UNIVERSITY of HOUSTON
DEPARTMENT OF COMPUTER SCIENCE

False sharing detection for OpenMP applications using OMPT API

Millad Ghani, **Abid M. Malik**, Barbara M. Chapman, and Ahmad Qawasmeh

HPCTools Group, University of Houston, TX

Agenda

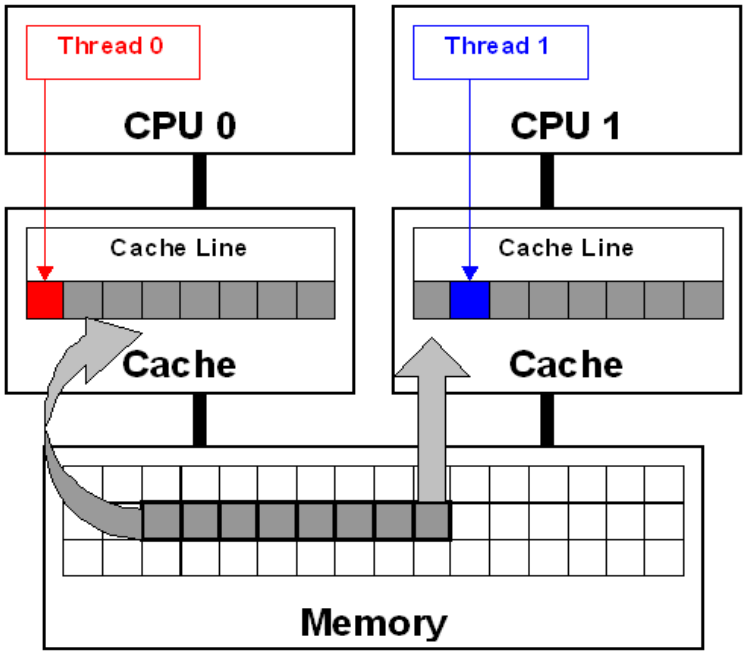
- Introduction to the problem
- Introduction to machine learning
- OMPT (OpenMP Tools) API
- Cost Modeling using OMPT API
- Experimentation and results
- Future work
- Conclusion

Introduction

- Parallel programming is unavoidable in the era of the multicore
- OpenMP simplifies shared memory parallel programming
  - More productivity
  - New classes of correctness and performance bugs
- Scalability can be a challenge
  - Data locality
  - False sharing
- Needs a deep understanding about program's dynamic behavior

False sharing effect

- **False sharing**: It occurs when threads on different processors modify variables that reside on the same cache line, as illustrated in the figure
- Ping-Pong effect on cache-line (due to cache-coherency protocol)



False sharing: program example

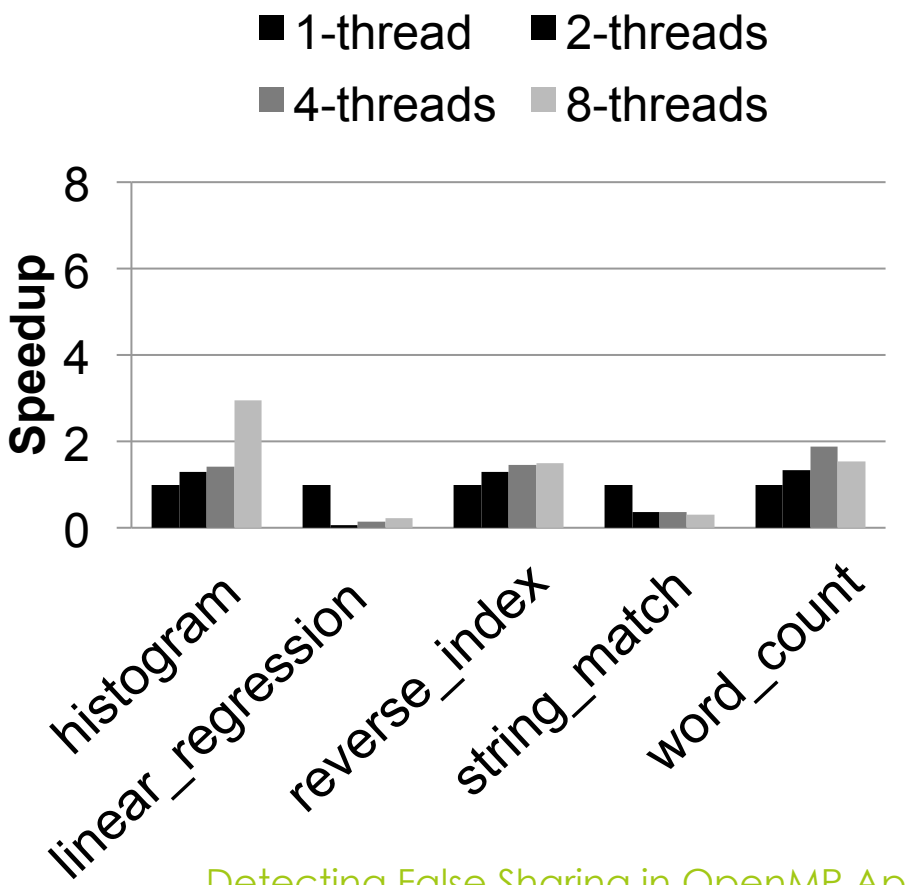
```
int *local_count = (int*)malloc(sizeof(int)*NUM_THREADS*PADDING);
int *vector = (int*)malloc(sizeof(int)*VECTOR_SIZE);
for(i=0;i<COUNT;i++)
{
#pragma omp parallel
    {
        int tid = omp_get_thread_num()*PADDING;
        if(tid < 0) tid = 0;

        #pragma omp for
        for(j = 0; j < VECTOR_SIZE; j++)
            local_count[tid] += vector[j]*2;

        #pragma omp master
        {
            int k;
            for(k = 0; k<NUM_THREADS; k++)
            result += local_count[k];
        }
    }
}
```

