BOAST

A Generative Language for Intensive Computing Kernels

Porting HPC applications to the Mont-Blanc Prototype

Kevin Pouget, Brice Videau, **Jean-François Méhaut**
UJF-CEA, CNRS, **LIG**

International Laboratory **LICIA**: UFRGS and LIG
Associated INRIA team **ExaSE**: UFRGS, USP, PUC Minas

EU FP7 IRSES HPC GA: INRIA, BCAM, UFRGS, UNAM, BRGM, **IST**erre
EU FP7 ICT Mont-Blanc: BSC, BULL, ARM, CNRS, INRIA, CINECA, Bristol, JSC, HLRS, LRZ...

HOSCAR Workshop, Gramado
September 17, 2014

Introduction

Scientific Application Optimization

- In the past, waiting for a new generation of hardware was enough to obtain performance gains.
- Nowadays, architecture are so different that performance regress when a new architecture is released.
- Sometimes the code is not fit anymore and cannot be compiled.
- Few applications can harness the power of current computing platforms.
- Thus, optimizing scientific application is of paramount importance.

Multiplicity of Computing Architectures

A High performance Computing application can encounter several types of architectures :

- General-Purpose Multicore CPU (Intel, AMD, PowerPC...)
- Graphical Accelerators (NVIDIA, ATI-AMD, ...)
- Computing Accelerators (Xeon Phi, MPPA-256, Tilera, CELL...)
- **Low power CPUs (ARM...)**

Those architectures can present drastically different characteristics.

Architectures Comparison

Architecture	AMD/Intel CPUs	ARM	GPUs	Xeon Phi
Cores	4-12	2-4	512-2496	60
Cache	3l	2l	2l incoherent	2l
Memory (GiB)	2-4 (per core)	1 (per core)	2-6	6-16
Vector Size	2-4	1-2	1-2	8
Peak GFLOPS	10-20 (per core)	2-4 (per core)	500-1500	1000
Peak GiB/s	20-40	2.5-5	150-250	200
TDP W	75	5	200	300
GFLOPS/W	1-3	2-4	2-7	3

Table : Comparison between commonly found architectures in HPC.

Exploiting Various Architectures

Usually work is done on a class of architecture (CPUs or GPUs or accelerators).

Well Known Examples

- Atlas (Linear Algebra CPU)
- PhiPAC (Linear Algebra CPU)
- Spiral (FFT CPU)
- FFTW (FFT CPU)
- NukadaFFT(FFT GPU)

No work targeting several class of architectures. What if the application is not based on a library ?

The Mont-Blanc European Projects

MONT-BLANC

Mont-Blanc 1 (2011-2015) Mont-Blanc 2 (2013-2016) :

- Develop prototypes of HPC clusters using low power commercially available embedded technology (ARM CPUs, low power GPUs...).
- Design the next generation in HPC systems based on embedded technologies and experiments on the prototypes.
- Develop a portfolio of existing applications to test these systems and optimize their efficiency, using BSC's OmpSs programming model (11 existing applications were selected for this portfolio).
- Build Software Stack (OS, runtime, performance tools,...)

Prototype : based on Exynos 5250 : ARM dual core Cortex A15 with T604 Mali GPU (OpenCL)

BigDFT a Tool for Nanotechnologies

Ab initio simulation :

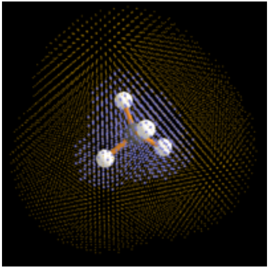
- Simulates the properties of crystals and molecules,
- Computes the electronic density, based on Daubechie wavelet.

This formalism was chosen because it is fit for HPC computations :

- Each orbital can be treated independently most of the time,
- Operator on orbitals are simple and straightforward.

Mainly developed in Europe :

- CEA-DSM/INAC (Grenoble)
- Basel, Louvain la Neuve,...



