Trading Programs

How the Finance industry has become so complex that today's products are similar to programs

The University of Texas at Austin
28-Nov-2023

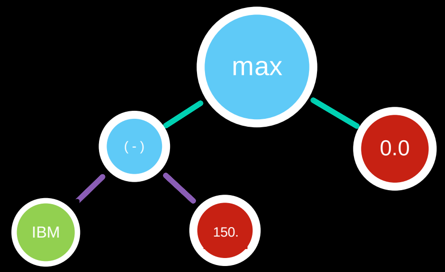
Varun Kohli, Yuecheng Shao
Software Engineer, Derivatives Pricing
vkohli5@bloomberg.net, yshao21@bloomberg.net

Bloomberg Engineering

TechAtBloomberg.com

Overview

- Data Volume / Throughput
- Data Representation
- Computation



Bloomberg – What is it?

- Founded in **1981**
- **325,000+** subscribers
- Customers in **170 countries**
- Over **19,000 employees** in 192 locations
- **More News Reporters than** the New York Times + Washington Post + Chicago Tribune

TechAtBloomberg.com

Bloomberg Tech - By the Numbers

- **More than 5,000 software engineers** (and growing)
- **100+ engineers and data scientists** devoted to machine learning
- One of the largest private networks in the world
- **100B+ tick messages** per day, with a peak of **more than 10 million** messages/second
- **2M news stories** ingested / published each day from 125K+ sources (that's >500 news stories ingested/second)
- More than **a billion messages** (emails and IB chats) processed each day

TechAtBloomberg.com

Bloomberg

Engineering

Data Volume / Throughput

- Real-time Volume
- Storage
- Live Analytics

TechAtBloomberg.com

Bloomberg

Engineering

Data Volume / Throughput

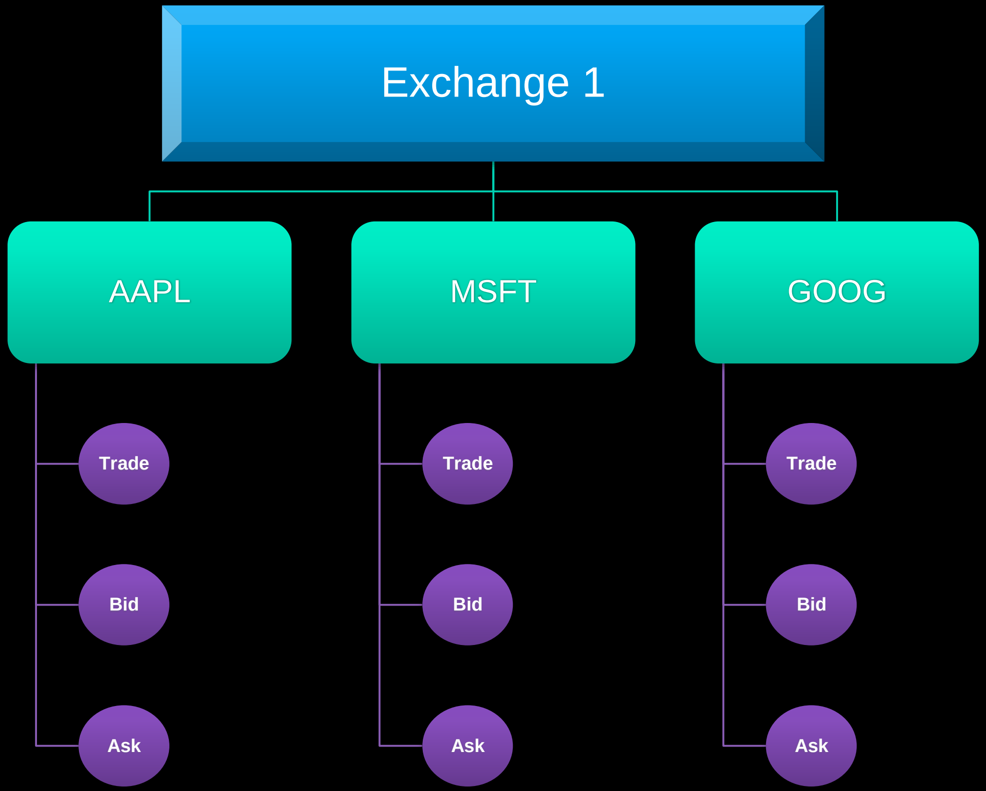
A tick is a message that describes a market data event

Examples:

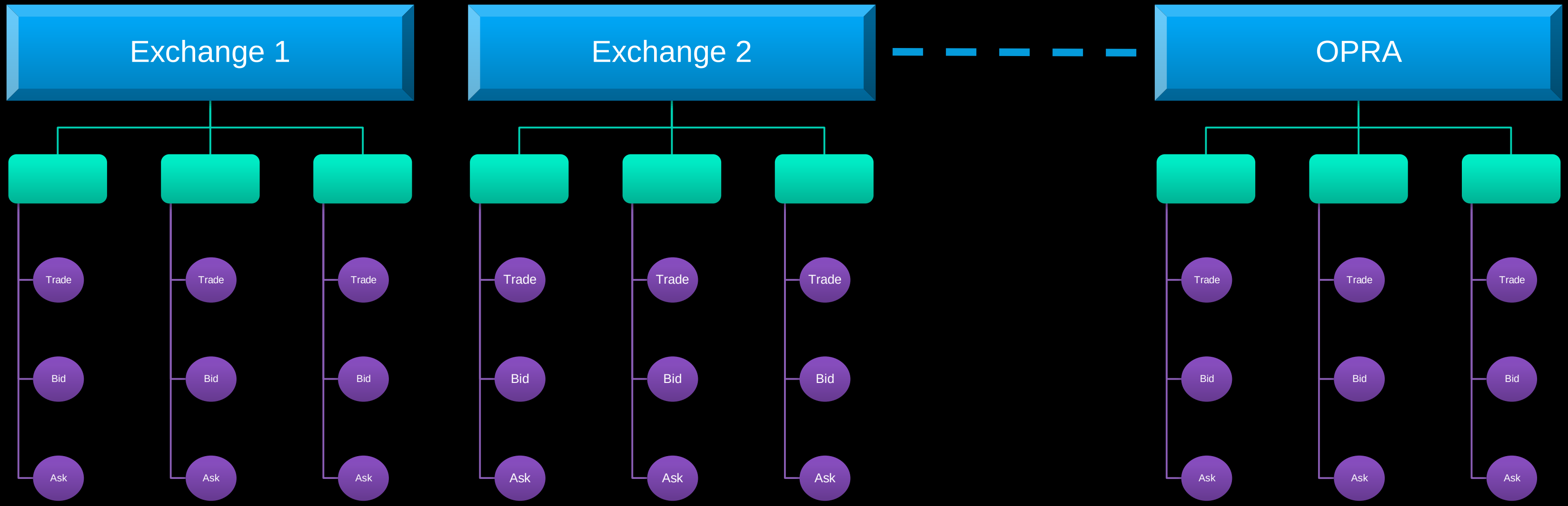
[TRADE] 57 IBM Stocks just traded for $155.2 each

[BID] Buying 32 Apple Stocks at $111.9 each

[ASK] Selling 7 Google 06/2015 $535.00 Call options at $28.8



Bloomberg Engineering



26,994,500/s

1-Second:

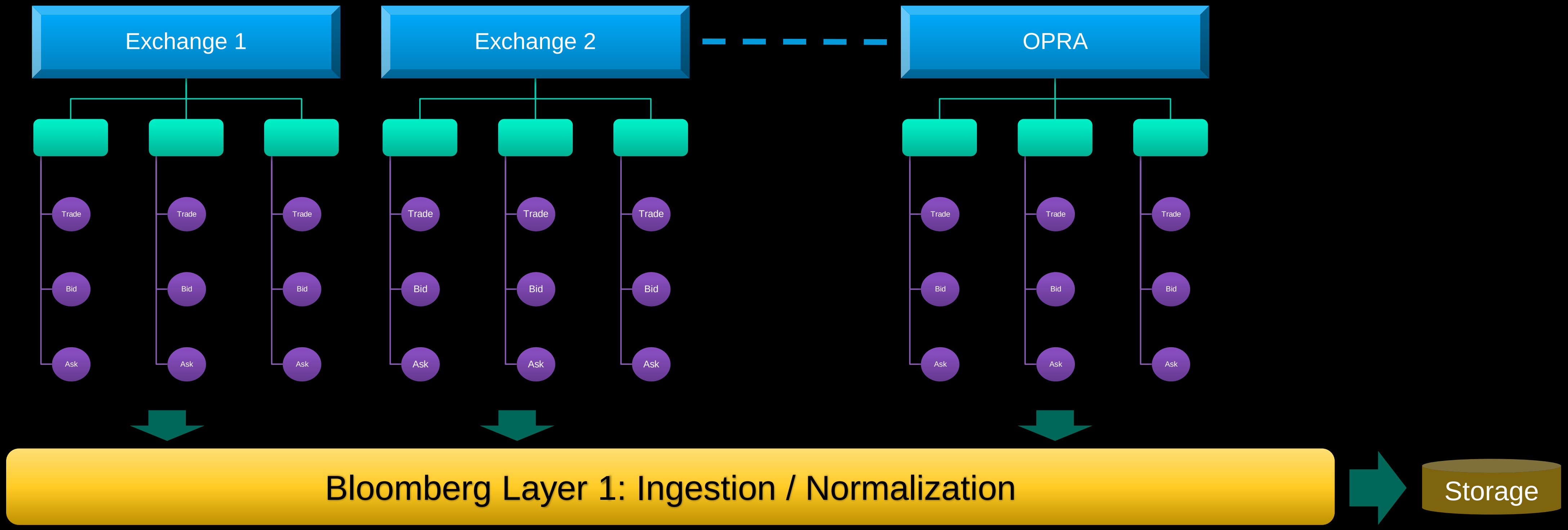
	Required Capacity Messages Per 1-Second	Bandwidth Gigabits Per Second	Bandwidth Plus 10% for Retransmissions	Total Messages Per Day
7/8/2014	22,938,500	5.51	6.06	26,448,050,000
1/6/2015	[illegible]	6.08	6.69	27,557,855,000
7/7/2015	26,994,500	6.48	7.13	29,903,390,500
1/5/2016	29,360,500	7.05	7.75	30,788,430,000

