Parallel implementation of the FAPEC data compressor

Sergio Garcia-Almoril (IEEC/UPC)
Enrique García-Berro (IEEC/UPC)
Jordi Portell (IEEC/DAPCOM/UB)

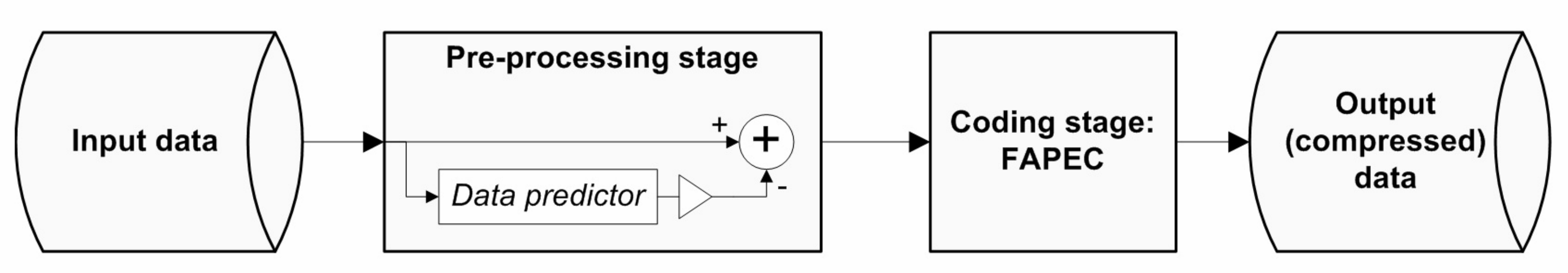
Presented by Jordi Portell

Data Compression in space

- Main Necessity
  - Large diversity of data must be compressed
  - Wish: General-purpose lossless coder offering the highest possible compression ratio in minimal execution time
- Limitations:
  - On-board processing power
    → low complexity
  - Communications channel reliability
    → small independent blocks
- Solution:
  - Fully Adaptive Prediction Error Coder (FAPEC)

