

Improve the applicability of highly efficient stencil compilers to a wider class of problems

DONFACK Simplice

Olaf Schenk

Matthias Christen

Radim Janalik

Patrick Sanan

Universita della Svizzera Italiana

September 19, 2014

Introduction

- Numerical simulation is one of the most greatest concerns in many scientific and engineering applications:
 - chemical engineering
 - materials and physical sciences
 - economic modeling
 - ...
- In order to simulate phenomena close to the reality, these applications give rise to large linear systems whose runtime requires an increasing computational power.

Introduction (cont.)

- The growing demand for efficient solutions requires the design of hardware technologies with increasing computational power.
- Architectural trends show a growing gap between time for processors to perform arithmetic operations and time they take to communicate.

Introduction (cont.)

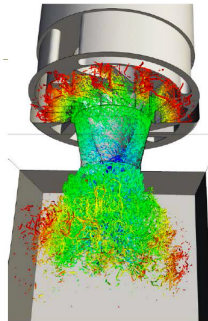
- The growing demand for efficient solutions requires the design of hardware technologies with increasing computational power.
- Architectural trends show a growing gap between time for processors to perform arithmetic operations and time they take to communicate.
 - Some existing algorithms will communicate too much compared to the potential gain that could be obtained from the power of the processors.
 - Most manufacturers agree that improving the latency tends to become a difficult barrier to overcome.

Introduction (cont.)

- The growing demand for efficient solutions requires the design of hardware technologies with increasing computational power.
- Architectural trends show a growing gap between time for processors to perform arithmetic operations and time they take to communicate.
 - Some existing algorithms will communicate too much compared to the potential gain that could be obtained from the power of the processors.
 - Most manufacturers agree that improving the latency tends to become a difficult barrier to overcome.
- Most existing algorithms should be redeveloped to take into account this growing gap.

Test case

- YALES2 aims at the solving of two-phase combustion from primary atomization to pollutant prediction on massive complex meshes.
- Several billions of elements.
[V. Moreau '10]
- Solvers require increasing computational steps.



YALES2, PRECCINSTA
burner 2.6B test case (V.
Moureau).

Courtesy of M.-C. Sawley (Intel).

Motivation

- Stencil codes are an important part of solvers.
- The variety of stencil computations make these codes difficult to optimize manually and to adapt to the architecture trends.
- Parallel solvers that contain global communication are performance bottlenecks for machines with very large node counts and correspondingly complex networks.

Goals

EU EXA2CT project:

- Brings experts together at the cutting edge of the development of **exascale solvers**, related algorithmic techniques, and HPC software architects for programming models and communication.
- Develop novel algorithms and programming models to tackle major obstacles to using solvers at exascale in many scientific codes.



Our work package goal: increase the arithmetic intensity of solvers.

Approach

Our approach in 2 parts:

- Use the help of stencil compilers to optimize the data locality and increase the arithmetic intensity of solvers.
- Use of communication avoiding algorithms.

Approach

Our approach in 2 parts:


- Use the help of stencil compilers to optimize the data locality and increase the arithmetic intensity of solvers.
- Use of communication avoiding algorithms.

My focus:

- Improve the applicability of highly efficient stencil compilers, used namely in solvers, to a wider class of problems.
- Design a stencil framework for CPUs/GPUs to target several reference applications whose solvers require increasing computational steps.
- Study the interoperability of stencil compilers with novel algorithms: e.g communication avoiding algorithms.

Stencil Example

Laplacian

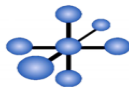
$$\frac{\partial u}{\partial t} - \alpha \nabla^2 u = 0$$


Weighted Jacobi: $U^{t+1} = U^t + \omega D^{-1} r^t$

Stencil Example

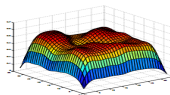
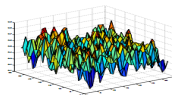
Laplacian

$$\frac{\partial u}{\partial t} - \alpha \nabla^2 u = 0$$




$$\text{Weighted Jacobi: } U^{t+1} = U^t + \omega D^{-1} r^t$$

Smoothing error



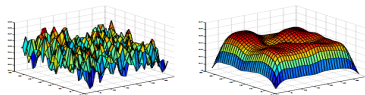
Stencil Example

Laplacian

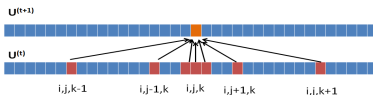
$$\frac{\partial u}{\partial t} - \alpha \nabla^2 u = 0$$


