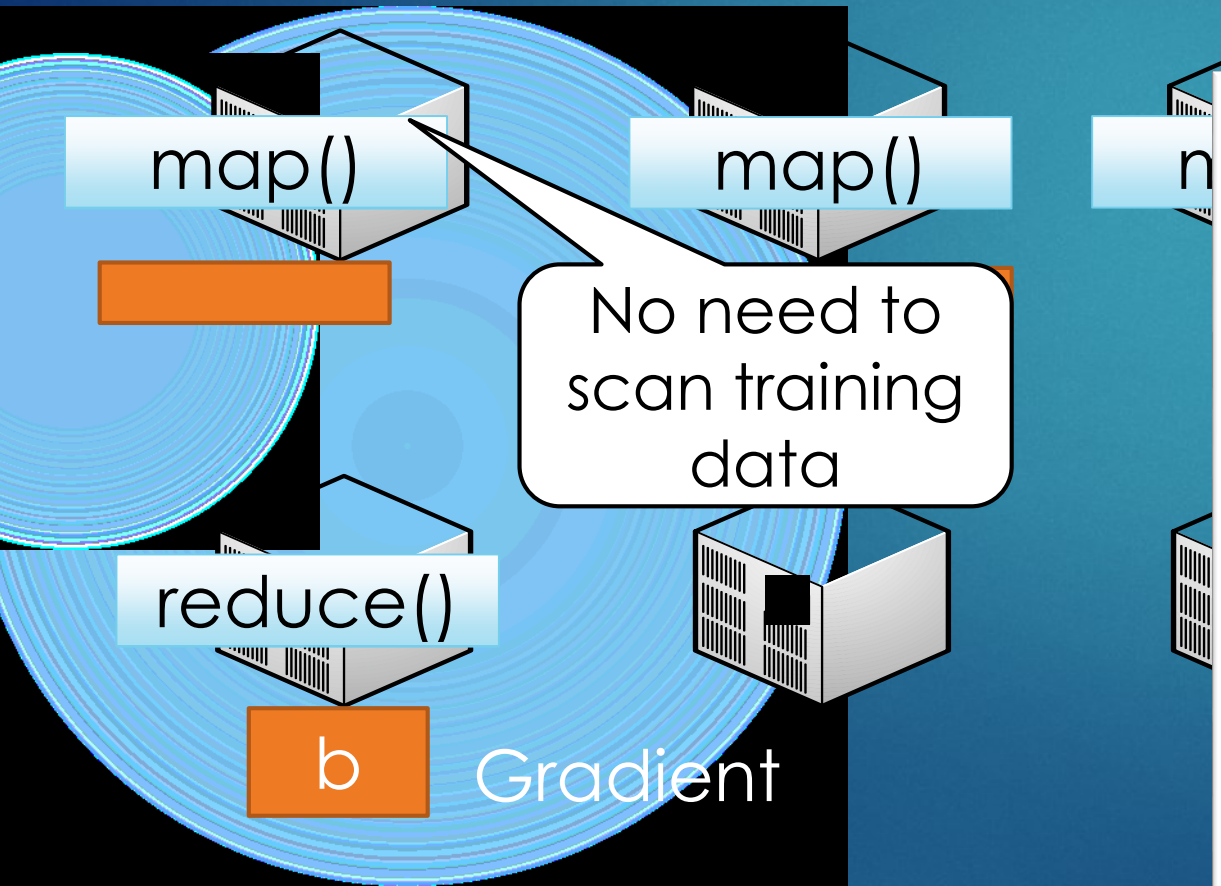
1 2

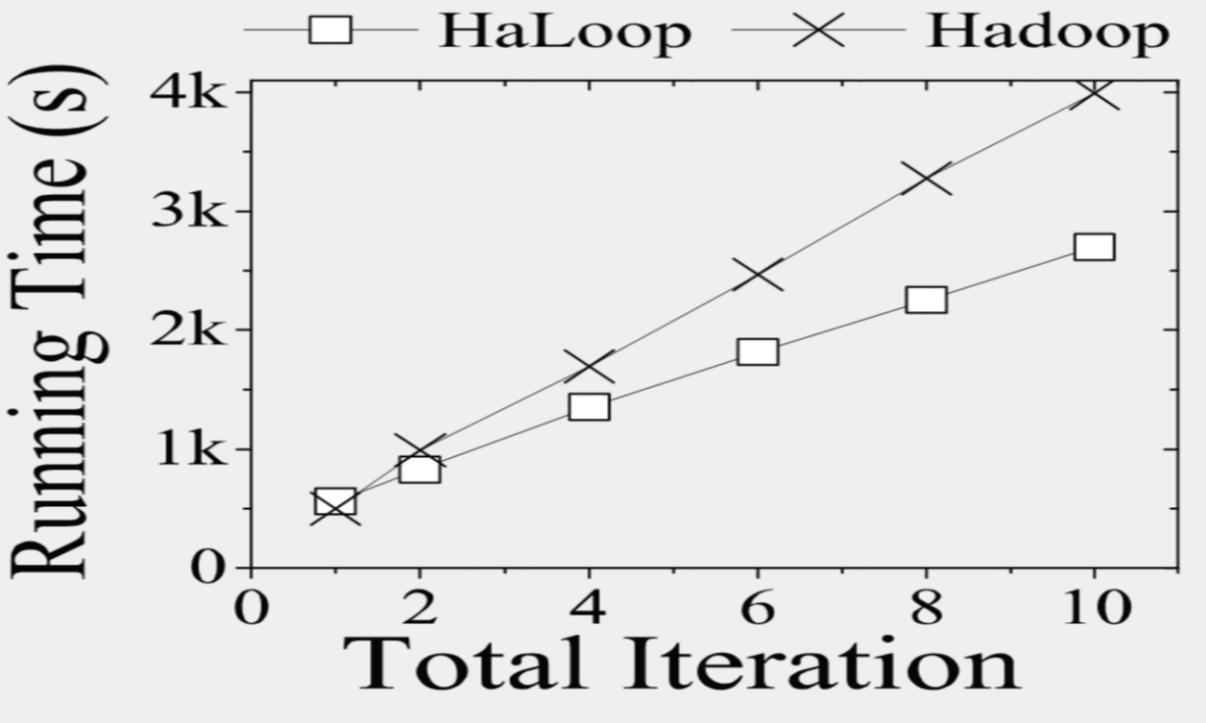
Model

a'

