



---

## Crusher Simulator: simulatore del framework CRUSHER per la generazioni di algoritmi

### Studenti:

Egidio Giacoia  
Raffaele D'Arco  
Sabato De Gregorio

---

**Professore:** Bruno Carpentieri

**Anno Accademico:** 2018-2019

# Contesto

Le **applicazioni scientifiche** producono, memorizzano e trasferiscono grandi mole di dati in **virgola mobile**

La compressione gioca un ruolo fondamentale in questo contesto per ridurre la quantità di dati

Molti utenti utilizzano framework come **HDF5** per la gestione dei dati:

- selezione di **filtri di compressione** da applicare su un insieme di dati



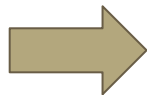
# Problema

La selezione del filtro di compressione da applicare avviene in base:

- alla **precisione** (singola o doppia) dei dati in virgola mobile
- devono essere **lossless** per non avere perdita di informazione

# Soluzione

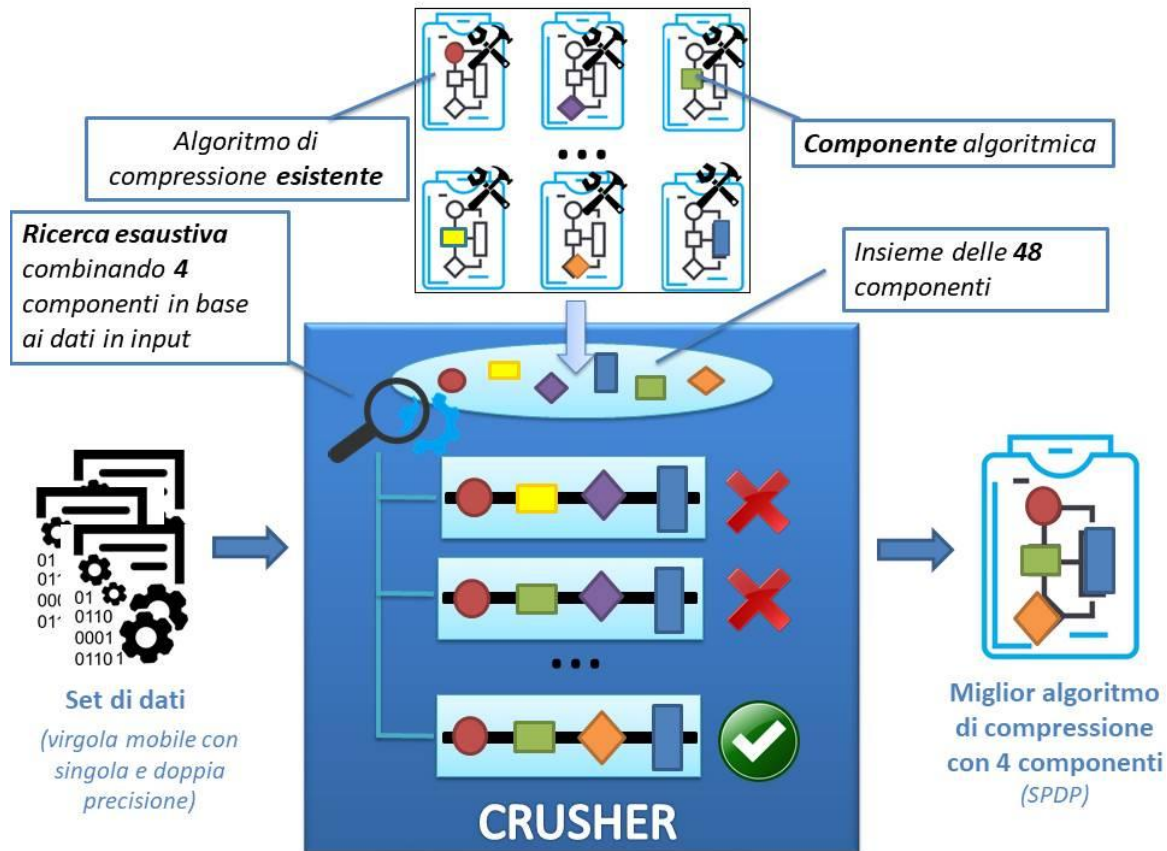
Un **unico** algoritmo che comprime in modo lossless e ottimale i dati in virgola mobile, **indipendentemente** dalla **precisione** utilizzata