100-Milliseconds:

	Required Capacity Messages Per 100-Milliseconds	Bandwidth gabits Per 100-Millisecond
7/8/2014	3,705,000	0.89
1/6/2015	4,077,450	0.98
7/7/2015	4,581,850	1.10
1/5/2016	5,024,500	1.21

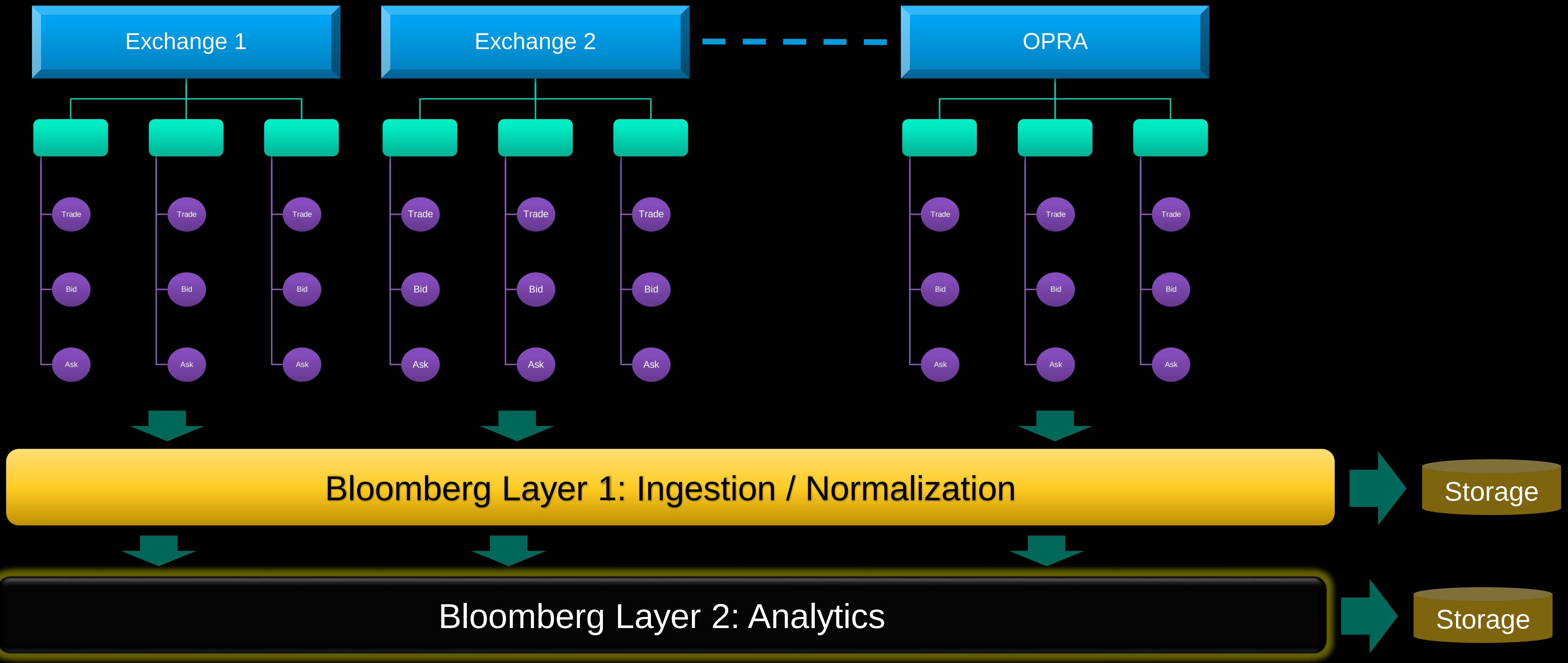
TechAtBloomberg.com

Bloomberg Engineering



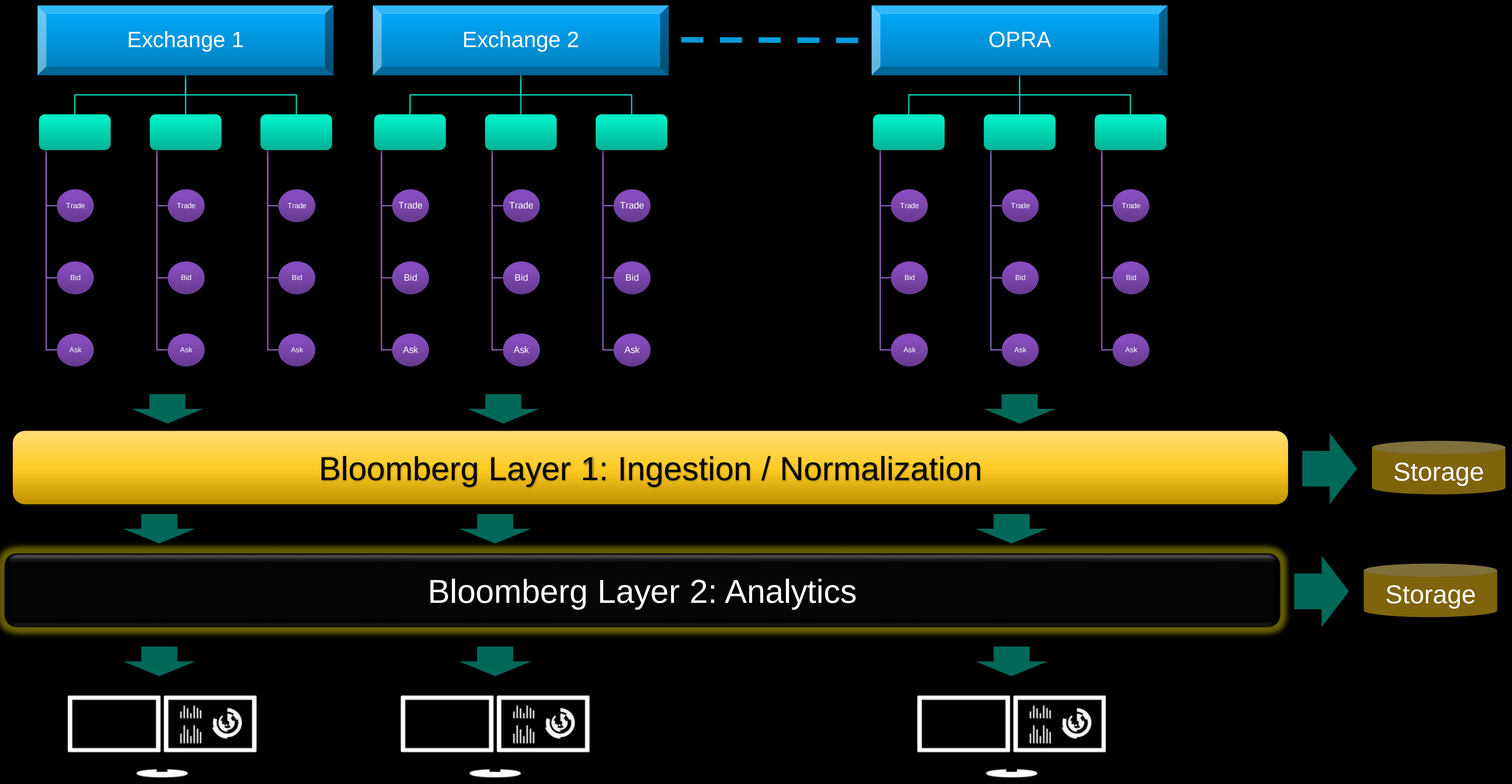
TechAtBloomberg.com

Bloomberg
Engineering



TechAtBloomberg.com

Bloomberg
Engineering



Bloomberg
Engineering

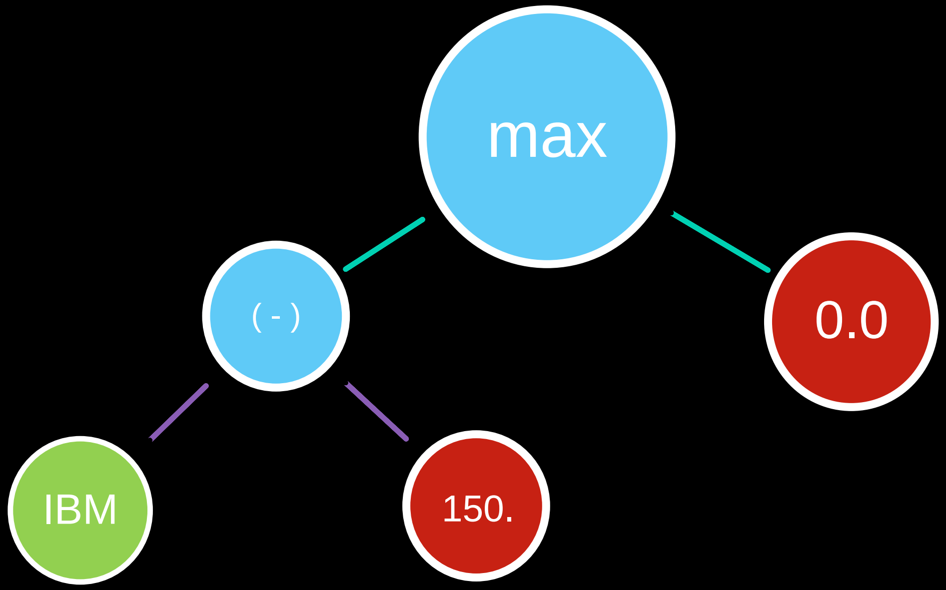
TechAtBloomberg.com

Bloomberg

Engineering

Data Representation

- Modeling exotic derivatives and smart contracts
- Allowing clients to ‘script’ financial instruments
- Automatically generating UI



What

Is

An

Exotic

Derivative?

HK$ Auto-Callable Snowball Notes
Linked to a Basket of Hong Kong Stocks due 2009

issued by Allegro Investment Corporation S.A.
pursuant to its
€10,000,000,000 Retail Secured Note Programme

Offer Period:	**From 9.00 a.m. on 26 July 2004 to 5.00 p.m. on 13 August 2004.**
Issue Price:	**100 per cent. of the principal amount**
Fixing Date:	**Expected to be 16 August 2004, on which date the Issue Size of the Notes and the Barrier Level in respect of each Share will be determined.**
Issue Date:	**Expected to be 20 August 2004 (which is four Business Days following the Fixing Date), and will not be later than 13 September 2004.**
Maturity Date:	**Expected to be 20 August 2009 (which is five years following the Issue Date)**

The Notes will be issued by the Issuer and all payments to be made by the Issuer under the Notes will only be made from the proceeds of a swap agreement (the "**Swap Agreement**") with Citigroup Global Markets Limited (the "**Swap Counterparty**").

Prospective purchasers of the Notes should ensure that they understand the nature of the Notes and should carefully study the matters set out in the sections headed "Risk Factors" in this Issue Prospectus and in the Programme Prospectus before they invest in the Notes.