Iteration 1

Iteration 2

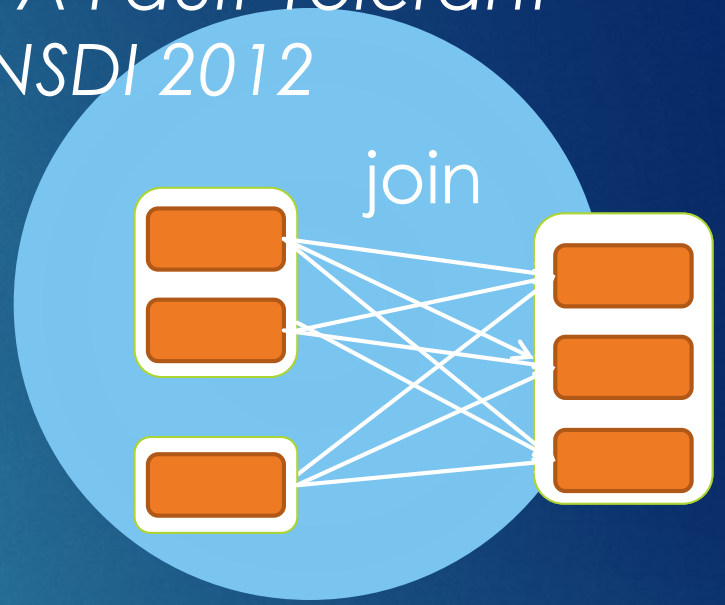




Spark: In-memory dataflows

M. Zaharia, et al., "Resilient Distributed Datasets: A Fault-Tolerant Abstraction for In-Memory Cluster Computing", NSDI 2012

- Key notion: Resilient Distributed Dataset (RDD)
  - Resilient = recoverable if failures occur
  - Distributed = partitioned across nodes
  - Also, immutable
- RDDs can be created from:
  1. scanning data in stable storage, or
  2. running an operator on other RDDs
- A dataflow consumes RDDs and outputs RDDs



filter or map

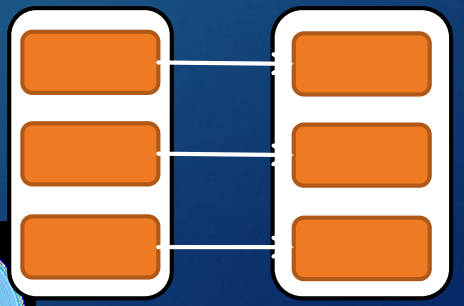


Illustration: BGD in Spark

Input Data

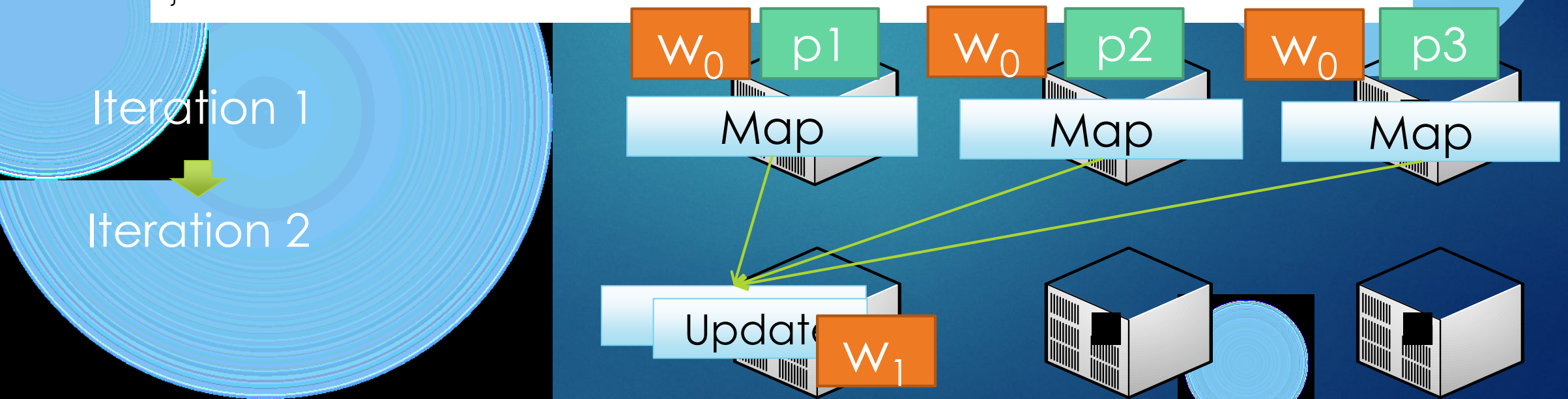
```
val points = spark.textFile(…).map(parsePoint).persist
var w = // random initial vector
for (i <- 1 to ITERATIONS) {
    val gradient = points.map {
        p=> p.x * (1/(1+exp(-p*y(w dot p.x)))-1)*p.y
    }.reduce((a,b) => a+b)
    w -= gradient
}
```

