Event Based Analysis

Stephen Blair-Chappell

Intel Compiler Labs

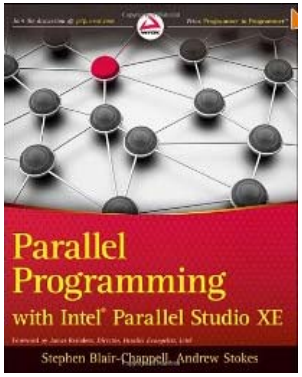
This training relies on you owning a copy of the following…

Parallel Programming with Parallel Studio XE

Stephen Blair-Chappell & Andrew Stokes

Wiley ISBN: *9780470891650*

Part I: Introduction
1: Parallelism Today
2: An Overview of Parallel Studio XE
3: Parallel Studio XE for the Impatient



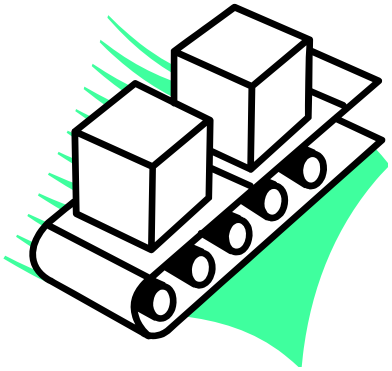
Part II: Using Parallel Studio XE
4: Producing Optimized Code
5: Writing Secure Code
6: Where to Parallelize
7: Implementing Parallelism
8: Checking for Errors
9: Tuning Parallelism
10: Advisor-Driven Design
11: Debugging Parallel Applications
12:Event-Based Analysis with VTune Amplifier XE

Part III :Case Studies
13: The World's First Sudoku 'Thirty-Niner'
14: Nine Tips to Parallel Heaven
15: Parallel Track-Fitting in the CERN Collider
16: Parallelizing Legacy Code

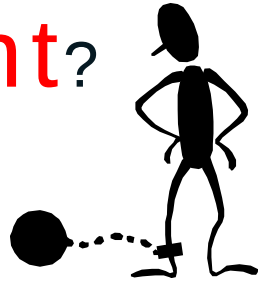
Testing the Health of an Application

Does it run Fast?

Does it get through lots of work?



Is any part of the code inefficient?



A program's performance

A program's performance can be impacted by

- System-wide activity
- Application Heuristics
- CPU architecture

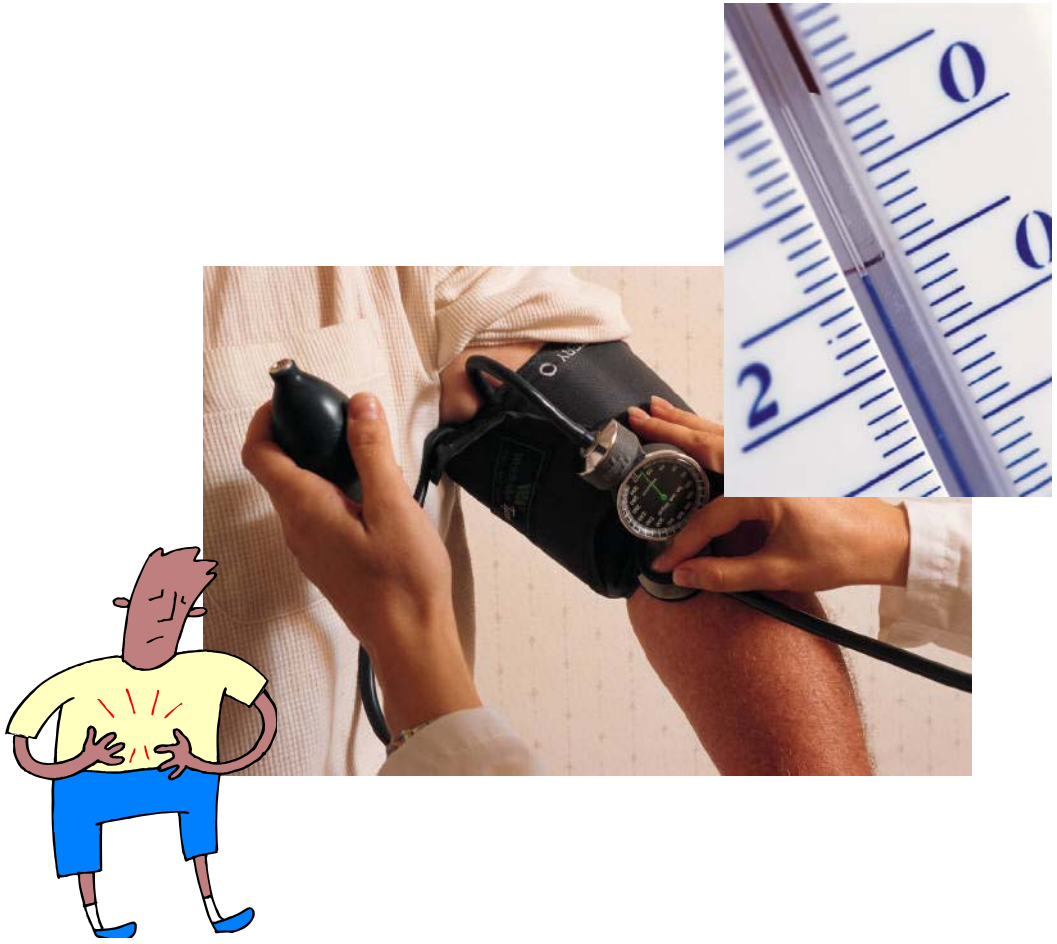
Any analysis Should be Done in This order

Are you Sick?

Fever?

High Pressure?

Aches & Pains



‘Is my program unwell?’

- Number of Cycles (clock ticks) a program consumes
- Number of Retired Instructions
- CPI – Cycles Per Instruction
  - Num Cycles / Num retired instr
  - Low good, High bad
  - Theoretical best 0.25***
  - Anything below 1.00 pretty good.

****NOTE: Xeon Phi best CPI is 0.5*

Using CPI can be misleading

- Some optimisation steps can lead to an increase in CPI
- Always keep an eye on the fundamentals!
- How long did my program take to run?

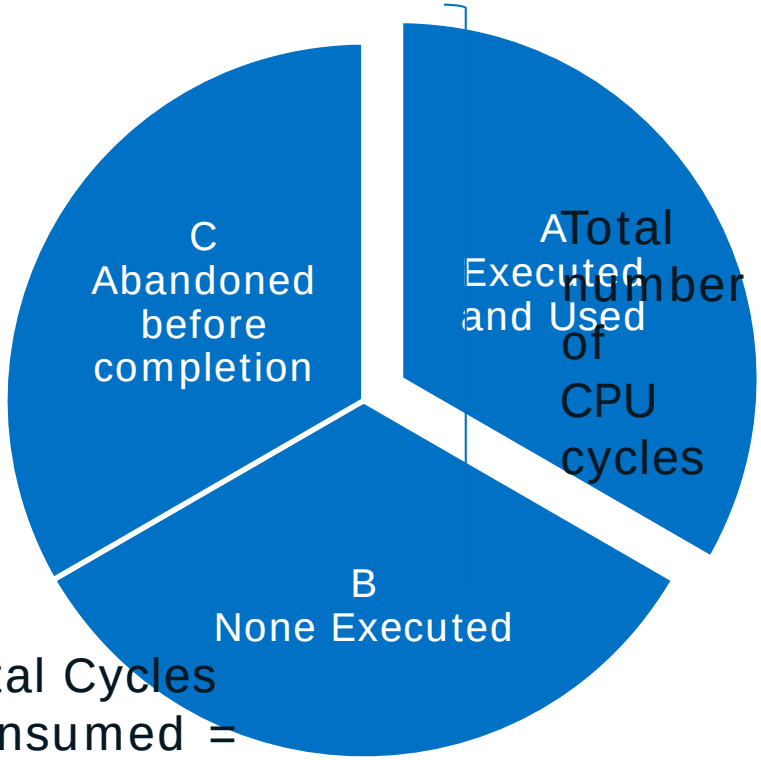
All programs consume cycles

These cycles consist of

Cycles where instructions are usefully executed

Cycles when nothing happens

Cycles where instructions are executed, but the results never used



Goal of performance tuning is to reduce each of these

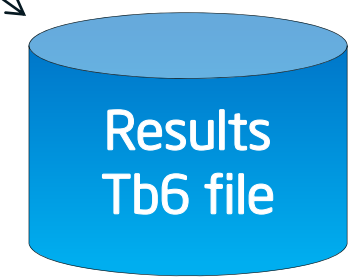
Total Cycles Consumed = A + B + C

Using Amplifier XE

① Profile

Collector

- Launch from Command line \ GUI *(Does not need GUI installed to run)*



② View the Results

⓪ You can add API to your source code *(optional)*

```
#include <ittnotify.h>
int main()
{
  __itt_domain* pD = __itt_domain_create( "Time" );
  pD->flags = 1; // enable domain

  for(int i=0;i< 100000;i++)
  {
    // mark the begining of the frame
    __itt_frame_begin_v3( pD,NULL);

    // simulate frames with different timings
    if(i%3)
      for(int j =0; j < 30000; j++);  // a delay
    else
      for(int j =0; j < 11200; j++);  // another delay

    // mark the end of the frame
    __itt_frame_end_v3( pD,NULL);
  }
  return 0;
} ,
```


Some tips to get you up
and running on VTune

Linux – Vtune is not recognised

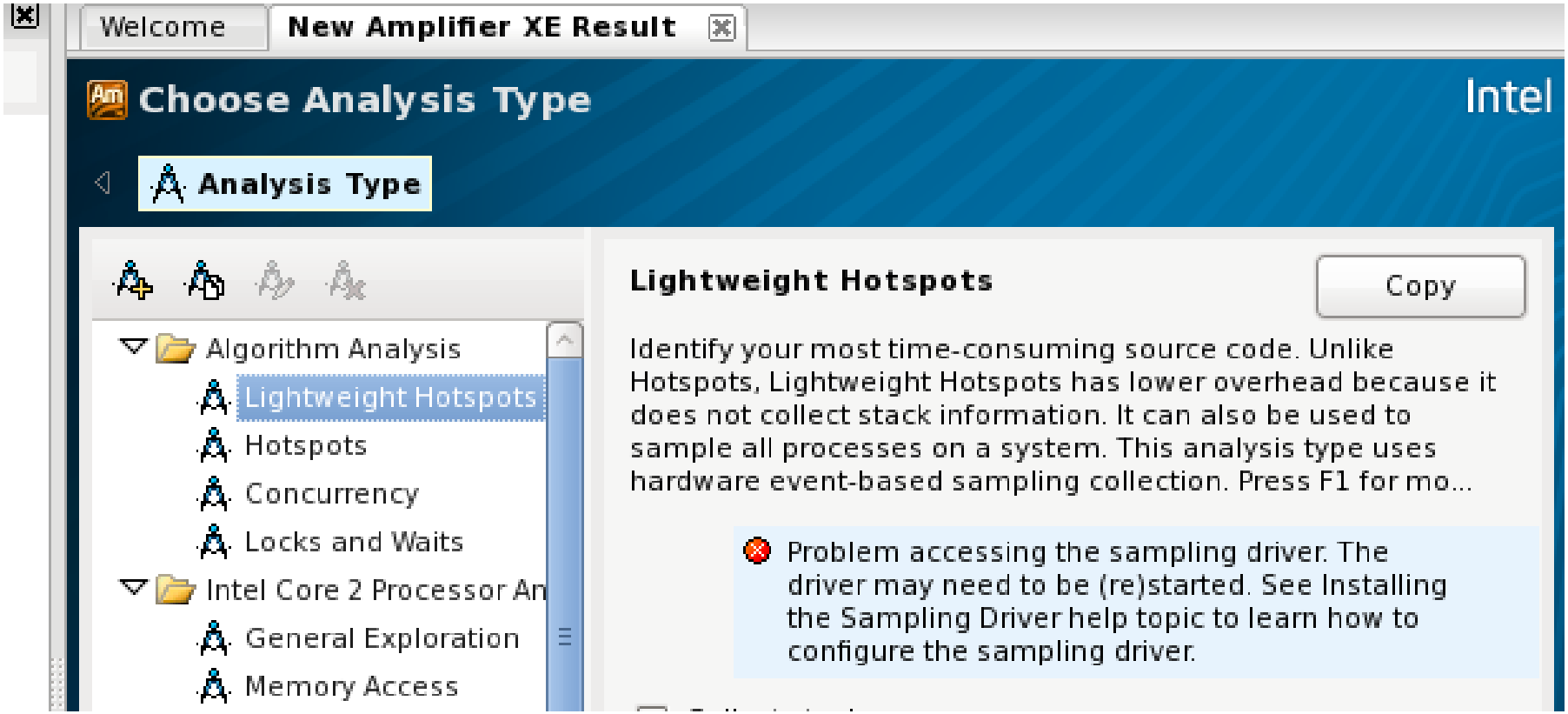
- You must ***source*** the path!

source /opt/intel/vtune_amplifier_xe/amplxe-var.sh

- To start from prompt.

