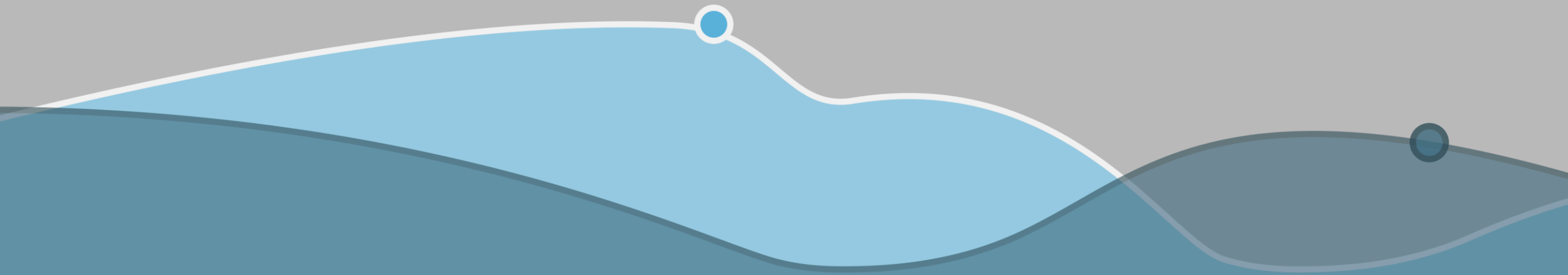
Microsoft

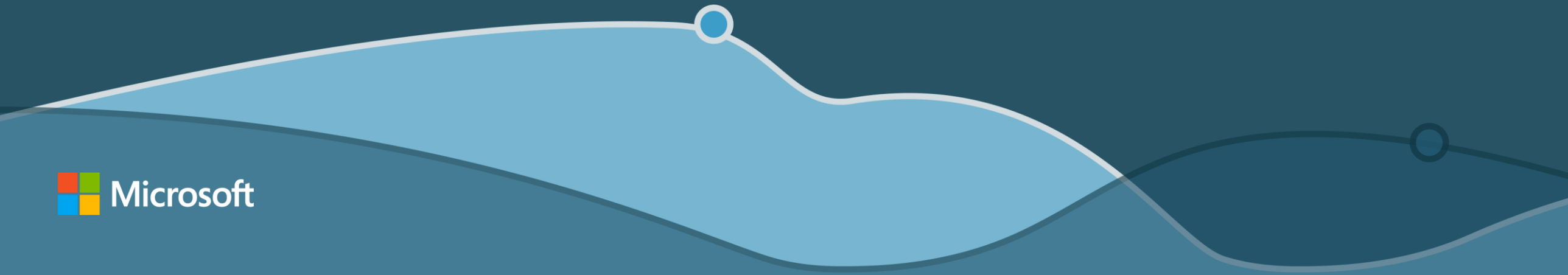
Cortana Analytics Workshop

Sept 10 – 11, 2015 • MSCC



Big Data on Azure: A Real-World Implementation Using Cortana Analytics

Brian Raymer
Sr. Consultant – Premier Developer Consulting



Microsoft

Agenda

- Who am I
- What is Premier Developer
- Architectural Overview of the Solution
- DocumentDB Details
- Azure Stream Analytics Details
- Azure Data Factory and HDInsight Details
- Q&A

Who am I?

8 years with Microsoft
22 years with SQL Server
28 years in IT

Premier Developer Consulting Team – Data Platform Specialist

Brian.Raymer@Microsoft.com
913-323-1269

What is Premier Developer (or PSfD)

Support for customers who desire long-term, relationship based services for developers

Reducing Risks and Costs

- Boost system availability & reliability
- Reduce the cost of support for LOB apps

