The Numerical Template Toolbox

An Architecture-aware EDSL for Scientific Computing

Joel Falcou
Pierre Esterie Antoine Tran tan
Jean-Thierry Lapresté
Mathias Gaunard

LRI - University Paris Sud XI - CNRS
Institut Pascal - Université Clermont II
MetaScale SAS

DSLDI - 01/07/2013

Context

In Scientific Computing ...

- there is *Scientific*

Context

In Scientific Computing ...

- there is *Scientific*
  - Applications are domain driven
  - Users $\neq$ Developers
  - Users are reluctant to changes

Context

In Scientific Computing ...

- there is *Scientific*
  - Applications are domain driven
  - Users $\neq$ Developers
  - Users are reluctant to changes
- there is *Computing*

Context

In Scientific Computing ...

- there is *Scientific*
  - Applications are domain driven
  - Users $\neq$ Developers
  - Users are reluctant to changes
- there is *Computing*
  - Computing requires performance ...
  - ... which implies architectures specific tuning
  - ... which requires expertise
  - ... which may or may not be available

Context

In Scientific Computing ...

- there is *Scientific*
  - Applications are domain driven
  - Users $\neq$ Developers
  - Users are reluctant to changes
- there is *Computing*
  - Computing requires performance ...
  - ... which implies architectures specific tuning
  - ... which requires expertise
  - ... which may or may not be available

The Problem

People *using* computers to do science want to do *science* first.

Talk Layout

What's NT^2 ?

A Scientific Computing Library

- Provide a simple, MATLAB-like interface for users
- Provide high-performance computing entities and primitives
- Easily extendable

A Research Platform

- Simple framework to add new optimization schemes
- Test bench for EDSL development methodologies
- Test bench for Generic Programming in real life projects

The NT^2 API

Principles

- `table<T,S>` is a simple, multidimensional array object that exactly mimics MATLAB array behavior and functionalities
- 300+ functions usable directly either on `table` or on any scalar values as in MATLAB

The NT^2 API

Principles

- `table<T,S>` is a simple, multidimensional array object that exactly mimics MATLAB array behavior and functionalities
- 300+ functions usable directly either on `table` or on any scalar values as in MATLAB

How does it works

- Take a `.m` file, copy to a `.cpp` file

The NT^2 API

Principles

- `table<T,S>` is a simple, multidimensional array object that exactly mimics MATLAB array behavior and functionalities
- 300+ functions usable directly either on `table` or on any scalar values as in MATLAB

How does it works

- Take a `.m` file, copy to a `.cpp` file
- Add `#include <nt2/nt2.hpp>` and do cosmetic changes

The NT^2 API

Principles

- `table<T,S>` is a simple, multidimensional array object that exactly mimics MATLAB array behavior and functionalities
- 300+ functions usable directly either on `table` or on any scalar values as in MATLAB

How does it works

- Take a `.m` file, copy to a `.cpp` file
- Add `#include <nt2/nt2.hpp>` and do cosmetic changes
- Compile the file and link with `libnt2.a`

MATLAB you said ?

```
1  R = I(:,:,1);
2  G = I(:,:,2);
3  B = I(:,:,3);
4
5  Y = min(abs(0.299.*R+0.587.*G+0.114.*B),235);
6  U = min(abs(-0.169.*R-0.331.*G+0.5.*B),240);
7  V = min(abs(0.5.*R-0.419.*G-0.081.*B),240);
```

Now with NT^2

```
1  auto R = I(_,_,1);
2  auto G = I(_,_,2);
3  auto B = I(_,_,3);
4  table<float> Y, U, V;
5
6  Y = min(abs(0.299*R+0.587*G+0.114*B),235);
7  U = min(abs(-0.169*R-0.331*G+0.5*B),240);
8  V = min(abs(0.5*R-0.419*G-0.081*B),240);
```

Embedded Domain Specific Languages

EDSL in C++

- Relies on operator overload abuse
- Carry semantic information around code fragment
- Generic implementation become self-aware of optimizations

Advantages

- Allow introduction of DSLs without disrupting dev. chain
- Semantic defined as type informations means compile-time resolution
- Access to a large selection of runtime binding

