Barcelona Supercomputing Center
Centro Nacional de Supercomputación

EXCELENCIA SEVERO OCHOA

Pushing RISC-V into HPC

Prof. Jesús Labarta

BSC Fellow & UPC

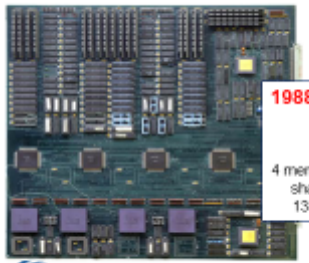
PPAM 2024

Ostrava, September 10th 2024

Disclaimer

- About myself

What I did




```
void Cholesky( float *A ) {
int i, j, k;
for (k=0; k<NT; k++) {
   spotrf (A[k*NT+k]) ;
   for (i=k+1; i<NT; i++)
      strsm (A[k*NT+k], A[k*NT+i]);
   for (i=k+1; i<NT; i++) {
      for (j=k+1; j<i; j++)
         sgemm( A[k*NT+i], A[k*NT+j], A[j*NT+i]);
      ssyrk (A[k*NT+i], A[i*NT+i]);
   }
}
```

```
#pragma omp task inout ([TS][TS]A)
void spotrf (float *A);
#pragma omp task input ([TS][TS]T) inout ([TS][TS]B)
void strsm (float *T, float *B);
#pragma omp task input ([TS][TS]A,[TS][TS]B) inout ([TS][TS]C)
void sgemm (float *A, float *B, float *C);
#pragma omp task input ([TS][TS]A) inout ([TS][TS]C)
void ssyrk (float *A, float *C);
```

2

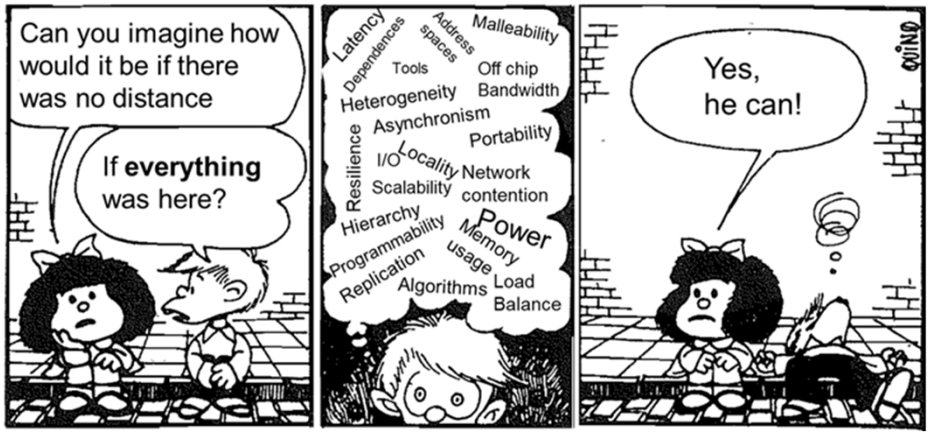
El abuelo cebolleta ataca de nuevo

The EPI FPA Objective

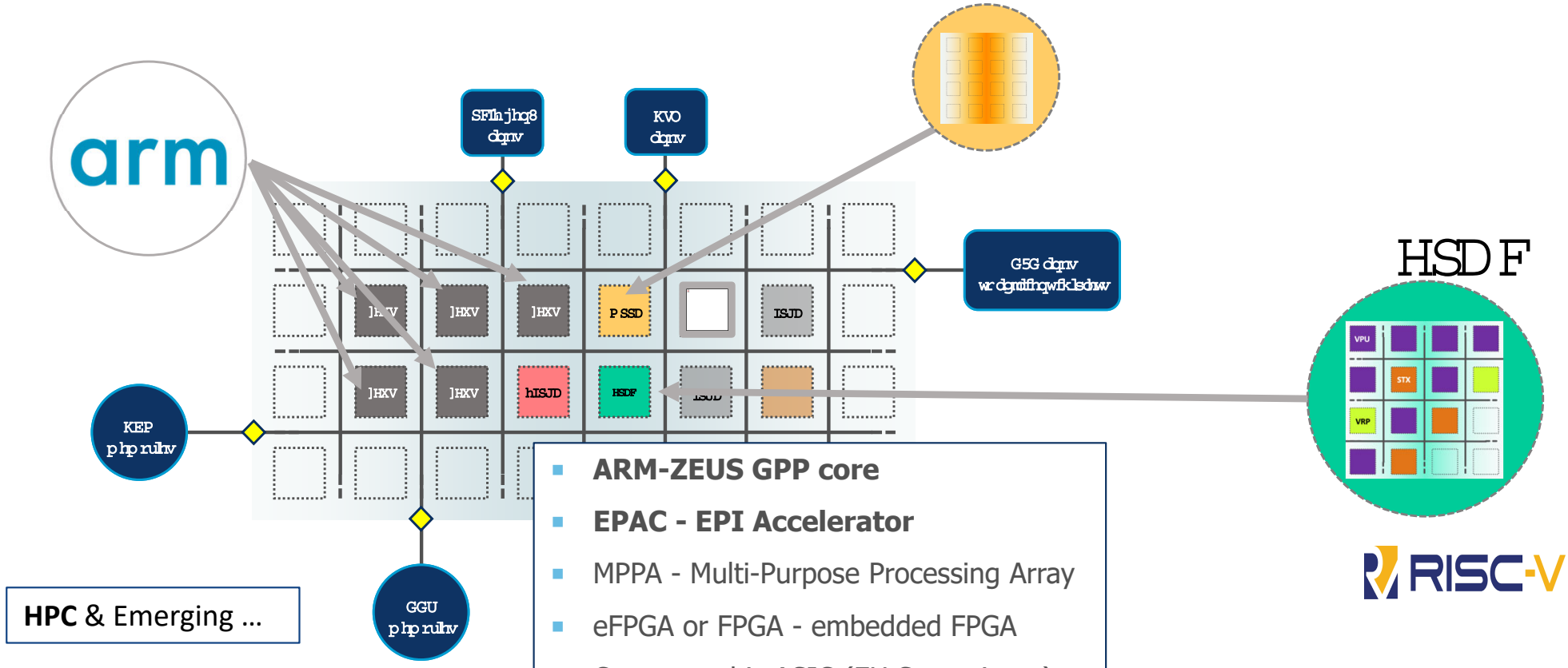
- Components (low power microprocessor technologies) …
  - ARM based SoC
  - RISCV based accelerator

- … to be combined to target
  - HPC
  - HPDA
  - Emerging
    - Automotive
    - …

European Processor Initiative



EPAC within EPI



- **ARM-ZEUS GPP core**
- **EPAC - EPI Accelerator**
- MPPA - Multi-Purpose Processing Array
- eFPGA or FPGA - embedded FPGA
- Cryptographic ASIC (EU Sovereignty)

HPC & Emerging …

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

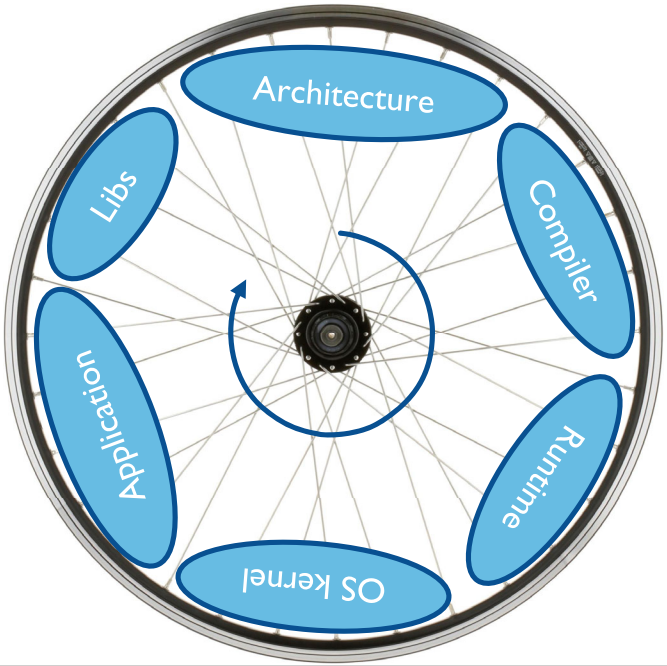
EXCELENCIA SEVERO OCHOA

Towards Holistic Co-design

Co-what ?

- Real vs buzzword ?
- You have to design !!
  - Co-design cycle?
  - Interaction timespan?
- Co-design vs co-dimension?
- Where is the compass heading?
  - My great ISA new instruction?
  - Changing every 6 months? For every code?
  - Looking for medals?
- Real insight – abstraction ?
  - Evidences? “Sufficient” quantification?
  - Models: analytic, …, cycle “accurate” simulators?
  - Granularity of analytics?
    - Time evolution/correlation/causality
  - Propagate “mistakes” of the past?
  - Overfitted ?
- Holistic vs limited focus/scope?
  - “Best” level to address an issue ? Split ?
  - Generality? Uses beyond motivating case ?
  - “Vendor/design” specific vs. fundamentals?

- Sense of co-st and maintainability/sustainability
- Co-opetition ?
  - Co-nfidence with designer? Not trivial, not many ?
  - Co-mmitment?



Co-nfidence

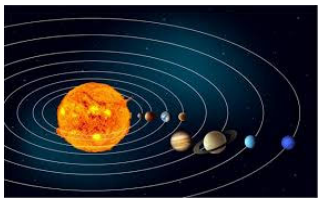
Holistic co-design

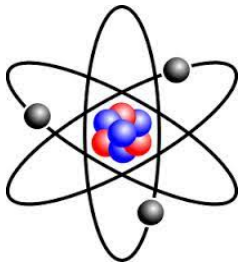
- Applications
- Libraries/Platforms
- Schedulers
- Compiler/Toolchain
- OS
- HW Systems
- CPUs/GPUs/ASICs

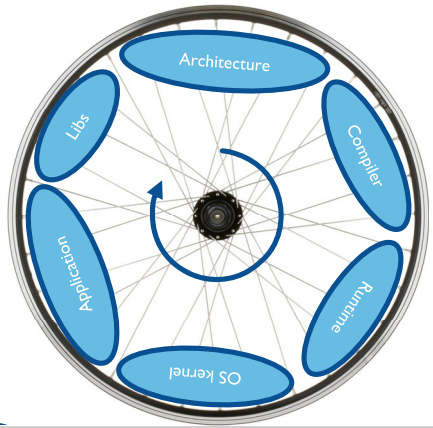
"As above, so below"
Similar concepts/mechanisms/tools at **all levels**

"Steered by a vision"

"Steered by detailed insight"







Co-nfidence

Scalability : not about size, about **dynamic range**!

The programming revolution

- **From the latency age …**
  - I need something … I need it now !!!
  - Performance dominated by latency in a broad sense
    - At all levels: sequential and parallel
    - Memory and communication, control flow, synchronizations

- **… to the throughput age**
  - Ability to instantiate “lots” of work and avoid stalling for specific requests
    - I need this and this and that … and as long as it keeps coming I am ok
    - At all levels
  - Performance dominated by overall availability/balance of resources

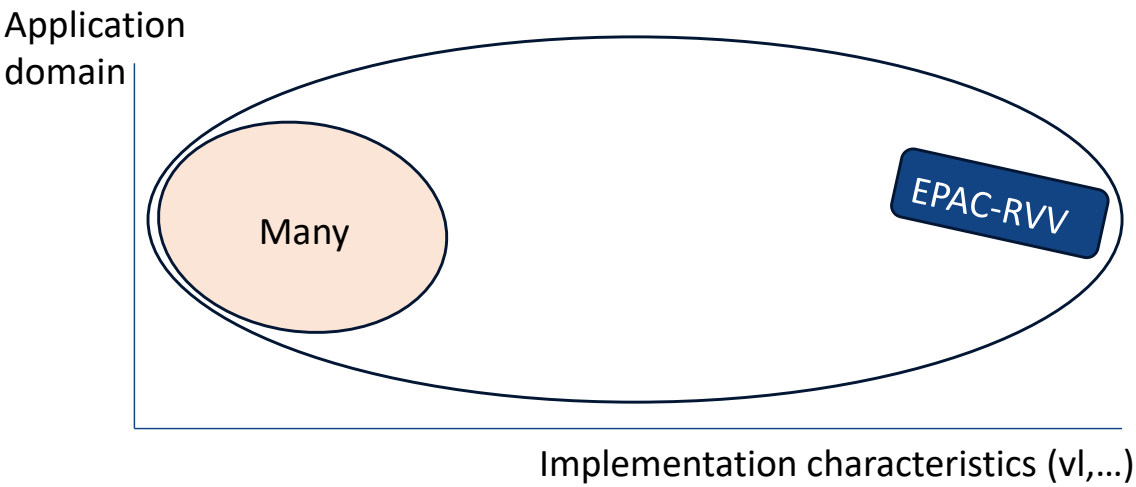
Vision

Long vectors

RISC-V … an enabler

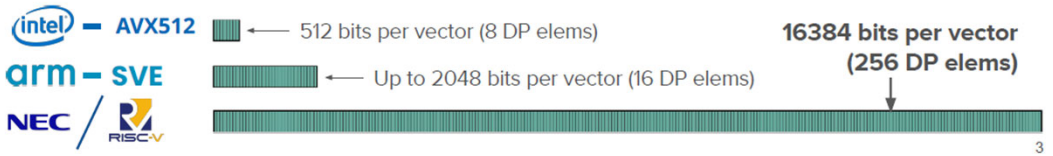
- Open Source & Standards
  - Have others work for you !
    - Leverage/build on others contributions
    - Let others use your contributions
- Flexible standards
  - Positioning / Differentiation
    - Potential for expansion

- flexible
- affordable



Long vectors

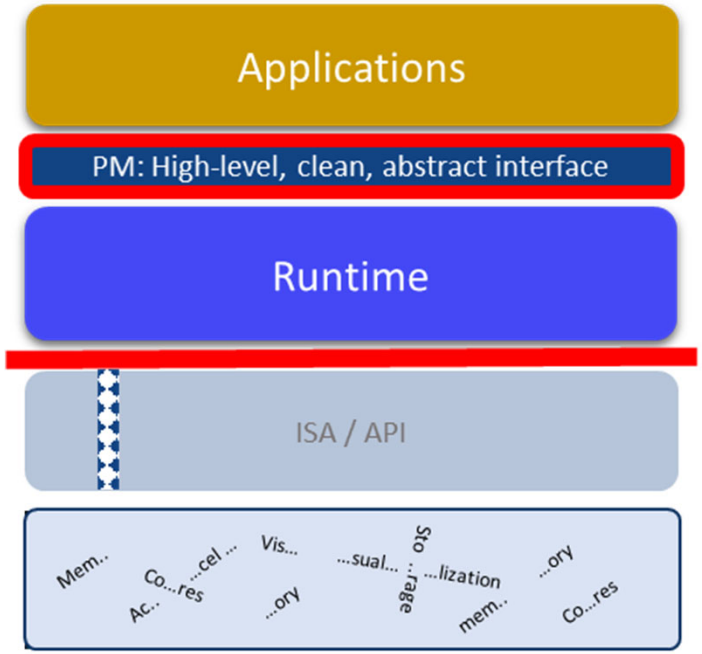
- **Long vector ISA**
  - Tolerate latency



- Decoupled vector processor
  - 1 instruction N cycles
  - Laxity for saving cost
  - Laxity for "intelligence"

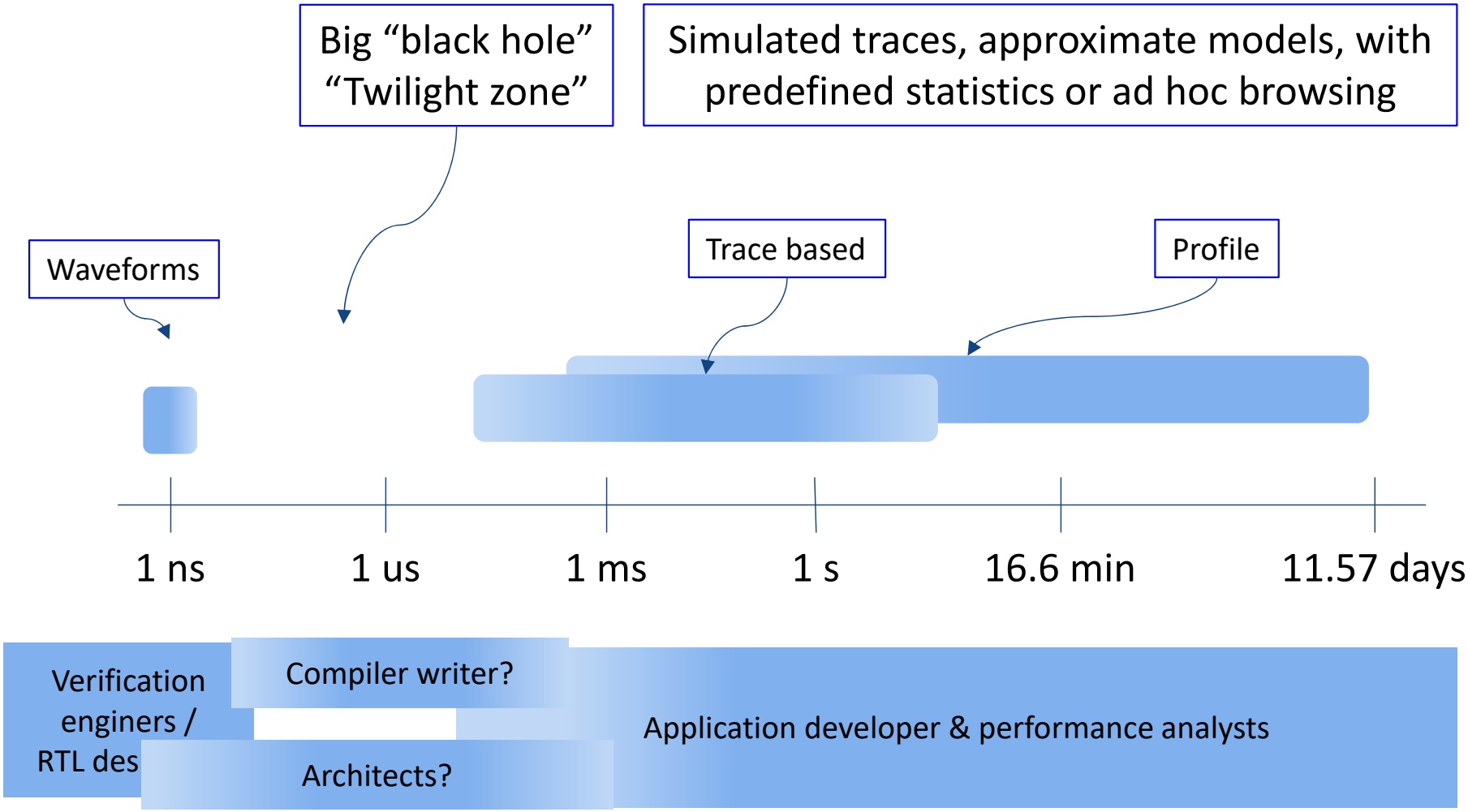
- The ISA osmotic membrane
  - Vector Length Agnostic (VLA)
  - Concurrency and access pattern semantics
  - Hints: locality, …

- Programming productivity
  - As close as possible to standard sequential
  - Mindset change
    - Relative importance
      - overheads, parallelism & locality
  - Incremental and reversible



