Vrije
Universiteit
Brussel

Software
Languages.Lab

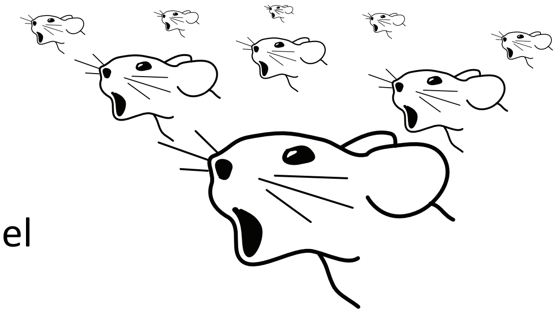
Fork me on GitHub

Sly and the RoarVM

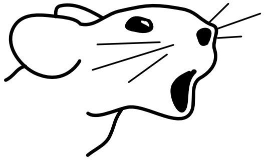
Exploring the Manycore Future of Programming

Renaissance Project

http://soft.vub.ac.be/~smarr/renaissance/



Stefan Marr
Software Languages Lab, Vrije Universiteit Brussel
Smalltalk Dev Room, FOSDEM, 2012-02-05



Agenda

- Parallel Smalltalk and the RoarVM
  - `[self demo] fork`
  - Project
- Sly and the Renaissance Project
  - Demo
  - Context
  - Language and Use-Cases

Disclaimer: It wasn’t me!

IBM Research

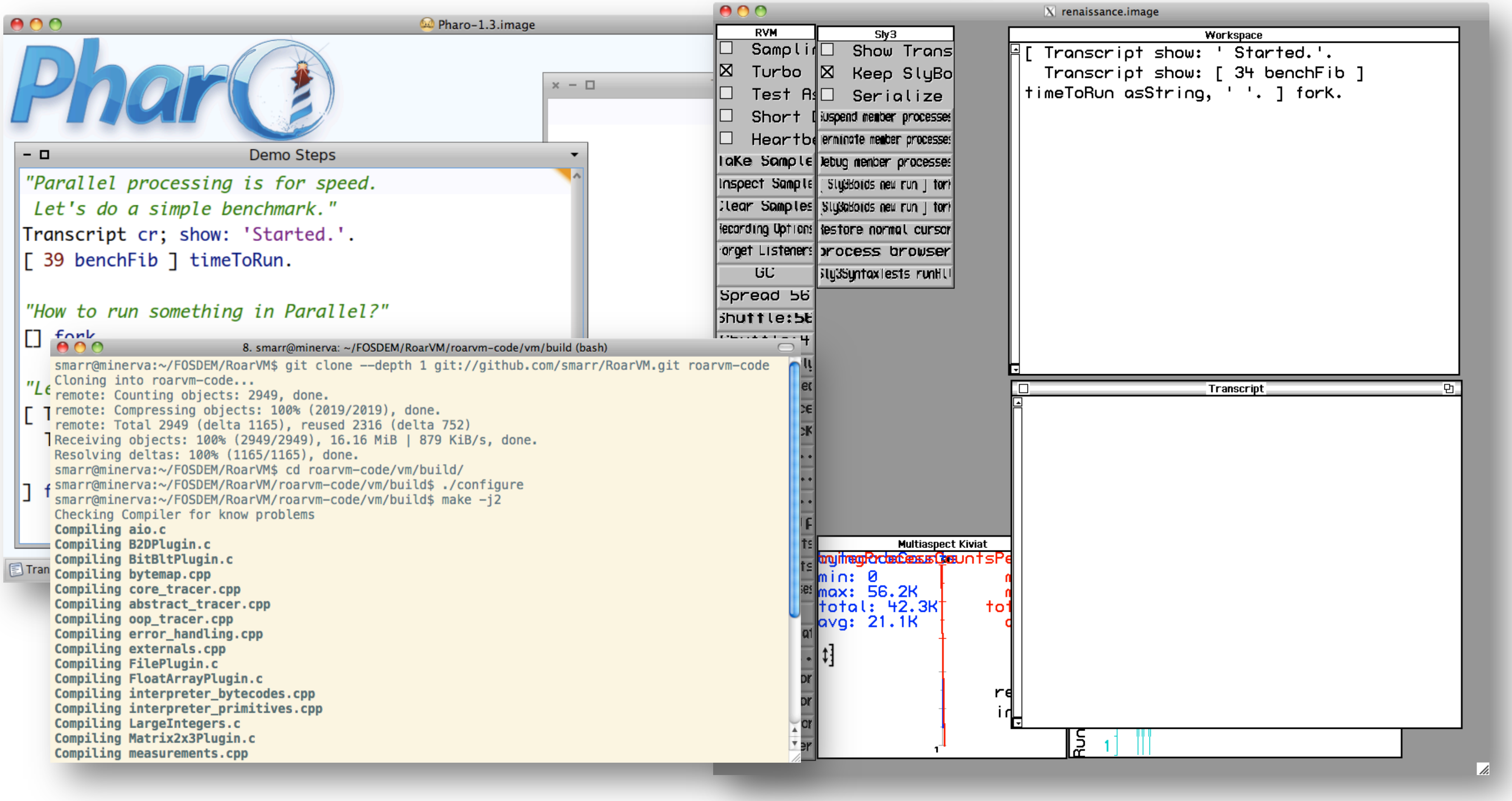
- **David Ungar**
- Sam S. Adams
- Doug Kimelman