Weighted Jacobi: $U^{t+1} = U^t + \omega D^{-1} r^t$

Smoothing error

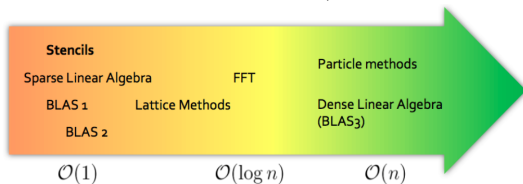


Memory access: $U[i, j, k] = 6.0 * U[i, j, k] - U[i - 1, j, k] - U[i + 1, j, k] - U[i, j - 1, k] - U[i, j + 1, k] - U[i, j, k - 1] - U[i, j, k + 1]$



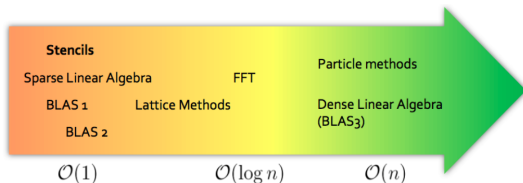
Arithmetic intensity bounds

Arithmetic intensity = Flops / Transferred data.

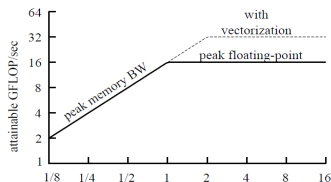


Arithmetic intensity bounds

Arithmetic intensity = Flops / Transferred data.



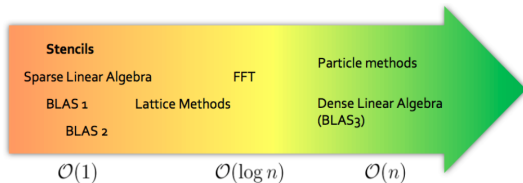
Performance roofline.



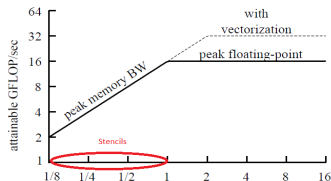
Arithmetic intensity (Flop/byte). [Williams '08]

Arithmetic intensity bounds

Arithmetic intensity = Flops / Transferred data.



Performance roofline.

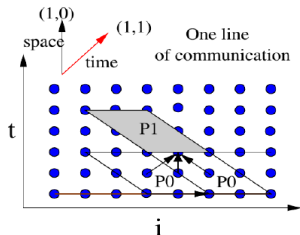


Arithmetic intensity (Flop/byte). [Williams '08]

- 1 Introduction
- 2 Stencil computations
- 3 Increase arithmetic intensity**
- 4 Applicability and Results
- 5 Conclusion

Stencil compilers (1)

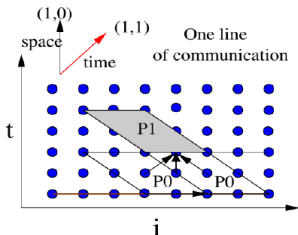
Pluto: an automatic parallelizer and locality optimizer for multicores that performs a source to source code transformation. [Bondhugula'08]



- Automatic loop parallelization.
- Locality optimizations based on the polyhedral model.
- Automatic loop vectorization.

Stencil compilers (1)

Pluto: an automatic parallelizer and locality optimizer for multicores that performs a source to source code transformation. [Bondhugula'08]



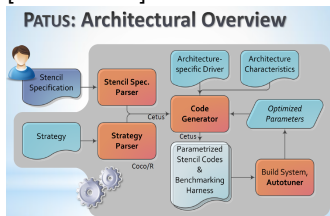
- Automatic loop parallelization.
- Locality optimizations based on the polyhedral model.
- Automatic loop vectorization.

Some drawbacks:

- The development has stagnated, based on old architectures.
- It is based on loops transformation, does not take into account (or only poorly) the stencil specifications.
- No GPU version.

Stencil compilers (2)

PATUS: a node-level stencil compiler developed in EXA2CT project.
[Christen'11]

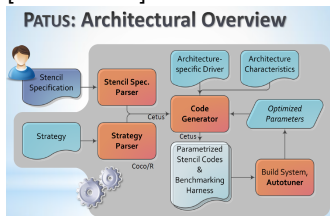


Courtesy of M. Christen.

- Cache Blocking
- Vectorization
- Loop unrolling
- Take into account the stencil specifications.
- Much more ...

Stencil compilers (2)

PATUS: a node-level stencil compiler developed in EXA2CT project.
[Christen'11]



Courtesy of M. Christen.

