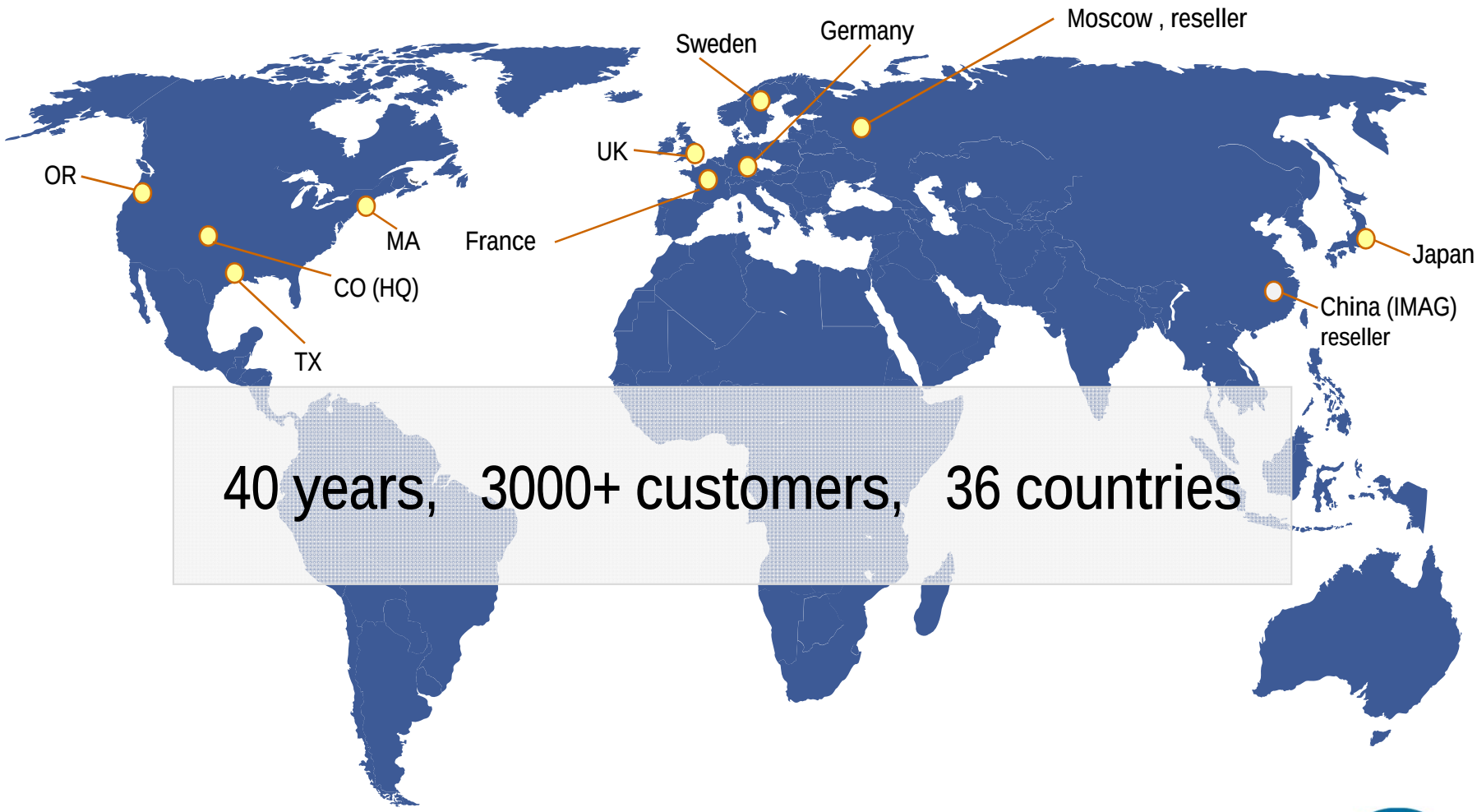
Optimizing Made Easy: ThreadSpotter

Erik Hagersten, Chief Scientist

Rogue Wave: A Global Company



ROGUE WAVE SOFTWARE

The largest independent provider of cross-platform software development tools for the next generation HPC applications

Visual Numerics®

Leader in embeddable math and statistics algorithms and visualization software for data-intensive applications.

TOTALVIEW TECHNOLOGIES

Industry-leading interactive analysis and debugging tools for the world's most sophisticated software applications.

acumem multicore performance

Leading provider of intelligent software technology which analyzes and optimizes the computing performance in single- and multi-core environments.

ROGUE WAVE® SOFTWARE

Rogue Wave Software: The HPC Tool Stack

ROGUE WAVE® SOFTWARE

Developing parallel, data-intensive applications is hard.
We make it easier.

Prototype:
PyIMSL Studio
PV-WAVE

Develop:
IMSL
SourcePro

Debug:
Total View:
Memory Scape
Replay Engine
CUDA
...

Optimize:
ThreadSpotter




ROGUE WAVE® SOFTWARE

Optimization: A Black Art for Experts Only?

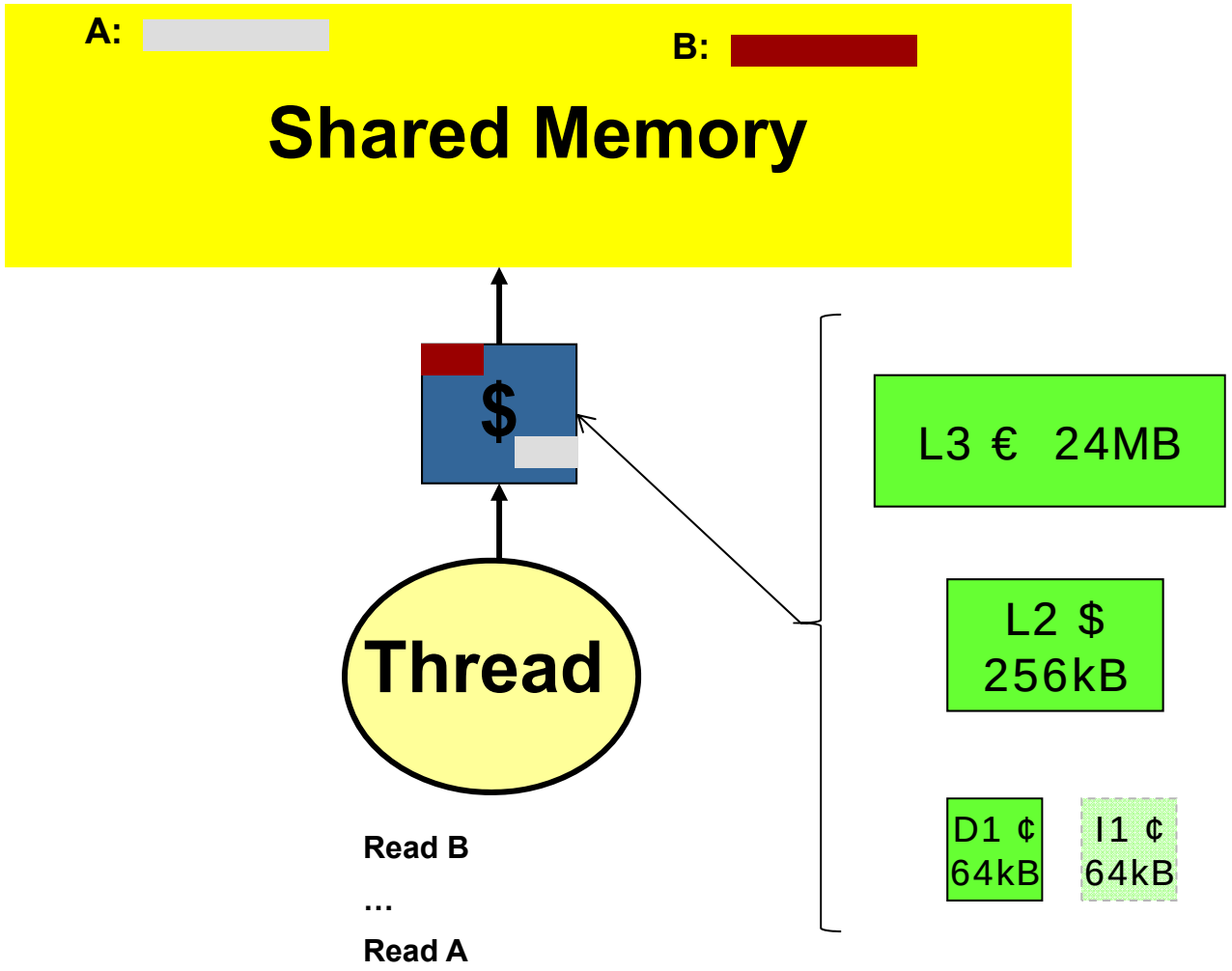
- If you do not optimize, you are not in HPC
- Requires extreme performance experts
- Outcome and project schedule hard to predict
- But, optimization can be very rewarding…

Developing parallel, data-intensive applications is hard.
We make it easier.

Outline

- Crash course on computer architecture
- Crash course on SW optimizations cookbook tricks
- ThreadSpotter technology overview
- Live demo
- Case studies
- Introducing the ThreadSpotter ”Guided Eval” CD

Caches are Friends not Enemies! Small is fast!



Cache implementation

Caches at all level roughly work like this:

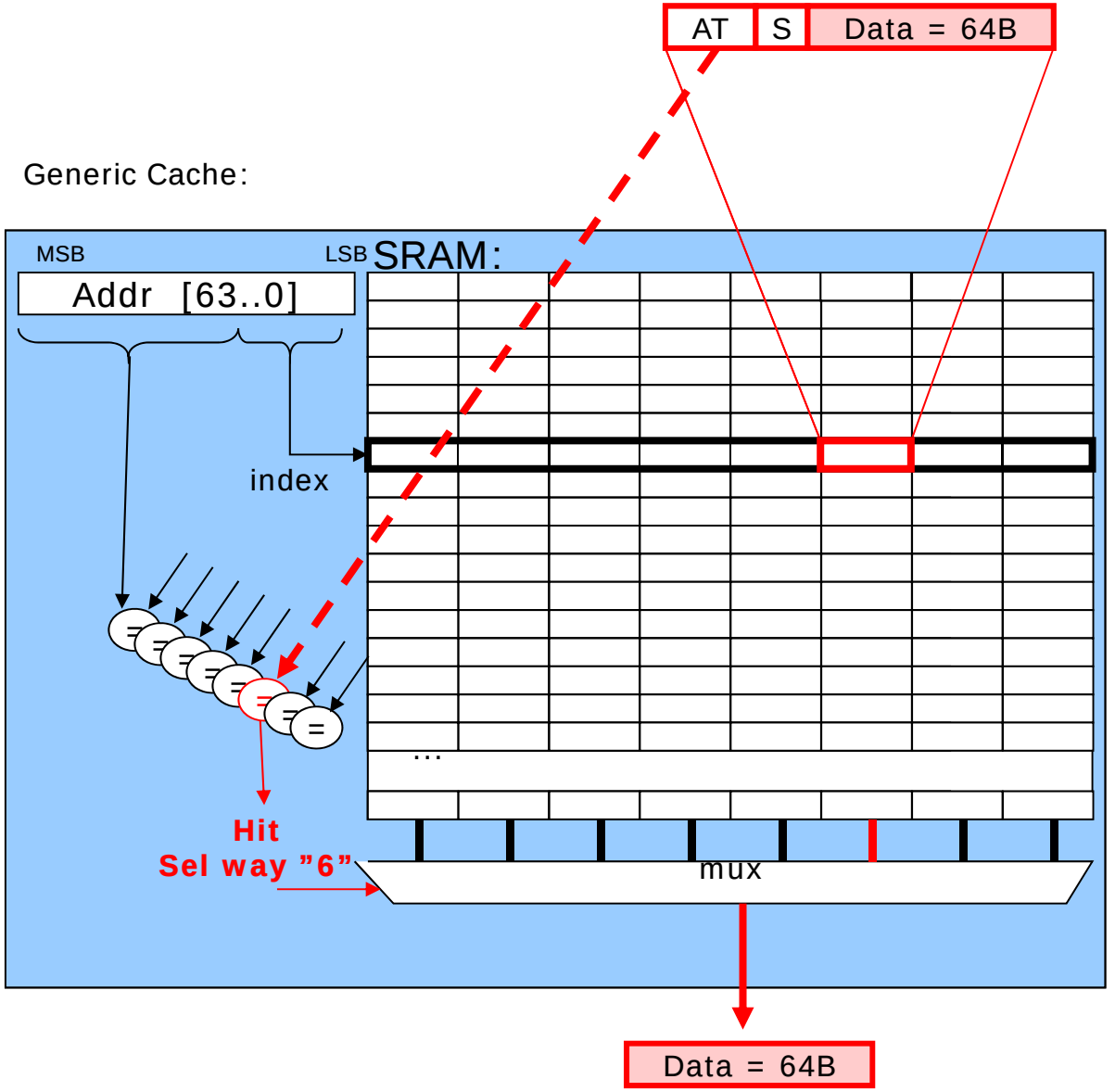
L3 € 24MB

L2 $ 256kB

D1 ¢ 64kB

I1 ¢ 64kB

Cacheline, here 64B:



UPPSALA UNIVERSITET

Cache lingo

Cacheline: Data chunk move to/from a cache

Cache set: Fraction of the cache identified by the index

Associativity: Number of alternative storage places for a cacheline

Replacement policy: picking the victim to throw out from a set (LRU/Random/Nehalem)

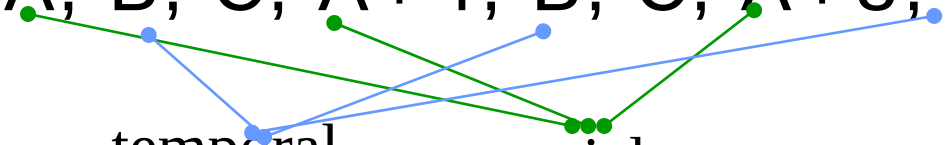
Temporal locality: Likelihood to access the same data again soon

