Automatic Parallelization for Shared Memory Scientific Multiprocessing
An Analysis & Comparison

Idan Mosseri[1,3], Re’em Harel[2,4], Harel Levin[1,5], Matan Rusanovsky[2,3], Gal Oren[1,3]

[1] Department of Physics, Nuclear Research Center - Negev, P.O.B. 9001, Be'er-Sheva, Israel.
[2] Israel Atomic Energy Commission, P.O.B. 7061, Tel Aviv, Israel.
[3] Department of Computer Science, Ben-Gurion University of the Negev, P.O.B. 653, Be'er Sheva, Israel.
[4] Department of Physics, Bar-Ilan University, IL52900, Ramat-Gan, Israel.
[5] Department of Mathematics and Computer Science, The Open University of Israel, P.O.B. 808, Ra'anana, Israel.

Introduction

- **Parallelization is essential** in order to exploit the full benefits of multi-core architectures, which have become widespread in recent years.
- Designing a valid parallelization for applications is **not always a simple task**.
- **Automatic parallelization** tools were proposed to **ease this process.**

Tool	License	Supported Language	Last Updated
AutoPar (ROSE)	Free	C, C++	May, 2017
Par4All (PIPS)	Free	C, Fortran, CUDA, OpenCL	May, 2015
Cetus	Free	C	Feb, 2017
SUIF compiler	Free	C, Fortran	2001
ICC	Proprietary	C, Fortran, C++	Jan, 2017
Polaris compiler	Free	Fortran 77	Unknown
S2P	Proprietary	C	Unknown

Our Focus

In this study, we overview and compare three of free up-to-date tools that were found to be most suitable for scientific code parallelization:

AutoPar

Pros

- + **suitable for OOP.**
- + Handles nested loops.
- + Verifies existing OpenMP directives.
- + Can be directed to add OpenMP directives regardless of errors.
- + Modifications are accompanied by clear explanation and reasoning in its' output.

Cons

- - **May require programmer intervention** to handle function side-effects, classes etc. (via annotation file).
- - May add incorrect OpenMP directives when given the "No-aliasing" option.

Par4All

Pros

- + **Suitable for GPUs.**
- + Automatically analyzes function side effects and pointer aliasing.
- + Supports many data types.
- + Supports Fortran.

Cons

- - **Dead code will not be parallelized.**
- - May change the code structure.

Cetus

Pros

- + **Loop size dependent parallelization**
- + Handles nested loops.
- + Provides cross-platform interface.
- + Verifies existing OpenMP directives.
- + Modifications are accompanied by clear explanation and reasoning in its' output.

Cons

- - **Does not support function calls.**
- - Adds Cetus's pragmas which create excess code.
- - May create uncompilable reduction clauses.

Comparison

We compare AutoPar, Par4All and Cetus on several variations of the Matrix Multiplication, with **each variant emphasizing a different** parallel shared memory management **pitfall**.

Array Reduction/Privatization: parallel directive can only be added with array reduction clause or with an array declared as private?

- **AutoPar** inserts OpenMP directives to the two outermost loops.
- **Par4All** inserts OpenMP directive only to the outermost loop.
- **Cetus** inserts OpenMP directives to all three loops.

Verify Alias Dependence: Does the tool check for dependencies other then pointer aliasing when the No-Aliasing option is turned on?

- **AutoPar** (when given the No-Aliasing option) ignored all data dependencies and inserted an incorrect directive to the innermost loop.
- **Par4All** did not insert any OpenMP directive (even with the No-Aliasing option).
- **Cetus** stepped into an internal error

Loop Unroll support: Will the tool insert OpenMP directives into unrolled loops?

- **AutoPar** did not insert any OpenMP directives.
- **Par4All** added an OpenMP directive only to the outermost loop as in the first test.
- **Cetus** Added OpenMP directives to all three loops. However, the innermost loops’ directive is uncompilable as it contains a reduction clause for multiple array cells.

Function call support: Will the tool insert OpenMP directives to loops containing function calls with/without side effects?

- **AutoPar** could not parallelize the code without an Annotation file.
- **Par4All** Inserts an OpenMP directive to the outermost loop as before.
- **Cetus** did not add any OpenMP directives.