## SPDP

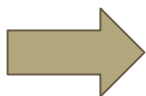
Single **P**recision  
Double **P**recision

# Framework CRUSHER



# Obiettivi del lavoro svolto

- ★ Simulare il framework CRUSHER



**Crusher Simulator**

- ★ Aggiunte altre componenti per tentare di aumentare il rapporto di compressione
- ★ Test e confronto con l'algoritmo SPDP



# Indice

## → Crusher Simulator

- ◆ Componenti utilizzate
- ◆ Modalità di interazione
- ◆ Reporting Structure
- ◆ Demo

## → Metodologie

- ◆ Datasets e HW utilizzato
- ◆ Metodi di misurazione delle performance

## → Risultati Sperimentali

- ◆ Algoritmi ottenuti
- ◆ Confronto con SPDP

## → Sviluppi Futuri

---

---

# Crusher Simulator

— Componenti utilizzate, modalità di  
interazione, Reporting Structure e  
Demo —

---







---





# Crusher Simulator - Layout

La cartella principale è `Crusher_Simulator` in cui all'interno troviamo:

 <code>test_file</code>	11/06/2019 15:05	Cartella di file	
 <code>components</code>	11/06/2019 16:47	Cartella di file	
 <code>tmp</code>	11/06/2019 17:39	Cartella di file	
 <code>CrusherSimulator.py</code>	11/06/2019 16:56	Python File	25 KB
 <code>file_compresso.crusher</code>	11/06/2019 17:40	File CRUSHER	1 KB
 <code>report.log</code>	11/06/2019 17:40	Documento di testo	17 KB

Il contenuto di `tmp` e `report.log` viene **sovrascritto** ad ogni esecuzione del tool

# Componenti Algoritmiche

Crusher Simulator ha a disposizione **7** componenti:

- **4 shifters** - non modificano la dimensione dei blocchi di dati
  - ROTATE, BWT, DIM [SPDP], LNVs [SPDP]
- **3 riduttori** - comprimono la lunghezza di un blocco di dati
  - RLE, HUFFMAN, LZ [SPDP]

Ciascun componente prende in input una sequenza di valori (una matrice), la trasforma e dà in output la sequenza trasformata

Ogni componente include un componente inverso corrispondente che esegue la trasformazione inversa dei dati

# Componenti Algoritmiche - Spazio di ricerca

Per facilitare la scelta del migliore algoritmo:

- le catene di componenti possono essere composte da un numero variabile di componenti, ma non si ripetono nella catena

$\#\_shifters!$

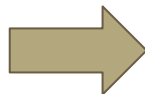
$(\#\_shifters - \#\_of\_components\_for\_chain - 1)!$

•  $\#\_compressors = \#\_combinations$

$\#\_shifters = 4$

$\#\_compressors = 3$

$\#\_of\_components\_for\_chain = 4$



**72** possibili algoritmi a 4  
componenti

# Modalità di interazione

Crusher Simulator ha **3** modalità di interazione:

- **Encode Mode**

Effettua la **compressione lossless** di un file dato in input utilizzando le componenti disponibili

- 1) **Ricerca** esaustiva andando a concatenare un certo # di componenti
- 2) **Determina** il miglior algoritmo tra quelli candidati (miglior compression ratio)
- 3) Output il file compresso (formato **.cruscher**)
- 4) Generazione file **report.log**

# Modalità di interazione

Crusher Simulator ha **3** modalità di interazione:

- **Decode Mode**

Effettua la **decompressione** di un file dato in input (formato **.cruscher**)

- 1) Lettura dal <<footer>> del file compresso l'ordine delle componenti
- 2) Applicazione ordine inverso delle componenti
- 3) Output il file decompresso
- 4) Generazione file **report.log**

# Modalità di interazione

Crusher Simulator ha **3** modalità di interazione:

- **Verify Mode**

Verifica se due file dati in input hanno la stessa firma hash

- la funzione di hashing utilizzata è **MD5**

# Reporting Structure

Crusher Simulator al termine delle modalità di encode mode o decode mode) genera un file **report.log** che contiene:

- parametri sul relativo processo per le varie combinazioni testate (encode mode)
  - Compression Ratio
  - Space Savings
  - Time Execution



```
21 #####
22 #####
23 #####
24 #####
25 #####
26 #####
27 #####
28 #####
29 #####
30 #####
31 #####
32 #####
33 #####
34 #####
35 #####
36 #####
37 #####
38 #####
39 #####
40 #####
41 [!] Trying: <<ROTATE - DIM - BWT - RLE>> (1/72)
42 Compressed File Size: 869184 - 848.81 KB
43 Compression Ratio: 2.01447886754
44 Space Savings: 0.644694436496
45 Time Execution (seconds): 0.22936201096
46 #####
47 [!] Trying: <<ROTATE - DIM - BWT - HUFFMAN>> (2/72)
48 Compressed File Size: 1017461 - 993.6 KB
49 Compression Ratio: 2.40494183071
50 Space Savings: 0.584085762171
51 Time Execution (seconds): 0.21213507662
52 #####
53 [!] Trying: <<ROTATE - DIM - BWT - LZ>> (3/72)
54 Compressed File Size: 1222514 - 1.17 MB
55 Compression Ratio: 2.00104047988
56 Space Savings: 0.500259984466
57 Time Execution (seconds): 0.15405998693
58 #####
```

# Reporting Structure

**Crusher Simulator** al termine delle modalità di encode mode o decode mode genera un file **report.log** che contiene:

- parametri sul relativo processo sulla miglior combinazione risultante
  - Input & Output file size
  - Compression Ratio
  - Space Savings
  - Time Execution
  - Numero di componenti usate
  - Numero di combinazioni effettuate
  - Miglior combinazione trovata

```
472 -----
473 Input_File_Name      = ./test_file/num_plasma.trace.gz - [e21c99705679dbff23fbc95f4af0e583 (md5)]
474 Compressed_File_Name = num_plasma_compr.crusher - [189ef36accb7f5120cb15f5f7bd23c2f (md5)]
475 Input_File_Size     = 2446300 - 2.33 MB
476 Compressed_File_Size = 249304 - 243.46 KB
477 Compression_Ratio    = 9.81251805025
478 Space_Savings        = 0.898089359441
479 Time_Execution(seconds) = 0.0737779140472
480 #_of_Components_Chain = 4
481 #_of_Combinations_Test = 72
482 Best_Combination      = DIM-ROTATE-LNVS-LZ
483 -----
484 See report.log for a complete view.
485 All files in tmp and report will be deleted each time the program is run
```





# Crusher Simulator in Action

**N.B.** La fase di compressione è estremamente onerosa in termini di tempo di esecuzione e risorse di sistema (decine di GB per file di input di dimensioni circa 150 MB)

---

---

# Metodologia

— Datasets, HW utilizzato, metodi di  
misurazione delle performance —

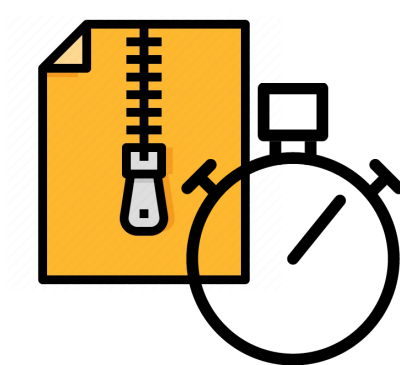
---

---

# Misurazioni

Le misure prese in considerazione sono:

- **rapporto di compressione:** dato dalla divisione tra la size del file non compresso e la taglia del file compresso
- **tempo di esecuzione**



# Datasets

Sono stati utilizzati **11 dataset** FPC a **doppia precisione**

I dataset rappresentano:

- Misurazioni da strumenti scientifici (obs)
- Simulazioni numeriche (num)
- Messaggi numerici (msg)



# Hardware utilizzato

- I test sono stati condotti su una macchina **AWS** con le seguenti caratteristiche
  - 1 x CPU Intel Xeon E5-2676 v3
  - 2 core @2.4GHz, formati nel seguente modo:
    - Cache L1 da 32Kb separate
    - Cache L2 da 256Kb unificata
    - Cache L3 da 30MB
- Memoria host da 8GB
- OS utilizzato: Ubuntu 18.04



---

# Risultati Sperimentali

— Analisi della struttura degli algoritmi —  
ottenuti e confronto con SPDP

---

# CrusherSimulator - Rapporto di compressione

FILE	TAGLIA INIZIALE	TAGLIA COMPRESSA	COMPRESSION RATIO	SPDP (lvl9) COMPRESSION RATIO	COMPONENTI
obs_error	56.28(MB)	36.99(MB)	1.60	1.61	DIM-LNVS-ROTATE-LZ
<u>obs_info</u>	18.05 (MB)	9.29 (MB)	1.94	1.95	ROTATE-DIM- <u>LNVS</u> -LZ
<u>obs_spitzer</u>	189 (MB)	174.45(MB)	1.08	0.98	DIM-ROTATE-BWT-HUFFMAN
<u>obs_temp</u>	38.08 (MB)	36.65 (MB)	1.04	1.03	ROTATE-DIM- <u>LNVS</u> -LZ
<u>num_plasma</u>	33.46 (MB)	982.48 (KB)	34.878	33.17	<u>LNVS</u> -ROTATE-DIM-LZ
num_comet	102.38 (MB)	86.42 (MB)	1.18	1.16	ROTATE-DIM-BWT-LZ
<u>num_brain</u>	135.27 (MB)	114.21 (MB)	1.18	1.20	DIM-ROTATE- <u>LNVS</u> -LZ
num_control	152.12 (MB)	144.72 (MB)	1.05	1.01	DIM-BWT-ROTATE-HUFFMAN
msg_bt	197.57(MB)	196.44(MB)	1.00	1.33	BWT-LNVS-ROTATE-HUFFMAN
msg_lu	185.13(MB)	149.32(MB)	1.23	1.26	DIM-ROTATE- <u>LNVS</u> -LZ
msg_sweep3d	119.91(MB)	42.22(MB)	2.84	3.01	DIM-ROTATE- <u>LNVS</u> -LZ

# CrusherSimulator - Tempo di esecuzione

Dataset	Tempo di esecuzione (in secondi)
obs_error	0,62
<u>obs_info</u>	0.19
<u>obs_spitzer</u>	31,36
<u>obs_temp</u>	0,53
<u>num_plasma</u>	0,32
num_comet	131,85
<u>num_brain</u>	2,09
num_control	30,90
msg_bt	30,49
msg_lu	2,40
msg_sweep3d	1,20



---

---

# Sviluppi Futuri

— Migliorie e nuovi campi —

---

---

# Sviluppi futuri

- Approccio su altri **domini** (file di immagini per esempio)
- Aggiungere nuove combinazioni al tool **Crusher Simulator** in modo da **ripetere** l'uso di una **componente** e vedere se si hanno miglioramenti nel rapporto di compressione
- Studiare altri compressori, come Zstd, ed **estrarne i componenti** chiave in modo da poter sintetizzare algoritmi ancora migliori

[illegible]