

Performance study of detector alignment software

Estefania Serrano
Supervisor:
Pawel Szostek

> **26/08/2015**

Millepede II

- › **Least square fit application.**
- › **Written in Fortran.**
- › **A lot of methods for solving equation systems.**
 - Most of them memory optimized . -> Medium problem 25 GB on memory
 - Computationally expensive. -> Whole week -end of execution
- › **Already using parallelization.**
 - OpenMP in some sections



<http://www.desy.de/~kleinwt/MP2/doc/html/index.html>



<http://www.desy.de/>

Is it possible to optimize it?

First steps: Characterization

- > Intel Compiler adaptation.
- > Study of the hotspots:

Intel VTune

Function Call Sites and Loops	Loop	Annotations	Recommendations	Compiler Diagnostic Details	Vectorized Loops	Instruction Set Analysis	Advanced					
Function Call Sites and Loops	Total Time %	Total Time	Self Time	Loop Type	Why No Vectorization?	Vector	Vector Length	Compil.	Tails	Data Types	Vector Widths	Advanced
*Total	100.0%	71716.9	0s									
*clone	98.5%	70636.9	0s									
*main_thread	98.5%	70636.9	0s									
*[OpenMP worker]	98.5%	70636.9	0s									2,14
*[Loop in [OpenMP worker]]	98.5%	70636.9	0s	Scalar								2,14

Intel Advisor

Elapsed Time: 670.229s

Checks: 51,620,893,731,224
 Instructions/Byte: 59,506,038,363,893
 CPU Cycles: 4,877

Filtered Pipeline Slots

- Function: 0.301
- Bad Speculation: 0.032
- Unfilled Pipeline Slots (Branch): 0.002
- Back-End Bound: 0.002
- Memory Bound: 0.024
- L1 Bound: 0.004
- L3 Bound: 0.001
- Cache/Access: 0.001
- Data Stream: 0.000
- L1 Latency: 0.021
- DRAM Bound: 0.000
- Local DRAM: 0.000
- Memory DRAM: 0.000
- Memory Cache: 0.012
- Store Bound: 0.007
- Core Bound: 0.267
- Front-End Bound: 0.069

Back-End Bound: 0.002

Identify slots where no cycle was delivered due to a lack of required resources for executing next slots in the back-end of the pipeline. Back-end metrics describe a portion of the pipeline where the out-of-order scheduler dispatches ready slots into their respective execution units, and, once completed, these slots get retired according to program order. Slots due to data-cache misses or stalls due to the overloaded divider unit are examples of back-end bound issues.

Memory Bound: 0.024

This metric shows how memory subsystem issues affect the performance. Memory bound measures a fraction of cycles where operations could be stalled due to demand load or store instructions. This occurs mainly for instructions in flight imply back-pressure on the pipeline.

L1 Bound: 0.004

This metric shows how often machine was stalled without missing the L1 data cache. The L1 cache typically has the shortest latency. However, in certain cases like branch mispredict on other stores, a load might suffer a high latency even though it is being satisfied by the L1.

L3 Bound: 0.001

Cache/Access: 0.001
Data Stream: 0.000
L1 Latency: 0.021

DRAM Bound: 0.000

Local DRAM: 0.000
Memory DRAM: 0.000
Memory Cache: 0.012

Store Bound: 0.007

This metric shows how store-to-memory issues limit the performance when you run out of DDD resources or are saturating certain execution units (for example, using 19-chained long-latency arithmetic operations).

Core Bound: 0.267

The number of cycles during which no part was utilized.

Front-End Bound: 0.069

Cycles of 1-Port Utilized: 0.036
Cycles of 2-Port Utilized: 0.235
Cycles of 3-Port Utilized: 0.237
Cycles of 4-Port Utilized: 0.172

Vectorization

› Compiler pragmas

```
!DIR$ ASSUME_ALIGNED A: 64, V: 64
```

› Instruction set changes

```
-axAVX2
```

Obvious changes for the programmer...

