SPARK+AI SUMMIT 2018

TuneIn: How to get your jobs tuned while sleeping

Manoj Kumar, LinkedIn
Arpan Agrawal, LinkedIn

#Res2SAIS

LinkedIn

OUR VISION

Create economic opportunity for every member of the global workforce

OUR MISSION

Connect the world’s professionals to make them more productive and successful

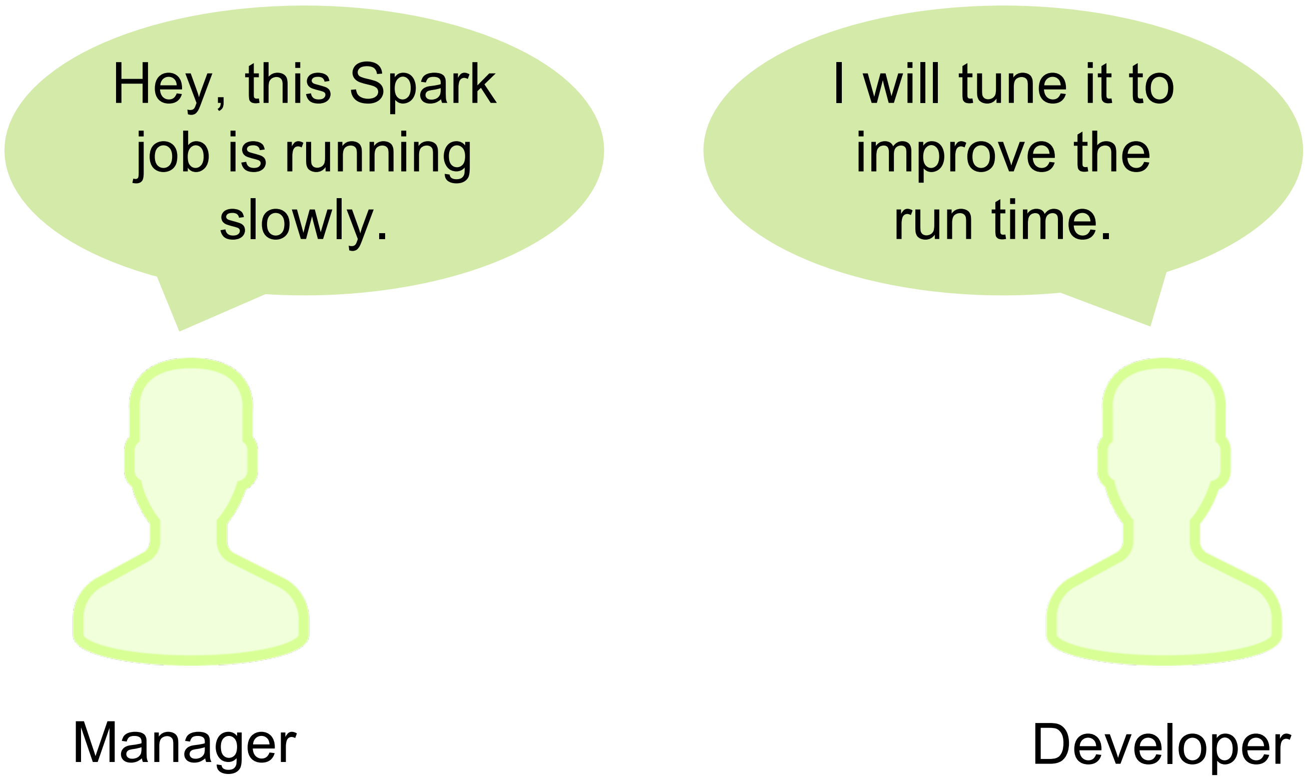
Agenda

- Why TuneIn?
- How does TuneIn work?
- Architecture and framework features
- Road ahead

Grid Scale at LinkedIn

2008	2018
1 cluster	10+ clusters
20 nodes	1000s of nodes
5 users	1000s of active users
MapReduce	Pig, Hive, Spark, etc.
Few workflows	10000s workflows

