Data compression methods for on-board software updates for the Innocube Satellite

Julius-Maximilians-
UNIVERSITÄT WÜRZBURG

in partnership with
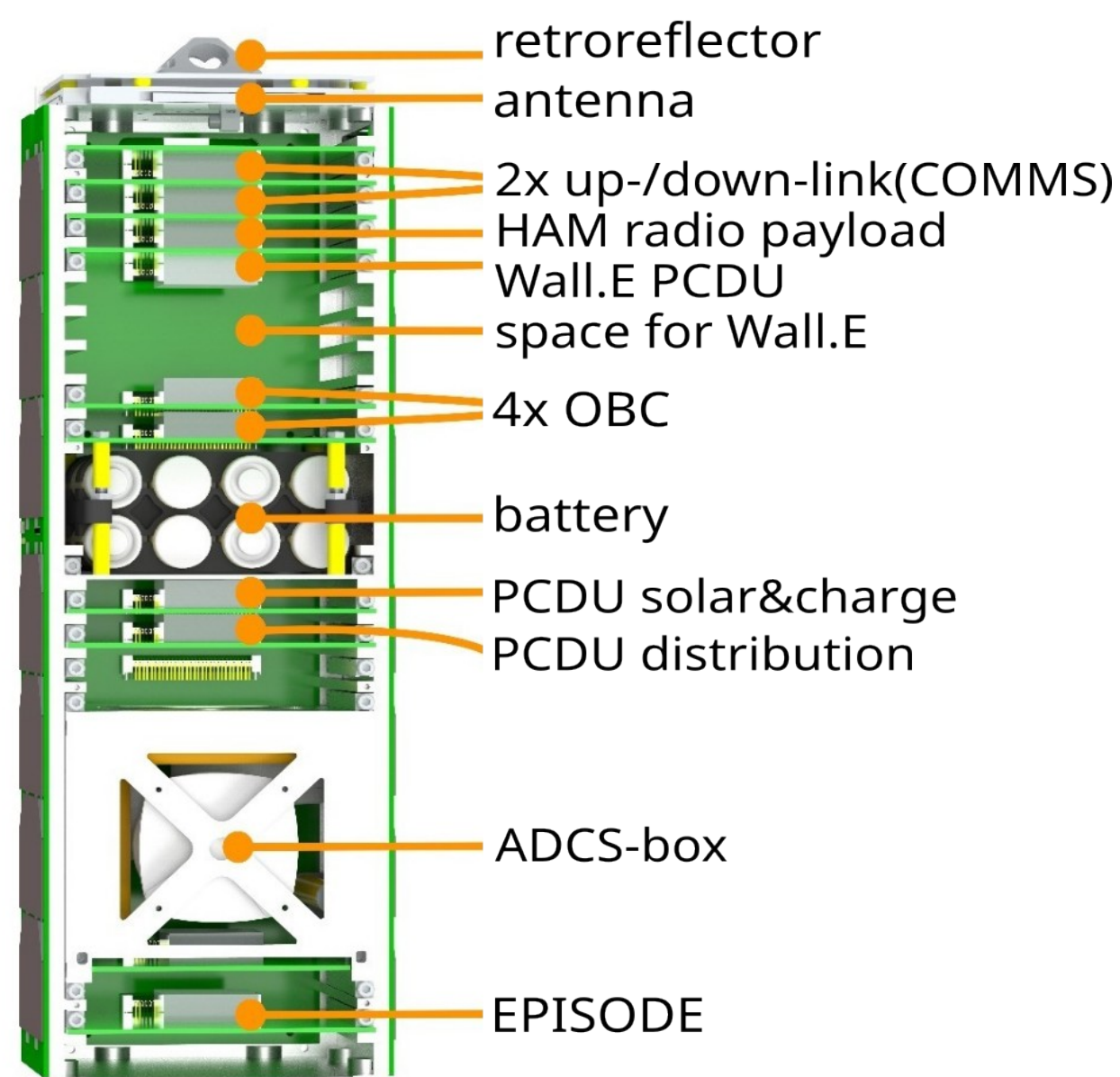TU berlin
Institut für Luft- und Raumfahrt
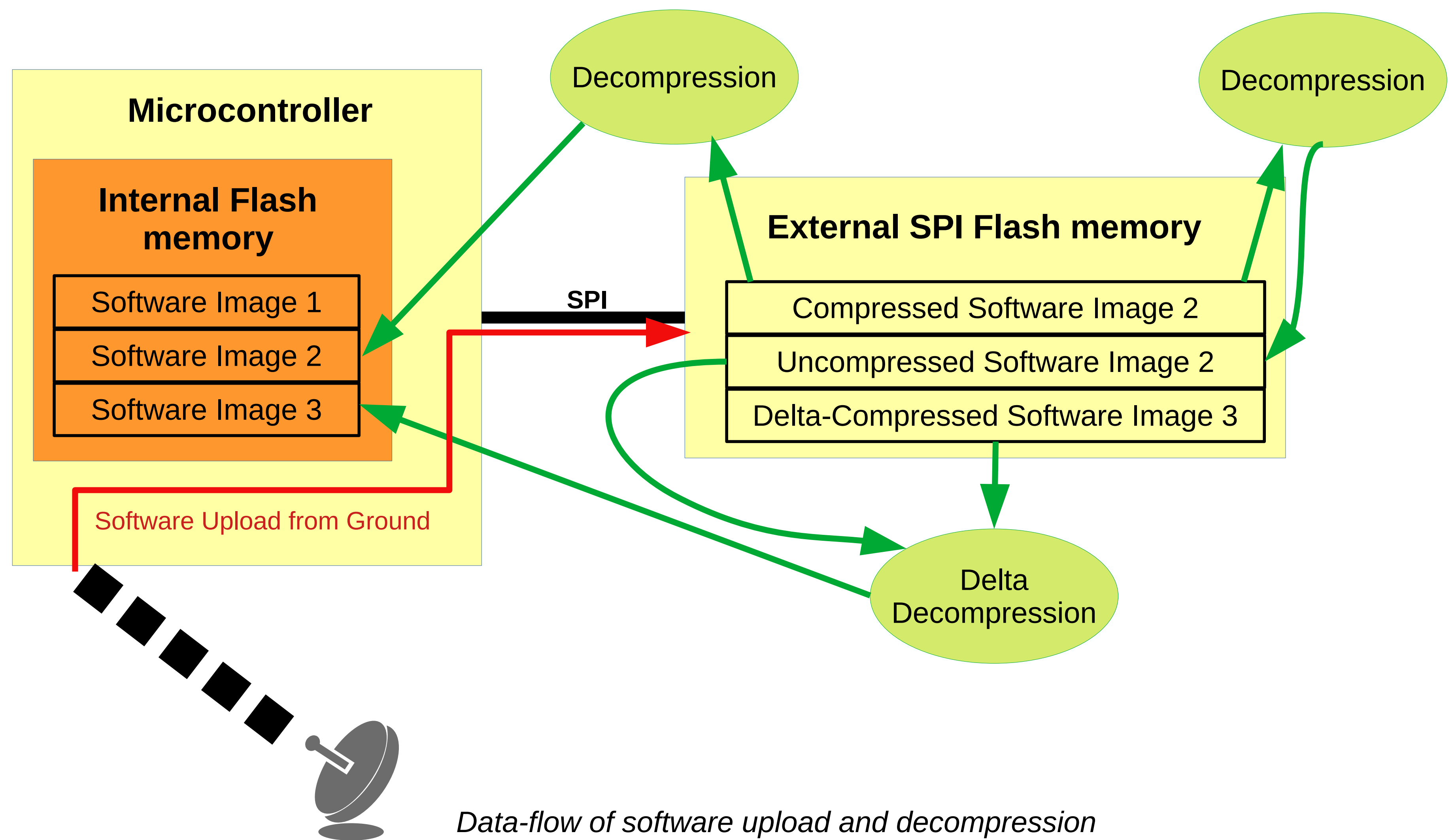ILR