Amplxe-gui &

Linux – problem accessing the sample driver!



Linux – problem accessing the sample driver!

1. Check if driver is loaded

```
cd /opt/intel/vtune_amplifier_xe/sepdk/prebuilt/
./insmod-sep3 –q
```

NOTE: The default installation expects users to be in the group 'vtune'

2. If not reload it

```
./insmod-sep3 -r -g democenter
```

<group>

3. if watchdog is causing a problem, disable it

"Warning: NMI watchdog timer is enabled. Turn off the nmi_watchdog tim
before running sampling."

```
echo 0 >  /proc/sys/kernel/nmi_watchdog
```

4. If not available, rebuild (see next slide)

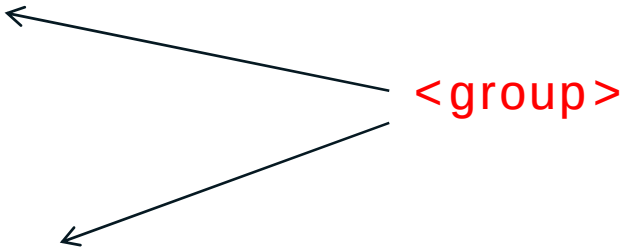
Linux – problem accessing the sample driver!

4. If not available, rebuild

```
cd /opt/intel/vtune_amplifier_xe/sepdk/src
./build-driver -ni --install-dir=../prebuilt
```

5. Load the driver

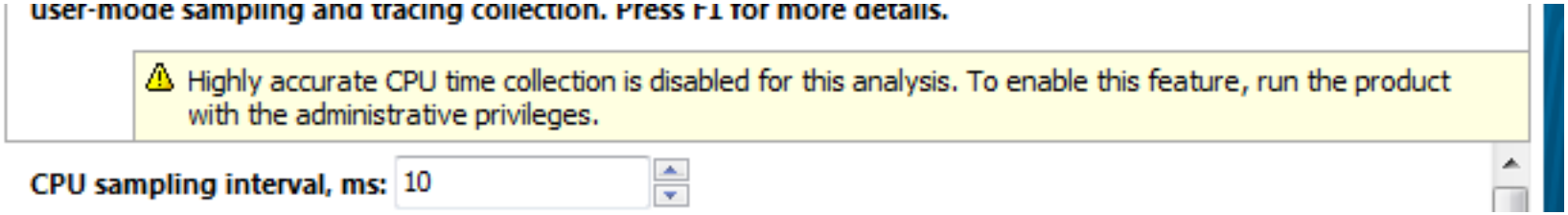
```
cd ../prebuilt
./insmod-sep3 -r -g democenter
```



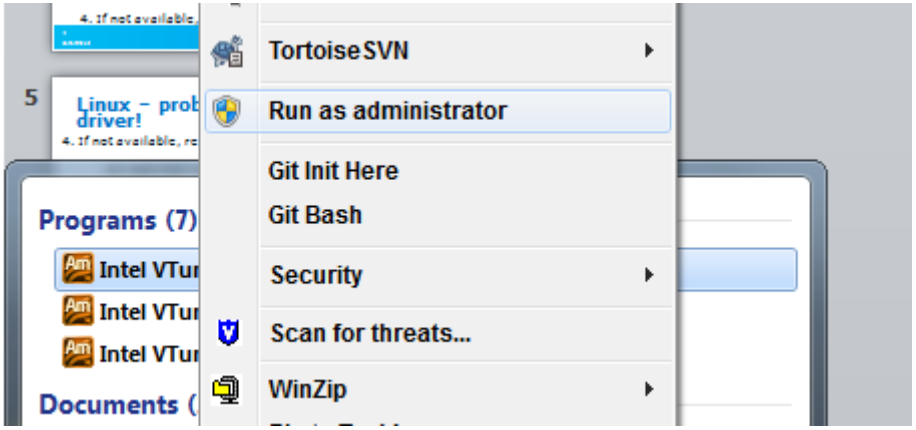
6. To load automatically on reboot

```
./boot-script --install -g democenter
```

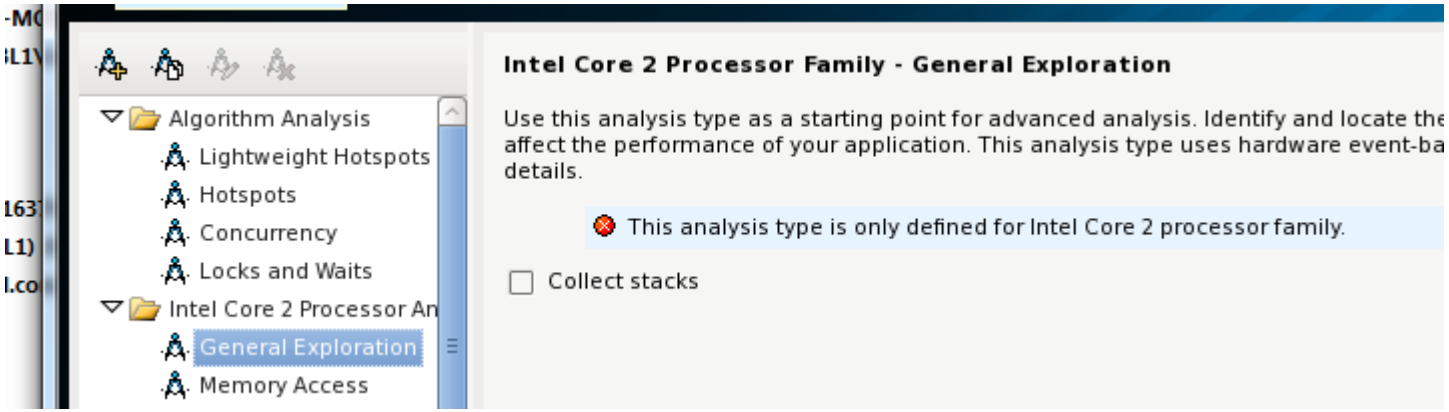
Windows – no accurate CPU time collection.



Start the program in Administrator mode:



Architectural Analysis not available



You are highlighting the wrong analysis type!

Linux – The source editor won't open

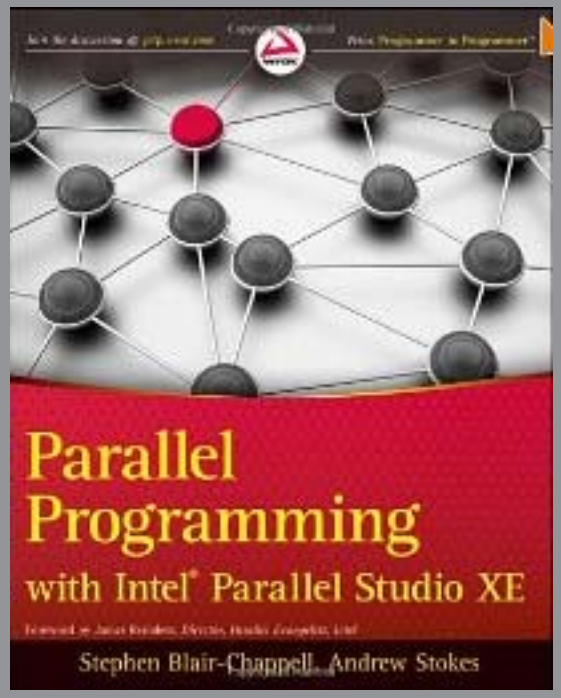
Set the EDITOR or VISUAL environment variable!

```
export EDITOR=gedit
```