You should contact one of the Distributors listed below during the Offer Period to invest in the Notes. Investments in the Notes may only be made through the Distributors, whose contact telephone numbers are listed on the following page. In order to invest in Notes through a Distributor you must already have, or you must open, a bank account and an investment account with that Distributor in the same currency as your Notes. No application form is being issued for the Notes. No Notes are available from the Issuer or the Arranger directly.

A copy of this Issue Prospectus has been registered by the Registrar of Companies in Hong Kong as required by Section 342C of the Companies Ordinance (Cap. 32) of Hong Kong (the "**Companies Ordinance**"). The Registrar of Companies in Hong Kong and the Securities and Futures Commission (the "**SFC**") take no responsibility as to the contents of this Issue Prospectus.

Arranger

CITIGROUP GLOBAL MARKETS LIMITED

Portuguese Train Company Was Run Over by a Snowball

May 2, 2014 by Matt Levine on Bloomberg View

https://www.bloomberg.com/view/articles/2014-05-02/portuguese-train-company-was-run-over-by-a-snowball

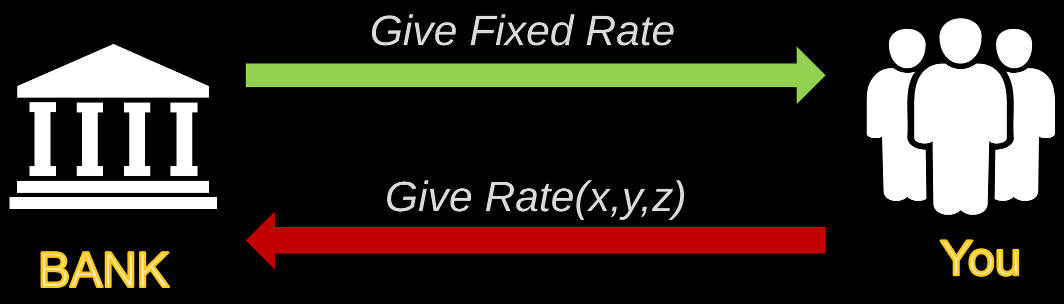
Train, snow, but not Portugal.

Portuguese Train Company Was Run Over by a Snowball

There is **no** giant Snowball in Portugal

… however …

It's a complex **derivative bond:**



TechAtBloomberg.com

Bloomberg

Engineering

Portuguese Train Company Was Run Over by a Snowball

- *Metro do Porto:* state-backed rail operator
- State company has a massive impact on the country’s economy
- This is real life. It affects people.

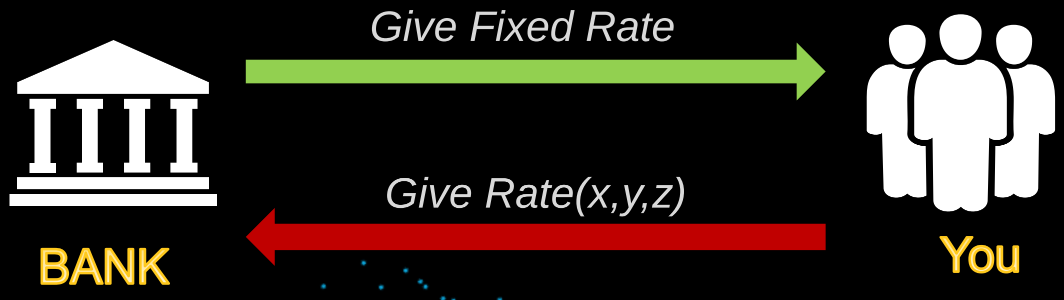
Portuguese Train Company **Was Run Over** by a Snowball

- Bad outcomes **DO** happen
- How could we prevent that ?

What is a Snowball?

- Or more precisely: ***Give Rate (x,y,z)***

- Let's look at the variables:
  - **EURIBOR 6 month:** The rate at which banks lend themselves EUR for 6 month periods. This can be used to represent interest rate and the value is updated daily.
  - **Previous Rate:** The rate at date **t** depends on rate at **t–1**

