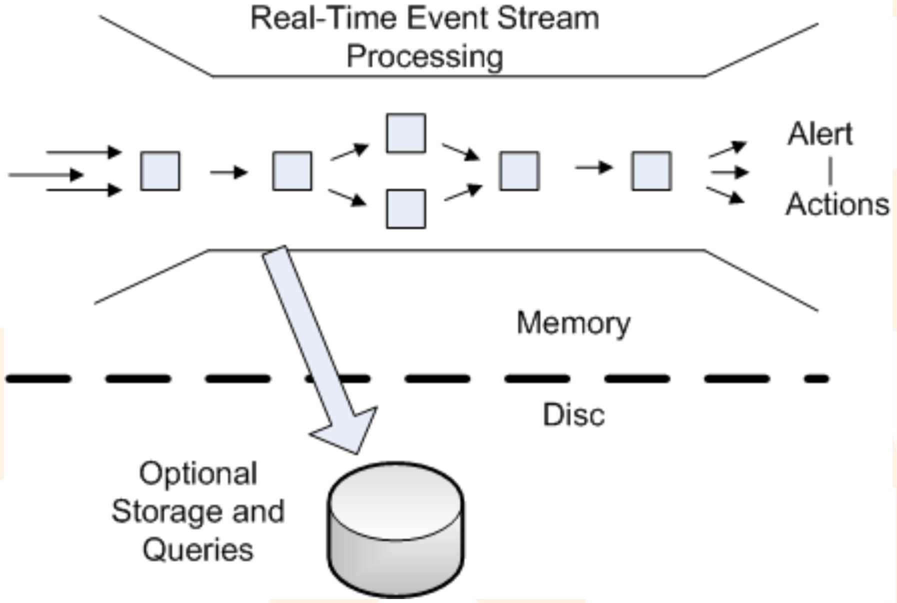
EXTENDA

CEP in Practice

Peter Norrhall – jDays 2012
peter.norrhall@extenda.se
@peternorrhall

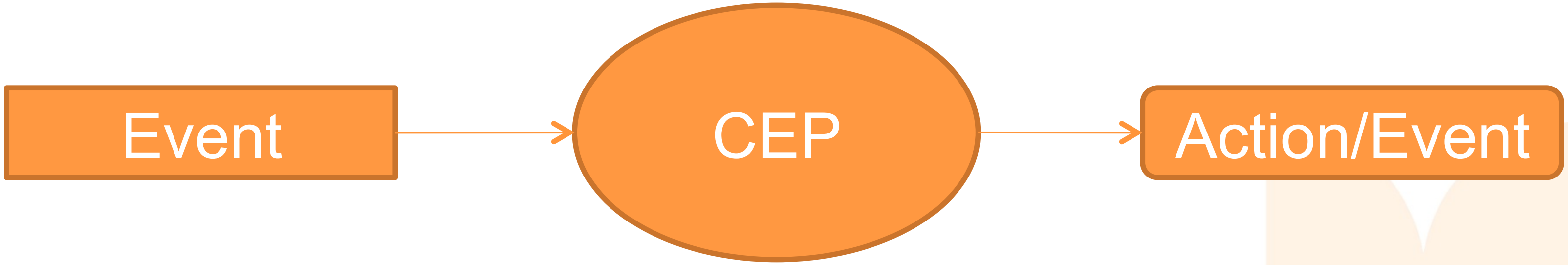
Agenda

- Complex Event Processing (CEP) an Introduction
- Alternative CEP Open Source frameworks
- Esper Essentials
- Extenda’s usage of Esper
- Additional Questions and Summary

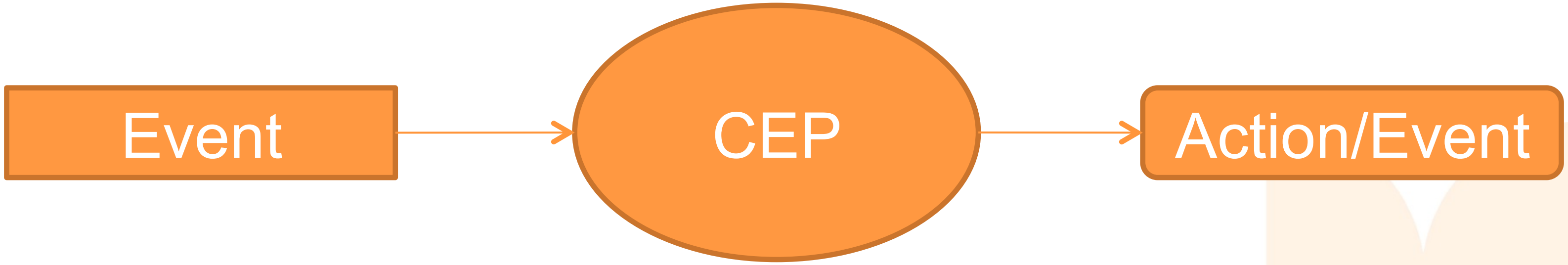

Real-Time Event Stream Processing
Alert | Actions
Memory
Disc
Optional Storage and Queries

Complex Event Processing (CEP)

- Analyze and react to *events*



Event Definition



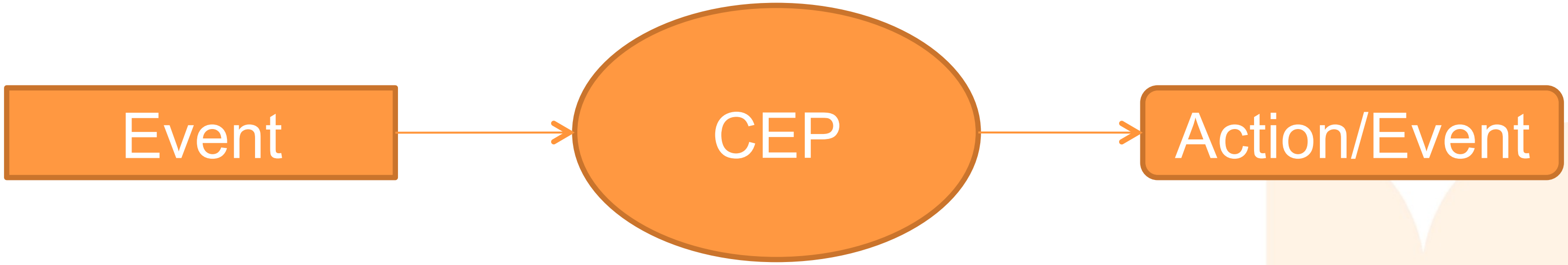
- Event
  - Physical Event
  - Event in a Programming/Logical Entity
  - Non-Event – Absence of an event

Why the word ”Complex”?

- **Simple Event Processing**
  - Acting on single event, filter in the ESP
- **Event Stream Processing**
  - Looking across multiple events
- **Complex Event Processing**
  - Looking across multiple event stream

Complex Event Processing (CEP)

- Analyze and react to *events*



BPM/BAM

Sensor Network App

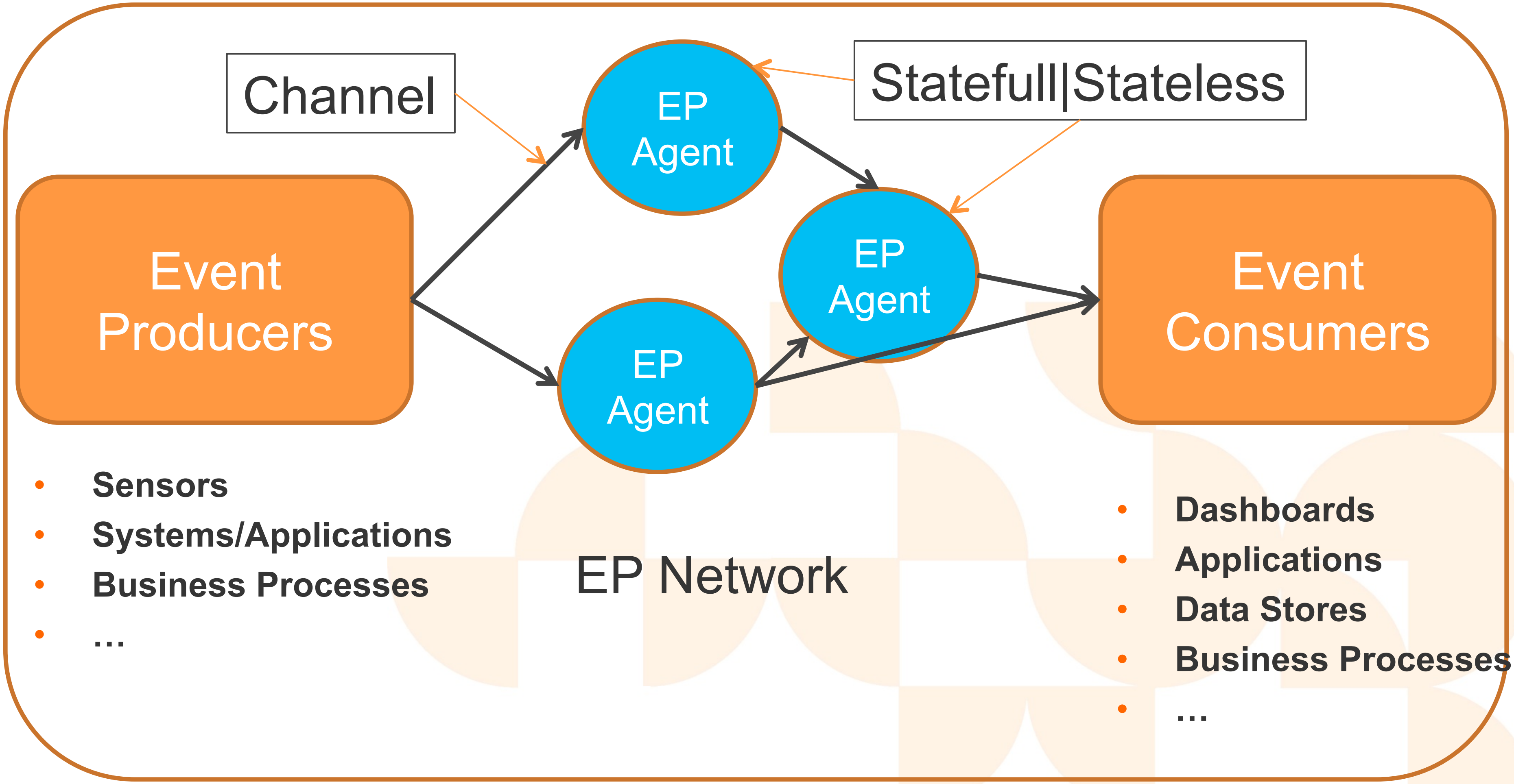
Finance (Trading/Fraud/Risk)

Network/App Monitoring

CEP – Event Stream Analysis

- High Throughput (1.000-100k events/s)
- Low Latency (ms – seconds)
- Complex Computations
  - Detect Patterns Among Events (Event Correlation)
  - Filter Events
  - Join Event Streams
  - Trigger on events and absence of events