Typical Conversations



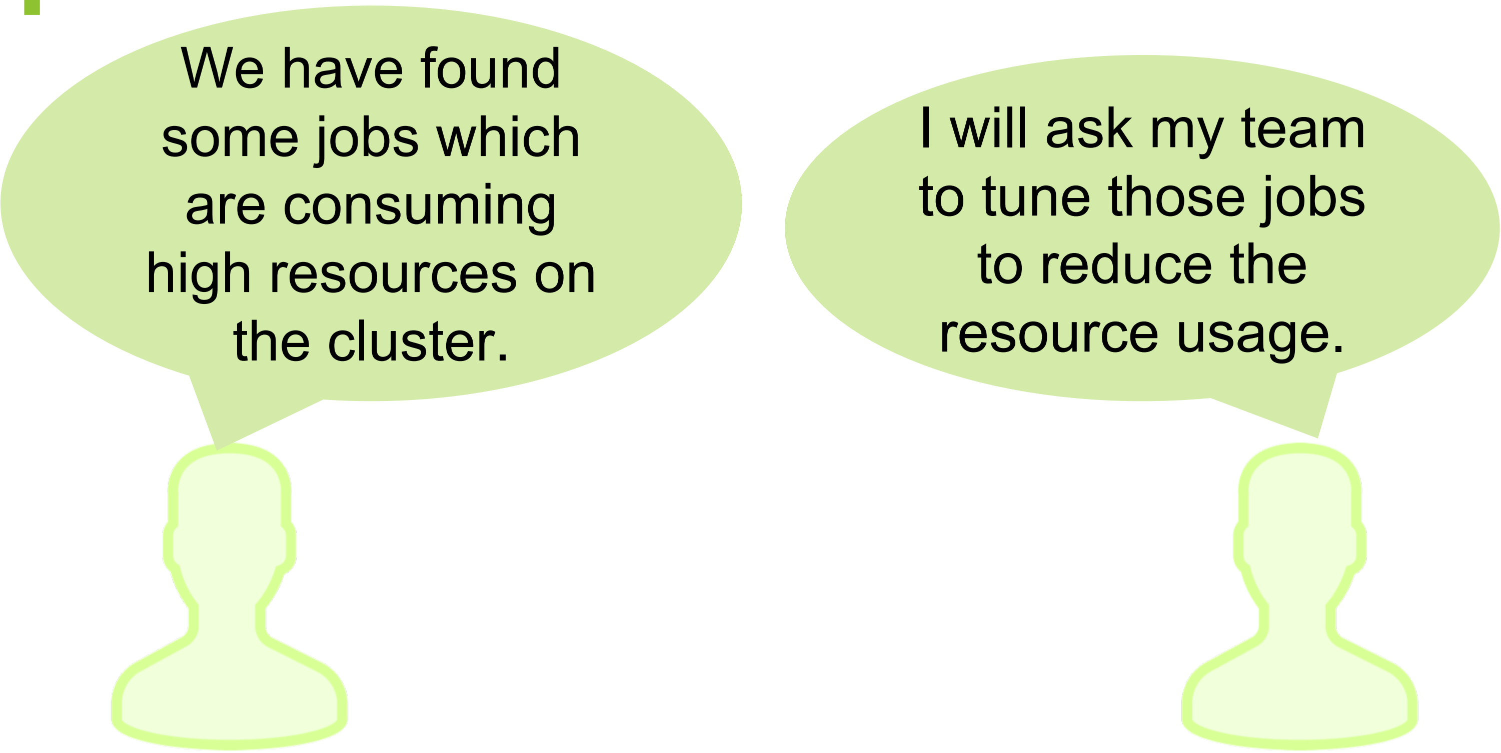
Hey, this Spark job is running slowly.

I will tune it to improve the run time.