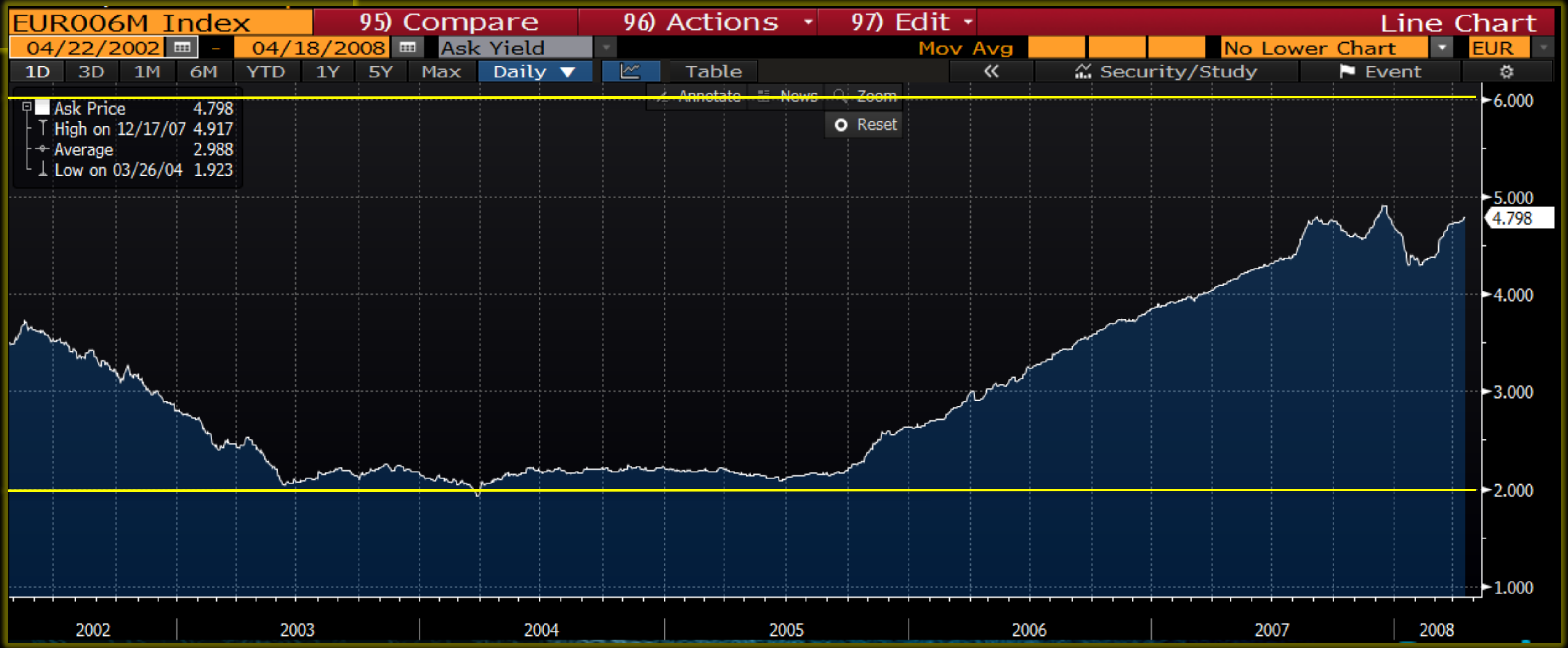


What is a Snowball?

```
let coupon_rate(t) = max(0, coupon_rate(t-1) + spread(t))

let spread(t) =
  if        EURIBOR > 6% then 2 * (EURIBOR - 6%)
  else if   EURIBOR < 2% then 2 * (2% - EURIBOR)
  else                        -0.5%
```

What is a Snowball?

```
let coupon_rate(t) = max(0, coupon_rate(t-1) + spread(t))

let spread(t) =
  if        EURIBOR > 6% then 2 * (EURIBOR - 6%)
  else if   EURIBOR < 2% then 2 * (2% - EURIBOR)
  else                         -0.5%
```



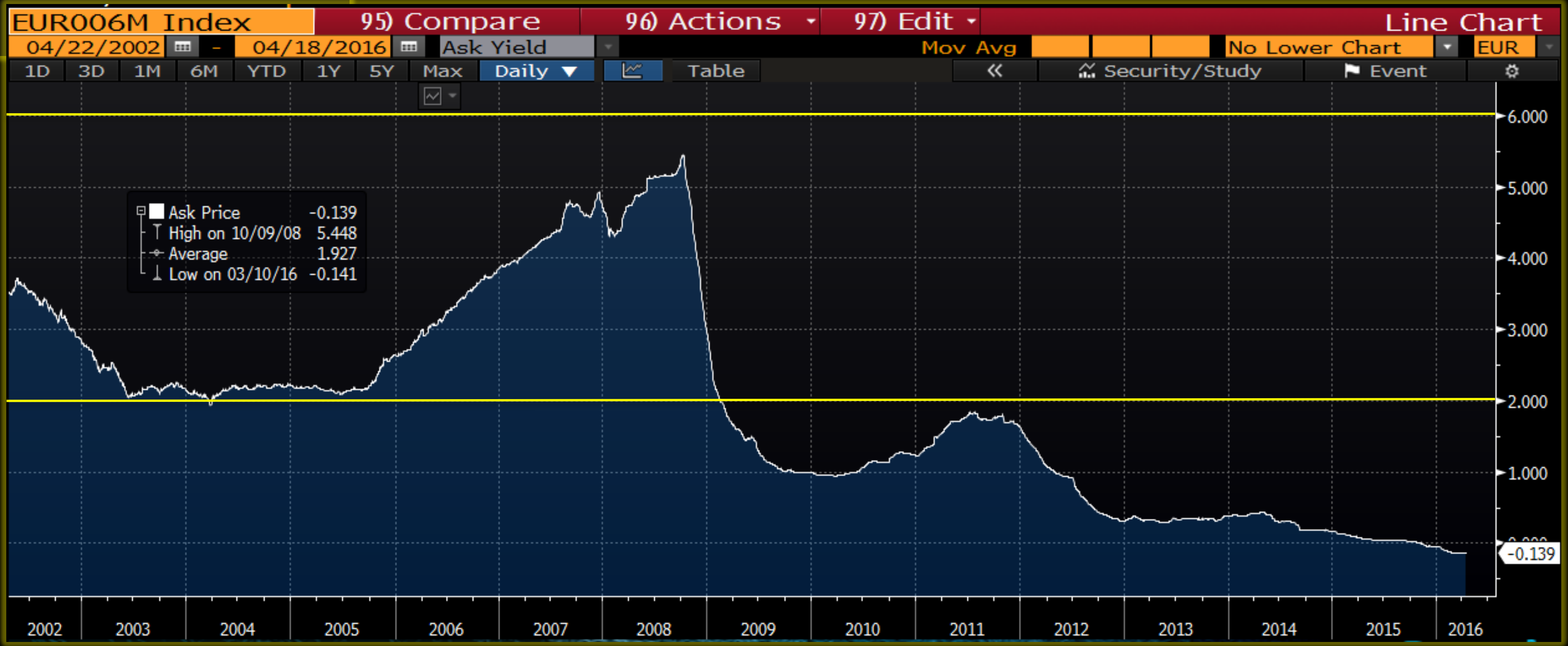
TechAtBloomberg.com

Bloomberg

Engineering

What is a Snowball?

```
let coupon_rate(t) = max(0, coupon_rate(t-1) + spread(t))

let spread(t) =
  if        EURIBOR > 6% then 2 * (EURIBOR - 6%)
  else if   EURIBOR < 2% then 2 * (2% - EURIBOR)
  else                        -0.5%
```



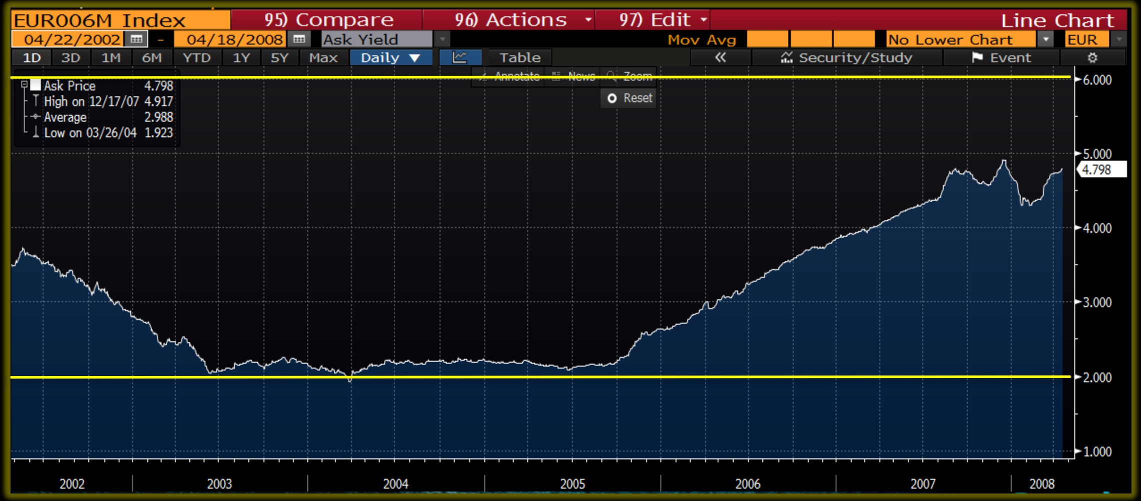
TechAtBloomberg.com

Bloomberg

Engineering

Why did they buy it?

