Big Challenges Building a Real-World Big Data System
Or, what I did with eight years of my life

Andy Frenkiel
Big Data Talk, RMIT, 16 May 2014
IBM RESEARCH

Isolated Research

Joint Projects

IBM Divisions, Clients, Universities

Radical Collaboration

The World is Our Lab

50s – '80s

Hardware

80s - '00s

+ Software & Services

IP in IBM Products

Proprietary

Leverage IP
for Income

Influence

Cross Licensing
for Freedom of Action

Income

+ Smarter Planet

00s …

2014-03-24

IBM Research - Australia                     GTO 2014
IBM Research Australia is located at Level 5, 204 Lygon Street Carlton, VIC 3053
~85 FTE now, growing to 150 by May 2016
IBM Research – Australia: Mission
“Evidence-based Decision Making at the Speed of Thought”

Natural Resource Management

Disaster Management

Life Sciences* Healthcare

Assimilation

Modeling and predictive analysis

Decision support

Capturing

Command & control

High Performance Computing
Stream Computing, Cloud Computing
Visualisation, Modelling, Analytics
Optimisation, Human Computing Interfaces

*IBM Life Sciences Collaboratory not in scope
We want to be famous for: Cognitive decision making (consequence management) delivered as a service
A few of the big challenges we encountered implementing IBM InfoSphere Streams, a big data product ...
Big Challenge #1

How do you write code to process streaming data?
Big Challenge #1

How do you write code to process streaming data?
Big Challenge #1, continued

How do you write code to process streaming data?

Operators share a common structure

- *italics* are sections to fill in

Reading an operator invocation

- Declare a stream *stream-name*
- With attributes from *stream-type*
- that is produced by *MyOperator*
- from the input(s) *input-stream*
- *MyOperator* behavior defined by *logic*, *parameters*, *windowspec*, and *configuration*; output *attribute assignments* are specified in *output*

For the example:

- Declare the stream *Sale* with the attribute *item*, which is a raw (ASCII) string
- Join the *Bid* and *Ask* streams with
- *sliding* windows of 30 seconds on *Bid*, and 50 tuples of *Ask*
- When *items* are equal, and *Bid* price is greater than or equal to *Ask* price
- Output the *item* value on the *Sale* stream

---

Anatomy of an Operator Invocation

```
stream<stream-type> stream-name
  = MyOperator(input-stream; ...)
{
  logic  logic ;
  window windowspec ;
  param  parameters ;
  output output ;
  config configuration ;
}
```

Example

```
stream<rstring item> Sale = Join(Bid; Ask)
{
  window Bid: sliding, time(30);
  Ask: sliding, count(50);
  param  match : Bid.item == Ask.item
         && Bid.price >= Ask.price;
  output Sale: item = Bid.item;
}
```
Interesting Challenge #1, continued

How do you write code to process streaming data?

Streams Application

Source Operator

Sink Operator
Big Challenge #2

How do you implement an application consisting of 1000s of operators and data streams?

- 4 wells, 10s of operators and streams
- 20 wells, 100s of operators and streams
- 100 wells, 1000s of operators and streams
Big Challenge #2, continued

How do you implement an application consisting of 1000s of operators and data streams?

... Use hierarchy
Big Challenge #2, continued

How do you implement an application consisting of 1000s of operators and data streams?

… Use code to build code

```perl
/**
 * For each well, generate SPL to compute the predicted alarm.
 * metrics, and output results to file.
 *
 * @param $i (int) - well number.
 * @param $well (string) - well name.
 *
 * @return void
 */
for ($i=0; $i < $num_wells; $i++) {
  //
  stream<PumpPredictionTypeNumeric> PumpDataScored_<$i> = WellHeadPumpPredictor(PumpData_<$i>); 
  param model window size : getSubmissionTimeValue("model window size");
  config placement : partitionColocation("Well_<$i>");
} as CountMetrics_<$i> = CountMetrics(PumpDataScored_<$i>); 
  config placement : partitionColocation("CountMetrics");
} as OutputScores_<$i> = WellHeadPumpSink(PumpDataScored_<$i>); 
  param output_file : concat("output/",concat("pump_output_file,_<$i>.csv"));
  config placement : partitionColocation("Well_<$i>");
} as PumpDataScored_3 = WellHeadPumpPredictor; 
  config placement : partitionColocation("WellHeadPumpPredictor");
} as CountMetrics_2 = CountMetrics; 
  config placement : partitionColocation("CountMetrics");
} as OutputScores_2 = WellHeadPumpSink; 
  config placement : partitionColocation("WellHeadPumpSink");
```
Big Challenge #2, continued

How do you implement an application consisting of 1000s of operators and data streams?

