Scilab 6
What's new?

July 1st, 2009

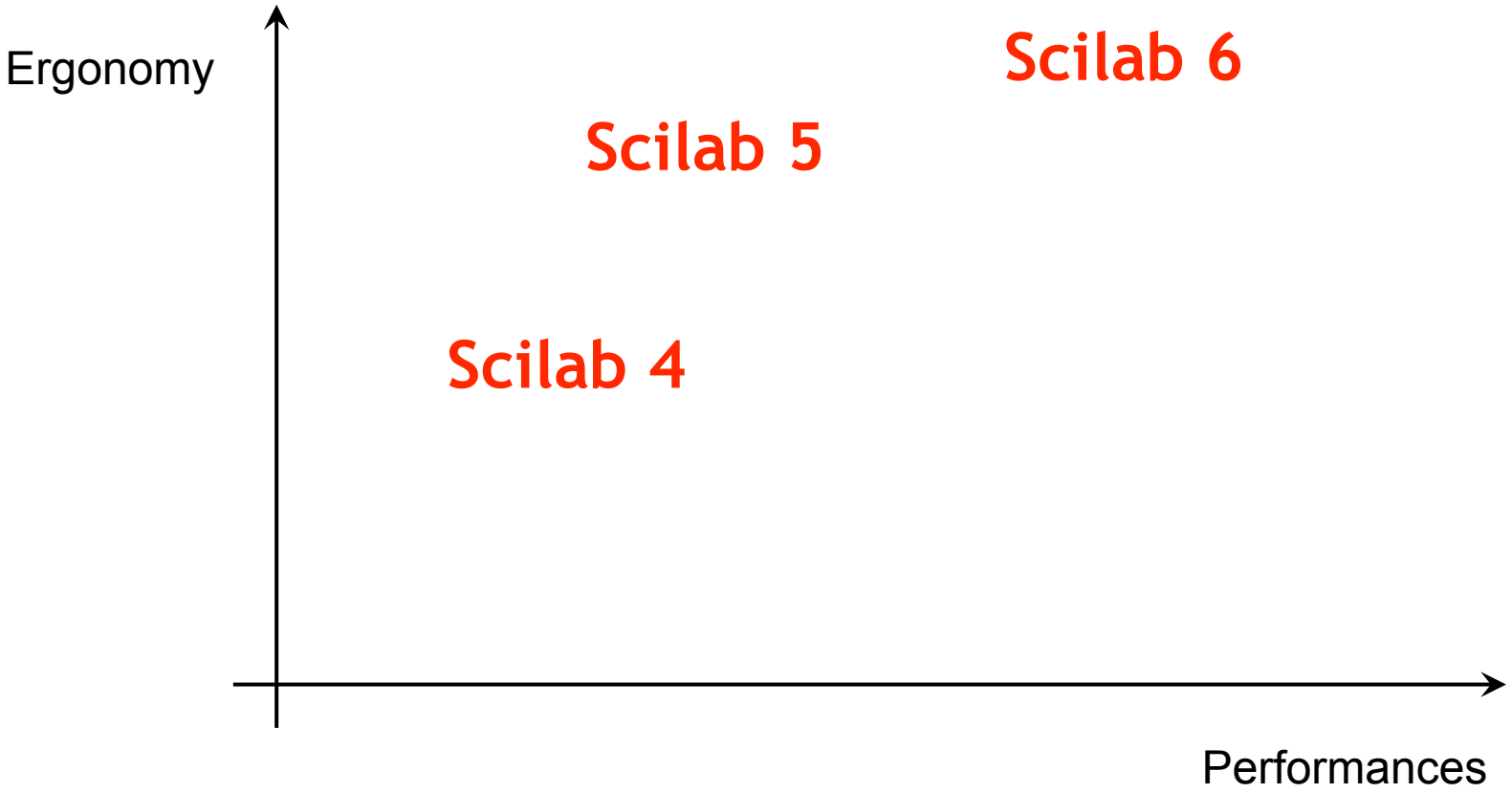
A new kernel for Scilab

Bruno JOFRET
Scilab Software Architect


Scilab

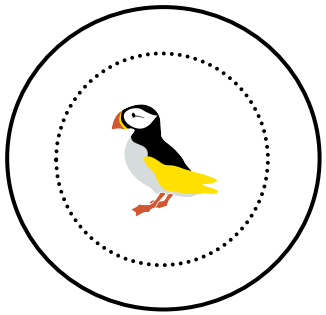
Scilab progress

Overview



Scilab 6 – A brand new Kernel

What's Scilab Kernel?



