Gvozden Nešković

Vectorized ZFS* RAIDZ Implementation

LUG 2016, Portland

neskovic@compeng.uni-frankfurt.de

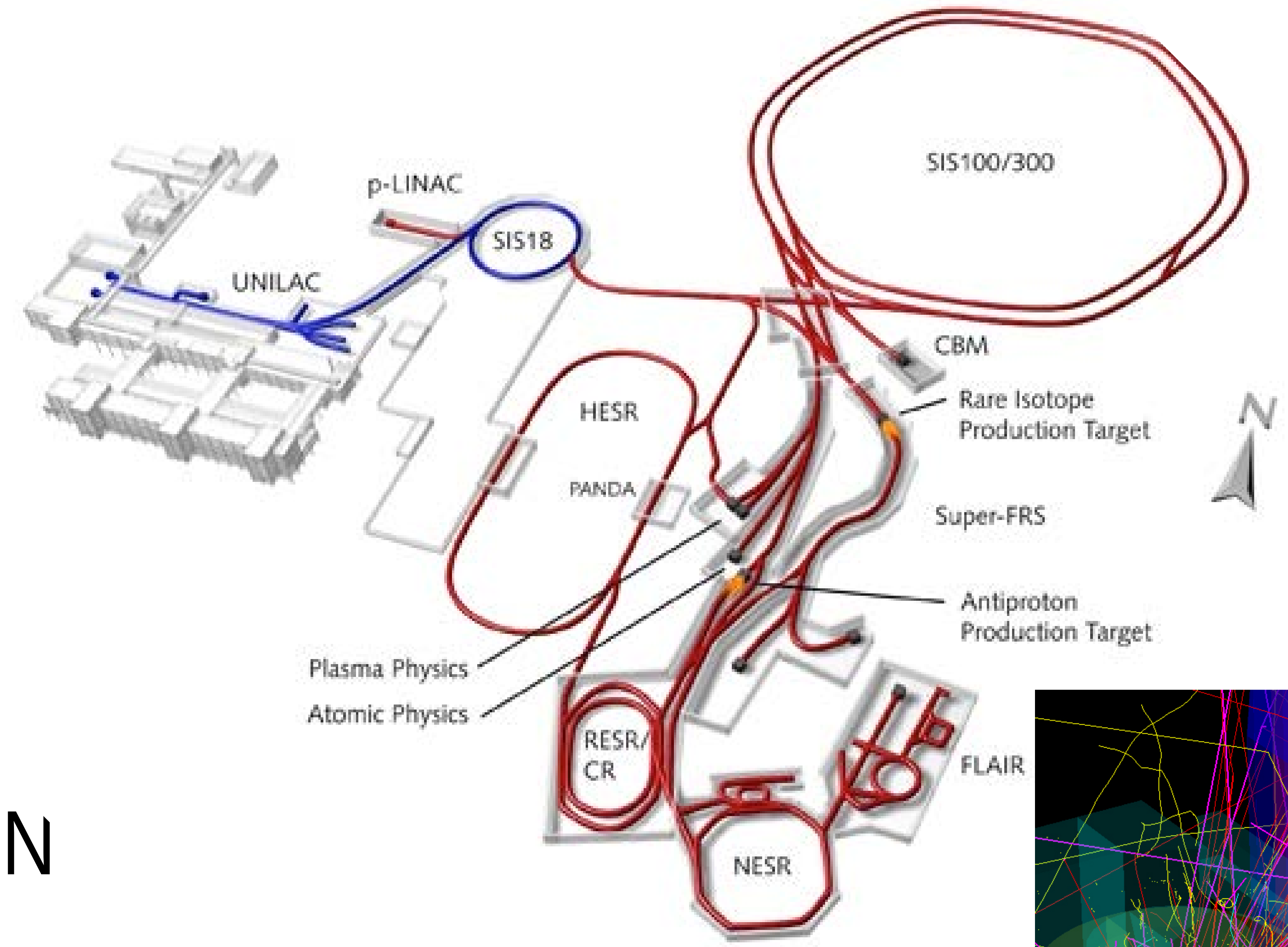
