Measuring Performance Quality Scenarios in Big Data Analytics Applications: A DevOps and Domain-Specific Model Approach

Cristian Camilo Castellanos¹
Carlos A. Varela²
Dario Correal¹

¹Department of Systems Engineering
Universidad de los Andes
Bogotá, Colombia

²Department of Computer Science
Rensselaer Polytechnic Institute
Troy, NY, USA
Deployment Gap

Software Architecture Model

DevOps

ACCORDANT

Detect Near Mid Air Collisions NMAC

Problem

Solution Domain

Proposal

Experimentation
Deployment Gap Phenomenon

- “Despite the increasing interest in BDA adoption, actual deployments are still scarce” [1]
- “50% of companies do not have a specific data science production procedure.” [2]
- Delayed deployment of ready-to-use models (months: 31%, or years: 30%) [3]
- Incompatibility across multiple tools and communication problems. [4]
- It is not yet clear how to define and monitor different QoS in BDA applications [5]

And, What if I need multiple iterations and configurations???

Big Data Analytics (BDA) development

**Business**
- Real-time NMAC (Near Mid Air Collisions) service
- Response time ≤ 3 s.

**Data Science/Analytics**
- Decision Tree model
- Filtering and cleaning
- Modeling and evaluation

**IT Architecture**
- Latency < 3s
- Kafka, Python, Spark
- Cloud vs Fog computing

**Functional Requirements**
- Quality Scenarios (QS)

**Data**

**Monitoring**

**Deployment Gap**
“months: 31%, or years: 30%”
Challenge
How to reduce the big data analytics deployment gap by specifying and measuring quality scenarios and speeding up their deployment and performance monitoring?
Proposal
ACCORDANT
An exeCutable arChitectural mOdel foR big Data ANalyTics

1. Strategy (DSM and DevOps)

2. Proposal Process
1- Proposal Strategy

ACCORDANT: A Domain Specific Model and DevOps Approach
A Domain Specific Model

Domain Specific Language (DSL)

Lambda Architecture
- Latency < 3s
- Kafka, Python, Spark
- Cloud vs Fog computing

Functional Viewpoint Model

Deployment Viewpoint Model

Automatic Code Generation:
- Software Components
- Infrastructure as Code
ACCORDANT DSL Example

```rust
1 FunctionalView UCIModel{
2 Components{
3  Ingestor reader {
4    procModel:BATCH, format:"JSON", conn:"hdfs://ads-b/input/"
5    ports[Port adsb:PROVIDED]
6  },
7  Estimotor nmacDetector {
8    procModel:BATCH, pmml:"file:///...NMAC_TModel.pmml"
9    ports[Port in:REQUIRED, Port out:PROVIDED]
10  },
11  Estimotor nmacCluster {
12    procModel:BATCH, pmml:"file:///...Kmecans.pmml.pmml"
13    ports[Port in:REQUIRED, Port out:PROVIDED]
14  },
15  Sink writer {
16    procModel:BATCH, format:"CSV", conn:"hdfs://ads-b/output/"
17    ports[Port in:REQUIRED]
18  }
19 }
20 Connectors {
21  ProcCall callDetect {
22    roles : {
23       Role src:IN -> reader.adsb,
24       Role dst:OUT -> nmacDetector.in
25     }
26  },
27  ProcCall callClust {
28    roles : {
29       Role src:IN -> nmacDetector.out,
30       Role dst:OUT -> nmacCluster.in
31     }
32  }
33 }
34
1 DeploymentView UCI_ClusterModel
2 {
3  devs{
4    Device a {
5      host: "a" type: MEDIUM
6      cpu : 2 storage: 100 memory: 8
7    },
8    Device b {
9      host: "b" type: MEDIUM
10     cpu : 2 storage: 100 memory: 8
11    },
12    Device c {
13      host: "c" type: MEDIUM
14      cpu : 2 storage: 100 memory: 8
15    }
16 }
18 deployments{
19  Deployment spark_worker {
20    replicas : 3
21    pods {
22      Pod spark_workerp{
23        envs {
24          ExecEnvironment spark_worker_ex{
25            image: "rashiser/spark:2.0.1"
26            cpu_mem: 0.3
27            ports [8081]
28            commands ["/spark-worker"]
29          artfacts {
30            Artifact nmacArtifact{
31              comp : UCIModel.nmacDetector
32              scenarios QS:(DEADLINE <= 3600.0 SECONDS)
33          }
34          }
35      }
36  }
37 }
38 }
```
ACCORDANT
An exeCutable arChitectural mOdel foR big Data ANalyTics

