technology
from seed

Monitoring Performance and Power for Application Characterization with Cache-aware Roofline Model

Diogo Antão, Luís Taniça, Aleksandar Ilic,
Frederico Pratas, Pedro Tomás, Leonel Sousa

Outline

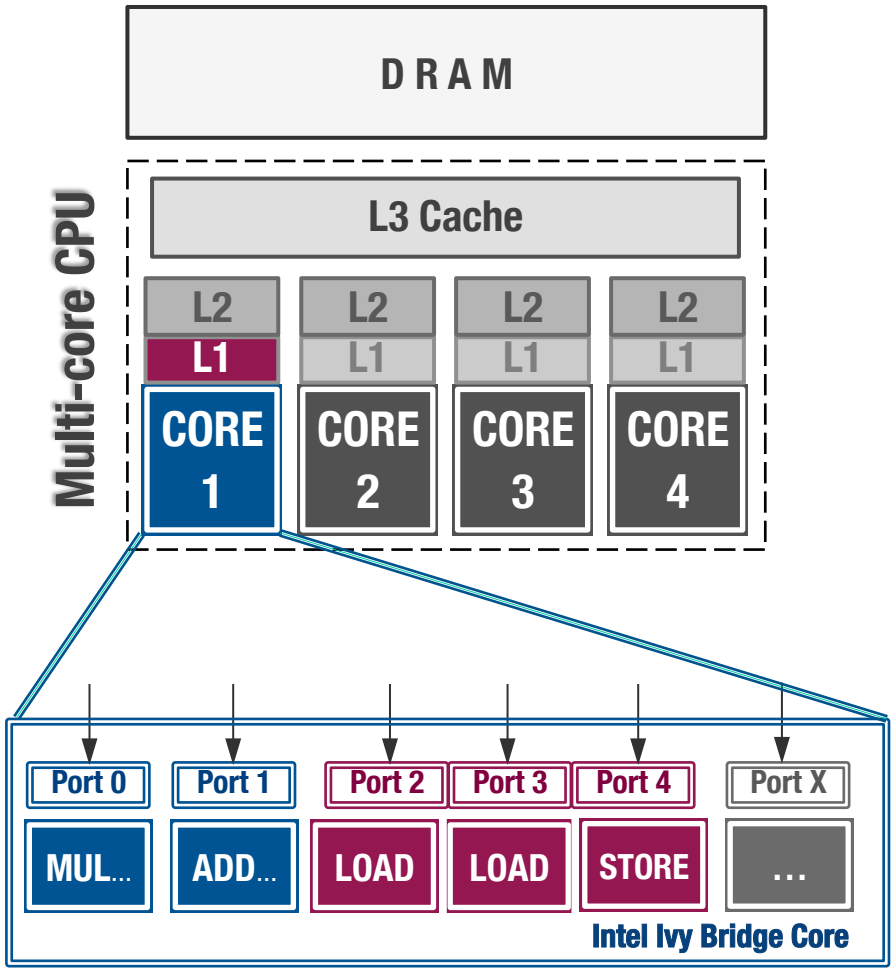
- Motivation
- Modern multi-cores and Cache-aware Roofline Model
- Application monitoring
  - SPYMON: user-space monitoring tool
  - KERMON: kernel-space monitoring approach
- Application Characterization
- Conclusions

Motivation

- Modern multi-core CPUs
  - Complex general purpose architectures
  - Provide high performance and energy-efficient computing

- How much performance can they deliver?
  - Is a real application able to efficiently exploit their full potential?
  - How can we measure the performance, power and energy consumption of a real application?

Cache-aware Roofline Model



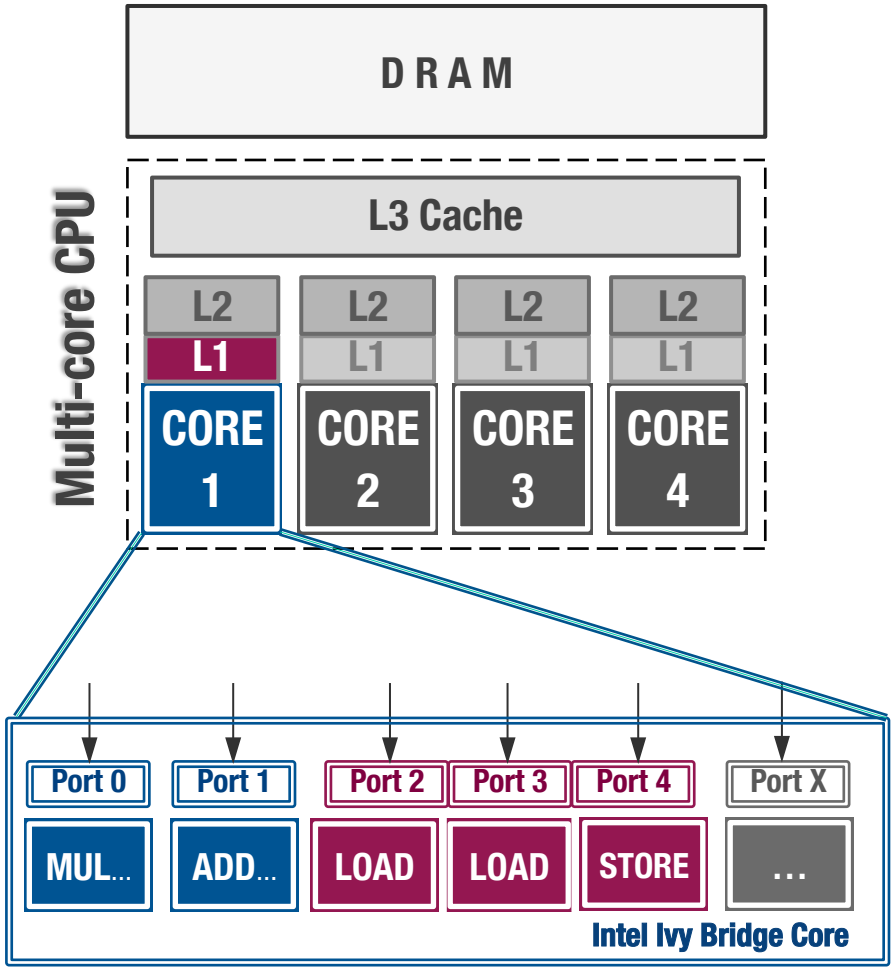
Computing Performance

- Parallelism across several (identical) cores
- In-core parallelism:
  - Several ports for different operations
  - Instruction-level parallelism (e.g., pipelining)

Memory Hierarchy

- Set of on-chip caches: private (L1, L2) or shared (L3)
- Global memory (DRAM)
- Caches hide the latency when accessing DRAM (also between successive cache levels)

Cache-aware Roofline Model



Computing Performance

- Parallelism across several (identical) cores
- In-core parallelism:
  - Several ports for different operations
  - Instruction-level parallelism (e.g., pipelining)

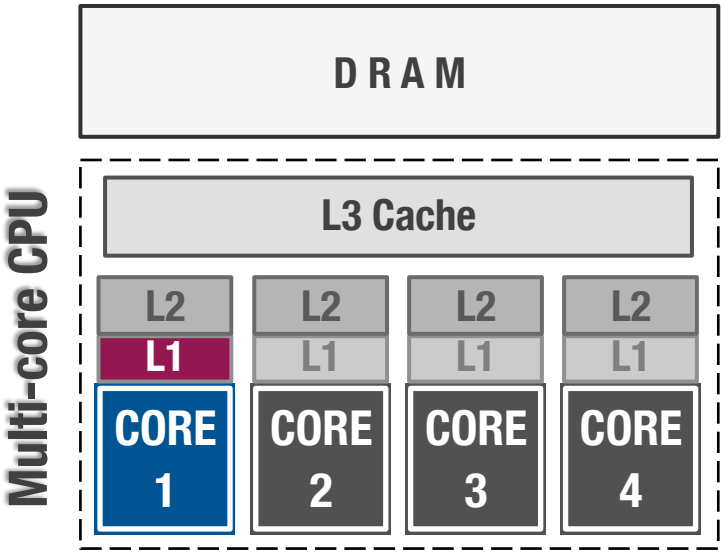
Memory Hierarchy

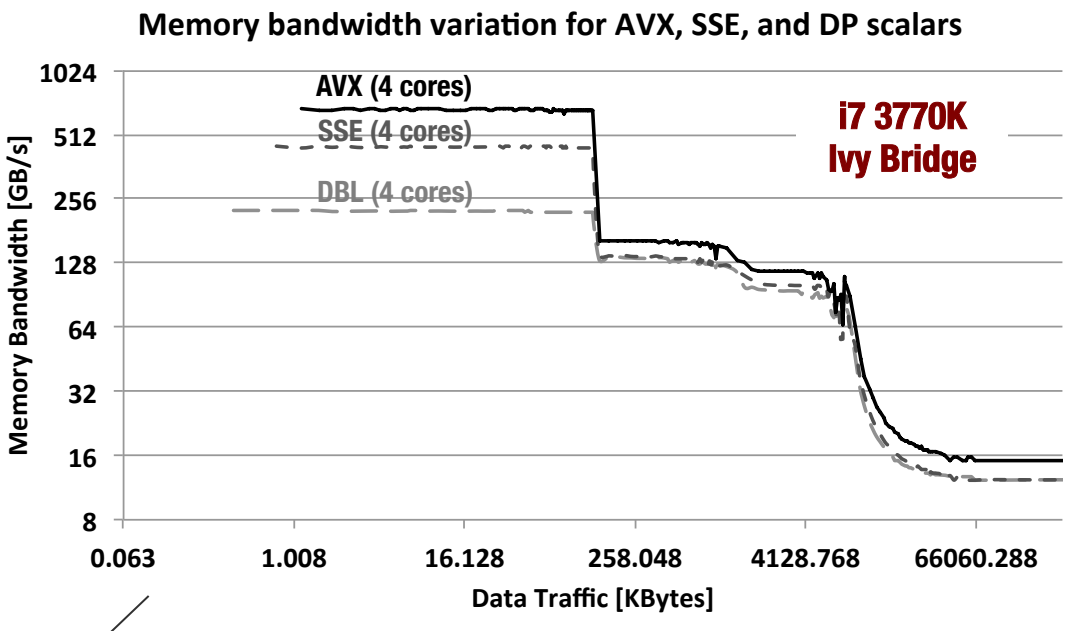
- Set of on-chip caches: private (L1, L2) or shared (L3)
- Global memory (DRAM)
- Caches hide the latency when accessing DRAM (also between successive cache levels)

Cache-Aware Roofline Model

- Insightful model of modern multi-core architectures, that relates:
  1) Maximum attainable performance Fp (flops/time)
  2) Operational intensity I (flops/bytes)
- **Takes into account the complete memory hierarchy**

Cache-aware Roofline Model
- Memory Hierarchy -



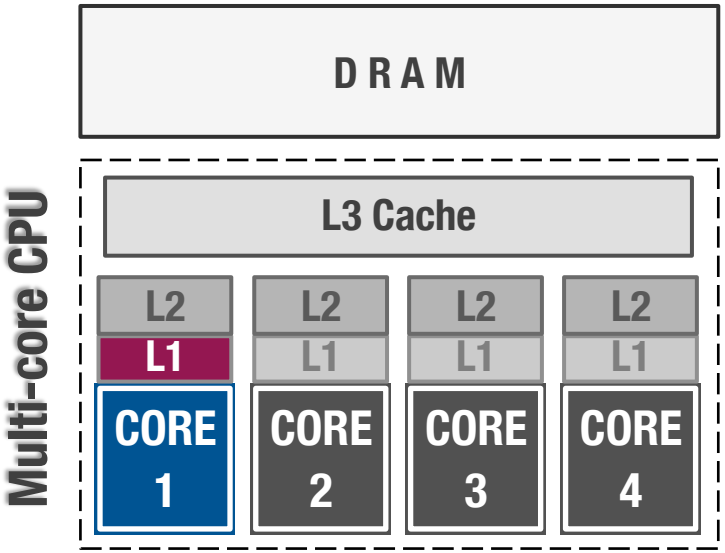


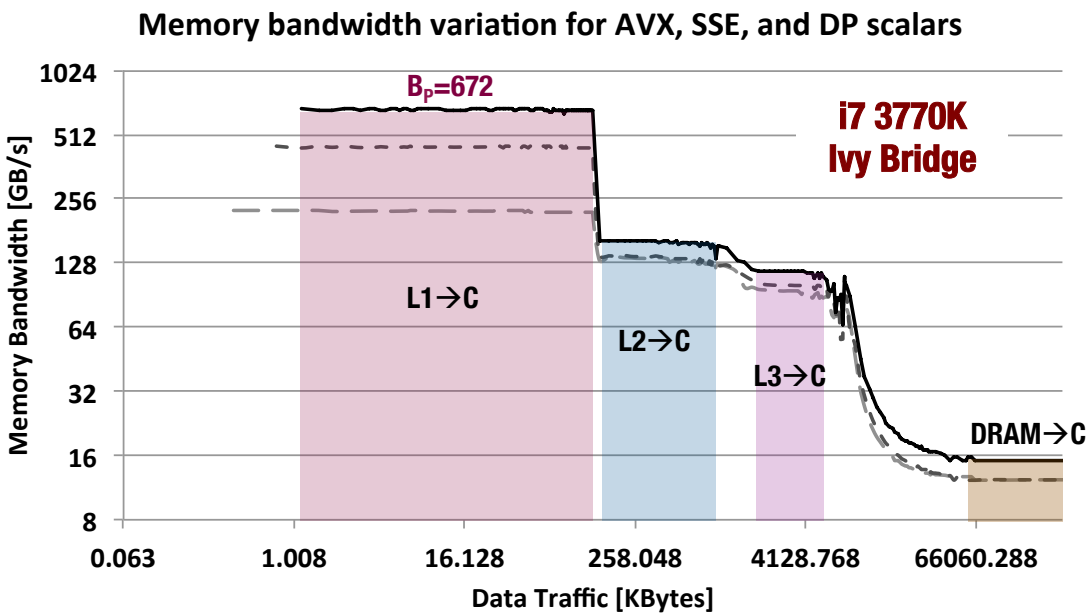
How to measure?

```
// Configured Performance Counters
CPU_CLK_UNHALTED.CORE/REF
MEM_UOP_RETIRED.ALL_LOADS
MEM_UOP_RETIRED.ALL_STORES
...
```

```
// AVX Assembly code: 2 Loads + 1 Store
vmovapd    0(%rax), %ymm0
vmovapd   32(%rax), %ymm1
vmovapd      %ymm2, 64(%rax)
vmovapd   96(%rax), %ymm3
vmovapd  128(%rax), %ymm4
vmovapd      %ymm5, 160(%rax)
...
```

Cache-aware Roofline Model
- Memory Hierarchy -

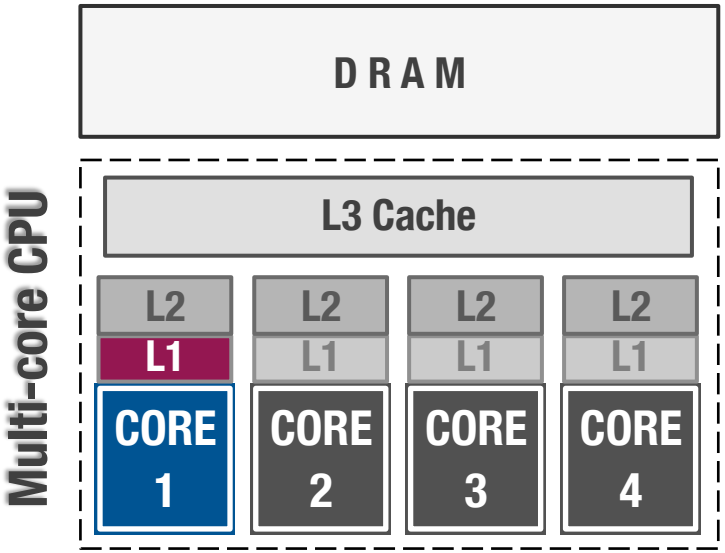


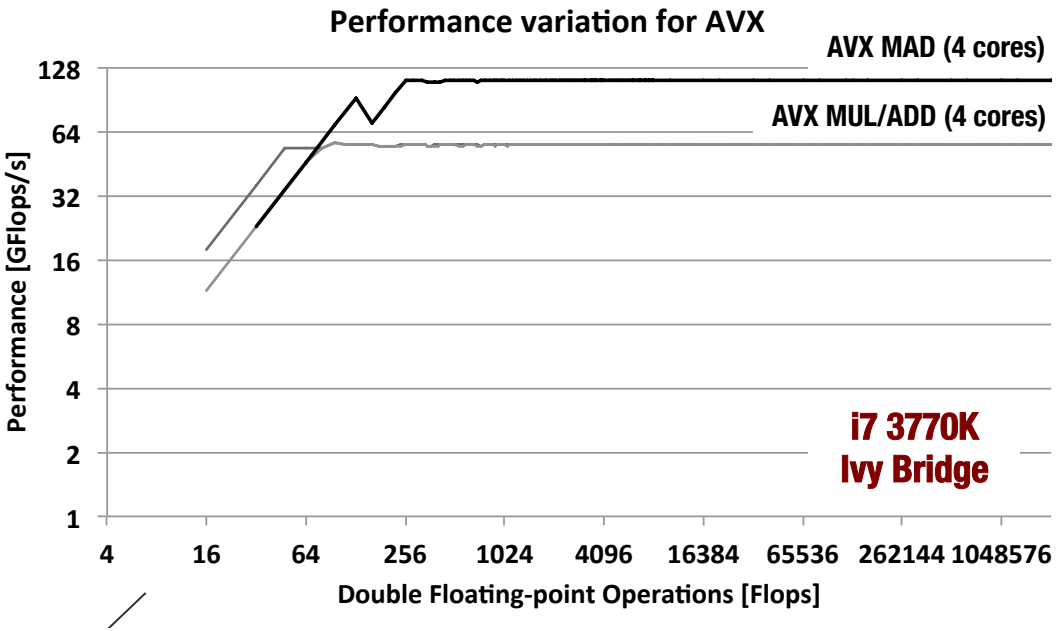


i7 3770K Ivy Bridge	Perf. [F_P] (GFlops/s)*	Bwidth L1→C [B_P] (GB/s)*
1 Core	28	168
4 Cores	112	672
*256-bit AVX double-precision floating-point instructions		

```
// AVX Assembly code: 2 Loads + 1 Store
vmovapd   0(%rax), %ymm0
vmovapd  32(%rax), %ymm1
vmovapd    %ymm2, 64(%rax)
vmovapd  96(%rax), %ymm3
vmovapd 128(%rax), %ymm4
vmovapd    %ymm5, 160(%rax)
...
```

