

# Základy práce se superpočítačem a měření výkonnosti aplikací

AVS - Architektury výpočetních systémů  
Cvičení 1, 2023/2024

**Jirka Jaroš**

Vysoké učení technické v Brně, Fakulta informačních technologií  
Božetěchova 1/2, 612 66 Brno - Královo Pole  
jarosjir@fit.vutbr.cz



# MĚŘENÍ VÝKONNOSTI S INTEL ADVISOR

- **Připojte se na Barboru**

```
ssh login@barbora.it4i.cz
```

- **Vaše PC: Připojte si disk z Barbory a nakopírujte tam obsah 1. cvičení**

```
mkdir /tmp/barbora  
sshfs login@barbora.it4i.cz: /tmp/barbora
```

- **Nastartujte job**

```
salloc -A DD-23-135 -p qcpu_exp -N 1 --comment 'use:vtune=2022.2.0' --x11
```

- nebo

```
salloc -A DD-23-135 -p qcpu -N 1 -t 1:0:0 --comment 'use:vtune=2022.2.0' --x11
```

- Pokud se nedočkáte ani za 5 minut, pracujte z loginu (dnes to nevadí)

## • Implementace

- sekvenční – 1 jádro (lab1.cpp)
- paralelní – 36 jader (lab1-par.cpp)

## • Součet dvou vektorů

```
for (size_t i = 0 ; i < size; i++)  
{  
    c[i] = a[i] + b[i];  
}
```

## • Skalární součin dvou vektorů

```
for (size_t i = 0 ; i < size; i++)  
{  
    res += a[i] * b[i];  
}
```

Velikost pole		Počet opakování
256B	(L1)	10M
4kB	(L1)	1M
96kB	(L2)	100k
2MB	(L3)	1k
16MB	(RAM)	100

## 1. Natáhněte modul kompilátoru a Advisoru

```
ml Advisor intel/2023a
```

## 2. Přeložte zdrojové soubory laboratorního cvičení

```
icpx -O3 -g lab1.cpp -o lab1
```

## 3. Otevřete si Intel Advisor

```
advixe-gui
```

## 4. Vytvořte nový projekt

- Zadejte název a cestu kam chcete uložit Advisor projekt
- Do pole Application zadejte binárku lab1

# Advisor – základní analýza (Survey analysis)

Kód přeložen pro SSE2 - default

Higher instruction set architecture (ISA) available  
Consider recompiling your application using a higher ISA.

Analysis Work...  
Vectorization and...  
Accuracy: Low Medi. High Cust...  
Overhead: ...  
Survey  
Characterizati...  
Memory Access ...  
Dependencies

Function Call Sites and Loops	Performance Issues	CPU Time	Type	Why No Vectorization?	Vectorized Loops	Instruction Set Analysis	Advanced
		Total Time	Self Time		Vec... Efficiency	Gai... VL ...	Traits
[Loop in vectorAdd<unsigned long>24576...		0.470s	0.470s	Vectorized (...)	SSE ~100%	5.32x 4	Float32 Unrolled by 4
[Loop in vectorDot<unsigned long>24576...	1 Potential...	0.430s	0.430s	Vectorized (...)	SSE ~100%	3.99x 4	Float32 Unrolled by 4
[Loop in vectorAdd<unsigned long>41943...		0.300s	0.300s	Vectorized (...)	SSE ~100%	5.34x 4	Float32
[Loop in vectorAdd<unsigned long>52428...		0.240s	0.240s	Vectorized (...)	SSE ~100%	5.34x 4	Float32
[Loop in vectorDot<unsigned long>41943...	1 Potential...	0.180s	0.180s	Vectorized (...)	SSE ~100%	4.00x 4	Float32 Unrolled by 4
[Loop in vectorDot<unsigned long>52428...	1 Potential...	0.160s	0.160s	Vectorized (...)	SSE ~100%	4.00x 4	Float32 Unrolled by 4
[Loop in vectorAdd<unsigned long>1024...		0.120s	0.120s	Vectorized (...)	SSE ~100%	4.95x 4	Float32 Unrolled by 4
[std::map<unsigned long, std::function<...>...		0.090s	0.090s	Function			Float...
[Loop in vectorDot<unsigned long>1024...	1 Potential...	0.080s	0.080s	Vectorized (...)	SSE ~94%	3.74x 4	Float32 Unrolled by 4
[Loop in vectorAdd<unsigned long>64...		0.070s	0.070s	Vectorized (...)	SSE ~84%	3.35x 4	Float32 Unrolled by 8
[vectorDot<unsigned long>64...		0.060s	0.060s	Function			Float32
[Loop in main at lab1.cpp:196]		1.020s	0.040s	Scalar			Shuffles
[Loop in generateData at lab1.cpp:88]	2 Data type...	0.040s	0.020s	Scalar			Type Conversions
[Loop in generateData at lab1.cpp:88]	2 Data type...	0.040s	0.020s	Scalar			Type Conversions
[Loop in vectorDot<unsigned long>100164...	1 Potential...	0.070s	0.070s	Vectorized (...)	SSE ~65%	2.66x 4	Float32 Unrolled by 8

Source Top Down Code Analytics Assembly Recommendations Why No Vectorization?

