

# Porting the LHCb Stack from x86 (Intel) to aarch64 (ARM)

CHEP 2018, Sofia

---

Laura Promberger<sup>1 2</sup>

Marco Clemencic<sup>1</sup>

Ben Couturier<sup>1</sup>

Aritz Brosa Iartza<sup>1 3</sup>

Niko Neufeld<sup>1</sup>

on behalf of the LHCb collaboration

July 12, 2018

<sup>1</sup>CERN

<sup>2</sup>Hochschule Karlsruhe - Technik und Wirtschaft

<sup>3</sup>Universidad de Oviedo (ES)



Hochschule Karlsruhe  
Technik und Wirtschaft  
UNIVERSITY OF APPLIED SCIENCES

# Motivation - The Upgrade In 2021

	Currently (Run 2)	Upgrade (Run 3)
Data acquisition rate	50 GB/s	4 TB/s
Data recording rate	0.7 GB/s	2 - 10 GB/s

For the upgrade

- Software needs major refactoring and usage of new technology
- New HLT farm

Goal

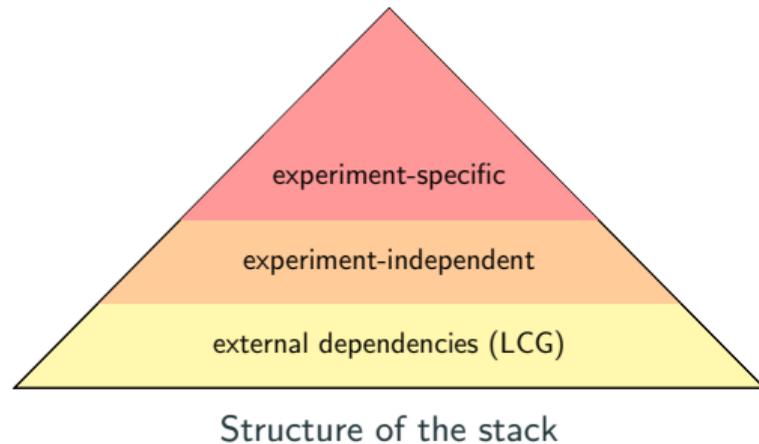
- Add cross-platform support to the LHCb stack
- More flexibility with the tender for the new HLT farm
- ⚡ Biggest Problem: Vectorization

# The LHCb Stack

- 5 million lines of code  
(experiment-specific projects)
- Multiple, large projects

For this work

- Old version of the LHCb stack (Oct 2017)
- Not multi-threaded



# Vectorization

		Vcl	Vc
Intel	AVX2	Yes	Yes
Intel	AVX512	Yes	In development
PowerPC	Altivec	No	No
ARM	NEON	No	In development
Vectorization Style		Wrapper for intrinsics	High-level, targets horizontal vectorization
Extensibility for new intrinsics		Medium (no unit tests)	Complex

→ Vcl allows 'fast' implementation of other platforms

# Port to aarch64 (ARM)

## LCG requires

- Changing compile flags
  - e. g. replace `-max-page-size=0x1000` by `-common-page-size=0x1000`
- Changing versions of the external dependencies
- Disabling unnecessary packages (e. g. Oracle, R)

## Other projects

- Changing compile flags
- Replacement of Vc by
  - Vcl
  - Scalar code

## Port to aarch64 - Problems

Default signedness of char

- Intel uses signed char
- ARM uses unsigned char

→ Use `-fsigned-char` to change the default to signed char

```
1 // Jenkins one-at-time hash function
2 static unsigned int hash32( const char* key )
3 {
4     unsigned int hash = 0;
5     for ( const char* k = key; *k; ++k ) {
6         hash += *k;
7         hash += ( hash << 10 );
8         hash ^= ( hash >> 6 );
9     }
10    hash += ( hash << 3 );
11    hash ^= ( hash >> 11 );
12    hash += ( hash << 15 );
13    return hash;
14 }
```

## Port to aarch64 - Problems II

### Cast double to unsigned int

- Intel assembly uses `vcvttsd2si`
- ARM assembly uses `fcvtzu`

```
1 if (m_xInverted == true){
2     strip = (unsigned int) floor((m_uMaxLocalu)/m_pitch) +0.5);
3 }
```

```
1 float x = -3.3;
2 unsigned int y = (unsigned int) x;
```

Problem

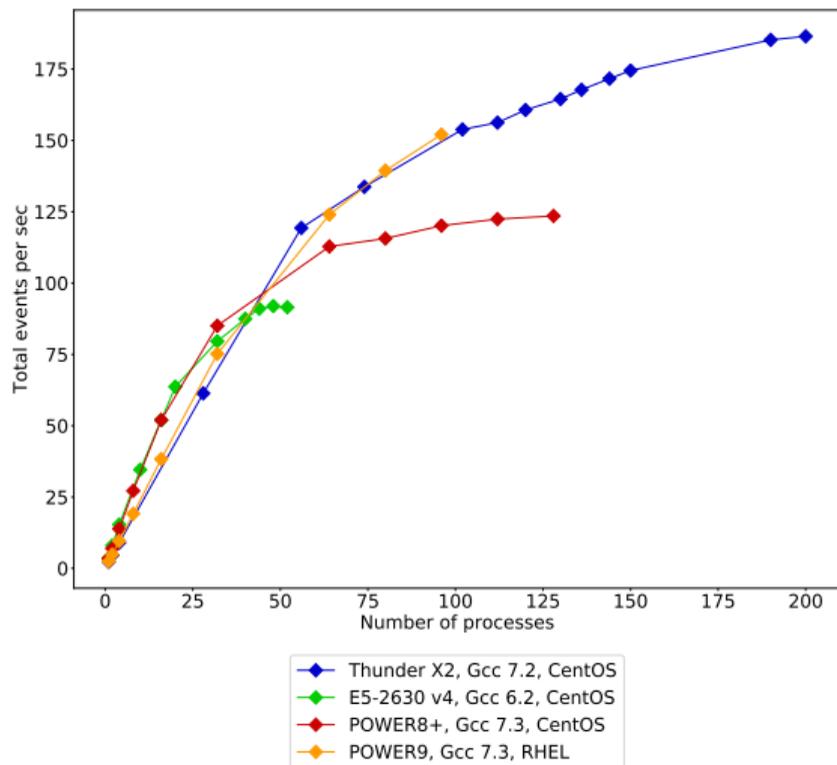
```
1 float x = -3.3;
2 uint32_t y = static_cast<uint32_t> (static_cast<int>(x));
```