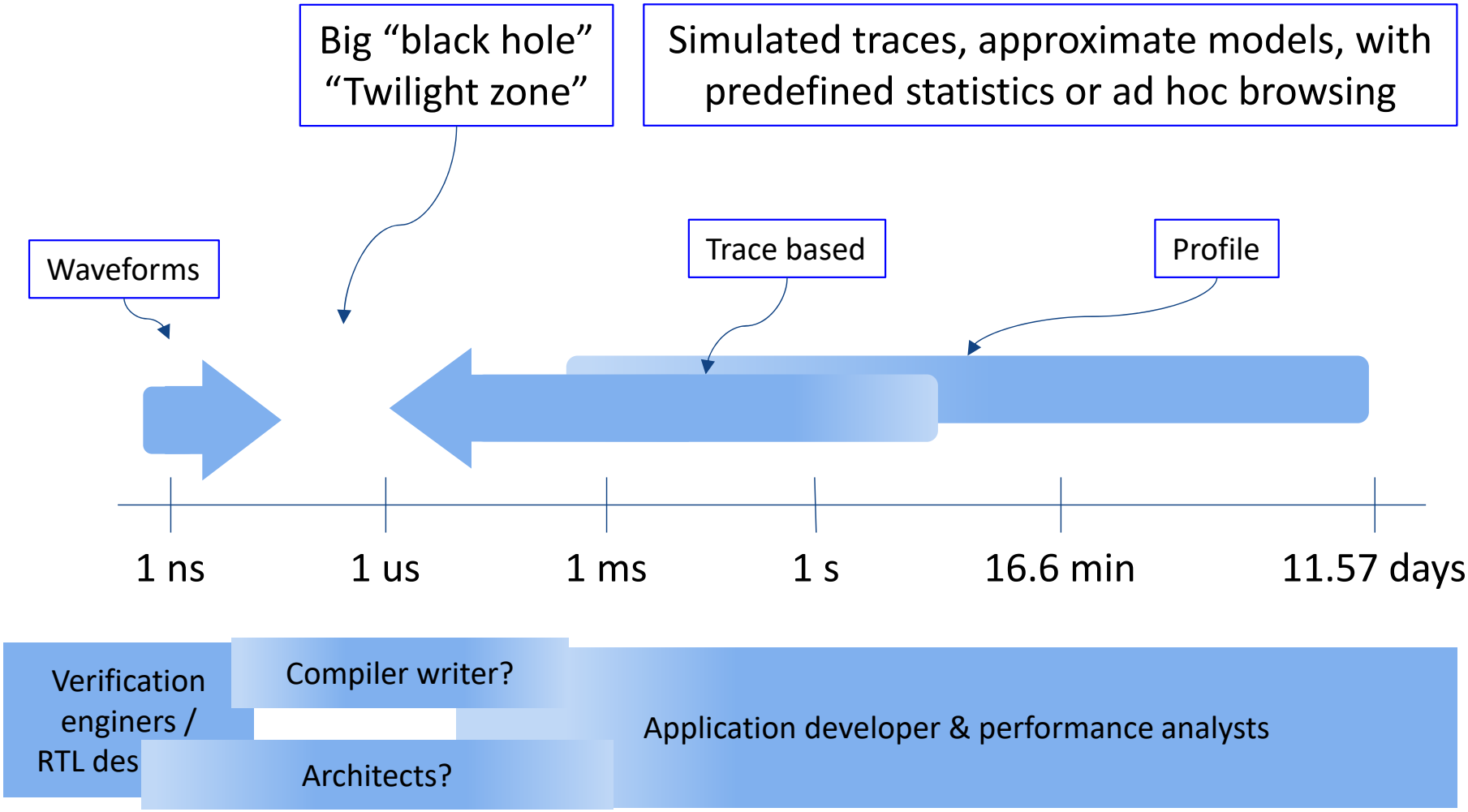
Scalability of analysis

- Microscopic “structure” determines macroscopic behavior



Scalability of analysis

- Microscopic “structure” determines macroscopic behavior



@production runs

- Real behavior, and “scales” …
- … to distill fundamental insight

JU CoE Projects

European Processor Initiative

THE EUPILOT

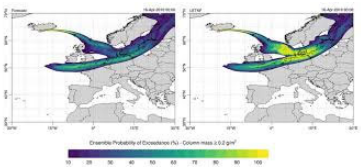
ChEESE

CEEC Centre of Excellence in Exascale CFD

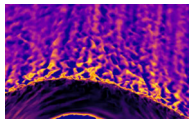
PLASMA PEPSC

Codes

FALL3D



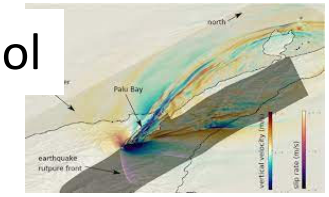
Vlasiator



FAST. FLEXIBLE. FREE. GROMACS

PyTorch

Seissol



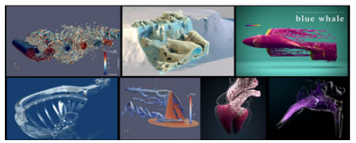
NECO

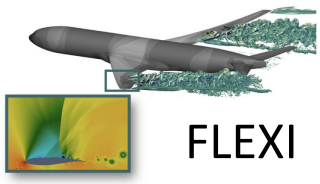
EC EARTH

OneDNN

oneAPI

Alya





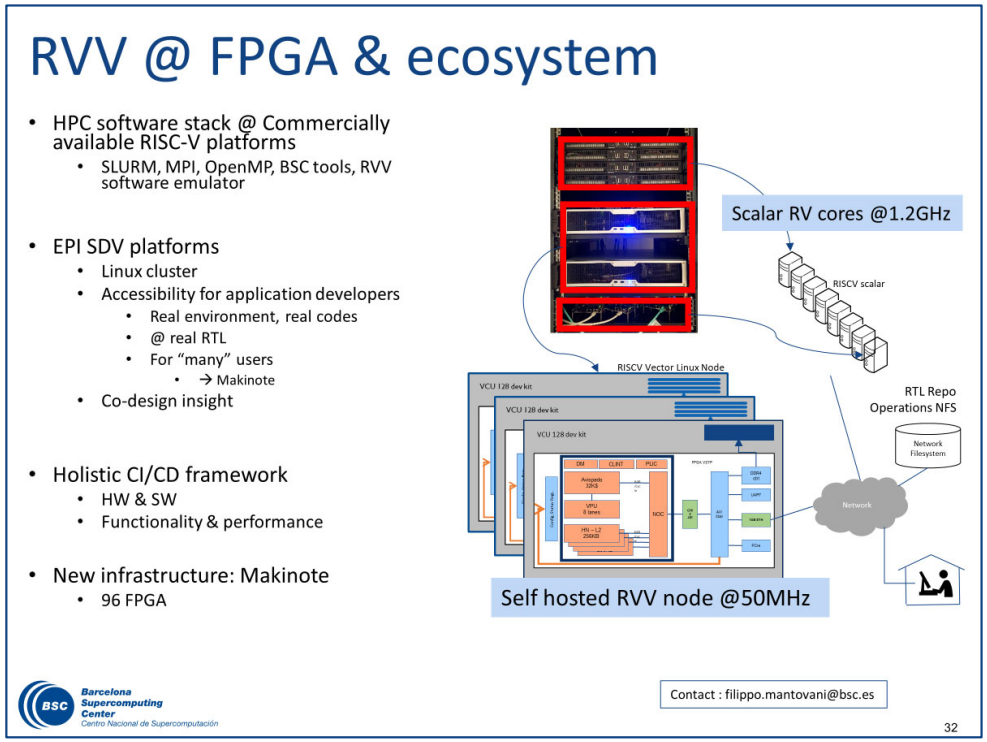
FLEXI

Gene-X

LLMs

Towards emulators for all !!

- Hardware emulators typically used by verification engineers and with “limited” analysis capability



- → “widely” available to performance analysts and application developers
- → with powerful analysis capabilities

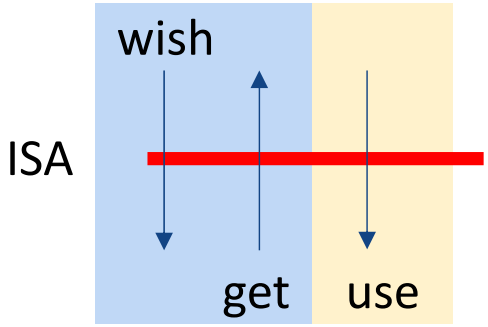
Barcelona Supercomputing Center
Centro Nacional de Supercomputación

EXCELENCIA SEVERO OCHOA

About RISC-V Vector extension

VLA programming

- Decouple program concurrency structure from a specific vector length
  - Negotiation demands – allocated resources
  - Program adapts its concurrency to available resource/problem size
  - Allocation policies
    - Static
    - Lots of opportunities for smart allocations
  - Portability, performance, code footprint, …

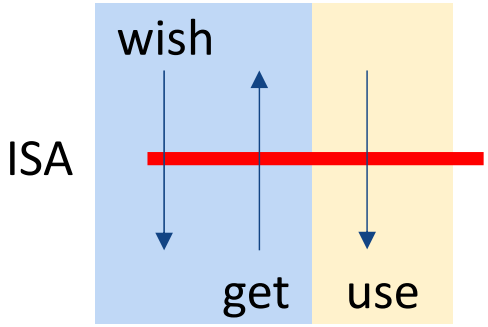


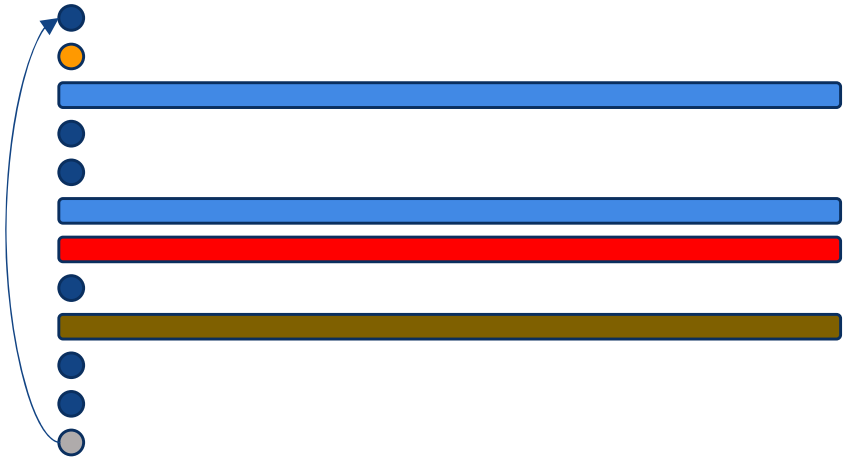
- Cooperation program – architecture
  - Supported by architecture: RVV, ARM SVE
  - Followed by programming practice

- Malleability: a good practice at all levels !!

RVV VLA programming

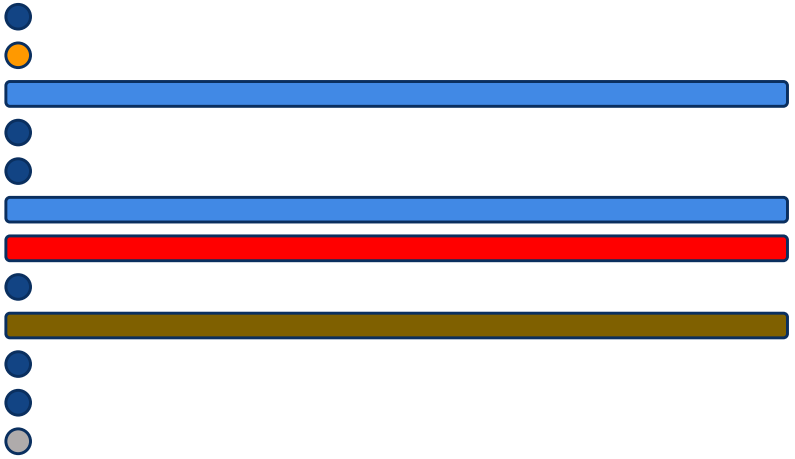
```
void axpy_intrinsics  (double a, double *dx, double *dy, int n) {
   int i;
   int gvl = __builtin_epi_vsetvl(n, __epi_e64, __epi_m1);
    __epi_1xf64 v_a = __builtin_epi_vbroadcast_1xf64(a, gvl);

   for (i=0; i<n; ) {
       gvl = __builtin_epi_vsetvl(n - i, __epi_e64, __epi_m1);
       __epi_1xf64 v_dx = __builtin_epi_vload_1xf64(&dx[i], gvl);
       __epi_1xf64 v_dy = __builtin_epi_vload_1xf64(&dy[i], gvl);
       __epi_1xf64 v_res = __builtin_epi_vfmacc_1xf64(v_dy, v_a, v_dx, gvl);
       __builtin_epi_vstore_1xf64(&dy[i], v_res, gvl);
       i += gvl;
    }
}
```



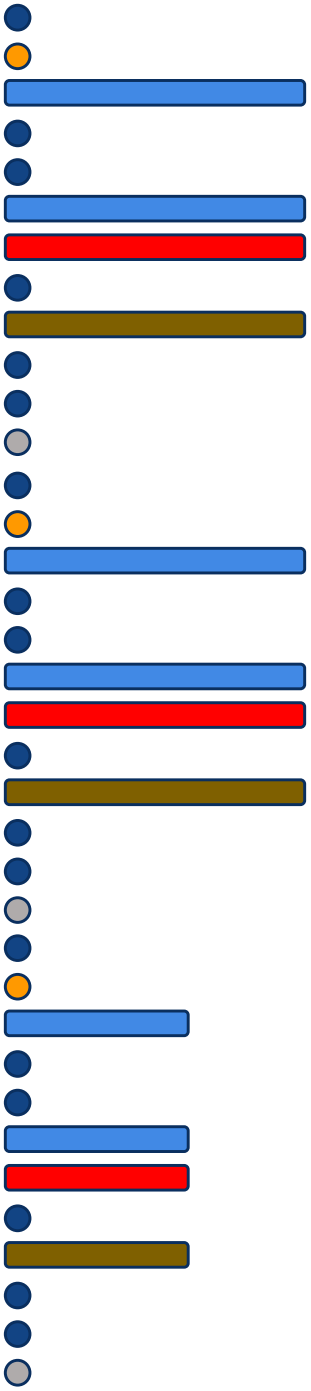


RVV VLA programming

Execution on "ideal" machine
Infinite vector length
"One" cycle execution

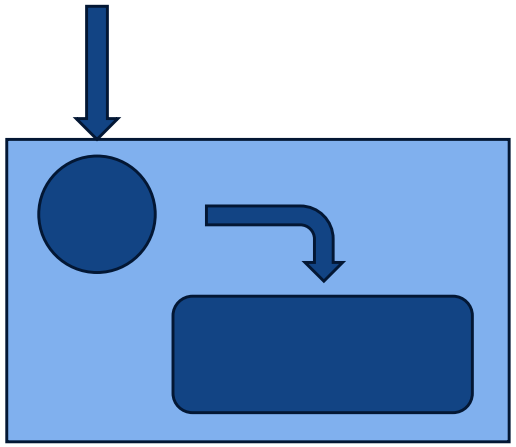


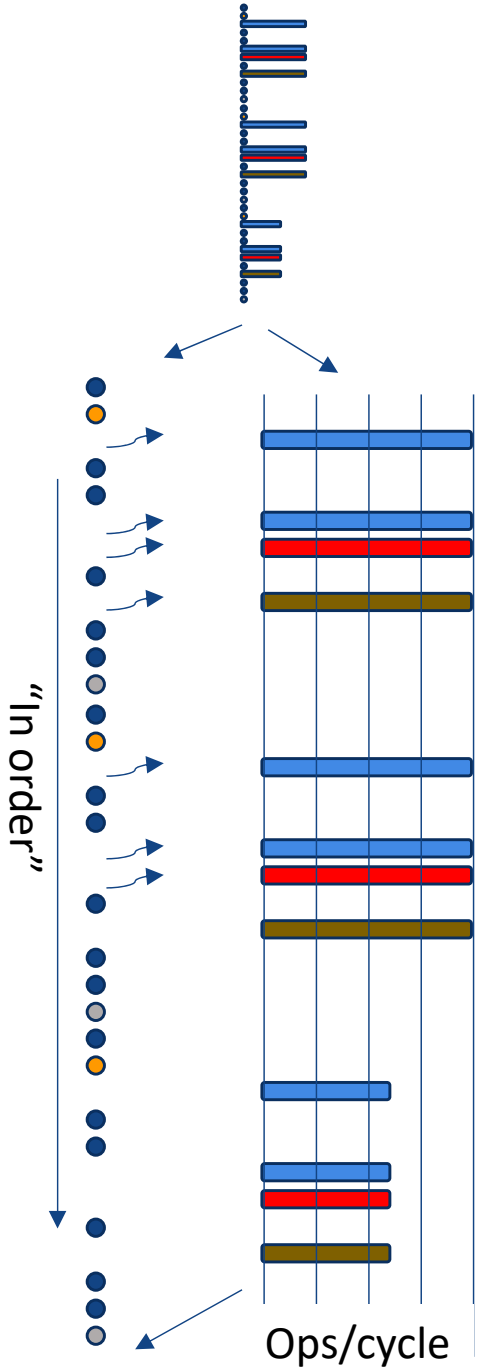
Execution on practical machine (finite MAXVL)



```
void axpy_intrinsics  (double a, double *dx, double *dy, int n) {
   int i;
   int gvl = __builtin_epi_vsetvl(n, __epi_e64, __epi_m1);
    __epi_1xf64 v_a = __builtin_epi_vbroadcast_1xf64(a, gvl);

   for (i=0; i<n; ) {
       gvl = __builtin_epi_vsetvl(n - i, __epi_e64, __epi_m1);
       __epi_1xf64 v_dx = __builtin_epi_vload_1xf64(&dx[i], gvl);
       __epi_1xf64 v_dy = __builtin_epi_vload_1xf64(&dy[i], gvl);
       __epi_1xf64 v_res = __builtin_epi_vfmacc_1xf64(v_dy, v_a, v_dx, gvl);
       __builtin_epi_vstore_1xf64(&dy[i], v_res, gvl);
       i += gvl;
    }
}
```

Decoupled vector architecture





Transparent "offloading"
Minimal cost
Single control flow
Low pressure on front end
Opportunity for intelligence

Limited OoO

Concurrent execution on different types of resources

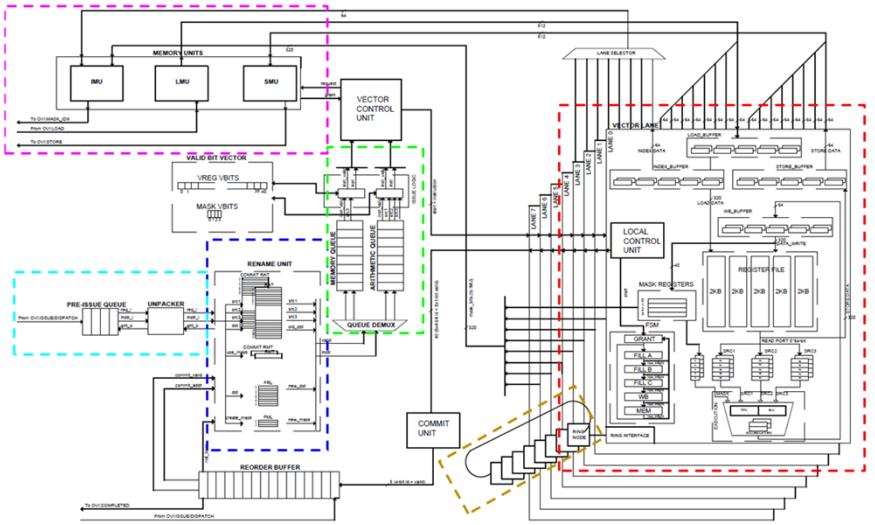
"Sporadic" dependences (try to minimize)