- Lexer & Parser
- Memory Manager
- Interpreter
- Functions

These are the key elements of Scilab.
The engine enables any mathematical/algorithmic capability to run.

Lexer/Parser

Before being a software, Scilab is a language

Lexer/Parser - Evolutions

Before being a software, Scilab is a language

- Evolutions:
  - Lexical and grammatical analysis before execution
  - Error catch / Better debug informations
  - Abstract Syntax Tree
- Opportunities:
  - Scilab grammar
  - Language evolutions:
    - function … end[function]
    - /* comment region */
  - Syntax sugar

Memory manager

Scilab has to be able to manage huge sets of data

- Old static allocation
- Dynamic allocation
  - Scilab 4/5: Up to 2GB Memory limited by Scilab stack
  - Scilab 6: No more internal limitation (OS)

Memory manager - Evolutions

Scilab has to be able to manage huge sets of data

- Better storage capabilities
- HPC
- No more variable number/name limitation
- Dedicated module:
  - Garbage Collection
  - Various Implementations (crashpad, ...)
- Namespace
- Variable Protection

Memory manager - Results

Scilab 5 static memory

```
         ___________________________________________
                        scilab-5.1.1

                   Consortium Scilab (DIGITEO)
                  Copyright (c) 1989-2009 (INRIA)
                  Copyright (c) 1989-2007 (ENPC)
         ___________________________________________


Startup execution:
  loading initial environment

-->M = [ 1 2 3 4 ; 1 2 3 4 ; 1 2 3 4 ; 1 2 3 4]
 M  =

    1.    2.    3.    4.
    1.    2.    3.    4.
    1.    2.    3.    4.
    1.    2.    3.    4.

-->for i = 1:8; M = [M M ; M M]; end

-->M = [M M ; M M];
                !--error 17
: stack size exceeded (Use stacksize function to increase it).


-->size(M)
 ans  =

    1024.    1024.

-->
```


Scilab

Memory manager - Results

Scilab 6 dynamic memory allocation

```
-*- Yet Another Scilab Project -*-


-->M = [ 1 2 3 4 ; 1 2 3 4 ; 1 2 3 4 ; 1 2 3 4]
M =

  1    2    3    4
  1    2    3    4
  1    2    3    4
  1    2    3    4

-->for i = 1:8; M = [M M ; M M]; end
-->M = [M M ; M M];
-->M = [M M ; M M];
-->M = [M M ; M M];
-->__echo(M)
=============================================
====    ECHO DEBUG FUNCTION    ====
=============================================
 --> Number of input argument = 1
=============================================
--> ARG[1]
    --> Type : 2
    --> Rows : 8192
    --> Cols : 8192
    --> Size : 67108864
=============================================
====    ECHO DEBUG FUNCTION    ====
=============================================
-->
```

Memory manager - Results

Scilab 5 variable name limitations

```
                scilab-5.1.1

            Consortium Scilab (DIGITEO)
          Copyright (c) 1989-2009 (INRIA)
          Copyright (c) 1989-2007 (ENPC)


Startup execution:
  loading initial environment

-->abcdefghijklmnopqrstuvwxyz = 2
Warning :
The identifier : abcdefghijklmnopqrstuvwxyz
 has been truncated to: abcdefghijklmnopqrstuvwx.

 abcdefghijklmnopqrstuvwx  =

    2.

-->
```

Memory manager - Results

Scilab 6 unlimited variable name

```
-*- Yet Another Scilab Project -*-


-->abcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyz = 2
abcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyz =

  2

-->abcdefghijklmnopqrstuvwxyz = 3
abcdefghijklmnopqrstuvwxyz =

  3

-->
-->
-->
-->
-->
-->
-->
```


Scilab

Execution

Scilab manages an abstract syntax structure

- Parallelism detection
- Syntax Tree Manipulation

HPC - Show me what you've got!