Event Processing - Definitions



Channel

Statefull|Stateless

Event Producers

EP Agent

EP Agent

EP Agent

Event Consumers

- Sensors
- Systems/Applications
- Business Processes
- …

EP Network

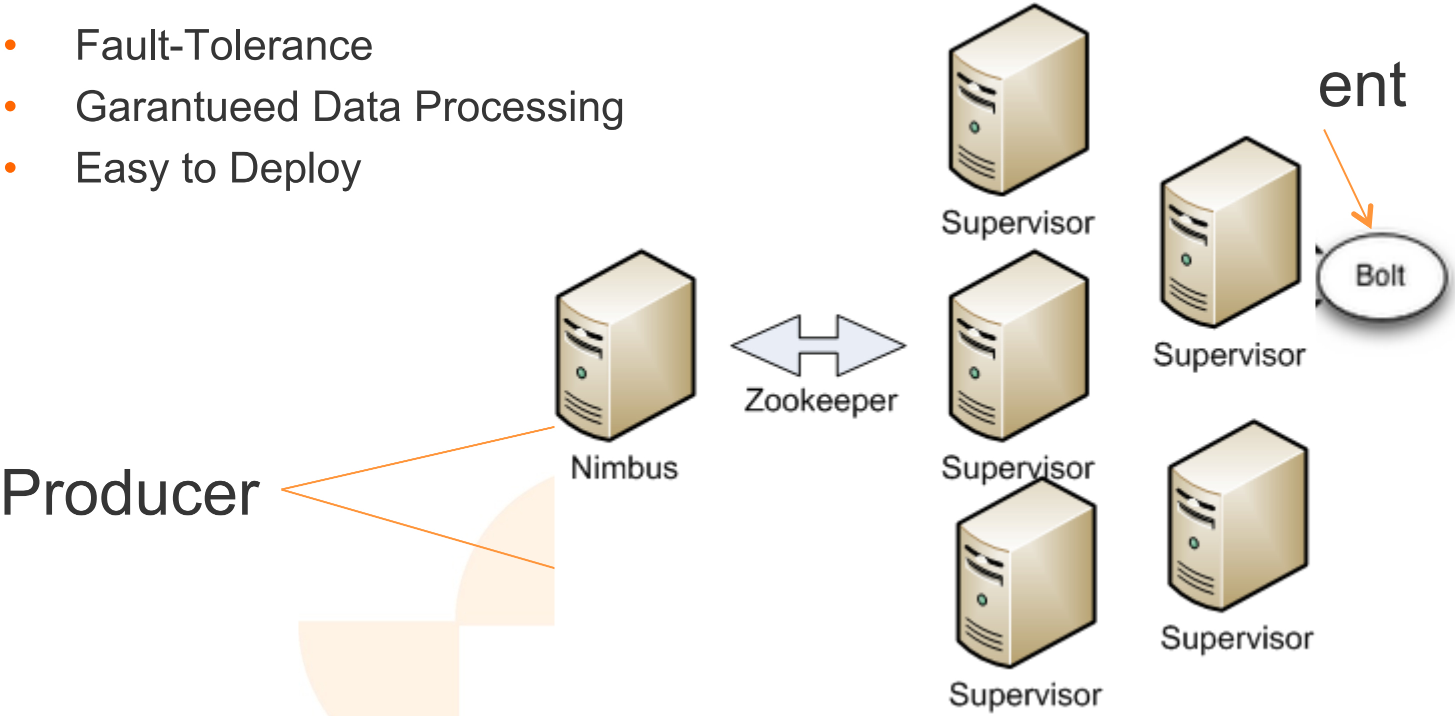
- Dashboards
- Applications
- Data Stores
- Business Processes
- …

Event Processing Languages

- Rule-oriented
  - Production (Drools)
  - Active
  - Logic
- Imperative (Storm/Disruptor)
- Stream-oriented (Esper)

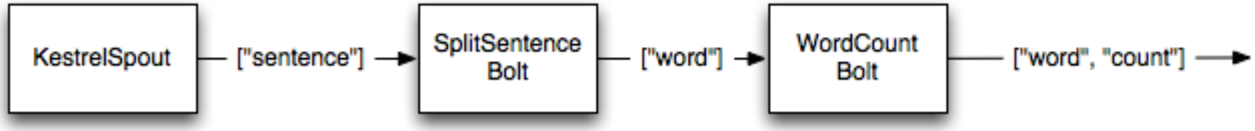
Storm – Twitter (Backtype)

- Scalable
- Fault-Tolerance
- Garantueed Data Processing
- Easy to Deploy



Storm Topology

BDD – WordCount story

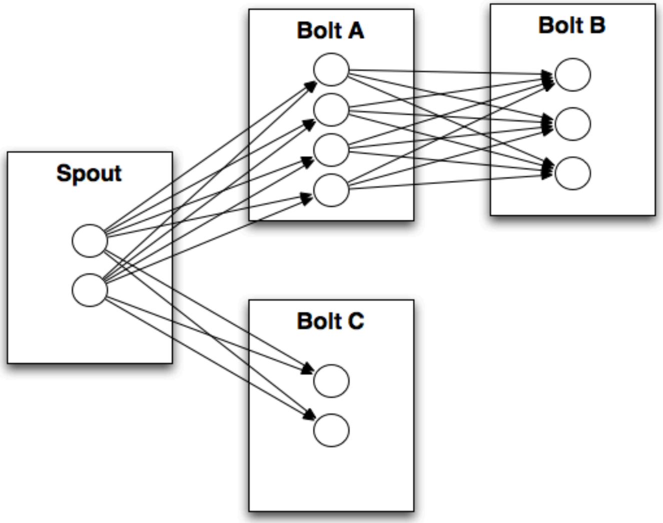


KestrelSpout → ["sentence"] → SplitSentence Bolt → ["word"] → WordCount Bolt → ["word", "count"] →

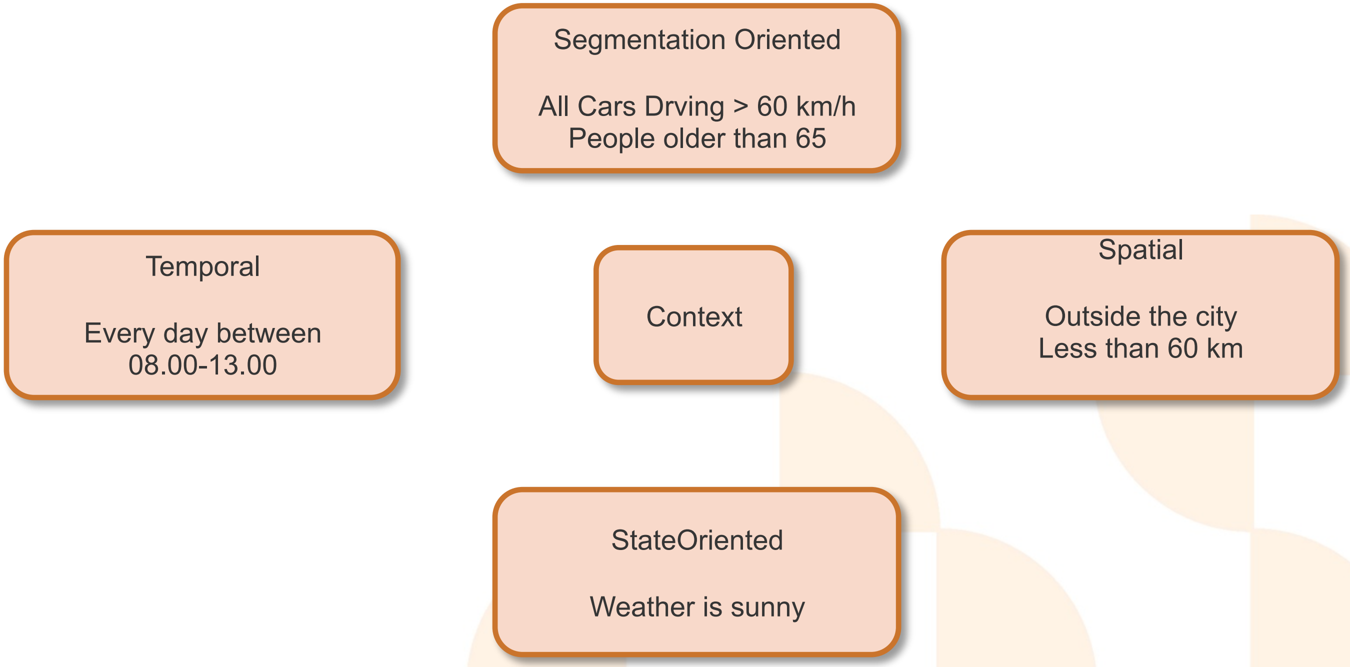
```
Word Count

Narrative:

In order to find out the most used words
As a Twitter Junkie
I want to get word count information

Scenario: Track word count out of randomly generated sentences

Given the sentences:
|sentence|
|the cow jumped over the moon|
|an apple a day keeps the doctor away|
|four score and seven years ago|
|snow white and the seven dwarfs|
|i am at two with nature|
When those are randomly selected and split 100 times
Then the words should be counted separately
```

Context

- A *context* is a named specification of conditions that **groups event instances** so that they can **be processed in a related way**. It assigns each event instance to one or more **context partitions** (windows). A context may have one or more context dimensions and can give rise to one or more context partitions.



Context Partitioning



Segmentation Oriented

All Cars Drving > 60 km/h
People older than 65

Temporal

Every day between
08.00-13.00

Context

Spatial

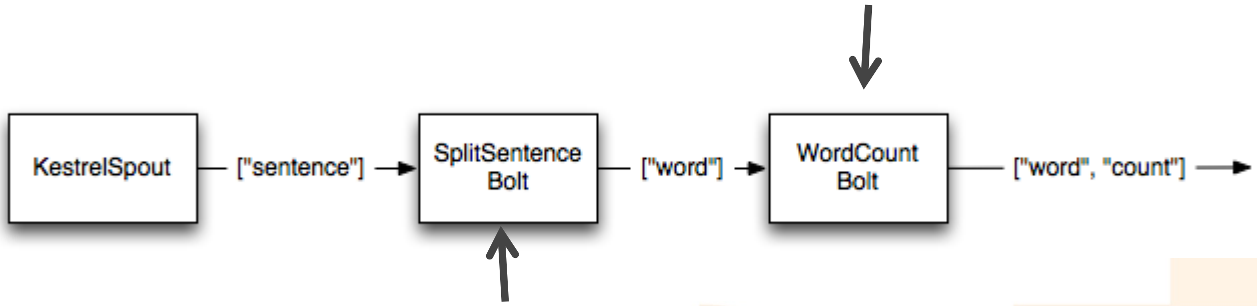
Outside the city
Less than 60 km

StateOriented

Weather is sunny

Topology & Context Partitioning

Field Grouping



Shuffle Grouping

```
TopologyBuilder builder = new TopologyBuilder();
builder.setSpout(1, new KestrelSpout("kestrel.backtype.com",
                                     22133,
                                     "sentence_queue",
                                     new StringScheme()));
builder.setBolt(2, new SplitSentence(), 10)
        .shuffleGrouping(1);
builder.setBolt(3, new WordCount(), 20)
        .fieldsGrouping(2, new Fields("word"))
```

