ActiveSheets

Stream Processing with a Spreadsheet

ECOOP’14

Mandana Vaziri, Olivier Tardieu, Rodric Rabbah, Philippe Suter, Martin Hirzel

IBM T.J. Watson Research Center

Introduction

- Continuous data streams
  - – Domains: telecommunications, finance, health care, transportation, etc…
  - – High volumes of data
  - – Domain experts analyze data

Price	Volume
194.77	2740
195.13	2141
195.56	2539
197.96	2639
200.69	3111
200.99	2567
199.07	2356
198.84	2987
199.15	2554

● ● ●

Introduction Cont.

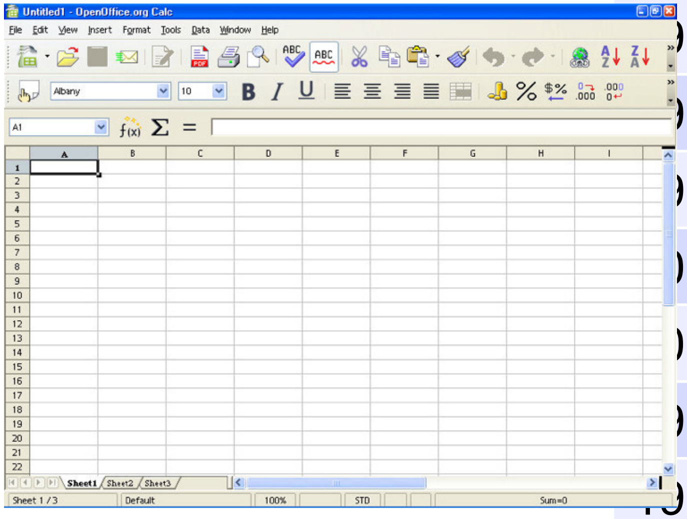
- Domain experts typically have no programming experience
  - – Rely on developers to write stream processing applications
  - – Our objective: Enable domain experts to develop stream applications directly

Price	Volume
194.77	2740
195.13	2141
195.56	2539
197.96	2639
200.69	3111
200.99	2567
199.07	2356
198.84	2987
199.15	2554

● ● ●

Introduction Cont.

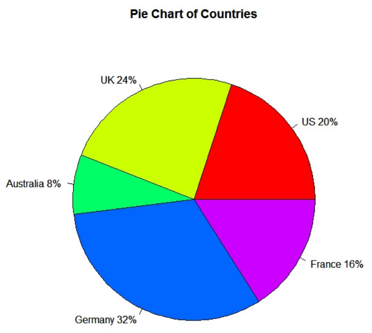
- Our solution: Principled programming model for non-programmers
  - Based on a familiar tool: spreadsheet
  - Support for live data, stateful computation
  - Formal semantics: spreadsheet calculus
  - Strong guarantees: determinism
  - Implementation in Microsoft Excel
  - Case studies to illustrate expressivity

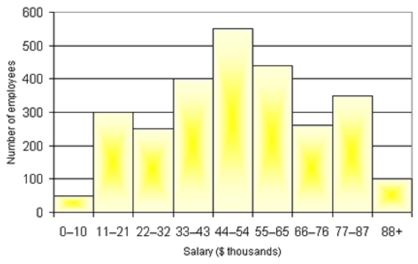

Untitled1 - OpenOffice.org Calc
File Edit View Insert Format Tools Data Window Help
Albany
10
A1
Sheet1 Sheet2 Sheet3
Sheet 1 / 3
Default
100%
STD
Sum=0

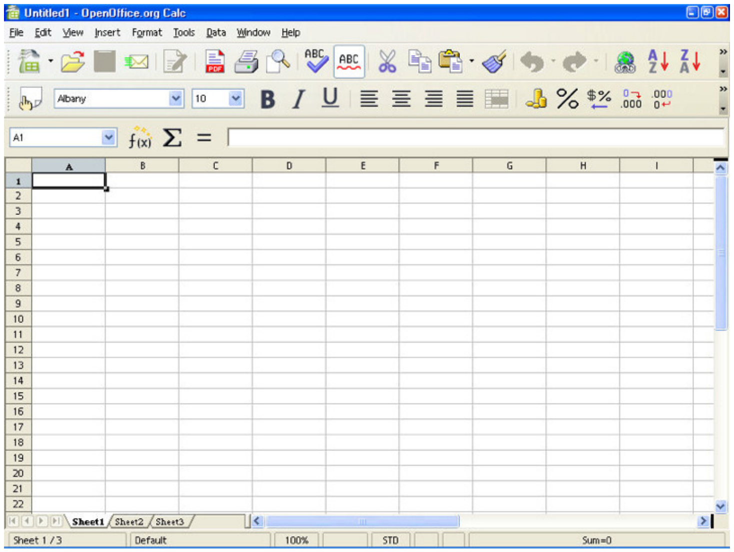
Price	Volume
194.77	2740
5.13	2141
5.56	2539
7.96	2639
0.69	3111
0.99	2567
9.07	2356
98.84	2987
199.15	2554