Manager

Developer

Typical Conversations



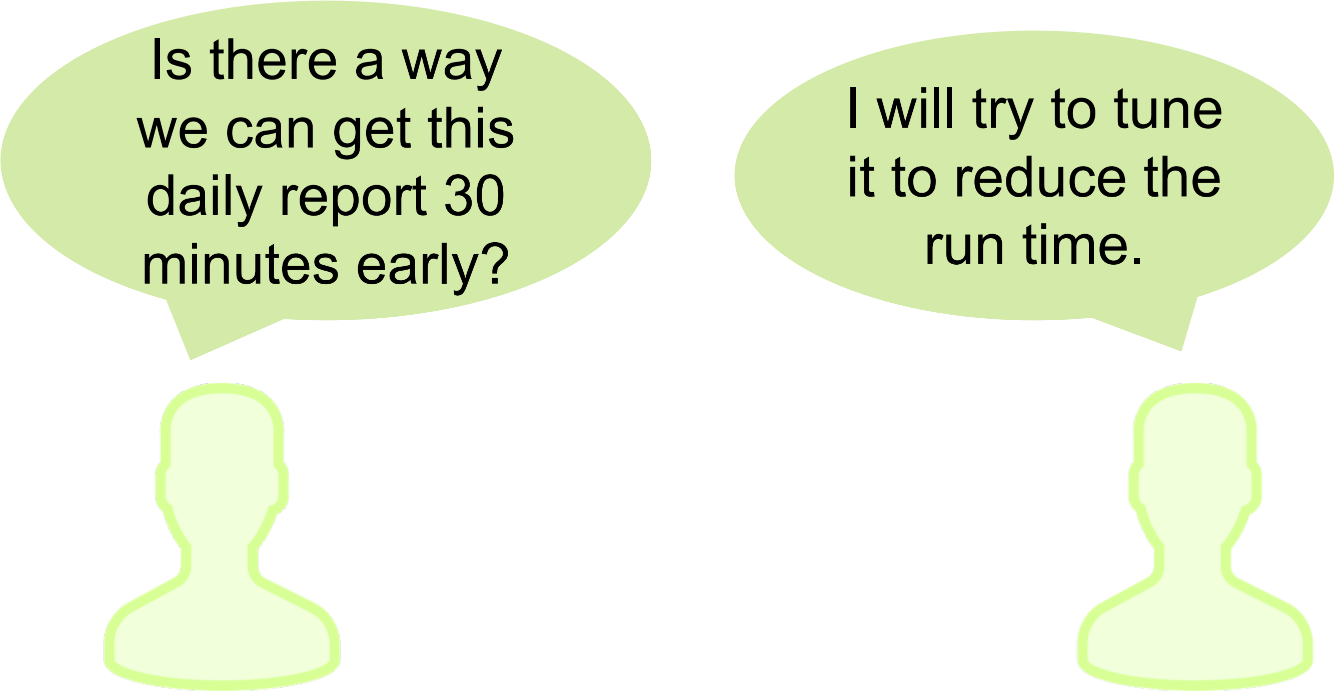
We have found some jobs which are consuming high resources on the cluster.

I will ask my team to tune those jobs to reduce the resource usage.

Hadoop Admin

Manager

Typical Conversations



Is there a way we can get this daily report 30 minutes early?

I will try to tune it to reduce the run time.

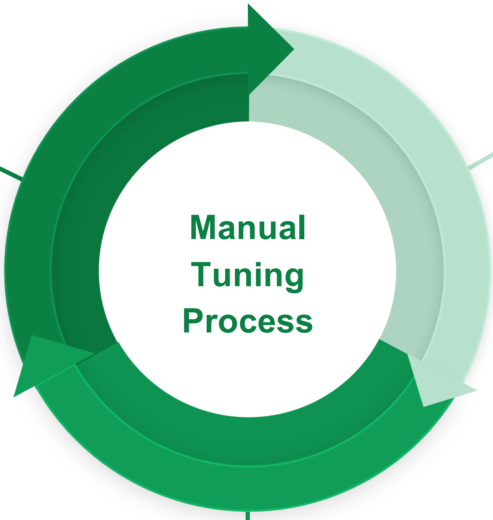
Client

Developer

Why Tuning?

- Optimal parameter configuration:
  - leads to better cluster utilization and thus savings
  - reduces the execution time
- Default configuration is not always optimal