or

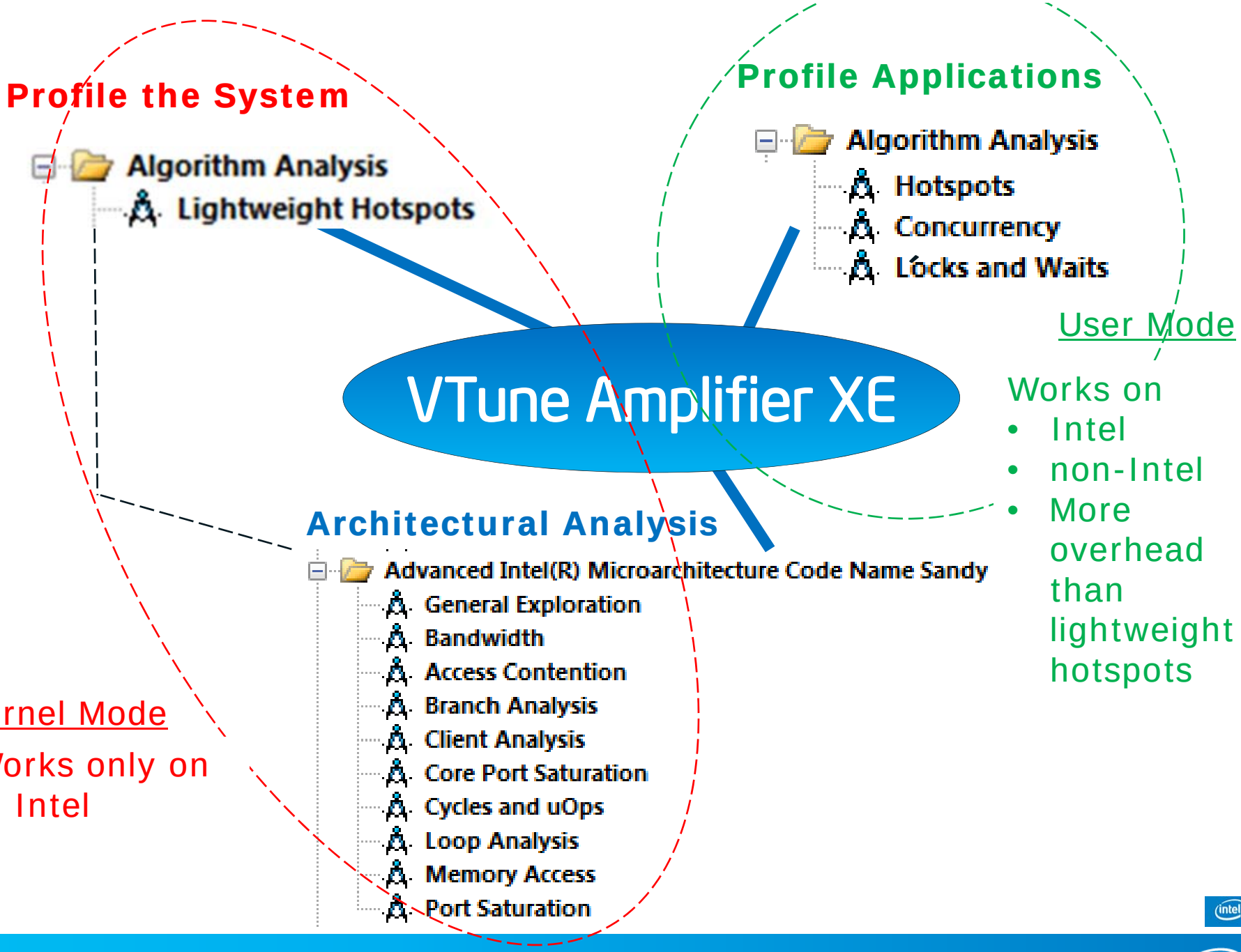
```
export EDITOR=vi
```

Hands-on Lab

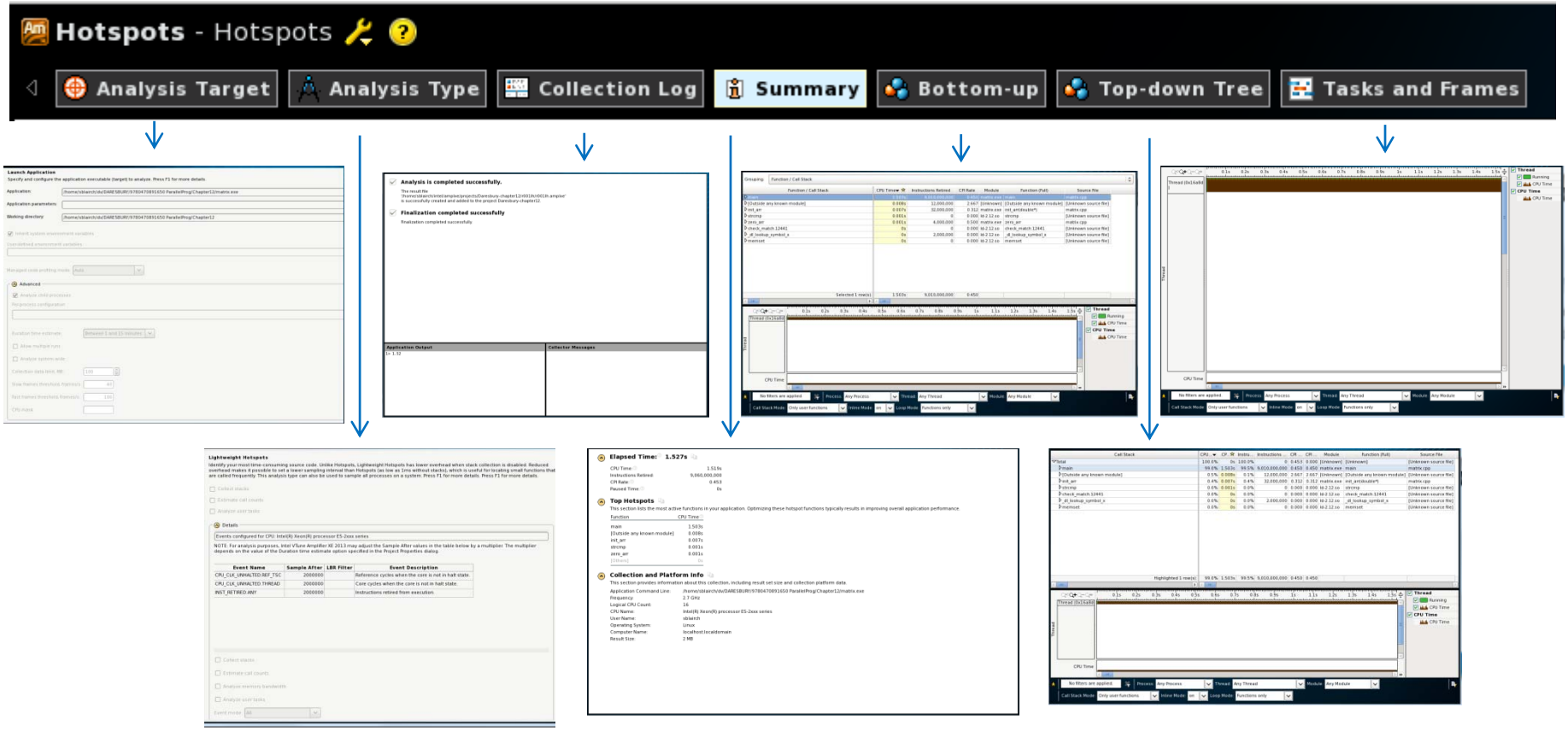


Activity 12-1 (Page 345)

Conducting a System-wide Analysis



Hotspots – The Results Tab



The Summary Page

Elapsed Time: 7.814s

CPU Time:	7.641s
Instructions Retired:	11,076,000,000
CPI Rate:	2.312

The CPI may be too high. This could be caused by issues such as memory...

Paused Time:	0s

Top Hotspots

This section lists the most active functions in your application

Function	CPU Time
main	7.549s
[vmlinux]	0.089s
init_arr	0.003s
_dl_relocate_object	0s

(a) Lightweight

Elapsed Time: 7.737s

Total Thread Count:	1
CPU Time:	7.736s
Paused Time:	0s

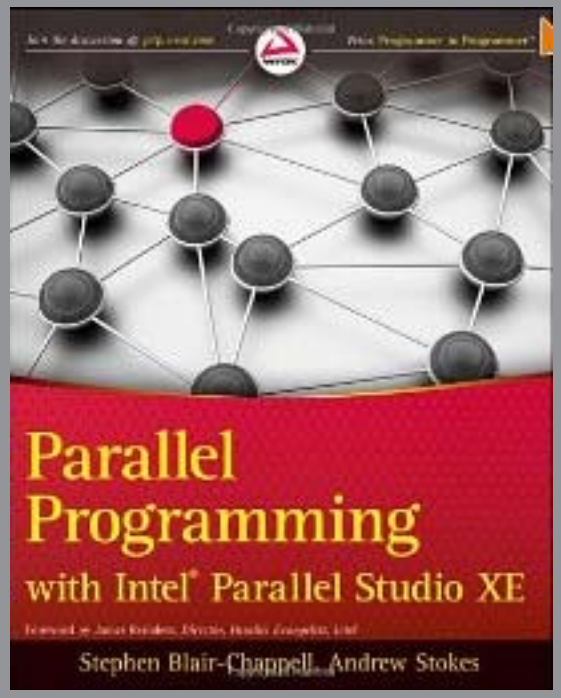
Top Hotspots

This section lists the most active functions in your application

Function	CPU Time
main	7.705s
init_arr	0.031s

(b) User Mode

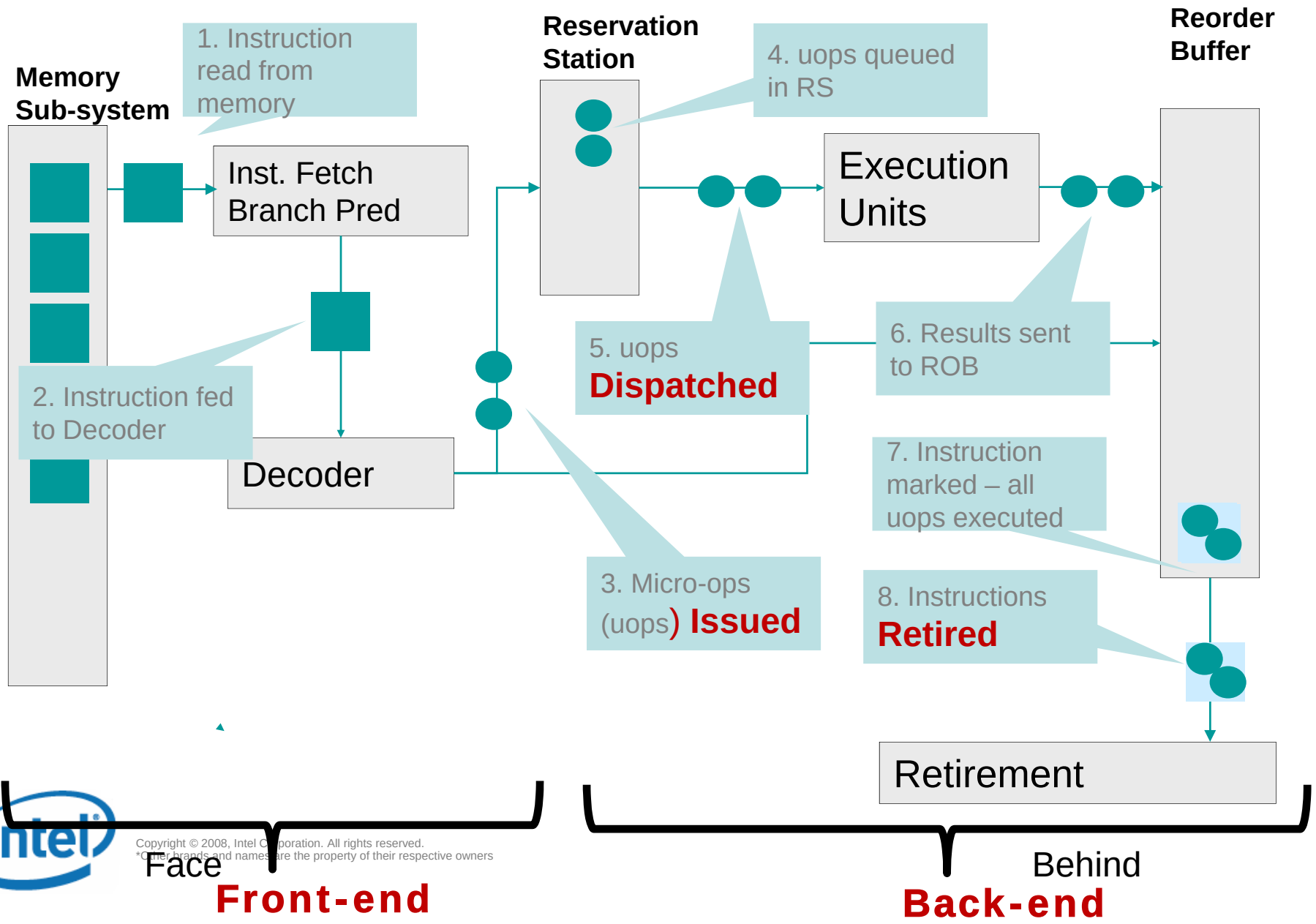
Hands-on Lab



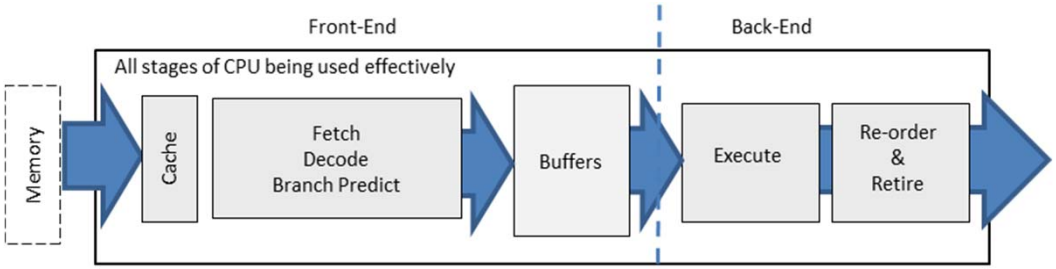
Activity 12-2 (Page 352)

