Apache Metron Profiler

Nick Allen, Committer and PMC Member, Apache Metron

DataWorks Summit / Hadoop Summit 2017

Birds of a Feather - Cyber Security and Apache Metron

Metron Version 0.4.0

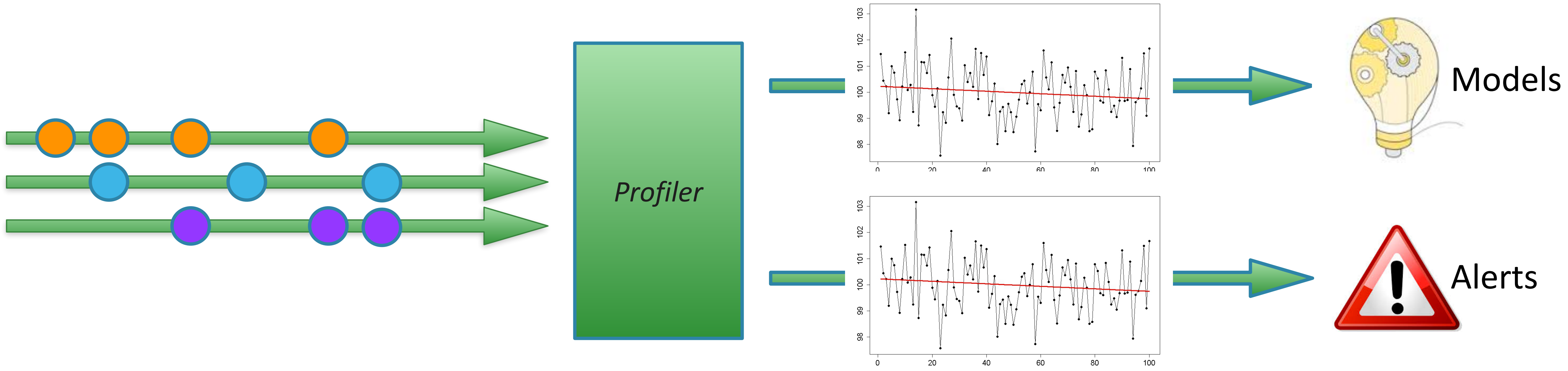
HORTONWORKS®
POWERING THE FUTURE OF DATA™

Agenda

- Introduction
- “Hello World” Profile
- Data Sketches
- Profiles
- Implementation