GOETHE UNIVERSITÄT FRANKFURT AM MAIN

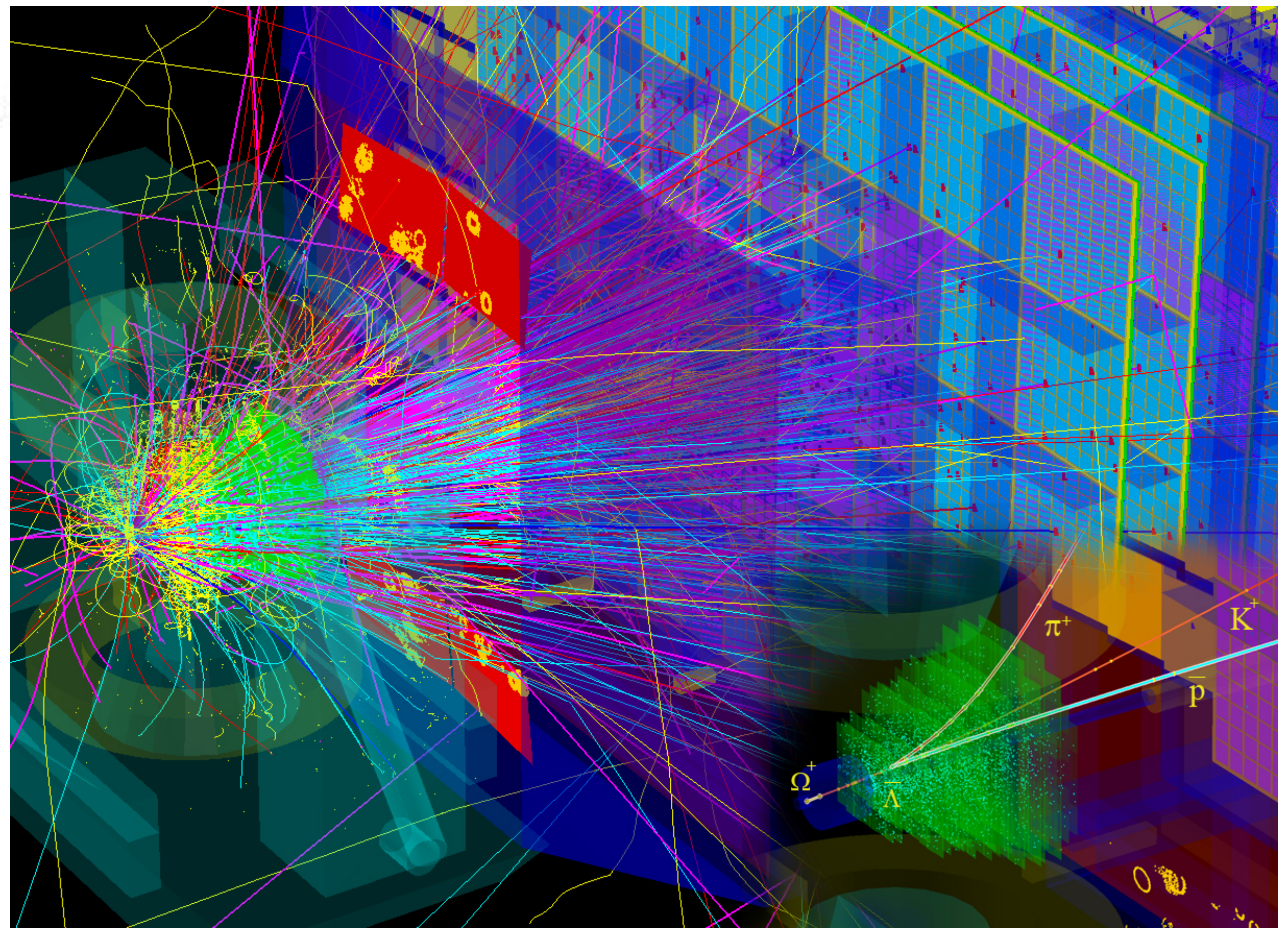
FIAS Frankfurt Institute for Advanced Studies

* All trademarks are the property of their respective owners.