Spatial locality: Likelihood to access nearby data again soon

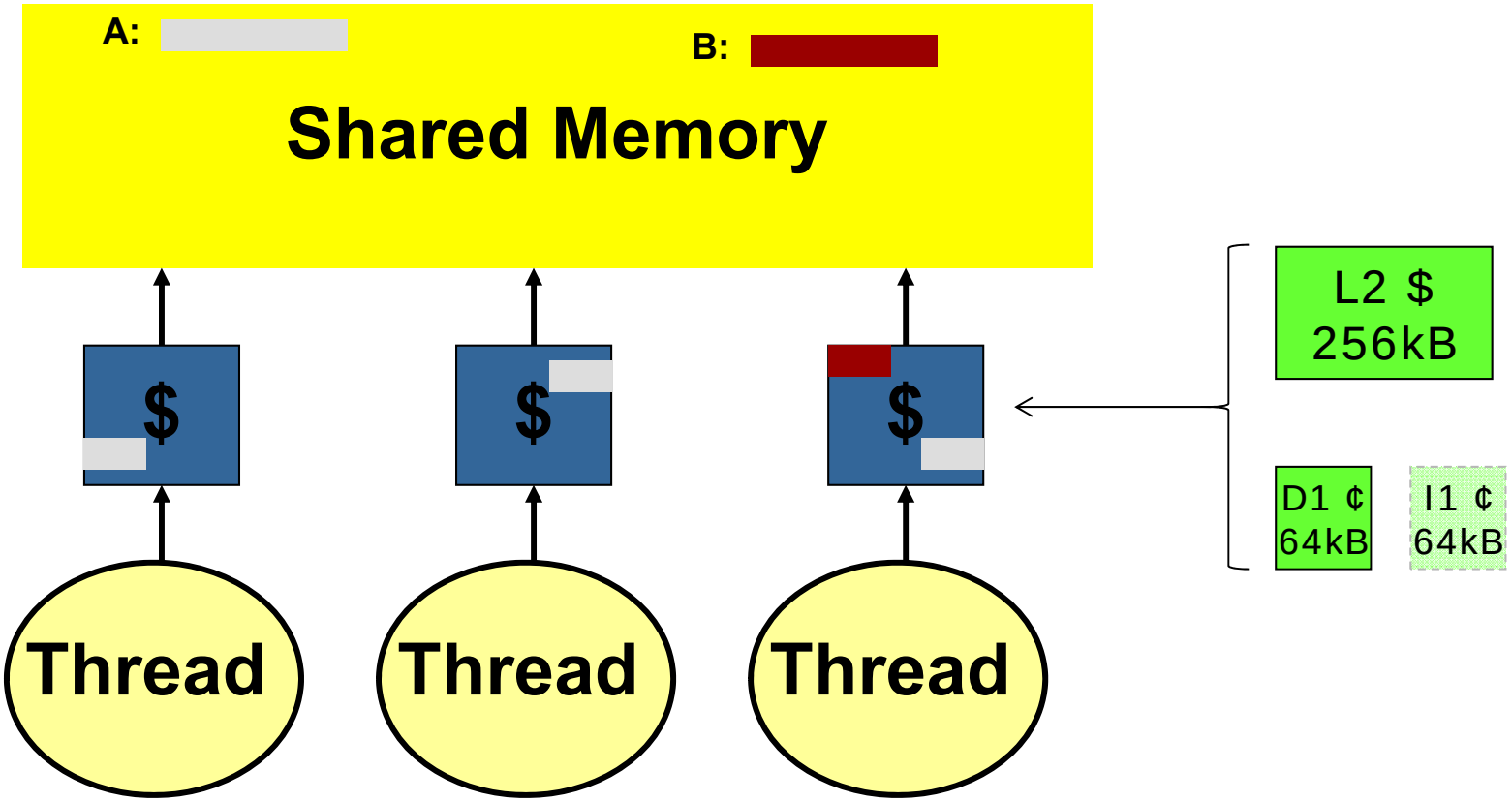
Typical access pattern:

(inner loop stepping through an array)

A, B, C, A+4, B, C, A+8, B, C, ...

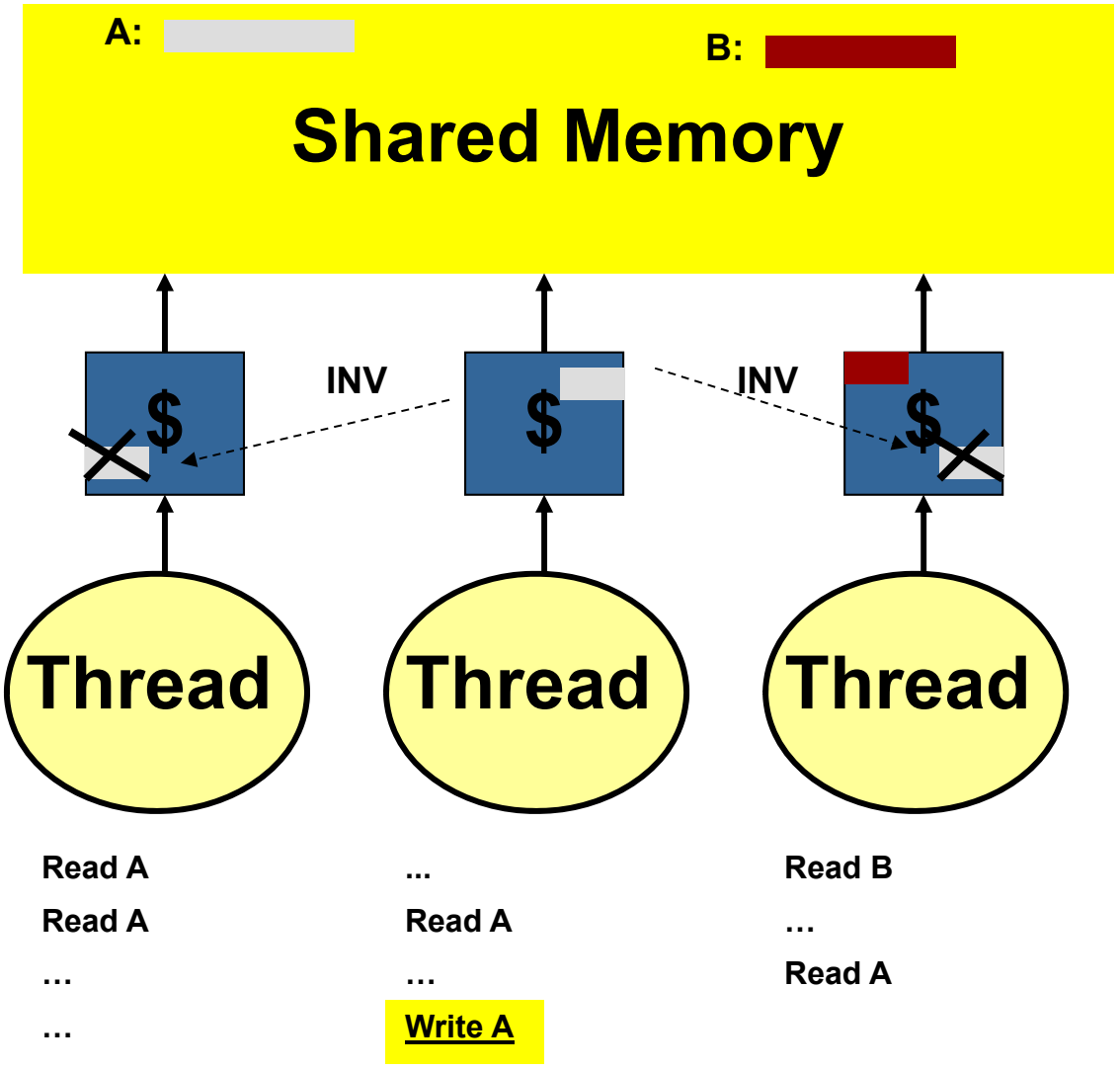


Caches & Coherence
Coherent Replication of Data



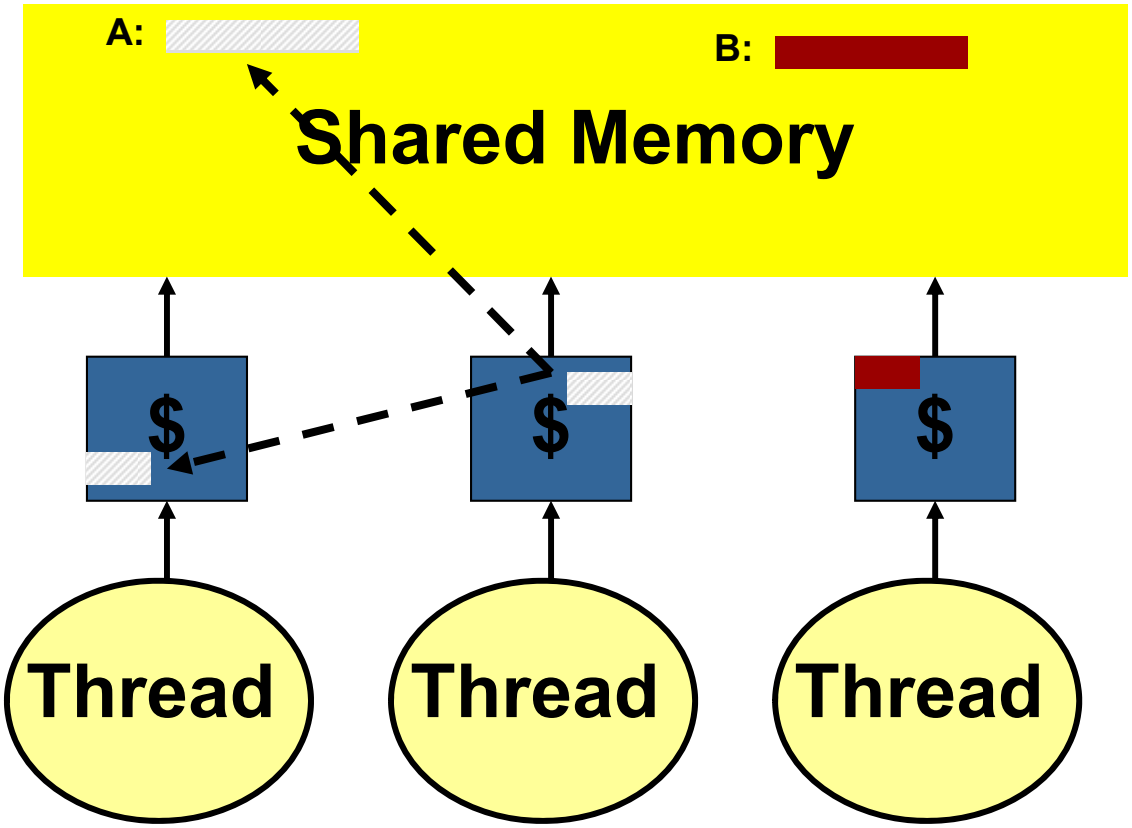
UPPSALA UNIVERSITET

The Cache Coherent Memory Syste



AVDARK 2012

The Cache Coherent 2



UPPSALA UNIVERSITET

Summing up Coherence

There can be many copies of a datum, but only one value

Too strong definition!

There is a single global order of value changes for each datum

AVDARK 2012

UPPSALA UNIVERSITET

Why do you miss in a cache?

- **Capacity miss** – too small cache
- **Conflict miss** – limited associativity
- **Compulsory miss** – accessing data the first time
- **Coherence miss** – The cache would have had the data unless it had been invalidated by someone else
- **Upgrade miss:** (only for writes) – The cache would have had a writable copy, but gave away readable data and "downgraded" itself to read-only state
- **False sharing:** Coherence/downgrade miss is caused by accesses to private data residing in the same cache line

False sharing example:

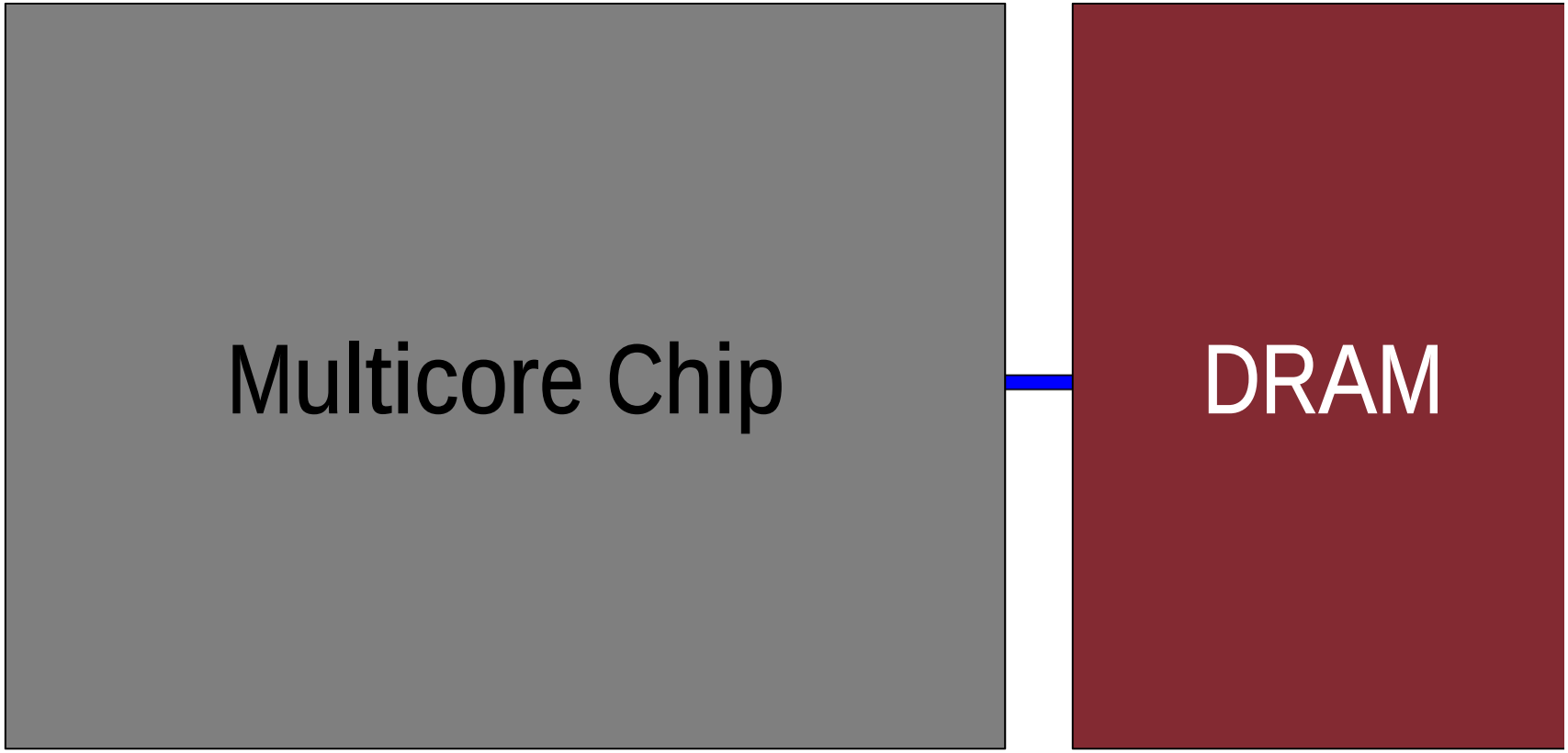
Read A	…
…	Read D
Write A	…
…	Write D
Read A	

cacheline:

A, B, C, D

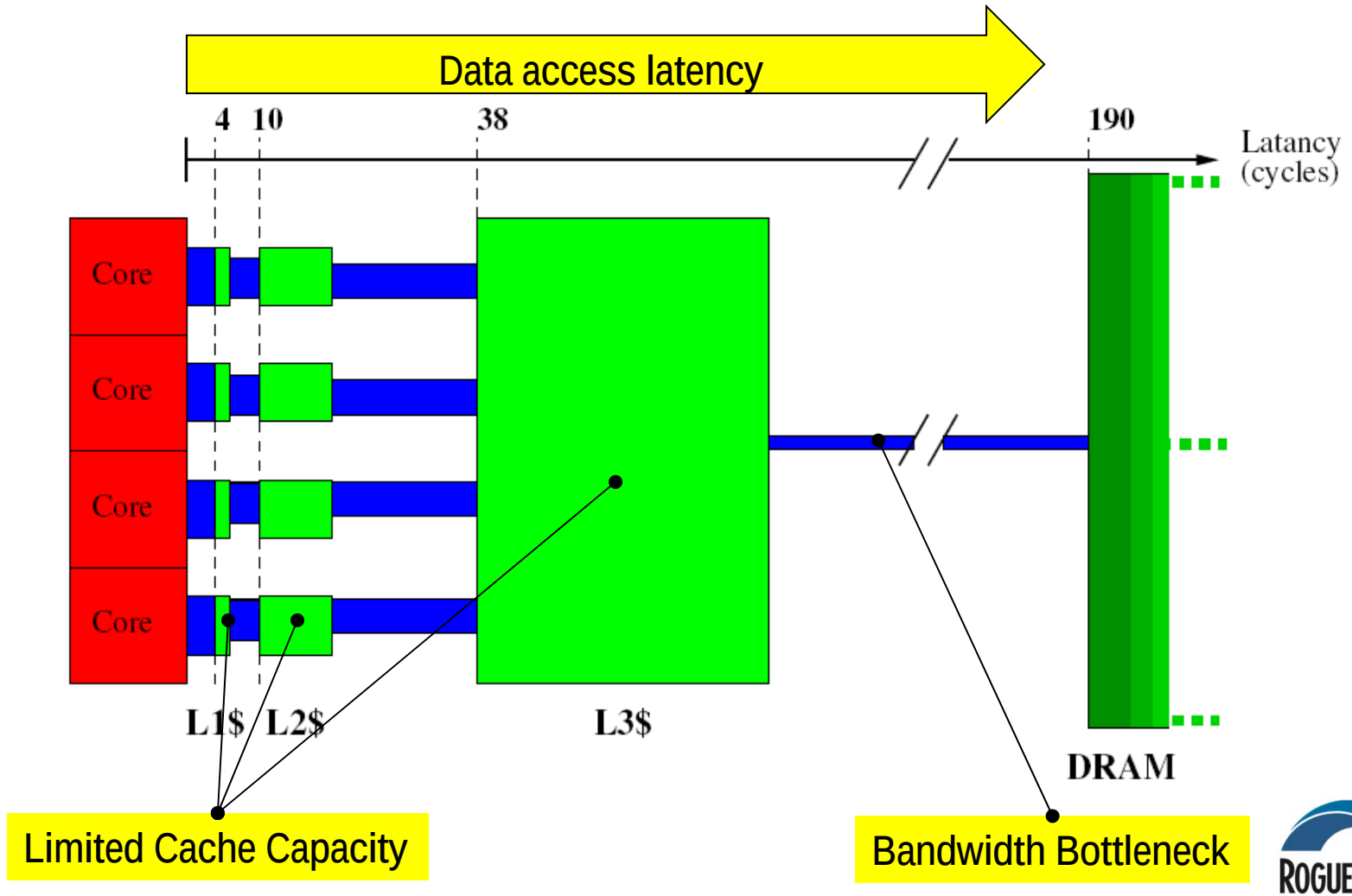
AVDARK 2012

Multicores Underneath the Hood

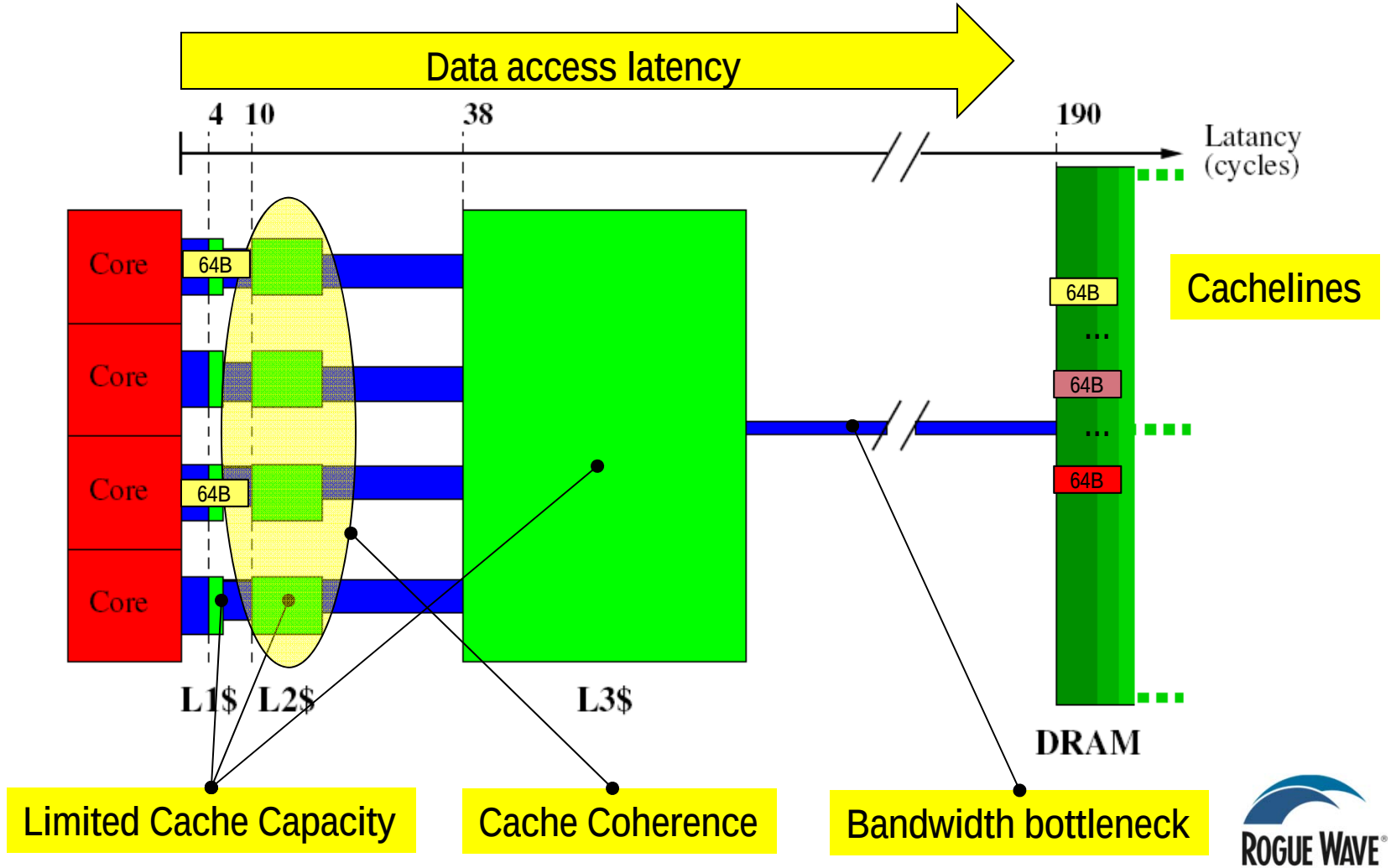


ROGUE WAVE SOFTWARE

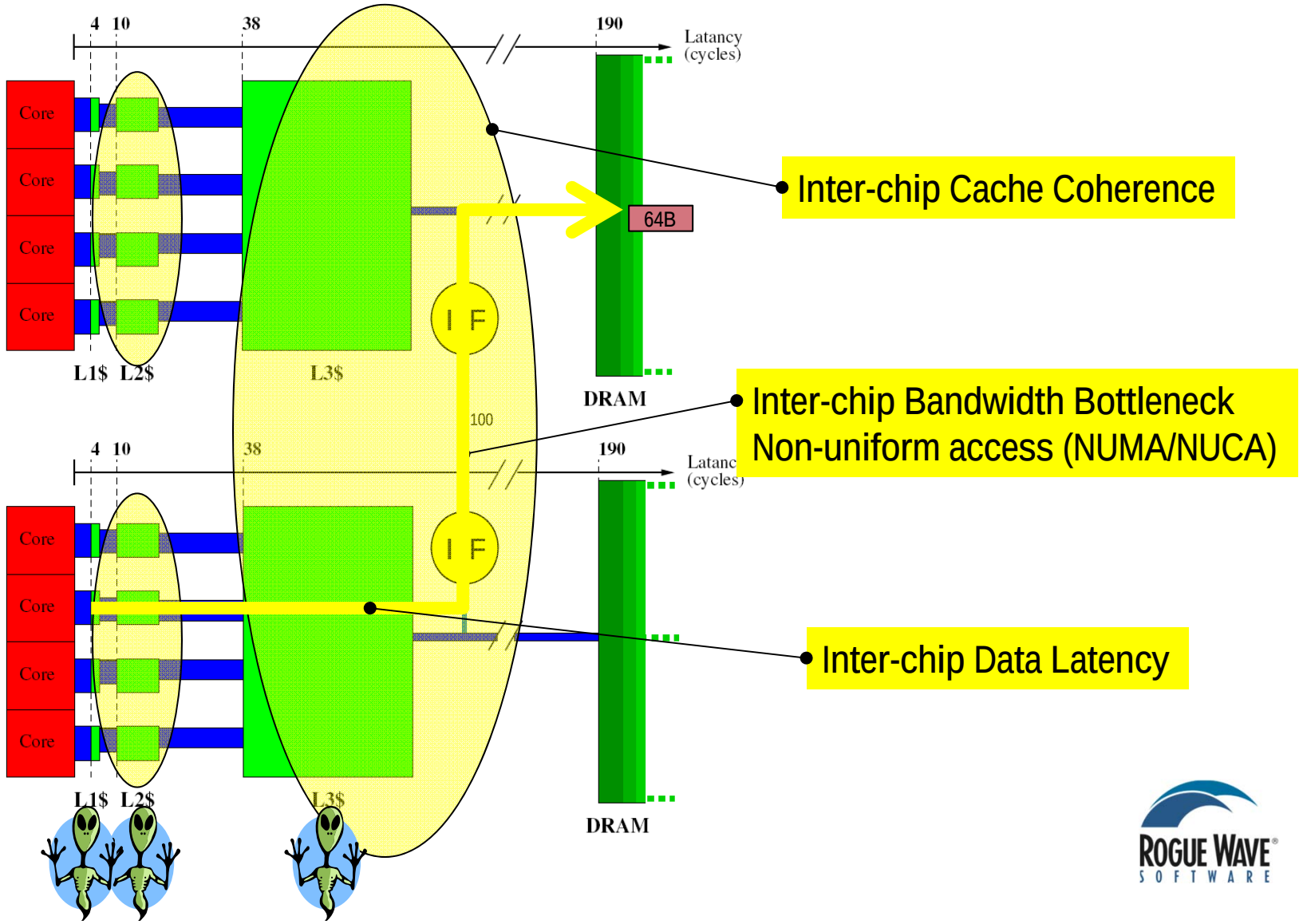
Multicores Underneath the Hood



Multicores Underneath the Hood

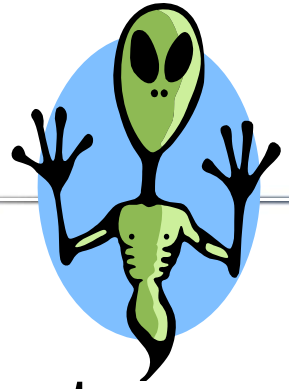


Multisocket – Increasing the Complexity



ROGUE WAVE SOFTWARE

Your friend the HW prefetcher



...a little green man that anticipates your next memory access and “prefetches” the data to a cache

- Sequential prefetching: Sequential streams [to a page]. Some number of prefetch streams supported. Often only for L2 and L3.
- PC-based prefetching: Detects strides from the same PC. Often also for L1.
- Adjacent prefetching: On a miss, also bring in an neighbor cache line. Often only for L2 and L3.