Model

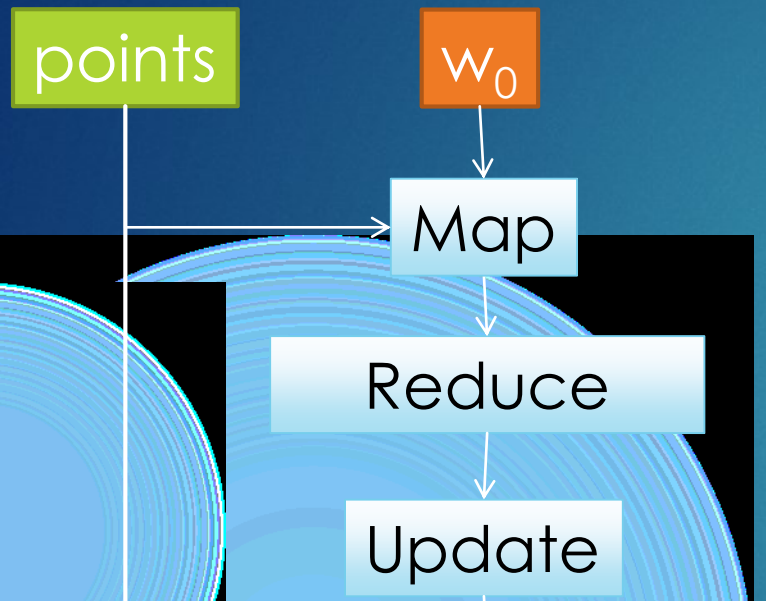
Map

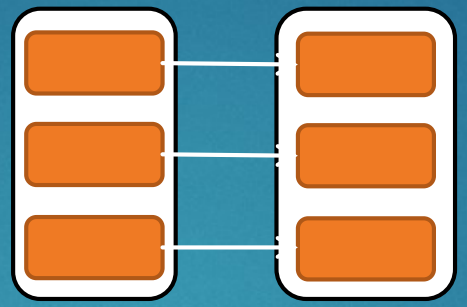
Reduce

Update

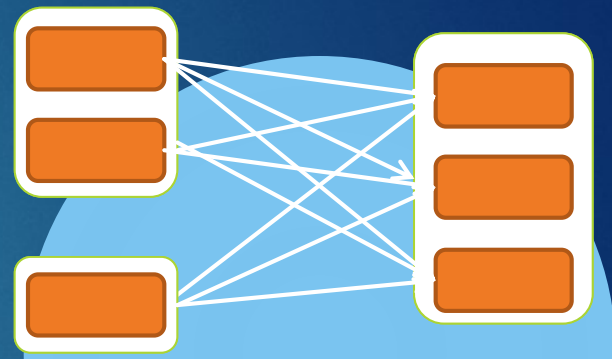


Lineage-based Recovery

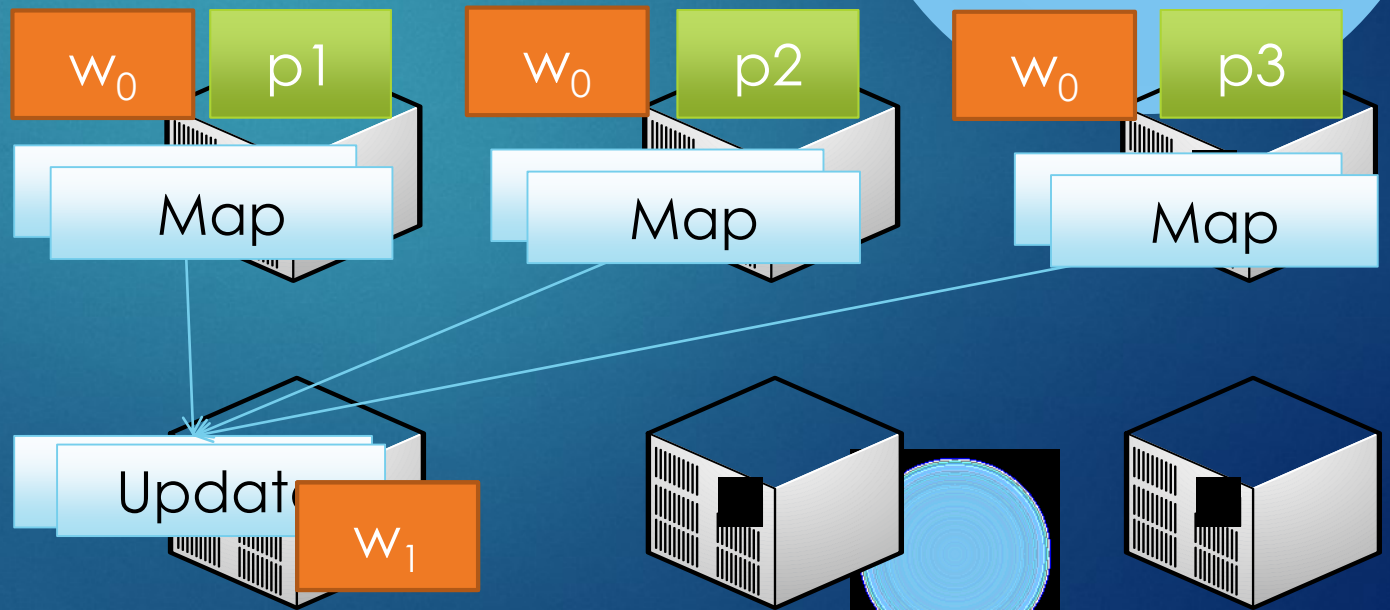




Narrow dependency
(e.g., Map)

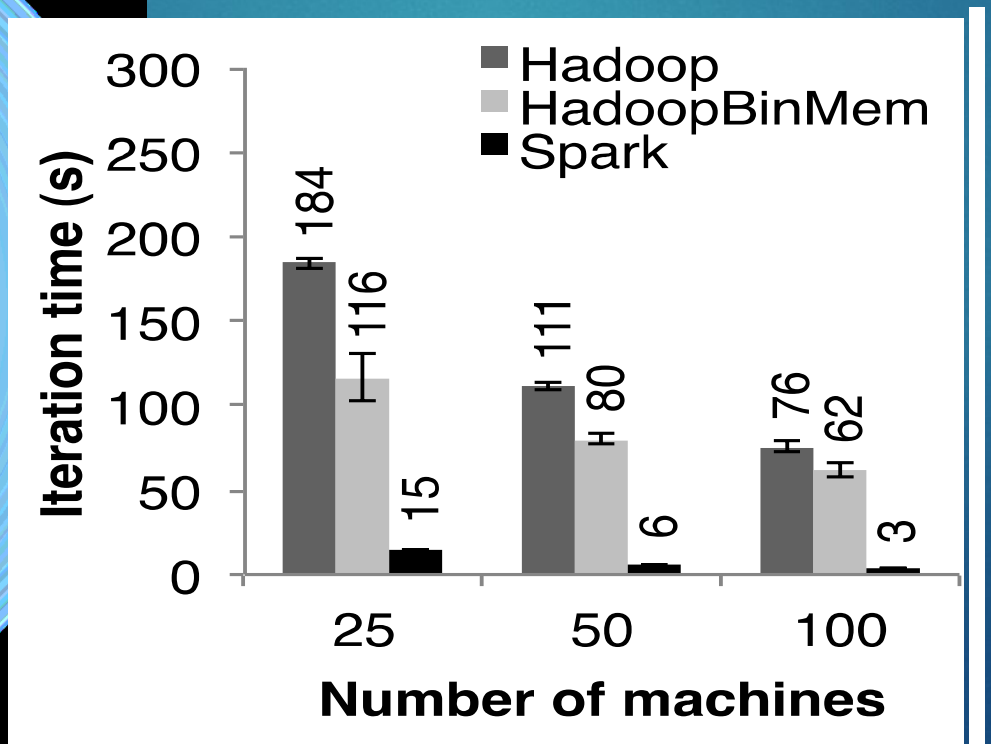


Wide dependency
(e.g., Reduce)



Spark's Execution Engine

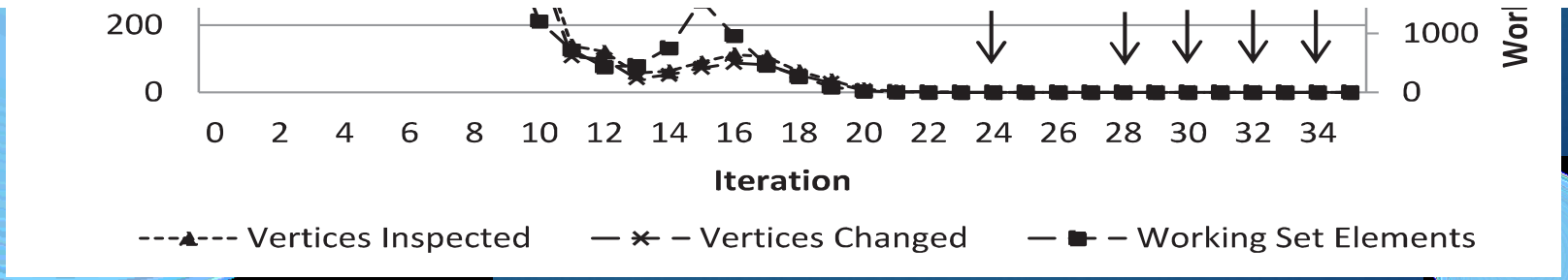
- Loop-invariant data can be pinned in memory
- Computation gets scheduled near data
- No (implicit) checkpoints, use lineage instead