Cache-aware Roofline Model
- Performance -



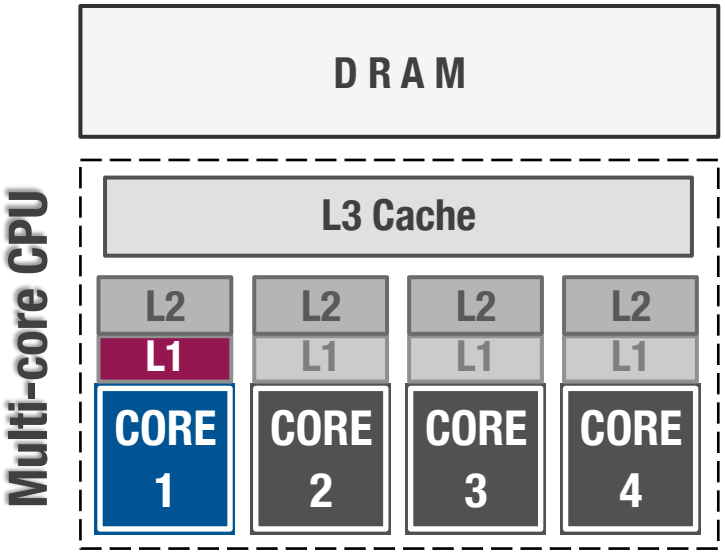


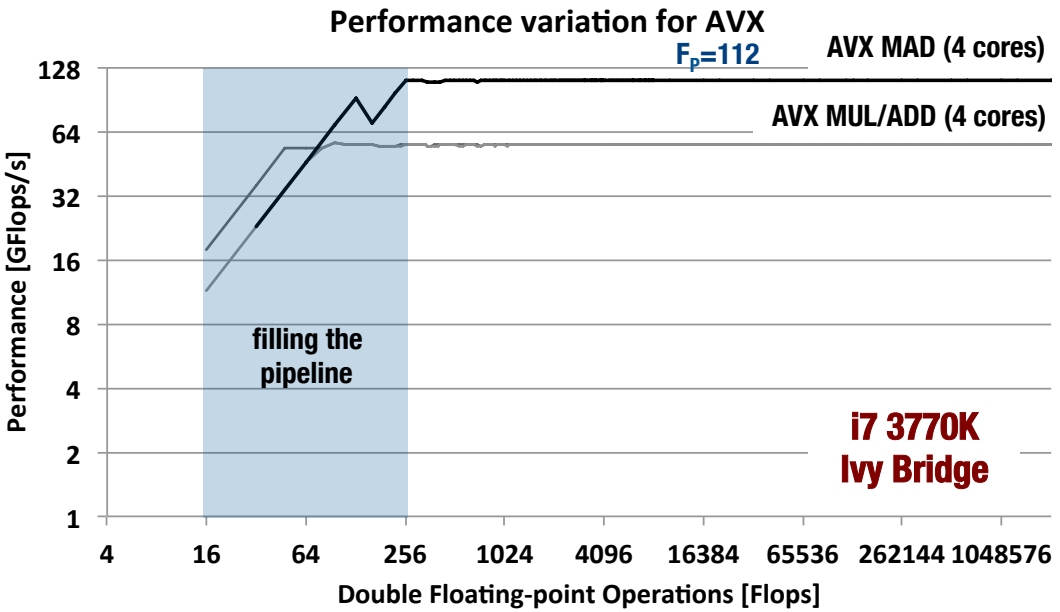
How to measure?

```
// Configured Performance Counters
CPU_CLK_UNHALTED.CORE/REF
FP_OPS_EXE_SSE_SCALAR_DBL
FP_OPS_EXE_SSE_FP_PACKED_DBL
SIMD_FP_256_PACKED_DBL
...
```

```
// AVX Assembly code: MUL+ADD
vmulpd    %ymm0, %ymm0, %ymm0
vaddpd    %ymm1, %ymm1, %ymm1
vmulpd    %ymm2, %ymm2, %ymm2
vaddpd    %ymm3, %ymm3, %ymm3
vmulpd    %ymm4, %ymm4, %ymm4
vaddpd    %ymm5, %ymm5, %ymm5
...
```

Cache-aware Roofline Model
- Performance -

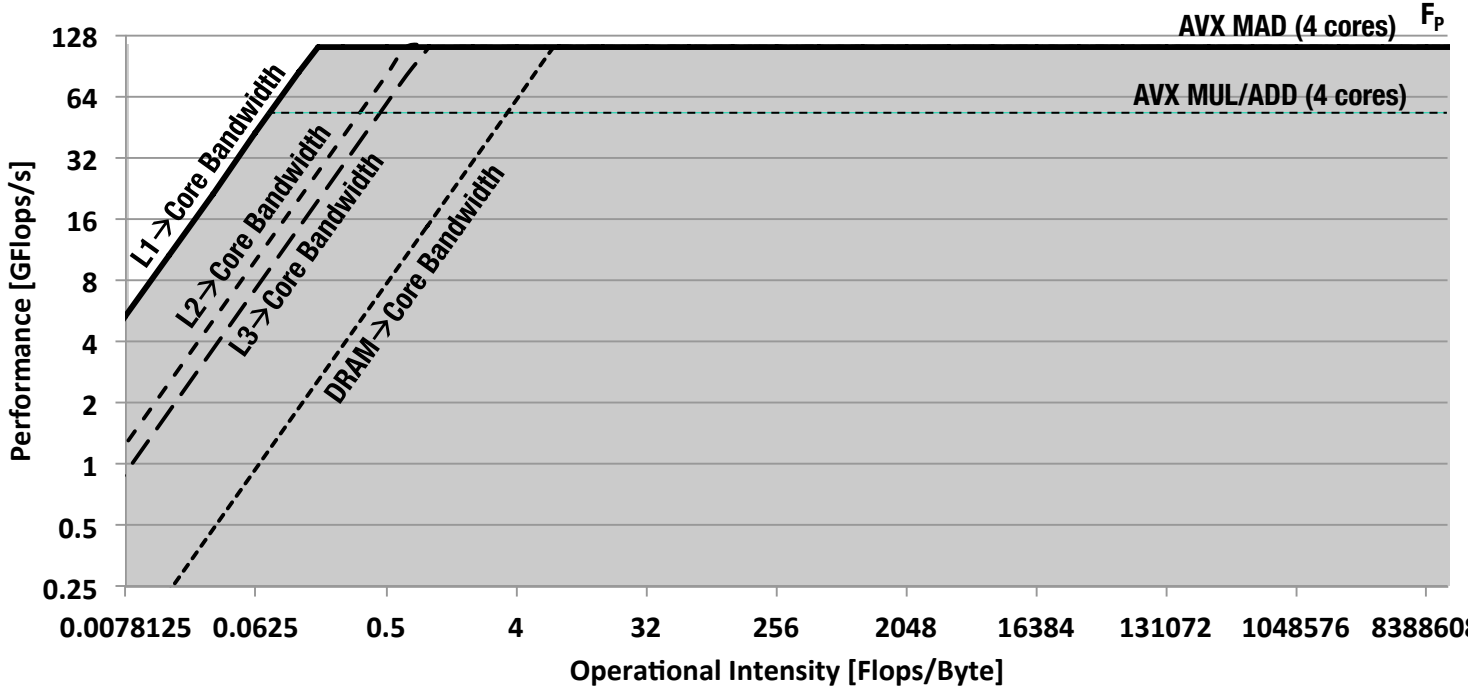




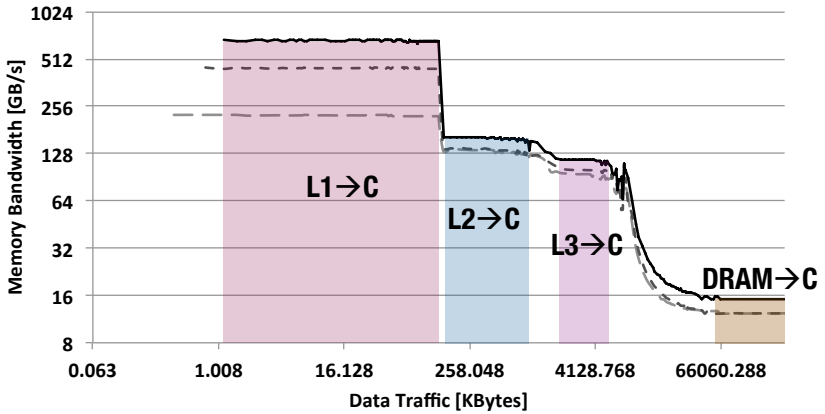
i7 3770K Ivy Bridge	Perf. [F_P] (GFlops/s)*	Bwidth L1→C [B_P] (GB/s)*
1 Core	28	168
4 Cores	112	672
*256-bit AVX double-precision floating-point instructions		

```
// AVX Assembly code: MUL+ADD
vmulpd    %ymm0, %ymm0, %ymm0
vaddpd    %ymm1, %ymm1, %ymm1
vmulpd    %ymm2, %ymm2, %ymm2
vaddpd    %ymm3, %ymm3, %ymm3
vmulpd    %ymm4, %ymm4, %ymm4
vaddpd    %ymm5, %ymm5, %ymm5
...
```

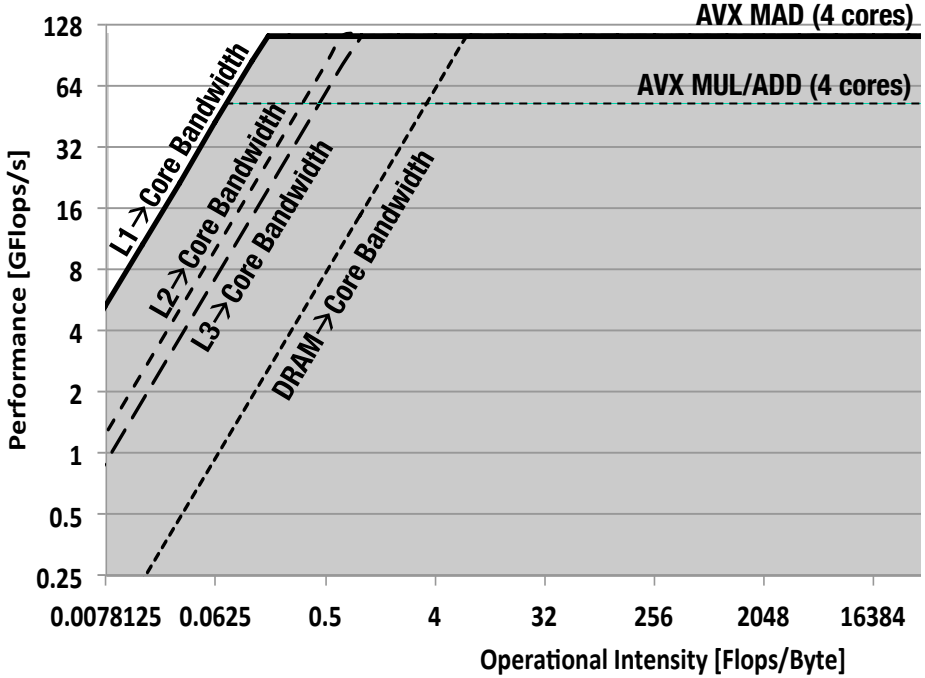
Cache-Aware Roofline Model
- Putting it all together -



i7 3770K Ivy Bridge	Perf. [F_P] (GFlops/s)*	Bwidth L1→C [B_P] (GB/s)*
1 Core	28	168
4 Cores	112	672
*256-bit AVX double-precision floating-point instructions		



Cache-Aware Roofline Model
- Putting it all together -

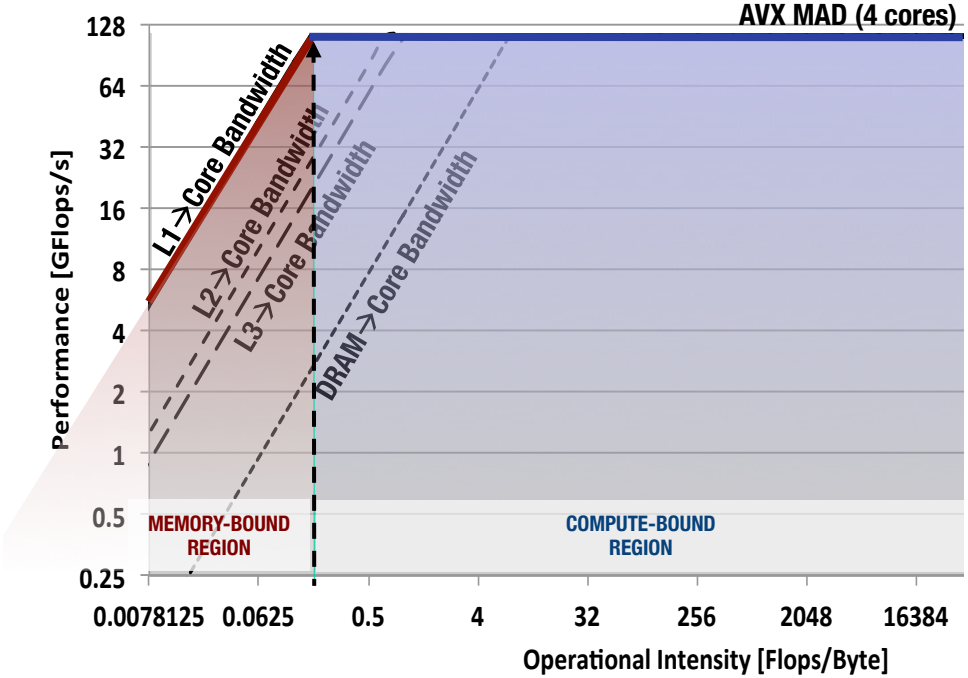


i7 3770K Ivy Bridge	Perf. [F_P] (GFlops/s)*	Bwidth L1→C [B_P] (GB/s)*
1 Core	28	168
4 Cores	112	672
*256-bit AVX double-precision floating-point instructions		

Some important **properties** :

- **Single plot model**
  - graphically shows the performance limits of the architecture
  - constructed once per architecture (from processor specifications or with micro-benchmarks)
- Explicitly considers **all levels of memory hierarchy**
  - models the influence of caches and DRAM to the attainable performance
- Applicable to the **other types of operations**
  - not necessarily floating-point
- **Useful for** characterization and optimization of a wide range of **applications**
- Development and understanding of current and future architectures

Cache-Aware Roofline Model
- Application characterization -



i7 3770K Ivy Bridge	Perf. [F_P] (GFlops/s)*	Bwidth L1→C [B_P] (GB/s)*
1 Core	28	168
4 Cores	112	672
*256-bit AVX double-precision floating-point instructions		

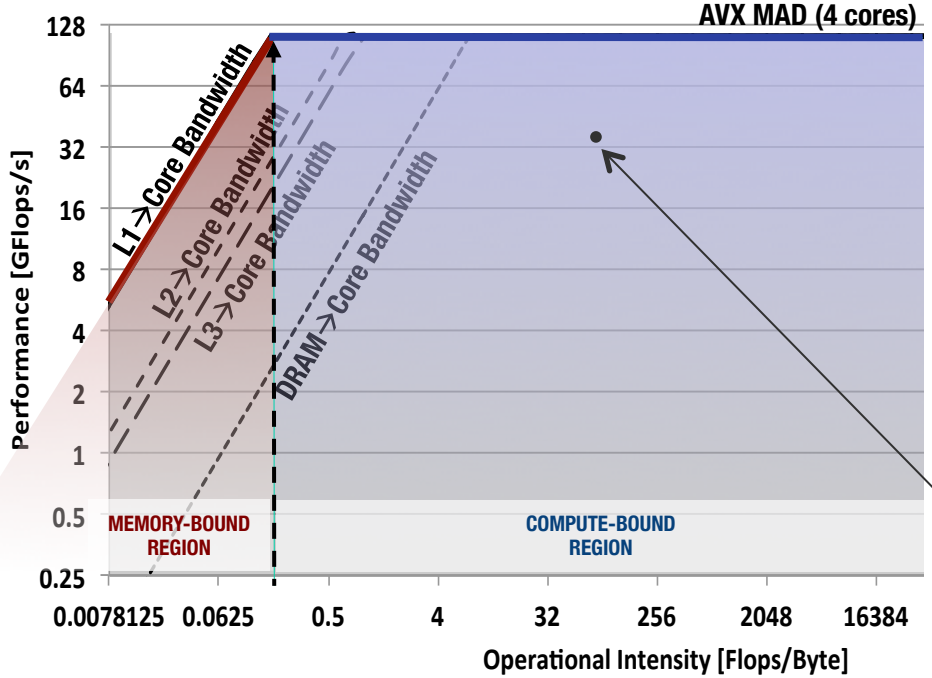
Application characterization:

- Memory-bound applications
- Compute-bound applications

Ridge point:

- Minimum OI to reach maximum performance
- Memory transfers and computations are completely overlapped

Cache-Aware Roofline Model
- Application characterization -



Application characterization:

- Memory-bound applications
- Compute-bound applications

Ridge point:

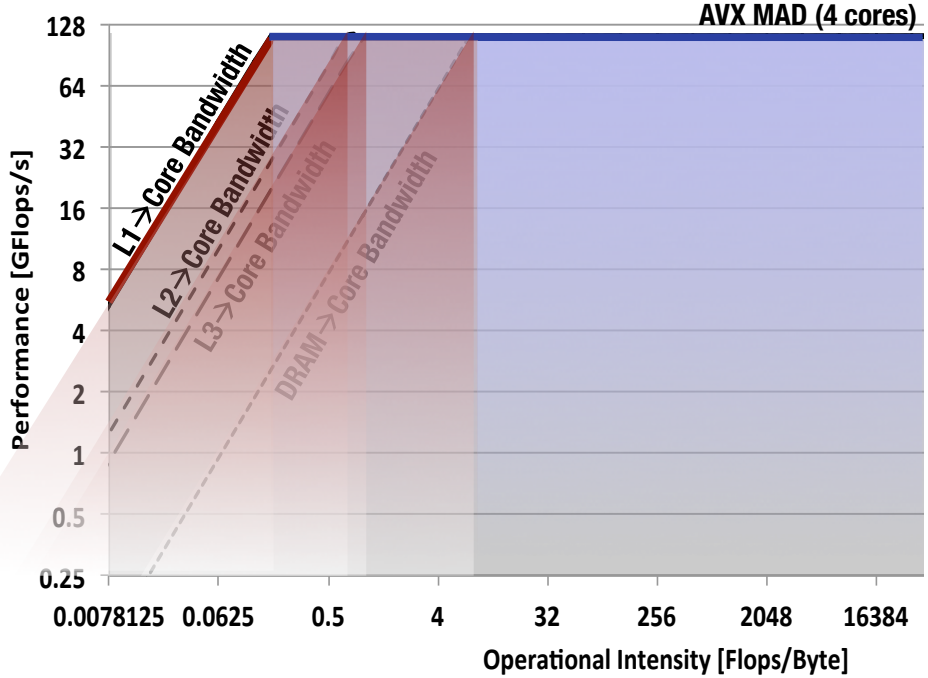
- Minimum OI to reach maximum performance
- Memory transfers and computations are completely overlapped

Application is a SINGLE POINT in the model!