Conducting a Lightweight Hotspot Analysis

The life of a program instruction



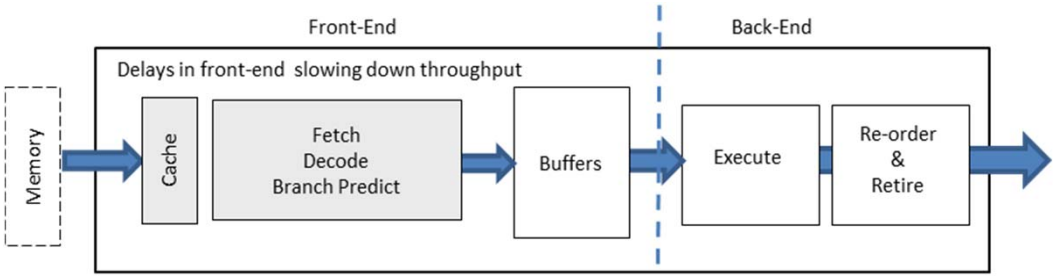
Retirement Dominated



- All stages of CPU used effectively
- Will have a low CPI (less than 0.4)
- Percentage CPU utilisation approaching 100%

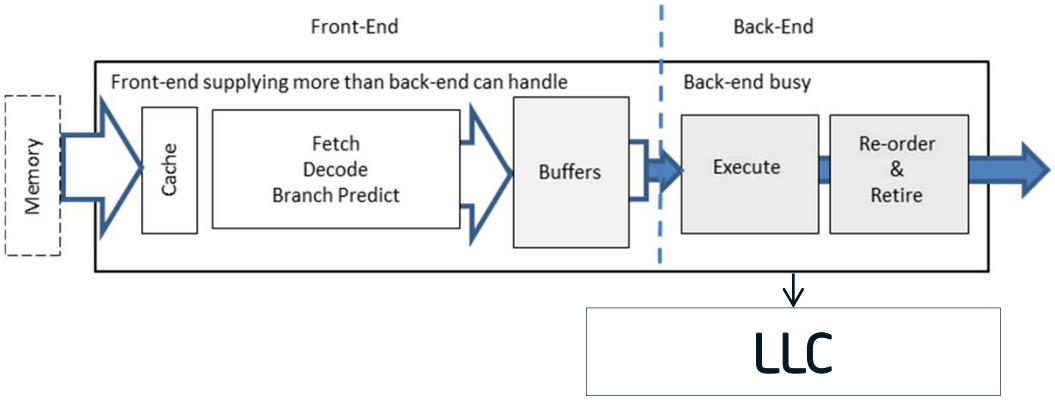
Biggest opportunity: reduce amount of code being executed

Front-end Bound



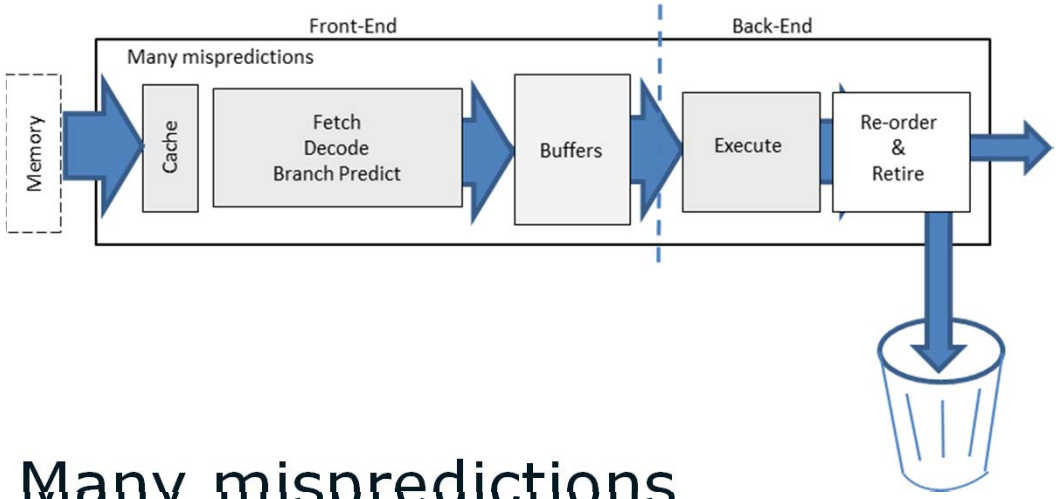
- Delays in Front-end slowing throughput
- Lots of cycles where no execution occurs
- High CPI
- E.g.
  - Time taken to decode instructions

Back-end Bound



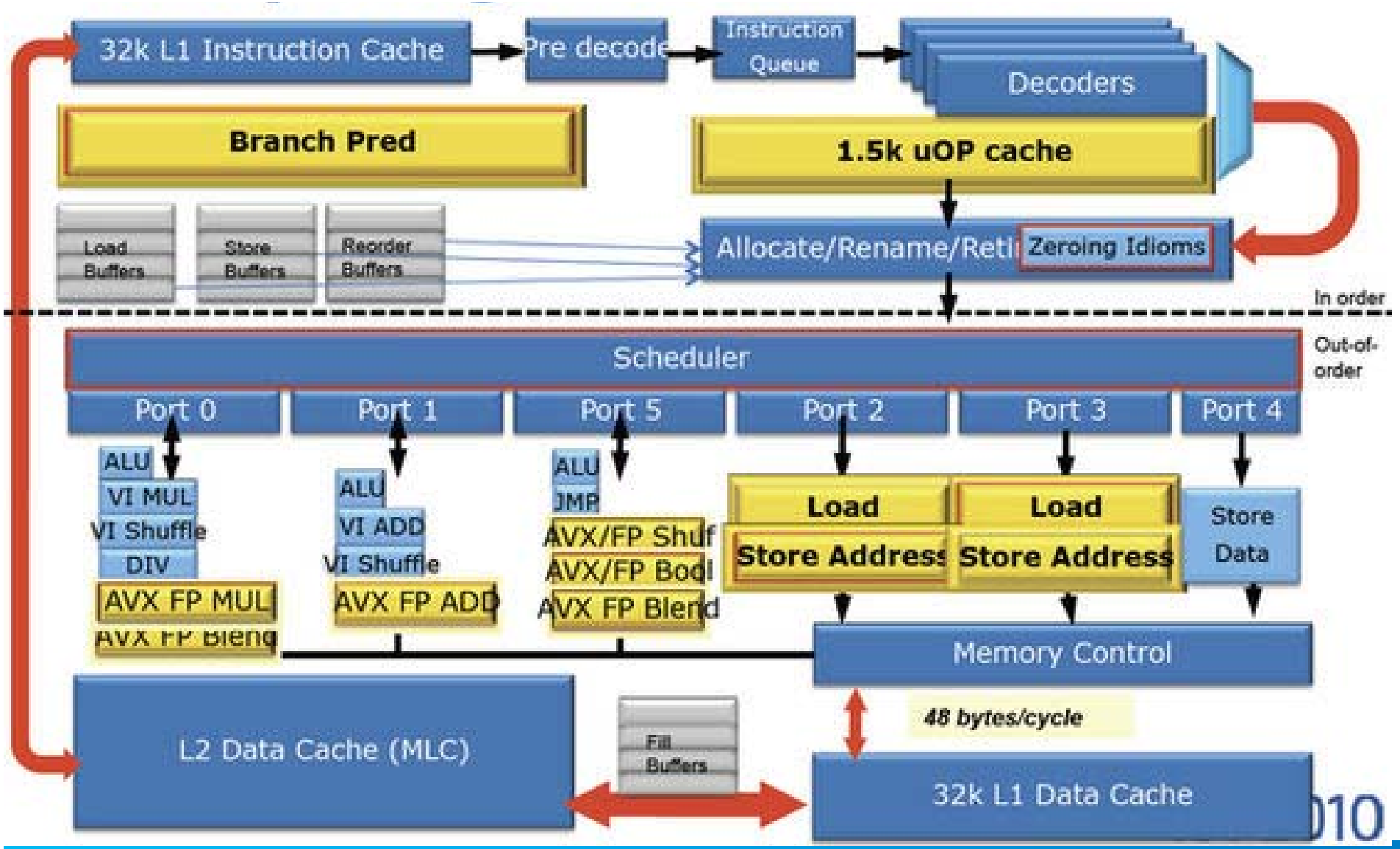
- Front-end supplying more than back-end can handle
- High CPI
- Usually caused by data-dependencies
- Back-end internal queues become full
- Cache misses

Cancellation Dominated

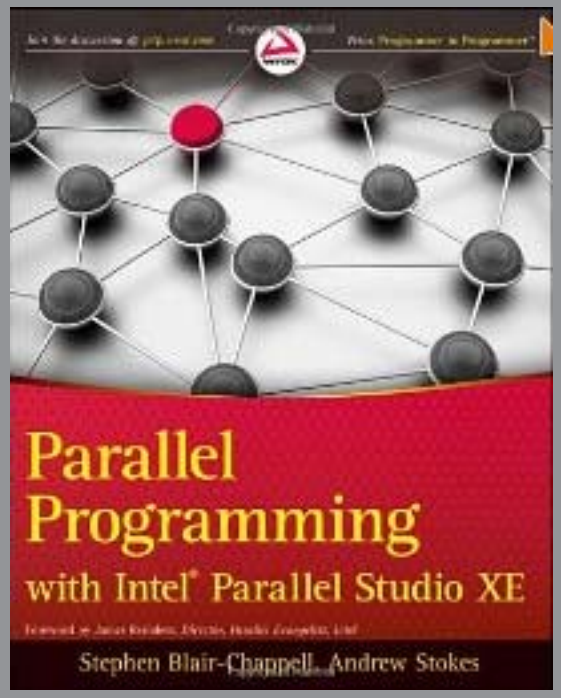


- Many mispredictions
- Micro-ops ‘thrown away’ before retirement
- High CPI
- e.g.
  - database applications
  - Pointer chasing

Sandy Bridge CPU



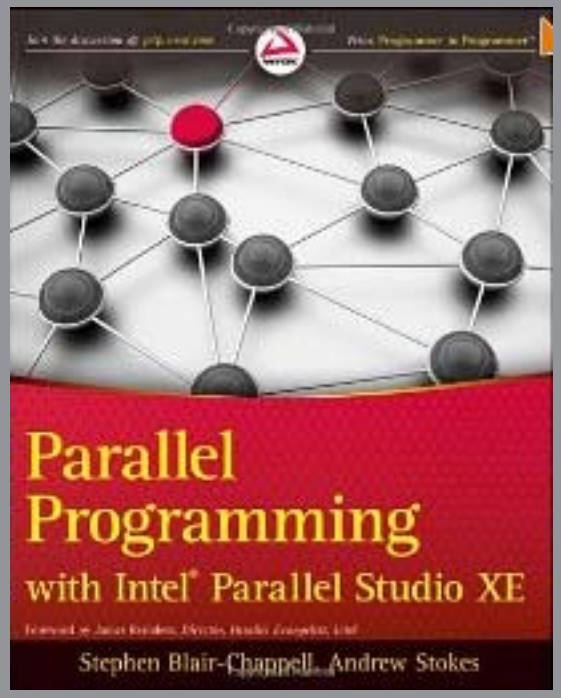
Hands-on Lab



Activity 12-3 (Page 354)

