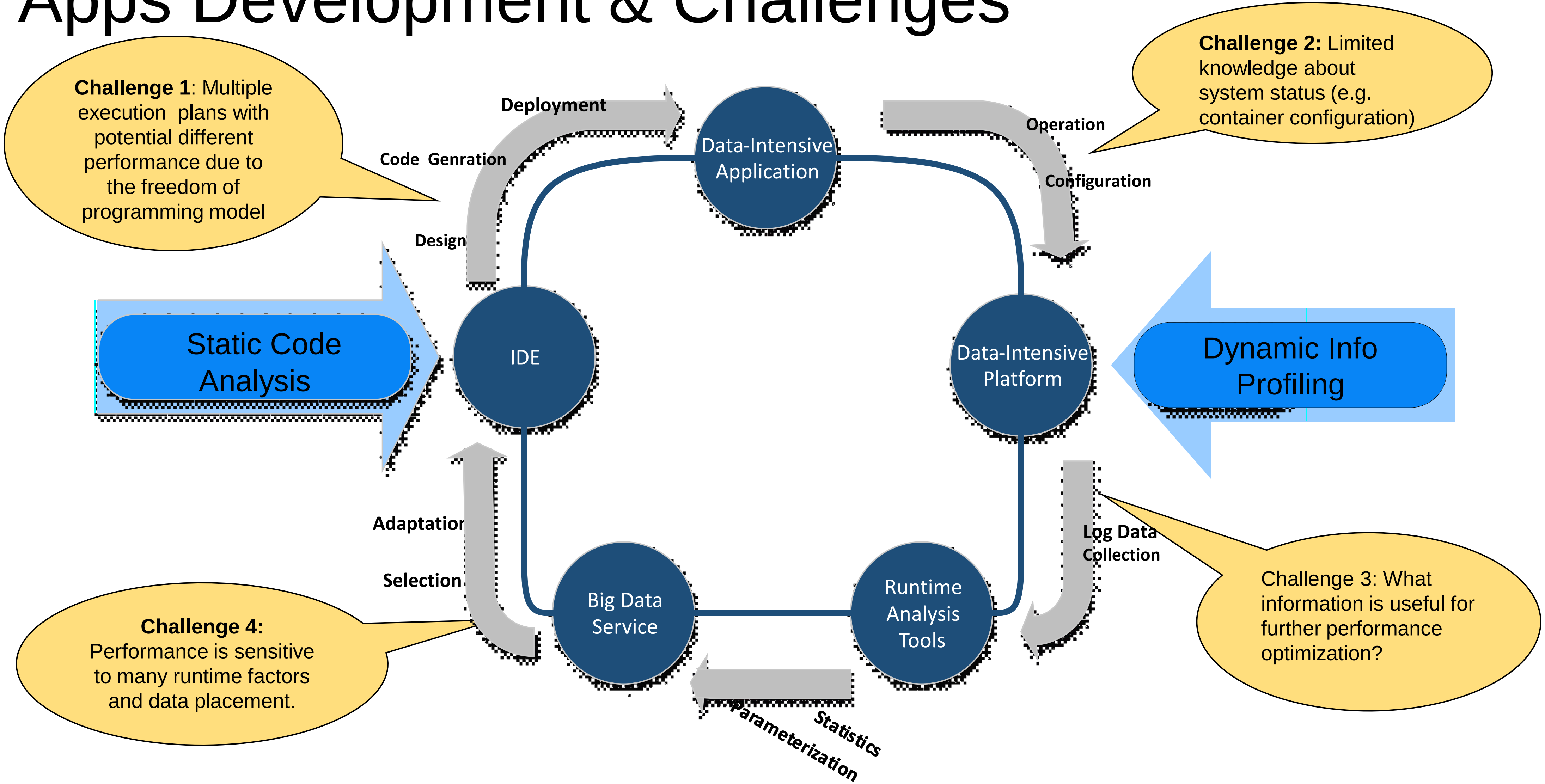
A Semantics-Aware Optimization Framework for Data-Intensive Applications Using Hybrid Program Analysis