Code version	1-thread	2-thread	4-thread	8-thread
Without padding	0.503	3.763	3.961	4.432
With padding	0.503	0.263	0.137	0.078

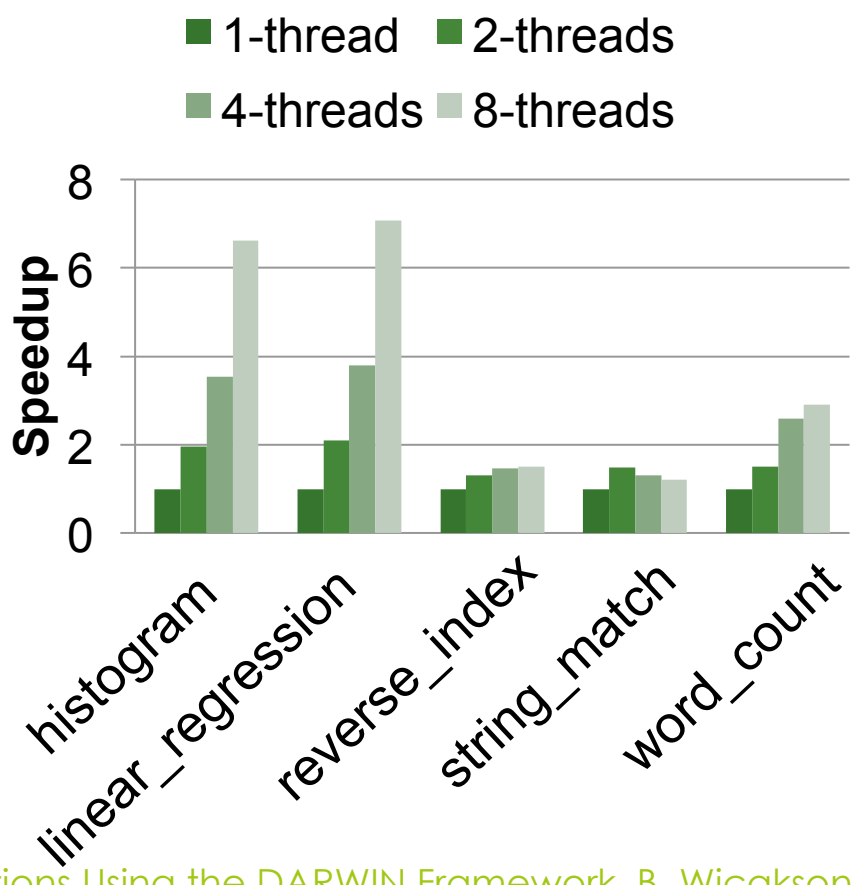
Execution time improvement

False sharing: program example

Original Version



Optimized Version



Detecting False Sharing in OpenMP Applications Using the DARWIN Framework. B. Wicaksono, M. Tolubaeva, Barbara Chapman, LCPC 2011.

