PayPal Risk Infra Tech Optimization Practices

Li, Bruce, AERIS Architect

QCON Shanghai, Oct 2018

CNUTCon 全球运维技术大会 [上海]

主办方 Geekbang> 极客邦科技 InfoQ

50+ 年末充电

开发&运维技术干货大盘点

容器 | Kubernetes | DevOps | 全链路压测 | Severless

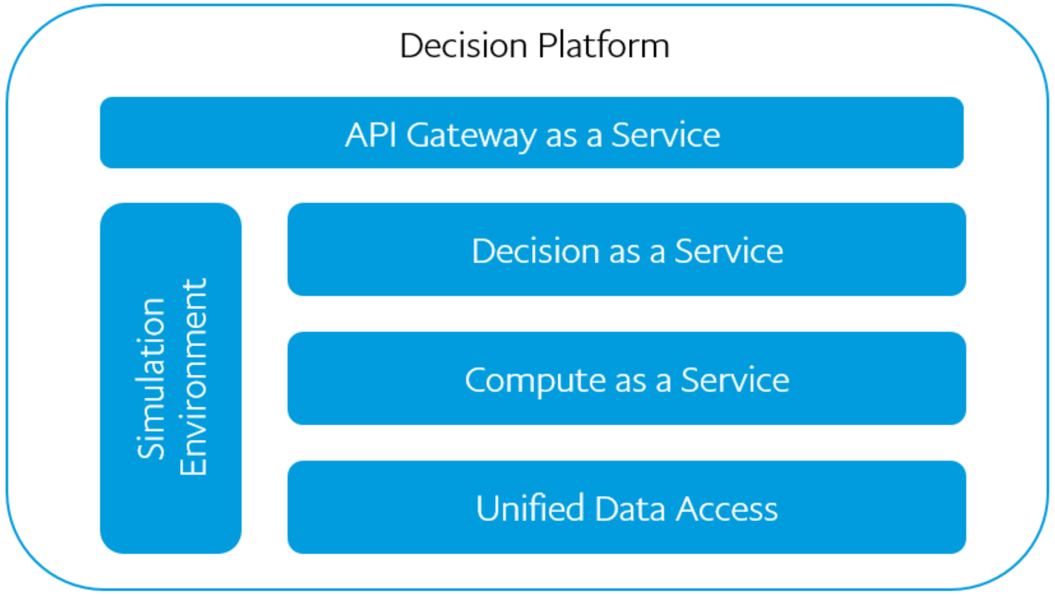
自动化运维 | Service Mesh | Elasticsearch | 微服务

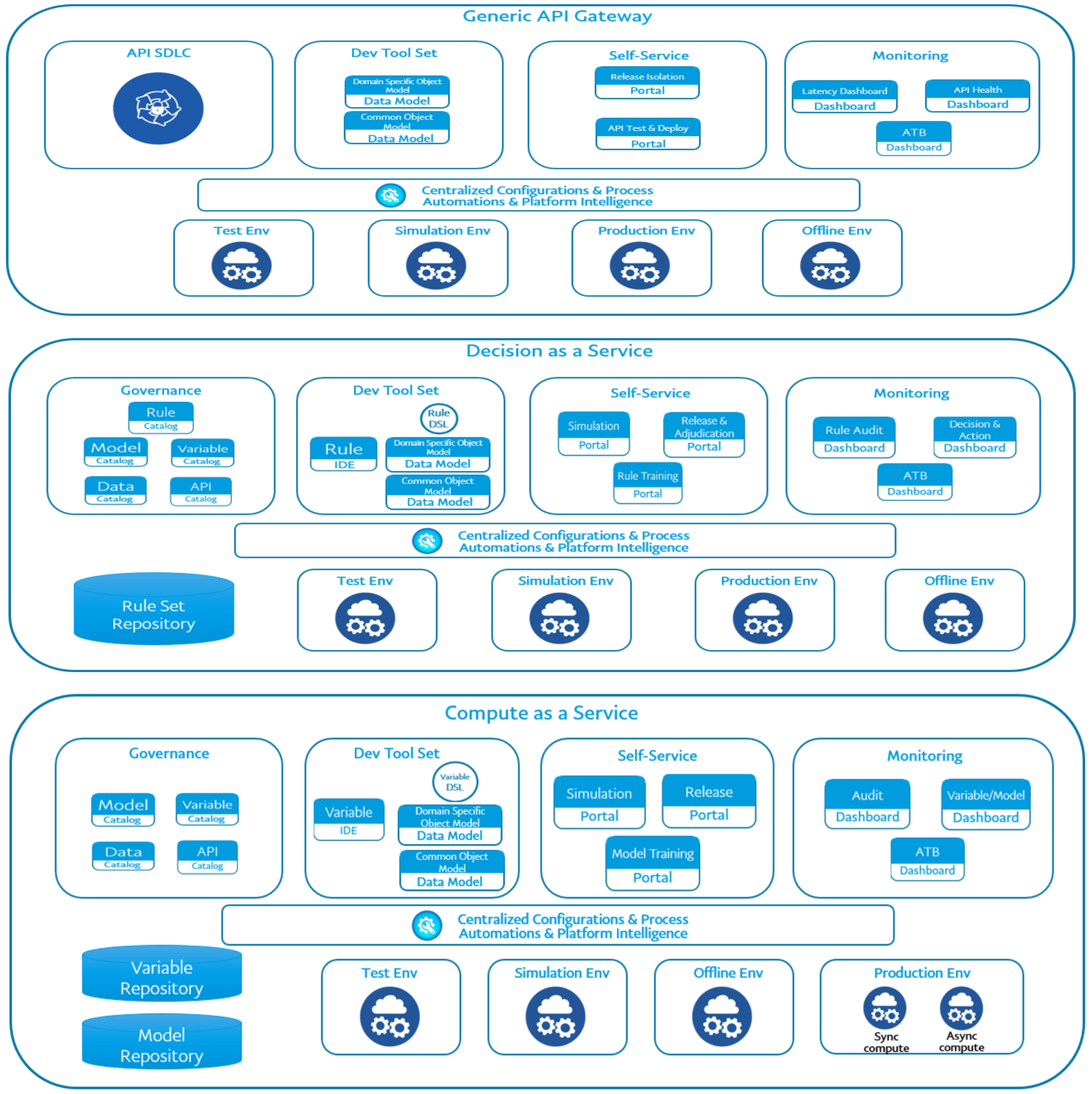
使用折扣码「QCon」优惠报名 咨询电话：13269078023

扫码锁定席位

Agenda

Online Decision Platform Overview





Unified Dependency Model & Dependency-Driven Execution

Given the variables/models/rules:

- What are the API requirements?
- How to execute the rules/models/variables in a highly efficient way?
- What is the latency expectation because of the data loadings & computations?
- …