Peeking inside iterations

- So far, each iteration is a coarse-grained op

```
val points = spark.textFile(…).map(parsePoint).persist
var w = // random initial vector
for (i <- 1 to ITERATIONS) {
      val gradient = points.map {
            p=> p.x * (1/(1+exp(-p*y(w dot p.x)))-1)*p.y
      }.reduce((a,b) => a+b)
      w -= gradient
}
```

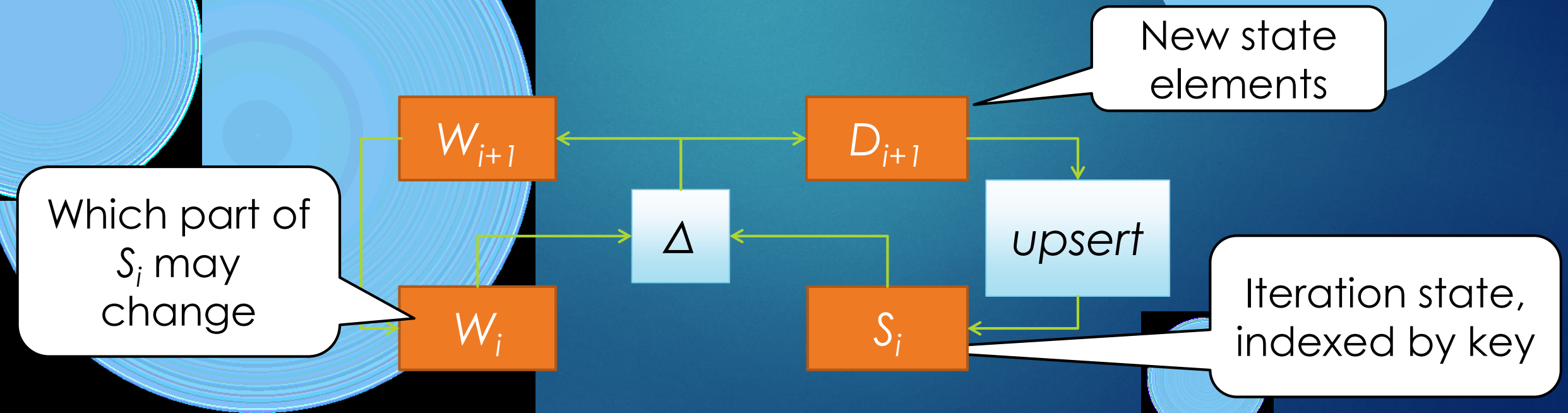


Incremental Iterations

S. Ewen et al., "Spinning Fast Iterative Data Flows", VLDB 2012

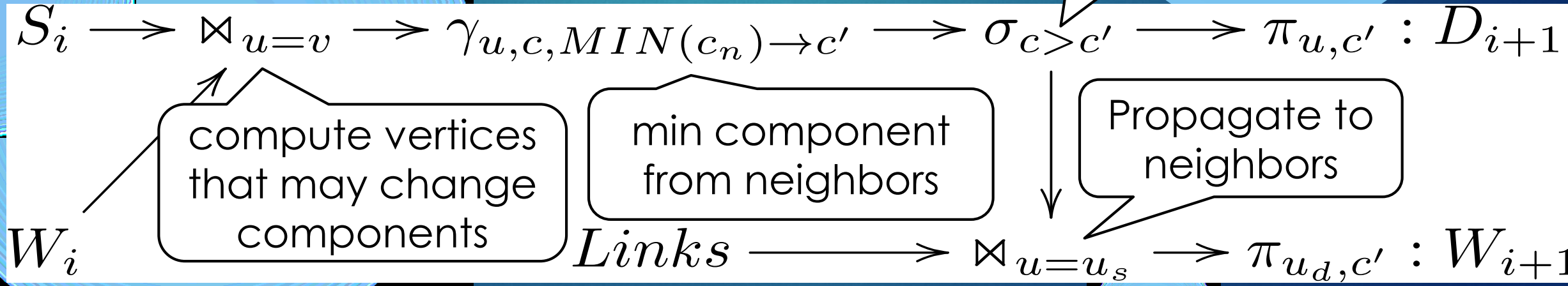
- Dataflow is a cyclic DAG of operators/UDFs
- An iterative part is specified as (Δ , S_0 , W_0)
- Δ: dataflow for the body of the iteration