Manual Tuning



Manual Tuning Process

PHASE 1

Execute Job

PHASE 2

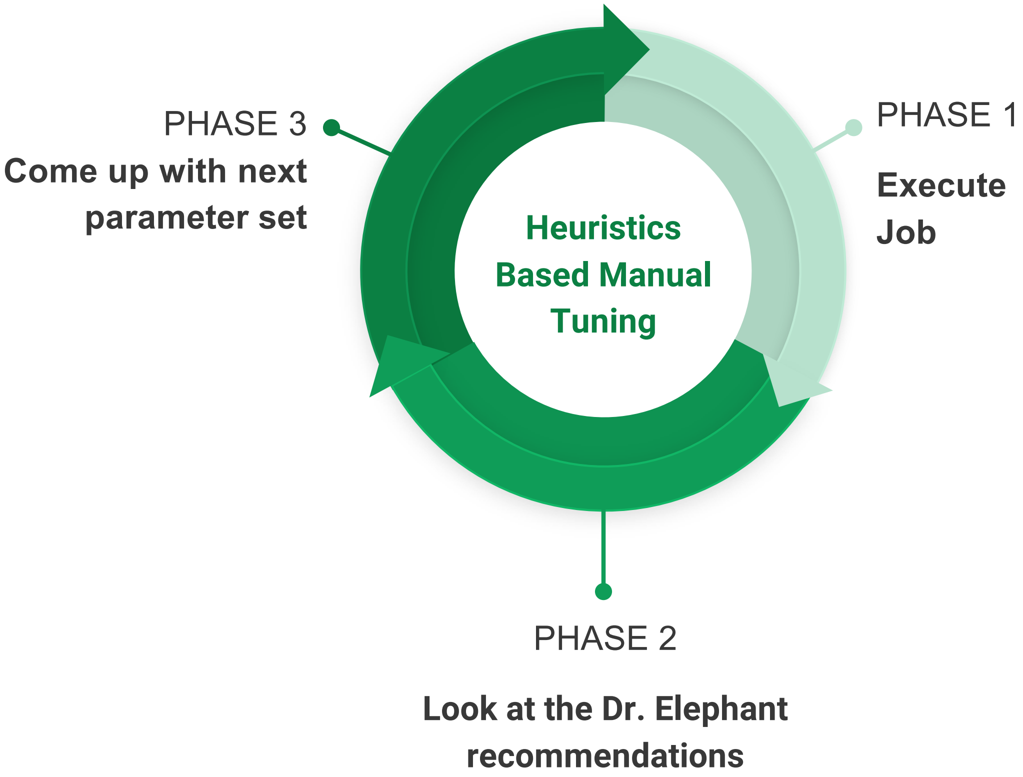
Observe the Execution Metrics

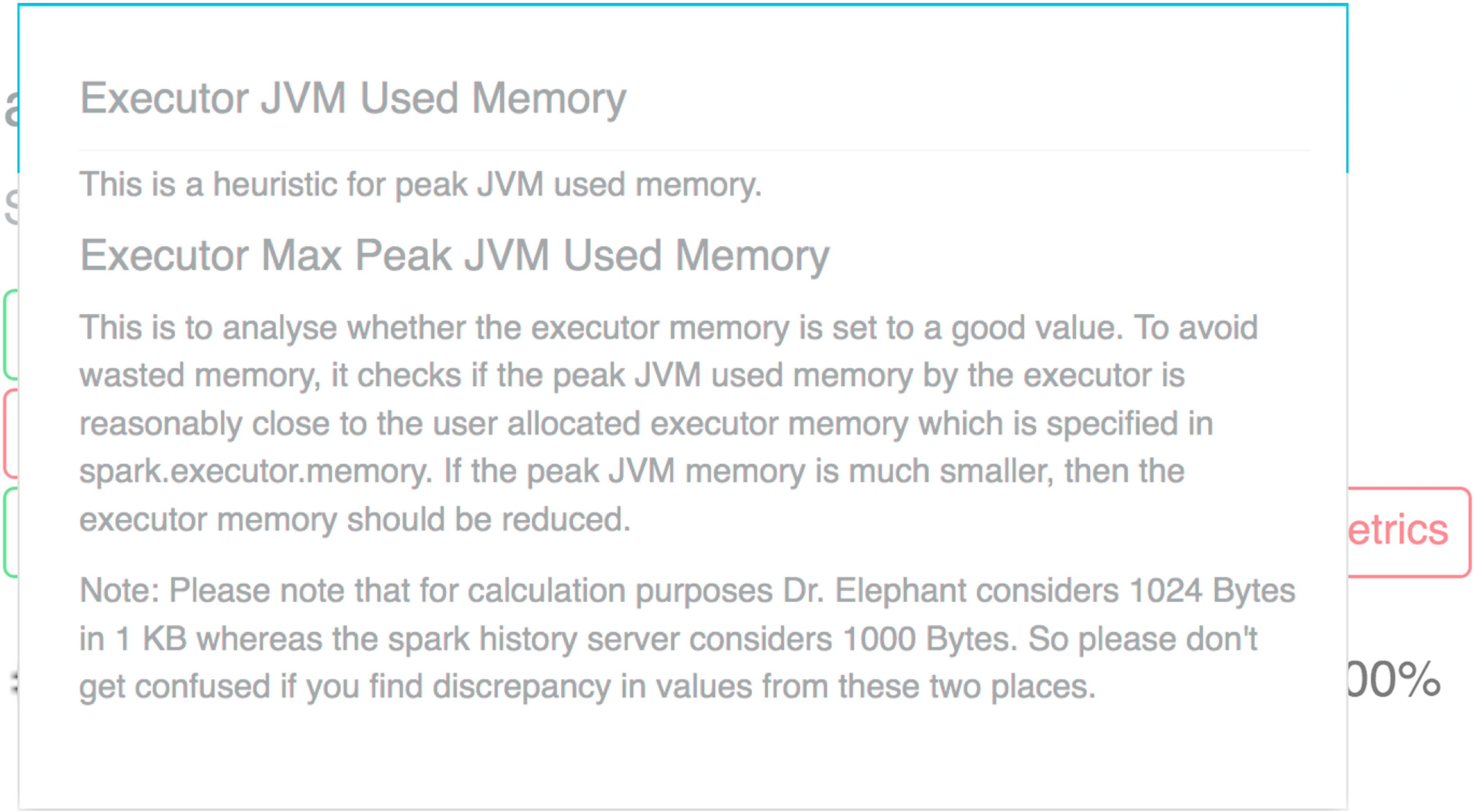
PHASE 3

Come up with next parameter set

Dr. Elephant: Heuristic based tuning

- Suggests tuning recommendations based on pre-defined heuristics
- No need to worry about the hundreds of counters and parameters
- Relies on user's initiative to use the recommendations
- Expects some user expertise


Heuristics Based Manual Tuning
PHASE 1
Execute Job
PHASE 2
Look at the Dr. Elephant recommendations
PHASE 3
Come up with next parameter set



Executor JVM Used Memory

This is a heuristic for peak JVM used memory.

Executor Max Peak JVM Used Memory