Detecting false sharing is hard

- The program is functionally correct
  - Only running much slower than possible
  - Major class of bugs
- There is no sharing at the program level
  - Two interfering variables that share a cache line are independent with no visible relationship
  - Program analysis will not find it
- Happens due to interaction among core
  - Looking within a single core does not reveal the problem

Recent work

- Detecting False Sharing in OpenMP Applications Using the DARWIN Framework. B. Wicaksono, M. Tolubaeva, Barbara Chapman, LCPC 2011.
- Dynamic cache contention detection in multi-threaded applications. Qin Zhao, David Koh, Syed Raza, Derek Bruening, Weng-Fai Wong, VEE 2011.
- SHERIFF: precise detection and automatic mitigation of false sharing. Tongping Liu Emery D. Berger, OOPSLA 2011.
- Detection of false sharing using machine learning. Sanath Jayasena et al, SC13.

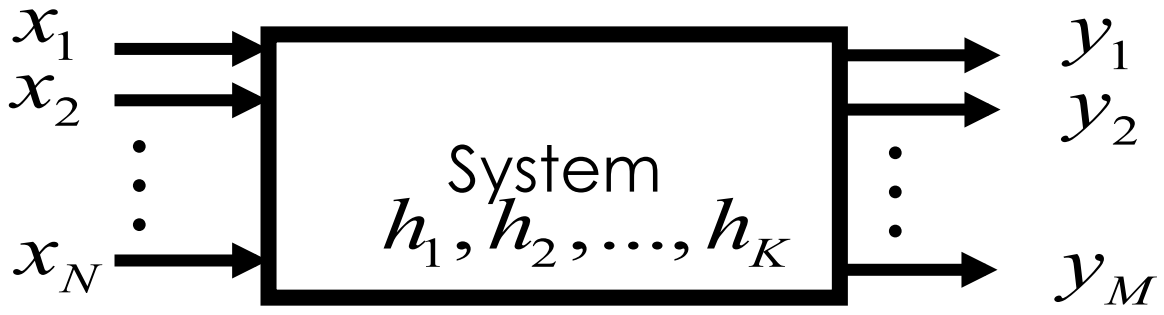
Our approach

- We use machine learning to analyze hardware performance event data using OMPT API
- We treat it as "**Binary Classification Problem**": *Given an OpenMP code and given number of threads, do we have a false sharing effect or not?*
- Basic idea: train a classifier with data from *OpenMP* mini-programs
- Separate classifiers for each number of threads
- Develop a set of mini-programs, with two possible modes of execution
  - Good (no false sharing)
  - Bad (with false sharing)

Machine learning

- Machine Learning (ML) is constructing computer programs that develop solutions and improve with experience
- Solves problems which can not be solved by enumerative methods or calculus-based techniques
- Intuition is to model human way of solving some problems which require experience
- When the relationships between all system variables is completely understood ML is not needed