Conducting a general exploration analysis

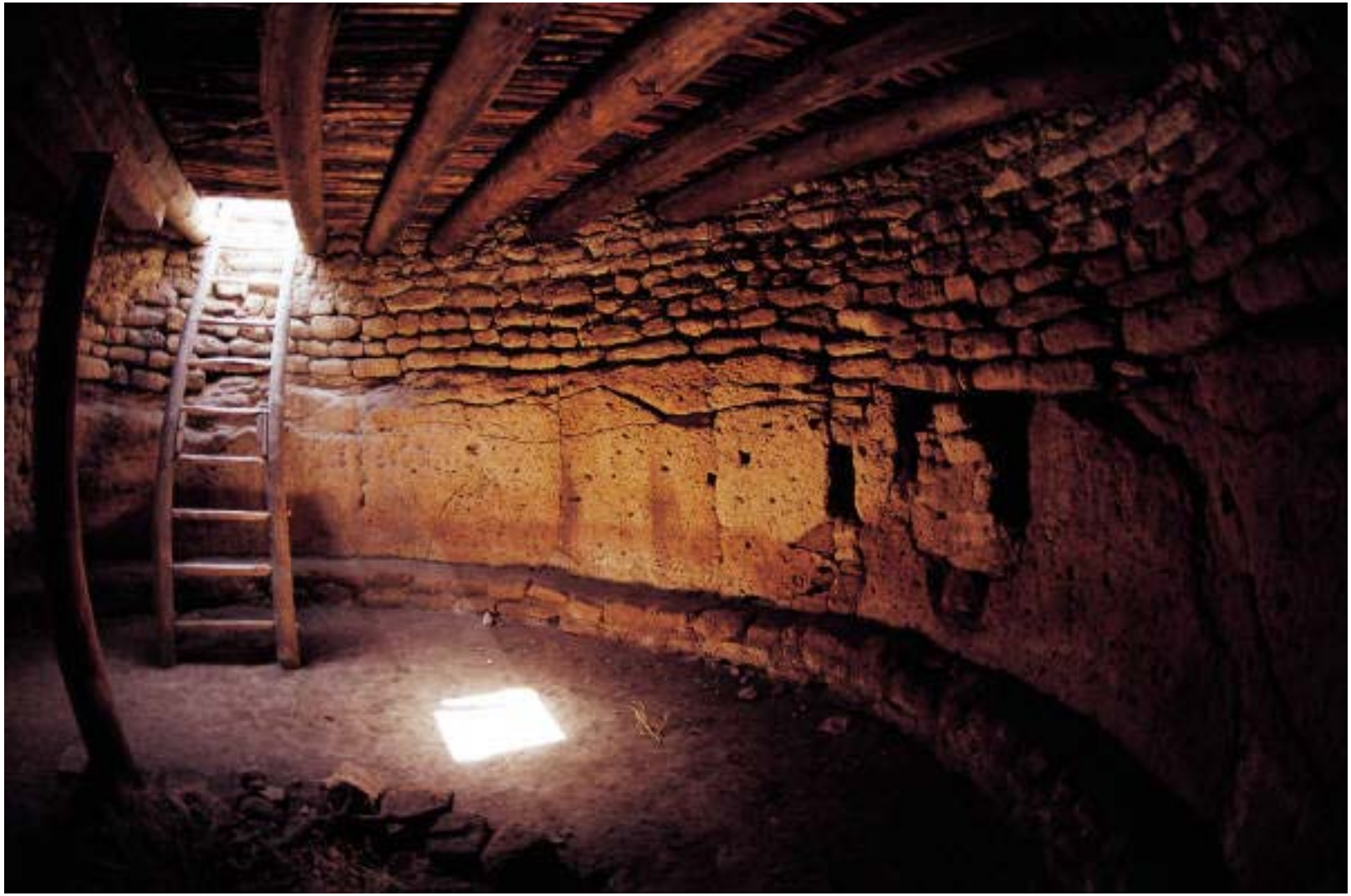
Hands-on Lab



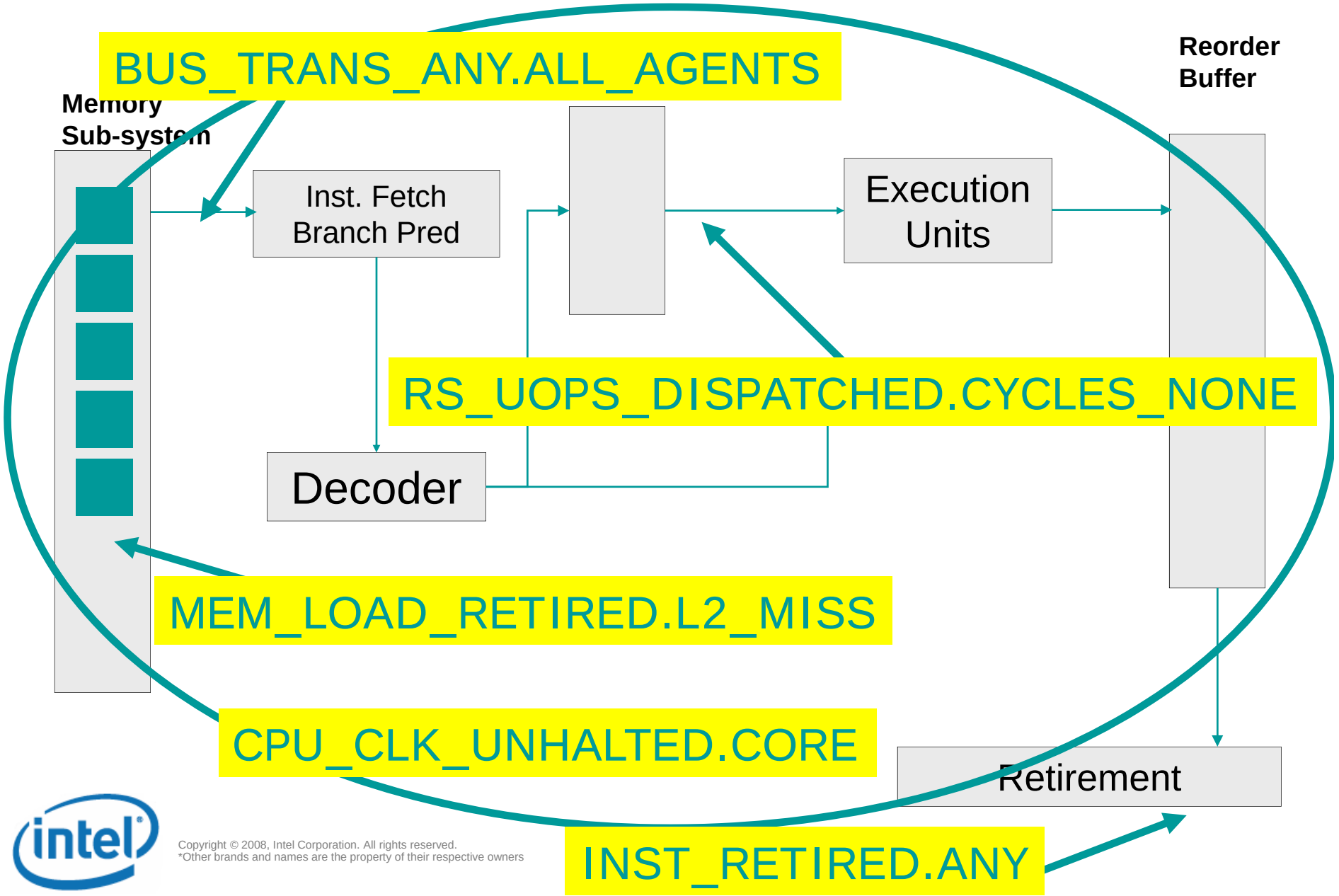
Activity 12-4 (Page 362)

Optimizing the Application

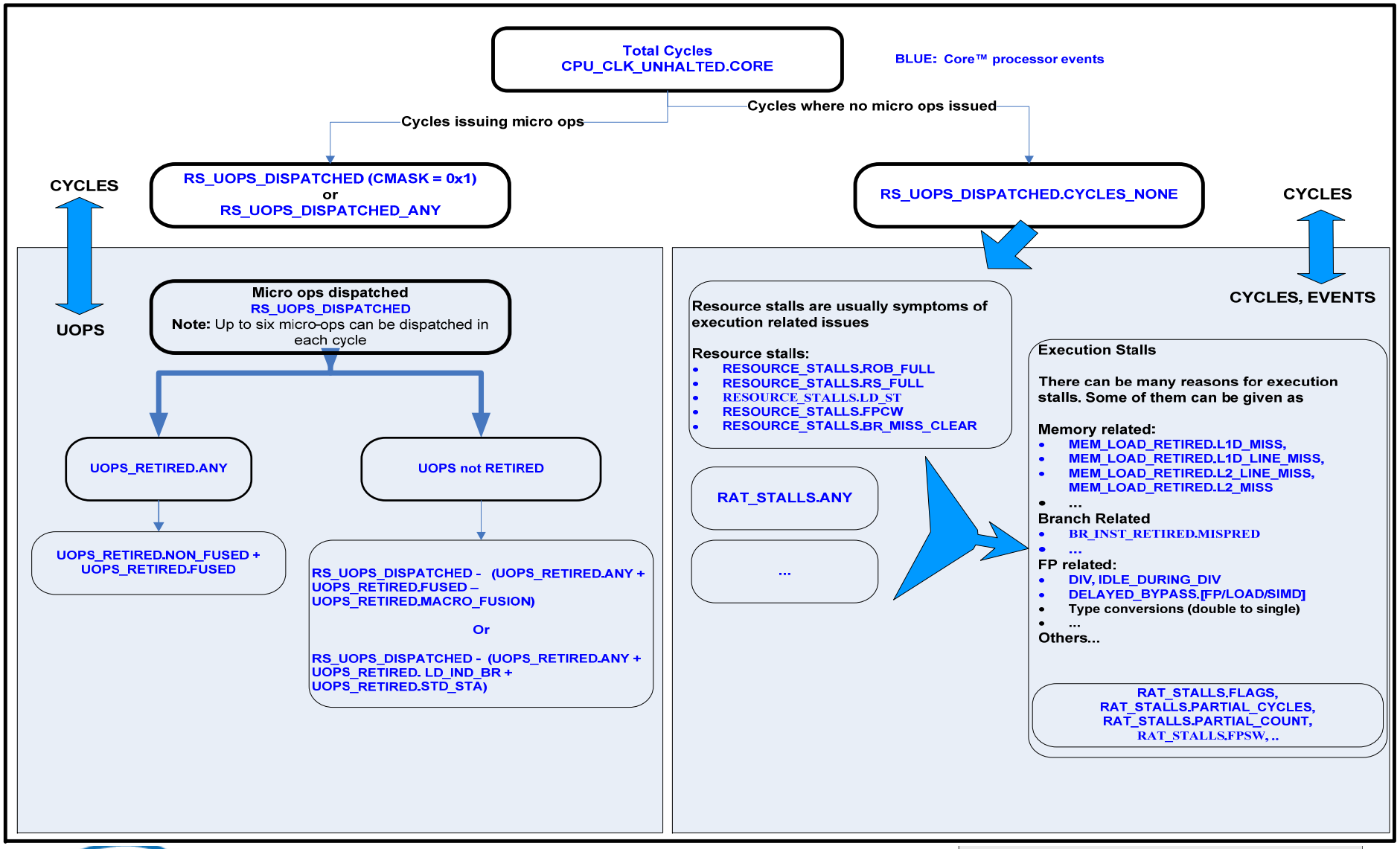
Down in the Cellar



Hardware Performance Events



Cycle Analysis summary



Suggested Order of Fixing Problems

Priority	Problem
1	Cache misses
2	Contested access
3	Other data access issues
4	Allocation Stalls
5	Micro Assists
6	Branch Mispredictions and machine clears
7	Other Front-end stalls

See slides in backup section for a more detailed description

Cache Misses

- **Why:** Cache misses raise the CPI of an application
  - Focus on long-latency data accesses coming from 2nd and 3rd level misses
- **How:** General Exploration profile, Metrics: *LLC Hit*, *LLC Miss*
- **What Now:**
  - If either metric is highlighted for your hotspot, consider reducing misses:
    - Use the cacheline replacement analysis outlined in the Intel® 64 and IA-32 Architectures Optimization Reference Manual, section **B.3.4.2**
    - Use software prefetch instructions
    - Block data accesses to fit into cache
    - Use local variables for threads
    - Pad data structures to cacheline boundaries
    - Change your algorithm to reduce data storage

B.3.4.2 Cache-line Replacement Analysis

When an application has many cache misses, it is a good idea to determine where cache lines are being replaced at the highest frequency. The instructions responsible for high amount of cache replacements are not always where the application is spending the majority of its time, since replacements can be driven by the hardware prefetchers and store operations which in the common case do not hold up the pipeline. Typically traversing large arrays or data structures can cause heavy cache line replacements.

