Applied Generic Programming to Accelerator Programming

Joel Falcou, Ian Masliah
Marc Baboulin

PARSYS - LRI

03/30/2015

Who are we

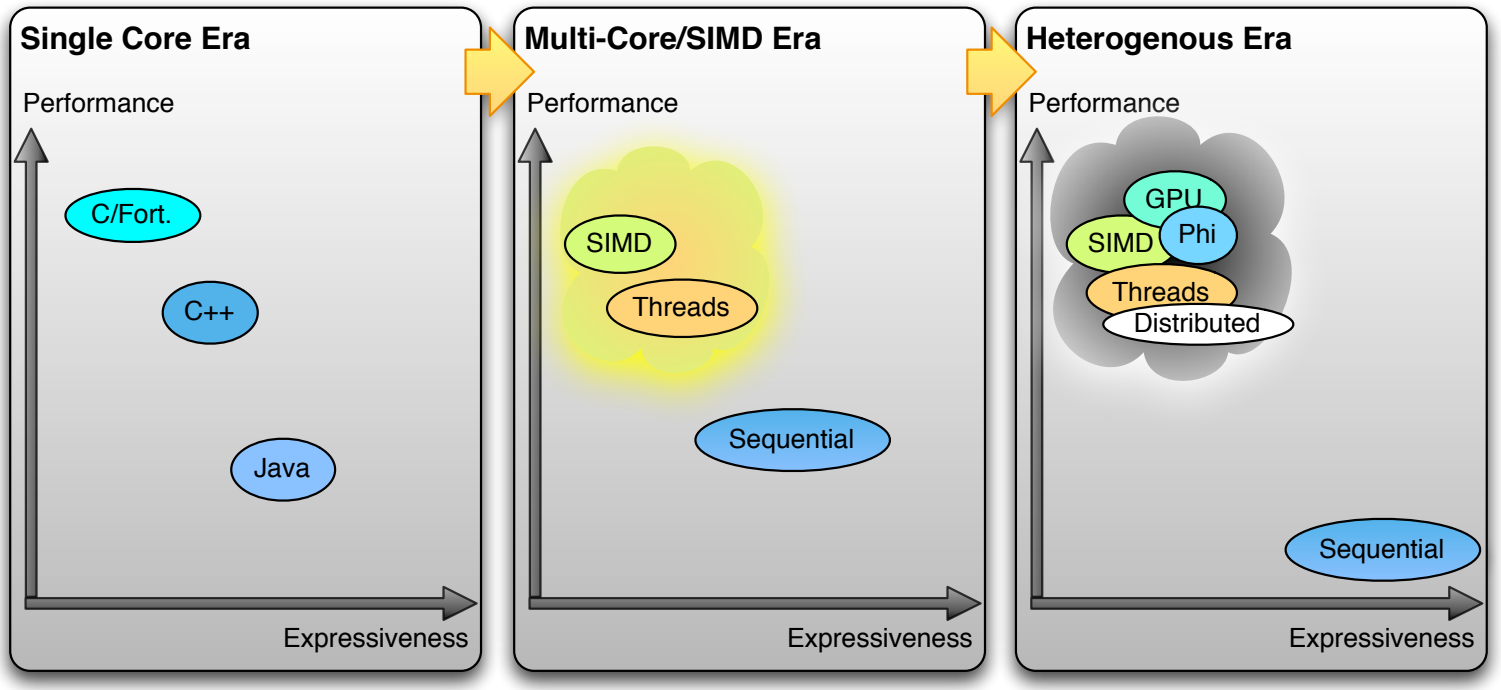
Parsys Team

- 6 permanents researchers
- 8 PHDs
- Spin-off company: NumScale

Research interests

- Algorithms for Computer Vision, Linear Algebra (LAPACK)
- High-Level parallel programming tools (Boost.SIMD, NT2)
- Hardware Exploration

The Usability Challenges of HPC



Designing tools for Scientific Computing

Challenges

1. Be non-disruptive
2. Domain driven optimizations
3. Provide intuitive API for the user
4. Support a wide architectural landscape
5. Be efficient

Designing tools for Scientific Computing

Challenges

1. Be non-disruptive
2. Domain driven optimizations
3. Provide intuitive API for the user
4. Support a wide architectural landscape
5. Be efficient

Our Approach

- Design tools as C++ libraries (1)
- Design these libraries as Domain Specific Embedded Language (DSEL) (2+3)
- Use Parallel Skeletons as parallel components (4)
- Use Generative Programming to deliver performance (5)

Domain Specific Embedded Languages

What's an DSEL ?