Electronic density around a methane molecule.

BigDFT as an HPC application

Implementation details :

- 200,000 lines of Fortran 90 and C
- Supports MPI, OpenMP, CUDA and OpenCL
- Uses BLAS
- Scalability up to 16000 cores of Curie and 288GPUs

Operators can be expressed as 3D convolutions :

- Wavelet Transform
- Potential Energy
- Kinetic Energy

These convolutions are separable and filter are short (16 elements).
Can take up to **90% of the computation time** on some systems.

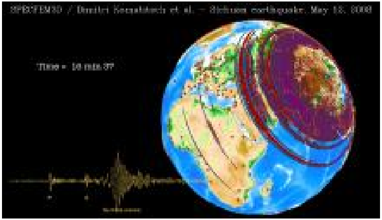
SPECFEM3D a tool for wave propagation research

Wave propagation simulation :

- Used for geophysics and material research,
- Accurately simulate earthquakes,
- Based on spectral finite element.

Developed all around the world :

- France (CNRS Marseille),
- Switzerland (ETH Zurich) CUDA,
- United States (Princeton) Networking,
- Grenoble (LIG/CNRS) OpenCL.



Sichuan earthequake.

SPECFEM3D as an HPC application

Implementation details :

- 80,000 lines of Fortran 90
- Supports MPI, CUDA, OpenCL and an OMPSs + MPI miniapp
- Scalability up to 693,600 cores on IBM BlueWaters

Talk Outline

Case Study 1 : BigDFT's MagicFilter

The simplest convolution found in BigDFT, corresponds to the potential operator.

Characteristics

- Separable,
- Filter length 16,
- Transposition,
- Periodic,
- Only 32 operations per element.

Pseudo code

```
double filt[16] = {F0, F1, ... , F15};
void magicfilter(int n, int ndat,
                 double *in, double *out){
  double temp;
  for(j=0; j<ndat; j++) {
    for(i=0; i<n; i++) {
       temp = 0;
       for(k=0; k<16; k++) {
         temp+= in[ ((i-7+k)%n) + j*n]
                 * filt[k];
       }
       out[j + i*ndat] = temp;
} } }
```

Case study 2 : SPECFEM3D port to OpenCL

Existing CUDA code :

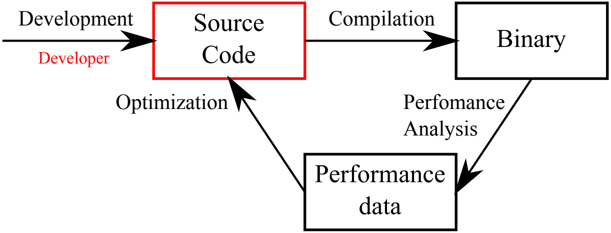
- 42 kernels and 15000 lines of code
- kernels with 80+ parameters
- $\sim$ 7500 lines of cuda code
- $\sim$ 7500 lines of wrapper code

Objectives :

- Factorize the existing code,
- Single OpenCL and CUDA description for the kernels,
- Validate without unit tests, comparing native Cuda to generated Cuda executions
- Keep similar performances.

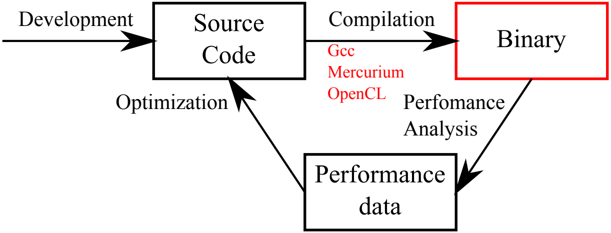
A Parametrized Generator

Classical Software Development Loop



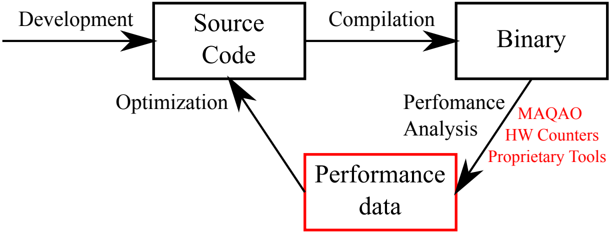
- Kernel optimization workflow
- Usually performed by a knowledgeable developer