File: lab1.cpp:113 vectorAdd<unsigned long>24576>

Source	Total Time	%	Loop/Function Time	%	Traits
113 for (size_t i = 0; i < size; i++) [Loop in vectorAdd<unsigned long>64> at lab1.cpp:113] Vectorized SSE loop processes Float32 data type(s) Multiversioned for data dependence, ver 1; loop was unrolled by 8; loop with unsigned induction variable [Loop in vectorAdd<unsigned long>1024> at lab1.cpp:113] Vectorized SSE loop processes Float32 data type(s) Multiversioned for data dependence, ver 1; loop was unrolled by 4; loop with unsigned induction variable	469.995ms				

```
icpx -O3 -xhost -qopt-zmm-usage=high -g lab1.cpp -o lab1
icpx -O3 -xhost -qopt-zmm-usage=high -qopenmp -g lab1-par.cpp -o lab1-par
```

## 1. Flat profile

- Seznam funkcí seřazených podle délky výpočtu
- Využití AVX jednotek
- Performance issues – kde to drhne
- Analýza instrukční sady

Elapsed time: 1.48s

Vectorized

Not Vectorized

FILTER: All Modules All Sources Loops And Functions All Threads

Customize View

INTEL ADVISOR 2019

Summary Survey & Roofline Refinement Reports

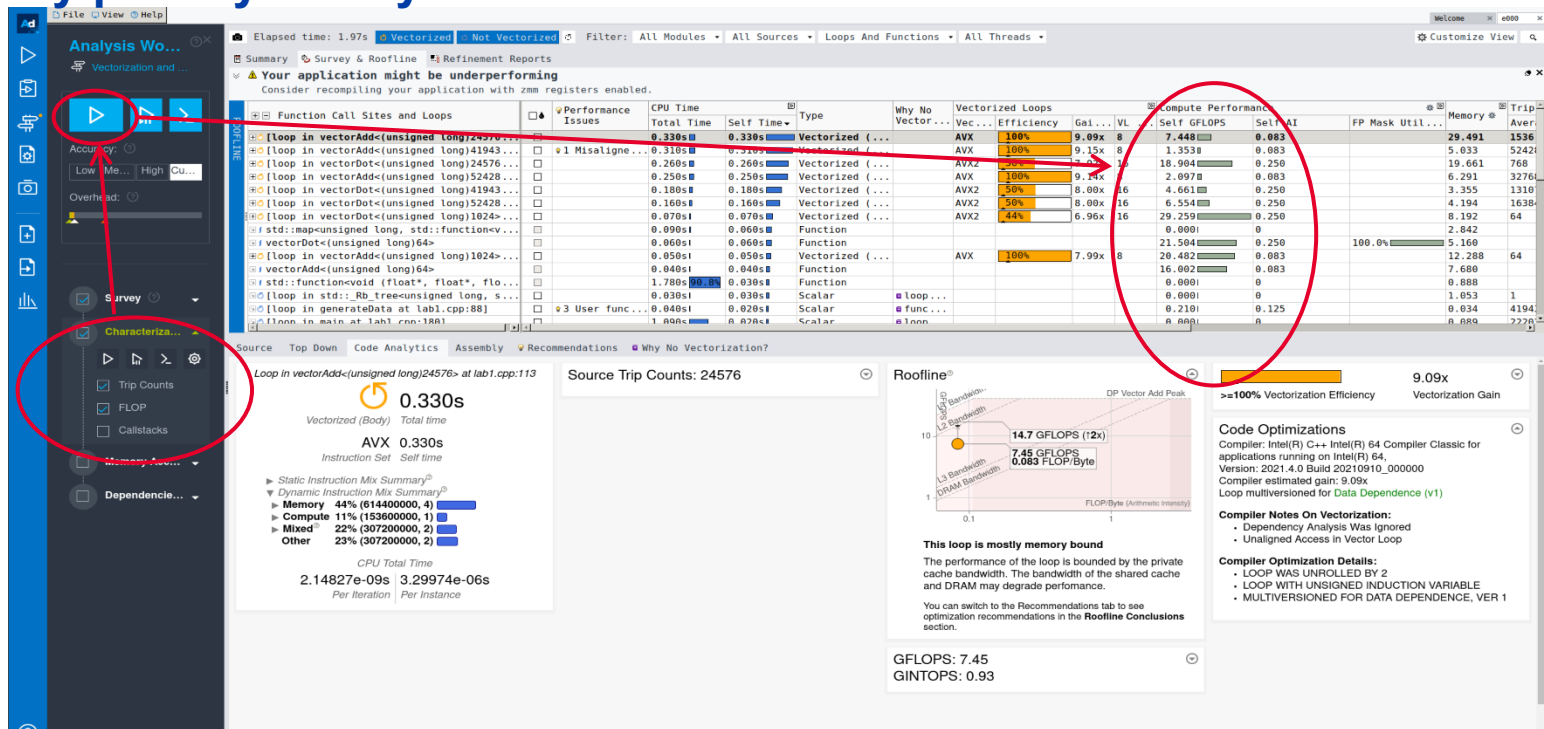
Function Call Sites and Loops	Performance Issues	CPU Time		Type	Why No Vectorization?	Vectorized Loops				Instruction Set Analysis		Adv...
		Total Time	Self Time			Vecto ...	Efficiency	Gain ...	VL (V ...	Traits	Data Typ ...	
[loop in vectorAdd<(unsigned long)4194304> at lab1.cpp]	1 Misaligned I...	0.380s	0.380s	Vectorized (Body)		AVX	~100%	9.15x	8		Float32	
[loop in vectorAdd<(unsigned long)8192> at lab1.cpp]		0.180s	0.180s	Vectorized (Body)		AVX	~100%	8.84x	8		Float32	Unroll
[loop in vectorDot<(unsigned long)4194304> at lab1.cpp]		0.150s	0.150s	Vectorized (Body)		AVX2	~50%	8.00x	16	FMA	Float32	Unroll
[loop in vectorDot<(unsigned long)1024> at lab1.cpp]		0.100s	0.100s	Vectorized (Body)		AVX2	~44%	6.96x	16	FMA	Float32	
[loop in vectorDot<(unsigned long)8192> at lab1.cpp]		0.100s	0.100s	Vectorized (Body)		AVX2	~48%	7.75x	16	FMA	Float32	Unroll
f std::map<unsigned long, std::function<void (float*, float*)>> at lab1.cpp]		0.130s	0.070s	Function								
[loop in vectorAdd<(unsigned long)131072> at lab1.cpp]		0.060s	0.060s	Vectorized (Body)		AVX	~100%	9.12x	8		Float32	Unroll
[loop in vectorAdd<(unsigned long)1024> at lab1.cpp]		0.060s	0.060s	Vectorized (Body)		AVX	~50%	7.99x	16		Float32	
f vectorDot<(unsigned long)64> at lab1.cpp]		0.050s	0.050s	Function						Extracts; FMA; Shuffles	Float32; ...	
[loop in main at lab1.cpp:195]		0.550s	0.050s	Scalar	loop control variable wa...							
[loop in std::Rb_tree<unsigned long, std::pair<unsigned long, unsigned long>, std::less<unsigned long>, std::pair<unsigned long, unsigned long>>::operator<> at lab1.cpp]		0.050s	0.040s	Scalar	loop control variable wa...							Unroll
[loop in vectorDot<(unsigned long)131072> at lab1.cpp]		0.030s	0.030s	Vectorized (Body)		AVX2	~50%	7.98x	16	FMA	Float32	Unroll
f vectorAdd<(unsigned long)64> at lab1.cpp]		0.030s	0.030s	Function							Float32	
f std::Function_handler<void (float*, float*, float*), void (float*, float*, float*)> at lab1.cpp]		0.020s	0.020s	Function								
[loop in main at lab1.cpp:179]		0.840s	0.020s	Scalar	loop control variable wa...							
[loop in generateData at lab1.cpp:87]	3 Data type con...	0.040s	0.010s	Scalar	function call cannot be ...					Type Conversions	Float32; ...	