Hardware & Specifications

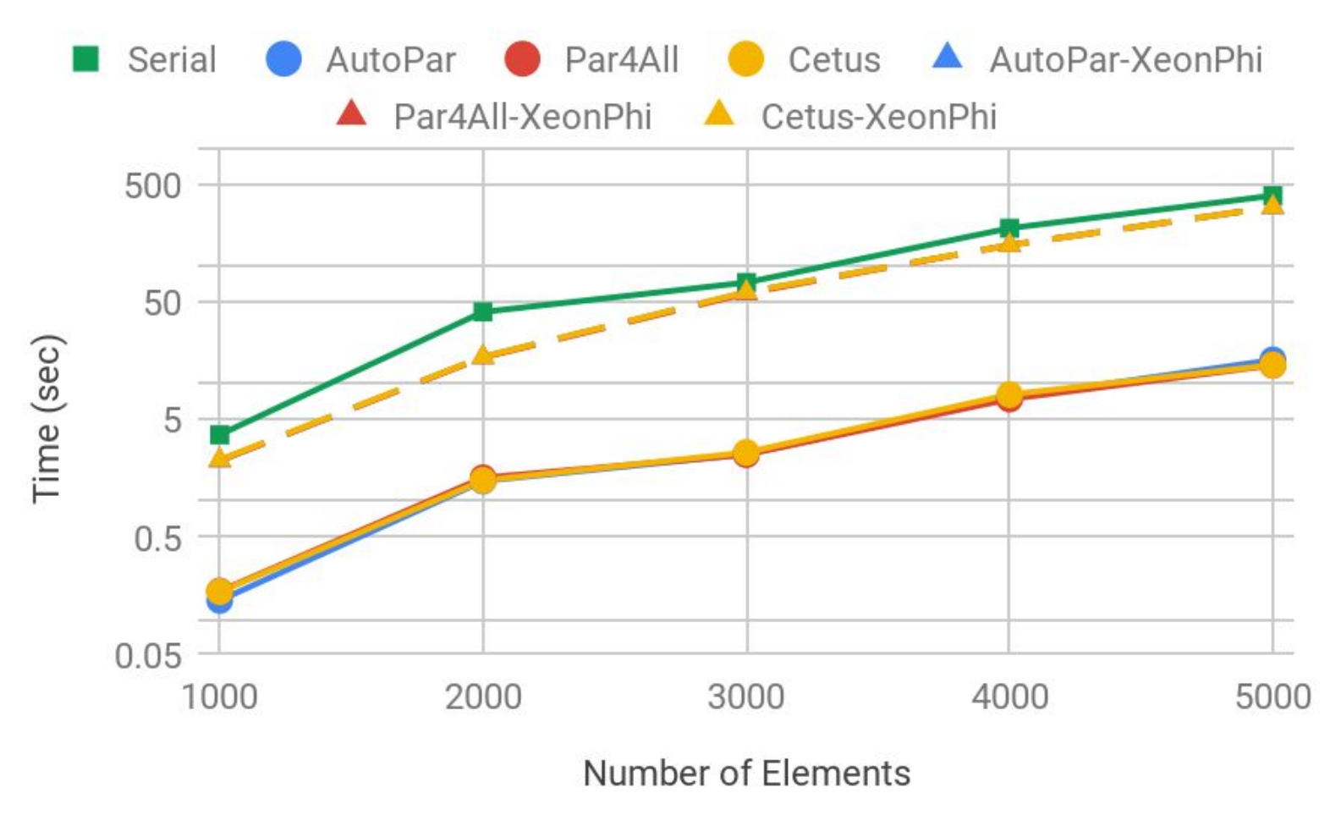
All test-cases were compiled

- Using Intel(R) C Compiler XE 2018.
- Update 5 for Linux*.
- Using internal optimizations (i.e. -O3). Unless stated otherwise