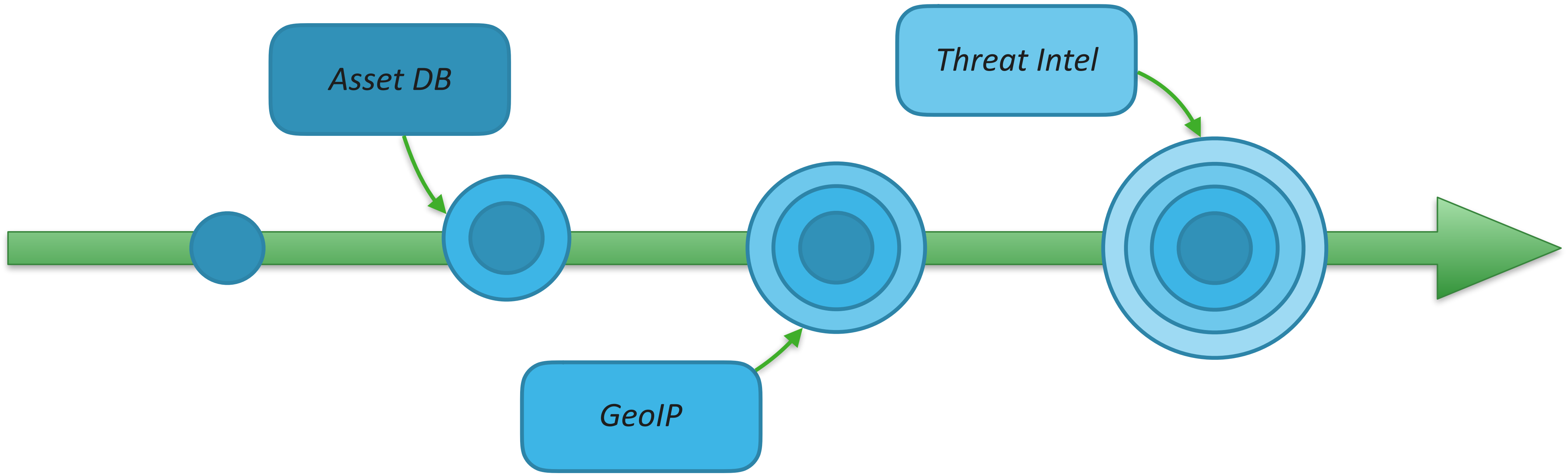
Introduction

The Profiler



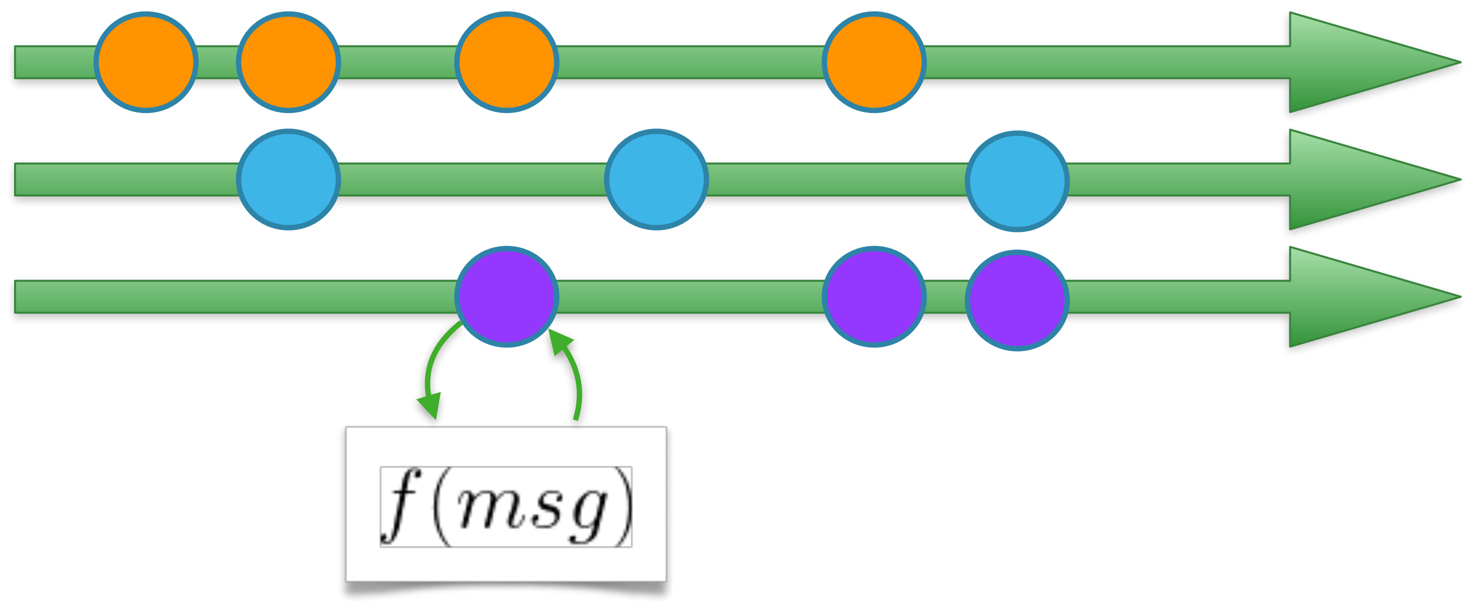
- A generalized, extensible solution for extracting feature sets from high throughput, streaming data
- Generates a <u>profile</u> describing the behavior of an <u>entity</u>; a host, user, subnet or application
- A foundational component for both security model building and alerting in Metron

