

Darach Ennis (darach.ennis@streambase.com)

StreamBase Systems

Erlang UG London - April 20th 2011

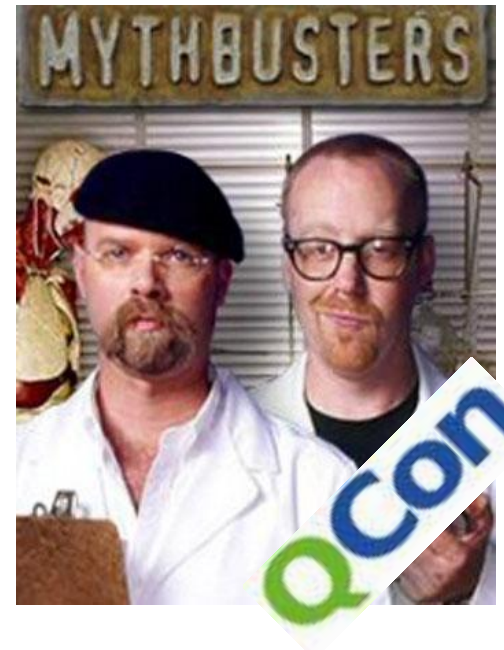
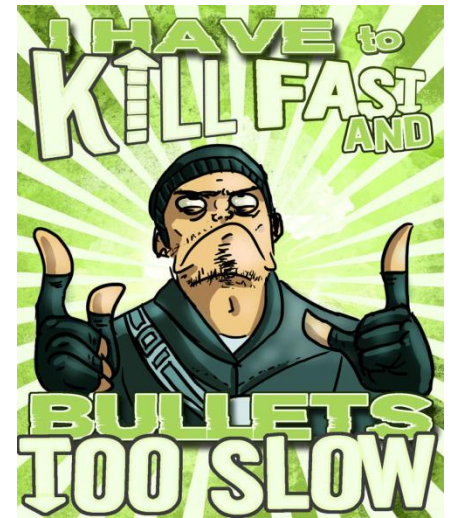
Complex ~~Event~~ **Erjang** Processing:
DSL for Low Latency High Frequency Computing



High Level DSLs : Myth Vs Reality

Myth: High level domain specific languages are too slow for HFT.

Reality: High level domain specific languages can deliver better performance than system programming languages when tailored to a specific task.



Complex Event Processing aka Event Processing

- **Software organized by events (compare object oriented)**
 - What's an event? What's an object?
 - And event is something can trigger processing, can include data.
 - Naturally but not usually represents a “real world” event or observation.
- **Complex Event Processing Platforms**
 - Software stack for event based systems, event driven architectures
 - Event Programming Language – SQL-based, Rules-based, or State-based
 - Commercial and open source: StreamBase, Progress, Microsoft, IBM, Oracle, SAP, Esper, Drools and many more
- **Adopted in financial services and other markets**
 - System monitoring, industrial process, logistics, defense/intelligence
- **Other Event Processing Approaches:**
 - Erlang, Actors, node.js, .NET Rx



Why a DSL?

- High level
- Graphical
- Appropriate for purpose
- Understandable
- Flexible

“Simplicity is always disruptive”