Why Spreadsheets?

- Easy-to-use, pervasive interface
  - – 500 million MS Excel users vs 10 million Java users (sources: mrexcel.com, wikipedia)
  - – Offer a variety of visualization possibilities
  - – User can easily compute new data
- Fluidity between code and data
  - – Unique interface where data and code that produced it can be viewed in the same place






Untitled1 - OpenOffice.org Calc
File Edit View Insert Format Tools Data Window Help
Albany
10
A1
A B C D E F G H I
Sheet1 Sheet2 Sheet3
Sheet 1 / 3
Default
100%
STD
Sum=0

ActiveSheets

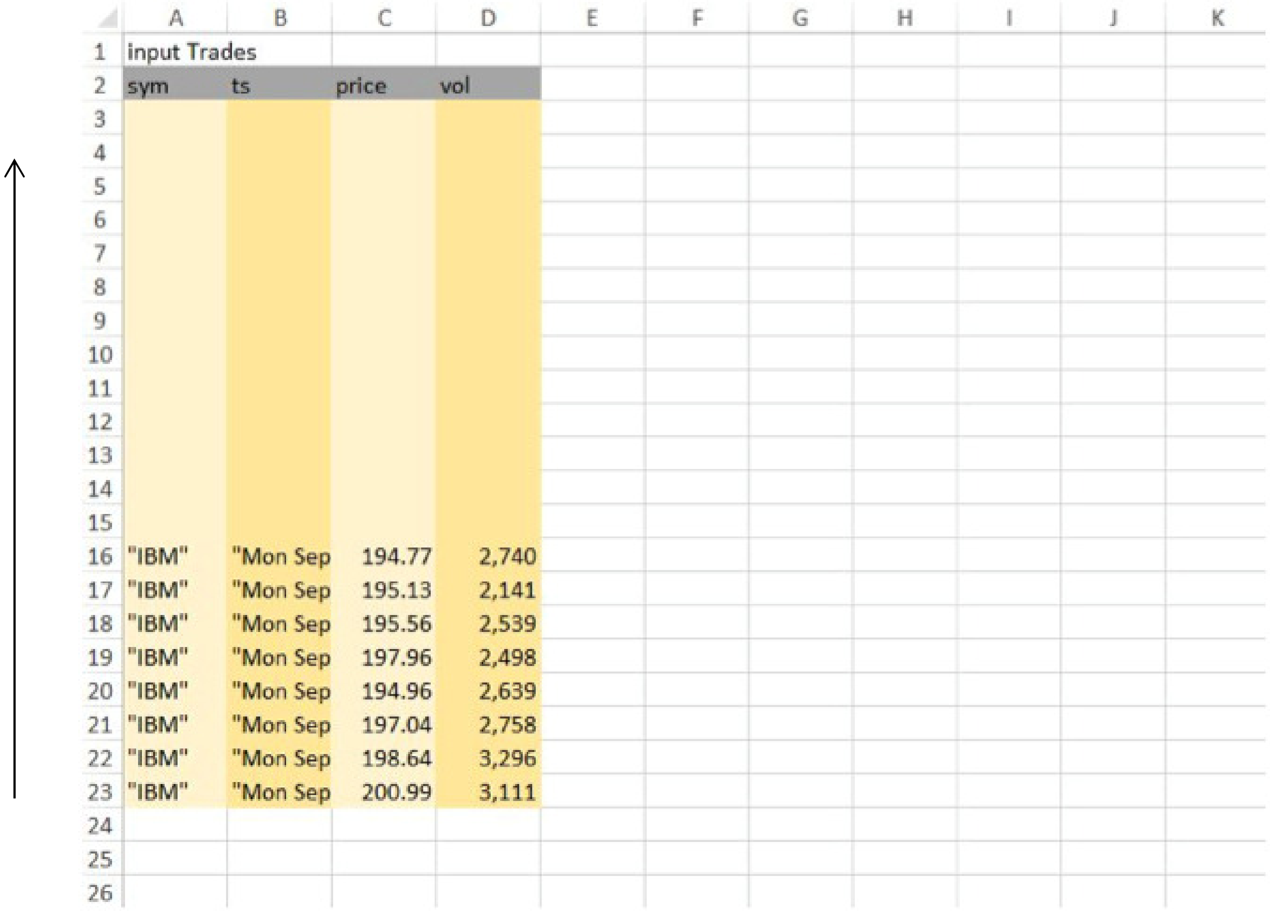
Example: Bargain Calculator for Stocks

- Program to determine bargains for stock quotes.
  - Quotes are compared to the volume-weighted average price of stocks (VWAP), and output if lower.
  - Inputs: Trades, Quotes

$$P_{\mathrm{VWAP}} = \frac{\sum_j P_j \cdot Q_j}{\sum_j Q_j}$$

P_{VWAP}	Volume Weighted Average Price
P_j	Price of Trade j
Q_j	Quantity of Trade j