$$(D_{i+1} , W_{i+1}) = \Delta(S_i , W_i)$$

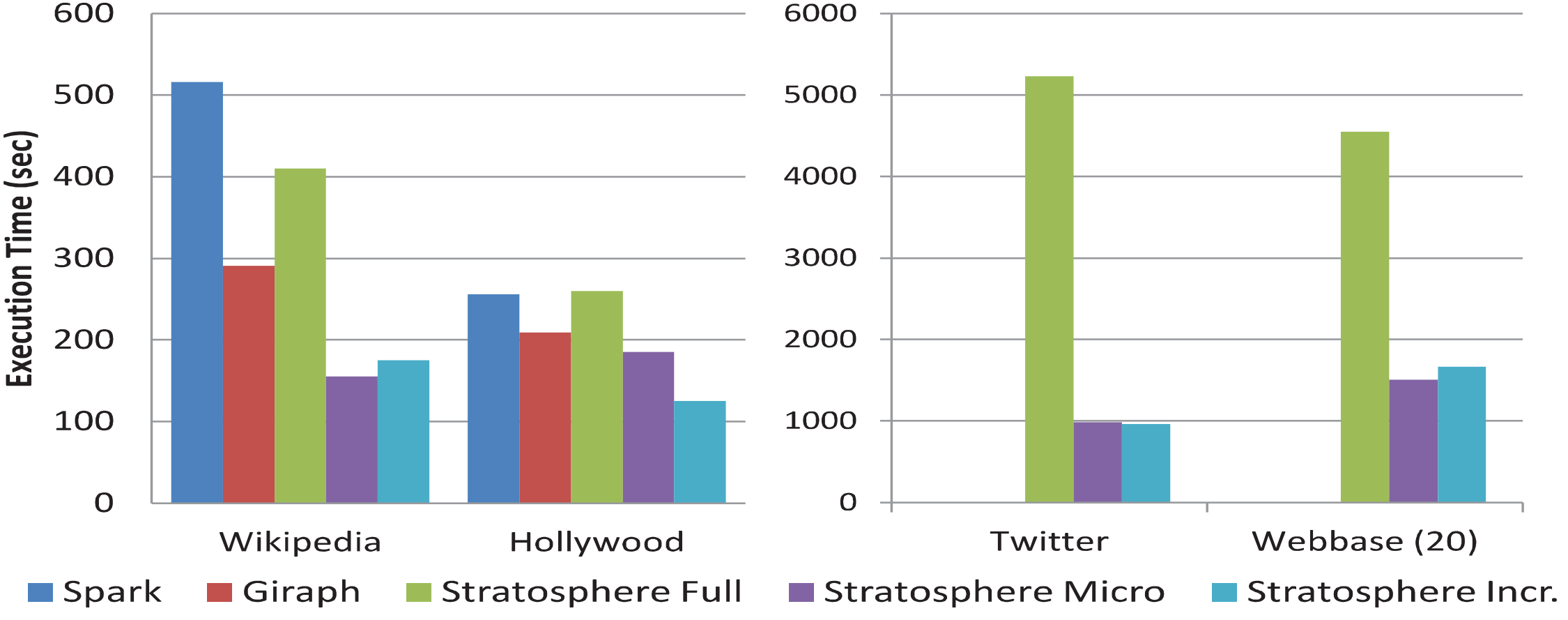


Example: CC

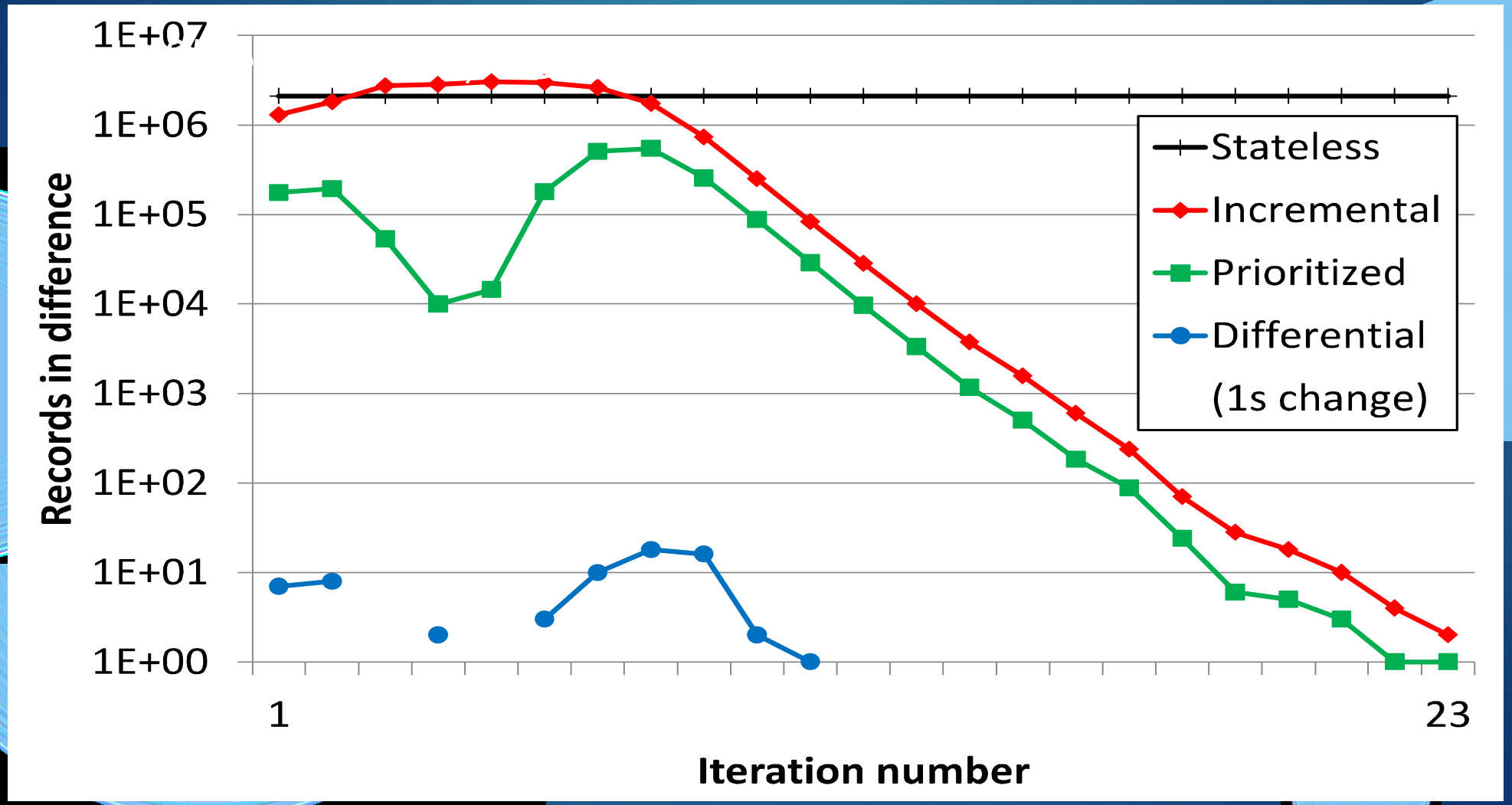
- S_i = { (*u*, *c*): *u* is a vertex, *c* is a component id }
- W_i = { (*v*, c_n): *v* is a vertex, c_n is the new component id of some neighbor *u*}
- Links = { (*u*,*v*): *u* is connected to *v* }



