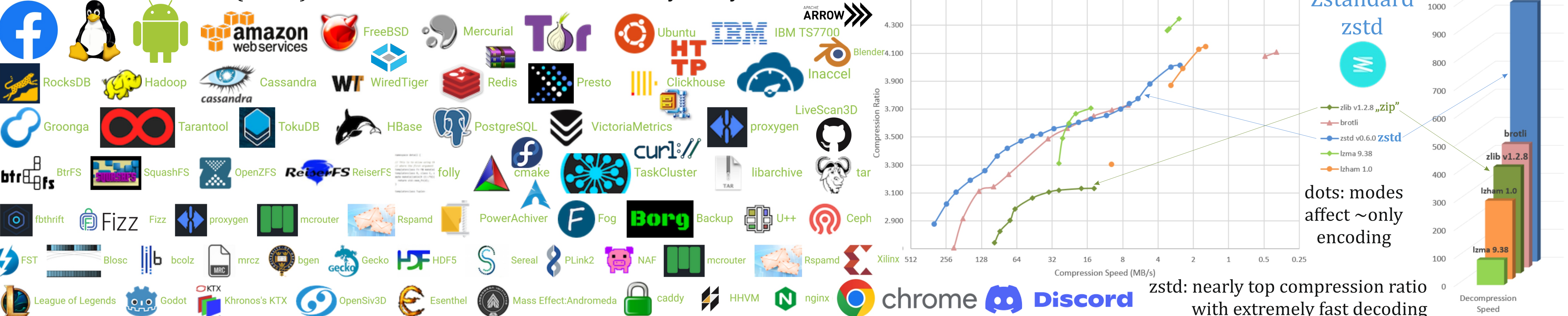


ASYMMETRIC NUMERAL SYSTEMS

Jagiellonian University methods **your data is written with** if using e.g.:

Apple: LZFS (tANS) compressor default since 2016 in iPhone, Mac  DNA compression - currently default is CRAM (rANS) in popular SAMtools

Facebook Zstandard (tANS) compressor, e.g. in Linux, Android kernel, ~3x faster decoding, ~5x encoding, much better compression than zlib/gzip: standardized for MIMO (email) and HTML as RFC 8478, now nearly everywhere:



And many more, e.g. (rANS) Google Draco 3D data compressor (of geometric meshes and point clouds), new coming every year JPEG XL (rANS) was standardized (ISO), mainly from Google, to replace 1992 JPEG "using 1/3 size", also improving old photos  

Nearly all **data** we use is **compressed**, reducing file size up to 1000 times (for video)

Data compression, after transformations and predictions, obtains **event/symbol stream**

This stream usually finally undergoes **entropy coding**: translating **symbols into bits**


We need n bits to choose from 2^n possibilities, what if we know percentage of '1' in 0/1 sequence?

$$\binom{n}{pn} \approx 2^{nh(p)} \quad \text{for} \quad h(p) = -p \lg p - (1-p) \lg (1-p) \quad \text{Shannon entropy}$$

Seen as weighted average: **symbol of probability p carries $-\lg p$ bits of information**

Huffman coding (1952) - assigns bit sequence to each symbol, optimal only for $p = \frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \dots$

Arithmetic/range coding (~1975) - nearly accurate probabilities, but costly/slow

Asymmetric Numeral Systems (Jarosław Duda, JU , uABS: 2006, tANS: 2007, rANS: 2013)

fast and accurate - ending the compromise, nearly default since 2014, **up to 30x speedup**:

Decoding speed/core of Intel i7	year 2013	year 2016
Approximate (Huffman)	~ 300 MB/s	~ 1000 MB/s
Accurate (arithmetic, ANS)	~ 50 MB/s	~ 1500 MB/s

NVIDIA rANS GPU: 100+ GB/s

Past: compromise

Now: ANS

or?

(prefix,) Huffman coding (also unary, Golomb, Elias, etc.)
fast (>300MB/s/core)
no multiplication, needs sorting
but **inaccurate**: $\Pr(s) \sim 2^{-r}$
e.g. for $\Pr(a)=0.01$, $\Pr(b)=0.99$ uses **1 bit/symbol**

arithmetic/range coding
slow (<< 100MB/s/core)
uses **multiplication**
uses nearly **accurate** $\Pr(s)$
e.g. for $\Pr(a)=0.01$, $\Pr(b)=0.99$ uses **~0.08 bits/symbol**

tANS: tabled - no multiplication
"Huffman generalized to **fractional bits**"
also allows for **simultaneous encryption**
mainly used for smaller models, fixed distributions
fast (> 500MB/s/core)
uses nearly **accurate** $\Pr(s)$
e.g. for $\Pr(a)=0.01$, $\Pr(b)=0.99$ uses **~0.08 bits/symbol**

rANS: range - direct replacement of arithmetic/range coding: with smaller state, less multiplications
mainly used for larger models, adaptive distributions

Coding with **accurate probabilities** requires **buffer handling fractional bits**

This buffer in **arithmetic coding** is **range - expensive** to work on

ANS advantage: store information in just a single natural number x

Assume **number $x \in \mathbb{N}$ contains $\lg(x)$ bits of information** (corresponds to $\Pr(x) \sim 1/x$ as in **Zipf law**)

We know **symbol s of probability p contains $\lg(1/p)$ bits**

So $C(s, x) := x'$ containing both has: $\lg(x') \approx \lg(x) + \lg(1/p)$ bits

in other words, **coding rule: $C(s, x) = x' \approx x/p$**

Like in standard binary system $x' = 2x + s \approx x/0.5$

optimal for $\Pr(s=0) = \Pr(s=1) = 0.5$

$x \rightarrow x$ -th appearance of even/odd, ANS: let's redefine even/odd split:

Example for binary system:

$$C(s, x) = 2x + s$$

"01111" $\rightarrow x = 47$

assuming 1/2-1/2 distribution

e.g. $x=1 \xrightarrow{s=0} 2 \xrightarrow{s=1} 5 \xrightarrow{s=1} 11 \xrightarrow{s=1} 23 \xrightarrow{s=1} 47$

some **asymmetric binary system** for $\Pr(0) = 1/4$, $\Pr(1) = 3/4$

redefine even/odd numbers - modify their densities:

$$x' \approx x/\Pr(s)$$

e.g. $x=1 \xrightarrow{s=0} 4 \xrightarrow{s=1} 6 \xrightarrow{s=1} 9 \xrightarrow{s=1} 13 \xrightarrow{s=1} 18$

for rANS (range ANS):

"01111" $\rightarrow x = 18$

more compressed thanks to better probability agreement:

"01111" is closer to 1/4 - 3/4

$x \rightarrow x$ -th appearance of 'even' ($s=0$) or 'odd' ($s=1$)

AI NVIDIA rANS variants: repeating division in ranges, e.g. of size 4:

$$\bar{s}(x) = 0 \text{ if } \text{mod}(x, 4) = 0, \quad \text{else } \bar{s}(x) = 1$$

to decode or encode 1, localize quadruple ($\lfloor x/4 \rfloor$ or $\lfloor x/3 \rfloor$)

if $\bar{s}(x) = 0$, $D(x) = (0, \lfloor x/4 \rfloor)$ else $D(x) = (1, 3\lfloor x/4 \rfloor + \text{mod}(x, 4) - 1)$

$$C(0, x) = 4x \quad C(1, x) = 4\lfloor x/3 \rfloor + 1 + \text{mod}(x, 3)$$

Analogously for large alphabet: rANS (range, 2013) is used as **simpler** direct replacement for arithmetic/range coding - **many times faster**

Handles **adaptive modification of probabilities**, is good for vectorization.

Now **symbol stream** \rightarrow **very large number x** , for long there is needed **renormalization**: ensure e.g. $x \in \{2^{16}, \dots, 2^{32} - 1\}$ by regularly transferring **accumulated (also fractional) bits** in buffer $x \leftrightarrow$ **bitstream**

ANS: "asymmetrize" numeral system for chosen probability distribution

uABS (uniform, 2006): direct generalization of binary system ($p = 1/2$)

'odd' numbers below x : $x_1 = \lfloor xp \rfloor$ remaining 'even': $x_0 = x - x_1$

encoding step $(s, x) \rightarrow x'$:

$$C(0, x) = \left\lfloor \frac{x+1}{1-p} \right\rfloor - 1 \quad p = \Pr(1) \quad D(x) = (s, x_s) \quad s = \lfloor (x+1)p \rfloor - \lfloor xp \rfloor$$
$$C(1, x) = \left\lfloor \frac{x}{p} \right\rfloor \quad D \circ C = id \quad C \circ D = id \quad x_1 = \lfloor xp \rfloor \quad x_0 = x - x_1$$

decoding step $x' \rightarrow (s, x)$:

tANS (tabled, 2007): put into table with renormalization, building **automaton**

example: binary alphabet $x \in \{4, 5, 6, 7\}$ states

$\Pr(a) > 1/2$ **a carries < 1 bit**

encoding:

$s=a$: $x=4$ ($p_4 \approx 0.321$), $x=5$ ($p_5 = 0.25$), $x=6$ ($p_6 \approx 0.241$), $x=7$ ($p_7 \approx 0.188$)

decoding:

$x \rightarrow s$, new x

$4 \rightarrow a$, $6+d_1$

$5 \rightarrow b$, $4+2d_2+d_1$

$6 \rightarrow a$, 4

$7 \rightarrow a$, 5

example: encoding \rightarrow decoding

s : bits $4 \rightarrow b \rightarrow 5 \rightarrow a \rightarrow 7 \rightarrow a \rightarrow 4 \rightarrow a \rightarrow 6 \rightarrow a \rightarrow 4 \rightarrow b \rightarrow 5 \rightarrow b \rightarrow 5$

example of tANS construction for $L=16$ states and size 3 alphabet

$t = \text{decodingTable}[x]$; use($t.symbol$); $x \rightarrow t.newX + \text{readBits}(t.nbBits)$;

- Approximate probabilities** as $p_s \approx L_s/L$
- Spread symbols:** L_s of symbol s (fast, step = 5)
- Enumerate appearances** from L_s to $2L_s - 1$
- Renormalize** to make x remain in $I = \{L, \dots, 2L-1\}$ range
- Encode/decode** - e.g. decoding 11100001101010011

$\{t = \text{decodingTable}[x]; \text{use}(t.symbol); x \rightarrow t.newX + \text{readBits}(t.nbBits); \}$

$x: 25 \xrightarrow{s=0} 11 \xrightarrow{s=1} 23 \xrightarrow{s=2} 30 \xrightarrow{s=0} 28 \xrightarrow{s=1} 18 \xrightarrow{s=0} 27 \xrightarrow{s=1} 24 \xrightarrow{s=2} 23 \xrightarrow{s=0} 29 \xrightarrow{s=1} 26 \xrightarrow{s=2} 16 \xrightarrow{s=0} 17 \xrightarrow{s=1} 19$