Background: Enrichments



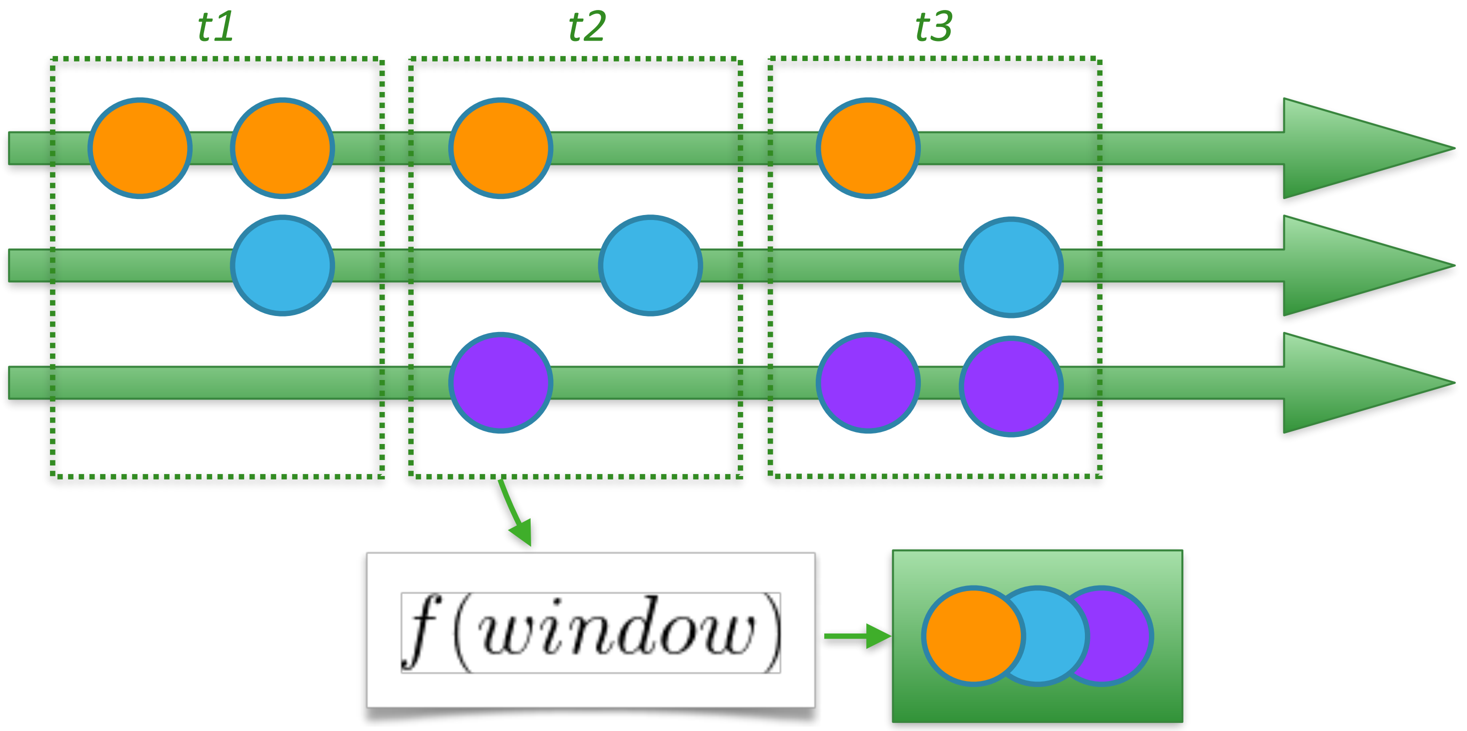
- Enriching telemetry with contextual clues is invaluable
  - Dramatically improves threat triage and response
  - Another foundational component for security model building (expands the feature set)
- Security telemetry in Metron is ‘sticky’
  - Enrichments ‘stick’ as the telemetry progresses through the system

The Problem



- Enrichments operate within the context of a single message
  - Simple, efficient and scalable for most enrichment and triage scenarios
- Insufficient for other scenarios
  - Looking across time; trending
  - Looking across data sources; correlation
  - Looking at aggregate behaviors; How is an “application” or “user” behaving as a whole?

The Profiler



- The Profiler creates logical windows that span both time and data sources
- The user defines a Profile that operates on these logical windows
  - A Profile consumes each message within a given window
  - Identifies an Entity; the subject of interest; a server, user, subnet, or application name
  - Produces a Result; a summary of an Entity's behavior within the window

Profiles are Time Series

- A Profile executed on a given window results in a unique Result for each Entity

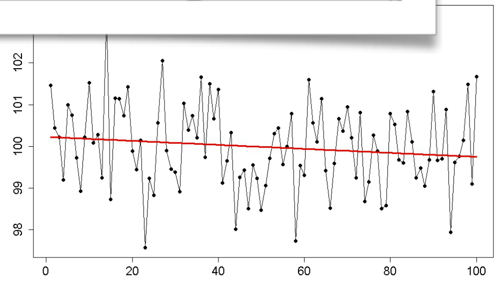
$$f(window_{t1}) \rightarrow (entity_1, result_{e1}), (entity_n, result_{en})$$

