



# StreamSQL on Spark: Manipulating Streams by “SQL” using Spark

Grace Huang, Jerry Shao  
Intel/SSG/Big Data Technology  
Contact us: [jie.huang@intel.com](mailto:jie.huang@intel.com)

# Agenda

- About us
- Problem statement
- What is StreamSQL & its benefit
- Inside POC & HelloWorld
- Next step

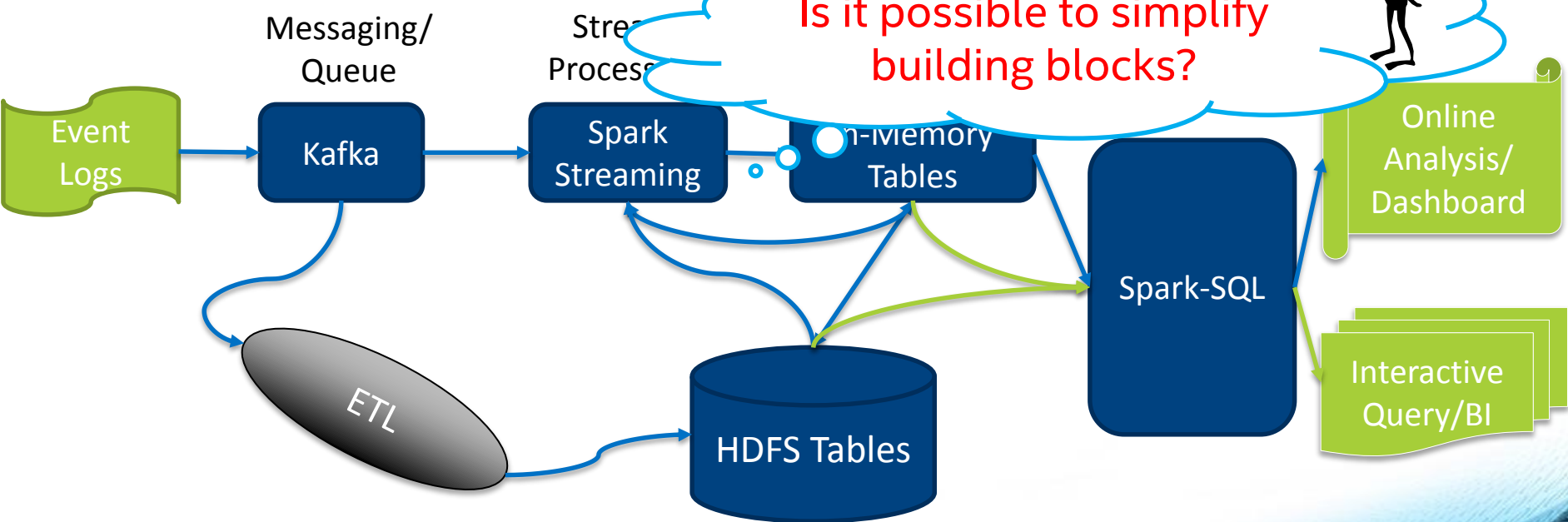
# About US – Big Data Technology in Intel

- Long history with Spark & plays key contributions to grow itself and its ecosystem
- Usability & Operability
  - Metrics & monitors in Spark-streaming
  - CLI supports in Spark-sql
  - More advanced functions in Spark-sql
  - Better Tachyon integration
- Performance
  - New pluggable shuffle (with Cloudera) & Pluggable storage
  - Improve scheduler for better locality
  - Join & Aggregation optimization in Spark-sql

# Spark empowers RTAP paradigm in big data



Is it possible to simplify building blocks?