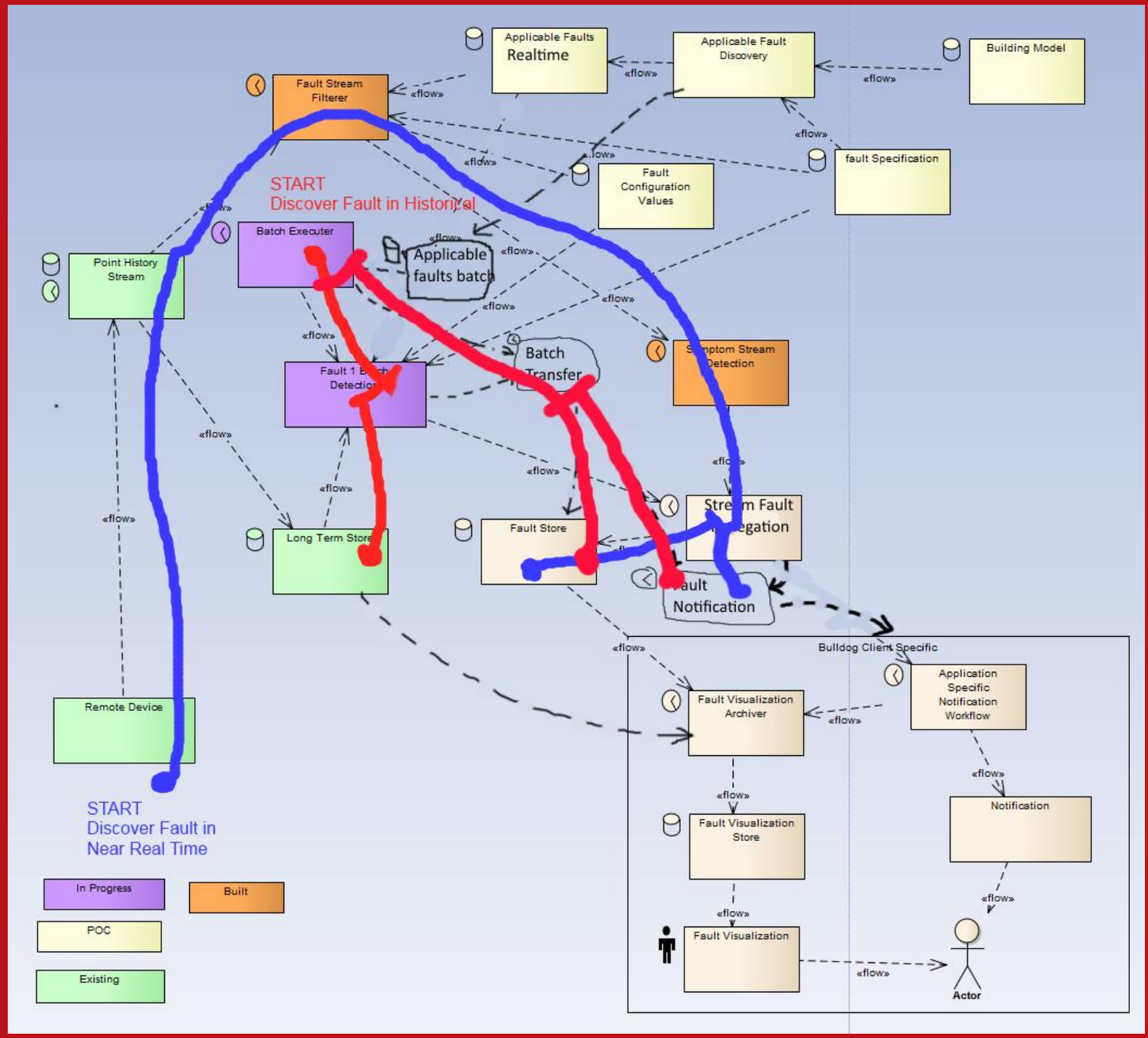
Increasing Productivity

- Quicker resolution of development issues
- Improve application scalability & performance

Being Proactive

- Enable a formalized development process
- Help understand & use Best Practices

Architectural Overview


Applicable Faults Realtime
Applicable Fault Discovery
Building Model
Fault Stream Filterer
fault Specification
Fault Configuration Values
START
Discover Fault in Historical
Batch Executer
Applicable faults batch
Point History Stream
Batch Transfer
Symptom Stream Detection
Fault 1 Batch Detection
Long Term Store
Fault Store
Stream Fault Aggregation
Fault Notification
Remote Device
Bulldog Client Specific
Application Specific Notification Workflow
Fault Visualization Archiver
Notification
Fault Visualization Store
Fault Visualization
Actor
START
Discover Fault in
Near Real Time
In Progress
Built
POC
Existing

DocumentDB

DocumentDB

Fault Definitions
Flexibility – No Schema
Fast Lookups

Azure Stream Analytics

Azure Stream Analytics

Near Real Time Fault Detection

- Read rules from DocumentDB (at setup)
- Consume point data from Event Hub
- Detect Faults
- Collect Fault data and prior period data
- Send to data visualization engine