Some drawbacks:

- Some limitations based on the stencil model, it does not match some reference applications.
- It takes only into account the spatial locality, but not the temporal locality.
- It is still under development, need support and more strong tests.

Contribution

1. Use the help of PLUTO.

- Use of PLUTO
 - generate the tiling representation of the code.
 - determine the appropriate tile size based on performance guide.
- Vectorize inner-tile loops and rearrange the memory acces.
- Parallelize the outer-loop using openmp or pthreads.

Contribution

1. Use the help of PLUTO.

- Use of PLUTO
 - generate the tiling representation of the code.
 - determine the appropriate tile size based on performance guide.
- Vectorize inner-tile loops and rearrange the memory acces.
- Parallelize the outer-loop using openmp or pthreads.

2. Use of PATUS.

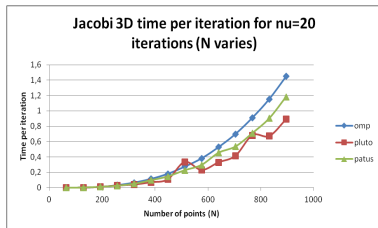
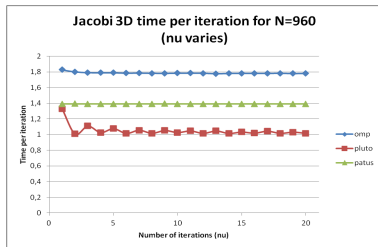
- Reformulate the target stencil codes to match PATUS specifications.
- Use of PATUS to generate and tune the stencil code.

- 1 Introduction
- 2 Stencil computations
- 3 Increase arithmetic intensity
- 4 Applicability and Results**
- 5 Conclusion

Jacobi 3D

Jacobi 3D 7-pt stencil with constant coefficients.

$$U[i, j, k] = 6.0 * U[i, j, k] - U[i - 1, j, k] - U[i + 1, j, k] \\ - U[i, j - 1, k] - U[i, j + 1, k] - U[i, j, k - 1] - U[i, j, k + 1]$$



Performance of PLUTO and PATUS compared to OPENMP on AMD opteron using 16 cores.

Jacobi 3D

#points\#iterations	Time omp/time Patus					
	1	2	4	8	16	20
64	0,98	0,86	0,77	0,78	0,74	0,71
128	1	0,98	0,99	0,99	1	1
192	0,86	0,82	0,82	0,81	0,8	0,8
256	1,28	1,28	1,28	1,27	1,26	1,27
320	1,3	1,3	1,3	1,3	1,29	1,3
384	1,16	1,17	1,17	1,17	1,17	1,17
448	1,22	1,21	1,22	1,22	1,22	1,22
512	1,25	1,26	1,25	1,22	1,25	1,2
576	1,29	1,29	1,29	1,28	1,28	1,28
640	1,17	1,16	1,16	1,16	1,16	1,16
704	1,31	1,32	1,31	1,31	1,3	1,3
768	1,3	1,28	1,29	1,29	1,28	1,28
832	1,3	1,28	1,28	1,28	1,28	1,28
896	1,25	1,23	1,23	1,22	1,22	1,22
960	1,31	1,29	1,28	1,28	1,27	1,27

Figure : Performance improvement of PATUS compared to OMP on AMD opteron using 16 cores. Greater than 1, better PATUS is.

Long range 3D

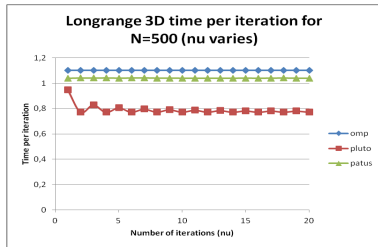
Long range 3D 25-pt stencil with variable coefficients.

```

1 // 3D long-range line update (single precision)
2 for(i=4; i<nnx-4; i++) {
3     lap = coef0 * V(i,j,k)
4     + coef[1] * ( V(i+1,j ,k ) + V(i-1,j ,k ) )
5     + coef[1] * ( V(i ,j+1,k ) + V(i ,j-1,k ) )
6     + coef[1] * ( V(i ,j ,k+1) + V(i ,j ,k-1) )
7     + coef[2] * ( V(i+2,j ,k ) + V(i-2,j ,k ) )
8     + coef[2] * ( V(i ,j+2,k ) + V(i ,j-2,k ) )
9     + coef[2] * ( V(i ,j ,k+2) + V(i ,j ,k-2) )
10    + coef[3] * ( V(i+3,j ,k ) + V(i-3,j ,k ) )
11    + coef[3] * ( V(i ,j+3,k ) + V(i ,j-3,k ) )
12    + coef[3] * ( V(i ,j ,k+3) + V(i ,j ,k-3) )
13    + coef[4] * ( V(i+4,j ,k ) + V(i-4,j ,k ) )
14    + coef[4] * ( V(i ,j+4,k ) + V(i ,j-4,k ) )
15    + coef[4] * ( V(i ,j ,k+4) + V(i ,j ,k-4) );
16    U(i,j,k) = 2.f * V(i,j,k) - U(i,j,k) + ROC2(i,j,k) * lap;
17 }

