ParaTools ThreadSpotter

ParaTools, Inc.

John C. Linford et al.
jlinford@paratools.com

Army HPC User Group Review
18 May 2017, ARL

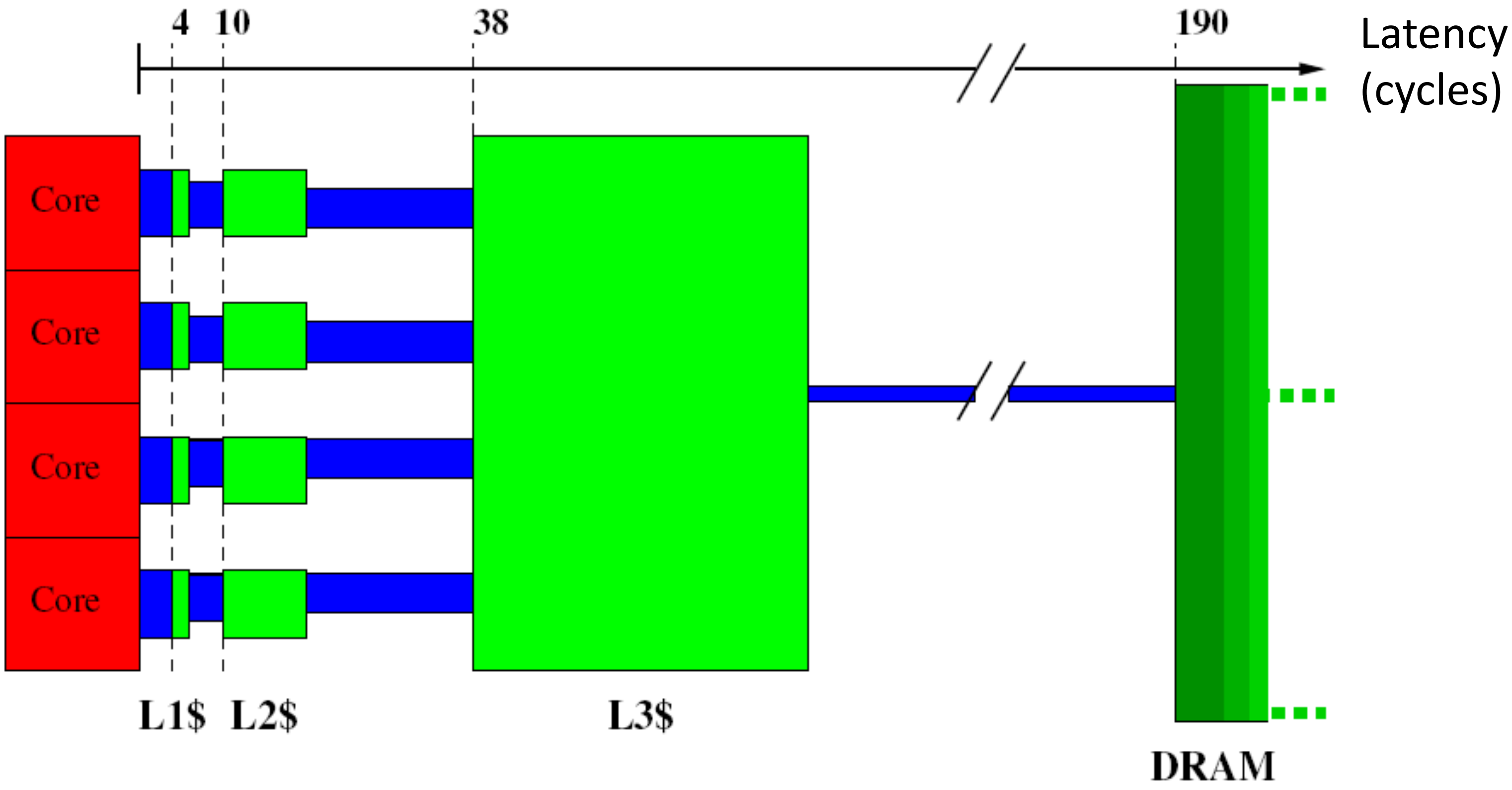
ParaTools ThreadSpotter

- Shrinking cache memory per core is causing some applications to spend over half their runtime waiting for data to arrive in cache.

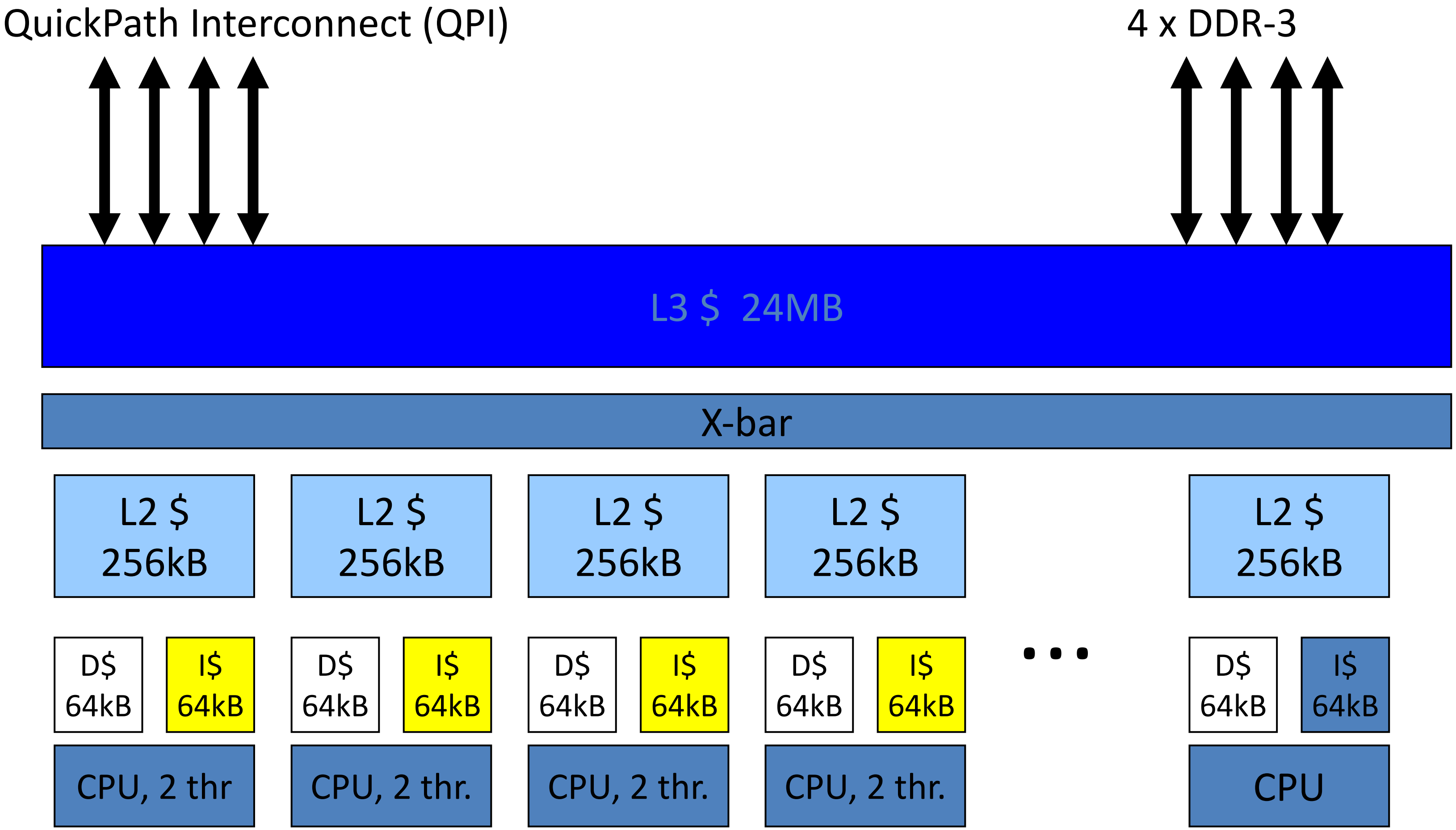
- ThreadSpotter can:
  - Analyze memory bandwidth and latency, data locality and thread communications.
  - Identify specific issues and pinpoints troublesome areas in source code.
  - Provide guidance towards a resolution.

- Supports **mixed language distributed memory** applications like CREATE-AV HELIOS and KESTREL.
  - Integrates with the TAU Performance System®.