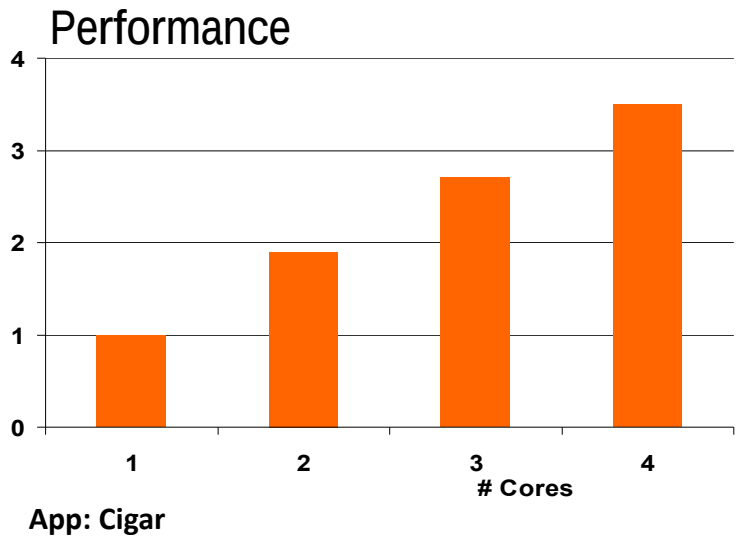
ThreadSpotter: Productive and Simple Optimization

Philosophy of ThreadSpotter

- Automatically collects sparse high-quality runtime data
- Automatically performs detailed performance analysis
- Presents advice on how to change the application
- Context-sensitive manual provides learn-as-you-go tutorial

ROGUE WAVE SOFTWARE

Example: A Scalable Parallel Application

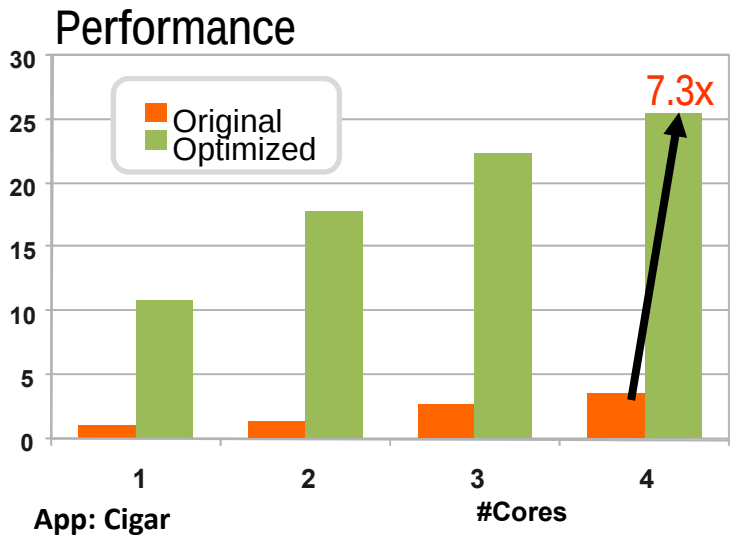


Looks like a perfect scalable application!

Are we done?

ROGUE WAVE SOFTWARE

Example: The Same Application Optimized



Looks like a perfect scalable application!

Are we done?

One simple trick (5 min) made this open source app run 7x faster

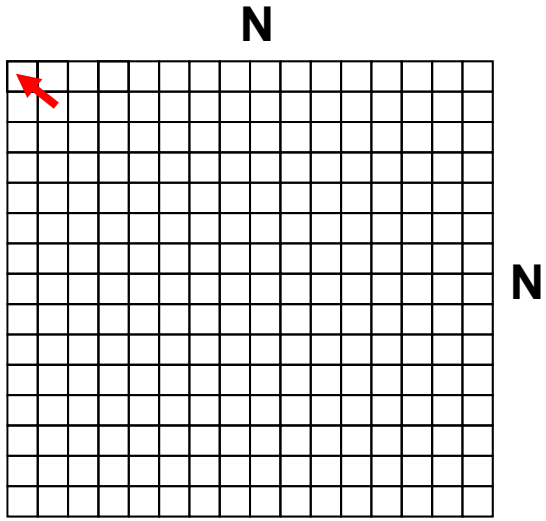
The question remains:

Which one, Where is it and How do I fix it?

ROGUE WAVE SOFTWARE

A Few Examples: What can go Wrong?

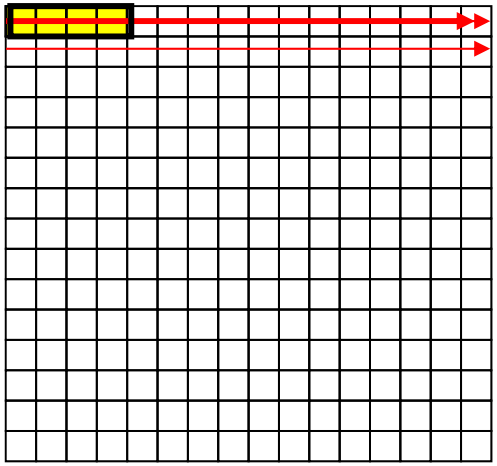
Perform a diagonal copy 10 times



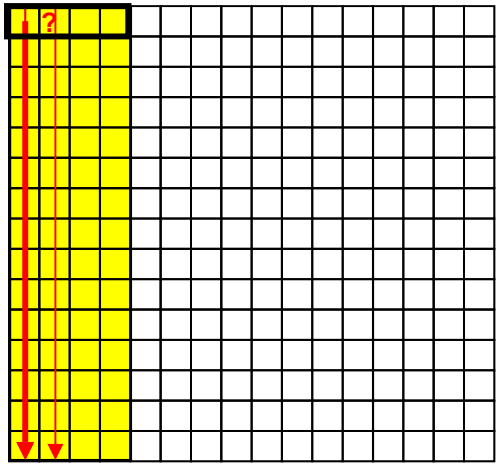
ROGUE WAVE SOFTWARE

Example: Loop order

```
//Optimized Example A
for (i=1; i<N; i++) {
   for (j=0; j<N-1; j++) {
      A[i][j]= A[i+1][j+1];
   }
}
```

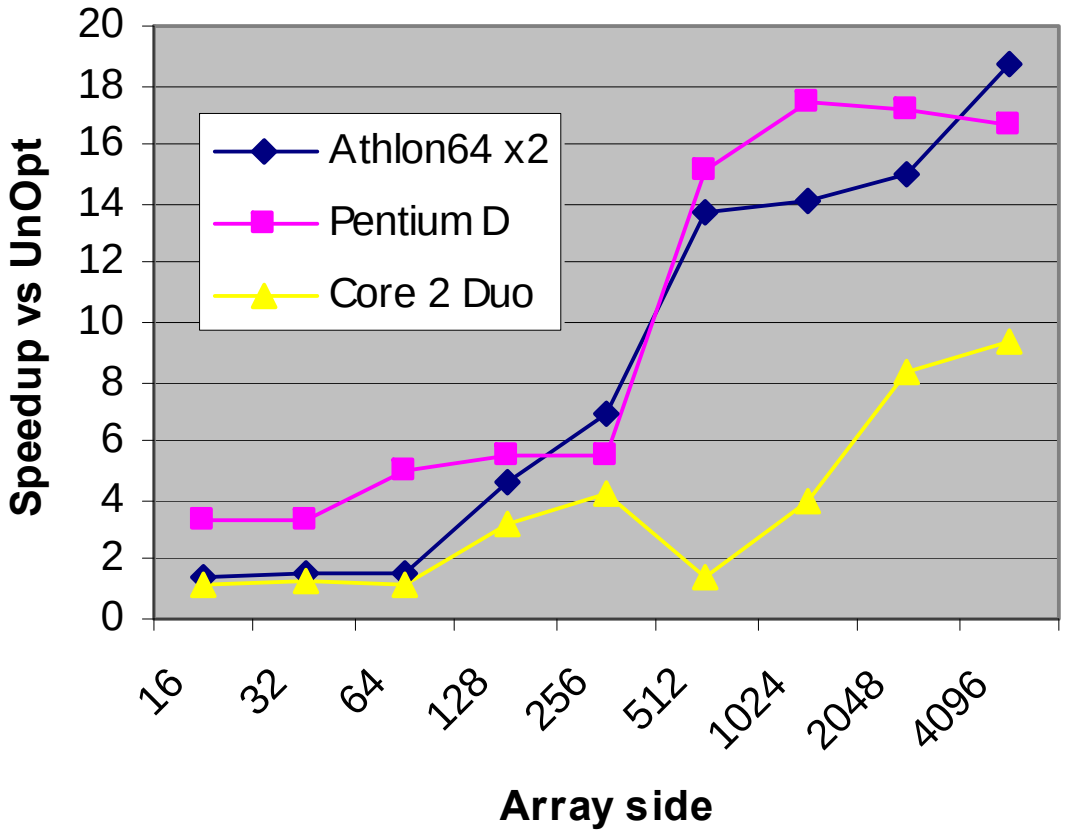


```
//Unoptimized Example A
for (j=1; j<N; j++) {
   for (i=0; i<N-1; i++) {
      A[i][j] = A[i+1][j+1];
   }
}
```



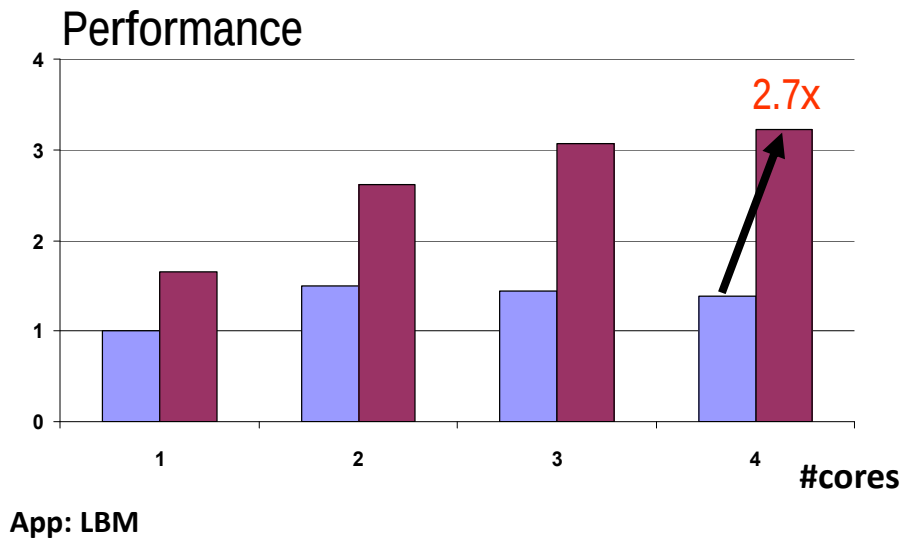
ROGUE WAVE SOFTWARE

Performance Difference: Loop order



ROGUE WAVE SOFTWARE

Example: SPEC2006: LBM



App: LBM

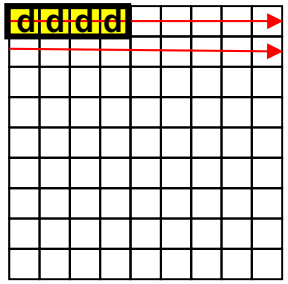
As suggested by ThreadSpotter

➔This fix required five lines of code to change

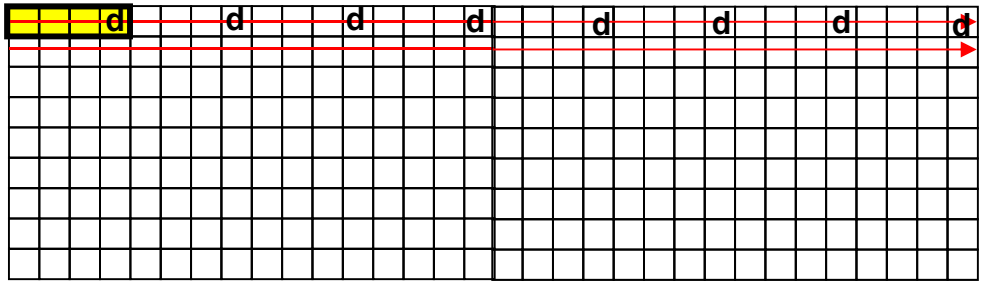
ROGUE WAVE SOFTWARE

Example: Sparse data usage

```
//Optimized Example A
for (i=1; i<N; i++) {
  for (j=1; j<N; j++) {
    A_d[i][j] = A_d[i-1][j-1];
  }
}
```