- DSL = Domain Specific Language
- Declarative language, easy-to-use, fitting the domain
- DSEL = DSL within a general purpose language

DSEL in practice

- Relies on operator overload abuse
- Carry semantic information around code fragment
- Library like code is self-aware of optimizations

Exploiting DSEL

- At the expression level: code generation for arbitrary hardware
- At the function level: inter-procedural optimizations

Parallel DSEL in practice

Objectives

- Apply DSEL generation techniques for different kind of hardware
- Demonstrate low cost of abstractions
- Demonstrate applicability of skeletons

Parallel DSEL in practice

Objectives

- Apply DSEL generation techniques for different kind of hardware
- Demonstrate low cost of abstractions
- Demonstrate applicability of skeletons

Our contribution

- BSP++ : Generic C++ BSP for shared/distributed memory
- Quaff: DSEL for skeleton programming
- Boost.SIMD: DSEL for portable SIMD programming
- NT2: Matlab like DSEL for scientific computing

Parallel DSEL in practice

Objectives

- Apply DSEL generation techniques for different kind of hardware
- Demonstrate low cost of abstractions
- Demonstrate applicability of skeletons

Our contribution

- BSP++ : Generic C++ BSP for shared/distributed memory
- Quaff: DSEL for skeleton programming
- Boost.SIMD: DSEL for portable SIMD programming
- NT2: Matlab like DSEL for scientific computing

NT^2

A Scientific Computing Library

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

Components

- Use SIMD for in-core optimizations
- Use recursive parallel skeletons
- Code is made independant of architecture and runtime

Parallel Skeletons in a nutshell

Basic Principles [COLE 1989]

- There are patterns in parallel applications
- Those patterns can be generalized in *Skeletons*
- Applications are assembled as combination of such patterns

Functionnal point of view

- Skeletons are *Higher-Order Functions*
- Skeletons support a compositionnal semantic
- Applications become composition of state-less functions
- Clear separation of concern between hard and soft

The Numerical Template Toolbox

Principles

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

The Numerical Template Toolbox

Principles

- `table<T,S>` is a simple, multidimensional array object that exactly mimics MATLAB array behavior and functionalities
- 500+ 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 Numerical Template Toolbox

Principles

- `table<T,S>` is a simple, multidimensional array object that exactly mimics MATLAB array behavior and functionalities
- 500+ 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 Numerical Template Toolbox

Principles

- `table<T,S>` is a simple, multidimensional array object that exactly mimics MATLAB array behavior and functionalities
- 500+ 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`

NT2 - From MATLAB ...

```
A1   = 1:1000;
A2   = A1 + randn(size(A1));

X = lu(A1*A1');

rms = sqrt( sum(sqr(A1(:) - A2(:))) / numel(A1) );
```

NT2 - ... to C++

```
table<double> A1 = _(1.,1000.);
table<double> A2 = A1 + randn(size(A1));

table<double> X = lu( mtimes(A1, trans(A1) );

double rms = sqrt( sum(sqr(A1(_) - A2(_))) / numel(A1) );
```

The Numerical Template Toolbox

Comparison to other libraries

Feature	Armadillo	Blaze	Eigen	MTL	uBlas	NT^2
MATLAB-like API	✓	–	–	–	–	✓
BLAS/LAPACK binding	✓	✓	✓	✓	✓	✓
SSE2+ support	✓	✓	✓	–	–	✓
AVX support	✓	✓	–	–	–	✓
AVX2 support	–	–	–	–	–	✓
Xeon Phi support	–	–	–	–	–	✓
Altivec support	–	–	✓	–	–	✓
ARM support	–	–	✓	–	–	✓
Threading support	–	–	–	–	–	✓
CUDA support	–	–	–	–	–	✓
MAGMA binding	–	–	–	–	–	✓