Execution in Stratosphere



Naiad: Differential Dataflow



Comparison

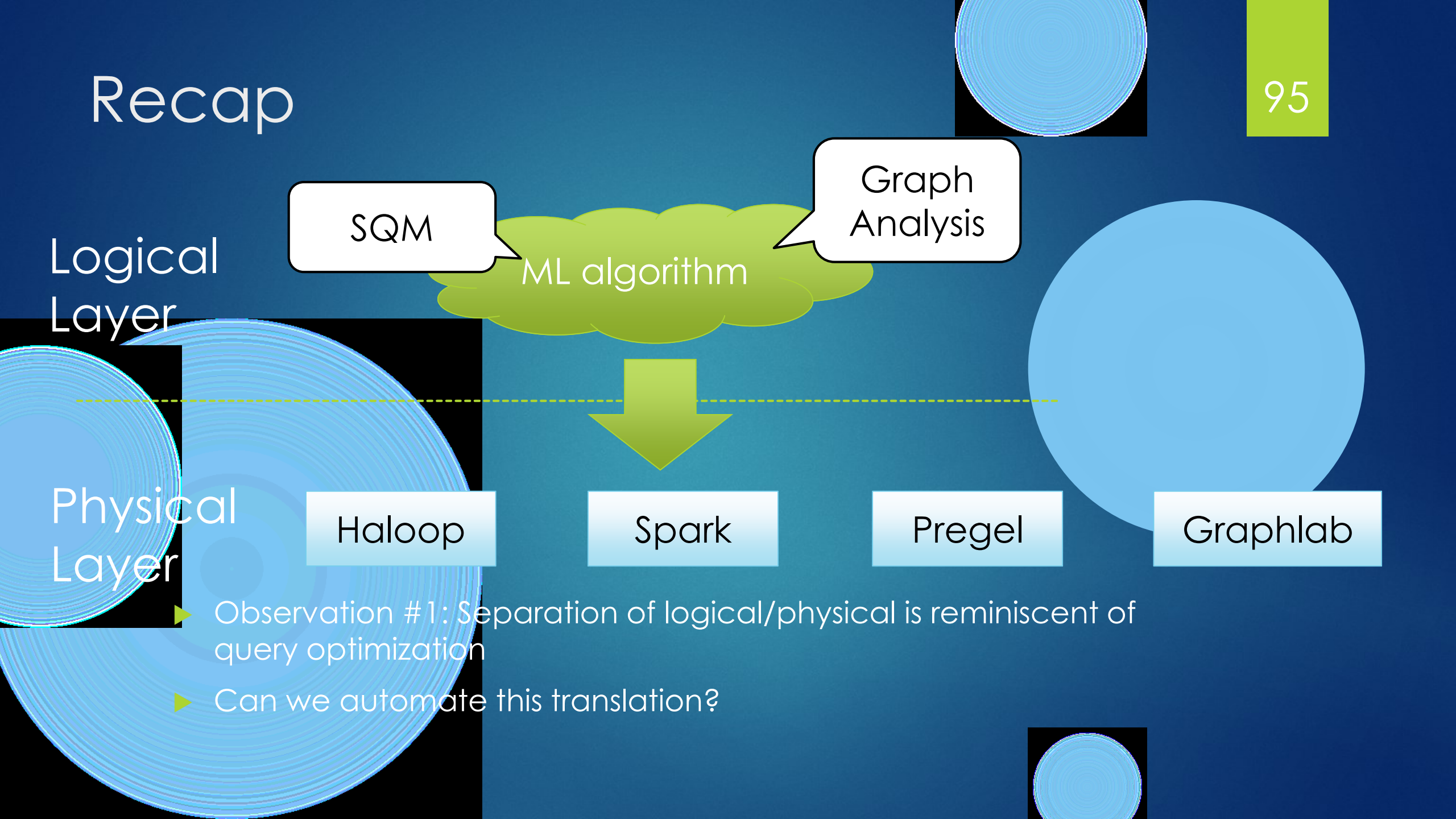
	MapReduce	Pregel	Stratosphere/Naiad	GraphLab
Programming Model	Fixed Functions – Map and Reduce	Supersteps over a data graph with messages passed	Iterative dataflow with operators and UDFs	Data graph with shared data table and update functions
Parallelism	Concurrent execution of tasks within map and reduce phases	Concurrent execution of user functions over vertices within a superstep	Concurrent execution of operators during a stage	Concurrent execution of non-overlapping scopes, defined by consistency model
Data Handling	Distributed file system	Distributed file system	Flexible data channels: Memory, Files, DFS etc.	Undefined – Graphs can be in memory or on disk
Task Scheduling	Fixed Phases – HDFS Locality based map task assignment	Partitioned Graph and Inputs assigned by assignment functions	Job and Stage Managers assign operators to available daemons/tasks	Pluggable schedulers to schedule update functions
Fault Tolerance	DFS replication + Task reassignment / Speculative execution of Tasks	Checkpointing and superstep re-execution	Operators/Task failure recovery	Synchronous and asychronous snapshots
Developed by	Google	Google	TU Berlin / Microsoft	Carnegie Mellon

Other Efforts

- REX: Similar to Stratosphere
- Madlib: SQL library for matrix multiplication
- VW: Specialized tool for linear models
- DistBelief: Specialized system for NN learning
- Mahout: A library of ML algorithms over Hadoop
- Hyracks*