Maximize lookahead

VPU: A processor in itself

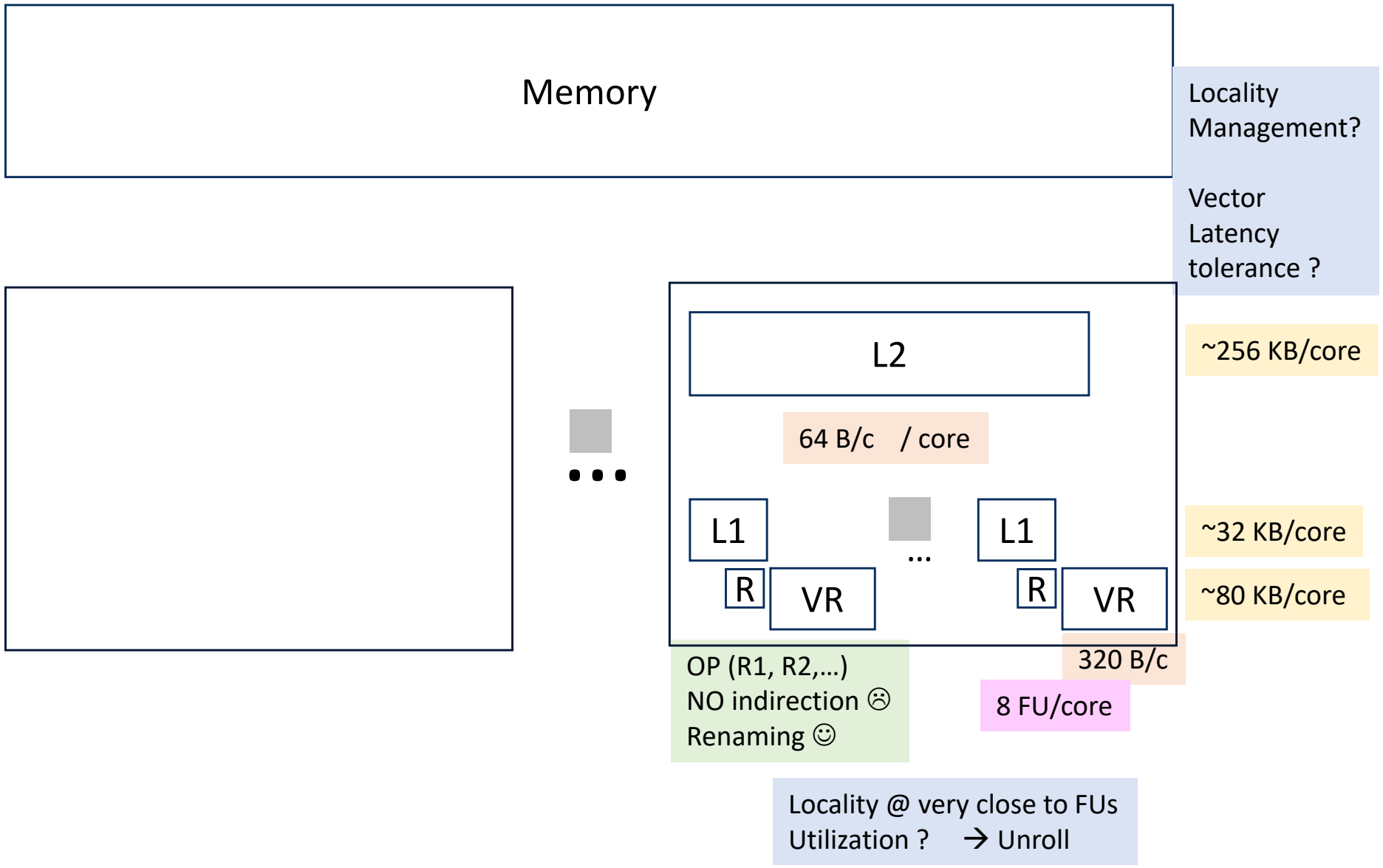
- Hierarchical “accelerator” integration
  - Program & data served by scalar core (Coherence; ~punch tape program ☺)
  - Fine grain “offloading” of “vector tasks” (directly hardware supported)
  - Homogenized heterogeneity under single “standard” ISA interface defining program order

- Implementation
  - **#FUs << VL** (lanes=8, VL=256)
  - Some OoO
    - Resources to overlap?
      - L/S, FU, shuffling
    - Renaming
      - 40 physical registers
  - Single ported register file
    - Large state
    - 5 banks/lane providing sufficient bandwidth for 1 op/cycle (latency/BW trading)
  - Data shuffling: directional ring



“Vitruvius: An Area-Efficient Decoupled Vector Accelerator for High Performance Computing”
F. Minervini, O. Palomar. RISC-V Summit 2021

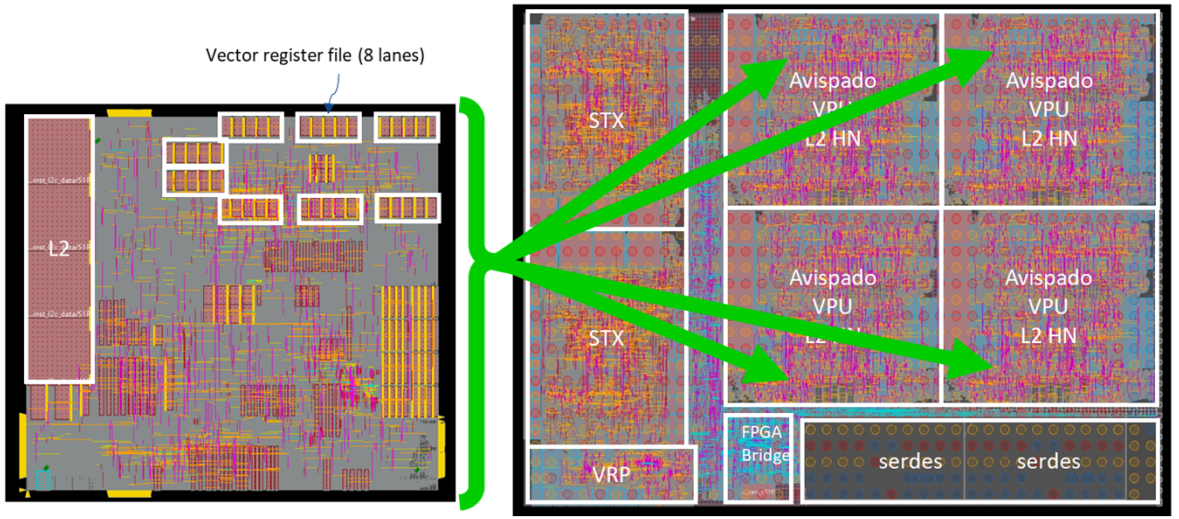
A bit on state hierarchy & locality



A more efficient way to use “comparable” amount of resources with less control flows ?
So crazy?

EPAC silicon

- EPAC 1.0
  - GF 22nm



  - Back from Fab & packaging: ~Sept 2021
  - Bring up: : Autum 2021

  - Bare metal runs

```
happy@epac$ axpy 1024
Running AXPY Scalar with 1024 array elements
init time: 45060 cycles

axpy scalar reference time: 23555 cycles

done
Result ok !!!
happy@epac$ vaxpy 1024
Running AXPY Vector with 1024 array elements
init time: 45043 cycles

axpy vector time: 932 cycles

done
Result ok !!!
happy@epac$
```

~25x ☺
while only 8x FPUs
→ Long vectors !!
→ Memory Bandwidth

RISC-V is extensible …

- Live standard and “private” instructions

- To scratchpad or not …
  - New separated address space
  - Every new address/name space is an issue !!!
    - Create, destroy
    - Explicit transfer to/from
    - Address translation?
    - Capacity limitation
    - Context switch?

- To overlay or not …
  - Two name spaces for same (set of) object(s)
  - At register level
  - ☹

RISC-V Matrix extension proposals

- Integrated Matrix Extension (IME)
  - Leveraging vector register file
  - Options
    - 1 vector register contains 1 matrix
    - 1 vector register several matrices of fixed size
    - Several vector registers collectively interpreted as 1 matrix

- Attached Matrix Extension (AME)
  - Dedicated register file
    - Data movement and computation instructions
    - Extend OS

Matrix Tile Extension

- An IME proposal by BSC (Marc Casas) and NEC (Erich Focht)
- Tile
  - Any of the 32 vector registers contains data that will be interpreted as a matrix by three operand matrix instructions
  - VL extended to tile descriptor describing how to interpret data in register.

```
typedef struct {
        unsigned int rows : 12;
        unsigned int cols : 12;
        unsigned int dtype : 7;
        unsigned int flags : 1;
} TileShape_32b;
```

  - Implicit VL register → 2 register source operands and 1 register for destination operands
    - (may be fused)
- Instructions
  - Set tile → Form Factor Agnostic (FFA) Matrix operations
  - 3 register ops → Ease OoO, no Special OS, …
  - Load/store 2 strides → Also useful for General HPC. On the fly layout changes M → R
- Leverage
  - Vector arithmetic instructions for element level operations
  - Vector L/S instructions for some address patterns

Barcelona Supercomputing Center Centro Nacional de Supercomputación 24

Matrix Tile Extension

- How GEMM might look like

Matrix GEMM $C \leftarrow \alpha AB + \beta C$

Input: A, B, C, M, N, K, LDA, LDB, LDC, α, β

Output: C

```
01:  for (int m = 0; m < M; ) {
02:    for (int n = 0; n < N; ) {
03:        tc = tile_get_shape_c_fp32(M - m, N - n);
04:        vc = vbroadcast_mask(0.0f, gvl)
05:        for (int k = 0; k < K;) {
06:            ta = tile_get_shape_a_fp32(tc.rows, K - k)
07:            tb = tile_get_shape_b_fp32(K - k, tc.cols)
08:            va = tile_load_fp32(&A[m * LDA + k], LDA, 1, ta);
09:            vb = tile_load_fp32(&B[k * LDB + n], LDB, 1, tb);
10:            vc = tile_mulacc_fp32(vc, va, vb, tc, ta, tb);
11:            k += ta.cols;
12:           }
13:        vt = tile_load_fp32(&C[m * LDC + n], LDC, 1, tc);
14:        vc = vfmul_fp32(vc, alpha, gvl);
15:        vc = vfadd_fp32(vt, beta, gvl);
16:        tile_store_fp32(vc, &C[m * LDC + n], LDC, 1, tc);
17:        n += tc.rows;
18:       }
19:      m += tc.cols
20:    }
```

Barcelona Supercomputing Center Centro Nacional de Supercomputación 25

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

EXCELENCIA SEVERO OCHOA

RVV Programming environment

Programming model

- MPI + OpenMP
  - Offloading
  - Tasks
  - SIMD
  - Intrinsics
- Pthreads, Python
- Frameworks, …

```
void axpy_omp_nest    (double a, double *dx, double *dy, int n) {
   int i, chunk;
   #pragma omp taskloop
   for (i=0; i<n; i+=TS) {
      chunk= n>i+TS? TS : n-i;
      #pragma omp target map(to:dx[i:i+chunk], tofrom:dy[i:i+chunk])
      axpy_XXXX  (a, &dx[i], &dy[i], chunk);
   }
```

```
void axpy_omp         (double a, double *dx, double *dy, int n) {
   int i, chunk;
   #pragma omp taskloop
   for (i=0; i<n; i+=TS) {
      chunk= n>i+TS? TS : n-i;
      axpy_SIMD (a, &dx[i], &dy[i], chunk);
   }
}
```

```
void axpy_SIMD        (double a, double *dx, double *dy, int n) {
   int i;
   #pragma omp simd
   for (i=0; i<n; i++) dy[i] += a*dx[i];
}
```

```
void axpy_intrinsics  (double a, double *dx, double *dy, int n) {
   int i;
   int gvl = __builtin_epi_vsetvl(n, __epi_e64, __epi_m1);
    __epi_1xf64 v_a = __builtin_epi_vbroadcast_1xf64(a, gvl);

   for (i=0; i<n; ) {
       gvl = __builtin_epi_vsetvl(n - i, __epi_e64, __epi_m1);
       __epi_1xf64 v_dx = __builtin_epi_vload_1xf64(&dx[i], gvl);
       __epi_1xf64 v_dy = __builtin_epi_vload_1xf64(&dy[i], gvl);
       __epi_1xf64 v_res = __builtin_epi_vfmacc_1xf64(v_dy, v_a, v_dx, gvl);
       __builtin_epi_vstore_1xf64(&dy[i], v_res, gvl);
       i += gvl;
    }
}
```

Compiler

- Issues
  - Maintaining 0.7 and 1.0 versions
  - Mixing Types in RVV 0.7

- VLA
  - Elegance vs possible issues?
    - Not knowing VL at compile time → some analyses not supported
  - Approaches for code generation
    - No prologs and epilogs | Vectorized epilogs
  - Impact on footprint, scheduling choices and performance, …

- Optimizations
  - Instruction scheduling
    - Cost models != scalar or SIMD
      - efficiency(vector length), overlapable resources, …
      - E.g:
        - -fno-vectorize
        - -fno-slp-vectorize
        - -mllvm –disable-loop-idiom-memset
        - -mllvm –combiner-store-merging=0
  - Loop transformations
    - Fusion, unroll and jam, …

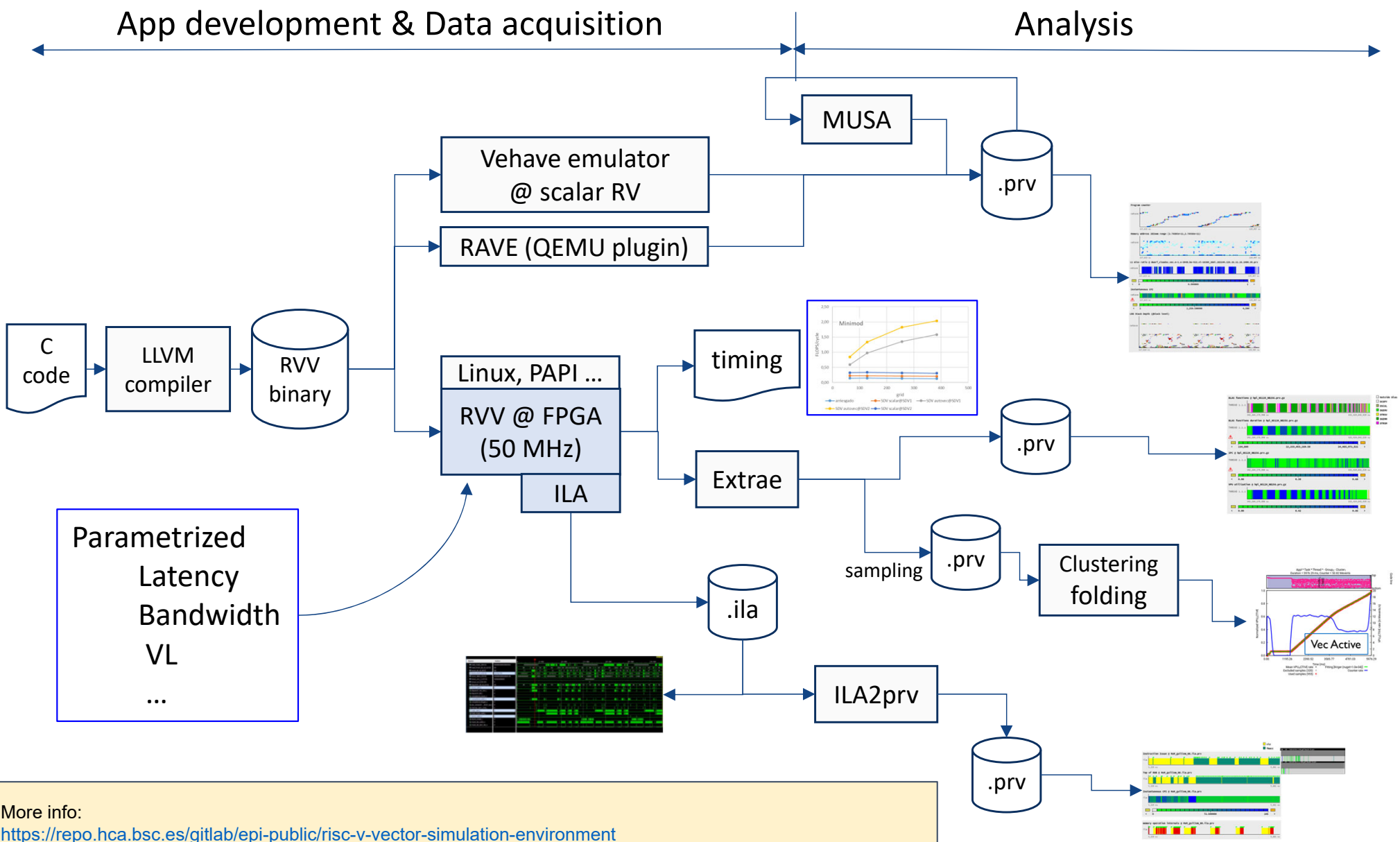
Barcelona Supercomputing Center
Centro Nacional de Supercomputación

EXCELENCIA SEVERO OCHOA

EPI co-design infrastructure

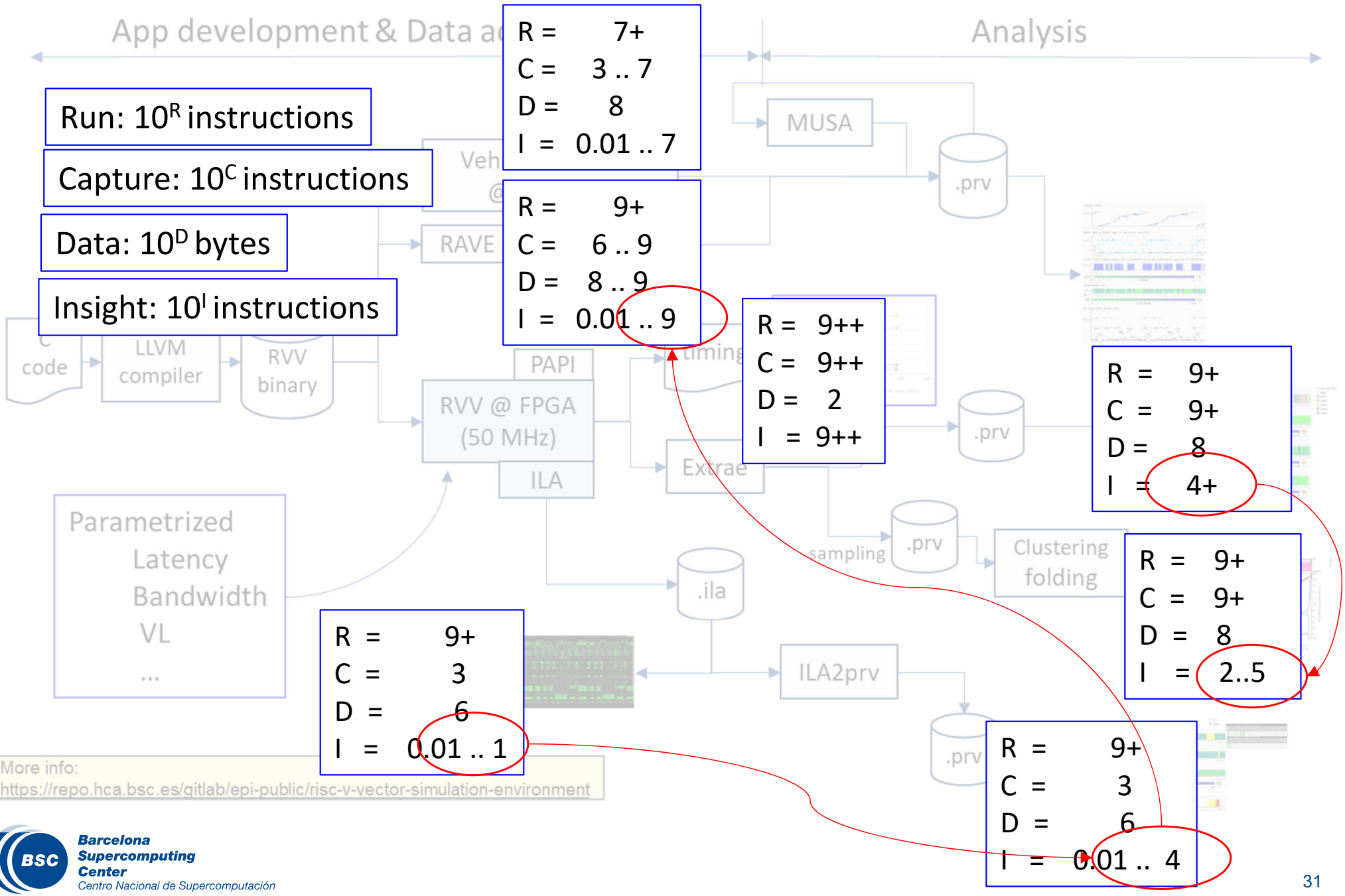
SDVs

LOD for “co-design”