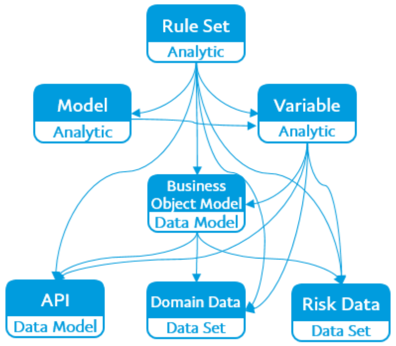
API Gateway Service

Decision Service

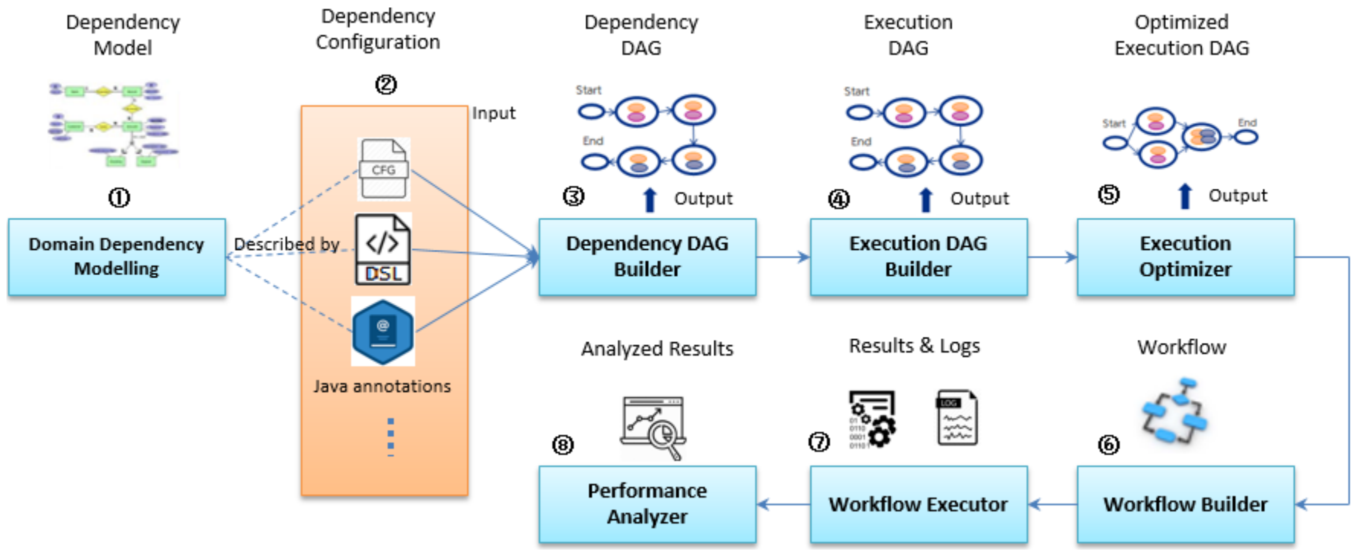
Compute Service

Unified Data Access (data assets)