- If no monitoring techniques are applied

i7 3770K Ivy Bridge	Perf. [F_P] (GFlops/s)*	Bwidth L1→C [B_P] (GB/s)*
1 Core	28	168
4 Cores	112	672
*256-bit AVX double-precision floating-point instructions		

Cache-Aware Roofline Model
- Application characterization -



i7 3770K Ivy Bridge	Perf. [F_P] (GFlops/s)*	Bwidth L1→C [B_P] (GB/s)*
1 Core	28	168
4 Cores	112	672
*256-bit AVX double-precision floating-point instructions		

Application characterization:

- Memory-bound applications
- Compute-bound applications

Ridge point:

- Minimum OI to reach maximum performance
- Memory transfers and computations are completely overlapped

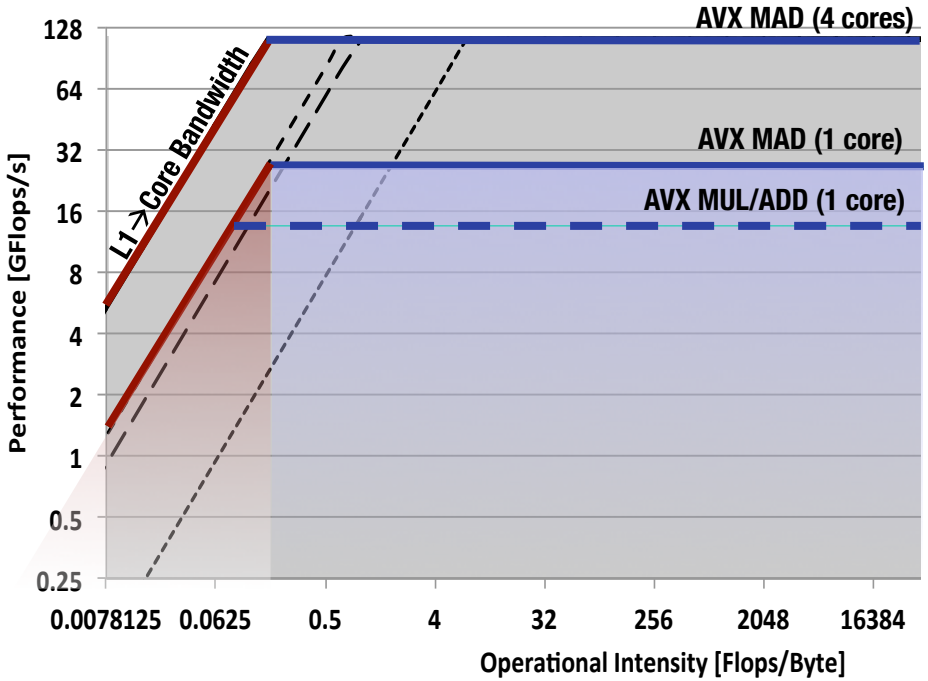
Application is a SINGLE POINT in the model!

- If no monitoring techniques are applied

Characterization in complex memory hierarchy

- Depends on where most accesses occur and their combination

Cache-Aware Roofline Model
- Single core models -

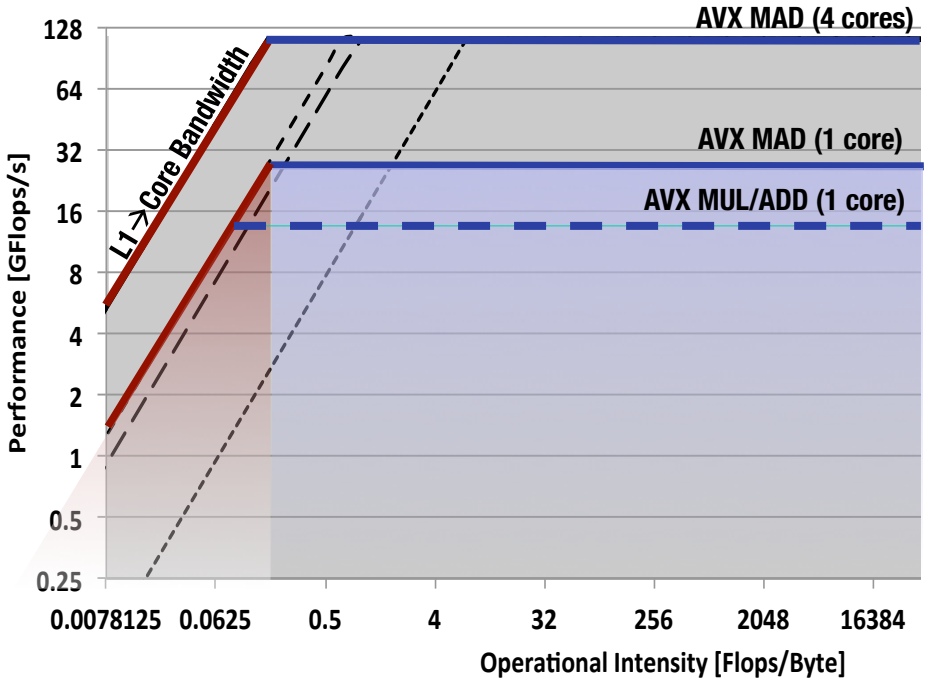


MODEL FOR 4 CORES

MODEL FOR 1 CORE

i7 3770K Ivy Bridge	Perf. [F_P] (GFlops/s)*	Bwidth L1→C [B_P] (GB/s)*
1 Core	28	168
4 Cores	112	672
*256-bit AVX double-precision floating-point instructions		

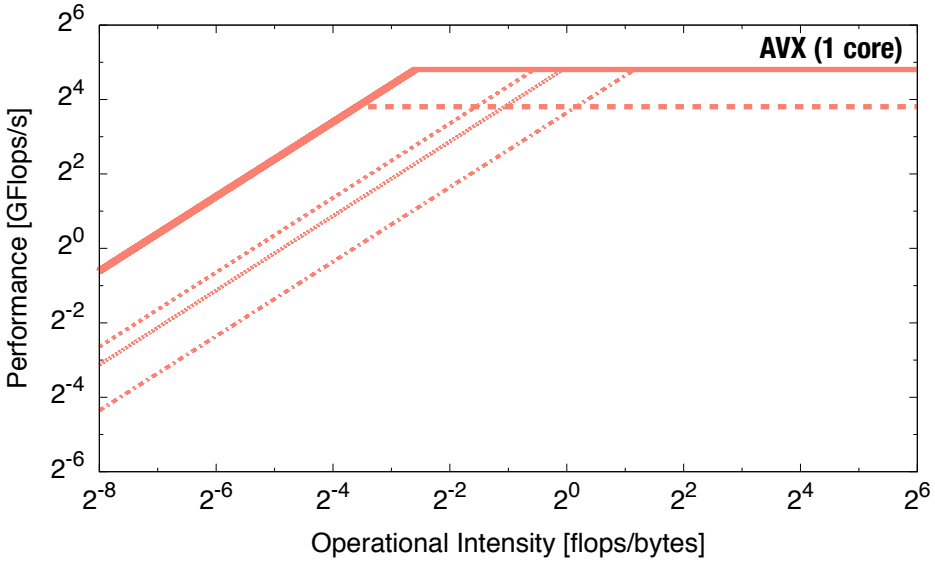
Cache-Aware Roofline Model
- Single core models -



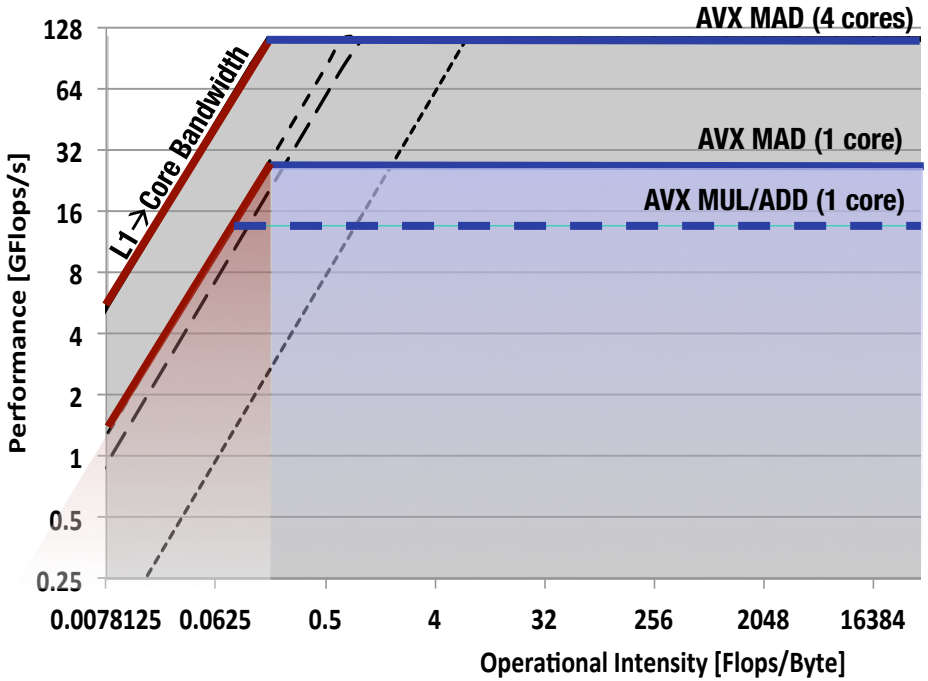
MODEL FOR 4 CORES

MODEL FOR 1 CORE

i7 3770K Ivy Bridge	Perf. [F_P] (GFlops/s)*	Bwidth L1→C [B_P] (GB/s)*
1 Core	28	168
4 Cores	112	672
*256-bit AVX double-precision floating-point instructions		



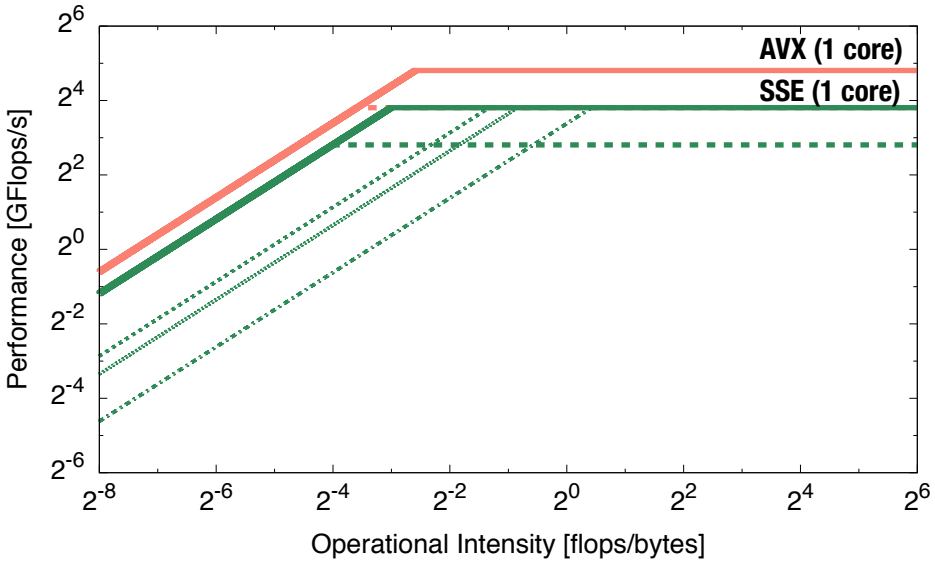
Cache-Aware Roofline Model
- Single core models -



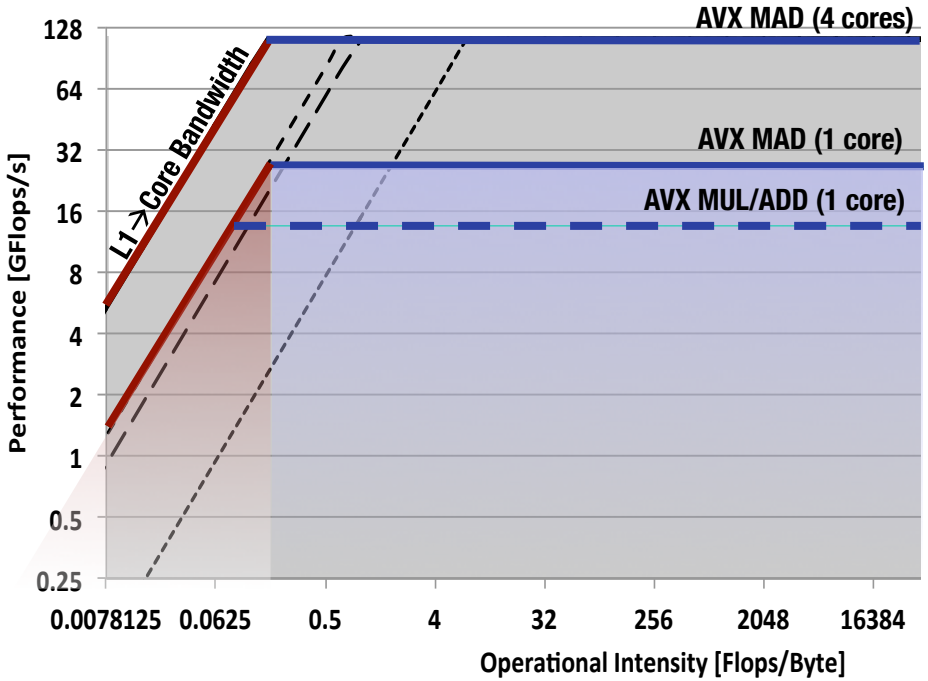
MODEL FOR 4 CORES

MODEL FOR 1 CORE

i7 3770K Ivy Bridge	Perf. [F_P] (GFlops/s)*	Bwidth L1→C [B_P] (GB/s)*
1 Core	28	168
4 Cores	112	672
*256-bit AVX double-precision floating-point instructions		



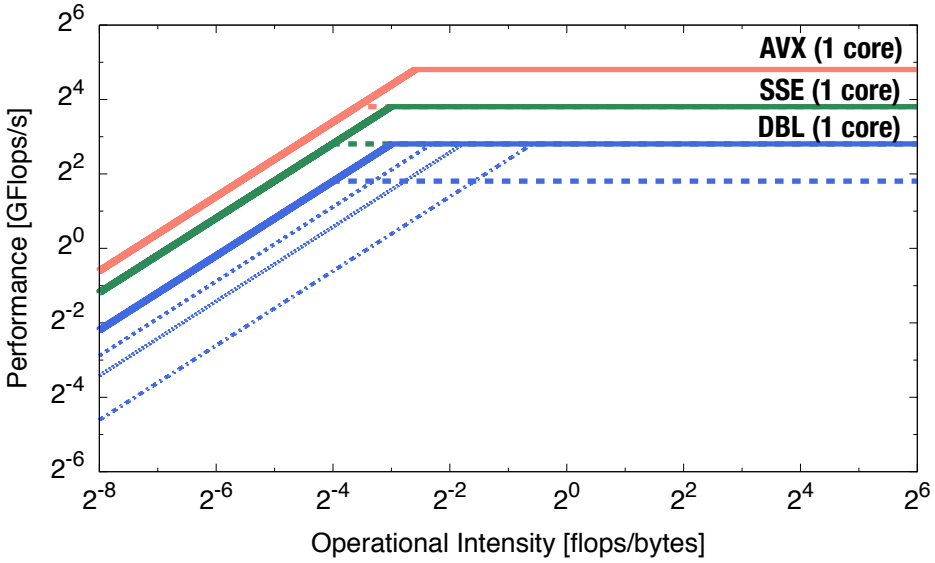
Cache-Aware Roofline Model
- Single core models -



MODEL FOR 4 CORES

MODEL FOR 1 CORE

i7 3770K Ivy Bridge	Perf. [F_P] (GFlops/s)*	Bwidth L1→C [B_P] (GB/s)*
1 Core	28	168
4 Cores	112	672
*256-bit AVX double-precision floating-point instructions		



Outline

- Motivation
- Modern multi-cores and Cache-aware Roofline Model
- Application monitoring
  - SPYMON: user-space monitoring tool
  - KERMON: kernel-space monitoring approach
- Application Characterization
- Conclusions

Application Monitoring

► SPYMON

- Application monitoring from the user space, **SPYMON**:
  - lightweight, simple and adjustable for the user needs
  - follows a core-oriented approach, monitoring the behavior of each individual logical core
  - captures the information of all running applications

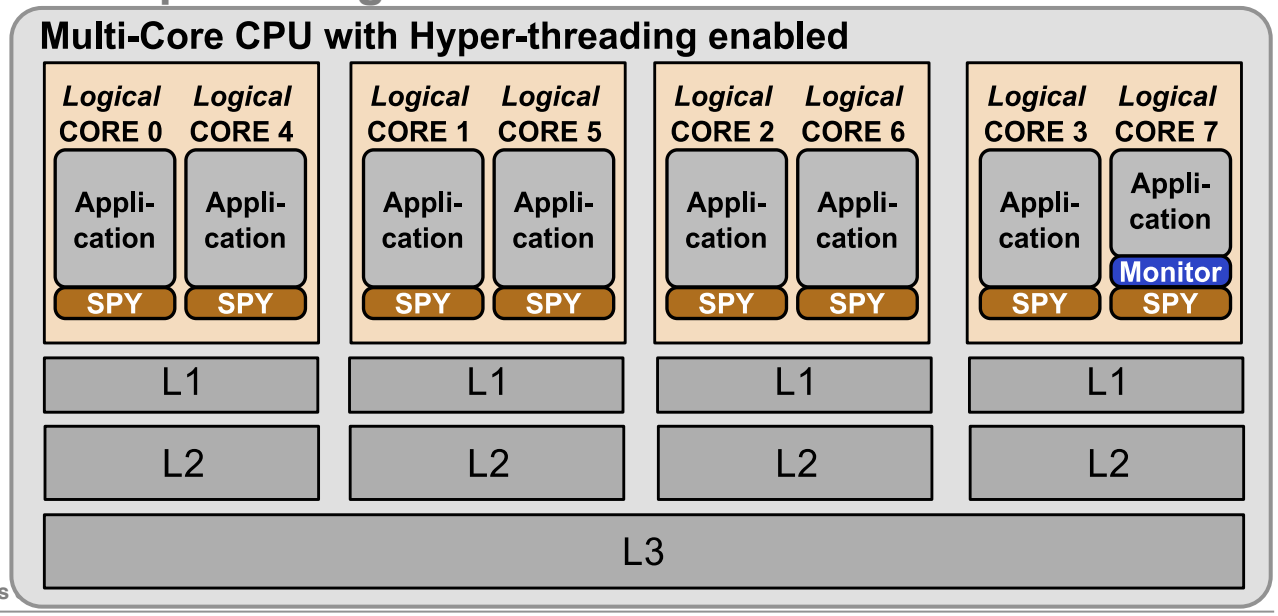
Monitor:

Responsible for all the required initializations, for data analysis and processing and for controlling the whole tool.