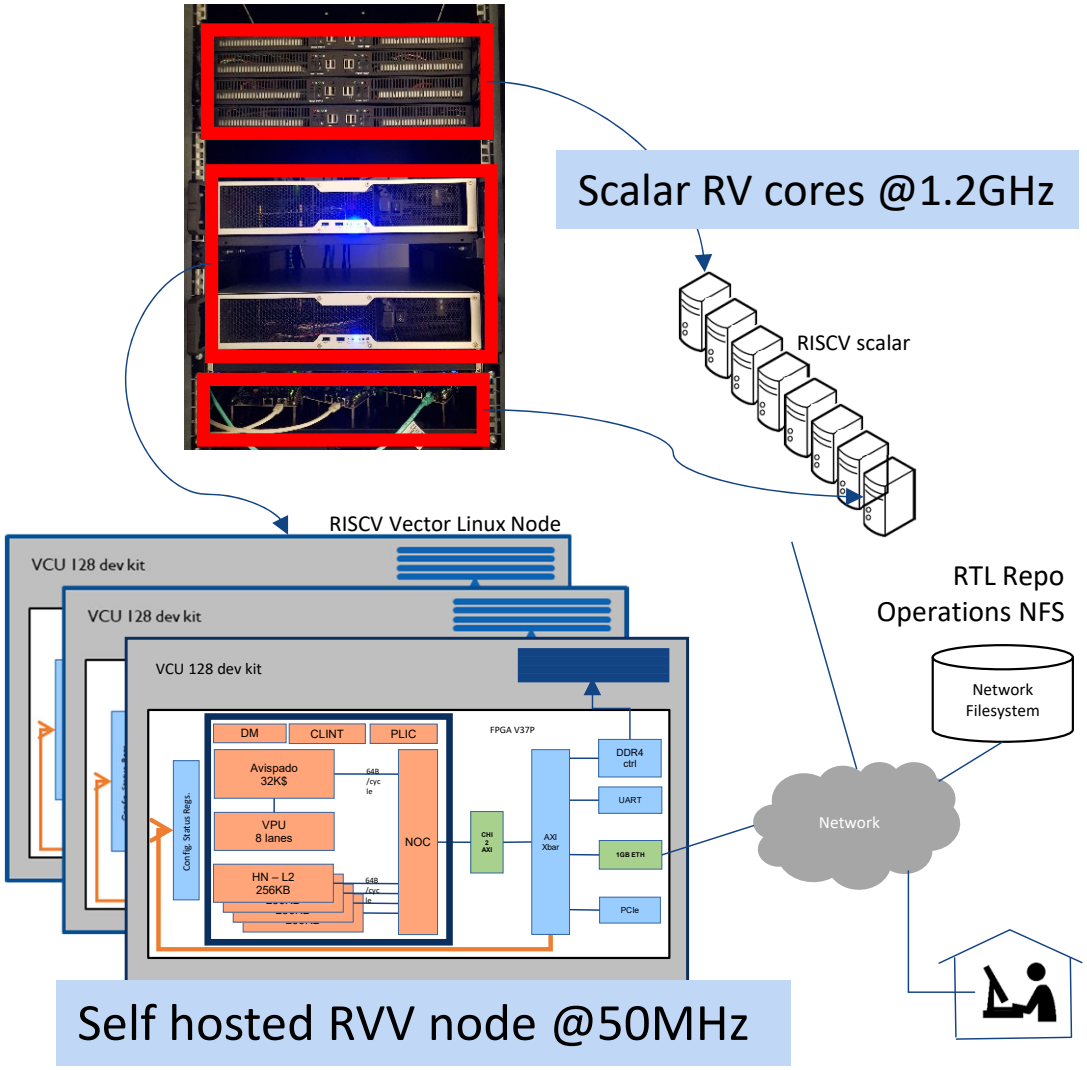
More info:
https://repo.hca.bsc.es/gitlab/epi-public/risc-v-vector-simulation-environment
https://repo.hca.bsc.es/gitlab/epi-public/risc-v-vector-simulation-environment/-/wikis/SDV-Vector-Analysis-Tutorial

Granularities



RVV @ FPGA & ecosystem

- HPC software stack @ Commercially available RISC-V platforms
  - SLURM, MPI, OpenMP, BSC tools, RVV software emulator
- EPI SDV platforms
  - Linux cluster
  - Accessibility for application developers
    - Real environment, real codes
    - @ real RTL
    - For “many” users
      - → Makinote
  - Co-design insight
- Holistic CI/CD framework
  - HW & SW
  - Functionality & performance
- New infrastructure: Makinote
  - 96 FPGA



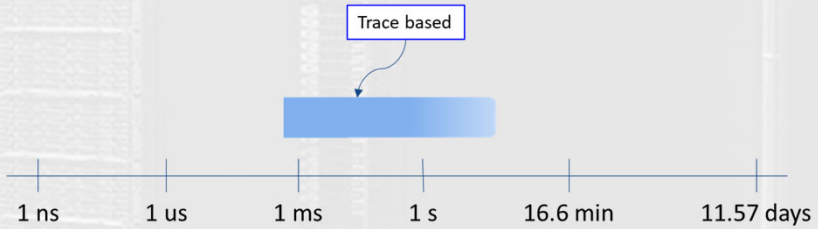
Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Contact : filippo.mantovani@bsc.es

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

EXCELENCIA SEVERO OCHOA

Extrae traces



Extrae @ SDV

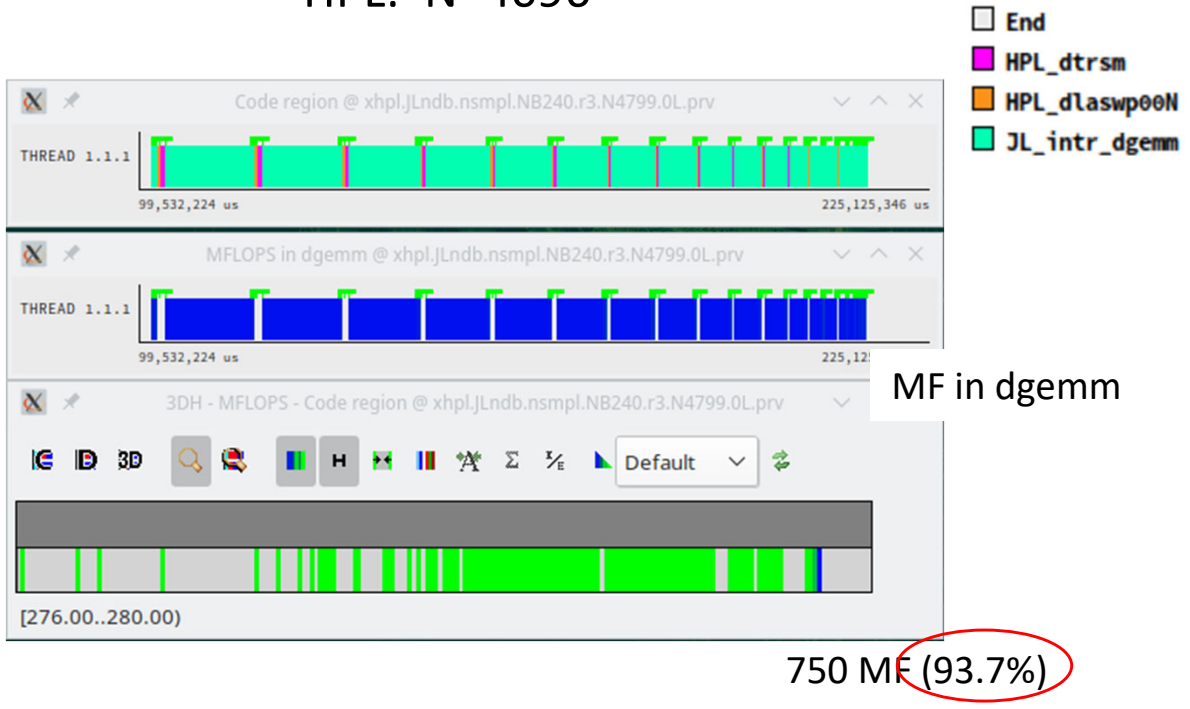
- The standard BSC tools instrumentation package (Extrae) is installed on the SDV
  - Instrumentation + sampling
  - Hardware counters

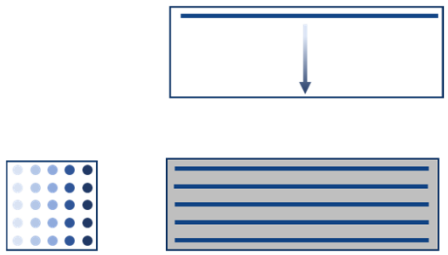
- Generates paraver traces that can be analyzed with the standard Paraver and analytics tools.

Linpack

- User events
  - API to flag region events
  - Additional ser events
    - M, N, K
    - From where Paraver can derive MFLOPS, form factors, …
  - HW counters
    - Typical: I, cyc, L1, TLB
    - Other: VEC stalls & utilization

HPL: N=4096

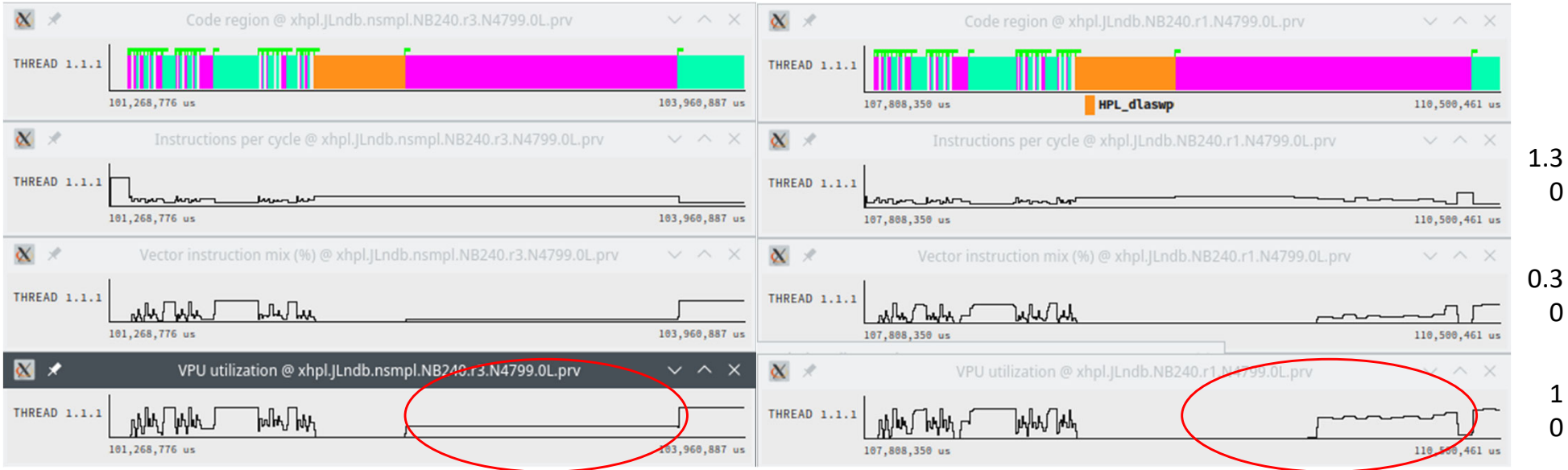




Instrumentation and sampling

No sampling

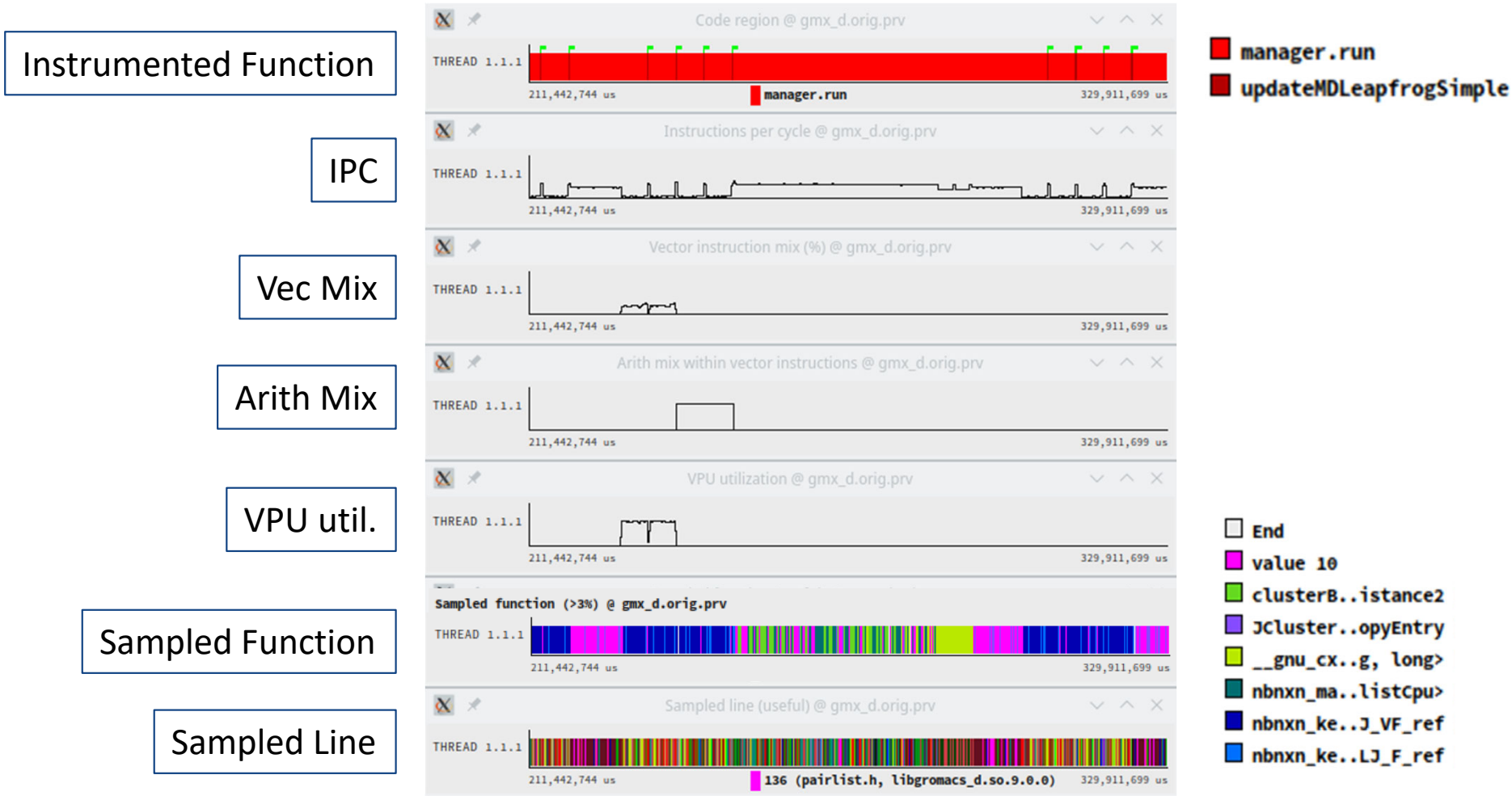
With sampling



DRTSM: Not all vectorized
intermix scalar and vector instructions
copies to temporary storage

GROMACS

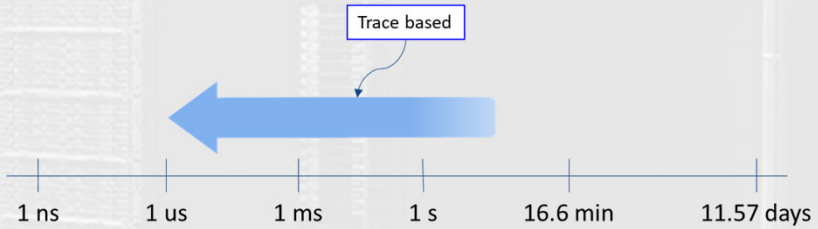
- Vectorized a single (large) loop with pragma
- Instrumentation and sampling (HWC multiplexing)



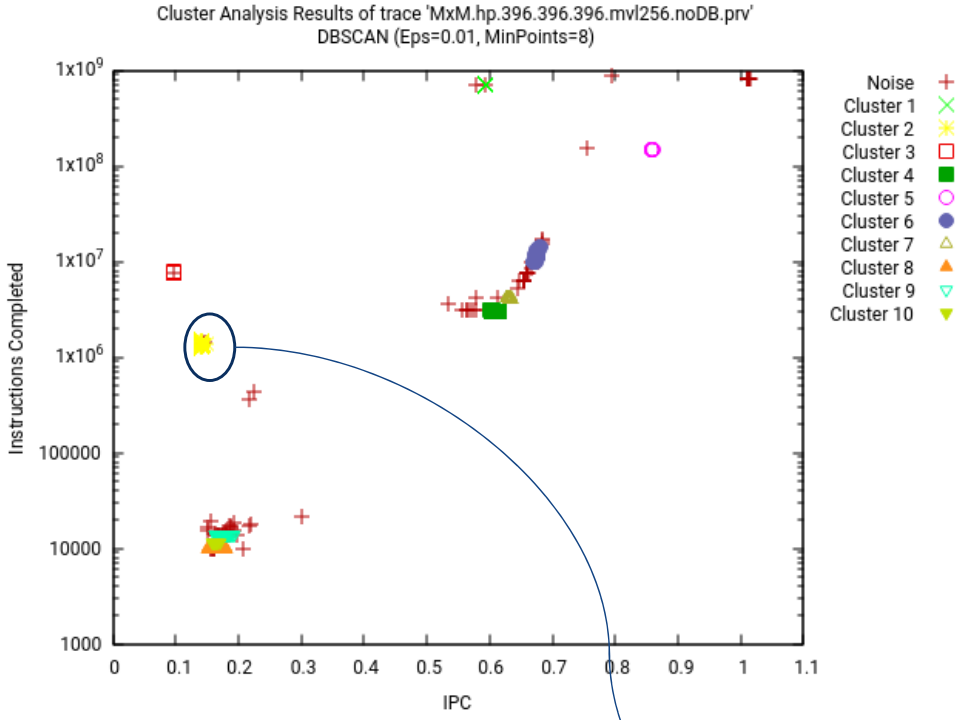
Barcelona Supercomputing Center
Centro Nacional de Supercomputación