Unified Dependency Model

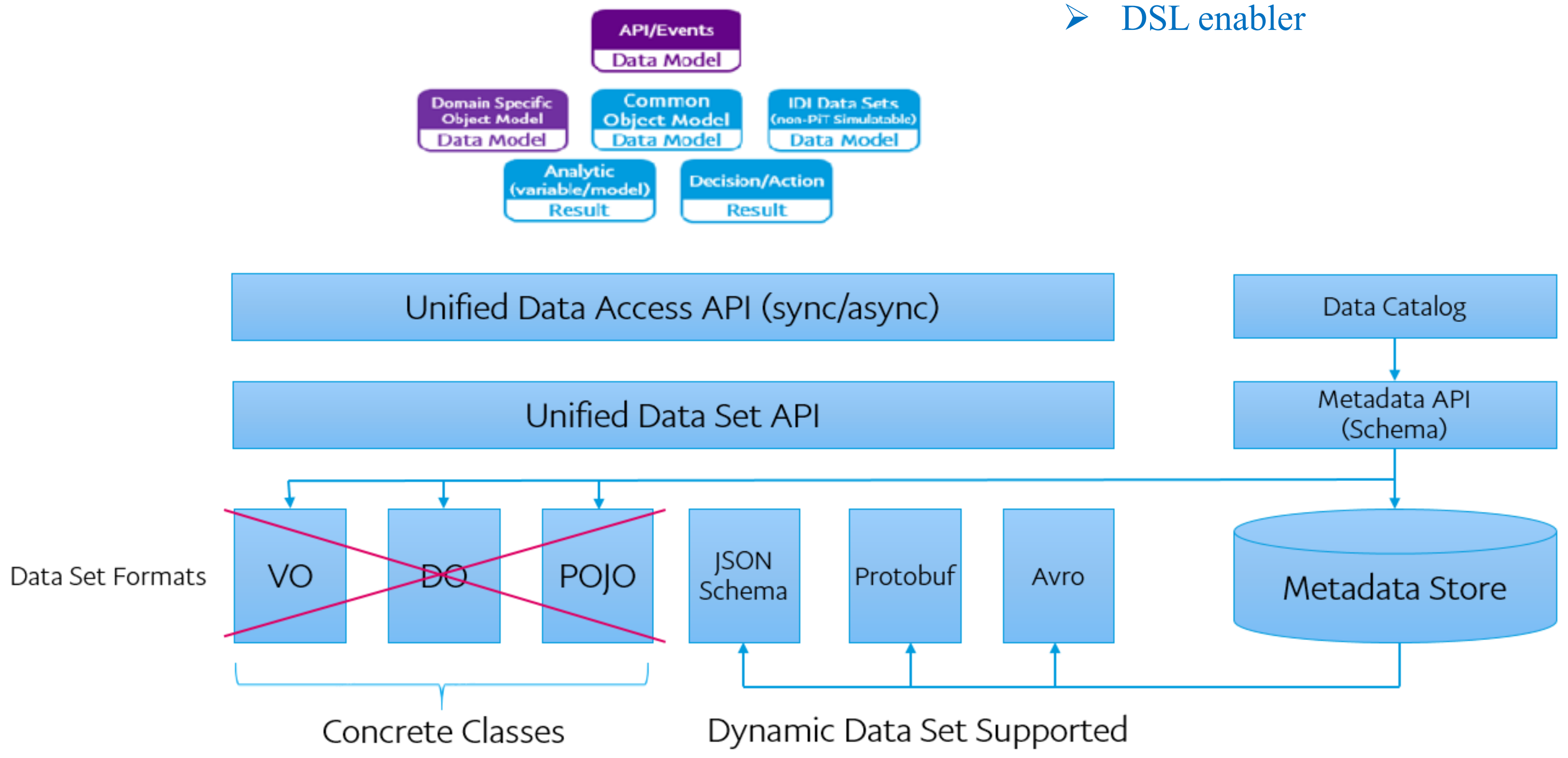


Execution

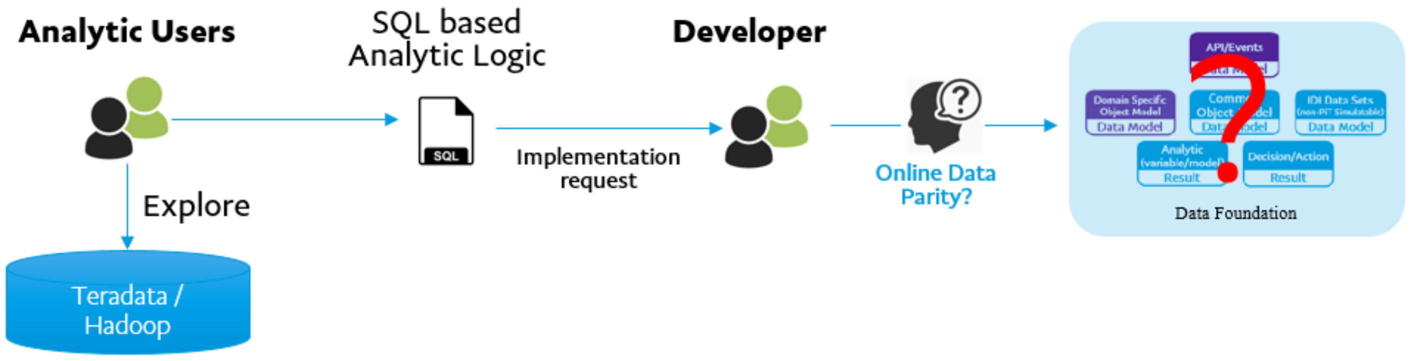


Data Set Abstraction & Unified Data Access

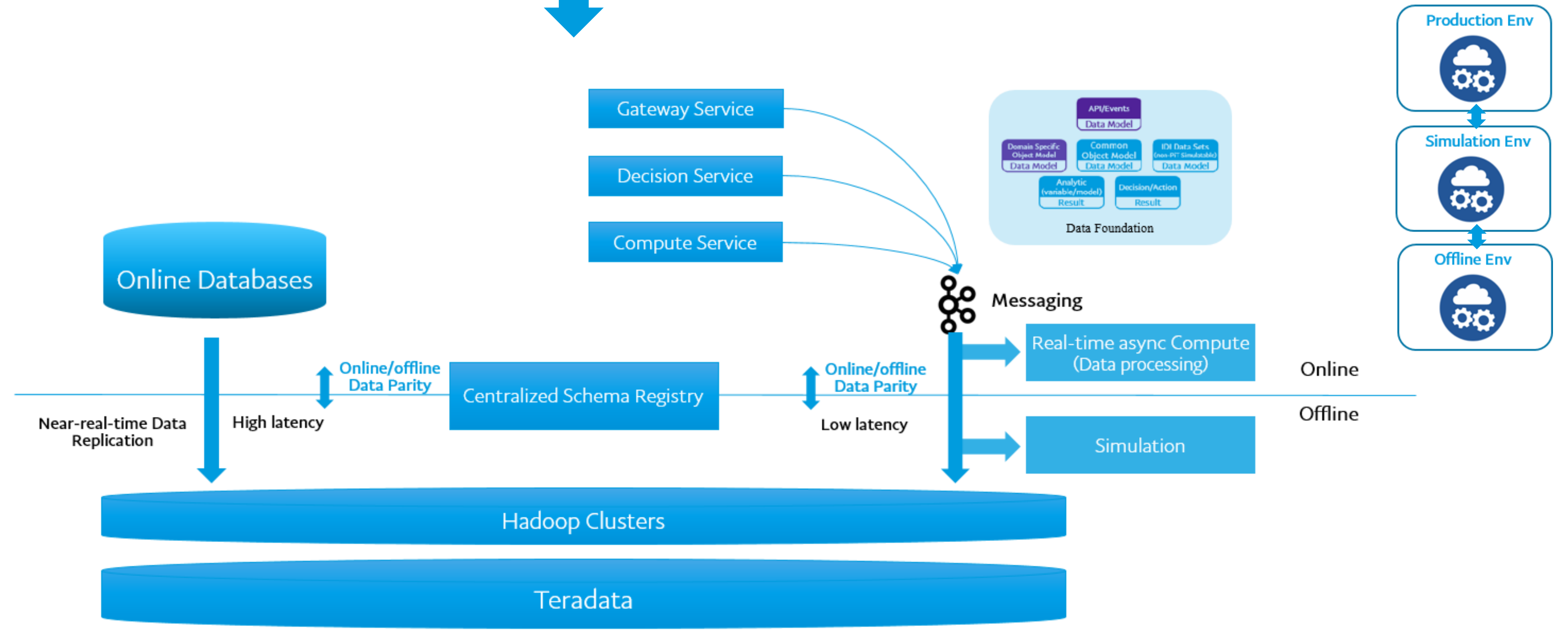
- Centralized configuration/metadata-driven data set management
- Enable centralized data governance
- Enable data dynamic release (no impact to services)
- Foundation of portable compute/decision engine
- DSL enabler