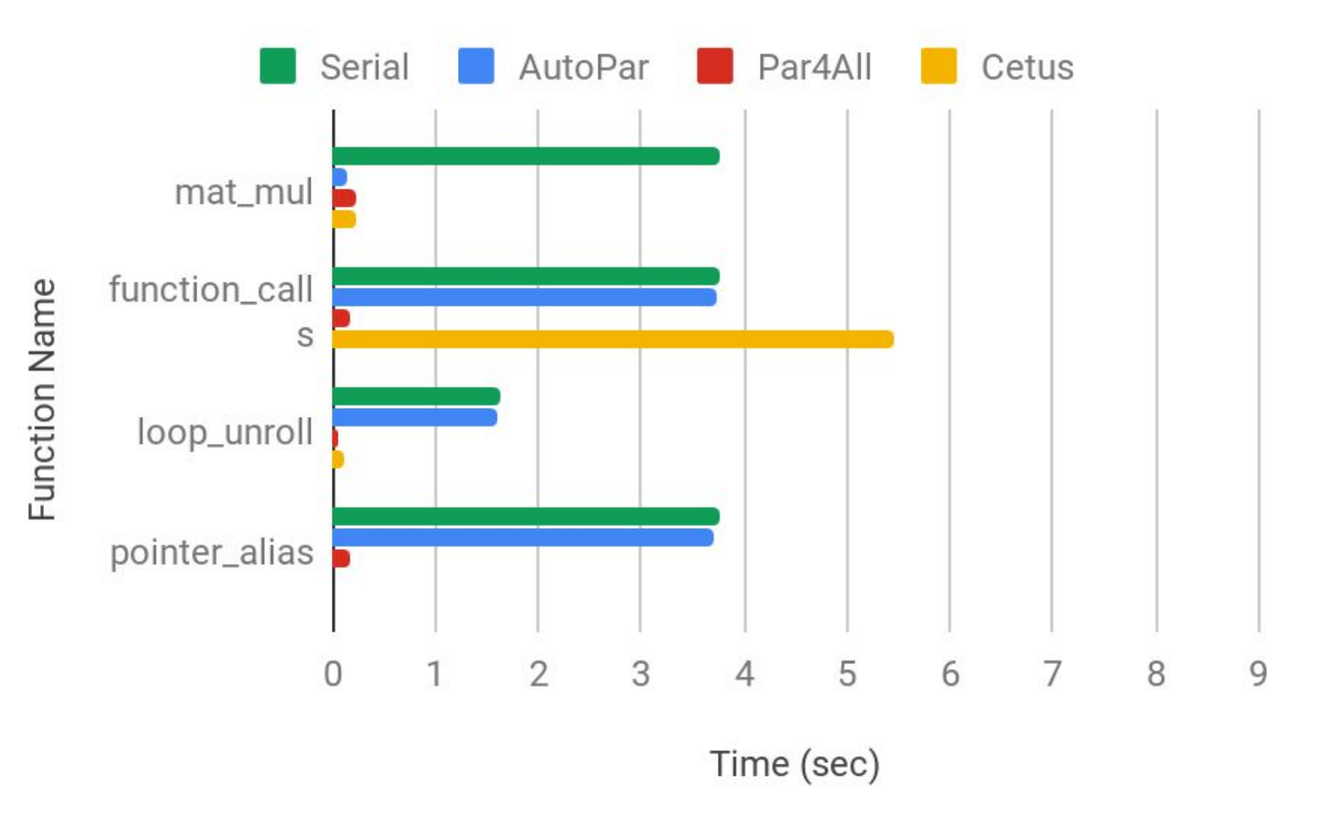
And executed on

- Machine with two Intel(R) Xeon(R) CPU E5-2683 v4 processors.
- Intel(R) Xeon-Phi co-processor 5100 series (rev 11) in native mode.
- NVIDIA(R) Tesla(R) P100-PCIE-16GB.