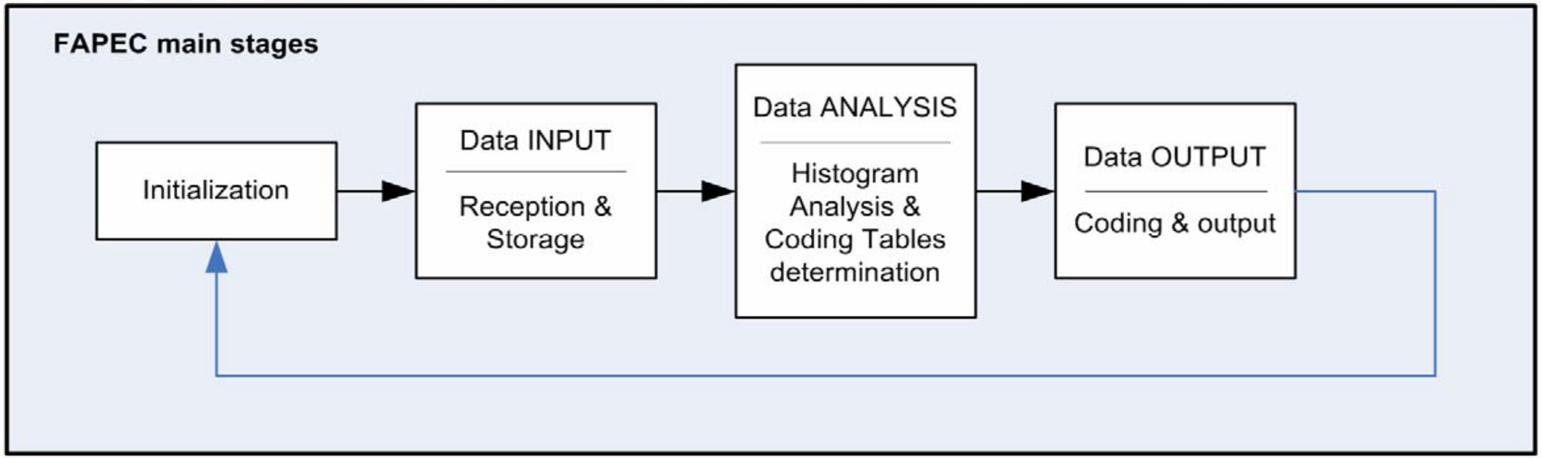
FAPEC data compressor

- FAPEC: Fully Adaptive Prediction Error Coder
- Quick entropy coding with excellent coding efficiency
- Adequate for data following unusual statistical distributions and/or affected by outliers
- Better resiliency to such cases than CCSDS 121.0
- Very good for large sample sizes



FAPEC compression

- Specifications
  - 4 functional stages
  - Some compression options available
  - Independent modules design
- System limitations:
  - Compression of large file sizes
  - Specify decompression options used in compression
  - Limited processing speed (yet quite good)



Parallel strategy

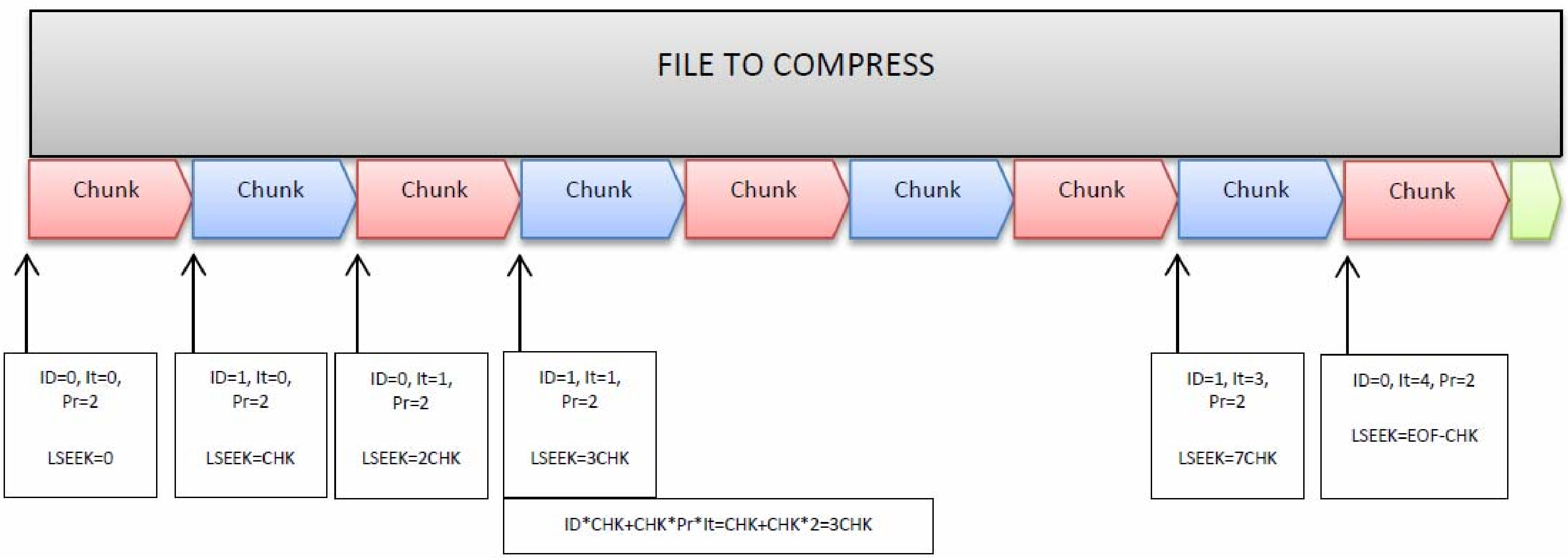
- Parallelism is becoming available in space
- Coarse-Grained Parallelism
  - Multiple division of entire file
  - Low inter-process communication
  - Independent compression sub-task
- Chunk-compression concept
  - Fixed chunk length
  - Independent & automatic generation of chunk
  - Inter-process synchronism
- Message
  - ID
  - Number of iterations
  - FAPEC configuration & coding options