The Innocube Mission

The main objective of InnoCube is to show the feasibility of the following three novel systems:
- **EPISODE**: SDR-GNSS for CubeSats
  (antenna, FPGA, SKITH pcb & software)
- **SKITH**: wireless satellite bus & protocol
- **WALL#E**: battery as supporting structure

Innocube consists of 7 different computing nodes which are connected by our novel wireless satellite bus. As having that many nodes presents a challenge in case of an in-orbit software update, we explored various methods for compression and data reduction in order to minimize the time required for software uploads.

retroreflector
antenna
2x up-/down-link(COMMS)
HAM radio payload
Wall.E PCDU
space for Wall.E
4x OBC
battery
PCDU solar&charge
PCDU distribution
ADCS-box
EPISODE

Innocube system overview

Microcontroller

Internal Flash memory
- Software Image 1
- Software Image 2
- Software Image 3

Decompression

SPI

External SPI Flash memory
- Compressed Software Image 2
- Uncompressed Software Image 2
- Delta-Compressed Software Image 3

Decompression

Delta Decompression

Software Upload from Ground

Data-flow of software upload and decompression

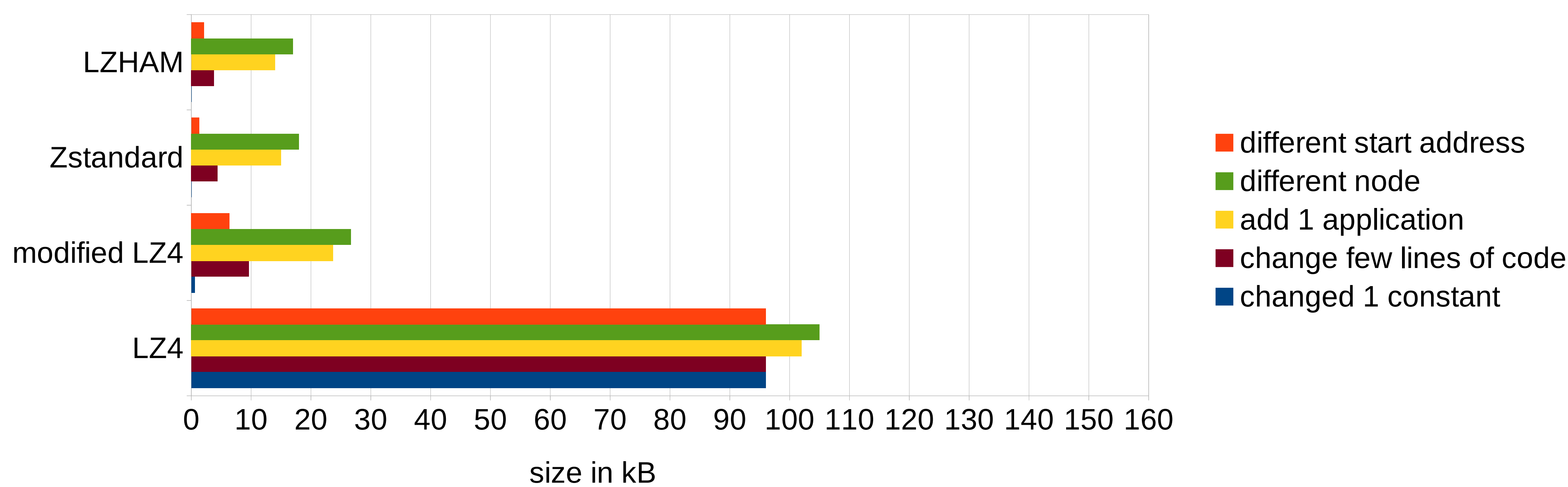## Compression Algorithm Considerations

- During on-board decompression we need to store decompressed data in external/internal flash memory
- Most implementations of compression algorithms expect to access already decompressed data in ram
- We need to modify an existing implementation or implement ourselves
- More advanced algorithms with better compression ratio are very complex
- LZ4 is very easy to implement, but has only a 64 Kb "look-back"-window, not enough for effective delta compression
- We created a slightly **modified LZ4** variant with a 1Mb "look-back"-window