Spy:

lightweight process, bonded to an individual logical core, for configuring and fetching the performance counter readings

Example configuration:



Application Monitoring

► SPYMON

- Communication made by means of signals and pipes to minimize cache pollution
  - Reduces the interference of the SPYMON tool on the running applications
  - Bidirectional pipes allow the master to change, at run-time, the events to be monitored in a logical core (e.g., because a running application changed state, or because it changed logical core)

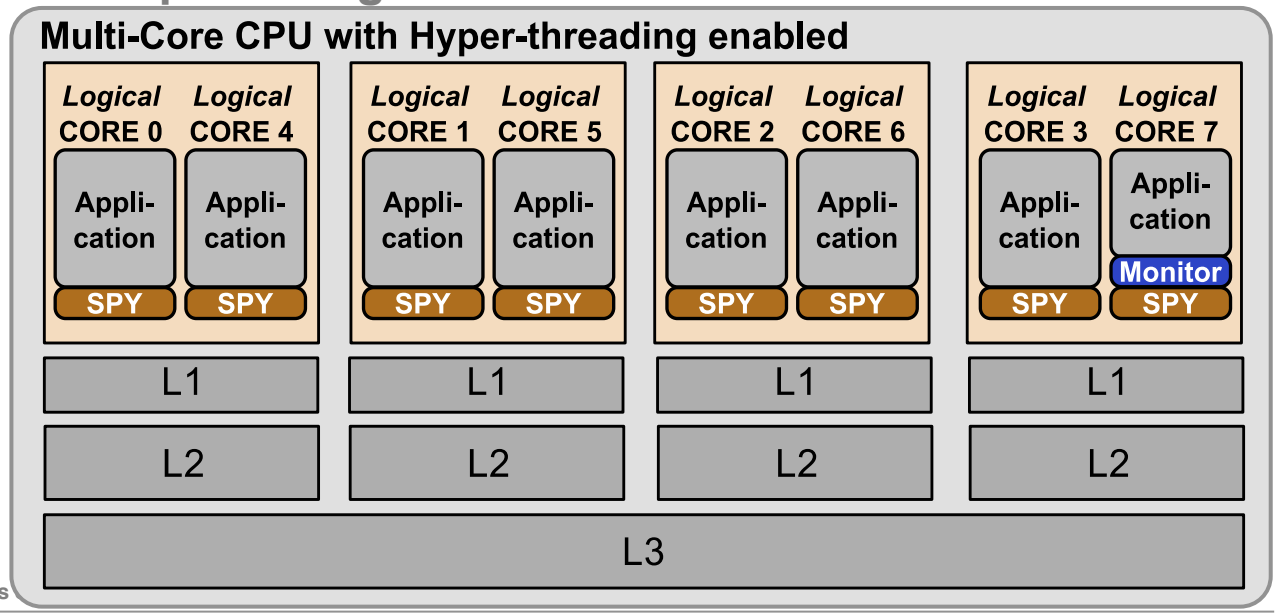
Monitor:

Responsible for all the required initializations, for data analysis and processing and for controlling the whole tool.

Spy:

lightweight process, bonded to an individual logical core, for configuring and fetching the performance counter readings

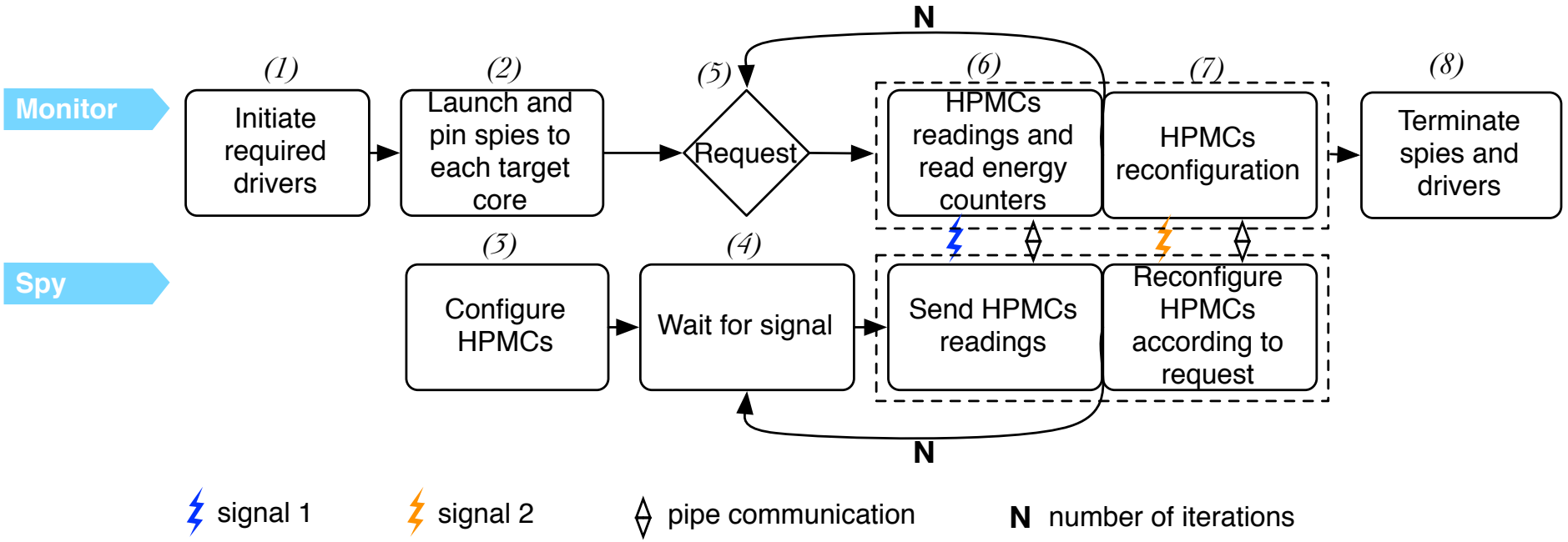
Example configuration:



Application Monitoring

▶ SPYMON

- SPYMON execution diagram



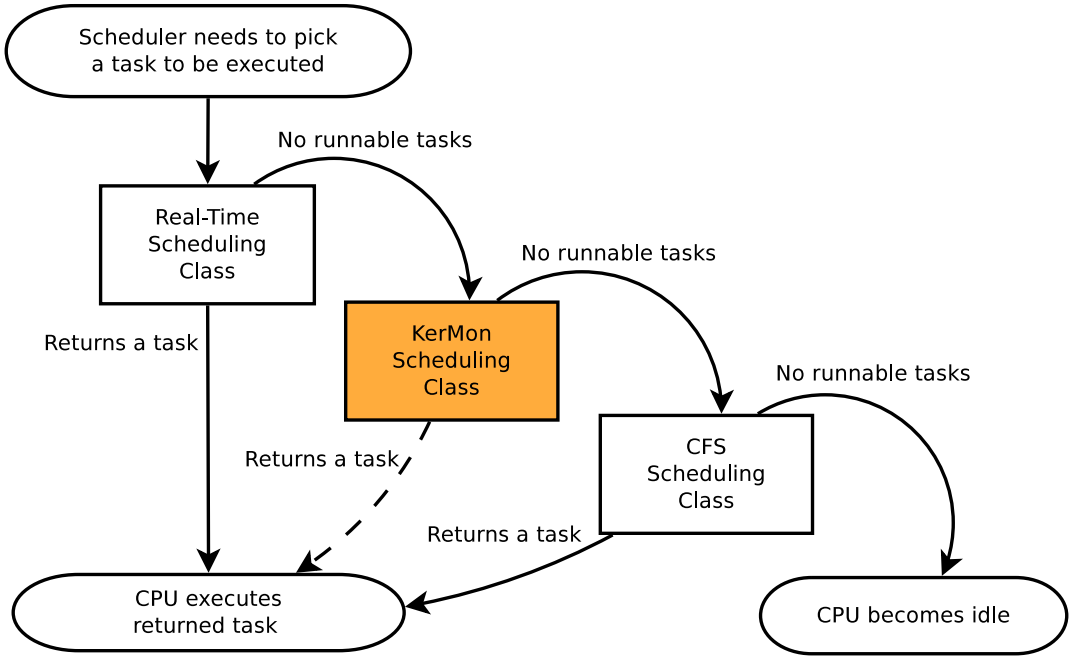
HPMC – Hardware performance measurement Counter

Application Monitoring

► KERMON

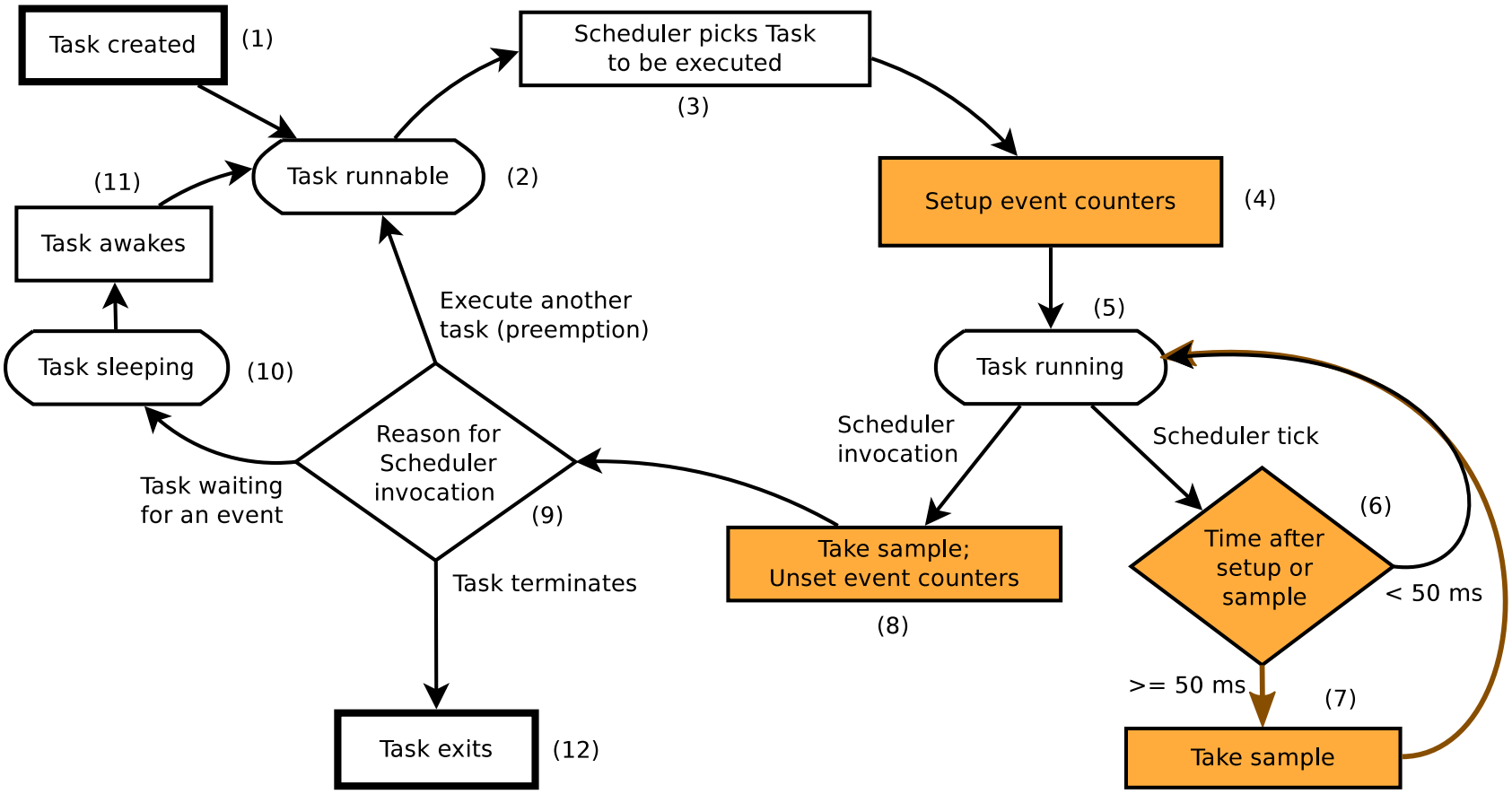
- Application monitoring from the kernel space, **KERMON**:
  - Allows for thread oriented application monitoring
  - Allows thread-level monitoring of an application that spawns multiple threads in real-time (e.g., OpenCL)
  - Requires patching the system kernel

- Based on the Completely Fair Scheduler (CFS)
  - The default scheduler from Linux 2.6.23
  - The introduced KerMon class has a higher priority than the CFS class to provide more precise monitoring and isolation



Application Monitoring

► KERMON



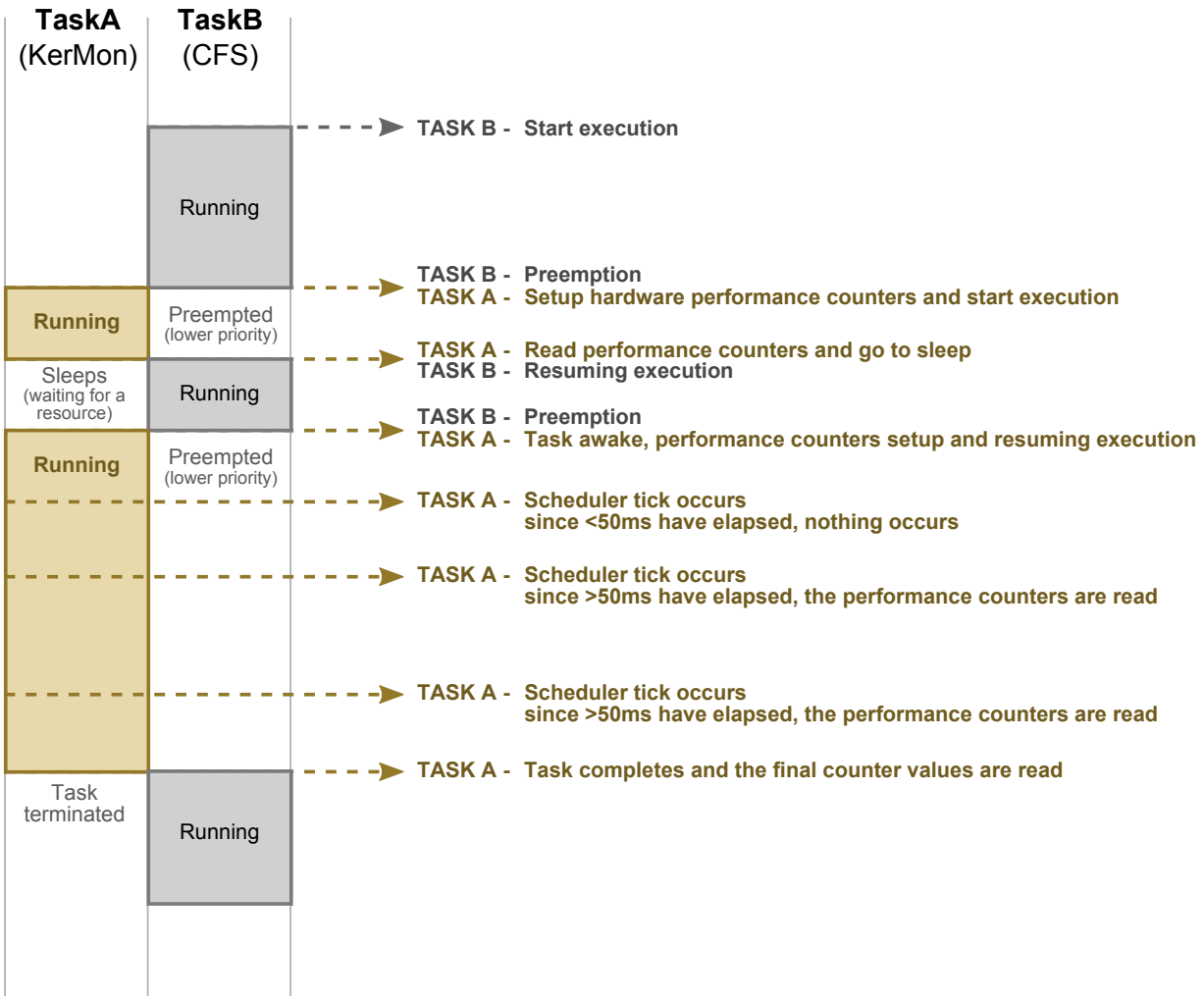
► KERMON

The KerMon class has a higher priority than the CFS class

- Task B is preempted whenever Task A becomes ready for execution

The performance counters:

- Are setup whenever the task starts/resumes execution
- Are read whenever the task:
  a) is preempted
  b) is finished
  c) is running for more than 50 ms



Outline

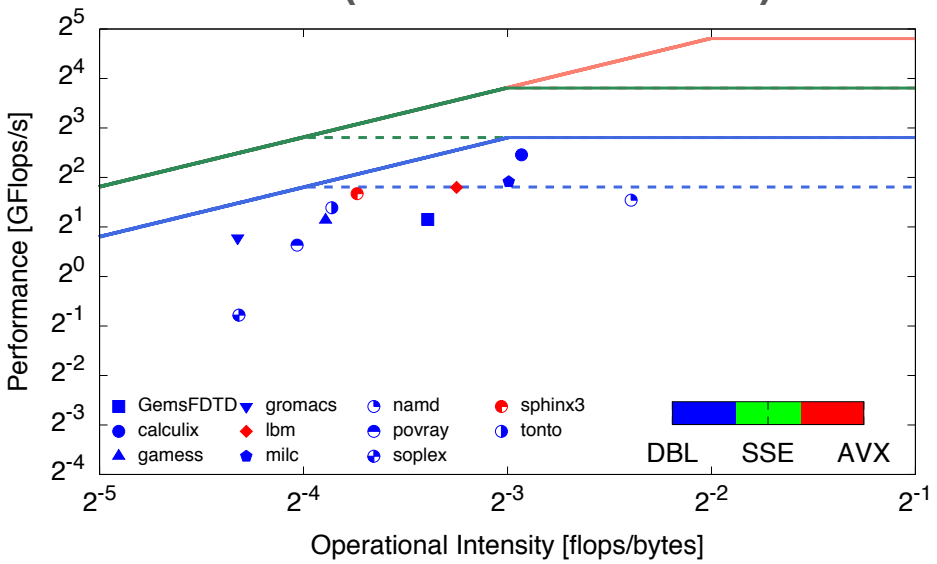
- Motivation
- Modern multi-cores and Cache-aware Roofline Model
- Application monitoring
  - SPYMON: user-space monitoring tool
  - KERMON: kernel-space monitoring approach
- Application Characterization
- Conclusions

Experimental Results: SPEC2006 - Floating-point benchmarks -

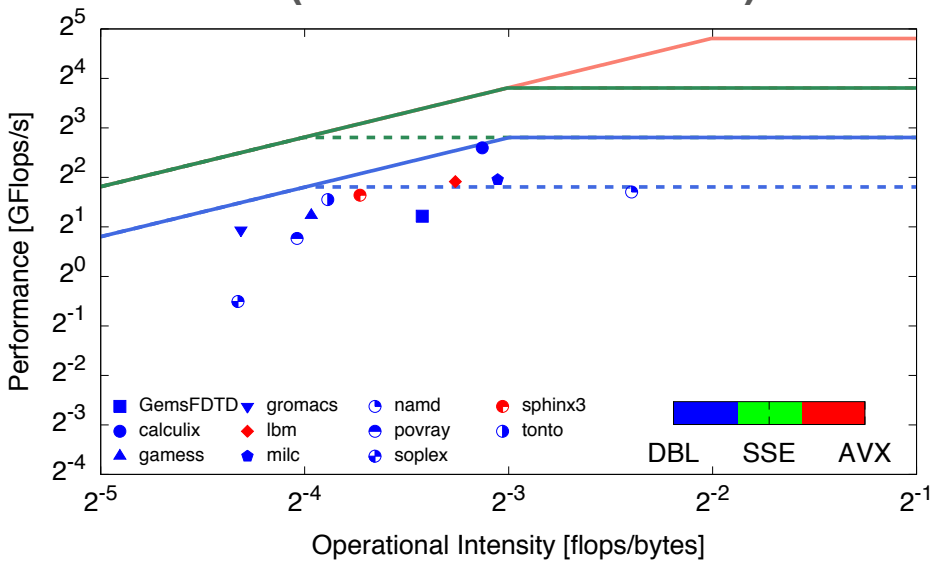
technology from seed

inesc id lisboa 10 anos

SPYMON (USER-SPACE MONITORING)



KERMON (KERNEL-SPACE MONITORING)

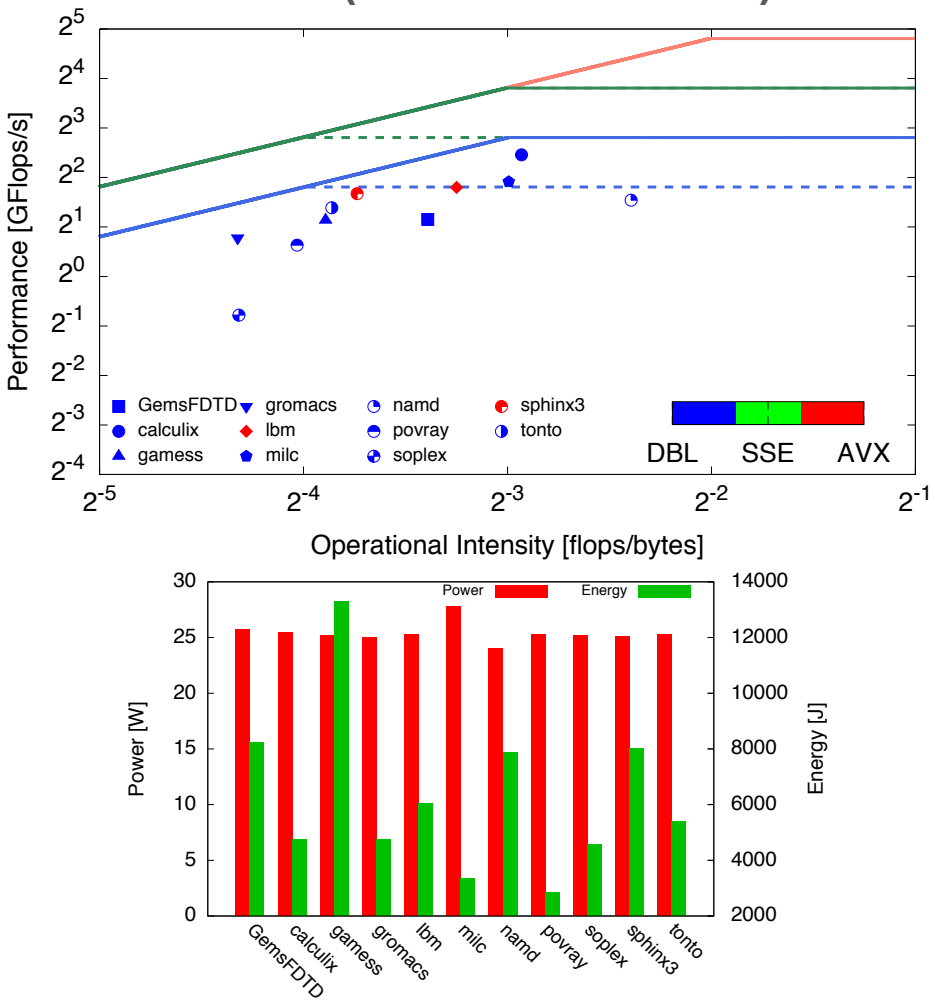


i7 3770K Ivy Bridge	
Frequency	3.5 GHz
L1/L2/L3 size	32/256/8192 KB
DRAM to L3 (DRAM3)	2 channels (8B)
	2x933 MHz

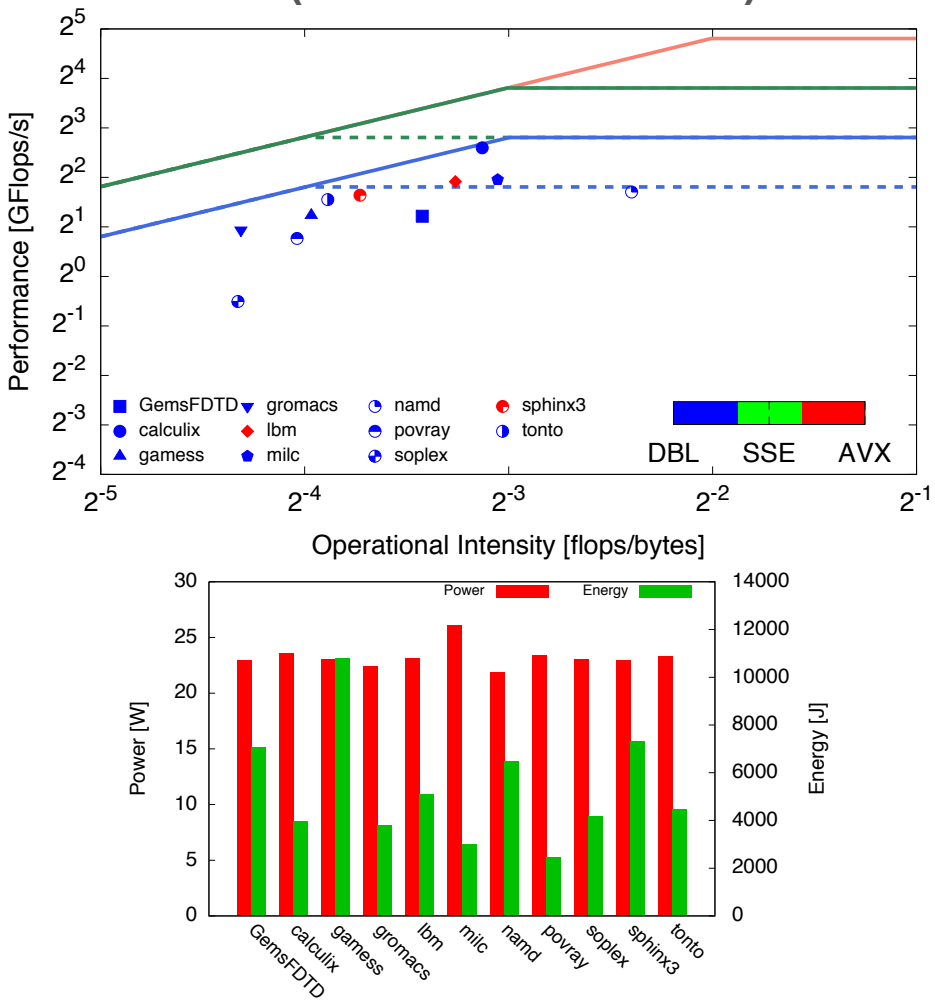
4 Event Sets	
Set 1	`RET_LDs, RET_STs, FP_x87, UOPS_RET`
Set 2	`FP_AVX_PD, FP_SSE_PD, FP_SSE_SD, UOPS_RET`
Set 3	`FP_AVX_PS, FP_SSE_PS, FP_SSE_SS, UOPS_RET`
Set 4	`L1D_REPL, L2_LINES_IN, OFF_CORE_REQ, UOPS_RET`

Experimental Results: SPEC2006 - Floating-point benchmarks -

SPYMON (USER-SPACE MONITORING)

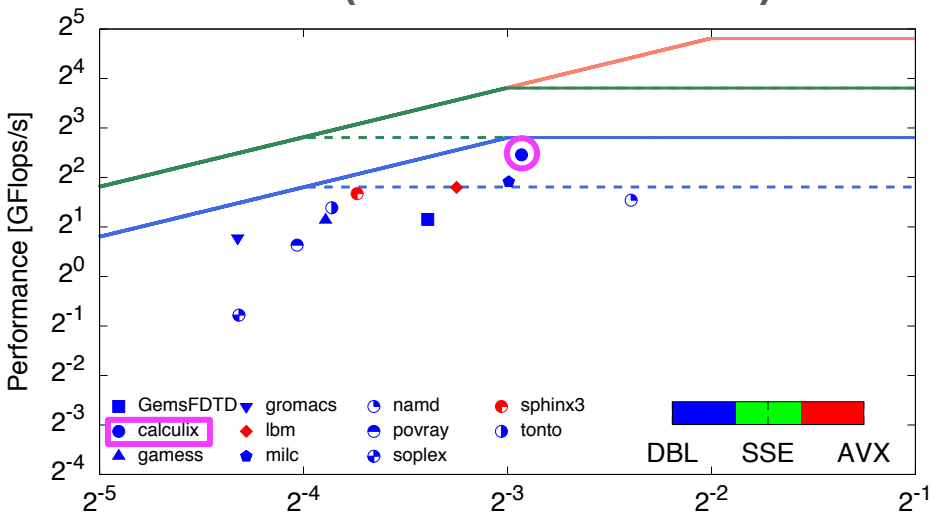


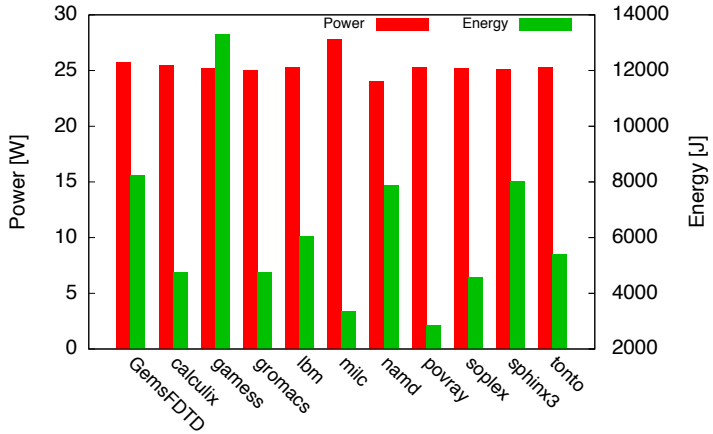
KERMON (KERNEL-SPACE MONITORING)



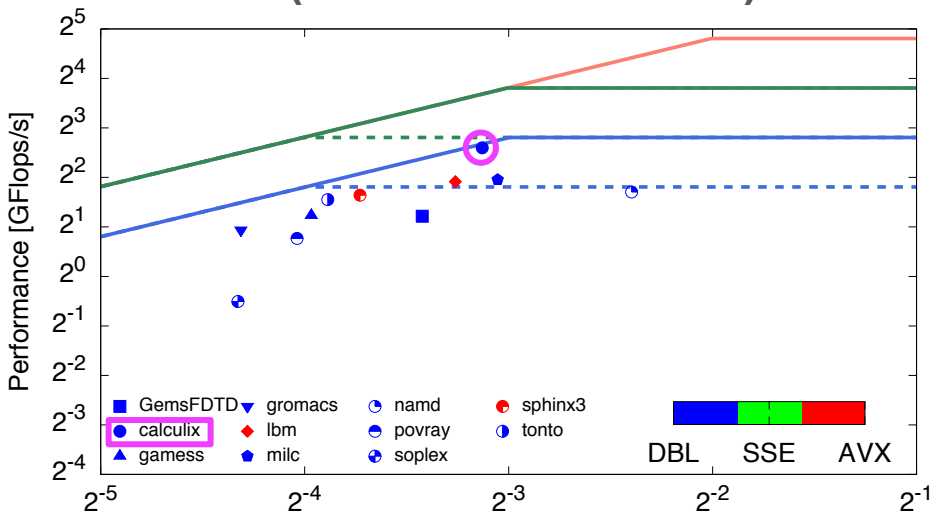
Experimental Results: SPEC2006 - Floating-point benchmarks -

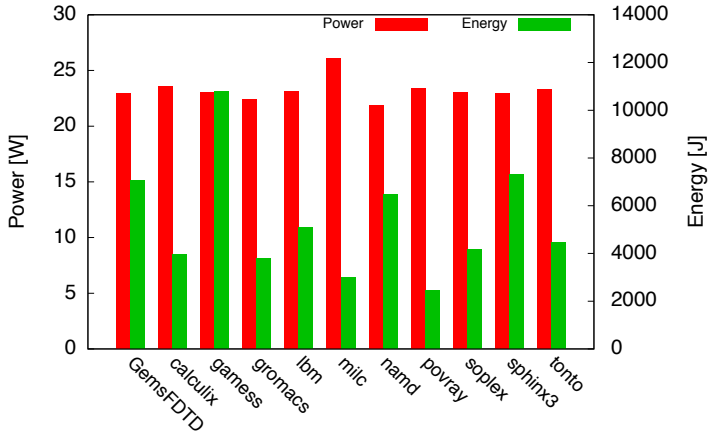
SPYMON (USER-SPACE MONITORING)





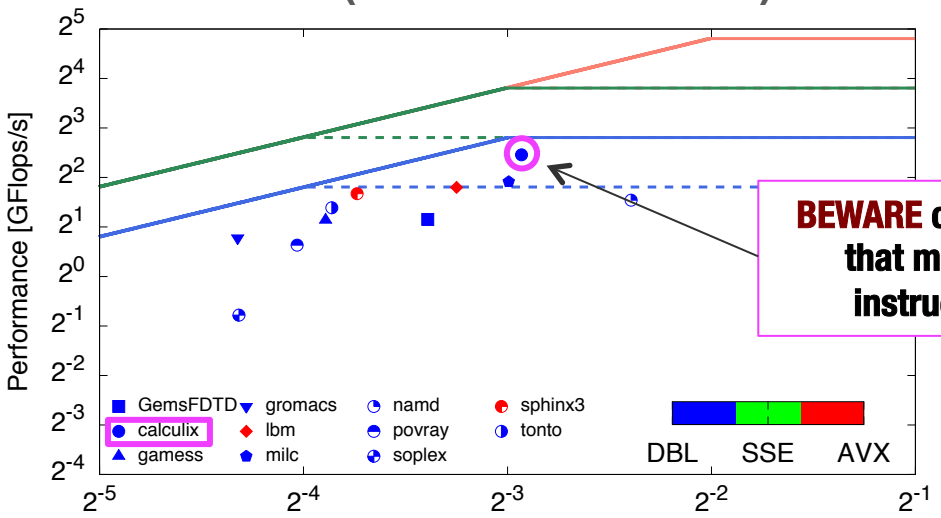
KERMON (KERNEL-SPACE MONITORING)