FAIR/GSI

- Facility for Antiproton and Ion Research
  - Linear and Ring particle accelerators
  - Heavy Ion Experiments
  - Medical irradiation facility for cancer therapy
  - Participation in the ALICE experiment at CERN
- HPC at FAIR/GSI
  - Green IT Cube data center
  - Compute clusters:
    - Prometheus (~9000 cores, QDR IB)
    - **Kronos** (~8000 cores, FDR IB)
  - Lustre storage clusters:
    - Hera (~7PB, Lustre 1.8)
    - **Nyx** (~7PB, 45 OSSs, Lustre 2.5 on ZFS)
    - ~1TB/s per experiment (10MHz event rate for CBM)





Images: GSI Helmholtzzentrum für Schwerionenforschung GmbH

Lustre* & ZFS Motivation

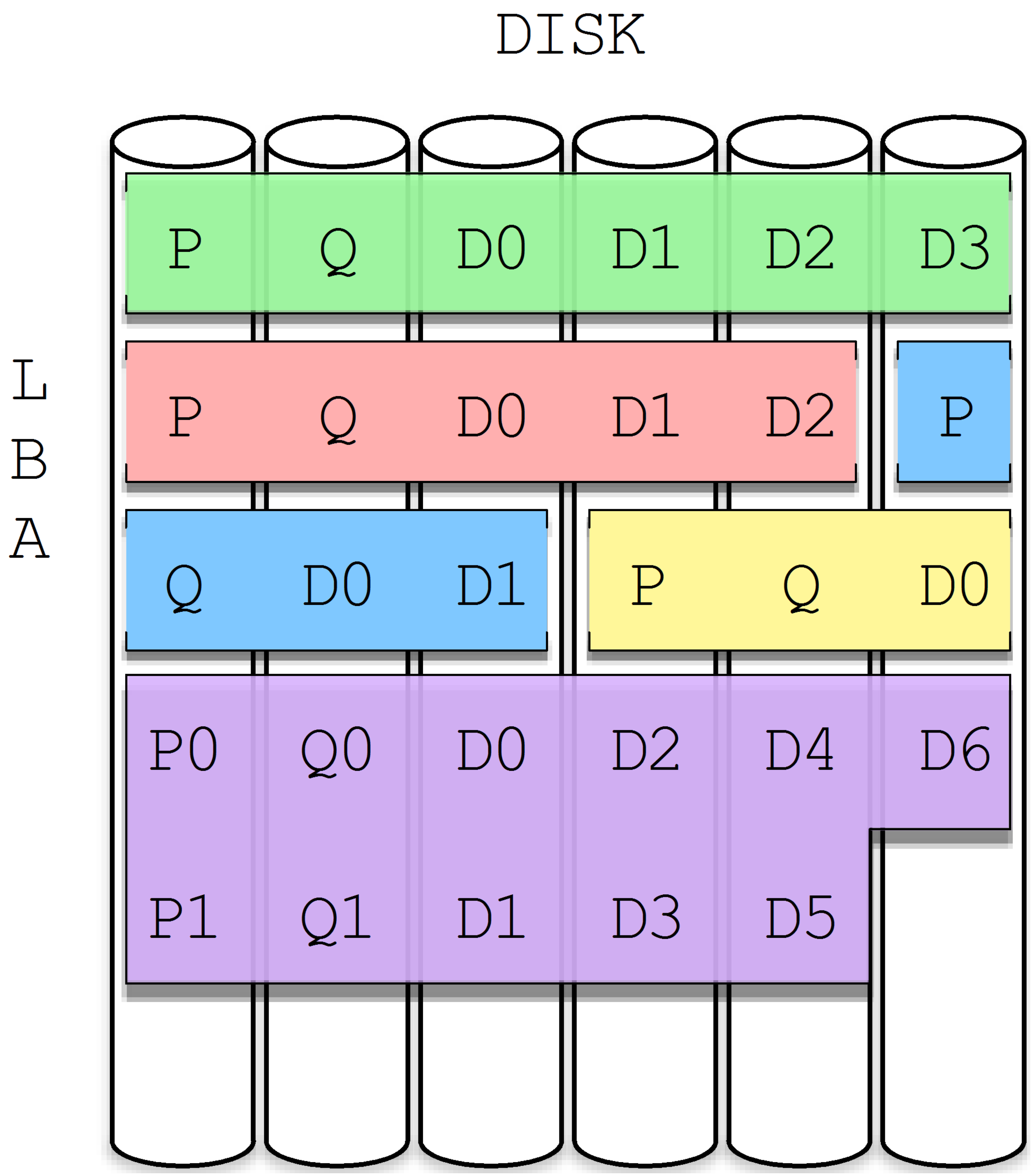
- Lustre on *ldiskfs*:
  - Version of *ext3/4*
  - Random writes limited by disk IOPS
  - Availability: long offline *fsck*
  - Reliability: hardware RAID controllers