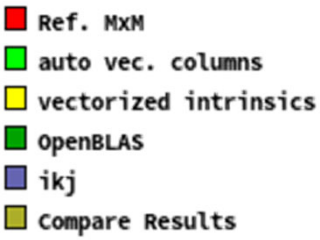
EXCELENCIA SEVERO OCHOA

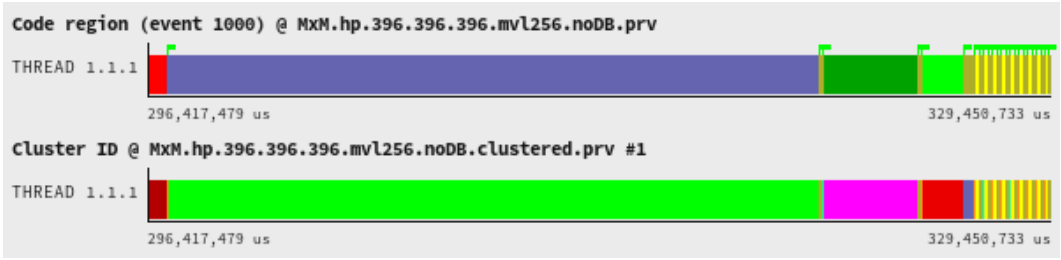
Sampling & folding

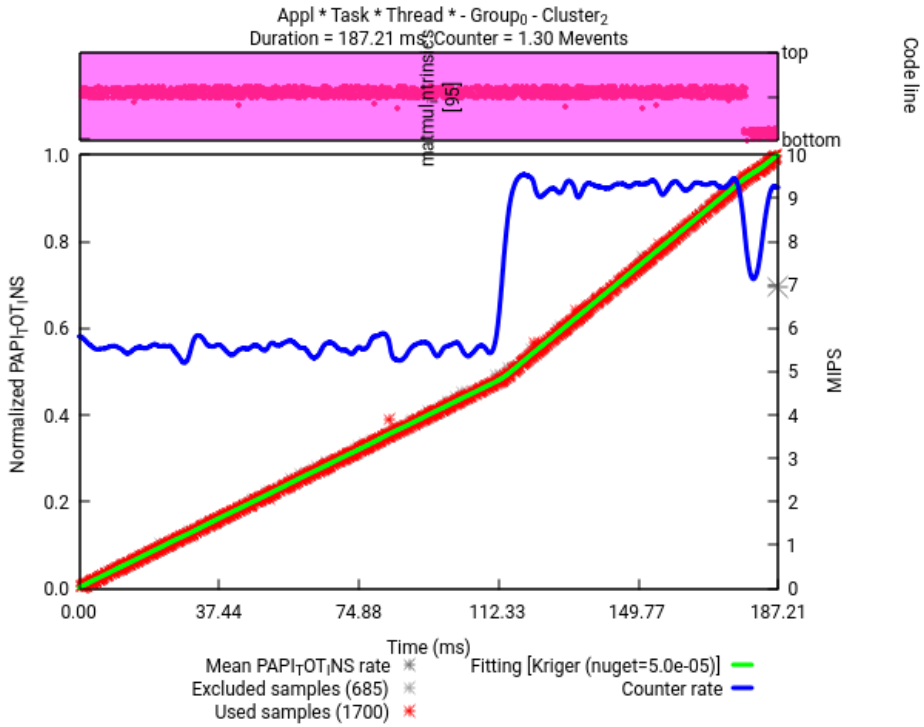


MxMs









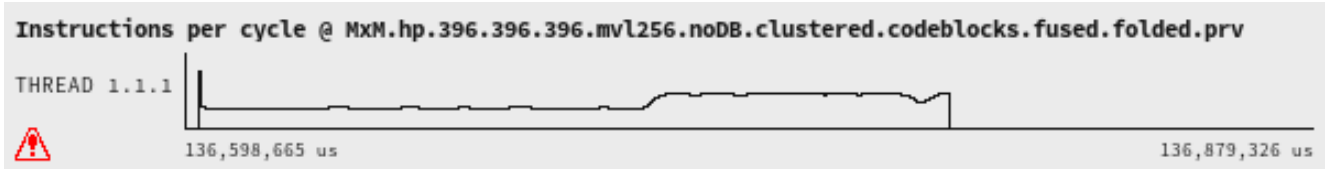
Harald Servat et al. "Detailed performance analysis using coarse grain sampling" PROPER@EUROPAR, 2009

Harald Servat et al. "Unveiling Internal Evolution of Parallel Application Computation Phases" ICPP 2011

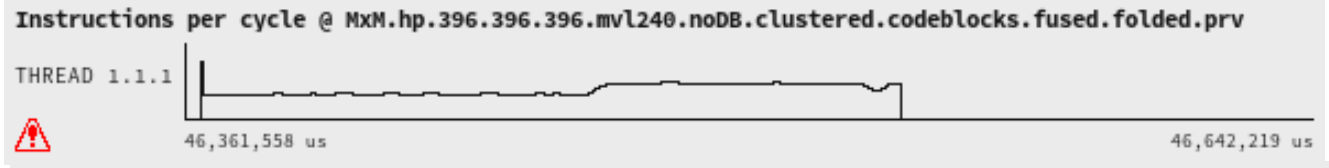
MxMs

- Impact of MAXVL

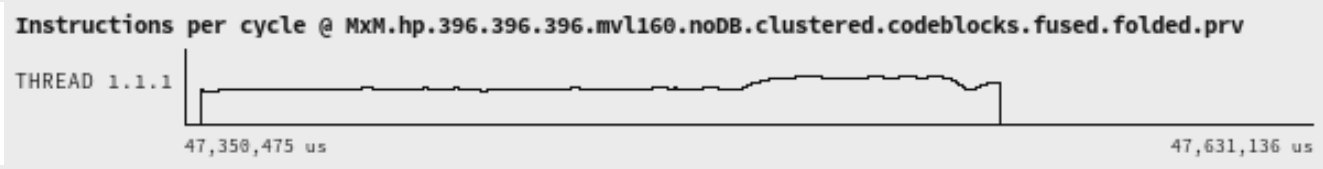
MVL 256



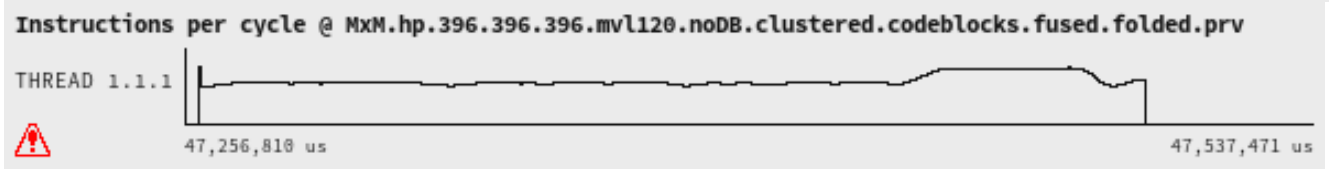
MVL 240



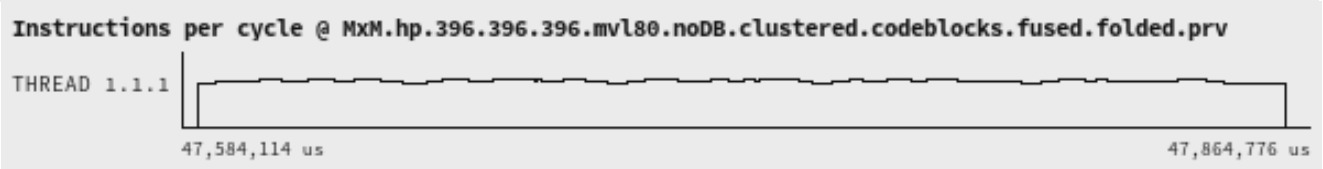
MVL 160



MVL 120



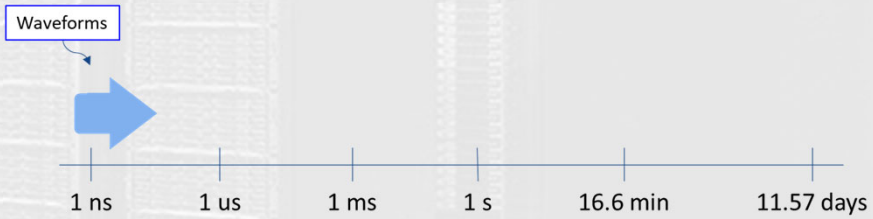
MVL 80



Barcelona Supercomputing Center
Centro Nacional de Supercomputación

EXCELENCIA SEVERO OCHOA

Waveforms



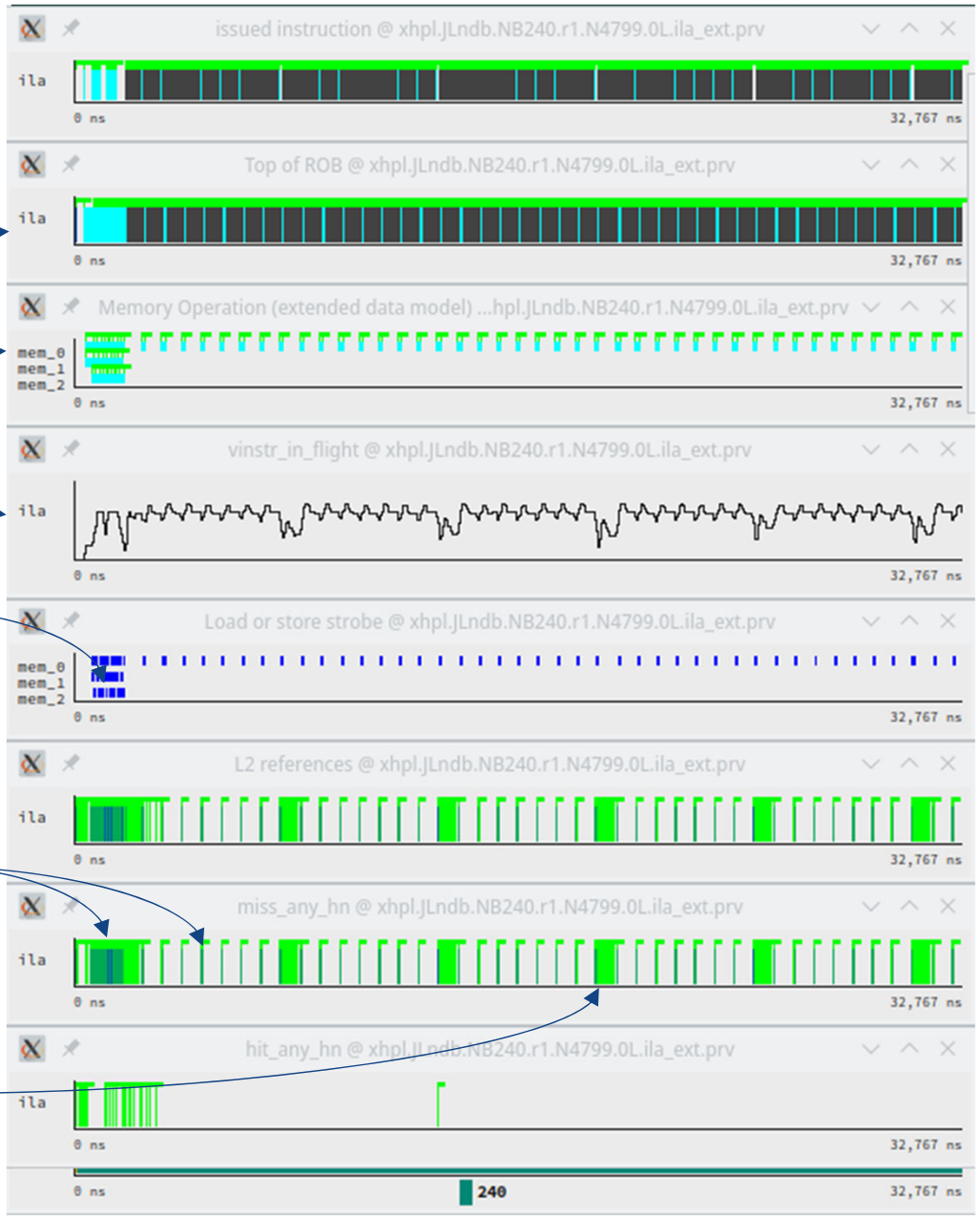
Barcelona Supercomputing Center
Centro Nacional de Supercomputación

EXCELENCIA SEVERO OCHOA

DGEMM

DGEMM ILA traces

- 32K cycles



Instruction at top of ROB

Execution of memory instr.

Instructions in flight

Data to VPU
High memory BW ?(% of peak)

Vector loads: always miss

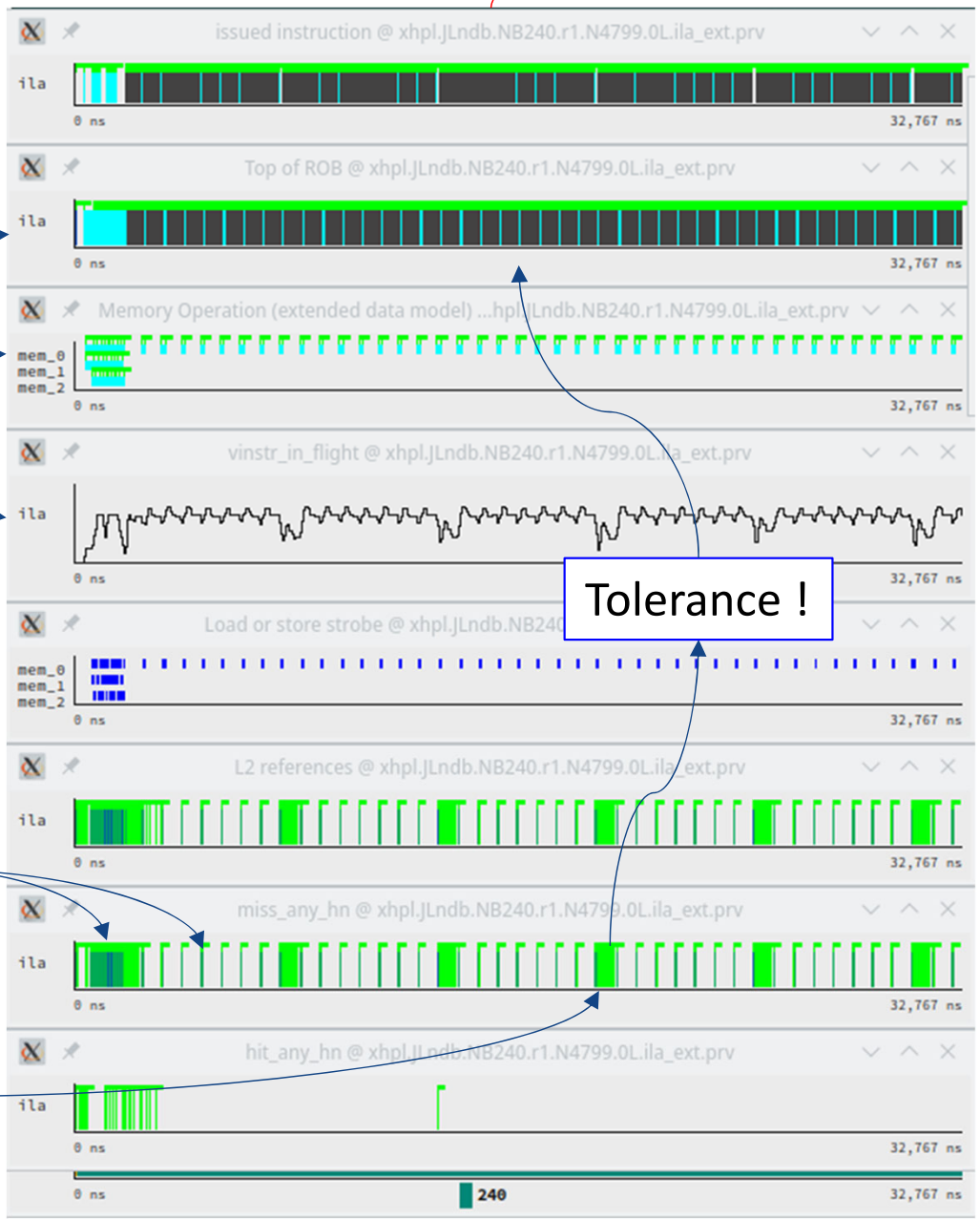
Periodic scalar misses

DGEMM ILA traces

- 32K cycles

→ 98% of peak !!

All good?



value 0
vle
vfmacc

Instruction at top of ROB

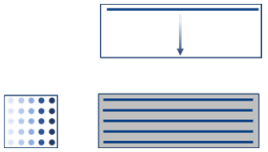
Execution of memory instr.

Instructions in flight

Tolerance !

Vector loads: always miss

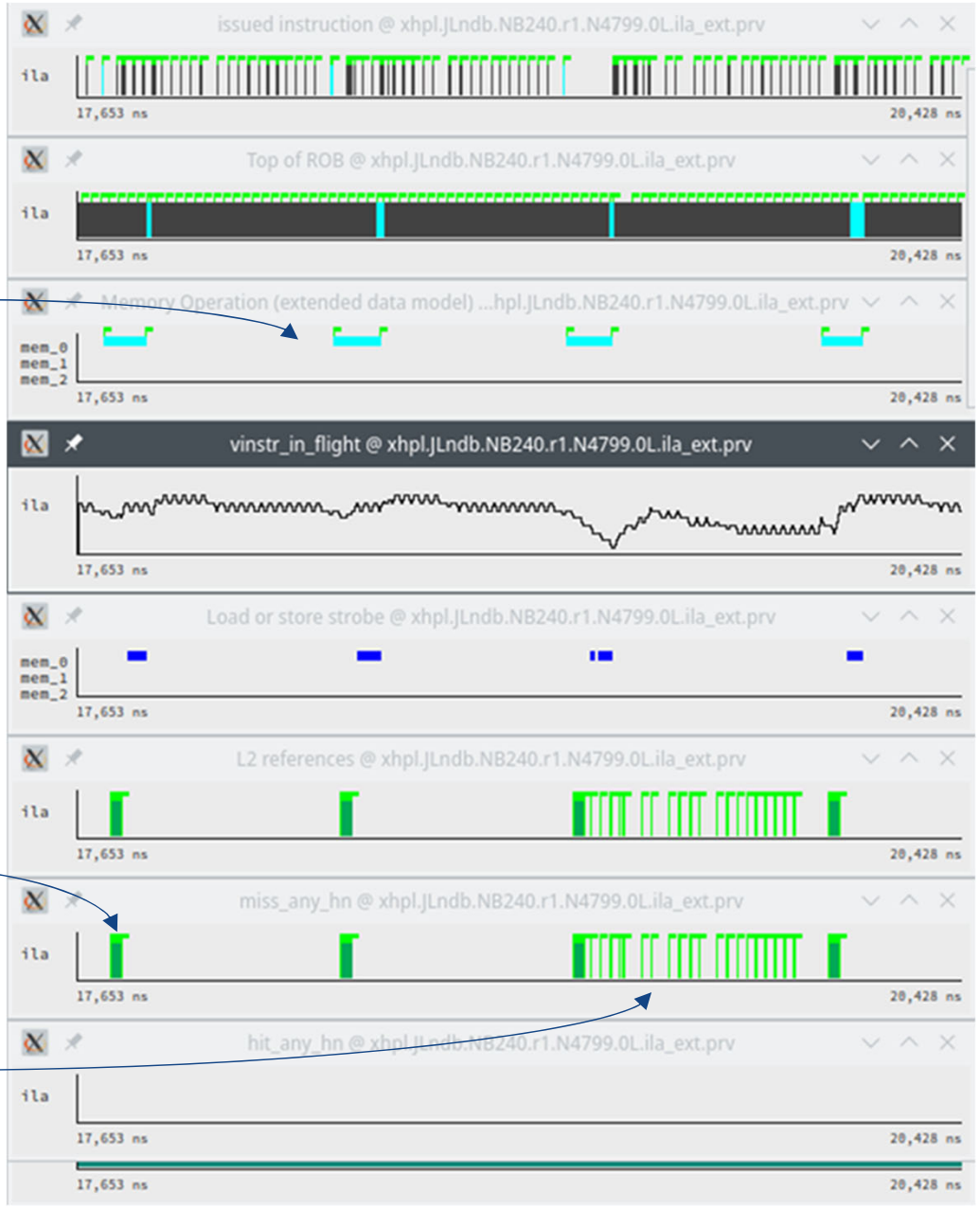
Periodic scalar misses



Detail

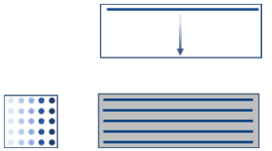
- Zoom

Load not issued till very late …
… still in time to overlap, but ..