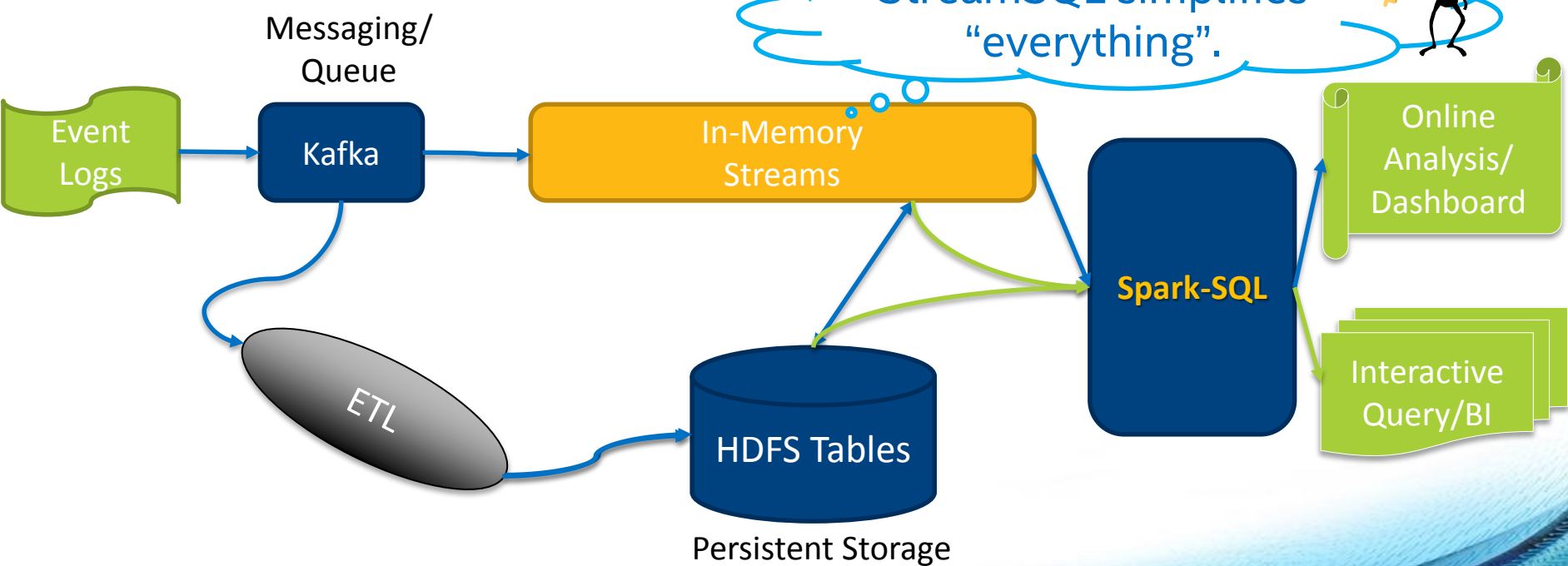


<https://github.com/thunderain-project/thunderain>

Software and Services

# Build DSMS using Spark

StreamSQL simplifies "everything".



# What is StreamSQL?

# Streams $\approx$ Tables means to ...

- To define structures in the stream-in records

```
CREATE TABLE words ( word STRING) STORED AS LOCATION 'hdfs://...';
```

- To manipulate stream data like a table

```
SELECT word, COUNT(*) FROM words;
```

- To drop stream similarly

```
DROP TABLE IF EXISTS words;
```

# Streams $\approx$ Tables means to ...

- To define structures in the stream-in records

```
CREATE STREAM words ( word STRING) STORED AS LOCATION 'kafka://...'
```

- To manipulate stream data like a table

```
SELECT word, COUNT(*) FROM words;
```

- To drop stream similarly

```
DROP STREAM IF EXISTS words;
```



# What's StreamSQL?

- DSMS  $\approx$  DBMS
- To manipulate stream-in data like static structured data in database
- To conduct query **continuously**

# To grasp streaming processing like SQL

## ▪ Spark-streaming

```
val weblog= KafkaUtils.createStream(...)
    .map(_._2)
    .flatMap(_.split("|")).map(_._2, _)

val category=
    sc.textFile(...).flatMap(_.split("|"))

val result
    = weblog.transform(r => r.join(category))
    .map(_._2._2, 1L))
    .reduceByKey(_ + _)

result.print()
```