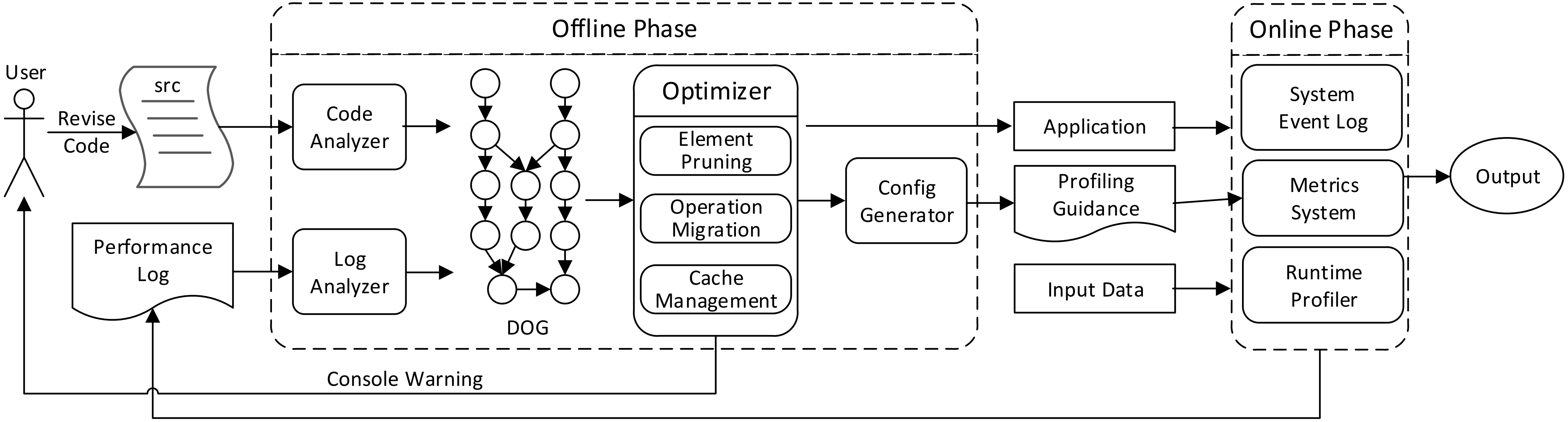
Liqiang Wang
Department of Computer Science
University of Central Florida

Apps Development & Challenges



Our approach

- A two-stage framework, offline and online stages, to assist programmers to design and optimize data-intensive applications semi-automatically.



The full life cycle of semantics-aware optimization approach for data-intensive applications

Semantics-Aware Data Model for Spark

- An abstract data model associated with semantic context regarding code, data and system to represent skeleton of an application and track evolution of dataset(s)
  - **Data Representation**: Attribute-Based Data Abstraction
  - **Data Manipulation**: Predefined Primitive Operations
  - **Application Representation**: Data Operational Graph (DOG)
- Operation Strategies
  - Element Pruning (remove redundant attributes in an element)
  - Operation Reordering (Filter pushdown)
  - Cache Management (persist a data block in memory)

Primitive Operations

- Define six primitive operations to abstract behaviors of a general data-intensive system, as shown in following Table

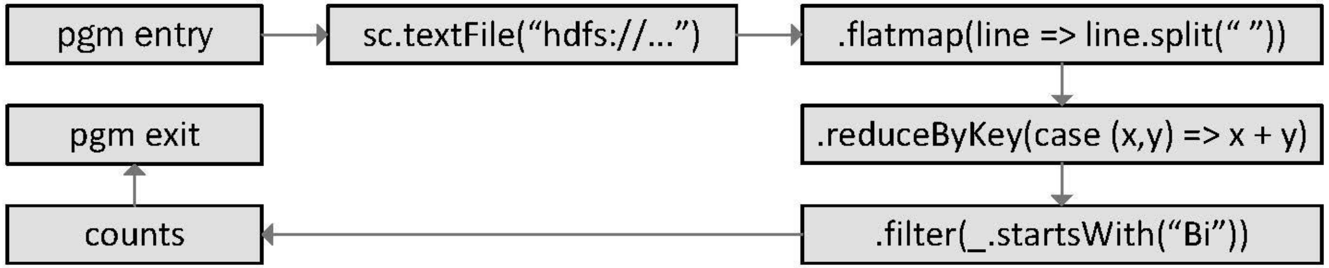
Operation	Notation	Examples in Apache Spark
Map	$Map : X \times f \mapsto Z$	`map, flatmap, mapValues,mapPartions`
$Filter$	$Filter : X \times f \mapsto Z$	`filter, sample, collect`
Set	$Set : X \times Y \times f \mapsto Z$	`++, intersection, union`
$Join$	$Join : X \times Y \times f \times K \mapsto Z$	`join, leftOuterJoin, rightOuterJoin, fullOuterJoin`
$Group$	$Group : X \times f \times K \mapsto Z$	`reduceByKey, groupByKey, aggregateByKey, foldByKey`
Agg	$Agg : X \times f \times init \mapsto reg$	`reduce, aggregate, fold, max, min`

Table 1: The Definition of Primitive Operations, where X, Y, Z represent datasets and reg is a returned value. f are UDFs working on one or a group of elements. The key K is a subset of attributes shared by two or more datasets, and $init$ is the initial value for aggregate operations. The last column lists representative operations for each category provided by Apache Spark RDD APIs.