Portland State University

- Andrew P. Black
- Max OrHai

Vrije Universiteit Brussel: Software Languages Lab

- Pablo Inostroza Valdera
- Stefan Marr
- Theo D'Hondt



Demo Steps:

```
"Parallel processing is for speed.
 Let's do a simple benchmark."
Transcript cr; show: 'Started'.
[ 39 benchFib ] timeToRun.

"How to run something in Parallel?"
[] fork
```

8. smarr@minerva: ~/FOSDEM/RoarVM/roarvm-code/vm/build (bash)

```
smarr@minerva:~/FOSDEM/RoarVM$ git clone --depth 1 git://github.com/smarr/RoarVM.git roarvm-code
Cloning into roarvm-code...
remote: Counting objects: 2949, done.
remote: Compressing objects: 100% (2019/2019), done.
remote: Total 2949 (delta 1165), reused 2316 (delta 752)
Receiving objects: 100% (2949/2949), 16.16 MiB | 879 KiB/s, done.
Resolving deltas: 100% (1165/1165), done.
smarr@minerva:~/FOSDEM/RoarVM$ cd roarvm-code/vm/build/
smarr@minerva:~/FOSDEM/RoarVM/roarvm-code/vm/build$ ./configure
smarr@minerva:~/FOSDEM/RoarVM/roarvm-code/vm/build$ make -j2
Checking Compiler for know problems
Compiling aio.c
Compiling B2DPlugin.c
Compiling BitBltPlugin.c
Compiling bytemap.cpp
Compiling core_tracer.cpp
Compiling abstract_tracer.cpp
Compiling oop_tracer.cpp
Compiling error_handling.cpp
Compiling externals.cpp
Compiling FilePlugin.c
Compiling FloatArrayPlugin.c
Compiling interpreter_bytecodes.cpp
Compiling interpreter_primitives.cpp
Compiling LargeIntegers.c
Compiling Matrix2x3Plugin.c
Compiling measurements.cpp
```

Workspace:

```
[ Transcript show: ' Started.'.
  Transcript show: [ 34 benchFib ]
timeToRun asString, ' '. ] fork.
```

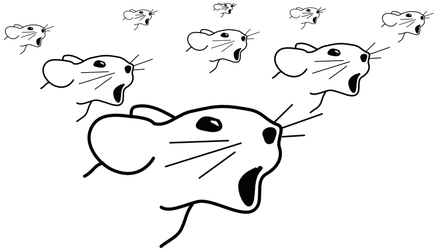
RoarVM and Parallel Smalltalk?