Semi-Normal Equation Solver

Context

- QR-based solver with least squares method
- Better behavior with regards to numerical precision
- Target applications: simulation, oil industry, engineering
- **Challenge**: Use mixed precision to improve performance
- **Challenge**: No current GPU implementation

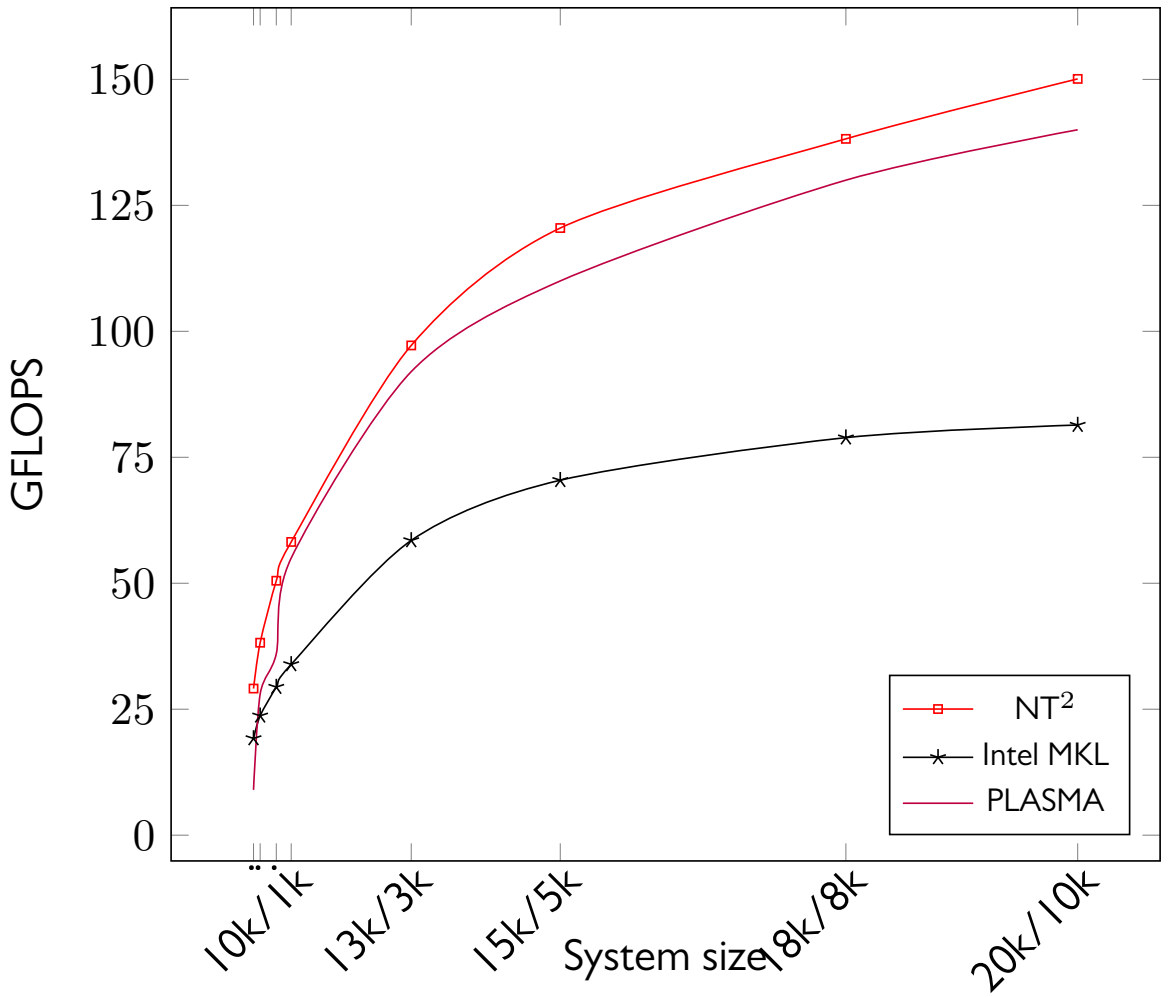
NT^2 Code

```
table<double> A,x,b;

// mixed_precision_ modify the semantic of the linear system A
x = linsolve(A, b, mixed_precision_);
```