- Výčet optimalizací, které kompilátor provedl
- Doba trvání jednotlivých funkcí
- A spoustu věcí na dalších záložkách

Source   Top Down   Code Analytics   Assembly   Recommendations   Why No Vectorization?					
File: lab1.cpp:112 vectorAdd<(unsigned long)4194304>					
Line	Source	Total Time	%	Loop/Function Time	Traits
99					
100	* Vector addition				
101	* @param c - Output array				
102	* @param a - Input array A				
103	* @param b - Input array B				
104	* @param size - Size of the arrays				
105					
106	*/				
107	template<size_t size>				
108	void vectorAdd(float* c,				
109	float* a,				
110	float* b)				
111	{				
112	for (size_t i = 0 ; i < size; i++)	39.995ms		380.014ms	
	[loop in vectorAdd<(unsigned long)1024> at lab1.cpp:112]				
	Vectorized AVX loop processes Float32 data type(s)				
	Multiversioned for data dependence, ver 1; loop with unsigned induction variable				
	[loop in vectorAdd<(unsigned long)8192> at lab1.cpp:112]				
	Vectorized AVX loop processes Float32 data type(s)				
	Multiversioned for data dependence, ver 1; loop was unrolled by 4; loop with unsigned induction variable				
	[loop in vectorAdd<(unsigned long)131072> at lab1.cpp:112]				
	Vectorized AVX loop processes Float32 data type(s)				
	Multiversioned for data dependence, ver 1; loop was unrolled by 4; loop with unsigned induction variable				
	[loop in vectorAdd<(unsigned long)4194304> at lab1.cpp:112]				
	Vectorized AVX loop processes Float32 data type(s)				
	Multiversioned for data dependence, ver 1; loop with unsigned induction variable				
Selected (Total Time):		39.995ms			



- Jak vypadá instrukční mix (AVX – pro add, AVX2 pro dotProduct)
- Jak využívám vektorové jednotky (backend) + FLOPS
- Rady pro zvýšení výkonu