Get the RoarVM

Smalltalk, Concurrency, and Parallelism

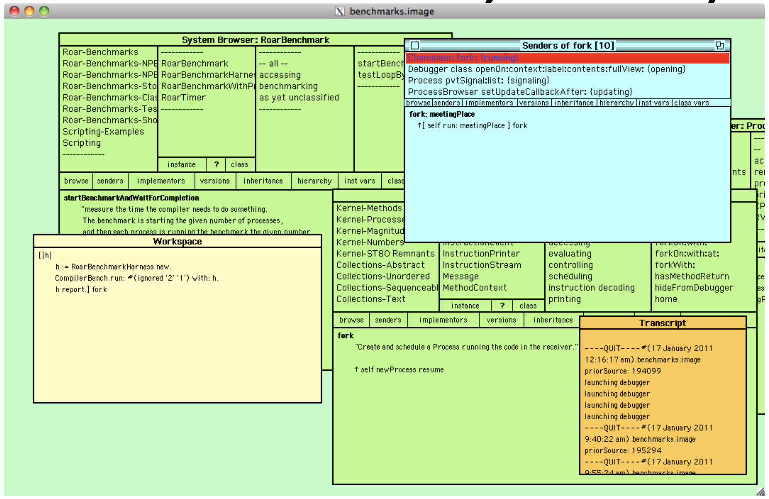
RoarVM Smalltalk

- `Process` class additions
  - `yieldIfFewerCoresThan: anInt`
  - `coreMask`, `coreMask: anInt`
    - `onlyOnCore: anInt`
  - `hostCore`
  - `thisProcess` instead of ~~`activeProcess`~~
- Uses standard Smalltalk facilities
  - `[self do] fork`
  - Semaphore class



RoarVM Status

- VM: rewritten Squeak, 20k SLOC C++
- MVC images run fine
- Squeak/Pharo: multicore instabilities
- Runs on: OSX, Linux, iOS, Tilera, up to 59 cores



Roar-Benchmarks
Roar-Benchmarks-NPE
Roar-Benchmarks-NPE
Roar-Benchmarks-Sto
Roar-Benchmarks-Clas
Roar-Benchmarks-Tes
Roar-Benchmarks-Sho
Scripting-Examples
Scripting

RoarBenchmark
RoarBenchmarkHarnes
RoarBenchmarkWithPr
RoarTimer

-- all --
accessing
benchmarking
as yet unclassified

----
startBench
testLoopBy

```
startBenchmarkAndWaitForCompletion
    "measure the time the compiler needs to do something.
    The benchmark is starting the given number of processes,
    and then each process is running the benchmark the given number
```

Chameleon fork: {running}
Debugger class openOn:context:label:contents:fullView: {opening}
Process pvtSignal:list: {signaling}
ProcessBrowser setUpdateCallbackAfter: {updating}

```
fork: meetingPlace
    ↑[ self run: meetingPlace ] fork
```

```
[|h|
    h := RoarBenchmarkHarness new.
    CompilerBench run: #(ignored '2' '1') with: h.
    h report.] fork
```

Kernel-Methods
Kernel-Processe
Kernel-Magnitud
Kernel-Numbers
Kernel-ST80 Remnants
Collections-Abstract
Collections-Unordered
Collections-Sequenceabl
Collections-Text