Required events

L1D.REPLACEMENT – Replacements in the 1st level data cache.

L2_LINES_IN.ALL – Cache lines being brought into the L2 cache.

OFFCORE_RESPONSE.DATA_IN_SOCKET.LLC_MISS_LOCAL.DRAM_0 – Cache lines being brought into the LLC.

Usages of events

Identifying the replacements that potentially cause performance loss can be done at process, module, and function level. Do it in two steps:

- Use the precise load breakdown to identify the memory hierarchy level at which loads are satisfied and cause the highest penalty.
- Identify, using the formulas below, which portion of code causes the majority of the replacements in the level below the one that satisfies these high penalty loads.

For example, if there is high penalty due to loads hitting the LLC, check the code which is causing replacements in the L2 and the L1. In the formulas below, the nominators are the replacements accounted for a module or function. The sum of the replacements in the denominators is the sum of all replacements in a cache level for all processes. This enables you to identify the portion of code that causes the majority of the replacements.

L1D Cache Replacements

%L1D.REPLACEMENT =
L1D.REPLACEMENT / SumOverAllProcesses(L1D.REPLACEMENT);

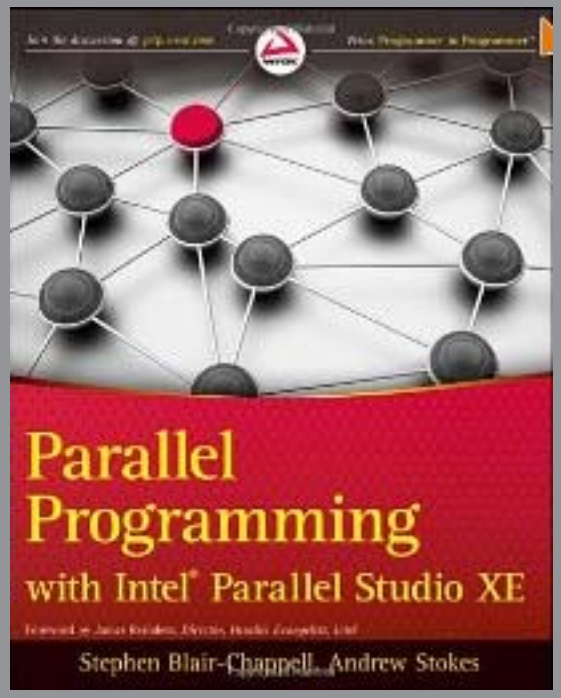
L2 Cache Replacements

%L2.REPLACEMENT =
L2_LINES_IN.ALL / SumOverAllProcesses(L2_LINES_IN.ALL);

L3 Cache Replacements

%L3.REPLACEMENT =
OFFCORE_RESPONSE.DATA_IN_SOCKET.LLC_MISS_LOCAL.DRAM_0/ SumOverAllProcesses(OFFCORE_RESPONSE.DATA_IN_SOCKET.LLC_MISS_LOCAL.DRAM_0);

Hands-on Lab



Activity 12-1 (Page 345)

Conducting a System-wide Analysis

Contested Accesses

- **Why:** Sharing modified data among cores can raise the latency of data access
- **How:** General Exploration profile, Metrics: *Contested Accesses*
- **What Now:**
  - If either metric is highlighted for your hotspot, locate the source code line(s) that is generating HITMs by viewing the source. Look for the MEM_LOAD_UOPS_LLC_HIT_RETIRED.XSNP_HITM_PS event which will tag to the next instruction after the one that generated the HITM.
  - Then use knowledge of the code to determine if real or false sharing is taking place. Make appropriate fixes:
    - For real sharing, reduce sharing requirements
    - For false sharing, pad variables to cacheline boundaries

Hit Modified Data

a cache line in a cache of another core and the cache line has not been invalidated.

MEM_LOAD_UOPS_LLC_HIT_RETIRED.XSNP_HITM_PS - Counts demand loads that hit a cache line in the cache of another core and the cache line has been written to by that other core. This event is important for many performance bottlenecks that can occur in multi-threaded applications, such as lock contention and false sharing.

B-46

Other Data Access Issues: Blocked Loads Due to No Store Forwarding

- **Why:** If it is not possible to forward the result of a store through the pipeline, dependent loads may be blocked
- **How:** General Exploration profile, Metric: *Loads Blocked by Store Forwarding*
- **What Now:**
  - If the metric is highlighted for your hotspot, investigate:
  - View source and look at the LD_BLOCKS_STORE_FORWARD event. Usually this event tags to next instruction after the attempted load that was blocked. Locate the load, then try to find the store that cannot forward, which is usually within the prior 10-15 instructions. The most common case is that the store is to a smaller memory space than the load. Fix the store by storing to the same size or larger space as the ensuing load.

2.2.4.4 Store Forwarding

If a load follows a store and reloads the data that the store writes to memory, the Intel Core microarchitecture can forward the data directly from the store to the load. This process, called store to load forwarding, saves cycles by enabling the load to obtain the data directly from the store operation instead of through memory.

The following rules must be met for store to load forwarding to occur:

- The store must be the last store to that address prior to the load.
- The store must be equal or greater in size than the size of data being loaded.
- The load cannot cross a cache line boundary.
- The load cannot cross an 8-Byte boundary. 16-Byte loads are an exception to this rule.
- The load must be aligned to the start of the store address, except for the following exceptions:
  - An aligned 64-bit store may forward either of its 32-bit halves
  - An aligned 128-bit store may forward any of its 32-bit quarters
  - An aligned 128-bit store may forward either of its 64-bit halves

Software can use the exceptions to the last rule to move complex structures without losing the ability to forward the subfields.

Other Data Access Issues: Cache Line Splits

- **Why:** Multiple cache line splits can result in load penalties.
- **How:** General Exploration profile, Metric: *Split Loads, Split Stores*
- **What Now:**
  - If the metric is highlighted for your hotspot, investigate by viewing the metrics at the source code level. The split load event, MEM_UOP_RETIRED.SPLIT_LOADS_PS, should tag to the next executed instruction after the one causing the split. If the split store ratio is greater than .01 at any source address, it is worth investigating.
  - To fix these issues, ensure your data is aligned. Especially watch out for mis-aligned 256-bit AVX store operations.

Other Data Access Issues: 4K Aliasing

- **Why:** Aliasing conflicts result in having to re-issue loads.
- **How:** General Exploration profile, Metric: *4K Aliasing*
- **What Now:**
  - If this metric is highlighted for your hotspot, investigate at the sourcecode level.
  - Fix these issues by changing the alignment of the load. Try aligning data to 32 bytes, changing offsets between input and output buffers (if possible), or using 16-Byte memory accesses on memory that is not 32-Byte aligned.

11.8 4K ALIASING

4-KByte memory aliasing occurs when the code stores to one memory location and shortly after that it loads from a different memory location with a 4-KByte offset between them. For example, a load to linear address 0x400020 follows a store to linear address 0x401020.

The load and store have the same value for bits 5 - 11 of their addresses and the accessed byte offsets should have partial or complete overlap.

4K aliasing may have a five-cycle penalty on the load latency. This penalty may be significant when 4K aliasing happens repeatedly and the loads are on the critical path. If the load spans two cache lines it might be delayed until the conflicting store is committed to the cache. Therefore 4K aliasing that happens on repeated unaligned Intel AVX loads incurs a higher performance penalty.

To detect 4K aliasing, use the LD_BLOCKS_PARTIAL.ADDRESS_ALIAS event that counts the number of times Intel AVX loads were blocked due to 4K aliasing.

To resolve 4K aliasing, try the following methods in the following order:

- Align data to 32 Bytes.
- Change offsets between input and output buffers if possible.
- Use 16-Byte memory accesses on memory which is not 32-Byte aligned.

Other Data Access Issues: DTLB Misses

Why: First-level DTLB Load misses (Hits in the STLB) incur a latency penalty. Second-level misses require a page walk that can affect your application's performance.

How: General Exploration profile, Metric: *DTLB Overhead*

What Now:

- If this metric is highlighted for your hotspot, investigate at the sourcecode level.
- To fix these issues, target data locality to TLB size, use the Extended Page Tables (EPT) on virtualized systems, try large pages (database/server apps only), increase data locality by using better memory allocation or Profile-Guided Optimization

Allocation Stalls

Why: Certain types of instructions can cause allocation stalls because they take longer to retire. These increase latencies overall.

How: General Exploration Profile, Metric: *LEA Stalls, Flags Merge Stalls*

What Now:

- If this metric is highlighted for your hotspot, investigate at the sourcecode level.
- Try to eliminate uses of 3-operand LEA instructions, Look for certain uses of an LEA instruction (see section 3.5.1.3 of Intel® 64 and IA-32 Architectures Optimization Reference Manual) or partial register use (see section 3.5.2.4 of Intel® 64 and IA-32 Architectures Optimization Reference Manual) and fix.

Microcode Assists

Retired

Why: Assists from the microcode sequencer can have long latency penalties.

How: General Exploration Profile, Metric: *Assists*

What Now:

- If this metric is highlighted for your hotspot, re-sample using the additional assist events to determine the cause.
- If FP_ASSISTS.ANY / INST_RETIRED.ANY is significant, check for denormals. To fix enable FTZ and/or DAZ if using SSE/AVX instructions or scale your results up or down depending on the problem
- If ((OTHER_ASSISTS.AVX_TO_SSE_NP*75) / CPU_CLK_UNHALTED.THREAD) or ((OTHER_ASSISTS.SSE_TO_AVX_NP*75) / CPU_CLK_UNHALTED.THREAD) is greater than .1, reduce transitions between SSE and AVX code

Cancelled

Branch Mispredicts

Why: Mispredicted branches cause pipeline inefficiencies due to wasted work or instruction starvation (while waiting for new instructions to be fetched)

How: General Exploration Profile, Metric: *Branch Mispredict*

What Now:

- If this metric is highlighted for your hotspot try to reduce misprediction impact:
- Use compiler options or profile-guided optimization (PGO) to improve code generation
- Apply hand-tuning by doing things like hoisting the most popular targets in branch statements.

Machine Clears

Cancelled

Why: Machine clears cause the pipeline to be flushed and the store buffers emptied, resulting in a significant latency penalty.

How: General Exploration Profile, Metric: *Machine Clears*

Now What:

- If this metric is highlighted for your hotspot try to determine the cause using the specific events:
- If MACHINE_CLEARS.MEMORY_ORDERING is significant, investigate at the sourcecode level. This could be caused by 4K aliasing conflicts or contention on a lock (both previous issues).
- If MACHINE_CLEARS.SMC is significant, the clears are being caused by self-modifying code, which should be avoided.

Front-end Stalls

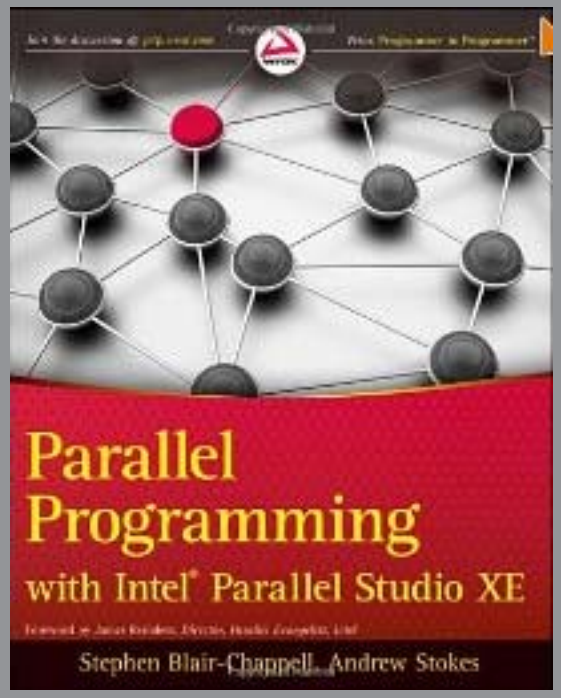
Why: Front-end stalls (at the Issue stage of the pipeline) can cause instruction starvation, which may lead to stalls at the execute stage in the pipeline.

