Salable CEP system

Background

After the first study of “partitioning techniques for CEP”, I understand the basic techniques to partition the CEP, or to scale out the CEP, then, how to fully use those techniques to achieve better performance?

Rely on special hardware

GPU & FPGA

1. In "Parallelization of Complex Event Processing on GPU", the author implement three parallel approaches,
   - A.Thread per event.
   - B.Thread per subscriber.
   - C.Thread per event-subscriber pair.

> The target system is public-subscribe system (not exactly CEP system), but the idea is somehow inspiring.

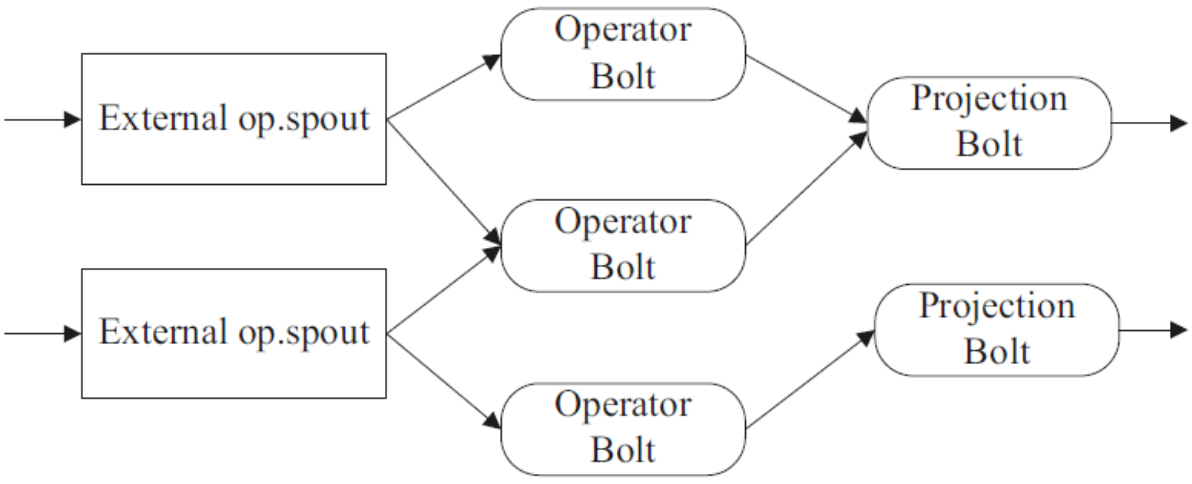
1. In "High performance FPGA and GPU complex pattern matching over spatio-temporal streams", the author proposed two main parallelism level approaches:
   - Inter-pattern parallelism : all pattern queries are evaluated in parallel.
   - Intra-pattern parallelism : individual predicates within a pattern are evaluated in parallel on different SPs.

> the idea in intra-pattern parallelism similar are very similar to "Scalable Complex Event Processing on Top of MapReduce", that the is doing filtering(predicates).

The above paper proposed a series optimization strategies specific to FPGA/GPU as well, but out of my interest.