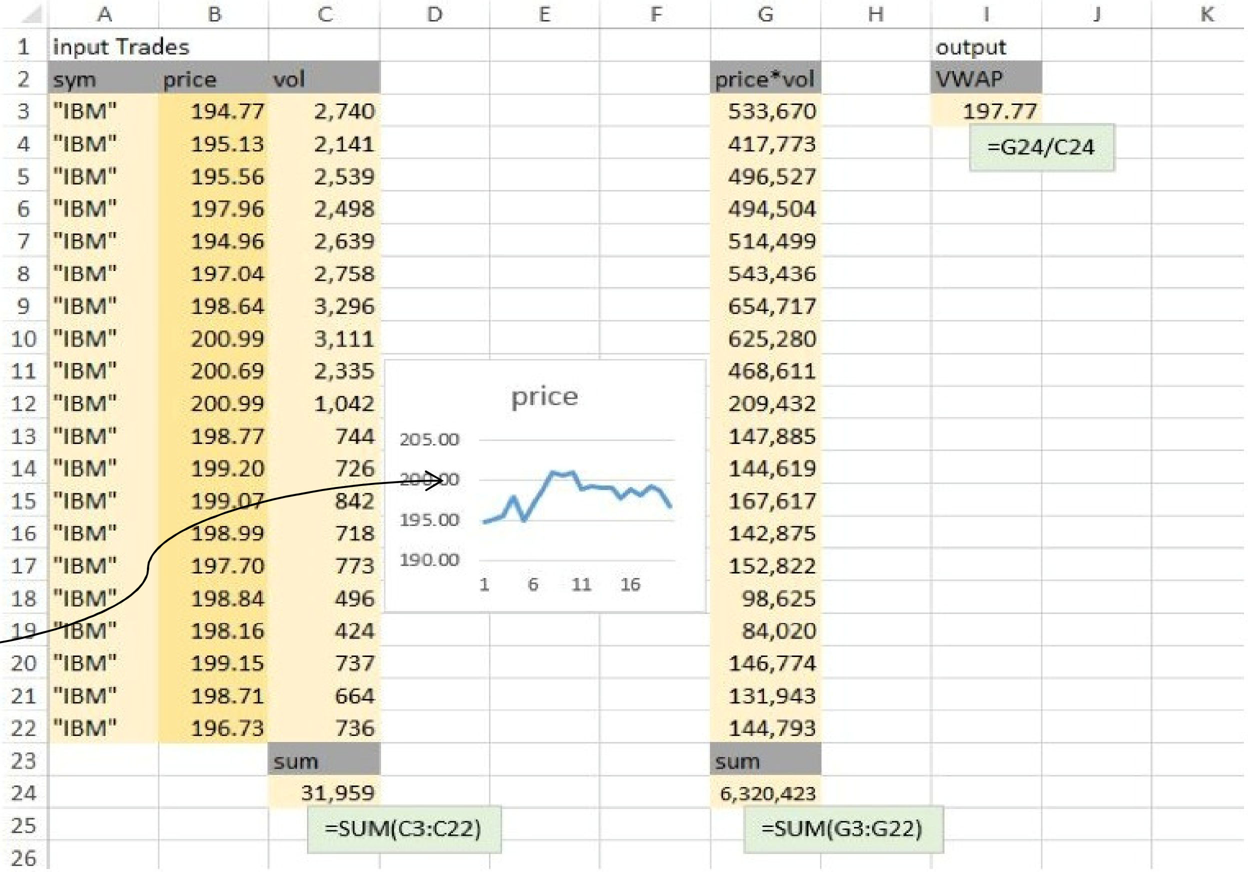
ActiveSheets: Subscribing to a Stream



	A	B	C	D	E	F	G	H	I	J	K
1	input Trades										
2	sym	ts	price	vol							
3											
4											
5											
6											
7											
8											
9											
10											
11											
12											
13											
14											
15											
16	"IBM"	"Mon Sep	194.77	2,740							
17	"IBM"	"Mon Sep	195.13	2,141							
18	"IBM"	"Mon Sep	195.56	2,539							
19	"IBM"	"Mon Sep	197.96	2,498							
20	"IBM"	"Mon Sep	194.96	2,639							
21	"IBM"	"Mon Sep	197.04	2,758							
22	"IBM"	"Mon Sep	198.64	3,296							
23	"IBM"	"Mon Sep	200.99	3,111							
24											
25											
26											

- Client/Server architecture
  - Server publishes streams
  - Client (spreadsheet) can subscribe to them
- Visualization of live data
- Ability to pause and continue a live data stream

VWAP in ActiveSheets



	A	B	C	D	E	F	G	H	I	J	K
1	input Trades								output		
2	sym	price	vol				price*vol		VWAP		
3	"IBM"	194.77	2,740				533,670		197.77		
4	"IBM"	195.13	2,141				417,773				
5	"IBM"	195.56	2,539				496,527				
6	"IBM"	197.96	2,498				494,504				
7	"IBM"	194.96	2,639				514,499				
8	"IBM"	197.04	2,758				543,436				
9	"IBM"	198.64	3,296				654,717				
10	"IBM"	200.99	3,111				625,280				
11	"IBM"	200.69	2,335				468,611				
12	"IBM"	200.99	1,042				209,432				
13	"IBM"	198.77	744				147,885				
14	"IBM"	199.20	726				144,619				
15	"IBM"	199.07	842				167,617				
16	"IBM"	198.99	718				142,875				
17	"IBM"	197.70	773				152,822				
18	"IBM"	198.84	496				98,625				
19	"IBM"	198.16	424				84,020				
20	"IBM"	199.15	737				146,774				
21	"IBM"	198.71	664				131,943				
22	"IBM"	196.73	736				144,793				
23			sum				sum				
24			31,959				6,320,423				
25											
26											