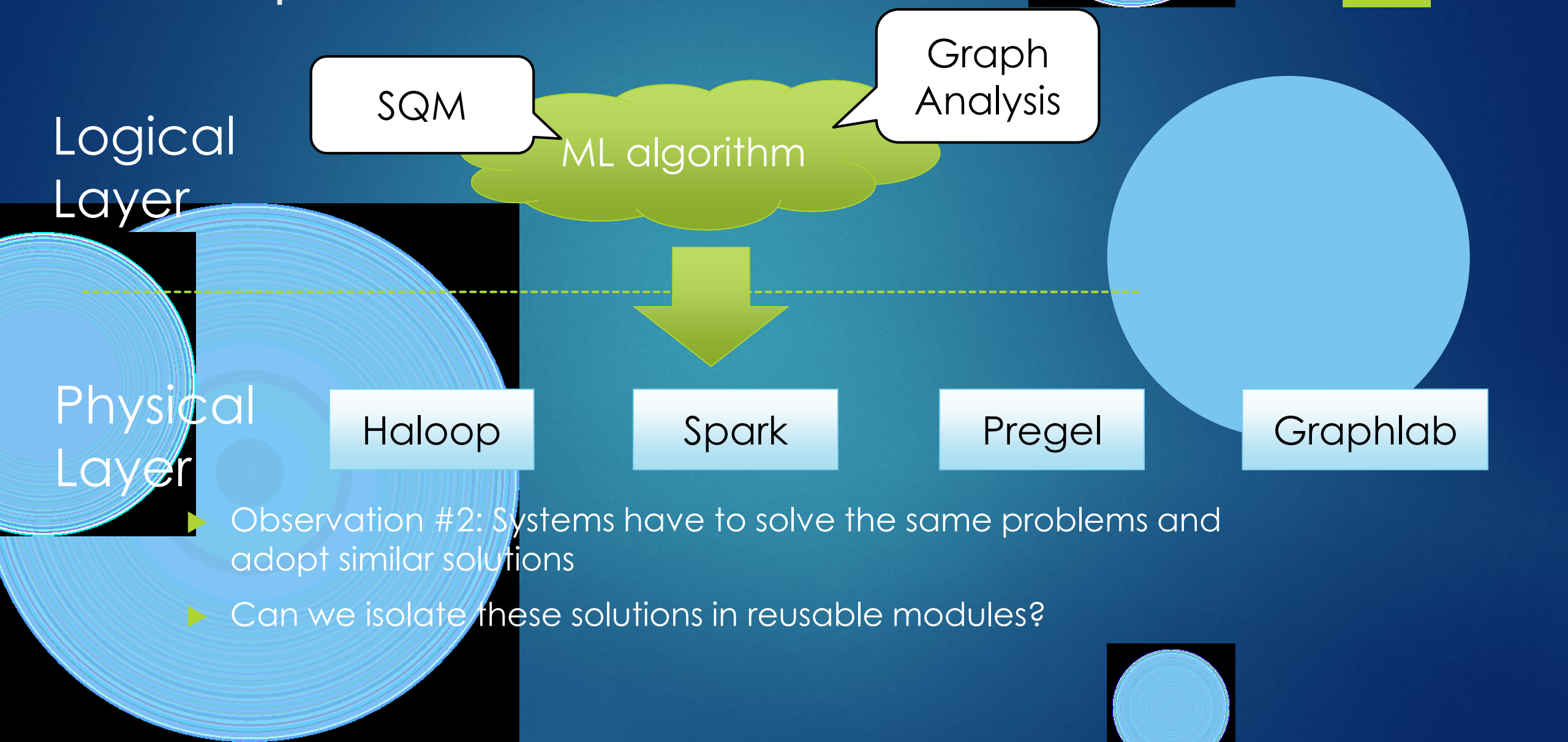
Conclusions and Outlook

Recap



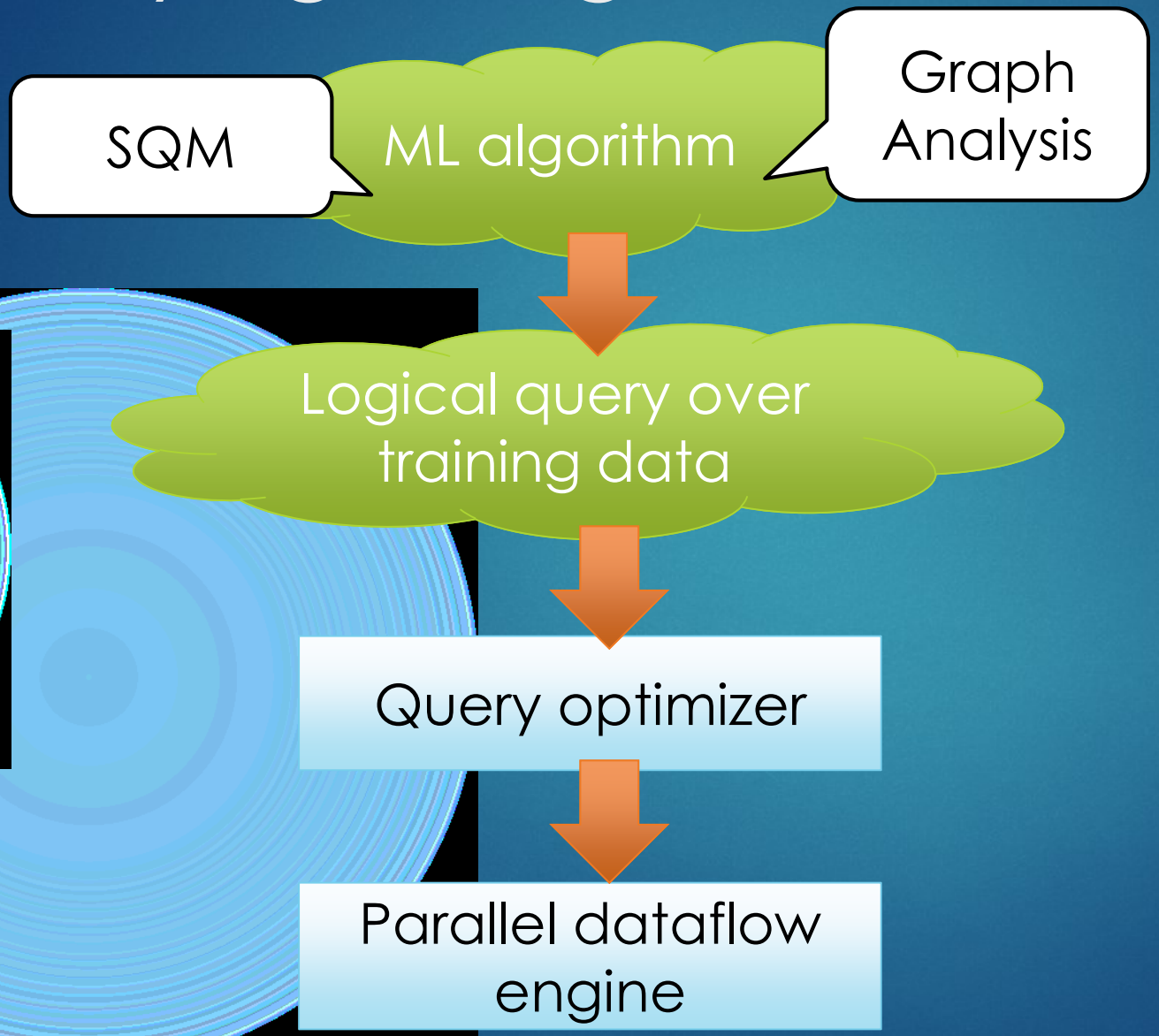
- Observation #1: Separation of logical/physical is reminiscent of query optimization
- Can we automate this translation?