Vector loads: always miss

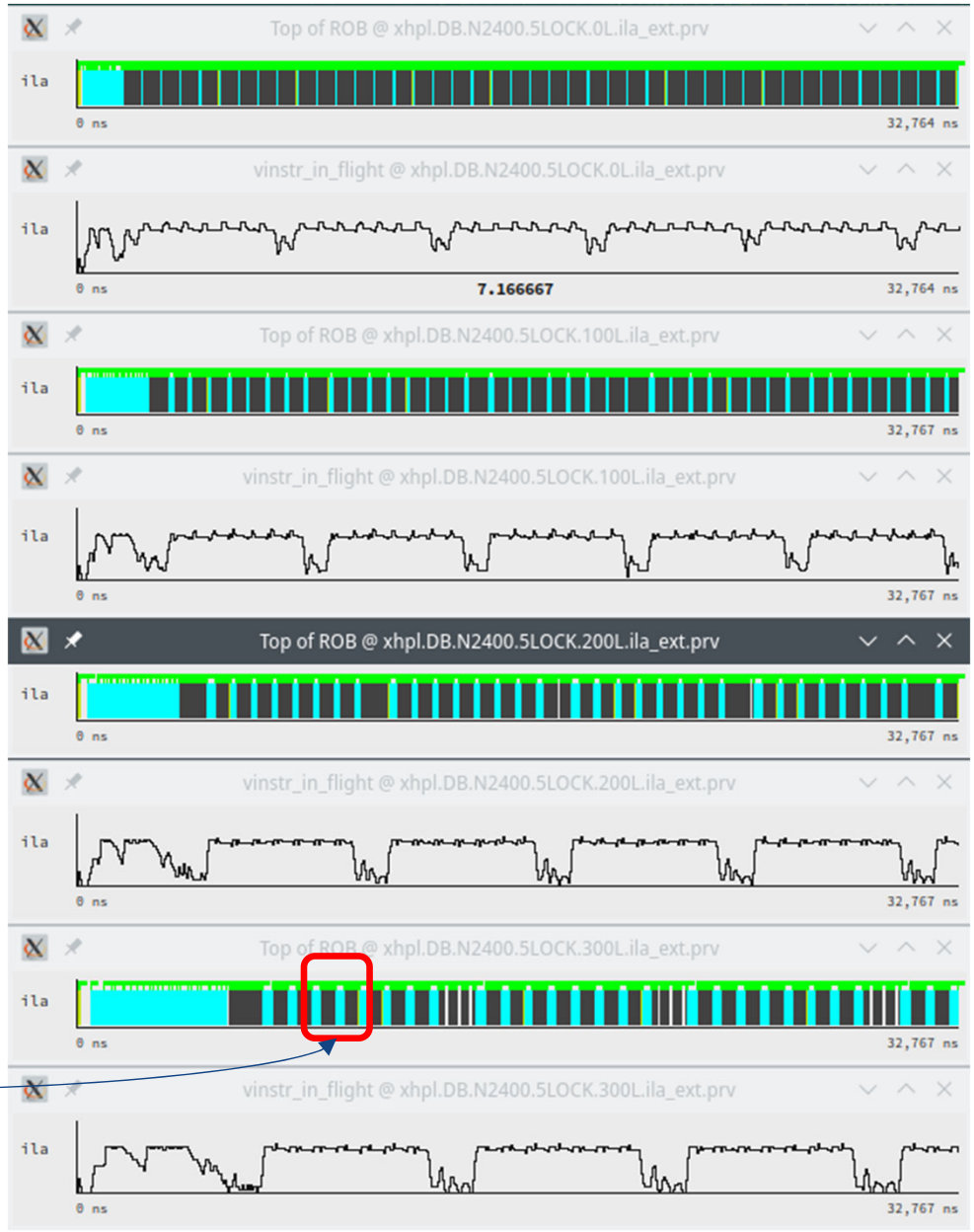
Periodic scalar misses



Parametric sweeps

- Understand impact of
  - architectural parameters
  - “environment”

- Impact of memory latency
  - + 0 cycles
  - +100 cycles
  - +200 cycles
  - +300 cycles



That bad?

Overlap memory - arithmetic

- +300 cycles

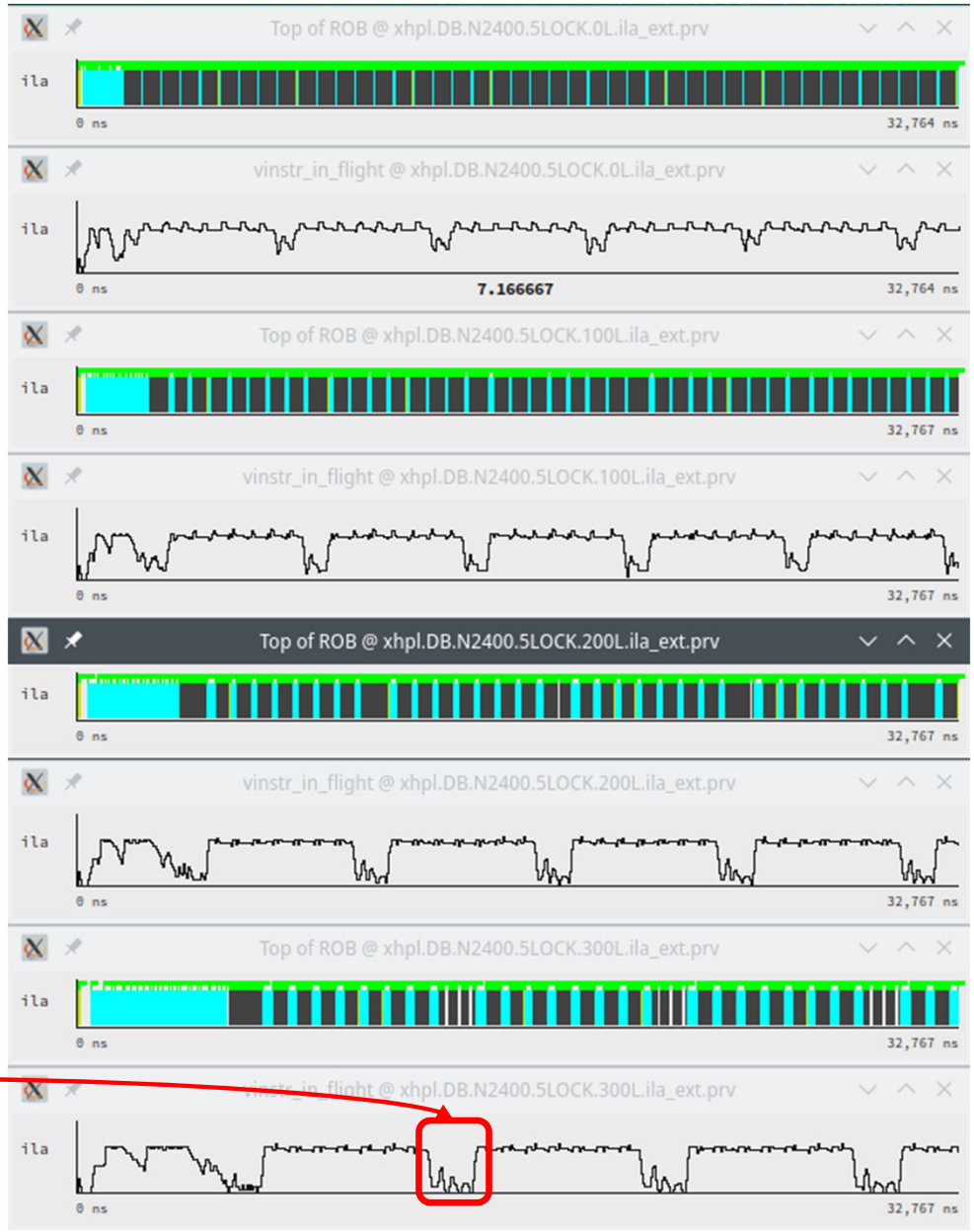
That good?

Fundamental reason of late issue ?

Parametric sweeps

- Understand impact of
  - architectural parameters
  - “environment”

- Impact of memory latency
  - + 0 cycles
  - +100 cycles
  - +200 cycles
  - +300 cycles

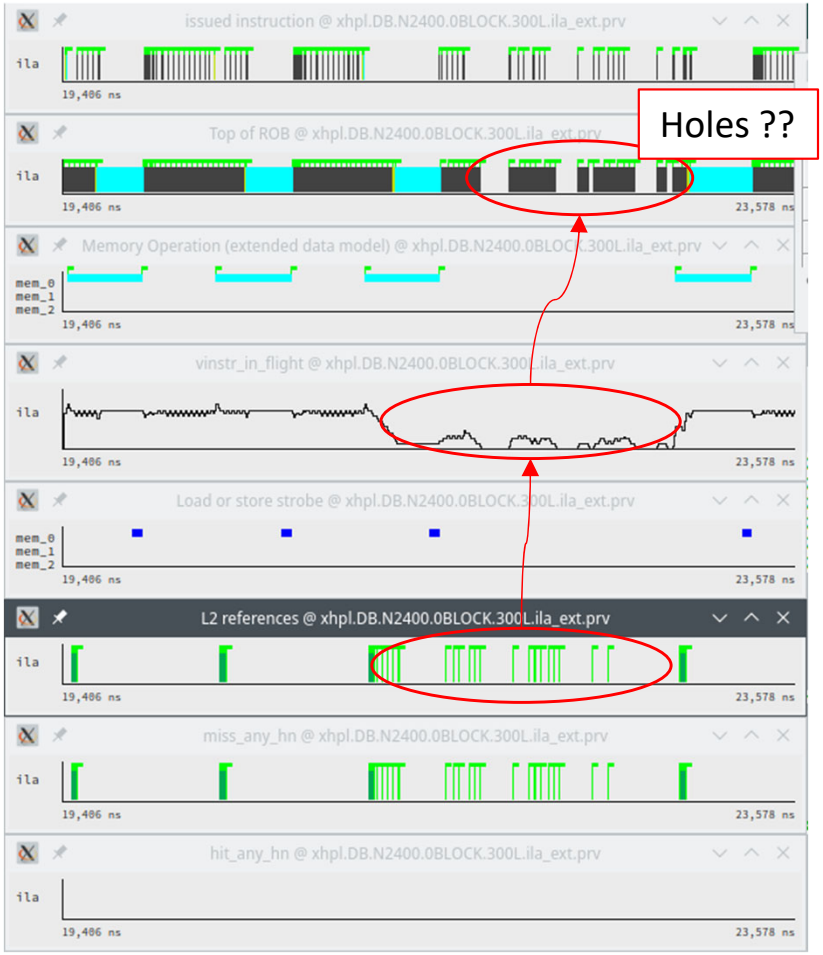


Why bad?

Vector scalar interactions

- +300 cycles

Ordering between scalar and vector accesses ?

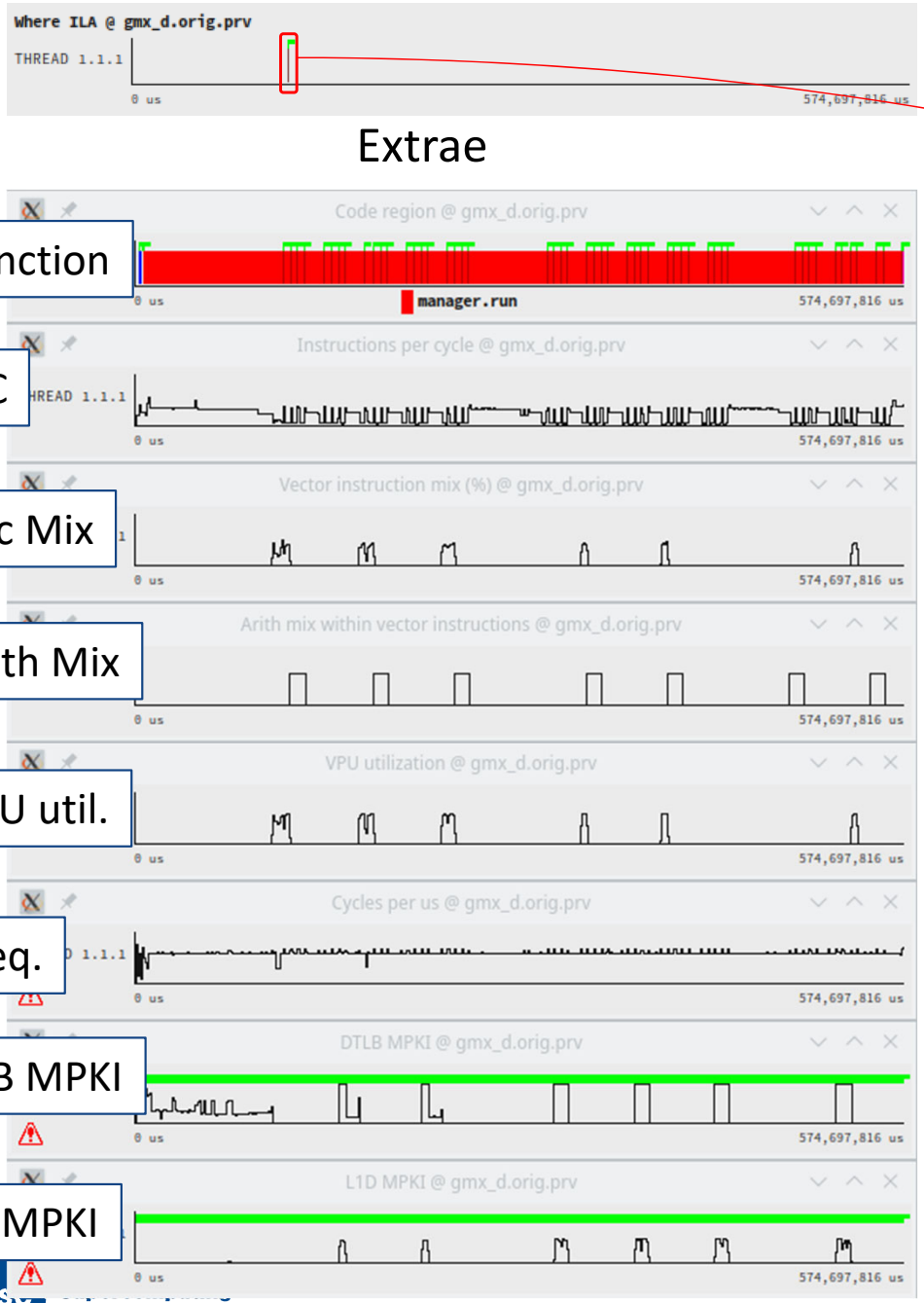


Barcelona Supercomputing Center
Centro Nacional de Supercomputación

EXCELENCIA SEVERO OCHOA

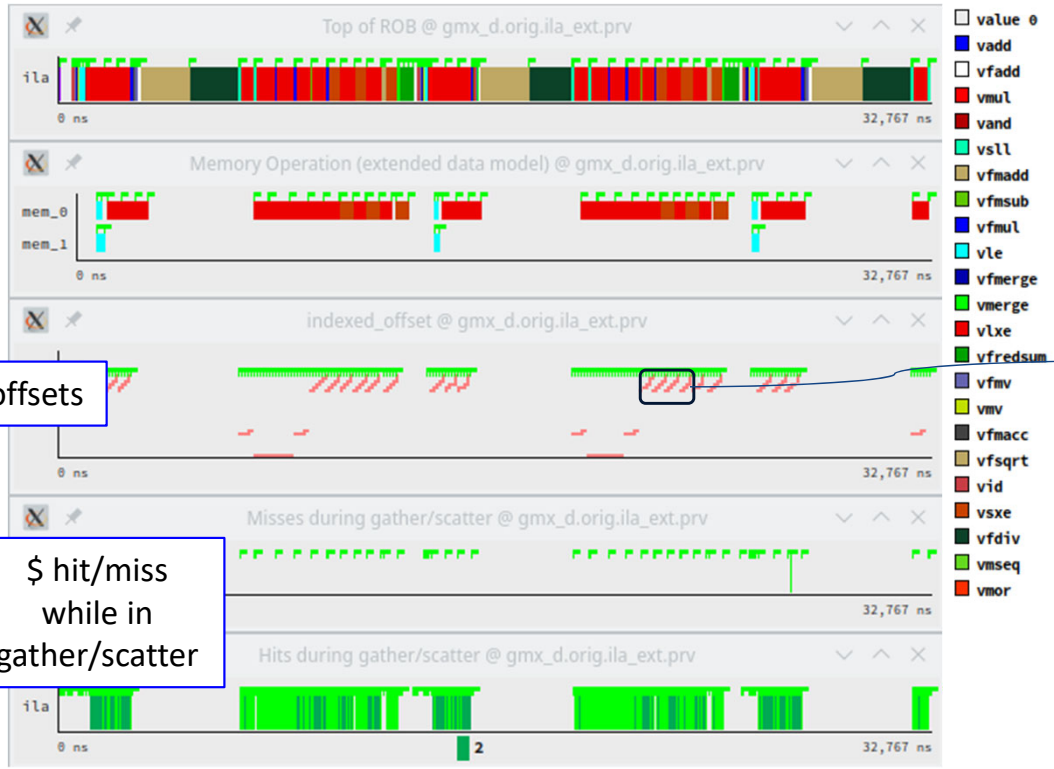
GROMACS

Dynamic range



Address patterns

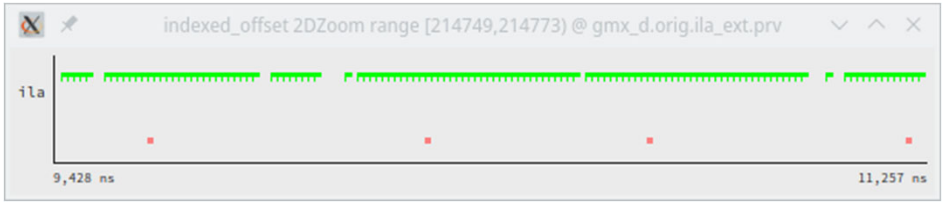
- Gather/Scatter offsets

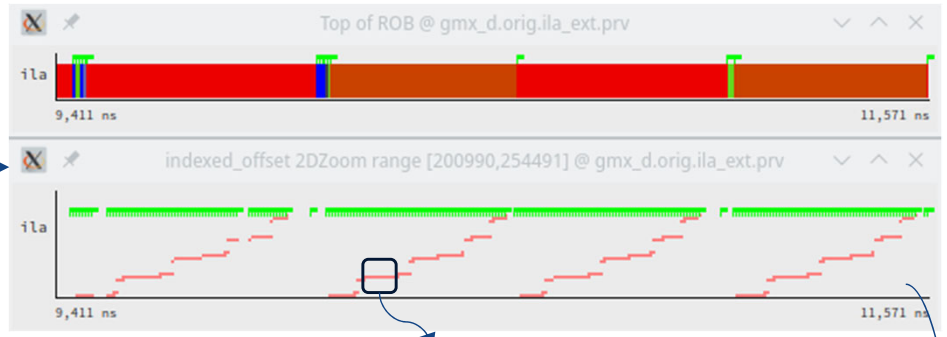


offsets

$ hit/miss while in gather/scatter

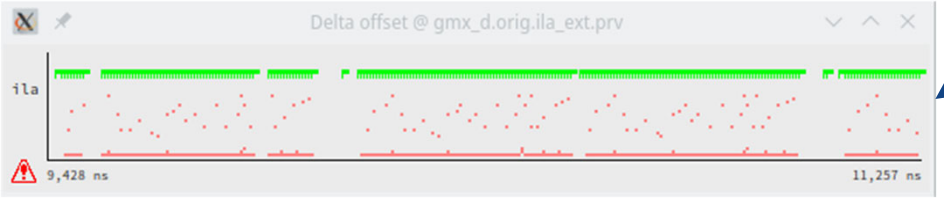
Same offset across gathers/scatters?