=G24/C24

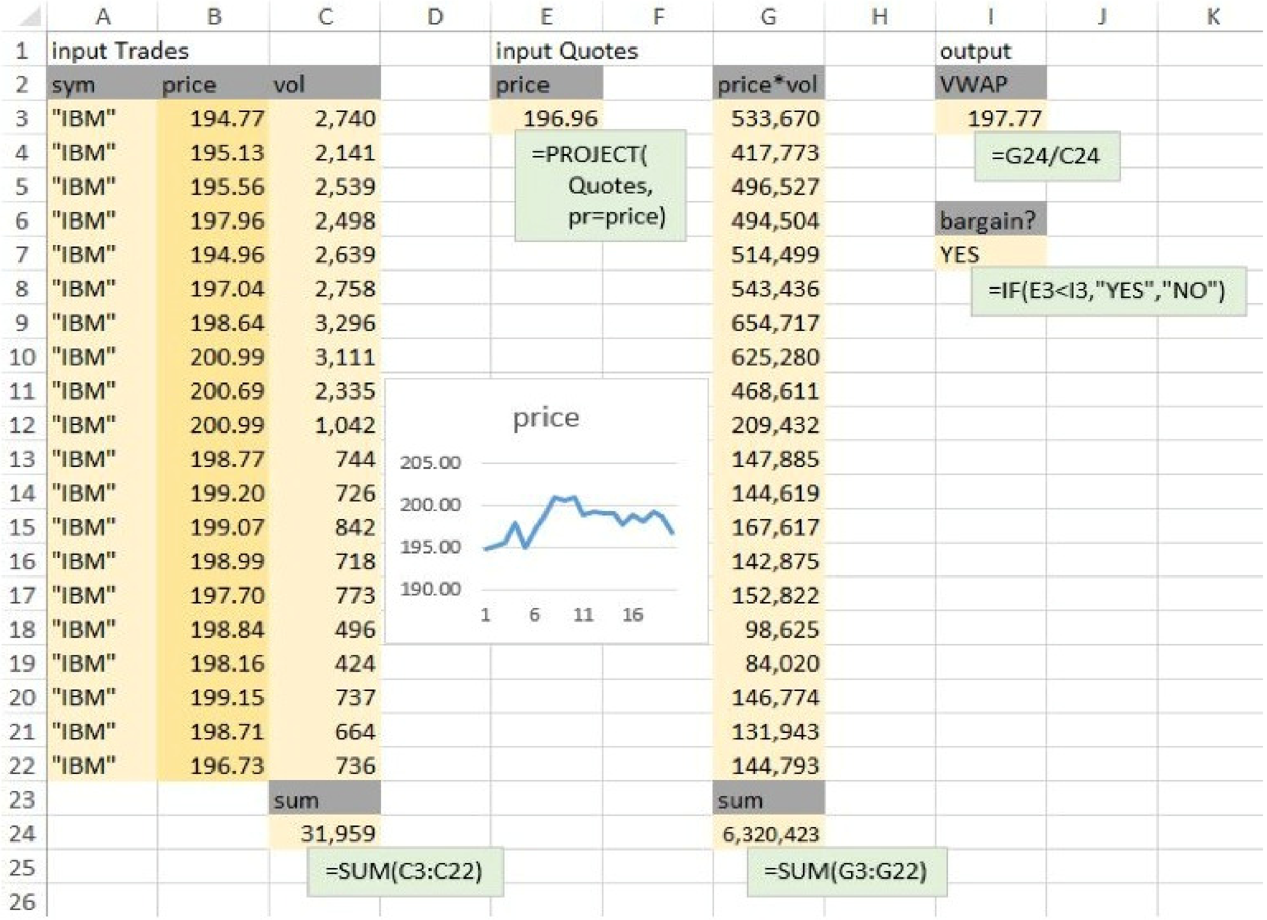
=SUM(C3:C22)

=SUM(G3:G22)