How: General Exploration profile, Metric: *Front-end Bound Pipeline Slots*

What Now:

- If this metric is highlighted for your hotspot, try using better code layout and generation techniques:
  - Try using profile-guided optimizations (PGO) with your compiler
  - Use linker ordering techniques (/ORDER on Microsoft's linker or a linker script on gcc)
  - Use switches that reduce code size, such as /O1 or /Os

Hands-on Lab



Activity 12-4 (Page 362)

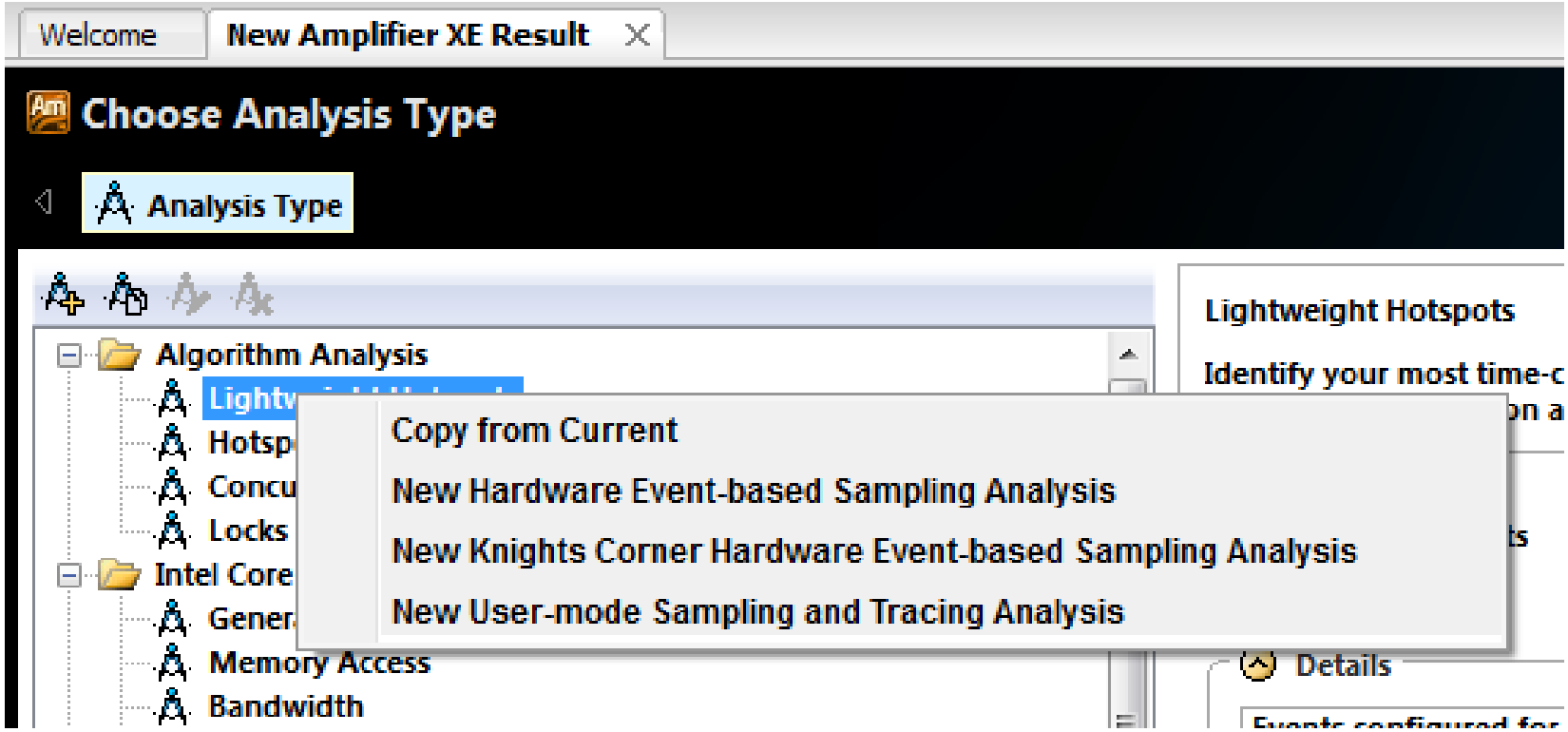
Optimizing the Application


User Defined Analysis Types

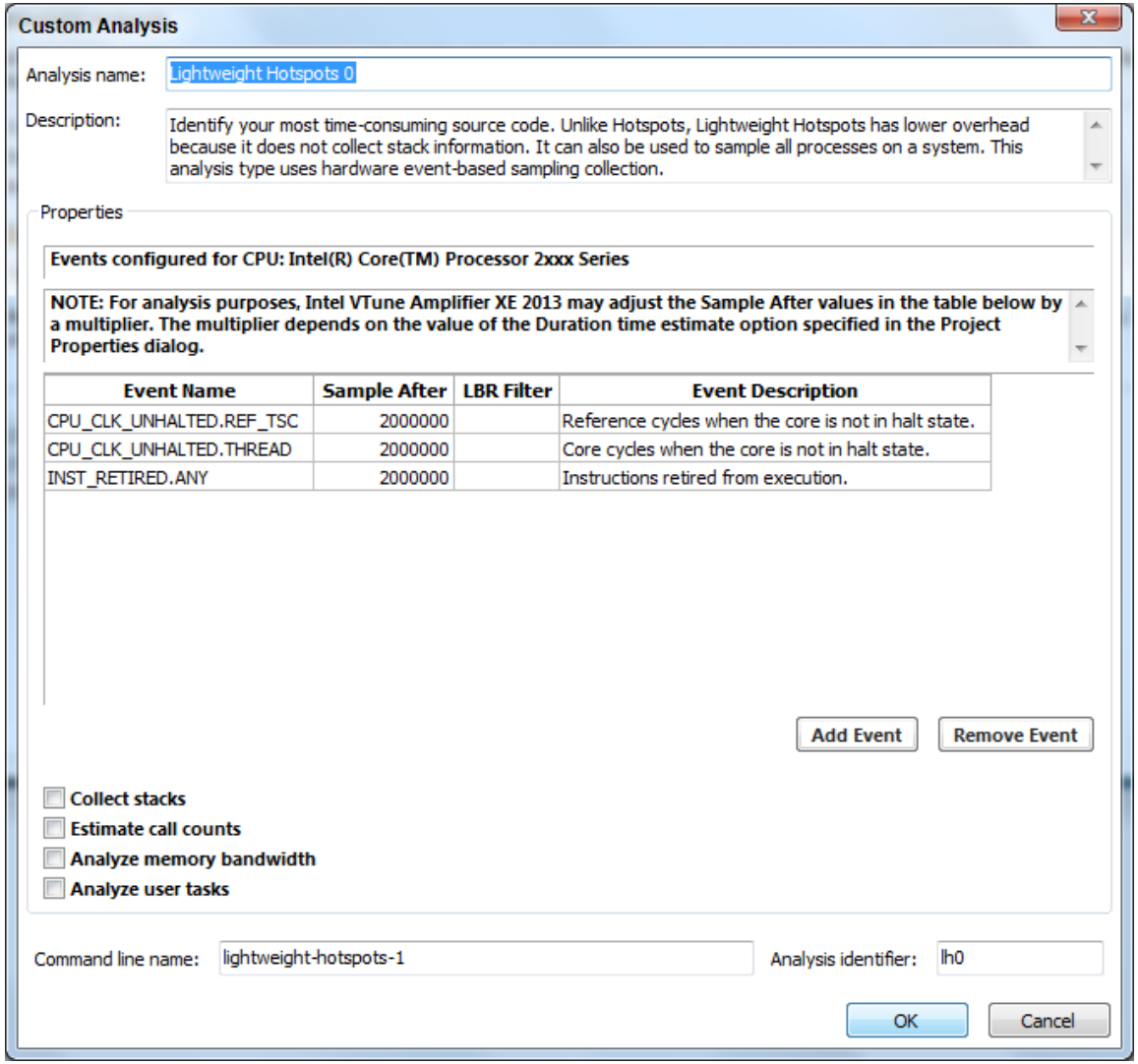
Two options

- Copy a new analysis type based on a current analysis type
  - Add and delete events
- Create a new analysis type from scratch (without help of GUI)
  - Add and delete events
  - Create new ratios
  - Create new views