A generic machine learning system



Input Variables: $\mathbf{x} = (x_1, x_2, \ldots, x_N)$

Hidden Variables: $\mathbf{h} = (h_1, h_2, \ldots, h_K)$

Output Variables: $\mathbf{y} = (y_1, y_2, \ldots, y_K)$

Feature selection for machine learning

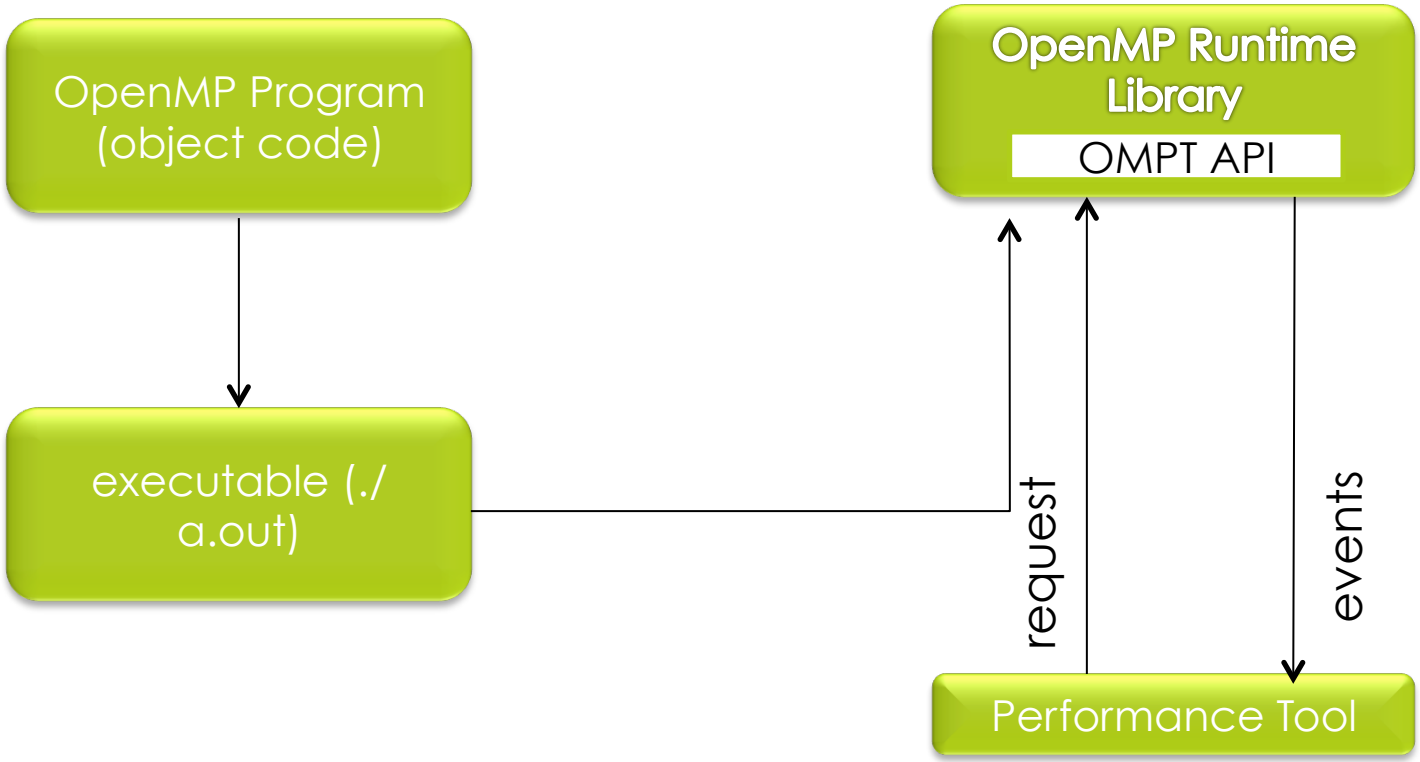
- The search space depends upon the number of input variables
- Large number of features means complex search space and hard job for a machine learning model to converge or learn
- We need to reduce the dimension of the search space in order to make the learning easy and effective
- Feature selection is a big research topic in statistical learning
- We use OMPT API to collect the features for machine learning for false sharing

OMPT: OpenMP Tools Application Programming Interface **

- OMPT API is designed to allow performance tools to gather useful performance and debugging information from applications and to hide low-level OpenMP implementation from users.
- OMPT API is based on experience with two prior efforts to define a standard OpenMP tools API: the POMP API and the Sun/Oracle Collector API
- OMPT API is a minimal set of features to support tools that employ asynchronous sampling to measure application performance

**Detailed report on OMPT API available on www.openmp.org

OMPT : OpenMP Tools Application Programming Interface



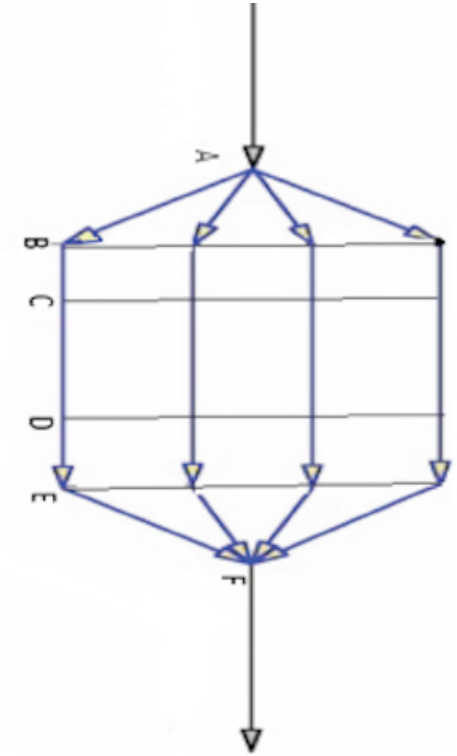
Currently, OpenUH and Intel OpenMP runtime libraries support OMPT API