- As a Profile is executed over time, a series of Results starts to emerge

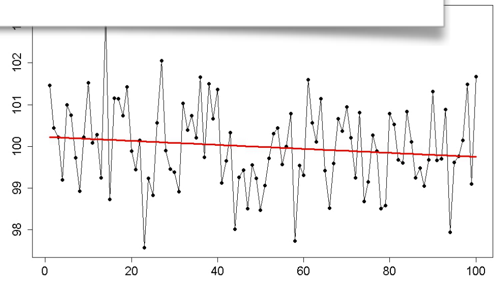
$$\begin{gathered} f(window_{t1}) \rightarrow (entity_1, result_{t1}), (entity_n, result_{t1}) \\ f(window_{t2}) \rightarrow (entity_1, result_{t2}), (entity_n, result_{t2}) \\ f(window_{tn}) \rightarrow (entity_1, result_{tn}), (entity_n, result_{tn}) \end{gathered}$$

- Rearranging the data slightly, we can see that a unique time series results for each (Profile, Entity) pair

$$(entity_1) \rightarrow result_{t_1}, result_{t_2}, result_{t_n}$$



$$(entity_n) \rightarrow result_{t1}, result_{t2}, result_{tn}$$



Hello World

“Hello World” Profile

- A Profile that counts the number of telemetry messages for each IP source address

```
{
 "profile": “hello-world",
 "foreach": "ip_src_addr",
 "init":    { "count": "0" },
 "update":  { "count": "count + 1" },
 "result":  “count"
}
```

“Hello World” Profile

- A Profile that counts the number of telemetry messages for each IP source address

```
{
 "profile": “hello-world",
 "foreach": "ip_src_addr",
 "init":    { "count": "0" },
 "update":  { "count": "count + 1" },
 "result":  “count"
}
```

Name; identifier for the profile

“Hello World” Profile

- A Profile that counts the number of telemetry messages for each IP source address

```
{
 "profile": “hello-world",
 "foreach": "ip_src_addr",
 "init":    { "count": "0" },
 "update":  { "count": "count + 1" },
 "result":  “count"
}
```

Name; identifier for the profile

Entity; a Stellar expression

Maintain a count for each unique source IP address

“Hello World” Profile

- A Profile that counts the number of telemetry messages for each IP source address

```
{
 "profile": “hello-world",
 "foreach": "ip_src_addr",
 "init":    { "count": "0" },
 "update":  { "count": "count + 1" },
 "result":  “count"
}
```

Name; identifier for the profile

Entity; a Stellar expression

Maintain a count for each unique source IP address

Initialize a counter at the start of each window

“Hello World” Profile

- A Profile that counts the number of telemetry messages for each IP source address

```
{
 "profile": “hello-world",
 "foreach": "ip_src_addr",
 "init":    { "count": "0" },
 "update":  { "count": "count + 1" },
 "result":  “count"
}
```

Name; identifier for the profile

Entity; a Stellar expression

Maintain a count for each unique source IP address

Initialize a counter at the start of each window

Increment the counter for each message in the window

HORTONWORKS®

“Hello World” Profile

- A Profile that counts the number of telemetry messages for each IP source address

```
{
 "profile": “hello-world",
 "foreach": "ip_src_addr",
 "init":    { "count": "0" },
 "update":  { "count": "count + 1" },
 "result":  “count"
}
```

Name; identifier for the profile

Entity; a Stellar expression

Maintain a count for each unique source IP address

Initialize a counter at the start of each window

Increment the counter for each message in the window

Return the count at the end of the window

HORTONWORKS®

"Hello World" Profile

- A Profile that counts the number of telemetry messages for each IP source address

```
{
 "profile": “hello-world",
 "foreach": "ip_src_addr",
 "init":    { "count": "0" },
 "update":  { "count": "count + 1" },
 "result":  “count"
}
```