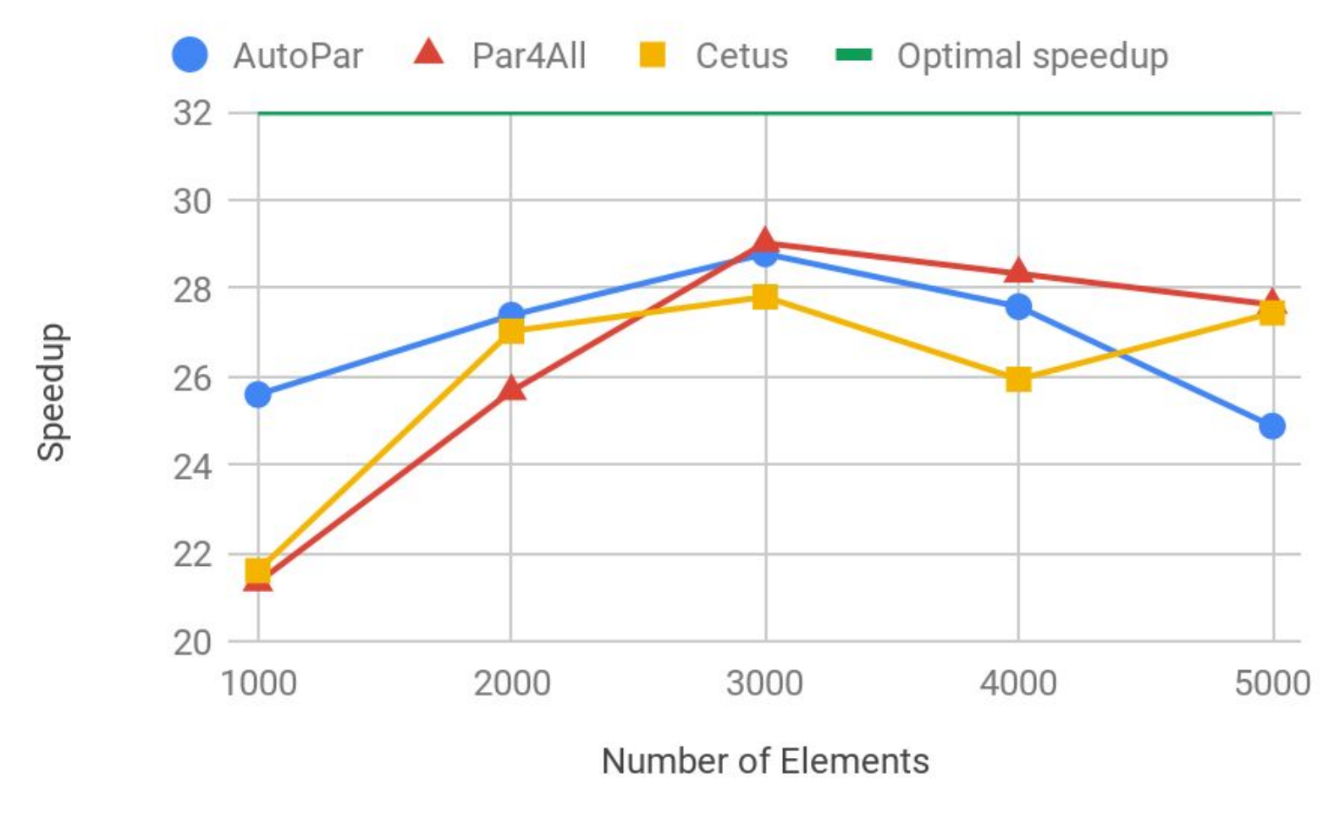
Runtime Analysis



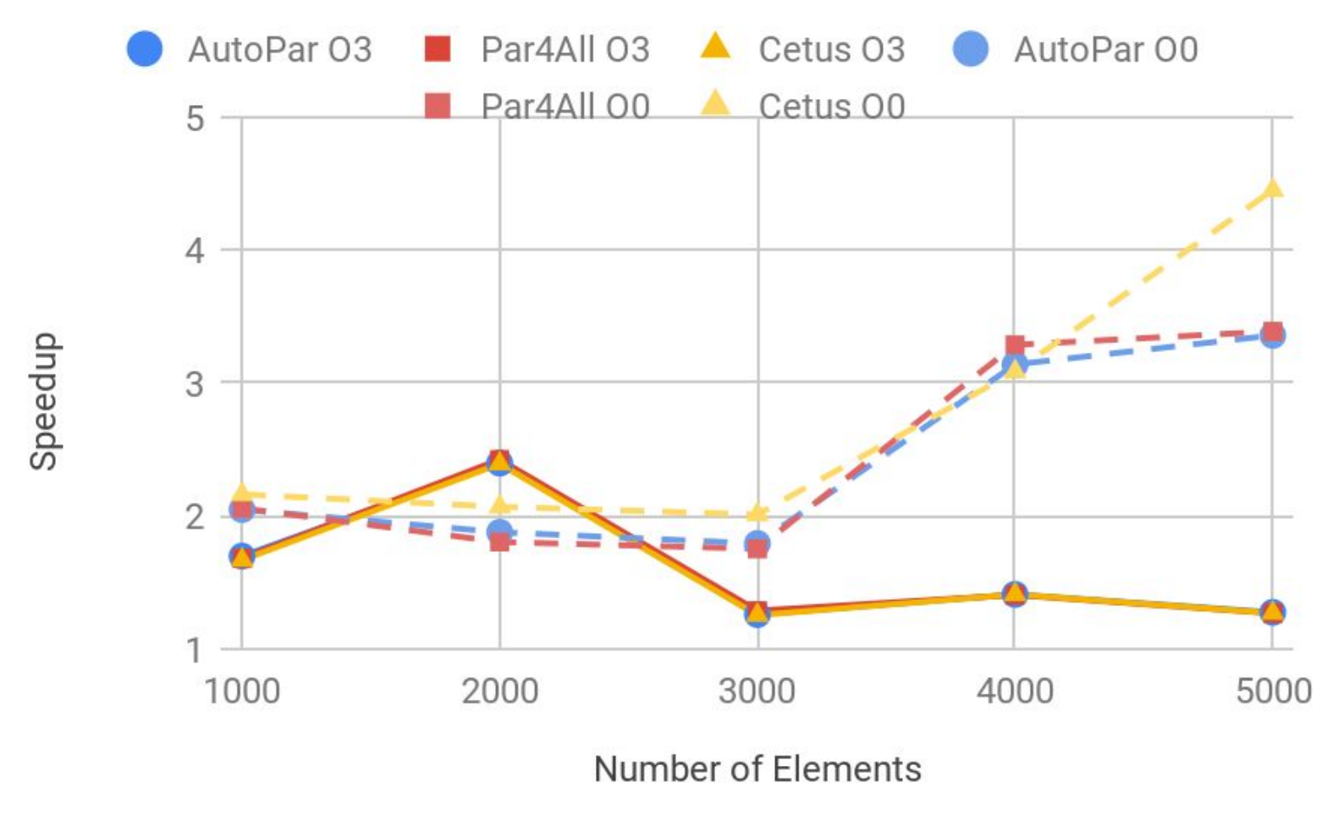
mat_mul speedup with different number of elements in log-scale.