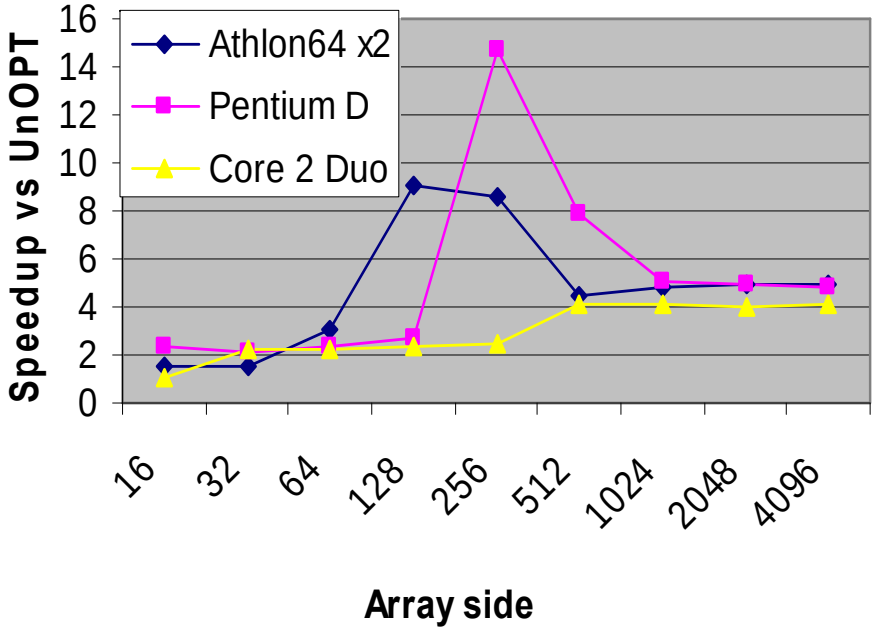

```
//Unoptimized Example A
for (i=1; i<N; i++) {
  for (j=1; j<N; j++) {
    A[i][j].d = A[i-1][j-1].d ;
  }
}
```




```
struct vec_type
{
        char a;
        char b;
        char c;
        char d;
};
```

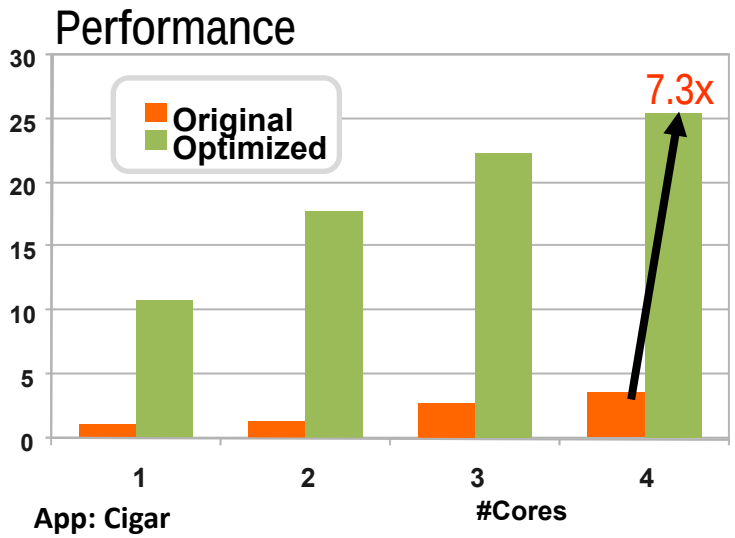

ROGUE WAVE SOFTWARE

Performance Difference: Sparse Data



ROGUE WAVE SOFTWARE

Example Sparse Data: Cigar



Suggestion from ThreadSpotter

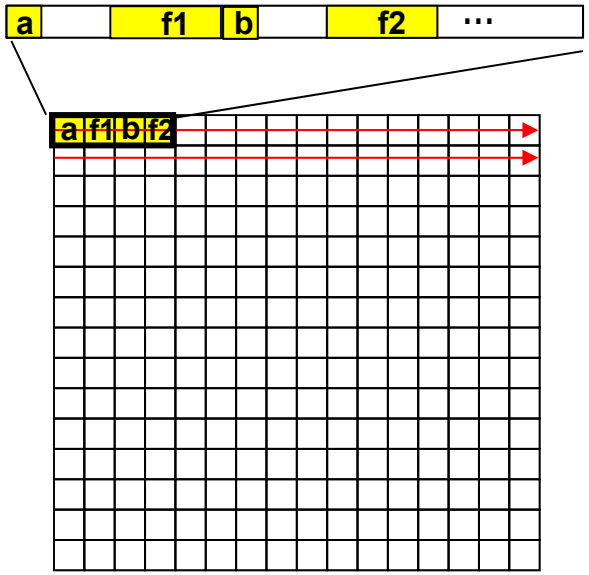
➔ Duplicate one data structure

ROGUE WAVE SOFTWARE

Example: Sparse data allocation

```
sparse_rec sparse [HUGE];

for (int j = 0; j < HUGE; j++)
{
    sparse[j].a = 'a'; sparse[j].b = 'b'; sparse[j].c = 'c'; sparse[j].d = 'd'; sparse[j].e = 'e';
    sparse[j].f1 = 1.0; sparse[j].f2 = 1.0; sparse[j].f3 = 1.0; sparse[j].f4 = 1.0; sparse[j].f5 = 1.0;
}
```

Sparse:




```
struct sparse_rec
{
                    // size 80B
                    char a;
                    double f1;
                    char b;
                    double f2;
                    char c;
                    double f3;
                    char d;
                    double f4;
                    char e;
                    double f5;
};
```

Dense:


```
struct dense_rec
{
                    //size 48B
                    double f1;
                    double f2;
                    double f3;
                    double f4;
                    double f5;
                    char a;
                    char b;
                    char c;
                    char d;
                    char e;
};
```

Performance imprevenemt: 60%

ROGUE WAVE SOFTWARE

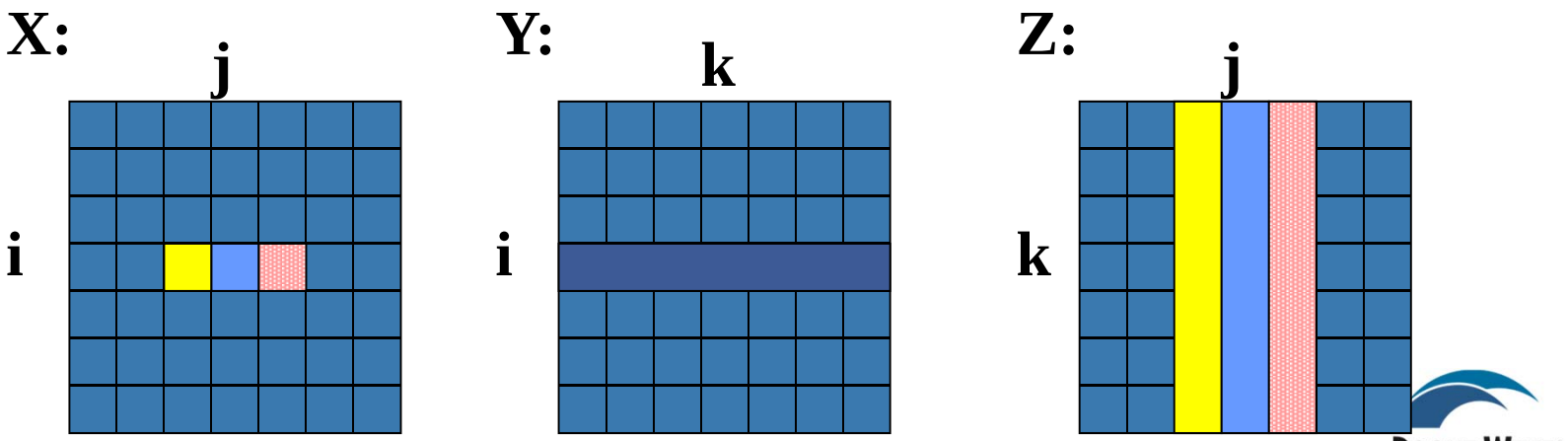
Example: Loop Merging

```
/* Unoptimized */
for (i = 0; i < N; i = i + 1)
   for (j = 0; j < N; j = j + 1)
        a[i][j] = 2 * b[i][j];
for (i = 0; i < N; i = i + 1)
   for (j = 0; j < N; j = j + 1)
        c[i][j] = K * b[i][j] + d[i][j]/2

/* Optimized */
for (i = 0; i < N; i = i + 1)
   for (j = 0; j < N; j = j + 1){
        a[i][j] = 2 * b[i][j];
        c[i][j] = K * b[i][j] + d[i][j]/2;
}
```

Performance imprevenemt: 50%

Blocking

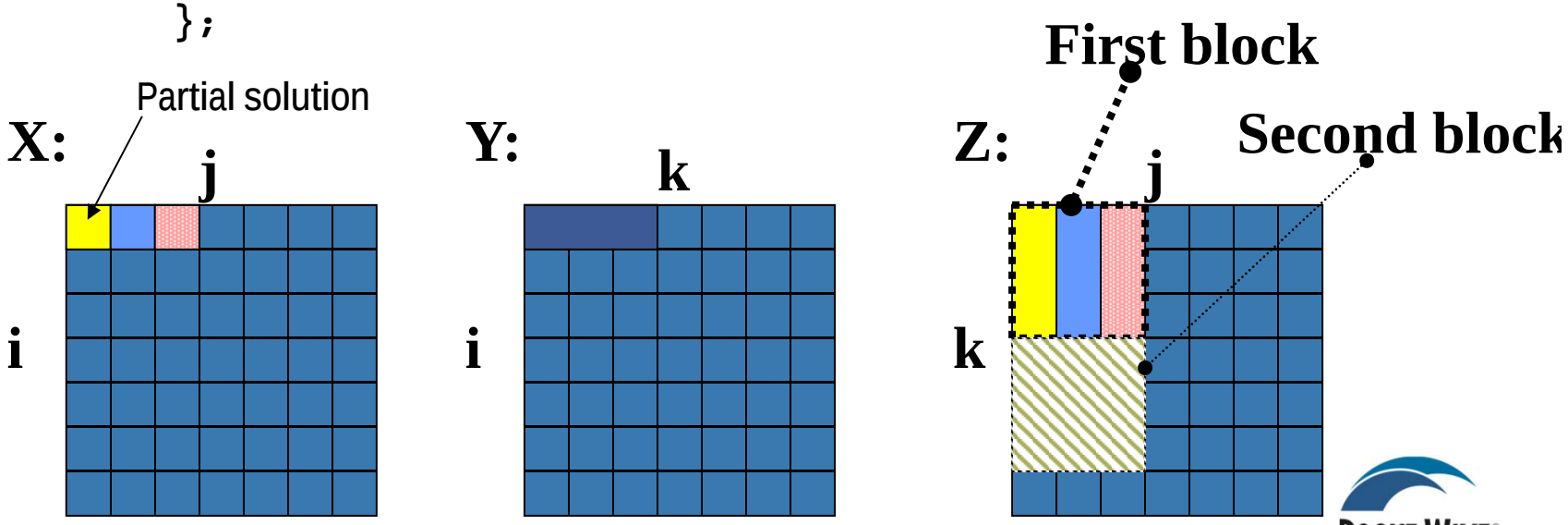
```
/* Unoptimized  ARRAY: x = y * z */
for (i = 0; i < N; i = i + 1)
   for (j = 0; j < N; j = j + 1)
        {r = 0;
         for (k = 0; k < N; k = k + 1)
                r = r + y[i][k] * z[k][j];
         x[i][j] = r;
        };
```



ROGUE WAVE SOFTWARE

Temporal Blocking

```
/* Optimized  ARRAY: X = Y * Z */
for (jj = 0; jj < N; jj = jj + B)
for (kk = 0; kk < N; kk = kk + B)
for (i = 0; i < N; i = i + 1)
   for (j = jj; j < min(jj+B,N); j = j + 1)
        {r = 0;
         for (k = kk; k < min(kk+B,N); k = k + 1)
                r = r + y[i][k] * z[k][j];
         x[i][j] += r;
        };
```



Performance imprevenemt: >10x

ROGUE WAVE SOFTWARE

Coherence traffic

ORIG:

Thread 0:

```
int a, total;
spawn_child()
for (int i; i< HUGE; i++) {
  /* do some work */
   a++;
 }
join()
total = a;
```

Child (Thread 1)

```
for (int i; i< HUGE; i++) {
   /* do some work*/
   a++;
}
```

OPT:

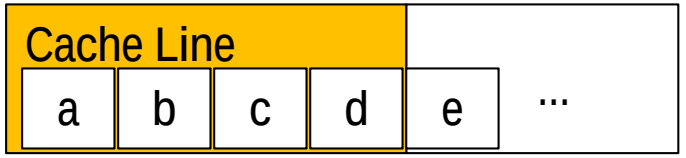
Thread 0:

```
int a, total;
spawn_child()
for (int i; i< HUGE; i++) {
   /* do some work */
   a++;
 }
join()
total += a;
```

Child (Thread 1)

```
int b;
for (int i; i< HUGE; i++) {
   /* do some work */
   b++;
 }
total += b;
```

False sharing



ORIG:

Thread 0:

```
int a, b;
spawn_child()
for (int i; i< HUGE; i++) {
  ...
   a++;
 }
join()
total = a + b;
```

Child (Thread 1)

```
for (int i; i< HUGE; i++) {
  ...
   b++;
}
```

OPT:

Thread 0:

```
int a;
spawn_child()
for (int i; i< HUGE; i++) {
  ...
   a++;
 }
join()
total += a;
```