Classical Software Development Loop



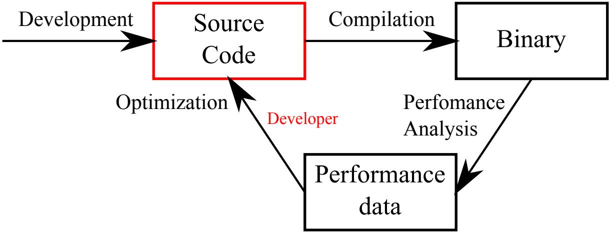
- Compilers perform optimizations
- Architecture specific or generic optimizations

Classical Software Development Loop



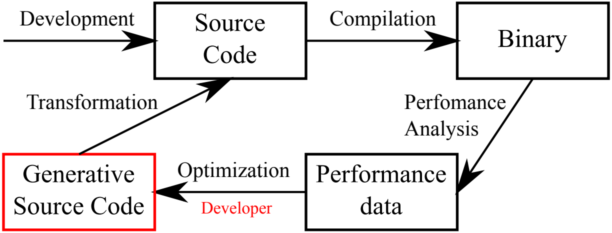
- Performance data hint at source transformations
- Architecture specific or generic hints

Classical Software Development Loop



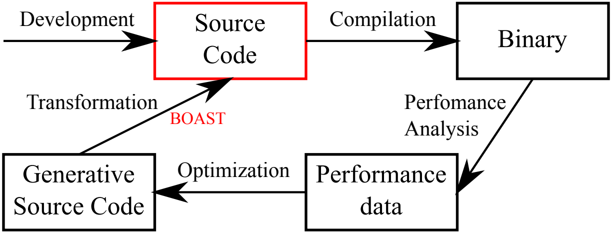
- Multiplication of kernel versions or loss of versions
- Difficulty to benchmark versions against each-other

BOAST Development Loop



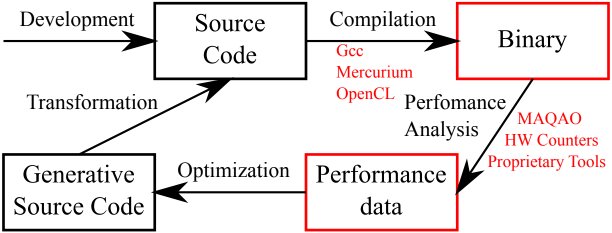
- Meta-programming of optimizations in BOAST
- High level object oriented language

BOAST Development Loop



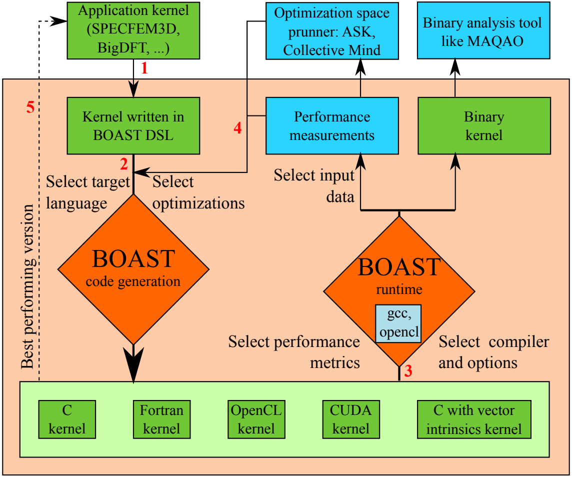
- Generate combination of optimizations
- C, OpenCL, FORTRAN and CUDA are supported