Function execution times.



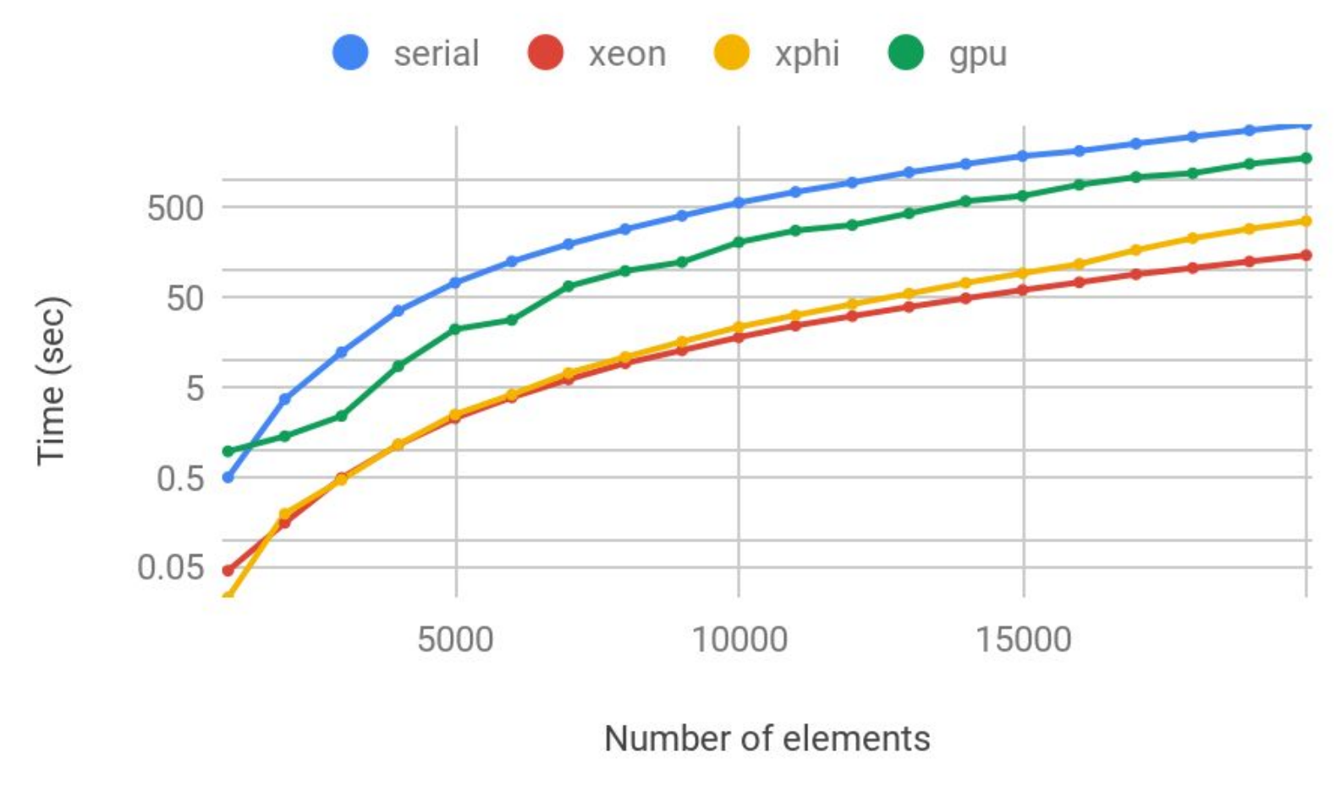
mat_mul speedup with different number of elements in log-scale.