InstructionPrinter
InstructionStream
Message
MethodContext

evaluating
controlling
scheduling
instruction decoding
printing

forkOn:with:at:
forkWith:
hasMethodReturn
hideFromDebugger
home

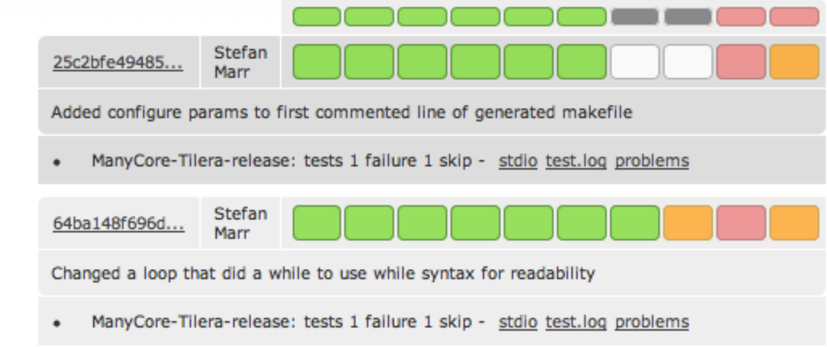
```
fork
    "Create and schedule a Process running the code in the receiver."

    ↑ self newProcess resume
```

```
----QUIT----#(17 January 2011
12:16:17 am) benchmarks.image
priorSource: 194099
launching debugger
launching debugger
launching debugger
launching debugger
----QUIT----#(17 January 2011
9:40:22 am) benchmarks.image
priorSource: 195294
----QUIT----#(17 January 2011
9:55:24 am) benchmarks.image
```

Project Infrastructure

- Buildbot: Automate compilation, testing
  - Linux, OS X, Tilera
- Performance Tracking
  - Codespeed
  - ReBench
  - SMark
- http://github.com/smarr/RoarVM



25c2bfe49485... Stefan Marr

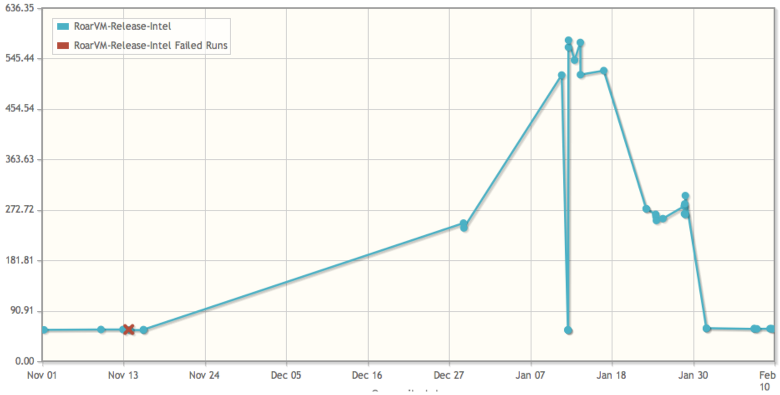
Added configure params to first commented line of generated makefile

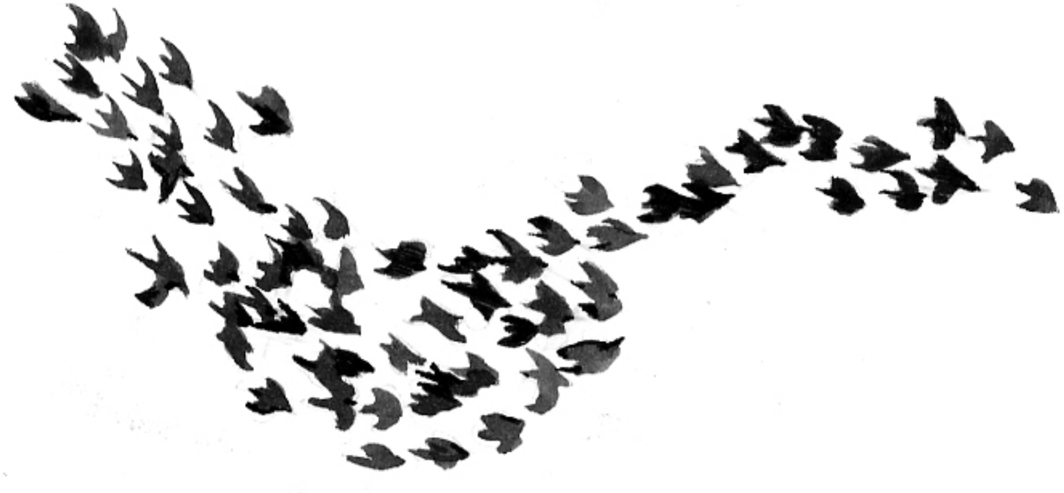
- ManyCore-Tilera-release: tests 1 failure 1 skip - stdio test.log problems