OMPT: OpenMP Tools Application Programming Interface

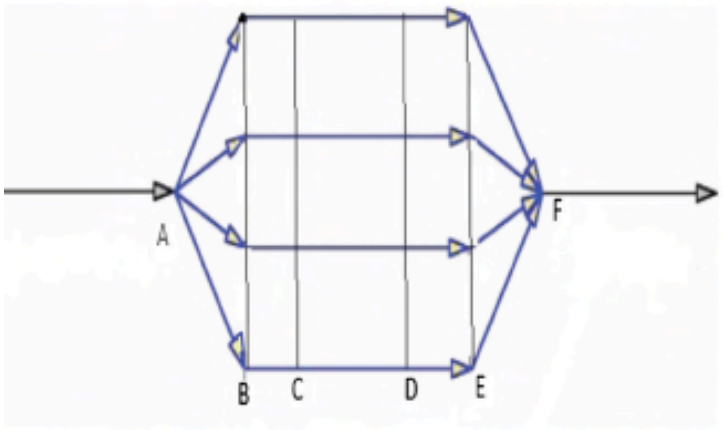
- The main goal behind the effort is to fill the gap for performance analysis for OpenMP programs
- This will help understand the performance bottlenecks for OpenMP programs for the future complex hardware
- The framework has the ability:
  - To build runtime adaptable approaches for OpenMP programs
  - To build cost models for various OpenMP performance challenges

Cost modeling using OMPT API

- OMPT API is used by a performance tool to capture OpenMP events
- Successive events give a user a window of opportunity to collect information
- The information can give a pattern that can be used for cost modeling



Representation for cost modeling

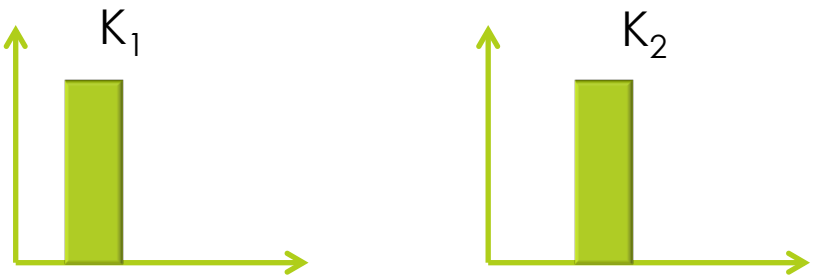


(a) Point B, C, D and E are fine level OpenMP events. Point A and F are coarse level OpenMP events.

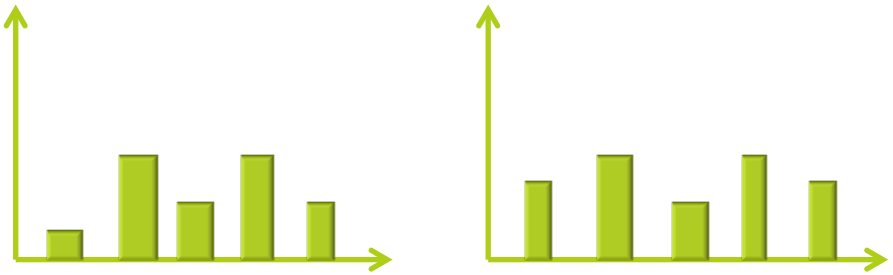
(b) Energy pattern for the Strassen application using OMPT [17].

Anilkumar Nandamuri, Abid M. Malik, Ahmad Qawasmeh and Barbara M. Chapman. "Power and Energy Footprint of OpenMP programs using OpenMP Runtime API". In 2nd International Workshop on Energy Efficient Super Computing (E2SC) held in conjunction with SC14, November, 2014, New Orleans, Louisiana, USA