Cache Capacity/Latency/BW

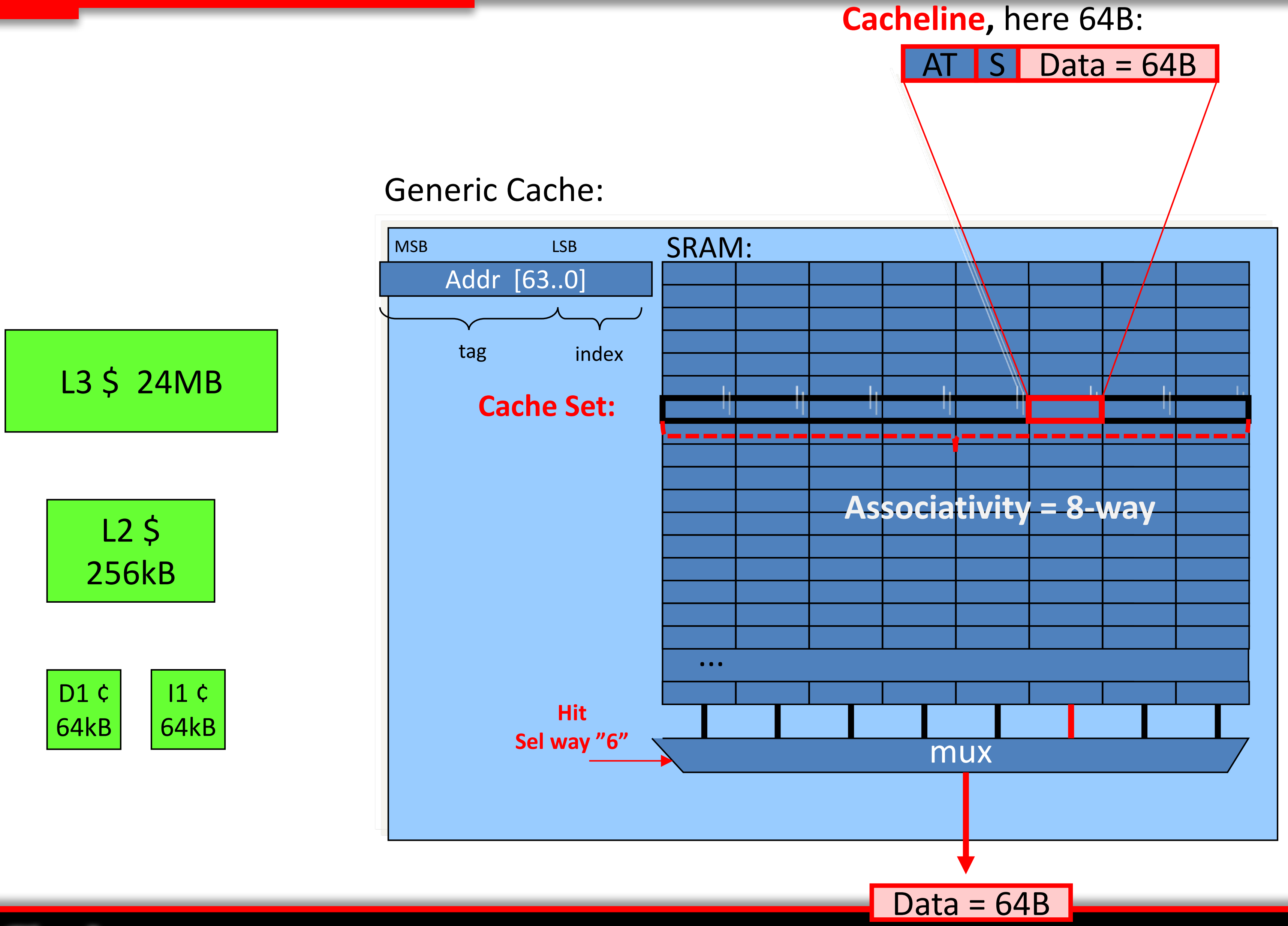


Intel i7



8 cores x 2 threads

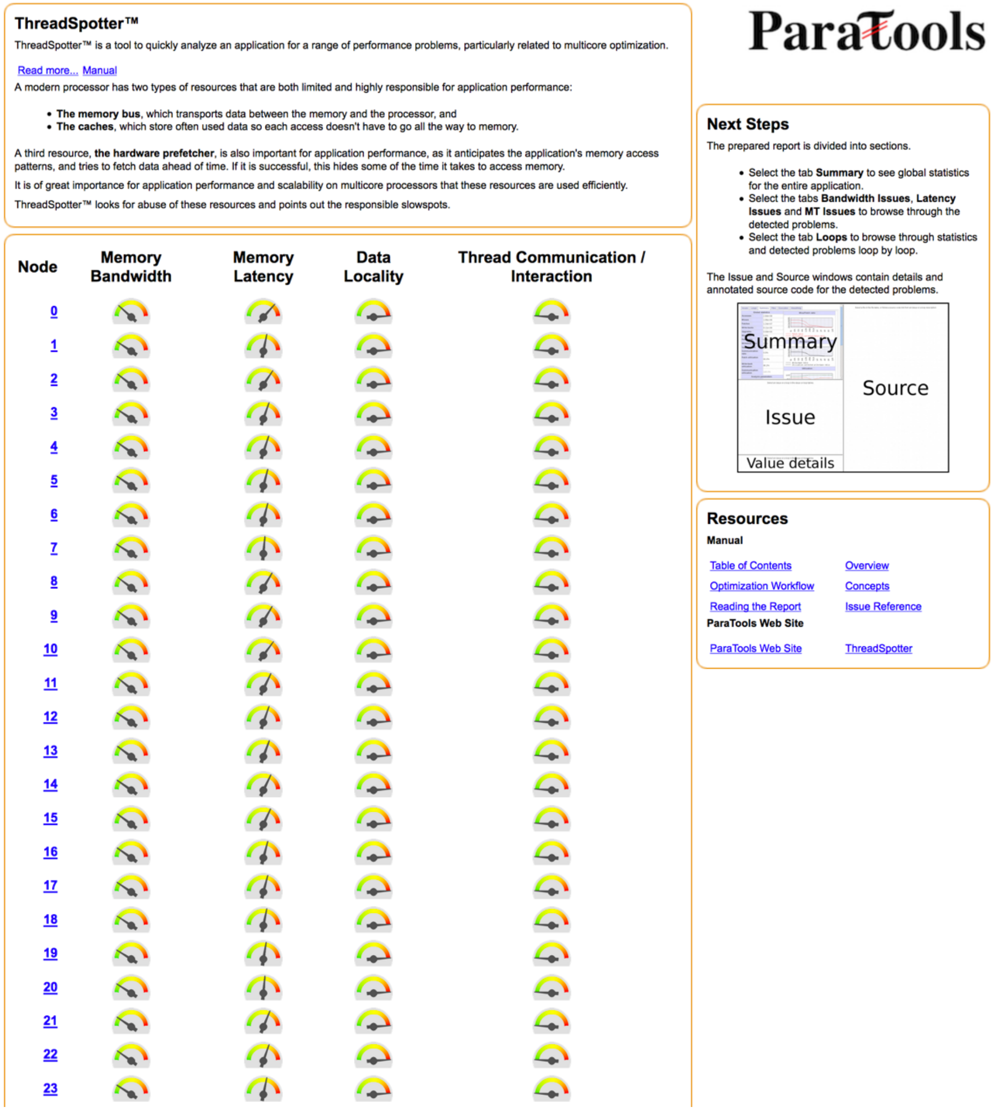
Typical Cache Implementation



Army HPC User Group Review

THREADSPOTTER REPORTS

Report Index for 2040 MPI Ranks



ThreadSpotter™

ThreadSpotter™ is a tool to quickly analyze an application for a range of performance problems, particularly related to multicore optimization.

Read more... Manual

A modern processor has two types of resources that are both limited and highly responsible for application performance:

- **The memory bus**, which transports data between the memory and the processor, and
- **The caches**, which store often used data so each access doesn't have to go all the way to memory.

A third resource, **the hardware prefetcher**, is also important for application performance, as it anticipates the application's memory access patterns, and tries to fetch data ahead of time. If it is successful, this hides some of the time it takes to access memory.

It is of great importance for application performance and scalability on multicore processors that these resources are used efficiently.

ThreadSpotter™ looks for abuse of these resources and points out the responsible slowspots.

Node	Memory Bandwidth	Memory Latency	Data Locality	Thread Communication / Interaction
0				
1				
2				
3				
4				
5				
6				
7				
8				
9				
10				
11				
12				
13				
14				
15				
16				
17				
18				
19				
20				
21				
22				
23				

Next Steps

The prepared report is divided into sections.

- Select the tab **Summary** to see global statistics for the entire application.
- Select the tabs **Bandwidth Issues**, **Latency Issues** and **MT Issues** to browse through the detected problems.
- Select the tab **Loops** to browse through statistics and detected problems loop by loop.

The Issue and Source windows contain details and annotated source code for the detected problems.

Resources

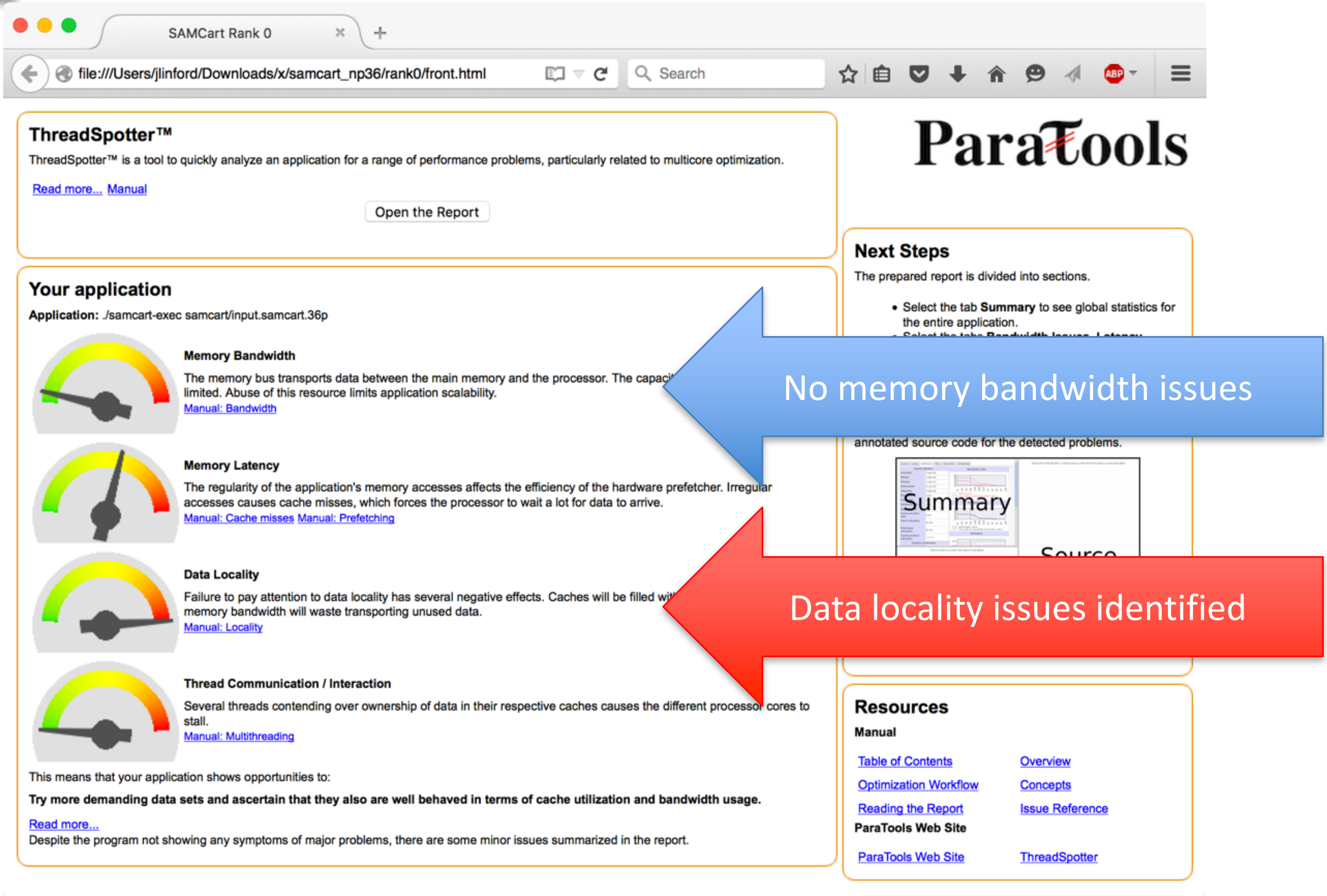
Manual

Table of Contents	Overview
Optimization Workflow	Concepts
Reading the Report	Issue Reference

ParaTools Web Site

ParaTools Web Site	ThreadSpotter

Report Cover page



SAMCart Rank 0
file:///Users/jlinford/Downloads/x/samcart_np36/rank0/front.html
Search
ParaTools

ThreadSpotter™

ThreadSpotter™ is a tool to quickly analyze an application for a range of performance problems, particularly related to multicore optimization.