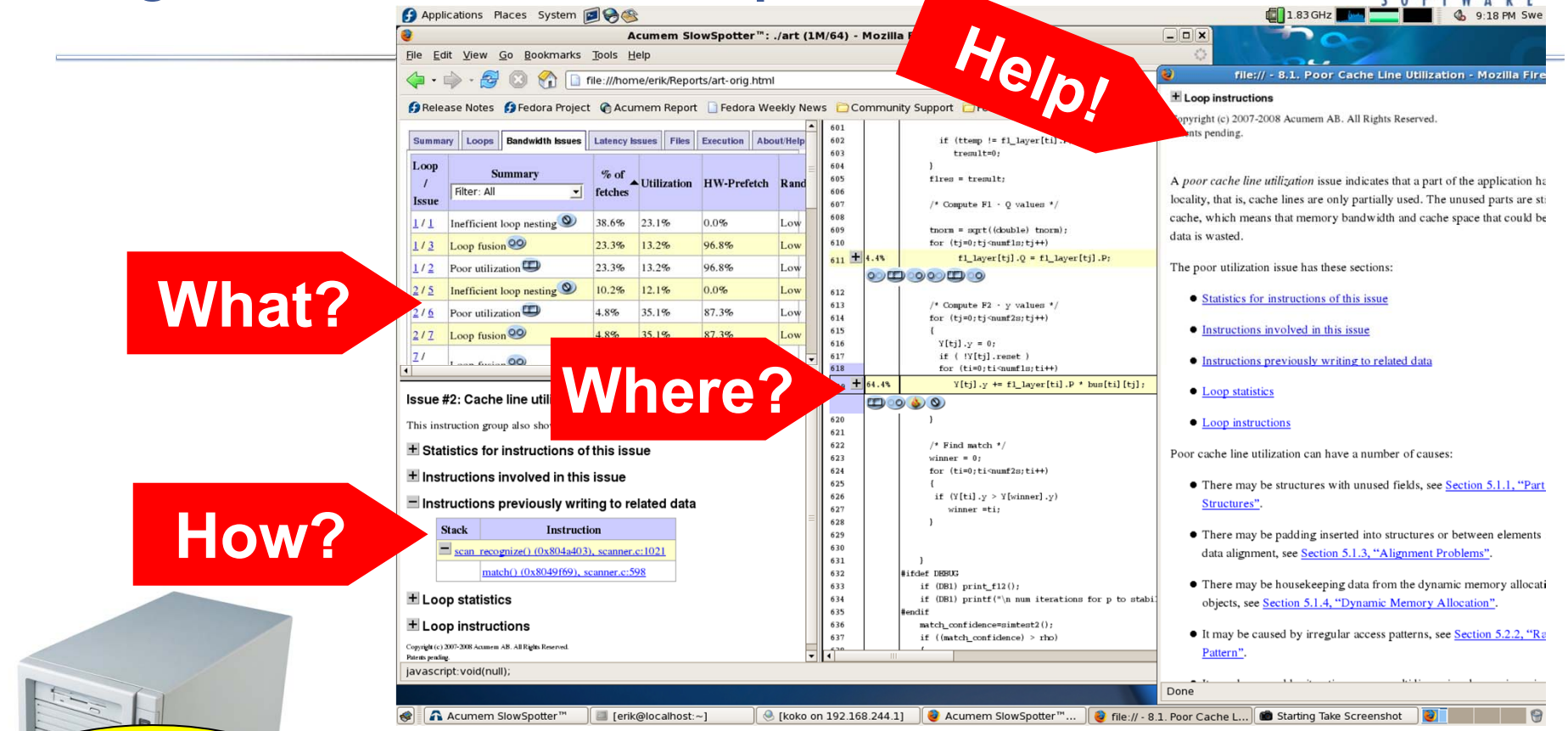
Child (Thread 1)

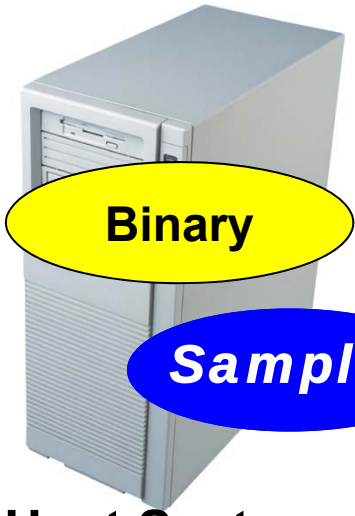
```
int b;
for (int i; i< HUGE; i++) {
  ...
   total += b;
}
```

Performance imprevenemt: 10x

ROGUE WAVE SOFTWARE

Rogue Wave: ThreadSpotter





Binary

Sampler

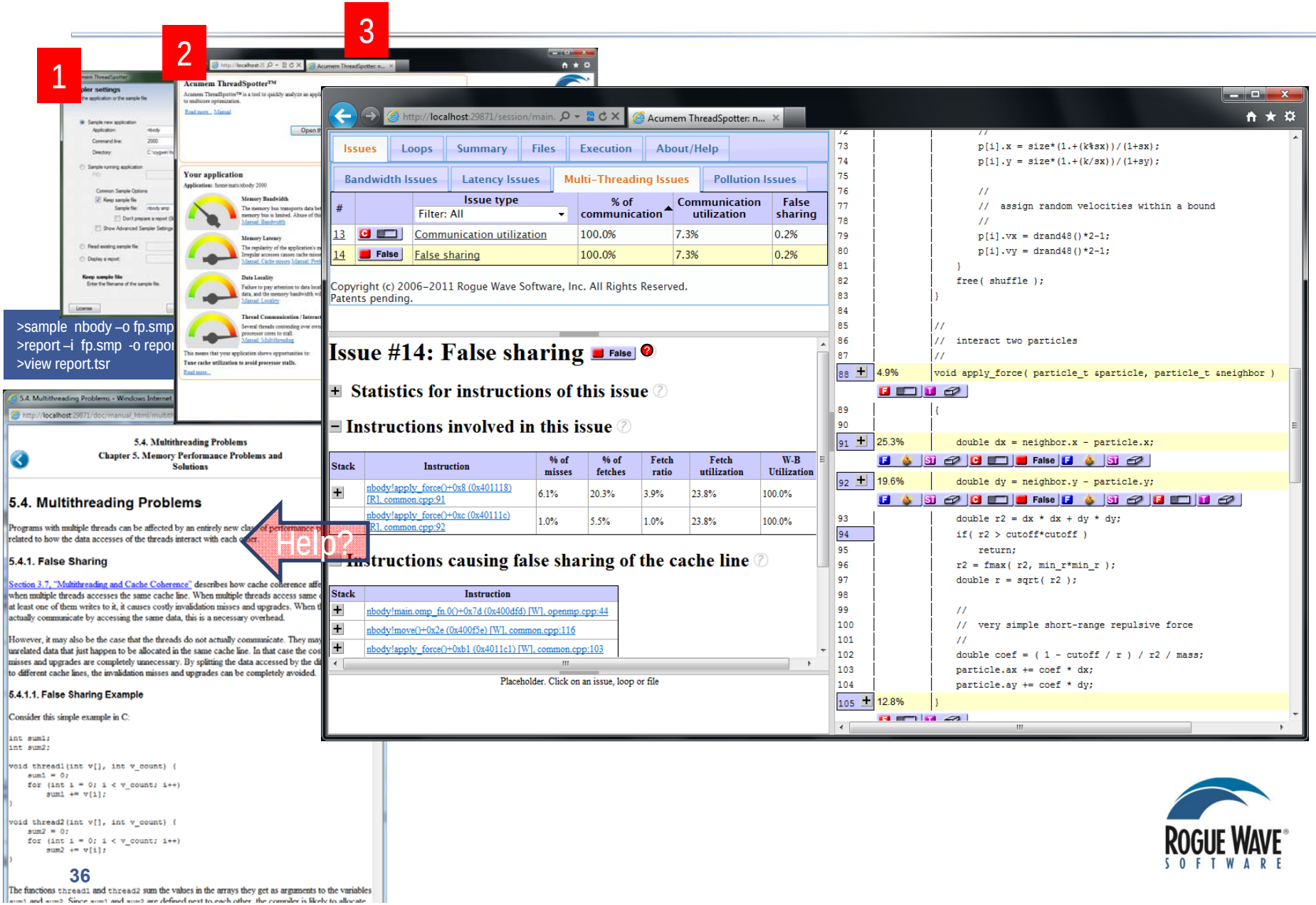
Host System

Sample file (~4MB)

Off-line analysis

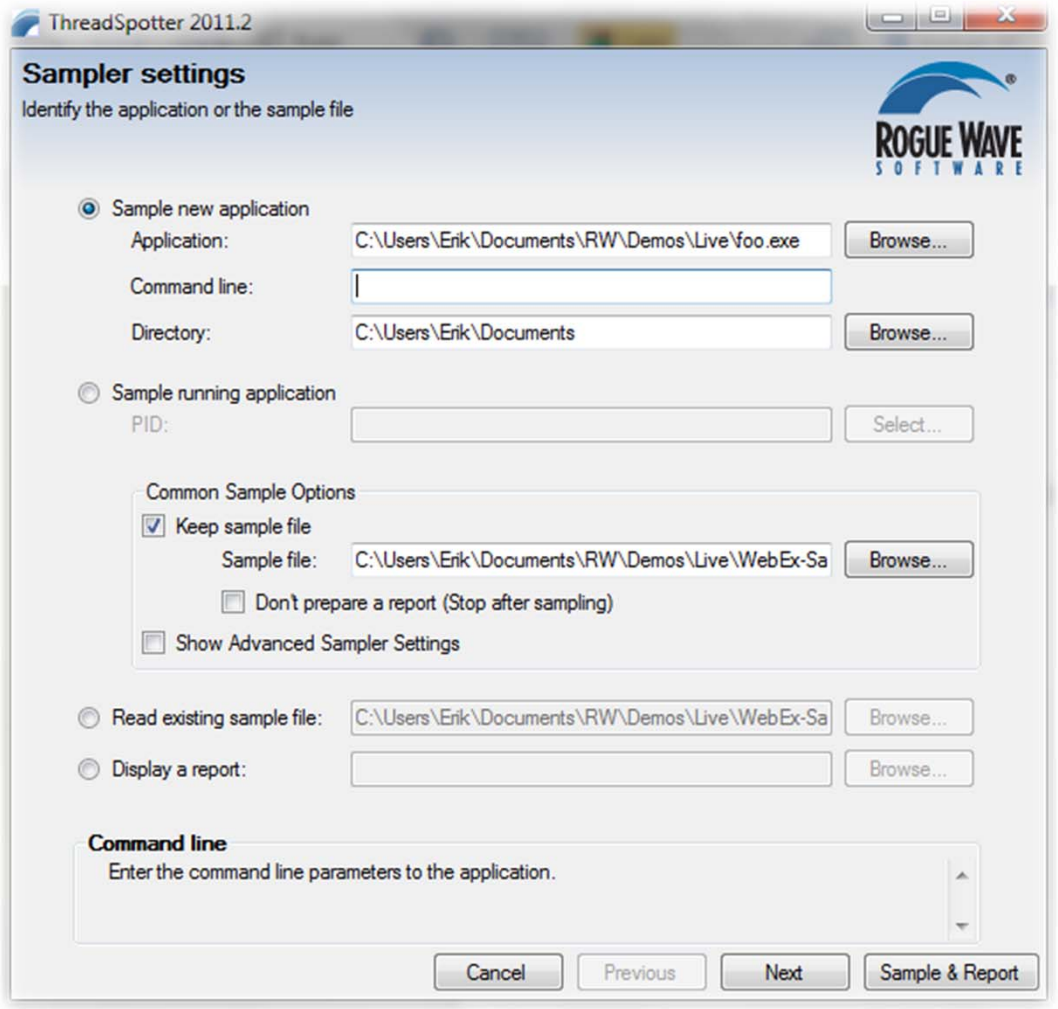
Target System Parameters

ThreadSpotter Automatically Find and Explains Problems



ROGUE WAVE SOFTWARE

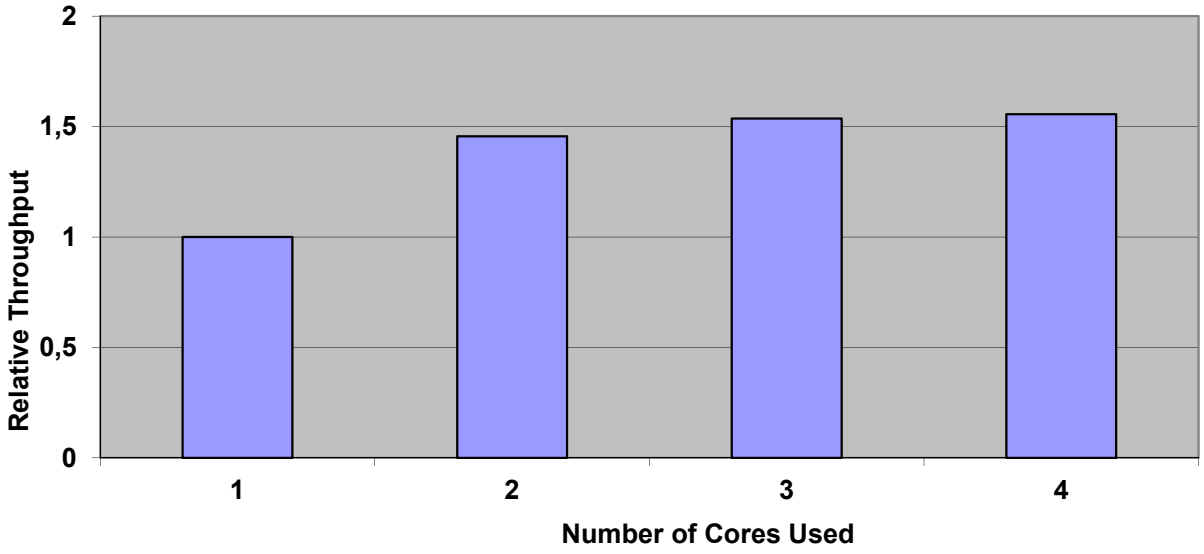
GUI Interface (Here: Windows version)