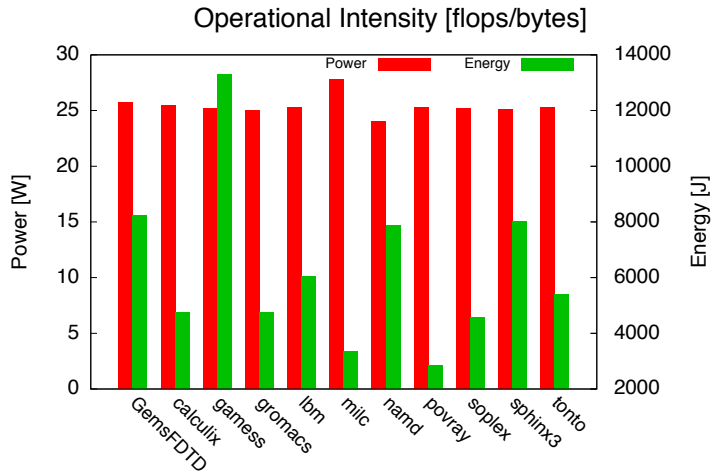




Experimental Results: SPEC2006 - Floating-point benchmarks -

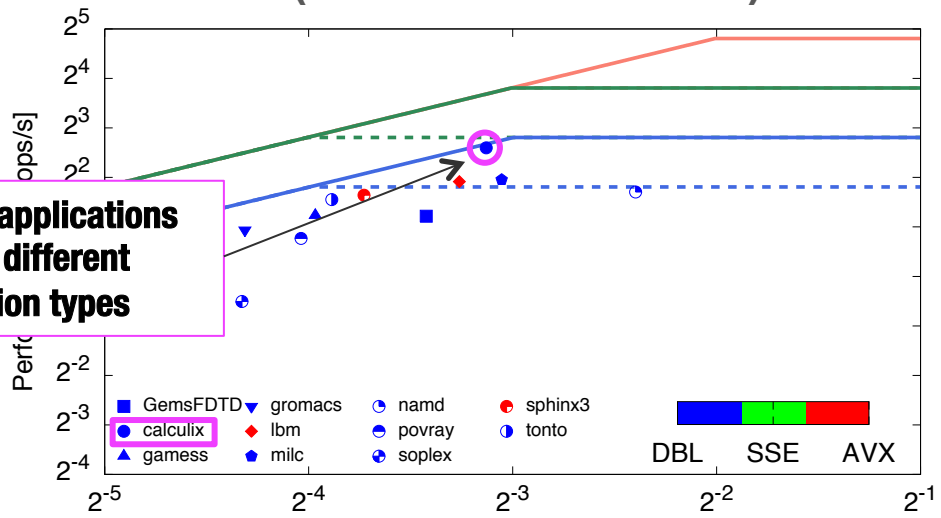
SPYMON (USER-SPACE MONITORING)

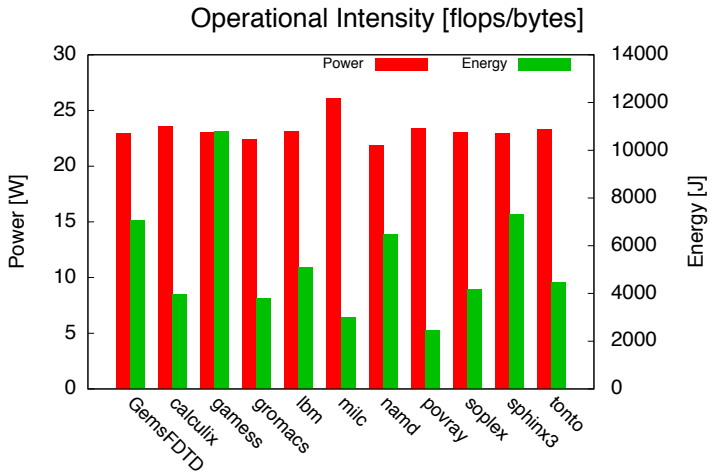




BEWARE of applications that mix different instruction types

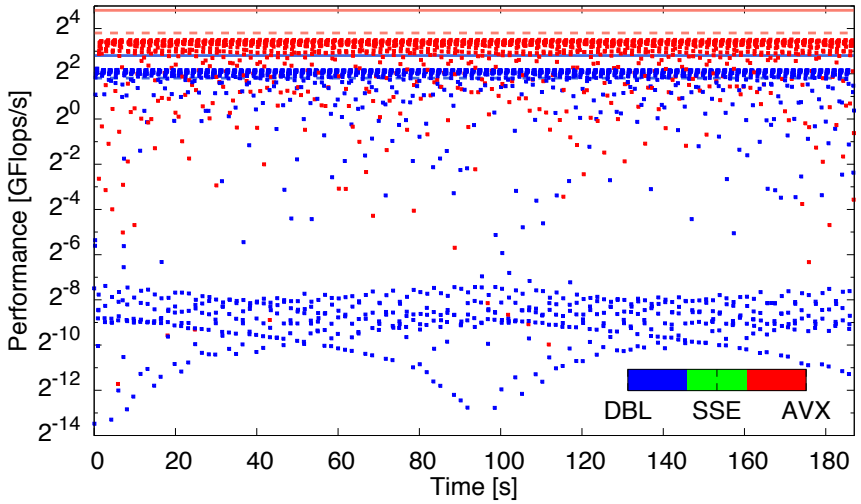
KERMON (KERNEL-SPACE MONITORING)





Experimental Results: SPEC06 Calculix - Application Monitoring -

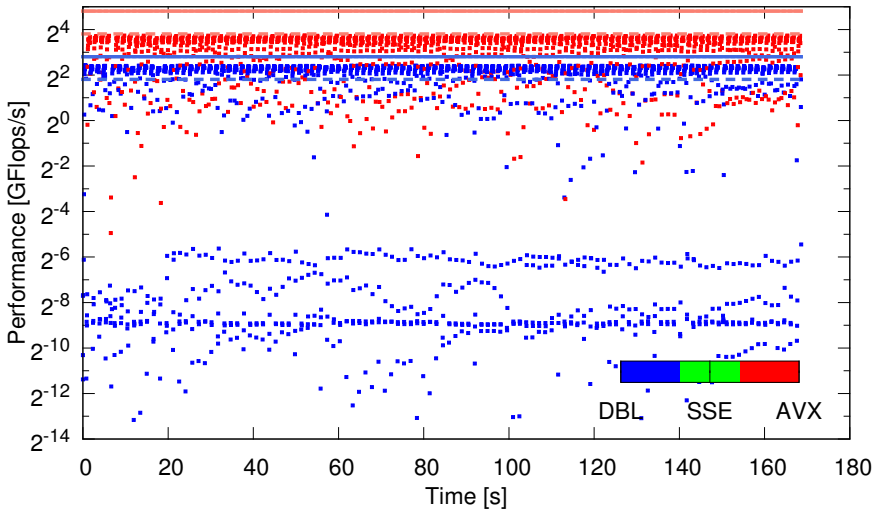
SPYMON (USER-SPACE MONITORING)



SPYMON configuration:

- Sampling interval: 50 ms
- 8 spy threads (2/core for HTT)

KERMON (KERNEL-SPACE MONITORING)



KERMON configuration:

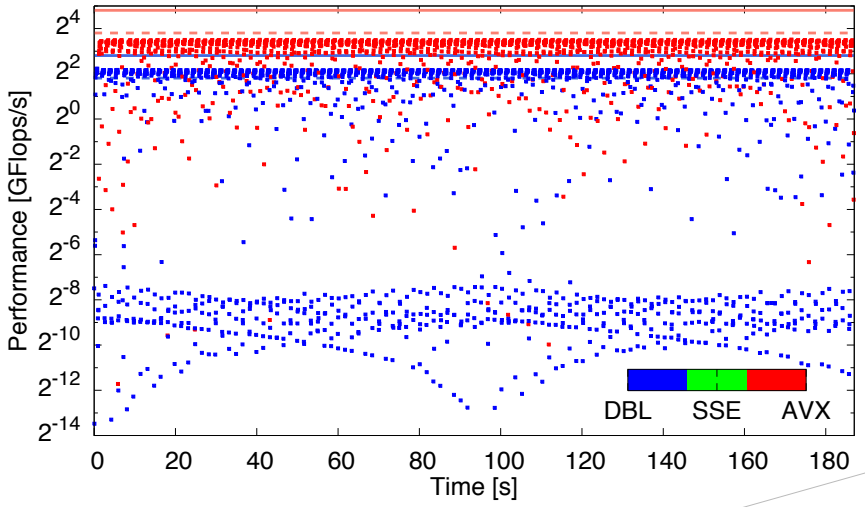
- Re-sampling interval: 50 ms
- Scheduling context of Linux 3.6.6 Kernel

i7 3770K Ivy Bridge	
Frequency	3.5 GHz
L1/L2/L3 size	32/256/8192 KB
DRAM to L3 (DRAM3)	2 channels (8B)
	2x933 MHz

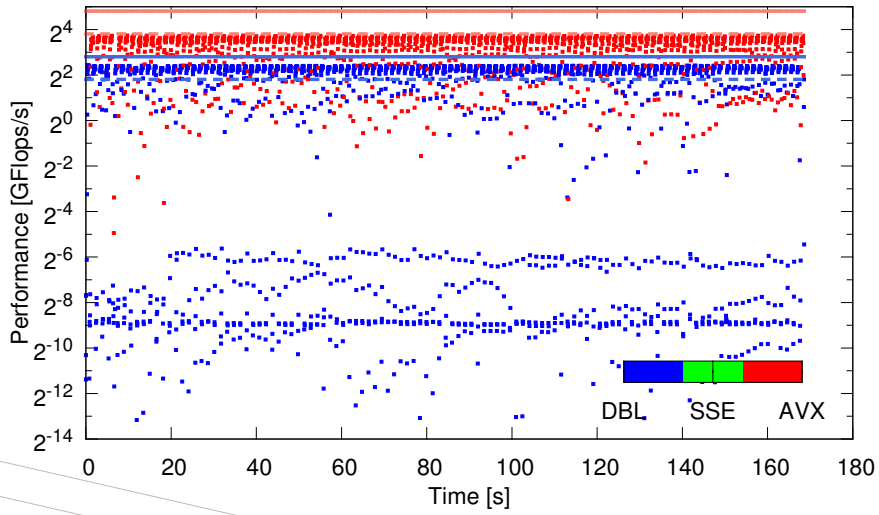
4 Event Sets	
Set 1	`RET_LDs, RET_STs, FP_x87, UOPS_RET`
Set 2	`FP_AVX_PD, FP_SSE_PD, FP_SSE_SD, UOPS_RET`
Set 3	`FP_AVX_PS, FP_SSE_PS, FP_SSE_SS, UOPS_RET`
Set 4	`L1D_REPL, L2_LINES_IN, OFF_CORE_REQ, UOPS_RET`

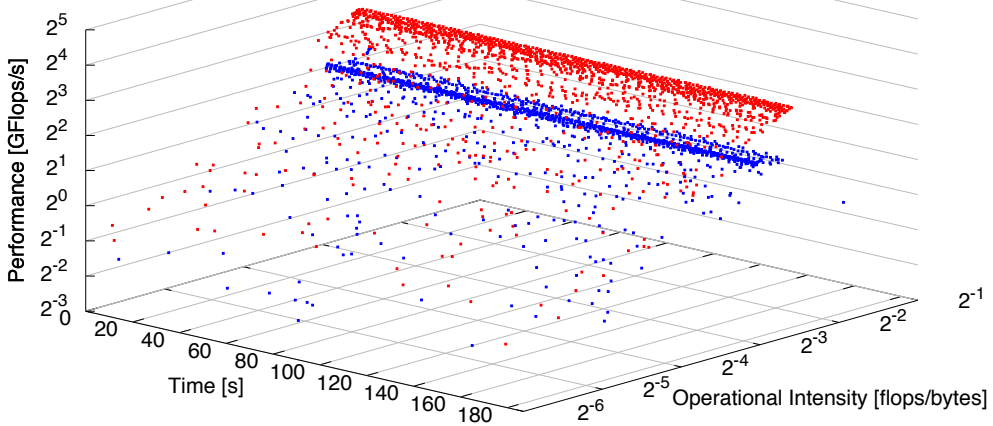
Experimental Results: SPEC06 Calculix - Application Monitoring -

SPYMON (USER-SPACE MONITORING)



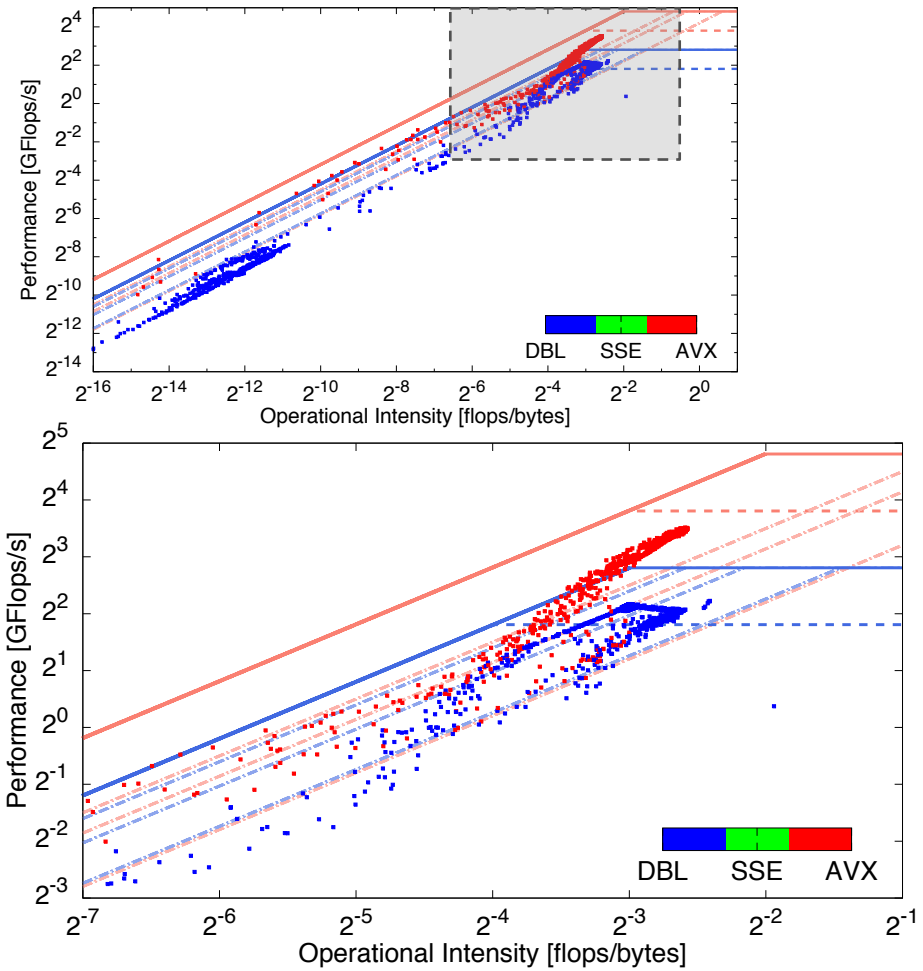
KERMON (KERNEL-SPACE MONITORING)



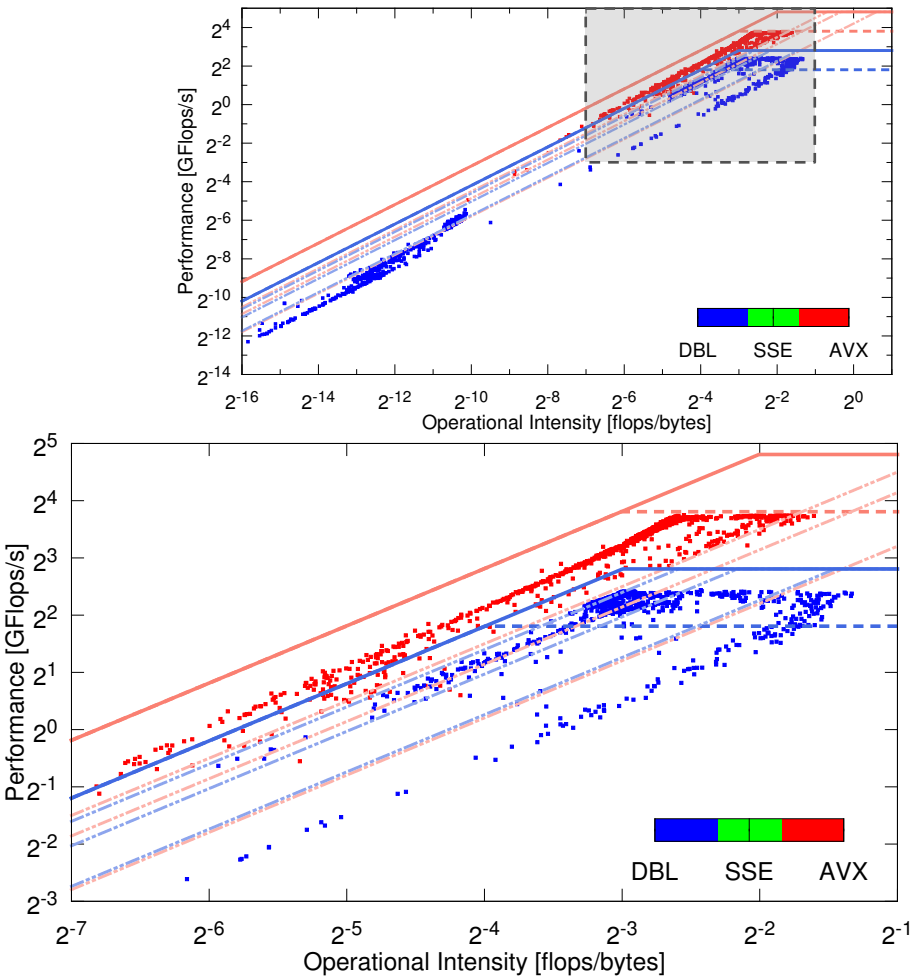


Experimental Results: SPEC06 Calculix
- Cache-aware Roofline Model -

SPYMON (USER-SPACE MONITORING)