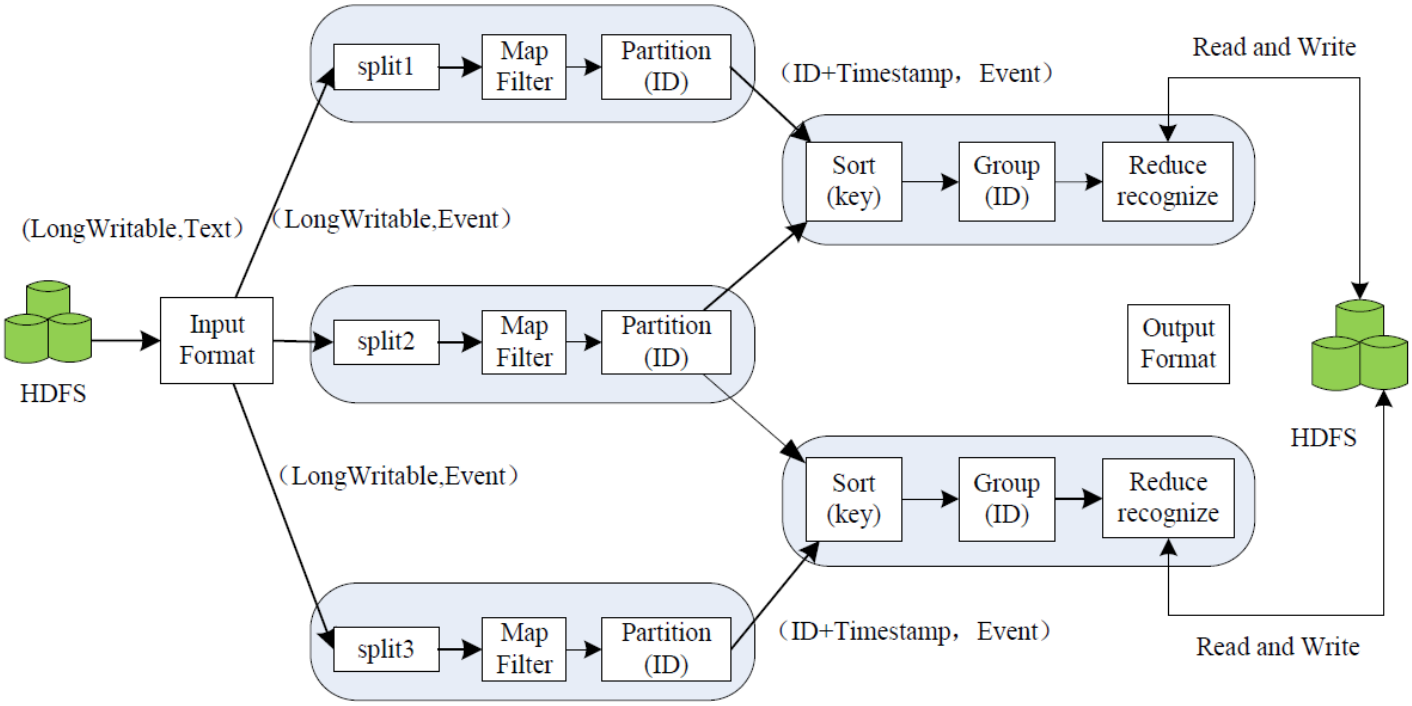
Rely on other framework

1. In ICCSPS'14, "High Efficient Complex Event Processing Based on Storm" proposed their efforts on implementing complex event processing system on storm, in dueling with the issue "supporting complex event detection in multiple sources environments". They hope to go further than distributing queries and achieves better scalability by parallelizing event detection.[But suspected to be based on pattern partitioning.] idea like bellow:



   According to the paper, the author treat spout and bolt as different operators. However, the implementation and experiment detail is unclear.

2. In APWeb'12, "Scalable Complex Event Processing on Top of MapReduce" proposed their efforts of building complex event processing framework on top of MapReduce. They claimed that although many efforts are put to modify the original mapReduce idea to support different applications, none of them support CEP. Their basic idea looks like the figure bellow:



- **Map** task is to *filter* events to reduce the cost of the sort and group phase.
- **Partition** uses a hash function on the attribute ID to ensure that the events with the same ID will be processed in one reduce task.
- **Sort** will sort the data according to the primary key of events.
- **Reduce** is to to match and identify events gradually for each group.

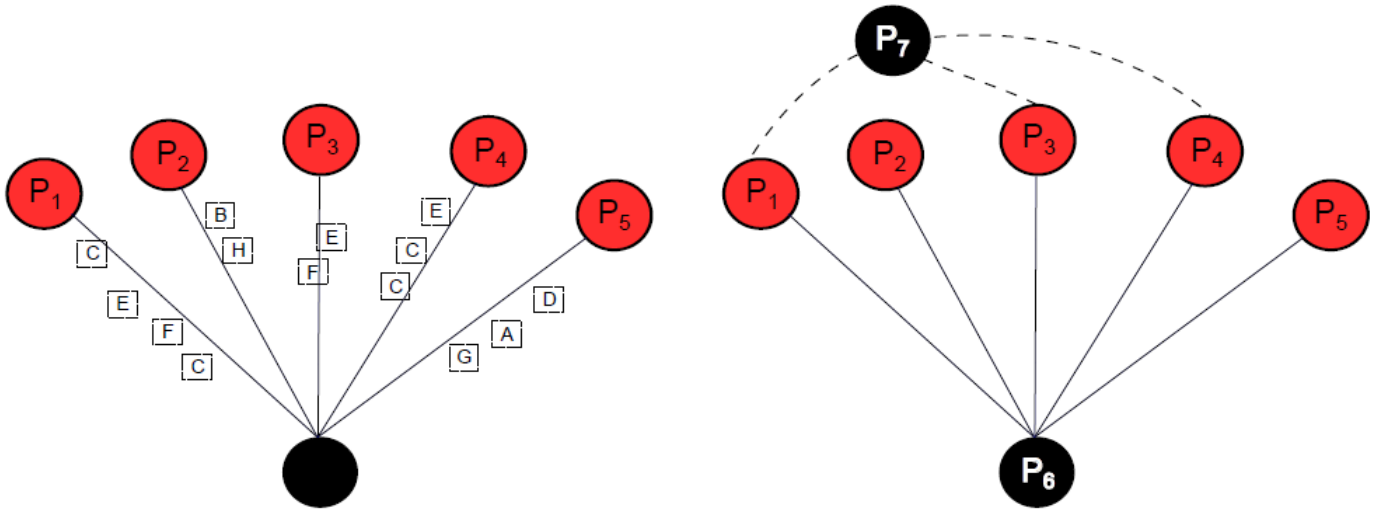
New semantic architecture

> Lots of relevant work in network community

Usually, CEP is developed on client-server style. (We can distribute server of-course, but it's still CS-model). In Intelligent Distributed Computing VII' 14, "Semantically Partitioned Peer to Peer Complex Event Processing",

> **Some nice sentences from author:** " Events maybe correlated together and may thusly cross both technological and domain boundaries. The process of correlating is referred to as **pattern matching**.
>
> Motivation for CEP nicely correlates with the current advent of Big Data, which is a data centric approach to extracting meaningful data. The problem here is not to store such data, but to retrieve it and to extract meaningful information.
>
> In CEP, data that's not processed is simply discarded.

their idea is simply show as bellow:



(a) Centralized CEP (b) Dynamic CEP deployment

they also proposed two heuristic algorithm , the algorithm dynamically add/remove CEP engines among peers to conduct pattern matching.

Replicated engine

> No prior work of CEP on this topic

In EuroSys'11, "Database Engines on Multicores, Why Parallelize When You Can Distribute?", the author shows that:"Deploy several replicated engines within a single multicore machine can achieve better scalability and stability than a single database engine operating on all cores." The author rather than redesigning the engine, they partition the *multicore* machine and allocate an unmodified database engine(PosgreSQL and MYSQL) to each partition. specially, their design belongs to "Single master replication with support for speciallized satellites and partial replication.

Elastic

> Some works are done by Thomas.