Resource Sharing Example

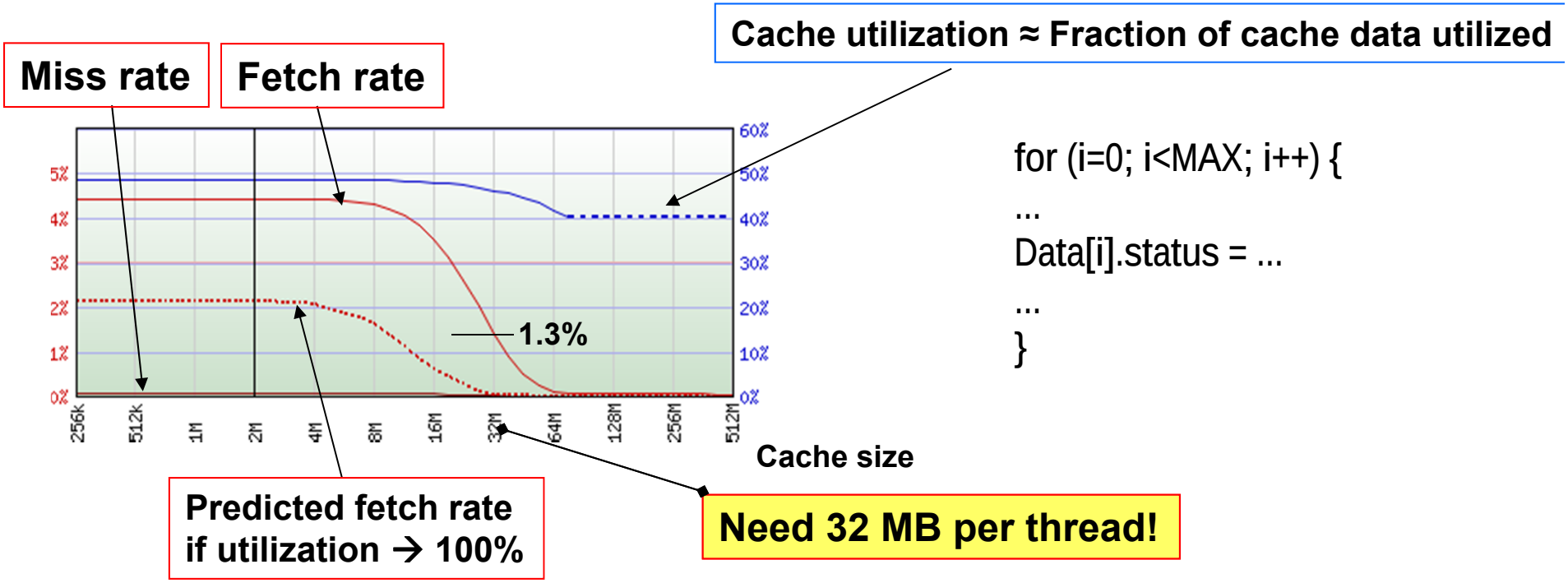
Libquantum
A quantum computer simulation
Widely used in research (download from: http://www.libquantum.de/)
4000+ lines of C, fairly complex code.
Runs an experiment in ~30 min

Original performance



Demo:
LQ
LQ-s
LQ-sl

Utilization Analysis



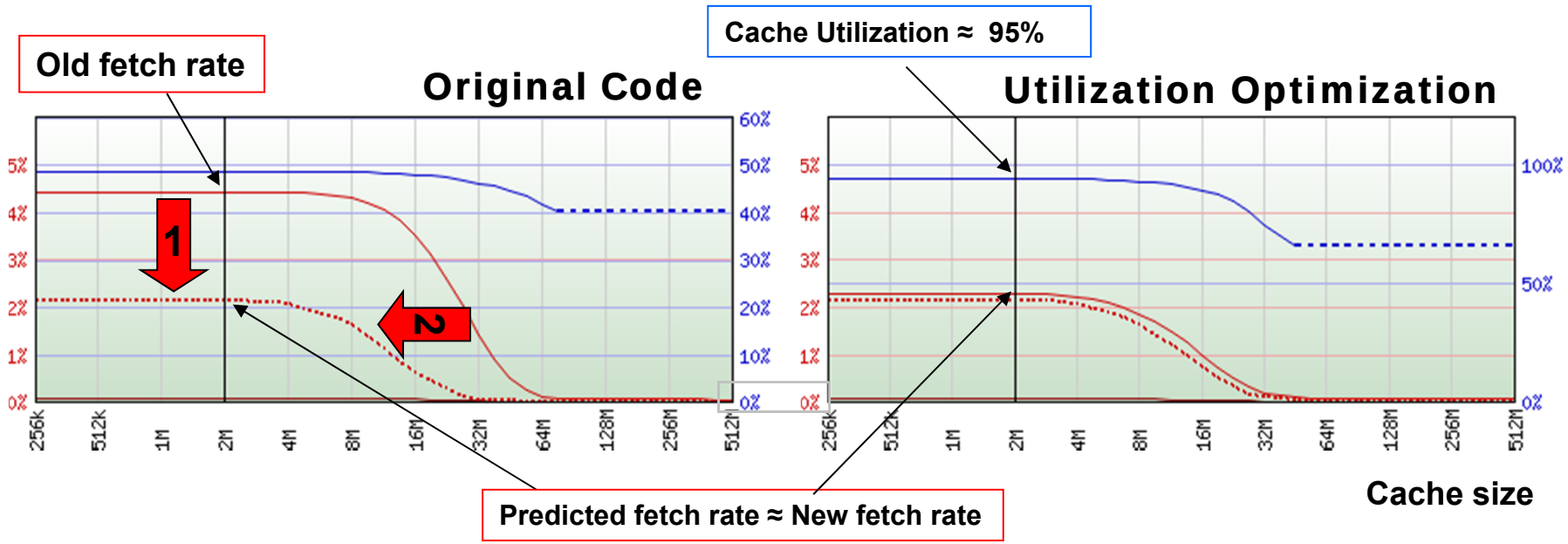
ThreadSpotter's First Advice: Improve Cache Utilization

➔Change one data structure

- Involves ~20 lines of code
- Takes a non-expert 30 min

39

After Utilization Optimization

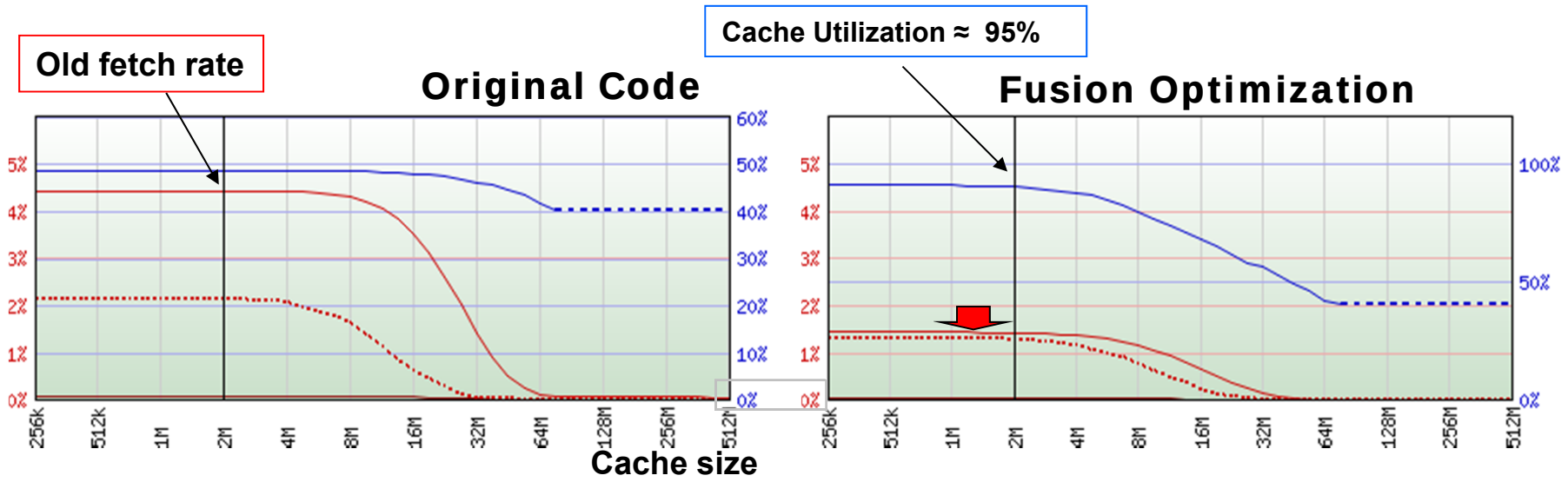


Second ThreadSpotter Advice: Improve reuse of data (loop fusion)

➔Fuse functions traversing the same data

- Here: four fused functions created
- Takes a non-expert < 2h

After Loop Fusion



- Fetch rate down to 1.3% for 2MB
- Same as a 32 MB cache originally

Demo:

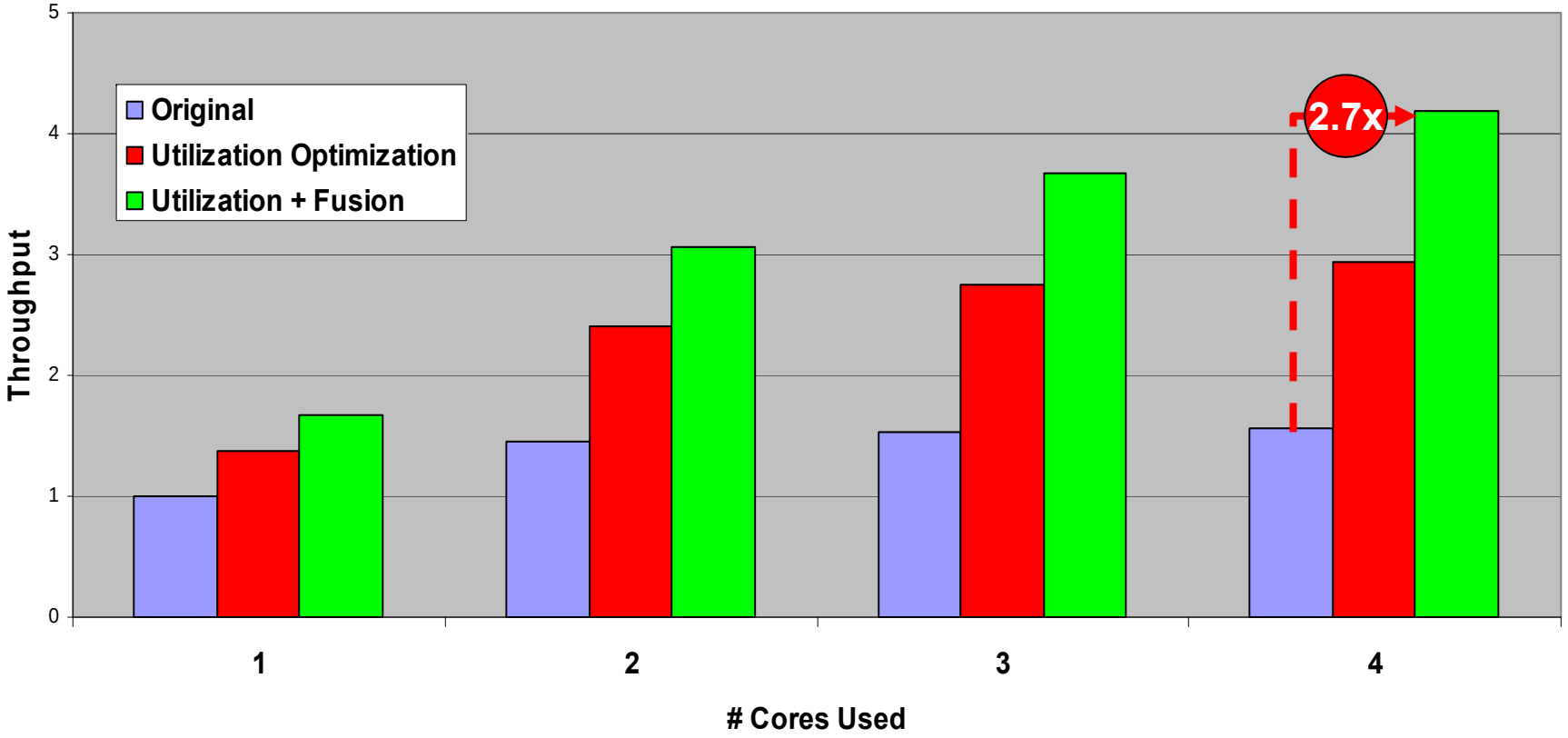
LQ

LQ-s

LQ-sl

ROGUE WAVE SOFTWARE

Summary: Libquantum



ROGUE WAVE SOFTWARE

Demo Links

Demo:

Cigar

LQ
LQ-s
LQ-sl

art
art-opt

Demo MT:

GS
GS-bind

ThreadSpotter CMD Line Interface Overview

- Sample:

  >sample –o samplefile.smp –r ./my-binary

Samples the program ./my-binary, producing samplefile.smp

- Make a report

  >report –i samplefile.smp –o report.tsr

Prepares a report file named report.tsr

- View the report

  >view report.tsr

View the report in an html browser

ROGUE WAVE SOFTWARE

Sampler settings: Starting the sampling

- Starting sampling
  - run (--run, -r)

    sample -r ./foobar arg1 arg2
  - pid (--pid, -p)

    sample -p 4711
  - wait (--wait -w)