Solution

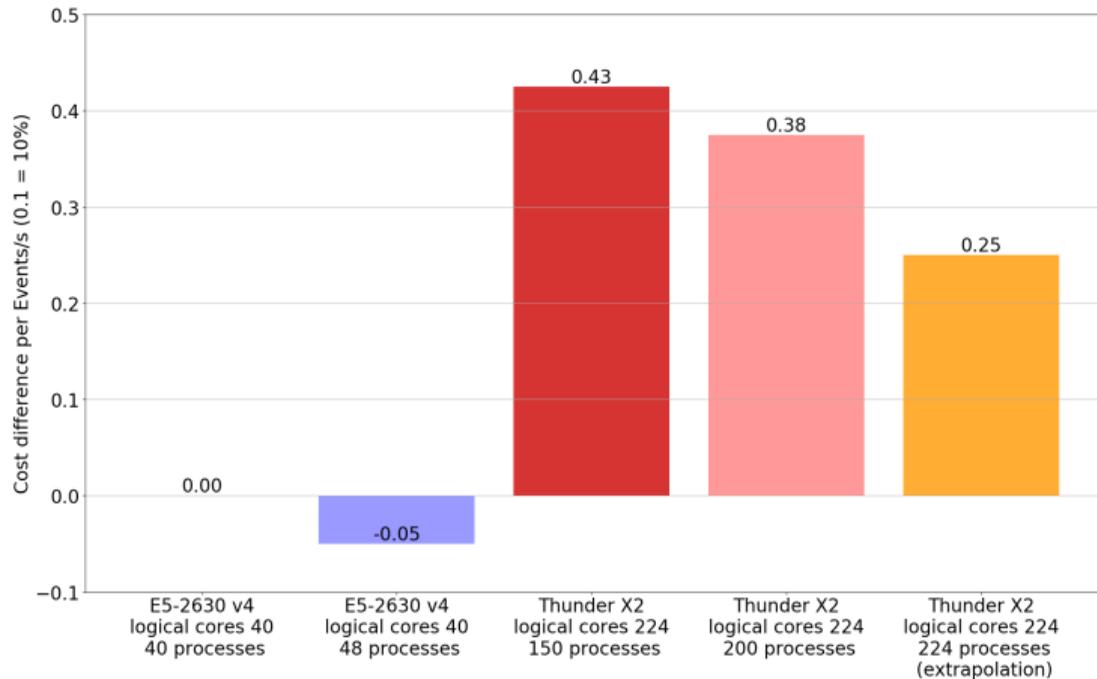
## Performance - The machines

	ThunderX2	E5-2630 v4	Power8+	Power9
Architecture	ARM	Intel	PowerPc	PowerPc
Platform	aarch64	x86_64	ppc64le	ppc64le
Compiler	GCC 7.2	GCC 6.2	GCC 7.3	GCC 7.3
Number logical cores	224	40	128	176
Threads per core	4	2	8	4
Cores per socket	28	10	8	22
Sockets/NUMA nodes	2	2	2	2
RAM (GB)	256	64	256	128
Largest intrinsic set	NEON	AVX2	AltiVec	AltiVec
CPU performance	top-notch high-tier	cost-efficient mid-tier		

# Performance - Scalability of the LHCb Stack



## Scalability II - Cost-Performance Estimations



- Long-term goal: Adding cross-platform support to the Run 3 LHCb stack
  - ⚡ Requires a fully functioning cross-platform vectorization library
- Finding a cross-platform vectorization library
  - ROOT plans to use VecCore which has both, UMESIMD and Vc as back end
    - LHCb evaluates to switch to VecCore instead of Vc and Vcl
- New vectorization intrinsic set for ARM: SVE
  - First official date for CPU release: Fujitsu - 2021
    - Too late for LHCb Run 3

- Cross-platform support of the LHCb stack for aarch64 and ppc64le
- Biggest problem: Vectorization
  - "Hackish" workarounds of Vc just for this study
- Cost-performance estimation
  - To be considered: pricing, not multi-threaded, less vectorization on aarch64
  - ARM and Intel quite close

→ Competitive tender for real evaluation necessary

Questions?

# Vectorization

		Vcl	Vc	UMESIMD
Intel	AVX2	Yes	Yes	Yes
Intel	AVX512	Yes	In development	Yes
PowerPC	Altivec	No	No	Early Example
ARM	NEON	No	In development	Early Example
Vectorization Style		Wrapper for intrinsics	High-level, targets horizontal vectorization	Wrapper for intrinsics
Extensibility for new intrinsics		Medium (no unit tests)	Complex	easy (unit tests available)

# Performance - Scalability of the LHCb Stack normalized