64ba148f696d... Stefan Marr

Changed a loop that did a while to use while syntax for readability

- ManyCore-Tilera-release: tests 1 failure 1 skip - stdio test.log problems



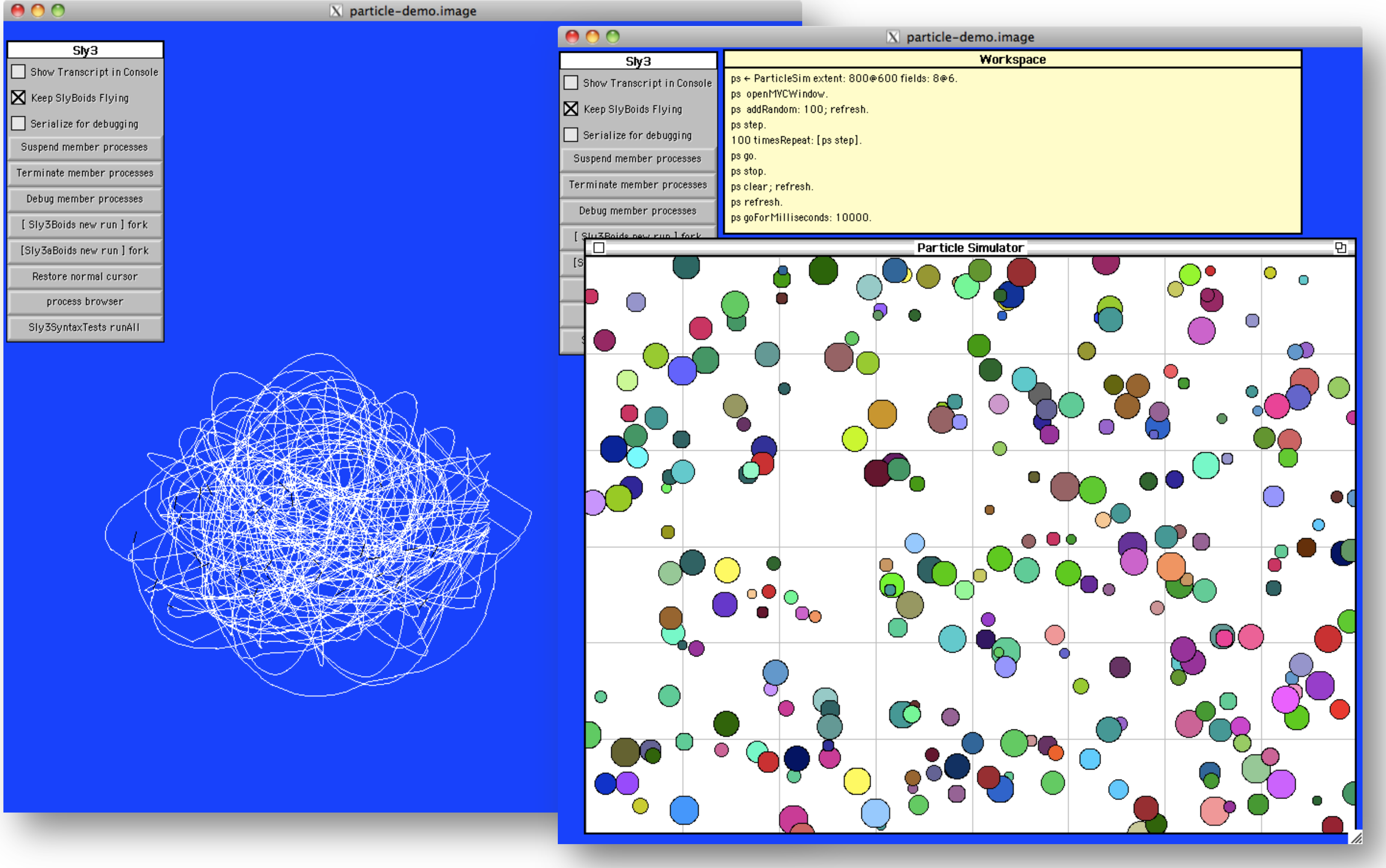


Avoid the Concurrency Trap by Embracing Non-Determinism

SLY AND
THE RENAISSANCE PROJECT

Renaissance Project

- Manycore: > 100 cores
  + non-uniform memory access
- Problem
  - Parallel systems are nondeterministic
  - Communication is expensive
  - Synchronization == bottleneck
- The vision
  - Embrace nondeterminism
  - Harness emergence
  - Confidence → Delay