This is to analyse whether the executor memory is set to a good value. To avoid wasted memory, it checks if the peak JVM used memory by the executor is reasonably close to the user allocated executor memory which is specified in spark.executor.memory. If the peak JVM memory is much smaller, then the executor memory should be reduced.

Note: Please note that for calculation purposes Dr. Elephant considers 1024 Bytes in 1 KB whereas the spark history server considers 1000 Bytes. So please don't get confused if you find discrepancy in values from these two places.

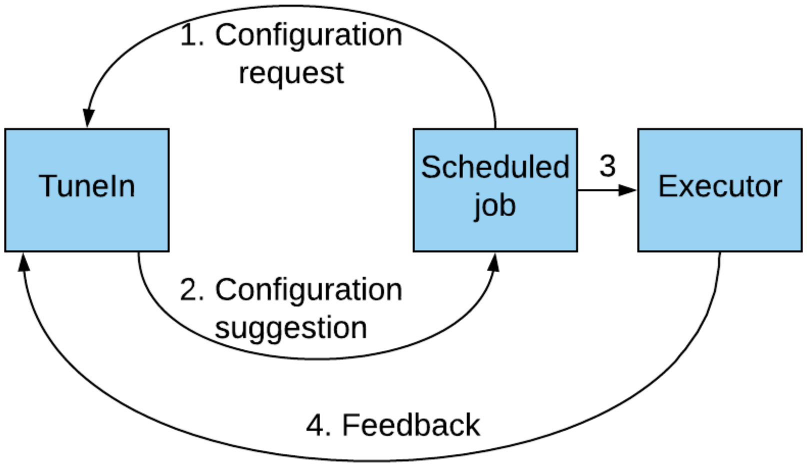
Why Auto Tuning?

- 10000s of jobs to tune
- Increases developer productivity
- Tunes without any extra effort
- No expertise is expected
- Option of which objective function to tune for
  - resource usage
  - execution time etc.

Let’s auto tune!

