Think? Compute! See!!
End User Programming for Thinkers

Dave Thomas
YOW! Conference, Bedarra Research Labs.
Queensland Uni of Technology and Carleton University
Ottawa, Canada

Thinker Persona (Computational Scientists)

- Domain Experts in Science, Engineering, Business, Arts for whom high performance computation is now an essential tool discovery and design.
- Model based exploratory programming with a strong emphasis on hypothesis formulation and visualization.
- Minimal training in CS/SE and associated languages, tools and practices. Reject software industry Agile + OO
- Willing to use any combination of tools to get their work done. Eg, FORTRAN, C++ library codes, Python, R, Matlab ...
- **Data Scientists** = Big Data + HPC (cloud/clusters/grids)
Thinker Compute Alternatives

1. Large Distributed Network of Small Machines
   - (2G - 4 cores ) * 100s cpus and/or gpus
   - simple disks with GFS + Map Reduce ...

2. Small Clusters of Large Machines
   - (128G - 1TB + 4 - many cores) * 10s of cpus/gpus
   - high performance RAID with hardware compression
   - column stores
   - SQL + functions

Thinker Data Environment

- Data Volumes in 10s of TB to 100 PBs
  - Memory, SSD and RAID Disk Array Based Column Stores, Distributed Data Sets and Databases
  - Software and Hardware data compression; encryption
  - Streams/Samples for huge data sets
- Examples
  - Main Memory DB; NoSQL DB (Triple Stores); Column Stores, Vector DB; Streaming DB; GraphDB...
  - MemCache, Oracle Coherence; RIAK, Mongo, Couch DB, Amazon Simple DB; Aster, Greenplum, Vertica, Neo4J

Unique Data Properties & Operations

- Time (Millenniums to Nanoseconds) and Timespans
- Missing Data, Out of Range Data, Uncertainty (45 % likely, highly unlikely)
- Operations over Huge Tables, Dictionaries, Lists, Arrays
- Visualization of Big Data
- Charts and Plots; Trees and Graphs ;Maps in 2 & 3 D, GIS, Human Body, Heat Maps...
- Examples - InfoViz, Graphviz, R ggplot, Tableau...,Processing (Processing.org, Processing.js)
Data Intensive Computing

Massive storage and processing enables living in a click/tic stream processing of raw un-normalized data - RFIDx, Clicks, Tics, Customer interactions, Sensor Events ...

Smart Algorithms which stream over data sets - Customer Life Time Value, Recommendation Engines, Web Analytics, Real-time Financials, Network and Sensor Monitoring, Complex Event Processing

All roads lead to some form of Functional CRUD

- Applied Functional Programming (aka Super CRUD)
- SQL + Functions + Streams – e.g. Greenplum ...
- NoSQL Databases – Dictionaries on Steroids (Big Table, CouchDB...)
- Map Reduce, Comprehensions
- Hybrid JVM, CLR/LINQ functional languages F#, Scala, Clojure
- Vector Functional Programming
- Graph Databases
- GPUs ...

End User Programming Models

- Textual or Visual(boxes and arrows) DSL
- Programming by Example (Abstract and Concrete)
- Programming By Demonstration
- Tables (Spreadsheets, Decision tables, State Tables)
- Forms and CRUD/SQL
- Rule/Deductive Programming
- Mathematical Programming
- Examples - NumPy, R, MatLab; LabView, Prograph, Google App Inventor, Yahoo Pipes; ORE, ORE.BBA; Trinker, Stage Cast, MSSRy; State Charts;
  4GL - Concert, Mappr, PowerHouse, PowerBuilder, Delphi, OutSystems, SQL, Expert Systems, Jvules; Agent Sheets, Datalog, Mathematical, R ...

But you need to be a FP wizard to live here?!

High Barrier Languages

Want to Be Clousrets

Math!?

Professor Lambda

Just eat your monads and defer all computations!

Needs Solution

Just want my application!

Planet FP
End User Programming for Thinkers

Democratize High Performance Domain Oriented Programming

- Very productive for specific applications
- Scaling problematic forcing often users to migrate to a “real” programming language
  - Limited Interoperability with outside world
  - 32 bit (limited data size) and single process (limited concurrency)
  - Restricted programming model (limited data types and operations e.g. SQL, OLAP, Spreadsheets)

*Is EUP only for Wimps? hence HPC only for Wizards?!

Big Data EUP Examples
- Apache Cascading, Pig, Hive
- Ripe for R
- Google Sawzall

Collaborative Analytics – A Thinker Example

- Analytic team consisting of cross jurisdictional domain experts assembled on demand for a critical situation
- Analysts need to be able work across big data in clouds to embedded sensors
- Analysts must be able to work visually as well as texturally to rapidly explore alternatives
- Fine grained version management, security controlled sharing and annotation of all assets (cells, images,...) Workflow versioning for big data computations (enables redo)
What Virtual Execution Environment (VEE)?

What programming model and runtime is well suited to Interactive model based computation?

Top Down VEE Design Choices

1. A few elegant and simple abstractions
2. A dynamic object model and garbage collector
3. Everything is an object (list, set, ...)
   - ! the first N < 5 implementations will suck in space and time
   - ! interop with native HW will trail HW
   - ! implementers will spend decades trying to make elegant => fast
4. Language is extended by libraries in the same language
   - ! the libraries will be bloated and of variable quality, with changing APIs ...

Bottom Up VEE Design Choices

1. Needs to be fast!
2. Needs to be small (compact)
3. Needs basic safety
4. Needs to support massive data
5. Needs scalable concurrency
6. It will be a challenge to design a normal developer language (i.e. the GPU problem)

Vector Functional VEE
Why Functional Vector VMs Kick Object VMs

- Array VMs vs. OVMs
  - No need for boxing and unboxing! ... Simpler GC
  - Support for all native machine types
  - Virtual machine is smaller ... can easily be held in instruction and data caches
  - Values are shared until modified
- Arrays are Column Stores => Table is a set of columns
  - Reduces the impedance between Objects and Records
  - Vectors are thinly serialized
  - Vector operations stream data through caches
- Array libraries use efficient algorithms code at machine level
- Arrays take less space than object collections
- Data parallelism is easily implemented

APL – The first array language

Think in Collections (Arrays in APL, later any) J, NIAL, K...
No Stinking Loops, Ultra Concise Programs
Operator (later function composition)

I want this on my mobile and iPad!

Transition to High Barrier Languages

And Now For Something Completely Different!

Leap of faith!

Change Curve

This will great!

Aha!

This Sucks

Kicks Butt!
**CARE Analyst and Expert Programming Models**

- 50:1 of ratio of analysts to expert developers
- Experts surface new functionality to analysts as DSL library extensions

**Analyst Application Programming Model**

Wide Spectrum DSLs (SQL, Sheets & Tables, Boxes and Arrows, ...)
Leverage existing standards and user models

**Expert Programming Model**

Wide Spectrum Functional Vector Language
Interoperable with current technologies: ODBC, Java, C#, C++, Web

**CARE Core Platform**

- Columnar storage
- Interoperable with current technologies: ODBC, Java, C#, C++, Web
- Virtual Execution Environment (VEE) – High Performance Vector Functional Runtime

**CARE - An Exploratory Collaborative Analytics Environment**

- Collaborative Interactive Development Environment (IDE)
- Analytic Tools (AT)
- Library and Runtime Environment (VEE)
- Big Data
- Spreadsheet
- Virtual Execution Environment (VEE) – High Performance Vector Functional Runtime

**Analyst Visual IDE Concept Maps and Sheets**

- Analyst Visual IDE
- Concept Maps and Sheets