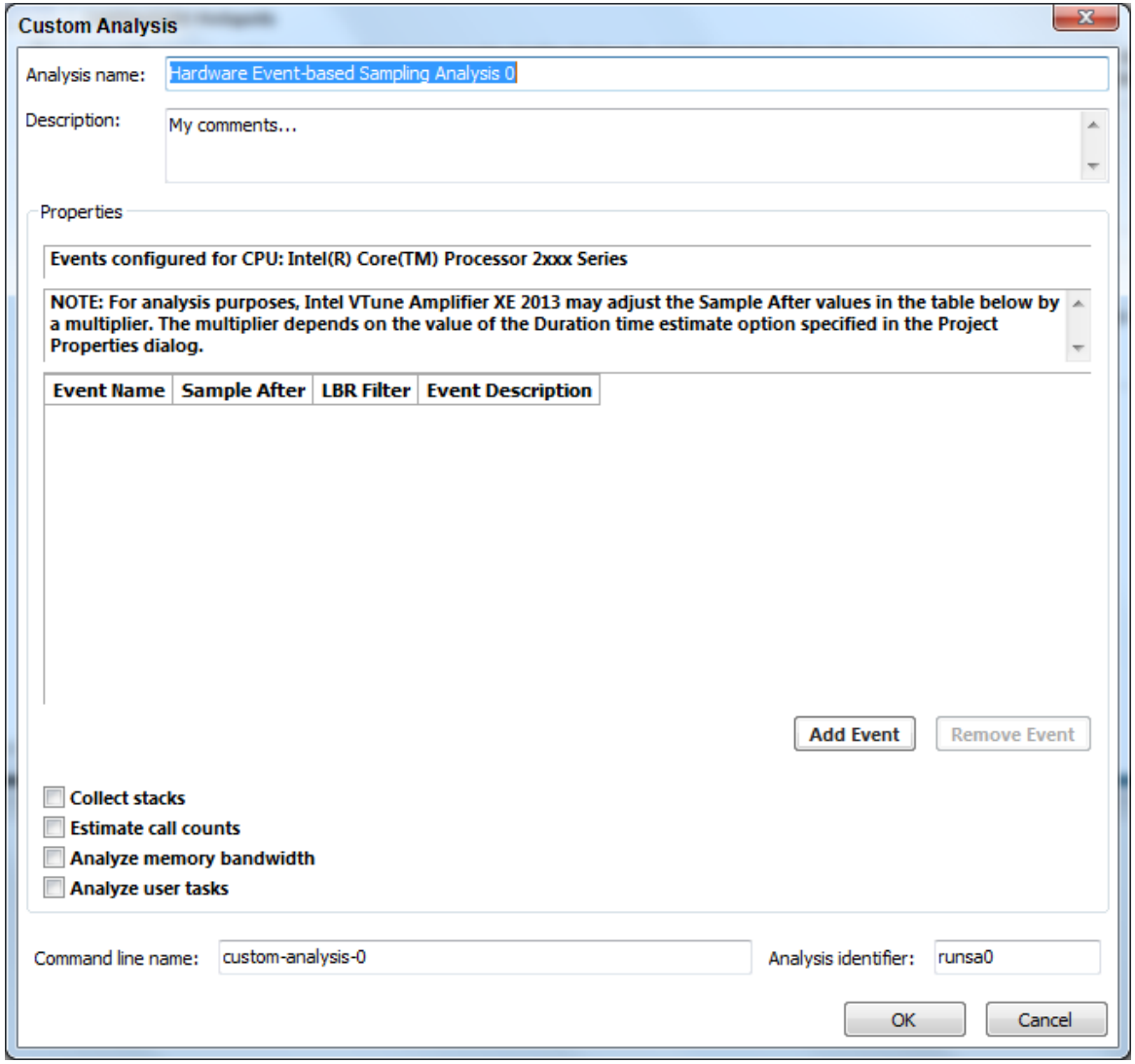
Creating New Analysis from GUI



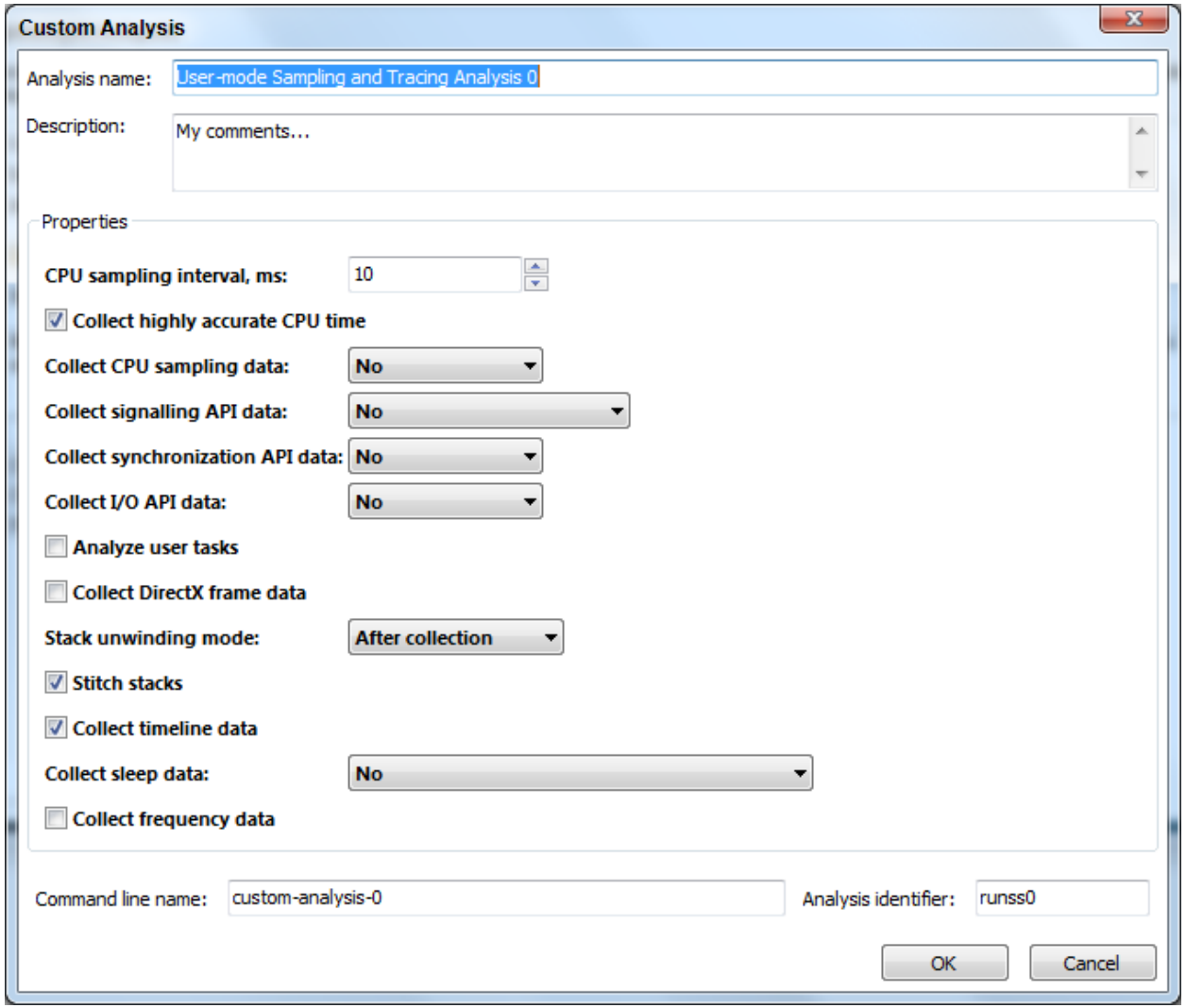
Copy from current



New Hardware Based Analysis



New User Mode analysis



You can create your own analysis types

- See the folder /<install_dir>/sdk/user_metrics/user_metrics.zip

- Generate your own viewpoint and analysis types
- Install your own types using *translate_metrics.pyc*

translate_metrics.pyc options

-i <path>: install to path

-m <file>: metrics file

--list-valid-display-types

--list-valid-architectures

--list-valid-predefined-metrics

Example

translate_metrics.pyc -m CPI_analysis.py -install-to-path /install/destination/

Generate an analysis type and a viewpoint for the analysis defined in the CPI_analysis.py file. The generated files will be created in the '/install/destination' directory.

The Metrics File

See *User Metrics Documentation.pdf*

- **PMU event metrics**
  Also referred to as *formulas* and *ratios*, are (mathematic) functions of one or more PMU events' counts.

- **Analysis**
  Describes which metrics should be viewed together, and how they should be arranged.

More information:

VTune User Forums:
http://software.intel.com/en-us/forums/intel-vtune-performance-analyzer/

VTune Amplifier XE Videos:
http://software.intel.com/en-us/articles/intel-vtune-amplifier-xe/

Intel® 64 and IA-32 Architecture Software Developer's Manuals:
http://www.intel.com/products/processor/manuals/index.htm

Optimization Guide for Intel® Microarchitecture Codename Nehalem:
http://software.intel.com/file/15529

For optimization of the integrated graphics controller on
Intel® Microarchitecture Codename Sandy Bridge:
www.intel.com/software/gpa

Thank You

Backup

What is Cache?

Whiteboard discussion

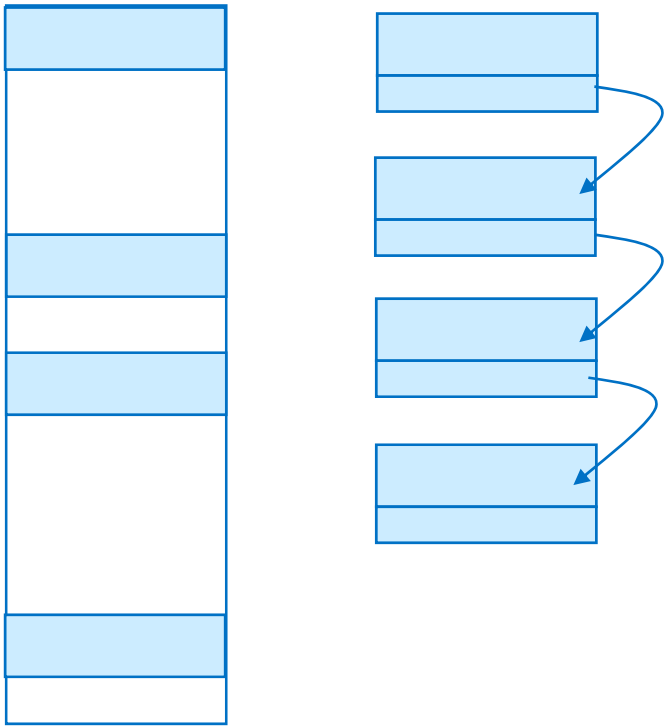
Discuss what cache is.

Whiteboard Discussion

Question: Linked List

A linked list dynamically allocates memory for each node.

- The addresses are at non-contiguous address.
- Iterating through the list causes cache misses

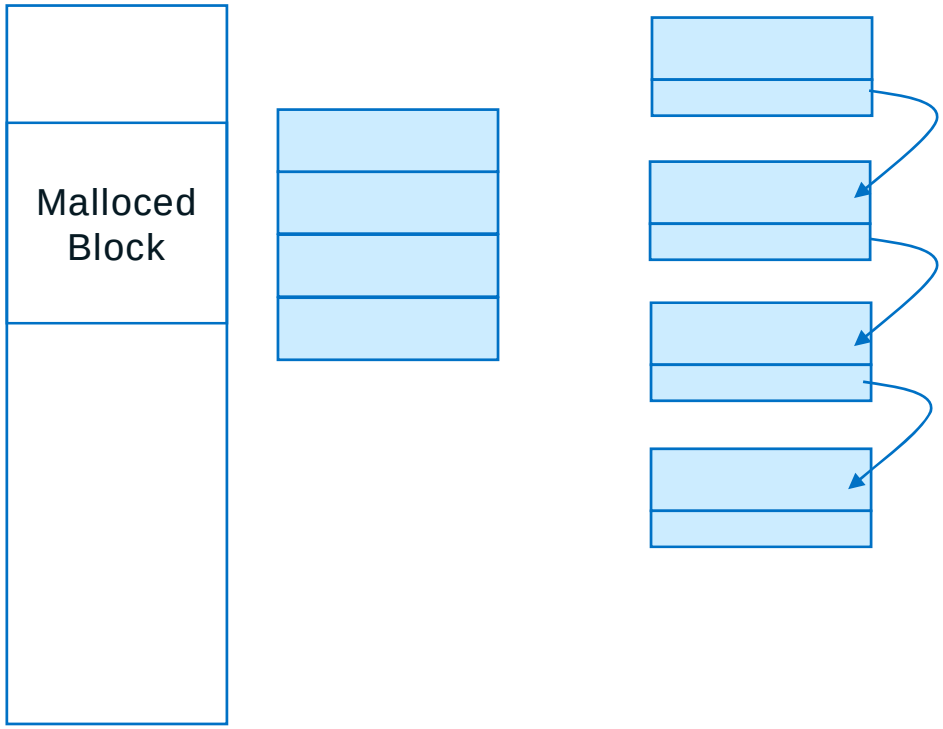


What is the likely impact of this?

What solution(s) can you recommend?

A Solution: Linked List

Overload memory allocation to use memory from an array.



Question: loopy data

A Piece of code accesses data in a loop

- The addresses are at non-contiguous address.
- Iterating through the loop causes cache misses

What is the likely impact of this?

What solution(s) can you recommend?

A solution: loopy data

Use prefetching

```
#define pref_dist 64
#ifdef PREFETCH
for (i=0; i<pref_dist;i+=4){
   _mm_prefetch((const char *)&b[i],2);
   _mm_prefetch((const char *)&d[i],2);
}
#endif

for(i=0; i<len; i++) {
   ……
   a[i] = tem + a1*d[i];

   #ifdef PREFETCH
   if (!(i % 4)) {
            _mm_prefetch((const char *)&b[i+pref_dist],2);
            _mm_prefetch((const char *)&d[i+pref_dist],2);
}
#endif
```

Legal Disclaimer & Optimization Notice

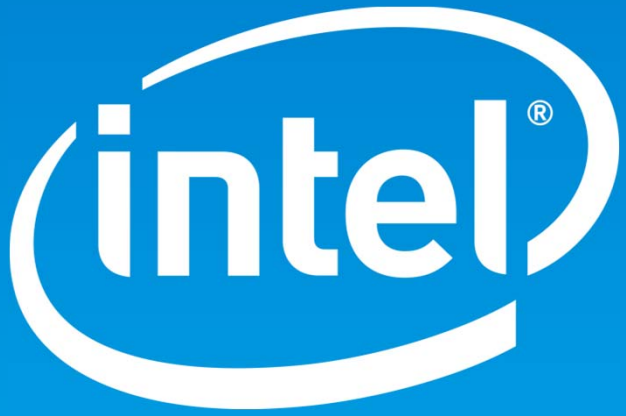
INFORMATION IN THIS DOCUMENT IS PROVIDED “AS IS”. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO THIS INFORMATION INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products.

Optimization Notice

Intel’s compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice.

Notice revision #20110804


intel