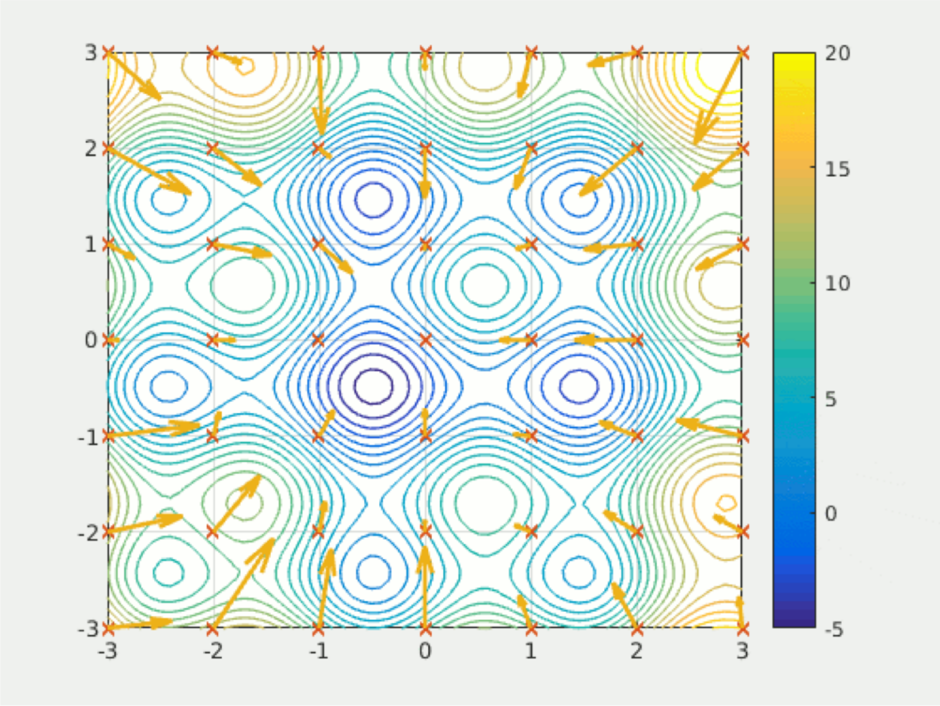
TuneIn

- Framework to automatically tune recurring Hadoop and Spark jobs
- Iteratively tries to reach the optimal configuration
- Results : 20-35% reduction in Resource Usage



Particle Swarm Optimization (PSO)[1]

- Mimics the behavior of swarm of birds searching food
- Introduces a population of candidate solution particles in the search space

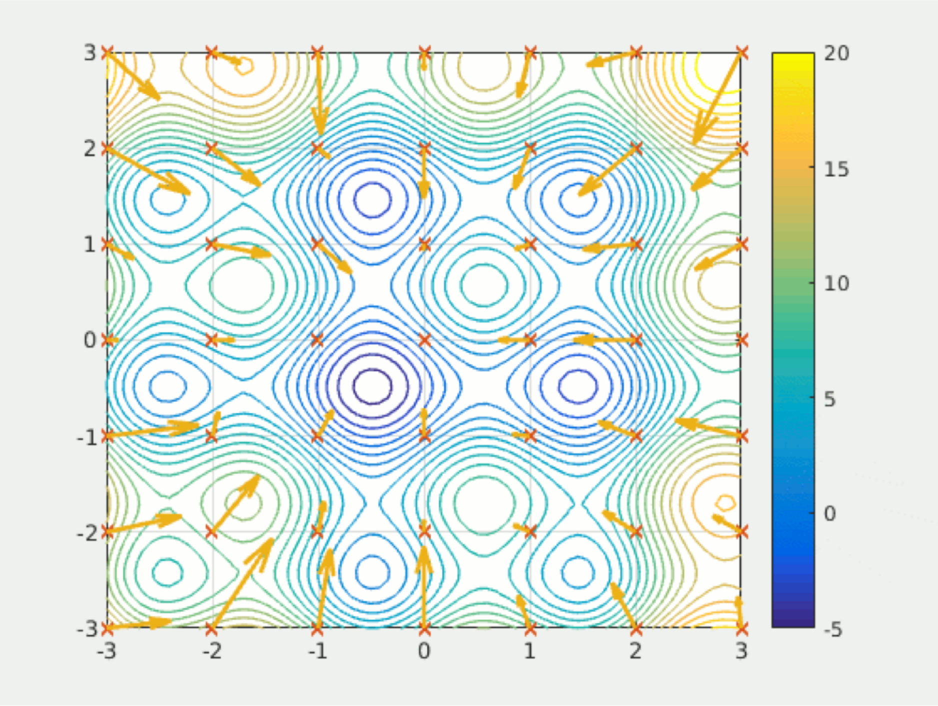


Source: Wikipedia

Particle Swarm Optimization by J. Kennedy et al., https://ieeexplore.ieee.org/document/488968/

PSO (contd.)

- Points of attraction: personal and swarm's best known positions
- Particles converge to the region with the minimum cost function value



Source: Wikipedia

Why PSO?