Name; identifier for the profile

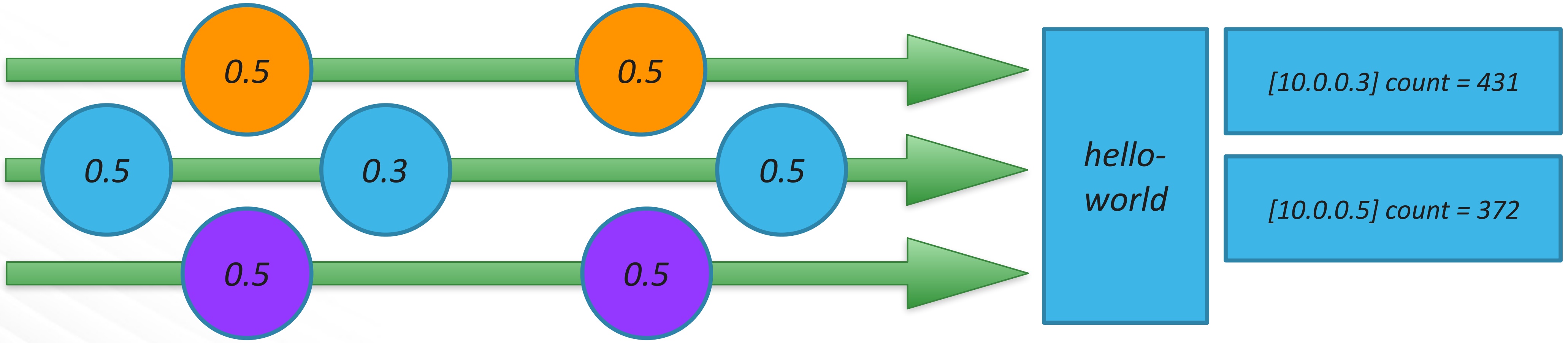
Entity; a Stellar expression

Maintain a count for each unique source IP address

Initialize a counter at the start of each window

Increment the counter for each message in the window

Return the count at the end of the window



“Hello World” Profile

- A Profile that counts the number of telemetry messages for each IP source address

```
{
 "profile": “hello-world",
 "foreach": "ip_src_addr",
 "init":    { "count": "0" },
 "update":  { "count": "count + 1" },
 "result":  “count"
}
```

Name; identifier for the profile

Entity; a Stellar expression

Maintain a count for each unique source IP address

Initialize a counter at the start of each window

Increment the counter for each message in the window

Return the count at the end of the window

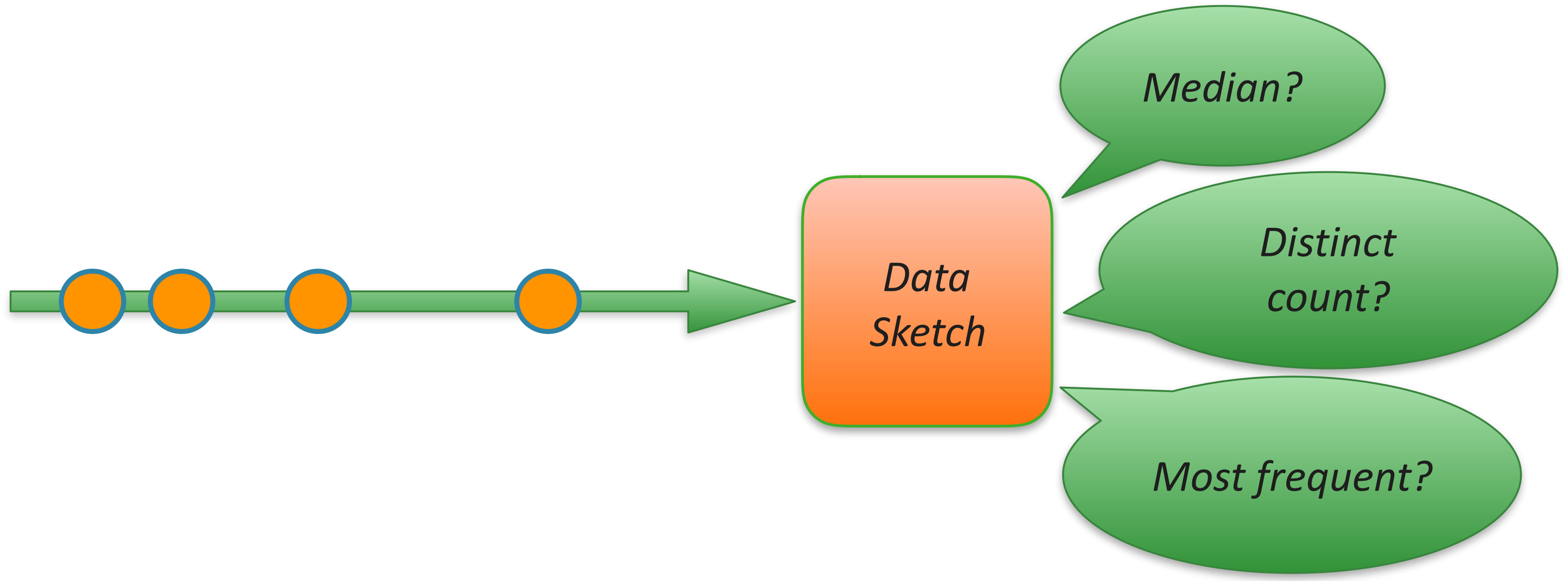
```
[Stellar]>>> PROFILE_GET(“hello-world", “10.0.0.3", PROFILE_FIXED(30, "MINUTES"))
[451, 448]

[Stellar]>>> PROFILE_GET(“hello-world", "10.0.0.5", PROFILE_FIXED(30, "MINUTES"))
[234, 176]
```