- **Lokálně, na loginu nebo na výpočetním uzlu**

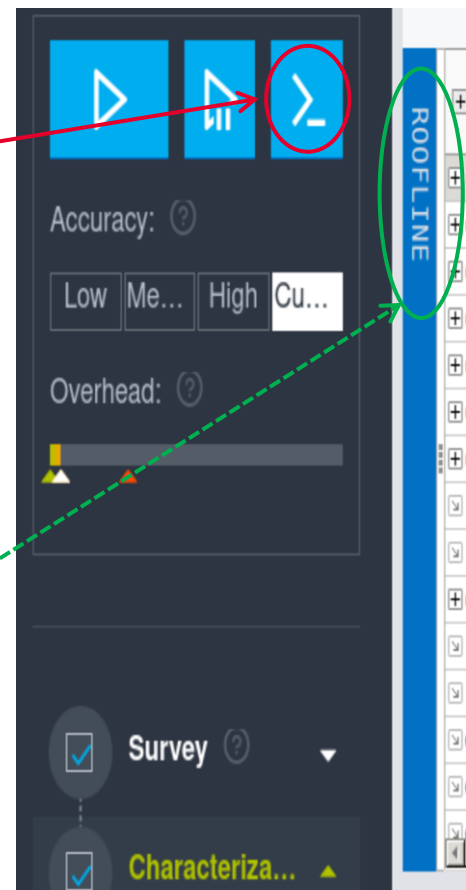
- Všechny intel tooly umožňují, udělat měření dávkově z commandline a pak si je prohlédnout na jiném stroji.

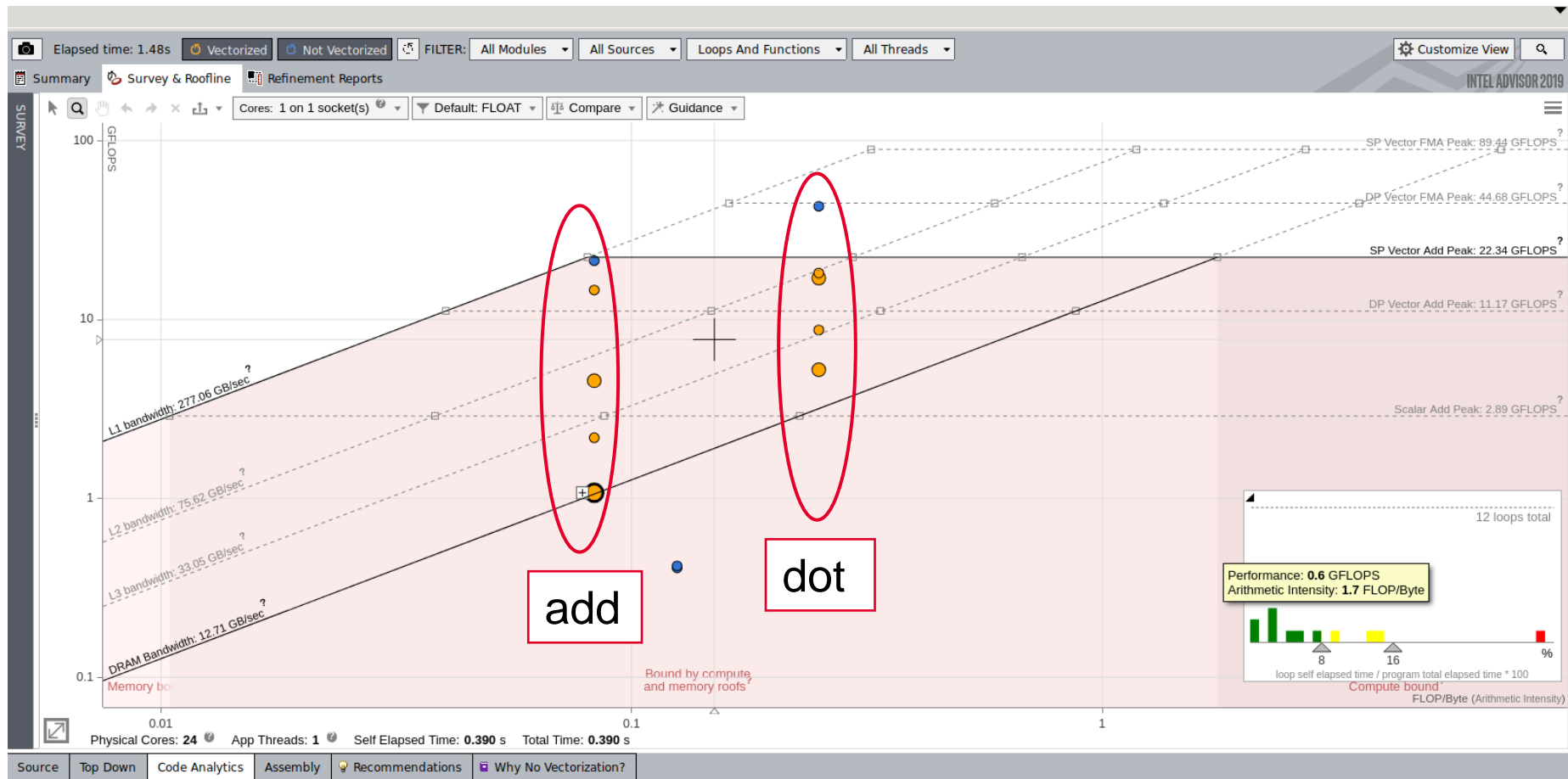
- **Generování dat na uzlu**

- Zkopírujeme si
- Zavřeme Advisor
- Spustíme příkaz v command line
  - Pozor, jsou to dva příkazy, které musíte vykonat (2x zmáčknout enter)

- **Zobrazení**

- Otevřete nový terminál na barboře
- Natáhněte moduly, otevřete Advisor znovu otevřete projekt
- Natáhněte si data – Místo new project, **Open Results**
- Prohlédněte si výsledky roofline analýzy





- Přidejte kód pro součet dvou vektorů následující kvadratickou formuli:

```
for (size_t i = 0 ; i < size; i++)  
{  
    c[i] = a[i] * a[i] - a[i] * b[i] + b[i] * b[i];  
}
```

- Jak se změnila aritmetická intenzita oproti sčítání dvou vektorů?
- Jak se změnil dosažený výkon?
- Kolik procent z teoretického maxima jste dosáhli?