Parallel strategy

- Header
  - Encapsulation of parallel configuration (chunk size)
  - Encapsulation of FAPEC configuration & options
  - Chunk header
- System synchronism
  - Shared queue
  - Shared semaphore
  - Mutual exclusion
- System distribution
  - Parallel execution
  - Fixed number of threads
  - Chunk-process distribution
  - I/O optimization

Parallel implementation

- INPUT stage
  - Concurrent access to file
  - Asynchronous chunk generation & compression (re-sync at the end, or once every N chunks)



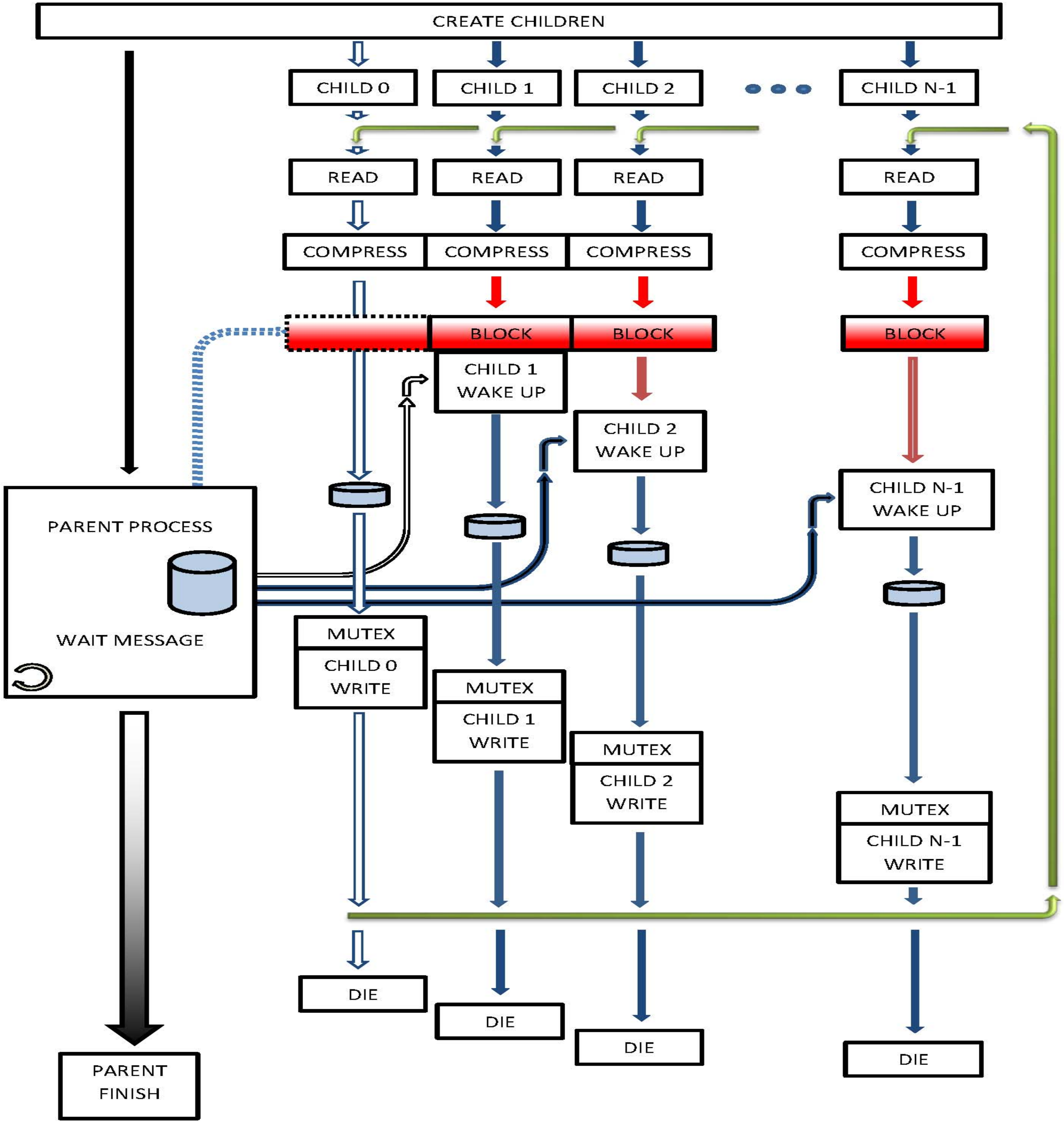
Parallel implementation

- OUTPUT stage problems
  - Different compressed chunk size
  - Writing without overwrite