Read more... Manual

Open the Report

Your application

Application: ./samcart-exec samcart/input.samcart.36p

Memory Bandwidth

The memory bus transports data between the main memory and the processor. The capaci
limited. Abuse of this resource limits application scalability.
Manual: Bandwidth

Memory Latency

The regularity of the application's memory accesses affects the efficiency of the hardware prefetcher. Irregular accesses causes cache misses, which forces the processor to wait a lot for data to arrive.
Manual: Cache misses Manual: Prefetching

Data Locality

Failure to pay attention to data locality has several negative effects. Caches will be filled wi
memory bandwidth will waste transporting unused data.
Manual: Locality

Thread Communication / Interaction

Several threads contending over ownership of data in their respective caches causes the different processor cores to stall.
Manual: Multithreading

This means that your application shows opportunities to:

Try more demanding data sets and ascertain that they also are well behaved in terms of cache utilization and bandwidth usage.

Read more...
Despite the program not showing any symptoms of major problems, there are some minor issues summarized in the report.

Next Steps

The prepared report is divided into sections.

- Select the tab **Summary** to see global statistics for the entire application.
- Select the tabs **Bandwidth Issues, Latency**

annotated source code for the detected problems.

Resources

Manual

Table of Contents
Overview
Optimization Workflow
Concepts
Reading the Report
Issue Reference

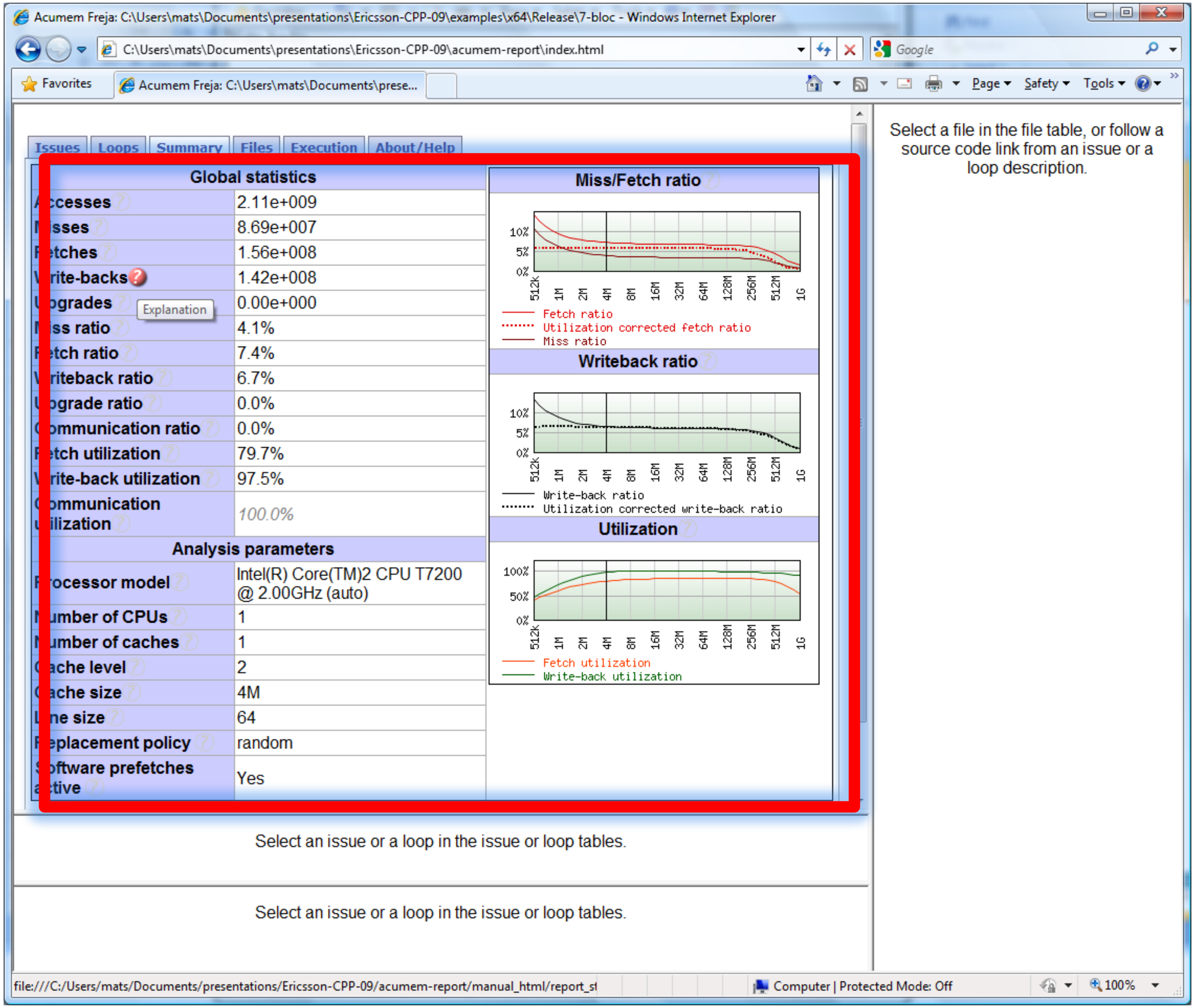
ParaTools Web Site

ParaTools Web Site
ThreadSpotter

No memory bandwidth issues

Data locality issues identified

Global statistics



Acumem Freja: C:\Users\mats\Documents\presentations\Ericsson-CPP-09\examples\x64\Release\7-bloc - Windows Internet Explorer

C:\Users\mats\Documents\presentations\Ericsson-CPP-09\acumem-report\index.html

Global statistics	
ccesses	2.11e+009
sses	8.69e+007
tches	1.56e+008
rite-backs	1.42e+008
pgrades	0.00e+000
ss ratio	4.1%
tch ratio	7.4%
riteback ratio	6.7%
pgrade ratio	0.0%
ommunication ratio	0.0%
tch utilization	79.7%
rite-back utilization	97.5%
ommunication ilization	*100.0%*
Analysis parameters	
ocessor model	Intel(R) Core(TM)2 CPU T7200 @ 2.00GHz (auto)
umber of CPUs	1
umber of caches	1
ache level	2
ache size	4M
ne size	64
eplacement policy	random
oftware prefetches tive	Yes

Select a file in the file table, or follow a source code link from an issue or a loop description.

Select an issue or a loop in the issue or loop tables.

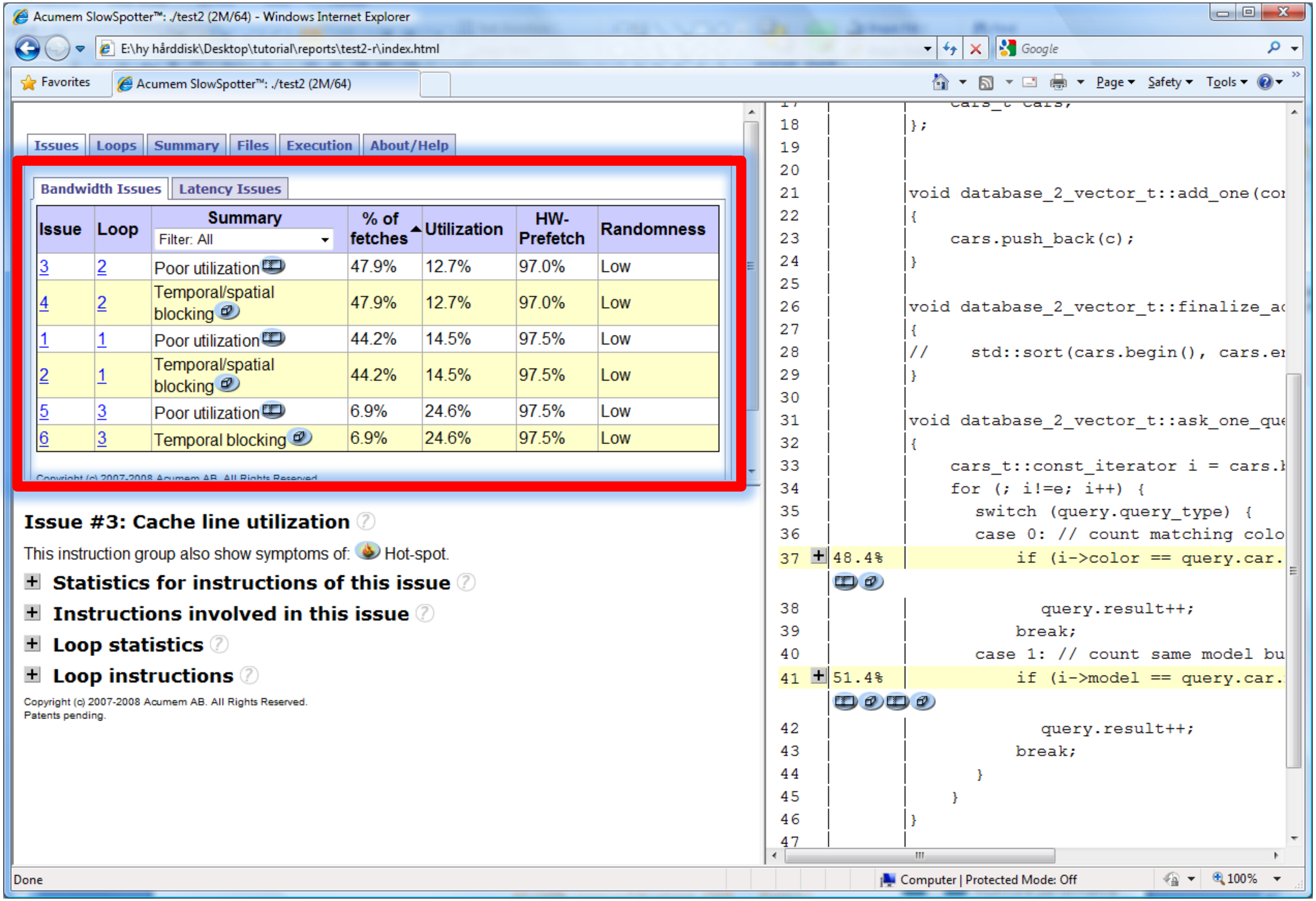
Select an issue or a loop in the issue or loop tables.

file:///C:/Users/mats/Documents/presentations/Ericsson-CPP-09/acumem-report/manual_html/report_st

ThreadSpotter Concepts

- *Issue*
  - A problem or opportunity to improve performance
  - Presented with backing statistics and source navigation
  - Online help to explain details
- *Loop*
  - An instruction cycle usually corresponding to a code loop
- *Instruction group*
  - A collection of instructions touching related data

