Advanced OpenMP Constructs, Tuning, and Tools at NERSC

Ahana Roy Choudhury[1], Yun (Helen) He[2], and Alice Koniges[2]
[1]Computer & Information Sciences Department, University of Alabama at Birmingham
[2]NERSC, Lawrence Berkeley National Laboratory, Berkeley, CA

THE UNIVERSITY OF ALABAMA AT BIRMINGHAM
Knowledge that will change your world

Abstract

NERSC's next generation supercomputer systems, e.g., Cori Phase 2 with KNL (Intel Knights Landing) architecture, have a large number of cores per node, so, it is important to consider advanced OpenMP concepts in order to achieve optimal performance on such systems.

In this project, we have written and implemented code snippets and used them to test a variety of tools for improving OpenMP performance and detailed their usage on various NERSC systems including KNL. We have explored the detection of issues such as false sharing and data races using tools and how they can be resolved. We have also explored advanced OpenMP concepts such as process and thread affinity and nested OpenMP.

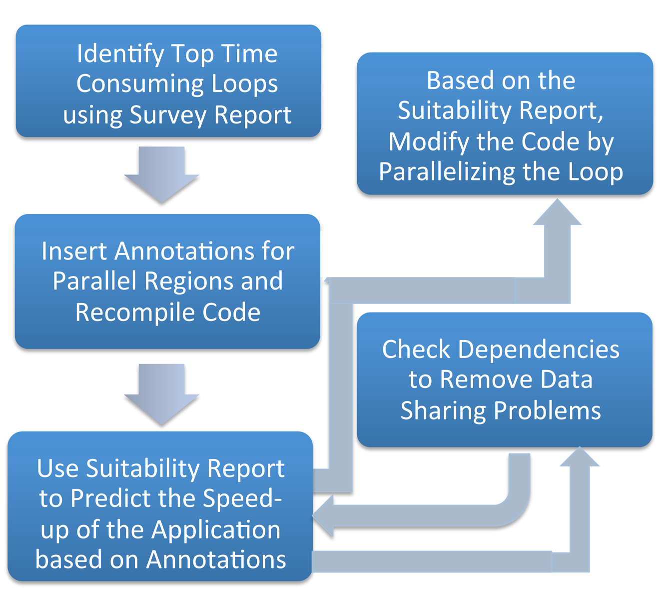
Questions and Challenges

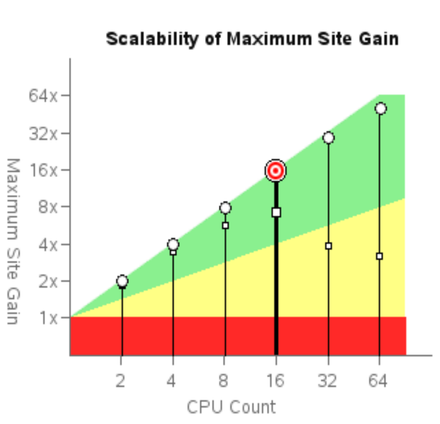
- Which part of the code can be safely parallelized?
- How to detect and avoid data races?
- How to detect and avoid memory leaks?
- How can tools be used to get suggestions on variable scope and OpenMP compiler directives?
- How to detect top time-consuming loops?
- How to detect and remove false sharing?
- Ensuring desirable process and thread affinity
- Using Nested OpenMP

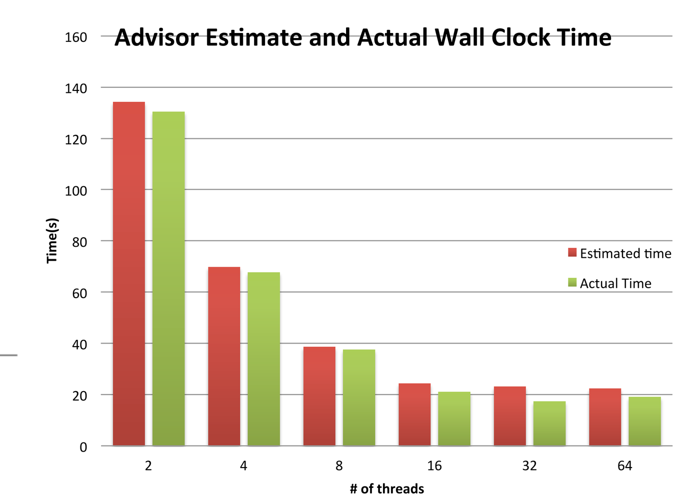
Improving Performance of Sample Codes using Tools

Threading Design using Intel Advisor

- Intel Advisor helps to ensure that Fortran, C and C++ applications take full performance advantage of today's processors.
- Threading Advisor is a threading design and prototyping tool that helps to analyze, design, tune, and check threading design options without disrupting normal development.