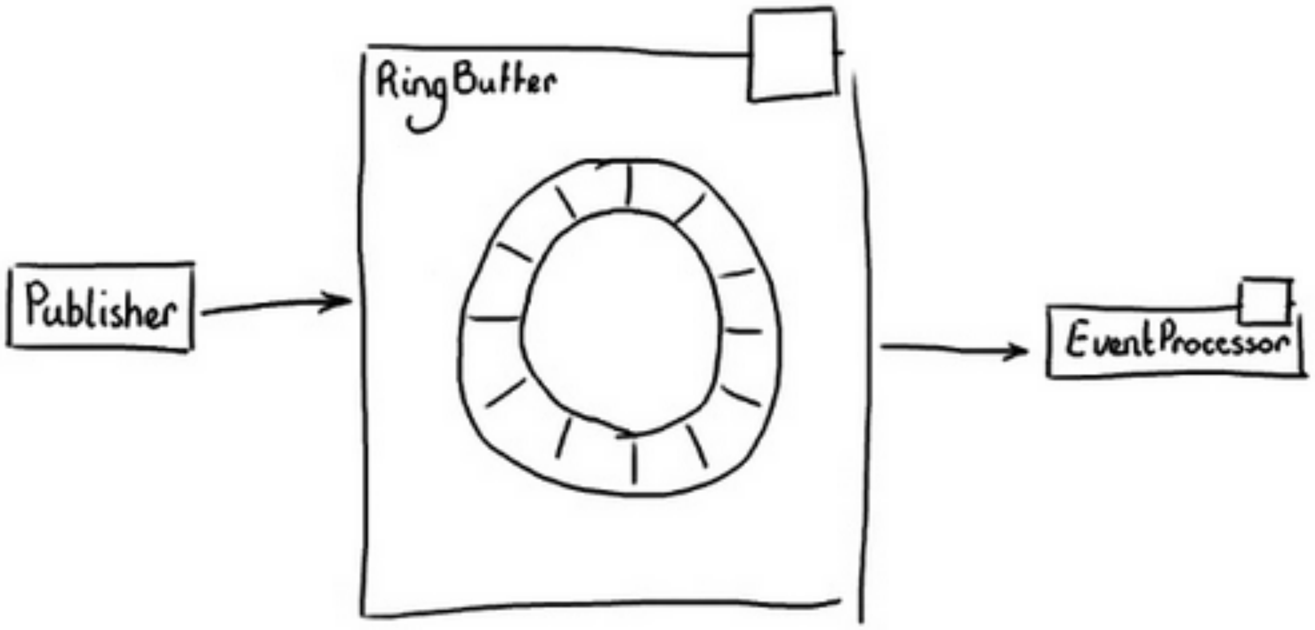
Demo – WordCount Storm

WordCountBolt

```
public static class WordCount extends BaseBasicBolt {
        Map<String, Integer> counts = new HashMap<String, Integer>();


        @Override
        public void execute(Tuple tuple, BasicOutputCollector collector)
            String word = tuple.getString(0);
            Integer count = counts.get(word);
            if(count==null) count = 0;
            count++;
            counts.put(word, count);
            collector.emit(new Values(word, count));
        }

        @Override
        public void declareOutputFields(OutputFieldsDeclarer declarer) {
            declarer.declare(new Fields("word", "count"));
        }
    }
```

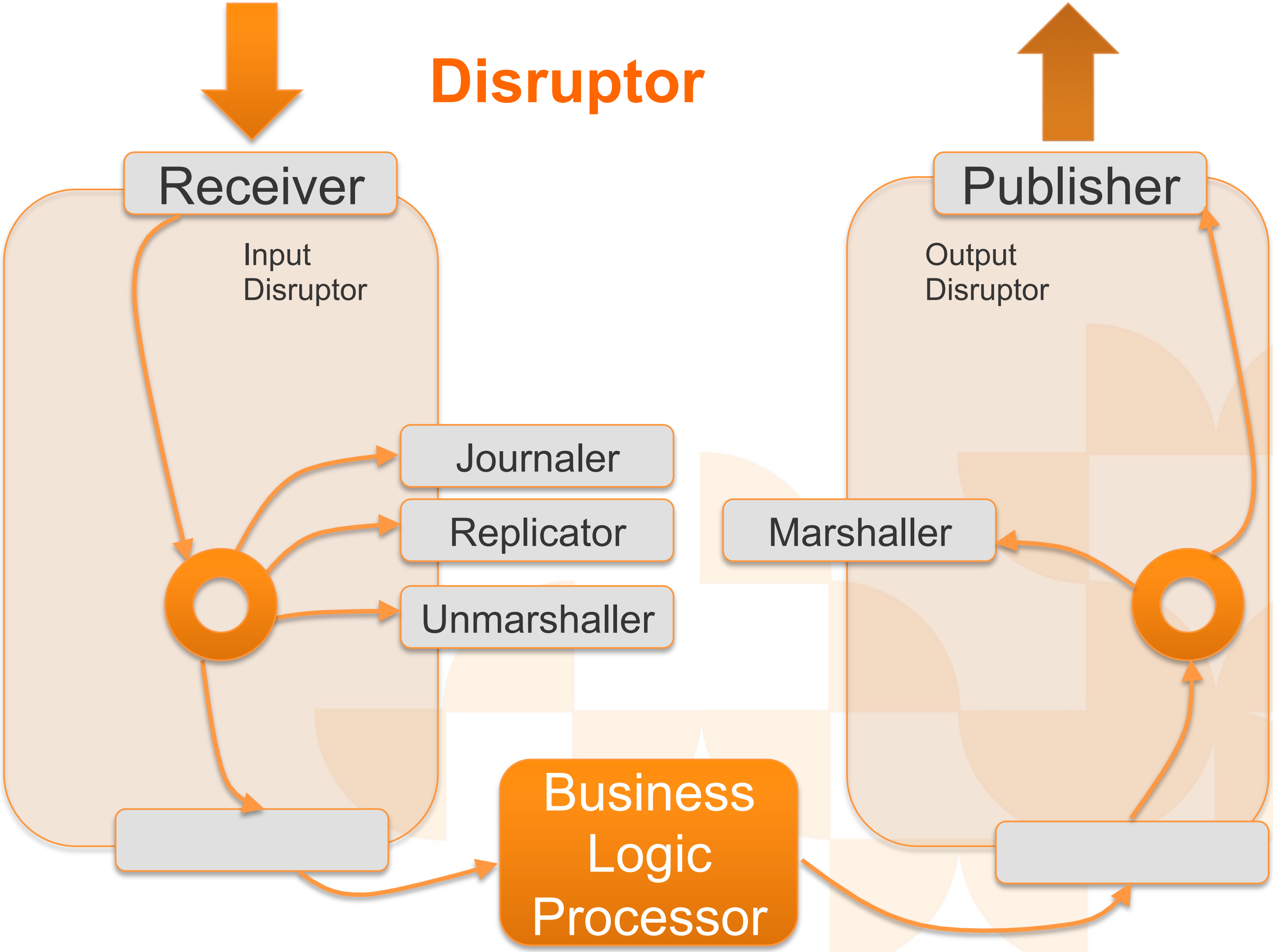
Disruptor

The Magic RingBuffer



http://www.slideshare.net/trishagee/introduction-to-the-disruptor

Disruptor



Receiver

Input Disruptor

Journaler

Replicator

Unmarshaller

Business Logic Processor

Marshaller

Publisher

Output Disruptor

Event Sourcing

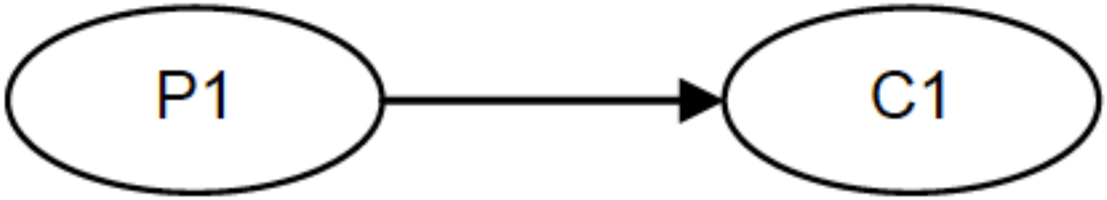
Event Sourcing ensures that all changes to application state are stored as a sequence of events.

Not just can we query these events,

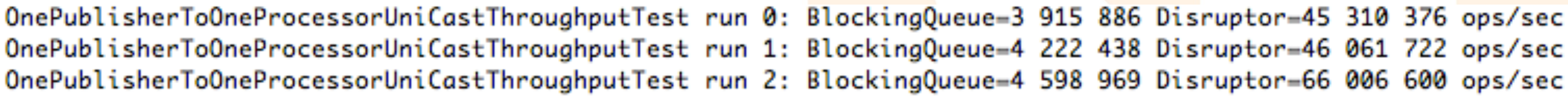
- we can also use the event log to reconstruct past states,
- and as a foundation to automatically adjust the state to cope with retroactive changes.

http://martinfowler.com/eaaDev/EventSourcing.html

Disruptor

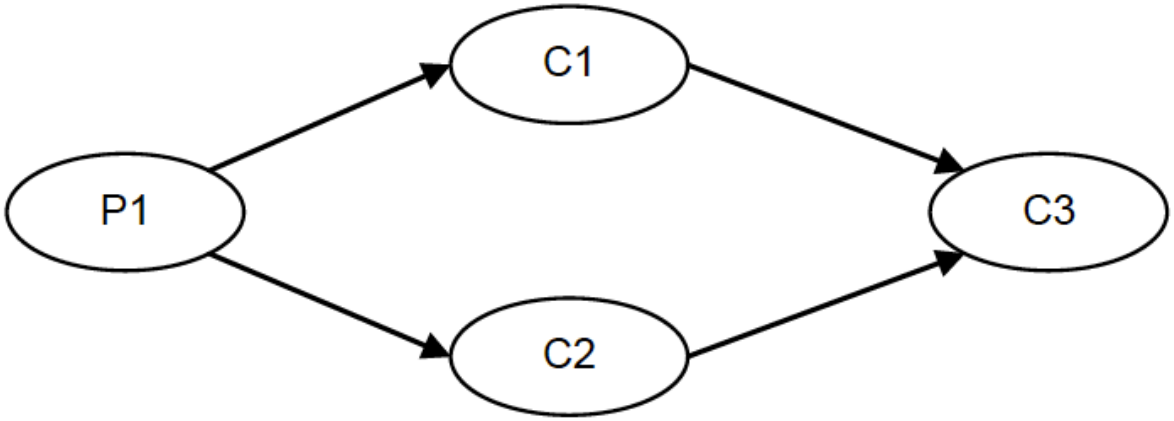


Unicast: 1P – 1C

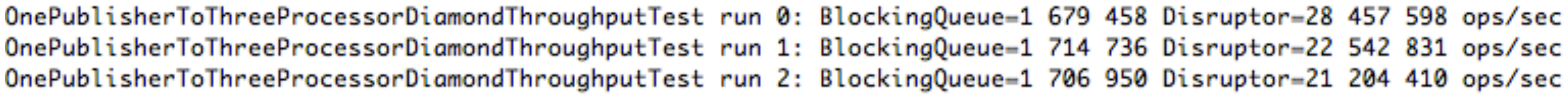


```
OnePublisherToOneProcessorUniCastThroughputTest run 0: BlockingQueue=3 915 886 Disruptor=45 310 376 ops/sec
OnePublisherToOneProcessorUniCastThroughputTest run 1: BlockingQueue=4 222 438 Disruptor=46 061 722 ops/sec
OnePublisherToOneProcessorUniCastThroughputTest run 2: BlockingQueue=4 598 969 Disruptor=66 006 600 ops/sec
```