- Přidejte kód pro výpočet sumy prefixů

```
for (size_t i = 1 ; i < size; i++)  
{  
    a[i] = a[i] + a[i-1];  
}
```

- Jak se změnila aritmetická intenzita oproti sčítání dvou vektorů?
- Proč dosahujete nižších výkonů než při sčítání dvou vektorů?

# MĚŘENÍ VÝKONNOSTI S INTEL VTUNE

## 1. Natáhněte modul kompilátoru a VTune

```
ml Vtune intel/2023a
```

## 2. Přeložte zdrojové soubory laboratorního cvičení

```
icpx -O3 -xhost -g lab1.cpp -o lab1  
icpx -O3 -xhost -qopenmp -g lab1-par.cpp -o lab1-par
```

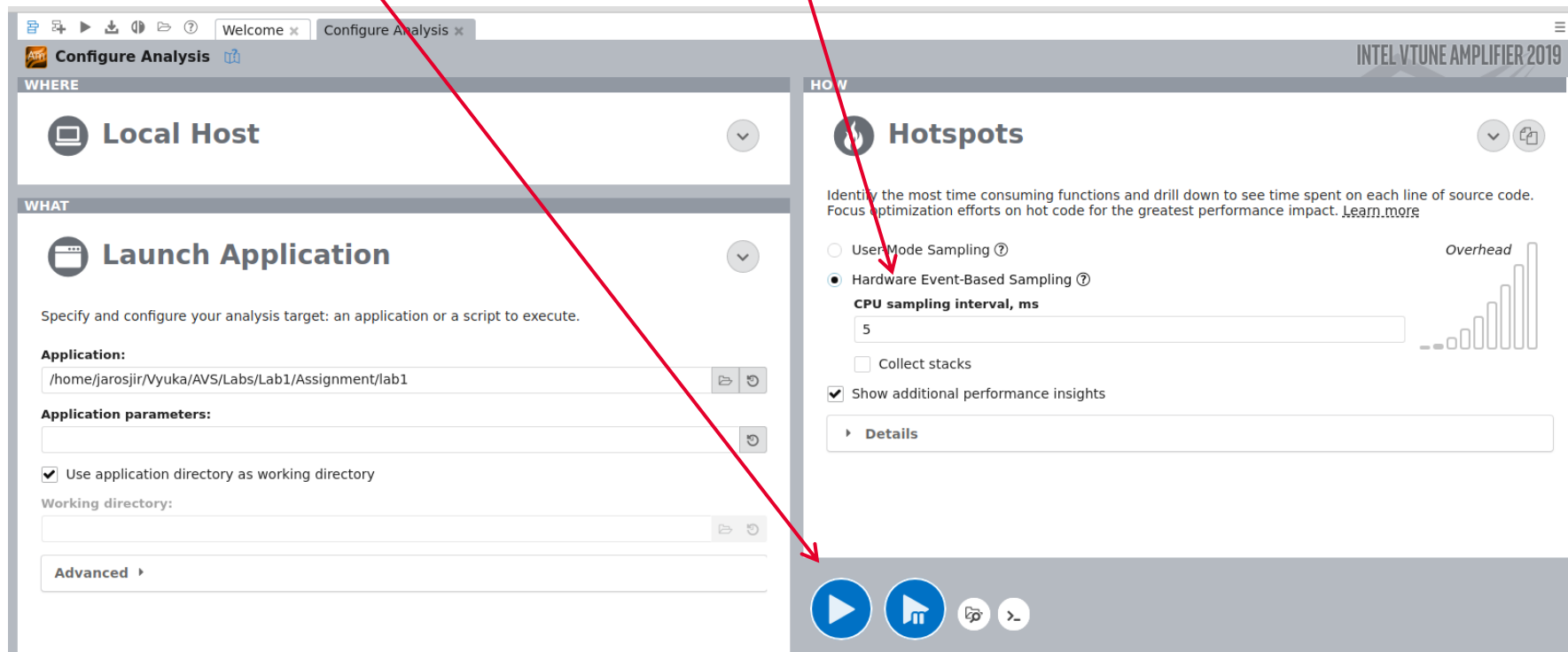
## 3. Otevřete si Intel Vtune

```
vtune-gui
```

## 4. Vytvořte nový projekt

- Zadejte název a cestu kam chcete uložit VTune projekt
- Do pole Application zadejte binárku lab1-par