Live

– Use familiar gestures to compute new data

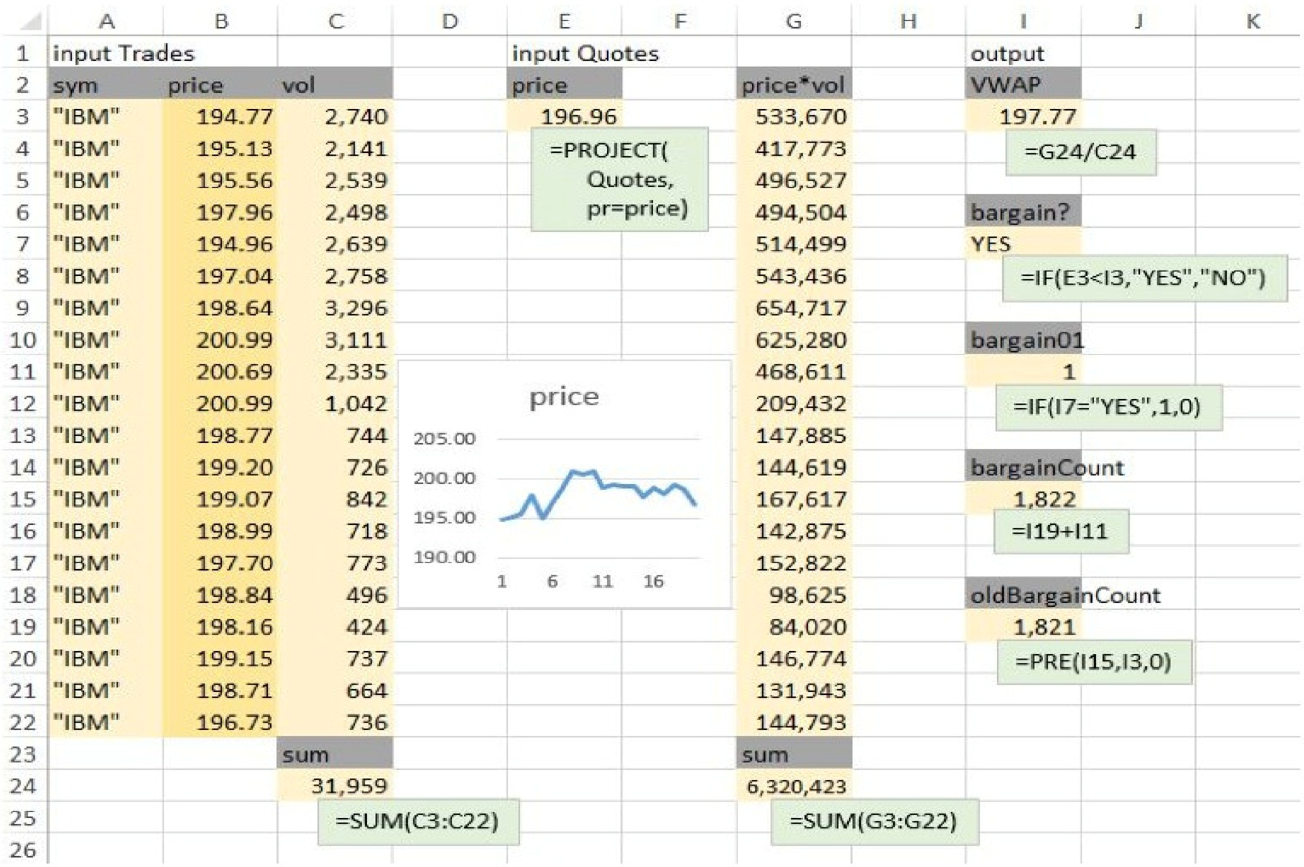
Bargain Calculation in ActiveSheets



	A	B	C	D	E	F	G	H	I	J	K
1	input Trades				input Quotes				output		
2	sym	price	vol		price		price*vol		VWAP		
3	"IBM"	194.77	2,740		196.96		533,670		197.77		
4	"IBM"	195.13	2,141		=PROJECT(Quotes, pr=price)		417,773		=G24/C24		
5	"IBM"	195.56	2,539				496,527				
6	"IBM"	197.96	2,498				494,504		bargain?		
7	"IBM"	194.96	2,639				514,499		YES		
8	"IBM"	197.04	2,758				543,436		=IF(E3<I3,"YES","NO")		
9	"IBM"	198.64	3,296				654,717				
10	"IBM"	200.99	3,111				625,280				
11	"IBM"	200.69	2,335				468,611				
12	"IBM"	200.99	1,042				209,432				
13	"IBM"	198.77	744				147,885				
14	"IBM"	199.20	726				144,619				
15	"IBM"	199.07	842				167,617				
16	"IBM"	198.99	718				142,875				
17	"IBM"	197.70	773				152,822				
18	"IBM"	198.84	496				98,625				
19	"IBM"	198.16	424				84,020				
20	"IBM"	199.15	737				146,774				
21	"IBM"	198.71	664				131,943				
22	"IBM"	196.73	736				144,793				
23			sum				sum				
24			31,959				6,320,423				
25			=SUM(C3:C22)				=SUM(G3:G22)				
26											

- Query language to obtain desired structures