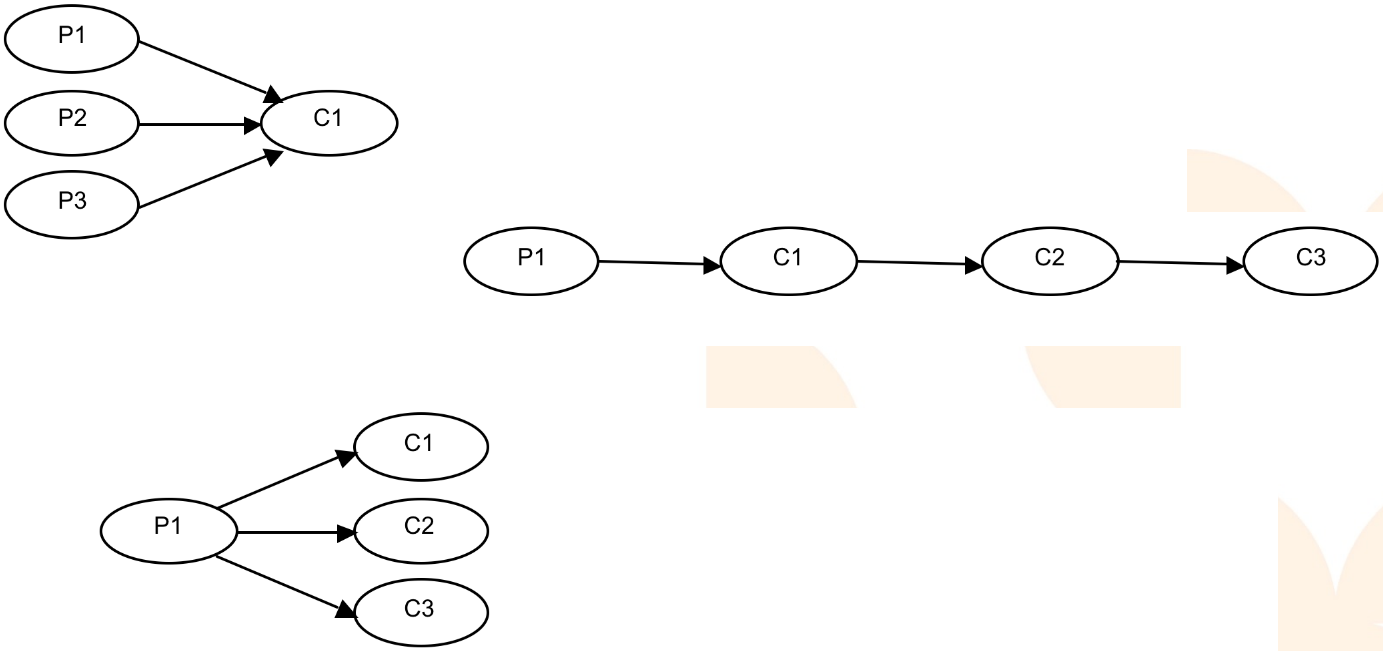
MacBook Pro i7



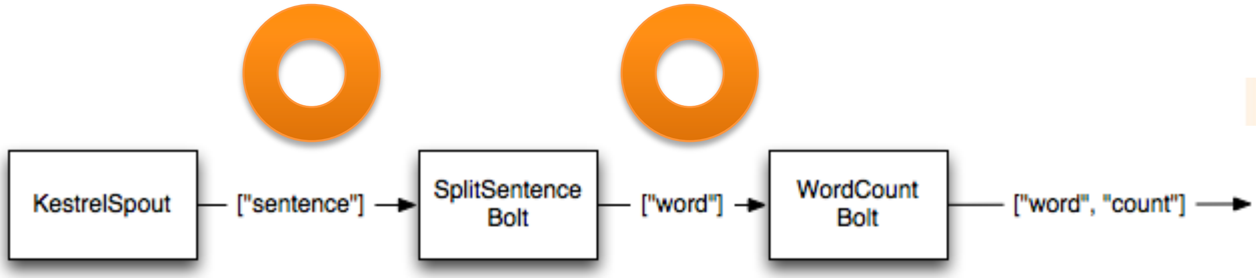
Diamond: 1P – 3C


```
OnePublisherToThreeProcessorDiamondThroughputTest run 0: BlockingQueue=1 679 458 Disruptor=28 457 598 ops/sec
OnePublisherToThreeProcessorDiamondThroughputTest run 1: BlockingQueue=1 714 736 Disruptor=22 542 831 ops/sec
OnePublisherToThreeProcessorDiamondThroughputTest run 2: BlockingQueue=1 706 950 Disruptor=21 204 410 ops/sec
```

Other sample patterns



Demo – WordCount with Disruptor



Comparison

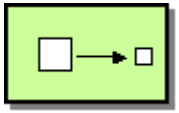
Framework/Method	Sentences / second
Storm	896
Disruptor	802568

Storm/Disruptor

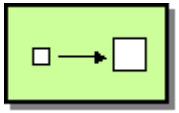
- ✓ High Throughput (1.000-100k events/s)
- ✓ Low Latency (ms – seconds)
- Complex Computations
  - Detect Patterns Among Events (Event Correlation)
  - ~ Filter Events
  - Join Event Streams
  - ✗ Trigger on absence of events

Computation (Patterns)

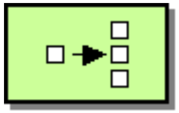
Stateless



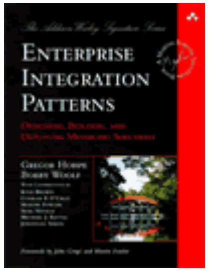
Filter



Enricher



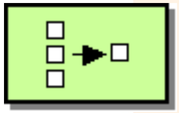
Splitter



Enterprise Integration Patterns

Statefull - Patterns

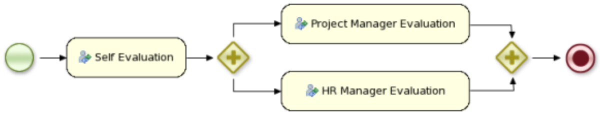
- Business Event Processing (BEP)
- Event Stream Processing (ESP)



Aggregator

Drools – Business Event Processing

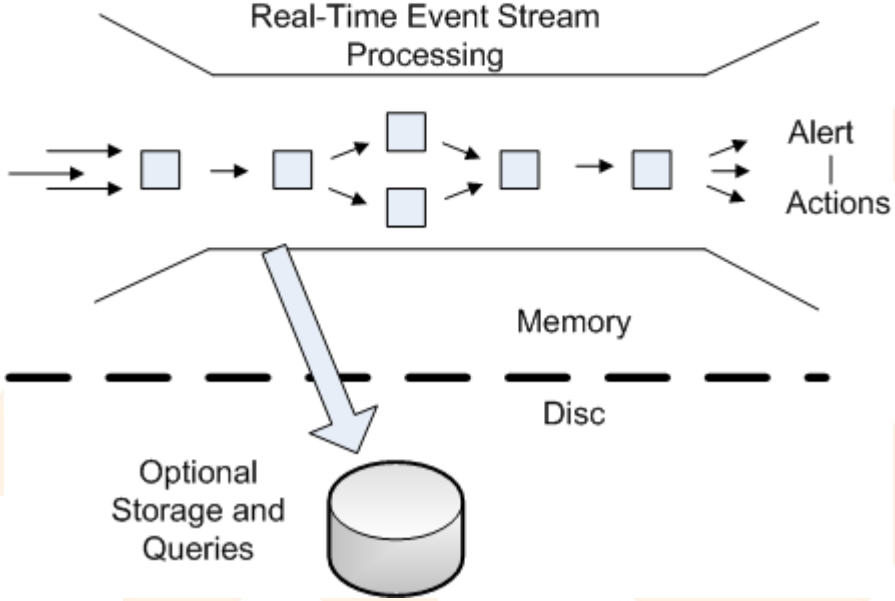
- Drools Guvnor – Business Rules Manager
- Drools Expert – Rule Engine
- jBPM – Process / Workflows



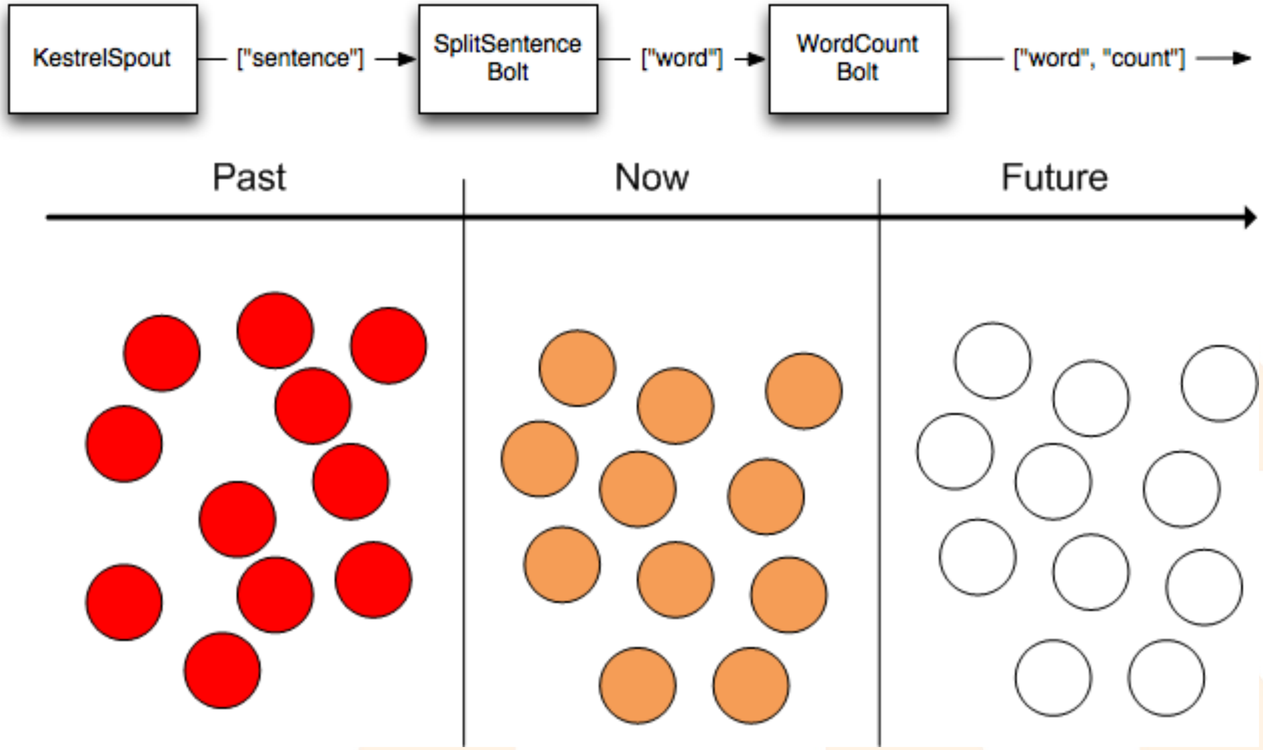
- **Drools Fusion – Event Processing**

Esper – Event Stream Processing

- Pattern Detect – Event Processing Language (EPL)
  1. Filtering
  2. Matching
  3. Derivation


Real-Time Event Stream Processing
Alert
|
Actions
Memory
Disc
Optional Storage and Queries

Event Stream vs SQL



```
SQL: select word, count(*)
from WordEvent
group by word
```

```
EPL: select word, count(*)
from WordEvent:win:time(1 min)
group by word
```

Esper Building Blocks

- Event
- Statement
- Sending Events
- UpdateListener

(Immutable) Event – POJO or Schema

```
public class WordEvent {
        private final String word;
        public WordEvent(String word) {
                this.word = word;
        }
        public String getWord() { return word; }
}
```

or

```
create schema WordEvent as (word string)
```

Statement (Stream)

```
@Name("WordCountBolt”)
select word, count(*) as count from WordEvent group by word

EPStatement wordCountStmt =
  epService.getEPAdministrator().createEPL(wordEventStmt);
```

Sending Events

```
epService.getEPRuntime().sendEvent(new WordEvent(word));
```

Or

```
Map<String,String> map = new HashMap<String, String>();
map.put("word", word);
epService.getEPRuntime().sendEvent(map, "WordEvent");
```

Or

Using EsperIO

UpdateListener

```
epService.getEPAdministrator().getStatement("WordCountBolt").addListener(wordCountListener);

public class WordCountListener implements UpdateListener {
  @Override
  public void update(EventBean[] newEvents,
                     EventBean[] oldEvents) {
    System.out.println("Word ” + newEvents[0].get("word") + " : " +
        newEvents[0].get("count"));
  }
}
...
Word keeps : 29
Word the : 145
Word doctor : 29
...
```