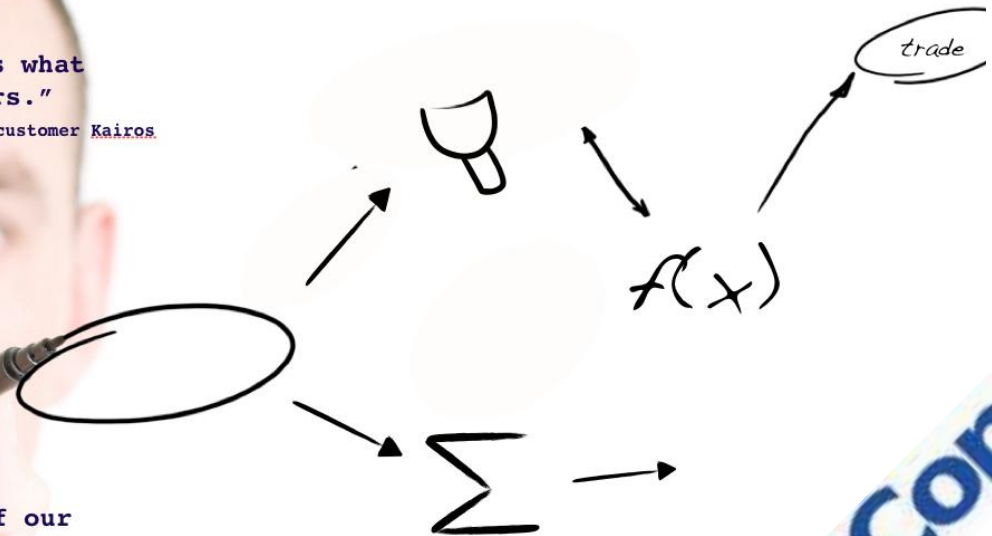
- Clayton Christensen

“We developed in 4 months what would have taken 4 years.”

- StreamBase customer Kairos

“We modify the behavior of our trading system every day.”

- StreamBase customer PhaseCapital



What does a CEP DSL or Language offer?

- **Continuously Observe, Orient, Decide Act (OODA) on event streams**
 - Continuous Incremental Query
 - Pattern matching within or across streams
 - Branch – Split, Causal Split, Filter.
 - Combine - Semi-Join, Union, Gather, Merge, Join, Pattern
 - Windows – Process sets of streaming data
 - Sliding or Tumbling, Overlapping or Non-Overlapping, Gaps or No Gaps
 - Finite (1 second, 1000 tuples), Infinite
 - Emission Policies: On Close, Every odd message
 - Predicate based – Roll your own window type
 - State Management – In memory, CSV files, CSV sockets, RDBMS, Parallel DBMSs, column stores, KV stores...
 - Nice to have:
 - Declarative concurrency, Interface Polymorphism, Distribution, Extensible



Challenges for CEP

■ Ultra Low Latency

- Sub-millisecond is standard, sub-100-micro is desirable.

■ Large Data Volumes

- Hundreds of thousands of events, thousands of decisions, per thread.
- Big Data. Hundreds+ of SMP nodes each ...

■ Demanding Operational Environment

- 24x7, 365 – in critical environments (trading, surveillance, utilities)

■ Sophisticated Data Processing (sometimes)

- Options pricing, yield curves, risk metrics, smart grid capacity planning, fraud detection.

■ How it's done (QCON London 2011):

- How LMAX did it? <http://bit.ly/fUeSOP>
- How we did it? <http://bit.ly/hM6NAP> <- Our CTO Richard Tibbetts talk



StreamBase Event Processing Platform

Developer Studio

Graphical StreamSQL for developing, back testing and deploying applications.

StreamBase Frameworks

StreamBase Component Exchange

Studio Integrated Development Environment



Applications

Visualization



Input Adapter(s)

Inject streaming (market data) and static (reference data) sources.

Adapters

StreamBase Server

Adapters

Event Processing Server

High performance optimized engine can process events at market data speeds.

Output Adapter(s)

Send results to systems, users, user screens and databases.