Azure Stream Analytics

```
create table input(co2Level float, AfdMatchId nvarchar(max), MinimumCO2Concentration bigint,
DurationForFault bigint, Timestamp DateTime);
--=============================================================
select AfdMatchId, fault_start_time, fault_end_time,
        (case when co2symptomDuration >= 3 then 'Fault'
              else 'No Fault' end) as fault_type
from (
       select sum(case when c1.co2Level <= MinimumCO2Concentration then 1
                       else 0 end)  as co2symptomDuration,
       max(DurationForFault)     as DurationForFault,
        c1.AfdMatchId            as AfdMatchId,
       min(TimeStamp)            as fault_start_time,
       max(Timestamp)            as fault_end_time
       from input c1 timestamp by Timestamp Partition By PartitionId
       Group By slidingwindow(minute, 3), c1.AfdMatchId
       ) level1symptom
```

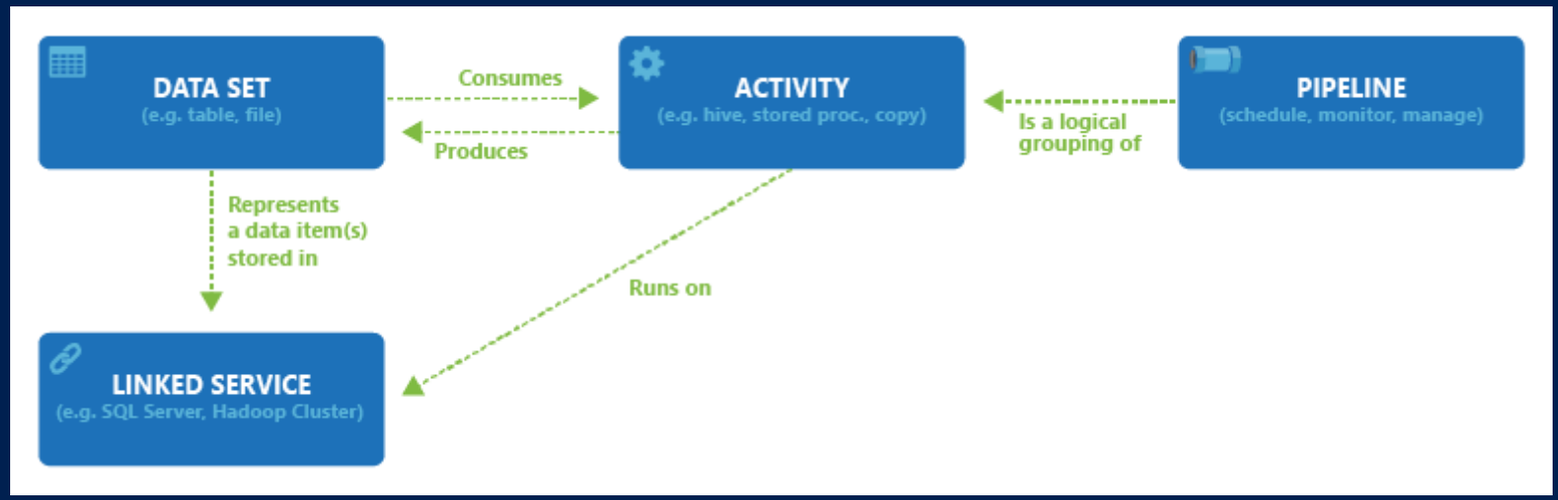
Azure Data Factory and HDInsight

Azure Data Factory

Control Batch Execution
Create Pipeline from Blob Storage to Visualization
Why did we need this?

Azure Data Factory

Linked Services
Data Sets
Pipelines
Activities



Azure Data Factory – Aligning Datasets

Source:

```
        "availability": {
            "frequency": "Hour",
            "interval": 1
        },
        "policy": {
            "externalData": {
                "dataDelay": "06:00:00",
                "retryInterval": "00:10:00",
                "retryTimeout": "00:10:00",
                "maximumRetry": 3
            }
```

Destination:

```
        "availability": {
            "frequency": "Hour",
            "interval": 6
        }
```

Azure Data Factory – Pipeline

```
 "properties": {
        "description": "Batch Fault Detection",
        "activities": [{
                "type": "HDInsightPig",
                "typeProperties": {
                    "scriptPath": "pigscripts\\1.2\\Data_Conditioning_Azure.pig",
                    "scriptLinkedService": "hwrdpdata",
                    "defines": {
                        "generic_params": "generic parameters",
Etc.
                    }
                },
                "inputs": [{"name": "c6432210-c590-4cf6-a79c-ab3e45cd7911-PointData"}],
                "outputs": [{"name": "c6432210-c590-4cf6-a79c-ab3e45cd7911-
DataConditioningResultData"}],
                "policy": {
                    "timeout": "01:00:00",
                    "delay": "06:00:00",
                    "concurrency": 1,
```