ParaTools ThreadSpotter report: Issues


Acumem SlowSpotter™: ./test2 (2M/64) - Windows Internet Explorer
E:\hy hårddisk\Desktop\tutorial\reports\test2-r\index.html
Google
Favorites
Acumem SlowSpotter™: ./test2 (2M/64)
Page
Safety
Tools
Issues
Loops
Summary
Files
Execution
About/Help
Bandwidth Issues
Latency Issues
Done
Computer | Protected Mode: Off
100%

Issue	Loop	Summary Filter: All	% of fetches	Utilization	HW-Prefetch	Randomness
3	2	Poor utilization	47.9%	12.7%	97.0%	Low
4	2	Temporal/spatial blocking	47.9%	12.7%	97.0%	Low
1	1	Poor utilization	44.2%	14.5%	97.5%	Low
2	1	Temporal/spatial blocking	44.2%	14.5%	97.5%	Low
5	3	Poor utilization	6.9%	24.6%	97.5%	Low
6	3	Temporal blocking	6.9%	24.6%	97.5%	Low

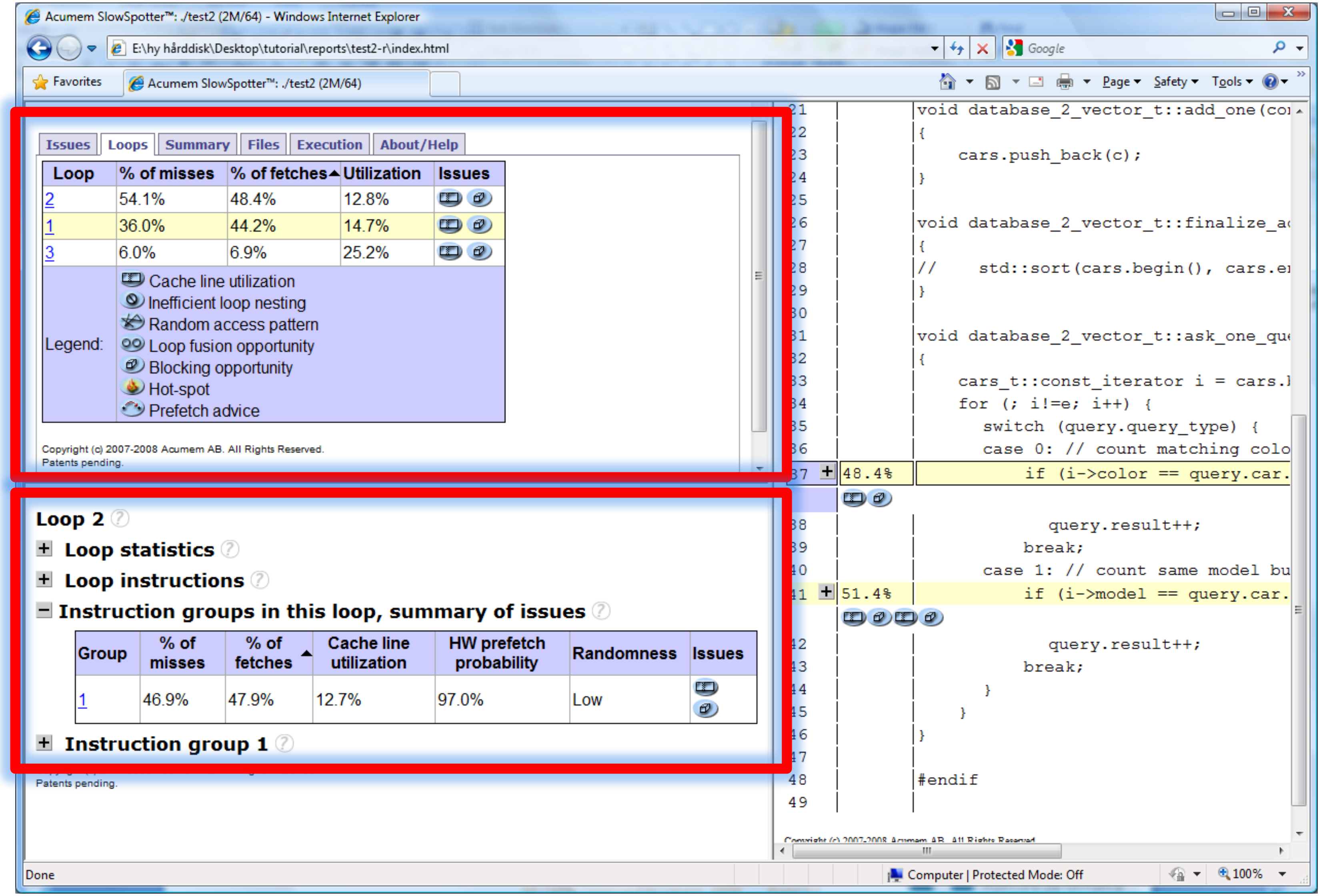
Issue #3: Cache line utilization

This instruction group also show symptoms of: Hot-spot.

- Statistics for instructions of this issue
- Instructions involved in this issue
- Loop statistics
- Loop instructions

```
          cars_t cars;
        };


        void database_2_vector_t::add_one(co
        {
            cars.push_back(c);
        }

        void database_2_vector_t::finalize_a
        {
        //     std::sort(cars.begin(), cars.e
        }

        void database_2_vector_t::ask_one_qu
        {
            cars_t::const_iterator i = cars.
            for (; i!=e; i++) {
              switch (query.query_type) {
              case 0: // count matching colo
+ 48.4%          if (i->color == query.car.

                   query.result++;
                 break;
              case 1: // count same model bu
+ 51.4%          if (i->model == query.car.

                   query.result++;
                 break;
              }
            }
        }
```

ParaTools ThreadSpotter report: Loops


Acumem SlowSpotter™: ./test2 (2M/64) - Windows Internet Explorer
E:\hy hårddisk\Desktop\tutorial\reports\test2-r\index.html
Google
Favorites
Acumem SlowSpotter™: ./test2 (2M/64)
Page
Safety
Tools
Issues
Loops
Summary
Files
Execution
About/Help
Done
Computer | Protected Mode: Off
100%

Loop	% of misses	% of fetches▲	Utilization	Issues
2	54.1%	48.4%	12.8%	▭ ⧉
1	36.0%	44.2%	14.7%	▭ ⧉
3	6.0%	6.9%	25.2%	▭ ⧉
Legend:	▭ Cache line utilization ⊘ Inefficient loop nesting ✲ Random access pattern ∞ Loop fusion opportunity ⧉ Blocking opportunity ● Hot-spot ◠ Prefetch advice			

Copyright (c) 2007-2008 Acumem AB. All Rights Reserved.
Patents pending.

Loop 2 ?

⊞ **Loop statistics** ?

⊞ **Loop instructions** ?

⊟ **Instruction groups in this loop, summary of issues** ?

Group	% of misses	% of fetches ▲	Cache line utilization	HW prefetch probability	Randomness	Issues
1	46.9%	47.9%	12.7%	97.0%	Low	▭ ⧉

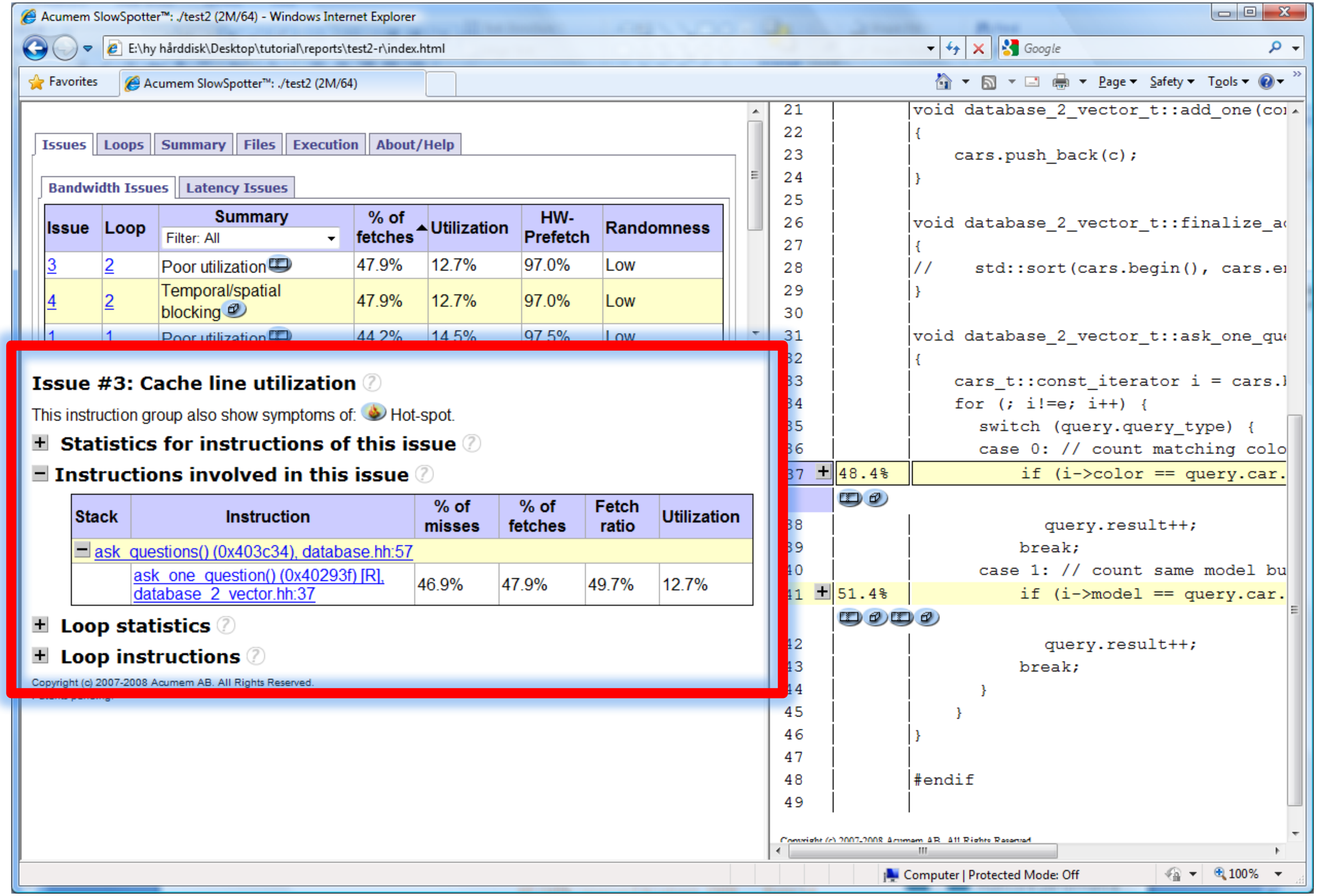
⊞ **Instruction group 1** ?