☐ Show Transcript in Console

☒ Keep SlyBoids Flying

☐ Serialize for debugging

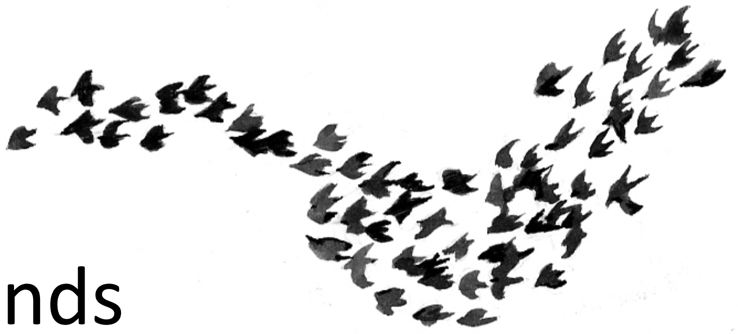
```
ps ← ParticleSim extent: 800@600 fields: 8@6.
ps  openMYCWindow.
ps  addRandom: 100; refresh.
ps step.
100 timesRepeat: [ps step].
ps go.
ps stop.
ps clear; refresh.
ps refresh.
ps goForMilliseconds: 10000.
```

Non-deterministic Boids and Particle Sorting

Flocking behavior of birds and particle bouncing as local-only synchronization strategies for scalability

Sly

- Ensembles: collections representing a whole, multi-part entities
  - e.g. a flock of birds



- Adverbs: modifiers to operands
  - `individualLY`, `serialLY`, `randomLY: n`
- Gerunds: reduction semantics
  - `averagING`, `selectINGpredicate`, `ensemblING`

Examples of Boids

- Every boid is its own thread, no synch.

```
computeCentroid
    ^ self flock boids averagINGserialLYposition

computeNeighbors
    ^ self flock boids selectINGisNear: self

matchVelocityWithNeighbors
    ^ self neighbors averagINGvelocity / 8.0
```