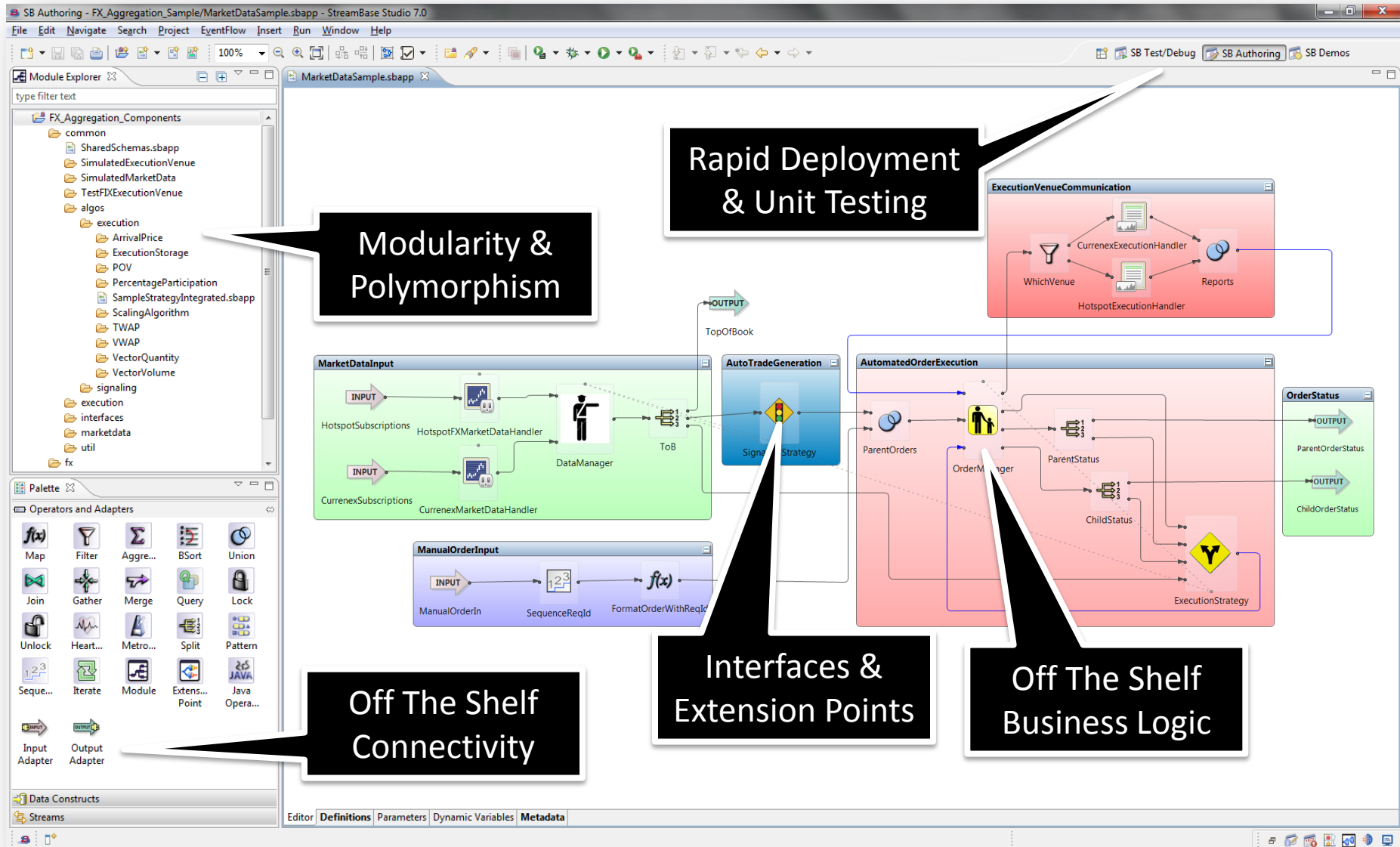


How did we do it?