1. Strategy (DSM and DevOps)

2. Proposal Process
<table>
<thead>
<tr>
<th>Context</th>
<th>Challenge</th>
<th>Proposal</th>
<th>Experimentation</th>
<th>Conclusions</th>
<th>Q&amp;A</th>
</tr>
</thead>
</table>

2 - Proposal Process
BDA Deployment Process

Requirements
- Business User defines 1- Quality Scenarios
- Data Scientist designs 2- Models and Transformations
- SW Architect designs 3- Software Architecture

Development
- designs
- ACCORDANT MM
- import PMML

Deployment
- 4- Integration
- 5- Code Generation
- 6- Code Execution

Operation
- BDA Solution

Context
- Challenge
- Proposal
- Experimentation
- Conclusions
- Q&A
• Specify performance **QS integrated with software architecture.**

• **Speed up BDA deployment and monitoring.**
Experimentation
Avionics BDA deployment

Business (FAA, private pilots)
- Real-time NMAC (Near Mid Air Collisions) service
- Response time ≤ 3 s.

Avionics Data Scientist
- Decision Tree model
- Filtering and cleaning
- NMAC detection Model

IT Architecture
- Latency < 3s
- Kafka, Python, Spark
- Cloud vs Fog computing

Functional Requirements
- Quality Scenarios (QS)

Data ADS-B

Deployment Gap
"months: 31%, or years: 30%"

Monitoring
Experimentation in Avionics

- Feasibility using Avionics use cases
  - **UC1**: Near Mid-Air Collision Analysis for route planning.
  - **UC2**: Near Mid-Air Collision Detection in operation.

- Deployment Effort
  - Time
  - Lines of Code (Complexity)

[Link to website](https://wcl.cs.rpi.edu/)
Business: Data Collection for 2, 20, and 200 nmi around JFK

- 2 nmi: **13,328 compares**
- 20 nmi: **656,177 compares**
- 200 nmi: **18,899,217 compares**

ADS-B Exchange

Automatic dependent surveillance – broadcast
Data Scientist: Build Analytics Model for NMAC Detection

Dtree.pmml

ADS-B Exchange
IT Architect: Define Software architecture of two use cases

UC1

- Batch NMAC Detection (UC1)

UC2

- Micro-Batch NMAC Detection (UC2)
IT Architect: Define Deployment Strategies

Functional View

Technology Assignments

Deployments
IT Architect: Specify Functional and Deployment Models

```
FunctionalView UC1Model{
  Components{
    Ingestor reader {
      procModel: BATCH, format: "JSON", conn: "hdfs://ads-b/input/"
      parts{Port adsb: PROVIDED}
    },
    Estimator nmacDetector {
      procModel: BATCH, pmml: "file:///...NMAC_TModel.pmml"
      parts{Port in: REQUIRED, Port out: PROVIDED}
    },
  }
  Connectors {
    ProcCall callDetect {
      roles :
        Role src:IN -> reader.adsb,
        Role dst:OUT -> nmacDetector.in
    },
    ProcCall callClust {
      roles :
        Role src:IN -> nmacDetector.out,
        Role dst:OUT -> nmacCluster.in
    }
  }
}
```

```
DeploymentView UC1_ClusterModel
{
  devs{
    Device a {
      host: "a" type: MEDIUM
      cpu : 2 storage: 100 memory: 8
    },
    Device b {
      host: "b" type: MEDIUM
      cpu : 2 storage: 100 memory: 8
    },
  }
  deployments{
    Deployment spark_worker {
      replicas : 3
      pods {
        Pod spark_workerp{
          envs {
            ExecEnvironment spark_worker_ex{
              image : "ramhiser/spark:2.1.0"
              cpu_req : 0.3
              ports [8080]
              commands ['/spark-worker']
            }
            Artifact nmacArtifact{
              artifacts {
                comp : UC1Model.nmacDetector
                scenarios {QS:(DEADLINE <= 3600.0 SECONDS)}
              }
            }
          }
        }
      }
    }
  }
}
```
Evaluator evaluator = EvaluatorUtil.createEvaluator("DTree.pmml");
TransformerBuilder pmmlTransformerBuilder = new TransformerBuilder(evaluator).withTargetCols()
    .withOutputCols().exploded(false);
List<StructField> fields = new ArrayList<StructField>();
fields.add(DataTypes.createStructField("a", DataTypes.IntegerType, true));
...
fields.add(DataTypes.createStructField("sz_norm", DataTypes.FloatType, true));
StructType schema = DataTypes.createStructType(fields);
Transformer pmmlTransformer = pmmlTransformerBuilder.build();
Logging.traceMetrics(Logging.DEADLINE, timestamp); //TRACING