Online-offline Data Parity

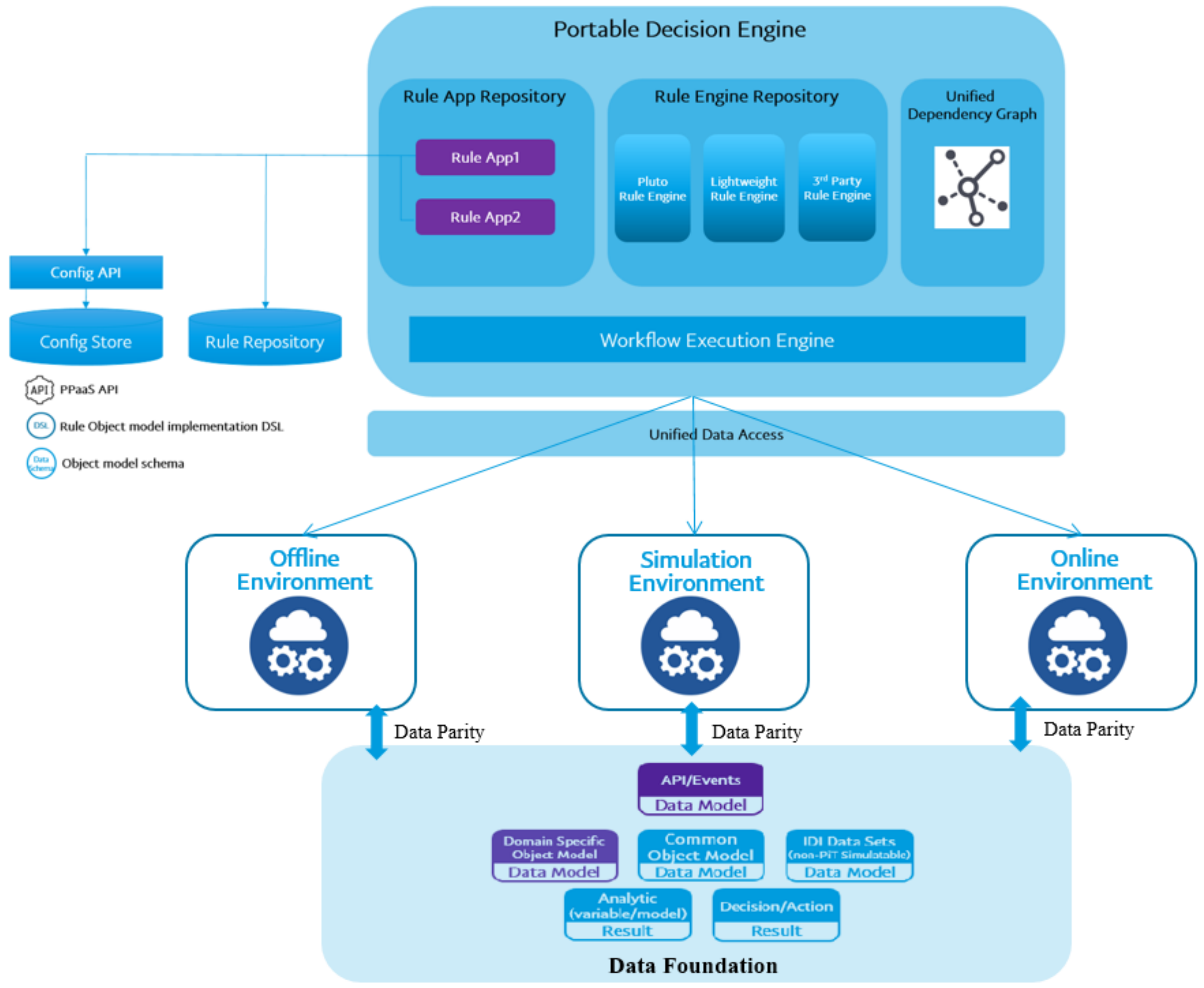


Change User Behavior

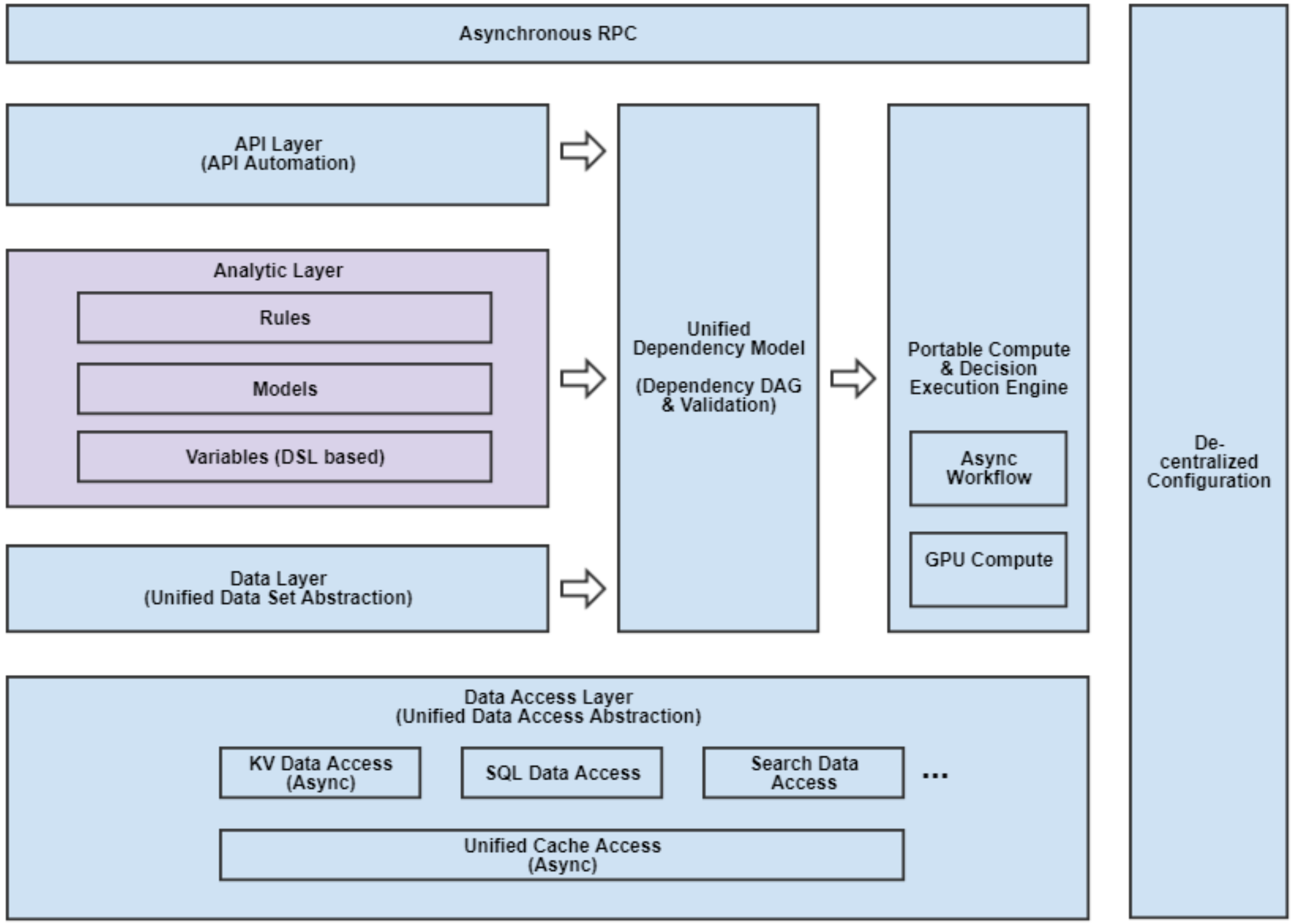


PayPal

Portable Execution Engine



Unified Decision/Computation Execution Stack


Asynchronous RPC
API Layer (API Automation)
Analytic Layer
Rules
Models
Variables (DSL based)
Data Layer (Unified Data Set Abstraction)
Unified Dependency Model
(Dependency DAG & Validation)
Portable Compute & Decision Execution Engine
Async Workflow
GPU Compute
De-centralized Configuration
Data Access Layer (Unified Data Access Abstraction)
KV Data Access (Async)
SQL Data Access
Search Data Access
...
Unified Cache Access (Async)

Agenda

Asynchronous Workflow

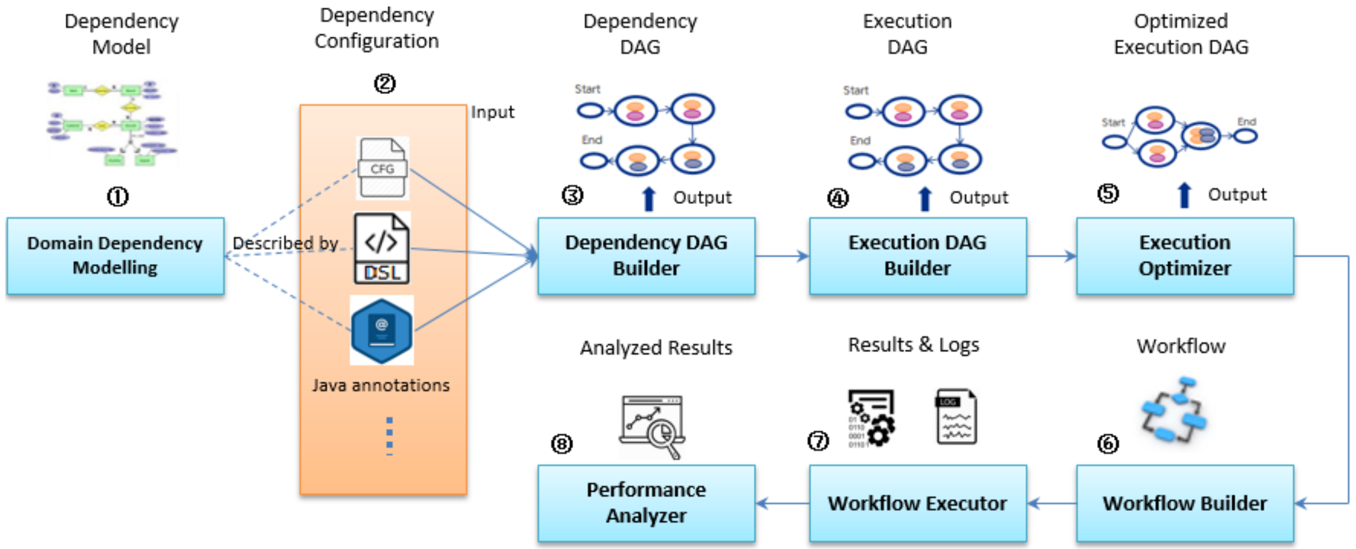
Why?

➢ Too many trivial nodes and the scheduling costs are high

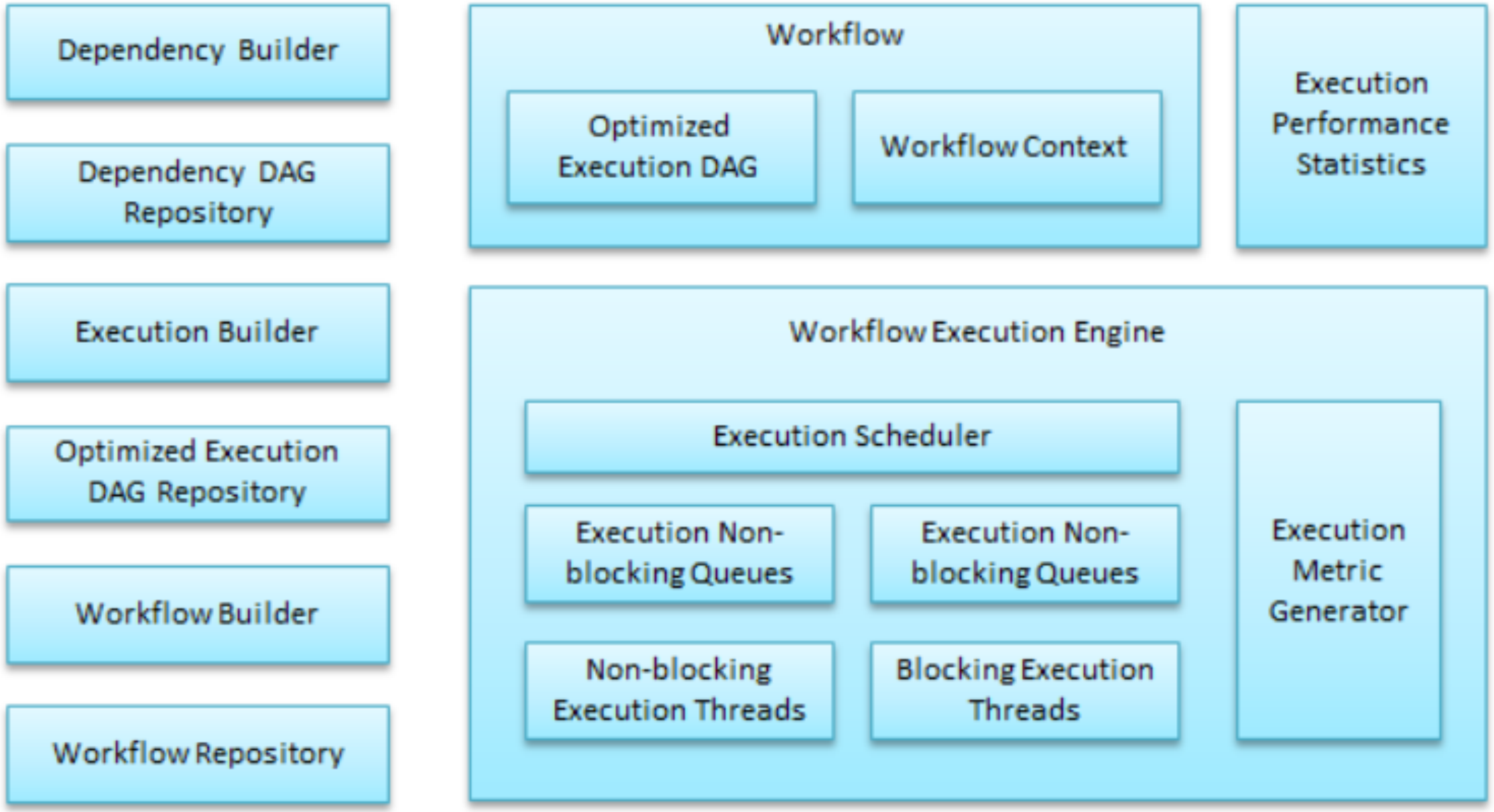
➢ Highly concurrent blocking IO operations require large number of parallel threads, make scheduling costs even higher

Solution:

1. Make trivial node scheduling/executions highly efficient
2. Generate more trivial nodes by making blocking operations non-blocking, which will benefit from #1