- Lustre on ZFS:
  - ZFS Transactional Object Layer
  - Random writes limited by disk bandwidth (COW)
  - Data & metadata checksums, compression, snapshots
  - Availability: online *scrubbing/resilvering*
  - Reliability: Replication, RAIDZ1/2/3

ZFS RAIDZ levels

- ZFS volume management:
  - Striping
  - Mirroring
  - RAIDZ levels

- RAIDZ-1/2/3 levels:
  - Error Correction Erasure scheme
  - Specialized Reed-Solomon Codes
  - Advanced Block layout



RAIDZ Parity

- Properties:
  - Based on Galois field *GF[2^8]* generated with $p(x)=x^8+x^4+x^3+x^2+1$
  - Erasure code

$$P = D_0 \oplus D_1 \oplus \cdots \oplus D_n$$

$$Q = 2^0 \bullet D_0 \oplus 2^1 \bullet D_1 \oplus \cdots \oplus 2^n \bullet D_n$$

$$R = 4^0 \bullet D_0 \oplus 4^1 \bullet D_1 \oplus \cdots \oplus 4^n \bullet D_n$$

, where $2 \equiv X^1$, $4 \equiv X^2$

- Addition:
  - XOR operation
  - Efficient in scalar and vector

- Multiplication:
  - By *2* and *4*
  - By a ***constant***:
    - Using *log* and *exp* look-up tables
    - $c \bullet a = \exp\{ \log(c) + \log(a) \}$

RAIDZ Parity Generation

- RAIDZ-1 parity generation:
  - Simple *XOR* parity (P code only)

- RAIDZ-2/3 parity generation:
  - Transform the Q and R equations:

$$Q = D_0 \oplus 2 \bullet \left(D_1 \oplus \cdots \oplus 2 \bullet \left(D_{n-1} \oplus 2 \bullet D_n\right)\right)$$

$$R = D_0 \oplus 4 \bullet \left(D_1 \oplus \cdots \oplus 4 \bullet \left(D_{n-1} \oplus 4 \bullet D_n\right)\right)$$

  - RAIDZ-2 requires fast GF multiplication by *2* (PQ codes)
  - RAIDZ-3 requires fast GF multiplication by *2* and *4* (PQR codes)

RAIDZ Data Reconstruction

- Trivial when using only P parity

- Direct solving:
  - Solve parity equations (matrix inversion method)
  - Used in the original RAIDZ3 reconstruction
  - Requires *n* GF multiplications per *word*

$$D_x = x_p \bullet P \oplus x_q \bullet Q \oplus x_0 \bullet D_0 \oplus \cdots \oplus x_{n-2} \bullet D_{n-2}$$
$$D_y = y_p \bullet P \oplus y_q \bullet Q \oplus y_0 \bullet D_0 \oplus \cdots \oplus y_{n-2} \bullet D_{n-2}$$

- Solving using syndromes:
  - Used in the original RADIZ2 reconstruction
  - Syndromes calculated first
    - o Parity calculation with zeroed missing data
  - Requires *1* to *5 GF* multiplications per *word*

$$P = P_{xy} \oplus D_x \oplus D_y$$
$$Q = Q_{xy} \oplus 2^x \bullet D_x \oplus 2^y \bullet D_y$$

$$D_x = a \bullet (P \oplus P_{xy}) \oplus b \bullet (Q \oplus Q_{xy})$$
$$D_y = D_x \oplus (P \oplus P_{xy})$$

Vectorizing GF multiplication

- GF multiplication:

$$a \bullet b = (a \times b) \bmod p$$

$$a \bullet b = L(a \times b) \oplus M(a, b)$$

  - Sum of carry-less multiplication and modulo parts [1)]
  - Computed efficiently using two lookup-tables [2)]
    - SSE variant computes 16 multiplication in parallel
    - AVX2 variant computes 32 multiplication in parallel

GF operation	Scalar	SSE	AVX2
Addition	8	16	32
Multiplication by 2/4	8/4	16	32
Multiplication	1	16	32