... Automatically generate operators for sub graphs that can be parallelised

```c++
composite Main {
    stream<Type> Src = Source() {}

    @parallel(width=2)
    stream<Type> Res = AB(Src) {}

    () as Snk = Sink(Out) {}  
}

composite AB(input In; output B) {
    graph
    stream<Type> A = Functor(In) {}
    stream<Type> B = Functor(A) {}
}
```

![Logical and Physical diagrams](image-url)
Big Challenge #3
What should run where?

... Streams Runtime Supports Placement Criteria

Host pools can force operators to be on hosts with SolidDB installed

Operator placement constraints allow for co-location, ex-location, and isolation of operators

SolidDB could be wrapped as a custom operator for dynamic deployment and relocation
Big Challenge #3, continued
What should run where?

... Streams runtime optimises at deploy time

Optimising scheduler assigns PEs to hosts, and continually manages resource allocation

Dynamically add hosts and jobs

New jobs work with existing jobs
Big Challenge #4
How do you debug the streaming data flow?

... Use a Streams Debugger

```bash
[streamadmin@streams bin]$ ./standalone
IBM Stream Debugger (SDB), pid: 3982
Standalone application execution is suspended.
Set initial probe points, then run "g" command to continue execution.
(sdb) o
#in  #out Operator                                      Class
1    1    QuoteFilter                                   QuoteFilter
1    0    SinkOp                                        SinkOp
1    1    TradeFilter                                   TradeFilter
0    1    TradeQuote                                    TradeQuote
1    1    PreVwap                                       PreVwap
2    1    BargainIndex                                  BargainIndex
1    1    Vwap                                          Vwap
(sdb) b TradeFilter TradeQuote
Set +  0   Breakpoint  TradeFilter o   0     stopped:false
(sdb) b PreVwap o 0
Set +  1   Breakpoint  PreVwap o   0     stopped:false
(sdb) g
+  0   Breakpoint  TradeFilter o   0     dropped:false stopped:true
price, 83.48, decimal64
volume, 74200, decimal64
ts, (1135654207,521000000,0), timestamp
ticker, "IBM", rstring
(sdb) c
(sdb)
+  0   Breakpoint  TradeFilter                                   o   0     dropped:false stopped:true
price, 83.45, decimal64
volume, 200, decimal64
ts, (1135654217,627000000,0), timestamp
ticker, "IBM", rstring
```

Set Breakpoints

```bash
(sdb) b Tra
TradeFilter TradeQuote
(sdb) b TradeFilter o 0
Set +  0   Breakpoint  TradeFilter o   0     stopped:false
(sdb) b PreVwap o 0
Set +  1   Breakpoint  PreVwap o   0     stopped:false
(sdb) g
+  1   Breakpoint  PreVwap o   0     dropped:false stopped:true
price, 83.48, decimal64
volume, 74200, decimal64
ts, (1135654207,521000000,0), timestamp
ticker, "IBM", rstring
```

Use a Streams Debugger

© 2013 IBM Corporation
Other Big Challenges

How do we support user-defined type-generic Streams operators and functions?

How can the runtime adapt dynamically to changes in workload?

How do we extend running applications?

How do we ensure that no tuples are dropped?

How do we make developing Streams programs easy?

How do we document Streams code, and enable understanding and sharing?

How do we enable Streams to readily integrate with other systems?
Challenge #5
How can I learn more?

http://www-01.ibm.com/software/data/infosphere/streams/quick-start/
Install and go downloads for Streams

Install and go downloads for BigInsights

http://www-01.ibm.com/support/knowledgecenter/SSCRJU_3.2.1
Product documentation

Forums, help, more downloads

https://github.com/IBMStreams
Useful Streams toolkits

http://bigdatauniversity.com/
http://www.ibmbigdatahub.com/
More reference information, videos, blogs, tools ...