Patents pending.

```
                void database_2_vector_t::add_one(coı
                {
                    cars.push_back(c);
                }

                void database_2_vector_t::finalize_a
                {
                //     std::sort(cars.begin(), cars.eı
                }

                void database_2_vector_t::ask_one_que
                {
                    cars_t::const_iterator i = cars.l
                    for (; i!=e; i++) {
                       switch (query.query_type) {
                       case 0: // count matching colo
+ 48.4%                   if (i->color == query.car.
  ▭ ⧉
                             query.result++;
                          break;
                       case 1: // count same model bu
+ 51.4%                   if (i->model == query.car.
  ▭ ⧉ ▭ ⧉
                             query.result++;
                          break;
                       }
                    }
                }

                #endif
```

Copyright (c) 2007-2008 Acumem AB. All Rights Reserved.

ParaTools ThreadSpotter report: Details


Acumem SlowSpotter™: ./test2 (2M/64) - Windows Internet Explorer
E:\hy hårddisk\Desktop\tutorial\reports\test2-r\index.html
Google
Favorites
Acumem SlowSpotter™: ./test2 (2M/64)
Page
Safety
Tools
Computer | Protected Mode: Off
100%

Issues | Loops | Summary | Files | Execution | About/Help

Bandwidth Issues | Latency Issues

Issue	Loop	Summary Filter: All	% of fetches	Utilization	HW-Prefetch	Randomness
3	2	Poor utilization	47.9%	12.7%	97.0%	Low
4	2	Temporal/spatial blocking	47.9%	12.7%	97.0%	Low
1	1	Poor utilization	44.2%	14.5%	97.5%	Low

Issue #3: Cache line utilization

This instruction group also show symptoms of: Hot-spot.

+ **Statistics for instructions of this issue**

− **Instructions involved in this issue**

Stack	Instruction	% of misses	% of fetches	Fetch ratio	Utilization
− ask_questions() (0x403c34), database.hh:57					
	ask_one_question() (0x40293f) [R], database_2_vector.hh:37	46.9%	47.9%	49.7%	12.7%

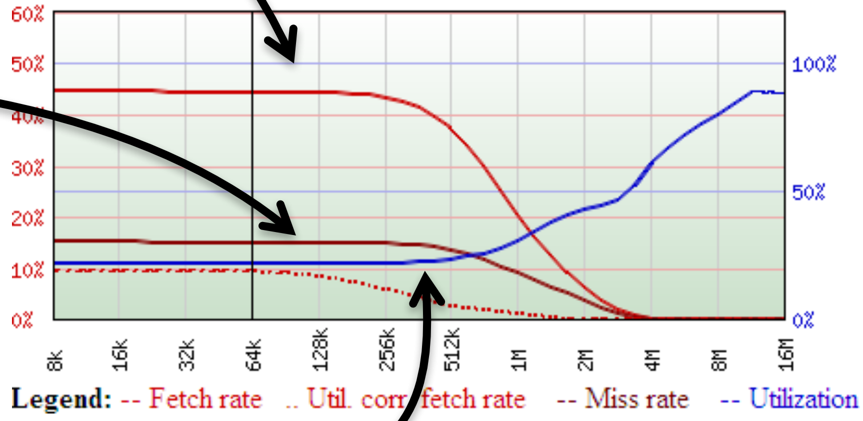
+ **Loop statistics**

+ **Loop instructions**

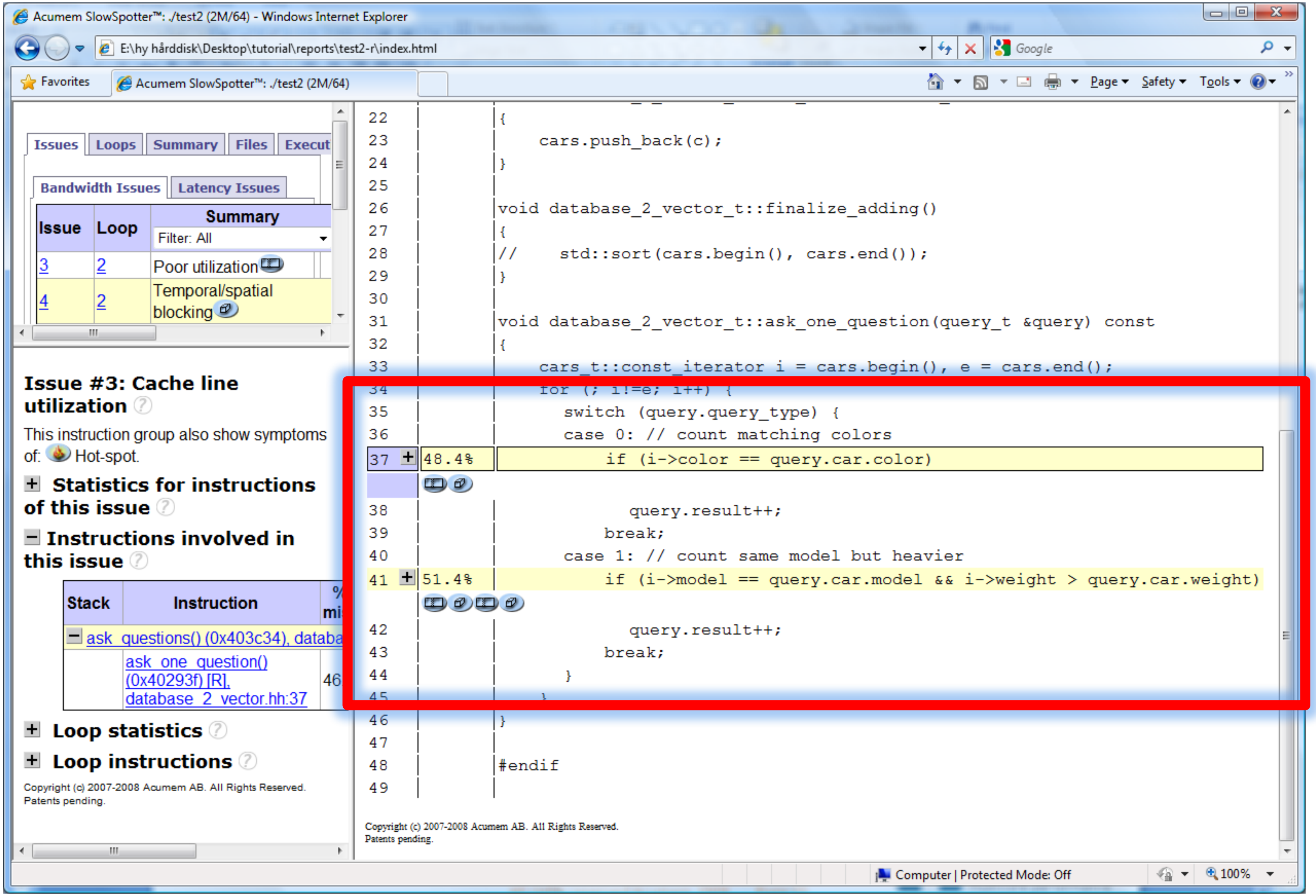
```
void database_2_vector_t::add_one(co
{
    cars.push_back(c);
}

void database_2_vector_t::finalize_a
{
//    std::sort(cars.begin(), cars.e
}

void database_2_vector_t::ask_one_que
{
    cars_t::const_iterator i = cars.
    for (; i!=e; i++) {
      switch (query.query_type) {
      case 0: // count matching colo
+ 48.4%   if (i->color == query.car.
          query.result++;
        break;
      case 1: // count same model bu
+ 51.4%   if (i->model == query.car.
          query.result++;
        break;
      }
    }
}

#endif
```

Metrics as a function of cache size

- Fetch ratio
  - Memory operations that cause a data transfer to/from RAM
- Miss ratio
  - Memory operations that stall due to cache misses.
- Fetch utilization
  - Fraction of the data loaded into the cache that are actually used



ParaTools ThreadSpotter report: Source annotation


Acumem SlowSpotter™: ./test2 (2M/64) - Windows Internet Explorer
E:\hy hårddisk\Desktop\tutorial\reports\test2-r\index.html
Google
Favorites
Acumem SlowSpotter™: ./test2 (2M/64)
Page
Safety
Tools
Issues
Loops
Summary
Files
Execut
Bandwidth Issues
Latency Issues
Computer | Protected Mode: Off
100%

Issue	Loop	Summary Filter: All
3	2	Poor utilization
4	2	Temporal/spatial blocking

Issue #3: Cache line utilization

This instruction group also show symptoms of: Hot-spot.

⊞ **Statistics for instructions of this issue**

⊟ **Instructions involved in this issue**

Stack	Instruction	% mi
⊟ ask_questions() (0x403c34), databa		
	ask_one_question() (0x40293f) [R], database_2_vector.hh:37	46

⊞ **Loop statistics**

⊞ **Loop instructions**

Copyright (c) 2007-2008 Acumem AB. All Rights Reserved.
Patents pending.