- **Compilation and Static Analysis**
 - Design the language for it
- **Modular abstraction, interfaces**
 - Quants and Developers Collaborate
- **Bytecode generation and the Janino compiler**
 - Optimized bytecodes, in-memory generation
- **Garbage optimization**
 - Pooling, data class, invasive collections
- **Integrations, C++ and Java plugins**
 - Efficient native interfaces
- **Adapter API, FIX Messaging**
 - Threading and API structure for ultra low latency
- **Parallelism, Clustering, Lanes and Tiers**
 - Scalability with latency in mind
- **Named Data Formats, Schemas**
 - Sharing data and semantics between apps



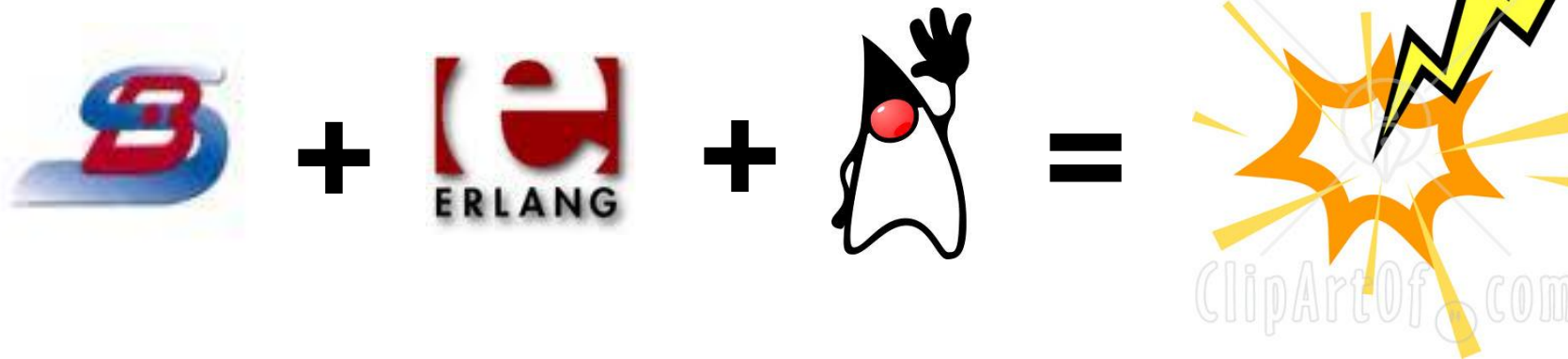
StreamBase StreamSQL EventFlow



Agenda

- Windows in the Wild
- Embedding Erlang via Erjang
- Define Erlang Behaviours to define a cross-domain Contract
- Implement & Test Behaviours in plain ole' Erlang
- Expose Behaviours to StreamBase
- Deploy & run Erlang extensions in StreamBase algorithms
- Shameless Plug, Acknowledgements, Q&A

*In a good way, like
'a pint of plain'
!!! 😊*



Windows in the Wild - #1 Riak

- **Riak Core:** https://github.com/basho/riak_core
 - Mixes the 'when' window dimension (time) with the what 'aggregation'

```
%% Sample snarfed from:
%%   https://github.com/basho/riak_core/blob/master/src/slide.erl

%% Create a new slide with an hourly window
T0 = slide:fresh(60*60),

%% Update every time an interesting event passes
T1 = slide:update(T0, Weight, slide:moment())

%% Eventually, emit interesting results
{NumberOfCars, TotalWeight} = slide:sum(TN, slide:moment()),
{NumberOfCars, AverageWeight} = slide:mean(TN, slide:moment())
{NumberOfCars, {MedianWeight,
  NinetyFivePercentWeight,
  NinetyNinePercentWeight,
  HeaviestWeight}} = slide:nines(TN, slide:moment())
```



Windows in the Wild - #2 tlack on BitBucket

■ Thomas Lackner - tlack - EMA:

- <https://bitbucket.org/tlack/erlang-exponential-moving-average/overview>
- Mixes the 'when' window dimension (time) with the what 'aggregation' and number of occurrences 'how many'

```
%% Sample snarfed from:
%%   https://bitbucket.org/tlack/erlang-exponential-moving-average/src/6ba1f3018836/ema.erl

%% start an instance tracking 1 sec, 10 sec, and 60 sec exponential moving avg
S = ema:start([1, 10, 60]).
ema:add(S, 5).
ema:add(S, 10).
ema:add(S, 8).

%% wait a few moments and then get current moving avg..
ema:ema(S).
```