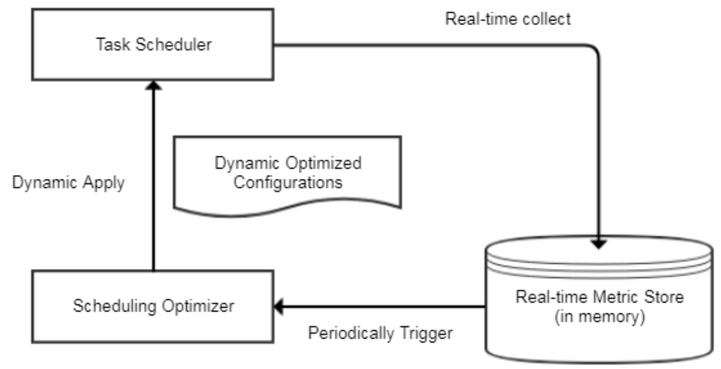


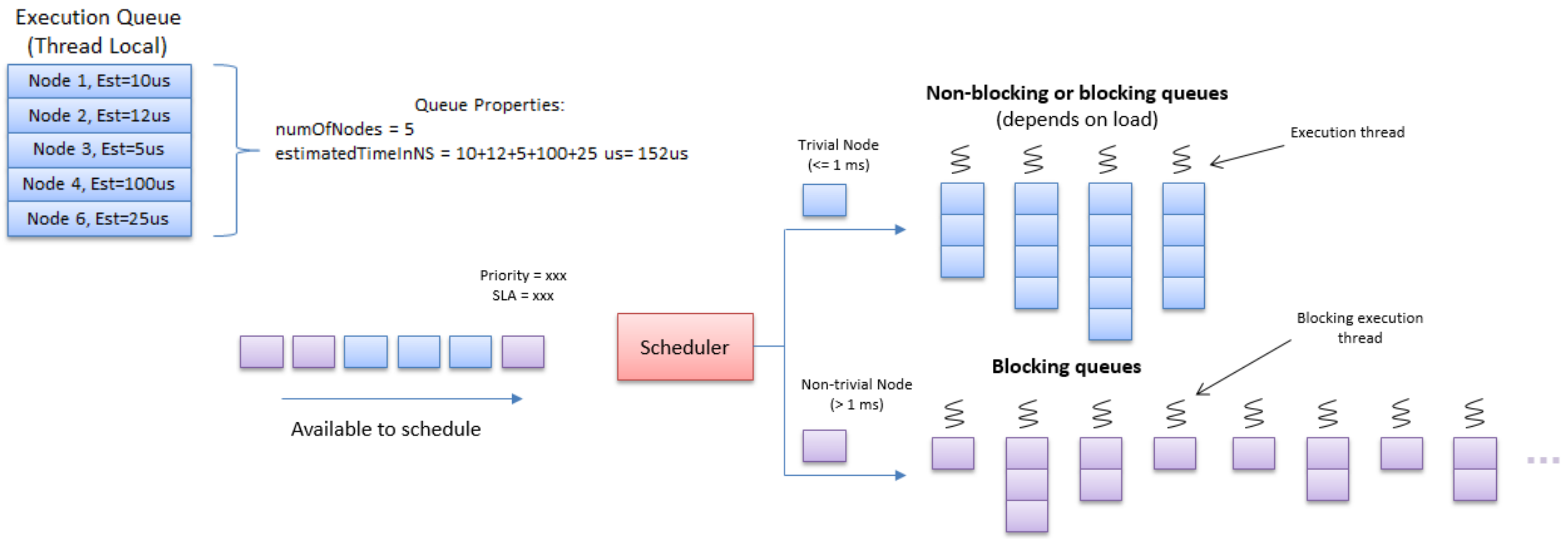
Asynchronous Workflow High Level Design


Dependency Builder
Dependency DAG Repository
Execution Builder
Optimized Execution DAG Repository
Workflow Builder
Workflow Repository
Workflow
Optimized Execution DAG
Workflow Context
Execution Performance Statistics
Workflow Execution Engine
Execution Scheduler
Execution Non-blocking Queues
Execution Non-blocking Queues
Non-blocking Execution Threads
Blocking Execution Threads
Execution Metric Generator

Tiered Node Scheduling

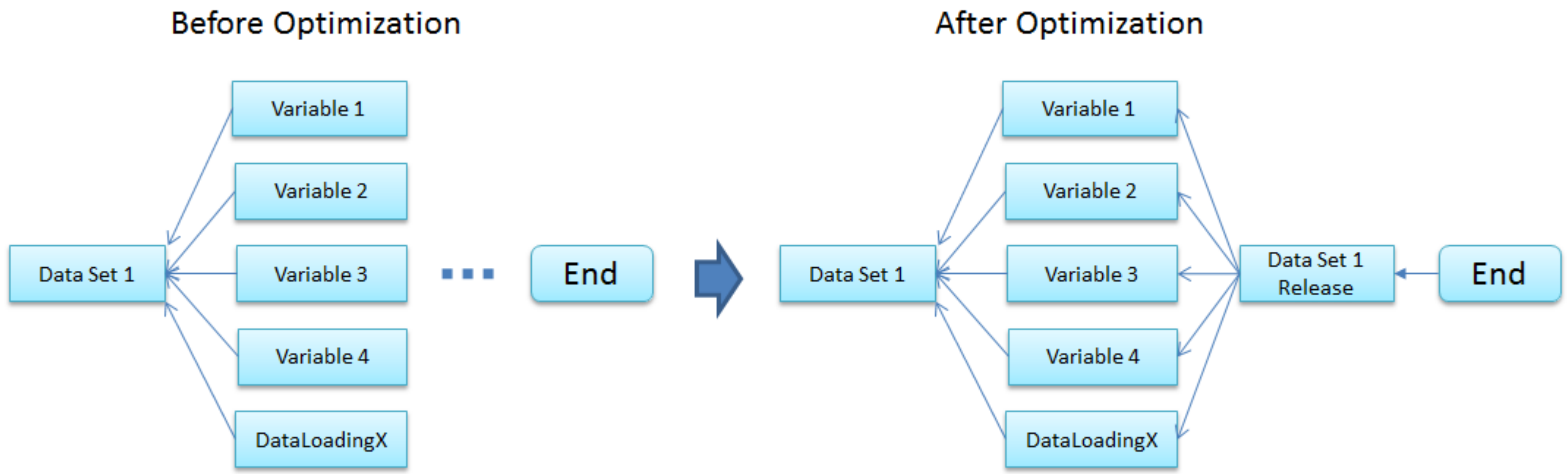
- Keep collecting node execution metrics
- Bind workflow thread to its queue (can be shared)
- Separate scheduling for trivial nodes and non-trivial nodes
- Smartly select scheduling algorithm for each tier:
  - Fair scheduler
  - Deadline scheduler
  - Prioritized scheduler