- Data export
- Computation export

Computing with State



	A	B	C	D	E	F	G	H	I	J	K
1	input Trades				input Quotes				output		
2	sym	price	vol		price		price*vol		VWAP		
3	"IBM"	194.77	2,740		196.96		533,670		197.77		
4	"IBM"	195.13	2,141		=PROJECT(		417,773		=G24/C24		
5	"IBM"	195.56	2,539		Quotes,		496,527				
6	"IBM"	197.96	2,498		pr=price)		494,504		bargain?		
7	"IBM"	194.96	2,639				514,499		YES		
8	"IBM"	197.04	2,758				543,436		=IF(E3<I3,"YES","NO")		
9	"IBM"	198.64	3,296				654,717				
10	"IBM"	200.99	3,111				625,280		bargain01		
11	"IBM"	200.69	2,335				468,611		1		
12	"IBM"	200.99	1,042				209,432		=IF(I7="YES",1,0)		
13	"IBM"	198.77	744				147,885				
14	"IBM"	199.20	726				144,619		bargainCount		
15	"IBM"	199.07	842				167,617		1,822		
16	"IBM"	198.99	718				142,875		=I19+I11		
17	"IBM"	197.70	773				152,822				
18	"IBM"	198.84	496				98,625		oldBargainCount		
19	"IBM"	198.16	424				84,020		1,821		
20	"IBM"	199.15	737				146,774		=PRE(I15,I3,0)		
21	"IBM"	198.71	664				131,943				
22	"IBM"	196.73	736				144,793				
23			sum				sum				
24			31,959				6,320,423				
25			=SUM(C3:C22)				=SUM(G3:G22)				
26											