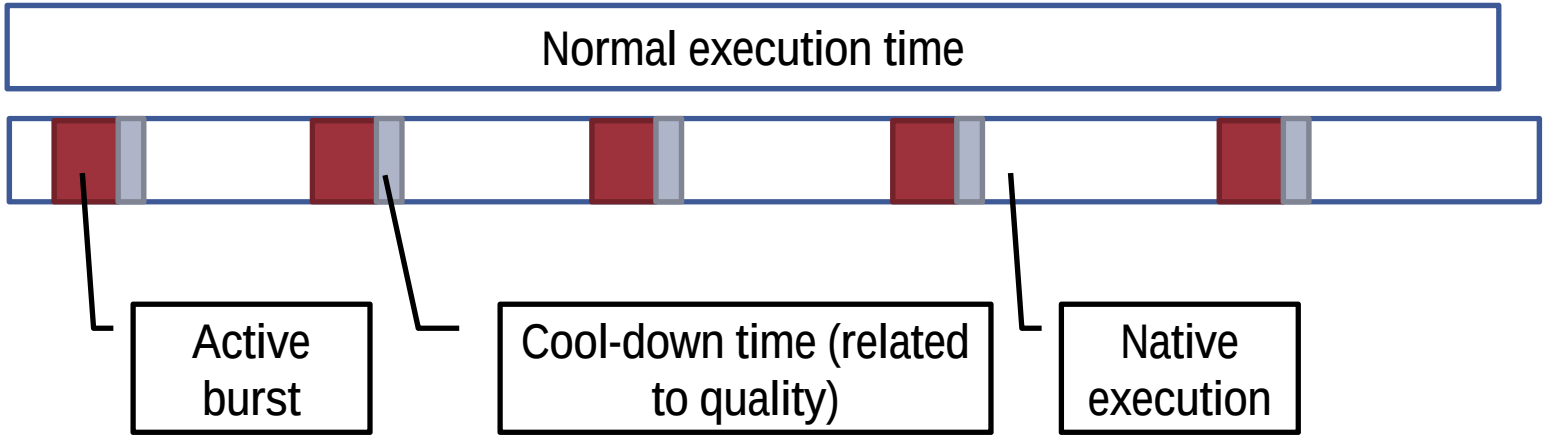
    sample -w gimp

Sampler settings: Start and stop control

- 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>

Sampler settings: Batch sampling

Burst mode enables sampling long executions with low overhead



- Burst time (approximate normal execution time [minutes])
- Burst quality

Sampler settings: Advanced options

- Advanced options
  - Sampler period
    - Initial period, automatically reduced, default 100000, decrease for short running applications
  - Cache line sizes
    - Select all the cache line sizes that you want to prepare reports for
    - Default 64 bytes.

Report settings: Specifying architecture

- Override architecture parameters
  - cpu (--cpu help, --cpu intel/nehalem_ex_4_8_12288)
  - level (Override target cache level, --level 2)
  - number-of-caches (Override number of caches at the current level, --number-of-caches 8)
  - cache-size (Override target cache size, --cache-size 6M)
  - line-size (Override target cache line size, you need to have sampled the same cache line size as well.)
  - replacement (Select the replacement model {random, lru})

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

Report settings: Misc

- Report generation
  - title
  - no-intro
  - input (input file)
  - output (report directory filename)
  - verbose (increase debug output level)
  - low-mem (don't cache too much internally)

Demo MT:

GS
GS-bind

- Multithreaded grouping
  - group

    List the thread-ids that are considered to share a cache on the target cache level

ThreadSpotter: Productive and Simple Optimization

ThreadSpotter

- Zero ramp-up time, learn-as-you-go tool
- Productivity: Automatically collects runtime info
- Productivity: Automatically performs the analysis
- Makes the expert more productive
- Enables non-experts to optimize
- Bells and whistles available when you are ready...

ROGUE WAVE SOFTWARE

Rogue Wave: Independent cross-platform tools

- Brings P:s to high-performance computing
  - Productivity
  - Portability
  - Performance
  - Power-savings

1. Domain-experts to prototype in hours (PyIMSL)
2. Simple path to portable and efficient production code (IMSL)
3. Top-of-the-line parallel debugging environment (TotalView)
4. Automatic analysis for better efficiency (ThreadSpotter)

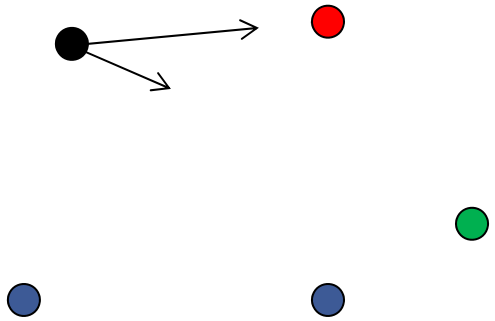
✓ Human productivity -- get results faster with less effort

✓ Hardware productivity -- good usage of HW investments and energy

ROGUE WAVE SOFTWARE

N-body Overview

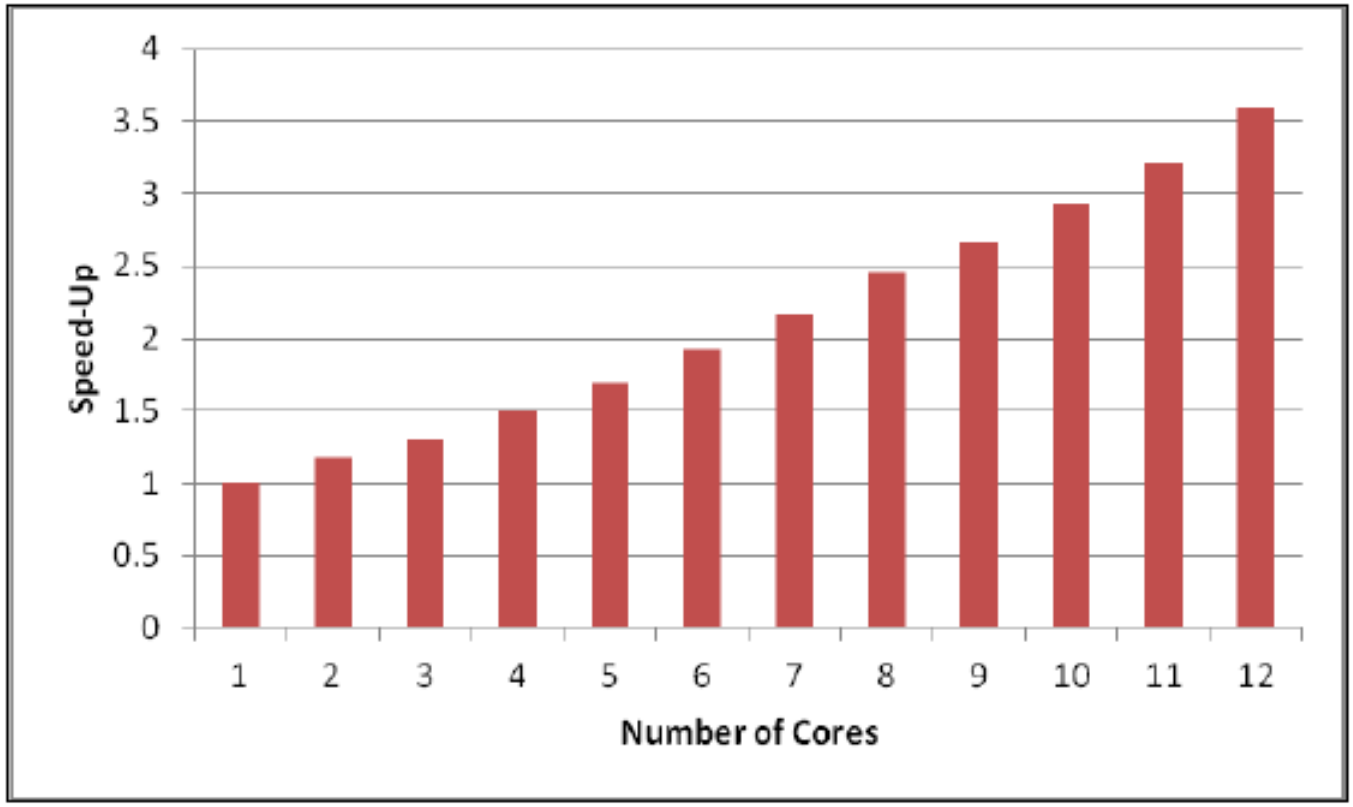
```
#omp parallel
for (i=0; i<MAX, i++)
  for (j=i; j<MAX, j++) {
   Force[i] += calculate_forces(i,j);
   Force[j] += calculate_forces(i,j);
}

#omp parallel
for (i=0; i<MAX, i++) {
  Velocity[i] = f(Forces[i], Velocity[i]);
  Position[i] = g(Velocity[i], Position[i]);
}
```



```
typedef struct {
        // Position
        double px, py;
        // Velocity
        double vx, vy;
        //Mass
        double m;
        //Force
        double fx, fy;
} body_t
```

 | | erik.hagersten@roguewave.com

ROGUE WAVE SOFTWARE



 | | erik.hagersten@roguewave.com

N-body Overview

```
#omp parallel
for (i=0; i<MAX, i++)
  for (j=i; j<MAX, j++) {
   Force[i] += calculate_forces(i,j);
   Force[j] += calculate_forces(i,j);
}

#omp parallel
for (i=0; i<MAX, i++) {
  Velocity[i] = f(Forces[i], Velocity[i]);
  Position[i] = g(Velocity[i], Position[i]);
}
```

```
typedef struct {
        // Position
        double px, py;
        // Velocity
        double vx, vy;
        //Mass
        double m;
} body_t

typedef struct {
        //Force
        double fx, fy;
} force_t
```

 | | erik.hagersten@roguewave.com

ROGUE WAVE SOFTWARE

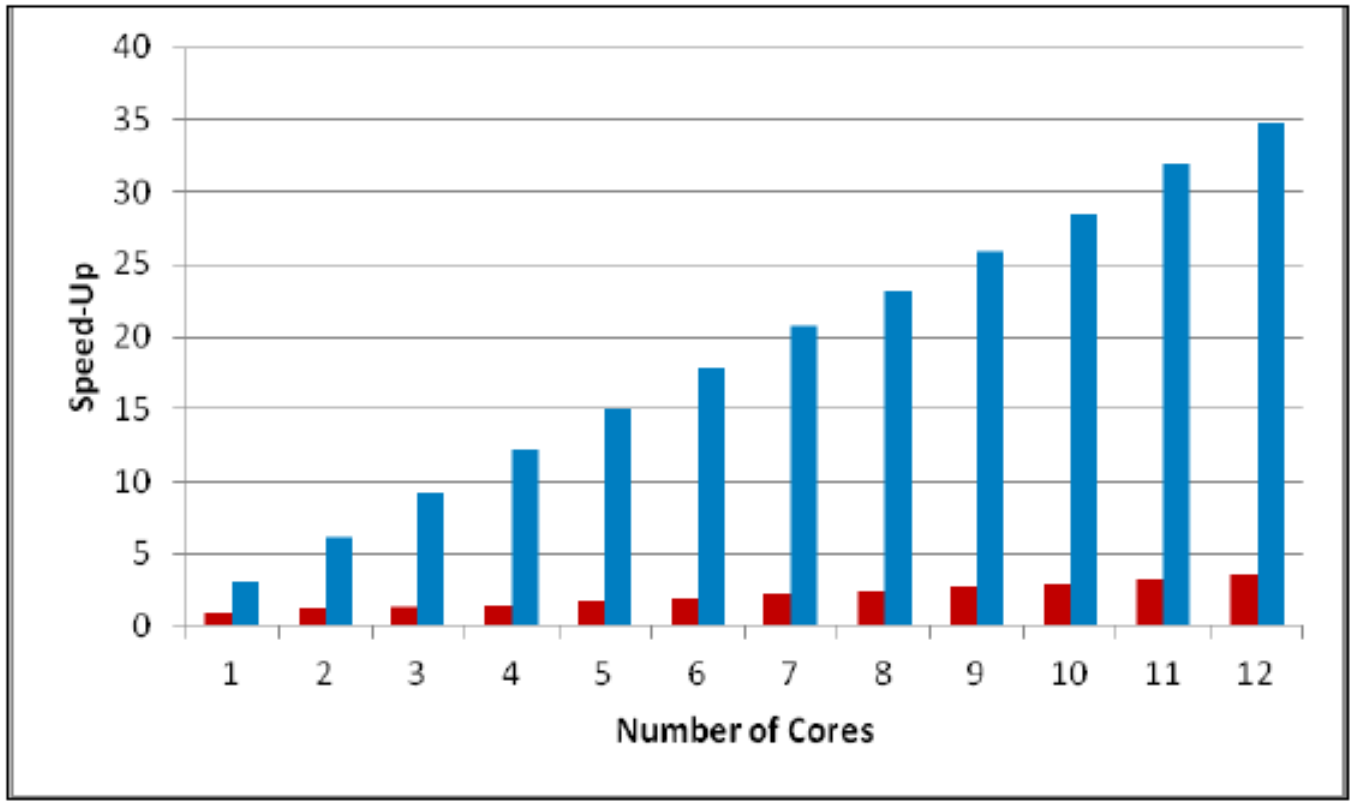
N-body Overview

```
#omp parallel
for (i=0; i<MAX, i++)
  for (j=i; j<MAX, j++) {
   local_Force[my_thread_id,i] += calculate_forces(i,j);
   local_Force[my_thread_id,j] += calculate_forces(i,j);
}

#omp parallel
for (i=0; i<MAX, i++) {
  for (t=0; t<number_of_threads, t++) {
     Force[i] = local_Force[t,i];
  }
  Velocity[i] = f(Forces[i], Velocity[i]);
  Position[i] = g(Velocity[i], Position[i]);
}
```

```
typedef struct {
        // Position
        double px, py;
        // Velocity
        double vx, vy;
        //Mass
        double m;
} body_t

typedef struct {
        //Force
        double fx, fy;
} force_t
```


erik.hagersten@roguewave.com

ROGUE WAVE SOFTWARE



 | | erik.hagersten@roguewave.com

Thank You

Contact me for the Guided Evaluation Kit