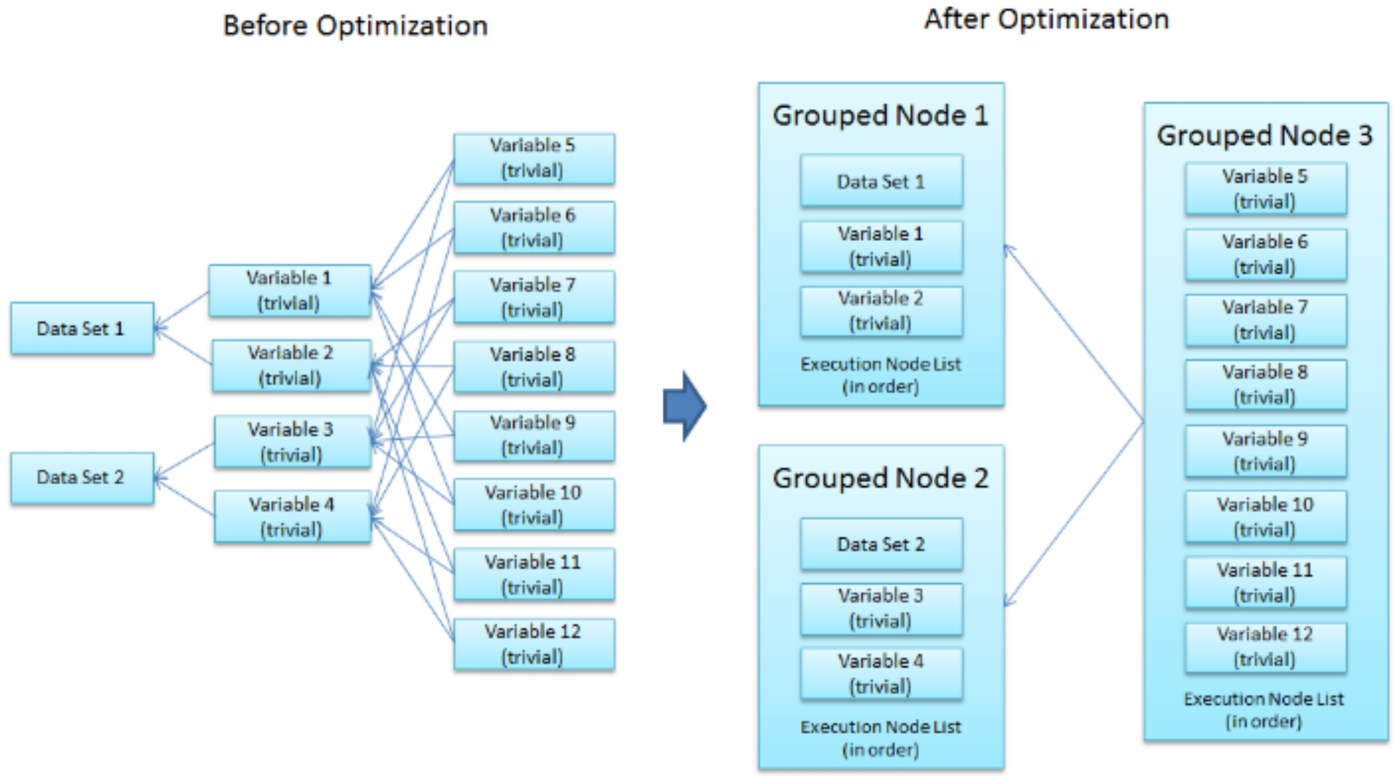
Execution Optimizer - Auto Release Optimizer

- ➢ Release objects as early as possible based on dependency understanding
- ➢ Improve GC performance



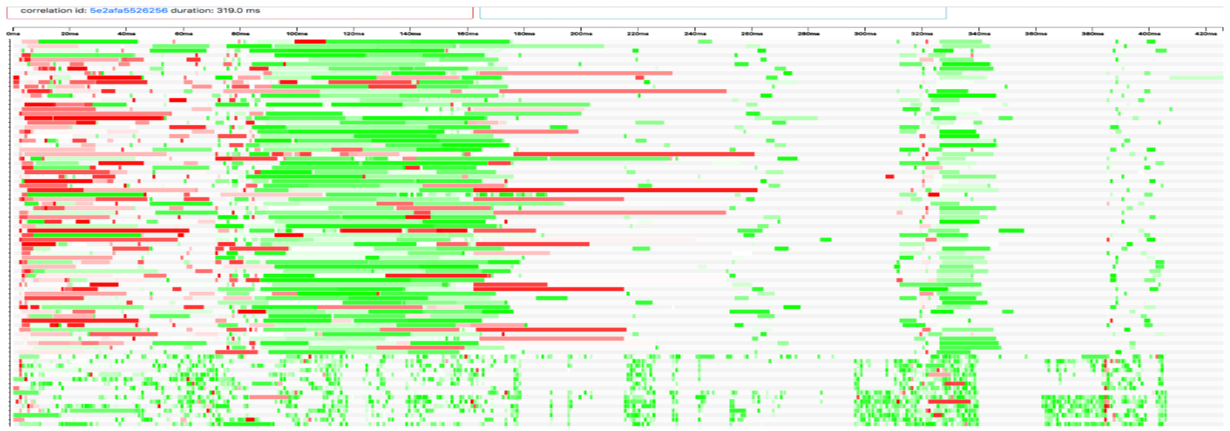
Execution Optimizer – Trivial Node Auto Group

- Reduce the number of trivial nodes
- Improve node scheduling cost

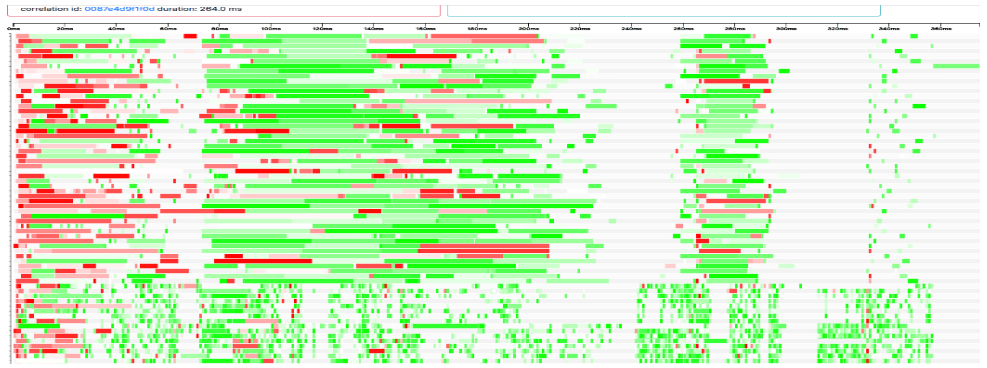


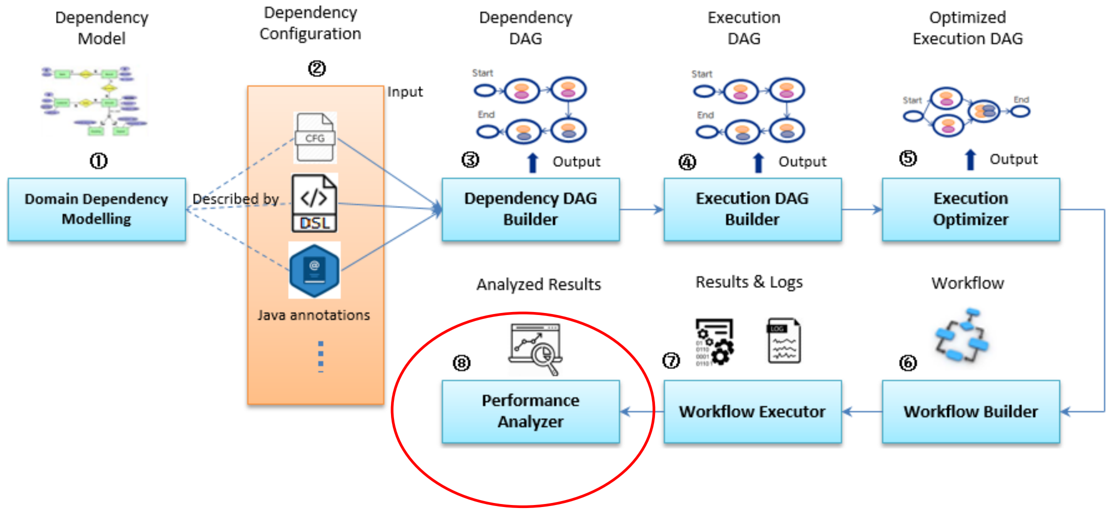
Workflow Performance Analyzer

Before optimization

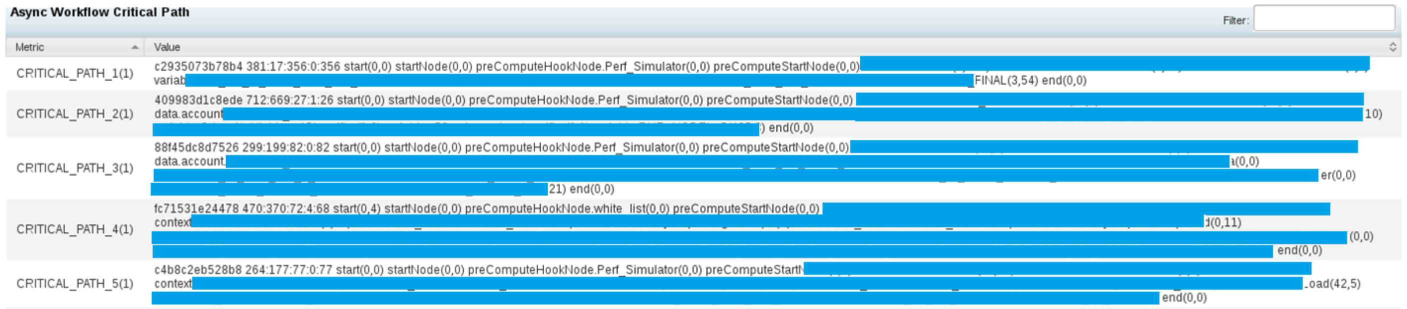


After optimization





Critical Path Analytics



Async Workflow Critical Path

Filter:

Metric	Value
CRITICAL_PATH_1(1)	c2935073b78b4 381:17:356:0:356 start(0,0) startNode(0,0) preComputeHookNode.Perf_Simulator(0,0) preComputeStartNode(0,0) [redacted] variab [redacted] FINAL(3,54) end(0,0)
CRITICAL_PATH_2(1)	409983d1c8ede 712:669:27:1:26 start(0,0) startNode(0,0) preComputeHookNode.Perf_Simulator(0,0) preComputeStartNode(0,0) [redacted] data.account [redacted] 10) [redacted]) end(0,0)
CRITICAL_PATH_3(1)	88f45dc8d7526 299:199:82:0:82 start(0,0) startNode(0,0) preComputeHookNode.Perf_Simulator(0,0) preComputeStartNode(0,0) [redacted] data.account [redacted] (0,0) [redacted] er(0,0) [redacted] 21) end(0,0)
CRITICAL_PATH_4(1)	fc71531e24478 470:370:72:4:68 start(0,4) startNode(0,0) preComputeHookNode.white_list(0,0) preComputeStartNode(0,0) [redacted] context [redacted] (0,11) [redacted] (0,0) [redacted] end(0,0)
CRITICAL_PATH_5(1)	c4b8c2eb528b8 264:177:77:0:77 start(0,0) startNode(0,0) preComputeHookNode.Perf_Simulator(0,0) preComputeStart [redacted] context [redacted] .oad(42,5) [redacted] end(0,0)

Agenda

1. Overall Introduction
2. Asynchronous Workflow
3. **Variable DSL**
4. Q&A

PayPal

Variable DSL

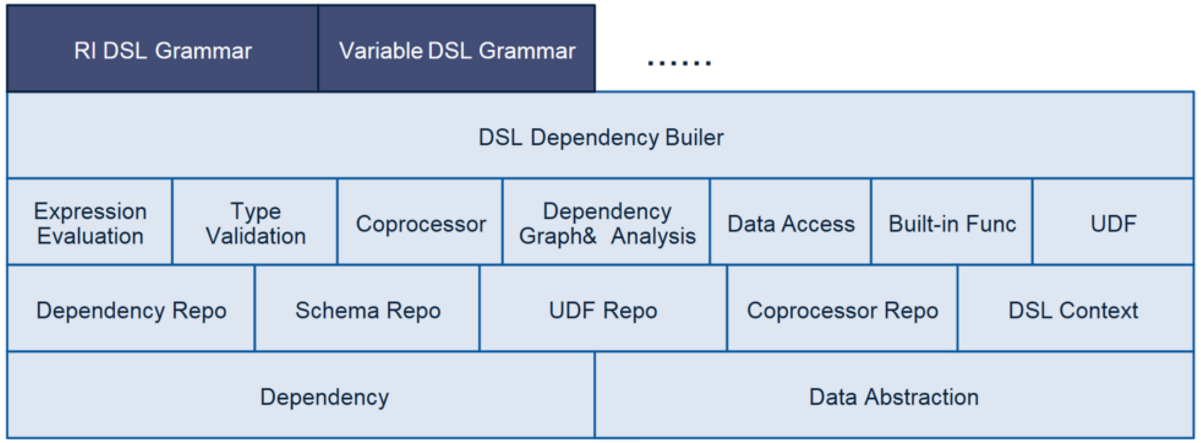
Why?

- Enable configuration-based analytic release
- Unified variable definition for both online and offline
- Dependency inline, no additional configuration needed
- Achieve more efficient executions than code based variables (exactly-once execution for any level of DSL expression with perfect hash optimizations)

About Expression:

- An expression must return a value
- Support Math Calc, Bool Calc and Compare
- Support property access, list access.
- Built-in func: map, reduce, if-else, load data
- Support UDF

…



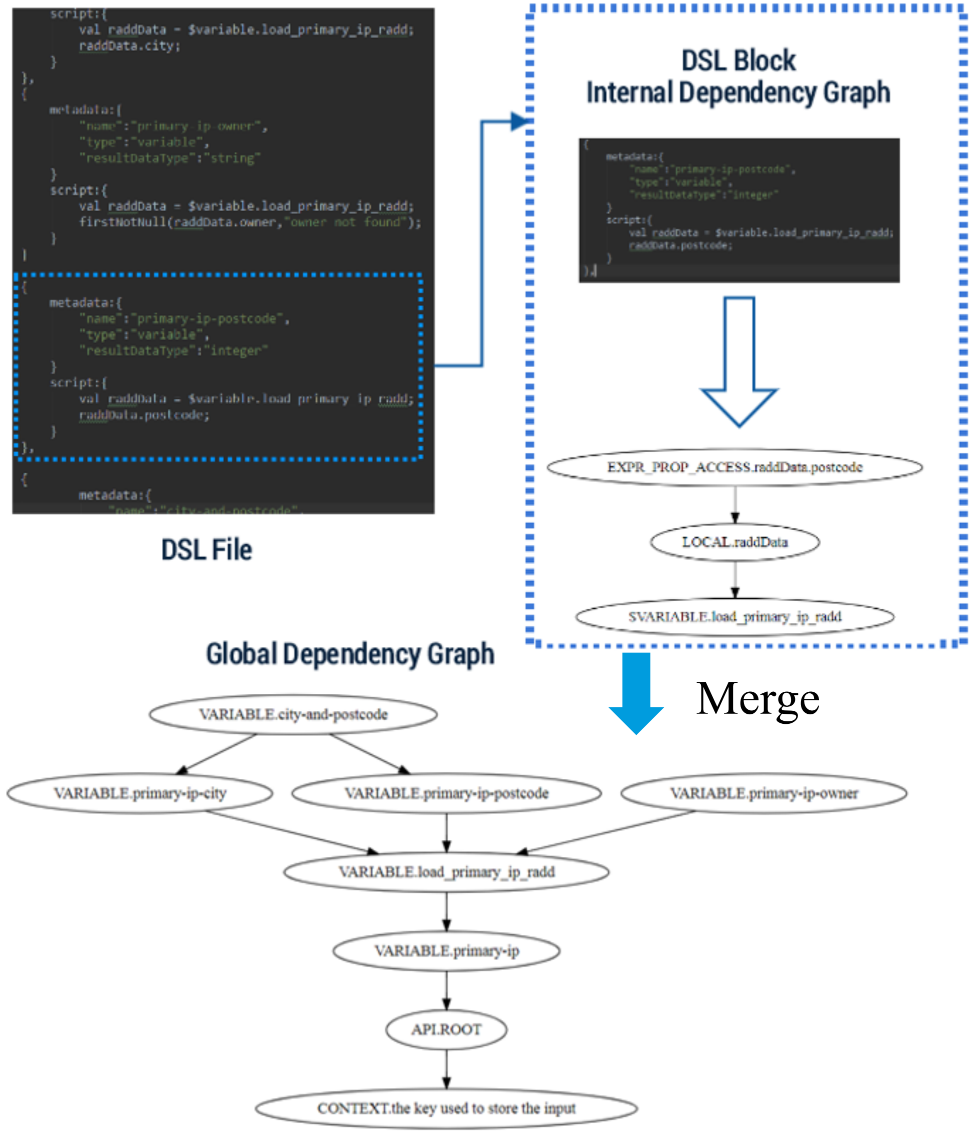
Dependency is the Key

Core APIs of dependencies

Set<Dependency> getThisDirectlyDependsOn();
Set<Dependency> getDependsOnThisDirectly();
String getDependencyId();
DependencyType getDependencyType();
Object calcValue(DSLContext context) ;

Two layers of dependencies

- Global Dependency
  - Input contexts
  - Variables
- Variable Internal Dependency
  - Literal Value, local value, global dependency
  - Math Calc, Bool calc, compare
  - If-else, map, reduce, create, UDF…
  - Data set loading



```
    script:{
        val raddData = $variable.load_primary_ip_radd;
        raddData.city;
    }
},
{
    metadata:{
        "name":"primary-ip-owner",
        "type":"variable",
        "resultDataType":"string"
    }
    script:{
        val raddData = $variable.load_primary_ip_radd;
        firstNotNull(raddData.owner,"owner not found");
    }
|
{
    metadata:{
        "name":"primary-ip-postcode",
        "type":"variable",
        "resultDataType":"integer"
    }
    script:{
        val raddData = $variable.load_primary_ip_radd;
        raddData.postcode;
    }
},
{
        metadata:{
            "name":"city-and-postcode",
```