- Solution
  - Blocking process when compression ends
  - ORDERLY wake up
  - Mutual exclusion included
  - Writing APPEND mode

Parallel-FAPEC compression

- Asynchronous chunk acquisition & compression data
  - Chunk pre-proc + compression with FAPEC
  - High speed performance
- Fully synchronous inter-process communication
  - Parent-children communication in every chunk
  - Control error and manage system execution
  - Non overwrite guaranteed
- Flowchart

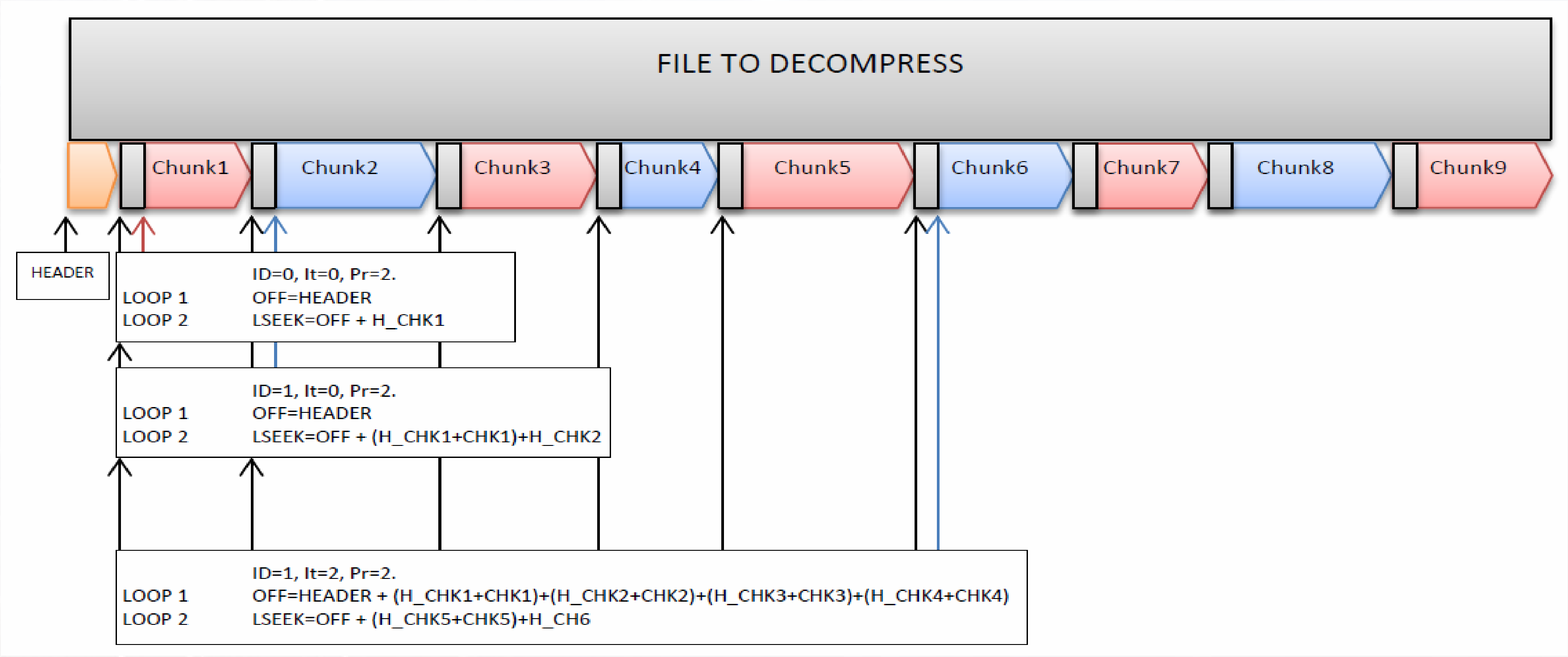

CREATE CHILDREN
CHILD 0
CHILD 1
CHILD 2
CHILD N-1
READ
READ
READ
READ
COMPRESS
COMPRESS
COMPRESS
COMPRESS
BLOCK
BLOCK
BLOCK
CHILD 1 WAKE UP
CHILD 2 WAKE UP
CHILD N-1 WAKE UP
PARENT PROCESS