- x := x + 1 becomes   x := pre(x) + 1
- Computing with histories

Programming Model

- Reactive Programming Model
  - Live input streams are clocks into the spreadsheet
  - Cells are registers that get updated at each tick
  - Simple control structure:

```
while(true){
    await(tick);
    calculate-spreadsheet();
}
```

- Benefits
  - Ease-of-use
    - No need to think about control (no sequencing, no loops)
    - Data manipulated directly
  - Guarantees
    - Determinism
    - Bounded computation and memory usage at each tick
  - Live Programming
  - Expressive for a range of stream applications.

Formal Semantics: Spreadsheet Calculus

- Motivation
  - When should cells be updated? With what value?
    - pre(I5,I3,0), project(quotes, pr=price)
- Core Calculus
  - Tick: strictly increasing series of non-negative numbers, captures logical time
    1,2,3,4,5,…
  - Feed: map from tick to values (corresponding to a single attribute of a stream)

    $1 \rightarrow \text{Red},\ 2 \rightarrow \text{Blue},\ 3 \rightarrow \text{Yellow}$
  - Server: collection of feeds
  - Client: collection of cells, consisting of a name and a formula

Core Calculus

- Formulas

$$f \ ::== \ \phi \mid op(c_1, \cdots, c_n) \mid c_0 \ @ \ c_1 \mid latch(c_0,c_1)$$

  - – c_0 @ c_1 ticks when c_1 does and evaluates to true
    - sample feed according to a Boolean condition
  - – latch(c_0,c_1) ticks when c_1 does and returns the value of c_0 at the previous tick of c_1
    - Mechanism to access a past value

- Example

time ↓

A1	A2: A1+1	A3: odd(A1)	A1@A3	latch(A1,A1)
0	1	false		0
1	2	true	1	0
2	3	false		1
3	4	true	3	2
4	5	false		3
5	6	true	5	4

Well-Formedness

- Immediate dependencies

$$\text{deps}(c) = \begin{cases} \varnothing & \text{if } c \equiv \phi \\ \{c_1, \cdots, c_n\} & \text{if } c \equiv \text{op}(c_1, \cdots, c_n) \\ \{c_0, c_1\} & \text{if } c \equiv c_0 \,@\, c_1 \\ \{c_1\} & \text{if } c \equiv \text{latch}(c_0,c_1) \end{cases}$$

- Well-Formedness
  - A client is well-formed iff the directed graph of immediate dependencies is acyclic, where vertices are cell names, and edges indicate immediate dependencies
- Example: X := X + 1
  - Incorrect:

    ```
    A1: latch(A2,A2)
    A2: A1 + 1
    ```

  - Correct:

    ```
    A1: latch(A2,A3)
    A2: A1 + 1
    A3: server feed
    ```

Extensions of the Core Calculus

- Live Calculus
  - Feed of formulas for each cell
  - A tick of a cell is the concatenation of the ticks of its successive formulas over time
  - A latch does not access values that predate the formula that contains the latch
- Stream Calculus
  - Enrich core calculus with richer streams and formulas
  - Reduction to core calculus
- Query language
  - Provides a way to populate a range of cells at once with relational operators
  - Reduction to stream calculus

Related Work

- Spreadsheets as a programming platform
  - Haxcel
  - Programming sensor networks
  - StreamBase Excel adapter
  - Cloudscale
- Programming models for streaming
  - Lustre
  - StreamIt
  - Lime
  - SPL
  - Spark Streaming
  - SQL-based languages: CQL, Microsoft StreamInsight
- Formal models
  - Synchronous programming languages

Summary and Future Work

- Summary
  - Spreadsheet as a programming platform for stream processing
  - Easy-to-use interface, familiar to spreadsheet users
  - Strong guarantees: determinism, bounded computation and memory usage
  - Variety of case studies to illustrate expressivity
- Future Work
  - Online spreadsheet client
  - Code synthesis for higher performance

Images in this presentation from http://www.getfreeimage.com/