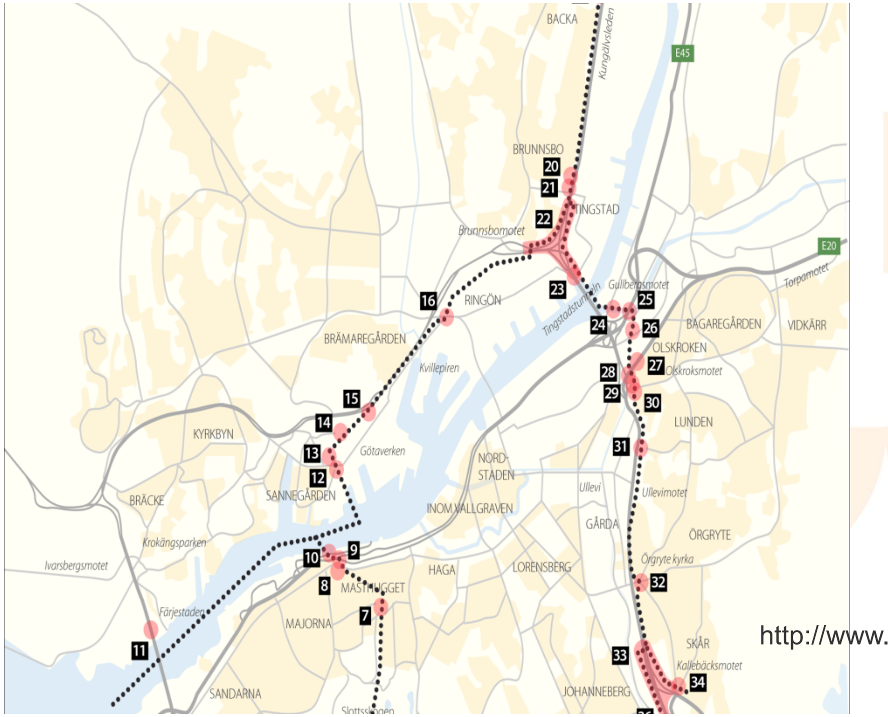
Demo – WordCount Esper

Summary

Framework/Method	Sentences / second
Storm	896
Esper	77399
Disruptor	802568
SingleThread	853242

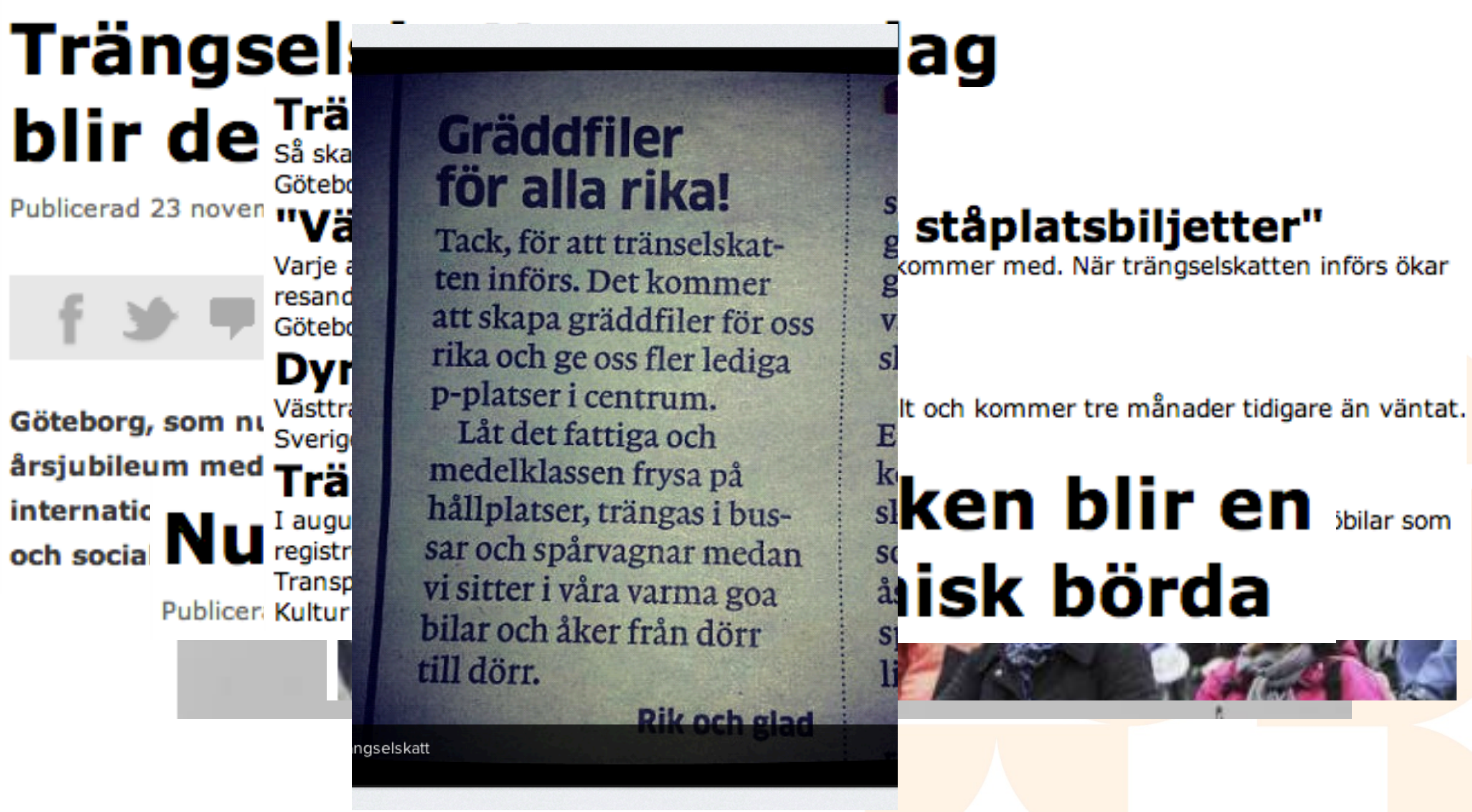
Keep it simple, stupid!

Traffic Jam Tax (Trängselskatt)



http://www.theredlinereport.com/2012/05/29/dead-traffic-light-leads-to-beijing-traffic-jam/

It engages people



Trängsel... ag

blir de

Publicerad 23 noven

Göteborg, som nu
årsjubileum med
internatic
och socia

Nu

Publicer

Trä
Så ska
Götebo

"Vä
Varje a
resand
Götebo

Dyr
Västtr
Sverig

Trä
I augu
registr
Transp
Kultur

Gräddfiler för alla rika!

Tack, för att tränselskatten införs. Det kommer att skapa gräddfiler för oss rika och ge oss fler lediga p-platser i centrum.

Låt det fattiga och medelklassen frysa på hållplatser, trängas i bussar och spårvagnar medan vi sitter i våra varma goa bilar och åker från dörr till dörr.

Rik och glad

ngselskatt

ståplatsbiljetter"

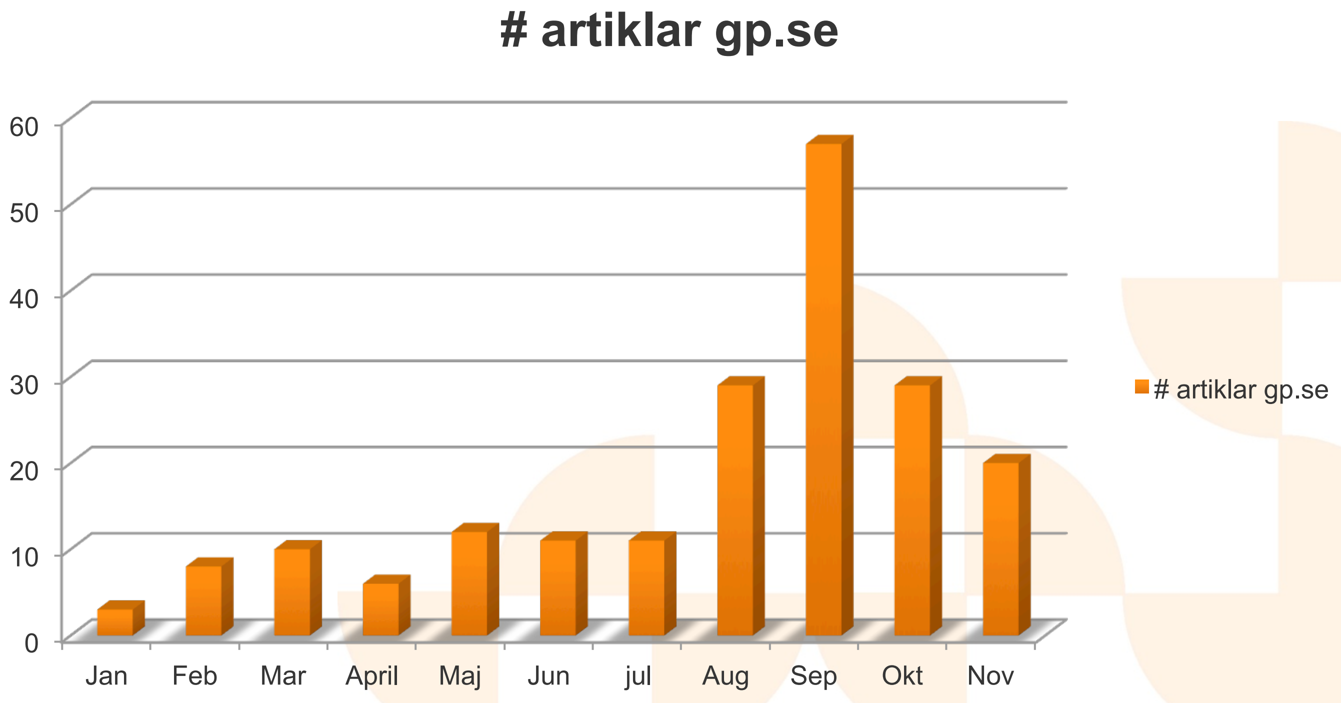
kommer med. När trängselskatten införs ökar

lt och kommer tre månader tidigare än väntat.