## ▪ StreamSQL

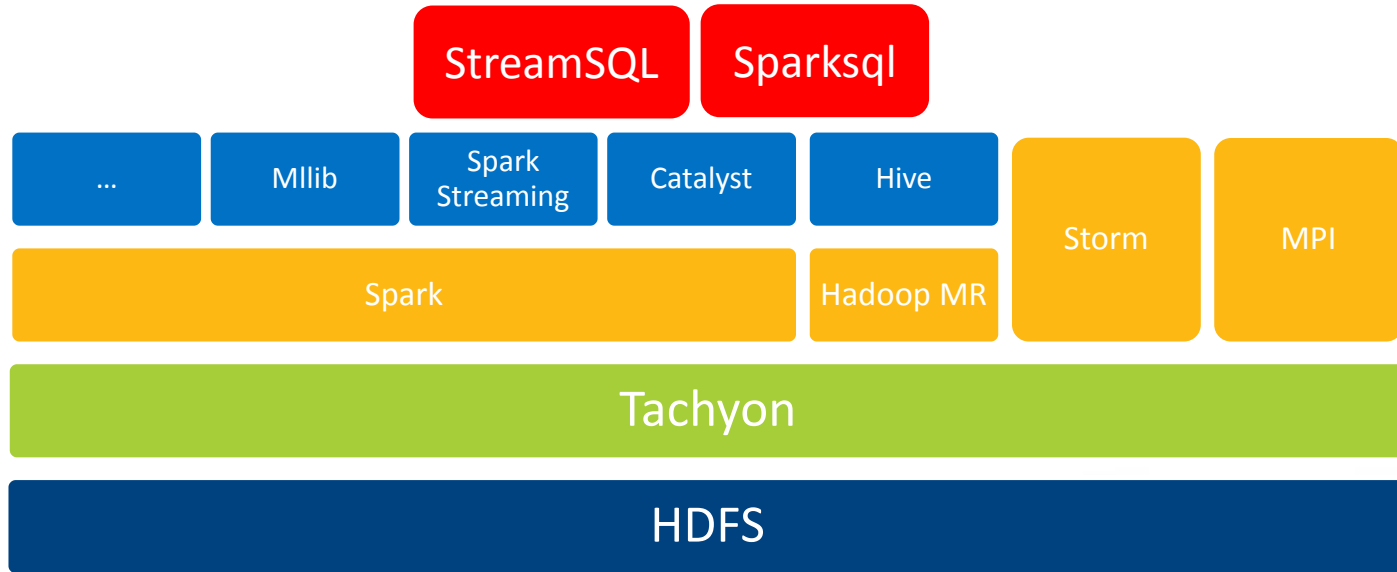
```
CREATE STREAM weblog ( sourceip STRING,
    cookie_id INT, item_id INT ...) STORED AS LOCATION
    'kafka://...'
```

```
CREATE TABLE category (item_id INT, category
    STRING);
```

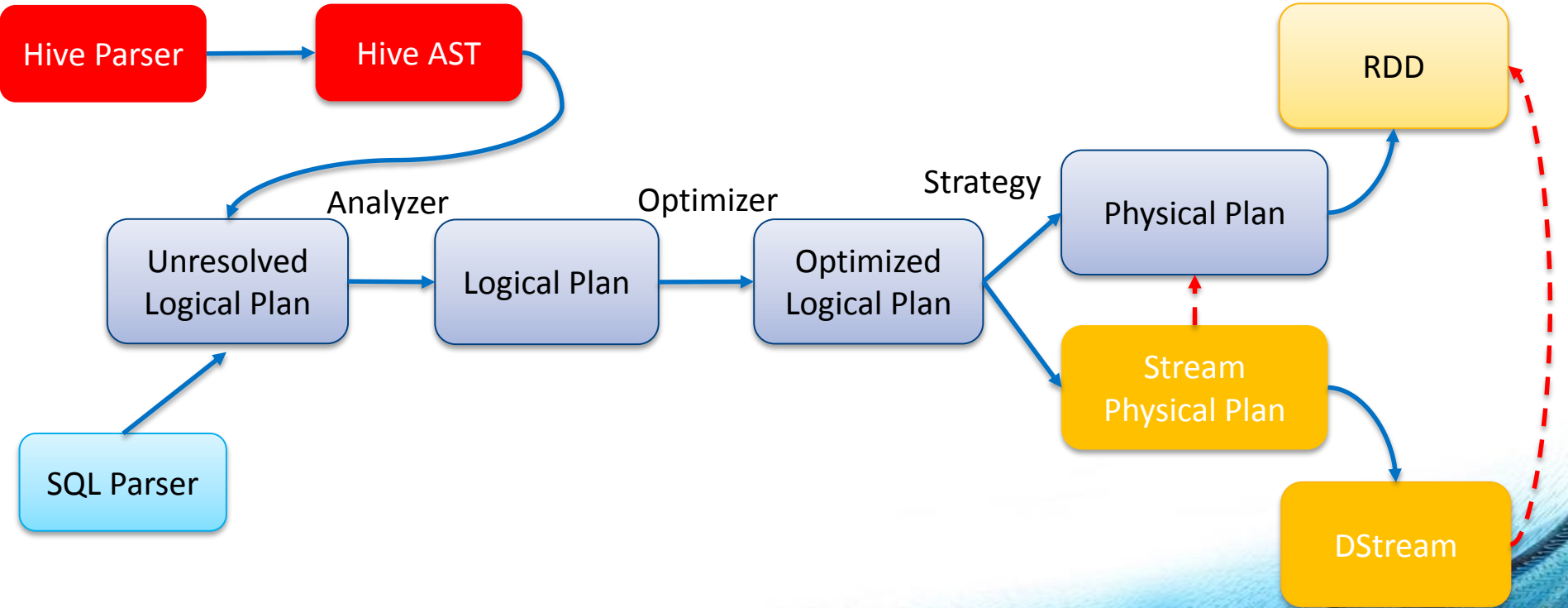
```
SELECT category, COUNT(*) FROM weblog JOIN
    category ON category.item_id = weblog.item_id
GROUP BY category.category;
```