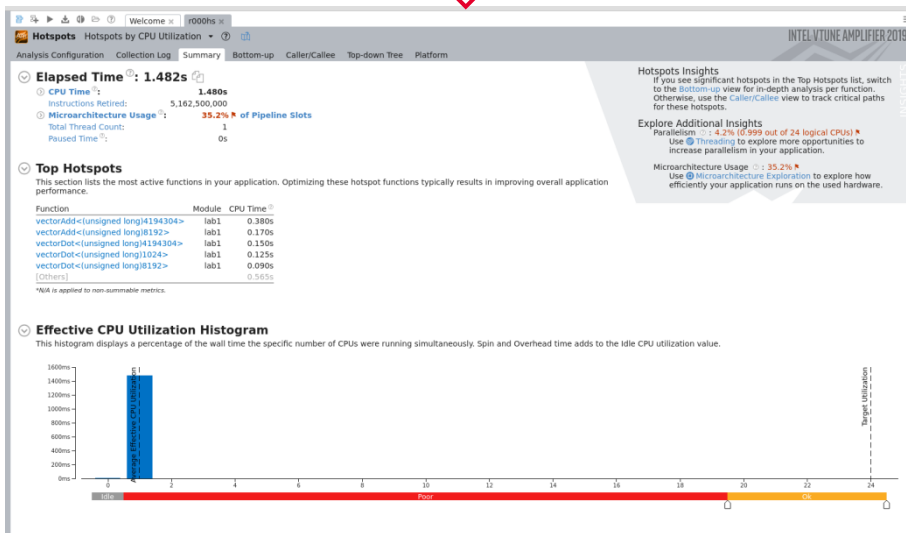
- Vyberte Hardware Event-Based Sampling
- Spustěte analýzu