DSL File

DSL Block
Internal Dependency Graph

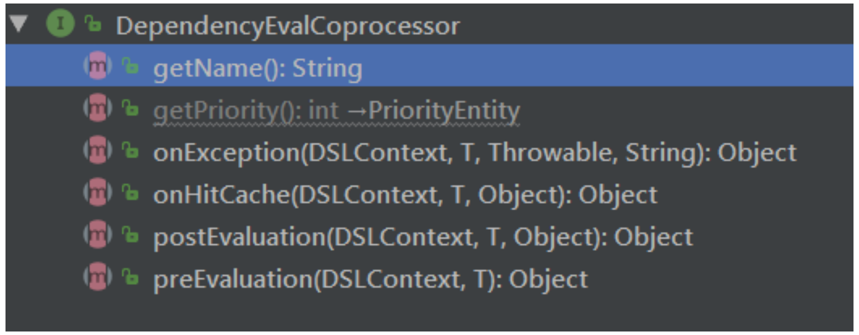
```
{
    metadata:{
        "name":"primary-ip-postcode",
        "type":"variable",
        "resultDataType":"integer"
    }
    script:{
        val raddData = $variable.load_primary_ip_radd;
        raddData.postcode;
    }
},
```

EXPR_PROP_ACCESS.raddData.postcode
LOCAL.raddData
$VARIABLE.load_primary_ip_radd
Merge
Global Dependency Graph
VARIABLE.city-and-postcode
VARIABLE.primary-ip-city
VARIABLE.primary-ip-postcode
VARIABLE.primary-ip-owner
VARIABLE.load_primary_ip_radd
VARIABLE.primary-ip
API.ROOT
CONTEXT.the key used to store the input

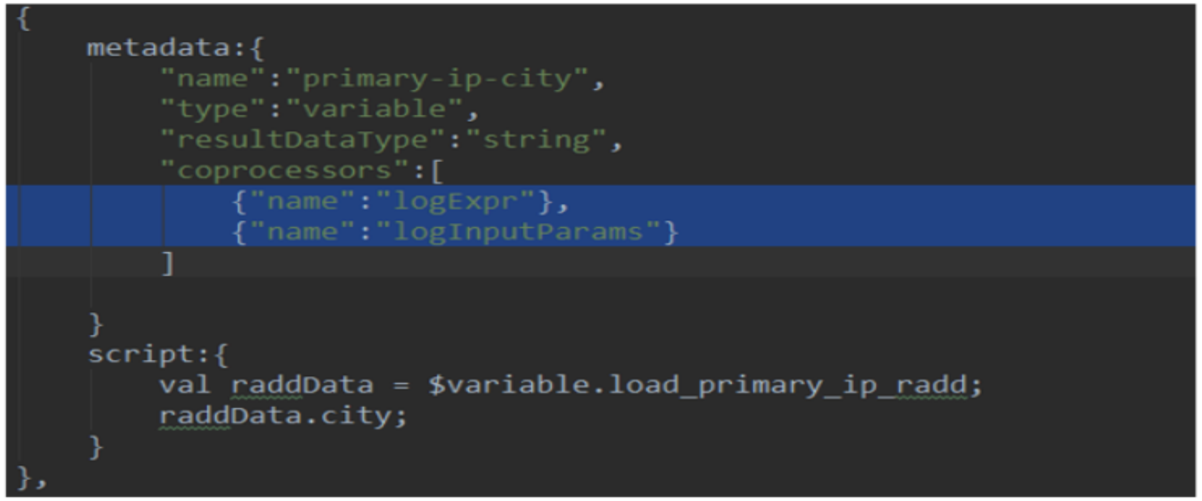
DSL Coprocessor

- DSL evaluation callback
- Non-intrusive inspection
- Make DSL easy to debug

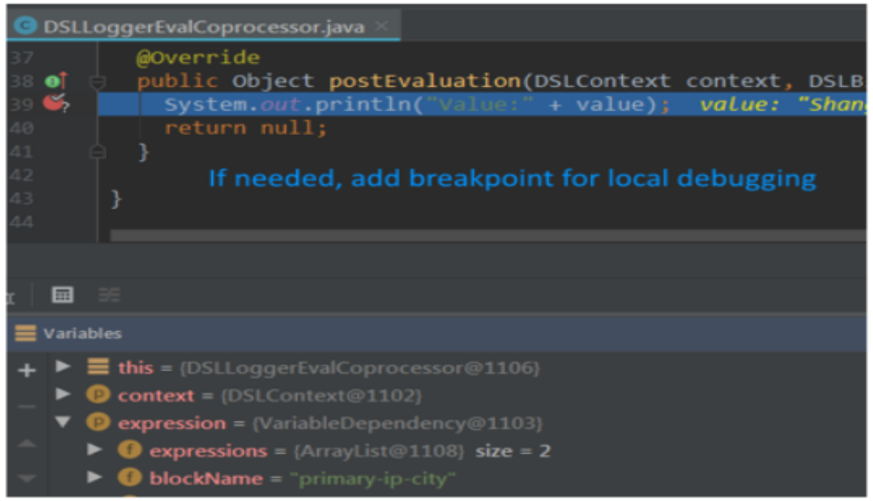
Coprocessor API:



```
DependencyEvalCoprocessor
    getName(): String
    getPriority(): int →PriorityEntity
    onException(DSLContext, T, Throwable, String): Object
    onHitCache(DSLContext, T, Object): Object
    postEvaluation(DSLContext, T, Object): Object
    preEvaluation(DSLContext, T): Object
```



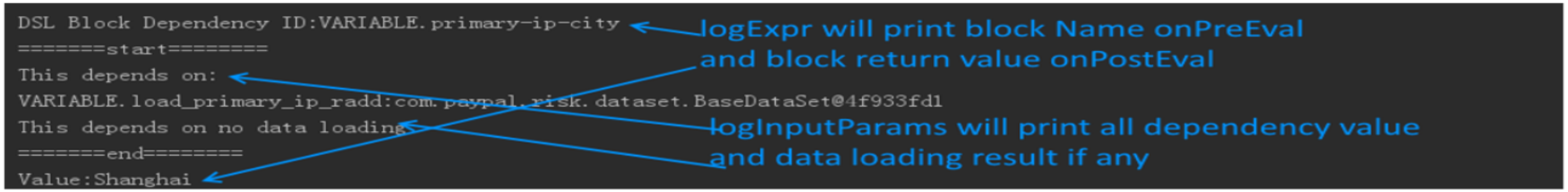
```
{
    metadata:{
        "name":"primary-ip-city",
        "type":"variable",
        "resultDataType":"string",
        "coprocessors":[
            {"name":"logExpr"},
            {"name":"logInputParams"}
        ]

    }
    script:{
        val raddData = $variable.load_primary_ip_radd;
        raddData.city;
    }
},
```



```
DSLLoggerEvalCoprocessor.java
    @Override
    public Object postEvaluation(DSLContext context, DSLB
        System.out.println("Value:" + value);   value: "Shan
        return null;
    }
}
```

If needed, add breakpoint for local debugging

```
Variables
▶ this = {DSLLoggerEvalCoprocessor@1106}
▶ context = {DSLContext@1102}
▼ expression = {VariableDependency@1103}
    ▶ expressions = {ArrayList@1108}  size = 2
    ▶ blockName = "primary-ip-city"
```



```
DSL Block Dependency ID:VARIABLE.primary-ip-city
=======start========
This depends on:
VARIABLE.load_primary_ip_radd:com.paypal.risk.dataset.BaseDataSet@4f933fd1
This depends on no data loading
=======end========
Value:Shanghai
```

logExpr will print block Name onPreEval and block return value onPostEval

logInputParams will print all dependency value and data loading result if any

ArchSummit 全球架构师峰会 [北京]

主办方 Geekbang 极客邦科技 InfoQ

技术创新的浪潮接踵而来，继续搬砖还是奋起直追？

云数据	AI	区块链	架构优化
高效运维	CTO技术选型	微服务	新开源框架

会议：2018年12月07-08日 培训：2018年12月09-10日

地址：北京·国际会议中心

Q & A
Thank You!