Embedding Erlang (Erjang)

```
// Create an embedded Erjang 'Session'
// Based on: https://github.com/trifork/erjang/blob/master/src/main/java/erjang/sample/RPCSample.java
//
public static class ErjangSession extends Thread {
    public ErjangSession() {
        start();
        RPC.wait_for_erjang_started(60*1000L);
    }

    public void run() {
        String[] ARGS = {
            "-progrname", "ej",
            "-home", System.getProperty("user.home"),
            "-root", "/home/streambase/otp-R13B04",
            "-noshell",
            "-noinput",
            "-pa", "/home/streambase/wo/erjang/SbErjang/erlang-src",
            "+A", "2",
            "+S", "1",
            "+e", "5.7.5",
            "-s", "rpc", "erjang_started"
        };

        try {
            erjang.Main.main(ARGS);
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
        }
    }
}
```

Prepend the dir defining our behaviors to the code path

Define Behavioral Contracts

```

%% Contract between StreamBase and erlang (erjang) embedded
%% Enables StreamBase aggregate plugin/extension functions to be written in erlang
%%

-module (sb_aggregate_fn).
-export ([behaviour_info/1]).

behaviour_info(callbacks) ->
[
  {init,1},
  {accumulate,2},
  {emit,1}
];

behaviour_info(_Other) ->
  undefined.

```

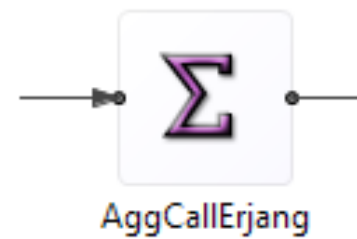
Add	*	lastval(*)
Add	Sum	ErjangAggregateFunction('sb_aggregate_fn_sum',tuple(Value as V),tu...
Add	Avg	ErjangAggregateFunction('sb_aggregate_fn_sma',tuple(Value as V),tu...
Add	Ema	ErjangAggregateFunction('sb_aggregate_fn_ema',tuple(Value as V, 0.5...
Add	Ema2	exp_moving_avg(Value,5,0.5)
Add	Ema26	ErjangAggregateFunction('sb_aggregate_fn_ema2',tuple(Value as V, 0....
Add	Ema12	ErjangAggregateFunction('sb_aggregate_fn_ema2',tuple(Value as V, 0....
Add	SEma26	exp_moving_avg(Value,26,.5)
Add	SEma12	exp_moving_avg(Value,12,.5)

Streams Functions » 1

Input Output

<> output1 (9 fields)

- Value double
- Sum tuple
- Avg tuple
- Ema tuple
- Ema2 double
- Ema26 tuple



```

%% Contract between StreamBase and erlang (erjang) embedded
%% Enables StreamBase simple plugin/extension functions to be written in erlang
%%

-module (sb_simple_fn).
-export ([behaviour_info/1]).

behaviour_info(callbacks) ->
  [{caller1,1}];

behaviour_info(_Other) ->
  undefined.

```

General Output Settings Concurrency

Input Fields All None

Additional Expressions Scroll to: Input Expressions

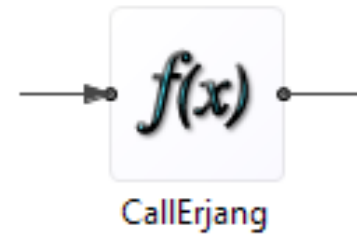
Action	Field Name	Expression
Add	Result	caller('sb_simple_fn_add',input1,tuple(double(null) as Result))

Streams Functions » 1

Input Output

<> input1 (2 fields)

- A double
- B double



Implement & Test in Erlang (or Erjang!)