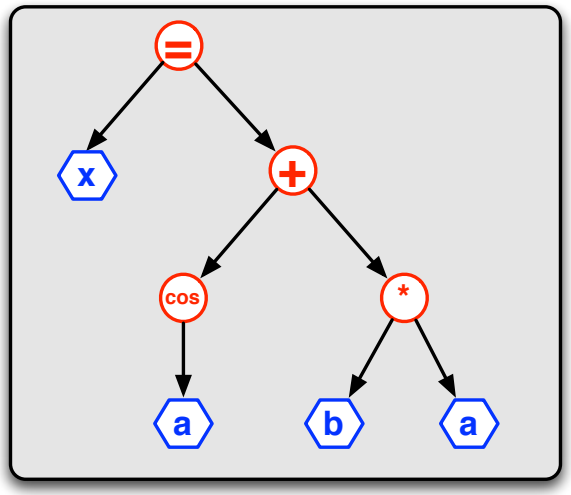
Expression Templates

```
matrix x(h,w),a(h,w),b(h,w);

x = cos(a) + (b*a);
```

```
expr<assign
    ,expr<matrix&>
    ,expr<plus
          , expr<cos
                ,expr<matrix&>
                >
          , expr<multiplies
                ,expr<matrix&>
                ,expr<matrix&>
                >
         >(x,a,b);
```

Arbitrary Transforms applied on the meta-AST




```
#pragma omp parallel for
for(int j=0;j<h;++j)
{
  for(int i=0;i<w;++i)
  {
    x(j,i) = cos(a(j,i))
           + (  b(j,i)
              * a(j,i)
           );
  }
}
```