WAIT MESSAGE
MUTEX
CHILD 0 WRITE
MUTEX
CHILD 1 WRITE
MUTEX
CHILD 2 WRITE
MUTEX
CHILD N-1 WRITE
DIE
DIE
DIE
DIE
PARENT FINISH

Parallel decompression

- HEADER
  - Extracting FAPEC options & configuration
  - Error control information
  - Parallel options
- INPUT stage
  - Concurrent access to file
  - Asynchronous chunk generation & decompression
- OUTPUT stage
  - Similar to decompression
  - Blocking, orderly wake up, mutual exclusion, APPEND mode

Decoder implementation

- INPUT stage
  - Header chunk calculation

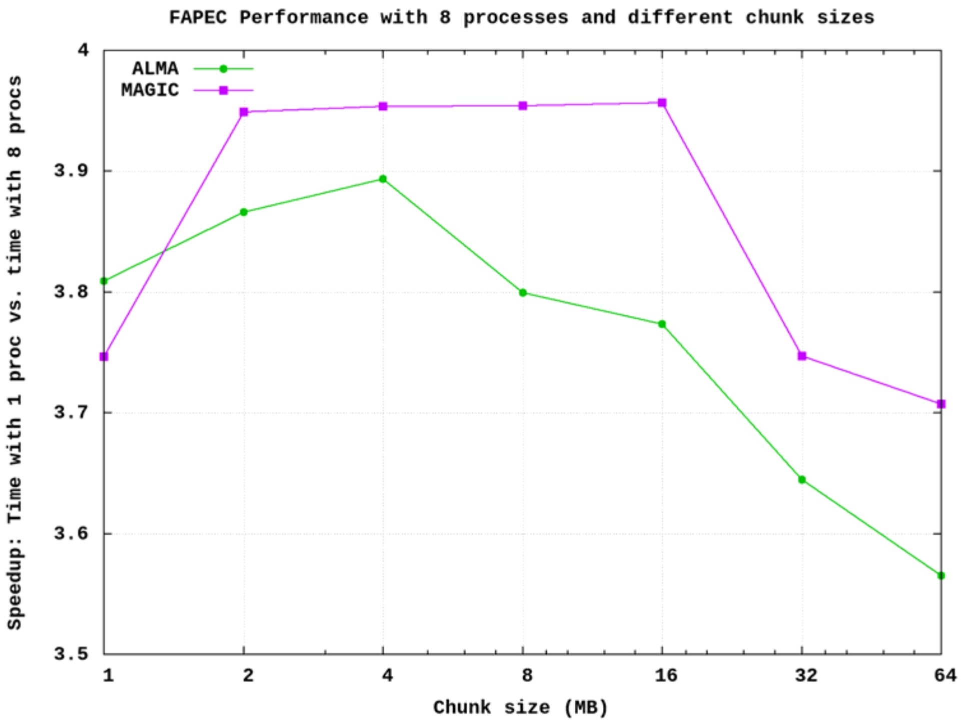


Test results: compression throughput

- Some example performances:
  (virtualized Linux in a mid-range 2-core laptop)