%% Sample: Exponential Moving Average (EMA)

```
-module (sb_aggregate_fn_ema2).  
-behaviour (sb_aggregate_fn).  
-export ([init/1,accumulate/2,emit/1]).
```

```
init(_State) -> {[], [], 100.0}.
```

```
accumulate (State, {A,Weight,Capacity}) ->  
  {Values,Weights,W} = State,  
  Exp = W * (1.0 - Weight),  
  { bounded_list:append (Values,A,Capacity),  
    bounded_list:append (Weights,W,Capacity),  
    Exp }.
```

```
emit (State) ->
```

```
  {Values,Weights, _W} = State,  
  lists:sum([ V * VW || {V, VW} <- lists:zip (lists:reverse (Values),Weights)]) / lists:sum (Weights) .
```

```
streambase@feck: ~/wo/erjang/SbErjang/erlang-src  
streambase@feck:~/wo/erjang/SbErjang/erlang-src$ erlc -pa . *  
streambase@feck:~/wo/erjang/SbErjang/erlang-src$ erl -pa .  
Erlang R14B02 (erts-5.8.3) [source] [64-bit] [smp:2:2] [rq:2] [async  
[kernel-poll:false]  
  
Eshell V5.8.3 (abort with ^G)  
1> l(sb_simple_fn_add).  
{module,sb_simple_fn_add}  
2> sb_simple_fn_add:caller1({1,2}).  
3  
3> l(sb_aggregate_fn_sma).  
{module,sb_aggregate_fn_sma}  
4> S0 = sb_aggregate_fn_sma:init(0).  
{0,[]}  
5> S1 = sb_aggregate_fn_sma:accumulate(S0,{5}).  
{5,[]}  
6> S1 = sb_aggregate_fn_sma:accumulate(S1,{5}).  
** exception error: no match of right hand side value {10,2}  
7> S2 = sb_aggregate_fn_sma:accumulate(S1,{5}).  
{10,2}  
8> sb_aggregate_fn_sma:emit(S2).  
5.0  
9> S3 = sb_aggregate_fn_sma:accumulate(S1,{100}).  
{100,2}  
10> sb_aggregate_fn_sma:emit(S3).  
52.5  
11> q().
```

Duh!
Typo!



Expose to StreamBase [Call by Behaviour/Mod] #1

```
public class ErjangAggregateFunction extends AggregateWindow {
    private EObject state;
    private String m;
    private Tuple h;

    // Called before a new window opens
    public void init() { }

    // Called when a tuple emission policy fires
    public Tuple calculate() {
        return SimpleEmbedded.callerl_emit(m, state, h);
    }

    @CustomFunctionResolver("accumulateCustomFunctionResolver0")
    public void accumulate(String mod, Tuple args, Tuple hint) {
        if (state == null) {
            state = SimpleEmbedded.callerl_init(mod, new EDouble(0));
            m = mod; h = hint; // Type Hint. Ensure 'free form' erlang tuple conforms with SB tuple's schema
        }
        state = SimpleEmbedded.callerl_accumulate(m, state, args, hint);
    }

    public static CompleteDataType accumulateCustomFunctionResolver0(
        CompleteDataType mod, CompleteDataType args, CompleteDataType hint) {
        return hint; // Keep the StreamBase type checking police happy!
    }

    public void release() {
        state = null;
    }
}
```