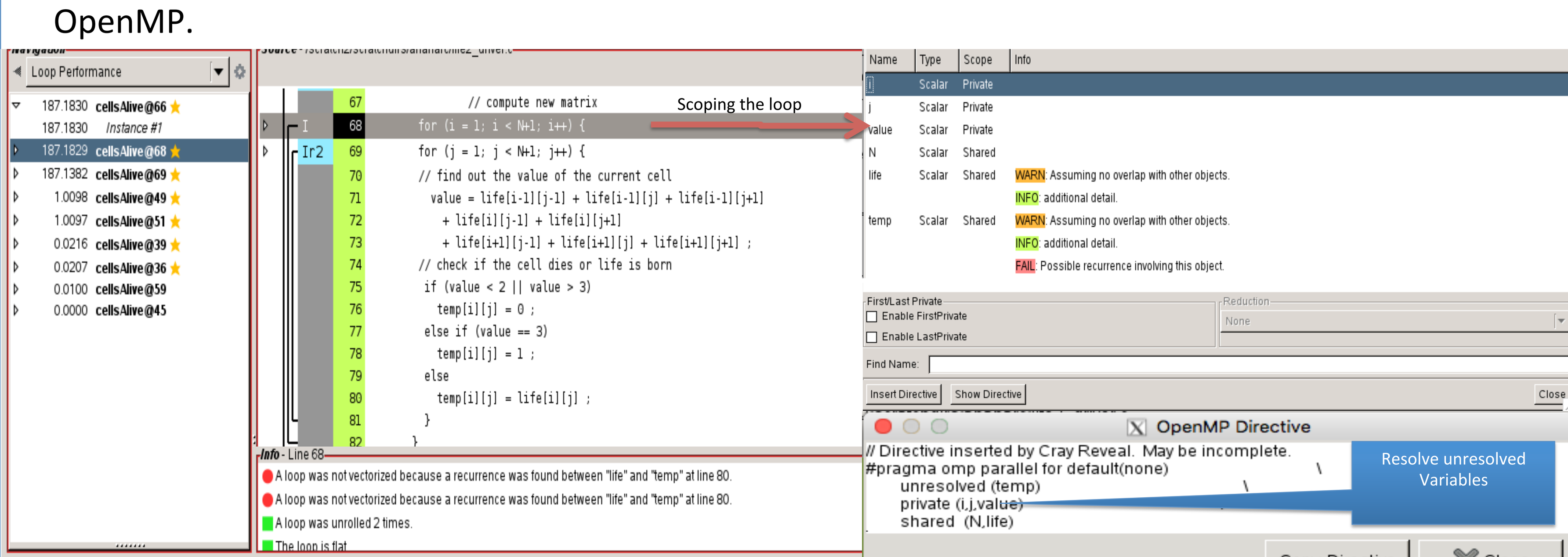




Comparison between Advisor estimated and measured wall clock times. The % variation range is 3-15% and increases with increasing numbers of threads.