- Cost function is noisy
  - PSO is gradient free and robust to noise [3]
- Spark and Hadoop are complex systems
  - PSO is a metaheuristic black box optimization algorithm
- Fastest convergence

K. E. Parsopoulos et al., “Particle Swarm Optimizer in Noisy and Continuously Changing Environments,” in Artificial Intelligence and Soft Computing

PSO Details[2]

- Swarm size of 3 gives the best result
  - neither too small to cover the search space
  - nor too big to do many first iteration random searches
- Good starting point is important to guide the swarm

Optimizing Hadoop parameter settings with gene expression programming guided PSO by Mukhtaj Khan et al.

Cost function

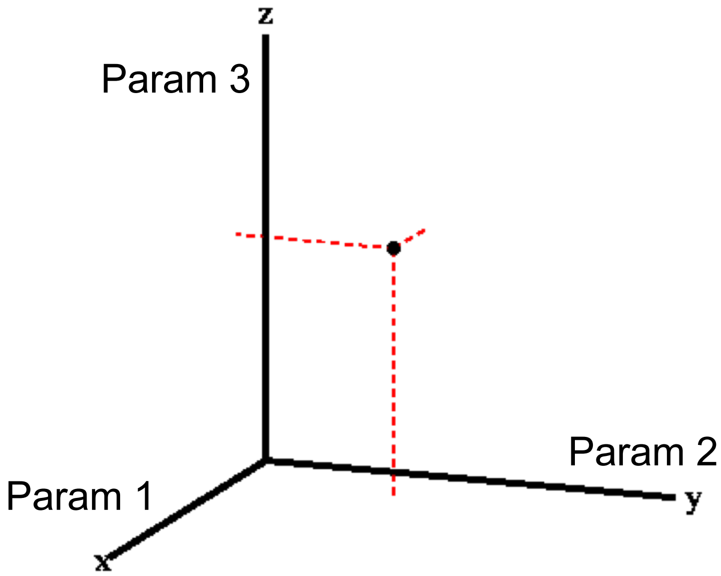
- Resource usage per unit input

$$\frac{\sum_{Containers} Container\ Memory\ * Container\ Uptime}{Total\ Input\ Size}$$

- Approximately input size invariant

Search Space

- Parameters being tuned constitutes the search space
- Parameters to tune depends on the cost function metric



Search Space

Cost function: Resource Usage	
Pig	**Spark**
mapreduce.map.memory.mb	spark.executor.memory
mapreduce.reduce.memory.mb	spark.executor.cores
mapreduce.task.io.sort.mb	spark.memory.fraction
mapreduce.task.io.sort.factor	spark.yarn.executor.memoryOverhead

Search Space Optimization

- Important to prevent failures
- Speeds up convergence
- Boundary parameter values
  - e.g. $spark.executor.cores \in [1, 10]$
- Parameter interdependent constraints
  - Captures the interdependence among the parameters
  - e.g. $mapreduce.task.io.sort.mb < 0.60 * mapreduce.map.memory.mb$

Avoiding over optimization

- Undesirable to squeeze memory so much that execution time shoots up significantly
- Updated cost function:

$$\frac{\sum_{Containers} Container\ Memory\ * Container\ Uptime}{Total\ Input\ Size} + Penalty$$