This is the history of EURIBOR before they bought the contract



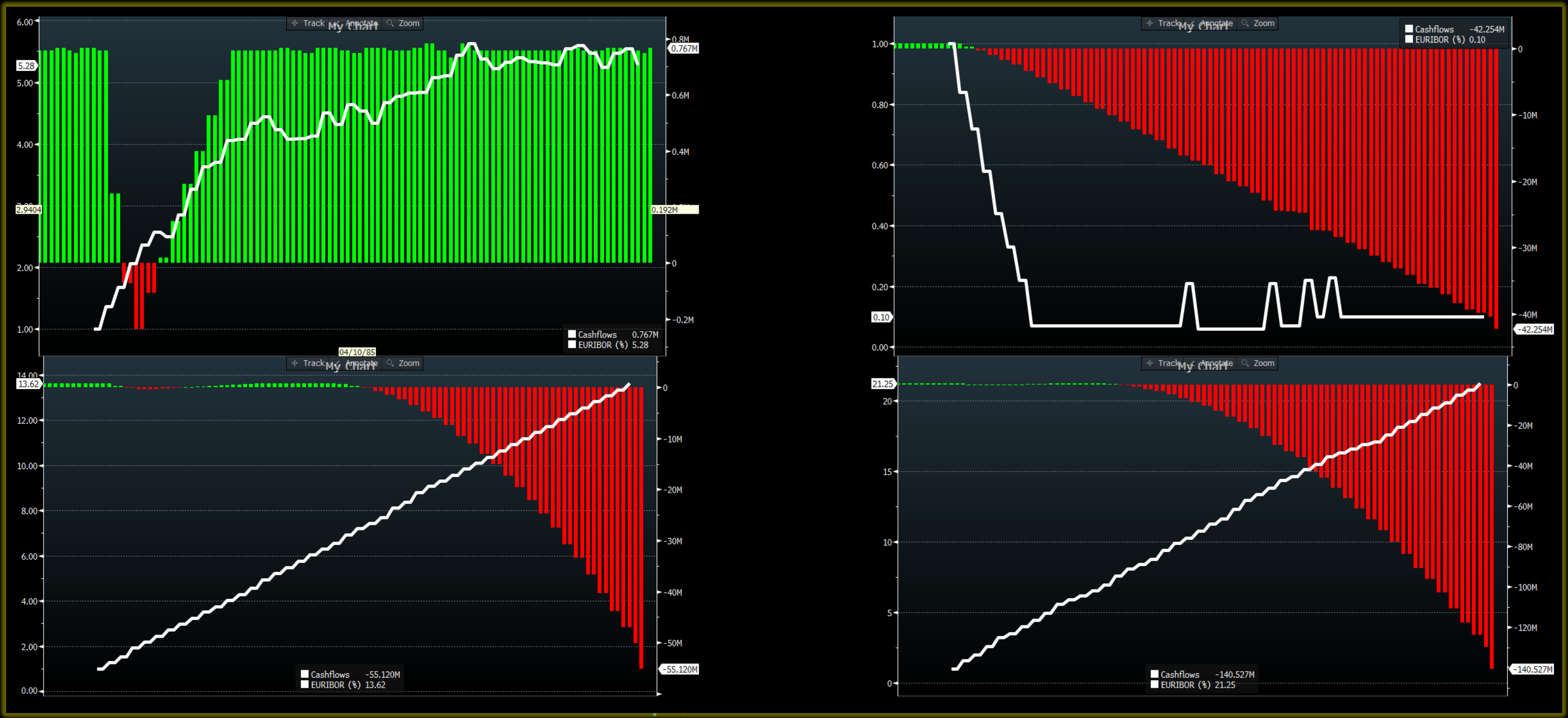
This is what happened to EURIBOR after they bought the contract

Where are they now?

Coupon Rate: 40.6% !!!!

- They actually stopped paying… there is a lawsuit
- What they lacked was the right tools to fully understand the impact of the trade

Preemptive Payoff Analysis



Contract Representation

- **Object Oriented/Imperative Approach**
  - 1 class for business representation
  - 1 class for UI
  - 1 class for Pricing (QFD)
  - 1 class for database layer
  - Lifecycle, model integration … needs to be implemented every time

Contract Representation

- **Functional Approach**
  - Algebraic representation of contract
    - 30 combinators are enough to build any financial contracts
  - Business representation + Pricing representation
  - Single OCaml type to define contract inputs
  - UI representation + Database layer

What is OCaml?

Functional language: functions are values

```ocaml
(* Name functions *)
let my_fun_function = fun x -> x + 1

(* Use functions as arguments *)
let my_funnier_function f x =
  let y = x * 2 in
  f y

(* Return functions as values *)
let the_funiest_function f =
  fun x -> my_funnier_function f x
```

OCaml Type System: Tuples

```
(* Product type (i.e. Tuples) *)
type t = float * string
let  a = (3.14, "thon")

(* Tuples of Tuples *)
type t = float * (string * int)
let  a = (1., ("for all, all for", 1))
```

TechAtBloomberg.com

Bloomberg

Engineering

OCaml Type System: Records

```
(* Records: Named tuples or structures *)
type a_thing = {
  quantity : float  ;
  of_what  : string ;
}

let a = {
  quantity = 3.14;
  of_what  = "thon"
}
```

OCaml Type System: Unions

```ocaml
(* Unions without parameters *)
type t =
   | Nothing
   | Something

(* With parameters *)
type t =
   | Nothing
   | Something of a_thing

type t =
   | Nothing
   | Something of a_thing * string
```

Floating Point Algebra (definition)

- OCaml is very well suited to represent and manipulate algebras
- Here is the representation of floating point algebra

```
type exp =
   | Constant of float
   | Plus     of exp * exp
   | Minus    of exp * exp
   | Multiply of exp * exp
   | Divide   of exp * exp
   | Var      of string

(* Expression for: (Y + 1) * 3  *)
let _ =  Multiply (Plus (Var "Y", Constant 1.), Constant 3.)  in
```

Floating Point Algebra: Simplifying Expressions

```
let rec simplify env a =
  match a with
  | Plus (l, r) ->
    (match (simplify env l), (simplify env r) with
    | Float 0.,  r         -> r
    | l        , Float 0. -> l
    | Float l , Float r   -> Float (l +. r)
    | l        , r         -> Plus  (l , r ))
  | Minus (l, r) ->
    ...
```

TechAtBloomberg.com

Bloomberg

Engineering

Floating Point Algebra: Simplifying Expressions

```
(* Summing any expression with 0 is
   equal to the expression *)
| Float 0.,  r        -> r
| l        , Float 0. -> l


(* Summing 2 constant expression equals a
   constant expression whose value
   is sum of the 2 constants *)
| Float l , Float r  -> Float (l +. r)
```

TechAtBloomberg.com

Bloomberg

Engineering

Algebra for financial contracts

Composing Contracts: An Adventure in Financial Engineering

September 2000, Simon Peyton Jones, Jean-Marc Eber , and Julian Seward

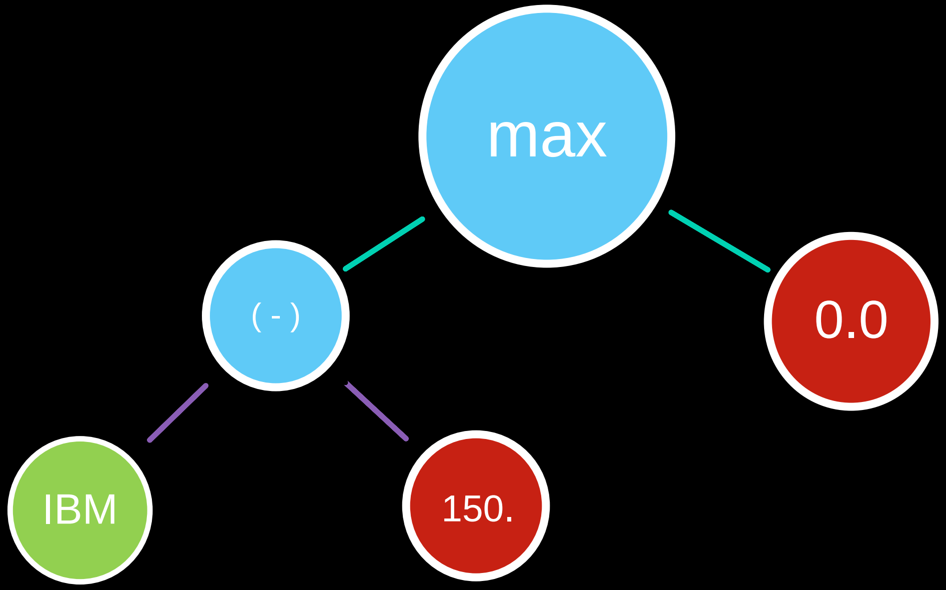
```
type cash = {
  payment_date : date;
  amount : float;
  currency : string;
}

(* Cashflow of $100 on 2020-01-01 *)
let _ = cash 2020-01-01 100. USD
```

Algebra for contract: Combining

```
type contract =
   | Cash of cash
   | Give of contract
   | And  of contract * contract
   | Or   of contract * contract

(* Creating a snowball! *)
let snowball =
  And (
        (Cash …),
        (Give (Cash …))
      )
```

Applications

Traverse the symbolic expression to derive other representations



Applications (1)

- Lifecycle Handling
  - **Cashflow Reporting:** For back-office purposes, one must be able to report all the cashflows defined by a financial contract

TechAtBloomberg.com

Bloomberg

Engineering

Applications (2)

- Pricing
  - Using the Monte Carlo technique, the contract can be priced. The algebra structure is used to generate C code.

```
for i in [1..20000]
  random_i    = Random_Number_X::generate
  path_i      = Model_Y::generatePath  (random_i)
  cashflow_i = Contract_Y::calculateCashflow (path_i)
end

price = average (present_value (cashflow_i))
```

Applications (2)