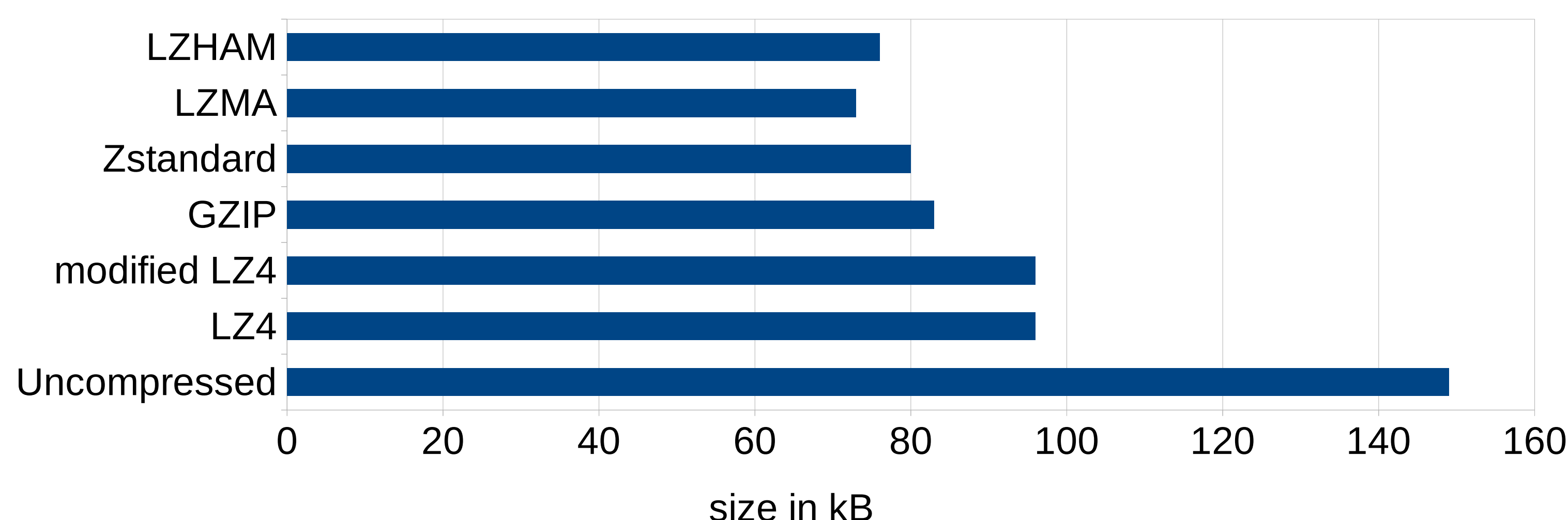|	Easy implementation	Supports delta compression
LZ4	yes	yes, not useful
modified LZ4	yes	yes
GZIP	no	no
Zstandard	no	yes
LZMA	no	no
LZHAM	no	yes

Delta Compression

- Our software images have a lot of shared code (OS, global Apps), so compressing a new image against an image that has been uploaded previously is very advantageous
- It compresses only the differences between the old and new data
- Compression algorithms that use dictionary coding have this ability naturally built in
- This works by virtually prepending the baseline data before the new data during compression/decompression

Delta compression of different new images against baseline image

- different start address
- different node
- add 1 application
- change few lines of code
- changed 1 constant

size in kB (0 10 20 30 40 50 60 70 80 90 100 110 120 130 140 150 160)

compression of baseline image with different algorithms

LZHAM, LZMA, Zstandard, GZIP, modified LZ4, LZ4, Uncompressed

size in kB (0 20 40 60 80 100 120 140 160)

Conclusions

- Using an easy compression algorithm like LZ4 already gives great benefits and should always be considered
- Depending on use case, delta compression gives more benefits than implementing a more complex algorithm
- Delta compression works great with images that just differ in memory start address. This eliminates the need to create position independent software images

Contact:
Prof. Dr.-Ing. Sergio Montenegro:
    sergio.montenegro@uni-wuerzburg.de
Erik Dilger: erik.dilger@uni-wuerzburg.de