Image	Size	121.0	FAPEC	FAPEC 2x	FAPEC 4x
banyoles-1024x600.raw (Ratio 1.3)	**0.6 MB**	**14.1 MB/s**	**20.0 MB/s**	**24.2 MB/s**	**27.8 MB/s**
pyrenees.raw (Ratio 1.4)	**5.3 MB**	**18.3 MB/s**	**33.1 MB/s**	**34.8 MB/s**	**43.4 MB/s**
pleiades_port_de_bouc.raw (Ratio 0.9)	**15.4 MB**	**17.0 MB/s**	**34.1 MB/s**	**34.4 MB/s**	**45.9 MB/s**
GOES-NOAA-1400x5504 (Ratio 1.4)	**32.4 MB**	**19.4 MB/s**	**39.3 MB/s**	**40.7 MB/s**	**51.2 MB/s**

Test results: Scalability

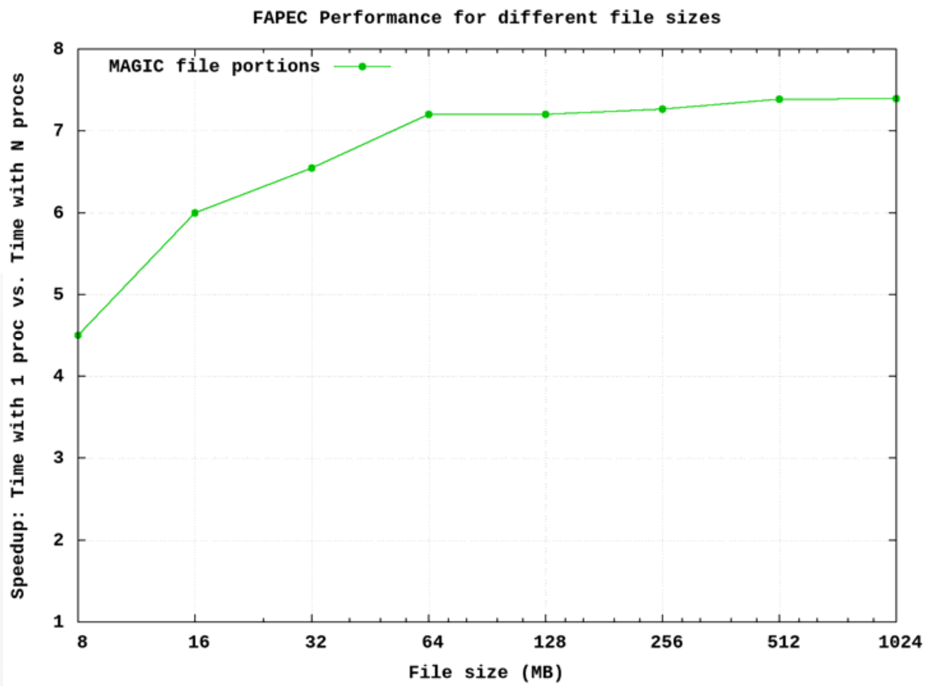


ALMA: 256 MB

MAGIC: 1,9 GB

4-core computer

Execution time vs. chunk size



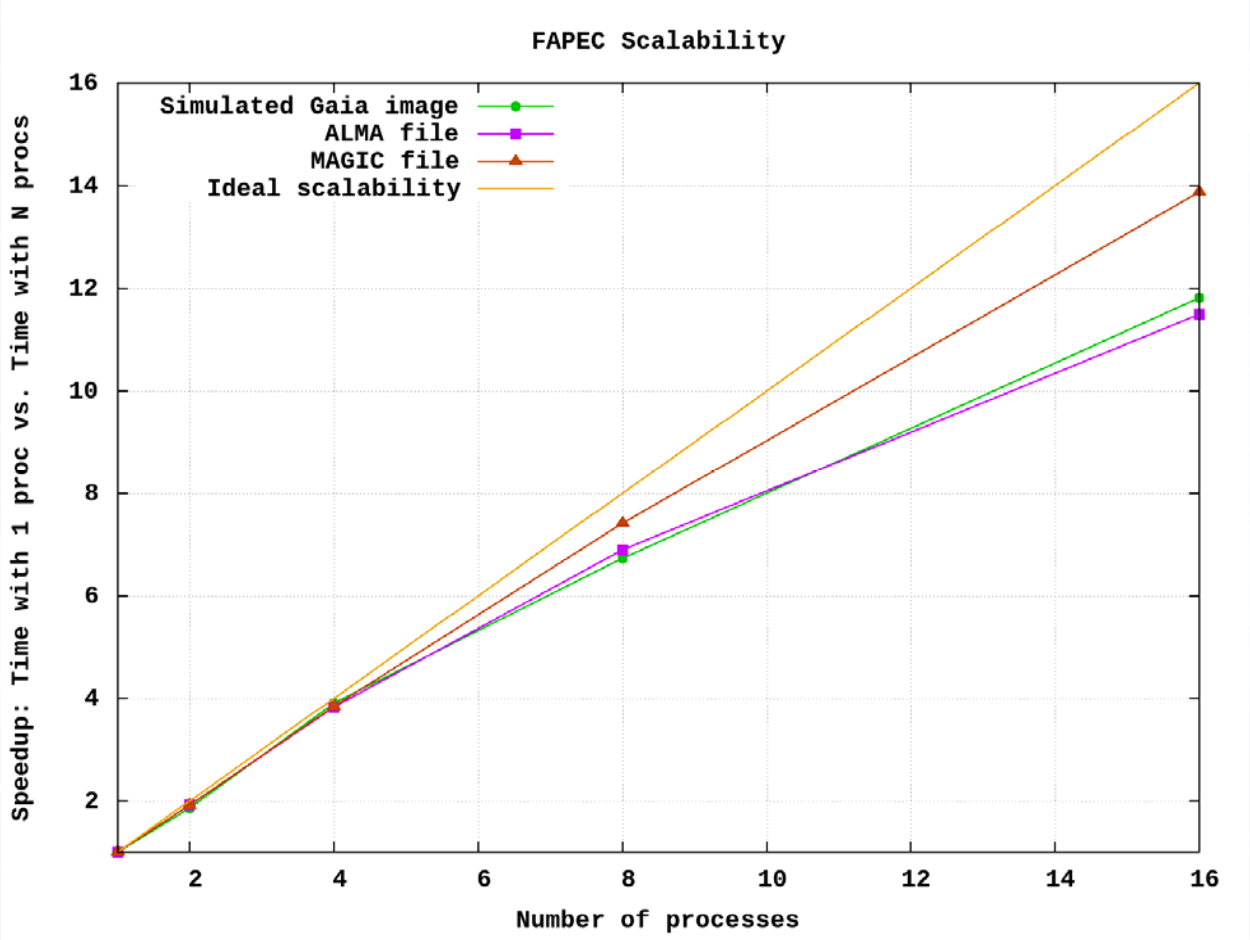
MAGIC

8-core computer

Execution time vs. file size

Test results: Scalability

- Execution time vs. Processors (BSC/MareNostrum 16-core node)



Gaia: 77MB
1MB chunk
2.68 ratio

ALMA: 256 MB
4MB chunk
0.99 ratio

MAGIC: 1.9GB
8MB chunk
2.43 ratio

Peak performance: **610 MB/s** on 16 cores

Conclusions and forthcoming work

- Time execution and scalability:
  - Up to 14x scalability with 16 threads
  - 7.4x and 3.9x with 8 and 4 threads
  - Best efficiency with larger data files
  - Same ratios than FAPEC (improving CCSDS 121.0)
  - Chunk-based operation
  - Embedded decompression information
- Very promising results
  - Massive data (de)compression
  - Feasibility of parallel operation of FAPEC
    - Same principle applicable to the FPGA implementation
- Future work:
  - Improve decompressor (random chunk access)
  - Use improved parallel API/standard
  - Study migration to FPGA

Parallel implementation of the FAPEC data compressor

Sergio Garcia-Almoril (IEEC/UPC)
Enrique García-Berro (IEEC/UPC)
Jordi Portell (IEEC/DAPCOM/UB) – ***jordi.portell@dapcom.es***

Presented by Jordi Portell