# To build StreamSQL using Spark

- The building block on top of Spark Streaming and Catalyst



# StreamSQL (POC) Architecture

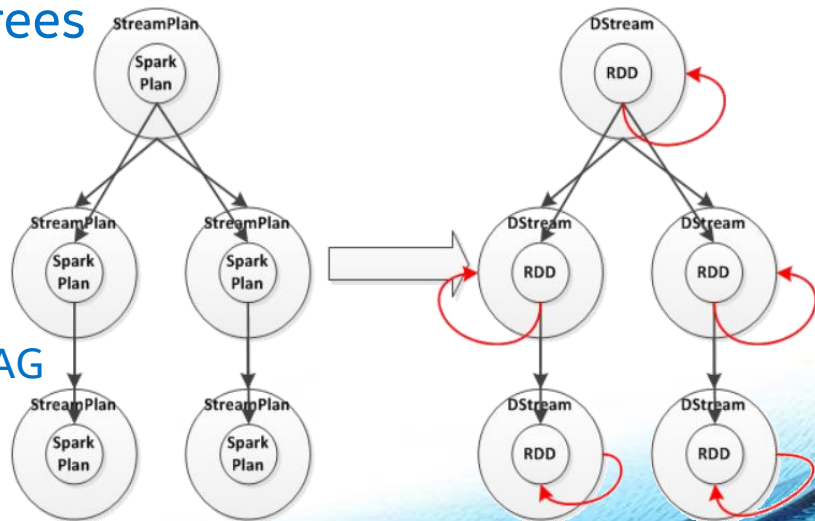


# DDL design (POC)

- Modified Hive parser to support newly introduced Stream syntax
- Shared Hive meta-store for easy integration
- Highly customized Stream Input/Output Format
  - To extract records from streams (equivalent to MR InputFormat)
  - To stream-out with specified write type
- Re-use Ser/Der to understand data structure

# DML design (POC)

- Re-use original LogicalPlan and its analysis rule
- To build two coupled PhysicalPlan trees
  - Re-created StreamPlan
  - RDD-based core SparkPlan
- Upgraded execution workflow
  - Evolve the physical plan into DStream DAG
  - Turn out to be RDD DAG essentially



# StreamSQL's HelloWorld

# Feature#1: HQL compatible

- Query snippet for Wordcount demo case

```
streamHiveContext.streamHql(  
    ""  
    CREATE STREAM IF NOT EXISTS words( word STRING)  
    ROW FORMAT DELIMITED FIELDS TERMINATED BY ' '  
    STORED AS  
    INPUTFORMAT 'org.apache.hadoop.mapred.TextKafkaInputFormat'  
    OUTPUTFORMAT 'org.apache.hadoop.mapred.DummyStreamOutputFormat'  
    LOCATION 'kafka://localhost:2181/topic=test/group=demo'  
    ""  
)  
  
streamHql("""SELECT word, COUNT(*) FROM words GROUPBY word""").print()
```



# Feature#2: Mutual operations between streams & tables (1)

## ▪ Register Table

```
case class Category(item_id: Int, category: String)

val category = sc.textFile("category").map(_.split("|"))
  .map(c => Category(c(0).toInt, c(1)))
sqlContext.createSchemaRDD(category).registerAsTable("category")
```

## ▪ Register Stream

```
case class Weblog (source_ip: String, item_id: Int, cookie: Int)

val weblog: DStream[Weblog] = ssc.socketTextStream(serverIP,serverPort)
  .map(_.split("|"))
  .map(log => Weblog (log(0), log(1).toInt, log(2).toInt))
weblog.registerAsStream("weblog")
```

## Feature#2: Mutual operations between streams & tables (2)

- Query snippet for category visits

```
val query = sql("SELECT category, COUNT(*) FROM webLog JOIN  
category ON category.item_id = webLog.item_id GROUP BY  
category.category;")
```