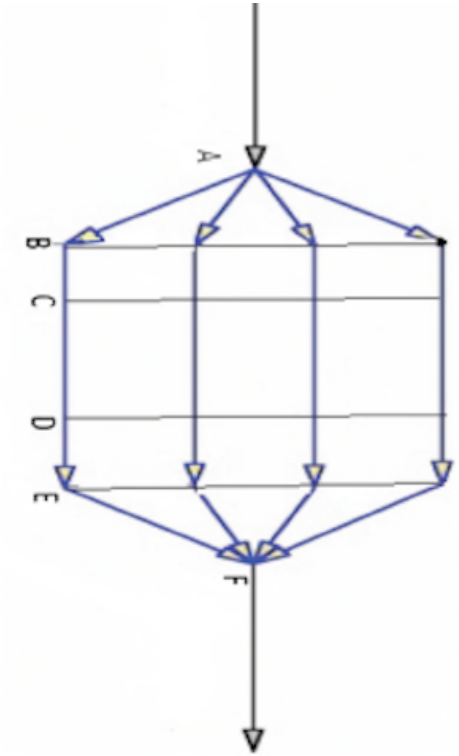
Cost modeling using OMPT API



Information at coarse level



Information at fine level using OMPT API



Feature vector representation for machine learning using OMPT API

$V_{AB} = \{f_1, f_2, f_3, \ldots, f_m\}$

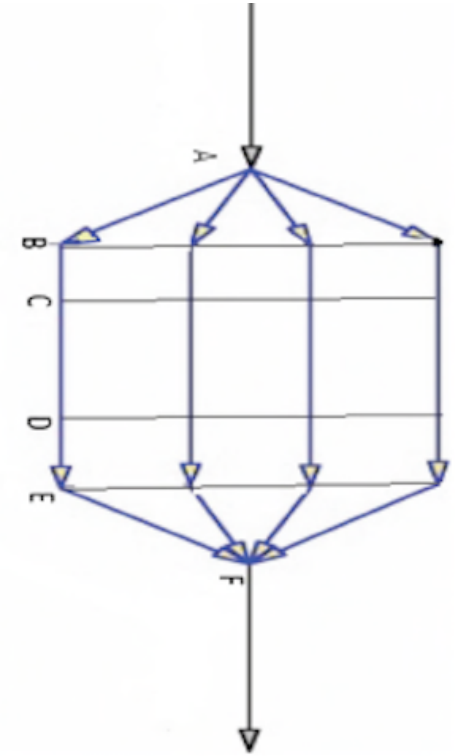
$V_{BC} = \{f_1, f_2, f_3, \ldots, f_m\}$

$V_{CD} = \{f_1, f_2, f_3, \ldots, f_m\}$

$V_{DE} = \{f_1, f_2, f_3, \ldots, f_m\}$

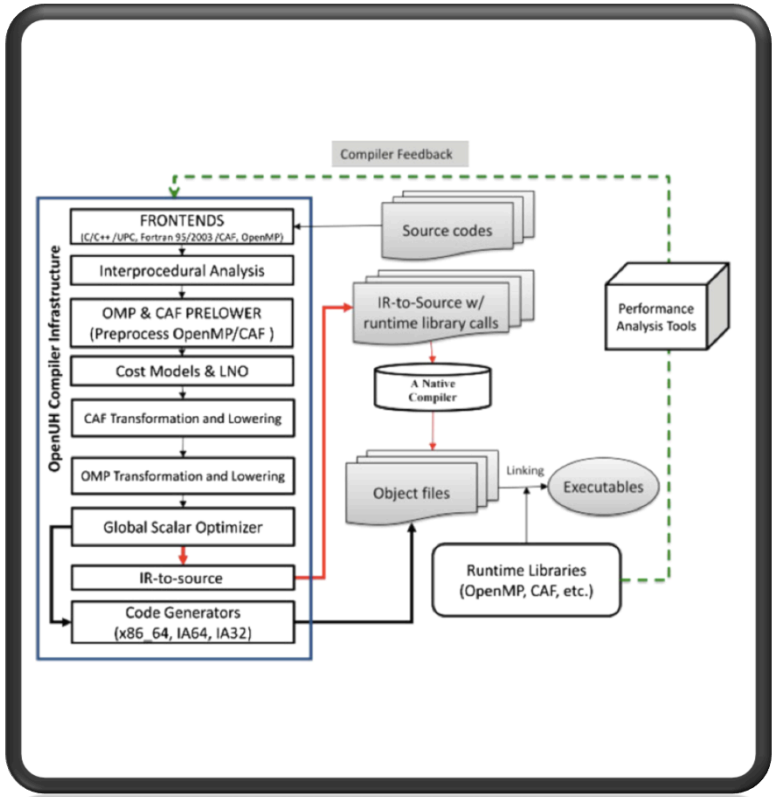
$V_{EF} = \{f_1, f_2, f_3, \ldots, f_m\}$

$V_{final} = \{V_{AB}, V_{BC}, V_{CD}, V_{DE}, V_{EF,}$ CLASS $\}$



Experimentation

- OpenUH compiler framework
- Processor: 2.4 GHz quad-core Intel Xeon E5-2665
- Number of Sockets = 2
- No. of Cores: 8 per Socket
- Main Memory: 8GB per Socket
- L1 64K, L2 256K, and cache line size is 64 bytes
- Built a tool to collect the information using OMPT APIs