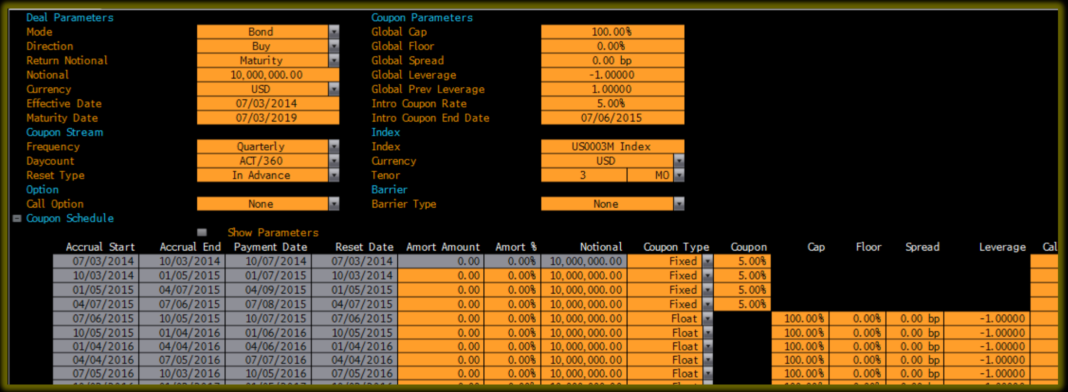
- Pricing
  - Using the Monte Carlo technique, the contract can be priced. The algebra structure is used to generate C code.
  - Example of `Contract_Y::calculateCashflow`

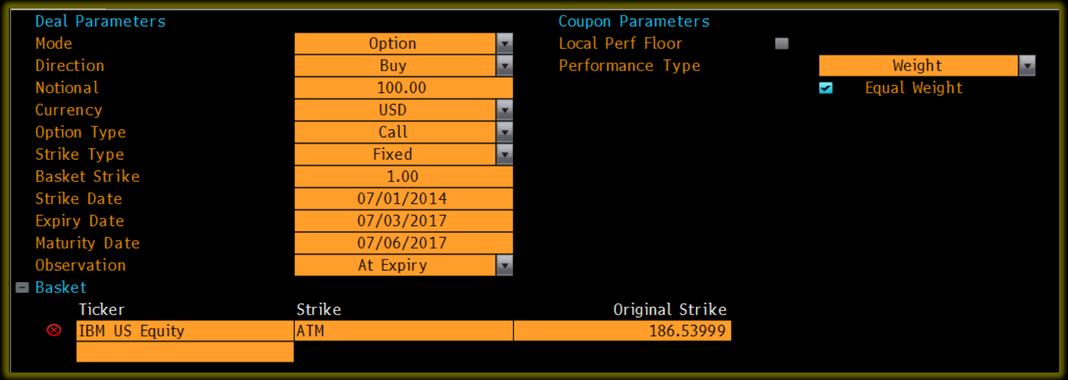
```
static void calculate_cashflows(matrix path) {
  cash_flow(100., 0);
  cash_flow((100. * fmax(0., ((path[0][0] / 180) - 1.))), 1);
  return;
}
```

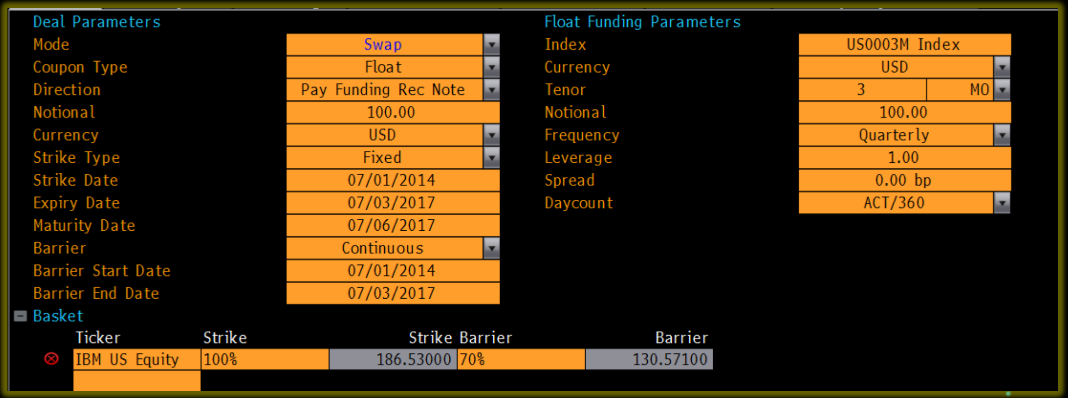
UI Representation of Contract

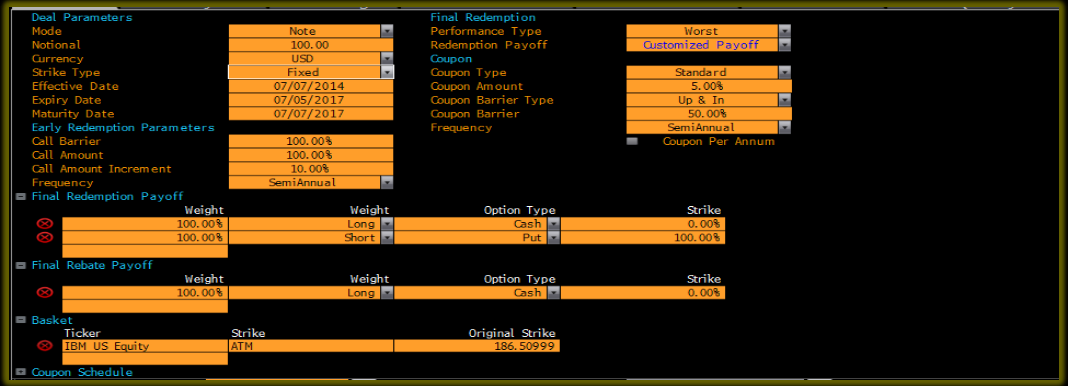
Problem : 100s of Template/Entry screens

Every week we have business requests





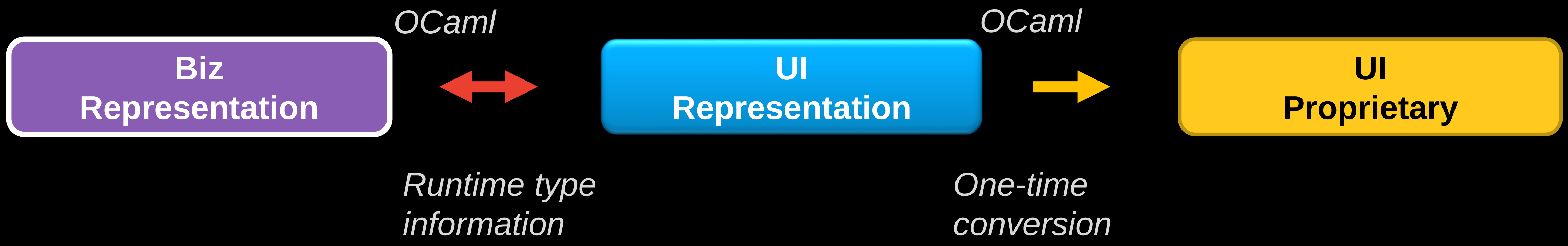




UI Representation of Contract

UI generation using type reflection

- Leveraging both the OCaml expressive type system and type reflection
- The UI is automatically generated



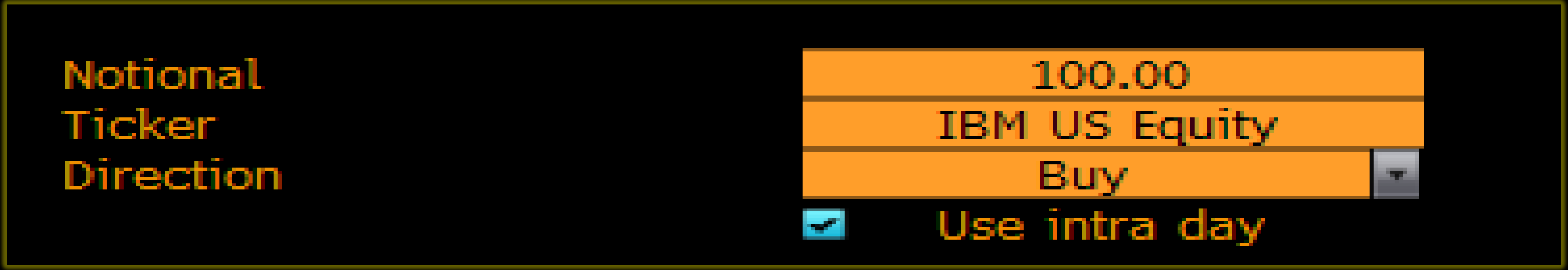
UI Representation of Contract

From OCaml to Type to the Terminal

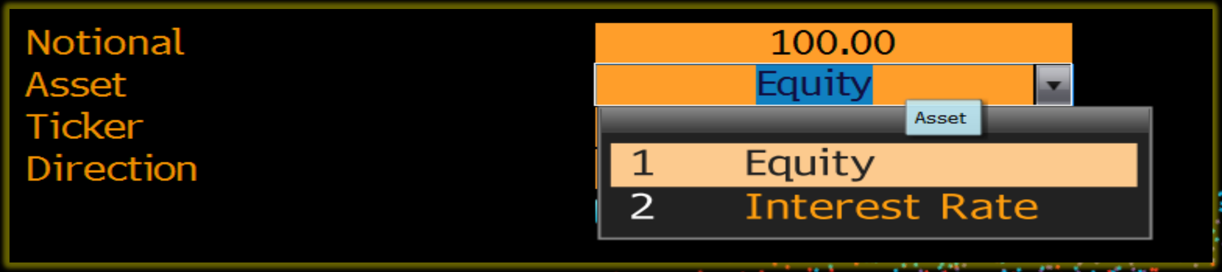
```
type parameters = {
   notional : float ;
   ticker   : string +  [autocomplete="Equity^1788633";];
}
```