## Feature#3: LINQ-like Scala DSL

- Query snippet for visits in certain categories

```
val query = sql("SELECT category, COUNT(*) FROM weblog JOIN  
category ON category.item_id = weblog.item_id GROUP BY  
category.category;")
```

```
val visits = query.where('category === worldcup'  
  || 'category === sports'  
  || 'category === family').toDstream
```

# Feature#4: Spark stack's integration

- Query snippet for filtering visits in certain categories

```
val query = sql("SELECT category, COUNT(*) FROM weblog JOIN  
category ON category.item_id = weblog.item_id GROUP BY  
category.category;")
```

```
val visits = query.where('category === worldcup'  
  || 'category === sports'  
  || 'category === family').toDStream
```

```
visits.filter(_._2 > 1000).map( c => "Category: " + c._1 + "  
Visit Counts: " + c._2).print()
```

# Future

- Time-based window function
- More generic physical plan design (rule-based)
- Enrich DDL operations
  - E.g., CREATE STREAM AS TABLE or STREAM
- Support more streaming source (Flume)
- CLI for Catalyst and StreamSQL
- ...

# Summary

- To mine and explore data in a real-time, streaming manner
- StreamSQL to manipulate *STREAM* by SQL just like *TABLE*
- Key supportive features in POC using Spark
  - HQL compatible & shared Hive metastore
  - Mutual operations between streams & tables
  - LINQ-like Scala DSL
  - Spark stack's integration
- StreamSQL stands on the shoulders of Spark and serves Spark with easy usage

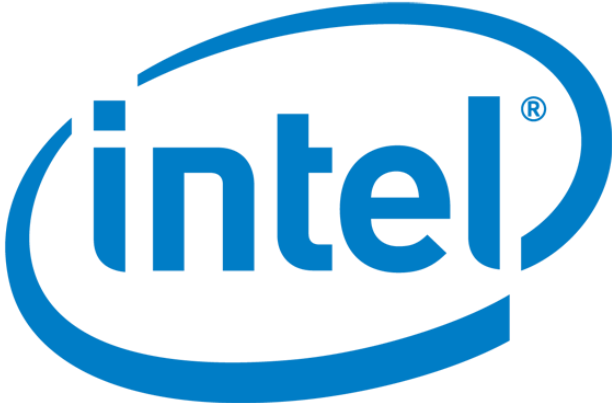
**Let's make both Better altogether!**

# Notices and Disclaimers

- INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.  
A "Mission Critical Application" is any application in which failure of the Intel Product could result, directly or indirectly, in personal injury or death. SHOULD YOU PURCHASE OR USE INTEL'S PRODUCTS FOR ANY SUCH MISSION CRITICAL APPLICATION, YOU SHALL INDEMNIFY AND HOLD INTEL AND ITS SUBSIDIARIES, SUBCONTRACTORS AND AFFILIATES, AND THE DIRECTORS, OFFICERS, AND EMPLOYEES OF EACH, HARMLESS AGAINST ALL CLAIMS COSTS, DAMAGES, AND EXPENSES AND REASONABLE ATTORNEYS' FEES ARISING OUT OF, DIRECTLY OR INDIRECTLY, ANY CLAIM OF PRODUCT LIABILITY, PERSONAL INJURY, OR DEATH ARISING IN ANY WAY OUT OF SUCH MISSION CRITICAL APPLICATION, WHETHER OR NOT INTEL OR ITS SUBCONTRACTOR WAS NEGLIGENT IN THE DESIGN, MANUFACTURE, OR WARNING OF THE INTEL PRODUCT OR ANY OF ITS PARTS.
- Intel may make changes to specifications and product descriptions at any time, without notice. Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined". Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The information here is subject to change without notice. Do not finalize a design with this information.
- The products described in this document may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.
- Contact your local Intel sales office or your distributor to obtain the latest specifications and before placing your product order.
- Copies of documents which have an order number and are referenced in this document, or other Intel literature, may be obtained by calling 1-800-548-4725, or go to: <http://www.intel.com/design/literature.htm>
- Intel, the Intel logo, Intel Xeon, and Xeon logos are trademarks of Intel Corporation in the U.S. and/or other countries.

\*Other names and brands may be claimed as the property of others.

Copyright © 2014 Intel Corporation. All rights reserved.





# Why StreamSQL



May I help you?

Any quick-start for streaming program?



Did you learn SQL?



Yes



Don't worry! You can do it NOW



How?



StreamSQL