```

Innermost loop shown only



Performance of PLUTO and PATUS compared to OMP on AMD opteron using 16 cores.

Long range 3D

#points\#iterations	Time omp/time Patus					
	1	2	4	8	16	20
64	0,85	0,85	0,86	0,85	0,85	0,85
128	1,21	1,23	1,22	1,23	1,24	1,25
192	1,6	1,6	1,59	1,6	1,59	1,61
256	4,64	4,82	5,5	5,56	4,67	5,09
320	1,35	1,36	1,36	1,37	1,39	1,4
384	1,31	1,31	1,31	1,33	1,33	1,33
448	1,35	1,38	1,38	1,38	1,38	1,38
512	3,4	3,51	2,82	3,37	3,7	3,23

Figure : Performance improvement of PATUS compared to OMP on AMD opteron using 16 cores. Greater than 1, better PATUS is.

Krylov methods

Krylov methods: $K_k(A, v) = \text{span} \{v, Av, A^2v, \dots, A^{k-1}v\}$

Help of polynomial preconditioning:

- Replace Av with a $P_m(A)v$ with a fixed coefficients.
 $K_k(P_m(A), v) = \text{span} \{v, P_m(A)v, P_m(A)^2v, \dots, P_m(A)^{k-1}v\}$
Where $P_m(A)$ is a low-order polynomial
- Use a stencil compiler to increase arithmetic intensity.

[Vanroose'13]

[Work in progress]

Polynomial preconditioned Conjugate Gradient

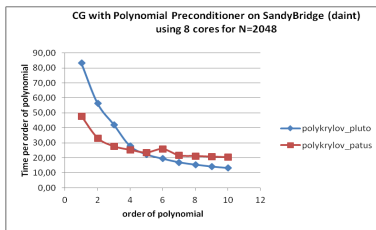
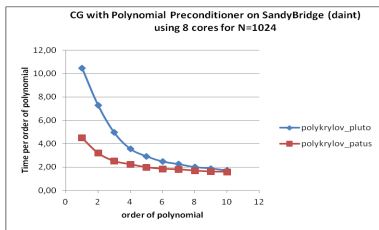


Figure : Solution time in polynomial preconditioned Conjugate Gradient on Sandy Bridge using 8 cores

- PATUS leads to better results for small problem and small order of the polynomial.
- PLUTO takes advantage of the time blocking.

Conclusion

- Use of PLUTO and PATUS to increase the arithmetic intensity in Jacobi 3D, long range 3D and preconditioner CG.
- PATUS optimizes the spatial locality of stencil codes while PLUTO takes good advantage of the time blocking.

Conclusion

- Use of PLUTO and PATUS to increase the arithmetic intensity in Jacobi 3D, long range 3D and preconditionner CG.
- PATUS optimizes the spatial locality of stencil codes while PLUTO takes good advantage of the time blocking.

Prospects:

- Take advantage of the time blocking for the code generated with PATUS.
 - Combine PATUS and PLUTO.
- Integrate some of our work in PATUS.

Conclusion

- Use of PLUTO and PATUS to increase the arithmetic intensity in Jacobi 3D, long range 3D and preconditionner CG.
- PATUS optimizes the spatial locality of stencil codes while PLUTO takes good advantage of the time blocking.

Prospects:

- Take advantage of the time blocking for the code generated with PATUS.
 - Combine PATUS and PLUTO.
- Integrate some of our work in PATUS.
- Leverage task parallelism.
- Take advantage of communication avoiding algorithms.

ICL collaborations

- Leverage task parallelism by expressing the algorithm in terms of tasks.
- Implement communication avoiding GMRES for structured problem or reference applications.
- Design the GPU version.
- Make the code fault-tolerant (global EXA2CT perspectives, informal discussion with G. Bolsica).

Thank you

Thank you