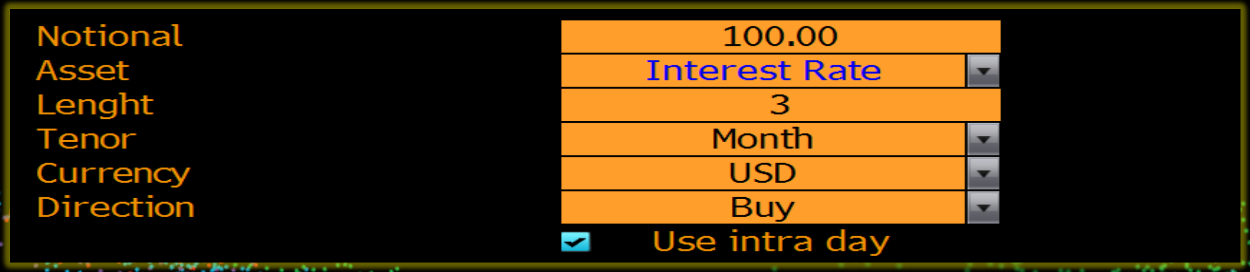
UI Representation of Contract

```
type direction = | Buy | Sell

type parameters = {
  notional     : float ;
  ticker       : string +  [autocomplete="Equity^1788633";];
  direction    : direction ;
  use_intra_day: bool ;
}
```

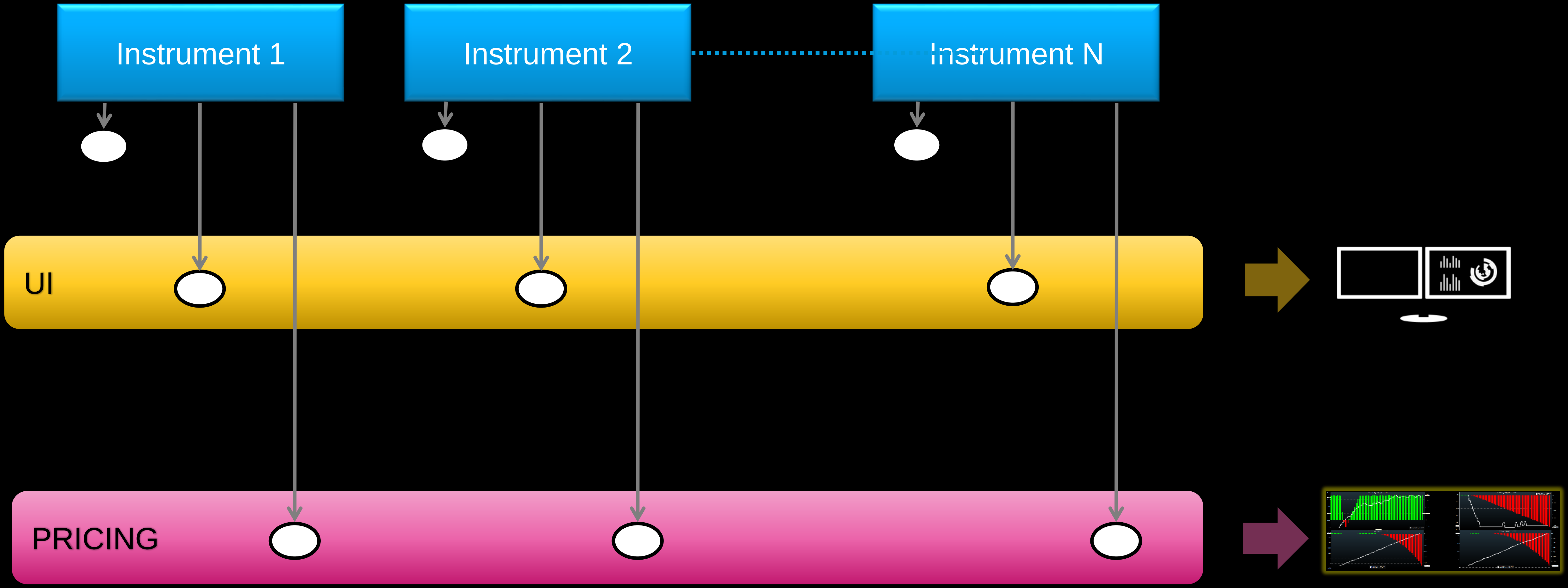


```
type equity_input = {
  ticker      : string +  [autocomplete="Equity^1788633";];
}

type unit_t =   | Month   | Year

type ir_input =   {
  lenght    : int + [init = "3" ];
  tenor     : unit_t + [init = "Month";] ;
  currency : currency;
}

type asset = | Equity of equity_input | Interest_Rate of ir_input

type direction = | Buy | Sell

type parameters = {
  notional     : float ;
  asset        : asset;
  direction   : direction ;
  use_intra_day: bool ;
}
```

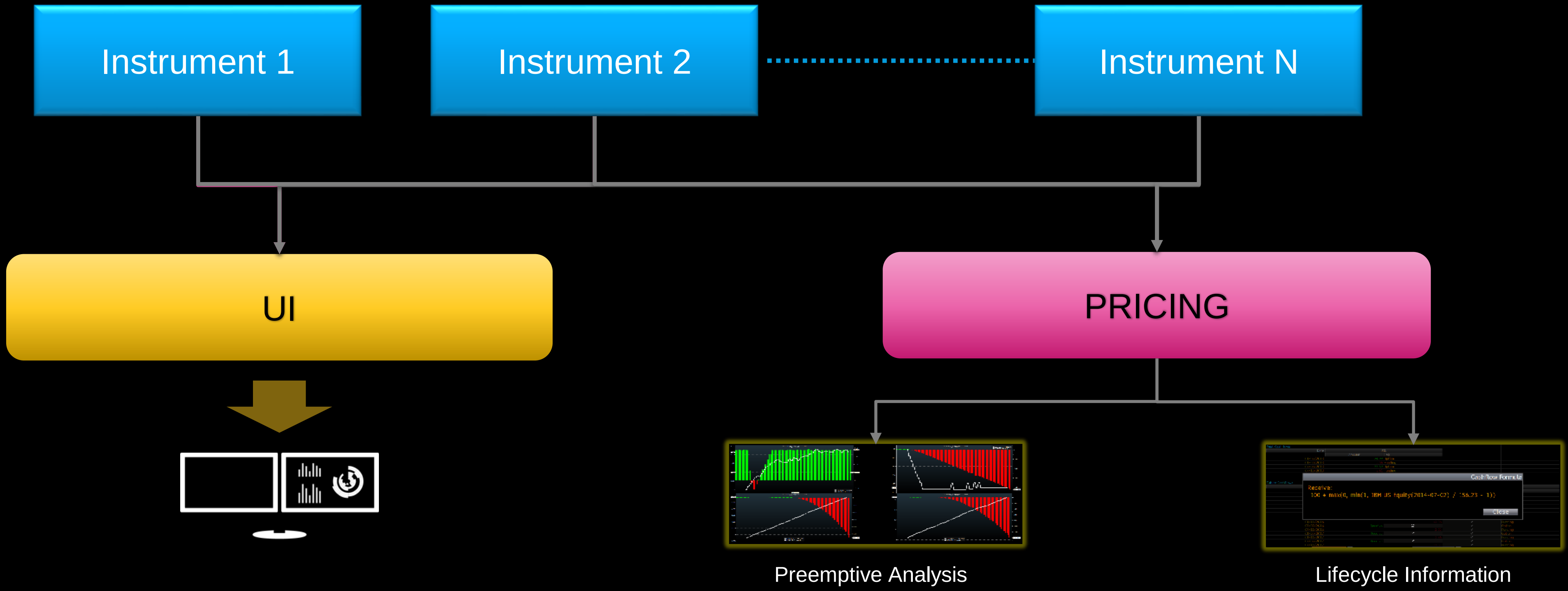




Impact of OCaml in the tech stack



Impact of OCaml in the tech stack



More on Functional Programming (OCaml)

- Strong typing
  - It really does help development

- More rigorous type system
  - So many errors and bugs are due to the inability of a type system to accurately represent data
  - Only the minimum amount of invariant-checking code should be needed