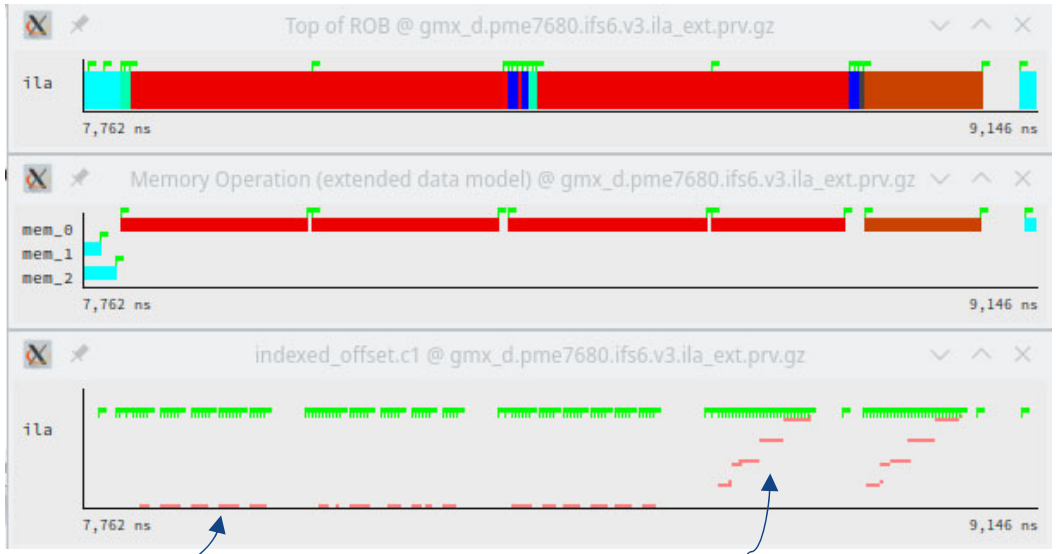
Stride?

Spatial locality?



Gathers

- Very frequent



Very few different offsets

Bursts of "consecutive" accesses

Microarchitectural optimizations

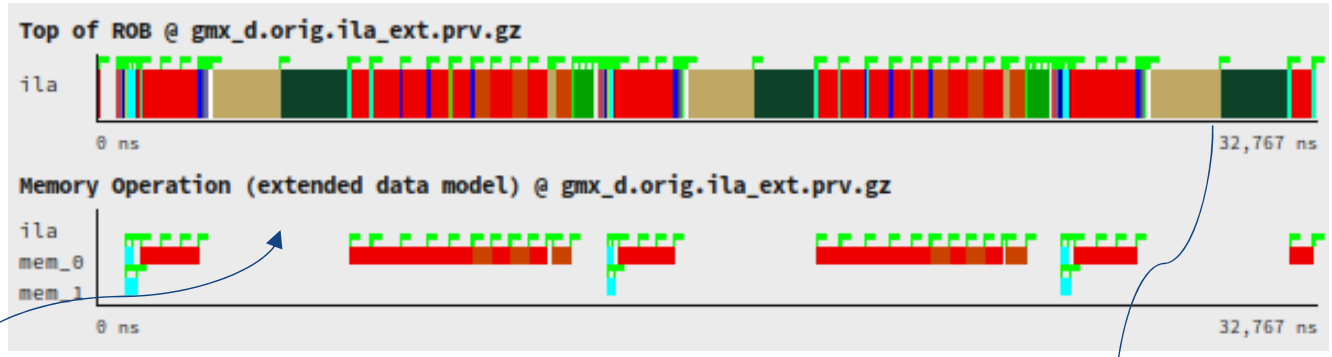
Actually stride 3 !!

AoS → SoA
Already implemented

- Detected implementation issues fixed for VEC

vfdiv & vsqrt

- Long latency … even if still doing many ops

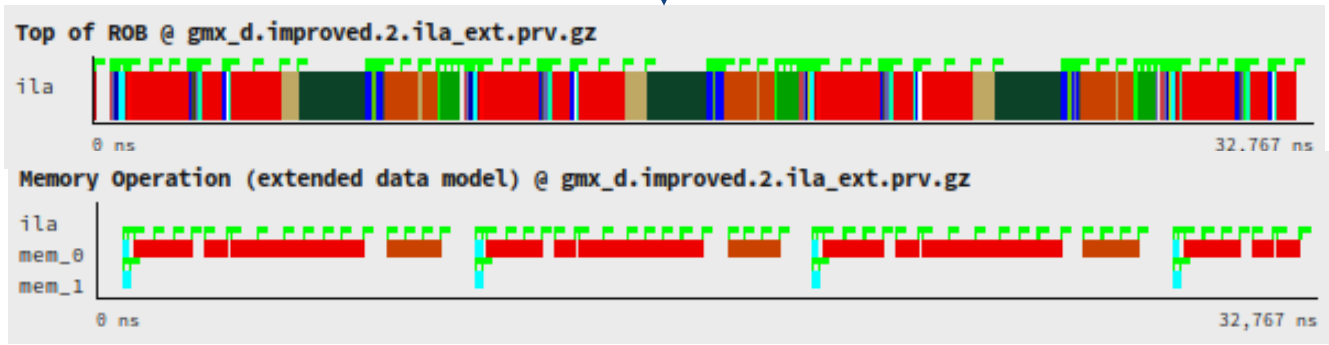


No overlap between arithmetic and memory ops

1/x and 1/sqrt(x) functional units
Already implemented for VEC

Already supported in source code

Instruction scheduling potential?
Manual core refactoring to achieve effect in RVV 0.7
Already implemented in compiler for RVV 1.0



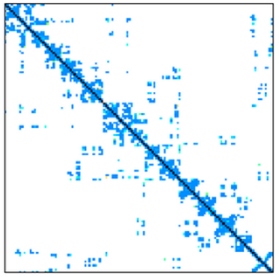
Barcelona Supercomputing Center
Centro Nacional de Supercomputación

EXCELENCIA SEVERO OCHOA

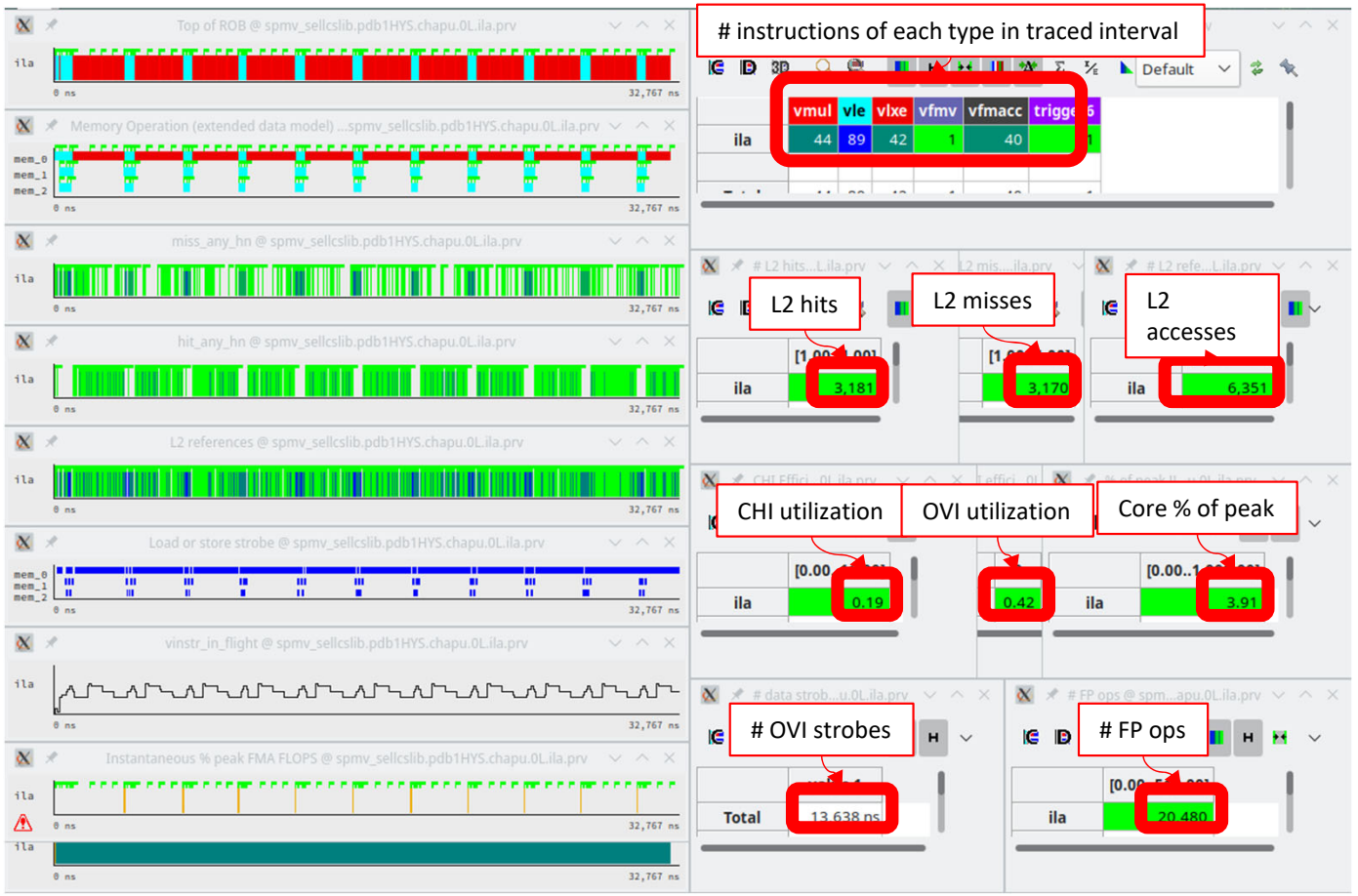
SpMV

SpMV

- pdb1HYS: Protein
- 2008
- nr = 36417
- nnz = 4344765



Within vlxe
L2 hit ratio: 3109/3413 = 0.91
elems/req. line 10587/3413 = 3.10



C. Gomez et al. "Efficiently running SpMV on long vector architectures" PPoPP21

Offsets (line)

cage12

GA19As19H42

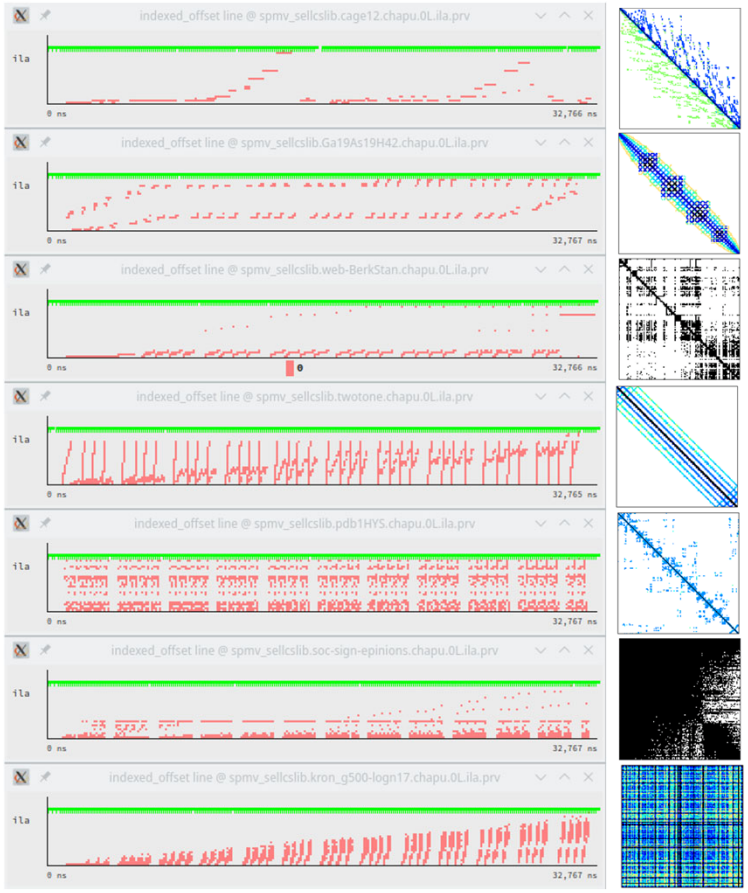
Web-BerkStan

twotone

pdb1HYS

Soc-sign-epinions

Kron_g500-logn17



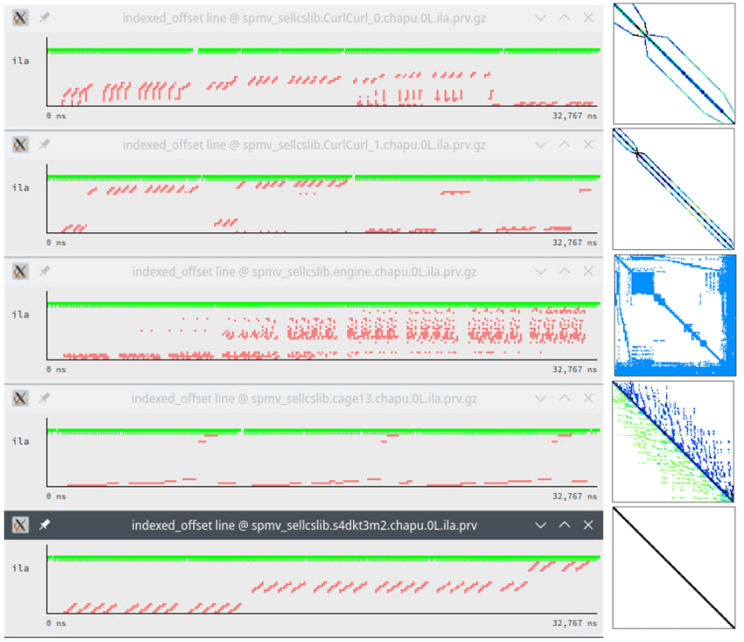
CurlCurl_0

CurlCurl_1

engine

cage13

s4dkt3m2



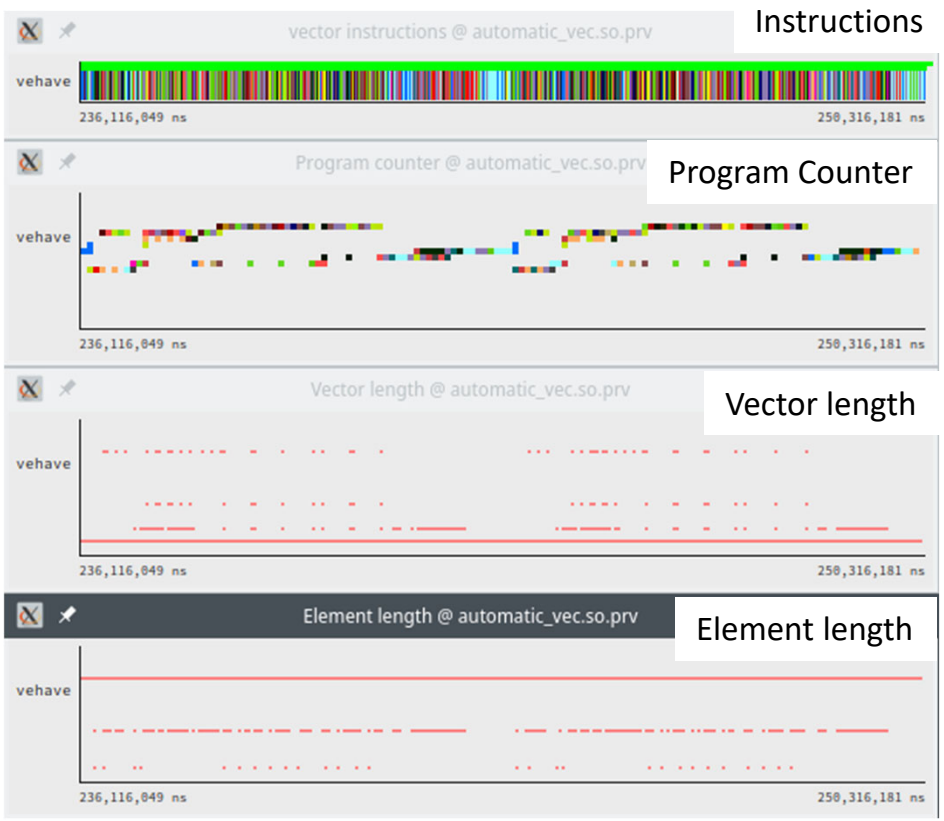
Barcelona Supercomputing Center
Centro Nacional de Supercomputación

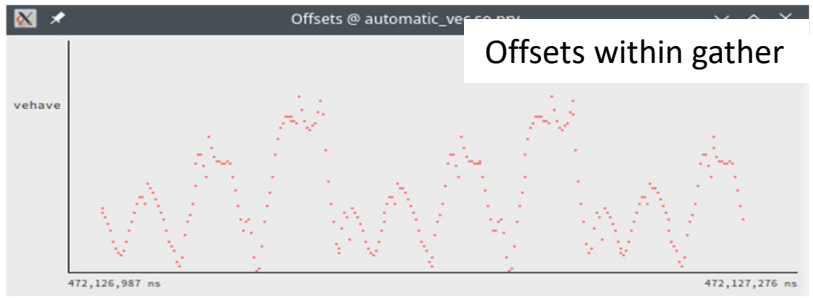
EXCELENCIA SEVERO OCHOA

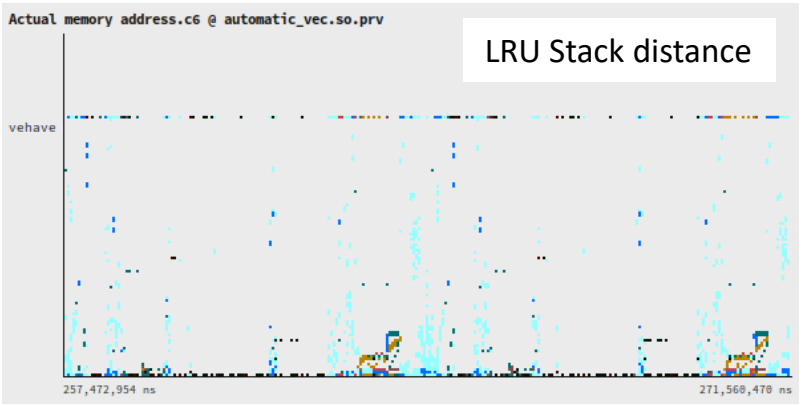
Software emulators

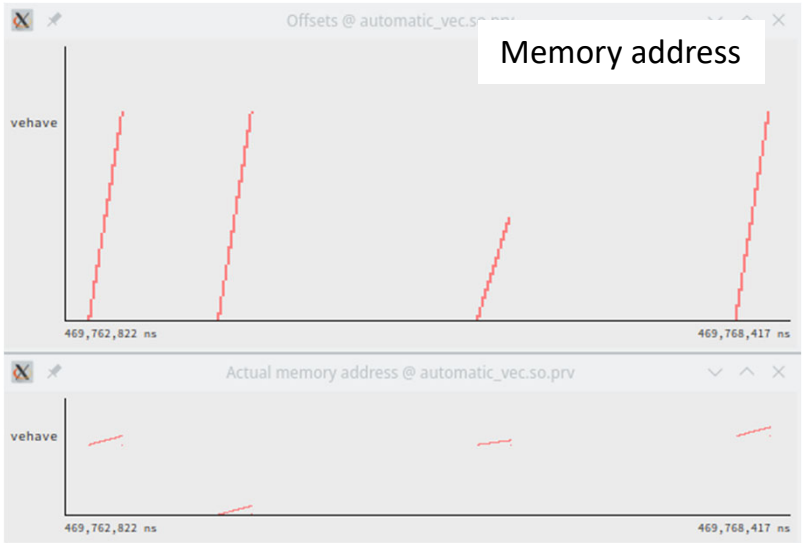
Vehave traces

- EsiWace climate code dwarfs
- Views:
  - Dynamic Instructions traces (sequence)
  - And parameters (regs, @, VL, data type, …)