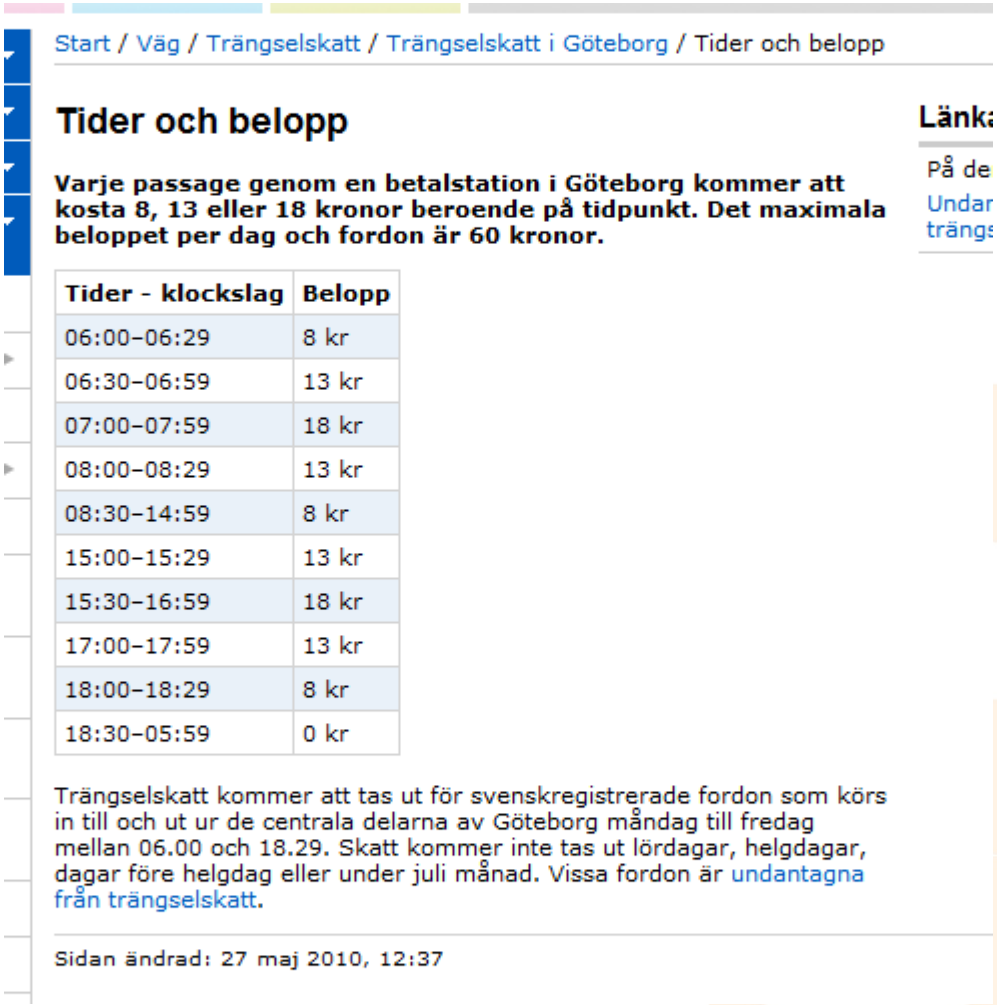
ken blir en
isk börda

bilar som

Key word – ”trängselskatt”



Business Rules



Start / Väg / Trängselskatt / Trängselskatt i Göteborg / Tider och belopp

Tider och belopp

Varje passage genom en betalstation i Göteborg kommer att kosta 8, 13 eller 18 kronor beroende på tidpunkt. Det maximala beloppet per dag och fordon är 60 kronor.

Tider - klockslag	Belopp
06:00–06:29	8 kr
06:30–06:59	13 kr
07:00–07:59	18 kr
08:00–08:29	13 kr
08:30–14:59	8 kr
15:00–15:29	13 kr
15:30–16:59	18 kr
17:00–17:59	13 kr
18:00–18:29	8 kr
18:30–05:59	0 kr

Trängselskatt kommer att tas ut för svenskregistrerade fordon som körs in till och ut ur de centrala delarna av Göteborg måndag till fredag mellan 06.00 och 18.29. Skatt kommer inte tas ut lördagar, helgdagar, dagar före helgdag eller under juli månad. Vissa fordon är undantagna från trängselskatt.

Sidan ändrad: 27 maj 2010, 12:37

Länka

På de

Undar
trängs

Business Rules as BDD

Vehicle Billing

Narrative:

In order to track the daily toll road billing
As a Vehicle Owner
I want to get notified via SMS in real time of the daily billing status

Billing

```
Scenario: Billing notification on work days

Given I am driving a car having <registrationNumber>
When I pass toll road at <date> and <time>
Then I should be notified with a billing statement of <billingamount> SEK for <date>

Examples:
registrationNumber|date|time|billingamount
PYZ123|2012-09-12|05:59|0
PYZ123|2012-09-12|06:00|8
PYZ123|2012-09-12|06:29|8
PYZ123|2012-09-12|06:30|13
PYZ123|2012-09-12|06:59|13
PYZ123|2012-09-12|07:00|18
PYZ123|2012-09-12|07:59|18
PYZ123|2012-09-12|08:00|13
PYZ123|2012-09-12|08:29|13
PYZ123|2012-09-12|08:30|8
PYZ123|2012-09-12|14:59|8
PYZ123|2012-09-12|15:00|13
PYZ123|2012-09-12|15:29|13
PYZ123|2012-09-12|15:30|18
PYZ123|2012-09-12|16:59|18
PYZ123|2012-09-12|17:00|13
PYZ123|2012-09-12|17:59|13
PYZ123|2012-09-12|18:00|8
PYZ123|2012-09-12|18:29|8
PYZ123|2012-09-12|18:30|0
PYZ123|2012-09-09|07:45|0
PYZ123|2012-07-09|07:45|0
```

TollBillingStatement

```
@Name("TollBillingStatment"
select *, BillingService.getBillingAmount(time) as billingAmount from TollEvent
```

Static library function call

Static Java library functions

```
// Import the static helper class BillingService
config.addImport(BillingService.class);

// Static helper class
public class BillingService {

public static int getBillingAmount(Calendar time) {
  int result = 0;
  int dayOfWeek = time.get(Calendar.DAY_OF_WEEK);
  if (((dayOfWeek >= Calendar.MONDAY) && (dayOfWeek
         Calendar.FRIDAY)) &&
         (time.get(Calendar.MONTH) != Calendar.JULY)) {
  int hour = time.get(Calendar.HOUR_OF_DAY);
  int minute = time.get(Calendar.MINUTE);
  if (hour < 6) {
    result = 0;
...
}
```

TollEvent

```
@Name(“TollBillingStatment”
select *, BillingService.getBillingAmount(time) as billingAmount from TollEvent
```

Control the timer at test

```
@When("I pass toll road at $date and $time")
@Alias("I pass toll road at <date> and <time>")
public void iPassTollRoadAt(@Named("date") String date,
         @Named("time")String time) throws Exception {

  trafficControllerService.setInternalTimer(getCalendar(date,
         time).getTimeInMillis());

  TollEvent tollEvent = new TollEvent(registrationNumber, date,   getCalendar(date,
time), "1");

  trafficControllerService.logTrafficeEvent(tollEvent);
}
```

TotalBilling

```
Scenario: Total Billing notification on work days

Given I am driving a car with PYZ123
When I pass toll road at 2012-09-12 and 05:50
And I pass toll road at 2012-09-12 and 07:10
And I pass toll road at 2012-09-12 and 08:30
And I pass toll road at 2012-09-12 and 16:44
Then I should be notified with a total billing statement of 44 SEK for 2012-09-12
```

TotalBilling

```
Scenario: Total Billing should not exceed 60 SEK per day

Given I am driving a car with PYZ123
When I pass toll road at 2012-09-12 and 05:50
And I pass toll road at 2012-09-12 and 07:10
And I pass toll road at 2012-09-12 and 07:20
And I pass toll road at 2012-09-12 and 08:30
And I pass toll road at 2012-09-12 and 16:44
Then I should be notified with a total billing statement of 60 SEK for 2012-09-12
```

TotalBilling

```
Scenario: Total Billing should be billed for each day

Given I am driving a vehicle with PYZ123
When I pass toll road at 2012-09-12 and 05:50
And I pass toll road at 2012-09-12 and 07:10
And I pass toll road at 2012-09-12 and 08:30
And I pass toll road at 2012-09-12 and 16:44
Then I should be notified with a total billing statement of 44 SEK for 2012-09-12
When I pass toll road at 2012-09-13 and 07:10
Then I should be notified with a total billing statement of 18 SEK for 2012-09-13
When I pass toll road at 2012-09-14 and 08:30
And I pass toll road at 2012-09-14 and 16:44
Then I should be notified with a total billing statement of 26 SEK for 2012-09-14
```

Events are automatically removed after 24 hours

TotalBilling – Named Window

```
create window VehicleBillingWindow.win:time(24 hours) as select date,
registrationNumber, totalBilling from VehicleBilling
```

Named *window* is a global data window ≈ SQL Table

TotalBilling – Named Window

```
create window VehicleBillingWindow.win:time(24 hours) as select date,
registrationNumber, totalBilling from VehicleBilling
```

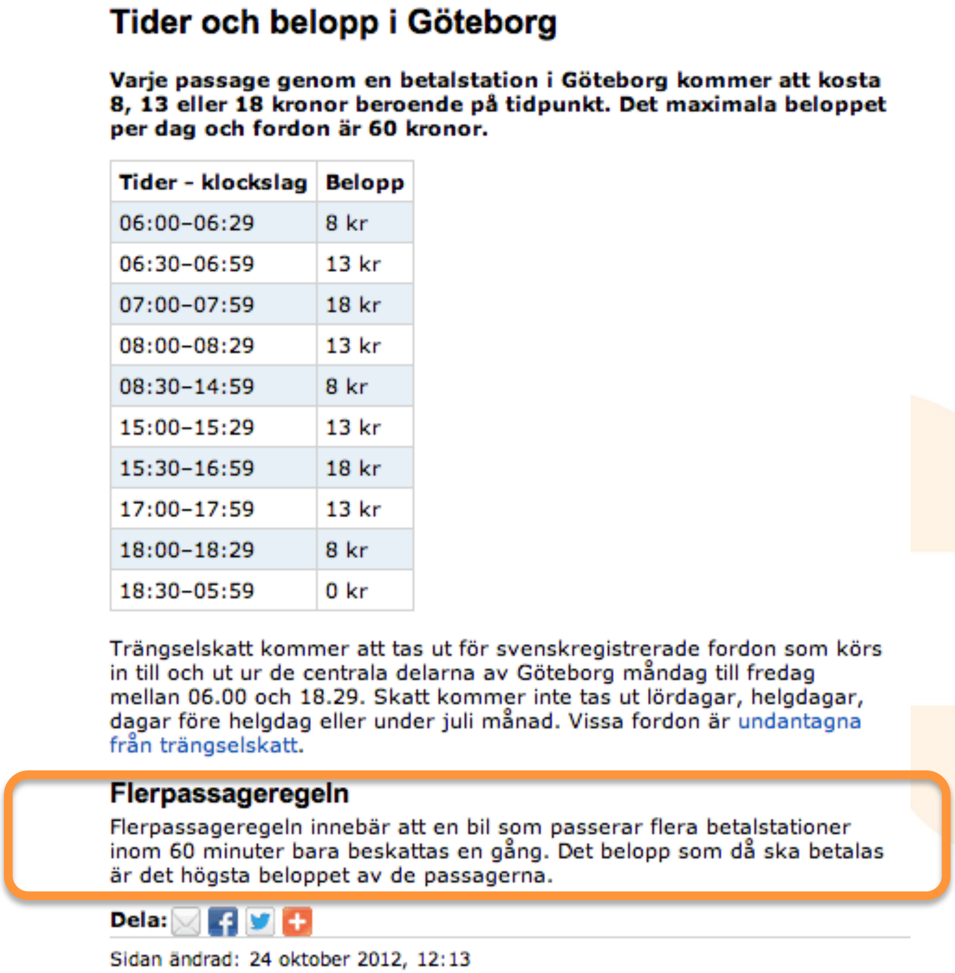
Named *window* handles mutable events

```
on TollEvent te
merge VehicleBillingWindow vbw
where te.registrationNumber = vbw.registrationNumber and te.date = vbw.date
  when matched and totalBilling < 60 then
    update set totalBilling =
         Math.min(totalBilling + BillingService.getBillingAmount(te.time), 60)
  when not matched then
    insert select date, registrationNumber,
         BillingService.getBillingAmount(time) as totalBilling
```

On-Demand Queries – ”Normal” SQL

```
public boolean validate(String registrationNumber, String date, int totalBilling) {
  boolean result = false;
  String query = "select * from VehicleBillingWindow where registrationNumber = \'" +
          registrationNumber + "\' and date = \'" + date + "\'";
  EPOnDemandQueryResult qryResult = serviceProvider.getEPRuntime().executeQuery(query);


  for (EventBean row : qryResult.getArray()) {
    String _registrationNumber = (String)row.get("registrationNumber");
    String _date = (String)row.get("date");
    if (registrationNumber.equalsIgnoreCase(_registrationNumber) &&
          (date.equalsIgnoreCase(_date))) {
      int _billing = (Integer)row.get("totalBilling");
      result = _billing == totalBilling;
      break;
    }
  }

  return result;
}
```

Changed Business Requirement



Tider och belopp i Göteborg

Varje passage genom en betalstation i Göteborg kommer att kosta 8, 13 eller 18 kronor beroende på tidpunkt. Det maximala beloppet per dag och fordon är 60 kronor.