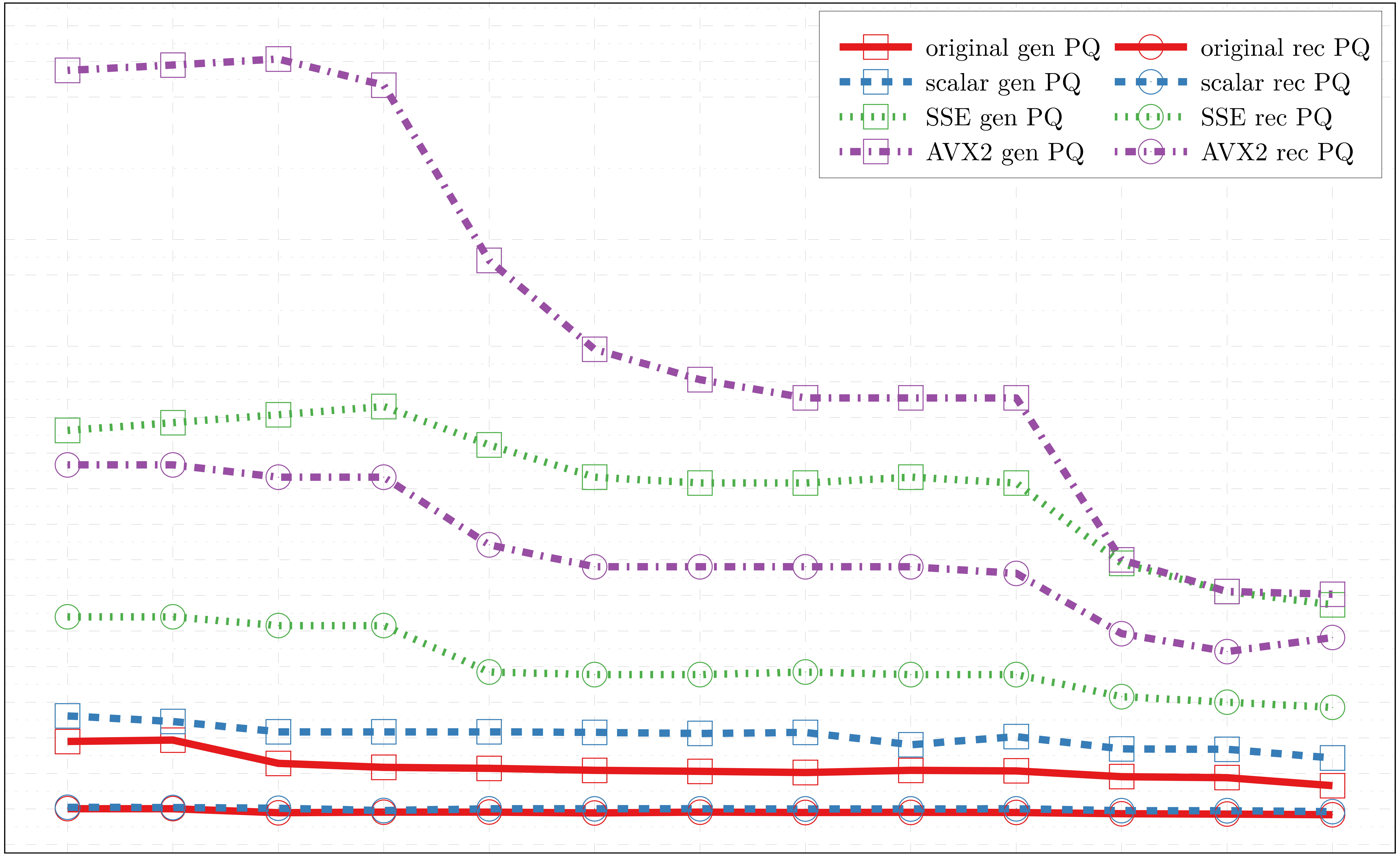
Word length in bytes for instruction set per operation

- Contributed implementations:
  - Scalar 32 and 64 bit
  - SSE 128bit
  - AVX2 256bit

1) "Intel* Carry-Less Multiplication Instruction and its Usage for Computing the GCM Mode"
2) "Optimizing Galois Field arithmetic for diverse processor architectures and Applications". K.Greenan et al. 2008