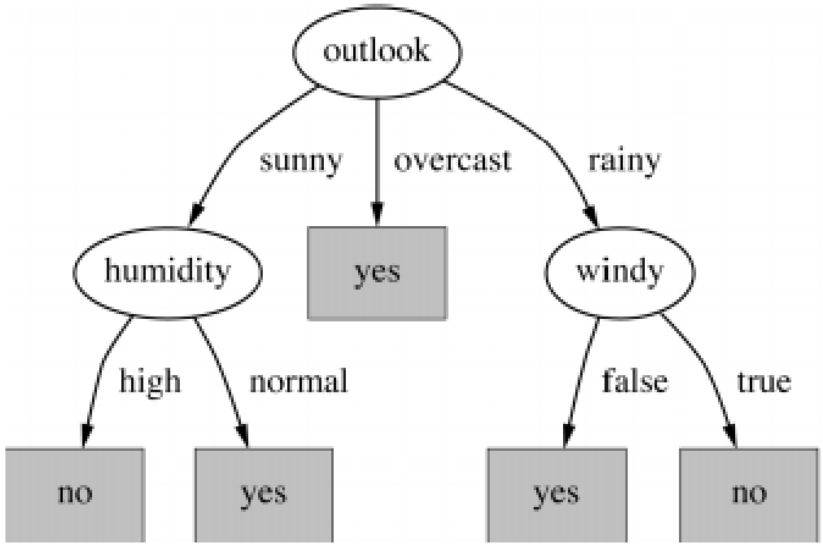


Experimentation

- Use the NAS parallel benchmarks to produce the training and testing instances
- Use 40 OpenMP kernels for training and testing
- Introduce the false sharing in the instances with different number of threads
- WEKA framework from the University of Waikato
- Tried various learning algorithms including SVM, ANN, K-mean clustering, and decision tree
- The decision tree approach gives the best performance in terms of accuracy
- Producing data for training phase is the most expensive parts

Decision tree algorithm

- Nodes of trees are attributes and the leaves are classes
- Use information gain statistical criterion to select the features
- Use heuristic (C4.5) to establish relationship among the features
- Features' values are used to establish the relationship
- Values can be discrete or continuous

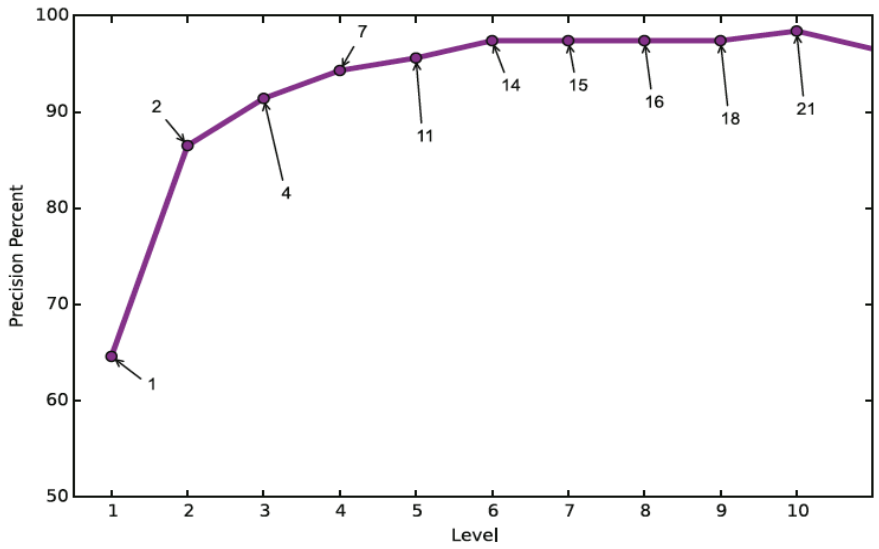


Machine Learning by Tom Mitchell

Feature selection for false sharing

- Collected hardware features using PAPI hardware counters
- Use Mutual Information (MI) criterion to select the best features
- The decision tree algorithm uses MI to order hardware features
- Best result is given by the top 12 features