- Type inference
  - Concise syntax

More on Functional Programming (OCaml)

- Large toolset to create domain specific languages easily
  - BLAN is an example

```
(**** Example Simple Vanilla Option ****)

let underlying          =   market("IBM US Equity") in
let strike_value        =   1.1   in
let pay_currency        =   "USD" in
let notional            =   100.  in

let strike_date         =   2018-08-27 in
let maturity_date       =   2019-08-27 in
let payment_date        =   2019-08-30 in

let initial_spot        =   fix(strike_date,   underlying) in
let final_spot          =   fix(maturity_date, underlying) in

flow(payment_date, pay_currency,
     max (notional * ( (final_spot/initial_spot) - strike_value), 0.0) )
```

TechAtBloomberg.com

Bloomberg

Engineering

More on Functional Programming (OCaml)

- Can be used to transpile
  - BuckleScript: OCaml -> JS compiler (https://github.com/BuckleScript/bucklescript)

OCaml

```ocaml


let rec append l1 l2 =
  match l1 with
    [] -> l2
  | hd :: tl -> hd :: (append tl  l2)

let rec sort ls =
  match ls with
  | [] -> []
  | x::xs ->
    append
    (sort (List.filter (fun u -> u <= x) xs))
    (x :: sort (List.filter (fun u -> u > x) xs))

let () =
    [| 1; 3 ; 2; 4; 5; 10; 23; 3|]
    |> Array.to_list
    |> sort
    |> Array.of_list
    |> Js.log
```

JavaScript

```
Warnings:

> 1,2,3,3,4,5,10,23

```

```javascript
// Generated by BUCKLESCRIPT VERSION 3.2.0, PLEASE EDIT WITH CARE
'use strict';

var List = require("./stdlib/list.js");
var $$Array = require("./stdlib/array.js");
var Caml_obj = require("./stdlib/caml_obj.js");

function append(l1, l2) {
  if (l1) {
    return /* :: */[
            l1[0],
            append(l1[1], l2)
          ];
  } else {
    return l2;
  }
}

function sort(ls) {
  if (ls) {
    var xs = ls[1];
    var x = ls[0];
    return append(sort(List.filter((function (u) {
                          return Caml_obj.caml_lessequal(u, x);
                        }))(xs)), /* :: */[
                x,
                sort(List.filter((function (u) {
                            return Caml_obj.caml_greaterthan(u, x);
                          }))(xs))
              ]);
  } else {
    return /* [] */0;
  }
}

console.log($$Array.of_list(sort($$Array.to_list(/* array */[
                  1,
                  3,
                  2,
                  4,
                  5,
                  10,
                  23,
                  3
                ]))));

exports.append = append;
exports.sort = sort;
/*  Not a pure module */

```

Computation

- Advanced quant models
- Computationally intensive calculations
- Memory footprint

TechAtBloomberg.com

Bloomberg

Engineering

Derivatives / Finance Challenges

- Derivative trade is an "Over The Counter" trade
  - No price available on exchange for that specific trade
  - However some 'similar' contracts are being priced on exchanges

- Market Value?
  - Finance Industry relies on mathematics to compute the market value
  - The underlying dynamic is modeled with *Stochastic Differential Equation*
  - Most famous: Black Scholes

$$\frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} + rS\frac{\partial V}{\partial S} - rV = 0$$

TechAtBloomberg.com

Bloomberg Engineering

Derivatives / Finance Challenges

- Computations
  - [1973] Black Scholes
  - [1997] Libor Market Models

$$\frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} + rS\frac{\partial V}{\partial S} - rV = 0$$

TechAtBloomberg.com

Bloomberg

Engineering

Derivatives / Finance Challenges

- Computations
  - [1973] Black Scholes
  - [1997] Libor Market Models

$$\frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} + rS\frac{\partial V}{\partial S} - rV = 0$$

$$dL_j(t) = \sigma_j(t)L_j(t)dW^{Q_{T_{j+1}}}(t).$$

$$dL_j(t) = \begin{cases} L_j(t)\sigma_j(t)dW^{Q_{T_p}}(t) - L_j(t)\sum_{k=j}^{p-1}\frac{\delta L_k(t)}{1+\delta L_k(t)}\sigma_j(t)\sigma_k(t)\rho_{jk}dt & j < p \\ L_j(t)\sigma_j(t)dW^{Q_{T_p}}(t) & j = p \\ L_j(t)\sigma_j(t)dW^{Q_{T_p}}(t) + L_j(t)\sum_{k=p}^{j-1}\frac{\delta L_k(t)}{1+\delta L_k(t)}\sigma_j(t)\sigma_k(t)\rho_{jk}dt & j > p \end{cases}$$

$$dW^{Q_{T_j}}(t) = \begin{cases} dW^{Q_{T_p}}(t) - \sum_{k=j}^{p-1}\frac{\delta L_k(t)}{1+\delta L_k(t)}\sigma_k(t)dt & j < p \\ dW^{Q_{T_p}}(t) & j = p \\ dW^{Q_{T_p}}(t) + \sum_{k=p}^{j-1}\frac{\delta L_k(t)}{1+\delta L_k(t)}\sigma_k(t)dt & j > p \end{cases}$$

Derivatives / Finance Challenges

- Computations
  - [1973] Black Scholes
  - [1997] Libor Market Models

$$\frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} + rS\frac{\partial V}{\partial S} - rV = 0$$

~ 1ms

$$dL_j(t) = \sigma_j(t) L_j(t) dW^{Q_{T_{j+1}}}(t).$$

~ 10 min

$$dL_j(t) = \begin{cases} L_j(t)\sigma_j(t)dW^{Q_{T_p}}(t) - L_j(t)\sum_{k=j}^{p-1}\frac{\delta L_k(t)}{1+\delta L_k(t)}\sigma_j(t)\sigma_k(t)\rho_{jk}dt & j < p \\ L_j(t)\sigma_j(t)dW^{Q_{T_p}}(t) & j = p \\ L_j(t)\sigma_j(t)dW^{Q_{T_p}}(t) + L_j(t)\sum_{k=p}^{j-1}\frac{\delta L_k(t)}{1+\delta L_k(t)}\sigma_j(t)\sigma_k(t)\rho_{jk}dt & j > p \end{cases}$$

$$dW^{Q_{T_j}}(t) = \begin{cases} dW^{Q_{T_p}}(t) - \sum_{k=j}^{p-1}\frac{\delta L_k(t)}{1+\delta L_k(t)}\sigma_k(t)dt & j < p \\ dW^{Q_{T_p}}(t) & j = p \\ dW^{Q_{T_p}}(t) + \sum_{k=p}^{j-1}\frac{\delta L_k(t)}{1+\delta L_k(t)}\sigma_k(t)dt & j > p \end{cases}$$

Derivatives / Finance Challenges

- Computations
  - [1973] Black Scholes

$$\frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} + rS\frac{\partial V}{\partial S} - rV = 0$$

~ 1KB

  - [1997] Libor Market Models

$$dL_j(t) = \sigma_j(t)L_j(t)dW^{Q_{T_{j+1}}}(t).$$

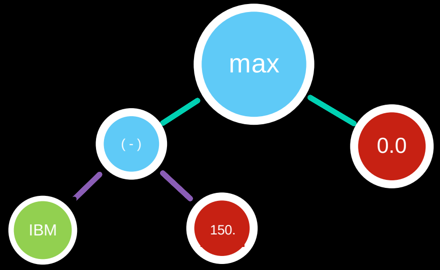
~ 10GB

$$dL_j(t) = \begin{cases} L_j(t)\sigma_j(t)dW^{Q_{T_p}}(t) - L_j(t)\sum_{k=j}^{p-1}\frac{\delta L_k(t)}{1+\delta L_k(t)}\sigma_j(t)\sigma_k(t)\rho_{jk}dt & j < p \\ L_j(t)\sigma_j(t)dW^{Q_{T_p}}(t) & j = p \\ L_j(t)\sigma_j(t)dW^{Q_{T_p}}(t) + L_j(t)\sum_{k=p}^{j-1}\frac{\delta L_k(t)}{1+\delta L_k(t)}\sigma_j(t)\sigma_k(t)\rho_{jk}dt & j > p \end{cases}$$

$$dW^{Q_{T_j}}(t) = \begin{cases} dW^{Q_{T_p}}(t) - \sum_{k=j}^{p-1}\frac{\delta L_k(t)}{1+\delta L_k(t)}\sigma_k(t)dt & j < p \\ dW^{Q_{T_p}}(t) & j = p \\ dW^{Q_{T_p}}(t) + \sum_{k=p}^{j-1}\frac{\delta L_k(t)}{1+\delta L_k(t)}\sigma_k(t)dt & j > p \end{cases}$$

Derivatives / Finance Challenges

- Data Volume / Throughput
- Data Representation
- Computation



We are hiring!

https://www.bloomberg.com/careers

Questions?

TechAtBloomberg.com