Not so obvious for the compiler!!!

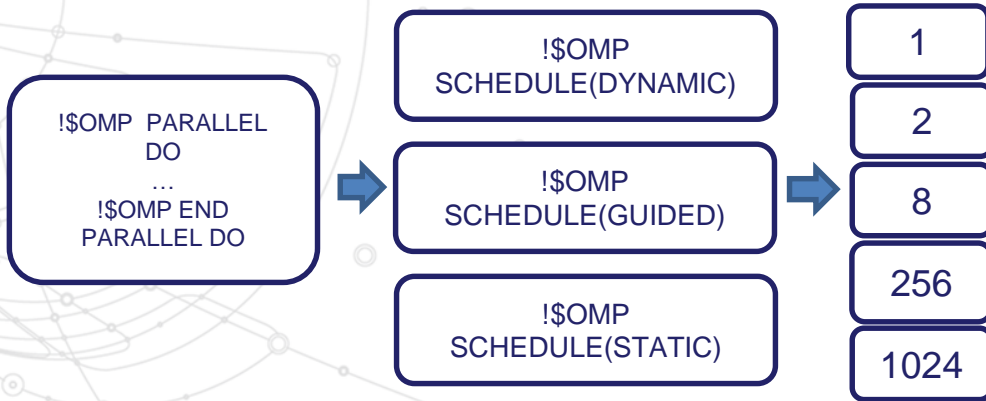
› Code modifications

```
b(j)=b(j)+globalMatD(offset+j)*x(j)
```

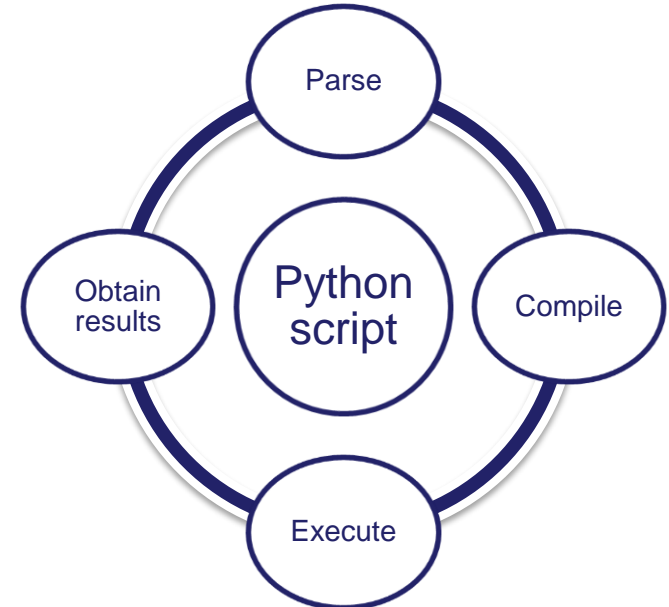


```
b(j)=b(j)+globalMatD(newj)*x(j)
```

- > **Parallelized sections with high imbalance**
 - Some threads get more work than others
- > **Optimization space: 170 executions**



Parallelization



› Coprocessor = big number of cores

Not a guarantee of better performance!!

Requirements:

Good vectorization and alignment ✓

Small sequential sections

Small memory usage

Not much use of I/O



Initial experiments:

Xeon processor:	200 s
Xeon Phi:	2000 s

10x slower!
Still work to do

www.clipartbest.com

<http://nexnet.files.wordpress.com/2013/02/kliponious-black-tick.png>

- › **Some room for optimization**
 - Medium problem: 1900 s to 1700 s
- › **Advises for the future:**
 - Modern programming language
 - Optimized mathematical libraries