BOAST Development Loop



- Compilation and analysis are automated
- Selection of best version can also be automated

BOAST


Application kernel (SPECFEM3D, BigDFT, ...)
Optimization space pruner: ASK, Collective Mind
Binary analysis tool like MAQAO
1
5
Kernel written in BOAST DSL
4
Performance measurements
Binary kernel
2
Select target language
Select optimizations
Select input data
Best performing version
BOAST
code generation
BOAST
runtime
gcc, opencl
Select performance metrics
Select compiler and options
3
C kernel
Fortran kernel
OpenCL kernel
CUDA kernel
C with vector intrinsics kernel

Use Case Driven

Parameters arising in a convolution :

- Filter : length, values, center.
- Direction : forward or inverse convolution.
- Boundary conditions : free or periodic.
- Unroll factor : arbitrary.

How are those parameters constraining our tool ?

Features required

Unroll factor :

- Create and manipulate an unknown number of variables,
- Create loops with variable steps.

Boundary conditions :

- Manage arrays with parametrized size.

Filter and convolution direction :

- Transform arrays.

And of course be able to describe convolutions and output them in different languages.

Proposed Generator

Idea : use a high level language with support for operator overloading to describe the structure of the code, rather than trying to transform a decorated tree.
Define several abstractions :

- Variables : type (array, float, integer), size...
- Operators : affect, multiply...
- Procedure and functions : parameters, variables...
- Constructs : for, while...

Sample Code : Variables and Parameters

```
#simple Variable
i = Int "i"
#simple constant
lowfil = Int( "lowfil", :const => 1-center )
#simple constant array
fil = Real("fil", :const => arr, :dim => [ Dim(lowfil,upfil) ])
#simple parameter
ndat = Int("ndat", :dir => :in)
#multidimensional array, an output parameter
y = Real("y", :dir => :out, :dim => [ Dim(ndat), Dim(dim_out_min, dim_out_max) ] )
```

Variables and Parameters are objects with a name, a type, and a set of named properties.

Sample Code : Procedure Declaration

The following declaration :

```
p = Procedure("magic_filter", [n,ndat,x,y], [lowfil,upfil])
open p
```

Outputs Fortran :

```
subroutine magicfilter(n, ndat, x, y)
  integer(kind=4), parameter :: lowfil = -8
  integer(kind=4), parameter :: upfil = 7
  integer(kind=4), intent(in) :: n
  integer(kind=4), intent(in) :: ndat
  real(kind=8), intent(in), dimension(0:n-1, ndat) :: x
  real(kind=8), intent(out), dimension(ndat, 0:n-1) :: y
```

Or C :

```
void magicfilter(const int32_t n, const int32_t ndat, const double * x, double * y){
  const int32_t lowfil = -8;
  const int32_t upfil = 7;
```

Sample Code : Constructs and Arrays

The following declaration :

```
unroll = 5
pr For(j,1,ndat-(unroll-1), unroll) {
#.....
  pr tt2 === tt2 + x[k,j+1]*fil[l]
#.....
}
```

Outputs Fortran :

```
do j=1, ndat-4, 5
  !......
  tt2=tt2+x(k,j+1)*fil(l)
  !......
enddo
```

Or C :

```
for(j=1; j<=ndat-4; j+=5){
/*...........*/
  tt2=tt2+x[k-0+(j+1-1)*(n-1-0+1)]*fil[l-lowfil];
/*...........*/
}
```

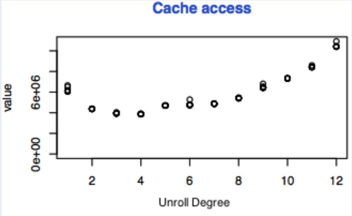
Generator Evaluation

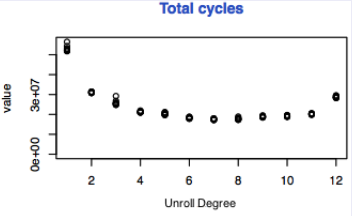
Back to the test cases :

- The generator was used to unroll the Magicfilter an evaluate it's performance on an ARM processor and an Intel processor.
- The generator was used to describe SPECFEM3D kernel.