Azure Data Factory – On Demand HDInsight

```
{
    "name": "HDInsightOnDemandCluster",
    "properties": {
        "hubName": “generic_hub",
        "type": "HDInsightOnDemand",
        "typeProperties": {
            "version": null,
            "clusterSize": 4,
            "location": null,
            "timeToLive": "00:30:00",
            "coreConfiguration": {},
            "hBaseConfiguration": {},"hdfsConfiguration": {},
            "hiveConfiguration": {},"mapReduceConfiguration": {},"oozieConfiguration": {},
            "sparkConfiguration": {},"stormConfiguration": {},"yarnConfiguration": {},
            "additionalLinkedServiceNames": [],"linkedServiceName": “generic"
        }
    }
}
```

HDInsight

Controlled by Data Factory

Batch Fault Detection Via Pig

Why did we need this?

HDInsight – Pig Scripts

```
set debug $script_debug_flag;
set verbose $script_verbose_flag;
set job.name '$jobid-Fault_State_Azure';
set default_parallel $script_parallelism;
set output.compression.enabled $output_compression_enabled;
set output.compression.codec $output_compression_codec;

--#jars registration;
register '$jars_path/$jar_files';

--## data load;
-- load conditioned raw point data (stored by script "Data_Conditioning_Azure.pig")
 point_data_raw = load '$if_result_data_path/$jobid/point_data_raw' using
PigStorage('$dc_result_file_seperator') as ($df_if_result_schema);
```

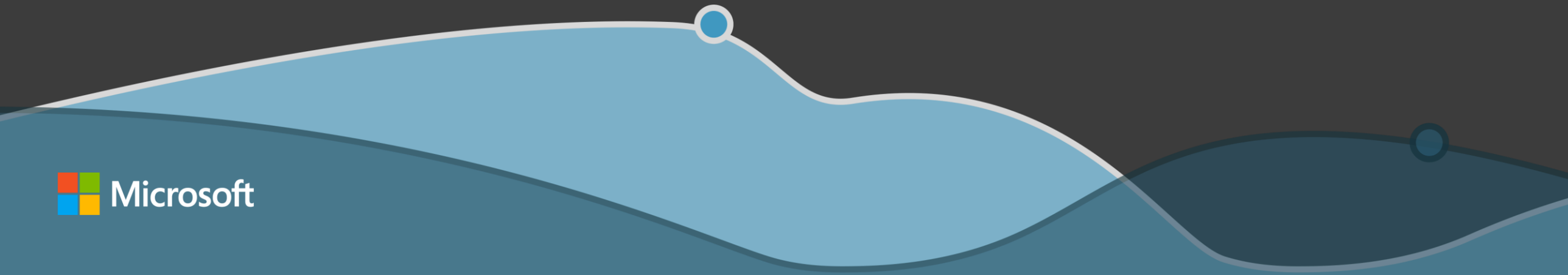
HDInsight – Pig Scripts

```
-- load information of faults in progress
 fault_data_agg = load '$if_result_data_path/$jobid/aggregate' using
        PigStorage('$sr_result_file_seperator') as ($sr_if_result_schema);

-- Join and filter to get timeseries of all faults in progress
fault_in_progress_output = foreach (filter
         (join fault_data_agg by (siteid,matchid), point_data_raw by (siteid,matchid))
        by  point_data_raw::sensortime >= fault_data_agg::seqstart)
generate CONCAT(CONCAT(point_data_raw::siteid,'/'),point_data_raw::matchid) as pathpartition,
        point_data_raw::matchid as matchid, point_data_raw::afdid as afdid, point_data_raw::siteid as
siteid,
        point_data_raw::sensortime as sensortime, point_data_raw::afdrole as afdrole,
point_data_raw::pointid as pointid,
        point_data_raw::pointvalue as pointvalue, point_data_raw::quality as quality,
point_data_raw::aggwindow as aggwindow,
        point_data_raw::aggtype as aggtype;

-- parition and store time series data.
STORE fault_in_progress_output INTO '$if_result_data_path/$jobid/timeseries'
USING org.apache.pig.piggybank.storage.MultiStorage('$if_result_data_path/$jobid/timeseries','0',
'$if_output_compression', '$if_result_file_seperator');
```

Q & A



Microsoft

Microsoft