Tider - klockslag	Belopp
06:00–06:29	8 kr
06:30–06:59	13 kr
07:00–07:59	18 kr
08:00–08:29	13 kr
08:30–14:59	8 kr
15:00–15:29	13 kr
15:30–16:59	18 kr
17:00–17:59	13 kr
18:00–18:29	8 kr
18:30–05:59	0 kr

Trängselskatt kommer att tas ut för svenskregistrerade fordon som körs in till och ut ur de centrala delarna av Göteborg måndag till fredag mellan 06.00 och 18.29. Skatt kommer inte tas ut lördagar, helgdagar, dagar före helgdag eller under juli månad. Vissa fordon är undantagna från trängselskatt.

Flerpassageregeln

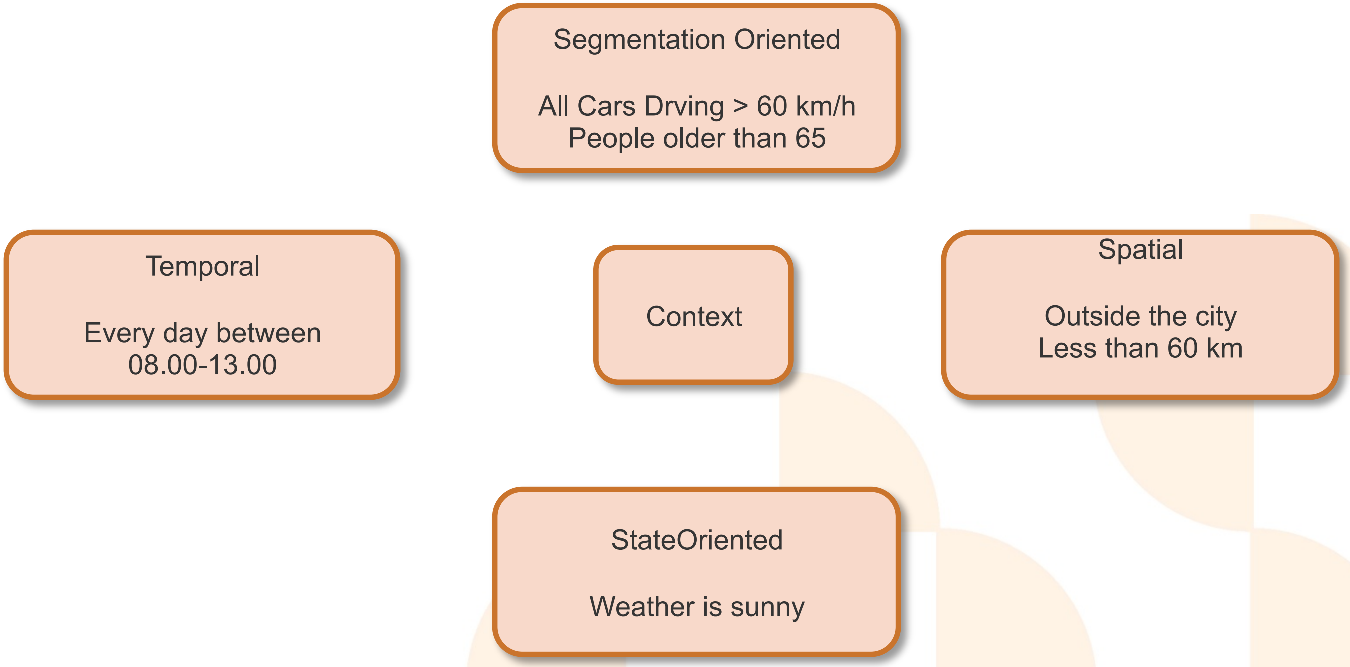
Flerpassageregeln innebär att en bil som passerar flera betalstationer inom 60 minuter bara beskattas en gång. Det belopp som då ska betalas är det högsta beloppet av de passagerna.

Dela:

Sidan ändrad: 24 oktober 2012, 12:13

Context of 60 minutes

Context Partition



Segmentation Oriented

All Cars Drving > 60 km/h
People older than 65

Temporal

Every day between
08.00-13.00

Context

Spatial

Outside the city
Less than 60 km

StateOriented

Weather is sunny

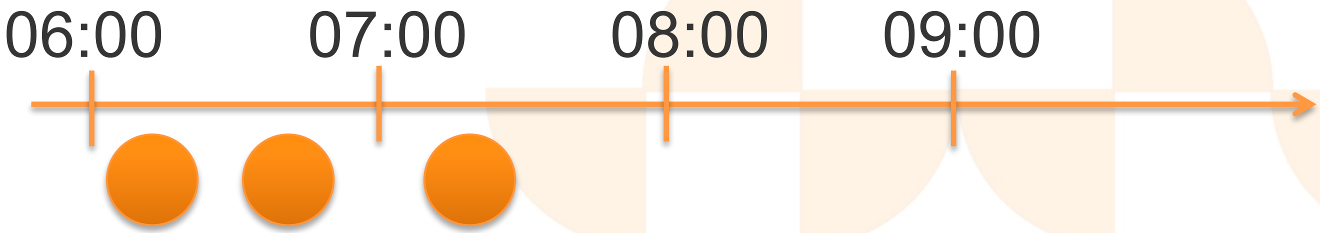
Non-Overlapping vs Overlapping Context

Non-Overlapping (start/end) – 2 context instances

```
create context TollEventHourCtx start TollEvent end after 1 hour;
```

Overlapping (initiated/terminated) – 3 context instances

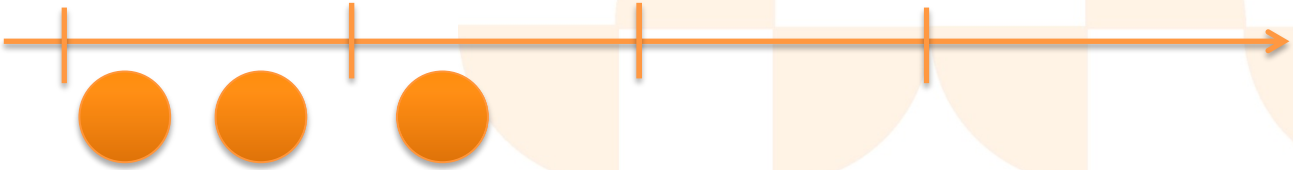
```
create TollEventHourCtx initiated TollEvent terminated after 1 hour;
```



Non-Overlapping combined context

Non-Overlapping (start/end) – 2 context instances

```
create context BillingHourRegNumber
  context RegNumberCtx partition by registrationNumber from TollEvent,
  context TollEventHourCtx start TollEvent end after 1 hour;
```



Calculate at the end of the context

```
@Name("BillingHourRegNumberStmt")
@Audit
context BillingHourRegNumber

select date, time,  registrationNumber,
max(BillingService.getBillingAmount(time)) as
totalBilling
from TollEvent

output when terminated;
```

Demo - VehicleBillingHour

```
Scenario: Total Billing notification on work days. Passing a toll road within 60 min
should be billed once

Given I am driving my car with PYZ123
When I pass toll road at 2012-09-12 and 05:50
And I pass toll road at 2012-09-12 and 07:40
And I pass toll road at 2012-09-12 and 08:30
And I pass toll road at 2012-09-12 and 16:44
And the clock is 2012-09-12 and 23:44
Then I should be notified with a total billing statement of 36 SEK for 2012-09-12
```

Variables

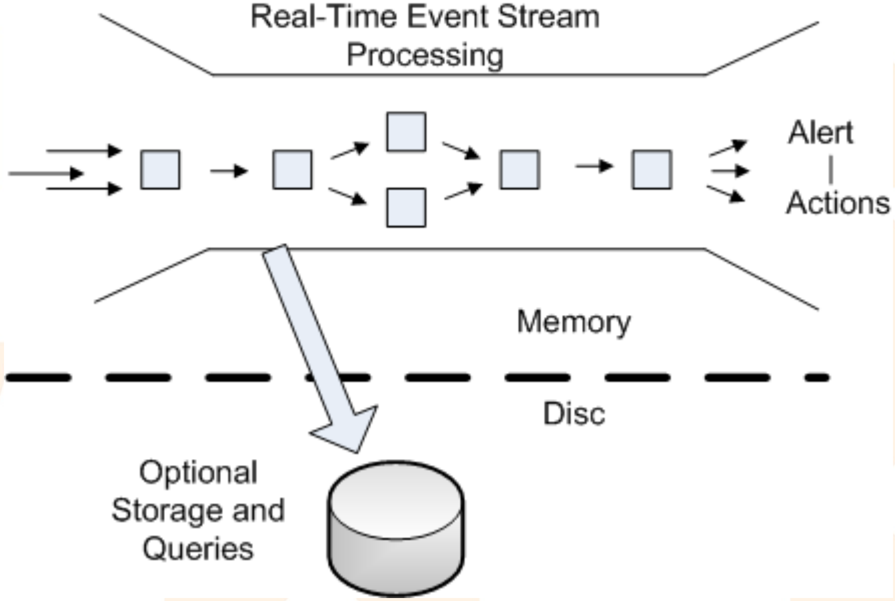
```
create variable int maxPerDay = 60


on TollEvent te
merge VehicleBillingWindow vbw
where te.registrationNumber = vbw.registrationNumber and te.date = vbw.date
  when matched and totalBilling < maxPerDay then
    update set totalBilling =
         Math.min(totalBilling + BillingService.getBillingAmount(te.time), maxPerDay)
  when not matched then
    insert select date, registrationNumber,
                  BillingService.getBillingAmount(time) as totalBilling
```

```
serviceProvider.getEPRuntime().setVariable(”maxPerDay”, 100);
```