HORTONWORKS®

Data Sketches

Data Sketches



- Data Sketches provide fast, approximate answers to queries about the underlying data
- There are a variety of different types of data sketches, but general characteristics include
  - Stream Friendly - Each item of a stream, examined only once, can quickly update a small sketch data structure
  - Scalable - Effective for queries that do not scale well; count distinct, quantiles, most frequent items
  - Approximate, but with predictable error rates
  - Sub-Linear in Size - Required storage space grows more slowly than the input size
  - Mergeable (additive) and thus easily support parallelization
    - `query(sketch(data1 + data2)) == query(sketch(data1) + sketch(data2))`

Data Sketches and the Profiler

- The Profiler can persist anything serializable; not just numbers
- Profiler + Data Sketches
  - Allows the Profiler to scale to very large data sets
  - Allows consumers to ask different queries of the same profile data
  - Allows consumers to change the time horizon
  - Produce small, lightweight objects that can be stored efficiently

Data Sketches - Example

- A simple Profile that tracks URL length over time

```
{
  "profile": ”http-length”,
  "foreach": ”’global’",
  "onlyif": “source.type == ‘bro’ and protocol == 'HTTP'",
  "update": { "sk": "STATS_ADD(sk, length)" },
  "result": "sk"
}
```

- These aren’t just numbers

```
[Stellar]>>> stats := PROFILE_GET( “http-length", "global", PROFILE_FIXED(24, "HOURS"))
[Stellar]>>> stats
[org.apache.metron.common.math.stats.OnlineStatisticsProvider@79fe4ab9, ...]
```

- Ask different queries of the same data

```
[Stellar]>>> STATS_MEAN( GET_FIRST( stats))
15979.0625
[Stellar]>>> STATS_PERCENTILE( GET_FIRST(stats), 90)
30310.958
```

- Merge to change the time horizon

```
[Stellar]>>> merged := STATS_MERGE( stats)
[Stellar]>>> STATS_PERCENTILE(merged, 90)
29810.992
```

Profiles

Failed Logins

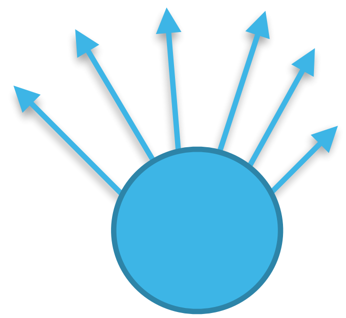
- A profile that tracks the number of bad logins by host

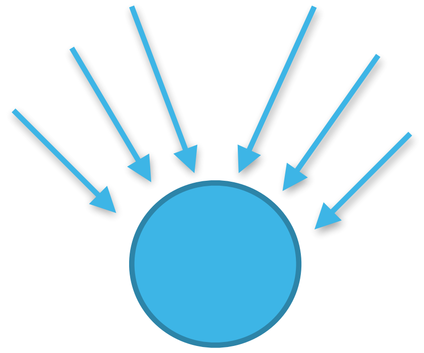
```
{
  "profile": ”bad-logins”,
  "foreach": ”ip_src_addr",
  "onlyif": ”source.type == ‘activedirectory’ and event.type == ‘failed_login’",
  “init” : { “count” : “0” }
  "update": { “count": ”count + 1" },
  "result": “count"
}
```

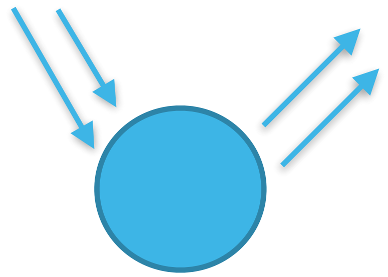
Vertex Degree

- View network communication as a directed graph
  - The in and out degree can distinguish behaviors
  - Anomalies over time can serve as an indicator of compromise

```
{
  "profile": “in-degree",
  "onlyif":  "source.type == 'yaf'"
  "foreach": "ip_dst_addr",
  "init":    { "in": "HLLP_INIT(5, 6)” }
  "update":  { "in": "HLLP_ADD(in, ip_src_addr)” }
  "result":  { “HLLP_CARDINALITY(in)” }
}
```