ZFS RAIDZ-2 Results

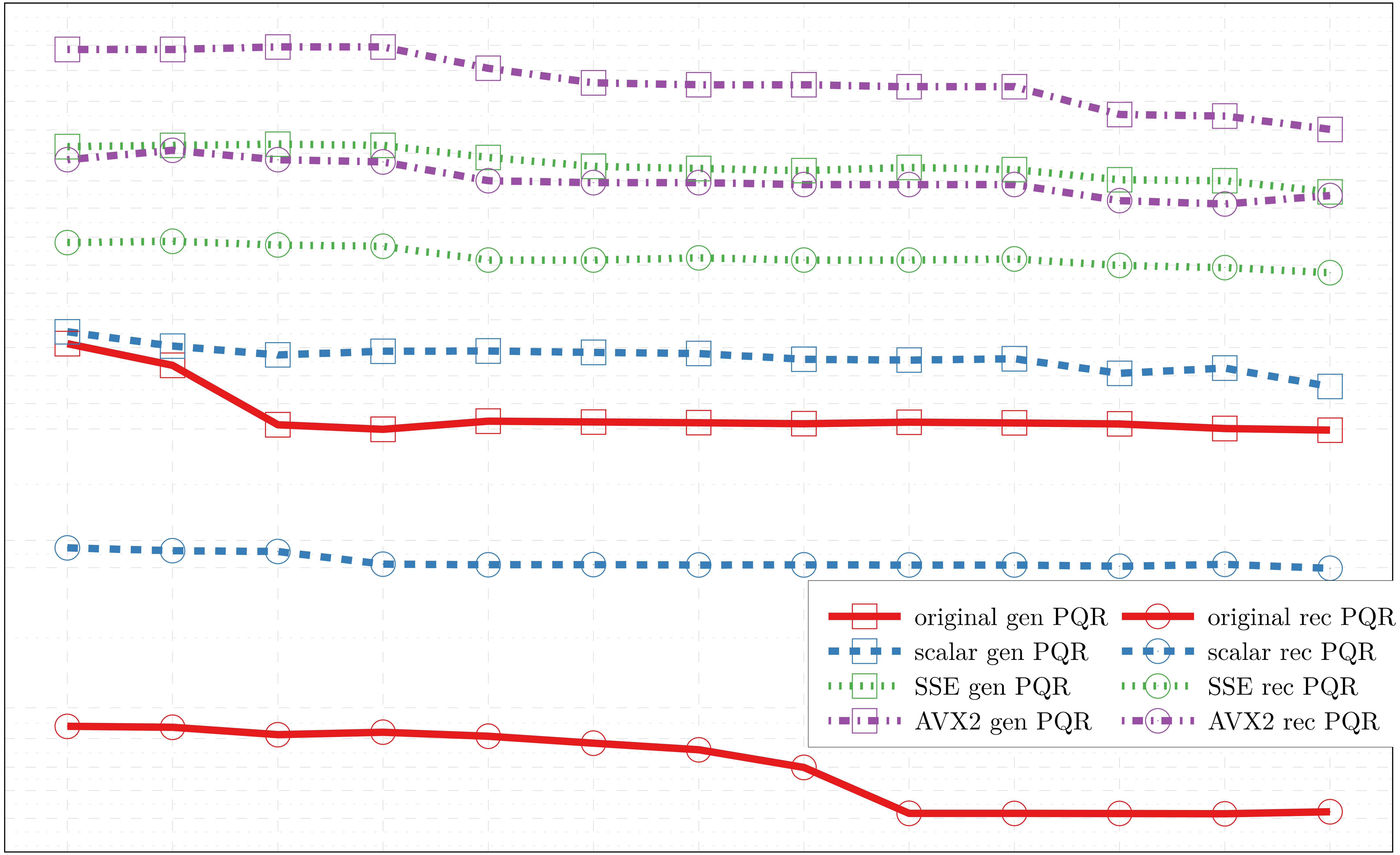
- RAIDZ-2
- 8 data disks
- 2 parity disks
- Generate PQ
- Reconstruct 2 disks



* All trademarks are the property of their respective owners.