apiVersion: apps/v1
kind: Deployment
spec:
  replicas: 3
  spec:
    containers:
    - name: spark-worker-ex
      image: ramhiser/spark:2.0.1
      ports:
      - containerPort: 8081
    resources:
      requests:
        cpu: 0.25
ACCORDANT: Monitoring application operation

**Context**

**Challenge**

**Proposal**

**Experimentation**

**Conclusions**

**Q&A**

---

**ACCORDANT XMI**

**Spark**

**kubernetes**

---

**LOG**

```
input  timestamp  processed
1       2.802862  1431
2       2.867957  2850
3       2.804899  3861
4       2.221732  3881
5       2.208233  3403
6       2.248742  3403
7       2.261614  3653
8       2.289406  4065
9       2.289406  4065
10      2.300308  4279
11      2.563807  4465
12      2.099357  3903
13      2.424423  4753
14      2.443361  4958
15      2.7113  3556
16      2.615949  5671
17      2.78935  5778
18      2.78935  5778
19      3.13471  6555
20      2.784451  6328
21      3.36729  4848
```
QS Monitoring of UC1

![Graph showing the comparison between uc1-cluster and uc1-local for different data ranges (2 nmi, 20 nmi, 200 nmi). The graph illustrates the performance in terms of seconds (log) for various data range scenarios.]
QS Monitoring of UC2

2 nmi

20 nmi
Results

Table 1: Effort Invested in Time(Hours) and Lines of Code

<table>
<thead>
<tr>
<th>UC</th>
<th>Approach</th>
<th>Design</th>
<th>Dev.</th>
<th>Infrastr.</th>
<th>(Re)Deploy</th>
<th>Total</th>
<th>FV LoC</th>
<th>DV LoC</th>
</tr>
</thead>
<tbody>
<tr>
<td>UC1</td>
<td>Traditional</td>
<td>0.5</td>
<td>18</td>
<td>2</td>
<td>2</td>
<td>22.5</td>
<td>196</td>
<td>74</td>
</tr>
<tr>
<td>UC2</td>
<td>Traditional</td>
<td>0.5</td>
<td>24</td>
<td>3</td>
<td>3</td>
<td>30.5</td>
<td>140</td>
<td>70</td>
</tr>
<tr>
<td>UC1</td>
<td>ACCORDANT</td>
<td>3</td>
<td>6</td>
<td>0.5</td>
<td>0.5</td>
<td>10</td>
<td>52</td>
<td>51</td>
</tr>
<tr>
<td>UC2</td>
<td>ACCORDANT</td>
<td>4</td>
<td>7</td>
<td>1</td>
<td>1</td>
<td>13</td>
<td>43</td>
<td>47</td>
</tr>
</tbody>
</table>

-57.3%  -73.47%  -32.86

Speed Up BDA deployment and monitoring iterations.

Deployment Gap Reduction
Conclusions
Contributions

- A DSM and DevOps approach to formalize and accelerate BDA solution development and deployment using FV and DV.
- A performance metrics specification and monitoring.
- An evaluation applied to avionics use cases with different deployment strategies and quality scenarios.

We believe that this work is a step forward towards deployment gap reduction!!
Future Work

- Train models to predict performance behavior.
- Architectural properties verification.

Open Challenges

- Design vs development effort.
- Adoption in other industry cases.
- Different deployment paradigms such as serverless or fog computing.
<table>
<thead>
<tr>
<th>Context</th>
<th>Challenge</th>
<th>Proposal</th>
<th>Experimentation</th>
<th>Conclusions</th>
<th>Q&amp;A</th>
</tr>
</thead>
</table>

Q & A Session

cc.castellanos87@uniandes.edu.co

Thanks!!!