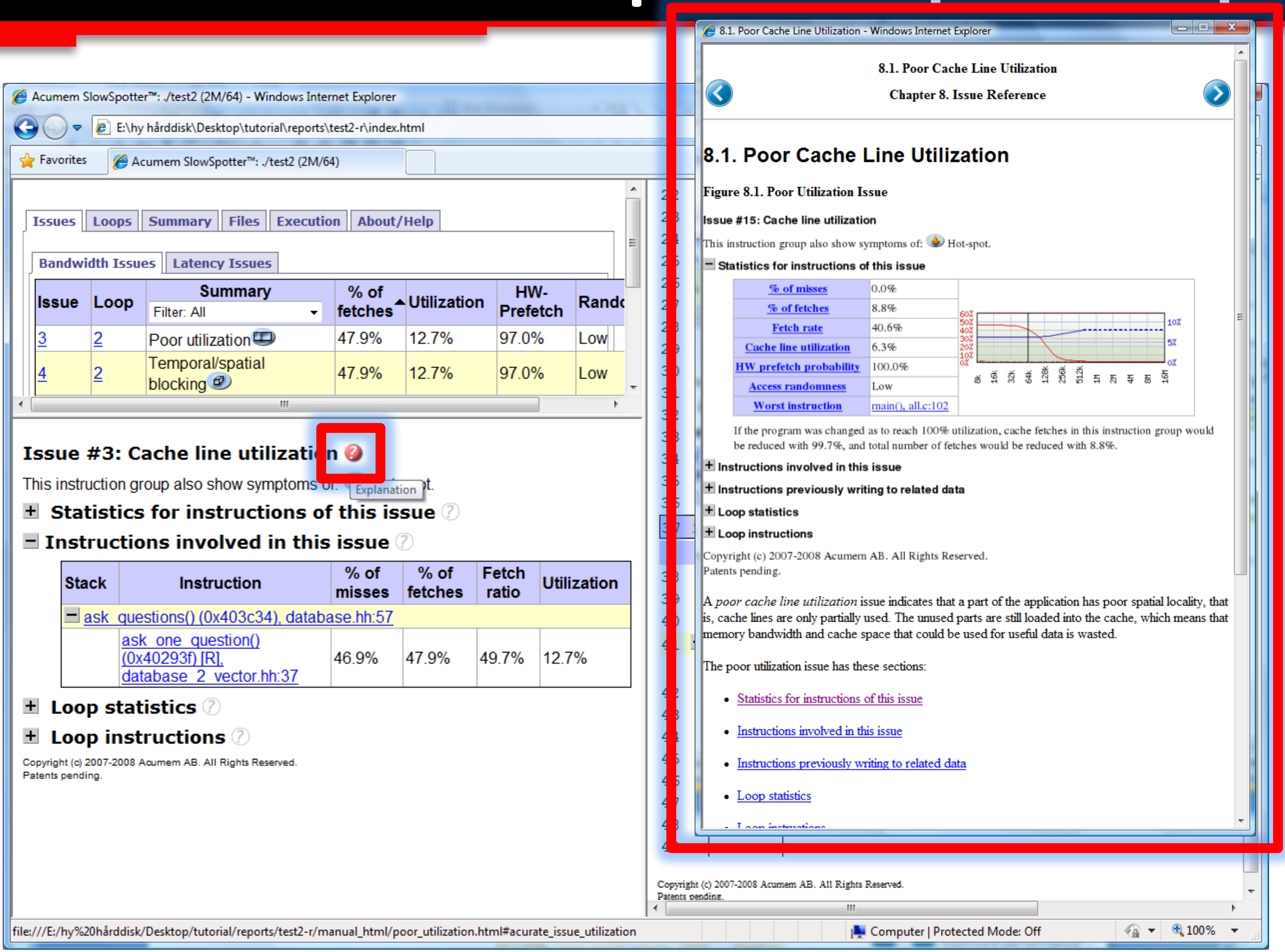
```
22                    {
23                        cars.push_back(c);
24                    }
25
26                    void database_2_vector_t::finalize_adding()
27                    {
28                    //     std::sort(cars.begin(), cars.end());
29                    }
30
31                    void database_2_vector_t::ask_one_question(query_t &query) const
32                    {
33                        cars_t::const_iterator i = cars.begin(), e = cars.end();
34                        for (; i!=e; i++) {
35                          switch (query.query_type) {
36                          case 0: // count matching colors
37  +  48.4%                    if (i->color == query.car.color)

38                                query.result++;
39                              break;
40                          case 1: // count same model but heavier
41  +  51.4%                    if (i->model == query.car.model && i->weight > query.car.weight)

42                                query.result++;
43                              break;
44                          }
45                        }
46                    }
47
48                    #endif
49
```

Copyright (c) 2007-2008 Acumem AB. All Rights Reserved.
Patents pending.

Report Vocabulary

- **Miss ratio**: What is the likelihood that a memory access will miss in a cache?
- **Miss rate**: Misses per unit, e.g. per-second or per-1000-instructions
- **Fetch ratio/rate**: What is the likelihood that a memory access will cause a fetch to the cache (including HW prefetching)
- **Fetch utilization**: What fraction of a cacheline was used before it got evicted
- **Writeback utilization**: What fraction of a cacheline written back to memory contains dirty data
- **Communication utilization**: What fraction of a communicated cacheline is ever used?

ParaTools ThreadSpotter report: help



Acumem SlowSpotter™: ./test2 (2M/64) - Windows Internet Explorer
E:\hy hårddisk\Desktop\tutorial\reports\test2-r\index.html
Favorites
Acumem SlowSpotter™: ./test2 (2M/64)
Issues
Loops
Summary
Files
Execution
About/Help
Bandwidth Issues
Latency Issues
Explanation
8.1. Poor Cache Line Utilization - Windows Internet Explorer
60%
50%
40%
30%
20%
10%
0%
10%
5%
0%
8k
16k
32k
64k
128k
256k
512k
1M
2M
4M
8M
16M
Computer | Protected Mode: Off
100%

Issue	Loop	Summary Filter: All	% of fetches	Utilization	HW-Prefetch	Rando
3	2	Poor utilization	47.9%	12.7%	97.0%	Low
4	2	Temporal/spatial blocking	47.9%	12.7%	97.0%	Low

Issue #3: Cache line utilization

This instruction group also show symptoms of: ...t.

+ Statistics for instructions of this issue

− Instructions involved in this issue

Stack	Instruction	% of misses	% of fetches	Fetch ratio	Utilization
− ask_questions() (0x403c34), database.hh:57					
	ask_one_question() (0x40293f) [R], database_2_vector.hh:37	46.9%	47.9%	49.7%	12.7%

+ Loop statistics

+ Loop instructions

file:///E:/hy%20hårddisk/Desktop/tutorial/reports/test2-r/manual_html/poor_utilization.html#acurate_issue_utilization

8.1. Poor Cache Line Utilization

Chapter 8. Issue Reference

8.1. Poor Cache Line Utilization

Figure 8.1. Poor Utilization Issue

Issue #15: Cache line utilization

This instruction group also show symptoms of: Hot-spot.

− Statistics for instructions of this issue

% of misses	0.0%
% of fetches	8.8%
Fetch rate	40.6%
Cache line utilization	6.3%
HW prefetch probability	100.0%
Access randomness	Low
Worst instruction	main(), all.c:102

If the program was changed as to reach 100% utilization, cache fetches in this instruction group would be reduced with 99.7%, and total number of fetches would be reduced with 8.8%.

+ Instructions involved in this issue

+ Instructions previously writing to related data

+ Loop statistics

+ Loop instructions

A *poor cache line utilization* issue indicates that a part of the application has poor spatial locality, that is, cache lines are only partially used. The unused parts are still loaded into the cache, which means that memory bandwidth and cache space that could be used for useful data is wasted.

The poor utilization issue has these sections:

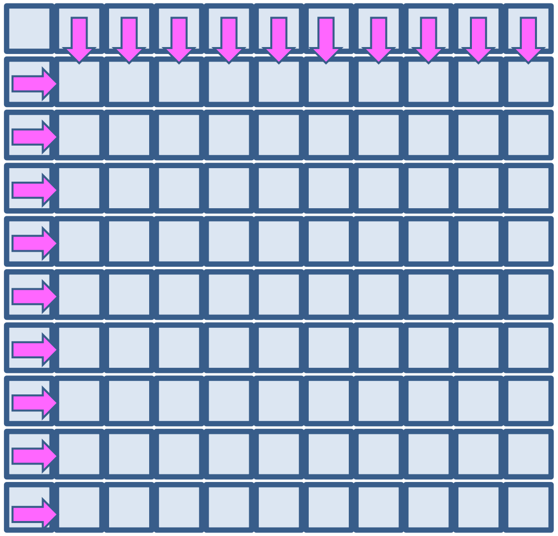
- Statistics for instructions of this issue
- Instructions involved in this issue
- Instructions previously writing to related data
- Loop statistics
- Loop instructions