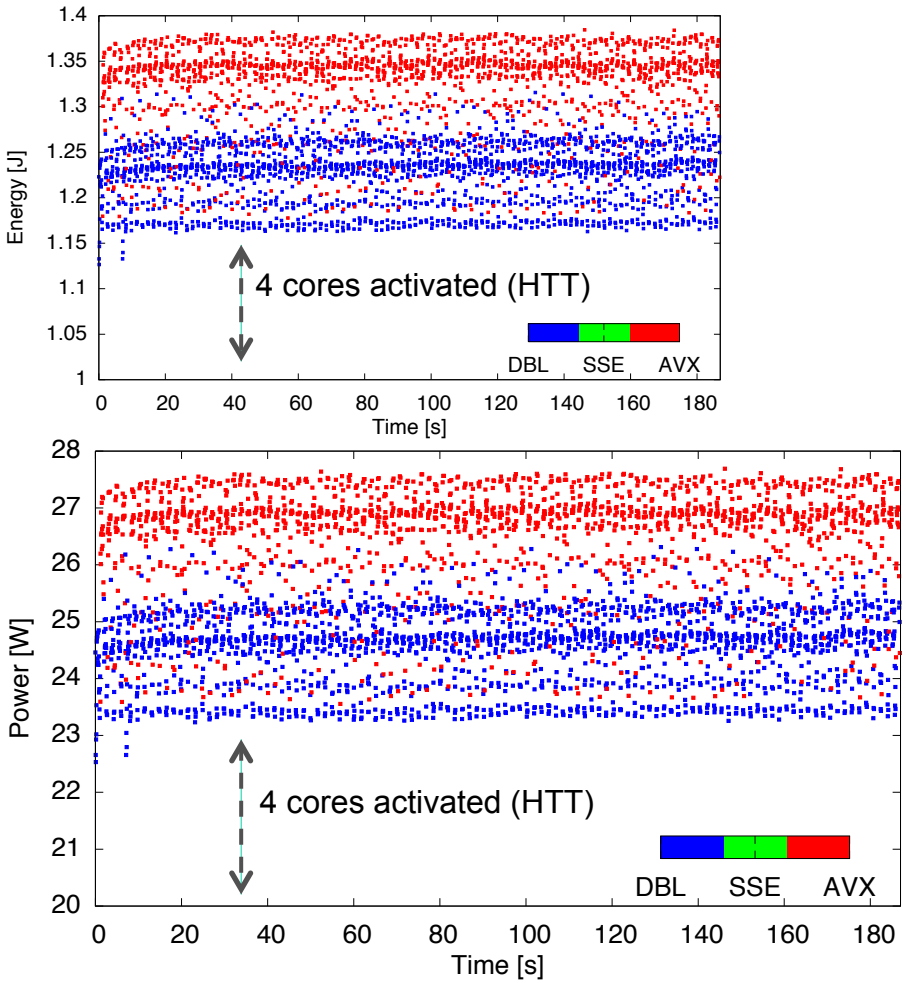
KERMON (KERNEL-SPACE MONITORING)



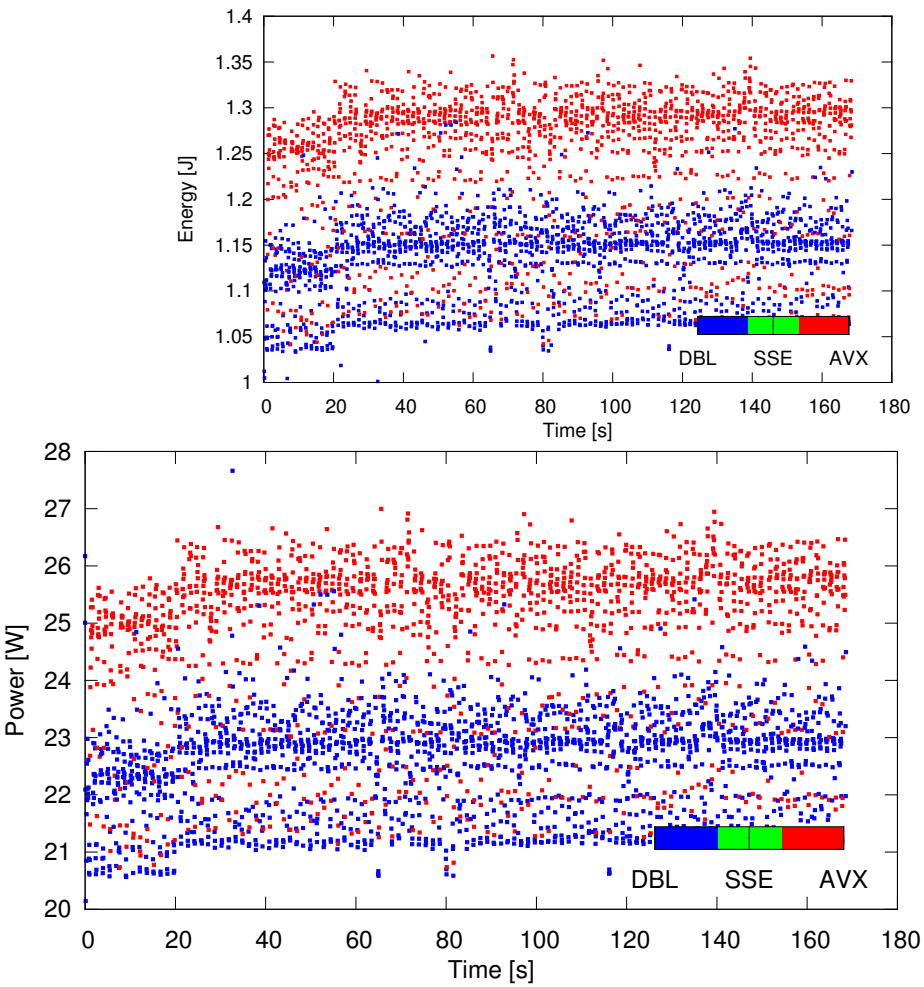
Experimental Results: SPEC06 Calculix - Power and Energy Consumption -

technology from seed

SPYMON (USER-SPACE MONITORING)

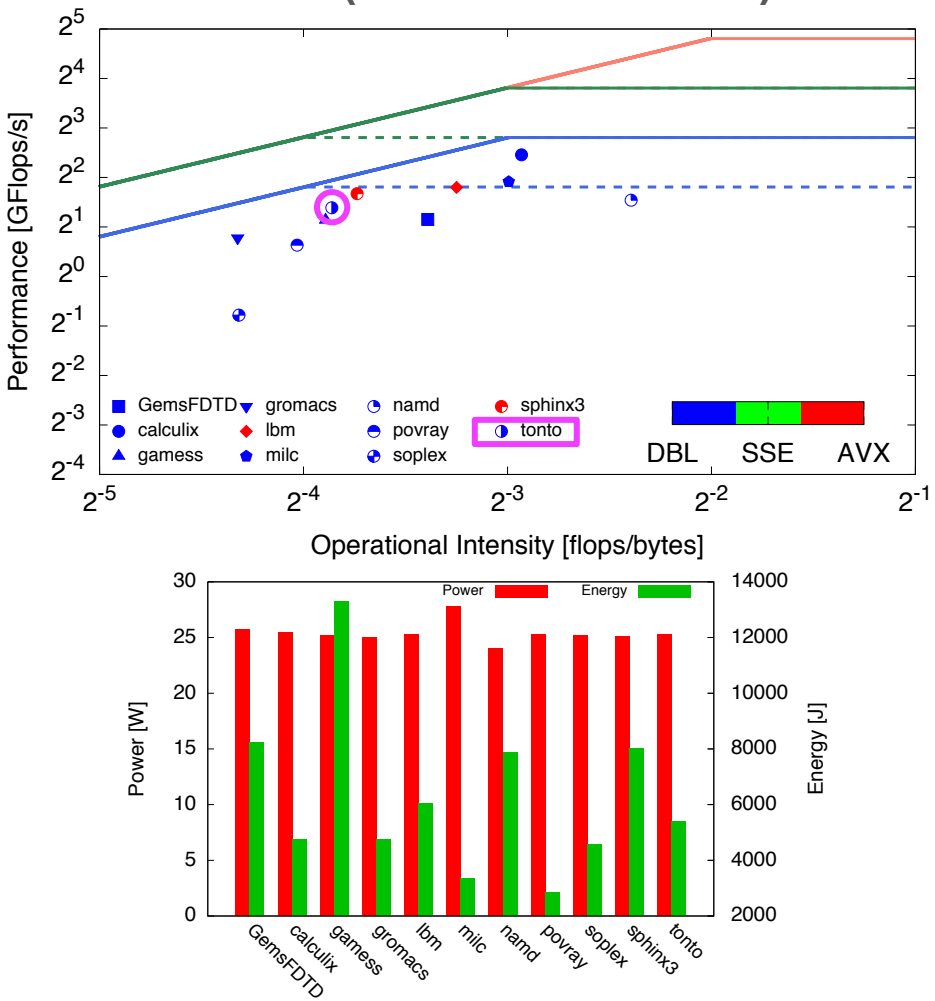


KERMON (KERNEL-SPACE MONITORING)

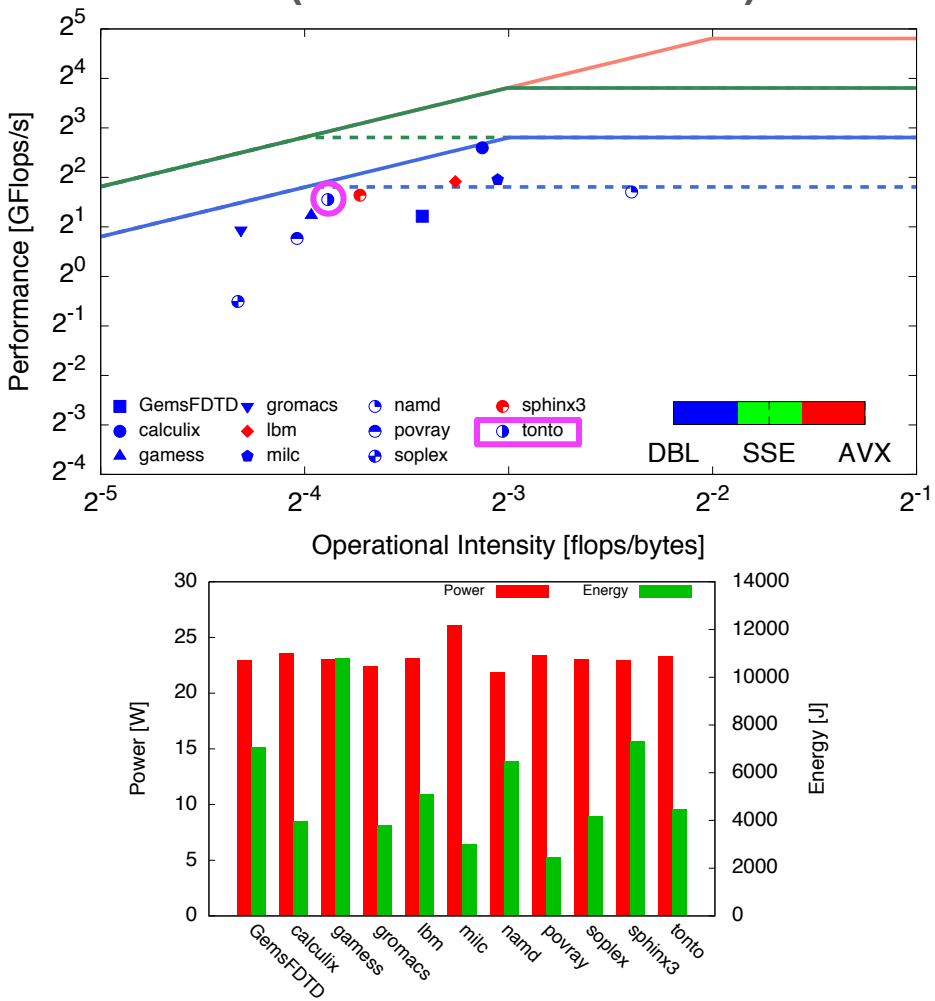


Experimental Results: SPEC2006 - Floating-point benchmarks -

SPYMON (USER-SPACE MONITORING)

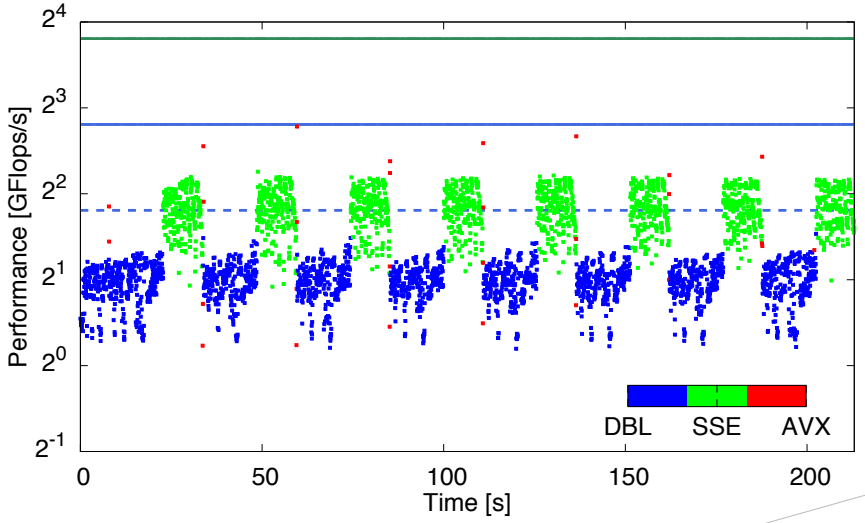


KERMON (KERNEL-SPACE MONITORING)

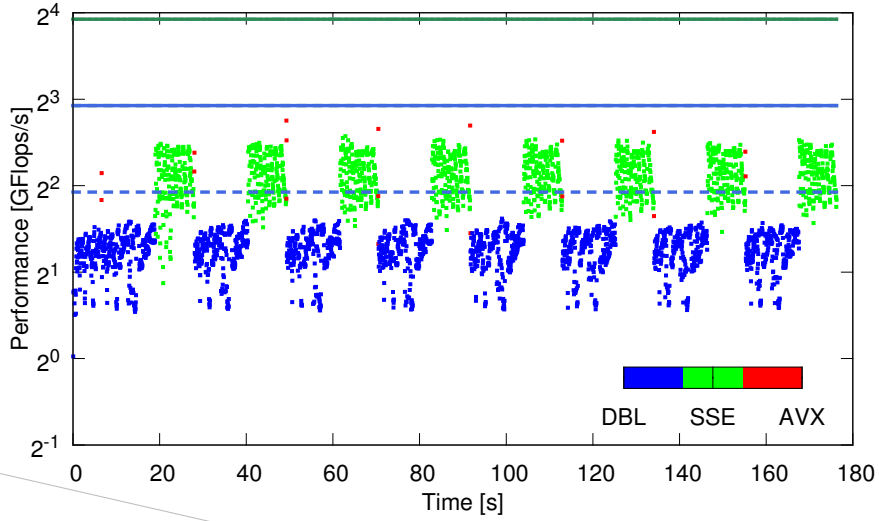


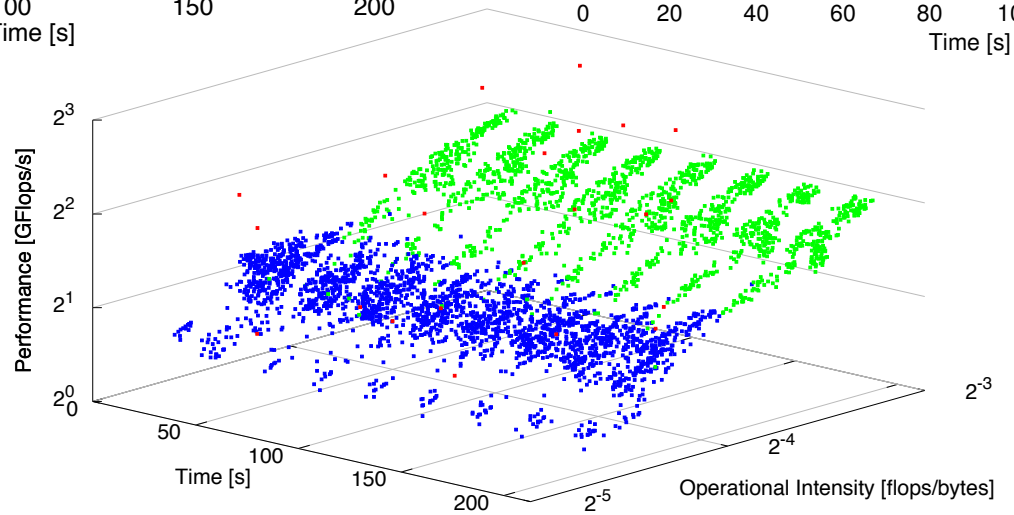
Experimental Results: SPEC06 Tonto - Application Monitoring -

SPYMON (USER-SPACE MONITORING)