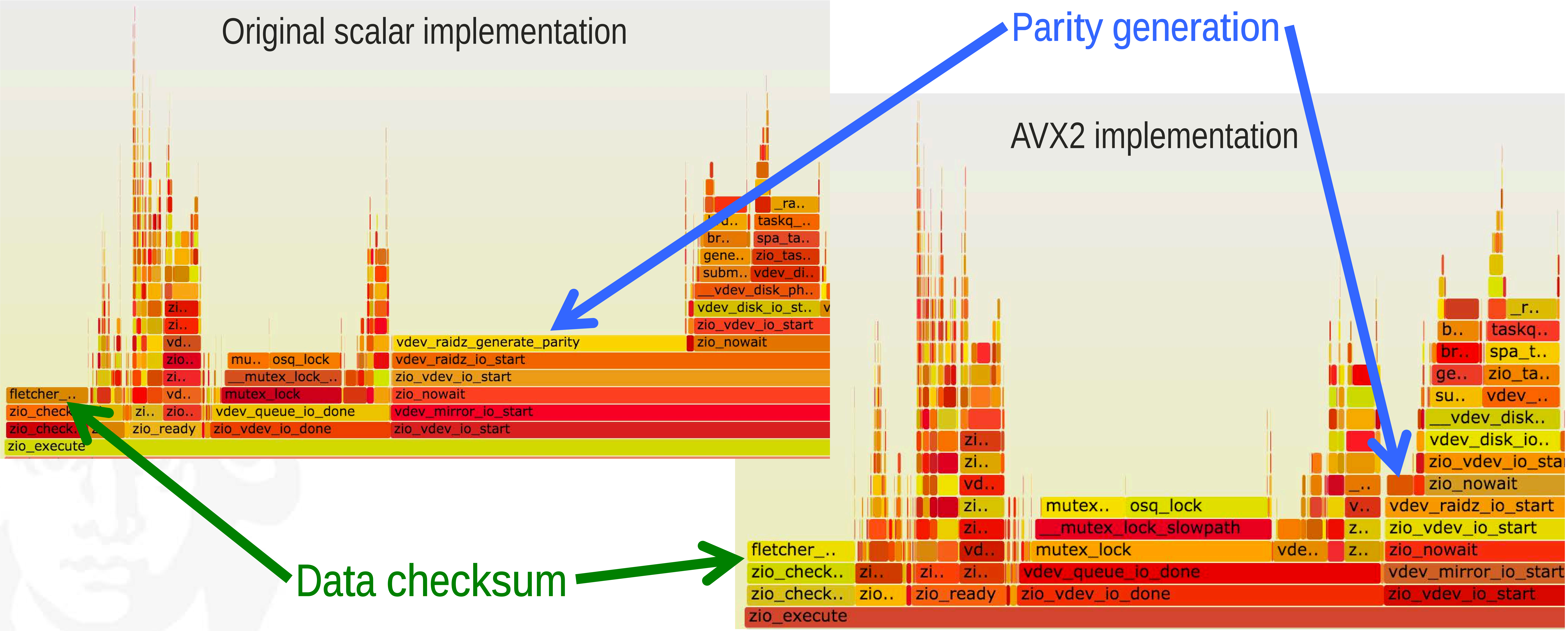
ZFS RAIDZ-3 Results

- RAIDZ-3
- 8 data disks
- 3 parity disks
- Generate PQR
- Reconstruct 3 disks



* All trademarks are the property of their respective owners.

Profiling with **perf** and FlameGraph[1)]



1) “Flame Graphs”, Brendan Gregg, http://www.brendangregg.com/flamegraphs.html

New RAIDZ Implementations speed-up

RAIDZ operation	Scalar	SSE	AVX2
P generate	2.2	2.4	2.6
P reconstruct	1.4	2.0	2.2
PQ generate	1.5	4.1	4.3
Q reconstruct	1.5	7.2	8.8
PQ reconstruct	1.2	4.7	7.1
PQR generate	1.4	5.6	8.8
R reconstruct	4.8	20.7	32.3
PR reconstruct	8.5	43.0	69.1
QR reconstruct	5.0	35.5	60.2
PQR reconstruct	5.9	50.1	85.8

Speed-up relative to the original RAIDZ methods

Summary & Future work

- Benefits of vectorized RAIDZ methods:
  - Faster parity generation
  - Faster recalculation of missing data
  - Shorter scrub and resilver times
  - Increased reliability
  - Decreased system acquiring and running costs

- Future work:
  - Test and verify the implementation[1)]
  - Upstream to *ZFS on Linux*
  - Linear scrub…

1) " A program can be made arbitrarily fast if you relax the requirement of correctness." - D.Knuth

The End

Thank you!

Questions?