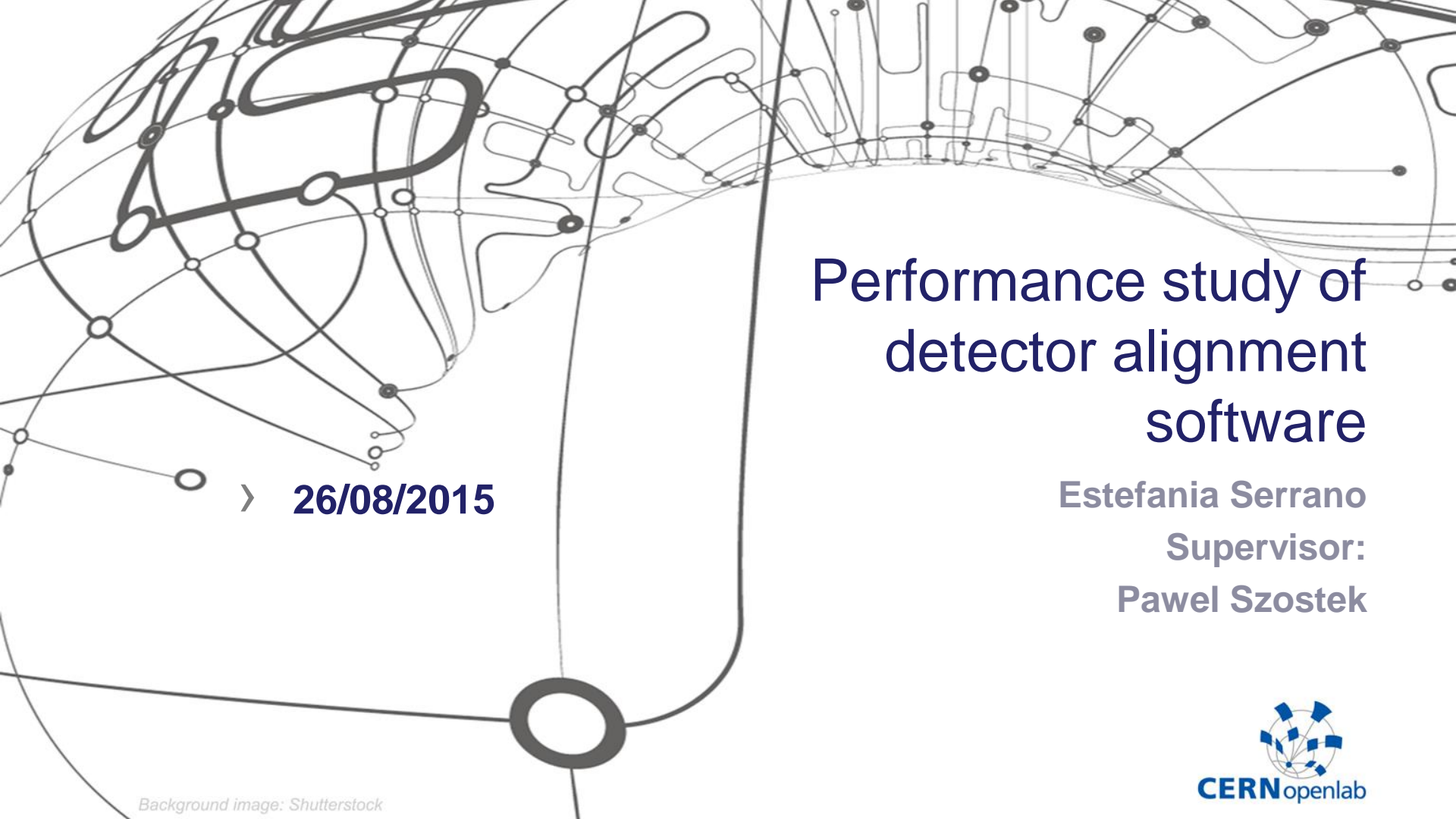
“It is much easier to create an optimized application than optimizing an already created one.”

Conclusion

Even fixed some bugs!!



http://pngimg.com/upload/bug_PNG3980.png



Performance study of detector alignment software

Estefania Serrano
Supervisor:
Pawel Szostek

> **26/08/2015**