Recap

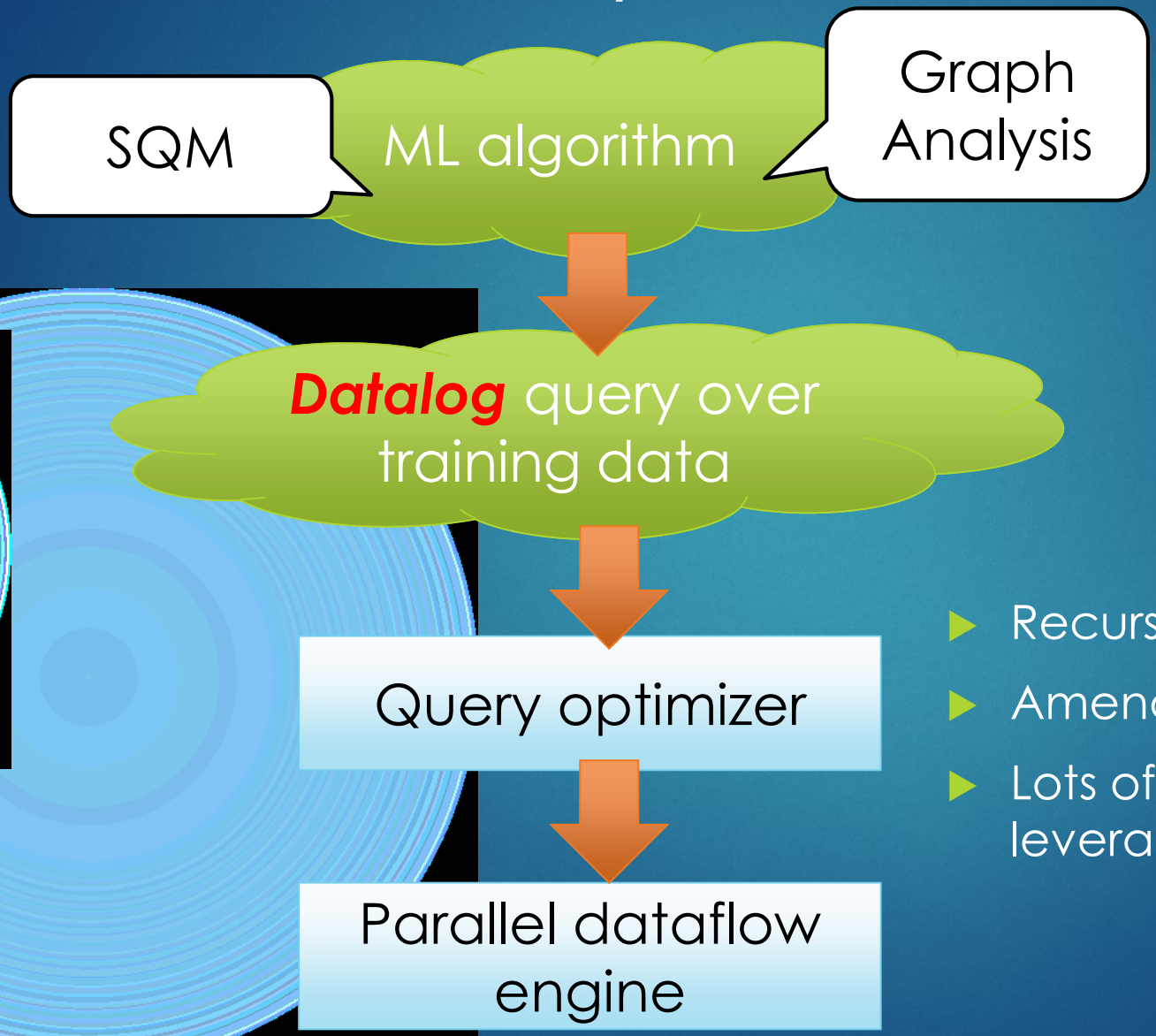


- Observation #2: Systems have to solve the same problems and adopt similar solutions
- Can we isolate these solutions in reusable modules?

A Unifying Design



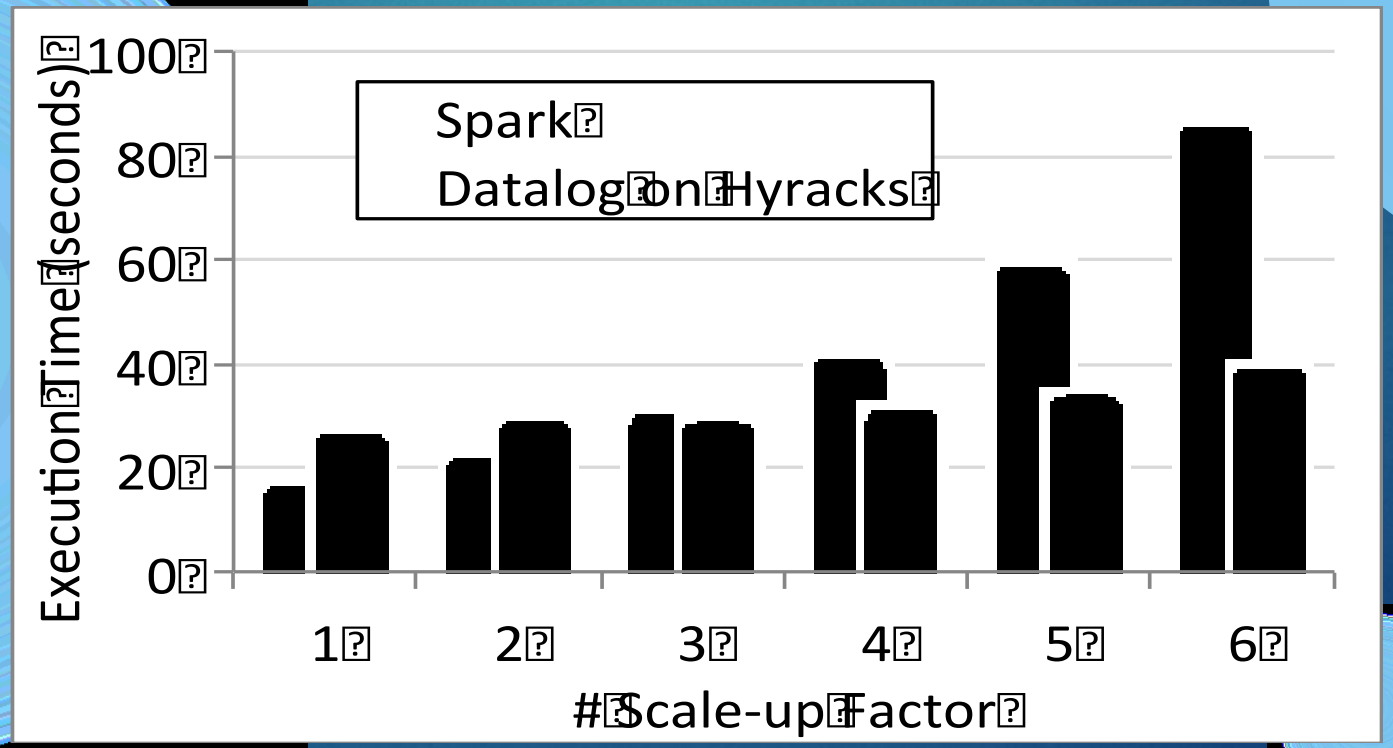
A Concrete Proposal



- Recursion is built-in
- Amenable to optimizations
- Lots of existing work that we can leverage

Version 0.1

- Implementation over Hyracks
- Supports both Iterative-MRU and Pregel
- Standard optimizations + some new tricks



Open Research Questions

- Iteration-aware query processing
  - Cost estimation for recursive computation
  - Cost models (time vs money)
  - Late- vs. early-stage processing
  - Effective handling of UDFs
- Computational models for graph analysis
- Provenance for triage
  - "My model misbehaves – why?"
- Tuning -- who is the "DBA"?
- Fault-awareness as a logical concept
  - Not all faults are catastrophic for ML
  - Algorithm-specific fault-tolerance
- Incremental learning

Thank you!

QUESTIONS?