Army HPC User Group Review

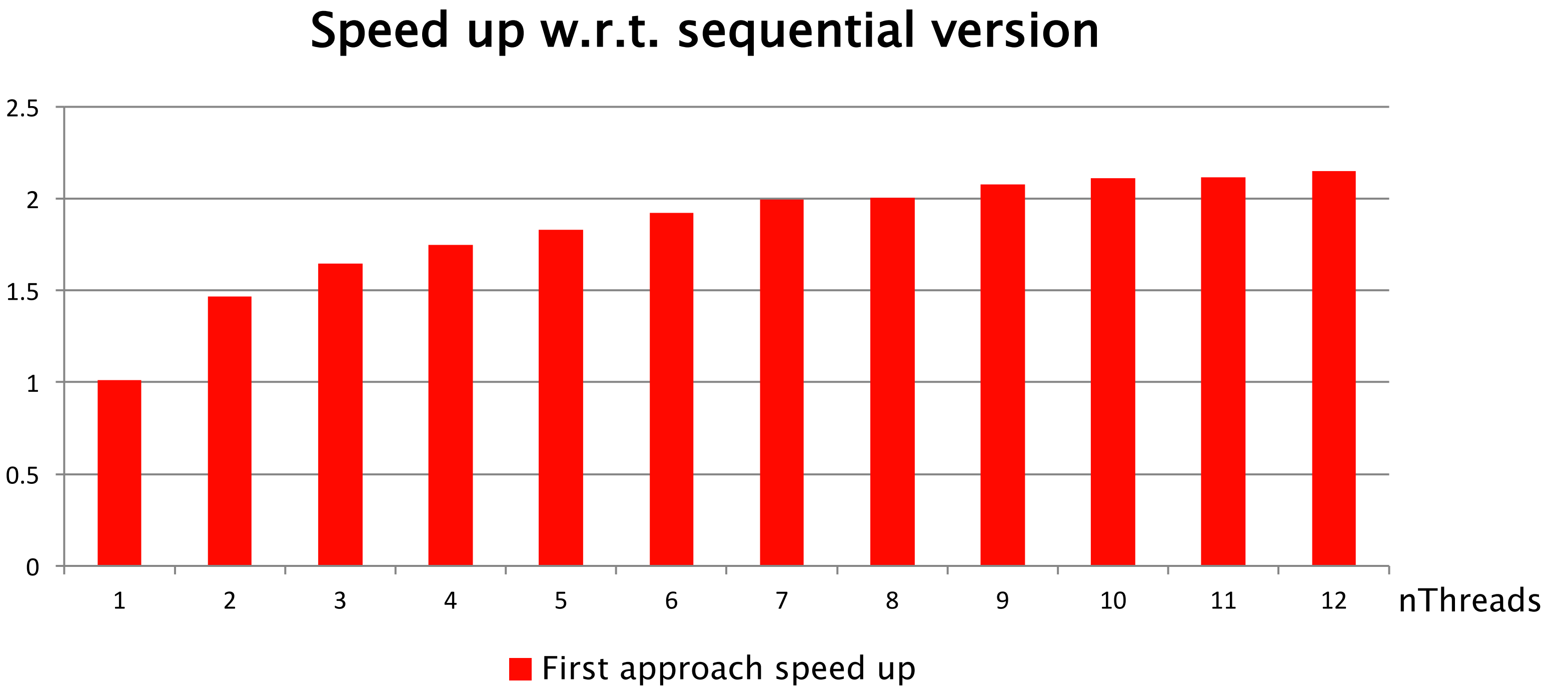
EXAMPLE: TEMPORAL BLOCKING

Overview of the forward elimination stage

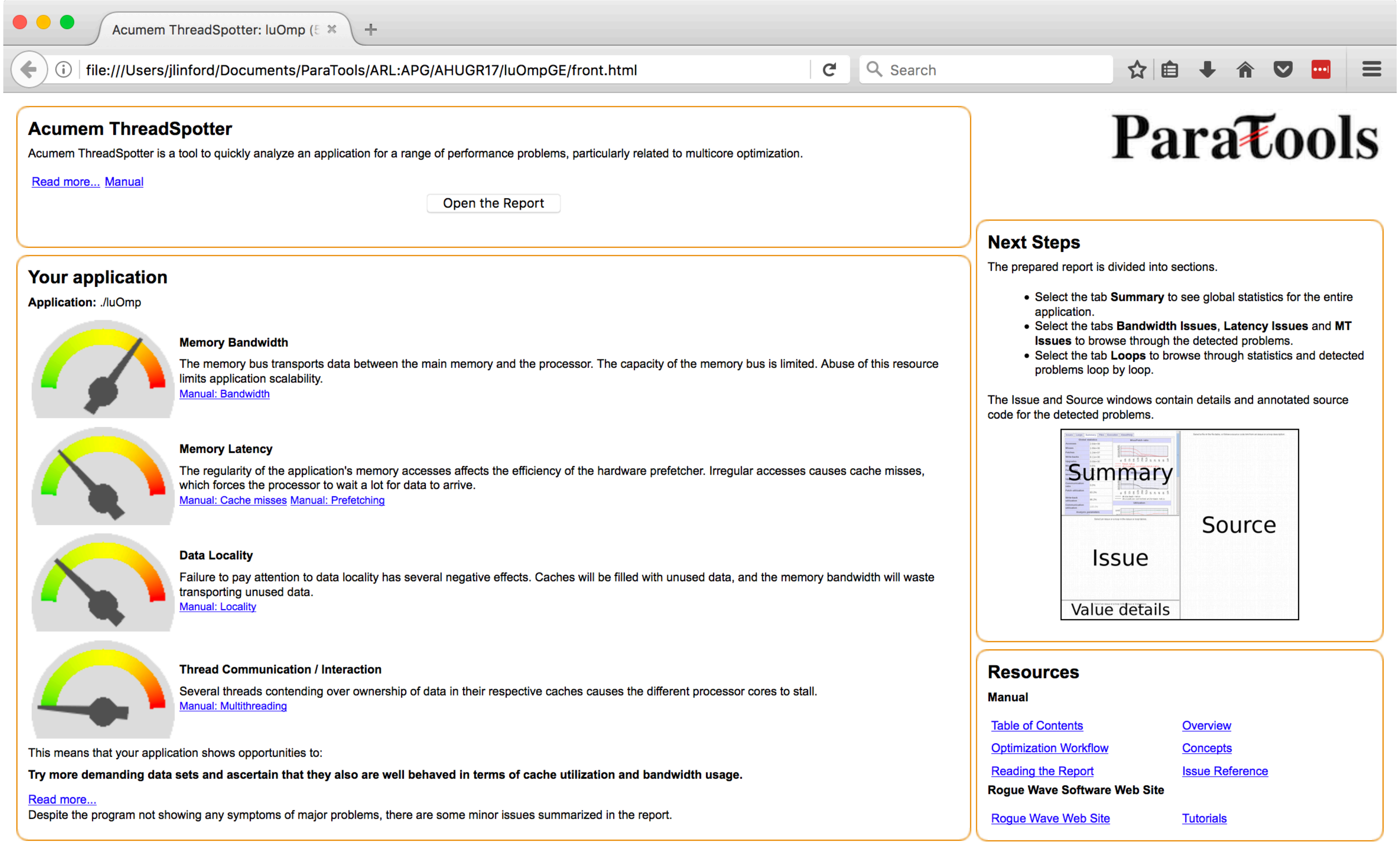
```
for i=1 to n-1
  find pivotPos in column i
  if pivotPos $\neq$ i
    exchange rows(pivotPos,i)
  end if

  for j=i+1 to n
    $A(i,j) = A(i,j)/A(i,i)$
  end for j
 !$omp parallel do private ( i ,j )
  for j=i+1 to n+1
    for k=i+1 to n
      $A(k,j)=A(k,j)-A(k,i)\times A(i,j)$
    end for k
  end for j
end for i
```



First approach speed up



[ParaTools ThreadSpotter report]()



Acumem ThreadSpotter: luOmp (E ×
file:///Users/jlinford/Documents/ParaTools/ARL:APG/AHUGR17/luOmpGE/front.html
Search

Acumem ThreadSpotter

Acumem ThreadSpotter is a tool to quickly analyze an application for a range of performance problems, particularly related to multicore optimization.

Read more... Manual

Your application

Application: ./luOmp

Memory Bandwidth

The memory bus transports data between the main memory and the processor. The capacity of the memory bus is limited. Abuse of this resource limits application scalability.

Manual: Bandwidth

Memory Latency

The regularity of the application's memory accesses affects the efficiency of the hardware prefetcher. Irregular accesses causes cache misses, which forces the processor to wait a lot for data to arrive.

Manual: Cache misses Manual: Prefetching

Data Locality

Failure to pay attention to data locality has several negative effects. Caches will be filled with unused data, and the memory bandwidth will waste transporting unused data.

Manual: Locality

Thread Communication / Interaction

Several threads contending over ownership of data in their respective caches causes the different processor cores to stall.

Manual: Multithreading

This means that your application shows opportunities to:

Try more demanding data sets and ascertain that they also are well behaved in terms of cache utilization and bandwidth usage.

Read more...

Despite the program not showing any symptoms of major problems, there are some minor issues summarized in the report.

Next Steps

The prepared report is divided into sections.

- Select the tab **Summary** to see global statistics for the entire application.
- Select the tabs **Bandwidth Issues**, **Latency Issues** and **MT Issues** to browse through the detected problems.
- Select the tab **Loops** to browse through statistics and detected problems loop by loop.

The Issue and Source windows contain details and annotated source code for the detected problems.

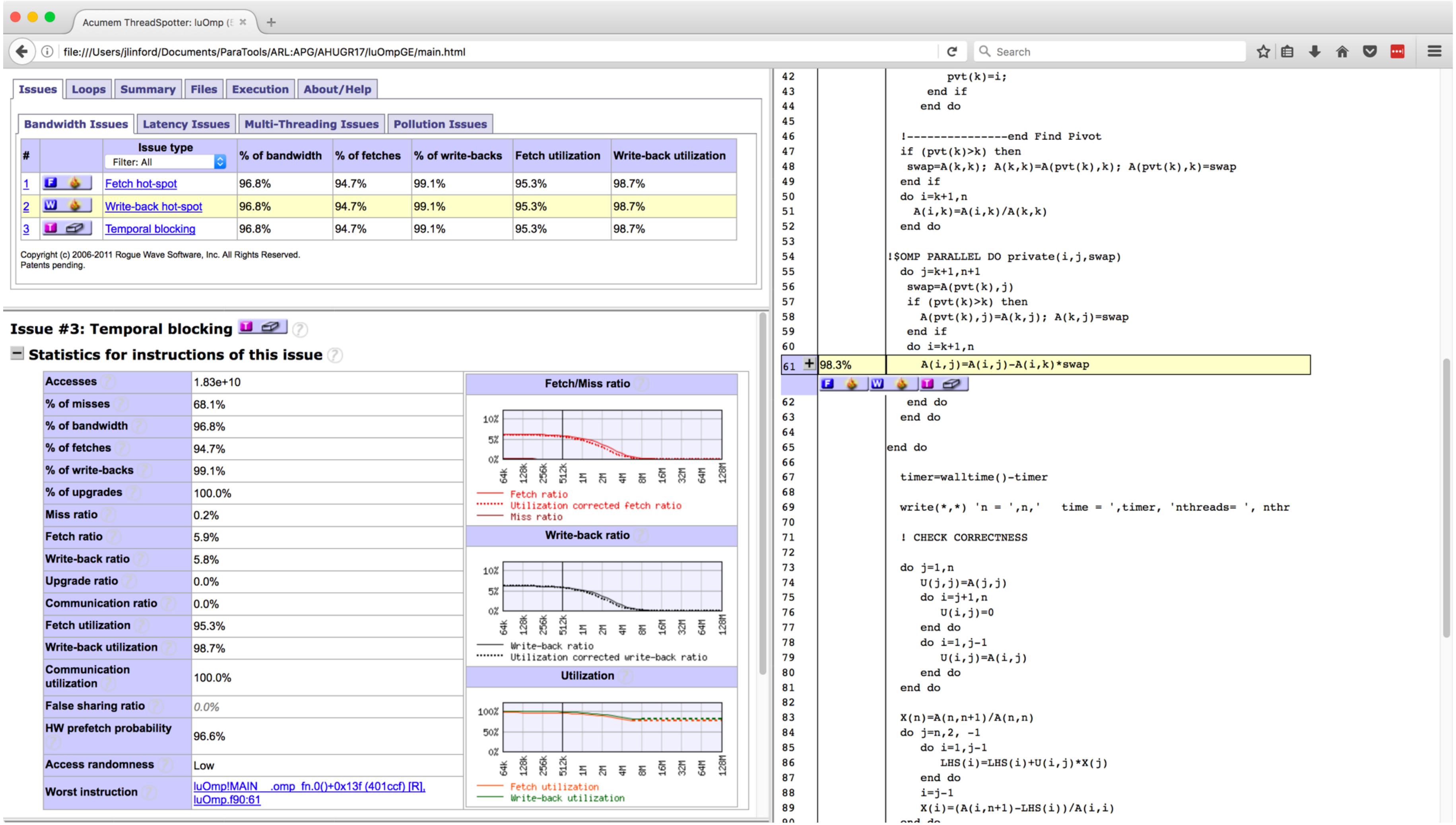
Resources

Manual

Table of Contents | Overview
Optimization Workflow | Concepts
Reading the Report | Issue Reference

Rogue Wave Software Web Site

Rogue Wave Web Site | Tutorials



#		Issue type Filter: All	% of bandwidth	% of fetches	% of write-backs	Fetch utilization	Write-back utilization
1	F	Fetch hot-spot	96.8%	94.7%	99.1%	95.3%	98.7%
2	W	Write-back hot-spot	96.8%	94.7%	99.1%	95.3%	98.7%
3	T	Temporal blocking	96.8%	94.7%	99.1%	95.3%	98.7%