Expose To StreamBase [Call by Behaviour/Mod] #2

```
private static final EAtom am_init = EAtom.intern("init");
private static final EAtom am_accumulate = EAtom.intern("accumulate");
private static final EAtom am_emit = EAtom.intern("emit");

public static EObject callerl_init(String mod, EObject state) {
    EAtom m = EAtom.intern(mod);
    ETuple et = (ETuple)RPC.call(m, am_init, state);
    return et.elm(2); // Unwrap Erjang RPC call response
}

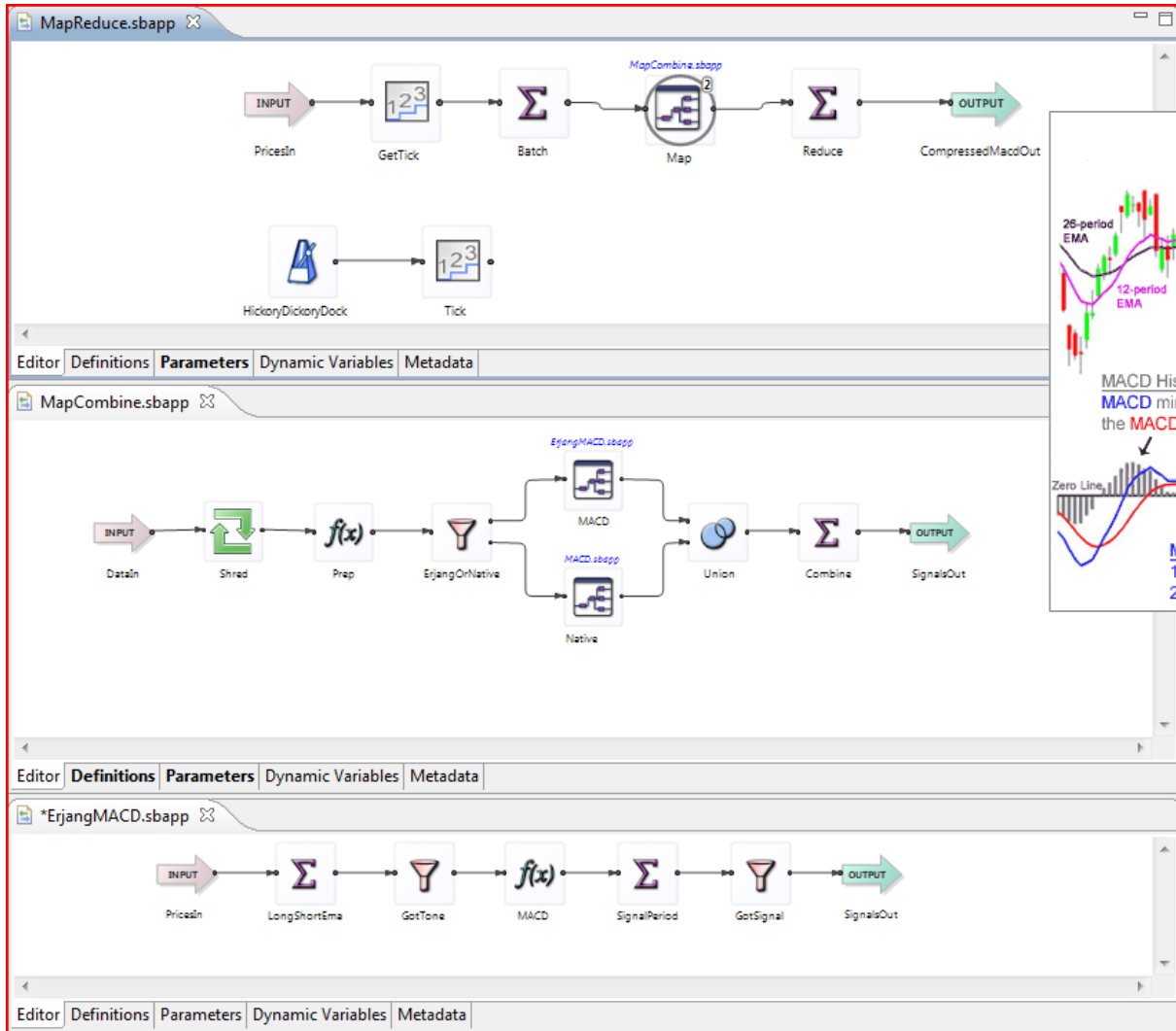
public static EObject callerl_accumulate(String mod, EObject state, Tuple args, Tuple hint) {
    EAtom m = EAtom.intern(mod);
    ETuple et = (ETuple)RPC.call(
        m, am_accumulate, state,
        sbToErjang(args, CompleteDataType.forTuple(args.getSchema())));
    return et.elm(2);
}

public static Tuple callerl_emit(String mod, EObject state, Tuple hint) {
    EAtom m = EAtom.intern(mod);
    EObject r = RPC.call(m, am_emit, state);
    try {
        ETuple2 t = (ETuple2)r;
        if (!t.elm(1).equals(EAtom.intern("ok"))) {
            // @NOTE: Response could be an error tuple - TBD!
            throw new StreamBaseRuntimeException("Feck");
        }
        BestGuess bg = erjangToCdt(t.elm(2)); // SB and Erlang type systems significantly different

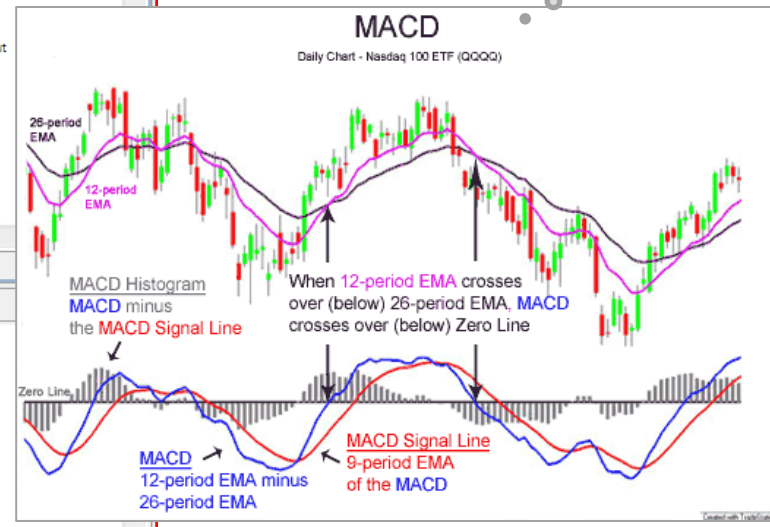
        return wrapTuple(bg);
    } catch (TupleException e) {
        throw new StreamBaseRuntimeException(e);
    }
}
```