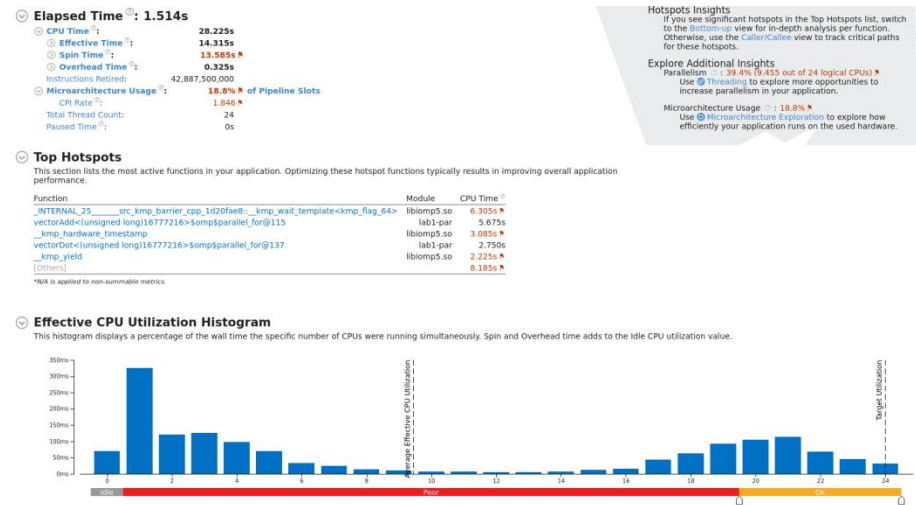
- Základní analýza ukazuje

- Nejtěžší smyčky
- Histogram vytížení vláken

lab1

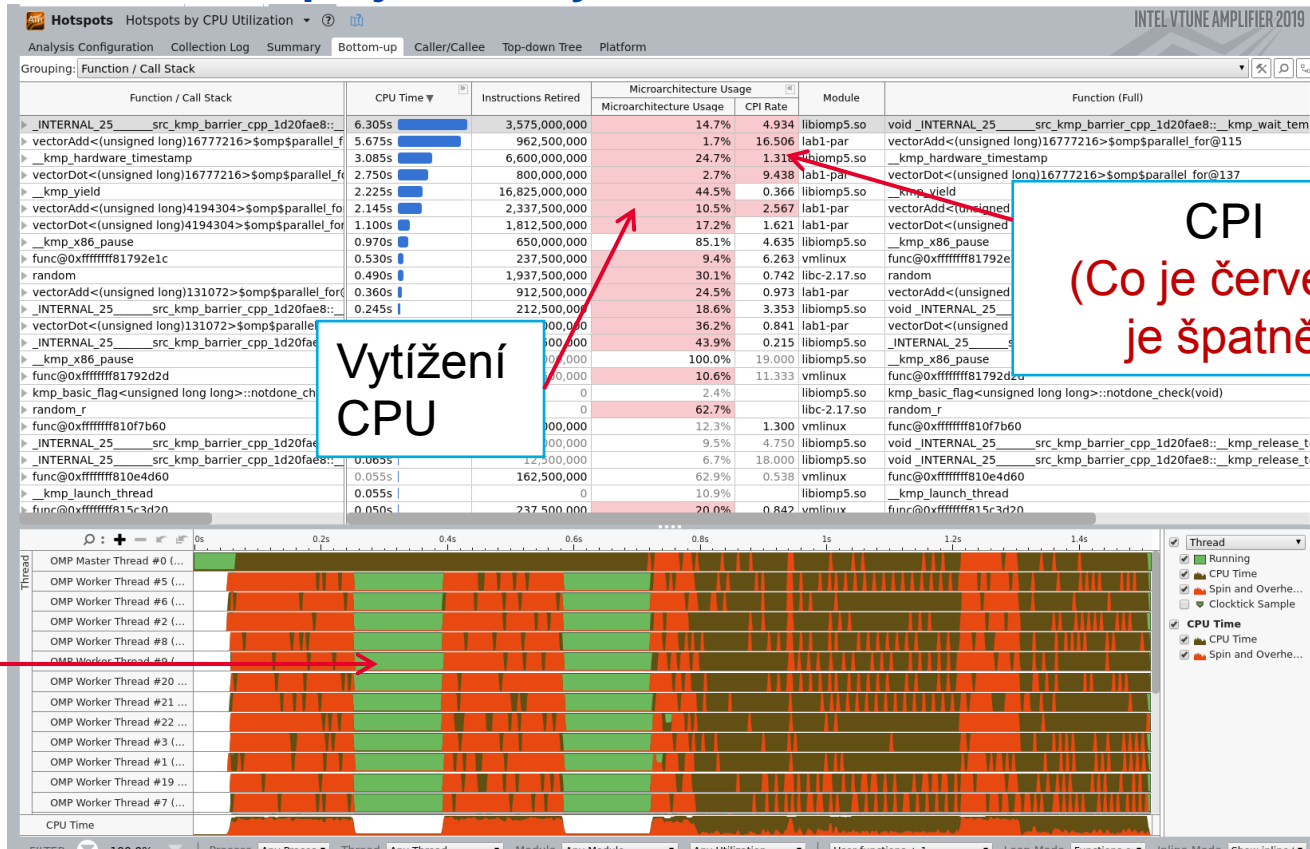


lab1-par





- Flat profile – rozbor po jednotlivých funkcích



Vytížení  
CPU

CPI  
(Co je červeně  
je špatně)

Práce  
Čekání  
Sync

Intel VTune Amplifier (on r112n8.lib0.smc.salomon.lt4.cz)

Project Navigator

- AVS-VTUNE-X
  - r000hs
  - r001hs

Configure Analysis

WHERE

Local Host

WHAT

Launch Application

Specify and configure your analysis target: an application or a script to execute.

Application:

/home/jarosjir/Vyuka/AVS/Labs/Lab1/Assignment/lab1-par

Application parameters:

☐ Use application directory as working directory

Working directory:

/home/jarosjir/Vyuka/AVS/Labs/Lab1/Assignment

Advanced

Find your analysis direction

Hotspots

Want to find out where your application spends time and optimize your algorithms?

Hotspots

Memory Consumption

Microarchitecture

Want to see how efficiently your code is using the underlying hardware?

Microarchitecture Exploration

Memory Access

Parallelism

Want to assess the compute efficiency of your multi-threaded application?

Threading

HPC Performance Characterization

Platform Analysis

Platform Profiler (preview)

System Overview

CPU/GPU Concurrency

GPU Compute/Media Hotspots

GPU In-kernel Profiling

GPU Rendering (preview)

Input and Output

CPU/FPGA Interaction (preview)