Parallelizing Codes using Cray Reveal

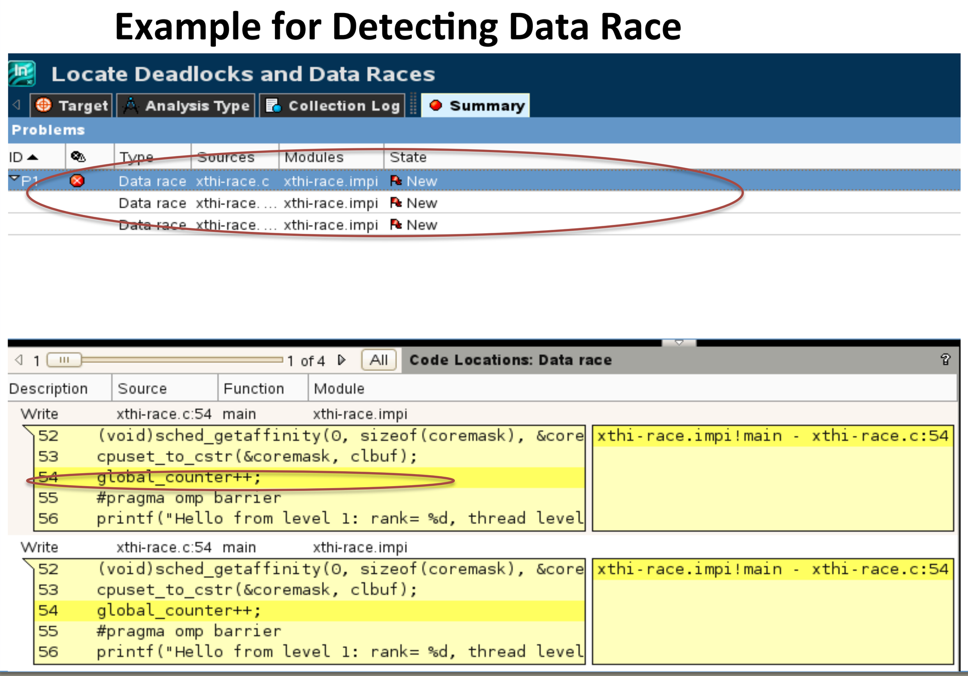
- Reveal is a tool developed by Cray that is part of the Cray Perftools software package.
- Helps to identify top time-consuming loops, dependencies and vectorization.
- Loop scope analysis provides variable scope and compiler directive suggestions for inserting OpenMP.



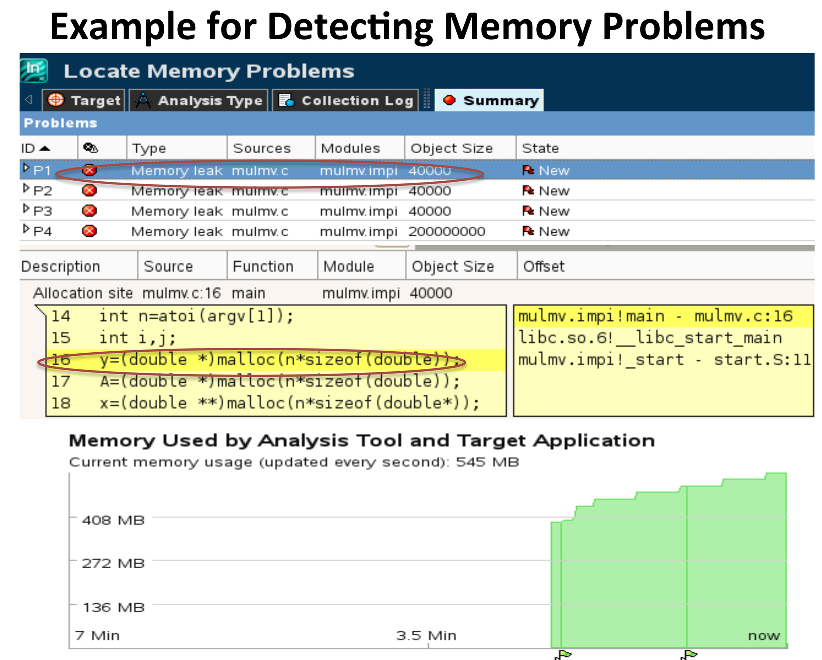
Threading and Memory Error Detection using Intel Inspector

- Intel Inspector is a dynamic memory and threading error checking tool for users developing serial and multithreaded applications.