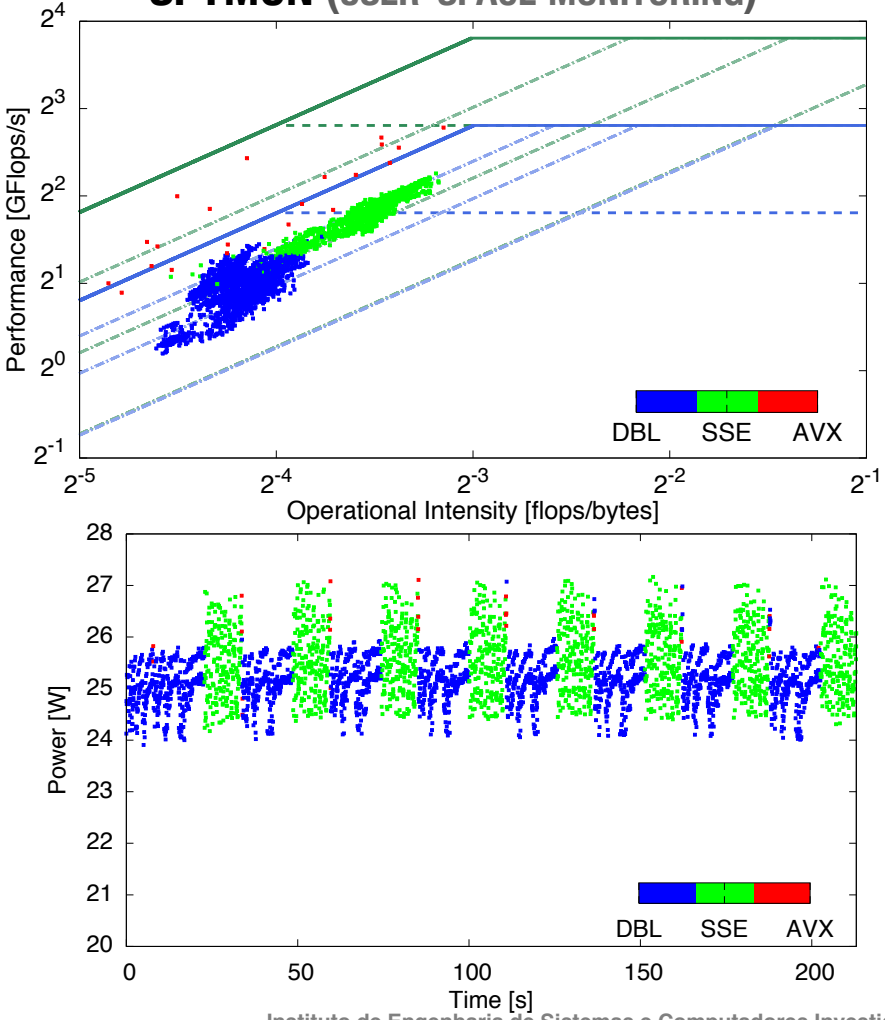
KERMON (KERNEL-SPACE MONITORING)



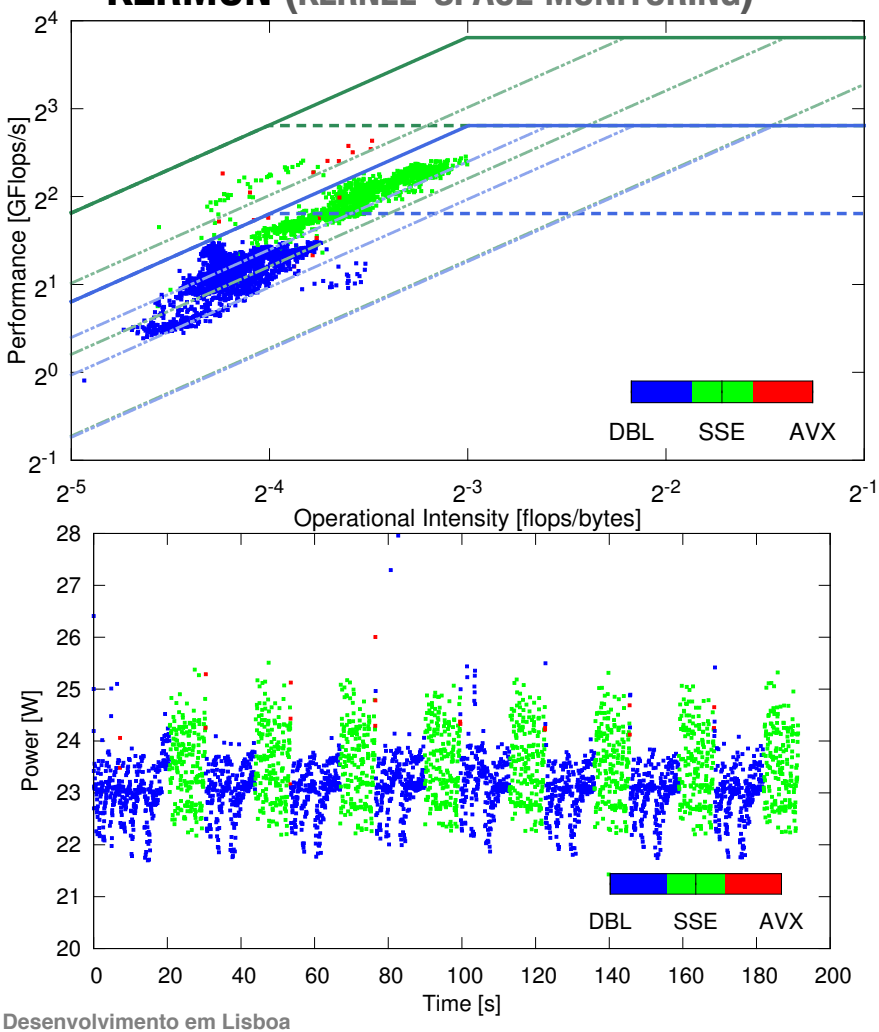


Experimental Results: SPEC06 Tonto - Model and Power Consumption -

SPYMON (USER-SPACE MONITORING)



KERMON (KERNEL-SPACE MONITORING)



Conclusions

- **KERMON: Kernel-space monitoring**
  - Higher monitoring accuracy due to task-based approach
  - Allows monitoring each individual thread spawn by a single application
  - More challenging to implement and requires patching the kernel

- **SPYMON: User-space monitoring**
  - Lightweight, easy-to-use and highly configurable approach
  - Integrated monitoring of the complete system in run-time
  - Less monitoring accuracy due to interference with existing OS tasks

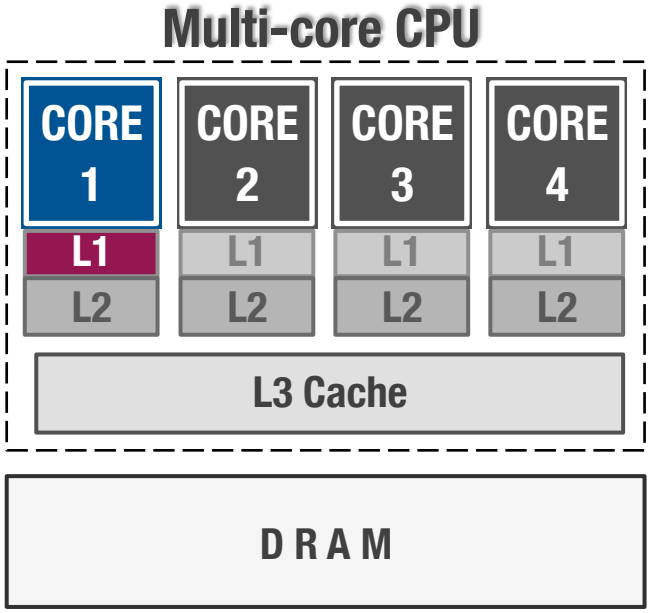
Conclusions

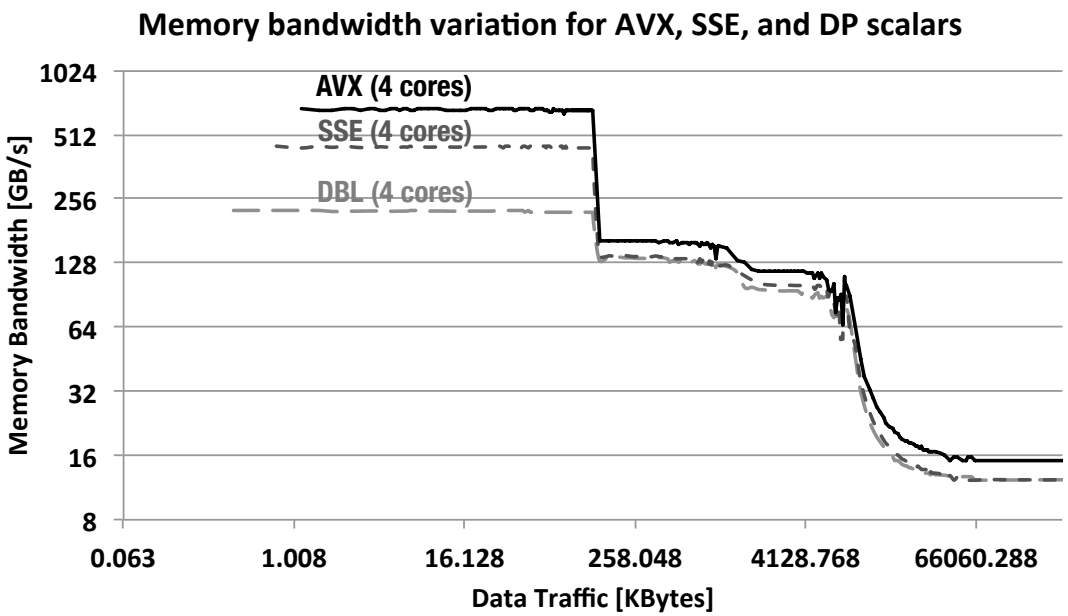
- **Cache-aware Roofline Model: Application Characterization**
  - Insightful modeling of modern multi-cores with complex memory hierarchy
  - When coupled **with monitoring tools** allows visualizing:
    - how close are the real applications to exploit the full potentials of the architecture
    - detect the main execution bottlenecks and
    - different execution stages

inesc id
lisboa
10 anos

Instituto de Engenharia de Sistemas e Computadores
Investigação e Desenvolvimento em Lisboa

Cache-aware Roofline Model
- Memory Hierarchy -

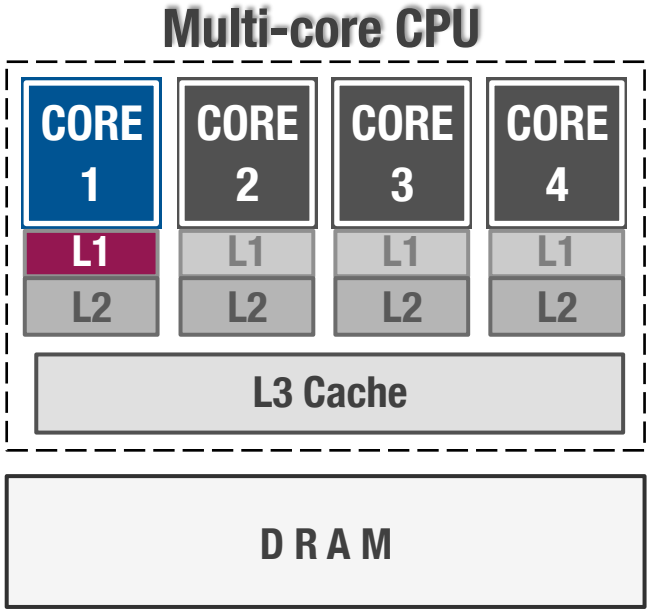


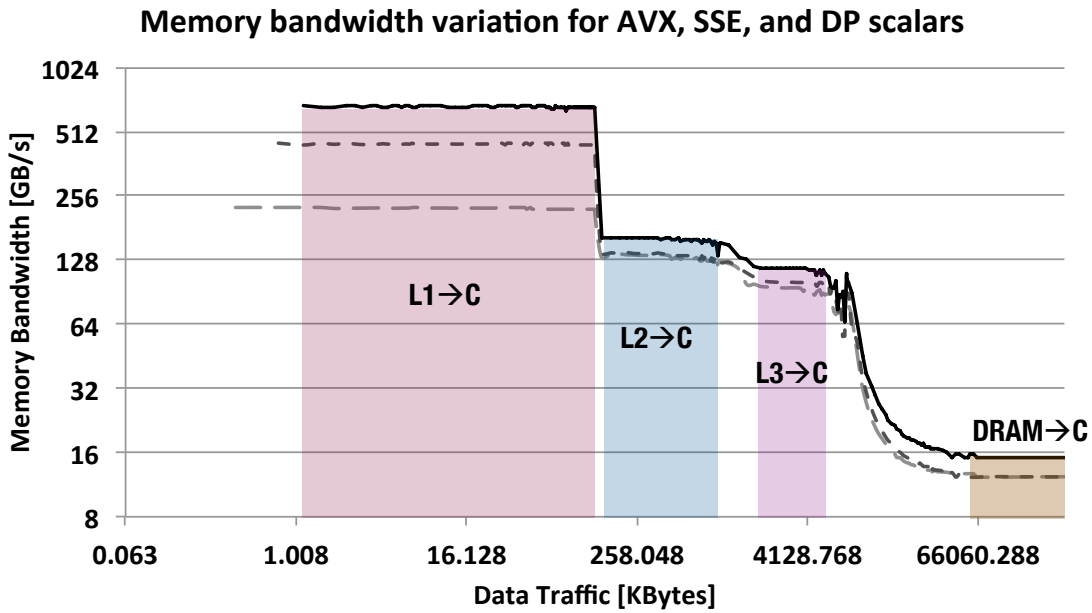


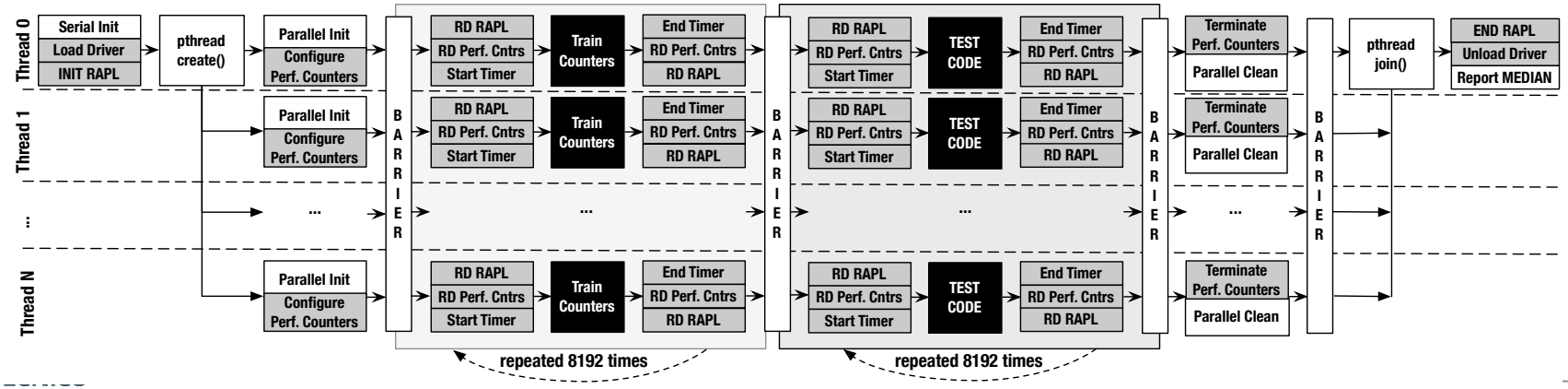
i7 3770K Ivy Bridge	Perf. [F_P] (GFlops/s)*	Bwidth L1→C [B_P] (GB/s)*
1 Core	28	168
4 Cores	112	672
*256-bit AVX double-precision floating-point instructions		

```
// AVX Assembly code: 2 Loads + 1 Store
vmovapd   0(%rax), %ymm0
vmovapd  32(%rax), %ymm1
vmovapd    %ymm2, 64(%rax)
vmovapd  96(%rax), %ymm3
vmovapd 128(%rax), %ymm4
vmovapd    %ymm5, 160(%rax)
...
```

Cache-aware Roofline Model
- Backup: Tool -

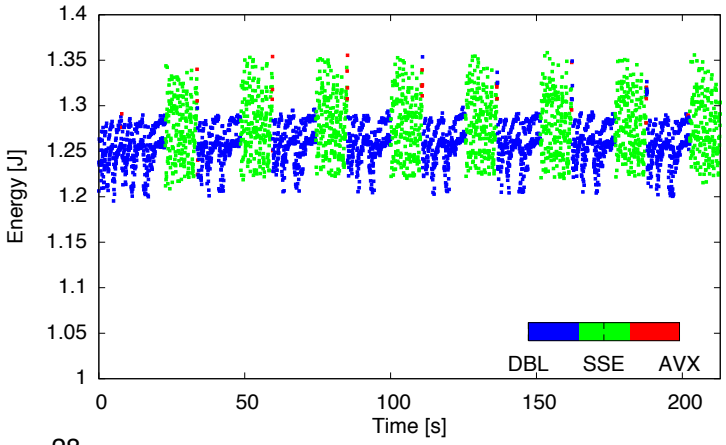


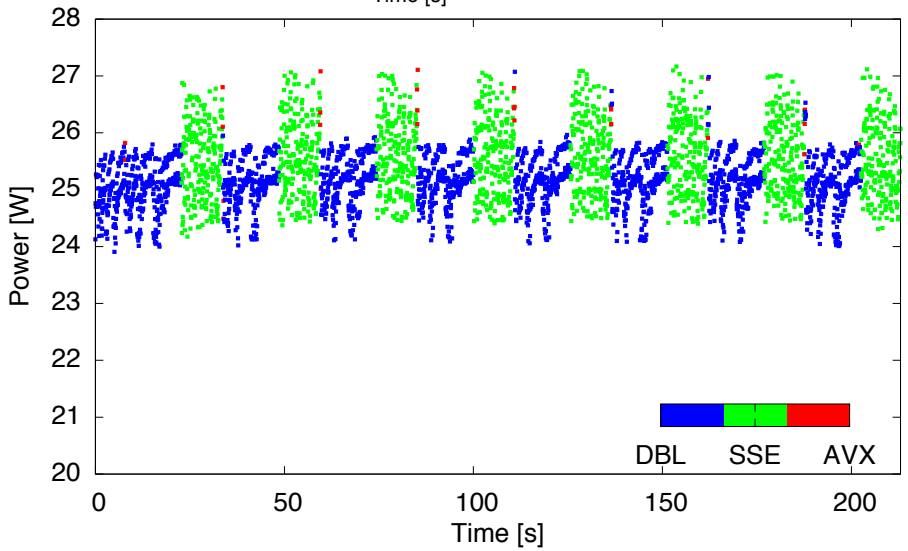




Experimental Results: SPEC06 Tonto - Power and Energy Consumption -

SPYMON (USER-SPACE MONITORING)





KERMON (KERNEL-SPACE MONITORING)