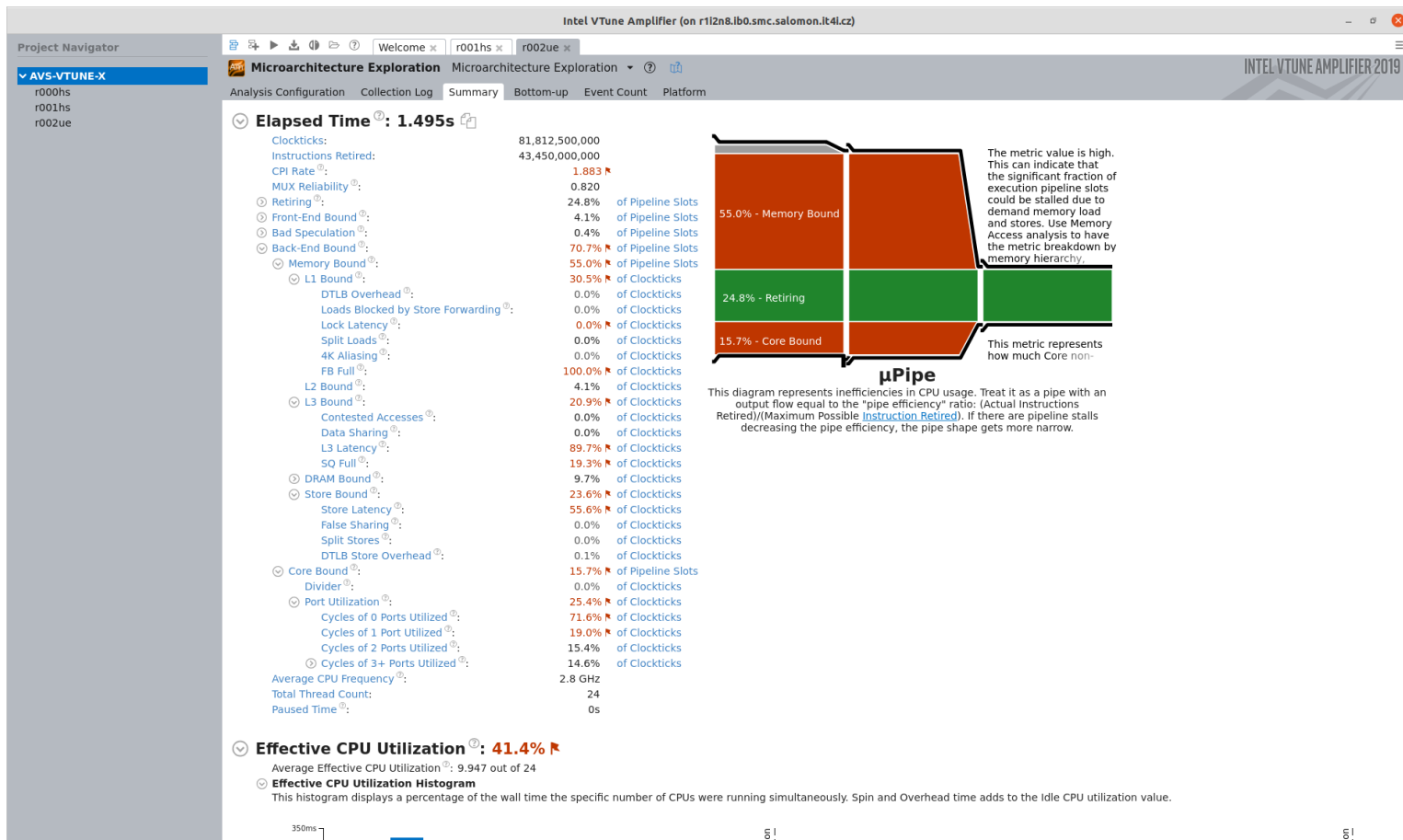
Custom Analysis

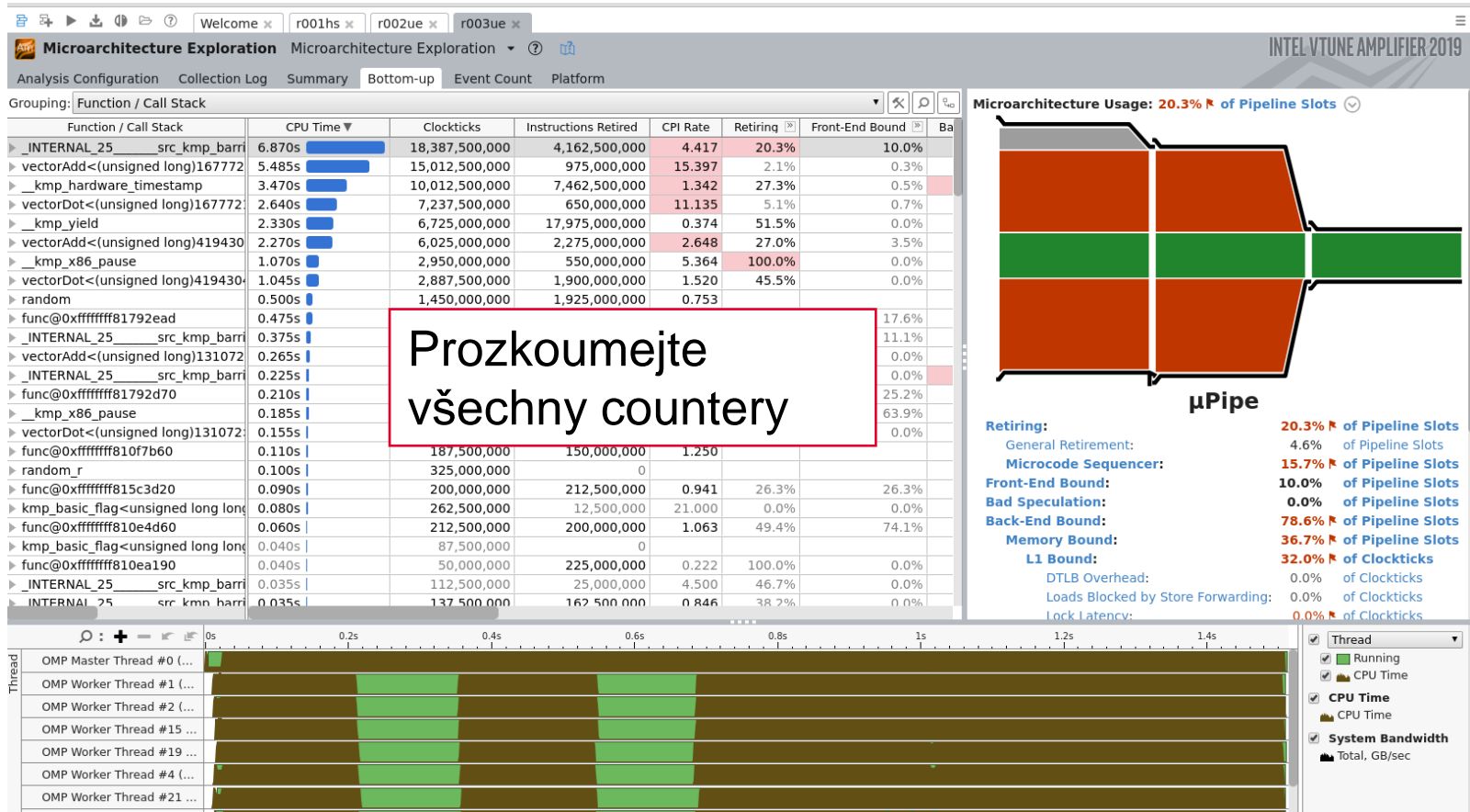
Hotspots 0

Hotspots 1

HPC Performance Characterization 0

HPC Performance Characterization 1





**Pokračování příště**