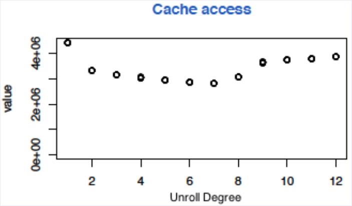
Performance Results

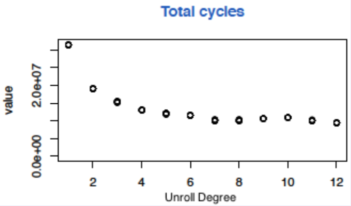
Tegra2



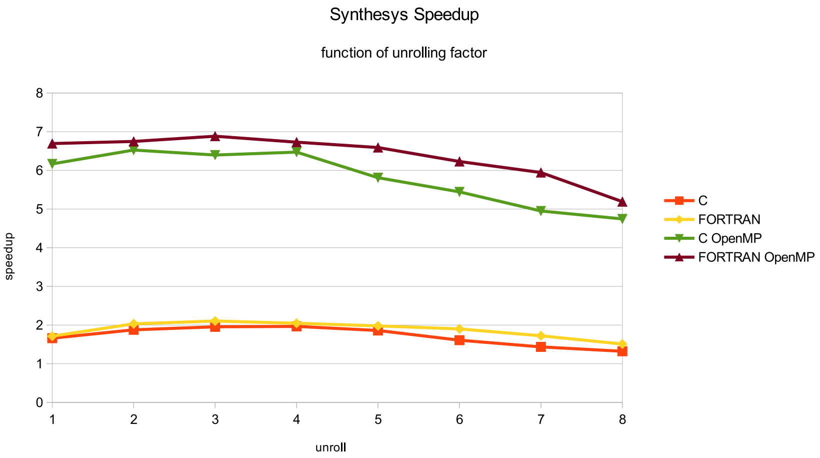


Intel T7500





BigDFT Synthesis Kernel



Improvement for BigDFT

- Most of the convolutions have been ported to BOAST.
- Results are encouraging : on the hardware BigDFT was hand optimized for, convolutions gained on average between 30 and 40% of performance.
- MagicFilter OpenCL versions tailored for problem size by BOAST gain 10 to 20% of performance.

SPECFEM3D OpenCL port

Fully ported to OpenCL with comparable performances (using the `global_s362ani_small` test case) :

- On a 2*6 cores (E5-2630) machine with 2 K40, using 12 MPI processes :
  - OpenCL : 4m15s
  - CUDA : 3m10s
- On an 2*4 cores (E5620) with a K20 using 6 MPI processes :
  - OpenCL : 12m47s
  - CUDA : 11m23s

Difference comes from the capacity of cuda to specify the minimum number of blocks to launch on a multiprocessor. Less than 4000 lines of BOAST code (7500 lines of cuda originally).

Conclusions and Future Work

Conclusions

Generator has been used to test several loop unrolling strategies in BigDFT.
Highlights :

- Several output languages.
- All constraints have been met.
- Automatic benchmarking framework allows us to test several optimization levels and compilers.
- Automatic non regression testing.
- Several algorithmically different versions can be generated (changing the filter, boundary conditions...).

Future Works and Considerations

Future work :

- Produce an autotuning convolution library.
- Implement a parametric space explorer or use an existing one (ASK : Adaptative Sampling Kit, Collective Mind...).
- Vector code is supported, but needs improvements.
- Test the OpenCL version of SPECFEM3D on the Mont-Blanc prototype.

Question raised :

- Is this approach extensible enough ?
- Can we improve the language used further ?