Example for Detecting Data Race



Example for Detecting Memory Problems

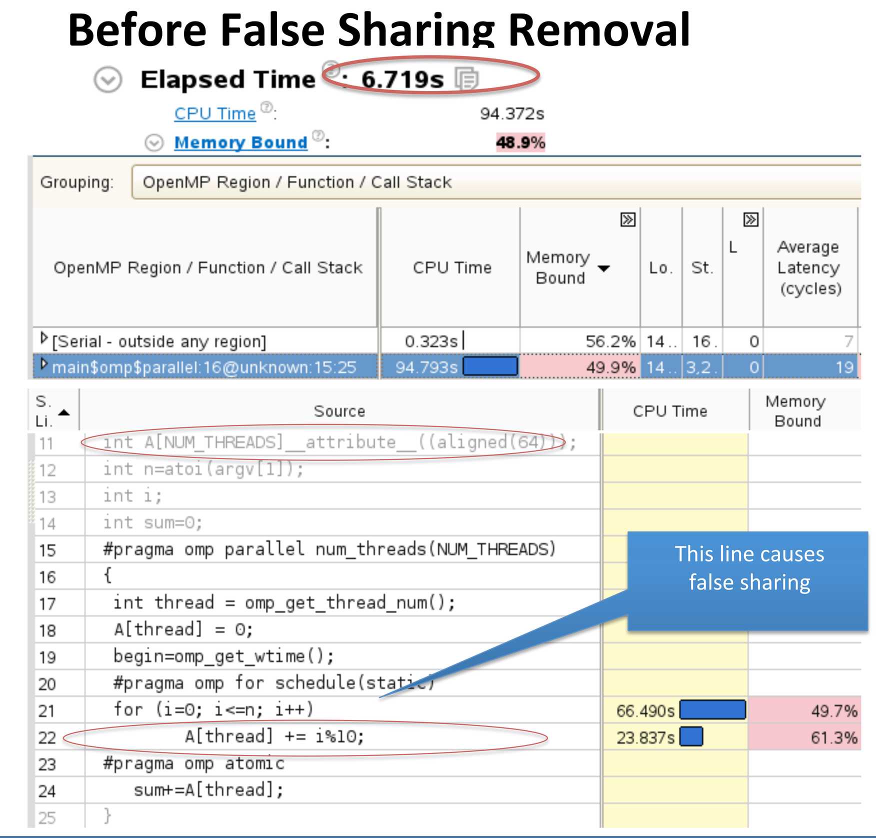


Improving Performance of Sample Codes using Tools (continued)