client
out >> in

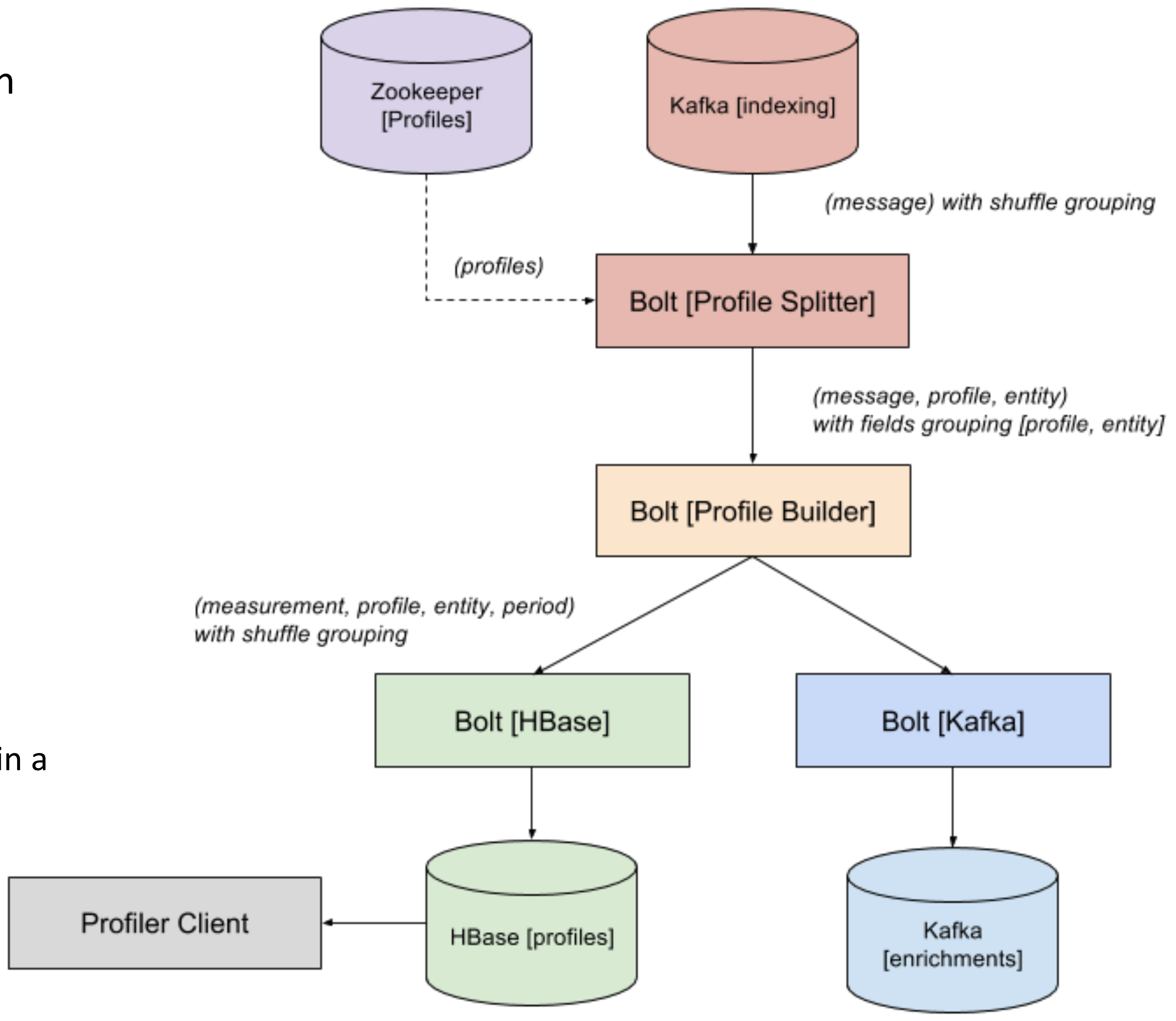

server
in >> out


middleware
in ~= out

Implementation

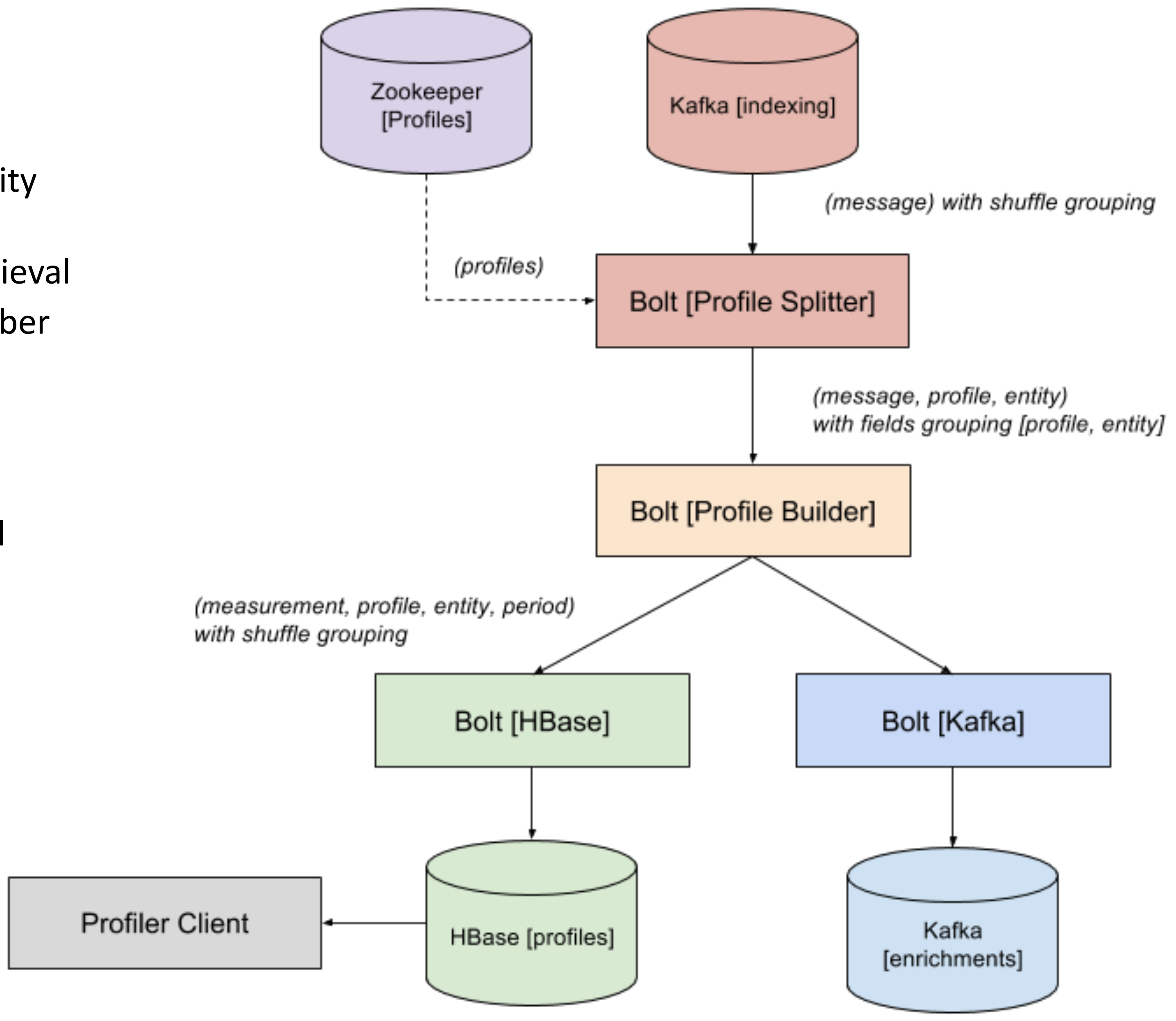
The Implementation

- A Storm topology that lives outside the critical path
- Profiles are defined in Zookeeper
- Profile Splitter Bolt
  - Reads all profile definitions in Zookeeper
  - Consumes each message and for each profile determines…
    - Is the message needed by the profile?
    - If needed, what is the entity?
  - Partitions the data by (Profile, Entity) using a fields grouping
- Profile Builder Bolt
  - Consumes the message, profile definition, and entity
  - Updates the state for a (Profile, Entity) pair
  - Flushes all state upon receiving a tick tuple resulting in a Profile Measurement



The Implementation

- HBase Bolt
  - Persists profile state in HBase
  - Generates a row key using a salt, profile name, entity and time
  - Row key needs to be deterministic to allow for retrieval
  - The time period is a monotonically increasing number identifying each period since the epoch
- Kafka Bolt
  - Pushes profile data back into Metron
  - Allows the user to create alerts based on abnormal profile values
- Profiler Client
  - Based on input parameters, calculates all of the necessary row keys
  - Submits a multi-get using these row keys
  - Some row keys may 'hit' and others 'miss'



Q&A

- Questions?

- Join the community
  - http://metron.apache.org/
  - https://github.com/apache/metron

- More information on the Profiler
  - http://metron.apache.org/current-book/metron-analytics/metron-profiler