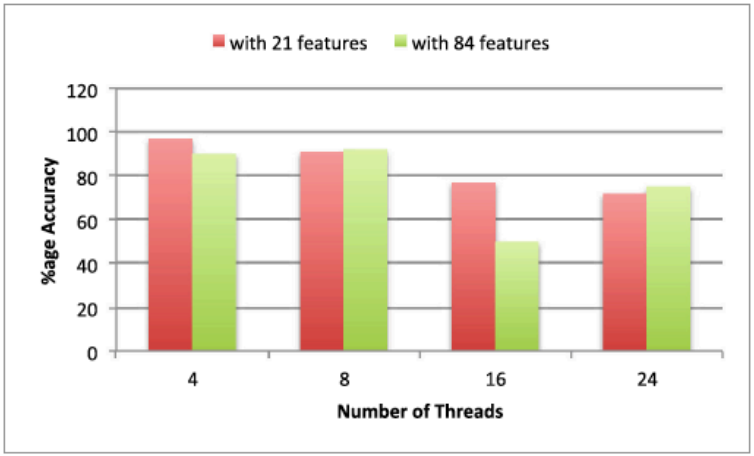
Accuracy of the classifier



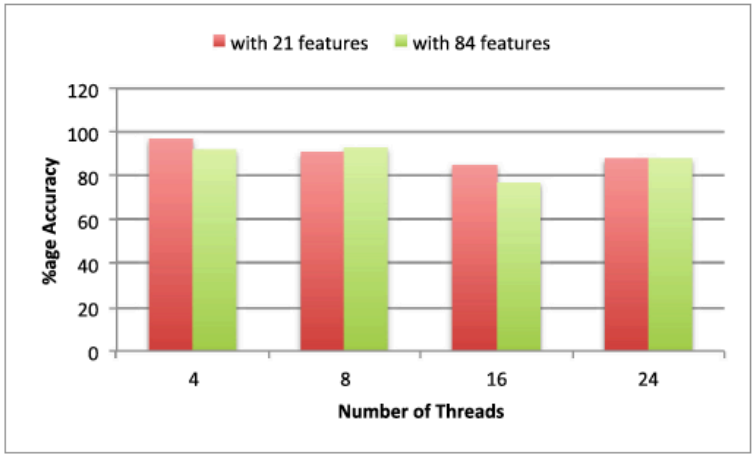
- 12 features give the best solution
- Increase in the number of features doesn't affect the accuracy of the classifier
- The result is consistent with the previous work

Results and analysis

Efficiency of the classifiers using 10 cross validation approach using the training phase



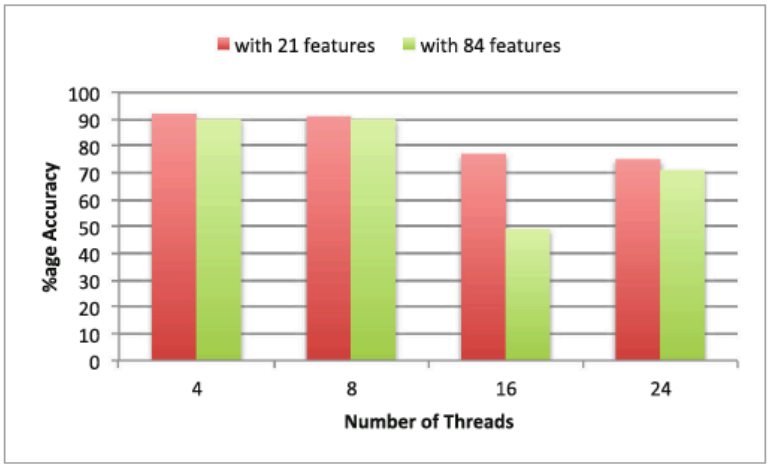
(a)



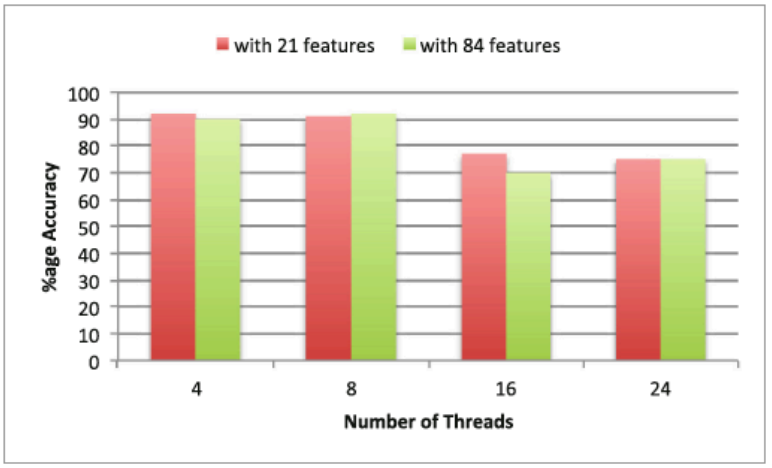
(b)

Results and analysis

Efficiency of the trained classifiers using unseen OpenMP kernels



(a)



(b)

Future work

- Improve the accuracy performance
- Feature vector needs improvement
- Validation using a real application
- Combine this approach with dynamic false sharing detecting approaches to lower its overhead
- OMPT API for other performance optimizations

Conclusions

- False sharing can seriously degrade performance
- We presented an effective and low over-head approach for detecting false sharing in OpenMP programs using OMPT API
- Use NAS Parallel benchmarks to validate our approach
- The accuracy of our approach is from 60 to 90%
- OMPT API has the ability to built effective cost models for OpenMP performance challenges

Acknowledgement

- This work is supported by the National Science Foundation under grant CCF-1148052

Thank you!

- Questions!