Use, Deploy & Run



Wikipedia : MACD



Example:

Continuous Streaming Map/Reduce with 1-second MACD compression. Linearly SMP scalable.

Just add boxes to scale!



Shameless Plugs

■ StreamBase

- You could build one of these yourself, or use ours...
- Download and test out the full product <http://www.streambase.com>
- Build something and submit to the StreamBase Component Exchange <http://sbx.streambase.com>
- Contact us to buy or to an OEM partner, offices London, Boston, New York
- We're hiring
- We're training
 - <http://www.streambase.com/developers-training-events.htm>

■ DEBS – Distributed Event Based Systems

- Academic (ACM) Conference outside NYC in July

■ EPTS – Event Processing Technology Society

- <http://ep-ts.org> industry consortium

Questions?

Copyright 2011 © StreamBase Systems, Inc.



Acknowledgements

- **Erlang, Erlang Solutions, Erlang UG London, & Forward**
 - <http://www.erlang.org/>
 - <http://www.erlang-solutions.com/>
 - <http://www.erlang-solutions.com/etc/usergroup/london>
 - <http://www.forward.co.uk/> - Thanks for the venue folk!
- **Erjang – The Java based Erlang Virtual Machine**
 - <https://github.com/trifork/erjang/wiki/>
- **erlIDE**
 - <http://erlide.sourceforge.net/>
- **@tibbetts – I ~~steal~~ borrowed some of his QCon slides!**

Source will be contributed to the StreamBase Component Exchange:

<http://sbx.streambase.com/>

... in due course!



Download StreamBase and More Information

<http://www.streambase.com>

Questions?



+



+



=