Code Sample

```
M = rand(2048, 2048);
N = rand(2048, 2048);
P = fft(M);
Q = fft(N);
```

Scilab 5 Execution

```
M = rand(2048, 2048);
N = rand(2048, 2048);
P = fft(M);
Q = fft(N);
```

- Stack size exceeded
- No parallel execution

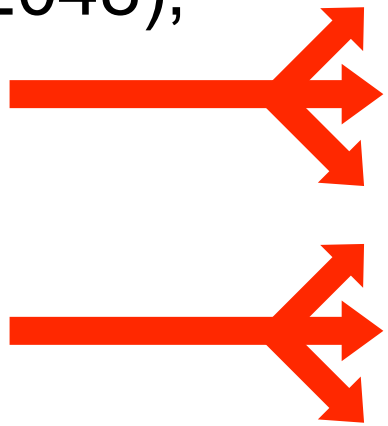
Scilab

Scilab 6 Execution

Algorithm parallelization

```
M = rand(2048, 2048);
N = rand(2048, 2048);
P = fft(M);

Q = fft(N);
```



- Multithread
- SoC / GPU
- …

Scilab

Scilab 6 Execution

User parallelization

```
M = rand(2048, 2048);
N = rand(2048, 2048);
P = fft(M);
Q = fft(N);
```

Scilab 6 Execution

User parallelization

```
blockStart();
M = rand(2048, 2048);
P = fft(M);
blockEnd();
```

```
blockStart();
N = rand(2048, 2048);
Q = fft(N);
blockEnd();
```


Scilab

Scilab 6 Execution

User parallelization

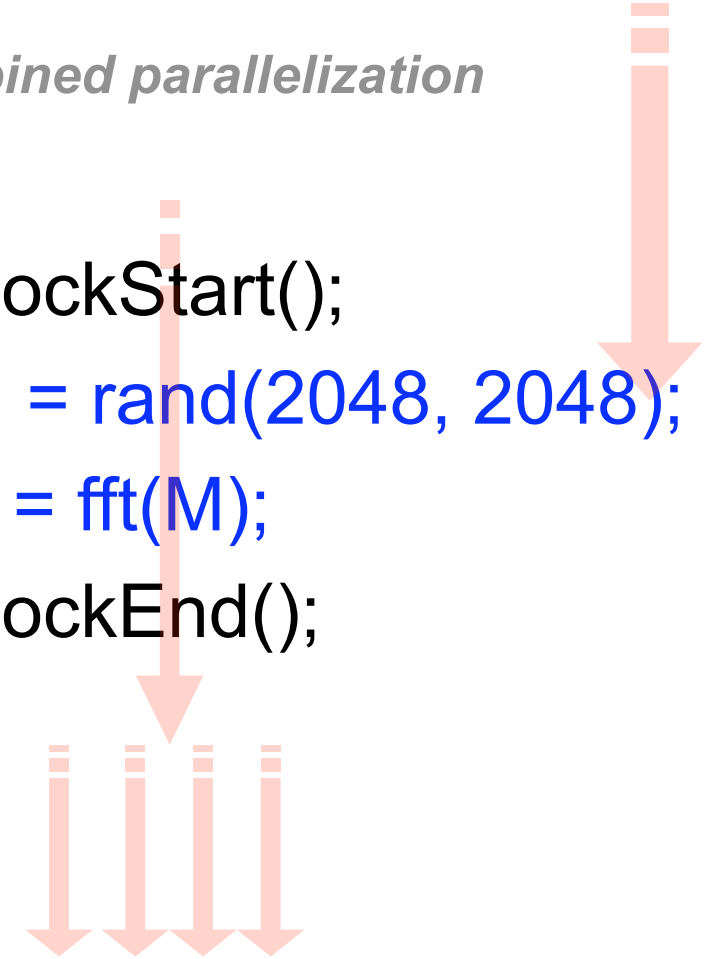
```
blockStart();
M = rand(2048, 2048);
P = fft(M);
blockEnd();
blockStart();
N = rand(2048, 2048);
Q = fft(N);
blockEnd();
```