More details: http://soft.vub.ac.be/~smarr/renaissance/sly3-overview/

Possible Real Life Application

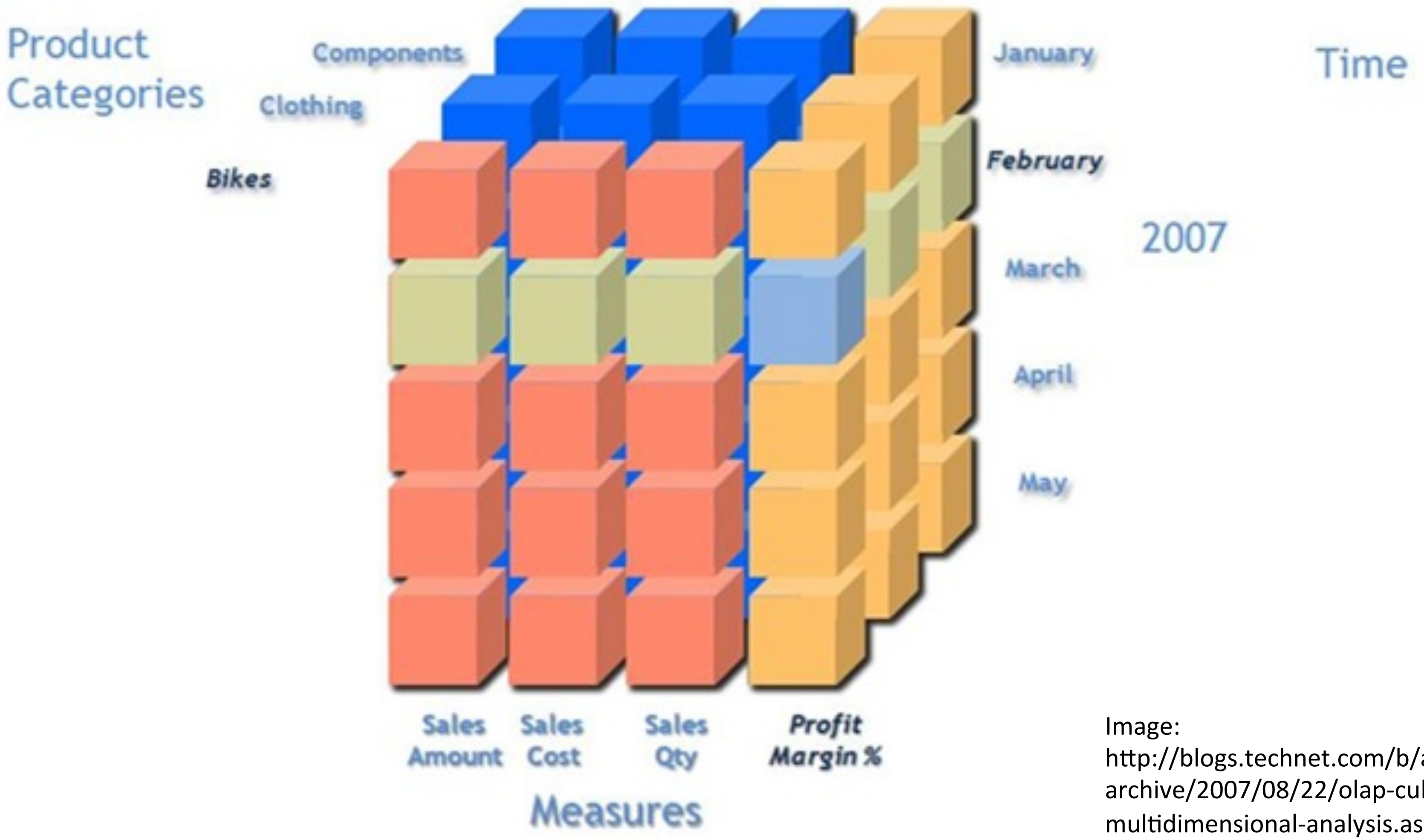
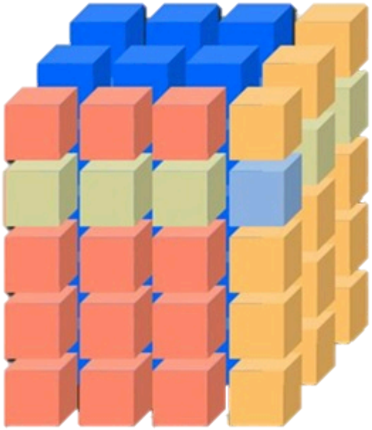
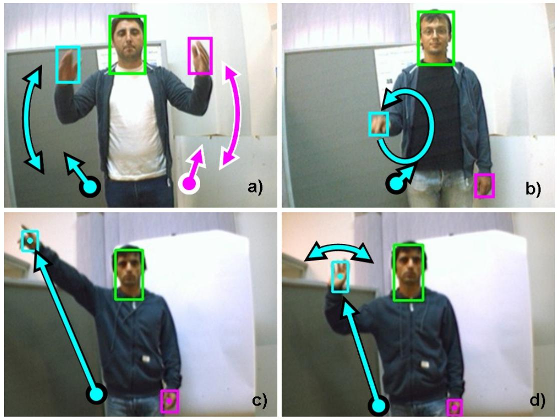


Image: http://blogs.technet.com/b/andrew/archive/2007/08/22/olap-cubes-and-multidimensional-analysis.aspx

Possible Real Life Application



- Online analytical processing (OLAP)
  - Business intelligence, data mining
  - Query multidimensional data sets
  - Calculated data dimensions



- Gesture recognition
  - Many rules, low-precision input
- Race-and-repair: how to bound error?