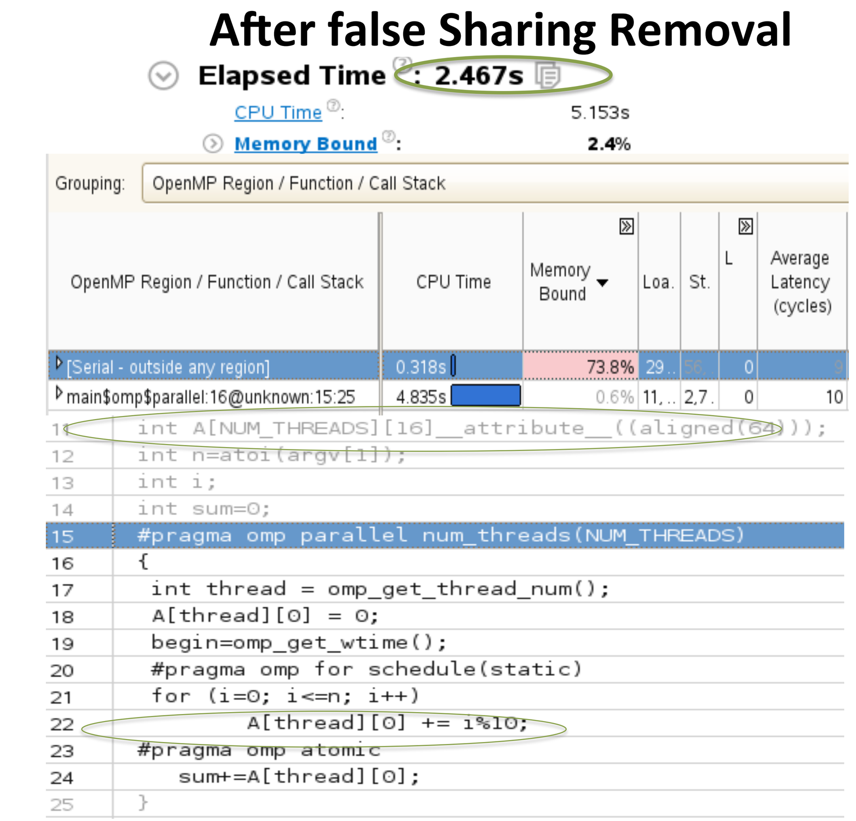
Threading Design using Intel Advisor

- **False sharing** occurs when threads on different processors attempt to modify variables that reside on the same cache line.
- It causes performance degradation due to coherence issues and should be avoided.
- **Intel VTune Amplifier** is a performance analysis tool that helps to analyze the algorithm and identify where and how applications can benefit from available hardware resources.
- To detect false sharing using VTune Amplifier, we wrote a code that causes false sharing, detected the problem and then resolved it using padding. We also noted that some compiler optimization choices remove false sharing.