Data Operational Graph (DOG)

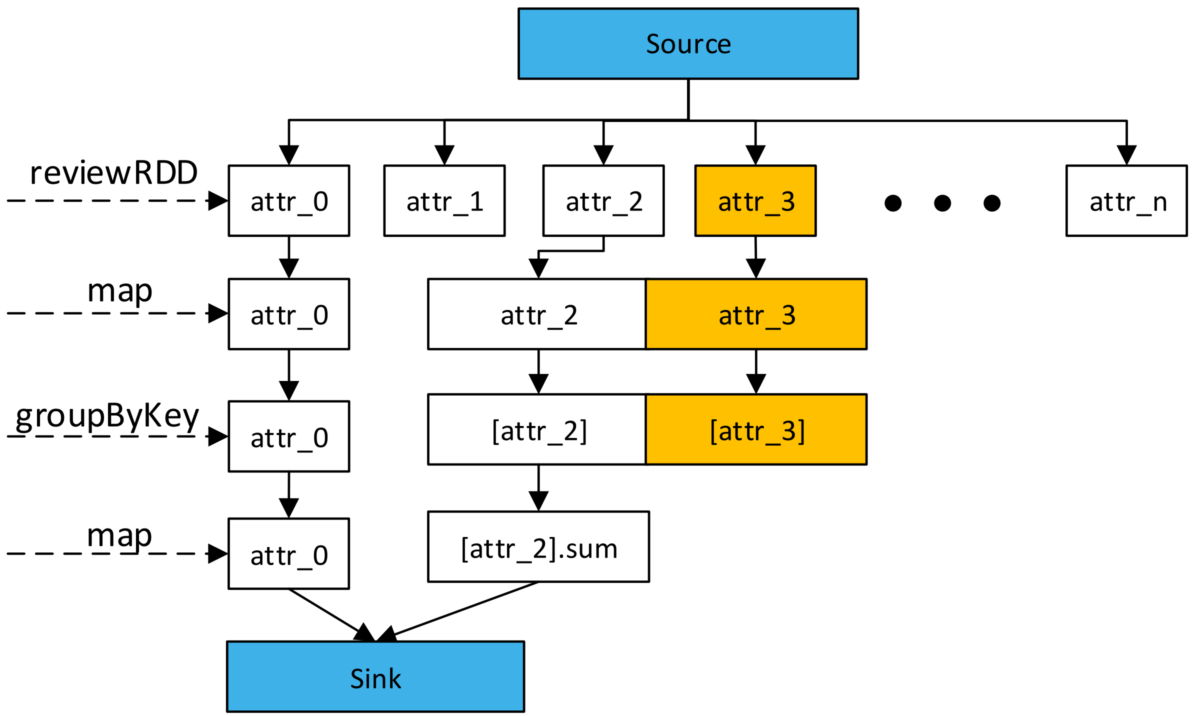
- A directed graph G = (V, E)
  - V: Data manipulated operations and the corresponding generated datasets
  - E: Data flows between operations
  - Semantics knowledge regarding code, data and system is attached to vertices and edges

```
val counts = sc.textFile("hdfs://...")
                 .flatmap(line => line.split(" "))
                 .map(word => (word, 1))
                 .reduceByKey(case (x,y) => x + y)
                 .filter(_._1.startsWith("Bi"))
```



Element Pruning

- It is a static optimization to eliminate unused attributes in an element by analyzing data dependency in the attribute level among operations.
  - Analyzing attribute dependency between the input and output dataset of an operation and its UDF
  - Building a directed data dependency graph (DDG) to represent the whole data flow of the application
  - Removing nodes that does not make contributions to output of the application



Operation Reordering

- Improve applications' performance by reordering operations along with data path, e.g. Filter Pushdown.
  - Statically ensuring identical semantics
  - Evaluating performance improvement using dynamic/profiling information
    - Using performance models to predict and evaluate performance behavior after reordering.

Cache Management

- Spark RDD cache/release
- Cache management determines a policy on the stage level to balance the gain and overhead of caching each RDD.
  - A very complex problem
  - We designed an approach based on convex optimization
    - Dataset size affects memory capacity and perf behaviors.
    - Executing order affects the performance

Experiment and Conclusion

Bechmark	Element Pruning (EP)	Operation Reordering (OR)	Cache Management (CM)
SLA	1.55%	0.77%	2.07%
CRA	6.38%	3.09%	59.57%
SNA	6.15%	9.70%	-7.88%
PPJ	7.47%	0.24%	2.96%

System speed up of individual optimization over the baseline implementation in RDD.

- Our semantic-aware optimization approach including two stages:
  - Static stage (Offline): code operation
  - Dynamic stage (online): customized profiling