- OpenMP
- Scilab new capabilities
- MPI
- …

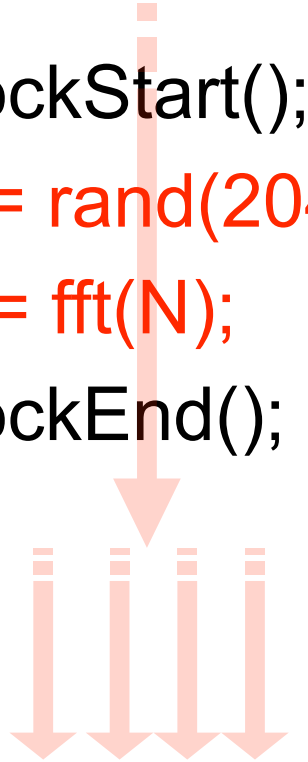
Scilab

Scilab 6 Execution

Combined parallelization



```
blockStart();
M = rand(2048, 2048);
P = fft(M);
blockEnd();
```

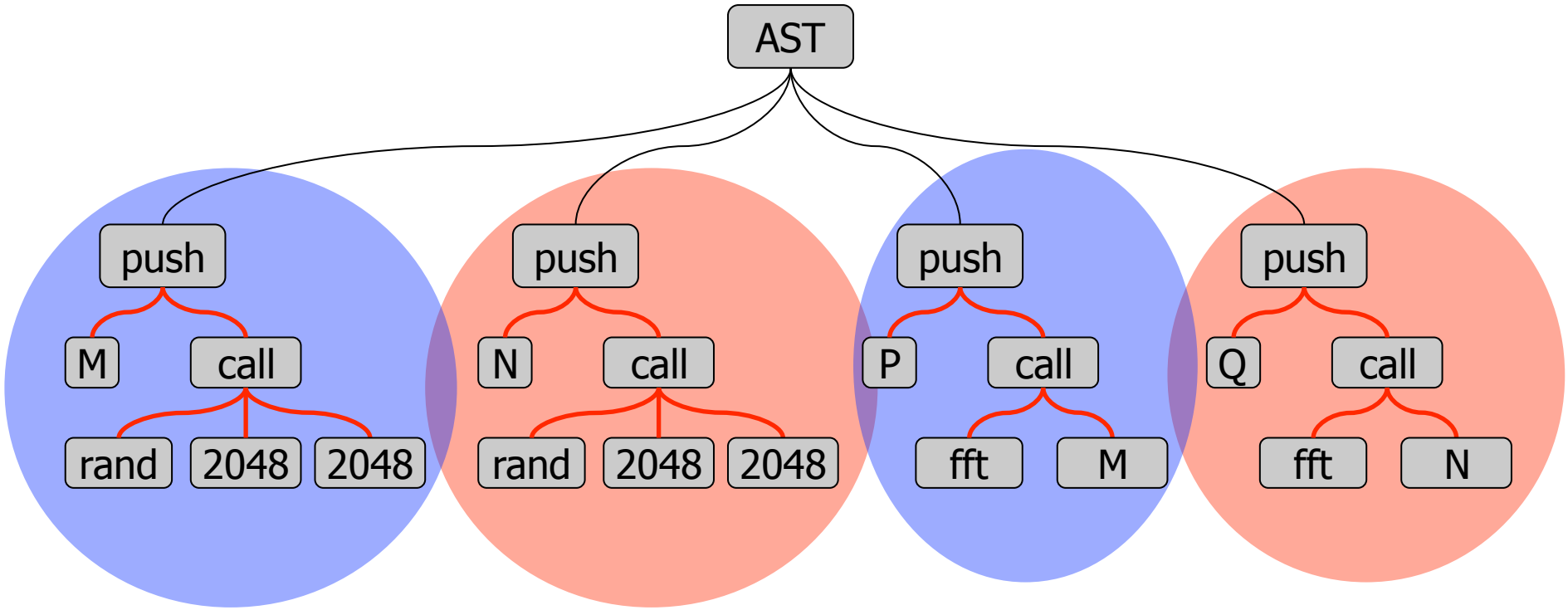


```
blockStart();
N = rand(2048, 2048);
Q = fft(N);
blockEnd();
```


Scilab

Scilab 6 Execution

Automatic parallelization?



Thanks for your attention

www.scilab.org

Scilab