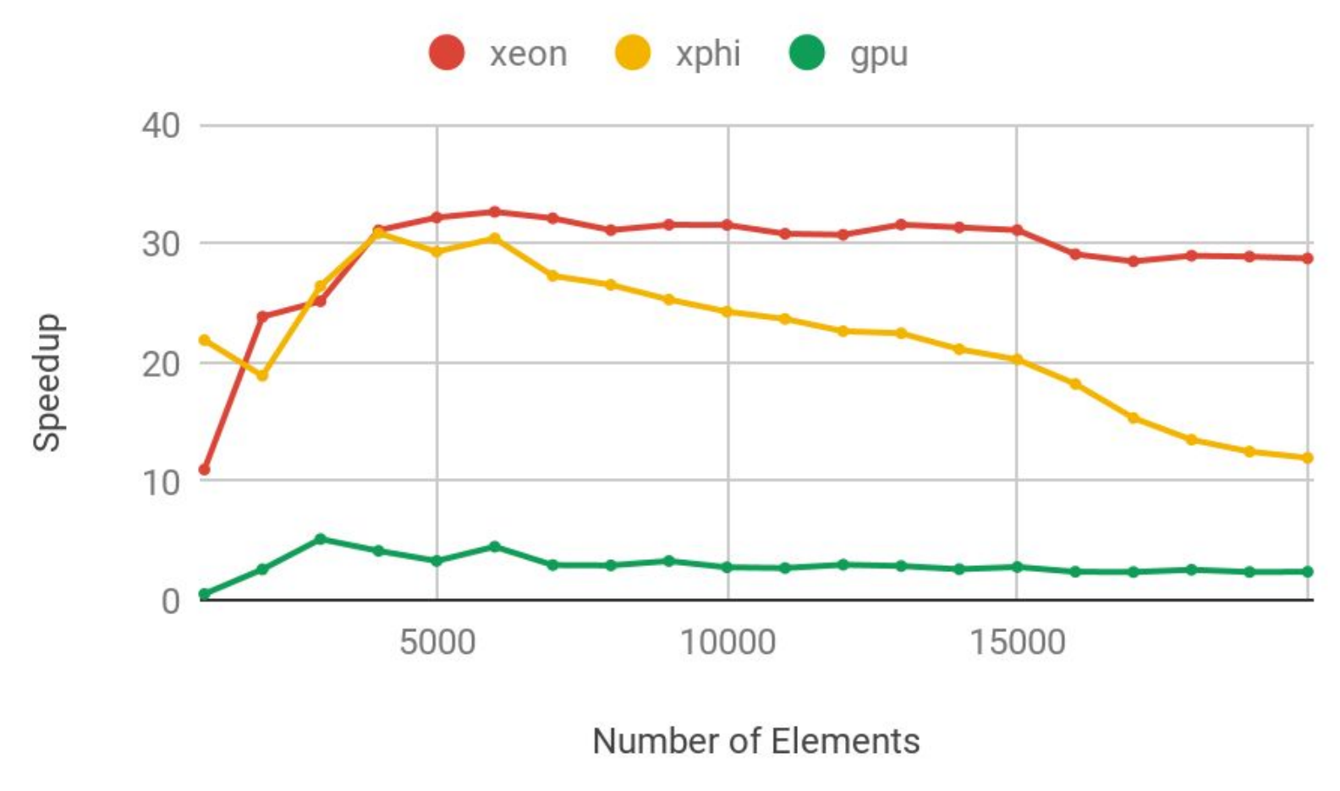
mat_mul speedup on Intel Xeon-Phi compared to serial run (on Intel Xeon) with -O3 and -O0.

Accelerators & Co-processors

The code was further modified so that Par4All would transform it to CUDA code. The graphs below compare the modified code with Par4All on Intel Xeon, Xeon-Phi and NVIDIA Tesla