Boost.Proto

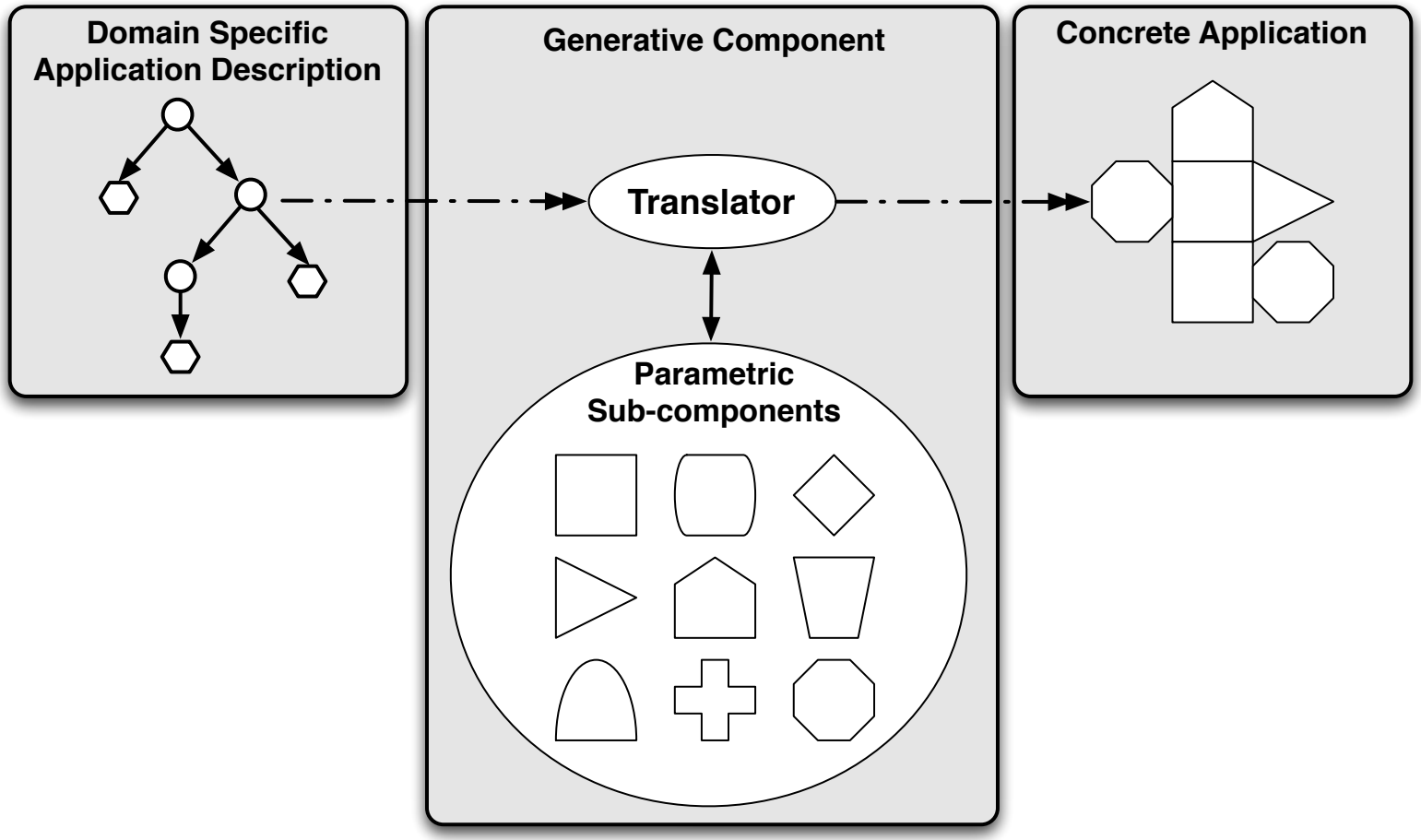
What's Boost.Proto

- EDSL for defining EDSLs in C++
- Generalize Expression Templates
- Easy way to define and test EDSL
- EDSL = some Grammar Rules + some Semantic Actions

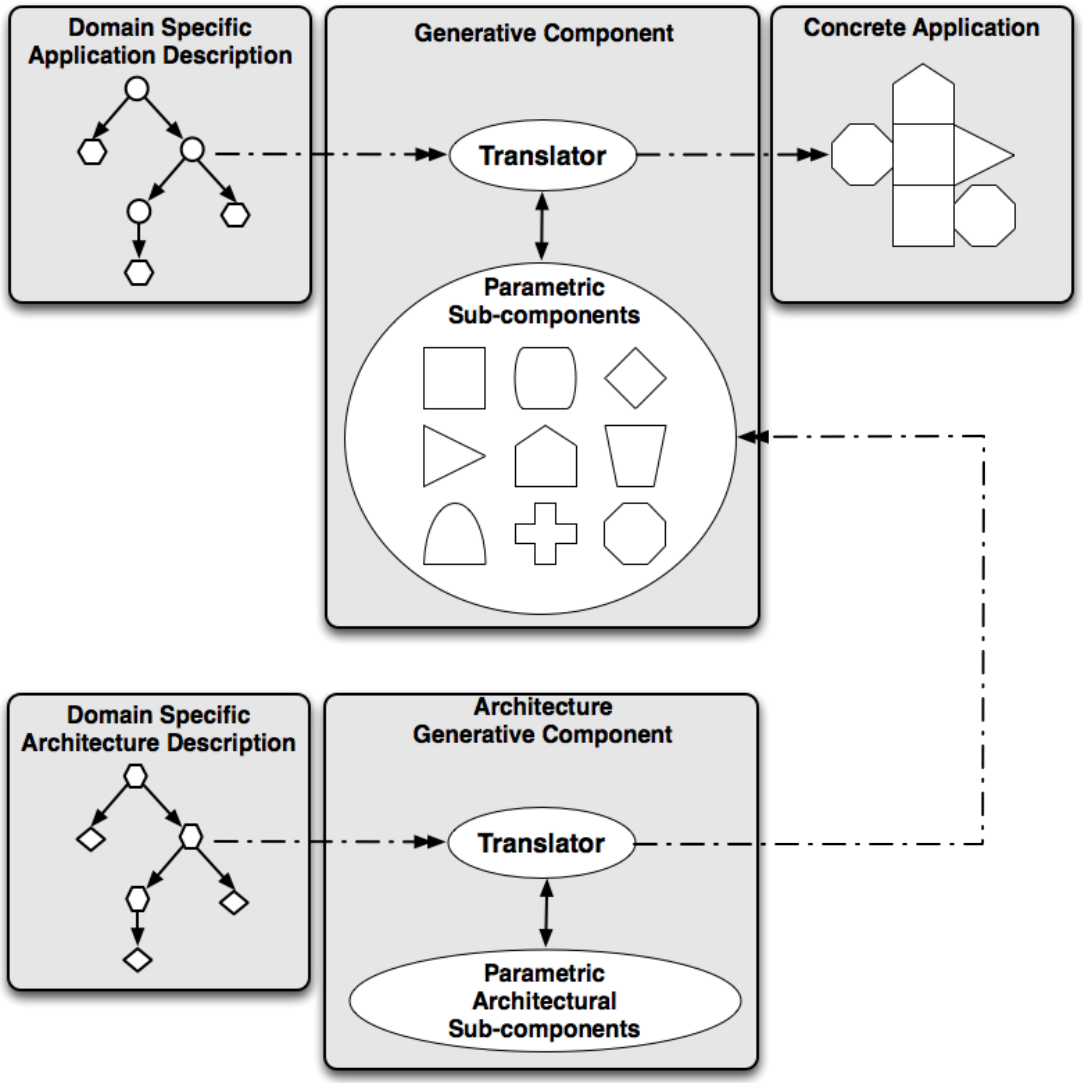
Boost.Proto Benefits

- Fast development process
- Compiler SDK: Grammar + Semantic + Code Generation process
- Easily extensible through user-defined AST Transforms