CONCLUSION

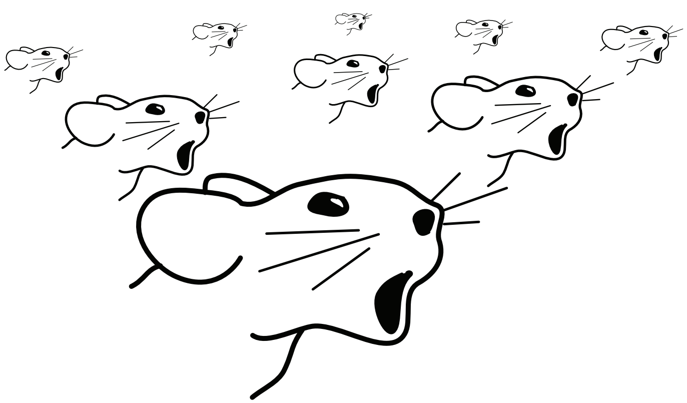
Conclusion

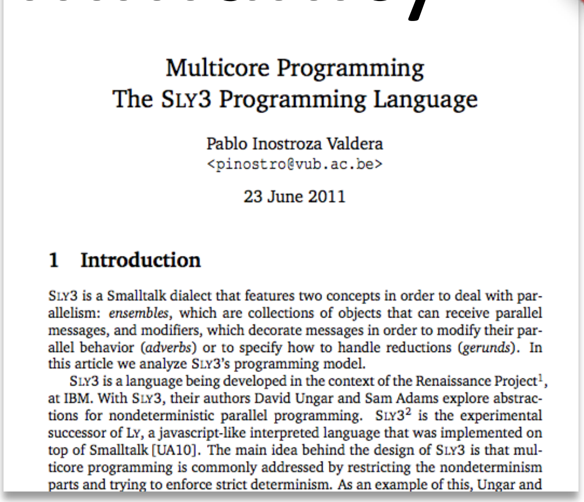
- Sly: map/reduce-like programming model
  - Based on ensembles, adverbs and gerunds

- RoarVM
  - Research artifact, stable, used daily
  - No optimization of sequential performance
  - Tested up to 59cores
  - Shows weak scalability

Fork me on GitHub

Project, Sly Intro, Community





Multicore Programming
The Sly3 Programming Language

Pablo Inostroza Valdera
<pinostro@vub.ac.be>

23 June 2011

1 Introduction

Sly3 is a Smalltalk dialect that features two concepts in order to deal with parallelism: *ensembles*, which are collections of objects that can receive parallel messages, and modifiers, which decorate messages in order to modify their parallel behavior (*adverbs*) or to specify how to handle reductions (*gerunds*). In this article we analyze Sly3's programming model.

Sly3 is a language being developed in the context of the Renaissance Project[1], at IBM. With Sly3, their authors David Ungar and Sam Adams explore abstractions for nondeterministic parallel programming. Sly3[2] is the experimental successor of Ly, a javascript-like interpreted language that was implemented on top of Smalltalk [UA10]. The main idea behind the design of Sly3 is that multicore programming is commonly addressed by restricting the nondeterminism parts and trying to enforce strict determinism. As an example of this, Ungar and

http://soft.vub.ac.be/~smarr/renaissance/

http://soft.vub.ac.be/~smarr/renaissance/sly3-overview/

http://github.com/smarr/RoarVM

@roarvm

github SOCIAL CODING

twitter

Papers

[1] J. Pallas and D. Ungar. Multiprocessor Smalltalk: A Case Study of a Multiprocessor-based Programming Environment. In PLDI ’88, p. 268–277. ACM, 1988.

[2] D. Ungar and S. Adams. Hosting an Object Heap on Manycore Hardware: An Exploration. In DLS’09, p. 99-110. ACM, 2009.

[3] D. Ungar, D. Kimelman, and S. Adams. Inconsistency Robustness for Scalability in Interactive Concurrent-Update In-Memory MOLAP Cubes. In Inconsistency Robustness Symposium 2011.