Modified mat_mul runtime on Intel Xeon Xeon-Phi and NVIDIA Tesla



Modified mat_mul Speedup on Intel Xeon Xeon-Phi and NVIDIA Tesla

NAS Parallel Benchmark

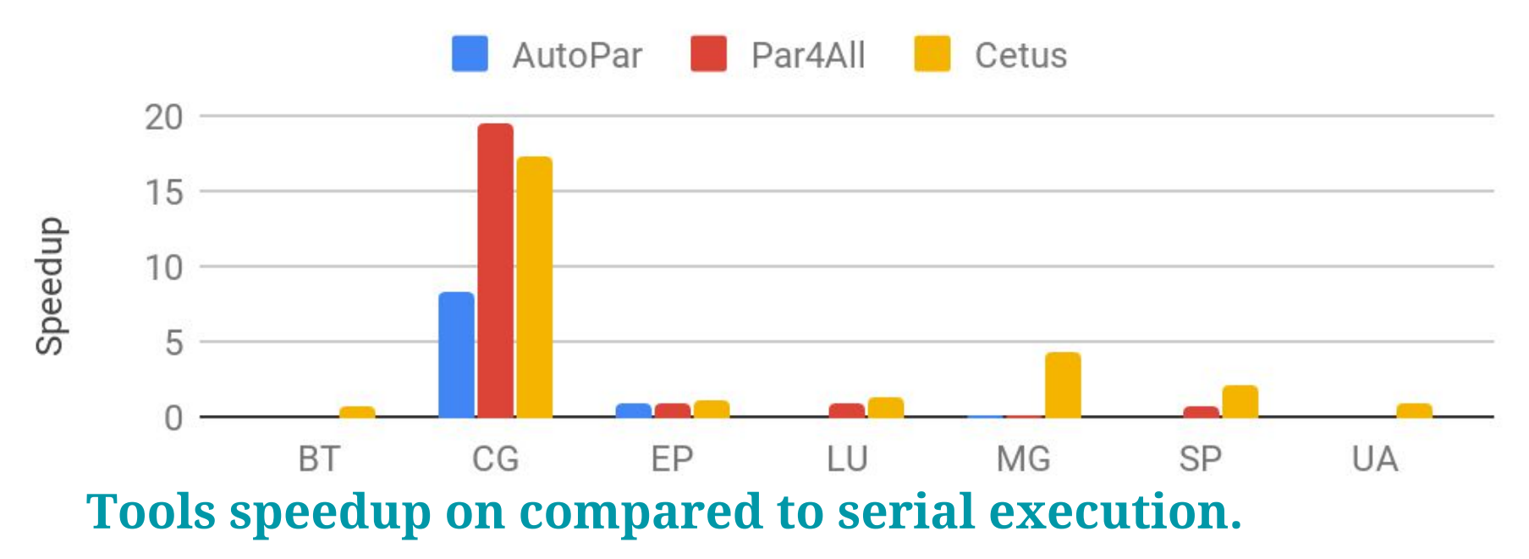
To further evaluate the tools capabilities, we introduce the Numerical Aerodynamics Simulations (NAS) Parallel Benchmarks.

The benchmarks that were included to evaluate the tools are

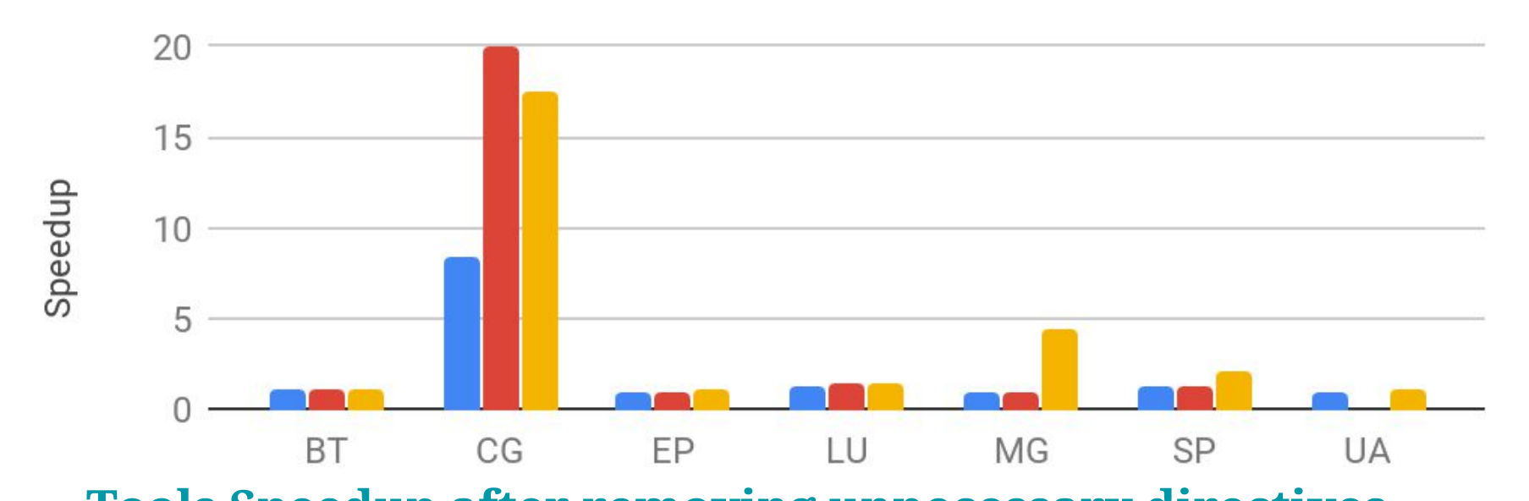
- Block Tri-diagonal solver (BT)
- Conjugate Gradient (CG)
- Embarrassingly Parallel (EP)
- Integer Sort (IS)
- Lower-Upper Gauss-Seidel solver (LU)
- Multi-Grid (MG), Scalar
- Penta-diagonal solver (SP)
- Unstructured Adaptive mesh (UA)