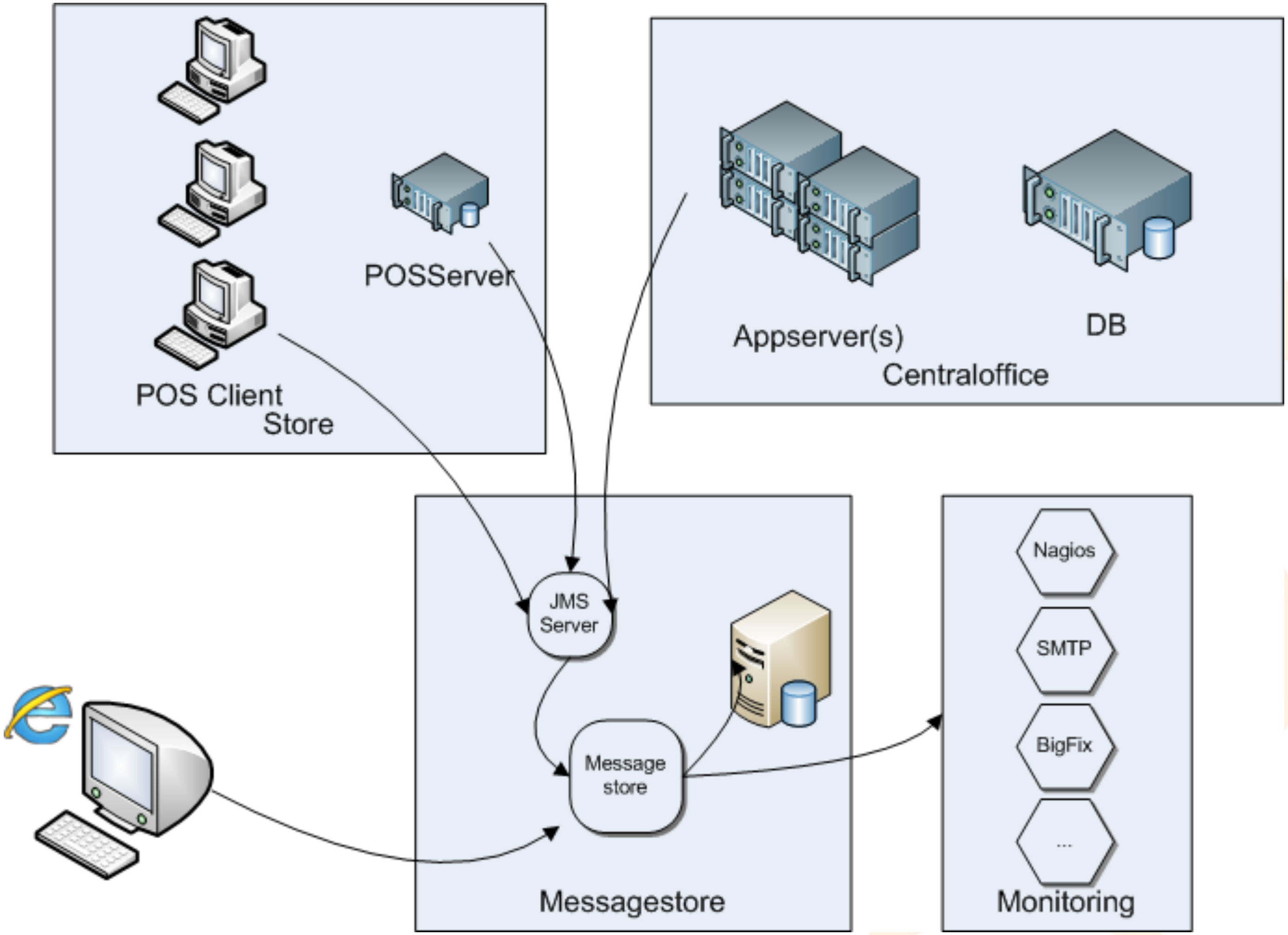
```
create constant variable maxPerDay = 100
```

Esper – Other things

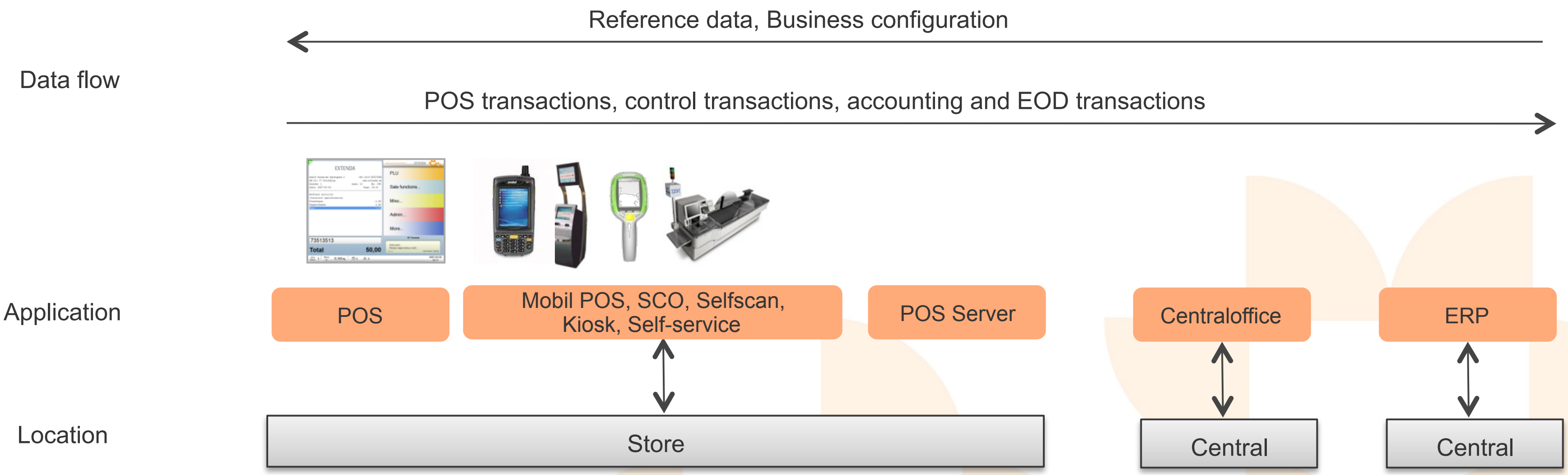
- Integration Adapters (In and/or Out) – File, JMS, AMQP, HTTP, Socket
- RDBMS support
- XML (In/Out) & JSON support (Out)
- Admin API
- HA Commercial version
- GPL license
- …


Real-Time Event Stream Processing
Alert
|
Actions
Memory
Disc
Optional Storage and Queries

Extenda - Esper



Data communication overview



Reference data, Business configuration

Data flow

POS transactions, control transactions, accounting and EOD transactions

Application

POS

Mobil POS, SCO, Selfscan, Kiosk, Self-service

POS Server

Centraloffice

ERP

Location

Store

Central

Central

POSLog – Sale Transaction

```
public class PosLog {

        private final String retailStoreId;
        private final String workStationId;
        private final long sequenceNumber;
        private final String beginTimeStamp;
        private final String endTimeStamp;
        private final String operatorId;
…
```

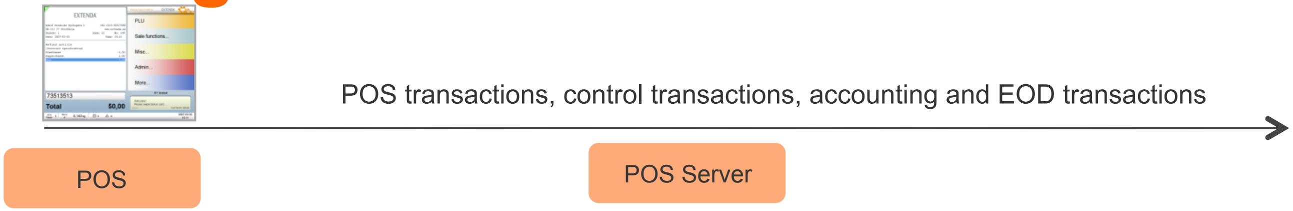
PosLog not stuck in server

```
Scenario: Track PosLog received by POSServer is sent from POSServer

Given PosLog created for store 001 at workstation 001 by operator 1
When PosLog sent to PosServer 1 at 2012-09-12 09:00:00
And PosLog sent from PosServer 1 at 2012-09-12 09:00:59
Then no alert should be sent
```

PosLog stuck in server

```
Scenario: Track PosLog received by POSServer is not sent from POSServer in time

Given PosLog created for store 001 at workstation 001 by operator 1
When PosLog sent to PosServer 1 at 2012-09-12 09:00:00
And PosLog sent from PosServer 1 at 2012-09-12 09:02:00
Then an alert should be sent

Scenario: Track PosLog received by POSServer is not sent from POSServer

Given PosLog created for store 001 at workstation 001 by operator 1
When PosLog sent to PosServer 1 at 2012-09-12 09:00:00
And the time is 2012-09-12 09:01:01
Then an alert should be sent
```

PosLog stuck in server



POS transactions, control transactions, accounting and EOD transactions

POS

POS Server

```
@Audit @Name("MissingPosLogOut") select rstream *
from PosLogServerIn.win:time(1 min) posLogServerIn
  full outer join PosLogServerOut.win:time(1 min) posLogServerOut on
        posLogServerOut.retailStoreId = posLogServerIn.retailStoreId and
        posLogServerOut.workStationId = posLogServerIn.workStationId and
        posLogServerOut.sequenceNumber = posLogServerIn.sequenceNumber
where posLogServerOut.retailStoreId is null
```

PosLog Missing

```
public void update(EventBean[] newEvents, EventBean[] oldEvents) {

  if (oldEvents == null) {
    // Don't care if PosLogServerIn or PosLogServerOut occured
    return;
  }
  // One min has passed after PosLogServerIn happened for the store
  System.out.println("Missing PosLogServerOut" +
       newEvents[0].get ("retailStoreId") + " : " +
       newEvents[0].get("workStationId") +
       newEvents[0].get("sequenceNumber"));
}
```

Aggregation - PosLog generated per store

```
select retailStoreId, count(*) as count
from PosLog.win:time(5 minutes)
group by retailStoreId
output all every 1 minutes
```

Fraud – Reuse of login

//Multiple Login at different store during the same day

```
select p1, p2
from pattern [every p1=PosLog -> p2=PosLog(operatorId=p1.operatorId and
retailStoreId <> p1.retailStoreId)
where timer:within(4 hours)]
```

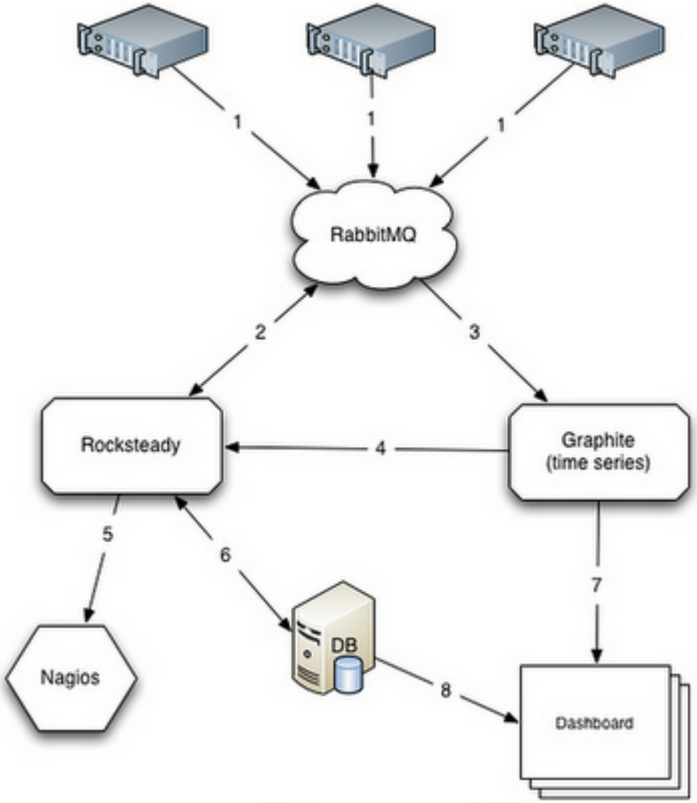
//Multiple Login at the same store

```
select p1, p2
from pattern [every p1=PosLog -> p2=PosLog(operatorId=p1.operatorId and
retailStoreId = p1.retailStoreId and workStationId <> p1.workStationId)
where timer:within(5 minutes)]
```

Esper - Experience

- Rather simple model to understand, yet powerful
- Well documented
- Solution Patterns and mailing lists
- Testability

Rocksteady – Open Source



References

- Event Processing - Books
  - ***Event Processing in Action*** - http://www.manning.com/etzion/
  - ***The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems,*** Luckham, David. 2002
  - ***Event Processing: Designing IT Systems for Agile Companies, 1st ed,*** Chandy, K. M., and W. R. Schulte. 2009.
- Event Processing – Sites
  - http://www.ep-ts.com
  - http://www.complexevents.com
- Esper - http://esper.codehaus.org
- Storm - http://storm-project.net/
  - https://github.com/nathanmarz/storm
  - http://vimeo.com/40972420
- Disruptor - http://code.google.com/p/disruptor/
  - http://martinfowler.com/articles/lmax.html
- Drools - http://www.jboss.org/drools/

Questions!

EXTENDA

Thank you for listening!

peter.norrhall@extenda.se

@peternorrhall