Generative Programming



Architecture Aware Generative Programming



Multipass EDSL Code Generation

Optimize

- AST Pattern Matching at expression construction
- E.g: `a+b*c` to `gemm` or `fma`, `x = inv(a)*b` to `x = linsolve(a,b)`

Multipass EDSL Code Generation

Optimize

- AST Pattern Matching at expression construction
- E.g: `a+b*c` to `gemm` or `fma`, `x = inv(a)*b` to `x = linsolve(a,b)`

Schedule

- Use AST node categorization to segment AST into forest
- Each sub-AST are tied to a **Parallel Skeleton**
- NT^2 skeletons can be hierarchically nested

Multipass EDSL Code Generation

Optimize

- AST Pattern Matching at expression construction
- E.g: `a+b*c` to `gemm` or `fma`, `x = inv(a)*b` to `x = linsolve(a,b)`

Schedule

- Use AST node categorization to segment AST into forest
- Each sub-AST are tied to a **Parallel Skeleton**
- NT^2 skeletons can be hierarchically nested

Run

- Generate code for each AST in the scheduled forest
- Potentially generate runtime calls for runtime optimization
- Classical EDSL code generation happens here on the correct hardware

NT2 before AADEMRAL

EDSL Core

- Code base was 11 KLOC
- 8KLOC was dedicated to the various Expression Template glue
- 3KLOC of actual smart stuff

Architectural support

- Altivec and SSE2 extension support
- Some vague pthread support
- Efforts were stagnant: adding a simple feature required multiple KLOC of code to change

NT2 after AADEMRAL

State of the code base

- Expression Template handling 1-2KLOC
- Skeleton handling : 1KLOC
- Actual features and function implementation : 10KLOC

Architectural support

- Support for everything SIMD from SSE to AVX, NEON, etc..
- Support for OpenMP, Intel TBB, openCL
- Time to complete new architecture support : 1 week to 1 month

Some Performances

RGB2YUV timing (in cycles/pixels)

Size	128x128	256x256	512x512	1024x1024
MATLAB 2010a (2 cores)	85	89	97	102
C (1 core)	23.6	23.8	23.9	24.0
NT^2 (1 core)	22.3	22.4	22.2	24.1
NT^2 OpenMP (2 cores)	11.4	11.4	11.5	12.2
NT^2 SIMD	5.5	5.6	5.6	5.9
NT^2 SIMD+OpenMP	2.9	2.9	2.9	3.0
NT^2 SIMD speed-up	3.9	3.9	3.9	4.0
NT^2 OpenMP speed-up	1.92	1.96	1.98	1.95
NT^2 vs MATLAB speed-up	29.3	30.7	33.5	34

Some Performances

Linear System Resolution (in GFLOPS)

```
nt2::tie(x,r) = linsolve(a,b)
```

GFLOPS	C++(`float`)	NT^2(`float`)
LAPACK GESV 1024*1024	85.3	83.1
LAPACK GESV 2048*2048	350.7	348.2
LAPACK GESV 12000*12000	735.7	738.1
MAGMA GESV 1024*1024	85.3	85.8
MAGMA GESV 2048*2048	235.8	238.6
MAGMA GESV 12000*12000	1297.8	1300.6

Let's round this up!

Parallel Computing for Scientist

- Software Libraries built as Generic and Generative components can solve a large chunk of parallelism related problems while being easy to use.
- Like regular language, EDSL needs informations about the hardware system
- Integrating hardware descriptions as Generic components increases tools portability and re-targetability

Let's round this up!

Parallel Computing for Scientist

- Software Libraries built as Generic and Generative components can solve a large chunk of parallelism related problems while being easy to use.
- Like regular language, EDSL needs informations about the hardware system
- Integrating hardware descriptions as Generic components increases tools portability and re-targetability

Recent activity

- Available at `http://www.github.com/MetaScale/nt2`
- A MATLAB to NT2 compiler has been designed
- Prototype for single source GPU support
- Toward a global generic approach to parallelism

Thanks for your attention