The benchmarks Fourier Transform (FT) and Integer Sort (IS) were excluded from this study due to the inability of AutoPar and Par4All to process them.

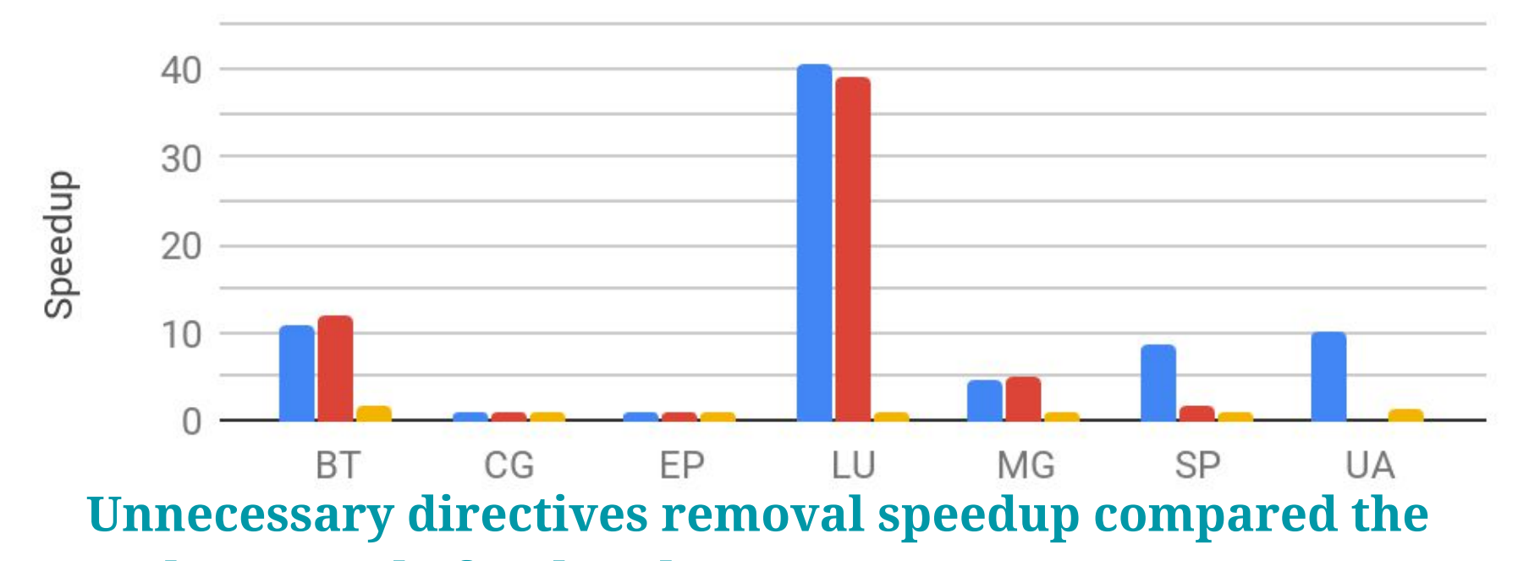
To show a scenario which involves the tools' output and minimal human intervention, the unnecessary OpenMP directives, created by the three tools were manually removed.



Tools speedup on compared to serial execution.



Tools Speedup after removing unnecessary directives compared to serial execution.



Unnecessary directives removal speedup compared the tools output beforehand.

Conclusion

AutoPar

- AutoPar’s Annotation file is a powerful tool, especially for function side effect, structures and object oriented programming, which makes it more suitable for bigger or OOP-based projects.
- AutoPar's No-Aliasing option should be used with caution.

Par4All

- Older, but still a forceful tool that can handle most cases automatically with minimal user intervention.
- Supports Fortran, making it more suitable for scientific legacy codes.

Cetus

- Its ability to take loop size into account makes it a very powerful tool.
- Generates reduction clauses on arrays. However, this reduction clause may be invalid.

Acknowledgments: This work was supported by the Lynn and William Frankel Center for Computer Science. Computational support was provided by the NegevHPC project.