Before False Sharing Removal



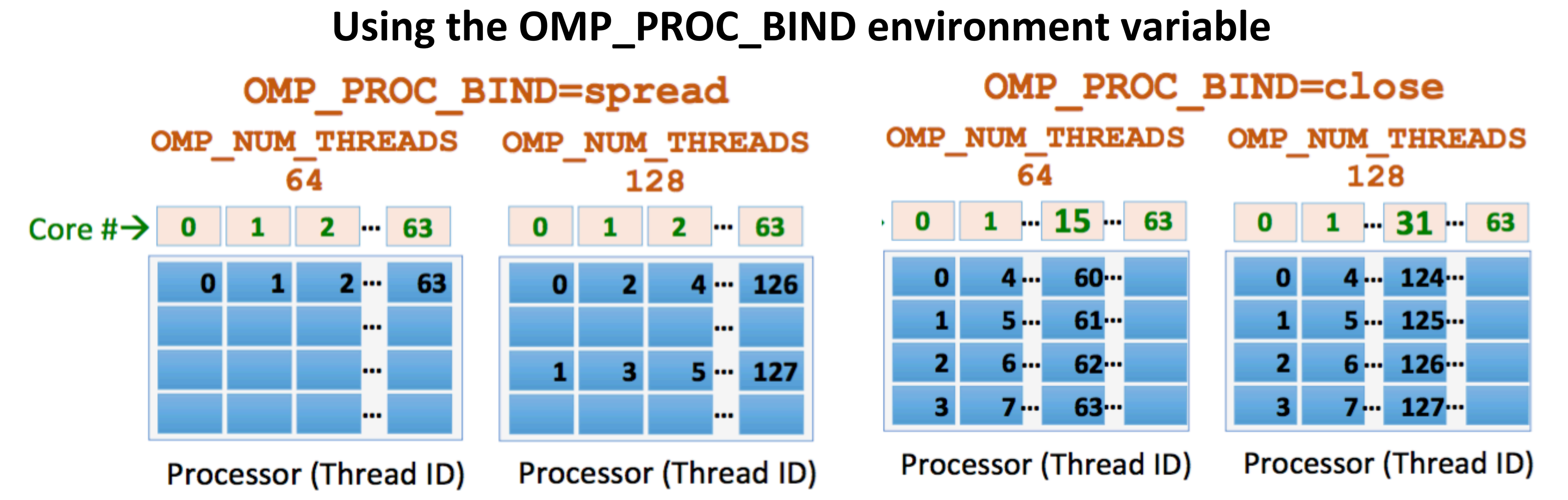
After false Sharing Removal



Process and Thread Affinity

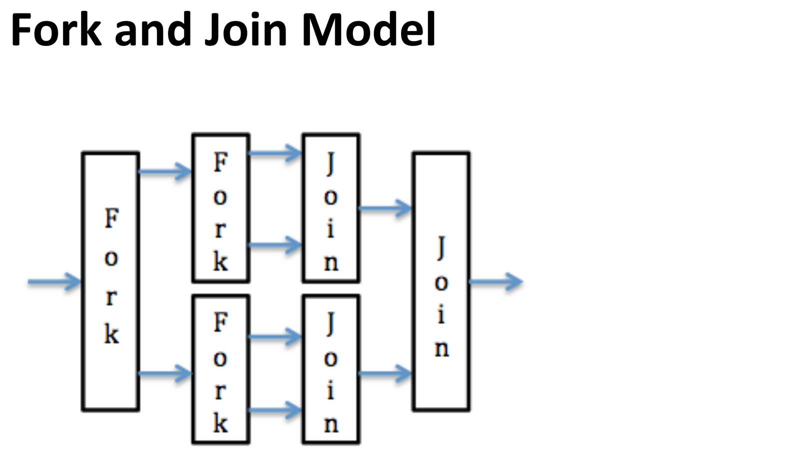
- Thread affinity binds each process or thread to run on a specific subset of processors, to take advantage of memory locality.
- Improper process/thread affinity could slow down code performance significantly.

Using the OMP_PROC_BIND environment variable



Nested OpenMP

Fork and Join Model



When to use Nested OpenMP

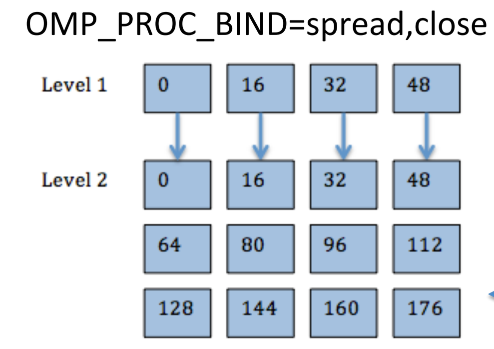
To achieve more fine-grained thread parallelism:

- When the top level OpenMP loop does not use all available threads
- When multiple levels of OpenMP loops are not easily collapsed
- For certain computation intensive kernels
- For multi-threaded MKL (Intel Math Kernel Library)

Thread Affinity for Nested OpenMP

OMP_NUM_THREADS=4,3

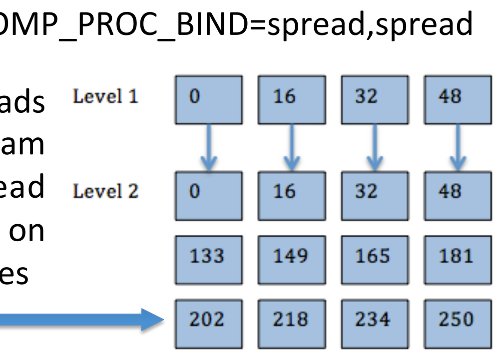
OMP_PROC_BIND=spread,close



Level 2 threads in same team bind to threads on same core

OMP_PROC_BIND=spread,spread

Level 2 threads in same team evenly spread on threads on different cores



Conclusions

In order to achieve good performance in OpenMP codes, it is important to use advanced OpenMP concepts like process and thread affinity and nested OpenMP. It is equally crucial to avoid false sharing and over-subscription. We have explored how tools can be used to facilitate the process of tuning OpenMP codes as well as worked on thread and process affinity and nested OpenMP. Future work involves using nested OpenMP to speed up full applications.

References

1. https://computing.llnl.gov/tutorials/openMP/
2. http://www.eweek.com/c/a/Application-Development/Oracle-and-Java-7-The-Top-10-Developer-Features-626145
3. **Using OpenMP** by Barbara Chapman, Gabriele Jost, Ruud van der Pas, The MIT Press, MIT, 2008.
4. https://software.intel.com/en-us/get-started-with-advisor-threading-linux
5. https://software.intel.com/en-us/intel-inspector-xe
6. https://www.olcf.ornl.gov/wp-content/uploads/2013/02/Cray_Reveal-HP1.pdf
7. http://www.nersc.gov/users/computational-systems/cori/application-porting-and-performance/improving-openmp-scaling/
8. https://software.intel.com/en-us/articles/avoiding-and-identifying-false-sharing-among-threads
9. https://software.intel.com/en-us/articles/finding-your-memory-access-performance-bottlenecks
10. https://www.nersc.gov/assets/Uploads/Nested-OpenMP-NUG-20151008.pdf
11. http://www.nersc.gov/users/training/events/advanced-openmp-training-february-4-2016/
12. https://drive.google.com/a/lbl.gov/file/d/0B9D5EnxRqcaZalR5WEh6bkhxNGs/view