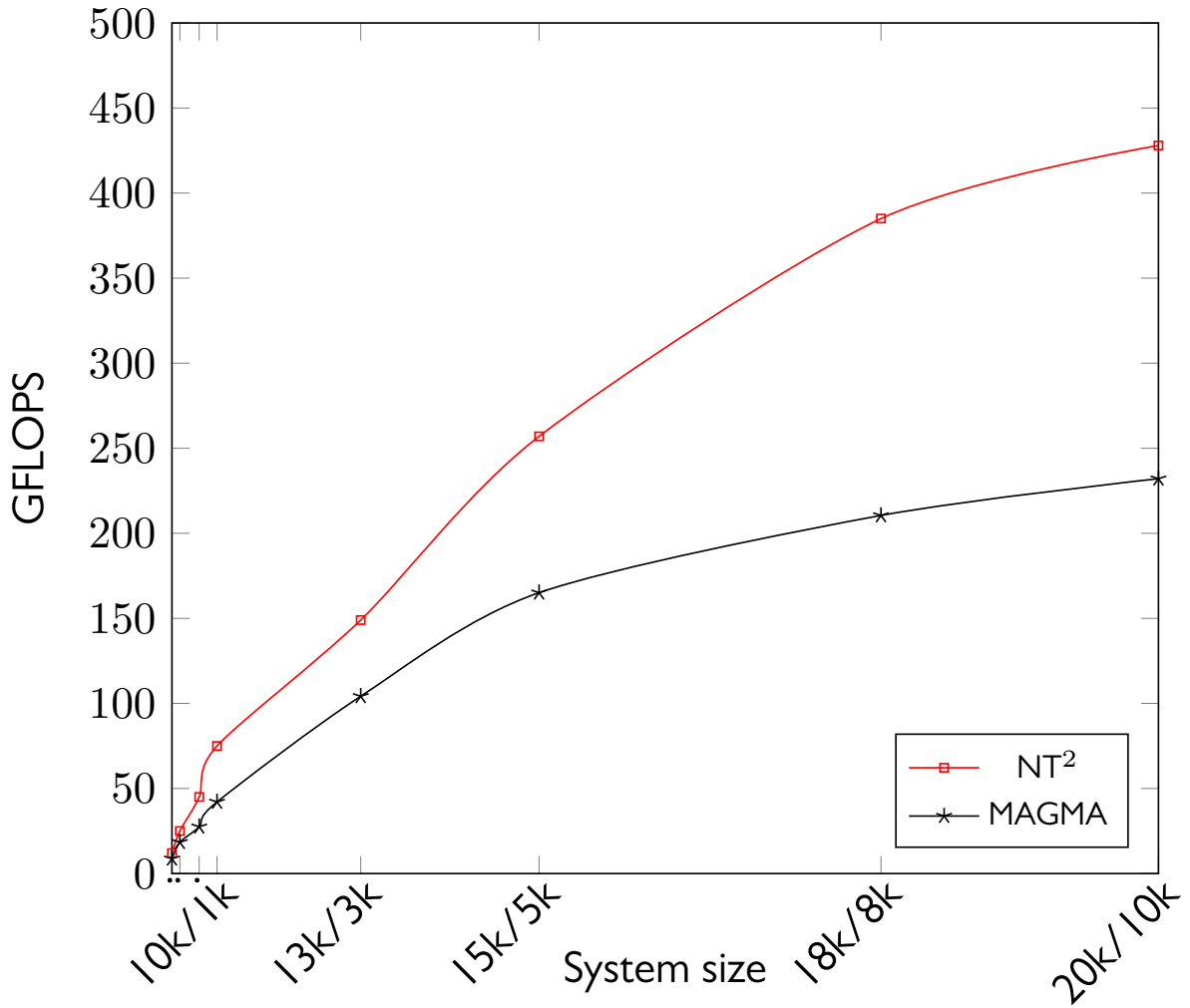
Semi-Normal Equation Method

Performance - CPU



Semi-Normal Equation Method

Performance - GPU



Interaction with NVIDIA

Software level

- Increase C++ support form NVCC
- Library/Compiler interaction at compiler level with NVCC
- Toward a bridging model of accelerators

Hardware level

- Basic access to latest version of NVIDIA Hardware
- Move toward mobility: access Tegra systems
- Better software optimisations if we knew what's going on in the GPU ;)

Thanks for your attention