Issue #3: Temporal blocking

Statistics for instructions of this issue

Accesses	1.83e+10
% of misses	68.1%
% of bandwidth	96.8%
% of fetches	94.7%
% of write-backs	99.1%
% of upgrades	100.0%
Miss ratio	0.2%
Fetch ratio	5.9%
Write-back ratio	5.8%
Upgrade ratio	0.0%
Communication ratio	0.0%
Fetch utilization	95.3%
Write-back utilization	98.7%
Communication utilization	100.0%
False sharing ratio	0.0%
HW prefetch probability	96.6%
Access randomness	Low
Worst instruction	luOmp!MAIN__.omp_fn.0()+0x13f (401ccf) [R], luOmp.f90:61

```
           pvt(k)=i;
        end if
       end do

     !---------------end Find Pivot
     if (pvt(k)>k) then
      swap=A(k,k); A(k,k)=A(pvt(k),k); A(pvt(k),k)=swap
     end if
     do i=k+1,n
       A(i,k)=A(i,k)/A(k,k)
     end do

   !$OMP PARALLEL DO private(i,j,swap)
     do j=k+1,n+1
      swap=A(pvt(k),j)
      if (pvt(k)>k) then
        A(pvt(k),j)=A(k,j); A(k,j)=swap
      end if
      do i=k+1,n
98.3%   A(i,j)=A(i,j)-A(i,k)*swap
      end do
     end do

   end do

     timer=walltime()-timer

     write(*,*) 'n = ',n,'   time = ',timer, 'nthreads= ', nthr

     ! CHECK CORRECTNESS

     do j=1,n
        U(j,j)=A(j,j)
        do i=j+1,n
           U(i,j)=0
        end do
        do i=1,j-1
           U(i,j)=A(i,j)
        end do
     end do

     X(n)=A(n,n+1)/A(n,n)
     do j=n,2, -1
        do i=1,j-1
           LHS(i)=LHS(i)+U(i,j)*X(j)
        end do
        i=j-1
        X(i)=(A(i,n+1)-LHS(i))/A(i,i)
```

What went wrong!

- For each prepared pivot, the whole matrix is accessed. The algorithm requires pivots to be calculated in order.
- Repeated eviction of the matrix' cache lines.

What went wrong!

- For each prepared pivot, the whole matrix is accessed. The algorithm requires pivots to be calculated in order.

Making things right!

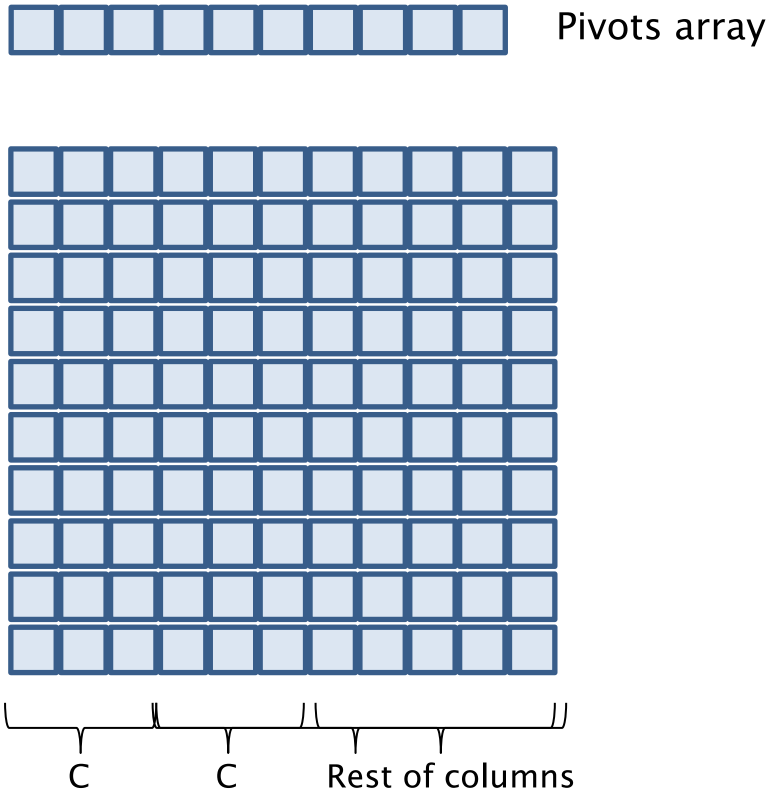
- Repeated eviction of the matrix' cache lines.
  - **Observation: Each column is an accumulation of eliminations using previous columns!**
  - **Temporal Blocking Advice says:**
    Use each column many times before it gets evicted

→ Arrange code to make more pivots available!

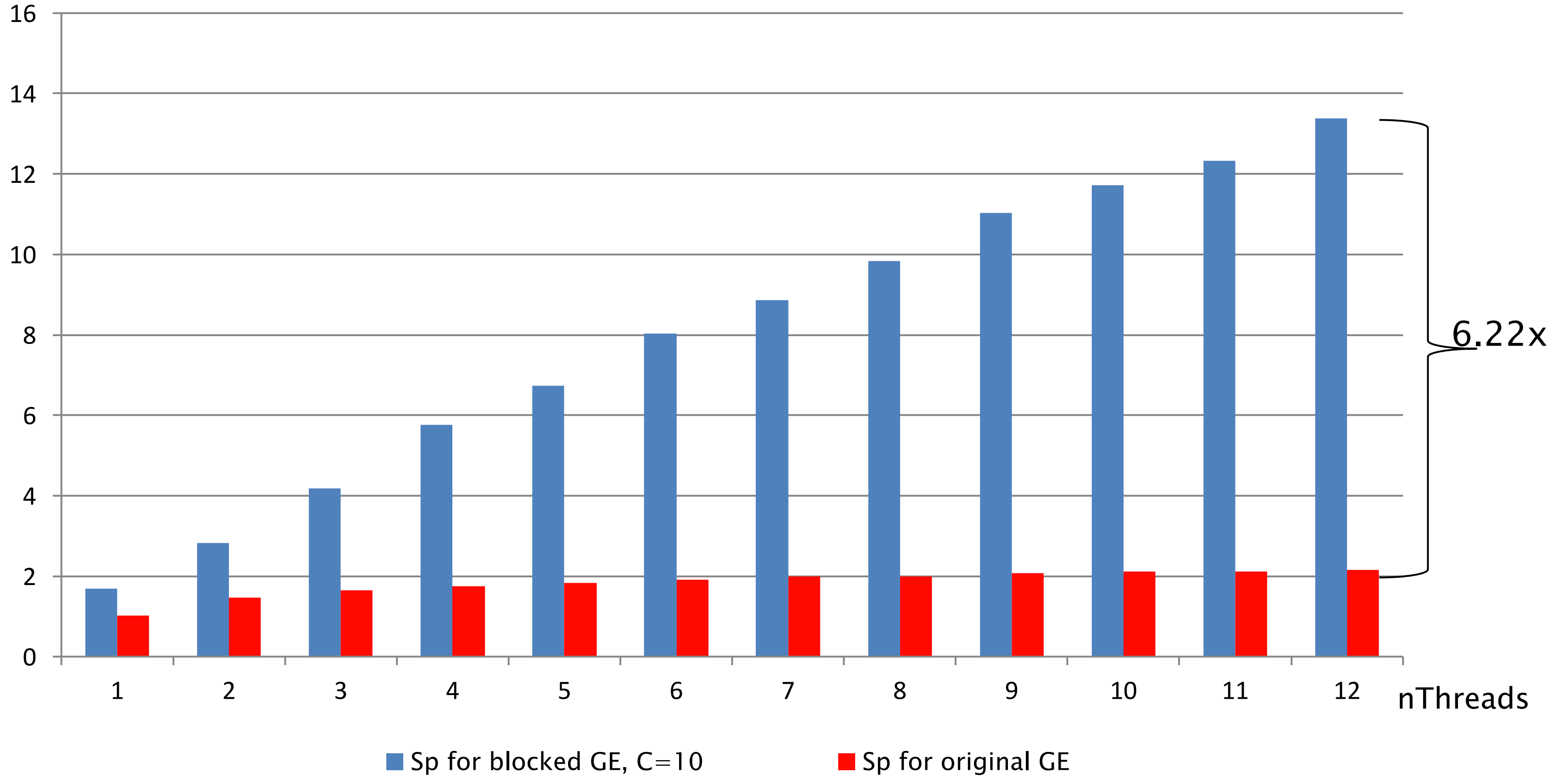
Blocking GE

```
for k=1 to n-1, step C
    BlockEnd=min(k+C-1,n)
  1 GE on A(k:n,k:BlockEnd) &
    Store C pivots' positions
  !$omp parallel do private ( i ,j )
    for each column j after BlockEnd
        for i=k to BlockEnd
  2     swap using pivots(i)
            elimination i on j
        end for i
    end for each j
End for k
```

The row exchange turned into a two-element swap before column elimination



Speed up w.r.t sequential time



Army HPC User Group Review

COMMAND LINE USAGE

PTTS Integration with TAU

- tau_exec **--ptts**
  - Sampling (sample_ts)
    - Generates sample files named ptts/sample.#.smp
  - Issue Identification (report_ts)
    - Generates report files named ptts/report.#.tsr
  - Reporting (view-static_ts)
    - Generates node_# directories with html reports
- Each rank produces log files for each step
  - sample_ts.#.log, report_ts.#.log, and view-static_ts.#.log

PTTS Integration with TAU

Flag	Description
-ptts	Use PTTS to generate performance data.
-ptts-post	Skip application sampling and post-process existing sample files. Useful for analyzing performance at each cache layer, or predicting performance on other architectures.
-ptts-num=<N>	Indicate number of MPI ranks if the size of the MPI communicator cannot be automatically detected, e.g. on Cray.
-ptts-sample-flags=<flags>	Additional flags to pass to the sample_ts command. Can also be specified in the TAU_TS_SAMPLE_FLAGS environment variable.
-ptts-report-flags=<flags>	Additional flags to pass to the report_ts command. Can also be specified in the TAU_TS_REPORT_FLAGS environment variable.

Sampler settings: Start & Stop

- Controlling when to start and stop sampling

- Start conditions
  - delay, -d <seconds>
  - --start-at-function <function name>
  - --start-at-address <0x1234123>
  - --start-at-ignore <pass count>
- Stop conditions
  - Duration, -t <seconds>
  - --stop-at-function <function name>
  - --stop-at-address <0x123123>
  - --stop-at-ignore <pass count>

Report settings: Issue selection

- depth
  - How long call stack to consider while differentiating instructions
  - --depth 0 – just consider the PC
  - --depth 1 – consider the PC and the site which called this function

- percentage
  - cutoff. Suppress uninteresting issues
  - --percentage 3 (% of fetches)

Report settings: Source/debug

- source directory – look for source in other places
  - Useful if the directories are not recorded in the debug information
  - -s directory
- binary – look for binaries (containing debug information in other places)
  - Useful if the binary has moved since sampling
  - -b path-to-actual-binary
- debug directory (for debug information stored external to the elf file)
  - -D directory
- debug level (varying degree of information)
  - --debug-level 0-3