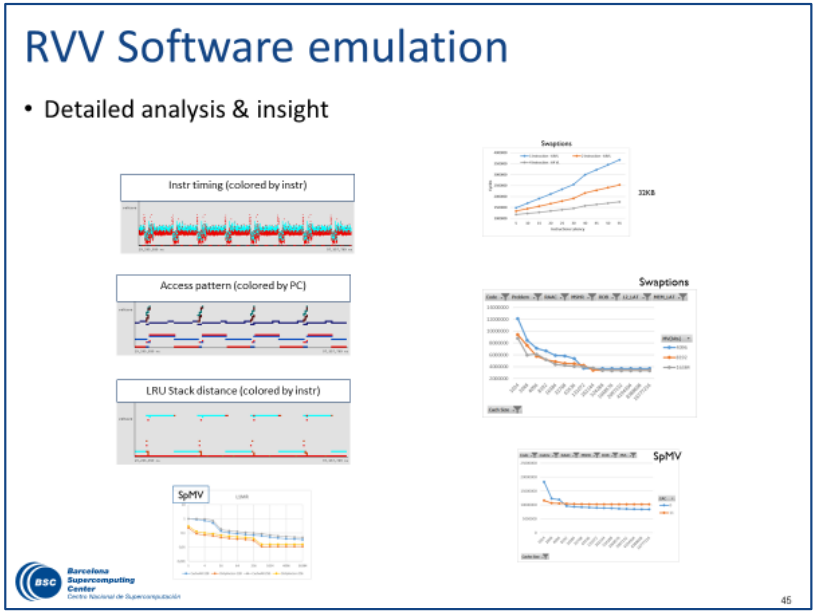


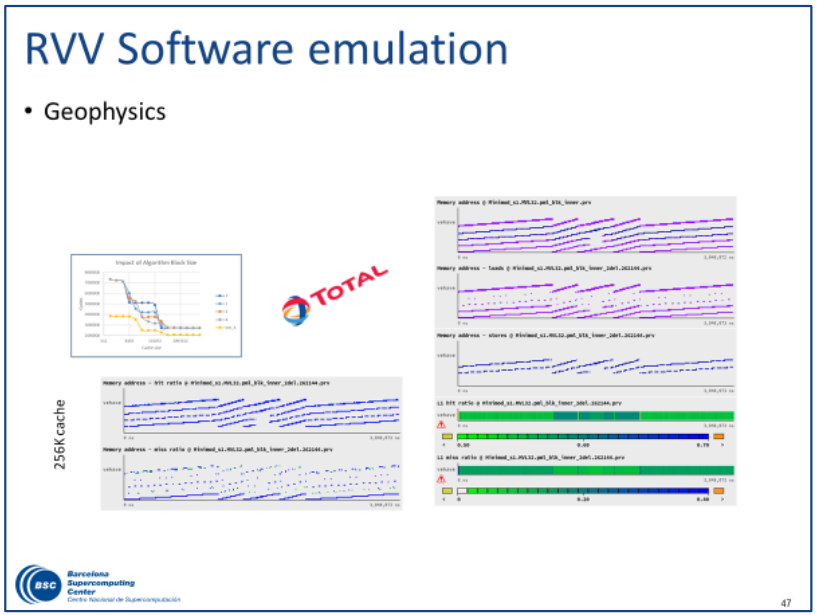




Architectural simulation

- Impact of …
  - Instruction latencies, ROB, Cache sizes, Memory latency, MAXVL, …
- Global metrics & **detail**
  - When are instructions issued, completed, graduated
  - Individual hits & misses





F. Martinez et al. "Exploiting Vector Code Semantics for Efficient Data Cache Prefetching", ICS 2024

V. Le Fèvre & M. Casas. "Efficient Execution of SpGEMM on Long Vector Architectures", HPDC 2023

RAVE

- QEMU plugin
- Emits information such as
  - Profile
  - Paraver trace
    - Instructions sequence (vector and summarized scalar) and
    - And parameters (regs, @, VL, data type, …)
    - Link to source
      - Manual instrumentation
      - Call stack

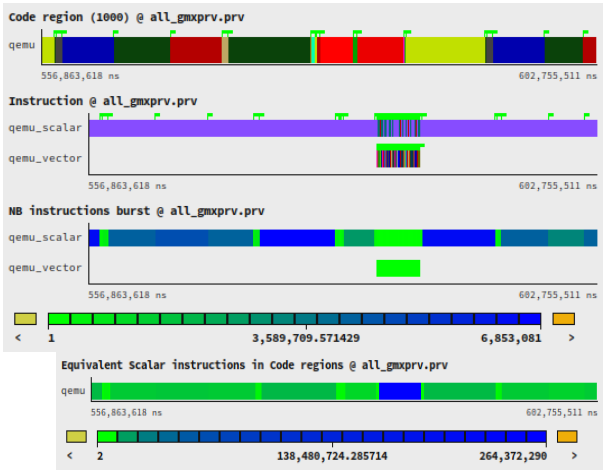
P. Vizcaino et al. “RAVE: RISC-V Analyzer of Vector Executions, a QEMU tracing plugin”, PPAM 2024 workshop on RISC-V

P. Vizcaino. “SDV tutorial”, https://repo.hca.bsc.es/gitlab/epi-public/risc-v-vector-simulation-environment/-/wikis/SDV-Vector-Analysis-Tutorial

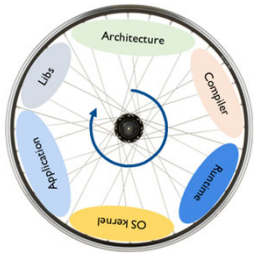
P. Vizcaino. “RAVE tutorial”, https://repo.hca.bsc.es/gitlab/epi-public/risc-v-vector-simulation-environment/-/wikis/Vector-emulation-with-RAVE

Gromacs

- Large runs !!!
  - 10s millions of instructions
  - 100s millions operations
- VL variation
- Number of scalar instructions btw vector



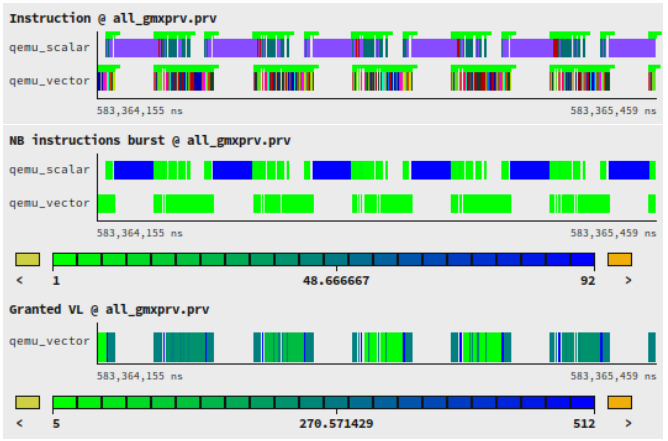
1 256



Microarchitectural optimizations

Compiler optimizations

Application optimizations



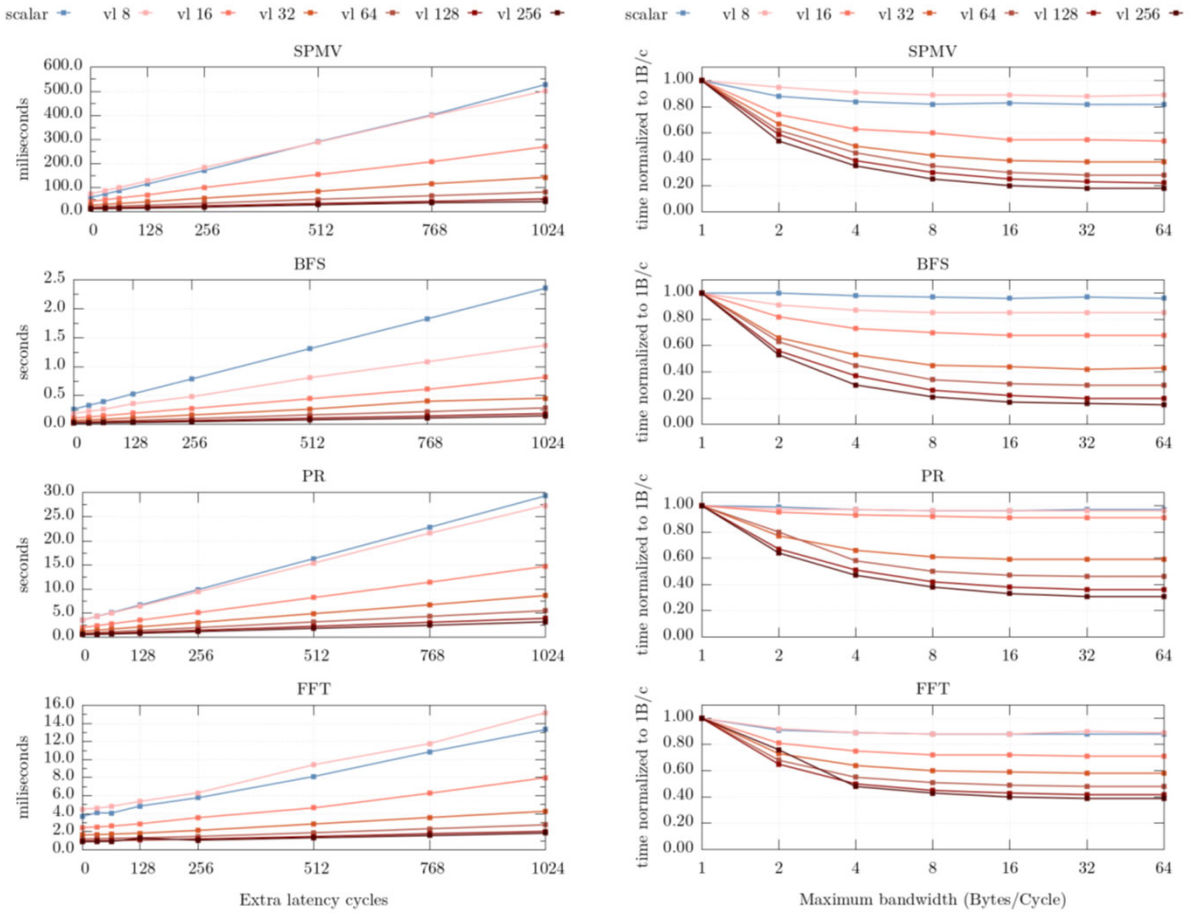
Barcelona Supercomputing Center
Centro Nacional de Supercomputación

EXCELENCIA SEVERO OCHOA

RVV in HPC and beyond

RVV in HPC and beyond

- Impact of latency on
  - FFT
  - SpMV
  - Graph algorithms



P. Vizcaino et al., "Short reasons for long vectors in HPC CPUs: a study based on RISC-V" RV workshop. SC23

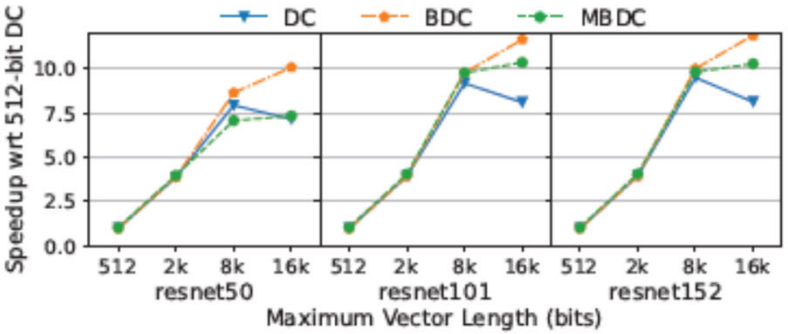
P. Vizcaino et al. "Acceleration with long vector architectures: implementation and evaluation of the FFT kernel on NEC SX-Aurora and RISC-V vector extension", Heteropar 2021

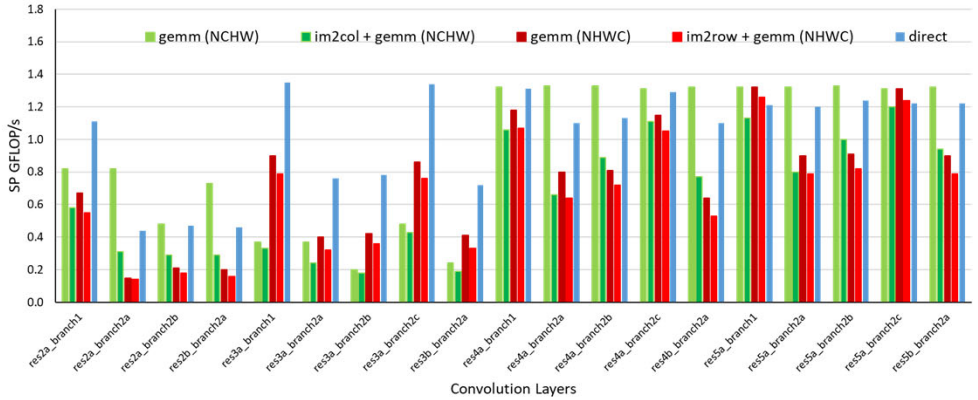
P. Vizcaino et al. "Graph Computing on Long Vector architectures (Yes, it works!)", IPDPS24 - PDSEC Workshop

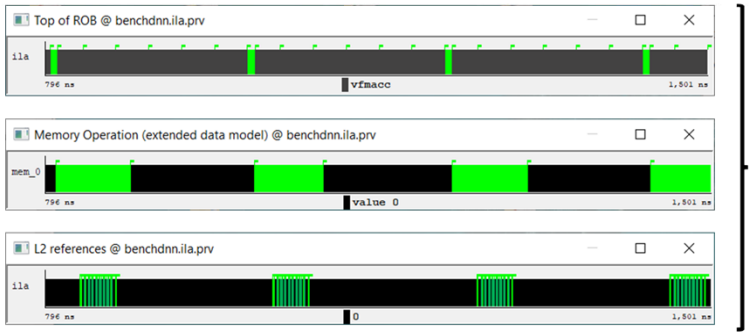
M. Blancafort et al. "Exploiting long vectors with a CFD code: a co-design show case", IPDPS 2024

RVV in HPC and beyond

- oneDNN @ RVV
  - Exploiting long vectors







Algorithm 2 The SIMD convolution on long vector machines

Input: Source Activation Tensor (S), Weights Tensor (W)
Output: Destination Activation Tensor (D)
Architectural variables: N_{vlen}, N_{vregs}

1: $OC_b = min(OC, N_{vlen})$
2: $IC_b = min(IC, N_{vlen})$
3: $RB_w = min(N_{vregs} - 1, OW)$
4: $RB_h = max((N_{vregs} - (1 + RB_w))/OH, 1)$
5: $vl = min(OC, N_{vlen})$ ▷ Vector Length
6: **for** $n = 0, N$ **do**
7: **for** $oc = 0, OC/OC_b$ **do**
8: **for** $ic = 0, IC/IC_b$ **do**
9: **for** $oh = 0, OH/RB_h$ **do**
10: **for** $ow = 0, OW/RB_w$ **do**
11: **for** $(h, w) = (0 : RB_h, 0 : RB_w)$ **do** ▷ Unrolled
12: $vo_{h,w} = vload(D[n][oc][oh + h][ow + w][0], vl)$
13: **for** $(kh, kw, i) = (0 : KH, 0 : KW, 0 : IC_b)$ **do**
14: $vw = vload(W[oc][ic][kh][kw][i][0], vl)$
15: **for** $(h, w) = (0 : RB_h, 0 : RB_w)$ **do** ▷ Unrolled
16: $vi_{h,w} = S[n][ic][oh + kh + h][ow + kw + w][i]$
17: $vo_{h,w} = vfma(vo_{h,w}, vi_{h,w}, vw, vl)$
18: **for** $(h, w) = (0 : RB_h, 0 : RB_w)$ **do** ▷ Unrolled
19: $vstore(vo_{h,w}, D[n][oc][oh + h][ow + w][0], vl)$

A. De Limas et al., "Efficient Direct Convolution Using Long SIMD Instructions" PPoPP23

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

EXCELENCIA SEVERO OCHOA

To conclude …

Prof. Jesús Labarta

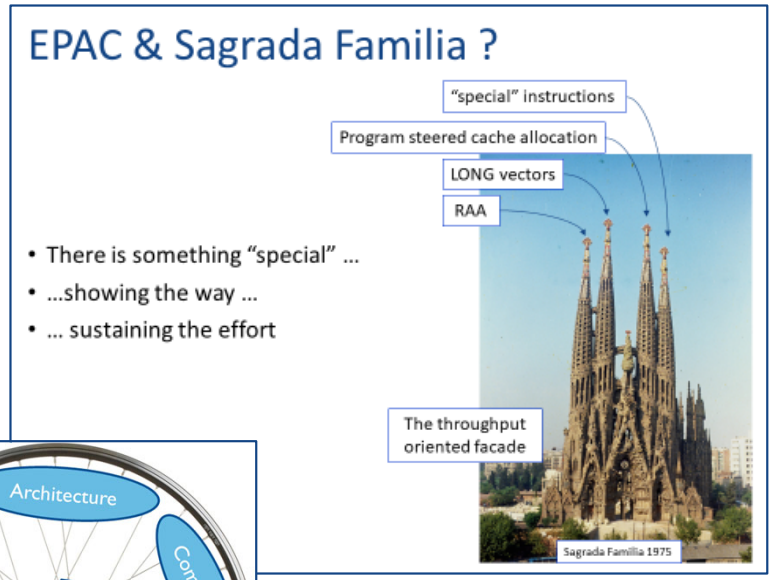
BSC & UPC

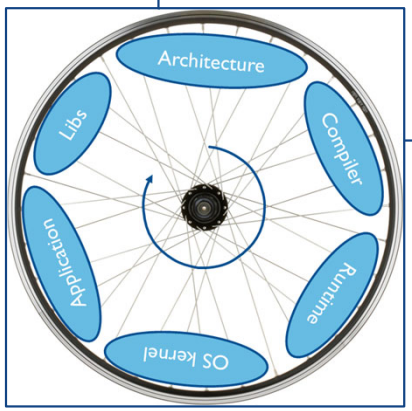
PPAM 2024

Ostrava, September 10th 2024

The importance of a vision

- Holistic throughput oriented vision
  - Based on **long vectors**
  - Programming productivity
- Emulators for all !!
  - Actual design evaluation
  - “In vivo” evaluations
  - Reduce “co-design” cycle length
- Performance analysis
  - Qualitative and detailed analytics
  - Dynamic range !!





RISC-V

European Processor Initiative

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

EXCELENCIA SEVERO OCHOA

Thanks