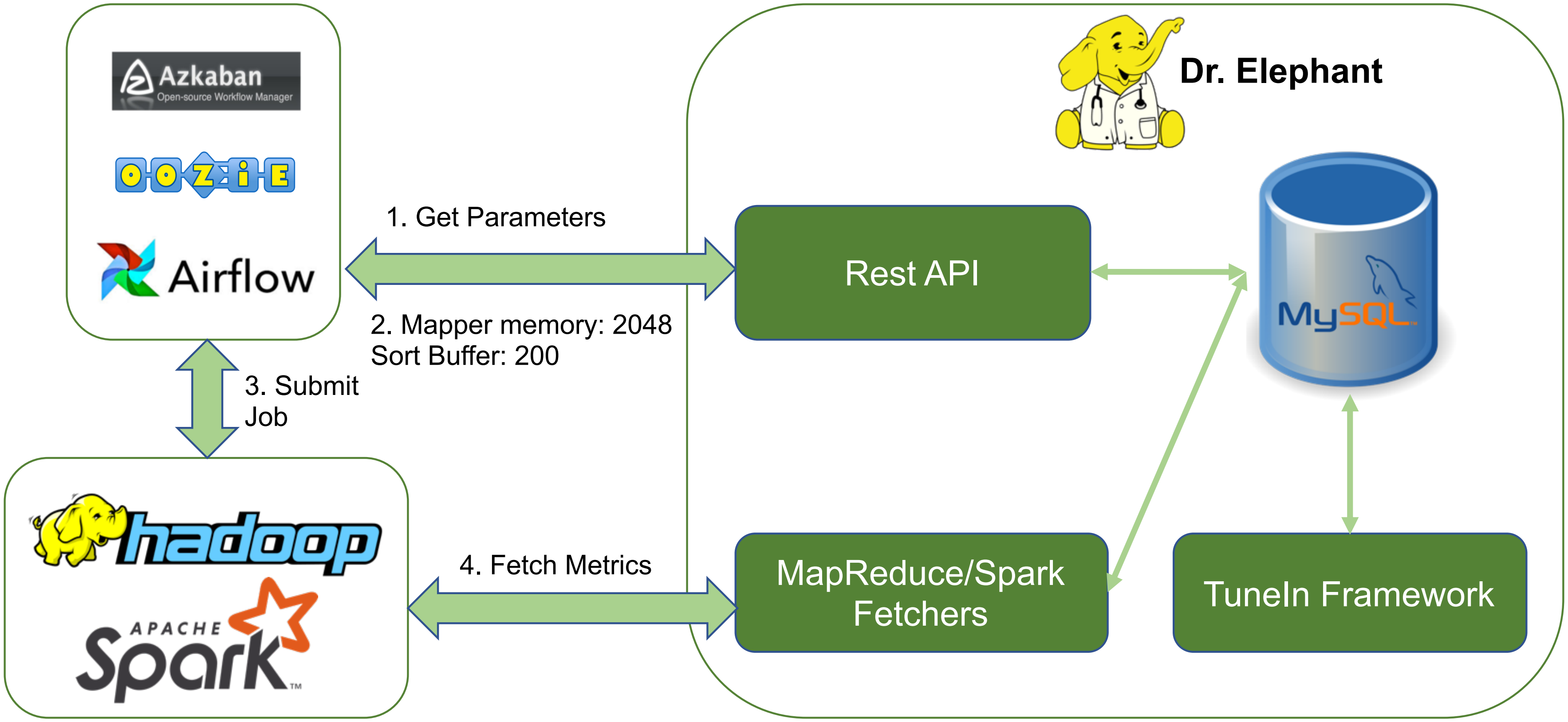
Convergence

- No theoretical bound on the steps to converge
- Practically converges in 20 job executions
- TuneIn gets turned off for the job automatically on convergence

Results

Job type	Metric	Average reduction
Spark	Resource Usage	30 - 40 % per job
Pig	Resource Usage	20 - 35 % per job

Architecture



Framework Features

Tuning During Regular Scheduled Runs

Generic Framework

- Resource Usage, Execution Time
- Pig, Hive, Spark
- Easy Integration

Failure Avoidance

- Constraints on parameters
- Automatic Failure Handling

Auto Switch Off

Road Ahead

- Tuning for execution time
- Faster convergence using Intelligent Parameter Space Optimization (IPSO)
- Smarter tuning switch on/off

References

1. Particle Swarm Optimization by J. Kennedy et al., https://ieeexplore.ieee.org/document/488968/
2. Optimizing Hadoop parameter settings with gene expression programming guided PSO by Mukhtaj Khan et al.
3. K. E. Parsopoulos et al., “Particle Swarm Optimizer in Noisy and Continuously Changing Environments,” in Artificial Intelligence and Soft Computing

Happy tuning!

Document: https://github.com/linkedin/dr-elephant/wiki/Auto-Tuning
Code: https://github.com/linkedin/dr-elephant/pull/338

Appendix

Algorithms experimented with

- Gradient descent
  - Brute force
  - Simultaneous perturbation
- Gradient free methods
  - Maximum likelihood region
  - Genetic algorithm
  - Differential evolution

Pig Interdependent Constraints

$$mapreduce.task.io.sort.mb < 0.6 * mapreduce.map.memory.mb$$

$$mapreduce.map.memory.mb - mapreduce.task.io.sort.mb > 768$$

$$pig.maxCombinedSplitSize < 1.8 * map.memory.mb$$

Penalty Function

$$Penalty\ = 3 * \frac{\max Desired\ Resource\ Usage}{Average\ Input\ Size}$$